



RedHawk

User Manual

Software Release 18.1

Manual Version: Production

Copyright © October 30, 2017

ANSYS, Inc.

Copyright Notice and Proprietary Information

No part of this document may be reproduced or transmitted in any form or by any means, electronic, or mechanical, for any purpose, without the express written permission of ANSYS INC (Apache Design Business Unit). This manual and the program described in it are owned by ANSYS INC and may be used only as authorized in the license agreement controlling such use, and may not be copied except in accordance with the terms of this agreement.

Copyright © 2013, 2014, 2015, 2016; ANSYS INC (Apache Design Business Unit). All rights reserved.

Disclaimer

ANSYS INC makes no warranty of any kind, expressed or implied, with respect to software or documentation, its quality, or performance. The information in this document is subject to change without notice and does not represent a commitment on the part of ANSYS INC.

Trademarks

ANSYS and any and all ANSYS INC brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, INC. or its subsidiaries in the United States or other countries. All other trademarks referred to are the property of their registered owners.

ANSYS, Inc.

2645 Zanker Road

San Jose, CA 95134

Tel:(408) 457-2000

Fax: (408) 428-9569

apache_sales@apache-da.com

support_center@apache-da.com

www.apache-da.com

TABLE OF CONTENTS

Chapter 1 - Introduction

Full-chip Static and Dynamic Power Integrity	1-1
Using RedHawk in the Design Flow	1-2
Design Planning	1-2
Design Development	1-2
Design Verification	1-3
Summary	1-4

Chapter 2 - RedHawk Flow

Introduction	2-5
Static Voltage Drop Analysis Flow	2-5
Dynamic Voltage Drop Analysis Flow	2-6

Chapter 3 - User Interface and Data Preparation

Introduction	3-9
TCL Command User Interface	3-9
TCL Command Summary	3-9
Graphical User Interface	3-10
Elements of the GUI	3-10
Menu Bar	3-11
Primary Display Area	3-12
Log Display Area	3-12
Control Buttons	3-12
Design View Area	3-13
TCL Command Line	3-13
Using the GUI	3-13
Exporting and Importing a GUI Configuration	3-13
RedHawk Data Preparation - Static and Dynamic Analysis	3-13
RedHawk Program Files	3-13
Multiple Vdd/Vss Analysis	3-16
Summary of Differences in Input Data Files	3-17
Liberty Library Syntax Differences for Multiple Vdd/Vss Domains	3-18
P/G Arc Definitions in Custom LIB Files	3-20
GSR Keywords for Multi-Vdd Domain Designs	3-22
APL Requirements for Multi-Vdd Designs	3-22
AVM Configuration File	3-23
Data Prep and Using the Automated 'rh_setup.pl' Script	3-23
Recommended RedHawk Directory Structure	3-23
Automated Script rh_setup.pl	3-24
GDS2DEF Setup with the 'gds_setup.pl' Script	3-27
Using the gds_setup Utility	3-27
Outputs	3-29
Manually Importing Design Data	3-29

Command Line Data Import	3-29
GUI Data Loading	3-31
RedHawk Configuration Files	3-31

Chapter 4 - Power Calculation, Static IR Drop and EM Analysis

Introduction	4-35
Power Calculation	4-36
Setup for Vectorless Power Calculation	4-37
Setting Frequency and Clock Parameters	4-38
Setting the Switching State of Instances and Blocks	4-39
Setting the Toggle Rate	4-39
Setting Toggle Rate Scaling Parameters	4-41
Using both clock and signal toggle rates in power calculation	4-43
Setting Extraction Parameters	4-43
Selecting Power Calculation Methodology	4-44
Specifying Supply Nets	4-45
Setting Bus and Hierarchy Delimiter Keywords	4-46
Setting up for Event-Driven (VCD File) Power Calculation	4-46
Setting GSR Keywords for Event-driven (VCD) Power Calculation	4-47
Power Calculation Procedure and Results Evaluation	4-50
Power Grid Resistance Extraction	4-53
Metal Density Calculation	4-53
TCL Command R Extraction	4-53
GUI Extraction	4-54
Examining Power/Ground Grid Weakness	4-54
Defining Pad and Package Parameters	4-57
Command Line Procedure for Package Modeling	4-57
GUI Procedure for Package Modeling	4-58
Package Compiler Utility	4-58
Running RedHawk-S (Static IR/EM Analysis)	4-58
Exporting and Importing Results to the Design Database	4-60
Exporting a Database	4-60
Database Compatibility	4-60
Importing a Database	4-60
Flexible Memory Caching for Database Reloading	4-61
Early Analysis Methodology	4-61
Overview	4-61
Input Data Required	4-62
Early Analysis Flow	4-62
Block Power and Current Assignment	4-62
Creating Decap Cells During BPA	4-68
Early Stage Decap Estimation	4-69
Reports Created	4-70
Example Analyses	4-70
Evaluating Results of Static IR Voltage Drop Analysis	4-72
Example Procedure to Fix IR Drop Problems	4-75

Example IR Drop Case	4-75
Modifying Power Pads	4-76
Adding Metal6 Straps	4-78
Resistivity Sensitivity	4-80

Chapter 5 - Dynamic Voltage Drop Analysis

Introduction	5-81
Preparing for Dynamic Voltage Drop Analysis	5-81
Perform Cell Characterization	5-81
Calculate Power	5-81
Network Extraction	5-82
Pad and Package Parameter Setup	5-82
Methods of Dynamic Voltage Drop Analysis	5-82
Vectorless Dynamic Analysis	5-84
Overview	5-84
Input Data and Assumptions	5-84
Standard Vectorless Analysis Procedure	5-84
Unified Clock Gate Handling and Analysis	5-85
Automatic Clock Gate Handling (default)	5-85
RTL VCD-Driven Vectorless Dynamic Analysis	5-88
Data Requirements for State Propagation	5-88
GSR Keywords to Support RTL VCD Flow	5-89
RTL VCD Reports	5-90
Gate Level VCD-Driven Vectorless Dynamic Analysis	5-90
Input Data and Assumptions	5-90
Analysis Procedure	5-91
VCD Dynamic Analysis	5-92
Input Data and Assumptions	5-92
VCD Critical Cycle Selection	5-92
Multi-Bit Sequential Cell Handling in VCD Analysis	5-93
Mixed Mode VCD and Vectorless Analysis	5-93
Analysis Procedure	5-94
Gated Clock Dynamic Analysis	5-95
Input Data	5-95
GSR Keywords for Clock Domain Gating	5-95
Troubleshooting Files and Tips	5-98
Scan Mode Dynamic Analysis	5-98
Early design vectorless scan mode analysis	5-98
Gate-level VCD scan mode analysis	5-101
Evaluating DvD Analysis Results	5-101
Types of DvD Results	5-101
Filtering Minimum DvD Results	5-102
Replaying a Previous RedHawk Session	5-103
TCL Commands to Run Dynamic Voltage Drop Analysis)	5-103

Chapter 6 - Reports

Introduction	6-104
RedHawk Log Files	6-104
Command Files	6-104
Log and Error Files	6-104
Links to Latest Log and Message Files	6-105
GUI Log Message Viewer	6-105
Reviewing Shorts in the Design	6-107
Tech File Viewer	6-108
DEF Import Summary	6-108
Summary Files for Power	6-109
power_summary.rpt File	6-109
<top_level_block>.power.rpt File	6-110
apache.power.info File	6-110
Results for Static and Dynamic Voltage and Current Analyses	6-110
EM, Static IR and DvD Colormap Displays	6-110
Static EM and IR Drop Results Files	6-111
Static and Dynamic Voltage Drop Results Files	6-112
<design>.ir.worst File	6-112
<design>.dvd File	6-112
Static and Dynamic Results for Vias	6-112
Current Report Files	6-113
<design>.ignd File	6-113
<design>.ipwr File	6-113
<design>.ivdd File	6-113
<design>.ivdd.vsrc File	6-114
<design>.ipwr.domain File	6-114
<design>.ignd.domain File	6-114
switch_dynamic.rpt File	6-114
decaps.rpt File	6-115
freqd_ipwr.out File	6-115
Pad Current File	6-115
probe_nodes.out file	6-116
CMM Constraint Violation Reports	6-116
Constraint Violation Summary	6-116
Dynamic Analysis Constraint Summary	6-117
Static Analysis Constraint Summary	6-117
Debugging Using Summary Files in the GUI	6-118
Output Files from Multiple Vdd/Vss Analysis	6-118
Other Files	6-119
Miscellaneous	6-119
Debugging	6-121
Dynamic Simulation Preparation	6-122
Low Power Ramp Up Analysis	6-122

Chapter 7 - Fixing and Optimizing Grid and Power Performance

Introduction	7-124
--------------------	-------

Manual Power Grid Modification	7-125
Changing a Metal Layer or Via Resistivity	7-125
Adding a Single Power/Ground Pad	7-125
Adding a Set of Power/Ground Pads over a Specified Area	7-125
Deleting a Power/Ground Pad	7-126
Adding a Via	7-126
Deleting a Via	7-126
Adding One or Multiple Power Straps	7-126
GUI option for 'eco add strap'	7-128
Editing/Deleting a Power Strap	7-129
Adding a Decap Cell	7-130
Adding Metal Layers and Vias or Via Arrays	7-132
Undo and Redo	7-133
Automated Grid Optimization and Fixing Procedures	7-133
Fixing and Optimization Flow for Static IR Drop Improvement	7-133
FAO Procedure for Multiple Vdd Designs	7-134
Grid Optimization	7-134
Mesh Commands	7-134
GSR Keywords for Region-based Grid Width Sizing (mesh optimize)	7-135
GSR Keywords for Hot-spot Based Grid Width Fixing (mesh fix)	7-135
Examples of Grid Optimization	7-136
Example A - Full Chip Optimization - Relaxing Static IR Drop	7-136
Example B - Full Chip Grid Fixing - Reducing Static IR Drop	7-137
Example C - Partial Chip Optimization, Relaxing Static IR Drop	7-138
Example D - Partial Chip Fixing, Reducing Static IR Drop	7-140
Example E - Mesh Optimize, -taper option, to reduce regional static IR drop	7-142
Example F - Mesh Fix, -taper option, to reduce hot spot static IR drop	7-143
Automated Fixing and Optimization (FAO) for DvD and Timing	7-143
Overview	7-143
Preparation for Decap FAO	7-145
Decap Modification Operations	7-145
Decoupling Capacitance Modification Commands	7-146
Decap Modification Constraints and Interfaces with Other Programs	7-146
GSR Keywords Controlling Decap Modification	7-146
Additional Tools for Fixing High Voltage Drop Areas	7-147
Supplemental Power Routing with 'route fix'	7-147
Cell Moving or Cell Swapping of "Hot Instances"	7-147
Examples of FAO for DvD	7-150
Example G - Full Chip DvD Reduction - Non-overlap Decap and Grid Fix	7-150
Example H - Full Chip DvD Reduction - Decap Overlap	7-152
Saving Design Changes with the ECO Command	7-153
Writing an ECO File	7-153
Reading an ECO File	7-153
ECO File Format Definition	7-153
ECO File Translation for Use by Place and Route Tools	7-154
Fixing and Optimization Command Reference	7-155
Descriptions of Mesh Optimization Commands	7-155

Descriptions of Decap Modification Commands	7-161
Supplementary Voltage Drop Fixing Commands	7-171
GSR Keywords Supporting FAO Functionality	7-171
Command and GSR Keyword Syntax Conventions	7-171

Chapter 8 - Analysis of DvD and Cross-coupling Noise Impacts on Timing

Introduction	8-172
Impacts on Timing	8-172
Overview	8-173
PJX Clock Tree Jitter Analysis	8-174
Analysis Modes	8-175
Timing Analysis Applications	8-175
False Violation Filter	8-175
Design Margin Adjustment	8-175
Methodology Calibration	8-175
Case Analysis for Stress Test	8-176
Silicon Correlation	8-176
LEF-SPICE Pin Mapping	8-176
PJX Fullchip Clock Tree Jitter Analysis	8-176
Overview	8-176
Flow and Interface Description	8-176
Input Data Preparation and Setup	8-177
Running PJX Fullchip Jitter Analysis	8-177
Analyzing Results	8-177
PJX Clock Tree Jitter Sign-Off Analysis	8-178
Overview	8-178
Required Inputs for Clock Tree Jitter Sign-Off Analysis	8-178
Input Data Preparation	8-179
Spice Cell Netlist (.CIR) File	8-179
SPICE Technology Library Data	8-179
Additional Timing Constraints	8-179
Clock Tree Jitter Analysis Setup Procedure	8-180
Edge-to-Edge DDR Clock Jitter Measurements	8-184
Write Mode	8-184
Read Mode	8-185
Running Clock Tree Jitter Sign-Off Analysis from the RedHawk GUI	8-185
Tools > Clock Jitter > Clock Network Summary	8-186
Dynamic > Power > Calculate/ Import	8-186
Running Clock Tree Jitter Sign-off Analysis in Batch Mode	8-186
Setup for Batch Mode Invocation	8-186
Jitter Analysis Command Line Invocation	8-186
Specifying clock instances	8-187
Clock Tree Jitter Results	8-188
Clock Tree Jitter Report	8-188
Clock Tree Browser Display	8-188

Clock Tree Jitter Details Report	8-191
Edge-to-edge DDR Clock Tree Jitter Reports	8-192
Waveform Plots	8-193
Clock Jitter Bottleneck Report	8-194
Jitter Color Map	8-196
Text Reports	8-198
Clock Tree Skew Analysis	8-199
Overview	8-199
Required Inputs and Data Preparation	8-199
Procedure for Clock Tree Skew Analysis	8-199
Running Clock Tree Skew Analysis from the RedHawk GUI	8-199
Running Clock Tree Skew Analysis in Batch Mode	8-199
Clock Tree Skew Analysis Results	8-200
Clock Tree Skew Summary Report	8-200
Clock Tree Analysis Configuration File Reference	8-202
Clock Tree Analysis Keywords	8-202
Input Data Settings	8-203
Jitter Analysis	8-204
Signal Waveforms	8-205
Simulation Controls	8-206
Spice Elements	8-207
Multi-task Controls	8-208
Constraint Settings	8-210
Application Type Selection	8-210
RedHawk Data Usage	8-211
Report Formats	8-212
Sample Timing Configuration File	8-212

Chapter 9 - Characterization Using Apache Power Library

Introduction	9-214
Overview of APL Characterization	9-215
Types of Cell Checking and Characterization	9-215
Pre-run Sample Integrity Checking	9-215
Fast Library Checking	9-215
Library (APL-DI) and Design-dependent (APL-DD) Characterization	9-215
Simulator Support	9-216
Characterization Functions	9-216
Multiple Machine Batch Management	9-217
Platforms Supported	9-218
APL Working Directory	9-218
Cell Characterization Data Preparation	9-218
Data Requirements	9-218
APL Configuration File Description	9-219
Required APL Configuration File Keywords	9-220
Required for Library APL (Design-independent) Configuration File Only	9-223
Optional APL Configuration File Keywords	9-226

Parallel Run Keywords	9-240
Custom Cell Characterization Data Preparation	9-242
Input Vector Files	9-243
Running Cell Characterization	9-246
Setup for a Design-independent (Library-based) APL Run	9-246
Setup for a Design-dependent APL Run	9-246
Setup for Enhanced Design-Independent Characterization	9-247
Running APL Characterization from a UNIX Shell	9-247
Sample APL Invocations	9-249
Low Power Design Characterization	9-250
Characterization for Low Power Designs	9-250
Switch Characterization with aplsw	9-251
Multi-Job Management in APL	9-251
Output Files	9-252
Overall Process Files	9-252
Process Log Files	9-252
Error and Warning Files	9-252
Status Log File	9-252
Results Files	9-253
Individual Cell Characterization Files	9-253
Characterization Results	9-253
Cell Log Files	9-254
APL Results Checking and Processing	9-254
Keywords for Checking Limits	9-255
Resistance, Capacitance, and Leakage Histogram	9-257
Reports of Cells with no APL Data or samples	9-257
Importing and Merging Characterization Data Files in RedHawk	9-258
Importing APL Files	9-258
Merging APL Result Files	9-259
aplccs (CCS2APL) Library Characterization	9-260
Introduction and Syntax	9-260
APLCCS Configuration File	9-261
Importing the CCS Lib Directly	9-261
I/O Cell Characterization	9-261
I/O Cell Characterization Procedure	9-261
Additional Keywords for I/O Cells (Optional)	9-263
Characterization of I/O Cells	9-264
Memory and IP Characterization	9-264
Sim2iprof Switching Current Characterization	9-264
ACE Decap and ESR Characterization	9-264
Identifying tied-off devices as decaps	9-265
Identifying special pins for PWCap characterization	9-265
ACE Configuration File	9-265
Running ACE Characterization	9-272
Output Result Files	9-272
AVM Datasheet Characterization	9-272
Running AVM standalone	9-273

AVM Configuration File	9-273
Running AVM	9-280
AVM Outputs	9-280
Troubleshooting APL Problems	9-280
Checking the Configuration File	9-280
Common Problems	9-280
Debugging Command Line APL Errors	9-281
Sample APL Configuration File	9-281

Chapter 10 - Memory and I/O Modeling

Introduction	10-284
Memory and I/O Modeling Methodology	10-286
Black-box Modeling	10-286
Pin and Grid-Based Abstractions	10-286
Detailed Memory Block Modeling	10-287
Extraction	10-287
GDS2DEF/ GDS2RH Configuration File for Memories	10-290
Required GDS2DEF/ GDS2RH keywords	10-290
Optional GDS2DEF/GDS2RH keywords:	10-290
gds2def -m/ gds2rh -m Configuration File Syntax	10-291
Current Profile Generation	10-292
Static Analysis	10-292
Dynamic Analysis	10-292
Detailed I/O Cell Modeling	10-293
Extraction	10-293
GDS2DEF/GDS2RH Configuration File for I/Os	10-293
Required GDS2DEF/GDS2RH Keywords	10-294
Optional GDS2DEF/GDS2RH Keywords for I/Os	10-294
Current Profile Generation	10-294
Static Analysis	10-294
Dynamic Analysis	10-294
Results and Analysis Including I/Os	10-295

Chapter 11 - Distributed Machine Processing

Introduction	11-298
DMP Flow	11-298
Prepare Data	11-299
Master machine	11-299
Reduction	11-300
Simulation	11-300
Supported Design and Features	11-301
Third Party Distributed Systems	11-301
IBM Platform LSF	11-301
SEQUENTIAL_LAUNCH in LSF	11-302
Oracle Grid Engine (previously known as Sun Grid Engine)	11-302
RTDA NetworkComputer	11-303

Useful Distribution System Commands	11-303
IBM Platform LSF	11-303
Oracle Grid Engine (previously known as Sun Grid Engine)	11-304
RTDA NetworkComputer	11-304
DMP Custom Launcher	11-304
SSH Password-Less Access	11-305
Running DMP	11-305
DMP Configuration File Keywords	11-306
DMP GSR Keywords	11-310
Report Generation	11-310
Log files	11-311
Consolidated redhawk.log	11-311
Important Files	11-311
Export/Import DB	11-312
Single license for Import DB (-view_only mode)	11-312
DMPstat- DMP Monitoring Utility	11-312
Launching dmpstat:	11-313
Monitoring and Debugging Using dmpstat	11-313
Live tracker – Memory, Disk space & Process tracking for each partition	11-313
Command window	11-313
Memory window	11-314
Disk Window	11-315
Process window	11-316
Performance window	11-317
Load Avg	11-319
Machines	11-319
File Menu	11-320
DMP Check	11-320
DMP Abort	11-320
Save	11-321
Heartbeat Monitoring	11-321

Chapter 12 - Package and Board Analysis

Introduction	12-323
Package and Board Models	12-323
Simple Package RLC Model	12-323
Distributed RLCK Package and Board Subcircuit Model	12-324
Support for Package K-parameters	12-326
Linear Current- and Voltage-Controlled Source Models	12-327
Linear Voltage-Controlled Voltage Source (E)	12-328
Linear Current-Controlled Current Source (F)	12-328
Linear Voltage-Controlled Current Source (G)	12-328
Linear Current-Controlled Voltage Source (H)	12-328
S-Parameter Package and Board Modeling for Static Analysis	12-329
S-Parameter Package and Board Modeling for Dynamic Analysis	12-329
Modeling Methodology	12-329

Connecting S-parameter models in the REDHAWK_PKG subcircuit	12-330
Specifying S-parameter models in RedHawk	12-332
Saving/reuse of rational approximation and passivity enforcement results	12-333
Recommendations and limitations in using S-parameter models	12-333
Usage summary and example	12-334
Recommendations for best S-parameter model extraction	12-336
Analysis of the Simulation Results	12-336
Mapping Package Port Names to Die Pad Names in the PLOC File	12-337
Chip-Die Mapping Using Package Compiler	12-337
Overview	12-337
Inputs	12-338
Command Syntax	12-339
Outputs	12-340
Known Restrictions	12-341

Chapter 13 - Low Power Design Analysis

Introduction	13-342
Analysis of Multiple Vdd/Vss Domain Designs	13-342
Analysis of Power Gating Designs	13-345
Types of Power Switches	13-346
Low Power Analysis Switch Modeling	13-346
ON State	13-347
OFF State	13-348
PowerUp and PowerDown States	13-349
Control of Power Gating Switches	13-350
Checking Switches with Two Enable Pins	13-350
Characterization and Implementation	13-351
Switch Configuration Files	13-351
Switch Model Generation	13-355
Defining Block Switching Status with the GSC File	13-356
Obtaining STA Timing Window Data	13-356
Importing Switches into RedHawk	13-356
Adding and Deleting Power Switches	13-357
Reporting on Switches in the Design	13-357
Design Characterization for Power-up Conditions	13-357
Running RedHawk Low Power Analysis	13-358
ON State Analysis	13-358
Ramp-up Analysis	13-358
Running in Mixed Mode	13-359
Power Gating Results	13-359
switch_static.rpt File	13-359
switch_dynamic.rpt File	13-359
charge_switch.rpt File	13-359
Analysis of IP Block Designs with Switched Power	13-360
Introduction	13-360
Data Requirements	13-361

Flow Overview	13-361
GDS2DEF Processing	13-363
Domain Text Labels Available	13-363
No Domain Text Labels Available	13-363
Switch Subcircuit Extraction	13-365
IP Switch Characterization with the aplsw Utility	13-366
ACE Characterization	13-366
Running RedHawk with Switch IP models	13-366
Analysis of Switched RAM Designs	13-367
Introduction	13-367
Types of Switched RAM Supported	13-367
Overview of Switched RAM Analysis	13-368
Model Generation	13-370
GDS Data Preparation	13-370
APLSW Data Preparation	13-376
On-state Analysis - Static IR and DvD Conditions	13-376
GSR Keyword Settings - Static IR Analysis	13-377
GSR Keyword Settings - DvD Analysis	13-378
Ramp-up Analysis	13-379
GSR Keyword Settings - Ramp-up Analysis	13-380
Switched RAM Analysis Timing Control Settings	13-382
Analysis of LDO Low Power Designs	13-384
Overview	13-384
LDO design modeling	13-384
Controlling LDO switching state using GSC file	13-386
Outputs generated in LDO-based analysis	13-387
LDO Modeling with APLDO	13-387
LDO DC model example configuration file	13-388
APLDO example configuration file for load regulation dynamic model	13-389
Generating the DC LDO model	13-391
Testing LDO models	13-391
Other practical LDO applications	13-392
Analysis of Gated Clock Designs	13-392

Chapter 14 - Chip Power Modeling (CPM)

Introduction	14-393
Design Flow	14-394
RedHawk Modeling of Chip Power Delivery Network	14-395
Modeling Choices Based on Number of Pads	14-395
CPM for Flip Chip Designs	14-395
CPM for Wirebond Designs	14-396
3DIC CPM Generation	14-396
Modeling Choices Based on Analysis Speed and Accuracy	14-397
High speed modeling	14-397
High Accuracy Modeling	14-397
ESD-aware CPM Models	14-398

CPM Simulation Procedures	14-399
Initial Setup and Preparation	14-399
Running CPM	14-399
Basic power integrity analysis	14-403
EMI modeling	14-404
User-specified Grouping for Port Creation	14-404
Modeling leakage resistance using arbitrary partitioning	14-405
iCPM- Internal Node Probing	14-406
Resonance frequency-aware mode	14-406
Power Transient Mode (variable power)	14-407
User-configurable mode	14-409
CPM LDO analysis support	14-411
CPM Outputs	14-412
CPM Model Files	14-413
get_cdie.sp File	14-413
*.cdie File	14-414
Using the Chip Power Model	14-415
Differential Voltage Waveforms	14-416
Validating the Model	14-416

Chapter 15 - Reliability and EM Analysis

Introduction	15-418
Temperature Setting for Power EM Calculation	15-419
RedHawk Methodology for Static Power EM Analysis	15-420
Setting Up EM Limits	15-420
Running Power EM Analysis	15-422
Analyzing Static EM Analysis Results	15-422
Methodology for Dynamic Power EM Analysis	15-423
Setting Up EM Limits	15-423
Analyzing Dynamic Power EM Violations	15-424
Current direction and EM violations	15-426
Fixing Power EM Violations	15-427
Methodology for Signal EM Analysis	15-427
Input Data Requirements	15-428
Setup for Signal EM Analysis	15-428
Waveform Specifications	15-428
Wire Merging	15-429
EM Limits	15-429
Custom Current File	15-430
Hierarchical analysis	15-430
Running Signal EM Analysis	15-431
Using the RedHawk GUI in Signal EM	15-432
Defining Equipotential Regions	15-432
Analyzing Results	15-432
Debugging Tips	15-433

Chapter 16 - Pathfinder™ ESD Analysis

Introduction - The ESD Problem	16-434
ESD Analysis	16-435
Overview	16-435
Clamp Cell Definition	16-436
Clamp Files	16-436
Clamp DB Creation	16-441
SPT tracing from clamp nodes	16-441
ESD Rules Files	16-442
Defining Groups of Nets	16-442
Reducing Design Size for Improved ESD Analysis	16-442
Topology and Connectivity Checking of Bumps and Clamps	16-443
Layout Resistance Checking of Bumps and Clamps	16-444
Overview	16-444
Bump-to-Bump (BUMP2BUMP, or B2B)	16-444
Bump-to-Clamp (BUMP2CLAMP, or B2C)	16-445
Bump-to-Instance (B2I)	16-445
Clamp-to-Clamp (CLAMP2CLAMP, or C2C)	16-445
Clamp-to-Inst (CLAMP2INST or C2I)	16-445
Clamp-to-Macro (CLAMP2MACRO or C2M)	16-446
Cross-domain DRIVER2RECEIVER (D2R)-to-Bump	16-446
Including Package Resistance	16-446
Data Flow	16-446
Description of Inputs	16-447
Inputs for 3D-IC Designs	16-448
Resistance Checking for B2B, B2C, and C2C Rules	16-449
Rules File Syntax for Types B2B, B2C, and C2C	16-450
Sample Rules File	16-456
Resistance Rule Checking Results	16-457
Resistance Checking for CLAMP2INST (C2I) and CLAMP2MACRO (C2M) Rules ..	16-457
Rules File Syntax for Types C2I and C2M	16-457
Examples of Rule Checking	16-460
Connectivity Checking	16-462
Combined Rules and Clamp Cell Pin Location File	16-464
Node creation on Clamp Pins	16-464
Sample Invocation	16-464
Resistance Checking for BUMP2INSTANCE Rules	16-465
Rule File Syntax for Types B2I	16-465
Resistance Checking from Bumps to Cross-domain Driver-receivers	16-467
Introduction	16-467
Cross-domain Resistance Check Rule Keywords	16-469
Multiple-clamp Zap R Checks	16-470
ESD Resistance Checking Reports	16-470
Report esdcheck command	16-470
Resistance Checking Output Reports	16-471
B2I results reports	16-472
Clamp Info Reports	16-473

Pass/Fail Reports	16-474
ESD Info Reports	16-477
ESD Summary Reports	16-478
Displaying Resistance Checking Results in the GUI	16-479
Special Resistance Calculations for ESD Analysis	16-487
Current Density Checking	16-489
Mode 1 Rules Files - all clamp paths	16-489
Rules Files - Specified clamp paths	16-493
Bump-to-Clamp and Clamp-to-Clamp Current Density Checking	16-494
Rules File	16-495
CD and arc-based resistance checks include clamp resistance	16-496
Point-to-point current density checks	16-497
Viewing Current Density Checking Results	16-498
ESD CD Report Command	16-498
Results in compressed mode	16-500
ESD-CD report esd_summary.rpt	16-500
ESD Current Density Reports for Pads	16-501
ESD EM Report for ESD-CD	16-501
Displaying Current Density Checking Results in the GUI	16-502
Peak and Differential Voltage Maps	16-502
Current Maps	16-503
Wire and Via Voltage Maps from Current Density Checks	16-504
General Rule File Inclusions and Exclusions	16-505
Clamp Element Exclusions	16-505

Chapter 17 - Memory and Mixed Signal Design Analysis

Introduction	17-506
--------------------	--------

Chapter 18 - Chip Thermal Modeling and Analysis

Introduction	18-507
CTM-Based Thermal Analysis Flow Overview	18-509
Data Preparation for CTM Generation	18-510
APL Library Characterization	18-510
GSR Keyword Settings	18-512
CTM Generation	18-513

Chapter 19 - Timing File Creation Using Apache Timing Engine (ATE)

Introduction	19-515
Overview	19-515
PJX - Fast Fullchip Clock Jitter Analysis	19-516
Setting up ATE	19-516
Configuration File	19-517

Command File	19-518
PJX flow uses STA voltage as nominal and control on threshold voltage	19-519
Handling Ideal Clocks	19-519
Multi-threading	19-520
Special ATE Variables	19-520
getSTA Command Options	19-521
Invoking ATE	19-522
ATE Command Line Options	19-522
Output Files	19-522
Specifying the STA file in RedHawk	19-523
ATE Validation	19-523
Ensuring Correct Creation and Use of the Timing File	19-523
Is the setup OK?	19-524
Did ATE run properly?	19-524
Is the STA file good?	19-525
Contacting Apache Support	19-526

Chapter 20 - Chip-Package Analysis (CPA)

Introduction	20-527
Co-simulation Flows	20-528
Integrated Chip Package Analysis and Chip Thermal Modeling	20-528
CTM Viewer	20-530
Temperature-Aware CTM	20-530
Configuration for Thermal Analysis	20-531
Configuration and Results	20-531
View Result buttons	20-531
Visible option for Vias, Bumps and Balls	20-531
Zoom to a pin	20-532
Integrated Chip and Package GUI	20-532
Chip and Pkg Auto-Connection and Pin Grouping	20-533
DC IR Co-simulation	20-533
AC Hotspot Co-simulation	20-535
HTML-based Reporting	20-538

Appendix A - Installation Procedure

Introduction	A-539
Downloading RedHawk Software	A-539
Program Installation	A-539
RedHawk Operating System/Platform Support	A-540
Setting Up the Apache License	A-540
Setting Up the RedHawk Environment	A-541
License File and Library Directory Setup	A-541
Binary Setup	A-541
Platform-Specific Binaries	A-542
Platform-Independent Binaries	A-542
Invocation	A-542

Appendix C - File Definitions

Introduction	C-544
RedHawk Input Files	C-544
RedHawk Output Files	C-544
Keyword Syntax Conventions	C-544
Apache Technology File (<i>*.tech</i>)	C-545
Encrypting and Decrypting a Tech File	C-545
Full File Encryption	C-545
Partial File Encryption	C-546
Technology File Keywords	C-546
Global Switching Configuration (GSC) File	C-572
Global System Requirements File (<i>*.gsr</i>)	C-574
GSR File Keywords	C-574
Input Data Keywords	C-574
Parameter Keywords	C-612
Custom Cell Modeling Keywords	C-621
Power calculation keywords	C-624
Electromigration Keywords	C-655
Extraction and Netlisting Keywords	C-674
Characterization Keywords	C-691
Timing Keywords	C-693
Simulation Keywords	C-697
DMP Keywords	C-710
FAO General Keywords	C-712
Grid Fixing and Optimization Keywords	C-714
Decap Optimization Keywords	C-718
Low Power Design Keywords	C-722
ESD Keywords	C-728
Name Mapping Keywords	C-732
Warning and Error Message Keywords	C-733
Ignore Function Keywords	C-735
GSR Macro Keywords	C-741
Pad, Power/Ground and I/O Definition Files	C-742
Unified Pad Input File Format	C-742
Individual Pad File Specification	C-745
Pad Cellname File (<i>*.pcell</i>)	C-746
Pad Instance Name File (<i>*.pad</i>)	C-746
Pad Location File (<i>*.ploc</i>)	C-747
Pad PSS File	C-748
Library Technology Files	C-748
Design Netlist (DEF) Files	C-749
Synopsys Library Files	C-749
Custom LIB File Syntax	C-749
Timing Data File	C-751
STA Compact Format Timing File	C-751
Legacy Format Timing File	C-754
Result Files	C-755

Appendix D - Command and GUI Reference

Introduction	D-756
Invoking RedHawk	D-756
Terminating Processes	D-757
TCL / Script Commands	D-757
TCL Syntax Conventions	D-757
TCL Command Summary	D-757
Running RedHawk in the TCL Script Mode	D-818
Starting the GUI at a designated step in batch mode	D-818
TCL Script Execution Examples	D-819
Sample TCL Scripts	D-820
Static IR Drop Analysis Example	D-820
Dynamic Voltage Drop Analysis Example	D-821
Automated Color Map Generation	D-821
RedHawk Graphic User Interface Description	D-822
Mouse Function	D-822
Left Mouse Button Object Selection, Highlighting and Query	D-823
Right Mouse Button Zoom	D-823
Mouse Wheel Zooming	D-824
Using GUI Dialog Box Settings in RedHawk	D-824
GUI Control Buttons	D-824
'View' buttons	D-824
Configuration' buttons	D-825
'View Results' buttons	D-827
' Query ' buttons	D-829
Coordinates readout area	D-831
Full design view area	D-831
User-defined Shortcuts for GUI Buttons	D-831
GUI Menu	D-832
File Menu	D-832
File -> Import Design Data	D-832
File -> Import Database	D-832
File -> Export Database	D-832
File -> Import ESD DB	D-832
File -> Export ESD DB	D-832
File -> Import ECO	D-832
File -> Export ECO	D-832
File -> Import GUI Config	D-832
File -> Export GUI Config	D-832
File -> Playback	D-832
File -> Exit	D-833
Edit Menu	D-833
Edit -> Undo	D-833
Edit -> Redo	D-833
Edit -> Add Pad	D-833
Edit -> Delete Pad	D-833
Edit -> Add Power Strap	D-833

Edit -> Edit Power Strap	D-833
Edit -> Add Via	D-833
Edit -> Delete Via	D-833
Edit -> Add Decap Cell	D-833
Edit -> Delete Decap Cell	D-834
Edit -> Edit Probe	D-834
Edit -> ESD Clamp ECO	D-834
Edit -> Chip Partition	D-834
Edit -> Ruler ->	D-834
Edit -> Single Key	D-834
View Menu	D-834
View -> Chip Layout Map	D-834
View -> Nets	D-835
View -> Connectivity ->	D-835
View -> Technology Layers	D-837
View -> Hierarchy Level	D-837
View -> EM Mode	D-837
View -> Map Configuration ->	D-837
View -> Power Maps ->	D-840
View -> Resistance Maps	D-840
View -> Voltage Drop Maps ->	D-842
View -> Current Maps ->	D-843
View -> Electromigration Maps	D-844
View -> Transistor Pin Maps ->	D-844
View -> Dynamic Instance DvD ->	D-845
View -> Decap Maps ->	D-846
View -> ESD Resistance Lists ->	D-847
View -> ESD Clamp Lists	D-848
View -> ESD Resistance Maps ->	D-848
View -> ESD Current Density->	D-849
View -> Impact on Timing Maps ->	D-851
View -> Clock Jitter Maps (PJX)	D-851
View -> Clock Jitter Maps ->	D-852
View -> STA Critical Path	D-852
Tools Menu	D-852
Tools -> Lowpower ->	D-852
Tools -> Signal EM ->	D-853
Tools -> Chip Power Model ->	D-853
Tools -> PJX Timing Paths (was Fullchip Critical Paths PPX)	D-854
Tools -> Clock Jitter ->	D-854
Tools -> PathFinder SOC	D-855
Tools -> Effective Resistance Computation	D-858
Static Menu	D-858
Static -> Power ->	D-858
Static -> Network Extraction	D-859
Static -> Pad Wirebond Package Constraints	D-859
Static -> Static Voltage Drop Analysis	D-859

Static -> EM Check	D-859
Dynamic Menu	D-859
Dynamic -> Power ->	D-860
Dynamic -> Network Extraction	D-860
Dynamic -> Pad Wirebond Package Constraints	D-860
Dynamic -> Dynamic Voltage Drop Analysis	D-860
Dynamic -> Vectorless Only Analysis	D-860
Dynamic -> VCD Only Analysis	D-860
Dynamic -> EM Check	D-860
Timing Menu	D-860
Timing -> Sign-off Clock Tree ->	D-860
Timing -> Sign-off Critical Path ->	D-860
Timing -> Generate MSDF	D-861
Results Menu	D-861
Results -> Log Message Viewer	D-861
Results -> List of Effective Grid Resistances	D-862
Results -> List of Effective Instance Resistances	D-862
Results -> List of Worst EM	D-862
Results -> List of Worst IR for Wires and Vias	D-862
Results -> List of Highest Power Instances	D-862
Results -> List of Metal-Via EM Ratio	D-862
Results -> List of Worst IR Instances (Static)	D-862
Results -> List of Worst Instance DVD	D-862
Results -> List of Worst Transistor Pin Voltages	D-863
Results -> Analysis Histograms	D-863
Results -> Movie ->	D-863
Explorer Menu	D-864
Explorer -> Generate	D-864
Explorer -> View Results	D-864
Windows Menu	D-864
Windows -> Multiple Pages	D-864
Windows -> Preferences	D-866
Windows -> Detach View Bar	D-868
Windows -> Detach Message Window	D-868
Help Menu	D-868
Help -> About	D-868
Help -> Manual	D-868
Defining Bindkey Functions	D-868
Multiple-key Functions	D-868
Direct Keybinding	D-869
Single-key Functions	D-869

Appendix E - Utility Programs

Introduction	E-871
vcctrans	E-871
vcdscan	E-872
fsdbtrans	E-873

ircx2tech	E-873
rhtech	E-873
gds2rh/gds2def	E-876
Comparison of GDS2DEF and GDS2RH Use	E-877
Distributed Processing of GDS2RH Runs	E-878
Creating the GDS2RH/GDS2DEF Configuration File	E-879
GDS2RH Sub-configuration Files	E-880
GDSII Files Keywords	E-880
Top Cell Definition Keywords	E-881
Nets Definition Keywords	E-883
Layer Map Definition Keywords	E-888
Input LEF Keywords	E-889
Geometry Extraction Keywords	E-891
Selective Cell Hierarchy Handling Keywords	E-894
Auto Pad Location Generation Keywords	E-897
DSPF/SPEF-based standard cell flow Keywords	E-899
Automated Switch Cell Handling Keywords	E-901
Other Keywords	E-903
Running gds2rh or gds2def	E-908
Special Applications	E-908
Boundary Layer Definition	E-908
Specific metal resistor support using the marker layer	E-909
Modeling through-via support	E-909
Bump Via Support	E-910
DSPF/SPEF file-based GDS2DEF Standard Cell Flow	E-911
Select Cell Hierarchy Modeling	E-914
Clamp Cell identification based on marker layer	E-915
gds2rh -m and gds2def -m	E-916
Pin-based Memory Modeling	E-916
Creating the gds2rh -m/ gds2def -m Characterization Configuration File	E-917
gds2rh -m/ gds2def -m Characterization Configuration File Keywords	E-917
Spice Netlist with X,Y Locations	E-919
Spice Netlist without X,Y Locations	E-920
Defining Memories in RedHawk	E-921
Running gds2rh -m and gds2def -m	E-922
pt2timing	E-922
sim2iprof	E-923
Running sim2iprof	E-924
Configuration File Example	E-935
Output	E-937
apleader	E-937
Running apleader	E-937
apleader Output	E-938
Current Outputs	E-938
Equivalent Device Capacitance and Resistance Outputs	E-939
Piecewise Linear Capacitance and Resistance Outputs	E-940
apldev2pwc	E-941

aplcdev2pwc Configuration File	E-941
aplchk	E-942
clampviewer	E-942

Appendix F - Third-Party Software Licenses

ALPHABETIC LIST OF GSR KEYWORDS

ACCURATE_METAL_DENSITY	C-645
ADD_PLOC_FROM_DEF	C-553
ADD_PLOC_FROM_TOP_DEF	C-554
ALLOW_SEPARATED_METAL_PSEUDO_VIAS	C-645
ANALYZE_NETS	C-628
ANALYZE_NETS_FILE	C-628
APACHE_DB_OVERWRITE	C-588
APACHE_FILES	C-588
APL_FILES	C-662
APL_INTERPOLATION_METHOD.....	C-662
APL_MODE	C-601
ATE_CONSTRAINT_FILES	C-664
ATE_USE_REDHAWK_DB	C-665
AUTO_INTERNAL_NET_EXTRACT	C-645
AUTO_PAD_CONNECTION_LAYERS	C-554
BIASPIN_CONFIG_FILE	C-601
BLOCK_INSTANCE_POWER_FILE	C-668
BLOCK_PAR	C-668
BLOCK_POWER_ASSIGNMENT	C-555
BLOCK_POWER_ASSIGNMENT_FILE	C-560
BLOCK_POWER_FOR_SCALING	C-601
BLOCK_POWER_FOR_SCALING_FILES	C-603
BLOCK_POWER_MASTER_CELL	C-561
BLOCK_POWERUP_FILES	C-691
BLOCK_SAIF_FILES	C-605
BLOCK_STA_FILES	C-562
BLOCK_TOGGLE_FILES	C-603
BLOCK_TOGGLE_RATE	C-604
BLOCK_TOGGLE_RATE_FILES	C-604
BLOCK_VCD_FILES	C-563
BOUND_SLEW_TO_MAX_TRANSITION	C-665
BPA_BY_CURRENT	C-564
BPA_BY_LAYER	C-565
BPA_CONN_DISTANCE	C-565
BPA_CONN_MARGIN	C-565
BPA_CURRENT_DENSITY	C-565
BRIDGE_WIRE_CONNECTION.....	C-638
BUS_DELIMITER	C-700
BUS_DELIMITER_STA	C-701
CACHE_DIRECTORY	C-588
CACHE_MODE	C-588
CAP_LIMIT	C-687
CELL_CURRENT_DIST_FILE	C-646
CELL_GEOMETRIES_AS_MFILL	C-565
CELL_PIN_FILE	C-566

CELL_PIN_FILE	C-646
CELL_RC_FILES	C-605
CELL_TOGGLE_RATE	C-606
CELL_TOGGLE_RATE_FILE	C-606
CELL_TYPE_FILE	C-566
CELL_WELL_CAP_FILE	C-567
CEXTRACTION_SPEF_LAYER_MAP	C-646
CEXTRACTION_USE_SPEF.....	C-646
CHARGE_SWITCH	C-691
CHECK_SWITCH_POWERON	C-691
CLEAN_VIAS_AFTER_WIRES	C-646
CLOCK_DOMAIN_TOGGLE_RATE	C-606
CLOCK_DOMAIN_TOGGLE_RATE_FILE	C-606
CLOCK_ROOTS	C-665
CMM_CELLS	C-596
CMM_EXCLUDE_FILES	C-597
CMM_EXPAND_PINS_AT_TOP	C-597
CMM_INSTANCES	C-598
CMM_LAYER_MAP_FILES	C-598
CMM_POWER_OVERRIDE_IPF	C-623
CMM_RIVETED_CONN	C-647
CONFIGURABLE_REPORT_FILE	C-629
CONNECT_SWITCH_PINS	C-647
CONNECTIVITY_RES_THRESHOLD	C-668
CONSISTENT_SCENARIO	C-668
COUPLEC	C-647
CPA_FILES	C-567
CYCLE_SELECTION_GRID_SIZE.....	C-567
DECAP_CELL	C-687
DECAP_CELL_FILES	C-687
DECAP_DENSITY	C-688
DECAP_TILE_MAX	C-688
DECOPLE_LDO_GROUND	C-692
DEF_FILES	C-567
DEF_HONOR_HALF_NODE_SCALE_FACTOR	C-568
DEF_IGNORE_LAYERS	C-568
DEF_IGNORE_NETS_WIDTH	C-568
DEF_IGNORE_PIN_LAYERS	C-568
DEF_IGNORE_SNNet_SHIELD	C-647
DEF_IGNORE_SPECIFIC_LAYERS	C-569
DEF_PG_NETS_FILE	C-589
DEF_READ_ALL_IO_NETS	C-569
DEF_READ_CLOCK_ONLY	C-569
DEF_SCALING_FACTOR	C-589
DEF_TRUE_PATH_EXTENSION	C-569
DEFER_VIA_CREATION	C-700
DEFINE	C-709

DELTA_T_RMS_EM	C-629
DESIGN_IMPORT_REGION	C-590
DETAILED_CONN_ISSUE_RPT	C-629
DIELECTRIC	C-525
DMP_DYNAMIC_POWER_MODEL	C-669
DMP_EP_FAST_MODE	C-607
DMP_SIM_KEEP_LAYERS	C-669
DMP_SP_FAST_MODE	C-607
DO_PININST_INTERNAL_CONN	C-647
DVD_GLITCH_FILTER	C-669
DYNAMIC_64BIT_SOLVER	C-670
DYNAMIC_ADAPTIVE_RON	C-692
DYNAMIC_ARC_LOW_POWER	C-670
DYNAMIC_BYPASS_SHORT	C-670
DYNAMIC_CELL_CROSS_CHECK	C-670
DYNAMIC_CLOCK_SCALE	C-670
DYNAMIC_DETACHED_POSTPROCESS	C-671
DYNAMIC_DISABLE_NEW_WFEXTRACT	C-671
DYNAMIC_EXTEND_VCD	C-671
DYNAMIC_FF_ADJUST_NTRIG	C-671
DYNAMIC_FRAME_SIZE	C-671
DYNAMIC_FREQUENCY_AWARE	C-672
DYNAMIC_GROUP_WIRECAP	C-672
DYNAMIC_GSC_CHECK	C-692
DYNAMIC_MCYC_TW	C-672
DYNAMIC_MIXED_CONSISTENT_SCENARIO	C-672
DYNAMIC_MSTATE_FILTER	C-672
DYNAMIC_PEAK_CURRENT_AWARE	C-672
DYNAMIC_PGARC_REPORT_BASE	C-673
DYNAMIC_POST_BATCH	C-673
DYNAMIC_PRECHECK	C-673
DYNAMIC_PRESIM_DCINIT_SCALE	C-671
DYNAMIC_PRESIM_TIME	C-673
DYNAMIC_RELAX_CONTROL_PIN_CONSTRAINT	C-674
DYNAMIC_REPORT_CLOCK_EVDD	C-674
DYNAMIC_REPORT_DECAP	C-674
DYNAMIC_REPORT_DVD	C-674
DYNAMIC_SAVE_WAVEFORM	C-675
DYNAMIC_SELECTIVE_SAVE	C-675
DYNAMIC_SIMULATION_TIME	C-675
DYNAMIC_SOLVER_MODE	C-675
DYNAMIC_SORT_BY_PERCENTAGE	C-675
DYNAMIC_TIME_STEP	C-676
DYNAMIC_VOLTAGE_CHECK	C-676
EFFECTIVE_VDD_WINDOW	C-676
EM_CCF_ONLY	C-630
EM_CHECK_2D	C-630

EM_CUSTOM_CURRENT_FILE	C-630
EM_DENSITY_ANALYSIS_LAYERS	C-630
EM_DUMP_PERCENTAGE	C-630
EM_LENGTH_USE_MAX_LENGTH	C-631
EM_MISSION_PROFILE	C-631
EM_MODE	C-632
EM_NET_INFO	C-632
EM_PEAK_EQUATION_SOURCE_TECHFILE	C-532
EM_REPORT_<mode>_PERCENTAGE	C-634
EM_REPORT_LINE_NUMBER	C-634
EM_REPORT_MINWIDTH	C-633
EM_REPORT_MODE_ONLY	C-633
EM_REPORT_PERCENTAGE	C-633
EM_REPORT_PERCENTAGE_BY_LAYER	C-634
EM_RULE_SET	C-526
EM_SCALE_DC	C-634
EM_SCALE_PEAK	C-634
EM_SCALE_RMS	C-634
EM_SLEW_NO_STA	C-635
EM_TECH_DC	C-635
EM_TECH_FILE	C-527
EM_TECH_FILE	C-635
EM_TECH_PEAK	C-635
EM_TECH_RMS	C-635
EM_TOPOLOGY_USE_ELECTRON_FLOW	C-636
ENABLE_ATE	C-666
ENABLE_AUTO_EM	C-636
ENABLE_AUTO_SCANLINE	C-676
ENABLE_BLECH	C-636
ENABLE_POLYNOMIAL_EM	C-636
EQUAL_POTENTIAL_AROUND_PAD	C-663
ERROR_COUNTS	C-702
ERROR_LOG_COUNTS	C-702
ESD_CLAMP_FILE	C-696
ESD_CLAMP_PIN_FILE	C-697
ESD_CLAMP_PIN_NODE_DISTANCE	C-698
ESD_DEF_IGNORE_LAYER	C-698
ESD_EXTRACT_CLAMP_NET	C-698
ESD_GSR_OPTIMIZE	C-698
ESD_LOCAL_NETS	C-699
ESD_RULE_FILE	C-699
ESD_SHORT_CLAMP_PIN	C-699
ESD_SIGNAL_NET_FILE	C-700
ESD_SIGNAL_NETS	C-699
EVA_PG_AWARE	C-590
EXCLUDE_REGION	C-570
EXPAND_CELL_PIN_FILE	C-648

EXTRACT_INTERNAL_NET	C-692
EXTRACT_PIN_VOL_INSTS	C-649
EXTRACTION_INC	C-648
EZ_MERGE_NON_RECT_WIRE	C-649
EZ_MERGE_NON_RECT_WIRE_MAX_LENGTH	C-649
FAO_ACCURATE_VOLTAGE	C-680
FAO_ADD_STACK_VIA	C-680
FAO_BYPASS_CUTSIZE_CHECK	C-681
FAO_DECAP_FILL_ALG	C-688
FAO_DECAP_FILL_NEW_FLOW	C-688
FAO_DECAP_OVERLAP	C-689
FAO_DRC_DROP_RATIO	C-683
FAO_DRC_OBS	C-689
FAO_DRC_PL_OBS	C-689
FAO_DVD_TYPE	C-683
FAO_DYNAMIC_MODE	C-683
FAO_ECO_NAME	C-689
FAO_HOLD_LIC	C-681
FAO_IGNORE_COMPRESS_VIAS	C-681
FAO_LAYERS	C-684
FAO_MAX_SHIFT	C-689
FAO_MISVIA_DISTANCE	C-684
FAO_MISVIA_ONE_FILE	C-681
FAO_MISVIA_RPT_SHORT	C-684
FAO_NETS	C-684
FAO_OBJ	C-681
FAO_PLACE_GRID	C-689
FAO_PRECISE_VOLTAGE	C-685
FAO_RANGE	C-685
FAO_REGION	C-682
FAO_ROW_SITE	C-690
FAO_ROW_VDD_SITE	C-685
FAO_SUB_GRID_NETS	C-685
FAO_SUB_GRID_SPEC	C-686
FAO_TURBO_MODE	C-682
FAO_WIDTH_CNSTR	C-686
FAO_WIRE_WIDTH_LOW_LIMIT	C-690
FAST_DEF_READ	C-570
FAST_DEF_READ_DIR	C-570
FAST_IMPORT_APL_MODE	C-662
FIND_ABUTTED_NONSTD_INSTS_PININSTS	C-650
FIX_WINDOW	C-682
FREQUENCY	C-590
GDS_CELLS	C-570
GDS_CELLS_FILE	C-571
GDS_BPFS_OVERRIDE_IPF	C-630
GENERATE_CPM	C-590

GND_NETS	C-590
GROUND_CURRENT_DISTRIBUTION	C-677
GSC_FILES	C-571
GSC_OVERRIDE_IPF	C-607
HALF_NODE_SCALE_FACTOR	C-527
HIER_DIVIDER	C-701
HIER_DIVIDER_STA	C-701
HIERARCHY_CONSISTENT_SCENARIO	C-677
HONOR_LEF_PIN_TYPE	C-572
HOOK_INTERNAL_PIN_POWER_EXTERNAL	C-572
IGNORE_APL_CHECK	C-663
IGNORE_APL_CHECK_SWITCH	C-663
IGNORE_APL_PROCESS_CORNER	C-663
IGNORE_CELLS	C-703
IGNORE_CELLS_FILES	C-703
IGNORE_DEF_ERROR	C-703
IGNORE_DUMMY_PGNET	C-650
IGNORE_DUPLICATE_PGNET	C-703
IGNORE_ERROR_POPUP	C-703
IGNORE_ESCAPE_CHAR	C-704
IGNORE_FILE_PREPARSE	C-704
IGNORE_FILLER_DECAP_CELL_REPORT	C-704
IGNORE_FLOATING_INSTANCE_MISSING_TW_CHECK	C-704
IGNORE_GDS2DEF_UNCONNECTS	C-704
IGNORE_GDSMEM_ERROR	C-704
IGNORE_HALF_NODE_SCALE_FOR_EM	C-637
IGNORE_INST_POSTPROCESS	C-637
IGNORE_INST_WITH_NO_MASTER	C-572
IGNORE_INSTANCES	C-705
IGNORE_INSTANCES_FILES	C-705
IGNORE_INSTANCES_ON_TOP_DEF_REGIONS	C-572
IGNORE_IO_POWER	C-706
IGNORE_LEF_DEF_MISMATCH	C-705
IGNORE_LEF_DEF_SCALE_FOR_EM	C-637
IGNORE_LEF_CELL_PIN_FILE	C-737
IGNORE_LEF_MACRO	C-706
IGNORE_LEF_PIN_DIRECTION	C-705
IGNORE_LIB_CHECK	C-706
IGNORE_MACROEEQ	C-650
IGNORE_NETS	C-706
IGNORE_NETS_FILES	C-707
IGNORE_OPC_METAL	C-650
IGNORE_PLOC_INTERNALNETS	C-573
IGNORE_PLOC_ON_OBS	C-678
IGNORE_PRECHECK_ERROR	C-707
IGNORE_PRIMARY_LOAD_DECAP	C-707
IGNORE_ROUTE	C-707

IGNORE_SHORT	C-708
IGNORE_SIMTIME_CHECK	C-708
IGNORE_TECH_ERROR	C-708
IGNORE_THICKNESS_VARIATION	C-650
IGNORE_UNDEFINED_LAYER	C-708
IGNORE_UNPLACED_INSTANCE	C-708
IGNORE_UPF_PGARC	C-708
IGNORE_VP_CONTROL_ERROR	C-709
IMPORT_NETS	C-573
IMPORT_NETS_FILE	C-573
IMPORT_REGION	C-573
IMPORT_REGION_CELL_OVERLAP	C-573
INACTIVE_NETS	C-666
INCLUDE	C-709
INPUT_TRANSITION	C-666
INST_CURRENT_DIST_MODE	C-650
INSTANCE_POWER_FILES	C-607
INSTANCE_TOGGLE_RATE	C-608
INSTANCE_TOGGLE_RATE_FILES	C-608
INTERCONNECT_GATE_CAP_RATIO	C-609
INTERNAL_CONNECT_PIN_CELLS_FILES	C-651
IP_MODEL_CELL_MAP	C-663
IP_MODEL_INST_MAP	C-663
IP_MODELS	C-664
IPF_ERROR_THRESHOLD	C-609
IR_REPORT_STACKVIA_METAL	C-651
ITERATIVE_SOLVER	C-677
ITR_OVERRIDE_BPFS	C-609
JITTER_ENABLE	C-666
KEEP_POWER_DATA	C-574
LEAKAGE_LIMIT	C-690
LEF_CELL_IGNORE_PIN_LAYERS	C-574
LEF_FILES	C-574
LEF_IGNORE_PIN_LAYERS	C-575
LEF_SCALING_FACTOR	C-591
LEF_USE_DEFAULT_PIN_CAP	C-579
LEFDEF_TECH_LAYER_MAP	C-596
LEFDEF_TECH_LAYER_MAP_FILE	C-574
LIB_FILES	C-575
LIB_HONOR_OUT_SIGNAL_SWING	C-575
LIB_IGNORE_CELL_LEAKAGE	C-576
LIB_IGNORE_IO_VOLTAGE	C-576
LIB_IGNORE_POWER_THRESHOLD	C-576
LIB_PIN_CAP	C-577
LIB2AVM	C-664
LIB2AVM_MSTATE	C-609
LIBERTY_DB	C-575

LICENSE_WAIT	C-592
LONG_WIRE_RES_CALC	C-652
LOWEST_METAL	C-652
MACRO_POWER_FILES	C-577
MACRO_IRDROP	C-652
MEM_RC_MODEL	C-652
MERGE_ABUTTED_ASYM_CUTS	C-653
MERGE_ABUTTED_CUTS	C-637
MERGE_WIRE	C-653
MESH_VIAS_FILE	C-653
METAL	C-527
METAL_DENSITY_BOUNDS	C-653
MIN_WIRE_DIMENSION	C-654
MINWIDTH_FROM_LEF	C-653
MISSING_VIA_CHECK_IGNORE_CELLS	C-708
MMX_ADAPTIVE_SAMPLING	C-677
MMX_RES_MAP_LIMIT	C-592
MPR_MODE	C-654
MPR_PARTITION_REDUCTION	C-654
MPR_POWER_LIMIT_FOR_RED	C-655
MULTI_CYCLE_SCENARIO	C-677
MULTI_GND_PACKAGE_MODEL	C-592
MULTI_THREADS	C-592
NAME_CASE_SENSITIVE	C-700
NET_LOAD_FILE	C-610
NET_TOGGLE_RATE	C-610
NET_TOGGLE_RATE_FILE	C-610
NEW_MERGE_WIRE	C-637
NODE_REDUCTION_MODE	C-655
NOISE_LIMIT	C-682
NOISE_REDUCTION	C-683
NUM_HOTINST	C-690
NUM_HOTSPOT	C-686
NX_SIM	C-678
NX_VECTORLESS	C-678
PACKAGE_SPICE_SUBCKT_INFO	C-578
PAD_FILES	C-578
PARA_CALC_POWER	C-611
PARTIAL_FLAT_SPEF	C-579
PGNET_HONOR_DEF_TYPE	C-579
PGPLOC_DEBUG	C-592
PIECEWISE_CAP_FILES	C-692
PIECEWISE_SWITCH_INPUT	C-679
PIN_DELIMITER	C-701
PIN_DELIMITER_STA	C-701
PIN_SLICE_CELL_LIST_FILE	C-655
PIN_SLICE_LIMIT	C-655

PLOC_SHORT_INTERNAL_NET	C-679
POWER_ALLOW_MULTIPLE_STATE	C-611
POWER_ALLOW_PER_PIN_IPF	C-611
POWER_ANALYSIS_MODE	C-612
POWER_CYCLE_SELECT_BLACK_BOX	C-612
POWER_CYCLE_SELECT_MODE	C-612
POWER_CYCLE_SELECT_POWER_NETS	C-612
POWER_CYCLE_SELECT_REPORT_VDD	C-613
POWER_CYCLE_SELECT_WHITE_BOX	C-613
POWER_DISABLE_SWITCH	C-613
POWER_DOMAIN_TOGGLE_RATE	C-613
POWER_DOMAIN_TOGGLE_RATE_FILE	C-613
POWER_DRIVER_TOGGLE_RATE	C-613
POWER_HIER_REPORT_LEVEL	C-613
POWER_HONOR_LIB_K_FACTOR	C-614
POWER_IGNORE_ASYNCNCH_PIN	C-614
POWER_LEAKAGE_SCALING_FACTOR	C-614
POWER_LEAKAGE_SCALING_FILE	C-614
POWER_MCF_MULTI_CLOCK	C-615
POWER_MISSING_IPF_LEAK	C-615
POWER_MISSING_IPF_POWER	C-615
POWER_MODE	C-615
POWER_REPORT_BIAS_PIN	C-616
POWER_STATE_DEPENDENT_LEAKAGE	C-616
POWER_TRANSIENT_ANALYSIS	C-616
POWER_USE_CCS	C-616
POWER_VCD_COVERED_THRESHOLD	C-616
POWER_VCD_EVENT_SEQUENCE	C-616
POWER_VCD_HONOR_GLITCH_EVENT	C-616
POWER_VCD_IGNORE_ERROR	C-617
POWER_VCD_LIMIT_TR	C-617
POWER_VCD_NUM_PROCESS	C-617
POWER_VCD_OVERRIDE_IPF	C-617
POWER_VCD_REUSE_EVENT	C-617
POWER_VCD_TO_FSDB	C-617
POWER_WORST_IO_PAD	C-617
POWER_WORST_LEAKAGE	C-618
POWER_WORST_MBFF	C-618
POWERUP_OUTPUT_HIGH_PROB	C-693
POWERUP_RANDOM_TOGGLE	C-680
POWERUP_SAVE	C-693
PPI_CELL_EDGE_MAX_NM_THRESHOLD	C-656
PPI_CELL_EDGE_THRESHOLD_PERCENT	C-656
PPI_STD_IGNORE_TOUCH_VIA_MET	C-655
PRIMARY_OUTPUT_LOAD_CAPS	C-618
PRIMARY_OUTPUT_LOAD_CAPS_FILE	C-644
PRIMARY_OUTPUT_LOAD_RC_MODEL	C-656

PRINT_EM_VIA_BOX	C-593
PRINT_EM_VIA_INFO	C-593
PRINT_ONE_PLOC_PER_PADINST	C-579
PROBE_NODE_FILE	C-680
PS_DMP_PERFORMANCE_MODE	C-619
PS_GENERATE_MCYC_SCENARIO	C-619
PS_GENERATE_VLESS_SCENARIO	C-619
PS_GLITCH_POWER_MODELING	C-619
PS_IN_FLOW_APL	C-664
PS_LIB_EXTRACT_CCS	C-619
PS_LIB_EXTRACT_CCS	C-637
PS_RTL_EP_MODE	C-619
PS_RTL_EP_REPORT	C-619
PS_SP_GATED_CLOCK_LOGIC	C-620
PS_VLSG_POWER_DOMAIN_HANDLING	C-620
PSI_SPICE_CELL_NETLIST_FILE	C-667
PUSH_PG_PININST	C-657
PUSH_PININST_CELLS_FILES	C-656
PUSH_SIGNAL_PININST	C-657
QUICK_MESH_WIRE_MERGE	C-658
RAMPUP_OFFSTATE_VOLTAGE	C-693
RAMPUP_SW_REPORT	C-693
RDL_CELL(S)	C-579
READ_LEF_OBS	C-581
RECOMMENDED_SIMULATION_FACTORS	C-708
REMOVE_PARENTLEF_GEOS	C-581
REPORT_ALL_UNCONNECT_PORTS	C-658
REPORT_DISCONN_MIN_LENGTH	C-593
REPORT_FLATTEN_LOG	C-593
REPORT_MAXCAP_VIOLATION	C-620
REPORT_PEAK_MEMORY	C-593
REPORT_PEAK_MEMORY	C-594
REPORT_REDUCTION	C-593
REVERSE_DEF_READ_ORDER	C-581
RTL_NAME_MAPPING	C-620
SAIF_FILE	C-620
SAVE_CONSOLIDATED_R	C-664
SCALE_CLOCK_POWER	C-621
SCAN_CLK_DUTY_CYCLE	C-621
SCAN_CONSTRAINT_FILES	C-621
SCAN_LAUNCH_CAPTURE_MODE	C-680
SCAN_PI_CONSTRAINT_FILE	C-622
SCANLINE_MERGE_LAYERS	C-658
SCANMODE	C-622
SCAN_SHIFTIN_LSB	C-632
SEM_ACCURACY	C-637
SEM_ANALYZE_NET_ONLY	C-638

SEM_CONNECT_NETS'	C-638
SEM_DEFAULT_PARAMETERS	C-638
SEM_DRV_CURRENT_FILE	C-639
SEM_DUTY_RATIO_ROOT	C-639
SEM_ENABLE_SHORTS_REPORT	C-640
SEM_EXTRACT_LONG_WIRE	C-640
SEM_HIERARCHICAL_MODE	C-640
SEM_IGNORE_DISCONNECT	C-640
SEM_IGNORE_NETS_MISSING_DATA	C-640
SEM_IMPORT_CONNECTED_NETS	C-640
SEM_INOUT_PIN_AUTO_SELECTION	C-640
SEM_KEEP_EMPTY_REPORT	C-641
SEM_NET_CHECK_SHORT	C-641
SEM_NET_FILTER_DRIVERS_FILE	C-641
SEM_NET_FILTER_MAX_PEAK	C-641
SEM_NET_FREQ	C-641
SEM_NET_INFO	C-642
SEM_NET_REPORT	C-642
SEM_NEW_SIGEM_INFO_RPT	C-642
SEM_POWER_PIN_HANDLING	C-670
SEM_RECOVERY_FACTOR	C-642
SEM_SLEW_OPTIMIZATION	C-643
SEM_SPLIT_LONG_WIRE	C-643
SEM_TURBO_CLEAN_WIRE	C-643
SEM_VECTORLESS_TIME_STEP	C-643
SIMULATION_CACHE_DIRECTORY	C-680
SINGLE_CUT_SLICE_LAYERS	C-658
SIZE_BASED_PININST_CURRENT_DISTRIBUTION	C-659
SKIP_CONNECTIVITY_CHECK	C-659
SKIP_LAYERS_FILE	C-659
SLEW_MIN_TRANSITION SLEW_MAX_TRANSITION	C-667
SP(CG)_CONSTRAINT_FILE	C-623
SP_CLOCK_GATING_ERROR_OUT_RATE	C-623
SP_CLOCK_GATING_RATIO	C-623
SP_LIMIT_TR	C-623
SPARAM_CHECK_LOWEST_FREQ	C-667
SPARAM_CHECK_REFERENCE_R	C-667
SPARAM_EXACT_DC	C-667
SPARAM_HANDLING	C-668
SPECIAL_SML	C-659
SPLIT_SPARSE_VIA_ARRAY	C-659
SPLIT_VDD_EXTRACT_LP	C-594
SPLIT_VDD_EXTRACT_LP_FSIZE	C-594
SPLIT_VIA_ARRAY	C-659
SPLIT_VIA_ARRAY_CELL	C-660
STA_CRITICAL_PATH_FILE	C-581
STA_FILES	C-582

STA_SLEW_SCALING	C-677
STANDARD_CELL_HEIGHT	C-583
STATE_PROPAGATION	C-623
STATIC_CONNECT_INST_FLOAT_GND	C-595
STATIC_IR_HIDE_DISCON_INST	C-660
STATIC_REDUCTION	C-660
STD_CELL_SINGLE_NOMINAL_VOLTAGE_ONLY	C-595
STEINER_TREE_CAP	C-625
SUBSTRATE	C-544
SUBSTRATE_PGARC	C-544
SWITCH_MODEL_FILES	C-693
SWITCH_MODEL_XTR_FILES	C-694
SWITCH_RES_FILES	C-694
TECH_CONNECTIVITY_FILE	C-583
TECH_FILES	C-583
TEMPERATURE	C-583
TEMPERATURE_DEVICE	C-595
TEMPERATURE_EM	C-643
TEMPERATURE_REPORT_LINE_NUMBER	C-644
TEMPERATURES	C-584
TEMPERATURES_EM	C-644
THERMAL_ANALYSIS	C-625
THERMAL_COUPLING_PITCH	C-654
THERMAL_MODEL	C-625
THERMAL_PROFILE	C-584
THERMAL_TEMP_RANGE	C-595
TOGGLE_RATE	C-626
TOGGLE_RATE_RATIO_COMB_FF	C-626
TSV_MODEL_FILE	C-660
UNITS	C-544
USE_CCS_PIN_CAPS	C-677
USE_DEF_ROW_HEIGHT	C-585
USE_DEF_VIAMODEL	C-585
USE_DEF_VIARULE	C-585
USE_DRAWN_WIDTH_FOR_EM	C-644
USE_DRAWN_WIDTH_FOR_EM_LOOKUP	C-644
USE_FAST_DECAP_ALG	C-626
USE_INTERPETCH_FOR_R	C-660
USE_LEF_FOR_LOGICAL_CONNECTION	C-585
USE_LIB_MAX_CAP	C-627
USE_MF_SWITCH_MODEL	C-695
USE_MVM_PIN_MODEL	C-661
USE_SIGNAL_LOAD_FROM_STA	C-586
USER_MCF_FILE	C-627
USER_STA_FILE	C-585
VCD_CONSISTENT_SCENARIO_FILTER	C-592
VCD_FILE	C-586

VCD_PREPARE_SCENARIO	C-627
VCD_SCENARIO_COMPRESS	C-627
VCD_TIME_ALIGNMENT	C-627
VCD_X_LOGIC_STATE	C-628
VDD_NETS	C-595
VECTORLESS_BLOCK	C-628
VIA <via_name>	C-545
VIA_COMPRESS	C-645
VIA_IR_REPORT	C-596
VIA_MAX_SPACE_FOR_COMP	C-661
VIAMODEL	C-550
VIAMODEL_NAME_CHECK	C-661
VP_CONTROL	C-695
WARNING_COUNTS	C-701
WARNING_LOG_COUNTS	C-702
WIRE_R_COMPENSATION	C-661
WIRE_SLICE_MIN_DIM	C-661
WIRE_SLICE_WIDTH	C-661

Chapter 1

Introduction

Full-chip Static and Dynamic Power Integrity

Apache's **RedHawk™** power integrity solution is a full-chip cell-based power/ground design and verification product with integrated **SPICE**, addressing static and dynamic power integrity from early in the design flow through verification and sign-off. Static IR drop, which determines the IR-drop based on the average value of the power distribution, is merely the DC component of the solution. The more significant components are the AC effects involving temporal relationships between switching events (of cells, clocks, memories, IP, and I/O buffers) and the impact that capacitance and inductance have on full-chip power integrity.

The **RedHawk** product suite consists of two core applications, **RedHawk-S** (static) and **RedHawk-EV** (static and dynamic). **RedHawk-S** supports full-chip static power, IR drop, and EM (electro-migration) analysis. **RedHawk-EV** supports full-chip Vectorless Dynamic™ power and voltage analysis, including the effects of on-chip inductance, package RLC models, and decoupling capacitance analysis and optimization. In addition, **RedHawk** support accurate analysis of advanced low power designs using techniques such as MTCMOS (power gating), clock gating, multiple Vdd/Vss, and substrate biasing. The name **RedHawk** is used interchangeably to refer to either **RedHawk-S** or **RedHawk-EV**, depending on the product bundle you have licensed.

The comprehensive physical power integrity solution offered by **RedHawk** encompasses several breakthrough technologies, including transistor-level analysis accuracy in a cell-based power solution. One of the underlying technologies is **RedHawk**'s Vectorless Dynamic statistical analysis engine, which computes realistic worst-case switching scenarios on a full chip without requiring functional vectors. The Vectorless Dynamic technique, driven by the static timing analysis (STA) timing window and current waveforms in **SPICE**-characterized Apache Power Libraries (APL), yield accurate voltage waveforms for each instance on a power-grid. Severe peak voltage drop areas are identified as hotspot regions, which can be fixed using new decap placement, grid resizing, supplementary power routing, and/or swapping key cell instances at power hotspots. Upon identifying these hotspots in the power networks, **RedHawk** helps determine how to modify the grid or the decap placement, or swap a cell instance at needed locations, to reduce the voltage drop.

RedHawk is fully interoperable with industry standard file formats for technology, design, and library descriptions. To support the requirements of complex SoC designs, **RedHawk** accepts hybrid input formats such as LEF/DEF for standard cell and GDSII for memories, flip-chip bumps, I/Os, cover macros, and IP blocks. In addition, **RedHawk** supports **SPICE**-characterized waveform data in Apache Power Libraries to augment the static data in Synopsys *.lib*, thereby ensuring "true" dynamic accuracy.

While accuracy is one of the primary goals of **RedHawk**'s physical power flow, an equally important mission is to provide designers with full-chip capacity and an easy-to-use product. **RedHawk** enables full-chip RLC power-ground extraction, transient simulation, electro-migration analysis, static and dynamic power and voltage drop analysis, voltage drop impact on key timing parameters such as clock jitter and delay, and power integrity design fixing and optimization in a single product. This enables designers to quickly examine the power and timing impacts on an SoC caused by physical implementation decisions throughout the design flow and insure accurate power performance and sign-off.

Using RedHawk in the Design Flow

RedHawk is fully compatible with industry standard formats and easily drops into existing ASIC vendor and COT flows. RedHawk's physical power methodology, illustrated in Figure 1-1, is easily integrated into all three primary stages of chip design: Design Planning, Design Development, and Design Verification. The use of RedHawk in these design phases is described in the following sections.

RedHawk supports two methodologies for dynamic analysis, depending on where you are in the design flow. The first method is based on `.lib` file data, which provides early feedback on dynamic hotspots. The second method is Apache Power Library (APL) based dynamic analysis, which provides transistor-level accuracy during verification, based on cell current waveforms in the characterized APL.

Design Planning

RedHawk enables early design analysis of static IR drop, and dynamic hotspot estimation using `.lib`-based analysis. At this stage, RedHawk considers wire load models, such as net-to-gate capacitance ratios, to calculate the power distribution. Typically, graphic presentation of IR drop and EM maps can reveal weaknesses of the power/ground mesh structures or the shortage of power/ground pads. Power-grid distribution issues can be easily remedied by wire-sizing or adding more straps or vias. At this point, switching to a flip-chip package may alleviate the static IR power distribution problems. In addition, by black-box consideration of memories and other content which is not yet available, dynamic hotspots due to simultaneous switching can be flagged in individual blocks or on the entire design.

Design Development

In the design development stage, after detailed cell placement, the design is represented by hybrid of input formats, such as LEF/DEF for logic core and GDSII for memories and flip-chip power bumps. If the SPEF or DSPF of the signal nets is unavailable, RedHawk uses Steiner tree wire estimates to compute power distribution. The use of slew information from the static timing analyzer (STA) further increases the accuracy of power analysis. Potential EM problems can be identified. Once the full-chip dynamic hotspot regions are identified, a rapid "what-if" analysis can be performed to assess the impact of package model inductance and intentional decap. Early analysis of the decaps ensure that proper area and location are allocated during placement, rather than after routing. Decap must be placed as close as possible to the victim to ensure proper protection from current spikes.

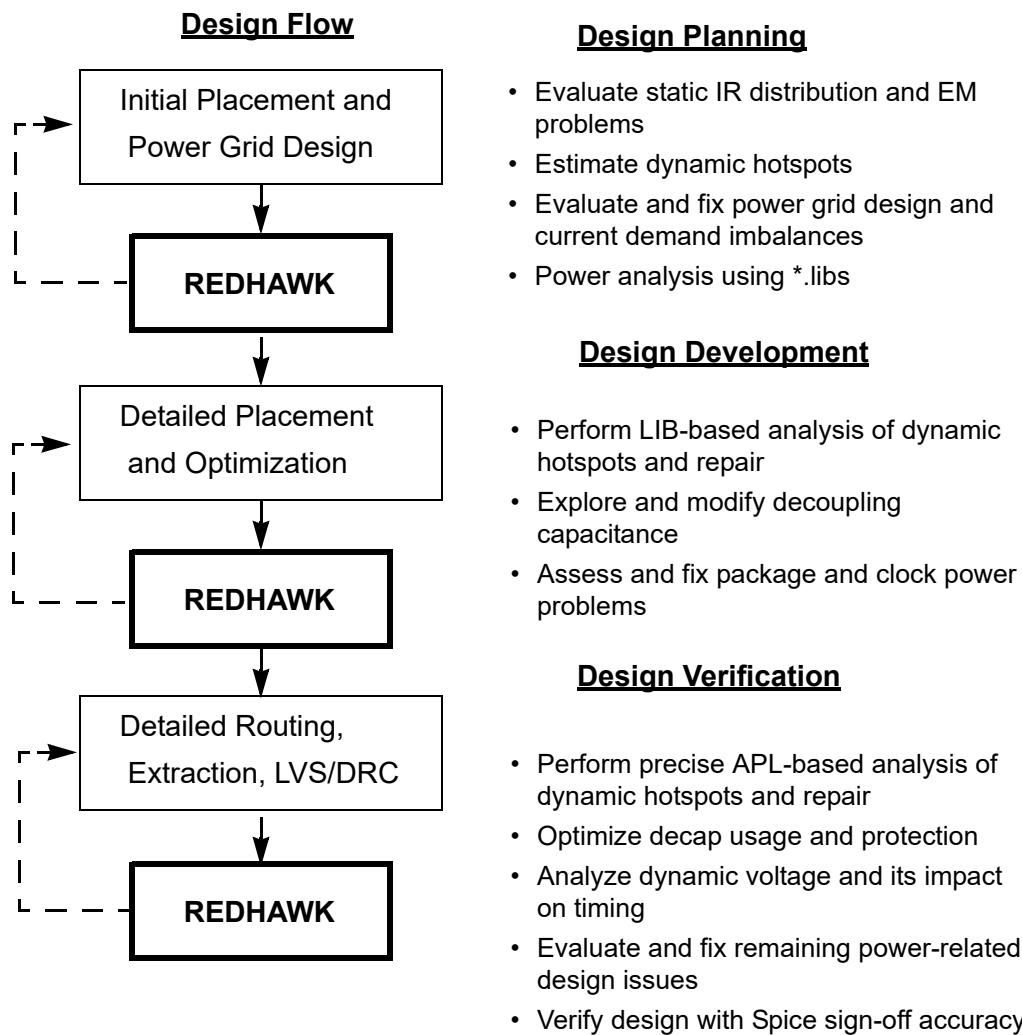


Figure 1-1 Using RedHawk throughout the design process

Design Verification

During the design verification stage, RedHawk uses the detailed resistive and capacitive loading of the signal nets from the extracted DSPF or SPEF file, as well as the final slew and delay information from the STA. The critical information that is required at the verification stage is the transient current waveform data for each cell library instance. The current waveforms are characterized in the Apache Power Libraries (APL), enabling transistor-level accuracy during cell-level verification. APL should be used during the final verification to fully benefit from RedHawk's transistor-level accuracy. This provides the assurance that the decap is optimally placed to protect dynamic hotspots. A final verification of dynamic voltage-induced delays to clock skew and timing is performed before tapeout.

Summary

Due to large power densities, smaller voltage supplies, and higher frequencies, full-chip dynamic and static power integrity is one of the key challenges for designs in 65nm technology processes and beyond. Dynamic power and peak voltage drop is difficult to analyze and correct, being a transient phenomena, and its impact on chip timing and yield is a growing concern.

Major ASIC and IC semiconductor houses and COT companies will benefit by using Apache's **RedHawk** tools to design and verify the dynamic power integrity of very large designs. At the 65nm process node and lower, dynamic power integrity is a design sign-off requirement.

A comprehensive cell-based methodology for full-chip power integrity analysis must address:

- Vectorless Dynamic voltage drop analysis and verification, including impacts on timing and clock trees
- Analysis and optimization of decoupling capacitance
- Waveform-based dynamic power libraries for cells and macros
- Power grid weakness analysis, optimization and fixing
- On-chip and off-chip inductive noise evaluation
- Multiple Vdd and Vss per instance

RedHawk's physical power solution delivers all of these capabilities in a dramatically faster, high-capacity, and easy-to-use design environment. Design teams can confidently deploy **RedHawk**'s full-chip physical power methodology throughout their design flows, taping out on-schedule, and with power integrity insured to silicon.

Chapter 2

RedHawk Flow

Introduction

RedHawk performs several types of power analysis on a circuit:

- static (IR) voltage drop with average cycle currents
- dynamic voltage drop with worst-case switching currents
- electromigration analysis
- critical path and clock tree impacts

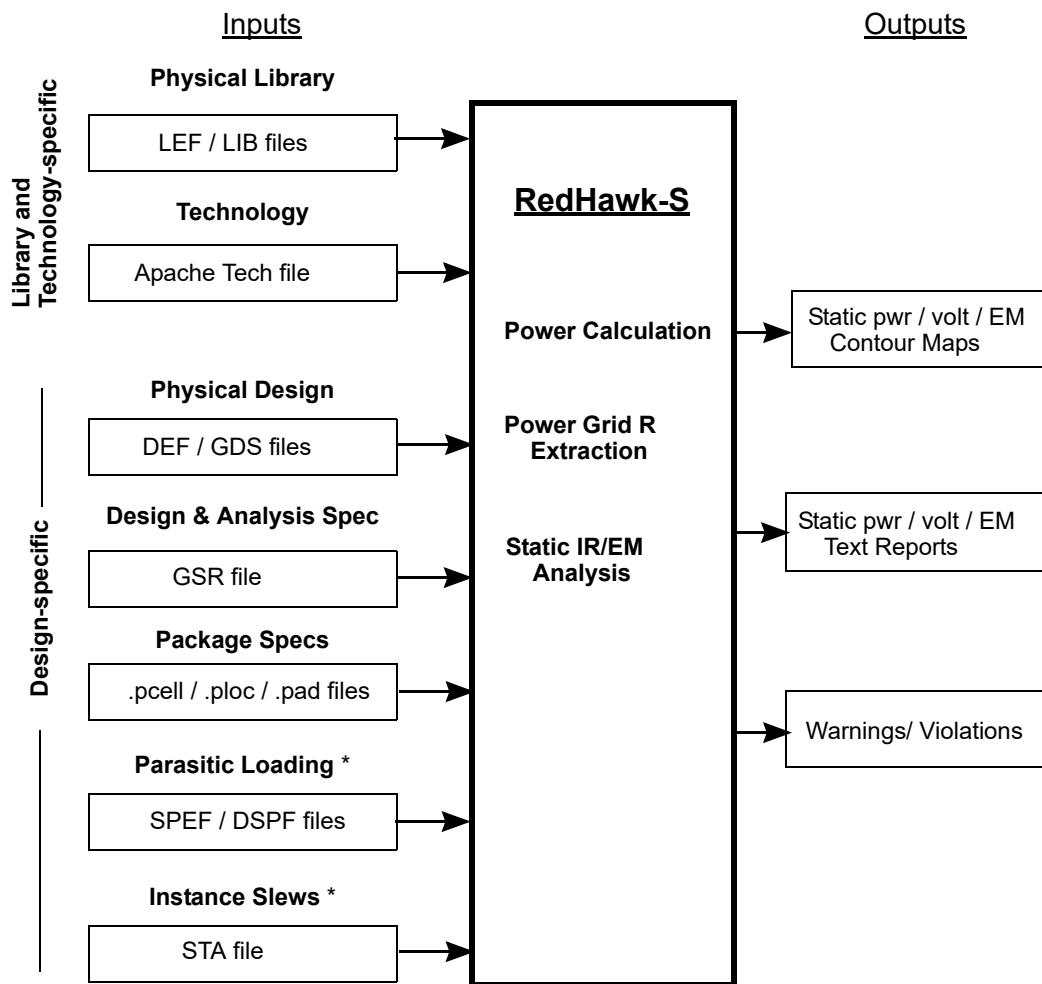
Each type of analysis can be run in different ways, depending on the input data available and the desired speed of analysis and accuracy of results. An overview of the data flow for static IR and dynamic voltage drop analyses is presented in the following sections.

Static Voltage Drop Analysis Flow

Figure 2-1 shows the design flow for running RedHawk-S, the static IR drop solution.

The following are the key steps in the static voltage drop analysis flow.

1. Prepare the design data and input files (see Chapter 3).
 - Prepare the RedHawk technology file data on the IC process (*tutorial.tech*).
 - Prepare the pad cell name, pad instance name, or pad location file. (*tutorial.pcell*, *tutorial.pad*, and/or *tutorial.ploc* file).
 - Generate the STA output file for slews, timing windows, and clock instances (*tutorial.sta*). See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#).
 - Prepare the Global System Requirements (GSR) file (including references to *.tech* file, pad files, STA file, LEF files, DEF files, and LIB files) for static IR/EM and/or dynamic voltage drop analysis (*tutorial.gsr*).
 - Import design data using GSR file (*tutorial.gsr*).
2. Perform power calculation from *.lib* cell data, or import power data if previously calculated (see Chapter 4).
3. Extract power grid (R network).
4. Perform static IR/EM analysis (see Chapter 4).
5. Generate and review maps and text reports of IR/EM results (see Chapter 6).
6. Perform “what-if” analysis and grid modification and optimization to fix areas of critical static IR drop (see Chapter 7).



* Optional for more accuracy

Figure 2-1 RedHawk-S static IR power analysis flow

RedHawk-S outputs include:

- IR voltage drop contour maps
- Electro-migration (EM) analysis
- Power density and average current maps
- Text report files of detailed static power, voltage, and current data
- Warnings and violations reports

Dynamic Voltage Drop Analysis Flow

Figure 2-2 shows the design flow for running RedHawk-EV, the dynamic voltage drop solution.

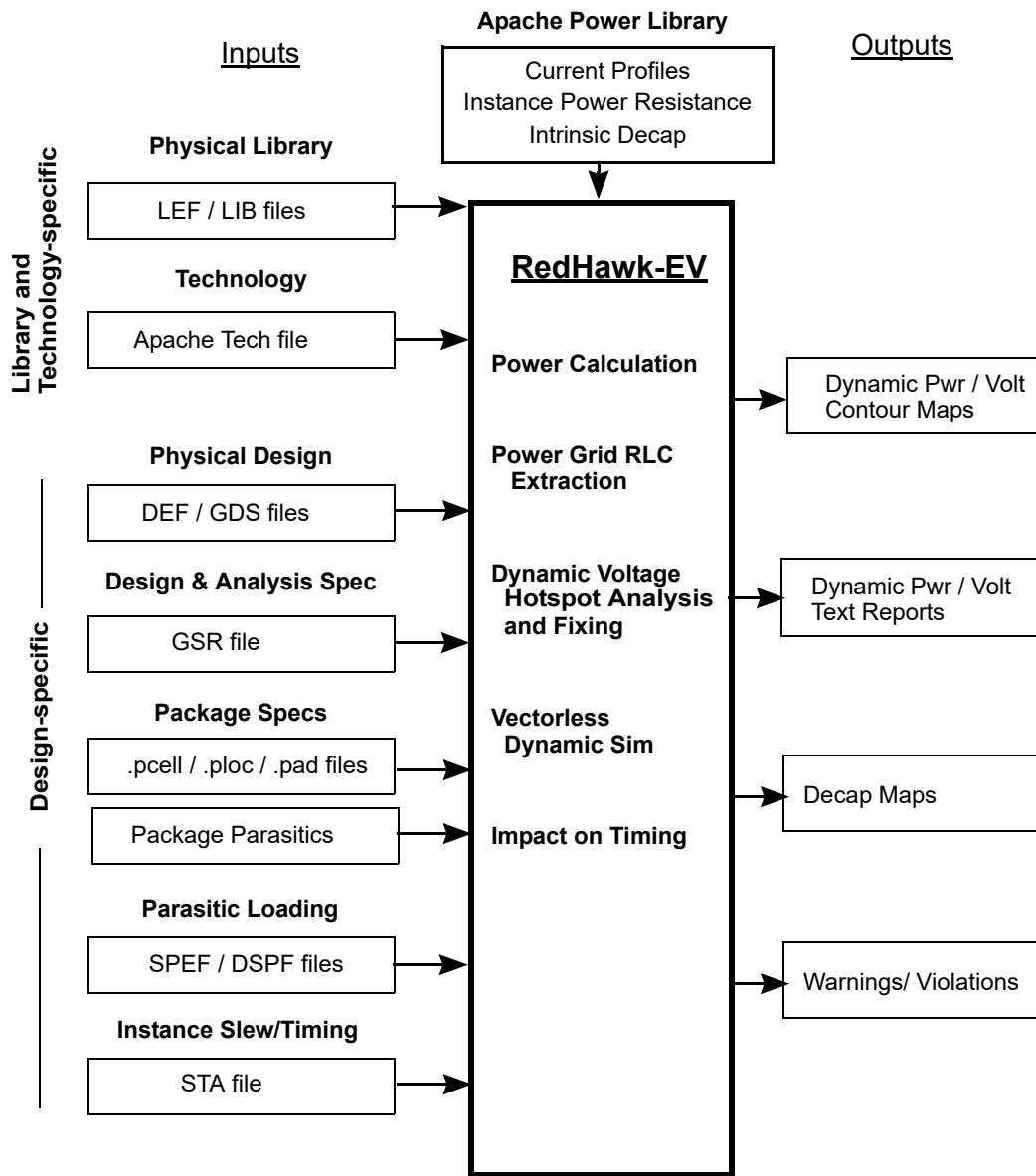


Figure 2-2 RedHawk-EV dynamic power analysis flow

The following are the key steps in the dynamic analysis flow.

1. Prepare the design data and input files (see Chapter 3).

Note: If RedHawk static IR drop analysis has been run, this step does not need to be repeated, except to set specific dynamic run parameters in the GSR file.

- Prepare the RedHawk technology file data on the IC process (*tutorial.tech*).
- Prepare the pad cell name, pad instance name, or pad location file. (*tutorial.pcell*, *tutorial.pad*, and/or *tutorial.ploc* file).
- Generate the STA output file for slews, timing windows, and clock instances. See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#).
- Prepare the Global System Requirements (GSR) file (including references to *.tech* file, pad files, STA file, LEF files, DEF files, and LIB files) for dynamic voltage drop

- analysis (*tutorial.gsr*) (see file definitions in [Appendix C, "File Definitions"](#)).
- Import design data using GSR file (*tutorial.gsr*).
 - 2. Prepare additional inputs required to run **RedHawk-EV**, in addition to those needed for static analysis (see [Chapter 3, "User Interface and Data Preparation"](#)):
 - Timing windows and slews from STA (recommended)
 - Extracted parasitics from SPEF or DSPF (recommended)
 - Pad, wirebond/bump, and package R, L, C, K information
 - Technology data - conductor thicknesses, dielectric thicknesses and dielectric constants (see [section "Apache Technology File \(*.tech\)"](#), page [C-544](#)).
 - SPICE model cards and library subcircuits. This is required to characterize the current waveforms in the Apache Power Libraries
 - SPICE subcircuits for all memories, I/Os, and IP blocks (optional)
 - 3. Calculate detailed power distribution from *.lib* cell data, or import power data if previously calculated (see Chapter 4).
 - 4. Extract power grid (RLC network).
 - 5. Run APL (Apache Power Library) characterization, to obtain current profiles under typical corner conditions, Effective Series Resistance (ESR) for the power circuit and as well as decoupling capacitance and leakage current (see [Chapter 9, "Characterization Using Apache Power Library"](#)).
 - 6. Perform dynamic voltage drop and peak current analysis (see [Chapter 5, "Dynamic Voltage Drop Analysis"](#)).
 - 7. Generate and review maps and text reports of dynamic analysis results (see [Chapter 6, "Reports"](#)).
 - 8. Perform “what-if” analysis and optimize decap placement, make grid modifications and perform instance swapping to fix “hotspots” -- areas of highest dynamic voltage drop (see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#)).

RedHawk-EV outputs include:

- Dynamic voltage drop color maps and power density color maps (see [Chapter 5, "Dynamic Voltage Drop Analysis"](#))
- Capacitance maps, including decap effects (see [Chapter 5, "Dynamic Voltage Drop Analysis"](#))
- Report files (see [Chapter 6, "Reports"](#))
- ECO script to place and route environment (see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#))

Chapter 3

User Interface and Data Preparation

Introduction

This chapter describes both the TCL command line user interface and the graphical user interface (GUI) available for controlling RedHawk, the proper directory organization for managing input and output data, as well as the input data files and preparation required before starting RedHawk analysis. See Appendix A for instructions on installing the software.

TCL Command User Interface

TCL Command Summary

The RedHawk program can be invoked and controlled using the TCL command line interface and a set of commands described in this section.

The basic RedHawk invocation is:

```
redhawk
```

Following is a summary of the TCL commands available for running RedHawk. Details on the TCL commands and their options and syntax are found in [section "TCL / Script Commands", page D-756](#).

- cell swap** - allows high power cells to be moved to a location that has lower voltage drop
- characterize** - runs Apache Power Library (APL) characterization to generate dynamic Vdd/Vss current waveforms and intrinsic/intentional decap values
- condition** - sets, displays and unsets various conditions for post-DvD analysis (such as plot and print)
- config** - sets GUI options for reviewing, analyzing and debugging of RedHawk results
- decap** - allows various types of modifications to decoupling capacitance to reduce dynamic voltage drop
- dump** - writes out colormaps in GIF format
- eco** - defines changes to the design database (GUI-based operation only), which is equivalent to GUI 'what-if' operation
- export** - exports data from RedHawk engine for use by other tools
- fao** - allows modification of the physical design
- generate** - generates specified files for review and/or future use
- get** - allows querying the database to find cell or instance names based on pattern matching to search and find design elements, execute scripts, or allow more detailed access to the design
- gsr** - retrieves GSR variable values, sets GSR values, displays the contents, or sends the GSR contents to a file
- history** - displays all previous commands typed on the command line during the session

import - imports specified files for use in RedHawk analysis, such as APL, AVM, DB, DEF, ECO, GSR, guiconf, LEF, LIB, PAD, POWER, or TECH

ircx2tech - converts an input iRCX file to an appropriate Apache tech file

marker - adds or removes design marker(s) specified either as an x,y position or as an instance name.

mesh - performs various types of modifications to power grids, including adding, deleting, and modifying widths and spacing, to reduce voltage drop

message - displays Error, Info and Warning message information

movie - sets up or plays an instance-based “movie” of Dynamic Voltage Drop history

perform - runs selected RedHawk analyses, such as extraction, power calculation, static analysis, dynamic analysis, and timing slew

pfs - executes various ESD analysis functions

plot - generates graphical plots based on specified conditions

print - prints text-based reports for specified conditions

probe - selects and de-selects particular nodes prior to extraction

psiwinder - performs various clock and timing analyses on a design

query - displays information and analysis parameters on selected objects in the design

report - creates reports on RedHawk analyses

ring - adds or deletes power rings

route fix - routes new power/ground wires and adds associated vias to reduce excessive voltage drop

save - saves the design database for later use

select - selects and highlights objects in the GUI

setup - sets up data and conditions required for performing RedHawk analysis

show - displays a colormap of specified RedHawk results in the GUI

window - changes the GUI window geometry so that the “dump gif” command can produce a higher resolution color map

zoom - zooms design view in and out

The RedHawk graphical user interface and its features are summarized in the following section.

Graphical User Interface

Elements of the GUI

The graphical user interface allows convenient viewing of design elements and the results of RedHawk analysis. Whenever RedHawk is invoked from the command line the GUI is automatically displayed.

The RedHawk GUI includes a number of panels with different functions, as shown in Figure 3-1. The functional areas of the GUI are:

- Menu bar
- Primary display area
- Multiple page view tabs
- Text display area

- Control buttons
- Design view area
- Cursor location readout
- TCL command line

The functionality of these GUI elements is summarized in the following sections. For details on the GUI commands and functions, see appendix [section "RedHawk Graphic User Interface Description", page D-821](#).

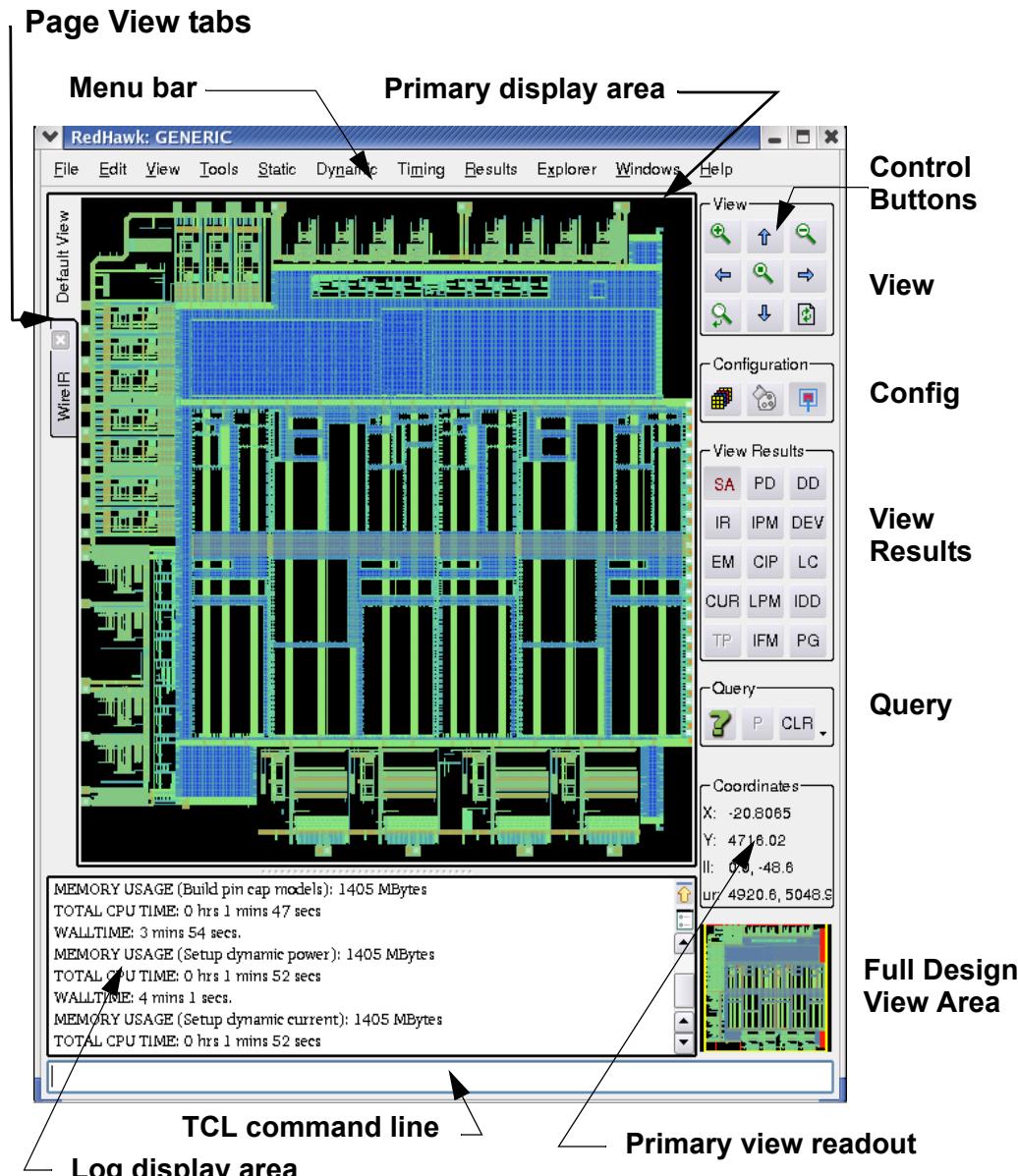


Figure 3-1 RedHawk graphical user interface

Menu Bar

The **RedHawk** functions available from the menu bar are as follows:

- **File** - allows you to import and export design data, databases, ECO information, and configuration information
- **Edit** - allows you to edit information in the GSR and Tech files, as well as modify elements of the power grid such as pads, straps, and vias.
- **View** - allows you to display and change the color map for viewing design elements (such as nets, layers, instances, capacitance, and general object properties) and analysis results (such as IR and dynamic voltage drop, power and current density, EM problems, slack and delay).
- **Tools** - allows you to invoke RedHawk tools, such as those for fixing and optimizing power grid and decoupling capacitance, low power circuit design analysis, and PDX critical path and clock tree analysis.
- **Static** - allows you to perform various functions associated with static voltage drop analysis, such as power calculation, network R extraction, pad and package constraints, and IR drop and EM analyses
- **Dynamic** - allows you to perform functions associated with dynamic voltage drop analysis, such as power calculation, network RLC extraction, pad and package constraints, and vectorless or VCD-based dynamic voltage drop analysis.
- **Timing** - allows you to perform functions associated with circuit timing analysis, such as instance slew, slack, K-factor, delay, clock networks, modified SDF data, and impact of voltage drop on timing
- **Results** - allows you to set up and generate results information on RedHawk analyses, such as histograms, log messages, and ordered lists of instances or nodes with worst-case IR drop, dynamic voltage drop, electromigration, power/ground design weakness. Also allows creation and display of “movies” of voltage drop by time steps.
- **Explorer** - controls the generation and display of analysis results
- **Windows** - allows you to set up and create multiple-page simultaneous displays, as well as set preferences for the windows display.
- **Help** - displays information about the RedHawk software version and provides access to the latest “RedHawk Users’ Manual” in PDF format.

Primary Display Area

The primary display area allows the full design or segments of the design to be displayed along with selected characteristics, highlights and analysis metrics.

Log Display Area

The log display area in the lower left corner of the GUI shows RedHawk text inputs, progress status, error messages, and results.

Control Buttons

The sets of control buttons down the right side of the GUI window invoke changes in the display or activate RedHawk commands. The function of the button is identified when the cursor is placed over it. The control buttons include the following functions:

- ‘View’ buttons - modify the size and center of the view in the primary display
- ‘Configuration’ buttons - view layers, set color range, and view power pads
- ‘View Results’ buttons - displays “maps” of various parameters superimposed on the design, such as power density, decap density, voltage drop, and current.
- ‘Query’ buttons - allows querying properties of selected design objects
- ‘Coordinates’ readout - indicates the x,y location of the cursor, and the bounding box

of the primary display area, in microns.

Design View Area

The min-view area always displays the ‘Show All’ view of the primary display selection, to show the context and orientation of the primary view.

TCL Command Line

At the bottom of the interface is a TCL command line for entering text commands. See section "TCL / Script Commands", page D-756, for syntax conventions and a complete list of TCL commands.

Using the GUI

Exporting and Importing a GUI Configuration

All of the settings that you can modify with the two ‘Configuration’ button dialog boxes can be saved for future use in a configuration file, either by using the menu command **File->Export GUI Config** or by using the TCL command:

```
export guiconf <GUI-config-filename>
```

Then to restore the GUI configuration to a saved version, there are three ways to import it:

1. Select from the menu **File->Import GUI Config**, and choose the saved configuration file you want.
2. Use the TCL command:

```
import guiconf <GUI-config-filename>
```
3. Set the environment variable ‘APACHE_CONF_FILE’ from the command line:

```
setenv APACHE_CONF_FILE <GUI-config-filename>
```

This method has the lowest priority; methods 1 and 2 override method 3.

The TCL command ‘config’ may be used to change viewlayers, viewnets and colormaps if you do not want to export/import the GUI configuration. The ‘config’ command can change the GUI configuration at any time from the command line.

For example, the ‘config colormap’ command can configure the maximum and minimum voltage drops displayed in either absolute drop or percentage, as in the following example:

```
config colormap -percent -min 10 -max 24 -instance
```

This sets the minimum instance voltage drop displayed to 10% of nominal Vdd and the maximum voltage drop percentage displayed to 24%.

RedHawk Data Preparation - Static and Dynamic Analysis

RedHawk Program Files

Data files required for RedHawk analysis come from three types of sources:

Des - standard files needed for or generated by the chip design process

Cr - Apache files that you need to create for RedHawk analysis

The input files may be required for either static or dynamic voltage drop analysis, as shown in Table 3-1, or they may be optional; files not strictly required usually provide more accurate analysis if provided. Files not required, but strongly recommended, are designated “Dyn-Rec”.

Table 3-1 Data Files for RedHawk Analysis.

File	Description	Source	Req'd
DEF (*.def) design files	Physical description of instances, power and ground network, and other circuit elements. The currently supported version of DEF is lefdef 5.7. If there are multiple .def files, you must specify each .def file on a line with its absolute or relative path from the RedHawk run directory in the ‘DEF_FILES’ GSR keyword (see section "DEF_FILES", page C-587), and the option “top” or “block” to indicate the top-level or block-level DEF file.	Des	Stat/Dyn
LEF (*.lef) library files	Library Exchange Format file. Physical description of library cells. The currently supported version of LEF is lefdef 5.7. If there are multiple LEF files, you must specify each .lef file on a line with its absolute or relative path from the RedHawk run directory in the ‘LEF_FILES’ GSR keyword (see section "LEF_FILES", page C-597). Also, all of the following four keywords must be in the technology file, although they can be commented out, for RedHawk to recognize it as the technology LEF file: LAYER, TYPE, WIDTH, ROUTING	Des	Stat/Dyn
LIB file (*.lib)	Synopsys-format library files contain logical descriptions of library cells, with power and timing tables, and are used for power calculation. As an alternative, the entire set of LIB file data can be included under the ‘LIB_FILES’ keyword in the GSR file. Also, up to five string variable parameters per operating condition can be handled.	Des	Stat/Dyn
Device models	BSIM device models for simulation.	Des	Dyn
Standard cell SPICE netlists	Spice netlists of standard cells and decoupling capacitor cells. SPICE model cards and library subcircuits. This is required to create the current profiles in the Apache Power Library.	Des	Dyn

STA file	<p>Static Timing Analysis file. Produces instance-specific minimum/maximum transition times and defines the timing windows and clock network data. Required for signoff accuracy.</p> <p>The STA output file must be defined by the keyword 'STA_FILE' in the <i>.gsr</i> file.</p> <p>Also improves the accuracy of static IR power and voltage drop analysis. If the timing window information is missing for an instance, the power of the instance is assumed to be zero unless the information is specified in the <i>.gsr</i> file with the BLOCK_POWER_FOR_SCALING keyword, or is available in the VCD_FILE. See Chapter 19, "Timing File Creation Using Apache Timing Engine (ATE)" for details on generating this file.</p>	Des	Dyn-Rec
SPEF file	<p>Standard Parasitic Exchange File. Instance-specific signal wire parasitic data. Req'd for signoff accuracy. Note that when you have a hierarchical design, the following cases are acceptable:</p> <ul style="list-style-type: none"> (1) Hierarchical DEF with hierarchical SPEF: every SPEF must be associated with a DEF block. (2) Hierarchical DEF with flattened SPEF (3) Flattened DEF with flattened SPEF. <p>To perform PJX timing analysis in signal integrity mode, coupling capacitance data must be provided.</p>	Des	Dyn-Rec
VCD file	<p>Value Change Dump file for determining net toggling activity and peak power. Required by <i>vcdtrans</i>/<i>vcdscan</i> utilities for VCD-based analysis. If VCD files are available, then the 'VCD_FILE' keyword can be used to read the net toggling information from the VCD files. If VCD files are not available, you can specify the default toggle rate and running frequencies in the <i>.gsr</i> file. See Figure 3-2 for a sample VCD plot of power use by cycle.</p>	Des	Dyn-Rec
GDSII file	<p>Physical description of circuit elements to create DEF views. Used by <i>gds2def/gds2rh</i> utilities. Required for accurate analysis of memories, flip-chip bump layer descriptions and analog blocks, which may only be available in GDSII format, and is converted using RedHawk's GDSII data preparation utilities, <i>gds2def/gds2rh</i>, <i>gds2def -m/gds2rh -m</i>. For details, see section "gds2rh/gds2def", page E-875.</p> <p>The GDS cells can be imported using the GSR file keyword GDS_CELLS, which lists the cells and the path to where the <i>gds2def/gds2rh</i> utility was run. See section "GDS_CELLS", page C-591, for the proper GSR syntax. Refer to Appendix E, Utilities, for details on the RedHawk GDS conversion utilities.</p>	Des	Dyn-Rec

Technology file (*.tech)	Design-specific file containing metal layer and package information for each process corner. Refer to Appendix C for a detailed description of the technology file contents and format.	Cr	Stat/Dyn
GDS layer map file	To enable GDS layer number to LEF layer name mapping.	Cr	Sat/Dyn
Global System Requirement file (*.gsr)	Design-specific information for operating conditions and analysis parameters such as global specs for VDD nets, clock roots and their respective frequencies, default logic toggle rate, and switching conditions of special nets such as reset nets, I/O nets, and I/O pin output loadings. The GSR also contains specifications for other input files. Make sure that the GSR file contains file paths and names for all needed files, such as TECH, pad, STA, DEF, LEF, LIB, VCD, GDS, SPEF, and package. Refer to Appendix C for a description of the contents of the .gsr file.	Cr	Stat/Dyn
Pad files (.ploc, .pad, or .pcell)	Power and ground pad descriptions, cell names, instance names, and X,Y locations. Refer to Appendix C for descriptions of the pad files, including the pad cell names (.pcell) and pad instance name (.pad) files.	Cr	Stat/Dyn

NOTE: RedHawk can read LEF, DEF, STA, SPEF, VCD, FSDB, SAIF and GDS files in *.gz (gzip) format.

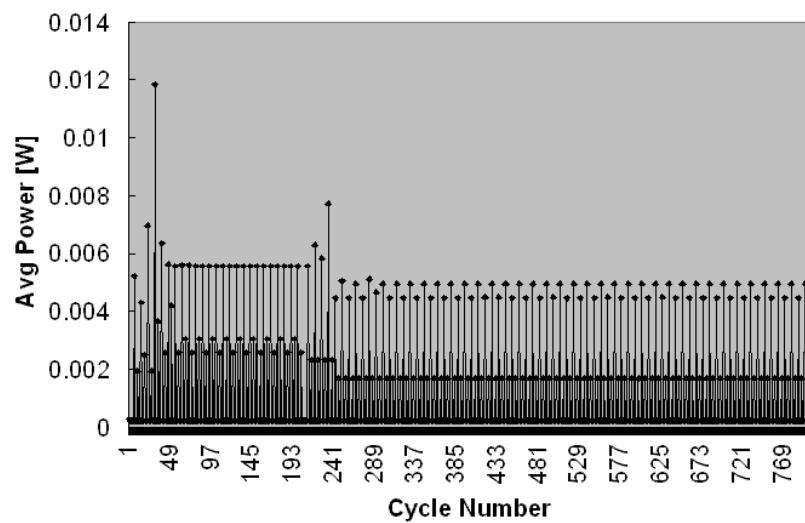


Figure 3-2 vcdscan plot of power by cycle

Multiple Vdd/Vss Analysis

Redhawk is natively aware of multiple power and ground domains. As such it accurately handles not only designs with multiple Vdd supplies, but also individual cells with multiple

power and ground domains, and always looks for data supporting multiple domains per cell. Common examples of multiple Vdd design elements are:

- level shifters
- flip-flops and latches with primary and retention power
- isolation logic used to ring blocks at different potentials
- memory/register file blocks with independent voltages for the array and peripheral sections
- I/O cells with both external and internal power and ground supplies

The data required to determine the proper modeling of multiple power domain cells for both static and dynamic analysis are listed in the following paragraphs.

For cells with single VDD and VSS supply pins, there may be changes required in **RedHawk** configuration files, but no input data changes are needed. In most cases, designs using single power/ground domain cells only require that any *gds2def -m/gds2rh -m* modeled cells are rebuilt. This is not the result of mVDD support, but rather an updated modeling construct for non-uniform current distribution in *gds2def -m/gds2rh -m* modeled IP.

For cells with multiple VDD pins and one or many VSS pins, additional data is required by **RedHawk**, both in terms of user data and also in configuration files.

Summary of Differences in Input Data Files

The following table summarizes the data file changes for multiple Vdd analysis. Those marked 'GEN' are related to a general change in **RedHawk** data handling and not related to multi-Vdd.

Table 3-2 Input data changes needed

Type of File	Type of cells requiring changes	Changes required: GEN/MVdd
LIB	Multi-domain only	MVdd. Certain attributes needed in the library, cell, and pin level definitions. Liberty format does not accept a cell with multiple ground pins, in which case a custom LIB file must be created.
LEF	Multi-domain Single-domain	GEN. All power and ground pins must have 'USE < >' and 'DIRECTION < >' attributes defined correctly.
User instance power file	Multi-domain only	MVdd. Specify pin-based power.
APL data	Multi-domain only	MVdd. Rerun APL characterization to create separate instance current profiles for each Vdd and Vss arc, and Cdev per pin.
Memory models	Multi-domain Single-domain	GEN. Memory models created using 'gds2def -m' /'gds2rh -m' should be re-extracted to use the enhanced memory handling capability. If using 'gds2def/gds2rh' no changes are needed.

Table 3-3 summarizes the changes needed to RedHawk configuration files to perform multiple Vdd/Vss domain analysis. Further descriptions of the changes needed are provided following the table.

Table 3-3 Configuration file changes needed - Multi Vdd

Type of File	Type of cells requiring changes	Changes required: GEN/MVdd
GSR	Multi-domain Single-domain	GEN. Use GSR-based specification for all input files. Use 'GDS_CELLS' keyword to use new memory models when using <i>GDS2RH -m</i> .
Custom LIB	Multi-domain with multiple GND pins	MVdd. Specify the power-ground arc pairs.
APL configuration	Multi-domain	MVdd. Specify LEF files for APL-DI.
AVM configuration	Multi-domain Single-domain	MVDD. Specify VDD and GND pin names in the configuration file, as follows: VDD_PIN <net names> GND_PIN <net names>
GSC	Multi-domain Single-domain	GEN. 'OFF' state is for power-gated blocks only. To turn regular cells 'Off', use 'DISABLE'.

Liberty Library Syntax Differences for Multiple Vdd/Vss Domains

The following section describes the requirements in multi-Vdd/Vss domain designs for LIB files.

'power_supply' group at library level

The 'power_supply' group captures all nominal information on voltage variation. It defines the default power supply name and the nominal voltage values of the supplies. An example of the new syntax follows:

Syntax:

```
power_supply () {
    default_power_rail : <domain_name> ;
    power_rail (<domain_name>, <voltage>) ;
    ...
}
```

Example:

```
power_supply () {
    default_power_rail : VDD ;
    power_rail(VDDNW,0.81);
    power_rail(VDDC,0.81);
    power_rail(VDD,0.81);
    power_rail(VDDIN,0.95);
}
```

'rail_connection' statement in the cell section

The 'rail_connection' attribute must define all multiple power supply and ground domain pins used in a cell. It maps the library power supply name to the cell LEF power pin name to which it is connected. An example of the new syntax follows:

Syntax:

```
    rail_connection ( <power_pin_name>, <power_domain_name> ) ;
    rail_connection ( <grnd_pin_name>, <grnd_domain_name> ) ;
    ...

```

Example:

```
rail_connection (VDDS, VDD1) ;
rail_connection (VDD_RET, VDD2);
rail_connection (VSS_A, VSS) ;
```

'leakage_power' statement

The 'leakage_power' statement defines rail-specific power values for internal/leakage power. An example of the new syntax follows:

```
leakage_power () {
    value : 2.500000E+00;
    power_level : "VDDNW";
}
leakage_power () {
    value : 6.740750E+03;
    power_level : "VDD";
}
```

'input_signal_level' attribute

The input_signal_level attribute defines the voltage level that should be applied to the input or inout pin/bus/bundle.

Syntax:

```
input_signal_level : <domain_name> ;
```

'output_signal_level' attribute

The output_signal_level attribute defines the voltage level that should be applied to the inout or output pin/bus/bundle.

Syntax:

```
output_signal_level : <domain_name> ;
```

internal_power

The internal_power attribute table defines short-circuit energy values during switching for all power arcs.

power_level

The 'power_level' attribute specifies the power supply used to characterize the internal power of a pin.

Syntax:

```
power_level : "<domain_name>" ;
```

Example:

```
pin ( Z ) {
    direction : input ;
```

```

        output_signal_level : VDD1;
        internal_power() {
            power_level : VDD1 ;
            power (power_template) {
                values ("1, 2, 3, 4");
            }
        }
    } /* end pin */
}

```

P/G Arc Definitions in Custom LIB Files

To support cells with multiple Vdd *and* multiple Vss pins, the P/G arcs must be specified, which define the current path between each VDD node to the associated GND node pair. When there are multiple power domains and multiple ground domains, defining the power domain arcs is required to provide the static and dynamic simulators with the current node pairs necessary to correctly assign currents.

If a UPF LIB is available, P/G arc info can be obtained from the 'related_power_pin' and 'related_ground_pin' values in LIB. Otherwise, for multiple Vdd/Vss cells P/G arc data must be provided in a custom LIB.

The 'pgarc' object can be defined at the design level in a custom LIB as follows:

```

pgarc {
    <vdd_pin_name> <vss_pin_name>
    ...
}

```

In this case the definition applies to all multiple power supply cells in the design. A pgarc can also be defined within a cell, in which case it is only applicable to that cell, as follows:

```

cell <cell_name> {
    pgarc {
        <vdd_pin_name> <vss_pin_name>
        ...
    }
}

```

P/G arc definitions in a custom library are not needed for:

- 1 power and N ground pins (N = 1 or more). If one power goes to multiple grounds, the current for the power pin is assumed to split evenly between the ground arcs. Note that APL captures the VDD currents, not the VSS currents.
- N power and 1 ground when all N power pins have a profile in the current file (or power in LIB)

P/G arc definitions in a custom library are needed for:

- N power and M ground pins (N > 1, M > 1)
- N power and 1 ground pin, very common in memory IP, with some power pins that have no profile in the current file (or missing power in LIB), so that VDD current can be assigned realistically

For example, assume that you have a library in which most cells have three power domains, VDD, VDD2 and VDDL, and one ground domain, VSS. However, one cell, ram_mvdd, has three power domains, VDD, VDD2 and VDDL, and two ground domains, VSS and VSS2. In the ram_mvdd cell both VDD and VDDL use VSS as ground, while VDD2 has VSS2 as ground. The primary design pgarc relationships do not require a

custom LIB file, since there is only one ground domain, but the pgarc relationships must be defined in a custom LIB file since the ram_mvdd cell also has two grounds as well as three power domains. See the example custom file format following:

```
pgarc {
    VDD VSS
    VDD2 VSS
    VDDL VSS
}
cell ram_mvdd {
    pgarc {
        VDD VSS
        VDD2 VSS2
        VDDL VSS
    }
}
```

Note that If there is a cell with p/g arcs that are not defined completely in a custom lib, RedHawk ignores the rest of the p/g arcs for this cell in LEF/LIB. If there are no arcs defined for this cell in a custom lib, RedHawk can read this information from LEF/LIB implicitly. However, implicit p/g arcs may not be correct due to the pin types in LEF or incomplete p/g arc information in LIB. So you must ensure that the p/g arcs defined in custom libs are complete and correct, since they have the highest priority and override the p/g arc information from LEF/ LIB. Moreover, custom libs also give you the flexibility to distribute current only to the power/ground pin(s) in which you are interested.

In the GSR file, the LIB_FILES keyword is required to specify the custom LIB filename (only one custom *.lib file may be specified):

```
LIB_FILES {
    <lib_filename> CUSTOM
    ...
}
```

If you are unsure as to which cells need to be specified in a custom library, then run the design through ‘setup design’ and look at the data in the file *adsRpt/apache.refCell.noPGArc*. RedHawk detects cells with multiple ground pins that have no custom LIB file and reports them in the report *adsRpt/apache.refCell.noPGArc*, along with all power and ground pins for each cell.

Example:

```
#<cell_name> <vdd_pin_names> <gnd_pin_names>
<SC_ANALOG> <VDD1A VDD2A> <VSS1A VSS2A>
```

State dependent leakage power

RedHawk can input a custom lib file with details on what all state/boolean combinations are to included/excluded from (state dependent) leakage power calculation.

For example, there are cases where state dependent leakage power for a certain state could be very high, especially for memories. This state would mostly be one, that is not used in regular modes. Custom lib settings can be given to not to consider this state's leakage power.

Syntax:

```
cell <cell_name> {
    exclude_states {
        pin <active_pin_name> {
            <boolean_expression1>
            <boolean_expression2>
        ..
    }
}
```

```
        }
        leakage_power {
            <boolean expression1>
            <boolean expression2>
            ..
        }
    }
    include_states {
        pin <active_pin_name> {
            <boolean expression1>
            <boolean expression2>
            ..
        }
        leakage_power {
            <boolean expression1>
            <boolean expression2>
            ..
        }
    }
}
```

Wildcards are accepted in boolean expressions.

GSR Keywords for Multi-Vdd Domain Designs

P/G arc errors

P/G arc errors can be suppressed using the IGNORE_PGARC_ERROR 1 keyword.

Block_Power_For_Scaling & Block_Power_For_Scaling_File keywords

The following keyword syntax should be used for handling cells with either single or multiple Vdd/Vss supplies in a Block_Power_for_Scaling definition file. However, in release 6.1 and before the 'fullchip <full_mVdd_inst_path>' option cannot be used in a GSR Block_Power_For_Scaling entry. See [section "BLOCK_POWER_FOR_SCALING", page C-624](#), for use of this keyword.

Syntax of BLOCK POWER FOR SCALING file:

```
{ CELLTYPE <cell_name> <cell_power_Watts> ? <Vdd_pin>?  
      ? <cell_current_A> <Vss_pin> ?  
...  
[FULLCHIP |<top_block_name>] <full_block_instance_path> <pwr_Watts>  
...  
fullchip <full_mvdd_inst_path> <power_Watts> ?<VDD_pin>?  
      ? <current_A> <Vss_pin> ?  
...  
}
```

API Requirements for Multi-Vdd Designs

Key elements of APL characterization for multiple Vdd/Vss domain cells are:

- No APL configuration file changes are required for Design Dependent characterization. RedHawk automatically generates the required files for mVdd characterization if the LEF/LIB files provided to RedHawk are complete.

- APL-DI requires a new keyword in re-characterization, LEF_FILES. For APL-DI, in the APL configuration file you must list the LEF files that cover all the multiple Vdd cells to be characterized, using the syntax:
`LEF_FILES {<fileA> ... <fileN> }`
- RedHawk uses the LEF files for deriving the P/G arc information. Current profiles and decap values are computed for the defined power/ground pin arcs in each cell.
Note: The *.current and *.cdev files are in binary format,
- Importing and merging mixed RedHawk release 5.x APL files are supported with some restrictions. The APL_FILES keyword structure in the GSR file is:

```
APL_FILES {
    <APL_binary_filename>    current
    <APL_binary_filename>    cdev
    <Avm.conf_filename>      avm
    <AVM_binary_filename>    current_avm
    <AVM_binary_filename>    cap_avm
    <APL binary filename>   pwcap
    ...
}
```

Note: In release 7.x <cell>.current files from releases 5.x and 6.x can be mixed on APL import and merge.

See [section "Importing APL Files", page 9-257](#), for more information.

AVM Configuration File

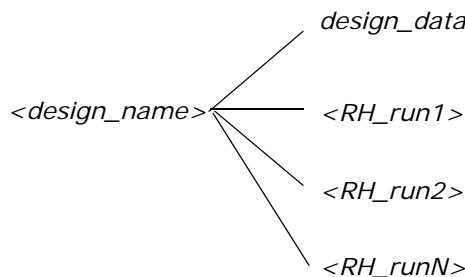
Specify the power/gnd pins for single or multiple-Vdd/Vss designs using the syntax described in detail in [section "AVM Configuration File", page 9-272](#).

Data Prep and Using the Automated 'rh_setup.pl' Script

This section describes how to specify the necessary input design files, create configuration and run-time command files to run RedHawk using the automated setup script, and also describes the recommended directory structure for input and working files.

Recommended RedHawk Directory Structure

For each design the following RedHawk directory structure is suggested:



The contents of the directories for each design are as follows:

`design_data/`: directory for all user-provided input data

`def/`: all DEF files for the design, with the extension `*.def`

Note: *.gz compressed files are also acceptable and are handled by RedHawk

lef/: all LEF files for the design, with the extension *.lef

lib/: all LIB files for the design, with the extension *.lib

tech/: RedHawk technology file

ploc/: voltage location, or pad cell file

The following *design_data/* sub-directories are optional:

timing/: timing data files (required for dynamic analysis)

spf/: full-chip or block-level SPEF/DSPF information

vcd/: VCD file

power/: instance-level power data

gds/: GDS file for all memory/macros in the design

gds2def/ or *gds2rh/*: directory for all GDS translation results files

apl/: directory containing APL/characterization data directories

memory/: directory for HSIM/Spice waveforms

avm/: directory for AVM results

<RH_runX>/: working directory

static/: static analysis results directory

setupApl/: directory in which to create APL data files

dynamic-dvd/: dynamic analysis results for vectorless method

dynamic-vcd/: dynamic analysis results for VCD input method

adsPower/: created by RedHawk for power calculation data

adsRpt/: created by RedHawk for Cmd, Error, Warn, and Log files in separate folders, as well as links to the latest file of each type

.apache/: created by RedHawk for holding working files for the run

Note: Working files in the *.apache* directory for any previous RedHawk run are deleted before starting a new run in the same run directory.

For best operation of the automated setup script the *design_data/* directory should have the structure shown above, but it is not mandatory. The setup script finds the needed files if the paths are specified.

The working directories *static/*, *dynamic-dvd/*, and *dynamic-vcd/* can be named as you prefer, but they should be at the same directory level as those in the *design_data/* directory.

Copy the required input files into the *design_data/* directory in preparation for setting up and running RedHawk.

Note: If you have a file named *.redhawkrc* in your home directory or in the current working directory that has one or more standard RedHawk TCL commands in it, RedHawk executes that file before it executes any other RedHawk command.

Automated Script *rh_setup.pl*

Setting up the Automated TCL Setup Script

In order to run Redhawk you must prepare an appropriate Global System Requirements (GSR) file. It is convenient to compile a command script to run each step desired. To automate these tasks, RedHawk provides the *rh_setup.pl* utility to assist in checking for the needed files, setting up the GSR file, and preparing a command file for running RedHawk the first time. The main features of using the setup script are:

- *rh_setup.pl* creates all the files needed to launch RedHawk simulation, either static, dynamic, or special evaluations, as follows:

- <design>.gsr - Global System Requirements keyword settings
- run_static.tcl - static run script
- run_dynamic.tcl - dynamic run script
- run_signalEM.tcl - run script for signal EM analysis
- run_lowpower.tcl - ramp-up run script for low power analysis
- Parameter values used in the generated GSR file can come from four sources, in the following priority, highest to lowest:
 - a. from rh_setup.pl command line parameters
 - b. from parameters of previous invocation of rh_setup.pl saved in rh_setup.init file
 - c. from template GSR file pointed by variable \$APACHEDA_TEMPLATE_GSR, if this file exists
 - d. from default values preset by RedHawk
- You don't need to remember the exact syntax of the GSR keywords. The GSR keywords generated with rh_setup will be syntactically correct.
- The script locates files automatically if the data directory structure used is as recommended (as described in the previous section). For example, if DEF files are found in the ../../data/def/*.def directory, they are all included in the generated GSR.
- The script supports UNIX wildcards in searching for required files in non-standard locations. For example, in using the file search pattern -def *.def */*.def */*/*.def, all DEF files in any directory up to two levels below will be found. If a required file is found with the same name in more than one directory, the file in the first directory specified is used.
- A central CAD engineering group can provide default values for specific RedHawk parameters (for example, the value of the desired dynamic simulation step) by means of a “template GSR” file. Use the global environmental variable \$APACHEDA_TEMPLATE_GSR to point to the template GSR file before running the setup script.
- Input to the script can be done incrementally. You can invoke the script with one or more command line arguments, and the script will prompt you if any of the required arguments are missing (for example, the power net name). You can re-invoke the script and add or change arguments, until all required information is complete. Then rh_setup.pl checks that all input files exist and creates the necessary files as listed above.

Using the rh_setup Utility

The procedure for using the setup script is as follows:

1. The script rh_setup.pl has the UNIX command line format:

```
rh_setup.pl <command_options> [<values>]  
...
```

2. When you execute the script command with no options

```
rh_setup.pl
```

definitions of the required options, parameter values and files are displayed, as well as those that are optional. These options are:

-top_cell - Required - name of the top cell in the design as specified in the corresponding DEF file (see DESIGN section of the DEF file).

-mode - the analysis mode, either 'early_analysis' or 'sign_off_analysis'. By default - 'sign_off_analysis'.

- analysis - the analysis type, ‘static’, ‘dynamic’, ‘low_power’, or ‘signalEM’. By default, TCL files for static and dynamic are created.
- vdd_nets - Required. Specifies Vdd net names and nominal voltage values
- vss_nets - Vss net names (no voltage specification). Optional input for static or signalEM runs, but required for dynamic or lowpower runs.
- frequency - Required. Defines the dominant operating frequency of the design (Hz), that is, the frequency that consumes a majority of the power in the design. If there is more than one dominant frequency, specify the lower frequency. See the GSR keyword description, [section "FREQUENCY", page C-614](#).
- tech_file - Required. Defines dielectrics, metal layer parameters and vias in the *tech*/directory. If not specified, the default path in *rh_setup.defaults* is searched.
- def_files - Optional if required files are in the *def*/directory. Specifies DEF files for top module, hierarchical and IP blocks.
- lef_files - Optional if required files are in the *lef*/directory. Specifies files for technology, library cells, hierarchical and IP blocks.
- lib_files - Optional if required files are in the *lib*/directory. Specifies Synopsys Liberty format (.lib) timing libraries. Files not needed for an early analysis or static runs, but required for dynamic, lowpower and sign-off runs.
- pad_files - Required. The automated flow picks files up from *ploc*/ directory, or you can specify the option ‘def_pins’ to use DEF pins as power pads for block level analysis, which forces RedHawk to place one pad in the center of every Vdd/Vss pin described in top-level DEF file. The RedHawk PLOC file provides x,y coordinates for ideal voltage sources. One of the pad files is required.
- sta_files - Optional if required files are in the *timing*/directory. Specifies timing files from STA, described in [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#). Files are required input for a sign_off_analysis run.
- spf_files - Optional if required files are in the *spf*/directory. Specifies hierarchical SPEF or DSPF files of post-layout parasitics.
- apl_files - RedHawk *cell.current* files containing current profiles from APL characterization. Required if the mode is ‘sign_off’ and the analysis type is ‘dynamic’ or ‘lowpower’.
- aplcap_files - RedHawk *cell.cdev* device capacitance file from APL characterization. Required if the mode is ‘sign_off’ and the analysis type is ‘dynamic’ or ‘lowpower’.
- aplpwcap_files - RedHawk *cell.pwcap* PWL decap file from APL power up characterization. Required if the mode is ‘sign_off’ and the analysis type is ‘lowpower’.
- gds_dirs - Optional if required files are in the *gds2def/gds2rh*/ directory. Specifies all LEF, DEF, and PRATION files created by *gds2def* or *gds2rh* in specified directories.
- toggle - Optional. The automated flow assigns a default value.
- input_transition - Optional. The automated flow assigns a default value.

Note: You may abbreviate command options, as long as they are unambiguous. For example, you may use the option ‘-top’ as an abbreviation for ‘-top_cell’.

3. Once the input data has been set up in the *design_data*/directory, the *rh_setup* command can be run anywhere, specifying the required values, such as the following example of input syntax:

```
rh_setup.pl -top_cell <name of the design>
              -vdd_nets <name of VDD net> <value>
              -vss_nets <names of VSS nets>
              -pad_files <name of pad/pcell file in ploc/ dir,
              OR specify def_pins for DEF pin-based analysis>
              -def_files <path and filenames>
              -lef_files <path and filenames>
              -lib_files <path and filenames>
              -tech_file <path and filenames>
```

Assuming all input information is found, this invocation creates the necessary run files:

4. The *rh_setup.pl* utility preserves all input values in a text file called *rh_setup.init* in the same directory in which the utility is invoked, so you can invoke the utility multiple times, each time providing one or more of the required arguments, until all the required data requirements are satisfied. At that point the run files are created. Also, you can edit the text file *rh_setup.init* file directly.
5. To execute the command scripts generated by *rh_setup.pl*, use the TCL command option -f, such as:

```
redhawk -f [ run_static.tcl | run_dynamic.tcl ]
```

Special GSR Keywords

Default values are maintained for some important GSR keywords:

- DYNAMIC_SIMULATION_TIME 2.56e9
- DYNAMIC_TIME_STEP 50ps - for an early_analysis run, and DYNAMIC_TIME_STEP 20ps - for sign_off run.
- INPUT_TRANSITION 200ps
- TOGGLE_RATE 0.3 (non-clock)

The following GSR keywords are taken into account only based on the analysis mode or kept commented in the GSR file.

- BLOCK_POWER_ASSIGNMENT - turned on during prototype analysis and prompts the user to edit their requirements
- SWITCH_MODEL_FILE - a file automatically generated by *aplsw* utility and added to the GSR section during a low power analysis run.

GDS2DEF Setup with the ‘gds_setup.pl’ Script

The **RedHawk gds_setup.pl** script helps automate the generation of necessary configuration files and running the *gds2def* extraction utility for multiple cells, in serial or parallel. The script accepts user data from two sources:

- the `\$cwd/gds_setup.init` file
- command line arguments, which are saved in *gds_setup.init* by the script

As with *rh_setup*, input to the script is incremental. First, the script examines LEF and GDS files to ensure that cells being extracted have both GDS and LEF descriptions.

To create the proper setup data you may invoke the script with one or more options, then re-invoke the script to add or change information at any time, until all required *gds2def* setup information is complete.

Using the gds_setup Utility

The procedure for using the setup script is as follows:

1. The script *gds_setup.pl* has the UNIX command line format:

```
gds_setup.pl <command_options> [<values>]
...

```

2. When you execute the script command with no options

```
gds_setup.pl
```

definitions of the required options, parameter values and files are displayed, as well as those that are optional. These options are:

-vdd_nets - Required. Specifies names of Vdd nets.

Example: -vdd_nets VDD vdda vddb, VDD_SOC

Segments labeled in GDS with 'vdda' and 'vddb' appear as 'VDD' in generated DEF file. 'VDD_SOC' is not renamed.

-vss_nets - Required. Specifies names of Vss nets.

Example: -vss_nets VSS vssa vssb, VSS_SOC

Segments labeled in GDS with 'vssa' and 'vssb' appear as 'VSS' in generated DEF file. 'VSS_SOC' is not renamed.

-cell_names - specifies cell names to be processed. When not specified, *gds2def* runs for cells for which the LEF and GDS data are provided.

-map_file - Required. Specifies the location of layer map file.

-lef_files - Required. Specifies individual LEF files, or specify *.lef to take in all LEF files.

Example: -lef_files ..//lef/hdsg1_10240x32cm16.lef ..//lef/

hdsg1_2048x40cm8bw.lef

-lef_files ../*.lef

-gds_files - Required. Specifies individual GDS files, or specify *.gds to take in all GDS files.

Example: -gds_files ..//gds/hdsg1_10240x32cm16.gds ..//gds/

hdsg1_2048x40cm8bw.gds

Example: -gds_files ..//gds/*.gds

-mem [on | off] - If set to 'on' indicates *gds2def* run is for a memory. Off by default.

-start_layer - specifies starting layer for extraction.

Example: -start_layer m1

-core_start_layer - specifies starting layer for core extraction.

Example: -core_start_layer m3

-options_file - specifies an input file containing a list of additional *gds2def* options to use.

-norun [on | off] - If set to 'On' creates only the config files and does not run *gds2def*. Off by default.

-bsub_run [on | off] - If set to 'on', submits the jobs to LSF farms for parallel execution on multiple machines. Off by default.

-bsub_command_file - specifies a file containing the 'bsub' command with additional options defining multiple machine execution requirements. If not specified, 'bsub' alone is used as the command.

3. Edit and update the values of the setup file options as required (see Figure 3-3, below), then run *gds2def*.

```
gds_setup.pl -vdd_nets VDD vdd vddx vddy, VDD_SOC \\
-vss_nets VSS gnd vss \\
```

```
-map_file gds_layer.map \\
-cell_names MEM22x64 MEM22x32 \\
-lef_files lef_dir/*.lef \\
-gds_files gds_dir/*.gds \\
-mem on \\
-norun on \\
-bsub_run on \\
-bsub_command_file bsub_options.txt
```

Figure 3-3 Sample gds_setup.pl file

Outputs

The 'OUTPUT' directory contains the DEF, LEF and PRATIO files generated from the *gds2def* runs, and the 'LOG' directory contains cell-specific log directories, which in turn contain the log files.

The 'top_report' file contains information on whether the cells have passed *gds2def* processing or not, as shown the example output in Figure 3-4:

```
hdsg1_10240x32cm16 Done
rfsg1_256x32cm4 Done
rfsg2_44x128cm2bw Fail
...
```

Figure 3-4 Sample 'top_report' file of gds2def results

Manually Importing Design Data

The previous section describes how to use scripts to prepare the input files to run **RedHawk**. If you do not want to use the scripts, design data and configuration files can be prepared manually using either the TCL command line or the GUI. These commands and procedures are described in the following sections.

Command Line Data Import

TCL commands for loading design data are shown in Table 3-4. Note that putting all design data into the GSR file and then importing the GSR is the recommended method for performing all design data input. See [section "Global System Requirements File \(*.gsr\)", page C-573](#), for more information about the contents and syntax of the GSR file.

Table 3-4 Commands for Loading Design Data

TCL	Purpose
import gsr <InputFileName>	Imports design information in the .gsr file.
import tech <InputFileName>	Imports .tech file. (Recommend using TECH_FILE keyword in GSR file.)
import lef <InputFileName>	Imports .lef file. (Recommend using LEF_FILES keyword in GSR file.)
import lib <InputFileName>	Imports .libs file. (Recommend using LIB_FILES keyword in GSR file.)

<code>import def <InputFileName></code>	Imports .def file. (Recommend using DEF_FILES keyword in GSR file.)
<code>import pad <InputFileName></code>	Imports pad cell .pcell, pad instance .pad, or pad location .ploc file. (Recommend using PAD_FILES keyword in GSR file.)
<code>import guiconf <InputFileName></code>	Imports GUI configuration file (optional)
<code>import eco <InputFileName></code>	Imports previous eco file (optional)
<code>setup design</code>	Sets up database after importing design data.
<code>import apl <InputFileName></code>	Imports <cell>.current file generated by APL (dynamic).
<code>import apl -c <InputFileName></code>	Imports <cell>.cdev file generated by APL (dynamic).
<code>import avm <configFileName></code>	Generates and imports memory model, vmemory.current and vmemory.cdev file from the config file.
<code>import db <InputFileName></code>	Imports the previously generated results after setup, extraction, or simulation.

You are now ready to load the technology and design information and run the **RedHawk** static IR drop analysis.

1. See [Appendix A, "Setting Up the RedHawk Environment"](#), for procedures for setting up license, environment and path variables for accessing the **RedHawk** binaries.
2. Type the following command in a UNIX shell:

```
redhawk
```

This command sends execution messages on the screen, while the following saves execution messages to *apache.log*:

```
redhawk >& apache.log&
```

The **RedHawk** GUI is displayed.

3. To prepare the design data using the command line, compile the required input files as described in the previous section and in [Appendix C, "File Definitions"](#).
4. If all needed input files are referenced in or included in the GSR file, then use:

```
import gsr <filename>
```

5. If all needed files are not referenced in the GSR, then use the TCL commands in the table above to import the required files, such as LEF, DEF, LIB, and *.tech.

6. After files are imported, to set up the design database, type:

```
setup design
```

The following data integrity checks insure **RedHawk** data are appropriate:

- Frequency not over 50GHz.
- Transition times not over 5ns.
- Simulation time or frame-size not over 200ns for dynamic voltage drop analysis, and not over 2us for power-up analysis.
- Pre-parsing of SPICE subcircuits makes sure there are no syntax or include errors, and no un-matched nodes with .ploc.

- e. Pre-parsing of switch models makes sure there are no syntax errors and values are reasonable.

GUI Data Loading

1. See [Appendix A, "Setting Up the RedHawk Environment"](#), for procedures for setting up license, environment and path variables for accessing the **RedHawk** binaries.
2. To prepare the design data using the GUI, first bring up the display by starting **RedHawk** from the command line:

```
redhawk
```

The GUI will be displayed.
3. Select the menu command **File -> Import Design Data**.
The 'Import Design Data' dialog box is displayed.
4. Click on **Select Filename** button for each input file required, and enter the path and filename in the dialog box that is displayed.
If all input files have the same path as the *.gsr file, you can click on the **Use Same Prefix** button at the bottom of the form to automatically enter the same path for all input files.
5. Complete the paths and filenames for all input files.
6. Make sure that the **Database setup** button is depressed.
7. Click on **OK** to load all files and set up the design database.
When loading the DEF files you can use either the VDD or VSS DEFs for faster VDD-only or VSS-only analysis.
8. After all files have been loaded, take a look at the layout view that appears.
 - Zoom in using the right mouse button. Change other view aspects with the 'View' buttons on the right side of the display.
 - Navigate using the Mini-view panel on the lower right side.
 - Turn layer view On/Off by using the **View Layers** button in the 'Configuration' panel on the right side.
 - Select design objects using the left mouse button. After selection, the object is highlighted in yellow.

To continue the analysis procedure, see [Chapter 4, "Power Calculation, Static IR Drop and EM Analysis"](#).

RedHawk Configuration Files

In addition to input data files, some **RedHawk** functions and utilities have required configuration files that must be prepared, as listed in Table 3-5.

Table 3-5 RedHawk Configuration Files

Filename	Purpose
<i>apl_config_file</i>	Used by Apache Power Library (APL) program to characterize cells, decaps, I/Os, and memories for dynamic voltage drop analysis. See Chapter 9, APL Characterization.

<i>switch_config_file</i>	Used by <i>aplsw</i> utility to characterize header/footer switches and/or generate piecewise linear current models for switches used in low power designs. See Chapter 13, Low Power Analysis.
<i>avm_config_file</i>	Used by <i>avm</i> (<i>import avm</i>) utility to generate switching current profile for memories.
<i>sim2iprof_config_file</i>	Used by the <i>sim2iprof</i> utility to extract Read/Write/Standby signals from third-party simulation output and to generate current profile waveforms. See Appendix E.
<i>gds2rh_config_file/</i>	Used by <i>gds2rh</i> utility to convert GDSII file to DEF format. See Appendix E. Used by <i>gds2rh -m</i> utility to generate detailed view of memory and other IP. See Appendix E.
<i>psi.ctl</i>	The PJX configuration and control file supports the PJX timing and clock tree analysis program. See Chapter 8.

Chapter 4

Power Calculation, Static IR Drop and EM Analysis

Introduction

This chapter describes how to run power calculation, static IR voltage drop and electromigration (EM) analyses in **RedHawk-S**, following the data preparation steps described in the Chapter 3.

An outline of the static IR drop and EM analysis flow is shown in Figure 4-1.

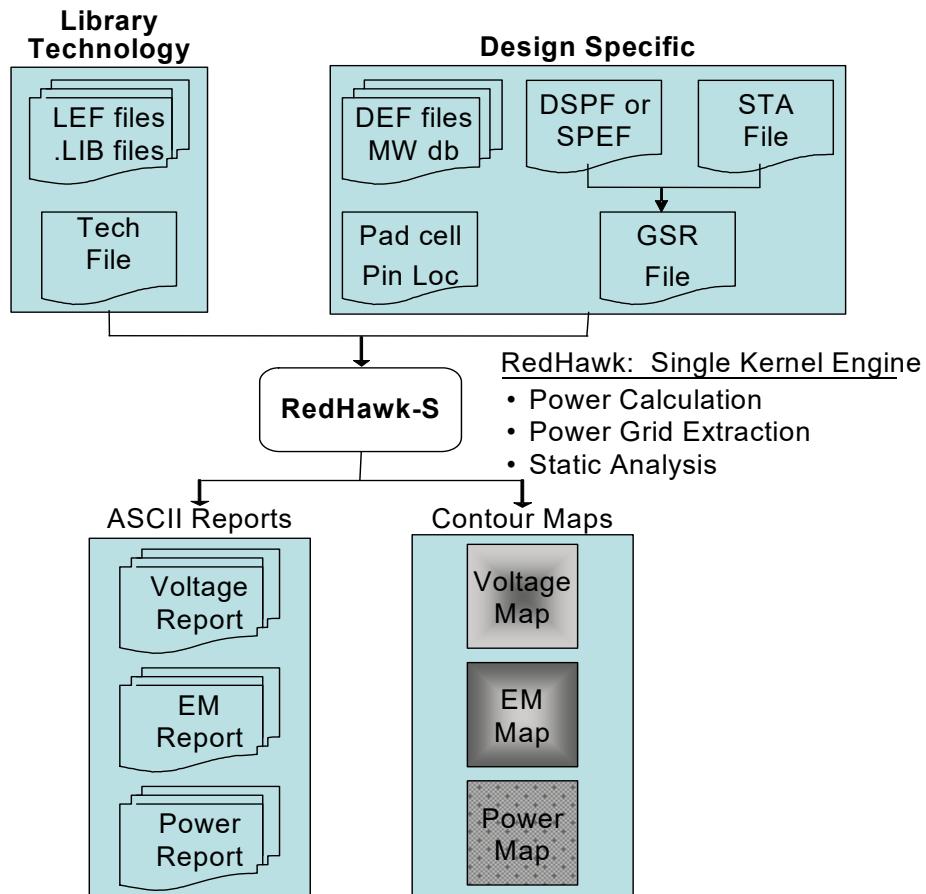


Figure 4-1 RedHawk-S (static) IR drop flow

The following are the key steps in the static IR drop and EM analysis flow.

1. Prepare design data files (see [Chapter 3, "User Interface and Data Preparation"](#)).
2. Import design data using the automated setup script or the GSR file (see Chapter 3)
3. Perform power calculation (see following section).

4. Perform power grid extraction for R network (see [section "Power Grid Resistance Extraction", page 4-52](#)).
5. Evaluate power/ground grid weakness (see [section "Examining Power/Ground Grid Weakness", page 4-53](#))
6. Define pad and package constraints (see [section "Defining Pad and Package Parameters", page 4-56](#)).
7. Perform static IR voltage drop and EM analysis (see [section "Running RedHawk-S \(Static IR/EM Analysis\)", page 4-57](#)).
8. Review static IR/EM summary reports and evaluate what other information is needed from the analysis.
9. Explore solutions to reduce excessive static IR drop with the **RedHawk** power grid Fixing and Optimization tools (see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#)).

Steps 3 through 8 are described in the following sections. After importing all design data, the full chip view should be displayed. Continue with the analysis procedure in the following sections.

Power Calculation

Power calculation uses information from a number of design files to evaluate the average cycle power consumption of all cell instances for both flat and hierarchical designs. The calculated power is used for both static and dynamic analysis. It then displays the power distribution results in a graphical “map” of the design. Power calculation is typically performed *once for each design*, and the results are summarized in several power reports. The power reported by power calculation is “ideal” power, based on the supply voltage specified in the VDD_NETS keyword in the GSR, or otherwise the nominal voltage specified in the LIB file. RedHawk honors .lib negative internal energy values for power calculation. The computed total power also includes power due to instances not connected to power nets or instance power pins connected to non-Vdd nets, hence the actual power may be different due to instances not connected in the design.

The setup procedure for power calculation involves providing the best design information available to **RedHawk**, in order to get the most accurate results. Three methods are available for modeling chip activity and setting up power calculation:

- vectorless, when no VCD file is available, and toggle-based state propagation methods are used.
- event-propagation, using full chip VCD file data

Power calculation is propagated through flip-flops in both State Propagation and Event Propagation methodology, which is enabled in both vectorless and RTL-VCD flows.

An outline of how to select the power calculation method based on the input data available is presented in Figure 4-2. The two primary propagation flows are shown in Figure 4-3.

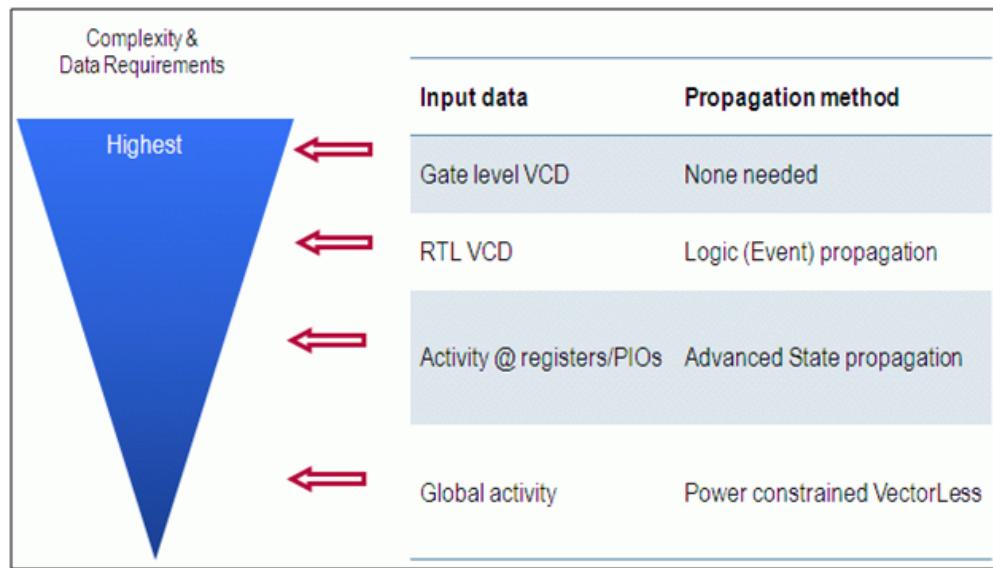


Figure 4-2 Selection of power calculation method

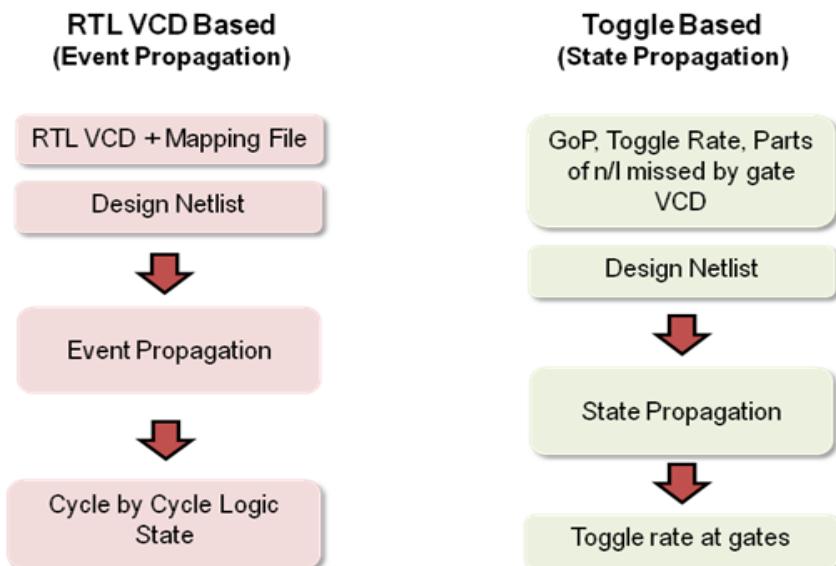


Figure 4-3 Event propagation and State propagation flows

The setup procedures and information needed by RedHawk for the different types of activity modeling are described in the following sections.

Setup for Vectorless Power Calculation

When no VCD file is available, an estimate of the toggle rate should be made, using values from a group of GSR keywords, as described in the following paragraphs.

Setting Frequency and Clock Parameters

Use the following GSR keywords to specify the primary design frequency and clocks.

- **STA_FILE**

If this keyword is defined, the power for each net is calculated from the transition times (“slew”) of each instance’s input/output pins, using the clock frequency domain for all the instances that are specified in the STA output file, instead of deriving its value from the **CLOCK_ROOTS** keyword.

For instances not defined in the STA output file or VCD file, the frequency of the instance is set to zero. Use the frequency specified by the **FREQ_OF_MISSING_INSTANCES** option in **STA_FILE** keyword for those instances that are not covered in the STA file. **FREQ_OF_MISSING_INSTANCES** is not required to be specified unless you know that STA does not cover the design well. If the **EXTRACT_CLOCK_NETWORK** option is specified as 1 (default 0), clockroots defined in the clock section of the STA output file are used for clockroot tracing, instead of the STA-identified clock instances.

See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#) for details on how to generate the STA output file. *Required for dynamic analysis; Optional for static analysis.*

Syntax:

```
STA_FILE
{
    FREQ_OF_MISSING_INSTANCES <frequency in hertz>
    EXTRACT_CLOCK_NETWORK [ 0 | 1 ]
    <top_level_block_name> <sta_output_file>
}
```

where

FREQ_OF_MISSING_INSTANCES <frequency in hertz> : specifies the operating frequency for instances not in the STA file but in the design. Default: 0.

EXTRACT_CLOCK_NETWORK : when set to 1, specifies that the clock network is extracted from clock roots defined in STA

<top_level_block_name>: the top level block name of the chip

<sta_output_file> : the file generated from STA analysis. See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#)

- **CLOCK_ROOTS**

Traces the nets from a specified clock root and finds the respective clock domain for all the nets and instances. Specifies the names of the clock roots (either net name or pin name) and the frequency. No wildcard matching is supported for this keyword. If **STA_FILE** is specified, **CLOCK_ROOTS** is not used. *Optional. Default: None.*

Syntax:

```
CLOCK_ROOTS
{
    <clock_root_name> <freq in Hz>
    ...
}
```

- **FREQUENCY**

Defines the dominant operating frequency on the chip, or the lowest frequency that includes a majority of the power consumption on the chip. More specifically, for a design in which there are several frequencies that consume significant power, the

frequency to be specified is the frequency for which less than 10% of the chip power is consumed at *lower* frequencies. For example, assume that there are three significant frequencies on the chip, and they consume the following power: 100 MHz (5mw), 200 MHz (20mw), and 400 Mz (70mw). The FREQUENCY value to be specified in this case would be 200 MHz, even though a majority of the power is consumed at 400 MHz. *Required. Default: none.*

Syntax:

```
FREQUENCY <value in Hertz>
```

Setting the Switching State of Instances and Blocks

Using the Global Switching Configuration (GSC) file, appropriate switching states for instances and blocks can be set, such as HIGH, LOW, STANDBY, DISABLE and OFF, to achieve more controlled power calculation. Using the GSC_FILE to establish switching states for power calculation is described below.

The GSC filename is specified using the GSR keyword 'GSC_FILE <gsc_filename>'.

The syntax of the GSC file is:

```
[<blockName> | <instanceName> ] ? <domain_name>?  
[ UNDECIDED | TOGGLE | HIGH | LOW | POWERUP | POWERDOWN |  
STANDBY | DISABLE | OFF | <custom_state_name> ]  
...
```

Refer to [section "Global Switching Configuration \(GSC\) File", page C-571](#), for details on the syntax and use of the GSC file.

Note that only HIGH, LOW, TOGGLE, STANDBY, DISABLE, and OFF states affect power calculation results.

If an instance has a power specified using the GSR keywords INSTANCE_POWER_FILE or BLOCK_POWER_FOR_SCALING, RedHawk uses these power values, even if it has been assigned a switching mode in GSC_FILE. This can be overridden by setting 'GSC_OVERRIDE_IPF 1' in the GSR (the default value is 0).

Setting the Toggle Rate

The toggle rate for an instance is defined as the average sum of the state changes from 0 ->1 and 1 ->0 within a clock cycle, with respect to the net's clock domain. For the most accurate power analysis you must compile and input the best toggle rate information available to indicate the average switching performance of the instances in the design. The general priorities for setting toggle rates are listed below, from highest to lowest. However, because there are interactions between values chosen for each keyword in a specific design, see the individual keyword descriptions following, or in Appendix C, for more information about toggle rate specification and instance toggle rate estimation. The priorities for toggle rate setting are (GSR keywords, unless otherwise noted; see [section "Global System Requirements File \(*.gsr\)", page C-573](#), for full keyword syntax):

1. INSTANCE_POWER_FILE
2. BLOCK_POWER_FOR_SCALING or BLOCK_POWER_FOR_SCALING_FILE
3. GSC_FILE

If the GSR keyword GSC_OVERRIDE_IPF is set to 1 and there is no VCD setting, the GSC_FILE has priority over IPF and BPFS settings. Also, if a VCD file setting exists, the GSC value overrides the VCD for any instances specified.

4. INSTANCE_TOGGLE_RATE or INSTANCE_TOGGLE_RATE_FILE

With the GSR key word ITR_OVERRIDE_BPFS set to 1, INSTANCE_TOGGLE_RATE will have higher priority than BLOCK_POWER_FOR_SCALING.

5. NET_TOGGLE_RATE or NET_TOGGLE_RATE_FILE or SAIF_FILE
6. VCD_FILE

With the GSR key word POWER_VCD_OVERRIDE_IPF set to 1, VCD will have higher priority than INSTANCE_POWER_FILE.
7. BLOCK_TOGGLE_RATE, BLOCK_TOGGLE_RATE_FILE, or CELL_TOGGLE_RATE
8. STATE_PROPAGATION
 - a. VCD-based STATE_PROPAGATION starts from the toggle rate derived from the VCD file.
 - b. Vectorless state propagation uses the toggle rate derived from evaluating parameters in items 3-7 and 9-11.
 - c. For details on use of the STATE_PROPAGATION keyword, see [section "STATE_PROPAGATION", page C-649](#).
9. A signal specified as 'CONST <pin>' in the STA_FILE has a toggle rate of 0.
10. CLOCK_DOMAIN_TOGGLE_RATE
11. POWER_DOMAIN_TOGGLE_RATE
12. TOGGLE_RATE_RATIO_COMB_FF <ratio>
13. TOGGLE_RATE

In general the highest priority keyword that has a toggle rate value for an instance is used in power calculation, and all other values are ignored.

Note that any of the keyword-estimated toggle rate values are ignored when they are specified in a VCD file, as described in the next section on setting up for VCD power calculation.

- INSTANCE_TOGGLE_RATE_FILE

Specifies the instance toggle rate file, which provides toggle rates for instances on the chip. The format of the file is as described in the INSTANCE_TOGGLE_RATE keyword syntax. *Optional. Default: None.*

Syntax:

```
INSTANCE_TOGGLE_RATE_FILE
{
  <instance_toggle_rate_file>
}
```

- INSTANCE_TOGGLE_RATE

Specifies average toggle rates for instances in the design. If there are a lot of instances in the chip, using this keyword is recommended, rather than using BLOCK_TOGGLE_RATE or BLOCK_TOGGLE_RATE_FILE keywords. No wildcard (*) is supported. Whether a clock network instance or not, the first TR entry applies to the network toggle rate, and the second, if present, applies to the clock buffer and clock pin toggle rate. If only one TR value is specified, it is used for the output/signal toggle rates associated with the instance. *Optional. Default: None.*

Syntax:

```
INSTANCE_TOGGLE_RATE
{
  <non-clock_inst_name> <output_TR> ?<clock_pin_TR>?
  <clock_network_inst_name> <output_TR> ?<clock_pin_TR>?
  ...
}
```

```
}
```

- **BLOCK_TOGGLE_RATE_FILE**

Specifies the file containing toggle rates for blocks/instances that are defined in the DEF file. The format of the file is as described in the BLOCK_TOGGLE_RATE keyword syntax. *Optional. Default: None.*

Syntax:

```
BLOCK_TOGGLE_RATE_FILE
{
    <toggle_rate_filename>
}
```

- **BLOCK_TOGGLE_RATE**

Defines the default toggle rate for nets in a user-specified block, or instances that are not otherwise specified. The block or instances should be defined in the DEF files. The <block-instance_name> specification can take a wildcard ‘*’. For example, ABC* matches all block/instance names starting with ABC. The <clock_network_TR> entry for a block applies to clock pins and clock buffer outputs. For an instance, whether a clock network instance or not, the first TR entry applies to the network toggle rate, and the second, if present, applies to the clock buffer and clock pin toggle rate. *Optional. Default: None.*

Syntax:

```
BLOCK_TOGGLE_RATE
{
    <block-instance_name> <non-clock_TR> ?<clock_network_TR>?
    <non-clock_inst_name> <output_TR> ?<clock_pin_TR>?
    <clock_network_inst_name> <output_TR> ?<clock_pin_TR>?

    ...
}
```

- **TOGGLE_RATE**

Specifies the default toggle rate for nets in the design that are not otherwise specified. The rate is the product of the probability that the nets will toggle times the actual clock toggle rate. Toggle rate is defined as the average sum of the state changes from 0->1 and 1->0 within a clock cycle with respect to the net's clock domain. For example, a clock net has a toggle rate of 2.0 with respect to its clock domain, since the net switches once from 0->1 and once 1->0 within a clock cycle. Note that if there is no power consumption table in .lib the toggle rate is taken from TOGGLE_RATE, and charge is scaled to meet power. *Optional. Default: 0.3.*

Syntax:

```
TOGGLE_RATE <non_clock_TR> ?<clock_network_TR>?
```

where

<non_clock_TR> : defines the probability for nets switching during a clock cycle
<clock_network_TR>: applies to both clock pins and to clock buffer outputs-- the actual network clock toggle rate (Default: 2.0)

Setting Toggle Rate Scaling Parameters

Since the power consumption of any part of the design is a linear function of the average toggle rate, scaling parameters allow RedHawk to adjust the effective toggle rates based on known and calculated power values. For example, if a block has known power consumption P_K , and the calculated power based on an estimated toggle rate TR_{E1} is

P_{C1} , then the program can scale the estimated toggle rate to T_{Re2} and recalculate power P_{C2} so that it is equivalent to the known power consumption, as shown in the following equations:

$$T_{Re2} / T_{Re1} = P_{C2}/P_{C1} \quad (\text{EQ } 1)$$

$$\text{we want } P_{C2} = P_K \quad (\text{EQ } 2)$$

$$\text{therefore, set } T_{Re2} = T_{Re1} * P_K / P_{C1} \quad (\text{EQ } 3)$$

So by providing known power consumption values for individual blocks and instances **RedHawk** can scale their toggle rates accordingly for analysis and make a much more accurate power calculation.

The power scaling keywords are described in the following paragraphs.

- **BLOCK_POWER_FOR_SCALING_FILE** (GSR Keyword)

Specifies the absolute or relative path from **RedHawk** run directory that contains the power specification, as in **BLOCK_POWER_FOR_SCALING** keyword. *Optional; default: None*

Syntax:

```
BLOCK_POWER_FOR_SCALING_FILE
{
    <Block_Power_FilePathName>
}
```

- **BLOCK_POWER_FOR_SCALING** (GSR Keyword)

Performs scaling of the estimated toggle rate for the block, instance, cell, or celltype, based on user-supplied known power data. Top-level block is specified for a flat design. Note that to avoid confusion a top cell name should not be the same as an included cell name. Pin-specific Vdd power and Vss current can be specified for multiple Vdd/Vss designs. *Optional. Default: none.*

Syntax:

```
BLOCK_POWER_FOR_SCALING {
    CELLTYPE <cell_name> [<pwr_W> ?<Vdd_pin>? | <cell_current_A> ?<Vss_pin> ?]
    ...
    [FULLCHIP |<top_block_name>] <full_leaf_inst_path> <pwr_W>
    [FULLCHIP |<top_block_name>] <full_block_path> <pwr_W>
    ? <domain> ?
    ...
    [FULLCHIP |<top_block_name>] <full_mVdd_inst_path>
        [<pwr_W> ?<VDD_pin>? | <current_A> ?<Vss_pin>?]
    ...
    CELLS [ comb | ff_latch | mem | clockinst | io ] <pwr_W>
    ...
    BLOCK [<full_block_path> <pwr_W> ?<domain>? | <full_leaf_inst_path> <pwr_W> ? <pin> ?]
    ...
}
```

See [section "BLOCK_POWER_FOR_SCALING", page C-624](#), for more detailed information on using this keyword.

- **SCALE_CLOCK_POWER** (GSR Keyword)

Selects scaling of toggle rate for clock network components. If set to zero, clock nets are not scaled. *Optional. Default: 1.*

Syntax:

```
SCALE_CLOCK_POWER [0 | 1]
```

- INSTANCE_POWER_FILE (GSR Keyword)

Defines absolute or relative path from RedHawk run directory to the power file, which contains a list of instances and their power consumption, in the following format: <instance_name> <power in Watts>. For instances not specified in the power file, the power is assumed to be zero. Therefore all significant power consumers should be specified, or none. *Optional; default: None.*

Syntax:

```
INSTANCE_POWER_FILE
{
    <instance_power_file_path-name>
}
```

- GSC_OVERRIDE_IPF (GSR Keyword)

When set to 1, overrides settings of BLOCK_POWER_FOR_SCALING and BLOCK_POWER_FOR_SCALING_FILE, INSTANCE_TOGGLE_RATE, BLOCK_TOGGLE_RATE, and TOGGLE_RATE, and the toggle rate value set in the GSC is honored. *Optional. Default: 0.*

Syntax:

```
GSC_OVERRIDE_IPF [0 | 1]
```

- Global Switching Configuration (GSC) File

Instances that have states defined in the GSC file are handled by power calculation in the following ways:

High – considers power when the instance switches from 0 to 1

Low – considers power when the instance switches from 1 to 0

Standby – averages power when a clock pin switches from 0 to 1 or 1 to 0

Disable – considers leakage power only

Off – sets all power to zero

Using both clock and signal toggle rates in power calculation

By default both the signal toggle rate and clock toggle rate are scaled uniformly to reach specified total power consumption; if you want to meet power targets by scaling the signal toggle rate only, set SCALE_CLOCK_POWER to 0 in the GSR. (Clock power includes both clock network power and clock pin power.) However, the power is not scaled to a value below specified leakage power.

If the specified block power is smaller than the RedHawk computed block leakage power, and SCALE_CLOCK_POWER is set to 1 (default), the power values are scaled uniformly to meet the specified block power.

Setting Extraction Parameters

The following keywords set parameters for extraction.

- CELL_RC_FILE

Defines the SPEF/DSPF interconnect parasitics file for a flat or hierarchical design. For extraction of detailed RC used in dynamic voltage drop analysis, set the suboption EXTRACT_RC to 1, which is its default. Otherwise, for static IR-drop analysis, optionally set EXTRACT_RC to 0. The CONDITION keyword allows selection of one of the capacitance value types from a three-value SPEF file.

Optional. Defaults: EXTRACT_RC: 1; CONDITION: typical.

Syntax:

```
CELL_RC_FILE
{
    EXTRACT_RC [ 0 | 1 ];
    CONDITION [best | typical | worst ]
    <cell_name> <path to DSPF-SPEF_file>
    ...
}
```

- INTERCONNECT_GATE_CAP_RATIO

Defines the ratio of the total interconnect capacitance of the nets relative to the total gate capacitance of the input pin fanouts. If none of the following keywords, INTERCONNECT_GATE_CAP_RATIO, CELL_RC_FILE, or STEINER_TREE_CAP have specified values, power calculation uses the default value of INTERCONNECT_GATE_CAP_RATIO. *Optional. Default: 1.*

Syntax:

```
INTERCONNECT_GATE_CAP_RATIO <value>
```

Example:

```
INTERCONNECT_GATE_CAP_RATIO 1.5
```

- STEINER_TREE_CAP

When specified, a Steiner Tree routing is performed and the resulting length is multiplied by the cap value specified in pF per um. *Optional. Default: none.*

Syntax:

```
STEINER_TREE_CAP <cap in pF per um>
```

Selecting Power Calculation Methodology

The following GSR keywords set parameters for power calculation.

- SCANMODE

Specifies whether scan pin power is included in vectorless power calculation or not. *Optional. Default: 0.*

Syntax:

```
SCANMODE[ 0 | 1 ]
```

- POWER_MODE

Specifies the primary data source for internal/switching power and leakage power calculation analysis. *Optional. Default: Mixed.*

Syntax:

```
POWER_MODE [ APL | LIB | MIXED | APL_PEAK | APL_PEAK1 ]
```

where

APL : specifies primary use of APL power data for internal/switching (*cell.ifprof*) and leakage power (*<cell>.cdev*). Where APL data are not available, *.lib* data are used.

LIB : specifies use of *.lib* power consumption data; cells without power data in *.lib* have no internal power consumption data, but have switching power information from RedHawk.

MIXED : specifies primary use of .lib power data; for cells without power components (internal power consumption or leakage power) in .lib, APL data are used.

APL_PEAK: uses the peak charge from APL to calculate the power for every cell in the design, and the current is derived from the charge.

APL_PEAK1: uses the peak current from APL for every cell in the design to calculate power. Peak current leads to a very pessimistic power calculation if that type of model is desired.

- INPUT_TRANSITION

Defines the input transition time for input pins of all instances not specified in an STA file. *Optional. Default: 10% of the value of the inverse of the frequency defined by the “FREQ” keyword.*

Syntax:

```
INPUT_TRANSITION <value in second>
```

Example:

```
INPUT_TRANSITION 0.2e-9 (or 0.2ns)
```

- NAME_CASE_SENSITIVE

Defines the name case sensitivity. If the value is set to 1, all .lib, lef/def, spef/dspf, vcd, and STA filenames are assumed to be case-sensitive. *Optional. Default: 1.*

Syntax:

```
NAME_CASE_SENSITIVE [ 0 | 1 ]
```

Example:

```
NAME_CASE_SENSITIVE 0
```

Specifying Supply Nets

The following keywords define the power supply nets and their voltages.

- VDD_NETS

Specifies the voltage for Vdd nets. The power domain names are defined in the DEF file using the SPECIALNETS keyword. *Required. Default: none.*

Syntax:

```
VDD_NETS
{
    <Vdd net_name in DEF> <value in Volt>
    ...
}
```

where

<Vdd net_name in def> : specifies the name for a power net, such as VDD for the core power domain, and VDDQ for the I/O power ring.

- GND_NETS

Specifies the voltage for Vss nets. The power/ground domain names are defined in the DEF file using the SPECIALNETS keyword. *Optional. Default: all SPECIALNETS are designated as USE GROUND and set to 0 volts.*

Syntax:

```
GND_NETS
{
    <Gnd net_name_in_DEF> <Volts>
```

```
    ...  
}
```

Setting Bus and Hierarchy Delimiter Keywords

The following keywords provide name mapping functions by defining delimiter characters for busses and design hierarchy.

- **BUS_DELIMITER**

Defines the character used to delimit bus bits in the RedHawk database and GSR.
Optional. Default: []

Syntax:

```
BUS_DELIMITER <delim>
```

- **PIN_DELIMITER**

Defines a special character to separate net or instance names from pin names in RedHawk working files. Any character can be used that is not reserved in LEF or used in pin names. *Optional. Default: :*

Syntax:

```
PIN_DELIMITER <delim>
```

- **HIER_DIVIDER**

Defines the character used to specify the hierarchy in the RedHawk database and GSR. *Optional. Default: /*

Syntax:

```
HIER_DIVIDER <divider>
```

- **BUS_DELIMITER_STA**

Defines the character used to delimit the bus bits in STA files. *Optional. Default: []*

Syntax:

```
BUS_DELIMITER_STA <delim>
```

- **PIN_DELIMITER_STA**

Defines the character between instance and pin names in STA files. *Optional. Default: /*

Syntax:

```
PIN_DELIMITER_STA <delim>
```

- **HIER_DIVIDER_STA**

Defines the character used to specify the hierarchy in STA files. *Optional. Default: /*

Syntax:

```
HIER_DIVIDER_STA <divider>
```

Setting up for Event-Driven (VCD File) Power Calculation

If you have a VCD file for the design it is strongly recommended that you perform power calculation based on VCD data, as described in this section.

The following GSR keywords described in the previous section for vectorless power calculation also need to be defined when using a VCD file. All keywords are used the same for both vectorless and event-driven calculation, so their descriptions are not repeated in this section:

- STA_FILE
- CLOCK_ROOTS
- BLOCK_POWER_FOR_SCALING_FILE
- BLOCK_POWER_FOR_SCALING
- SCALE_CLOCK_POWER
- CELL_RC_FILE
- INTERCONNECT_GATE_CAP_RATIO
- STEINER_TREE_CAP
- SCANMODE
- POWER_MODE
- INPUT_TRANSITION
- NAME_CASE_SENSITIVE
- VDD_NETS
- GND_NETS
- Bus and Hierarchy Delimiter Keywords

Setting GSR Keywords for Event-driven (VCD) Power Calculation

General Setup

The following keyword is used to specify VCD-based power calculation. See [section "vcdscan", page E-871](#) and [section "fsdbtrans", page E-872](#), for information on VCD-based utilities.

- VCD_FILE

The VCD_FILE keyword reads in original VCD files directly for power calculation purpose. Note that instance switching specified in the GSC file overrides the VCD file. See [section "VCD_FILE", page C-609](#) for more details on this keyword.

Optional. Default: None.

Syntax:

```
VCD_FILE
{
    <top_block_name> <VCD filepath>
        FILE_TYPE [ VCD | FSDB ]
        FRONT_PATH <"string">
        SUBSTITUTE_PATH <"string">
        FRAME_SIZE <integer value in ps>
        START_TIME <integer value in ps>
        END_TIME <integer value in ps>
        TRUE_TIME [ 0 | 1 ]
        MAPPING_RULE_FILE <file_name>
}
```

where

<top_block_name> <VCD file>

FILE_TYPE [VCD | FSDB]

FRONT_PATH <"string">: specifies the string that needs to be replaced by

SUBSTITUTE_PATH <"string"> to match the DEF hierarchy.

FRAME_SIZE <value_ps> : specifies the duration per frame for cycle-by-cycle power calculation.

START_TIME and END_TIME <value_ps> : optional, specifies the start and end times in the VCD for power calculation.

TRUE_TIME : if =0, uses STA timing data and assumes no glitches; if =1, uses VCD switching and timing data; default=0.

MAPPING_RULE_FILE : points to file that contain list of rules for name mapping between VCD and DEF names. This is used in RTL VCD based analysis where there will be unfamiliar changes between vcd/netlist names. (see following section)

Name Mapping using MAPPING_RULE_FILE

The following are various rules used in MAPPING_RULE_FILE for name mapping between VCD and DEF names:

- change_gate_name <string1> <string2>

'string1' is replaced with 'string2' for def (gate level) names when attempting to map as in the below example:

```
change_gate_name ricc icc
```

if you want to remove a particular string (say, 'extra') fully from a name, you can specify as:

```
change_gate_name extra ""
```

- change_rtl_name <string1> <string2>

'string1' is replaced with 'string2' for vcd (rtl level) names when attempting to map Example is:

```
change_rtl_name gen_pqr_i(%d) pqr_ixx%dx/
```

- define_bus_rule <rtl_bus> <gate_transform> -

generally buses defined in rtl of form string[%d] (say, cnt[13]) appear in gate netlist as string_reg_%d (cnt_reg_13), which is what the algorithm looks to map, by default. But there can be situations when the synthesis tool uses some other bus rule conventions to give names corresponding to each bus in the gate netlist. Here define_bus_rule can help. Please note that the output pin name also has to be specified along with gate name. Example is:

```
define_bus_rule [%d] %d__regxx/_Q
```

- skip_gate_element <string>

This option is used when one knows that a particular name in GATE level netlist/ DEF will never appear in RTL VCD, and doesn't need to be mapped. It is aimed at improving the runtime for the algorithm. Example is:

```
skip_gate_element top/block_name
```

Please note that only names that start with the string are skipped.

- generate_label <string> :

There are generate constructs in rtl which get synthesized in a very specific way when the netlist is generated. For example :

```
rtl name : top/label[2]/block/cnt[2]
gate name : top/block2/cnt[2]
```

Notice how the string 'label' is gone and the number along with it moves to the next hierarchy. Using the rule generate_label, we can give this particular string (label) and RedHawk will take care of the rest of mapping part. So, here, specifying 'generate_label label' will ensure proper mapping.

Wildcards '%d' are also supported in rules 'change_gate_name' and 'change_rtl_name' where %d denotes a number.

Example for wildcard usage:

If RTL and GATE names are as follows :

```

RTL : blockA/case_move_gen[0]/abc_move/rise_edge
GATE : blockA/case_movexx0x/rise_edge_reg
RTL : level_three_6/abc/abc_ip/gen_pqr_i(1)/pqr_i/gen_xyz_i(1)/xyz_i/
      xyz_defg_ctrl_reg[14]
GATE : level_three_6/abc/abc_ip/pqr_ixx1x/xyz_ixx1x/
      xyz_defg_ctrl_reg_regx14x
RTL : level_three_6/abc/abc_ip/gen_pqr_i(2)/pqr_i/gen_xyz_i(2)/xyz_i/
      xyz_defg_ctrl_reg[14]
GATE : level_three_6/abc/abc_ip/pqr_ixx2x/xyz_ixx2x/
      xyz_defg_ctrl_reg_regx14x

```

These can be mapped easily by using the following rule:

```

change_rtl_name case_move_gen[%d]/abc_move/ case_movexx%dx/
change_rtl_name gen_pqr_i(%d)/pqr_i/ pqr_ixx%dx/
change_rtl_name gen_xyz_i(%d)/xyz_i/ xyz_ixx%dx/

```

MBFF Name Mapping using MAPPING_RULE_FILE

In RTL-VCD based analysis, the presence of Multi Bit Flip Flops in the design can cause limitations in mapping RTL-VCD names to gate level names. Multi Bit Flip Flops have multiple inputs/outputs and are used to instantiate multiple bits of a register. Typically there are 20-30 D pins and Q pins in an MBFF. Generally the name of MBFF instance has all the individual register names concatenated. Mostly, there are delimiters/prefix names that identify MBFFs. Mapping of flip-flops with pin name o, o1, o2.. etc is supported. In scan mode, user can shift the input values among all output pins (Q0->Q1->Q2->Q3).

RedHawk internal mapping can take care of Multi Bit Flip Flops (MBFF) naming issues if the MBFFs are specified in a GSR mapping rule file option.

For example, for a 2-bit MBFF:

```

inst: xxx/MBIT__cp15_st_reg_1_MB_cp15_st_reg_2_
pin:Q0, mapped to RTL Name : xxx/cp15_st[1]
pin:Q1, mapped to RTL Name : xxx/cp15_st[2]

```

Note that here “MBIT_” is the prefix and “MB_” is the delimiter.

For a 4-bit MBFF:

```

MBIT_cp_addr_reg_reg_4_MB_cp_addr_reg_reg_5_MB_cp_addr_reg_reg_6_MB_cp_addr_
_reg_reg_7_
pin:Q0, mapped to xxx/cp_addr_reg[4]
pin:Q1, mapped to xxx/cp_addr_reg[5]
pin:Q2, mapped to xxx/cp_addr_reg[6]
pin:Q3, mapped to xxx/cp_addr_reg[7]

```

To map MBFFs, the following VCD_FILE GSR option can be used:

```

VCD_FILE {
    ...
    MAPPING_RULE_FILE <file_name>
}

```

The MAPPING_RULE_FILE provides the common rules needed for mapping RTL names to gates. For describing MBFFs, the available rules are as follows:

- define_mbit_prefix <prefix_name>
- where Mbit_prefix denotes the string that comes before the concatenated register names in an MBFF. In the example below,

- XXX/MBIT__cp_addr_reg_reg_0_MB_cp_addr_reg_reg_1_
- MBIT__ is the prefix. This can be provided in MAPPING_RULE_FILE as define_mbit_prefix MBIT__
- define_mbit_delim <delimiter_name>
where mbit_delim denotes the string that separates the two register names. In the example below:
XXX/cp_addr_reg_reg_0_MB_cp_addr_reg_reg_1_
MB_ is the delimiter. This can be provided in MAPPING_RULE_FILE as define_mbit_delim MB_
 - define_mbit_lsb <0/1> (default value is 0)
where mbit_lsb denotes the least significant bit in the MBFF array.

MBFF Name Mapping can support Bus Style Outputs: RedHawk can map register names in RTL VCD to corresponding gate level names in netlist (def), even if these have been synthesized with Multi Bit Flip Flop (MBFF) in gate level. Certain specifics regarding MBFF names have to be specified via a rule file to enable this. This particular MBFF mapping flow has been further enhanced to map bus-style output pin names for MBFF. Pin names of style Q[0], Q[1] etc are supported.

Power Calculation Procedure and Results Evaluation

With all design data imported and the appropriate GSR keyword values set, you can perform power calculation from the TCL command line or using the GUI. The TCL command is:

```
perform pwrcalc
```

If you have already run power calculation on the design and have made no changes, you can import the calculated power directory data from your previous calculation by using the menu command **Static > Power > Import**, and select the previous power directory, such as *adsPower*. Figure 4-4 shows the form for importing power information.

Or perform the import using the TCL command:

```
import power <Power_Dir_Name>
```

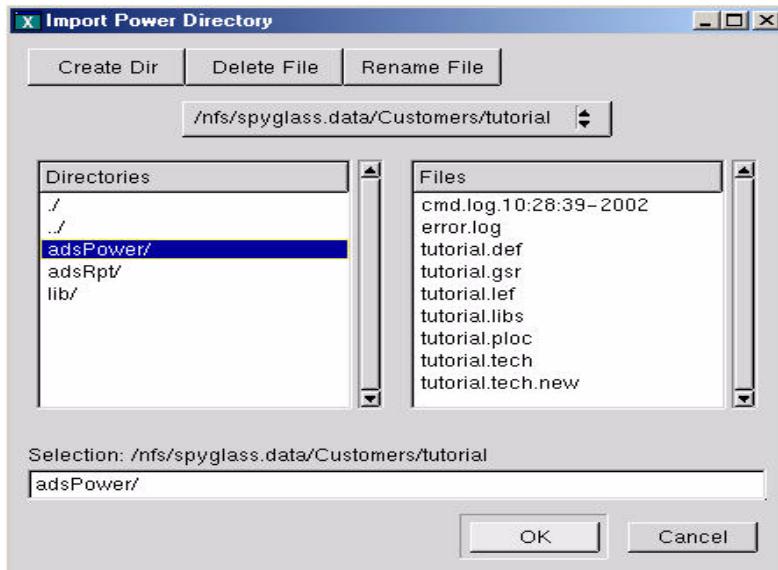


Figure 4-4 Importing adsPower directory information

Or, to perform power calculation using the GUI and then evaluate the results, follow the GUI-based steps below.

1. Calculate the power using the **Static > Power > Calculate** or **Dynamic > Power > Calculate** command.
2. Review the power summary in the file *power_summary.rpt* in the *adsRpt* directory, as shown in Figure 4-5 example. The Power Summary Report recommends the simulation time needed to capture a majority of power events, and lists the calculated total_pwr, leakage_pwr, internal_pwr, switching_pwr and percentage of total power consumption by frequency domain, by power domain, and by cell type. Note that imported power values are scaled during power calculation to meet instance-specific power consumption numbers.
3. Generate a power density distribution map, using commands such as:
 - **View > Power Maps > Power Density Map**
 - **View > Power Maps > Power Map of Instances**
 - **View > Power Maps > Power Map of Clock Instances**

or use the corresponding control buttons on the right side of the GUI.

----- Power Summary Report -----
Recommended dynamic simulation time, 5000psec, to include 100%
of total power for DYNAMIC_SIMULATION_TIME in GSR.

```
INFO: Importing user specified power file: demo.inst.power
INFO: Instances not specified in this file will have zero power.
INFO: 100% of instances specified in instance_power_file have corresponding
instances in the design.
0 out of 4142 instances do not have corresponding instance in the design.
INFO: For complete list of PWR-121 and PWR-122 WARNINGS, please see adsRpt/
apache.inst_pwr_file.mismatch.
```

The power is based on the INSTANCE_POWER FILE: demo.inst.power
Redhawk honors instance-by-instance power as specified in the above file.
Individual components of power, like switching power, internal power,
however, are calculated by RedHawk and may have been scaled to
meet the total instance-specific power number.

Power of different frequency (MHz) domain in Watts:

Frequency	total_pwr	leakage_pwr	internal_pwr	switching_pwr	%_total_pwr
4.000e+02	1.729e-02	1.704e-04	1.107e-02	6.052e-03	8.333e+01%
2.000e+02	3.4595e-03	2.1869e-05	3.1000e-03	3.3758e-04	1.666e+01%
0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00%

Power of different Vdd domain in Watts:

Vdd_domain	total_pwr	leakage_pwr	internal_pwr	switching_pwr	%_total_pwr
VDD (1.1V)	2.075e-02	1.922e-04	1.417e-02	6.389e-03	1.000e+02%

Power of different cell types in Watts:

cell_type	total_pwr	leakage_pwr	internal_pwr	switching_pwr	%_total_pwr
combinational	7.111e-03	1.221e-04	1.754e-03	5.235e-03	3.426e+01
latch_and_FF	4.292e-03	3.890e-05	3.323e-03	9.301e-04	2.067e+01
memory	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
I/O	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
clocked_inst	9.352e-03	3.119e-05	9.096e-03	2.246e-04	4.505e+01
decap	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00

where clocked_inst are instances that cannot be classified as latch_and_FF, memory, or I/O, but have clock pin(s).

Total chip power, 0.020757 Watt including core power and other domain power.
Total clock network only power, 0.003225 Watt. Total clock power, including
clock network and FF/latch clock pin power, 0.005992 Watt.

Figure 4-5 Example Power Summary Report

Some of the information that can be checked in the power summary report is:

- Does the total power consumption make sense?
- Are clock network and the clock power values reasonable?
- Is the power reported by clock frequency what would be expected?

Having several elements that consume a lot of power close to several elements that consumes little power may not create a voltage drop problem. But having an area where many elements consume a lot of power can create a voltage drop problem if the area is not well supplied. The power consumption maps give important feedback regarding the power distribution, i.e., the quality of the placement/floorplanning vs. power consumption.

Some aspects of the power calculation results to investigate:

- What is the total power consumption in the design?
- Is the overall power demand well distributed?
- Will the power pads handle this load safely?
- How is the power balanced between the different frequency domains?
- How much power is used in the clock network?

You can click on any instance to see a text report of its power consumption, as well as its associated capacitance and applied voltage.

Power Grid Resistance Extraction

After power calculation, the next step is to perform network resistance extraction, using either TCL or GUI commands. At technology nodes now being used for design, very precise estimates of wire resistance, and therefore wire thickness, is necessary. This effect of metal density on resistance calculation is described in the following section.

Metal Density Calculation

To obtain very accurate resistance values for wire segments, both width and thickness values must be accurately determined. To achieve an accurate estimate of wire thickness, an accurate assessment of the metal density for all metal geometries in the region of the wire is needed, because of the effects of CMP on wire thickness. The procedure for estimating metal density around wire segment is as follows.

1. Read geometries from DEF/LEF, or from GDS2RH/DB. Include all geometries from LEF/DEF (NETS, SPECIALNETS, PINS, MFILL), as well as all geometries from GDS2DB (including MFILL).
2. Merge and flatten all metal geometries per layer, which eliminates redundant geometries from different hierarchies or from GDS.
3. Calculate the metal density over the whole design in small gridded regions that contain wires.
4. Perform extraction. Calculate the density of segments based on average density of regions around each segment.
5. By default each uniform wire geometry uses the density from an average of all regions that the wire passes through.
6. When you require the most accurate resistance calculation, such as for long wires that go through multiple regions of varying metal density, set the GSR keyword LONG_WIRE_RES_CALC 1. In this case, the metal density of every region that contains the long wire geometry is used in the calculation.
7. Then calculate the final wire thickness based on the calculated metal density-- for example, using the POLYNOMIAL_BASED_THICKNESS_VARIATION Tech file keyword setting.

TCL Command R Extraction

.To run static resistance extraction for IR drop analysis using the TCL command, execute:
`perform extraction [-power | -ground]`

To run RLC extraction for dynamic voltage drop analysis using the TCL command, execute:

`perform extraction [-power | -ground] -l -c.`

The ‘`perform extraction`’ command builds connectivity and performs extraction for the selected elements of the selected power/ground nets.

GUI Extraction

To use the GUI commands to perform extraction on both VDD and VSS, use the following steps:

1. For static IR analysis, use the extraction command **Static > Network Extraction**. From the extraction form displayed, select resistance (R), and both VDD and VSS.
2. For faster static analysis of either the power or the ground network, either VDD or VSS can be selected. However, both VDD and VSS must be selected for dynamic analysis.

The key information you can get from the extraction step is:

- Are all power and ground nets connected to a source? If not, a warning message is displayed: "Net VDD in cell not driven by any pad".
- Are there any missing connections? (reported in the file *adsRpt/apache.*.unconnect*).

Examining Power/Ground Grid Weakness

Excessive voltage drop can occur due to weak power/ground grid structures. RedHawk can identify problems in P/G structures early in the design cycle after placement and CTS is completed, even without STA and SPEF files. P/G weakness analysis can report two different measures of grid weakness:

- perform gridcheck - an estimate of the upper bound on grid resistance for all instances in the design, normalized to the highest instance grid resistance
- perform res_calc - the calculated effective grid resistance for a specified number of instances in the design

For gridcheck, an instance that has a weakly connected Power or Ground network shows up as high impedance in the generated report. Using the GUI you can also review power grid resistance maps showing the distribution of power and ground resistance across the design. Also, pins not connected to VDD or VSS are listed as "floating" in the PG Weakness report, and are listed together at the end of the report.

Note that you do not need to run power calculation or static analysis prior to doing this assessment. Once the design is imported and P/G extraction is performed, the design environment can generate the resistance report using the TCL interface. This can be done early in the design cycle when detailed routing or timing information is not available.

The procedure for generating and evaluating a P/G grid weakness report is as follows:

1. **Normalized resistance estimation.** On the TCL command line execute:

```
perform gridcheck -o <output_file_name> ? -limit <line_limit>?
```

The format of the resistance report displays R_{VDD} and R_{VSS} for each instance or arc as percentages of the total P/G resistance distribution, as shown in the example report below (Figure 4-6). For multiple P/G arc designs, *adsRpt/apache.gridcheck* is constructed as follows:

- a. The arc that has the highest relative R value is displayed first and continues up to the specified line limit.
- b. The rest of the arcs are displayed up to the specified line limit, or until the relative R becomes smaller than the last displayed relative R in the previous arc that reached the line limit.

```
# Power/Grnd arcs listed separately, there are 2 P/G arcs w/ non-zero R.  
#
```

```
# Max. Resistances (%) of Power/Ground Arcs:  
# Arc1 ( VDD_INT VSS) 100.0000  
# Arc2 ( VDD VSS) 93.7410  
{ Power/Ground Arc1:  
# Total VDD_INT(%) VSS(%) Location (x y) ArcID Instance_name  
100.000 49.5167 50.4833 3316.37 3950.55 Arc1 inst_129973/inst_366358  
99.6966 50.8138 49.1862 3318.21 3965.32 Arc1 inst_129973/inst_366355  
99.5191 50.4213 49.5787 3316.83 3961.62 Arc1 inst_129973/inst_366346  
...  
}  
{ Power/Ground Arc2:  
# Total VDD(%) VSS(%) Location (x y) ArcID Instance_name  
93.7410 50.0202 49.9798 2099.90 3888.64 Arc2 inst_129228/inst_465808  
92.9774 50.7174 49.2826 2131.64 3903.41 Arc2 inst_129228/inst_412486  
92.9138 50.8630 49.1370 2131.18 3903.41 Arc2 inst_129228/inst_412485  
92.8791 50.0223 49.9777 2131.64 3896.03 Arc2 inst_129228/inst_466552  
...  
}  
  
# Floating Instances: Location (x y) Name  
floating 0 977.5 io/pad/VSSC/extr/left/4/adsU1  
floating 10000 2956.24 io/pad/VDDC/extr/right/23/adsU1  
floating 5435.09 191 io/pad/extr/VDDC/11/adsU1  
floating 3935.09 191 io/pad/mpi/VDDC/2/inst/adsU1  
floating 7300.05 0 io/pad/extr/VSSC/15/adsU1  
...
```

Figure 4-6 Sample P/G Weakness Report

The instance P/G resistance, R_{inst} , is normalized such that the instance or arc with the highest total effective resistance is assigned a value of 100 and the instance or arc with the lowest total effective resistance has a normalized value of 0, using the following equation:

$$(R_{inst} - R_{min}) / (R_{max} - R_{min}) * 100 \quad (\text{EQ 4})$$

where, R_{inst} is the total effective P/G resistance for an instance ($R_{VDD} + R_{VSS}$), R_{min} is the minimum value of all R_{inst} and R_{max} is the maximum value of all R_{inst} .

The first column of the report, Total, lists the normalized resistance for every cell as a percentage of the maximum R_{inst} . The second column lists the percentage of R_{VDD} grid resistance to the total effective resistance for that instance. For multiple P/G arcs, the % resistance numbers are normalized to the highest resistance of all the domains.

The third column provides a similar number for R_{VSS} . The report values are sorted based on the highest total relative resistance percentage. The sum of 'VDD(%)' and 'VSS(%)' entries is always 100.

A very large imbalance between R_{VDD} and R_{VSS} (in percentage terms) indicates a significant weakness in the P/G grid at that instance, particularly for instances that have large values of total R .

The list of P/G weakness instances also can be viewed without the TCL 'perform gridcheck' command by using the GUI pulldown menu **Results > List of Weak PG Instances**, which brings up an ordered list of high total resistance instances.

Clicking on any entry and on 'Go To Location' zooms to and highlights the weak P/G instance. The list can be sorted based on Total resistance, VDD(%) or VSS(%).

2. View Resistance Maps

The resistance maps can be displayed using the GUI menu option **View -> Resistance maps**

The Resistance maps display either Total Resistance, R_{VDD} , or R_{VSS} , which are the relative effective resistances for all instances in the design. To see the meaning of each color, click on the 'Set Color Range' Configuration button at the right side of the GUI. A 'Resistance Color Map' dialog is displayed, indicating the resistance "gradient" (normalized resistance) for each color. The display and the resistance ranges can be changed using the dialog.

3. The **View -> Nets** menu option can be used to select the resistance maps to be displayed by net.
4. To view a grid problem area, you can highlight the weak instances in GUI using the resistance report described above. Use the following TCL command to highlight weak P/G instance:

```
select addfile high_VDD_res_instance  
to highlights instances with weak VDD structures, or  
select addfile high_VSS_res_instance  
to highlight instances with weak VSS structures.
```

5. **Effective Grid Resistance Calculation.** Reviewing the highlighted high impedance instances, you can now start debugging the causes of weak P/G structures in more detail. An important tool in finding out more about weak grid areas is the 'perform res_calc' TCL command (see [section "perform", page D-784](#), for command syntax description).

The 'perform res_calc' command calculates the effective P/G grid resistance from all pads to selected instances or locations, and provides absolute resistance values. The default invocation, without any options, creates a resistance report of the worst instances, as indicated by an initial quick estimation. Using the options '-instance', '-inst_file', or '-box' allow the effective resistance of particular instances or identified weak areas of the grid to be investigated in more detail. The -cell_file <CellFile> option specifies a text file that contains the cell names, one per line, for res_calc to compute the equivalent grid resistances. A sample 'perform res_calc' output file is shown below.

#	Ohm	Location(x y)	Layer	Net	Instance
4.54617	1005.4	8755.2	MET3	VSS	inst_1234
4.39744	1055.6	8855.3	MET5	VDD	inst_4134
3.32929	1855.7	8455.6	MET3	VDD	inst_2324
...					

6. **Special Node Resistance Reports.** Two other options allow you to select the type of node resistance reports, for standard cells or macro blocks, as follows:

```
perform gridcheck ?-stdcell [ ave| min| max| all| none ]?  
?-macro [ ave| min| max| all| none ]?
```

The option 'ave' is the average resistance for all nodes in each instance selected, and is the default. For the 'min' and 'max' options, the node with the minimum or the maximum resistance value for the instance is reported, instead of the average of all nodes in the instance. The 'none' option is used to eliminate reporting on all standard cell or macro instances. For 'all', the largest 5000 (default) node resistances in the design are reported.

The report format is like 'perform res_calc', where VDD/VSS values are listed point-by-point, without pairing up. An example report follows:

Example all-point gridcheck resistance listing for instances:

Resistance(%)	Location(x y)	Layer	Net	Instance
87.4927	4459.77	443.855	METAL3	VDD
87.3918	2839.76	443.855	METAL3	VDD
42.4231	3741.59	609.345	METAL3	VDD
33.1267	1874.49	119.18	METAL3	VDD

Defining Pad and Package Parameters

Before performing static IR drop and EM analysis, the pad, package wirebond or flip-chip bumps and associated electrical package parameters must be defined.

Command Line Procedure for Package Modeling

The pad, wirebond or flip-chip bump, and package parasitics are typically specified in the **RedHawk**.tech file. The units used for the package TCL commands are: R in Ohms, C in picoFarads, and L in picoHenrys.

The following three TCL commands define simplified lump models for all pad RC, wirebond/bump RLC, and package RLC circuits, respectively.

```
setup pad [-power | -ground] [-r <R_Ohms> | -c <C_pF>]
```

```
setup wirebond [-power | -ground]
[-r <R_Ohms> | -l <L_pH> | -c <C_pF>]
```

```
setup package -r <R_Ohms> -l <L_pH> -c <C_pF>]
```

where

-power -ground : selects which P/G net to define

-r <R_Ohms> : specifies equivalent resistance value in Ohms

-c <C_pF> : specifies equivalent capacitance value in picoFarads

-l <L_pH> : specifies equivalent inductance value in picoHenrys

To specify individual bump/pad RLC values, see the PLOC file specifications in [section "Pad Location File \(*.ploc\)", page C-746](#). If you need a more accurate Spice package subcircuit for your design, see [Chapter 12, "Package and Board Analysis"](#), for more information about package subcircuit modeling.

To set up your package parameters such that you can easily look at the effects of different package designs on power integrity, after extraction you can use the TCL command 'setup pss', which defines PLOC and package subcircuit files to be used in simulation, using the follow syntax:

```
setup pss -pad_file <filename> -subckt <pss_ckt_name>
```

where

-pad_file <filename> : specifies a PLOC pad definition file that must have a .ploc extension and be in PSS PLOC format.

-subckt <pss_ckt_name> : specifies the Spice package subcircuit file

After running **RedHawk** with one set of PLOC and package subcircuit files, you can then modify one or both of the files, rerun the 'setup pss' command, and then rerun **RedHawk** to see the effects of the new package data.

GUI Procedure for Package Modeling

1. Set the package and pad constraints by selecting the **Static -> Pad, Wire_bond/Bump and Package Constraint** command. The data form is displayed, as shown in Figure 4-7.
2. Enter appropriate RLC pad and package values for your design.

The package model RLC values are needed to include the impact of package parameters on the circuit analysis. The package can have a significant impact on voltage drop. If you do not have accurate RLC values from testing or Spice models, you can provide reasonable default values to see the effects on the voltage drop results.

Note: For static analysis only, R values are sufficient. However, for dynamic analysis good L and C values are necessary for accurate results.

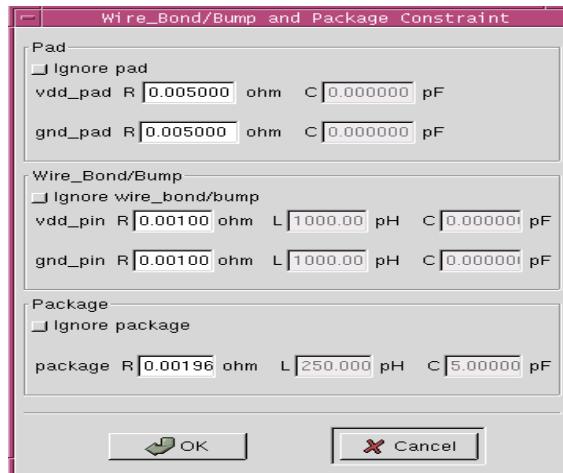


Figure 4-7 Pad, Wire-bond/Bump, and Package constraints form

Package Compiler Utility

The **RedHawk** Package Compiler is a versatile utility that checks the die-package interface, as follows:

- checks package Spice syntax
- checks package RLCK passivity
- calculates effective package Inductance for each voltage domain
- matches die and package pins and creates an annotated PLOC

RedHawk Package Compiler makes use of the Chip Package Protocol (CPP) header information to determine the following:

- identifies package and die pins
- identifies which nodes belong to the die side and which belong to the PCB side
- uses CPP header data to differentiate between power nets and ground nets

For details on using Package Compiler, see section "Chip-Die Mapping Using Package Compiler", page 12-336.

Running RedHawk-S (Static IR/EM Analysis)

If all of the previous steps have been performed properly, you are now ready to run static IR drop and EM analysis, as follows:

1. Use the command **Static > Static IR-drop & EM Analysis**. The execution of this command typically takes a few minutes on a Linux machine.
 You can also run static analysis from the TCL command line, using the command
`perform analysis -static`
2. When the analysis is complete, in the Log window the five worst Voltage drops for each Power/ground net are listed.
3. Evaluate the overall static voltage drop by clicking on the **IR** button, as shown in an example VDD plot in Figure 4-8 and VSS plot in Figure 4-9.

The following are some important issues that can be identified by the static voltage drop maps:

- number and location of hot spots (expected or not)
- unexpected color jumps may indicate missing straps or connections
- any unexpected black areas (which could mean a black box element, missing data, a missing logical connection, or a missing physical connection)
- if a color change from source to hot spot make sense

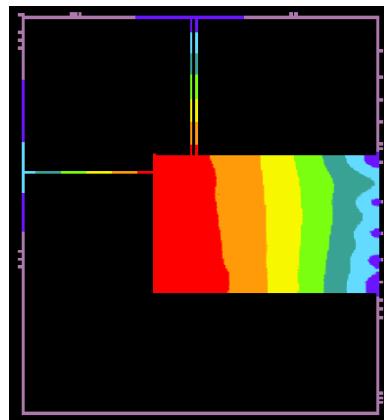


Figure 4-8 **VDD static IR drop map**

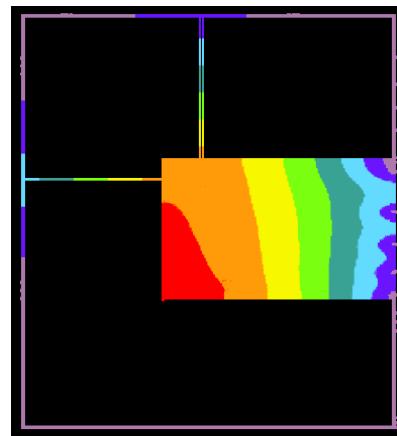


Figure 4-9 **VSS static IR drop map**

Exporting and Importing Results to the Design Database

The results of intermediate **RedHawk** operations and final results can be exported to the design database from the command line using

```
export db <Output_filename>
```

or from the GUI using the **File -> Export DB** menu command. The exported DB is platform-independent among Linux, Solaris, and AMD 64-bit Linux systems.

The results can be imported back into **RedHawk** as needed using the command

```
import db <DB_filename>
```

or the **File -> Import DB** command in subsequent runs.

Exporting a Database

The following procedure describes how to use the *export db* command:

1. Select the **Export DB** menu item. A dialog window is displayed.
2. Type in the design directory name and click on the **OK** button.
3. A snapshot directory is created, containing binary files with the following names:
 - *general* (general information such as *.gsr*, *.tech*, etc.)
 - *library.0*, *library.1*, ... (library information)
 - *design.0*, *design.1*, *design.2*, ... (design netlist, hierarchy, data, etc.)

Some text files, such as *apache.gsr*, which are needed for power calculation and simulation, are also included in **Export DB**.

Database Compatibility

The compatibility of **RedHawk** databases from different releases is as follows:

- If the database version and **RedHawk** version are of the same major release, but from different minor releases or patches:
 - a. A database generated by an older version of **RedHawk** can be loaded with a newer version of **RedHawk**, but new features in the newer versions of **RedHawk** are not available.
 - b. A database generated by a newer version of **RedHawk** cannot be loaded by an older version of **RedHawk**.
- If the database version and **RedHawk** version are from different major releases (for example, 2007.x and 2008.x), the database cannot be loaded across these versions.

Apache recommends that the database version and the RedHawk version used to load the DB be the same. The “database version” is the version of the tool used to generate the database.

Importing a Database

The following procedure describes how to use the *import db* command:

1. Select the **Import DB** menu item, a dialog window is displayed.
2. Select the design directory name and click on the **OK** button.
3. If the entry field of **Selection: ...** is empty, as may happen on AMD64-bit and Enterprise 3.0 platforms, type in the directory name in the selection field. For example, enter */home/design/MY_DB*, where *MY_DB* is the DB directory that needs to be imported.

NOTE: You may only load the design into the snapshot directory. No editing or updates are allowed in any of the binary files. No query operations are supported.

For version control purposes, a label is encoded into the binary file *general*. Therefore, DB snapshots can only be reloaded using the same version of RedHawk that was used to run ‘`export db`’. If there is a mismatch between DB and RedHawk versions, an Error message is displayed and the RedHawk session terminates, with the following type of error message:

```
Data/File version is not RedHawk2004100-BINARY!
```

Flexible Memory Caching for Database Reloading

To accommodate the physical memory size of different machines that are used to run RedHawk, use the `-cache_mode` option when importing a database, particularly when the design size is larger than the available physical memory. This allows smart caching of the part of the database that does not fit into the memory. So regardless of the memory size of the machine from which the DB was exported, to efficiently import a saved database `dynamic_run1.db` into a machine with limited physical memory, use the command

```
import db dynamic_run1.db -cache_mode 1
```

Note that setting the GSR keyword ‘CACHE_MODE 1’ also invokes RedHawk adaptive memory caching. Setting a zero value turns caching off in both types of invocations.

Early Analysis Methodology

Overview

For designs that are in early stages of design and do not have complete placement and routing information, RedHawk allows you to perform power grid verification early in the design process to ensure that the grid meets initial design guidelines. This can verify, at an early stage, the placement of power pads and check electromigration issues at the pad connections or at other key locations on the grid. The usage model is extremely flexible and allows you to run the analysis with different types of design abstraction. The following levels of design abstraction and power grid completion are supported:

1. Design has power and ground (P/G) routing only, with no block placement information. Power consumption numbers are available for either the entire chip or for the entire chip along with information on power consumption in specific regions.
2. Design has P/G routing and macro placements only. Macros do not have any port views (LEF pins) or detailed views (P/G routing).
3. Design has P/G routing and macro placements. Macros have port views (LEF pins) but do not have detailed views (P/G routing).
4. Design has P/G routing and macro placements. Macros can have either port views (LEF pins) or detailed views (P/G routing).

RedHawk can help analyze designs at any of the stages listed above. Based on the data provided and the constraints specified, RedHawk creates realistic current sinks in the design and assigns user-specified current at the current sink locations. Several examples

of different early analysis scenarios, with different levels of block information, are shown in Figure 4-10.

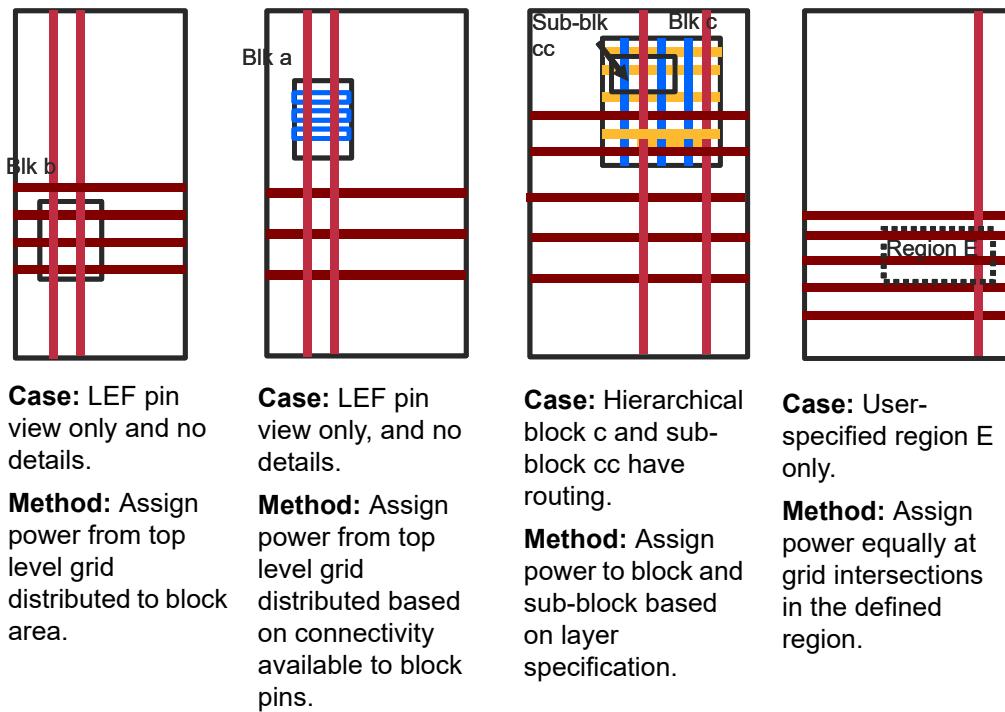


Figure 4-10 Early block analysis scenarios

Input Data Required

The data requirements for executing early design static analysis are as follows:

1. RedHawk technology file
2. LEF file that describes the macros placed in the design. If current sinks have to be created at the ports of the macros, the LEF file must contain the PIN definitions.
3. DEF file for the macros for which detailed routing must be considered
4. top level DEF file for the design
5. voltage source location definitions
6. GSR file

Early Analysis Flow

Block Power and Current Assignment

For performing power analysis on early stage designs RedHawk uses information specified using the GSR keyword BLOCK_POWER_ASSIGNMENT, which defines power parameters for particular blocks, pins, regions and the top level in early stages of design. RedHawk creates current sinks in the design based on the constraints specified with BLOCK_POWER_ASSIGNMENT and distributes power among these current sinks based on user-defined power and/or current numbers. Particular regions and blocks can also be excluded from power assignment, if desired, and power calculation data then can be used for those excluded.

Because of the large number of options for different types of applications, the syntax is broken down into different types of applications in the following syntax section.

Syntax:

```
BLOCK_POWER_ASSIGNMENT {
```

- To assign power to existing instances from LEF/DEF:

```
<instName> [BLOCK|PIN] <layerSpec> <netName> [<powerW>|<currentA>]  
where  
<layerSpec> specifies either the layer name, or ALL to include all layers, or TOP  
or BOTTOM to include either the highest or the lowest mask layers  
overlapping the BPA instance.  
[<powerW>|<currentA>] : specifies either the power in Watts for power nets, the  
ground current in Amps for ground nets, or '-1' to allow power calculation to  
decide the power.
```
- To create a new instance and assign power:

```
<BPA_instName> REGION <layerSpec> <netName> [<powerW>|<currentA>]  
<llx> <lliy> <urx> <ury>
```
- To create a new instance for cells declared in DECAP_CELL or
BLOCK_POWER_MASTER_CELL keywords:

```
<BPA_instName> <cellName> <layerSpec> <netName>  
[<powerW>|<currentA>] <llx> <lliy> <orientation>
```
- To assign power in a region the size of the entire design:

```
<BPA_instName> FULLCHIP <layerSpec> <netName>  
[<powerW>|<currentA>]
```
- To define sub-regions inside an existing BPA instance (multiple declarations on
the same instance is allowed) for power assignment:

```
<BPA_instName> BLOCK AREA <llx> <lliy> <urx> <ury>  
<BPA_instName> BLOCK RECTILINEAR <x1 y1 x2 y3 x4 y5 ..>
```
- To exclude a region from BPA power assignment:

```
<regionName> REGION EXCLUDE <llx> <lliy> <urx> <ury>
```
- To exclude an overlapping instance in the design from BPA power assignment:

```
<instName> BLOCK EXCLUDE
```
- To modify an area defined by 'EXCLUDE' to form rectilinear regions:

```
<regionName> REGION INCLUDE <llx> <lliy> <urx> <ury>
```
- To use the boundary box of an existing instance to do the same as above:

```
<instName> BLOCK INCLUDE
```
- To exclude macro instances overlapping an BPA instance for power
assignment:

```
<BPA_instName> BLOCK EXCLUDE_MACRO
```

- To exclude instances defined in the BLOCK_POWER_FOR_SCALING keyword and which overlap the BPA instance for power assignment:
`<instName> BLOCK EXCLUDE_BPFS`
- To exclude an area occupied by all instances overlapping the BPA instance:
`<BPA_instName> BLOCK OVERLAP_OK`

Using the BLOCK_POWER_ASSIGNMENT (BPA) keyword you can specify VDD power and/or VSS current for any block or region for multiple metal /via layers on which current sinks are created. Or if “-1” is specified, power calculation determines the appropriate power/current based on other information available, such as toggle rate, BPFS and APL characterization. If metal layers are specified, all top and bottom vias for those layers that satisfy other criteria act as current sinks, and the assigned power or current is distributed equally among these current sinks. If a via layer is specified, the current sinks are created in the lower metal layer, if possible. Otherwise, the current sink is created in the upper metal layer of the via. Current sinks can be created for different power and ground nets in the design. All current sinks are analyzed in a single run. The keyword ‘ALL’ can be used to include all layers in the power/current analysis.

The total current drawn by the current sinks in a power net depends on the power (Watts) assigned to that net for a specific region, block or full-chip. The total current drawn by the current sinks in a ground net, on the other hand, depend on the current (Amps) assigned to that ground net for a specific region, block, or full-chip. Negative currents may be assigned as needed for particular design methodologies.

BLOCK_POWER_ASSIGNMENT supports MMX pin-based region power assignment for MMX instances that have many P/G pins inside to represent transistors using the MMX_PIN and MMX_REGION keywords. See [section "Power Assignment to MMX Pin-based Regions", page 4-66](#).

The INCLUDE and EXCLUDE capabilities make it easier for you to define areas that are not complete rectangles to be specified for power assignment. The EXCLUDE option allows you to define rectangular regions to be excluded from areas occupied by BPA blocks/regions, and the INCLUDE option then can *add back* areas to be included *within* EXCLUDE areas.

The power calculation engine can estimate power and current for early stage blocks by setting their BLOCK_POWER_ASSIGNMENT values to -1. This can be used for either cell instances (blocks) or regions. You must provide correct and complete power calculation data so that power for the block can be calculated accurately. See [section "Power Calculation", page 4-35](#), for details.

Use one line in the BPA specification for every domain (net) in the block, such as:

```
regionA REGION met3 vdd1 -1 100 200 300 400
regionA REGION met4 vdd2 -1 100 200 300 400
regionA REGION met3 vss -1 100 200 300 400
```

Note that if one net in the block is assigned -1 for power/current in BPA, power/current for *all* domains in the block is determined by the power calculation engine.

All other areas and elements of the design *not specified* in the Block_Power_Assignment statement are treated as in a normal RedHawk analysis. So early stage blocks that have power assigned in BLOCK_POWER_ASSIGNMENT (BPA) can be simulated together with other “regular” blocks (fully designed and specified), including those specified using the keyword BLOCK_POWER_FOR_SCALING (BPFS). For this mixed process any BPA

declaration overrides a BPFS declaration for the same instance. Power assignment for FULLCHIP must be defined in BPFS, not in BPA.

So a general guideline is to use BPFS for “ready” completed instances, and BPA for “early” blocks (such as regions, or macro (LEF) cell instances inside “early” design blocks). If you need to assign power with FULLCHIP, using BPFS is highly recommended. This generally applies to “mixed” early designs (that is, when the design has some blocks that are completed, mixed with some early blocks).

You can also use the following GSR keywords to refine current sink assignment for instances defined in BLOCK_POWER_ASSIGNMENT (see [page C-575](#)):

BPA_BY_LAYER
BPA_CONN_DISTANCE
BPA_CONN_MARGIN
BPA_CURRENT_DENSITY

Block Power Assignment On-the-fly

Early analysis can also support interactive block power assignment and re-definition, allowing you to experiment, using the TCL command

```
gsr set BLOCK_POWER_ASSIGNMENT_FILE <BPA_filename>
```

anytime before and after 'setup design' to initialize and re-initialize the BPA settings. Some example cases follow:

To display the current BPA settings in **RedHawk**:

```
gsr get BLOCK_POWER_ASSIGNMENT
```

To display the block power master cells defined:

```
gsr get BLOCK_POWER_MASTER_CELL
```

When using interactive changes to BPA, **RedHawk** compares any newly-loaded BPA statements to what is already in **RedHawk**, and proceeds as follows:

- If there are errors in the newly-loaded set, such that the BPA statement cannot be understood, **RedHawk** does not change that BPA setting.
- If a new BPA statement is not in **RedHawk**, it is added.
- If a BPA statement is in **RedHawk**, but not in the new set, it is deleted.
- For BPA statements in both versions, the new BPA parameters are used.

Hierarchical Power/Current Assignment

For power/current assigned to hierarchical elements in a design, power/current assigned to child elements are included in the parent's assignment. So a parent's power assignment is the sum of the power assigned specifically to its children and also the power assigned to areas that are outside of its children. See the hierarchical assignment example below:

```
CHIP FULLCHIP m1 vdd 10
A BLOCK m1 vdd 8
A/B BLOCK m1 vdd 5
A/B/C BLOCK m1 vdd 2
```

So for this case BLOCK 'A/B/C' is assigned 2W, 'A/B outside of 'C' is assigned 5 - 2 = 3W (a total for 'A/B' of 3+2=5W), and the parent BLOCK 'A' outside of 'B' has the remainder of 3W (a total of 3+5=8W for 'A'). For nodes in CHIP that are not in 'A', 10-8 = 2W is assigned, and the whole chip has 10W.

The ‘REGION’ specs in Block_Power_Assignment are designed as “place holders” for design elements that are not available yet, so a warning is displayed if the coordinates of two different REGIONS overlap, except for “sub-regions”. That is, for “regionA” and “regionA/sub1”, “regionA/sub1” is expected to be inside “regionA”. Also, if a BPA region overlaps all or part of a regular instance, the instance is considered “inactive”, and is ignored in the analysis.

Every region declared in BLOCK_POWER_ASSIGNMENT can be referenced as an instance in the design using the region name as the instance name. The cell name associated with the instance is then the region name for most cases, except for regions that have a hierarchical name. For example, in the cell name “U10/I5”, the “/” is replaced by a “_” character in the cell name, “U10_I5”.

Optionally rectilinear areas inside BPA REGIONS or in LEF/DEF blocks, that share a common master cell can be defined. Rectilinear area definitions enable you to distribute power to specified sub-areas within a rectilinear block. Power/current assigned to REGIONS/blocks is hooked up within the user-defined rectilinear areas. You must define the common master cells and the rectilinear regions within it using a BLOCK_POWER_MASTER_CELL definition and the BLOCK_POWER_ASSIGNMENT keyword. Different regions can share the same master cell. You can use the GSR keyword BLOCK_POWER_MASTER_CELL to achieve this, with the following syntax:

```
BLOCK_POWER_MASTER_CELL {  
    <master_cell_name1> <BB_llx> <BB_lly> <BB_urx> <BB_ury>  
    ...  
    ?<master_cell_name2> <bbox> {  
        <sub-area bbox>  
        ...  
    }?  
}
```

Multiple regions can share a common master cell using BLOCK_POWER_ASSIGNMENT by referencing a cell name declared in BLOCK_POWER_MASTER_CELL, as follows:

```
BLOCK_POWER_ASSIGNMENT {  
    <region> <master_cell_name> <layer> <net_name>  
    [<power_W>|<gnd_I>] <BB_llx> <BB_lly> <orientation_code>  
    ...  
}
```

A typical example of using the Block_Power_Assignment keyword follows:

RegionABC	FULLCHIP	via7	VDD	2.0	# in Watts
RegionABC	FULLCHIP	via5	VDDC	0.4	# in Watts
BlckA	BLOCK	MET6	GND	1.0	# in Amps
BlckB	PIN	MET6	VDD	0.2	# in Watts
BlckC	BLOCK	via2	VDD	0.4	# in Watts
Region1	REGION	via3	VDDC	0.3 30.0 65.0 60.0 95.0	
IOL	REGION	MET5 VDD_L	0.005 100 1050 1000 9900		
IOL	REGION	MET6 VDD_L	0.010 100 1050 1000 9900		
regionA	REGION	ALL VSS	0.1 10.0 20.0 30.0 4.0		

To specify power in regions inside an MMX instance, you can use region names with the prefix “adsU1/”-- for example, “adsU1/regionA”. In this way RedHawk knows that “adsU1/regionA” is a sub-region inside ‘adsU1’ and the BLOCK_POWER_ASSIGNMENT specifications can be interpreted correctly.

The steps for executing an early design static analysis flow are summarized in the following TCL commands:

```
setup design <GSR_NAME>.gsr
perform pwrcalc
perform extraction -power -ground
perform analysis -static
```

Power Assignment to MMX Pin-based Regions

BLOCK_POWER_ASSIGNMENT supports MMX pin-based region power assignment for MMX instances that have many P/G pins inside to represent transistors using the MMX_PIN and MMX_REGION keywords. So the power is only distributed to P/G pins, while regular region-based BPA assigns power to nodes on the given layer within the region. This feature is used to change the power distribution inside a MMX instance for static analysis only.

You can assign power :

- to transistor pins inside MMX instances using the MMX_PIN keyword in BPA.
- to regions inside MMX instances using MMX_REGION keyword and a “/” character to indicate hierarchy in the first column.
- hierarchically by defining a region inside another region using the MMX_REGION keyword. Use “/” character to indicate hierarchy.

For example:

```
BLOCK_POWER_ASSIGNMENT {
    BlockABC/ram64/adsU1 MMX_PIN all VDD 0.18
    BlockABC/ram64/adsU1 MMX_PIN all VSS 0.1
    BlockABC/ram64/adsU1/R1 MMX_REGION all VDD 0.04 1789 980 1822 1031
    BlockABC/ram64/adsU1/R1 MMX_REGION all VSS 0.01 1789 980 1822 1031
    BlockABC/ram64/adsU1/R10 MMX_REGION all VSS 0.03 1842 934 1888 966
    BlockABC/ram64/adsU1/R10/R11 MMX_REGION all VSS 0.02 1843 935 1863 965
    BlockABC/ram64/adsU1/RR REGION all VDD 0.09 0 0 40 40
    BlockABC/ram64/adsU1/RR REGION all VSS 0.05 0 0 40 40
}
```

In general, block power is assigned hierarchically, so power of a child block of the same type as the parent is considered part of the parent's power assignment.

In the above example, the final power for this instance is 0.18W for VDD and 0.1A for VSS. Power/current in region R1 is 0.04W for VDD and 0.01A for VSS. Current in region R10 is 0.03A for VSS. Current in region R10/R11 is 0.02A for VSS.

The last two lines in this BPA example are regular metal/via region-based BPA settings. Instance *BlockABC/ram64/adsU1/RR* is created and is assigned 0.09W for VDD and 0.05A for VSS. However, since the */RR region is not type MMX, its power is not included in the MMX region assignments.

Note that unlike metal/via region-based BPA, pin-based region BPA does not create instance hierarchy for the given region, but it redistributes and adjusts the transistor pin power/current inside the MMX instance, while keeping the total power/current of the MMX instance unchanged.

Also, pin-based region BPA and metal/via-based region BPA can be used in the same design at the same time, as follows:

1. You can apply pin-based BPA to MMX instance while assigning metal/via based BPA to place outside MMX instance.
2. It is possible to assign pin-based region BPA and metal/via based region BPA to the same MMX instance. But the metal/via based region BPA cannot cover any MMX transistor pins, or it would conflict with the assumption. Use MMX_REGION

to handle such case. So in the above example, pin-based region BPA is used while assigning power/current to “BlockABC/ram64/adsU1/RR” with metal/via existing, but the metal/via region BPA does not include any MMX transistor pins.

MMX BPA Restrictions

The following restrictions on MMX pin-based region BPA use are checked to avoid analysis errors:

1. An MMX instance cannot be inside an MMX_REGION.
2. An MMX_REGION must be defined inside an MMX instance.
3. An MMX_REGION can only be within another MMX_REGION if they belong to the same instance.
4. Regions cannot overlap.

Power Assignment to OBS Regions in LEF Macros

Early stage analysis allows assignment of power to ‘OBS’ regions in LEF blocks. The regions are defined in the OBS section in LEF for cell macros. There are two elements to this methodology:

- RedHawk reads the specified parts of the OBS section in LEF to get the regions to be assigned power, when a new GSR keyword is turned on (default off, 0):

```
READ_LEF_OBS 1
```

The only parts of the OBS data read in are LAYERs defined as OVERLAP and shapes defined as RECT. All other items defined in the LEF OBS section are ignored. For example, the following OBS section items in LEF would be included in power assignment:

```
OBS
    LAYER OVERLAP ;
    RECT 0.0 0.0 800.0 400.0 ;
    RECT 0.0 400.0 400.0 600.0 ;
END
```

- Also, in the BLOCK_POWER_ASSIGNMENT (BPA) GSR keyword, a block type keyword ‘OBS’ is used. So LEF/DEF instances are categorized as BLOCK, PIN, or OBS. User-defined regions are specified by two GSR keywords:
 - REGION in BPA
 - <cellName> in BLOCK_POWER_MASTER_CELL

Areas defined in the specified LEF OBS sections are used to assign power to a block only when the block type is also OBS in the BPA statement. The OBS type keyword is a block-level attribute by implementation, not a net-level attribute; BPA syntax is “per net” by definition. So if a user specifies the following:

```
U1 BLOCK metall1 VDD 0.2
U1 OBS metall2 VSS 0.15
```

it would be interpreted as applying OBS to ALL nets specified in BPA for U1.

So it should be noted that the new OBS syntax assigns power/current to the whole specified OBS area, and if one net is assigned to OBS for a block, all nets in the block are assigned to OBS.

Creating Decap Cells During BPA

You can use the GSR keyword DECAP_CELL and BPA to create new decap cells/blocks and assign decap values,. These decap blocks are also considered by the 'print decap' command. Decap cells can either come from LEF or be created in the flow. BPA

instantiates the existing decap cells to put decap instances in the design, using the syntax:

```
APL_FILES {
    # optional
    mydecap.cdev cap
    ...
}

BLOCK_POWER_ASSIGNMENT {
    DecapABC_10 DecapABC METAL1 VDD 0.0 3800 4200 N
    DecapABC_10 DecapABC METAL1 VSS 0.0 3800 4200 N
    DecapABC_20 LEF_decapl METAL1 VDD 0.0 3900 4300 N
    DecapABC_20 LEF_decapl METAL1 VSS 0.0 3900 4300 N
}

DECAP_CELL {
    # decap cells defined in LEF
    LEF_decapl
    ...

    # decap cells to be created, use the syntax:
    # <decap_cell> <width> <height> <C_pF> <R_ohm> <layer> <leakage_A>
    DecapABC 100 100 0.1 10 METAL1 0.001
}
```

In this example RedHawk creates a decap cell (instead of a regular leaf cell) for 'DecapABC', and 'DecapABC_10' and 'DecapABC_20' are decap instances.

When a decap instance is selected in the GUI, the instance name, the cell name and the decap value assigned to it are all displayed in the log window.

Early Stage Decap Estimation

Early stage decap estimation can be performed to help meet your global dynamic voltage drop (DvD) requirements, using the following estimation procedure:

1. Run DvD analysis on the design that has rough placement or CTS completed. Look at DvD values.
2. Place user-specified decap cells uniformly over all standard cell rows to fill a desired percentage of rows (may result in overlaps with existing standard cells), using the 'decap fill' command for uniform coverage:

```
decap fill -uniform <percentage> -pattern {<list decap masters>}
```
3. Rerun DvD analysis to get new DvD values. Delete the placed decaps if the global DvD target is not met (ignore local hot spots).
4. Repeat steps 2 and 3 with different combinations of percentage coverage values and decap master cells until the global DvD target is met.
5. Feed back the amount of decap required (and optionally, placement info) to the place and route tool for the pre-placement design database.
6. Continue with regular placement and routing steps.

The command creates a report of results in a tabular format, with the number and amount of decaps added for each decap master. A sample output report is shown below:

DECAP_MASTER_NAME	Number_of_Instance	Amount_of_Decap_Added
cell_698	333548	457.397700 pF

cell_200	333548	93.843726 pF
Total_Amount	667096	551.241426 pF

Reports Created

The reports created from early analysis are very similar to those available from a standard **RedHawk** static analysis, including:

- wire-based voltage drops
- pad currents
- potential EM problem areas

Example Analyses

The following cases describe how you can simulate early designs for different design phase scenarios.

Case 1

Conditions: Design has power and ground (P/G) routing only, with no block placement information available. Power consumption numbers are available for either the entire chip or for the entire chip along with information on power consumption in specific regions.

For this case you can specify top (FULLCHIP) level power and define region specific power. He or she can also define the metal or via layer on which the current sinks can be inserted and these definitions can be unique for each region and for the full-chip.

The Block_Power_Assignment keyword syntax is as follows:

```
BLOCK_POWER_ASSIGNMENT {
    [ <DEF_inst_name> [ BLOCK | PIN ] |
      <regionName> [ FULLCHIP | REGION ] ]
    [ <layer_name> | ALL | TOP | BOTTOM |
      <via_name> ] <pwr_domain_name>
    [ <domain_power-W> | <gnd_net_current-A> | -1 ]
    ?<Bounding box- x1 y1 x2 y2 - req'd for REGION, same line>?
}
```

So for this case current sinks in top level metal6 consume 1.0W of power in the VDD_X domain:

```
RegionXYZ FULLCHIP     metal6 VDD_X   1.0
...
...
```

For current sinks in via4 in a region bounded by opposite corner locations 100,100 and 200,200, and draw 0.35 Amps in the GND_X net, the entry is:

```
RegionA   REGION via4   GND_X 0.35 100.0 100.0 200.0 200.0
...
}
```

RedHawk inserts current sinks in all via4 shapes that fall inside “RegionA”. These current sinks draw a total of 0.35Amps in the GND_X net. Specifying a metal layer name for this region means that all top and bottom vias for the specific layer that falls inside “RegionA” would have current sink locations. Or, if you specify a via layer name for this region, all vias of that layer type that fall inside “RegionA” would have current sink locations.

Case 2

Conditions: Design has P/G routing and macro placements only. Macros do not have port views (LEF pins) or detailed views (P/G routing).

For this case, you can specify top (FULLCHIP) level power and define block-specific power. You also can define the metal or via layer on which the current sinks can be inserted; these definitions can be unique for each block and for the full-chip.

Using the Block_power_assignment syntax described above, for this case the current sinks at top level metal6 consume 1.0W of power in VDD_X domain would be specified:

```
RegionMNOP    FULLCHIP metal6 VDD_X 1.0
...
...
```

And for current sinks in via4 in a block that consumes 0.35Amps in the GND_X net:

```
BlockA BLOCK via4 GND_X 0.35
BlockA BLOCK via4 VDD_X 0.42
...
}
```

For current sinks in metal3 in a block that consumes 0.72W in the VDD_X net:

```
BlockB BLOCK metal3 VDD_X 0.72
...
}
```

RedHawk inserts current sinks in all top and bottom vias that intersect the specified layer at the top level. In this case all via5 and via6 shapes in the top level outside “BlockA” and “BlockB” have 1.0W assigned (since in both of these two blocks, VDD_X net is covered, which is the same one specified in the top level). The power assigned to the FULLCHIP current sinks is 1.0W, so the power for the top level should be specific to the top level current sinks exclusively.

RedHawk inserts current sinks in all via4 shapes that fall inside “BlockA”. These current sinks draw a total of 0.35Amps in the GND_X net.

All current sinks on metal3 geometries inside “BlockB” are designated as current sinks specific to “BlockB”. Power of 0.72W is assigned to the current sinks in the VDD_X net.

Case 3

Conditions: Design has P/G routing and macro placements. Macros have port views (LEF pins) but do not have detailed views (P/G routing).

For this case, you can specify top (FULLCHIP) level power and define block-specific power. You also can define the metal or via layer on which the current sinks are inserted, and these definitions can be unique for each block and for the full-chip.

The same GSR keyword syntax is used:

```
BLOCK_POWER_ASSIGNMENT {
    [ <DEF_inst_name> [ BLOCK | PIN ] |
      <regionName> [ FULLCHIP | REGION ] ]
    [ <layer_name> | ALL | TOP | BOTTOM |
      <via_name> ] <pwr_domain_name>
    [ <domain_power-W> | <gnd_net_current-A> | -1 ]
    ?<Bounding box x1 y1 x2 y2 - req'd for REGION, same line>?
    ...
}
```

So for current sinks in top level metal6 that consume 1.0W of power in the VDD_X domain:

```
RegionCDEF FULLCHIP metal6 VDD_X 1.0
...
...
```

For current sinks in via4 in a block that consumes 0.35Amps in the GND_X net:

```
BlockA PIN      via4      GND_X   0 .35
BlockA PIN      via4      VDD_X   0 .42
...
}
```

For current sinks in metal3 in a block that consumes 0.72W in the VDD_X net:

```
BlockB PIN      metal3   VDD_X   0 .72
...
}
```

RedHawk inserts current sinks in all top and bottom vias that intersect the specified layer at the top level -- in this case all via5 and via6 shapes in the top level outside “BlockA”. “BlockB” has 1.0W power assigned. The power assigned to the FULLCHIP current sinks is 1.0W, so the power for the top level should be specific to the top level current sinks exclusively.

RedHawk inserts current sinks in all via4 shapes that fall inside “BlockA” and that intersect with all the PIN geometries defined in the LEF view of “BlockA”. These current sinks draw a total of 0.35Amps in the GND_X net.

All via2 and via3 shapes that intersect with metal3 geometries inside “BlockB” that intersect with all the PIN geometries defined in the LEF view of “BlockB” are designated as current sinks specific to “BlockB”. A power of 0.72W is assigned to the current sinks in the VDD_X net.

Evaluating Results of Static IR Voltage Drop Analysis

A number of useful analysis techniques are available for viewing the results of voltage analysis, which are discussed in the following steps.

1. View IR voltage drop by layer by selecting **View -> Map Configuration -> IR Drop Color Map**.
2. Zoom in so that individual instance voltages can be seen, as shown in Figure 4-11.

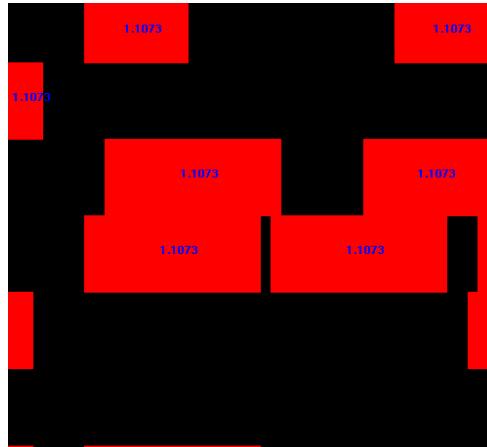


Figure 4-11 View individual instance IR drop values with **View -> Map Configuration -> IR Drop Color Map**

3. You can change the range of IR drop violations displayed and the display colors by using the ‘Set Color Range’ button in the ‘Configuration’ panel (middle button) on the right hand side. Figure 4-12 shows the form for setting the IR voltage drop color

range, which enables you to change either the percentage of the voltage drop to VDD or the absolute value of the voltage drop. You can also click on any of the color buttons to customize the color display and the range of voltage drops each color represents. The colors show the varying levels of voltage drop, from the highest in red to the lowest in blue and magenta.

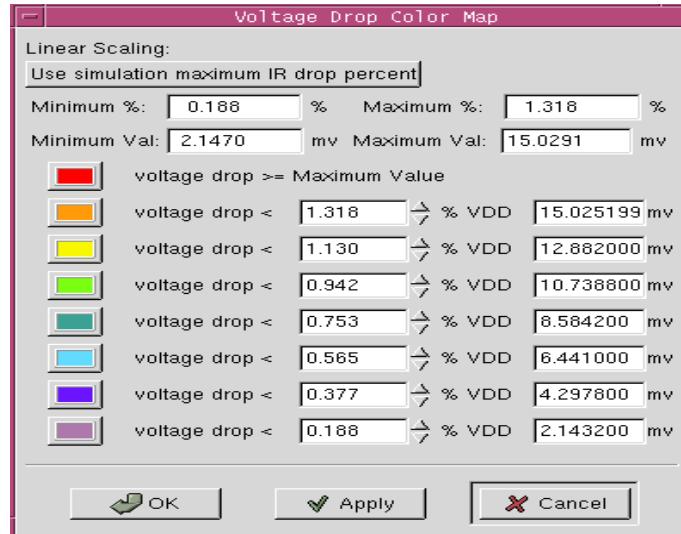


Figure 4-12 settings for the voltage drop color map

4. Zoom in on the voltage drop color map and see more detail, as shown in Figure 4-13.
5. Observe the instances with the highest power usage using the **IPM** button on the ‘View Results’ panel. Click on a key hot instance, as shown in Figure 4-14.
6. Observe the power density distribution of the design using the **PD** button on the ‘View Results’ panel.
7. Observe the electromigration profile and possible EM violations using the **EM** button on the ‘View Results’ panel.

Once the static results are acceptable (analysis runs without a problem, results are well understood, no obvious power grid/layout issues), and once you have done all the needed modifications, it is time to proceed to the voltage drop analysis in dynamic mode.



Figure 4-13 Zoom in to view the worst voltage drop violation area

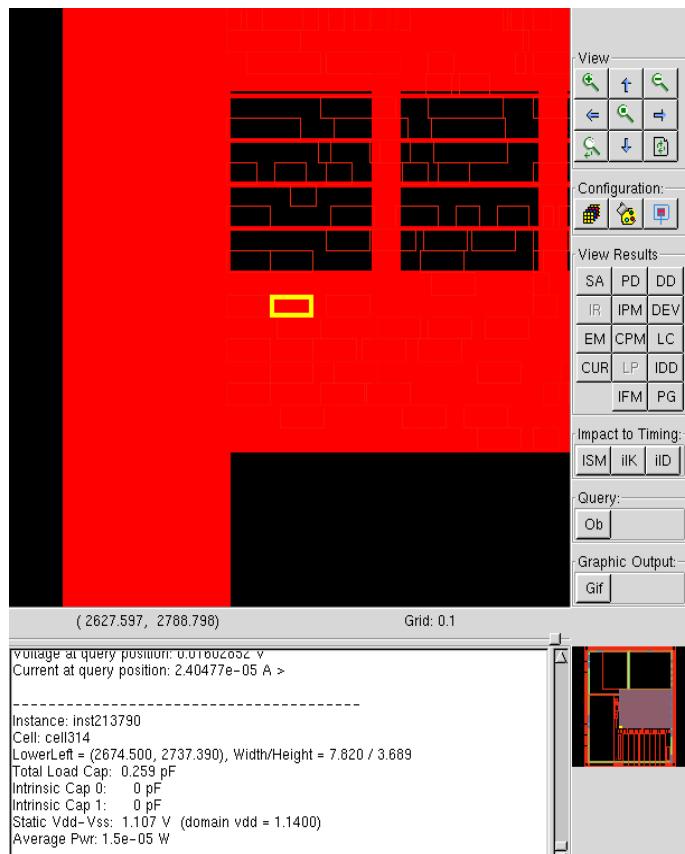


Figure 4-14 Click on a hot instance for more information

Example Procedure to Fix IR Drop Problems

This brief example section uses an example design to illustrate how to fix IR drop problems discovered during static analysis. The example demonstrates how the addition of three power pads, two metal straps and a metal layer resistivity change can reduce static IR drop. This example assumes that you are familiar with procedures for running IR drop and EM analysis, described previously in this chapter. If you want more details on the power grid modification commands used in this example, see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#).

Example IR Drop Case

The original IR drop map of the example design is shown in Figure 4-15. The results show a worst-case Vdd - Vss voltage differential of 1.1073V in the upper left corner of the chip, which contains several high-power instances. The RedHawk screen displays the following message highlighting the worst-case IR drop.

```
The worst IR drop of the top cover cell
voltage = 1.1073 at node (2679.182500,2737.390000)
```

The existing metal4 straps extending from the top and the metal6 strap extending from the left side do not supply adequate power to the high-power instances.

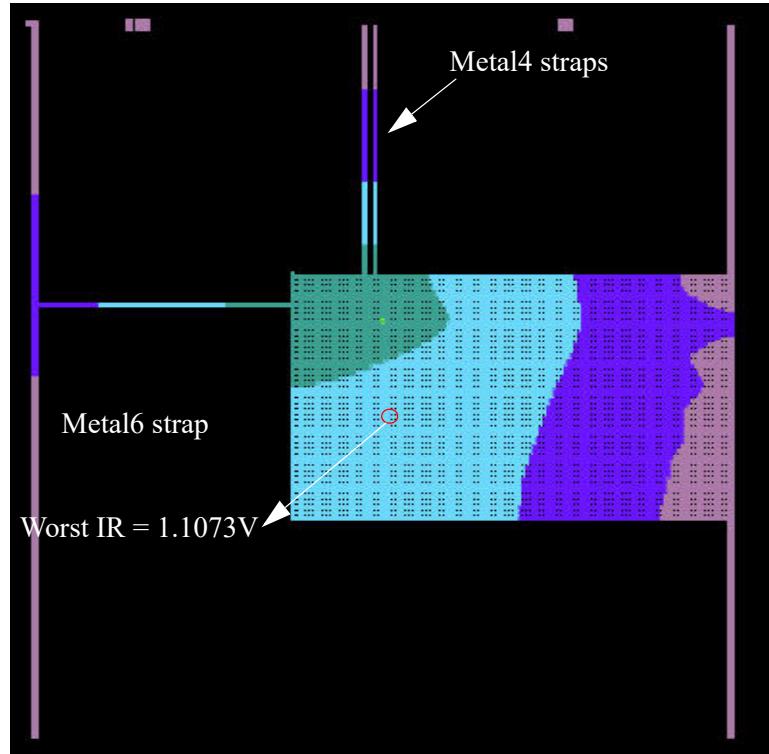


Figure 4-15 Original IR drop map

Modifying Power Pads

Adding new power pads and straps is described, and then analyzing the impact on IR drop.

1. Add two power pads on the two metal4 straps by using **Edit > Add Pad**, as shown in Figure 4-16.

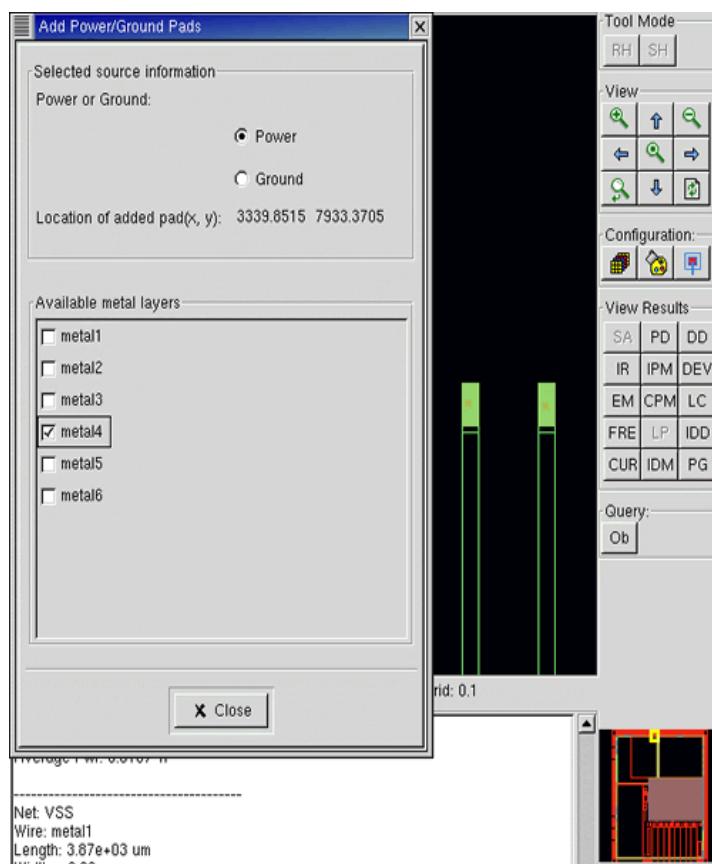


Figure 4-16 Adding two power pads on two metal4 straps

2. Then add one power pad on the metal6 strap using the command **Edit > Add Pad**, as shown in Figure 4-17.

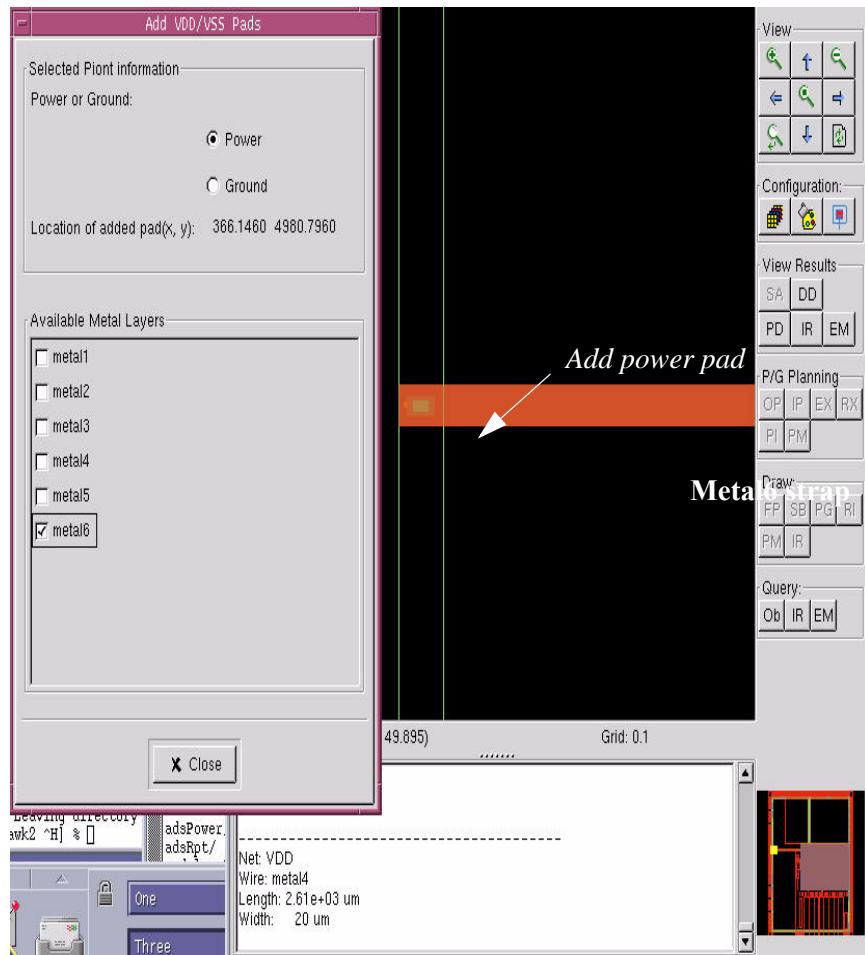


Figure 4-17 Adding a power pad on the metal6 strap

Adding Metal6 Straps

3. Add two metal6 straps on top of the two metal4 straps by using **Edit > Add Power Strap**, as shown in Figure 4-18. This reduces the resistance of the grid segment supplying the hotspot. Make the following selections in the pop-up menu when adding the metal6 straps.
 - Deselect **Add strap by text input** to use drawing input method.
 - Select a **Vertical** power strap.
 - Input the strap width as 20um.
 - Select metal6 for the stack via top metal layer.
 - Select metal4 for the stack via bottom metal layer.
 - Select metal6 for the strap metal layer.

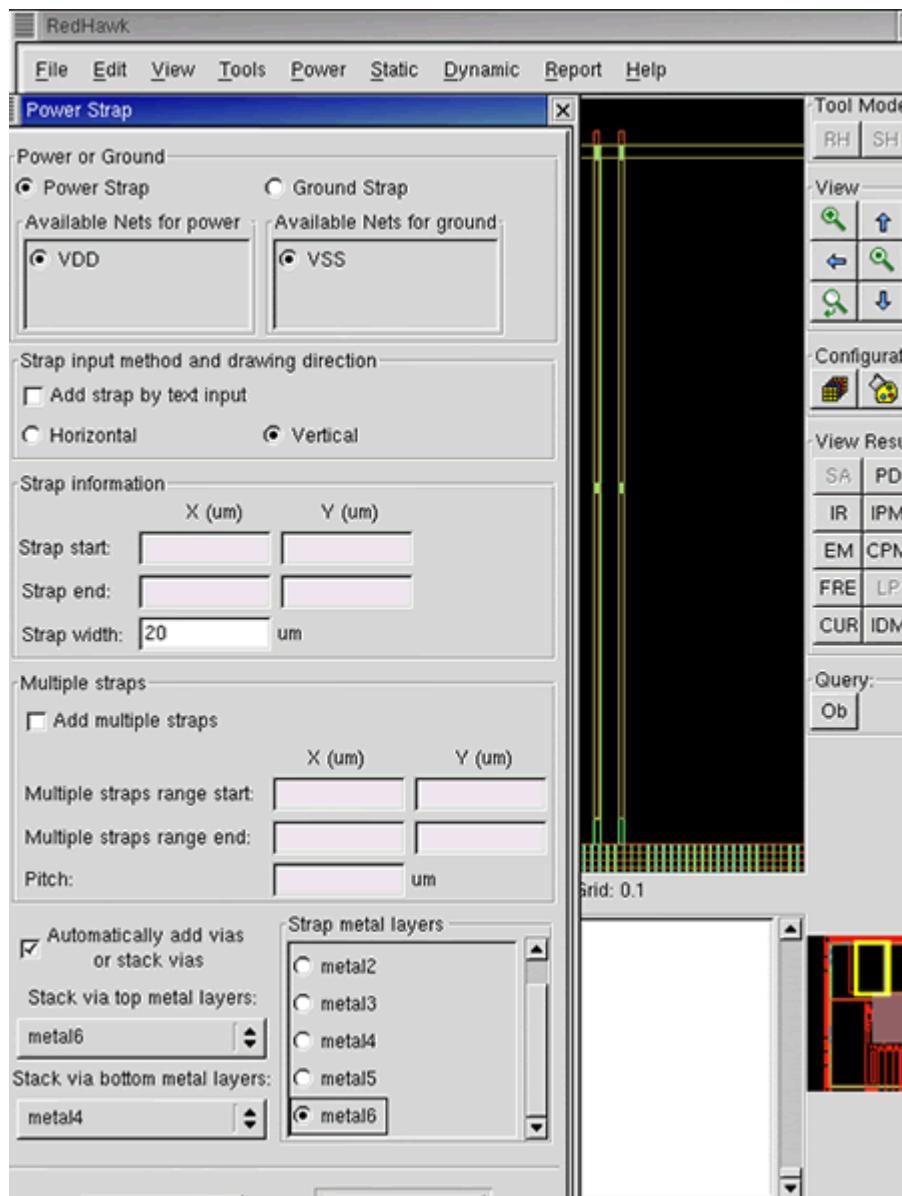


Figure 4-18 Adding two metal6 power straps to two existing metal4 power straps

4. Draw two metal6 power straps that cover the length of the metal4 power straps. This is indicated by a red rectangular box that covers the width and length of the metal4 straps.
5. Click on **Commit Adding** to add the power straps to the design.
6. Now rerun **Static->Network Extraction** and **Static->Static IR-drop & EM analysis**.

The results show that the worst Vdd - Vss differential is now 1.1076V, showing very little improvement from the initial IR drop run.

Resistivity Sensitivity

1. As a way of evaluating its impact on voltage drop, reduce the metal6 resistivity from 0.027 to 0.014 in the RedHawk technology file.
2. Now rerun **Static->Network Extraction**, **Static->Power->Import**, and **Static->Static IR-drop & EM analysis**, using the modified technology file.
The worst Vdd - Vss differential is now 1.109V. This shows slightly more improvement from the previous IR drop analysis.
3. This would indicate that further improvement in IR drop could be obtained by adding an extra layer of metal on top of metal6, or by changing the wire-bond package to a flip-chip package.

If you feel there are additional IR drop problems that need to be resolved before proceeding to Dynamic Analysis, see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#).

When you have finished evaluating and fixing IR drop performance, proceed to [Chapter 5, "Dynamic Voltage Drop Analysis"](#).

Chapter 5

Dynamic Voltage Drop Analysis

Introduction

This chapter describes how to run dynamic voltage drop analysis using **RedHawk-EV**.

As described in [Chapter 3, "User Interface and Data Preparation"](#), the key required inputs for running dynamic voltage drop analysis are:

- LEF files for cell library, including standard cells, memories, and I/Os
- Flat or hierarchical DEF files
- Synopsys .lib library files
- **RedHawk** .tech technology file - conductor and via resistance, dielectric thicknesses and dielectric constants, EM current density limits
- Pad instance, pad cell, or pad location files
- **RedHawk** Global System Requirements (GSR) file, containing information on toggle rates, frequency, clock roots, default slews, and block power
- Timing windows and slews from STA (recommended).
- Extracted parasitics from SPEF or DSPF (recommended)
- Pad, wirebond/bump, or package RLC information (recommended)
- VCD vector file (recommended if available)
- SPICE subcircuits for all memories, I/Os, and IP blocks (optional)
- GDSII for memories, I/Os, and IP blocks (optional)

It is assumed that input data has been prepared as described in Chapter 3, and static IR voltage drop analysis has been performed, as described in Chapter 4.

The following sections describe the dynamic voltage drop analysis methodology and procedures.

Preparing for Dynamic Voltage Drop Analysis

Perform Cell Characterization

1. To create the dynamic current profiles, effective power resistance, and decoupling capacitance for cells in your design, use the APL tool to perform characterization, as described in [Chapter 9, "Characterization Using Apache Power Library"](#).
2. In the GUI, import the dynamic current profiles (`<cell>.current`) generated from APL characterization using the **APL-> Import** command. The decoupling capacitance data (`<cell>.cdev`) is imported using the same form.

Calculate Power

3. If you have already performed power calculation, for example before performing static analysis, you can import the results using the command **Dynamic-> Power-> Import**.

4. If you have not already performed power calculation, refer to [Chapter 4, "Power Calculation, Static IR Drop and EM Analysis"](#), which describes the procedure for setting up the proper GSR keywords and running power calculation.

Network Extraction

5. Perform network RLC extraction using the **Dynamic -> Network Extraction** command. Any combination of R, L and C extraction can be selected, but use all three for best accuracy.
If ECO changes have been made to decaps, vias, or pins and extraction must be performed again, with the GSR keyword EXTRACTION_INC set to 1 (default), an incremental extraction is performed only in the areas of ECO changes. Full design extraction is always performed for changes to wires.

Pad and Package Parameter Setup

6. Set up pad and package parameters by selecting **Dynamic-> Pad, Wirebond and Package Constraints**.

See [Chapter 12, "Package and Board Analysis"](#), for more information about package and board modeling.

Methods of Dynamic Voltage Drop Analysis

There are several methods of vectorless and VCD-driven dynamic voltage drop analysis available in **RedHawk**. The chosen method of performing dynamic voltage drop analysis depends primarily on whether a VCD file is available, and if so, the quality of the VCD information. The goal is to use the best switching information available to construct a realistic switching scenario with the information available. Available methods for performing dynamic analysis are summarized in the Figure 5-1 diagram following.

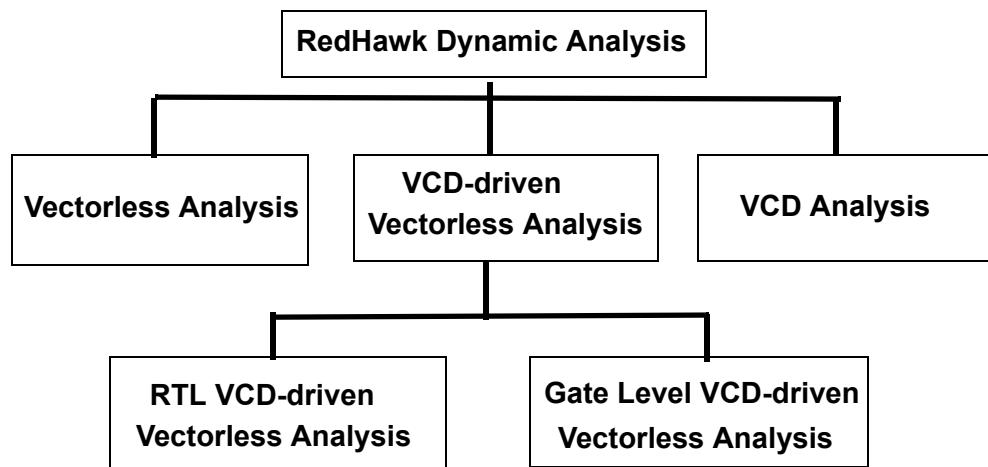


Figure 5-1 Types of RedHawk Dynamic Analysis

Use the diagram, which shows key characteristics of vectorless, VCD-driven vectorless, and VCD dynamic analysis, to help decide what type of dynamic analysis to perform.

Characteristics of vectorless dynamic analysis

- No VCD file available for design.

- Covers realistic worst case switching scenario.
- Uses GSR keywords and the timing file to define realistic toggle rate and switching times. See section "[Setup for Vectorless Power Calculation](#)", page 4-36, for methods of defining the toggle rate accurately.
- Switching scenario is derived statistically and depends on several design-aware factors, such as:
 - areas with structural weaknesses
 - instance timing windows
 - logic/power/toggle rate of instances
- If Boolean function multi-states are defined in either SIM2IPROF or AVM, the state names *should be specified in the GSC file per instance*. RedHawk power calculation and dynamic analysis then automatically use those state definitions in the analysis.
If a GSC file definition is not provided:
 - power calculation uses the average charge of all states
 - simulation randomly selects one customer state to toggle, unless the toggle rate is very low

Characteristics of VCD-driven vectorless dynamic analysis

- VCD file is available, but may not represent the worst case.
- Helps drive vectorless switching scenario creation.
- Two types of VCD files are available.
 - RTL VCD
 - Available in early design stages.
 - Requires mapping RTL names to DEF. Using formal verification tools such as Conformal or Formality, an RTL-to-gate level name mapping file is created. Instance level name mapping, in addition to instance/pin and net level mapping, also can be performed, using the mapping file format:
`adsInst <instance_in_RTL> <instance_in_DEF>`
 - Switching activity at state points (PIs, FF, latches) is propagated through the logic using the state propagation algorithm.
 - Clock gating is inherently supported in this flow
 - For missing VCD coverage, a constraint file can be used to assign toggle rates, in addition to available RTL VCD, to improve switching activity.
 - Realistic toggle rate distribution, which drives vectorless switching scenario creation.
 - Gate-level VCD
 - Gate level VCD can be directly mapped to the design
 - Peak power cycle is identified by scanning through the entire VCD
 - Toggle rates are derived from VCD for all nets and used to guide vectorless switching scenario creation

Characteristics of Event-driven VCD dynamic analysis

- Gate-level VCD with timing back annotation is used for a pure VCD analysis.
- VCD used for representing worst case and for measurement/correlation purposes.
- Peak power cycle(s) are identified by scanning through entire VCD.
- Specific cycle analysis with switching events and delays are derived from VCD.

The following sections describe the procedures for performing each type of dynamic analysis.

Vectorless Dynamic Analysis

Overview

If there is no VCD file available for the design, then a vectorless methodology can be used to analyze the areas of significant dynamic voltage drop. There are two vectorless methods that can be utilized, one that is beneficial early in the design process with preliminary information that is very fast, and also normal vectorless analysis with good information when high accuracy is required:

- Accelerated Dynamic (AD) analysis
 - can be used for fast initial prototyping of the design, but is not intended as a replacement for regular RedHawk analysis (not for sign-off)
 - fast simulation time, with average speed-up of 5-6x
 - average DvD accuracy degradation of 10-15%
 - does not change DvD hot spot trend
- normal vectorless analysis

The input data and setup for both types of vectorless analysis is the same, as described in the following section.

Input Data and Assumptions

For any vectorless type of analysis STA information is required to achieve reasonable accuracy. The following information defined in the RedHawk timing file (compiled from STA and typically named `<design>.timing`) is used for vectorless analysis:

- Clocks have a specified operating frequency, and domain information. Clock buffers toggle twice every cycle, once on each positive or negative transition. An analysis of the expected switching of instances in areas of P/G weakness can be performed using the EVA_PG_AWARE keyword set to 1 (see [section "EVA_PG_AWARE", page C-613](#)) and TOGGLE_RATE set to 0.
- Timing windows for each instance output pin have specified maximum and minimum rise times and maximum and minimum fall times. Instances that belong to a clock network are identified by their clock index number.
- Transition times (“slew”) for each instance pin have specified maximum and minimum rise and fall times. The transition times for input pins are also used during design-specific APL characterization.
- Boolean multi-state definitions can be used in vectorless analysis if *they are defined in the Global Switching Configuration file* (see [section "Global Switching Configuration \(GSC\) File", page C-571](#)).

The following assumptions are made during dynamic analysis for the instances that switch:

- Any instance that is determined to switch and has its timing window in the first cycle of the simulation period will switch in the first cycle of the simulation period. In the next cycle of simulation the instance will also switch, but in the opposite direction.
- For memories, if the instance is in 'write' mode during the first cycle, the instance will be in the 'read' mode during the second cycle.

Standard Vectorless Analysis Procedure

The following are the key steps in performing vectorless dynamic voltage drop analysis:

1. For the most accurate power analysis you must compile and input the best toggle rate information available to indicate the average switching performance of the instances in the design. The general priorities for setting toggle rates using GSR keyword values are listed below, from highest to lowest. However, because there are interactions between values chosen for each keyword in a specific design, see [section "Setting the Toggle Rate", page 4-38](#), or Appendix C, for more information about toggle rate specification and instance toggle rate estimation. The general priorities for toggle rate determination are described in Chapter 4. See [section "Setting the Toggle Rate", page 4-38](#).
2. To control the switching scenario, set the switching state parameters for instances in the Global Switching Configuration (GSC) file and refer to the file with the GSR keyword GSC_FILE.
3. When the best toggle rate information available has been input into the GSR, use the **Dynamic -> Vectorless Dynamic Voltage Drop Analysis** command to start execution.
4. A worst-case switching scenario for the sequential nets is derived by RedHawk based on the user-specified toggle rate information. The switching scenario constructed by RedHawk is constrained to meet the known total average chip power specified in the GSR, so that a realistic switching scenario is obtained.
5. Using the worst-case switching scenarios and current profiles from either the LIB files, or for more accuracy, from APL characterization, RedHawk performs a simulation of peak current contributions from all switching events to obtain a realistic time-varying current over the simulation time at each node.
6. To support multi-state analysis, if Boolean function states are defined in either SIM2IPROF or AVM, and the state names are specified in the GSC file per instance, RedHawk power calculation and dynamic analysis automatically use those state definitions in the analysis.
7. From the time-varying node currents RedHawk computes the associated time-varying dynamic voltage drop at each node over the simulation time.

Note: You can import a different STA file during or after analysis to create a different simulation scenario using an 'import sta' command in the batch file or from the TCL command line.

Unified Clock Gate Handling and Analysis

There are two ways to handle clock gates-- Gated On Percentage (GOP) and Automatic Clock Gate Handling. Both of these types of analysis flows are merged using the STATE_PROPAGATION GSR keyword, with different options. If this keyword is *not* defined, state propagation is *off* and instance switching is determined by default toggle rates in the design. The dynamic part of state propagation is controlled by a separate option, now called FLOP_ON_PERCENTAGE.

Automatic Clock Gate Handling (default)

The GSR setting for basic State Propagation is

```
STATE_PROPAGATION {  
    PROPAGATION_MODE probability  
}
```

With this invocation the gate enable ratio is automatically set at the output of the clock gates (the enable ratio specifies the probability that any clock gate is On, and is the value used to scale the clock toggle rate down). The enable rate is derived by propagating

upstream toggle rate values from the primary inputs. You can turn this off by setting the SP option `AUTO_CLOCK_TRACING` to 0 (On by default).

Two Flow Types

Basic clock gate handling can be divided into two types of flows, Ratio-Based and On/Off-Based. The basic elements of the two flows are shown graphically in Figure 5-2.

- Ratio-Based flow
 - The toggle rates at clock gate outputs are scaled down.
 - Can be used for both static and dynamic analysis.
- ON/OFF-Based flow
 - Each clock gate is set to either ON or OFF.
 - Can be used to control switching scenarios during dynamic analysis.

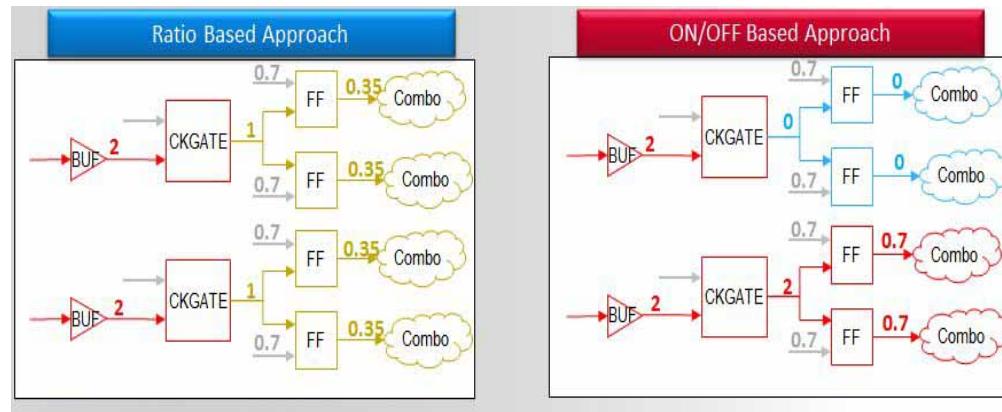


Figure 5-2 Two types of clock gate handling

Ratio-Based Approach

The syntax for the ratio-based approach is:

```
STATE_PROPAGATION {
    PROPAGATION_MODE probability
    CLOCK_GATE_ENABLE_RATIO <ratio>
}
```

In this approach, a single clock gate enable ratio value is specified, which is applied to all clock gates in the design, and the output clock toggle rates are all scaled down by this ratio. Note that this setting is the same as using GOP, along with the TCL command ‘`setup analysis_mode static`’, but a warning is displayed if GOP is also used, and this option is obsolete.

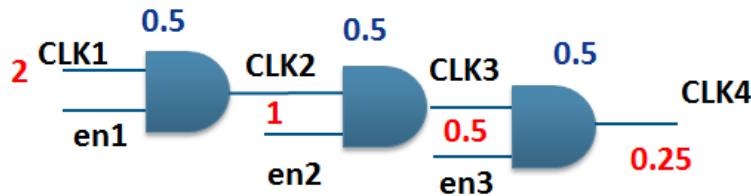
Note that the SP option `CLOCK_GATE_ENABLE_RATIO` by default affects both Inferred and Instantiated clock gates. Instantiated clock gates are a type of clock gate by lib definition, while Inferred Clock Gates are instances such as AND gates, which can be used as clock gates. However, there is also an option to provide separate control for inferred clock gates, as follows:

```
STATE_PROPAGATION {
    PROPAGATION_MODE probability
    CLOCK_GATE_ENABLE_RATIO <>
```

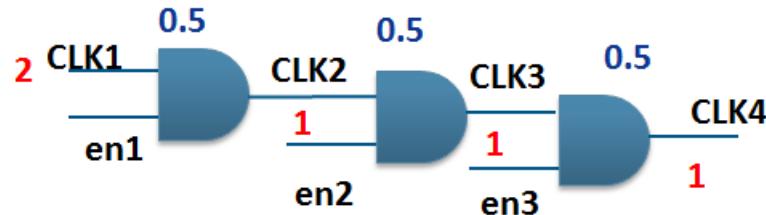
```
    INFERRED_CLOCK_GATE_ENABLE_RATIO <ratio>
}
```

You can set the option “INFERRED_CLOCK_GATE_ENABLE_RATIO 1” if you do not want inferred clock gates to affect the clock toggle rate.

The unified flow also offers the flexibility to choose between *cascaded* and *non-cascaded* modes of clock gate handling. The following diagram explains cascaded and non-cascaded flows in clock gate handling. When the gating ratio is cascaded, the gating ratio at any gate depends on the upstream gates and their probability of its switching. The following is an example of cascaded flow:



And the following is a non-cascaded flow.



The default flow is *non-cascaded*, to be consistent with the traditional GOP flow. You can enable cascaded-type handling (which is more realistic) using the following setting:

```
STATE_PROPAGATION {
    PROPAGATION_MODE probability
    CLOCK_GATE_ENABLE_RATIO <ratio>
    ENABLE_CASCADED_CLOCK_GATING 1
}
```

Custom clock gate enable ratio values also can be specified for desired clock gates with the help of a user-defined clock gates file. The keyword setting is:

```
STATE_PROPAGATION {
    PROPAGATION_MODE probability
    CLOCK_GATE_ENABLE_RATIO_FILE <cg_ratio.file>
}
```

where the <cg_ratio.file> format is:

```
#<clock_gate_name> <enable_ratio>
...
```

ON/OFF-Based Approach

In the ON/OFF-based gate handling approach, each clock gate is set either ON or OFF. This is the same as the dynamic behavior of the previous similar option GATED_ON_PERCENTAGE. The GOP option name is changed to FLOP_ON_PERCENTAGE to reflect more accurately the behavior of the option. When the FOP fraction is set 0.5, it does not mean that 50% of the clock gates are ON, it only means that 50% of flops are

ON. This is achieved by controlling clock gates upstream of these flops. PowerStream finds the number of flops controlled by each clock gate, looks at the specified FOP fraction value, assesses the number of flops to be turned ON to reach the target, and decides which clock gates must be set ON to achieve this target. The syntax is:

```
STATE_PROPAGATION {  
    PROPAGATION_MODE probability  
    FLOP_ON_PERCENTAGE <FOP_fraction>  
    FOP_CONTROL_FILE <cg_on_off.file>  
}
```

Just as for the Ratio-based approach, there is also a custom flow setting file for the ON/OFF approach, using FOP_CONTROL_FILE <cg_on_off.file>, where the <cg_on_off.file> format is:

```
#<clock_gate_name> <ON/OFF>  
...
```

RTL VCD-Driven Vectorless Dynamic Analysis

Designs with gate level VCD files have a complete set of vectors available for accurate power analysis at the instance level. However, for designs with RTL level VCD data, detailed switching information at the instance level must be estimated. Designers often want to employ unit-delay or true-timing gate-level simulation results as stimuli to achieve very accurate DvD analysis. However, true-timing gate-level simulation data often takes a prohibitive amount of time to generate, and it is often not available until the end of the design cycle.

RedHawk provides a state propagation engine to achieve a statistically realistic method of evaluating power in both clock and logic circuits under complex switching conditions, such as clock gating methodologies, without gate-level simulation data. This method automatically produces an accurate set of toggle rates for each instance in the design.

RedHawk supports state propagation-based power calculation in the RTL-VCD flow, as well as event propagation-based power calculation. In the RTL VCD-based State Propagation flow the toggle rates for the primary inputs and flop/latch outputs are calculated from the RTL VCD, and then this activity is propagated using State Propagation. This provides a compromise between RTL VCD-driven Event propagation and probabilistic State Propagation, which is useful when you want to calculate power for a longer duration of the VCD, and event propagation engine would have a run time penalty. State propagation analysis is controlled using STATE_PROPAGATION GSR keyword options, as described in Appendix C.

Data Requirements for State Propagation

RTL VCD or FSDB files

VCD is the output of functional simulators such as VCS, NC-Verilog and ModelSim. FSDB is a binary and reduced-size version of the VCD file. Both block-level and top-level VCD or FSDB files can be imported into **RedHawk**. A switching toggle rate is extracted from them for a given time frame (or for the entire VCD simulation period) for all state points. You can also set toggle rates, or override those in VCD file, using a constraint file. For nodes missing in the VCD file, default toggle rates can be used.

Mapping files

RTL level netlist names map to different gate-level DEF netlist names. RedHawk requires a mapping file in a two-column format, associating each node in the RTL level with the corresponding DEF netlist name, with the RTL names on the left and the DEF names on the right. This mapping information can be easily extracted from tools like Conformal and Formality. Note that in the RTL VCD mapping file the RTL names are on the left and DEF names are on the right.

Mapping inverted elements

The RedHawk RTL VCD flow supports inverted mapping, such that while annotating from RTL VCD, an inversion is performed before annotating activity to the QN pin. The AutoNameMapping function maps inverted flops using the following syntax in the name mapping file:

```
<RTL_signal_path_RTL_VCD> <Gate_pin_path_design> -inverted
```

For example,

```
top/block/count top/block/count_reg/QN -inverted
```

For all inverted mapping, RedHawk first inverts the signal state in RTL VCD and then does annotation to the GATE pin.

GSR Keywords to Support RTL VCD Flow

The Global System Requirements file keywords VCD_FILE (see [section "VCD_FILE", page C-609](#)) and BLOCK_VCD_FILE (see [section "BLOCK_VCD_FILES", page C-584](#)) specify RTL VCD files, along with parameters affecting how the VCD files will be imported. If multiple VCD files need to be imported, use the BLOCK_VCD_FILE keyword.

The VCD_FILE option FILE_TYPE should be set to RTL_VCD or RTL_FSDB, and the option VCD_DRIVEN should be set to "1", then the State Propagation engine is activated in RTL VCD-driven mode. It scans the RTL VCD to derive the toggle rate at state points (FF/registers), and propagates the toggle rate to downstream logic. See [section "Global System Requirements File \(*.gsr\)", page C-573](#), for more information on using these RTL-VCD related keywords.

State Propagation Constraint File

For nodes that are missing in the VCD/FSDB files, toggle rate constraints can be set through a constraint file designated with the CONSTRAINT_FILE option of STATE_PROPAGATION. A template of this constraint file listing all the state points in the design can be generated with the RedHawk TCL command 'dump_sptemplate'. The generated template file from this command is *adsRpt/state_propagation.template*. An example of a state propagation constraint file is shown below:

```
# analysis mode: static
# Format: <port/pin> <toggle_rate (2.0-based)>
# 1. Clock roots
clk_A 0
clk_B 1.8

# Primary inputs
scan_mode 0.05
scan_en 0.08

# Register outputs
i_pram_00/mem3/do[11] 0.15
```

Note: The maximum toggle rate is 2.0, which means that the signal makes 2 transitions (0>1 and 1>0) every cycle. Fully active clocks have a toggle rate equal to 2.

You can propagate a toggle rate of 2 to MUXs during state propagation, independent of the 'Select' pin toggle rate, using the GSR keyword 'SP_CLKMUX_AUTO 1'. Since it is very tedious to identify the Select pins of multiplexers and assign the correct state to them, a toggle rate of 2 can be assigned from a multiplexer onward in the clock network.

When using STATE_PROPAGATION without any VCD files, toggle rates are still propagated based on the constraint file or the default toggle rate. Using the constraint file, you can specify toggle rates for clock roots, primary inputs, and outputs of registers. For memories, both input and output toggle counts need to be included.

Following setting up the appropriate keyword values and constraint files for estimating toggle rates and performing state propagation, RedHawk can run static and dynamic analysis normally without any additional command.

RTL VCD Reports

There are several reports generated by RTL VCD analysis:

- the *adsRpt/vcd_uncovered_syms* file contains the VCD symbols that are not in the design.
- the *adsRpt/vcd_uncovered_mems* file contains the memory instances that are not covered in the design.
- the *adsRpt/vcd_uncovered_ffs* lists ffs/latches that are not covered and are not in the clock network
- the *adsRpt/vcd_uncovered_clk_ffs* file lists ffs/latches that are not covered and in the clock network

Also note that several other reports are generated if the PS_RTL_EP_REPORT GSR keyword is set, as follows:

- *adsRpt/vcd_covered_ffs* : list of ff/latches that are “covered” in VCD/FSDB, meaning those in which one of the outputs is covered.
- *adsRpt/vcd_covered_nets* : list of nets that are covered in VCD/FSDB
- *adsRpt/propagated_insts* : list of instances that are propagated
- *adsRpt/propagated_nets* : list of nets that are propagated through

Gate Level VCD-Driven Vectorless Dynamic Analysis

Input Data and Assumptions

To use the VCD-driven vectorless method in a hierarchical design, specify the VCD file in the GSR file, along with the hierarchical prefix in the VCD and the start and end times, as follows:

```
BLOCK_VCD_FILE {
    VCD_FILE
    {
        <Hierarch_path/DEF_instance_name1> <VCD file>
        FILE_TYPE <VCD | FSDB | RTL_VCD | RTL_FSDB >
        FRONT_PATH <redundant_path_string>
        SUBSTITUTE_PATH <substitute_path_string>
        FRAME_SIZE <cycle_time>
        START_TIME <analysis_start_time>
        TRUE_TIME [0|1]
```

```

        MAPPING <filename>
    }
    VCD_FILE
    {
        <Hierarch_path/DEF_instance_name2> <VCD file>
        ...
    }
}

```

where

<Hierarch_path/DEF_instance_name> <VCD file>: specifies the hierarchical block name that matches that identified in the DEF and the absolute or relative path to the VCD or FSDB file

FILE_TYPE [VCD | FSDB | RTL_VCD | RTL_FSDB] : identifies the type of input FRONT_PATH <redundant_path_string> : specifies the string that does not match the DEF path. Must be replaced by SUBSTITUTE_PATH string.

SUBSTITUTE_PATH <substitute_path_string>: the substitute path string to match the DEF hierarchy.

FRAME_SIZE <value_ps> : specifies the duration per frame for cycle-by-cycle power calculation; the cycle time in ps (1/FREQ); default is 5000 ps.

START_TIME <value_ps> : optional, specifies the analysis start time in the VCD for power calculation; start default time = 0, end default time is end of VCD/ FSDB file.

TRUE_TIME : if =0, uses STA timing data and assumes no glitches; if =1, uses VCD switching and timing data; default=0.

MAPPING <filename> : name of map file with RTL VCD instance names and corresponding DEF instance name

Analysis Procedure

The following are the key steps in performing VCD-driven dynamic voltage drop analysis:

1. After putting the appropriate information into the VCD_FILE GSR keyword, use the **Dynamic -> Vectorless Dynamic Voltage Drop Analysis** command.

Alternatively, the TCL command for invoking vectorless dynamic analysis is:

```
perform analysis -vectorless
```

2. For a VCD-driven vectorless dynamic simulation, the switching and transition times are obtained automatically from the STA file. The switching scenario is created based on the toggle rate obtained from the VCD.
3. Using the worst-case switching scenarios and current profiles from either the LIB files, or more accurately, from APL characterization, RedHawk performs a simulation of peak current contributions from all switching events to obtain a realistic time-varying current waveform over the simulation time at each node.
4. From the time-varying node currents RedHawk computes the associated time-varying dynamic voltage drop at each node over the simulation time.
5. The worst case node voltages are then compiled into an ordered list.

VCD Dynamic Analysis

Input Data and Assumptions

If the VCD file for the full chip or block level is available and the timing information is accurate in the VCD files, then VCD dynamic analysis can be used. Accurate timing in the VCD can be obtained if delays (that is, through SDF) are back-annotated during simulation. In the case of VCD dynamic analysis, the following assumptions are made:

1. **Switching scenario.** The switching scenario is determined entirely from the VCD files. Every active net should be described in the VCD; for any net not in the VCD file, it is assumed that instances attached to the net are inactive.
 2. **Instance switching.** Any instance that is determined to switch from the VCD file will switch in simulation, with the switching time defined in the VCD file. STA timing windows are not used.
- Note:** Instances specified in the GSC file override the VCD file net specification.
3. **Transition times.** Switching instances use the transition times in the STA file. If an STA file is not provided, the value of the INPUT_TRANSITION keyword in the GSR is used for every instance in the design. Therefore, using STA file values is strongly recommended.
 4. **Cycle Selection.** For automatic cycle selection by RedHawk, set the following option of the VCD_FILE GSR keyword:

```
VCD_FILE
{
  ...
  SELECT_RANGE <start_time> <end_time>
  ...
}
```

If -1 is set for start and end times, the full VCD period is included in cycle selection.

5. **Simulating entire VCD.** When DYNAMIC_SIMULATION_TIME is set to 0 and START-END_TIME are not specified, entire VCD is used for power calculation and simulation.
6. **Multi-state Boolean functions.** Boolean definitions of the multi-states are checked in the VCD file. Where there is a match with definitions in AVM or SIM2IPROF, the state is triggered using the current profile defined for the state in AVM or SIM2IPROF.

VCD Critical Cycle Selection

When the critical cycle selection feature is used, RedHawk automatically selects the most critical cycles from a large vector dump. Cycle selection supports both True-time and non-True-time gate and RTL level VCD analysis in power calculation. The method calculates the power for each cycle and finds the cycle consuming the highest power. The time span (width of each cycle) is decided by the keyword DYNAMIC_SIMULATION_TIME. Each span is swept by a small time-step and power is calculated across this span using a sliding window.

An ordered list of critical cycles identified during cycle selection is automatically provided in the report file *adsRpt/worst_power_cycle.rpt*.

PRUNE Feature in RTL VCD flow

The RTL VCD PRUNE feature improves the performance of RTL_VCD cycle selection by optimizing selection of candidate power-intensive cycles for event propagation, and reducing the total time required for cycle selection. The key features of Pruning are:

- No event propagation - The algorithm extracts net weights from the design directly and does not need to propagate the events. The toggling nets determine the cycle powers directly.
- No timing delay calculation - It is assumed that a net toggling results in subsequent toggling immediately. This saves most of the work on clocking analysis.
- VCD-covered nets only - Typically an RTL-VCD case has VCD coverage less than 10%. The algorithm focuses on the VCD-covered nets only, and hence reduces memory usage and I/O operations.
- Block-based pruning methodology (that is, pruning analysis performed on each block separately when multiple RTL-VCD files are present)
- RTL-VCD file reuse (pruning results are reused for duplicate blocks)
- Support for cycle selection mode (VCD_FILE option WORST_DPDT_CYCLE).
- Support for VCD_FILE option STA_VCD_FREQ_RATIO and GSR keyword POWER_VCD_NUM_PROCESS.

Pruning is most effective for the following design conditions:

- Extensive VCD/FSDB data - The overhead to collect pruning data offsets the performance gain if the cycle number is not high.
- Progressive power profiles - Pruning is more effective when switching intensity changes in a progressive manner. It is more difficult for cycle powers that vary within a narrow range and in a rapidly oscillating fashion.
- Sporadic impulsive power changes - Pruning is most effective if the cycle power profile has sporadic impulsive changes. The time spans between the pulses can all be pruned.

The pruning process is enabled by adding the ‘PRUNE 1’ option of the VCD_FILE GSR keyword, such as:

```
VCD_FILE {
    core_top top.vcd
    FILE_TYPE RTL_VCD
    PRUNE 1
    SELECT_RANGE 10000 100000
}
```

Multi-Bit Sequential Cell Handling in VCD Analysis

Multi-bit sequential cell handling can include individual edge-triggered events, where all input/output pins do not have to trigger off the same event, such as the clock toggle. Multi-bit sequential cell handling considers edge-triggered events if they are defined using the sim2iprof configuration file COMPOSITE_STATE keyword (see section “COMPOSITE_STATE”, page E-926).

Mixed Mode VCD and Vectorless Analysis

RedHawk supports input data including both VCD and Vectorless settings, which provides more freedom to fine-tune the settings at block level. Some key features are:

- A block level VCD with a top-level vectorless approach is supported, where you can provide one or more block VCDs.
- The GSR keyword BLOCK_VCD_FILE defines one or more VCD blocks for analysis. Each block is specified using the VCD_FILE option with a block name (hierarchy path).
- Mixed mode allows you to specify a mix of True time and Non-True time VCDs.
- S-parameter package models are supported.

- CPM creation is enabled.
- Allows for scaling VCD and STA frequencies with STA_VCD_FREQ_RATIO. Note that block level STA_VCD_FREQ_RATIO is supported only in Mixed Mode.
- GSR keywords DYNAMIC_SIMULATION_TIME and DYNAMIC_PRESIM_TIME define the VCD simulation time span. VCD_FILE 'Start_Time' is honored, but 'End_Time' is not used. DYNAMIC_PRESIM_TIME defines the presim time span at 'Start_Time' backward. Auto Presim, that is the default DYNAMIC_PRESIM_TIME, or set to -1, is honored.
- Low power analysis is enabled in mixed mode. For procedures to run dynamic voltage drop analysis on low-power designs, see [Chapter 13, "Low Power Design Analysis"](#).

The following GSR keywords are supported for non-VCD blocks:

- GSC_FILE
- BLOCK_POWER_FOR_SCALING(_FILE)
- STATE_PROPAGATION
- STA_FILE CONST
- TOGGLE_RATE_RATIO_COMB_FF
- INSTANCE_TOGGLE_RATE(_FILE)
- BLOCK_TOGGLE_RATE(_FILE)
- TOGGLE_RATE

If you provide a block VCD file for a block that has a sub-block not covered in the VCD, then the instances inside sub-block are always Off, unless otherwise specified in the GSC. If simulation time is a multiple of FREQ, then the multi-cycle vectorless approach is used for the instances that are not covered by VCD.

Analysis Procedure

1. To use VCD files in the analysis, define the desired VCD constraints and parameters in the GSR keyword VCD_FILE.
2. Invoke the **Dynamic -> VCD-based Dynamic Voltage Drop Analysis** menu selection in the GUI, and select the **Use VCD Timing** option in the dialog box. The start and end times for VCD simulation can be for any duration in the VCD file. The utility *vcdscan* can also be used to determine the start and end times for simulation.
3. For worst-case DvD cycle selection, the procedure is:

```
setup design
perform pwrcalc # calculates power for the critical cycle
perform extraction
setup package
perform analysis -vcd # dynamic run for the critical cycle
```

A sample portion of a GSR file for a mixed mode design follows:

```
...
FREQ 200e6
TOGGLE_RATE 0.2
BLOCK_VCD_FILE
{
    VCD_FILE
    {
        ABCD10_SEL user_data/vcd.1
        front_path "TOP/CHIP/"
```

```
        substitute_path ""
        start_time 100000
        true_time 1
    }
    VCD_FILE
    {
        ABCD20_TEG user_data/vcd.2
        front_path "TOP/CHIP/"
        substitute_path ""
        start_time 600000
        true_time 1
    }
}
DYNAMIC_PRESIM_TIME 10n
DYNAMIC_SIMULATION_TIME 20n
...
```

This example has two VCD blocks, ABCD10_SEL and ABCD20_TEG. The ABCD10_SEL block has a VCD presim time (90ns-100ns) and simulation time (100ns-120ns). The ABCD20_TEG block has a VCD presim time (590ns-600ns) and simulation time (600ns-620ns). The remaining sections of the design will be handled as vectorless.

Gated Clock Dynamic Analysis

Gated clock control activity is typically defined in the VCD file, and, in the case of RTL-VCD flow, it is propagated through the clock network and logic. However, if no VCD is available, the gated clock domain activity is controlled using the GSR keyword STATE_PROPAGATION and option GATED_ON_PERCENTAGE. The state propagation engine is used to propagate the assigned logic level at the gated clock control throughout the downstream clock network, as well as through the data paths controlled by this clock domain. State Propagation uses a vectorless method to model statistically several aspects of gated clock domain activity for static/dynamic analysis. State propagation automatically identifies gated control nets through back-tracing from FF clock pins or STA data.

Input Data

The required input data for gated clock domain analysis is as follows:

- Technology file
- LEF file that describes macros placed in the design
- Top-level DEF file
- Voltage source locations (from PLOC/DEF file)
- Synopsys Library models (.LIB models)
- STA/TIMING file, having TWs defined for all the instances in clock tree network
- SPEF file (optional)

GSR Keywords for Clock Domain Gating

State propagation allows users to control gated clock domain activity using the GSR keywords STATE_PROPAGATION and its option GATED_ON_PERCENTAGE <On_fraction>, GATED_CONTROL_FILE <control_filename>, and SP_CLOCK_GATING_RATIO.

State Propagation can find the gating ratio at each clock gate automatically by determining the duty cycle at each enable net and propagating toggle rates from upstream logic to obtain gating ratios for each clock gate in the design. For RTL_VCD State Propagation this helps achieve very accurate power values for sequential logic. For vectorless State Propagation, this provides a much better estimation of chip power, as the gating ratio is derived through propagation from primary inputs. The GSR keyword PS_SP_GATED_CLOCK_LOGIC can be set to 0 to disable this feature.

For more control over the gating ratio, set a global gating ratio for all clock gates in the design, using the GSR keyword 'SP_CLOCK_GATING_RATIO <ratio>', where the <ratio> value must be between 0 and 1.

Gated clock control activity can be specified globally using GATED_ON_PERCENTAGE, which counts flop/latches/memory instances at leaf level, and includes always-on leaf cells. Note that by specifying GATED_ON_PERCENTAGE as the fraction 0.5, for example, RedHawk randomly picks 50% of the gated control cells and turns them "On". However, for dynamic analysis this may not ensure that the clock network power is scaled down to exactly 50%, since it depends upon the gated clock control granularity. Individual settings for certain gated clock instances can be specified using the GATED_CONTROL_FILE option, with the remaining unspecified individual states being assigned according to the global On setting.

Modeling methods and the associated keywords for clock gating are described below. The POWER_TRANSIENT_ANALYSIS GSR keyword can be used to specify configuration files to define gated clock control activities for each time frame of interest. Multiple time frames define different clock activities over defined groups of cycles.

GSR Settings for State Propagation

State propagation analysis is controlled using the following GSR syntax:

```
STATE_PROPAGATION {  
    GATED_ON_PERCENTAGE <On_fraction>  
    PROPAGATION_MODE PROBABILITY  
    ? CONSTRAINT_FILE <constraint_filename> ?  
    ? GATED_CLOCK_COVERAGE [ 0 | 1 ]?  
    ? GATED_CONTROL_FILE <control_filename>?
```

where

GATED_ON_PERCENTAGE <on_fraction>: specifies the percentage of buffers that are on, as follows:

All gating cells are on if GATED_ON_PERCENTAGE is 1

All gating cells are off if GATED_ON_PERCENTAGE is 0

Increasing the value of GOP turns on additional buffers, and does not turn off any that are already turned on. So, any clock buffer that is on with a GOP of, for example, 0.6, is also on when GOP is set to 0.8.

PROPAGATION_MODE : PROBABILITY automatically determines switching probability for instances.

CONSTRAINT_FILE : specifies a file describing the toggle rate of key instances or nets for accurate toggle rate propagation.

GATED_CLOCK_COVERAGE : when set to 1, provides better design coverage when using GATED_ON_PERCENTAGE, and allows choosing different clock gate turn-on scenarios for each frame so that overall analysis coverage is increased. Note that POWER_TRANSIENT_ANALYSIS must also be turned on.

GATED_CONTROL_FILE : specifies a file defining the On/Off status of individual gated control cells, with the following syntax: The format of the gated control file data is as follows for both static and dynamic analyses:

```
<gated_clock_instance> [ On | Off ]
```

Setting State Propagation Constraints

The State Propagation flow has a convenient method for assigning gating ratios for clock gates, as well as activity factors for primary inputs. Even before the power calculation step, RedHawk can dump files containing all clock gates and primary inputs. You can then edit these files to set custom clock gating ratios at the required clock gates, and also specify activity at the desired primary inputs. State Propagation then uses these settings to determine toggle rates for all nets in the design.

The Tcl command is:

```
dump sp_constraints
```

This command dumps files *state_propagation.cg* and *state_propagation.pi* in the *adsRpt* directory. The formats of these files are shown below:

Format for file *adsRpt/state_propagation.cg*:

```
<clock-gate_instance> <gating_ratio> <level>
```

Format for file *adsRpt/state_propagation.pi*:

```
<net-name> <toggle_rate> <duty_cycle>
```

These files can be edited to give specific gating ratios to clock gates, and also to set activity for the primary nets. All clock gates whose gating ratios have not been defined through a constraint file have their gating ratio derived by default from upstream logic, or from global gating ratio keyword settings. The edited files can be passed back to RedHawk using two GSR keywords.

For Clock Gates:

```
SP(CG)_CONSTRAINT_FILE state_propagation.cg
```

For Primary Inputs:

```
NET_TOGGLE_RATE_FILE state_propagation.pi
```

If there is already a *NET_TOGGLE_RATE_FILE*, then the constraints file *state_propagation.pi* is appended to it.

***adsRpt/state_propagation.GCControls* file**

Each state propagation run dumps out a list of all gated clock control cells' On/Off states in the *adsRpt/state_propagation.GCControls* file. If you have your own list of control signals present in the design, you can modify this file with your list and use it as input to the **GATED_CLOCK_CONTROL** option. The format of the file is (per gate):

```
<gc_instance_name> <cell_type> [On | Off] <user_setting> <On_fraction>
```

where

gc_instance_name : gated clock instance name

On/Off : whether the gate is set to On or Off

user_setting : 0 - no user setting , 1 – user set to On, 2- user set to Off

On_fraction: actual realized Gated On Percentage so far. The *<On_fraction>* for static analysis is the On-percentage/100 for a gated clock instance. For dynamic analysis the *<On_fraction>* is the cumulative On percentage up to the gated clock instance, in ascending order.

Troubleshooting Files and Tips

Files that can be helpful in looking at problems are:

- *adsRpt/state_propagation.template*
- *adsRpt/state_propagation.GCControl*
- *.apache/apache.tr* - toggle rate information for each net
- *.apache/apache.tr.i* - toggle rate information for each instance

The following steps can be helpful in finding issues in clock domain analysis:

1. Make sure *.lib* models are defined for all cells present in the clock tree network.
2. Make sure LEF models are defined for all macros used in the design/clock tree network.
3. Check for *adsRpt/apache.refCell.** files in the *adsRpt* directory.
4. Make sure all the instances present in the clock tree network have timing windows defined in STA/TIMING file.
 - Check *adsRpt/apache.tw0* for a list of instances that have missing timing windows.
 - State Propagation gets the clock network information based on the timing file read in.

Scan Mode Dynamic Analysis

One of the largest peak current events in SOC designs occurs during scan chain activation. Since clock skew is always minimized as standard practice, all toggling occurs almost exactly at the same time. The subsequent propagation of changes races down the logic networks. Even if the test frequency is slow relative to the functional frequency, all current consumption occurs in the short period following the edges of the clock.

For example, as shown in a typical scan current analysis in Figure 5-3, the high current peaks in scan mode can be 10-100 times the current in normal mode due to the simultaneous switching of most of the registers.

A sample output is shown in Figure 5-3 below:

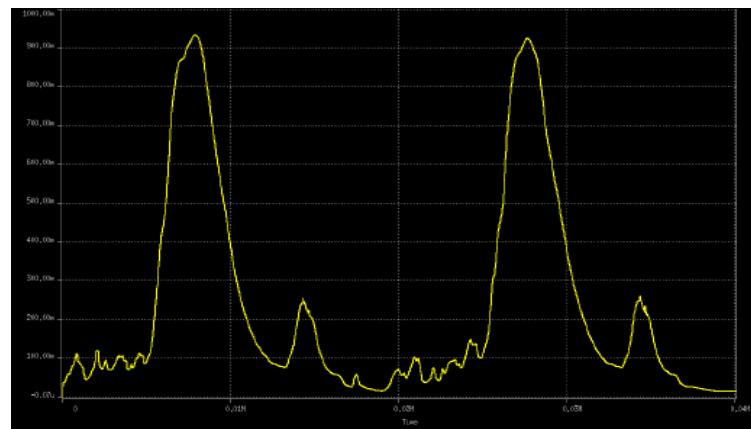


Figure 5-3 Switching current from scan mode analysis

Early design vectorless scan mode analysis

Full VCD gate-level scan mode analysis is the same as for functional mode analysis, and is described in section "VCD Dynamic Analysis", page 5-91.

Gate-level timing-annotated VCD with scan activity is usually available very late in design cycle. Therefore, evaluating potential scan clock network problems early in design (based on initial placement) by using this scan methodology, in which RedHawk generates the scenario files from constraint file specifications, can avoid many cases of chip failure and yield reduction.

The scan mode flow is a non-true timing flow, in which PowerStream decides the sequence of switching, but the actual time of switching is determined by the STA file. RedHawk can drive the data outputs of scan registers according to the scan outputs, and their behavior is based on the functions described in LIB file.

GSR keywords DYNAMIC_SIMULATION_TIME and PRESIM_TIME are also honored in this flow

Key characteristics

- Toggle activity at scan flip-flops derived from input pattern
- Toggle activity at scan flip-flops propagated through combinational logic
- Scan input patterns can be simple, such as “1010” (stress pattern), or more design-specific from ATPG
- Activity at all nodes in design highly deterministic (not random)

Input Scan File Preparation

The input data required for early scan mode analysis requires only preparation of the scan constraint files, and specifying the files and the required patterns using the SCAN_CONSTRAINT_FILES GSR keyword, as follows:

```
SCAN_CONSTRAINT_FILES {
    <scan_chain_file_1> <pattern_1>
    <scan_chain_file_2> <pattern_2>
    ...
}
```

The format of the constraint file contents is:

```
<inst_name>/<pin_name>
```

Example contents:

```
FF_inst/Q
...
```

Example 1: All scan chains have the same pattern, 01:

```
SCAN_CONSTRAINT_FILES {
    scan_chain.spc
}
```

If no pattern is specified, the default pattern “01” is assumed.

Example 2: All scan chains have the same pattern, 0011:

```
SCAN_CONSTRAINT_FILES {
    scan.spc 0011
}
```

Example 3: Multiple scan chains, each has its own pattern:

```
SCAN_CONSTRAINT_FILES {
```

```
scan_chain_1.spc 01
scan_chain_2.spc 101
scan_chain_3.spc 0011
...
}
```

Example 4: Using a scan chain order file. You can provide the actual scan chain information, along with the scan pattern, using a scan chain order file. Each scan chain order file contains one scan chain. A sample scan_chain_order file is shown below:

```
reg_1/Q          # driver for reg_2
reg_2/Q          # receiver for reg_1, driver for reg_3
reg_3/QB         # receiver for reg_2
...

```

Scan-chain order files dumped from the implementation environment can be converted in the above format using simple scripting. The switching sequence of the scan fflops is determined by the given pattern, whereas switching time is annotated from the STA file. The switching pattern can be directly specified in the GSR-- for example, “10101”, or using a pattern file (in case the pattern is long). If the pattern length is less than the chain length, the pattern is repeated for the remaining chain. If the pattern or pattern-file is not specified, a pattern of “0101...” is used.

Sample pattern files, either a single bit in each line:

```
0
1
1
0
...

```

or a whole pattern in a single line:

```
0110...
```

During simulation, the pattern is shifted, and that determines the switching activity on the fflops, as well as subsequent data-paths. For example, consider there are 10 registers in the chain, and the scan pattern is “1001”. RedHawk repeats the pattern to make it the same length as the scan-chain length -- so “1001100110”. During simulation, the last fflop output switches from 0 to 1, second to last fflop output switches from 1 to 1, and so on. The same pattern is continued in subsequent clock cycles, as shown below:

```
0 1 1 0 0 1 1 0 0 1 - at the beginning of simulation
0 0 1 1 0 0 1 1 0 0 - after first clock cycle
1 0 0 1 1 0 0 1 1 0 - after second clock cycle
```

Auto Tracing Scan Method

RedHawk also can perform auto-tracing of scan chains, using the AUTO_TRACING option, as follows:

```
SCAN_CONSTRAINT_FILES {
    AUTO_TRACING [<pattern>]
}
```

When auto-tracing is specified, PowerStream finds all scan chains in the design and applies the specified <pattern> to those chains.

Running RedHawk analysis

The normal dynamic analysis flow commands for scan mode are:

```
setup design <design>.gsr
perform pwrcalc
perform extraction -power -ground -c
perform analysis -vcd
```

The -vcd option has to be specified in the ‘perform analysis’ command so that simulation picks up the scenario created by scan flow.

Gate-level VCD scan mode analysis

Scan mode analysis flow is the same as normal functional mode gate-level VCD analysis, as described in [section "VCD Dynamic Analysis", page 5-91](#).

Key characteristics

- Determines worst power or worst dynamic voltage drop cycles
- Activity derived in chosen cycles from VCD
- Dynamic analysis replicates scenario in VCD

Evaluating DvD Analysis Results

Types of DvD Results

After completing dynamic analysis, select the **IR** menu button to display the results.

RedHawk-EV outputs include:

- Dynamic voltage drop contour maps
- Power density and current maps
- Capacitance maps, including decap effects
- Report files (see [Chapter 6, "Reports"](#)), including summary files for dynamic voltage drop, power, EM and pad current, Error and Warning reports, and command log files.

To view a list of high voltage drop instances and their VDD-VSS values, use the **Results -> List of Worst DVD Instances for Dynamic Simulation** command. A table is displayed showing instances in order of highest DvD, and a number of other related parameters, including

- ‘Ave DV’ - average dynamic voltage within the timing window
- ‘Max DV’ - maximum dynamic voltage within the timing window
- ‘Min DV’ - minimum dynamic voltage within the timing window
- ‘Min DV WC’ - minimum dynamic voltage for the whole cycle
- location of instance
- name of instance

The list can be sorted based on any of the four DV parameters.

- ‘Max VDD-VSS’
- ‘Min VDD-VSS’ over the simulation cycle
- ‘Min VDD-VSS over timing window’ of the instance
- ‘Avg VDD-VSS over the timing window’.

These parameters are shown in Figure 5-4 below.

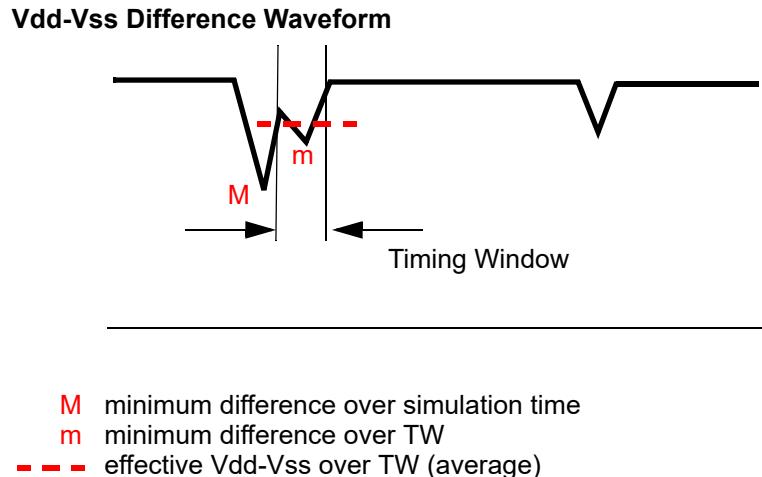


Figure 5-4 Instance DvD Analysis Criteria

The highest voltage drop instances can also be viewed in a voltage drop map by selecting **'View -> Dynamic Instance Result'** and then selecting one of the four maps.

- View the wire based voltage drop map
- View the instance based voltage drop map
- View the decoupling capacitance used in the analysis using the **DD** button.
- View the dynamic power density using the **PD** button.

Note the differences between each.

All the log files and data files are under *adsRpt* directory. Look in that directory to see the files that have been created. Files specific to a dynamic analysis are kept in the Dynamic sub-directory.

The file *<designName>.ipwr* has the time domain current information of the total current demand by all instances in the design, while the file *<designName>.ivdd* has the time domain current information for the current supplied by the pads.

Power and ground waveforms can be plotted using the TCL commands:

```
plot current [-vdd | -gnd | -pwr | -net ?-name <net_names> ?|  

             -pad ? -name <pad_names>? ] ?-o <output>? ?-sv?
```

where

-vdd | -gnd | -pwr | -pad: plots VDD, GND, PWR, or PAD current waveforms. If lists of <net_names> or <pad_names> are not given for -net or -pad, all net or pad current waveforms are displayed.

By default the waveforms are extracted and rendered in 'xgraph', while the '-sv' option plots it using the Apache 'sv' program.

Filtering Minimum DvD Results

Using the keyword **DVD_GLITCH_FILTER**, conditions for filtering (ignoring) some values of minimum DvD over the timing window (called "minTW"), based on voltage level and glitch width conditions, can be set, as shown in the Figure 5-5 diagram. Minimum DvD values can be filtered globally or using instance-specific conditions. An output report listing the included and filtered DvD values is written to the file *ladsRpt/Dynamic/<designName>.minTW_filtered*. **Note: do not use glitch filtering in scan mode or**

with power cycle selection. For details on filtering minimum DvD values, see the keyword definition in [section "DVD_GLITCH_FILTER", page C-697](#).

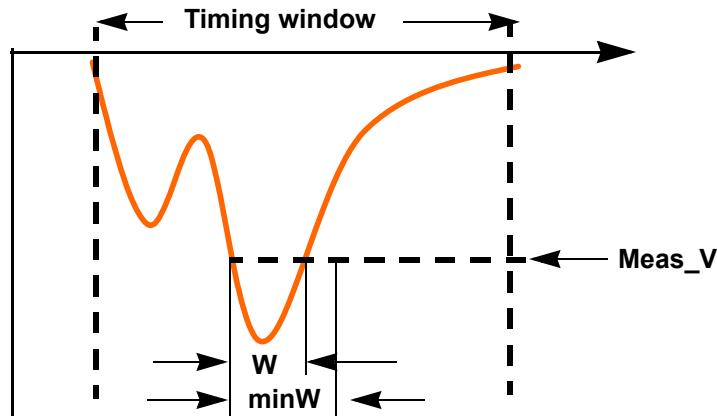


Figure 5-5 Glitch width filtering with DvD_Glitch_Filter

Replaying a Previous RedHawk Session

To use the GUI to start a script-based **RedHawk** run of a previous session, use the menu command **File -> Playback**, which brings up a project directory dialog box that allows you to select the run script from a command log of a previous session. The command log files have filenames of the type: *cmd.log.<date-time>*. You must know the date and time of the session to be able to select the correct session to play back.

TCL Commands to Run Dynamic Voltage Drop Analysis)

The follow table lists TCL commands available for running DvD analysis.

TCL	Purpose
<code><vcd_cycle_time>? ?-a <start_sim_time>?<end_sim_time>? ?-w <logical_path> <physical_path>? ?-d <adsPower_directory>? ?-o <outputFileName>? ?-msg <msgFileName>? ?-cmd <cmdFileName>? ?-tt <presim_time_ps> ?</code>	Sets up for full VCD dynamic DvD analysis when the GSR contains the VCD_FILE section defining the VCD constraints for analysis.
<code>perform analysis [-lowpower -vcd -vectorless]</code>	Runs dynamic DvD analysis.

Chapter 6

Reports

Introduction

Following the selected analysis, a number of method of reviewing the results are available. Many methods of color mapping results are available from the **Tools** and **Results** menus. See the GUI menu descriptions ([section "RedHawk Graphic User Interface Description", page D-821](#)).

RedHawk displays the progress and results of each analysis command to a standard text output. In addition, summary reports are written to the *adsRpt* and *adsPower* directories. The various types of **RedHawk** results, maps and report files available are described in this section.

The most complete set of results and outputs analysis tools can be found in Explorer utility, under the **Explorer** menu.

RedHawk Log Files

Command Files

A command file is created in the *adsRpt/Cmd* directory for each **RedHawk** session, which contains a record of all the top-level commands that were executed, such as design setup, extraction, power calculation and static IR drop analysis. The date and time-stamp extension of the filename represents the time when the **RedHawk** session was invoked:

`redhawk.cmd.<date>-<time-stamp>`

The command file can be used as a playback file to run **RedHawk** in batch mode, if a replay is desired.

Log and Error Files

A complete **RedHawk** log file is created in the *adsRpt/Log* directory for each **RedHawk** session, using the date and time when the session was invoked as the extension of the filename, such as:

`redhawk.log.<date>-<start-time>`

The log file captures the standard output from a complete **RedHawk** run, including all Error and Warning messages. Session information is also displayed in the log message window, as well as written to the log file.

The Error and Warning messages are also written into individual files in separate subdirectories *adsRpt/Error* and *adsRpt/Warn*, as follows:

`Warn/redhawk.warn.<date>-<time-stamp>`

`Error/redhawk.err.<date>-<time-stamp>`

You can chose a different directory and/or filename to redirect the **RedHawk** log and error/ warning files into by using the '-log' option, as follows:

`redhawk -log <user_log_dir/filename>`

This creates a log file in the selected directory and filename, and also Warning and Error files in the same directory with similar names (no time stamps):

`Log: <user_log_dir/filename>`

Warning: <user_log_dir/filename>.warn
 Error: <user_log_dir/filename>.err

Links to Latest Log and Message Files

Under the *adsRpt* directory, **RedHawk** (and also other tools such as GDS2DEF/GDS2RH and APL) creates top level Log, Command, Warning and Error files that are soft links to the latest version of the file, as follows:

Log: *Log/redhawk.log.<date>-<time-stamp>* (link to latest file)
 Command: *Cmd/redhawk.cmd.<date>-<time-stamp>* (link to latest file)
 Error: *Error/redhawk.err.<date>-<time-stamp>* (link to latest file)
 Warning: *Warn/redhawk.warn.<date>-<time-stamp>* (link to latest file)

APL, *gds2def/gds2rh*, and other **RedHawk** tools follow the same protocol.

GUI Log Message Viewer

From the GUI, you can quickly find specific Info, Warning and Error reports by going to the **Results** menu and selecting the **Log Message Viewer**, as shown in Figure 6-6.

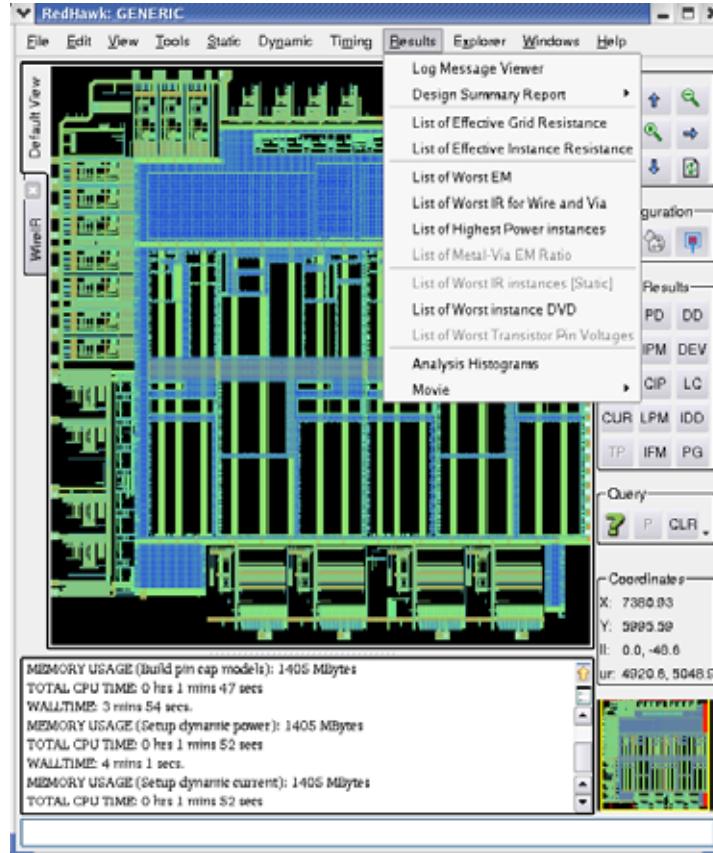


Figure 6-6 Opening the Log Message Viewer

In the upper window by default you will get an **Error/Warnings Summary** of the Info, Warning, and Error log messages generated during the current session, as shown in Figure 6-7.

By clicking on one of the Log Message categories you can get a list of messages in that category, as shown in Figure 6-8. If you then click on an individual message the message is highlighted in the session log in the lower view window. By right clicking on the message in the upper view window you can get additional information on that particular Error or Warning.

By clicking on **CPU/Memory Usage** you get a summary of the session CPU and Memory usage by stage up to the present time.

The **Setup Design** button displays all input files processed during the setup step, color coded to indicate if there were any problems with the file, as follows:

- green - no problems were encountered in processing the file
- orange - there was one or more Warning messages generated in processing the file
- red - there was one or more Error messages generated in processing the file



Figure 6-7 List of Messages in Log Viewer

The **Power** button displays the Power Summary report of results from the Power Calculation step.

At the bottom of the 'Log Message Viewer' the name and path of the log file displayed is shown in the 'Log File' window. The **Browse** and **Apply** buttons allow you to select and display log files from other RedHawk sessions. If a session has continued since you displayed the log file, you can update the log message display by using the **Refresh** button.

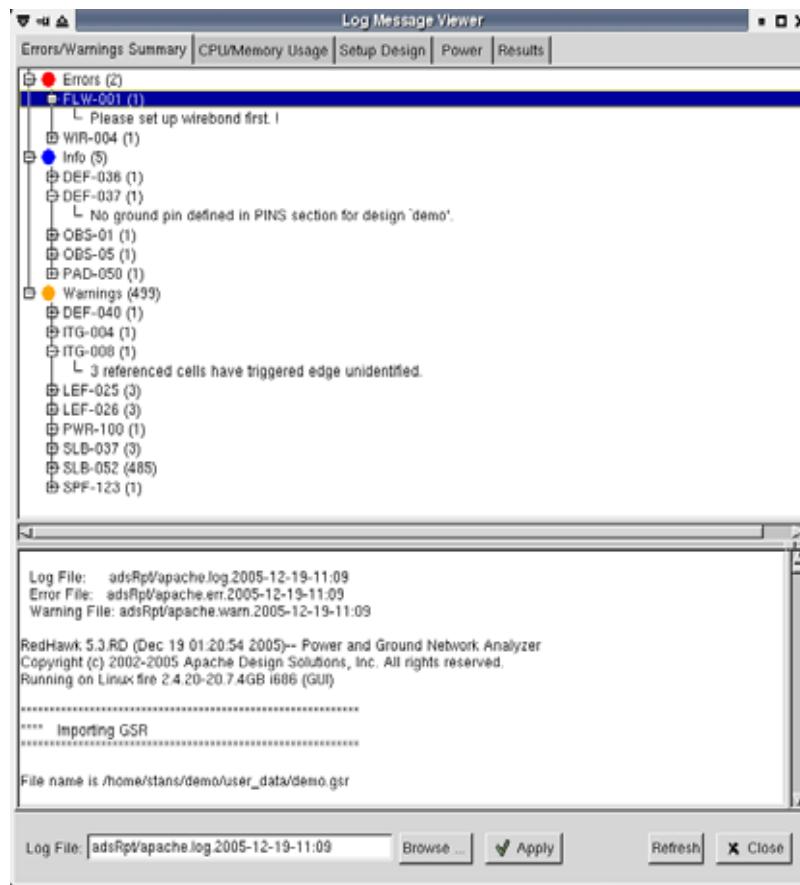


Figure 6-8 Reviewing the List of Error Messages

Reviewing Shorts in the Design

RedHawk can detect and report shorts between any two nets, including those in the same domain (for example, if both are power nets), or in different domains (one power, one ground). Shorts can occur between wires, a wire and a via, a wire and a pin, a via and a pin, and so on. Shorts can be displayed in the GUI using the menu command **View -> Connectivity -> Shorts**.

This brings up a list of shorts in the design, which you can use to zoom to each location by clicking on the item in the list.

RedHawk also lists shorts in the file *adsRpt/Error/redhawk.err<timestamp>* file, with message IDs CON-109, CON-110 and CON-111. There are several reporting cases, depending on whether both shorted items are extracted or not and the setting of the IGNORE_SHORT keyword in the GSR.

1. If one shorted net is to be extracted (that is, the net is in the VDD_NET list or GND_NET list in the GSR) and the other is not (that is, the net is in neither the VDD_NET list nor the GND_NET list), then RedHawk reports the short in a CON-111 message, regardless of the IGNORE_SHORT keyword setting:

WARNING (CON-111): A short between net <A> and is detected on layer <M1> at (x1,y1 x2,y2) by wires. But since net is not extracted, net<A>'s R network is not affected.

2. If both shorted nets are extracted (that is, they are listed in either the VDD_NET list or GND_NET list in the GSR), then depending on the IGNORE_SHORT setting in the GSR, RedHawk reports either a CON-109 or a CON-110 message:
 - a. If IGNORE_SHORT 1 (default), RedHawk does NOT short the R network of the two nets and reports a CON-110 message:
WARNING (CON-110): A short between net <A> and is detected on layer<M1> at (x1,y1 x2,y2) by wires. The short is ignored and their R network will not be shorted!
 - b. If IGNORE_SHORT 0, RedHawk shorts the R network of the two nets and reports a CON-109 message:
ERROR (CON-109): A short between net<A> and is detected on layer<M1> at (x1,y1 x2,y2) by wires, their R network will be shorted!

Tech File Viewer

Through the GUI interface the Tech File Viewer can present different views of the Tech File. Use **View>Technology Layers** to see more information about the layer parameters in the Tech file, as shown in Figure 6-9. The Technology Viewer has a **Zoom** button to make information larger, and also a **GIF** button for obtaining a GIF format output of the Technology Viewer.

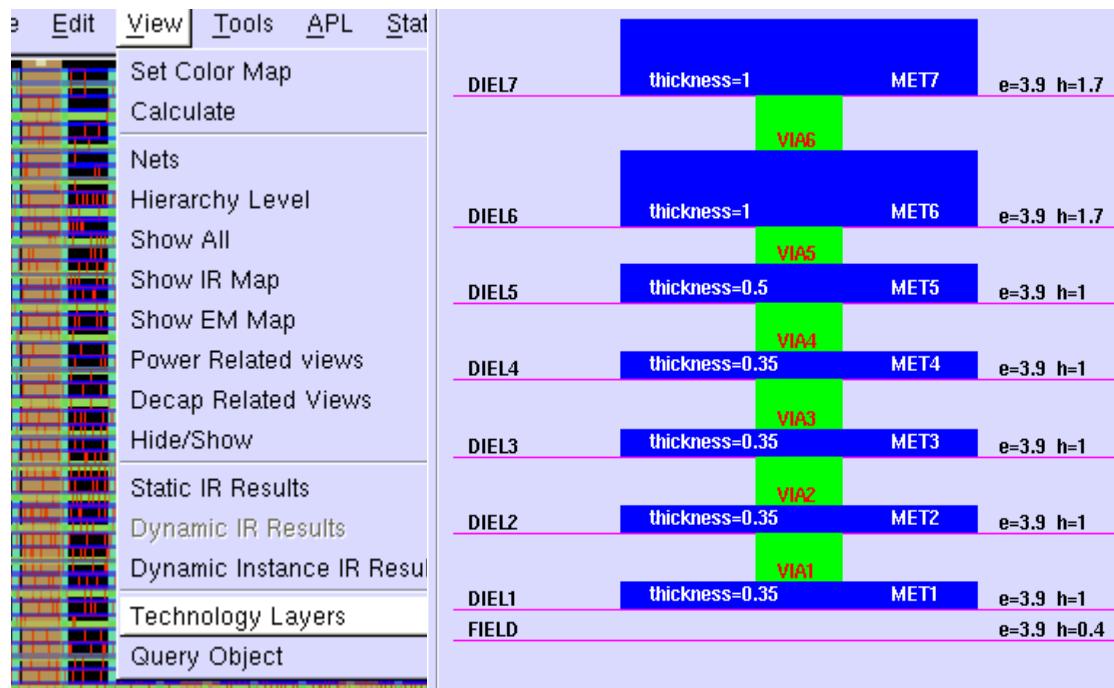


Figure 6-9 Tech File Viewer use

You can also view up to three via models between two metal layers in the technology viewer, activated using the 'Substrate button' in the technology viewer dialog.

DEF Import Summary

A file *adsRpt/tech_summary.rpt* is created after importing DEF, which reports which layers are used and which layers are ignored in the design.

Summary Files for Power

power_summary.rpt File

The **Power > Calculate Power** command from the GUI, or the perform `pwrCalc` from the TCL shell automatically generates a summary results file for power called `power_summary.rpt` in the `adsRpt` directory. This file contains a detailed breakdown of power for each block in the chip. An example of a `power_summary.rpt` is shown below:

Recommended dynamic simulation time, 5000psec, to include 100% of total power for DYNAMIC_SIMULATION_TIME in GSR.

```
INFO: Importing user specified power file: demo.inst.power
INFO: Instances not specified in this file will have zero power.
INFO: 100% of instances specified in instance_power_file have corresponding instances in the design.
0 out of 4142 instances do not have corresponding instance in the design.
INFO: For complete list of PWR-121 and PWR-122 WARNINGS, please see adsRpt/apache.inst_pwr_file.mismatch.
```

The power is based on the INSTANCE_POWER FILE: `demo.inst.power`. Redhawk honors instance-by-instance power as specified in the above file. Individual components of power, like switching power, internal power, however, are calculated by Redhawk and may have been scaled to meet the total instance-specific power number.

Power of different frequency (MHz) domain in Watts:

Frequency	total_pwr	leakage_pwr	internal_pwr	switching_pwr	%_total_pwr
4.000e+02	1.729e-02	1.704e-04	1.107e-02	6.052e-03	8.333e+01%
2.000e+02	3.4595e-03	2.1869e-05	3.1000e-03	3.3758e-04	1.666e+01%
0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00%

Power of different Vdd domain in Watts:

Vdd_domain	total_pwr	leakage_pwr	internal_pwr	switching_pwr	%_total_pwr
VDD (1.1V)	2.075e-02	1.922e-04	1.417e-02	6.389e-03	1.000e+02%

Power of different cell types in Watts:

cell_type	total_pwr	leakage_pwr	internal_pwr	switching_pwr	%_total_pwr
combinational	7.111e-03	1.221e-04	1.754e-03	5.235e-03	3.426e+01
latch_and_FF	4.292e-03	3.890e-05	3.323e-03	9.301e-04	2.067e+01
memory	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
I/O	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
clocked_inst	9.352e-03	3.119e-05	9.096e-03	2.246e-04	4.505e+01
decap	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00

where `clocked_inst` are instances that cannot be classified as `latch_and_FF`, `memory`, or `I/O`, but have clock pin(s).

Total chip power, 0.020757 Watt including core power and other domain power.

Total clock network only power, 0.003225 Watt. Total clock power, including clock network and FF/latch clock pin power, 0.005992 Watt.

<top_level_block>.power.rpt File

A file called `<top_level_block>.power.rpt` is created in the `adsRpt` directory by power calculation. This file contains the following power information for each instance (all power is specified in Watts, and current is specified in Amps). For multiple-Vdd/Vss designs, each power pin has a separate set of data (pin name at end):

```
<inst_name> <cell_name> <freq_Hz> <toggle_rate> <leakage_power>
<switching_power> <internal_power + clock_pin_power> <total_power>
<X_loc_um> <Y_loc_m> <VDD/VSS_domain>
<[leakage_data_source]_[internal_pwr_data_source]>
<domain_type: [ 1 | 2 | 0 ]> <leakage_current> <total_current>
<LIB_name> <cell_p/g_pin_name>
```

where

- <inst_name>: name of instance
- <cell_name>: cell name
- <freq_Hz>: operating frequency
- <toggle_rate> : average number of value changes per clock cycle
- <leakage_power>: leakage power
- <switching_power>: switching power
- <internal_power + clock_pin_power> : internal power including clock pin power
- <total_power>: total power consumption of pin
- <X_loc_um> <Y_loc_m> : coordinates of block
- <VDD/VSS_domain> : name of power domain to which pin is connected
- <[leakage_data_source]_[internal_pwr_data_source]>: each specified as 'apl' or 'lib'
- <domain_type>: 1 = non-multi_rail_pwr_domain; 2 = multi-rail_Vdd_domain; 0 = multi-rail_VSS_domain
- <leakage_current> : leakage current
- <total_current> : total current, A
- <LIB_name> : cell .lib name (if cell is not defined in a .lib, "lef2lib" is shown)
- <cell_pg_pin_name>: P/G pin name

apache.power.info File

A file called `apache.power.info` is created in `adsRpt` directory containing additional information, such as missing nets in VCD and STA files to help guide the users in debugging their power calculation results.

Results for Static and Dynamic Voltage and Current Analyses

EM, Static IR and DvD Colormap Displays

Most of the results of RedHawk analyses can be displayed as full chip color maps in the GUI, by clicking on the appropriate menu command or button (see [section "RedHawk Graphic User Interface Description", page D-821](#)), or using the TCL 'show' command (see [section "TCL / Script Commands", page D-756](#)).

Static EM and IR Drop Results Files

The following static summary files can be found in the *adsRpt/Static* directory.

Note: net names listed are actual domain names.

The *<design>.em.worst* file specifies the worst EM violations for wire pieces and vias in decreasing order.

The report contains the following information on each line.

For Wires:

```
<metal_layer> <location> <EM_ratio> <domain_name> <metal_width>
```

For Vias:

```
<via_name> <location> <EM_ratio> <VDD | VSS>
```

The *<design>.ir.worst* file specifies the nodes with the worst static IR drop (voltage drop and ground bounce) in decreasing order. The report contains the following information on each line:

```
<actual_voltage> <ideal_voltage> <domain_name> <x_y_location> <layer_name>.
```

A sample file follows:

```
#voltage #volt #net #x_y_location #layer_name
1.0734 1.0800 VDD2 ( 261.700, 155.760)MET2
1.0734 1.0800 VDD2 ( 262.780, 155.760)MET2
1.0736 1.0800 VDD2 ( 261.700, 232.980)MET2
...

```

The *<design>.inst.worst* file specifies the instances with the worst static IR drop (voltage drop and ground bounce) in decreasing order. The report contains the following information on each line:

```
<inst_vdd> <vdd_drop> <gnd_bounce> <ideal_vdd> <domain_name> <x_y_loc>
<inst_name>
```

An example file follows:

```
#inst_vdd #inst_drop #gnd_bounce #ideal_vdd #pwr_net #x_y_location
#inst_name
1.0787 0.0006 0.0006 1.0800 VDD1( 60.7200, 273.7350) inst1
1.0787 0.0006 0.0006 1.0800 VDD1( 71.6100, 273.7350) inst2
1.0786 0.0008 0.0006 1.0800 VDD1( 28.3800, 265.1550) inst3
...

```

The *switch_static.rpt* report is for instances of header/footer switches used in low power design application only. This file specifies the voltage and current information for the header/footer switches. The report contains the following information on each line:

```
<internal_node_voltage> <voltage_on_switch> <average_current>
<header | footer> <instance_name>
```

An example file follows:

```
#int_node_volt #volt_on_switch #avg_current #type #instance_name
1.0800 0.0037 0.6248 header QNP_1_CORNER_HS_CELL_002
1.0800 0.0033 0.5533 header QNP_1_CORNER_HS_CELL_001
1.0796 0.0049 0.8231 header QNP_1_BOTTOM_HS_CELL_115
...

```

Static and Dynamic Voltage Drop Results Files

The following dynamic summary files can be found in the *adsRpt/Dynamic/* directory. Note that intermediate graph results of *ivdd* and *ipwr* plots can be viewed during long runs using the 'xgraph <filename>' command.

Note: net names listed are actual domain names.

<design>.ir.worst File

The *<design>.ir.worst* file specifies the location of the worst dynamic voltage drop (voltage drop and ground bounce) in decreasing order. The report contains the following information on each line:

<actual_voltage><ideal_voltage><domain_name><x_y_location><layer_name>.

An example file follows.

```
#voltage #ideal_volt #net #x_y_location #layer_name
0.9262    1.0800    VDD1 (261.700, 155.760) MET2
0.9275    1.0800    VDD1 (262.780, 155.760) MET2
0.9504    1.0800    VDD1 (174.500, 155.760) MET2
...
...
```

<design>.dvd File

The *<design>.dvd* file reports voltage drop in terms of VDD-VSS differential for each instance, in decreasing order. The report contains the following information on each line:

```
<x_loc> <y_loc> <effective_vdd-vss_over_TW> <max_vdd-vss_over_TW>
<min_vdd-vss_over_TW> <min_vdd-vss_of_clockcycle>
<min_vdd_voltage_over_TW> <max_gnd_bounce_over_TW> <VDD | VSS>
<instance_name>
```

To obtain a valid effective Vdd, the values are averaged over several small sample windows, not over the entire timing window defined in STA file. By sliding a small window through the entire timing window of the instance, several average effective Vdd numbers are obtained. The worst of the averages computed is then reported as the effective (Vdd-Vss) voltage value. If no timing window data are available, the DvD values based on timing window are not reported.

The <TW_flag> parameter indicates the Timing Window coverage in the dynamic simulation time. If it is an empty string, the TW is covered in the dynamic simulation time. If it is "^", the TW is partially covered in the dynamic simulation time. If it is "*", the TW is not covered in the dynamic simulation time.

A sample file follows.

```
#loc_x loc_y eff_vdd max_pg_tw min_pg_tw min_pg_sim min_vdd_tw max_vss_tw
domain instance_name
218.122 167.320 0.9671 0.9794 0.9555 0.8798 1.0012 0.0458 VDD inst241
267.134 157.905 0.9808 0.9904 0.9736 0.9181 1.0122 0.0386 VDD inst3469
264.996 153.615 1.0034 1.0079 0.9992 0.9635 1.0069 0.0085 VDD inst2463
...
...
```

Static and Dynamic Results for Vias

RedHawk can generate a Via Voltage Drop and Current report for both static and dynamic analysis. The report is generated in the *adsRpt* folder and contains voltage drop and current values for vias. This option can be turned On using the GSR keyword 'VIA_IR_REPORT 1'. Alternatively, you can use the TCL command

```
report [ ir | dvd ] -via -o <output_file>
```

to generate this report. The output filenames are:

- Static: *adsRpt/Static/<>.via.ir.worst*
- Dynamic: *adsRpt/Dynamic/<>.via.ir.worst*

A sample output report is shown in Figure 6-10

```
#Report via location (x,y) with worst voltage drops on top and bot layers, as well as current through via
#bottome_voltage(V) #top_voltage(V) #drop_across_via(V) #net #current(A) #current_dir(UP/DN) #via_layer #via_name #coordinates
1.0021 0.9892 0.0129 vddlw 0.000544034 UP VIA2 VIA23_CFCORE_SUBCHIP ( 66.150, 78.675)
1.0015 0.9892 0.0124 vddlw 0.000519468 UP VIA2 VIA23_CFCORE_SUBCHIP ( 66.150, 78.525)
1.0000 0.9892 0.0109 vddlw 0.000457096 UP VIA2 VIA23_CFCORE_SUBCHIP ( 46.550, 78.675)
0.9997 0.9892 0.0106 vddlw 0.00044365 UP VIA2 VIA23_CFCORE_SUBCHIP ( 46.550, 78.525)
0.9995 0.9890 0.0105 vddlw 0.000441033 UP VIA2 VIA23_CFCORE_SUBCHIP ( 60.550, 78.675)
```

Figure 6-10 Via voltage drop and current report

Current Report Files

<design>.ignd File

The *<design>.ignd* file reports the summation of all the instances' Vss current over the simulation time. The report contains the following information on each line:
<simulation_time> <current_value>. A sample file follows.

Time	i (gnd)
0.000000	0.0
-3480	-0.0038748
-3470	-0.00309187
-3460	-0.00320901
...	

<design>.ipwr File

The *<design>.ipwr* file reports the summation of all the instances' Vdd current demand over the simulation time. The report contains the following information on each line:
<simulation_time> <current_value>. A sample file follows.

Time	i (pwr)
0.000000	0.0
-3480	0.0038748
-3470	0.00309187
...	

<design>.ivdd File

The *<design>.ivdd* file reports the chip power supply current over the simulation time. This power supply current is supplied by the voltage sources for each power/ground source. Without a package, ivdd measures the current at the pads of the design, and therefore includes the effect of all on-chip parasitics, decaps, and other capacitative effects. With a package, ivdd measures the current at the voltage sources external to the package, and therefore includes the effect of package parasitics, in addition to on-chip parasitics, and other capacitative effects. The report contains the following information on each line: *<simulation_time> <current_value>*. A sample file follows.

Time	i (vdd)
0.000000	0.0
-3490.0000	0.00265285

```

-3480.0000      0.0036665
-3470.0000      0.00341904
...

```

<design>.ivdd.vsrc File

Following dynamic analysis the *<design_name>.ivdd.vsrc* file is generated in the *adsRpt/Dynamic* directory, which contains the supply current waveforms for all power sources. See example in Figure 6-11.

The TCL command that plots the total supply current from all pads that belong to a specified net is:

```
plot current -net -name <net_name> -pad -sv
```

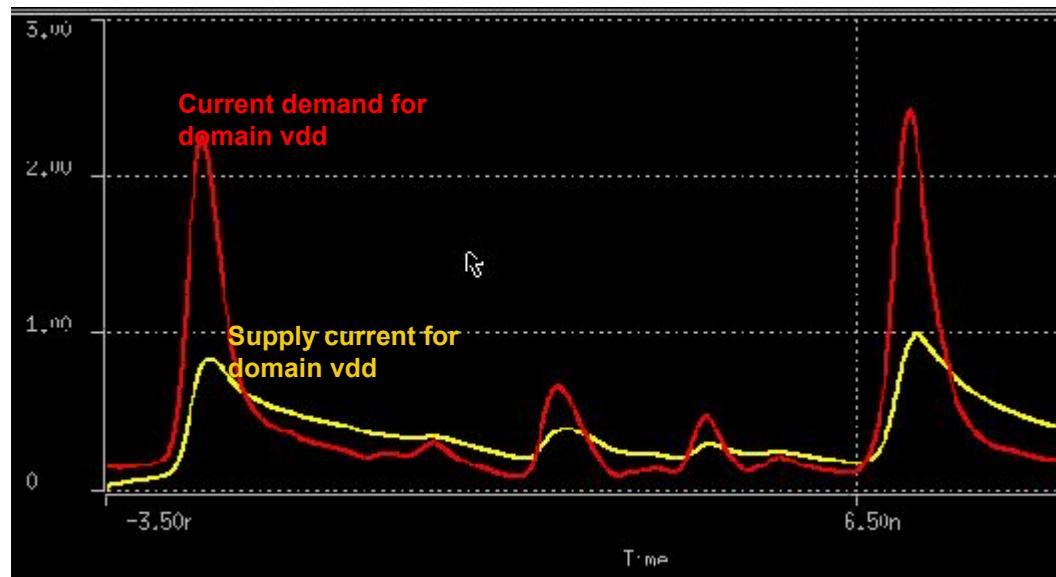


Figure 6-11 current demand and supply current waveforms

<design>.ipwr.domain File

The *<design_name>.ipwr.domain* file is generated after dynamic analysis, containing current demand waveforms for each power domain defined in the GSR keyword VDD_NETS. See example in Figure 6-11.

The TCL command that plots the total current demand from a specified net is:

```
plot current -net -name <net_name> -sv
```

<design>.ignd.domain File

The *<design_name>.ignd.domain* file is generated after dynamic analysis containing ground current waveforms for each power domain defined in the GSR keyword VSS_NETS.

switch_dynamic.rpt File

The *switch_dynamic.rpt* report contains analysis results for instances of header/footer switches used in low power design applications only.

For vectorless switch analysis, this file specifies the worst VDD-VSS differential, and the maximum voltage and current over simulation time for the header/footer switch instances.

The report contains the following information on each line: instance name, switch type, worst Vdd-Vss voltage, and maximum switch voltage and current. A sample file follows.

```
#<inst_name> <type> <worst_int_Vdd-Vss_Volt> <max_Vsw_Volt> <max_Isw_Amp>
    QNP_1_BOTTOM_HS_CELL_083 H 1.051716 0.024256 0.004053
    QNP_1_BOTTOM_HS_CELL_082 H 1.051716 0.025019 0.004180
    QNP_1_BOTTOM_HS_CELL_081 H 1.051716 0.026545 0.004434
    ...

```

Ramp-up analysis reports include the max switch current and the status of each header or footer switch at the end of the simulation period (OFF or ON). The file syntax and an example follows:

```
#instance_name\ttype maximal_Isw(Amp) status (SAT)
#SAT: maximal_Isw > saturation_current as specified in switch model file
switch_inst_R17_C52 header 5.000000e-09 OFF SAT
switch_inst_R17_C52 header 2.603736e-09 OFF
```

This allows you to quickly see which switch turned ON and its efficiency.

decaps.rpt File

Depending on the value of the DYNAMIC_REPORT_DECAP keyword in the GSR, the *decaps.rpt* file reports: for keyword value = 1, the maximum current reported for all intentional decaps as defined in the GSR; for value = 2, in addition to decaps reported in mode 1, reports also the maximum current associated with non-switching instances. Or if the keyword has value = 0 (default), off (there is no decap report).

freqd_ipwr.out File

This file is generated for the baseline DvD flow; for all other modes it is not generated. Each instance in a design can be associated with a particular clock and hence a frequency domain. The *freqd_ipwr.out* file contains a waveform representing the sum of all switching currents for the instances associated with each frequency domain. The current waveforms may be displayed using the *xgraph* command as follows:

```
xgraph freqd_ipwr.out
```

The file has two columns to define the total current waveform for each clock frequency: time and current in Amps.

Pad Current File

The *pad.current* file specifies the current from the pad cells or pad locations, excluding presim time current. This file is very helpful for determining whether the connectivity is complete from the specified pad locations. This report is generated for both static IR/EM and dynamic voltage drop analysis. The report generated from static IR/EM analysis is written to *adsRpt/Static* directory and contains the following information on each line: <current_value> <x_y_location_of_pad_center> <pad_name>. A sample file follows.

```
#current #pad_center_location #pad_name
2.4745 ( 65.282, 0.082) Vdd_130564_1645
2.1274 ( 163.329, 0.082) Vdd_326658_1645
2.4745 ( 263.206, 0.130) Vdd_526412_2605
...

```

The report generated from dynamic voltage drop analysis is written to *adsRpt/Dynamic* directory and contains the pad current over simulation time, in the following format:

```
<pad_name_a>
<simulation_time1 > <current_value >
...

```

```
<pad_name_b>
<simulation_time1 > <current_value >
...

```

A sample file follows:

```
"Vss1
-1750 -1.13646e-05
-1740 -0.00240712
...
-20 -0.0511077
-10 -0.0508567
0 -0.0506185
10 -0.0503398
20 -0.0500513
...
2490 -0.0914897
2500 -0.0909729

"Vdd1
-1750 1.13646e-05
-1740 0.0023354
...
2490 0.0911803
2500 0.0906709
```

probe_nodes.out file

The probe_nodes.out file reports probe node voltage and current report for static and Dynamic analysis in adsRpt/Static/ or adsRpt/Dynamic/ directories respectively. User need to specify probe locations in the file specified with PROBE_NODE_FILE GSR keyword to generate voltage and current for that nodes.

Sample probe_nodes.out file contents:

#Probe_Name	I	V
cell_198_VDD_INT	3.454429e-05	1.091716e+00
cell_198_VSS	2.562570e-05	9.402982e-02
cell_719_VDD_INT	8.298721e-06	1.091396e+00
cell_719_VSS	9.167607e-07	9.248270e-02

CMM Constraint Violation Reports

Constraint Violation Summary

After top-level analysis, if there are any constraints defined in sub-block CMMs, a violation summary table is recorded in the *redhawk.log* file after post-processing of simulation results. An example is shown below:

Summary of CMM Constraint Violation:

Constraint Type No. of violations

Top Connection Min Voltage 50

```
Top Connection Max Voltage 50
Pin Min Voltage 500
Pin Max Voltage 500
There are total 1100 constraint violations in the CMM
instances, please look at adsRpt/Dynamic/
CMM_constraintViolation.rpt for detail!
```

Dynamic Analysis Constraint Summary

For dynamic analysis, the final CMM constraint violation report is recorded in the file *adsRpt/Dynamic/CMM_constraintViolation.rpt*. The file format is:

```
# Top connection max voltage constraint violation:
# <inst> <cell> <x,y,layer> <net_name> <constraint> <voltage_value>
#-----
# Top connection min voltage constraint violation:
# <inst> <cell> <x,y,layer> <net_name> <constraint> <voltage_value>
#-----
# pin max voltage constraint violation:
# <inst:pin> <cell> <x,y> <constraint> <voltage_value>
#-----
# pin min voltage constraint violation:
# <inst:pin> <cell> <x,y> <constraint> <voltage_value>
#-----
```

For example:

```
# Top connection min voltage constraint violation:
# <inst> <cell> <x,y,layer> <net_name> <constraint> <voltage_value>
#-----
ABC_CORE/ram64/adsU1 ram32x8x2 (1759.3600,783.0350, METAL2) VDD 1.7820
1.7309
# Top connection max voltage constraint violation:
# <inst> <cell> <x,y,layer> <net_name> <constraint> <voltage_value>
#-----
ABC_CORE/ram64/adsU1 ram32x8x2 (1763.9600,778.4350, METAL2) VSS 0.0010
0.0528
# Pin min voltage constraint violation:
# <inst:pin> <cell> <x,y> <constraint> <voltage_value>
#-----
ABC_CORE/ram64/adsU1:VDD ram32x8x2 (1792.3700,727.8150) 1.7460 1.7294
# Pin max voltage constraint violation:
# <inst:pin> <cell> <x,y> <constraint> <voltage_value>
#-----
ABC_CORE/ram64/adsU1:VSS ram32x8x2 (1790.1550,727.6450) 0.0100 0.0534
```

Static Analysis Constraint Summary

For Static, the final report is recorded in the file *adsRpt/Static/CMM_constraintViolation.rpt*. The file format is:

```
# Top connection IR constraint violation:
# <inst> <cell> <x,y,layer> <net_name> <constraint> <voltage_value>
#-----
```

```
# pin IR constraint violation:  

# <inst:pin> <cell> <x,y> <constraint> <voltage_value>  

#-----
```

For example:

```
# Top connection IR constraint violation:  

# <inst> <cell> <x,y,layer> <net_name> <constraint> <voltage_value>  

#-----  

ABC_CORE/ram64/adsU1 ram32x8x2 (1759.3600,783.0350, METAL2) VDD 1.7960  

1.7909  

# Pin IR constraint violation:  

# <inst:pin> <cell> <x,y> <constraint> <voltage_value>  

#-----  

ABC_CORE/ram64/adsU1:VDD ram32x8x2 (1792.3700,727.8150) 1.7910  

1.7909
```

Debugging Using Summary Files in the GUI

After generating the text results reports, they can be brought up in the GUI for interactive debugging. The following GUI commands enable interactive debugging of EM violations and static IR/dynamic voltage drop violations. The execution of any of these commands generates a ranked list of violations in a pop-up window. An individual violation can be selected and highlighted on the layout using the following menu selections:

Results > List of worst EM for static simulation
Results > List of worst IR instances for static simulation
Results > List of worst IR for wire and via for static simulation
Results > List of worst DvD instances for dynamic simulation

Output Files from Multiple Vdd/Vss Analysis

Table 6-1 summarizes the files and reports available for multiple Vdd analysis.

Table 6-1 Multiple Vdd Result Files

Name of File	Information
<i>adsRpt/apache.refCell.noPGarcs</i>	Lists cells with missing P/G arc information, along with the names of the P/G pins.
<i>adsRpt/Dynamic/<design>.dvd.arc</i>	Dynamic voltage drop, as in standard <i><design>.dvd</i> file, but on a per arc basis.
<i>adsRpt/Dynamic/<design>.dvd.pin</i>	Dynamic voltage drop, as in standard <i><design>.dvd</i> file, but on a per pin basis.
<i>adsRpt/Static/<design>.inst.pin</i>	Static instance voltage drop data, one line for each P/G pin value for every instance.
<i>adsRpt/Static/<design>.inst.arc</i>	Static instance voltage drop data, one line for each P/G arc value for every instance.

Other Files

RedHawk writes out various files in the *adsRpt* directory that provide additional information in preparation for or after an analysis run.

Miscellaneous

The following files are generated by RedHawk to report nonstandard library or simulation conditions.

File	Description
<i>apache.noLefPins</i>	List of cells with no pins defined in their LEF macro definition.
<i>apache.noLibPins</i>	List of cells with no pins defined in their library definition.
<i>apache.rcNoDriver</i>	List of nets with no drivers.
<i>apache.refCell.noLefLib</i>	List of cells present in design, but not in library or lef files.
<i>apache.refCell.noLib</i>	List of cells present in design, but not in library files.
<i>apache.refCell.noLef</i>	List of cells present in design, but not in lef files.
<i>apache.refCell.noPwr</i>	List of cells present in design without any internal power tables specified in their library definition.
<i>apache.refCell.noTriggerEdge</i>	List of cells present in design with no identifiable clock triggering edges in the library files.
<i>apache.inst.libCurrent</i>	For instances with missing APL data, this file provides current profile information from the <i>.lib</i> files.
<i>apache.switchCellnoTW</i>	List of names of instances that are switch cell type but have no timing window from the STA file for their control pin.
<i>clock.untraced</i>	List of nets that are STA clock nets, but are untraceable by RedHawk tracer.
<i><design>.power.unused</i>	List of instances that are not hooked to the power network.

<code><design>.power.worst</code>	List of instances that have the highest static power consumption.
<code><design_name>.rcFail</code>	Information on those nets for which RedHawk was unable to create a pi-model for their output load.
<code><design_net>.Via.unconnect</code>	List of vias belonging to the net that are physically isolated from a power source.
<code><design_net>.Wire.unconnect</code>	List of wires belonging to the net that are physically isolated from the driving source.
<code><design_net>.Pin.unconnect</code>	List of pins belonging to the net that are physically isolated from a power source.
<code><design_net>.PinInst.unconnect.lo gic</code>	List of PinInsts which are logically disconnected but physically connected and their connected nets.
<code><design>.unplaced</code>	List of instances designated "UNPLACED" in the DEF file.
<code><design>.stackvia</code>	List of missing vias in the working directory or <code>adsRpt</code> directory, to cover complex stacked via scenarios.
<code>GC.unreachable.fflatch</code>	Lists percent of flip flops and latches that are reachable from the gated clock controls, and also lists the unreachable flip-flops and latches.
<code>libParser.errMsgs</code>	List of error messages generated during parsing of the Synopsys Liberty library.
<code>ppi_multi_group_pin_geo_cells</code>	After setup design, list of cells in the design with pins having multiple groups of disconnected geometries and significant IR drops.
<code>undefined_cells</code>	List of all master cell names instantiated in the design with no definition in DEF or LEF.

<i>PG.ploc</i>	<p>List of Power/Ground sources (connections), x,y locations, and layer assignments, which RedHawk identifies during network extraction.</p> <p>If <i>.pad</i> or <i>.pcell</i> files are used for specifying pad instances or cells, when only pad instance/pad cell names are used, a source is automatically identified by the intersection of each P/G pin's geometries and routing wires on the same routing layer.</p> <p>When a specific pin of a pad cell-instance is specified in the <i>.pad/.pcell</i> file, only the intersections of the geometries of the specified pin are examined. If there are multiple connections between a pin's geometries and routing, then the format for the name of each source (connection), in the first column of the <i>PG.ploc</i> file, is</p> <pre><instance_name>:<net_name>: <# of source>.</pre> <p>If a *.ploc file is used, the <i>PG.ploc</i> file should be exactly the same, unless there are disconnected sources in the imported *.ploc file.</p>
<i>DEF.ploc</i>	<p>A reformatted version of the top cell DEF file PINS section, which can be used for P/G source debugging and post-processing.</p> <p>The <i>adsRpt/DEF.ploc</i> file is created after importing DEF if the GSR keyword 'ADD_PLOC_FROM_TOP_DEF' is set to 1 (which means that the design's top cell's pins --i.e., VDD or GND sources-- are read directly from the top cell DEF instead of from user-specified <i>.ploc/.pad/.pcell</i> files).</p> <p>Note that RedHawk exits if a script contains both import pad commands and 'ADD_PLOC_FROM_TOP_DEF 1'.)</p>

Debugging

The following files provide additional information that is useful in debugging RedHawk results.

File	Description
<i>apache.clkPin0</i>	List of all clock pins that are connected to non-clock nets.
<i>apache.dupLIBCells</i>	List of all cells with multiple references in the library files.
<i>apache.rc0Net</i>	List of nets with zero capacitance value.
<i>apache.rcBogus</i>	List of nets and lines in SPEF file that cannot be mapped to a design.
<i>apache.rcMismatch</i>	Reports of inconsistent drivers between SPEF and DEF files.

<i>apache.slew0</i>	List of instances which are missing slew data in timing file. For sequential instances, CLK and OUT pins missing slew are reported and for combinational instances IN and OUT pin missing slew are reported
<i>apache.staBogus</i>	List of incorrect STA related data, such as timing window for instances that are not present in the design.
<i>apache.staFlatten</i>	List of instances that are flattened inside of the RedHawk database and their timing window will be ignored.
<i>apache.tw0</i>	List of instances not covered in the provided STA file.
<i>apache.twclk0</i>	List of sequential instances whose clock pins do not have timing window covered in the STA file.
<i>apache.twclkLate</i>	List of sequential instances whose output pins switch before the clock pin.
<i>inst.reactivated</i>	List of instances not covered in STA file but whose toggle rate is forced to a non-zero state.
<i><design>_<net>.collided_vias</i>	List of vias that touch or collide with other vias.
<i>aplchk.log</i>	List of APL-related errors and warnings.

Dynamic Simulation Preparation

The following report files are created when preparing for dynamic simulation.

File	Description
<i>Dynamic/cell.rpt</i>	Lists of cells that have and do not have APL characterization data.
<i><design>.PG.unconnect</i>	List of instances belonging to analyzed PG network that are physically isolated from VDD and GND. These instances will be ignored during dynamic analysis.
<i><design>.VDD.unconnect</i>	List of instances belonging to analyzed PG network that are physically isolated from VDD. These instances will be ignored during dynamic analysis.
<i><design>.GND.unconnect</i>	List of instances belonging to analyzed PG network that are physically isolated from GND. These instances will be ignored during dynamic analysis.
<i><design>.noTiming</i>	List of instances belonging to analyzed PG network that do not have valid timing windows. These instances will not toggle/switch during dynamic analysis.

Low Power Ramp Up Analysis

The following report files are created from low power ramp-up analysis.

File	Description
<i>virtual_domain_total_i.rpt</i>	Total ramp-up current waveforms for each voltage domain controlled by a header/footer switch in the design. Two columns: time and current in Amps.

virtual_domain_worst_v.rpt:

Worst case node voltage waveforms during ramp-up for each voltage domain controlled by a header/footer switch in the design. Two columns: time and voltage in Volts.

Chapter 7

Fixing and Optimizing Grid and Power Performance

Introduction

After RedHawk static (IR) or dynamic voltage drop (DvD) analyses have identified areas in the design that have unacceptable voltage drops, RedHawk Fixing and Optimization (FAO) functions allow you to modify the power grid and/or allocate new decoupling capacitance to meet specific static and dynamic voltage drop targets, and also power-related circuit timing problems. Fixing voltage drop problems also can reduce noise from ground bounce. Operations on instances with multiple Vdd power sources is handled transparently.

Once a trial power grid and initial placement information is available, RedHawk FAO can be used to globally modify power grid parameters to meet specific voltage drop targets.

RedHawk enables you to easily add or delete power/ground pads, straps, and vias at any stage of a run to explore their impact on voltage drop and electro-migration (EM).

RedHawk is capable of incrementally taking in new changes during IR drop and EM analysis. After detailed placement, and prior to detailed routing, RedHawk can be used to identify high dynamic voltage drop areas, or areas with significant timing impacts from DvD, and fix them through a combination of power grid wire changes and targeted decap placement.

The first part of this chapter describes the manual grid and decap modification commands available in RedHawk, as follows:

- Changing existing layer or via resistivity
- Adding a power/ground pad
- Deleting a power/ground pad
- Adding a via
- Deleting a via
- Adding a power strap
- Editing/deleting a power strap
- Adding a decap cell
- Adding metal layers and via or via arrays
- Writing an ECO file
- Reading an ECO file

As the design moves through final placement and routing, you can use RedHawk Fixing and Optimization functions to automatically modify the grid parameters and decap placement to meet specified goals for both voltage drop and also power-related circuit timing. When all of your voltage drop constraints are met, RedHawk can export an ECO file with the needed design changes and then perform a final Spice-accurate power analysis signoff of the design.

You can perform Undo or Redo operations after making FAO changes, allowing you to try out FAO changes and then reverse them if the results are not satisfactory. Any number of Undo or Redo commands can be performed consecutively until the commands in the stack are exhausted.

The fixing and optimization functions are described in the following sections.

Manual Power Grid Modification

Changing a Metal Layer or Via Resistivity

An existing metal layer or via resistivity can be changed in the **RedHawk** technology (.tech) file and then **RedHawk** can be rerun to see the effects of the changes. Please refer to [Appendix C, "File Definitions"](#), for the format of the Apache .tech file.

Adding a Single Power/Ground Pad

A new power or ground pad can be created using the **Edit > Add Pad** command, as shown in Figure 7-1. A pop-up menu allows the selection of the metal layer to which the pad will be connected. Pick the location for the new pad on the metal layer by clicking at the desired location. The new pad is added to the design in the selected location.

New power pads are displayed as orange rectangular boxes and new ground pads are displayed as white boxes. Use **File > Export ECO** to export the added pad locations and use **File > Import ECO** to import the ECO file to back annotate for future runs. See the section ["Saving Design Changes with the ECO Command", page 7-152](#), for more information on the **ECO** command.

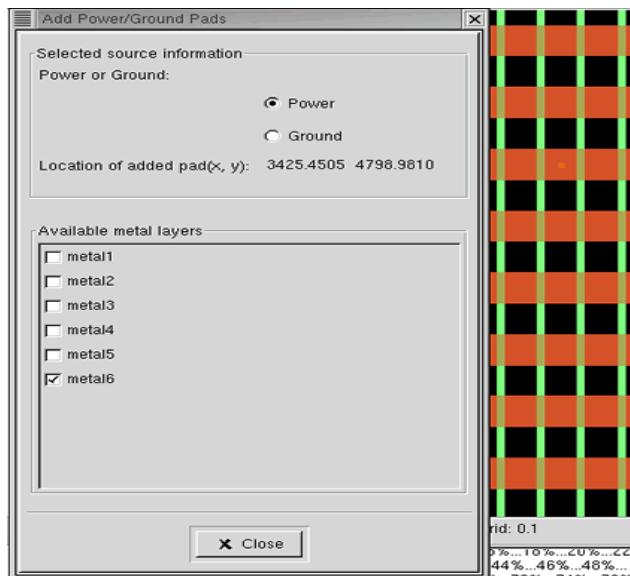


Figure 7-1 **Edit > Add Pad Menu**

Adding a Set of Power/Ground Pads over a Specified Area

The command 'fao add pad' adds a set of power /ground pads over an entire mesh, or in a defined window. You must specify a window, a metal layer, a net, and also a pitch in both x and y directions by which to distribute pads evenly over the specified region. **Undo/Redo** commands can be used. The command line syntax is:

```
fao add pad -window {llx lly urx ury} -metal <layer>
          -net <net_name> -pitchX <microns> -pitchY <microns>
          -r <resistance> -l <inductance> -c <capacitance>
```

where

- window { }: defines lower left and upper right x,y coordinates of the rectangular area in which pads are added (by default, the entire mesh).
- metal: defines the routing layer (required). Only one metal layer is allowed, since every metal has its own placement.
- net: defines the net name (required)
- pitchX -pitchY: defines the incremental distance between pads, starting at lower left corner of the rectangle specified by '-window { }' (required) .
For example, for “-window {a b c d}”, starting at {a b}, RedHawk adds pads at {a b}, {a+pitchX b}, {a+2*pitchX b} ... , {a+pitchY}, {a+pitchX b+pitchY}, {a+2*pitchX b+pitchY} ... , over the entire window.
- r -l -c: specifies pad RLC parameters (units the same as in the PLOC file)

Deleting a Power/Ground Pad

An existing power/ground pad can be selected and deleted using the command **Edit > Delete Pad**. Use the pop-up menu to choose a pad for deletion.

Adding a Via

New vias can be added using the command **Edit->Add Via**. Click on the design in the desired location. Use the pop-up form to define the top and bottom metal layers for the inserted via stack. When adding stacked vias, make sure that the appropriate number of vias and via layers are selected. The new via or stacked vias are added to the design and displayed as a blue box, after saving the changes.

Deleting a Via

An existing via can be selected and deleted using the command **Edit > Delete Via**. You may have to make the vias visible by turning them on with the **View Layers** Configuration button, or the command **View -> Map Configuration -> Layers Color Map**, and then select **Fill** or **Outline** display for the set of vias you want to look at. First select the via for deletion by clicking on it with the left mouse button. The selected via is highlighted in yellow. Use the pop-up menu to choose an existing via for deletion.

Adding One or Multiple Power Straps

1. To add power straps graphically (recommended), select **Edit -> Add Power Strap**. A ‘Power Strap’ dialog box is displayed.
2. Deselect **Add strap by text input**.
3. Display the layers that are important in relation to the new strap.
4. Use the right mouse button to draw a desired strap on the layout. If you want to add several identical straps, draw the first strap representing one edge of the set you want to add, and select **Add multiple straps**.
5. Calculate and enter in the dialog box the location of the outside edge of the last strap to be added and also the desired pitch. RedHawk will add the maximum number of legal straps meeting the constraints specified.
6. Select the layer on which to add the new straps (metal1 is the default).

7. Click on **Commit adding** when the specification and drawn straps are correct (there is no undo function). If there is a physical error or short, an error message will be displayed. A successful strap addition is illustrated in Figure 7-3.
8. If you make a mistake or want to change the added straps, use the **Edit->Power Strap** function to select and then delete or change a strap.

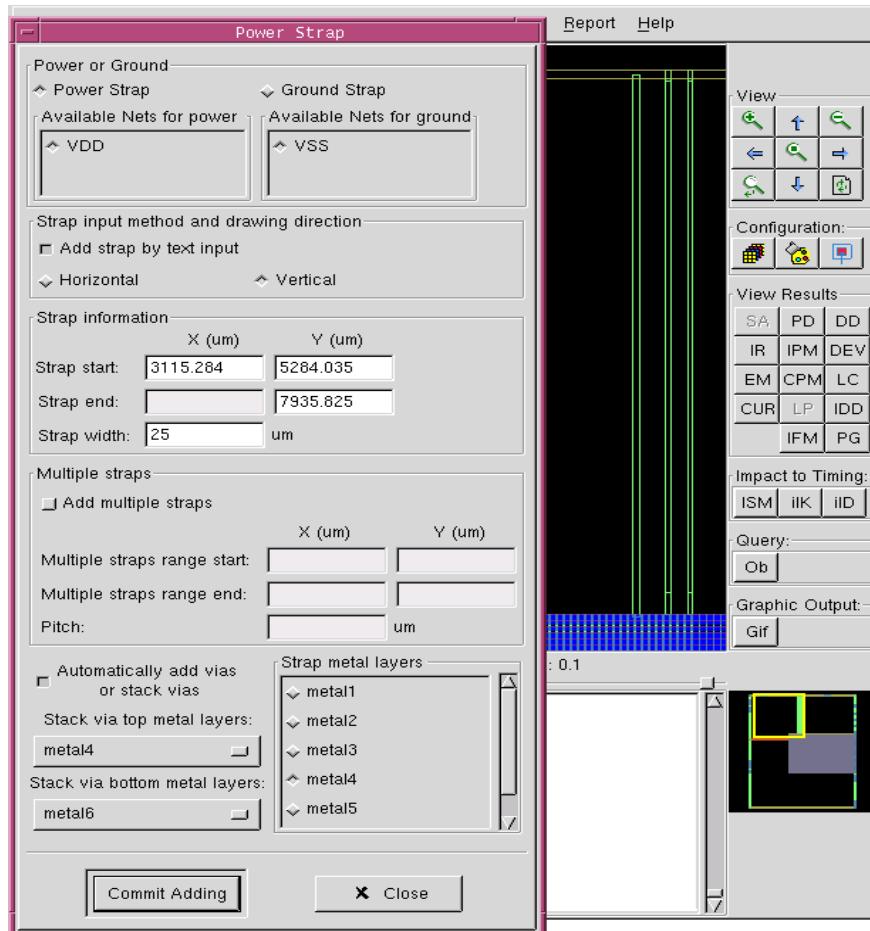


Figure 7-2 Form for adding a strap by text input

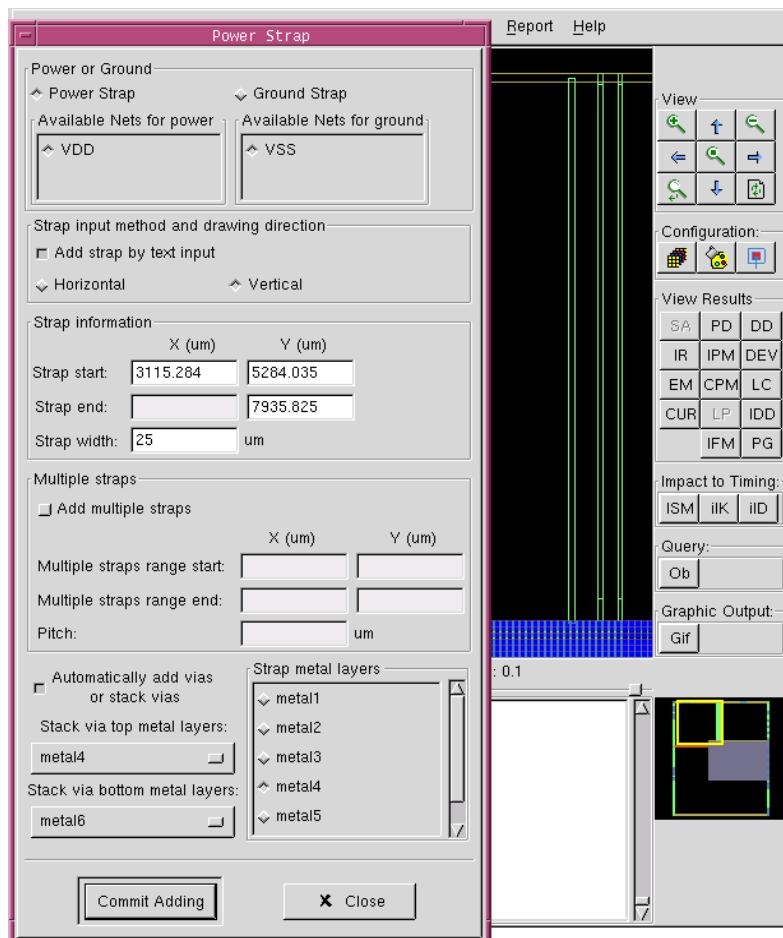


Figure 7-3 Adding multiple straps with Add multiple straps

For an example of adding straps, please refer to the [section "Example Procedure to Fix IR Drop Problems", page 4-74](#).

GUI option for 'eco add strap'

To automatically add stack vias for two or more layers above or below the wiring on the GUI, select the checkbox 'Include stackvias'.

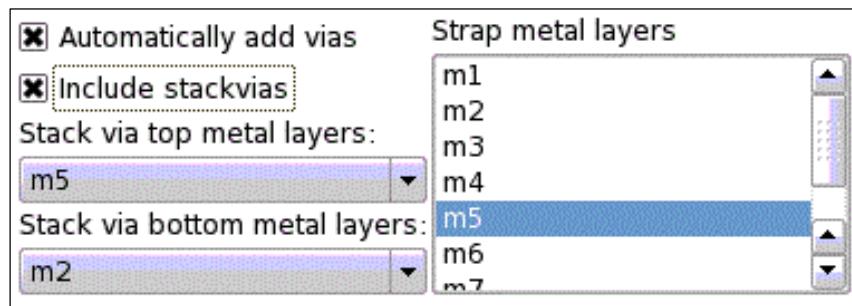


Figure 7-4 GUI option for ‘eco add strap’

The behavior is:

- When the checkbox ‘Include stackvias’ is selected: eco add strap adds vias for all possible metal layers. For example, if you are adding a wire A on M2, it will search if there are wires on all metals layers (except for M2) that overlaps wire A, and add vias if possible.
- When ‘Include stackvias’ checkbox is not selected: eco add strap adds vias for adjacent metal layers only. For example, if you are adding a wire A on M2, it will only search if there are wires on M1/M3 that overlaps wire A, and add vias if possible.

Editing/Deleting a Power Strap

An existing power strap can be selected and edited by using the command **Edit > Edit Power Strap**. First, select the power strap by clicking on it with the left mouse button. The selected strap will be highlighted in yellow. Use the pop-up menu to change the length and width, as well as the x and y start locations of the selected power strap, as illustrated in Figure 7-5. The strap can also be deleted from the design database.

Note: New vias will NOT be automatically added as a result of editing an existing power strap.

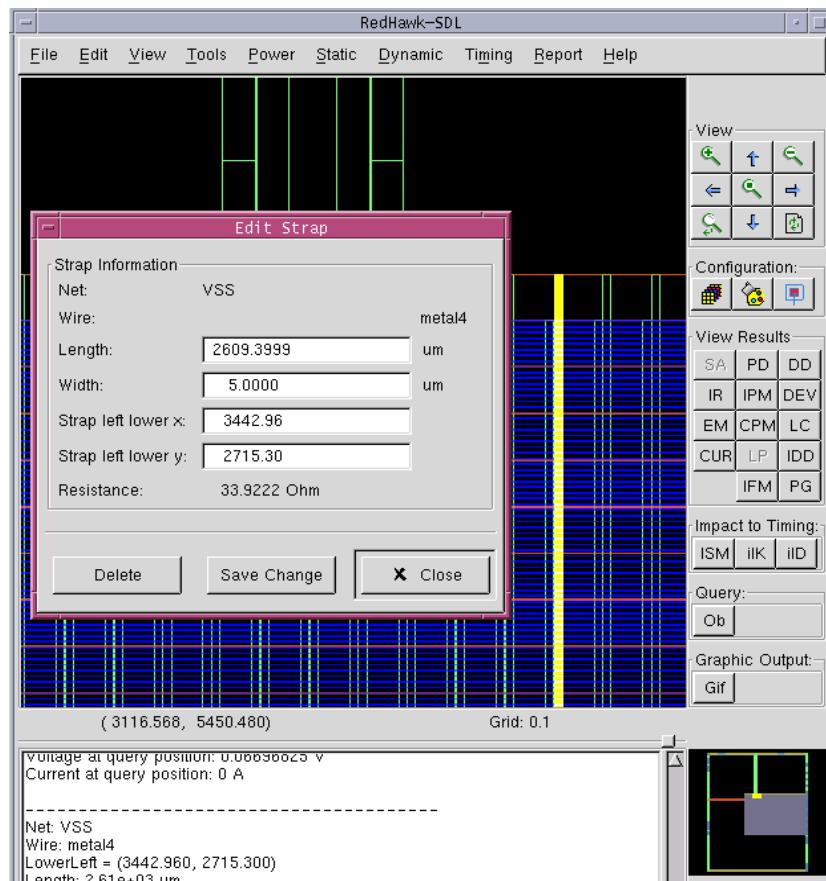


Figure 7-5 Edit->Edit Power Strap

Adding a Decap Cell

One or more intentional decap cells can be added by using the command **Edit > Add Decap Cell**. Click on the **Select Vdd** button on the Add Decap Cell form and then click on the VDD wire location on the lowest metal layer where the decap should be inserted. Next, click on the **Select Gnd** button on the form and click on a VSS location very close to the previously selected VDD location. A list of available decap models will be shown on the form, which were previously defined by the 'DECAP_CELL' keyword in the GSR file. If no appropriate models have been defined, you must provide them in the GSR file.

Note: Corresponding DECAP cell definitions must exist in the LEF file and in the GSR file.

After intentional decaps are characterized through the RedHawk APL flow, obtain the ESC (effective power circuit capacitance) and ESR (effective power circuit resistance) values from the `<cell>.cdev` file. Click on **Commit Adding** to see the decap appear on the layout. The addition of decap is now complete, as illustrated in Figure 7-6 and Figure 7-7. Use **File -> Export ECO** to write out the added decaps and export the ECO files for inclusion in the design on future runs.

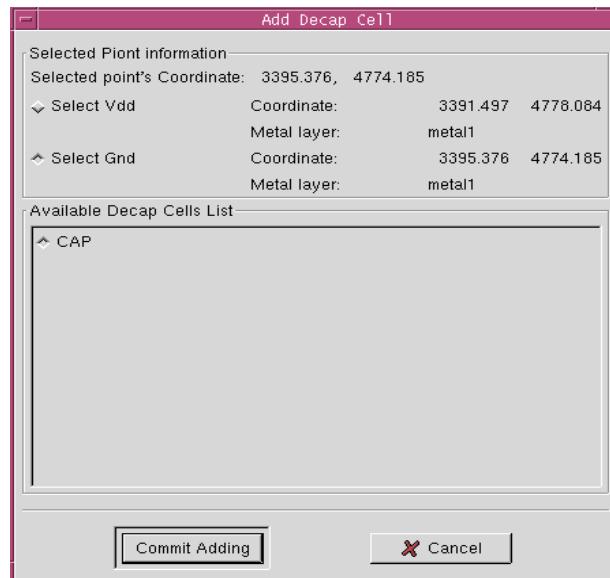


Figure 7-6 Edit -> Add Decap Cell

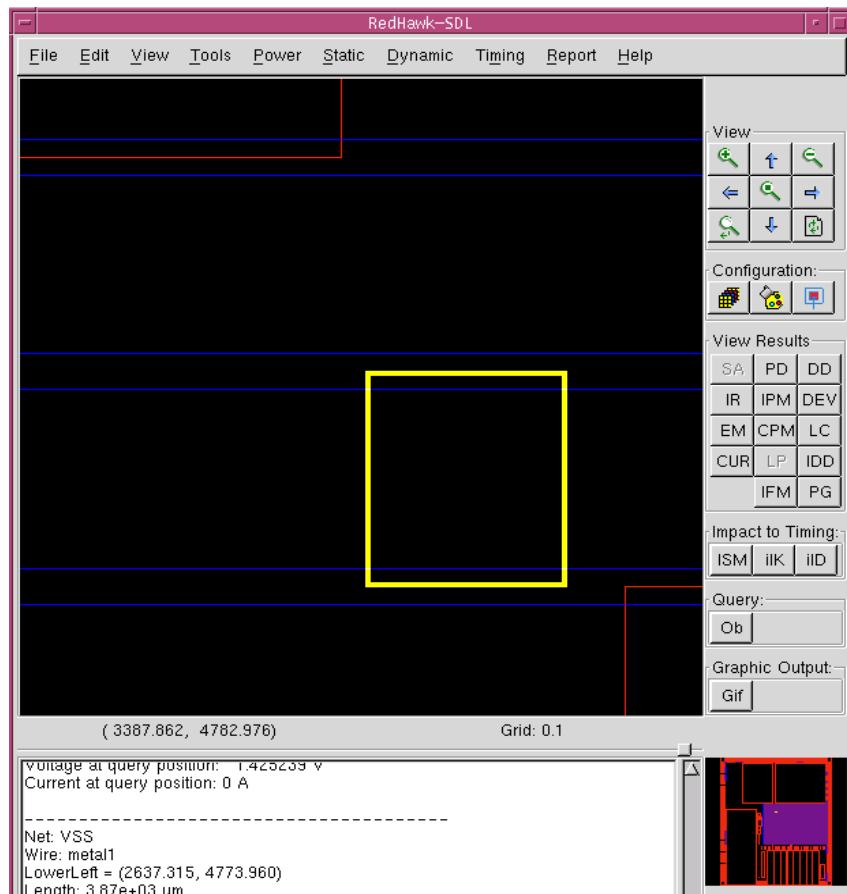


Figure 7-7 Decap added on layout

Adding Metal Layers and Vias or Via Arrays

The following steps allow you to add straps on an additional layer of metal, in addition to the existing metal layers. For example, if the existing layers of metal are metal1 to metal6, the additional layer of metal should be specified as metal7, on top of metal6.

1. Add the extra layer definition to LAYER section of the *.lef* file, which contains the technology definition for layers. In the example below, there are six metal layers. Edit the *.lef* file to add metal7 and via67 as shown below. Make sure that the sequence and numbering are correct (i.e., specify VIA67 and METAL7 after METAL6).

```
LAYER VIA67
    TYPE CUT ;
END VIA67
LAYER METAL7
    TYPE ROUTING ;
END METAL7
VIA via6 DEFAULT
# (Worst case resistance model for via5 = 2.54 ohm/ct) = 2.5400e+00
RESISTANCE 2.5400e+00 ;
LAYER METAL6 ;
    RECT -0.240 -0.190 0.240 0.190 ;
LAYER VIA67 ;
    RECT -0.180 -0.180 0.180 0.180 ;
LAYER METAL7 ;
    RECT -0.270 -0.270 0.270 0.270 ;
END via6
```

Add the metal layer and via information the **.tech* file, as shown below.

```
metal METAL7 {
    Thickness 0.4
    Resistance 0.065
    TC 0
    EM 1.22
    Above DIEL6
}

via VIA67
{
    Width { 0.19 }
    Resistance 1.2
    EM 0.126
    UpperLayer metal7
    LowerLayer metal6
}
```

2. Create the required via arrays in the VIAS section of the top-level *.def* file. Since there is no existing via array definition, you must create the required via arrays in the VIAS section. The via arrays for VIA67 can be created by copying the VIA66 definition in the existing *.def* file and changing the parameters to match VIA67, as shown below:

```
- $TCARY$via6_0_2_1_1420_1420_F + RECT METAL6 ( -1200 -380 )
( 1200 380 ) + RECT METAL7 ( -1260 -540 ) ( 1260 540 ) + RECT VIA67
( -1080 -360 ) ( -360 360 ) RECT VIA67 ( 360 -360 ) ( 1080 360 ) ;
```

3. After the metal layer and via arrays are created, follow the same steps to add pads and straps, as previously discussed.

Undo and Redo

You can perform Undo or Redo operations after making FAO changes, allowing you to try out FAO changes and then reverse them if the results are not satisfactory. Any number of **Edit > Undo** or **Edit > Redo** menu commands can be performed consecutively until the commands in the stack are exhausted. On the TCL command line, ‘fao undo’ and ‘fao redo’ commands also can be executed with the same results.

Automated Grid Optimization and Fixing Procedures

Fixing and Optimization Flow for Static IR Drop Improvement

The primary steps in the FAO static voltage drop optimization flow are outlined in Figure 7-8 below.

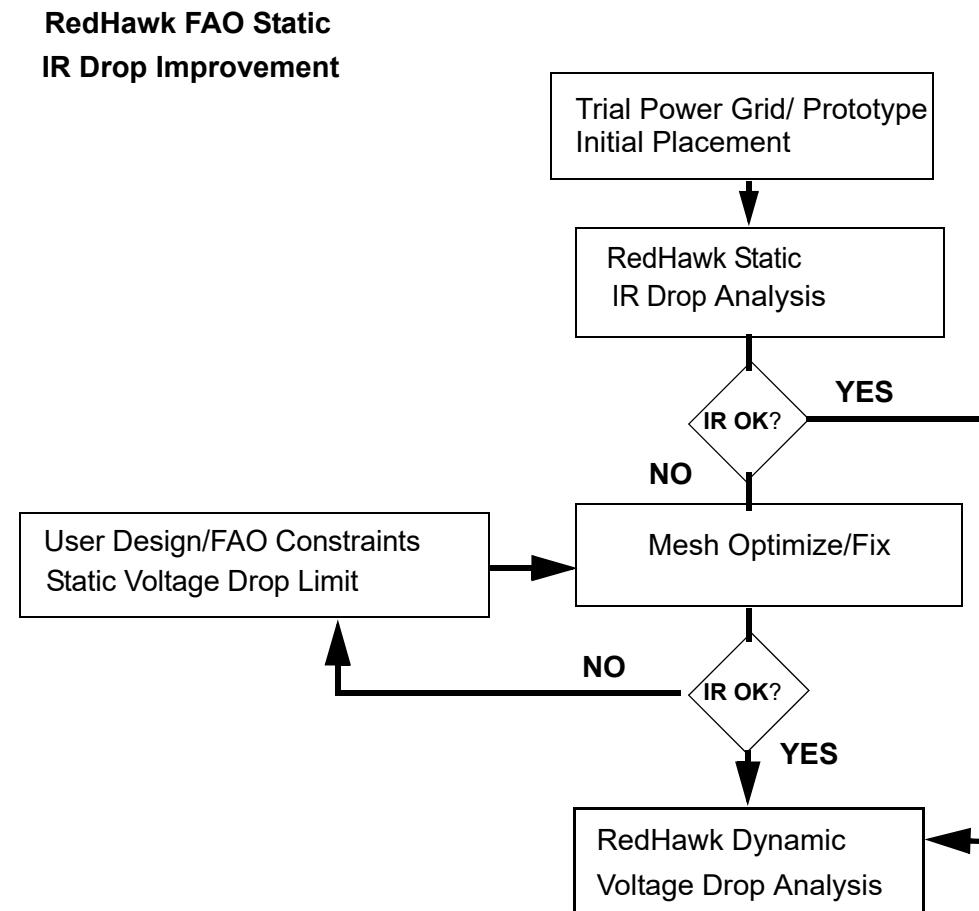


Figure 7-8 RedHawk flow for static IR drop analysis

FAO Procedure for Multiple Vdd Designs

For multiple Vdd domain designs, only one power domain is analyzed and modified by FAO at a time. Specify each power domain in order using the GSR keyword FAO_NETS . For example, if there are two power domains, VDD and VDD_B, first invoke:

```
gsr set fao_nets VDD
```

Now you can run FAO analysis, and perform the needed VDD grid and decap modifications. Following fixing and optimization on the VDD net, use the command:

```
gsr set fao_nets VDD_B
```

Then run FAO analysis and power problem mitigation on the VDD_B net.

If there is more than 1 power net defined in fao_nets, only the first one listed is considered, and RH returns the following warning:

```
WARNING: =====
```

```
WARNING: *** FAO_NETS has > 1 VDD net: Only consider <VDD>
```

```
WARNING: *** Set fao_nets for EACH power net and apply fao separately.
```

```
WARNING: =====
```

Grid Optimization

Mesh Commands

The key commands for using FAO to adjust grid widths are invoked in the GUI or TCL format, and have the following functionality:

- `mesh optimize` - investigates a grid width solution within a defined area (fao_region). All wire segments on the specified metal layers will be adjusted to the same width to solve the voltage drop problem, either for the full chip or within the width of the specified region (but the full height of the chip). With the -taper option, grid wire segments are only resized within fao_region, and not outside of it (not the full height of the chip).
- `mesh fix` - investigates a grid width solution within a rectangular region that includes all “hot spots” (high voltage drop nodes) and their associated fix_window areas. For all hot spots within fao_region, wire segments in vertical and horizontal bands (full width and height of chip) defined by the dimensions of the fix_window, will be adjusted to an appropriate width to solve the voltage drop problem. With the -taper option, grid wire segments are only resized within the specified region (fao_region), and not outside of it.
- `mesh snsalc` - for a selected region and set of layers and nets, mesh snsalc calculates and reports a number representing the relative average sensitivity for modifying the width of wires associated each layer and net selected--the higher the sensitivity number reported (in mv of voltage drop reduction per micron of additional wire width), the more effective a wire width change would be in reducing the worst-case voltage drop within the selected region. Using this command is recommended before performing any grid resizing.
- `mesh sub_grid` - adds a new subgrid mesh, with an area defined by the GSR keyword 'fao_region', including power nets defined by 'fao_sub_grid_nets' and with physical characteristics defined by 'fao_sub_grid_spec'. The new subgrid is targeted to achieve a voltage drop specified by the GSR keyword 'noise_reduction'. The min/max width of the new grids to be sized is specified by 'fao_range'. The minimum width is also used to create the initial subgrid before sizing analysis.

- `mesh set_width` - allows the mesh wire width to be modified for the specified layer(s). By default the wire width is modified equally on both sides of the mesh center line, but the width change can be made on only one side of the wire by specifying the ‘expand_dir’ (or it could be the contracting direction if you are making the width smaller).

Several additional commands are available for modifying the power grid, but are not needed frequently, and are summarized below.

- `mesh add` - allows you to add a specified mesh section to the grid, including specific layers and regions of the design. Useful for easily modifying the grid and evaluating the effect on voltage drop
- `mesh delete` - allows you to delete a specified mesh section of the grid, including specific layers and regions of the design. Useful for easily modifying the grid and evaluating the effect on voltage drop
- `mesh vias` - allows you to add optimal sized vias in a region of the design using specified via models
- `mesh generate` - generates a prototype grid based on the user constraints that are provided in the design constraints file (.dcf)
- `ring add/delete` - adds or deletes a specified power/ground ring

Following is a brief summary of the GSR keywords available to control grid modification. The general TCL syntax for setting GSR keyword values is: `gsr set <variable name> <value>`. All FAO options and GSR keyword values will be collected in **RedHawk** and then the GSR file can be used to set initial values and then save-and-reload will set the GSR keyword values properly. More detailed descriptions of these GSR keywords and their options and syntax are provided in [section Grid Fixing and Optimization Keywords](#) on page C-713.

GSR Keywords for Region-based Grid Width Sizing (mesh optimize)

1. To select IR drop or DvD analysis: FAO_DYNAMIC_MODE (default - off)
2. To set a target voltage drop reduction: NOISE_REDUCTION (input required)
3. To specify a net or group of nets for optimization: FAO_NETS (input required)
- 4, 5. To specify layer and grid width constraints: fao_layers, FAO_RANGE (input required for both)
6. To select accurate or turbo mode: FAO_TURBO_MODE (default is on)
7. To specify an optimization area: FAO_REGION (default - whole chip)
8. To select the number of high voltage drop areas to optimize: NUM_HOTSPOT (default - 1)
9. To specify layer/width relationships: FAO_WIDTH_CNSTR (default: no specification)
10. To specify the percentage voltage drop reduction to be achieved on specific layers: MESH_SEARCH (default - even distribution on selected layers)

GSR Keywords for Hot-spot Based Grid Width Fixing (mesh fix)

GSR keywords 1 through 10 described above for mesh optimize also can be used for `mesh fix`. In this command ‘fao_region’ is used to define the overall evaluation area, and a “fix window” around each hotspot within the region is investigated for a grid width modification, as follows:

11. To select the horizontal and vertical dimension of the area, centered on each hotspot, in which to modify the grid widths: fix_window (default - 100u x 100u)

Several example static IR drop analyses are provided in the following section.

Examples of Grid Optimization

Example A - Full Chip Optimization - Relaxing Static IR Drop

Conditions: Early in design development. Your design allows you to relax (increase) the existing worst case voltage drop and perhaps save grid resources. You want to use full chip uniform grid optimization methodology, so **mesh optimize** is used.

Goal: Perform full power grid optimization, increasing the static IR drop target 15%.

1. Decide on what GSR keyword settings you want to set for performing a ‘mesh optimize’ operation. Assume in this case the initial worst case static IR drop is 18mV on VDD, and we can accept a slightly higher voltage drop without problems. Therefore we will do a full grid optimization to relax this drop by 15%, as shown in the following TCL input lines:

```
gsr set fao_turbo_mode 0
gsr set fao_nets VDD
gsr set fao_layers { {metal6 hor <= 29.4} }
gsr set fao_range { {metal6 8 29.4} }
gsr set noise_reduction -15
```

No region has been defined using ‘fao_region’, so the entire power grid is optimized (default). For this run FAO will do an ‘accurate’ mode search of low static voltage drop areas and optimize only the VDD net. It will examine all METAL6 wires running horizontally of width less than or equal to 29.4um (the existing grid width) as its target wires. FAO will try to relax the worst case static voltage drop by 15%. Note that this will create a reduction in metal usage, so you must specify a width range less than the existing width, such as the 8 to 29.4u range in this example.

2. Run mesh optimize as follows, specifying an output eco file:

```
mesh optimize -eco <filename>.eco
```

3. Based on the requested grid optimization parameters, RedHawk reports the wires it optimized in the region defined (the whole grid in this case), as follows

```
ECO File: fao_mesh_opt_23:15:47_022242005.eco
Region: <0 0 6817.32 8316.08>
FAO Mode: accurate
Simulation Mode: static
Net: VDD
Noise Reduction: -15%
+++++
Initial Worst Static Noise: 17.578mv
Voltage Noise Constraints: 20.2147mv
*=====
*      Searching Target Grids      *
*=====
Net <VDD>:
    Layer <metal6>: identified 44 wires
*=====
*      Sizing Target Grids      *
*=====
```

Following the run, FAO reports 44 wire change recommendations, as shown in the log file below. In this case, there will be less metal usage in the metal6 VDD net as

a result of the relaxation in the static voltage drop limit. Note that new via models also are specified to match the wire changes

```

Optimization succeed: no error
Optimized width of layer metal6: 23.8um
*=====
Metal Usage Change Report
Net <VDD>:
    Layer <metal6>:5.05281e+06um^2 => 4.09119e+06um^2 (-19.03%)
    Total: 5.05281e+06um^2 => 4.09119e+06^2 (-19.03%)
*=====
Replacing optimized grids...
Replace <metal6> mesh grids :
    net <VDD> : 44 wires replaced.
DEF file for new via models: SH_VIA_VDD.def

```

To do a final check on the fix, rerun **RedHawk** with the same setup commands and options as described in the initial preparation steps. Package RLC units are Ohms, pH and pF, respectively:

```

perform extraction
setup pad
setup wirebond
setup package
perform analysis -static

```

The final static voltage drop on the VDD network is 21mV (a 15% relaxation), with a 19% saving in metal resources over the entire design.

Example B - Full Chip Grid Fixing - Reducing Static IR Drop

Conditions: Early in design development. Your design requires a lower voltage drop. You want to use full chip uniform grid optimize methodology, so **mesh optimize** is used.

Goal: Perform full power grid optimization for static voltage, reducing the static IR drop target from the existing 18 mv, as shown in Figure 7-9.

1. Decide on what GSR keyword settings are necessary. This time we will tighten the voltage drop, evaluating the grid width on metal6 for whole design.

Keep in mind if a tighter voltage drop limit is specified (using ‘noise_reduction’), then the fao_range should be increased to allow FAO to increase grid widths . For example, if you want to reduce the voltage limit by 10%, then the following GSR keyword settings should be used:

```

gsr set fao_turbo_mode 0
gsr set fao_nets VDD
gsr set fao_layers {{metal6 hor <= 29.4}}
gsr set fao_range {{metal6 10 35}}
gsr set noise_reduction 10

```

The allowable range of horizontal metal6 widths for this analysis is now 10-35 microns, instead of 8-29.4u in the previous example, and the ‘noise_reduction’ value is now a positive number, 10 percent.

2. Run mesh optimize as follows:

```
mesh optimize -eco <filename>.eco
```

3. Look at your results as described previously.

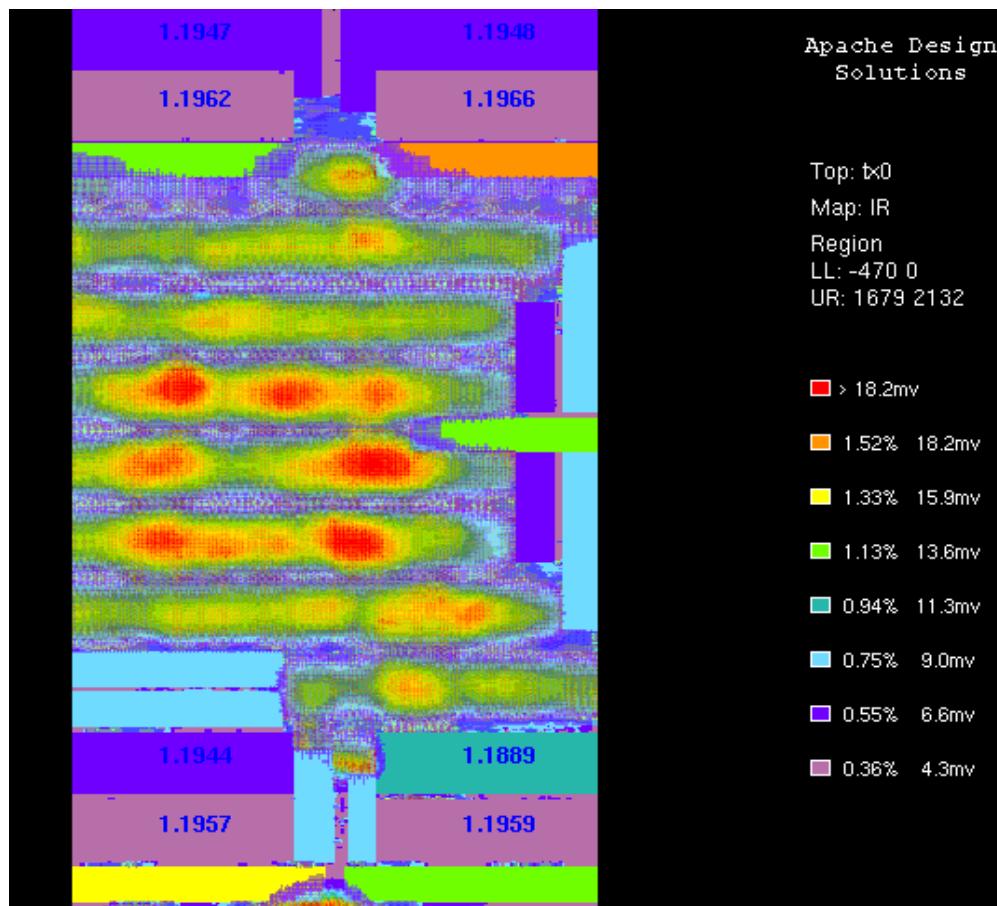


Figure 7-9 Static IR drop map

Example C - Partial Chip Optimization, Relaxing Static IR Drop

Conditions: Later in design development. Your design can accept a higher voltage drop than 18mv, but you do not want to make changes over the entire grid. You want to specify a “fix window” of a particular size, so mesh fix is used.

Goal: To perform a limited power grid fix to relax the static IR drop 15% by modifying wire widths for only a selected area of the power grid.

1. Decide on what GSR keyword settings are necessary. This time we will allow the voltage drop to increase, evaluating the grid width on metal6 for part of the design, as shown in the GSR keyword values set below:

```

gsr set fao_region {2600 2677 6487 5304}
gsr set fao_turbo_mode 0
gsr set fao_nets VDD
gsr set fao_layers { {metal hor <= 29.4} }
gsr set fao_range { {metal6 8 29.4} }
gsr set noise_reduction -15
gsr set fix_window {600 600}
gsr set num_hotspot 1

```

By using the above GSR keyword values, we have defined a bound area with corner locations (2600,2677) and (6487,5304) for grid fixing. RedHawk FAO searches for “hot-spots” within this area (in this run only one is being evaluated),

perform an ‘accurate’ mode voltage drop analysis, and fix only the VDD net within the specified area. The recommended changes to the grid is performed in the selected “fix window” with the hotspot at the center. Note that the grid will not be modified just within the fix window, but across the entire width and height of the chip with the specified x,y dimensions (as modified by the metal layer constraints). FAO will search all METAL6 wires running horizontally of width less than or equal to 29.4um in the “fix window”. It fixes the specified portion of the grid to relax the voltage drop by 15%. Since the specified operation will entail a reduction in metal usage, you must specify an analysis range for METAL6 width less than the existing width (in this case between 8 and 29.4 microns).

2. Run mesh fix as follows:

```
mesh fix -eco <filename>.eco
```

3. Based on the specified grid fixing parameters, RedHawk reports the wires to be fixed within the defined region, finding 1 hotspot, as shown in the following output to the screen and to the log file:

```
ECO File: fao.static.eco
Region: <2600 2677 6487 5304>
FAO Mode: accurate
Simulation Mode: static
Net: VDD
Noise Reduction: -15%
Mesh Style: uniform
Hot-spots Number: 1
Fix-Window Size: 600x600
+++++
Initial Worst Static Noise: 17.578mv
Voltage Noise Constraints: 20.2147mv
Identified 1 hots-spots:
(3123.26, 3058.42)
Fixing Window: <2823.26 2758.42 3423.26 3358.42>
*=====
*           Searching Target Grids           *
*=====

Net <VDD>:
    Layer <metal6>: identified 11 wires
*=====
*           Sizing Target Grids           *
*=====

Optimization succeed: no error
Optimized width of layer metal6: 17.9um
*=====

Metal Usage Change Report
Net <VDD>:
    Layer <metal6>: 5.05281e+06um^2 => 4.56558e+06um^2 (-9.64%)
    Total: 5.05281e+06um^2 => 4.56558e+06^2 (-9.64%)
*=====
Replacing optimized grids...
Replace <metal6> mesh grids :
    net <VDD> : 11 wires replaced.
DEF file for new via models: SH_VIA_VDD.def
```

The recommended fix involves 11 metal6 wire changes. Since the voltage drop was allowed to increase, there is less metal usage in the wires identified for modification.

4. To check on the new static IR drop, execute the standard RedHawk IR analysis again:

```
perform extraction
setup pad
setup wirebond
setup package
perform analysis -static
```

The final static IR drop on the VDD network is 21mV (a 15% relaxation) with a 10% saving in metal resources.

Example D - Partial Chip Fixing, Reducing Static IR Drop

Conditions: Later in design development. Your design requires a lower voltage drop than 18mv, but you do not want to make changes over the entire grid. You want to specify a fix window, so **mesh fix** is used.

Goal: To perform a limited power grid fix to reduce the worst static IR drop 15% by modifying wire widths for only a selected area of the power grid.

1. Decide on what GSR keyword settings are necessary. This time the voltage drop is tightened, and the grid wire widths evaluated for a portion of the design. Since a tighter limit is specified for noise_reduction, the fao_range must be increased to allow wider metal strips. For example, if you want to tighten the voltage drop by 10%, the following GSR keyword settings should be used:

```
gsr set fao_region {2600 2677 6487 5304}
gsr set fao_turbo_mode 0
gsr set fao_nets VDD
gsr set fao_layers { {metal hor <= 29.4} }
gsr set fao_range { {metal6 10 40} }
gsr set noise_reduction 10
gsr set fix_window {1200 1200}
gsr set num_hotspot 1
```

As before, a limited part of the grid is specified for evaluation, and a fix window of 1200 by 1200 microns is specified around each hotspot. The range of grid width is widened to allow wider wires this time to achieve a noise reduction.

2. Run mesh fix as follows:

```
mesh fix -eco <filename>.eco
```

3. The screen and log file output summarizing the results from this run is shown below, and are shown graphically in Figure 7-10.

```
ECO File: drop_improv_15.eco
Region: {2600 2677 6487 5304}
FAO Mode: accurate
Simulation Mode: static
Nets: VDD
Noise Reduction: 10%
Mesh Style: uniform
Hot-spots Number: 1
Fix-Window Size: 1200x1200
Initial Worst Static Noise(mv): 17.578
```

```

Voltage Noise Constraint(mv): 15.8202
Identified 1 hot-spots:
(3123.26, 3058.42)
Fixing window: <2523.26 2458.42 3723.26 3658.42>
*=====
*          Searching Target Wires          *
*=====

Net <VDD>:
    Layer <metal6>: identified 17 wires
*=====
*          Sizing Target Wires          *
*=====

Optimization succeed: no error.
Optimized width of layer metal6: 37.47um
*=====

Metal Usage Change Report
Net <VDD>:
    Layer <metal6>:5.05281e+06um^2 => 5.58121e+06um^2 (+10.46%)
Total: 5.052814e+06um^2 => 5.581216e+06um^2 (+10.46%)
*=====

Replacing optimized grids...
Replace <metal6> mesh grids :
    net <VDD> : 17 wires replaced.
DEF file for new via models: SH_VIA_VDD.def
*****
**** FAO Verify Power Grids
*****
```

Resulting Voltage Noise(mv): 15.8578

Note that the target reduction in voltage drop was achieved by increasing the width of 17 metal6 wires by 10.46%.

4. As before, verify the new static IR drop by executing the standard **RedHawk** IR analysis:

```

perform extraction
setup pad
setup wirebond
setup package
perform analysis -static
```

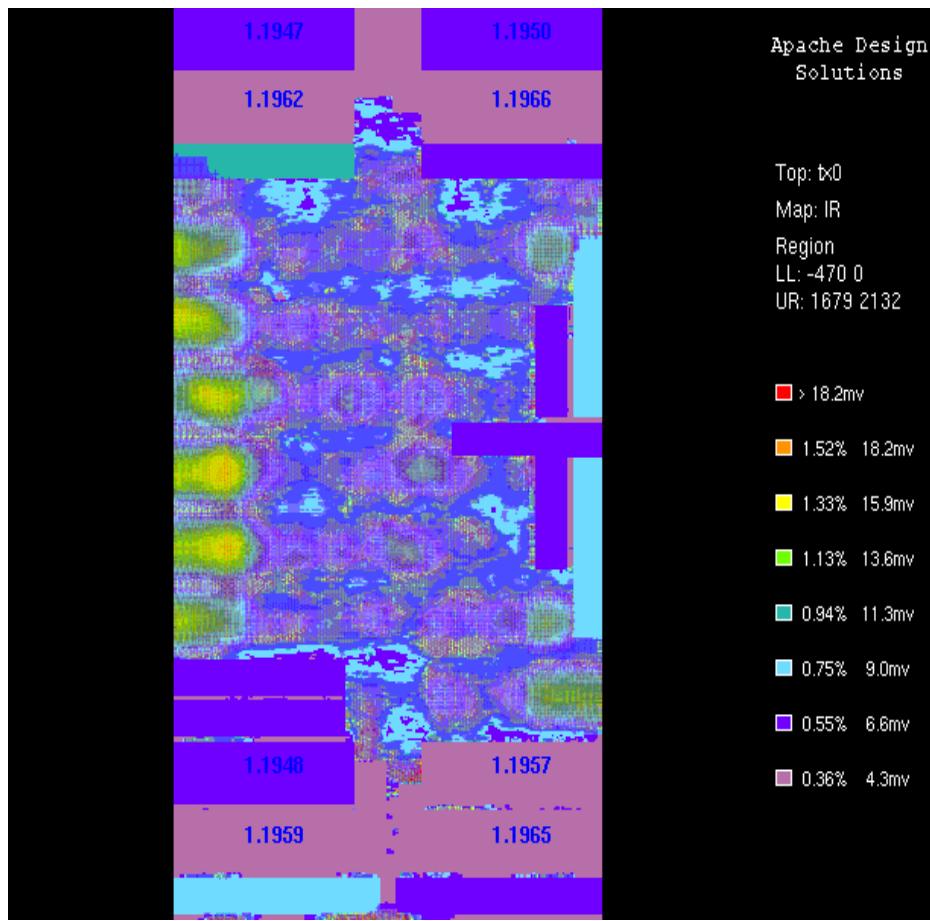


Figure 7-10 Static IR drop map after FAO operations

Example E - Mesh Optimize, -taper option, to reduce regional static IR drop

Conditions: Your design requires a lower voltage drop on two layers in a particular region, so **mesh optimize -taper** is used.

Goal: To perform a 5% static voltage drop reduction with changes on METAL3 and METAL4 layers in a defined FAO_REGION. We choose to limit the width of new wires to between 2 and 5 microns. The '-taper' option is used to ensure that the grid modifications are performed only within the 'fao_region' and not across the entire width and height of the chip.

1. Decide on what GSR keyword settings are necessary. With the conditions described above, the following scripted GSR keyword settings should be used:

```

gsr set fao_nets VDD
gsr set fao_drc 0
gsr set fao_region {x1 y1 x2 y2}
gsr set fao_layers { {METAL4} {METAL3} }
gsr set fao_range { { METAL4 2 5 } { METAL3 2 5 } }
gsr set noise_reduction 5
mesh optimize -taper
export eco mesh_optimize.eco

```

2. By executing the above script, FAO searches for all METAL4 and METAL3 wires in the defined fao_region that can be modified to achieve the voltage drop of 5%. The '-taper' option ensures that the grid fix is performed only within the 'fao_region' and not across the entire width and height of the chip.
3. The ECO changes from this procedure would be the deletion of a number of small wire segments within the defined fao_region on METAL3 and/or METAL4, and larger wires added in their place.

Example F - Mesh Fix, -taper option, to reduce hot spot static IR drop

Conditions: Your design requires a lower voltage drop for hot spots on two layers in a particular region, so **mesh fix -taper** is used.

Goal: To achieve a 2% static voltage drop reduction for hot spots in a defined FAO_REGION with changes on METAL3 and METAL4 layers. We choose to limit the width of new wires to between 2 and 5 microns. The '-taper' option is used here to ensure that the grid modifications are performed only within the 'fix_window' areas around each hot spot and not across the entire width and height of the chip.

1. Decide on what GSR keyword settings are necessary. With the conditions described above, the following scripted GSR keyword settings should be used:

```
gsr set fao_dynamic_mode 1
gsr set fao_nets VDD
gsr set FAO_DRC 0
gsr set fao_region { x1 y1 x2 y2 }
gsr set fao_layers { {METAL4} {METAL3} }
gsr set fao_range { { METAL4 2 10 } { METAL3 2 10 } }
gsr set noise_reduction 2
gsr set fix_window { 600 600 }
gsr set num_hotspot 5
mesh fix -taper
export eco mesh_fix.eco
```

2. By using the above GSR keyword values, we have defined an FAO_REGION area with corner locations (x1,y1) and (x2,y2) for identifying hot spots and grid fixing. FAO searches for 5 hot-spots within this area and finds grid changes that can reduce their voltage drop by at least 2%.
3. The ECO changes from this procedure would be the deletion of a number of small wire segments within the specified 600x600 fix_window area around 5 identified hot spots, and larger wires added in their place.

Automated Fixing and Optimization (FAO) for DvD and Timing

Overview

An overview of the flow for using **RedHawk** to fix and optimize dynamic voltage drop and timing problems is shown in Figure 7-11. Three criteria for running FAO are available:

- instances having the worst-case dynamic voltage drop (DvD) (hot instances)
- all instances having the highest delta slack and delta delay caused by DvD
- instances in critical paths having the highest delta slack and delta delay caused by DvD

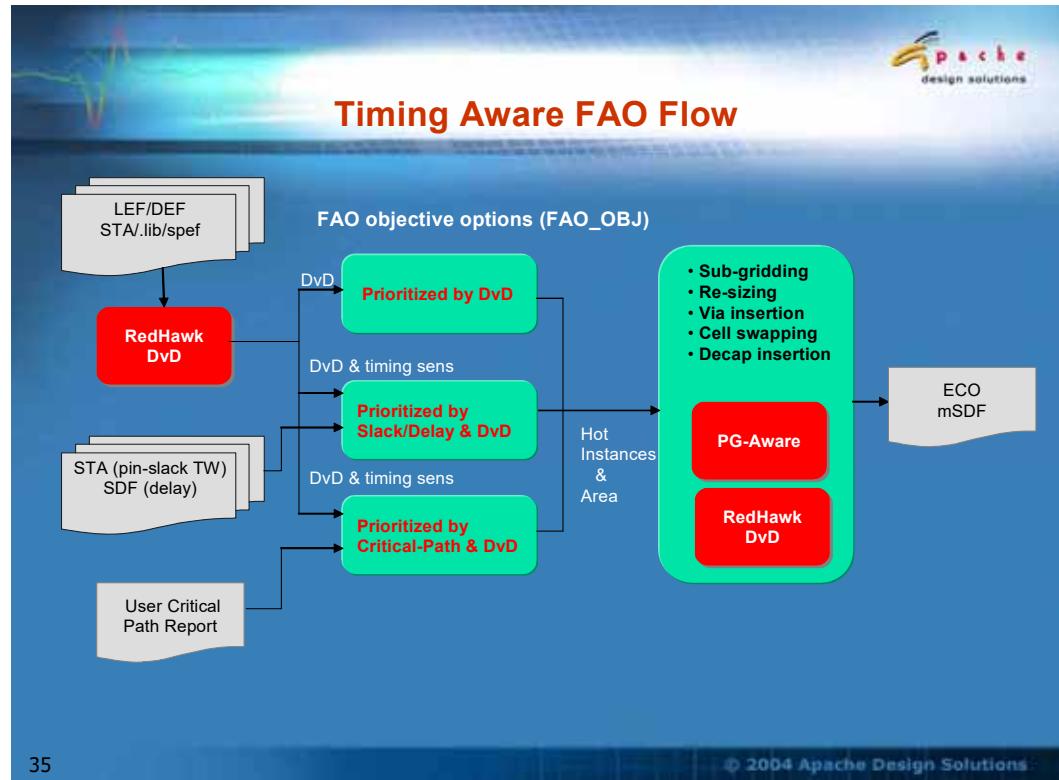


Figure 7-11 FAO Flow

Based on the specified value of the GSR keyword/FAO GSR keyword FAO_OBJ, the program creates an ordered list of instances that, for pure DvD runs have the worst case DvD values, or for the timing-based runs are most likely to be seriously impacted by DvD. The three types of FAO analysis are described below as invoked using the FAO_OBJ keyword:

FAO_OBJ

FAO keyword and GSR keyword that selects the type of instance prioritization and results desired. *Optional; default: DVD*

Syntax

```
FAO_OBJ [ DVD | TIMING ?<slack threshold>? |  
         PATH ?<slack threshold>? ]
```

where

DVD : instances with highest voltage drop are selected for FAO

TIMING <slack_threshold>: all instances in the design with high delta delay and delta slack values caused by DvD are selected for FAO. The default value of the slack threshold is 0.

Note: To use FAO_OBJ TIMING you must execute 'import sdf xxx.sdf' before cell swapping, or the candidates for swapping will be zero. Also, fao_region is still honored to filter out candidates from critical path or timing/slack info.

PATH <slack_worse than> : instances in critical timing paths with high delta delay and delta slack values caused by DvD are selected for FAO. The default value of the slack threshold is 0.

The target instances to be investigated and fixed can be further filtered using parameters such as the value of DvD, the total number of instances, or a special region of interest, as described in the following sections.

Preparation for Decap FAO

Prior to running decap FAO, verify that the available decap cells are properly prepared as follows:

1. Define the list of decaps in the GSR file, using the option DECAP_CELL:

```
DECAP_CELL
{
    <decap_cellname> ? <width_um> <height_um> <C_pF> <R_ohm>
    <metal_layer_name> ?
    ...
}
```

Usually only the list of decap cell names is required, since the other information is in the LEF files.

2. Run APL characterization for decaps in **RedHawk**, which produces the *<cell>.cdev* file containing the R, C and/or leakage power definitions.
3. Insure that the proper physical information is defined in the LEF file and included in LEFS with power/ground pins defined for the layer.

```
MACRO cell_name
    CLASS <class_type> ;
    ORIGIN <value> ;
    SIZE <value> BY <value> ;

    PIN GND
    USE GROUND ;
    PORT
        LAYER <layer_name> ;
        RECT <x1> <y1> <x2> <y2> ;
    END
    END GND

    PIN VDD
    USE POWER ;
    PORT
        LAYER <layer_name> ;
        RECT <x1> <y1> <x2> <y2> ;
    END
    END VDD
END cell_name
```

Decap Modification Operations

RedHawk FAO also allows you to investigate the use of decoupling capacitance to fix dynamic voltage drop and timing problems.

Decoupling Capacitance Modification Commands

The key commands for fixing decoupling capacitance can be invoked in TCL command line or from the GUI. The TCL commands have the following functionality:

- decap advise - finds the total amount of decap that can be placed very quickly, without providing specific placements. The '-place' option uses an algorithm to determine the largest amount of decap that fits in the available row width; saves the result in a user-specified decap placement file (*dpf*).
- decap fill - places decaps using one of several methods:
 - targeted - within a defined rectangle around each specified hot instance (DvD)
 - hot instance area - within a rectangular area encompassing a specified set of hot instances
 - region - within a specified area
 - prefill - adjacent to hot instances after making space in the row
 - uniform - uniformly over all standard cell rows to fill a specified percentage of rows.
- decap remove - removes all ineffective inserted decaps within a specified area, as defined by a specified peak current (default: 30 uA or less).

Additional information on the functions, syntax and usage of these commands is provided in the FAO command reference at the end of the chapter.

Following is a brief summary of GSR keywords available to control the operation of FAO to investigate modifying decoupling capacitance. The general TCL syntax for setting GSR keyword values is: 'gsr set <variable name> <value>' .

A detailed description of these GSR keywords and their options and syntax is provided in section "[GSR File Keywords](#)", page C-573.

Decap Modification Constraints and Interfaces with Other Programs

- Commands 'decap advise -place', 'decap fill' and 'cell swap' honor the DEF file option '+FIXED' so that no modifications are made to fixed instances.
- Commands 'decap advise -place', 'decap fill' and 'cell swap' honor DEF placement blockage if the GSR keyword 'FAO_DRC_PL_OBS' is set to 1 (On).
- In case decaps include metal1 structures, decap placement detects DRC check against metal1 signal routes with GSR keyword 'FAO_DRC_OBS' set to 1 (On). If set to 0 (Off), there is no DRC metal1 signal route check.
- Decaps listed in GSR are not moved during cell swap if designated with option '-fixed_inst/cell' in cell swap command.
- Cells with tie high/low connections are not moved unless the '-include_thtl' option is used (but can move prior to routing).

GSR Keywords Controlling Decap Modification

The following GSR keywords can be used to control the operation of FAO commands: 'decap advise' and 'decap fill'.

1. To select the target voltage drop reduction: NOISE_REDUCTION (input required)
2. To specify the percent improvement in voltage drop: NOISE_LIMIT (no default)
3. To select a fixing area: FAO_REGION (default - whole chip)
4. To select an area around a hot instance to be fixed: FIX_WINDOW (default -40 X 40 u)
5. To select the number of highest voltage drop areas to fix: NUM_HOTINST (default - 10)

6. To limit the total decoupling capacitance added: CAP_LIMIT (default - infinity)
7. To limit the power leakage from all added decaps: LEAKAGE_LIMIT (default - infinity)

The following GSR keywords can only be used for ‘decap advise -place’:

8. To allow decap placement in illegal locations where there is no available space: FAO_DECAP_OVERLAP (default - no overlap)
9. To specify placement of a multiple of each legal decap for placement: DECAP_DENSITY (default - 1)

Other GSR keywords are available that are specific to individual commands. See the command reference at the end of the chapter.

Additional Tools for Fixing High Voltage Drop Areas

Supplemental Power Routing with ‘route fix’

In some situations a high voltage drop node can be repaired by routing a supplementary power wire from it to a nearby low voltage drop node on the same net, providing a parallel path to equalize the voltage drop between the two nodes. This can be performed using the ‘route fix’ command, which uses a shape-based point-to-point router. It is invoked by the following syntax:

```
route fix [-from { x1 y1 x2 y2 <layer>} | -from_pt { xf yf <layer>} ]  
[ -to {x1 y1 x2 y2 <layer>} | -to_pt { xt yt <layer>} ] -width <float>
```

This command finds the shortest DRC-clean legal route for new power/ground wires and vias from a specified target “from” point (or area) and layer to another specified target “to” point (or area) and layer. The “export eco” command defines the added wires/vias for the design database. If the existing via models are not correct for the new routing, any new vias required will be specified and documented in the file ‘adsRpt/faoVIAS.def’. The width of the new routing is specified by the value of option ‘-width’.

This command is used to improve power supply wiring from a stronger point to a weaker point (in terms of voltage drop), as identified from PG-Aware reports.

Cell Moving or Cell Swapping of “Hot Instances”

Some high voltage drop areas can be mitigated by moving a “hot instance” cell to a nearby node on the same grid with better voltage drop conditions, redistributing and equalizing current demand on the chip.

After running **RedHawk** dynamic analysis, the FAO process for moving or swapping instances to improve DvD or timing is as follows:

1. Select the analysis criteria by setting the value of FAO_OBJ keyword:
 - c. DVD - evaluate all instances that have significant DvD impacts
 - d. TIMING - evaluate all instances that have significant slack and delay impacts from DvD
 - e. PATH - evaluate critical paths instances that have significant slack and delay impacts from DvD
2. Select fix_window size - the area around each hot instance to investigate swapping cells (default: 40 X 40 microns)
3. Select DvD criteria desired setting ‘cell swap’ command options:
`-eff_vdd_tw : use average effective voltage (Vdd-Vss) over the timing window to define swapping targets (default DvD criteria)`

- max_vdd_tw : use maximum effective voltage (Vdd-Vss) over the timing window to define swapping targets
 - min_vdd_tw : use minimum effective voltage (Vdd-Vss) over the timing window to define swapping targets
 - min_vdd_all : use minimum effective voltage (Vdd-Vss) over clock cycle to define swapping targets
4. Eliminate any cells from swapping by declaring them “fixed”, or define an exclusive list of cells or instances to swap, using ‘cell swap’ options.
 5. Invoke ‘cell swap’ using the command syntax at the end of this procedure.
 6. FAO generates a list of candidate hot instances to attempt to swap based on the selected criteria and instance constraints.
 7. For each hot instance in the candidate list, FAO first attempts to move it to all open locations in the fix_window rectangle surrounding the instance. If the criteria are improved by moving any candidate cells, it makes the change.
 8. Next FAO attempts to swap the candidate with all other legal instances in the fix_window. If the selected performance criteria is improved by any attempted swaps, it makes the change.
 9. FAO iterates through all swap candidates on the specified instance list.
 10. An ECO report of instances swapped is automatically prepared.
 11. Run the -report and/or -plot options to view the swapping results.

The detailed syntax and usage for the ‘cell swap’ command is as follows:

```
cell swap ?-eco <file_name>?
    ?-decap_cells {cell1 cell2 ...}?
    ?-fixed_cells {cell1 cell2 ...}?
    ?-fixed_insts {inst1 inst2 ...}?
    ?-removable_cells {cell1 cell2 ...}?
    ?-glob|-regexp|-exact?
    ?-include_clock ?
    ?-inst_list {inst1 inst2 ...}?
    ?-inst_file <file_name>?
    ?-search_only?
    ?-eff_vdd_tw? ?-max_vdd_tw?
    ?-min_vdd_tw? ?-min_vdd_all?
    ?-report? ?-plot?
```

where

- eco: specifies the filename of the ECO file describing the changes to the design
- decap_cells: : specifies the names or families of master decap cells that can be used in cell swapping with hot instances. Default: any cells with no logic connection can be used for swapping. Three options define how the cells are specified:
 - glob: allows global wildcard symbols (such as *) to be used to define families of decap cells allowed for swapping
 - regexp: allows regular expressions to be used to define categories of decap cells allowed for swapping
 - exact: supplies the specific names of decap cells allowed for swapping
- fixed_cells {cell1 cell2 ...} : specifies the names of fixed master cells *not allowed* for swapping

- fixed_insts {inst1 inst2 ...} : specifies the names of fixed instances *not allowed* for swapping
- removable_cells {cell1 cell2 ...} : specifies the names or families of decap cells that can be removed from the design, if necessary, during cell swapping.
Default: no decap cells can be removed. The -glob, -regexp, and -exact sub-options are also available for this option, with the same usage as for the -decap cell option.
- include_clock : allows clock elements to be included in cell swapping
- inst_list {inst1 inst2 ...} : specifies exclusive list of instance names for cell swapping
- inst_file <file_name> : specifies file containing exclusive list of instance names for cell swapping
- search_only : only generates file with list of candidate instances for swapping based on selection criteria; no swapping is performed (default file name: *cell_swap_<time>.list*. If you want to change the name, use the '-inst_file <filename>' option to specify.)
- eff_vdd_tw : use average effective voltage (Vdd-Vss) over the timing window to define swapping targets (default DvD criteria)
- max_vdd_tw : use maximum effective voltage (Vdd-Vss) over the timing window to define swapping targets
- min_vdd_tw : use minimum effective voltage (Vdd-Vss) over the timing window to define swapping targets
- min_vdd_all : use minimum effective voltage (Vdd-Vss) over clock cycle to define swapping targets
- report : generates text table of last cell swapping result *cell_swap.rpt*, showing instances swapped, and the change in voltage drop (see Figure 7-12)
- plot : generates histogram of last cell swapping result in file *cell_swap.hist*, showing the number of instances vs. the change in criteria Vdd value selected.

```
*****
**** FAO Cell Swapping Report
*****
Cell Swapping Result Summary
    Total swapping instance number   : 2182
    Actually Moved instance number  : 436
    Actually Deleted instance number: 0
    maximum voltage change (mv)    : 171
    minimum voltage change (mv)    : -18
    **** Please see adsRpt/cell_swap.rpt for detail DvD change.
    **** To view histogram of DvD change, please execute xgraph
        adsRpt/cell_swap.hist.

MEMORY USAGE: 2265 MBytes
TOTAL CPU TIME: 1 hrs 17 mins 52 secs
WALLTIME: 1 hrs 27 mins 56 secs.
*****
```

Figure 7-12 Sample Cell Swapping Report

There are several GSR keywords that specify how the cell swap command operates:

- fao_region: specifies the target analysis area containing high power instances to be fixed
- fao_nets: when there are multiple VDD nets in the design, specifies the name of the VDD net with which the hot instances are associated
- noise_limit: for the list of worst case hot instances, the noise_limit specifies the voltage drop threshold below which the hot instance will not be considered for swapping. For example, if noise_limit is set at 5%, hot instances with less than a 5% voltage drop would not be considered for swapping.
- num_hotinst: specifies the number of worst case hot instances to be swapped in the region
- fix_window: specifies the size of the area centered on each identified hotinst within which the cell would be moved or swapped (default: 40 x 40 u)
- fao_verbose: (0/1) controls the volume of output messages. (default: 0, small amount)

Examples of FAO for DvD

Several examples of dynamic voltage drop improvement using FAO are provided in the following section.

Example G - Full Chip DvD Reduction - Non-overlap Decap and Grid Fix

Conditions: Early in design development. For this example, the initial worst case dynamic voltage drop (DvD) has been evaluated at 306mv by RedHawk. Your design requires a voltage drop at least 10% lower.

Goal: Perform 'decap advise -place' over the full design, attempting to reduce DvD by 10% (to 276mV) by adding decoupling capacitance. If the target is not met with decap additions, follow up with a grid fixing run.

1. Decide on the GSR keyword settings to use. For this run, use all default settings for decap insertion, except specify a noise reduction of 10%, using the command:
`gsr set noise_reduction 10`
2. Run decap placement as follows:
`decap advise -place`
3. During the evaluation, FAO reports the following:

|Target Worst Average DvD percent over TW: 24.1895%(0.27576) (v.s. Vdd=1.14)

This report shows the that the target DvD reduction of 276mv would be 24.2% of Vdd. The measured DvD is the worst case average over the timing window (TW). (The timing window is the range of time in which a particular gate could be switching, according to a Static Timing Analysis (STA) tool).

4. After completing the evaluation, FAO reports the following decap changes:
`Placed decap cells for 29 hot instances.
Added decap 5.092 pF`

FAO then automatically performs another dynamic analysis to determine how close to the target the result is with the initial decap changes, and adds more capacitance for the hot instances if needed. It then reports the total decap added in the second run, so the total decap added in the two placements is now 209.8pf.
`Placed decap cells for 29 hot instances.
Added decap 204.718 pF`

5. Now rerun RedHawk extraction and dynamic analysis as follows, to check the improvement achieved:

```
perform extraction -power -ground -c
setup pad -power -r 0.005 -ground -r 0.005
setup wirebond -power -r 0.001 -l 1000 -ground -r 0.001 -l 1000
setup package -r 0.001 -l 250 -c 5
perform analysis -vectorless
```

In this case the dynamic voltage drop has improved to 284mV, with a target value of 276mV.

6. Obtain a report of the decaps added and the improvement achieved using the 'decap report' command. Then export the required changes into an ECO file, as follows:

```
decap report
export eco decap.1.eco
```

The decap report for the operation follows:

```
*****
**** FAO Decap Report on top Hot Instances
*****
Inserted Decap Inst Num : 3579
Inserted Decap Total : 209.81 pF
V0/V1 : Average (VDD - GND) over TW before/after decap placement.
DvD0/DvD1 : Dynamic Voltage Drop (VDD - V0/V1) before/after decap
placement. Improvement % : (DvD0 - DvD1)/DvD0 %.
inst      V0      V1      DvD0      DvD1      Improvement%
-----
INFO(FAO-75): inst194510 : 0.8343 0.8568 0.3057 0.2832 7.36017%
INFO(FAO-75): inst117100 : 0.8346 0.8574 0.3054 0.2826 7.46563%
INFO(FAO-75): inst76925 : 0.8395 0.8607 0.3005 0.2793 7.05491%
-----
New Top Hot Instances:
inst      &      avgVoltage_TW
-----
inst194510 0.8568
inst117100 0.8574
inst76925 0.8607
-----
Overall DvD improvement = 7.3601 % v.s. initial DvD
MEMORY USAGE: 622 MBytes
*****
```

7. In the ECO file, filter the list of instance names of the added decaps (those with a decap* prefix), then use the RedHawk command:

```
select add "decap*" -glob -linewidth 3
```

to highlight them for viewing on screen.

8. Using the 'decap report' (above) showing a DvD improvement of 7.36%, assign the remaining improvement needed (2.64%) to the next step, wire fixing. Set the new target voltage drop as follows:

```
gsr set noise_reduction 2.64
```

9. Now on the target layers run ‘mesh optimize’ (if you want to look at the entire grid), or ‘mesh fix’, with a specified ‘fix window’ if you want a localized fix, and see if that achieves the desired voltage drop goal.
10. As in the previous examples, rerun RedHawk dynamic simulation to check on your results.

Example H - Full Chip DvD Reduction - Decap Overlap

Conditions: Early in design development. For this example, the initial worst case dynamic voltage drop (DvD) is again found to be 306mv by RedHawk. Your design requires a 10% lower voltage drop. This time we are going to search for a solution in which decaps can be placed anywhere, whether there is legal space or not.

Goal: Perform decap placement over the full design, attempting to reduce DvD by 10% to 276mV by adding decoupling capacitance.

1. Decide on the GSR keyword settings to use. Using the same RedHawk design inputs, this time use the ‘overlap true’ GSR keyword setting for ‘decap advise -place’. This allows the needed decaps to be inserted both legally and also where there is no placement space presently available, to attempt to reduce the local DvD. Set GSR keyword values as follows to request a 10 % reduction in DvD and allow decaps to be placed in illegal areas:

```
gsr set fao_decap_overlap 1
gsr set noise_reduction 10
```

2. As in previous example, verify the amount of decap added and the final DvD savings achieved. A sample output report is shown below:

```
*****
**** FAO Decap Report on top Hot Instances
*****
Inserted Decap Inst Num : 305
Inserted Decap Total      : 229.938 pf
V0/V1 : Average (VDD - GND) over TW before/after decap fix.
DvD0/DvD1:Dynamic Voltage Drop (VDD - V0/V1) before/after decap fix.
Improvement % : (DvD0 - DvD1)/DvD0 %.
inst          V0      V1 DvD0 DvD1 Improvement%
-----
INFO(FAO-75): inst117100 : 0.8346 0.8573 0.3054 0.2827 7.43288%
INFO(FAO-75): inst194510 : 0.8344 0.8609 0.3056 0.2791 8.67146%
INFO(FAO-75): inst76925  : 0.8396 0.8623 0.3004 0.2777 7.55658%
-----
New Top Hot Instances:
inst      & avgVoltage_TW
-----
inst117100 0.8573
inst194510 0.8609
inst76925  0.8623
-----
Overall DvD improvement = 7.73499 % v.s. initial DvD
```

Saving Design Changes with the ECO Command

Once a voltage drop target is met, the ECO file created by the mesh and decap placement operations can be read into RedHawk by using the **File > Import ECO** command from the GUI. (The ECO file is written in an ASCII text format such that it can be easily exported to designers' routing tools to complete the physical design and verification flow.)

Writing an ECO File

The command **File > Export ECO** exports all the recent design changes to an ECO file in the RedHawk working directory. When selected, the pop-up form will query you for a file name, otherwise use the default name *apache.eco*. The format of the ECO file is described below. For the TCL invocation of the 'export eco' command, see [section "export", page D-773](#).

Reading an ECO File

The ECO file can be imported by using the **File > Import ECO** command after the design is read in with the **File > Import Design Data** command. Database setup must be completed also. When this is performed, all what-if changes as specified in the ECO file or *apache.eco* will be included in the design during IR drop and EM analysis.

ECO File Format Definition

NOTE: The ECO file syntax is described below to help you understand the file contents, which are generated automatically by RedHawk. Do not try to create or edit an ECO file manually.

The following is an example of a tabular *.eco file created by the **File -> Export ECO** command:

Keyword	Obj	Obj name (uniq ID)	P/G name	Layer/ via	Rotation	Coordinates	Direction
add	pad	pvdd1	Vdd	metal1	None	x_new y_new (center)	None
delete	pad	pvdd2	Gnd	metal1	None	x_old y_old (center)	None
add	via	via67_new	Vdd1	via7_1	<code>	x_new y_new (center)	None
delete	via	via67_old	Vdd1	via7_2	None	x_old y_old (center)	None
add	wire	wire1	Vdd2	metal7	None	x1 y1 (lower left) x2 y2 (upper right)	[horizontal vertical]
add	wire45	wireCD	Vdd	metal7	None	x3 y3 x4 y4 x5 y5 x6 y6 (trapezoid corners)	None
delete	wire	wire2	Vdd	metal7	None	x1 y1 (lower left) x2 y2 (upper right)	None

The ECO file format description, by column, is as follows:

Column 1: 'Keyword' represents the type of ECO change:

add - a new object added to database

delete - an existing object deleted from database

Column 2: 'Obj' represents the type of object that was changed:

pad - a pad object

via - a via object

wire - a horizontal or vertical wire object

wire45 - a wire object, not on grid

Column 3: 'Obj name (uniqu ID)' represents the unique name of the object.

Column 4: 'P/G Name', for pads is 'VDD' or 'GND', and for wires and vias is the net name

Column 5: 'layer/via' is the name of metal layer or via defined in LEF:

metal1 - metal layers defined in LEF

via7_1 - via definition defined in LEF

Column 6 : represents the coordinates of the object being changed by ECO, except for 'add via':

For pads, indicates the x,y coordinates of the center of the power/ground pad being added or deleted

For 'add via', indicates the rotation code associated with the via added, as follows:

-1 No rotation

0-6 RedHawk rotation code

For 'delete via', indicates the center of the via to be deleted.

For wires, indicates the lower left and upper right x,y coordinates of the bounding box for the wire being added or deleted. Note that you can delete a non-grid wire segment with the command 'delete wire'.

For wire45 objects, x3 y3, x4 y4, x5 y5, and x6 y6 represent the four corners of the trapezoidal wire segment to be added. Any corner can be the first corner specified, but the corners must be listed in counterclockwise order.

Column 7: For 'add via', indicates the x,y coordinates of the center of the via to be added. This via and its size/layer must have already been defined in LEF.

Column 8: For 'add wire', indicates the direction--vertical or horizontal-- of the wire to be added.

Note: ECO added wires are not merged by RedHawk, so if an added wire overlaps another wire on the same net and on the same layer, the capacitance extracted for the overlap portion of the two wires will be double counted (too large), and the resistance for the overlap portion will be that much smaller due to the double counting. For this reason, the amount of overlap should be minimized.

ECO File Translation for Use by Place and Route Tools

ECO file formats can be easily post-processed into formats that are readable by EDA place and route tools. This flexibility provided by RedHawk ensures a smooth data flow between RedHawk and P&R tools.

Fixing and Optimization Command Reference

Descriptions of Mesh Optimization Commands

The following section describes the commands available for performing grid optimization and fixing in RedHawk FAO (alphabetic order). The GSR keywords associated with each command are described in the next section. See the last section for syntax conventions.

Mesh and ring commands - FAO functions to optimize and fix grids.

```
mesh [ add| delete| fix| generate| optimize| sns calc |  
      sub_grid | set_width | vias ] ? args ?  
ring add ? args ?  
ring delete <ring_name>
```

Table 7-1 Grid Optimization and Fixing Commands

Command	Description
mesh add	<p>Allows you to add a specified mesh to the power grid.</p> <p>Associated GSR keywords: none</p> <p>Options</p> <ul style="list-style-type: none">[-vertical -horizontal] Select either horizontal or vertical direction for adding mesh.-window { <x1> <y1> <x2> <y2> } Add mesh in rectangular area specified by corner coordinates 'x1','y1', and 'x2','y2'-layer <layer_name> Add mesh layer named 'layer_name'.-offset <offset_value_microns> Offsets new layer as specified from left edge of selected window vertically, or from bottom edge of window horizontally .-space <space_amount> Add 'space_amount' in microns between the power nets specified in the -net option.-pitch <pitch> Add mesh with pitch of 'pitch' microns.-width <mesh_width> Add mesh with width of 'mesh_width' microns.[-novia -viatop <metal_layer_name_above> -viabottom <metal_layer_name_below>] Add mesh either without a via or with vias going from 'metal_layer_name_below' up to 'metal_layer_name_above'-net { <net1> <net2> ... <netn> } Add mesh associated with net names 'net1' to 'netn'-clip_cell {<macro1> <macro2> ... } Blocks metal from being added inside the boundary of the specified cells, to prevent shorts of pin geometries inside cells.-exclude_regions {{x1 y1 x2 y2} {<excluded_region2_coords>} ... } Excludes one or more specified regions for mesh addition.

Table 7-1 Grid Optimization and Fixing

mesh sns calc	Calculates a number representing the relative voltage drop sensitivity (in mv/um) for changes to wire width for the selected layers and nets within the selected region.
	Associated GSR keywords: fao_region, fao_nets, fao_layers
	Options
	? -taper ? Resizes grid wires on target layers only within the specified region (fao_region), not the full height or width of the chip.
mesh delete	Allows you to delete specified parts of the power grid.
	Associated GSR keywords: None
	Options
	[-vertical -horizontal] Delete mesh in either horizontal or vertical direction.
	-window { <x1> <y1> <x2> <y2> } Delete mesh in rectangular area specified by corner coordinates 'x1', 'y1', and 'x2', 'y2'
	-layer <layer_name> Delete mesh layer named 'layer_name'.
	-net { <net1> <net2> ... <netn> } Delete mesh associated with net names 'net1' to 'netn'.
	-width { ?<Oper>? <width-1> ?<Oper> <width-2>? ? <width>? } Defines grids to be deleted by function of width. <Oper> is a relational operator of the group: =, >, <, >=, or <=. "Width" is defined orthogonal to the wire direction
	-length { ?<Oper>? <length-1> ?<Oper> <length-2>? ?<length>? } Defines grids to be deleted by function of length. <Oper> is a relational operator of the group: =, >, <, >=, or <=. "Length" is defined in the wire direction.
	-out_of_ratio { n m } Defines partial grid wire deletion, or how many grid wires, "n wires out of every m wires", are to be deleted.
	-all Specifies that all available mesh object parameters are deleted except as specifically listed in the 'mesh delete' command.
	-exclude_regions {x1 y1 x2 y2} {excluded_region2_coords} ... Specifies one or more regions to be excluded from the 'mesh delete' command specified, where x1, y1, x2, y2 are lower left and upper right coordinates for the region to be excluded.

Table 7-1 Grid Optimization and Fixing

mesh fix	Allows you to perform hot-spot based grid width fixing.
	Associated GSR keywords: fao_region, mesh_search, fao_layers, fao_turbo_mode, fao_dynamic_mode, fao_nets, fao_range, fao_width_cnstr, noise_reduction, fix_window, num_hotspot
	Options
	? -taper Resizes grid wires on target layers only within the specified region (FAO_REGION), or FIX_WINDOW if defined, not the full height or width of the chip.
	? -eco <eco_filename> Prepare ECO report file 'eco_filename' in RedHawk ECO text format.
mesh optimize	Allows you to investigate and fix voltage drop problems using uniform grid modifications on selected parts of the grid system.
	Associated GSR keywords: fao_region, mesh_search, fao_layers, fao_turbo_mode, sim_mode, fao_nets, fao_range, fao_width_cnstr, noise_reduction, num_hotspot
	Options
	? -taper Resizes grid wires on target layers only within the specified region (FAO_REGION), not the full height or width of the chip.
	? -eco <eco_filename> Prepare ECO report file 'eco_filename' in RedHawk ECO text format.
mesh sub_grid	Adds a set of subgrids defined in a region defined by 'fao_region', nets defined 'fao_sub_grid_nets', and with layers, pitch and spacing defined by 'fao_sub_grid_spec'. The program sizes the new subgrid to meet a preset voltage drop target 'noise_reduction'. The min/max width of the new grids considered is specified by 'fao_range'. The minimum width is also used to create the initial subgrid before adjusting the size.
	Associated GSR keywords: fao_region, fao_sub_grid_nets, fao_sub_grid_spec, fao_range, noise_reduction, fao_turbo_mode, fao_dynamic_mode
	Options
	? -eco <eco_filename> Prepare ECO report file 'eco_filename' in RedHawk ECO text format.

Table 7-1 Grid Optimization and Fixing

mesh vias	<p>Adds specified optimal-sized stacked vias to improve voltage drop - that is, adds the via with the largest possible dimension in the rectangular region overlapping the upper and lower metal layers.</p> <p>Associated GSR keywords: none</p>
	Options
	<p>-toplayer <top_layer_name> Specifies top layer connection of vias to be added.</p>
	<p>-bottomlayer <bottom_layer_name> Specifies bottom layer connection of vias to be added.</p>
	<p>-viarule {<generate_viarule_name>} Specifies which LEF VIARULE_GENERATE rule is used.</p>
	<p>[-window { <x1> <y1> <x2> <y2> } -inst { <inst1> <inst2> ... }] Specifies corner coordinates 'x1', 'y1' and 'x2', 'y2' for window in which to add stacked vias, or specifies windows determined by the locations of cell instances 'inst1' to 'instn'.</p>
	<p>?-net { <n1> <n2> ... }? Specifies nets 'n1' to 'nn' for which to add stacked vias.</p>
	<p>?-topwidth {<tw1> <tw2> ... }? ?-bottomwidth { <bw1> <bw2> ... }? Defines the top widths 'tw1' to 'twn' of the existing top layer and bottom widths 'bw1' to 'bw2' for which stacked vias will be added in the overlapped area.</p>
	<p>?-viamodel { <m1> <m2> ... }? ?-replace? Specifies viamodels 'm1' to 'mn' to be used.</p>
	<p>?-between_layer {<layer1> <layer2>} Specifies a via to be connected between defined layers. The -toplayer and -bottomlayer options having a different direction are used to define the location of the new via using their intersections with <layer1>/<layer2>. No via connections are made outside of the <layer1> to <layer2> range in this usage.</p>
	<p>?-eq_net {<net1> <net2>} Specifies a via be dropped between the wires of the specified top and bottom layers, even though they have different names.</p>
	<p>?-pt_file <filename> Specifies a file listing potential via x,y coordinates. The file format is '<x_coord> <y_coord>', with one entry per line, or when using a script such as <i>report_missing_vias.tcl</i>, entries may use either the form '<x_coord> <y_coord>', or the command form 'marker add -position <x_coord> <y_coord>'.</p>
	<p>?-delete -window { x1 y1 x2 y2 } -viamodel {Via1 Via2 ... } -cut_layer <layer> Specifies vias to be deleted for a specified rectangular area, viamodel, or layer, or viamodel. If you do not define -window, then FAO uses the FAO_Region GSR keyword definition as the window. If FAO_Region is not defined, then the whole chip is used. If -viamodel is not defined, all viamodels in the box are deleted. If neither -window nor -viamodel are defined, then all types of vias in FAO_Region, or on whole chip are deleted. Viamodels are defined in Tech LEF or in top DEF.</p>

Table 7-1 Grid Optimization and Fixing

mesh vias (cont.)	<pre>? -vr_percent_x <value> ? ? -vr_percent_y <value> ? ? -vr_x <value> ? ? -vr_y <value> ? Specifies percentage or length and width constraints to add a via array. Where : -vr_percent_x <value> : percentages of length of via arrays on metal overlap area . -vr_percent_y <value> : percentages of width of via arrays on metal overlap area. -vr_x <value> : length of via array (unit um). -vr_y <value> : width of via array (unit um).</pre>			
mesh vias (cont.)	<p>Manipulates and reports on specified stacked vias to improve voltage drop.</p> <p>Associated GSR keywords: FAO_LAYERS, FAO_LAYERS, FAO_NETS.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">Options</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;"> <pre>?-report_missing ? -exclude_stack_via ? ?-check_valid_vias? Reports if metal overlap can accommodate a valid via model. Note that this option does not work with '-min_width' and GSR 'FAO_LAYERS'. All other switches and keywords are supported. ?-exclude_cell/-exclude_instance : excludes the area covered by all instances of a given cell, or an area covered by a specified list of instances from the missing via report. ?-inst {<inst1> ...} -cell { <cell1> ...} -window { x1 y1 x2 y2 } ? ?-exclude_regions { x1 y1 x2 y2 } ?-exclude_regions_file <file_name> ? ?-sort_by_hot_inst ? ? -reverse_sort_v ? ?-toplayer <layer1> -bottomlayer <layer2> ? ? -gds ? ?-threshold [-1 <voltage_V>] ? ? -min_width <value>? ?-fao_layers { {layer1 [h v]} {layer2 [h v]} ...} ? -marker ? ?-no_partial_overlap ? ?-pitch <via_center-center distance> ?-from_edge [0 1] ? -x_pitch <value> ?? -y_pitch <value> ?-x_offset <value> ? -y_offset <value> ? -ignore_inter_med ?-adjust_offset Specifies missing vias to be reported, by default includes stacked vias. Supports FAO_REGION and 45-degree wires. If the '-exclude_stack_via' option is used, stack vias are excluded. Using the -inst, -cell, or '-window' options reports only missing vias of the specified inst or cell type, or overlapping the defined window boundaries. The default is the full chip. Multiple rectilinear/rectangular regions are also supported in '-window' option in the format "-window {{x1 y1 x2 y2 x3 y3 ..} ...{x4 y4 x5 y5 ...}}".</pre> </td></tr> <tr> <td style="padding: 2px;"> Using '-exclude_regions', multiple rectilinear/rectangular region can be excluded from missing via check (same format as '-window' option). -exclude_regions_file specifies exclude region coordinates in a file. Within the file:</td></tr> </tbody> </table>	Options	<pre>?-report_missing ? -exclude_stack_via ? ?-check_valid_vias? Reports if metal overlap can accommodate a valid via model. Note that this option does not work with '-min_width' and GSR 'FAO_LAYERS'. All other switches and keywords are supported. ?-exclude_cell/-exclude_instance : excludes the area covered by all instances of a given cell, or an area covered by a specified list of instances from the missing via report. ?-inst {<inst1> ...} -cell { <cell1> ...} -window { x1 y1 x2 y2 } ? ?-exclude_regions { x1 y1 x2 y2 } ?-exclude_regions_file <file_name> ? ?-sort_by_hot_inst ? ? -reverse_sort_v ? ?-toplayer <layer1> -bottomlayer <layer2> ? ? -gds ? ?-threshold [-1 <voltage_V>] ? ? -min_width <value>? ?-fao_layers { {layer1 [h v]} {layer2 [h v]} ...} ? -marker ? ?-no_partial_overlap ? ?-pitch <via_center-center distance> ?-from_edge [0 1] ? -x_pitch <value> ?? -y_pitch <value> ?-x_offset <value> ? -y_offset <value> ? -ignore_inter_med ?-adjust_offset Specifies missing vias to be reported, by default includes stacked vias. Supports FAO_REGION and 45-degree wires. If the '-exclude_stack_via' option is used, stack vias are excluded. Using the -inst, -cell, or '-window' options reports only missing vias of the specified inst or cell type, or overlapping the defined window boundaries. The default is the full chip. Multiple rectilinear/rectangular regions are also supported in '-window' option in the format "-window {{x1 y1 x2 y2 x3 y3 ..} ...{x4 y4 x5 y5 ...}}".</pre>	Using '-exclude_regions', multiple rectilinear/rectangular region can be excluded from missing via check (same format as '-window' option). -exclude_regions_file specifies exclude region coordinates in a file. Within the file:
Options				
<pre>?-report_missing ? -exclude_stack_via ? ?-check_valid_vias? Reports if metal overlap can accommodate a valid via model. Note that this option does not work with '-min_width' and GSR 'FAO_LAYERS'. All other switches and keywords are supported. ?-exclude_cell/-exclude_instance : excludes the area covered by all instances of a given cell, or an area covered by a specified list of instances from the missing via report. ?-inst {<inst1> ...} -cell { <cell1> ...} -window { x1 y1 x2 y2 } ? ?-exclude_regions { x1 y1 x2 y2 } ?-exclude_regions_file <file_name> ? ?-sort_by_hot_inst ? ? -reverse_sort_v ? ?-toplayer <layer1> -bottomlayer <layer2> ? ? -gds ? ?-threshold [-1 <voltage_V>] ? ? -min_width <value>? ?-fao_layers { {layer1 [h v]} {layer2 [h v]} ...} ? -marker ? ?-no_partial_overlap ? ?-pitch <via_center-center distance> ?-from_edge [0 1] ? -x_pitch <value> ?? -y_pitch <value> ?-x_offset <value> ? -y_offset <value> ? -ignore_inter_med ?-adjust_offset Specifies missing vias to be reported, by default includes stacked vias. Supports FAO_REGION and 45-degree wires. If the '-exclude_stack_via' option is used, stack vias are excluded. Using the -inst, -cell, or '-window' options reports only missing vias of the specified inst or cell type, or overlapping the defined window boundaries. The default is the full chip. Multiple rectilinear/rectangular regions are also supported in '-window' option in the format "-window {{x1 y1 x2 y2 x3 y3 ..} ...{x4 y4 x5 y5 ...}}".</pre>				
Using '-exclude_regions', multiple rectilinear/rectangular region can be excluded from missing via check (same format as '-window' option). -exclude_regions_file specifies exclude region coordinates in a file. Within the file:				

mesh vias (cont.)	<ol style="list-style-type: none">1. Each line can have only 1 shape.2. User can provide rectangular/rectilinear shapes in the following syntax: # rectangle shape x1 y1 x2 y2 # rectilinear shape x1 y1 x2 y2 x3 y3 x4 y4 x5 y5 x6 y6 ..3. Comments can also be included in the file, but not in the same line of co-ordinates. For example, the following format is invalid 0 0 5 5 #xxx Using the '-sort_by_hot_inst' option sorts missing vias by voltage drop, where default sorting is by average delta V between defined layers. Using -reverse_sort_v reverses the order of sort. The -gds option flags missing via locations both inside the GDS block region and in routes over GDS blocks. By default missing via check ignores all items inside or overlapping the area of a GDS2DEF/GDSMMX block. To flag all missing vias within and overlapping the gds*-based cell region, the -gds option is required. '-threshold <>' is used to set the voltage threshold between overlapped geometries that trigger a missing via report. If the option is set to '-1' the voltage check is not used as a filtering criteria-- for example, if dynamic analysis has not been performed and there are no voltage drop values. Default: 1mV. -min_width ignores segments with less than specified width. -fao_layers <layers...> specifies layers and their horizontal or vertical orientation for which missing vias are to be reported. Length based constraint is also supported. By default, width-based constraint is available -marker specifies the missing via is highlighted in GUI. -no_partial_overlap ignores partial metal overlaps with missing vias. This option works by setting GSR keyword 'FAO_IGNORE_COMPRESS_VIAS' -pitch <via center-center dist> specifies the pitch for missing vias on parallel wires crossing or touching GDS blocks. A pitch may start from either left-edge or right-edge for a horizontal overlap (that is, width > height), or a pitch may start from either bottom-edge or top-edge for a vertical overlap. Use option '-from_edge [0 1]' to specify this, using '-from_edge 0' for parallel nets from left/bottom edge and '-from_edge 1' for parallel nets from right/top edge. - ignore_inter_med ignores wires on all layers between the specified -toplayer <tl> and -bottomlayer <bl>. -x_offset, -y_offset specifies the offset from the overlapping geometry edge to the location of the first via. -adjust_offset specified along with -x_offset/-y_offset and -x_pitch/-y_pitch. -x_offset/-y_offset is a must and is used to generate missing via locations on empty overlaps. If there are some vias existing in the overlap region, abandon the given -x_offset/-y_offset and generate new -x_offset calculated from the left-most via present to the left edge with -x_pitch. Likewise, new -y_offset from bottom most via present to the bottom edge with -y_pitch
------------------------------	---

mesh set_width	Modifies the mesh wire width on specified layers.
	Associated GSR keywords: Fao_Region, Fao_Sub_Grid_Nets, Fao_Layers
	Options
	{ { <layer1> <new_mesh_width1> ? <width_change_dir1> ? } ... { <layerN> <new_mesh_widthN> ? <width_change_dirN> ? } }
	where
	<layerN> Specifies name of layer to have mesh width changed.
ring add	<new_mesh_widthN> Specifies new mesh wire width for <layerN> in microns.
	? <width_change_dirN> ? Specifies direction of width change, if not equal on both sides of centerline (default).
	Direction values are [left right top bot].
ring delete	Adds specified power/ground rings to design.
	Associated GSR keywords: None
	Options
	?-name <ring_name>? ?-window { x1 y1 x2 y2 } ? { [-top bottom left right] -layer <layer_name> -offset <offset> -width <width> -space <space> -net <net1 net2 ...netN> }
	Specifies location and dimensions of two or more power/ground rings. [top bottom left right] are sides of the rings to be added >window { x1 y1 x2 y2 } are corners of a specified window in the design (or chip boundary by default) <layer_name> - name of layer where ring segment is added <offset> - distance (u) from defined window (negative value is toward inside of window) <width> is the width of ring (u), measured toward outside of the chip <space> is the space (u) between rings. Multiple 'width' and 'space' pairs are added (equal to the number of nets specified) <netN> is the net name associated with the rings in the order defined.
	Deletes all elements of the specified power/ground ring from the design.
	Associated GSR keywords:None
	Options
	<ring_name> Name of previously-added ring to be deleted.

Descriptions of Decap Modification Commands

The following section describes the commands available for performing decoupling capacitance fixing in **RedHawk** FAO. The GSR keywords associated with each command are described in the next section.

Decap commands - the following commands provide FAO functions to perform modifications on decoupling capacitance. They are described in the table below.

```
decap [ advise | place | fill | remove | report ] ? arg ?  
  
cell swap ? arg?
```

Table 7-2 Decap Modification Commands

	After running a decap fao command creates an output report in the file <i>adsRpt/faoOverlapDecap</i> listing the total number of decaps, the cap value, and the associated DvD improvement.
	Associated GSR keywords: none
	Options
	? -overlap ? Includes report of number of overlapping decaps in the decap report in the format: <decap_master> loc_x loc_y <num_decaps>
decap report	? [-eff_vdd_tw -min_vdd_tw -max_vdd_tw -min_vdd_all] ? Sets up 'decap report' for the specified DvD criteria, which are defined as: eff_vdd_tw : average effective voltage (Vdd-Vss) over the timing window to define fixing targets (default DvD criteria) max_vdd_tw : maximum effective voltage (Vdd-Vss) over the timing window to define fixing targets min_vdd_tw : minimum effective voltage (Vdd-Vss) over the timing window to define fixing targets min_vdd_all : minimum effective voltage (Vdd-Vss) over clock cycle to define fixing targets. (Default: -eff_vdd_tw)
	Creates a DRC report on FAO decap changes, and, with the '-fix' option, fixes DRC violations for user-placed decap cells.
	Associated GSR keywords: none
	Options
	? -fix ? Uses flipping and removal techniques to fix DRC violations on pin geometries in user-placed decap cells that violate signal/shield routes.
	? -o <filename> ? Specifies the output file describing DRC locations and fixes in a decap DRC report.
decap drc	

Table 7-2 Decap Modification Commands

decap advise	<p>Provides an assessment of the amount decap that could be placed in specified areas of the design, and creates a *.dpf file of results.</p> <p>Associated GSR keywords: fao_region, noise_reduction, noise_limit, num_hotinst, cap_limit, fao_decap_overlap, cap_limit, leakage_limit, decap_density,</p>
	Options
	? -place ? Implements maximum decap placement based on 'advise' analysis. Creates *.dpf file describing placements.
	? -dpf <dpf_filename> ? Specifies the name of the input decap placement file. (Default: adsRpt/fao.dpf)
	? -distribute ? Ignores 'noise_limit' and 'noise_reduction' constraints and fixes decap cells up to 'cap_limit' around 'num_hotinst' of hot instances in proportion to their criteria DvD values. (Default: off)
	? [-eff_vdd_tw -min_vdd_tw -max_vdd_tw -min_vdd_all] ? Specifies which type of DvD criteria is used to select candidate hot instances to investigate for fixing. Instances and associated criteria values are defined in the adsRpt/Dynamic/<design>.dvd lookup table. The criteria are: eff_vdd_tw : use average effective voltage (Vdd-Vss) over the timing window to define fixing targets (default DvD criteria) max_vdd_tw : use maximum effective voltage (Vdd-Vss) over the timing window to define fixing targets min_vdd_tw : use minimum effective voltage (Vdd-Vss) over the timing window to define fixing targets min_vdd_all : use minimum effective voltage (Vdd-Vss) over clock cycle to define fixing targets. (Default: -eff_vdd_tw)
	? [-inst_file <filename> -inst {inst1 inst2 ... }] ? -inst_file <file_name> : defines a file containing exclusive list of instances as target areas for filling. -inst { inst1 inst2 ... } : specifies exclusive list of instances as target areas for filling
	? repeat [-iter <n>] ? Specifies automatic repeat up to n times of 'decap advise - place', then extraction, and DvD analysis. At the end of each iteration, the DvD value is checked against the noise_limit. If the noise_limit is met, less than 1% DvD improvement is achieved, or n is reached, iteration stops. Default n =1.
	?-mmx? Provides an assessment of the amount of decap that could be placed in a custom block or design modeled with MMX methodology, and records the results in the log file.

Table 7-2 Decap Modification Commands

decap place	Provides specific decap placements at hot instance locations based on ‘decap advise’ results showing how much is needed.
	Associated GSR keywords: fao_decap_overlap, cap_limit, leakage_limit, decap_density, noise_limit
	Options
	? -dpf <dpf_filename> ? Specifies the input decap placement file. (Default: <i>adsRpt/fao.dpf</i>)
decap qor	? -eco <eco_filename> ? Prepares ECO report file in RedHawk ECO text format.
	Measures QoR (results quality) of decaps inserted by FAO.
	Associated GSR keywords: decap_fill.
	Options
	? -i <input_decap_inst_file> Specifies input decap instance list file.
	? -o <output_dir> Specifies the output directory name.

Table 7-2 Decap Modification Commands

decap fill	<p>Provides decap filling in available spaces. If neither '-fao_region' nor '-hot_area' are defined, spaces in the fix_window area around each hot instance are filled. Leakage of the decap is considered when you use the GSR keyword 'FAO_DECAP_FILL_ALG'.</p> <p>Associated GSR keywords: fao_region, noise_reduction, noise_limit, num_hotinst, cap_limit, fao_decap_overlap, cap_limit, leakage_limit, decap_density, fao_max_shift, fao_row_vdd_site.</p>
	Options
	<p>? repeat [-iter <n>]?</p> <p>Specifies automatic repeat of 'decap fill'. After each iteration, DvD value is checked against the noise_limit. After the first normal 'decap fill' run, subsequent iterations place decaps (specified with 'density_step') at the top 5% hot instance locations in the 'fix_window', then extraction and DvD analysis. When the noise_limit is met, less than 1% DvD improvement is achieved, or n is reached, iteration stops. Default n =3.</p>
	<p>? -density_step <s> ?</p> <p>Specifies the number of decaps to add at each of the top 5% hot instance locations in the 'fix_window' for each 'repeat' iteration after the first 'decap fill'. Default s=1.</p>
	<p>? -eco_name_pattern <string> ?</p> <p>Allows a specified alpha-numeric string to be added to the names of a group of decaps being placed during a 'decap fill' operation. Default string = SH. Names can be of the form: '<decap_cell>_<eco_string><number>'</p> <p>The value of this option overrides any value set using the ECO_NAME_PATTERN GSR keyword.</p>
	<p>? -no_adjust_fix_window ?</p> <p>Turns off default proportional adjustment of the fix window size per voltage drop so all fix_windows are specified size (default 300x300 microns). If this option is <i>not</i> invoked the highest DvD hot instance has the specified fix_window size, and the size of the fix_window for other hot instances is reduced in proportion to the DvD down to 50% of the maximum fix_window dimension for the lowest hot instance specified.</p>
	<p>? -clone_cells { <list of block cellnames> } ?</p> <p>For a block instantiated multiple times at top level, replicates the fill of decap cells exactly the same in all instances of the block, including different orientations. An ECO report is generated listing the decaps added and the numbered clones of the decap instantiation in each block.</p>
	<p>?-prefill ? -no_ff_move ? ? -ignore_fixed ?</p> <p>First performs decap prefill described in the '-prefill_only', then performs the selected 'decap fill' command. If '-no_ff_move' used, no FFs defined in LIB file are moved. Typically used after CTS. If '-ignore_fixed' is used, "fixed" elements such as instances can be moved in executing the decap fill operation.</p>
	<p>?-uniform <percentage></p> <p>Places decap cells uniformly over all standard cell rows to fill the specified percentage of rows (may cause overlaps with existing standard cells).</p>

Table 7-2 Decap Modification Commands

Options	
decap fill (cont.)	<p>? -fao_region ? If selected, uses definition in GSR keyword for x,y coordinates of rectangle within which all spaces will be filled with decap. (default: whole design)</p>
	<p>? -hot_area? If selected, uses definition in GSR keyword for rectangular area to be filled that includes all hot instances meeting constraints on total capacitance, leakage, and noise_limit; (default: if -fao_region and -hot_area are not defined, the fix_window area around each hot instance is filled).</p>
	<p>? [-eff_vdd_tw -min_vdd_tw -max_vdd_tw -min_vdd_all] ? Specifies which type of DvD criteria is used to select candidate hot instance areas to investigate for filling. Instances and associated criteria values are defined in the <i>adsRpt/Dynamic/<design>.dvd</i> lookup table. The DvD criteria are:</p> <ul style="list-style-type: none"> eff_vdd_tw : use average VDD over the timing window to define filling targets (default) max_vdd_tw : use maximum VDD over the timing window to define filling targets min_vdd_tw : use minimum VDD over the timing window to define filling targets min_vdd_all : use minimum VDD over clock cycle to define filling targets
	<p>? [-inst_file <filename> -inst { inst1 inst2 ... }] ? -inst_file <file_name> : defines a file containing exclusive list of instances as target areas for filling. -inst { inst1 inst2 ... } : specifies exclusive list of instances as target areas for filling (Default: null)</p>
	<p>? -prefill_only ? Moves cells adjacent to each defined hot instance as far as possible away from them in the row, up to the value of GSR keyword FAO_MAX_SHIFT (5 um default). Decap cells are then placed in the larger of space available on either side of hot instance. Cells that have +FIXED flag in DEF, or are listed in the '-fixed_insts' or '-fixed_cells' options, are not shifted.</p>
	<p>? [-fixed_insts -fixed_cells] { ... } ? Fixes the location of listed instances or cells, and prevents any movement by the '-prefill_only' or '-prefill' commands.</p>
	<p>? -replace_filler {<filler1> ...} ? or ? 'replace_filler_file <filename>' ? Specifies existing filler cells to be replaced with decap cells (GSR keyword DECAP_CELL). Either list the filler cells, or specify a file containing a list of filler cells to be replaced. -inst_type specifies a 'FIXED' or 'PLACED' attribute for -replace_filler, or 'ALL' includes all types of filler cells (default). -replace_filler_hot_area automatically generates a "hot box" to replace ineffective fillers (the most significant area to replace fillers with decaps, based on the selected DvD criteria option. Wild cards for <filler_name> are allowed.</p>

Table 7-2 Decap Modification Commands

decap remove	<p>Removes all “ineffective” decap cells, as defined by the value of the ‘-imax_less_than’ criteria (default: 30uA).</p> <p>Associated GSR keywords: fao_nets, fao_region, fix_window, noise_limit, noise_reduction, num_hotinst, cap_limit, fao_decap_overlap, cap_limit, leakage_limit, decap_density, fao_verbose</p>
	Options
	<p>? -imax_less_than <peak_current> ?</p> <p>Specifies the smallest peak decap current to keep a decap in place and not remove it (default: 30 uA). Decap list: <i>adsRpt/Dynamic/decaps.rpt</i>.</p>
	<p>? -dvmax_less_than ?</p> <p>Specifies a small delta voltage associated with the peak decap current as a threshold below which no decap removal is performed (default: 0)</p>
	<p>? -fao_region ?</p> <p>If selected, specifies that decaps would be removed only within the rectangular region defined by the GSR keyword, according to other option values set or their default values.</p>
	<p>? -hot_area ?</p> <p>If selected, specifies a rectangular area that includes all hot instances (as defined by noise_limit and num_hotinst) that will have no decaps removed.</p>
	<p>? -user_decap ?</p> <p>If selected, also removes user-specified decaps, in addition to FAO-added decaps.</p>
	<p>? [-eff_vdd_tw -min_vdd_tw -max_vdd_tw -min_vdd_all] ?</p> <p>Specifies which type of DvD criteria is used to select candidate hot instances to investigate for decap removal. Instances and associated criteria values are defined in the <i>adsRpt/Dynamic/<design>.dvd</i> lookup table. The criteria are:</p> <ul style="list-style-type: none"> eff_vdd_tw : use average effective voltage (Vdd-Vss) over the timing window to define removal targets (default) max_vdd_tw : use maximum effective voltage (Vdd-Vss) over the timing window to define removal targets min_vdd_tw : use minimum effective voltage (Vdd-Vss) over the timing window to define removal targets min_vdd_all : use minimum effective voltage (Vdd-Vss) over clock cycle to define removal targets
	<p>? -ignore_cap_limit ?</p> <p>After removing ineffective decaps based on the imax criteria, this option stops decap removal, even if cap_limit is exceeded (default: off - decap removal continues on decaps associated with lowest-ranked “best” DvD instances in the target list until both capacitance and leakage limits are achieved).</p>
	<p>? -no_decap_report</p> <p>Cancels decap removal reporting to speed up repetitive removal process.</p>

Table 7-2 Decap Modification Commands

decap remove (cont.)	? -ignore_leakage_limit ? After removing ineffective decaps based on imax criteria, this option stops decap removal, even if leakage_limit is exceeded (default: off - decap removal continues on decaps associated with lowest-ranked “best” DvD instances in the target list until both capacitance and leakage limits are achieved).
	? -clone_cells ? Removes all identical decap cells from all cloned instances if none of them cause a degradation of DvD performance. If any decap removals would increase DvD, none of cloned decap cells are removed.
	? -all ? Regardless of other options set or their default values, removes all decaps on the chip, or, if the -fao_region option is set, within the specified region.

Table 7-2 Decap Modification Commands

cell swap	Moves defined hot instance cells to open spaces, or swaps them with either decap cells or cells with lower power demand in a specified area.
	Associated GSR keywords: fao_region, fao_nets, fix_window, noise_limit, num_hotinst, fao_verbose
	Options
	? -eco <ECO_filename> ? Prepares ECO report file ' <i>eco_filename</i> ' in RedHawk ECO text format.
	? -decap_cells: ? Specifies the names or families of master decap cells that can be used in cell swapping with hot instances. Default: any cells with no logic connection can be used for swapping. Three options define how the cells are specified: -glob: allows global wildcard symbols (such as *) to be used to define families of decap cells allowed for swapping -regexp: allows regular expressions to be used to define categories of decap cells allowed for swapping -exact: supplies the specific names of decap cells allowed for swapping
	? -fixed_cells { cell1 cell2 ... } ? Specifies the names of fixed master cells <i>not allowed</i> for swapping
	? -fixed_insts { inst1 inst2 ... } ? Specifies the names of fixed instances <i>not allowed</i> for swapping
	? -removable_cells { cell1 cell2 ... } ? Specifies the names or families of decap cells that can be removed from the design, if necessary, during cell swapping. The '-glob', '-regexp', and '-exact' sub-options are also available for this option, with the same usage as for the '-decap cells' option (default - no decap cells are removed).
	? -include_clock ? Allows clock elements to be included in cell swapping.
	? [-inst_file <filename> -inst { inst1 inst2 ... }] ? -inst_file <file_name> : defines a file containing an exclusive list of instances as targets for swapping. -inst { inst1 inst2 ... } : specifies an exclusive list of instances as targets for swapping
	? search_only Only generates a file with a list of candidate instances for swapping based on selection criteria; no swapping is performed (default file name: <i>cell_swap_<time>.list</i> . If you want to change the filename, use the '-inst_file <filename>' option to specify).

cell swap (cont.)	[-eff_vdd_tw -min_vdd_tw -max_vdd_tw -min_vdd_all] Specifies which type of DvD criteria is used to select candidate hot instances to investigate for decap swapping. Instances and associated criteria values are defined in the <i>adsRpt/Dynamic/<design>.dvd</i> lookup table. The criteria are: eff_vdd_tw : use average effective voltage (Vdd-Vss) over the timing window to define swapping targets (default) max_vdd_tw : use maximum effective voltage (Vdd-Vss) over the timing window to define swapping targets min_vdd_tw : use minimum effective voltage (Vdd-Vss) over the timing window to define swapping targets min_vdd_all : use minimum effective voltage (Vdd-Vss) over the clock cycle to define swapping targets
	? -report ? Generates text table of the last cell swapping results in file <i>cell_swap.rpt</i> , showing instances swapped and the change in voltage drop.
	? -plot ? Generates a histogram of the last cell swapping results in file <i>cell_swap.hist</i> , showing the number of instances vs. the change in the criteria DvD value selected.
	? -swappable_cells {cell1 cell2 ...} ? Lists master cells available for swapping.
	? -ff_only ? Specifies that only instances directly connected to a flip-flop input should be moved. Used in conjunction with the '-swappable_cell' option, only listed cells that connect to a flipflop input are swapped to improve their voltage drop.
	? -pre_proc ? Specifies that hot instances in the row with the highest average voltage drop have their cells swapped first. Then hot instances in rows with the next highest average voltage drop are done in order.
	? -to_scan_in_only <cell1 cell2 ...> ? Specifies that only instances that are attached to scan-in pins of specified scan cells are included in cell swapping.
	? -include_thtl ? Specifies that instances with tie-high and tie-low pins can be included in cell swapping. By default these cells are excluded.
	? -drc_m2 ? Specifies that any instances that have metal2 features cannot be swapped to a target area with metal2 access.

Supplementary Voltage Drop Fixing Commands

Table 7-3 Route Fix Commands

Command	Description
route fix	<p>Finds the shortest DRC-clean legal route for a new power/ground wire and vias from the specified target “from” point (or area) and layer to another specified target “to” point (or area) and layer.</p> <p>Associated GSR keywords: None</p> <p>Options</p> <p>[-from { x1 y1 x2 y2 <layer>} -from_pt { xf yf <layer>}] Specifies the origin area or x,y point and layer for new pwr/grnd routing.</p> <p>[-to {x1 y1 x2 y2 <layer>} -to_pt { xt yt <layer>}] Specifies the terminating area or x,y point and layer for new pwr/grnd routing.</p> <p>-width <routing-width> Specifies the width of the new pwr/grnd routing in um.</p>

GSR Keywords Supporting FAO Functionality

Special GSR keywords modify the function of the ‘mesh optimize’, ‘mesh fix’, ‘route fix’, ‘decap advise’/-place’, and ‘cell swap’ commands, and are described in detail in the [section "FAO General Keywords", page C-711](#), [section "Grid Fixing and Optimization Keywords", page C-713](#), and [section "Decap Optimization Keywords", page C-717](#).

Command and GSR Keyword Syntax Conventions

< x >	x is a variable name or value to be specified
[a b c]	one of a, b, or c must be selected
? x ?	x is an optional parameter
{ a b c }	a, b and c parameters all must be specified
{ x } ...	an element such as {x} can be added
+ - * /	standard arithmetic operators for add, subtract, multiply, divide

Chapter 8

Analysis of DvD and Cross-coupling Noise Impacts on Timing

Introduction

As designs moved below 65 nm scale, smaller geometries and higher densities increased crosstalk noise and coupling delay, adversely affecting the chip's overall timing. Today, most existing timing analysis tools are based on static analysis at the cell-level (or gate-level) and only consider cell delay and coupling delay in timing analysis.

However, as more designs use 32 nm scale technologies and beyond, the traditional methods of static analysis are no longer adequate, especially when considering the impact of power integrity on timing. It is becoming a necessity to consider dynamic voltage drop and ground bounce caused by simultaneous switching of core, I/O, and memories in analyzing and closing overall chip timing.

Typically, design tools rely on cell-based abstraction, coarse table-lookup model, and semi-accurate fast SPICE engines, which can result in inaccurate, overly constrained, and marginally guard-banded designs. In reality, sub-90nm designs require a solution that can concurrently analyze all the nanometer scale physical effects found in real silicon, such as coupling capacitance and dynamic voltage drop, or risk significant timing errors or noise problems close to tapeout.

Impacts on Timing

The waveforms in Figure 8-1 demonstrate the impact of dynamic voltage drop and power/ground noise on the circuit timing. The waveform labeled “without PI and SI noise” shows a typical, well-behaved rising waveform.

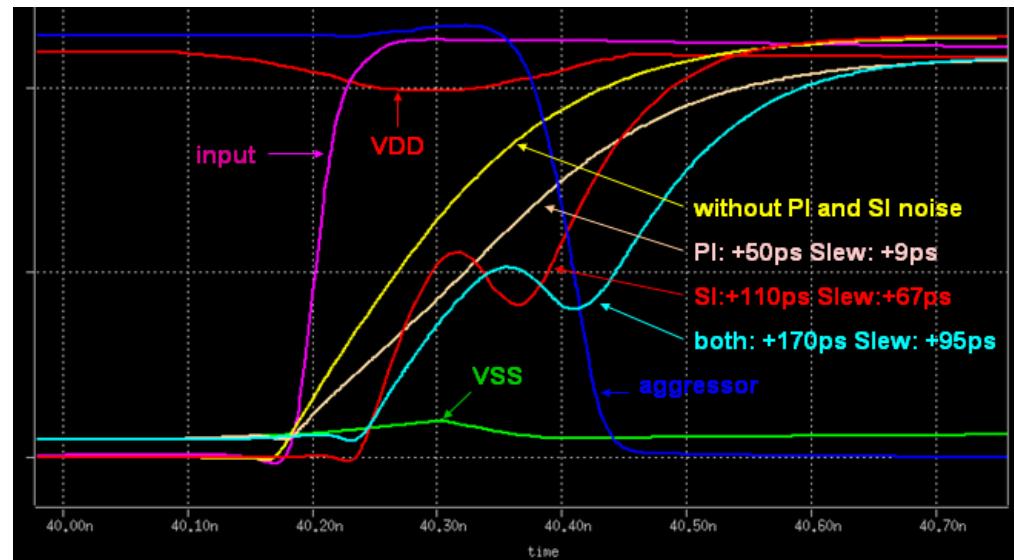


Figure 8-1 Signal noise (SI) and voltage drop (PI) impact on timing.

Overview

The **RedHawk** timing analysis tools can identify and analyze potential signal crosstalk and timing problems caused by power and ground noise, so that corrections can be made prior to design signoff. The tools leverage **RedHawk**, a foundry-certified, sign-off accurate, full-chip dynamic power integrity solution, and NSPICE, a high capacity, high performance SPICE simulator, to deliver the most accurate timing analysis solution for designs at 65nm gate size and below. For jitter analysis, both fast Fullchip and also Spice-based Sign-off solutions are available.

Timing analysis considers the concurrent effects of signal integrity and dynamic power integrity on clock tree and critical timing paths, including I/O, IP, and memories. It accurately computes the gate propagation delay, coupled delays, interconnect delays, rise time degradation for low supply voltage and cross-talk influence from adjacent wires, while precisely tracking the signal waveform, cross-talk noise, and glitches. is used close to sign-off to provide timing results that reflect actual silicon performance and quickly reach noise closure. Figure 8-2 shows the overall design flow.

By default in the **RedHawk** integrated flow it concurrently analyzes the effects of power integrity (PI) and signal integrity (SI) to consider coupling RLC networks, aggressor drivers and receivers, worst/best case coupling delays, aggressor vector sensitization and alignment, timing window filtering, and power/ground dynamic voltage drop. Signal integrity and power integrity analyses also can be performed independently if desired.

Timing analysis is most conveniently run within the **RedHawk** environment, either using the TCL command line or the GUI menus. Many types of useful graphical results are then available, in the form of waveforms, clock trees and jitter. However, if the necessary input files have been generated in a previous **RedHawk** run, timing analysis can be run in batch mode from a TCL command line, and the results evaluated using text file outputs.

This chapter describes the procedure for running **RedHawk** impact on timing analysis.

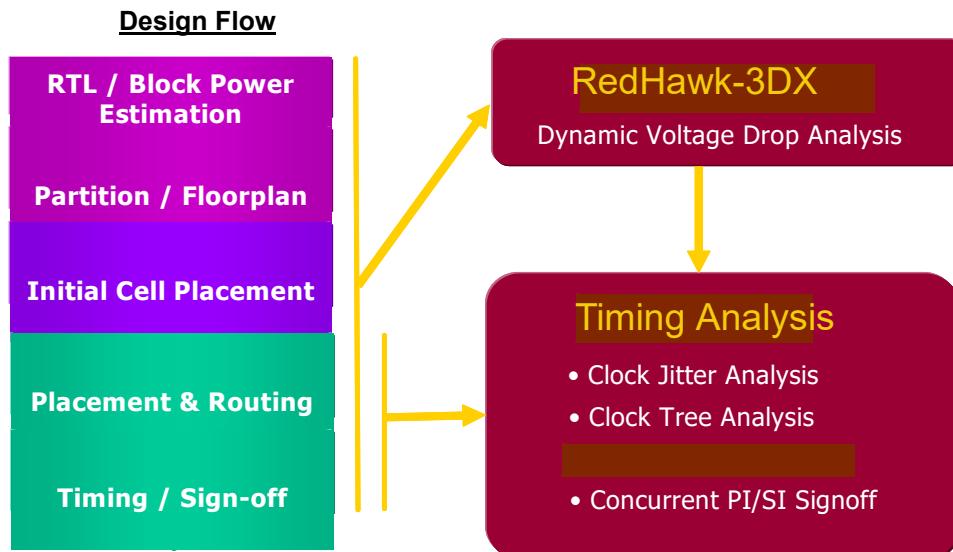


Figure 8-2 RedHawk timing analysis design flow.

The following types of **RedHawk** circuit timing analyses are available to help avoid adverse timing effects and functional failures in a design:

- Clock tree analysis, which includes both jitter and skew analysis

- Critical path delay analysis

These features are discussed in more detail in the following sections.

PJX Clock Tree Jitter Analysis

Clock signal quality is essential for timing closure in high frequency designs. Clock tree jitter in particular can have serious impacts on clock signal arrival time, which usually causes timing failure. Clock tree jitter can be caused by several noise sources, such as crosstalk noise and simultaneous switching noise. RedHawk performs multiple-cycle Spice simulation to accurately predict clock tree jitter results. Designers can verify that the jitter in their clock networks in the presence of Vdd/Gnd noise is within specification and that the duty cycle variation has been accounted for.

RedHawk provides two Clock Jitter Solutions--a High Capacity Fullchip level ATE-based Jitter Solution and a Sign-off accurate NX-based Jitter Solution. The two types of RedHawk clock tree jitter analysis are described in Figure 8-3. You can perform separate fullchip and sign-off jitter analyses.

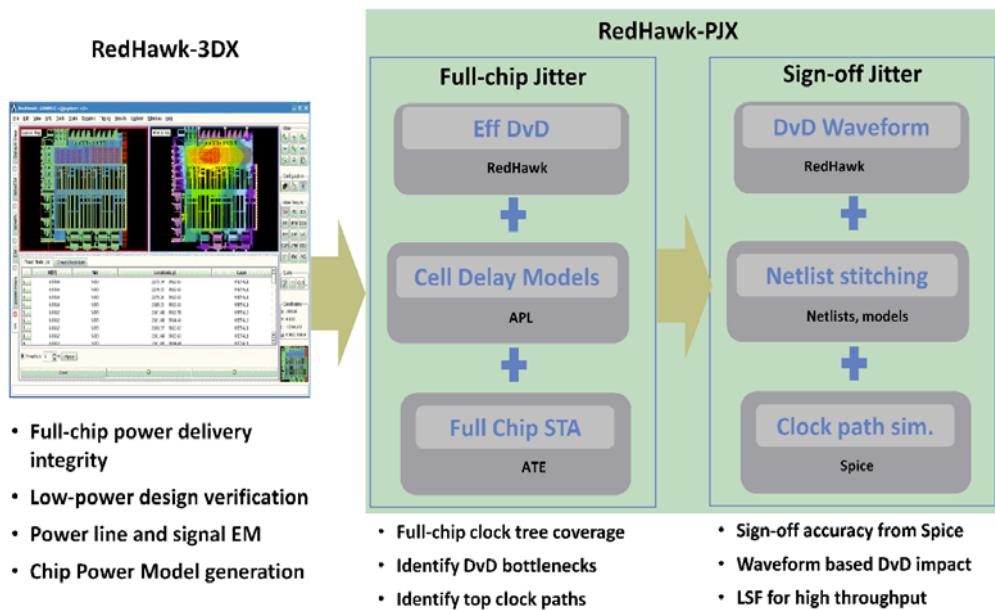


Figure 8-3 RedHawk PJX Clock Tree Jitter Analysis

The work flow for integrated RedHawk PJX Clock Tree Jitter Analysis is as follows:

**Setup Design -> Power Calculation -> Extraction -> DvD Analysis ->
Fullchip Jitter Analysis -> Sign-Off Analysis**

Skew Analysis

Optimized clock trees and clock meshes are essential to good digital circuit design, primarily for clock skew control. RedHawk allows you to easily analyze how cross-talk and power / ground grid noise affects clock skew. Clock tree and clock mesh design can be optimized using the high quality SPICE accuracy of RedHawk.

Analysis Modes

There are four modes available to run clock tree jitter and clock tree skew analyses, as described below:

- Basic analysis mode - uses ideal voltages to evaluate clock tree and clock mesh parameters with no capacitance coupling analysis. This mode is typically used to verify timing results with those from PrimeTime. Supports analysis of multiple Vdd/Vss designs.
- Signal integrity analysis only mode - uses ideal voltages to perform effective coupling analysis between key aggressor and victim nets-- signal and/or clock nets that have slew, delay and slack adversely affected by capacitive coupling. No dynamic voltage drop analysis is necessary.
- Power integrity analysis only mode - evaluates the adverse effects of dynamic voltage drop on slew, delay and slack for signal and/or clock nets. No capacitive coupling analysis is performed.
- Simultaneous power and signal integrity analysis mode - evaluates the adverse effects on signal and/or clock nets of simultaneous impacts of both capacitive coupling and dynamic voltage drop.

For clock tree jitter analysis, power integrity analysis is always considered. You can choose to also consider signal integrity analysis with the appropriate setup conditions.

Timing Analysis Applications

Accurate timing analysis is useful in a nanometer scale IC design flow, where a successful silicon design depends upon catching problems related to power and timing before manufacturing. This is particularly valuable when a design faces the challenges of dynamic voltage drop, ground bounce and cross-talk noise. The following are typical applications that can be very useful.

False Violation Filter

Many design houses use a cell-based methodology to verify timing and signal integrity. The setup time check is vital to final silicon performance. Furthermore, the hold time and peak noise checks find potential errors that can cause the chip to fail functionally. Cell libraries are built to check each cell type, but are not accurate enough for each cell instance. Characterization of standard cell libraries is deliberately conservative. The conservativeness of cell library methodology may create many unwanted violations. These false violations add an extra burden on the designers. SPICE-accurate results can serve as a golden reference for identifying real violations. As a result, the designers can focus on fixing the real problems.

Design Margin Adjustment

A design specification usually requires that additional guard-band be added to account for the inaccuracies caused by either timing/signal integrity tools or model abstraction. With timing analysis, the designer may be able to reduce or eliminate the guard band requirements. The designer can rely on the SPICE accuracy as the tape-out reference.

Methodology Calibration

Some design houses use cell-based tools for low turn-around time and transistor-level tools for accuracy. Because **RedHawk** is a SPICE-based tool, it can pinpoint the sources of inaccuracy from cell modeling and library characterization and improve the overall accuracy of the design flow. In this way, even though the design project still uses a cell-

based design flow, the transistor-level accuracy will guide the design into silicon convergence.

Case Analysis for Stress Test

Occasionally, the designer wants to ensure the reliability of the design by running stress tests. Their attention usually focuses on critical design spots such as bus design, I/O pads, long wire nets, and clock nets. This tool is for discovering and analyzing the effects of coupled delay and noise on vulnerable signal nets, with voltage drop and ground bounce on power/ground nets.

Silicon Correlation

The designer can test different process corner scenarios, supply voltages, and temperatures to correlate the design with real silicon. A good correlation will help the designer or manufacturer to optimize yields under various process conditions. After refinement of the critical paths, the design can be fine-tuned for volume production with good manufacturing yields.

LEF-SPICE Pin Mapping

Designers can map LEF pin names to Spice pin names if they are different, using the timing analysis configuration file keyword LEF2SPICE_PIN_MAPPING, which provides mapping between LEF pin names and SPICE pin names. The syntax is as follows:

```
LEF2SPICE_PIN_MAPPING {  
    <LEF_pin_name> <Spice pin_name>  
    ...  
}
```

PJX Fullchip Clock Tree Jitter Analysis

Overview

PJX-full-chip is a fast, full-chip analysis tool, which provides jitter-related voltage drop results in the context of impact on chip timing.

PJX calculates voltage-only induced jitter in order to identify grid defects/weaknesses that cause jitter. The focus is on voltage induction and not SI (Signal Integrity) effects.

PJX is an STA-based tool for preliminary high level analysis and not simulation. It is not a replacement for SPICE-based jitter calculation that includes PI, SI and other factors causing timing jitter.

Flow and Interface Description

PJX-full-chip analysis performs N cycles of timing computation using the STA-based built-in multi-threaded Apache Timing Engine (ATE). In each cycle it uses RedHawk per instance cycle voltage and APL models to compute voltage derated delays. It then performs timing analysis to calculate clock arrival time at leaf pins (network endpoints) for all clocks. Jitter at clock endpoints is based on the differences among calculated N arrival times.

PJX-full-chip offers a GUI to analyze timing impact result for jittery clocks, endpoints, path-traces, bottleneck regions and instances .

The PJX-full-chip GUI is integrated into the RedHawk GUI to allow overlaying jittery path traces (as flylines) on various RedHawk colormaps, to assist in diagnosis through visual

analysis of grid weaknesses that can cause jitter, such as low decaps values around jittery clock elements, or portions of traces.

Input Data Preparation and Setup

The following **RedHawk** inputs and settings ensure that PJX-full-chip analysis has sufficient accuracy and coverage :

- APL files should be specified in the GSR file, and should be complete so as to cover all clock network cells. PJX uses APL info for cell delay derating.
- RedHawk DvD analysis should be performed in multi-cycle mode, so simulation time should be sufficiently large to let each clock run for multiple cycles. Specify the option “-mcycle” with the “perform analysis” command for vectorless dynamic analysis. No such option is required for VCD-based dynamic analysis or for static analysis.
- The STA timing file used in **RedHawk** should be consistent with the DEF netlist and SDC file. If the STA file comes from an external source that uses a Verilog netlist, you must ensure the consistency of that Verilog netlist. Otherwise, set the GSR keyword ‘ENABLE_ATE 1’ to allow RedHawk to generate its STA (timing) file using ATE, which is consistent with the DEF and SDC data that is being used for RedHawk and PJX.
- Specify the dynamic simulation time range that experiences jitter.
- Specify enough presim time to avoid transient disturbances in the initial cycles to appear as a cause of jitter.
- PJX requires a redhawk_pjx license.
- Specify required SDC files using the following GSR keyword :

```
ATE_CONSTRAINT_FILES {  
    <SDC_file1>  
    <SDC_file2>  
    ...  
}
```

Note: the SDC files should be consistent with the DEF netlist specified in the GSR, otherwise mismatch-related errors are reported in the “Load Timing Constraint” section in the *fullchipjitter.log* file, which can impact PJX-full-chip results.

- If the imported **RedHawk** DB lacks an SDC specification, then SDC files can be specified on-the-fly on the **RedHawk** command line before launching PJX, as follows :

```
gsr set ATE_CONSTRAINT_FILES { /path/to/sdc/file.sdc }
```

Running PJX Fullchip Jitter Analysis

PJX full-chip analysis can be launched in the **RedHawk** GUI by clicking :

Tools / Clock Jitter / Fullchip Jitter / Analyze

Or you can add the command “perform jitter -fullchip” in the **RedHawk** command file after completion of multi-cycle DvD Analysis or Static Analysis.

Analyzing Results

All PJX full-chip results can be viewed in the PJX GUI, which is integrated with the **RedHawk** GUI under the menu command **Tools -> Clock Jitter ->Fullchip Jitter**. The

integrated GUI allows display of fly-lines showing worst jitter path traces overlaid on relevant RedHawk colormaps, such as DvD and decap, to give a global perspective and visual orientation of key jitter-related issues over the whole chip.

Click on **Tools -> Clock Jitter ->Fullchip Jitter ->Long Term Jitter Report** to display all clocks, rank ordered on the basis of worst long-term jitter (% of period), observed in their respective domains. Long Term Jitter at a clock destination pin (end-point) is defined as the difference between the earliest-arrival-time and latest-arrival-time among the recorded arrival times of N cycles.

In the Long Term Jitter Report, for a selected entry the following results can be displayed:

Show Summary : brings up the summary report of the worst 1000 endpoints (leaf pins) of the selected clock. In this report, **Show Browser** displays the schematic connectivity from clock sources to the selected endpoint.

“Highlight” in the “Show Browser” report displays the flyline of the path trace to the selected endpoint. “Show Trace” shows the path trace from source to endpoint with all relevant details, such as the earliest and latest cycles, cycles voltages for instances on the path, absolute jitter, and jitter as a percent of period.

Show Detail : brings up the detail report with jitter information for all elements in the entire network of the selected clock. You can use the search box to get the elements of interest and other features such as “Show Browser”, and highlighting and zooming instances.

Show Browser : brings up a schematic view of the entire network of selected clocks. Clicking **Highlight** on this report displays the entire network as flylines in the RedHawk GUI.

You can use various **RedHawk** maps and other GUI commands to diagnose grid defects and weaknesses that caused each path-trace (flyline) to become a worst jitter victim.

Similarly, **Tools -> Clock Jitter ->Fullchip Jitter -> Long Term Cycle-2-Cycle Jitter Report** displays results in similar format, of jitter type “Long-Term Cycle-2-Cycle”, Long Term Cycle-2-Cycle Jitter at a destination pin is defined as the worst (max) jitter among the differences of arrival times of consecutive cycles.

The file *adsRpt/fullchipjitter.log* records the PJX-full-chip analysis. The errors and warnings recorded from PJX-full-chip analysis are captured in files *adsRpt/fullchipjitter.err* and *adsRpt/fullchipjitter.warn* respectively.

PJX Clock Tree Jitter Sign-Off Analysis

Overview

After successfully running DvD analysis in **RedHawk**, you can perform clock tree jitter analysis.

Required Inputs for Clock Tree Jitter Sign-Off Analysis

The specific input data files that are needed are listed below, along with their purpose and the operations performed on them.

1. SPICE cell netlist file (CIR) for each gate in the cell library
 - Defines voltage levels for I/O pad cells.

Jitter analysis matches the SPICE cell netlists to cell instances and connects instance power/ground ports to local voltage sources.

2. SPICE technology library file (SPICE format)
 - Defines transistor models
 - Defines process corners for SPICE simulation
 - Defines On-chip process variation (OCV) for Spice simulation.
 - Specifies process/voltage/temperature (PVT) values for process corner case simulation
3. Additional timing constraints, using the configuration file keyword **DEFINE_DESIGN_CONSTRAINTS** ([section "DEFINE_DESIGN_CONSTRAINTS", page 8-209](#)).

Input Data Preparation

The following sections describe the preparation steps needed for each type of required input prior to running an analysis.

Spice Cell Netlist (.CIR) File

RedHawk reads the Spice netlist (defined in the **SPICE_NETLIST** configuration file keyword) of each cell from a specified file for later Spice simulation. The Spice cell file should contain all necessary cells under analysis. You can also use the ‘.include’ directive to include all related files, but only the full file path name is accepted. For example:

```
.include '/projects/AABB15/src/abcd18_3.cir'  
.include '/projects/AABB15/src/abcd18io_4.cir'
```

Each cell found in the Spice netlist should have a corresponding Spice sub-circuit netlist to describe its electrical model in detail. Sub-circuit calls—that is, X-statements—inside the cell sub-circuit construct are allowed. The cell netlist is complete as long as all sub-circuit calls can be found.

When **RedHawk** matches cell sub-circuits into netlist cell instantiations, the sub-circuit ports will connect to the nets in the design by name. In some cases, power and ground are present as cell sub-circuit ports. You may want to keep the same name for this kind of global power and ground nets for simulation. However, if the given subcircuit port is called ‘gnd’, then execute the following to rename it (since GND is the global virtual ground node defined in Spice language):

```
SPICE_CELL_TOKEN_RENAME gnd vss
```

The consistency of sub-circuit ports and design cell port instantiations is checked. If there are any mismatches, a warning is issued in the run log.

SPICE Technology Library Data

Spice technology library data for Spice device models are included by using the **DEVICE_MODEL_LIBRARY** keyword in the jitter configuration file.

A few examples of how to specify Spice technology library paths follow:

```
INCLUDE /usr/cadtool/technology/logic018param.1  
DEVICE_MODEL_LIBRARY /usr/cadtool/technology/logic018io.1 ss  
DEVICE_MODEL_LIBRARY '/projects/BCM7315B0/technology/logic018.1' ss
```

The last statement above uses single quotations to preserve the upper-case characters in the names.

Additional Timing Constraints

Additional constraints may be needed for the information missing in the input data. Specify design constraints in the `DEFINE_DESIGN_CONSTRAINTS` keyword section of the timing configuration file ([section "DEFINE_DESIGN_CONSTRAINTS", page 8-209](#)). For the certain special cases you need to set constraint inputs as follows:

1. If external loading is missing in the given parasitic SPF file. For example, the output node of the critical path is a chip/block boundary port. Add the loading information to the configuration file, as follows:

```
set_load 4000 "sdram_dqm[7]"
set_load 3500 "sdram_cke"
```

The additional loading is specified in femtoFarads (ff).

2. If net resistance is missing in the given parasitic SPF file. Add the net resistance information (Ohms) and port or net name in the configuration file as follows:

```
set_resistance 100 "net_clock"
```

3. If you want to control the input state of instances, then add the input logic state setting, as in the following example:

```
set_logic_one/ set_logic_zero "mux_1/sel"
```

The logic state of the specified port is set to one/zero when performing vector sensitization.

4. If you want to analyze the clock tree through a divider circuit, then add the clock divider pin name in the constraint, as in the following example:

```
set_clock_divider div_reg:q
```

5. If you want to stop clock tree/jitter analysis at specified pins, then add the selected clock leaf pins in the constraints, as in the following example:

```
set_clock_leaf clk_mux:i0
```

6. If you want to exclude a subset of a tree below a specified pin in clock tree or jitter analysis, add the excluded pin in the constraints, as in the following example:

```
exclude_clock_leaf clk_reg:d
```

Clock Tree Jitter Analysis Setup Procedure

Use the following steps to set up and run clock tree jitter analysis:

1. RedHawk Data Preparation

a. Most required files should already have been prepared for DvD analysis.

b. **Prepare and use package netlist** or package lumped RLC model in analysis.

c. **Set up GSR keywords** as described below:

- **Jitter awareness enable**

```
JITTER_ENABLE 1
```

This is a required keyword to make **RedHawk** flow “jitter-aware”. Dynamic simulation is optimized to extract only the clock instance waveform.

- **Cycle selection**

```
CYCLE_SELECTION [ WORST_JITTER_CYCLE | WORST_PERIOD_JITTER_CYCLE
                  | WORST_MIN_PERIOD_JITTER_CYCLE | WORST_MAX_PERIOD_JITTER_CYCLE
                  | WORST_C2C_JITTER_CYCLE ]
```

where

WORST_JITTER_CYCLE : selects cycles for both worst period jitter and cycle-to-cycle jitter analysis

- WORST_PERIOD_JITTER_CYCLE: selects cycles for worst period jitter analysis
- WORST_MIN_PERIOD_JITTER_CYCLE: selects cycles for jitter analysis to get minimum period
- WORST_MAX_PERIOD_JITTER_CYCLE: selects cycles for jitter analysis to get maximum period
- WORST_C2C_JITTER_CYCLE: selects cycles for worst case cycle-to-cycle jitter analysis

- **Using the power histogram to select the most critical jitter cycles**

```
IGNORE_JITTER_FASTSIM 1
```

The default is 0/Off, meaning “do not ignore simulation-based cycle selection”.

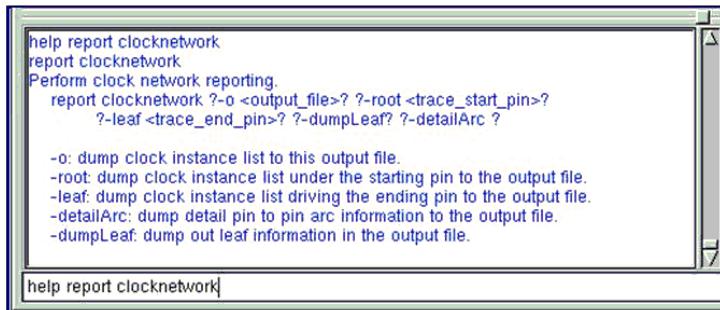
- d. In **RedHawk** analysis, provide a SPEF file with coupling capacitance data in order to perform cross-talk analysis.

2. Selecting Clock Roots

After **RedHawk** dynamic analysis, use the following **RedHawk** TCL command to generate the clock network summary.

```
report clocknetwork -o <output_filename>
```

The summary includes clock tree instance and leaf information on a clock domain basis, and also a summary table in the log file. This also shows clock root, domain frequency, buffer/gate count, leaf count, maximum level, minimum level, and effective Vdd information for each clock domain. Command syntax help can be obtained by requesting it on the command line, as shown in Figure 8-4.



Sample Command:

```
report clocknetwork -o clk.network.out -root <instance_name>:<output_pin>
  -leaf <instance_name>/<subcell_inst>:<output_pin>
```

Figure 8-4 Log window and clocknetwork command help

The clock network summary can also be invoked using the menu command **Tools > Clock Jitter > Clock Network Summary**

as shown in Figure 8-5.

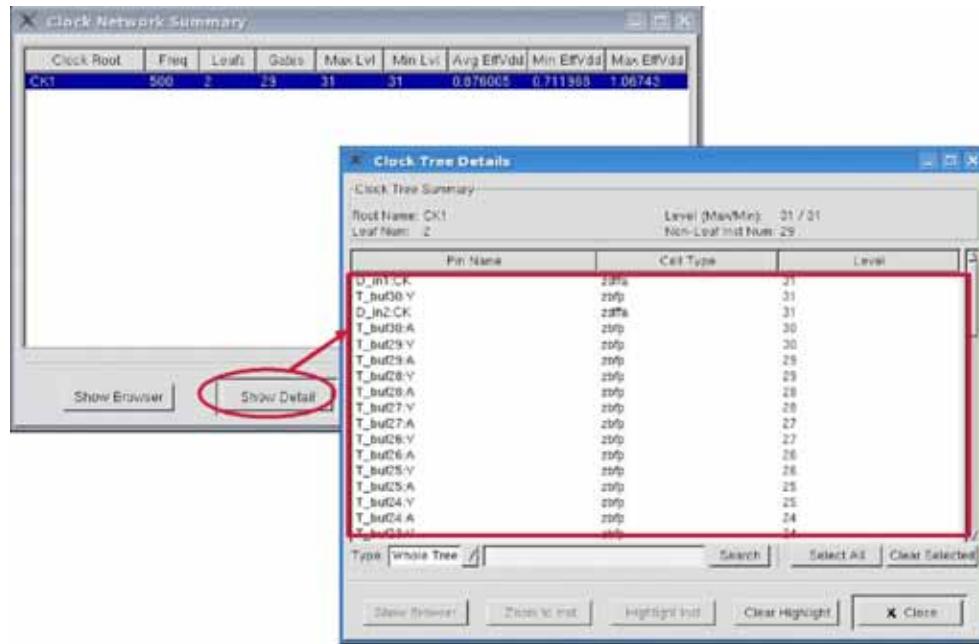


Figure 8-5 Clock Network Summary

Based on the clock network summary report, you may select which clock roots should be used for jitter analysis purposes, depending on the DvD values for each clock root, as shown in Figure 8-6.

Selecting a candidate clock for analysis that has:

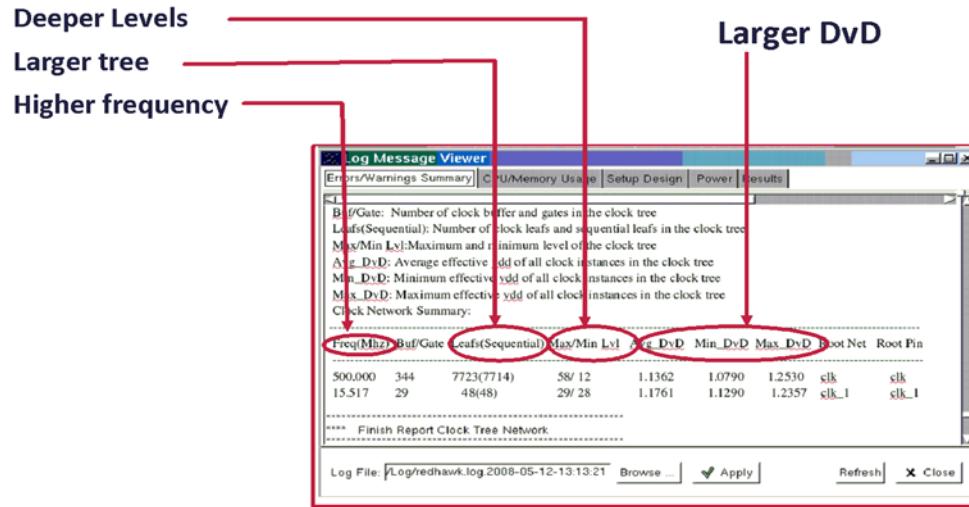


Figure 8-6 Clock root selection dialog

- Jitter analysis preparation. Set up the configuration file, to describe the Spice cell netlist, Spice technology data, additional timing constraints, and other variables. The timing configuration file must have the following keywords:

- SPICE_NETLIST/SPICE_SUBCKT_DIR
- DEVICE_MODEL_LIBRARY/INCLUDE

- **CLOCK_SOURCES** (if not specified from the TCL command line)

Other optional keywords that can be used to optimize jitter analysis performance, with little reduction in accuracy, are:

- **CLOCK_TREE_MAX_LEVEL**
- **DEFINE_CASE_ANALYSIS**

When set, reduces signal crosstalk pessimism (make less likely to fail) using options 'pgvolt', 'clock_jitter_worst' and 'clock_jitter_best' (default PGVOLT).

- **EXTEND_CYCLE_SELECTION [-1 | 0 | 1]**

Manages cycle selection for sets of non-consecutive cycles, as follows:

-1 : removes non-consecutive cycles

0 : keeps all selected cycles, consecutive or non-consecutive

1 : adds more cycles to make cycles consecutive

For example, if the specified cycles are 10 15 16 17,

- setting 'EXTEND_CYCLE_SELECTION -1' means cycle 10 is discarded,
- setting 'EXTEND_CYCLE_SELECTION 1' means cycles 10 11 12 or 9 10 11 are added, based on worst-best-worst/best-worst-best ranking pattern
- setting 'EXTEND_CYCLE_SELECTION 0' retains cycles as specified (10 15 16 17)

- **FALL_SLEW <value>**
- **FALL_SLEW_AGRS <value>**
- **RISE_SLEW <value>**
- **RISE_SLEW_AGRS <value>**

The units for the *_SLEW transition time keywords are in ns, so input values should be greater than 0 and less than or equal to 1.0 ns. If input data is out of this range, such as a value in ps, an error message is displayed and processing stops. The default value for all is 0.05 ns.

- **JITTER_FAST_MODE**
- **JITTER_CUTOFF_FREQ**
- **LEF2SPICE_PIN_MAPPING**

You can map LEF pin names to Spice pin names if they are different using this keyword, with the syntax:

```
LEF2SPICE_PIN_MAPPING {
    <LEF_pin_name> <Spice_pin_name>
    ...
}
```

- **SETTLE_TIME**
- **SETTLE_TIME_RATIO**

Sets the value of SETTLE_TIME based on the specified clock period ratio.

For example, if SETTLE_TIME_RATIO is set to 0.5, a SETTLE_TIME of '0.5*<clock_period>' is used. This is useful in cases in which multiple clock frequencies are present in the design. The default value is 0.75.

The syntax is:

```
SETTLE_TIME_RATIO <value>
```

- **STA_SWITCH_TIME_READ [0 | 1]**

If there are switch times missing from the **RedHawk** DvD waveforms, they are obtained from the STA file if this keyword is set.

- **TIMING_WINDOW_OVERLAP_RATIO_THRESHOLD <overlap_factor>**

Controls the selection of aggressor nets based on the Timing Window overlap ratio, which is the fraction of the TW overlap (0.0-1.0).

- **DEFINE DESIGN CONSTRAINTS**

For setting other design constraints, such as defining a divider FF Q-pin as 'set_clock_divider <Q_pin>'.

For a more descriptions of timing configuration file keywords, please refer to [section "Clock Tree Analysis Configuration File Reference", page 8-201](#).

4. **TW filtering for Jitter Analysis.** For jitter analysis with signal integrity, you can control the aggressor selection for coupling based on TW overlap threshold. By default all aggressors having any TW overlap with a victim are selected for coupling, which may increase runtime. The timing configuration file keyword TIMING_WINDOW_OVERLAP_RATIO_THRESHOLD is available for filtering coupling calculations based on the amount of timing window overlap of aggressor and victim. The syntax is:

```
TIMING_WINDOW_OVERLAP_RATIO_THRESHOLD <value>
```

For more details, see [section "TIMING_WINDOW_OVERLAP_RATIO_THRESHOLD", page 8-204](#).

Edge-to-Edge DDR Clock Jitter Measurements

Edge-to-edge jitter is the variation in the delay between a triggering event and a response event. Edge-to-edge jitter assumes an input signal, and so is only defined for driven systems. It is an input-referred jitter metric, meaning that the jitter measurement is referenced to a point on a noise-free input signal, so the reference point is fixed.

Write Mode

For every rise/fall edge of the Reference signal (Clock), the Setup and Hold measurements are performed to the closest rising or falling edge of the Data signal, as shown in Figure 8-7. Setup and Hold measurements should be made for every rise/fall edge of the clock signal, regardless of whether the Data signal rising or falling edge belongs to another bit cycle. The configuration file syntax is:

```
DDR_DIFFERENTIAL_JITTER
{
    WRITE_PATH ? -idealBitPeriod <bit_period> ?
        -clock <clock_pin> -data {<data_pin1> <data_pin2> ... }
}
```

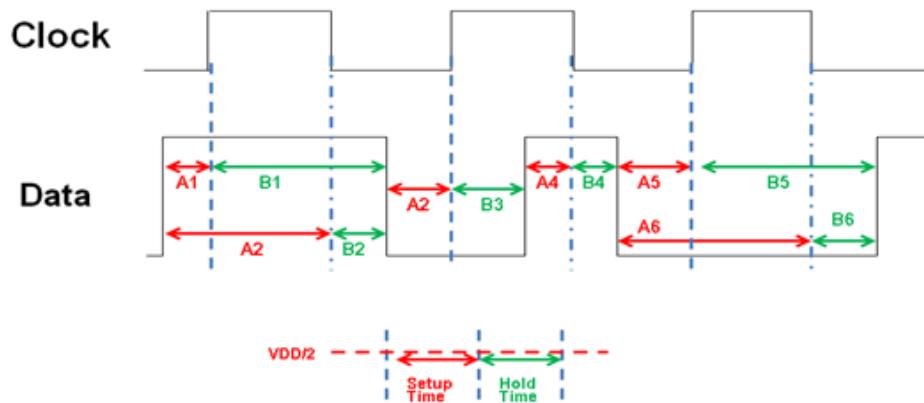


Figure 8-7 Edge-to-edge jitter - Write Mode

Read Mode

For each Clock signal, only the rising edge is utilized to capture the Setup/Hold timing measurements relative to the closest corresponding Data signal rising or falling edge. The diagram below (Figure 8-8) illustrates the end point timing. The configuration file syntax is:

```
DDR_DIFFERENTIAL_JITTER
{
    READ_PATH ? -idealBitPeriod <bit_period> ?
    -clock <clock_pin> -data {<data_pin1> <data_pin2> ... }
}
```

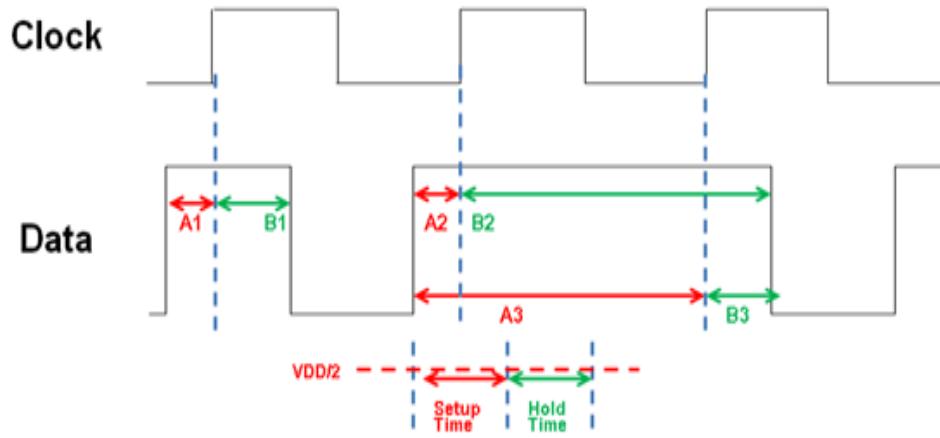


Figure 8-8 Edge-to-edge jitter - Read Mode

Running Clock Tree Jitter Sign-Off Analysis from the RedHawk GUI

After completing the **RedHawk** data setup and creating a timing configuration file, the Clock Tree Jitter Analysis flow can be invoked using the **RedHawk Tools** menu.

If DvD analysis has not already been performed, clock tree jitter commands on the **Tools** menu can be run in the order in which they appear, as follows:

Tools -> Clock Jitter -> Clock Network Summary

Dynamic -> Power -> Calculate/ Import

Network Extraction

Pad, Wirebond and Package Constraints

RedHawk performs either vectorless or VCD-based jitter cycle selection, based on the STATE_PROPAGATION or VCD_FILE GSR options specified in the GSR.

Clock Jitter Analysis

If DvD analysis has already been performed, then just use this command. RedHawk runs either vectorless or VCD True Time mode dynamic simulation. Then, multiple-cycle power ground waveforms are extracted and multiple-cycle Spice simulation is performed to obtain clock tree jitter results.

Clock Jitter Report

Use this command to check and debug the jitter results and waveforms through the GUI interface.

Clock Jitter Bottleneck Report

Use the jitter bottleneck ranking report to determine the worst jitter weakness locations. Fixing jitter on pins with highest rankings maximizes the benefit in reducing the overall jitter problem.

Running Clock Tree Jitter Sign-off Analysis in Batch Mode

Before performing clock tree jitter analysis in batch mode, do the same setup procedure as for the GUI invocation, as described previously.

Setup for Batch Mode Invocation

If DvD analysis has not already been performed, for power Integrity only, or for simultaneous signal integrity and power integrity modes, run RedHawk through the package setup stage, or load a database saved before dynamic analysis. The set of TCL commands required to set up a database for power integrity mode is given in the following example:

```
import gsr design_dynamic.gsr
setup design
import apl cell.current
import apl -c cell.cdev
perform pwrcalc
perform extraction -power -ground -c
setup pad -power -r 0.010000 -c 0.5
setup pad -ground -r 0.010000 -c 0.5
setup wirebond -power -r 0.245000 -l 1420 -c 5
setup wirebond -ground -r 0.24500 -l 1420 -c 5
setup package -r 0.002 -l 100 -c 5
```

Jitter Analysis Command Line Invocation

Following is the method of running clock tree jitter analysis in batch mode, relying on having proper keyword definitions in the timing configuration file. Run through RedHawk dynamic analysis and then perform clock tree jitter analysis:

```
[perform jitter_cycle_select] #Optional to select cycles
perform analysis [options]
perform jitter [options]
show jitter [options]
```

The full syntax for invoking clock tree jitter is as follows:

```
perform jitter -config <psi_jitter_config> [-clk_source
<clk_src>]
[-leaf_inst_pin <leaf_inst_pin>][-no_data_check ]
[-mode [ waveform_pg | effVDD | eff_pg ]]
```

where

waveform_pg: jitter analysis based on the power/ground voltage waveforms from **RedHawk** analysis

effVDD: jitter analysis based on the multi-cycle effective Vdd reported by **RedHawk** analysis

eff_pg: jitter analysis based on the effective power/ground voltages calculated

In batch mode, you can only specify a single 'clk_src' and 'leaf_inst_pin'.

A sample TCL invocation of clock tree jitter analysis is:

```
setup design test.gsr
perform pwrcalc
perform extraction -power -ground
perform jitter_cycle_select -type WORST_JITTER_CYCLE -vcd
perform analysis -vcd
perform jitter -config psi.jitter_config
show jitter c2c rise
```

Also, 'perform jitter' analysis can be run on a post-dynamic **RedHawk** database, by using the following commands:

```
import db post_dynamic.db
perform jitter -config
...
```

After 'perform jitter' has been successfully completed, you can export the entire database (including jitter results) with the following command:

```
export db post_jitter.db
```

This exported database can be imported to view jitter results, as follows:

```
import db post_jitter.db
```

Specifying clock instances

To specify the instances to be simulated, you must first import the DvD analysis database, then perform the following steps:

1. Run the TCL command

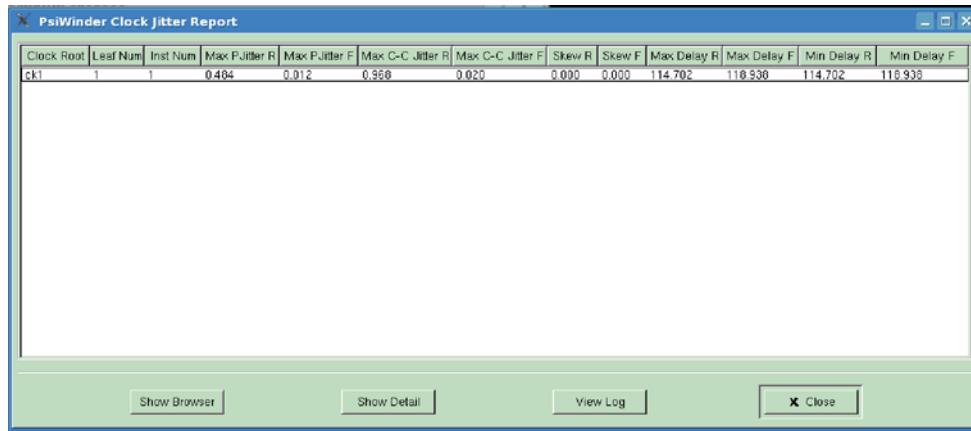

```
report clocknetwork -root <root_name> -o <clk_out>
```
2. Then manually reduce the clock buffers by editing the <clk_out> file (do not add new instances).
3. Specify the edited instance file in the configuration file keyword CLK_INST_FILE <edited_clk_out>, and ensure that the CLOCK_SOURCES <clock> keyword specification is the same as the <root_name> specified in the 'report clocknetwork' command.
4. Run the TCL command 'perform jitter -config psi.config', and then 'perform jitter'.

Clock Tree Jitter Results

A number of types of graphic displays and output data of jitter results are produced, as described in the following sections.

Clock Tree Jitter Report

If you perform clock jitter analysis, the command **Tools -> Clock Jitter -> Sign-off Jitter -> Clock Jitter Report** creates a clock tree summary table of period jitter data, as well as cycle-to-cycle jitter data. A sample Clock Jitter Report is shown in Figure 8-9.



Clock Root	Leaf Num	Inst Num	Max P Jitter R	Max P Jitter F	Max C-C Jitter R	Max C-C Jitter F	Skew R	Skew F	Max Delay R	Max Delay F	Min Delay R	Min Delay F
CK1	1	1	0.404	0.012	0.968	0.020	0.000	0.000	114.702	110.930	114.702	110.930

Figure 8-9 Clock Tree Jitter Report

The data displayed in the Clock Jitter Report columns are as follows:

- Clock Root - clock tree root pin name
- Leaf Num - number of terminating leafs in the clock tree
- Inst Num - number of instances in the clock tree
- Max P Jitter R - maximum rise period jitter among all leafs
- Max P Jitter F - maximum fall period jitter among all leafs
- Max C-C Jitter R - maximum rise cycle-to-cycle jitter among all leafs
- Max C-C Jitter F - maximum fall cycle-to-cycle jitter among all leafs
- Skew R - difference in longest and shortest rising insertion delay (ps)
- Skew F - difference in longest and shortest falling insertion delay (ps)
- Max Delay R - maximum rising delay in the clock tree, root to leaf (ps)
- Max Delay F - maximum falling delay in the clock tree, root to leaf (ps)
- Min Delay R - minimum rising delay in clock tree, root to leaf (ps)
- Min Delay F - minimum falling delay in clock tree, root to leaf (ps)

Clock Tree Browser Display

To display the Clock Tree Browser, click on **Tools->Clock Jitter->Clock Jitter Report**. The ‘Clock Tree Report’ dialog box is displayed. Now click on the **Show Browser** button to see the ‘Clock Tree Browser’ dialog box (see Figure 8-10).

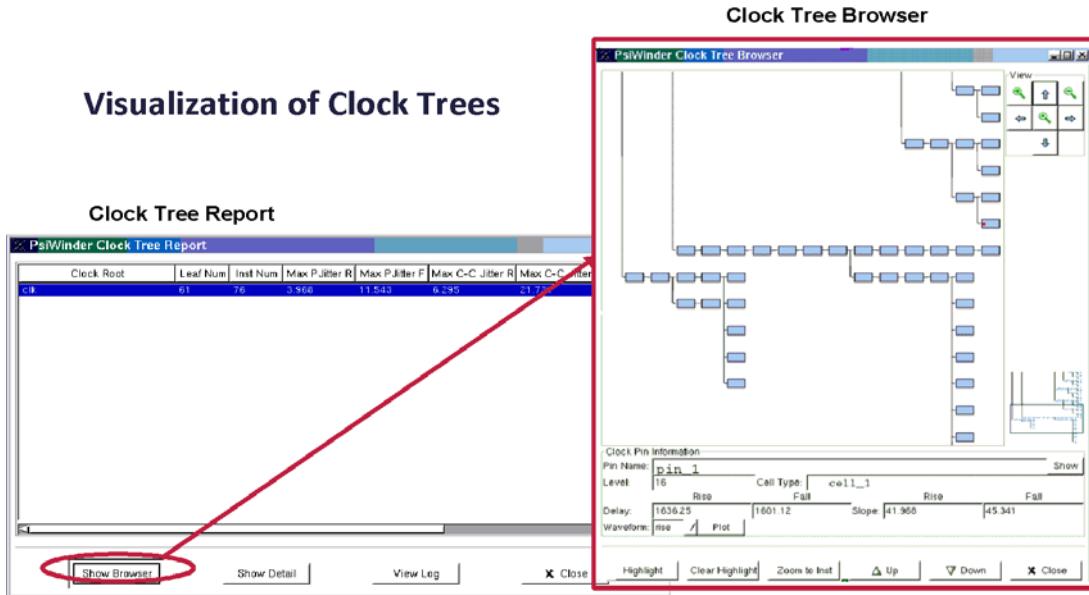


Figure 8-10 Clock tree browser display

A maximum of 500 paths can be displayed in the ‘Clock Tree Browser’. Information and functions on the clock tree schematic viewer is as follows:

- Clock Path Symbol Summary: displays symbols for the Interconnect path (timing arc between cells), IO path (timing arc inside cell), Highlighted pin (pin selected to obtain pin information), Common branch (branch common to selected pins), and pins connecting Selected pin or pins.
- Clock Pin Information: clicking any node on the schematic clock tree path browser brings up detailed clock pin information, such as Pin Name, Level, Cell Type, Delay Rise and Fall times, and Slope (transition) Rise and Fall times (ps).
- Highlight button: highlights the entire selected clock path from root to the selected pins in the GUI design layout window. If multiple pins are selected, the common path will be highlighted in yellow. See Figure 8-11.

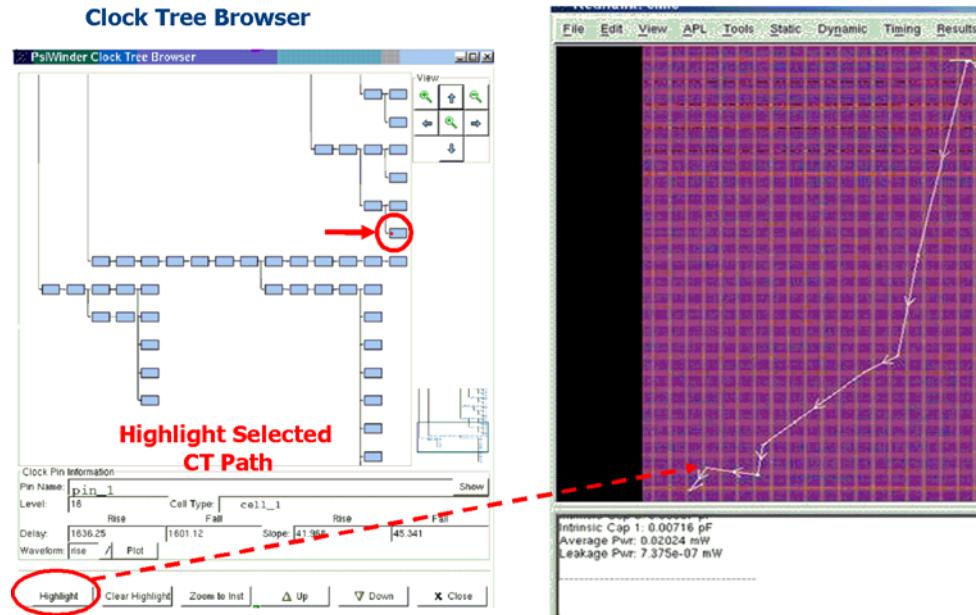


Figure 8-11 Highlighted clock network

- Clear Highlight button: clears highlighted clock paths.
- Zoom to Instance button: highlights and zooms to the selected instance in the GUI layout window. See Figure 8-12.

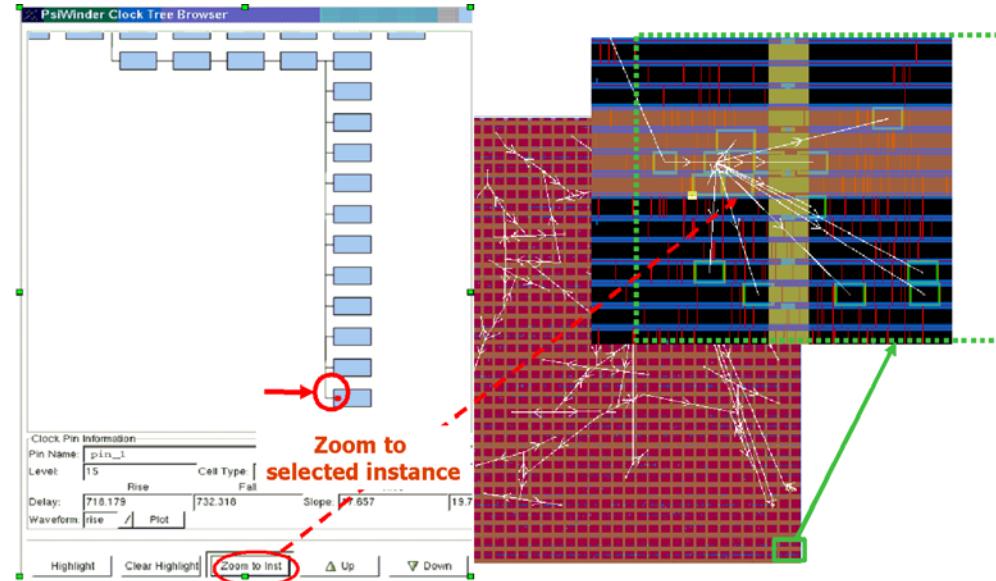


Figure 8-12 Zoom to clock instance

- Up button: selects and highlights the next upstream pin in the path displayed in the schematic viewer. The clock pin information changes accordingly.
- Down button: selects and highlights the next downstream pin along the path in the schematic viewer. The clock pin information changes accordingly.

- Close button: closes the tree browser.

Clock Tree Jitter Details Report

To obtain additional details on the clock jitter analysis select a clock tree from the ‘Clock Jitter Report’ window and click on the ‘Show Detail’ button, which displays the ‘Clock Jitter Details’ window, with detailed results of clock jitter analysis by pin, as shown in Figure 8-13. All instances and pins in the clock tree are listed, and by selecting them they can be identified in the design using the ‘Zoom to Inst’ and ‘Highlight Inst’ buttons. You can sort the list by values in any column by clicking the column header. Default sort order is by rise transition Vdd voltage (‘VDrop’).

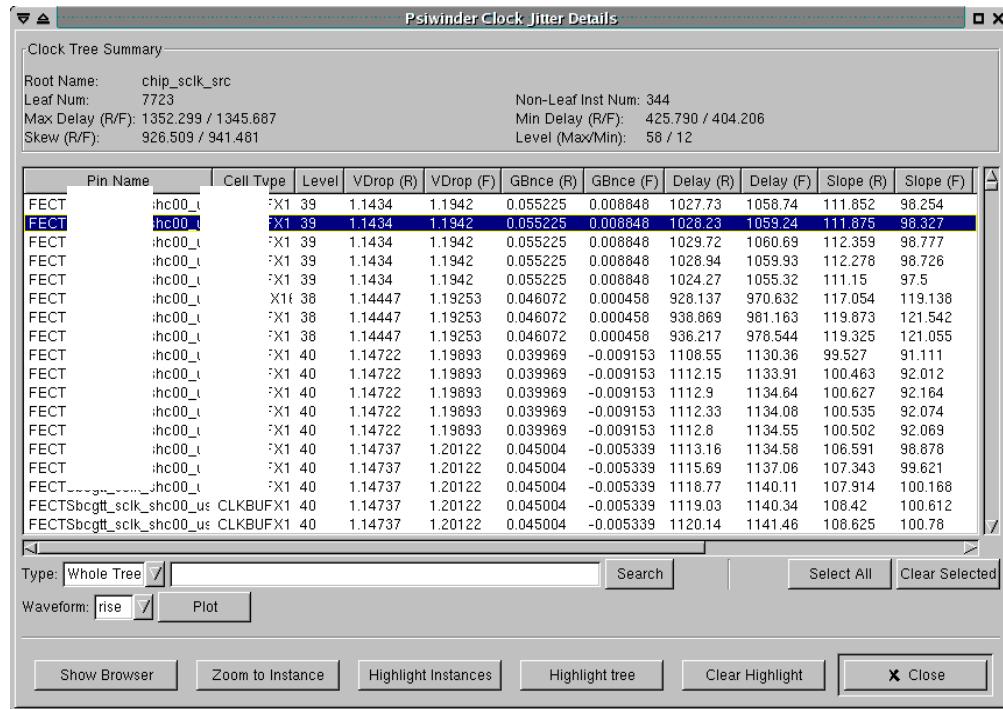


Figure 8-13 Clock Tree Jitter Details display

The data displayed in the Clock Jitter Details window columns, left to right, are:

- Pin Name - name of pin
- Cell Type - master cell name
- Level - number of stages below root
- VDrop (R) - rail voltage of instance at rising switching time (v)
- VDrop (F) - rail voltage of instance at falling switching time (v)
- GBnce (R) - ground voltage of instance at rising switching time (v)
- GBnce (F) - ground voltage of instance at falling switching time (v)
- Delay (R) - delay from root to pin (rising root transition) (ps)
- Delay (F) - delay from root to pin (falling root transition) (ps)
- Slope (R) - rising transition time at pin (ps)
- Slope (F) - falling transition time at pin (ps)
- P Jitter (R): period jitter at pin when root transition is rising

- P Jitter (F): period jitter at pin when root transition is rising
- C-C Jitter (R): cycle-to-cycle jitter at pin when root transition is rising
- C-C Jitter (F): cycle-to-cycle jitter at pin when root transition is falling

Features below the ‘Clock Jitter Details’ list assist in reducing and sorting items on the list:

- Type button: specifies a set of sorting categories for the list: Whole Tree, Pin Name, Cell Type, Level, Leaf, or Non-leaf. For anything but Whole Tree, Leaf and Non-leaf, a name fitting the Type selected must be typed into the adjacent box.
- Search button: allows you to sort the whole tree and display a smaller list of items
- Waveform selection button: selects ‘rising’ or ‘falling’ waveforms to plot
- Plot button: plots several waveforms associated with the instance selected. See description in the following section.
- Select All: selects all items in the list
- Clear Selected: clears all selected items in the list

Along the bottom of the ‘Clock Jitter Details’ dialog are additional function buttons:

- Show Browser button: select one or more pins in the Clock Tree Details list and click on the ‘Show Browser’ button to display the paths from root to the selected pins in a schematic viewer. This display is described in the section ["Clock Tree Browser Display", page 8-187](#).
- Zoom to Instance button: zooms GUI display to the instances selected
- Highlight Instances button: after selecting one or more pins on the list, click on the ‘Highlight Instances’ button to highlight the instances in the GUI.
- Highlight tree: highlights selected portion of clock tree in GUI display
- Clear Highlight button: clears all highlighted objects in the GUI

Edge-to-edge DDR Clock Tree Jitter Reports

Report 1: Edge-to-edge jitter report

After DDR clock jitter analysis, edge-to-edge jitter results are available in the file *adsRpt/Jitter/edge_to_edge.jitter*. The report summarizes the edge-to-edge relative jitter between the Clock and Data path timing analysis, as shown in the example format below. This measurement should be captured at the logical end points of the Clock and Data logical paths.

```
# CLOCK : <clock pin name>
# DATA_1 : <data pin name>
# DATA_2 : ...
# ...
# Ideal Bit Period : <ideal bit period>
```

CLOCK	DATA_1		DATA_2		
Rise Edge No	Setup Time (ps)	Hold Time (ps)	Setup Time (ps)	Hold Time (ps)	..
1
2
Min tDS/tDH(ps)	<min setup time>	<min hold time>	<min setup time>	<min hold time>	..
Edge-to-edge jitter (ps)	<jitter>		<jitter>		

Report 2: Periodic DDR clock jitter report

A quick summary of the period and cycle-to-cycle jitter is reported in the *adsRpt/Jitter/tree_*_pi/peak_to_peak.jitter* file for each instance along the path being measured. The instance pins are reported in sequential order from the beginning of the logical path to the end of the logical path. The report format is shown below:

Period Jitter (ps)		Cycle-to_cycle Jitter (ps)		Peak-to_peak Jitter (ps)		Pin Name
Max	Min	Max	Min	Period Jitter	Cycle-to-Cycle Jitter	..
Rise / Fall	Rise / Fall	Rise / Fall	Rise / Fall			..

Waveform Plots

To plot waveforms, after selecting a pin on the clock jitter list, click the **Plot** button in the ‘Clock Tree Browser’ dialog box and the waveform plot is displayed (see Figure 8-14). You can see multiple cycles of clock signal waveforms at the selected points. The difference in time between the earliest and latest transitions is the clock jitter.

PsiWinder – Plot Report

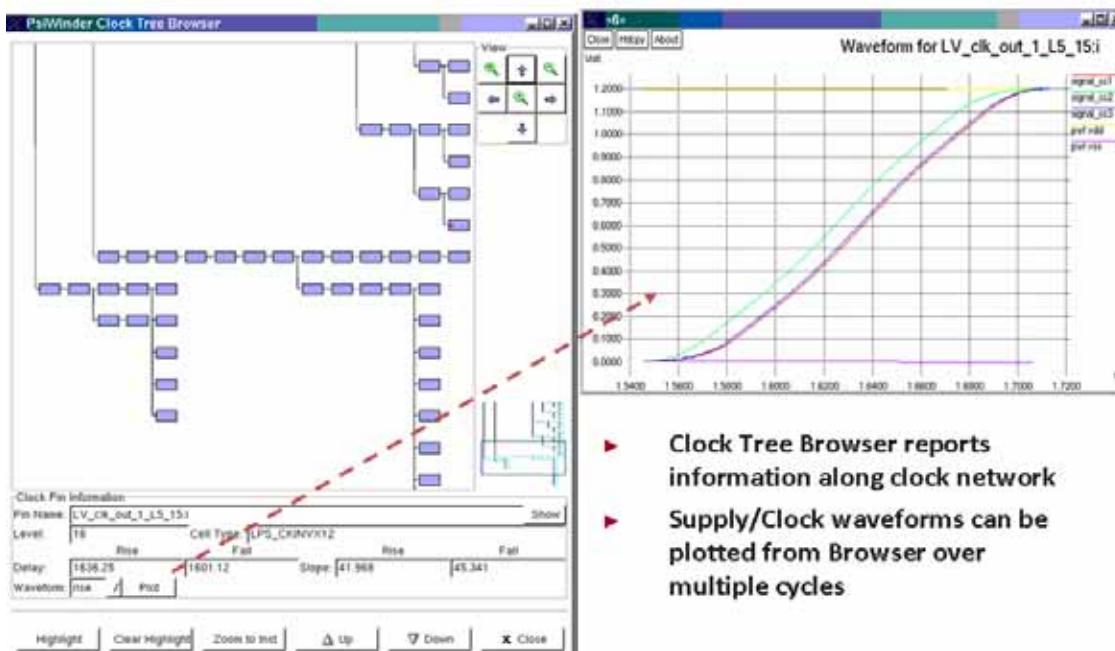


Figure 8-14 Plot waveform display

Examples of the rising and falling waveform jitter measurements are given in Figure 8-15 and Figure 8-16:

- signal waveforms for cycles 1, 2, 3
- power waveforms for cycles 1, 2, 3
- ground waveforms for cycles 1, 2, 3

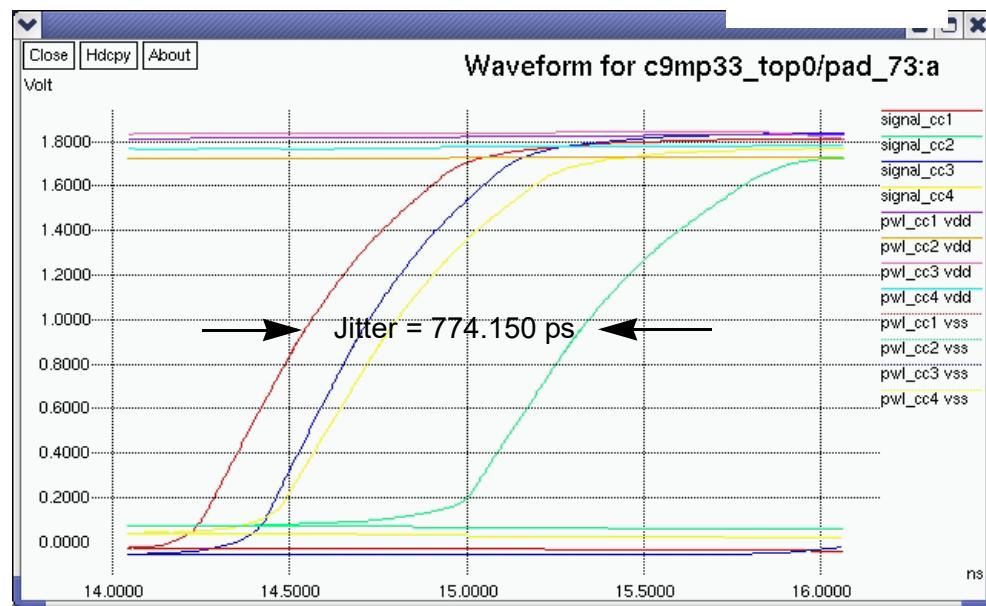


Figure 8-15 Jitter Rise Waveforms

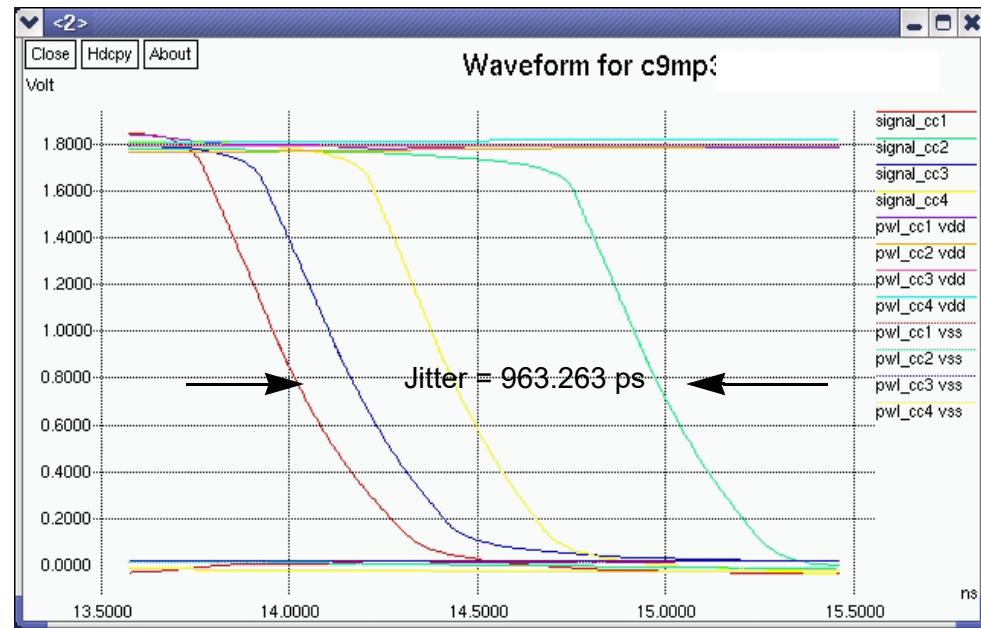


Figure 8-16 Jitter Fall Waveforms

Clock Jitter Bottleneck Report

Executing the menu command **Tools -> Clock Jitter -> Sign-off jitter -> Clock Jitter Bottleneck Report** creates a ranking table displaying the worst jitter weakness locations.

Fixing jitter on pins with the highest rankings maximizes the reduction in the overall jitter problem in the design. A sample Jitter Bottleneck Report is shown in Figure 8-18.

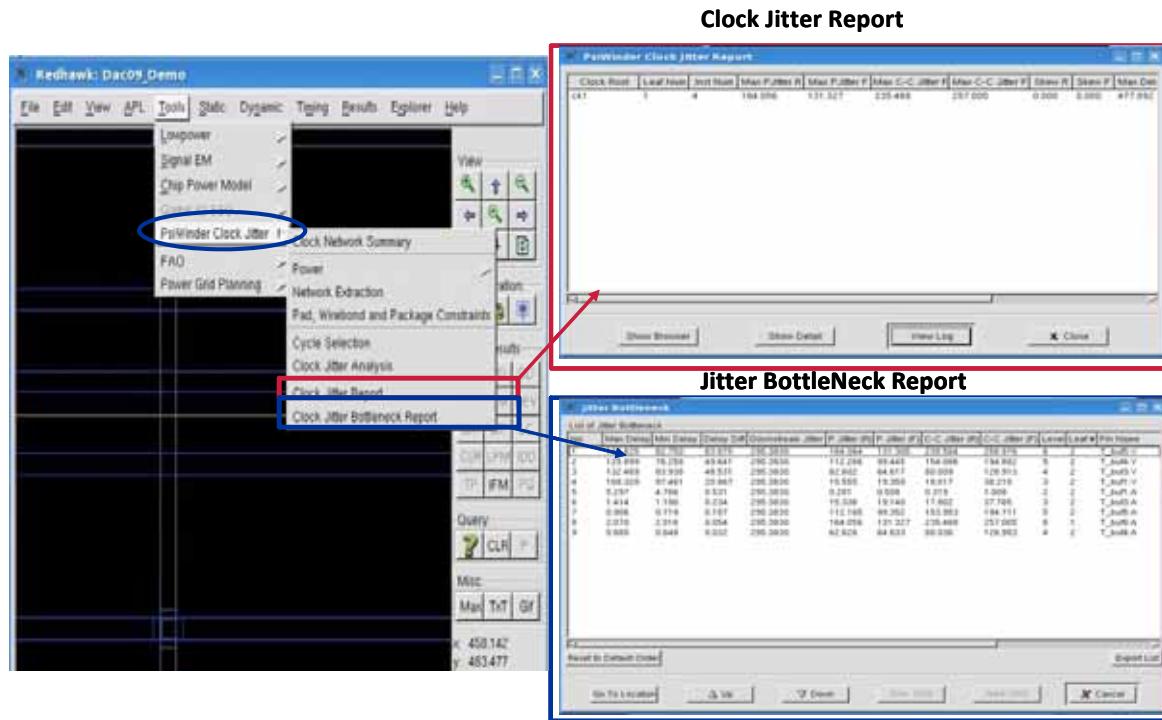


Figure 8-17 Jitter Bottleneck Report

- Clock tree jitter bottleneck report - Click on **Tools-> Clock Jitter-> Sign-off Jitter -> Clock Jitter Bottleneck Report**. The ‘Jitter Bottleneck Report’ dialog box is displayed. (see Figure 8-18).

Jitter Bottleneck											
List of Jitter Bottlenecks											
No.	Max Delay	Min Delay	Delay Diff	DS Leaf Jitter	P Jitter (R)	P Jitter (F)	C-C Jitter (R)	C-C Jitter (F)	Level	Leaf #	Pin Name
1	71.199	68.008	3.191	477936.3125	13.382	3.858	26.569	7.607	11	15446	src_L
2	59.633	57.023	2.610	477936.3125	15.924	4.217	31.709	7.819	12	15446	src_L
3	45.812	43.268	2.524	477875.8125	18.315	4.290	36.628	7.536	13	15440	:hc0C
4	56.358	53.972	2.386	477936.3125	9.152	3.732	17.981	7.225	9	15446	rton1
5	44.638	42.602	2.036	477650.2812	35.050	6.699	69.345	11.573	36	15438	tt_sc
6	47.414	45.418	1.996	477936.3125	3.322	1.828	6.468	3.425	5	15446	rton1
7	43.168	41.502	1.666	477936.3125	5.687	3.508	11.024	6.547	7	15446	rton1
8	33.685	32.120	1.565	477650.2812	33.015	5.669	65.911	9.451	35	15438	tt_sc
9	39.127	37.778	1.349	477850.2812	31.723	4.466	63.169	6.996	34	15438	tt_sc
10	20.265	18.992	1.273	477850.2812	19.586	4.161	39.137	7.186	14	15438	tt_sc
11	23.331	22.166	1.165	477936.3125	10.309	3.769	20.304	7.419	10	15446	rton1
12	25.754	24.607	1.147	477936.3125	6.834	3.677	13.273	6.695	8	15446	rton1
13	46.212	44.304	1.908	363072.8125	36.959	7.772	72.505	13.889	37	11778	tt_sc
14	63.364	61.636	1.728	363072.8125	38.690	9.364	75.364	17.256	38	11778	tt_sc
15	21.463	20.603	0.880	477850.2812	29.764	3.456	59.067	5.035	32	15438	tt_sc
16	17.851	17.058	0.793	477850.2812	30.469	3.782	60.565	5.638	33	15438	tt_sc
17	14.237	13.471	0.766	477850.2812	20.994	3.900	41.767	6.552	16	15438	tt_sc
18	25.592	24.827	0.765	477936.3125	1.325	1.426	2.532	2.719	4	15446	rton1
19	16.429	15.729	0.700	477936.3125	4.021	2.246	7.828	4.176	6	15446	rton1...

Figure 8-18 Clock Jitter Bottleneck Report

Pin timing parameters can be sorted by any header criteria by clicking on the column header. The parameter values shown on the Clock Jitter Bottleneck Report are:

- No: ranking order based on the sorting criteria chosen
- Max Delay: maximum rising or falling transition delay from driver to pin for multiple cycles
- Min Delay: minimum rising or falling transition delay from driver to pin for multiple cycles
- Delay Diff: difference between maximum and minimum delay at the pin
- DS Leaf Jitter: sum of the jitter at all leafs below the pin
- P Jitter (R): period jitter at pin when root transition is rising
- P Jitter (F): period jitter at pin when root transition is falling
- C-C Jitter (R): cycle-to-cycle jitter at pin when root transition is rising
- C-C Jitter (F): cycle-to-cycle jitter at pin when root transition is falling
- Level: the clock tree level from root
- Leaf #: the number of leafs below the pin in the tree
- Pin Name: instance pin name

Buttons at the bottom of the Jitter Bottleneck report are:

- Reset to Default Order: resets ranking list to default sorting order (considering both highest delay difference between pin maximum and pin minimum delay and downstream leaf jitter)
- Go To Location: highlights and zooms to selected location on design. If selected, highlights each new location selected on the list.
- Up: selects jitter bottleneck location above the one presently selected.
- Down: selects jitter bottleneck location below the one presently selected
- Previous 1000: selects the previous 1000 worst case pins in the design, based on the sorting criteria selected
- Next 1000: selects the next 1000 worst case pins in the design, based on the sorting criteria selected

Jitter Color Map

Selecting the menu command **View -> Clock Jitter Map ->** allows you to choose the type of jitter map data to display. The choices are:

- Rise Period Jitter - displays period jitter at pin when root transition is rising. Click on the 'Set Color Range' button on the **RedHawk** 'Configuration' palette to see the meaning of the colors, as a percentage of the maximum jitter in the design.
- Fall Period Jitter - displays period jitter at pin when root transition is falling
- Rise C-to-C Jitter - displays cycle-to-cycle jitter at pin when root transition is rising
- Fall C-to-C Jitter - displays cycle-to-cycle jitter at pin when root transition is falling

A jitter colormap can be displayed using the following TCL commands:

```
show jitter [period | c2c] [rise | fall]
```

One of each of the two options must be chosen.

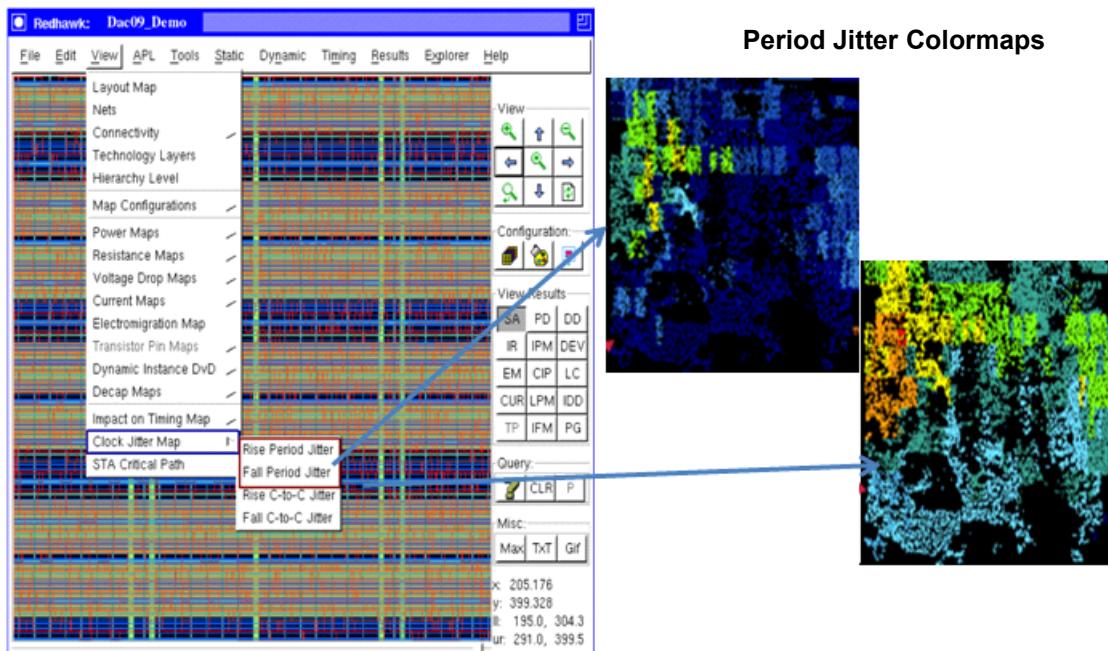


Figure 8-19 Period jitter colormaps

- Period jitter instance colormaps - Click on **View->Clock Jitter Map->Rise/Fall Period Jitter**. The selected colormap is displayed (see Figure 8-19). This can also be displayed using the TCL ‘show jitter’ command.

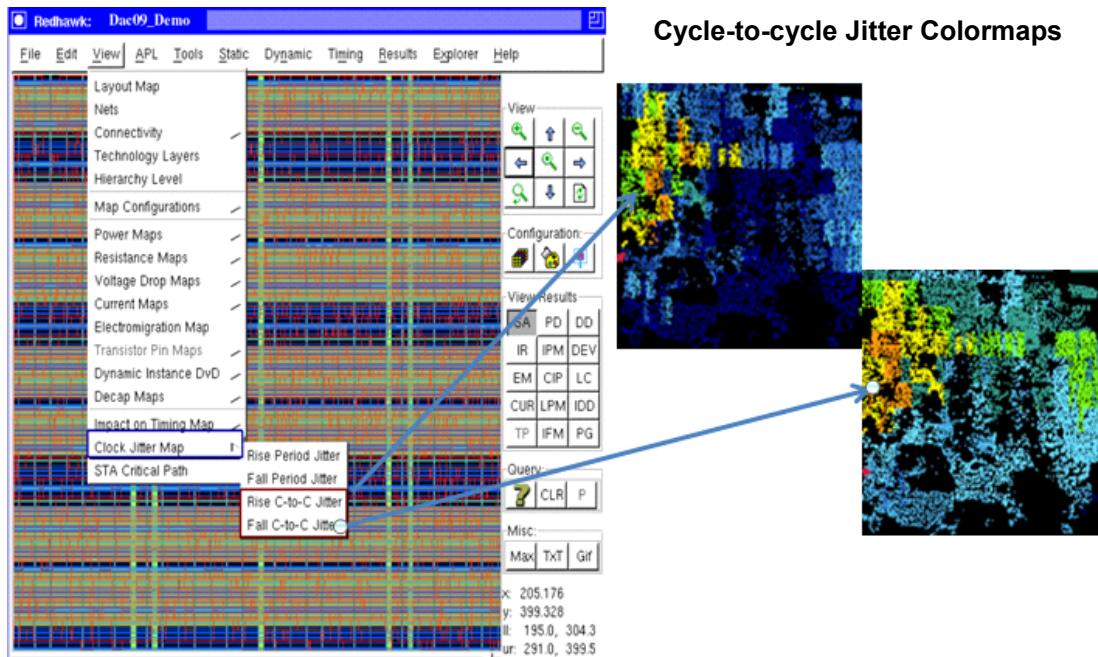


Figure 8-20 Instance cycle-to-cycle jitter colormaps

- Instance cycle-to-cycle jitter colormaps - Click on **View->Clock Jitter Map->Rise/Fall C-to-C Jitter**. The selected colormap is displayed (see Figure 8-20). This can also be displayed using the TCL ‘show jitter’ command.

Text Reports

Sign-off clock tree jitter analysis produces the following files, which are written into the directory ‘*adsRpt/Jitter*’, as shown in the directory structure of Figure 8-21 below. The *jitter.log*, *jitter.err* and, and *jitter.warn* files are updated dynamically to provide intermediate results during a run. Each clock tree root is saved to the sub-directory under ‘*adsRpt/Jitter*’. The subdirectories are labeled ‘*tree_1_pi*’, ‘*tree_2_pisi*’, and so on, for clock tree jitter analysis. All the temporary files created during jitter analysis go to the *apache/jitter* directory.

(Jitter analysis outputs data under the **RedHawk** run directory)

```

|__ adsRpt/Jitter
    |__ jitter.log          ( complete jitter analysis log file )
    |__ jitter.err          ( complete jitter analysis error file )
    |__ jitter.warn         ( complete jitter analysis warning file )
    |__ tree_<id>.map      ( map b/w tree_<id>_<suffix> dir and clock tree roots )
    |__ tree_<id>_<suffix>  ( <suffix> is one of: pi, pisi )
        |__ tree.sdf          ( sdf format output )
        |__ tree.skw          (delay, skew summary report )
        |__ tree.leaf          ( leaf summary report and histogram)
        |__ tree.jitter         ( period and cycle2cycle jitter summary report )
        |__ jitter.bottleneck  (jitter bottleneck summary report )
        |__ tree.jitter_leaf   cycle2cycle jitter summary report and histogram )
        |__ tree.jitter_detail (long term jitter summary report )
        |__ jitter.wfm          (waveform and duty cycle report )
        ...
    |__ DomainRpt/           (duplicate results organized by sub-tree)
        |__ subtree_1/          (sub-domain clock tree reports)
            |__ subtree.skw
            |__ subtree.leaf
            |__ subtree.jitter
            |__ subtree.bottleneck
        |__ subtree_2/
        ...

```

Figure 8-21 Timing analysis directory structure - jitter

To turn off generation of the additional *DomainRpt* directory and subtree files, use the following configuration file keyword setting:

```
CLOCKTREE_DOMAIN_SPLIT_REPORT 0
```

Clock Tree Skew Analysis

Overview

During clock skew analysis, **RedHawk** computes gate propagation delays, interconnect delays, and slew, and reports rise and fall skew values for the clock tree. The results reflect actual silicon performance related to the impact of DvD on timing, and allow quick noise-aware timing closure.

Required Inputs and Data Preparation

Required inputs and data preparation are the same as for clock tree jitter analysis. See [section "Required Inputs for Clock Tree Jitter Sign-Off Analysis", page 8-177](#) and [section "Input Data Preparation", page 8-178](#).

Procedure for Clock Tree Skew Analysis

Running Clock Tree Skew Analysis from the RedHawk GUI

To enable analysis of crosstalk impacts, set 'SIGNAL_INTEGRITY 1' in the timing configuration file.

To use the GUI menu commands, after completing the setup procedures for a DvD run, or loading an appropriate **RedHawk** database:

1. Execute the GUI command **Timing-> Sign-off Clock Tree-> Analysis** to perform the analysis. A 'Clock Tree Set up' dialog box is displayed to specify the 'config file Name'.
2. Insert the appropriate configuration file name, and click on **Run**.

Running Clock Tree Skew Analysis in Batch Mode

Following is the method of running clock tree skew analysis in batch mode, relying on proper keyword definitions in the timing configuration file. The full TCL syntax is:

```
perform clockskew -config <config_file>
?-clk_source <clk_src>? ?-leaf_inst_pin <leaf_inst_pin> ?
?-no_data_check? ?-mode [basic | static | dynamic]?
```

As sample TCL invocation of clock tree skew is:

```
setup design test.gsr
...
perform clockskew -config psi.skew_config
```

Following are the details for running batch mode skew analysis, depending on the selected analysis mode.

Basic Analysis Mode

The basic analysis mode can be performed right after 'setup design' in **RedHawk**, using the command

```
perform clockskew -config <config_file> -mode basic
```

Note that the above command overwrites 'POWER_INTEGRITY 1' in the timing configuration file.

Signal Integrity Analysis Mode

Signal integrity analysis also can be performed right after 'setup design' in RedHawk, with 'SIGNAL_INTEGRITY 1' set in the timing configuration file, using the same command

```
perform clockskew -config <config_file> -mode basic
```

Make sure that coupling capacitance information is available in the imported SPEF file. Also note that the above setting overwrites 'POWER_INTEGRITY 1' in the configuration file.

Power Integrity Analysis Mode

This mode considers the effects of static or dynamic voltage drops in the skew analysis. The default mode is dynamic, which is equivalent to

```
perform clockskew -config <config_file> -mode dynamic
```

Static voltage drop impacts are analyses using the command

```
perform clockskew -config <config_file> -mode static
```

Note that the above setting overwrites 'POWER_INTEGRITY 0' in the configuration file.

Simultaneous Power and Signal Integrity Analysis Mode

Simultaneous power and signal integrity analysis mode is invoked by

`perform clockskew -config <config_file> -mode [static|dynamic]`
and also specifying 'SIGNAL_INTEGRITY 1' in the configuration file. Make sure that coupling capacitance information is available in the imported SPEF file.

Note that the above setting overwrites 'POWER_INTEGRITY 0' in the configuration file.

Clock Tree Skew Analysis Results

Various graphic displays and output data are produced by Clock Tree Skew Analysis, as described in the following sections.

Clock Tree Skew Summary Report

After performing clock tree skew analysis, use the menu command **Timing > Sign-off Clock Tree > Clock Tree Report** to create a summary, as shown in Figure 8-22.

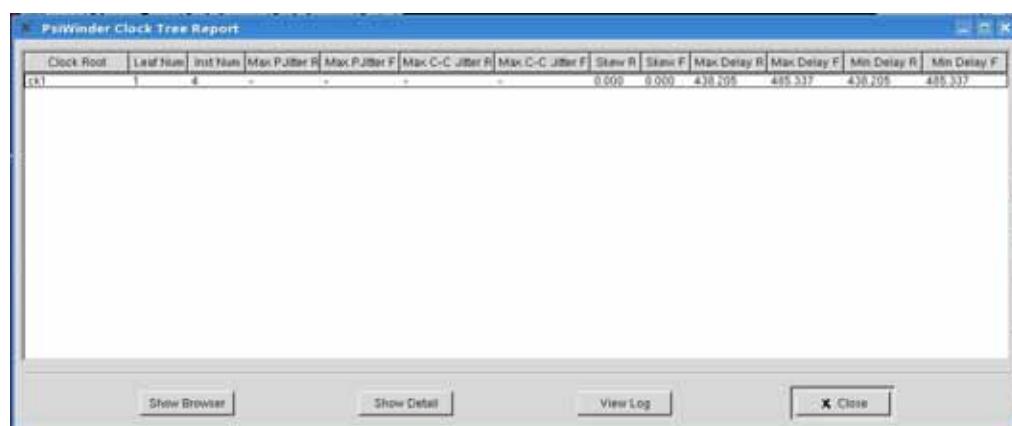


Figure 8-22 Clock Tree Skew summary report

The data displayed in the Clock Skew Report columns are as follows:

- Clock Root - clock tree root pin name

- Leaf Num - number of terminating leafs in the clock tree
- Inst Num - number of instances in the clock tree
- Skew R - difference in longest and shortest rising insertion delay (ps)
- Skew F - difference in longest and shortest falling insertion delay (ps)
- Max Delay R - maximum rising delay in the clock tree, root to leaf (ps)
- Max Delay F - maximum falling delay in the clock tree, root to leaf (ps)
- Min Delay R - minimum rising delay in clock tree, root to leaf (ps)
- Min Delay F - minimum falling delay in clock tree, root to leaf (ps)

Clock Tree Skew Analysis Text Reports

RedHawk clock tree skew analysis produces the following files, which are written into the directory ‘*adsRpt/Clkskw*’, as shown in the directory structure of Figure 8-23 below. Each clock tree root is saved to the sub-directory under ‘*adsRpt/Clkskw*’. The subdirectories are labeled ‘*tree_1_pk*’, ‘*tree_2_pk*’, and so on, for clock tree skew analysis. All the temporary files created during clock tree skew analysis go to the *.apache/clkskw run* directory, as shown in Figure 8-23.

(Clock tree skew analysis outputs data under the **RedHawk** run directory)

```

|__ adsRpt/Clkskw
    |__ skew.log          ( complete skew log file )
    |__ skew.err          ( complete skew error file )
    |__ skew.warn         ( complete skew warning file )
    |__ tree_id_map ( map b/w tree_<id>_<suffix> dir and clock tree roots )
    |__ tree_<id>_<suffix> ( <suffix> is one of: basic, pi, si, pisi )
        |__ tree.sdf       ( sdf format output )
        |__ tree.skw       (delay, skew summary report )
        |__ tree.leaf      ( leaf summary report and histogram)
        ...
    |__ DomainRpt/        (duplicate results organized by sub-tree)
        |__ subtree_1/   (sub-domain clock tree reports)
            |__ subtree.sdf
            |__ subtree.skw
            |__ subtree.leaf
        |__ subtree_2/
        ...

```

Figure 8-23 timing analysis directory structure - skew

To turn off generation of the additional *DomainRpt* directory and subtree files, use the following configuration file keyword setting:

```
CLOCKTREE_DOMAIN_SPLIT_REPORT 0
```

Clock Tree Analysis Configuration File Reference

The timing analysis configuration file allows users to customize the application environment in which they run. The following sections describe the available configuration file keywords and their usage, organized by function:

- Clock Tree Analysis
- Input Data Settings
- Jitter Analysis
- Signal Waveforms
- Simulation Controls
- Spice Elements
- Multi-task Controls
- Constraint Settings
- Application Type Selection
- RedHawk Data Usage
- Report Formats

Clock Tree Analysis Keywords

The following keywords are associated with clock tree analysis.

CLOCK_SOURCES

Specifies clock tree sources/roots for skew/jitter analysis.

Syntax:
CLOCK_SOURCES {
 <clock_root1>
 <clock_root2>
 ...
}

or for a single root, the simplified syntax is:

CLOCK_SOURCES <clock_root>

CLOCK_PATHS

Specifies the clock paths to be analyzed, by pairing root name and leaf names of interest.

Syntax:
CLOCK_PATHS {
 ROOT <clock_root_1> -leafs {<leaf_1> <leaf_2> ... }
 ROOT <clock_root_2> -leafs {<leaf_1> <leaf_2> ... }
 ...
}

COMMON_CLOCK_SIMULATION

Common simulations for all clock roots defined in the keyword CLOCK_SOURCES, speeding up overall simulation run time.

Syntax:
COMMON_CLOCK_SIMULATION [0 | 1]

CLOCKTREE_BYPASS_UNDEFINED_GRAY_CELL

Continues clock tracing through “undefined” gray cells. A gray cell is undefined if it is missing its Spice netlist data.

Syntax:

```
CLOCKTREE_BYPASS_UNDEFINED_GRAY_CELL [ 0 | 1 ]
```

CLOCK_TREE_MAX_LEVEL

Sets the limit on the number of levels of clock trees for simulation, which is useful for pipe clean testing.

Syntax:

```
CLOCK_TREE_MAX_LEVEL <value>
```

LEAF_INST_PINS

Specifies instance pins to stop clock tracing from clock source/root.

Syntax:

```
LEAF_INST_PINS {
    <leaf_pin1>
    ...
}
```

Input Data Settings

The following keywords are associated with input data settings:

CLOCK_INST_FILE

Supports a customer-specified clock instance file, which lists instances that should be simulated. The keyword syntax is:

```
CLK_INST_FILE <fileName>
```

The file format is as follows:

```
#| Clock Tree Instance Summary
# Root=CK1 Period=2000(ps) Freq=500.000(Mhz)
# InstName CellName (InstType) Freq EffVdd
T_buf1 zbfp (C_COMB) 500.000MHz
T_buf3 zbfp (C_COMB) 500.000MHz
T_buf4 zbfp (C_COMB) 500.000MHz
T_buf5 zbfp (C_COMB) 500.000MHz
...
# Buffer/Gate=29 Sequential(Total) Leafs=2(2)
# Buf/Gate: Number of clock buffer and gates in the clock tree
# Leafs(Sequential): Number clock leafs and sequential leafs in the
# clock tree
# Max/Min Lvl: Maximum and minimum level of the clock tree
# Clock Network Summary:
# -----
# Freq(Mhz) Buf/Gate Leafs(Sequential) Max/Min Lvl Root_Net Root_Pin
# -----
# 500.000      29      2( 2)          31/ 31      CK1      CK1
```

SPICE_NETLIST

Specifies one or multiple Spice cell netlist files.

Syntax:

```
SPICE_NETLIST {  
    <Spice_file1>  
    ...  
}
```

or for a single netlist, the simplified syntax is:

```
SPICE_NETLIST <Spice_file>
```

SPICE_SUBCKT_DIR

Specifies the directory path containing the Spice cell netlist files.

Syntax:

```
SPICE_SUBCKT_DIR <dir_path>
```

STA_CONST_READ

Turning this off (0) suppresses reading “CONST” lines in the STA file.

Syntax:

```
STA_CONST_READ [ 0 | 1 ]
```

Jitter Analysis

The following keywords are associated with jitter analysis:

CYCLE_SELECTION

Selects criteria for automatic jitter cycle selection to determine the number of cycles to be included in the jitter analysis, or turns cycle selection off.

Syntax:

```
CYCLE_SELECTION  
[ DISABLE | WORST_C2C_JITTER_CYCLE | WORST_PERIOD_JITTER_CYCLE |  
WORST_MIN_PERIOD_JITTER_CYCLE | WORST_MAX_PERIOD_JITTER_CYCLE ]
```

DYNAMIC_SIMULATION_CYCLES

Specifies the number of specific simulation cycles to be used for clock tree jitter analysis, either by listing each one, or by specifying first and last.

Syntax:

```
DYNAMIC_SIMULATION_CYCLES  
[ <cycle1> <cycle2> ... <cycleN> | <cycle1> : <cycleN> ]
```

JITTER_CUTOFF_FREQ

Specifies the minimum simulation clock frequency for jitter analysis to allow elimination of clock domains, or instances with very low frequencies, and to speed up runtime. Default 4 MHz.

Syntax:

```
JITTER_CUTOFF_FREQ <value_MHz>
```

JITTER_FAST_MODE

Setting this keyword to 1 turns on fast jitter simulation analysis with eff DvD for PWL waveforms. Setting to 2 turns on multi-cycle effective VDD support. Default 0 - Off.

Syntax:

```
JITTER_FAST_MODE [ 0 | 1 | 2 ]
```

TIMING_WINDOW_OVERLAP_RATIO_THRESHOLD

In jitter analysis with signal integrity, this keyword can control the aggressor selection for coupling based on the amount of timing window overlap of aggressor and victim. The overlap ratio is calculated as the fraction of the maximum possible period that the aggressor and victim timing windows overlap, where the maximum period is the smaller of the two 'On' periods. For example, if an aggressor has a TW between time 2 and 10 and the victim has a TW between time 8 and 20, the overlap is between time 8 and 10, or 2. The maximum overlap is the smaller timing window. That is, $10 - 2 = 8$, which is less than $20 - 8 = 12$. So the overlap ratio = $2/8 = 0.25$. Aggressors with overlap ratios greater than the user-specified threshold value (between 0 and 1.0) are selected for coupling analysis for the victim.

Syntax:

```
TIMING_WINDOW_OVERLAP_RATIO_THRESHOLD <value>
```

Signal Waveforms

The following keywords are associated with signal waveform specifications:

FALL_SLEW

Sets the fall time in ns. Default 0.05 ns.

Syntax:

```
FALL_SLEW <value>
```

FALL_SLEW_AGRS

Sets the coupling aggressor fall time in ns. Default 0.05 ns.

Syntax:

```
FALL_SLEW_AGRS <value>
```

PWL_WAVEFORM

Defines waveform measurement points, which must be a fraction of Vdd and must be in ascending order. The delay is measured at point 'M:' keyword. The slope is measured between two points, 'L:' and 'H:'. A maximum of 10 measurement points is allowed. Default values: 0.1, L: 0.3, M: 0.5, H: 0.7, 0.9.

Syntax:

```
PWL_WAVEFORM <val1>...<valN>
```

RISE_SLEW

Sets the rise time in ns. Default 0.05 ns.

Syntax:

```
RISE_SLEW <value>
```

RISE_SLEW_AGRS

Sets the coupling aggressor rise time in ns. Default 0.05 ns.

Syntax:

RISE_SLEW_AGRS <value>

Simulation Controls

The following keywords are associated with Spice simulation control:

SPICE_SIMULATOR

Specifies the type of Spice simulator. The command name is the same executable binary or shell script as the Spice simulator. The binary (or shell script) must be accessible in the run environment.

Syntax:

SPICE_SIMULATOR [NSpice | HSpice | Eldo]

SETTLE_TIME

Sets the simulation data transition time interval in ns. Default 0.

Syntax:

SETTLE_TIME <value>

SPICE_TIME_STEP

Sets the time increment for Spice transient analysis in ns. Default 0.01.

Syntax:

SPICE_TIME_STEP <value>

SPICE_PROBE_USE

Turns on printing of “.probe” lines in the Spice deck and generating *.ta0 signal waveform file. Default 0.

Syntax:

SPICE_PROBE_USE [0 | 1]

SPICE_FILE_BACKUP

If set to 1, writes out individual Spice decks for each run. Default 0.

Syntax:

SPICE_FILE_BACKUP [0 | 1]

GROUP_CLUSTERING

Allows performing cluster-based jitter simulation. **RedHawk** splits the clock tree into smaller clusters and simulates them serially, which improves overall simulation time. The output clock waveform from each cluster is propagated to the subsequent one to ensure the accuracy of the results. Default: DISABLE.

Syntax:

GROUP_CLUSTERING [ENABLE | DISABLE]

GROUP_CLUSTERING_DEPTH

Defines the maximum number of logic levels for each Spice run. Allows configuring the depth of clock tree clusters per group. The default value is 3, which means that each cluster created has a maximum depth of 3.

Syntax:

```
GROUP_CLUSTERING_DEPTH <cluster_depth_per_group>
```

SIMULATION_REDO_MAX

Defines the maximum number of iterations allowed for PWL waveform alignment. Default 10.

Syntax:

```
SIMULATION_REDO_MAX <value>
```

DEVICE_MODEL_LIBRARY

Specifies the full file path to the SPICE device model library (parameterized device models). The model files can be defined multiple times. Use the process corner specified in the LIB file, such as "TT", "FF", "SS".

Syntax:

```
DEVICE_MODEL_LIBRARY <file_path>
```

INCLUDE

Specifies the full file paths for SPICE device models (.models) or files for other parameter values needed. The models can be defined multiple times. This keyword directly passes specified models to NSPICE's 'INCLUDE' function.

Syntax:

```
INCLUDE <Spice_device_model_file_path>
```

OPTION

Specifies any needed Spice simulation options, such as 'OPTION mode=turbo'.

Syntax:

```
OPTION <Spice option1>
OPTION {
    <Spice option2>
    ...
}
```

Spice Elements

The following keywords are associated with Spice elements:

SPF_COUPLE_CAP_MULTIPLIER

Defines coupling capacitor multiplier globally in SPEF. Default 1.

Syntax:

```
SPF_COUPLE_CAP_MULTIPLIER <value>
```

COUPLE_CAP_MULTIPLIER

Defines grounded coupling capacitor multiplier globally. Default 1.

Syntax:

```
COUPLE_CAP_MULTIPLIER <value>
```

CAP_MULTIPLIER

Defines ground capacitor multiplier globally. Default 1.

Syntax:

```
CAP_MULTIPLIER <value>
```

RES_MULTIPLIER

Defines resistor multiplier globally. Default 1.

Syntax:

```
RES_MULTIPLIER <value>
```

COUPLING_CAP_RATIO_THRESHOLD

Specifies the ratio between 0 and 1.0 of aggressor net coupling capacitance to the total victim net capacitance. Includes the aggressor for coupling analysis if its ratio \geq the threshold. Increasing the ratio value reduces runtime and accuracy. Default 0.05.

Syntax:

```
COUPLING_CAP_RATIO_THRESHOLD <value>
```

COUPLING_WINDOW_MARGIN

Specifies the allowable extension of the victim TW (in number of rise times) to check possible effects of overlapping with aggressor TW. Default=1.0 (value of 1 rise time in both positive and negative directions).

Syntax:

```
COUPLING_WINDOW_MARGIN <value>
```

CAP_LOAD_UNIT

Specifies the capacitance unit for 'set_load' command in configuration file keyword DEFINE DESIGN CONSTRAINTS. Default unit is femto-Farad (1e-15).

Syntax:

```
CAP_LOAD_UNIT <value>
```

SPEF_CAPACITANCE

Specifies the capacitance value for missing SPEF data for the net in specified SPEF file(s) in femtoFarads. Default 0.001.

Syntax:

```
SPEF_CAPACITANCE <value>
```

SPEF_RESISTANCE

Specifies the value for missing SPEF resistance data for nets in specified SPEF file(s), in Ohms. Default 0.001

Syntax:

```
SPEF_RESISTANCE <value>
```

Multi-task Controls

The following keywords are associated with multi-task controls:

GRID_TYPE

Specifies the multi-tasking job management platform. Default MULTICPU.

Syntax:

```
GRID_TYPE [ LSF | SUNGRID | MULTICPU ]
```

JOB_COUNT

Specifies the maximum number of parallel jobs to run. Note that the number should not exceed the number of 'nspice_psi' licenses. Set the keyword to '0' to run in single job mode. Default= number of CPU's * 2.

Syntax:

```
[ JOB_COUNT | LSF_JOB_COUNT ] <max number of jobs >
```

PARALLEL_SIMULATIONS

Allows simulating multiple clock networks in parallel to improve run time for cases with multiple clock roots. Set JOB_COUNT also to control CPU use. Default OFF.

Syntax:

```
PARALLEL_SIMULATIONS [OFF | ON]
```

BATCH_QUEUEING_COMMAND

Selects batch queuing type. Specify 'bsub' for LSF and 'qsub' for SUNGRID.

Syntax:

```
BATCH_QUEUEING_COMMAND [bsub | qsub ]
```

BATCH_QUEUEING_OPTIONS

Specifies options needed in the bsub command arguments or qsub shell script file.

Syntax:

```
BATCH_QUEUEING_OPTIONS <options>
```

QUEUE

Defines name of LSF/SUNGRID queue. For SUNGRID, add comma between names of queues, such as 'apl24.q, apl25.q, apl26.q'.

Syntax:

```
QUEUE <string>
```

EXEC_PATH

Defines the path to the bsub/qsub binary, which can be different from the environment setting.

Syntax:

```
EXEC_PATH <binary_path>
```

TIMER

Defines the desired wait time to check parallel job status, in seconds. Default 60.

Syntax:

```
TIMER <wait time>
```

LSF_JOB_TIMEOUT

Defines timeout limit for pending parallel jobs, in hours. Aborts if there are no active jobs and all pending jobs exceed the timeout limit.

Syntax:

```
LSF_JOB_TIMEOUT <hrs>
```

Constraint Settings

The following keywords are associated with constraint settings:

DEFINE_DESIGN_CONSTRAINTS

Defines timing constraints as follows:

- set_dc <pin_name> <voltage>: sets DC voltage for the Spice netlist ports during simulation
- set_clock_divider: specifies the clock divider circuitry to allow analysis passing through the flip-flop.
- set_logic_zero or set_logic_one: sets pin logic state.
- create_clock : specifies clock source for the clock period in the duty cycle report. Unit is seconds.
- set_disable_timing: specifies the blockage for clock signal propagation.
- set_propagated_clock_cell: specifies timing arc for clock propagation through.
- set_black_cell: sets the cell to be a black box cell. RedHawk does not try to simulate the cell and continues propagating through it, whether the cell has a Spice netlist or not.
- set_clock_leaf: sets port name or pin name to be a leaf

Syntax:

```
DEFINE_DESIGN_CONSTRAINTS
{
    set_dc <pin_name> <voltage>
    set_clock_divider <port_name>
    set_logic_zero <port_name>
    set_logic_one <port_name>
    create_clock -period <value> -waveform { <value> <value> }
        <clock_root>
    set_disable_timing [-from <pin_name> -to <pin_name>]
        [get_cells {inst_name}]
    set_propagated_clock_cell <cell_name> <pin_name1><pin_name2>
    set_black_cell <cell_name>
    set_clock_leaf <port_name/pin_name>
}
```

Application Type Selection

The following keywords are associated with application type selection:

POWER_INTEGRITY

Turns off power/ground noise impact analysis. Clock Tree Jitter Analysis automatically turns on P/G noise analysis.

Syntax:

```
POWER_INTEGRITY [ 0 | 1 ]
```

SIGNAL_INTEGRITY

Turns on capacitive coupling noise impact analysis.

Syntax:

```
SIGNAL_INTEGRITY [ 0 | 1 ]
```

RedHawk Data Usage

The following keywords are associated with **RedHawk** data usage.

DVD_ALIGNMENT_CONFIGURATIONS

In Jitter analysis with effective P/G waveforms, P/G noise introduces delay in clock switching, which causes misalignment between the clock and the P/G waveform. This keyword allows back annotation of the delay due to P/G noise, and aligning the dynamic waveforms with the instance switching times.

Syntax:

```
DVD_ALIGNMENT_CONFIGURATIONS {
    MAX_SIMULATION_REDO <num_iterations>
    PG_WAVEFORM_SHIFT_PERCENT <Max_percent>
}
```

DVD_WAVE_WINDOW_RANGE

Defines the time span of power/ground waveforms with respect to each cell switching point for Spice simulation. Two numbers define the spans for starting time and end time at the switching point. Defaults 0.5, 2 ns.

Syntax:

```
DVD_WAVE_WINDOW_RANGE <val1> <val2>
```

DVD_WAVEFORM_ALIGN

Sets alignment of P/G noise switching time to the cell Spice simulation switching time. Set to '1' to align P/G noise switching time to the cell Spice simulation switching time. Set to '0' to disable P/G waveform alignment. Set to '-2' to plug original P/G waveform into Spice simulation, without automatic alignment. Default 1.

Syntax:

```
DVD_WAVEFORM_ALIGN [ 0 | 1 | -2 ]
```

DEF_CONNECTIVITY

When set to 1, turns on DEF connectivity data from **RedHawk** as reference. Otherwise uses SPEF data to define net connectivity (0).

Syntax:

```
DEF_CONNECTIVITY [ 0 | 1 ]
```

EXTRACT_NODE_LIMIT

Specifies the maximum limit of node numbers in one batch for power/ground waveform extraction. The size control can improve waveform extraction runtime. Default 50000.

Syntax:

```
EXTRACT_NODE_LIMIT <value>
```

Report Formats

The following keywords are associated with report formats:

JITTER_VERBOSE_REPORT

Turns on printing of complete jitter report with statistics and details. Default 0.

Syntax:

```
JITTER_VERBOSE_REPORT [ 0 | 1 ]
```

PERIOD_JITTER_DEFINITION

Turns on period jitter definition as the difference of max and min jitter periods. Default '0' defines period jitter as the deviation from the ideal period.

Syntax:

```
PERIOD_JITTER_DEFINITION [ 0 | 1 ]
```

CLOCKTREE_DOMAIN_SPLIT_REPORT

Turns on printing of clock sub-tree report from master tree report separated by clock divider. Default 1.

Syntax:

```
CLOCKTREE_DOMAIN_SPLIT_REPORT [ 0 | 1 ]
```

REPORT_JITTER_AS_PERCENT_OF_ROOT_FREQUENCY

Turning on (1) generates the jitter report based on clock root frequency. Default 0.

Syntax:

```
REPORT_JITTER_AS_PERCENT_OF_ROOT_FREQUENCY [ 0 | 1 ]
```

SORT_JITTER_REPORT

Sorts the jitter report based on Jitter_Number or Level. Default 'Jitter_Number'

Syntax:

```
SORT_JITTER_REPORT [ Jitter_Number | Level ]
```

Sample Timing Configuration File

The following is an example of a timing configuration control file, with inactive keywords commented out:

```
#| -----
#| Timing Config File:
#| -----



CLOCK_SOURCES
{
    chip_sclk_src
    edt_clock
    tc_tck_routes
}

SPICE_NETLIST
{
    user_data/SPI/cell.spi
```

```

        user_data/SPI/iocell.spi
    }

DEVICE_MODEL_LIBRARY /circuits/spice/models/spice_model.hlib ss

# The following keywords are optional, shown commented out
# --- SIGNAL WAVEFORM --- #
# PWL_WAVEFORM 0.1 RL: 0.25 FL: 0.45 M: 0.5 RH: 0.55 FH: 0.75 0.9

# --- SIMULATION CONTROL --- #
# CIRCUIT_SIMULATOR nspice
# TEMPERATURE 125
# SETTLE_TIME 10
# SPICE_TIME_STEP 0.01
# GROUP_CLUSTERING_DEPTH 5

# --- CONSTRAINT SETTING --- #
# DEFINE_DESIGN_CONSTRAINT
# {
#     set_logic_zero mux1:s1
#     set_logic_one mux1:s2
#     set_clock_divider div_reg:q
#     set_disable_timing -from A -to y [get_cells {sndk/ABCck_id_5} ]
#     set_propagated_clock_cell xyzvc a y 0.12
#     set_black_cell Mnop3_bsfbm
# }

# --- APPLICATION TYPE SELECTION --- #
# SIGNAL_INTEGRITY 0

# --- JITTER ANALYSIS --- #
# JITTER_CUTOFF_FREQ 40
# JITTER_FAST_MODE 1

# --- CLOCK TREE ANALYSIS --- #
# COMMON_CLOCK_SIMULATION 1
# CLOCKTREE_BYPASS_UNDEFINED_GRAY_CELL 0
# CLOCK_TREE_MAX_LEVEL 5

# --- INPUT DATA SETTINGS --- #
# STA_CONST_READ 0

```

Chapter 9

Characterization Using Apache Power Library

Introduction

The Apache Power Library program is used to characterize cells, creating accurate switching current waveforms (profiles), output-state dependent decoupling capacitance (intrinsic decap), equivalent power circuit resistance (called ESR, “Effective Series Resistance”), switching delay, and leakage current, all at multiple conditions, for the desired set of cells. These data are required for accurate RedHawk dynamic analysis. APL has three basic modes:

- fast library checking mode
- design-independent full library characterization mode
- design-dependent characterization mode, for cells under particular design conditions for sign-off quality

The general APL flow is shown in Figure 9-1.

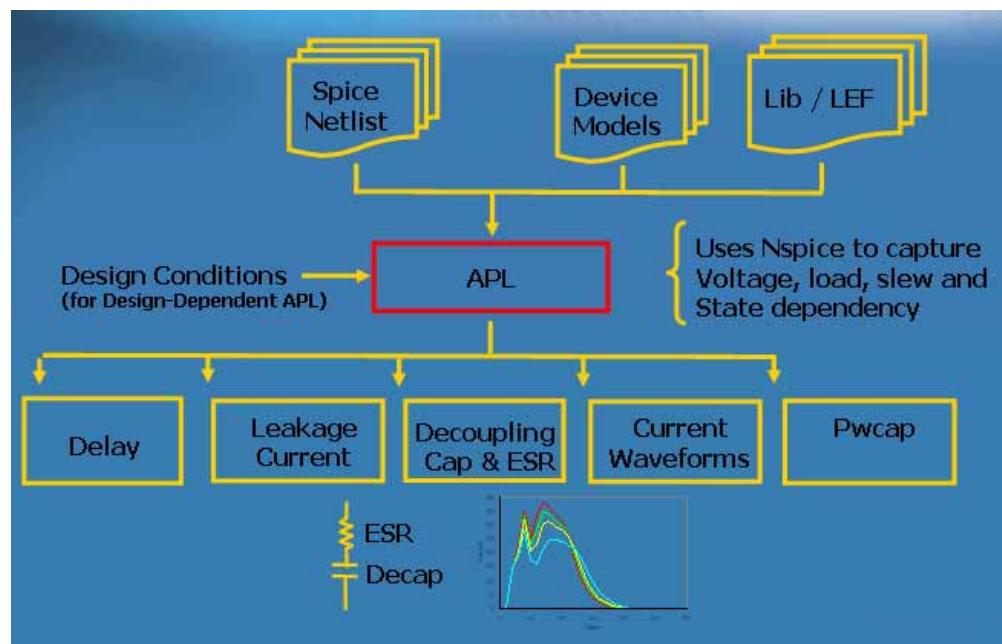


Figure 9-1 APL data flow for cell characterization

Each mode of APL produces a set of “sample” switching waveforms for each cell for selected values of output capacitance loading, transition time (“slew”) and Vdd. In fast library checking mode only one sample is generated—that is, one set of waveforms for each cell at one Vdd, Cload, and slew condition. For APL-DI characterization, switching current profiles for many combinations of Vdd, and Cload/slew are generated to cover the expected usage range of each cell. In design-specific characterization, waveform samples for the particular design conditions are generated, hence fewer samples are generated, but they are more accurate for the design than are those from design-

independent characterization. Separate APL runs are required to characterize decap cells for regular, as well as low-voltage, designs with header and footer switch components.

Overview of APL Characterization

Types of Cell Checking and Characterization

RedHawk has several types of cell checking and characterization, depending upon the scope, accuracy and speed of analysis you require, as follows:

- Pre-run Sample Integrity Checking
- Fast Library Checking
- Library, or Design-Independent (APL-DI) Characterization
- Design-Dependent (APL-DD) Characterization

These methodologies for cell checking and characterization are compared in the following section.

Pre-run Sample Integrity Checking

To check input sample data integrity for transition times (“slew”), Cload and Vdd values, APL checks data from `*.lib` or other similar tools to screen for out-of-range data that can corrupt APL results and significantly reduce computation speed.

Sample checking is performed before characterization starts upon invoking either Fast Library Checking or APL-DI. The APL configuration file keywords used to set the data out of range limits for transition times, Cload values, and Vdd values are `WARN_SLEW_CHECK`, `ERROR_SLEW_CHECK`, `WARN_CLOAD_CHECK`, `ERROR_CLOAD_CHECK`, `WARN_VDD_CHECK`, `ERROR_VDD_CHECK`, and `MIN_VDD_CHECK`.

Fast Library Checking

The fast library checking mode runs an error check of the Spice netlist, device models, LEF/DEF and Synopsys LIB files for each library cell at one corner condition, as a way of verifying basic library data integrity. This is recommended as a first step before characterization, but no actual characterization is performed in this step. Typical fast checking mode run time for a 2000-cell library would be a few hours on a desktop computer.

Library (APL-DI) and Design-dependent (APL-DD) Characterization

All characterization functions can be run on a single machine or multiple machines from a UNIX command line using the command `aplди`, both full library (DI, the default) and design-specific (DD) characterization, except for memories. To characterize memories, use the `sim2iprof` utility (see [section "sim2iprof", page E-922](#)) and ACE utility (see [section "ACE Decap and ESR Characterization", page 9-263](#)). For library characterization, `aplди` reads cell library data from `*.lib` files, generates representative samples based on cell delay and load tables, and runs on a local machine or multiple machines (through multi-job management software). It generates characteristics for each specified process corner.

Applicability

Library characterization, or design-independent characterization, is performed on the whole cell library, generally before any real design starts. APL-DI picks representative switching conditions from a matrix of input slew, output capacitive load, and Vdd values for each cell and runs characterization to extract each cell's dynamic switching current

and switching delay, as well as the intrinsic device capacitance and leakage current. Design-dependent characterization is performed for the cells in a design from SPEF and STA data, so all switching relationships to input slew, output Cload, and Vdd values are specific to the design.

Accuracy

Since APL-DI is usually run on a full library before a design is available, and the specific operating conditions are not known, it is less accurate than using APL-DD design-specific data. However, APL-DI analysis results typically are within 10% of DD results.

Run time

APL-DI is run one time on the complete library, whereas APL-DD is run for each new design. For a DI run, depending on the library size and computing resource used, the run time can be a day to a week (for example, a 2000-cell library characterization time can be around 3 days using a 10-machine LSF farm). For DD, the run time depends strongly on the number of cells and the number of samples chosen. Due to the large number of cells/samples to be characterized, using a batch computing farm (that is, LSF or SunGrid), or a local machine with multiple CPUs, to run the characterization is recommended.

Simulator Support

By default APL uses its internal NSPICE simulator for characterization, which has tested Spice accuracy, but it also supports direct use of the HSPICE simulator using the APL configuration file keyword APL_HSPICE <binary_path>, and also the ELDO simulator with the APL configuration keyword APL_ELD0 <binary_path>.

Characterization Functions

From a characterization viewpoint, there are three categories of design components, which are handled somewhat differently in the characterization flow:

- standard library cells, I/O cells, and custom IP
- intentional decoupling capacitance cells
- memories

There are three types of information required to provide complete operational characterization:

- current waveforms under switching conditions, as well as delay and slew data
- values of intrinsic device decap, leakage current and effective power circuit resistance (“ESR”) at nominal voltage for On and Off conditions.
- piecewise linear voltage functions for intentional decap values and library cell performance under power-up conditions for power gating applications.

To perform characterization on a complete set of components for a library or design, the following Apache characterization program runs are required, each of which includes an appropriate number of corner conditions for the accuracy requested:

1. To obtain switching current, switching delay and slew for all library standard cells, I/O cells, and custom IP other than memories, use the *aplди* program
2. To obtain intrinsic decap, ESR, and leakage current for all library cells, I/O cells, and custom IP other than memories, use *aplди -c*.
3. To obtain intrinsic decap, ESR, and leakage current for intentional decap cells, use *aplди -p*.

4. For memories, use *sim2iprof* to obtain switching current, switching delay and slew, and ACE to obtain intrinsic decap, ESR, and leakage current. Or for faster characterization, use *avm* to perform datasheet-based memory characterization.

For low power designs such as power gating, with component on and off cycles, additional characterization runs are needed to analyze their behavior under voltage ramp-up conditions:

5. To obtain piecewise linear voltage relationships for intentional decap and switching current, delay, and slew for standard cells, I/O cells, and custom IP, use *apldi -w*.
6. To create characterization data for low-power header and footer switches, use *aplsw*.

Apache cell characterization software is summarized in Table 9-1.

Table 9-1 Apache characterization software

Program	Description
<i>apldi</i>	Performs characterization on all library cells, using a single machine or multiple machines in parallel. Generates waveform-based Vdd/Vss switching current profiles, equivalent power circuit series resistance and leakage current, and intrinsic decaps. In design-dependent mode, characterizes cells for a set of specific design conditions.
<i>apldi -c</i>	Generates intrinsic decap, ESR, and leakage current for all library cells, I/O cells, and custom IP other than memories
<i>apldi -p</i>	Generates intrinsic decap, ESR, and leakage current for intentional decap cells
<i>aplsw</i>	Generates characterization data for header and footer switches used in power-gating low power designs
<i>apldi -w</i>	Performs low-power characterization of standard cells, intrinsic decap, leakage current and ESR, creating a piecewise linear function of voltage
<i>sim2iprof</i> and ACE	Performs characterization on custom cells, I/O cells, memories, and IP macros. Generates waveform-based Vdd/Vss switching current profiles, equivalent power circuit series resistance and leakage current., and intrinsic decaps.
<i>avm</i>	Generates fast datasheet-based Vdd/Vss current profiles and device decaps for memory and IP macros.

Multiple Machine Batch Management

To run parallel APL characterization jobs, Platform LSF or Sun Grid batch management package should be installed on the network, so that the program can be accessed by all needed computing machines. To use Platform LSF or Sun Grid programs, set up the appropriate commands and utilities, similar to a C-shell environment. For example:

```
For Platform LSF: %source /appl$/lsf/conf/cshrc.lsf
For Sun Grid: %source /appl$/sge/default/common/settings.csh
```

Note that on all participating machines the working directory should be readable and writable. For more information on Platform LSF and Sun Grid use, please refer to their respective user manuals.

The environment variable APACHEROOT should be defined before running *apldi*. For example:

```
setenv APACHEROOT /install_dir/apacheda/<RedHawk_release>
```

See the [section "Running APL Characterization from a UNIX Shell", page 9-246](#), for the syntax for batch submittal.

Platforms Supported

APL currently supports the following five platforms:

1. ix86-ent3-32b - 32-bit RedHat Enterprise version 3
2. opt-ent3-64b - 64-bit Opteron RedHat Enterprise version 3.x
3. ix86-ent3-32b - 32-bit CentOS version 4
4. opt-ent3-64b - 64-bit CentOS version 4
5. sparc-sol8-64b - 64-bit Solaris version 8.x

APL Working Directory

During characterization, APL needs a temporary directory to store the intermediate working files, such as Spice netlist files and simulation result files. The default working directory is `.apache/APL`. You can change the working directory by using the `TMP_DIR` keyword in the APL configuration file.

It is important to make sure that the temporary directory has enough disk space to work with, especially if there are many large Spice subcircuit netlist files to be included in the characterization. For example, for a typical library of 500 cells, 500 MB or more of available disk space is recommended for characterization.

Cell Characterization Data Preparation

Data Requirements

The data required for standard cell APL characterizations are described in the this section. The vectors used in characterization are automatically generated by APL. Before running APL you need the following files:

- *.lib files - cell library data files, including cell function, pin definitions, and state tables.
Required for running library APL characterization. (Standard design files.)
- P/G arc definitions - for designs that have cells with multiple Vdd and multiple Vss pins, the P/G arcs must be defined for decap characterization, which arc-based. For design-independent (DI, full library) characterization, APLDI can interpret the P/G arc data if the `.lib` “related_power/ground_pin” keywords exist (otherwise, the P/G arcs must be defined in a custom `.lib` file). The syntax in LIB is:

```
pin(D) {  
    direction : input;  
    related_power_pin : VDD;  
    related_ground_pin : VSS;  
}
```

APLDI reads the 'related_power_pin' and 'related_ground_pin' to get the P/G arcs as VDD to VSS. If overlaps are found in the P/G arc data between the custom lib and the `.lib` files, then the priority to choose the P/G arcs is as follows:

- a. cell level P/G arcs from custom lib
- b. cell level P/G arcs from LIB

c. file level P/G arcs from custom lib

For design-specific (DD) characterization, the file `.apache/apache.pgarc` generated by RedHawk is needed. APL automatically searches for this file.

- Spice subcircuit files - the Spice netlist for each cell. Must be in HSpice-compatible format. You can have multiple subckt files. The pin order can be different. Also, VDD/GND pins are usually not defined in the cell library. This is allowable; APL matches the missing VDD/GND pins automatically. (Standard design files.)
- Spice device model files - the device model file to run Spice. Model of process corners for each device referenced in the Spice netlist. Must be in HSpice-compatible format. Multiple library model files/entries are allowed. (Standard design files.)
- Spice to LEF pin mapping - to handle P/G pins with different names in SPICE and LEF APL outputs LEF P/G pin, but uses Spice P/G pin internally. The APL keyword `SPICE2LEF_PIN_MAPPING` applies to switching currents, CDEV, and PWCAP data for all cells.

Only the following mapping is allowed:

1 LEF pin <-> 1 Spice pin

Multiple LEF pins <-> 1 Spice pin

If no pin mapping is given in the APL configuration file, the P/G pin name must be the same in LEF and Spice.

Example:

```
SPICE2LEF_PIN_MAPPING {  
    Vdd_spice      Vdd_lef  
    Vss_spice      gnd_lef  
}
```

- APL configuration file - specifies all of the above files. Usually called `apldi.config` or `apl.config`. In the configuration file you can also specify other parameters such as temperature, ideal power supply voltage, power/ground node names, and scaling factors. The APL configuration file and the required and optional keywords is described in detail in the following section. (Special file required for APL.)
- cell list file - for design-specific cell characterization (if library APL has not been performed) or for custom cell characterization, you need to create a text file listing the cells to be characterized. (Optional special file for APL.)
- input vector files - input vector definitions (Special APL file needed for custom cell characterization. See the [section "Custom Cell Characterization Data Preparation", page 9-241](#))

NOTE: The load capacitance used for the characterization of the cells comes from SPEF/DSPF files defined in the `.gsr` file. Otherwise, Steiner tree approximations are used.

The slew rate used for the characterization of the cells comes from the PrimeTime data defined in the `.gsr` file. Otherwise, the default slew value specified in the `.gsr` is used.

APL Configuration File Description

The configuration file contains the information required to perform cell characterization. There are several categories of keywords in this section, which are divided into

- keywords required for all types of APL runs

- keywords required for library APL (design-independent) only
- optional keywords

Required APL Configuration File Keywords

The following is a list of required keywords used in the configuration file for cell characterization.

APL_RUN_MODE

APL_RUN_MODE defines the APL run mode, either design-dependent (DD) or library-based design-independent (DI). The default is DI.

Note: If there is no DESIGN CORNER definition, the run is design-dependent, even if DI is specified for APL_RUN_MODE.

Syntax:

APL_RUN_MODE [DD | DI]

Example:

APL_RUN_MODE DI

DC

Allows specification of special DC voltage values for particular pins during characterization.

Syntax:

DC <pinName> <voltage_value>

Example:

DC VSSV 0

Note that if the pins are defined as DC pins, they should be present on a ‘.subckt’ line in the subcircuit definitions.

DEVICE_MODEL_LIBRARY

Specifies the SPICE device model library (parameterized device models) file used for cell characterization. The path to the model library file must be a full path. The model files can be defined multiple times. Use the process corner specified in the LIB file, such as “TT”, “FF”, “SS”.

Syntax:

DEVICE_MODEL_LIBRARY
 <SPICE_model_lib_path> <process_corner>

Example:

DEVICE_MODEL_LIBRARY /home/design/13um.lib TT

INCLUDE

Specifies files for SPICE device models (.models) or files for other parameter values needed for cell characterization. The path to the files must be a full path. The models can be defined multiple times. This keyword directly passes specified models to NSPICE’s INCLUDE function.

Syntax:

INCLUDE <SPICE_device_model_filePath> ...

Example:

INCLUDE /home/design/spice.models

PROCESS

To comply with the MSDF flow, each characterized `<cell>.current` file needs to provide the process corner name for RedHawk simulation. Although this information is usually available from the Spice device library (corner name), the meaning of the library names may not be clear to the timing tool, so this specification is required. For RedHawk computation, WC=SS=slow, BC=FF=fast, and TC=TT=typical. Note that if the DESIGN_CORNER keyword is specified, PROCESS should be a DESIGN_CORNER sub-keyword, but otherwise PROCESS is specified separately.

Syntax:

```
PROCESS [ FF | TT | SS | BC | TC | WC ]
```

Example:

```
PROCESS FF
```

REDHAWK_WORKING_DIRECTORY

Defines the path to the working directory for the RedHawk run.

Syntax:

```
REDHAWK_WORKING_DIRECTORY <RedHawk_working_directory>
```

Example:

```
REDHAWK_WORKING_DIRECTORY /home/apache_work
```

SIZE_SCALE

Optional size scaling factor for all devices in the SPICE netlist. The factor scales the MOSFET's drawn channel length and width. The SIZE_SCALE factor should be set to 1 (default) if the device sizes are specified in microns (*um*), and set to the proper scale value if the device sizes are not in *um* units. For example, if a MOSFET's channel length (*L*) is specified in micron units in the netlist, then the size scale factor should be 1. Likewise, if a MOSFET's *L* is specified in meters, then the correct size scale factor would be 1.0e-6.

Specifying the correct factor value is extremely important, since this directly affects the electrical behavior of each transistor, thus the overall behavior of the cell.

Syntax:

```
SIZE_SCALE <value_for_scaling_factor>
```

Example:

```
SIZE_SCALE 1.0e-6
```

SPICE_NETLIST

Specifies the SPICE netlist file for the subcircuit models used for cell characterization.

Syntax:

```
SPICE_NETLIST <Spice_netlist>
```

Example:

```
SPICE_NETLIST Spice.sp
```

SWEEP_TEMPERATURE

Required keyword in APL thermal leakage characterization to specify a list of characterization temperature samples, in degrees Centigrade.

Syntax:

```
SWEEP_TEMPERATURE <t1> <t2> ...
```

Example:

```
SWEET_TEMPERATURE 25 50 125
```

TEMP

Defines the temperature used for cell characterization. The value should be consistent with the value in the corresponding Synopsys .LIB and parasitic extraction (SPEF or DSPF) files.

Syntax:

```
TEMP <char_temp_in_centigrade>
```

Example:

```
TEMP -40
```

VDD

Defines the voltage supply value at the Vdd pin. The value should be the one specified in LIB nom_voltage.

Syntax:

```
VDD <voltage>
```

Example:

```
VDD 1.32
```

VDD_VALUES

Specifies the VDD values for multi-VDD case. User can explicitly specify voltage values for all domains. If both VDD_VALUES and VDD keywords are specified in config file, then VDD_VALUES will have higher preference. This keyword also honors user defined VDD values for leakage current characterization.

Syntax (for multi-VDD):

```
VDD_VALUES {
    <vdd_pin_name1> <value1> <value2> ...
    <vdd_pin_name2> <value1> <value2> ...
    ...
}
```

Syntax (for leakage current characterization):

```
VDD_VALUES {
    <vdd_pin_name1> <vdd value>
    <vdd_pin_name2> <vdd value>
    ...
}
```

VDD_PIN_NAME / GND_PIN_NAME

Specifies pin names for cells with single power/ground grids (multiple Vdd/Vss pins are specified in LEF). The Vdd names should match the power pin names found in the Spice subcircuit netlist. If more than one pin name is specified, names following the first one are considered alias pins. These names may be different than the design's power net names specified in the .gsr file.

Syntax:

```
VDD_PIN_NAME <vdd_pin_name> <vdd_alias_pin_name1> <...>
GND_PIN_NAME <gnd_pin_name> <gnd_alias_pin_name1> <...>
```

Example:

```
VDD_PIN_NAME VDD
GND_PIN_NAME VSS
```

Required for Library APL (Design-independent) Configuration File Only

The following is a list of keywords used in the configuration file for APL-DI standard cell characterization.

APL_RUN_PASS

APL_RUN_FAIL

These keywords - APL_RUN_PASS and APL_RUN_FAIL- can be used to dump spice netlist and output waveform file while doing current characterization.

Syntax:

```
APL_RUN_PASS [ NONE | DECK | ALL ]
APL_RUN_FAIL [ NONE | DECK | ALL ]
```

CUSTOM_LIBS_FILE

Typically, multiple Vdd/Vss pins are specified in LEF files. APL uses the nominal voltages defined in the .lib files and creates custom LIB files for characterization. If LEF files are not available, CUSTOM_LIBS_FILE can be used to specify one (only) manually-created custom LIB file needed to define the Vdd/Vss pins. APL-DI extracts the Vdd/Vss pins and puts them in the Pin section of the *adsLib.output* file.

Syntax:

```
CUSTOM_LIBS_FILE {
    <.lib_file>
}
```

Example:

```
CUSTOM_LIBS_FILE {
    /nfs/apl1/user_data/lib/ABC.lib
}
```

DESIGN_CORNER

Each design process corner (a combination of a temperature and voltage) generate one set of cell characteristics. Multiple design corners generate multiple sets of cell characteristics. There can be multiple LIB_ENTRY, SUBCKT, and LIB_FILES/DIR entries for each corner definition. If the TEMP, VDD, LIB_ENTRY, SUBCKT, or LIB_FILES/DIR is not defined for a corner, its common definition in the configuration file is applied to all corners. The PROCESS keyword defines the standard process corner. The <lib_type> parameter is a customized name that specifies which library type is used, which could be a standard name (i.e., Typical (TT), Fast (FF), and Slow (SS)), or a different name designated by the user.

Syntax:

```
DESIGN_CORNER {
    ? <corner_name1> ? {
        TEMPERATURE <value in Celsius>
        PROCESS [ FF | TT | SS | WC | BC | TC ]
        [LIB_ENTRY | MODEL | DEVICE_MODEL_LIBRARY] <lib_file> ?<lib_type>?
        [SUBCKT[_DIR] | SPICE_NETLIST[_DIR]]<subckt_file/dir_name>
        ? VDD <vdd value>?
        LIB_FILES <lib_file_name>
        ...
        ? LIB_FILES {
```

```

<File1>
...
<Dir1>
...
} ?
? Custom_lib_file <filename>
Custom_lib_file {
    <custom_file>
} ?
? LIBS_DIRECTORY <lib_dir_name> ?
...
}
? <corner_name2> {
    TEMPERATURE <value in Celsius>
    PROCESS [ FF | TT | SS | WC | BC | TC ]
    [LIB_ENTRY | MODEL | DEVICE_MODEL_LIBRARY ]<lib_file> ?<lib_type>?
    [SUBCKT[_DIR] | SPICE_NETLIST[_DIR]]<subckt_file/dir_name>
    ? VDD <vdd value>?
    LIB_FILES <lib_file_name>

    ? LIB_FILES
    {
        <File1>
        <File2>
        ...
        <Dir1>
        <Dir2>
        ...
    } ?
    Custom_lib_file <filename>
    Custom_lib_file {
        <custom_filename>
    ? LIBS_DIRECTORY <lib_dir_name> ?
    ...
    }?
}

```

Example:

```

DESIGN_CORNER {
    TT_25 {
        TEMP 25
        PROCESS TT
        VDD 1.0
        MODEL /nfs/apl1/model TT
    }
    FF_125 {
        TEMP 125
        PROCESS FF
        VDD 1.1
        MODEL /nfs/apl1/model FF
        LIB_FILES lib_dir1
        LIB_FILES lib_dir2/clkGen_fast.lib

```

```
}
```

```
}
```

LEF_FILES

RedHawk supports APL characterization for cells having multiple power and ground pins, as defined in the LEF files. APL captures the current/cap profiles for different power/ground pins separately. LEF_FILES defines the LEF files containing the P/G pin specifications. With no LEF pin specifications, you must manually define multiple Vdd/Vss pins using custom LIB files.

Syntax:

```
LEF_FILES {
    <lef_file_1>
    <lef_file_2>
    ...
}
```

Example:

```
LEF_FILES {
    design_data/lef/cella.lef
    design_data/lef/cellb.lef
}
```

If LEF files are not available while running APLDI, you can specify P/G pins for each multi-voltage cell in a custom LIB file. The format for the custom LIB file is as follows:

```
cell <cell_name> {
    pin <pin_name> {
        type <vdd | gnd>
    }
}
```

Example:

```
cell CELLA {
    pin VDDIN {
        type vdd
    }
    pin VDD {
        type vdd
    }
    pin VSS {
        type gnd
    }
}
```

Specify the custom LIB filename in the APLDI config file using the keyword 'CUSTOM_LIBS_FILE'.

LIB_FILES

Specifies Synopsys library files (*.lib) or a custom library P/G arc file to be used in the design (only one custom *.lib file may be specified). If a directory is specified, all files in the directory are selected.

Syntax:

```
LIB_FILES {
    [ <lib_filename> | <lib_file_dir> ] ? CUSTOM ?
```

```
...
}
```

where

<lib_filename> : specifies library or p/g arc filename

<lib_file_dir>: specifies library directory

CUSTOM: specifies one custom P/G arc file or directory

Examples:

```
LIB_FILES
{
    libs/special/custom.lib CUSTOM
    libs/typical
    libs/memory/mem.lib
}
LIB_FILES
{
    abc_lib
    libs/special/pgarc.lib CUSTOM
}
```

where

abc_lib: specifies the library name

libs/special/pgarc.lib CUSTOM: specifies a file that contains P/G arc definitions of the following form (see [section "P/G Arc Definitions in Custom LIB Files", page 3-20](#) for more details):

```
pgarc {
    <vdd_pin_name> <vss_pin_name>
    ...
}
```

Handling large device capacitances

APLDI relaxes the upper limit on cdev values for special cells, which changes ESC values for large cdev. Note that although large cdev values can be extracted successfully using this step, these values cannot be treated as normal cdev values by RedHawk and APL utilities, since they have checks on cdev values in place and import would fail. So to handle large cdev values, the following steps should be taken:

1. When running aplchk/aplreader/aplmerge, the option '-icheck' must be specified to ignore large cdev values. And when running APL characterization, do not specify '-o <outfile>' in APL, and do not put 'MERGE_RESULT 1' in the config file, as in those cases aplmerge would be launched (without the '-icheck' option) and cause characterization failure.
2. When importing large cdev values into RedHawk, set the GSR keyword 'IGNORE_APL_CHECK 1', otherwise it cannot run through the checking stage and APL data cannot be imported successfully.

Optional APL Configuration File Keywords

API_ELD0

Allows use of the ELDO simulator to perform switching current characterization, as well as cdev/pwcdev switch characterization. Eldo handles both switching current characterization and ACE, as well as cdev/pwcdev/switch characterization flow. To use the ELDO simulator for characterization, instead of the default NSPICE, the location of the binary must be specified with API_ELD0.

Syntax:

API_ELD0 <full_path_to_binary>

Example:

```
APL_ELD0 abc/sim/eldo
```

APL_ELD0_WDB

APL supports ELDO syntax natively, and also supports WDB-only modes for the APL-ELDO interface when this keyword is set (default is 0).

Syntax:

```
APL_ELD0_WDB [ 0 | 1 ]
```

APL_FORMAT_CORRECTION

When set to 1, aplcopy automatically identifies incorrect cell types based on the toggle rates for the cells, and corrects the cell types automatically. The default cell type is 'Gate' if the toggle value is two. Default is 0.

```
APL_FORMAT_CORRECTION [ 0 | 1 ]
```

APL_HSPICE

To use the HSPICE simulator for characterization, instead of the default, NSPICE, the location of the binary must be specified with APL_HSPICE.

Syntax:

```
APL_HSPICE <path_to_binary>
```

Example:

```
APL_HSPICE abc/sim/hspice
```

APL_SPECTRE

To use the Spectre simulator for characterization, instead of the default NSPICE, the location of the binary must be specified with APL_SPECTRE.

Syntax:

```
APL_SPECTRE <path_to_binary>
```

Example:

```
APL_SPECTRE abc/sim/spectre
```

APL_RESULT_DIRECTORY

APL_RESULT_DIRECTORY allows you to specify a directory for writing APL result files for individual cells, with filenames of the form

- for current, *<APL_dir_name>/<corner>/CURRENT/<cellname>.spiprof*,
- for decap, *<APL_dir_name>/<corner>/CAP/<cellname>.cdev*, and
- for pwcaps, *<APL_dir_name>/<corner>/PWC/<cellname>.pwcdev*,

as well as log file names of the form

- for current, *<APL_dir_name>/<corner>/CURRENT/<cellname>.current.<time>.log*
- for decap, *<APL_dir_name>/<corner>/CAP/<cellname>.cap.<time>.log*, and
- for pwcaps, *<APL_dir_name>/<corner>/PWC/<cellname>.pwcdev.<time>.log*

where *<time>* is a timestamp of the form *<date-time>*.

The default directory for the output and log files is *APLDD_<time>/* or *APLDI_<time>/*. To specify a single file with all output result files merged, use the '*-o <output_filename>*' option during APL invocation.

Syntax:

```
APL_RESULT_DIRECTORY <APL_dir_name>
Example:
APL_RESULT_DIRECTORY Apl-out-ABC
```

APL_SAMPLE_MODE

Selects the type of APL characterization to be performed, based on the accuracy desired. The number of samples is automatically selected based on the range of voltage, load and slew values in the library. Default mode is 1 (DEFAULT). (Previously called APLDI_SAMPLE.)

Syntax:

```
APL_SAMPLE_MODE [ FAST_CHECK | DEFAULT ]
```

where

FAST_CHECK: specifies fast library checking, uses one voltage/load/slew sample per cell to perform basic library data integrity (completeness) checks

DEFAULT: specifies default characterization mode with multiple samples per cell, depending on the range of voltage, load and slew values in the library.

Example:

```
APL_SAMPLE_MODE FAST_CHECK
```

APL_TAIL_REDUCTION

By default, APL uses an equal time step waveform to save switching current. Due to the intrinsic limitation of equal time step waveforms, there is a trade-off between waveform accuracy and file size. To reduce the number of APL waveform time points and data size, there is a back-trace process in APL after current extraction to cut part of the tail current waveform when the tail is long, but not reduce sharply down to its leakage value, which is typically in the nA range. This 'moderate' mode makes the tail reduction process more stable and less sensitive to time step changes in simulation results.

Syntax:

```
APL_TAIL_REDUCTION [ 0 | 1 | moderate ]
```

where

0: turns off waveform tail reduction process (default)

1: turns on original waveform tail reduction process

'moderate' : turns on improved waveform tail reduction (works with 'EXTRACTION_METHOD pwl' keyword)

APL_TECH_LEVEL

When set, switching current characterization is performed using PWL input waveforms, providing improved accuracy and support for custom vector flow. APL characterization is speeded up by reducing number of APLDI samples, which also reduces the disk file size requirement by 50%. Note that this feature is available only for NSpice and HSpice interfaces.

Example:

```
APL_TECH_LEVEL [ 0 | 1 ]
```

APL_VOLTAGES

For design-independent libraries, APL_VOLTAGES specifies characterization voltage values as a fraction of LIB nominal voltage, which are used for creating the current profiles. Voltage fractions should be specified in either ascending or descending order.

If no voltage fraction at least 1.15 times nominal is specified, APLDI automatically adds an additional voltage point equal to the highest among the VDD values in the config file, or 1.15 times the Lib_nom_voltage. Default: APL_VOLTAGES 4 1.15 1.0 0.9 0.75

Syntax:

```
APL_VOLTAGES <Num_volt_samples> <fraction 1> ... <fraction N>
```

Example:

```
APL_VOLTAGES 4 0.8 0.9 1.0 1.2
```

which creates current profiles for following four voltage values:

0.8 x Nom_V, 0.9 x Nom_V, 1.0 x Nom_V, 1.2 x Nom_V

CDEV_AC_OPEN_INPUT

When set, cdev characterization *excludes* the capacitance of all input pins from cell intrinsic capacitance. The default value is 0 (off).

Syntax:

```
CDEV_AC_OPEN_INPUT [ 0 | 1 ]
```

CELL_PROPERTY_FILE

Specifies a file that provides a flexible interface for you to specify the output load for each cell. The format of the cell property file contents is:

```
cell <cell name> {
    pin <pin name> {
        load <load value>
    }
    ...
}
```

Syntax:

```
CELL_PROPERTY <cell property filename>
```

CONVERT_CCS_CAP

When set, converts CCS libraries that contain intrinsic parasitic ESC/ESR elements and leakage current into APL cdev data. Default: 0.

Syntax:

```
CONVERT_CCS_CAP [ 0 | 1 ]
```

DC_TABLE_THRESHOLD

Enables LDO dc table to have reduced data points when user specifies very fine sweep step in spice dc simulation

Syntax:

```
DC_TABLE_THRESHOLD <slope_threshold>
```

Example:

```
DC_TABLE_THRESHOLD      0.08
```

DEBUG

DEBUG sets the debug flag, which saves intermediate and error files for debugging and analysis.

Syntax:

```
DEBUG [ 0 | 1 ]
```

DESIGN_CORNER

Specifies voltages to be defined in the APL leak characterization output file (*.leak) in the thermal flow.

Syntax:

```
DESIGN_CORNER { <corner_name> { VDD <voltage>} }
```

Example:

```
DESIGN_CORNER {  
    Corner1 {  
        PROCESS TT  
        VDD 0.8  
        TEMP 125  
    }  
}
```

In this case the voltage dumped in the *.leak file is 0.8V, although the cell leak characterization is done at the NOMINAL_VOLTAGE derived from the .lib file.

ELDO_VECTOR_MODE

Selects between two different ELDO vector handling modes in APL extraction. By default APL treats ELDO vectors as HSpice-compatible, in which the time point in digital vectors are their initial values. When this keyword is set to 1, ELDO is in native digital vector handling mode, and treats the time point in digital vectors as their final voltage values.

Syntax:

```
ELDO_VECTOR_MODE [ 0 | 1 ]
```

EXTRACTION_METHOD

Allows changing the APL waveform extraction method to PWL (piecewise linear). The default APL waveform model using equal time steps can result in a very large file size for very rapidly changing waveforms. The pwl extraction method retains nearly the accuracy of the original Spice waveform, but the file size increase is negligible. Default: 0.

Syntax:

```
EXTRACTION_METHOD [ 0 | pwl ]
```

FAST_CHECKING

If FAST_CHECKING is set to 1, a data check is performed on each cell in the library at one sample condition. A list of cells that cannot be characterized properly is created, the same as using the ‘-fc’ option in apldi, or setting the GSR keyword ‘APLDI_SAMPLE_MODE’ to ‘fast_check’. The default is 0 (regular APL characterization).

Syntax:

```
FAST_CHECKING [ 1 | 0 ]
```

GZIP_RESULT

When set, dumps a compressed current file (default is 0).

```
GZIP_RESULT [ 0 | 1 ]
```

IGNORE_DC_CHECK

When turned On, APL ignores input pin checking for user-specified DC bias; the global DC bias can be used for the whole library characterization. This helps in situations in which you must use a single configuration file for cells with and without a particular DC bias pins in the netlist. If not set, APL characterization errors out, so you must create two configuration files and split the cell list. The default is 0.

`IGNORE_DC_CHECK [0 | 1]`

IGNORE_NETLIST_CHECK

Supports P/G pin pre-checking before APL characterization. If set to 0, APL checks the netlist for pin mismatches and proceeds only if the P/G pin names specified in the configuration file match those in the netlist (default 1).

Syntax:

`IGNORE_NETLIST_CHECK [1 | 0]`

IGNORE_RESET_PIN

When set, if defined Set/Reset pins are not found for a cell, RedHawk displays a Warning and continues. By default (off), pins that are defined but do not exist generate an Error out condition.

Syntax:

`IGNORE_RESET_PIN [0 | 1]`

IGNORE_UPF_PGARC

When set, if the UPF lib does not have p/g pins, pg arc information is defined using the custom lib file (default 0).

`IGNORE_UPF_PGARC [0 | 1]`

INCREMENTAL_APL

If characterization for some cells fail or several new cells added, INCREMENTAL_APL can be set to 1 and only the failed or new cells are characterized. After fixing the problem with the failed cells and executing the last command prior to the failure, APL identifies and re-characterizes the failed cells if INCREMENTAL_APL is set to 1. The default value is 0, which is normal characterization of all cells. Note that when an incremental run is executed, the APL_RESULT_DIRECTORY must be specified in the APL config file.

Syntax:

`INCREMENTAL_APL [0 | 1]`

INPUT_TYPE

Allows using a more realistic input stimulus waveform model than the preset stimulus waveform used for characterization by default. Especially for new technologies, the threshold interval (linear segment) becomes shorter and shorter. Therefore, this keyword creates a stimulus waveform that minimizes the error introduced by this.

Syntax:

`INPUT_TYPE [0 | 2 | 3 | 4]`

where

0 : 1-segment ramp (default)

1 : 5-segment PWL

2 : ramp+RC filter

3 : 5-segment PWL+RC filter

4 : 50-segment PWL (recommended for accuracy)

LEAKAGE_DEVICE_MODEL_LIBRARY

Provides a separate Spice device model file for leakage current characterization.

Syntax:

```
LEAKAGE_DEVICE_MODEL_LIBRARY <devMod_file> ?<proc_corner>?
```

LEAK_HIGH_TO_MID_RATIO_LIMIT

LEAK_HIGH_TO_LOW_RATIO_LIMIT

For pwcaps files, these keywords specify leakage current limits for the aplchk command. LEAK_HIGH_TO_MID_RATIO_LIMIT specifies the ratio of current leakage at the highest Vdd value (1.1xV) divided by the current leakage at the mid-voltage value (0.5xV). Those values less than the ratio are not reported in the log. Keyword LEAK_HIGH_TO_LOW_RATIO_LIMIT specifies the ratio of current leakage at the highest Vdd value (1.1xV) divided by the current leakage at the lowest Vdd value (0.1xV). Those values less than the ratio are not reported in the log. Violation warnings are recorded in *adsRpt/aplchk.log*. There are no default ratios; if no value is specified, the comparison is not computed.

Syntax:

```
LEAK_HIGH_TO_MID_RATIO_LIMIT <ratio>
LEAK_HIGH_TO_LOW_RATIO_LIMIT <ratio>
```

MAX_LOAD_SAMPLE

MIN_LOAD_SAMPLE

Sets minimum and maximum values for capacitive load (in Farads) used in characterization, which may be either a wider or narrow range of values than those in LIB files. Note that these values override the input range settings in the WARN_CLOAD_CHECK and ERROR_CLOAD_CHECK parameters. No default.

Syntax:

```
MIN_LOAD_SAMPLE <load_Farads>
MAX_LOAD_SAMPLE <load_Farads>
```

Example:

```
MIN_LOAD_SAMPLE 10e-15
MAX_LOAD_SAMPLE 500e-15
```

MAX_SLEW_SAMPLE

MIN_SLEW_SAMPLE

Sets minimum and maximum values for transition times (in seconds) used in characterization, which may be either a wider or narrow range of values than those in LIB files. Note that these values override the input range settings in the WARN_SLEW_CHECK and ERROR_SLEW_CHECK parameters. No default.

Syntax:

```
MIN_SLEW_SAMPLE <trans_time-sec>
MAX_SLEW_SAMPLE <trans_time-sec>
```

Example:

```
MIN_SLEW_SAMPLE 10e-12
MAX_SLEW_SAMPLE 500e-12
```

MEMORYFILE | INPVECFILE

MEMORYFILE and INPVECFILE specify the input vector file for characterizing a memory cell or a special custom cell.

Syntax:

```
[ MEMORYFILE | INPVECFILE ] </nfs/filename>
```

Example:

```
MEMORYFILE /nfs/apl1/memfile
```

MERGE_RESULT

APL now generates characterization results in separate files in the APL results directory by default. If MERGE_RESULT is set to 1, the result files are merged into a single file in the run directory.

Also, the results are merged if the command ‘apldi -o <filename>’ is executed on the command line. Files should merged only for APL design dependent (DD) runs, or one corner condition design independent (DI) runs.

Syntax:

```
MERGE_RESULT [ 0 | 1 ]
```

MULTI_CORE

Allows several APL jobs to be run in parallel on a multiple-CPU local machine. The default is '1', which means multi-core capability is turned On and APL submits multiple jobs, depending on the number of CPUs and the memory available. If set to '0', multi-core is turned off and only one job is submitted.

Syntax:

```
MULTI_CORE [ 0 | 1 ]
```

MULTI_NOMINAL

Leakage current for a cell changes with different supply voltage, which can be accommodated in APL. MULTI_NOMINAL should be turned on for multi-voltage cdev/ leakage characterization to capture leakage current at every voltage point. The generated cdev file can be read by RedHawk, and PowerStream computes the variable leakage power depending on the voltages of the cell.

Syntax:

```
MULTI_NOMINAL [ 0 | 1 ]
```

MULTI_NOMINAL_VOLTAGES_METHOD

When set, APL supports multiple nominal voltage characterization, and uses all of them to do scaling. In low power designs, one cell may operate under multiple nominal voltages (voltage islands). This keyword specifies the voltages to be used in characterization for such multi-nominal voltage cells.

In default mode (0), only the first nominal voltage in .LIB is used for voltage derating. This method ignores other nominal voltage values, making the characterization less accurate under the missed nominal voltages. When set to 1, APL honors all nominal voltages from .LIB, and combines the derated voltages with the nominal voltages to form the final voltage list for characterization. In this way, all nominal voltages and also their derated values are included in characterization. Also, if a cell has multi-nominal voltage as well as multiple Vdd domains, the same derating factor is applied for all multiple-nominal voltages across all Vdd domains.

Syntax:

```
MULTI_NOMINAL_VOLTAGES_METHOD [ 0 | 1 ]
```

MULTI_SPICE

APL supports a multi-level hierarchical netlist by splitting the Spice run into different Vdd values, which automatically adjusts the VIH values in the input vector file into several values that are percentages of the nominal VDD value. For each Vdd value, APL generates one Spice deck as *<cellname>_apl#.sp*, where '#' is a serial index number. Custom vector cells also have a different vector file for each Vdd value, and the VIH values also are appropriate percentages of the nominal Vdd value, as determined by APL. This feature is turned off by default to minimize run-time (on a single-CPU machine).

Syntax:

```
MULTI_SPICE [ 0 | 1 ]
```

OPENPIN

When an input or output pin has no defined connection or function, the OPENPIN keyword can be used to identify this condition and allow characterization. To provide a circuit connection for characterization, APL assigns a very large resistance to ground for specified open pins.

Syntax:

```
OPENPIN <pinA> <pinB> ...
```

Example:

```
OPENPIN ABCpin3A MNOpin14BC
```

OPTION

OPTION specifies any needed SPICE simulation options.

Syntax:

```
OPTION <Spice option1>
OPTION <Spice option2>
...
or
OPTION {
    <Spice option1>
    <Spice option2>
    ...
}
```

Example:

```
OPTION mode=turbo
```

OUTPUT_STATE_CHECK

When set, improves APL cdev characterization that uses STATE_TABLE for vector generation from the .LIB file, in order to better identify HIGH and LOW states in cdev generation. Default 0.

Syntax:

```
OUTPUT_STATE_CHECK [ 0 | 1 ]
```

PIN_LIST

PIN_LIST specifies the pin names and associated tie pins.

Syntax:

```
PIN_LIST {
    <pin_name> <tie_pin>
    ...
}
```

where

`pin_name`: specifies the name of the pin

`tie_pin`: specifies the name of the pin to which `<pin_name>` is connected.

Example:

```
PIN_LIST {
    A1 VDD1
}
```

In this case, A1 is connected to VDD1.

PRIMARY_GND_PIN

When there are multiple ground pins in a cell, PRIMARY_GND_PIN defines which ground pin is used to connect the output load. Although all ground pins are declared in the `.lib` file, there is no property to specify the output load connectivity. There could be multiple ground pin scenarios, in which there are both internal and external ground pins, connected by footer switches, or there could be core and I/O ground pins. The PRIMARY_GND_PIN is the closest pin to the output load; as such it carries the most discharging current, and can be a Spice ground name (for multi-rail cells).

Syntax:

```
PRIMARY_GND_PIN <primary_gnd_pin>
```

Example:

```
PRIMARY_GND_PIN VSS0
```

PWCAP_MULTI_SPICE

When set, improves pwcap characterization for runs that include many SPICE simulations for 10%, 20%... of nominal VDD value, which otherwise are sequentially performed and could take substantial run time. To reduce APL run-time, specify a `<parallel run count>` value to allow SPICE simulations with different VDD values to be performed simultaneously. Note that PWCAP_MULTI_SPICE applies only to multi-core/CPU runs, and not to LSF. This feature has no impact on accuracy of results.

Syntax:

```
PWCAP_MULTI_SPICE <parallel run count>
```

RUN_TIME_LIMIT

Allows specification of a limit on LSF and Sun Grid Engine (SGE) processes. The RUN_TIME_LIMIT ensures that APLDI continues at the end of the specified run time limit, even through processing errors.

Syntax:

```
RUN_TIME_LIMIT <limit_hrs>
```

SAMPLE_SPLIT

When turned on, SAMPLE_SPLIT distributes characterization jobs sample-by-sample to different machines to speed up characterization and avoid negative network impacts. When the keyword is unset (default), APL internally determines cell-by-cell, based on how large the cell netlist is, whether the sample split mode is used. When set to '0', no

cells are submitted using sample split mode, and when set to '1', all cells large or small are submitted using sample split mode (a single sample may be submitted). To avoid unnecessary management overhead, turning on SAMPLE_SPLIT is not recommended for cells with small netlists. SAMPLE_SPLIT is not supported for multi-corner runs.

Note that the keyword LSF_RUN_TIME_LIMIT is available for both 'SAMPLE_SPLIT 1' in the SGE environment and also for 'SAMPLE_SPLIT 0'.

Syntax:

```
SAMPLE_SPLIT [ 0 | 1 ]
```

SCANMODE

SCANMODE turns scan mode characterization On or Off for flip-flops and other cells with scan logic. Default is off (0).

Syntax:

```
SCANMODE [ 0 | 1 ]
```

SCAN_OUTPUT_LOAD

Allows specifying the attachment of a constant load for various samples for APL characterization. If specified, APL attaches the specified constant load to the scan pin of the cell, if present. The libreader automatically identifies the scan pin. Default: none.

Syntax:

```
SCAN_OUTPUT_LOAD <load>
```

SIM_TIME_SCALE

Allows improvement in run time for Spice transient analysis by scaling the pulse width/period and tstop in SPICE transient simulation in APL cdev/leakage characterization. For example, when 'SIM_TIME_SCALE 0.1' is set, the pulse width/period and tstop are reduced to 0.1x, then the SPICE simulation run-time would be reduced accordingly. Note that by default the keyword value is 1.0 for the cdev flow. For temperature-based leakage characterization, the default value is 0.1.

Syntax:

```
SIM_TIME_SCALE <value>
```

SIMULATION_COMMAND

The keyword SIMULATION_COMMAND should be used for any situation in which you have your own wrapper to submit APL simulation jobs through LSF.

Syntax:

```
SIMULATION_COMMAND <sim_command>
```

Example:

```
SIMULATION_COMMAND bsub -q normal -q linux -Ip -R
" select [type==LINUX64 && mem> 8000 ] rusage[hsim=1:duration=2]
  " hsim -i $input_file -o $output_file
```

In this example, \$input_file and \$output_file are replaced by files with the file names required by APL. The \$input_file specification is required, while \$output_file is optional. The simulator type is still determined by the existing keyword, such as 'APL_HSIM ~/bin/hsim' for using HSIM.

SIMULATOR_COMMAND_OPTION

Supports the efficient use of NSpice, HSpice, Eldo, and Spectre tools. Adding this keyword appends the specified user options to the Spice simulation command line.

Syntax:

```
SIMULATOR_COMMAND_OPTION <option>
```

SMOOTH_PWC_LEAK

To smooth the leakage value glitches for lower voltage samples for APL pwdev characterization. *Default: 0*

Syntax:

```
SMOOTH_PWC_LEAK [ 1 | 0 ]
```

SPICE_SUBCKT_DIR

The SPICE_SUBCKT_DIR option specifies the directory path containing the Spice subcircuit files for the library cells to be characterized. APLDI automatically reads the corresponding Spice subckt file while characterizing a particular cell.

Under the <directory_path> specification each cell has its own Spice subckt file. The suffix of the subckt filename can be anything, but the <cell name> has to be the cell name (can be case-insensitive), followed by a period “.”. Subckt files should not contain any other subckt files.

Syntax:

```
SPICE_SUBCKT_DIR <directory path>
```

Example:

```
SPICE_SUBCKT_DIR ./spice_netlists
```

Note: The SPICE_SUBCKT_DIR and SPICE_NETLIST keywords are mutually exclusive. The first one specified is processed and the second one is ignored. Except for hierarchical netlists, use SPICE_SUBCKT_DIR to specify subcircuit files for best processing performance.

SPICE2LEF_PIN_MAPPING

To handle P/G pins with different names in SPICE and LEF, APL uses Spice pin names internally, but outputs LEF pin names. SPICE2LEF_PIN_MAPPING applies to all cells. If no pin mapping is given in the APL configuration file, P/G pin names output to LEF are the same as in Spice. Mapping is allowed *only* between one Spice pin to one LEF pin or one Spice pin to multiple LEF pins. SPICE2LEF_PIN_MAPPING supports switching currents, CDEV, and PWCAP data.

Syntax:

```
SPICE2LEF_PIN_MAPPING {
    <Spice_pin_name> <LEF_pin_name>
    ...
}
```

Example:

```
SPICE2LEF_PIN_MAPPING {
    Vdd_spicel Vdd_lef1A
    Vss_spicel Vss_ref1B
    ...
    Vss_spiceN gnd_lefM
}
```

SUPPRESS_INTERNAL_OPTION

When set, provides an interface to control simulation convergence. That is, APL does not specify convergence-related options (such as method=gear) for SPICE simulation. You may specify other options with the keyword OPTION. Default is 0.

```
SUPPRESS_INTERNAL_OPTION [ 0 | 1 ]
```

SWEEPSOURCE

For header and footer switches in low power designs, the associated power or ground pin name can be defined for APL characterization. If SWEEPSOURCE is not specified, the name is set by keywords PRIMARY_VDD_PIN or PRIMARY_GND_PIN, if either one is specified. If none of them is specified, the default is Vdd.

Syntax:

```
SWEEPSOURCE [<power_pin_for_header> | <gnd_pin_for_footer> ]
```

Example:

```
SWEEPSOURCE vdd1
```

SWEEPVALUE

Specifies the effective voltage values (Vdd - Vss) to be used for characterization instead of the default values. The effective Vdd values must be larger than 0 and less than '10 * Vdd'. There are eleven default values used for characterization: 1.1*Veff, Veff, 0.9*Veff, 0.8*Veff, 0.7*Veff, 0.6*Veff, 0.5*Veff, 0.4*Veff, 0.3*Veff, 0.2*Veff, and 0.1*Veff .

Syntax:

```
SWEEPVALUE <eff_vdd_value1> <eff_vdd_value2> ...
```

Example:

```
SWEEPSOURCE vdd
SWEEPVALUE 1.0 0.9 0.8 0.7
```

The output is a list of device capacitance values ("cdev") at Vdd values of 1.0, 0.9, 0.8, and 0.7 volts.

Example:

```
SWEEPSOURCE vss
SWEEPVALUE 1.0 0.9 0.8 0.7
VDD 1.0
```

The output is a list of cdev values at Vss values of 0, 0.1, 0.2, and 0.3 volts.

SWITCH_PRE_DRIVER

Switch pre-drivers are devices inside switch cells that drive the switch transistors. You can characterize switch pre-driver ESR and ESC values for power/ground arcs by using APLSW with this keyword. The DC pin specification must make the output of the pre-driver '1' for header switches and '0' for footer switches, so the switch is in its Off state. The results of pre-driver characterization are saved in the following format:

```
CDEV: <arc-based_cap_F> <arc-based_R_ohms>
```

Syntax:

```
SWITCH_PRE_DRIVER {
    VDD_PIN_NAME <VDD_pinname_pre-driver>
    VSS_PIN_NAME <VSS_pinname_pre-driver>
    ? DC <input_pinname_pre-driver> <voltage_dc_pin> ?
    ...
}
```

Example:

```
SWITCH_PRE_DRIVER {
    VDD_PIN_NAME VDD_EXT
    GND_PIN_NAME VSS
    DC POWERUP_CNTL 1.08
    DC POWERON_CNTL 1.08
}
```

TMP_DIR

Defines the directory used for writing out temporary files.

Syntax:

```
TMP_DIR <directory_path>
```

Example:

```
TMP_DIR /home/tmp
```

WARN_CLOAD_CHECK

ERROR_CLOAD_CHECK

These options set limits for pre-checking load capacitance values from *.lib files. Default values are shown in the syntax below. Units: picoFarads.

Syntax:

```
WARN_CLOAD_CHECK {
    S <std_cell_warn_val> # Default: 5
    M <mem_cell_warn_val> # Default: 50
    I <I/O_cell_warn_val> # Default: 50
}
ERROR_CLOAD_CHECK {
    S <std_cell_error_val> # Default: 50
    M <mem_cell_error_val> # Default: 500
    I <I/O_cell_error_val> # Default: 500
}
```

Example:

```
WARN_CLOAD_CHECK {
    S 10
    M 20
    I 30
}
ERROR_CLOAD_CHECK {
    S 75
    M 300
    I 300
}
```

WARN_SLEW_CHECK

ERROR_SLEW_CHECK

These options set limits for pre-checking input transition time values from *.lib files. Default values are listed in the syntax below. Units: nsec.

Syntax:

```
WARN_SLEW_CHECK {
    S <std_cell_warn_val> # Default: 5
```

```
M <mem_cell_warn_val> # Default: 10
I <I/O_cell_warn_val> # Default: 10
}
ERROR_SLEW_CHECK <slew_error_val> {
    S <std_cell_error_val> # Default: 50
    M <mem_cell_error_val> # Default: 100
    I <I/O_cell_error_val> # Default: 100
}
Example:
WARN_SLEW_CHECK {
    S 2
    M 7
    I 20
}
ERROR_SLEW_CHECK {
    S 30
    M 70
    I 80
}
```

WARN_VDD_CHECK

ERROR_VDD_CHECK

MIN_VDD_CHECK

These options set limits for pre-checking nominal Vdd values from .lib files. Default value for a Warning notice is above 5V, and for an Error notice is above 10V, or below 0.5V minimum value.

Syntax:

```
WARN_VDD_CHECK <Vdd_warn_val>
ERROR_VDD_CHECK <Vdd_high_error_val>
MIN_VDD_CHECK <Vdd_low_error_val>
```

Example:

```
WARN_VDD_CHECK 3
ERROR_VDD_CHECK 5
MIN_VDD_CHECK 0.7
```

Parallel Run Keywords

The following ten keywords are optional for parallel runs on Platform LSF or Sun Grid, and can be defined in the APL configuration file:

BATCH_QUEUEING_COMMAND

BATCH_QUEUEING_COMMAND specifies the queue command. The default is 'bsub' for GRID_TYPE LSF. If the GRID_TYPE is SUN_GRID, the default queuing command is 'qsub'.

Syntax:

```
BATCH_QUEUEING_COMMAND <command_name>
```

Example:

```
BATCH_QUEUEING_COMMAND qsub
```

BATCH_QUEUEING_OPTIONS

BATCH_QUEUEING_OPTIONS specify run options specific to Platform LSF or Sun Grid software.

Note: only one BATCH_QUEUEING_OPTIONS keyword can appear in the file and all options must be listed in a continuous line.

Syntax:

```
BATCH_QUEUEING_OPTIONS <options>
```

Example:

```
BATCH_QUEUEING_OPTIONS -P mary -cwd -b y -p -50 -j y -shell n
                           -hard -l wrapper=TRUE -V -l aplchar=1 -l arch=lx24-amd64
                           (for a Sungrid submittal) -rerun <num_times_auto_rerun>
```

EXEC_PATH

EXEC_PATH specifies the path to the Platform LSF or Sun Grid binaries.

Syntax:

```
EXEC_PATH <path name>
```

Example:

```
EXEC_PATH /appls/lsf/6.0/linux2.4-glibc2.3-x86/bin
```

GRID_TYPE

GRID_TYPE defines the parallel run platform. The default is LSF.

```
GRID_TYPE <LSF | SUN_GRID>
```

Example:

```
GRID_TYPE LSF
```

JOB_COUNT | LSF_JOB_COUNT

JOB_COUNT | LSF_JOB_COUNT specifies the maximum number of simultaneous characterization jobs that can be submitted (one job per cell). Default is 10. The maximum is 100 for Sun Grid.

Syntax:

```
[ JOB_COUNT | LSF_JOB_COUNT ] <max number of jobs >
```

Example:

```
JOB_COUNT 20
```

LSF_ARRAY_SUBMIT_LIMIT

Specifies array submission limit. The job array size that apldi2 uses for first time submit is based on this keyword. Default: 1000.

Syntax:

```
LSF_ARRAY_SUBMIT_LIMIT <value>
```

LSF_JOB_TIME_OUT

APL jobs running under the Platform LSF batch program that are suspended by the system are killed after a suspended period specified by LSF_JOB_TIME_OUT, which has a default value of 1 hour. The killed jobs are resubmitted one more time to the LSF farm.

APL also kills jobs that hang because of program or data problems if they take longer than 12 hours.

Syntax

```
LSF_JOB_TIME_OUT <hours>
```

Example:

```
LSF_JOB_TIME_OUT 2
```

Before starting *apldi* or *avm* characterization, set the environment variable APACHEROOT in the *csh* environment, as follows.

```
setenv APACHEROOT <path_to_redhawk_release_directory>
```

LSF_SUBMIT_MODE

LSF_SUBMIT_MODE specifies the Platform LSF job submission mode. 1= bsub, 2= LSF API. Both modes do the same batch submission. Which one to use depends somewhat on the user environment. Often LSF API mode 2 runs successfully when bsub mode 1 does not. The default is 1 (bsub).

Syntax:

```
LSF_SUBMIT_MODE [ 1 | 2 ]
```

Example:

```
LSF_SUBMIT_MODE 2
```

QUEUE

QUEUE allows you to specify the queues for running APL jobs.

Syntax:

```
QUEUE <queue name> ...
```

Example:

```
QUEUE short
```

TIMER

TIMER specifies the time in seconds between checks on the status of the jobs submitted. The default value is 60 seconds.

Syntax:

```
TIMER <value>
```

Example:

```
TIMER 10
```

Custom Cell Characterization Data Preparation

APL can be used to characterize custom cells, including combinational and sequential cells. For custom cells, you must specify the vectors used for cell characterization and therefore it requires several user input steps.

Custom cell characterization requires the same data as standard cell characterization, with the addition of an input vector file. An input vector file must exist for every cell that needs to be characterized. The configuration file must contain the keyword VECTOR_DIR to specify the location of the input vector files.

VECTOR_DIR

Specifies the location of the input vector files. The path to the input vector file must be a full path.

Syntax:

```
VECTOR_DIR <vector_dirPath>
```

Example:

```
VECTOR_DIR /home/design/input_vectors
```

Input Vector Files

In APL-DI mode APL calls the *libreader* program to create the vector file. In APL-DD mode RedHawk calls *libreader* to create the vector file. For custom cells you must create the vector file.

The following parameters are specified in the input vector file.

- Input vectors to be used.
- Pins whose timing arcs determine the delay of the cell.
- Input bias values for intrinsic decoupling capacitance and leakage current estimation.

The name of the input vector file must be *<cell_name>.inv* and must be created for every cell that needs to be characterized. The following is a list of keywords used in the input vector file for custom cell characterization.

- Define the DC bias level for inputs. The name of the dc bias level should be the same as the Vdd/Vss pin names specified in the APL configuration file. The DC bias level may also be set by specifying the voltage value or using the voltage value defined in the param statement, as shown below.

Note: Be careful with the voltage spec, to avoid a mismatch with the APL configuration file.

Syntax:

```
param <bias_level> <value>
...
```

```
dc <pin1> <bias_level>
dc <pin2> <bias_level>
...
```

- Specify the primary inputs and outputs. The primary-input-to-primary-output path defines the primary timing arc that determines the delay for that cell.

Syntax:

```
active_input <input_pin1> <input_pin2>
<...>
active_output <output_pin>
```

- Define input vectors.

Syntax:

```
vector {
  vname <input1> <input2> <...>
  tunit ps
  vih [ <Vdd_name> | <VIH_value> ]
  ...
  <time_step> <input_state_Pin1><input_state_Pin2><...> <state_name>
}
```

where

vname <input1>... : specifies the input pins, which must be listed in the same order as the input states.

tunit: defines the time unit.

vih [<Vdd_name> | <VIH_value>]: if Vdd_name is specified, and the APL config file keyword MULTI_SPICE is set to 1, the VIH value is set to a percentage of the specified nominal Vdd value. If a VIH_value is given, the high-state voltage for the specified input pin does not change.

<time_step>: defines multiples of APL-generated unit time step, which depends on slew, time unit, and slew threshold values. Note that the value is not in actual time (i.e., ns); APL scales the time appropriately. A time_step_num 0 is required for initialization. Subsequent time step values denote when the inputs change to toggle the output.

<input_state_Pinx>: defines the input state for Pinx at each time step.
Note: No spaces between vector state values.

<state_name>: user-specified name for state at time step

The following is an example of an input multiple vector file for a five-input combinational gate. For multiple vector definitions, the tunit and time steps must be the same for all.

```
dc c vdd
dc d 0
active_input a
active_output y
vector {
vname a b
tunit ps
vih Vdd1
0 10
1 01
5 10
}
vector {
vname e
tunit ps
vih Vdd2
0 1
1 0
5 1
}
```

Note that input pins ‘a’ and ‘b’ have a different power domain (‘vdd1’) than pin ‘e’ (‘vdd2’). For combinational logic, a sufficient number of input vector toggles should be specified to ensure a 0-to-1 and 1-to-0 transition at the output. The following shows the input states for the above input vector definition. The signal profiles generated for the defined vectors are shown in Figure 9-2.

- At time step 0, Pin A = 1, Pin B =0
- At time step 1, Pin A = 0, Pin B =1
- At time step 5, Pin A = 1, Pin B =0

This ensures that the output toggles from 0 to 1 and from 1 to 0. The current profiles for these two output transitions capture the behavior of a combinational cell during dynamic simulation.

For a sequential cell, a sufficient number of clock toggles and input data pin toggles must be specified to ensure that the following output states are captured.

- output toggling from 0 to 1, TRAN01

- output toggling from 1 to 0, TRAN10
- clock triggering edge with non-toggle output, TRAN11
- clock non-triggering edge with non-toggle output, TRAN00



Figure 9-2 Signal profiles generated for a two-input combination cell.

The following is an example of an input vector file for a sequential cell:

```

dc scen 0
dc scin 0

active_input clock
active_output nq

vector {
vname clock d
tunit ns
vih 1.2
0 00
2 10
4 01
6 11 tran10
8 01
10 11
12 01
14 10 tran01
16 00 tran00
18 10 tran11
20 00
}

```

In this example the “tranxx” names are user-assigned state names for the switching conditions.

The signal profiles generated for the defined vectors are shown in Figure 9-3

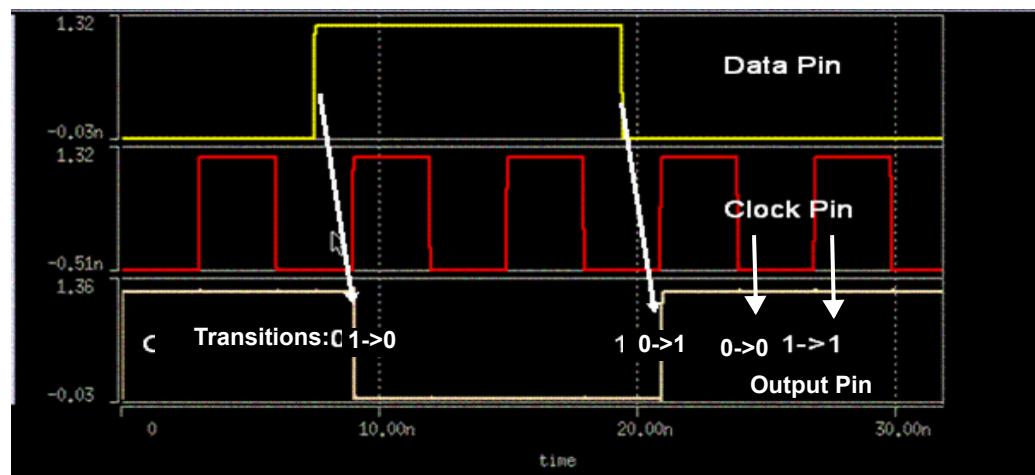


Figure 9-3 Signal profiles generated for a sequential cell.

In Figure 9-3, transitions ‘0->0’ means a non-triggering clock edge, ‘1->1’ means a triggering clock edge, and neither causes the output to toggle (output stays at 1 or 0).

NOTE: All input vector files, `<cell_name-n>.inv`, must reside in the directory specified by the `VECTOR_DIR` keyword in the configuration file.

Running Cell Characterization

Setup for a Design-independent (Library-based) APL Run

1. Prepare the APL configuration file defining characterization conditions desired, as described previously in [section "APL Configuration File Description", page 9-218](#).
2. Run APL -DI to obtain switching current profiles, device capacitance and resistance for cell high and low states, and leakage current for all library cells.

Setup for a Design-dependent APL Run

For design-specific characterization:

1. Prepare the GSR file for running **RedHawk**. Refer to Appendix C, “File Definitions”.
2. Select the GUI command **APL -> Setup** in **RedHawk**, or use the TCL command
`setup apl -dir <dir_name>`

This generates two intermediate files required for APL characterization, `apache.apl` and `adsLib.output`, and places them in the `<dir_name>/apache` directory.

In addition, four types of APL configuration file templates are created in the specified `<dir_name>` directory, to assist in preparing the proper configuration files for characterization:

- `apl.custom.config`
- `apl.io.config`
- `apl.memory.config`
- `apl.std.config`

- Cell list files for all four types of cells are also placed in the `<dir_name>` directory.
3. Prepare the APL configuration file defining characterization conditions desired, as described previously in [section "APL Configuration File Description", page 9-218](#).
 4. Run APL-DD to obtain switching current profiles, device capacitance and resistance for cell high and low states, and leakage current for all design cells.
 5. For low power designs with power-up cycles, run APL additionally for header and footer components and for decap (pwcaps). See [section "Low Power Design Characterization", page 9-249](#).

Setup for Enhanced Design-Independent Characterization

To automatically generate additional design-specific samples to supplement those from APL-DI library characterization, set the GSR keyword APL_DID to 1. During the RedHawk run, APL runs automatically to cover instances that are not covered well in the APL-DI library, providing a transparent, quick and automatic incremental characterization with APL-DI accuracy.

To make Spice models and libraries available, a “packaging” step is performed during the APL-DI phase. A sub-directory ‘PACKAGE’ is created under the directory

`APLDI_[yyyymmdd]/corner[n]/CURRENT/`

which contains all Spice files, the `apache.apldi` file (APL-DI sample file), and a design-specific configuration file derived from the APL-DI configuration file. All the necessary Spice files are collected here, and the ‘include’ file paths in these files are modified to be local.

For previously-generated DI libraries, a utility ‘`aplgenpkg`’ can be used to create the PACKAGE sub-directory without having to rerun APL-DI. To do this, go to the APL-DI run directory and execute the command:

`aplenpkg -d APLDI_[yyyymmdd] -c <apldi_config_file>`

where `APLDI_[yyyymmdd]` is the directory containing the current files you are using.

You must import APL-DI using the directory format (instead of a single `*.current` file) during RedHawk ‘import apl’ in order to take advantage of this enhanced characterization capability. Use the TCL command:

`import apl <file>`

where `<file>` is of the form

`APLDI/APLDI_20070310/corner1/CURRENT`

Running APL Characterization from a UNIX Shell

All APL characterization procedures can be executed from a UNIX shell using the following syntax. By default design-independent current characterization is performed on the full cell library.

```
% apldi [-c] [-w] [ -l <cell_file> | -p <decap_file> ]
    ? -o <output_file>? ? -s [1|2]? ?-j [1|2]? ?-v?
    ?-fc ? <apl_config_file>
```

where

-c: runs decap characterization on library cells

-w: runs pwcaps characterization for low power designs

-l <cell_file>: runs APL only on the cells specified in the cell file.

-o <output_file>: writes the characterization result to specified file. The default is `*.spcurrent` for DD switching current characterization, `*.cdev` for decap characterization, and `*.pwcdev` for low power characterization.

Note: Do not specify the reserved names `*.spiprof` (intermediate working files), `*.spcurrent`, `*.cdev` or `*.pwcddev` as extensions for output file names.

- p <decap_file>: specifies intentional decap cells for characterization. An example of a decap file is shown with the decap example invocations in the following sections. The input pins of the decap cells are defined in the sub-circuit library, with arbitrary ordering. (Should not be combined with -c option.)
- s [1 | 2] (for APL_RUN_MODE DI only): 1 - generates samples only, does not run characterization; 2 - runs APL without generating samples, such as for a design-dependent characterization, when samples for the design have already been generated during the ‘setup_apl’ phase. This option is ignored if APL_RUN_MODE is set to DD.
- j [1 | 2]; indicates a multiple machine run with 1: Platform LSF bsub mode, or 2: Platform LSF API mode (allows use of more options)

Note: The `-j` option also must be used for a SunGrid submittal; the number is ignored if specified.

- v: displays messages in detail (“verbose” mode)
- fc: runs fast library checking mode for potential data problems in the cell library.
- <apl_config_file>: specifies the configuration filename. Note that this file is required and should be the last argument in the command.

Note that the characterization mode by default is *design independent*. The mode can be set to design-specific using the APL_RUN_MODE keyword in the APL configuration file. lmwait is supported in apldi utility to wait for license availability for a defined amount of time rather than immediate exit

Usage:

```
setenv APACHEDA_LICENSE_WAIT <xxx:unit sec>  
Apldi utility waits for time as set by above variable before erroring out
```

Multiple Vdd /Vss Decap Cells

Since decap cells are not present in the LIB files, their characterization is different from standard cells. For multiple Vdd/Vss decap cells, Vdd and Vss values and P/G pin information must be specified, as shown below:

1. Vdd values are specified using the 'DECAP_VDD_PIN' keyword in the APL config file (*aplди.config*), as in the example below:

```
DECAP_VDD_PIN {  
    VDD 1.2  
    VDDPST 3.3  
}
```

2. P/G pin information is read from the custom lib file, specified by the 'CUSTOM_LIBS_FILE' keyword in the APL config file:

```
CUSTOM_LIBS_FILE {  
    <custom.lib>  
}
```

The format of the specified <custom.lib> file is as follows:

```
cell TIEOFF_LL {  
    pin VDD {  
        type vdd  
    }  
    pin VDDPST {  
        type vdd  
    }  
}
```

```
pin VSS {
    type gnd
}
pin VSSPST {
    type gnd
}
```

In addition to the specifications above, since LIB does not have any information on intentional decap cells, power/ground arcs for designs with multiple Vdd and multiple Vss decaps must be defined manually by the user in the *pgarc.lib* file (default name), along with other pgarc definitions. Examples of these declarations in the *pgarc* file are given below:

```
cell <decap_cell_name> {
    pgarc {
        <vdd> <vss>
        ...
    }
}
```

For example:

```
cell decap234 {
    pgarc {
        VDD      VSS
        VDDPST  VSSPST
    }
}
```

Sample APL Invocations

The following are examples of UNIX shell invocations of APL for design-specific characterization:

```
% apldi -s 2 -l cell_list2 apl.config
```

Runs APL switching current characterization on the cells found in file *cell_list2* on a local machine. Results are in the file *./cell.spcurrent*.

```
% apldi -j 2 -s 2 -l cell_list2 apl.config
```

Runs APL switching current characterization on the cells found in file *cell_list2* on a parallel batch farm.

```
% apldi -l cell_list1 -c apldi.config
```

Runs APL decap characterization on the cells designated on a local machine. The results are in default file *./<cell>.cdev*.

```
% apldi -p decap_cell_list6 apl.config
```

Runs APL decap characterization on the intentional decap cells found in the *decap_cell_list6*. The results are in file *./<cell>.cdev*.

The syntax for a decap list file is as follows:

Syntax:

```
<cell_name1>
...
<cell_nameN>
```

Example:

```
DECAP123
DECAP678
```

The following are examples of UNIX shell invocations of APL for library cell (design-independent) characterization:

```
% apldi -j 2 apldi.config
```

Runs APL switching current characterization on the cells designated on a parallel batch farm. The results are in the default file *corner*.current*, where * is a corner number.

```
% apldi -j 2 -c -l cell_list_dc apldi.config
```

Runs decap characterization in multiple machine batch mode on cells found in file *cell_list_dc*. The results are in file *corner*.cdev*.

```
% apldi -j 2 -w cell_list_pw apldi.config
```

Runs APL in multiple machine batch mode on low power cells found in file *cell_list_pw*. The results are in file *corner*.pwcdev*.

Low Power Design Characterization

Characterization for Low Power Designs

Accurate power-up simulation using header or footer switches requires characterization of the cells in the design to get the piecewise linear models for intrinsic capacitances, leakage currents, and effective series resistance, as a function of voltage.

For multiple Vdd and multiple Vss low power designs, pwlcap characterization is arc-based. The arc is characterized based on the keywords PRIMARY_VDD_PIN and PRIMARY_GND_PIN in APL configuration file, with a format the same as that for single-rail cells. The syntax for these keywords is provided below:

```
PRIMARY_VDD_PIN <vdd_pin_name>
PRIMARY_GND_PIN <gnd_pin_name>
```

which specify the primary Vdd and Vss pin names. The Vdd names should match the power pin names found in the SPICE subcircuit netlist. These names may be different than the design's power net names specified in the GSR file.

The APL utility *aplди -w -s 2* is used to characterize a list of standard cells or decap cells present in a low power design, or in a complete library (*aplди -w*), to generate the required PWL model. The UNIX shell command syntax is:

```
% apldi -w [ -l <cell_list_file> | -p <decap_list_file> ]
[ -o <output_file> ] <apl_config_file>
```

Basically the same configuration file is used for low power designs as for other APL characterizations. When characterizing piecewise linear capacitance, leakage, and resistance data for standard cells, the power or ground pin is specified as a sweep source. For header- based designs, the effective voltage on the Vdd pin is swept from near-zero to Vdd, while in footer- based designs the effective voltage on the Vss pin is swept from near-Vdd to zero. The associated power/ground pin name for the header or footer switch can be specified with the APL configuration file keyword (default pin name Vdd):

```
SWEEPSOURCE [<power_pin_for_header> | <gnd_pin_for_footer> ]
```

The default set of eleven characterization voltages (0.1 fractions of the effective voltage, Vdd-Vss, from 0.1*Veff to 1.1*Veff) can be replaced using the configuration file keyword:

```
SWEEPVALUE <eff_vdd_value1> <eff_vdd_value2> ...
```

The output file from the above characterization can be included for a subsequent **RedHawk** analysis by using the GSR command:

```
PIECEWISE_CAP_FILE <PWL_cap_file>
```

Switch Characterization with aplsw

Header and footer switches in low power designs are characterized using the **apls** utility and a switch model file. Non-ideal piecewise linear control pin inputs can be accommodated in the current modeling. The UNIX shell command syntax is:

```
aplsw [-c] [-d] [-o <output_file>] <sw_config_list.conf>
```

where

-c: check generated result

-d: debug mode

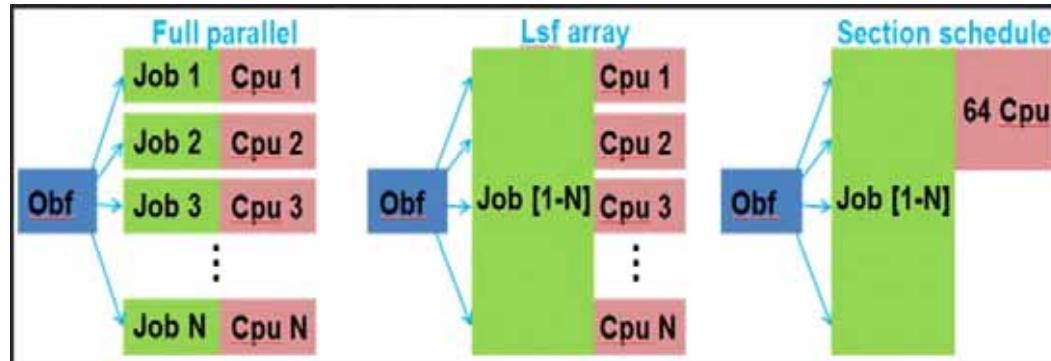
-o <output_file>: output file name

<sw_config_list.conf>: switch configuration list file

For more information about the modeling and analysis of switches, see [section "Low Power Analysis Switch Modeling", page 13-345](#).

Multi-Job Management in APL

APL submits jobs on "9.x LSF" version where LSF env uses feature called Job array to perform Job Management. All jobs in the array share the same job ID and parameters.



Usage:

- Specify LSF submit mode in APL config

```
GRID_TYPE lsf
BATCH_QUEUEING_COMMAND bsub
LSF_SUBMIT_MODE 3 //-- array submission
BATCH_QUEUEING_OPTIONS -J shortjob
```

'-J' option helps to "Assign the specified name to the job". APL will assign each job with unique ID to identify the job. If user does not specify, APL will use aplchar as job array name

- Make sure to specify "-j 1". Array submission is supported ONLY in bsub mode.

Example:

```
$APACHEROOT/bin/apldi2 -l list -j 1 apl.current.config
```

- From this version, APL uses its own mechanism to monitor job, won't query LSF daemon anymore.

The default LSF Array size of LSF 9.x is 1000. So, in sample split mode, please make sure the LSF array size is bigger than APL jobs number, else, submission will fail.

Output Files

Overall Process Files

Process Log Files

The primary process log files are written in the *adsRpt/Log/* directory and contain the detailed status of the characterization process, including messages from subprograms, and information on which computations for each cell have been completed successfully, which ones have failed, which ones need to be characterized, the characterization conditions, Spice error messages and a numerical summary of these statistics. The process log files have the following filename formats:

- current: *apl.current.log.<time_stamp>*
- decap: *apl.cap.log.<time_stamp>*
- pwcap (low power): *apl.pwcap.log.<time_stamp>*

where *<time_stamp>* takes the form ‘2006-05-02-11:52’. Under */adsRpt/* each characterization run creates a top level *log* file that soft-links to the latest results file, although previous versions are still available in the directory.

The previous *apldi.*.log* files have been replaced by the *apl.*.<time_stamp>* files in the *./adsRpt/Log/* directory.

An example current log file is shown below:

```
Nominal VDD used: 1.8
Temperature used: 25
Netlist used: /.../spice/cellabc.sp
Voltage range: 2.07 1.8 1.62 1.35
#
#----- slevs -----|
stat #smpl #smpl_done c_min c_max rs_min rs_max fs_min fs_max vec cell_name
pass 70 70 1.00 278.8 0.012 2.500 0.012 2.500 LIB bufx1
```

Error and Warning Files

Error flags during characterization are written into *adsRpt/Error* directory files with the following filenames:

- current: *apl.current.err.<time_stamp>*
- decap: *apl.cap.err.<time_stamp>*
- pwcap (low power): *apl.pwcap.err.<time_stamp>*

Warning flags during characterization are written into *adsRpt/Warn/* directory files with the following filenames:

- current: *apl.current.warn.<time_stamp>*
- decap: *apl.cap.warn.<time_stamp>*
- pwcap (low power): *apl.pwcap.warn.<time_stamp>*

Under */adsRpt/* each characterization run creates a top level */warn/err* file that soft-links to the latest results file, although previous versions are still available in the directory.

Status Log File

The output of the `aplstat <apl_config_file>` command displays on screen, and writes to a log file `aplstat.log` in the run directory, a summary of the cell characterization process as it progresses (the cells that have been characterized successfully, the cells that have failed and the cells still pending), including the reason for failure for each cell characterization that was unsuccessful. A sample portion of an APLSTAT log is shown below:

```
APLSTAT: Apache Power Library Cell-characterization
          Status Monitor

          ##### Cell Status #####
          ##### 1606 Cells Succeeded #####
          AND2_wxyz pass
          AND2_nmop fail
          AND2_rstu pass
          AND2_qvwx pass
          ...
          . . .
```

Results Files

Each type of characterization generates a set of output files for holding results. Both design-independent library characterization and design-specific characterization can generate three types of files as desired, one set for switching waveforms, one for decap and one for piecewise linear decap for ramp-up low power applications. Each type of output has a default filename, but the filename can be specified using the option

```
apldi -o <filename>
```

APL-DI Output Files

- Switching waveform files - contain a set of current profiles for all switching conditions for each cell in the library. Default filename: `/<corner_name>.current`
- Decap files - contain the intrinsic decap values, as well as ESR and leakage current for each cell in the library, including specified intentional decap cells. Default filename: `/<corner_name>.cdev`
- Power-up decap files - contain the piecewise linear voltage relationships for intrinsic decap values, as well as for ESR and leakage current for each cell in the library, including specified intentional decap cells. Default filename: `/<corner_name>.pwcdev`

APL-DD Output Files

- Switching waveform files - contain a set of current profiles for all switching conditions for each cell in the design. Default filename: `/<cell>.spcurrent`. The maximum number of nodes in the `*.spcurrent` file that can be used in analysis is 1 billion.
- Decap files - contain the intrinsic decap values, as well as ESR and leakage current for each cell in the design, including specified intentional decap cells. Default filename: `/<cell>.cdev`
- Power-up decap files - contain the piecewise linear voltage relationships for intrinsic decap, as well as for ESR and leakage current for each cell in the design, including specified intentional decap cells. Default filename: `/cell.pwcdev`.

Individual Cell Characterization Files

Characterization Results

APL generates a set of result files in a default results directory. For each characterization process APL generates cell results files with the following default filenames and directories:

APL-DI Cell Results Files

- cells: <results_dir>/corner*/CURRENT/<cellname>.spiprof
- decaps: <results_dir>/corner*/CAP/<cellname>.cdev
- low power: <results_dir>/corner*/PWCAP/<cellname>.pwcdev

The default <results_directory> for these output files is *APLDI_<time_stamp>*/ under the run directory.

APL-DD Cell Results Files

- cells: <results_dir>/CURRENT/<cellname>.spiprof
- decaps: <results_dir>/CAP/<cellname>.cdev
- low power: <results_dir>/PWCAP/<cellname>.pwcdev

The default <results_directory> for these output files is *APLDD_<time_stamp>*/ under the run directory.

Using the APL configuration file keyword ‘`APL_RESULT_DIRECTORY <results_dir>`’ allows you to specify a different results directory for the output cell characterization files.

Cell Log Files

In the APL results directory there is also a cell log file for each cell that records all Info, Warning, and Error messages recorded during characterization of that cell. These files have the following default directories and filename formats:

APL-DI Cell Log Files

- current: <results_dir>/corner*/CURRENT/<cellname>.current.<time_stamp>.log
- cap: <results_dir>/corner*/CAP/<cellname>.cap.<time_stamp>.log
- pwcaps: <results_dir>/corner*/PWCAP/<cellname>.pwcaps.<time_stamp>.log

The default <results_directory> for these output files is *APLDI_<time_stamp>*/ under the run directory.

APL-DD Cell Log Files

- current: <results_dir>/CURRENT/<cellname>.current.<time_stamp>.log
- cap: <results_dir>/CAP/<cellname>.cap.<time_stamp>.log
- pwcaps: <results_dir>/PWCAP/<cellname>.pwcaps.<time_stamp>.log

The default <results_directory> for these output files is *APLDD_<time_stamp>*/ under the run directory.

Using the APL configuration file keyword ‘`APL_RESULT_DIRECTORY <results_dir>`’ allows you to specify a different results directory for the output cell log files.

APL Results Checking and Processing

To check the general validity of APL results data, look at final statistics summaries from the cell results files, or to convert formats or sort cell data, use the *aplchk* utility from a Unix command line. The command evaluates data in the specified APL result file or directory for data range and missing information, or filters and compiles data for particular cells. It also checks the consistency of cell types (combinational /sequential) and their

transitions (0->0, 1->1, 0->1, 1->0) in the current profiles to ensure that current profiles for sequential cells have all four transitions and combinational cells have only 0->1 and 1->0 transitions. The *aplchk* utility reports corrupted cells in the log file and continues to scan cells. When all checks are completed, the list of failed cells is reported in the *adsRpt/aplchk.log* file. The syntax of the command is:

```
aplchk <input_file/dir> ?-v ? ?-l <list_file>? ?-w <output_file>?  
?-c? ? -pwc <pwcdev_file>? ? -conf <API_config_filename> ?  
?-spice <apl config file>?
```

where

<input_file/dir>: name of input file or directory containing switching current, cdev, or pwcdev decap data to process. APLCHK can identify the checking object; whether it is a single APL file or a directory, APLCHK checks all APL data in that file or directory.

-v: prints warnings in std error format (verbose mode)

-l <list_file>: writes out result information on cells named in <list_file> to the specified <output_file>

-w <output_file>: specifies an output filename

-c: specifies checking *cdev* decap files

-pwc: specifies the pwcdev characterization results file to be used by *aplchk*. When both old format and new format pwcdev files are to be merged, the merged result is in the new format.

-lib <config_filename>: specifies APL config file containing APL results and LIB files.

-spice <apl config file>: dumps spice waveforms to validate *cdev* modelling of an instance.

The following is an example of an *aplchk* command:

```
aplchk -l listAB -w cellMN.current
```

which writes out data on the cells in listAB into the *cellMN.current* result file. The *aplchk.log* file reports information such as the tool version, run path, starting time of run, command used to run, and runtime/memory details, as well as *aplchk* results.

Keywords for Checking Limits

The *aplchk* utility supports global parameter check limits, cell type-based check limits and cell name- based check limits. The following APL configuration file keywords can be used to define global standard cell checking limits for *aplchk*:

IMAX_STDCELL_WARN <i_peak>

Specifies the max peak current warning limit for standard cells (uAmps). Default: 100,000uA

ITAIL_STDCELL_WARN <i_max_tail>

Specifies the max tail current warning limit for standard cells (uAmps). Default: 10uA or 0.3*Peak current value, whichever is larger.

SLEWMAX_STDCELL_WARN <max_trans_time>

Specifies the max output transition time (“slew”) warning limit for standard cells (ps). Default: 3,000,000 ps

DELAYMAX_STDCELL_WARN <max_delay>

Specifies the max input-output delay warning limit for standard cells (ps). Default: 30,000 ps

FIREMAX_STDCELL_WARN <max_fire_time>

Specifies the max input-output firing time warning limit for standard cells (ps). Default: 30,000 ps

CMAX_STDCELL_WARN <max_cap>

Specifies the threshold for allowable standard cell capacitance. The default allowable capacitance value is 1000pf.

RMAX_STDCELL_WARN <max_ESR>

Specifies the threshold for allowable standard cell resistance. The default allowable resistance value is 1000K Ohms.

LEAKMAX_STDCELL_WARN <Warning_leakage_threshold>**LEAKMAX_STDCELL_ERROR <Error_leakage_threshold>**

Specifies the Warning and Error thresholds for allowable standard cell leakage current. The default Warning leakage threshold is 10uA and the default Error leakage threshold is 1A

LEAKMAX_MEMORY_WARN <Warning_leakage_threshold>**LEAKMAX_MEMORY_ERROR <Error_leakage_threshold>**

Specifies the Warning and Error thresholds for allowable memory leakage current. The default Warning leakage threshold is 0.5A and the default Error leakage threshold is 1A.

PWC_VDD_MAX_WARN <Max_Vdd>

Specifies the Warning threshold for the maximum allowable Vdd value. The default allowable Vdd value is 5.0V.

The following configuration file keyword and options can be used to set cell-specific checking limits, defined the same as for the global limits above.

CELL_CHECK_LIMITS <cellname> {

IMAX_WARN <uAmps> Defaults: standard cell - 1.0e+5; memory cell - 1.0e+6

ITAIL_WARN <uAmps> Defaults: standard cell - 10; memory cell - 0.5e+6

SLEWMAX_WARN <ps> Defaults: standard cell - 30,000; memory cell - 30,000

DELAYMAX_WARN <ps> Defaults: standard cell - 30,000; memory cell - 30,000

FIREMAX_WARN <ps> Defaults: standard cell - 30,000; memory cell - 30,000

}

The following APL config file keywords can be used to define global memory and I/O cell checking limits for aplchk:

IMAX_MEMORY_WARN <I_peak>

Specifies the max peak current warning limit for memory cells (uAmps). Default: 1.0e+6uA

ITAIL_MEMORY_WARN <I_max_tail>

Specifies the max tail current warning limit for memory cells (uAmps). Default: 0.5A or 0.3*Peak current, whichever is larger.

SLEWMAX_MEMORY_WARN <max_trans_time>

Specifies the max output transition time (“slew”) warning limit for memory cells (ps). Default: 3,000,000 ps

DELAYMAX_MEMORY_WARN <max_delay>

Specifies the max input-output delay warning limit for memory cells (ps). Default: 3,000,000 ps

FIREMAX_MEMORY_WARN <max_fire_time>

Specifies the max input-output firing time warning limit for memory cells (ps). Default: 3,000,000 ps

CMAX_MEMORY_WARN <max_cap>

Specifies the max capacitance warning limit for memory cells (pf). Default: 10pf

RMAX_MEMORY_WARN <max_ESR>

Specifies the max effective power circuit resistance warning limit for memory cells (Ohms). Default: 1.0e5 Ohms

LEAKMAX_MEMORY_WARN <max_leakage>

Specifies the max leakage current warning limit for memory cells (Amps). Default: 1.0A

Resistance, Capacitance, and Leakage Histogram

A histogram of the distribution of capacitance (F), resistance (Ohms), and leakage current (A) values is created for each *aplchk* run, for both logic low and logic high states. A histogram file for an *aplchk* devcap invocation, such as:

```
aplchk -c <cell>.cdev
```

is shown in the following example:

Capacitance range:	No. of cells c0	No. of cells c1
<1.000000e-15	0	0
1.000000e-14	0	0
1.000000e-13	178	172
1.000000e-12	27	33
>=1.000000e-12	0	0

Resistance range:	No. of cells r0	No. of cells r1
<1.000000e+01	0	0
1.000000e+02	0	0
1.000000e+03	150	140
1.000000e+04	55	65
>=1.000000e+04	0	0

Leakage range:	No. of cells leak0	No. of cells leak1
<1.000000e-12	0	0
1.000000e-10	29	12
1.000000e-08	176	192
1.000000e-06	0	1
1.000000e-04	0	0
>=1.000000e-04	0	0

Reports of Cells with no APL Data or samples

Cells without APL current profiles are reported by RedHawk in the file *adsRpt/apache.inst.libCurrent*.

Cells without decap information are reported in the file *adsRpt/apache.refCell.noApICap*.

Cells without voltage-dependent decap information (for low power analysis) are reported in the file `adsRpt/apache.refCell.noAplPwcap`.

A report file is generated during APL import, `apache.refcell.noAplSample`, identifying cells with APLDI data, but no APL samples.

Importing and Merging Characterization Data Files in RedHawk

Importing APL Files

There are several ways of importing APL `*.current` and `*.cdev` files and AVM characterization files into **RedHawk**. The recommended method is by specifying the files by type using the GSR keyword `APL_FILES`, as follows:

```
APL_FILES {
    <APL_binary_filename>    current
    <APL_binary_filename>    cdev
    <Avm.conf_filename>      avm
    <AVM_binary_filename>   current_avm
    <AVM_binary_filename>   cap_avm
    <APL binary filename>   pwcap
    ...
}
```

Another importing method is to use the TCL command

```
import apl <APL_file>
```

Note that '`import apl <current-file>`' and '`import apl -c <cdev-file>`' commands have accumulative effects. That is, subsequent multiple imports are merged.

The `APL_FILES` method is preferred because it can handle current/cap/leak/pwc files and directories, while the GUI menu and TCL commands can only handle one current and cap file at a time.

Note: If there is both an '`import apl <file>`' in the command file and `APL_FILES` in the GSR file, then the command takes over and `APL_FILES` is ignored.

To allow control of improper file inputs, there are several checks that are made on imported characterization files. **RedHawk** stops on some APL import errors, such as an APL utilities returned error code, and provides ways to continue the **RedHawk** flow with flexibility. You have the following options to control import errors:

- By default differences in model, Vdd, and temperature in input files are ignored.
- Ignore all types of errors with the command

```
gsr set ignore_apl_check 1
```
- Fix the data and re-run APL.
- To continue, type the TCL command

```
import apl
to redo importing from APL_FILES.
```
- Or, instead of using `APL_FILES` input, you can use the GUI command **APL -> Import** menu command, and fill out the dialog form.

Merging APL Result Files

The ‘aplmerge’ utility, invoked from a Linux/Solaris command line, quickly checks the compatibility of corner conditions (P, V, T) for similar types of APL files to be merged, such as `<cell>.current` and `<cell>.cdev`, and, if they are compatible, merges them into a single output file. The ‘aplmerge’ utility supports three types of input-source specifications:

- individual APL data
- a directory containing APL data
- a file containing a list of cells to be merged

and each input-source specification is independent of the others. If any of the input data is not in the current working directory, its path must be specified.

Also, ‘aplmerge’ can merge `<cell>.spcurrent` files that have different resolutions, such as 50-point files and 400-point files.

The syntax of the command is:

```
aplmerge [-c] [-pwc] [-rep] [-l <cell_list_filename>]
         [-im] [-t] [-o <output_file>] [-avm] [-ilimit]
         [<file1> [<file2> [...]]] [<directory>] [-fl <list_file>]
aplmerge [-c] [-pwc] [-rep]
[-l <cell_list_file> | <directory> | [<file1> <file2> ...] ]
[-im] [-t] [-o <output_file>] [-avm] [-ilimit]
```

where

- c: specifies merging intrinsic decap files
- pwc: specifies merging piecewise linear decap files for power-up analysis
- rep: specifies replacing all information on cells in `<file1>` with information on cells with the same name in `<file2>`. Cell information not duplicated in the second file remains the same in the output file.
- l `<cell_list_file>`: specifies the file that defines the cells to be included in the merged output file. If the cell files are not in the run directory, the file path should be specified. Used for `.spiprof` and `.cdev` files with one cell per file. In the cell list file, list the cell names whose APL data need to be merged, along with the full path to the data, in the following format, one line for each entry: ‘`<full or relative path>/<cellname>`’. Do not specify the names of the files or the directory.
Note that the option ‘-l’ is independent of any `<directory>` specified at the end of syntax line.
- im: specifies that model entry checking is to be ignored
- t: specifies turbo mode, which runs faster by skipping duplicate cell checks
- o `<output_file>`: defines the output file. Default: `cell.spcurrent.merge`
- avm: merge AVM current regardless of header differences
- ilimit: ignore checking limit
- `<file1> <file2> ...` : specifies files to be merged into the output file
- `<directory>`: specifies a directory containing cells to be merged
- fl : allows users to specify a list file containing files to be merged. The format of the list file is:
 - Cell1.spcurrent
 - Cell2.spcurrent
 - Cell3.spcurrent

Typical examples:

```
aplmerge -o <cellA>.current -l CellListABC
```

```
aplmerge -o <cellB>.current CellDirEFG
```

You can also provide wildcard filenames with aplmerge, as in the following example:

```
aplmerge -o <cell>.current *.spiprof
```

which merges all input files with the extension `*.spiprof` into an output file called `<cell>.current`.

aplccs (CCS2APL) Library Characterization

Introduction and Syntax

The ‘aplccs’ utility converts CCS library data to APL switching current data format.

The advantages of using CCS2APL are:

- transparent to RedHawk tools. You can generate CCS2APL DI library before a RedHawk run.
- CCS2APL data are fully supported by APL utilities
- very fast conversion time.
- timing information in CCS2APL data are hard values

For standard cell conversion

- supports diagonal-type sparse cload table.
- multi-nominal Vdd value merging for a same cell is supported only when the same slew normalization factor/temperature/pg_pins/states/samples are found among the different nominal Vdd libraries.
- CCS lib has no cload concept for C00/C11 toggles. aplccs performs sample expanding for C00/C11 toggles, which means for a C00/C11 toggle, the waveforms for a specific slew value are all the same.

For memory, I/O and custom cells:

- supports characterization of cells that have C-load independent states by using a default Cload value of 1 fF for such cells.
- supports multi output cell.

Syntax for invoking aplccs is as follows:

```
aplccs [-o <output_file>] -mstate_cells <cell_list_file>  
-conf <aplccs_config_file> [-skip_ps]
```

where

- o <output_file> : specifies the output filename (default `-ccs.current`)
- mstate_cells : specifies a multi-state cell list file supporting multiple-state APL current characterization data for standard cells
- conf: specifies a aplccs configuration file
- skip_ps : skips PowerStream run when PSLIB file `.apache/apache.pslib` is generated.

Example when generating PSLIB `apache/apache.pslib`:

```
aplccs -skip_ps -o ccs.current
```

Example for no PSLIB:

```
aplccs apl.config -o ccs.current
```

APLCCS Configuration File

The aplccs run-related configuration file keywords in apldi config file are shown following:

1. LIB_FILES/ LIBS_DIRECTORY/ LIBS_FILE
See [section "APL Configuration File Description", page 9-218.](#)
2. RETRY_VECTORLESS [0 | 1]
To be used when aplccs fails to convert the CCS lib to APL using the default vectors, which is the same one used in APL characterization. Specify this keyword to force the conversion using random vectors. Note that this may cause the inexact comparison to APL.
3. IGNORE_CELLS {
 <cellname1>
 ...
}

Used to void the CCS2APL current generation failure due to some cells with unexpected CCS lib formats (corner case). Using this keyword ignores the failed cells and generates the CCS2APL current file for the rest of cells in the library.

Importing the CCS Lib Directly

Instead of using the aplccs utility to do the conversion before a RedHawk run, the CCS lib can be directly imported into RedHawk; CCS2APL conversion is transparent in the RedHawk flow. The GSR keyword to turn on CCS lib import is "LIB_USE_CCS 1".

The keyword "CCS_OVERRIDE_APL 1" is only for cases in which CCS has a higher priority than APL when both exist. "CCS_OVERRIDE_APL 1" works only when "LIB_USE_CCS 1" is set.

There is also an aplccs keyword to convert the CCS lib (which contains intrinsic_parasitic ESC/ESR and leakage_curernt) to APL cdev, as follows:

CONVERT_CCS_CAP [0 | 1]

The default is 0; set to 1, CCS=>CAP conversion is performed.

Note that the user has the responsibility to ensure that the meaning of CCS intrinsic_parasitic ESC and ESR is the same as APLCAP. This feature is off by default.

I/O Cell Characterization

Input/output cells can inject a significant amount of noise into the core, especially if the Vss is shared between I/O and core, and therefore I/Os should be included in dynamic analysis. I/O cells are special, since they have multiple Vdd pins for the core and I/O supply. Therefore, both core and I/O Vdd/Vss pins' current must be characterized.

An I/O cell typically has multiple input or control pins, which must be accurately characterized. The correct values for input and control pins are obtained by preparing an input vector file for the I/O cell, as described in the section below, and in the [section "Custom Cell Characterization Data Preparation", page 9-241.](#)

I/O Cell Characterization Procedure

1. Define the I/O Vdd pin and its value. Note the I/O Vdd pin is defined in the SPICE '.subckt' line, not in the .lib file.

```
dc <io_vdd_pin_name> <value>
2. Define open pins for monitoring or scanning, where no connections are necessary.
```

```
openpin <open_pins>
```

```
3. Define voltage source parameters and define the I/O Vdd pin value.
```

```
param vddo=1.5
```

```
param vref=1.2
```

```
dc VDDQ15 vddo
```

For input signals, you may let APL automatically generate the input stimuli, or use the VECTOR statement.

```
4. For automatic stimuli generation, define the ‘active_input’ pins. Do not use automatic stimuli generation for differential I/O cells.
```

```
active_input clk
```

For user-generated stimulus, define the VECTOR statement as shown below. All timing parameters use TUNIT as time unit. The values can be either “0” (low) or “1” (high). APL assumes that the first signal is the active input pin.

```
VECTOR {
    VNAME <signal_names>
    TUNIT <ps|ns|us|ms>
    VIH <input_high_value>
    SLOPE <slope_value>
    <time0> <value_pin1><value_pin2><...>
    <time1> <value_pin1><value_pin2><...>
    ...
}
```

```
5. Loading. An I/O cell usually drives large loads all the way to the board. Therefore, it is often necessary to add an off-chip load to the I/O cell’s output to obtain realistic current profiles. This is realized by including the load circuitry and its respective subcircuit definitions in the input vector file (<design>.inpvec), as shown below.
```

```
OUTPUT <output_pin_name> <output_load_subckt_name>
```

Following is an example use of the OUTPUT statement.

```
OUTPUT Y0 pkg_board_load
```

where the subcircuit, *pkg_board_load*, is defined in the included file called *pkg_board_load.subckt*.

```
include pkg_board_load.subckt
```

whereas the included file, *pkg_board_load.subckt*, contains the following subcircuit:

```
.subckt pkg_board_load chip_node
X1 chip_node bga_node pkg_load
X2 bga_node far_end_node board_load
.ends pkg_brd_load
```

```
.param P_DELAY=2
.subckt board_load nearend farend
```

```
*pc board
CLOAD1 nearend 0 0.5pF
TPCB nearend 0 L11 0 Z0=50 TD='180P*P_DELAY'
CLOAD2 L11 0 0.5pF
Rload L11 farend 5
```

```
CLOAD3 farend 0 3.5pF
.ends board_load

* signal I/O package
.subckt pkg_load n1 n2
c1 n1 0 10f
L1 n1 n2 10pH r=5m
c2 n2 0 10f
.ends pkg_load
```

Additional Keywords for I/O Cells (Optional)

The following keywords are available to further customize I/O cell characterization.

TSTEP, TSTOP

TSTEP and TSTOP define the step size and stop time parameters for transient simulation.

Syntax:

```
TSTEP <step size><unit>
TSTOP <stop time><unit>
```

Example:

```
TSTEP 0.1ns
TSTOP 50.0ns
```

OP

OP defines the dump time for decap characterization.

Syntax:

```
OP <time1><unit> <time2><unit>
```

where

<time1>: specifies the time output goes high

<time2>: specifies the time output goes low

Example:

```
OP 10ns 15ns
```

tranXX

The tranXX parameter defines switching window starting times. Note that the window should be large enough to allow adequate sampling points.

Syntax:

```
tranXX <time><unit>
```

where

tranXX <time>: XX defines the output value change and <time><unit> specifies the time of occurrence after t=0 and the time unit.

Example:

```
tran01 10.0ns
tran10 30.1ns
tran00 5.0ns
tran11 15.0ns
```

Note: tranxx also can be put at the end of the <time_step_num> line in the input 'Vector' section, in which case it has a higher priority than when specified by itself. Refer to the [section "Input Vector Files", page 9-242](#).

ioprobe

The ioprobe parameter defines the Vdd and/or Vss pins to probe and their current polarity. Note that in Spice positive current direction is INTO the positive pin of a voltage source.

Syntax:

```
ioprobe <VDDname> <+ | -> <VSSname> <- | +>
```

Example:

```
ioprobe VDD - VSS +
```

NOTE: Only one representative sample is considered for an I/O cell.

Characterization of I/O Cells

To obtain the most accurate I/O cell characterization use the *sim2iprof* utility for current profiles (see [section "sim2iprof", page E-922](#)) and the ACE utility for device capacitance characterization (see [section "ACE Decap and ESR Characterization", page 9-263](#)). For a much faster, but less accurate, process, use the AVM datasheet characterization utility (see [section "AVM Datasheet Characterization", page 9-271](#)).

Memory and IP Characterization

To obtain characterization data for memory, custom IP and I/Os, the same as APL creates for standard cells, there are two choices, depending on the need for either very accurate current profiles or a fast run time and good current profile accuracy:

- Accurate *sim2iprof* and ACE characterization

The *sim2iprof* utility provides very accurate switching current profiles for memories and I/Os, while ACE provides equivalent power circuit resistance, device capacitance, and leakage current data.

- Fast AVM datasheet characterization

AVM is a rule-based utility for quick generation of power circuit characterization data for memory and other types of IP. AVM converts a datasheet-based input specification into triangle-based or trapezoidal current profiles, equivalent power circuit resistance, device capacitance, and leakage current.

Sim2iprof Switching Current Characterization

For details on preparing the configuration file and running *sim2iprof* to generate memory switching current profiles, see [section "sim2iprof", page E-922](#).

ACE Decap and ESR Characterization

ACE estimates the P/G pin intrinsic and intentional decap values for all types of memory and other complex IP blocks. It uses a unique tracing algorithm to find the P/G network in a cell, and calls SPICE to characterize representative MOSFET's gate capacitance, and generates various types of capacitance and effective resistance values for the cell in APL format, as well as an estimate of leakage current. ACE also traces R elements in power/

ground nets using DSPF information (*|NET, *|I ...) to identify relevant capacitance. You can then inspect, merge, and import ACE results into the RedHawk database to perform DvD analysis.

ACE can recognize header/footer switches embedded inside a memory or a functional block. This is especially useful for low-power design, since many cells come with the switches built-in. Since ACE uses a net tracing algorithm, it runs extremely fast. For a half-million MOSFET cell, the typical run time is approximately a minute on a Linux box.

Identifying tied-off devices as decaps

ACE automatically identifies tied-off devices as decaps, including the following configurations of device connections:

For tied-off conditions, treat as decaps.

a. nmos:

- source/gate ties to ground, drain to power.
- drain/gate ties to ground, source to power.

b. pmos:

- source/gate ties to power, drain to ground.
- drain/gate ties to power, source to ground.

For the following combinations, assign zero current, but do not treat as decap:

a. nmos:

- source/gate ties to ground, drain to signal.
- drain/gate ties to ground, source to signal.

b. pmos:

- source/gate ties to power, drain to signal.
- drain/gate ties to power, source to signal.

Identifying special pins for PWCap characterization

ACE automatically identifies special pins, such as retention pins in power management cells. These pins are identified from the Liberty attributes. The correct retention condition for such cells is picked from the lib and used during pwcap characterization. This process has no effect on current or cap characterization.

ACE Configuration File

ACE requires an input configuration file, using the keywords described on the following pages (names are not case sensitive). The required keywords are:

- External_power_net
- External_ground_net
- Subckt
- Modelfile

Note that new nanometer device model libraries are often too complicated to be specified using 'Modelfile'. The libraries also can be encapsulated using the 'Include' keyword instead of 'Modelfile'.

- VddValue

Other configuration file keywords are optional.

Ace_HSpice <path/binary>

Specifies that HSpice is used as the characterization simulator. You only need to specify this keyword when HSpice is needed. The default simulator is NSpice.

Ace_Eldo <path/binary>

Specifies that ELDO is used as the characterization simulator. You only need to specify this keyword when ELDO is needed. The default simulator is NSpice.

Ace_Spectre <path/binary>

Specifies that Spectre is used as the characterization simulator. You only need to specify this keyword when Spectre is needed. The default simulator is NSpice.

ACE_OPTION {<options>} }

Provides options to control ACE simulation, such as

`SIMULATOR_COMMAND_OPTION -spectre`

which invokes the '-spectre' option when using the Finesim simulator.

Decap_Subckt

Defines sub-circuit-based intentional decap instances in the Spice netlist. The syntax and usage is described following.

```
DECAP_SUBCKT {
    <decap_subckt_name1> <port_type_list1> param <param_list1>
    <decap_subckt_name2> <port_type_list2> param <param_list2>
    ...
    <decap_subckt_nameN> <port_type_listN> param <param_list3>
}
```

where

<decap_subckt_nameN>: describes the subcircuit name of the intentional decap instance

<port_type_listN>: defines the type of subckt port connection, either power, ground, or signal net, designated as 'power', 'ground', or 'na' in the port type list ('na' designates a signal net connection)

Note that there must be one item in the <port_type_list> for each pin name.

param <param_list>: defines the parameters that determine the size of the intentional decap instance. For example, in a Spice netlist as follows:

```
XDECAP1    PLUS    MINUS    NMOSCAP    lr=6.3u    wr=3.92u    m=4 ...
XDECAP2    PLUS    MINUS    NMOSCAP    lr=3.3u    wr=1.92u    m=8 ...
...
```

External_Power_Net <ext_power_pin> ...

Defines the external power pin(s) of the cell. Different power pins represent different power domains. The power pin names must be Spice pin names declared in the Spice subckt file. You must provide power pin names using this keyword for ACE to trace the power nets inside the Spice subckts. See also Internal_Power_Net and VP_Pairing keyword usage.

External_Ground_Net <ext_ground_pin> ...

Defines external ground pin(s) of a cell. Different ground pins represent different ground domains. The ground pin names must be Spice pin names declared in the Spice subckt file. You must provide ground pin names using this keyword for ACE to trace the ground nets inside the Spice subckts. See also Internal_Ground_Net and VP_Pairing keyword usage.

FINFET_UNIT_WIDTH <w_meters>

ACE identifies a unique length for transistors for cdev characterization. FinFET transistors are defined by the number of fins (NFIN) and their length (L), instead of device width and length. In 16nm technology ACE only traces the unique size transistors with different lengths, and you must specify the fin width using this keyword. In 16 nm FinFET technology the transistor width parameter no longer exists.

FLIP_WELL

Using this keyword ACE can handle cells made from FDSOI technology, or the flip-well process. FLIP_WELL must be set to 1 to enable this feature.

Global <node_name> [<node_name> ...]

Defines global nodes (nodes that are connected in all hierarchical levels).

Ignore_Decap_Subckt_Checking [0 | 1]

Allows you to control of whether ACE errors out or just prints a warning message when a decap subckt is not instantiated. Set the keyword to 1 to prevent ACE from erroring out. Default is 0.

Ignore_Subckt

Specifies capacitance contributions to be excluded from the specified subckt(s) and their descendants. This is useful when you want to extract, for example, only the capacitance contributed by the top level elements, and ignore subckt contributions. The syntax is:

```
Ignore_subckt {  
    <subckt_name>  
    ...  
}
```

Include <file>

Used to include needed Spice files, such as a subckt definition, or Spice model file.

Internal_Power_Net <internal_power_pin> ...

Internal_Ground_Net <internal_gnd_pin> ...

Specifies separate internal P/G nets for header or footer switches in the ACE configuration file. So for switch cases, internal nets have different entries in the <>.mcap file. ACE handles hierarchical internal nets by using hierarchical power/ground pin names, with “.” as a hierarchy divider. In this way you can include all RC elements related to this net in ACE characterization, but this can be used only with a hierarchical subcircuit netlist. Also, Cdev characterization for internal nets can be performed using these keywords. The switch cap between external and internal nets is ignored. The output Cdev contains entries for both external and internal nets. Use also the keyword VP_PAIRING, as follows.

```
EXTERNAL_POWER_NET <external_power_pin/net> ...
```

```

EXTERNAL_GROUND_NET <external_gnd_pin/net> ...
INTERNAL_POWER_NET <internal_power_pin/net> ...
INTERNAL_GROUND_NET <internal_gnd_pin/net> ...
VP_PAIRING    <internal_net> <external_net>
? SWITCH_SUBCKT [<switch_subckt_name> |
<switch_filename>]?

```

An example subcircuit appears as follows:

```

.SUBCKT BLOCK in out vdd vss nd5
Xswitch ctrl vdd net_int SWITCH
Xinv net_int vss in out INV
.ends

.SUBCKT testckt in1 in2 out1 out2 vdd vss
Xblock1 in1 out1 vdd vss nd5 BLOCK
Xblock2 in2 out2 vdd vss nd5 BLOCK
.ENDS

```

Sample ACE configuration file settings are shown following:

```

EXTERN_POWER_NETS vdd
INTERNAL_POWER_NET Xblock1.net_int Xblock2.net_int
VP_PAIRING (Xblock1.net_int vdd) (Xblock2.net_int vdd)
EXTERN_GROUND_NETS vss
VDDVALUE vdd 0.85
switch_sub SWITCH

```

The output *mcap* file then contains Cdev entries for both external and internal nets.

An example aplreader output would be:

```

Info: Reading cdev file- 'testckt.mcap'
Info: cell= testckt
Info: pin= vdd cap= 0.389455 pf, res= 2.56772 ohm, leak= 20 uA
Info: pin=Xblock1.net_int cap= 13.866 pf, res= 0.072121 ohm, leak= 40 uA
Info: pin=Xblock2.net_int cap= 30.1777 pf, res= 0.03313 ohm, leak= 60 uA
Info: pin= vss cap= 16.2794 pf, res= 0.0615381 ohm, leak= 120 uA

```

K_GROUND_CAP/K_FLOAT_CAP

Using these keywords ACE can extract signal net capacitance from the DSPEF netlist, and adds its value in the cdev file generated (ACE classifies parasitic capacitance into three types: P/G parasitic cap, grounded cap, and floating cap.)

K_GROUND_CAP defines the fractional contribution of grounded signal cap to P/G parasitic cap, and **K_FLOAT_CAP** defines the fractional contribution of floating signal cap to P/G parasitic cap. Their default value is 0.5, and their allowable range is 0 to 1.0.

```

K_GROUND_CAP <value>
K_FLOAT_CAP <value>

```

Leakage_i

Specifies per-arc leakage current in Amps. Required when the low power '-pwc' option is used in ACE, with the syntax:

```

Leakage_i {
<vdd_pin1> <gnd_pin1> <leakage_A>
...
}

```

Metal_Resistor/ Metal_Resistor_File

These keywords allow ACE to trace through resistors defined as subcircuits and include transistors that are connected beyond the subcircuit resistors, which then enables cap characterization of MOS transistors connected after the resistors. To perform resistor trace-through, the subcircuit names must be defined using the syntax:

```
METAL_RESISTOR <model_name1> <model_name2> ...
```

or

```
METAL_RESISTOR_FILE <path to metal resistor file>
```

The format of the contents of the specified METAL_RESISTOR_FILE is as follows:

```
<Resistor_model_name1>
<Resistor_model_name2>
<Resistor_model_name3>
...
```

ModelFile <file> <corner>

Specifies Spice device model library and process corner case.

MOS_Report [0 | 1]

When set to 1, produces a *<cellname>.mos.report* file, as shown in the following example of a MOS report for a memory:

```
Mos Name W L Count Domain
psvtlp 0.875 0.06 12 vdda
nsvtlp 0.46 0.06 14 vss
psvtlp 2.5 0.06 12 vdda
```

NOMINAL_VDD <vdd_domain1> <v1> ?<vdd_domain2> <v2>?

Specifies the nominal Vdd value for each Vdd domain when they are different from the value specified by the keyword VDD_VALUE, which is user-specified in the APLMMX configuration file, or in the ACE input file. If 'NOMINAL_VDD' is not specified when SIGNAL_PARASITIC_C is turned on, a warning is displayed and the nominal Vdd value is assumed to be the same as specified by VDD_VALUE.

NMOS_Model_Name <model_name>

Defines the NMOS model name if a device model file is not specified.

PMOS_Model_Name <model_name>

Defines the PMOS model name if a device model file is not specified.

Option <Spice_option>

Specifies any standard Spice option to be passed to the Spice deck for ACE decap characterization. Multiple OPTION definitions are allowed on separate lines.

Example: OPTION scale=1e-6

Option parhier=[local | global]

Specifies the priority of hierarchical parameter. The default value is 'global'.

Preprocess_Subckt_File [0 | 1]

When set, improves handling of the configuration view Spectre netlist, which contains the test bench setup, so that the “tran” statements in the test bench and any other Include files that are not relevant for capacitance characterization are automatically skipped, reducing overall ACE runtime (default 0).

Primary_Power_Net <pin_name>
Primary_Ground_Net <pin_name>

Supports PWL cap data for low power analysis using the ACE '-pwc' option. When using the '-pwc' option, you must specify leakage values in the ACE configuration file using keyword 'Leakage_i' to get an accurate value. If you specify the primary pin names, ACE uses the specified arc to perform post-processing and generates pwc data. If you do not specify the primary pin name, ACE by default uses the first P/G pins listed as the primary P/G pins.

SIGCAP_FACTOR <value>

Specifies the fraction of the signal net capacitance to be included in P/G pin cap (<value> between 0 and 1). The default value is 0.5.

SIGNAL_PARASITIC_C [0 | 1]

When SIGNAL_PARASITIC_C is turned On, signal net Cload is added to P/G net intrinsic capacitance. The default is 0, off.

Simulator_Command_Option <option>

Supports use of NSpice, HSpice, Eldo, and Spectre tools. Adding this parameter to the ACE configuration file appends specified user options to the Spice simulation command line.

SPICE_SIMULATOR <option>

Supports the '-spectre' option when invoking the “Finesim” option simulator. In addition, the following ACE configuration file keyword should be specified:

```
ACE_OPTION {  
    SIMULATOR_COMMAND_OPTION -spectre  
}
```

Subckt <file>

Defines the Spice subckt netlist for the cell. The P/G pin names in the netlist must be consistent with the pin names specified in External_Power/Ground_Net.

SweepValue <v1> <v2> ...

Specifies voltage sweep values for PWCAP data generation when the ACE '-pwc' option is specified. If you do not specify voltage sweep values, ACE by default uses 11 standard Vdd values to do scaling (10%, 20%, ..., 100%, and 110% of Vdd).

Switch_Subckt [<switch_subckt_name> | <switch_subckt_filename>]

Defines the name of a header/footer switch subckt or a Spice-format switch subcircuit filename. Note that the subckt(s) definition must be encapsulated in file(s) specified by SUBCKT or INCLUDE constructs. For low power cells that have the switch built-in, you must declare the switch subckt using this keyword. For cells that do not have header/footer switches, you do not need to specify this keyword.

Temperature <value>

Specifies the temperature value for MOSFET characterization.

Toggle_Rate_Assignment

Assigns different toggle rates to different sub-circuits inside a block or IP, in addition to the top level. To use this feature, add the following in the ACE configuration file:

```
TOGGLE_RATE_ASSIGNMENT {
    <subcircuit_1> <toggle_rate1>
    <subcircuit_2> <toggle_rate2>
    ...
}
```

VP_Mapping (<LEF_pin> <Spice_pin>) [...]

Defines the P/G pin mapping between LEF and Spice netlists. If the P/G pin names are different in LEF and Spice, you need to use this keyword to define the mapping output result file. ACE uses Spice pins during characterization, and uses the LEF pin name in the output result file.

VddValue <vdd_domain> <voltage> [<vdd_domain> <voltage> ...]

Specifies Vdd domain name(s) and corresponding value(s). These values are used during MOSFET characterization. The Vdd domain names must be consistent with those specified in External_Power_Net.

ACE supports multiple nominal Vdd characterization using the following syntax:

```
VDDVALUE <vdd_pin1 value1a> [<vdd_pin2 value1b>...]
VDDVALUE <vdd_pin1 value2b> [<vdd_pin2 value2b>...]
```

In this syntax <*value1a> and <*value1b> represent different voltage corners of multiple nominal characterization.

VP_Pairing (<internal_net1> <external_net1>) (...)

Defines P/G pairs for internal and external nets. If there are any switches present in the cell, you must specify the P/G pairs using this keyword. Otherwise this keyword is not necessary. See also use of Internal/External_Power_Net and Internal/External_Ground_Net keywords.

VTH_FACTOR <value>

The scaling factor VTH_FACTOR specifies whether signal capacitance is to be added to Vdd pin intrinsic capacitance. When SIGNAL_PARASITIC_C is set to 1, for each Vdd domain, if VDD_VALUE > VTH_FACTOR*NOMINAL_VDD, the signal net Cload for that domain is added to P/G pin capacitance accordingly. The VTH_FACTOR <value> must be a floating point number between 0 and 1. Default is 0.5.

WRAPSUB <subckt_name> [...]

When a subckt name is specified using this keyword, ACE looks for 'm' instead of 'x' as the leading character in the element line, for instances in the HSIM-wrapper processed netlist. This rule only applies to specified subckts. Other subckts still follow the normal SPICE rule, which uses 'x' as the leading character for subckt instances.

XMOS_INST_PARAM [ON|OFF]

When set, parses all instance parameters of the first XMOS instance in a table of unique size listings. The default value is off.

XMOS_PARAM_MAPPING ? width <w>? ?length <l>?

Allows specification of “width” and “length” options for devices in the device model file.

Running ACE Characterization

After preparing the ACE input configuration file as described above (*<cellname>.smin*), the command for invoking ACE from a UNIX command line is:

```
ace [-d] [-o <output_file>] [-toggle_rate <On_fraction>] [-pwc ]  
<cellname>.smin
```

where

-o <output_file>: specifies output filename for the CDEV file

-toggle_rate <On_fraction>: controls the fraction of On time for intrinsic MOSFETs, and thus their effective capacitance (ESC) value. The <On_fraction> should be between 0 and 1. A larger toggle_rate results in a smaller intrinsic ESC value, since it means more transistors are switching and acting less like capacitance.

-pwc: specifies generation of PWL capacitance data for low power analysis.

Output Result Files

ACE generates three files in the APL result directory:

- <cellname>.mcap - default ACE output file, CDEV file format (pin-based capacitance), to be imported into RedHawk
- <cellname>.ace.mmx - more detailed pin capacitance data (intentional, intrinsic, parasitic) to be imported into MMX
- <cellname>.ace.decap - detailed intentional decap information, including a list of each unique sized MOS decap, its sizes (W, L), model name, decap value, and the P/G pins it connects to. Another list is the total decap contribution to each P/G arc, with each decap cell's instance count.

AVM Datasheet Characterization

For dynamic analysis, if APL characterizations are not available for some memory cells, RedHawk can automatically collect the necessary data from the design, LIB files and the GSR and create an AVM (Apache Virtual Model) configuration file in *adsRpt/avm.conf*, which describes parameters such as read/write/standby mode, power consumption, access time, setup time, load information, and current waveform type, for each memory type. RedHawk internally uses this config file to generate and use rule-based current profiles, leakage current and decoupling capacitance for memory blocks. The AVM utility then creates triangular current profiles for these memory cells that have no APL data.

The GSR keyword that invokes automated AVM characterization is:

```
LIB2AVM [ 0 | 1 | 2 | 3 | 4 ]
```

By default (1), a double triangular-shaped profile is provided (to accurately simulate complicated profiles). When set to 2, a trapezoidal-shaped current profile is provided. When set to 3, load-dependent triangular-shaped profile is provided. A value of 4 provides load-dependent trapezoidal-shaped current profile. Any available APL memory characterization data overrides internally-generated AVM data. A value of 0 turns off automated AVM config file generation. Memories are identified based on the following attributes in *.lib for AVM generation: ‘memory()’, or ‘timing(){mode...}’. If multiple internal power tables with different related pins are available, the Cpd value calculation is prioritized as:

1. active pins (VCD)

2. if a related pin is a clock pin the power table with related pin clock will be used for Cpd calculation.
3. if none of the related pins is active and no clock pin the power table leading to the highest Cpd will be used.

Some key features of AVM are:

- Can be run standalone, independent of the design (see next section).
- Applies to dynamic flow only.
- If Ipeak is specified, the base width is adjusted to honor both Ipeak and Cpd.
- Curve generation is enhanced to minimize the difference between the AVM results and user-specified Ipeak and Cpd.
- The tail current is set to 0 for all curves.
- The % difference between the AVM results and user-specified Ipeak and Cpd values is reported during the AVM run.
- To invoke the accurate characterization mode, include the following keyword:

```
CHARACTERIZATION_MODE ACCURATE
```

- To check the base width of the AVM current profile, use the keyword:

```
MAX_FREQ <frequency in Hz>
```

AVM base width at nominal Vdd should be < 1/MAX_FREQ.

- Memories with multi-state power values are intelligently handled by lib2avm, in that only the states present in the VCD file are considered for power calculation. The power values in the .lib corresponding to the Boolean state of the VCD determine the total power of the instance. The GSR keyword ‘LIB2AVM_MSTATE 1’ should be set.
- AVM peak time is modified in accordance with any voltage derating that occurs. Peak voltage derating maintains the same power level, which extends the time base and peak of the triangular waveform.

Running AVM standalone

The LIB2AVM utility can be run standalone. By providing transition time and load information, memory cells can be characterized separately, from the UNIX command line, using the following syntax:

```
lib2avm <lib2avm_config_filename> -l <cell_list_filename>
```

You can specify the equivalent gate count for the block using the keyword ‘EQUIV_GATE_COUNT <count>’ in the LIB2AVM configuration file.

AVM Configuration File

Before running AVM characterization a configuration file must be prepared. The basic AVM configuration file syntax, which supports multiple nominal Vdd/Vss domain cells, is as follows:

```
<memory name> {
    MEMORY_TYPE [ SRAM | BLOCK | DRAM | CAM | ROM | RegFile | MSRAM | IP | BLOCK
                  | IO | MEMORY ]
    EQUIV_GATE_COUNT <count>
    Cload <load_in_F>
    VDD_PIN <VDD_pin1> <VDD_pin2> ...
    GND_PIN <GND_pin1> <GND_pin2> ...
    CHARACTERIZATION_MODE [ accurate | ultra_accurate | PWL |
    PWL1 ]
    PROCESS [ SS | TT | FF ]
    C_decap {
```

```

        <VDD_pin_m> <GND_pin_n> <C_decap-ARC1>
        <VDD_pin_o> <GND_pin_p> <C_decap-ARC2>
        ...
    }
c_decap [ (<Volt_a2> <Volt_b2> ...) ] {
...
leakage_I [ (<Volt_a1> <Volt_b1> ...) ] {
<vdd_pin_m><gnd_pin_n> <leakage_i_ARC1>
<vdd_pin_o><gnd_pin_p> <leakage_i_ARC2>
...
}
leakage_I [ (<Volt_a2> <Volt_b2> ...) ] {
...
}
VDD <voltage1_VDD_pin1> <voltage1_VDD_pin2> {
    ck2q_delay <delay_in_sec>
    tr_q <time_in_sec>
    tf_q <time_in_sec>
    tsu <time_in_sec>
    Cpd_read {
        <VDD_pin_m> <GND_pin_n> <Cpd_read-ARC1>
        <VDD_pin_o> <GND_pin_p> <Cpd_read-ARC2>
        ...
    }
    Cpd_write {
        <VDD_pin_m> <GND_pin_n> <Cpd_read-ARC1>
        <VDD_pin_o> <GND_pin_p> <Cpd_read-ARC2>
        ...
    }
    Cpd_standby {
        <VDD_pin_m> <GND_pin_n> <Cpd_read-ARC1>
        <VDD_pin_o> <GND_pin_p> <Cpd_read-ARC2>
        ...
    }
}
VDD <voltage2_VDD_pin1> <voltage2_VDD_pin2> {
    ck2q_delay <delay_in_sec>
    tr_q <time_in_sec>
    tf_q <time_in_sec>
    tsu <time_in_sec>
    Cpd_read {
        <VDD_pin_m> <GND_pin_n> <Cpd_read-ARC1>
        <VDD_pin_o> <GND_pin_p> <Cpd_read-ARC2>
        ...
    }
    Cpd_write {
        <VDD_pin_m> <GND_pin_n> <Cpd_read-ARC1>
        <VDD_pin_o> <GND_pin_p> <Cpd_read-ARC2>
        ...
    }
    Cpd_standby {

```

```
<VDD_pin_m> <GND_pin_n> <Cpd_read-ARC1>
<VDD_pin_o> <GND_pin_p> <Cpd_read-ARC2>
...
}
...
}
Leakage_I {
    <VDD_pin_m> <GND_pin_n> <Leakage_I-ARC1>
    <VDD_pin_o> <GND_pin_p> <Leakage_I-ARC2>
...
}
...
}
```

where

<memory name>: name of memory macro as defined in LEF. Units follow the same conventions as in the GSR file (see Appendix C).

MEMORY_TYPE [SRAM | BLOCK |DRAM | CAM | ROM | RegFile | MSRAM | IP | BLOCK | IO | MEMORY]: identifies the type of item

If MEMORY_TYPE 'BLOCK' is specified, default values for 'tsu', 'tr_q', 'tf_q' are 200 ps, and 'Cpd_standby' is set to 10 pF. The default values for 'Cpd_read' and 'Cpd_write' are set to 20 pF;

If MEMORY_TYPE 'IO' is specified, a single triangular shape is used as the default;

If MEMORY_TYPE 'MEMORY' is specified, internally the same as SRAM, this is the generic type for lib2avm purpose;

The default values can be changed by specifying different values for the parameters in the AVM config file.

EQUIV_GATE_COUNT <count>: specifies the equivalent gate count, which can be derived from the number of bit cells. For example, a 512x70 memory should be 35840 equivalent gates.

Cload: loading capacitance on an output pin, in Farads. Default is 100 f F

VDD_PIN <VDD_pin1> <VDD_pin2>/ GND_PIN <GND_pin1> <GND_pin2>: By default, VDD_PIN and GND_PIN are optional, and their default values are VDD and GND respectively. If the power and/or ground names are different or multiples, you must specify these keywords. The Cpd and decap must be specified per power pin. If there is a single ground pin, the current profile of the ground pin is the sum of all the Vdd pins. If there are multiple grounds, the current profile is defined according to the P/G arc file described in the section "P/G Arc Definitions in Custom LIB Files", page 3-20.

CHARACTERIZATION_MODE : set to 'accurate' creates a waveform with 400 time points (equally-spaced samples) to store the APL current profile. 'ultra_accurate' mode has 1000 time points (equally-spaced samples), and PWL creates a Piece-Wise-Linear waveform. If set to PWL1, AVM can handle "MEMORY_TYPE RegFile" For all other CHARACTERIZATION_MODE, waveform is created as "MEMORY_TYPE SRAM".If CHARACTERIZATION_MODE is not set, the default waveform generated by AVM has 50 time points (equally-spaced samples).

PROCESS : specifies a process type, where SS is slow, TT is typical, and FF is fast. Required.

C_decap <VDD_pin_m> <GND_pin_n> <C_decap-ARCn>: specifies decap values for all P/G pin arc combinations. <Volt_a1> and <Volt_b1> represent different voltage corners for multiple nominal characterization, and <vdd_pin_m> and <gnd_pin_n> represent the vdd-vss arc of the capacitance value specified.

VDD <voltage1_VDD_pin1> <voltage1_VDD_pin2>: specifies voltage combinations for various VDD pins

ck2q_delay: specifies clock-to-output pin delay (sec) for the associated VDD pin voltages

tr_q: specifies output pin rising transition time (sec) for the associated VDD pin voltages

tf_q: specifies output pin falling transition time (sec) for the associated VDD pin voltages

tsu: specifies setup time (sec) for the associated VDD pin voltages

Cpd_read <VDD_pin_m> <GND_pin_n> <C_decap-ARCn>: specifies Cpd_read values for all P/G pin arc combinations, in Farads

Cpd_write <VDD_pin_m> <GND_pin_n> <C_decap-ARCn>: specifies Cpd_write values for all P/G pin arc combinations, in Farads

Cpd_standby <VDD_pin_m> <GND_pin_n> <C_decap-ARCn>: specifies Cpd_standby values for all P/G pin arc combinations, in Farads

Additional optional AVM configuration file keywords are described below:

- Specifying voltage scaling factors:

```
AVM_VOLTAGES <num_voltages> <v_scale_factor1>
              <v_scale_factor2> ...
```

- Specifying trapezoidal or triangular current profiles. Default is triangular.

```
WAVEFORM_TYPE [trapezoidal | triangular]
```

- Specifying leakage current, in Amps.

```
leakage_i {
    <VDD_pin> <GND_pin> <current_in_Amps for ARC1>
    <VDD_pin> <GND_pin> <current_in_Amps for ARC2>
    ...
}
```

- Specifying peak current values for Write, in Amps, for triangular waveform model.

```
Peak_I_Write {
    <VDD_pin> <GND_pin> <current_in_Amps for ARC1>
    <VDD_pin> <GND_pin> <current_in_Amps for ARC2>
    ...
}
```

- Specifying peak current values for Read, in Amps, for triangular waveform model.

```
Peak_I_Read {
    <VDD_pin> <GND_pin> <current_in_Amps for ARC1>
    <VDD_pin> <GND_pin> <current_in_Amps for ARC2>
    ...
}
```

- Specifying peak current values for Standby, in Amps, for triangular waveform model.

```
Peak_I_Standby {
    <VDD_pin> <GND_pin> <current_in_Amps for ARC1>
    <VDD_pin> <GND_pin> <current_in_Amps for ARC2>
```

- Specifying peak times for Write, in seconds, for triangular waveform model.

```
Peak_T_Write {
    <VDD_pin> <GND_pin> <time_in_sec for ARC1>
    <VDD_pin> <GND_pin> <time_in_sec for ARC2>
    ...
}
```

- Specifying peak times for Read, in seconds, for triangular waveform model.

```
Peak_T_Read {
    <VDD_pin> <GND_pin> <time_in_sec for ARC1>
    <VDD_pin> <GND_pin> <time_in_sec for ARC2>
    ...
}
```

- Specifying peak times for Standby, in seconds, for triangular waveform model.

```
Peak_T_Standby {
    <VDD_pin> <GND_pin> <time_in_sec_ARC1>
    <VDD_pin> <GND_pin> <time_in_sec_ARC2>
    ...
}
```

- Specifying minimum period of memory from starting edge for calculating Eff VDD, in seconds.

```
TMIN <time_sec>
```

- Specifying delay derating factor for delta voltage:

```
Delay_Derating <factor range between 1 to 10>
```

- Specifying capacitance of the most primitive gate, such as an inverter:

```
Gate_cap <cap in F>
```

- Specifying design-dependent operating temperature in degrees C (default is 25):

```
TEMPERATURE <actual_temperature>
```

- Specifying average power for high block activity:

```
Avg_High_Power <power in Watts>
```

- Specifying average power for low block activity period:

```
Avg_Low_Power <power in Watts>
```

- Specifying operating frequency:

```
Freq <freq in Hz>
```

- Specifying the fraction of clock power consumption relative to total block power:

```
Cknt_Power_Ratio <power fraction>
```

- Specifying the fraction of sequential element power consumption relative to total block power:

```
Seq_Power_Ratio <power fraction>
```

- Specifying charge_ratio value, for triangular waveform model.

```
CHARGE_RATIO <factor between 0.0 to 1.0>
```

Specifies the second triangular waveform's area (charge) divided by the total area (charge). The default value is 0.7.

- Specifying the data version.

```
DATAVERSION [ 7v1 | pre7v1 ]
```

Specifies the data version of the *vmemory.current* file relative to RedHawk version 7.1. The default is pre7v1.

- Specifying the maximum frequency for checking the base width at nominal Vdd value; the base width should be < 1/MAX_FREQ.

MAX_FREQ <freq>

The following is a sample *avm* configuration file:

```
Cell1 {
    MEMORY_TYPE RegFile
    EQUIV_GATE_COUNT 6663
    Cload 2f
    VDD_PIN VDDPR VDDP
    GND_PIN VSS
    C_decap {
        VDDPR VSS 11.0022509561f
        VDDP VSS 11.0022509561f
    }
    VDD 1.08 1.08 {
        ck2q_delay 1537p
        tr_q 30.194p
        tf_q 27.621p
        tsu 523.65p
    Cpd_read {
        VDDPR VSS 1158.88203017832647f
        VDDP VSS 11.88203017832647f
    }
    Cpd_write {
        VDDPR VSS 2213.82030178326475f
        VDDP VSS 22.82030178326475f
    }
    Cpd_standby {
        VDDPR VSS 409.293552812071331f
        VDDP VSS 4.293552812071331f
    }
}
VDD 1.32 1.32 {
    ck2q_delay 2537p
    tr_q 40.194p
    tf_q 37.621p
    tsu 623.65p
    Cpd_read {
        VDDPR VSS 2158.88203017832647f
        VDDP VSS 18.88203017832647f
    }
    Cpd_write {
        VDDPR VSS 3213.82030178326475f
        VDDP VSS 13.82030178326475f
    }
    Cpd_standby {
        VDDPR VSS 509.293552812071331f
        VDDP VSS 9.293552812071331f
    }
}
```

```
        }  
    }
```

AVM Configuration File for Multi-state mode

If multiple state Boolean switching scenarios are specified, the configuration file must be modified. Note that multi-state mode supports only memory/IP cells, and not combination and sequential cells. If keywords 'Cpd_write', 'Cpd_read', 'Cpd_standby', 'leakage_I_write', 'leakage_I_read', 'peak_I_write', 'peak_I_read', 'C_decap_write', 'C_decap_read' are in the file, they are ignored. Two additional keywords should be added to the multi-state AVM configuration file, as follows:

CUSTOM_STATE_FILE <filename> (see [page E-928](#))

or

```
STATE_BOOLEAN <state_name> <Boolean_eq> <active_pin/>  
clock_pin? <output_pin>?  
...
```

and

```
Cpd <state_name> {  
<vdd_pin> <vss_pin> <arc_cap_F>  
...  
}
```

where

<state_name>: user-defined state name in CUSTOM_STATE_FILE

<vdd_pin> <vss_pin> : vdd/vss pins for specified state

<arc_cap_F>: capacitance for specified state and Vdd/Vss arc

In addition, several optional keywords for multi-state mode are available:

- Optional Peak current, in Amps.

```
peak_I <state name> {  
<vdd_pin> <vss_pin> <arc_current_A>  
...  
}
```

- Optional peak time, in seconds.

```
peak_T <state name> {  
<vdd_pin> <vss_pin> <peak_time_sec>  
...  
}
```

Sample AVM configuration file for multi-state

```
CUSTOM_STATE_FILE <custom_state_file>  
ABC_Mem1 {  
    ...  
    Cpd M0 {  
        VDD VSS 15pF  
    }  
    Cpd M1 {  
        VDD VSS 20pF  
    }  
    Cpd M2 {  
        VDD VSS 10pF
```

```
    }
    Cpd M3 {
        VDD VSS 10pF
    }
    ...
}
```

Running AVM

Once the AVM configuration file is created, run *avm* from the shell command:

```
avm <avm_configuration_file>
```

AVM Outputs

AVM generates the current profile (*vmemory.current*) and decap information (*vmemory.cdev*) from the datasheet specification, and automatically incorporates it into the RedHawk simulation. The percent difference between AVM results and user-specified Ipeak and Cpd also are reported for the AVM run. Other AVM output files are *avm.log*, *avm.warning* and *avm.error*.

Troubleshooting APL Problems

Checking the Configuration File

The following are recommended APL configuration file checks:

- Are all required keywords specified properly?
- Are Vdd values consistent with those in the GSR file?
- Are temperature values consistent with those in *.lib and the Tech files?
- Are specified names of Vdd and Gnd pins consistent with the Spice netlist?
- Are device sizes in the Spice netlist in um? If not, set the SIZE_SCALE keyword to a value such that the result is microns, such as 1e-6.

Common Problems

- Vector information missing in *.apache/adsLib.out*.
 - LIB file missing
 - Function statement or state table missing inside LIB file
- Spice netlist missing.
- Incorrect Spice model file
 - Missing parameter definitions
 - Wrong scale parameters used
 - Unsupported Spice cards used

- Trying to characterize unintended cells:
 - Doing current characterization on tie-off cells, antenna diodes, etc.
 - Doing characterization on decap cells without using the ‘-p’ option.
 - Trying to characterize complex cells without the required custom vectors
- Vectors used in the simulation not causing a rise/fall transition at the output.
- Insufficient disk space. Watch TMP_DIR.
- Monitor and sample count.

Debugging Command Line APL Errors

- Try debugging only one cell at a time. Create a *<cell>.list* file containing only one cell.
- Run characterization in non-LSF mode. You can see STDOUT messages.
- For current characterization, make sure that *.apache/adsLib.output* has the required vector data for the cell.
- Make sure that the Spice netlist exists for the cell in the SUBCKT file.
- Set DEBUG 1 flag in APL configuration file.
- Use verbose mode (-v) to get more detailed messages.
- Use debug mode (-d)
- Make sure that the function statement/state table is present in LIB file
- Look at the Spice netlist (*.apache/APL/<cell_name>.sp*)
- Look at the Spice waveforms using SV and make sure that you are seeing rise / fall transitions.
- Create a small Spice deck for the cell. Run it in NSpice and see whether it is creating the transitions.
 - *nspice cell.test.spi*
- Look at whether the failures have some common aspects
 - Look at the machine/platform/Queue of failing cells.
 - Look at the cell type/functionality
- If APL characterization failed with the following NSpice error

```
##error## Isolated node (node name) detected. Simulation stopped.
```

which means there is no DC path between the node and ground, you can add ‘OPTION gshunt=1e-9’ in the APL configuration file to avoid simulation failure.
- If APL samples cannot be generated due to incomplete data, RedHawk lists the names of the cells with missing data in the file *adsRpt/apache.refCell.noApISample*, and issues Warning message (ITG-022).

Sample APL Configuration File

The following is an example library APL configuration file, with comments explaining each entry.

```
#Specify the APL run mode, design dependent or design independent
APL_RUN_MODE DI

#Specify .lib files
LIB_FILES {
    /nfs/apl1/user_data/lib/IO.lib
```

```
/nfs/apl1/user_data/lib/ANALOG.lib
}

#Define design corner
DESIGN_CORNER {
{
    TEMP 25
    PROCESS TT
    VDD 1.0
    MODEL /nfs/apl1/model TT
}
{
    TEMP 125
    PROCESS FF
    VDD 1.1
    MODEL /nfs/apl1/model FF
}
}
#Specify parallel run platform
GRID_TYPE LSF

#Specify the job submit command for parallel run
BATCH_QUEUING_COMMAND bsub

#Specify LSF or Sun Grid job submit options
BATCH_QUEUING_OPTIONS -r -R select [type==any]

#Specify the path of the LSF or Sun Grid binaries
EXEC_PATH /appls/lsf/6.0/linux2.4-glibc2.3-x86/bin

# Specify LSF job submission mode: use bsub or LSF API
LSF_SUBMIT_MODE 1

#Specify the maximum number of simultaneous jobs
JOB_COUNT 20

#Specify Spice subckt netlist file
SPICE_SUBCKT {
/nfs/apl1/user_data/spice/subckt/spice.lib
}

#Specify Spice Include files (netlist, option, model, etc)
INCLUDE
{
/nfs/apl1/model/model.typ
}

#Specify your working directory
WORKING_DIR /nfs/apl1/apldi_test

#Specify the transistor dimension scale factor
```

```
SIZE_SCALE 1

#Nominal Vdd value
VDD 1.5

#Vdd names
VDD_NAME VDD
#Ground names
GND_NAME GND1
#Specify the directory for temporary working files
TMP_DIR /nfs/apl1/apldi_test/temp
```

Chapter 10

Memory and I/O Modeling

Introduction

RedHawk determines a chip's Dynamic Voltage Drop (DvD) using high time resolution full-chip transient analysis, including package models, on-chip RLC extraction, and cell/macro VDD/VSS switching current profiles. In 90 nm scale and beyond process technologies, memories and I/O cells play an increasing role and are key contributors to large switching currents

Embedded memories are taking an increasing share of real estate in SoC designs. Given their increasing complexity and power consumption, it is necessary to model embedded memory blocks accurately to consider their impact in a full-chip dynamic voltage drop (DvD) analysis. Voltage drop inside memory blocks poses a problem for deep sub-micron designs. Also, high power demand from memory instances can cause DvD in the surrounding logic, thus impacting their timing and functionality. At a full-chip level, the concern lies in ensuring that the memory blocks get adequate power based on their specifications.

I/O cells are key contributors to the large switching currents in I/O VDD and I/O VSS. While the I/O VDD is often isolated from the core VDD, I/O VSS can be shared with the core VSS. If the VSS is shared, the large current from the simultaneous switching of the I/O buffers can contribute significantly to the VSS bounce of the core, possibly resulting in functional or timing failures for the chip. The I/O buffers are usually only available in LEF and GDSII data.

RedHawk offers several options for modeling memories and I/Os, with varying levels of abstraction—from fully detailed models for accurate block level analysis to more abstract models for full-chip dynamic analysis. Table 10-1 highlights these different options.

Table 10-1 RedHawk memory and I/O modeling options

Model Option	Power Grid	Sink, Cap Locations	Memory Current, Decap Values	I/O Current Values
Black-box	LEF / Pin-based	At the pins	Based on .lib, AVM, sim2iprof specifications	Based on .lib / power specified
Grey box - considering power grid	As in layout	At the pins	Based on .lib, AVM, sim2iprof specifications.	Based on .lib / power specified
White box - considering power grid and internal current sinks	As in layout	Distributed internally	Using .lib, AVM, sim2iprof specifications.	Using .lib / power specified / APL

This chapter describes **RedHawk**'s memory and I/O characterization flow, including input data requirements, data preparation, and usage model. Memory modeling for use in **RedHawk** is partitioned into two parts: (1) extraction of the power grid and assignment of current sources and decaps, and (2) generation of the switching current profiles, leakage currents, and decap values. For the first part, several utilities are available to perform the extraction and current source/decap assignment for the required levels of abstraction. For the second part, several methods are available to capture the power draw, average static

current, and dynamic switching current. Figure 10-1 illustrates this modeling methodology for memory blocks.

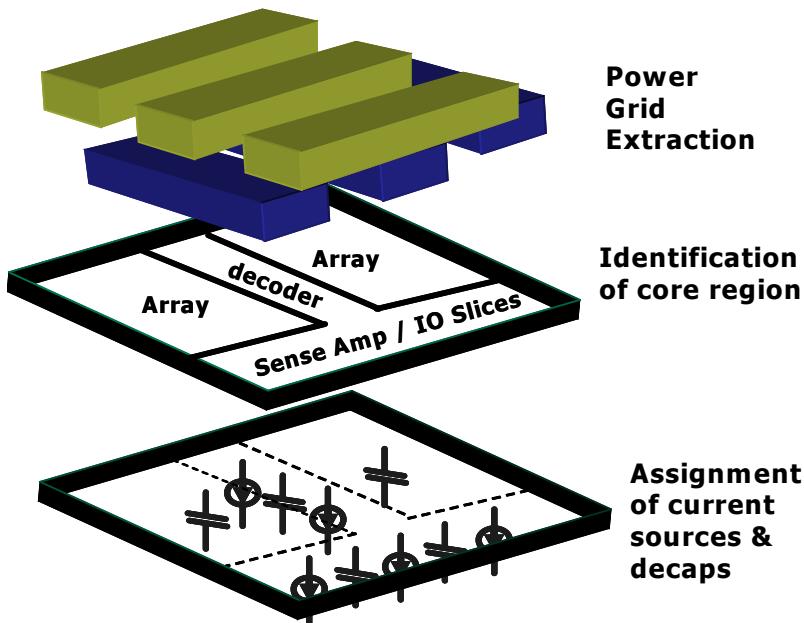


Figure 10-1 RedHawk modeling methodology for memories and I/P

Besides I/O buffers, there are VDD/VSS pads, gap filling pads, bumps, and RDL (Redistribution Layer) data that often are only available in GDSII format, without any detailed routing information in DEF. RedHawk can take the input data in GDSII form and convert it to DEF using `gds2def/gds2rh`, as shown in Figure 10-2. See [section "gds2rh/gds2def"](#), page E-875, and [section "gds2rh -m and gds2def -m"](#), page E-915 for details.

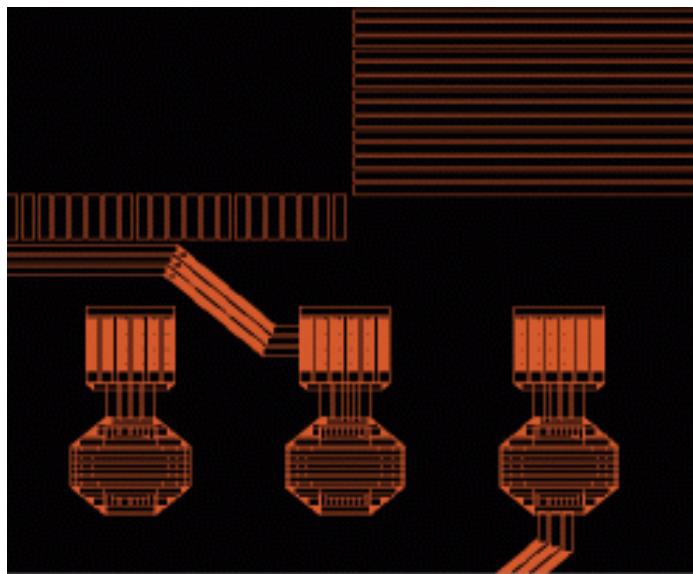


Figure 10-2 C4 bump and RDL layer converted from GDSII using the gds2def/gds2rh utility

Memory and I/O Modeling Methodology

Black-box Modeling

For the initial set of runs, especially at the full-chip level, it is a good idea to abstract all memories, I/O cells and VDD/VSS pad cells, as black-boxes to ensure correct connectivity and to debug data-related issues. Black-boxed memories and I/Os consume significantly less physical memory and run much faster, allowing successive runs during the initial data preparation stages.

To run memories and I/Os as black-boxed abstractions, no additional data preparation is required. No design representation (that is, DEF information) for the memory blocks and I/Os is required. However, LEF and LIB models should exist for all memory blocks and I/Os. The power pins from the LEF files are considered in the memory and I/O models.

For black-box modeling, the memories are considered without their internal power grids and all current demand is distributed among the power pins of the blocks as defined in their LEF files.

Sometimes the black-box modeling of the I/Os and VDD/VSS pads have incomplete geometry information on the pins, which can result in problems, such as VDD/VSS shorts or very large static IR drop. In this case, use a more detailed level of abstraction for I/O modeling.

Pin and Grid-Based Abstractions

In grid-based modeling, the memories and I/Os are considered with their internal power grids. The current draw is distributed among the power pins of the block that connect to the top level grid. This method of modeling is particularly useful in designs for which full-chip power grid connectivity is maintained through the power grid of the memory blocks and I/Os. For example, if the top-level power grid of a design is in metal layers 3 and 4, and the memory blocks and I/Os have power grids in metal layers 3 and 4, then the inclusion of the memory blocks and I/Os along with their power grids will ensure connectivity for the power routes in metal layers 3 and 4. If these blocks were black-boxed, then the continuity and connectivity of the power routes in these layers would be lost. Figure 10-3 illustrates this situation for memories.

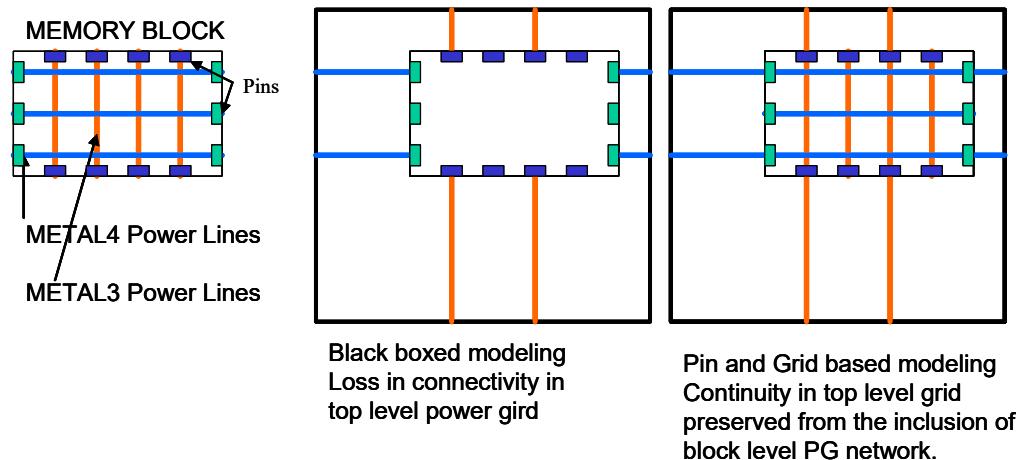


Figure 10-3 Modeling memories with and without their internal power networks.

However, in a grid-based modeling framework, the current draw is considered at the pins of the memory block. So it is useful when faster run-times are required or when memory capacity is an issue. Considering the current demand inside the memories or I/Os down to the lower levels of metals, as opposed to at the pins, adds significant overhead in computation time and physical memory usage. But this method has the advantage over the black-boxed modeling in that it considers the memory and I/O power grids during analysis.

To abstract the memory and I/O power grids for grid-based modeling, use the *gds2def/gds2rh* utility to create the necessary design files from the GDSII. The *gds2def/gds2rh* utility requires the following input data:

- Configuration file. Please refer to the *gds2def/gds2rh* information for memories in [section "gds2rh -m and gds2def -m", page E-915](#), for details. Please refer to the *gds2def/gds2rh* information for I/Os in the following section.
- Layer map file containing the information for mapping the GDSII layer numbers to the corresponding layers defined in the LEF/DEF.
- GDSII file for the memory blocks and I/Os.
- LEF file for the memory blocks and I/Os (optional in GDSII flow). Needed for I/O cells that draw current; but not needed for VDD/VSS pads that do not draw current.

The output from the utility *gds2def/gds2rh* is a DEF file (*<toplevel>.def*) containing the power network information for the memory and I/O blocks. This DEF file should be included in the list of DEF files given as input to **RedHawk** during dynamic analysis. In addition, if the LEF file is available and defined, a new LEF file (*<toplevel>_adsgds.lef*) containing information, such as the PINS section, is created.

Note: Neither the Spice-based flow nor the contact-based pin creation flow are supported in GDS2RH. Customers should use the Totem transistor-level flow.

The next sections describe in detail the modeling of memory blocks and I/O cells.

Detailed Memory Block Modeling

Extraction

In detailed modeling, the power grid of a memory block is extracted and current sources, along with decoupling capacitances, are placed at appropriate locations *inside* the memory. This allows for an accurate consideration of current flow inside the memory blocks. It also allows for modeling of a memory's dynamic switching effect on its surrounding logic.

For detailed modeling of memories, use the *gds2def -m* or *gds2rh -m* to generate the detailed view. The *gds2def -m/gds2rh -m* utility requires the following input data:

- Configuration file. Please refer to *gds2def -m* or *gds2rh -m* information in [section "gds2rh -m and gds2def -m", page E-915](#) for details.
- Layer map file containing the information for mapping the GDSII layer numbers to the corresponding layers defined in the LEF/DEF.
- GDSII file for the memory blocks.
- LEF file for the memory blocks required for extracting pin information.

The output from *gds2def -m / gds2rh -m* is a set of DEF, LEF, and *.pratio files for each memory block, which contain placement information for the current sources and created decaps, power/ground routing geometries, cell abstraction, and each cell's relative power

dissipation. The *.pratio* files specify weighting factors for the current distribution for P/G pins in the memory. These files should be included as input files for RedHawk analysis.

For greater accuracy, you should provide the hierarchical SPICE netlist for the memory block with x,y coordinate information for the transistors:

The size and location of the transistors in the Spice netlist control the placement and properties of the current sources inside the memory model. For a full-chip transient dynamic analysis, it is computationally impossible to include all transistors in the simulation. Hence, *gds2def -m / gds2rh -m* groups transistors together, based on their size and location, to form virtual cells that provide a level of abstraction without much loss in accuracy.

The inclusion of the memory power grids and the consideration of current sources connected to the bottom layers of metal allow for complete modeling at the full-chip level and for an accurate transient dynamic analysis. Full-chip run-times or physical memory usage is not affected when the number of memory blocks is small. However, run-time and physical memory usage can be an issue for a large number of memory blocks. This is due to the large number of nodes introduced from the inclusion of the lower levels of metals in the block P/G networks and the creation of large number of passive elements. To circumvent such scenarios, *gds2def -m / gds2rh -m* has some advanced features that allow for selective extraction of metal layers. Two such procedures are described below.

- **Extraction of specified layers only:**

Users can choose to extract only certain layers from the memory GDSII file by using the keyword

```
EXTRACTION_STARTING_LAYER <lowest_metal_layer> ? traceall?
```

in the *gds2def -m / gds2rh -m* configuration file. The P/G network in the starting layer and higher layers, then are extracted. The current sources and decoupling capacitances are hooked up to the lowest layer of specified metal.

If layers lower than the starting layer have been used to connect to higher layers, use the ‘traceall’ option, which considers connected elements on all metal layers during network tracing. However, if the option ‘traceall’ is specified, note that GDS2RH has different behavior than GDS2DEF. GDS2DEF, after tracing through all metal layers, discards all geometries below the specified EXTRACTION_STARTING_LAYER. GDS2RH internally invokes the APACHE_PHYSICAL_MODEL approach, which does not discard geometries at all layers below the EXTRACTION_STARTING_LAYER, but keeps connections between the pins (current sources/sinks) created at the ESL layer, through lower metal/via layers, for better voltage-drop modeling of the block.

- **Selective extraction of metals from certain regions.**

Typically, the current draw from the power grid inside the memory array regions is small and can be ignored without much loss in accuracy for a voltage drop analysis, especially at the full-chip level. If the geometries and segments from the lower metal layers in the memory array regions only form local connections to the devices in the array region, then their exclusion will not affect the full-chip analysis. Figure 10-4 shows the metal 2 geometries inside the memory core region of a memory block. As evident from the figure, the metal 2 geometries do not form power routes, but serve more as local interconnects. These geometries, if excluded from the analysis, do not affect the power analysis accuracy, but can provide significant savings in the node count, thereby allowing for considerable savings in physical memory usage and run-times.

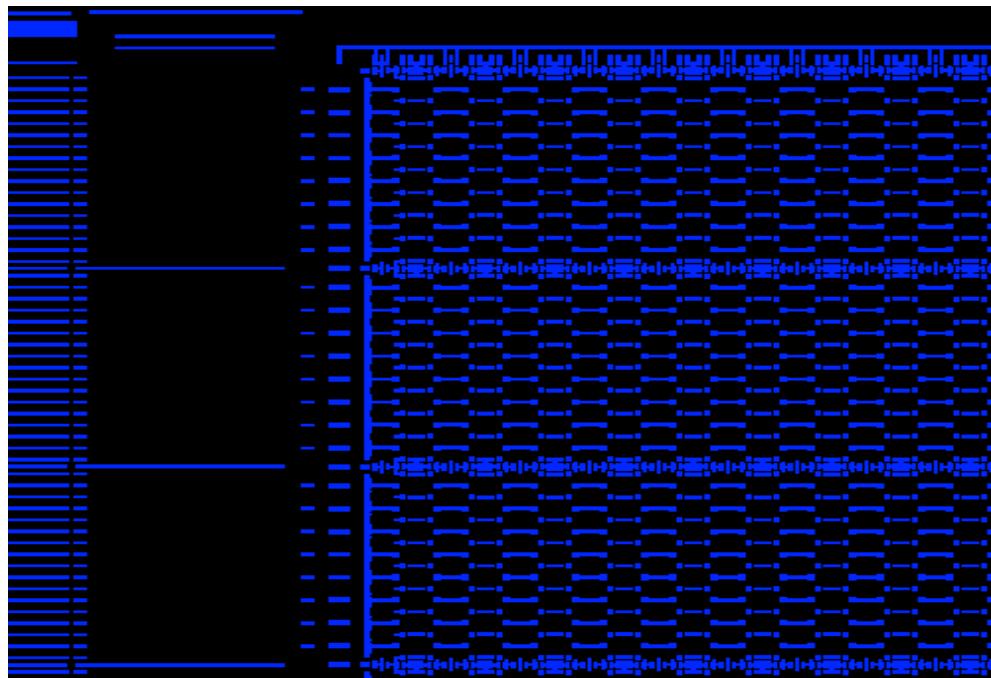


Figure 10-4 Metal 2 geometries in the memory core region of a memory

The `gds2def -m / gds2rh -m` utility can automatically identify the memory bit cell and subsequently identify the memory array region from GDSII. It can then extract only certain layers from the array region, while extracting all other layers from the other parts of the memory. This helps to preserve all metal geometries in the high power consumption regions, such as row decoders, drivers, and sense-amplifiers, while preserving the connectivity of the memory block to the top-level power grids in all the layers.

To utilize these capabilities, the keyword `CORE_EXTRACTION_STARTING_LAYER <layer_name>` should be specified in the `gds2def -m / gds2rh -m` configuration file. This directs the `gds2def -m / gds2rh -m` utility to extract all layers including and above the specified layer from the memory array region. For the rest of the memory block, the layer specified in the keyword `EXTRACTION_STARTING_LAYER` is honored. If either of these two keywords is not specified, then by default all layers, starting from the lowest layer specified in the layer map file, are extracted in the respective regions. If `EXTRACTION_STARTING_LAYER` is specified and `CORE_EXTRACTION_STARTING_LAYER` is not, then the `gds2def -m / gds2rh -m` utility extracts all layers starting two layers above the layer specified in `EXTRACTION_STARTING_LAYER`.

To use the `CORE_EXTRACTION_STARTING_LAYER` option requires proper identification of the memory array region, which can be achieved in following three methods.

- By using the keyword `MEMORY_CELL auto_detect <cellname>`, for bit cells in the Spice netlist, you can use the automatic cell detection capability built into `gds2def -m / gds2rh -m`.
- By using the keyword `MEMORY_BIT_CELL auto_detect <cellname>`, for bit cells in GDSII, you can use the automatic bit cell detection capability built into `gds2def -m / gds2rh -m`.
- You can specify the name(s) of the memory bit cell(s) using the keyword

`MEMORY_BIT_CELL { }` and `gds2def -m / gds2rh -m` identifies the memory array region from the location of the specified bit-cells in the GDSII file.

- For certain complex memories, the auto-detection of the memory core region can be incorrect. In this case, you can choose to explicitly specify the rectangular regions that make up the memory core region through the keyword `MEMORY_CORE_REGIONS <file_name>`, where the specified file lists all the rectangular regions.

Figure 10-5 illustrates the extraction of metal 1 P/G grid network of a memory for: (a) all regions of the block, and (b) all regions except the array region of the memory block. Only metal layer 3 and above are extracted.

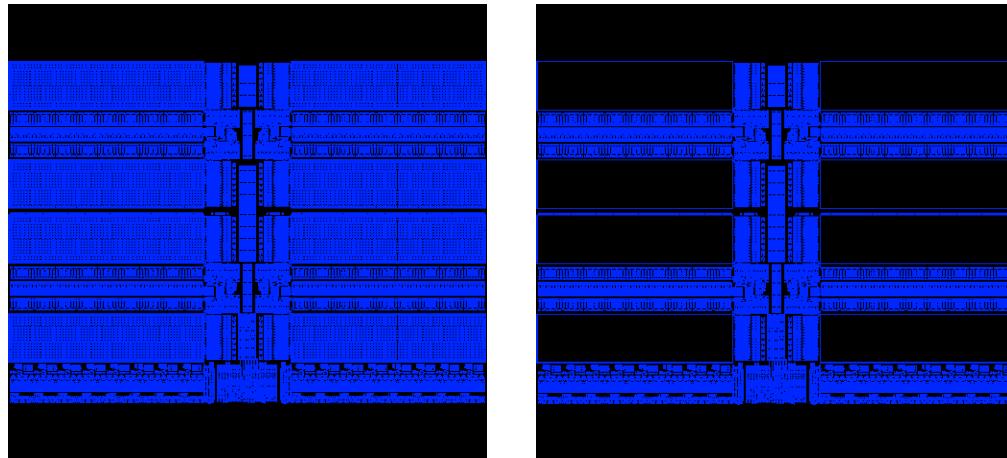


Figure 10-5 Global and selective extraction of metal1 P/G network for a memory block.

GDS2DEF/ GDS2RH Configuration File for Memories

This section summarizes the configuration file keywords needed for the memory modeling modes described previously. See section "[Creating the gds2rh/gds2def Configuration File](#)", page E-878, for more detailed information on keyword syntax and usage for memories.

Required GDS2DEF/ GDS2RH keywords

- `TOP_CELL`
- `GDS_FILE`
- `GDS_MAP_FILE`
- `LEF_FILE`
- `VDD_NETS`
- `GND_NETS`

Optional GDS2DEF/GDS2RH keywords:

- `OUTPUT_DIRECTORY`
- `DATATYPE`
- `NET_NAME_CASE_SENSITIVE 1`
- `EXTRACTION_STARTING_LAYER`
- `GENERATE_PLOC`

- LEF_PIN_POWER
- USE_LEF_PINS_FOR_TRACING
- BUSBITCHARS

gds2def -m/ gds2rh -m Configuration File Syntax

The following section describes the *gds2def -m / gds2rh -m* configuration file keyword syntax, both those that are required first to run the program, and then the optional ones.

Required keywords:

```
TOP_CELL $cell
GDS_FILE $cell.gds
GDS_MAP_FILE layer_map_file
LEF_FILE memory_lef_file
VDD_NETS
{
<power_net_name_to_be_used_in_output_DEF> {
    <power_net_name> @ <gds_layer_number>
    <gds_x_position>  <gds_y_position>
    (...)

    <power_pin_name_in_LEF>
    <power_net_name_in_GDS>
    (...)

}
}

GND_NETS
{
<ground_net_name_to_be_used_in_output_DEF> {
    <ground_net_name> @ <gds_layer_number>
    <gds_x_position>  <gds_y_position>
    (...)

    <ground_pin_name_in_LEF>
    <ground_net_name_in_GDS>
    (...)

}
}
```

Optional keywords:

```
DATATYPE
NET_NAME_CASE_SENSITIVE 1
SPICE_NETLIST
SPICE_XY_SCALE
MEMORY_CELL [ auto_detect | OFF ]
MEMORY_CELL {
    <names of memory cell subcircuit in core array>
}
NMOS_MODEL_NAME {
    <NMOS model names in Spice netlist>
}
...
PMOS_MODEL_NAME {
    <PMOS model names in Spice netlist>
}
```

```
...
WORD_LINE_DIMENSION <number of word lines>
BUSBITCHARS <char>
USE_LEF_PINS_FOR_TRACING 1
MEMORY_CORE_REGIONS <file name>
MEMORY_BIT_CELL [ auto_detect | OFF ]
MEMORY_BIT_CELL {
    <names of the memory bit cell(s) in core array>
}
EXTRACTION_STARTING_LAYER <lowest metal layer> [traceall]
CORE_EXTRACTION_STARTING_LAYER <lowest metal layer-memory>
```

The following keywords are related to Switch Memory modeling:

- [DEFINE_SWITCH_CELLS](#) - see usage in [section "DEFINE_SWITCH_CELLS", page E-900](#)
- [EXTRACT_SWITCH_CELLS](#) - see usage in [section "EXTRACT_SWITCH_CELLS", page E-900](#)
- LEF_FILES
- SWITCH_CELLS
- VP_PAIRS

Current Profile Generation

Static Analysis

For static analysis, a power number is required to estimate the current drawn in the memories. [RedHawk](#) can estimate the number directly from the library models of the memories, or you can choose to specify the power numbers through the GSR keyword, BLOCK_POWER_FOR_SCALING. The triangular waveform based on .lib power data is scaled according to the value of BLOCK_POWER_FOR_SCALING.

You can estimate the power of the memory from (a) datasheets of the memory blocks, (b) the memory vendor or memory design teams, or (c) other power estimation tools. The power you estimate should consider different modes of operation of the memory block and should also include the memory leakage current.

Dynamic Analysis

Dynamic analysis in [RedHawk](#) is a true transient analysis. Thus for every instance, current profiles are needed as a function of time for each mode of operation. There are four different ways current profiles can be generated for memories.

- **Triangular profiles using static average power values.** This form of modeling is the least accurate, but requires less effort and can be used if the other more accurate means of providing the current profiles are not available. Based on the static average power number, [RedHawk](#) generates single or double triangular current profiles.
- **Rule-based switching current profile generation.** The AVM (Apache Virtual Memory) utility generates switching current profiles for different modes of operation. This utility requires a configuration file, which is described in the [section "Memory and IP Characterization", page 9-263](#). The configuration file describes power consumption, access time, setup time, and load information for each of the different operation modes. AVM generates rule-based current profiles that are tailored for different families of embedded memories such as SRAMs, Register Files, or DRAMs. This utility also generates leakage current and the decoupling capacitance information for

the memory blocks. AVM can be run either outside RedHawk using the command

```
avm <avm_configuration_file>
```

or within RedHawk through the GUI or TCL interface.

- **Accurate switching current profile generation.** The *sim2iprof* utility uses third-party simulation output files, such as *fsdb*, *hout*, *tr0*, or *pwl* format files as inputs to obtain Read/Write/Standy mode data for memories, and generates accurate current profiles in a *cell.current* file for RedHawk power analysis. For details on using *sim2iprof*, see section "sim2iprof", page E-922.

Detailed I/O Cell Modeling

Extraction

In detailed modeling, the power grid of an I/O block is extracted and current sources along with decoupling capacitances, are placed at appropriate locations *inside* the I/O. This allows for an accurate consideration of current flow inside the I/O blocks. It also allows for modeling of an I/O's dynamic switching effect on its surrounding logic.

Detailed modeling of I/Os is similar to memories and requires the *gds2def/gds2rh* utility to generate the detailed view. The *gds2def/gds2rh* utility requires the following input data:

- Configuration file. Please refer to Chapter 8 for details.
- Layer map file containing the information for mapping the GDSII layer numbers to the corresponding layers defined in the LEF/DEF.
- GDSII file for the I/O blocks.
- LEF file for the I/O blocks required for extracting pin information.

The output from *gds2def/gds2rh* is a set of *DEF*, *LEF*, and **.pratio* files for each I/O block containing placement information for the current sources and created decaps, power/ground routing geometries, cell abstraction, and each cell's relative power dissipation. The *.pratio* files specify weighting factors for the current distribution among P/G pins in memory. These files should be included in input files for RedHawk analysis.

For greater accuracy, you should provide the following additional information.

- Hierarchical SPICE netlist for the I/O block, with x,y coordinate information for the transistors:
- The size and location of the transistors in the Spice netlist control the placement and properties of the current sources inside the I/O model. For a full-chip transient dynamic analysis, it is computationally impossible to include all transistors in the simulation. Hence, *gds2def/gds2rh* groups transistors together, based on their size and location, to form virtual cells that provide a level of abstraction without much loss in accuracy.
- The inclusion of the I/O power grids and the consideration of current sources connected to the bottom layers of metal allow for complete modeling at the full-chip level and for an accurate transient dynamic analysis. Full-chip run-times or physical memory usage is usually not affected much when including all the I/Os if the number are not more than hundreds.

GDS2DEF/GDS2RH Configuration File for I/Os

This section describes configuration files needed for the I/O modeling modes described earlier. The GDS2DEF/GDS2RH utility also can extract P/G grids from bump or RDL layers to be included in RedHawk analysis.

To prepare to run GDS2DEF/GDS2RH, create a configuration file, including the keywords described in this section, as needed. The I/O configuration file keywords are listed below. For descriptions and syntax of the keywords, see [section "Creating the gds2rh/gds2def Configuration File", page E-878.](#)

Required GDS2DEF/GDS2RH Keywords

- TOP_CELL
- GDS_MAP_FILE
- VDD_NETS
- GND_NETS
- GDS_FILE

Optional GDS2DEF/GDS2RH Keywords for I/Os

- APACHE_PHYSICAL_MODEL
- ADJUST_POLYGON
- BUSBITCHARS
- CREATE_LEF_MACRO_FOR_BOX_CELLS
- DEF_FILE_DEFINITIONS
- EXTRACTION_STARTING_LAYER
- GENERATE_PLOC
- INTERNAL_DBUNIT
- LEF_FILE
- LEF_PIN_POWER
- MULTI_TASKS
- NET_NAME_CASE_SENSITIVE
- OUTPUT_DIRECTORY
- USE_LEF_PINS_FOR_TRACING

Current Profile Generation

Static Analysis

For static analysis, a power number is required to estimate the current drawn in the I/Os. RedHawk can estimate the number directly from the library models of the I/Os, or you can choose to specify the power numbers using the GSR keyword, BLOCK_POWER_FOR_SCALING.

Dynamic Analysis

Dynamic analysis in RedHawk is a high time resolution true transient analysis. Thus, for every instance, current profiles should be provided as a function of time for each mode of operation. The following describes the different ways that the current profiles as a function of time can be generated.

- **Triangular profiles using static average power values.** This form of modeling is the least accurate, but requires less effort and can be used if the other means of providing the current profiles are not available. Based on the average static power, RedHawk will generate the triangular current profiles.

- **SPICE profiles using APL.** If you prefer to use SPICE-based current profiles in your RedHawk analyses you can use APL to characterize the I/O blocks. APL automatically generates the input vectors based on the parameters defined in the input configuration file. Using these input vectors, APL characterizes the I/O blocks using NSPICE. Currently, core VDD and VSS current profiles are captured during APL characterization. If the core VSS is shared with I/O VSS, the current in core VSS is much larger than the current in core VDD of the I/O cells. RedHawk is able to take in asymmetric current profiles in VDD and VSS for dynamic analysis. With the large capacity and fast run-time capabilities of NSPICE, the characterization of I/O blocks is typically not an issue. Please see [Chapter 9, "Characterization Using Apache Power Library"](#), for more details on APL.

Results and Analysis Including I/Os

With the inclusion of I/Os, the following demonstrates some of RedHawk's capabilities for analyzing results. Figure 10-6 shows the power density map of a chip without the simultaneous switching I/O buffers, while Figure 10-7 shows a power density map with switching I/O buffers. After DvD analysis, the power density map shows the instances in which actual switching occurs. Power density is defined as the power dissipated per unit area. Power density is a good indication of potential problem regions, since a higher power density indicates more power demand is concentrated in a small region.

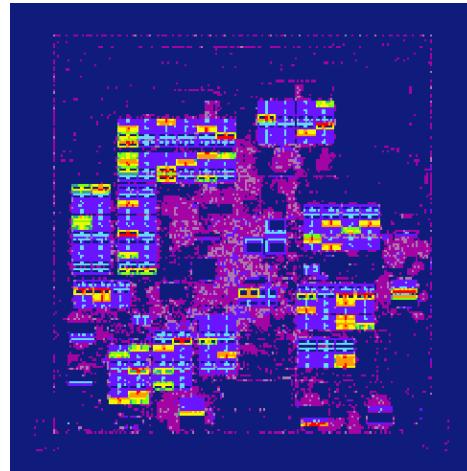


Figure 10-6 Power density map without simultaneous switching I/Os

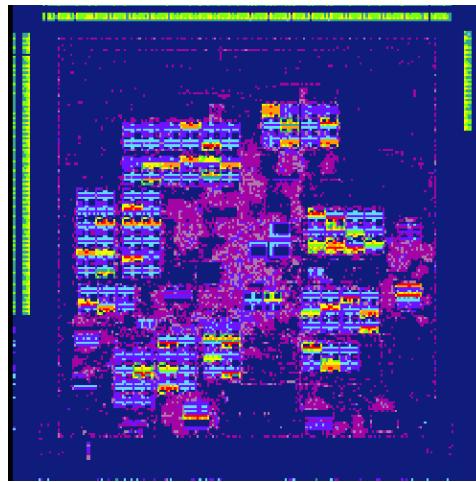


Figure 10-7 Power density map with simultaneous switching I/Os

Figure 10-8 shows the static IR drop of the VSS network. However, with dynamic voltage drop, the VSS bounce can be much higher, as seen in Figure 10-9 (without switching I/Os) and Figure 10-10 (with switching I/Os). Equally important is that the regions with high static IR drop can be completely different than the regions with high dynamic voltage drop. This demonstrates that I/O switching-induced VSS bounce can no longer be ignored, especially when VSS is shared between the I/O and the core.

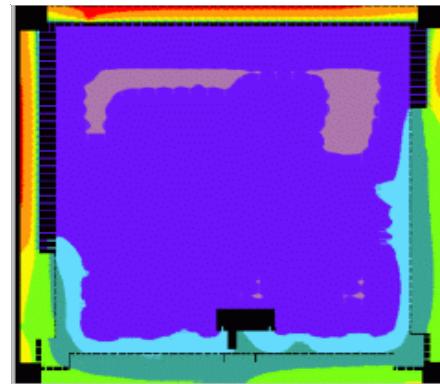


Figure 10-8 Static IR-drop with simultaneous switching I/Os; Maximum

VSS at 30mV

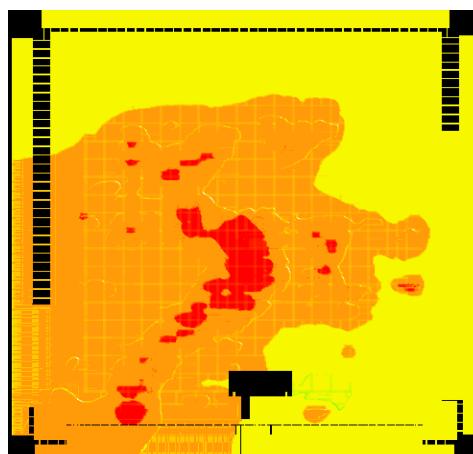


Figure 10-9 Dynamic voltage drop (DvD) without simultaneous switching I/Os; Maximum VSS bounce at 150mV

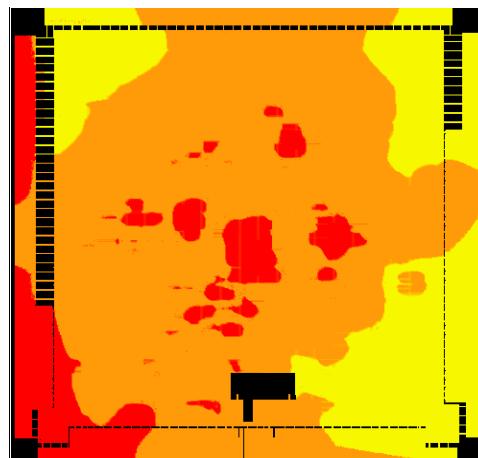


Figure 10-10 DvD with simultaneous switching I/Os; Maximum VSS bounce at 200mV

Chapter 11

Distributed Machine Processing

Introduction

RedHawk Distributed Machine Processing (DMP) offers significant memory reduction and runtime improvement over flat runs by dividing the design into multiple partitions and processing the partitions in parallel. This technology is useful especially for very large designs that initially require ultra high memory and long runtime. Although DMP will by default distribute all major RedHawk engine tasks to various parallel machines, it is possible to target runtime and memory improvements for only the simulation stage.

Every partition communicates with other partitions and creates a reduced view for the rest of the design-- thus accuracy is maintained. This technology leverages multiple computing resources to achieve performance and capacity improvements. Key features are:

- DMP is supported over LSF/SSH/SGE/RTDA grid types
- Aggressive runtime improvement and memory reduction in the typical range of 2~3X for top level runs, depending on design styles and number of machines used
- Voltage drop and EM results are the same as in the traditional flow

DMP demonstrates benefits in:

- Designs with large number of instances and very high node counts
- Designs in which RedHawk simulation performance is a bottleneck

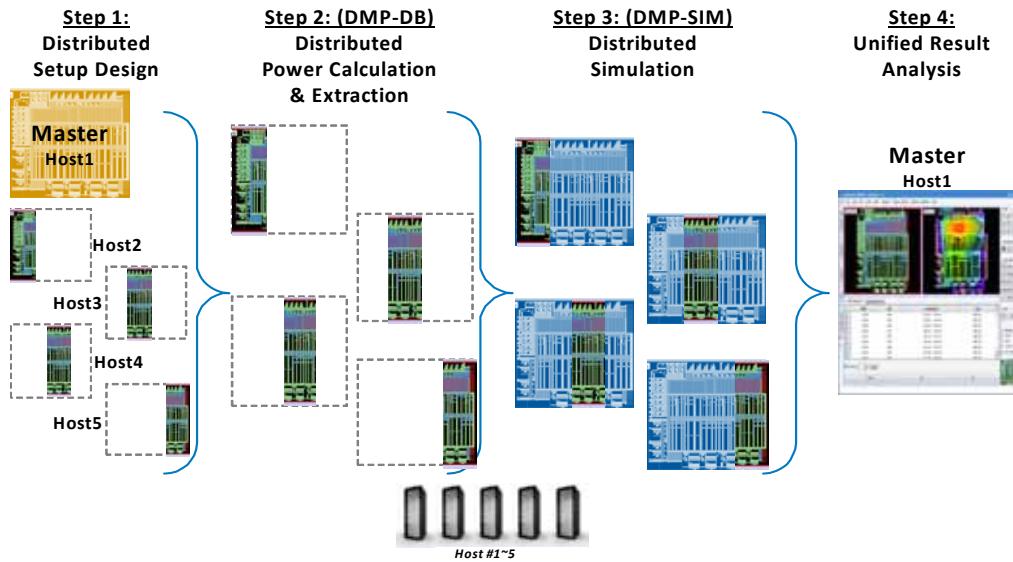


Figure 11-1 DMP methodology

DMP Flow

At launch, DMP splits the design into several partitions whose granularity can be specified by the user. The partitions are sent to different slave machines across the network for

processing, where each individual machine requires a fraction of the memory and runtime required for a single-machine full-chip run.

Prepare Data

DMP partitions the design vertically and uses distributed computing to process in parallel. Based on the number of jobs setting, DMP requests an equivalent number of compute resources to load balancer farm. After the load balancer provides the resources, DMP takes one host as a master process and the rest of the hosts as slave processes. Steps upto 'perform extraction' are part of prepare data.

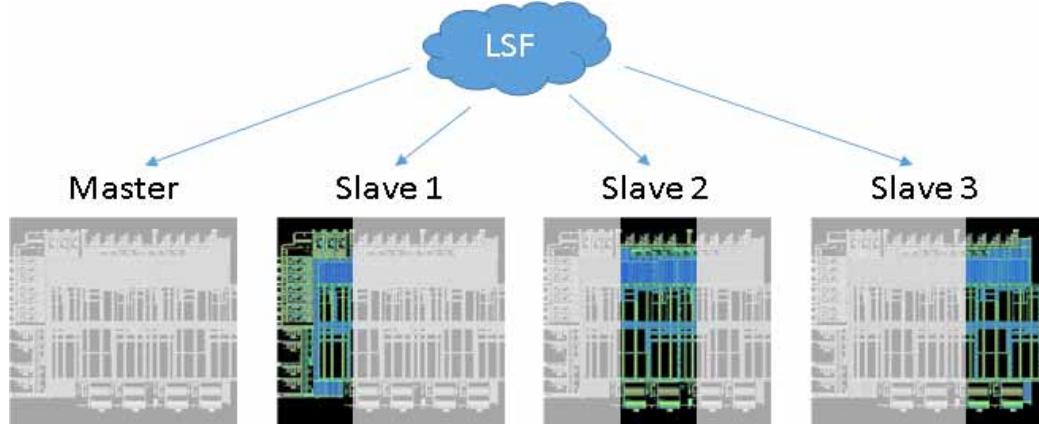


Figure 11-2 Resource allotment with a single master and 3 slave machines.

Master machine

The master process is used to monitor the slave processes. It maintains the GUI and controls the synchronization between slaves. The unified GUI on master shows results from all partitions and all GUI and Tcl commands can be parsed to the slaves only through the master. After the analysis, the results from every slave gets concatenated and will be present at the master adsRpt directory. Usually the master memory consumption is very less in comparison with slaves.

Slave machines

Each slave process handles its own region's Import DB, Setup DB, PowerStream, and Extraction. The read behavior for each input file is as follows:

LEF	All partitions read all LEFs
LIB	All partitions read all LIBs
DEF	If FAST_DEF_READ <num>: Each partition reads <num> number of DEFs at a time. Default 4. If FAST_DEF_READ 0: All partitions read all DEFs. Multi-Threaded reading
Pkg/Ploc	All partitions read complete pkg/ploc
STA	Default parallelization turned on. Each partition reads a portion of STA
SPEF	Each partition reads its own SPEF files
APL	Each partition reads all APL files

VCD File	Each partition reads its own VCD file
----------	---------------------------------------

Each partition has its own adsRpt directory in run area - adsRpt.1 for 1st partition, adsRpt.2 for 2nd partition etc which finally gets concatenated and is available in master adsRpt directory.

Reduction

Once data preparation is completed for each partition, slave process enters the reduction stage where the design is divided into smaller blocks, wherein a reduced model is created for each small block. Note that this is only for simulation. These reduced models are read in simulation.

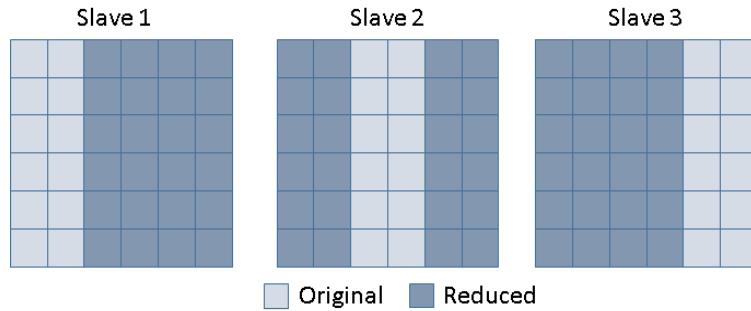
1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35
6	12	18	24	30	36

Figure 11-3 Reduction Stage

Slave 1	Slave 2	Slave 3
1	7	
2	8	
3	9	
4	10	
5	11	
6	12	
	13 19	25 31
	14 20	26 32
	15 21	27 33
	16 22	28 34
	17 23	29 35
	18 24	30 36

Simulation

Once the reduction jobs are finished, the reduced blocks are sent to each slave process, in order to form corresponding full chip representations. The full chip representation of each slave process consists of an unreduced region (from its own extraction) and reduced regions (from other slave processes). Each slave process then simulates its own full chip representation.



Supported Design and Features

- Static and Dynamic - Gridcheck/res_calc/SPT, Vectorless IR/DvD, Global Toggle Rate, Block Power For Scaling, Block Toggle Rate, State Propagation, Vectorless Scan, Fullchip, Block RTL/Gate VCD (Event and State Propagation), True Mixed Mode (Gate+RTL VCD+ Vectorless), EM (Power and Signal)
- Package - WBS, Spice model, CPA, S-Param package
- FAO - Missing Via checks
- CPM
- CTM
- SAIF Input
- MBFF support
- CMM (Raw Model)
- Pin-based MMX BPA from CMM
- Explorer

Third Party Distributed Systems

DMP supports third party distributed systems. This gives user the load balancing capability and eliminates the possibilities of exhausting single machine resources that may lead to incomplete results. User can install one of the three distributed systems (LSF / SGE / RTDA) to enable DMP access. However, if no distribution system is available, SSH password-less access is available for running DMP across computer networks.

IBM Platform LSF

Requires LSF version 7.0 or above. In addition, the following scripts/binaries needs to be installed in LSF and requires execution permission.

```
mpirun.lsf
openmpi_wrapper
Pam
```

Sample config file:

```
NUMBER_OF_JOBS 16
GRID_TYPE LSF
QUEUE_NAME dmp_queue
SEQUENTIAL_LAUNCH True
```

```
ARGUMENTS_FOR_LARGE_JOBS "-q dmp_queue -  

R"rusage[mem=130000]" "
```

- User can specify ‘ARGUMENTS_FOR_LARGE_JOBS’ to provide queue specific settings like memory reservation, queue names, job names or passwords etc.
- User can specify the reservation size, queue names, job names or passwords for Master using ARGUMENTS_FOR_MASTER_JOB config keyword.
- By default, unique machines are chosen for each slave. User can control this by using `-R "span[ptile=<num>]"` option in ARGUMENTS_FOR_LARGE_JOBS. This defines number of partitions to be launched per machine (default =1 in DMP)
- LSF output and errors are stored in `.dmp/lsf.out` and `.dmp/lsf.err`

SEQUENTIAL_LAUNCH in LSF

SEQUENTIAL_LAUNCH in config file helps to control job reservations and memory limits on LSF.

When **SEQUENTIAL_LAUNCH 0/False**

- Output of ‘bjobs’ (NUMBER_OF_JOBS 4)

```
76603  indqa  RUN  normal    sjoindqa384 sjoindqa384 *aunch_lsf Feb 15 22:20
        sjoindqa384-3
        sjoindqa512-1
        sjoindqa384-1
```

- Single job launched over LSF, reserving unique machines for 4 processes
- Depending on LSF setup, job memory reservation might be considered cumulative of all 4 processes
- ‘ptile’ has an impact on reservation, since all jobs are linked

When **SEQUENTIAL_LAUNCH 1/True** (default):

- Output of ‘bjobs’ (NUMBER_OF_JOBS 4)

```
76604  indqa  RUN  normal    sjoindqa384 sjoindqa384 *62000.4.3 Feb 15 22:20
76606  indqa  RUN  normal    sjoindqa384 sjoindqa384 *62000.4.1 Feb 15 22:20
76607  indqa  RUN  normal    sjoindqa384 sjoindqa384 *62000.4.0 Feb 15 22:20
```

- 4 unique jobs launched, one per process. Memory reservation is per job
- ‘ptile’ has no effect since all jobs are unique. Jobs will be launched purely based on LSF determination of free memory. Multiple jobs may land on same machine if enough memory is available

Oracle Grid Engine (previously known as Sun Grid Engine)

Requires Sun Grid Engine 6.2 update 5 or above. In addition, a Parallel Environment (PE) must be defined for submitting parallel jobs in RedHawk. This setup requires the access of Grid Engine Administrator. To set up a PE named ‘rh_pe’, follow below example:

```
% qconf -ap rh_pe
pe_name          rh_pe
slots            99999
user_lists       NONE
xuser_lists      NONE
start_proc_args  NONE
```

```

stop_proc_args      NONE
allocation_rule     $round_robin
control_slaves      TRUE
job_is_first_task   FALSE
urgency_slots       min
accounting_summary  FALSE

```

Then, the PE configuration ‘rh_pe’ must be added to the queue’s ‘pe_list’.

```

% qconf -mq all.q
...
pe_list      make rh_pe
...

```

User can check the PE by using below command.

```
% qconf -sp rh_pe
```

Sample dmp.cfg file for SGE:

```

GRID_TYPE SGE <PE_name>
NUMBER_OF_JOBS 4
QUEUE_NAME large
ARGUMENTS_FOR_LARGE_JOBS -l mfree=100G

```

- PE_name is the name of the parallel environment setup. If SEQUENTIAL_LAUNCH true (default from 18.0) is used, user need not explicitly specify the PE name. If SEQUENTIAL_LAUNCH is false, PE_name is mandatory.
- Machines will be picked up from the SGE queue based on available memory
- ‘-l mfree=xG’ is used to reserve xGB memory for a particular job.

RTDA NetworkComputer

Requires RTDA NC 2013.03 or above with RedHawk patch. Please check if ‘nc_redhawk’ is available under <nc_installation>/common/eda/Ansys/redhawk/nc_redhawk. In addition, RedHawk environment must be added by the IT. Please check if <nc_installation>/common/local/environments/REDHAWK.start.csh exists.

Useful Distribution System Commands

IBM Platform LSF

bsub	Submit a job
bkill	Kill a job
bjobs	Check running jobs
bhosts	Check hosts availability

bqueues	Check queue information
---------	-------------------------

Oracle Grid Engine (previously known as Sun Grid Engine)

qsub	Submit a job
qdel	Kill a job
qstat	Check running jobs
qhost	Check hosts availability
qconf	Add/Modify/Check configurations
qmon	GUI interface

RTDA NetworkComputer

nc run	Submit a job
nc stop	Kill a job
nc forget	Forget a job
nc list	Check running jobs
nc gui	GUI interface for checking current jobs
nc mon	GUI interface for viewing system setup

DMP Custom Launcher

Some grids, like UGE, are uniquely customized at individual accounts. Standard SGE launcher will not work in these cases. DMP has the ability to read in custom launchers built around the customer grid in such cases.

Usage in DMP Config file:

```
NUMBER_OF_JOBS <num>
GRID_TYPE CUSTOM
CUSTOM_SCRIPT <path_to_script>
<other arguments like ARGUMENTS_FOR_LARGE_JOBS> # Only
valid config file arguments
```

All MPI launch commands must be inside the script. ‘redhawk.dmp’ just sources this script and expects ‘redhawk.exe’ to be launched. Please use ‘\$APACHEROOT/bin/redhawk.dmp’ for reference.

SSH Password-Less Access

DMP is able to run without leveraging the third party distribution system. This method is good for testing in a secure, private network environment. However, load balancing is in the hands of the user. Therefore, a very careful and strategic setting of the computing resources should be considered to provide the best performance.

To run DMP in this flow, user must obtain SSH password-less access to the specified machines. Please follow below steps:

```
## To setup passwordless SSH
1.mkdir ~/.ssh
2.cd ~/.ssh
3.ssh-keygen -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (your_local_home/.ssh/id_rsa):(press enter)
Enter passphrase (empty for no passphrase):(press enter)
Enter same passphrase again:(press enter)
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
18:6a:e3:78:ab:2d:0c:8e:f9:67:f7:30:32:44:77:34 user@server1

4.scp ~/.ssh/id_rsa.pub user@server2_remote_server
example:scp ~/.ssh/id_rsa.pub mafiz@sjo2ae128-2
5.ssh user@server2_remote_server
example:ssh sjo2ae128-2y
Type your password
6.chmod 700 .ssh
7.cat ./ssh/id_rsa.pub >> .ssh/authorized_keys
8 chmod go-w $HOME/.ssh
9. chmod 600 $HOME/.ssh/authorized_keys
10. chown 'whoami' $HOME/.ssh/authorized_keys
```

Running DMP

RedHawk DMP provides a seamless interface to launch RedHawk with the distributed system. Settings are read in from the configuration file and the keywords are described in the following table.

```
redhawk -lmwait -dmp <dmp_config_file> run.tcl
```

Where:

<dmp_config_file>: DMP configuration file

DMP configuration file provides informations like Number of partitions, Grid type, launch constraints etc.

Sample DMP Config file:

```
NUMBER_OF_JOBS 16
GRID_TYPE LSF/SSH/RTDA/SGE
QUEUE_NAME dmp_queue
```

```
ARGUMENTS_FOR_LARGE_JOBS " -q dmp_queue -R
"rusage[mem=130000]""
ARGUMENTS_FOR_SMALL_JOBS " -q dmp_queue -R
"rusage[mem=130]""
ARGUMENTS_FOR_MASTER_JOB " -q dmp_queue -R
"rusage[mem=1300]""
```

DMP Configuration File Keywords

NUMBER_OF_JOBS

Specifies total number of jobs required to run the design. It includes the sum of total number of partitions and a single Master. *Default: 4*

Syntax:

```
NUMBER_OF_JOBS <number>
```

Example:

For 8 partitions, number must be 8 (partitions) + 1(master) = 9

```
NUMBER_OF_JOBS 9
```

GRID_TYPE

Specifies the distributed system used for run. For SGE, user must specify the parallel environment name when SEQUENTIAL_LAUNCH is false. *Optional.*
Default: LSF

Syntax:

```
GRID_TYPE [LSF|RTDA|SGE] <parallel environment>
```

QUEUE_NAME

Defines the queue name used for run. This can also be defined within ARGUMENTS_FOR_SMALL/LARGE_JOBS. *Optional. Default: None*

Syntax:

```
QUEUE_NAME <queue>
```

CANDIDATE_HOST_LIST

Specifies the list of machines to launch the jobs. Works with LSF/SSH grid Types.
 Required for SSH. If number of machines in list is smaller than NUMBER_OF_JOBS, then machines in list will be reused for all additional partitions.

Syntax:

```
NUMBER_OF_JOBS 4
GRID_TYPE SSH
CANDIDATE_HOST_LIST
{
    machine1 # will be considered master machine
    machine2
    machine2 # user can specify same machine multiple times to
    launch more partitions on it
    machine3
```

..
}

Sample dmp.cfg file for SSH running 2 parallel jobs on slave_1:

```
GRID_TYPE SSH
NUMBER_OF_JOBS 4
CANDIDATE_HOST_LIST {
    master
    slave_1
    slave_1
    slave_2
}
```

ARGUMENTS_FOR_LARGE_JOBS

Specify additional options for the distributed system. *Optional. Default: None*

Syntax:

```
ARGUMENTS_FOR_LARGE_JOBS <cmd_opt_1> [... <cmd_opt_n>]
```

Sample dmp.cfg file for LSF choosing machine that has more than 100g free memory:

```
NUMBER_OF_JOBS 4
ARGUMENTS_FOR_LARGE_JOBS -R rusage[mem=100000]
```

Sample dmp.cfg file for SGE choosing machine that has more than 100g free memory:

```
GRID_TYPE SGE rh_pe
NUMBER_OF_JOBS 4
QUEUE_NAME large
ARGUMENTS_FOR_LARGE_JOBS -l mem_free=100G
```

NOTE: mem_free just confirms that there is 100G of free memory available, but does not reserve it. In order to reserve a specified amount of memory for a job, use the option mfree.

ARGUMENTS_FOR_SMALL_JOBS

Specifies the option for dispatching reduction jobs to another queue or resources. Ensure that the queue specified is available otherwise it will slow down the DMP process if it is queued up for finding available machines. When not specified, it will launch on existing machines based on "ARGUMENTS_FOR_LARGE_JOBS" keyword. *Optional. Default: None.*

Syntax:

```
ARGUMENTS_FOR_SMALL_JOBS <cmd_opt_1> [... <cmd_opt_n>]
```

Sample dmp.cfg file for submitting reduction jobs to different queue and requests smaller machines:

```
NUMBER_OF_JOBS 4
ARGUMENTS_FOR_LARGE_JOBS -q large -l mem_free=100g
ARGUMENTS_FOR_SMALL_JOBS -q small -l mem_free=10g
```

ARGUMENTS_FOR_MASTER_JOB

Defines a different queue to launch the master job as it requires less memory.
Mainly used for LSF/SGE/RTDA grids. *Optional. Default: None*

Syntax:

```
ARGUMENTS_FOR_MASTER_JOB <arguments_like -q smaller_queue  
-R ...>
```

or

```
ARGUMENTS_FOR_MASTER_JOB  
{  
<arguments>  
}
```

Sample dmp.cfg file:

```
GRID_TYPE LSF  
NUMBER_OF_JOBS 5  
SEQUENTIAL_LAUNCH TRUE  
ARGUMENTS_FOR_LARGE_JOBS -q large -R "rusage[mem=100000]"  
ARGUMENTS_FOR_MASTER_JOB -q medium -R "rusage[mem=25000]"
```

ARGUMENTS_FOR_RH_JOBS

Specifies different reservation criteria for each partition. The keyword overrides ARGUMENTS_FOR_LARGE_JOBS and ARGUMENTS_FOR_MASTER_JOB specifications. Partition number is specified using “-rh_dmp_job_id” argument, where 0 is for Master, 1 for Slave1, 2 for Slave2 etc. Multiple partitions can be assigned the same reservation. First line can be default argument. Any partitions not listed will be assigned this value. Default: None

Syntax:

```
ARGUMENTS_FOR_RH_JOBS {  
## Default settings for all unlisted partitions  
-q rh1 -R "rusage[mem=2000]"  
# For master  
-n 16 -q rh -R "rusage[mem=1000]" -rh_dmp_job_id 0  
# For Slave 1  
-n 16 -q rh -R "rusage[mem=2000]" -rh_dmp_job_id 1  
#For Slaves 2 and 3  
-q rh -R "rusage[mem=3000]" -rh_dmp_job_id 2,3  
}
```

Example:

```
ARGUMENTS_FOR_RH_JOBS {  
-R "rusage[mem=50000]" -rh_dmp_job_id 1,3  
-R "rusage[mem=10000]" -rh_dmp_job_id 0,2  
}
```

WORKING_DIR_LIST

Specifies a file which contains the local working directory (to be used to store the simulation files) for specific machine. This can be used either to improve speed or

when local disk space is low in the run area. Wildcard(*) specified in the list is the default working directory. *Optional. Default: None*

Syntax:

```
WORKING_DIR_LIST <file_name>
```

Example:

```
WORKING_DIR_LIST <machinelist>
```

Syntax of machinelist:

```
<machinename_of_partition1> <path_to_directory>
<machinename_of_partition2> <path_to_directory>
<machinename_of_partition3> <path_to_directory>
....
```

Example machinelist file:

```
* /tmp
```

Here every machine will store the simulation data in its /tmp directory. This improves the runtime. Ensure that /tmp has enough space available.

WORKING_DIR_NAME

Specify the directory name for WORKING_DIR_LIST. *Optional. Default: <user_name>.nwopt**

Syntax:

```
WORKING_DIR_NAME <name>
```

Sample dmp.cfg file that uses local working directory:

```
NUMBER_OF_JOBS 4
WORKING_DIR_LIST ./dir.list
WORKING_DIR_NAME .demo_dmp_case

dir.list:
host1 /data/user1
host2 /data/user1
* /tmp
ARGUMENTS_FOR_LARGE_JOBS -q large -l mem_free=100g
ARGUMENTS_FOR_SMALL_JOBS -q small -l mem_free=10g
```

GLOBAL_DIR_LIST

Specifies a file which contains a list of directories to which all .apache files for each partition has to be saved. This can be used if the space in current run area is insufficient. *Optional. Default: None*

Syntax:

```
GLOBAL_DIR_LIST <file_name>
```

Example:

```
GLOBAL_DIR_LIST <filelist>
```

Syntax of filelist:

```
<dir1>
<dir2>
<dir3>
```

GLOBAL_DIR_NAME

Specifies the directory name for GLOBAL_DIR_LIST. *Optional. Default:* <user_name>.nwopt

Syntax:

GLOBAL_DIR_NAME <name>

CONFIG_LOCAL_DIR_AS_RH_MM_CACHE

When set to 1, ‘.MM’ directories of the slaves are created inside the directory, working_dir/<username>_nwopt*. *Optional. Default: ON when WORKING_DIR_LIST is used.*

Syntax:

CONFIG_LOCAL_DIR_AS_RH_MM_CACHE [1 | 0]

SEQUENTIAL_LAUNCH

When set to TRUE, it picks up unique machine for each slave and helps to get machines quicker. *Default TRUE for LSF and SGE grids.*

Syntax:

SEQUENTIAL_LAUNCH [TRUE | FALSE]

HEARTBEAT_CONFIG

Invokes a different DMP job monitoring and killing algorithm. Default algorithm uses MPI and existing Grid (LSF/SGE/RTDA) to monitor jobs. If existing grid is too loaded or unstable, user can use this algorithm. It will kill all jobs if one slave does not respond within specified time. *Optional. Default: 60 sec.*

Syntax:

HEARTBEAT_CONFIG <time_in_seconds>

Example:

HEARTBEAT_CONFIG <360>

DMP GSR Keywords

See [section "DMP Keywords", page C-709](#)

Report Generation

To generate power reports (both power_summary.rpt and <design>.power.rpt) use the following Tcl command:

```
report power -all ?-dmp_merge?
```

Where:

-dmp_merge is an optional switch to generate adsPower/*power

- When ‘REPORT REDUCTION max’ is used in GSR, instance wise DvD reports are not generated by default. To generate these, use the following Tcl command

```
report dvd -instance ?-o <output_file>? ?-limit <number>?
```
- The files *.inst, *.inst.arc, *.isnt.pin, layer_drop.rpt, *power.unused, ir.out, via.ir can be generated in adsRpt/ or .apache/ on demand using the following Tcl commands:

```
dmp_collect_result -static ?-inst ?-instPin ?-instArc ?-
layerDrop ?-unusedPower ?-irOut ?-viaIR ?-all
dmp_collect_result -dynamic ?-layerDrop ?-unusedPower ?-
irOut ?-viaIR ?-all
```

Log files

A complete high level DMP log file is created as adsRpt/nwopt_apache.log. This log file captures runtime statistics and error messages (if DMP run fails) for DMP session, including reduction, simulation, and post-simulation data collection.

DMP run creates the following output directories based on the runs carried out:

```
nwopt_apache.static - Created for static analysis
nwopt_apache.dynamic - Created for dynamic analysis
```

These directories include nwopt.log file which contains runtime and memory details for reduction and simulation runs.

Consolidated redhawk.log

Master log file concatenates redhawk.log from all slaves, section by section. The combined redhawk.log file can be reduced to its original master log file by striping away all slaves contents using "dafconvert" utility under \$APACHEROOT/bin.

Usage:

```
dafconvert d (or dafconvert d -i <combined.log> -o
<reduced.log>)
```

Important Files

.run.<lsf sgc rtda>	DMP launch commands
.apache*/dmp.partition	DMP partition positions
./adsRpt/nwopt_apache*.log	DMP Simulation log file
.apache/.debug.<lsf sgc rtda>	Scheduler output
./nwopt_apache*/apache.nwoptRunInfo	Overall running status
./nwopt_apache*/apache.nwoptTmpFileInfo_*	Tracking remote machine disk space
./nwopt_apache*/apache.sim_cmd	Command to re-launch DMP simulation only
./nwopt_apache*/p_*/*done	Reduction status indication
./nwopt_apache*/sp_*/*done	Simulation status indication
./nwopt_apache*/sp_*/.log	Simulation log file
./nwopt_apache*/sp_*/.debug*	Simulation debug info

Export/Import DB

'Export db' can be invoked after any stage in DMP similar to a flat run, to save the database.

Importing a DB in DMP requires same number of machines as when the DB was exported. The DB directory has a saved DMP config file, user can use the same in import db runs. Importing db via GUI (Invoke empty GUI using redhawk -dmp <config file>) as well as via Tcl file are supported.

Single license for Import DB (-view_only mode)

This feature enables to import the DB saved from a DMP run by checking out only single license irrespective of the number of partitions. To enable this, set "-view_only" argument while launching RedHawk.

Syntax

```
redhawk -view_only -dmp <config> <args>
```

The following functions can be performed in the -view_only mode:

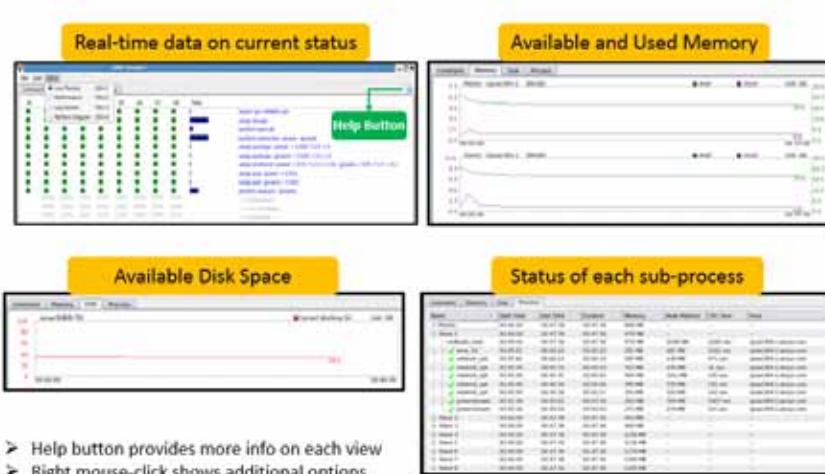
- Import DB
- View Results
- View Heatmaps and plots
- Use TCL commands to query the design
- Launch existing Explorer DB (Cannot launch new Explorer session in this mode)

DMPstat- DMP Monitoring Utility

dmpstat utility performs Live Monitoring, Tracking and Debugging of the DMP runs. Issues may occur during a DMP run due to different factors like the machine, network, setup etc. which demands continuous monitoring of the runs.

dmpstat gathers information into following segments -

1. **Live Tracker** contains per partition information on command being performed, memory and disk space consumption, CPU, Duration, Memory etc information on each sub-process.
2. **Performance** contains stage-wise breakdown of the runtimes (Wall and CPU) along with Memory usage, node count, instance count and other useful information per partition.
3. **Log Viewer** displays per partition logs. It can also create a combined consolidated log.
4. **Partition Diagram** contains snapshot of the design layout with the partition boundaries highlighted.



Launching dmpstat:

dmpstat can be invoked in multiple ways:

- directly when launching RedHawk

```
redhawk -dmp <config> -f <tcl_file> -dmpstat
```

- or can be launched anytime during or after the run from the runarea:

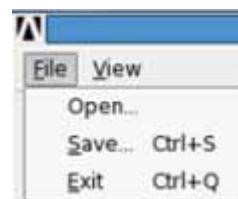
```
<path_to_redhawk_installation>/bin/dmpstat
```

- Can also be launched anytime during or after the run from any area:

```
<path_to_redhawk_installation>/bin/dmpstat  

<path_to_runarea>
```

It reads all information from .dmp and .apache* directories in the run area. Also user can export DB of dmpstat and save for later reference.



By default, all runtime, memory, disk space and other data from the run is actively tracked and updated every 60 secs.

Monitoring and Debugging Using dmpstat

Live tracker – Memory, Disk space & Process tracking for each partition

The Live tracker functions are summarized in the following sections:

Command window

Command window lists out the commands being executed in real-time. It can be used to check the current stage of run, when run is improgressive. The 'Help window' provides additional details

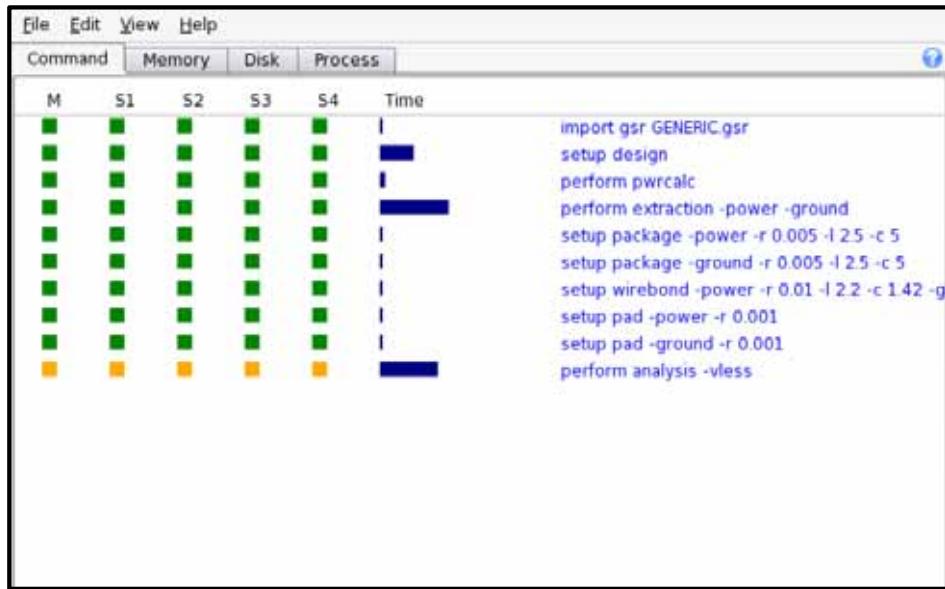


Figure 11-4 Command window

Right mouse-click displays following additional options:



Memory window

The main functions include Real-time tracking of the Used and Available memory on each partition. It is also possible to place cursor, Zoom-in/out, unify scales of Available and Used Memory etc. It also includes Cache memory tracking in the Available memory graph. The tab indicates whether a partition failure was due to out-of-memory issue.



Figure 11-5 Memory window

Following options can be enabled using Right mouse click.

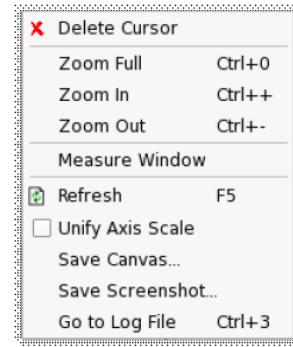


Figure 11-6 Additional options window for Memory tab

Disk Window

Primarily used for real-time tracking of available disk space on all drives used by each partition. It also includes CACHE_DIR and the WORKING_DIR_LIST. This can be used to monitor the available disk on all drives during the run to confirm any erroneous behavior due to disk space issues.

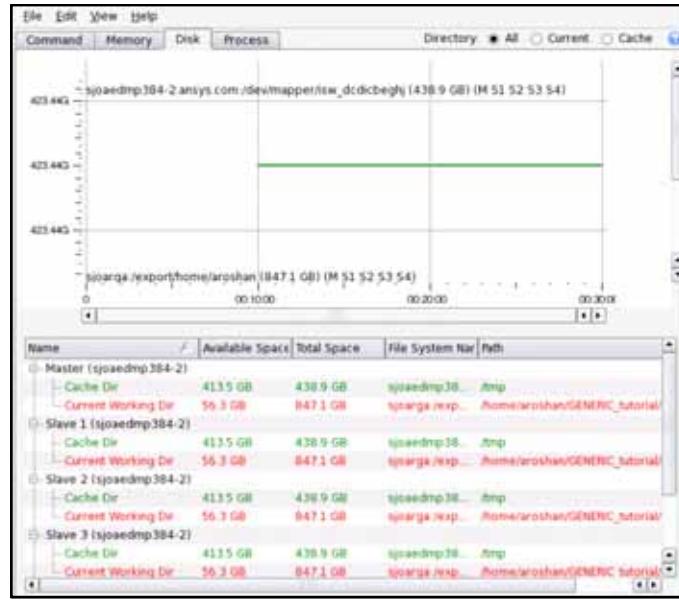


Figure 11-7 Disk window

Process window

Enlists all the child processes launched by master and each slave. It displays Start/End time, Duration, Peak Memory etc of each sub-process. It can be used to identify the processes that are still running or failed to launch, to verify the memory consumed per sub-process, to debug memory related issues.

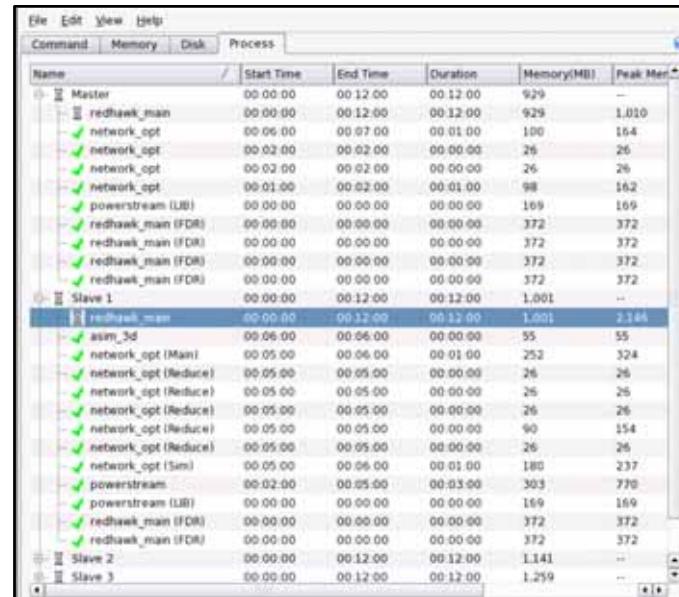
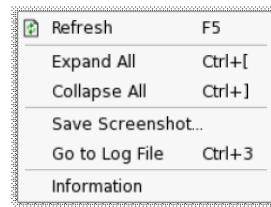


Figure 11-8 Process window

Additional options window can be enabled using right mouse click. Information option in this window gives the PID and command used launch the process.



Performance window

Reports stage wise breakdown of different parameters like Wall time, Memory, Node count etc per partition. Both Graphical and Table views are available for this tab and user can switch between both as required.

- Wall Time:** Reports stage-wise breakdown of runtime per partition.

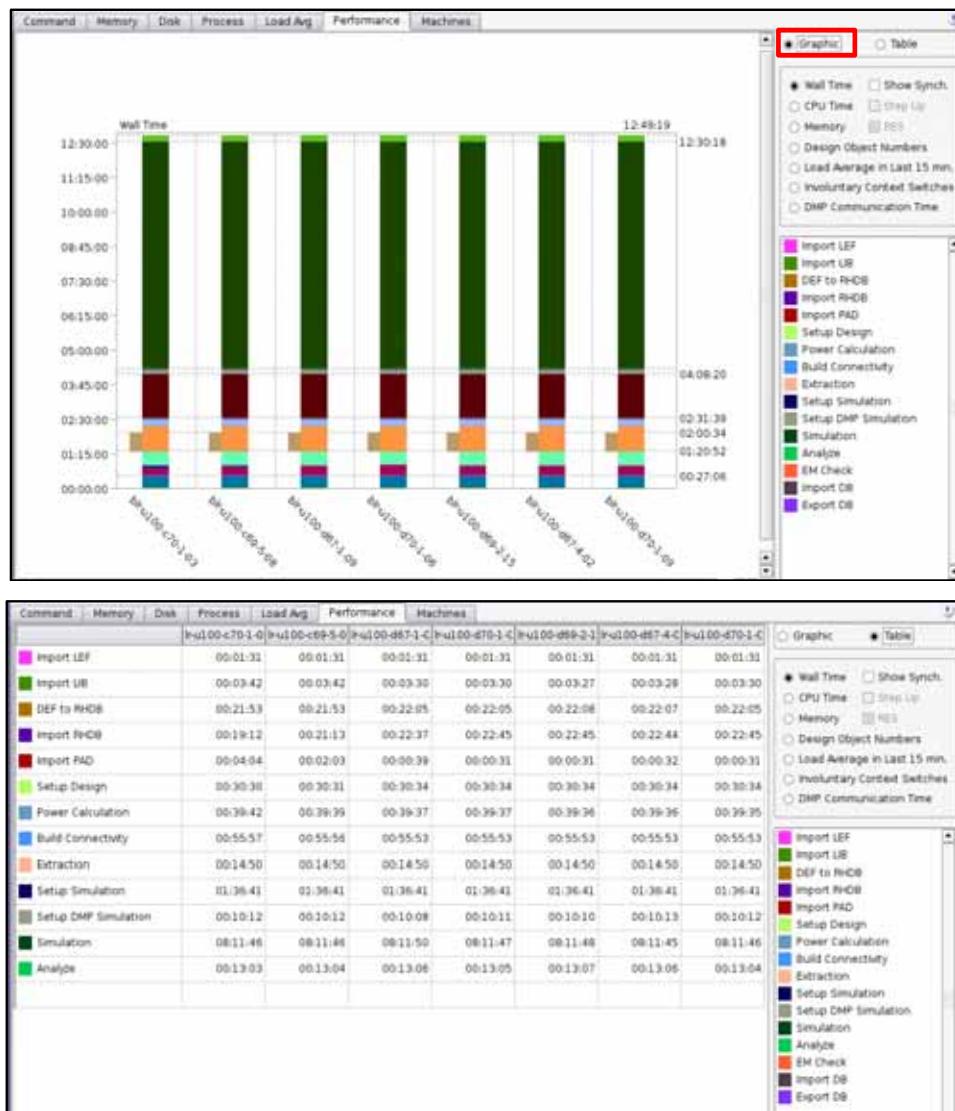


Figure 11-9 Graphical and Tabular view of Wall time

- Memory:** Reports stage-wise breakdown of memory usage for each partition. Select

the RES option (Resident Memory) to view the actual RAM usage

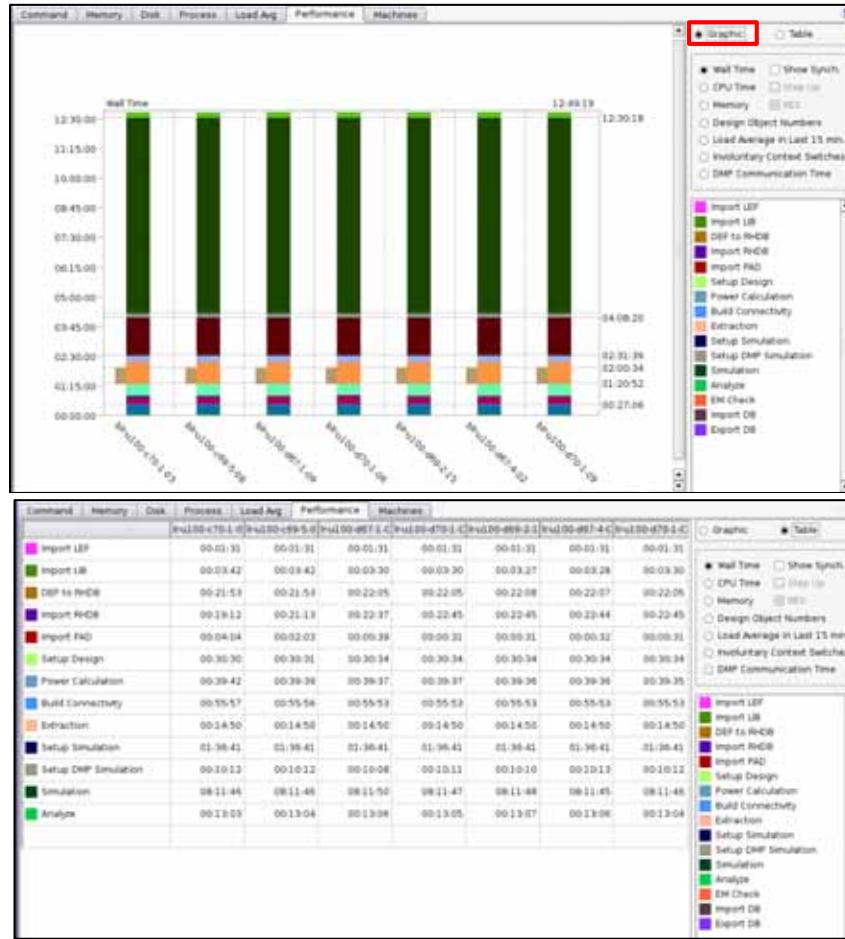


Figure 11-10 Graphical and Tabular View for Memory usage

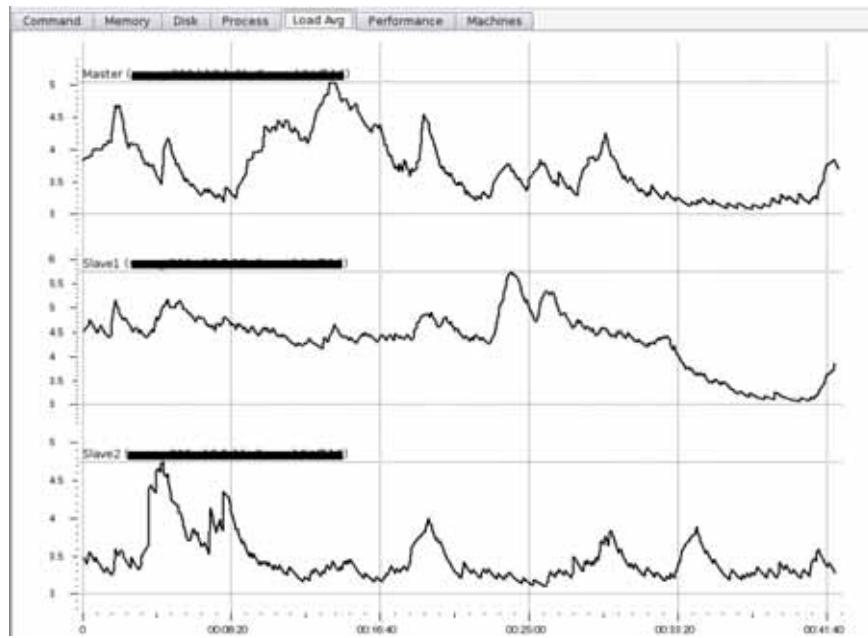
- c. **Design Object Numbers:** Reports stage-wise breakdown of node count, instance count etc per partition. An option to add up the node count for individual partitions is available.

Command	Memory	Disk	Process	Load Avg	Performance	Machines	Graphic	Table
Instance	7058885	8038613	12553339	9621724	6667475	8536753	9042413	
Net	3513948	3615508	4840176	3681087	4040484	3334425	1982338	
Wire	7520252	7477252	8799489	7904070	9843887	10130554	9520909	
SHA	30766073	28818077	37087244	307676070	40872098	39564420	93471557	
DIModule	74097128	69433188	81412205	749436000	95327787	85245724	70176004	
Dimension	106015786	98620089	125877348	104138058	134320555	121102953	100854882	
Addressode	73945268	85480304	104275459	95809731	111595724	101811025	84343517	
Addressessor	116950348	132644253	159218400	147891490	169830400	154976189	132279127	

Figure 11-11 Tabular view of design Object Numbers

Load Avg

Graphically represents the average load on each partition with respect to time



Machines

Displays important machine properties such as OS version, number of threads, CPU type etc. Machine/ OS related issues can be debugged using this window as reference.

	Name	Partition	Linux OS	Kernel Version	Cores	HT	Memory	CPU Specification
1	blr-u100-c58-5-15	Master	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
2	blr-u100-d70-1-10	Slave 1	SLES 11		16	16	516GB	Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz
3	f16-4-blr-02	Slave 2	SLES 11	3.0.101-0	16	16	258GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
4	blr-u100-c61-3-09	Slave 3	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
5	blr-u100-c58-4-02	Slave 4	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
6	blr-u100-c62-4-12	Slave 5	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
7	blr-u100-c59-1-04	Slave 6	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
8	d1-1-blr-06	Slave 7	SLES 11	3.0.101-0	16	16	258GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
9	d2-3-blr-01	Slave 8	SLES 11	3.0.101-0	16	16	258GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
10	blr-u100-c58-5-14	Slave 9	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
11	d2-3-blr-10	Slave 10	SLES 11	3.0.101-0	16	16	258GB	Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz
12	blr-u100-c53-3-07	Slave 11	SLES 11		16	16	257GB	Intel(R) Xeon(R) CPU E5-2673 v2 @ 3.30GHz

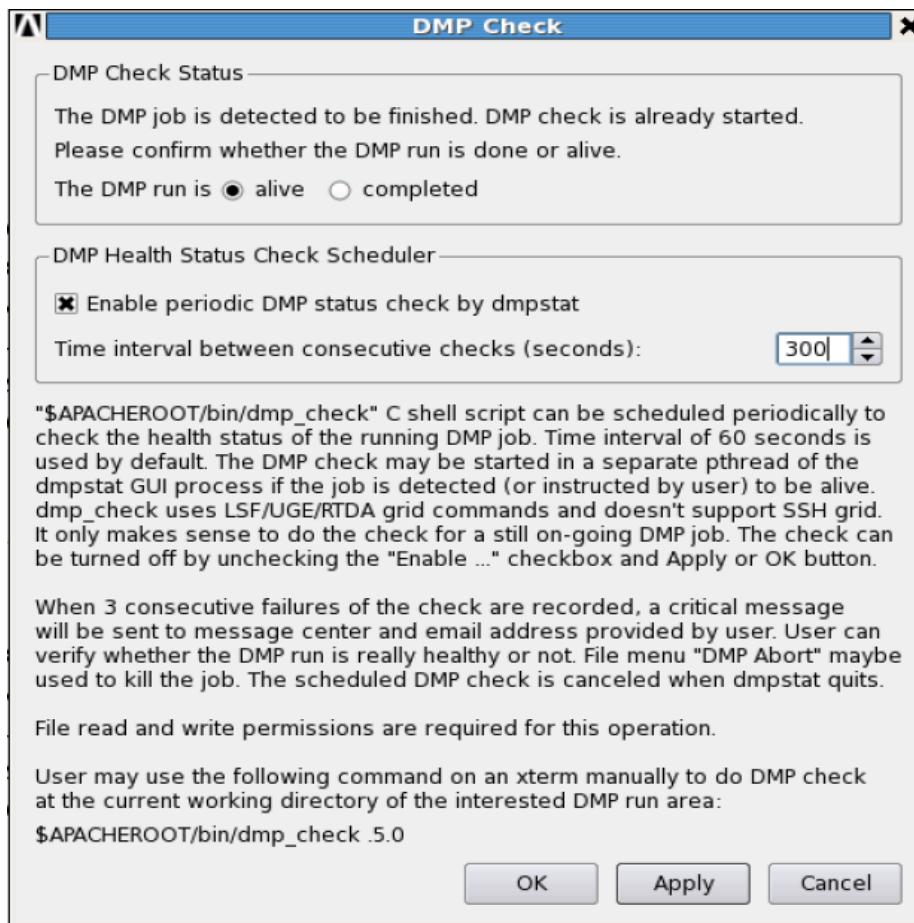
Figure 11-12 Machines window

NOTE: Note that the "Help" button provides more info on each view. Right mouse-click shows the additional options available for each "View" options.

File Menu

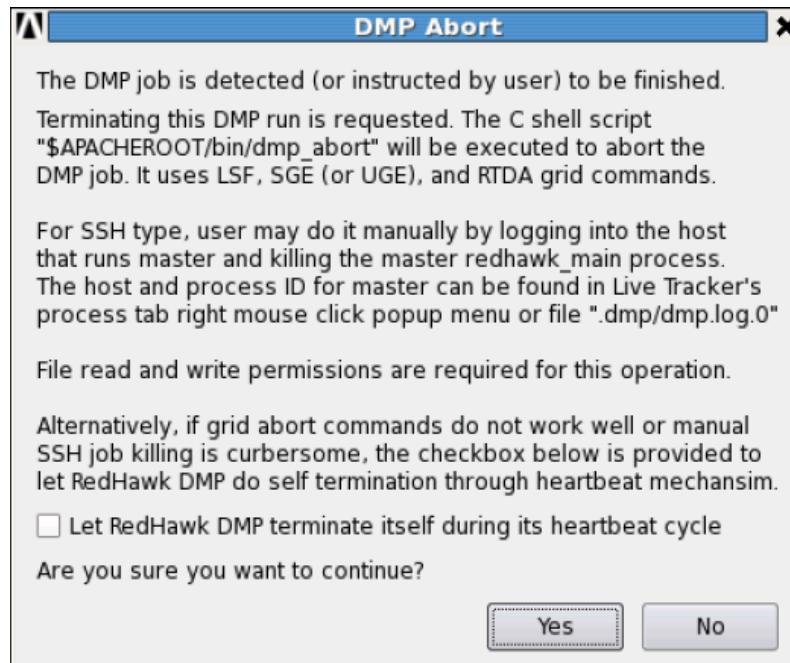
DMP Check

Using DMP Check option, you can turn ON/OFF job status monitoring. *dmpstat* automatically tries to detect if job is alive or complete. This is still new, so may fail in some cases. User can manually set the status as well. If job status is set to alive it can enable auto-detection of hung or unresponsive jobs.



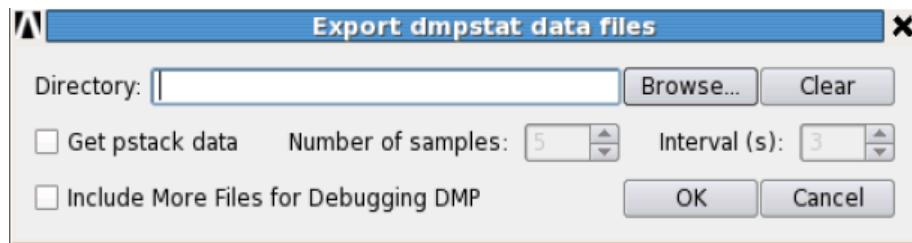
DMP Abort

Used to kill a DMP job. This option sends the grid specific kill command to the job. For unresponsive grids or SSH, user can select check-box "Let RedHawk DMP terminate itself during its heartbeat cycle". This will force kill the jobs the next time a partition check-in its status (determined by interval in DMP Check).



Save

The option saves *dmpstat* session into a directory for later reference. It bundles a lot of files like redhawk.log, .nx.log, .statistic etc which can be transferred to RD for debug. The *dmp_files* script is not required in this case. User can collect pstack output of redhawk_main processes from each slave.



Heartbeat Monitoring

Using this feature the tool queries the state of all processes at every 60th second and if any process dies, then others would exit automatically.

To enable this:

1. Create a new config file (say heartbeat.cfg) with below contents:

```
timeout <value_in_seconds> Ex: 60
```

2. Launch RedHawk

```
redhawk -heartbeat <heartbeat_config_file> -dmp
<conf_file>
```

where:

-heartbeat: controls worker ping and auto-kill in case any one of the workers does not respond. This is OFF by default. If the option "-heartbeat <heart_beat_config>" is mentioned and no timeout value is defined in the config file, then it is 60 seconds by default.

Chapter 12

Package and Board Analysis

Introduction

High quality models of off-chip RLC circuit elements such as the package and board can be very significant in achieving an accurate simulation of circuit power. This chapter describes the procedures for creating the required package and board-related circuit models and mapping package port names to die pad names.

RedHawk can perform chip-package thermal co-analysis using the integrated Chip Package Analysis (CPA) GUI. Thermal analysis of the package with Chip Thermal Modeling (CTM) is natively supported in **RedHawk**. You can perform CTM generation, and CTM+package thermal analysis in the same **RedHawk** session. See [Chapter 20, "Chip-Package Analysis \(CPA\)"](#).

Package and Board Models

Three types of package models are available, a simple model in which all power pads have the same RLC values and all ground pads have the same RLC values. More complicated package modeling, in which different values can be assigned to each pad, are provided by a Spice subcircuit model or an S-parameter model. These three types of models are described in the following sections.

Simple Package RLC Model

A simple package circuit model representation in **RedHawk** is shown in Figure 12-1. Package parasitics represent RLC values associated with the package substrate itself. The wirebond represents the bondwire (lead frame to bondpad connection) parasitics. For Flip Chip designs, where there is no lead frame, this can be included as a part of package parasitics. Padwire parasitics represent the parasitics of the pad (or bump) wire routing (bondpad/bump to I/O pad cell connection). If this routing is already included in the DEF, **RedHawk** extracts the information automatically; you do not need to specify these values.

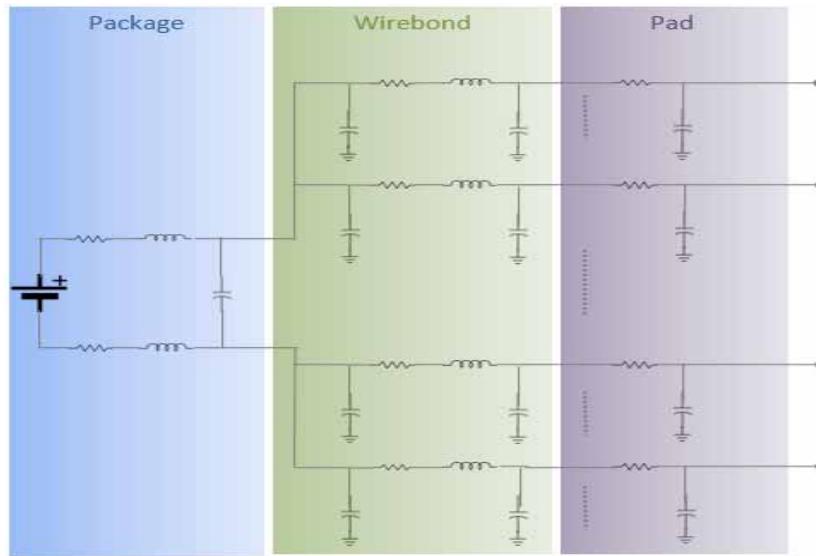


Figure 12-1 simple package RLC model

The simple model annotates single RLC values for all pads in the design using TCL commands. This can be done separately for power pads and ground pads. All power pads and all ground pads have the same parasitic values with this command.

The commands used for annotating simple RLC package values are:

```
setup package -r <in Ohm> -c <in pF> -l <in pH>
setup wirebond -power/-ground -r <in Ohm> -c <in pF> -l <in pH>
setup pad -power/-ground -r <in Ohm> -c <in pF>
```

Note that asymmetric power and ground package models are supported, and wirebond and pad constraints can be annotated separately for power and ground. Also, an inductance value is not annotated with pad parasitics, as this routing is significantly small, and generally is considered to be zero in simulation.

An example of the simple model specification follows:

```
setup pad -power -r 0.010000 -c 0.5
setup pad -ground -r 0.010000 -c 0.5
setup wirebond -power -r 0.245000 -l 1420 -c 5
setup wirebond -ground -r 0.245000 -l 1420 -c 5
setup package -r 0.001000 -l 10 -c 10
```

RedHawk also supports more detailed distributed RLCK and S-parameter package and board models for simulation; these models are described in the following sections.

Distributed RLCK Package and Board Subcircuit Model

For RLCK package subcircuit models to accurately analyze the dynamic voltage drop associated with these variations, the package parasitics must be in the form of a subcircuit description following conventional SPICE syntax. Ideal voltages are applied at the package balls through the subcircuit, which is connected to the die during analysis. The list of ports in the subcircuit definition may include signal, power and ground ports.

Note that asymmetric power and ground models are supported. The following circuit elements in the subcircuit netlist are supported.

Symbol	Element
R	Resistance
L	Self inductance
Kxx	Mutual inductance
C	Capacitance
H	Current-controlled voltage source
I	Current source
V	Voltage source
E	Linear voltage-controlled voltage source
F	Linear current-controlled current source
G	Linear voltage-controlled current source
H	Linear current-controlled voltage source

All elements must be fixed, with no dependencies on any other parameter.

In addition, the following functions in the subcircuit netlist are supported. .

Function	Purpose
.inc or .include	Inclusion of other SPICE-compatible files
subckt	Definition of additional subcircuits.

The entire off-chip network must be captured in a SPICE subcircuit netlist named 'REDHAWK_PKG'. In addition, the power and ground ideal sources must be defined in the top-level netlist. All definitions and instantiations in the netlist must follow conventional SPICE syntax.

Note: Voltage values provided in the package netlist are used in RedHawk analysis and the voltages specified in the GSR file are overridden.

Figure 12-1 is an illustration of the way package and board parameters are modeled in RedHawk. Each port of the REDHAWK_PKG subcircuit can be connected to one or more pads on the die (note that no node can have “0” as a name).

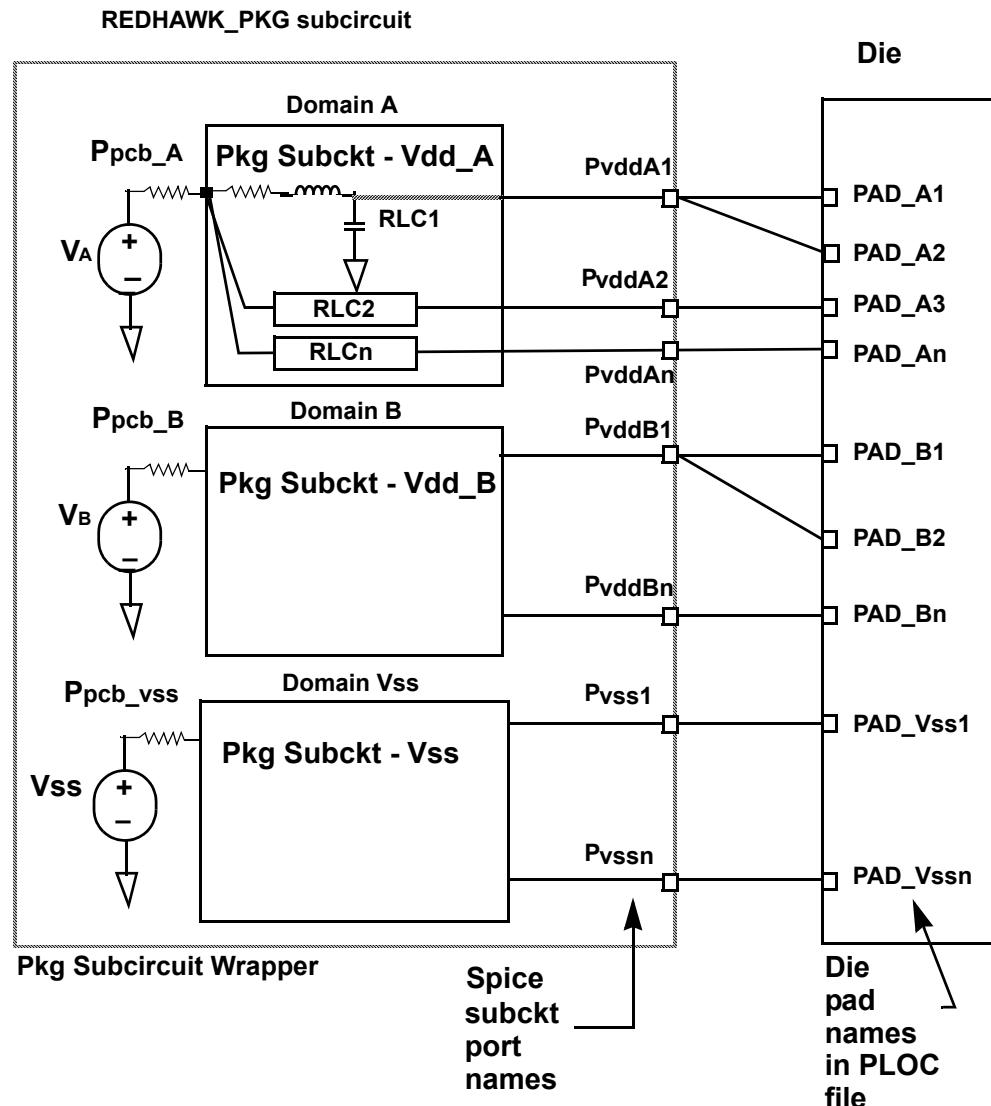


Figure 12-1 Modeling package parameters in RedHawk

Support for Package K-parameters

RedHawk supports package subcircuit models to more accurately analyze the dynamic voltage drop with off-chip package and board effects. The mutual inductor coefficient K is supported, in addition to basic Spice circuit components (RLC). The mutual inductor coefficient represents the coupling effects among inductors. If the mutual inductance between inductors L1 and L2 is M12, the coefficient K is defined as ‘M12 / sqrt(L1*L2)’. In the package Spice netlist it appears as:

K_L1_L2 L1 L2 0.5

The mutual inductance coefficient must have an absolute value between 0 and 1.0. You can specify the K parameter for each pair of inductors for an inductor group ($i = 1 \dots N$). So the self inductances L_{11} to L_{NN} , along with mutual inductances M_{ij} ($i = 1 \dots N, j = 1 \dots N$) constitutes an inductance matrix. In package modeling, such an inductance matrix is usually extracted by field solvers.

Following is an example of a package Spice netlist, *top.sp*, in a single file, for the model illustration above, including mutual inductance between inductors, K12:

```
* Top level subcircuit
* The file name must be specified with a GSR keyword
* All ports are mapped (i.e., connected) to RedHawk pads defined
* in the *.ploc file.
* The subcircuit must be named REDHAWK_PKG

.subckt REDHAWK_PKG PvddA1 PvddA2 PvddB1 PvddB2 Pvss1 Pvss2
* PCB voltage supplies
Va Ppcb_A 0 1.1v
Vb Ppcb_B 0 1.0v
Vvss Ppcb_vss 0 0v
* Instantiate domain subcircuits. Connect them to the
* corresponding PCB voltage supplies and output ports of
* REDHAWK_PKG
XVDDA_RLC Ppcb_A PvddA1 PvddA2 VDDA_RLC
XVDDB_RLC ...
XVSS_RLC ...
.ends

* Package RLC for each voltage domain can be captured in a
* separate subcircuit
.subckt VDDA_RLC Ppcb Pdie1 Pdie2
R1 Ppcb N1 1e-9
L1 N1 Pdie1 1e-12
K12 L1 L2 0.3
C1 Pdie1 0 1e-10
R2 Ppcb N2 0.5e-09
L2 N2 Pdie2
.ends
.subckt VDDB_RLC Ppcb Pdie1 Pdie2
...
.ends
.subckt VSS_RLC Ppcb Pdie1 Pdie2
...
.ends
```

In order for **RedHawk** to include the subcircuit model in the simulation, the top-level SPICE netlist must be specified in the **RedHawk** GSR file, as follows:

```
PACKAGE_SPICE_SUBCKT top.sp
```

Linear Current- and Voltage-Controlled Source Models

The following Spice linear current- and voltage-controlled sources are available for package modeling in **RedHawk**:

- Type “E” - Linear Voltage-Controlled Voltage Source
- Type “F” - Linear Current-Controlled Current Source
- Type “G” - Linear Voltage-Controlled Current Source
- Type “H” - Linear Current-Controlled Voltage Source

The syntax for describing these linear source models is given below. Note that Spice source-type keywords, such as ‘VCVS’ and ‘CCCS’, are optional in the RedHawk syntax.

Linear Voltage-Controlled Voltage Source (E)

E* N+ N- ?VCVS? NC+ NC- VGain

where

E*	name of the source, starting with E
N+	name of positive node
N-	name of negative node
NC+	name of positive controlling node
NC-	name of negative controlling node
VGain	voltage gain

Linear Current-Controlled Current Source (F)

F* N+ N- ?CCCS? Vname IGain

where

F*	name of the source, starting with F
N+	name of positive node
N-	name of negative node. Current flows from the positive node through the source to the negative node
Vname	name of the voltage source through which the controlling current flows.
IGain	current gain

Linear Voltage-Controlled Current Source (G)

G* N+ N- ?VCCS? NC+ NC- TransC

where

G*	name of the source, starting with G
N+	name of positive node
N-	name of negative node
NC+	name of positive controlling node
NC-	name of negative controlling node
TransC	transconductance (voltage to current conversion factor)

Linear Current-Controlled Voltage Source (H)

H* N+ N- ?CCVS? Vname TransR

where

H*	name of the source, starting with H
N+	name of positive node
N-	name of negative node
Vname	name of the voltage source through which the controlling current flows
TransR	trans-resistance in Ohms (current to voltage conversion factor)

Note: The dummy sources for types “F” and “H” must have a voltage of 0.

S-Parameter Package and Board Modeling for Static Analysis

S-parameter package models or PCB models in REDHAWK_PKG subcircuit support static analysis in RedHawk. The same package and PCB models that are used in dynamic analysis can be used in static analysis.

The procedure to use S-parameter models in static analysis follows:

1. Make sure that S-parameters data is available at a sufficiently low frequency, ideally at DC ($f=0$). Sufficiently low frequency for any design depends on the electrical length of the structure, and the values of decoupling capacitances present on the PCB. For example, for a package model, 1 KHz would be a sufficiently low frequency. For a PCB model with large decoupling capacitors, $f \leq 10$ Hz would be required. Having an S-parameter model outside of these guidelines could cause modeling problems. Using DC ($f=0$) data is recommended.
2. Instantiate the S-parameter models in the REDHAWK_PKG subcircuit using the same syntax as used for the dynamic simulation.

Note that there are no practical limitation on the number of ports, and the memory and runtime requirements are low. The approach involves converting the S matrix at DC to a Y matrix, and synthesizing the Y matrix as a network of resistors.

A general S-parameter modeling diagram is shown in Figure 12-2 for both static and dynamic analysis. Note that multiple ports on the BGA/VRM side are allowed.

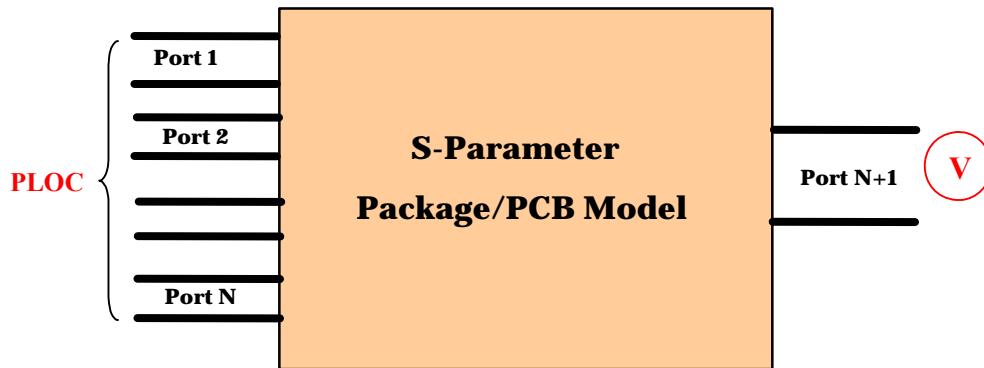


Figure 12-2 General S-parameter diagram, static and dynamic analysis

S-Parameter Package and Board Modeling for Dynamic Analysis

As the clock frequency of the chip increases, so does the importance of capturing accurate frequency behavior of the package and PCB over the multi-GHz range. Designers typically utilize full-wave electromagnetic solvers, which create frequency-dependent scattering (S) parameters. The resulting package/PCB model is represented by a frequency-domain S-parameter model, in Touchstone format. This feature allows you to utilize the results of the full-wave EM solvers in RedHawk dynamic simulations. For clock frequencies above 500 MHz, S-parameter modeling should be used to achieve accurate results.

Modeling Methodology

The S-parameter model describes, at each specified frequency point, the voltage and current relationships among all the “ports” of a black-box network. Conceptually, the S-parameters model the transfer of power from the source to the load, in terms of the incident (a) and reflected (b) waves, $b = Sa$ at a particular frequency. Both a and b are vectors of size N , and S is a complex $N \times N$ matrix, where N is the number of ports.

The k -th component of the incident wave a_k and the k -th component of the reflected wave b_k is related to the port voltages V_k and currents I_k through the relationships:

$$a_k = \frac{V_k + Z_0 I_k}{2\sqrt{Z_0}} \quad b_k = \frac{V_k - Z_0 I_k}{2\sqrt{Z_0}}$$

The definition of the S-parameters includes the value of the normalization (reference) impedance Z_0 , as shown in an example below in Touchstone format ($Z_0=2$ Ohms in this example):

```
# HZ S RI R 2.000
```

RedHawk presently only supports the scalar reference impedance, thus all ports have the same reference impedance.

In order to perform dynamic modeling of package and PCB using S-parameters, ports need to be defined at the package die pad, BGA pad locations, and/or board voltage regulator module (VRM) location. In package/PCB power delivery network modeling, a port is usually defined across a set of Vdd (power) nodes and a set of Vss (ground) nodes. In network analysis terminology, the Vss nodes are called the “reference” nodes of the defined port.

A typical port setup scheme for a flip-chip package is as follows:

1. Partition the package die pad area into N partitions.
2. Set up a port in each partition between Vdd nodes and Vss nodes.
3. On the package BGA side, set up ports as required between all Vdd nodes and Vss nodes.

Note that automatic pre-simulation time determination is provided with S-parameter packages and PCB models, the same way for RLCK models, but only in the vectorless (DvD) flow. The presim time determined in this way has an upper limit of 120 ns and a lower limit of 3.5 ns.

Connecting S-parameter models in the REDHAWK_PKG subcircuit

There are two modes of connection, common reference (ports referred to global ground), and differential (floating) reference. In practice, differential connection is used more often, as most EM solvers produce package /PCB models for ground nets with differential port definitions (between the node pairs, and not referred to global ground).

S-parameter models with a common reference node

The most intuitive way of connecting the S-parameter models is with all ports referred to global ground (SPICE node 0). In this case, each port is defined between a node N_k and the global ground. The use model for the S-parameters is essentially the same as for RLCK models. Specifically, the absolute node voltages (those with respect to the global ground) make sense, and there is no condition imposed on port currents. The sum of the port currents does not have to be zero, as there is also the reference current I_{ref} .

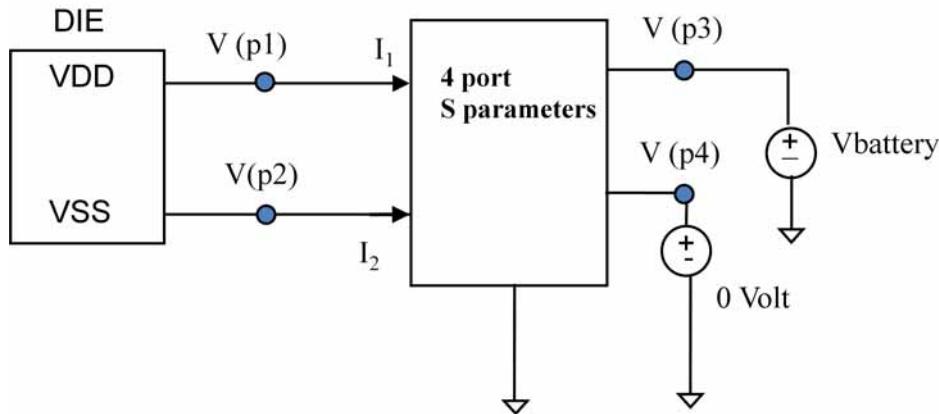


Figure 12-3 Basic global ground connection S-parameter model

Figure 12-3 shows the use of S-parameter models with all ports defined between a node and global ground (SPICE node 0). In this case the simplest possible package is a 4-port S-parameter model. From a user standpoint, there is no difference between using an RLCK package model and using an S-parameter model.

However, there are several problems in using S-parameters with ports referred to global ground:

- Many industrial EM solvers cannot extract S-parameters with ports referred to the global ground, as sometimes the definition of “global ground” or a common reference node is not clear.
- The number of ports is twice the number for the differential connection method described below.
- RedHawk simulation uses decoupled methodology by default, so having S-parameters with global ground as reference causes additional inaccuracy, as the RC branches connected to the global ground get shorted by the package.

So to use S-parameters with the ports referenced to the global ground, you must use the coupled mode (GSR keyword 'DYNAMIC_SOLVER_MODE 1') in simulation. For these reasons using differential connection of S-parameters is recommended, as described in the following section.

Differential connection of S-parameter models

RedHawk automatically detects differential connection of the S parameter packages and PCB models, and disables log reports of absolute voltages (Vdd/Vss), which are not valid for S-parameter models with differential connections.

For differential connection of S-parameters, each port voltage is defined between a pair of non-ground nodes, k(pos) and k(neg), where k is the node number. Thus an N-port S-parameter model is connected between 2N nodes. The simplest example, with 2 ports and 4 nodes, is shown in Figure 12-4, with differential voltages. All VDD pads are grouped together, and all VSS pads are grouped together.

Only the differential voltages are defined: ($V_1 = V_{1(\text{pos})} - V_{1(\text{neg})}$) and ($V_2 = V_{2(\text{pos})} - V_{2(\text{neg})}$). S-parameters also force a zero sum of currents ($I(VDD) = -I(VSS)$) by construction. The S-parameter model in the differential connection does not provide any DC path to global ground that would normally be available through the package and supply voltage sources.

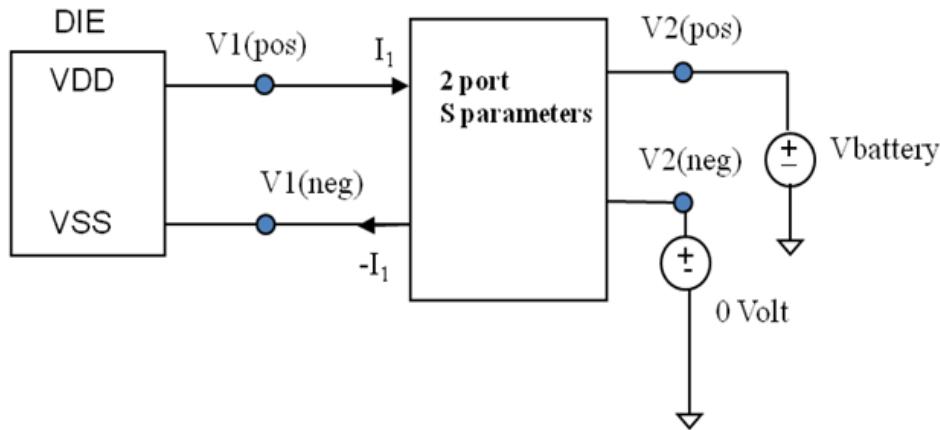


Figure 12-4 Basic differential connection S-parameter model

RedHawk considers an S-parameter model to have differential connections unless one of the following conditions is satisfied, indicating a common reference.

- The negative node of all ports is global ground, Spice node 0, which is the recommended method for S-parameter models with ports referred to a common reference.
- The negative nodes of the ports are connected to global ground with $R < 1.2e-5$ Ohms.
- The negative nodes of the ports are connected to global ground with a zero-volt ideal voltage source.

In these cases, a message is written to the log file that indicates that the S-parameter model has connections with a common reference.

Specifying S-parameter models in RedHawk

RedHawk treats an N-port S-parameter model as a 2N terminal device. In general, an N-port S-parameter network can be described in the following format:

```
Nxxxxx S(N) n1p n1n n2p n2n .... nNp nNn <modelname>
```

where Nxxxx is the network's name, 'n1p' and 'n1n' are the nodes for port 1, 'n2p', 'n2n' are the nodes for port 2, and 'N' is the number of ports in the network. The <modelname> parameter is the model statement name that contains the S-parameter data.

The model statement for an S-parameter model is defined as follows:

```
.model <modelname> nport file = "<filename>" np = <N_value>
```

where 'nport file' and 'np' are the keywords. The <filename> parameter is the name of the S-parameter data file, and <N_value> specifies the number of ports.

Figure 12-5 shows a method of partitioning flip chip bumps, where Vdd elements are red and Vss elements are black. Each partition would be represented by one port in the S-parameter model.

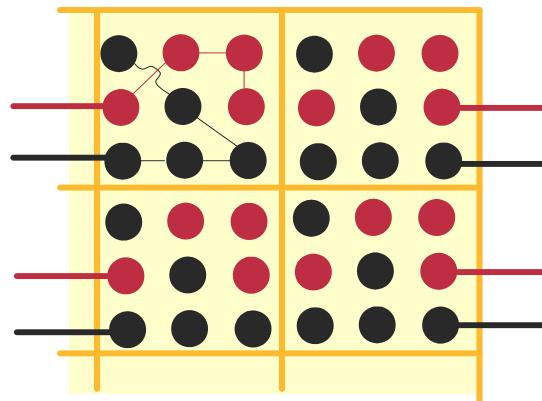


Figure 12-5 Partitioned bump model for flip chip package

The following GSR keyword SPARAM_HANDLING is used to control the behavior of the S-parameter model:

SPARAM_HANDLING 2 - enables full passivity enforcement (default)

SPARAM_HANDLING 0 - fits an RLCK model accurate at low frequencies

The following is the simplest example **RedHawk** package SPICE netlist, including a two-port s-parameter model:

```
.subckt REDHAWK_PKG pVdd pVss
Npkg S(2) pVdd pVss bgaVdd bgaVss PKG_MODEL
.model PKG_MODEL nport file = "pkg.s2p" np = 2

Vvdd bgaVdd      0      1.2
Vvss bgaVss      0      0.0
.ends
```

Note that multiple S-parameter models in the same REDHAWK_PKG are supported.

DC voltages at package nodes are included in the file *adsRpt/Dynamic/Vdc.txt*, which enables debugging package connectivity problems before finishing transient simulation.

Saving/reuse of rational approximation and passivity enforcement results

For a large number of ports, processing an S-parameter model may take significant time and hence the results of this processing are saved in a subdirectory called *SparmCache* in the run directory. These results are then available for reuse. The *SparmCache* directory contains a copy of the original user-specified Touchstone file (*.s#p), and also the file containing the results of rational approximation / passivity enforcement, called *.r#p.

If a separate run directory is used, you can copy the *SparmCache* directory, and the results of the rational approximation passivity enforcement is reused.

Recommendations and limitations in using S-parameter models

1. The number of ports should not exceed 100, with 50 to 60 ports being a practical limit for improved runtime / reliability.
2. Only one reference impedance for all ports is supported (the vector of reference impedances is not supported at this time).

3. Touchstone 2.0 format is not yet supported.
4. The touchstone file should contain the S-matrix, not Y or Z matrices. The second keyword in the header of touchstone files should be 'S' for S-matrix :


```
# HZ S RI R 2.000
```
5. RedHawk's automatic presim time determination does not currently support S-parameter models. The presim time can be determined through a transient Spice analysis of CPM + Pkg + Board models.

Usage summary and example

1. Make sure the GSR keyword 'SPARAM_HANDLING 2' is set (default).
2. Perform a basic check of the S-parameter model: In the SUtility, check that at low frequencies there is transmission between ports that have DC connections. For example, if ports 1 and 6 are connected at DC, and no other ports have DC connection to ports 1 and 6, S(1,6) should approach 1 (0 dB) at DC.
 If a VRM port connects to 10 other ports, the corresponding transmission term will be around 0.1 in magnitude (-20 dB).
3. Do not combine multiple S-parameters models into one. Multiple S-parameter models in Redhawk_PKG subcircuit are supported.
4. Create the REDHAWK_PKG subcircuit (required), which defines the ports of the S-parameter models between a node pair (for the differential connection of S-parameters). See the 21-port example following.

S-parameter differential model example (21 ports)

```
.subckt REDHAWK_PKG
+ bg0_VDD bg0_VSS
+ bg1_VDD bg1_VSS
+ bg2_VDD bg2_VSS
+ bg3_VDD bg3_VSS
+ bg4_VDD bg4_VSS
+ bg5_VDD bg5_VSS
+ bg6_VDD bg6_VSS
+ bg7_VDD bg7_VSS
+ bg8_VDD bg8_VSS
+ bg9_VDD bg9_VSS
+ bg10_VDD bg10_VSS
+ bg11_VDD bg11_VSS
+ bg12_VDD bg12_VSS
+ bg13_VDD bg13_VSS
+ bg14_VDD bg14_VSS
+ bg15_VDD bg15_VSS
+ bg16_VDD bg16_VSS
+ bg17_VDD bg17_VSS
+ bg18_VDD bg18_VSS
+ bg19_VDD bg19_VSS

* Instantiate package model
*! Port[1] = bg0
*! Port[2] = bg1
*! Port[3] = bg2
*! Port[4] = bg3
```

```

*! Port[5] = bg4
*! Port[6] = bg5
*! Port[7] = bg6
*! Port[8] = bg7
*! Port[9] = bg8
*! Port[10] = bg9
*! Port[11] = bg10
*! Port[12] = bg11
*! Port[13] = bg12
*! Port[14] = bg13
*! Port[15] = bg14
*! Port[16] = bg15
*! Port[17] = bg16
*! Port[18] = bg17
*! Port[19] = bg18
*! Port[20] = bg19
*! Port[21] = bottom

Npkg S(21)
+ bg0_VDD bg0_VSS
+ bg1_VDD bg1_VSS
+ bg2_VDD bg2_VSS
+ bg3_VDD bg3_VSS
+ bg4_VDD bg4_VSS
+ bg5_VDD bg5_VSS
+ bg6_VDD bg6_VSS
+ bg7_VDD bg7_VSS
+ bg8_VDD bg8_VSS
+ bg9_VDD bg9_VSS
+ bg10_VDD bg10_VSS
+ bg11_VDD bg11_VSS
+ bg12_VDD bg12_VSS
+ bg13_VDD bg13_VSS
+ bg14_VDD bg14_VSS
+ bg15_VDD bg15_VSS
+ bg16_VDD bg16_VSS
+ bg17_VDD bg17_VSS
+ bg18_VDD bg18_VSS
+ bg19_VDD bg19_VSS
+ BGA_VDD BGA_VSS
+ PKG_MODEL

.model PKG_MODEL nport file = "package.s21p" np=21

* finally, we need a power supply to drive all this stuff-
* core supply
V_vdd BGA_VDD 0 1.2
V_vss BGA_VSS 0 0.000

.ends

```

Note: the supply sources on the BGA side should be between a node and ground (SPICE node 0), as shown in bold type above. Also note that presim time is automatically determined.

Recommendations for best S-parameter model extraction

The success of RedHawk dynamic simulation depends on having S-parameter models that accurately describe the behavior of the package and/or PCB in the frequency range of interest, with sufficient resolution in the frequency domain. The guidelines for achieving this are given below:

1. Frequency range in the Touchstone file: recommend fmin=0 (DC) or sufficiently close; fmax=2.5 GHz to 5 GHz. The log grid of frequency points should be used from 1 Hz to 100 MHz, then linear grid above that.
2. For PCBs with large decoupling capacitors (tens of microfarads), start at a very low frequency ($f=1$ Hz, for example), and use 15 to 20 points per decade for the log grid part.
3. Number of frequency points in the Touchstone file: typically around 250 to 500, depending on how complicated the frequency dependence is. For complicated frequency dependencies, 1000 to 2000 frequency points may be needed.
4. Reference impedance : $Z_0=2$ Ohms. A good choice of reference impedance improves the accuracy of the results. For power/ground networks, the classic reference impedance of 50 Ohms is too large. Extracting S-parameters with a reference impedance of $Z_0=2$ Ohms is the best approach, since this is the value used by default in RedHawk.
5. Number of significant figures to use in Touchstone file data: 12 to 14 decimal places are recommended. There is no penalty for extra decimal places, but having too few could be a problem.
6. Maximum number of ports: While the absolute maximum is 100 ports, limiting the number of ports to 50-60 is best.

Analysis of the Simulation Results

Special care is required when analyzing simulation results obtained with S-parameter models with the differential connections. Remember that only differential voltages (voltage differences) are meaningful using this method. For this reason when displaying voltage waveforms, only display differential voltages, $V(vdd)-V(vss)$, not single-ended voltages $V(vdd)$ and $V(vss)$. In Signal Viewer (sv), you can use the **Calculator** function to generate the differential waveform $V(vdd)-V(vss)$ from the two waveforms $V(vdd)$ and $V(vss)$. For pad voltages, you can use the plotting utility 'gnuplot' and the file *adsRpt/Dynamic/Vpad.data* that is created when a package model is present.

No such transformation is needed for current values; currents are always "absolute", and wire voltages should be ignored. The following results retain significance:

- Differential voltages at the pads $V(vdd)-V(vss)$. You can plot the voltage difference at a pair of pads of dynamic simulation results, as follows:

```
plot voltage -pad_pair {<pad1_name> <pad2_name>}  
-sv -o <file_name>
```

Example:

```
plot voltage -pad_pair {VDD1 VSS90} -sv -o vdrop.txt
```

Note that pad names are specified inside curly brackets, with no comma between pad names.

- Pad currents and battery currents
- Instance voltage drops

- Worst VDD-VSS voltage report.

Mapping Package Port Names to Die Pad Names in the PLOC File

Mapping of ports on the package to one or more pads on the die is achieved using definitions in the PLOC file. Some package vendors have protocols that can be used in mapping ports and pads. If a protocol is not available, the port/pad locations must be mapped manually.

Following is the syntax for a standard *.ploc file with no package model:

```
<die_pad_name> <X-coord> <Y-coord> <layer> <POWER | GROUND>
```

If you want to include a package subcircuit model, then the syntax of the .ploc file should be modified as follows:

```
<die_pad_name> <X-coord_um> <Y-coord_um> <layer>  
<power/ground_domain_name> <Spice_pkg_port_name>
```

where the power/ground domain name is the name specified in the VDD_NETS GSR keyword, and none of the ports can be named “0”. Signal ports in the package subcircuit can be floating. Multiple pads can be connected to a single port of the package. For example, in the package model provided multiple Vdd_A domain pads can be connected to port PvddA1.

The following is a sample .ploc file corresponding to the example design and package:

```
Pad_A1 3125 4340 metal16 VDD_A PvddA1  
Pad_A2 3225 4340 metal17 VDD_A PvddA1  
Pad_A3 3325 4340 metal16 VDD_A PvddA2  
Pad_A4 3425 4340 metal16 VDD_A PvddA2  
Pad_B1 5125 4340 metal17 VDD_B Pvddb1  
Pad_B2 5225 4340 metal17 VDD_B Pvddb1  
Pad_B3 5335 4340 metal16 VDD_B Pvddb2  
Pad_B4 5335 4340 metal16 VDD_B Pvddb2  
Pad_VSS1 6125 4340 metal17 VSS Pvss1  
Pad_VSS2 6225 4340 metal16 VSS Pvss1  
Pad_VSS3 6325 4340 metal17 VSS Pvss2  
Pad_VSS4 6425 4340 metal17 VSS Pvss2  
...
```

The *.ploc file is imported into RedHawk using standard procedure.

Chip-Die Mapping Using Package Compiler

Overview

Accurate analysis of chip designs require a model of the associated package, in terms of an accurate RLCK Spice subcircuit. The RedHawk Package Compiler can provide this as an extension of the Chip Package Protocol (CPP) header. The Package Compiler has several functions:

- wrap the package Spice model into RedHawk-compatible format.
- match die and package pins and create an annotated PLOC
- compute effective inductance
- calculate effective package Inductance for each voltage domain
- perform RLCK passivity checks
- perform package Spice syntax checks

RedHawk Package Compiler makes use of the Chip Package Protocol (CPP) header information to determine the following:

- identifies package and die pins
- identifies which nodes belong to the die side and which belong to the PCB side
- uses CPP header data to differentiate between power nets and ground nets

If Package Compiler has already been run, if invoked again it identifies the already-wrapped **RedHawk** package model and the already-annotated PLOC file, and reruns the same set of checks on the output files for completeness and correctness.

An overview of the Package Compiler process and data flow is shown in Figure 12-6.

The key features of Package Compiler is “wrapping” the package Spice model subcircuit into one that is **RedHawk**-compatible, as well as calculating effective inductance for each voltage domain and matching die and package pins.

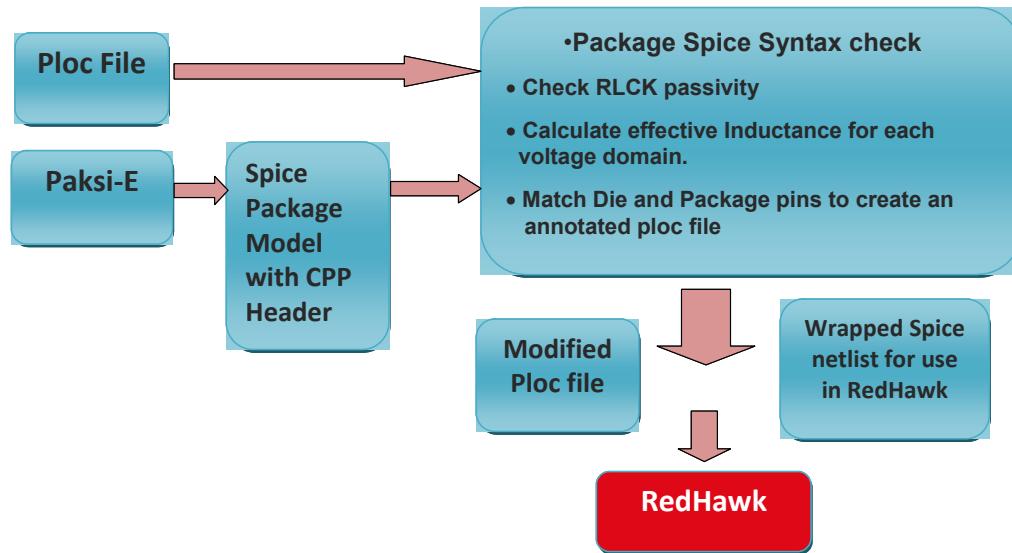


Figure 12-6 Overview of Package Compiler flow

Inputs

The required inputs are:

- PLOC file - pin location file to be annotated (*.ploc). The format of the PLOC file is as follows:


```
<pin/pad_name> <x_loc> <y_loc> <layername> <P/G_domain_name>
```

 Note that the last column should **not** be “Power” /“Ground”, but the domain name.
- package Spice file - target package Spice model with Chip Package Protocol (CPP) header to be mapped, wrapped, syntax checked, passivity checked, and effective inductance calculated.
- GSR file : standard **RedHawk** GSR file that contains information about the voltages to apply to each die power and ground net, specified with keywords

```

VDD_NETS {
    <die_VDD_name> <voltage>
    ...
}
GND_NETS {
  
```

```
<die_VSS_name> <voltage>
...
}
```

Command Syntax

The syntax for using package compiler from a UNIX command line is:

```
compile_pkg
[ -m <pkg_model_file> ] [ -p <ploc_file> ] [ -n <p_n_t_file> ]
[ -t <transform_file> ] [ -vv <port_value_file> ]
[ -vg <gsr_file> ] [ -g <generic_pkg_file> ]
[ -c ] [ -nma <#_of_match_iter> ] [ -no_leff_calc ]
[ -op <output_ploc_file> ] [ -om <wrapped_model> ]
[ -ow <working_directory> ] [ -icf <input_config_file> ]
[ -start_tol <min_dist> -end_tol <max_dist> ]
[ -snap ] [ -ee]
```

where

- m <pkg_model_file> : specifies the package model Spice file.
- p <ploc_file> : specifies the die pin locations and corresponding die nets.
- n <p_n_t_file>: specifies the file defining the package net types
- t <transform_file> : specifies the transformation operations to use for die/ package pin matching.
- vv <port_value_file> : specifies the voltages to assign to package power/ground PCB pins.
- vg <gsr file> : reads the die net voltages from the GSR file.
- g <generic_pkg_file>: If no -m file supplied, this file can be used to indicate package pin locations and corresponding package nets.
- c : Checks the data between the already annotated ploc file and -m pkg file.
- nma <#_of_match_iter> : specifies the maximum number of matching attempts.
- no_leff_calc : the effective inductance is not calculated, which reduces runtime.
- op <output_ploc_file> : specifies the name of the output PLOC file.
- om <wrapped_model> : specifies the name of the wrapped Spice model
- ow <working directory> : specifies the working directory for the intermediate files
- icf <input_config_file> : specifies the file that contains input arguments to this utility
- start_tol <min_dist> -end_tol <max_dist> : specifies the minimum and maximum separation distance in mapping die and package pins (meters)
- snap : snaps each die pin to the package pin closest to it, even if outside the tolerance radius.
- ee : forces die ground pins to attach only to package ground pins and die power pins to attach only to package power pins

Note that if you run Package Compiler *without* the '-ee' and '-snap' options, a file with a list of unmatched ploc pins, called *adsPackage/unmatchedPlocPins.txt*, is generated, with entries such as the following:

```
+ Best Match: 1/3126 pins lying outside of tolerance 1.8um,
with greatest offset of 264.552452um
+ The list of unmatched ploc pins: [ adsPackage/
unmatchedPlocPins.txt ]
!! NOTE: Since -snap option was NOT specified, all pins
```

lying outside of the tolerance will be commented out in the annotated ploc file.

Using the ‘-ee’ and ‘-snap’ options, Package Compiler snaps any PLOC points lying outside of the tolerance threshold to the closest related package point and no *unmatchedPlocPins.txt* file is generated.

Outputs

The primary output is the conversion of the original package Spice netlist, *<pkg_spice>*, of the form:

```
subckt pkg die1 die2 pcb1 pcb2
...
.ends
```

into the “wrapped” RedHawk-compatible package Spice netlist, *redhawk_pkg*, that has the format:

```
include <pkg_spice>
.subckt redhawk_pkg die1 die2
V1 pcb1 0 1.2
V2 pcb2 0 0
Xpkg die1 die2 pkg
.ends
```

Note that the new Spice file includes the original package Spice file. The subckt netlist excludes the nodes on the side of the PCB, which are assigned appropriate voltages described by the voltage file.

Additional outputs are described following:

- *<ploc_file>.1* - fully-annotated ploc file after successful die and package pin pair mapping.
- standard checks of circuit syntax, such as:
 - annotated Spice nodes and nets exist in the package Spice model
 - annotated Spice nodes belong to the correct package net
 - die side Spice nodes are not used as voltage sources in the Spice file
 - Spice nodes do not have multiple types
 - Spice nodes do not belong to multiple nets in CPP header
 - PLOC file is fully annotated
 - PCB Spice nodes have valid voltages
- passivity check - checks RLCK passivity for the package Spice model
- package net types file (*.pnt) - automatically created upon successful pin-pair mapping, and required for effective inductance calculation. In the event that the file is unavailable (if Package Compiler has not been run yet), it can be provided by using the ‘-n’ option. The file format is:

```
.POWER
<pkg_net1>
<pkg_net2>
...
.GROUND
<pkg_neta>
<pkg_netb>
...
```

- effective inductance calculation for each Power and Ground net pair. The Package Compiler makes use of the CPP (Chip Package Protocol) header to gather information needed for the calculation. To determine which package nets are power and which are ground, it makes use of the file, which is automatically created upon successful pin-pair mapping. For example, for the sample PNT file above, it calculates the inductances:

```
<pkg_net1> to <pkg_neta>
<pkg_net1> to <pkg_netb>
<pkg_net2> to <pkg_neta>
<pkg_net2> to <pkg_netb>
```

A testbench is created for each calculation, named:

```
<pkg_model>. <power_name>_<gnd_name>.leff.sp
```

for example, *model.sp.pkg_net1_pkg_neta.leff.sp*.

By default, the utility uses the NSpice binary to extract the effective impedance (\$APACHEROOT/bin/nspice)

You can use the 'sv' waveform viewer to view the waveform file generated [*.aa0].

- *.ploc_file.map*.*1* file - contains the list of die -to -package pin mappings.
- *.ploc_file.T* file - contains the transformation parameters (rotation, mirror, translation) used to match the die to the package
- *.leff.sp files - testbench Spice files used to extract the effective inductances between power and ground package net pairs.
- *.leff.aa0 files - contain the effective inductance graphs that can be viewed with the 'sv' viewer.

Known Restrictions

The following restrictions must be observed in package models:

- Only the previously-described SPICE keywords are supported.
- Mapping the nodes of the package to their locations on-chip is currently supported only through the *.ploc* file.
- The user-created top-level SPICE netlist must use the 'REDHAWK_PKG' keyword as the subcircuit model.

Chapter 13

Low Power Design Analysis

Introduction

VLSI designs, especially those manufactured in a 90nm scale process or smaller, and those targeted for wireless, mobile computing, and other low-power applications, are employing power saving modes for extended battery performance, greater control over the speed-versus-power trade-off and better thermal performance. Sub-micron designs suffer from increased device leakage currents, less control of threshold voltage levels from negative bias temperature instability (NBTI), increased current drive strengths, and higher device densities. Designers are becoming increasingly aware of the need to make their designs “power-aware” at several levels – system, architectural, design, and technological. Though the most significant impact on power saving may be achieved through changes at the system or architectural levels, most of the visible efforts are focused on design and technological level changes to mitigate the harmful effects of higher power consumption in today’s designs.

Some commonly employed power saving design techniques, include:

- Multiple Vdd/Vss domains
- Power gating, in which Vdd /Vss supplies for selected areas of the circuit, or individual cells are controlled by one or more switches or sleep transistors. Techniques such as this are also sometimes referred to as “MTCMOS”.
- Clock gating, which controls the clock signals supplied to selected storage elements in a design
- Voltage islands, in which selected areas of the design are driven by different voltage supplies
- Multi-Vth (threshold voltage) circuit design styles, in which higher Vth transistors are strategically placed in the circuit to reduce leakage without adversely affecting critical path delay
- Active gate (body) back-biasing technique, in which Vth levels are dynamically controlled to minimize leakage power
- Retention flip-flops, in which data/states for sections of the circuit are preserved during temporary power-down periods

RedHawk™ can provide accurate dynamic power analysis for designs using the power saving techniques described above. Designers can perform full-chip and block-level *transient* dynamic voltage analysis and average static voltage drop analysis by accurately modeling one or more of the above power saving modes in their designs. They can use the textual result reports and the graphical displays, including time point based movie mode display of dynamic voltage drop results, to analyze and improve their designs.

Analysis of Multiple Vdd/Vss Domain Designs

One of the popular techniques in designing low power chips is to use multiple voltage sources. RedHawk supports design techniques based on multiple VDD domains. The input of multiple VDD domains is accomplished by specifying the multiple VDD domain (net) in the *.gsr* file, as shown in the following example:

VDD_NETS

```
{  
VDD0 1.8  
VDD 1.6  
}
```

You can have any number of VDD domains, but you can specify a maximum of eight different voltage values. One instance connected to multiple VDD nets is supported by defining power distribution among multiple VDD/VSS domains using P/G arc definitions. Power/ground arcs for multiple VDD/VSS designs are defined in [section "P/G Arc Definitions in Custom LIB Files", page 3-20](#), and [section "Cell Characterization Data Preparation", page 9-217](#).

Also, if the instance is an I/O cell connected to I/O VDD, core VDD, and I/O VSS (which can be the same as core VSS), RedHawk can handle it with an asymmetric current profiles in dynamic analysis. This is how I/O cells' impact on core VDD and VSS is handled.

To view the results of a particular net analysis, you can select a net from [View -> Nets](#). Figure 13-1 shows the voltage drop color map when all nets are selected. Figure 13-2 shows when only one VDD net is selected.

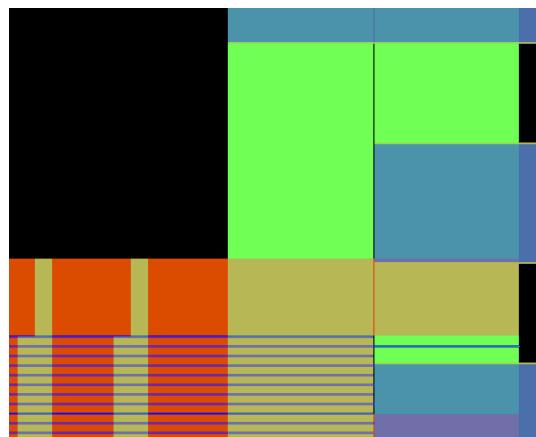


Figure 13-1 Viewing multiple VDD domain nets

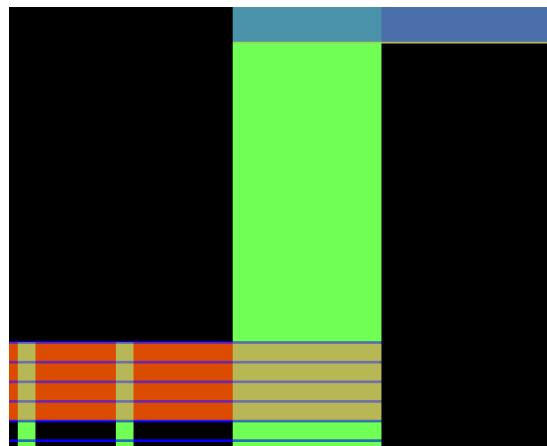


Figure 13-2 Viewing only one VDD net

You can zoom in and click on the node on a net to view the voltage at that location, as shown in Figure 13-3.

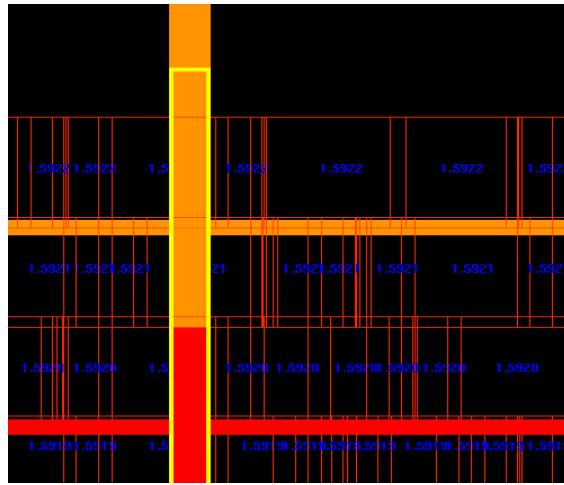


Figure 13-3 Zooming in on a net to view voltage drop information

The following are sample results shown in the message window:

```
Net: VDD
Wire: M2
LowerLeft = (3710.510, 4388.160)
Length: 101 um
Width: 1.46 um
Resistance: 6.95 Ohm
Voltage at query position: 1.595129 V
Current at query position: 0.000168779 A v
```

You can click on an instance, as shown in Figure 13-4, to view the information on that instance.

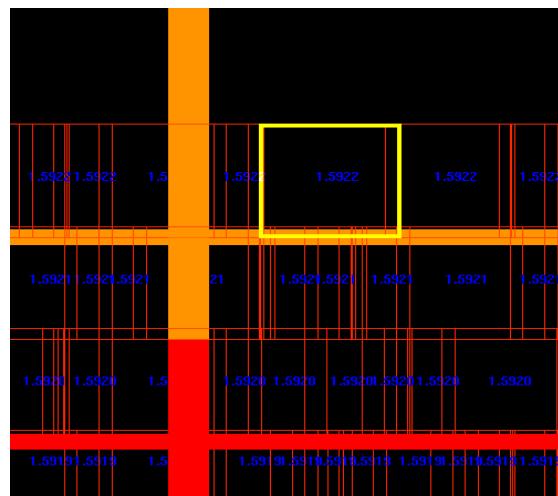


Figure 13-4 Zooming in on an instance to view voltage drop information

Following are the results shown in the message window.

```
Instance: Test/U594
Cell: test1
```

```
LowerLeft = (3713.860, 4482.880), Width/Height = 5.220 / 4.639
Total Load Cap: 0.131 pF
Intrinsic Cap 0: 0 pF
Intrinsic Cap 1: 0 pF
Static Vdd-Vss: 1.592 V (domain vdd: 1.6000)
Average Pwr: 6.72e-10 W
```

You can also select **Results -> List of Worst IR instance for Static Simulation** or **List of Worst DvD instance for Dynamic Simulation** to view up to 1000 instances with the highest voltage drop. In the ‘Report of Worst IR Instance’ window, you can select a net of interest from the combo box **Vdd Domain**, and click **Apply**. The default is **All**, for which the report prints the instances with the highest voltage drop on the full chip.

The report contains the following five fields.

1. The instance rank (top 1000 instances with the highest voltage drop)
2. The lowest VDD-VSS differential for the instance
3. Ideal VDD value
4. Location (x, y)
5. Instance name

Analysis of Power Gating Designs

This section describes the data preparation steps for using **RedHawk** in analyzing designs employing *power gating* techniques. Two levels of power gating can be analyzed--the full chip level involving blocks that are powered up and powered down, and also each block may contain subcircuits with independent power gating switches. The basic analysis procedure includes the following steps, and is shown in Figure 13-5:

1. Create and configure individual power gating switch models for ON, OFF, and Power-up conditions using configuration (.conf) file, and Spice netlists/models.
2. Perform APL characterization of switch models using *ap/sw* utility.
3. Define chip-level block On-Off switching with GSC file and STA timing window information.
4. Use the APL utility *apl/di -w -s2* to characterize all cells in the design and generate the required PWL models.
5. Import switch definitions and block power conditions into **RedHawk** using the GSR file.
6. Import standard design information into **RedHawk** database using the Apache Tech file, and GSR file (LEF, DEF, LIB, GDSII).
7. Perform accurate, SPICE-based full-chip static and transient dynamic analysis in **RedHawk**.

These steps are described in the following sections.

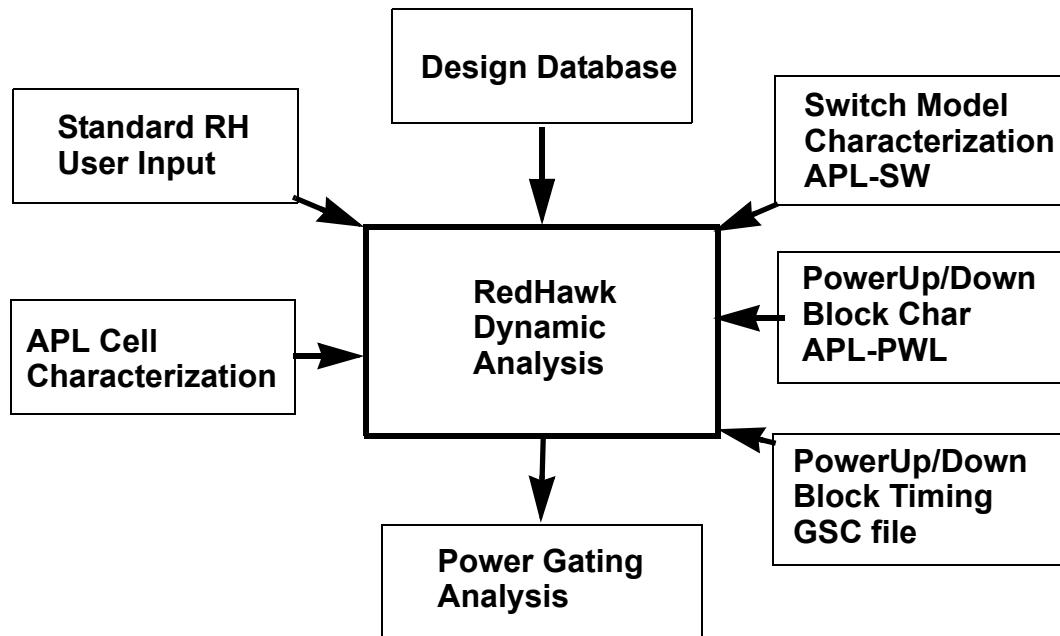


Figure 13-5 Flow for Power Gating and Switch Analysis

Types of Power Switches

There are two types of power gate switches that can be used in low power designs:

- charge switches - large high-current switches that are used to insure that the supply voltage for associated functional switches have reached the proper threshold voltage before the first functional switch is turned on
- functional switches - standard switches used in power gating designs

For designs that use charge switches, the cell names and their threshold voltage must be declared using the GSR keyword CHARGE_SWITCH. For a description of its use, see section "[CHARGE_SWITCH](#)", page C-722.

Low Power Analysis Switch Modeling

Some designs include circuitry that provides the ability to turn power to a cell, block or subcircuit ON or OFF as needed. This is usually accomplished through the addition of a switch circuit in series with the power or ground net serving the subcircuit. An example of such a network is shown in Figure 13-6, with the external VDD supply connected to the

power supply internal to the subcircuit through a switch circuit. This type of switch is commonly referred to as a *header* switch.

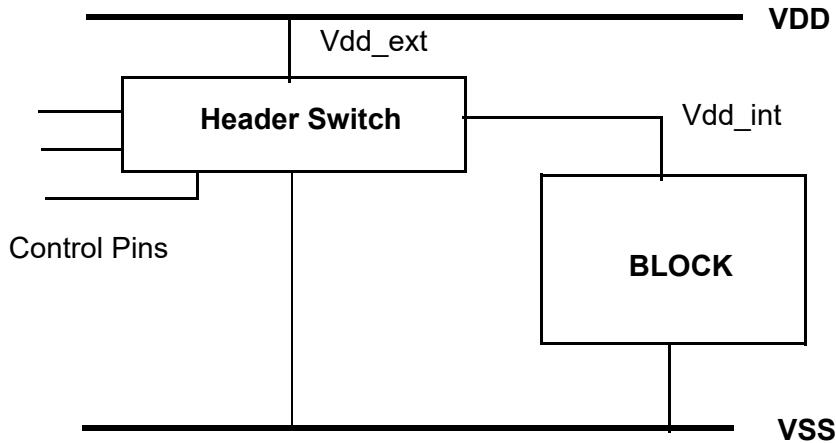


Figure 13-6 Example of a design using VDD power switching through a header switch.

When a switch controls the conduction path between the external and internal ground network, it is commonly referred to as a *footer* switch, shown in Figure 13-11. A particular subcircuit or block typically would have only one or the other type of power gating, but it could have both if there was a design reason.

During the power-up sequence of a design, the switches may be turned on or off all at once or one at a time, and may take several clock cycles to reach full On or Off state. Usually the individual power gated circuits are not turned off or on simultaneously, to reduce the associated large transient current flow and complications of simultaneously switching output (SSO) and voltage drop noise. RedHawk accurately considers the powerup sequence of blocks and individual switches , allowing designers to estimate the number of cycles needed for powerup in order to avoid large transient current spikes. It models the switch behavior in a way that reflects their turn-on and turn-off mechanisms.

To provide accurate analysis, there are continuous power-up sequences for each switched section of the circuit that must be adequately modeled. Each switched section of the design requires four different analysis modes: ON, OFF, and POWERUP, and POWERDOWN. The switch models and methods needed to perform detailed analysis of these four conditions are described in the following sections. It is assumed that the switch contains at least one transistor, but the actual design can be determined by the user to maximize speed and minimize leakage.

ON State

During the ON-state, the header and footer switches form a low resistance element between the “external” and “internal” power networks. Regular dynamic and static analysis of a design can be performed in this mode. RedHawk reports the voltage across the switches and the current through the switches from the dynamic and/or static analyses.

Figure 13-7 shows the equivalent circuit model for a header switch in the ON state, which is characterized using the *aplsw* utility, which automatically generates a switch model file containing switch performance data for ON, OFF and PowerUp states. The UNIX shell command syntax is:

```
aplsw [-c] [-d] [-o <output_file>] <sw_config_list.conf>
```

where

- c : check generated result
- d : debug mode
- o <output_file> : output file name
- <sw_config_list.conf> : switch configuration list file

The key components of the On model are a small saturation R value, some capacitance, and a small leakage current that is relatively independent of the voltage applied or the internal resistance.

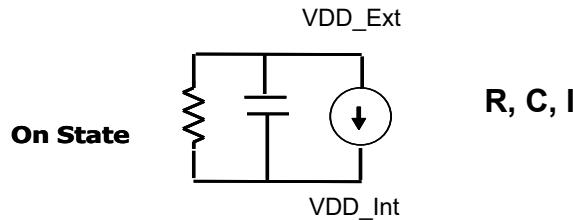


Figure 13-7 ‘ON’ state electrical model for a header switch

Adaptive ON Resistance

For improved accuracy, RedHawk supports adaptive modification of the switch On resistance (R_{on}) based on the supply voltage variation seen at the control pin of the PFET (header) or the NFET (footer). For example, for a footer-based device, if the supply voltage variation at the gate during transient simulation is significant compared to a particular threshold, then RedHawk chooses the appropriate switch R_{on} for that particular combination of terminal voltages at the simulation time point. The gate voltage is obtained by monitoring the appropriate voltages (VDD for headers and VSS for footers) during an ON-state dynamic analysis in the neighborhood of the switch cells. Then RedHawk uses this gate voltage relationship to determine the appropriate switch R_{on} from the switch model file (see section "Switch Model Generation", page 13-354). This feature is activated using the GSR keyword:

```
DYNAMIC_ADAPTIVE_RON [0|1] <variation_threshold_%>
```

For example, 'DYNAMIC_ADAPTIVE_RON 1 8' specifies that the threshold for R_{on} change is an 8% variation in power or ground voltage. The default value is 0 (Off).

The power switch should ideally be characterized using the APLSW keyword 'MD_PWL 1' to use the multi-dimensional switch model file, which provides the most accurate electrical representation of the switch. If the default switch model is used, RedHawk uses internal heuristics to scale the switch R_{on} .

OFF State

In the Off state, the voltage levels in the power and ground networks are determined from the equilibrium reached between the leakage currents in the instances in the design and the leakage currents through the switches. Figure 13-8 shows the equivalent circuit model for a header switch in the OFF state, which is characterized using the `aplsw` utility and

the header switch model file. The key components are a capacitance and a small voltage-dependent leakage current.

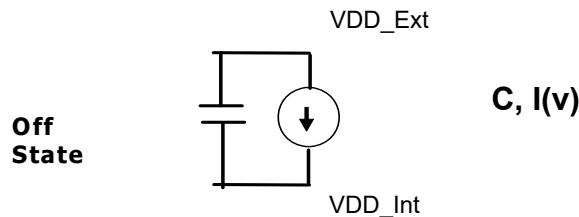


Figure 13-8 ‘Off’ state electrical model for a header switch

RedHawk allows for the accurate determination of the OFF state equilibrium voltage in the internal power or ground networks, which in turn helps to determine the leakage current flowing in the design in the Off mode. As shown in the Figure 13-9 below, APL considers only the current I_{sw_leak} for leakage current characterization of the switch cell. The OFF state analysis is required to determine the initial voltage for a power-up sequence.

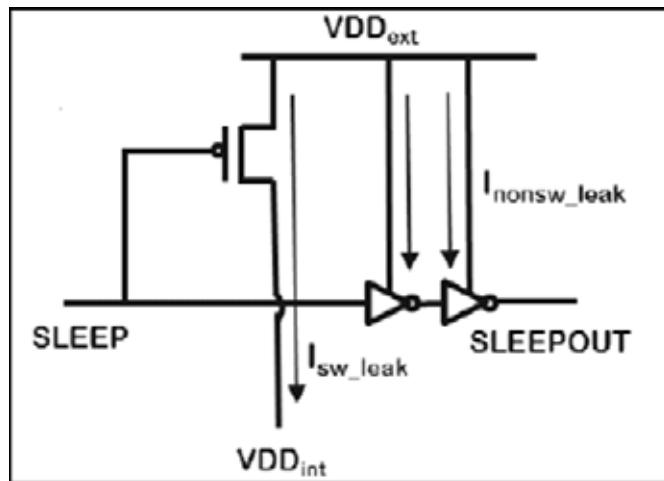


Figure 13-9 Leakage switch model in OFF state

PowerUp and PowerDown States

A design employing header switches, during power-down the internal power network, which is initially close to the supply voltage, settles to a low voltage level. The final Off voltage is determined by the leakage in the instances connected to the internal power network and the leakage current through the header switches that control the voltage of that network.

Conversely, for a design employing footer switches, during power-down the internal ground network rises to a voltage value determined by the leakage in the instances connected to the internal ground network and the leakage current through the footer switches that control the voltage of the internal ground network.

Powerup reverses these conditions and returns the circuit to the original On state over some period of time that is determined by the characterization process. RedHawk allows for the accurate determination of power up and power down conditions.

The equivalent model used to characterize a switch during power-up periods is shown in Figure 13-10.

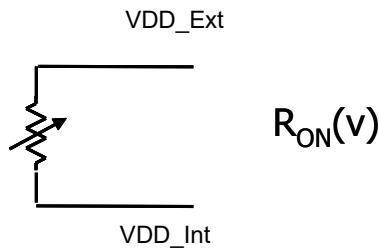


Figure 13-10 ‘PowerUp’ and ‘PowerDown’ state electrical model

Control of Power Gating Switches

Control pins are used to manage *when* power is applied to the block or subcircuit and when it is removed. Pins required to connect and control a *header* switch have the following types: control (one or several), vss supply, vdd_external/supply, vdd_internal, vdd_high/bias. A corresponding set of pin types are needed for a *footer* switch: control (one or several), vdd supply, vss_external/supply, vss_internal.

Enhanced mode switch characterization takes enable pin transitions into account.

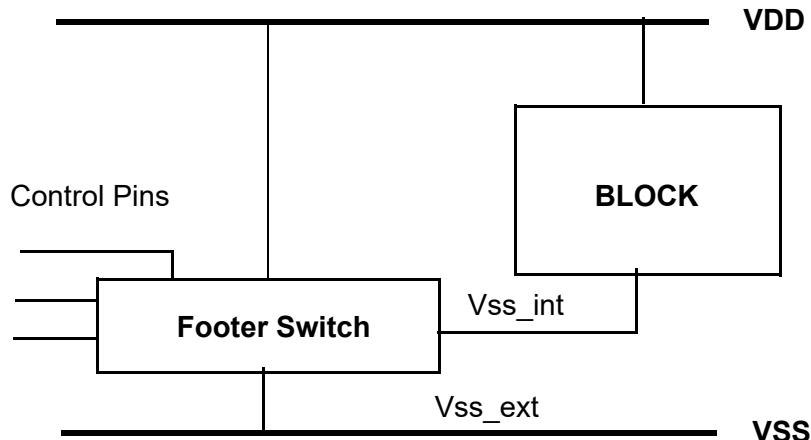


Figure 13-11 General power gating circuit using a footer switch

The next section describes the configuration and characterization process that defines the actual pin names, how they are controlled, and the voltage levels associated with each condition.

Checking Switches with Two Enable Pins

When one power switch is controlled by two enable pins, you need to make sure that the proper threshold voltage is reached when the second pin is enabled. Using the GSR keyword **CHECK_SWITCH_POWERON** ensures the proper working of the switch. The syntax for the keyword is:

```
CHECK_SWITCH_POWERON {
    [ THRESHOLD <threshold_all_V> |
      <cellname> <cell_threshold_V> ]
    ...
}
```

```
}
```

where

<threshold_all_V> : specifies the threshold voltage (Volts) for all switch cells
 <cellname> <cell_threshold_V> : specifies the threshold voltage (Volts) for switch cells of type <cellname>

A report, *switch_poweron.rpt*, lists the switches in the design by threshold voltage and identifies any violations, as shown in the sample report in Figure 13-12. In this example, for a timing window of 2.2 ns, the footer switch 'sw_inst1' only discharges to a value of 0.272049 V, while it should reach 0.1 V for proper operation, so a threshold violation is reported.

Threshold= 0.1 , TW for sw_inst1/ECLK is 22.6e-10
 => Violation

#Report switch power-on voltage	#instance_name	threshold	power_on_voltage	violation
	sw_inst23	0.1	n/a	n/a
	sw_inst22	0.1	n/a	n/a
	sw_inst21	0.1	n/a	n/a
	sw_inst20	0.1	n/a	n/a
	sw_inst19	0.1	n/a	n/a
	sw_inst18	0.1	n/a	n/a
	sw_inst17	0.1	n/a	n/a
	sw_inst16	0.1	n/a	n/a
	sw_inst15	0.1	n/a	n/a
	sw_inst14	0.1	n/a	n/a
	sw_inst13	0.1	n/a	n/a
	sw_inst12	0.1	n/a	n/a
	sw_inst11	0.1	n/a	n/a
	sw_inst10	0.1	n/a	n/a
	sw_inst9	0.1	n/a	n/a
	sw_inst8	0.1	n/a	n/a
	sw_inst7	0.1	n/a	n/a
	sw_inst6	0.1	n/a	n/a
	sw_inst5	0.1	n/a	n/a
	sw_inst4	0.1	n/a	n/a
	sw_inst3	0.1	n/a	n/a
	sw_inst2	0.1	n/a	n/a
	sw_inst1	0.1	0.272049	YES

Figure 13-12 Sample *switch_poweron.rpt* for a footer switch

Characterization and Implementation

The header or footer switches need to be characterized to model their electrical properties for the different analysis modes. Three different characterizations are performed to determine switch properties for the following modes: On, Off, PowerUp, and PowerDown.

Switch Configuration Files

A configuration file must be created for each switch type to specify the different pins in the switch and their voltage settings. The following are examples of the keyword settings required in the switch configuration file, along with a definition and the proper syntax. Note that multiple voltage switch characterization can be performed, and that in this case 'pin <valueN>' settings for all the states must be consistent with the VDDVALUE specification.

See section ["APL Configuration File Description"](#), page 9-218, for more details on keyword use.

SWITCH_TYPE HEADER

Specifies either header or footer switch type. Syntax:

SWITCH_TYPE [HEADER | FOOTER]

DEVICE_MODEL_LIBRARY 0.13um.lib TT

Defines SPICE device model library. Syntax:

DEVICE_MODEL_LIBRARY <model_file> <process_corner>

TEMPERATURE 25

Defines characterization temperature. Syntax:

TEMPERATURE <temperature_degreesC>

SPICE_NETLIST my_switches.sp

Defines switches' Spice netlist file. Syntax:

SPICE_NETLIST <switch_name.sp>

INCLUDE spice.models

Defines Spice model card file. Syntax:

INCLUDE <model_card_name>

EXT_PIN vdd_ext

Defines switch input (supply connection) pin name. Syntax:

EXT_PIN <pin_connection_to_external_supply>

INT_PIN vdd_int

Defines switch output (circuit connection) pin name. Syntax:

INT_PIN <pin_connection_to_cell>

GND_PIN_NAME vss

Defines GND pin name -- for HEADER switches ONLY. Syntax:

GND_PIN_NAME <gnd_pin_name>

VDD_PIN_NAME vdd

Defines VDD pin name -- for FOOTER switches ONLY. Syntax:

VDD_PIN_NAME <vdd_pin_name>

VDDVALUE 1.1

Defines one or more VDD supply values. Syntax:

VDDVALUE <value1> [<value2> ...]

ON_STATE {

ENABLE1 0

ENABLE2 1.1

}

Defines the voltages on control pins needed to turn the switch fully ON. Syntax:

ON_STATE {

<control_pin> <ON_V1> [<ON_V2> ...]

}

OFF_STATE {

ENABLE1 1.1

ENABLE2 0

}

Defines the voltages on control pins needed to keep the switch OFF. Syntax:

OFF_STATE {

<control_pin> <OFF_V1> [<OFF_V2> ...]

}

```
POWER_UP {
    ENABLE1 0
    ENBALE2 1.1
}
```

Defines the voltages on the control pins needed to turn ON the switch. Syntax:

```
POWER_UP {
    <control_pin> <Up_V1> [<Up_V2> ...]
}
```

Note that when there are two control pins, the value of one of the control pins in the POWER_UP state must equal its value in OFF_STATE, then this control pin is considered to fire second in operation. Without this information, the pin firing order cannot be decided. For example:

```
OFF_STATE {
    Control2 0
    Control1 0
}
POWER_UP {
    Control2 0.99
    Control1 0
}
```

In this example, pin ‘Control2’ fires first.

```
POWER_DOWN {
    PinA Va
    PinB Vb
}
```

Defines the voltages on control pins needed to turn OFF the switch. Syntax:

```
POWER_DOWN {
    <control_pin> <Down_V1> [<Down_V2> ...]
}
```

Note that when there are two control pins, the value of one of the control pins in the POWER_DOWN state must equal its value in ON_STATE, then this control pin is considered to turn off second in operation. Without this information, the pin firing order cannot be decided. For example:

```
ON_STATE {
    Control2 1
    Control1 0
}
POWER_DOWN {
    Control2 0
    Control1 0
}
```

In this example, pin ‘Control 2’ turns off first.

SIM_ON_STATE_ONLY

Enables only on-state simulation while characterization of switch cells. Syntax:

```
SIM_ON_STATE_ONLY [ 0 | 1 ]
```

CONTROL_PIN ENABLE1 R F

CONTROL_PIN ENABLE2 R F

Specifies the control pin names and the transitions (rise or fall) that turn the switch On. This uses the proper transitions in the STA file during RedHawk analysis. For no rise or fall spec, use “.”. (don’t care). Syntax:

```
CONTROL_PIN <subckt_pin_name> [for ramp-up: R | F | - ]
[for ramp-down: F | R | - ]
```

DC_BIAS vdd_high 1.1

Defines input control pins’ required DC value. Syntax:

```
DC_BIAS <subckt_pin_name> <value>
```

OPENPIN open1

Defines pins that should be kept open. Syntax:

```
OPENPIN <subckt_pin_name>
```

SPICE2LEF_PIN_MAPPING {

```
abcd fghi
```

```
}
```

Defines Spice to LEF pin mapping. Can be applied to match the switch LEF pin names in the physical model: <macro_name>_adsgds1.lef. Syntax:

```
SPICE2LEF_PIN_MAPPING {
    <subckt_pin_name> <LEF pin name>
    ...
}
```

SWITCH_TIMING_CHAR 1

Turns on switch timing characterization. Syntax:

```
SWITCH_TIMING_CHAR [0 | 1]
```

EXTERNAL_CONTROL_PIN poweron_ads

Defines external control pin; required keyword for timing characterization. Syntax:

```
EXTERNAL_CONTROL_PIN <external control pin name>
```

INTERNAL_CONTROL_PIN poweron_ads_int

Defines internal control pin; required keyword for timing characterization. Syntax:

```
INTERNAL_CONTROL_PIN <internal control pin name>
```

CONTROL_SLEW 2 100p 200p

Defines transition times in ps (default, 100 ps). Syntax:

```
CONTROL_SLEW <number of values> <transition time 1>
[ <transition time 2> ... ]
```

INT_TIMING_ON2OFF POWERON_ADS F

```
4.18423e-11 4.64905e-11 6.99287e-11 5.5511e-11
```

Specifies internal pin timing for On-to-Off transitions. Syntax:

```
INT_TIMING_ON2OFF <external_ctrl_pin> [R | F]
< In>Out delay in S for transition 1 >
<internal slew in S for transition time 1>
[< In>Out delay in S for transition 2 >
<internal slew in S for transition time 2>... ]
```

where [R | F] specify the <external_ctrl_pin> waveform is rising or falling to make the switch OFF2ON or ON2OFF.

INT_TIMING_OFF2ON POWERON_ADS R

```
4.16594e-11 4.44381e-11 7.21476e-11 5.44803e-11
```

Specifies internal pin timing for Off-to-On transitions. Syntax:

```
INT_TIMING_OFF2ON <external_ctrl_pin> [R | F]
<in>out delay in S for transition time 1>
<internal slew in S for transition time 1>
[<in>out delay in S for transition time 2>
<internal slew in S for transition time 2> ... ]
```

Advanced Switch Characterization

Note: if all switch enable waveforms are relatively fast and linear, with transition times on the order of a few picoseconds, the advanced switch characterization described in this section is *not* required.

For switches that have multiple control pins, significant transition times (for example, a single driver might be controlling a group of switches), or non-linear behavior, using the 'MD_PWL 1' keyword setting in the switch configuration file is recommended.

Using the keyword SPLIT_MD_PWL [0 | 1] controls whether simulation runs serially or in parallel (keyword "MD_PWL 1" must be set). The default is "0", which means the simulations run in serial. If it is set to "1", the simulations run in parallel, so characterization is faster.

The behavior of non-linear or slow transition control pin waveforms can be specified in the GSR using the keyword 'PIECEWISE_SWITCH_INPUT <filename>', which specifies a switch input file that describes the input waveforms for the required switch pins in piecewise linear detail. The format of the switch input file is as follows:

```
<switch_inst_name> <ctrl_pin_name> ( <t1> <volt1> <t2> <volt2> ... )
```

```
...
```

where times are in seconds and voltages are in volts.

Switch Model Generation

Once a configuration file is created for each switch, list the location of all configuration files in another master configuration file, one configuration file per line, along with the name of the switch cell (should be the same as that specified in the configuration file and as specified in the Spice netlist). For example, a design with two switches, switch_ab and switch_cd needs two configuration files, such as *switch_ab.conf* and *switch_cd.conf*. The location of the individual switch configuration files must be listed in a master configuration list file, with the following internal format:

```
switch_ab <path to config file ab>/switch_ab.conf
switch_cd <path to config file cd>/switch_cd.conf
```

The APL utility, *aplsw*, is then executed on the master switch configuration list file to generate the switch models used in low power analysis, using the syntax:

```
aplsw [-c] [-d] [-o <output_file>] <sw_config_list.conf>
```

where

-c : check generated result

-d : debug mode

-o <output_file> : output file name

<sw_config_list.conf> : switch configuration list file

Non-ideal piecewise linear control pin inputs can be accommodated in the current modeling.

After `ap1sw` completes execution, a switch model file, called `ap1sw.out` by default, is automatically generated, containing data for each switch model in the APL run directory. The `ap1sw.out` file contains a header to identify the nature of the contents, which includes the pin connections and the electrical parameters for each of the four switch states, as well as the voltage and leakage current relationships for a specified number of data points for each condition. Leakage current outputs are provided by 2D or 3D table lookup for up to two control pin inputs, as well as Vdd, so if piecewise linear control pin voltages are available, this additional characterization accuracy can be used.

Note: Users should not attempt to edit the switch model files.

Defining Block Switching Status with the GSC File

The state of instances in a block for low power analysis must be defined in the GSC file. You have three options to specify the state in low power analysis:

1. Specify the state of BOTH switch instance and power-gated block-instance:

```
<switch_instance> POWERUP  
<gated_block> POWERUP
```

2. Specify domain_name to include everything in that domain:

```
* <domain_name> POWERUP
```

3. Combine 1 and 2 for more specifically defined case:

```
<switch_instance> <domain_name> POWERUP  
<gated_block> <domain_name> POWERUP
```

Wildcards (*) can be used for defining sets of blocks and instances.

For information setting states for switches by using the GSC file, see the section "[Global Switching Configuration \(GSC\) File](#)", page C-571.

Obtaining STA Timing Window Data

Information about the time-dependent behavior of a block is obtained from the timing window data in the STA file. See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#) for details of how to generate the STA output file, and how to get STA file data for switch enabled pins. The following three special PT variables are used during STA file generation:

- ADS_CELLS_NEED_INPUT_TW
- ADS_INPUT_PINS_NEED_TW
- ADS_PINS_NO_TW

Importing Switches into RedHawk

Use the following GSR entry to import the switch models generated by the `ap1sw` utility into [RedHawk](#):

```
SWITCH_MODEL_FILE {  
    <switch_model_file>  
}
```

You must ensure that all external power and ground networks that are connected through the switches and need to be simulated by [RedHawk](#) are listed in VDD/GND_NETS, along with their associated voltage values. As long as the EXTRACT_INTERNAL_NET GSR keyword is set to 1 (default), internal nets do not have to be listed. Following is an example of the referenced GSR keywords.

```
VDD_NETS {
```

```

        VDD_EXT 1.08
    }
GND_NETS {
    VSS_EXT 0
}

```

For proper consideration of switch models in RedHawk analysis, the following data requirements must be satisfied:

1. Define each switch in one of the LEF files, using the same name as in the switch model file.
2. Instantiate the switch in one or more of the DEF files.
3. Define the switches used to switch a block in the DEF file for that block.
4. Define the INT_PIN of the switch in LEF file with direction OUTPUT or INOUT.
5. Define the EXT_PIN of the switch in LEF file with direction INPUT or INOUT.
6. Timing window information for switch controls from the STA file.

Once the switch models are read in and the database is set up, connections are made between the external power and/or ground networks through the header and/or footer switches, respectively, using the electrical parameters defined in the switch models.

Note that a new switch model file with different switch models can be substituted during a RedHawk run using the TCL command

```
gsr set SWITCH_MODEL_FILE <switch_model_filename>
```

which overrides the existing switch model file. Then design extraction and analysis can be easily rerun with different switch models specified in the new file.

Adding and Deleting Power Switches

Power gating switches can be added and deleted using the TCL commands 'eco add switch' and 'eco delete switch'. See [section "eco", page D-769](#), for details on command usage.

Reporting on Switches in the Design

RedHawk reports the switch names traced in a separate file, *adsRpt/apache.memsw.info*. The *redhawk.log* file points to the file *adsRpt/apache.memsw.info* containing the names of the macros containing switches, the switch master names, the switch count, and the switch internal domains.

Design Characterization for Power-up Conditions

Accurate power-up simulation using header or footer switches requires characterization of the cells in the design to get the piecewise linear models for intrinsic capacitances, leakage currents, and effective series resistance, as a function of voltage.

The APL utility, *aplди -w -s2*, is used to characterize all cells and decap present in the design to generate the required PWL model. The UNIX shell command syntax is:

```
% aplди -w -s2 [-l <cell_list_file>] <apl_config_file>
and
```

```
% aplди -w -s2 [-p <decap_list_file>] <apl_config_file>
Use the same configuration file as for other APL characterizations.
```

The output file from the above characterization can be included for a subsequent RedHawk analysis by using the GSR keyword:

```
PIECEWISE_CAP_FILE <name_of_PWL_cap_file>
```

Following the generation and importation of the switch models and the design database files, RedHawk static and dynamic power analysis can be performed.

Running RedHawk Low Power Analysis

ON State Analysis

ON state analysis is run the same as for normal RedHawk power analysis except for two additional steps:

1. The switch models must be defined using the GSR keyword SWITCH_MODEL_FILE.
2. External and internal P/G nets associated with header and footer switches must be defined using GSR keywords VDD_NETS and GND_NETS.

Ramp-up Analysis

For ramp-up analysis, the following steps must be performed in addition to normal power analysis steps.

1. The switch models must be defined using the GSR keyword SWITCH_MODEL_FILE.
2. External and internal P/G nets associated with header and footer switches must be defined using GSR keywords VDD_NETS and GND_NETS.
3. Instances that are powering up must be defined in the GSC file. See [section "Global Switching Configuration \(GSC\) File", page C-571](#).
4. The piecewise linear capacitance file must be defined using either the GSR keyword

```
PIECEWISE_CAP_FILE <filename>
```

or

```
API_FILES {  
    <filename> pwcapi  
}
```

If there is no PWL capacitance file, use the GSR keyword RAMPUP_OFFSTATE_VOLTAGE to define initial voltage conditions.

5. The STA timing file must specify the timing window for the control pins of the switch instances.

Sample Command File for Ramp-up Analysis

For rush current simulations, the following command sequence is recommended:

```
setup analysis_mode lowpower  
import gsr <filename>  
setup design  
perform pwrcalc
```

Note: the 'setup analysis_mode lowpower' command uses certain optimizations that enable faster run-times specifically for low power analysis.

```
perform extraction -power -ground -c
```

```
perform analysis -lowpower
```

Running in Mixed Mode

Low power analysis can be performed in mixed mode, with both VCD and Vectorless inputs. To invoke mixed mode for low power analysis, use the TCL commands 'setup analysis_mode lowpower' and also 'perform analysis -dynamic'. A sample TCL file for mixed mode is shown below:

```
setup design GENERIC.gsr
perform pwrcalc
setup analysis_mode lowpower
perform extraction -power -ground -c
setup package
setup wirebond
setup pad
perform analysis -dynamic
l
```

Power Gating Results

switch_static.rpt File

Following RH static analysis, the *adsRpt/Static/switch_static.rpt* file is generated. This lists the voltage at the internal node of all the switches, the voltage drop across them, and the current through them, as seen in the static analysis. The report has the following format:

```
#6.1.RD (Oct 31 10:09:57 2006)
#Report static results of switch voltage and current (milli-amperes)
#instance_name    type   internal_node_volt   volt_across_switch   avg_current
SW/FSW10          footer  1.946112e-05  1.515508e-05  2.117815e-04
SW/FSW20          footer  1.857796e-05  1.482933e-05  2.072293e-04
SW/FSW30          footer  1.940953e-05  1.542746e-05  2.155878e-04
SW/FSW40          footer  1.888943e-05  1.490737e-05  2.083198e-04
SW/FSW50          footer  1.862162e-05  1.502620e-05  2.099804e-04
```

switch_dynamic.rpt File

Following RH dynamic analysis, the *adsRpt/Dynamic/switch_dynamic.rpt* file is generated, which lists the peak current through active header and footer switches for ramp-up analysis (in Amps). The report has the following format:

```
#Report dynamic results of switch voltage and current
#instance_name      type   maximal_Isw(Amp)
SW/FSW1            footer  OFF
SW/FSW2            footer  3.139074e-03
SW/FSW9            footer  OFF
SW/FSW3            footer  OFF
SW/FSW2            footer  3.137452e-03
SW/FSW7            footer  OFF
```

charge_switch.rpt File

For designs using charge switches, which insure that a correct threshold voltage is reached before functional switch activation, *adsRpt/Dynamic/charge_switch.rpt* is created, to report on charge switch and functional switch performance, with a format as follows:

```
#Initial Function Switch Rampup Time (IFSRT): 10000
***** Charge Switch Report *****
-----
[instance_name] [threshold] [voltage@IFSRT] [time_reach_SRT]
[violation]
-----
inst_CSW_VDD      0.7  0.293019  12970  Y
***** Functional Switch Report *****
#Time of last charge switch reaching SRT: 12970
-----
[instance_name] [rampup_time] [violation]
-----
fsw_inst_VDD-1    15000 N
fsw_inst_VDD-2    15000 N
fsw_inst_VDD-3    15000 N
fsw_inst_VDD-4    15000 N
fsw_inst_VDD-5    15000 N
fsw_inst_VDD-6    15000 N
...

```

where

Initial Function Switch Rampup Time (IFSRT): turn-on time for the switch (ps)

Charge Switch Report section:

instance_name: the instance name for charge switches

threshold: required threshold voltage to be provided by the charge switch (V)

voltage@IFSRT : the internal charge switch node voltage at rampup time

time_reach_SRT: time for switch to achieve rampup threshold voltage

violation: "Y" indicates charge switch rampup time that exceeds specification

Functional Switch Report section:

instance_name: the instance names for the functional switches

rampup time: max time allowed for functional switch to meet specification

violation: "Y" indicates functional switch rampup time that exceeds specification

The GSR keyword CHARGE_SWITCH specifies the names of the charge switches used in the design and their threshold voltages.

Analysis of IP Block Designs with Switched Power

Introduction

A diagram representing the elements of a typical macrocell power switching scheme is given in Figure 13-13.

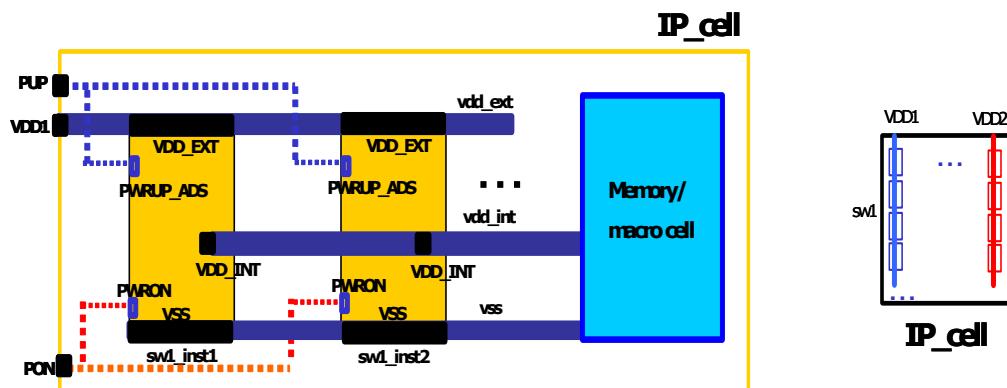


Figure 13-13 Power switching diagram for IP macrocell

Data Requirements

In order to perform an analysis of internally-switched IP blocks, the following data rules must be met:

- have hierarchical GDS with the switches defined as placed master cells.
- have SPICE for the memory with switch subckt names matching GDS cell names
- have external and internal (virtual) P/G name mapping between LEF/GDS view and SPICE view
- have top-level and cell-level switch control definitions (similar to current PowerGate) for characterization

Flow Overview

There are two different GDS2DEF flow steps, depending on whether the design has voltage domain text labels or not. If it does not have domain text labels, several extra steps are required, including two GDS2DEF runs. The Figure 13-14 and Figure 13-15 diagrams describes the two flows, which are only different in that if you have domain text

labels, only one GDS2DEF run is required, as shown in Figure 13-14. These flows are described in the following sections.

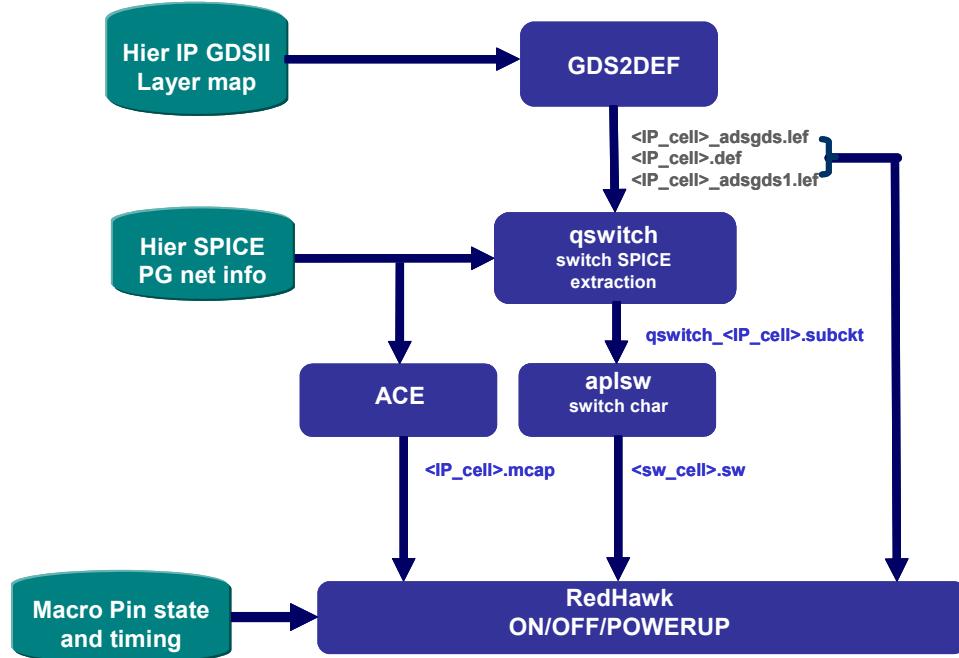


Figure 13-14 Flow for analysis of switched IP design, with text labels

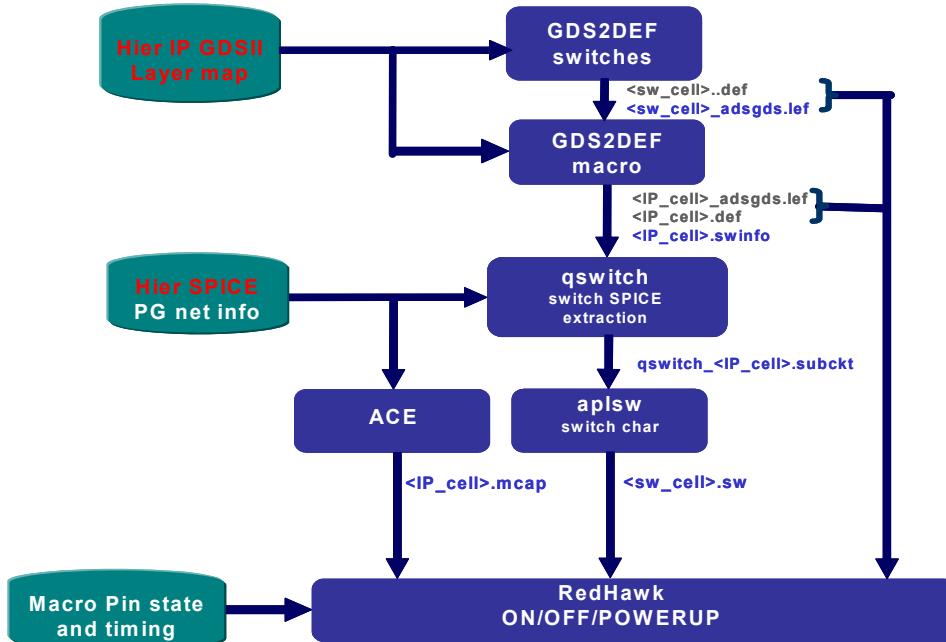


Figure 13-15 Flow for analysis of switched IP design, no text labels

GDS2DEF Processing

Domain Text Labels Available

1. Run a single *gds2def* run, using the following keywords in the configuration file:
 DEFINE_SWITCH_CELLS - see usage in [section "DEFINE_SWITCH_CELLS", page E-900](#)
 EXTRACT_SWITCH_CELLS - see usage in [section "EXTRACT_SWITCH_CELLS", page E-900](#)
2. The VDD_NETS and GND_NETS definition statements in the GDS2DEF configuration file must specify the names of all the nets (internal and external) that you plan to extract using the “text label GDS” flow.
The <IP_cell>_adsgds1.lef output file contains the LEF for all switches extracted.
3. Skip the following section, and go to [section "Switch Subcircuit Extraction", page 13-364](#).

No Domain Text Labels Available

If your design has no domain text labels, the remainder of this section describes the proper flow. Some important aspects of this flow are:

- GDS2DEF can extract the LEF for the switch automatically
- QSWITCH extracts the Spice sub-circuit of the switch
- APLSW characterize the switch model
- unique virtual domain device capacitance is created for the switched macro
- each power domain can have only one kind of master switch cell

The following sections describe the steps in the switch analysis.

LEF Extraction

If the LEF for the embedded switches is not available, then you can use *gds2def* to create a LEF model for use in the flow. The following example of a basic configuration file, and the embedded comments, explain the procedure.

```
TOP_CELL <switch cell> # ---> switch cell name in the GDS file
GDS_FILE <IP cell GDSII file>
GDS_MAP_FILE <layer map file>
VDD_NETS {
    # < list the texted nets connected to the switch; examples below >
    <VDD_EXT>
    <VDD_INT>
}
GND_NETS {
    <VSS>
}
OUTPUT_DIRECTORY .
GENERATE_LEF_PINS 1 # '_ADS' is added to each pin name
OUTPUT_APACHECELL 0 # '0' means to extract the LEF file
```

Extract the LEF file with the command

```
gds2def <config_file>
```

After running *gds2def*, the switch LEF file, *<cellname>_adsgds.lef*, is generated. Repeat this process for each unique switch master cell in the switched IP GDS file. A switch model is shown in Figure 13-16.

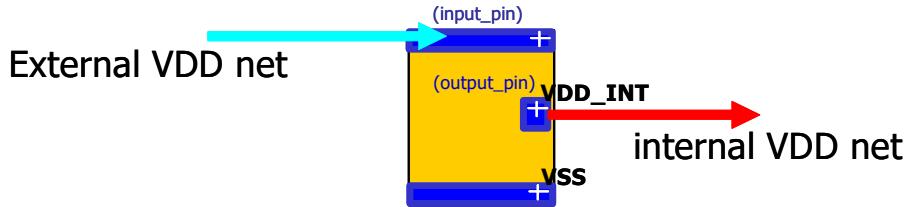


Figure 13-16 switch model

Creating the Switch Macro Model

The following four keywords added to the basic *gds2def* configuration file provide the information needed to build the switch IP block model:

- **LEF_FILES** - defines input block and switch LEF files provided by user or generated from previous step
- **SWITCH_CELLS** - maps cellnames to output_pin_names and input_pin_names, based on pin names in the switch LEF files, and generates a file called *<macro_name>.swinfo*, which is used in the follow-on step to extract the switch subcircuit for characterization.
- **VP_PAIRS** - maps internal_net_name to external_net_name
- **USE_LEF_PINS_FOR_TRACING** - when set to 1, uses geometries defined in the PINS section of the LEF file of the block being extracted to start tracing its power and ground nets during GDS translation. No text or label-based extraction is performed. However, user-specified coordinates are honored.

The syntax for these configuration keywords is as follows:

```

LEF_FILES {# input LEF files
    <IP LEF file>
    <switch_cell_1>_adsgds.lef
    <switch_cell_2>_adsgds.lef
    ...
}

USE_LEF_PINS_FOR_TRACING [ 1 | 0 ]

SWITCH_CELLS {# cellname output_pin_name input_pin_name
    <sw1 cellname> <VDD1_INT>_ADS <VDD1_EXT>_ADS
    <sw2 cellname> <VDD2_INT>_ADS <VDD2_EXT>_ADS
}

VP_PAIRS {# internal_net_name external_net_name
    <vdd1_int> <vdd1_ext>  <vdd2_int> <vdd2_ext>
}

```

An example of the steps to create the IP switch LEF file, <IP_cell>.swinfo, using the gds2def command is shown in Figure 13-17 below.

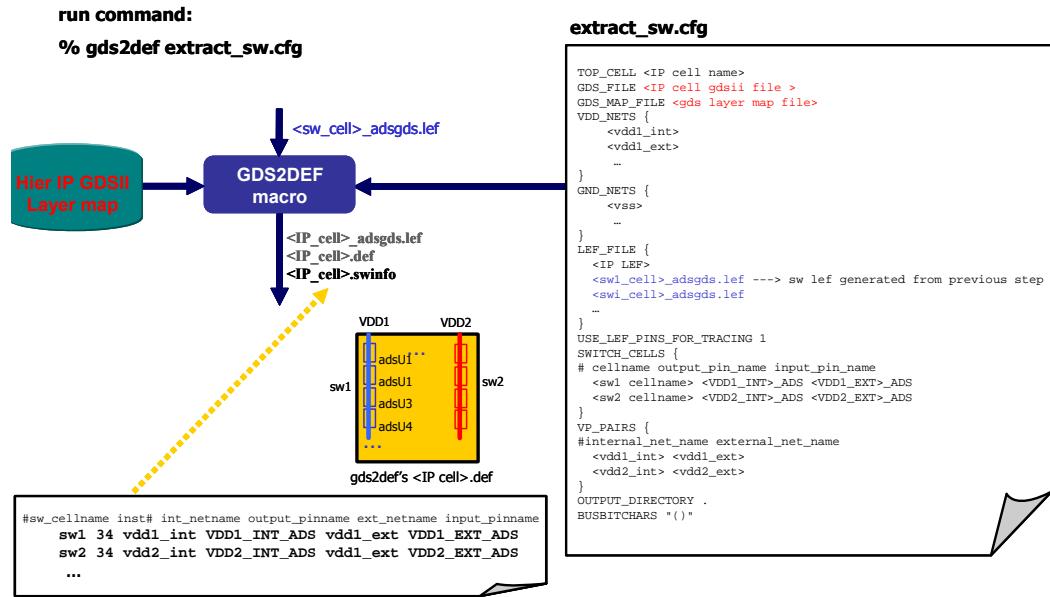


Figure 13-17 Creating the switch block LEF file

Switch Subcircuit Extraction

The switch SPICE subckts can be extracted from the top level IP SPICE netlist if you do not have the SPICE for each of the switches. A program called ‘qswitch’ uses a configuration file with the ‘vp_mapping’ keyword to extract the switch subcircuits. The ‘vp_mapping’ keyword describes the mapping between the GDS/LEF power domain and Spice netlist node name. The ‘switch_info’ keyword defines the file generated from the gds2def run. An example ‘qswitch’ configuration file follows:

```

# example configuration file for generating <IP_cell>.smin
vp_mapping <GDS/LEF_power_domain> <Spice_netlist_node_name>
vp_mapping <vdd1_int> <vdd1_int_sp>
vp_mapping VDDPR vddpr
vp_mapping VDDA vddar
vp_mapping VDDAR vddarchip
switch_info <IP_macro>.swinfo
subckt <IP_macro>.nsp
include <model files>
# if needed include SPICE options
option scale=le-6
...

```

To run *qswitch*, use the command

```
qswitch <IP_cell>.smin
```

After running *qswitch* the subcircuit file, <IP_macro>.subckt, for each switch is generated. With these subcircuit files, you can run the *aplsw* utility to generate the switch model file for RedHawk. A sample switch IP subckt file follows:

```
*-- qswitch extraction tool
```

```

*-- extracted switch subckt for cell 'sw1'
.SUBCKT sw1 VDD1_EXT_AMPS VDD1_INT_AMPS PWRUP_AMPS PWRON_AMPS M0
+ VDD1_EXT_AMPS POWERUP_AMPS VDD1_INT_AMPS VDD1_INT_AMPS PCH L=0.3W=0.35
+ M=1 M1 VDD_AMPS POWERUP_AMPS
...
*--- Switch Subckt for cell 'sw2'
.SUBCKT sw2 VDD2_INT_AMPS VDD2_EXT_AMPS PWRUP_AMPS PWRON_AMPS
+ M3 VDD2_EXT_AMPS PWRUP_AMPS VDD2_EXT_AMPS VDD2_EXT_AMPS PCH
+ L=0.085 W=0.2 M=1...
*-- user specified options
.option scale=1e-6
.inc ../../lib.model

```

IP Switch Characterization with the aplsw Utility

The APL utility *aplsw* is used to characterize the IP switches for RedHawk analysis. See section "Characterization and Implementation", page 13-350, for more details on syntax and an example *aplsw* configuration file.

An example of the *aplsw* output and the characterized model is shown in Figure 13-18.

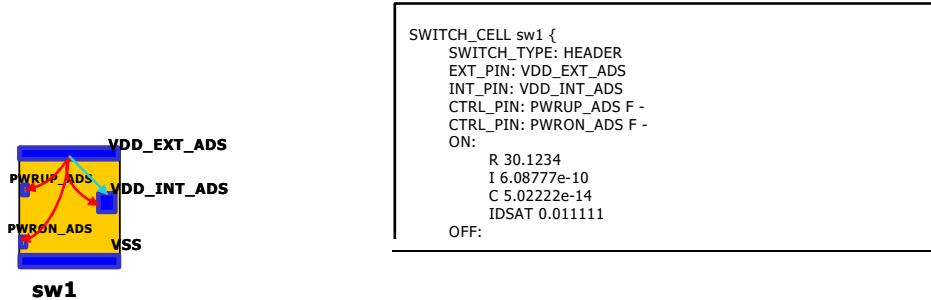


Figure 13-18 aplsw model of IP switch

ACE Characterization

The ACE utility provides equivalent power circuit resistance, device capacitance, and leakage current data. For details on running ACE, see section "ACE Decap and ESR Characterization", page 9-263.

Running RedHawk with Switch IP models

Running RedHawk with switched IP blocks is similar to other RedHawk analyses using *gds2def* modeled blocks. You should identify the *gds2def* models using the GSR keyword **GDS_FILE**. along with the following two additional keywords:

EXTRACT_INTERNAL_NET 1

When set to 1 this keyword means the user does not need to specify the internal nets in **VDD_NET_LIST**; RH automatically traces the internal power nets and extracts them. The Virtual Power control file must be included.

```
VP_CONTROL {
    <control_filename>
}
```

The VP_CONTROL keyword specifies a file that describes the controlling pin information for switch IP macros. See [section "VP_CONTROL", page C-726](#), for details on the contents, format and usage of the file.

Analysis of Switched RAM Designs

Introduction

Switched RAM designs are different than other types of power switching designs in that the switches are an integral part of the RAM blocks and therefore they must be extracted in a different manner. This section describes the methodology for handling various types of switched RAM designs for On-state static IR and DvD analyses and also for Ramp-up analysis.

Types of Switched RAM Supported

The following four types of switched RAM can be analyzed in RedHawk:

- Case 1 - single external/internal P/G net pair connected by a switch cell
- Case 2 - multiple external/internal P/G net pairs connected by an unique switch cell respectively
- Case 3 - single external P/G net connecting to multiple internal P/G nets by the same switch cell
- Case 4 - single external P/G net connecting to multiple internal P/G nets by the different switch cells

These types of supported switch arrangements are shown diagrammatically in Figure 13-19.

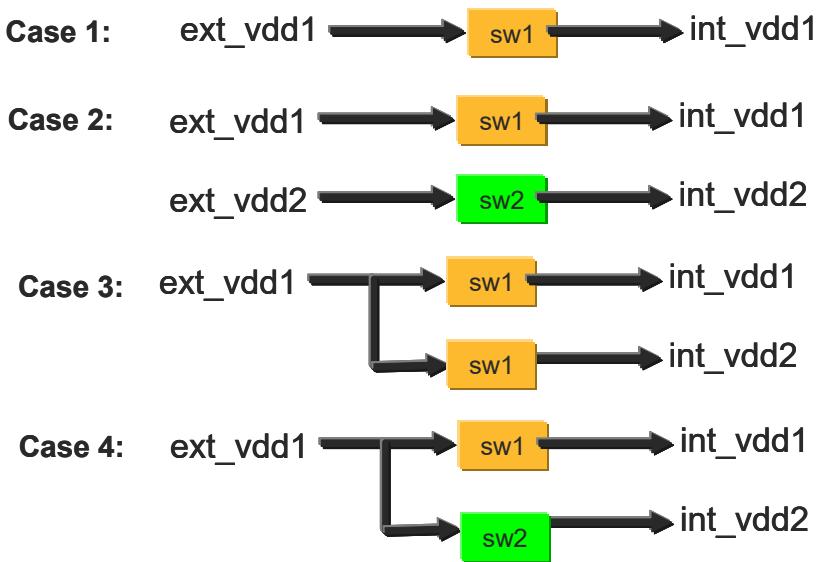


Figure 13-19 Supported switched RAM configurations

The following switched RAM configurations represent design violations and are not supported in RedHawk analysis:

- Cases 5 and 6 - multiple external P/G nets connecting to a single internal P/G net
- These types of unsupported switch arrangements are shown diagrammatically in Figure 13-20.



Figure 13-20 Unsupported switched RAM configurations

Overview of Switched RAM Analysis

Analysis of Static, Dynamic Voltage Drop (DVD) and Ramp-up conditions in switched RAM designs can be performed by RedHawk as needed. Overviews of the data flows for each of these types of analyses are shown on the following pages in Figure 13-21, Figure 13-22, and Figure 13-23.

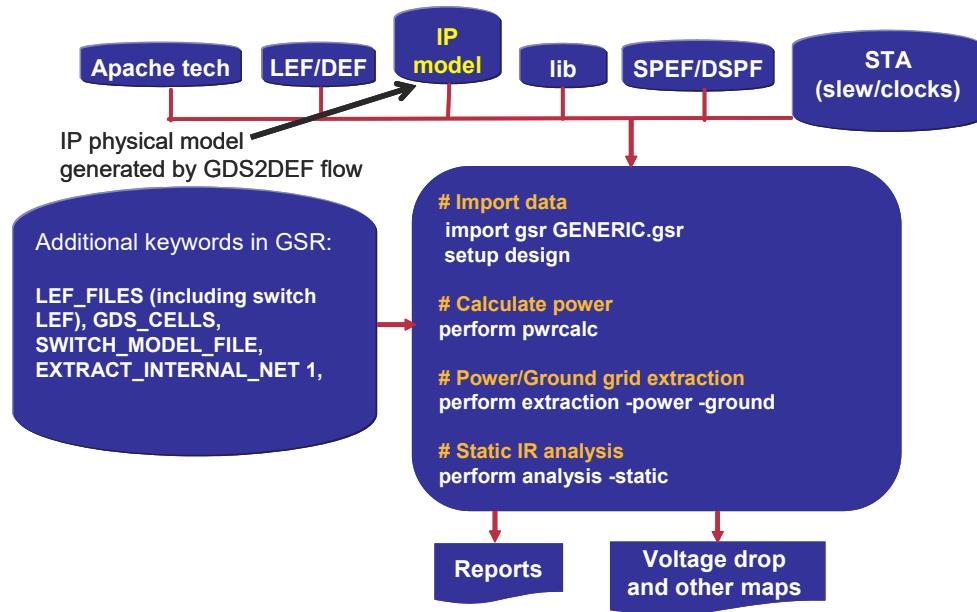


Figure 13-21 Static IR analysis in switched RAM flow

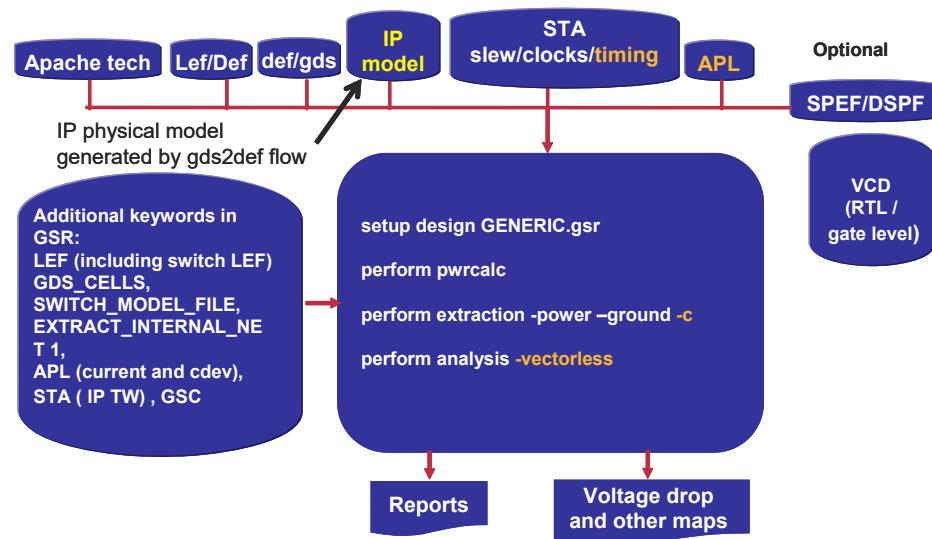


Figure 13-22 Dynamic DvD analysis in switched RAM flow

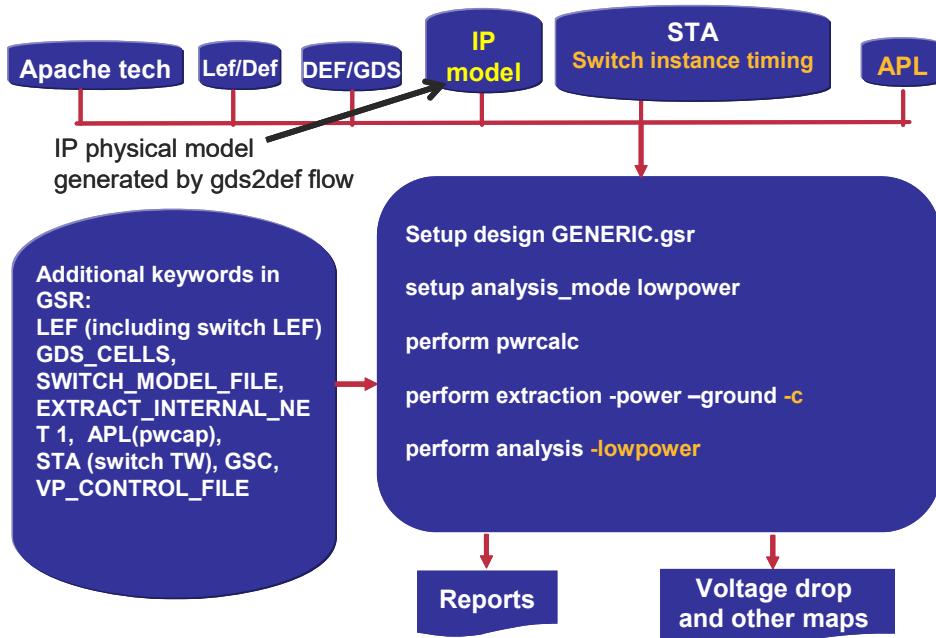


Figure 13-23 Ramp-up analysis in switched RAM flow

Model Generation

GDS Data Preparation

GDS2DEF flow supports two types of physical model generation, an On-state model and a Ramp-up model. The On-state model is used in static IR and dynamic analyses, while the ramp-up model is used for turn-on conditions. The GDS2DEF physical model flow is shown in Figure 13-24.

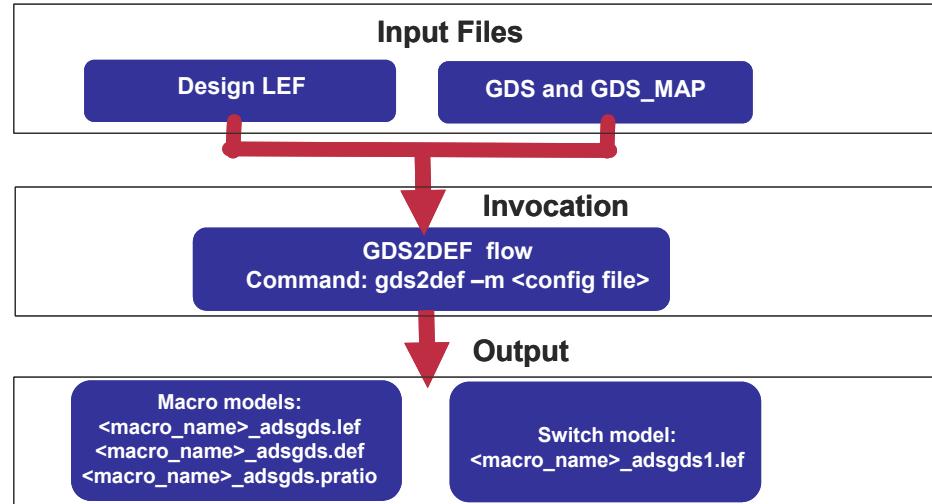


Figure 13-24 GDS2DEF model flow

On-State Model

A diagram of the On-state physical model is shown in Figure 13-25.

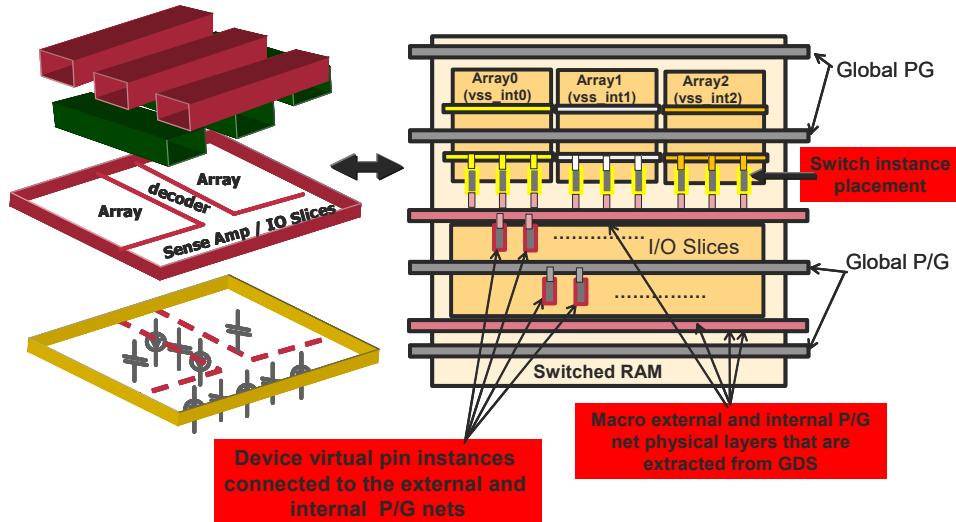


Figure 13-25 On-state physical model

The On-state model files include:

- the DEF file, which contains the switch instance placement data, all the internal and external P/G net physical layers, the P/G pin logical connectivity of the switch

- instances, and virtual pin instances
- the LEF file, which contains the device virtual pin instances that are connected to the internal and external P/G nets
- the Pratio file, which contains the relative strength data for device virtual pin instances

Ramp-up Model

A diagram of the ramp-up physical model is shown in Figure 13-26.

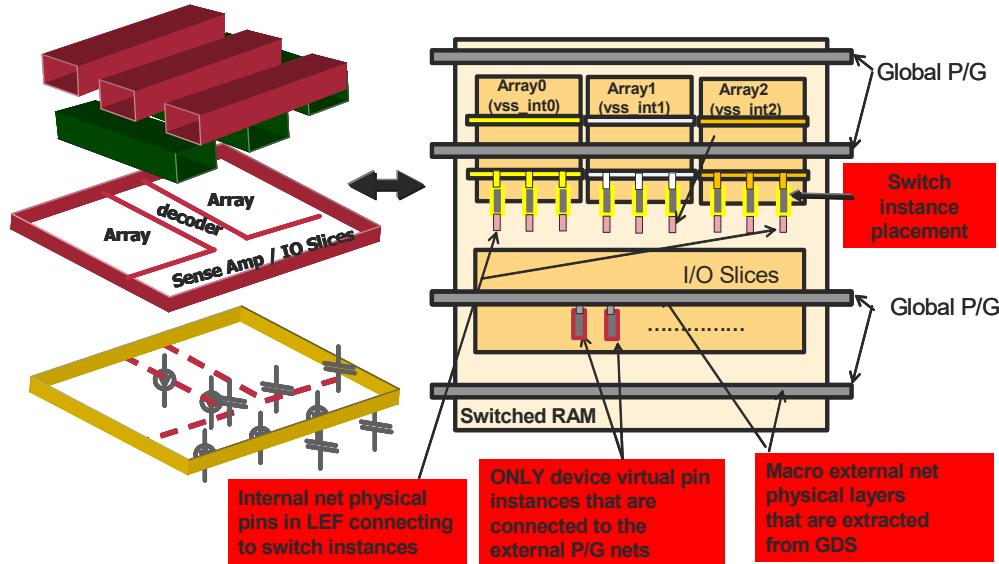


Figure 13-26 Ramp-up physical model

The ramp-up model has reduced output data (compared to the On-state model), for better ramp-up flow performance. It can also be used in the On-state flow.

The ramp-up model files include:

- the DEF file, which contains the switch instance placement data, *only* the external P/G net physical layers, the P/G pin logical connectivity (applied to the switch instances and external domain virtual pin instances, but NOT the internal domain pin instances)
- the LEF file, which contains *only* device virtual pins connected to external P/G nets and internal domain pins (which are connected to the switch instances)
- the Pratio file, which contains the relative strength data for the device virtual pin instances

General GDS2DEF Configuration File Setup

The 'EXTRACT_SWITCH_CELLS' and 'DEFINE_SWITCH_CELLS' configuration file keywords are required for ramp-up analysis, but optional for On-state mode. The keywords must be used together, with the following syntax:

```

EXTRACT_SWITCH_CELLS {
    <switch_name> [HEADER | FOOTER] <extern_net_name>
    <intern_net_name>
    ...
}
DEFINE_SWITCH_CELLS {
    <switch_name> [HEADER | FOOTER] <input/ext_LEF_pin_name>
    <output/int_LEF_pin_name>

```

```
<control pin 1> <control pin 2> ... <control pin N>
}
```

RedHawk supports multiple domain switching by the same switch cell (case 3), as follows:

```
EXTRACT_SWITCH_CELLS {
    sw1 HEADER ext_vdd1 int_vdd1
    sw1 HEADER ext_vdd2 int_vdd2
}
```

To set up for ramp-up model analysis, you must specify internal P/G domains using the keywords VDD_NETS and GND_NETS in all cases, and also set the keyword 'SWITCH_MODEL_MODE RAMPUP'.

To set up for the On state modeling, you do not need to specify internal nets in VDD/GND_NETS , EXTRACT_SWITCH_CELLS and DEFINE_SWITCH_CELLS, unless you want to perform internal domain simulation. If the EXTRACT_SWITCH_CELLS and DEFINE_SWITCH_CELLS keywords are specified, the internal nets must be specified in VDD/GND_NETS. Also insure that 'SWITCH_MODEL_MODE ONSTATE' is set (the default value).

A sample GDS2DEF configuration file for the general case is shown below:

```
TOP_CELL <design_name>
GDS_FILE <gds_pointer>
GDS_MAP_FILE <layer map_pointer>
VDD_NETS {
    <External_Power_net_group_name> {
        VDD:
        VDD_ext
    }
    <Internal_Power_domain_name> {
        Vint:
        Vint @ <layer_id> <x_coord_um> <y_coord_um>
    }
}

GND_NETS {
    <Ground_net_group_name> {
        VSS
    }
}
LEF_FILE {
    <design_LEF_file_pointer>
}
USE_LEF_PINS_FOR_TRACING 1
EXTRACT_SWITCH_CELLS {
    <switch name> <HEADER | FOOTER>
    <external net name> <internal net name>
}
DEFINE_SWITCH_CELLS {
    <switch_name> <HEADER|FOOTER> <ext_LEF_pin>
    <int_LEF_pin> <control_pin1> ... <control_pinN>
}
CHECK_TRACING 1
SWITCH_MODEL_MODE [RAMPUP | ONSTATE]
```

Note that ‘Vint @ <layer_id> <x_coord_um> <y_coord_um>’ syntax is used for P/G net tracing using location, and ‘<Ground net group name> {VSS ...}’ syntax is used for P/G net tracing using text labels.

Remove the keyword ‘USE_LEF_PINS_FOR_TRACING 1’ if the Text tracing method is used. The keyword ‘DEFINE_SWITCH_CELLS’ is required for switch LEF generation.

Note: the control pin name must match the one in the APL switch model.

Case 1 - GDS2DEF Configuration File

Single external/internal P/G net pair connected by a switch cell:



Case 1 configuration switches should have a GDS2DEF configuration file as follows:

```

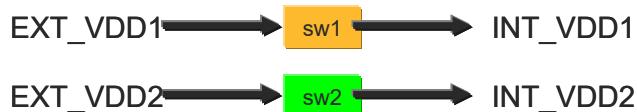
TOP_CELL <design_name>
GDS_FILE <gds_pointer>
GDS_MAP_FILE <layer_map_pointer>
VDD_NETS {
    Ext_VDD1
    Int_VDD1
}
GND_NETS {
    VSS
}
EXTRACT_SWITCH_CELLS {
    sw1 HEADER EXT_VDD1 INT_VDD1
}
DEFINE_SWITCH_CELLS {
    sw1 HEADER VDD_EXT VDD_INT EN1
}
# For Net tracing using LEF pins
LEF_FILE {
    <design_LEF_file_pointer>
}
USE_LEF_PINS_FOR_TRACING 1
SWITCH_MODEL_MODE [RAMPUP | ONSTATE]

```

For case 1 switches, configuration file specifications for EXT_VDD1 and INT_VDD1 are *optional* for ‘SWITCH_MODEL_MODE ONSTATE’ and *mandatory* for ‘SWITCH_MODEL_MODE RAMPUP’.

Case 2 - GDS2DEF Configuration File

Multiple external/internal PG net pairs connected by an unique switch cell respectively:



Case 2 configuration switches should have a GDS2DEF configuration file as follows:

```
TOP_CELL <design_name>
```

```

GDS_FILE <gds_pointer>
GDS_MAP_FILE <layer_map_pointer>
VDD_NETS {
    Ext_VDD1
    Ext_VDD2
    INT_VDD1
    INT_VDD2
}
GND_NETS {
    VSS
}
EXTRACT_SWITCH_CELLS {
    sw1 HEADER EXT_VDD1 INT_VDD1
    sw2 HEADER EXT_VDD2 INT_VDD2
}
DEFINE_SWITCH_CELLS {
    sw1 HEADER VDD_EXT VDD_INT EN
    sw2 HEADER VDD_EXT VDD_INT EN
}
# For Net tracing using LEF pins
LEF_FILE {
    <design LEF file pointer>
}
USE_LEF_PINS_FOR_TRACING 1
SWITCH_MODEL_MODE [RAMPUP | ONSTATE]

```

For case 2 switches, configuration file specifications for sw1 ... swN pins EXT_VDDn and INT_VDDn are *optional* for ‘SWITCH_MODEL_MODE ONSTATE’ and *mandatory* for ‘SWITCH_MODEL_MODE RAMPUP’.

Case 3 - GDS2DEF Configuration File

Single external P/G net connecting to multiple internal P/G nets by the same switch cell:



Case 3 configuration switches should have a GDS2DEF configuration file as follows:

```

TOP_CELL <design_name>
GDS_FILE <gds_pointer>
GDS_MAP_FILE <layer_map_pointer>
VDD_NETS {
    Ext_VDD1
    Ext_VDD1
    Int_VDD1
    Int_VDD2
}
GND_NETS {
    VSS
}

```

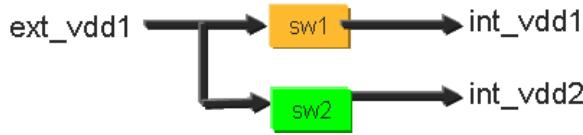
```

EXTRACT_SWITCH_CELLS {
    sw1 HEADER EXT_VDD1 INT_VDD1 EN1
    sw1 HEADER EXT_VDD1 INT_VDD2 EN1
}
DEFINE_SWITCH_CELLS {
    sw1 HEADER VDD_EXT VDD_INT EN
}
    
```

For case 3 switches, configuration file specifications for sw1 pins EXT_VDDn and INT_VDDn are *optional* for ‘SWITCH_MODEL_MODE ONSTATE’ and *mandatory* for ‘SWITCH_MODEL_MODE RAMPUP’.

Case 4 - GDS2DEF Configuration File

Single external P/G net connecting to multiple internal P/G nets with different switch cells:



Case 4 configuration switches should have a GDS2DEF configuration file as follows:

```

TOP_CELL < design name >
GDS_FILE < gds pointer >
GDS_MAP_FILE < layer map pointer>
VDD_NETS {
    Ext_VDD1
    Int_VDD1
    Int_VDD2
}
GND_NETS {
    VSS
}
EXTRACT_SWITCH_CELLS {
    sw1 HEADER EXT_VDD1 INT_VDD1 EN1
    sw2 HEADER EXT_VDD1 INT_VDD2 EN2
}
DEFINE_SWITCH_CELLS {
    sw1 HEADER VDD_EXT VDD_INT EN
    sw2 HEADER VDD_EXT VDD_INT EN
}
# For Net tracing using LEF pins
LEF_FILE {
    <design LEF file pointer>
}
USE_LEF_PINS_FOR_TRACING 1
SWITCH_MODEL_MODE [RAMPUP|ONSTATE]
    
```

For case 4 switches, configuration file specifications for sw1 ... swN pins EXT_VDDn and INT_VDDn are *optional* for ‘SWITCH_MODEL_MODE ONSTATE’ and *mandatory* for ‘SWITCH_MODEL_MODE RAMPUP’.

APLSW Data Preparation

To prepare the APLSW data perform the following steps.

- Check the switch cell name and switch pin name consistency in the APL switch models generated from the Spice netlist and the physical models from GDS.
- To ensure switch pin name matching, use the SPICE2LEF_PIN_MAPPING keyword for switch model characterization.
- Set up APLSW configuration file keywords. See an example set of keywords in section "Switch Configuration Files", page 13-350, and detailed descriptions of each keyword in section "APL Configuration File Description", page 9-218.

On-state Analysis - Static IR and DvD Conditions

A diagram of the analysis for Static IR and DvD conditions are shown in Figure 13-27.

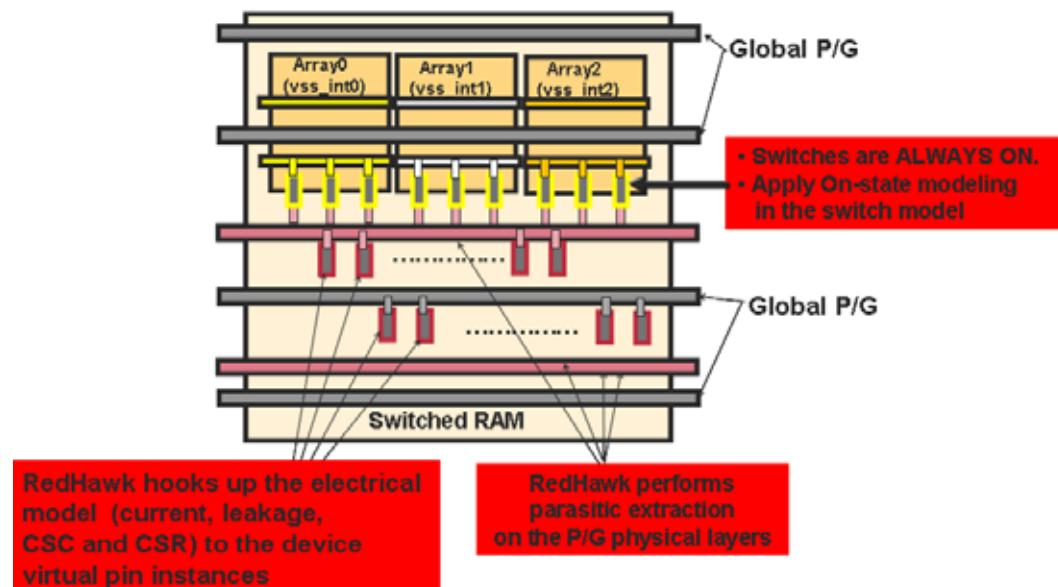


Figure 13-27 On-state analysis scheme

The following are key elements of switched RAM On-state analysis:

- Supports standalone macro on-state analysis.
- Switched cell APL model must be specified.
- Only the external nets must be specified with VDD_NETS or GND_NETS keywords.
- By setting EXTRACT_INTERNAL_NET 1, flow automatically traces the internal nets.
- Optional power ON control using VP_CONTROL keyword to assign internal delay to the block switching time.
- Results checking:
 - Make sure the power is successfully calculated.
 - Check the connectivity of the P/G pin instances.
 - Perform 'plot current –net –name <internal net>' command to make sure appropriate current is assigned.

GSR Keyword Settings - Static IR Analysis

The following are the key GSR keywords that must be defined for On-state switched RAM analysis for Static IR conditions. See the descriptions in Appendix C for more details on their syntax.

- TECH_FILE <tech file>
Defines the tech file.
- VDD_NETS {
 <VDD net name> <VDD voltage>
}
Defines Vdd nets and voltages-- internal nets should not be included.
- GND_NETS {
 <VSS net name> <VSS voltage>
}
Defines Ground nets and voltages-- internal nets should not be included.
- BLOCK_POWER_FOR_SCALING {
 <Block_power data>
}
Defines block power data. See Appendix C description for details.
- LIB_FILES {
 <lib file names>
 ...
}
Defines LIB files.
- LEF_FILES {
 <user IP LEF file>
 <gds2def path>/<macroname>_adsgds1.lef
}
Defines switch LEF file generated in gds2def flow.
- DEF_FILES {
 <DEF file>
 ...
}
Defines LEF files.
- GDS_CELLS {
 <cellname> <gds2def path for the on-state physical model>
 ...
}
Defines the GDS cells included in the analysis.
- FREQ <frequency>
Defines the operating frequency.
- SWITCH_MODEL_FILE {
 <APL switch model file >

```
        }
    • PAD_FILES {
        <pad files>
        ...
    }
    Defines pad files.
• EXTRACT_INTERNAL_NET 1
    Sets internal net extraction.
```

GSR Keyword Settings - DvD Analysis

The following are the key GSR keywords that must be defined for On-state switched RAM analysis for Dynamic Voltage Drop conditions. See the descriptions in Appendix C for more details on their syntax.

- TECH_FILE <tech file >
Defines the tech file.
- VDD_NETS {
 <VDD net name> <VDD voltage>
}
Defines Vdd nets and voltages-- internal nets should not be included.
- GND_NETS {
 <VSS net name> <VSS voltage>
}
Defines Ground nets and voltages-- internal nets should not be included.
- APL_FILES {
 <current_model> current
 <cap_model> cap
 ...
}
Defines APL current and capacitance data. See Appendix C description for details.
- LIB_FILES {
 <lib file names>
 ...
}
Defines LIB files.
- LEF_FILES {
 <user IP LEF file>
 <gds2def path>/<macroname>_adsgds1.lef
}
Defines switch LEF file generated in gds2def flow.
- DEF_FILES {
 <DEF file>
 ...

```
    }
    Defines DEF files.

• GDS_CELLS {
    <cellname> <gds2def path for the on-state physical model>
    ...
}
Defines the GDS cells included in the analysis.

• FREQ <frequency>
    Defines the operating frequency.

• SWITCH_MODEL_FILE {
    <APL switch model file >
}

• DYNAMIC_SIMULATION_TIME <sec>
    Defines dynamic simulation time.

• PAD_FILES {
    <pad files>
    ...
}
Defines pad files.

• EXTRACT_INTERNAL_NET 1
    Sets internal net extraction.

• GSC_FILE <GSC file>
    Defines GSC filename. GSC content format:
        <instance name> [ HIGH | LOW ]
    Example: mem_inst HIGH

• STA_FILE {
    <design name> <timing file>
}
Defines STA timing data file.

• POWER_MODE APL
    Defines APL power mode.

• VP_CONTROL {
    <Switch timing control file>
}
Defines optional control file.
```

Ramp-up Analysis

A diagram of the analysis for Ramp-up conditions are shown in Figure 13-28.

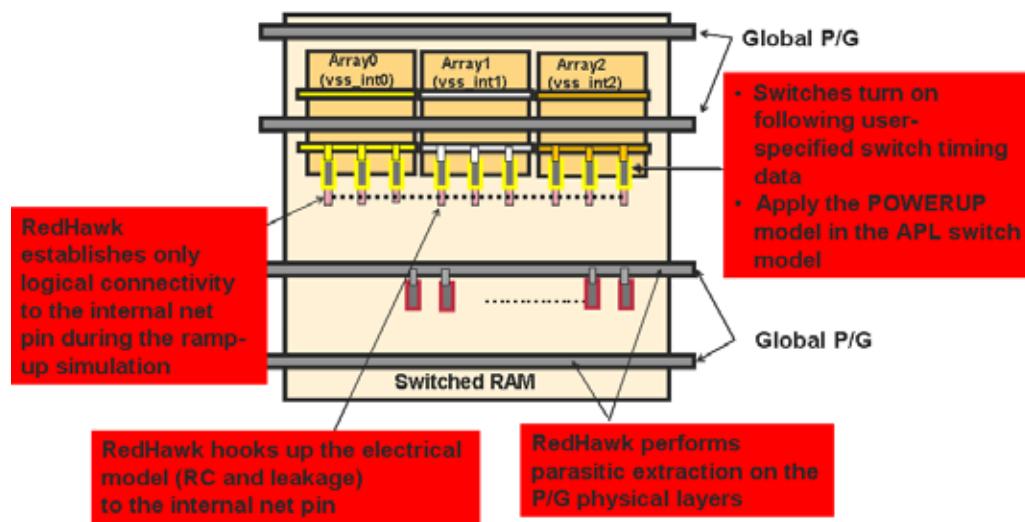


Figure 13-28 Ramp-up analysis scheme

The following are key elements of switched RAM ramp-up analysis:

- The VP_CONTROL keyword is a mandatory setting in the GSR file, which is used to specify the internal delay of the switch control pin inside the memory block. This delay is added to the timing value from the STA file to form the final switch TW.
- For multiple-domain ramp-up flow, only the device capacitance (cdev) APL model is supported. You can use RAMPUP_OFFSETSTATE_VOLTAGE keyword to specify the initial off-state voltage in the ramp-up simulation. The default value is 1/3 of nominal voltage specified in VDD_NETS.
- For single-domain ramp-up flow, you can use either the cdev or pwcap model. Use the aplcdev2pw utility to convert cdev to pwcap.
- Only the external nets must be specified in the VDD_NETS or GND_NETS keywords.
- By setting EXTRACT_INTERNAL_NET 1, the flow automatically traces the internal nets.
- The switch cell APL model must be specified.
- Capacitance is connected to the switch instances, which are logically connected in the ramp-up simulation.
- Does not support standalone macro ramp-up flow. In order to run macro analysis, a dummy top DEF must be created to instantiate the macro.
- Supports automated daisy-chain delay assignment using the VP_CONTROL keyword
- After the proper GSR keywords are set (see the following section), ramp-up analysis is invoked with the following command:

```
-perform analysis -lowpower -alp3d"
```

The -alp3d option provides approximately 3X overall speed-up for ramp-up analysis, with an accuracy impact of only 10-15%. Note that the time step must be <= 150 ps.

GSR Keyword Settings - Ramp-up Analysis

The following are the key GSR keywords that must be defined for Ramp-up switched RAM analysis. See the descriptions in Appendix C for more details on their syntax.

- TECH_FILE <tech file>
Defines the tech file.
- VDD_NETS {
 <VDD net name> <VDD voltage>
}
Defines Vdd nets and voltages-- internal nets should not be included.
- GND_NETS {
 <VSS net name> <VSS voltage>
}
Defines Ground nets and voltages-- internal nets should not be included.
- APL_FILES {
 <pwcap APL model> pwcap
 ...
}
Defines APL pwcap or cdev model data.
- LIB_FILES {
 <lib file names>
 ...
}
Defines LIB files.
- LEF_FILES {
 <user IP LEF file>
 <gds2def path>/<macroname>_adsgds1.lef
}
Defines switch LEF file generated in gds2def flow.
- DEF_FILES {
 <DEF file>
 ...
}
Defines DEF files.
- GDS_CELLS {
 <cellname> <gds2def path for the on-state physical model>
 ...
}
Defines the GDS cells included in the analysis.
- FREQ <frequency>
Defines the operating frequency.
- SWITCH_MODEL_FILE {
 <APL switch model file >
}
- DYNAMIC_SIMULATION_TIME <sec>

Defines dynamic simulation time.

- PAD_FILES {
 - <pad files>
 - ...
 - }

Defines pad files.

- EXTRACT_INTERNAL_NET 1
 - Sets internal net extraction.
- GSC_FILE <GSC file>
 - Defines GSC filename. GSC content format:
 - <instance name> <internal net> <POWERUP>
 - Example: mem_inst1 vdd_int POWERUP
- STA_FILE {
 - <design name> <timing file>
 - }

Defines STA timing data file.

- POWER_MODE APL
 - Defines APL power mode.
- VP_CONTROL {
 - <Switch timing control file>
 - }

Defines the optional control file. Refer to the next section for details.

Switched RAM Analysis Timing Control Settings

Switched RAM analysis provides as much flexibility as possible in controlling the turn-on sequence of memories that have power gating built-in. To do this RedHawk honors the memory ENABLE/Control pin timing window as a starting point. Based on an optional user-specified time, call it “delta”, turns on RedHawk-selected switches at TW, TW+1*delta, TW+2*delta, and so on. If delta=0 (no user-specified delta value), then all switches turn on at TW.

Typically during ‘gds2def’ translation, the DEF created does not have the relationship between the switches. So the turn-on order is not well controlled. So to solve this issue you can specify the switch turn-on delay (delta), either a global or an instance-specific (adsU<>) value that is created in the GDS2DEF flow to control the timing of the specific switches, as described below.

VP_CONTROL is a mandatory keyword for the ramp-up flow, to specify the timing of switch instances inside memory macros. VP_CONTROL contains MACRO-based models. There are two methods of specifying the timing, global and instance-specific.

Global Timing Assignment

For global timing assignment, you can set the POWERUP and POWERON (optional) keywords. The format of the global timing setting is shown below:

```
<macro name> {
  POWERUP <block ctrl pin1> <int domain1> <powerup_time_sec>
    ?<daisy delay_sec>?
  ...
  POWERUP <block ctrl pinN> <int domainN> <powerup_time_sec>
```

```
?<daisy delay_sec>?
POWERON <block ctrl pin1> <int domainN> <poweron_sec> (optional)
...
POWERON <block ctrl pinN> <int domainN> <poweron_sec> (optional)
}
```

For example:

```
Mem1 {
    POWERUP ctrl vdd_int 1e-9
}
```

In this example, all switch instances inside IP instances with master cell ‘mem’ switch at:
 $T=TW_of_ctrl+1ns$

The block control pin in the POWERUP setting is used in the ramp-up flow to extract the external macro switch timing from the STA timing file.

Instance-based Timing Assignment

For instance-based timing assignment the POWERUP, POWERON (optional) and SWITCH_RAMPUP_TIMING keywords must be used together. The format of the instance-specific timing specifications is shown below:

```
<macro name> {
    POWERUP <block ctrl pin1> <int domain1> <powerup_time_sec>
    ...
    POWERUP <block ctrl pinN> <int domainN> <powerup_time_sec>
    POWERON <block ctrl pin1> <int domain1> <poweron in sec>
    (optional)
    ...
    POWERON <block ctrl pinN> <int domainN> <poweron_sec>
    (optional)
    SWITCH_RAMPUP_TIMING <switch inst name adsU#>
        <powerup_time_sec>
        ?<poweron_time_sec>?
    SWITCH_RAMPUP_TIMING <switch inst name adsU#>
        <powerup_time_sec>
        ?<poweron_time in sec>?
}
```

The block control pin in the POWERUP setting is used in the ramp-up flow to extract the external-macro switch timing from the STA timing file.

For switch instances whose timing is not specified in SWITCH_RAMPUP_TIMING, the timing in POWERUP is used. For example, if there are three switch instances in the design, namely adsU[2-4]:

```
mem1 {
    POWERUP ctrl vdd_int 500e-12
    SWITCH_RAMPUP_TIMING adsU2 1e-9
}
```

Then adsU2 and adsU3 in mem1 will switch at:

$T=TW_of_ctrl+1ns$

The switch adsU4 in mem1 will switch at:

$T=TW_of_ctrl+500ps$

Analysis of LDO Low Power Designs

Overview

To support advanced techniques related to supply voltage control, such as multi-supply multi-voltage (MSMV), and dynamic voltage and frequency scaling (DVFS), on-chip low drop-out regulators (LDOs) are widely used to control each power domain. So LDO-aware voltage drop analysis has become significant for verifying the stability of the voltage supplied from the LDOs. This section describes the generation of LDO models and how to use them in RedHawk to perform static and dynamic voltage drop analysis that accurately capture the response of on-chip LDOs. The LDO-based flow in RedHawk is shown in Figure 13-29.

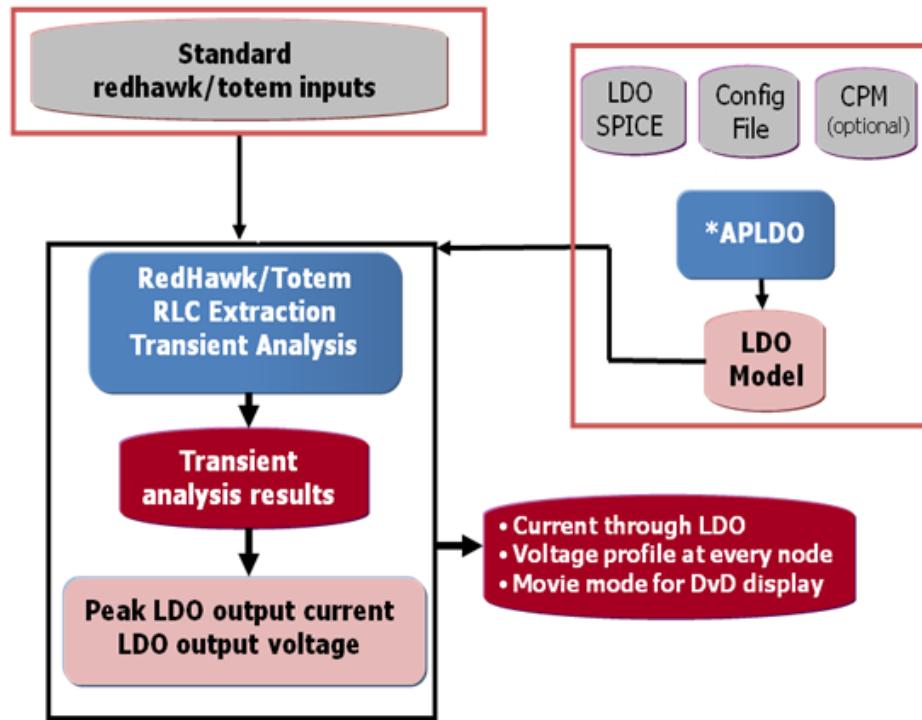


Figure 13-29 LDO analysis flow

LDO design modeling

Figure 13-30 shows a schematic of a typical LDO used in designs. To generate the LDO model, get DC biasing information for all the inputs to the LDO netlist.

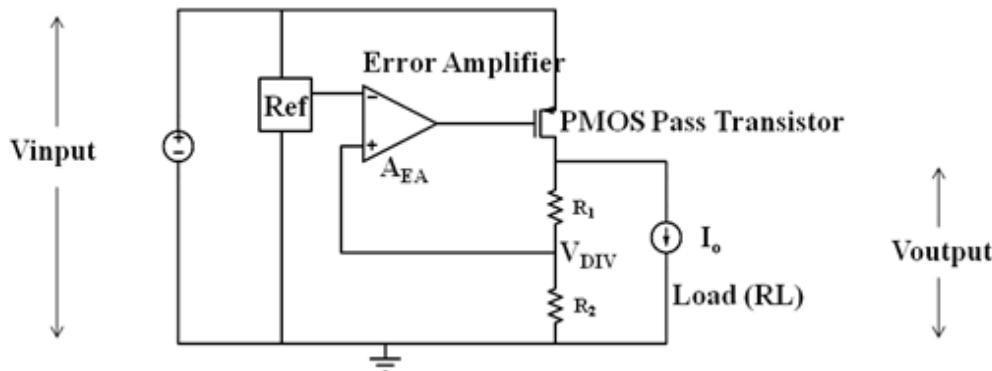


Figure 13-30 Typical LDO schematic

There are three types of LDO models generated by RedHawk, as follows:

- ideal LDO model with constant output voltage
- static models are current-controlled DC transfer voltage sources (first order approximation)
- dynamic/transient models also capture the transient behavior of LDOs, which account for the instantaneous drop in the voltage caused by high current demand (most accurate).

The LDO-based data flow is shown in Figure 13-31. The details of the procedure are given in the following paragraphs.

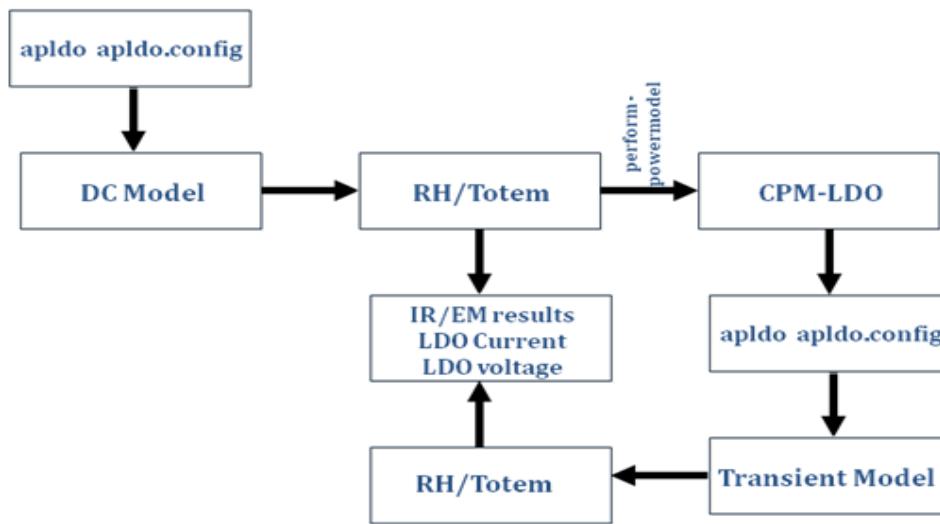


Figure 13-31 LDO data flow

There are four basic steps in LDO-based dynamic flow. In the LDO DC-based flow, only the first two steps are required.

1. Generate an LDO DC model using the APLDO utility (see section "Analysis of LDO Low Power Designs", page 13-383).
2. a. Generate a CPM model that includes the LDO input/output ports and possibly some internally-probed nodes.

To use LDO models, the following keyword must be in the GSR file:

```
LDO_MODEL_FILE {
    <ldo_model_file_from_APLDO>
}
```

The keyword ‘DECOUPLE_LDO_GROUND 1’ must be set in both the GSR and in the LDO configuration file for the LDO-based flows. When running the LDO flow in RedHawk and ‘DECOUPLE_LDO_GROUND 0’ is used, the LDO output voltage subtracts the Vss voltage at each time point, but ground bounce can be a problem. If ‘DECOUPLE_LDO_GROUND 1’ is specified, the original LDO output voltage is reported, which is usually desired. The default is 0.

Also, for LDO analysis the following keyword should be set in the *gds.config* file if an LDO is present in GDS:

```
WHITE_BOX_CELLS {
    <ldo_cell>
}
```

b. Use the generated DC LDO model in the RedHawk to create a CPM-LDO of the die, using the following command:

```
perform powermodel -nx <num_x_partitions> -ny
<num_y_partitions> -vcd -pincurrent -global_gnd -o CPM_LDO
-probe -reportcap
```

where

-probe : reads the <probe_node_file> defined in the PROBE_NODE_FILE GSR keyword, which is required if you want to check the DvD of the nodes in CPM+LDO_xtor+pkg runs outside of RedHawk. The format of <probe_node_file> is the following:

```
<x in um> <y in um> <layer> <node_name>
```

For full syntax description, see [section "perform powermodel"](#), page D-792.

3. For dynamic models, generate an AC LDO model, either for a load regulation dynamic model considering load variation, or for a line regulation dynamic model considering input variation as well, with CPM+LDO_xtor [+pkg] with APLDO (see the APLDO section for details).
4. Run RedHawk again with the AC LDO model for either VCD-based or vectorless-based analysis, to get accurate LDO output voltage/current responses depending on the load.

Controlling LDO switching state using GSC file

RedHawk can turn on/off LDO instances in dynamic analysis based on the settings in GSC file.

1. User can decide on-state or off-state for each LDO instance based on GSC setting. The state for a LDO instance cannot be changed during dynamic analysis. Wildcard(*) characters are accepted in this case.

Syntax:

```
<LDO instance name> [DISABLE | ENABLE]
```

Example:

```
LDO_* DISABLE
LDO_1 ENABLE
```

It means that all LDO instance with name matching LDO_* will be off-state, except for LDO_1, which will be on-state.

2. User can change any state for a LDO instance during dynamic analysis. This usage does not support wildcard(*) characters. The two new GSC file keywords 'LDO_INST_CONFIG' and 'END_LDO_INST_CONFIG' should be specified as in below format to enable this.

Syntax:

User should add comments using "#" before these 2 keywords.

```
# LDO_INST_CONFIG
<LDO instance name> [disable|enable] <time:pico sec.>
[disable|enable] <time:pico sec.> [disable|enable]
<time:pico sec.> .....
# END_LDO_INST_CONFIG
```

Example:

```
# LDO_INST_CONFIG
LDO_3 enable 10000 disable 15000 enable 20000 disable 25000
# END_LDO_INST_CONFIG
```

Example for Case1 + Case 2:

This usage doesn't support wildcard(*). Assuming that there are 4 LDO instances :

```
LDO_1 DISABLE
LDO_2 ENABLE
LDO_4 DISABLE
# LDO_INST_CONFIG
LDO_3 enable 10000 disable 15000 enable 20000 disable 25000
# END_LDO_INST_CONFIG
```

To check the state of an LDO instance, user can run "plot current -ldo -name <LDO instance name>". The file *adsRpt/Dynamic/ldo.current* will be impacted by this feature.

Outputs generated in LDO-based analysis

The following outputs are generated in LDO-based analysis in the directory *adsRpt/Dynamic*.

- *ldo.current* (output current waveforms)
- *ldo.voltage* (output voltage waveforms)
- *ldo_dynamic.rpt*
 - Min/Max load current value
 - Min/Max output voltage value
 - Min/Max input voltage value
 - Min/Max load current di/dt value

LDO Modeling with APLDO

APLDO is the LDO modeling utility that generates behavioral LDO models used by RedHawk. The following are the inputs and outputs for running APLDO:

- Input Data
 - Spice netlist and model parameters
 - proper configuration (keywords to enable modeling features and parameters)
 - CPM file (optional for optimal dynamic LDO models)

- package (optional when package decap data available for LDO output net)
- Output Data
 - encrypted LDO model file (NSpice/HSpice accept it as a subckt)
 - DC model - I-V table
 - transient model - RLC & control sources

Following are example configuration templates for the generation of DC, load regulation dynamic, and line regulation dynamic LDO models used in the APLDO utility:

LDO DC model example configuration file

```
# users can control the duration of the current ramp using this keyword
# (default is 300us). During LDO model creation, the I-V characteristics are
# extracted by simulating LDO using a current ramp.
DC_RAMP_TIME <load current ramp-up time>

# name of the LDO cell (required)
LDO_CELL LDO_cell

# name of output file (optional with default <ldo_cell>.apldo.mdl>)
#OUTPUT_FILE

# Spice netlist containing subcircuit definitions. Can be more than one
SPICE_NETLIST ../totem/netlist_subckt_vcc

# temperature condition (optional with default value 25 degree C)
TEMPERATURE 25

# DEVICE_MODEL_LIBRARY defines Spice device model (optional)
DEVICE_MODEL_LIBRARY ./user_data/SPICE/device.lib TT

# include files (optional)
INCLUDE ./totem/tt

# special Spice options (optional)
OPTION GSHUNT=le-12

# path to the Spice executable (optional -- NSpice is used by default).
# SPICE_SIMULATOR hspice /appl/synopsys/C-2009.09-SP1/hspice/linux/hspice

# minimum idle current (required)
# start current value in the model
MIN_IDLE_CURRENT 1.0mA

# minimum load current (A) (required)
# the minimum load current in the application
# generally, may set it the same as MIN_IDLE_CURRENT
MIN_LOAD_CURRENT 1.0mA

# maximum load current (A) (required)
MAX_LOAD_CURRENT 320mA
```

```
# dc current sweep step
DC_CURRENT_SWEEP_STEP 2mA

# output power pin name and ideal voltage (required)
POWER_OUTPUT_PIN vccr 2.2

# minimum output voltage (optional)
# MIN_OUTPUT_VOLTAGE 1.08

# minimum input voltage (optional)
# MIN_INPUT_VOLTAGE 2.40

# analog power input pin name (required)
POWER_INPUT_PIN vcc_in 3.0

# general ground pin name (required)
GROUND_PIN gnd 0

# dc bias
# <pin_name> <voltage>
DC_BIAS vcc_in      3.00
DC_BIAS gnd        0.0
DC_BIAS vbgr       1.25
```

APLDO example configuration file for load regulation dynamic model

```
# name of the LDO cell (required)
LDO_CELL LDO_cell

# name of output file (optional with default <ldo_cell>_apldo.mdl>)
#OUTPUT_FILE

# Spice netlist containing subcircuit definitions. Can be more than one.
SPICE_NETLIST ../totem/netlist_subckt_vcc

# temperature condition (optional with default value 25 degrees C)
TEMPERATURE 25

# DEVICE_MODEL_LIBRARY defines Spice device model (optional)
DEVICE_MODEL_LIBRARY ./user_data/SPICE/device.lib TT

# include files (optional)
INCLUDE ./totem/tt

# special Spice options (optional)
OPTION GSHUNT=le-12

# path to the Spice executable (optional -- NSpice is used by default).
#SPICE_SIMULATOR hspice /appl/synopsys/C-2009.09-SP1/hspice/linux/hspice
```

```

# minimum idle current (required)
# the start current value in the model
MIN_IDLE_CURRENT 1.0mA

# minimum load current (A) (required)
# the minimum load current in the application
# generally, may set it the same as MIN_IDLE_CURRENT
MIN_LOAD_CURRENT 1.0mA

# maximum load current (A) (required)
MAX_LOAD_CURRENT 320mA

# dc current sweep step
DC_CURRENT_SWEEP_STEP 2mA

# output power pin name and ideal voltage (required)
POWER_OUTPUT_PIN vccr 2.2

# transient model characterization for multiple output ground arcs.
POWER_ARC {
    POWER_OUTPUT_PIN <vdd_pin1> <voltage>
    ...
}
POWER_ARC {
    POWER_OUTPUT_PIN <vdd_pin2> <voltage>
    ...
}

# minimum output voltage (optional)
# MIN_OUTPUT_VOLTAGE 1.08

# minimum input voltage (optional)
# MIN_INPUT_VOLTAGE 2.40

# analog power input pin name (required)
POWER_INPUT_PIN vcc_in 3.0

# general ground pin name (required)
GROUND_PIN gnd 0

# dc bias
# <pin_name> <voltage>
DC_BIAS vcc_in      3.00
DC_BIAS gnd        0.0
DC_BIAS vbgr       1.25

# Defines the top level subckt name of CPM + LDO test bench (required)
CPM_LDO_MODEL adsPowerModel

# CPM_LDO_FILE <file name>
```

```

# To specify filename of the CPM + LDO test bench (required)
CPM_LDO_FILE ./power_model_CPM-LDO

# CPM_LDO_PORT <port name> <value>
# Defines the bias conditions for CPM_LDO_MODEL. Users must check the
# PLOC and set the corresponding pad value to the ports of CPM_LDO_MODEL.
# Wildcards supported to specify the port name of CPM_LDO_MODEL (required)
CPM_LDO_PORT *:VCC 3.0
CPM_LDO_PORT *:VSS 0

# CPM_OPEN_PORT <port name>
# Defines the open ports of CPM_LDO_MODEL. The defined ports remain floating
# while characterizing the LDO model. Wildcards are supported to specify the
# port name (optional)
CPM_OPEN_PORT VBB:*

# CPM_LDO_TRANSIENT_TIME <time>
# Defines the transient simulation time in APLDO characterization.
# Users may refer to the RedHawk simulation time for CPM model generation
# and specify the same in CPM_LDO_TRANSIENT_TIME for LDO models (optional).
CPM_LDO_TRANSIENT_TIME 120ns

# the subckt name of PKG model (optional)
# but is required when CPM_LDO_PORT is not specified
# PACKAGE_MODEL <subckt name>
PACKAGE_MODEL REDHAWK_PKG

# the PKG model filename (optional)
# it is required when PACKAGE_MODEL is given
# PACKAGE_FILE <pkg_file_name>
PACKAGE_FILE redhawk_pkg_0.spi

```

Generating the DC LDO model

Use the following command to generate the DC LDO model:

```
% apldo apldo.config
```

For APLDO debugging help, set "DEBUG 1" in configuration file, which keeps the .apache/APLDO directory for the following files:

- ldo_dc_vdd.cir (for dc sweep simulation)
- ldo_ac_vdd.cir (for ac sweep simulation)

Testing LDO models

Generated LDO models are encrypted, so you can perform sanity checking on the model by running Spice simulations on the <>_LOAD_REGULATION_TEST_BENCH.sp file in the apldo working directory.

Also, the command 'aplreader -ldo <LDO model>' can be used to plot LDO output I-V curves.

Other practical LDO applications

The LDO methodology described not only analyzes voltage drop, but also can assist in adjusting the on-chip LDO size (that is, the drive strength) and the locations for power reduction and chip shrink. For example, by replacing a normal-sized on-chip LDO with one that is half-sized, designers can compare the two voltage waveforms at the output pin of those LDOs, as shown in Figure 13-32. If the half-sized one meets design constraints in voltage drop, designers can use it instead of the normal-sized one. This means that as another application, this method provides a what-if analysis for adjusting on-chip LDO size. Moreover, designers can adopt the method without any modifications to low power techniques such as MSMV and DVFS. And designers have other advantages in chip modeling, such as Chip Power Model (CPM), for chip-package co-design, using this method.

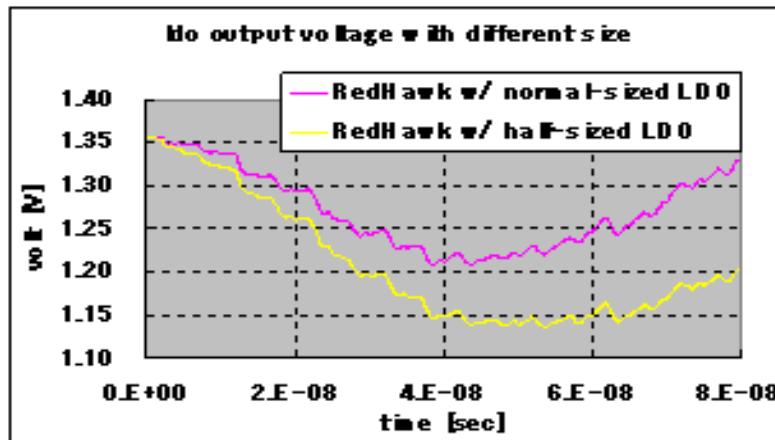


Figure 13-32 LDO output voltage for normal and half drive strength

Analysis of Gated Clock Designs

In order to support the many mobile and high-density applications, low power consumption has become a necessity in today's IC design. One type of low power design turns on only parts of the circuit at a time. These power modes can be switched in and out by turning different clock domains on/off along with their corresponding logic. The multiple clock domains are usually controlled by multiplexing logic and clock control inputs, referred to as "gated clock" design.

From a power analysis point of view, it is unrealistic to analyze the whole chip with all clocks on. The key is to identify the various power modes, and analyze the critical ones in terms of dynamic voltage drop and impact on timing. For details on performing Gated Clock design analysis, see [section "Gated Clock Dynamic Analysis", page 5-94](#).

It is important to recognize that for gated clock designs, each "mode" must be analyzed separately, with its mode-specific static timing analysis file.

Chapter 14

Chip Power Modeling (CPM)

Note: The generation of CPM is considerably more complex than regular dynamic voltage drop analysis, since the CPM generation process must preserve both the time and frequency domain response of the underlying circuit for a wide range of frequency points (DC to multi-GHz). However, the typical usage model of CPM creation assumes a continuation of a dynamic voltage drop analysis, whose settings and network formulation may not be appropriate for CPM generation, resulting in additional complexities.

Starting in version 12.2, RedHawk requires users to use a separate CPM creation flow (using the GSR keyword setting: GENERATE_CPM 1), which ensures that the CPM generation flow is distinct from dynamic voltage drop analysis. This provides RedHawk the flexibility to provide additional optimization necessary for CPM creation without affecting standard dynamic voltage drop analysis.

Introduction

Chip package and printed circuit board (PCB) designers need an accurate and relatively simple IC power model to design and optimize effective chip packages and boards -- including key parameters such as impedance and resonant frequencies of the global power delivery network (PDN). An equivalent circuit for the chip PDN must provide not only an accurate multiple-terminal impedance model, but also accurate current waveforms to represent a realistic worst-case switching scenario for the chip.

Apache's Chip Power Model (CPM) enables die-package-board co-design and co-verification for dynamic power integrity. Built on the **RedHawk** full-chip dynamic power integrity platform, CPM generates a compact and accurate model of the full-chip power delivery network at various key stages of chip and package design.

CPM supports the VectorLess™ clock gating switching mode, in addition to the default vectorless and VCD modes. In clock gating Vectorless mode, **RedHawk** generates a switching scenario that mimics the transition from one clock gating mode to another during multi-cycle transient analysis. A CPM created in this mode captures the current transitions when this clock gating scenario change occurs. This change in current demand is particularly useful to understand the associated Ldi/dt effect in package and board level analyses. CPM has a Resonance Frequency Aware Excitation Mode, in which the **RedHawk** VectorLess engine generates a multiple-cycle switching scenario for the current signature that introduces most of the energy around the chip-package-system resonance frequency. You only need to specify the system resonance frequency when creating the model, and the CPM technology automatically creates the frequency-based form of on-die excitation, while maintaining the logic and timing properties of the circuit.

CPM bridges the design of the power delivery network between the IC and the associated package and PCB. System designers can use CPM to guide and verify the off-chip PDN design by evaluating the impact of on-die parasitics over the global PDN impedance, diagnose potential chip-package LC resonance, validate the package/board dynamic voltage noise margin, as well as to optimize the off-chip decoupling capacitor placements.

Both current signatures and parasitic network are distributed across a multiple-terminal equivalent circuit to reflect their temporal and spatial dependencies. The simplest element of the CPM model can be considered to be a serially-connected R_{DIE} and a C_{DIE} , in parallel with a current switch, as shown in Figure 14-1.

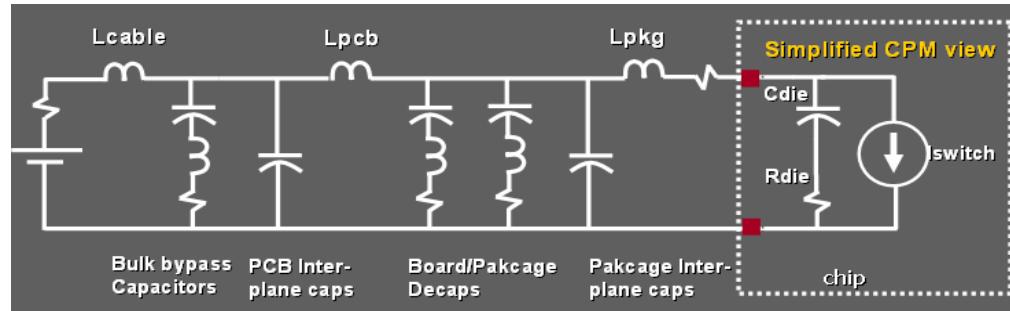


Figure 14-1 Power Delivery Network and simple Chip Power Model

SPICE-compatible CPM includes parasitics of non-linear switching and non-switching devices, parasitics of power/ground wires, decoupling capacitors, and effective RC's of the loading capacitors from signal interconnects. In addition, CPM contains full-chip switching current signatures based on transistor-level SPICE simulations.

Design Flow

CPM technology can be useful at all stages of the package-IC system co-design process, as shown in the Figure 14-2 flow diagram.

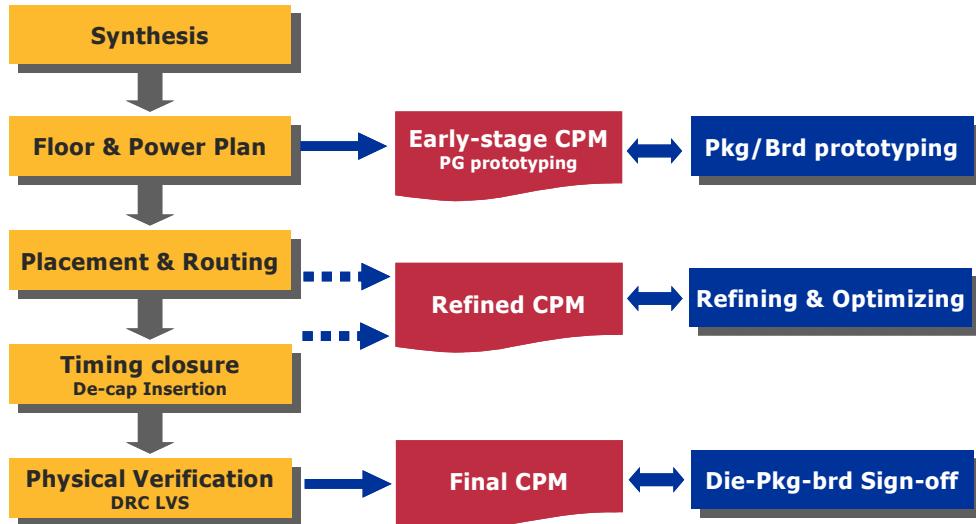


Figure 14-2 CPM-based IC system co-design process

The generation of CPM is considerably more complex than regular dynamic voltage drop analysis, since the CPM generation process must preserve both the time and frequency domain response of the underlying circuit for a wide range of frequency points (DC to multi-GHz).

An improved CPM creation flow using the required GSR keyword 'GENERATE_CPM 1' ensures that the CPM generation flow is distinct from the dynamic voltage drop analysis flow, and provides RedHawk the flexibility to optimize CPM creation without affecting dynamic voltage drop analysis.

RedHawk Modeling of Chip Power Delivery Network

A chip power delivery network can be modeled in several ways by CPM, depending upon the number of power/ground pads involved and also the desired trade-off between speed and accuracy of analysis.

Modeling Choices Based on Number of Pads

If a large number of die pads are involved, such as several thousand in a typical flip chip package design, the die pad area is partitioned into smaller rectangular areas and CPM generates a set of terminals corresponding to each Vdd and Vss net within each partition. For wirebond designs with a large number of pads, they also can be grouped into individual partitions of pads for better analysis based on the Spice node name specification in the PLOC file. For designs with a smaller manageable number of power/ground pads, all pads are represented individually by CPM terminals.

All significant sources of capacitance in a chip, including parasitics and decoupling, are included in the **RedHawk** chip model, as shown in the circuit of Figure 14-3:

- Intentional de-caps, from **RedHawk** characterization
- Intrinsic device de-caps, from **RedHawk** characterization
- Signal loading capacitance, from SPEF (StarRC)
- Coupling capacitance between power and ground wires, from **RedHawk** extraction

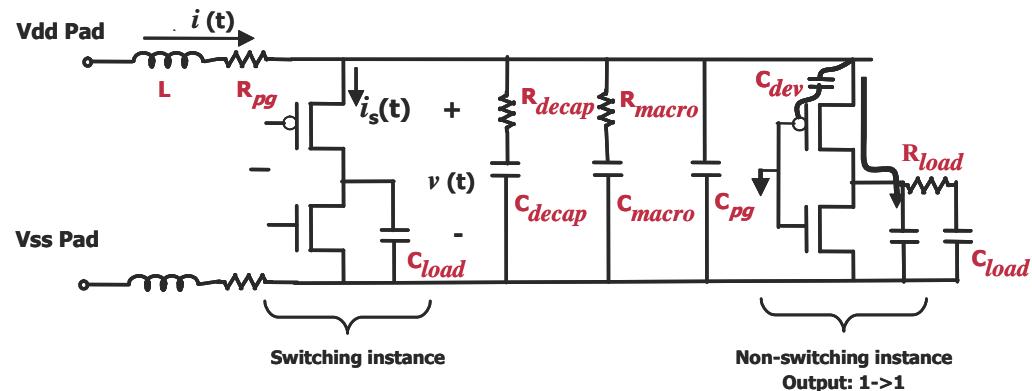


Figure 14-3 RedHawk chip power modeling circuit

CPM for Flip Chip Designs

For flip chip designs, the chip bump area can be partitioned into $N \times M$ tiles. Assuming there are K power/ground nets (that is, one or more Vdd nets and one or more Vss nets), then the maximum number of external terminals of the resulting CPM will be $K \times N \times M$. Figure 14-4 shows a flip chip example with one VDD net and one VSS net. The number of partitions is 4 ($N=2$, $M=2$).

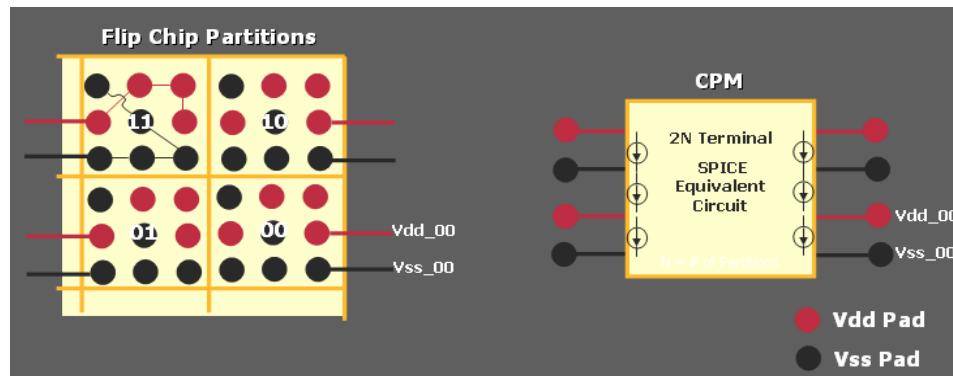


Figure 14-4 Modeling a chip power circuit for a flip-chip package

CPM for Wirebond Designs

For wire-bond designs, an N-terminal CPM is created, for which typically each wire-bond pad corresponds to a terminal of the model, as shown in Figure 14-5.

For PLOC files that have grouped ports (package nodes) and named them accordingly, CPM by default uses the PLOC port groups and generates a set of external ports that follow the grouping pattern in the PLOC file.

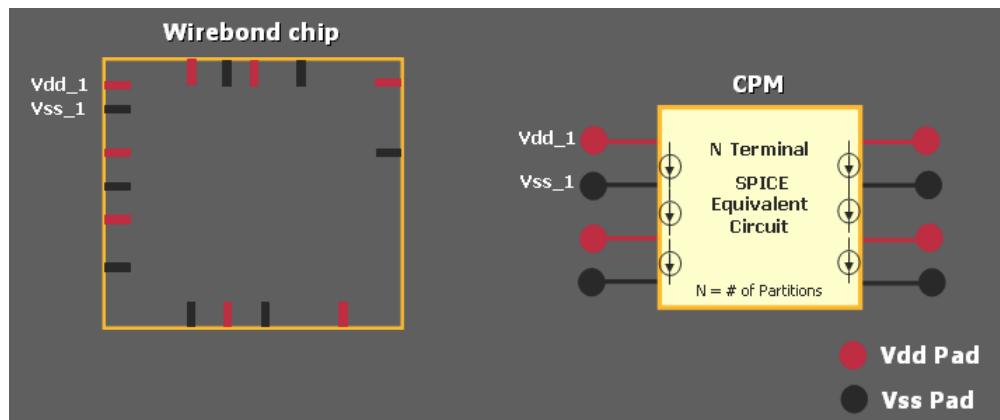


Figure 14-5 Modeling a chip power circuit for a wirebond package

3DIC CPM Generation

A combined CPM can be generated for the complete 3DIC structure, as shown in Figure 14-6. For example, if there is a memory die stacked on a logic die, CPM generation for both logic and memory die combined can be performed using the following Tcl commands:

```
import gsr mem.gsr -die mem
import gsr logic.gsr -die logic
setup design
perform pwrcalc
perform extraction -power -ground -r -c
perform powermodel -nx 1 -ny 1 -o 3dic_cpm.sp
```

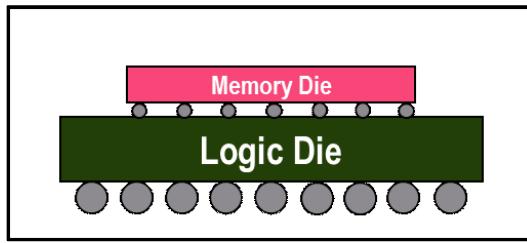


Figure 14-6 3DIC CPM generation

Modeling Choices Based on Analysis Speed and Accuracy

High speed modeling

The “model order reduction” (MOR) methodology is based on a DC/zero frequency-centered solution, that becomes more inaccurate as the switching frequencies of the design become higher, but saves runtime for conditions that do not require high frequencies and high accuracy.

In general an assumption of power-ground symmetry is made, which means that package power and ground impedances are considered similar. However, setting the GSR keyword DYNAMIC_SOLVER_MODE to 1 allows a P/G solution without this assumption, increasing the accuracy, although this makes it a more time-consuming and memory-intensive methodology.

High Accuracy Modeling

“AC” Modeling

The default AC analysis-based version of CPM significantly improves the accuracy of the model, and has far better correlation with RedHawk simulations. CPM model construction consists of two parts: calculation of current signatures and the reduction of the passive part of the circuit.

The AC analysis-based CPM generation uses grid RC reduction applied to the original RC network that contains not just the resistance and capacitance representing the parasitics of the power/ground wires, but also the series RC branches that model the parasitics of non-switching instances. Switching instances are modelled as time and voltage-dependent current sources.

The analysis takes the large network of resistors and capacitors (typically with millions of nodes) and generates a much smaller network (typically with tens or hundreds of nodes), whose frequency domain response at the specified ports for the specified frequency range matches that of the original network, with a tolerance better than 0.2 %. The resulting network, which consists of R, C, and possibly L and G (constant Voltage Controlled Current Source) elements, is exported as a SPICE netlist. The netlist is guaranteed to be passive by construction, and thus suitable to be used in transient simulations with SPICE or another transient simulator. This netlist can be used with any version of SPICE (such as NSPICE from Apache and HSPICE from Synopsys) as it only uses classic circuit elements.

S-Parameter Modeling

CPM can generate an S-parameter model of the RLC grid and export it to a Touchstone file. This allows you to generate frequency-dependent S-parameters for the on-die power

grid netlist. The S-parameters are calculated at the frequencies at which the AC analysis is performed. To use this capability, you need to specify the configuration file in the command

```
perform powermodel -grid [RC | RLC ]
    -options <config_file_path>'
```

or *RC_reduction.config* in the run directory. In the configuration file, you must specify the path to the port definition file as:

```
portFileName=<port file name>
```

and specify the flag to enable S-parameter model generation ('sParam=true'). You can also specify the value of the reference impedance (*Zo=<value>*).

Example configuration file:

```
sParam=true
Zo=50
portFileName=port.file
```

In this case, the reference impedance is 50 Ohms (default is *Zo*=2 Ohms), and the port file is *port.file* in the run directory. In the port file, the ports of the S-parameter model are specified as CPM port (terminal) pairs. The format of the port file is:

```
<CPM_port_name1> <CPM_port_name2>
```

Note that the CPM port name is either the pad name, if there is no grouping (no 6th column in the ploc file), or the name given in the 6th column of the PLOC file. An example of a port file follows:

```
pwr_0 gnd_0
pwr_1 gnd_1
pwr_2 gnd_2
pwr_3 gnd_3
pwr_4 gnd_4
pwr_5 gnd_5
```

Note: there is no S-parameter port name; the first port is between the CPM ports (terminals) *pwr_0*, *gnd_0*, the second port is between (*pwr_1*, *gnd_1*), and the 10th port is between (*pwr_9*, *gnd_9*).

Example CPM command:

```
perform powermodel -grid RC -plocname -options rc.config -o output.sp
```

The S-parameter (Touchstone) filename is based on the specified output filename. For example, for an output specification '*-o rc_grid.sp*', the Touchstone filename is *rc_grid.Snp*, where *n* is the number of ports. For example, if the output file is *output.sp* and there are 10 ports, the S parameter file is *output.s10p*, written to the run directory. There is also a regular CPM grid netlist output *output*sp* file.

A sample PLOC file follows:

```
VDD01 10 20 metal5 POWER VDD_group1
VSS01 11 21 metal5 GROUND VSS_group1
VDD02 5 20 metal1 POWER VDD_group2
VSS02 6 21 metal1 GROUND VSS_group2
```

Note that automatic presim time determination is supported in S-parameter package models. Set the GSR keyword 'DYNAMIC_PRESIM_TIME -1' to enable this function.

ESD-aware CPM Models

ESD-aware CPM models can be created for system-level ESD analysis. Clamp connections are modeled as ports in the CPM model, and sub-circuits for clamp devices then are included in the CPM model. The command steps are as follows:

```
setup analysis_mode esd
```

```
setup design <GSR_file>
...
perform extraction -power -ground
...
perform powermodel -esd ?-esd_clamp <file>? ...
```

Note that the ‘-esd_clamp’ option is not needed if a clamp file is specified using the ‘ESD_CLAMP_FILE’ keyword in the GSR.

CPM Simulation Procedures

Initial Setup and Preparation

The required input data for CPM is the same as you need for a full-chip RedHawk DvD analysis. You must use the correct APL characterization data, or AVM characterization if APL data is missing for memories and custom macros (see [Chapter 9, "Characterization Using Apache Power Library"](#)).

The switching current component of the CPM depends on the switching activity in the design during transient simulation. Hence setting proper switching rates is critical. This can be achieved either by using the Vectorless algorithm (see [Chapter 5, "Dynamic Voltage Drop Analysis"](#)), or by using a VCD file (either gate level or RTL level VCD). In contrast to the DvD analysis flow, you do NOT need to specify any package models during a CPM run.

Additional input parameters to be decided and specified for a flip chip CPM model are the number of partitions desired in the X and Y directions.

Running CPM

The Chip Power Model is created from the RedHawk flow, by reading the input data, performing power calculation and on-die P/G extraction, and then creating the electrical model of the chip power delivery network and current profiles based on the defined partitioned network.

The RedHawk command line setup and invocation command steps are as follows:

1. Import and set up design.

The GSR file contains the paths to all design and library files (including APL/AVM) data and simulation conditions. See [section "Global System Requirements File \(*.gsr\)", page C-573](#), for a description of the GSR keywords. The keyword ‘GENERATE_CPM 1’ must be set or the CPM run fails.

```
setup design <design>.gsr
```

2. Specify a temporary file location on a local disk. Particularly when multiple processors are used and for large designs (more than 10 M nodes), writing temporary files to a disk on the network slows down execution because of I/O operations. Put temporary files on a local disk by setting the CACHE_DIR GSR keyword. After chip power modeling finishes, the temporary files (linear.*) in CACHE_DIR are deleted.

3. Determine appropriate toggle rates and calculate average power.

```
perform pwrcalc
```

4. Extract RLC parasitics of on-die P/G networks.

```
perform extraction -power -ground -c -l
```

5. Set up the files to determine an accurate switching scenario for dynamic analysis:
 - a. To use a vectorless method to determine the switching scenario, including clock gating, see [section "Vectorless Dynamic Analysis", page 5-83](#), for instructions on setting up the proper GSR keyword settings.
 - b. To use a VCD file to specify the switching scenario, use the GSR keyword **VCD_FILE**. Refer to [section "VCD_FILE", page C-609](#), for syntax and usage.

6. The command syntax for generating a CPM is as follows:

```
perform powermodel [ -wirebond | <flip chip partitions>] -cdie| -static
| -lowpower | -esd ] ? -esd_clamp <file> ? ? -parasitic ? ?-vcf ?
?<no model option-default>? ? -pincurrent ? ? -rleak ? ? -rleak_par?
?-solver mor? ?-plocname? ? -ind? ? -no_afs? ? -passive ?
? [ -noglobal_gnd | -global_gnd ] ? ? -high_capacity 1 ?
? -repeat_current [ <start_time> | presim | best ]? -probe ?
? -internal_node -cell_file <cell_list_filename> ? ? -reportcap ?
? -o <output_filename> ? ? -reuse ? ? -io ?
```

where

-wirebond : specifies a wirebond package

<flip chip partitions> : -nx <num_x_partitions> -ny <num_y_partitions>, specifies the number of partitions in the x and y directions. For -nx 1, -ny 1, a Cdie/Rdie report is generated in the *adsRpt/CPM/apache.Cdie* file.

-cdie : sets nx=1 and ny=1, and connects all power nets together and all ground nets together to obtain a single-port solution to obtain the equivalent Cdie and Rdie values for the chip. No current waveform is generated. A Cdie/Rdie report is generated in the *adsRpt/CPM/apache.Cdie* file.

Note that the **-cdie** option just provides faster run time by not calculating the current waveforms. For all other purposes it is equivalent to using '**-nx1 -ny1**'.

-static : creates static analysis chip power model, using DC conditions, a resistance-based circuit, and average current.

-lowpower: CPM generation support for ramp-up analysis.

-esd: creates a CPM that includes clamp device models/characteristics, along with the PDN model Spice deck. Clamp connections are modeled as ports in the CPM model, and sub-circuits for clamp devices are included in the CPM model. You must also specify the clamp files in the GSR. If the clamp files are not present in the GSR, use the "**-esd_clamp**" option to specify the clamp file.

-parasitic: generates only the passive part of the CPM, without performing transient simulation, to generate the current signatures of the CPM ports, an extension of **-cdie** option for multi-partition CPMs. This can save time in transient simulation. However, the CPM model generated with **-parasitic** can only be used for DC and AC analysis (not usable for transient analysis).

-vcf : uses VCD file as basis for determining the worst case switching scenario

-pincurrent: specifies that CPM generate the model without current conservation (balanced current between Vdd and Vss) to achieve better correlation with RedHawk dynamic simulation results. In general, CPM enforces current conservation. However, in cases when RedHawk does not produce balanced VDD and VSS currents, this option can be used.

-rleak: causes the leak resistance to be added between ports on the VDD (power) net and the reference port on the VSS net.

-rleak_par: inserts leakage resistance between the VDD and VSS ports of each defined partition. Note that this option only works with partitioned CPM models. See the following section for more details on usage.

- solver mor: turns off the default ‘solver ac’ function, which is an accurate frequency-based linear solver AC solution with passivity enforcement.
- plocname: specifies pad/group names for CPM port names. If the '-wirebond' option is used, the subcircuit terminal names are taken from the pad names (if no grouping is specified), or the group names from the 6th column of the *.ploc* file. These group names are also known as SPICE node names, as this mechanism is used to connect a package using the keyword 'PACKAGE_SPICE_SUBCKT'. If this option is used with '-nx # -ny #' options, the node names are generated in the following form: PAR_0_0_VDD1, PAR_0_0_VSS2, ... For wirebond CPMs without any grouping, the subckt terminal name is the ploc name.
- ind: accounts for on-chip inductance, if the option '-l' of the RedHawk 'perform extraction' command has been used

Note: If you specify the '-l' option of the 'perform extraction' command, but not '-ind', inductance is ignored. Not specifying the '-l' option and using '-ind' is an error.

- no_afs : turns off the default AFS function. Adaptive Frequency Sweep for AC mode execution intelligently selects seven to nine frequencies (enough to reach convergence) to perform AC analysis, as opposed to 26 frequency samples that are performed by default in ‘solver ac’ mode. This option is recommended for design sizes exceeding 50 ports. Port count can be determined by computing $N * M * P$, where N = number of X partitions in the CPM (-nx option), M = number of Y partitions in the CPM (-ny option), and P = number of power and ground domains.
- passive : only effective with the MOR function, which uses MOR to generate a passivity-enforced SPICE model, which can reduce accuracy. This mode can be useful if the SPICE simulation of the package/PCB CPM has convergence issues. The SPICE netlist is significantly smaller than using the default mode. But CPM is a compact model, so complexity is not a concern for cases with, for example, 100 bond pads, or up to 10x10 partitions for a flip-chip design.
- noglobal_gnd | -global_gnd: specifies the type of parasitic modeling in CPM, either (a) without Spice Node 0, using option '-noglobal_gnd' (the default), where there is a direct connection between the power and ground ports without going through Spice node 0, or (b) using Spice Node 0, using option '-global_gnd', in which the RLC parasitics from power and ground are connected to Spice global ground (node 0).
- high_capacity 1: specifies use of the asim3d solver for CPM. The default CPM generation solver is ASIM3D(on by default). Specifying -high_capacity 0 will switch the solver to asim_mixed.
- repeat_current: specifies that the CPM current signature is repeated starting from the specified time point. Either a <start_time> in ns, the ‘presim’ time, or ‘best’ (chosen to cause the best continuity at the repeat point), can be chosen as the starting point of the repeating waveform. A warning is issued for incorrect values (such as a negative value or value greater than the maximum time of the PWL source). For non-zero values the closest time in the PWL definition is used. For example, if a ‘-repeat_current 1n’ option is specified, and PWL time values ..., 900, 930, 960, 990, 1020, ... ps are defined, ‘R=990 ps’ is used. This is required for SPICE to accept the netlist. For the ‘best’ option to work well, the presim time and the transient simulation time need to be set to “n*T”, where n is a positive integer, and T is the period of the clock frequency. With this option, the repeat time may be different for each individual PWL current source. Usage examples:

```
perform powermodel -nx 2 -ny 2 -repeat_current 0
which repeats starting from the beginning, t=0. Or,
```

perform powermodel -nx 2 -ny 2 -repeat_current 2n
 which repeats starting from the time value in the PWL source definition
 closest to the 2ns time specified.

-probe: invokes the iCPM utility that enables the visibility of sensitive P/G connections in the design, and allows you to probe device locations inside the chip. You must set the PROBE_NODE_FILE GSR keyword, as described in section "[iCPM- Internal Node Probing](#)", page 14-405.

-internal_node: specifies additional ports located at P/ G pins on the same net that are to be shorted together to form one internal port. These nodes are named with the format '<instance name>_<netname>' in the CPM, such as "inst_1_VDD". Note that the option "-internal_node" is not supported in static analysis.

Note that the options '-internal_node' and '-cell_file' are required to execute this feature, and the '-pincurrent' option should be used to keep the CPM currents at the correct value without further modifying the port currents, so that the sum of all port currents is zero. This feature also allows you to include/exclude the instance current profile and device capacitance from CPM creation, using the EXCLUDE option in the GSC file. To exclude instance current profiles and device capacitance, use the GSC syntax: '<instance name> EXCLUDE'.

-cell_file <cell_list_filename>: specifies a file containing a list of the instances whose internal P/G pins are to be exposed.

-reportcap: creates a log file report of all capacitance components included in the CPM generation. Example output:

```
Capacitance components -
    Intentional Decap - 0.000000e+00 pF
    Intrinsic   Decap - 2.404236e+02 pF
    Load        Decap - 1.945542e+02 pF
    Power grid  Decap - 3.260729e+00 pF
    Well        Decap - 0.000000e+00 pF
    Total       - 4.382385e+02 pF
```

-o <output_filename> : specifies output filename (default - *PowerModel.sp*, with the passive part in the file *PowerModel.sp.inc*)

-reuse : allows reuse of the generated current waveforms after one CPM run, if chip power models with different partitioning schemes are desired

-io : specifies that the I/O cells are to be included in the chip power model

7. So to generate a CPM for a flip chip design:

- to use a vectorless methodology for an accurate solution:

```
perform powermodel
    -nx <num_x_partitions>
    -ny <num_y_partitions> -o <output_filename>
```

- to use a VCD scenario, use the same command as in step (a) and use the '-vcd' option.

- to generate a CPM quickly for a wirebond design:

```
perform powermodel -wirebond -solver mor
    -o <output_filename>
```

- to generate the passive part of a CPM (in a file *cpm.sp.inc*) with 2 x 3 partitioning:

```
perform powermodel -nx 2 -ny 3 -parasitic -o cpm.sp
```

Note that AC mode analysis in Chip Power Modeling can provide a fast solution using the linear solver with multi-threading. You can set the maximum number of processors to be used, which by default is the lesser of the number of processors available or 8. To use more than 8 processors, or fewer, you can set the Linux/UNIX environment variable '`'MAX_CPU <number>`'. Then the number of processors used is the lesser of the processors available and the `MAX_CPU` value. For example, using '`setenv MAX_CPU 4`' limits the maximum number of processors in use to four.

Basic power integrity analysis

The following GSR keywords settings are recommended for creating CPMs for basic power integrity analysis:

`DYNAMIC_TIME_STEP <ps>` : time step should be less than 20ps

`DYNAMIC_PRESIM_TIME < time > <TSM=1>` : do not use `TSM > 1.0`. For example:

`DYNAMIC_PRESIM_TIME 10n 1.0`

For multiple-Vdd designs, the following GSR keyword must be used. Note that run time and memory use will increase significantly for these cases.

`DYNAMIC_SOLVER_MODE 1`

The recommended command options to generate CPM for power integrity analysis are:

```
perform powermodel [-wirebond | -nx <x-part.> -ny <y-part.>
] -plocname -repeat_current <start_time> -rleak -ind
```

The effects of using the `-pincurrent` option are described in Table 14-1 below.

Use of option -pincurrent	Dynamic Solver	I(t)	Passive portion
Not used	0	Current from N-1 ports flow to one common VSS port.	Coupled between VDD and VSS. Not affected by these options.
Not used	1	Current from N-1 ports flow to one common VSS port.	Coupled between VDD and VSS. Not affected by these options.
Used	0	Current for all N ports flow to Spice node 0.	Coupled between VDD and VSS. Not affected by these options.
Used	1	Current for all N ports flow to Spice node 0.	Coupled between VDD and VSS. Not affected by these options

Table 14-1 Effects of -pincurrent option use

The effects of using the `-noglobal_gnd` and `-global_gnd` options are described in Table 14-2 below.

Use of options -noglobal_gnd and -global_gnd	Dynamic Solver	I(t)	Passive portion
<code>-global_gnd</code>	0	Not affected by these options. Controlled by <code>-pincurrent</code> .	Capacitance connected to Spice node 0 (global ground)

Use of options -noglobal_gnd and -global_gnd	Dynamic Solver	I(t)	Passive portion
-global_gnd	1	Not affected by these options. Controlled by -pincurrent.	Coupled between Vdd and Vss.
-noglobal_gnd (default)	0	Not affected by these options. Controlled by -pincurrent.	Coupled between Vdd and Vss.
-noglobal_gnd (default)	1	Not affected by these options. Controlled by -pincurrent.	Coupled between Vdd and Vss.

Table 14-2 Effects of -noglobal_gnd and -global_gnd option use

EMI modeling

The following GSR keywords settings are recommended for creating CPMs for EMI analysis:

```

COUPLEC 1
DYNAMIC_SOLVER_MODE 1
DYNAMIC_TIME_STEP <ps> : time step should be less than 20ps
DYNAMIC_PRESIM_TIME < time > <TSM=1> : do not use TSM > 1.0

```

The recommended command options to generate CPM for EMI analysis are:

```

perform powermodel -pincurrent -couple_gridc [-wirebond |
      -nx <x-part.> -ny <y-part.> ]
      -plocname -repeat_current <start_time> -rleak -ind

```

User-specified Grouping for Port Creation

CPM port generation has two methodologies, wirebond and flip chip, matching the pack designs. For designs with a low number of power and ground connections, such as wirebond designs, the wirebond option ('perform powermodel -wirebond') is recommended. By using the wirebond option, a CPM is created with a port representing each of the power and ground connections, as defined by the PLOC file.

For designs with a higher port count or those containing a distributed sea of power and ground connections, such as a flip chip design, the flip-chip configuration is recommended. In this approach, the chip is divided into regions specified by the user (that is, 'perform powermodel -nx 4 -ny 4'). In each region, a CPM port is created for each analyzed domain.

In some cases, the user may want use a hybrid approach. For example, for some domains, per-pad resolution (wirebond) is required, but for other domains a grouped representation is preferred. This section describes the procedure to enable user-specified grouping for the CPM wirebond option.

Wirebond Grouping Procedure

Arbitrary partitions are created by adding a 6th column to the PLOC file, and then using the '-wirebond' option of the 'perform powermodel' command. In the following example, there are 8 individual entries to the PLOC file. The 6th column represents the user-specified group name. The first two power source locations (DVDD1 and DVDD2) are grouped into a single CPM port (GROUP_POWER_1). Similarly, the first two ground sources (DVSS1 and DVSS2) are grouped together into a single CPM port

(GROUP_GROUND_1), as shown in Figure 14-7. The remaining PLOC sources are represented by a unique CPM port.

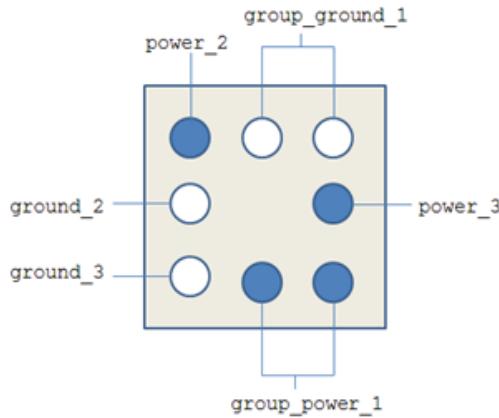


Figure 14-7 Wirebond port grouping

Example PLOC file:

```
#source name #loc_x #loc_y #layer name #POWER/GROUND #Group name
DVDD1 10 5 METAL6 POWER GROUP_POWER_1
DVDD2 15 5 METAL6 POWER GROUP_POWER_1
DVDD3 10 5 METAL6 POWER POWER_2
DVDD4 15 10 METAL6 POWER POWER_3
DVSS1 10 15 METAL6 GROUND GROUPGROUND_1
DVSS2 15 15 METAL6 GROUND GROUPGROUND_1
DVSS3 10 5 METAL6 GROUND GROUND_2
DVSS4 5 5 METAL6 GROUND GROUND_3
```

To generate a CPM with user-configured grouping, use ‘perform powermodel -wirebond’. If the ‘-plocname’ option is used, the CPM port names reflect the 6th column group names from the PLOC file.

Modeling leakage resistance using arbitrary partitioning

You can use the ‘-leak_par’ option to capture leakage resistance in CPM between POWER/GROUND ports defined in arbitrary partitions. Arbitrary partitions are created by adding a 6th column to the PLOC file and then using the ‘-wirebond’ option of the ‘perform powermodel’ command. Pads with the same “package node” (the entry in the 6th column of the PLOC file) are grouped together, forming a so-called “CPM port”. Unlike geometric grouping using ‘-nx # -ny #’ partitioning, an additional file *partition.txt* is needed to define which VCC and VSS ports belong to the same partition. The format of the *partition.txt* file is ‘<package node> < group name>’.

The following is an example partition definition in a *partition.txt* file:

```
bump_vcc1 part1
bump_vcc2 part2
ref1 part1
ref2 part2
```

In this example, the partition named ‘part1’ has ‘bump_vcc1’ as its VCC port, and ‘ref1’ as its VSS port. The partition named ‘part2’ has ‘bump_vcc2’ as its VCC port, and ‘ref2’ as its VSS ports. So if you specify ‘-leak_par’, two leakage resistances are added, one

between 'bump_vcc1' and 'ref1', and the other between 'bump_vcc2' and 'ref2'. The file *partition.txt* should be placed in the run directory.

The CPM command is:

```
perform powermodel -rleak_par -wirebond
```

Example PLOC file

```
#source name #loc_x #loc_y #layer name #POWER/GROUND #Group name
DVDD41 35 3426 METAL6 POWER GROUP41_POWER
DVSS41 35 3548 METAL6 GROUND GROUP41_GROUND
DVDD42 35 3730 METAL6 POWER GROUP42_POWER
DVSS42 35 3790 METAL6 GROUND GROUP42_GROUND
DVDD43 35 3976 METAL6 POWER GROUP43_POWER
DVSS43 35 4028 METAL6 GROUND GROUP43_GROUND
DVDD44 35 4212 METAL6 POWER GROUP44_POWER
DVSS44 35 1270 METAL6 GROUND GROUP44_GROUND
```

The leakage resistance in the output CPM model appears as follows:

```
R0_A320 GROUP44_GROUND GROUP44_POWER 99660.431993
R0_D821 GROUP43_POWER GROUP43_GROUND 64882.488351
R0_N863 GROUP42_POWER GROUP42_GROUND 87442.938928
```

iCPM- Internal Node Probing

iCPM enables access to sensitive P/G connections in the design, and allows you to probe device locations inside the chip. This capability can evaluate the drop through the on-die PDN in a fast system-level simulation. You can not only see the impact of changes at the boundary between the die and the package (at the pads or C4 bumps), but also deep inside the chip at critical locations, such as near the PLL or the FGU blocks.

CPMs created iCPM technology have terminals at user-defined device locations, in addition to the traditional C4 bump and/or pad locations. The iCPM model is created in an incremental step after simulation:

```
perform powermodel -nx 1 -ny 1 -probe -o cpm_probe.sp
```

In the GSR, add the keyword 'PROBE_NODE_FILE <file_path>'. The file format is:

```
<X_location> <Y_location> <Metal_layer_name> <Port_name>
```

For example:

```
68.81 47.76 METAL1 port_vdd
67.25 40.10 METAL1 port_vss
```

Or you can use the options '-internal_node' and '-cell_file <instance_list_file>' to specify particular instances that should have probe access.

Resonance frequency-aware mode

In the Resonance Frequency Aware mode, the RedHawk VectorLess™ engine generates a multiple-cycle switching scenario for the current signature that introduces most of the energy around the chip-package-system resonance frequency. You only need to specify the system resonance frequency when creating the model, and CPM analysis automatically creates the frequency-based form of on-die excitation, while maintaining the logic and timing properties of the circuit, as shown in the Figure 14-8.

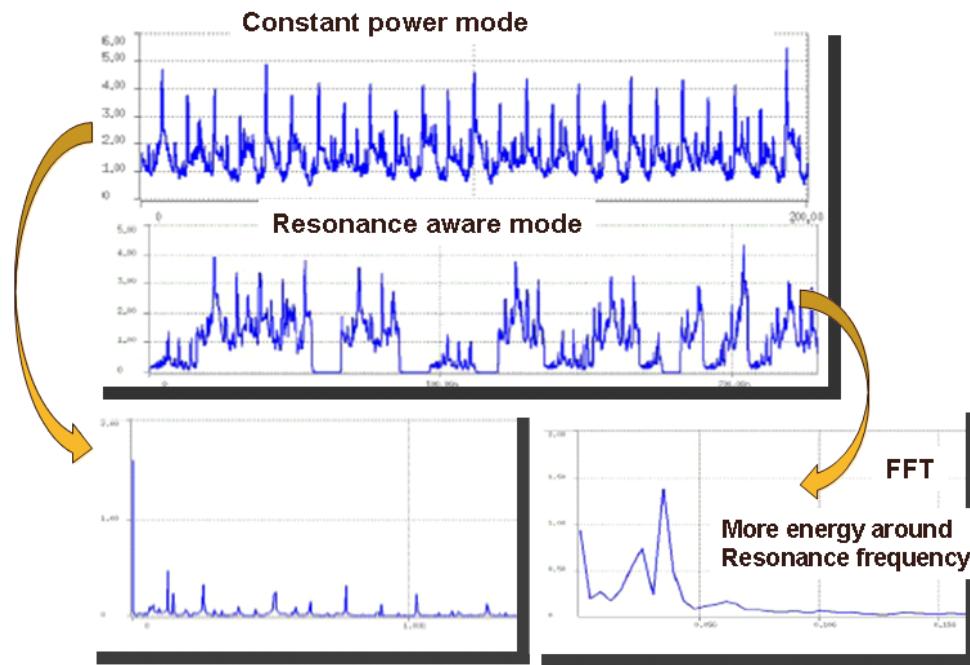


Figure 14-8 Frequency-aware CPM mode

To use this option, first generate a regular CPM model and use its passive part to perform AC analysis for the circuit that includes the CPM and package (and PCB if desired). This AC analysis yields the resonance frequency. Then a second CPM generation run is performed with the GSR setting: DYNAMIC_FREQUENCY_AWARE <frequency in Hz>

For example, for a resonance frequency of $f = 40$ MHz:

```
DYNAMIC_FREQUENCY_AWARE 40e6
```

Power Transient Mode (variable power)

Power Transient Analysis, also known as Variable Power, provides greater flexibility to simulate frame-by-frame power/current transient behavior, such as for Gated Clock operations, and more accurately models power transients on the chip.

While the resonance-aware mode generates current transients that match a particular resonance frequency, variable power mode has more flexibility to include a range of frequencies. By capturing the power stepup, the impact of various chip transitions--such as from reset to running mode, from traffic to no-traffic mode, or from memory-on to memory-off mode--can be modeled, as shown in Figure 14-9. These power transients affect the entire VRM, board, package, and die system to varying degrees, based on the duration and amplitude of the power step. The variable power CPM also can be used to create a low-frequency spectrum, as opposed to a traditional high-frequency spectrum, to allow testing of the package and PCB in the low-to-middle frequency range.

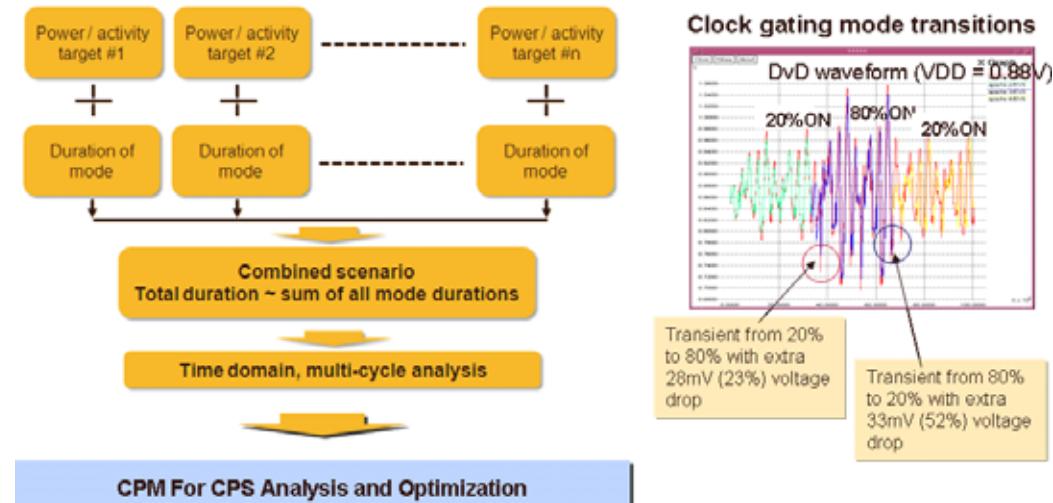


Figure 14-9 CPM Power Transient mode

CPM gets the Power Transient Analysis from the **RedHawk** dynamic run. The following GSR keyword are used to specify its operation:

```

POWER_TRANSIENT_ANALYSIS {
    <duration_frame_1_sec> <config_file_1>
    <duration_frame_2_sec> <config_file_2>
    <duration_frame_3_sec> <config_file_3>
    ...
}
  
```

Power transient configuration file

The CPM configuration file supports the following GSR keywords:

- GSC_FILE
- INSTANCE_POWER_FILE
- GSC_OVERRIDE_IPF
- BLOCK_POWER_FOR_SCALING (and BLOCK_POWER_FOR_SCALING_FILE)
- STATE_PROPAGATION {
 - PROPAGATION_MODE
 - GATED_ON_PERCENTAGE
 - CONSTRAINT_FILE
 - GATED_CONTROL_FILE
}
- TOGGLE_RATE_RATIO_COMB_FF
- INSTANCE_TOGGLE_RATE (and INSTANCE_TOGGLE_RATE_FILE)
- BLOCK_TOGGLE_RATE (and BLOCK_TOGGLE_RATE_FILE)
- TOGGLE_RATE

Settings in the configuration file versus the GSR file

The configuration keyword, if defined, overrides the same keyword in the GSR. The GSR keyword is used if the configuration file keyword is not defined, except for

- GSC_FILE
- STATE_PROPAGATION

which are ignored in the GSR if the Config file keyword is not defined.

Note that at present Power Transient Analysis works only in the Vectorless mode, and it does not support keywords other than the list above. It does not support:

- (a) VCD_FILE or BLOCK_VCD_FILE
- (b) STA_FILE or BLOCK_STA_FILE
- (c) BLOCK_POWER_ASSIGNMENT
- (d) PAR or BLOCK_PAR

Also note that simulation uses charge from the first frame for its simulation reference, so the dynamic simulation result may vary depending on the first frame config setting.

User-configurable mode

The user-configurable mode enables package and board designers to divide and customize a CPM for multiple individual system simulations, each targeting a specific operating mode or area of the chip, without having to regenerate the model for each single operating mode. The user-configurable CPM allows package and board engineers to mix and match contributions from different user-specified regions, or functional blocks of the die, creating unique scenarios that reflect different operating states of the chip. For example, an analysis of EMI from different blocks on the chip could be performed separately with one run.

First, you must partition the chip into groups of functions by hierarchical blocks and instances. Each group is represented by its unique current signature that can be enabled or disabled independently of the current signatures of other groups. The current signature sources corresponding to different groups are connected in parallel to each other, with sources belonging to each group marked by SPICE comments to activate one or more groups in a particular SPICE simulation, as shown in the Figure 14-10. Note that the sum of currents of all groups is equal to the current of the original CPM model. The passive part is also identical to that of the original CPM model.

The partitioning of the IC into multiple groups is achieved by specifying a list of blocks and instances that belong to each group in a “group file.” The resulting CPM model can be constructed for a vectorless or a VCD case.

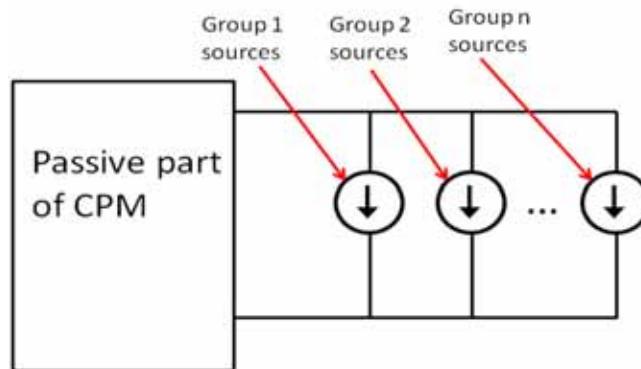


Figure 14-10 Creating a user-configurable CPM

To use the user-configurable feature, the following options of the 'perform powermodel' command are used:

- user_config : enables the user-configurable feature (note that it does nothing specifically involving emi)
- group_file <file name>: specifies the name of the file used to define the partitioning of the chip into groups of hierarchical blocks and instances. A full or relative path to the file is allowed, but tilde '~' and environment variables are not allowed. The purpose of the group file is to assign switching instances (in the apache.scenario file) to groups. Note that instances can be grouped in hierarchical blocks, and the separator for blocks is slash ('/'). Users can assign a whole hierarchical block, a sub-block, or an individual instance to a group. Any instances that do not belong to any user-defined group are assigned to group Ng+1 (Ng = number of groups), with the name "others".

The format of the group file is as follows:

```
GROUP <group name1>
INSTANCE <inst_name_1A> ... ;
GROUP <group name2>
INSTANCE <inst_name_2A> ... ;
...
...
```

The keywords are case-sensitive, as are instance names. Wildcard names are supported, and lines starting with '#' are considered comments.

Example group file:

```
GROUP GS1
INSTANCE Bl_1
Bl_2/Inst1
Inst2
Bl_3/MEM* ;
...
...
```

In this example, 'Bl_1' could be a hierarchical block name, 'Bl_2/Inst1' could be an instance name (or level 2 hierarchical block name), and 'Inst 2' may be an individual instance name. The syntax does not specify this. So group 1 contains all instances whose full name starts with "Bl_1", followed either by '/' or white space, and also "Bl_2/Inst1" followed either by '/' or white space, and "Inst2" followed either by '/' or white space. If P = number of CPM ports, the resulting CPM model has $(P-1)*Ng$ current signature PWL sources without the '-pincurrent' option, or $P*Ng$ current sources with the '-pincurrent' option. The PWL sources belonging to the same group have names starting with the group name, such as "lcsig1_1", "lcsig1_2", for a group named "csg1", and so on. This allows third party tools to easily enable or disable sources belonging to a particular group.

User-configurable CPMs can easily be applied to power delivery analysis because the current contribution from different portions of the chip has impact on voltage drops. For example, the worst voltage drop can be observed and optimized by turning off the current contribution of certain blocks in the chip. The source of noise can be identified using this approach, where different die activity scenarios are explored, to understand the block-level contribution to the overall system noise. Results for a sample design are shown in Figure 14-11 and Figure 14-12.

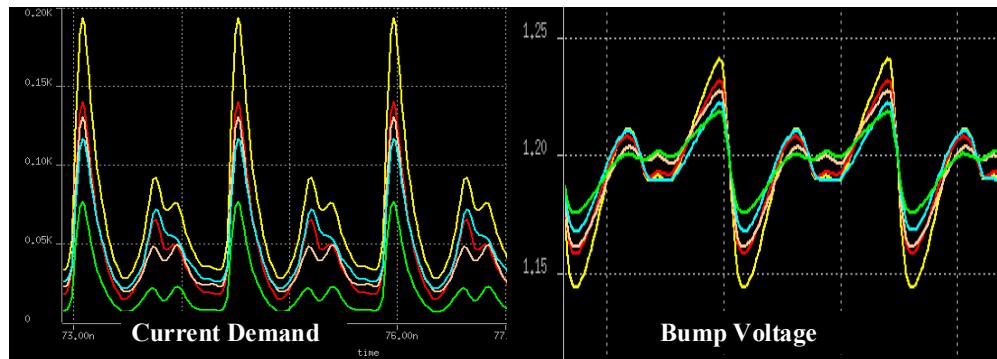


Figure 14-11 Analysis of multiple group current/voltages

Block 1	Block 2	Rest of Chip	Peak Current	Peak-to-Peak DvD
ON	ON	ON	192A	95mV
OFF	ON	ON	139A	72mV
ON	OFF	ON	130A	66mV
ON	ON	OFF	116A	53mV
OFF	OFF	ON	76A	42mV

Figure 14-12 Example CPM analysis results

CPM LDO analysis support

CPM models can be generated for chips with LDO features using the 'perform powermodel' command with no additional options. The number of terminals created corresponds to the behavioral LDO models that APLDO generates. The presence of LDO features is detected automatically. The CPM option '-plocname' is always enabled if LDOs are present. CPM generates additional internal ports at the pins (terminals) of the LDO models. The main CPM subcircuit (.SUBCKT adsPowerModel) has terminals only for the CPM ports at the pads, the same as it would without the LDO models. The CPM contains comments indicating the name of the LDO instance and the name of the SPICE subcircuit instantiated for the LDO instance. For example:

```
* ***** CPM-LDO ***
* LDO cell | SPICE subcircuit name
* -----
* VDD_REG_FC_ISO_580173      VDD_REG_FC_ISO
* -----
* VDDCO_REG_FC_ISO_580174     VDDCO_REG_FC_ISO
* -----
```

If LDOs are present, the top level CPM subcircuit *adsPowerModel* instantiates the LDO subcircuits with one per LDO instance present. For example:

```
**** Instantiation of the LDO subcircuits : ****
X_VDD_REG_FC_ISO_580173 VDD_REG_FC_ISO_580173_vddhv
VDD_REG_FC_ISO_580173_vddcore0
+ VDD_REG_FC_ISO_580173_vddcore1 VDD_REG_FC_ISO_580173_vss
```

```
VDD_REG_FC_ISO  
X_VDDCO_REG_FC_ISO_580174 VDDCO_REG_FC_ISO_580174_vddhvcore  
    VDDCO_REG_FC_ISO_580174_vddcore0  
+ VDDCO_REG_FC_ISO_580174_vddcore1 VDDCO_REG_FC_ISO_580174_vss  
    VDDCO_REG_FC_ISO  
*****
```

To perform system-level simulation with CPM, you must create a SPICE netlist that includes the models for package/PCB and supply sources. If LDOs are present, you must also provide the LDO subcircuits, which can be either transistor-level SPICE models or behavioral models. Transistor-level LDO models provide the greatest accuracy, while the behavioral models may have better simulation speed. If transistor-level LDO models are used, they typically have additional terminals, such as for the bias pins. They must be “wrapped”, along with the bias voltage sources and other required circuitry, to generate the subcircuit that is invoked by the CPM.

For best accuracy, the following two options of the 'perform powermodel' command are recommended: -pincurrent and -global_gnd.

Notes on using the LDO methodology:

1. No LDO subcircuits are instantiated in the binary CPM (Sentinel-PI format).
2. The CPM header includes coordinates and layer information for the LDO subcircuit terminals.
3. LDO can be used with the CPM '-probe' option to create additional CPM ports at internal nodes, but not with the '-internal_node' or '-cell' options of the 'perform powermodel' command. Example:

```
perform powermodel -wirebond -global_gnd -pincurrent  
-o ABCD_ldo_de BZ.sp -probe
```

4. The -probe option requires that the PROBE_NODE_FILE GSR keyword be used, as shown below:

```
PROBE_NODE_FILE <file_path>
```

Format of the probe node file:

```
<x coordinate> <y coordinate> <layer> <probe name>  
...
```

5. It is possible to have a probe node exactly at the location of the LDO terminal. In this case, the probe node becomes one CPM port and it becomes exposed. Normally, the terminals of the LDO subcircuits are not exposed.

CPM Outputs

The outputs from CPM processing are the following:

- CPM model files
- *get_cdie.sp* NSpice netlist file
- *.cdie file, effective die series resistance and capacitance values for '1 X 1' partition models
- differential output voltages from S-parameter models

These outputs are described in the following paragraphs.

CPM Model Files

The CPM model has several elements: the passive part of the CPM model (`*.sp.inc`), and the current signatures by partition (`*.sp`), as shown in Figure 14-13. The output of RedHawk CPM simulation is a hierarchical SPICE netlist that includes the switching current signature, as well as the reduced-order model of the die power delivery network. The complexity of the SPICE netlist is determined by the number of partitions over which the CPM model is created.

Three types of models are generated simultaneously during the run, and written to the `adsRpt/CPM` directory:

- standard SPICE (default)
- CPM optimized for HSPICE (`adsRpt/CPM/<design_name>.hspice.sp`). For designs with a large number of ports (100+), the use of HSPICE-specific elements (Foster VCCS) is important, to provide a way to eliminate internal nodes, improve simulation speed and reduce memory consumption.
- Sentinel-PI CPM (`adsRpt/CPM/<design_name>.snpi.sp`), which is the same as the HSPICE format, but in binary form.

Hierarchical Structure of CPM Output File

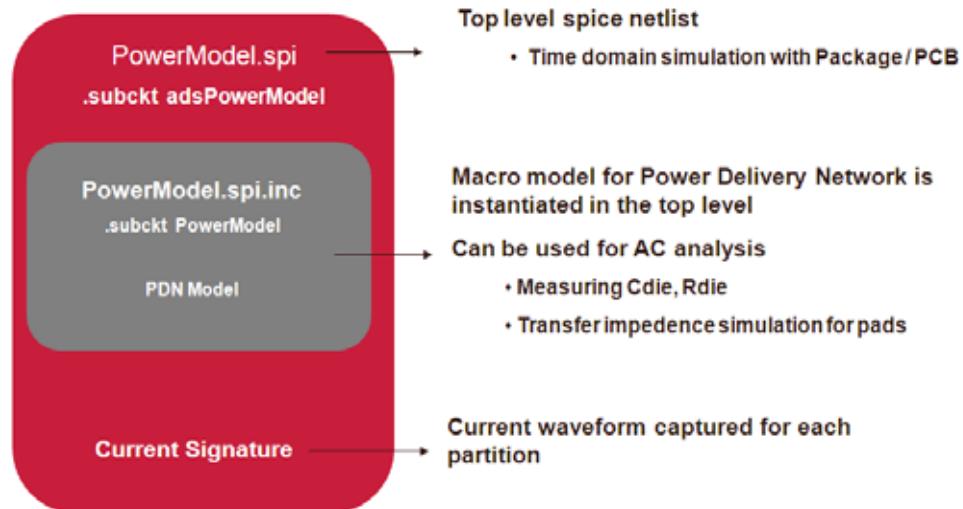


Figure 14-13 CPM model elements

get_cdie.sp File

The NSPICE netlist is exported to the file `get_cdie.sp`, which can be used to calculate R_{die}/C_{die} at the specified frequency (50 MHz by default, unless you modify it in the `get_cdie.sp` file), and to plot $C_{die}(f)$ and $R_{die}(f)$ for frequencies ranging from 1 MHz to 1 GHz. This file eliminates user errors, which are likely for a CPM model with a large number of ports, and also saves time that would be required to create the netlist by hand.

For designs with multiple power/ground nets RedHawk generates a separate `get_cdie*.sp` file for each pair of nets. If there are only two nets, the file name is still `get_cdie.sp`. Otherwise file names of the form `get_cdie_<pwr_net>_<gnd_net>.sp` are written to the working directory.

*.cdie File

If you run CPM simulation with the 1×1 partition option using ‘-nx 1 -ny 1’ (even for a wirebond design), or the -cdie option, a *.Cdie file is generated in the <workdir>/adsRpt/CPM/apache.Cdie directory. The *.Cdie file contains the effective capacitance, as well as the effective series resistance, of the die PDN at various frequency points. This is useful as a quick ballpark assessment of die parasitics.

The following is an example of a *.Cdie file:

```
Cdie and Rdie between nets VDD and VSS
4.700000e+08 Hz, Cdie=2.662143e-11 F, Rdie=2.902836e+00 Ohm
6.250000e+08 Hz, Cdie=2.635451e-11 F, Rdie=2.874828e+00 Ohm
9.350000e+08 Hz, Cdie=2.571951e-11 F, Rdie=2.811440e+00 Ohm
1.250000e+09 Hz, Cdie=2.501593e-11 F, Rdie=2.746562e+00 Ohm
1.875000e+09 Hz, Cdie=2.362147e-11 F, Rdie=2.634761e+00 Ohm
2.500000e+09 Hz, Cdie=2.229990e-11 F, Rdie=2.548309e+00 Ohm
```

Note that the *.Cdie file reports results only at the frequencies at which actual calculation of the Y matrix has been performed, and no approximations are involved in calculating the Rdie and Cdie (an exact calculation that does not rely on the equivalent circuit).

For AC analysis, you should connect to the parasitics part only (.inc file), and not connect the CPM port to global ground (Spice node 0). For current signature, by default, CPM uses the Norton current model. It picks one of the ground ports as the negative terminal for every PWL current source. The sum of all port currents will be zero. If you generate a CPM with the option ‘-pincurrent’ it does not use the Norton current model. Every PWL is referenced to node 0. Note that some tools do not support non-Norton current models.

The CPM is described in hierarchical subcircuit-based SPICE netlists. The main CPM Spice deck includes piecewise linear switching current signatures, which are connected to the parasitic model of the die power delivery network, which is in turn described in a second SPICE netlist file (.inc file). A two-port equivalent model is shown in Figure 14-14.

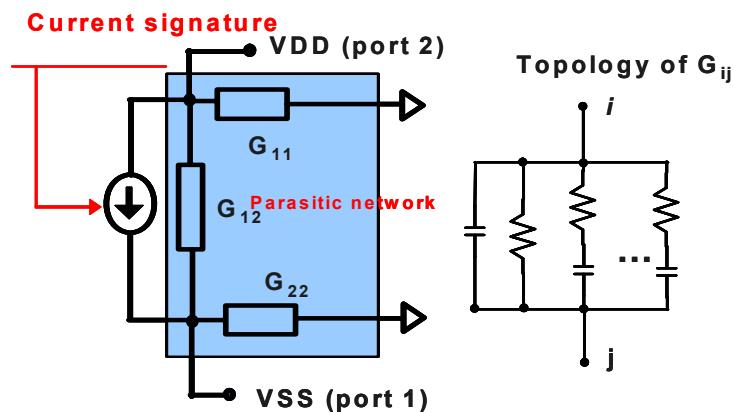


Figure 14-14 Two-port CPM equivalent circuit

The following is a simple CPM generated for a die (one Vdd and one Vss net) using a 1×1 partition. The Chip Package Protocol section lists the name of the die pad, x and y location of the pad, the corresponding SPICE node name, as well as the partition and net information of the pad.

```
*****
* Apache RedHawk Chip Power Model [ Ver 1.00 ]
* @Apache Design Solutions 2002 - 2006
* Presimulation time 3500.000000ps
*****  

.  

.INCLUDE "PowerModel.sp.inc"  

* Begin Chip Package Protocol --->
* die_area 0 0 293.04 313.17
* Vdd_130564_1645 : (65.000000 0.000000) : p1 = PAR_0_0_VDD
* vdd_326658_1645 : (163.000000 0.000000) : p1 = PAR_0_0_VDD
* Vdd_526412_2605 : (263.000000 0.000000) : p1 = PAR_0_0_VDD
* Vdd_527135_6258365 : (264.000000 313.000000) : p1 = PAR_0_0_VDD
* Vdd_327130_6258415 : (164.000000 313.000000) : p1 = PAR_0_0_VDD
* Vdd_126096_6259955 : (63.000000 313.000000) : p1 = PAR_0_0_VDD  

..... * End Chip Package Protocol <---  

.subckt adsPowerModel
+ p1 p2  

Xpdn
+ p1 p2
+ PowerModel1  

Icurlsig1 p1 p2 pw1(
+ 0.000000ps 0.000178
+ 10.000000ps 0.000657
+ 20.000000ps 0.005198
+ 30.000000ps 0.007278
....
```

Note that the *PowerModel.sp* file reports the presimulation time in the header. This is useful for CPM-to-**RedHawk** correlation, as you must shift the **RedHawk** waveform by the presimulation time.

The *PowerModel.sp.inc* file, generated with AC analysis CPM, has the following header (note “Accurate RC reduction”):

```
*****
* Apache RedHawk Chip Power Model [Accurate RC reduction]
* Model Subcircuit of Die PDN
* @Apache Design Solutions 2007
*****
```

Using the Chip Power Model

You can attach the SPICE netlist created from CPM to the package/PCB models and simulate those using NSPICE or any other SPICE compatible simulator.

Some of the key applications of Chip Power Model technology are:

- Determining global Power Delivery Network impedance, identifying IC-package

resonance using AC analysis (use the parasitic part only).

- Performing dynamic voltage noise budgeting at board and package level
- Package and board decap optimization

Differential Voltage Waveforms

Since designers typically want to look only at differential voltages to review results, S-parameter models used for RedHawk package-aware analysis are differential in nature. For this reason the wire voltage drop summary in *redhawk.log* is disabled, and differential voltage waveform probing at the pads is enabled, using the TCL command

```
plot voltage -pad_pair {<vdd_pad> <vss_pad>} ?-sv? ?-o <output_file> ?
```

Example command:

```
plot voltage -pad_pair {VDD1 VSS2} -sv -o vdrop.txt
```

Validating the Model

One type of accuracy validation of the Chip Power Model involves the following steps:

1. Run RedHawk without the package model to generate the CPM. The accuracy improves if the time step is reduced. A time step of 10ps provides the best accuracy (there is no need to use a smaller one). To change the time step, add the following to the GSR file:

```
DYNAMIC_TIME_STEP 10e-12
```

The best accuracy is achieved if you let the presimulation time step be the same as during simulation. However, the time step can be changed as follows (see section "DYNAMIC_PRESIM_TIME", page C-702, for details):

```
DYNAMIC_PRESIM_TIME [<presim_time_ps>| -1] ?<TSM> ?  
?<TSM_fraction>?
```

The time step during the presimulation time is TSM*TS, where TS is the time step specified in the DYNAMIC_TIME_STEP GSR keyword. For example,

```
DYNAMIC_PRESIM_TIME 2e-9 1
```

The value of 1 sets the presimulation time step at the normal length.

2. Attach a package model to the CPM model and run using SPICE.
3. Measure the current and the voltage at the connection points (die pads) of the CPM model to the package model.
4. Attach the same package model to RedHawk and run DvD analysis (with the same pre-simulation time as used for CPM generation). Remember to shift the RedHawk waveform by the pre-simulation time, which is prior to t=0 in RedHawk and after it in Spice. The presimulation time is reported in the header of the CPM output file.
5. Measure the current profile for all the bumps and the VDD/GND waveform at any bump.
6. Compare the two waveforms.

Figure 14-15 shows waveforms from the measurement of current from one test-case using the above procedure. The red current profile is from measuring the bump current in

a RedHawk run, while the yellow current profile is from measuring the bump current from the SPICE run with the CPM model.

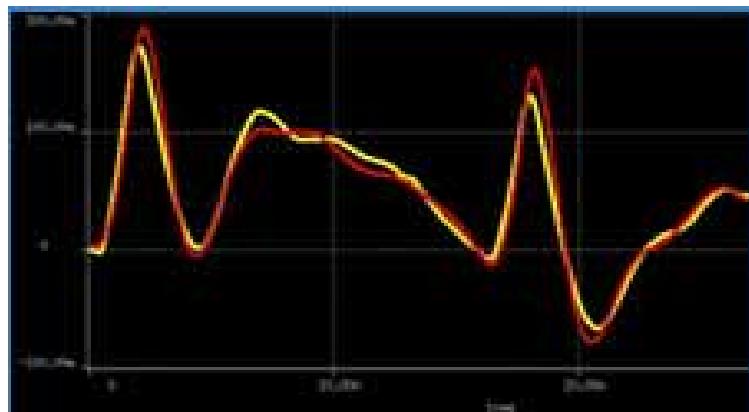


Figure 14-15 Comparison of CPM model with full RedHawk chip simulation.

Chapter 15

Reliability and EM Analysis

Introduction

Electromigration (EM) is the movement of material that results from the transfer of momentum between electrons and metal atoms under an applied electric field. This momentum transfer causes the metal atoms to be displaced from their original positions. This effect increases with increasing current density in a wire, and at higher temperatures the momentum transfer becomes more severe. Thus in sub-100nm designs, with higher device currents, narrower wires, and increasing on-die temperatures, the reliability of interconnects and their possible degradation from EM is a serious concern.

The transfer of metal ions over time from EM can lead to either narrowing or hillocks (bumps) in the wires. Narrowing of the wire can result in degradation of performance, or in extreme cases can result in the complete opening of the conduction path. Widening and bumps in the wire can result in shorts to neighboring wires, especially if they are routed at the minimum pitch in the newer technologies.

Foundries typically specify the maximum amount of current that can flow through a wire under varying conditions. These EM limits depend on several design parameters, such as wire topology, width, and metal density. EM degradation and EM limits depend on the temperature at which interconnects operate, as well as on the material properties of the wires and vias, on the direction of current flow in the wire, and on the distance of the wire segment from the driver(s). EM current characteristics are defined in Figure 15-1.

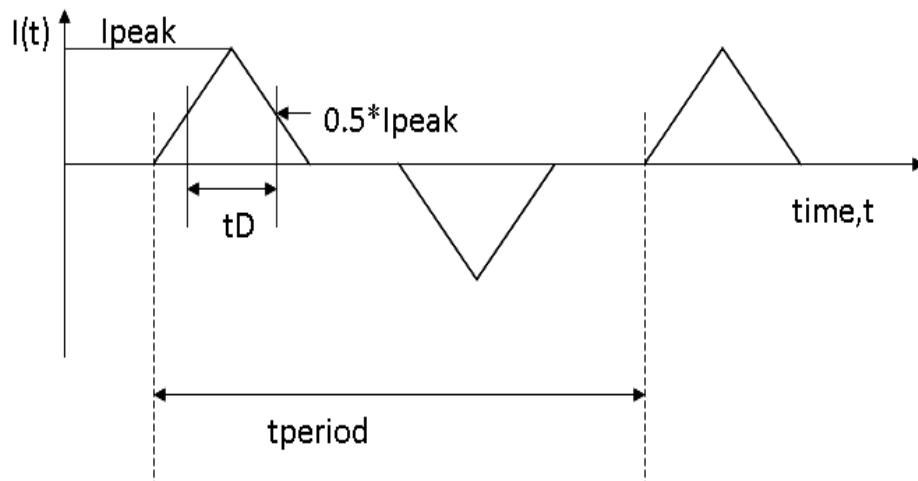


Figure 15-1 EM current definitions

Key EM current characteristics are defined as follows, where $t_p = t_{\text{period}}$:

$$R = \frac{t_D}{t_p}$$

$$I_{\text{peak}} = \max|I(t)|$$

$$I_{avg} = \int \frac{I(t)dt}{t_p} \text{ for P/G and Signal current with an APL-simulated waveform.}$$

$$I_{avg} = \int |I(t)dt|/t_p \text{ for Signal current with an estimated waveform.}$$

$$I_{rms} = \sqrt{\int I(t)^2 dt / t_p}$$

One common EM check employed is to measure the average or DC current density flowing through a wire and compare it against foundry-specified limits. The impact of average or DC current in wires in a design is typically quantified using Black's equation, which is used to measure and compare the Mean-Time-To-Failure for interconnects with different parameters, such as the average current density, temperature, and activation energy. Another common check employed is to measure the peak and RMS current flowing through interconnects and check them against foundry-specified targets. These checks are to ensure that metal failures do not occur because of Joule or self-heating in the wires.

RedHawk™ provides a single platform approach in which to analyze EM of both power grid and signal interconnects in a design. Power EM analysis is performed as an integral part of static and/or dynamic analysis. Signal EM analysis, which is performed in a separate run, checks for average (uni-directional or bi-directional), RMS, and peak current densities in all signal wires and vias in a design.

Temperature Setting for Power EM Calculation

Operating temperature for Static and Dynamic EM calculation can be set independently of the temperature value used for analysis, if desired, using the required tech file keyword TNOM_EM and the required GSR keywords TEMPERATURES_EM and TEMPERATURE_EM. The associated power analysis temperature keywords are TNOM in the tech file and TEMPERATURES and TEMPERATURE in the GSR. These temperature keywords affecting EM calculation are described below.

In the Tech file, specify the nominal and final temperatures with the syntax:

```
metal <metal_layer_name> {
    EM <max_wire_current_density>
    TNOM_EM <EM_nom_temp>
    ? T_EM <EM_final_temp> ?
    TNOM <Power_nom_temp>
    ? T <Power_oper_temp>?
    ...
}
```

In the GSR, use the following keywords for EM calculation:

```
TEMPERATURES_EM {
    <layer_1> <temp_1 °C>
    ...
    <layer_n> <temp_n °C>
}
, or
TEMPERATURE_EM <temp °C>
```

The associated extraction GSR keywords for temperature are:

```
TEMPERATURES {
```

```

<layer_1> <temp_1 °C>
...
<layer_n> <temp_n °C>
}
      or
TEMPERATURE <temp °C>
```

Then for EM calculation, the priorities for temperature-setting keywords are:

1. TEMPERATURES_EM - layer-based, in GSR
2. TEMPERATURE_EM - global, in GSR
3. T_EM - layer-based, in the tech file
4. TEMPERATURES - layer-based, in GSR, for extraction
5. TEMPERATURE - global, in GSR, for extraction
6. T - in the tech file, for extraction
7. TNOM_EM - global, in tech file

RedHawk records the temperatures used in extraction and in EM checking in the *redhawk.log* file.

RedHawk Methodology for Static Power EM Analysis

RedHawk™ automatically performs power EM analysis based on currents obtained as part of static or dynamic voltage drop analysis. The primary difference is that static EM analysis is based on true average current values, which is covered in this section, whereas dynamic power EM analysis is based on either peak, true average or RMS current values (see [section "Methodology for Dynamic Power EM Analysis", page 15-422](#)).

There is no separate command required for performing static EM analysis. It is performed by default if you have EM limits specified in the technology file. In electro-migration analysis, **RedHawk** checks the actual current density for METAL wires, or current per cut or area for a VIA segments, against the EM limit specified in the technology file. The setup for static power EM analysis involves defining the desired EM limits in the tech file, which can be done in several different ways, as described in the following section.

Setting Up EM Limits

The simplest EM limit is specified per layer, which defines the allowed current density value for a specific METAL or VIA layer. For a metal layer, the current density limit is defined as the current flowing per unit width. It can be specified in the tech file, as in the following example:

```

metal METAL1
{
    Thickness     1.45
    Resistance   0.2343
    EM           2.7
    above        PASS4
}
```

In the above example the current density limit for layer METAL1 is defined as 2.7. The unit for current density comes from the units for length and current in the 'Units' section of the technology file, as in the following example:

```
units {
```

```

capacitance 1p
resistance 1
length 1u
current 1m
voltage 1
time 1n
frequency 1me
}
```

For VIA layers, the EM limit is defined by layer on a per-cut basis: the allowed current per cut of the via or via array, as in the following example:

```

via VIA78
{
    Area      { 9 }
    Resistance 0.041
    T          25
    Tnom       110
    Coeff_RT1 0.00337
    Coeff_RT2 -7.91e-7
    EM         7
    UpperLayer METAL8
    LowerLayer METAL7
}
```

There are more advanced methods for specifying the EM limits in the tech file, using the following options of the 'metal' tech file keyword:

```

EM_Adjust
Width_Based_EM
Blech_JLC
EM_Temp_Rating
Thom_EM
```

For a description of how to use these options to define metal and via EM limits, see the section "["METAL"](#)", page [C-547](#), of section "[Apache Technology File \(*.tech\)](#)", page [C-544](#).

If you are creating the tech file using the Apache utility 'rhtech', you can use the option '-e <EM_FILE>' to populate the tech file with two of the EM-related keywords, or to read a user-specified polynomial-based EM rule file, use the option '-pe <file>', instead of '-e' for simple EM rules. Only the EM and EM_ADJUST options can be used inside this rule file; you should include the other EM-related keywords manually in the tech file. The following is the syntax for using the 'rhtech' utility:

```

rhtech -i <input_file> [-o <output_file>]
[-m <layer_mapping_file>]
[-e <EM_file>] [-t <temp_file>]
```

Following is an example of an EM rule file for passing EM tech file keywords:

```

## <layer_in_itf/nxtgrd> <EM_limit-mA/um for metal; mA for vias> <EM adjust>
metals5 0.9308 0.02
metals4 0.9308 0.02
metals3 0.9308 0.02
metals2 0.9308 0.02
metals1 0.716 0.02
via4 0.06766
via3 0.06766
via2 0.06766
```

vial 0.0676

Running Power EM Analysis

For procedures for setting up and running power EM analysis, see [section "Running RedHawk-S \(Static IR/EM Analysis\)", page 4-57](#). RedHawk by default does not perform EM analysis as a part of static and dynamic simulation. However, you can set the GSR keyword 'ENABLE_AUTO_EM 1' to perform the EM check during regular post simulation processing. Then the TCL command 'perform emcheck' can be used to perform power EM analysis for specific modes and nets, using the command:

```
perform emcheck ?-mode [AVG | RMS | PEAK | all]?
?-net <netname>?
```

The default mode is 'all'.

Analyzing Static EM Analysis Results

Once the simulation is performed you can click on the **EM** button in the **View Results** panel to see the EM violations map, such as shown in Figure 15-2, which displays the EM violations in different ranges in different colors. RedHawk highlights all metal or via segments with current density or current per cut exceeding the specified EM limit (default) in red, and lesser fractions of the limit in other colors.

You can change the EM ranges and their color display using the ElectroMigration Color Map dialog, available by clicking on the 'Set Color Range' button in 'Configuration' panel in the GUI.

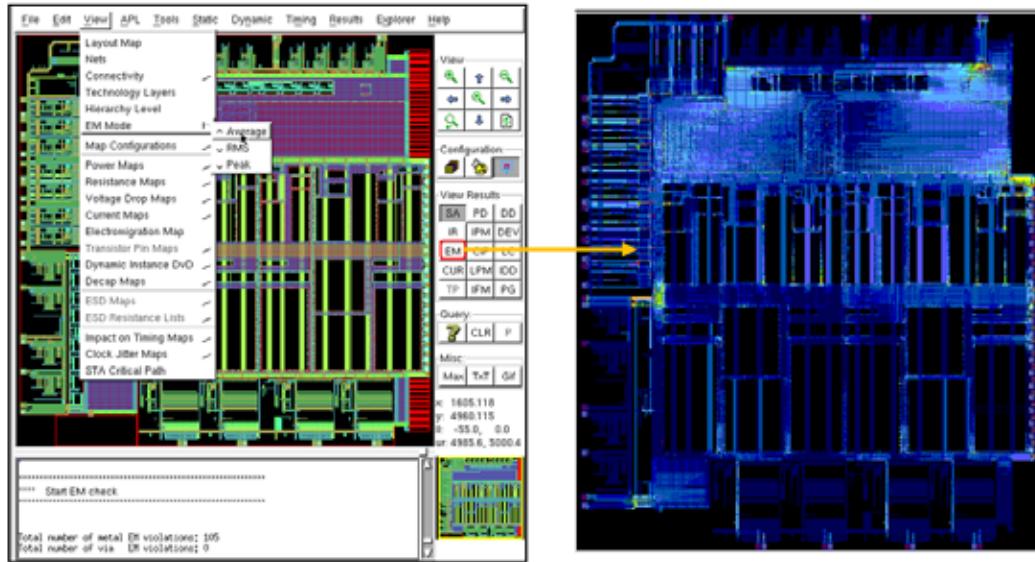


Figure 15-2 EM results display

After static analysis, EM violations are also listed in the report *adsRpt/Static/<design_name>.em.worst*, which lists the details of all METAL or VIA segments with average current density or current exceeding the default EM limit of 100%, in decreasing order. You can change the default limit using the GSR keyword, **EM_REPORT_PERCENTAGE**. By default, RedHawk dumps only the top 1000 violations. You can increase this number using the GSR keyword **EM_REPORT_LINE_NUMBER**. If you want to dump EM violations in a specific range, you can use the GSR keyword

EM_DUMP_PERCENTAGE. For example, if you specify “EM_DUMP_PERCENTAGE 50 60” in the GSR, it dumps all violations having an EM ratio between 50% and 60% in the file *adsRpt/Static/<design_name>.em*. This file is not in sorted order and is not generated unless you specify the above keyword in GSR.

In the EM report file RedHawk reports all METAL EM violations using the following format.

#Layer	#End-to-end_coordinates	#EM_Ratio	#Net	#Width
METAL4	(4905.670,3398.849 4905.670,3400.562)	469.016%	VDD	25.000

This report is also available using the GUI menu command **Results -> List of Worst EM**. In this GUI display list you can zoom in to any violation using the **Go To Location** button. Then click on the geometry to see more details on the violation in the log window. This information can be used for calculating the EM ratio of this segment. Actual current density is calculated by dividing the current by the effective wire width. If you specify the EM_ADJUST parameter in the tech file, RedHawk subtracts this value from the actual width to get the effective width.

Methodology for Dynamic Power EM Analysis

Dynamic power EM analysis is similar to static EM analysis. In static analysis, the EM analysis is based on true average current density for wires and current per cut (or area) for vias. In dynamic analysis, RedHawk can perform EM wire/via analysis based on three different types of currents: peak, RMS or average. The GSR keyword EM_MODE is used to select the mode for the current analysis. By default, RedHawk does not perform an EM analysis on the specified EM_MODE (Avg, RMS or Peak). You can set the GSR keyword ENABLE_AUTO_EM to 1 to enable EM checking as part of simulation. You can then use the TCL command 'perform emcheck' to perform EM analysis for specific modes and nets.

Setting Up EM Limits

You can set up simple EM limits by specifying EM limit values for each layer in the tech file. If these are the only EM limits specified, they are used for all EM_MODEs. You can set up separate EM rules for each EM_MODE, in three different ways:

1. POLYNOMIAL_BASED_EM rules have a format that specifies the target EM_MODE as part of the keyword. If you specify a POLYNOMIAL_BASED_EM_PEAK rule in your base tech file, that rule will only be applied to a Peak EM analysis. POLYNOMIAL_BASED_EM_AVG and POLYNOMIAL_BASED_EM_RMS keywords are also accepted.
2. Separate EM rule tech files. Another way to specify separate EM rules for each mode is to supply those rules in separate EM rule tech files. These files are identified in the GSR file:

```
EM_TECH_DC <DC_em_rule_file>
EM_TECH_RMS <RMS_em_rule_file>
EM_TECH_PEAK <Peak_em_rule_file>
```

The basic structure and syntax for these files are similar to the base tech file. They accept a section for each 'metal' and 'via' layer, followed by EM rule specifications. The rules read in from the EM_TECH_DC file are applied to DC (average) EM analysis. Note that DC rules are also used for any Static EM analyses.

3. Rule sets. A more flexible and general way to specify EM rules is to group them together in named rule sets using the tech file keyword EM_RULE_SET. Rule sets allow you to read in any number of different sets of EM rules, and then selectively use them for different analyses during a RedHawk session. The format of an EM_RULE_SET file is similar to the EM rule files above. The first line has the

EM_RULE_SET keyword followed by the user-assigned name for the set. This is followed by 'metal' and 'via' sections containing EM rules, as in the example following:

```
EM_RULE_SET <user_rule_name1>
  metal <layer name> {
    <metal EM rules>
    ...
  }
  ...
  via <layer name> {
    <via EM rules>
    ...
  }
  ...

EM_RULE_SET <user_rule_name2>
...
```

To load the EM rule sets into **RedHawk**, identify the file using the GSR keyword '**EM_TECH_FILE <ruleset_file>**'. Once loaded into **RedHawk**, you can specify which **EM_RULE_SET** to use for each analysis with the following GSR command:

```
EM_RULE_SET [AVG|PEAK|RMS] <user_rule_name>
```

You also can change the EM rule set being used for an analysis with the TCL command:

```
gsr set EM_RULE_SET [AVG|PEAK|RMS] <new_user_rule_name>
```

and then rerun the 'perform emcheck' command with the new **EM_RULE_SET**.

Analyzing Dynamic Power EM Violations

After running dynamic analysis you can generate additional power EM reports based on different modes and specific net names using the TCL command 'perform emcheck', whose syntax is:

```
perform emcheck ?-mode [AVG|RMS|PEAK|all]?
?-net <netname>?
```

Skip the -mode and -net options to perform analysis for all modes and all nets in the design.

Dynamic EM analysis generates reports of the worst EM violations for each mode with the same filenames (*<design>.em.worst.peak*, **.avg*, and **.rms*) as in static analysis, located in the *adsRpt/Dynamic* directory. The report format is also the same. This is useful with several EM file limits as described in the previous section. In this case **RedHawk** automatically generates a detailed report for all the modes for which EM analysis has been performed. An example report generated after performing EM analysis for PEAK mode is shown below:

```
EM MODE is PEAK
# This file reports the EM violations, i.e. EM_Ratio =
Actual_Current_Density/Current_Density_Limit > 100% (default or from
"em_report_percentage") for wire pieces and vias in decreasing order. Unit
used for coordinates and dimensions is um.

# For wires: #layer #end-to-end_coordinates #EM_Ratio #net #width
# Blech_length
# For vias: #via_name #x-y_coordinates #EM_Ratio #net #blech_length
METAL3 (886.190,1018.712 887.192,1018.712) 1081.01% VDD 2.003 141.951
```

METAL3	(885.190,1020.695 886.190,1020.695)	1081.01%	VDD	2.000	141.951
METAL3	(2944.210,2479.555 2944.210,2480.055)	1024.24%	VDD	2.000	10.500
METAL3	(2942.980,2479.305 2942.980,2479.555)	1024.24%	VDD	0.500	10.500
METAL3	(1108.770,2479.555 1108.770,2480.055)	1023.81%	VDD	2.000	10.500
METAL3	(1107.540,2479.305 1107.540,2479.555)	1023.81%	VDD	0.500	10.500
...					

In the violation display in Figure 15-3, EM_ADJUST for METAL4 is defined as 0.016. So RedHawk calculates the actual current density as follows:

$$\text{Actual I density} = (\text{Current/Eff_Width}) = 219.664 / (25-0.016) = 8.7922 \text{ mA/u}$$

And the EM limit for METAL4 in the tech file is defined as 1.874 mA/u.

RedHawk calculates the EM_Ratio using the following equation.

$$\text{EM_Ratio} = 100 * (\text{Actual density of current}) / (\text{EM limit in tech file}).$$

Therefore

$$\text{EM_Ratio} = 100 * 8.7922 / 1.874 = 469.2$$

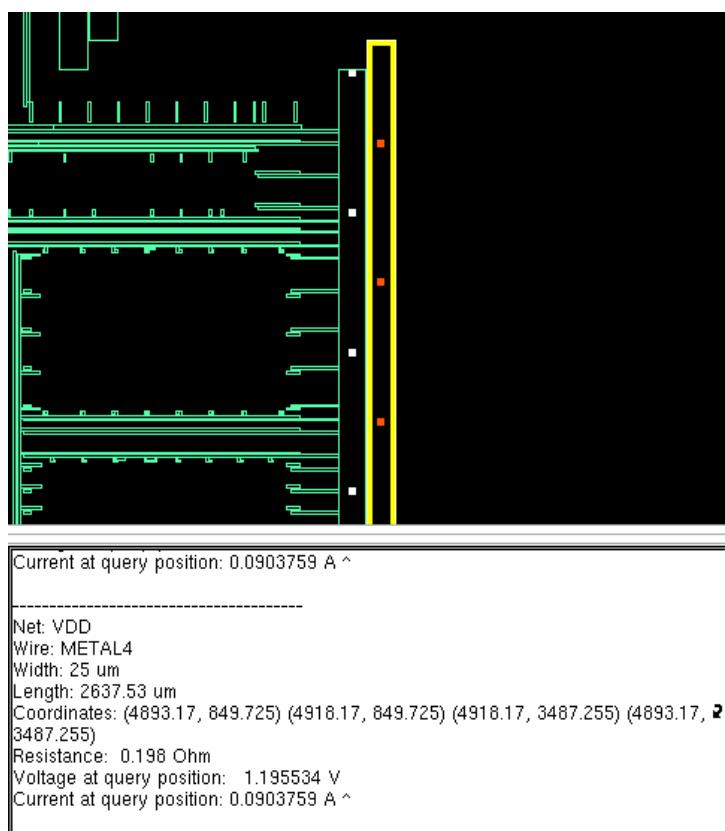


Figure 15-3 Investigating a potential metal EM violation

For VIAs, RedHawk reports the EM violations in the following format:

#Via_name	#x-y_coordinates	#EM_Ratio	#net
via via3Array_87	(3154.820,893.390)	172.29%	GND

The EM_ratio of a VIA layer is calculated as follows (percentage):

$$\text{EM_Ratio} = (\text{Current through the VIA}) * 100 / (\text{EM limit})$$

When you click on a via in the GUI, the **RedHawk** log window displays additional information that helps you in analyzing the violation, as shown in Figure 15-4.

```
Net: VSS
Via: via3Array_87
Bbox: (3153.8200,893.1400 3155.8200,893.6400)
Resistance: 0.127 Ohm
Average Voltage: 0.01951997 V
Bottom Voltage: 0.01970378 V
Top Voltage: 0.01933016 V
Current: 0.00294815 A (up)
EM percentage: 172.29% (EM limit: 0.00171114 A)
```

Figure 15-4 Investigating a potential via EM violation

In the example above, **RedHawk** reports both EM percentage and the EM limit value used in the calculation. The EM limit displayed is the limit calculated for the specific VIA array. As described earlier, inside the tech file Via EM limits are defined on a per-cut basis.

RedHawk calculates the EM limit for every via array in the design, considering the number of via cuts in the via array and total area of the cuts included. For example, if the via array has 10 cuts, the EM limit reported is (Tech file EM limit * 10).

Note that, depending up the type of EM rule selected, additional data relating to checks associated with that rule are also displayed.

Current direction and EM violations

To accurately analyze EM violations in a design, you need to understand the actual current directions. After EM analysis, when you click on any METAL or VIA geometry in **RedHawk**, the assumed current direction is indicated by a symbol in the log display window, along with the current value, as shown in Figure 15-5:

```
-----
Net: VDD
Wire: METAL4
Width: 25 um
Length: 2637.53 um
Coordinates: (4893.17, 849.725) (4918.17,
3487.255)
Resistance: 0.198 Ohm
Voltage at query position: 1.195522 V
Current at query position: 0.220086 A ^
```

Figure 15-5 Log display showing current direction

The symbol conventions followed for displaying the current direction are:

- > : For metals, current flow from left to right.
- < : For metals, current flow from right to left.
- ^ : For metals, current flow from bottom to top.
- V : For metals, current flow from top to bottom
- Up : For vias, current flow from lower metal layer to upper metal layer
- Down : For vias, current flow from upper metal layer to lower metal layer.

Using the correct current direction, you can identify the actual path for the current flow. You can also look at the Current map (**CUR** button in **View Results** GUI panel) to see any discrepancy in the current flow.

Fixing Power EM Violations

EM violations are mostly caused by weak power grid connections feeding current to high power-consuming regions or blocks in the design. If this is the case, increasing the metal width to reduce the current density is a typical solution. Similarly, for a via EM violation, you can increase the number of vias to fix potential EM issues. You also can provide additional straps for the current supply, thereby reducing the current-per-strap value. Layer switching is another option; typically, upper metal layers in the technology have higher current driving capability (due to greater thickness). So you can use these layers for designing the major power grids (grids with higher current flow) in the design.

You can use the what-if and Fix and Optimize capabilities (see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#)) in [RedHawk](#) to modify the grid for fixing EM violations. Using this method you can edit any existing strap or add any new straps or vias. [RedHawk](#) has incremental extraction capability, which helps you in analyzing the design quickly after making any modification.

The following example helps in understanding how you can resolve EM issues quickly using the “What-if” capability in [RedHawk](#). An example EM violation is caused by an insufficient number of vias between the power straps. More vias are added in this area using the 'Add Via' option in [RedHawk](#) to fix the problem, as shown in Figure 15-6 below:

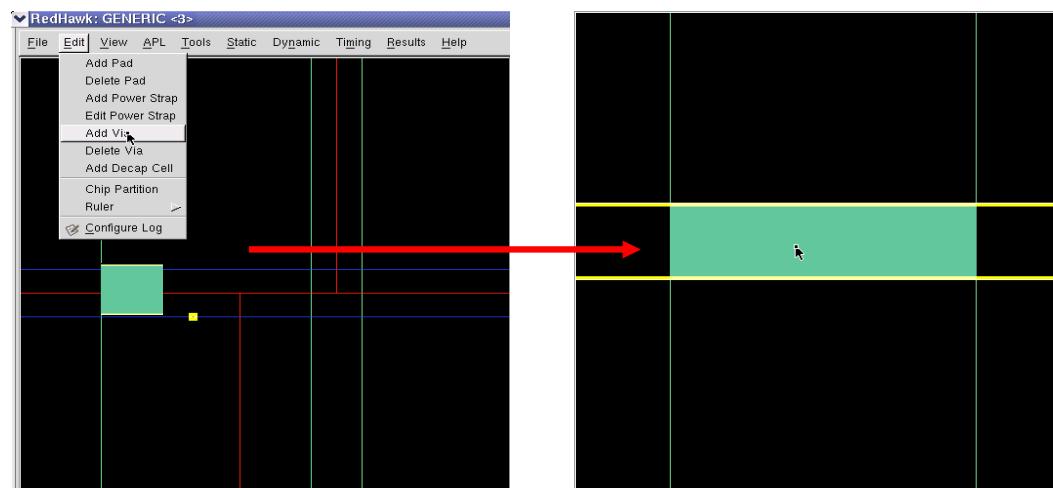


Figure 15-6 Using FAO to repair EM violations

Methodology for Signal EM Analysis

Signal EM analysis should estimate the I_{avg} , I_{rms} , and I_{peak} for every signal wire in the design, especially those belonging to the clock tree network, and compare them against their respective EM limits. The signal EM flow in [RedHawk](#) estimates these different current values based on cell design parameters. [RedHawk](#) analyzes EM for all signal wires in a design -- both inside cells (intra-cell) and between cells (inter-cell).

The average current flowing in every wire (I_{avg}) is estimated from the capacitive load seen at the output of each cell, its operating frequency, its supply voltage and toggle rate (which is defined as 2.0 or 200% for clock pins).

$$I_{avg} = f(\text{capacitive load, frequency, toggle rate, supply voltage})$$

Since the current flowing in a signal net is usually bi-directional, the true average current is usually very small. Therefore in [RedHawk](#) a rectified average current is calculated.

Estimation of RMS current requires an understanding of the current profile that is seen during the charging or discharging process in any output net. The RMS current is defined as follows, where the current is integrated from 0 to T_{clk} , the clock period:

$$I_{rms} = \sqrt{\frac{\int I^2(t) dt}{T_{clk}}}$$

RedHawk by default approximates the switching current profile at the output of a cell using a polynomial-based profile whose shape and size depend on the average current of the net and the transition times (for rising or falling edges). Once the current profile is constructed, I_{rms} can be calculated using the above equation. You also have the choice of creating a different polynomial-based waveform for estimation of I_{rms} . Using that profile, RedHawk determines the peak current (I_{peak}).

The currents (I_{avg} , I_{rms} , and I_{peak}) are estimated for every signal wire segment in the design. The current values are typically highest closest to the driver cell and decrease gradually as distance from the drivers increases. RedHawk reads in the signal net routing and geometry information, along with signal net parasitic data, to determine the current values in each net, from its driver(s) to its receiver(s).

Once the current values are determined for each of these modes, the current density is estimated at each wire segment and via and compared to the relevant EM limits. The EM limits can be specified in the technology file as dependent on physical parameters such as the width of the wire, size of the via, and temperature of the die.

Input Data Requirements

An inter-cell EM analysis (wires outside of cells) requires the same data preparation as a regular RedHawk static analysis. Some of the key data requirements for an inter-cell EM analysis include:

- Routed design netlist in the DEF format
- Signal parasitic information in the SPEF/DSPF format
- Timing information (frequency, slew, clock nets)
- VCD or instance-specific toggle information [optional]
- Technology file with specified EM limits

Setup for Signal EM Analysis

The command files and setup needed for signal EM analysis are similar to those needed for static analysis. RedHawk estimates three different measures of current for every net in a single run. You can display different EM modes in the GUI by changing setting in ‘ElectroMigration Color Map’ dialog box, which is accessed using the ‘Set Color Range’ button in the GUI.

Waveform Specifications

The current profile used to estimate the RMS and peak current can be either a triangular or a polynomial waveform. You can select the mode using the GSR keyword (polynomial is the default):

```
EM_WAVEFORM_TYPE [polynomial | triangle ]
```

Besides waveform types, you can set the slew type with a GSR keyword to estimate the RMS and peak current, as follows (default is Min):

```
EM_USE_SLEW [ Max | Min | Average ]
```

When set to 'Max', RedHawk uses the maximum available slew value (transition time) for the net. If set to 'Min', the minimum available slew value is used. If 'Average' is specified, an average of the minimum and maximum slew values are used (in ns units). Using the 'Min' option is recommended.

The current profiles that are constructed depend on the output slew and load at every net, among other parameters. RedHawk gets slew information for signal and clock nets from the STA data provided and RC loading data. For cells missing the STA and RC data, you can control the profile construction using the following keywords (which also have default values to cover cells with missing slew data).

EM_SLEW_SIG_PERCENTAGE	p1 (default 0.25)
EM_SLEW_SIG_TIME	t1 (default 0.8 ns)
EM_SLEW_CLK_PERCENTAGE	p2 (default 0.1)
EM_SLEW_CLK_TIME	t2 (default 0.3 ns)

For signal nets, slew is calculated as

```
min(p1*clock_period, t1)
```

For clock nets, slew is calculated as

```
min(p2*clock_period, t2)
```

For signal nets missing load information in the SPEF/DSPF data, RedHawk estimates the missing slew information based on routing data. The GSR keyword TOGGLE_RATE should be set in order to assign a default toggle rate to signal and clock nets.

```
TOGGLE_RATE <signal toggle rate> <clock toggle rate>
```

Wire Merging

RedHawk tries to merge all signal wires that overlap, unless there are 45-degree wires in the design, when wire merging is disabled. Also overlapping nets are automatically dropped in analysis if there are any wires that it cannot merge. These nets are reported in the file *adsRpt/SignalEM/topcell.droppedSignalNets*. Note that you can force wire merging by setting the GSR keyword 'MERGE_WIRE 1' for designs with 45-degree wires.

EM Limits

For signal EM analysis on high technology designs, the peak current EM spec includes a duty ratio value for each net. To include the duty ratio data in RedHawk signal EM analysis, set the following GSR keyword to 1:

```
EM_USE_DUTY_RATIO 1 (default 0)
```

In the RedHawk technology file, in addition to the parameters needed for a regular analysis, the following additional EM limits can be set in the GSR:

```
EM_TECH_RMS <tech_file_nameA>
EM_TECH_DC <tech_file_nameB>
EM_TECH_PEAK <tech_file_nameC>
```

The format of these tech files is the same as a standard RedHawk tech file, except that only EM-related data are included. Three formats are acceptable, as follows:

```
metal <metal_layer> {
    EM <limit_value>
}
or
metal <matal_layer> {
    WIDTH_BASED_EM {
        width { <width1> <width2> ... <widthN> }
        em { <limit_width1> <limit_width2> ... <limit_widthN+1> }
    }
}
```

```

}
or
via <via_name> {
    EM <limit_value>
}
```

The list of EM limit values for WIDTH_BASED_EM correspond to the list of widths, with an additional EM value for any wire widths bigger than the largest width in the list.

These special tech files should **only contain EM limits**. All other tech file information, such as resistivity, thickness, and EM_adjust, should be specified in the regular RedHawk tech file, since they are the same across all EM modes (signal and power EM). If the additional tech files for EM limits are not provided, the EM limits in the regular tech file will be used.

If the special tech files for an EM mode contains no via-related information, via EM will not be checked for that EM mode. For example, the signal EM specification has no Peak or RMS EM limit for vias.

If there are nets with no driver port or physical connection the nets are dropped from the analysis and RedHawk errors out. A file *adsRpt/SignalEM/* *<top_cell>.droppedSignalNets* is generated to report the dropped nets.

Custom Current File

You can create a custom current file to specify current at signal net driving points using the GSR keyword 'EM_CUSTOM_CURRENT_FILE <filename>'. The format of the custom current file is as follows:

```

<net_name> INST <inst_name> <pin_name> <avg> <rms> <peak>
...
<net_name> CELL <cell_name> <pin_name> <avg> <rms> <peak>
...
where
```

net_name: specifies the net name

INST <inst_name>: specifies the instance name

pin_name: specifies the pin name

<avg> <rms> <peak>: specifies the current value for each type of current, in Amps. Current types that are not to be set should be given values of '-1'.

Example custom current file:

```

net1 INST ANA1 VREGO -1 -1 1e-3
net2 INST ANA1 SIGPIN -1 -1 2e-3
```

The example sets PEAK current to '1e-3' for net1 and '2e-3' for net2 at driving points VREGO and SIGPIN, respectively. Other current values are unspecified.

Using the custom current file you can also limit EM analysis to those nets that are specified. In above example, only nets 'net1' and 'net2' would be reported if you set the GSR keyword EM_CCF_ONLY to 1.

Hierarchical analysis

Signal EM analysis can be performed at the full-chip level, but verify only the interface nets and the top level nets. Interface nets are those that connect standard cells in a DEF to the primary I/Os in the DEF. Running in hierarchical mode reduces the overall run time and memory requirements by ignoring all internal nets and analyzing only the top level nets and the nets that connect from sub-block to sub-block. The GSR keyword to enable hierarchical analysis is:

```
SEM_HIERARCHICAL_MODE [ 0 | top_only | internal_only ]
```

Running Signal EM Analysis

The EM analysis flow uses a unified engine for both Power and Signal EM, so advanced EM rules are supported, with the following features:

- RMS and Peak current calculation consider RC load
- adding extra pin capacitance in the SEM_NET_INFO file is possible
- peak current calculation is adjusted based on saturation current
- ability to specify default slew, toggle-rate, and pin-cap parameters for all nets with missing information
- improved wire merging
- waveform estimation-based Signal EM analysis
- Signal EM flow calculates I(rms) and I(Peak) of the primary output nets using the Ceff values of the primary output nets. To enable this, you must provide Ceff values for the nets in the EM_NET_INFO file, which has the following format:

```
#<net_name> <trans_time_sec> <toggle_rate> <freq> <uni-dir_scale>
<bi-dir_scale> <extra-cap> #<driver_cell> <pin> ?<Ceff>?
```

The signal EM flow includes command steps as described below, so that you have access to the results of each step individually:

```
# set the analysis mode to signal EM
setup analysis_mode signaleM

# import the GSR file
import gsr <gsr_file_name>

# set up the design
setup design

# Perform power calc step to find toggle rates of instances
perform pwrcalc

# extract signal nets using the SPEF file
perform extraction -signal

# perform signal EM analysis
perform analysis -signaleM

# create EM analysis report for each set of rules
perform emcheck <options>
```

Note:

1. The ‘setup analysis_mode’ step must be first for signal EM, as opposed to the power EM analysis procedure, when it is run after ‘setup design’.
2. If user wants to perform only-clock extraction or only-signal extraction then the following option can be used in extraction step in SignalEM analysis.
 perform extraction -signal -exclude_clock -> for signal-only extraction
 perform extraction -clock -> for clock-only extraction

Using the RedHawk GUI in Signal EM

In the GUI, there are several useful ways to analyze EM-related results:

- Current information can be obtained by clicking on any metal wire:

Bidirectional current:

RMS 9.061e-02 A

Peak 1.812e-01 A

Avg 4.530e-02 A

- To check current EM_MODE, use the command:

gsr get EM_MODE

- To change the EM_MODE, use the command:

gsr set EM_MODE [rms | avg | peak]

or use the appropriate GUI button in the EM Color Map dialog.

- To view nets, select from the menu **View->Nets**

- To highlight instances related to a net, use the command :

select add [get instbynet <netname>]

- Also, the command

select add [get instbynet -driver <netname>] -color red

can be used to highlight a driver in a different color, such as red.

Defining Equipotential Regions

You can define equipotential regions by creating a single resistor to represent the metal segment in that region, if the analysis is creating too many nodes and dividing the voltage and current among them. This technique is useful for special shapes, such as octagonal pads. In order to do this, the equipotential regions must be defined in the pad file, using the format described below. Equipotential support is available for P/G nets as well as signal nets for signal EM analysis, and can be defined on any layer. The PLOC file format for defining the regions is as follows:

<area_name> <x1> <y1> <x2> <y2> <layer> <net_name>

For example:

Equi_pot_areal 203.840 341.000 323.840 461.055 ZA TEST_NET

Analyzing Results

You generate signal EM reports based on netname, mode, or temperature using the TCL command described below, after EM checks have been performed for all nets in the design. The invocation syntax is:

perform emcheck ?-mode [AVG | RMS | PEAK |all]?
?-net <netname>?

The default mode is 'all'.

After **RedHawk** completes signal EM analysis, the following text reports and maps are available

- *adsRpt/apache.sigem.info* file

Check signal nets that have no loading capacitance, timing, or driver instance information.

- *adsRpt/redhawk.err* file

Reports all signal EM flow setup and analysis errors.

- *adsRpt/SignalEM/<design_name>.rms_em.worst* file

Reports the highest 1000 J_{rms} (RMS current density) violations.

To report more than 1000 violations, use the GSR keyword:

```
EM_REPORT_LINE_NUMBER [ Max_line ]
```

Also, EM_REPORT_PERCENTAGE (default 100) can be used to specify the threshold for reporting violations, the same as Power EM. Sometimes you must use a lower EM_REPORT_PERCENTAGE to get any entries in the *em.worst output files.

- *adsRpt/SignalEM/<design_name>.avg_em.worst*

Reports highest 1000 J_{avg} (average current density) violations.

- *adsRpt/SignalEM/<design_name>.peak_em.worst*

Reports highest 1000 J_{peak} (peak current density) violations.

- *adsRpt/SignalEM/<design_name>_sem_toggle_summary*

Reports toggle rates by net type.

Report format:

Net_Type	Count	Avg_Toggle_Rate
Data	<count>	<TR>
Clock	<count>	<TR>

- *adsRpt/SignalEM/<design_name>_sem_layer_wiseViolation_summary*

Reports layer-based EM violations.

Report format:

```
# Layer AVG_EM_VIOLATIONS RMS_EM_VIOLATIONS PEAK_EM_VIOLATIONS
Metall1 <violation_count> <violation_count> <violation_count>
```

Debugging Tips

- When encountering unexpected EM results, first check the *adsRpt/apache.err* file for error messages.
- Check the *adsRpt/SignalEM/<design>.em.warn* file.
- Use the ‘IGNORE_NETS { <netname> ... }’ GSR keyword to eliminate from consideration nets that are not of interest and reduce output data.
- Use the GSR keyword:

```
ANALYZE_NETS {
    <net_name_1>
    <net_name_2>
    ...
}
```

Chapter 16

Pathfinder™ ESD Analysis

Introduction - The ESD Problem

Electrostatic Discharge (ESD) is a common reliability problem in chip design that involves very high discharge currents that can fatally damage sensitive circuit elements, or adversely affect their operation. As many as 35% of total IC field failures are reported as ESD-induced. ESD represents a serious problem for the electronics industry, which must insure proper operation of chips during all phases of manufacture, test, packaging, and installation into products. Serious semiconductor problems caused by ESD are: oxide punch-through, junction burnout, and metalization burnout. ESD phenomena are typically classified into three discharge/test models:

- Human Body Model (HBM), an ESD discharging event occurring when a charged human body contacts an electronic device directly
- Machine Model (MM), an ESD pulsing event when charged machinery discharges when touching an IC during testing
- Charged-device Model (CDM), a self-induced discharging ESD event when ungrounded electronic parts are charged up during assembly and then discharge through a grounded pin.

These three ESD modes and their discharge waveforms are shown in Figure 16-1.

The traditional approach to ESD design and verification consists of several steps:

1. design at circuit level and verify on test chips
2. integrate ESD test cells into I/O cells of the SoC
3. review results by ESD experts before tape-out.

These empirical approaches are error-prone and involve high costs when the chip fails due to ESD problems.

A further complication is that ESD is a time-domain transient behavior, and has a very short duration. Some ESD failures can only be captured in time-domain circuit-level transient simulation. Unfortunately, there is no circuit-level simulator that can do dynamic ESD simulation at this time. Pure Spice-level simulation fails in properly handling negative resistance from the snap-back behavior of clamp devices under ESD stress. The tools that designers can use now are Spice analysis plus a device simulator. Capacity and long run-times are significant limitations of such solutions, so they can only be applied to a very small portion of a design.

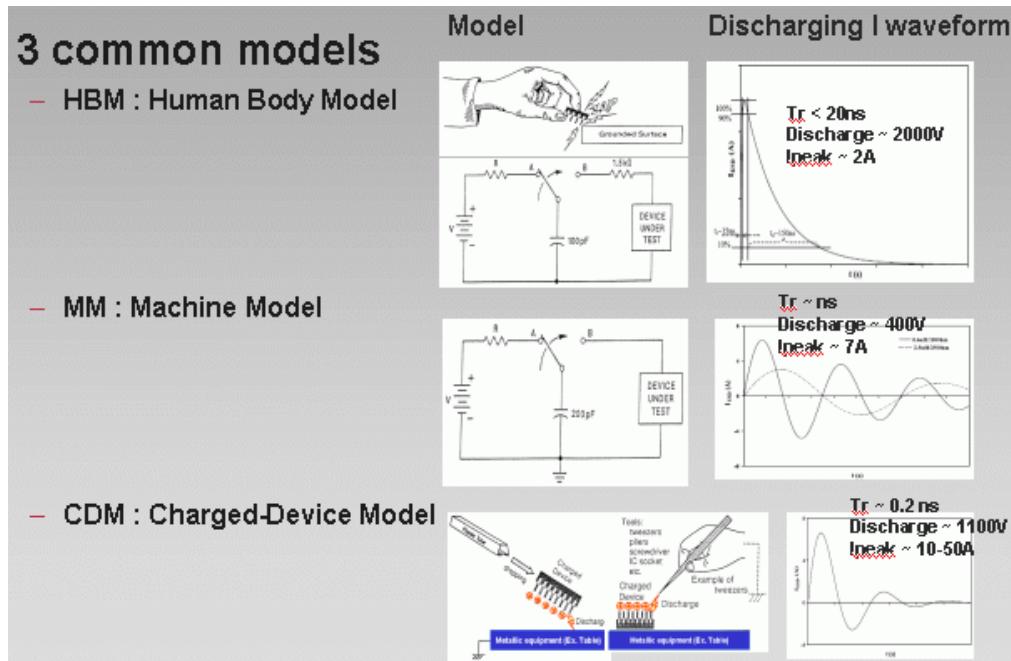


Figure 16-1 Three common ESD event models and discharge waveforms

Chip design has benefited from EDA solutions in many areas, but an EDA solution for ESD analysis has been a missing piece of the complete EDA design and analysis flow. Furthermore, ESD problems are now occurring more often due to advanced processes, larger design sizes, and mixed-signal and low power design methodologies.

Dealing with the difficulty of ESD transient simulation and long run times involved for covering all the discharging paths, a set of static techniques have been developed, primarily checking the resistance of the potential discharge paths, assuming ESD protection clamps are turned on. The RC time constant that controls the turn-on order of the clamps is constrained by the threshold resistance value, so the assumed actual resistance for the discharge path from a bump to another bump, or from a core instance to a clamp, can be calculated reasonably. If the resistance of a discharge path is much larger than the threshold resistance value, it often indicates a potential problem that will show up during the ESD testing, since there is no obvious discharge path for a bump in HBM/MM condition, or for a large custom macro in CDM condition.

ESD Analysis

Overview

PathFinder™-S provides a comprehensive set of technologies for performing ESD verification and analysis for both analog and SoC designs. The PathFinder tool uses static techniques (rule-based checks), while PathFinder-D uses dynamic transient simulation techniques for ESD analysis. Differences in PathFinder and PathFinder-D analyses are shown in Figure 16-2. For Netlist-based rule checks and dynamic-based simulation capability, please refer to the PathFinder Application Note “ESD Analysis and Verification in Totem for Analog and Mixed-signal Designs.”

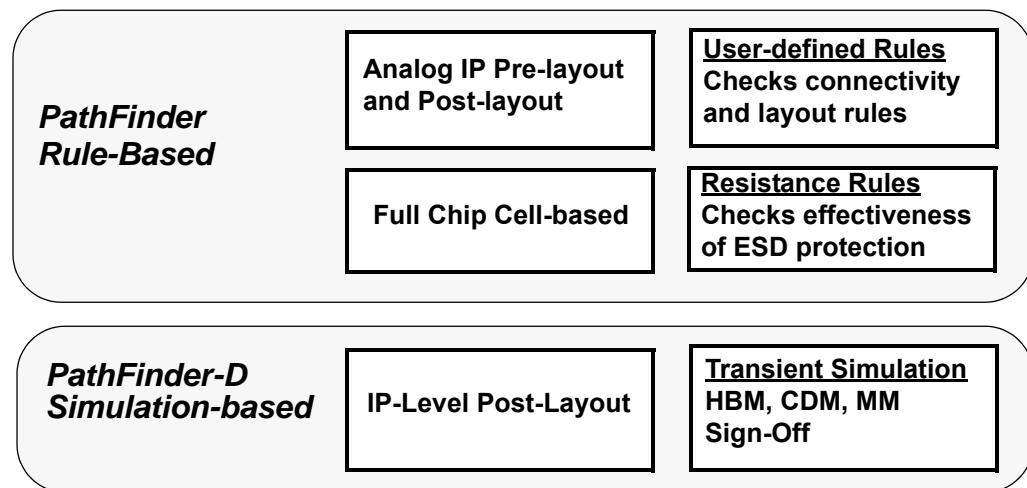


Figure 16-2 ESD analysis and coverage verification of PathFinder

PathFinder can be used for design planning (such as clamp cell placement, number of clamp cells, and design of clamp cells) or for sign-off (such as meeting discharge guidelines, and passing device-level threshold criteria). It can be used both on pre-layout circuits and also on post-layout designs. The checks provided by PathFinder are applicable for all three primary modes of electrostatic discharge events:

- Human Body Model (HBM)
- Machine Model (MM)
- Charged Device Model (CDM)

The capabilities provided in PathFinder can be classified into two broad categories:

- Verification of the placement of clamp circuits with respect to each other, the pads, and other cells in the design.
- Verification of the design of individual clamp circuits (at the schematic or layout level).

One set of checks evaluates the connectivity and topology of all elements of the circuit. In addition, resistance-based checks calculate accurate resistance values between the pads and the clamps, the clamps to the clamps, or from the clamps to other cells in the design. This capability is described in this chapter.

Clamp Cell Definition

Clamp Files

For all types of ESD analysis, you must specify I-V curves for the relevant diodes/clamps. A clamp cell can be defined either in its own clamp cell file or in a combined clamp cell/rule file. In a manually-generated clamp file, you can also specify the clamp I-V curve as in the following example:

```
BEGIN_CLAMP_CELL
    NAME  <cell name>
    TYPE <user_type_name>
    ? PIN [<pin_name> | NA] [<x_loc> <y_loc>]
        [ BOTTOM | TOP | <layername> ] <locID> ?
    ? XTOR <xtorName>:<net> <x_loc> <y_loc> <layer> <locID> ?
```

```
...  
? RON <clamp_On_Resistance_Ohms> ?  
? ESD_PIN_PAIR <loc_ID1> <loc_ID2> [  
    [ <RON> | [ <RON+>|OFF] [<RON->|OFF] |  
    <I-V_clamp_name> ]?  
? IMAX <I1> [<I2>] ?  
? VMAX <V1> [<V2>] ?  
END_CLAMP_CELL  
  
BEGIN_CLAMP_IV  
NAME <I-V_clamp_name>  
RON <RON+> [<RON->]  
VT1 <VT1+> [<VT1->]  
VH <VH+> [<VH->]  
ROFF <ROFF+> [<ROFF->]  
ION2 0.9  
RON2 0.1  
IT2 3.0  
RT2 1.0  
VT2 0.5  
IMAX <I1> [<I2>]  
VMAX <V1> [<V2>]  
TLP_FILE <tlp_filename>  
TLP_I_RESOLUTION <value>  
END_CLAMP_IV
```

For the CLAMP_CELL file, the keywords are:

NAME: specifies the name string to identify this cell or I-V device.

TYPE : user-specified name for the clamp cell type

PIN : specifies clamp cell pins by either <pin name>, by <x_loc> <y_loc>, by BOTTOM /TOP layer or <layername>, or by <locID>.

Note that only “-” should be used as a placeholder for clamp pin names, and *not* “NA”; ‘NA’ is treated the same as any other pin name in the cell. If “-” is specified, PF-S gets the pin location from wires/vias of the closest top level domain net for the BOTTOM/TOP layer or <layername> specified. If BOTTOM or TOP is chosen, the specified <x/y> location is snapped to the closest bottom or top layer of the <pin_name> net defined in LEF. If only a legal pin name is specified and no other information, by default a node with near minimum resistance to the pads is chosen. If no clamp cell pins are specified, one pin per domain is identified automatically that has near the minimum resistance to a pad. If only a subset of all clamp cell pins is specified, only the pins specified are used for ESD check. If a legal pin name is given, then this is also the default <locID> used in the ESD_PIN_PAIR clamp file keyword (defined below).

XTOR: defines ESD clamp device connections on transistors for MMX blocks.

Options are defined the same as for PIN, except that ‘:<net>’ is the name of the net that the transistor pin is connected to.

RON : resistance value when the clamp device is On; default: 0.0001 Ohms, RON+ is the resistance for current flowing from the positive to negative terminal; RON- is the resistance for current flowing in the opposite direction.

OFF: specifies the zapping direction is “off”; that is, RedHawk treats the clamp device as an open circuit with that zapping polarity. Note that it is considered a syntax error if both directions of a pin pair is specified “off”.

ESD_PIN_PAIR : defines a legal ESD discharge path for B2B rules, and resistance values for both current discharge directions. <RON+> defines the On resistance for discharges from <loc_ID1> to <loc_ID2>, and <RON-> defines the On resistance for discharges from <loc_ID2> to <loc_ID1>. <RON> defines symmetric resistance between the node pair; if defined, it overrides the RON definition above, or otherwise a default value of 0.0001Ohms is used.

IMAX: for a clamp cell rule, specifies the current thresholds for the positive and negative breakdown checks for the associated ESD_PIN_PAIR. When only one value is specified, the threshold is the same in both directions.

For a clamp I-V device rule, specifies the current thresholds for that device.

For an ESD_PIN_PAIR, the limits specified in the clamp I-V rule have the highest precedence, followed by those in the clamp cell rule, then in the ESD rule.

VMAX : for a clamp cell rule, specifies the voltage thresholds for the positive and negative breakdown checks for the associated ESD_PIN_PAIR. When only one value is specified, the threshold is the same in both directions. For a clamp I-V device rule, specifies the voltage thresholds for that device. For an ESD_PIN_PAIR, the limits specified in the clamp I-V rule have the highest precedence, followed by those in the clamp cell rule, then in the ESD rule.

For the CLAMP_IV file, the keywords are described below and identified in Figure 16-3.

ROFF: resistance value when the clamp device is Off; default: 1e6 Ohms.

ROFF+ is the off resistance for current flowing from the positive to negative terminal;

ROFF- is the off resistance for current flowing in the opposite direction.

VT1: threshold voltage that turns On the clamping device; default: 0. That is, the clamp is modeled as a linear resistor with RON.

VH: holding voltage when the clamping device is On; default for VH: VH = VT1; the supported range for VH is 0 <= VH <= VT1.

ION2: current threshold for On resistance to change from RON to RON2.

RON2 : resistance in the upper conducting region

IT2: current threshold for I-V curve to enter breakdown region.

RT2: resistance in the breakdown region.

VT@: threshold voltage at the beginning of the breakdown region.

TLP_FILE : specifies a file with Transmission Line Pulse (TLP) curve data in ESD clamp I-V definitions for both resistance and CD checks. Sample TLP file:

```
0 , 0
1.075228 , -5.57006E-06
2.181616 , 3.30643E-05
3.273627 , 7.19229E-06
4.283446 , 7.90741E-05
5.371665 , 7.80592E-05
```

where ', ' can also be a space or tab. And '#' at the beginning of a line for comments is allowed.

TLP_I_RESOLUTION: Based on the TLP data PF-S engine automatically decides a value for 'TLP_I_RESOLUTION' by default and user does not have to set it. The typical range is from 0 to 0.001.

Most clamp I-V parameters have two versions, '+' for the positive-to-negative current direction, and the '-' for the reverse direction.

The executable script 'pfscrypt' can be used to encrypt IV_clamp curve in CLAMP_FILE

Usage:

```
Pfscrypt <clampfile> [options]
```

options:

- h print this help message
- d turn on decryption mode
- p <Password> set the password for encrypt/decrypt
- o <OutputFile> set the output file name; Default: <FileName>.enc

Sample output:

When read encrypted clamp_file, Clamp_IV is show below

```
adsRpt/ESD/ClampInfo.rpt
```

```
# CLAMP IV DATA:
BEGIN_CLAMP_IV
    NAME diode_iv # Encrypted
END_CLAMP_IV
BEGIN_CLAMP_IV
    NAME pwr_iv    # Encrypted
END_CLAMP_IV
```

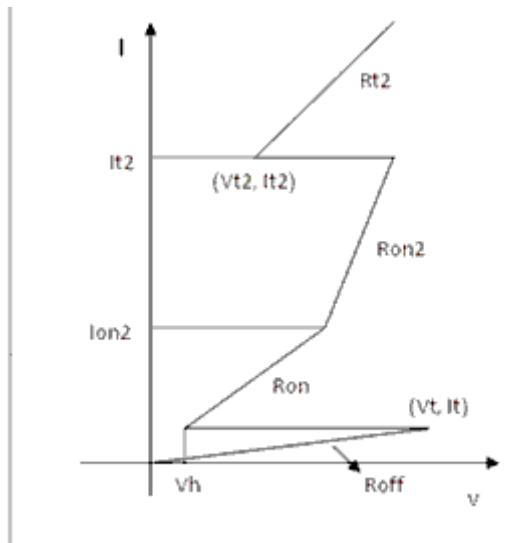


Figure 16-3 Diode/clamp device I-V characteristics

Sample clamp cell file:

```
BEGIN_CLAMP_CELL
    NAME IO_cell
    PIN PAD 7.29 362.14 METALL1 pad_1
    PIN VSS 7.39 362.04 METAL2 vss_1
    PIN VDD 7.29 363.66 METALL1 vdd_1
```

```
PIN VDD 7.29 369.4 METAL2 vdd_2
PIN VSS 7.0 334.5 METAL3 vss_2
RON 0.01
ESD_PIN_PAIR pad_1 vss_1 0.001 0.05
ESD_PIN_PAIR vdd_1 vss_1 0.02 OFF
ESD_PIN_PAIR vdd_2 vss_2 OFF 0.0001
END_CLAMP_CELL
```

Sample transistor clamp cell settings are shown following:

```
BEGIN_CLAMP_CELL
NAME IO
# INST : mos1__2 PIN : VDD
XTOR mos1__2:VDD 2.24 328.14 BOTTOM
# INST : mos1__2 PIN : VSS
XTOR mos1__2:VSS 2.34 328.14 BOTTOM
# INST : mos4__1 PIN : VDDO
XTOR mos4__1:VDDO 1.81 309.23 BOTTOM
# INST : mos4__1 PIN : VSS
XTOR mos4__1:VSS 1.91 309.23 BOTTOM
# INST : mos4__2 PIN : VDDO
XTOR mos4__2:VDDO 1.81 317.31 BOTTOM
# INST : mos4__2 PIN : VSS
XTOR mos4__2:VSS 1.91 317.31 BOTTOM
ESD_PIN_PAIR mos1__2:VDD mos1__2:VSS diode
ESD_PIN_PAIR mos4__1:VDDO mos4__1:VSS sbback
ESD_PIN_PAIR mos4__2:VDDO mos4__2:VSS b2b_diode
END_CLAMP_CELL
```

```
BEGIN_CLAMP_IV
NAME diode
RON 0.1 100K
VT1 0.4
ROFF 100K
END_CLAMP_IV
```

```
BEGIN_CLAMP_IV
NAME b2b_diode
RON 0.2
VT1 0.2
END_CLAMP_IV
```

```
BEGIN_CLAMP_IV
NAME sbback
RON 0.3 0.2
VT1 1.2 1.5
VH 0.1 0.15
END_CLAMP_IV
```

In this example, VT1, VH, and RON are specified according to the parameters defined in Figure 16-3. By default, VH = VT1, <RON> = <RON+>, <VT1> = <VT1+>, and <VH> = <VH+>. If none of <RON>, <ROFF>, <VT1>, <VH> are specified, they assume the default values defined above.

For more information on clamp device connection and modeling, see GSR keyword section "[ESD_CLAMP_PIN_FILE](#)", page C-728, and TCL command [section "pfs"](#), page D-798.

You can perform B2B checks before doing DC analysis. Loop resistance used for DC analysis does not need to be the same as that used for the B2B check. The clamps in “pass loops” in the B2B check are used for DC analysis. When there are no “pass loops”, all clamps are off, and DC analysis cannot be performed. If the B2B check has not been performed, RedHawk performs iterative convergence to decide which clamps are on and off in the final solution.

Clamp DB Creation

To make clamp definition and reuse more efficient, PathFinder creates a binary DB for clamp devices (such as cells, instances, nodes, esd pin pairs, I-V data) to be used by the general ESD checking command 'perform esdcheck' (for syntax details, see [section "perform esdcheck", page 16-448](#)) for both clamp rules and also for bump rules. With general clamps definition in the DB, specific clamp parameters can be added or removed from particular rules files. The basic controls to invoke checking relative to clamps are described in this section. Using the clamp-related options, the syntax is:

```
perform esdcheck
    -clamp <clamp_file> ? -setupClamp ? ? -shortNode [ 0 | 1 |
2 ] ?
```

where

-clamp <clamp_filename> : specifies the clamp filename. Clamp statements can be combined in the rule file or separately defined in a clamp file. See the clamp file section for a description of the clamp file syntax and an example clamp file. If a clamp file is specified using the GSR keyword ESD_CLAMP_FILE, this option is not required.

-setupClamp : imports and saves the clamp info into the DB. You can check Errors/Warnings in the clamp file before running ESD checks. Saves significant run-time without having to import a clamp file using “-clamp <file>” each time ESD checks are run. You can save or load the clamp data by including the following keywords in the rule file:

```
SAVE_CLAMP_DB 1
LOAD_CLAMP_DB 1
```

Both keywords are default off. Note that

- Only one of SAVE or LOAD can take effect. When both are specified, LOAD is used.
- When you specify an ESD rule without LOAD_CLAMP_DB 1, and if you do not specify clamp info either in rule file or with -clamp <file>, RedHawk loads the existing clamp DB.
- If neither option -setupClamp or the rule 'SAVE_CLAMP_DB 1' are specified, if the clamp DB does not exist, the clamp info is saved into the DB.

-shortNode: Specifies the node shorting options. The available options are:

0 (off): Do not short ESD clamp pin nodes during extraction.

1 (all): Short all ESD clamp pin nodes during extraction.

2 (loc): Short ESD clamp pin nodes when x/y/layer are specified in ESD_CLAMP_FILE during extraction.

SPT tracing from clamp nodes

You can perform SPT (minimum resistance path) tracing from clamp nodes using the 'perform min_res_path' command. To do this, the clamp DB should be ready. That is, the 'perform esdcheck -clamp <clampFile> -setupClamp' command must be executed before

doing SPT. After the clamp DB is set, the following two methods can be used to trace SPT paths from clamp nodes:

1. Add the “-fromClamp” option to the “perform min_res_path –instance < >” command, or
2. Execute the ‘perform min_res_path -set fromclamp [on | 1]’ command to turn the feature on. After it is turned on, the specified SPT paths are automatically traced from clamp nodes until it is turned off. This feature is designed for SPT tracing from the GUI.

Example:

```
perform min_res_path -instance inst_13ab/inst_50cd  
-fromClamp
```

ESD Rules Files

The parameters for ESD checks must be specified using a rules file that defines key elements of each desired check, such as TYPE (BUMP2BUMP, BUMP2CLAMP, CLAMP2CLAMP, PIN2CLAMP, PIN2PIN), a user-assigned rule NAME, limits on ARC_R, LOOP_R, and PARALLEL_R, the number of stages, and also the clamp types, bumps and nets that are to be specifically included or excluded in that check.

Defining Groups of Nets

ESD rules files can specify user-defined net groups to simplify including or excluding certain groups of nets in ESD testing. The rule syntax for defining a net group is as follows:

```
BEGIN_NET_GROUP  
    NAME <userDefinedGroupName>  
    NET <netName1>  
    NET <netName2>  
    ...  
END_NET_GROUP
```

The creation of various types of specific rules files are described in the following sections.

Reducing Design Size for Improved ESD Analysis

To speed up performance on large designs, PathFinder can generate a special GSR file that selectively removes ESD-irrelevant cells/layers to reduce design size, while keeping acceptable accuracy. The methodology is as follows:

1. In the original GSR file, add the following GSR keywords:

```
ESD_CLAMP_FILE <clampFile>  
ESD_RULE_FILE <ruleFile>
```

where <clampFile> and <ruleFile> are the clamp and rule files to be used for the “perform esdcheck” command.
2. Add the “ESD_GSR 1” rule keyword in the appropriate rule file.
3. Run RedHawk through the setup design stage. A new *esdGsr.gsr* file with modified GSR keywords is generated:
 - a. Unnecessary standard-cell LEF cells are removed from analysis with the IGNORE_CELLS keyword, when no C2I/C2M/B2I rules are involved.
 - b. SPLIT_VIA_ARRAY or SPLIT_SPARSE_VIA_ARRAY keywords are removed, if no current density rules are involved.

- c. Lower metal layers from the block/top level DEF files are ignored using the DEF_IGNORE_SPECIFIC_LAYERS keyword.

4. Replace the original GSR file with the *esdGsr.gsr* file and rerun RedHawk.

The performance and capacity improvement of running with the new *esdGsr.gsr* could be up to 50%, depending on the design.

Topology and Connectivity Checking of Bumps and Clamps

PathFinder evaluates and reports on clamp instances that are not connected to P/G and signal nets, and bumps isolated from clamps. You can obtain a text report using the command 'perform clampcheck' as follows:

```
perform clampcheck ?-o <file>? ?-instConn?  
? -isolatedBump? ?-cell <cell_name>? ?-celltype <type>?  
? -inst <inst_name>? ? -volt <voltage>? ?-net <net_name>?  
? -netConn <net1> <net2> ...? ? -allNetConn ?  
? -bumpConn <bump_name> ? ? -allBumpConn ? ? -loop?  
? -loopLength <stage_num>? ? -b2bLoopLength <stage_num>?  
? -roff <R_thresh> ? ? -rptDisConn ?  
? -rule <rule_file> ? ? -esdStage <stage_num>?  
? -detail? ?-append?
```

where

- o <file>: specifies the output filename
- instConn: reports clamp instances that have missing connections to the P/G grid.
- isolatedBump: reports bumps that do not connect to any clamp
- cell <cell_name>: reports information on specified clamp cell(s)
- celltype <type>: reports information on clamp cells of a particular clamp type
- inst <inst_name>: reports information on specified clamp instance(s).
- volt <voltage>: reports a list of clamps connected to a particular node voltage.
- net <net_name>: reports information on all clamp nodes connected to specified net(s), both directions.
- netConn: checks connectivity of each net pair (both directions), so for “-netConn {netX netY}”, the connectivity of both netX->netY and netY->netX is checked. If “-netConn netX” is specified, the connectivity from netX to all other nets and from all other nets to netX is checked. For each net pair reported, a symbol “->” or “<->” is added between the two nets to indicate the connectivity direction between them.
- allNetConn: reports clamp connectivity to all nets and domains, in both directions
- bumpConn: reports clamp connectivity to all bump connectivity from the specified bump
- allBumpConn: reports clamp connectivity of all bumps
- loop: lists the clamp instances of all B2B loops for queried net pairs.
- loopLength: -b2bLoopLength : set the EXACT stage number(s) to be reported for queried net pairs. Note that “-esdStage” sets the RANGE of stage numbers, while “-loopLength” sets the EXACT stage numbers. -loopLength can specify more than 2 numbers.
- roff: sets the resistance threshold value at which clamp devices are considered “OFF”. The default threshold value is 1e6 Ohms.
- rptDisConn: reports net-pairs with no clamps between

- rule: specifies the name of the rule file for checking. Multiple rule files can be represented in two formats: {rule1 rule2 ...}, or rule1,rule2, ... (no spaces in this form).
- esdStage: specifies the number of clamp stages for net/domain pair checks
- detail: reports detailed information on net/domain pair.
- append: appends results to existing results in the output file

An example invocation would be:

```
report clampcheck -rptDisConn -allNetConn
```

which reports all net-pairs between which no clamps are found.

A sample ‘Unconnected Clamp Instances’ report is shown following for the command:

```
perform clampcheck -o instConn.txt -instConn
```

```
##### List of Unconnected Clamp Instances #####
#
# <CELL NAME>:
#     <PIN_NAME> <X> <Y> <LAYER> <LOCID> <INSTANCE_NAME>
inst_ndiode_0:
VDDO 1.07374e+06 1.07374e+06 metall1 VDDO inst_ndiode_0/adsU4
```

A sample ‘Bumps Isolated from Clamps’ report is shown following for the command:

```
perform clampcheck -o isolated_bumps.txt -isolatedBump
```

```
##### List of Bumps Isolated from Clamps #####
#
# <BUMP_NAME> <BUMP_X> <BUMP_Y> <NET_NAME> <BUMP_LAYER>
1_pad 4569.13 10992.5 net1 metal8
```

Layout Resistance Checking of Bumps and Clamps

Overview

PathFinder can perform the following five types of ESD-related resistance checks:

- Bump-to-Bump (BUMP2BUMP, or B2B), including multi-stage checks
- Bump-to-Clamp (BUMP2CLAMP, or B2C)
- Bump-to-Instance (BUMP2INSTANCE, or B2I)
- Clamp-to-Clamp (CLAMP2CLAMP, or C2C)
- Clamp-to-Inst (CLAMP2INST, or C2I)
- Clamp-to-Macro (CLAMP2MACRO)

Note that ESD checking is automatically turned off in low power analysis mode.

Bump-to-Bump (BUMP2BUMP, or B2B)

For bump-to-bump resistance checking, you must specify the list of bumps/pads (or bump locations) that form pairs. Each bump pair must be connected by one or more clamp cells.

The first step in this calculation is the estimation of the loop resistance, which is the resistance for each bump->clamp->bump path. Loop resistance is defined as the actual Vdd/Vss resistance from a bump to another bump. For bump pairs that have multiple clamp cells connecting them, an equivalent number of paths for loop resistances is calculated.

Once the loop resistances are calculated, they are compared against the loop resistance threshold that has been defined. The loops (bump->clamp->bump paths) having an effective resistance greater than the specified threshold are considered as failing the loop resistance check. The clamps that are in these failing loops are deemed “invalid” and are discarded from the subsequent bump-to-bump parallel resistance calculation, in which the effective resistance from one bump to another is calculated, considering all the valid clamp cells connecting these two bumps. The invalid bumps (from the loop resistance checks) are considered to *not* provide a discharge path between a pair of bumps.

This rule also calculates and performs ESD resistance checks on multi-stage ESD BUMP to BUMP pairs. PF-S calculates the resistance from each bump to ESD clamp instances, forms multi-stage bump to bump paths, and performs ESD resistance checks on these multi-stage bump2bump paths.

Bump-to-Clamp (BUMP2CLAMP, or B2C)

Next, the effective resistance from a pad to a clamp cell is calculated and reported. You can provide the pad location, or the tool can determine it automatically. Similarly, for the clamp cell, the location to which the resistance is calculated can be specified (recommended), or it can be determined by the tool. If the tool determines it automatically, it attempts to identify the location with the highest effective resistance. This pad-to-clamp resistance is compared against the threshold specified using the command line option ‘-arc_R’.

For B2C resistance checking, to apply different resistance thresholds from the pad to the anode of the HBM diode, and from the pad to the cathode of the HBM diode, use the following keywords to select the clamp pins to be tested in separate B2C runs:

- To apply the rule only to the anode of the HBM diode:

```
CLAMP_POS_PIN 1
```

- To apply the rule only to the cathode of the HBM diode:

```
CLAMP_NEG_PIN 1
```

Bump-to-Instance (B2I)

B2I rules support Bump-to-Instance resistance checks. Using this feature you can check if the resistance to an instance from the bump is greater than the resistance to the clamp.

Clamp-to-Clamp (CLAMP2CLAMP, or C2C)

Finally the effective resistance between any two clamp cells connected to each other is reported, and compared against the threshold specified in the ‘-arc_R’ command line option.

Clamp-to-Inst (CLAMP2INST or C2I)

The effective resistance from a core instance to a clamp cell is calculated and reported to identify potential CDM failures. You must provide a list of the instances. Then locations with minimum effective resistances are chosen for standard cells and locations with maximum effective resistances are selected for macro cells (such as memories, IPs, etc.) automatically. The rationale for these choices is that macro cells are much larger in size compared to standard cells and therefore the worst location with maximum effective resistance should be considered to catch potential C2I failures. The same location

selection method can be applied to clamp cells. Both Loop_R and Arc_R are reported with pass/fail criteria with respect to Arc_R and Loop_R constraints specified in the rule.

Clamp-to-Macro (CLAMP2MACRO or C2M)

The effective resistance from a macro instance to a clamp cell is calculated and reported to identify potential CDM failures. You must provide a list of the instances. Then all node locations on the bottom/top/<layer> of the LEF pin geometries with minimum effective resistances with respect to one of the clamp cells are chosen (such as memories, IPs, etc.) automatically. The difference from C2I selection is that many more node locations are considered for checking potential CDM failures (therefore achieving better location coverage) . The same location selection method can be applied to the clamp cells. Only arc_R is reported with pass/fail criteria with respect to arc_R constraints specified in the rule.

Cross-domain DRIVER2RECEIVER (D2R)-to-Bump

Checks bump resistance for driver-receiver circuits that cross voltage domains. Rules for these tests are used to check the difference/ratio of the resistance from P/G pins of cross-domain driver-receiver instance pairs to all bumps in design that are connected directly or through clamps.

Including Package Resistance

RedHawk PathFinder ESD analysis can include the effects of package subcircuits. The package can have a significant impact on some ESD results, since it modifies the resistive paths between many nodes in the chip. For ESD resistivity analysis, inductances in the package subcircuit are short-circuited, and mutual inductances and capacitances are open-circuited. Clamps are identified, a radius formed, valid bumps are identified, and then resistances are calculated and compared. The same ESD checks are performed as in chip analysis, except that the package resistance from the bumps to their package BGA sources (for both the VDD and VSS) is included. So the check starts from the BGA side of the package, through the package, through the VDD bumps, through the P/G network to the clamp, and then returns to VSS bumps and through the package to the BGA pins.

Several types of rules are added to the 'perform esdcheck' command for computing resistances beginning or ending at the package pins. The setup of these rules is similar to the corresponding BUMP-related checks. The package-related rules are:

- PIN2CLAMP (or P2C) - external package pins to clamp nodes, similar to B2C
- PIN2PIN (or P2P) - external package pins to external package pins, similar to B2B, including multi-stage cases.

Package subcircuits can be included in **RedHawk** analysis by the same mechanism as is used for static and dynamic simulation. The most common method is to specify the package subcircuits using the GSR keyword PACKAGE_SPICE_SUBCKT. Once a package subcircuit is included, ESD checks can be performed with or without the package subcircuits. The package subcircuit is required and is automatically included for the P2C, P2P and P2PM checks, in which all resistive paths start or end at the package pins. For other resistive ESD rule types, the default behavior is to exclude the effects of the package.

Data Flow

The Layout Resistance Checker of PathFinder can be run inside both **RedHawk** and Totem tools. When running inside **RedHawk**, it helps verify the placement of clamp cells in SoCs and other digital circuits. It typically works with a LEF/DEF-based input, although GDS is also supported. When running inside **Totem**, it helps perform similar checks,

particularly for analog, custom, and mixed signal circuits. In this case, it works with GDS-based inputs, with additional data coming from the Spice netlist. The data flow is shown in Figure 16-4.

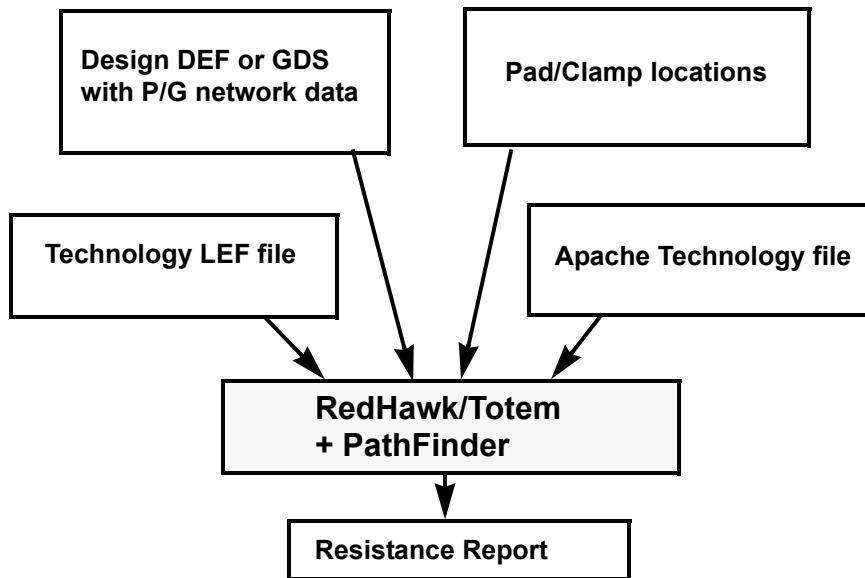


Figure 16-4 PathFinder Data Flow

Description of Inputs

The inputs for the Layout Resistance Checker are as follows:

- Technology LEF file - contains the layer and via descriptions for a particular library or process node in the LEF format available from the library team.
- Apache Technology file - specifies key technology parameters, such as resistance and dielectric constant, for **RedHawk** to use for extraction.
- Pad location (ploc) file - specifies the locations of voltage sources in the regular PLOC file format, as used in **RedHawk**. An example of a PLOC file with single VDD and VSS source points and is as follows:


```
# pad location points
<unique name> <x-loc> <y-loc> <metal> <power/ground>
vdd1    6808   11091   metal7    POWER
vss1    7038   11612   metal7    GROUND
```
- ESD static check includes geometry-based resistance checks for HBM/MM/CDM modes, and current density checking for HBM/MM.
- In the command file, include ‘setup analysis_mode ESD’ for ESD-specific control and GUI display.
- Rule definitions - see following sections
- Clamp cell definitions - a clamp cell can be defined either in its own clamp cell file or in a combined clamp cell/rule file, as described in [section "Clamp Cell Definition", page 16-435](#).
- Job control - PathFinder supports multi-threading and multi-processing capabilities. For multi-threading, use the '-thread' option; and for multi-processing, use the '-jobCount' option. To perform esdcheck with the -jobCount option, it is advisable to set up the clamp DB using 'perform esdcheck -clamp <clampfile> -setupClamp', and then execute 'perform esdcheck -rule <ruleFile> -jobCount <N>'. You can do multi-

processing by starting a session from scratch, or by importing a previously-exported DB. In that case, make sure the DB is exported from a session with 'setup analysis_mode esd' as the first TCL command executed.

If the machine has multiple CPU's, PathFinder splits the job and executes them. One job is performed by the parent PathFinder session, and the rest of the jobs are executed in different sessions in adsESD1, adsESD2, ... as the working directory. The job count and save directory options of 'perform esdcheck' are defined as follows:

- jobCount <n>: the number of jobs to execute at a time.
- saveDBDir : directory to save the DB and then import it when executing esdcheck in the same run

For detailed 'perform esdcheck' syntax, see [section "perform esdcheck", page 16-448](#).

You can also control splitting of rules into separate jobs by creating a job file using the '-jobFile <job_defin_file>' option, such as the sample job file below:

```
BEGIN_ESD_JOB
  NAME  <jobName>
  RULE   <ESD_rule_file_name>
  CLAMP  <clamp_file_name>
  DIR    <work_dir>
END_ESD_JOB
```

where NAME, CLAMP and DIR specifications are optional.

Inputs for 3D-IC Designs

Most inputs for ESD analysis are very similar for 3D-IC designs. The following are keywords that are specific for 3D-IC designs.

Current Density and Point-to-Point checks for 3D-IC

ESD check types "Current Density" and "Resistance", the following ESD rule keywords are available:

- FROM_DIE <name>: connects the high terminal of zap source in this die
- TO_DIE <name> : connects the ground terminal of zap source in this die
- DIE_NAME <name>: the CD/R analysis is performed for this die only

Also, DIE_NAME <name> can be used to specify that a clamp cell is in this die, as in the following example.

```
BEGIN_CLAMP_CELL
  NAME  <cellName>
  DIE_NAME <dieName>
  ...
END_CLAMP_CELL
```

Also, you can use keywords BEGIN_DIE /END_DIE to specify the die name with a list of clamp cells, as follows:

```
BEGIN_DIE <dieName>
  BEGIN_CLAMP_CELL
    NAME  <cellName>
    ...
  END_CLAMP_CELL
  ...
END_DIE
```

Resistance Checking for B2B, B2C, and C2C Rules

The resistance checking command is 'perform esdcheck'. There are a number of basic options available, which are described in the following syntax descriptions. Additional controls on the checking are provided by rules file keywords in the following section.

```
perform esdcheck
    -rule {<file1> <file2> ...}
    -clamp {<file1> <file2> ...}
    ? -thread <num_threads> ?
    ? -optimize <mode>? ? -detail ? ? -append ?
    ? -outDir <results_dir> ? ? -ignoreError ?
    ? -saveDBDir <path> ? ? -jobCount <num_jobs> ?
    ? -collate ? ? -jobFile <job_defin_file> ?
```

where

- rule <rules_files> : specifies the names of the rules files containing the rules to be checked. At least one rule must be specified. If the ESD check for a rule has an error, the default behavior is to stop execution. The '-ignoreError' option allows checking to continue past any errors, if possible, and attempt to finish all rules in the rule file. See the following section for a definition of the rules file and a sample rules file.
- clamp <clamp_files> : specifies the clamp filenames. Clamp statements can be combined in the rule file or separately defined in a clamp file. See the Clamp file section for a description of the clamp file syntax and an example clamp file.
- thread <num_threads>: specifies the number of threads/processes to be used, to speed up the calculation. All available CPUs may be used to support the multiple threads unless '-mcore' is specified. Default=2.
- optimize <mode>: as the mode specified increases from 0 to 3, more optimization is done, for greater performance gains.
Mode values 0 and 1 are only recommended to work around issues that have caused resistance computation to fail:
 - 0: all optimization off
 - 1: performs only basic netlist optimizations (up to 2X performance gain)Mode values 2 and 3 are recommended for normal use.
 - 2: performs aggressive netlist optimization for greater performance gain (up to 20X possible). This is the default.
 - 3. In addition to optimization of mode 2, performs additional optimization for greater coverage (can double the performance as with 2). This setting would be effective only for C2I, C2M, and B2I resistance checks.
- detail : provides detailed reports on B2B and multi-stage checks, including data on x,y locations, netnames, metal layers and the cell names of the clamp instances, so you do not have to refer to any other files to get details.
- append : appends results to the output files, instead of overwriting the previous output file
- outDir <results_dir>: specifies the directory for the results files. Default: *adsRpt*.
- ignoreError : where the <rules_file> contains multiple rules to be checked, if there are data issues or error conditions such that the ESD check for a rule has an error, the default behavior is to stop execution. '-ignoreError' allows checking to continue past any errors, if possible, and attempt to finish all rules in the rule file.

- saveDBDir: saves the DB into the specified directory. You can then execute esdcheck in the same run used for exporting the DB.
- jobCount: specifies the number of jobs to execute at a time when the rule file has multiple rules in it, or when an ESD current density check rule has many 'from/to' point pairs to be analyzed. This option can be used to partition the rule check into multiple sub-tasks, which are executed in parallel when the host machine has multiple CPUs. One job is performed by the parent PF session and the rest of the jobs are executed in different sessions in *adsESD1*, *adsESD2*, ... as the working directory.
The recommended usage is to start the session by importing a previously-exported DB, then to run ESD checking with the -jobCount option specified.
- collate: concatenates ESD text reports for the esdcheck jobs specified with -jobCount into the directory specified by -outDir option or the default master adsRpt/ESD.
- jobFile : specifies the job definition rules file

To export ESD checking results to a specified directory, the command is:

```
export esdcheck <esdcheck_db_directory>
```

To Import ESD checking results from a specified directory, or optionally import results for only a specific rule type, use the command:

```
import esdcheck <esdcheck_db_directory> ?-<ruleType>?
```

To specify the type of reports to be created for the ESD checks, the command is 'report esdcheck <options>'. For full command syntax, see [section "Report esdcheck command", page 16-469](#).

The default B2B report syntax is:

```
# BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) <->
(<BUMP2> <X> <Y> <LAYER> <NET>)
# PASS/FAIL: <LOOP_R> <ARC1_R> <ARC2_R> <CLAMP> <LOCID1>
<LOCID2> <INT_NET> <CLAMP> <LOCID3> <LOCID4>
# PARALLEL R: <VALUE>
```

For example:

```
FAIL: 4.34594 3.0401 1.30564 adsU1 RAIL_gnde gndo gndo
adsU2 gndo RAIL_vdde
```

With the -detail option:

```
# BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) <->
(<BUMP2> <X> <Y> <LAYER> <NET>)
# PASS/FAIL: <LOOP_R> <ARC1_R> <ARC2_R> <CLAMP> (<X> <Y>
<LAYER> <NET>) (<X> <Y> <LAYER> <NET>) <INT_NET>
<CLAMP> (<X> <Y> <LAYER> <NET>) (<X> <Y> <LAYER> <NET>)
# PARALLEL R: <VALUE>
```

For example:

```
FAIL: 4.34594 3.0401 1.30564 adsU1 (573.825 32.585 M6Z
RAIL_gnde_001) (535.85 105.21 M7Z gndo) gndo adsU2 (535.85
105.21 M7Z gndo) (573.825 53.585 M6Z RAIL_vdde_001)
```

Rules File Syntax for Types B2B, B2C, and C2C

A description of the syntax supported in the ESD rules file is shown below, including package integration.

```
BEGIN_ESD_RULE
TYPE [ BUMP2BUMP | BUMP2CLAMP | CLAMP2CLAMP |
PIN2CLAMP | PIN2PIN ]
```

```

NAME <rule name>
ARC_R <value_Ohms>
LOOP_R <value_Ohms>
PARALLEL_R <value_Ohms>
ESD_STAGE <min> <max>
PACKAGE [0 |1]
PACKAGE_PIN [0 |1]
PACKAGE_PIN_FILE <pkg_fileName>
PAD_FILE <ploc/pcell_filename>
EXCLUDE_PAD_FILE <ploc/pcell_filename>
EXCLUDE_BUMP <bump1> ... <bumpN>
EXCLUDE_BUMP_PAIR <bump1> <bump1A> <bump2> <bump2A> ...
EXCLUDE_BUMP_PAIR_FILE <file_name>
BUMP_LIST {<bump1> ... <bumpN> }
NET_PAIR <net1> <net2>
NET_PAIR_FILE <filename>
INTERNAL_NET <net1> <net2> ...
TERMINAL_NET <net1> <net2> ...
TERMINAL_NET_GROUP { POWER | GROUND | SIGNAL }
INTERNAL_NET_GROUP { POWER | GROUND | SIGNAL }
B2B_NET1 <net1> <net2> ...
B2B_NET2 <net1> <net2> ...
REPORT_TO_CLAMP [ OFF | MIN | ALL ]
CLAMP_TYPE <type_name>
CLAMP_ROFF <R_limit>
SHORT_CLAMP_NODE [ 0 | 1 ]
FROM_CLAMP_TYPE <clamp_type>
TO_CLAMP_TYPE <clamp_type>
CLAMP_POS_PIN [0 |1]
CLAMP_NEG_PIN [0 |1]
USE_CLAMP_CELL
    <cellName> [<locID1> [<locID2>]]
    ...
END_CLAMP_CELL
USE_CLAMP_INST
    <instName> [<locID1> [<locID2>]]
    ...
END_CLAMP_INST
USE_CLAMP_FILE <file>
B2B_CHECK_IV [0|1]
B2B_LOOP_LENGTH <clamp_count1> <clamp_count2> ...
RADIUS <R1_u> ?<R2_u>?
INCREMENTAL_MODE [0 | 1]
REPORT_B2C [0 |1]
REPORT_C2B [0 |1]
REPORT_C2C [0 |1]
B2B_PAIR_COUNT <num>
B2B_LOOP_COUNT <num>
MAX_ESD_STAGE <num>
CLAMP_SELECT [min |max]
EM_MODE [ ave | peak ]

```

```

DETAILED_MODE [0 | 1]
SHORT_BUMP_IN_NET <net name1> <net name2> ...
NO_SHORT_BUMP_IN_NET <net name1> <net name2> ...
SHORT_BUMP_IN_NET_GROUP [ POWER|GROUND|SIGNAL ] ?ALL ?
SHORT_MODE [0 |1]
B2B_MODE [0 |1]
ALL_PAD_NODE [0 |1]
SAME_CLAMP_INST [0|1]
DIFF_CLAMP_INST [0|1]
END_ESD_RULE

```

where

TYPE: specifies the checking rule type

NAME : user-specified rule name; reserved TYPE names are not allowed as rule names. PIN* rule names involve package pin checks.

ARC_R : specifies the resistance check threshold (Ohms) for BUMP2CLAMP, CLAMP2CLAMP, CLAMP2INST or CLAMP2MACRO checks. If specified in the 'perform esdcheck' command, the rules file value of ARC_R is overridden. If no Arc_R defined in the rule, PFS gets the Arc_R values from the ESD database. If no database exists, the 'report esdcheck' command stops with a message.

LOOP_R : loop resistance check threshold for BUMP2BUMP. specifies the resistance constraints for checking the loop resistance of a single ESD path (bump->clamp (ON resistance)->bump or core->clamp (ON resistance)->same_core). If specified in the 'perform esdcheck' command, the rules file value of LOOP_R is overridden.

PARALLEL_R : parallel resistance check threshold for BUMP2BUMP. specifies the pass/fail checking criteria for the total effective resistance of a bump pair. If specified in the 'perform esdcheck' command, the rules file value of Parallel_R is overridden.

Note: Bump to bump checking can be performed without specifying PARALLEL_R in the rule file. In such cases, the LOOP_R value is used as the pass/fail criteria for the B2B check. If any single Loop R passes, the B2B pair is considered to pass. If all the Loop R tests fail, then B2B pair fails. In the ESD reports, the PARALLEL_R is displayed as the minimum of all Loop R values for the B2B pair.

ESD_STAGE <min> <max> : used in B2B multistage rules. PathFinder checks all B2B net pairs that are connected by at least <min> clamps and no more than <max> clamps, and computes the bump-to-bump equivalent resistances for every bump pair in the net (subject to the RADIUS constraint). The value of <max> is limited to 5. If only one value is specified for ESD_STAGE, all bump pairs that have at least one bump-bump path (loop) with the specified number of clamps are analyzed. You can also control the number of best bump-clamp and bump-bump loops (loops with the lowest loop resistances) in the output report by using the keyword B2B_LOOP_COUNT. The maximum number of stages can be set using the option MAX_ESD_STAGE <num>, which can override the limit of 5 in ESD_STAGE.

PACKAGE 1 - includes the package subcircuits in ESD checking for the rule in which the keyword is included.

PACKAGE_PIN : specifies pin-to-pin checks with net-group options, such as FROM_NET/TO_NET/TERMINAL_NET between package pins. This option is different from the option 'PACKAGE 1', in which PathFinder uses bumps as

reference, but 'PACKAGE_PIN 1' considers package pin connections as reference.

PACKAGE_PIN_FILE <pkg_fileName> : for P2P, P2C and P2PM rules, specifies a file identifying nodes in the package subcircuit that should be treated as external pins, so that ESD checks compute resistances beginning and ending at those nodes. The format of this external file is as follows:

```
<Spice_nodename> <x_location> <y_location>
```

```
...
```

The <x,y locations> are only used for package reporting purposes. They are specified in user units, and must be floating point numbers. The node names in this file are case-insensitive, since Spice is case-insensitive. If no file is specified, the voltage sources in the package subcircuit are used as the external pins, and resistance computations originate and terminate at those nodes. The package pins specified are used only when checking the rule in which the keyword is included, and is meaningful only if package subcircuits are included for this rule.

PAD_FILE : specifies ploc/pcell file listing bumps to be included in ESD checks

EXCLUDE_PAD_FILE : excludes consideration of checks in specified pad file

EXCLUDE_BUMP : list of bumps to be excluded in ESD check

EXCLUDE_BUMP_PAIR : specifies sets of bump pairs that should be excluded

EXCLUDE_BUMP_PAIR_FILE: specifies a filename that provides a list of bump pairs to be excluded.

BUMP_LIST <pad1/bump1> ... <padN/bumpN>: lists names of pads/bumps on which the ESD check is to be performed. Optional.

NET_PAIR <net1> <net2>: designates net pairs to be analyzed for ESD interaction. B2B and CD checks are performed for all the bump pairs between the specified nets.

NET_PAIR_FILE <filename>: specifies a file path containing a list of net pairs to be analyzed.

INTERNAL_NET <net1> : specifies bumps in domains to be filtered out and **not** included in B2B checking, for either 1 or 2-stage systems.

TERMINAL_NET: specifies the **only** net(s) between bump pairs to be analyzed in B2B checking. All other nets are considered INTERNAL_NET and are not used in B2B checking. In the following sample rules file:

```
BEGIN_ESD_RULE
    NAME esd_b2b_rule
    ...
    TERMINAL_NET VSS A[ 3 ] VDDS
END_ESD_RULE
```

all bump pairs connected between nets VSS, A[3] and VDDS are investigated and reported.

TERMINAL_NET_GROUP : specifies the **only** net types included in the analysis -- POWER, GROUND or SIGNAL.

INTERNAL_NET_GROUP : specifies the only net types **not** included in the analysis -- POWER, GROUND or SIGNAL.

B2B_NET1 : specifies the net(s) in the B2B_NET1 group that are to be paired with the net(s) in the B2B_NET2 group for B2B checks. Previously called 'FROM_NET'.

B2B_NET2 : specifies the net(s) in the B2B_NET2 group that are to be paired with the net(s) in the B2B_NET1 group for B2B checks. Previously called 'TO_NET'.

REPORT_TO_CLAMP : specifies TO_NET end on clamps and not bumps. The options are:

OFF: Disabled (default)

MIN: Report at the clamp with minimal resistance

ALL: Report resistances to all clamps

CLAMP_TYPE : user-specified name for the clamp cell type to be considered, which allows limiting checking only to specified clamp type(s)

CLAMP_ROFF: sets the clamp resistance limit for reporting. B2B checking treats clamps with $R \geq CLAMP_ROFF$ as 'off'; and not involved in the loop/parallel R computation, and these clamps are not reported.

SHORT_CLAMP_NODE : shorts clamp nodes "on-the-fly" in ESD-R and ESD-CD checks when resistances are computed.

FROM/TO_CLAMP_TYPE : specifies checking FROM/TO <clamp_type> as defined in the clamp cell file with the 'TYPE' keyword. The clamps specified in FROM_CLAMP_TYPE are used as reference clamps and all reporting is done with respect to the "FROM*" list items.

CLAMP_POS_PIN [0|1] : when set to 1 (default is 0), ESD B2C resistance check will pick up only the nodes connected to the positive/negative pins of a clamp diode device; for example, as in:

```
ESD_PIN_PAIR Vsig Vdd 0.1 off
```

where the positive pin is Vsig, and the negative pin is Vdd.

CLAMP_NEG_PIN [0|1] : same behavior as CLAMP_POS_PIN for negative pins

USE_CLAMP_CELL / END_CLAMP_CELL : defines clamp cells used in the check, with and without locIDs. Wildcards are honored.

USE_CLAMP_INST/END_CLAMP_INST : where user can define clamp instances, with and without locIDs. Wildcards are honored.

USE_CLAMP_FILE <file> : in specified file, the following syntax can be defined.

```
USE_CLAMP_CELL
    <cellName> [<locID1> [<locID2>]]
    ...
END_CLAMP_CELL
USE_CLAMP_INST
    <instName> [<locID1> [<locID2>]]
    ...
END_CLAMP_INST
```

Note that wildcards are honored in the clamp cell and instance names, and the locIDs are optional. The above keywords can be applied to B2B/B2C/CD rules for clamp selections.

B2B_CHECK_IV : To include clamp voltage and current and I-V data in B2B resistance check report. This keyword supports new B2B report format (that has additional VT1, RON information of current format) with or without ZAP_CURRENT for B2B checking.

B2B_LOOP_LENGTH : <clamp_countN> values specify the number of clamps in each B2B path (positive integers less than or equal to MAX_ESD_STAGE). Values of <clamp_countN> *not included* mean that B2B paths with that number of clamps are excluded from parallel R computation.

RADIUS : specifies the maximum distance between B2B, B2C or C2C pairs that is checked (in microns) for the rule. Note that in B2B checking, if no bump pairs fall within the radius specified, PathFinder automatically extends the RADIUS value to include about 10% of the number of bump pairs specified in the option B2B_PAIR_COUNT. A message about what the RADIUS value is and how many pairs are included is displayed. Two radius values may be provided analyze the area between the two radii.

INCREMENTAL_MODE : default 0; provides incremental resistance check coverage when you have already run with one radius and want to rerun with an increased radius (uses previous B2C results).

REPORT_B2C : default 0; reports B2C results and B2B checks (bump-centric reporting)

REPORT_C2B : default 0; reports C2B results and B2B checks (clamp-centric reporting). The report format in *esd_fail.rpt/esd_pass.rpt* is :

```
# CLAMP: <X> <Y> <LAYER> <NET> <LocID> <CLAMP_INST>
# <R> <X> <Y> <LAYER> <NET> <BUMP_NAME>
```

```
CLAMP: 2606.25 3170 METAL3 VDD ADS_VDD cinst_123(diode)
      5.65142 1616 4968 METAL6 VDD DVDD31
      4.79307 4316 39 METAL6 VDD DVDD30
```

...

REPORT_C2C : default 0; reports C2C results and B2B checks

B2B_PAIR_COUNT <num>: defines the maximum number of B2B pairs to be computed. The default pairs limit is 100,000, which is a “soft limit” in that the actual B2B count is based on the RADIUS value, but it should be close to the specified number in practice. Whether a RADIUS is specified or not, when the B2B count exceeds the B2B_PAIR_COUNT value, PathFinder automatically adjusts the RADIUS value so that the B2B count is close to the B2B_PAIR_COUNT. When a small RADIUS is used, and no B2B pair exists within the specified RADIUS, the RADIUS is automatically increased to compute about 10% of B2B_PAIR_COUNT (or 10% of the pairs in the design, whichever is smaller).

B2B_LOOP_COUNT: controls the number of lowest-R bump-clamp and bump-bump loops (loops with loop resistances less than LOOP_R) in the output report. The default for B2B_LOOP_COUNT is 5.

MAX_ESD_STAGE : specifies the maximum number of stages that can be checked (default 5).

CLAMP_SELECT : specifies the method of selecting nodes for clamp cells in each net using the minimum or maximum resistance from grid checking. Used only if x,y location is not specified for all the pins in a domain. Default is minimum.

EM_MODE : specifies ‘avg’ or ‘peak’ values to be used for current density checking (default peak). If EM MODE is specified in both the GSR and in the ESD rules file, the rules file setting has priority.

DETAILED_MODE : B2C reports a pair as failing only if all the arc resistances to it are failing. In other words, if at least one arc to a clamp passes (arc resistance less than or equal to ARC_R), then all arc resistances for that clamp are reported as passing. If this keyword is set to 1, then if any of the arcs fail, a clamp failure is reported in *esd_fail.rpt*. Default off.

SHORT_BUMP_IN_NET: specifies a one or more nets for which all bumps are shorted together to calculate minimum resistance, and the minimum resistance path is then traced from the clamp to the nearest bump on the

shorted net. Multiple entries of this keyword can be used to specify multiple nets, but only nets specified in a single rule entry are shorted together.

SHORT_BUMP_IN_NET_GROUP: shorts bumps together in each domain in the specified group. If 'All' is specified, shorts bumps in all domains in the specified group together.

NO_SHORT_BUMP_IN_NET : allows excluding specified nets otherwise shorted as a group using SHORT_BUMP_IN_NET_GROUP.

SHORT_MODE : when set, produces a one-line text report for each B2B pair, with the format:

```
(<BUMP1> <X> <Y> <LAYER> <NET>) <-> (<BUMP2> <X> <Y>
<LAYER> <NET>) <para_R>
```

B2B_MODE : specifies N- stage B2B checking. Default 0, Uses original methodology and reporting for 1- and 2-stage B2B checks.

ALL_PAD_NODE : controls node selection when plocs are created from pins in pcells. If set, all ploc nodes are picked up for resistance and CD checks. By default, only one node is picked up for each pcell pin

SAME_CLAMP_INST & DIFF_CLAMP_INST : to compute and report resistances between same/different clamp instances when more than one point are declared on a pin of clamp cell, for C2C resistance computation and ESD-CD analysis. Both keywords are default 0 (off), i.e, will report all clamp-to-clamp resistances.

Note: PathFinder recognizes wildcard characters in the rule file for keywords related to domain filtering, to allow you to more easily filter nets in ESD analysis. The ESD keywords that support wildcards are:

- DOMAIN
- B2B_NET1 (previously 'FROM_NET')
- INTERNAL_NET
- SHORT_BUMP_IN_NET
- TERMINAL_NET
- B2B_NET2 (previously 'TO_NET')

Sample Rules File

An example of the syntax used in the ESD rules file for the types above is shown below:

```
BEGIN_ESD_RULE
  TYPE [ BUMP2BUMP | BUMP2CLAMP | CLAMP2CLAMP ]
  # Note that a TYPE statement is required
  # Syntax for CLAMP2INST and CLAMP2MACRO rule types are described below
  NAME rules1 # the reserved TYPE names are not allowed for the rule name
  RADIUS 1000 # described following
  ARC_R 1.5 # resistance check threshold for BUMP2CLAMP or CLAMP2CLAMP
  LOOP_R 10.85 # loop resistance check threshold for BUMP2BUMP
  PARALLEL_R 2.45 # parallel resistance check threshold for BUMP2BUMP
  ESD_STAGE <min> <max>
  B2B_LOOP_COUNT <count>
  MAX_ESD_STAGE <stage>
  PACKAGE 1
END_ESD_RULE
```

For example:

```
BEGIN_ESD_RULE
  TYPE BUMP2BUMP
```

```

NAME    rules1
RADIUS   1000
LOOP_R   10.85
PARALLEL_R 1.45
ESD_STAGE 1 6
B2B_LOOP_COUNT 4
MAX_ESD_STAGE 6
PACKAGE 1
END_ESD_RULE

```

For rules1, BUMP2BUMP is checked, since LOOP_R and PARALLEL_R are defined. The *esd.clamp* file contains the clamp cell definition, as described in the following section.

Resistance Rule Checking Results

Resistance rule checking report output files from the ‘report esdcheck’ command have the following formats:

For B2C|C2C:

```
<Rule_Name> <ESD_source> <x> <y> <Layer> <ESD_sink> <x> <y> <Layer> <ARC_R>
```

For B2B:

```
<Rule_Name> <ESD_source> <x> <y> <Layer> <ESD_sink> <x> <y> <Layer> <ARC_R1>
<LOOP_R> <PARALLEL_R> <ARC_R2>
```

By default for B2B or B2BM PathFinder reports only the best resistance loop. If all loop reports are desired, you can specify the ‘-detail’ option. With this option, the report appears has the following format:

For B2B:

```
<Rule_Name> <ESD_source> <x> <y> <Layer> <ESD_sink> <x> <y> <Layer> <ARC_R1>
<LOOP_R1> <PARALLEL_R> <ARC_R2>
<Rule_Name> <ESD_source> <x> <y> <Layer> <ESD_sink> <x> <y> <Layer> <ARC_R1>
<LOOP_R2> <PARALLEL_R> <ARC_R2>
```

```
(esd_stage > 1)
# For B2B<Stage:x>:
<Rule_Name> <ESD_source> <x> <y> <Layer> <ESD_sink> <x> <y>
<Layer> <ARC_R1> <LOOP_R1> <PARALLEL_R> <ARC_2> <ARC_3>
```

Resistance Checking for CLAMP2INST (C2I) and CLAMP2MACRO (C2M) Rules

The same basic resistance checking command, ‘perform esdcheck’, is used for the bump rules as for clamp rules (see [section “perform esdcheck”, page 16-448](#) for detailed syntax).

Use the ‘CHECK_CONNECTION 1’ rule setting to invoke this check.

Rules File Syntax for Types C2I and C2M

A description of the syntax in the ESD rules file for CLAMP2INST and CLAMP2MACRO checking is shown below:

```

BEGIN_ESD_RULE
TYPE [CLAMP2INST(or C2I) | CLAMP2MACRO (or C2M)]
NAME <rule_name>
ARC_R <maxR>

```

```

THREAD_BY_NET [ 0 | 1 ]
LOOP_R <R_Ohms>
LAYER [TOP | BOTTOM | <layer>]
DOMAIN <net1> [<net2> ... ]
DOMAIN_GROUP { POWER | GROUND | SIGNAL | <userDefinedGroupName> ]
CLAMP_CELL <cell1> [<cell2> ... ]
CLAMP_INST_FILE <filename>
CLAMP_INST_NAME <inst1> [<inst2> ... ]
CLAMP_AREA <llx> <lly> <urx> <ury>
SHORT_CLAMP_NODES [0|1]
CELL <cellname1> [<cellname2> ... ]
CELL_FILE <filename>
INSTANCE <inst1> [<inst2> ... ]
INST_FILE <filename>
AREA <llx lly urx ury>
SAMPLE_INST <%_instances_sampled>
INST_COUNT <num_instances_sampled>
SAMPLE_POINT <%_points_per_inst>
POINT_COUNT <num_points_per_inst>
SAMPLE_MODE [ UNIFORM ?<region_num>? | RES_BY_AREA ?<region_num>? ]
B2B_LOOP_LENGTH <clamp_count1> <clamp_count2> ...
PACKAGE 1
PACKAGE_PIN_FILE <pkg_fileName>
EXCLUDE_CELL <cellName>
EXCLUDE_CELL_FILE <fileName>
EXCLUDE_INSTANCE <instName>
EXCLUDE_INSTANCE_FILE <fileName>
CLAMP_SELECT [min| max ]
IGNORE_MID_BLOCK [0|1 ]
STDCELL_SELECT [min| max ]
MACRO_SELECT [min| max ]
END_ESD_RULE

```

where

TYPE : specifies the checking rule type, either CLAMP2INST(C2I) or CLAMP2MACRO (C2M). Required

NAME : user-specified rule name; reserved TYPE names are not allowed as rule names. Required input that specifies the resistance check threshold (Ohms) for BUMP2CLAMP or CLAMP2CLAMP rule, and CLAMP2INST and CLAMP2MACRO checks. If specified, the value overrides the arc_R value defined in the ESD rules file.

ARC_R : specifies resistance check threshold (Ohms). If not specified, defaults to 0.5X LOOP_R. Either ARC_R or LOOP_R must be given.

THREAD_BY_NET : by default (0), allows using more than one thread per net when arc-based C2I analysis is performed (if set to 1, allows only one thread per net)

LOOP_R: loop resistance checking threshold (Ohms) of a single ESD path . Overrides the loop_R value defined in the ESD rules file. If not specified, defaults to 2X ARC_R. Either ARC_R or LOOP_R must be given.

LAYER [TOP| BOTTOM | <layer>] : checks nodes of pins on the selected layer

DOMAIN <net1> [<net2> ...] : specifies only nets to be considered

DOMAIN_GROUP - allows specifying a group of domain nets of a certain type to be analyzed in arc-based C2I (and other checks that support the DOMAIN option). Also allows user-defined and named net groups, using the BEGIN_NET_GROUP rules definition.

CLAMP_CELL <cell1> [<cell2> ...] : specifies clamp cells. Multiple entries can be used. ESD clamp cells can be automatically identified and extracted from GDS by specifying marker layers in the layer map used in GDS2DEF. The presence of a marker layer inside the cells causes GDS2DEF to classify them as ESD devices and output them in LEF/DEF format.

CLAMP_INST_FILE <filename> : specifies an instance file filename. Multiple entries can be used.

CLAMP_INST_NAME <inst1> [<inst2> ...]: specifies clamp instance names to be considered. Multiple entries can be used.

CLAMP_AREA <l1x l1y urx ury> : specifies clamp area in a bbox to be considered.

SHORT_CLAMP_NODES : supports ESD-R and ESD-CD checks by shorting clamp nodes “on-the-fly” when the resistances are computed.

CELL : specifies one or more cell names to be checked. Wildcard “*” can be used to represent one or more characters to specify groups of cells.

CELL_FILE: specifies a file that lists cells to be checked. Wildcard “*” can be used to represent one or more characters to specify groups of cells.

INSTANCE <inst1> [<inst2> ...] : specifies core instance names to be considered. Multiple entries can be used. Wildcard “*” can be used to represent one or more characters to specify groups of instances.

INST_FILE <filename>: specifies a file that contains core instance names. Multiple entries can be used. Wildcard “*” can be used to represent one or more characters to specify groups of instances.

ARC_R <maxR> : specifies the threshold for arc resistance checking; required for B2C, C2C, C2I, and C2M rules (resistance above threshold considered a failure)

AREA <l1x l1y urx ury> : specifies a core area in a bbox to be considered

SAMPLE_INST : specifies the percentage of instances in the design to sample in analysis in C2I analysis.

INST_COUNT : specifies the number of instances to sample in the design

SAMPLE_POINT: specifies the percentage of points per instance to be sampled for macro instances. At least one point per net per instance is sampled.

POINT_COUNT: specifies the number of points per instance to be sampled for macro instances. At least one point per net per instance is sampled.

SAMPLE_MODE : by default C2I samples at least one node from each net of every macro-instance. The distribution of instance sampling can be modified using the options:

UNIFORM: samples an equal number of instances from each design region

RES_BY_AREA: samples more instances from design regions with higher gridcheck resistance

<region_num>: specifies the number of regions to be created in the design for either UNIFORM or RES_BY_AREA modes. For example, if the <region_num> is set to 5, the design is uniformly partitioned into 5 X 5 = 25 regions. The default <region_num> is 10 (100 regions), and the maximum region number setting is 31 (961 regions).

B2B_LOOP_LENGTH : <clamp_countN> values specify the number of clamps in each B2B path (positive integers less than or equal to MAX_ESD_STAGE).

Values of <clamp_countN> *not included* mean that B2B paths with that number of clamps are excluded from parallel R computation.

ARC_R <maxR>: specifies the maximum allowable arc resistance.

PACKAGE 1 : includes the package subcircuit in analysis

PACKAGE_PIN_FILE <pkg_fileName>: specifies a file identifying nodes in the package subcircuit that should be treated as external pins, so that ESD checks compute resistances beginning and ending at those nodes.

EXCLUDE_CELL <cellName> : excludes instances with specified cell names. Wildcard "*" can be used to represent one or more characters to specify groups of cells.

EXCLUDE_CELL_FILE: specifies a file that lists cells to be excluded from checks. Wildcard "*" can be used to represent one or more characters to specify groups of cells.

EXCLUDE_INSTANCE <instName> : specifies instances to be excluded. Wildcard "*" can be used to represent one or more characters to specify groups of instances.

EXCLUDE_INSTANCE_FILE <fileName>: specifies a file name that contains a list of instance names to be excluded from CDM checks. Wildcard "*" can be used to represent one or more characters to specify groups of instances.

CLAMP_SELECT : specifies the method of selecting nodes for clamp cells in each net using the minimum or maximum resistance from grid checking. Used only if x,y location is not specified for all the pins in a domain. Default is minimum.

IGNORE_MID_BLOCK : filters mid-level block instances from the instance list for C2M/C2I analysis. With this option specified, PathFinder checks B2I/C2I/ C2M only for leaf level instances. If the instance list or native sampled instance list contains block instances, they are filtered out. Also, it reports the selected instances in the *esd_info.rpt* file in the *adsRpt/ESD* directory by default, or in the specified output directory. Default: off (0)

Note that for MMX blocks, where the original cell is a DEF cell, it has an 'adsU1' instance in it, and selecting the adsU1 instance in this case is the only way for PF-S to perform the analysis, regardless of the IGNORE_MID_BLOCK flag setting.

STDCELL_SELECT: for C2I, selects nodes with maximum or minimum path resistance, as computed by gridcheck. Default is minimum.

MACRO_SELECT : for C2M, selects nodes with maximum or minimum path resistance, as computed by gridcheck. Default is maximum.

Examples of Rule Checking

Assume two instantiated clamp instances C1 and C2 in a design specified in the clamp file using '-clamp' option, and three core instances, I1, I2, and I3, specified in the file *inst.list*. Several examples of rule files for this case are given following.

Example 1 for C2I Rule Check

To run a C2I rule check, run ESD check on the three core instances I1, I2 and I3 to all clamp instances, including C1 and C2. Use the following settings in the rule file:

```
BEGIN_ESD_RULE
  NAME core_usage_1
  TYPE C2I
  INST_FILE inst.list
  ARC_R 2.3
```

```
LOOP_R 4
END_ESD_RULE
```

Example 2 for C2I Rule Check

Run an ESD check on the three core instances I1, I2 and I3 to the clamp instance 'C1' while a clamp file is defined with '-clamp' option on the 'perform esdcheck ...' command line. Use the following rule settings:

```
BEGIN_ESD_RULE
  NAME core_usage_2
  TYPE C2I
  CLAMP_INST_NAME C1
  INST_FILE inst.list
  ARC_R 2.3
  LOOP_R 4
END_ESD_RULE
```

Example 3 for C2I Rule Check

Run an ESD check on the full chip, with automatic instance sampling to all the instantiated clamp instances from the clamp file defined with the '-clamp' option in 'perform esdcheck ...' command line. You can optionally control the instance sampling percentage using the SAMPLE_INST keyword. Use the following rule settings:

```
BEGIN_ESD_RULE
  NAME core_usage_2
  TYPE C2I
  ARC_R 2.3
  LOOP_R 4
  SAMPLE_INST <value in %>
END_ESD_RULE
```

Example 1 for C2M Rule Check

Run an ESD check on the three macro instances I1, I2 and I3 to all clamp instances instantiated from a clamp file defined with the '-clamp' option in 'perform esdcheck ...' command line. Use the following rule settings:

```
BEGIN_ESD_RULE
  NAME C2M_usage_1
  TYPE C2M
  INST_FILE inst.list
  LAYER BOTTOM # choose nodes from the bottom layer
  SAMPLE_POINT 20 #
  ARC_R 2.3
END_ESD_RULE
```

Note in this case 20% of nodes are selected; if this option is not specified, automatic sampling is performed that considers both resistance weakness and a reasonable run time.

Example 2 for C2M Rule Check

Run an ESD check on three MMX instances I1, I2 and I3 (can be a Custom Macro Model from [Totem](#); refer to the “Totem User Manual”) to all clamp instances instantiated from a clamp file defined with the ‘-clamp’ option in ‘perform esdcheck ...’ command line. To check transistor pins in MMX instances, use the ‘TRANSISTOR_PIN 1’ rule setting, as shown in the rule file following.

```
BEGIN_ESD_RULE
    NAME C2M_usage_1
    TYPE C2M
    INST_FILE inst.list
    TRANSISTOR_PIN 1
    LAYER BOTTOM # choose the nodes from the bottom layer
    SAMPLE_POINT 20 # with 20% of node selections; if not
    specified, automatic sampling that considers resistance
    weakness in a reasonable run time
    ARC_R 2.3
END_ESD_RULE
```

Connectivity Checking

For C2I and C2M modes, the rule ‘CHECK_CONNECTION 1’ is provided to check the connectivity of core instances to the clamp instances. The rule and clamp files are the same as for a C2I or C2M run. Example rule file:

```
BEGIN_ESD_RULE
    NAME C2I_rule
    TYPE CLAMP2INST
    CHECK_CONNECTION 1
END_ESD_RULE
```

CLAMP2INST Connectivity

Most of the other C2I or C2M option/rule entries are inactive with the ‘-checkConn’ option, except for the ones that are related to clamps, such as clamp cells/insts/pins. For a given set of clamps, these are checked for all instances in the design.

1. If an instance is connected to clamp(s) for both power and ground, it lists those that are completely disconnected or partially disconnected. For example, in the checkConn report, the PWR and GND status are reported separately, so if either P or G is not connected to clamp, it is reported. An example is:

```
# POWER      GROUND    <instName>
PWR_OK     NO_GND    I1/I2
NO_PWR     GND_OK    I3/I4
NO_PWR     NO_GND    I5/I6
```

2. For the connected instances, the ones that are marked as “macro” as determined by [RedHawk](#), are listed first. A “macro” is usually a memory, IP, or very large cell, which is automatically selected for C2M analysis if there is no user-specified instance file.
3. All switch instances in the design are listed so you know that it is justified for switch instances to have a one-ended connection in the loop-R report. When the rule type is CLAMP2INST, PF-S checks and reports the selected node per domain in each instance that is not connected to any clamp instances. The connectivity info is recorded in the *adsRpt/ESD/esd_info.rpt* file.

A sample Instance and Connectivity Report for Rule <C2I> (CLAMP2INST) follows:

```
BEGIN_ESD_RULE
  NAME C2I
  TYPE CLAMP2INST
  CHECK_CONNECTION 1
END_ESD_RULE

# Total number of unprotected instances: 8
# POWER GROUND <INSTANCE>
NO_PWR NO_GND inst_129199
PWR_OK NO_GND inst_129203
NO_PWR NO_GND inst_129208
NO_PWR GND_OK inst_129216
...

# Total number of macro instances: 44
# MACRO INSTANCE NAMES:
inst_129400/adsU1
inst_129422/inst_7693
...

# Total number of switch instances: 884
# SWITCH INSTANCE NAMES:
inst_129973/switch_inst_R1_C1
inst_129973/switch_inst_R2_C1
inst_129973/switch_inst_R3_C1
...
```

CLAMP2MACRO Connectivity

When the rule type is CLAMP2MACRO, PF-S checks and reports points (nodes) in macros (as automatically determined in the RedHawk database, or as specified with INSTANCE_FILE or ‘-instfile <file>’), that are not connected to any clamp instances. The report format is:

```
# Macro Instance and Connectivity Report for Rule <C2IM>
(CLAMP2MACRO)

BEGIN_ESD_RULE
  NAME C2IM
  TYPE CLAMP2MACRO
  CHECK_CONNECTION 1
END_ESD_RULE

# INST <NAME> <X1 Y1 X2 Y2> <NUM_UNPROTECTED_POINTS>
# <XY LOCATION> <LAYER> <DOMAIN> <PIN>
INST I1/u428/I356 1931.8 0 3900 3915 103
2572 3516.6 m5 vdd vdd
2658.4 3574.2 m5 vss vss
2658.4 2969.4 m5 vdd1 vdd1
2572 3574.2 m5 vvdd vdd
...
```

Summary: Found 3 instances with 204 unprotected points.

When the '-transistor' option is used, only nodes on transistor pins for MMX blocks are checked and the transistor names are listed in the report:

```
# INST <NAME> <X1 Y1 X2 Y2> <NUM_UNPROTECTED_POINTS>
# <XY LOCATION> <LAYER> <DOMAIN> <PIN> <TRANSISTOR>
INST FIFO0/TX0FCORE/ram64/adsU1 1757.36 785.035 1903.34 915.375 105
1818.41 871.955 METALL1 VSS VSS.gds299 X0.X67.X0.X2.M5
1844.05 858.335 METALL1 VDD VDD.gds1651 X50.X0.M30
1859.39 858.335 METALL1 VDD VDD.gds1729 X53.X0.M30
```

Combined Rules and Clamp Cell Pin Location File

The ESD rules file, which defines the rule name, and the clamp cell pin location file, which specifies the pin location of each clamp cell, can also be merged into a single file, as specified in the '-rule' option. The combined file can be a simple concatenation of the two files as described above. The option '-clamp' is not needed for a combined file.

Node creation on Clamp Pins

You can create nodes at specific locations on clamp pins to get accurate arc resistance, as sometimes a node is created at a distant location and thus resistance calculation may be incorrect. The requirements for this operation are:

- The x, y location and layer must be specified in the Pin statement in the ESD clamp file to create a specific new node.
- The ESD clamp filename must be specified with the GSR keyword:

```
ESD_CLAMP_FILES {
    <clamp_file_name>
}
```

An example clamp file follows:

```
BEGIN_CLAMP_CELL
    NAME PVSS_D1
    PIN VDD 25 180 M3
    ESD_PIN_PAIR VDD VSS PP
END_CLAMP_CELL
```

Sample Invocation

There are two ways to invoke ESD checking in RedHawk, either through the GUI under the menu command **Tools->Path Finder S->ESD Resistance Check**, or using the TCL command 'perform esdcheck'.

When the menu command is used the ESD Resistance Check' dialog box is displayed, as shown in Figure 16-5.

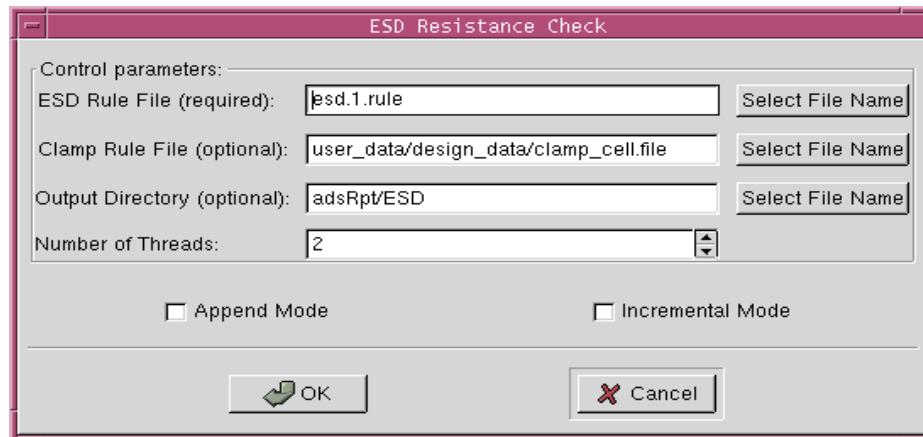


Figure 16-5 ESD Resistance Check dialog box

Resistance Checking for BUMP2INSTANCE Rules

PathFinder also supports Bump-to-Instance resistance checks. For every bump, it compares the resistances to instances and the resistance to its protecting clamp. The protecting clamp of a bump is the clamp with the smallest resistance to the bump. If a bump has any resistance to an instance that is smaller than the resistance to its protecting clamp, the bump fails the B2I check. The same ESD resistance checking command, ‘perform esdcheck’, is used for performing B2I checks. The most common options used for performing B2I checking are similar to those for performing C2I checking.

Rule File Syntax for Types B2I

A description of the syntax in the ESD rules file for BUMP2INSTANCE checking is shown below:

```
BEGIN_ESD_RULE
    TYPE [ BUMP2INST | B2I ]
    NAME <rule_name>
    B2C_NAME <b2c_rule_name>
    B2I_ARC_COUNT <b2i_arc_num>
    CELL <cellname1> [<cellname2> ... ]
    CELL_FILE <filename>
    INSTANCE <inst1> [<inst2> ... ]
    INST_FILE <filename> B2I_ARC_COUNT <b2i_arc_num>
    ZAP_VOLTAGE <voltage_Volts>
    ZAP_R <resistance_Ohms>
    ZAP_CURRENT <current_in_Amps>
    EXCLUDE_CELL <cellName>
    EXCLUDE_CELL_FILE <fileName>
    EXCLUDE_INSTANCE <instName>
    EXCLUDE_INSTANCE_FILE <fileName>
    EXCLUDE_BUMP <bump1> ... <bumpN>
    BUMP_FILE <filename>
```

```

EXCLUDE_BUMP_FILE <filename>
PAD_FILE <ploc/pcell_filename>
EXCLUDE_PAD_FILE <ploc/pcell_filename>
DOMAIN <net1> <net2> ...
TRANSISTOR_PIN [ 0 | 1 ]
B2I_REPORT_PARTIAL_PASS [ 0 | 1 ]
B2I_MACRO_MODE [ 0 | 1 ]
RES_RATIO <ratio>
SAMPLE_INST <%_instances_sampled>
INST_COUNT <num_instances_sampled>
SAMPLE_MODE [ UNIFORM ?<region_num>? | RES_BY_AREA ?<region_num>? ]
END_ESD_RULE

```

where

B2C_NAME: specifies the previously-performed B2C rule name to obtain the resistance to the protecting clamp of each bump. If this keyword is not specified, B2C resistance for the bumps is computed.

B2I_ARC_COUNT: specifies the number of B2C arcs to be reported in *esd_pass.rpt* and *esd_fail.rpt*. The default number of B2C arcs reported is 5.

CELL : specifies one or more cell names to be checked. Wildcard “*” can be used to represent one or more characters to specify groups of cells.

CELL_FILE: specifies a file that lists cells to be checked. Wildcard “*” can be used to represent one or more characters to specify groups of cells.

INSTANCE <inst1> [<inst2> ...] : specifies core instance names to be considered. Multiple entries can be used. Wildcard “*” can be used to represent one or more characters to specify groups of instances.

INST_FILE <filename>: specifies a file that contains core instance names. Multiple entries can be used. Wildcard “*” can be used to represent one or more characters to specify groups of instances.

ZAP_VOLTAGE: specifies the source voltage for nonlinear clamp I-V cases (including cases with different RON and ROFF for ESD_PIN_PAIR)

ZAP_R: specifies the series resistance to the zapping source resistance for nonlinear clamp I-V cases.

ZAP_CURRENT: specifies the current of the zapping source for nonlinear clamp I-V cases. When this is used, ZAP_VOLTAGE and ZAP_R keywords are ignored.

EXCLUDE_CELL <cellName> : excludes instances with specified cell names. Wildcard “*” can be used to represent one or more characters to specify groups of cells.

EXCLUDE_CELL_FILE: specifies a file that lists cells to be excluded from checks. Wildcard “*” can be used to represent one or more characters to specify groups of cells.

EXCLUDE_INSTANCE <instName> : specifies instances to be excluded. Wildcard “*” can be used to represent one or more characters to specify groups of instances.

EXCLUDE_INSTANCE_FILE <fileName>: specifies a file name that contains a list of instance names to be excluded from CDM checks. Wildcard “*” can be used to represent one or more characters to specify groups of instances.

EXCLUDE_BUMP : list of bumps to be excluded in checking

BUMP_FILE : name of file that lists bumps to be included in checking

EXCLUDE_BUMP_FILE : name of file that lists bumps to be excluded in checking

PAD_FILE : specifies ploc/pcell file listing bumps to be included in ESD checks
EXCLUDE_PAD_FILE : excludes consideration of checks in specified pad file
DOMAIN : list of nets to be checked. Others are excluded.
TRANSISTOR_PIN : when set, transistor names are included in the *esd_pass.rpt* and *esd_fail.rpt* reports, and also in the *esd_inst.rpt* file.
B2I_REPORT_PARTIAL_PASS : when set to 1 (default is 0), “partial pass” bumps are reported in *esd_pass.rpt*. A partial pass bump is a bump with some failing B2I arcs. By default, only bumps with no failing B2I arcs are reported in *esd_pass.rpt*.
B2I_MACRO_MODE : when set to 1 (default is 0), multiple points per net per instance for larger instances (macro instances) are sampled. By default only one point per net per instance is sampled.
RES_RATIO : specifies the value at which the “Resistance (B2C path)/Resistance (B2I path)” is reported as failed.
SAMPLE_INST : specifies the percentage of instances in the design to sample in analysis in B2I analysis.
INST_COUNT : specifies the number of instances to sample in the design
SAMPLE_MODE : by default B2I samples at least one node from each net of every macro-instance. The distribution of instance sampling can be modified using the options:
 UNIFORM: samples an equal number of instances from each design region
 RES_BY_AREA: samples more instances from design regions with lower gridcheck resistance
 <region_num>: specifies the number of regions to be created in the design for either UNIFORM or RES_BY_AREA modes. For example, if the <region_num> is set to 5, the design is uniformly partitioned into 5 X 5 = 25 regions. The default <region_num> is 10 (100 regions), and the maximum region number setting is 31 (961 regions).

B2I rules also support all instance sampling rule keywords used for C2I/C2M, such as INSTANCE and AREA.

The associated ‘report esdcheck’ command syntax is :

```
report esdcheck -type b2i -name esd_b2i_rule -b2cName esd_b2c_rule
```

The default output directory is *adsRpt/ESD/*, which can be changed using the ‘-outDir’ option. By default in the *esd_pass.rpt* and *esd_fail.rpt* report files five instance names and five B2I arcs are specified. You can control this feature using the rule keyword ‘B2I_ARC_COUNT’.

Resistance Checking from Bumps to Cross-domain Driver-receivers

Introduction

Cross-domain driver-receiver design is generally susceptible to CDM discharging events. A typical design example that includes a VDD1 pin discharging in a CDM event is shown in Figure 16-6.

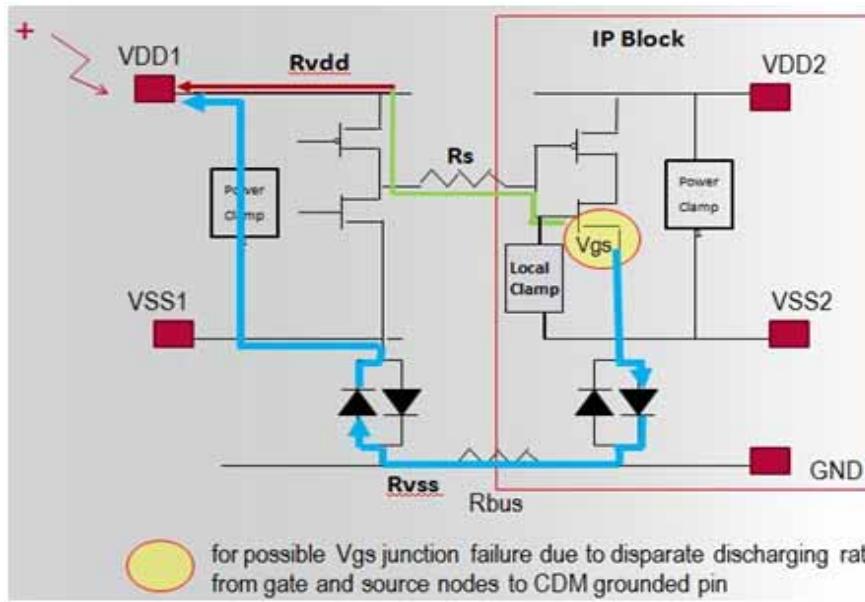


Figure 16-6 Cross-domain driver-receiver discharging paths

Usually, the Vgs junction on the receiver side sees the most stress, potentially leading to oxide breakdown (particularly for thin-oxide) due to different discharging rates from gate and source nodes to the CDM grounded pin. The common solution to this potential weakness includes insertion of resistance from the driver output of driver to the fan-in of the receiver, resulting in a voltage divider effect on the stress seen across the Vgs junction. In addition, a local clamp such as a GGNMOS can be added between the gate, the local ground for both parasitic BJT snapback discharging paths, and a parasitic diode to ground. A potential significant problem in today's multiple-IP multiple-domain SoC designs is that the driver is outside of the driven macro where the receiver resides. So the responsibility for protecting the cross-domain driver-receiver system therefore falls on the SoC designer to provide the needed ESD protection. The same macro also can be reused in different SoC designs, and therefore the Vgs stress observed is different in this case. One of the key cross-domain driver-receiver CDM failure mechanisms can be caused by a very unbalanced impedance path from the gate and source node of the Vgs junction to the CDM grounded pin. This is the reason for the following checks on cell-based designs as the first step in assessing this potential weakness.

The first check for cross-domain driver-receiver systems is focused on the cell-based cross-domain driver-receivers, as illustrated in Figure 16-7. The check compares the resistance value of path A1 versus A2, and path B1 versus B2, in absolute value, and also in percentage of R difference. After you provide a resistance difference threshold, PF-S highlights those driver-receiver pairs that failed the check. When there are not sufficient core clamps, you may see a large resistance difference for the cross-domain driver-receiver pair, and this may contribute to failure in a CDM discharging event.

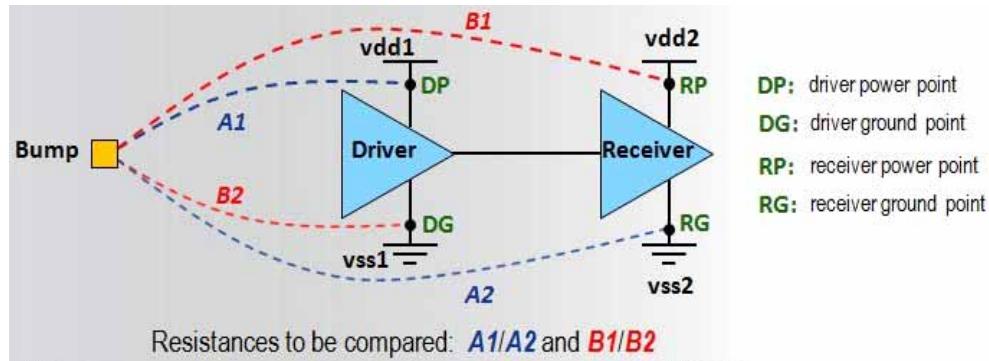


Figure 16-7 Cell-based cross-domain driver-receiver checking

Cross-domain Resistance Check Rule Keywords

Rules for checking driver-receiver circuits that cross voltage domains are called DRIVER2RECEIVER (D2R) rules, and are used to check the difference or the ratio of the resistance from P/G pins of cross-domain driver-receiver instance pairs to the resistance of all bumps in designs that are connected directly or through clamps. D2R ESD rule keywords are:

D2R_P2P_CHECK [1 |0] : activates driver-receiver point-to-point P/G resistance checks. Default: 0.

D2R_DIFF <res_diff>: specifies the resistance difference between driver and receiver P/G nets as a pass/fail threshold.

D2R_RATIO <res_ratio>: specifies the resistance ratio between driver and receiver P/G nets as a pass/fail threshold.

D2R_PAIR_MODE [ALL |SAME |GND(VDD)_SAME |GND(VDD)_DIFF]: specifies the type of driver-receiver pairs to be checked (if not specified, by default ALL driver-receiver pairs with different P/G domains are checked). The values are:

ALL - all driver-receiver pairs are checked.

SAME = all driver-receiver pairs of the same type are checked

GND(VDD)_SAME - driver-receiver pairs with the same GDN/VDD domains are checked

GND(VDD)_DIFF - driver-receiver pairs with different GDN/VDD domains are checked.

D2R_PG_MODE [P2G | G2P | G2G| P2P] : if not specified, all resistance pairs are checked.

P2G - only resistance pairs of drivers' POWER nets and receivers' GROUND nets are checked

G2P only resistance pairs of receivers' POWER nets and drivers' GROUND nets are checked

G2G - computes only ground-to-ground point pairs

P2P - computes only power-to-power point pairs

D2R_PAIR_COUNT <number> : specifies the maximum number of estimated worst-case D2R point pairs to be computed. Default: 1000.

D2R_MIN_BOUND [1 |0] : activates the capability to reduce total pair count for arc resistance calculation (if the ARC_R rule keyword is also specified). Default: 0.

D2R_NODE_FILE <node_file>: specifies a file listing driver-receiver node pairs to be checked. The node file syntax is as follows:

```
<Inst1> <Pin1/Tran1> ?<x> <y> ?<layer>? ? <inst2> <Pin2/Tran2> ?<x> <y>
?<layer>? ?
```

Note that one of either D2R_DIFF or D2R_RATIO is required for valid D2R checking, and that D2R_PAIR_MODE/D2R_PG_MODE and D2R_NODE_FILE keywords are mutually exclusive

An example rule file follows:

```
BEGIN_ESD_RULE
NAME d2r
TYPE D2R
D2R_DIFF 20
END_ESD_RULE
```

Multiple-clamp Zap R Checks

You can also perform ESD resistance checking with multiple point to multiple point zaps. An extra rule keyword, 'TYPE RESISTANCE', must be added in the rule file to enable this feature. Other rule syntax is similar to CD checking, but the new rule file entry needed is:

```
TYPE Resistance or
TYPE R
```

Results are reported in the same way as in ESD-CD checking, as follows:

```
ESD Resistance Check <b2ccd_max_4> (Rule: b2ccd_res):
Zapping source: 1.2 A
Equivalent resistance: 1.41216 Ohms (1.69459 V, 1.2 A) (PASS)
Zap from: Bump(DVSS4) (4880 1298.85 METAL4 VSS)
Zap to: Clamp(inst_1, to_vss) (4451.41 2267.57 METAL4 VSS)
Zap to: Clamp(inst_3, avss) (4601.04 3172.46 METAL3 VSS)
```

The Tcl command 'report esdcheck' supports the keyword -type R, as follows:

```
report esdcheck -type R -summary
report esdcheck -type R -name <rulename> ?-arcR <value>
```

ESD Resistance Checking Reports

Report esdcheck command

After an ESD resistance check is performed with the 'perform esdcheck' command, RedHawk saves the results into the DB. The data is identified by the rule name. You can get text reports for any ESD resistance check with the TCL command 'report esdcheck' to specify the type of reports to be created :

```
report esdcheck -type <ruleType> ? -outDir <output_dir>?
? -name <rule_name>? ?-[glob |-regexp ] ?
? -arcR <arc_Res_thres>? ? -loopR <loop_Res_thresh>?
? -parallelR <parallel_threshold>? ? -append? ? -excel ?
? -failedOnly? ? -detail?
```

where

- type: reports results for the specified type of ESD checks. Default is all types.
- outDir : specifies the report files directory. Default is *adsRpt/ESD*

- name: specifies the name of the rule tested. If multiple checks have been specified within a single rule-- for example, two bump pairs are specified in the same rule-- PathFinder internally splits them into sub-rules, each with a single bump pair. Then it appends '(_\d+)+' to the base rule name. So when using 'report esdcheck', you can specify this derived name to look at the particular results desired.
- glob|regexp: specifies the type of wildcard expressions used. PathFinder reports the results for all subrules that match the specification; 'glob' supports regular wildcard expressions and 'regexp' supports normal wild cards and also extended character strings. For example:


```
report esdcheck -detail -perNet -type B2C -name abc* -glob
report esdcheck -type B2C -name {M_test_[3-5]} -regexp
```
- arcR: specifies the threshold for arc resistance checking; required for B2C, C2C, C2I, and C2M rules (resistance above threshold considered a failure)
- loopR: specifies the threshold for loop-R checking; required for C2I
- parallelR: specifies the threshold for parallel-R checking; required for B2B
- append: appends results to existing results in the output directory.
- excel : creates a consolidated ESD report file in Excel format, *adsRpt/ESD/esd_excel.rpt*, for resistance and CD checks, so that the information can be easily sorted and processed using Excel.
- failedOnly: only reports failed results.
- detail : provides more detailed report results for B2B rules

You can perform basic filtering of the resistance checking results based on nets, clamp instances and clamp types. The types of filtering rules supported are:

- CLAMP_CELL <cellName>: reports only B2C (C2C) arcRs that connect to clamp instances belonging to cell <cellName>
- CLAMP_INST <instName>: reports only B2C (C2C) arcRs that connect to clamp instance <instName>
- CLAMP_TYPE <clampType>: reports only B2C (C2C) arcRs that connect to clamp instances with type <typeName>
- TERMINAL_NET <netName>: reports only B2C (C2C) arcRs that connect to net <netName>

Resistance Checking Output Reports

PathFinder resistance checking generates five types of resistance check report files for each rule, describing pad information and the resistance of each ESD path, in the directory *adsRpt/ESD*, as follows:

- *ClampInfo.rpt* - information on ESD clamp cells and instances
- *esd_pass.rpt* - information on passed ESD arcs/loops for each rule, listing one arc/loop for each passed instance.
- *esd_fail.rpt* - information on failed ESD arcs/loops for each rule, listing one arc/loop for each failed instance.
- *esd_info.rpt* - more information on passing and failing instances
- *esd_summary.rpt* - contains a summary of the passing and failing results for each rule check.

ESD reports *esd_summary.rpt*, *esd_pass.rpt*, and *esd_fail.rpt* include the resistance range for pass/fail reporting for resistance checks.

Sample output:

```
Bump-to-clamp R check summary for rule <B2C.RULE>
(BUMP2CLAMP):
Rule: 1.2 Ohms, Min: 0.403649 Ohms, Max: 2.7844 Ohms
Fail count: 8 (25.00% of 32)
Failed bump-to-clamp resistance range: [1.2451, 1.4961]
```

Following are descriptions of different types of ESD reports for different rules.

B2I results reports

The results of a B2I check are generated after the ‘perform esdcheck’ command is executed with a B2I-type rule. The default output directory is *adsRpt/ESD/*. The results also can be generated by using the ‘report esdcheck’ command. An example ‘report esdcheck’ command follows:

```
report esdcheck -type b2i -name esd_b2i_rule -b2cName esd_b2c_rule
-outDir b2i_report
```

Using the ‘-b2cName’ option is the same as using the “B2C_NAME” rule keyword. A sample *esd_fail.rpt* is shown below:

```
# ESD Check Results for Rule <esd_b2i_rule> (BUMP2INST)

BEGIN_ESD_RULE
  NAME esd_b2i_rule
  TYPE BUMP2INST
END_ESD_RULE

# Bump: <B2C_R> <X> <Y> <LAYER> <NET> <Bump_NAME>
# <MIN_B2I_RES> <MAX_B2I_RES> <PASS/FAIL_RATE>
# <B2I_R> <XY LOCATION> <LAYER> <NET> <INST_NAME>
Bump: 1.3208 4880 938.85 METAL4 VSS DVSS1 0.7537 4.0961 0.29%(7/2391)
      0.7537 4690.91 925.52 METAL4 VSS inst_129425/inst_7769
      1.1636 4460.62 893.39 METALL1 VSS inst_149915
      1.1969 4562.74 893.39 METALL1 VSS inst_148929
      1.20924589.42 893.39 METALL1 VSS inst_149950
      1.2929 4459.7 893.39 METALL1 VSS inst_148901

# Bump: <B2C_R> <X> <Y> <LAYER> <NET> <Bump_NAME> <MIN_B2I_RES>
# <MAX_B2I_RES> <PASS/FAIL_RATE>
# <B2I_R> <XY LOCATION> <LAYER> <NET> <INST_NAME>
Bump: 1.3923 4880 1058.85 METAL4 VSS DVSS2 0.9712 4.3061 0.08%(2/2391)
      0.9712 4690.91 925.52 METAL4 VSS inst_129425/inst_7769
      1.3458 4460.62 893.39 METALL1 VSS inst_149915
```

Passing and failing bumps and arcs are summarized in the report *esd_summary.rpt*, as shown in the sample following:

```
# ESD Check Summary for Rule < esd_b2i_rule > (BUMP2INST)

BEGIN_ESD_RULE
  NAME esd_b2i_rule
  TYPE BUMP2INST
END_ESD_RULE

Bump summary:
```

```

        8 bumps are checked.
        8 failed bumps. (100.00%)
        0 passed bumps. (0.00%)

Instance summary:
    348382 instances.
        2391 instances computed completely. (0.69%)
        0 instances computed partially. (0.00%)
    345991 instances to be computed. (99.31%)

Arc-R summary:
    19128 arc-Rs are checked.
        74 failed arc-Rs. (0.39%)
    19054 passed arc-Rs. (99.61%)
=====
```

Clamp Info Reports

Detailed information on clamps is saved in a report file *adsRpt/ESD/ClampInfo.rpt* for B2B, B2C, C2C, C2I, and C2M rules. Since only clamp cell names are specified, PF-S saves all ESD instances found in *adsRpt/ESD/ClampInst.rpt*. ESD cells that are undefined in the clamp info file are reported in *adsRpt/ESD/esd_clamp_unused.rpt*.

Sample Clamp Info report (ClampInfo.rpt) for all ESD rule runs

For example:

```

RULE rules2 (CLAMP2INST):
BEGIN_ESD_RULE
    NAME rules2
    TYPE CLAMP2INST
    ARC_R 6
    LOOP_R 10
END_ESD_RULE
COVERAGE INFO:
Coverage area: 0 0 8640 9570
Number of instances: 782404
Number of selected instances: 4129
Number of valid instances: 4129
CLAMP CELL INFO:
BEGIN_CLAMP_CELL
    NAME TEST_VDDO_APACHECELL
    PIN VDDC 20 314 metal3
    PIN VSSC 20 10 metal3
    RON 0.1
END_CLAMP_CELL
CLAMP INSTANCE INFO:
CLAMP CELL: TEST_VDDO_APACHECELL {
#<INSTANCE> <X1 Y1 X2 Y2> <ORIENT>
#<XY LOCATION> <LAYER> <NET> <PIN> [<locID>]
chip_pads/test_VDDO_6_186/adsU1      5900.000000
0.000000      5930.000000      382.000000      N
5915.000000      313.000000      metal3  VDD      VDDC
5920.280000      10.000000      metal3  VSS      VSSC
}
```

Pass/Fail Reports

For a **BUMP2BUMP** rule, an example report file for *esd_fail.rpt* is shown below (the format for *esd_pass.rpt* and *esd_fail.rpt* files is the same). Only one arc/loop is listed for each passed instance.

```
*****
# ESD Check Results for Rule <B2B_RULE> (BUMP2BUMP)

BEGIN_ESD_RULE
  NAME B2B_RULE
  TYPE BUMP2BUMP
  LOOP_R 3.4
  PARALLEL_R 2.5
END_ESD_RULE

# BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) <-> (<BUMP2> <X> <Y> <LAYER>
<NET>)
# PASS/FAIL: <LOOP_R> <ARC1_R> <ARC2_R> <CLAMP> <LOCID1> <LOCID2>
# PARALLEL R: <VALUE>

BUMP PAIR: (VDD_1 700 1600 M4 VDD) --> (VSS_2 1300 400 M4 VSS)
PASS: 3.18274 2.20349 0.969255 PVSS_D1_I17 VDD VSS
PASS: 3.39066 1.49413 1.88652 PVSS_D1_I16 VDD VSS
FAIL: 3.43285 2.12204 1.21081 PVDD_D1_I15 VDD VSS
PASS: 3.30182 1.2371 1.96472 PVDD_D1_I14 VDD VSS
PARALLEL R: 2.7047
```

The report for B2B shows resistance from one power pad to another power pad. The first line shows the I-th pad pair, pad names and pad locations. Then each ESD loop (single ESD path) is reported, including the checking results (pass or fail), resistance, location of the ESD clamp cell on this ESD path, and its instance name. After all ESD loops are reported, the total parallel effective resistance is calculated from the multiple ESD paths, and the result is compared against the parallel R limit.

The format in the example above is for a normal run. Using the '-detail' option, the output report format syntax is as follows:

```
BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) <-> (<BUMP2>
  <X> <Y> <LAYER> <NET>)
PASS/FAIL: <LOOP_R> <ARC1_R> <ARC2_R> <CLAMP> (<cell>)
  (<X> <Y> <LAYER> <NET>) (<X> <Y> <LAYER> <NET>)
PARALLEL R: <VALUE>
```

Reporting high R due to high clamp resistance limited

B2B resistance checks do not report loop/parallel R that has a very high value due to high clamp resistance, as in the case when the clamp is characterized with the clamp I-V. An example clamp I-V file is shown below:

```
BEGIN_CLAMP_IV
  NAME Diode
  RON 0.1 1e6
  ROFF 1e6
  VT1 0.5
END_CLAMP_IV
```

In the *esd_fail.rpt* file for B2B checking, it reports:

```
BUMP PAIR: (VSSC_Def 1600 400 M4 VSSC) --> (VDD_Def 700 1600 M4 VDD)
FAIL: 1e+06 2.31593 2.14624 [PVDD_D1_I15 ADS_VSS1 VDD 1e+06]
PARALLEL R: 1e+06 MIN_LOOP
```

In above example, PF-S B2B treats clamps with $R \geq 1.0e+6$ as “off”; and not involved in the loop/parallel R computation, and hence they are not reported. You can use the ESD rule keyword, CLAMP_ROFF, to change this limit if desired, as in the following example:

```
CLAMP_ROFF 1.0e7
```

A sample *esd_fail.rpt* for **BUMP2BUMP for multiple stages** is shown following :

```
*****
# ESD Check Results for Rule <B2B_RULE> (BUMP2BUMP)

BEGIN_ESD_RULE
  NAME B2B_RULE
  TYPE BUMP2BUMP
  LOOP_R 3.5
  PARALLEL_R 2.5
  ESD_STAGE 2
END_ESD_RULE

# BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) <-> (<BUMP2> <X> <Y> <LAYER>
<NET>)
# PASS/FAIL: <LOOP_R> <ARC1_R> <ARC2_R> <CLAMP> <LOCID1> <LOCID2> <INT_NET>
<CLAMP> <LOCID3> <LOCID4>
# PARALLEL R: <VALUE>

BUMP PAIR: (VDD_1 700 1600 M4 VDD) --> (VSSIO_9 1600 1600 M4 VSSIO)
FAIL: 5.00087 2.20349 1.75207 PVSS_D1_I17      VDD      VSS      VSS
PVSS_D1_I16      VSS      ADS_VSS1
PASS: 3.35621 1.49413 1.75207 PVSS_D1_I16      VDD      VSS      VSS
PVSS_D1_I16      VSS      ADS_VSS1
FAIL: 5.08782 2.12204 1.75207 PVDD_D1_I15      VDD      VSS      VSS
PVSS_D1_I16      VSS      ADS_VSS1
PASS: 3.4518 1.2371 1.75207 PVDD_D1_I14      VDD      VSS      VSS
PVSS_D1_I16      VSS      ADS_VSS1
PASS: 3.36413 1.2371 2.02703 PVDD_D1_I14      VDD      ADS_VSS1
PARALLEL R: 3.13897
```

In the report the resistance from one power pad to another power pad is reported in this report file. The first line shows the l-th pad pair, pad names and pad locations. Then each ESD loop (single ESD path) is reported, including the checking results (pass or fail), resistance, location of the ESD clamp cell on this ESD path, and its instance name. After all ESD loops are reported, the total parallel effective resistance is calculated from the multiple ESD paths, and the result is compared against the parallel R limit.

The format in the example above is for a normal run. With the '-detail' option the report syntax is:

```
BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) <->
(<BUMP2> <X> <Y> <LAYER> <NET>)
```

```
PASS/FAIL: <LOOP_R> <ARC1_R> <ARC2_R> <CLAMP> (<cell>)
          (<X> <Y> <LAYER> <NET>) (<X> <Y> <LAYER> <NET>)
          <INT_NET> <CLAMP> (<cell>) (<X> <Y> <LAYER> <NET>)
          (<X> <Y> <LAYER> <NET>)

PARALLEL R: <VALUE>
```

A sample *esd_fail.rpt* for **BUMP2CLAMP** is shown following

```
*****
# ESD Check Results for Rule <B2C_RULE> (BUMP2CLAMP)

BEGIN_ESD_RULE
  NAME B2C_RULE
  TYPE BUMP2CLAMP
  ARC_R 2
END_ESD_RULE

# BUMP: <X> <Y> <LAYER> <NET> <BUMP_NAME>
# <R> <X> <Y> <LAYER> <NET> <LocID> <CLAMP_INST> <TO_NET>
BUMP: 1600    400      M4    VSSC    VSSC_15
2.36746 1025    20      M3    VSSC    ADS_VSS1      PVDD_D1_I15      VDD
2.0798   1325    20      M3    VSSC    ADS_VSS1      PVSS_D1_I17      VSS
```

Summary for rule <B2C_RULE> (BUMP2CLAMP):

Failed bump-to-clamp resistances: 6 (30.00% of 20)

For each power pad, its location, layer name and net name are reported. Then the resistance from this pad to each ESD cell which is connected to this net are reported, including the resistance, port location on the ESD clamp cell, and cell name.

A sample *esd_fail.rpt* for **CLAMP2BUMP** is shown following

```
*****
# CLAMP: <X> <Y> <LAYER> <NET> <LocID> <CLAMP_INST> <FROM_NET>
# <R> <X> <Y> <LAYER> <NET> <BUMP_NAME>
CLAMP: 1325    180      M3    VDD     VDD    PVSS_I17    VSS
2.31719 700     1600      M4    VDD     VDD_1
1.51344 1000    400      M4    VDD     VDD_0
```

A sample *esd_fail.rpt* for **CLAMP2CLAMP** rule is shown following:

```
*****
# ESD Check Results for Rule <C2C_RULE> (CLAMP2CLAMP)

BEGIN_ESD_RULE
  NAME C2C_RULE
  TYPE CLAMP2CLAMP
  ARC_R 1
END_ESD_RULE

# CLAMP POINT: <X> <Y> <LAYER> <NET> <LocID> <CLAMP_INST> <FROM_NET>
```

```
# <R> <X> <Y> <LAYER> <NET> <LocID> <CLAMP_INST> <TO_NET>
CLAMP POINT: 575 1930 M3 VDDIO ADS_VDD2 PIO_I33 DQ<0>
1.49508 1930 1475 M3 VDDIO ADS_VDD2 PIO_I32 DQ<4>
0.862399 1475 1930 M3 VDDIO ADS_VDD2 PIO_I36 DQ<3>
```

```
Summary for rule <C2C_RULE> (CLAMP2CLAMP):
Failed clamp-to-clamp resistances: 4 (28.57% of 14)
```

For each ESD clamp cell, its port location, layer name, port name, and cell name are reported. Then the resistance from this ESD clamp cell to other ESD clamp cells is reported.

The format for output reports in *esd_fail.rpt* for **C2I** rules is shown following:

```
*****
# <loop_r> <arc_r> <x,y location> <layer> <domain> <pin> <arc_r>
<x,y location> <layer> <domain> <pin> <inst> <cell> <clamp> <dist>
<locid1> <locid2>
```

The format for output reports in *esd_fail.rpt* for **C2M** rules is shown following:

```
*****
# inst <name> <cell> <x1 y1 x2 y2> <num_arcs> <num_passed_arcs>
<num_failed_arcs>
# <arc_r> <x,y location> <layer> <domain> <pin> <clamp>
<x,y location> <layer>
```

ESD reports the cell name in the output file for the instances analyzed in C2I or C2M analysis when either 'DETAILED_MODE 1' is specified in the rule file, or the '-detail' option is specified with the 'perform esdcheck' command.

ESD Info Reports

The ESD Info file reports contains additional information on clamps and circuit ESD protection, as shown in the following examples.

Bumps Not Connected to Clamps Info report

```
# ESD Check for Bumps Not Connected to Clamps for Rule <B2B_RULE> (BUMP2BUMP)
```

```
BEGIN_ESD_RULE
  NAME B2B_RULE
  TYPE BUMP2BUMP
  LOOP_R 3.5
  PARALLEL_R 2.5
  ESD_STAGE 2
END_ESD_RULE

# <BUMP> <X> <Y> <LAYER> <NET>
DVSS_11 400 1300 M4 DVSS
AVSS_7 1600 1000 M4 AVSS
AVSS_6 700 400 M4 AVSS
TX_22 1750 1150 M4 TX
S<4>_21 1750 850 M4 S<4>
```

S<3>_20 1750 550 M4 S<3>

Bumps and Domain Info Report

```
# ESD Check for Bump and Domain Info for Rule <B2B_RULE> (BUMP2BUMP)

BEGIN_ESD_RULE
    NAME B2B_RULE
    TYPE BUMP2BUMP
    LOOP_R 3.5
    PARALLEL_R 2.5
    ESD_STAGE 2
END_ESD_RULE

# DOMAIN PAIR <ID>: (<net1> <net2>) {
#   INTERNAL DOMAINS: <num>
#     <net_names>
#   BUMP PAIRS CHECKED: <num>
#   BUMP LIST FOR DOMAIN <net1> (<count>) {
#     <bump_list1> (<name> <xy_location> <layer> <net>)
#   }
#   BUMP LIST FOR DOMAIN <net2> (<count>) {
#     <bump_list2> (<name> <xy_location> <layer> <net>)
#   }
# }

DOMAIN PAIR 1: (VSSIO VSSC) {
    INTERNAL DOMAINS: 2
    VSS
    VDD
    BUMP PAIRS CHECKED: 2
    BUMP LIST FOR DOMAIN VSSIO (1) {
        VSSIO_9      1600      1600      M4      VSSIO
    }
    BUMP LIST FOR DOMAIN VSSC (1) {
        VSSC_15      1600      400       M4      VSSC
    }
}
```

ESD Summary Reports

A sample *esd_summary.rpt* for **BUMP2BUMP** rule is shown following

```
*****
# ESD Check Summary for Rule <B2B_RULE> (BUMP2BUMP)

BEGIN_ESD_RULE
    NAME B2B_RULE
    TYPE BUMP2BUMP
    LOOP_R 3.4
    PARALLEL_R 2.5
END_ESD_RULE
```

```

DOMAIN PAIR 1: (VSSIO VSS) {
    Clamp connections between (VSSIO VSS): 1
    Bump-to-bump pairs checked: 4
    Bump-to-bump loop resistances:
        Pass:      2 ( 50.0% of      4)
        Fail:      2 ( 50.0% of      4)
    Bump-to-bump parallel resistances:
        Pass:      0 ( 0.0% of      4)
        Fail:      4 (100.0% of      4)
}
    
```

Displaying Resistance Checking Results in the GUI

RedHawk provides a rich set of GUI features to graphically display ESD resistance and current density rule checking results in addition to the textual results in *adsRpt/ESD* (default ESD report directory), or a directory that you specify.

After running through PF-S analysis with the ‘perform esdcheck’ command, or using the menu **Tools->PathFinder S > ESD Resistance Check**, complete the information in the resistance check dialog displayed. **Tools->PathFinder S > ESD Resistance Check** performs the same function as the TCL command ‘perform esdcheck’.

PathFinder can report a list of isolated bumps that have no clamps connected to them, based on B2B/B2C checks. The command is **View ->ESD Connectivity Lists -> Isolated Bump List**, as shown in Figure 16-8.

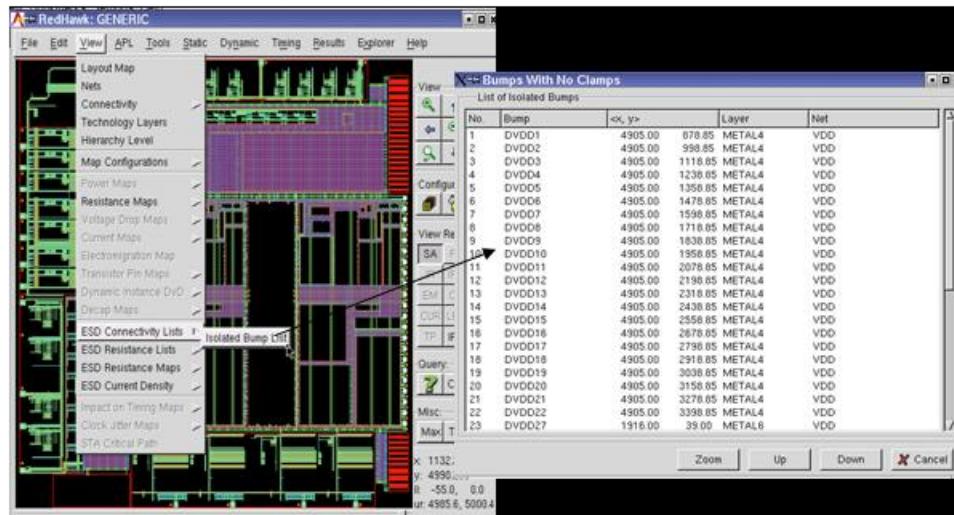


Figure 16-8 Isolated bump list

The **View->ESD Resistance Lists** command shows fail/pass reports in table form for different rule types, with the capability to show flight lines and minimum resistance paths.

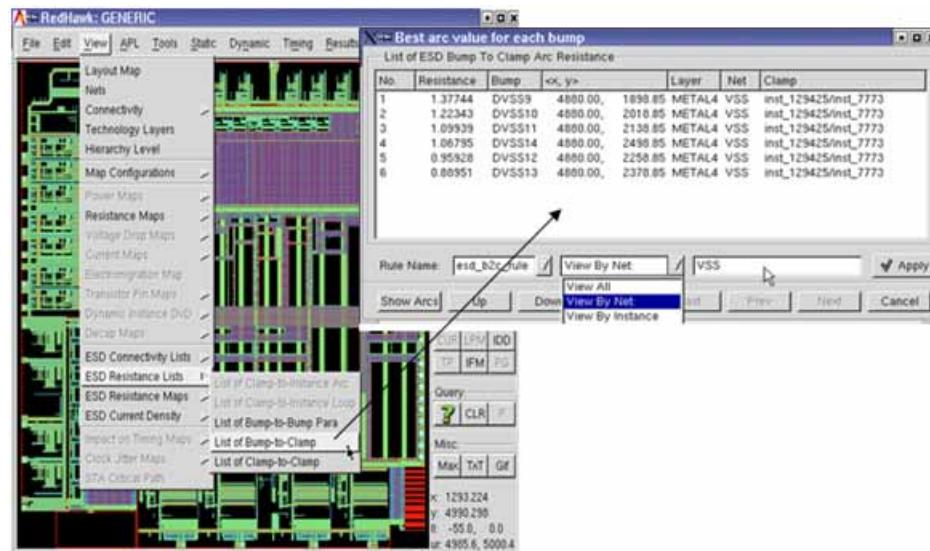


Figure 16-9 Resistance checks by net and instance name

You can get resistance checking results from B2C and C2C checks on the basis of net name or instance name by clicking on **View->ESD Resistance Lists->List of Bump-to-Clamp**, or **List of Clamp-To-Clamp** to get resistance result lists. At the bottom of the dialog is a dropdown box to allow filtering results on the basis of net name or instance name, as shown in Figure 16-9.

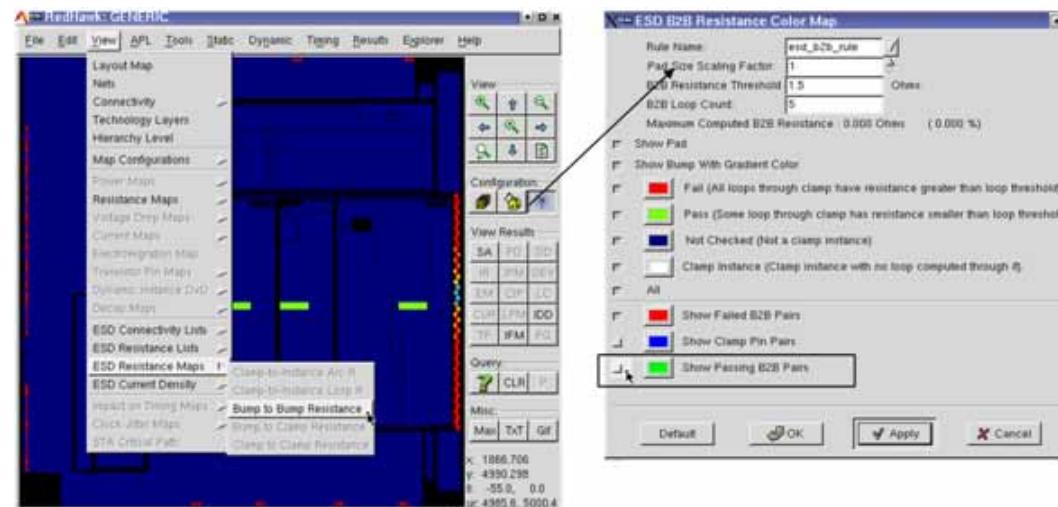


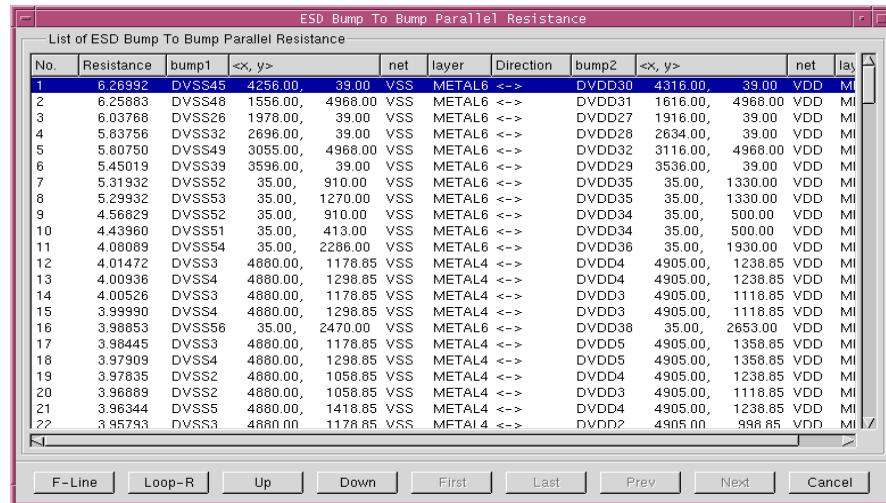
Figure 16-10 Bump-to-bump resistance display

For B2B (BUMP2BUMP) rule checking, the status of each clamp from the Bump-to-Bump check are displayed, as shown in Figure 16-10. Using the **View->ESD Resistance Maps->Bump to Bump Resistance** command, the participating clamps are shown in green (that is, some B2B paths go through this clamp passing the PARALLEL_R threshold), or in red (meaning that none of the B2B paths go through this clamp passing the PARALLEL_R threshold). To display passing pairs, you can select the menu command

ESD Resistance Maps ->B2B/B2C Resistance, and then can enable/disable **Show Passing B2B Pairs** in the dialog box, as shown in Figure 16-10.

Selecting the **View->ESD Resistance Lists->List of Bump-to-Bump Para** command displays the top failed/passed bump pairs, based on user provided PARALLEL_R constraint. By default 200 pairs are displayed, but this can be modified. Figure 16-11 shows the top-ranked (in terms of worst Parallel_R resistance, checked against the user's Parallel_R limit) Bump-to-Bump pairs. You can click on the **F-Line** or **SPT** button to get more information on a particular B2B path or minimum resistance path information, as shown in the following figures.

Clicking on the **F-Line** button when a B2B pair is highlighted in the list, all the LOOP_R paths are shown as flight lines on GUI, as shown in Figure 16-12. To customize set color for ESD resistance check "Pass / Fail" FlyLines, right click on "FlyLine" button which opens a configuration dialog window. Hover mouse above "FlyLine" button to get "info balloon" message.



No.	Resistance	bump1	<x, y>	net	layer	Direction	bump2	<x, y>	net	lay
1	6.26992	DVSS45	4256.00, 39.00	VSS	METAL6 <->		DVDD30	4316.00, 39.00	VDD	MI
2	6.25883	DVSS48	1556.00, 4968.00	VSS	METAL6 <->		DVDD31	1816.00, 4968.00	VDD	MI
3	6.03768	DVSS26	1978.00, 39.00	VSS	METAL6 <->		DVDD27	1916.00, 39.00	VDD	MI
4	5.63756	DVSS32	2696.00, 39.00	VSS	METAL6 <->		DVDD28	2634.00, 39.00	VDD	MI
5	5.80750	DVSS49	3055.00, 4968.00	VSS	METAL6 <->		DVDD32	3116.00, 4968.00	VDD	MI
6	5.45019	DVSS39	3596.00, 39.00	VSS	METAL6 <->		DVDD29	3536.00, 39.00	VDD	MI
7	5.31932	DVSS52	35.00, 910.00	VSS	METAL6 <->		DVDD35	35.00, 1330.00	VDD	MI
8	5.29932	DVSS53	35.00, 1270.00	VSS	METAL6 <->		DVDD35	35.00, 1330.00	VDD	MI
9	4.56823	DVSS52	35.00, 910.00	VSS	METAL6 <->		DVDD34	35.00, 500.00	VDD	MI
10	4.43960	DVSS51	35.00, 413.00	VSS	METAL6 <->		DVDD34	35.00, 500.00	VDD	MI
11	4.08069	DVSS54	35.00, 2286.00	VSS	METAL6 <->		DVDD36	35.00, 1930.00	VDD	MI
12	4.01472	DVSS3	4880.00, 1178.85	VSS	METAL4 <->		DVDD4	4905.00, 1238.85	VDD	MI
13	4.00936	DVSS4	4880.00, 1298.85	VSS	METAL4 <->		DVDD4	4905.00, 1238.85	VDD	MI
14	4.00526	DVSS3	4880.00, 1178.85	VSS	METAL4 <->		DVDD3	4905.00, 1118.85	VDD	MI
15	3.99990	DVSS4	4880.00, 1298.85	VSS	METAL4 <->		DVDD3	4905.00, 1118.85	VDD	MI
16	3.98853	DVSS56	35.00, 2470.00	VSS	METAL6 <->		DVDD38	35.00, 2653.00	VDD	MI
17	3.98445	DVSS3	4880.00, 1178.85	VSS	METAL4 <->		DVDD5	4905.00, 1358.85	VDD	MI
18	3.97909	DVSS4	4880.00, 1298.85	VSS	METAL4 <->		DVDD5	4905.00, 1358.85	VDD	MI
19	3.97635	DVSS2	4880.00, 1058.85	VSS	METAL4 <->		DVDD4	4905.00, 1238.85	VDD	MI
20	3.96689	DVSS2	4880.00, 1058.85	VSS	METAL4 <->		DVDD3	4905.00, 1118.85	VDD	MI
21	3.96344	DVSS5	4880.00, 1418.85	VSS	METAL4 <->		DVDD4	4905.00, 1238.85	VDD	MI
22	3.95793	DVSS3	4880.00, 1178.85	VSS	MFTA1 4 <->		DVDD2	4905.00, 998.85	VDD	MI

Figure 16-11 Worst parallel resistance list

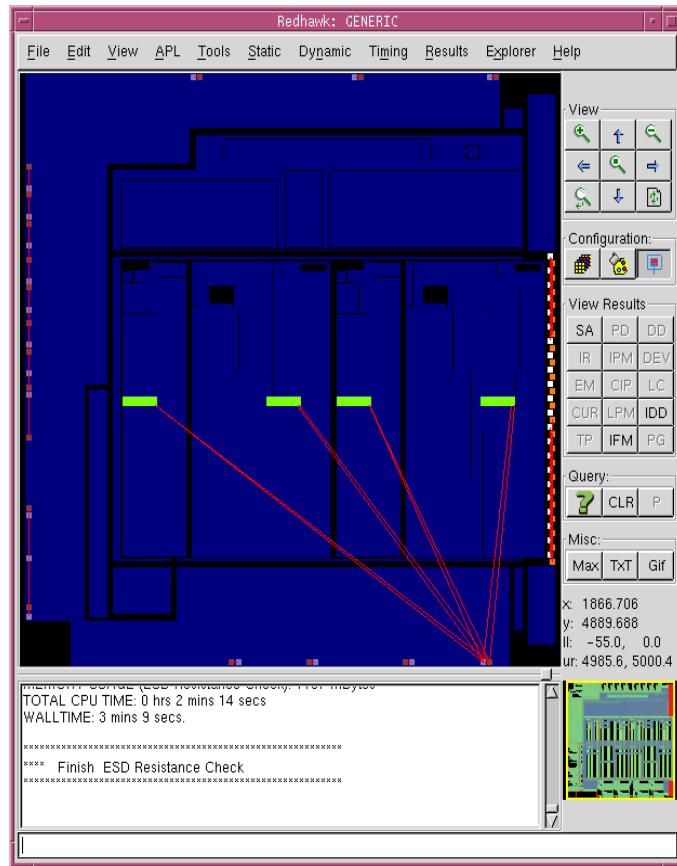
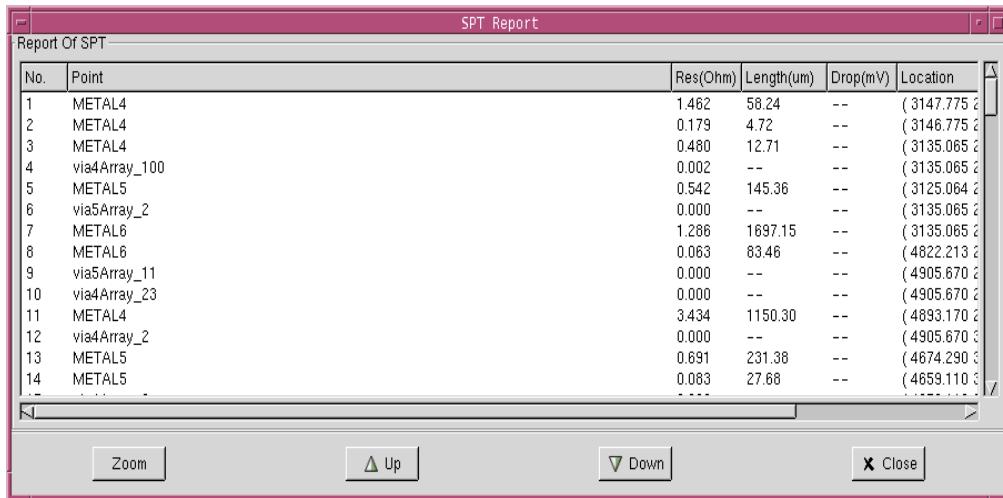


Figure 16-12 F-line loop R path display

All loops for bump pair <DVSS45> - <DVDD30>							
BUMP PAIR: (DVSS45 4256.000000 39.000000 VSS METAL6) <-> (DVDD30 4316.000000 39.000000 VDD METAL6)							
List of ESD Bump To Bump Loop Resistance							
No.	Pass/Fail	Total R	R 1	R 2	LocId1	LocId2	Stages
1	Fail	6.26992	2.89631	3.37361	VSS	VDD	1
2	Fail	6.63974	3.11447	3.52527	VSS	VDD	1
3	Fail	6.64016	3.12401	3.51615	VSS	VDD	1
4	Fail	6.69397	3.11037	3.58360	VSS	VDD	1

Figure 16-13 List of all failed/passed loop_R paths

After clicking on **Loop_R** button in the Worst list dialog, with a bump pair selected, all the failed/passed loop_R paths are listed, as shown in Figure 16-13.



The screenshot shows a software window titled "SPT Report" with a sub-header "Report Of SPT". The main content is a table with the following data:

No.	Point	Res(Ohm)	Length(um)	Drop(mV)	Location
1	METAL4	1.462	58.24	--	(3147.775 2
2	METAL4	0.179	4.72	--	(3146.775 2
3	METAL4	0.480	12.71	--	(3135.065 2
4	via4Array_100	0.002	--	--	(3135.065 2
5	METAL5	0.542	145.36	--	(3125.064 2
6	via5Array_2	0.000	--	--	(3135.065 2
7	METAL6	1.286	1697.15	--	(3135.065 2
8	METAL6	0.063	83.46	--	(4822.213 2
9	via5Array_11	0.000	--	--	(4905.670 2
10	via4Array_23	0.000	--	--	(4905.670 2
11	METAL4	3.434	1150.30	--	(4893.170 2
12	via4Array_2	0.000	--	--	(4905.670 3
13	METAL5	0.691	231.38	--	(4674.290 3
14	METAL5	0.063	27.68	--	(4659.110 3

At the bottom of the window are four buttons: "Zoom", "Up", "Down", and "Close".

Figure 16-14 Minimum resistance path

After clicking on the **SPT** button in the Worst list, with a loop_R path selected, the minimum resistance path is displayed, as shown in Figure 16-14.

The **View->ESD Maps->Clamp-to-Instance Loop R** command shows all flight lines of the clamp to instance resistance check. The flight lines in red indicate that the C2I loop resistance exceeds the user-set LOOP_R threshold in the C2I rule, as shown in Figure 16-15.

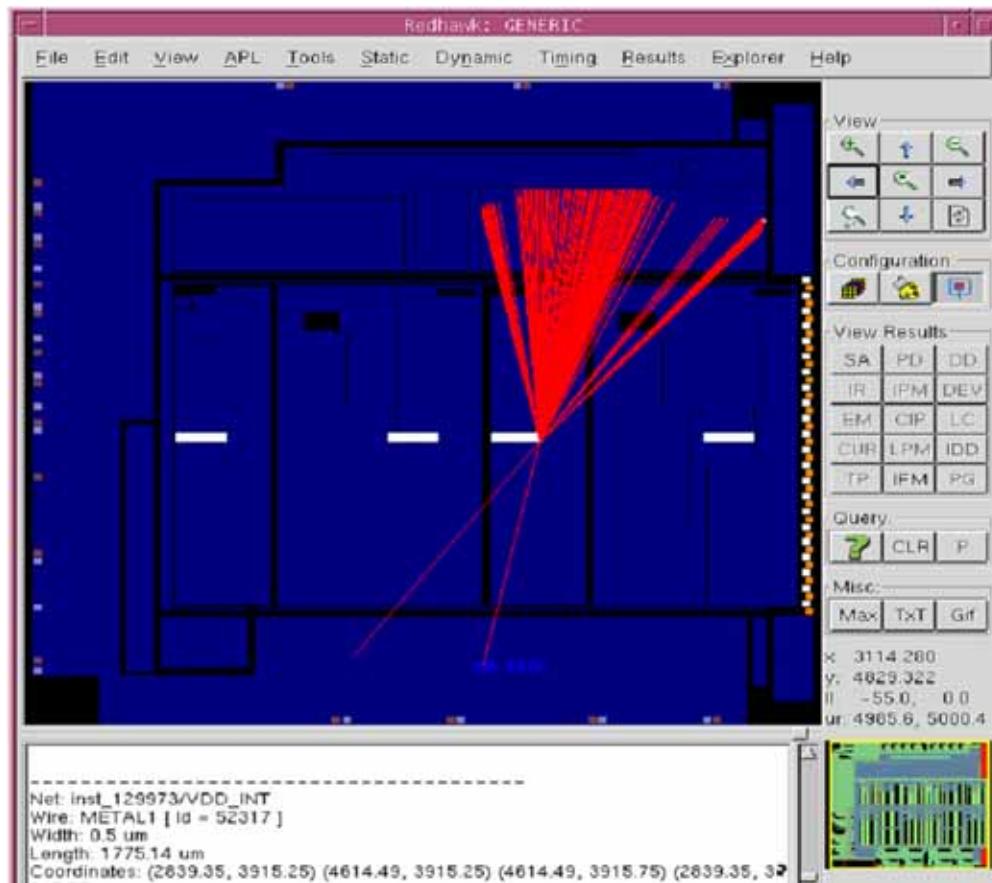


Figure 16-15 Display of loop R paths exceeding resistance limit

List of ESD C2I Loop Resistance							
No.	Loop R	Arc1 R	Arc2 R	Instance	Clamp	LocID1	LocID2
1	80.44183	69.99240	10.44942	inst_129747/adSU1	inst_129424/inst_92357	VDD	N.A.
2	11.84655	8.92506	2.92150	inst_129973/inst_354556	inst_129424/inst_92357	VDD	N.A.
3	11.82340	8.91485	2.90855	inst_129973/inst_354557	inst_129424/inst_92357	VDD	N.A.
4	11.83227	8.73881	2.89346	inst_129973/inst_355156	inst_129424/inst_92357	VDD	N.A.
5	11.53160	8.96717	2.56442	inst_129973/inst_354346	inst_129424/inst_92357	VDD	N.A.
6	11.51534	8.71987	2.79547	inst_129973/inst_344768	inst_129424/inst_92357	VDD	N.A.
7	11.43911	8.48106	2.95505	inst_129973/inst_355155	inst_129424/inst_92357	VDD	N.A.
8	11.43652	8.52797	2.90855	inst_129973/inst_343061	inst_129424/inst_92357	VDD	N.A.
9	11.42484	8.96717	2.45767	inst_129973/inst_353950	inst_129424/inst_92357	VDD	N.A.
10	11.37642	8.61748	2.75893	inst_129973/inst_342993	inst_129424/inst_92357	VDD	N.A.
11	11.30224	8.60205	2.70019	inst_129973/inst_425477	inst_129424/inst_92357	VDD	N.A.
12	11.29971	8.54077	2.75893	inst_129973/inst_343012	inst_129424/inst_92357	VDD	N.A.
13	11.25512	8.91485	2.34027	inst_129973/inst_353954	inst_129424/inst_92357	VDD	N.A.
14	11.12898	8.51541	2.61355	inst_129973/inst_425478	inst_129424/inst_92357	VDD	N.A.
15	11.09090	8.51175	2.58735	inst_129973/inst_413297	inst_129424/inst_92357	VDD	N.A.
16	11.07467	8.62363	2.45105	inst_129973/inst_354357	inst_129424/inst_92357	VDD	N.A.
17	11.06666	8.52797	2.53869	inst_129973/inst_354757	inst_129424/inst_92357	VDD	N.A.
18	11.02409	8.41356	2.61053	inst_129973/inst_354991	inst_129424/inst_92357	VDD	N.A.
19	10.97935	8.37964	2.59972	inst_129973/inst_343241	inst_129424/inst_92357	VDD	N.A.
20	10.95568	8.42877	2.52691	inst_129973/inst_425475	inst_129424/inst_92357	VDD	N.A.
21	10.92582	8.42511	2.50071	inst_129973/inst_413298	inst_129424/inst_92357	VDD	N.A.
22	10.81761	8.22730	2.59031	inst_129973/inst_344556	inst_129424/inst_92357	VDD	N.A.

Figure 16-16 List of worst clamp-to-instance paths

Use the **View->ESD Resistance Lists->List of Clamp to Instance Loop** command to display the worst clamp to instance loop_R paths, as shown in Figure 16-16. Click on the **F-Line** or **SPT** buttons to get more information on the particular loop path or minimum resistance path, as shown in the following figures.

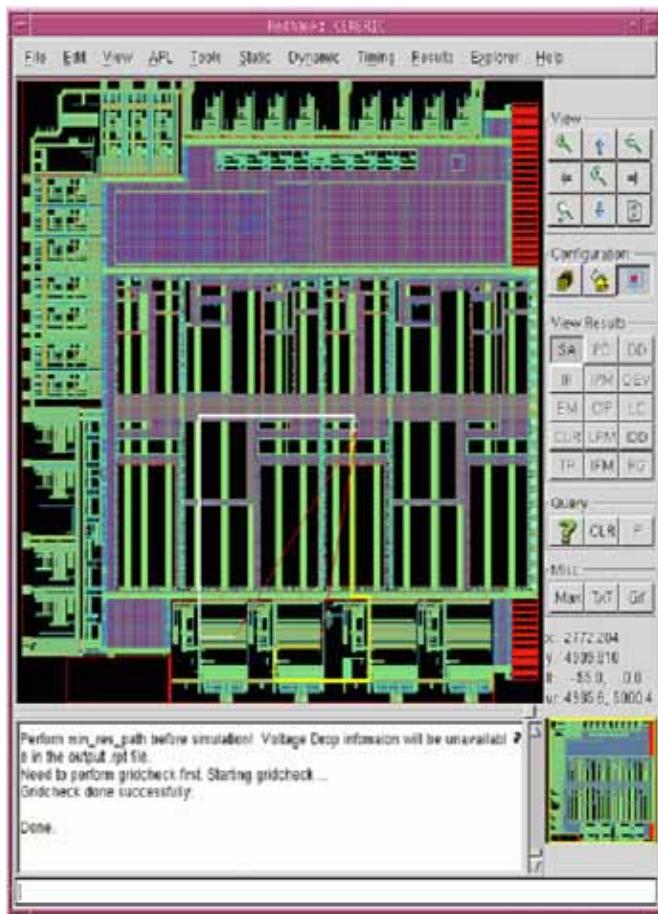


Figure 16-17 Display of selected clamp-to-instance paths

From the list of worst clamp to instance paths (Figure 16-16), a selected failure path can be chosen to display the minimum resistance path when the **SPT** button is clicked, as shown in Figure 16-17 and Figure 16-18.

SPT Report					
No.	Point	Res(Ohm)	Length(um)	Drop(mV)	Location
1	METAL3	24.296	316.68	--	(1708.535 532.475) (2025)
2	METAL3	0.117	1.22	--	(1732.990 533.475) (1733)
3	METAL3	0.005	0.40	--	(1733.390 534.695) (1733)
4	METAL3	0.364	29.16	--	(1704.625 534.695) (1733)
5	VIA34_190_190_compress_cell_50_compress_1000_1000_10_10_0_0_100	0.004	--	--	(1729.390 537.645)
6	METAL4	0.371	29.71	--	(1703.730 534.695) (1733)
7	METAL4	0.356	26.41	--	(1677.315 535.125) (1703)
8	VIA45_190_190_compress_cell_50_compress_1000_1000_16_10_0_0_160	0.002	--	--	(1682.345 537.905)
9	METAL5	0.044	5.87	--	(1677.315 532.040) (1687)
10	VIA56_360_360_compress_cell_50_compress_2000_0_5_1_0_0_5	0.012	--	--	(1684.815 532.040)
11	VIA56_360_360_compress_cell_50_compress_2000_0_5_1_0_0_5	0.012	--	--	(1684.815 532.040)
12	METAL6	0.181	107.47	--	(1677.815 532.040) (1686)
13	VIA56_360_360_compress_cell_50_compress_2000_0_5_1_0_0_5	0.012	--	--	(1684.780 636.685)
14	METAL5	0.200	26.79	--	(1677.315 636.680) (1687)

Figure 16-18 Display of selected minimum resistance C2I paths

A histogram of ESD rule checking results can be displayed using the menu command **Results->Analysis Histogram**. All rules can be displayed in a histogram, including the number of paths included in a rule check (such as a C2I clamp-to-instance count) as it relates to the specified resistance constraint. The setup dialog is shown in Figure 16-19, and a sample histogram is displayed in Figure 16-20.

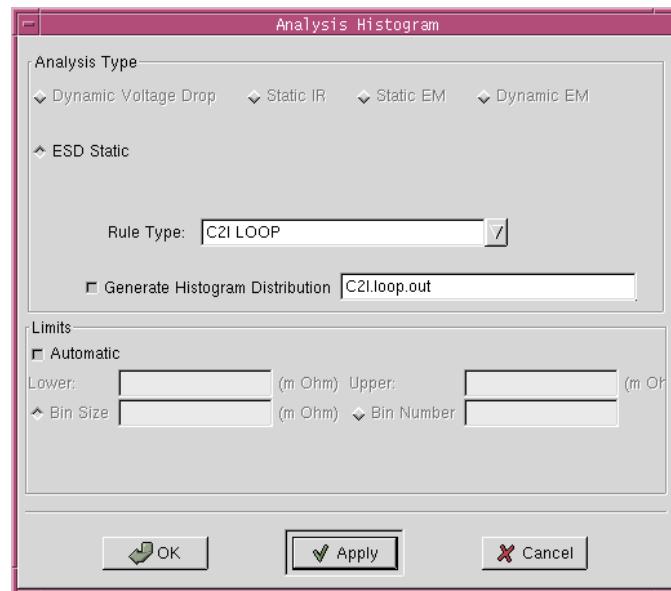


Figure 16-19 Histogram dialog

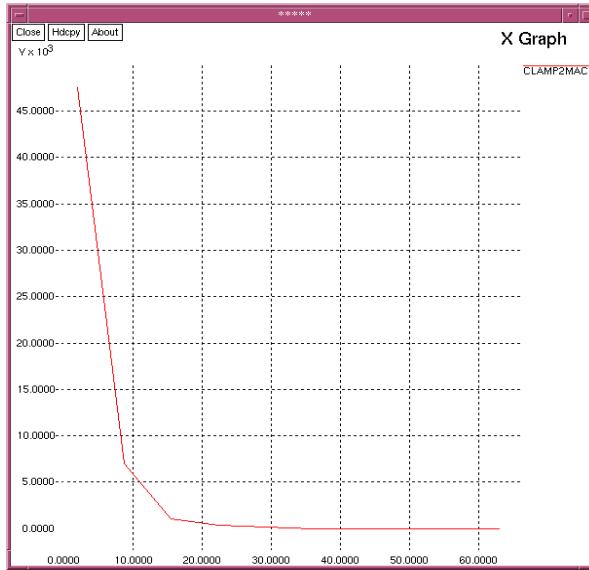


Figure 16-20 Histogram of ESD checking results

Special Resistance Calculations for ESD Analysis

The ‘perform res_calc’ command can be used for a number of special ESD analyses, and has some options that require a PF license to execute. These options are described in this section.

```
perform res_calc
? [-instance <name_list>]? ?[-instFile <filename>]?
? [-cell <name_list>]? ?-cellFile <cell_filename>?
? -from {<x1 y1> <layerName1> ?<netName>? }?
? -to {<x2 y2> <layerName2> ?<netName>?} ?
? -fromFile <file1> -toFile <file2> [-pairwise|-straightPair]?
? -fullchip? ?-constrFile <res_Constr_file> ?
```

where

- instance : displays a report of the worst resistance location for the specified instance(s). The report contains worst case resistance for one location per domain per instance.
- instFile : same as the -instance option, except that you can specify a file containing a list of instances. The report contains worst case resistance for one location per domain per instance.
- cell : creates a resistance report of the worst resistance locations of all instance(s) for the specified cell masters. The report has one location per domain for each instance. <name_list> can be represented in two formats: {<name1> <name2> ... }, or <name1>,<name2>, ... (no spaces in this form)
- cellFile : specifies a text file that contains the cell names, one per line, for res_calc to compute the equivalent grid resistances.
- from/-to : performs resistance calculation from one node at or near <x1 y1> on layer <layerName1> to another node at or near <x2 y2> on layer <layerName2>. Specifying the associated net names is optional.
- fromFile/-toFile: performs resistance calculation between points listed in two files in one of two ways: “-pairwise” calculates p2p resistances for all pairs of

points in the two files, while the “-straightPair” option only calculates and reports p2p resistances for “straight pairs”— that is, between the 1st point in the -fromFile and the 1st point in the -toFile, and between the 2nd points in each file, and so on. One requirement of the -straightPair’ option is that the -fromFile and -toFile must have the same number of points. Also, the content of the specified files has the following format:

```
<x> <y> <layer> <net>
```

```
...
```

where <layer> and <net> are optional, but highly recommended.

-fullchip: specifies that effective resistances for all nodes in the design, not just the pin nodes in the instances, and including those for wires/vias, are to be computed.

-constrFile <res_Constr_file> : specifies a file defining bump-to-instance pin resistance checks to be performed

The constraint file format is:

```
IP_RESISTANCE
    CHECKBUMP [ RES_TOP|RES_BOT|DIS_TOP|DIS_BOT | SHORT |
        ALL | RES_MAX | RES_MIN | DIS_MAX | DIS_MIN ]
        <num_bumps>
    CELL <cell_name>
        PIN <Pin_name> <Res_threshold> ?<x> <y> ?<layer>? ?
        ...
    END CELL
    INSTANCE <inst_name>
        PIN <Pin_name> <Res_threshold> ?<x> <y> ?<layer>? ?
    END INSTANCE
END IP_RESISTANCE
```

where

CHECKBUMP specifies the type of check and the number of bumps to be checked, which applies to all pins in the constraint file, and:

RES_TOP - checks nodes with the highest resistance, up to <num_bumps>

RES_BOT - checks nodes with the lowest resistance, up to <num_bumps>

DIS_TOP - checks nodes at longest distance, up to <num_bumps>

DIS_BOT - checks nodes at shortest distance, up to <num_bumps>

SHORT: checks and reports the effective resistance from IP pins to all associated bumps shorted for calculation purposes.

ALL: checks and reports point-to-point effective resistance from each pin to all associated bumps.

RES_MAX : checks and reports the maximum point-to-point effective resistance among all pin-bump pairs.

RES_MIN : checks and reports the minimum point-to-point effective resistance among all pin-bump pairs.

DIS_MAX : checks and reports the maximum geometric distance among all pin-bump pairs.

DIS_MIN : checks and reports the minimum geometric distance among all pin-bump pairs.

The default output file is *adsRpt/<design_name>.res_calc*.

A sample output file format follows:

```
# Grid Resistance Calculation Report
# CELL CONSTRAINT: <cell> <Pin> <Res_Constr> ?<x> <y> ?<layer>? ?
# INSTANCE CONSTRAINT: <cell> <Pin> <Res_Constr> ?<x> <y> ?<layer>? ?
# PIN: <pin>: (<x> <y> <layer> <net>) <instance> <cell>
# CELL CONSTRAINT: cell_55 VSS 3 100 200 METAL4
# Check against the resistive furthest 5 bumps
{
# PIN: VSS (2346.05 1935.28 METAL4 VSS) inst_129422/inst_7768 cell_55
# <Res> PAD:(<x> <y> <layer> <net>)
# FAIL:
    3.19957 DVSS45 (4256 39 METAL6 VSS)
# PASS:
    2.9993 DVSS26 (1978 39 METAL6 VSS)
    2.95331 DVSS32 (2696 39 METAL6 VSS)
    2.829 DVSS39 (3596 39 METAL6 VSS)
    2.71838 DVSS52 (35 910 METAL6 VSS)
```

Current Density Checking

RedHawk PathFinder provides current density analysis for user-specified B2B, P2P, and P2PM ESD paths, and reports the IR voltage and EM values in report files, as well as displaying results in GUI maps, to help users perform DC analysis of ESD paths and analyze possible IR voltage and EM violations. Current density analysis is implemented in two modes:

Mode 1 - performs IR voltage analysis considering all clamp paths with iterative convergence to reach a solution that includes a set of On clamps and Off clamps for the specified bump pairs.

Mode 2 - specifies the clamp paths and bump pairs on which the analysis should be performed.

The ESD command is ‘perform esdcheck’. For details, see [section "perform esdcheck", page 16-448](#).

Mode 1 Rules Files - all clamp paths

The format of the rules file for Mode 1 current density analysis is as follows:

```
BEGIN_ESD_RULE
NAME <name>
TYPE CURRENT_DENSITY # "DC" or "CD" for short
LOOP_R <resistance_Ohms>
ESD_STAGE 2
B2B_RULE_NAME <b2b_rule>
BUMP_PAIR/PIN_PAIR <bump/pin Name1> <bump/pin Name2>
BUMP_PAIR_FILE <filename>
PIN_PAIR_VTH <V_diff>
SHOTGUN_MODE [ 0 | 1 ]
CDM_ALL_NET [ 0 | 1 ]
NET_PAIR_DOMAIN [ SAME | DIFF ]
FROM_NET <net_name1> <net_name2> ...
TO_NET <net_name1> <net_name2> ...
TERMINAL_NET <net_name1> <net_name2> ...
```

```

FROM_NET_GROUP ?Power? ?Ground? ?Signal?
TO_NET_GROUP ?Power? ?Ground? ?Signal?
TERMINAL_NET_GROUP ?Power? ?Ground? ?Signal?
PACKAGE_PIN [ 0 | 1 ]
SHORT_BUMP_IN_NET <net_1 <net_2> ...
SHORT_BUMP_IN_NET_GROUP [ POWER|GROUND|SIGNAL ] ?All?
NO_SHORT_BUMP_IN_NET <net_1 <net_2> ...
JEDEC_MODE [BIDIR |POS |NEG ]? OPEN ?
B2B_LOOP_LENGTH <clamp_count1> <clamp_count2> ...
PAD_FILE <ploc/pcell_filename>
EXCLUDE_PAD_FILE <ploc/pcell_filename>
ZAP_B2B_NET <net1> <net2> ...
ZAP_B2B_NET_GROUP {POWER | GROUND | SIGNAL }
ZAP_VOLTAGE <voltage_Volts>
ZAP_R <resistance_Ohms>
ZAP_CURRENT <current_in_Amps>
ZAP_FROM <name> <x> <y> ?<layer>? ?<net>?
ZAP_TO <name> <x> <y> ?<layer>? ?<net>?
EXTEND_CLAMP_CONN [ 0 | 1 ]
IGNORE_EM_IN_CLAMP [ 0 | 1 ]
EM_SCALE <scale_factor>
EM_RULE_SET <rule_set_name>
NET_PAIR_VTH <VTH value> <net_name1> ...
PEAK_VOLT <peak_V>
DIFF_VOLT <differential_V>
CLAMP_IMAX <I1> [<I2>]
CLAMP_VMAX <V1> [<V2>]
SHORT_CLAMP_NODE [ 0 | 1 ]
CACHE_EM <value>
EM_THRESHOLD <value>
REFF_THRESHOLD <value>
I_THRESHOLD <value>
V_THRESHOLD <value>
SAVE_CLAMP_FAIL [0|1]
SAVE_EM [0|1]
EXCEL 1 (or EXCEL_REPORT 1)
END_ESD_RULE

```

where

NAME : rule name

TYPE CURRENT_DENSITY: performs current density analysis

LOOP_R: loop resistance threshold for a single ESD path

ESD_STAGE <min> <max> : for multistage rules, PathFinder checks all net pairs that are connected by at least <min> clamps and no more than <max> clamps.

Note that if you specify only the ESD_STAGE rule, PathFinder does not compute B2B resistance, to reduce runtime. If either LOOP_R or PARALLEL_R keywords are specified, then B2B resistance is computed.

B2B_RULE_NAME: uses previously-executed B2B Resistance check results in ESD current density analysis to improve runtime when non-linear clamp devices are used.

BUMP_PAIR: specifies the bump pair to be tested. The first bump in the pair is connected to the positive terminal of the zapping source. Multiple BUMP_PAIR line entries can be used in the same rule file.

PIN_PAIR: specifies the pin pairs to be tested.

BUMP_PAIR_FILE: specifies a file containing a list of bump pairs to be analyzed.

PIN_PAIR_VTH : reports pin pair differential voltages in ESD-CD checks. If the CD check involves more than one net, the pin pair differential voltages that are above the positive voltage threshold <V_diff> are reported in the file *esd_pinPair.rpt*.

SHOTGUN_MODE: when turned on, analyzes multiple nets together to reduce the run time for arc-based analyses, such as when two terminals of the zapping source are connected in the same net and CD checks are needed from all I/O signal bumps to the protecting clamp devices.

CDM_ALL_NET : applies static charge to all domains connected to the discharge point by clamp devices. By default, the static charge is applied to the domain where the discharge point is.

NET_PAIR_DOMAIN : checks and reports violating pairs connected to the same power/ground domain (SAME) or different power/ground domains (DIFF).

FROM_NET/TO_NET: current density checks are made FROM all bumps on the specified net(s) TO all bumps are the specified net(s).

TERMINAL_NET: current density checks are made from/to all bumps on the specified net(s)

FROM_NET_GROUP: current density checks are made from all bumps belonging to one or more of the specified net groups Power, Ground, and Signal.

TO_NET_GROUP: current density checks are made to all bumps belonging to one or more of the specified net groups Power, Ground, and Signal.

TERMINAL_NET_GROUP: current density checks are made from/to all bumps belonging to one or more of the specified net groups Power, Ground, and Signal.

PACKAGE_PIN : specifies pin-to-pin CD checks with net-group options, such as FROM_NET/TO_NET/TERMINAL_NET between package pins. This option is different from the option 'PACKAGE 1', in which case PathFinder still uses bumps as reference, but considers package level connections.

SHORT_BUMP_IN_NET: specifies the net in which all bumps are to be shorted, in order to consider the effects of the package, since the bumps are connected through a very low resistance path through package.

SHORT_BUMP_IN_NET_GROUP: shorts bumps together in each domain in the specified group. If 'All' is specified, shorts bumps in all domains in the specified group together.

NO_SHORT_BUMP_IN_NET : allows excluding specified nets otherwise shorted as a group using SHORT_BUMP_IN_NET_GROUP.

JEDEC_MODE : (default Off) specifies zapping from bumps in each individual net in a defined net group to the rest of the nets in the group, where the zap direction is chosen. Unless OPEN is selected, the bumps in the "to" group are shorted by default.

B2B_LOOP_LENGTH : <clamp_countN> values specify the number of clamps in each B2B path (positive integers less than or equal to MAX_ESD_STAGE). Values of <clamp_countN> *not included* mean that B2B paths with that number of clamps are excluded from parallel R computation.

PAD_FILE : specifies ploc/pcell file listing bumps to be included in ESD checks

EXCLUDE_PAD_FILE : excludes consideration of checks in specified pad file

ZAP_B2B_NET : provides zapping between bump pairs belonging one or more specified nets.

ZAP_B2B_NET_GROUP : provides zapping to all bump pairs of one type of net--POWER, GROUND, or SIGNAL. Depending on the number of bumps, CD checks are performed on all combinations.

ZAP_VOLTAGE: specifies the voltage of the zapping source.

ZAP_R: specifies the series resistance to the zapping voltage source.

ZAP_CURRENT: directly specifies the zapping current. When this is used, ZAP_VOLTAGE and ZAP_R keywords are ignored.

ZAP_FROM/ ZAP_TO : specifies the user-assigned name for the “from” and “to” points, x, y, and the optional layer and nets involved.

EXTEND_CLAMP_CONN: uses the discharge paths through clamp devices even for cases when from/to zap points are part of same net. For example, when checking resistance between two ground bumps, PF also takes into account the parallel discharge path through the power-rails that are connected to ground through clamp devices. When set, this keyword also supports clamp devices that have both terminals connected to the same net for ESD-CD and P2P R checks. Also note that when 'EXTEND_CLAMP_CONN 1' is set, B2B-related ESD rule keywords such as ESD_STAGE, LOOP_R, and PARALLEL_R have no effect. When 'EXTEND_CLAMP_CONN 1' is set with ZAP_B2B_NET/ZAP_B2B_NET_GROUP specified in ESD rule, RedHawk performs bi-directional ESD-CD/R check.

IGNORE_EM_IN_CLAMP: disable EM violation in Clamp instance in DC check, default value is 0

EM_SCALE: scales the ESD currents to match different Zap scenarios. You can provide one EM current limit in the technology file and then scale the current limit to that desired for ESD.

EM_RULE_SET : specifies a rule set file that enables importing different EM current density limit checks for different rules in same RedHawk session. Note that EM_RULE_SET values should be present as a part of the EM_TECH_FILE keyword in the GSR file.

NET_PAIR_VTH: reports driver-receiver voltage threshold pairs for specified nets that violate the voltage difference VTH. Optionally, one or more nets with the same VTH value can be specified.

PEAK_VOLT : specifies the peak voltage to be used in current density checks

DIFF_VOLT : specifies the differential voltage to be used in current density checks

CLAMP_IMAX: specifies the current thresholds for the positive and negative breakdown checks for power clamps and diodes. When only one value is specified, it is applied to all ESD_PIN_PAIRs, and the threshold is the same in both directions. You can also specify the I-V limits at clamp cell level, or for individual clamp devices, using the clamp I-V specification. The default value is 0.0, which means no check is performed.

CLAMP_VMAX: specifies the voltage thresholds for the positive and negative breakdown checks for power clamps and diodes. When only one value is specified, it is applied to all ESD_PIN_PAIRs, and the threshold is the same in both directions. You can also specify the I-V limits at clamp cell level, or for individual clamp devices, using the clamp I-V specification. The default value is 0.0, which means no check is performed.

SHORT_CLAMP_NODE : shorts clamp nodes “on-the-fly” in ESD-R and ESD-CD checks when resistances are computed.

CACHE_EM: for ESD checks for multiple from/to points, setting this to 2 significantly reduces the EM checking runtime and CD analysis time. Setting to 1 gives an intermediate speed-up. Note that when using this keyword EM results for a few corner cases can be slightly less accurate (<5%). Setting to 3 stores EM limits in the PF-S DB, resulting in improved run and reload time for ESD CD checks with multiple ‘from/to’ zap points. Default is 0 (off).

EM_THRESHOLD : saves the ESD-CD database if the EM % is \geq this threshold value.

REFF_THRESHOLD : saves the ESD-CD database if the effective R between zap from/to points is \geq this threshold value.

I_THRESHOLD : saves the ESD-CD database if the zapping current between from/to points is \geq this threshold value.

V_THRESHOLD : saves the ESD-CD database if the zapping voltage between from/to points is \geq this threshold value.

Note: when none of the above thresholds are specified, all the ESD-CD test results are saved to the database.

SAVE_CLAMP_FAIL : when set to 1, the DB is saved when any clamps fail the breakdown threshold

SAVE_EM : when set, saves EM results. The saved EM results can be directly loaded back into RedHawk without rerunning EM checking, using the TCL command ‘import escdc <rule_name>’.

EXCEL 1 (or EXCEL_REPORT 1): creates a consolidated ESD report file in Excel format, *adsRpt/ESD/esd_excel.rpt*, for resistance and CD checks, so that the information can be easily sorted and processed using Excel.

Example rules file:

```
BEGIN_ESD_RULE
  NAME mydc
  TYPE DC
  LOOP_R 6.0
  BUMP_PAIR VSS1 VDD5
  ZAP_VOLTAGE 100
  ZAP_R 1000
  NET_PAIR_VTH 0.2 VDD VSS
  PEAK_VOLT 25.0
  DIFF_VOLT 7.0
END_ESD_RULE
```

Rules Files - Specified clamp paths

The format of the rule file for Mode 2 current density analysis is as follows:

```
BEGIN_ESD_RULE
  NAME <name>
  TYPE CURRENT_DENSITY
  BUMP_PAIR <bumpName1> <bumpName2> # or use PIN_PAIR
  FROM_NET <net_name1> <net_name2> ...
  TO_NET <net_name1> <net_name2> ...
  TERMINAL_NET <net_name1> <net_name2> ...
  FROM_NET_GROUP ?Power? ?Ground? ?Signal?
  TO_NET_GROUP ?Power? ?Ground? ?Signal?
```

```

TERMINAL_NET_GROUP ?Power? ?Ground? ?Signal?
PACKAGE_PIN [ 0 | 1 ]
ZAP_VOLTAGE <voltage_in_Volts> # or use ZAP_CURRENT
    ZAP_R <resistance_in_ohms>
    CLAMP_INST_PIN <instance> <locID1> <locID2>
    USE_CLAMP_FILE <file>
    NET_PAIR_VTH <VTH Value> <net_name1> ...
    PEAK_VOLT <peak_V>
    DIFF_VOLT <differential_V>
END_ESD_RULE

```

where

USE_CLAMP: specifies an ESD_PIN_PAIR in a clamp instance for DC analysis.
 Multiple entries of CLAMP_INST_PIN are allowed.

USE_CLAMP_FILE <file> : in the specified file, the following syntax can be defined.

```

USE_CLAMP_CELL
    <cellName> [<locID1> [<locID2>]]
    ...
END_CLAMP_CELL
USE_CLAMP_INST
    <instName> [<locID1> [<locID2>]]
    ...
END_CLAMP_INST

```

Note that wildcards are honored in the clamp cell and instance names, and the locIDs are optional. The above keywords can be applied to B2B/B2C/CD rules for clamp selections.

Example rules file for Mode 2:

```

BEGIN_ESD_RULE
    NAME dc
    TYPE DC
    BUMP_PAIR VDD8 VSS4
    ZAP_VOLTAGE 100
    ZAP_R 1500
    NET_PAIR_VTH 0.2 VDD VSS
END_ESD_RULE

```

Outputs are reported in the following files:

Out_dir/esd_summary.rpt : reports IR values across the analyzed clamp instances

Out_dir/esd_em.rpt : reports EM values of the B2B paths

In addition to these output reports, results of the analysis are also viewable in several maps, using the menu options identified in ESD results map menu: See next section.

Bump-to-Clamp and Clamp-to-Clamp Current Density Checking

In addition to bump-to-bump checks through clamps, PathFinder also supports arc-based current density checks for bump-to-clamp, clamp-to-pin and clamp-to-clamp rules.

Rules File

The following rules file keywords are available for arc-based ESD-current density checks:

ZAP_BUMP_CLAMP <bumpName> <instName> <locID> : checks from bump to clamp instance

ZAP_CLAMP_BUMP <instName> <locID> <bumpName>: checks from clamp to bump

ZAP_CLAMP_CLAMP <inst1> <locID1> <inst2> <locID2> : checks from clamp to clamp

ZAP_PIN_CLAMP <pinName> <instName> <locID> : checks from pin to clamp

ZAP_CLAMP_PIN <instName> <locID> <pinName> : checks from clamp to pin.

To enable current density checks within a specific net, the following keywords are available:

ZAP_B2C_NET <netName> : bump to clamp for the net(s) specified

ZAP_C2B_NET <netName> : clamp to bump for the net(s) specified

ZAP_C2C_NET <netName> : between clamps belonging to the net(s) specified

ZAP_P2C_NET <netName> : pin to clamp for the net(s) specified

ZAP_C2P_NET <netName>: clamp to pin for the net(s) specified

Also, the same set of CD checks can be run for all nets of a specified type, as follows:

ZAP_B2C_NET_GROUP {POWER | GROUND | SIGNAL}

ZAP_C2B_NET_GROUP {POWER | GROUND | SIGNAL}

ZAP_C2C_NET_GROUP {POWER | GROUND | SIGNAL}

ZAP_P2C_NET_GROUP {POWER | GROUND | SIGNAL}

ZAP_C2P_NET_GROUP {POWER | GROUND | SIGNAL}

These ZAP_*_NET_GROUP rules provide zapping to all bump/clamp/pin pairs of one type of net -- POWER, GROUND, or SIGNAL.

For C2C and B2C current density checks, you can define a RADIUS within which a clamp instance check is selected. And for C2C current density checks, you can specify the type of clamps between which the check is to be performed. The keywords for these functions are:

FROM_CLAMP_TYPE <type> :

TO_CLAMP_TYPE <type>

SAME_CLAMP_TYPE [0 | 1] : if On, checks the same clamp type

DIFF_CLAMP_TYPE [0 | 1] : if On, checks different clamp types

TIE_REF_CLAMP <method> ?<m>?: allows multiple parallel clamps ("m") to be included (shorted together) to perform current density checks from clamps on the reference side as "zap from". This enables handling of multiple-finger diodes in clamp-to-clamp ESD-CD checks. The <method> options are:

PARALLEL: clamps connected in parallel. When *m* is not specified, includes all such clamps.

INST: clamps in the same instance (such as many fingers in a cell). When *m* is not specified, includes all such clamps.

MIN_DIST: includes *m* closest clamp nodes. The default for *m* is 1; in other words, the same as when 'TIE_REF_CLAMP' is not specified. So for 'MIN_DIST', you should specify *m* greater than 1.

For arc-based current density checking you can also select the closest set of clamps, based on the ESD rule keyword 'FROM_TO_SELECT', which allows selecting points to be connected to the zapping source in the analysis. The syntax is:

```
FROM_TO_SELECT <method> <count> ?<1_to_many [0|1]>?
```

where <method> is one of the options:

```
FROM_MIN_DIST
FROM_MIN_RES
TO_MIN_DIST
TO_MIN_RES
```

and

FROM*/TO* : indicates at which end of the zapping source the selection takes place. For example, for a net specified by 'ZAP_B2C_NET', the bumps in the net are connected to the positive terminal of the zapping source, and the points in the clamps are grounded.

*MIN_RES : selects from/to points that are connected with smallest effective resistances.

*MIN_DIST: selects from/to points that are close in physical distance in the layout,

<count>: optional, specify how many points to selected by <method>. It takes a positive integer value, default 1.

<1_to_many [0|1]>: specifies connecting (shorting) all selected points together to do 1-to-many zapping. Default is 0 (off). In effect only when the <count> setting is greater than 1.

Example: "FROM_TO_SELECT FROM_MIN_DIST 10" means to select 10 bump points to connect to the positive terminal of the zapping source, and to be zapped with each clamp point, respectively.

An example of this rule file syntax is:

```
BEGIN_ESD_RULE NAME rule_name
    TYPE CURRENT_DENSITY
    ZAP_CURRENT 1
    ZAP_B2C_NET <> (or) ZAP_C2C_NET
    FROM_CLAMP_TYPE clampA
    TO_CLAMP_TYPE clampB
    FROM_TO_SEL TO_MIN_DIST 5 1
END_ESD_RULE
```

CD and arc-based resistance checks include clamp resistance

CD and arc-based resistance checks can include clamp resistance for better current distribution. This includes R-type point-to-point resistance checks as well to perform a subset of the potential path resistances for B2C and C2C arc-based analysis.

ESD rule syntax example:

```
BEGIN_ESD_RULE
    NAME BUMP_TO_D2_D4
    TYPE CD      # 'TYPE R' also supported
    ZAP_CURRENT 1.3
    ZAP_B2C_NET_GROUP SIGNAL
```

```
# ZAP_C2C_NET, ZAP_BUM_CLAMP, also supported

# To include all clamps connected to the zap from/to
points:
    USE_CLAMP_IV 1
# To specify particular clamp devices to be used:
    USE_CLAMP_IV
        CELL  <cellName(s)>
        INST  <instName(s)>
        TYPE  <clampType(s)>
    END_CLAMP_IV
END_ESD_RULE
```

Point-to-point current density checks

Multiple point-to-point zapping checks for both current and voltage can also be performed using special options of the 'perform esdcheck' command, as follows:

```
perform esdcheck
    -from {<x> <y> <layer> <net>} -to {<x> <y> <layer><net>}
    ...
    ? [{-zapI <I> | -zapV <V> -zapR <R>} -ruleName <name>
        -clamp <file> ]?
```

where

- from/-to: specifies the zapping check points, and optional layer and net
- zapI <current> : zapping current (default 1A) for point-to-point ESD resistance checks on a pair of points in the layout
- zapV <voltage> -zapR <resistance> : zapping voltage and zapping source resistance (default is '-zapI') for point-to-point checks
- ruleName <name>: saves the zapping results into the ESD DB so it can be loaded back. When not specified, the DB is not saved.
- clamp <file>: specifies the clamp file name (see the clamp file section for a description of the clamp file syntax and an example clamp file).

For the 'from'/'to' point specified in the command line, the command finds the closest nodes that match the specified points. Other command options and rules are also available for checking 'from'/'to' zapping paths, as follows:

- between bumps


```
-fromBump <bumpName> -toBump <bumpName>
ZAP_FROM_BUMP <bumpName> ZAP_TO_BUMP <bumpName>
```
- between clamps


```
-fromClamp {<instName> <locID>} -toClamp {<instName> <locID>}
ZAP_FROM_CLAMP <clampInst> <locID> ZAP_TO_CLAMP <clampInst>
<locID>
```
- between instance nets and pins


```
-fromInst {<instName> <netName>} -toInst {<instName> <netName>}
-fromInst {<instName> <pinName>} -toInst {<instName> <pinName>}
ZAP_FROM_INST <instName> {-net:<name>|-pin:<name>}
ZAP_TO_INST <instName> {-net:<name>|-pin:<name>}
```

For the specified instance, a node in the specified net connected to the pins of the instance is selected. More than one '-from' and '-to' check can be specified, where all of the '-from' points are shorted and connected to the positive terminal of the zapping source, and all the '-to' points are connected to the ground terminal of the zapping source. All points to be shorted should be in the same net (or connected by power gating switches), and also physically connected in the layout. Otherwise checking is not performed.

PF-S also can form bump pairs between different nets in the specified list of nets to perform current density analysis using FROM/TO_NET rules.

Viewing Current Density Checking Results

ESD CD Report Command

After an ESD CD check is performed with the 'perform esdcheck' command, RedHawk saves the results into the ESD DB. The data is identified by the rule name. If multiple ESD-Current Density checks are performed with the same rule name, a suffix "_1", "_2" ... is added to the specified rule name to provide separate current density check files. You can get text reports for any ESD current density check with the TCL command:

```
report esdcheck -type <ruleType> ? -outDir <output_dir>?
? -name <rule_name>? ?-[glob | -regexp ] ?
? -arcR <arc_threshold>? ? -loopR <loop_threshold>?
? -parallelR <parallel_threshold>? ? -append? ? -excel?
? -cell { <cell1> ... <cellN> }? ? -failedOnly?
? -detail? ? -netPairVth <Vth_value>? ,<net1>, . . . ,<netN>?
? -peakVolt <peak_V>? ? -diffVolt <diff_V>?
```

where

- type: reports results for the specified type of ESD checks. Default is all types.
- outDir : specifies the report files directory. Default is *adsRpt/ESD*
- name: specifies the name of the rule tested. If multiple checks have been specified within a single rule-- for example, two bump pairs are specified in the same rule-- PathFinder internally splits them into subrules, each with a single bump pair. Then it appends '(_\d+)+' to the base rule name. So when using 'report esdcheck', you can specify this derived name to look at the particular results desired.
- glob|-regexp: specifies the type of wildcard expressions used. PathFinder reports the results for all subrules that match the specification; 'glob' supports regular wildcard expressions and 'regexp' supports normal wild cards and also extended character strings. For example:

```
report esdcheck -detail -perNet -type B2C -name abc* -glob
report esdcheck -type B2C -name {M_test_[3-5]} -regexp
```

- arcR: specifies the threshold for arc-R checking; required for B2C, C2C, C2I, C2M
- loopR: specifies the threshold for loop-R checking; required for C2I
- parallelR: specifies the threshold for parallel-R checking; required for B2B
- append: appends results to existing results in the output directory.

- cell : reports CD results for specified cell names
- failedOnly: only reports failed results.

- detail : provides more detailed report results for B2B rules
- netPairVth: reports driver-receiver voltage pair threshold violations for current density (CD) checking rules. Optionally, one or more nets with the same VTH value can be specified, separated by commas.
- peakVolt: specifies the peak voltage to be used for CD checks
- diffVolt : specifies the differential voltage to be used for CD checks

You can load results from a current density check named, for example, ‘RuleABC’ with the following command :

```
import esdcd RuleABC
```

A sample output report in the *adsRpt/ESD/esd_inst.rpt* file is shown following (voltage drop):

```
# ESD-CD Analysis Instance Voltage for Rule <esd_cd> (CURRENT_DENSITY)
BEGIN_ESD_RULE
NAME esd_cd
TYPE CURRENT_DENSITY
BUMP_PAIR DVSS13 DVDD14
ZAP_VOLTAGE 1000 +
ZAP_R 1000
END_ESD_RULE
# <MaxVoltage> <DiffVoltage> <X> <Y> <Inst Name> <Cell Name>
0.7956 0.7202 4820.535 2377.665 inst_129425/inst_508698 CellABC
0.7956 0.7208 4820.535 2381.345 inst_129425/inst_508699 CellFGH
```

In detailed reporting mode, 'report esdcheck -detail <options>', for the header lines representing the instances of the detailed node voltages, the format is:

```
# <Inst Name> <Cell Name> (<MaxVoltage> <DiffVoltage> <X_loc> <Y_loc>)
<node_data> ...
```

A sample output file is shown following for a rule “esd_cd_1” (current density):

```
Rule name: esd_cd_1
EM mode: PEAK
Worst EM: 7748.7%
Zap From:
Bump(DVSS11) (4880 2138.85 METAL4 VSS)
Zap To:
Bump(DVDD11) (4905 2078.85 METAL4 VDD)
Zapping source: 1000 V @ 1000 Ohms
Equivalent resistance: 1.4031 Ohms (1.40114 V, 0.998599 A)
Currents on clamp devices: min 157.482 mA, max 421.800 mA
# Clamp device I(mA)/V(volt)/R(Ohm) list:
# <I> <V> <RON> (<V> <X> <Y> <LAYER> <NET>)
# (<V> <X> <Y> <LAYER> <NET>) <INST> <locID1> <locID2>
421.800 0.0000 0.0001 (0.7372 4521.65 2201.16 METAL4 VSS)
(0.7371 4543.78 2197.32 METAL4 VDD) inst_129425/inst_7773 VSS VDD
252.563 0.0000 0.0001 (0.7593 3196.97 2201.16 METAL4 VSS)
(0.7593 3199.43 2197.32 METAL4 VDD)
```

Regular output results for current density checks are reported in the following files:

Out_dir/esd_em.rpt : reports EM values of the B2B paths.

Out_dir/esd_summary.rpt : reports IR values across the analyzed clamp instances. Print out the Diode/Clamp instance current in *esd_summary.rpt* during CD check when 'USE_CLAMP_IV 1' is set in CD check rule file and 'jobCount' option is specified in 'perform esdcheck' command

Sample output in *esd_summary.rpt*

```
# Clamp device I(mA)/V(volt)/R(Ohm) list:
# <I> <V> <Ron> (<V> <X> <Y> <LAYER> <NET>) (<V> <X> <Y>
<LAYER> <NET>) <INST> <locID1> <locID2>
374.953 0.0375 0.1000 (0.0375 19.74 224.868 pdiff gnd1)
(-0.0000 19.88 224.868 ndiff pad2) D1_2(D1) gnd1 pad2
```

Results in compressed mode

Current density checking results are saved in compressed mode by default, which provides a 50% to 90% data size reduction. You can choose *not* to save the results in compressed form by using the '-noCompress' option in the 'perform esdcheck' command, or by specifying the 'COMPRESS_DB 0' keyword in the ESD rule file.

ESD-CD report *esd_summary.rpt*

The following is an sample *esd_summary.rpt* for **ESD-CD**:

```
# DC Analysis Summary for Rule <DC_RULE> (CURRENT_DENSITY)

BEGIN_ESD_RULE
    NAME DC_RULE
    TYPE CURRENT_DENSITY
    BUMP_PAIR VDD_1 VSS_2
    ZAP_VOLTAGE 1000
    ZAP_R 1000
END_ESD_RULE

# BUMP PAIR: (<BUMP1> <X> <Y> <LAYER> <NET>) --> (<BUMP2> <X> <Y> <LAYER>
<NET>)
BUMP PAIR: (VDD_1 700 1600 M4 VDD) --> (VSS_2 1300 400 M4 VSS)
Zapping source: 1000 V @ 1000 Ohms
Equivalent bump-to-bump resistance: 3.47856 Ohms (3.4665 V, 0.996533 A)
Clamp device current: 0.000 mA to 550.579 mA

# Clamp device I(mA)/V(volt)/R(Ohm) list:
# <I> <V> <RON> (<V> <X> <Y> <LAYER> <NET>) (<V> <X> <Y> <LAYER> <NET>
<INST> <locID1> <locID2>
550.579 0.6955 1.2632 (1.6574 1325 180 M3 VDD) (0.9619 1325 130 M3 VSS)
PVSS_D1_I17 VDD VSS
445.952 0.6945 1.5573 (2.0644 1025 1820 M3 VDD) (1.3700 1025 1870 M3 VSS)
PVSS_D1_I16 VDD VSS
0.000 0.9109 off (2.2396 725 1820 M3 VDD) (1.3288 725 1870 M3 VSS)
PVDD_D1_I14 VDD VSS 0.000 0.7686 off (1.7610 1025 180 M3 VDD) (0.9925 1025
130 M3 VSS) PVDD_D1_I15 VDD VSS
0.000 0.4350 off (1.8050 1025 1980 M3 VSSIO) (1.3700 1025 1870 M3 VSS)
PVSS_D1_I16 ADS_VSS1 VSS
```

ESD Current Density Reports for Pads

PathFinder creates a Pad current report and maps for ESD current density checks, as shown in the following sample output report, *adsRpt/ESD/esd_pad.rpt*:

```
# ESD-CD Analysis Pad Current for Rule <esd_cd_1> (CURRENT_DENSITY)
BEGIN_ESD_RULE
  NAME esd_cd_1
  TYPE CURRENT_DENSITY
  BUMP_PAIR DVDD11 DVSS11
  ZAP_VOLTAGE 1000
  ZAP_R 1000
  B2B_MIN_BOUND 1
END_ESD_RULE
# <Current(mA)> <X/Y Location> <Pad Name>
  0.0000 ( 4880.000, 2138.850) DVSS11
  0.0000 ( 4880.000, 2378.850) DVSS13
```

You can view a color map of pad current using the command **ESD Current Density -> Pad Current Map**

ESD EM Report for ESD-CD

RedHawk supports the 'get em' command for ESD current density checking. The following is a sample output *esd_em.rpt* for **ESD-CD**, which reports EM violations, that is, $\text{EM_Ratio} = \text{Actual_Current_Density}/\text{Current_Density_Limit} > 100\%$ (default or from "em_report_percentage") for wire pieces and vias in decreasing order. The unit used for coordinates and dimensions is um. By setting 'EM_BY_LIMIT_ONLY 1' in ESD_RULE, the worst EM percentage lower than EM_REPORT_PERCENTAGE value is not reported in *esd_em.rpt* file.

```
# ESD EM Check Results for Rule <DC_RULE> (CURRENT_DENSITY)

BEGIN_ESD_RULE
  NAME DC_RULE
  TYPE CURRENT_DENSITY
  BUMP_PAIR VDD_1 VSS_2
  ZAP_VOLTAGE 1000
  ZAP_R 1000
END_ESD_RULE

# For wires: #layer #end-to-end_coordinates #EM_Ratio #net #width
# For vias: #via_name #x-y_coordinates #EM_Ratio #net

M2  (1334.000,106.000 1334.000,112.000) 4044.41%  VSS  5.000
M2  (728.000,1838.000 728.000,1894.000) 3932.45%  VDD  5.000
M2  (704.000,1838.000 704.000,1894.000) 3733.18%  VDD  5.000
M2  (1310.000,106.000 1310.000,112.000) 3621.23%  VSS  5.000
via VIA2_800_800_9 (704.000,1838.000)    2828.35%  VDD
via VIA2_800_800_9 (1310.000,106.000)    2743.53%  VSS
M4  (1320.858,224.142 1320.858,226.232) 1916.46%  VSS  20.000
```

```
M4 (1312.501,232.500 1304.142,238.768) 1916.46% VSS 20.000
M4 (1320.858,226.232 1312.501,232.500) 1916.46% VSS 20.000
```

Displaying Current Density Checking Results in the GUI

To display in GUI the list of zap tests in PathFinder Driver-Receiver CDM check, use "View->ESD Current Density->Current Density Test List". In addition to text output reports, results of the current density analysis are also viewable in several types of color maps, using the menu options shown in Figure 16-21, which shows a color map selected from a Test List.

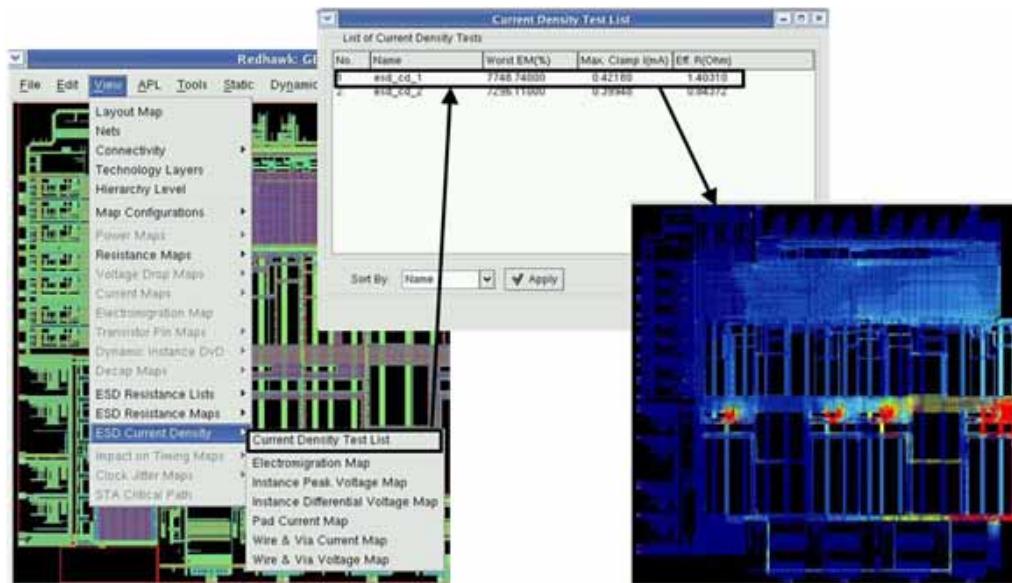


Figure 16-21 ESD color map menu

Peak and Differential Voltage Maps

Selecting and viewing peak and differential voltage drop maps is shown in Figure 16-22:

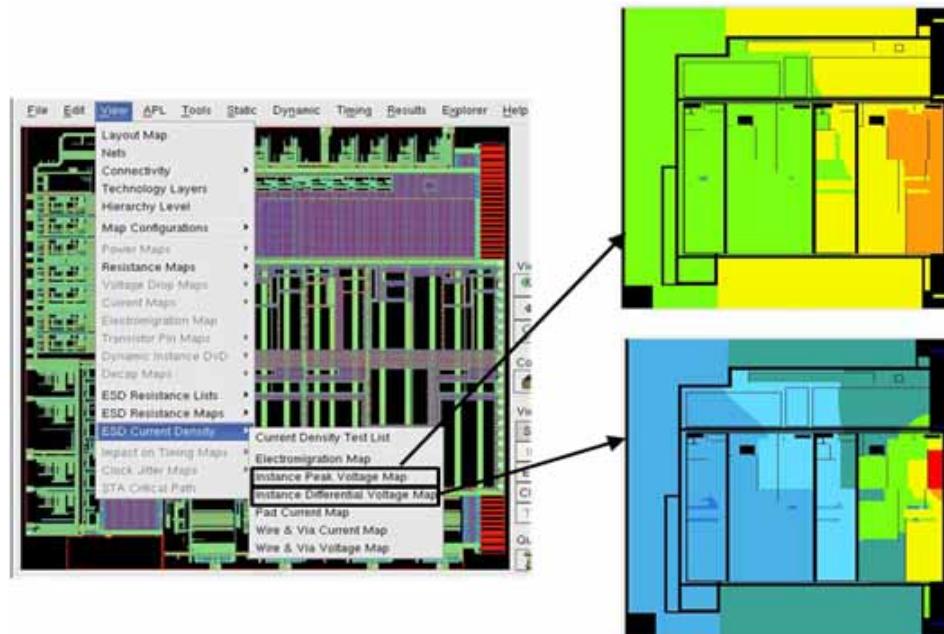


Figure 16-22 Peak and differential current density maps

Current Maps

To run current density checks on wires and vias in the design, use the command **View->ESD Maps->Current Map**. The results of current density checking (rule type “CURRENT_DENSITY”, or “DC” or “CD” as abbreviations) can be viewed in three types of plots, as shown in the following examples.

Figure 16-23 shows the current of wires and vias from ESD current density checking.

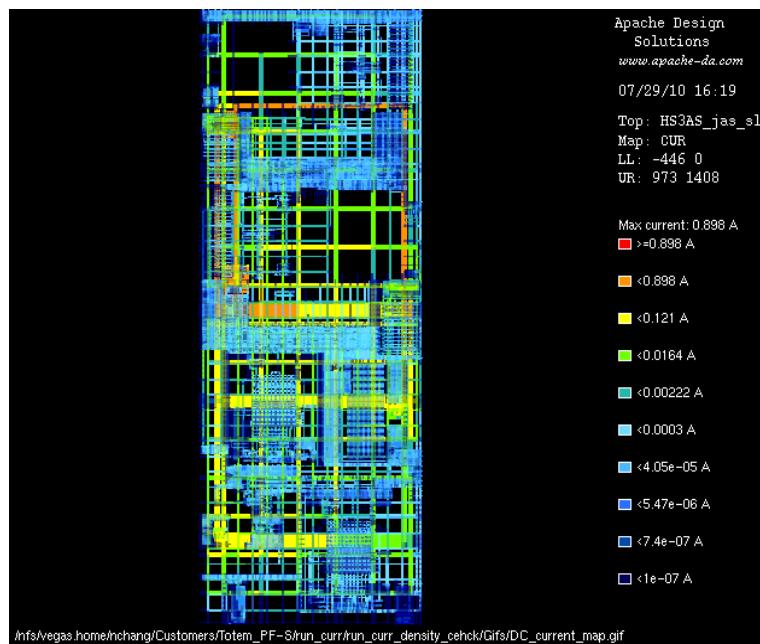


Figure 16-23 Results of current density checking

Wire and Via Voltage Maps from Current Density Checks

Use the menu command **View->ESD Maps->Wire & Via Voltage Map** to display the voltages for wires/vias after ESD current density checking. A sample map of wire/via voltages is shown in Figure 16-24. You can also zoom in and select a particular wire/via/clamp to see detailed current/voltage values displayed in the Log window.

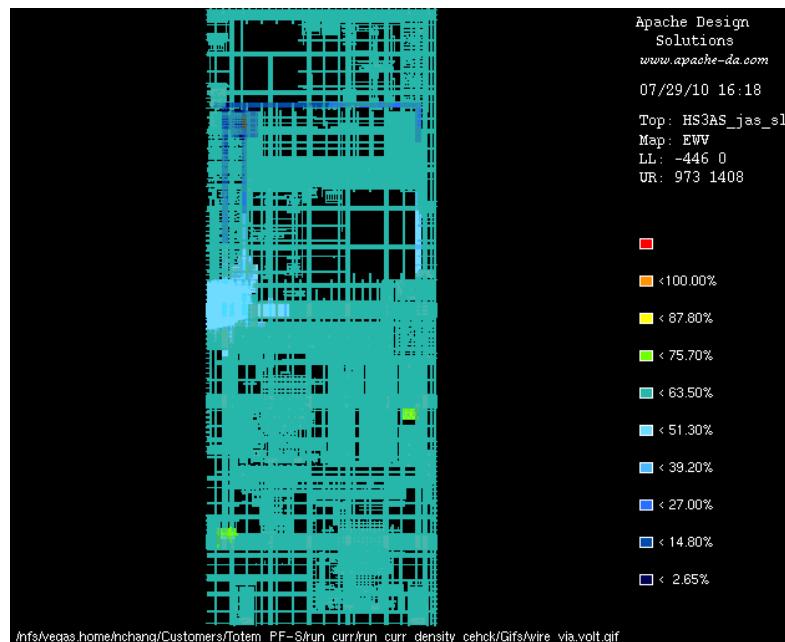


Figure 16-24 ESD checking results for wire/via voltages

Electromigration Maps

A sample electromigration map is displayed in Figure 16-25.

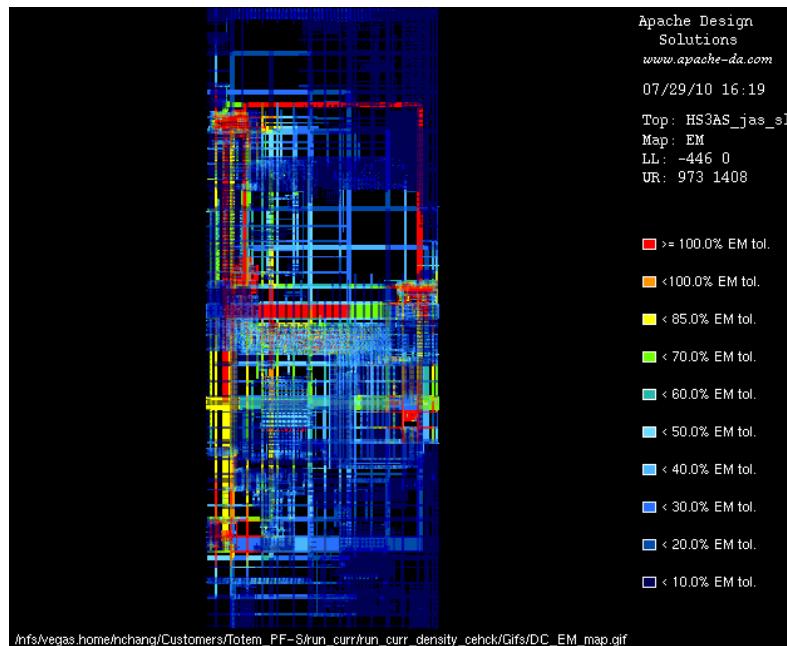


Figure 16-25 EM checking results

General Rule File Inclusions and Exclusions

Both resistance checking and current density checking rules files can have specifications for clamps, bumps, instances, or pins that are to be included or excluded for a particular ESD check. Some examples are given in this section.

Clamp Element Exclusions

You can exclude specific clamp cells, instances, or pins from an ESD check by including the following keywords in the rule file:

```
EXCLUDE_CLAMP <cellname>
EXCLUDE_CLAMP_INST <instance_name>
EXCLUDE_CLAMP_CELL_PIN <cellname> <loc_ID>
EXCLUDE_CLAMP_INST_PIN <cellname> <loc_ID>
```

Chapter 17

Memory and Mixed Signal Design Analysis

Introduction

Memory and Mixed Signal (MMX) analysis executed at transistor level is more and more important in today's advanced semiconductor design scenarios. Key MMX applications that benefit from accurate power integrity analysis include:

- High performance I/Os, such as DDR2, PCI-X, USB, Serdes, PLL, and DLLs
- Embedded memory macros, such as compiled memories, register files, TLBs, and cache memories
- Memory chips, such as CAM, DRAM, SRAM, and Flash memory

The advantages of MMX modeling are:

- Full-chip dynamic analysis on SOCs with a mixture of custom macros and cell-based digital blocks
- Large capacity and shorter runtime (compared to a fast Spice approach)
- Layout-based GUI for ease of debugging and diagnosis

For detailed information about performing accurate Memory and Mixed Signal (MMX) power analysis, power/ground grid checks, static IR drop analysis, and electromigration and dynamic voltage drop analysis on memories and custom macros with transistor-level accuracy, see the [Apache “Totem User Manual”](#). Using this detailed characterization modeling approach, device models are accurate, both spatially (relative to location) and temporally (relative to time of switching).

For information on creating physical GDS models in [RedHawk](#), see [section "gds2rh/gds2def"](#), page E-875.

Chapter 18

Chip Thermal Modeling and Analysis

Introduction

With the emergence of SiP (System-in-Package) and 3D-IC methodology, special handling of heat transfer is required within multiple stacked dies. Otherwise, unexpected chip failures may occur from thermal runaway caused by die temperature and leakage current interactions. Power-thermal integrity analysis is a must for multiple-die SiP or 3D-IC designs. Figure 18-1 shows the heat transfer mechanisms in a typical chip-package combination. Electrical energy on the chip is converted to thermal energy, which is then conducted away through the molding, substrate, solder joints, and thermal board to the exterior surfaces, and then is dissipated to the ambient air by convection and radiation heat transfer.

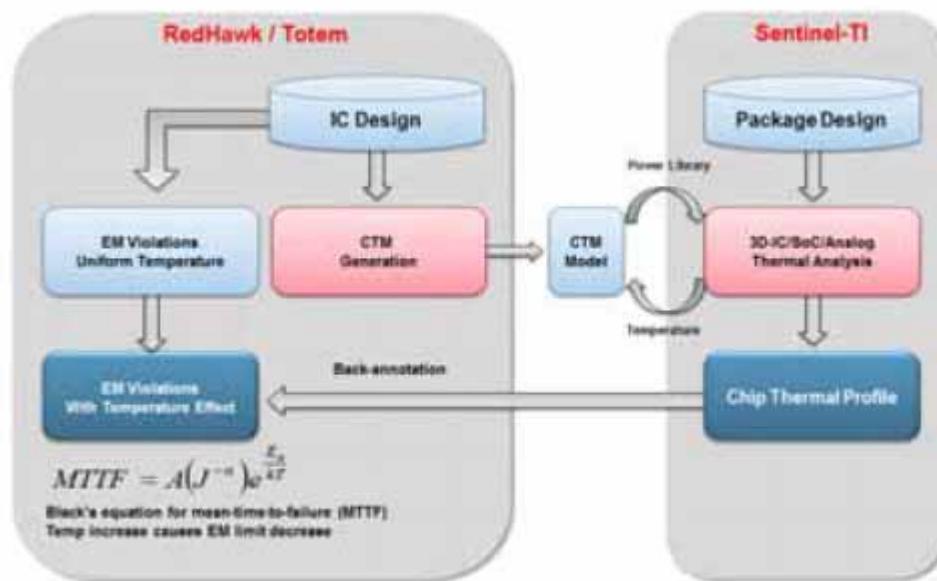


Figure 18-1 Heat transfer mechanisms in a typical package environment

Figure 18-2 is an example of a typical SiP configuration with two stacked dies. The close spacing of lower and upper chips requires greater heat dissipation capability than if these components were in single-chip packages. Higher on-chip power consumption naturally leads to higher die temperatures. However, junction temperatures cannot safely exceed 120°C. Therefore, understanding the actual on-chip power is critical in determining if the thermal design is feasible. In addition, the on-chip power is temperature-dependent, thus making accurate power and temperature analyses inseparable.

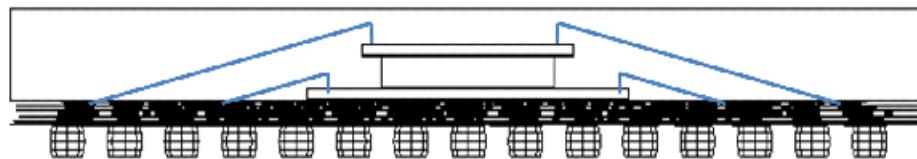


Figure 18-2 Configuration of SiP with two stacked dies, with spacer

For designs at the 28nm technology node and below, analyzing and managing leakage current has become one of the key design challenges for achieving successful high performance chip designs. Leakage current is strongly (exponentially) dependent on temperature. Hence to accurately analyze individual instance leakage currents, designers must consider temperature variation across the chip produced by unevenly distributed power consumption. Local chip temperatures also affect metal resistivity, interconnect self-heating, electromigration, and voltage drop in the design. An example of a power-thermal analysis loop is shown in Figure 18-3. The total power and maximum temperature variations can be more than 20% in this example.

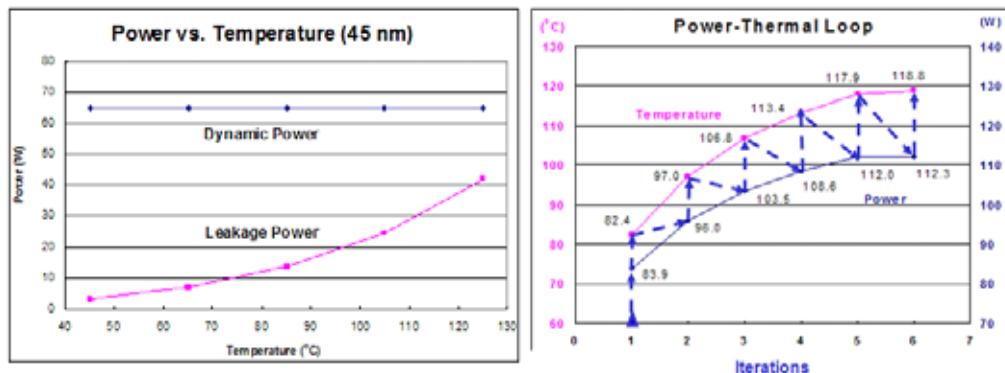


Figure 18-3 Left - exponential increase of leakage vs. temperature. Right - power-thermal analysis loop needed to reach final equilibrium temperature and leakage power

Apache's **Sentinel-TI** temperature-dependent chip-power database in CTM, along with a detailed package-on-board thermal model, creates an on-chip power and temperature convergence map. **Sentinel-TI**'s tightly integrated CTM-based thermal analyses deliver accuracy, capacity, performance, and ease-of-use for fast convergence of power and thermal distribution. **Sentinel-TI** consists of a two-step process, as shown in Figure 18-4 and Figure 18-5.

The first phase is CTM generation, which includes determination of on-chip power, temperature-dependent leakage power, and a per metal layer tile density map in **RedHawk**. The second phase is standalone thermal analysis performed in **Sentinel-TI** using CTM. **Sentinel-TI** supports multi-chip package configurations, with or without the CTM of each individual chip. The converged temperature distribution on individual chips supports full-chip integrated power-thermal analysis.

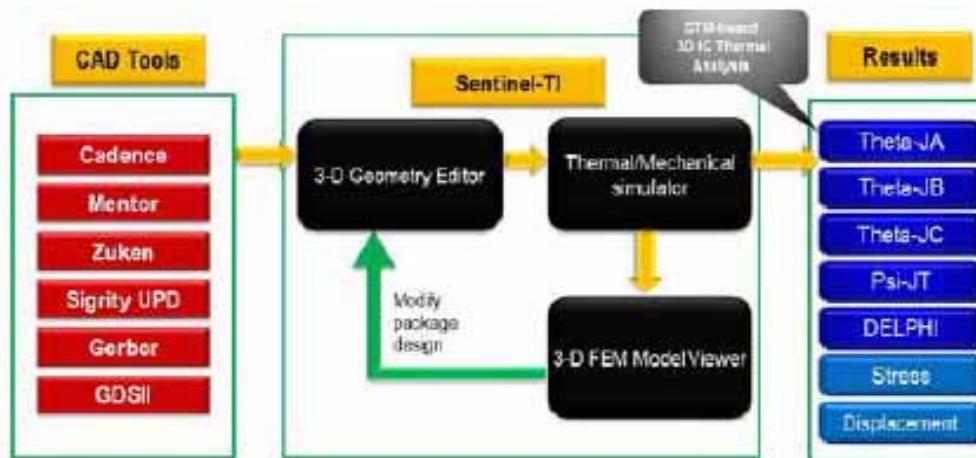


Figure 18-4 Task flow in CTM-based thermal analysis with Sentinel-TI

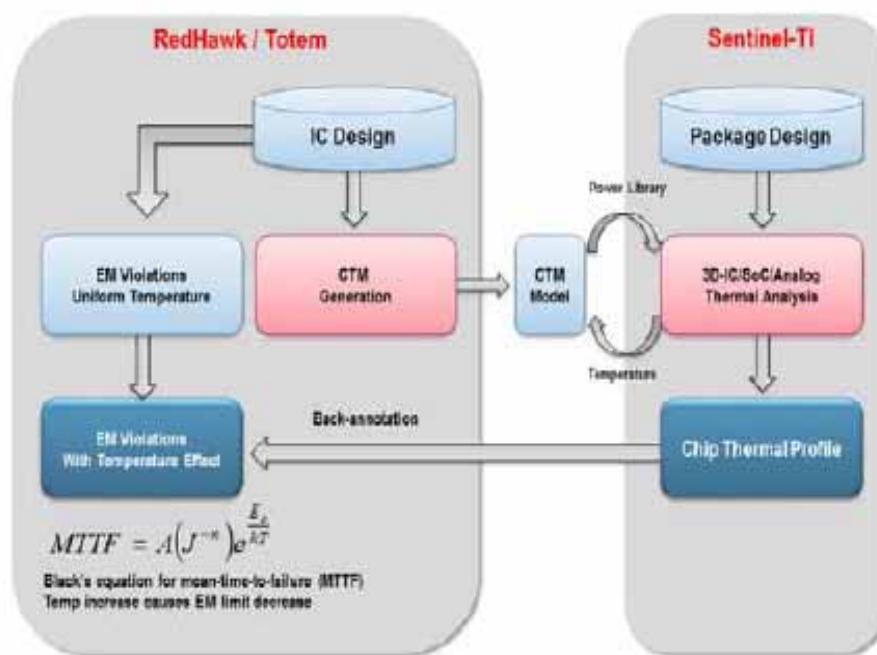


Figure 18-5 Chip-package thermal analysis methodology

CTM-Based Thermal Analysis Flow Overview

The key steps for running CTM-based Thermal Analysis are as follows:

1. CTM generation using **RedHawk** for each chip (by chip designer)
 - a. Generate temperature-dependent leakage library using Apache Power

- Library (APL), and optionally the per metal layer tile density map.
- b. Prepare design data and input files:
 - technology file (*.tech) data for the IC process.
 - pad cellname, pad instance name, or pad location file.
 - Global System Requirement (GSR) file (including the required data from tech files, pad files, STA file, LEF files, DEF files, and LIB files).
 - c. Import design data defined in the GSR file.
 - d. Execute ‘perform thermalmodel -layer’ in RedHawk to generate portable power/temperature library.

These steps are described in more detail in the following section.

2. Thermal Analysis of the SiP using Sentinel-TI with the CTM of each chip (by package designer).
 - a. Generate finite-element thermal-model for SiP or 3D-IC package.
 - b. Set path to package model file and CTM libraries.
 - c. Iterate on-chip temperature/power analysis inside the analysis model for the converged temperature and power .
 - d. Report/display converged temperature/power profile for each chip.
 - e. Export on-chip temperature map for further evaluation of chip performance in RedHawk, such as for EM.

Please refer to the *SiP Chip-Package Thermal Analysis Application Note* for details on Sentinel-TI thermal data preparation and analysis.

Data Preparation for CTM Generation

APL Library Characterization

Since leakage current is highly temperature-dependent, leakage power calculation for a chip must consider the local chip temperatures. The APL utility ‘apldi -leak’ characterizes temperature-dependent leakage current for efficient power calculation.

To start APL characterization, perform the ‘setup apl’ command on the design in RedHawk, similar to any design-dependent APL characterization. The steps typically involved are:

```
setup design <gsr_filename>
setup apl # in dir APL
```

In the directory ‘APL’, RedHawk creates three files: <APL config template file>,.apache/.adsLib.output, and .apache/apache.apl, which are used for the subsequent APL characterizations.

APLDI Leak

The ‘apldi -leak’ program is used for temperature-dependent leakage current characterization of standard cells. It should be run in the same directory where you have the .apache directory containing the apache.apl and adsLib.output files created as described above.

The configuration file setting is the same as for a typical APL run, except for the temperature setting APL configuration file keyword ‘SWEEP TEMPERATURE’, which specifies the cell temperatures for characterization. Since ‘apldi -leak’ supports multiple-temperature characterization, SWEEP TEMPERATURE can specify multiple temperature

settings. Performing characterization for at least five temperatures in the expected temperature range is recommended.

Note: 400 cells across five temperatures takes approximately 6 hours to run on a 3 GHz Linux32 machine.

The use of the SWEEP TEMPERATURE configuration file keyword is as follows:

Syntax:

```
SWEEP TEMPERATURE <temp1> <temp2> ... <tempN>
```

Example for 5 temperature points:

```
SWEEP TEMPERATURE 25 55 80 105 120
```

For other configuration settings, please see [Chapter 9, "Characterization Using Apache Power Library"](#) for details. The syntax for ‘aplди -leak’ is as follows:

```
aplди -leak <config_file> [-c][-d][-log][-l <list_file>]  
[-v]
```

where

- <config_file>: specifies the input control file
- c: runs intrinsic decap and leakage characterization
- d: debug mode
- log: redirects output message to log file
- l <list_file>: specifies file containing list of cells
- v: prints out messages in detail (verbose mode)

Example:

```
aplди -leak apl_std.conf -l std_cell.list
```

The output file is called *cell.leak*. For memory cells, the AVM program can be used to generate temperature-dependent leakage current (see the AVM section below).

APPLEAKMERGE

If more than one *.leak file is generated, ‘appleakmerge’ can be used to merge the files. The syntax for ‘appleakmerge’ is as follows:

```
appleakmerge [-d][-v][-o <out_file>] <file1> <[file2 [file3 [...]]]>  
-fl <list_filename>
```

where

- d: debug mode
- v: print messages in detail (verbose mode)
- o <out_file>: output file
- <file[1,2,3, ...]>: specifies files to merge
- fl <list_filename> : specifies a file containing a list of all leak files to be merged.

Example:

```
appleakmerge file1 file2 *.leak -o cell.leak
```

AVM

AVM is a datasheet-based program for easy characterization of memories. Since ‘avm’ does not run memory simulation, the models can be obtained relatively quickly. AVM also supports temperature-dependent leakage current by using the ‘-t’ option, followed by a reference *.leak file. The *.leak file can be taken from the output of standard cell characterization using ‘aplди -leak’. The temperature-dependent leakage current data is used as the temperature derating factor for ‘avm’. For example, if you simulate five similar cells with four temperature points, the average leakage current at different temperatures is used to obtain the derating factor for scaling. The syntax for invoking avm is:

```
avm <config_file> [-c][-t <ref_leakage_file>]
```

where

- <config_file>: specifies the input control file
- c : changes all cellnames to upper case
- t <ref_leakage_file> : specifies the reference file for the temperature derating factor

Example:

```
avm avm.conf -t cell.leak
```

GSR Keyword Settings

The preparation of the GSR file is essentially the same as in standard RedHawk chip analysis. A few additional settings required for CTM generation are discussed in this section.

APL_FILES

Imports the APL characterized files or directories. For temperature analysis and leakage characterization, you must use the option ‘leakage’ or ‘leak’.

Syntax:

```
APL_FILES {
    <file/directory> ?leakage?
    ...
}
```

Example:

```
APL_FILES {
    qcell_file.leak leak
    cell_leak_dir leakage
}
```

POWER_MODE

Defines the source for internal and leakage power calculation analysis. For CTM generation, temperature-dependent leakage current is used if cell leakage files are included in APL_FILES.

Syntax:

```
POWER_MODE [ APL | LIB | MIXED ]
```

where

APL: uses APL/AVM characterized leakage current and power for power calculation

LIB : uses .lib data for power calculation

MIXED: uses .lib data if it contains internal power, otherwise uses APL data

Example:

```
POWER_MODE APL
```

THERMAL_MODEL

When set, enables generation of a CTM and temperature-dependent power calculation.

- If “THERMAL_MODEL 1” is specified, only cells covered by cell.leakage data are updated with temperature-dependent leakage.
- If “THERMAL_MODEL 2” is specified, in addition to steps included in “1”, cells not covered by cell.leakage data use the leakage-temperature scaling factor from covered cells for temperature-dependent leakage current updates-- that is, the cells having

- <cell>.cdev files.
- If “THERMAL_MODEL 3” is specified, cells are updated the same way as in setting “2”, except that the base leakage power is always calculated from LIB/CDEV.
 - If THERMAL_MODEL is not set, or is set to 0 (default), no thermal modeling is performed. However, if the THERMAL_PROFILE keyword is specified, and THERMAL_MODEL is not set, it defaults to 1.

Syntax:

```
THERMAL_MODEL [1 | 2 | 3 | 0]
```

THERMAL_PROFILE

With the THERMAL_MODEL keyword set to 1,2, or 3, and the ‘perform pwrcalc’ command run, specifying this keyword generates the specified chip thermal model file for thermal analysis, which maps tile and layer-based temperatures in the thermal profile to instances in the design, with the following format:

```
<inst name> <Tavg_C> <Tmax_C> <Tmin_C>
```

If the instance is in the intersection of the grid defined in <thermal_profile_file>, the maximum temperature is used from the intersected grid. *Optional; default: none.*

Syntax:

```
THERMAL_PROFILE {  
    FILE <thermal_profile_file>  
}
```

where the <thermal_profile_file> is from [Sentinel-TI](#) package modeling.

CTM Generation

After data preparation, characterization and setting the proper keywords, the CTM is generated with the following [RedHawk](#) TCL commands:

```
import gsr <file>  
setup design  
perform thermalmodel -layer ?-self_heat? ?-inst?  
?-die ? ?-d <Thermal dir> ? ? -ctmfilecheck ?
```

where

- layer: (required) generates layer metal distributions for use in chip thermal modeling in [Sentinel-TI](#)
- self_heat: generates self-heat power for individual layers, in addition to total power on the device layer, for use in CTM-based thermal analysis of [Sentinel-TI](#). The self-heat data are on power nets only by default.
- inst : generates the *ins_power_loc* file that provides instance name, range, and power for hot spot query uses in [Sentinel-TI](#).
- die: specifies the die name for a 3D-IC configuration.
- d : a <Thermal dir>.tar.gz file is generated in the thermal directory. If -d <Thermal dir> is not specified, an *adsThermal.tar.gz* file is generated in addition to the *adsThermal* directory.
- ctmfilecheck : generates the text format ctm file ‘adsThermal/chip.ctm.txt’

The generated CTM is a set of files for use by [Sentinel-TI](#) in chip thermal analysis. When ‘-layer’ is specified, it includes a header file (text), the *power_T[x].ctm* files (binary) at each temperature point specified in the leakage file, *metal_density.ctm* (binary) for metal

distributions in chip layers. If `--self_heat` is specified, `SH_T[1].ctm` and `SH_T[<last>].ctm` (binary) are also generated.

The `CTM_header.txt` file has the following content and format:

```
# version 2.0
LAYER <num_layers> <layer1> <L1_thick_um> <layer2> <L2_thick_um> ... // bottom to top
DIE <die x_ll> <die y_ll> <die x_ur> <die y_ur> // die outline in RedHawk (>110% enlarged)
TEMP <T1> <T2> <T3> <T4> <T5> // five temperature points of interest
TILE <nx> <ny> // number of tiles in x direction and y direction
RESOLUTION <size of tile in um> // 10.000 for 10um
DIE_ORIG <die x_ll> <die y_ll> <die x_ur> <die y_ur> // metal range in RDL
```

Chapter 19

Timing File Creation Using Apache Timing Engine (ATE)

Introduction

For accurate static and dynamic analysis in RedHawk, the Apache Timing Engine (ATE) provides timing information such as clock domains, slew (transition times) and timing windows. This information is provided through the timing file (also called the STA file).

In static analysis, RedHawk uses the timing file for deriving the frequency and transition time information, which is then used for calculating the average power for the design during the power calculation step. RedHawk also uses the clock network information to determine the total clock network power. For dynamic analysis, RedHawk uses timing window information from STA file. The timing window represents the earliest and latest (min/max) timing events for a particular instance pin. It is not needed for static analysis, but is needed for dynamic analysis in order to calculate switching time for each instance in the design. ATE uses multi-threaded Verilog processing.

The timing file contains the following information:

- Clock domains and their frequency
- Min/max rise/fall slews for instance pins
- Min/max rise/fall arrival times for instance pins
- Constant instance pins (pins which are not switching)
- Clock domain associated with instance pins
- Clock tree identification

For timing file syntax refer to [section "Timing Data File", page C-750](#).

In static analysis, if power values are imported using the ‘import power’ command, the STA file is not used.

In dynamic analysis, if VCD data is being used and it is in “true time” mode, meaning it contains timing information, timing windows from STA file are not used, since they are derived from VCD. The slew and clock tree information is still derived from the STA file.

If a timing file is *not* provided to RedHawk:

- In static analysis, INPUT_TRANSITION and FREQUENCY GSR keywords are used to get slew and frequency values for all instances in calculating power. If the CLOCK_ROOTS GSR keyword is used, clock tree and instance frequency values are obtained from this, which can affect power calculation and static IR drop results.
- In dynamic analysis, frequency, slew and clock tree information can also be assigned using GSR keywords the same as in static analysis. However, RedHawk cannot determine switching times associated with each instance in the design unless true time VCD is used.

Overview

The data flow in Figure 19-1 shows the input files, analysis steps and corresponding content of the STA file.

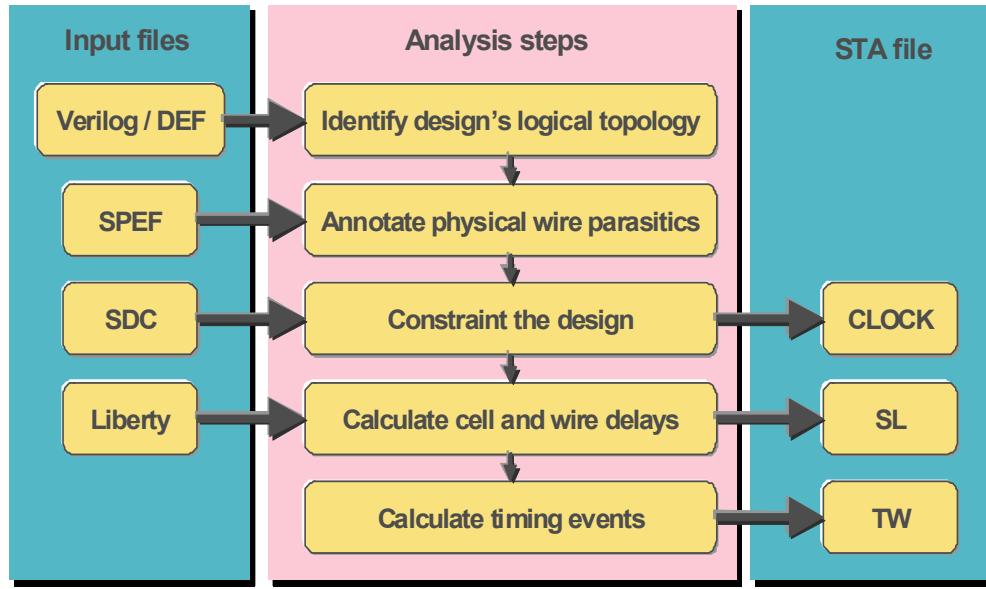


Figure 19-1 ATE data flow

ATE is integrated into RedHawk and can be launched by setting the GSR ENABLE_ATE keyword to 1. The SPEF files are parsed using the ATE engine and annotated to the DEF. Advantage of the ATE spef parser is that it can annotate the SPEF data to instance/nets in the DEF irrespective of hierarchy limitations (that are present in the default RedHawk spef parser) and hence provide the best coverage/annotation for the SPEF input data. The standard output from ATE is in *adsRpt/ate.log*. Other ATE files go to *.apache/ATE/*. ATE-based SPEF import is more accurate in handling various corner cases related to DEF/SPEF hierarchy mismatches. The only additional files needed are SDC files specified using the GSR keyword ATE_CONSTRAINT_FILES.

PJX - Fast Fullchip Clock Jitter Analysis

RedHawk has a fast fullchip flow for Clock Jitter Analysis, with the following features:

- uses RedHawk voltages to derate per instance timing.
- uses SPICE-based APL characterization models to perform cell delay derating.
- uses STA to calculate clock arrival times at each clock pin.
- calculates differences in clock arrival time with N per cycle voltages per instance.
- provides text/GUI trace reports to identify the source of jitter.
- built into the multi-threaded Apache Timing Engine (ATE).
- the TCL command “perform full_chip_jitter” is available after multi-cycle voltage drop analysis.

Setting up ATE

Configuration File

The ATE configuration file contains a list of file pointers to the design and library data. It must be named `<top_design_name>.ate`. ATE automatically finds it in the current directory based on its suffix and derives the top design name from its prefix. The file format is as follows:

```
set synopsys_lib {  
    <list of Liberty files>  
}  
set verilog_netlist {  
    <list of Verilog files>  
}  
set spef {  
    <list of signal SPEF files>  
}  
set timing_constraints {  
    <list of SDC files>  
}
```

ATE also can take a DEF netlist as input instead of using a Verilog netlist. In order to specify DEF files as the design netlist, the 'def_netlist' keyword is used instead of 'verilog_netlist'.

The order of files in the above lists is not important except that the first file in Liberty list is used for obtaining default settings about libraries (for example, reporting time unit, operating condition, and slew thresholds). Therefore typically the main library is listed first.

Furthermore, since the unit conventions are based on the first Liberty file, the unit convention used in it must match the unit convention used in SDC files. For example, if SDC time values are based on 'ns', the first Liberty file should be the one that has 'ns' as its time_unit, or SDC values are incorrectly interpreted. SDC generators typically insert a 'set_units' command into the SDC file to allow the SDC reader to sanity check the convention, but this command does *not* change the unit convention.

After top design name is derived from the configuration file name, the corresponding module is found from the list of netlist files and the rest of the modules are linked to it.

The hierarchy levels of the netlist and SPEF files do not have to be same. ATE's SPEF stitching handles arbitrary combinations.

ATE uses the 'DESIGN' keyword inside SPEF files to find corresponding reference cells in the design, so there is no need to specify a block name for the SPEF files. For example :

```
set synopsys_lib {  
    /path/to/lib/file1.lib  
    /path/to/lib/file2.lib  
    /path/to/lib/file3.lib  
}  
set verilog_netlist {  
    /path/to/verilog/fileA.v  
    /path/to/verilog/fileB.v  
    /path/to/verilog/fileC.v  
    /path/to/verilog/fileD.v  
}  
set spef {  
    /path/to/spef/fileK.spef.gz  
    /path/to/spef/fileL.spef.gz
```

```
    /path/to/spef/fileM.spef.gz  
    /path/to/spef/fileN.spef.gz  
    /path/to/spef/fileO.spef.gz  
}  
set timing_constraints {  
    /path/to/sdc/fileX.sdc  
    /path/to/sdc/fileY.sdc  
}
```

In some cases it may be necessary to override the 'DESIGN' keyword in SPEF files to assign a particular SPEF file to a particular reference cell. To do so, the 'cell_spef' keyword can be used in the configuration file instead of 'spef'. For example:

```
set cell_spef {  
    { cell1_name /path/to/spef/fileK.spef.gz }  
    { cell2_name /path/to/spef/fileL.spef.gz }  
}
```

In some cases it may be necessary to assign a particular SPEF file to a particular instance. To do this the 'inst_spef' keyword can be used in the configuration file instead of 'spef'. For example:

```
set inst_spef {  
    { inst1_name /path/to/spef/fileK.spef.gz }  
    { inst2_name /path/to/spef/fileL.spef.gz }  
}
```

A mixture of SPEF, cell_spef and inst_spef keywords can be used. Precedence is given first to instance level definitions, then to cell level ,and then to the 'DESIGN' keyword in the SPEF file.

The appropriate case analysis and constraints must be set up in the SDC file. Case analysis is required to get the correct operating conditions for clock gating or muxing.

Command File

The following is a typical command file, which in most cases can be used as shown:

```
set errorAction      continue  
  
LoadGeneralParam  
  
DataPreparation     -files all  
LoadLibrary         -error_action $errorAction  
LoadNetlist          -error_action $errorAction  
LoadTimingConstraint -error_action $errorAction  
LoadParasiticFile   -error_action $errorAction -ground_coupled_caps  
  
ta_set_clock_delay -propagated [get_clocks *]  
  
getSTA * -gz
```

Above sequence of commands loads the library and design data, marks all clocks as propagated and then launches 'getSTA', which does analysis and then generates the STA timing file.

Clocks are marked as propagated, since otherwise clock pins at instances receive ideal clocks, which creates inaccurate RedHawk simulations.

PJX flow uses STA voltage as nominal and control on threshold voltage

The RedHawk-PJX flow can control the generation of Timing Delay derating factors (early/late) by a user-specified voltage threshold value, and also by using STA/LIB voltage as nominal in cases of mismatches between STA and GSR voltages. The following keywords are used to enable this:

Threshold value

```
ta_ppx_derate_factors_threshold_voltage <voltage>
```

Specifies the threshold value for creating Timing Delay Derate Factors. The instances that have Effective Voltage (Veff for non min-max mode, and Vmin-Vmax for min max mode) above this threshold do not have Derate Factors in the derate_factors file.

Example:

```
setvar ta_ppx_derate_factors_threshold_voltage 0.95
```

Default: Nominal Voltage is the threshold if no value is provided.

Derate factors file

```
ta_ppx_derate_factors_domain_threshold_file <domain_threshold_file>
```

Specifies a file that contains Voltage Domain Threshold values for PJX to consider when creating Timing Delay Derate Factors. The thresholds can be linked to the VDD domain or VDD-GND pairs. The threshold specified using VDD-GND has a higher priority than thresholds specified using VDD domain.

Example:

```
setvar ta_ppx_derate_factors_domain_threshold_file threshold.txt
```

Example file *threshold.txt* contents:

```
vdd1 gnd1 1.14  
vdd1 0.95
```

STA nominal voltage

```
ta_ppx_use_sta_nominal_voltage [0|1]
```

Specifies PJX to use STA Voltage as Nominal in case of conflict between STA Nominal Voltage (specified through STA environment - .lib/SDC) and RedHawk Nominal Voltage (specified through VDD_NETS keyword). The Timing Delay Derate Factors created are used to scale the instance cell delays from STA Nominal Voltage to the Effective Voltage annotated to the instance after RedHawk Static/Dynamic analysis.

Example:

```
setvar ta_ppx_use_sta_nominal_voltage 1
```

Default: false. Nominal voltage specified in RedHawk environment is considered as reference.

Handling Ideal Clocks

If there are ideal clocks in the design (that is, clock tree synthesis at some clocks is not completed and the root clock driver drives many instances directly), they should not be marked as propagated. Otherwise, since the clock driver has a very large fanout,

RedHawk results would be very pessimistic. To *not propagate* ideal clocks while propagating other, the following can be inserted before the ‘getSTA’ command above:

```
ta_set_clock_delay -propagated [get_clocks *]  
ta_set_clock_delay -ideal [get_clocks clkA]  
set ADS_ALLOW_IDEAL_CLOCKS 1
```

In the example above, all clocks are marked as “propagated” except clkA, which is set as “ideal”.

By default getSTA errors out if there are ideal clocks in the design. Therefore, ‘ADS_ALLOW_IDEAL_CLOCKS’ is set to 1.

If there are ideal clocks, ignoring their drivers is recommended during **RedHawk** analysis by using the IGNORE_INSTANCES keyword. You can also limit the maximum load cap on such driver cells using the GSR keyword setting ‘USE_LIB_MAX_CAP 1’.

Multi-threading

ATE by default runs in multi-threaded mode, utilizing all available processors. If it is necessary to limit the number of processors that ATE should use, the following can be added at the beginning of command file:

```
setvar max_threads <number_of_processors>
```

If the number specified is less than two, ATE runs in single-threaded mode.

Special ATE Variables

Special ATE variables are used in the ATE command file to set parameter values and functionality, and are described in this section. Default values for the variables have been set to give optimal accuracy and performance. If necessary, they can be set before invoking getSTA, using the syntax:

```
set <variable_name> <value>
```

ADS_ALLOW_IDEAL_CLOCKS [0|1]

0 : does not generate a timing file if clocks are ideal. Generating a timing file with ideal clocks yields inaccurate **RedHawk** results, since the registers driven by this clock do not see clock tree latency. Default.

1: the timing file is generated even if clocks are ideal.

ADS_ALLOWED_PCT_OF_NON_CLOCKED_REGISTERS <percent_registers>

If more than the specified percent of registers in the design are not clocked, ATE errors out, since all combinational logic following such registers do not get a TW, and this reduces the accuracy of dynamic analysis. Default: 5 percent.

ADS_CELLS_NEED_INPUT_TW { null | <ref_cell1> <ref_cell2> ... } | ALL

Specifies cells requiring an input timing window. Wild cards are allowed to specify cell names.

{ } : A TW for non-clock input pins is not generated in order to save runtime and disk space, since they are not used by **RedHawk**. However, if the design contains power switches (header and footer switches), **RedHawk** needs a TW for their control pins for ramp-up analysis. Default.

<ref_cell1> ... : a TW is generated for all pins of all instances of specified reference cells. Wildcards using “*” can be used for names.

ALL: a TW is generated for all pins.

ADS_INPUT_PINS_NEED_TW {null | <pin_1> <pin_2> ... }

Specifies input pins requiring a timing window. This variable is mutually exclusive with ADS_PINS_NO_TW. Wild cards are allowed to specify pin names.

{ } : when processing non-clock input pins of cells listed using variable
ADS_CELLS_NEED_INPUT_TW, creates a TW for all input pins of those
cells. Default.

{ pin_1 pin_2 ... } : when processing non-clock input pins of the cells using
ADS_CELLS_NEED_INPUT_TW, creates a TW only specified input pins of
those cells. A typical situation for using this keyword includes handling
memories with internal power switches. Memories often have a large number
of input pins, such as DATA, ADDRESS, and power switch control pin(s).
Wildcards using "*" can be used for names.

ADS_PINS_NO_TW { null | <pin_1> <pin_2> ... }

Specifies pins for which no TW is created. This variable is mutually exclusive with
ADS_INPUT_PINS_NEED_TW.

{ } : when processing non-clock input pins of the cells specified using
ADS_CELLS_NEED_INPUT_TW, creates a TW for all input pins of those
cells. Default.

{ pin_1 pin_2 ... } : when processing non-clock input pins of the cells specified
using ADS_CELLS_NEED_INPUT_TW, do not create a TW for specified
pins of those cells. Note: this variable should not be used to exclude output
pins, since TWs for output pins are required for RedHawk Dynamic Analysis,
except for output pins of power switch cells.

getSTA Command Options

The syntax for the getSTA command in the ATE shell is:

```
getSTA <net_pattern> [<optional_arguments>]
```

where

<net_pattern> is usually set to "*", meaning that timing information should be
generated for all nets.

Timing information for specific net patterns can be generated using wildcard characters,
such as:

```
getSTA block1/*
```

Optional arguments are as follows:

- output <file> : timing file name (default: <top_design_name>.timing)
- block : print timing window for primary IO ports
- compact : generate the file in sta-compact format (default)
- nocompact : generate the file in legacy format
- gz : generate the file in gzip format

Using sta-compact format (default) is recommended to save disk space, to create a
significantly smaller timing file without loss of information, and with negligible impact on
ATE runtime, as well as saving runtime when RedHawk loads the timing file later on. See
[section "Timing Data File", page C-750](#), for the syntax of the timing file. Using the '-gz'
gzip option further reduces file size. There is no runtime penalty for this unless multi-
threading has been disabled.

If ATE is run at block level, as opposed to the top level, the '-block' option should be used,
which generates timing windows for primary input/output ports also.

Invoking ATE

The environment variable APACHEROOT should be defined before running ATE. For example:

```
setenv APACHEROOT /install_dir/apacheda/<RedHawk_release>
set path = ($APACHEROOT/bin $path)
```

Then invoke:

```
ate ate.cmd >& ate.log
```

ATE Command Line Options

The following options are available on the ATE command line:

-lmwait : waits for license, if not available

-v : displays ATE version

Output Files

ATE creates the following files and directories:

<top_design_name>.timing : timing file

ads_non_clocked_registers.rpt : list of unclocked registers

.ate/ : directory for temporary internal files

In multi-threaded runs, ATE stores the output of the LoadParasiticFile stage in the *./LoadParasiticFile.log* file, to avoid overwriting its output.

<top_design_name>.html : provides high-level run statistics, such as tabular runtime/memory breakdown for each task and subtask of the session, version, and host. This should be the first file to look at when evaluating ATE performance. An example of one of the tables is given in Figure 19-2.

Task	Run Time					Memory In Use				Wire Length
	Wall Time	Self User CPU Time	Self System CPU Time	Child Process User CPU Time	Child Process System CPU Time	Self Total Memory	Self Physical Memory	Child Process Total Memory	Child Process Physical Memory	
LoadGeneralParam	0:0:0	0:0:0	0:0:0	0:0:0	0:0:0	124.82M	12.79M			
DataPreparation	0:0:26	0:0:0	0:0:0	0:0:23	0:0:0	125.36M	13.33M	732.95M	402.52M	
LoadLibrary	0:0:0	0:0:0	0:0:0	0:0:0	0:0:0	202.57M	90.81M			
LoadNetlist	0:0:4	0:0:3	0:0:0	0:0:0	0:0:0	402.62M	274.48M			
LoadParasiticFile	0:0:24	0:0:24	0:0:0	0:0:0	0:0:0	899.31M	745.89M			
LoadTimingConstraint	0:0:6	0:0:5	0:0:0	0:0:0	0:0:0	1255.55M	1097.30M			
getSTA	0:0:59	0:1:0	0:0:0	0:0:0	0:0:0	1509.62M	1345.14M			
Entire Run	0:1:59	0:1:32	0:0:0	0:0:23	0:0:0	1509.62M	1345.14M	732.95M	402.52M	

Figure 19-2 ATE performance table

Specifying the STA file in RedHawk

The timing file can be specified in RedHawk using the STA_FILE GSR keyword:

```
STA_FILE {  
    <top_design_name> <file>  
}
```

See section "STA_FILES", page C-604, for more details. RedHawk then imports the file during the 'setup design' stage.

Alternatively, the timing file can be imported using the 'import sta' TCL command, which allows for incremental analysis using different timing files. For example:

```
import gsr design.gsr  
setup design  
perform pwrcalc  
perform extraction -power -ground -c  
perform dynamic  
import sta new.timing  
perform dynamic
```

ATE Validation

Ensuring Correct Creation and Use of the Timing File

In order to effectively review timing file creation and use, consider the data flow in Figure 19-3, which shows the input files, related ATE tasks for STA analysis steps, and the corresponding content of the STA file.

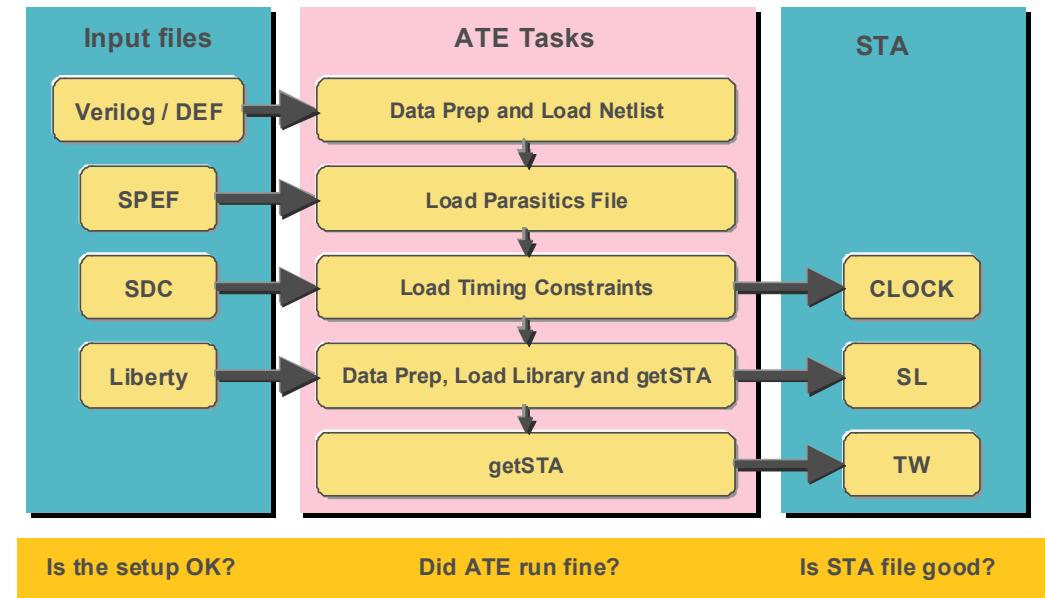


Figure 19-3 ATE data flow and tasks

Do a quick check of tasks using this flow, so that basic setup issues, such as using the wrong files or missing clock definitions, can be resolved quickly, without having to backtrack after generating incorrect RedHawk results.

In addition, this flow can be used to reduce possible problems, without spending time with unrelated input files or run logs. For example, if there are issues with CLOCK entries in the STA file, the first place to look is the SDC file, and the messages coming from the Load Timing Constraints task. Then you can systematically trace back to previous tasks in the flow and the relevant input files.

Key checks of results are summarized in the following sections.

Is the setup OK?

1. Confirm that all the files specified in configuration file exist. Any missing files are reported.
2. Confirm that there is no behavioral Verilog file. STA is based on gate level Verilog files, and behavioral Verilog files trigger syntax errors.
3. Confirm that there is not a significant number of cells with missing Liberty and/or Verilog module definitions. ATE converts such cells into a black-box to keep going and reports the following message for each cell with missing data:

VLG-0027: Creating black box for ...

There should be no such message for cells that are required for RedHawk analysis.

4. Confirm that input files are consistent (that is, from the same version of design database):
 - Verilog/DEF vs. SDC
 - Verilog/DEF vs. SPEF
 - Verilog/DEF given to ATE vs. DEF provided to RedHawk
 - Liberty, SPEF given to ATE vs. those provided to RedHawk
5. Confirm that SDC files have proper TCL syntax. For example, the files should not refer to undefined variables, and they should not contain commands/variables specific to third-party tools. Standard TCL, SDC and collection commands are supported.
6. Confirm that the SDC file contains all clock definitions in the 'create_clock' or 'create_generated_clock' statements.
7. Confirm that if an SDC command refers to a library (the '-library' option is used), that library is among the Liberty files given to ATE.
8. Confirm that the library cells referred in SDC file (for example, at set_driving_cell command) exist in the Liberty files given to ATE.
9. Confirm that the unit convention used in the first Liberty file matches the convention used in the SDC file. ATE displays a SHL-0661 message if the SDC file used the set_units command and a discrepancy is detected.
10. Confirm that there are no unintentional ideal clocks. If there are legitimate ideal clocks in the design, see the section "Handling Ideal Clocks", page 19-518. ATE displays a SHL-0628 message in the getSTA task for these cases.

Did ATE run properly?

1. Review error messages starting from the first task. In most cases, resolving a prior error resolves some of the subsequent ones also.
2. Check basic design statistics reported in the Load Netlist task, as in the following example:

```
Info: SHL-0581: Number of instances : 133523773
Info: SHL-0581: Number of nets       : 136465158
```

```
Info: SHL-0581: Number of terminals : 1378
```

3. Confirm that SPEF has been annotated for the majority of nets in the design, by reviewing the “RC annotation” table in the file *LoadParasiticFile.log*.
4. Confirm that all SDC constraints that are required for STA file creation have been recognized. Review errors in the Load Timing Constraint task. If some clocks have not been created, see the SHL-0631 message in the getSTA task to check whether the number of unclocked registers is negligible.
5. Review the HTML file to see if any subtask is causing a runtime/memory bottleneck. If there is, review the UTL-0031 and UTL-0032 messages coming from that task to identify the bottleneck.

Is the STA file good?

1. Confirm that a majority of registers are connected to a clock. See the SHL-0631 messages in the getSTA task.
2. Confirm that all clocks have been captured in the CLOCK section of the timing file, with the correct parameters.
3. Confirm that timing window entries have been created for a majority of instances by reviewing TIMING_WINDOW_OK metrics at the end of the file, as shown in the following example:

```
# ##### SUMMARY #####
# Total processed pins : 510831
# -
# Pin statistics:
#   TIMING_WINDOW_OK      :      504915  (99%)
#   NO_TIMING_WINDOW      :       5278  (1%)
#   CONST                  :        638  (0%)
```

For example, for an instance pin that has no timing window, there may be an entry as follows:

```
#BlockA/InstB/Y NO_TW { }
```

which could be caused by several things, including:

- A flip-flop is not connected to a clock. All logic starting from this FF will not have a TW.
- A black box in the timing path stops propagation of the TW. All instances after this black box will not have a TW.
- Incorrect case analysis settings, leading to broken timing arcs in clock muxes.

4. Check statistics reported by RedHawk, as in the following example:

```
Instances (CONST/assigned/Total) 1340/30667/36232
Nets (CONST/assigned/Total) 1675/35447/38885
Pin slew (missing/assigned) 26893/105005
Instance input/clock slew missing 2564 (use GSR default
input_transition)
Instance output slew missing 2850 (use average input slew)
Total number of instances traced (CONST/STA/tracer) =
32007/0 (1340/30667/0)
Total number of nets traced (CONST/STA/tracer) = 37122/0
(1675/35447/0)
Total number of leaf clock nets = 394
```

```
Total number of CLK pins traced as clock = 6630 / 6635
Active/Quiet Summary:
  Instance Summary:
    Total = 36232
    clock = 859 (active 859)(quiet 0)
    non-clock = 35373 (active 29808)(quiet 5565)
  Net Summary:
    Total = 38885
    clock = 867 (active 867)(quiet 0)
    non-clock = 38018 (active 34580)(quiet 3438)
```

5. Check the files created by RedHawk in the *adsRpt* directory after STA import.

apache.staBogus: list of instances in the STA file that cannot be mapped into the design (DEF netlist)

apache.tw0: list of instances in the design not covered in the STA file.

apache.twclk0: list of instances with clock pins not covered in the STA file.

apache.twclkLate: list of instances for which the clock arrives later than the output signal.

apache.clkPin0: list of instances for which the CLK pin is connected to a non-clock net.

Contacting Apache Support

If you need Apache AE Support, the following information is needed:

- command file
- configuration file
- standard output and *LoadParasiticFile.log*

If a crash has occurred, also send 'trace*' files generated in the run directory. If possible, also send the design data used for the run.

Chapter 20

Chip-Package Analysis (CPA)

Introduction

RedHawk-CPA (or “CPA”) provides high-resolution (per-bump level) package PDN extraction and co-simulation capability. The package model is in RLCK form, and is convenient for system-level chip+package co-simulation. In addition to model extraction, detailed color maps of bump level resistance and inductance are available, superimposed on the physical layout. The quick turn-around time of extraction, coupled with bump level parasitic data makes this an ideal tool for “what-if” analysis of packages. The CPA model can be readily imported into **RedHawk** for system-level high-resolution analysis.

An overview of the RedHawk-CPA analysis flow is shown in Figure 20-1.

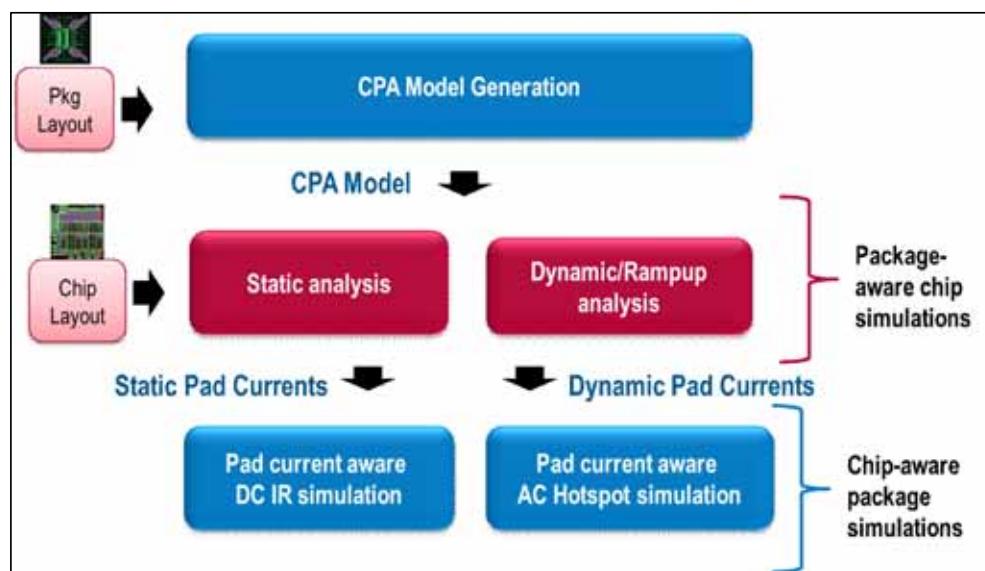


Figure 20-1 Overview of CPA model generation in RedHawk

Key features of CPA analysis are:

- Fast, high resolution (per-bump level) RLCK extraction, customized for **RedHawk**
- Native creation of annotated ploc and REDHAWK_PKG subckt wrapper with “one click” model import into **RedHawk**
- Seamless import of the CPA model into **RedHawk** to facilitate chip-package co-simulation
- Automatic chip-to-package connection and grouping based on pin location matching, with powerful GUI with “Use Pin Names” feature.
- Voltage Regulator Module (VRM) voltage can be specified, or imported from the GSR file
- Leverages **RedHawk** matrix solver
- Static Pad current-aware DC IR simulation of package
- Dynamic Pad current-aware AC Hotspot simulation of package

- 3D Fast Finite Element Method (FFEM) engine
- Aids in package design and optimization
- Includes package decaps
- Large package fast turn-around time
- “Impedance Analysis” feature
- Output netlist size reduction
- Comprehensive HTML reports with summary
- Enhanced Pad Voltage and Current maps and histograms
- High resolution Resistance and Inductance maps at per-bump resolution
- Other reporting features:
 - Sort by voltage in DC IR summary
 - Reporting unconnected pins of packages displayed in HTML
 - Advanced wirebond component handling

Co-simulation Flows

Two types of Chip-Package co-simulation analysis flows are supported in RedHawk:

- DC IR
- AC hotspot

Both flows require package extraction to be performed as the first step. Once the CPA extracted model database is available the co-simulation flow can be launched, as described in the following sections.

Integrated Chip Package Analysis and Chip Thermal Modeling

In the RedHawk GUI, thermal analysis shares the same geometry editing UI as CPA, which helps users perform thermal analysis more efficiently. CPA users can set up and perform thermal analysis using the same data folder and files for signal integrity analysis. An example GUI interface displays are shown in Figure 20-2 and Figure 20-3.

- The integrated GUI enables package layout import and CPA model generation using the RedHawk GUI.
- The same GUI is used to view package layout, pin resistance, and pin inductance maps, along with chip views.
- Automatic hook-up to chip layout is provided, maintaining pin-to-pin mapping.
- The integrated chip and package GUI is activated upon detection of the GSR keyword CPA_FILES.
- Legacy flows based on the CPA_MODEL keyword and separate chip/package GUIs are also supported
- For compatibility, a CPA model generated by standalone RedHawk-CPA can be used in the integrated flow. If the standalone RedHawk-CPA project is located at <old_project_path>, specify the CPA_FILES keyword as:

```
CPA_FILES {
    PACKAGE  <old_project_path>/Result/adsCPA/dB/
    layout_filename.xfl
    MODEL    <old_project_path>/Result/adsCPA
}
```

- All standalone RedHawk flows (without the CPA_FILES keyword) display the regular

GUI (without the package views enabled).

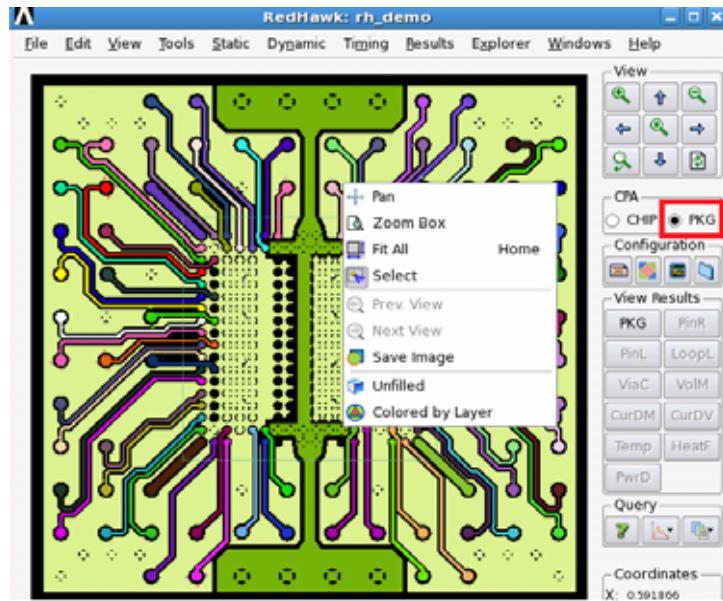


Figure 20-2 Integrated CPA CTM GUI interface

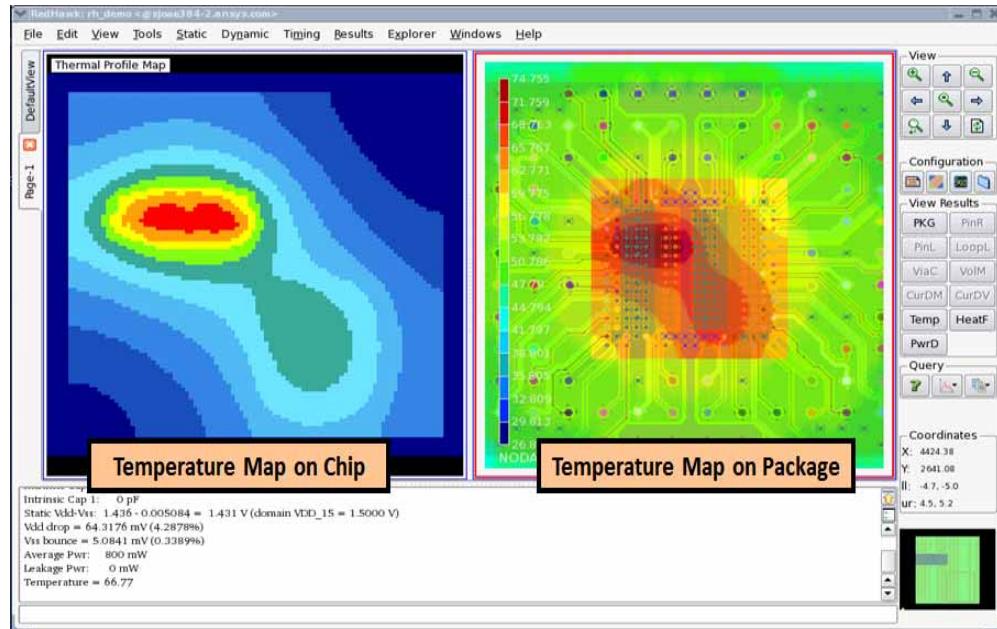


Figure 20-3 Chip and package temperature maps

The integrated thermal capability can perform the following thermal analyses:

- Automatic thermal model generation and analysis based on exact metal distributions in BGA substrates.
- Interface with realistic temperature-dependent layer-aware power map on chips (CTM)

from RedHawk, and back-annotation for Electro Migration (EM) reliability on chip.

- Provide pre-layout package modeling and analysis
- Include GSR keyword setting on “Options” GUI pages for easy mesh and performance controls.
- The display of text in color maps is provided with the option **Text Color** that allows users to specify a proper color for texts on the color maps,

CTM Viewer

CPA thermal analysis uses an enhanced CTM Viewer that includes the following features:

- Added protection when importing a legacy .ctm that has the previous v0 format not supported by CTM_Visualizer;
- When an invalid .ctm file is imported, the message “*Please import a valid Layer-aware CTM file!*” is displayed.
- Ability to launch the CTM Viewer when the *ctm* in .tar or .tar.gz format is readable.
- Cleans up terminal messages when clicking the CTM Viewer.

Temperature-Aware CTM

The Multi-Heat Source Editor dialog supports direct adoption of .ctm for thermal analysis using a GUI option **Enable T-aware CTM**, as shown in Figure 20-4. **Enable T-aware CTM** check box is optional. It is by default auto-selected after clicking the **Save T-aware CTM** button. Also, the file path of the exported .ctm displays in the text box below the **Enable T-aware CTM** check box. Checking this option allows you to use the .ctm power for thermal simulation. Otherwise, the multi-heat source power is used.

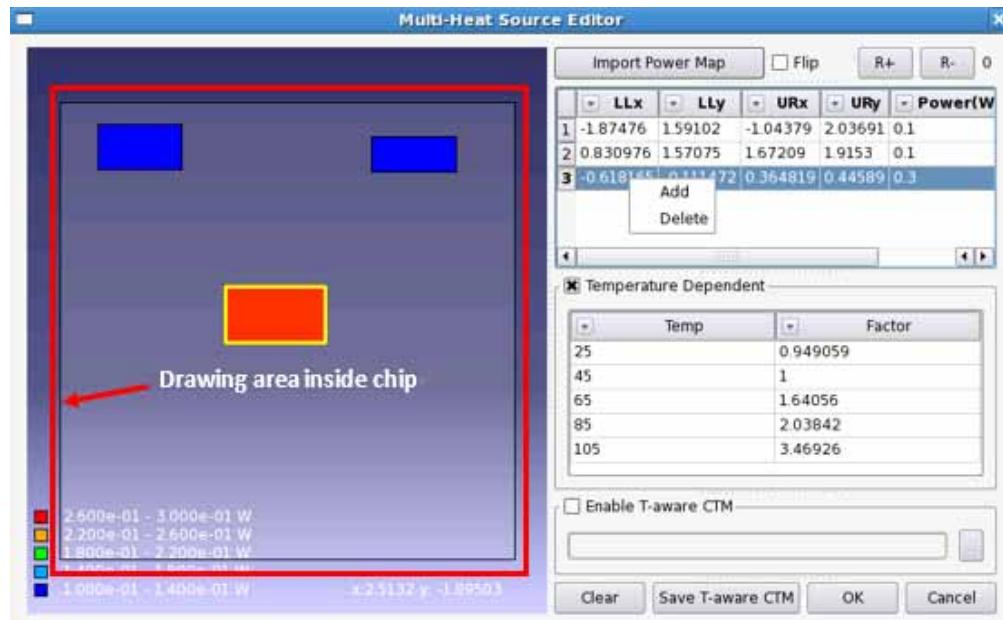


Figure 20-4 Multi-Heat source editor

The exported *ctm* file can be imported back to integrated thermal by **Die tab->CTM**. Note that the *.ctm* cannot be viewed by **CTM Viewer**. The **Clear** button deletes all blocks with power assignments in the block table.

Configuration for Thermal Analysis

The GSR keyword setting 'THERMAL_ANALYSIS 1' turns on thermal analysis.

For backward compatibility, the thermal model generated by standalone Sentinel-TI can be used in the CPA flow. If the standalone Sentinel-TI project is located at <old_proj_path>, define the **CPA_FILES** keyword as:

```
CPA_FILES {
    PACKAGE <old_proj_path>/layout_filename.xfl
    MODEL <old_proj_path>
}
```

Then rename <old_proj_path>/Results_TI to <old_project_path>/thermal.

Configuration and Results

You must set up the package and run thermal analysis using the **Setup->Thermal Setup** and **Do Extraction->Thermal-Sim** buttons.

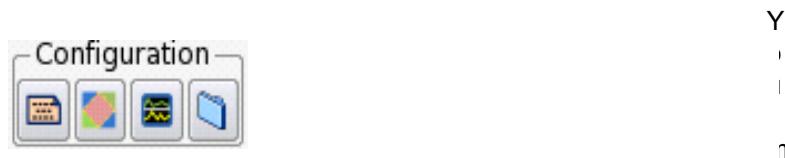


Figure 20-5 Thermal configuration buttons

Setup->Thermal Setup:

This menu command brings up the **Thermal Setup** dialog that has six tabs on the right side: **PKG Config**, **Die**, **Material**, **Boundary**, **Heat Sink** and **Simulation**. In the pop-up window you can set up PKG Config, assign CTM to Die, or deploy constant power or multiple heat, cast external conditions, enable heat sink, and set up simulation options for thermal simulation.

Do Extraction->Thermal-Sim

This menu command performs electrical DRC checks first. If any violations are found, a DRC report is prepared before continuing to extraction. Thermal simulation users can ignore the electrical DRC reports to continue thermal simulation. When simulation finishes, three color maps -- nodal temperatures, heat flux and power density -- are generated in the <project_path>/thermal file.

View Result buttons

After thermal simulation, the **Temp**, **HeatF**, **PwrD** View Result buttons can be used to view the thermal results graphically:

- **Temp**: displays temperature contour per metal layer.
- **HeatF**: displays heat flux vector plots
- **PwrD**: displays power density contour on each layer of die.

TCL commands for thermal simulation are not available within **RedHawk**

Visible option for Vias, Bumps and Balls

The Property page for Via, Solder Bump and Solder Ball managers has a **Visible** option to enable and disable visibility of vias, bumps and balls in the GUI.

Zoom to a pin

The RedHawk CPA GUI can display a **PKG/Die Pad Info** report window, as shown in Figure 20-6, where you can view ploc connected pkg pin properties, including pkg net and pkg pin (x, y) coordinates. Right click on a pin and select **Zoom to this pin** in the context menu. The layout view highlights the selected pin in the layout view.

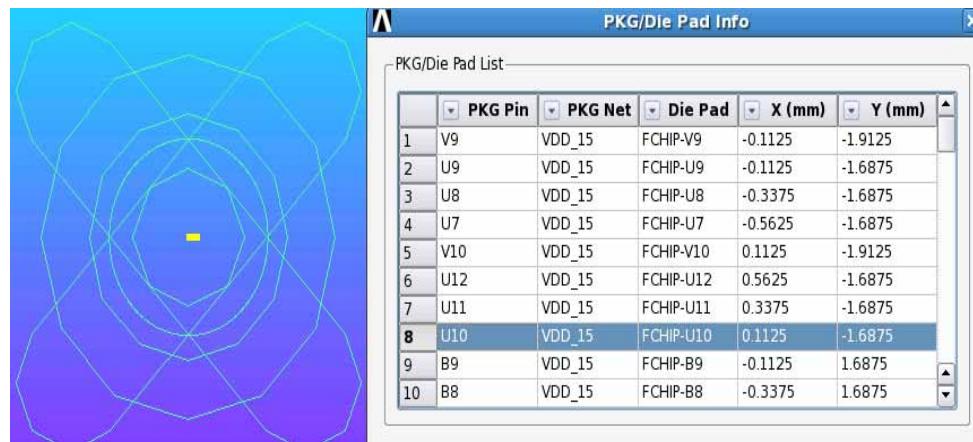


Figure 20-6 Pkg/Die Pad Info dialog

Integrated Chip and Package GUI

The integrated GUI displays chip and package layout and map views together, with the following features:

- The integrated GUI enables package layout import and CPA model generation using the **RedHawk** GUI.
- The same GUI is used to view package layout, pin resistance, and pin inductance maps, along with chip views.
- Automatic hook-up to chip layout is provided, maintaining pin-to-pin mapping.
- The integrated chip and package GUI is activated upon detection of the GSR keyword **CPA_FILES**.
- Legacy flows based on the **CPA_MODEL** keyword and separate chip/package GUIs are also supported
- For compatibility, a CPA model generated by standalone **RedHawk-CPA** can be used in the integrated flow. If the standalone **RedHawk-CPA** project is located at <old_project_path>, specify the **CPA_FILES** keyword as:

```
CPA_FILES {
    PACKAGE  <old_project_path>/Result/adsCPA/dB/
    layout_filename.xfl
    MODEL    <old_project_path>/Result/adsCPA
}
```

- All standalone **RedHawk** flows (without the **CPA_FILES** keyword) display the regular GUI (without the package views enabled).

Chip and Pkg Auto-Connection and Pin Grouping

RedHawk-CPA can automatically connect ploc to package using the pin names, using the dialog shown in Figure 20-7. This is recommended, particularly when the package and ploc have the same pin names.

The default pin grouping is limited to 4000 for packages having >4K P/G bumps. You can override the default grouping in the “Die Model Connection” window.

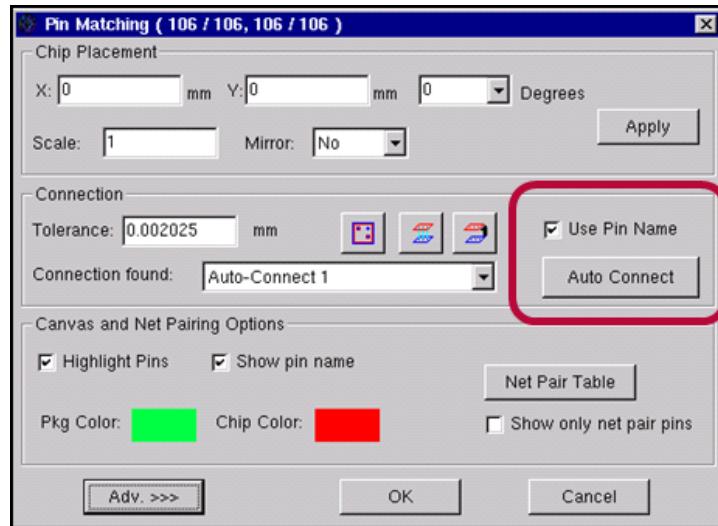


Figure 20-7 Pin matching dialog in CPA

DC IR Co-simulation

The following TCL command is used to launch CPA from inside RedHawk:

```
perform cpa -static
```

The following command is used to launch CPA from the UNIX command line:

```
rhcpa <toplevel_CPA_folder> <static_pad_current_file> -static
```

where

<toplevel_CPA_folder> : specifies the folder containing the CPA extracted database. The default folder name is *adsCPA*.

<static_pad_current_file> : specifies the static pad current filename generated by RedHawk.

-static : launches DC IR co-simulation

The above commands load the dB and launch the RedHawk-CPA dialog, as shown in Figure 20-8.

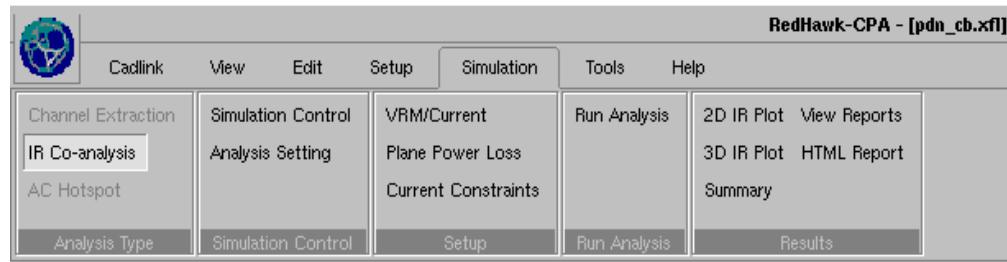


Figure 20-8 CPA dialog

Select the “IR Co-analysis” button and the “VRM Current” button.

The VRM and Die current dialog is displayed, already populated with the supply voltage and the die pad static currents, as shown in Figure 20-9.

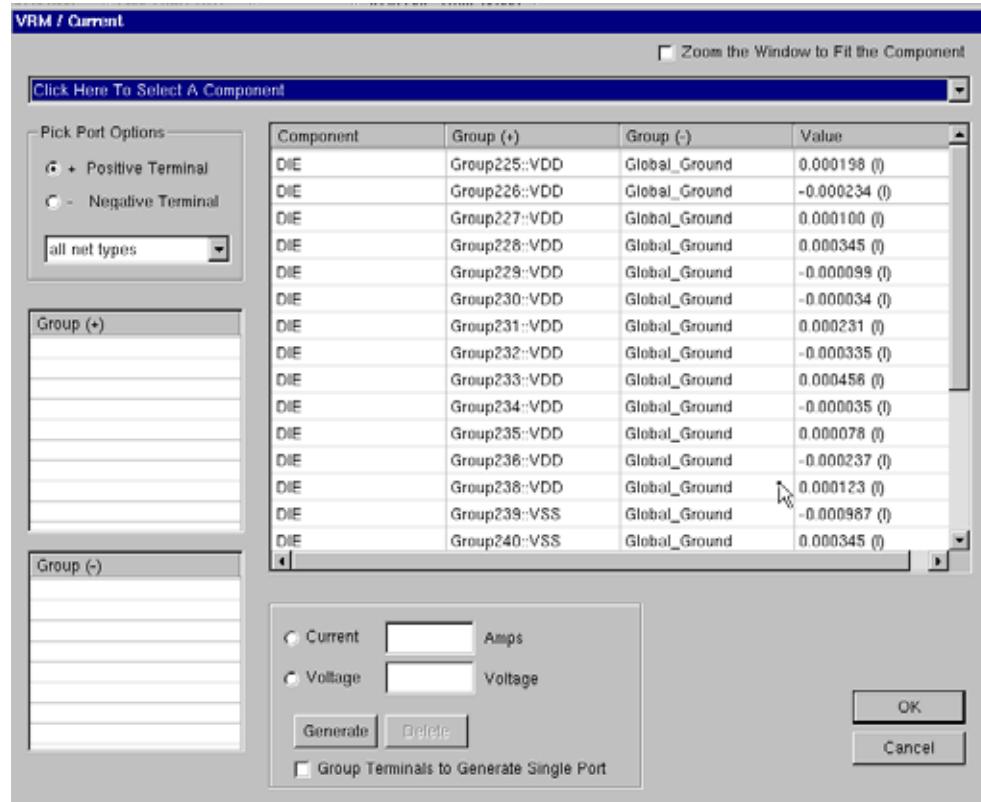


Figure 20-9 VRM and die current dialog

Click on “Run Analysis” in the CPA dialog to perform IR analysis of the package. Once the simulation is completed, package IR drop maps, a summary of the results, and HTML-based reports can be selected in the dialog and viewed.

AC Hotspot Co-simulation

The following command can be used to launch the CPA from inside RedHawk:

```
perform cpa -hotspot
```

The following command is used to launch the CPA from a UNIX command line:

```
rhcpa <toplevel_CPA_folder> <dynamic_pad_current_file> -hotspot
```

where

<toplevel_CPA_folder> : specifies the folder containing the CPA extracted database. The default folder name is *adsCPA*.

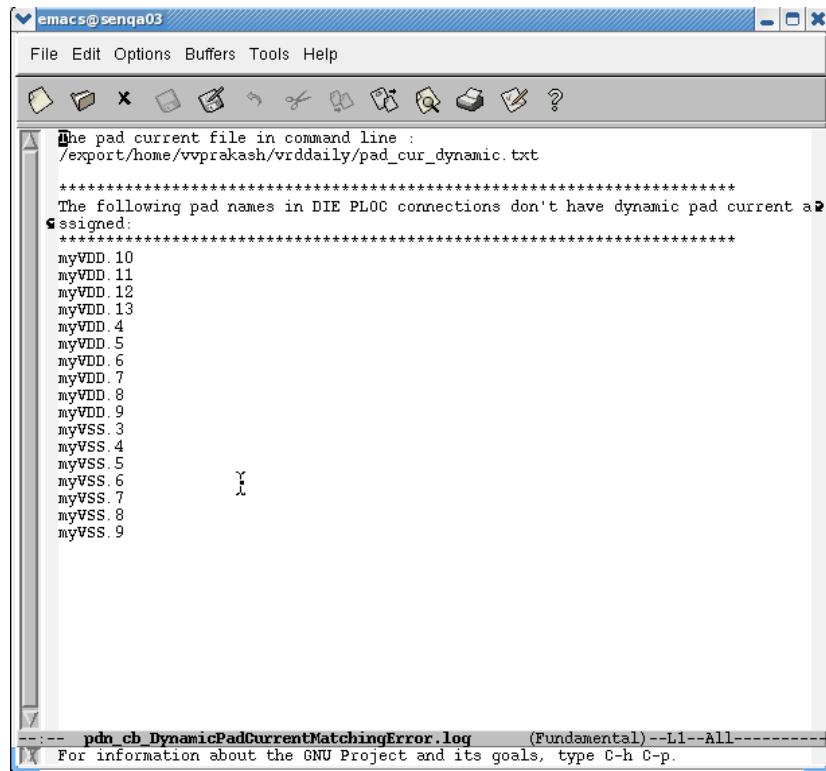
<dynamic_pad_current_file> : the dynamic pad current file generated by RedHawk.

-hotspot : the command to launch AC hotspot co-simulation

The above commands load the dB and launch the RedHawk-CPA dialog.

Select the “AC Hotspot” button in the CPA dialog.

A list of all the package pins that do not have dynamic pad currents is displayed in a text file, as shown in Figure 20-10.



The screenshot shows a terminal window titled "emacs@senqa03" with a menu bar: File, Edit, Options, Buffers, Tools, Help. Below the menu is a toolbar with various icons. The main window contains a text buffer with the following content:

```
The pad current file in command line :  
/export/home/vvprakash/vrddaily/pad_cur_dynamic.txt  
*****  
The following pad names in DIE PL0C connections don't have dynamic pad current assigned:  
*****  
myVDD.10  
myVDD.11  
myVDD.12  
myVDD.13  
myVDD.4  
myVDD.5  
myVDD.6  
myVDD.7  
myVDD.8  
myVDD.9  
myVSS.3  
myVSS.4  
myVSS.5  
myVSS.6  
myVSS.7  
myVSS.8  
myVSS.9
```

At the bottom of the terminal window, there is a status bar with the text "pdn_cb_DynamicPadCurrentMatchingError.log (Fundamental) --L1--All---" and "For information about the GNU Project and its goals, type C-h C-p."

Figure 20-10 CPA pad pins list

Impedance Analysis simulates package layout for the user-specified frequency range. You can define ports on the die side and view the impedance response of the package from the defined die ports. Impedance analysis can be run after channel extraction in RedHawk-CPA. You should specify the desired die-side ports and frequency range prior to running the hot spot analysis, as shown in Figure 20-11.

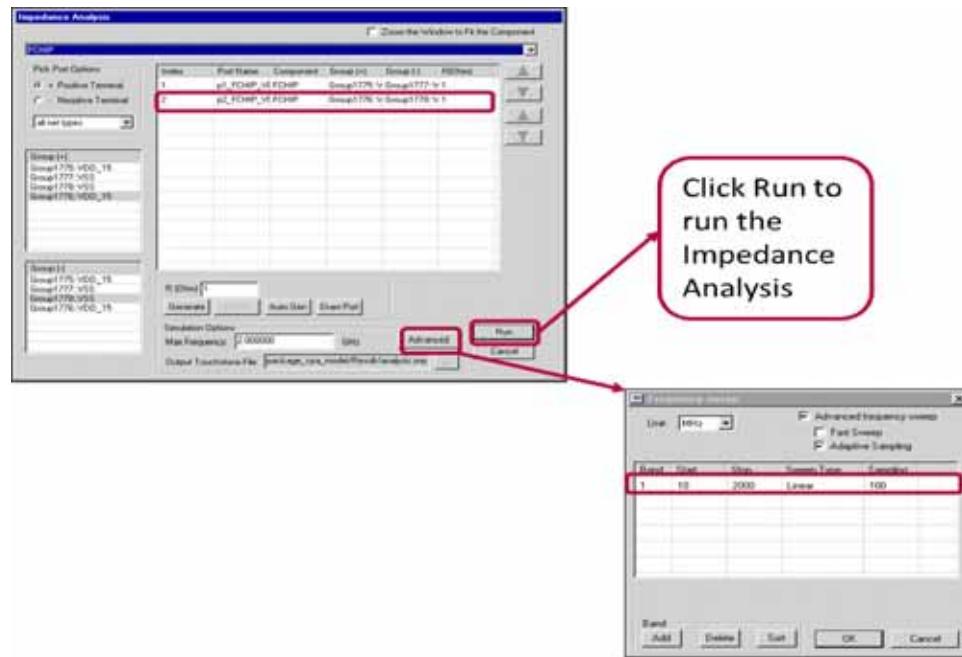


Figure 20-11 Impedance and frequency sweep dialogs

Probing the voltage and current waveforms at the component pins, and the plane voltages can be done through “Observation Setup” in the CPA dialog. Click on the “pin Voltage” button and the pin voltages are displayed in the dialog, as shown in Figure 20-12.

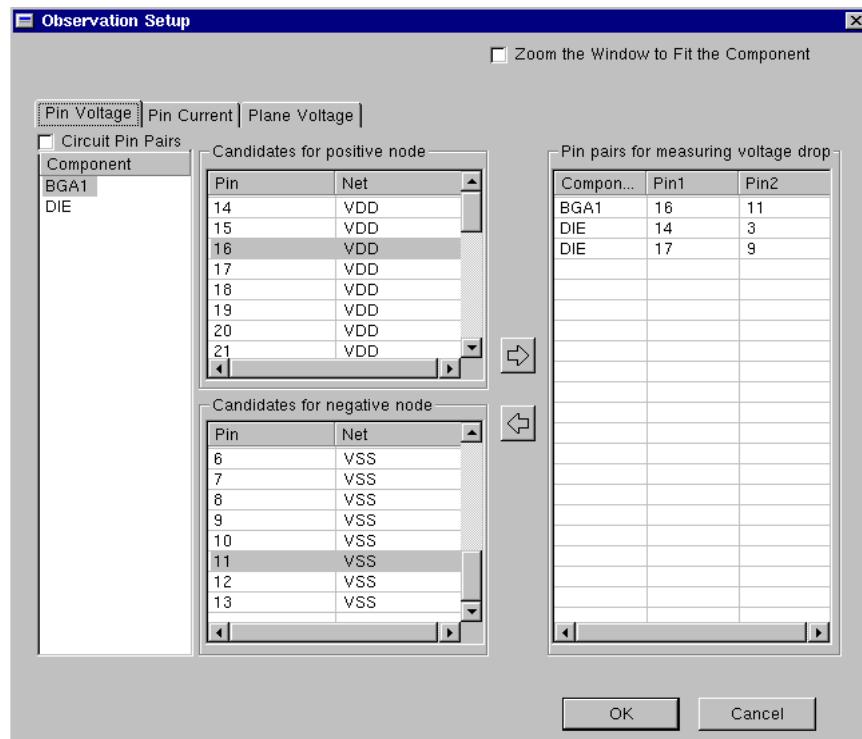


Figure 20-12 CPA Observation Setup - Pin voltage

Click on the “Pin Current” button and the pin currents are displayed in the dialog, as shown in Figure 20-13.

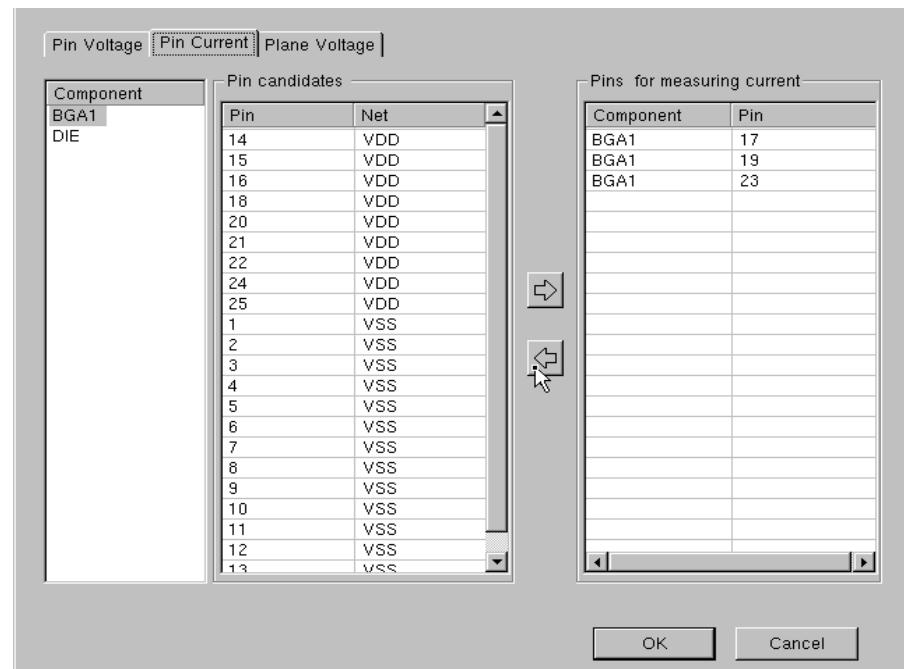


Figure 20-13 CPA Observation Setup - pin current

You can view the impedance curves using the built-in S-utility, as shown in Figure 20-14.

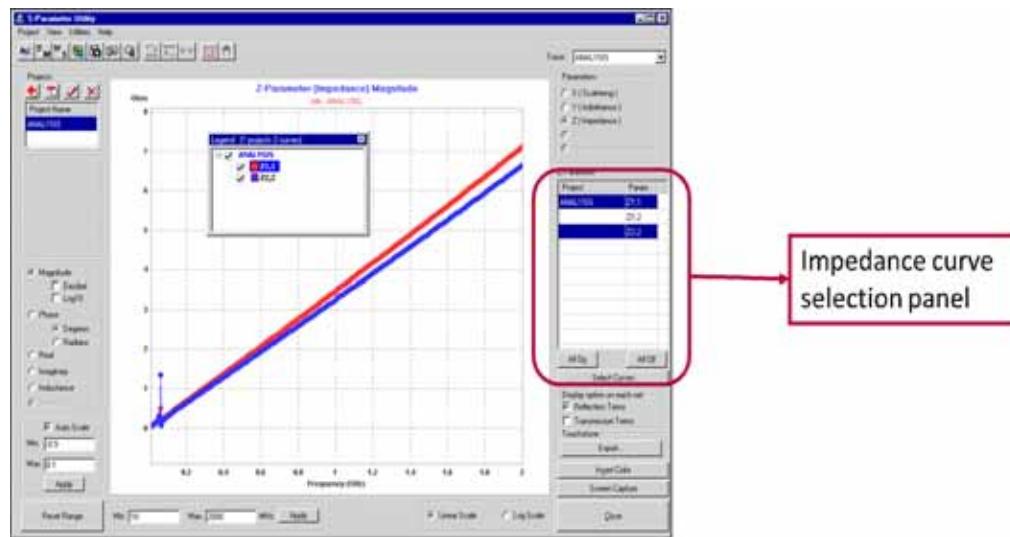


Figure 20-14 Impedance plots using S-utility

Click on “Run Analysis” in the CPA dialog to perform AC hotspot analysis of the package. Once the simulation is completed, the package voltage and plane current maps, probing point voltage and current waveforms can be selected and viewed, in addition to HTML reports. The plane voltage dialog is shown in Figure 20-15.

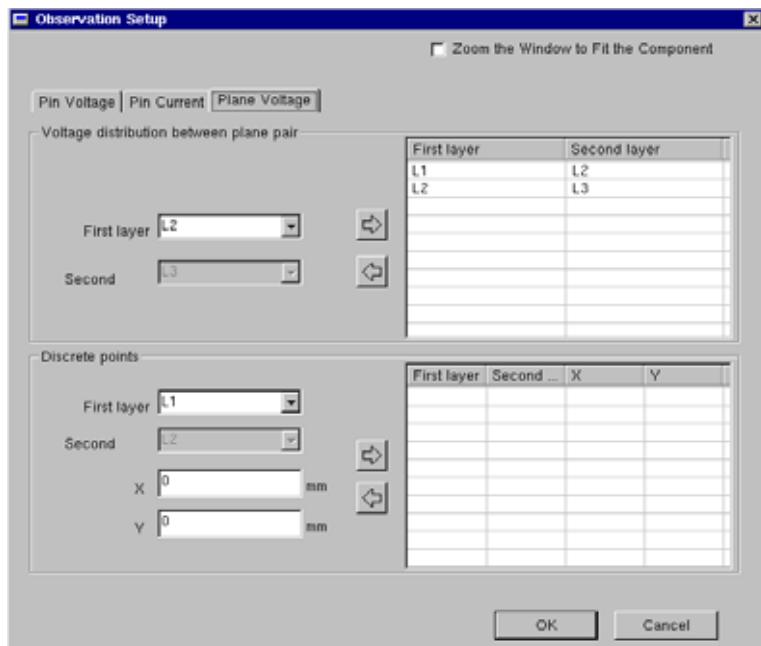


Figure 20-15 CPA Observation Setup - Plane voltage

HTML-based Reporting

Comprehensive HTML-based reports are available for channel extraction, IR drop analysis and AC hotspot analysis flows. Key features:

- The DC IR summary text report is sorted by voltage.
- Unconnected pins on the package are reported in the file *Result/adsCPA/Extraction/Die_Pkg_Connection_Info.txt*.
- Package decap values are directly reported in Channel Extraction report under the “Component Spice Models” section.
- Either a Firefox or Conqueror browser is required to open the reports. When the “HTML Report” button is clicked, the location of the HTML file is displayed in the bottom message window so that you can access the file and open it directly.
- Note that the “Images” sub-folder, which is found in the same location as the HTML file, is required for distribution.

The output reports include the following.

- General project information
- Tool version and build data
- Machine details
- Geometry data for each layer
- Nets for analysis
- Electrical setup
- Simulation results

Appendix A

Installation Procedure

Introduction

This chapter describes how to download and install the **RedHawk** program, in the following steps:

1. Download software from the **Apache** FTP site
2. Perform the installation
3. Set up the **RedHawk** software license
4. Set up the **RedHawk** environment

Downloading RedHawk Software

The current FTP address, filename, and password can be obtained from your **RedHawk** technical support contact.

```
ftp ftp.apache-da.com
name (ftp.apache-da.com:user): anonymous
password: <email address>
ftp> bin
ftp> passive (some sites may require this)
ftp> cd releases/RedHawk/<version>
ftp> get <filename>
ftp> quit
```

You may also log in at the Apache Customer Support Center website at:

<http://www.apache-da.com/apache-da/Home/CustomerSupportCenter.html>
for download instructions.

Program Installation

Download the software in the *<tarball_directory>*, such as */disk1/ecad/*, from where you want to install **RedHawk**. Follow the instructions below, using the *csh/tcsh* shell (or other shells).

```
% cd <tarball_directory>
% gtar xzvf RedHawk_<platform>_<version>.tar.gz
% setenv APACHEROOT <RedHawk Installation Directory>
```

This will allow execution of the installed and associated **RedHawk** binaries.

After installing the **RedHawk** software, the following directory structure will be created in the *<RedHawk_installation_directory>*, along with the software Release Notes.

<u>Directory</u>	<u>Partial Contents</u>
<i>license/</i>	Apache license file
<i>bin/</i>	Contains all the RedHawk related executables, such as “flow setup” utilities <i>rh_setup.pl</i> and <i>gds_setup.pl</i> .

<i>lib/</i>	Machine dependent libraries
<i>lmbin/</i>	FlexLM manager and utilities
<i>platform/</i>	LSF support for APL and PsiWinder
<i>scripts/</i>	Useful Perl scripts for assisting in RedHawk execution.
<i>scripts/atcl</i>	Helpful TCL utilities for TCL query capabilities in RedHawk
<i>doc/</i>	Software Release Notes and latest <i>RedHawk Users' Manual</i>

RedHawk Operating System/Platform Support

RedHawk supports RHEL (RedHat)/CentOS versions 5 and 6 and SLES (SUSE) version 11.

Setting Up the Apache License

RedHawk uses the industry-standard FlexLM licensing scheme. The license file (*apacheda.lic*) can be obtained from your Apache technical support contact. Write the license file into the *license/* directory. It is recommended that your system administrator installs the license and executables and makes a backup copy of the release directory and license.

To start the license daemon, issue the following commands on the license server:

```
% cd /disk1/ecad/apache (as the <RedHawk Installation Dir>)
% lmbin/<platform>/lmgrd -c license/apacheda.lic
    -l /var/tmp/apacheda.log
```

where

lmgrd : the FlexLM license daemon program,
apacheda.lic : the license file, and
apacheda.log : the license log file.

Note that the log file is put under the */var/tmp* directory. The system will automatically remove the log file when it reboots. If you want to maintain the license log file even after the system is shutdown and reboots, you can specify another directory other than */var/tmp* (for example, use */tmp*). However, you must ensure that you have write permission to that directory when you start the license daemon process.

NOTE: Do not start *lmgrd* as root.

You can now check the status of the license daemon by using the following commands,

```
% lmbin/<platform>/lmstat -v apacheda
% ps -al | grep apacheda
```

If you need to stop the license daemon for any reason, use following command:

```
% lmbin/<platform>/lmdown -c license/apacheda.lic
```

where *apacheda.lic* is the license file.

NOTE: Do not use the Unix command `% kill -9`, as this may leave the vendor daemon in the background and cause problems.

Setting Up the RedHawk Environment

License File and Library Directory Setup

The license file setup is required for every user who runs **RedHawk**. The lines below may be added to the `.cshrc` file. Set up the license file in the `csh/tcsh` environment:

```
% setenv APACHEDA_LICENSE_FILE <absolute_path_to_license-file>  
or,  
% setenv APACHEDA_LICENSE_FILE port_number@lic_server_ip_addr  
% set path=($path <abs_path_to_release>/RedHawk)
```

For example,

```
% setenv APACHEDA_LICENSE_FILE /disk1/ecad/apache/license/apacheda.lic  
or through network for the license file,  
% setenv APACHEDA_LICENSE_FILE 1881@129.186.1.10  
% set path=($path $APACHEROOT/bin)  
% rehash
```

If the library directories are not in the same location as **RedHawk**, set the following environment variables in the `.cshrc` shell environment:

```
setenv LD_LIBRARY_PATH <RedHawk_installation_dir/lib>  
or if $LD_LIBRARY_PATH already exists, set up the following instead:  
setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH:<RedHawk_installation_dir/  
lib>"
```

```
setenv SHLIB_PATH <RedHawk_installation_dir/lib>  
or if $SHLIB_PATH already exists, set up the following instead:  
setenv $SHLIB_PATH "$SHLIB_PATH:<RedHawk_installation_dir/lib>"
```

If 'source ./setup.ksh' was not executed in the ksh shell environment, then set the following environment variables in the `.kshrc` shell environment.

```
set LD_LIBRARY_PATH=<RedHawk_installation_dir/lib>  
or if $LD_LIBRARY_PATH already exists, execute the following instead:  
set LD_LIBRARY_PATH="$LD_LIBRARY_PATH:<RedHawk_installation_dir/lib>"  
export LD_LIBRARY_PATH  
set SHLIB_PATH=<RedHawk_installation_dir/lib>  
or if $SHLIB_PATH already exists, execute the following instead:  
set SHLIB_PATH="$SHLIB_PATH:<RedHawk_installation_dir/lib>"
```

Binary Setup

There are two **RedHawk** binary options:

- platform-specific
- platform-independent

These two types of invocations are discussed in the following sections.

Platform-Specific Binaries

Available platform-specific binaries for **RedHawk** are “Linux32” and “Linux64”, which can be run on RedHat and SLES (SUSE) platforms. These binaries are for customers with only one or two types of machines that want platform-specific binaries for reasons of disk space or FTP download time.

Before using **RedHawk**, source `.cshrc` or refresh `.kshrc`. Alternatively, if ‘source `./setup.csh`’ was not executed in the `csh` or `tcsh` shell environment, then set the following environment variables in the `.cshrc` shell environment:

```
setenv APACHEROOT <RedHawk_installation_dir>
set PATH=($APACHEROOT/bin $path) [ if desired ]
```

If ‘source `./setup.ksh`’ was not executed in the `ksh` shell environment, then set the following environment variables in the `.kshrc` shell environment.

```
set APACHEROOT=<RedHawk_installation_dir>
export APACHEROOT
```

Platform-Independent Binaries

The platform-independent binary under the “Suites” package includes all **RedHawk**-supported platforms, with transparent access without regard to machine. Utilities APL and GDS2DEF/GDSMMX under the “Suites” package also support 64-bit Solaris version 8.x.

If you want to use a relative invocation path, a PATH variable must be defined for the desired binary location, but the `$APACHEROOT` variable is not required, and will be ignored if incorrect. The correct binary installation for the machine is found on **RedHawk** invocation.

Invocation

In order to test the installation, issue the following command in the application directory:

```
redhawk
```

The **RedHawk** GUI should appear, as shown in Figure A-1 below. The data files can now be loaded and **RedHawk** run.

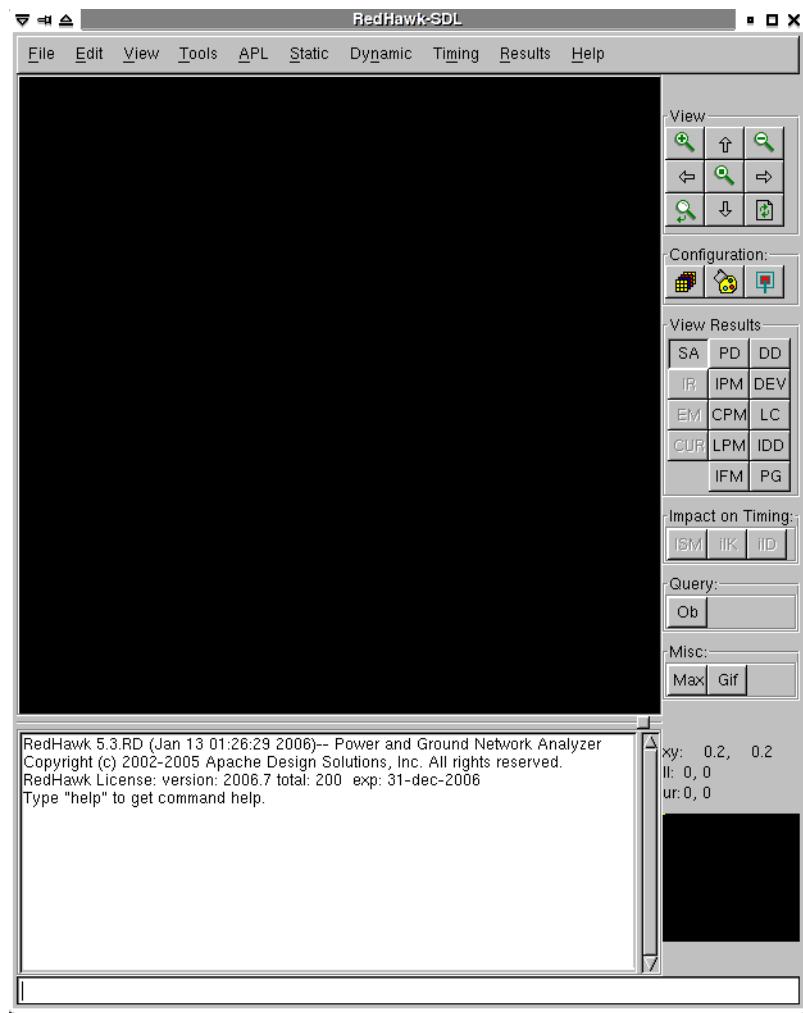


Figure A-1 Initial GUI of the Apache RedHawk program

Appendix C

File Definitions

Introduction

This appendix describes the format and contents of internal input and output files supported by RedHawk.

RedHawk Input Files

1. Technology file (*.tech)
 2. Global Switching Configuration file (*.gsc)
 3. Global System Requirements file (*.gsr)
 4. Pad instance, cell, or location files (*.pad, *.pcell, *.ploc)
- The information from the following files is required. The file reference information should be included in the GSR file.
5. Library technology files
 6. Design netlist files
 7. Synopsys library files

RedHawk input files can be prefixed by the design name. The symbol "#" preceding a line is treated as a comment for all these files.

NOTE: RedHawk can read LEF, DEF, STA, SPEF, VCD, FSDB and GDS files in compressed *.gz (gzip) format.

RedHawk Output Files

Results are generated in each phase of the analysis process. See details of report files and graphic displays available in the individual subject chapters and in [Chapter 6, "Reports"](#), for more details.

Keyword Syntax Conventions

The syntax conventions used for defining RedHawk commands, options and keywords are as follows:

<x>	x describes a variable or value to be specified
[a b c]	one of a, b, or c must be selected
? x ?	x is an optional parameter
a b c d	a, b, c, and d parameters all must be specified
<x> ...	elements of the same type as <x> can be added
{ {a b c ...} {j k l ...} ... }	similar sets of elements may be added. Note that if only one set of elements is included, two sets of brackets may still be required
+ - * /	standard arithmetic operators for add, subtract, multiply, divide

Apache Technology File (*.tech)

The RedHawk technology file (.tech) provides technology information about the process and specifies parameters for each metal layer and type of via. A separate technology file is needed for each IC process. The following technology information is specified:

- conductor name of each metal layer from the .lef file
- thickness, resistivity, resistance temperature coefficient, and EM limit for each metal layer
- via name, resistance per via, and via EM limit
- resistance, inductance, and capacitance of the wire-bond or flip-chip solder bumps
- thickness, height, and dielectric constant of dielectric layer
- thickness and resistivity of the substrate layers

When specifying values in the .tech file, RedHawk supports the following units and prefixes. Unit is case sensitive.

Unit prefix conventions:

- terra = t = 1e+12
- giga = g = 1e+9
- mega = M or me = 1e+6
- kilo = k = 1e+3
- milli = m = 1e-3
- micro = u = 1e-6
- nano = n = 1e-9
- pico = p = 1e-12
- femto = f = 1e-15

Example: 1.3p = 1.3e-12. There is no space between the number and unit symbol.

Unit length conversions

- 1 mil = 0.001 inch = 2.54e-5 meters
- 1 inch = 2.54e-2 meters
- 1 micron = 1.0e-6 meters

Encrypting and Decrypting a Tech File

Full File Encryption

Two utility programs in the bin directory allow a RedHawk ASCII Tech file to be encrypted using a password, and RedHawk can then run the file in encrypted form. As needed, the encrypted file then can be decrypted with the correct password.

1. To encrypt a tech file, on a UNIX command line run the command:

```
techEncrypt <tech_filename>
```
2. Provide a password at the prompt.
3. An encrypted file, <tech_filename>.enc.tech, is created.
4. The encrypted tech file now can be used in RedHawk the same as a non-encrypted tech file.

5. To decrypt a tech file, on a UNIX command line run the command:

```
techDecrypt -i <encrypted_tech_file> -o <tech_file>
```
6. Provide the correct password at the prompt.
7. A decrypted ASCII file, *<tech_file>*, is created.

Partial File Encryption

RedHawk supports encryption/decryption of specified sections of the tech file, so that sensitive process information can be hidden from end users. The main RC tech file has two sections; the first section has metal/vias/dielectric information that is not encrypted and is visible in plain text. The next section has entries for some metals/vias/dielectrics that is encrypted. This section may only contain the values related to RC, heights, or dielectric constants for the same metals/dielectric that is specified in unencrypted section. The 'start' and 'end' encryption lines are comment lines, so if there are "#ENCRYPT_START" and "#ENCRYPT_END" lines in a tech file, the designated section between the lines is automatically encrypted as specified.

Technology File Keywords

NOTE: Keyword and option names are case insensitive.

DIELECTRIC

Defines the dielectric layer parameters. Values for all layers must be defined when capacitance and inductance extraction is performed. By default, dimensions are in microns. *Optional; default: None*

Syntax:

```
DIELECTRIC <dielectric_layer_name> {
    constant <ER_value>
    thickness <value>
    [ Height <value> |
      Above <dielectric_layer_name> ]
    ER_VS_SI_SPACING {
        (<spacing_1>, <ER1>)
        ...
        (<spacing_n>, <ERn>)
    }
}
```

where

dielectric <dielectric_layer_name> : specifies name of dielectric layer
constant <ER_value> : specifies the value of dielectric constant (relative permittivity for the layer, except as otherwise defined (no units)).
thickness <value> : specifies the thickness of the dielectric layer
Height <value> : specifies the height above defined base layer (0)
Above <dielectric_layer_name> : specifies that the layer is just above the defined layer

ER_VS_SI_SPACING: specifies the relative permittivity of dielectric layers between conductors as a function of spacing. Note that you must create the tech file from the ITF file using rhtech, and then run the 'captab' utility to update the tech file:

```
captab <tech_file> -layer all -all
```

Also note that the spacing specified is the on-silicon value.

Example:

```
dielectric IMD3b {
    thickness 0.22000
    constant 2.9
    ER_VS_SI_SPACING {
        { 0.2 2.8 }
        { 0.21 2.9 }
        { 0.23 2.99 }
        { 0.25 2.6 }
        { 0.26 3.0 }
    }
    above IMD3a
}
```

EM_PEAK_EQUATION_SOURCE_TECHFILE

Allows foundries to build custom Duty Ratio factors directly into the Peak EM limit equations, and bypass the automatic Duty Ratio factor ($1/\sqrt{r}$) calculation to apply to each base Peak EM limit.

EM_RULE_SET

Allows EM_MODE to specify different EM rule sets in the same tech file for CD checking, which allows you to define EM rule names for different corners in same tech file. Or, for example, EM_MODE can use different EM limits in static analysis, whereas earlier only 'EM_MODE avg' was supported for static analysis. And EM rules can be grouped together using EM_RULE_SET. The keyword can be applied to any of the EM analysis modes (AVG, RMS, PEAK). A sample rule set syntax is:

```
EM_RULE_SET <user_rule_name1>
    metal <layer_name> {
        <metal EM rules >
        ...
    }
    ...

    EM_RULE_SET <user_rule_name2>
        metal <layer_name> {
            <metal EM rules >
            ...
        }
        ...
    ...
}
```

You can select which of the defined rule sets to use for each EM analysis, using GSR keyword settings, such as:

```
EM_RULE_SET_AVG typical_avg
EM_RULE_SET_PEAK worst_peak
EM_RULE_SET_RMS worst_rms
```

EM_TECH_FILE

Specifies the name of a file that can define EM rules for different corners, or EM_MODE can use different EM limits in static analysis, in same tech file. The EM_TECH_FILE keyword can contain any number of rule sets, separated by a header line that gives the rule set name. Note that mode names, such as

typical_avg and worst_peak, should be consistent with the names specified in the EM tech file, and be unique names.

Syntax

```
EM_TECH_FILE <EM rule set file>
```

HALF_NODE_SCALE_FACTOR

Specifies the scaling factor for RC extraction parameters and geometry, which is taken by the *rhtech* utility from the same keyword in the *itf/.nxtgrd* file used by the foundry, or the *ircx2tech* utility and the *iRCX* file can be used. This scaling affects extracted RC parameter values and all parameters in the *itf/nxtgrd* file as if dimensions were reduced by the specified factor, but it does not scale parameter values in the Tech file, or actual dimensions in the RedHawk layout view, or GUI. All geometry reporting (location, metal width/length) keeps the original dimensions so that you can fix/cross probe from the original database.

When the tool calculates the width of a piece of metal, it takes the drawn width and multiplies it with the HALF_NODE_SCALE_FACTOR value, such as 0.9. For EM, the EM limits in the tech file are for post-shrink values.

The associated APL model scaling is performed in the BSIM model using an equivalent keyword item ‘.option geoshrink=<factor>’. As an alternative, you can use GSR keywords DEF_SCALING_FACTOR and LEF_SCALING_FACTOR, for which the layout dimensions *are* scaled. *Optional; Default: no scaling.*

Syntax:

```
HALF_NODE_SCALE_FACTOR <factor>
```

Example:

```
HALF_NODE_SCALE_FACTOR 0.8
```

METAL

Defines the physical characteristics of each metal layer, and the effective size of the wires in them. Resistivity must be defined, using either a constant or a process parameter-based calculation method. If more than one value is defined, the more accurate process-based values are used. RedHawk records which method is used in the session log file. Many of the process parameter-based keywords and values are taken from standard technology files and are not incorporated or edited directly.

An important parameter value for determining EM is the *wire width* to be used. There are two width definitions for wires, *drawn width*, W(D), and *silicon width*, W(S). If the option WIDTH_SI_VS_WIDTH_AND_SPACING is defined, then:

W(S) = w (looked up from the table using W(D)) – EM_ADJUST.

Otherwise, if options RESISTIVE_ONLYETCH or ETCH_VS_WIDTH_AND_SPACING are defined, then

W(S) = W(D) – etchL – etchR – EM_ADJUST,

where ‘etchL’ and ‘etchR’ are the left (down) and right (up) side etching effects calculated from the etch table for the wire. If none of above options are defined, then ‘W(S) = W(D) – EM_ADJUST’, where the EM_ADJUST is the keyword for each layer.

For half-node scaling, when the tool calculates the width of a piece of metal, it takes the drawn width and multiplies it with the HALF_NODE_SCALE_FACTOR, such as 0.9. For EM, the EM limits in the tech file are for post-shrink values.

There are also two widths used in EM calculation, ‘w(lookup)’ is the width used to look up the value in the EM conditional rule table, and ‘w’ is the width used to determine current density to get EM current. There are two GSR keywords to

control the type of width used, as shown in the following table. Note that the default is foundry-dependent.

Defining Wire Width for EM Calculation	USE_DRAWN_WIDTH_FOR_EM_LOOKUP = 1	USE_DRAWN_WIDTH_FOR_EM_LOOKUP = 0
USE_DRAWN_WIDTH_FOR_EM = 1	W(lookup) = W(D) W = W(D)	N/A
USE_DRAWN_WIDTH_FOR_EM = 0 (default)	W(lookup) = W(D) W = W(S)	W(lookup) = W(S) W = W(S)

Required for all metal layers, except as noted.

Note: The first part of this definition includes basic metal specifications, including the effective wire width to be used in EM calculations. See the following sections for advanced metal resistance and capacitance specification methods based on process parameter relationships for: a. temperature, b. thickness, c. width and spacing, d. etch geometry.

Syntax:

```

METAL <metal_layer_name> {
    ? MINWIDTH <drawn_wire_width_um> ?
    ? MINSPACE <wire_spacing_um> ?
    WIDTH_ADJUST <adjustment>
        ? WIDTH-SPACE {
            {<w1> <spacing1>}
            {<w2> <spacing2>}
            ...
        }?
        Pitch <value of length>
        Thickness <t_value>
        Resistance <rpsq>
        RHO <value_Ohm-microns>
        CAP_DENSITY <value_fF/um^2> <associated_layer>
        EM <max_wire_current_density>
        ? WIDTH_SI_VS_WIDTH_AND_SPACING ?
        ? T_EM <final_temp> ?
        TNOM_EM <nom_temp>
        EM_TEMP_RATING {
            { <temp1> <derating_factor1> }
            { <temp2> <derating_factor2> }
            ...
        }
    }
    ? EM_adjust <wire_width_adj_um> ?
    ? EM_thickAdjust <wire_thickness_adj_um>?
    ? WIDTH_BASED_EM {
        width { <silicon_width1> ... <silicon_widthN> }
        em {<limit_per_u-sil_width1> ... <limit_per_u-sil_widthN+1>}
        }?
    ? BLECH_JLC <Blech_product>
        [ Height <length-value> |
        Above <dielectric_layer_name_below_metal> ]?
}
```

```

? EM_TIME_CURRENT {
    { t1 Ipeak_1 }
    ...
    { tn Ipeak_n }
} ?
POLYNOMIAL_BASED_EM_[DC | PEAK | RMS ] {
    COND_RULE {<cond_rule1> ... ? AND (UPSTREAM)? }
    EM_POLYNOMIAL { <polynomial expression> }
    COND_RULE {<cond_ruleA> ... ? AND (DOWNSTREAM)? }
    EM_POLYNOMIAL { <polynomial expression> }
    EM_POLYNOMIAL { ANTI_DERATE (polynom expression)
        IMAX ( DERATE ( MIN ((polynom expression )))) }
    ...
}

```

Example:

```

POLYNOMIAL_BASED_EM_DC {
    COND_RULE { L>10 AND w>0.18 AND (UPSTREAM) }
    EM_POLYNOMIAL { 0.02*w IMAX sqrt(3*(w -0.01)^2)
    COND_RULE { L>10 AND w>0.18 AND (DOWNSTREAM) }
    EM_POLYNOMIAL { 0.04*w IMAX sqrt(4*(w -0.01)^2)
}
POLYNOMIAL_BASED_EM_PEAK {
    COND_RULE { W/L > 2 AND W/L < 10 }
    EM_POLYNOMIAL { 10 * ( w - 0.1 ) }
}
POLYNOMIAL_BASED_EM_RMS {
    COND_RULE { L <= 10 }
    EM_POLYNOMIAL { sqrt( 1 * delta_T *( w - 0.1 ) ^ 2 )
}
COND_RULE { L >= 20.0 AND w > 0.18 }
EM_POLYNOMIAL { 1.3*w IMAX DERATE(min((w* 43.6/1),
( 0.06*l/25 ))
EM_POLYNOMIAL { ANTI_DERATE ( 0.01*w ) IMAX (
DERATE ( MIN (( 0.001 + 0.00223086*w ),
( 0.032/w )) )
}
}

```

where

MINWIDTH <drawn_wire width_um> / MINSPACE <wire_spacing_um> :
 specifies the minimum wire width and minimum spacing between wires for DRC, based on the WMIN and SMIN values in the ITF file, respectively. Note that the LEF WIDTH parameter is used first for specifying metal wire width when creating DEF net wires.

WIDTH_ADJUST <adjustment>: specifies an adjustment to width spec such that effective wire width for analysis = original width minus <adjustment>

WIDTH-SPACE <w1> <spacing1> : specifies pairs of drawn wire widths and associated spacing values for wires present in the design.

Pitch <distance> : specifies the distance between wire centers for the grid (um)

Thickness <t_value> : specifies metal layer thickness (um)

Resistance <rpsq> : specifies the metal sheet resistance, in Ohms per square.
 Effective resistance is calculated as $rpsq * \text{length} / \text{width_si}$, where length and width_si are length and silicon width of the metal segment, respectively.

RHO <value_Ohm-microns> : specifies the global value of resistivity for wires

CAP_DENSITY: user-specified capacitance density (fF/um²) between the given layer and the specified <associated_layer> (instead of extracting cap value)

EM <max_wire_current_density> : max current density allowable for acceptable EM conditions, in current per wire width (units specified in 'Units' section) (default: 1.0)

WIDTH_SI_VS_WIDTH_AND_SPACING: specifies W(S) based on width and spacing data (taken from appropriate vendor data)

T_EM <final_temp> : specifies final operating temperature used for EM calculations (degrees C).

TNOM_EM <nom_temp>: nominal temperature for EM calculations (degrees C)

EM_TEMP_RATING <temp1> <derating_factor1> : specifies the ratio of maximum current density between <temp1> and TNOM_EM. So the maximum current density limit at <temp1> can be calculated by multiplying the <derating_factor1> by TNOM_EM.

EM_TEMP_DERATE : When set to 1 within POLYNOMIAL_BASED_EM_RMS or POLYNOMIAL_BASED_EM_PEAK, user can specify separate EM_TEMP_RATING for RMS and Peak limits within the individual rule section.

EM_Adjust : adjustment to wire width.
 If specified, calculated current density = true current in wire / (drawn width of wire - EM_Adjust)

EM_thickAdjust : adjustment to actual wire thickness for Blech JLC length calculation

WIDTH_BASED_EM {...} : specifies EM limit values per micron for metal segments based on their *silicon* width (by table lookup). So for width <silicon_width1>, use EM value <limit_per_u-sil_width1>, up to <silicon_widthN>, use EM value <limit_per_u-sil_widthN>, and for widths greater than <silicon_widthN>, use EM value <limit_per_u-sil_widthN+1>. If specified, EM_ADJUST values are applied to width values. No interpolation or extrapolation is performed.

BLECH_JLC <Blech_product> : specifies a maximum product (mA/um) of worst case wire length and current density for groups of connected and active segments on a layer. This feature is turned on by setting the ENABLE_BLECH GSR keyword to 1. The EM limit calculation is:
 $\text{EM limit} = \text{Blech_JLC} * (\text{wire_thickness}/\text{length}) * \text{width}$

Height <value> : specifies the height above defined base metal layer (0)

Above <dielectric_layer_name_below_metal> : specifies that metal layer is just above defined dielectric layer. The bottom of the metal is coincident with the top of the dielectric.

EM_TIME_CURRENT : allows consideration of current peak width when performing EM analysis, to avoid unwanted EM warnings for small current peaks. The specified tn value is the pulse width in ns, and the associated Ipeak_n value is the peak EM current density in mA/um. Warning "AlertN" messages are issued when the specified peak levels are exceeded. The GSR keyword 'EM_MODE' must be set to "peak" to use this feature. Note that specification of EM limits must be ordered such that width tn+1 is larger than width tn.

POLYNOMIAL_BASED_EM_* : defines tables of relationships for DC (or AVG), RMS, and PEAK maximum EM current, based on specified values for LENGTH_RANGES and WIDTH_RANGES. EM_POLYNOMIAL(S) entries can define an equation for maximum EM current as a function of metal width and length for each combination of width range and length range. If WIDTH_RANGES is specified before LENGTH_RANGES, then WIDTH_RANGES is used as the primary table index. Units: mA.
 If there is only one polynomial that is applied to all lengths for a condition, the range can be represented by any very large number.

COND_RULE <>: defines a conditional rule limit, such as “L >= <length> AND w <= <width>”. This option can be defined using equations.

UPSTREAM/DOWNSTREAM : optionally defines conditional rules that are different for opposite current flow directions, that is, via electron flow toward higher layers (UPSTREAM) and toward lower layers (DOWNSTREAM)

IMAX <lmax polyn> entry : specifies an additional polynomial for the maximum allowed current (lmax) in the polynomial-based EM equation, which can be applied selectively for a desired length/width range. Current computed from the regular polynomial is checked against lmax, and limited to the lmax value if exceeded. Temperature derating is applied only to regular polynomials and not to lmax.

ANTI_DERATE: turns off default temperature and EM_MISSION_PROFILE derating for specified parts of the polynomial. Should not be applied to IMAX polynomials.

DERATE : the IMAX clause defines the absolute maximum limit, so no derating factors are applied to it. DERATE is used in the IMAX equation to specify that the enclosed value or equation needs to have temperature and EM_MISSION_PROFILE derating factors applied to it. Should not be applied to non-IMAX polynomials.

Note: Max peak current used for EM check is the value obtained from above current rules, divided by sqrt(r), where r is the duty cycle. For PG EM, r = 1.

Example 1:

```
METAL metal7
{
    THICKNESS    0.7
    RESISTANCE   0.025
    EM           2.4
    EM_ADJUST    0.02
    BLECH_JLC   250
    ABOVE        diel
}
```

Example 2:

```
...
    POLYNOMIAL_BASED_EM_RMS {
        LENGTH_RANGES { 1e38 }
        EM_POLYNOMIAL {sqrt(15.00*delta_T*(w - 0.015)^2*
                           (w - 0.015 + 0.155)/(w - 0.015 + 0.055)) }
    }
```

where the ‘delta_T’ parameter is defined by the GSR keyword DELTA_T_RMS_EM.

Example 3:- Using EM_POLYNOMIAL format

```
...
```

```

POLYNOMIAL_BASED_EM_DC {
    LENGTH_RANGES { 5.00 20.00 }
    WIDTH_RANGES { 0.05 0.15 }

    # For L<5
    EM_POLYNOMIAL { 4*0.388*( w -0.016) }
    EM_POLYNOMIAL { 4*0.388*( w -0.016) }
    EM_POLYNOMIAL { 4*1.388*( w -0.016) }
    EM_POLYNOMIAL { 4*1.388*( w -0.016) }
    EM_POLYNOMIAL { 4*2.388*( w -0.016) }

    # For L=5
    EM_POLYNOMIAL { 4*0.388*( w -0.016) }
    EM_POLYNOMIAL { 4*0.388*( w -0.016) }
    EM_POLYNOMIAL { 4*1.388*( w -0.016) }
    EM_POLYNOMIAL { 4*1.388*( w -0.016) }
    EM_POLYNOMIAL { 4*2.388*( w -0.016) }

    # For 5<L<20
    EM_POLYNOMIAL { (20/L)*0.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*0.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*1.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*1.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*2.388*( w -0.016) }

    # For L=20
    EM_POLYNOMIAL { (20/L)*0.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*0.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*1.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*1.388*( w -0.016) }
    EM_POLYNOMIAL { (20/L)*2.388*( w -0.016) }

    # For L>20
    EM_POLYNOMIAL { 0.388*( w -0.016) }
    EM_POLYNOMIAL { 0.388*( w -0.016) }
    EM_POLYNOMIAL { 1.388*( w -0.016) }
    EM_POLYNOMIAL { 1.388*( w -0.016) }
    EM_POLYNOMIAL { 2.388*( w -0.016) }

}

```

Example 4 - Using EM_POLYNOMIALS format:

```

...
POLYNOMIAL_BASED_EM_DC {
    LENGTH_RANGES { 5.00 20.00 }
    WIDTH_RANGES { 0.05 0.15 }

    # For L<5
    EM_POLYNOMIALS {
        { 4*0.388*( w -0.016) }
        { 4*0.388*( w -0.016) }
        { 4*1.388*( w -0.016) }
        { 4*1.388*( w -0.016) }
        { 4*2.388*( w -0.016) }
    }

    # For L=5
    EM_POLYNOMIALS {
        { 4*0.388*( w -0.016) }
        { 4*0.388*( w -0.016) }
        { 4*1.388*( w -0.016) }
    }
}

```

```

{ 4*1.388*( w -0.016 ) }
{ 4*2.388*( w -0.016 ) }
}
...

```

Process parameter-based resistance-capacitance modeling

Users that have statistical process variation data in StarRC-XT, *.itf and *.nxtgrd tech files can bring the more accurate tech file parameters and equations into the technology file using the *rhtech* translation utility. Process variation adjustments for combinations of wire width, wire thickness, wire spacing, resistivity, and operating temperature are accommodated in the RedHawk tech file. The values of wire resistance and capacitance used by RedHawk during simulation is then the result of these process variation adjustments available from computations using the new tech file equations and coefficients.

a. Resistance vs. temperature

RedHawk has the capability of modeling temperature-dependent resistance for conducting layers and vias. Temperature is specified in degrees centigrade. Resistance is modeled in the following way, the same method as is used in Spice:

$$R_{\text{eff}} = R_{\text{nominal}} * [1 + \text{Coeff_RT1} * (T - T_0) + \text{Coeff_RT2} * (T - T_0)^2]$$

R_{nominal} is the resistance value at the nominal temperature T_0 . The syntax for specifying the necessary temperature parameters to calculate resistance is as follows:

```
?[ T <operating temp-C>
    Tnom <nominal temp-C>
    Coeff_RT1 <R variation_first order coeff>
    Coeff_RT2 <R variation_second order coeff> ]?
```

where

T <operating temp-C>, T_{nom} <nominal temp-C>: optionally specifies the chip operating temperature and nominal temperature in degrees C. If a temperature model is used using the following temp coefficients, both T and T_{nom} must be specified.

Note: GSR keywords TEMPERATURES (layer-based) and TEMPERATURE (global) take priority over this tech file specification.

Coeff_RT1 <R variation-first order coeff> : specifies the first order metal resistance temperature coefficient, in Ohms per degree C. (default: 0)

Coeff_RT2 <R variation-second order coeff> : specifies the second order metal resistance temperature coefficient, in Ohms per degree C.

Coeff_RT1 and Coeff_RT2 are linear and quadratic temperature coefficients, and R_{eff} is the modeled resistance at the operating temperature T . Notice that the calculated resistance R_{eff} exactly equals the nominal resistance R_{nominal} if $T=T_0$, or if both $\text{Coeff_RT1}=0$ and $\text{Coeff_RT2}=0$.

The coefficient options Coeff_RT1 , Coeff_RT2 , T and T_0 can be specified in the Apache Technology file on a per-layer basis. The default values for Coeff_RT1 and Coeff_RT2 should be set to zero. Values for nominal temperature (T_{nom} or T_0) and operating temperature T must be specified for all layers.

A sample tech file entry for temperature variation is shown below.

```
METAL m1
{
    Thickness      0.2046
```

```

T           125
Tnom        25
Coeff_RT1   0.00265
Coeff_RT2   -2.641e-07
...
}

```

To specify resistance-temperature coefficients as a function of silicon wire width, the following syntax is used:

```

CRT_VS_SI_WIDTH {
    {w1 <Coeff_RT1_1> <Coeff_RT2_1> }
    {w2 <Coeff_RT1_2> <Coeff_RT2_2> }
    ...
}

```

where

w* : specifies silicon wire widths

<Coeff_RT1_*> <Coeff_RT2_*> : specifies the first and second order temperature coefficients of the resistance equation associated with wire width w*. Interpolation used to obtain values.

b. Adjustments for Thickness

A number of process-based adjustments to wire thickness are available for improving the accuracy of resistance and capacitance values. The syntax for these adjustments is described in the following paragraphs.

```

THICKNESS_VS_WIDTH_AND_SPACING [RESISTIVE_ONLY|CAPACITIVE_ONLY] {
    ? SPACINGS { s1 s2 s3 ... } ?
    ? WIDTHS { w1 w2 w3 ... } ?
    VALUES {
        { t11 t12 t13 ... }
        { t21 t22 ... }
        { ... }
    }
}

```

where

Resistive_only/Capacitive_only : by default, thickness values are used for both resistance and capacitance calculation

RESISTIVE_ONLY: thickness values used for resistance calculation only.

CAPACITIVE_ONLY: thickness values used for capacitive calculation only.

WIDTHS { w1 w2 ... } : specifies the widths of the wires in the design

SPACINGS { s1 s2 ... } : specifies the spacings between adjacent wires associated with specific wire widths

VALUES : specifies the thickness change txy corresponding to the proper width and/or spacing values in the WIDTHS/SPACINGS array (at least one parameter array must be provided). The new thickness T = old_T*(1+ t), where t is the value from VALUES table. In the VALUES table, column headers are SPACINGS, and row titles are WIDTHS. Interpolation is used to obtain intermediate values.

```

THICKNESS_VS_DENSITY [RESISTIVE_ONLY | CAPACITIVE_ONLY] {
    {d1 t1} {d2 t2} ...
}

```

where

Resistive_only/Capacitive_only: by default, thickness values are used for both resistance and capacitance calculation

RESISTIVE_ONLY: thickness values used for resistance calculation only.

CAPACITIVE_ONLY: thickness values used for capacitive calculation only.

{ d1 t1 } : specifies the thickness change tx corresponding to the metal density dx for particular area of the chip. The new thickness $T = \text{old_T} * (1 + t)$, where t is the value from the density table. RedHawk uses a 100um X 100um bounding box to calculate density. Interpolation used to obtain values.

```
DENSITY_BOX_WEIGHTING_FACTOR {
    {s1 w1} {s2 w2} {s3 w3} ...
}
```

where

sn : specifies size (in um) of a set of inclusive square bounding boxes for density calculation, where s1 is included in s2, and s2 is included in s3 ... up to sn.

wn : specifies the density weighting factor associated with square sn (from -10 to 10). By default, a table with one 100um X 100um sized box and weighting factor 1.0 is used.

This keyword is used with THICKNESS_VS_DENSITY to calculate the effective thickness by applying the effective density, where the effective density = SUM (density-of-square-sn* wn). Can be used for resistance or capacitance calculation, depending on whether THICKNESS_VS_DENSITY applies to resistance and/or capacitance. No interpolation/extrapolation.

```
BOTTOM_THICKNESS_VS_SI_WIDTH {
    RESISTIVE_ONLY
    CAPACITIVE_ONLY
    {w1 t1} {w2 t2} ...
}
```

where

RESISTIVE_ONLY: specifies only resistive elements are included

CAPACITIVE_ONLY : specifies only capacitive elements are included

{ w1 t1}: specifies the thickness change tx corresponding to the silicon width at the lower plane of the wire. The new $T = \text{old_T} * (1 + t)$, where t is the change in thickness from the width table. Note that the thickness change is made to the bottom of the metal, so the thickness of the dielectric below the metal and distance between metal layers also change correspondingly, which affects capacitance extraction. Interpolation used to obtain values.

POLYNOMIAL_BASED_THICKNESS_VARIATION

Modeling methodology for thickness has significantly changed from 90nm to 65nm technology. In 90 nm technology, the thickness of metal layers, T_{Si} , is modeled by a linear equation that is a function of only the actual ("silicon") area density, D, of each metal layer:

$$T_{Si} = aD + b$$

and

$$\text{Density}, D = (\text{Area}_{Si\text{-all polygons}}) / (\text{Area of meas window})$$

In 65 nm technology, the thickness is modeled by a polynomial equation that is a function of both actual metal area density and drawn width:

$T_{Si} / T_{minW_minS} = a * D^4 + b * D^3 + c * D^2 + d * D + e + 1$

where coefficients a, b, c, d and e are also polynomials and functions of drawn width, and 'T_minW_minS' is the thickness when the width and spacing are minimum values.

Thickness has a similar polynomial relationship as a function of drawn width. The form of the RedHawk Tech file that supports this more accurate thickness modeling is shown in the following example:

```
POLYNOMIAL_BASED_THICKNESS_VARIATION {
    DENSITY_POLYNOMIAL_ORDERS { 4, 3, 2, 1, 0 }
    WIDTH_POLYNOMIAL_ORDERS { 4, 3, 2, 1, 0 }
    WIDTH_RANGES {0.18 12.0000}
    POLYNOMIAL_COEFFICIENTS {
        {0 7.18007E+03 -3.74408E+03 6.25290E+02 3.33498E+01 }
        {0 -8.41826E+03 4.36173E+03 -7.20647E+02 3.75707E+01 }
        {0 3.01180E+03 -1.56096E+03 2.57063E+02 -1.31704E+01 }
        {0 -3.69306E+02 1.93110E+02 -3.19649E+01 1.58144E+00 }
        {0 -2.73935E+00 -9.12962E-01 4.76445E-01 5.41430E-03 }
    }
    POLYNOMIAL_COEFFICIENTS {
        {-3.5524E-03 1.3434E-02 -3.7701E-01 2.1844E+00 -1.1489E+00 }
        {8.6884E-03 6.6935E-02 3.6091E-03 -3.6206E+00 1.9171E+00 }
        {-1.0863E-02 -1.2601E-01 8.4772E-01 1.4232E+00 -9.2238E-01 }
        {7.8418E-03 4.6979E-02 -5.8063E-01 1.1264E-01 6.4299E-02 }
        {-2.0450E-03 -3.3022E-03 1.2592E-01 -1.7997E-01 1.0073E-01 }
    }
    POLYNOMIAL_COEFFICIENTS {
        {0 0 0 0 0 }
        {0 0 0 0 0 }
        {0 0 0 0 0 }
        {0 0 0 0 0 }
        {0 0 0 0 -0.001955 }
    }
}
```

In the table above the columns represent the coefficients for the density polynomial orders 4,3,2,1,0 from left to right. The rows represent the coefficients for the width polynomial orders 4,3,2,1,0 from the first through fifth rows. And the three coefficient tables represent, from top to bottom, a width range of (1) <0.18, (2) 0.18 to 12.0, and (3) >12.0 microns.

Using the example above, the calculation of the ratio of silicon thickness to nominal thickness (ST/NT) is performed as follows:

$ST/NT = alpha*D^4 + beta*D^3 + gamma*D^2 + delta*D + epsilon + 1$

where D = density and coefficients alpha, beta, gamma, delta, epsilon are calculated from the POLYNOMIAL_COEFFICIENTS table:

```
{ a1 b1 c1 d1 e1}
{ a2 b2 c2 d2 e2}
{ a3 b3 c3 d3 e3}
```

```
{ a4 b4 c4 d4 e4}
{ a5 b4 c5 d5 e5}
```

as:

```
alpha = a1*W^4 + b1*W^3 + c1*W^2 + d1*W + e1
beta = a2*W^4 + b2*W^3 + c2*W^2 + d2*W + e2
gamma = a3*W^4 + b3*W^3 + c3*W^2 + d3*W + e3
delta = a4*W^4 + b4*W^3 + c4*W^2 + d4*W + e4
epsilon = a5*W^4 + b5*W^3 + c5*W^2 + d5*W + e5
```

and W = scaled Width.

For W=2 and D=0.4 use the second (middle) POLYNOMIAL_COEFFICIENTS table:

```
{-3.5524E-03 1.3434E-02 -3.7701E-01 2.1844E+00 -1.1489E+00 } => alpha =
1.7624936
{ 8.6884E-03 6.6935E-02 3.6091E-03 -3.6206E+00 1.9171E+00 } => beta =
-4.6351692
{-1.0863E-02 -1.2601E-01 8.4772E-01 1.4232E+00 -9.2238E-01 } => gamma =
4.133012
{ 7.8418E-03 4.6979E-02 -5.8063E-01 1.1264E-01 6.4299E-02 } => delta =
-1.5316402
{-2.0450E-03 -3.3022E-03 1.2592E-01 -1.7997E-01 1.0073E-01 } => epsilon =
0.1853324
```

Therefore, in this example the ST/NT ratio = 0.9824272474.

c. Resistivity and resistance vs. wire width and spacing

If more than one of the following parameters for metal resistance are specified, only one is used, in the following order of priority:

- RHO_VS_SI_WIDTH_AND_THICKNESS
- RHO_VS_WIDTH_AND_SPACING
- RPSQ_VS_WIDTH_AND_SPACING
- RPSQ_VS_SI_WIDTH
- RESISTANCE (wires)

c1. Resistivity

Resistivity, rho (Ohm-microns), resistance, and capacitance are functions of actual metal width and thickness values, which are specified in an M-by-N matrix. For the appropriate width and thickness values, the value of rho is picked from the matrix by RedHawk, as shown in the syntax for the table below. Columns represent width values left to right and rows represent thickness values top to bottom. Linear interpolation is used if a width or thickness is not found in the table.

```
RHO_VS_SI_WIDTH_AND_THICKNESS {
    ? WIDTH { w1 w2 w3 ... wn }?
    ? THICKNESS { t1 t2 t3 ... tn } ?
    VALUES {
        {rho_11 rho_12 rho_13 ... rho_1n }
        {rho_21 rho_22 rho_23 ... rho_2n }
        {rho_31 rho_32 rho_33 ... rho_3n }
        {rho_41 rho_42 rho_43 ... }
        {rho_51 rho_52 rho_53 ... }
        ...
        {rho_n1 ... }
    }
```

where

WIDTHS : specify the wire widths w*

THICKNESS: specifies the thicknesses of the metal t*

VALUES are rho values rho_* corresponding to the width and/or thickness values in the WIDTHS and THICKNESS table. At least one parameter array must be provided. In the 2-D VALUES table, row titles are THICKNESS, column headers are WIDTH. Linear interpolation is used to obtain intermediate values.

Example:

```
RHO_VS_SI_WIDTH_AND_THICKNESS {
    WIDTH { 0.5 1.0 2.0 5.0 10.0 }
    THICKNESS { 0.875 1.25 1.5 1.625 }
    VALUES {
        { 0.0222 0.0209 0.02 0.02 0.02 }
        { 0.0221 0.0218 0.0215 0.0213 0.0211 }
        { 0.022 0.0217 0.0214 0.0212 0.021 }
        { 0.02 0.0199 0.0199 0.0199 0.0199 }
    }
}
```

Resistivity also is a function of actual metal width and spacing values, which are specified in an MxN matrix. For the appropriate width and spacing values, the rho value is picked from the matrix by RedHawk, as shown in the syntax for the table below. Columns represent spacing values left to right and rows represent width values top to bottom.

```
RHO_VS_WIDTH_AND_SPACING {
    ? SPACINGS { S1 S2 S3 ... Sm }?
    ? WIDTHS { W1 W2 W3 ... Wn }?
    VALUES {
        {rho_11 rho_12 rho_13 ... rho_1m }
        {rho_21 rho_22 rho_23 ... rho_2m }
        {rho_31 rho_32 rho_33 ... rho_3m }
        ...
        {rho_n1 ... }
    }
}
```

where

SPACINGS : specifies the spacings s* between metal wires

WIDTHS : specifies the associated silicon wire widths w*

VALUES are rho values rho_* corresponding to the width and/or spacing values in the WIDTHS and SPACINGS table. In the 2-D VALUES table, column headers are SPACINGS and row titles are WIDTHS. Linear interpolation is used to obtain intermediate values.

c2. Resistance

For advanced processes with very small wire dimensions, the sheet resistance for metal lines changes with their width and spacing. RedHawk estimates resistance for metal lines based on their width and/or spacing relative to neighboring metal lines. RedHawk determines the appropriate sheet resistance value to use for a metal line in one of the following ways:

- Using nominal width-spacing pairs specified by the WIDTH-SPACE array, RedHawk associates a particular spacing to a wire based on specified widths. It takes the appropriate sheet resistance value based on that width-spacing value.
- If a nominal WIDTH-SPACE array is not specified, then RedHawk partitions the design into several tiles and determines the routing density for each metal layer in every tile. Based on the routing density in a tile, RedHawk associates a spacing for all metal routes in that layer within a tile. It then determines the sheet resistance to be used based on the spacing value identified for every metal route.

The syntax for this type of resistance model is as follows:

```
? RPSQ_VS_WIDTH_AND_SPACING {
    ? SPACINGS { s1 s2 ... } ?
    ? WIDTHS { w1 w2 ... } ?
    VALUES {
        { RPSQ11 RPSQ12... }
        { RPSQ21 }
        { ... }
    }?
}
? RPSQ_VS_SI_WIDTH {
    {w1 rpsq1} {w2 rpsq2}
}?
```

where

RPSQ_VS_WIDTH_AND_SPACING : defines a table of resistance values, in Ohms per square, for looking up sheet resistance. SPACINGS (s1,s2, ...) define the column headers for the table. Ideal silicon WIDTHS (w1, w2, ...) define the row titles for the table. VALUES are the sheet resistance values (RPSQxy ...) in the WIDTHS/SPACINGS array (at least one parameter array must be provided). Width-Spacing rules:

- unit values are defined in the units section
- the width array must contain at least 1 value, and if more, must be in increasing order. Hence the first element is the minimum width.
- the spacing array must contain at least 1 value, and if more, must be in increasing order. Hence the first element is the minimum spacing. Linear interpolation is used to obtain intermediate values.

RPSQ_VS_SI_WIDTH: specifies the sheet resistance rpsq* corresponding to silicon widths w*. Interpolation used to obtain intermediate values.

Example:

```
METAL metall
{
    MINWIDTH 0.2
    MINSPACE 0.3
    PITCH 0.5
    THICKNESS 0.4
    RESISTANCE 0.091
    EM 0.91
    BLECH_JLC 10.0
    HEIGHT 1
    ABOVE DIEL1
    RPSQ_VS_WIDTH_AND_SPACING {
```

```

        WIDTHS { 0.42 0.84 1 2 }
VALUES {
    { 0.02 }
    { 0.03 }
    { 0.04 }
    { 0.041 }
}
}
```

Simple wire resistance is specified using the syntax:

Resistance <rpsq>

where

<rpsq> specifies the metal sheet resistance, in Ohms per square. Effective resistance is calculated as (rpsq * length) / width_si, where length and width_si are length and silicon width of the metal segment, respectively.

d. Wire Resistance and Capacitance vs. Etch Width and Spacing

Several process variation adjustments for etch adjustment are made for wire resistance and capacitance calculation using the parameter formula:

Silicon_width = drawn_width - etch_adjust

The syntax for providing etch adjustments to width for both resistance and capacitance calculations are given below in priority order. If more than one of the keywords is provided for resistance or capacitance, the higher priority keyword value is used.

```

RESISTIVE_ONLYETCH : Resistance_only
ETCH_VS_WIDTH_AND_SPACING : R and C
CAPACITIVE_ONLYETCH : Capacitance only
SIDE_TANGENT: Capacitance only
ETCH : R and C
```

These parameters are defined on the following pages.

RESISTIVE_ONLYETCH <silicon_width_microns>

Specifies an etch adjustment to resistance for wires throughout the chip.

```

ETCH_VS_WIDTH_AND_SPACING ?[ RESISTIVE_ONLY| CAPACITIVE_ONLY ] ?{
    ? [ PERPENDICULAR_TO_REFERENCE |
        PARALLEL_TO_REFERENCE ]?
    {
        ? SPACINGS { s1 s2 s3 ... } ?
        ? WIDTHS { w1 w2 ... } ?
        VALUES {
            { e11 e12 e13 ... }
            { e21 e22 ... }
            { ... }
        }
    }
}
```

where

RESISTIVE_ONLY/CAPACITIVE_ONLY : specify etch adjustments to width for either resistance or capacitance calculations only. Without

RESISTIVE_ONLY or CAPACITIVE_ONLY, etch values apply to both. For RESISTIVE_ONLY and CAPACITIVE_ONLY, etching on top and bottom

surfaces of the metal is the same amount, that is, $W_{top} = W - 2^*e$ and $W_{bot} = W - 2^*e$, where W is the original drawn width, and W_{top} and W_{bot} are top and bottom silicon wire widths after etching.

Note: the metal wire still has a rectangular cross section

PERPENDICULAR_TO_REFERENCE : when the tech file keyword **REFERENCE_DIRECTION** is set to **VERTICAL**, the parasitic calculations for all vertically-oriented wires use the **PARALLEL_TO_REFERENCE** **ETCH_VS_WIDTH_AND_SPACING** data, and for horizontally-oriented wires the **PERPENDICULAR_TO_REFERENCE** data are used.

PARALLEL_TO_REFERENCE : when the tech file keyword **REFERENCE_DIRECTION** is set to **PARALLEL**, the parasitic calculations for all vertically-oriented wires use the **PERPENDICULAR_TO_REFERENCE** **ETCH_VS_WIDTH_AND_SPACING** data, and for horizontally-oriented wires the **PARALLEL_TO_REFERENCE** data are used.

Note: If any of the three tech file keywords are missing (**REFERENCE_DIRECTION**, **PARALLEL_TO_REFERENCE** or **PERPENDICULAR_TO_REFERENCE**), wire orientation-related etch rate effects are ignored and no etch compensation is done.

WIDTHS : specifies drawn widths of the metal wires in the design

SPACINGS : specifies the spacing dimension between wires

VALUES are etch values corresponding to the width or spacing in the **WIDTHS** or **SPACINGS** tables. In the **VALUES** table, columns headers are **SPACINGS** and row titles are **WIDTHS**. Interpolation used to obtain values.

Example 1: In the example below spacing is defined by columns left to right and widths are defined by rows top to bottom:

```
ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY {
    SPACINGS { 0.378 0.42 0.5 0.84 0.9 1 2 3 }
    WIDTHS   { 0.42 0.84 1.68 2.52 }
    VALUES   {
        { 0.0218 0.0217 0.0214 0.0206 0.0205 0.0203 0.0196 0.0194 }
        { 0.0233 0.0231 0.0228 0.0221 0.0220 0.0218 0.0209 0.0205 }
        { 0.0242 0.0241 0.0239 0.0233 0.0232 0.0230 0.0220 0.0215 }
        { 0.021 0.0245 0.0243 0.0238 0.0237 0.0236 0.0227 0.0221 }
    }
}
```

Example 2:

```
metal M3 {
    ETCH_VS_WIDTH_AND_SPACING CAPACITIVE_ONLY PERPENDICULAR_TO_REFERENCE
        <width and spacing data>
    ETCH_VS_WIDTH_AND_SPACING CAPACITIVE_ONLY PARALLEL_TO_REFERENCE
        <width and spacing data>
    ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY PARALLEL_TO_REFERENCE
        <width and spacing data>
    ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY PERPENDICULAR_TO_REFERENCE
        <width and spacing data>
}
```

ETCH_VS_WIDTH_AND_SPACING ETCH_FROM_TOP (capacitance only)

For **ETCH_FROM_TOP**, the etching occurs in opposite directions on the top and bottom surfaces of the wires. That is, for positive etch from the top, $W_{top} = W - 2^*e$

and $W_{\text{bot}} = W + 2^*e$, and the negative etch from the top case is the reverse. After etching, the metal has a trapezoidal cross section, as shown in Figure C-1 below.

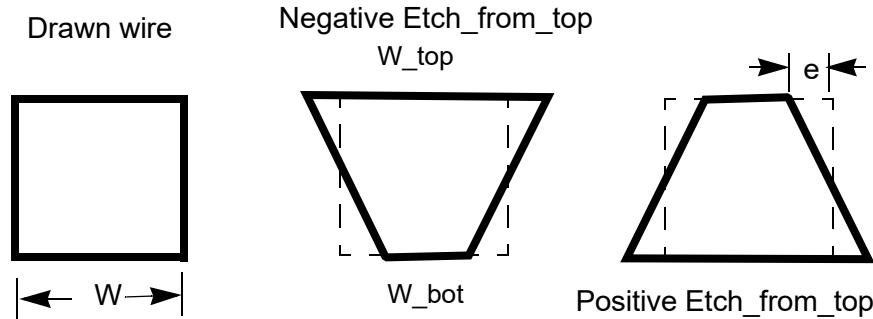


Figure C-1 Etch_from_top parameters

The syntax for the etch from the top capacitance correction is:

```
ETCH_VS_WIDTH_AND_SPACING ETCH_FROM_TOP
? SPACINGS { s1 s2 s3 ... } ?
? WIDTHS { w1 w2 w3 ... } ?
VALUES {
{ e11 e12 e13 ... }
{ e21 e22 ... }
{ ... }
}
```

where

WIDTHS : specifies drawn widths of the metal wires in the design

SPACINGS : specifies the spacing between wires

VALUES are etch values corresponding to the width and spacing in the WIDTHS and SPACINGS table. In the VALUES table, columns headers are SPACINGS and row titles are WIDTHS. Interpolation used to obtain values.

Note that only WIDTHS **or** SPACINGS parameters can be provided, so the VALUES array is one-dimensional.

CAPACITIVE_ONLYETCH <silicon_width_microns>

Specifies an etch adjustment for wire capacitance calculation throughout the chip.

SIDE_TANGENT (capacitance only)

Side_tangent defines the angle of the sides of the wires after etching, relative to the top surface, as shown in Figure C-2. In this type of etch specification, v is the angle from vertical of the conductor sides. After applying this keyword, $W_{\text{top}} = W + v*t$ and $W_{\text{bot}} = W - v*t$, where W is the original drawn width, W_{top} is the top width

after applying SIDE_TANGENT, and t is the thickness of the metal. The metal wire has a trapezoidal cross section.

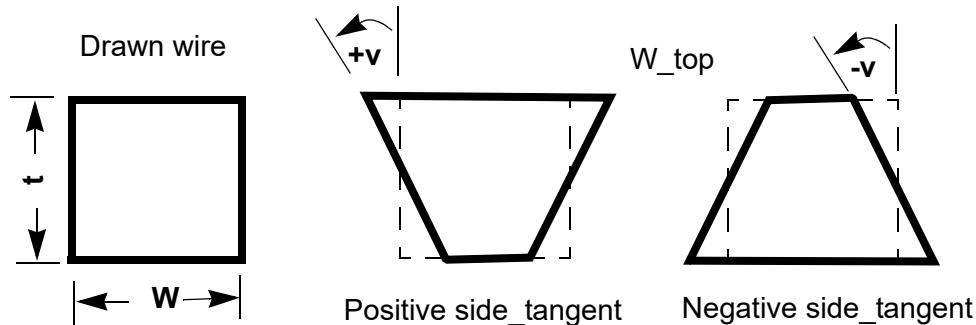


Figure C-2 Side_tangent parameters

The syntax for the side tangent capacitance correction is:

SIDE_TANGENT <v_side_angle>

where

<v_side_angle> : specifies the angle of the side of the wire from vertical after etching

ETCH <silicon_width_microns>

Specifies a default etch adjustment for wire resistance or capacitance calculation throughout the chip.

e. Metal Fill-Related Parameters (capacitance effect)

Several parameters are used to specify non-circuit metal fill topology for adjusting capacitance effect. They are FILL_RATIO, FILL_SPACING, FILL_WIDTH, and FILL_TYPE. When FILL_RATIO is specified for a layer, any empty space encountered during extraction is modeled as though it were filled with floating metal of the same layer. This affects only the vertical capacitance. The keyword syntax is described below.

FILL_RATIO <ratio>

Specifies the ratio of metal fill allowed relative to the legal space available, as constrained by the FILL_SPACING and FILL_WIDTH parameters. The range is 0 to 1.0.

FILL_SPACING <spacing>

Specifies the fixed lateral spacing separating P/G nets and filled objects, in um. Required if FILL_RATIO is specified.

FILL_WIDTH <width>

Specifies the fixed width of metal fill objects, in um. Required if either FILL_SPACING or FILL_RATIO is specified.

FILL_TYPE [GROUNDED | FLOATING]

Specifies whether the filled metal segments are treated as grounded or floating. Default: grounded.

f. Conductor Cladding Parameters

Three keywords are used for specifying cladding effect on conductor resistance, DROP_FACTOR, CLAD_RPSQ and CLAD_THICKNESS. They are described below.

DROP_FACTOR <vert_ht>

where

vert_ht : specifies the absolute height (in um) of the decrease in base height of all conductors above the current conductor that do not have geometries in a given layout area (default 0).

CLAD_RPSQ <rpsq>

where

rpsq : specifies the effective resistance per square of cladding on a conductor, in Ohms/square. This keyword must be used with CLAD_THICKNESS.

CLAD_THICKNESS <thickness>

where

t : specifies the effective thickness of cladding on a conductor, in um.

SUBSTRATE

Defines the thermal conductivity (W/m-degree K), thickness (microns), resistivity (Ohm-cm), and capacitance (pF) for the four layers comprising the substrate.

Optional. Default: none

Syntax:

```
SUBSTRATE [ BULK | EPI | PWELL | NWELL ]
{
? THERMAL_CONDUCTIVITY <value>?
? THICKNESS <value>?
? RESISTIVITY <value>?
? CAPACITANCE <value>?
}
```

Example:

```
SUBSTRATE BULK
{
THERMAL_CONDUCTIVITY 120
THICKNESS 500
RESISTIVITY 0.012
}
```

UNITS

Defines units for parameters in the technology file. *Optional; defaults: shown in table below.*

Syntax

```
units
{
capacitance    <value> (default: 1 pF; 1e-12 Farad)
inductance     <value> (default: 1 nH; 1e-9 Henry)
resistance      <value> (default: 1 Ohm)
length          <value> (default: 1 um; 1e-6 meter)
current         <value> (default: 1 mA; 1e-3 Ampere)
voltage         <value> (default: 1 Volt)
power           <value> (default: 1 Watt)
time            <value> (default: 1 ns; 1e-9 second)
frequency       <value> (default: 1 MHz; 1e+6 Hertz)
}
```

Example:

```
units
{
    capacitance 1p
    inductance 1n
    resistance 1
    length 1u
    current 1m
    voltage 1
    power 1
    time 1n
    frequency 1M
}
```

VIA <via_name>

Specifies the via name and physical parameters defining the physical characteristics of each via. Note that multiple vias can be defined under different via names.

Note that Via parameter definitions are taken first from the Apache Tech file, but for data not present in the Tech file, then Technology LEF file data is used. *Required for all vias.*

Syntax:

```
VIA <via_name> {
    Resistance <resistance_value>
    UpperLayer <metal_layer_above>
    LowerLayer <metal_layer_below> ? <metal_layer_name2>?
    Coeff_RT1 <R variation-first order coeff>
    Coeff_RT2 <R variation-second order coeff>
    TNOM <nominal_temp>
    ...
    ? RPV_VS_AREA {
        {<min-area_1> <resistance-1>
        ...
        <max-area_n> <resistance-n>
        ...
    }
    }?
    ? RPV_VS_WIDTH_AND_LENGTH {
        LENGTHS { <L1>, <L2>, ... }
        WIDTHS { <W1>, <W2>, ... }
        VALUES {
            { <Res_W1_L1> <Res_W1_L2> ... }
            { <Res_W2_L1> <Res_W2_L2> ... }
            ...
        } ?
        WIDTH {<via_width>}
        AREA <via_area>
        WIRE_R_COMPENSATION [0|1]
        RPV_VS_NUM {
            {<num_cuts1>, <resistance1>} {<num_cuts2>, <resistance2>}
```

```

    ...
}

EM <value of EM current limit>
? CRT_VS_AREA {{ <Area1> c11 c12 } {<Area2> c21 c22}
{<Area3> c31 c32 } ... } ?
? TOPOLOGY_BASED_EM {
# number of cuts-based calculation
[ downflow {
    via_number { <n1> <n2> ... }
    metal_width { <w1> <w2> ... }
    em {
        { <em_n1w1> <em_n1w2> <em_n1w3> ... }
        { <em_n2w1> <em_n2w2> <em_n2w3> ... }
    }
}
upflow {
    ...
}
| 
# area of cuts-based calculation
TOPOLOGY_BASED_EM {
    downflow {
        via_area ( <a1> <a2> ... )
        metal_width { <w1> <w2> ... }
        em {
            { <em1> }
            { <em2> }
            ...
        }
    }
    upflow {
        (( same as downflow ))
    }
}
}

EMV_VS_AREA {<areal> <arith_expr_for_EM_limit1>
<area2> <arith_expr_for_EM_limit2>
...
}
}
EM_TEMP_RATING {
    downflow {
        { <temp1> <derating_factor1> }
        { <temp2> <derating_factor2> }
        ...
    }
    upflow {
        { <temp1> <derating_factor3> }
        { <temp2> <derating_factor4> }
        ...
    }
}

```

```
? DIRECTION_BASED_EM_TEMP_RATING {
    downflow {
        { <temp1> <derating_factor1> }
        { <temp2> <derating_factor2> }
        ...
    }
    upflow {
        { <temp1> <derating_factor3> }
        { <temp2> <derating_factor4> }
        ...
    }
}
```

}

}

?

Note: when Upflow and Downflow derating values are equal, then the syntax can be:

```
EM_TEMP_RATING {
    { <temp1> <derating_factor1> }
    { <temp2> <derating_factor2> }
    ...
}

POLYNOMIAL_BASED_EM_DC {
    COND_RULE { ( Wv == 0.5 AND Lv == 0.5 ) AND ( Lb <= 10 )
                AND ( Lu <= 10 ) }
                EM_POLYNOMIAL { 3 * 0.1 }
    COND_RULE { ( Wv == 0.5 AND Lv == 0.5 ) AND ( Lb > 10
                AND Wb >= 2 ) AND ( Lu <= 10 ) }
                EM_POLYNOMIAL { 2*0.1 }
    COND_RULE { ( ( Wv == 0.5 AND Lv == 0.5 ) AND
                  ( NOT ( ( ( Lb <= 10 ) AND ( Lu <= 10 ) ) ) AND
                  ( ( Lb > 10 AND Wb >= 2 ) AND ( Lu <= 10 ) ) ) )
                EM_POLYNOMIAL { 1*0.1 }
    COND_RULE { ( Wv == 1 AND Lv == 1 ) AND ( Lb <= 10 ) AND
                ( Lu > 10 AND Wu >= 10 ) }
                EM_POLYNOMIAL { 3*0.4 }
    COND_RULE { ( Lu >= 5 OR Lb >= 5 OR Wb > 0.18 ) }
                EM_POLYNOMIAL { 2 * 0.187
                    WIRE_MAX 2 * 0.570 * (Wb-0.0085) }
    COND_RULE { ( 10.0 >= Lu >= 4.9 AND
                  Wb <= 0.18 ) AND ( Lu > Lb ) }
                EM_POLYNOMIAL { 1 * 0.187
                    WIRE_MAX (10/Lu) * 0.570 * (Wb-0.0085) }
    ...
}
```

where

L/W: specifies Metal Length/Width

Lv and Wv: specifies the Length/Width of the via (post-shrink size if it is a half node process)

Lu/Wu: specifies the Length/Width of the upper metal layer connecting to this via

Lb/Wb: specifies the Length/Width of the bottom metal layer connecting to this via

WIRE_MAX : optional EM calculation made for both the via and its connected wires . When WIRE_MAX is specified, the via EM criteria before it in the syntax and connected wire EM criteria following it in the syntax are compared, and resulting worst-case EM value is used in the EM analysis.

Note: syntax prior to v11.2 for specifying **POLYNOMIAL_BASED_EM_*** for vias is:

```
? POLYNOMIAL_BASED_EM_[ DC | RMS | PEAK ] {
    LENGTH_RANGES { <Lo> ... <Lx> ... <Ln> }
    EM_POLYNOMIAL {<polynom for EM current for L <= Lo >}
    ...
    EM_POLYNOMIAL {<polynom for EM current for Lx-1 < L <= Lx >}
    ...
    EM_POLYNOMIAL {<polynom for EM current for L > Ln >}
}?
```

where

Via <via_name>: name of via

Resistance <value of resistance> : specifies via resistance in Ohms for a cut with the minimum width w1 from the width table, cross-sectional area w1², and no TEMPERATURE scaling.

UpperLayer : specifies the name of the metal layer above the via

LowerLayer: specifies the name(s) of one or more metal layers below the via

Coeff_RT1 (or _RT2) : specifies the first/second order metal resistance temperature coefficient, in Ohms per degree C. (default: 0)

TNOM: specifies nominal temperature in degrees C

RPV_VS_AREA {a1 r1} : the resistance of a via r* (Tech file resistance units) depends on its area, listed in order of increasing area. Interpolation is performed if the specific area is not in the table, but no extrapolation.

RPV_VS_WIDTH_AND_LENGTH: models the resistance of virtual vias in trench contacts according to the via width and length.

WIDTH <via_width>: width of via (square)

AREA <via_area> : specifies basic via area in um² (from appropriate vendor file). Note that vias on same layer with different cut areas and properties can be defined under different via names.

Note: either WIDTH or AREA must be defined for R and EM calculation.

WIRE_R_COMPENSATION: By default extraction always includes the resistance of wire segments directly below and above via cuts (usually referred as metal-overhang). Set this keyword to 1 if you want to exclude resistance of this type from R extraction for a particular via, and use only the vertical via resistance. Default: 0 (include).

Note: The global WIRE_R_COMPENSATION value is set by the GSR keyword, but does not include pseudo-vias, which must be set in this tech file VIA section. Also, if this keyword is not set in the GSR file, it must be specified in every VIA section in the tech file.

RPV_VS_NUM : specifies the via resistance (Ohms or standard units, unless specified) based on the number of via cuts

EM <value of EM current limit> : EM current limit in mA for a cut with the minimum width w1 from the width table and cross-sectional area w1²

CRT_VS_AREA: specifies area-dependent temperature coefficients of resistance in the Tech file for vias having different cut areas. <Area1> is one via cut area, c11 is first order temperature coefficient of resistance for <Area1> and

c12 is second order temperature coefficient of resistance for <Area>1. Values for specific areas not listed are interpolated.

TOPOLOGY_BASED_EM : can be specified using either a *number-of-cuts* topology based EM limit, or using a *total via area* topology-based EM limit, as described in the syntax descriptions following.

Note that RedHawk supports topology-based via EM rules for vias that are sometimes foundry-dependent. For certain foundry tech-files, RedHawk treats the EM limits as per-via-cut based and all other tech-files, the EM limits are treated as a total limit for multiple via cuts. RedHawk recognizes certain tech files using a tag in the header section, as shown below.

```
# DRM           : T-N28-xx-xx-002 v1.0
```

This tag is automatically inserted by ircx2tech while generating the tech file. If you are using a tech-file for the foundry that is not created through ircx2tech, you must make sure that this tag is inserted in the header section. This feature only affects TOPOLOGY_BASED_EM rules.

Number of cuts-based calculation: EM limit values are the total EM limit for the number of cuts specified. *One exception: for certain foundry rule decks (as identified by the IRCX "# DRM" comment line in the tech file), EM limit values are per-cut, with no interpolation or extrapolation.* For the number of cuts and metal width of the via, the EM limit is picked from the matrix by RedHawk. Columns represent width values left to right, and rows represent the number of cut values top to bottom

downflow: specifies the EM value for vias in which the predominant electron flow direction is from the upper to the lower layer.

upflow: specifies the EM value for vias in which the predominant electron flow direction is from the lower to the upper metal layer.

via_number {<n1> <n2> ...}: specifies the number of cuts in the via

metal_width { w1 w2 ...}: list of widths of the metal wires in the flow direction for the layer that electrons are flowing *from*. If electron flow direction is upward, use the lower metal width, otherwise use the upper layer width.

em_n1w1 em_n1w2 ... em_nwxw } : specifies the maximum EM value for specified combinations of via cuts and specified metal_widths.

Area-of-cuts-based calculation: EM limit values are specified per-cut. For each total via area, the specified maximum EM current is picked from the list by RedHawk. Interpolation and extrapolation are used between limit values.

downflow: specifies the EM value for vias in which the predominant current flow direction is from the upper to the lower layer.

upflow: specifies the EM value for vias in which the predominant current flow direction is from the lower to the upper metal layer.

via_area (<a1> <a2> ...}: list of total area for vias in design

metal_width { w1 w2 ...}: list of widths of the metal wires in the flow direction for the layer that current is flowing *from*. If current flow direction is upward, use the lower metal width, otherwise use the upper layer width.

em { <em1> <em2> ... }: specifies maximum EM current associated with each via area: current <em1> for area <a1>, and so on. Units: mA.

EMV_VS_AREA : specifies the maximum EM current for an area of the via cut. Interpolation is used for missing areas, but not extrapolation. If multiple cuts exists, max current is multiplied by the number of cuts. Units: mA. Regular arithmetic operations are supported to represent EM limits based on area, including square root. Note that there should be no spaces between operators and numbers representing a single number.

See the use of the GSR keyword MERGE_ABUTTED_CUTS to combine

abutted cuts into one ([section "MERGE_ABUTTED_CUTS", page C-664](#)). This option is also used to support slot (non-square) vias in newer technology designs.

EM_TEMP_RATING <temp1> <derating_factor1>: specifies the ratio of maximum current density between <temp1> and TNOM_EM, for derating that is the same in both directions. The maximum current density limit at <temp1> is calculated by multiplying the <derating_factor1> by TNOM_EM. Different ratings for upflow and downflow current directions can be specified. *Note that this setting only affects Average EM limits, and not RMS and PEAK limits.*

DIRECTION_BASED_EM_TEMP_RATING: specifies the ratio of maximum current density between <temp1> and TNOM_EM, for derating that is different in the UPFLOW and DOWNFLOW directions. The maximum current density at <temp1> is calculated by multiplying the <derating_factor1> by TNOM_EM. *Note that this setting only affects Average EM limits, and not RMS and PEAK limits.* Also note that this is outside of the **TOPOLOGY_BASED_EM** keyword, and that

via_area (<a1> <a2> ...): specifies total areas for vias in design

em { <em1> <em2> ... }: specifies maximum EM current associated with each via area: current <em1> for area <a1>, and so on. Units: mA.

POLYNOMIAL_BASED_EM_* : defines tables for maximum DC (or AVG), RMS, and PEAK EM current, based on the lengths and widths of the attached metal, and **EM_POLYNOMIALS** entries define maximum EM limit as functions of length and width for each length and width range. The larger of the lengths of attached Upper and Lower metal layer segments is used in the via EM calculation equation (units: mA).

Each group of **EM_POLYNOMIALS** represents a specified LENGTH_RANGE, and each line represents a specified WIDTH_RANGE. If WIDTH_RANGES is specified before LENGTH_RANGES, then WIDTH_RANGES is used as the primary table index.

Temperature derating only for DC (AVG). Use one very large length entry to indicate no length dependency. For the RMS polynomial relationship, delta_T can be included to represent the expected temperature change due to Joule heating (set the delta_T value with the GSR keyword DELTA_T_RMS_EM).

VIAMODEL

Required for **RedHawk** if WIDTH sub-keyword is not defined in VIA keyword. This information is usually available from the LEF file.

Syntax:

```
VIAMODEL <model_name>
{
    <metal_layer_name>  <X_loc_lower_left> <Y_loc_lower_left>
    <X_loc_upper_right> <Y_loc_upper_right>
    <via_layer_name>    <X_loc_lower_left> <Y_loc_lower_left>
    <X_loc_upper_right> <Y_loc_upper_right>
    <metal_layer_name>  <X_loc_lower_left> <Y_loc_lower_left>
    <X_loc_upper_right> <Y_loc_upper_right>
}
```

Example:

```
VIAMODEL via1
{
    metal2  -0.2 -0.2 0.2 0.2
```

```
v1      -0.095 -0.095 0.095 0.095
metall1 -0.21  -0.21 0.21 0.21
}
```

Global Switching Configuration (GSC) File

The Global Switching Configuration file specifies the switching status of blocks, instances, and voltage domains in the design for both power calculation and simulation steps, using the fixed state keywords or multi-state definitions from SIM2IPROF, AVM, or APLMMX.

You can simulate multiple states in a vectorless simulation for macros with a multiple-state current model, in a GSC-based vectorless flow. You must ensure that the multiple states are captured in the current file as defined with the CUSTOM_STATE_SIM_TIME configuration file keyword in SIM2IPROF or APLMMX. Use predefined and user-defined custom states in the GSC file. With extended simulation time the same sequence is repeated.

Note the following GSC behavior for different design conditions:

- If all GSC setting values are for a single cycle, RedHawk repeats the state over the full simulation time.
- If any GSC setting includes multiple states, RedHawk triggers multi-cycle GSC flow.
- During multi-cycle GSC flow, for cells with multiple state settings in the GSC, RedHawk repeats the pattern.
- For cells with a single state setting in the GSC, RedHawk honors the first cycle state and randomly picks states for succeeding cycles.
- If any instance is repeated in multiple GSC files, values that are changed in a later GSC file are used, and warning GSC-028 is displayed to indicate that previous values for the instance are overwritten. Unchanged values are retained in a later GSC file .

During GSC file parsing at the database setup stage, RedHawk checks for instances not controlled by power gates that are assigned a low-power related state. If switching states are detected for instances that are not part of a power gating network, a warning message is issued and these instances are assigned a “DISABLE” state. This check is controlled by the GSR keyword ‘DYNAMIC_GSC_CHECK’, which is On by default . This check prevents problems in analysis caused by incorrect state assignments.

The name of the GSC file is specified in the GSR file (see [section "GSR File Keywords", page C-573](#)) and the state definitions are in the GSC config file:

Syntax (in GSR file):

```
GSC_FILES <gsc_FilePathName>
```

Syntax (in GSC file):

```
[<blockName> | <instanceName> ] ? <domain_name>?
<state1> ?<state2> ... ?
```

where <stateN> can be

```
[ UNDECIDED | TOGGLE |HIGH | LOW | POWERUP | POWERDOWN |
POWERUP_DOWN | POWERDOWN_UP |STANDBY | DISABLE | ENABLE |
OFF | <custom_state_name> ]
```

Wildcards (*) can be used for defining sets of blocks and instances. The default state of an instance or block is UNDECIDED unless defined otherwise.

NOTE: Reading large GSC files sometimes can cause long run times because of iteration to map hierarchical names, particularly if some are not in the design. For GSC files that contain only instance names (that is, no wildcard or block names in the GSC file), you may add a comment line in GSC file

```
# EXACT_INSTANCE_NAME_MATCH
```

which instructs RedHawk to do faster exact and direct name mapping.

Following any section of exact name matching, a comment line “#
 END_EXACT_INSTANCE_NAME_MATCH” must be inserted to differentiate
 exact matches from any subsequent normal wild card entries. If an “#
 END_EXACT_INSTANCE_NAME_MATCH” comment is missing in the GSC file,
 it will automatically be appended at the end of the GSC file.

The following keywords are used to define the state of cells (instances of standard cells, memories, I/Os, hard macros), blocks, or domains during simulation:

<blockName>/<instanceName> : specifies the name of an instance or block for
 which you want to set the state

<domain_name>: optionally specifies a domain net name in <blockName>/
 <instanceName>, for which you want to set the state

UNDECIDED (default): the state of the instance/block is determined by RedHawk
 during simulation.

TOGGLE: specifies that the instance *will* toggle during the simulation, and
 RedHawk determines its switching direction. If a block is specified, *all*
instances in the block's hierarchy toggle during simulation. This state is not
 recommended for realistic analysis.

HIGH: identifies the high switching charge condition, which could be either Write
 or Read for memory, or a 0->1 change for a single output, or the
 representative output.

Note: HIGH and LOW designate the order of APL characterization, not necessarily
 the power level.

LOW: identifies the low switching charge condition, which could be either Write or
 Read for memory, or a 1->0 change for a single output, or the representative
 output.

POWERUP: specifies that the switch instance or the power-gated block/instance
 will power up during simulation.

Note: the state of both switch instances AND power-gated block/instances must be
 specified.

POWERDOWN: specifies that the switch instance or the power-gated block/
 instance will power down during simulation.

POWERUP_DOWN: specifies that the switch instance or the power-gated block/
 instance will power up, and then power down, during the simulation period.

POWERDOWN_UP: specifies that the switch instance or the power-gated block/
 instance will power down, and then power up, during the simulation period.

STANDBY: specifies that no output or signal will toggle for the instance/block, but
 its clock (if available) is still switching.

DISABLE: specifies that the block or instance is inactive. However, instances in
 this state consume leakage power.

ENABLE: specifies that the block or instance is active.

OFF: specifies that the switch instance or power-gated block/instance is in the
 OFF state. Instances in this state consume zero leakage power.

<state2> ... : added state definitions for cycle 2 and after, which can be any of the
 predefined or user-defined custom states; the first state defines cycle 1,
 <state2> defines cycle 2, and so on.

Example:

```
GSC_FILES DesignABC/g1_sw2.gsc
```

Then in the GSC file *DesignABC/g1_sw2.gsc*, the contents could be:

```
blockABC Powerup
```

```
block2C toggle
instB* standby
instC1 LOW StateB2 HIGH StateD1
ABC VDD1 High
```

Note that to read in a GSC file on the fly, use the TCL command

```
import gsc ABC.gsc
```

where *ABC.gsc* is the GSC filename. You **cannot use the command**

```
gsr set gsc ABC.gsc
```

Global System Requirements File (*.gsr)

The Global System Requirements (GSR) file contains the input design file specifications, operating conditions for chip for power calculation, static IR drop and EM analysis, and dynamic voltage drop analysis.

When specifying values in the GSR file, RedHawk supports the following units and prefixes. Units are case sensitive.

Unit prefixes

- terra = t = 1e+12
- giga = g = 1e+9
- mega = M or me = 1e+6
- kilo = k = 1e+3
- milli = m = 1e-3
- micro = u = 1e-6
- nano = n = 1e-9
- pico = p = 1e-12
- femto = f = 1e-15

Example: 1.3p = 1.3e-12 (there is no space between the number and the unit symbol)

Unit length conversions

- 1 mil = 0.001 inch = 2.54e-5 meters
- 1 inch = 2.54e-2 meters
- 1 micron = 1.0e-6 meters

GSR File Keywords

NOTE: Keyword names make no distinction between upper and lower case.

Input Data Keywords

ADD_PLOC_FROM_DEF

Allows specifying an additional DEF file that defines PLOCs, in addition to the top DEF file. In this case RedHawk only uses PIN information from this DEF and does not read any other info from it. The DEF file pathname specified in the keyword must be the same as specified in the DEF_FILES keyword, whether an absolute or a relative path. Note that input DEF files in gzipped format are supported. *Optional. Default: none.*

Syntax:

```
ADD_PLOC_FROM_DEF <DEF file path>
```

ADD_PLOC_FROM_TOP_DEF

If set to 1, the PINS section in the top level DEF is used as the power and ground pad locations, provided that power and ground pin geometries in the PINS section are well-defined with physical shapes for connection to top level. You can also select a single metal layer from which the plocs are created. This option is most useful in block level analysis, where block level pins are defined in DEF. If no metal layers are specified, plocs from all metal layers are created. The specification should be on only one line. If a non-existent metal layer name is specified, the process errors out with an appropriate message.

Note that RedHawk exits if a script contains both import pad commands and 'ADD_PLOC_FROM_TOP_DEF 1', and also if neither power nor ground pins are defined.

(The keyword 'ADD_PAD_LOC_FROM_TOP_DEF' is the same function, but is now obsolete.) *DMP compatible. Optional; default: 0 (no automatic copying from DEF).*

Syntax:

```
ADD_PLOC_FROM_TOP_DEF [0|1] { <layerName1> <layerName2> ... }
ADD_PAD_LOC_FROM_TOP_DEF [0|1] { <layerName1> <layerName2> ... }
```

AUTO_PAD_CONNECTION_LAYERS

Creates pad instance connection plocs automatically. RedHawk searches within the specified range of each pad cell by layer and generates one P/G source for each pad cell at the closest location on the specified layers. For the 'eco add pad' command, if x,y locations outside of the wire geometry are specified, and AUTO_PAD_CONNECTION_LAYERS is set, the x,y locations are snapped to existing geometry location and the pad is generated. Note that you cannot specify net names in the PLOC when using AUTO_PAD_CONNECTION_LAYERS or an error message is generated. *Optional. Default: none.*

Note: this keyword replaces previous keyword PLOC_SNAP_DISTANCE.

Syntax:

```
AUTO_PAD_CONNECTION_LAYERS {
    <layerA> <snap_range>
    <layerB> <snap_range>
    ...
    DEFAULT <default_snap_range>
    REMOVE_FLOATING_PINS [ yes | no ]
}
```

where

If a layer has no snap range specified, but the DEFAULT range is set, the default value is used.

If no range is set for a layer and no DEFAULT range is set, a pin is not snapped for the layer.

REMOVE_FLOATING_PINS yes: removes plocs that cannot be snapped to valid locations and remain at their original position, potentially creating shorts or warnings later in the flow.

Example:

```
AUTO_PAD_CONNECTION_LAYERS {
    M1 10.0
    M2
    M3 15.0
```

```
    DEFAULT 5.0
}
```

BLOCK_POWER_ASSIGNMENT

For designs that are in early stages of development, and do not have complete placement and routing information, the BLOCK_POWER_ASSIGNMENT keyword allows designers to evaluate the grid response to power assignments within unfinished blocks or sub-areas of the chip. These areas for analysis can be without assigned instances or they may contain instances that do not have power information yet. So in the early stages of design, you can make power assignment assumptions about a block or region and then analyze the grid response at nodes in the assignment area in terms of voltage drop, current, and EM values.

There are four primary ways in which BPA is used to assign power for analysis:

- use an existing defined block using the BLOCK/PIN options
- create a new block definition using FULLCHIP/REGION options, or
- define a modified area within a block for analysis
- pin-based block power assignment

Each of these types of analysis has its own syntax and usage, which are described in the following sections. *Required for early design analysis. Default: None.*

a. Existing block power assignment

- To assign power to instances of an existing cell from LEF/DEF, the syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    [ <instName> [BLOCK | PIN]
    [ <layer name> | ALL | TOP | BOTTOM ]
    <netName>? :<pinName>?
    [ <powerW> | <GndCurrentA> | -1 ]
    ...
}
```

- To assign power to all instances of a master cell of an existing block from LEF/DEF, the syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    <cellName> CELLTYPE
    [ <layer name> | ALL | TOP | BOTTOM ]<pinName>
    [ <powerW> | <GndCurrentA> | -1 ]
    ...
}
```

where

<instName>: name of existing instance

<layer name> : specifies the layername for assignment

ALL : includes all layers. Multiple layer specifications can be provided for the same instance and net, in which case the power or current is added and distributed equally between nodes on the P/G grid on the specified layers. If ALL layers are specified, all nodes on the grid are considered.

TOP/BOTTOM : includes either the highest or the lowest mask layers overlapping the BPA instance. When layername, TOP, or BOTTOM specifications are used, *all* power/current is assigned to the BPA blocks on the top, bottom or the specified metal layer in the layout.

<netName>: specifies the power/ground net to have power assigned. All nets to be included must be specified on a separate line. Power for switch sub-nets (internal nets) needs to be explicitly specified. Note that power specified for switch ext. net does not include that consumed by internal nets.

<pinName>: Specifies power/current for a particular pin. Note that different power or current for different pins on the same net may be assigned. Also, **<pinName>** is optional when **<type>** is not CELLTYPE.

Syntax:

```
<blockName> <type> <layerName> <netName>:<pinName>
<power/current> ...
```

When **<type>** is CELLTYPE, the syntax is:

```
<cellName> CELLTYPE <layerName> <pinName> <power/
current>
```

<powerW>/<GndCurrentA> : specifies either the power in Watts for power nets, the ground current in Amps for ground nets,

'-1' : allows power calculation to determine the appropriate power/current based on other information available, such as toggle rate, BPFS and APL characterization. This can be used for either cell instances (blocks) or regions. You must provide correct and complete power calculation data so that power for the block can be calculated accurately. See [section "Power Calculation", page 4-35](#), for details. Use one line in the BPA specification for every domain (net) in the block. Note that if one net in the block is assigned '-1' for power/current in BPA, power/current for *all* domains in the block is determined by the power calculation engine.

<cellName> : specifies the master cell name

b. Create a new block definition for power assignment

b1. To assign power to a user-defined block/instance and assign power, the syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    <BPA_instName> REGION
    [ <layer name> | ALL | TOP | BOTTOM ]
    <netName> ?:<pinName>?
    [ <powerW>|<GndCurrentA> ] <llx> <lly> <urx> <ury>
    ...
}
```

where

<BPA_instName> REGION : specifies a user-defined name for the region to have power assigned, which *cannot* be the design name, nor the name of an existing cell or instance. This also means **<BPA_instName>** becomes an instance in the design and must be used in other file references also, such as STA and GSC.

For backward compatibility, note that for prior-release GSR files that have used the former "FULLCHIP FULLCHIP ..." syntax in B_P_A, **RedHawk** substitutes the region name "adsFULLCHIP" to allow static analysis to proceed. However, you must use the "adsFULLCHIP" **<BPA_instName>** also in STA, USER_STA and GSC files to perform dynamic analysis.

<llx> <lly> <urx> <ury>: specifies lower left and upper right corner coordinates of defined region

b2. To create a new instance for cells defined in DECAP_CELL, or to use the BLOCK_POWER_MASTER_CELL keyword (see section "BLOCK_POWER_MASTER_CELL", page C-581), and also allow overlaps, the syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    <BPA_instName> <cellName>
    [ <layer name> | ALL | TOP | BOTTOM ] <netName>
    [ <powerW> | <GndCurrentA> ]
    <llx> <lly> <orientation>
    ...
    ?<BPA_instName> <cellName> OVERLAP_OK?
}
```

where

<cellName> : cell name specified in GSR keyword DECAP_CELL or BLOCK_POWER_MASTER_CELL
<llx>,<lly> : specifies lower left corner of cell
<orientation> : specifies orientation of cell
OVERLAP_OK : for instances that overlap defined BPA regions, ensures that the instance is not deactivated in such cases. The BPA regions that overlap any instance other than the parent instance must be defined as 'OVERLAP_OK'.

b3. To assign power in a region that includes the entire design, the syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    <BPA_instName> FULLCHIP
    [ <layer name> | ALL | TOP | BOTTOM ]
    <netName>?:<pinName>?
    [ <powerW> | <GndCurrentA> ]
    ...
}
```

where

FULLCHIP: specifies that the whole design is included in the <BPA_instName>

b4. For BPA REGION assignments with multiple power/ground nets, you can define pins and P/G arcs for the master cell of the BPA region in the custom library file, and specify net-pin connections in BPA with a '<netName>:<pinName>' format.

For example, for a BPA region *TOP/region1* in power domains VDD1 and VDD2, an example custom lib file would look as follows:

```
cell TOP_region1 {
    pin vdd1 {
        type vdd
    }
    pin vdd2 {
        type vdd
    }
    pin vss {
        type vss
    }
    pgarc {
        vdd1 vss
        vdd2 vss
    }
}
```

```
    }
}
```

And the associated example GSR would be as follows:

```
BLOCK_POWER_ASSIGNMENT {
    TOP/region1 REGION M1 VDD1:vdd1 0.1 100 100 300 450
    TOP/region1 REGION M1 VDD2:vdd2 0.1 100 100 300 450
    TOP/region1 REGION M1 VSS:vss 0.2 100 100 300 450
}
```

BPA region instances are listed in the file *<design>.inst.pin*. For static analysis, BPA REGION instances are listed in files *<design>.inst.arc* and *<design>.inst.worst*.

You can also exclude regions from power assignment in early analysis for designs at a stage where place and route is complete for some hierarchical blocks, and you want to use power calculation data for the completed blocks and use power assignment for rest of the design. See [section "Power Assignment to MMX Pin-based Regions", page 4-66](#), for more details on MMX BPA use.

For power/current assigned to hierarchical elements in a design, power/current assigned to child elements are included in the parent's assignment. So a parent block's power assignment is the sum of the power assigned specifically to its children and also the power assigned to areas that are outside of its children. For more details, see [section "Early Analysis Flow", page 4-61](#).

c. Define a modified area within a block for power assignment

To define sub-regions inside an existing BPA instance (multiple specifications on the same instance are allowed) for power assignment, the syntax is:

```
BLOCK_POWER_ASSIGNMENT (
    <BPA_instName> BLOCK AREA <llx> <lly> <urx> <ury>
    <BPA_instName> BLOCK RECTILINEAR <x1 y1 x2 y3 x4 y5 ...>
    <BPA_inst_name> BLOCK CURRENT_DENSITY <node_spacing>
    [<powerW> | <GndCurrentA> ]
    ...
}
```

where

BLOCK AREA <llx> <lly> <urx> <ury>: specifies a rectangular area to be assigned power with x,y coordinates

BLOCK RECTILINEAR <x1 y1 x2 y3 x4 y5 ...>: specifies corners of the rectilinear in order (either direction). The sides must be either horizontal or vertical. To avoid repetition of the same coordinate value, you can use a reduced format for providing the coordinates, in which a repeated x or y value can be dropped, such as “x1 y1 x2 y3 x4 y5 ...”, where y2 x3 y4 x5 ... are repeated values and do not need to be re-entered. Also, to assist in keeping track of x,y values for the reduced format, the following format is allowed “20x 15y 190x 30y 150x 80y 70x 20y 20x”, in which case the letters are ignored. However, for ease of use in understanding and reading back the x and y values, the full syntax with repeated x and y coordinates is often preferred.

BLOCK CURRENT_DENSITY <node_spacing> : assigns current sinks and sources with different spacing for different BPA instances (microns)

To define and exclude a region from BPA power assignment, the syntax is:

```
BLOCK_POWER_ASSIGNMENT (
    <regionName> REGION EXCLUDE <llx> <lly> <urx> <ury>
    ...
)
```

}

The INCLUDE and EXCLUDE capabilities make it easier for you to define areas that are not complete rectangles to be specified for power assignment. The EXCLUDE option allows you to define rectangular regions to be excluded from areas occupied by BPA blocks/regions, and the INCLUDE option then can *add back* areas to be included *within* EXCLUDE areas.

To define and exclude an overlapping instance in the design from BPA power assignment, or exclude certain cellnames, the syntax is:

```
BLOCK_POWER_ASSIGNMENT (
    [ <instName> BLOCK | <cellname> CELLTYP ] EXCLUDE
    ...
}
```

To modify an area defined by 'EXCLUDE' to form rectilinear regions, the syntax is:

```
BLOCK_POWER_ASSIGNMENT (
    <regionName> REGION INCLUDE <llx> <lliy> <urx> <ury>
    [<powerW> | <GndCurrentA> ]
    ...
}
```

To use the boundary box of an existing instance to do the same as described above, add the line:

```
<instName> BLOCK INCLUDE
```

To exclude macro instances overlapping a BPA instance for power assignment, add the line:

```
<BPA_instName> BLOCK EXCLUDE_MACRO
```

To exclude instances defined in the BLOCK_POWER_FOR_SCALING keyword, and which overlap the BPA instance for power assignment, add the line:

```
<instName> BLOCK EXCLUDE_BPFS
```

The BLOCK EXCLUDE_MACRO/BPFS options specify that areas under any macro that overlap the specified block are ignored, and the power assigned to the block is not distributed to the excluded macros/memories. This is applicable to blocks defined as FULLCHIP, BLOCK or REGION.

To exclude an area occupied by all instances overlapping the BPA instance, add the line:

```
<BPA_instName> BLOCK OVERLAP_OK}
```

Note that BPA assignments can be performed interactively on-the-fly. For details, see [section "Block Power Assignment On-the-fly", page 4-64](#).

Example:

```
BLOCK_POWER_ASSIGNMENT {
    65NM_ABC FULLCHIP M6 VDD 0.6
    region1 REGION via3 VDDC 0.3 120.0 450.0 200.5 530.5
    INST1 BLOCK EXCLUDE
    regionNotForBPA REGION EXCLUDE 1000 1000 4000 4000
    INST1/I2 BLOCK INCLUDE
    subRegionForBPA REGION INCLUDE 1500 1500 2000 2000
    X0/adsU1/RR BLOCK OVERLAP_OK
```

}

In the above example, areas inside excluded regions, as occupied by instance INST1/I2, and the rectangle (1500 1500 2000 2000) are included and used for power assignment. Note that regions to be included or excluded can be specified either by coordinates, or by instance name, and that INCLUDE statements can only be applied to and modify EXCLUDED areas.

d. Define pin-based block power assignment

MMX pin-based region power assignment is supported for MMX instances that have many P/G pins inside to represent transistors using keywords MMX_PIN and MMX_REGION. So the power is only distributed to P/G pins, while regular region-based BPA assigns power to nodes on the given layer within the region. This feature is used to change the power distribution inside a MMX instance for static analysis only. To assign power to a DEF instance, or a region of a mixed signal design, the syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    [<DEF_inst_name> MMX_PIN ALL <NetName>
     [<power_W> | <GndCurrentA> ]
     ...
     [<DEF_inst_name>/<BPA_region_name>/<subregion_name>/...
      MMX_REGION ALL <NetName>
      [<powerW> | <GndCurrentA> ]
      <llx> <lly> <urx> <ury> ]
      ...
    }
}
```

where

- <DEF_inst_name> MMX_PIN ALL : specifies the name of an entire DEF instance to have power assigned
- <NetName>: specifies the name of net to be analyzed
- <DEF_inst_name>/<BPA_region_name>/<subregion_name> : specifies the hierarchy of a defined and named region to have power assigned
- MMX_REGION ALL : indicates a sub-area of the full instance is to be defined and considered
- <llx> <lly> <urx> <ury> : specifies the lower left and upper right x,y coordinates of the region to be analyzed

e. Defining white box cells in Block Power Assignment

The **Redhawk** White Box Cell + BPA flow can accept an input BPA file along with the cell names from GDS to assign power using BPA option MMX_CELLTYPE to create MMX_REGION data for those cell names. You are not required to white box any of the cell names provided in the BPA file. The syntax is:

```
BLOCK_POWER_ASSIGNMENT {
    <cell_name> MMX_CELLTYPE ALL <net_name>
    [<power_Watts> | <Gnd_current_Amps>]
    ...
}
```

In the following example, 0.022 W is assigned to white box cell x1adda for net VDD:

```
BLOCK_POWER_ASSIGNMENT {
    adsU1 MMX_PIN ALL VDD 0.0001
    x1adda MMX_CELLTYPE ALL VDD 0.022
}
```

BLOCK_POWER_ASSIGNMENT_FILE

Specifies one or more files for defining parameters for early power analysis (information specified in BLOCK_POWER_ASSIGNMENT keyword). User can set this GSR keyword prior to setup design using TCL command “gsr set BLOCK_POWER_ASSIGNMENT_FILE <BPA_filename>”. *Optional. Default: none.*

Syntax:

```
BLOCK_POWER_ASSIGNMENT_FILE <file1>
...
```

or

```
BLOCK_POWER_ASSIGNMENT_FILE {
<file1>
...
}
```

Example:

```
BLOCK_POWER_ASSIGNMENT_FILE {
bpa.txt
BPA_xyz.txt
}
```

The syntax and sample contents of a block power assignment file is as follows:

```
BLOCK_POWER_MASTER_CELL {
<BPMC_parameter_values>
...
}

BLOCK_POWER_ASSIGNMENT {
<BPA_parameter_values>
...
}

BLOCK_POWER_MASTER_CELL {
CELL2 0 0 1980 1720
}
BLOCK_POWER_ASSIGNMENT {
INST1 CELL2 METAL4 vdd_i 0.01 1000 2000 N
INST1 CELL2 METAL4 vss 0.00920 1000 2000 N

REGION2 REGION METAL6 vdd_e 0.00500 14250 1500 16750 3500
REGION2 REGION METAL6 vss 0.005259 14250 1500 16750 3500
}
```

The specified values in the BPA section of the GSR are recorded in the output file *adsRpt/apache.blockPowerAssign*.

BLOCK_POWER_MASTER_CELL

Specifies multiple regions, and optionally rectilinear areas inside BPA REGIONS or in LEF/DEF blocks, that share a common master cell. Rectilinear area definitions enable you to distribute power to specified sub-areas within a rectilinear block. Power/current assigned to REGIONS/blocks is hooked up within the user-defined rectilinear areas. You must define the common master cells and the rectilinear

regions within it using a BLOCK_POWER_MASTER_CELL definition and the BLOCK_POWER_ASSIGNMENT keyword.

For each master cell the bboxes inside the brackets {} define rectilinear subareas inside the cell for power assignment. You must make sure that the bboxes inside each cell fall within the bbox of the cell; areas outside of the master cell bbox are discarded. *Optional. Default: none.*

Syntax:

```
BLOCK_POWER_MASTER_CELL {
    <master_cell_name1> <BB_llx> <BB_lly> <BB_urx> <BB_ury> {
        ? AREA <llx> <lly> <urx> <ury> ?
        ...
        ? RECTILINEAR <x1 y1 x2 y3 x4 y5 ...>?
        ...
    }
}
```

Syntax:

```
BLOCK_POWER_MASTER_CELL {
    <master_cell_name1> <BB_llx> <BB_lly> <BB_urx> <BB_ury>
    ...
    ?<master_cell_name2> <bbox> {
        <sub-area bbox>
        ...
    }?
}
```

Example:

```
BLOCK_POWER_MASTER_CELL {
    rcell 0 0 400 200
    myRegion 0 0 200 100 {
        0 0 100 100
        100 0 200 50
    }
    defCell 0 0 200 300
    {
        0 0 200 100
        0 100 100 300
    }
}

BLOCK_POWER_ASSIGNMENT {
    regionA rcell MET3 vdd 0.12 100 100 N
    regionA rcell MET3 vss 0.12 100 100 N
    regionB rcell MET1 vdd 0.1 500 100 E
    regionB rcell MET1 vss 0.1 500 100 E
    REGION1 myregion MET5 VDDC 1 1850 1030 N
    REGION1 myregion MET5 VSS 1 1850 1030 N
    INST2 BLOCK MET5 VDDC 0.5
    INST2 BLOCK MET5 VSS 0.5
}
```

In this example, regions ‘regionA’ and ‘regionB’ share the same master cell, ‘rcell’, and ‘defCell’ is the master cell of ‘INST2’.

BLOCK_STA_FILES

Running STA non-hierarchically is the most accurate and recommended method. However, BLOCK_STA_FILES allows hierarchical STA files to be used by RedHawk in analysis. The hierarchical STA files are compiled into a composite file as if the STA data was flat. Multiple instantiations of a hierarchical STA block are allowed, but for best results every block specified should have its own STA data with correct boundary conditions. Using BLOCK_STA_FILES for a top level STA file is also acceptable.

Hierarchical STA works best under the following conditions:

- Blocks have donut style designs, with all paths having FFs on both input and output.
- All instances driven by block specific FFs have updated timing windows.
- Instances of the same block have symmetrical instantiations.
- If there is a top level timing block, it should be specified in BLOCK_STA_FILES

Conditions in which hierarchical STA analysis is *not recommended* are:

- Many feedthrough paths in the block instances.
- Boundary conditions are very different between blocks.
- Blocks have different STA modes and need to be simulated as such.

FREQ_OF_MISSING_INSTANCES <frequency_in_Hertz> specifies the operating frequency for instances not in the STA file, but in the design (the default is 0).

INSTTYPE is optional with STA file of top block and instances.

CELLTYPE should be used with STA file of any hierarchical CELL. Each instance of this cell is annotated with the given STA file.

For a block, if the STA file is provided separately using INSTTYPE, and this instance also matches a cell type for which the STA file is specified through the CELLTYPE keyword, then the STA file provided through the separate line (with INSTTYPE) gets priority.

Note: If both BLOCK_STA_FILES and STA_FILES are used, then BLOCK_STA_FILES is ignored, and if a USER_STA_FILE is also used, any common data in the specified USER_STA_FILE overrides data in either of the other files.

See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#) for more information on how to use hierarchical STA blocks. *DMP compatible*.

Optional. Default: none.

Syntax:

```
BLOCK_STA_FILES {  
    FREQ_OF_MISSING_INSTANCES <frequency_in_hertz>  
    [<top_block> | <Hierarchy_path/instance_name>]  
    <filename> ? INSTTYPE ?  
    ...  
    <cell_name> <filename> CELLTYPE  
    ...  
}
```

Example:

```
BLOCK_STA_FILES {  
    Top_level_block top.timing  
    InstA1 abc12.timing
```

```

InstB2  efc23.timing
InstC3  xyz56.timing
InstC3/instd1 ghi78.timing INSTTYPE
BlockZ1 blockz1.timing CELLTYPE
}

```

BLOCK_VCD_FILES

Specifies files and parameter values for VCD-based analysis. *DMP compatible.*
Optional. Default: None

Syntax:

```

BLOCK_VCD_FILES {
    VCD_FILE {
        <Hierarch_path/DEF_instance_name1> <VCD file1>
        <VCD file1 options>
    }
    VCD_FILE {
        <Hierarch_path/DEF_instance_name2> <VCD file2>
        <VCD file2 options>
        ...
    }
}

```

where

<Hierarch_path/DEF_instance_name> <VCD file>: specifies the hierarchical block name that matches that identified in the DEF and the absolute or relative path to the VCD or FSDB file
<VCD file options> : for VCD_FILE keyword option definitions, see [section "VCD_FILE", page C-609](#).

Example:

```

BLOCK_VCD_FILES {
    VCD_FILE {
        mydesign/block_A mydesign.vcd
        FILE_TYPE VCD
        FRONT_PATH "mydesign_vcd/blockA"
        true_time 1
    }
    VCD_FILE {
        top2 vcd.2
        front_path "TOP/CHIP/top2/"
        true_time 1
    }
}

```

BPA_BY_CURRENT

When set to 1, allows specifying current values (Amps) for both power and ground domains in performing BLOCK_POWER_ASSIGNMENT, as well as in Watts.
Optional; default 0 (off).

Syntax:

```
BPA_BY_CURRENT [ 0 | 1 ]
```

Example:

```
BPA_BY_CURRENT 1
BLOCK_POWER_ASSIGNMENT {
    adsU1/R1 REGION ALL vddcp 0.02 75.849 265.795 185.445 642.312
    adsU1/R1 REGION ALL gnda 0.0111 75.849 265.795 185.445 642.312
}
```

In this example, a 0.02 Amp current is assigned to domain vddcp, and 0.0111 A to domain gnda.

BPA_BY_LAYER

By default power assigned using the BLOCK_POWER_ASSIGNMENT keyword is distributed equally among the nodes of all layers specified in the BPA statement, without taking into consideration per layer breakup. When turned on, this keyword allows assignment per layer. However, note that for any BPA blocks that have nets or layers assigned a power/current of -1 (or negative), 'pwrcalc' computes the power for the block, which is distributed uniformly for all nodes on different layers without regard for this keyword setting. *Optional; default 0 (off).*

Syntax:

```
BPA_BY_LAYER [ 0 | 1 ]
```

BPA_CONN_DISTANCE

Defines a distance around the original bbox of a BPA instance to search for a node to be connected to the BPA instance for current assignment. Note that just one nearest node outside of the bbox is selected. This is suitable for cases in which you want the BPA region to perform as a single current sink point and the BPA area is too small to accurately specify a bbox that is small enough to cover only one node. *Optional; Default: 0.0 (use the original BPA bbox).*

Syntax:

```
BPA_CONN_DISTANCE <distance_um>
```

BPA_CONN_MARGIN

Defines a margin to be extended from the original bbox of a BPA instance, to enlarge the area where nodes can be selected for power/current assignment. This is suitable for cases in which the nodes are at or very near the BPA boundary, and where small inaccuracies or roundoff in co-ordinate values can cause the nodes to lie outside of the BPA boundary and cause disconnects. *Optional; Default: 0.0 (use the original BPA bbox).*

Syntax:

```
BPA_CONN_MARGIN <distance_um>
```

BPA_CURRENT_DENSITY

Allows power/current sinks to be distributed uniformly on metal layers, not based on via density, but on the global pitch defined by this keyword. This early analysis scenario simulates a condition in which standard cells are already placed. *DMP compatible. Optional. Default: 0.0 (no extra nodes added).*

Syntax:

```
BPA_CURRENT_DENSITY <value>
```

where '<value>' is the pitch in microns (≥ 1).

CELL_GEOMETRIES_AS_MFILL

Provides control over MFILL pushing of LEF cell geometries to achieve high accuracy in TSMC N10 resistance correlation cases, by allowing obstruction and signal pin geometries to have the most influence on metal etch calculations, including color mask information and exact spacing. This keyword interacts with the GSR keyword READ_LEF_OBS in that setting CELL_GEOMETRIES_AS_MFILL either to OBS or ALL disables the other average spacing flow for obstruction impact on etching. To use the older flow for obstruction modeling, this keyword must be set to [NONE/0] or SIGNAL. *DMP compatible.* *Optional.* *Default:* 0.

Syntax:

```
CELL_GEOMETRIES_AS_MFILL { OBS | SIGNAL | ALL/1 | NONE/0 }
```

where

OBS: pushes all LEF obstructions to metal wires on the MFILL net

SIGNAL: pushes all signal pin instances to metal wires on the MFILL net

[ALL|1]: pushes both obstruction and signal pin geometries to wires on MFILL

[NONE|0]: no geometries are pushed (default)

CELL_PIN_FILE

Specifies a cell pin file that helps reduce extraction runtime in designs involving cells that have many overlapping pins in different metal layers, such as metal1 and metal2. Using this keyword you can control which metal layer should be used for hooking up the current sinks in the analysis. You can specify the cell name, or use the option STD_CELL to cover all the standard cells. *Optional. DMP compatible.*

Default: None

Syntax:

```
CELL_PIN_FILE <filename>
```

The internal file syntax is as follows:

```
CELL [ STD_CELL | <cell_name> ]
PIN [ <Vdd_pin_name> | <Gnd_pin_name> ]
LAYER <metal_layer>
END PIN
END CELL
```

Pin file content example:

```
CELL STD_CELL
PIN VDD1
LAYER M1
END PIN
PIN VSS1
LAYER M1
END PIN
END CELL
```

CELL_TYPE_FILE

When specified, removes false via collisions due to incorrect via compression (combining via groups and including cover cells improperly). If the design has cover cells, use this keyword to specify a file that defines cell types, to insure proper recognition of cover cells and standard cells. If a cell type file is not used, instance count statistics are used to identify a cell as a standard cell or not, but it does not recognize cover cells. The contents of the cell type file have the following format:

```
<cellname1> [ s | c ]
<cellname2> [ s | c ]
...
...
```

where 's' indicates a standard cell and 'c' indicates a cover cell. *Optional. Default: None.*

Syntax:

```
CELL_TYPE_FILE <filename>
```

CELL_WELL_CAP_FILE

If CELL_WELL_CAP_FILE is defined for PAD/filler cells , RedHawk parses and marks well cap cells and sends them to the simulation input file *apache.well* to be included in dynamic simulation. The format of the well cap file is as follows:

```
<cell_name> <num_arcs> {<Vdd_name> <Vss_name> <Cap> <Eff_R>} ...
```

Optional. Default: none.

Syntax:

```
CELL_WELL_CAP_FILE <filename>
```

CPA_FILES

Specifies CPA package design data files. *DMP compatible. Optional: Default: None*

Syntax:

```
CPA_FILES {
    PACKAGE <package_layout_filename>
    MODEL <project_path>
}
```

where

PACKAGE: specifies the package layout file to be displayed and imported into RedHawk-CPA. The supported formats are *xfl*, *mcm*, *sip* and *brd*. The layout file can be located at a remote location with only read permission. The layout file name is used as the project name.

MODEL: defines the path to the CPA project directory. A new directory is created if it does not exist. If the MODEL keyword points to a previously-saved project, the existing project is loaded, and the package layout is not re-imported.

DEF_FILES

Required keyword that specifies the Database Exchange Files (*.def) to be used for the design. The last file listed must be the “top” file for the design.

Note: if different nets connect to the same instance pin, this creates an unacceptable connectivity (connection) collision, and RedHawk can only pick one of the nets. The chosen net depends on the format of the instance name. The DEF file SPECIALNETS section supports wildcards (*) in instance names, so the priority of net connection selection is as follows:

1. exact instance name, such as *MAC/VCCL_MACRO* (*MACRO/inst01 DVDD*)
2. wildcard instance name, such as *MAC/VCCK_MACRO* (*MACRO/* DVDD*), or *VCCK (* DVDD)*

So if there is a net name collision for an instance whose name has a wildcard, the exact name is kept and RedHawk issues WARNING (DEF-012) and continues; that is, the wildcard entry has a lower priority in net connectivity. When a net name

collision occurs for an exact instance name (no wildcard), RedHawk issues WARNING (PIS-605) and replaces the first name encountered with the later one. The DEF construct + VIA <via_name> <point_coordinate> is supported in SPECIALNETS section as via definitions.

DMP compatible. Note that DEF files should be imported directly using this keyword. Required. Default: None.

Syntax:

```
DEF_FILES
{
    <def_FilePathName-1>.def block
    ...
    <def_fFilePathName-n>.def top
}
```

where

<def_FilePathName-1>.def : specifies the path and filename for a block; the last .def file should be the top-level DEF file.

Example:

```
DEF_FILES
{
    ../design_data/ABCDE.def top
}
```

DEF_HONOR_HALF_NODE_SCALE_FACTOR

In the absence of tech LEF data, RedHawk picks the width of routes from the MINWIDTH option of the METAL keyword defined in the tech file. By default RedHawk calculates the width as (MINWIDTH)/(HALF_NODE_SCALE_FACTOR), where HALF_NODE_SCALE_FACTOR is a GSR setting. To turn off this behavior, set this keyword to off. *Optional. Default: On.*

Syntax:

```
DEF_HONOR_HALF_NODE_SCALE_FACTOR [ 0 | 1 ]
```

DEF_IGNORE_LAYERS

Specifies layers in the DEF file to be ignored during design data input for RedHawk analysis. *DMP compatible. Optional. Default: None.*

Syntax:

```
DEF_IGNORE_LAYERS {
    <layer_name1>
    ...
    <layer_nameN>
}
```

Example:

```
DEF_IGNORE_LAYERS {
    m4
    m6
}
```

DEF_IGNORE_NETS_WIDTH

Ignores unnecessary NET section width definitions in DEF generated by place and route tools. *Optional. Default: 1 - On*

Syntax:

```
DEF_IGNORE_NETS_WIDTH [ 1 | 0 ]
```

DEF_IGNORE_PIN_LAYERS

Specifies PIN layers in the DEF file to be ignored during design data input for RedHawk analysis. *DMP compatible. Optional. Default: None.*

Syntax:

```
DEF_IGNORE_PIN_LAYERS [
<layer_name1>
...
<layer_nameN>
]
```

Example:

```
DEF_IGNORE_PIN_LAYERS {
metal3
metal5
}
```

DEF_IGNORE_SPECIFIC_LAYERS

For LEF/DEF importing, allows you to ignore specified layers for specified blocks. *DMP compatible. Optional. Default: None*

Syntax:

```
DEF_IGNORE_SPECIFIC_LAYERS {
<def_block1_path> "<ignored_layer1> ..."
<def_block2_path> "<ignored_layer2> ..."
...
}
```

where '<def_blockN_path>' is a specified block path, followed by layers to be ignored.

DEF_READ_ALL_IO_NETS

When set, enables importing all primary I/O nets in all DEF blocks in the design hierarchy, so you *do not need to* specify signal net names in each DEF block individually in ESD_SIGNAL_NETS in order to analyze them. RedHawk automatically traverses all connected metals from all hierarchies of DEF, irrespective of the names in lower level DEFs. *Optional. Default: 0.*

Syntax:

```
DEF_READ_ALL_IO_NETS [ 0 | 1 ]
```

DEF_READ_CLOCK_ONLY

When set along with IMPORT_NETS GSR keyword, only the clock nets (nets with "USE CLOCK" attribute in the DEF) are imported for SignalEM analysis. *Default: 0.*

Syntax:

```
DEF_READ_CLOCK_ONLY [ 0 | 1 ]
IMPORT_NETS {
```

```
" * [all] "
}
```

DEF_TRUE_PATH_EXTENSION

Allows control over the extension of wires in DEF by a specified value (or half wire width). This keyword is automatically turned on in the signal EM flow, when you use the command: ‘setup analysis_mode signalEM’.

In general, DEF uses the following syntax to define routing points in the NETS or SPECIALNETS sections:

```
( <x> <y> <extenValue> ) { ( <x> <y> <extenValue> ) | <viaName> [orient] } ...
where
```

extenValue : specifies the amount by which the wire is extended past the endpoint of the segment. The extension value must be greater than or equal to zero.

For example:

```
NETS 6 (or SPECIALNETS 6)
-vss
(inst4 SE)
+ SOURCE NETLIST
+ USE POWER
+ FIXED MET2 ( 10500 1957 <extenValue>) ( * 2350 <extenValue> )
NEW MET2 (10500 2350 <extenValue>)( 10700*<extenValue>) ;
```

This DEF wire extension feature is used as follows:

- If a routing point has an <extenValue> in DEF, both NETS and SPECIALNETS are extended by that amount past the end point, regardless of the value of the DEF_TRUE_PATH_EXTENSION keyword.
- If a routing point has no <extenValue>, the following conditions apply:

for DEF_TRUE_PATH_EXTENSION= 0 :

NETS---> wires not extended

SPECIALNETS---> wires not extended

for DEF_TRUE_PATH_EXTENSION =1 :

NETS---> wires extended half wire width

SPECIALNETS---> wires not extended

DMP compatible. Optional. Default: 0, Off.

Syntax:

```
DEF_TRUE_PATH_EXTENSION [ 0 | 1 ]
```

DESIGN_IMPORT_REGION

Specifies the region of interest in the complete design context (not a DEF block). This region can cover multiple instances of the same DEF block or different def blocks. The difference between DESIGN_IMPORT_REGION and IMPORT_REGION is that IMPORT_REGION is specific to a DEF block. Even if the user specifies the region of the top DEF, it is applicable for the geometries in that top and not to the geometries inside a block. For example, with a region x1, y1, x2, y2 in the top, all the block DEFs are read in full detail. Only the top DEF is retained for that region. So, if there is any instance (including the sub block) that falls/ overlaps with this region, the whole instance of it is retained. However, in DESIGN_IMPORT_REGION, the bounding box will be applied and only the portion

of the block that overlaps within this region is retained and the rest discarded. This approach is lesser efficient as RedHawk has to read all the blocks and retain all of them before discarding the unwanted regions. However, this is more usable.

Syntax:

```
DESIGN_IMPORT_REGION {  
    <ll_x> <ll_y> <ur_x> <ur_y>  
    <ll_x> <ll_y> <ur_x> <ur_y>  
    ...  
}
```

FAST_DEF_READ

Specifies the number of additional processes to be used for importing DEFs, which is recommended for use in the DMP flow. *Optional. Default: 0 for flat and 4 for DMP run.*

Syntax:

```
FAST_DEF_READ <num_proc>
```

FAST_DEF_READ_DIR

Specifies the area to store RHDB. For DMP flow, customer should specify a shared location for this keyword, else RedHawk will quit abnormally.

Syntax:

```
FAST_DEF_READ_DIR <path>
```

EXACT_PINPORT_VM

To create new via models for single cut lef/def via if no matching via model is found.
(Default: 0 in which tool picks a via model from the tech LEF)

Usage:

```
EXACT_PINPORT_VM [ 0 | 1 ]
```

EXCLUDE_REGION

Specifies one or more rectangular regions of a cell or design to be excluded from DEF, using lower left and upper right x,y coordinates. *Optional. Default: None.*

Syntax:

```
EXCLUDE_REGION {  
    <cell_name/design_name> {  
        <ll_x> <ll_y> <ur_x> <ur_y>  
        ...  
    }  
}
```

If both EXCLUDE_REGION and IMPORT_REGION are included, **RedHawk** exits after importing all defs.

GDS_CELLS

Specifies a set of GDS cells converted by *gds2def/gds2def -m* or *gds2rh/gds2rh -m*, and where the converted LEF, DEF, and LIB files are located. The box cell **adsgds[1/2].lef* files generated by *gds2def/gds2rh* are automatically read in, using the GDS_CELLS keyword. Also see use of GDS_CELLS_FILE keyword.

Usage Notes:

- a. If you use GDS_CELLS input for *gds2def* or *gds2def -m* cells in the design, you must use the LEF_FILES, DEF_FILES, and LIB_FILES GSR keywords

for input of any additional LEF, DEF, and LIB files (not converted by *gds2def/gds2def -m*). In other words, when using GDS_CELLS input, do not use the ‘import’ command for any cell files.

- b. If there are multiple cells from GDS conversion defined in GDS_CELLS, the top-level hierarchy should be the last entry.

DMP compatible. Optional. Default: none.

Syntax:

```
GDS_CELLS
{
    <GdsCellName-1> <path_to_cell-1> ? mmx ?
    ...
    <GdsCellName-n> <path_to_cell-n> ? mmx ?
}
```

where

<GdsCellName-1> <path_to_cell-1> : specifies the cell name and path to the file.

mmx : option that identifies a mixed mode transistor model rather than a detailed memory model

Example:

```
GDS_CELLS
{
    mem_cell1 ./user_data/gds2def
}
```

GDS_CELLS_FILE

Specifies a file that lists the locations of all cells, the same as using the keyword GDS_CELLS directly. The contents of the GDS_CELLS_FILE are of the form:

```
cellA my_gds_dir
cellB my_gds_dir
cellC my_gds_dir2
...
...
```

DMP compatible. Optional. Default: None.

Syntax

```
GDS_CELLS_FILE <filename>
```

GSC_FILES

Specifies the Global Switching Configuration file to be used and the status of any instances or blocks in the design where power switching occurs. If the GSR keyword GSC_OVERRIDE_IPF is set to 1 and there is no VCD setting, the GSC_FILES has priority over IPF and BPFS settings. However, by default the OVERRIDE keyword is set to 0 and GSC does not have priority. Also, if a VCD file setting exists, the GSC value overrides the VCD for any instances specified. You can enable/disable LDOs using the GSC file, but you can only set On/Off-states for LDOs as an initial condition, and cannot change the On/Off states of LDOs for dynamic simulation. Wildcard (*) entries are supported for instance names.

For details of the contents and syntax of the GSC file, see [section "Global Switching Configuration \(GSC\) File", page C-571](#). Note that to read in a GSC file on the fly, use the TCL command

```
import gsc ABC.gsc
```

where *ABC.gsc* is the GSC filename. You now cannot use the command

```
gsr set gsc ABC.gsc
```

DMP compatible. Optional. Default: None.

Syntax in GSR file:

```
GSC_FILES <gsc_FilePathName>
```

Syntax in GSC file:

```
[<blockName> | <instanceName> ] ? <domain_name>?  

[ UNDECIDED | TOGGLE | HIGH | LOW | POWERUP | POWERDOWN |  

STANDBY | ENABLE | DISABLE | OFF | <custom_state_name> ]  

...
```

Example in GSR file:

```
GSC_FILES DesignABC/gl_sw2.gsc
```

Example in GSC file DesignABC/gl_sw2.gsc:

```
blockABC Powerup
block2C toggle
instB* standby
instC1 stateB2
ABC VDD1 High
```

HONOR_LEF_PIN_TYPE

When set, disables “name-based” P/G pin/net type setting (default). When Off, LEF nets having “VDD”, “vdd”, “VCC”, or “vcc” as part of their names are considered as power nets, and nets having “VSS”, “vss”, or “gnd” as part of their names are considered ground nets. Used along with keyword PGNET_HONOR_DEF_TYPE for DEF name handling. *DMP compatible. Optional. Default: 1.*

Syntax

```
HONOR_LEF_PIN_TYPE [0|1]
```

HOOK_INTERNAL_PIN_POWER_EXTERNAL

When set, extracts internal power pins from .lib to map the pins in LEF, linking internal power pins to external power pins based on the pg_function description. *Optional. Default: 0.*

```
HOOK_INTERNAL_PIN_POWER_EXTERNAL [0|1]
```

IGNORE_INSTANCES_ON_TOP_DEF_REGIONS

Can be used to specify the corners of regions in which to ignore instances. The regions must encompass the instances you want to ignore completely, otherwise they remain in the simulation. The units are um. *Optional. Default: None.*

Syntax:

```
IGNORE_INSTANCES_ON_TOP_DEF_REGIONS {
    <ll_x> <ll_y> <ur_x> <ur_y>
    ...
}
```

Example:

```
IGNORE_INSTANCES_ON_TOP_DEF_REGIONS {
    0 0 100 100
    150 150 200 250
```

}

IGNORE_INST_WITH_NO_MASTER

Specifies handling of instances that have no master cell defined. The behavior depends upon whether the flow is ERV or non-ERV, and also on the setting of the keyword IGNORE_LEF_DEF_MISMATCH, as described following:

IGNORE_INST_WITH_NO_MASTER = 0 (default)

- ERV --> creates new cells and quits after reading all DEFs
- Non-ERV

 IGNORE_LEF_DEF_MISMATCH = 0 (default) --> creates new cells and quits after reading all DEFs

 IGNORE_LEF_DEF_MISMATCH = 1 --> creates new cells and continues process

IGNORE_INST_WITH_NO_MASTER = 1

 Reports cells with no LEF file in the *apache.refCell.noLef* file.

 IGNORE_LEF_DEF_MISMATCH = 0 (default) --> does not create new cells and quits after reading all DEFs

 IGNORE_LEF_DEF_MISMATCH = 1 --> does not create new cells and continues process

IGNORE_INST_WITH_NO_MASTER = 2 --> does not create new cells and quits after reading all DEFs

DMP compatible. Optional. Default: 0.

Syntax:

 IGNORE_INST_WITH_NO_MASTER [0 | 1 | 2]

IGNORE_PLOC_INTERNALNETS

When set to 0, RedHawk places plocs on internal nets. By default (1), any plocs placed on internal nets are ignored. *DMP compatible. Optional. Default: 1.*

Syntax:

 IGNORE_PLOC_INTERNALNETS [0 | 1]

IMPORT_NETS

Specifies nets to be imported. In the Signal EM flow, only the net(s) specified are imported, and no overlapping other nets. Note that the net names should be full path names after flattening. The keyword also supports equivalent PG net tracing. When this keyword is enabled in GSR, 'FAST_DEF_READ 4' is automatically set. *DMP compatible. Optional. Default: none.*

Syntax:

```
    IMPORT_NETS {  
        <net1_name>  
        <net2_name>  
        ...  
    }
```

The keyword also supports the format "blocka/*clk*[all]". Therefore, in the signal EM flow, user can import all the clock nets with specified pattern within the specified block or within the inner hierarchies inside the specified block.

Syntax:

```
    IMPORT_NETS {
```

```
"<block_name>/<net pattern> [all]"
}
```

Example:

```
IMPORT_NETS {
    " blocka/*clk* [all]"
}
```

In this case, "*clk*" nets, which are defined only within "blocka" or with in the blocks instantiated under "blocka" will be imported. This format removes the redundancy of specifying "*clk*" with each and every block instantiated under "blocka" like:

```
IMPORT_NETS {
blocka/*clk*
blocka/blockb/*clk*
blocka/blockb/blockc/*clk*
}
```

where blockb, blockc are the blocks instantiated under blocka, blocka/blockb respectively.

IMPORT_NETS_FILE

Specifies an input file that contains a list of net to be imported for a selective nets project. If the number of nets to be imported is very large, using the GSR keyword IMPORT_NETS to specify a list of net names can creates a very large GSR file, which may extend run time parsing the GSR. so the child processes can skip parsing this file in the FDR flow. *Optional. Default: none.*

Syntax:

```
IMPORT_NETS_FILE <nets_file>
```

IMPORT_REGION

Specifies one or more rectangular regions of a cell or design to be imported from DEF, using lower left and upper right x,y coordinates. *DMP compatible. Optional. Default: None.*

Syntax

```
IMPORT_REGION {
    <cell_name/design_name> {
        <ll_x> <ll_y> <ur_x> <ur_y>
        ...
    }
    ...
}
```

IMPORT_REGION_CELL_OVERLAP

When set to 'full', import blocks that are completely within the region specified by the GSR keyword IMPORT_REGION. When set to 'partial', Partial cells at the region boundaries are also imported. *Default: partial.*

Syntax

```
IMPORT_REGION_CELL_OVERLAP [ partial | full ]
```

KEEP_POWER_DATA

Allows control of unnecessary power data generation. If turned Off, regenerates RedHawk data for power calculation for each power calculation. By default RedHawk power data is retained. *Optional. Default: On.*

Syntax:

```
KEEP_POWER_DATA [ 0 | 1 ]
```

LEF_CELL_IGNORE_PIN_LAYERS

When set, ignores specified cell pin layers (supports wildcards), and overrides the GSR keyword LEF_IGNORE_PIN_LAYERS. If both keywords are set, INFO message GSR-201 is displayed in the log file. *Optional. Default: none.*

Syntax:

```
LEF_CELL_IGNORE_PIN_LAYERS {  
    <cell_name> <POWER | GROUND | SIGNAL | ALL | BOTH |  
    <pin_name> <layer1, layer2,...>  
}
```

where

BOTH: ignores both Power/Ground LEF geometries in LEF file.

ALL: ignores Power/Ground/Signal LEF geometries in LEF file.

Example:

```
LEF_CELL_IGNORE_PIN_LAYERS {  
    cellA VDD2 MET2,MET1  
    cellA VDD* MET2  
    cellA VDD1 MET3  
    cell* VDD2 MET3,MET4  
    cell* VDD* MET1  
    cellB VDD2 MET5,MET3  
}
```

LEFDEF_TECH_LAYER_MAP_FILE

Layers of the same type present in *LEF/DEF/tech.lef*, but with different names, can be mapped using this keyword. LEF/DEF layer names are mapped to the ones defined in the tech file during LEF/DEF imports. It can also map LEF/DEF layer names used in RDL_CELL and PAD_FILE to the ones defined in the tech file. The layer map file format is:

```
<LEF/DEF_layer_name> <techlayer_name>  
DMP compatible. Optional. Default: none.
```

Syntax:

```
LEFDEF_TECH_LAYER_MAP_FILE <layermap_file>
```

LEFDEF_TECH_LAYER_MAP

Specifies the layer names to be mapped from LEF/DEF to TECH directly within the GSR file. The GSR keyword "LEFDEF_TECH_LAYER_MAP_FILE" specifies the layer name mapping within a file. Note that within a GSR file only one keyword should be set, else RedHawk will exit.

Syntax:

```
LEFDEF_TECH_LAYER_MAP {  
    <LEF/DEF_layer_name> <techlayer_name>  
    ...  
}
```

Example:

```
LEFDEF_TECH_LAYER_MAP {  
    VIA0 VIA0_M0_STI  
    M0 M0_STI  
}
```

LEF_FILES

Required to specify the physical Library Exchange Files (*.lef) to be used in the design. *Note that LEF files should be imported directly using this keyword. DMP compatible.* Required. Default: None.

Syntax:

```
LEF_FILES {  
    <lef_FilePathName-1>.lef  
    ...  
    <lef_FilePathName-n>.lef  
}  
where
```

<lef_FilePathName-1>.lef : specifies the path and name of the LEF file; the first .lef file should contain the technology information.

Example:

```
LEF_FILES {  
    ../design_data/ABCD.lef  
}
```

LEF_IGNORE_PIN_LAYERS

Specifies PIN layers in the LEF file to be ignored during design data input for RedHawk analysis. *DMP compatible.* Optional. Default: None.

Syntax:

```
LEF_IGNORE_PIN_LAYERS {  
    <layer_name1>  
    ...  
    <layer_nameN>  
}
```

Example:

```
LEF_IGNORE_PIN_LAYERS {  
    M4  
    M5  
}
```

LIBERTY_DB

Liberty files are parsed only once in the RedHawk flow. A binary intermediate file is generated as a Liberty database (part of the RedHawk DB), and all previous data operations on Liberty files are replaced by functions in the database. The Liberty database also includes CCS library info, and is saved/reloaded using normal DB operations. CMM generation also stores this. Power calculation processes Liberty files and generates the Liberty database, so it can support Liberty input from both original Liberty files and the Liberty database. This feature has three values. *Optional. Default: off.*

Syntax:

`LIBERTY_DB [0 | 1 | 2]`

where

0 - off : Liberty parser is called multiple times to process the Liberty files.

1 : dumps all the Lib cells to DB

2 : dumps the Lib cells that have corresponding LEF definitions in the DB

LIB_FILES

Required to specify Synopsys library files (*.lib) or custom lib files to be used in the design. If a directory is specified, all files in the directory are selected. Note that LIB files should be imported directly using this keyword. Also, only one custom lib file may be specified. *DMP compatible. Required. Default: None.*

Syntax:

```
LIB_FILES {
    [ <lib_filename> | <lib_file_dir> ] ? CUSTOM ?
    ...
}
```

where

<lib_filename> : specifies library filename

<lib_file_dir> : specifies library directory

CUSTOM : specifies a custom library file. Only one may be specified. For example, this allows including or excluding particular states from power calculation using a custom LIB file. You can do this for a particular cell, or if one or more cells is *not* specified, it applies globally for all cells. User has options to include/exclude specific states of a cell when calculating power When multiple logic conditions are given, “OR” implementation is done for these. Wildcards are allowed for pin names (and not logic conditions). Logic conditions must be within quotes (“ ”) The syntax for a custom cell-level include/exclude Lib file would be:

```
cell <cell_name> {
    exclude_states {
        pin <active_pin_name> {
            <Full/Partial Boolean expression1>
            <Full/Partial Boolean expression2>
            ...
        }
    }
    include_states {
        pin <active_pin_name> {
            <Full/Partial Boolean expression1>
            <Full/Partial Boolean expression2>
            ...
        }
    }
}
```

Example 1:

```
LIB_FILES {
    special/custom.lib CUSTOM
    dirAB/memory/mem.lib
}
```

Sample contents of custom.lib file:

```

cell SDFFX1HVT {
    exclude_states {
        pin CK {
            "EN * !(SE) * !(Q)"
        }
    }
    cell cell_abc {
        exclude_states {
            pin * {
                "RST&!CS"
            }
        }
    }
}

```

LIB_HONOR_OUT_SIGNAL_SWING

When set, RedHawk supports LVDS pad concept of reduced power swing. *Default: 0.*

Syntax:

LIB_HONOR_OUT_SIGNAL_SWING [1 | 0]

LIB_IGNORE_CELL_LEAKAGE

When set, forces power calculation to use the average of state-dependent leakage power values. *Default: 0.*

Syntax:

LIB_IGNORE_CELL_LEAKAGE [1 | 0]

LIB_IGNORE_IO_VOLTAGE

Controls reading of Vimax, Vimin, Vih, Vil, Vol, Voh, Vomin, and Vomax values from LIB files in RedHawk when encountering Liberty files with unsupported Vi^*/Vo^* formats. The default value 0 allows reading in these values from LIB, while setting it to 1 ignores the values. . *Optional; default: 0.*

Syntax:

LIB_IGNORE_IO_VOLTAGE [1 | 0]

LIB_IGNORE_POWER_THRESHOLD

When set, ignores unreasonably high power values in lib. That is, when the value in lib is larger than the specified threshold value, the lib value is ignored. Threshold values should be specified in units of Joules, in floating point format. *DMP Compatible. Optional. Default: None.*

LIB_IGNORE_POWER_THRESHOLD <thresh_value_J>

LIB_PIN_CAP

When set to max/min, RedHawk uses the maximum/minimum lib value between the rise/fall_capacitance_range and the pin's capacitance. When this keyword is not set, the typical value of pin capacitance is used. *Optional; default: typical capacitance value.*

Syntax:

LIB_PIN_CAP [min|max|typ]

MACRO_POWER_FILES

Specifies one or more files in which the regions of memories or other macros are described, including their types, power pins, instantiations, current consumption, and bounding boxes. In cases when detailed GDS modeling is unnecessary and requires excessive resources, the MACRO_POWER_FILES keyword can be used to efficiently model macros. Current can be distributed or weighted non-uniformly among LEF pins such that different functional portions of the macro are accounted for. For example, the total current can be distributed among the sense amp, X and Y decoder and the memory array regions of a memory cell, depending on the current sourced in that particular functional block. In this way a better granularity of current distribution over the memory/macros is achieved, without having to use a full GDS (*gds2def/gds2rh*) flow. The LEF pins of the macro are grouped into appropriate regions and assigned the appropriate power. The LEF pins may be split into multiple pins in the case where a pin is a stripe that crosses regions. *Optional; default: None.*

Syntax:

```
MACRO_POWER_FILES {
    <memory_macro_filename>
    ...
}
```

Example:

```
MACRO_POWER_FILES {
    memory_macroMN.pwr
}
```

The specified macro definition file has a format as in the following example:

```
LEF_MACRO rf16x72cm1bw {
    REGION_TYPE {
        <type-name>< Vdd-pin> <Vss-pin> <current_fraction>
        senseamp      VDD      VSS      .6
        predecode     VDD      VSS      .10
        memarray      VDD      VSS      .13
        xdecode       VDD      VSS      .02
    }
    REGION_BBOXES {
        <region-bbox-name> <region-name> <ll_x> <ll_y> <ur_x> <ur_y>
        sensemap1      senseamp      0.0      0.0      40.0      10.0
        sensemap2      senseamp      60.0      0.0      100.0     10.0
        predecode      predecode     40.0      0.0      60.0      10.0
        memarray1      memarray      0.0      10.0      40.0      50.0
        memarray2      memarray      60.0      10.0      100.0     50.0
        decode         xdecode       40.0      10.0      60.0      50.0
    }
}
```

The values on each line in the REGION_TYPE section specify the region type, the connected Vdd/Vss nets defined for each region type, and the fraction of the total current consumed by the region. Values on each line in the REGION_BBOXES section specify the instantiations of each region for the IP/Memory LEF_MACRO, and the corner coordinates for the bounding box of each region instantiation.

An extracted memory/IP model using the user-specified region and the associated current distributions are created for RedHawk after import of DEF for each LEF_MACRO specified.

PACKAGE_SPICE_SUBCKT_INFO

Defines the package Spice subcircuit model to be used in simulation. When used, the TCL commands of 'setup pad', 'setup wirebond', and 'setup package' are not in effect. However, if the TCL command 'setup pss' is used, the Spice subcircuit specified in the '-subckt' option replaces the previous value of this keyword. This keyword replaces the previous keyword, PACKAGE_SPICE_SUBCKT, and now allows hierarchical Spice decks.
Optional. Default: None.

Syntax:

```
PACKAGE_SPICE_SUBCKT_INFO {  
    PATH <path_package_Spice_file>  
    ? TOP <top subckt name> ?  
}
```

Note that the PATH option must be specified. If TOP is not specified, then the first subckt is treated as the top subckt.

PAD_FILES

Specifies one or more pad definition files to be used in the design, of format type .pad, .pcell, or .ploc. Supports .pdie and .tsv extensions for 3DIC analysis. DMP compatible. *Optional; .pad, .pcell, or .ploc files can be imported separately through <design>.pad, <design>.pcell, or <design.ploc> files. Default: none.*

Syntax:

```
PAD_FILES {  
    <padFilePathName-1>  
    ...  
    <padFilePathName-n>  
}
```

where

<padFilePathName-1> : the path and filename for the pad file

Example:

```
PAD_FILES  
{  
    MNOPQ.ploc  
    MNOPQ.pad  
    MNOPQ.pcell  
}
```

For a detailed description of the syntax for .ploc, .pad, and .pcell files, see the section "Pad, Power/Ground and I/O Definition Files", page C-741.

PARTIAL_FLAT_SPEF

When set, corrects duplicate SPEF net coverage between top and block SPEF files. *Optional. Default: 0 (off)*

Syntax:

```
PARTIAL_FLAT_SPEF [ 0 | 1 ]
```

PGNET_HONOR_DEF_TYPE

Ignores name-based net type assignments, and honors the DEF net type assignment, regardless of name. When set to 0, a net name that includes the strings 'VDD', 'vdd' 'VCC', 'vcc', 'GND', 'gnd', 'VSS' or 'vss' are considered P/G nets, no matter what type is specified in DEF. Used with keyword HONOR_LEF_PIN_TYPE for handling LEF names. *DMP compatible. Optional.*
Default: 1 (On)

Syntax:

```
PGNET_HONOR_DEF_TYPE [ 0 | 1 ]
```

PRINT_ONE_PLOC_PER_PADINST

When set to 1, RedHawk creates a special *adsRpt/PG_simple.ploc* file, in which each pad instance has just one ploc specified (instead of multiple plocs in the *PG.ploc* file), and includes an RLC declaration for each bump. This file facilitates PSS node hook-up with the PSS *.ploc* file. Users need to manually change the *PG_simple.ploc* file, adding the PSS subckt nodes, and then re-run RedHawk with this modified *ploc* file. *DMP compatible. Optional. Default: 0 (off)*

Syntax:

```
PRINT_ONE_PLOC_PER_PADINST [ 0 | 1 ]
```

RDL_CELL(S)

Redistribution layer geometries for flip chip designs are typically available only in GDS format. To include RDL data in RedHawk analysis, two steps are required:

- a. Conversion of RDL GDS to DEF form with the '*gds2def/gds2rh*' utility
- b. Instantiating this DEF from the top level design

When the RDL_CELL(S) keyword is correctly specified, RedHawk automatically reads in one or more RDL master cell DEFs (after importing top DEF), creates the RDL instance(s), places them according to the LOCATION specification, and connects all pins listed to top level nets. Note that all RDL-related information is needed only in the RDL_CELL keyword specification. If there are multiple definitions for OFFSET, LOCATION, and ORIENTATION, the last values override previous ones. Note that undefined child instances are ignored. *Required for RDL cells. Default: none.*

To correctly use this GSR keyword, use the following guidelines:

- Prepare the RDL master cell DEF
- Make sure all fields in RDL_CELL are correctly specified, including the master cell file path, and the instance LOCATION relative to the top DEF coordinate system.
- The RDL DEF file path is not needed in DEF_FILES in the GSR.
- The RDL instance is not needed in a GDS_CELLS definition, nor as a component of the top DEF.

Syntax

```
RDL_CELL(S) {
    <inst_name1> <DEF_master_cell_name1> <path_RDL_master_DEF_file>
    LOCATION <x_loc> <y_loc>
    ? OFFSET <x_offset> <y_offset>
    ? ORIENTATION <orientation> <dummy> ?
    NET <Vdd_net_name> <power_pin_name>
    NET <Vss_net_name> <ground_pin_name>
```

```

    ...
    NEW_INSTANCE NEW_MASTER NEW_PATH
    <inst2> <mastercell2> <DEF_path2>

        <inst_nameB> <DEF_master_cell_nameB> <path_RDL_master_DEF_file>
        LOCATION <x_loc> <y_loc>
        ? OFFSET <x_offset> <y_offset>
        ? ORIENTATION <orientation> <dummy> ?
        NET <Vdd_net_name> <power_pin_name>
        NET <Vss_net_name> <ground_pin_name>
        ...
    NEW_INSTANCE NEW_MASTER NEW_PATH
    <instC> <mastercellC> <DEF_path2>
    ...
}
```

where

<inst_name>: instance name of RDL cell
 <DEF_master_cell_name>: RDL master cell name
 <path_RDL_master_DEF_file> : path to RDL master cell defined in DEF
 LOCATION <x_loc> <y_loc>: x,y location relative to top DEF coordinate system
 (microns)
 OFFSET : specifies x,y offset distance relative to top DEF coordinate system
 Default: 0,0. (microns)
 ORIENTATION : orientation specification using the same nomenclature as in the
 top DEF. Default: N. A third entry is required on this line that is ignored
 (dummy).
 NET <Vdd/Vss_net_name> <power/ground_pin_name>: describe the RDL
 instance's pin connections to the top level nets. Each line describes only one
 pin. Multiple such lines may occur in the block.
 NEW_INSTANCE, NEW_MASTER, NEW_PATH: after the first cell definition,
 subsequent different RDL cells can also be defined, using the same syntax
 as for a single RDL cell.

Example

```

RDL_CELL {
    rdl_inst chip_rdl /home/proj/data/def/chip_rdl.def
    LOCATION 500 250
    OFFSET -25 -25
    ORIENTATION FS dummy
    NET VDD VDD
    NET VSS VSS
    NEW_INSTANCE NEW_MASTER NEW_PATH
    rdl_inst_A chip_rdl_C /home/proj/data/def/chip_rdl.def
}
```

READ_LEF_OBS

When this keyword is turned On, RedHawk reads the specified parts of the “OBS” section in LEF to determine the regions to be assigned power for early stage analysis. The only parts of the OBS data read in are LAYERs defined as OVERLAP

and shapes defined as RECT. All other items defined in the LEF OBS section are ignored. *Optional. Default: off, 0*

Syntax:

```
READ_LEF_OBS 1
```

Example: OBS section items in LEF included in power assignment:

```
OBS
    LAYER OVERLAP ;
    RECT 0.0 0.0 800.0 400.0 ;
    RECT 0.0 400.0 400.0 600.0 ;
END
```

REMOVE_PARENTLEF_GEOS

When set, drops the LEF of cells from gds2def/gds2rh blocks--the parent (original) LEF. *Optional. Default: off, 0.*

Syntax

```
REMOVE_PARENTLEF_GEOS [ 0 | 1 ]
```

REVERSE_DEF_READ_ORDER

By default the top DEFs are imported first, and then a block DEF is only imported if its parent DEF is already imported. In this way RedHawk avoids reading unneeded block DEFs that are not in the design, speeding up processing. When set to zero, all block DEFs are imported. *Optional. Default: 1 (On).*

Syntax

```
REVERSE_DEF_READ_ORDER [ 0 | 1 ]
```

STA_CRITICAL_PATH_FILE

To perform fixing and optimization for timing of instances on critical paths in a design ('FAO_OBJ PATH'), you must specify the critical path report using this keyword. *Optional. Default: none.*

Syntax

```
STA_CRITICAL_PATH_FILE
{
    <critical path report path>
}
```

Example

```
STA_CRITICAL_PATH_FILE
{
    RedHawk/STA/TYP_S_INTERNALS.path
}
```

STA_FILES

If the keyword is defined, the power for each net is calculated from the transition times ("slew") of each instance's input/output pins, using the clock frequency domain for all the instances that are specified in the STA output file, instead of deriving its value from the CLOCK_ROOTS keyword.

For instances not defined in the STA output file or VCD file, the frequency of the instance is set to zero. Use the frequency specified by FREQ_OF_MISSING_INSTANCES keyword for those instances that are not

covered in the STA file. FREQ_OF_MISSING_INSTANCES is not recommended to be specified unless you know that STA does not cover the design well. If the EXTRACT_CLOCK_NETWORK option is specified as 1 (default 0), clockroots defined in the clock section of the STA output file are used for clockroot tracing, instead of the STA-identified clock instances.

Note: If both BLOCK_STA_FILES and STA_FILE are used, then BLOCK_STA_FILES is ignored, and if a USER_STA_FILE is also used, any common data in the specified USER_STA_FILE overrides data in either of the other files.

The STA output file is generated from STA analysis. See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#) for details of how to generate the STA output file. *DMP compatible. Required for dynamic analysis; Optional for static analysis.*

Syntax:

```
STA_FILES {
    FREQ_OF_MISSING_INSTANCES <frequency in hertz>
    APPLY_FREQ_OF_MISSING_INSTANCES_TO_CLOCK_PINS [0|1]
    EXTRACT_CLOCK_NETWORK [ 0 | 1 | 2 ]
    <top_level_block_name> <sta_output_file>
}
```

where

FREQ_OF_MISSING_INSTANCES <frequency in hertz> : specifies the operating frequency for instances not in the STA file but in the design. Default: 0.

APPLY_FREQ_OF_MISSING_INSTANCES_TO_CLOCK_PINS : when set to 1 (default 0), both the signal pin min/max arrival times and clock min/max arrival times of the missing instances are assigned, according to the specified FREQ_OF_MISSING_INSTANCES.

EXTRACT_CLOCK_NETWORK : when set to 1, specifies that the clock network is extracted from clock roots defined in STA. When set to 2, the clock domain for an instance is updated using the Clock Roots defined in the Clock section of the STA file provided. Setting to 2 also supports the clock names in STA where pin delimiter is different from RedHawk's default pin delimiter.

<top_level_block_name>: the top level block name of the chip

<sta_output_file> : the file generated from STA analysis. See [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#) for details of how to generate the STA output file.

Example:

```
STA_FILES
{
    top_level_block block1.sta
}
```

STANDARD_CELL_HEIGHT

Specifies allowable heights for all standard cells in the design (microns). By default RedHawk identifies standard cell heights from the heights defined in the ROWS section in DEF. When both the ROWS section in DEF and the keyword STANDARD_CELL_HEIGHT are specified, the keyword specification takes priority. If neither the keyword nor the ROWS section is specified, RedHawk uses a statistical approach to identify the standard cell heights in the design. To disable standard cell height identification from DEF, you can set the GSR keyword “USE_DEF_ROW_HEIGHT 0”. *DMP compatible. Optional. Default: none*

Syntax

```
STANDARD_CELL_HEIGHT {
    <cell_height1>
    <cell_height2>
    ...
}
```

TECH_CONNECTIVITY_FILE

Specifies a file containing LVS-style connectivity statements of the form:

```
CONNECT <layer1> <layer2> ?BY <via>?
...
```

RedHawk uses this information, along with the layer heights and thicknesses, to block connections between all layers that touch or overlap vertically, but have no CONNECT statement saying they should be connected during extraction. The effect is identical to the *ircx2tech* ircx-to-tech file conversion flow with LVS connection file information input to *ircx2tech* using the ‘-ct <LVS_connect_Info_File>’ option, except that there is no facility for mapping the LVS names to RedHawk names. All names in the TECH_CONNECTIVITY_FILE are assumed to be tech file names. This allows foundries to release the LVS connectivity information separately from the core tech file. *Optional; default: none.*

```
TECH_CONNECTIVITY_FILE <filename>
```

TECH_FILES

Required to specify the RedHawk technology file (*.tech) to be used in the design. *DMP compatible. Required. Default: none.*

Note: the tech file should be imported directly using this keyword and not in a separate ‘import’ command.

Syntax:

```
TECH_FILES <techFilePathName>.tech
```

Example:

```
TECH_FILES abcDesign4.tech
```

TEMPERATURE

Specifies global design-dependent operating temperature in degrees C. Nominal temperature is the process-related, design-independent value specified in the Tech file. This TEMPERATURE specification overrides the value of T in the Tech file for resistance calculations, but is overridden by the layer-based keyword TEMPERATURES. For temperature-setting in EM calculation, see section “Temperature Setting for Power EM Calculation”, page 15-418. *DMP compatible. Optional. Default: none.*

Syntax:

```
TEMPERATURE <actual_temperature_°C>
```

Example:

```
Temperature 100
```

TEMPERATURES

Specifies layer-based design-dependent operating temperatures in degrees C. The temperature-setting priorities for R extraction are:

1. TEMPERATURES - layer-based GSR keyword

2. TEMPERATURE - global GSR keyword
3. T - in the tech file
4. TNOM - in the tech file

For temperature-setting in EM calculation, see [section "Temperature Setting for Power EM Calculation", page 15-418](#). *Optional. Default: none.*

Syntax:

```
TEMPERATURES {  
    <layer_1> <temp_1 °C>  
    ...  
    <layer_n> <temp_n °C>  
}
```

Example:

```
TEMPERATURES {  
    M1 111  
    M2 112  
    M3 113  
    M4 114  
    ...  
}
```

THERMAL_PROFILE

With the THERMAL_PROFILE keyword and the 'perform pwrcalc' command, the chip thermal model file is generated for thermal analysis, which maps tile and layer-based temperatures in the thermal profile to instances in the design, with the following format:

```
<inst name> <Tavg_C> <Tmax_C> <Tmin_C>
```

If the instance is in the intersection of the grid defined in <thermal_profile_file>, the max temperature is used from the intersected grid. The wire/via temperature values are based on the per layer tile-based thermal profile file generated by Sentinel. The EM limit is modified by the temperature data specified in the tech file keyword EM_TEMP_RATING (*only affects Average EM limits, and not RMS and PEAK limits*). *Optional; default: none.*

Syntax:

```
THERMAL_PROFILE {  
    FILE <thermal_profile_file>  
}
```

where the <thermal_profile_file> is from Sentinel-TI package modeling.

USE_DEF_ROW_HEIGHT

When set to 1 (default), selects standard cells with heights up to 4x normal height from the ROWS section in DEF. Setting to 0 disables DEF row height selection.
DMP compatible. Optional. Default: 1.

Syntax

```
USE_DEF_ROW_HEIGHT [ 0 | 1 ]
```

USE_DEF_VIAMODEL

When set to 0, ignores DEF VIA model when there is conflict with TECHLEF VIA model. *Default: 1.*

Syntax:

```
USE_DEF_VIAMODEL [ 0 | 1 ]
```

USE_DEF_VIARULE

If there is a via cut size mismatch between “DEF viamodel<model_name>” and “LEF viarule<VIAGENTOP>”, set the GSR keyword USE_DEF_VIARULE to honor the via cut size in DEF viamodel. *Default: 0.*

Syntax:

```
USE_DEF_VIARULE [ 0 | 1 ]
```

USE_LEF_FOR_LOGICAL_CONNECTION

Specifies SPEF-based flow model cells as white box cells in the GDS2RH+SPEF flow. These cells are by default modeled as gray box cells (GRAY_BOX_CELL 1). *Optional. Default: gray.*

Syntax

```
USE_LEF_FOR_LOGICAL_CONNECTION {  
    <cellname1> white  
    <cellname2> white  
    ...  
}
```

USER_STA_FILE

USER_STA_FILE allows you to specify the essential timing and slew data needed by RedHawk, such as instance timing windows defined by switching times, and whether the instance is part of a signal or clock network. Data in this file supersedes the same data in the regular STA file, so it can be easily used for “what-if” analysis during power ramp-up analysis. For example, if you want to incrementally set the timing window for instances, use the following steps:

```
gsr set USER_STA_FILE <file name>  
import sta
```

USER_STA_FILE is intended to be a much more understandable format and syntax to use than the normal STA file.

Note: If BLOCK_STA_FILES or STA_FILES is also used, any common data in the specified USER_STA_FILE overrides data in the other files. *DMP compatible.*
Optional. Default: none.

Syntax

```
USER_STA_FILE <filename>
```

The allowed format of information in the specified USER_STA file is different depending on whether a signal/clock net or a switch is specified, as follows

- Signal or clock timing window

```
<inst_name>/<pin_name> TW <min_arriv_time> <max_arriv_time> ?<Freq>? ?[s|c]?
```

- Switch timing window

```
<inst_name/><pin_name> SW <min_arriv_time> <max_arriv_time> ?<Freq>?
```

Note: the “/” is required to specify pin name

- Slew

```
<inst_name> SL <rise_trans_time> <fall_trans_time>
```

```
<inst_name>/<pin_name> SL <rise_trans_time> <fall_trans_time>
```

```
...
```

where

- <inst_name>: name of instance
- <inst_name>/<pin_name> : specifies a pin_name if allowed. Transition times are applied to that pin; otherwise data is applied to the instance
- TW/SL/SW: identifies a Timing Window, Slew, or Switch data line
- <min_arriv_time> <max_arriv_time>: specifies minimum and maximum arrival times in seconds
- <Freq> : (optional) operating frequency of instance.
 If <Freq> is *not* specified:
 - (1) if the instance is in the STA file, use frequency defined there.
 - (2) if the instance is not covered in STA, or the frequency is not given, use the FREQ_OF_MISSING_INSTANCES value defined in STA_FILES section in GSR. If FREQ_OF_MISSING_INSTANCES value is not specified, the frequency is assumed to be 0.
- [s | c] : optional specification of signal type, signal/data or clock
- <rise_trans_time> <fall_trans_time> : specifies rising and falling transition times in seconds

Example of User STA file contents:

```
top/block/inst1      SL 0.1e-9 0.1e-9
top/block/inst1      TW 4e-9 4e-9 100e6 c
top/block/inst1/CLK SL 0.01e-9 0.01e-9 100e6
top/block/inst2      TW 5.1e-9 5.2e-9 200e6
top/power_gate/SW1/E SW 1.0e-9 1.0e-9
```

USE_SIGNAL_LOAD_FROM_STA

When set to 1, and signal load information (that is, C1-R-C2 data) is available in the STA file, RedHawk reads it from the STA file and skips SPEF reading. This load information is available in ATE-generated STA files (see [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#)). If set to 0 (default), or if there is no signal load data in the STA file, or if there is no STA file, then the SPEF data specified in the GSR is used. The STA file header indicates whether or not the file contains signal load data. Note that the STA file should be imported using the STA_FILES keyword and not BLOCK_STA_FILES. *DMP compatible. Optional; default: 0.*

Syntax:

```
USE_SIGNAL_LOAD_FROM_STA [ 0 | 1 ]
```

VCD_FILE

The VCD_FILE keyword reads in original VCD files directly for power calculation purposes. Note that instance switching specified in the GSC file overrides the VCD file. *DMP compatible. Optional. Default: None.*

Syntax:

```
VCD_FILE {
    <top_block_name> <VCD file path>
    ? FILE_TYPE [ VCD | FSDB | RTL_VCD | RTL_FSDB ]?
    ? DUMP_SAIF_FILE <filename>
    ? VCD_DRIVEN [0|1]?
    ? FRONT_PATH <"string"> ?
    ? SUBSTITUTE_PATH <"string"> ?
```

```

? SELECT_RANGE <start_time> <end_time> ?
? SELECT_TYPE [ WORST_POWER_CYCLE | WORST_DPDT_CYCLE | 
    [WORST_TILE_POWER_CYCLE | WORST_TILE_DPDT_CYCLE ]?
? START_TIME <start> ? ? END_TIME <end> ?
? TRUE_TIME [0|1] ?
? MAPPING <map_file> ?
? PROPAGATE 2 ?
? STA_VCD_FREQ_RATIO <Ratio> ?
}

```

where

<top_block_name> : specifies name of top block

<VCD filepath> : specifies path to VCD file

FILE_TYPE [VCD | FSDB | RTL_VCD | RTL_FSDB] : selects file type (default: VCD)

DUMP_SAIF_FILE : creates a specified SAIF file reflecting activities at all nets in the design, after Event Propagation. The SAIF file not only includes the instances covered in VCD/FSDB, but also includes the instances that are propagated.

VCD_DRIVEN : turns VCD-driven state propagation based power calculation on and off (default off). When set, RTL VCD State Propagation honors the END_TIME option (default 0). If END_TIME is not specified, PowerStream uses the VCD end time. Also, when set along with Gate VCD/FSDB, RedHawk performs average toggle rate based processing wherein, only toggle rates are taken from VCD/FSDB rather than events. This is more suited for static analysis where power value rather than VCD events are passed to simulation. This can improve performance avoiding performance/capacity bottlenecks when the VCD/FSDB duration is very large.

FRONT_PATH "<redundant_path_string>" : specifies the string describing the VCD file hierarchical path of the instance. This string is then replaced by a substitute path string in order to match the path of the instance in the DEF.

SUBSTITUTE_PATH "<substitute_path_string>": specifies the string describing the DEF file hierarchical path of the instance. This option can be used internally to substitute the VCD path (specified by FRONT_PATH) with the DEF path. (This option is not required for RedHawk to properly identify the equivalent instances and nets in the VCD and DEF. For top level VCD, this is automatically set to " ", and for block levels, to the block name).

SELECT_RANGE: specifies the start_time and end_time for performing automatic critical cycle selection during power calculation. If START_TIME and END_TIME values are set to -1, the full VCD period is included in cycle selection.

SELECT_TYPE : default value WORST_POWER_CYCLE ranks cycles based on highest cycle power. Set to WORST_DPDT_CYCLE invokes DPDT (delta power/delta time) ranking of cycle selection, since large cycle-to-cycle changes in power level (high DPDT) may cause very high di/dt values, leading to large dynamic voltage drops. WORST_TILE_POWER_CYCLE, and WORST_TILE_DPDT_CYCLE select tile-based cycle selection, which can be modified by BBox exclusion areas using the GSR keyword POWER_CYCLE_SELECT_BLACK_BOX or by White-box inclusion areas using POWER_CYCLE_SELECT_WHITE_BOX GSR keyword.

START_TIME <time> : specifies VCD or FSDB file start time for power calculation. If both START_TIME and SELECT_RANGE are specified, SELECT_RANGE is ignored, so cycle selection is skipped. Default start time

= 0. If the specified START_TIME is within the dump-off range (where activity is not considered), the START_TIME is automatically reset to the start of the next dump-on time.

END_TIME <time> : specifies VCD or FSDB file end time for VCD-driven state propagation-based power calculation. For other flows, END_TIME is ignored, and it is determined by START_TIME + DYNAMIC_SIMULATION_TIME. Default end time is the end of the VCD/FSDB file.

TRUE_TIME : if set to 0, uses STA timing data and assumes no glitches; if set to 1, uses VCD switching and timing data; default = 0.

MAPPING <filename> : name of map file with RTL VCD instance names and corresponding DEF instance name

PROPAGATE : When set to 2, evaluates switching activities of the instances missing from true time gate VCDs (SDF back-annotated) by propagating from upstream. If found to be switching, the timing window for switching is taken from the STA/timing file. The propagated instances and their details are reported in the file *adsRpt/propagated_insts.gz*. Also, the file *adsRpt/unpropagated_insts.gz* holds instances that could not be propagated.

STA_VCD_FREQ_RATIO : matches the VCD frequency to the STA frequency (the specified <ratio> = STA_freq/VCD_freq). If this option is used, VCD data is processed according to the STA frequency and the specified ratio. If **STA_VCD_FREQ_RATIO > 1**, VCD is contracted to make it faster.

STA_VCD_FREQ_RATIO < 1, VCD is expanded to make it slower. Note that this option does not affect the frequency selection in simulation.

Example:

```
VCD_FILE
{
    top top.vcd
    FILE_TYPE FSDB
    FRONT_PATH "test/setup/"
    SELECT_RANGE 230724203 285724203
    TRUE_TIME 1
    STA_VCD_FREQ_RATIO 2.0
}
```

VCD_CONSISTENT_SCENARIO_FILTER

VCD Parsing Stage will have consistent results between power calculation and simulation when this GSR keyword is ON along with 'TRUE_TIME 1'. *Default: off for averaged mode*

Syntax:

```
VCD_CONSISTENT_SCENARIO_FILTER [ 0 | 1 ]
```

Parameter Keywords

APACHE_DB_OVERWRITE

Specifies whether the previous version of the Apache database is overwritten by new analysis data or not. *DMP compatible. Optional. Default: 1 (overwrite).*

Syntax:

```
APACHE_DB_OVERWRITE [ 1 | 0 ]
```

APACHE_FILES

Specifies the amount of internal backup files to be saved on exit from a RedHawk run. Values are:

- normal - some files removed
- clean - most files removed

DMP compatible. Optional. Default: normal.

Syntax:

```
APACHE_FILES [ normal | clean ]
```

Example:

```
APACHE_FILES clean
```

CACHE_DIRECTORY

Specifies the directory where data is cached. Should be a disk on the machine where RedHawk is running for efficient operation. *DMP compatible. Optional. Default: current working directory.*

Syntax:

```
CACHE_DIR <path to cache directory>
```

Example:

```
CACHE_DIR /local/cache
```

CACHE_MODE

Enables adaptive disk caching in RedHawk. Using this “smart” disk caching is recommended whenever a run requires more than the available physical memory, to reduce the need for generic system memory swapping and to make memory use much more efficient. Make sure that there is local disk space to use for disk caching. *DMP compatible. Optional. Default: 0 (off).*

Syntax:

```
CACHE_MODE [ 0 | 1 ]
```

DEF_PG_NETS_FILE

Allows clock and signal nets (defined in SPECIALNETS in DEF) to be defined as power/ground nets for purposes of EM analysis, using a file specified with this keyword. The specified file should use the following format:

```
<design> <net> [ power| ground| signal |clock ]
```

For example, an entry in the file would be:

```
blockABC2 clockAB power
```

In this example, net ‘clockABC’ is treated as a power net for EM analysis. This keyword also can be used for ESD analysis to run current density checks for bump-to-bump checks from pad-Vdd and pad-Vss. *DMP compatible. Optional. Default: none.*

Syntax:

```
DEF_PG_NETS_FILE <filename>
```

DEF_SCALING_FACTOR

Specifies the scaling factor to be applied to the length units used in each .def file. This allows shrinking a design DB from an older process technology (e.g., 0.15um) to a newer process technology (e.g., 0.11um).

For the top cell, you should specify the file path name of the top DEF similar to specifying blocks. The keyword option 'all' is valid to specify all blocks should be scaled, including the top cell.

Note: If 'All' or the top cell scaling factor is not specified, make sure that the *.ploc* locations correspond to the modified-dimension layout. Otherwise, you may get an ERROR message that the net has no driver, and the simulation may fail.

DMP compatible. Optional. Default: 1.0.

Syntax:

```
DEF_SCALING_FACTOR {
    [ All <scale_factor> |
      <DEF_top_block_path-1> <scale_factor>
      ...
      <DEF_top_block_path-n> <scale_factor> ]
    IGNORE_FOR_PLOC [0|1]
}
```

where

All <value> : specifies the length scaling factor applied to all DEF files.

<DEF_top_block_path-n> <value> : specifies the DEF top block file path and the length scaling factor to use on the file.

IGNORE_FOR_PLOC: when set to 1 (default is 0), the x,y coordinates in the PAD_FILE are *not affected* by the DEF_SCALING_FACTOR setting.

Examples:

```
DEF_SCALING_FACTOR
{
    INVERT_ABC 0.85
    all 0.75
}

DEF_SCALING_FACTOR
{
    user_data/LEFDEF/INST-Y0.def 0.75
    IGNORE_FOR_PLOC 1
}
```

EVA_PG_AWARE

Allows consideration of anomalous switching due to power and ground design weaknesses during RedHawk vectorless DvD analysis. *DMP compatible. Optional. Default: 0 (off, do not consider P/G weakness).*

Syntax:

```
EVA_PG_AWARE [ 1 | 0 ] ?<PG_switch_ratio>?
```

where

<PG_switch_ratio> : optionally specifies the ratio defined as: the maximum number of instances that switch due to P/G weakness divided by the total number of instances in the design. For example, if the ratio is set to 0.2, and TOGGLE_RATE is set to 0.0, not more than 20% of the total number of instances switch, all due to the ranking of P/G weakness. It is recommended that TOGGLE_RATE be set to 0.0 when EVA_PG_AWARE is set to 1 to focus on just the switching due to P/G weakness. (Default ratio: 0.0009)

Example:

```
EVA_PG_AWARE 1 0.2
```

FREQUENCY

Defines the dominant operating frequency on the chip, or the lowest frequency that includes a majority of the power consumption on the chip. Also this keyword provides a frequency for Cycle Selection when all instances in the design have zero frequency defined in the STA file.

For a design in which several frequencies consume significant power, the frequency to be specified is the frequency at which more than 10% of the chip power is consumed at *lower* frequencies. For example, assume that there are three significant frequencies on the chip, and they consume the following power: 100 MHz (5mw), 200 MHz (20mw), and 400 Mz (70mw). The FREQUENCY value to be specified in this case would be 200 MHz, even though a majority of the power is consumed at 400 MHz. *DMP compatible. Required. Default: none.*

Syntax:

```
FREQUENCY <value in Hertz>
```

Example:

```
FREQUENCY 160e6 (or 160M, 160me)
```

Note that 'me' represents mega.

GENERATE_CPM

When set, invokes a special flow for CPM generation, which is distinct from the dynamic voltage drop analysis flow. **This keyword must be set for CPM generation.** *DMP compatible. Optional. Default: 0.*

Syntax:

```
GENERATE_CPM [ 0 | 1 ]
```

GND_NETS

Specifies the voltage for Vss nets, and also defines equivalent ground domain net names in DEF, which are merged in RedHawk. The "*" character is acceptable as a wildcard in net names. The ground domain names are defined in the DEF file using the SPECIALNETS keyword. The expanded list of nets is recorded in the file *adsRpt/vdd_gnd_nets.rpt*. *DMP compatible.*

Optional. Default: all the SPECIALNETS are designated as USE GROUND and set to 0 volts.

Note: A net can only be merged into one domain (the first one specified). There can be no net hierarchy.

Syntax:

```
GND_NETS
<Vss_domain_net_name> <value_Volts> {
    <equiv_Vss_net_name1>
    ...
}
```

where

<Vss_domain_net_name> : specifies DEF name for ground domain net, such as VSS for the core power domain.

<value_Volts> : nominal voltage

<equiv_Vss_net_name1> : equivalent ground net name in same domain

Example:

```
GND_NETS
{
    GND 0
    VSS 0
}
```

LEF_SCALING_FACTOR

Specifies the scaling factor to be applied to the length units used in the .lef file. This is for shrinking the design DB from an older process technology (e.g., 0.15um) to a newer process technology (e.g., 0.11um). *DMP compatible. Optional. Default: 1.0.*

Syntax:

```
LEF_SCALING_FACTOR
{
    [ All <value> |
    <LEF_FilePathName-1> <value>
    ...
    <LEF_FilePathName-n> <value> ]
}
```

where

All <value> : specifies the length scaling factor applied to all the LEF files.

<LEF_FilePathName-1> <value> : specifies the path and filename of the LEF file and the length scaling factor to use on the design.

Example:

```
LEF_SCALING_FACTOR
{
    lefs/MNOPQ.lef 0.8
}
```

LICENSE_WAIT

Allows a RedHawk run that does not have an available RedHawk license to wait until one is available, instead of exiting. This is useful after an initial static analysis run, when another special RedHawk license is needed in the session and the option '-lmwait' was not used. *DMP compatible. Optional. Default: 0 (off).*

Note: The process is controlled by the license manager and hence appears to be hung while in waiting mode. If -lmwait is used, a wait time limit may be specified.

Syntax:

```
LICENSE_WAIT [ 0 | 1 ]
```

MMX_RES_MAP_LIMIT

The **Transistor Pin Actual Resistance Map** menu command generates a color map of the actual resistance for all transistor pins in the MMX analysis. For run-time reasons only the highest 10,000 pin resistances are calculated by default, based on P/G grid weakness computations. To change the maximum number of resistances calculated, use this keyword. *Optional. Default: 10000.*

Syntax:

```
MMX_RES_MAP_LIMIT <pin_limit>
```

MULTI_GND_PACKAGE_MODEL

When set, connects each ground domain through a package resistor to ideal ground independently,. That is, ground domains are not connected together first. By default, multi-ground domain designs have all ground domains connected together, and then connect through a package resistor to the ideal ground. *Optional. Default: 0.*

```
MULTI_GND_PACKAGE_MODEL [0|1]
```

MULTI_THREADS

Controls the number of threads that RedHawk uses, which can provide a significant reduction in run time in power calculation and simulation by using multiple CPUs, compared to single-CPU use. By default all available CPUs are used. MPR network reduction and ERV model generation are supported, as well as Import LEF, DEF, SPEF, STA, and SIR (save/reload) operations. *DMP compatible. Optional. Default: 1.*

Syntax:

```
MULTI_THREADS [On=1|Off=0] ?<max_num_threads>?
```

Examples:

```
MULTI_THREADS 1 (uses all available CPUs - default)  
MULTI_THREADS 0 (uses single-thread)  
MULTI_THREADS 1 1 (uses single-thread)  
MULTI_THREADS 1 4 (uses multi-thread with four CPUs)
```

PGPLOC_DEBUG

Generates domain information for bumps in the PLOC file. You can get power/ground domain names for each bump in RedHawk-generated PLOC file, *adsRpt/PG.ploc*. Also displays RLC data for each bump in the *adsRpt/PG_simple.ploc* file if the GSR keyword *PRINT_ONE_PER_PADINST 1* is used. *DMP compatible. Optional. Default: 0 (Off).*

Syntax:

```
PGPLOC_DEBUG [ 0 | 1 ]
```

PRINT_EM_VIA_BOX

Specifies that via_cut_bounding_box_coords and current_direction be included in the *.em report of EM and current data, as requested using the *EM_DUMP_PERCENTAGE* GSR keyword (see details on page C-657). *DMP compatible. Optional. Default: 0.*

Syntax:

```
PRINT_EM_VIA_BOX [ 0 | 1 ]
```

PRINT_EM_VIA_INFO

When set, prints complete via EM information in the EM worst file. *DMP compatible. Optional. Default: 0.*

Syntax:

```
PRINT_EM_VIA_INFO [ 0 | 1 ]
```

REPORT_DISCONNECT_MIN_LENGTH

Specifies the minimum wire length above which disconnected-wire reports are created. When set, the number of reported disconnects can be reduced. *Optional.*
Default: 0.

Syntax:

```
REPORT_DISCONNECT_MIN_LENGTH <min_length_microns>
```

REPORT_FLATTEN_LOG

When set, records debug data in the flatten stage in the file *adsRpt/flatten.rpt*. Logical connectivity issues can cause certain nets to be merged and incorrectly renamed. To resolve such cases, you can use this keyword to create a file that lists all the nets that are merged and renamed when the design is flattened. *DMP compatible.* *Optional.* *Default:* 0.

Syntax:

```
REPORT_FLATTEN_LOG [0 | 1]
```

REPORT_PEAK_MEMORY

When set, reports peak memory usage for each stage in the log file. By default memory use at the end of each stage is reported. *DMP compatible.* *Optional.*
Default: Off.

Syntax:

```
REPORT_PEAK_MEMORY [0 | 1]
```

REPORT_REDUCTION

This keyword controls the number of output reports generated, their compression, and the contents of the *adsRpt* folder, depending on the value set:

off/ 0: complete information (default)

normal/ 1: files that can be created from the database using the 'report' TCL command (currently only files *adsRpt/*.power.rpt*, *adsRpt/Dynamic/<>.dvd.mmx*, and *adsRpt/Static/<>.ir.mmx*) are *not* generated. To generate the *.power.rpt file, use the command 'report power -all ?-o <output_file>?'. You can invoke on the fly the following commands to get reports *adsRpt/Static/<>.ir.mmx* and *adsRpt/Dynamic/<>.dvd.mmx*:

after static analysis, use the Tcl command:

```
report ir -mmxpin ?-o <file>? ?-limit <num_per_domain>?
```

after dynamic analysis, use

```
report dvd -mmxpin ?-o <file>? ?-limit <num_per_domain>?
```

If '-o <file>' is not specified, the default output filename is used.

In addition, *adsRpt/Dynamic/*.dvd** files are not created when set to 'normal', but a *dvd histogram is plotted, and the *dvd* report files can be created using the following Tcl commands:

For *.dvd files, use: report dvd -instance <-output file>, where the default file name is *adsRpt/Dynamic/<design_name>.dvd*.

For .dvd.pin files, use report dvd -instance -pin <-output file>, where the default file name is *adsRpt/Dynamic/<design_name>.dvd.pin*.

For .dvd.arc files, use report dvd -instance -arc <-output file>, where the default file name is *adsRpt/Dynamic/<design_name>.dvd.arc*.

For *mcyc_effvdd.rpt* files, use report dvd -instance -mcyc <-output file>, if related GSR DYNAMIC_REPORT_DVD, or Tcl '-mcyc' is set. The default file name is *adsRpt/Dynamic/mcyc_effvdd.rpt*.

Note that a *dvd histogram is plotted, even if the *dvd.gz file is unzipped and "REPORT_REDUCTION" is set

max (maximum file reduction)/ 2: same as 'normal', and in addition, files that cannot be retrieved from the database (*adsRpt/Static/<design>.inst.arc*, *adsRpt/Static/<design>.inst.pin*), and simulation-related output files, are gzipped. PowerStream imports either gzipped or non-gz power files.

Note that power reports *power_summary.rpt*, *.power.rpt in *adsRpt*, and the "Power Summary" section in the log file in the "import power" flow are generated regardless of the REPORT_REDUCTION setting. *DMP compatible. Optional. Default: Off.*

Syntax:

REPORT_REDUCTION [off/0 | normal/1 | max/2]

REPORT_PEAK_MEMORY

When set, reports peak memory usage for each stage in the log file. By default memory use at the end of each stage is reported. *Optional. Default: Off.*

Syntax:

REPORT_PEAK_MEMORY [0 | 1]

SPLIT_VDD_EXTRACT_LP

SPLIT_VDD_EXTRACT_LP_FSIZE

For the 'plot voltage' command, if the number of nodes is greater than 5M , when the 'plot voltage' command is executed the first time, the vdd file is split into multiple vdd files, with 1M nodes in each vdd file. From second time the command is executed onwards, the 'plot voltage' command reads the split vdd files. The default 5M node threshold at which point the files are split can be adjusted using this keyword. *Optional. Default: 5 M.*

Syntax:

SPLIT_VDD_EXTRACT_LP <value>

The number of nodes in each split vdd file can be adjusted using the keyword 'SPLIT_VDD_EXTRACT_LP_FSIZE <value>'. *Optional. Default: 1 M.*

Syntax:

SPLIT_VDD_EXTRACT_LP_FSIZE <value>

STATIC_CONNECT_INST_FLOAT_GND

When set, excludes instances with floating ground pins from static analysis. By default, instances with floating grounds are a part of static simulation. You may set this keyword to 1 to exclude them from analysis. Such instances then are removed from output reports and in IR maps they are shown in red. *Optional. Default: 0.*

Syntax:

STATIC_CONNECT_INST_FLOAT_GND [0 | 1]

STD_CELL_SINGLE_NOMINAL_VOLTAGE_ONLY

If set to 1, checks the value of the APL nominal voltage keyword VDD against the one or more voltage values specified in the GSR keyword VDD_NETS. All standard cells have only one nominal voltage. Any discrepancy between voltage values

specified in VDD_NETS and APL nominal voltage values are flagged with an Error and APL importing is halted. *Optional. Default: 0.*

Syntax:

```
STD_CELL_SINGLE_NOMINAL_VOLTAGE_ONLY [ 0 | 1 ]
```

TEMPERATURE_DEVICE

If TEMPERATURE_DEVICE is set, the value of the APL configuration file keyword 'TEMP' or the value of the ./lib parameter 'nom_temperature' is checked against the value of this keyword for all devices. The <T> value specifies the temperature (degrees C), and <delta> specifies the allowable tolerance. If the temperature in the ./lib/APL file is out of the specified range (T-<delta>, T+<delta>), importing the ./lib/APL file is rejected with an error message. The delta value is optional; its default value is 5 degrees. *DMP compatible. Required.*

Syntax:

```
TEMPERATURE_DEVICE <T> ? <delta> ?
```

THERMAL_TEMP_RANGE

When specified, checks that temperatures specified in the APL leakage file fall within the temperature range setting (degrees Centigrade), and issues a warning message if not. *Optional. Default: none.*

Syntax:

```
THERMAL_TEMP_RANGE <min_temp> <max_temp>
```

VDD_NETS

Specifies the voltage for Vdd nets, and also defines equivalent power domain net names in DEF, which are merged in RedHawk. The “*” character is acceptable as a wildcard in net names. The power domain names are defined in the DEF file using the SPECIALNETS keyword. The expanded list of nets is recorded in the file *adsRpt/vdd_gnd_nets.rpt*. *DMP compatible. Required. Default: none.*

Note: A net can only be merged into one domain (the first one specified). There can be no net hierarchy.

Syntax:

```
VDD_NETS {
    <Vdd_domain_net_name> <value_Volts> {
        <equiv_Vdd_net_name1>
        ...
    }
    ...
}
```

where

<Vdd_domain_net_name> : specifies DEF name for power domain net, such as VDD for the core power domain, and VDDQ for the I/O power ring.

<value_Volts> : nominal voltage

<equiv_Vdd_net_name1> : equivalent power net name in same domain

Example:

```
VDD_NETS {
    VDD 1.1 {
        VDD1CORE
    }
}
```

```
VDDQ 3.3 {
    VDDQ1
}
```

VIA_IR_REPORT

When turned on, generates a Via Voltage Drop and Current report for both static and dynamic analysis. The report contains voltage drop and current values through the vias, up to a limit of 1000 lines. Also, you can use the TCL command 'report [ir | dvd] -via -o <output_file>' to generate this report. The default output files are:

- Static: *adsRpt/Static/< >.via.ir.worst*
- Dynamic: *adsRpt/Dynamic/< >.via.ir.worst*

If set to 2, the report file **.via.ir.worst* is sorted by the voltage drop value across the via. *DMP compatible. Optional. Default: 0 (no report).*

Syntax:

```
VIA_IR_REPORT [ 0 | 1 ]
```

Custom Cell Modeling Keywords

CMM_CELLS

Specifies the available Custom Macro Model (CMM) cells to be used in the design, to speed up handling of hierarchical memory blocks and other macros used multiple times in a design. The 'optimize' option reduces the RLC network within CMM instances, resulting in fewer nodes and resistors, and thus reducing simulation memory use and runtime. If desired, you can use the full model by specifying the 'original' option. The 'raw' model is not the original input design data, such as a Spice netlist and GDS; it contains the processed LEF and DEF electrical and physical data, including a GSR and TCL template, which allows you to regenerate a CMM. The raw model can solve issues with different release versions, and allow regenerating a CMM with a different technology file to resolve a CMM version incompatibility, or a technology layer stacking consistency limitation. If desired, you can use the full model by specifying the 'original' option. The present values can be displayed using 'gsr get CMM_CELLS'. *DMP compatible. Optional. Default: "optimize" for cell_view and "original" for mmx_view models.*

Syntax:

```
CMM_CELLS {
    <cell_name> <model_cell_path> [original | optimize | raw ]
    ...
}
```

CMM_EXCLUDE_FILES

By default, all available files in CMM models are imported at top level when importing the CMM. To exclude some of these files by CMM model and file type, this keyword can be used. *Optional. Default: none.*

Syntax:

```
CMM_EXCLUDE_FILES {
    <modelname1> <file_type1> <file_type2> ...
    <modelname2> <file_type1> <file_type2> ...
    ...
}
```

}

where

<modelname1>: CMM model name
 <file_type> : type of file to be excluded

Example:

```
CMM_EXCLUDE_FILES {
    M1 ipf vcd spf sta ...
    M2 bpfs gsc
}
```

So in the example, file types IPF, VCD, SPF and STA should not be imported for CMM model M1, and BPFS and GSC information should not be imported for model M2. Note that BLOCK_POWER_ASSIGNMENT (BPA) *cannot be excluded* from the CMM.

Extra caution is needed when importing any instance power file (IPF), since any instance not covered by IPF is assigned a power of 0, which may not be desirable. So if any IPF file is imported, a warning is displayed that an IPF was imported for some CMM models, but not for top level or other CMM.

CMM_EXPAND_PINS_AT_TOP

Turning on this keyword expands DEF pins that are smaller than the shapes of wire geometries that they overlap, to eliminate CMM model connectivity issues at the top level of the design. This improves the accuracy of modeling of the connectivity between IP and the top level. Note that pins on all metal layers are considered for expansion, not just pins on the top layer. *Optional. Default: 0.*

Syntax:

```
CMM_EXPAND_PINS_AT_TOP [ 0 | 1 ]
```

CMM_INSTANCES

This keyword can be used to select either the 'optimize' or 'original' RLC network model for a specific CMM instance, rather than specifying the model at the CMM cell level. You can specify either 'optimize' or 'original' model for a CMM cell using the 'CMM_CELLS' keyword.

For CMM instances not listed in the CMM_INSTANCES keyword, or the CMM_INSTANCES keyword is absent, then the instance use the RLC view as defined at cell level in CMM_CELLS. Note the interaction between the CMM_CELLS and CMM_INSTANCES keywords:

- a. If optimization is needed for a CMM cell based on the CMM_CELLS keyword value, then the optimized model is generated, regardless of its actual use by any CMM Instance.
- b. If a CMM instance is specified to use the optimized model, but no optimized model is generated for the CMM cell (due to cell level options), then the CMM instance uses the original model.

You can verify which model was used for an instance in the output *adsRpt/apache.CMM.rpt* file. *Optional. Default: none.*

Syntax:

```
CMM_INSTANCES {
    <cmm_instance_path> {original | optimize}
    ...
}
```

}

CMM_LAYER_MAP_FILES

When specified, provides layer mapping when the technology and LEF layers are named differently in the CMM and in the top level design tech files. This feature is used in the top level run for a design containing CMMs. Layer maps are CMM cell specific; their layer names are mapped to layer names in top level design using the specified mapping files, which should specify the full name translation from layer names in CMM to top level layer names. The layer stacking order and matching layers' properties must be the same. The format of a CMM layer map file is:

```
<CMM_layer_name> <top_layer_name>
...
...
```

If you have a layer used in CMM but not in the top level, you should specify the relevant top tech layer as “UNUSED”, such as in the following file example:

```
pdiff UNUSED
co UNUSED
ndiff UNUSED
po UNUSED
METAL0 M0
VIA0 V0
Metall M1
...
...
```

The AUTO option creates a layer mapping file automatically by layer order for the CMM cell, assuming cells have the same layer order and position in both CMM and top level design. So for example, if technology file 1 has a fourth layer with name “m1”, and technology file 2 has a fourth layer named “metal1”, then if the ‘AUTO’ option is used, layer m1 is mapped to metal1, since both are fourth layers in their own tech files. *DMP compatible. Optional. Default: none.*

Note: If neither a layer map file nor AUTO is specified for a block, its layers are merged automatically based on layer height.

Syntax:

```
CMM_LAYER_MAP_FILES {
    <CMM_cellname> [<filename> | AUTO | UNUSED]
    ...
}
```

Example 1 - How the file is created :

Assume that your CMM tech file has 12 layers, C1 to C12, and your top tech file has 16 layers, T1 to T16. In the layer map file, you would specify layers C1 to C4 as “UNUSED”, then layers C5 to C12 map from T1 to T8 (these must be in stack order in one-to-one mapping, so C5 <-> T1, C6<->T2, ..., C12<->T8). The combined tech file for the top then is C1, C2, C3, C4, T1, T2, ..., T16. The layer properties for C1 to C4 layers in top tech are copied from the CMM tech file, the C5 to C12 layers in the CMM tech are dropped, and T1 and T8 layer properties from top tech are used for the layers in CMM. RedHawk does not check the property of layers C5 to C12 to see if they match T1 to T8; RedHawk only checks that the temperatures used for extraction for layer C5 to C12 matches the temperatures of layers T1 to T8.

Example 2:

For three CMM blocks in a top level run, CMM_LAYER_MAP_FILES could describe the layers as follows:

```
CMM_LAYER_MAP_FILES {
    dpL1DDATARAM ./myLayerMapFile
```

```
dpIDATARAM AUTO
# dpLSReqQueCam
}
```

So for CMM block ‘dpL1DDATARAM’, layers are merged based on the layer map file “myLayerMapFile”. For ‘dpIDATARAM’, first a layer map file is created automatically by layer order and then the tech files are merged. For ‘dpLSReqQueCam’, neither a layer map file nor AUTO are specified, so its layers are merged automatically based on layer height.

CMM_POWER_OVERRIDE_IPF

When set to 1, honors CMM Raw model BPA/BPFS instead of instance power file (IPF). *Default: 0*.

Syntax:

```
CMM_POWER_OVERRIDE_IPF [ 0 | 1 ]
```

Power calculation keywords

APL_INTERPOLATION_METHOD

When set to 4, enables APLDI to DD interpolation algorithm for long nets, high strength drivers, to enable accurate DvD result. *DMP compatible*. Default 0.

Syntax:

```
APL_INTERPOLATION_METHOD [ 0 | 4 ]
```

APL_MODE

In default (Accurate) mode, some grouping of APL samples is performed, which reduces the number of samples and hence run time and memory demand. In Extended mode, all unique samples are used in both power calculation and simulation. *Optional. Default: Accurate*.

Syntax:

```
APL_MODE [ Accurate | Extended ]
```

BIASPIN_CONFIG_FILE

When set, supports multiple-bias voltages by specifying a configuration file listing the bias pins for each instance and their corresponding bias voltages. Instance specification supports wildcards, and bias voltage values should be consistent with those characterized in the APL. Based on this file specification, PowerStream is able to select the right .lib/APL to be used for each instance in the design. *DMP compatible. Default: None*.

Syntax:

```
BIASPIN_CONFIG_FILE <filename>
```

The configuration file format is as follows:

```
<instance_name><biasPin1>=<biasV_1> [... <biasPinN>=<biasV_N>]
...
```

BLOCK_INSTANCE_POWER_FILE

To provide cell type (master cell for block) wise power number so that tool can assign the same power for all its instantiations. *DMP compatible*.

Syntax:

```
BLOCK_INSTANCE_POWER_FILE {
```

```

    FULLCHIP <block_name> <file_name>
    CELLCYPE <block_cell_name> <file_name>
}

```

where

<file_name> gives the detailed per instance power under the block, unspecified instance will be assigned zero power.

<block_cell_name> is the master block name which is instantiated multiple times in the design.

Please note:

- Blocks covered under FULLCHIP IPF file, will be filtered out to assign the user given power specified by CELLCYPE.
- If both INSTANCE_POWER_FILE and BLOCK_INSTANCE_POWER_FILE are set for an instance, priority will be given to INSTANCE_POWER_FILE power number.

BLOCK_POWER_FOR_SCALING

Specifies user-known block and instance power consumption values per block or instance, and for multiple Vdd/Vss designs, per pin, which RedHawk uses to scale individual component toggle rates to achieve more accurate analysis for instances and individual blocks, to match the total power consumption for the chip. The top-level block is specified for a flat design. The full hierarchical path must be supplied for all blocks and instances.

For multiple Vdd/Vss designs, power consumption is assigned and scaled by pin. If Vdd pin names are *not* specified, all VDD pin power is proportionally scaled with a common toggle rate. Note that this is supported only for leaf level cells for MVdd; hierarchical blocks are currently not supported. Also note that for MVDD specifications by pin, the Vss pin *current* is specified, not the power.

You can also control power assignment on a “celltype” basis, using the ‘CELLS’ option. Warning messages are displayed when BPFS settings create unrealistic toggle rates (greater than 1 for non-clock instances and greater than 2 for clock instances. The existing parameter values can be changed using ‘gsr set BLOCK_POWER_FOR_SCALING’ command prior to setup design. *DMP compatible. Optional. Default: none.*

Syntax:

```

BLOCK_POWER_FOR_SCALING {
    CELLCYPE <cell_name> [<pwr_W> ?<Vdd_pin>? | <cell_current_A> ?<Vss_pin> ? ]
    ...
    [FULLCHIP | BLOCK | <top_block_name>] <full_leaf_inst_path>
        <pwr_W>
    [FULLCHIP | BLOCK | <top_block_name>] <full_block_path>
        <pwr_W> ? <domain> ? ? <pin> ?
    ...
    [FULLCHIP | BLOCK | <top_block_name>]
        <full_mvdd_inst_path>
            [<pwr_W> ?<VDD_pin>? | <current_A> ?<Vss_pin>?]
    ...
    CELLS [ comb | ff_latch | mem | clockinst | io ] <pwr_W>
    ...
    STDCELL <cell_name> <power_W> <pin_name>
    ...
}

```

}

Example (GSR):

```
BLOCK_POWER_FOR_SCALING
{
    CELLCODE cellA1 0.01
    CELLCODE cell3 0.001
    FULLCHIP designABC 1.5
    BLOCK block1 0.5
    FULLCHIP block1/instance2 0.00322
    BLOCK block1/sub_blockA 0.1
    FULLCHIP block2/AN210J 0.001 VDD
    FULLCHIP ia32/nand342 0.005 VDD5
    CELLS comb 0.046
    CELLS ff_latch 0.0073
    CELLS mem 0.011
    CELLS clockinst 0.004
}
```

where

CELLTYPE <cell_name> : specifies individual cell name

<pwr_W>: power consumption for associated block, instance, or pin, in Watts

FULLCHIP | BLOCK | <top_block_name> : specifies power for the chip or block, or per domain or pin in the block. The sub-keyword or <top block name> must be defined in DEF after running RedHawk through power calculation.

<full_block_path> : specifies full hierarchical path for the block

<full_leaf_inst_path> : specifies full hierarchical path for the leaf instance

<full_mVdd_inst_path> <power_Watts>: full hierarchical path to multiple Vdd/Vss design instance and pin-based power consumption listing.

<VDD_pin> : optional - VDD pin name associated with specified power. If '< VDD pin name>' is *not* specified, all VDD pin power is proportionally scaled with a common toggle rate.

<VSS_pin> : optional - VSS pin name associated with specified current. If '< VSS pin name>' is *not* specified, all VSS pin current is proportionally scaled with a common toggle rate.

CELLS [comb | ff_latch | mem | clockinst | io]: assigns power to all cells of a particular celltype

STDCELL : assign scale power to the pins of standard cells. Cells recognized as standard cells are reported in the file *adsRpt/apache.stdCells*.

Example (in a Block_Power_For_Scaling file):

```
CELLTYPE cellA1 0.01
CELLTYPE cell3 0.001
FULLCHIP designABC 1.5
BLOCK block1 0.5
FULLCHIP block1/instance2 0.00322
BLOCK block1/sub_blockA 0.1
FULLCHIP block2/AN210J 0.001 VDD
FULLCHIP block2/AN210J 0.003 VDDC
FULLCHIP ia32/nand231 0.001 VDDQ
FULLCHIP ia32/nand342 0.005 VDD5
```

where

“designABC” is the name of the top level block. Since RedHawk flattens the design, the designABC is the full-chip.

“cellA1 0.01” and “cell3 0.001”, power of .01W is assigned any instance that uses cellA1as master cell and .001W is assigned to any instance that uses cell3.

The total power defined by “FULLCHIP block1 0.5”, includes power defined for <full_block_instance_path>/block1.

“block1/instance2 0.00322”, power is assigned to instance2, which is an instance in block1. For “block1/sub_blockA 0.1”, power is assigned to sub_blockA, which can be a clustered block inside block1.

The last four items are pin-based power specifications for mVdd.

BLOCK_POWER_FOR_SCALING_FILES

Specifies absolute or relative path from RedHawk run directory that contains the power specification, as in BLOCK_POWER_FOR_SCALING keyword. DMP compatible. Optional; default: None

Syntax:

```
BLOCK_POWER_FOR_SCALING_FILES
{
    <Block_Power_FilePathName>
}
```

Example:

```
BLOCK_POWER_FOR_SCALING_FILES
{
    designABC/block1-a_scaling
}
```

BLOCK_TOGGLE_FILES

Note: Using keywords VCD_FILE and BLOCK_VCD_FILES instead of this keyword is recommended.

Specifies the block toggle file for each block. The specified toggle file defines the average toggle rate for each net that is used to calculate power data.

The <toggle_file_name> is generated by running the *vcdtrans* utility on the respective VCD (Vector Change Dump) files and can be specified for each block. For more details on using *vcdtrans*, please refer to [section "vcdtrans", page E-870](#). If VCD is available, use VCD_FILE (defined below) for power calculation. DMP compatible. Optional. Default: None.

Syntax:

```
BLOCK_TOGGLE_FILES
{
    <block_name_N> <toggle_file_name_N>
    ...
}
```

where

<block_name_N> : name of block

<toggle_file_name_N> : specifies toggle filename

Example:

```
BLOCK_TOGGLE_FILES
{
```

```
    top_block top_block.toggle
}
```

BLOCK_TOGGLE_RATE

Defines the default toggle rate of the nets in a user-specified block or instance that are not otherwise specified and optionally also any associated clock nets. The blocks or instances should be defined in the DEF files. When BLOCK_TOGGLE_RATE is defined for a specific block, it supersedes the TOGGLE_RATE defined for the whole chip, but not a specified INSTANCE_TOGGLE_RATE for the leaf block only. The <clock_network_TR> entry for a block applies to clock pins and clock buffer outputs. For an instance, whether a clock network instance or not, the first TR entry applies to the network toggle rate, and the second, if present, applies to the clock buffer and clock pin toggle rate. The <block_mastercell/instance_name> specifications can take a wildcard ‘*’. For example, ABC* matches all block instance and instance names starting with ABC. *DMP compatible. Optional. Default: clock rate: 2.0.*

Syntax:

```
BLOCK_TOGGLE_RATE
{
    <block_inst_name/inst_name> <output_TR> ?<clock_network_TR>?
    <non-clock_inst_name> <output_TR> ?<clock_network_TR>?
    <clock_network_inst_name> <output_TR> ?<clock_network_TR>?
    ...
}
```

Example:

```
BLOCK_TOGGLE_RATE
{
    block1 0.5 1.0
    block2 0.4
    block3* 0.2
}
```

BLOCK_TOGGLE_RATE_FILES

Specifies the file containing toggle rates for blocks/instances defined in the .def file. The toggle rate settings in this file supersede the TOGGLE_RATE defined for the whole chip, but not a specified INSTANCE_TOGGLE_RATE for a block. The format of the file is as described for the BLOCK_TOGGLE_RATE keyword syntax. *DMP compatible. Optional. Default: None.*

Syntax:

```
BLOCK_TOGGLE_RATE_FILES <toggle_rate_filename>
```

Example:

```
BLOCK_TOGGLE_RATE_FILES ABCDE.block_TR
```

BLOCK_SAIF_FILES

This keyword supports use of hierarchical block SAIF activity files, and allows you to specify for hierarchical blocks the SAIF file path, the root instance path, and a substitute instance path. *DMP compatible. DMP compatible. Optional. Default: none.*

Syntax:

```
BLOCK_SAIF_FILES {
```

```

        SAIF_FILE {
            <block_name> <SAIF_file_path>
            ROOT_INSTANCE <root_instance_path>
            SUBSTITUTE_PATH <substitute_instance_path>
        }
        ...
    }

```

Example:

```

BLOCK_SAIF_FILES {
    SAIF_FILE {
        green_core3 green.saif
        ROOT_INSTANCE green_tst/inst1/
        SUBSTITUTE_PATH green_core3/
    }
}

```

CELL_RC_FILES

Defines the SPEF/DSPF interconnect parasitics file for each cell in a flat or hierarchical design. When the EXTRACT_RC option is set to 1 (default), RedHawk builds a c1-r-c2 equivalent circuit for each instance output RC network specified in the SPEF file. Only the instance output driving a signal net (P/G nets are ignored) gets an equivalent C1/R/C2 when set. When set to 0, RedHawk uses the total output C for each instance output instead. For static IR-drop analysis, set EXTRACT_RC to 0. The CONDITION keyword allows selection of one of the capacitance value types from a three-value SPEF file. The syntax allows either listing the cellnames and paths, or listing SPEF files and then including the file the same as LEF/DEF/LIB files. The '<>.spfs' filename specifies the file containing the list of SPEF/DSPF files. This file must be the last item in the list and its name must end with *.spfs*, to indicate that the alternate format is being used. The contents of the *<>.spfs* file simply lists the cells and pathnames.

Note that you should specify at least one of the following keywords:
 CELL_RC_FILES, INTERCONNECT_GATE_CAP_RATIO, or
 STEINER_TREE_CAP. If all three keywords are specified, then the
 CELL_RC_FILES value is used for the nets in the cells/blocks that have the SPF
 file or DSPF file defined. For the rest of the blocks, if defined, the value of
 STEINER_TREE_CAP is used. If no STEINER_TREE_CAP or CELL_RC_FILES is
 specified, then INTERCONNECT_GATE_CAP_RATIO is used. If none of these
 keywords are specified, then the default value of 1 for the
 INTERCONNECT_GATE_CAP_RATIO is used.

Note that if you want to run PJX clock tree analysis, the files specified must contain coupling capacitance data. *DMP compatible. Optional. Default: EXTRACT_RC: 1; CONDITION: typical.*

Syntax:

```

CELL_RC_FILE
{
    EXTRACT_RC [ 0 | 1 ];
    CONDITION [best | typical | worst ]
    [ <cell_name> <path_to_dspf-spef_file>
        ...
    | <user_data>/abc.spfs ]
}

```

Example:

```
CELL_RC_FILE
{
    CONDITION worst
    cell1 SPEF_files/cell1.spef
    cell2 SPEF_files/cell2.spef
}
```

CELL_TOGGLE_RATE

This keyword allows specifying individual toggle rates by leaf cell name. *DMP compatible. Optional. Default: None.*

Syntax:

```
CELL_TOGGLE_RATE {
    <leaf_cell_name> <toggle_rate> ?<clock_toggle_rate>?
    ...
}
```

CELL_TOGGLE_RATE_FILE

Specifies a file or files containing specific toggle rates for cells defined in the .def file. The toggle rate settings in this file supersede the TOGGLE_RATE defined for the whole chip. *DMP compatible. Required. Default: none.*

Syntax:

CELL_TOGGLE_RATE_FILE <toggle_rate_file>
or

```
CELL_TOGGLE_RATE_FILE {
    <toggle_rate_file1>
    <toggle_rate_file2>
    ...
}
```

CLOCK_DOMAIN_TOGGLE_RATE

When set, allows you to set a default toggle rate for instances in certain clock domains, and optionally the clock toggle rate. *Default: none.*

Syntax:

```
CLOCK_DOMAIN_TOGGLE_RATE <clock_domain_name>
    <toggle_rate> ? <clock_toggle_rate> ?
```

CLOCK_DOMAIN_TOGGLE_RATE_FILE

When set, specifies a file that describes default toggle rates for instances in certain clock domains, and optionally the clock toggle rate. *Default: none.*

Syntax:

```
CLOCK_DOMAIN_TOGGLE_RATE_FILE <filename>
```

The format of the file is shown below.

```
<clock_domain_name1> <toggle_rate1> ?<clock_toggle_rate>?
...

```

CYCLE_SELECTION_GRID_SIZE

In cycle selection flow, this enables user to specify the grid size (the width by which window has to slide). When this keyword is not set, tool sets this value automatically as the inverse of highest frequency found in design. *Default: none.*

Syntax:

```
CYCLE_SELECTION_GRID_SIZE <time_in_seconds>
```

GDS_BPFS_OVERRIDE_IPF

When set to 1, honors target power values in BLOCK_POWER_FOR_SCALING for GDS cells over power assignments in instance power file. *Default: 0.*

Syntax:

```
GDS_BPFS_OVERRIDE_IPF [ 0 | 1 ]
```

GSC_OVERRIDE_IPF

When set to 1, RedHawk ignores all IPF power entries for instances with any specified GSC states. Also, when set to 1 RedHawk overrides the settings of INSTANCE_TOGGLE_RATE, BLOCK_TOGGLE_RATE, TOGGLE_RATE, and the power entries specified in INSTANCE_POWER_FILES. However, power values in BPFS and the toggle rates set in the GSC file are honored. *DMP compatible.*

Optional. Default: 0.

Syntax:

```
GSC_OVERRIDE_IPF [ 0 | 1 ]
```

Example:

```
GSC_OVERRIDE_IPF 1
```

INSTANCE_POWER_FILES

Defines the absolute or relative path from the RedHawk run directory to the power file, which contains a list of instances and their power consumption, as well as hierarchical prefixes to add the instance names in the IPF files to match them to the design instances. For instances not specified in the power file, the power is assumed to be zero. Therefore, all significant power consumers should be specified, or none. Note that filler and tap cells are ignored when checking IPF coverage. For mixed mode analysis, you can use VCD for block power and INSTANCE_POWER_FILES to assign power to the top level instances (excluding the block). For non-mixed mode flows, INSTANCE_POWER_FILES still takes the highest priority and overrides all other keywords defining power/activity for instances. *DMP compatible. Optional; default: None.*

Syntax:

```
INSTANCE_POWER_FILES [ <inst_power_file_path> |
{ <inst1_power_file_path1>
<inst2_power_file_path2>
<inst3_power_file_path3> <inst3_hierpath>
<inst4_power_file_path4> <inst4_hierpath>
...
} ]
```

where

<instN_power_file_pathN> : specifies the paths to the IPF files

<instN_hierpath> : specifies the hierarchical prefix to add the instance names in the IPF files to match them to the design instances. If the hierarchical path

prefix is not specified, the <instN_power_file_pathN> should specify the instance name with complete hierarchy. If the hierarchical path prefix is specified, each instance name in <instN_power_file_pathN> is prefixed with the <instN_hier_pathN> to match them to the design instances in hierarchy.

Example:

```
INSTANCE_POWER_FILES
  ( designABC/abc/instPowerFileABC HIER_CD
  )
```

The format for specifying pin power/current information in the Instance Power File for multiple Vdd/Vss designs, one pin per line, is:

```
<instance_name> <pin_power_W> <Vdd_pin_name>
<instance_name> <pin_current_A> <Vss_pin_name>
...
```

To specify pin power for single Vdd/Vss designs, the format is:

```
<instance_name> <total power_W>
...
```

INSTANCE_TOGGLE_RATE

Specifies average toggle rates for instances in the design. If there are a lot of instances in the chip, using this keyword is recommended, rather than using BLOCK_TOGGLE_RATE or BLOCK_TOGGLE_RATE_FILE keywords. Wildcard characters are supported. Whether a clock network instance or not, the first TR entry applies to the network toggle rate, and the second, if present, applies to the clock buffer or clock pin toggle rate. If only one TR value is specified, it is used for the output/signal toggle rates associated with the instance. *DMP compatible*.

Optional. Default: clock: 2.0.

Syntax:

```
INSTANCE_TOGGLE_RATE {
  <non-clock_inst_name> <output_TR> ?<clock_network_TR>?
  <clock_net_inst_name> <output_TR> ?<clock_network_TR>?
  ...
}
```

Example:

```
INSTANCE_TOGGLE_RATE
{
  instance1 0.5 1.5
  instance2 0.3
}
```

INSTANCE_TOGGLE_RATE_FILES

Specifies the instance toggle rate file, which provides toggle rates for instances on the chip. The format of the contents of the file is:

```
<non-clock_inst_name> <output_TR> ?<clock_network_TR>?
```

A message is displayed and recorded in the log file showing the number of instances listed in the INSTANCE_TOGGLE_RATE_FILES successfully read in and also the number of listed instances not found in the design or floating. Wildcard characters are supported. *DMP compatible. Optional. Default: None.*

Syntax:

```
INSTANCE_TOGGLE_RATE_FILES <filename>
```

Example:

```
INSTANCE_TOGGLE_RATE_FILES ABCDE_inst_TR_file
```

INTERCONNECT_GATE_CAP_RATIO

Defines the ratio of the total interconnect capacitance of the net relative to the total gate capacitance of the input pin fanouts. If none of the following keywords, INTERCONNECT_GATE_CAP_RATIO, CELL_RC_FILE, or STEINER_TREE_CAP have specified values, power calculation uses the default value of INTERCONNECT_GATE_CAP_RATIO. *DMP compatible. Optional.*
Default: 1.

Syntax:

```
INTERCONNECT_GATE_CAP_RATIO <value>
```

Example:

```
INTERCONNECT_GATE_CAP_RATIO 1.5
```

IPF_ERROR_THRESHOLD

Setting IPF_ERROR_THRESHOLD overrides the default value, so that if at least the specified percentage of instances is not defined in IPF, Power Calculation errors out. *DMP compatible. DMP compatible. Optional. Default: 50%.*

Syntax:

```
IPF_ERROR_THRESHOLD <percent_instances>
```

ITR_OVERRIDE_BPFS

When set, instances that have an INSTANCE_TOGGLE_RATE setting are skipped when performing BLOCK_POWER_FOR_SCALING.

BPFS scaling is implemented in the last step of power calculation. To meet the power target, the toggle rate is adjusted. The keyword GSC_OVERRIDE_IPF applies to the instances having GSC settings. ITR_OVERRIDE_BPFS is for the instances that have INSTANCE_TOGGLE_RATE settings in the GSR; these instances honor the instance toggle rate and are **not** scaled by BPFS.

So if power P1 is specified in BPFS for the whole design, then after retaining all individual instance toggle rate settings the final power is P1, unless all instances have toggle rates specified. *DMP compatible. Optional. Default: 0.*

Syntax

```
ITR_OVERRIDE_BPFS [0 | 1]
```

LIB2AVM_MSTATE

When set, memories with multi-state power values are intelligently handled by lib2avm, so that only the states present in the VCD file are considered for power calculation. The power values in the .lib corresponding to the Boolean state of the VCD determine the total power of the instance. For the RTL VCD flow and the 'LIB2AVM_MSTATE 1' GSR keyword set, load-independent memory custom states can be generated with either 'FILE_TYPE VCD' or 'FILE_TYPE RTL_VCD' option settings under the BLOCK_VCD_FILES GSR keyword. When set to 2, load dependent AVM config file is generated. *DMP compatible. Optional. Default: 0.*

Syntax

```
LIB2AVM_MSTATE [0 | 1 | 2]
```

NET_LOAD_FILE

Specifies a file that describes the load capacitance for a list of nets, in the format:

```
<net_name1> <load_cap1>
<net_name2> <load_cap2>
...
```

Also, the specified file can enable a scaling factor function to adjust the RedHawk-calculated Instance Cload (for Dynamic-IR simulation) and Instance Power (for Static-IR simulation). This function permits bounding the IR-drop calculations by adding a percentage error (derived from Spice simulations) as a scaling factor. To enable this enhancement, in the Net Load File set the keyword "OUTPUT_PIN_CAP_SCALE 1", and provide the desired scaling factors in the format :

```
<Inst_name>/<Pin_name> <scaling_factor>
...
```

Example Net Load File scaling:

```
$OUTPUT_PIN_CAP_SCALE 1
ucpu0/clk_drvr_13/Y 1.2
ucpu1/clk_drvr_13/Y 1.2
```

DMP compatible. Optional. Default: None.

Syntax

```
NET_LOAD_FILE <filename>
```

NET_TOGGLE_RATE

Specifies toggle rate and optionally the duty cycle (fraction of time between 0 and 1.0 that the net is high) for particular nets in the design. If only one value is specified, it is considered to be the toggle rate only. The toggle rate for nets not specified by this keyword is defined by the TOGGLE_RATE GSR keyword value. These settings also override toggle rates derived from GATED_ON_PERCENTAGE keyword. Wild cards for <net_name> are allowed. *DMP compatible. Optional. Default: TR: none; duty cycle: 0.5.*

Syntax:

```
NET_TOGGLE_RATE {
    <net_name1> <toggle_rate1> ?<duty_cycle1>?
    <net_name2> <toggle_rate2> ?<duty_cycle2>?
    ...
}
```

Example:

```
NET_TOGGLE_RATE {
    net1 0.5 0.1
    net2 0.3
}
```

NET_TOGGLE_RATE_FILE

Specifies a file that defines toggle rates and optionally duty cycles for particular nets in the design. In the file specified, the nets for which toggle rates are to be changed can be specified using wildcard characters. The content of the file is a list of nets and their toggle rates and duty cycles, one net per line. RedHawk performs basic checks on entries in this file and verifies that the toggle rate and duty cycle given for each net are in allowable range. Error/Warning messages are displayed for values out of range. Duty cycle values that are out of range are reset to 1. Toggle rate values that are out of range are retained as they are. *DMP compatible. Optional. Default: None.*

Syntax

```
NET_TOGGLE_RATE_FILE <file_path>
```

POWER_ALLOW_MULTIPLE_STATE

In general, PowerStream and simulation only allow one output signal pin event in each cycle, and extra events are filtered out, including FF/Latches and memories without state definitions, as well as multiple-state cells. The settings are defined as follows:

- 0: filter all extra events
- 1: only allow extra events for custom state cells; for multiple-state cells all events are dumped into the scenario file, and no events are filtered out.
- 2: all cells are allowed to have multiple events.

DMP compatible. Optional. Default: 1.

Syntax:

```
POWER_ALLOW_MULTIPLE_STATE [ 0 | 1 | 2 ]
```

POWER_ALLOW_PER_PIN_IPF

By default when the IPF (Instance_Power_File) is used for multi-domain instances, RedHawk matches power values for each pin (each domain) through charge scaling, in which case the toggle rate remains unchanged. This keyword, when set to 0, provides an alternate way to handle instance power for multi-domain IPF instances. When set to 0, instance power is taken as the sum of the powers of all pins from IPF and the toggle rate is scaled to meet this power target. For example, for a multi-domain instance 'Top/mem1' with the following per pin dynamic power, and with the default toggle rate (no IPF):

```
Top/mem1 vdd1 1e-04  
Top/mem1 vdd2 2e-04
```

The total dynamic power in this case would be 3e-04. But if the IPF dynamic power given is:

```
Top/mem1 vdd1 3e-04  
Top/mem1 vdd2 3e-04
```

then the total dynamic power needed is 6e-04. In this approach, the ratio of total dynamic power is considered, rather than using per pin charge scaling, and the toggle rate is scaled accordingly. In this case, the toggle rate is scaled by two. *DMP compatible. Optional. Default: 1.*

Syntax:

```
POWER_ALLOW_PER_PIN_IPF [ 1 | 0 ]
```

POWER_ANALYSIS_MODE

VCD event-based state-dependent power calculation mode can be invoked using this keyword set to ‘averaged’. Only events defined in a gate VCD/FSDB file are used as a basis for power calculation in ‘averaged’ mode, although multiple-block VCDs are supported. Note also that in “averaged” mode both rise and fall events are used when calculating switching power. The default ‘averaged_fast’ mode uses the state with the highest power for power calculation. *DMP compatible. Optional. Default: averaged_fast.*

Syntax:

```
POWER_ANALYSIS_MODE [ averaged | averaged_fast ]
```

POWER_CYCLE_SELECT_BLACK_BOX

When specified, tile-based cycle selection is enabled if VCD_FILE options WORST_TILE_POWER_CYCLE or WORST_TILE_DPDT_CYCLE are specified. Only the tiles *outside* of the bounding box are included in cycle selection. *DMP compatible. Optional. Default: none.*

Syntax:

```
POWER_CYCLE_SELECT_BLACK_BOX {
    <x1> <y1> <x2> <y2>
    <x3> <y3> <x4> <y4>
    ...
}
```

POWER_CYCLE_SELECT_MODE

Specifies desired characteristics of fast power calculation to trade off runtime, memory demand, and accuracy in performing cycle selection. There are accuracy differences between power calculated during cycle selection and the final power computed for the worst cycle, which can be controlled using this keyword setting. *DMP compatible. Optional. Default: 0.*

Syntax:

```
POWER_CYCLE_SELECT_MODE [ 0 | 1 ]
```

where

0: runtime : low, memory use : low, accuracy: low (default)

1: runtime : high, memory use : high, accuracy: high (default when GSR “POWER_ANALYSIS_MODE averaged” is set)

POWER_CYCLE_SELECT_POWER_NETS

When set, specifies a subset of power nets on which to perform cycle selection. *DMP compatible. Optional. Default: none.*

Syntax:

```
POWER_CYCLE_SELECT_POWER_NETS {
    <net1>
    <net2>
    ...
}
```

POWER_CYCLE_SELECT_REPORT_VDD

When set, specifies the Vdd nets (only) for which PowerStream domain cycle power is reported in the file *adsRpt/cycle.domain.pwr* in the cycle selection flow. *DMP compatible. Optional. Default: none.*

Syntax:

```
POWER_CYCLE_SELECT_REPORT_VDD {
    <Vdd_a>
    <Vdd_s>
    ...
}
```

POWER_CYCLE_SELECT_WHITE_BOX

When specified, tile-based cycle selection is enabled if VCD_FILE options WORST_TILE_POWER_CYCLE or WORST_TILE_DPDT_CYCLE are specified. Only the tiles *within* the bounding box are included in cycle selection. *Default:* none.

Syntax:

```
POWER_CYCLE_SELECT_WHITE_BOX {  
    <x1> <y1> <x2> <y2>  
    <x3> <y3> <x4> <y4>  
    ...  
}
```

POWER_DISABLE_SWITCH

When set, the toggle rate for all switch cells is set to zero. *DMP compatible.*
Optional. *Default:* 0.

Syntax:

```
POWER_DISABLE_SWITCH [ 0 | 1 ]
```

POWER_DOMAIN_TOGGLE_RATE

When set, allows you to set a default toggle rate for instances in certain power domains, and optionally the clock toggle rate. *Optional.* *Default:* none.

Syntax:

```
POWER_DOMAIN_TOGGLE_RATE <power_domain_name>  
<toggle_rate> ? <clock_toggle_rate> ?
```

POWER_DOMAIN_TOGGLE_RATE_FILE

Specifies a file that describes default toggle rates for instances in certain power domains, and optionally the clock toggle rate. *Default:* none.

Syntax:

```
POWER_DOMAIN_TOGGLE_RATE <filename>
```

The format of the file is shown below.

```
<power_domain_name1> <toggle_rate1> ?<clock_toggle_rate1>?  
...  
...
```

POWER_DRIVER_TOGGLE_RATE

Propagates the instance toggle rate to driver nets to insure consistency between the instance toggle rate and the driver net toggle rate. *Optional.* *Default:* 1 (On).

Syntax:

```
POWER_DRIVER_TOGGLE_RATE [ 0 | 1 ]
```

POWER_HIER_REPORT_LEVEL

When set, generates a Hierarchical Power Report. The <level> argument specifies the depth of hierarchy required in the report . For level = 0, the hierarchical report is disabled (default); for level = -1, power at all physical hierarchical levels is reported. The report file is *adsRpt/design_name.power.hiers*.

You also can generate the report on the fly using “gsr set” in the GUI, but ‘perform pwrcalc’ must be redone after resetting the keyword on the fly. *DMP compatible.*
Optional. *Default:* Off.

Syntax:

```
POWER_HIER_REPORT_LEVEL <level>
```

Example:

```
POWER_HIER_REPORT_LEVEL 2
```

In this case, power for all the blocks to the 2nd hierarchy level are reported, including the top level.

POWER_HONOR_LIB_K_FACTOR

When set, controls the behavior of power calculation/analysis based on the specified voltage-related lib k-factors. Instance leakage /internal power values are scaled according to the following formulas:

$$\text{Scaled leakage power (volt)} = \langle \text{leakage_power} \rangle * (1 + (\text{Vp} - \text{Nom_volt}) * \langle \text{k_volt_cell_leakage_power} \rangle)$$

$$\text{Scaled internal power (volt)} = \langle \text{internal_power} \rangle * (1 + (\text{Vp} - \text{Nom_volt}) * \langle \text{k_volt_internal_power} \rangle)$$

where Vp is the instance power pin value, Nom_volt is the lib nominal voltage, and k_volt_cell_leakage_power and k_volt_internal_power are voltage-related lib k-factor multipliers. *Optional. Default: 0 (off, no scaling).*

Syntax:

```
POWER_HONOR_LIB_K_FACTOR [ 0 | 1 ]
```

POWER_IGNORE_ASYNC_PIN

When set, power calculation automatically identifies asynchronous input power pins of memory cells and ignores them when calculating power. *Optional. Default: 0 (off)*

Syntax:

```
POWER_IGNORE_ASYNC_PIN [ 0 | 1 ]
```

POWER_INTERNAL_SCALING_FILE

To scale the internal power in .lib using the per cell internal power scaling values in the scaling factor file specified. All internal power values in Lib are multiplied by the specified factor for the cell. Scaling can be done on a per pin basis as well. Wildcards are allowed for cell names in the file. *Default: off*

Syntax:

```
POWER_INTERNAL_SCALING_FILE <scaling_factor_filename>
```

Format of scaling factor file:

```
<cell_name1> <scaling_factor1> <power pin name>
<cell_name2> <scaling_factor2> <power pin name>
```

POWER_LEAKAGE_SCALING_FACTOR

When set, all leakage power values in Lib are multiplied by the specified factor. *DMP compatible. Optional. Default: 1.0.*

Syntax:

```
POWER_LEAKAGE_SCALING_FACTOR <factor>
```

POWER_LEAKAGE_SCALING_FILE

Specifies a file to define leakage power scaling, including per domain per cell. leakage power for different domains. Wildcards are also supported for cell names. For one cell, multiple pins can be defined in the file. All leakage power values in Lib

are multiplied by the specified factor for the cell. The format of the leakage scaling factor file is :

```
<cell_name1> <leakage_scaling_factor1> ?<domain_name>?
<cell_name2> <leakage_scaling_factor2> ?<domain_name>?
...
...
```

File example:

```
Cell_1a 0.2
Cell_2d 0.4 VDD
Cell_2d 0.8 VDBB
* 0.4 VDBB
* 0.6 VDD
```

DMP Compatible. Optional. Default: none.

Syntax:

```
POWER_LEAKAGE_SCALING_FILE <scaling_filename>
```

POWER_MCF_MULTI_CLOCK

When turned on, supports power calculation by automatically controlling the generation of vectors for multiple clock cells as defined in the MCF file. *Optional. Default: 0.*

Syntax:

```
POWER_MCF_MULTI_CLOCK [ 0 | 1 ]
```

POWER_MISSING_IPF_LEAK

For instances with missing INSTANCE_POWER_FILES values, this keyword by default preserves leakage power values. When set to 0 (for cells missing in the IPF -- no entry at all), zero power is assumed for these instances, including leakage power. *DMP compatible. Optional. Default: 1.*

Syntax:

```
POWER_MISSING_IPF_LEAK [ 0 | 1 ]
```

POWER_MISSING_IPF_POWER

For instances with missing INSTANCE_POWER_FILES values, this keyword by default (Off) sets missing power values to 0. If the keyword is set to 1, power calculation uses the global toggle rate to compute the power values for instances without IPF values. *DMP compatible. Optional. Default: 0.*

Syntax:

```
POWER_MISSING_IPF_POWER [ 0 | 1 ]
```

POWER_MODE

Specifies the primary data source for internal/switching power and leakage power calculation analysis. *DMP compatible. Optional. Default: Mixed.*

Syntax:

```
POWER_MODE [ APL | LIB | MIXED | APL_PEAK | APL_PEAK1 ]
```

where

APL : specifies primary use of APL power data for internal/switching (*cell.ifprof*) and leakage power (*<cell>.cdev*). Where APL data are not available, *.lib* data are used.

LIB : specifies use of `.lib` power consumption data; cells without power data in `.lib` do *not* have internal power consumption data, but have switching power information from RedHawk.

MIXED : specifies primary use of `.lib` power data; for cells without power components (internal power consumption or leakage power) in `.lib`, APL data are used.

APL_PEAK: uses the peak charge from APL in power calculation for every cell in the design, and the current is derived from the charge.

APL_PEAK1: uses the APL peak current values for every cell in the power calculation. Power for each instance is computed as $\text{Power} = (\text{peak current}) * (\text{supply voltage}) * (\text{toggle rate})$. Using peak current leads to a higher power value if a more conservative model is desired.

Example:

```
POWER_MODE APL_PEAK
```

POWER_REPORT_BIAS_PIN

When On (default), instance bias pin power is not included in the power summary report, but setting the keyword does not change the instance power file. *Optional.*

Default: 1.

Syntax:

```
POWER_REPORT_BIAS_PIN [ 0 | 1 ]
```

POWER_STATE_DEPENDENT_LEAKAGE

When turned On, calculates state-dependent leakage power in vectorless and VCD analyses. *DMP compatible.* *Optional.* *Default:* 0.

Syntax:

```
POWER_STATE_DEPENDENT_LEAKAGE [ 0 | 1 ]
```

POWER_TRANSIENT_ANALYSIS

Power Transient Analysis, also known as “Variable Power”, provides more flexibility to simulate frame-by-frame power/current transient behavior. The keyword **POWER_TRANSIENT_ANALYSIS** allows you to specify a configuration file for key frames that refer to one or more power control keywords for use in analysis of that frame. The following power control GSR keywords are supported by Power Transient Analysis:

GSC_FILES

INSTANCE_POWER_FILES

STATE_PROPAGATION

GATED_CONTROL_FILE (option under STATE_PROPAGATION)

GSC_OVERRIDE_IPF

BLOCK_POWER_FOR_SCALING

TOGGLE_RATE_RATIO_COMB_FF

INSTANCE_TOGGLE_RATE

BLOCK_TOGGLE_RATE TOGGLE_RATE

Note that Power Transient Analysis works only in vectorless mode. *DMP compatible.* *Optional.* *Default:* None.

Syntax:

```
POWER_TRANSIENT_ANALYSIS {
```

```
<duration_frame1_sec> <config_file1>
<duration_frame2_sec> <config_file2>
...
}
```

POWER_USE_CCS

When set, utilizes CCS current data captured in the Liberty file for power computation. The current waveform corresponding to load/slew is converted to internal energy values. *Optional. Default: 0.*

Syntax:

```
POWER_USE_CCS [ 0 | 1 ]
```

POWER_VCD_COVERED_THRESHOLD

Specifies the threshold for gate VCD coverage check. RedHawk terminates, if the VCD coverage is less than this threshold. *Default: 0.05*

Syntax:

```
POWER_VCD_COVERED_THRESHOLD <value>
```

POWER_VCD_EVENT_SEQUENCE

When set, RedHawk performs event sequence numbering. In cases where multiple pins are toggling at the same time, the order of these toggling is captured in FSDB, which then decides the state at which the corresponding instances are. Such occurrences are commonly found in memories/IPs.

Please note that the FSDB needs to be created after setting the environment variable NOVAS_FSDB_ENV_DUMP_SEQ_NUM for sequence numbering to be present.

Syntax:

```
POWER_VCD_EVENT_SEQUENCE [ 0 | 1 | 2 ]
```

where

0 : default value, don't consider any sequence number

1: consider symbols sequence number and clock pin trigger sequence number

2: only consider symbols sequence number, don't consider clock pin trigger sequence

POWER_VCD_HONOR_GLITCH_EVENT

When set to 1, RedHawk calculates separate glitch power component for VCD based power calculation. Two or more close transitions are considered a glitch bunch, if they are more close than the average slew (average of rise and fall slew) for the pin. The total dynamic power due to these toggles are scaled down by a ratio and accounted separately as glitch power. The scaling ratio is:

Ratio = (time difference between close toggles/average slew) ^ 2

To enable this ‘POWER_ANALYSIS_MODE averaged’ must be set in GSR. Events that are marked out as glitch will be excluded from simulation

A separate section `glitch_power` will also be seen next to switching/internal/leakage power in power summary reporting.

Power of different cell types in Watt:						
cell_type	total_pwr	leakage_pwr	internal_pwr	switching_pwr	glitch_pwr	%_total_pwr
combinational	7.0114e-05	5.9209e-05	7.5090e-10	7.7530e-06	3.1520e-06	8.0346e+01
latch_and_FF	1.7150e-05	1.7150e-05	0.0000e+00	0.0000e+00	0.0000e+00	1.9652e+01
memory	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
I/O	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
misc_seq	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
decap	1.5681e-09	1.5681e-09	0.0000e+00	0.0000e+00	0.0000e+00	1.7969e-03
Total	8.7266e-05	7.6360e-05	7.5090e-10	7.7530e-06	3.1520e-06	1.0000e+02
(clock_network power is part of this total power)						
clock_network_power 2.5720e-08 2.5720e-08 0.0000e+00 0.0000e+00 0.0000e+00 2.9473e-02						
where misc_seq are instances that can not be classified as latch_and_FF, memory, or I/O but have clock pin(s).						
Total chip power, 8.7266e-05 Watt including core power and other domain power.						
Total glitch power, 3.152e-06 Watt accounts for 3.612(%) percent of total power.						

Figure C-3 Glitch power reporting in power summary report

Also, individual glitch powers will be present in the file `adsRpt/ <design_name>.glitch.rpt`

Syntax:

```
POWER_VCD_HONOR_GLITCH_EVENT [ 0 | 1 ]
```

POWER_VCD_IGNORE_ERROR

When set to 1, if encountering an incorrect block name, RedHawk skips the VCD file and continues with power calculation, and also ignores invalid VCD symbols. By default RedHawk errors out when a block name is not specified correctly. *Optional. Default: 0.*

Syntax:

```
POWER_VCD_IGNORE_ERROR [ 0 | 1 ]
```

POWER_VCD_LIMIT_TR

When set, changes the toggle rate to 1 prior to VCD-based power calculation for any instance whose cycle time (1/frequency) is lower than the duration (END_TIME - START_TIME) in the VCD_FILE keyword. This avoids unrealistically large instance toggle rates (number of toggles / number of cycles) due to use of a number of cycles smaller than 1. *Optional. Default: 0.*

Syntax:

```
POWER_VCD_LIMIT_TR [ 0 | 1 ]
```

POWER_VCD_NUM_PROCESS

Specifies the number of processors to use for VCD analysis. The multi-threading function is On by default, so RedHawk automatically decides how many VCDs are read in parallel. *DMP compatible. Default: '-1' for flat and '1' for DMP flow.*

Syntax:

```
POWER_VCD_NUM_PROCESS <num_proc>
```

POWER_VCD_OVERRIDE_IPF

When set to 1, IPF data is ignored if the instance is covered by VCD information. The default value of 0 maintains behavior prior to version 12.1, in which, for Mixed-Mode analysis, when the design had VCD files as well as IPF for both the block and the TOP-level instances, IPF power was considered where there were differences in the switching scenario and power values. *DMP compatible. Optional. Default: 0.*

Syntax:

```
POWER_VCD_OVERRIDE_IPF [ 0 | 1 ]
```

POWER_VCD_REUSE_EVENT

For multiple VCD file flows, if there are multiple blocks using the same VCD/FSDB file with the same VCD_FILE options FRONT_PATH and START_TIME, this keyword when set to 1, applies first block events to other blocks using the same VCD file. That is, RedHawk reads the VCD/FSDB only once, checks if the global nets are reusable and reuses the VCD for other blocks, to speed up VCD analysis. This feature can be used in ‘averaged’ mode also. If this keyword is set to 2, RedHawk reuses VCD without performing check for global nets reusability. *DMP compatible. Optional. Default: 1 (On).*

Syntax:

```
POWER_VCD_REUSE_EVENT [ 0 | 1 | 2 ]
```

where

0: turn off

1: default value, will check if global nets are reusable and keeps accuracy

2: does not check global nets, if many VCD's have same front path and other options, then it will reuse and speed up the run.

POWER_VCD_TO_FSDB

Converts VCD to FSDB automatically and thereby speeds up PowerStream.
Default: 1 (on).

Syntax:

```
POWER_VCD_TO_FSDB [ 0 | 1 | 2 ]
```

Where:

0 : do not convert VCD to FSDB

1 : converts VCD to FSDB without keeping the sequential number

2 : converts VCD to FSDB with the sequential number

POWER_WORST_IO_PAD

When set, when simulation picks a current profile for an I/O instance (against slew/load combinations), the slew of only the active input corresponding to the simulated state ID is considered, making results more realistic. *Optional. Default: 0.*

Syntax:

```
POWER_WORST_IO_PAD [ 0 | 1 ]
```

POWER_WORST_LEAKAGE

When set, automatically picks the worst-case state-dependent leakage power value from all states, instead of just “cell_leakage_power” or average leakage power, from all the states in the library. *Optional. Default: 0.*

Syntax:

```
POWER_WORST_LEAKAGE [ 0 | 1 ]
```

POWER_WORST_MBFF

By default the Power transient flow handles APL current profiles for multiple-bit flip flops (MBFF) by scaling the current based on the number of bits switching in MBFFs. If set to 1, no current scaling is performed. *Optional. Default: 0.*

Syntax:

```
POWER_WORST_MBFF [ 0 | 1 ]
```

PRIMARY_OUTPUT_LOAD_CAPS

Specifies the capacitance of the output driver nets, usually with a fanout of 1. The option <output_driver_net_name> supports “*” wild card matching. If DEFAULT_LOAD is specified, the default Cload specified is applied to all of the primary I/O output nets, even if the connected standard cell is not defined as an I/O pad cell in the lib. *DMP compatible. Optional. Default: none.*

Syntax:

```
PRIMARY_OUTPUT_LOAD_CAPS
{
    IGNORE_FANOUT_CAP [ 1 | 0 ]<powerNet_name>
    DEFAULT_LOAD <output_cap_Farads> <Vdd_name>
    <output_driver_net_name> <output_cap_Farad> <Vdd_name>
    ...
    extra_load_cap [ 1 | 0 ]
}
```

where

IGNORE_FANOUT_CAP [0 | 1] <powerNet_name> : value 1 ignores fanout cap in the specified power net.

<Vdd name> : Vdd name that should be defined in the VDD_NETS keyword.

extra_load_cap : when set to 1 for a power EMIR run, the wire cap is also accounted for from the SPEF for the driver instances driving a PRIMARY_OUTPUT port. By default, this option is off and RedHawk only accounts for the load cap directly from the GSR that is provided under PRIMARY_OUTPUT_LOAD_CAPS for a power EMIR run

Example:

```
PRIMARY_OUTPUT_LOAD_CAPS
{
    IGNORE_FANOUT_CAP 1
    DEFAULT_LOAD 20p
    PAD_g* 50p
}
```

Note: You can use DEFAULT_LOAD only for assigning loads to I/O cell nets. For all other nets in the design, specify the net name using the following syntax, without the DEFAULT_LOAD option (wild cards allowed):

```
PRIMARY_OUTPUT_LOAD_CAPS {
    IGNORE_FANOUT_CAP 1
    Clk_net123 15pF
    Clk* 15p
}
```

The capacitance can be denoted by scientific notation (e-12), or as “p” for pico, “f” for femto, and so on.

PRIMARY_OUTPUT_LOAD_CAPS_FILE

Specifies primary output load capacitance within a file.

Syntax:

```
PRIMARY_OUTPUT_LOAD_CAPS_FILE ./<file_pointer>
```

File contents:

```
<net_name> <load_cap_value_in_F>
```

PS_GENERATE_MCYC_SCENARIO

When set to 1, generates multi-cycle scenarios in VLSG flow, with each cycle having different switching scenario but same power target. A multi-cycle switching scenario for the vectorless instances is computed and written to switching scenario file, *apache.scenario*. When set to 2, improves switching instance coverage and handles the MBFF instances in multicycle VLSG flow. *Default: -1 (off)*

Syntax:

```
PS_GENERATE_MCYC_SCENARIO [-1 | 1 | 2]
```

PS_GENERATE_VLESS_SCENARIO

When set to a value greater than 0, power calculation generates a switching scenario file, *apache.scenario*, for vectorless designs to be used in simulation. This file generates more diverse and realistic switching scenario utilizing the logic view of power calculation, and can be used for future simulation runs. Note also that the following flows and special conditions are not supported in the VLSG flow, and

RedHawk uses the old flow:

- “import apl” or “import avm” command after “perform pwrcalc”
- “mesh *” or “eco *” commands
- “POWER_MODE” is “apl_peak”, “apl_peak1” or “apl1”
- If “NEW_STATE_PROPAGATION” is set to 0, then VLSG is disabled; GSR keywords DYNAMIC_MULTI_SCENARIO_PAIR, and MULTI_CYCLE_SCENARIO, and option ‘-mcycle’ are not supported by VLSG.
- PARA_CALC_POWER is set to 0.

DMP compatible. Optional. Default: -1.

Syntax:

```
PS_GENERATE_VLESS_SCENARIO <integer>
```

PS_GLITCH_POWER_MODELING

When set to a value greater than 0, estimates the glitch power of a vectorless design, based on the Verilog netlist and timing windows of the instances. Total glitch power is then reported in the power summary.

Glitch power represents a larger and larger portion of the power usage as semiconductor geometry gets smaller and smaller. However, for gate level designs, the VCD file often is too large to handle and hence impractical for glitch power computation. For this reason, the necessity of glitch power modeling arises in the absence of gate level VCD data. *Optional. Default: -1.*

Syntax:

```
PS_GLITCH_POWER_MODELING <integer>
```

PS_LIB_EXTRACT_CCS

When turned on, controls the library parser to generate CCS output current and receiver capacitance, which are used in signal EM analysis. When set to 1 it is enabled; when set to 0 it is disabled. *Optional. Default: 0*

Syntax:

```
PS_LIB_EXTRACT_CCS [ 0 | 1 ]
```

PS_RTL_EP_MODE

When set, disables propagation through flip-flops in Event Propagation. *Optional. Default: 0.*

Syntax

```
PS_RTL_EP_MODE [ 0 | 1 ]
```

PS_RTL_EP_REPORT

In the RTL-VCD flow, when turned on, RedHawk generates the following output files:

- *adsRpt/vcd_covered_ffs* : list of ff/latches that are “covered” in VCD/FSDB, meaning those in which one of the outputs is covered.
- *adsRpt/vcd_covered_nets* : list of nets that are covered in VCD/FSDB
- *adsRpt/propagated_insts* : list of instances that are propagated
- *adsRpt/propagated_nets* : list of nets that are propagated through

DMP compatible. Optional. Default: 0, off.

Syntax:

```
PS_RTL_EP_REPORT [ 0 | 1 ]
```

PS_SP_GATED_CLOCK_LOGIC

When turned On, State Propagation auto-detects turn-on ratios at clock gates, even if no value is specified, by using the propagated duty cycle value of the net connected to the enable pin of the clock gate. *Optional. Default: 1 (On).*

Syntax

```
PS_SP_GATED_CLOCK_LOGIC [ 0 | 1 ]
```

PS_VLSG_POWER_DOMAIN_HANDLING

When set, improves the correlation results between DMP and FLAT when power domain number is more than 16.

Syntax

```
PS_VLSG_POWER_DOMAIN_HANDLING [ 0 | 1 ]
```

PARA_CALC_POWER

When set (default), performs power calculation and extraction in parallel threads, which improves performance. If 'import apl' commands are given between 'perform pwrcalc' and 'perform extraction' in TCL command file, these APL files are read in before performing power calculation for better accuracy. Note that this is turned off in DMP power transient flow. *DMP compatible. Optional. Default: 1.*

Syntax

```
PARA_CALC_POWER [ 0 | 1 ]
```

REPORT_MAXCAP_VIOLATION

Reports LIB max cap violations to the file *adsRpt/<design>.maxCapViolation*. When set to -1, all violations are reported. When set to a positive integer, n, the number of violations reported is limited to n. The output report format is:

```
<inst_name:pin> <cell_name> <library> <max_cap> <spef_cap>
```

When both 'USE_LIB_MAX_CAP 1' and 'REPORT_MAXCAP_VIOLATION <limit>' keywords are set, the report is prepared and the maxcap limit is honored in LIB. When only REPORT_MAXCAP_VIOLATION is set, only the max-cap violations are reported, and SPEF/Steiner tree estimates (or whatever is available) are used.

Optional. Default: 0 (off).

Syntax

```
REPORT_MAXCAP_VIOLATION [ 0 | n | -1 ]
```

Example

```
REPORT_MAXCAP_VIOLATION 150
```

RTL_NAME_MAPPING

When set, PowerStream performs RTL-to-gate mapping of net names if not created by the AE TCL script command 'GenerateMapfile'. If mapping is not provided, PowerStream creates map points internally. If a mapping file is provided, PowerStream attempts to create map points for any missing flip-flops. When set to 3, advanced mapping methods are used (default). Intermediate mapping strategies are used when set to 1 or 2. All automatic mapping is turned off when set to 0.

Optional. Default: 3.

Syntax

```
RTL_NAME_MAPPING [ 0 | 1 | 2 | 3 ]
```

SAIF_FILE

Specifies the root instance name and the Switching Activity Interface File that contains design toggle information. *Optional. Default: none.*

Syntax:

```
SAIF_FILE {  
    <block_name> <SAIF_filename>  
    ROOT_INSTANCE <root_instance_name>  
    SUBSTITUTE_PATH <substitute_instance_path>  
}
```

where

<block_name>: can be <design name>, <block name>, or FULLCHIP
<SAIF_filename>: specifies the full path to the SAIF filename
ROOT_INSTANCE: specifies the hierarchical path to the root instance; the <root_instance_name> is the mapping between the SAIF names and the DEF names.

For an example SAIF file:

```
(SAIFILE  
(SAIFVERSION "2.0")  
...  
(INSTANCE tb  
  (INSTANCE ldo_bench  
    (NET
```

```
(A\[0\]
    (T0 70000) (T1 0) (TX 0)
    (TC 0) (IG 0)
)
...
)
)
```

the top DEF design name is : ldo_design, and the GSR setting would be:

```
SAIF_FILE {
    ldo_design user_data/ldo_design.saif
    ROOT_INSTANCE tb/ldo_bench
}
```

SCALE_CLOCK_POWER

Includes scaling of toggle rates for clock network components in the BLOCK_POWER_FOR_SCALING calculation. If set to zero, clock net toggle rates are not scaled.

If specified block power is smaller than the RedHawk computed block leakage power, and SCALE_CLOCK_POWER is set to 1, the power values are scaled uniformly to meet the specified block power. *DMP compatible. Optional. Default: 1.*

Syntax:

```
SCALE_CLOCK_POWER [0|1]
```

SCAN_CLK_DUTY_CYCLE

Specifies the clock duty cycle in the Event Propagation-based scan mode flow. *DMP compatible. Optional. Default: 0.5.*

Syntax:

```
SCAN_CLK_DUTY_CYCLE <clock_duty_cycle>
```

SCAN_CONSTRAINT_FILES

Specifies one or more scan constraint files, and the associated scan pattern. If no pattern is specified, a default pattern of “01” is assumed. When AUTO_TRACING is specified, PowerStream finds all scan chains in the design and applies the specified pattern to those chains. AUTO_TRACING_BLOCK supports per-block scan chain auto-tracing. You can apply different scan patterns to each block of the design, or you can exclude selected blocks from tracing using the EXCLUDE_BLOCK option. AUTO_TRACING_BLOCK_FILE specifies a file that lists the hierarchical blocks to be traced and the patterns to be used. *DMP compatible. Optional. Default: none.*

Usage notes:

- a. Make sure that the pattern specified in front of each block name is the same; otherwise it is an error.
- b. Per-block auto-tracing cannot be used with the keyword AUTO_TRACING, which is for TOP level auto-tracing. If the two keywords are both used, an error message is displayed.
- c. AUTO_TRACING_BLOCK and AUTO_TRACING_BLOCK_FILE cannot be used together with AUTO_TRACING_EXCLUDE_BLOCK and AUTO_TRACING_EXCLUDE_BLOCK_FILE. If they are both used, an error message is displayed.

- d. When tracing scan chains in a block, the tracing stops at the boundary of the block.
- e. The Vectorless scan flow supports synchronizer cells (that is, a latch followed by multiple flops), takes multiple clock cycles, depending upon the number of sequential elements used, to shift the pattern completely out of the cell. The format for this type of scan file is:

```
<synchronizer_instance>/<scan_out_pin> <clock_cycles - 1>
```

Syntax:

```
SCAN_CONSTRAINT_FILES {
    <scan_chain_file_1> <pattern_1>
    ? <scan_chain_file_2> <pattern_2>?
    ...
    ? AUTO_TRACING <pattern>?
    ? AUTO_TRACING_BLOCK <hier_block_name>
        [<pattern> | <patternfile>]?
    ? AUTO_TRACING_BLOCK_FILE <path_to_file> ?
    ? AUTO_TRACING_EXCLUDE_BLOCK <hier_block_name>
        [<pattern> | <patternfile>]?
    ? AUTO_TRACING_EXCLUDE_BLOCK_FILE <path_to_file> ?
}
```

Example

Multiple scan chains, each has its own pattern:

```
SCAN_CONSTRAINT_FILES {
    scan_chain_1.spc
    scan_chain_2.spc 101
    scan_chain_3.spc 0011
    ...
}
```

SCANMODE

Specifies whether scan pin power is included in vectorless power calculation or not.
Optional. Default: 0.

Syntax:

```
SCANMODE[ 0 | 1 ]
```

SCAN_PI_CONSTRAINT_FILE

When specified for the vectorless scan flow, user-defined values are propagated from the specified primary inputs, rather than using event propagation to determine the switching scenarios of downstream logic. The format of <pi_file> is:

```
<primary_input1>
<primary_input2>
...
```

The <pi_file> contains a list of PIs that should be constant 1 during simulation. The rest of the PIs (except Clock, Scan enable, Primary scan input; if it directly connects to the first scan resistor through buffers/inverters) are kept at 0. *Optional. Default: none.*

Syntax:

```
SCAN_PI_CONSTRAINT_FILE <pi_file>
```

SCAN_SHIFTIN_LSB

Controls STIL file patterns that get shifted into the scan chain in scan capture mode. By default, it is shifted from LSB. To shift from MSB, set this keyword to 0.

Default: 1.

Syntax

```
SCAN_SHIFTIN_LSB [ 1 | 0 ]
```

SP(CG)_CONSTRAINT_FILE

Specifies a constraint file listing clock instances and their gating ratios for state propagation. the format of the file is:

```
<instance_name> <gating_ratio>
```

When specified, state propagation reads the file and implements the gating ratio for each instance (range 0.0 to 1.0), which affects the clock network and state propagation results. *Default:* none.

Syntax

```
SP(CG)_CONSTRAINT_FILE <CG_defin_file>
```

SP_CLOCK_GATING_ERROR_OUT_RATE

PowerStream errors out by default when more than 20% of the clock gates have incorrect formats in the GATED_ON_PERCENTAGE constraint file (GCC file).

When this keyword is set to 1, the feature is disabled. *DMP compatible. Optional.*
Default: 0.

Syntax

```
SP_CLOCK_GATING_ERROR_OUT_RATE [ 0 | 1 ]
```

SP_CLOCK_GATING_RATIO

Controls the global gating ratio for all clock gates in the design, where the <ratio> setting must be between 0 and 1.0. By default the clock gating ratio is automatically set by RedHawk. *Optional. Default:* -1.0.

Syntax

```
SP_CLOCK_GATING_RATIO <ratio>
```

SP_LIMIT_TR

Provides realistic toggle rate values in state propagation. Instance frequency and net frequency (from STA file) may be different in some cases, resulting in an unrealistic toggle rate, and hence create a high power number for the instance. This keyword adjusts the toggle rate setting of instances to a realistic value. When set to 0, this feature is turned off. *Optional. Default:* 1.

Syntax:

```
SP_LIMIT_TR [ 0 | 1 ]
```

STATE_PROPAGATION

STATE_PROPAGATION controls the operation of state propagation analysis for vectorless analysis. State propagation is designed to determine the probability that instances will switch during the simulation window of interest, based on a long simulation time. If this keyword is *not* defined, state propagation is off and instance switching is determined by default toggle rates in the design. If STATE_PROPAGATION is defined, when the Power Calculation command is

invoked, state propagation is performed first and the toggle rates obtained from state propagation are then used as the basis for calculating instance power consumption in the design. The analysis includes fine-grained switching coverage, support for power transient flow, and enabling different sets of Clock Gates in different frames.

There are two ways to handle clock gates-- Automatic Clock Gate Handling using a Ratio-based Flow and the CLOCK_GATE_ENABLE_RATIO option, and an On-Off based flow using the FLOP_ON_PERCENTAGE option. See [section "Gated Clock Dynamic Analysis", page 5-94](#) for more details about the State Propagation flow.

DMP compatible. Required for state propagation analysis.

Syntax

```
STATE_PROPAGATION {
    PROPAGATION_MODE probability
    CLOCK_GATE_ENABLE_RATIO <ratio>
        ENABLE_CASCADED_CLOCK_GATING [0|1]
        INFERRRED_CLOCK_GATE_ENABLE_RATIO <ratio>
        CLOCK_GATE_ENABLE_RATIO_FILE <cg_ratio.file>
        FLOP_ON_PERCENTAGE <On_fraction> ?
            GATED_CONTROL_FILE <control_filename> ?
            FOP_CONTROL_FILE <cg_on_off.file>
            ? GATED_CLOCK_COVERAGE [0|1]?
            ? SEQ_CELL_MODE [1|0] ?
}
```

where

PROPAGATION_MODE probability - automatically determines switching probability for instances.

CLOCK_GATE_ENABLE_RATIO : specifies the probability that any clock gate is On, applied to all clock gates in the design, and is the value used to scale the clock toggle rate down. Otherwise, the clock gate enable rate is derived by propagating upstream toggle rate values from the primary inputs, so the default gate enable ratio is set at the output of the clock gates.

ENABLE_CASCADED_CLOCK_GATING : the gating ratio is cascaded, meaning that the gating ratio at any gate depends on the upstream gates and their probability of switching

INFERRRED_CLOCK_GATE_ENABLE_RATIO : specifies the gate enable ratio for inferred gates only, where inferred Clock gates are instances such as AND gates, which can be used as clock gates. If you do not want inferred clock gates to affect the clock toggle rate, set this value to 1.

CLOCK_GATE_ENABLE_RATIO_FILE : specifies a CGER filename, in which custom clock gate enable ratio values can be specified for desired clock gates, using the format:

```
#<clock_gate_name> <enable_ratio>
```

```
...
```

FLOP_ON_PERCENTAGE <On_fraction>: specifies the fraction, between 0 and 1.0, of flip-flop gates are on, as follows:

All FF cells are On if FLOP_ON_PERCENTAGE is 1.

All FF cells are Off if FLOP_ON_PERCENTAGE is 0.

Increasing the value of FOP turns on additional gates, and does not turn off any that are already turned on. So, any FF that is on with a GOP of, for example, 0.6, is also on when FOP is set to 0.8.

If FOP is not set, any clock gate not otherwise specified has a default FOP value of 1 (On).

GATED_CONTROL_FILE : specifies a file defining the On/Off status of individual gated clock control cells. The format of the gated clock control file data is as follows for both static and dynamic analyses:

```
<gated_clock_instance> [ On | Off ]
```

...

GATED_CLOCK_COVERAGE : when set to 1, provides better design coverage when using FLOP_ON_PERCENTAGE, and allows choosing different clock gate turn-on scenarios for each frame so that overall analysis coverage is increased. Note that POWER_TRANSIENT_ANALYSIS must also be turned on.

SEQ_CELL_MODE : enables logic propagation for flip-flop and latch cells (default 1)

STEINER_TREE_CAP

When specified, Steiner Tree routing is performed and the resulting length is multiplied by the capacitance density value specified, in pF per um. *DMP compatible. Optional. Default: none.*

Syntax:

```
STEINER_TREE_CAP <cap in pF per um>
```

Example:

```
STEINER_TREE_CAP 2.0e-4
```

SUBSTRATE_PGARC

When set to 1, creates P/G arcs between the bulk PG pins even in the default mode i.e. even if user does not extract the substrate network and thereby improves current distribution in bulk domains. *Default:0.*

Syntax:

```
SUBSTRATE_PGARC [ 1 | 0 ]
```

THERMAL_ANALYSIS

Specifies the type of thermal analysis to run.

- THERMAL_ANALYSIS 0

If the GSR keyword CPA_FILES is set, and also 'THERMAL_ANALYSIS 0', a *redhawk_cpa* license is checked out, and only CPA setup and analysis are allowed.

- THERMAL_ANALYSIS 1

If the GSR keyword CPA_FILES is set, and also 'THERMAL_ANALYSIS 1', a *redhawk_cta* license is checked out, and only CTA setup and analysis are allowed.

- THERMAL_ANALYSIS 2

If the GSR keyword CPA_FILES is set, and also 'THERMAL_ANALYSIS 2', *redhawk_cta* and *redhawk_cpa* licenses are checked out, and both CTA and CPA setup and analysis are allowed.

Optional. Default: 0.

Syntax:

```
THERMAL_ANALYSIS [ 0 | 1 | 2 ]
```

THERMAL_MODEL

For CTM (Chip Thermal Model) generation and power calculation, this keyword specifies the way leakage power is to be handled, using values 0, 1 or 2, as follows:

- 0 - no temperature-dependent leakage power is considered (default). When the THERMAL_PROFILE GSR keyword is specified, the value of THERMAL_MODEL is automatically changed to 1.
- 1 - no temperature-dependent leakage power is considered for cells not defined in a user-specified leakage current file (using the APL_FILES GSR keyword)
- 2 - temperature dependency for cells not defined in a leakage file in APL_FILES is taken from temperature-dependent scaling factors for cells included in leakage current files in APL_FILES. It is recommended that all cells are defined in <cell>.cdev files.

Optional. Default: 0.

Syntax:

```
THERMAL_MODEL [ 0 | 1 | 2 ]
```

TOGGLE_RATE

Defines the default signal toggle rate of the nets on the chip that are not otherwise specified. The rate is the product of the probability that the nets will toggle times the actual clock toggle rate. Toggle rate is defined as the sum of the state changes from 0->1 and 1->0 within a clock cycle with respect to the net's clock domain. For example, a clock net generally has a toggle rate of 2.0 with respect to its clock domain, since the net switches once from 0->1 and once 1->0 within a clock cycle. Note that if there is no power consumption table in .lib the toggle rate is taken from TOGGLE_RATE, and charge is scaled to meet power. *DMP compatible. Optional. Default: non-clock: 0.3; clock: 2.0.*

Syntax:

```
TOGGLE_RATE <non_clock_TR> ?<clock_network_TR>?
```

Example:

```
TOGGLE_RATE 0.2 1.0
```

where

<non_clock_TR> : defines the probability for nets switching during a clock cycle
<clock_network_TR>: applies to both clock pins and to clock buffer outputs-- the actual network clock toggle rate (Default: 2.0)

TOGGLE_RATE_RATIO_COMB_FF

Defines the ratio of the toggle rate of the nets driven by combinational cells relative to the toggle rate of the nets directly driven by flip-flops and latches. Depending on the depth of the combinational logic, the ratio can be from 0.3 to 0.75. Only the toggle rate of the combinational instances specified by TOGGLE_RATE, BLOCK_TOGGLE_RATE, BLOCK_TOGGLE_RATE_FILE, INSTANCE_TOGGLE_RATE, and INSTANCE_TOGGLE_RATE_FILES are affected by this keyword. The toggle rate defined by the VCD_FILE keyword is *not* affected. *DMP compatible. Optional; default: 1.0.*

Syntax:

```
TOGGLE_RATE_RATIO_COMB_FF <TR_ratio_combin_re_FF>
```

Example:

```
TOGGLE_RATE_RATIO_COMB_FF 0.7
```

USE_FAST_DECAP_ALG

When set, invokes “Special Decap Handling” (SDH) flow, which provides a reduction in the number of nodes/resistors created for decap instance pins, and improves performance in filler and decap-dominated designs. Auto-detection of decap cells is disabled and all the decap cells need to be specified under ‘DECAP_CELL’ GSR keyword. Decap instances in unconnect files (*.GND.unconnect, *.VDD.unconnect and *.PG.unconnect) are also reported. *DMP Compatible. Optional. Default: 1 On. (except set Off by default when the MPR_MODE keyword is set).*

Syntax:

```
USE_FAST_DECAP_ALG [ 0 | 1 ]
```

USE_LIB_MAX_CAP

In early design stages, when clock tree synthesis is not available, instances that drive very large fan-out, such as dummy clock buffers, may have unreasonably high calculated power and IR drop because of the high fanout. If USE_LIB_MAX_CAP is turned on, RedHawk uses the cell maximum capacitance value (max_cap) in the .lib file, instead of the calculated combination of interconnect and fanout capacitance, to achieve a more realistic early value for Cload. *DMP compatible.*

Optional. Default: 0 (Off)

Syntax:

```
USE_LIB_MAX_CAP [ 0 | 1 ]
```

USER_MCF_FILE

Allows you to specify the Memory Configuration File, in order to calculate memory power more accurately. The MCF, .apache/apache.mcf, defines clock and data timing. *DMP compatible. Optional. Default:*

Syntax:

```
USER_MCF_FILE <MCF_file>
```

VCD_PREPARE_SCENARIO

Supports design flow when several VCD/FSDB files are used, and avoids processing the VCD/FSDB file more than once in the 'perform analysis -vcf' stage. Even in Averaged mode the switching scenario file generated during power calculation is reused during simulation. *DMP compatible. Optional. Default: 1 (On).*

Syntax

```
VCD_PREPARE_SCENARIO [ 0 | 1 ]
```

VCD_SCENARIO_COMPRESS

When set, compresses the apache.scenario file in gzip format in both power calculation and simulation. Note that this keyword works only when the START_TIME/END_TIME options are specified using the VCD_FILE GSR keyword. *DMP Compatible. Optional. Default: 0 (off)*

Syntax

```
VCD_SCENARIO_COMPRESS [ 0 | 1 ]
```

VCD_TIME_ALIGNMENT

Allows the ability to align multiple VCDs in hierarchical VCD flow. When Off (0), specifies that VCD start times are not aligned and independent cycle selection for

each VCD is performed. By default (1), VCD start times are aligned and cycle power ranking based on total power is performed.

For DPDT-related cycle selection (set under the SELECT_TYPE option of VCD_FILE), for 'VCD_TIME_ALIGNMENT 0', different SELECT_TYPE values are allowed among multiple VCDs, and for 'VCD_TIME_ALIGNMENT 1', the SELECT_TYPE value should be consistent among all VCDs. *DMP compatible.*
Optional. Default: 1.

Syntax:

```
VCD_TIME_ALIGNMENT [ 0 | 1 ]
```

VCD_X_LOGIC_STATE

RedHawk has the flexibility to handle X-states in VCD to meet requirements, based on the setting of VCD_X_LOGIC_STATE, as follows:

When set to -1, logic-to-X is ignored and X-to-logic toggles are considered (default).

When set to 0, X is always treated as 0.

When set to 1, X is always treated as 1.

When set to 2, both logic-to-X and X-to-logic are considered a toggle.

When set to 3, X -to-logic is considered as transition, if next logic state is different from previous logic state. Thus 1-> X -> 1 and 0 -> X -> 0 is ignored, while 1 -> X -> 0 and 0 -> X -> 1 are considered as 1 transition.

DMP compatible.

Syntax:

```
VCD_X_LOGIC_STATE [ -1 | 0 | 1 | 2 | 3 ]
```

VECTORLESS_BLOCK

The mixed mode flow supports the scenario in which the top design has VCD, but some of the blocks need to be simulated in vectorless mode. You can specify block names for the VCD that is unavailable using this keyword. *DMP compatible.*
Optional. Default: none.

Syntax

```
VECTORLESS_BLOCK {
    <Block1>
    <Block2>
    ...
}
```

Electromigration Keywords

ANALYZE_NETS

Specifies signal nets to be selectively extracted, simulated and checked with EM rules for improved performance in the signal EM analysis flow. Note that this keyword *cannot* be set using the TCL command “gsr set ANALYZE_NETS {}”.
DMP compatible. Optional. Default: none.

Syntax

```
ANALYZE_NETS {
    <signal_net_name_1>
    <signal_net_name_2>
    ...
}
```

}

ANALYZE_NETS_FILE

An extension to the keyword ANALYZE_NETS, this keyword specifies a file name that lists signal nets to be extracted, simulated and checked with EM rules in the signal EM analysis flow. Note that this keyword *cannot* be set using the TCL command “gsr set ANALYZE_NETS_FILE { }”. *DMP compatible. Optional. Default: none.*

Syntax:

```
ANALYZE_NETS_FILE <file_with_signal_nets_names>
```

BRIDGE_WIRE_CONNECTION

When set honors bridge connection wires (do not touch wires on the same layer, but connect to wires on other layers through vias).

Syntax:

```
BRIDGE_WIRE_CONNECTION 1
```

CONFIGURABLE_REPORT_FILE

Specifies the configurable Signal/Power EM report filename, which specifies the data desired in the report, and their order. The default list of items that are included in the configuration file for each type of analysis is as follows:

Default Contents of Report file for power EM:

```
EM_WORST_POWER
WIRESegment $LAYER $STARTX $STARTY $ENDX $ENDY $EM_RATIO $NET_NAME $WIDTH
$CURRENT $BLECH_LENGTH
VIA $VIA_NAME $CX $CY $EM_RATIO $NET_NAME $CURRENT $BLECH_LENGTH
OPTIONS $GROUP_BY_FIRST_COL
OPTIONS $EM_NETS_ONLY
OPTIONS $USE_OLD_FORMAT
```

Default Contents of Report file for signal EM:

```
EM_WORST_SIGNAL
WIRESegment $LAYER $STARTX $STARTY $ENDX $ENDY $EM_RATIO $NET_NAME $WIDTH
$CURRENT $EM_LIMIT $CURRENT_DIR $FREQUENCY
VIA $VIA_NAME $CX $CY $EM_RATIO $NET_NAME $CURRENT $EM_LIMIT $CURRENT_DIR
$FREQUENCY
OPTIONS $GROUP_BY_FIRST_COL
```

The keyword “OPTIONS \$GROUP_BY_FIRST_COL” in the configurable report file allows grouping values for any parameter specified in the first column together. So when this keyword is used, whatever parameter is listed in the first column of the configurable report file, the entries are grouped together in the output reports. In the case of \$NET_NAME in the first column, for example, the generated .em.worst files are organized with wire segment EM values together.

The keyword “OPTIONS \$EM_NETS_ONLY” under DUMP_RES_NETWORK section in config file excludes wire segments or vias without EM limits from appearing in res_network. The option “USE_OLD_FORMAT” can be used if user wants user to retain the original/default header of the ‘dump res_network’ when options such as EM_NETS_ONLY or INCLUDE_WIRE_STUBS are used.

DMP Compatible. Optional. Default: none.

Syntax:

```
CONFIGURABLE_REPORT_FILE <config_filename>
```

DELTA_T_RMS_EM

Specifies the expected wire temperature increase due to Joule heating when calculating peak RMS EM current in tech file 'metal' section, using the 'delta_T' option in POLYNOMIAL_BASED_EM_RMS. *DMP compatible. Optional. Default: none.*

Syntax:

```
DELTA_T_RMS_EM <temp_increase_deg C>
```

DETAILED_CONN_ISSUE_RPT

When set, reports nets that are unconnected to the receiver instance pin, or the receiver LEF is missing, in the *adsRpt/SignalEM/connectivity_issue.rpt* file. *Optional. Default: 0.*

Syntax:

```
DETAILED_CONN_ISSUE_RPT [ 0 | 1 ]
```

EM_CHECK_2D

Controls perpendicular direction EM checking. When set, checks current in both directions. For dump resistor network values to be consistent with the GUI and EM reporting, set 'EM_CHECK_2D 0' so that there is only one EM value for each wire. *DMP compatible. Optional. Default: 1, On.*

Syntax:

```
EM_CHECK_2D [ 0 | 1 ]
```

EM_CUSTOM_CURRENT_FILE

You can use EM_CUSTOM_CURRENT_FILE to create a custom current file to specify the current at signal net driving points. Using the custom current file you can also limit EM analysis to those nets that are specified in the file. Only nets specified in file are reported if you also set the GSR keyword 'EM_CCF_ONLY' to 1.

The format of the custom current file is as follows:

```
<net_name> INST <inst_name> <pin_name> <avg> <rms> <peak>
...
<net_name> CELL <cell_name> <pin_name> <avg> <rms> <peak>
...
```

where

net_name: specifies the net name

INST <inst_name>: specifies the instance name

pin_name: specifies the pin name

<avg> <rms> <peak>: specifies the current value for each type of current, in Amps. Current types that are not to be set should be given values of '-1'.

Example custom current file contents:

```
net1 INST ANA1 VREGO -1 -1 1e-3
net2 INST ANA1 SIGPIN -1 -1 2e-3
```

The example sets PEAK current to '1e-3' for net1 and '2e-3' for net2 at driving points VREGO and SIGPIN, respectively. Other current values are unspecified. *Optional. Default: None*

Syntax:

```
EM_CUSTOM_CURRENT_FILE <filename>
```

Example:

```
EM_CUSTOM_CURRENT_FILE abc2.ccf
```

EM_CCF_ONLY

When specifying an EM_CUSTOM_CURRENT_FILE, only nets specified in the file are reported if you also set 'EM_CCF_ONLY' to 1. *Optional. Default: 0 (off).*

Syntax:

```
EM_CCF_ONLY [ 1 | 0 ]
```

EM_DENSITY_ANALYSIS_LAYERS

Specifies the layer names for which user needs to do wires grouping for EM analysis, wire conditioning and to identify choke points.

Syntax:

```
EM_DENSITY_ANALYSIS_LAYERS {  
    layer1  
    layer2  
    ...  
}
```

EM_DUMP_PERCENTAGE

EM_DUMP_PERCENTAGE specifies the range of EM ratio percentages to be reported in the results file *adsRpt/Static/*.em*, where EM_Ratio = actual_current_density/ current_density_limit. The *.em file reports EM-related parameters for wires and vias, but is not generated if EM_DUMP_PERCENTAGE is not specified. *DMP compatible. Optional. Default: None*

Syntax:

```
EM_DUMP_PERCENTAGE [ <min_percent> | <min_percent>-<max_percent> ]  
where <min_percent> defines the minimum EM ratio percentage to be reported and  
<max_percent> is the maximum EM ratio percentage to be reported. If only  
<min_percent> is specified, all EM ratio percentages higher than <min_percent>  
are reported. Values of <min_percent> and <max_percent> are integers.
```

Example:

```
EM_DUMP_PERCENTAGE 0-20
```

For wires, the line format for EM data in the *.em file is as follows:

```
<layer> <segment_end_coords> <EM_Ratio> <Current_value> <netname> <width>
```

For vias, the line format for the EM data is:

```
<via_name> <x-y_coord> <EM_Ratio> <current_value> <netname>  
      <via_cut_bounding_box_coords> <current_direction>
```

Current values are in Amps, and coordinates and dimensions are in um. To include values of <via_cut_bounding_box_coords> and <current_direction> in the report, users must set the GSR keyword PRINT_EM_VIA_BOX to 1.

EM_LENGTH_USE_MAX_LENGTH

When set, specifies wire length for EM rule lookup is defined based on all connected wire segments and terminating at a via, using the segment centerline. The calculation rules are:

- When set to “0”, RedHawk uses the total metal length for EM rule lookup, that is, all connected segments (default).
- When set to “1”, RedHawk uses the segments representing the longest metal length for EM rule lookup, as shown in the figure below.

DMP compatible. Optional. Default:0.



Syntax:

EM_LENGTH_USE_MAX_LENGTH [0|1]

EM_MISSION_PROFILE

Specifies a file used to define temperature-dependent coefficients to relax EM limits for metal and vias. By default all coefficients are set to 1, in which case no scaling takes place. Supports also two-dimension EM mission tables corresponding to Temperature and Life_Cycle for defining coefficients to relax EM limits for metal wires and vias. There is no interpolation (must match exactly) for Temperature/Life_Cycle values. If the option "REGULAR_TEMP_OFF" is specified in EM_Mission profile, then the temperature derating is disabled and EM Mission Temperature / Life_Cycle / Fail_rate factor is applied to EM limits. *Optional. Default: none.*

Syntax:

EM_MISSION_PROFILE <path_to_file>

Contents of a sample mission profile are shown below:

```
via VIA12 {
    { 0 5 }
    { 25 10 }
    { 105 100 }
    { 110 0.697 }
    { 125 0.25 }
    { 150 0.053 }
}
metal METAL1 {
    { 0 5 }
    { 25 11 }
    { 105 111 }
    { 110 0.697 }
    { 125 0.20 }
    { 150 0.053 }
}
```

In the above example, at a temperature of 125° C, the EM limit for METAL1 is scaled by a factor of 0.20.

Two dimensional example mission profile:

```
metal M1 {
    Temperature { 0 25 50 100 125 }
    Life_Cycle { 1000 2000 3000 }
    VALUES {
        { 1 2 3 4 5 }
        { 1 2 3 4 5 }
        { 1 2 3 4 5 }
```

}

EM_MODE

Specifies the type of EM results to be displayed from dynamic analysis. The true average, root-mean-squared, or peak value of current over the simulation time is compared to the specified EM limit. This keyword does not change static analysis methodology, which provides true average current values only. *DMP compatible.* *Optional. Power EM default: peak; Signal EM default: rms.*

Syntax:

```
EM_MODE [ avg | rms | peak ]
```

Example:

```
EM_MODE peak
```

EM_NET_INFO

Defines a file specifying toggle rates and/or transition times, uni-directional and bi-directional current scaling factors, and extra receiver capacitance (“extra-cap”) per signal net, and specify the slew (transition time) range for all nets, including nets that are set as CONST in the STA file. These values override the global values of TOGGLE_RATE and EM_SLEW_* keywords, as well as any toggle rate values set in power calculation or transition times in the STA file. The uni-directional scale entries must be entered in column 5, bi-directional scale entries in column 6 and the extra capacitance in column 7. The I(rms) and I(Peak) of the primary output nets can be calculated using the Ceff values of the primary output nets. To enable this, you must provide Ceff values for the nets in the EM_NET_INFO file. You can input the slew range as a comment in the header, such as “# slew_range <value>”, where <value> is any floating point number between 0 and 1. Columns 3 ~ 12 values are optional, but all columns to the left of a specified value must have at least a “-” placeholder. Column 12 lists the voltage values to be applied to the input net names. The file format is as follows:

```
#<net_name> <trans_time_sec> <toggle_rate> <freq>
<uni-dir_scale> <bi-dir_scale> <extra-cap> <driver_cell>
<pin> <Ceff> <-> <voltage>

# slew_range 0.8
net1 1e-13 2 2e09 0.7 - 1e-14 - - - 5 1.2
...
```

In the example above, 'net1' is assigned a transition time of 1e-13 seconds, within the slew range of 80%. It's also assigned a toggle rate of 2 and frequency of 2 GHz. Its uni-directional currents are scaled by 0.7. No bi-directional scaling is used, but an extra receiver side load pin cap of 1e-14 F is applied. Wild cards for <net_name> are allowed. *DMP compatible.* *Optional. Default: none.*

Syntax:

```
EM_NET_INFO <filename>
```

EM_REPORT_MINWIDTH

Allows modifying the default minimum wire width for reporting EM violation. When this keyword is set, EM violations are reported for wires whose width is more than the specified value. *Optional. Default: 0.005um.*

Syntax:

```
EM_REPORT_MINWIDTH <min_width_um>
```

EM_REPORT_MODE_ONLY

The speed of signal EM analysis can be increased by generating only the report desired using EM_REPORT_MODE_ONLY. The EM_MODE keyword determines the type of report displayed: average, rms (default), or peak, but all three types of reports are generated. If you turn EM_REPORT_MODE_ONLY On (1), only the selected report is generated, which for 'rms' means that only the file `<design>.rms_em.worst` is generated (under the directory `adsRpt/SignalEM`).
Optional. Default: 0 (off)

Syntax:

```
EM_REPORT_MODE_ONLY [ 0 | 1 ]
```

Example:

```
EM_REPORT_MODE_ONLY 1
```

EM_REPORT_PERCENTAGE

Specifies the portion of calculated EM current ratios that should be reported in the `esd_em.rpt` file. For each wire segment and via RedHawk reports the ratio of the current in it relative to the “critical” current value set by the user in the `.tech` file, which is the “EM percentage”. So if the EM_Report_Percentage is set to 50, all calculated EM current ratios that exceed 50 percent are reported (up to the specified Report_Line_Number limit). However, note that at least the worst EM percentage value per wire or via is reported when the worst EM percentage values are less than the specified EM percentage limit. If set to -1, all percentage values are reported. *DMP compatible. Optional. Default: 100 (percent)*

Syntax:

```
EM_REPORT_PERCENTAGE <percentViolationThresh>
```

Example:

```
EM_REPORT_PERCENTAGE 80
```

EM_REPORT_PERCENTAGE_BY_LAYER

Specifies the portion of calculated EM current ratios that should be reported by specified layer(s). For each specified layer, RedHawk reports the ratio of the current in wire segments and vias relative to the “critical” current value set by the user in the `.tech` file, which is the “EM percentage” in %. If the layer is not included in the block, then the global value specified by the keyword “EM_REPORT_<mode>_PERCENTAGE <percent_of_limit> is used. If there is no mode specified, the specified percent of limit values are applied to all three modes.
Optional. Default: none.

Syntax:

```
EM_REPORT_PERCENTAGE_BY_LAYER ?[AVG| RMS| PEAK] ?{  
    <layerName1> <percent_of_limit1>  
    <layerName2> <percent_of_limit2>  
    ...  
}
```

EM_REPORT_<mode>_PERCENTAGE

Sets the threshold percentage of the EM limit for reporting DC, RMS, and PEAK mode EM violations, each of which creates a file. You can set the percentage to any positive value. When set to -1, RedHawk reports all EM violations. The default value (-1000) means that this keyword value has not been set by users, and the value of the EM_REPORT_PERCENTAGE GSR keyword is then used (default 100). The output violation reports are:

```
<cell>.em.worst.avg
<cell>.em.worst.rms
<cell>.em.worst.peak
```

The reports list EM values exceeding the specified limits for each mode. Individual mode limits that are not specified default to the limit in the EM_REPORT_PERCENTAGE keyword. *DMP compatible. Optional. Default: -1000.*

Syntax:

```
EM_REPORT_DC_PERCENTAGE <percent_of_limit>
EM_REPORT_RMS_PERCENTAGE <percent_of_limit>
EM_REPORT_PEAK_PERCENTAGE <percent_of_limit>
```

EM_REPORT_LINE_NUMBER

Specifies the maximum number of lines to be reported in the output EM current ratio report, based on the EM_REPORT_PERCENTAGE keyword specification, for both power and signal EM analysis, or no limit to number of lines (-1). *DMP compatible. Optional. Default: 1000.*

Syntax:

```
EM_REPORT_LINE_NUMBER [<Max_lines> | -1]
```

Example:

```
EM_REPORT_LINE_NUMBER -1
```

EM_SCALE_DC

EM_SCALE_PEAK

EM_SCALE_RMS

Specifies a multiplier for the normal DC, peak, and RMS currents to be used for EM analysis in CD checking. *DMP compatible. Optional. Default: 1.0.*

Syntax:

```
EM_SCALE_DC <scale_value>
EM_SCALE_PEAK <scale_value>
EM_SCALE_RMS <scale_value>
```

EM_SLEW_NO_STA

Specifies the behavior of signal EM analysis when

- a net is driven by a cell, and
- the net has no STA statistical timing information.

The GSR EM_SLEW_NO_STA keyword has three arguments representing three different ways of modeling STA data:

- 'ideal' (default) - means that the driver is considered ideal, with no internal resistance, so the transition time is very short. So if a net has a significant load (large capacitance) this generates large currents, particularly in PEAK current values.
- 'derived' - means that the transition time is computed the same as if there were no driver driving the net. So any values set by other related GSR keywords-- for example, EM_SLEW_SIG_TIME and EM_SLEW_SIG_PERCENTAGE-- are taken without any changes.
- 'input_transition' - means that the argument from GSR keyword INPUT_TRANSITION (which has a default of 100ps) is used as the value for the missing STA.data. Note the difference in this option compared to 'derived';

the 'derived' value is NOT considered a replacement for STA, but is taking the time argument directly. Fundamentally, the 'input_transition' value still takes the actual net load into account and computes from both the net load and the given INPUT_TRANSITION number its own transition time in a non-linear fashion.

(Note that signal nets that are not driven at all can also be considered, but this keyword is not applicable to them.) *DMP compatible. Optional. Default: ideal.*

Syntax:

```
EM_SLEW_NO_STA [ ideal | derived | input_transition ]
```

Example:

```
EM_SLEW_NO_STA derived
```

EM_TECH_DC

EM_TECH_PEAK

EM_TECH_RMS

Specifies files that define the DC, peak, and RMS current density EM limits by layer to be used in calculating potential EM violations. The data in the specified EM files has the same syntax as the corresponding EM-related 'metal' and 'via' keywords in the tech file. If this keyword is not defined, the EM limit values are taken from the tech file. *DMP compatible. Optional. Default: tech file.*

Syntax:

```
EM_TECH_DC <EM_ave_limit_filename>
EM_TECH_PEAK <EM_peak_limit_filename>
EM_TECH_RMS <EM_RMS_limit_filename>
```

EM_TECH_FILE

Specifies a file that defines the AVE, PEAK, and RMS current density EM limits by layer to be used in calculating potential EM violations. The data in the specified EM file has the same syntax as the corresponding EM-related 'metal' and 'via' keywords in the tech file. The EM_TECH_FILE can contain any number of rule sets, separated by a header line that gives the rule set name. *Optional. Default: tech file.*

Syntax:

```
EM_Tech_FILE <EM rule set file>
```

EM_TOPOLOGY_USE_ELECTRON_FLOW

Determines which wire width is used for EM calculation by the Topology-Based EM rule for vias. By default, RedHawk uses the width of the wire that **current** flows to for the calculation. When set to 1, RedHawk uses the width of the wire that **electrons** flow to for the calculation. *Optional. Default: 0.*

Syntax:

```
EM_TOPOLOGY_USE_ELECTRON_FLOW [ 0 | 1 ]
```

ENABLE_AUTO_EM

When set to 1, automatically performs EM checking during regular post-simulation processing. *Starting in v13.2, by default you must execute a separate 'perform emcheck' step. Optional; Default: 0.*

Syntax

```
ENABLE_AUTO_EM [ 0 | 1 ]
```

ENABLE_BLECH

By default the Blech length filtering function is on, which considers potential EM violations for circuit segments that have a Blech product less than the critical BLECH_JLC product value specified in the tech file. RedHawk uses a conservative approach in this calculation, using the worst case current density J and worst case sum of wire lengths for each group of connected and active wire segments on a layer. That is, the wires considered are all on the same layer directly connected to each other, and in an active part of the subnet (no stubs). If these conditions on the wires in the group are satisfied, their EM values are filtered out. *DMP compatible*. *Optional. Default: 1(On)*.

Syntax:

```
ENABLE_BLECH [ 0 | 1 ]
```

ENABLE_POLYNOMIAL_EM

Supports equation-based EM without using the Blech effect, which enables you to use polynomial-based EM when ENABLE_BLECH is set to 0. EM values are still calculated based on the polynomials defined in the tech file. The different setting conditions are shown following. *DMP compatible*. *Optional. Default: 0*:

<u>ENABLE_BLECH</u>	<u>ENABLE_POLYNOMIAL_EM</u>	<u>Polynomial EM used</u>
0	0	NO
0	1	YES
1	don't care	YES

Syntax:

```
ENABLE_POLYNOMIAL_EM [ 0 | 1 ]
```

EQUAL_POTENTIAL_AROUND_PAD

When set to 1, RedHawk will treat the largest continuous square region around the pad as an equipotential region. *DMP Compatible. Default: 0*.

Syntax:

```
EQUAL_POTENTIAL_AROUND_PAD [ 0 | 1 ]
```

IGNORE_HALF_NODE_SCALE_FOR_EM

When set, uses original design dimensions and ignores half node scale factor for computing EM limit violations. *DMP compatible. Optional. Default: 0 (Off)*

Syntax:

```
IGNORE_HALF_NODE_SCALE_FOR_EM [ 0 | 1 ]
```

IGNORE_INST_POSTPROCESS

When set, for EM only runs, disables instance static/dynamic instance voltage drop post-processing, and produces only EM results. *Optional. Default: 0 (Off)*

Syntax:

```
IGNORE_INST_POSTPROCESS [ 0 | 1 ]
```

IGNORE_LEF_DEF_SCALE_FOR_EM

When set, turns off the effects of LEF/DEF scaling when doing EM analysis. That is, RedHawk backs out any LEF/DEF scaling applied to the design wires when performing EM analysis. This does not affect other scale factors. This keyword is

allowed only if just one LEF/DEF scale factor has been applied to all cells in the design. *Optional. Default: 0.*

Syntax:

```
IGNORE_LEF_DEF_SCALE_FOR_EM [ 0 | 1 ]
```

NEW_MERGE_WIRE

When set, improves signal EM analysis with better geometry handling of jog patterns in EM checking. *Optional. Default: 0 - Off.*

Syntax:

```
NEW_MERGE_WIRE [ 0 | 1 ]
```

MERGE_ABUTTED_CUTS

For designs with via cuts touching each other, RedHawk by default applies EM limits per via cut based on area, but you can use MERGE_ABUTTED_CUTS to merge these cuts into a single cut to obtain a common EM limit. To use the keyword, provide values in the EMV_VS_AREA table in the Tech file, and then set 'MERGE_ABUTTED_CUTS 1'. *Optional. Default: 0.*

Syntax:

```
MERGE_ABUTTED_CUTS [ 0 | 1 ]
```

PS_LIB_EXTRACT_CCS

When set, Composite Current Source-based Signal EM flow reads CCS libraries to get driver output current waveforms, and receiver capacitances to do Dynamic Signal EM analysis. *Optional. Default: 0.*

Syntax:

```
PS_LIB_EXTRACT_CCS [ 0 | 1 ]
```

SEM_ACCURACY

When set, enables either the Clock Mesh or CCS Dynamic SEM flows. For the Clock Mesh flow, keyword SEM_DRV_CURRENT_FILE must also be specified.

The CCS Dynamic flow supports the reading of driver current and receiver cap info from CCS .lib files for SEM analysis. This flow enables improved accuracy by taking into account the segmented receiver cap model and the driver current waveform captured in CCS lib.

There is also a hybrid flow of CCS Dynamic Signal EM plus the Clock mesh solution. The following GSR keyword settings are required to enable the CCS-based Dynamic Signal EM + Clock Mesh flow;

```
SEM_ACCURACY ccs_clockmesh
SEM_DRV_CURRENT_FILE <drv current file>
```

You must specify the triangular driver output pin current waveform in the driver current file "drvCurrentFile" *Optional. Default: none.*

Syntax:

```
SEM_ACCURACY [clockmesh_moderate | ccs_dynamic ]
```

SEM_ANALYZE_NET_ONLY

The speed of signal EM analysis can be increased by limiting the number of EM objects included and generating only the type of analysis desired. This option only impacts signal net current calculation and EM checking. It does not affect the

extraction stage. The default value ‘ALL’ analyzes all signal nets. ‘LAYER’ and ‘VIA’ options provide a means of filtering analysis results. Only one option value may be specified at a time. (Note that previous name EM_ANALYZE_NET_ONLY still is functional, but will be phased out in a future release.) *Optional. Default: ALL. DMP Compatible.*

Syntax:

```
SEM_ANALYZE_NET_ONLY [ CLOCK | SIGNAL | LAYER | VIA | ALL ]
```

where

- CLOCK: limits the EM report to signal nets that are of type 'clock' (part of a clock tree)
- SIGNAL: limits the EM report to signal nets that are not of type 'clock'
- LAYER: limits the EM report to signal net metal layers, but not vias
- VIA: limits the EM report to signal net vias, but not metal layers
- ALL: reports EM results on all signal nets

Example:

```
SEM_ANALYZE_NET_ONLY LAYER
```

SEM_CONNECT_NETS'

When set in the automatic signal-net connection flow, connects unconnected signal nets for early analysis. *Optional. Default: 0.*

```
SEM_CONNECT_NETS [ 0 | 1 ]
```

SEM_DEFAULT_PARAMETERS

Specifies the input parameters such as the slew (transition time), load, and toggle rates for the primary input, primary output and constant nets used in signal EM analysis. These settings are applicable for all corresponding nets in the design. The available parameter options are:

PRIMARY_INPUT_PIN_SLEW <transition_time_sec> : value to be assigned to any primary input pin does not have slew information from input data.

PRIMARY_INPUT_PIN_SLEW_FILE <filename> : specifies a file that includes net-specific pin slew information in the format:

```
<net name> <transistion_time_sec>
```

```
...
```

PRIMARY_OUTPUT_PIN_CAP <load_cap_Farads> : value to be used when any primary output pin does not have load specified from the input data

PRIMARY_OUTPUT_PIN_CAP_FILE <filename> : specifies a file that includes net-specific load capacitance values in the format:

```
<net name> <load_cap_Farads>
```

```
...
```

Note that specifications in SEM_NET_INFO, STA and USER_STA override these keywords; the keyword PRIMARY_INPUT_PIN_SLEW_FILE has higher priority than the PRIMARY_INPUT_PIN_SLEW keyword. If you want to analyze constant nets (those that are not part of any clock domain), specify the slew and the toggle rate for the constant nets using the following options. The values in

PRIMARY_OUTPUT_PIN_CAP_FILE have priority over

PRIMARY_OUTPUT_PIN_CAP, and SEM_NET_INFO values override both.

CONST_NET_SLEW <transition_time_sec> : specifies the transition time for the constant nets

CONST_NET_TOGGLE <toggle_rate> : specifies the toggle rate for constant nets.

Optional. Default values: none.

Syntax:

```
SEM_DEFAULT_PARAMETERS {
    PRIMARY_INPUT_PIN_SLEW <transition_time_sec>
    PRIMARY_INPUT_PIN_SLEW_FILE <filename>
    PRIMARY_OUTPUT_PIN_CAP <load_cap_Farads>
    PRIMARY_OUTPUT_PIN_CAP_FILE <filename>
    CONST_NET_SLEW <transition_time_sec>
    CONST_NET_TOGGLE <toggle_rate>
}
```

SEM_DRV_CURRENT_FILE

When set, along with keyword SEM_ACCURACY, enables the Clock Mesh flow. You must specify the triangular driver output pin current waveform in the driver current file “*drvCurrentFile*”. The format of the driver current file is as follows:

```
<driver instance name> <output pin name> <offset>
<IpeakNeg> <INeg duration> <IpeakPos> <IPos duration>
```

Example of a driver current file:

```
u_noram/u_dside/u_dbiu/U6101 z 1e-11 -0.1 5e-11 0.1 5e-11
```

Optional. Default values: none.

Syntax:

```
SEM_DRV_CURRENT_FILE <drv current file>
```

SEM_DUTY_RATIO_ROOT

Allows you to specify a root value in the expression for duty ratio. The duty ratio is defined as $r = \text{signal duration} / (1/(f \cdot TR))$, and $I_{peak_new} = I_{peak} * r^{(1/\text{root})}$, where f is the clock frequency and TR is the signal toggle rate. *Optional. Default: 2.*

Syntax:

```
SEM_DUTY_RATIO_ROOT <root_value>
```

Example:

```
SEM_DUTY_RATIO_ROOT 3
```

In this example, $I_{peak_new} = I_{peak} * r^{(1/3)}$

SEM_ENABLE_SHORTS_REPORT

When set, reports all shorts between signal nets in the design in the file *adsRpt/shorts.rpt*. *Optional. Default: 0.*

Syntax:

```
SEM_ENABLE_SHORTS_REPORT [ 0 | 1 ]
```

SEM_EXTRACT_LONG_WIRE

When set, allows control of the length of RC-PI segments in long wires, as specified with the keyword SEM_SPLIT_LONG_WIRE. *Optional. Default: 0.*

Syntax:

```
SEM_EXTRACT_LONG_WIRE [ 0 | 1 ]
```

SEM_HIERARCHICAL_MODE

Signal EM analysis can be performed in hierarchical mode using this keyword to reduce the overall run time and memory requirements. By default, all signal nets are analyzed at the same time. *Optional. Default: 0 (all signal nets).*

Syntax:

```
SEM_HIERARCHICAL_MODE [ 0 | top_only | internal_only ]
```

Where:

0 (default) : all signal nets are analyzed at the same time.

top_only: only the interface nets and the top level nets are analyzed. Interface nets are those that connect standard cells in DEF to the primary I/O's in DEF. The internal nets are ignored in the analysis and are not reported in `<design_name>.droppedSignalNets`

internal_only : only the internal nets that are completely internal to the block will be analyzed. The primary input/output signal nets are ignored in the analysis and are not reported in `<design_name>.droppedSignalNets`

SEM_IGNORE_DISCONNECT

When set, signal Nets with Top DEF pins defined with the same name having only logical connectivity is reported in file `droppedSignalNets`. *Optional. Default: 0.*

Syntax:

```
SEM_IGNORE_DISCONNECT [ 0 | 1 ]
```

SEM_IGNORE_NETS_MISSING_DATA

When set to 1, RedHawk ignores nets for which input data (such as slew, frequency, routing and SPEF) are missing or insufficient, as in the following cases:.

- Driver output pin slew missing from STA.
- Net cannot be traced from STA or CLOCK_ROOTS
- No frequency from STA or CLOCK_ROOTS
- Parasitic data not found from SPEF, or net has no info in SPEF.
- No routing from the NETS section.

These missing/ignored nets are reported in the `adsRpt/SignalEM/*droppednet` report. The `redhawk.log` reports the following warning message whenever nets are ignored in Signal EM.

```
"WARNING(SEM-202): Some of the nets are dropped from Signal  
EM, please check the file adsRpt/Signalem/  
demo.droppedSignalNets for more details"
```

DMP compatible. Optional. Default: 0.

Syntax:

```
SEM_IGNORE_NETS_MISSING_DATA [ 0 | 1 ]
```

SEM_IMPORT_CONNECTED_NETS

When set, imports logically connected nets that are connected to the instances which are connected to the specified net in the design for signal EM analysis. The reporting will still be only for the nets specified under `IMPORT_NETS`. *Default: 0*

Syntax:

```
SEM_IMPORT_CONNECTED_NETS [ 0 | 1 ]
```

SEM_INOUT_PIN_AUTO_SELECTION

Specified for the nets with inout pins only. When set to 1, the pin with the least pin capacitance is automatically selected as the driver in SignalEM flow. If the net has INOUT pins & ports, ports will be assigned as driver and the pins as receiver.

Default: 0

Syntax:

```
SEM_INOUT_PIN_AUTO_SELECTION [ 0 | 1 ]
```

SEM_KEEP_EMPTY_REPORT

RedHawk reports nets that are dropped from the analysis and dirty nets in the following report files.

adsRpt/SignalEM/<design>.droppedSignalNets

adsRpt/SignalEM/<design>.dirtySignalNets

If there are no such nets, by default the above files are not created. When SEM_KEEP_EMPTY_REPORT is set, RedHawk creates these files, but with no nets reported in them. *Optional. Default: Off*

Syntax:

```
SEM_KEEP_EMPTY_REPORT [ 0 | 1 ]
```

SEM_MULTI_DRIVER_FILE

To provide non-zero start time for screening the input FSDB in multi-driver Signal EM analysis. The FSDB waveforms are applied from the beginning of the FSDB + the start time which listed in the last column of the filename specified. *Default : off*

Syntax:

```
SEM_MULTI_DRIVER_FILE <filename>
```

In <filename>,

```
<Signal net name> <Driver Instance name> <fsdb file> <fsdb probe name> <start_time>
```

Example:

```
net_1 drv_inst_1 filename.fsdb probename_1 1.1e-9
```

SEM_NET_CHECK_SHORT

When set, checks signal EM nets for driver-receiver shorts. Nets with shorts are dropped from analysis. *Optional. Default: 0.*

Syntax:

```
SEM_NET_CHECK_SHORT [ 0 | 1 ]
```

SEM_NET_FILTER_DRIVERS_FILE

When set, provides auto signal net filtering of signal nets whose driver is an instance of a master cell specified by the user. You can specify the cell names in the file to filter out the nets driven by those cells. The cells specified are small enough in terms of drive strength, and hence would not violate the EM limit of the driven nets. You must run RedHawk as follows:

```
redhawk -semnetfilter -f run_sigem.tcl
```

Default location of generated file containing the list of filtered nets is *adsRpt/SignalEM/<driver_cell_file>.net*. *Optional. Default: none.*

Syntax:

```
SEM_NET_FILTER_DRIVERS_FILE <filename>
```

SEM_NET_FILTER_MAX_PEAK

When set, provides auto signal net filtering, to filter out signal nets based on a specified peak net current threshold using this keyword:

You must run RedHawk twice in this flow, in the same directory. The process is:

First run: Run RedHawk as “redhawk -semnetfilter -f/b <cmd file>”, which reads the data in the setup design stage (ignoring the geometry to improve performance), then dumps a file including net names, net slew, net Total Cap, and net VDD.

Second run: RedHawk uses the simple criteria $I_{peak} = VDD * Cap / slew / 2$, and also the user-specified threshold value I_{max_peak} . If I_{peak} current for the net is greater than I_{max_peak} , then the net is analyzed with the Selective Net Import method.

Optional. Default: 0.

Syntax:

SEM_NET_FILTER_MAX_PEAK < I_{max_peak} >

Example

SEM_NET_FILTER_MAX_PEAK 1e-03

SEM_NET_FREQ

When set to 1, honors net frequency from the STA file instead of driver clock frequency. This is needed for PLL/frequency synthesizer nets in which the output net frequency is different from the input clock net frequency. *Default: 0*

Syntax:

SEM_NET_FREQ [0 | 1]

SEM_NET_INFO

Defines a file specifying toggle rates and/or transition times, uni-directional and bi-directional current scaling factors, and extra receiver capacitance (“extra-cap”) per signal net, and specify the slew (transition time) range for all nets. These values override the global values of TOGGLE_RATE and EM_SLEW_* keywords, as well as any toggle rate values set in power calculation or transition times in the STA file. The uni-directional scale entries must be entered in column 5, bi-directional scale entries in column 6 and the extra capacitance in column 7. You can input the slew range as a comment in the header, such as “# slew_range <value>”, where <value> is any floating point number between 0 and 1. Columns 3 ~ 12 values are optional, but all columns to the left of a specified value must have at least a “-” placeholder. Column 12 lists the voltage values to be applied to the input net names. A format example follows:

```

<net_name> <trans_time_sec> <toggle_rate> <freq> <uni-dir_scale>
<bi-dir_scale> <extra-cap> <driver_cell> <pin> <Ceff>
<-> <voltage>
# slew_range 0.8
net1 1e-13 2 2e09 0.7 - 1e-14 - - - -
net_154* 10e-12 0.5 2e9 7 - 1e-14 - - - 5 0.85
...

```

In the example above, ‘net1’ is assigned a transition time of 1e-13 seconds, within the slew range of 80%. It’s also assigned a toggle rate of 2 and frequency of 2 GHz. Its uni-directional currents are scaled by 0.7. No bi-directional scaling is used, but an extra receiver side load pin cap of 1e-14 F is applied. Wild cards for <net_name> are allowed. *DMP compatible. Optional. Default: none.*

Syntax:

SEM_NET_INFO <filename>

SEM_NET_REPORT

When set to 1, generates a consolidated signal net report *adsRpt/apache.sigem.netreport*, which includes various types of signal EM information.
 The format/content of the file is as follows:

```
net_name net_type voltage(V) toggle_rate frequency(Hz) driver_inst
driver_cell driver_pin wire_cap(F) pin_cap(F) external_cap(F) total_cap(F)
avg_current(A) rms_current(A) peak_current(A)
```

Optional. Default: 0 (no report).

Syntax:

SEM_NET_REPORT [0 | 1]

SEM_NEW_SIGEM_INFO_RPT

When set, reports power, ground, and clock domains associated with each instance, and the number of drivers per net, in the *apache.sigem.info* file. *Optional. Default: 0.*

Syntax:

SEM_NEW_SIGEM_INFO_RPT [0 | 1]

SEM_POWER_PIN_HANDLING

When set to 1, honors 'related_power_pin' attribute in the liberty file for signal nets in multi VDD cells in Signal EM analysis. When not specified, tool randomly picks up the first power pin present in adsPower/*power for that signal pin. *Default: 0.*

Syntax:

SEM_POWER_PIN_HANDLING [0 | 1]

SEM_RECOVERY_FACTOR

Allows calculation of signal EM avg current as a specified relationship between the charging current and discharging current, by specifying a discharge current recovery factor, R. By default, RedHawk calculates the rectified average current as

$$I_{avg} = C \cdot V \cdot F \cdot TR$$

where C is effective capacitance, V is the ideal voltage, F is the frequency, and TR is the toggle rate. When SEM_RECOVERY_FACTOR is specified, the average total current including recovery is calculated as follows:

$$I_{rec-avg} = C \cdot V \cdot F \cdot TR \cdot (1 - R) / 2$$

where $I_{charge} = I_{discharge}$ is assumed. The recovery factor R is the fraction of the charging current that is assigned to discharge, and can be set to values from 0.0 to 1.0. *DMP compatible. Default: none.*

Syntax:

SEM_RECOVERY_FACTOR <R>

SEM_SLEW_OPTIMIZATION

Controls the accuracy of slew optimization adjusted for load, to provide better current correlation with respect to Spice. When set in vectorless signal EM, this keyword considers the resistance shielding effect in slew optimization to improve waveform construction and obtain more accurate RMS and PEAK current results. Accuracy increases as the setting is raised from 1 to 3. *Optional. Default: 3.*

Syntax:

```
SEM_SLEW_OPTIMIZATION [ 1 | 2 | 3 ]
```

SEM_SPLIT_LONG_WIRE

When the keyword SEM_EXTRACT_LONG_WIRE is set, signal EM divides long wires into multiple RC-PI models and multiple nodes, based on the unit length specified, allowing more resolution in reporting EM values in long wires. *Optional.* *Default: 16 um.*

Syntax:

```
SEM_SPLIT_LONG_WIRE <PI_model_length>
```

SEM_TURBO_CLEAN_WIRE

Controls parallel signal EM wire merging and clean-up. When set, RedHawk releases memory after wire geometry cleanup. This affects only RedHawk DB cleanup performance and memory consumption, and has no impact on extraction and EM checking. *Optional. Default: 0 (off).*

Syntax:

```
SEM_TURBO_CLEAN_WIRE [ 0 | 1 ]
```

SEM_VECTORLESS_TIME_STEP

When set, allows you to set up time step controls for vectorless mode signal EM analyses. For mixed mode, you can also specify DYNAMIC_TIME_STEP to control time steps for both vectorless and vector-based nets. *Optional. Default 10ps.*

Syntax:

```
SEM_VECTORLESS_TIME_STEP <step_sec>
```

TEMPERATURE_EM

Specifies global operating temperature for calculating static and dynamic EM effects. *DMP compatible. Optional. Default: see section "Temperature Setting for Power EM Calculation", page 15-418.*

Syntax:

```
TEMPERATURE_EM <temp °C>
```

TEMPERATURE_REPORT_LINE_NUMBER

Wire temperatures are automatically reported after running emcheck, sorted by temperature. This keyword controls the maximum number of lines in the report file. *Optional. Default: 1000 lines.*

Syntax:

```
TEMPERATURE_REPORT_LINE_NUMBER <max_lines>
```

TEMPERATURES_EM

Specifies per layer operating temperatures for calculating static and dynamic EM effects. *Optional. Default: see section "Temperature Setting for Power EM Calculation", page 15-418.*

Syntax:

```
TEMPERATURES_EM {  
    <layer_1> <temp_1 °C>  
    ...
```

```
<layer_n> <temp_n °C>
}
```

THERMAL_COUPLING_PITCH

Performs RMS hot spot filtering in signal EM flow using TCL command '`perform emcheck -hotspot_filter`'. The default pitch multiplier is 15 for all metal layer.

Syntax:

```
THERMAL_COUPLING_PITCH {
    <ALL| metal layer> [spacing as a multiplier to metal pitch]
    <metal layer> [spacing as a multiplier to metal pitch]
    <metal layer> [spacing as a multiplier to metal pitch]
    ...
}
```

Example:

```
THERMAL_COUPLING_PITCH {
    ALL 20 #assign 20 as multiplier for metal layer
    METALL1 10 #metal 1 now gets 10 as pitch multiplier
    fpoly 0
}
```

USE_DRAWN_WIDTH_FOR_EM

By default RedHawk uses the drawn width to calculate current density for wires. When this option is turned off, silicon width is used for current density calculation. *DMP compatible. Optional. Default: the default setting for this keyword is foundry-dependent. Please consult Apache Support for appropriate usage.*

Syntax:

```
USE_DRAWN_WIDTH_FOR_EM [ 0 | 1 ]
```

	USE_DRAWN_WIDTH_FOR_EM_LOOKUP 1	USE_DRAWN_WIDTH_FOR_EM_LOOKUP 0
USE_DRAWN_WIDTH_FOR_EM 1	W (lookup) = W(D) W=W(D)	Not Supported
USE_DRAWN_WIDTH_FOR_EM 0	W (lookup) = W(D) W=W(S)	W (lookup)=W(S) W=W(S)

Some foundries require EM to be computed based on silicon width, but lookup based on drawn width for 20 nm and lower nodes. This flow is enabled by setting the two keywords as shown in the table above.

USE_DRAWN_WIDTH_FOR_EM_LOOKUP

By default, uses drawn wire width for the EM rule lookup table, rather than silicon width (when set to 0). *DMP compatible. Optional. Default: the default setting for this keyword is foundry-dependent. Please consult Apache Support for appropriate usage.*

Syntax:

```
USE_DRAWN_WIDTH_FOR_EM_LOOKUP [ 0 | 1 ]
```

VIA_COMPRESS

Can be used to control compression (grouping) of single cut vias in cases when run times are too long. When turned off, RedHawk leaves single cut vias as they are. When turned on, RedHawk groups single cut vias into arrays whose maximum size is the lesser of the value specified in the GSR keyword SPLIT_SPARSE_VIA_ARRAY (see page C-687) and a criteria allowing a small decrease in resistance accuracy, which limits array size. If the SSVA keyword is not set, the via grouping is based on the resistance accuracy criteria alone. Turning off via compression alters signal EM analysis violation results typically less than 0.1%, because of small changes to the underlying resistor modeling. *DMP compatible.*
Optional. Default: 1 (On).

Syntax:

```
VIA_COMPRESS [1 | 0]
```

Extraction and Netlisting Keywords

ACCURATE_METAL_DENSITY

Provides more accurate metal density calculation for designs that contain overlapping or redundant metal geometries. This keyword was off by default prior to v13.2, and can be turned off to save disk space. *DMP compatible. Optional.*
Default: 1 (On).

Syntax:

```
ACCURATE_METAL_DENSITY [1 | 0]
```

ALLOW_SEPARATED_METAL_PSEUDO_VIAS

Specifies when pseudo vias are inserted, based on what type of tech file is being used. By default (AUTO), when a particular foundry tech file for a 20nm or smaller node is detected, pseudo-vias are only formed between layers that either touch vertically or have an RCSD resistance table defined. This blocks pseudo vias from being extracted except between base layers for particular foundry tech files (contact your Apache AE for details on use). The default setting for all other tech files connects all non-touching vertically coincident layers with pseudo vias. *Optional.*
Default: AUTO.

Syntax:

```
ALLOW_SEPARATED_METAL_PSEUDO_VIAS [ AUTO | 0 | 1 ]
```

where

AUTO (default) - as described above, depending on what foundry tech file is used.

0 - no via connections between non-touching vertically-coincident layers

1 - inserts pseudo vias between all non-touching vertically-coincident layers

AUTO_INTERNAL_NET_EXTRACT

When set, automatically extracts internal P/G nets connected to external signal nets for switch cells. If it is off, RedHawk does not automatically extract internal nets.
Default 1, On.

Syntax:

```
AUTO_INTERNAL_NET_EXTRACT [ 0 | 1 ]
```

CELL_CURRENT_DIST_FILE

Specifies a file that contains a list of cells that receive equal current distribution on all nodes on each pin, while all other cell pin current is assigned to only one of the nodes connected to each pin instance. If there are no cells specified in this file, pin current distribution is controlled by the keyword INST_CURRENT_DIST_MODE.
DMP compatible. Optional. Default: none.

Syntax:

```
CELL_CURRENT_DIST_FILE <filename>
```

CELL_PIN_FILE

Specifies a file that defines cell pins, including cell names (with wildcard support), pin type, and layer. The 'LOWEST' sub-option to the 'LAYER' keyword enables setting the current sink on the lowest layer of any pin. The format of the specified cell pin file is :

```
CELL [ STD_CELL | <cell name> | <cell name pattern>* ]  
PIN [ <Vdd_pin_name> | <Gnd_pin_name> | VDD_TYPE | GND_TYPE ]  
LAYER [ <metal layer name> | LOWEST ]  
END PIN  
END CELL
```

DMP compatible. Default: Off (0)

Syntax:

```
CELL_PIN_FILE <file_name>
```

CEXTRACTION_USE_SPEF

When set, supports back-annotation of SPEF for Signal EM analysis such that SPEF capacitances are used in place of the RedHawk extracted capacitances. The SPEF file must be annotated with layer information on nodes, and have a layer map header. The QRC format SPEF file is supported for capacitance back annotation flow. QRC SPEF differs from the Synopsys SPEF in three ways:

1. They have a *PORTS section in the header
2. They have the *NAME_MAP section after the *LAYER_MAP section, rather than first the *NAME_MAP and then the *LAYER_MAP
3. They have \$LEVEL's instead of \$lvl's for layer number annotation

The SPEF layer map may be overridden using a layer map file specified using the "CEXTRACTION_SPEF_LAYER_MAP" keyword in cases in which the SPEF layer names do not match the RedHawk tech file layer names. *Optional. Default: Off (0)*

Syntax:

```
CEXTRACTION_USE_SPEF [ 0 | 1 ]
```

CEXTRACTION_SPEF_LAYER_MAP

When set, the SPEF layer map is overridden using the layer map file specified, in cases in which the SPEF layer names do not match the RedHawk tech file layer names. *Optional. Default: none.*

Syntax:

```
CEXTRACTION_SPEF_LAYER_MAP <filename>
```

CLEAN_VIAS_AFTER_WIRES

When set, interchanges the via compression step with the wire merge step, so that compression occurs after merging, and also allows wires that come from pushed LEF pins, to improve via compression. This deals with the special cases in which metal1 is only in LEF, and via1 and metal2 are only in DEF, and generally allows more freedom for via compression. *Default: 0 (1 by default in v14.2)*.

Syntax:

```
CLEAN_VIAS_AFTER_WIRES [ 0 | 1 ]
```

CMM_RIVETED_CONN

This keyword improves the accuracy of CMM voltage drops when there are wire geometries at the top level overlapping CMM pin geometries, such as metal4 CMM pins connecting to overlaying metal4 wire geometries. In such cases RedHawk could identify duplicate metal4 shapes, and create parallel resistive paths that reduce the accuracy of the connections. Using this keyword removes the duplicate paths and improves accuracy. *DMP compatible. Optional. Default: On (1)*.

Syntax:

```
CMM_RIVETED_CONN [ 0 | 1 ]
```

CONNECT_SWITCH_PINS

For switch cells, allows shorting of different combinations of internal and external pins. *DMP compatible. Optional. Default: 1*.

Syntax:

```
CONNECT_SWITCH_PINS [ 0 | 1 | 2 | 3 ]
```

where

0 : shorts only INT switch pin instance shapes

1 : shorts both INT and EXT switch pin instance shapes (default)

2 : shorts neither INT nor EXT switch pin instance shapes. Note that for designs with switches described as MACRO cells, this setting allows current propagation in the ring, as internal and external pins are not merged.

3 : shorts only EXT switch pin instance shapes

COUPLEC

If set to 1, extracts coupling capacitance between different nets in power and ground domains, and also floating metal geometries, which can contribute metal capacitance on one side to Vdd and on the other side to Vss. *Optional. Default: 0 (off)*.

Syntax:

```
COUPLEC [ 0 | 1 ]
```

DEF_IGNORE_SNEN_SHIELD

When set, wires and vias of nets defined in the “SPECIALNETS” section of DEF under “+ SHIELD” are ignored. *Optional. DMP Compatible. Default: 1 (SHIELD SPECIALNETS ignored)*.

Syntax:

```
DEF_IGNORE_SNEN_SHIELD [ 0 | 1 ]
```

DO_PININST_INTERNAL_CONN

When set, models P/G pins on all instances as internally connected to each other per domain, even though the LEF definition does not show that. *Optional. Default: 0 (off).*

Syntax:

```
DO_PININST_INTERNAL_CONN [ 0 | 1 ]
```

ENABLE_AUTO_SCANLINE

When set to 1, tool intelligently picks up layers in SCANLINE_MERGE_LAYERS keyword for advanced technology nodes. *Default: 0.*

Syntax:

```
ENABLE_AUTO_SCANLINE [ 0 | 1 ]
```

EXPAND_CELL_PIN_FILE

Supports modifying connections of switch internal/external pins to the rest of the power grid, for cases in which real physical connectivity is missing in DEF. This feature is particularly useful when switch cells are used in early stage analysis.

The syntax for specifying cell/ pin switch modifications in an “expand file” is as follows (distance units are microns):

```
VERSION <1.0>
CELL <cellname>
PIN <pin_name>
COPY_PIN_LAYER FROM <layer_name> TO <layer_name1> ...
LAYER <layer_name>
EXPAND <dist_left> <dist_right> <dist_up> <dist_down> FROM PIN
...
END PIN
EXPAND <dist_left> <dist_right> <dist_up> <dist_down> FROM CELL
...
END CELL
```

An example cell/pin “expand file” for pins would look as follows:

```
VERSION 1.0
CELL SWITCH_CELL_1A
PIN VDD_EXT
COPY_PIN_LAYER FROM metal4 TO metal5 metal6 metal7
LAYER metal7
EXPAND 20 40 20 40 FROM PIN
END PIN
PIN VDD_INT
COPY_PIN_LAYER FROM metal4 TO metal5 metal6 metal7
LAYER metal7
EXPAND 20 40 20 40 FROM PIN
END PIN
PIN VSS
COPY_PIN_LAYER FROM metal4 TO metal5 metal6 metal7
LAYER metal7
EXPAND 20 40 20 40 FROM PIN
END PIN
END CELL
```

The COPY_PIN_LAYER option can be used before the PIN definition also.

Syntax:

```
EXPAND_CELL_PIN_FILE <expand_filename>
```

EXTRACTION_INC

If ECO changes have been made to decaps, vias, or pins, and extraction must be performed again, with this keyword set to 1, an incremental extraction is performed only in the areas of ECO changes. Full design extraction is always performed for changes to wires. *Optional. Default: 0 (off).*

Syntax:

```
EXTRACTION_INC [ 0 | 1 ]
```

EXTRACT_PIN_VOL_INSTS

Allows boundary pins for blocks to be extracted for use in static and dynamic block-only runs. (Blocks are hierarchical instances that contain child instances.) To extract a block's pins in a full-chip top-down analysis, EXTRACT_PIN_VOL_INSTS specifies the post-flattened block names, and captures the “minimum” worst-case voltage seen at the block boundary pins during the top level dynamic simulation. This allows you to perform block level analysis with worst-case dynamic boundary voltages.

With the block definitions RedHawk extracts the pins for each specified block during ‘setup design’, based on P/G net connections between the block and top-level routes. The boundary pins and their voltages are available after full chip simulation. These block pin voltages are then available to run simulation on individual blocks. *Optional. Default: None.*

Syntax:

```
EXTRACT_PIN_VOL_INSTS {  
<hierarchical_path>/<block_inst1>  
...  
}
```

Example:

```
EXTRACT_PIN_VOL_INSTS {  
abcd3560/ifd_top_inst  
abcd3560/vdec_pri_inst  
abcd3560/ter_eq1  
abcd3560/test_mips_inst  
}
```

This example creates a probe for each power/ground pin specified in DEF files describing these hierarchical blocks. To see the locations of these probes (block pins) use the menu command **View -> Connectivity -> Show Block Pins**. The names of the block pins that are created are reported, along with their measured voltages, in the file *adsRpt/fullchip_PV.rpt*.

For each block that is specified in the GSR file, a voltage source file (PLOC) is created. To perform a block level analysis, in a block-level GSR file, specify the corresponding *ploc* file generated from the top-level dynamic run. Then run block level static or dynamic analysis.

EZ_MERGE_NON_RECT_WIRE

Used to turn on 45-degree wire merging to correctly handle 45-degree wire segments with overlapping geometries. *Default : 0 (off).*

Syntax:

```
EZ_MERGE_NON_RECT_WIRE [ 0 | 1 ]
```

EZ_MERGE_NON_RECT_WIRE_MAX_LENGTH

Used to control the maximum wire length in the 45-degree wire merging process.
Default: maximum length 100 um.

Syntax:

```
EZ_MERGE_NON_RECT_WIRE_MAX_LENGTH <max_length_um>
```

FIND_ABUTTED_NONSTD_INSTS_PININSTS

When set, checks if port geometries of pinInsts of non-standard cell instances are physically overlapped with those already connected to nets and should be logically connected. These geometries are then connected to their associated nets. *Default: 0 (off).*

Syntax:

```
FIND_ABUTTED_NONSTD_INSTS_PININSTS [ 0 | 1 ]
```

IGNORE_DUMMY_PNET

When set, turns on checking of dummy nets specified in GSR keywords VDD_NETS and GND_NETS. *DMP compatible. Optional. Default: 0.*

Syntax:

```
IGNORE_DUMMY_PNET [ 0 | 1 ]
```

IGNORE_MACROEEQ

When set it to 1, when parsing the LEF file, MACRO EEQCELLS are ignored to avoid incorrect instance reporting in *adsRpt/<design>.power.rpt*. *DMP compatible. Optional. Default: 0.*

Syntax:

```
IGNORE_MACROEEQ [ 0 | 1 ]
```

IGNORE_OPC_METAL

When set, ignores all isolated wires with a length or width smaller than the specified dimension (microns.) *Default : 0 (off - do not ignore any wires).*

Syntax:

```
IGNORE_OPC_METAL <threshold_dimension>
```

IGNORE_PLOC_ON_OBS

When set to 1, ignores pads (from ploc file) which touch the obs area of instances.
Default:0.

Syntax:

```
IGNORE_PLOC_ON_OBS [ 0 | 1 ]
```

IGNORE_THICKNESS_VARIATION

When set, ignores metal thickness variation due to metal density during resistance extraction. The net thickness, and therefore its resistance, is impacted by the local metal density (if the technology file includes a thickness variation description). If the metal density value is not reliable because of data input issues or early analysis, you can ignore its impact by setting this keyword to 1. *Optional. Default: 0.*

Syntax:

```
IGNORE_THICKNESS_VARIATION [ 0 | 1 ]
```

INST_CURRENT_DIST_MODE

When set to 1, equally distributes instance current to all nodes located in the pin instance region. When set to 0, and the GSR keyword CELL_CURRENT_DIST_FILE has no cells specified, assigns current to only one of the nodes connected to each pin instance in the design (no current distribution regardless of design size). For cells defined in CELL_CURRENT_DIST_FILE, current is distributed equally to all nodes located in the pin instance region for those cells. By default (-1), current distribution is turned on only if the number of instances in the design is less than 100k. *DMP compatible. Optional. Default: -1.*

Syntax:

```
INST_CURRENT_DIST_MODE [ -1 | 0 | 1 ]
```

INTERNAL_CONNECT_PIN_CELLS_FILES

RedHawk is accurate in modeling LEF definitions of a cell because power/ground pins defined in the LEF are pushed to the upper level of hierarchy as wires, and treated the same as other geometries in the design; there is no assumption of internal connection. However, in some cases, such as when pins need to be modeled as internally connected to each other, even though the LEF definition does not show that, the modeling can be changed for specified cells by using this keyword, INTERNAL_CONNECT_PIN_CELLS_FILES. For this case, the specified <cell_list_file> must list all cells to be modeled. *DMP compatible. Optional; default: None.*

Syntax:

```
INTERNAL_CONNECT_PIN_CELLS_FILES <cell_list_file>
```

Example:

```
INTERNAL_CONNECT_PIN_CELLS_FILES Int_conn_cells
```

IR_REPORT_STACKVIA_METAL

When set, reports IR drop values of metal geometries within via stacks. Note that using this keyword increases run time, especially when there are many stack vias with intermediate metals. *Optional; default: 0.*

Syntax:

```
IR_REPORT_STACKVIA_METAL [ 1 | 0 ]
```

LONG_WIRE_RES_CALC

When set, wire resistance calculation uses the density for each resistive section of the wire, rather than the average density of the whole wire, which produces a more accurate resistance calculation. The density variation is only relevant in cases in which metal density information in THICKNESS_VS_DENSITY and POLYNOMIAL_BASED_THICKNESS_VARIATION tech file keyword tables is used to look up factors that determine the final wire thickness. *Optional. Default: Off (0).*

Syntax:

```
LONG_WIRE_RES_CALC [ 0 | 1 ]
```

LOWEST_METAL

Specifies the lowest metal layer name (defined in the *.tech* file) for capacitance extraction. *DMP compatible. Optional. Default: None.*

Syntax:

```
LOWEST_METAL <metal_layer_name>
```

Example:

```
LOWEST_METAL Metal1
```

MACRO_IRDROP

When set, enables Dynamic IR drop reporting at Pin layer for Macro-Cells in Dvd report file *adsRpt/Dynamic/<Design_Name>.dvd**. The <instance_name> is the DvD result of the macro instance reported in this file. *DMP compatible. Default: 0.*

Syntax:

```
MACRO_IRDROP [ 0 | 1 ]
```

MEM_RC_MODEL

When set, speeds up extraction process for memory cells and blocks. *Optional. Default: 0.*

Syntax:

```
MEM_RC_MODEL [ 0 | 1 ]
```

MERGE_ABUTTED_ASYM_CUTS

When set, merges abutted asymmetric via cuts, and thereby avoids false EM violations. *Optional. Default: Off.*

Syntax:

```
MERGE_ABUTTED_ASYM_CUT [ 0 | 1 ]
```

MERGE_WIRE

MERGE_WIRE is On by default (unless there are 45-degree wires in the design, when wire merging is disabled), and tries to merge all wires that overlap. RedHawk automatically drops overlapping nets in analysis if there are any wires that **MERGE_WIRE** cannot merge. These nets are reported in the file *adsRpt/SignalEM/topcell.droppedSignalNets*. Note that you can force wire merging setting '**MERGE_WIRE 1**' for designs with 45-degree wires. *DMP compatible. Optional. Default: 1 (On).*

Syntax:

```
MERGE_WIRE [ 0 | 1 ]
```

MESH_VIAS_FILE

Specifying the filename *ecoRouting.tcl* with this keyword enables RedHawk to do ECO routing at the beginning of the 'setup design' step. Normal ECO routing, as specified in the command file, is after setup design, in which case some cell pin instance geometries may not be pushed out when they depend on ECO routing geometries. *Optional. Default: none.*

Syntax:

```
MESH_VIAS_FILE ecoRouting.tcl
```

METAL_DENSITY_BOUNDS

When set, restricts the computed metal density value between specified minimum and maximum values per metal layer. The metal “density” is the fraction of the layer covered by metal, a value between 0.0 and 1.0. This keyword is useful for early analysis, or when the data is not mature enough for accurate density computation. The min and max density values can usually be obtained from the foundry. Metal density impacts metal thickness consideration during resistance extraction, depending on the technology file description. *Optional. Default: none.*

Syntax:

```
METAL_DENSITY_BOUNDS {  
    <layer> <min_density> <max_density>  
}
```

Example:

```
METAL_DENSITY_BOUNDS {  
    M4 0.1 0.3  
}
```

In this example the computed metal area density for M4 layer metal is restricted between 0.1 and 0.3.

MINWIDTH_FROM_LEF

Specifies that the minimum wire width should be taken from the LEF WIDTH parameter. *Optional. Default: 1 (On).*

Syntax:

```
MINWIDTH_FROM_LEF [ 0 | 1 ]
```

MIN_WIRE_DIMENSION

When MERGE_WIRE is set to 1, this keyword allows changing the resolution threshold for the snapping process, with a one-side resolution of 0.006um, or 1/10 of the minimum width, whichever is smaller. Modifying the resolution reduces the number of wires resulting from the merge/cut process (merging merges all connected metal polygons into a single polygon to remove overlaps, while cutting cuts the polygon formed in the merging step into many no-overlap rectangular geometries). *DMP compatible. Optional. Default: 12.*

Syntax:

```
MIN_WIRE_DIMENSION <resolution_in_internal_db_unit>
```

Example:

```
MIN_WIRE_DIMENSION 12
```

means a resolution of 0.006 um, since the internal DB unit is 2000 and 12/2000 =0.006.

MPR_MODE

Provides control of power grid extraction and modeling using RedHawk's advanced Mesh Pattern Recognition technology. MPR uses both circuit element pattern recognition and network reduction to increase analysis performance by reducing memory requirements and run time for designs with dense P/G grids. There are three active modes, with different amounts of network reduction:

0 - no MPR (default)

- 1 - uses pattern recognition, but no network reduction, which may improve RedHawk memory requirements and run time, but not simulation. This setting is recommended for IR/EM sign-off.
- 2 - uses pattern recognition and moderate network reduction, which may improve RedHawk memory requirements, but not as much in simulation as mode 3. However, this mode has better accuracy than mode 3.
- 3 - uses pattern recognition and aggressive network reduction. For designs with dense P/G grids, memory requirements may be decreased by 40% and run time by 50% for dynamic simulation where there are capacity bottlenecks. This setting is recommended for DvD/CPM analysis for large designs with capacity bottlenecks.

You can change the MPR mode between static and dynamic runs using the command:

```
gsr set MPR_MODE [ 0 | 1 | 2 | 3 ]
```

Also, MPR supports Shortest Path Tracing (minimum resistance path tracing). *DMP compatible. Optional. Default : 0 (off).*

Syntax:

```
MPR_MODE [ 0 | 1 | 2 | 3 ]
```

MPR_PARTITION_REDUCTION

When turned on, reduces the design R/N ratio (number of resistors divided by the number of nodes) in MPR3 runs for large patterns. *Optional. Default : off.*

Syntax:

```
MPR_PARTITION_REDUCTION [ 0 | 1 ]
```

MPR_POWER_LIMIT_FOR_RED

When set, improves the accuracy of MPR reduction mode by triggering non-uniform network reduction if tile power is less than the specified value. That is, any patterned network covered by the tile area is not reduced if the tile power is less than the specified value. Aggressive non-uniform network reduction is performed if the power is less than the specified value. Improved network reduction is achieved for cases having higher current densities in certain regions. The recommended value is 0.001W or above. *Optional. Default: None.*

Syntax

```
MPR_POWER_LIMIT_FOR_RED <power_W>
```

NODE_REDUCTION_MODE

When set for low power analysis, improves ALP accuracy, and node reduction is not performed.

Note that to avoid long computation times caused by no node reduction, using the 'perform analysis -lowpower -alp3d' option is recommended to speed up mixed-mode ramp-up noise coupling voltage analysis (the time step should be set to less than 150ps). *Optional. Default: ALP_ACCURATE (no node reduction).*

Syntax

```
NODE_REDUCTION_MODE [ 0 | ALP_ACCURATE ]
```

PIN_SLICE_CELL_LIST_FILE

Specifies a file containing a unique list of cells for which the keyword PIN_SLICE_LIMIT should apply. *Optional. Default: none.*

Syntax:

```
PIN_SLICE_CELL_LIST_FILE <cell_list_file>
```

PIN_SLICE_LIMIT

Globally splits large single transistor pins into many pins based on the specified distance factor (pitch), unless the keyword PIN_SLICE_CELL_LIST_FILE is set, when it only applies to cells listed in this file. This approach enables the distribution of currents on a large transistor pins over the entire area of the pin. The specified pitch is the distance between the new split pins.

When set, if a pin is created that is longer than the limit, additional nodes are created, starting at the left/bottom, at intervals specified in microns. Note that existing nodes for wire or via connections are not considered. *Optional. Default: none.*

Syntax:

```
PIN_SLICE_LIMIT <spacing_um>
```

Example:

```
PIN_SLICE_LIMIT 5
```

This setting creates pin nodes every 5 microns.

PPI_STD_IGNORE_TOUCH_VIA_MET

When turned on, touching pin geometries with vias are not considered and dropped from analysis (not pushed out). *DMP compatible. Optional. Default: Off.*

Syntax:

```
PPI_STD_IGNORE_TOUCH_VIA_MET [ 0 | 1 ]
```

PPI_CELL_EDGE_MAX_NM_THRESHOLD

For a pin instance geometry that meets the threshold percent criteria to be pushed out (see PPI_CELL_EDGE_THRESHOLD_PERCENT keyword following), its absolute length must also be not less than the cell box length minus the specified value (default is 50 nm).

Syntax:

```
PPI_CELL_EDGE_MAX_NM_THRESHOLD <max_length>
```

Example

(See example for next keyword)

PPI_CELL_EDGE_THRESHOLD_PERCENT

When a pin instance geometry has a length (in the cell rail direction) that is not less than the cell bounding box length by more than the specified percentage, it is pushed out as a routing wire. Note that the absolute length criteria (see previous PPI_CELL_EDGE_MAX_NM_THRESHOLD keyword) also must be satisfied.

Optional. Default: 1%.

Syntax:

```
PPI_CELL_EDGE_THRESHOLD_PERCENT <percent>
```

Example

Assume a cell with two pin instance geometries in the cell rail direction of:

- a. 800 nm
- b. 900 nm.

```
PPI_CELL_EDGE_THRESHOLD_PERCENT 20
```

```
PPI_CELL_EDGE_MAX_NM_THRESHOLD 150
So 20% of 1000 nm = 200 nm => 1000 nm – 200 nm = 800nm
So both pin instance geometries are > 800nm and are satisfying the
PPI_CELL_EDGE_THRESHOLD_PERCENT criteria.
In addition, the PPI_CELL_EDGE_MAX_NM_THRESHOLD needs to be met.
Cell length 1000nm – 150 nm = 850 nm
So only pin instance geometries b.) 900 that are > 850 nm will be pushed out.
```

PRIMARY_OUTPUT_LOAD_RC_MODEL

Specifies an RC model as load at the primary output pin by net, which is used in simulation to calculate the current for the primary output net. So the simulation result is more accurate for those primary output nets connected with distributed caps, and the shielding effect of the resistance on the load cap can be considered.
Optional. Default: none.

Syntax:

```
PRIMARY_OUTPUT_LOAD_RC_MODEL {
    <net1> <r1> <c1>
    <net2> <r2> <c2>
    ...
}
```

PUSH_PININST_CELLS_FILES

Specifies a file that lists all cells whose pin instance geometries should be pushed out unconditionally during extraction. Pushes out only the pins specified for each cell in the file when GSR keyword “PUSH_SIGNAL_PININST” is On. In some types of designs most cells in the design would have to be added to the <ppi_cell_file> to catch all desired intra-cell connections. The file also can be used to specify shorted pin nodes within a single layer and within certain bounding box regions inside a cell. Wild card characters in the cell names and pin names are allowed. The types of syntax allowed in the PPI file are given below:

Original format:

```
<cellName> [ All |<pinName1> { <layer1> <layer2> ... } <pinName2> { ... } ... ]
```

Newer format:

```
<cellName>[All|PIN {<pinName1> ...} LAYER {<layerName1> ...} {BBOX <bbox1>}]
```

```
...
```

```
<cellName>[All|PIN {<pinName1> ...} LAYER {<layerName1> ...} {BBOX <bboxN>}]
```

For both types of syntax above, wildcards can be used to specify both <cellName> and <pinName>, using the single-character wildcard “?”, and the multiple-character wildcard “*”. The <bbox> is specified as the x,y coordinates of the lower left and upper right corners of the region. BBoxes and layers are the same for all wildcard members. Multiple lines with the same cellname and different boundary boxes can be specified. The layername and bbox parameters are optional. A sample push pininst cells file is shown below:

```
AB2 PIN { VDDS } LAYER { MET1 } BBOX {0 0 5 5}
AB2 PIN { VDDS } LAYER { MET1 } BBOX {10 10 20 20}
```

In the above example, all the nodes in the region {0 0 5 5} for cell AB2 on layer MET1 for pin VDDS are shorted together. Similarly, for the region {10 10 20 20} for the cell AB2 on layer MET5 for pin VDDS, all nodes are shorted together. However, these two regions are not shorted. NOTE: Only pins of the same layer and of the same net can be shorted together. *DMP compatible. Optional. Default: None.*

Syntax:

```
PUSH_PININST_CELLS_FILES <ppi_cell_file>
```

PUSH_PG_PININST

When set, extracts and processes P/G pin instances in standard cells, even if they are not on a boundary, since in some low-power designs they are important and need to be extracted. When PUSH_PG_PININST is turned On :

- If cell is not a standard cell, all pins are extracted
- If cell is a standard cell (it has rail pins), and
 - it has P/G pins on the boundary (rail pins), pins are extracted
 - it has P/G pins not on the boundary, pins are extracted
 - for non-P/G pins, pins not extracted

The default behavior is that standard cell pins *not* on a boundary are *not* extracted.
DMP compatible. Optional. Default: 0, off.

Syntax:

```
PUSH_PG_PININST [ 0 | 1 ]
```

PUSH_SIGNAL_PININST

When set to 1, signal pins are pushed out, making pin connections when signal wires do not overlap LEF pins completely or signal pins are abutted to DEF route (not overlapped) . Otherwise these wires would not be connected. *DMP compatible. Optional. Default: 0, off.*

Syntax:

```
PUSH_SIGNAL_PININST[ 0 | 1 ]
```

QUICK_MESH_WIRE_MERGE

When turned on, merges dense mesh wire structures to improve wire management and reduce setup design time. *Optional. Default: 0, off.*

Syntax:

```
QUICK_MESH_WIRE_MERGE [ 0 | 1 ]
```

REPORT_ALL_UNCONNECT_PORTS

Specifies how to report pins containing one or more unconnected groups of geometries in the file *adsRpt/<design name>.PinInst.unconnect.md*. (a “group” is a set of geometries connected together, although not necessarily to the pin), which has the following content syntax:

```
PinInst<multi_port_macro_1:gnd> (total groups 5)  
1/5 at ( 32.000000 27.000000 ) M2
```

This shows that one group of five associated with the pin is unconnected, and indicates one x,y location identifying the unconnected group. *DMP Compatible. Optional. Default: Off.*

Syntax:

```
REPORT_ALL_UNCONNECT_PORTS [ off | all | macro_only |  
macro_detail | macro_std_all ]
```

where

off : if all pin geometries are unconnected, the instance is reported as unconnected (default). Used for standard cells.

all: if any pin geometries are unconnected, the instance is reported as unconnected. Used for other than standard cells.

macro_only: standard cell macros are reported as in the 'off' option; non standard cell macros are reported as in the 'all' option.

macro_detail: all macro pin groups not connected are reported, even if some groups are connected.

macro_std_all : provides detailed reporting for all standard cell disconnects and macro pinInst disconnects in the file *adsRpt/<design name>.PinInst.unconnect.md*.

SAVE_CONSOLIDATED_R

To perform more accurate min res path tracing at wide wire junctions where the R-network is made of multiple fanout connections with higher resistances. By setting "SAVE_CONSOLIDATED_R 1" in GSR, the junction resistances with modeling resistor count greater than or equal to a default lower limit for any junction are saved. To use this feature, the switch '-consolidatedR' needs to be used in 'perform min_res_path' TCL command.

Syntax:

```
SAVE_CONSOLIDATED_R [ 0 | 1 ]
```

SCANLINE_MERGE_LAYERS

Improves accuracy of 45-degree wire extraction when cleaning wire geometries on all layers or on the layers specified, including substrate layers. The geometry merging algorithm merges same net wires with different mask colors and RedHawk will choose the mask color with lower index after wire merge. *DMP compatible*.

Optional. Default: none.

Syntax:

```
SCANLINE_MERGE_LAYERS {  
    [ ALL | <layer1> <layer2> ... ]  
}
```

SINGLE_CUT_SLICE_LAYERS

Specifies layers on which to slice multiple-cut via geometries into separate single cuts for cases in which compression of multi-cut vias is needed. (RedHawk does not compress multi-cut via groups, so using this keyword changes the input vias to single cuts, allowing compression.) *Optional. Default: none.*

Syntax:

```
SINGLE_CUT_SLICE_LAYERS {  
    <layer1>  
    <layer2>  
    ...  
}
```

SIZE_BASED_PININST_CURRENT_DISTRIBUTION

When set, RedHawk analysis distributes current according to geometry size when there are many nodes in the same section of the pin. *DMP compatible. Optional. Default: 1.*

Syntax:

```
SIZE_BASED_PININST_CURRENT_DISTRIBUTION [ 0 | 1 ]
```

SKIP_CONNECTIVITY_CHECK

When set, skips the connectivity checking step when performing extraction, to speed up extraction without any significant loss of accuracy. *Optional. Default: 0.*

Syntax:

```
SKIP_CONNECTIVITY_CHECK [ 0 | 1 ]
```

SKIP_LAYERS_FILE

Specifies a file containing a list of layer names, one per line, as follows:

```
# comment
<layer1>
<layer2>
...
...
```

The specified layers have minimum (effectively zero) resistance extracted for them, whether they are metal, via or pseudo-via layers. The typical use of this keyword is to run an extracted model that gives gridcheck results that only includes resistance for particular layers, generally with the layers below metal1 removed. This is useful for examining the effect of metal1 and above wiring separately. *Optional. Default: none.*

Syntax:

```
SKIP_LAYERS_FILE <filename>
```

SPECIAL_SML

Enables the special handling of complex structures in scanline merging and reduces node count. *Default: none.*

Syntax:

```
SPECIAL_SML {
    Layer1
    Layer2
    ...
}
```

SPLIT_SPARSE_VIA_ARRAY

Improves via compression (grouping) in defined via arrays by setting a maximum length for arrays, without losing accuracy in resistance extraction. If this keyword is specified, via arrays longer than the specified maximum length L are split into individual arrays of length L or less for circuit modeling purposes. This can be used to achieve more accurate voltage drop numbers. For asymmetric via arrays (via cuts that have different sizes), extraction splits asymmetric via arrays into individual vias to improve accuracy. Note that SPLIT_SPARSE_VIA_ARRAY is overridden by SPLIT_VIA_ARRAY if both are set. *DMP compatible. Optional. Default: none.*

Syntax:

```
SPLIT_SPARSE_VIA_ARRAY <max_length_microns>
```

SPLIT_VIA_ARRAY

Allows per layer specifications of the via split threshold (in um). Note that keyword SPLIT_SPARSE_VIA_ARRAY is overridden by SPLIT_VIA_ARRAY if both of them are set. *DMP compatible. Optional. Default: none.*

Syntax:

```
SPLIT_VIA_ARRAY {
```

```

<layer1> <split_threshold1>
<layer2> <split_threshold2>
...
}
```

SPLIT_VIA_ARRAY_CELL

Specifies cell-based Split Sparse Via Array (SSVA) formation, allowing split vias only inside specified cells, at the given threshold dimension or less. Note that **SPLIT_VIA_ARRAY_CELL** has a higher priority than the keyword **SPLIT_SPARSE_VIA_ARRAY**, and the latter is overridden if both keywords are set.
DMP compatible. Optional. Default: none.

Syntax:

```

SPLIT_VIA_ARRAY_CELL {
    <cellname1> <split_threshold1_u>
    <cellname2> <split_threshold2_u>
    ...
}
```

STATIC_IR_HIDE_DISCON_INST

When set, hides disconnected instances in static IR instance maps. *DMP compatible. Optional. Default: 0.*

Syntax:

```
STATIC_IR_HIDE_DISCON_INST [ 0 | 1 ]
```

STATIC_REDUCTION

Performs network optimization for the static flow before sending the R-network to simulation. Turn optimization off by setting this keyword to 0. *Optional. Default: 1 (On).*

Syntax:

```
STATIC_REDUCTION [ 0 | 1 ]
```

TSV_MODEL_FILE

Supports TSV (Thru Silicon Via) model sub-circuits in the 3D IC flow, which allows a Sentinel TSV model file to be imported for each die.

Note that in order to create the TSV model file in Sentinel, the TSV via information first must be generated for each die using the TCL command “dump tsv_via”, which dumps the Sentinel TSV information into the file *adsRpt/<dieName>/<dieName>.tsv_via*. Then **RedHawk** must be rerun with this keyword in the GSR file. *Optional. Default: none.*

Syntax:

```
TSV_MODEL_FILE <Sentinel_model_file>
```

USE_INTERPETCH_FOR_R

When set to 1, correctly interpolates.tech file curve flatness checks of the “ETCH_VS_WIDTH_AND_SPACING” table. *Optional. Default: 1.*

Syntax:

```
USE_INTERPETCH_FOR_R [ 0 | 1 ]
```

USE_MVM_PIN_MODEL

For designs with standard cells having both MET1 and MET2 parallel rails with vias between, faster runtimes and better memory use can be achieved by turning on this keyword, which also improves performance for standard cells having pin geometries with “fish-bone” structures. More than 50% reduction in node count can be achieved in most of these cases. *DMP compatible. Optional. Default: none.*

Syntax:

```
USE_MVM_PIN_MODEL [ 0 | 1 ]
```

VIA_MAX_SPACE_FOR_COMP

Using this keyword can reduce the number of vias and nodes in simulation. In extraction vias are compressed based on a calculated maximum spacing value in microns between vias involving the cut width and the values of the MINWIDTH and MINSPACE ‘metal’ options in the Tech file. So there is compression if the space between via cuts is smaller than the calculated max spacing value. If this keyword is set, it overrides the calculated max spacing value. If via compression causes net shorting, it can be eliminated by setting this keyword to a value smaller than the spacing of the closest via cuts. This may compress fewer via cuts, so more vias may appear, and the number of nodes could increase, but the risk of shorting nets then is much lower. *DMP compatible. Optional. Default: none.*

Syntax:

```
VIA_MAX_SPACE_FOR_COMP <max_space_um>
```

VIAMODEL_NAME_CHECK

When set, reduces the number of viaModels created by RedHawk during import def (Slice viaArray). Allows creating just one viaModel when several viaModels are actually the same (same topLayer, topGeo, botLayer, botGeo, cutLayer, cutRect, orientIndex, useLandRule), but their viaModel names are different. *DMP compatible. Optional. Default: 1.*

```
VIAMODEL_NAME_CHECK [ 0 | 1 ]
```

WIRE_R_COMPENSATION

By default extraction always includes the resistance of wire segments directly below and above via cuts (usually referred to as metal overlap or metal overhang). Set this keyword if you want to exclude all resistance of this type from R extraction, and use only the vertical via resistance.

Note: Compensation for pseudo-vias is not turned on by this keyword, and is only on if the appropriate tech file VIA section has WIRE_R_COMPENSATION set. Also, if this keyword is set only in the tech file, it must be specified in every VIA section in the tech file. *Optional. Default: 0 (include).*

Syntax:

```
WIRE_R_COMPENSATION [ 0 | 1 ]
```

WIRE_SLICE_MIN_DIM

WIRE_SLICE_WIDTH

These keywords support improved 45-degree wire extraction accuracy for trapezoidal shapes. For 45-degree wire segments with bounding box x,y dimensions of at least <min_dimension> in microns, the geometry is sliced into the

specified width to obtain better extraction accuracy for R and C values (um).
Optional. Default: no slicing performed.

Syntax:

```
WIRE_SLICE_MIN_DIM <min_dimension>
WIRE_SLICE_WIDTH <width>
```

Characterization Keywords

APL_FILES

APL_FILES can be used instead of the ‘import apl’ command to specify multiple APL input files and directories at a time (a single ‘import apl’ command cannot support multiple files or any directory imports). The files are listed in a block in which the first column specifies a filename or directory and the second column specifies what type of file it is, such as current, device capacitance, pwcapp, or avm. At the end of the ‘setup design’ stage, all files/directories in the specified list are processed and imported. If there is a TCL command ‘import apl <current-file>’ or ‘import apl -c <cap-file>’ in the RedHawk command file, then that command is processed and the files/directories in the APL_FILES keyword are ignored. If there is a keyword ‘PIECEWISE_CAP_FILES <pwc-file>’ in the GSR file, then this piecewise file specification supersedes those listed in the APL_FILES list.

Note that when there is a very large number of APL files, the merging and validating process is much faster if only directories are specified, rather than a long list of files.
DMP compatible. Optional. Default: none.

Syntax:

```
APL_FILES {
    <input_file> [ current | cdev | pwcapp | avm |
        current_avm | cap_avm]
    ...
    ? <input_dir> [ current | cdev | pwcapp ]?
    ...
}
```

Example:

```
APL_FILES {
    aaa.spcurrent current
    bbb.cdev cdev
    ccc.pwcdev pwcapp
    avm.config avm
    vmememory.current current_avm
    vmememory.cdev cap_avm
    aaa_current_dir current
    bbb_cdev_dir cdev
    ccc_pwcapp_dir pwcapp
}
```

FAST_IMPORT_APL_MODE

When set, APL import is optimized for parallel interpolation data preparation and peak memory usage to reduce APL import time. *Optional. Default: 0.*

Syntax:

```
FAST_IMPORT_APL_MODE [ 0 | 1 ]
```

IGNORE_APL_CHECK

When importing APL data into RedHawk, ‘aplchk’ checks the quality of the data, which determines the quality of voltage drop results. When this keyword is turned on, this data checking is disabled. *DMP compatible. Optional. Default: 0 (do not ignore).*

Syntax:

```
IGNORE_APL_CHECK [ 0 | 1 ]
```

IGNORE_APL_CHECK_SWITCH

When set, ignores incorrect switch cell counts or multiple headers in switch models, which can occur when files are manually modified to merge multiple switch models into one, and errors can result. When set to 1, if APL switch cell counts do not match the number of cells in the switch cell file, RedHawk continues without an error. *DMP compatible. Optional. Default: 0 (do not ignore).*

Syntax:

```
IGNORE_APL_CHECK_SWITCH [ 0 | 1 ]
```

IGNORE_APL_PROCESS_CORNER

When turned on, supports reading of APL data for different process corners that have been merged in the same run. To merge data from different process corners in APL, use the ‘aplmerge’ command:

```
aplmerge -ignore_corner <filename1> <filename2> -o abc  
DMP compatible. Optional. Default: 0 (off).
```

Syntax

```
IGNORE_APL_PROCESS_CORNER [ 0 | 1 ]
```

Note: Importing different corner APL data is not recommended in most cases, since it may create incorrect RedHawk results.

IP_MODEL_CELL_MAP

Specifies that the default view of the model is to be used for all instances of an IP. Must be used with the IP_MODELS keyword. *DMP compatible. Optional. Default: None.*

Syntax

```
IP_MODEL_CELL_MAP {  
    <cell name> <default view name>  
}
```

IP_MODEL_INST_MAP

Specifies those instances of IP using a different view, rather than the default view, to expand the usage of keyword IP_MODELS. Must be used with keywords IP_MODELS and IP_MODEL_CELL_MAP. *DMP compatible. Optional. Default: None.*

Syntax

```
IP_MODEL_INST_MAP {  
    <inst_name> <cellname> <specified_instance_view>  
    ...  
}
```

IP_MODELS

Specifies the IP name and the path to its unified model directory. Each IP directory includes multiple views. RedHawk can read them and have all the possible views for the same IP. Then you can choose the specific view of the model for different instances of the same IP. *DMP compatible. Optional. Default: None.*

Syntax

```
IP_MODELS {  
    <IP name> <IP_model_directory>  
}
```

LIB2AVM

If APL characterizations are not available for some memory cells, for the dynamic flow RedHawk can automatically collect the necessary data from the design, LIB files and the GSR and create an AVM (Apache Virtual Model) configuration file in *adsRpt/avm.conf*, which describes parameters such as read/write/standby mode, power consumption, access time, setup time, load information, and current waveform type, for each memory type. RedHawk internally uses this config file to generate and use rule-based current profiles, leakage current and decoupling capacitance for memory blocks in dynamic analysis. The AVM utility then creates current profiles for these memory cells that have no APL data. The AVM models can be load dependent or not.

By default (1), a triangular-shaped profile is provided (which is composed of multiple triangles for complicated profiles). When set to 2, a trapezoidal-shaped current profile is provided. When set to 3, load-dependent triangular-shaped profile is provided. A value of 4 provides load-dependent trapezoidal-shaped current profile. Any available APL memory characterization data overrides internally-generated AVM data. A value of 0 turns off automated AVM config file generation. Memories are identified based on the following attributes in *.lib for AVM generation: ‘memory ()’, or ‘timing(){mode... }’. *DMP compatible. Optional. Default: 1 (triangular waveform).*

Syntax

```
LIB2AVM [ 0 | 1 | 2 | 3 | 4 ]
```

Example:

```
LIB2AVM 0
```

PS_IN_FLOW_APL

When set, the APL import flow is merged with power calculation to improve run time with large amounts of APL data. In particular, it correctly handles state- dependent ratios in Raw cell_view models. The GSR keywords *APL_INTERPOLATION_METHOD* and *FAST_IMPORT_APL_MODE* are supported in this flow. *DMP compatible. Optional. Default: 1(On).*

Syntax

```
PS_IN_FLOW_APL [ 0 | 1 ]
```

Timing Keywords

It is recommended that you specify the *CLOCK_ROOTS* keyword if *STA_FILES* is not specified, since RedHawk can automatically trace all the clock nets and clock domains for each instance that is traceable from the clock root. Note that clock root tracing is *not* performed if any STA files are specified.

ATE_CONSTRAINT_FILES

When this keyword is used and ENABLE_ATE is set, ATE generates the STA file during the setup design phase, and SPEF import is skipped. Since ATE already processes SPEFs for signal loads, RedHawk re-uses this C1-R-C2 information. *DMP compatible. Optional. Default: none.*

Syntax:

```
ATE_CONSTRAINT_FILES {
    <SDC_file1>
    <SDC_file2>
    ...
    USER_OPTIONS_FILE <TCL_file_name>
    FREQ_OF_MISSING_INSTANCES <freq-Hertz>
}
```

where

<TCL_file_name>: specifies a TCL file to enable you to pass options to ATE (such as various ADS_* variables or ATE setvars). If you provide such a file, it is added to the top of *ate.cmd* file it creates. This is the first thing in the *ate.cmd* file, so that users cannot override options that RedHawk intentionally sets.

FREQ_OF_MISSING_INSTANCES : specifies the operating frequency for instances missing the timing window in the STA file generated by ATE.

ATE_USE_REDHAWK_DB

When set, allows the ATE flow to obtain netlist information from the RedHawk database, instead of re-parsing DEF files. It is used when ENABLE_ATE is set to 1. Optionally, if you also specify SDC files using ATE_CONSTRAINT_FILES, the ATE generated STA file is used for both C1-R-C2 and STA purposes. *DMP Compatible. Optional. Default: 1.*

Syntax:

```
ATE_USE_REDHAWK_DB [ 0 | 1 ]
```

BOUND_SLEW_TO_MAX_TRANSITION

Controls whether to apply SLEW_MIN/MAX_TRANSITION limitations on instance pin transition time values. *DMP compatible. Optional. Default: 1 (On).*

Syntax:

```
BOUND_SLEW_TO_MAX_TRANSITION [ 0 | 1 ]
```

CLOCK_ROOTS

Traces the nets from a specified clock root and finds the respective clock domain for all the nets and instances. Specifies the names of the clock roots (either net name or pin name) and the frequency. No wildcard matching is supported for this keyword. Note that if any STA files are specified, CLOCK_ROOTS are not used. *DMP compatible. Optional. Default: None.*

Syntax:

```
CLOCK_ROOTS
{
    <clock_root_name> <freq in Hz>
    ...
}
```

Example:

```
CLOCK_ROOTS
```

```
{
    clock_root_1 160e6
    clock_root_2 160e6
    clock3:CLK 140e6
}
```

ENABLE_ATE

The Apache Timing Engine (ATE) is integrated into RedHawk. If this keyword is set to 1, the SPEF files are parsed using the ATE engine and annotated to the DEF. Advantage of the ATE spef parser is that it can annotate the SPEF data to instance/nets in the DEF irrespective of hierarchy limitations (that are present in the default RedHawk spef parser) and hence provide the best coverage/annotation for the SPEF input data. Optionally, if you also specify SDC files using ATE_CONSTRAINT_FILES, the ATE-generated STA file is used for both C1-R-C2 and STA purposes. The standard output from ATE is in file *adsRpt/ate.log*. Other ATE files go to .apache/ATE/. When set, the ATE SPEF parser also supports Signal EM analysis mode. *DMP compatible*. *Optional*. *Default: 0 (Off)*.

Syntax:

ENABLE_ATE [0 | 1]

INACTIVE_NETS

Specifies the running frequency of clock nets that are not specified by CLOCK_ROOTS, or for special nets such as Reset and Scan. The frequency of these nets is fixed throughout the analysis of power calculation and IR drop. Note that any STA file specifications override both INACTIVE_NETS and CLOCK_ROOTS specifications. *DMP compatible*. *Optional*. *Default: None*

Syntax:

```
INACTIVE_NETS
{
    <net_name> <frequency>
    ...
}
```

Example:

```
INACTIVE_NETS
{
    test 10e6
    reset 0
    ...
}
```

INPUT_TRANSITION

Defines input transition time for the input pins of all instances. *DMP compatible*. *Optional*. *Default: 10% of the value of the inverse of the frequency defined by the "FREQ" keyword.*

Syntax:

INPUT_TRANSITION <value in seconds>

Example:

INPUT_TRANSITION 0.2e-9 (or 0.2ns)

JITTER_ENABLE

Sets up conditions for PJX clock tree jitter analysis. For VCD-based analysis, the PJX Sign-off timing flow uses the scenario file dumped by PowerStream. However, this file contains only o/p pins' switching waveforms (it does not have switching information for control/input pins). A separate file is generated for the PJX Sign-off timing flow, `nxpsi_apache.scenario`, which also has the bias voltages for the control pins. When this keyword is set, RedHawk reads the `nxpsi_apache.scenario` file during simulation. *Optional. Default: 0.*

Syntax:

```
JITTER_ENABLE [ 0 | 1 ]
```

PSI_SPICE_CELL_NETLIST_FILE

File path specification used by PJX timing analysis to provide the path to the Spice cell netlist.

Syntax:

```
PSI_SPICE_CELL_NETLIST_FILE
{
    <spice_netlist_file_path>
}
```

Example:

```
PSI_SPICE_CELL_NETLIST_FILE
{
    Redhawk/Spice/design4B
}
```

SLEW_MIN_TRANSITION | SLEW_MAX_TRANSITION

For assigning a range for transition times to instance pins in the STA file and USER_STA_FILE (prior to slew normalization), you can set the minimum and maximum transition times using these keywords, in seconds. Note that these keywords do not control instance rise time values. *DMP compatible. Optional. Defaults: minimum - 5e-12, maximum - 1e-9.*

Syntax:

```
SLEW_MIN_TRANSITION <min_rise_sec>
SLEW_MAX_TRANSITION <max_rise_sec>
```

SPARAM_CHECK_LOWEST_FREQ

When on, checks the S-parameter package model to insure that the lowest frequency is <= 1 Hz. Otherwise, it errors out. Default is off.

Syntax:

```
SPARAM_CHECK_LOWEST_FREQ [ 0 | 1 ]
```

SPARAM_CHECK_REFERENCE_R

When on, checks the S-parameter package model to insure that reference impedance Zo is <= 2 Ohms. Otherwise, it errors out. Default is off.

Syntax:

```
SPARAM_CHECK_REFERENCE_R [ 0 | 1 ]
```

SPARAM_EXACT_DC

When On, more weight is assigned to DC accuracy for full-wave model generation of the S-parameter model. *Default: off.*

Syntax:

```
SPARAM_EXACT_DC [ 0 | 1 ]
```

SPARAM_HANDLING

When set, specifies the type of S-parameter package model to be created relative to passivity enforcement and behavior at low frequencies. PJX timing analysis supports multiple S-parameter models for the same REDHAWK_PKG if SPARAM_HANDLING is not set to 0, using the following keyword values:

- 0 - fits an RLCK model accurate at low frequencies (default)
- 1 - enables limited passivity enforcement
- 2 - enables full passivity enforcement

DMP compatible. Optional; default 0.

Syntax:

```
SPARAM_HANDLING [ 0 | 1 | 2 ]
```

STA_SLEW_SCALING

Scales the slew values read from STA file pointed in GSR by the Scaling Factor provided. *Default: 1*

Syntax

```
STA_SLEW_SCALING <scaling factor>
```

USE_CCS_PIN_CAPS

When set to 1, RedHawk derives slew-dependent receiver capacitance values from CCS Libraries that have receiver-slew v/s input-pin-cap tables. This keyword is only supported with ATE flow (ENABLE_ATE 1). STA file is dumped out from ATE with the slew dependent Cpin value for each instance. *Default: 0*

Syntax

```
USE_CCS_PIN_CAPS [ 0 | 1 ]
```

Simulation Keywords

BLOCK_PAR

Defines the ratio of peak cycle power to average power for all chip instances. Peak cycle power is defined as the power averaged over the worst cycle or the worst few cycles. The average power is defined as the power averaged over long normal operation cycles. *DMP compatible. Optional. Default: 1.0.*

Syntax:

```
BLOCK_PAR
{
    PAR <ratio_peak_power_to_avg_power>
}
```

Example:

```
BLOCK_PAR
{
    PAR 1.25
}
```

CONNECTIVITY_RES_THRESHOLD

Specifies acceptable resistance values for connection wires. To avoid reporting false shorts between nodes of packages connected with high resistance values, RedHawk static flow honors this keyword setting. Values at or above the threshold are not reported as shorts. *Optional. Default: 1e6.*

Syntax:

```
CONNECTIVITY_RES_THRESHOLD <threshold>
```

CONSISTENT_SCENARIO

In some cases different switching scenarios can be generated between SOC level runs for different conditions on the same or similar designs. When turned on (default), this keyword insures that the same or very similar designs have the same switching scenarios. *DMP compatible. Optional. Default: 1.*

Syntax:

```
CONSISTENT_SCENARIO [ 0 | 1 ]
```

DVD_GLITCH_FILTER

Defines conditions for filtering (ignoring) some values of minimum DvD over the timing window (called “minTW”), based on voltage value, and glitch width or glitch area conditions, for all clock cells and GDS macros, as shown in the Figure C-1 diagram. Minimum DvD values can be filtered globally or using instance-specific conditions. An output report listing the included and filtered DvD values is written to the file */adsRpt/Dynamic/<designName>.minTW_filtered*. *DMP Compatible. Optional. Default: none.*

Note: This keyword should not be used in scan mode or with power cycle selection.

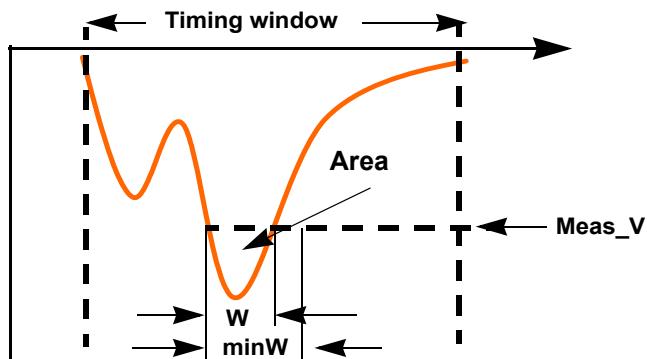


Figure C-1 Glitch width filtering with DvD_Glitch_Filter

Syntax:

```
DVD_GLITCH_FILTER {
    VOLTAGE_LEVEL <Meas_V>
    [ GLITCH_AREA <minA_psv> | GLITCH_WIDTH <minW_sec> ]
    CELL_LIST_FILE <filename>
}
```

where

VOLTAGE_LEVEL <Meas_V>: specifies the global voltage at which to measure glitch width, W

GLITCH_AREA <minA_psv> : specifies the global minimum glitch area value to report min DvD values over the time window. Units are pico Volt*seconds, abbreviated “psv”.

GLITCH_WIDTH <minW_sec>: specifies the global minimum glitch width value in seconds to report minimum DvD values over the time window

CELL_LIST_FILE <filename> : specifies the file containing glitch width or area conditions for specific instances. The file has the format:

```
<cell_name> <meas_V> <local_min_glitch_width/area>
```

which defines specific cell names, and the associated measurement voltage and minimum glitch width or area to be ignored for each instance (wildcards are acceptable). The cell-specific values override the global measurement voltage and minimum glitch width or area settings.

Example:

```
DVD_GLITCH_FILTER {
    VOLTAGE_LEVEL 1.15
    GLITCH_WIDTH 150ps
    CELL_LIST_FILE DvD_glitch
}
```

DYNAMIC_64BIT_SOLVER

When set, turns on 64-bit solver for analysis. However, the 64-bit solver is automatically used if the design has a very large number of nodes and also has a very high number of resistors per node. *DMP compatible. Optional. Default: 0 (off).*

Syntax:

```
DYNAMIC_64BIT_SOLVER [0 | 1]
```

DYNAMIC_ARC_LOW_POWER

When set, for multiple-Vdd cells in which one pin is connected to an always-On domain and another is connected to a Ramp-up domain, RedHawk performs normal dynamic simulation for the always-On domain and Ramp-up simulation for the ramp-up domain. For this kind of design noise coupling checks are performed, and DvD reports are generated. *Optional. Default: 0.*

Syntax:

```
DYNAMIC_ARC_LOW_POWER [0 | 1]
```

DYNAMIC_BYPASS_SHORT

When set, for RedHawk or CPM flow does not perform shorts checking during simulation. *Optional. Default: 0 (no bypass).*

Syntax:

```
DYNAMIC_BYPASS_SHORT [0 | 1]
```

DYNAMIC_CELL_CROSS_CHECK

When set, cross checks are performed at an early stage for extra pins defined in PGARC, but not in APL. These extra pins are then ignored in simulation, and the run continues. By default simulation errors out with an “inconsistent data” message when pin definitions are different between PGARC and APL. *Default: 0.*

Syntax:

DYNAMIC_CELL_CROSS_CHECK [0 | 1]

DYNAMIC_CLOCK_SCALE

When DYNAMIC_CLOCK_SCALE is turned On, RedHawk scales specified non-integer clock toggle rates. So if the clock toggle rate is specified as 1.5, 75% of clock instances toggle and the other 25% are idle (a rate of 2.0 is considered 100% toggling). If DYNAMIC_CLOCK_SCALE is turned Off there is no scaling; specified toggle rate between 1.5 and 2.0 is set to 2, and cells with specified rates below 1.5 are treated as regular cells. *DMP Compatible. Optional. Default: 0 (no scaling).*

Syntax:

DYNAMIC_CLOCK_SCALE [0 | 1]

DYNAMIC_DETACHED_POSTPROCESS

When set, uses a separate thread to perform post-processing during simulation. *Optional. Default: 0.*

Syntax:

DYNAMIC_DETACHED_POSTPROCESS [0 | 1]

DYNAMIC_PRESIM_DCINIT_SCALE

When the RedHawk starts presimulation, it computes initial conditions for the current flowing from the battery into the chip and package system being analyzed, in order to reduce the simulation time needed to get the system to steady state. This particularly effects designs with large response times between package (inductance) and die. During this process, RedHawk accounts for the initial voltage drop and the corresponding reduction in the current demand due to the voltage drop, and internally scales the initial current down. Use this keyword to specify a scaling value different than the default. Note that this variable only affects the initial starting point of pre-simulation; pre-simulation proceeds independently after the initial point has been determined. *Optional. Default: 0.8.*

Syntax:

DYNAMIC_PRESIM_DCINIT_SCALE <scale_value>

DYNAMIC_DISABLE_NEW_WFEXTRACT

When set to 1, turns off normal splitting of the Vdd file when executing the “plot voltage” command. By default the Vdd files with waveforms are split into multiple files to speed up run time, but this causes significant I/O operations. *DMP compatible. Optional. Default: 0 (off).*

Syntax:

DYNAMIC_DISABLE_NEW_WFEXTRACT [0 | 1]

DYNAMIC_EXTEND_VCD

By default if the presimulation period is not fully covered by VCD, no activity is assumed. When this keyword is set, presimulation activity is extended to cover the auto presim period, based on the VCD, so that simulation results are more realistic. *DMP compatible. Optional. Default: 0 (Off).*

Syntax:

DYNAMIC_EXTEND_VCD [0 | 1]

DYNAMIC_FF_ADJUST_NTRIG

When set, applies negative trigger switching (StandByNtrig C11 state) for all cycles on clock falling edges for FF/MEM instances. *Optional. Default: 1.*

`DYNAMIC_FF_ADJUST_NTRIG [0 | 1]`

DYNAMIC_FRAME_SIZE

When set, specifies the frame size in seconds for switching scenarios used for vectorless simulation. Or when set to ‘-1’ simulation automatically determines the frame size that is optimum for dynamic simulation-- that is, the frame size for which most of the instances switch within the frame. *DMP compatible. Default: 1/FREQ.*

Syntax:

`DYNAMIC_FRAME_SIZE [<frame_size_sec> | -1]`

DYNAMIC_FREQUENCY_AWARE

Invokes a frequency-aware vectorless flow that assists you in creating a simulation scenario in which the current demand from the circuit is modulated with respect to the frequency specified in the keyword. This scenario in turn can be used for DvD analysis or CPM creation. It is useful to create a voltage drop scenario with a resonance effect with the package and board netlists, to help understand the voltage drop issues when the die has this condition. *Default: none.*

Syntax:

`DYNAMIC_FREQUENCY_AWARE <freq_Hz>`

Example:

`DYNAMIC_FREQUENCY_AWARE 300e6`

In the example, the Ipwr is modulated at 300 MHz.

DYNAMIC_GROUP_WIRECAP

When set, for designs with large numbers of coupling caps, simulation groups neighboring coupling caps to reduce matrix complexity, and speeds up processing significantly. *Default: 0 (off).*

Syntax:

`DYNAMIC_GROUP_WIRECAP [1 | 0]`

DYNAMIC_MBFF_MODE

To enable multi-bit flip-flop handling to align the starting time of the triggering clock transition and all individual bit transitions at the same time. When set to 2, enables Multi-cycle Vectorless switching scenarios for Multi-bit Flip-flop to exhibit a new switching scenario every 2 clock cycles, similar to 1-bit Flip-flops and Combinational cell. *Default : off*

Syntax:

`DYNAMIC_MBFF_MODE [0 | 1 | 2]`

DYNAMIC_MCYC_TW

When turned on, DYNAMIC_MCYC_TW reports IR drop after the first toggle of the instance occurs in VCD analysis in the *mcyc_effvdd.rpt* file, which enables you to extract effective VDD values for clock cycles before an instance switches. *DMP Compatible. Optional. Default: 0 (Off).*

Syntax:

```
DYNAMIC_MCYC_TW [ 0 | 1 ]
```

DYNAMIC_MIXED_CONSISTENT_SCENARIO

When set to 1, removes clock power influence of VCD instances in vectorless scenario generation. *DMP compatible. Optional. Default: 0.*

Syntax:

```
DYNAMIC_MIXED_CONSISTENT_SCENARIO [ 0 | 1 ]
```

DYNAMIC_MSTATE_FILTER

When turned off, does not filter out events with different trigger pins for custom state cells during simulation, which is useful when custom state cells are included in the design. *DMP compatible. Optional. Default: 1 (on).*

Syntax:

```
DYNAMIC_MSTATE_FILTER [ 0 | 1 ]
```

DYNAMIC_PEAK_CURRENT_AWARE

When set, selects peak currents from APL for cells specified in the cell list. *Optional. Default: none.*

Syntax:

```
DYNAMIC_PEAK_CURRENT_AWARE <cell_list>
```

DYNAMIC_PGARC_REPORT_BASE

RedHawk ranks and reports the worst percentage DvD values for all arcs for the selected base parameter (default avgTW). You can select a different base value, since the worst MinTW, maxTW, or minWC values from one arc may not be the worst among multiple arcs for the same instance. Using this keyword you can change the base parameter on which to base the DvD reports. *DMP compatible. Optional. Default: avgTW.*

Syntax:

```
DYNAMIC_PGARC_REPORT_BASE [ avgTW | maxTW | minTW | minWC ]
```

Example:

```
DYNAMIC_PGARC_REPORT_BASE minTW
```

DYNAMIC_POST_BATCH

When large memory usage is required during post-processing (using GSR keyword 'DYNAMIC_REPORT_DVD 1'), this keyword controls the number of batches the output reports are divided into, and also gzips *mcyc_effvdd.rpt* when 'REPORT_REDUCTION 2(max)' is specified. By default **RedHawk** selects the appropriate number of output batches based on the size of the design. *Optional. Default: auto batch selection*

Syntax:

```
DYNAMIC_POST_BATCH <num>
```

DYNAMIC_PRECHECK

Performs simulation data pre-checking of key data integrity issues. *Optional. Default: On.*

Syntax:

```
DYNAMIC_PRECHECK [ 0 | 1 ]
```

DYNAMIC_PRESIM_TIME

Specifies the presimulation time in seconds to initialize capacitance charge and inductor current before time t=0 of simulation. You can also specify a presim Time Step Multiplier (TSM) to speed up the pre-simulation time, multiplying the length of the normal time step (set in the GSR keyword DYNAMIC_TIME_STEP). You can also select just an initial portion of the presim time to use the bigger TSM steps using the <TSM_fraction> option, with the remainder of the presim time at a normal time step. In the syntax below <presim_time_ps> is the specified presimulation time, <TSM> is the time step multiplier, and <TSM_fraction> is the fraction of the total presim time (0 to 1.0) at the beginning during which the longer time steps are used. If no values are set RedHawk sets the pre-simulation time to a value between 3.5 and 40ns in order to get accurate circuit starting conditions. The automatically-selected value of presim time is shown in the RedHawk log, and the TSM can be observed by looking at the waveform in the current waveform file, *adsRpt/Dynamic*. There is no upper or lower boundary on the presim time when set by the user. *DMP compatible. Optional. Default presim time set by RedHawk. (-1); default TSM=3.*

Syntax:

```
DYNAMIC_PRESIM_TIME [<presim_time_sec>] -1] ?<TSM> ?  
?<TSM_fraction>?
```

Example:

```
DYNAMIC_PRESIM_TIME 10e-09 5 0.9
```

In this example the presim time is set at 10e-09 seconds, the TSM is 5 and TSM_fraction is 0.9. This specifies a time step 5 times the length of the specified time step for the first 90% of the presim time, and a regular time step for the remaining 10% of presimulation.

DYNAMIC_RELAX_CONTROL_PIN_CONSTRAINT

Used for designs that have switch cells with multiple control pins and multiple control pins for the same switch are switching at almost the same time. When set to 1, insures that simulation honors the second control pin switching time, even if it is less than 200ps after the first control pin switches. This feature is default Off, meaning that simulation ignores the second control pin if it is switching too close to the time the first pin switches. *Optional. Default: 0.*

Syntax:

```
DYNAMIC_RELAX_CONTROL_PIN_CONSTRAINT [ 0 | 1 ]
```

DYNAMIC_REPORT_CLOCK_EVDD

When this keyword is turned on (1), RedHawk reports sequential instances' effective Vdd values based on the clock timing window, rather than based on the output TW. In the normal default flow, all the effective Vdd values are calculated based on output switching TW. *Optional. Default: 0.*

Syntax:

```
DYNAMIC_REPORT_CLOCK_EVDD [ 0 | 1 ]
```

DYNAMIC_REPORT_DECAP

Specifies what decap values are to be reported in the *adsRpt/Dynamic/decaps.rpt* output file. Values: 0: off (no decap report); 1: maximum current reported for all intentional decaps as defined in the GSR; 2: in addition to decaps reported in mode

1, reports also the maximum current associated with zero-power instances. *DMP compatible. Optional. Default: 0 (off).*

Syntax:

```
DYNAMIC_REPORT_DECAP [ 0 | 1 | 2 ]
```

Example:

```
DYNAMIC_REPORT_DECAP 1
```

DYNAMIC_REPORT_DVD

When set, enables reporting of TW-based effective voltage drop for all switching cycles in a dynamic simulation run. By default, for basic vectorless analysis, RedHawk reports TW-based voltage drop for the first switching cycle, and minCyc is reported for the full simulation. For all simulations covered by VCD files, for power transient simulation, and for multiple-cycle vectorless simulation ('perform analysis -mcyc'), RedHawk reports the TW-based voltage drop from all switching cycles. To report effectiveVDD in the first and last sliding windows, set "DYNAMIC_REPORT_DVD 2". The output file is *mcyc_effvdd.rpt*. *Optional. Default: 0 (off)*

Syntax:

```
DYNAMIC_REPORT_DVD [ 0 | 1 | 2 ]
```

DYNAMIC_REPORT_PRESIM_STATE

When set to 1, marks an instance with VCD as switching, if it is switching in the presimulation. *Default : 0*

Syntax:

```
DYNAMIC_REPORT_PRESIM_STATE [ 1 | 0 ]
```

DYNAMIC_SAVE_WAVEFORM

When set, controls creation of waveform plots for all Vdd/Vss nodes of all instances (or all transistors for MMX designs). When set to 2, enables plotting current per instance pin using TCL command 'plot current -instance {instname pinname}'. *Optional. Default: 1.*

Syntax:

```
DYNAMIC_SAVE_WAVEFORM [ 0 | 1 | 2 ]
```

DYNAMIC_SELECTIVE_SAVE

When turned off (0) saves all dynamic waveform files. By default (1) saves only waveforms for pad and pin nodes, and switch nodes for low power designs, which reduces disk space use for intermediate files. *Optional. Default: 1 (only key waveforms).*

Syntax:

```
DYNAMIC_SELECTIVE_SAVE [ 1 | 0 ]
```

DYNAMIC_SIMULATION_TIME

If specified, dynamic simulation uses the specified start and end times. For backward compatibility, if only one number is specified, that number is interpreted as the end time, and the assumed start time is t=0. There is a maximum simulation time limit of 200us for low-power analysis and 30us for vectorless analysis, unless it is overridden by setting the keyword IGNORE_SIMTIME_CHECK 1'. *DMP compatible. Optional. Default: 1/Frequency.*

Syntax:

```
DYNAMIC_SIMULATION_TIME ?<start_time_sec>? <end_time_sec>
```

Example:

```
DYNAMIC_SIMULATION_TIME 3200e-12
```

is equal to

```
DYNAMIC_SIMULATION_TIME 0 3200e-12
```

DYNAMIC_SOLVER_MODE

To perform high speed circuit modeling in RedHawk dynamic and CPM, by default an assumption of power-ground symmetry is made, meaning that the package power and ground impedances are considered similar. Setting DYNAMIC_SOLVER_MODE to 1 allows a P/G solution without this assumption, which increases the accuracy, although it is then a more time-consuming and memory-intensive method. This solver mode is recommended for asymmetric package models and when coupling between power and ground networks on the chip should be considered in simulation. *DMP compatible. Optional; default 0.*

Syntax:

```
DYNAMIC_SOLVER_MODE [ 0 | 1 ]
```

DYNAMIC_SORT_BY_PERCENTAGE

When turned on, analysis results for instances in the *.dvd file can be sorted by dynamic voltage drop percentage (minimum voltage drop over timing window), rather than by absolute voltage drop values, if this keyword is turned on. *Optional; default 0 (off).*

Syntax:

```
DYNAMIC_SORT_BY_PERCENTAGE [ 0 | 1 ]
```

DYNAMIC_TIME_STEP

Specifies the dynamic simulation time step. Note that specifying a time step over 30 psec is *not recommended. DMP compatible. Optional. Default: 10 psec.*

Syntax:

```
DYNAMIC_TIME_STEP <time in sec>
```

Example:

```
DYNAMIC_TIME_STEP 20e-12
```

DYNAMIC_VOLTAGE_CHECK

By default, voltage checking for node voltages greater than 8*(ideal VDD) are performed. If violating node voltages are found, the simulation run is terminated with a SIM-436 warning in the log file. To turn off these voltage checks, set this keyword to 0. Note that the substrate noise injection flow, using the 'perform analysis -inject_noise' command, automatically turns off this check. *DMP compatible. Optional. Default: 1.*

Syntax:

```
DYNAMIC_VOLTAGE_CHECK [ 0 | 1 ]
```

EFFECTIVE_VDD_WINDOW

The effective Vdd values reported in *adsRpt/Dynamic/<design>.dvd* file are based on (Vdd-Vss) values averaged over the timing window of that instance, as defined in the STA file. To obtain a valid effective Vdd, the values are averaged over several

small sample windows, not over the entire timing window defined in STA file. By sliding a small window through the entire timing window of the instance, several average effective Vdd values are obtained. The worst of the averages computed is then reported as the effective (Vdd-Vss) voltage value. If no timing window data are available, the DvD values based on timing window are not reported.

The size of the small measurement window sliding through the full timing window is instance-specific. By default the sliding window period starts when the instance's input signal is at the 50% level and ends when the instance's output signal is at the 50% signal level (as captured in the APL library for that cell). To change the definition of the sliding measurement window, use this one or more options of this keyword. The IN2IN option can be used for very narrow windows from one level to another level on the input signal. *Optional. Default: (input: 50% V, output: 50% V, minimum width: one time step).*

Syntax

```
EFFECTIVE_VDD_WINDOW {
    ? IN2OUT <inputV_fraction> <outputV_fraction> ?
    ? IN2IN <inputV_fraction1> <inputV_fraction2> ?
    ? MINIMUM <minWidth_ps> ?
}
```

Example:

```
EFFECTIVE_VDD_WINDOW {
    IN2OUT 0.2 0.7
    MINIMUM 10
}
```

In the above example, the sliding window is the time interval between when the instance's input signal is at the 20% level to the time the instance's output signal is at the 70% level, with a minimum sliding window duration of 10ps.

GROUND_CURRENT_DISTRIBUTION

When set, enables more accurate modeling of ground current distribution due to signal net loading on the chip. *DMP compatible. Optional. Default: 0.*

Syntax

```
GROUND_CURRENT_DISTRIBUTION [ 0 | 1 ]
```

HIERARCHY_CONSISTENT_SCENARIO

Supports consistent switching scenarios across multiple instantiations of the same block. For different instantiations of the same block that have the same power values, the switching scenario can be made similar to provide consistency in results. For example, a sample *scenario.cfg* file could be:

```
BLOCK_CONSISTENT_SCENARIO {
    blockA ?<dieA>?
    blockB ?<dieB>?
    ..
}
```

In this case the same switching scenario is generated for blockA and blockB having same power values. *Optional. Default: none.*

Syntax

```
HIERARCHY_CONSISTENT_SCENARIO <config_file>
```

ITERATIVE_SOLVER

Specifies the type of solver to use for Static simulation, as follows:

- 1 (default): RedHawk automatically decides the type of solver to use based on node count in the design.
- 0: uses the direct solver regardless of design parameters
- 1: uses the iterative solver regardless of design parameters

Note that the iterative solver uses less memory, but has a longer run time. The accuracy of the iterative solver also depends on the precision setting. *Optional.*
Default: -1. *DMP Compatible.*

Syntax

```
ITERATIVE_SOLVER [ -1 | 0 | 1 ]
```

Note: for Dynamic simulation control, see keyword NX_SIM.

MMX_ADAPTIVE_SAMPLING

Dynamic simulation during power analysis applies adaptive sampling of current profiles of MMX cells by default. When set to 0, this is most effective for memory designs. *Optional. Default: On.*

Syntax

```
MMX_ADAPTIVE_SAMPLING [ 0 | 1 ]
```

MULTI_CYCLE_SCENARIO

When set, makes the charge value consistent between cycles in multi-cycle analysis, enabling matching simulation results with frame-based PowerStream power for multiple-cycle and power-transient flows. *DMP compatible. Optional.*
Default: Off.

Syntax

```
MULTI_CYCLE_SCENARIO [ 0 | 1 ]
```

NX_SIM

Specifies the type of solver to use to optimize dynamic simulation performance, as follows:

- 0 - direct solver (CASI). Constant step waveform sampling is used, to reduce peak memory usage.
- 1 - direct solver (PRSIM). Provides faster run times, for example in simulation of dense grid APL3D cases.
- 2 - iterative solver (PRSIM).
- 3 - RedHawk decides the best solver based on design parameters. For example, it automatically selects the PRISM solver if the R/N ratio (ratio of the number of resistors to the number of nodes) is very high.

DMP compatible. Optional; default 0.

Syntax:

```
NX_SIM [ 0 | 1 | 2 | 3 ]
```

Note: for Static simulation control, see keyword ITERATIVE_SOLVER.

NX_VECTORLESS

The VLSG (VectorLess Scenario Generation) flow, which generates the switching scenario during the power calculation stage by PowerStream, is default On, as

shown in Figure C-2. This keyword is set to 0 to disable this feature. Note that VLSG is default Off if GSR VCD_PREPARE_SCENARIO 0' is set. Note also that the following flows and special conditions are not supported in the VLSG flow, and RedHawk uses the old flow:

- “import apl” or “import avm” command after “perform pwrcalc”
- “mesh *” or “eco *” commands
- “POWER_MODE” is “apl_peak”, “apl_peak1” or “apl1”
- If “NEW_STATE_PROPAGATION” is set to 0, then VLSG is disabled; GSR keywords DYNAMIC_MULTI_SCENARIO_PAIR, and MULTI_CYCLE_SCENARIO, and option ‘-mcycle’ are not supported by VLSG
- PARA_CALC_POWER is set to 0.

DMP Compatible. Optional; default 1.

Syntax:

NX_VECTORLESS [0 | 1]

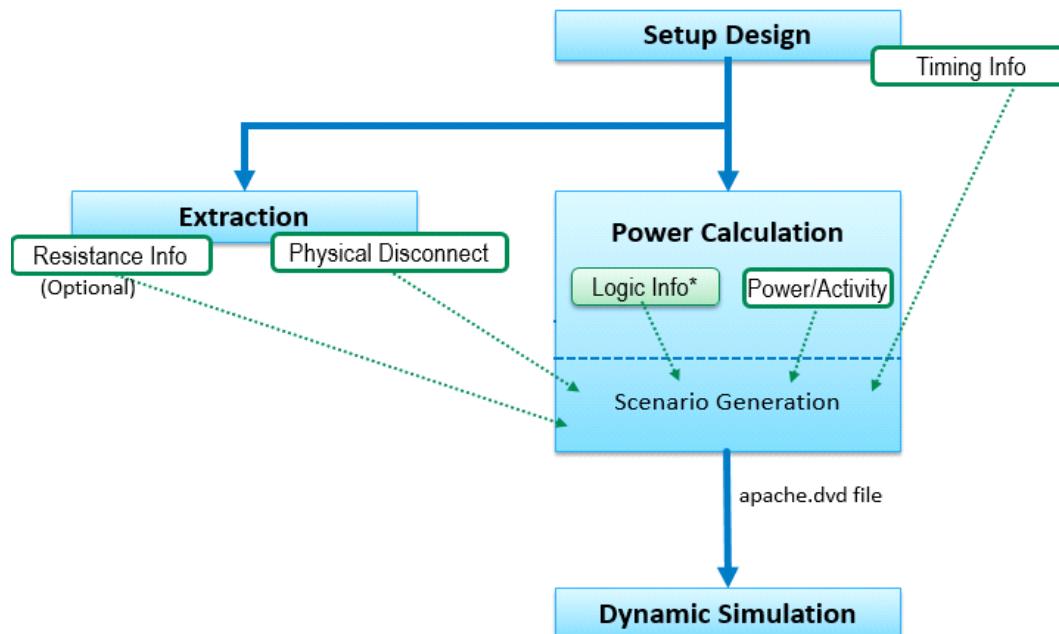


Figure C-2 Vectorless scenario generation

PIECEWISE_SWITCH_INPUT

This keyword is used to specify the file that defines the piecewise linear waveforms applied at the controlling pin of a power switch. In the definition file, each line has the following form:

```
<switch_inst_name> <ctrl_pin_name> ( <t1> <volt1> <t2> <volt2> ... )
...

```

where times are in picoseconds and voltages are in volts. *DMP compatible. Optional; default : none.*

Syntax:

PIECEWISE_SWITCH_INPUT <file_name>

PLOC_SHORT_INTERNAL_NET

In some designs internal net routing goes through the package along with die. When set, this keyword provides a lower resistive path for those instances that are sitting far away from the switch, which mimics the package effect in these cases when a package file is not available, it can be achieved by shorting all bumps on the required internal net. You must specify internal net names for bumps that need to be shorted together, as well as RLC values as appropriate. If RLC values are not provided, pads on internal nets are shorted to the simulated node voltage (not an ideal voltage source). *DMP compatible. Optional; default : none.*

Syntax:

```
PLOC_SHORT_INTERNAL_NET {  
    <Net1> ? <R1> <L1> <C1> ?  
    <Net2> ? <R2> <L2> <C2> ?  
    ...  
}
```

POWERUP_RANDOM_TOGGLE

When set, RedHawk starts random switching of instances in low-power domains when ramp-up voltage is more than half of ideal value. *DMP compatible. Optional. Default Off (no switching).*

Syntax

```
POWERUP_RANDOM_TOGGLE [ 0 | 1 ]
```

PROBE_NODE_FILE

Specifies a file containing a list of named probe locations to query for current and voltage results. The format of file contents is as follows:

```
<x_location> <y_location> <metal_layer> <probe_name>
```

Sample PROBE_NODE_FILE contents:

```
1023.2 3457.1 metal11 nodeAB  
1045.6 2828.1 metal22 nodeCD  
1980.2 1453.7 metal33 nodeEF
```

Note that the ‘probe’ TCL command also can be used to specify probes, and the command ‘plot current -probe -name <name>’ displays the current waveform. *DMP compatible. Optional. Default: none.*

Syntax:

```
PROBE_NODE_FILE <path_to_file>
```

SCAN_LAUNCH_CAPTURE_MODE

When set, executes scan capture mode as a post process for scan mode analysis. By default the shift procedure shifts test patterns serially into the combinational part of a sequential design prior to simulation. Capture mode captures test response of combinational logic parts after the shift process. *DMP compatible. Optional. Default: 0.*

Syntax:

```
SCAN_LAUNCH_CAPTURE_MODE [ 0 | 1 ]
```

SIMULATION_CACHE_DIRECTORY

Specifies a separate cache directory for simulation, to prevent issues with not enough disk space. *Optional. Default: none.*

Syntax:

```
SIMULATION_CACHE_DIRECTORY <sim_dir_path>
```

RECOMMENDED_SIMULATION_FACTORS

Specify scaling factors to set DYNAMIC_SIMULATION_TIME and DYNAMIC_FRAME_SIZE based on the recommended dynamic simulation time in log file. The SIMULATION_TIME_FACTOR and FRAME_SIZE_FACTOR sub-keywords specified within "RECOMMENDED_SIMULATION_FACTORS" is multiplied with "Recommended dynamic simulation time" reported by PowerStream to set DYNAMIC_SIMULATION_TIME and DYNAMIC_FRAME_SIZE respectively. Note that this keyword has effect only when DYNAMIC_SIMULATION_TIME and DYNAMIC_FRAME_SIZE keywords are not specified.

Syntax:

```
RECOMMENDED_SIMULATION_FACTORS {  
    SIMULATION_TIME_FACTOR <scale_factor1>  
    FRAME_SIZE_FACTOR <scale_factor2>  
}
```

Example:

```
RECOMMENDED_SIMULATION_FACTORS {  
    SIMULATION_TIME_FACTOR 4.0  
    FRAME_SIZE_FACTOR 2.0  
}
```

and redhawk.log contents:

Recommended dynamic simulation time, 2560psec ,to include 97.0370% of total power for DYNAMIC_SIMULATION_TIME in GSR.

Here, DYNAMIC_SIMULATION_TIME is set to 10240ps (that is 2560ps x 4.0) and DYNAMIC_FRAME_SIZE is set to 5120ps (2560ps x 2.0), where recommended dynamic simulation time is 2560psec from redhawk.log file.

STATIC_IR_HIGHLIGHT_PART_CON_INST

When set to 1, displays in IR map, the instances having multiple VDD pin with some pins unconnected. *Default: 0*

Syntax:

```
STATIC_IR_HIGHLIGHT_PART_CON_INST [ 1 | 0 ]
```

DMP Keywords

DMP_DYNAMIC_POWER_MODEL

When set to 1, supports DMP for the current and loading model flow in simulation, particularly for designs with millions of instances. *DMP compatible. Optional. Default: 0.*

Syntax:

```
DMP_DYNAMIC_POWER_MODEL [ 0 | 1 ]
```

DMP_EP_FAST_MODE

When set, when performing RTL VCD event propagation in the DMP flow, only one event is shared between partitions, instead of sharing multiple events. This provides faster event propagation and better performance, with some loss of

accuracy. By default, the number of shared events is the number of partitions.
Optional. Default: 0.

Syntax:

```
DMP_EP_FAST_MODE [ 0 | 1 ]
```

DMP_SETUP_BOUNDARY

Enables user to define the partition boundaries manually. The number of boundaries defined must be one lesser than the number of slave processes. Also it is normal that the final partition boundaries are slightly different from user-specified ones due to the internal algorithm used.

Syntax:

```
DMP_SETUP_BOUNDARY {  
bound1_x  
bound2_x  
...  
}
```

Example:

The below settings in GSR define the boundaries for a 4-way DMP run

```
DMP_SETUP_BOUNDARY {  
286.440002  
586.440002  
886.440002  
}
```

For cases where FDR is not able to create balanced partitioning (primarily due to via merging resulting in non-uniform merging during flattening design), the tool will dump out the coordinates that user can use for manual boundary setup in next runs in the log. A sample below:

```
dmpBalanceInfo: if this run has node balance problem, please  
set gsr KW,DMP_SETUP_BOUNDARY {  
1770  
3310  
}
```

DMP_SIM_KEEP_LAYERS

When set, user can perform selective layer-based network reduction. By default, DMP will do RC network reduction on all metal layers and all PG Nets. By using this GSR keyword, user can keep layers and nets out of any network reduction. It is useful for DMP flow in handling wirebond type of designs by keeping top metal layers out of reduction.

Syntax:

```
DMP_SIM_KEEP_LAYERS {  
<LAYER_NAME> VDD(VSS)_NET_NAME  
}
```

Example:

```
DMP_SIM_KEEP_LAYERS {  
RDL VDD18  
METAL9 VDD18  
RDL VSS  
METAL9 VSS  
}
```

"ALL" can be used either in layer name or net name, such as:

```
DMP_SIM_KEEP_LAYERS {  
    METAL9 ALL  
}
```

DMP_SP_FAST_MODE

When set, state propagation runs faster, with limited impact on the quality of results.
Optional. Default: 0.

Syntax:

```
DMP_SP_FAST_MODE [ 0 | 1 ]
```

PS_DMP_PERFORMANCE_MODE

When set, enables PowerStream to run on single machine similar to flat power calculation flow. This feature is not supported if "PS_IN_FLOW_APL 0" is set in GSR. *Default: 0*

Syntax:

```
PS_DMP_PERFORMANCE_MODE [ 0 | 1 ]
```

FAO General Keywords

FAO_ACCURATE_VOLTAGE

By default, performs accurate voltage difference calculations among nets improperly shorted together, by ignoring the shorts between nets. When turned off, voltage values with the nets shorted are obtained *Optional. Default: 1.*

Syntax:

```
FAO_ACCURATE_VOLTAGE [ 0 | 1 ]
```

FAO_ADD_STACK_VIA

When set (default), turns on stack via insertion for the FAO 'mesh'/'mesh via' commands. *Optional. Default: 1.*

Syntax

```
FAO_ADD_STACK_VIA [ 0 | 1 ]
```

FAO_BYPASS_CUTSIZE_CHECK

When set, turns off the cut size checks of minimum via size, which are based on the LEF VIARULE keyword setting for missing via reports. *Optional. Default: On.*

Syntax

```
FAO_BYPASS_CUTSIZE_CHECK [ 0 | 1 ]
```

FAO_HOLD_LIC

When set to 1, disables FAO license check-in and check-out, and holds the FAO license until RedHawk exits (default 0). This is useful when multiple FAO commands are executed in sequence. *Optional; default 0 (off).*

Syntax

```
FAO_HOLD_LIC [ 0 | 1 ]
```

FAO_IGNORE_COMPRESS_VIAS

When set, turns on layer minimum width computation at the beginning of setup design for DMP runs, and bypasses the FAO effects of compressing vias, which can cause many missing via differences between DMP and non-DMP runs. Sets min_width for both top metal layer and bottom metal layer, so it is more accurate.

Optional. Default: 0

Syntax:

```
FAO_IGNORE_COMPRESS_VIAS [ 0 | 1 ]
```

FAO_MISVIA_ONE_FILE

When set to 0, separates combined list of overlapping wire missing vias and stacked missing vias into two files. By default, lists are combined into one file. *DMP compatible. Optional; default 1.*

Syntax

```
FAO_MISVIA_ONE_FILE [ 0 | 1 ]
```

FAO_OBJ

FAO keyword and GSR keyword that selects the type of instance prioritization and results desired. *Optional; default: DVD*

Syntax

```
FAO_OBJ [ DVD | TIMING ?<slack threshold>? |  
         PATH ?<slack threshold>? ]
```

where

DVD : instances with highest voltage drop are selected for FAO

TIMING <slack_threshold>: all instances in the design with high delta delay and delta slack values caused by DvD are selected for FAO. The default value of the slack threshold is 0.

Note: To use FAO_OBJ TIMING you must execute 'import sdf xxx.sdf' before cell swapping, or the candidates for swapping are zero. Also, fao_region is still honored to filter out candidates from critical path or timing/slack info.

PATH <slack_worse than> : instances in critical timing paths with high delta delay and delta slack values caused by DvD are selected for FAO. The default value of the slack threshold is 0.

Example:

```
FAO_OBJ DVD
```

FAO_REGION

Specifies the boundary coordinates of a region for which optimization or modification of components intersecting the region should be performed, in micron units. This keyword also supports non-rectangular regions, but only with the 'decap fill', 'decap show' and 'decap remove -all' commands. Rectangular coordinates can also be used with 'mesh optimize', 'mesh fix', 'decap advise -place', and 'cell swap', and missing via reports. *DMP compatible. Optional; default: whole chip.*

Syntax:

```
fao_region { <x1> <y1> ... <xn> <yn> }
```

where

<xn> <yn> are the x,y coordinates of the points bounding the region, which should be specified in clockwise order starting in the lower left corner.

Example:

```
FAO_REGION {140 140 140 220 220 220 220 200 160 200 160 140 }
```

FAO_TURBO_MODE

Specifies fast FAO computation mode if set to 1 (default), or for the most accurate mode, which requires a longer calculation time, set value to 0. *Optional; default: 1.*

Syntax:

```
fao_turbo_mode [ 1 | 0 ]
```

FIX_WINDOW

Specifies the size of the horizontal and vertical bands in which the grid widths should be modified, centered on each voltage drop hotspot. This allows you to control the area in which wire changes can occur in the design. Can extend beyond the specified analysis region. Units: microns. Use with commands: mesh fix, decap advise -place, decap fill, decap remove, cell swap. *Optional; default: 100um x 100um*

Syntax

```
fix_window { x_dim y_dim }
```

Example:

```
fix_window { 400 400 }
```

NOISE_LIMIT

For the ordered list of worst case hot instances, the noise_limit specifies the voltage drop value below which the hot instance are not considered for mesh or decap optimization. For example, if noise_limit is set at 5%, hotspots or hot instances with less than a 5% dynamic voltage drop would not be considered for fixing. Use with commands: mesh optimize, mesh fix, and decap optimization. *Required for grid fixing; optional for decap optimization; default 0*

Syntax:

```
noise_limit <percent_voltage_drop>
```

Example:

```
noise_limit 5
```

NOISE_REDUCTION

Selects the target amount of change (decrease or increase) in voltage drop desired as a percentage of the existing voltage drop (from the last RedHawk run). For example, for an existing voltage drop of 100mV, to get an improvement (reduction) of 20mV requires setting this GSR keyword to 20. Use with commands: mesh optimize, mesh fix, decap optimize. *Optional; default: 0.*

Syntax:

```
noise_reduction <percentage_reduction>
```

Example:

```
noise_reduction 10
```

which targets a 10% reduction in present voltage drop

```
noise_reduction -10
```

which targets a 10% relaxation (increase) in present voltage drop

Grid Fixing and Optimization Keywords

FAO_DRC_DROP_RATIO

Specifies the fraction of maximum grid width (specified in ‘fao_range’) to use when there are potential metal spacing DRC problems. For example, an 0.8 value allows 80% of the max grid width to be used. A grid is not changed if the width change would cause a DRC error. Use with commands mesh optimize, mesh fix. *Optional; default: 1 (100% of max width)*

Syntax:

```
fao_drc_drop_ratio <fraction_of_max_width>
```

Example:

```
FAO_DRC_DROP_RATIO 0.8
```

FAO_DVD_TYPE

Specifies the type of dynamic voltage drop optimization to use for the FAO commands ‘mesh optimize/fix’. Integer values are assigned to select the following optimization choices:

- 1 - eff_vdd_tw : use average effective voltage (Vdd-Vss) over the timing window
- 2 - max_vdd_tw : use maximum effective voltage (Vdd-Vss) over the timing window
- 3 - min_vdd_tw : use minimum effective voltage (Vdd-Vss) over the timing window
- 4 - min_vdd_all : use minimum effective voltage (Vdd-Vss) over clock cycle

Optional; default: 1.

Syntax:

```
FAO_DVD_TYPE <integer 1-4>
```

Example:

```
FAO_DVD_TYPE 2
```

FAO_DYNAMIC_MODE

Specifies that FAO uses either static or dynamic mode for fix/optimize. Use with commands: mesh optimize, mesh fix. *Optional; Default: 0 (off)*

Syntax:

```
FAO_DYNAMIC_MODE [ 0 | 1 ]
```

FAO_LAYERS

Defines the layers or specific metal lines to be fixed or optimized by running FAO. Units: microns. Use with commands mesh optimize, mesh fix. *Required.*

Syntax:

```
FAO_LAYERS { { <layer name> ?L ?<direction>?  
?<wire_size_constraints> }... } ?
```

where

<layer name>: layer name as defined in the technology file

L: Length based constraint (l is also accepted) is supported. By default, width-based constraint is available.

<direction> : ver (wires running vertically) or hor (wires running horizontally)

<wire_size_constraints> : defines width allowed for specific wires. The following relational operators are allowed: >, <, >=, <=

Example 1:

```
FAO_LAYERS { { met4 ver > 1 < 3 } { met5 hor = 12 } }
```

Note: even if there is only one layer item, double brackets are required.

Example 2: With Length based constraint

```
FAO_LAYERS { { met4 ver > 1 < 3 } { met5 L hor = 12 } }
```

FAO_MISVIA_DISTANCE

When set, missing vias within the specified distance of each other are reported only once. By default, all missing vias are reported. *DMP compatible. Optional; Default: 0.*

Syntax:

```
FAO_MISVIA_DISTANCE <value_um>
```

FAO_MISVIA_RPT_SHORT

When set, places vias in a shorted area if this keyword is turned on. *Optional; Default: 0 (off).*

Syntax:

```
FAO_MISVIA_RPT_SHORT [ 1 | 0 ]
```

FAO_NETS

Specifies the power and ground nets to be fixed or optimized. The net names must be defined in the GSR file. Use with commands mesh optimize, mesh fix. *Required.*

Syntax:

```
FAO_NETS [ <net1> | {
    <net1>
    <net2>
    ...
} ]
```

Example:

```
FAO_NETS VDD
FAO_NETS {
    VDD2
    VSS2
}
```

FAO_PRECISE_VOLTAGE

When set, voltages reported in the missing via report match the values obtained from GUI clicks at the same location. *Optional. Default: 1.*

Syntax

```
FAO_PRECISE_VOLTAGE [ 0 | 1 ]
```

FAO_RANGE

Defines the minimum and maximum grid widths that are considered in grid fixing, and optionally limits the direction in which wire changes can occur. Units: microns. Use with commands mesh optimize and mesh fix. *Required.*

Syntax:

```
FAO_RANGE { { <layer> <min_width> <max_width>
    ? <extend_dir> ? } ... }
```

where

<extend_dir>: specifies the direction in which the width change should be made: bot/top/left/right.

Example:

```
FAO_RANGE { { met5 10 30 bot} { met6 15 40 } }
```

Note: Even if there is only one layer item, double brackets are required.

FAO_ROW_VDD_SITE

Allows FAO to handle multiple power domains simultaneously for the 'decap fill' command. The site name in a decap cell's LEF is checked against the site name of the row in DEF and the appropriate decap cells are then picked for the power domain, when the power domain and site names are specified using this command.

Syntax:

```
FAO_ROW_VDD_SITE {  
    <power_domain_name> <sitename1> <sitename2> ...  
}
```

Example:

```
FAO_ROW_VDD_SITE {  
    VDD1 CORE  
    VDD2 CORE2  
}
```

In the above example, DEF has two ROW sites defined: CORE that belongs to net VDD1 and CORE2 that belongs to VDD2. FAO adds decaps only in the domain(s) and sites defined by the keyword. This is currently supported only for the FAO command 'decap fill'.

FAO_SUB_GRID_NETS

Defines the wire ordering from lower/left to upper/right when creating new mesh strips. Use with command mesh sub_grid. *Optional; default: none*

Syntax:

```
fao_sub_grid_nets { <list of power/ground nets> }
```

Example:

```
fao_sub_grid_nets { VDD VSS },
```

where

{ VDD VSS} for a vertical mesh is for each pitch to create a wire for net VDD first, then to its right create a wire for a VSS net, with a pitch-to-pitch space specified by 'fao_sub_grid_spec'.

FAO_SUB_GRID_SPEC

Defines mesh parameters for creating subgrids (could be more than one layer). Use with command 'mesh sub_grid'. *Optional; default: none*

Syntax:

```
FAO_SUB_GRID_SPEC { { <layer> <dir> <space> <pitch>  
    <top_via_layer> <bot_via_layer> } ... }
```

where

<layer> is the mesh layer to create,

<dir> = h for horizontal or v for vertical mesh creation,

<space> specifies pitch-to-pitch spacing between wires of power nets specified in fao_sub_grid_nets,

<pitch> specifies center-to-center distance for wires created from the first net in fao_sub_grid_nets to the following set of power bus,

<top/bot_via_layer> specifies which top and/or bottom layer to drop (stack) vias; 0 for not dropping any vias.

Example:

```
FAO_SUB_GRID_SPEC { { MET3 h 6 12 MET4 MET2 }
{ MET2 v 5 10 0 MET1 } }
```

Note: Even if there is only one layer item, double brackets are required.

FAO_WIDTH_CNSTR

Specifies arithmetic constraint relationships between different metal layers. This GSR keyword can be used to relate the width of a particular wire to that of another mathematically. This GSR keyword is optional and intended for advanced usage. Units: microns. Use with commands: mesh optimize, mesh fix. *Optional; default: no specification.*

Syntax:

```
fao_width_cnstr { { layer1*<value> -/+ layer2*<value>
>/</>=/<= <value> } ... }
```

Example:

```
fao_width_cnstr { { met4 - met3*2 > 0 }
{ met5 - met4*3 > 10 } }
```

Note: Even if there is only one relationship item, double brackets are required.

NUM_HOTSPOT

Defines the number of highest voltage drop hot-spots that are targeted for fixing during mesh fixing or optimization. Use with commands: mesh optimize, mesh fix. *Optional; Default: 1*

Syntax:

```
num_hotspot <int>
```

Example:

```
num_hotspot 5
```

Decap Optimization Keywords

CAP_LIMIT

Defines an upper limit on the total amount of new decoupling capacitance that can be inserted by FAO. Units: pF. Use with commands decap advise, decap place, decap advise -place, decap fill. *Optional; default: unlimited*

Syntax:

```
cap_limit <cap_pF>
```

Example:

```
cap_limit 1000
```

which defines 1000 pf as the maximum added decap.

DECAP_CELL

Specifies the names of decap cells to be used in the design and their key physical parameters --cell width and height, effective capacitance and resistance, metal layer name, and leakage current. For information not specified in DECAP_CELL, RedHawk automatically extracts the needed parameter values from the LEF and APLCAP files. Information in DECAP_CELL supersedes that in APLCAP files. New

decap cells can be added to the design after dynamic analysis by using the 'gsr append DECAP_CELL' TCL command. The existing declared cells and parameter values can be changed using 'gsr set DECAP_CELL' prior to setup design. The present values can be displayed using 'gsr get DECAP_CELL'. Wild cards in cell names are supported.

You can use DECAP_CELL and BPA to create new decap cells/blocks and assign decap values. These decap blocks are also considered by the 'print decap' command. Decap cells can either come from LEF or be created in the flow. BPA instantiates the existing decap cells to put decap instances in the design. See the section "[Creating Decap Cells During BPA](#)", page 4-67, for more information. *DMP compatible. Optional (required for decap insertion in what-if and FAO analysis).*
Default: none.

Syntax:

```
DECAP_CELL {
    <decap_cellname> ? <width_um>? ?<height_um>? ?<C_pF>?
    ?<R_ohm>? ?<metal_layer_name>? ?<leakage_A>?
    ...
}
```

Example:

```
DECAP_CELL {
    decap1
    decapAB 0.25 0.75 0.01 1000 MET1 1e-9
    decapXY 0.75 0.75 0.03 1500 MET1 2e-9
}
```

DECAP_CELL_FILES

Allows importing multiple files that contain a list of decap cells. *DMP compatible.*
Optional; default: None.

Syntax:

```
DECAP_CELL_FILES {
    <decap_file1>
    ...
}
```

Example:

```
DECAP_CELL_FILES {
    user_data/demo1.dec
    user_data/demo2.dec
}
```

DECAP_DENSITY

Specifies a multiple of legally-placed decaps to be placed in non-overlap mode, to test for improvement in voltage drop performance. Use with commands: decap advise -place. *Optional; default: 1.0.*

Syntax:

```
decap_density <integer multiple>
```

Example:

```
decap_density 3
```

DECAP_TILE_MAX

Sets the decap value per tile (display rectangle) that is displayed as red (or the chosen color) in all types of decap maps. When set with a value greater than 0 pF per tile, each decap color map has the maximum capacitance displayed in red set to the same maximum value and the decap ranges for other colors are scaled accordingly. This allows different decap maps with different color range settings to be compared more easily. *Optional. Default: 0 (off).*

Syntax:

```
DECAP_TILE_MAX <value_pF>
```

Example:

```
DECAP_TILE_MAX 100
```

FAO_DECAP_FILL_ALG

When set to 1, allows you to maximize the amount of decap placed and minimize leakage. Provides multiple smaller decaps rather than the largest decap that fits in the available space if that provides a larger cumulative decap value. Also, if there are different types of decaps with the same decap value, but with different leakage values, decaps with lower leakage are selected. When set to 2, maintains good balance between DvD improvement and increased leakage power; it adds only enough decaps to supply the demand current for each instance. Optional. Default: 0 (off).

Syntax:

```
FAO_DECAP_FILL_ALG [ 0 | 1 | 2 ]
```

FAO_DECAP_FILL_NEW_FLOW

When turned on, ‘decap fill’ ignores small differences in rail wire width when snapping pins in a row or column of cells (enables acceptance of irregular rail widths). You can select the lower limit for wire width using the keyword FAO_WIRE_WIDTH_LOW_LIMIT. *Optional. Default: 0 (off).*

Syntax:

```
FAO_DECAP_FILL_NEW_FLOW [ 0 | 1 ]
```

FAO_DECAP_OVERLAP

Allows FAO to “place” decaps in non-legal placement areas (where there is not enough available space in present design), to analyze how the placement would affect dynamic voltage drop. Use with commands: decap advise -place. Turns on overlap if value is set to ‘1’ or ‘true’, Overlap is off if value is set to ‘0’ or ‘false’. *Optional; Default: false.*

Syntax:

```
fao_decap_overlap [true/1 | false/0]
```

Example:

```
fao_decap_overlap true
```

FAO_DRC_PL_OBS

When turned on, specifies that placement blockages in DEF are checked for DRC violations when performing ‘decap fill’. *Optional: default: 1 (on)*

Syntax:

```
fao_drc_pl_obs [ 0 | 1 ]
```

FAO_DRC_OBS

When turned on, specifies that signal net routing in DEF are checked for DRC violations when performing 'decap fill'. *Optional; default: 0 (off)*

Syntax:

```
fao_drc_obs [ 0 | 1 ]
```

FAO_ECO_NAME

Changes the default instance name pattern for decap cells. *Optional; default: <eco_string>=SH.*

Syntax:

```
fao_eco_name <decap_cell>_<eco_string><number>
```

Example:

```
fao_eco_name decapABC_XY3
```

FAO_MAX_SHIFT

Specifies a maximum movement distance when pushing cells away from a hot instance to make space for prefilling decaps. Use with commands: decap fill -prefill and decap fill -prefill_only. *Optional; default: 5 um*

Syntax:

```
fao_max_shift <max dist to move cells>
```

Example:

```
fao_max_shift 3
```

FAO_PLACE_GRID

Specifies an adjustment to the grid resolution ("tile" width) in microns, to get the correct placement for decaps on placement grid spacing, when the LEF file has incorrect or no SITE information. *Optional; default: none.*

Syntax:

```
fao_place_grid <grid_adjustment>
```

Example:

```
fao_place_grid 0.2
```

FAO_ROW_SITE

Specifies a particular set of "sites" (a placement grid) by its name in DEF, to avoid conflicting and overlapping decap insertions from multiple library row definitions. LEF "sites", also called "unit-tiles", are defined in LEF and used in the DEF/ROW section to create "row boxes" to allow P/R tools to create multiple clusters using different cell grid libraries, and thus may create overlapping ROW boxes. By default, without using FAO_ROW_SITE, FAO takes all ROW grids defined in DEF to create placeable decap areas. *Optional; default: all defined DEF row grids.*

Syntax:

```
FAO_ROW_SITE {  
<LEF_site_name>  
...  
}
```

Example:

For a DEF file ROW section containing several row definitions as follows:

```
ROW CORE_ROW_37549 bonuscoreaxg 3547600 2680600 N DO 7793 BY 1 STEP 690 0 ;
```

```
ROW CORE_ROW_52855 core 8927530 2680600 N DO 2054 BY 1 STEP 230 0 ;
ROW CORE_ROW_72819 bonuscore 8927530 2680600 N DO 684 BY 1 STEP 690 0 ;
```

Using 'FAO_ROW_SITE core', only rows defined by the 'core' site name are created for FAO decap placement.

FAO_WIRE_WIDTH_LOW_LIMIT

For the 'decap fill' command, this keyword sets a width tolerance limit (microns) for P/G wires composed of several segments with different widths when snapping pins in a row or column of cells. Wire segments whose widths are smaller than or equal to the specified limit are not considered by 'decap fill'. Set the keyword 'FAO_DECAP_FILL_NEW_FLOW 1' to enable the acceptance of irregular wire widths. *Optional; default: 0.*

Syntax:

```
FAO_WIRE_WIDTH_LOW_LIMIT <tolerance_u>
```

LEAKAGE_LIMIT

Specifies an upper limit on the total leakage power allowed from decaps added to the design. Units: Watts. Used with command 'decap advise -place'. *Optional; default: infinity*

Syntax:

```
leakage_limit <max decap power leakage>
```

Example:

```
leakage_limit 10e-6
```

NUM_HOTINST

Defines the number of high voltage drop instances that are targeted for improvement during decap placement. Used with commands 'decap advise', 'decap advise -place'. *Optional; default: 1*

Syntax:

```
num_hotinst <integer>
```

Example:

```
num_hotinst 3
```

Low Power Design Keywords

BLOCK_POWERUP_FILES

Compiles powerup data from multiple individual block analyses, and their associated time offset in seconds from simulation time t=0. Used for full-chip power analysis. *DMP compatible. Optional. Default: time offset =0.*

Syntax:

```
BLOCK_POWERUP_FILES
{
    <block_name-1> <FilePathName-1> [ time_offset-1 ]
    ...
    <block_name-n> <FilePathName-n> [ time_offset-n ]
}
```

Example:

```
BLOCK_POWERUP_FILES block_powerup.file
```

CHARGE_SWITCH

Specifies the names of switch cells used for charge switch operation in low power designs and their ramp-up threshold voltages (Volts). The RAMPUP_THRESHOLD option applies to charge switches without specific threshold values assigned. If a charge switch cell has both a charge control pin and function control pin, they must be specified. *Optional. Default: none.*

Syntax:

```
CHARGE_SWITCH {
    RAMPUP_THRESHOLD <Volts>
    <cell_name1> ? <Volts>?
    <cell_name2> <charge_control_pin> <funct_control_pin> ?<Volts>?
    ...
}
```

Example:

```
CHARGE_SWITCH {
    RAMPUP_THRESHOLD 0.8
    Charge_Sw_AB 0.7
    Charge_Sw_BCD CCpin3 FCpin7
}
```

CHECK_SWITCH_POWERON

When one power switch is controlled by two enable pins, you need to make sure that the proper threshold voltage is reached when the second control pin is enabled. Using CHECK_SWITCH_POWERON checks the proper turn-on of the switch. *Optional. Default: None*

Syntax:

```
CHECK_SWITCH_POWERON {
    [ THRESHOLD <threshold_all_V> |
      <cellname> <cell_threshold_V> ]
    ...
}
```

Example:

```
CHECK_SWITCH_POWERON {
    SW_ABC 0.2
    SW_CDE 0.1
    SW_xyz 0.15
```

DECOPPLE_LDO_GROUND

This keyword must be set both in the GSR and in the Aplo configuration file in order to ignore the LDO ground bounce effect in low power analysis. *Optional. Default: 0.*

Syntax:

```
DECOPPLE_LDO_GROUND [ 0 | 1 ]
```

DYNAMIC_ADAPTIVE_RON

Specifies adjustments to switch On resistance based on the variation in the gate voltage, and sets a threshold variation value. For most accurate results the power switch should be characterized using a multi-dimensional switch model file, enabled with the APLSW keyword 'MD_PWL 1'. If the default switch model is used, RedHawk uses internal heuristics to scale the switch Ron. *Optional; default: 0, Off.*

Syntax:

```
DYNAMIC_ADAPTIVE_RON [ 0 | 1 ] <variation_threshold_%>
```

Example:

```
DYNAMIC_ADAPTIVE_RON 1 8
```

The example specifies that the threshold for Ron change is an 8% variation in power or ground voltage.

DYNAMIC_GSC_CHECK

If this keyword is On, RedHawk checks for instances not controlled by power gates that are assigned a low-power related state (ON, OFF, POWERUP, POWERDOWN, POWERUP_DOWN, POWERDOWN_UP). If switching state assignments are detected for instances *not* part of a power gating network, a warning message is issued and the instances are assigned a “DISABLE” state. This prevents problems in analysis caused by incorrect state assignments. *Optional.* Default: 1 (on).

Syntax:

```
DYNAMIC_GSC_CHECK [ 0 | 1 ]
```

EXTRACT_INTERNAL_NET

When set, specifies that you do not need to specify the internal nets in the VDD_NETS list; RedHawk automatically traces the internal power nets and extracts them. *DMP compatible.* *Optional.* Default: 1.

Syntax:

```
EXTRACT_INTERNAL_NET [ 0 | 1 ]
```

PIECEWISE_CAP_FILES

Specifies the file that contains the piecewise linear (PWL) capacitance and leakage data for cells in a design, used for low power ramp-up analysis. *Optional.* Default: none.

Syntax:

```
PIECEWISE_CAP_FILES <file name>
```

Example:

```
PIECEWISE_CAP_FILES pwl
```

POWERUP_SAVE

Specifies the file for saving powerup data for a single block, for subsequent hierarchical full-chip analysis utilizing the BLOCK_POWERUP_FILE keyword. Used for low power design analysis. *Optional.* Default: none.

Syntax:

```
POWERUP_SAVE <FilePathName>
```

Example:

```
POWERUP_SAVE powerup_save.file
```

POWERUP_OUTPUT_HIGH_PROB

Specifies the probability that the output of any instance in a low power block is high during ramp-up analysis. *Optional.* Default: 0.5.

Syntax:

```
POWERUP_OUTPUT_HIGH_PROB <probability>
```

Example:

```
POWERUP_OUTPUT_HIGH_PROB 0.7
```

RAMPUP_OFFSTATE_VOLTAGE

If this keyword is set, the initial off-state voltage of all the internal power/ground nets are set to this value (volts). By default this keyword is not set, and simulation automatically calculates the initial off-state voltage. This keyword should only be used when you do not have a piecewise-linear capacitance library. *DMP compatible.* *Optional.* *Default: none.*

Syntax:

```
RAMPUP_OFFSTATE_VOLTAGE <initial_voltage>
```

Example:

```
RAMPUP_OFFSTATE_VOLTAGE 0.315
```

RAMPUP_SW_REPORT

When set in low power analysis, reports the 'worst_int_Vdd/Vss' (Volt) and 'worst_ext_Vdd/Vss' (Volt) into the switch_dynamic.rpt file. *Default: 0*

Syntax:

```
RAMPUP_SW_REPORT [ 1 | 0 ]
```

SWITCH_MODEL_FILES

Specifies the location of a switch model file, as required for power-up analysis and provides the capability to update the switch cell models. You can analyze the usability and effects of different size footer/header cells in a power-gated design. The switch model file is generated by the *aplsw* utility in characterization of the switches. *DMP compatible.* *Optional.* *Default: none.*

Syntax:

```
SWITCH_MODEL_FILES
{
    <switch_model_file>
}
```

Example:

```
SWITCH_MODEL_FILES
{
    SW_MOD_AB
}
```

SWITCH_MODEL_XTR_FILES

Allows you to define transistor pairs and their relative current distribution weight using specified files. When this keyword is used, the 'USE_MF_SWITCH_MODEL' GSR keyword is automatically turned on. The syntax for the contents of a model transistor file is as follows:

```
SWITCH_CELL <cell1> {
    <ext_x1> <ext_y1> <int_x1> <int_y1> <weight1>
    <ext_x2> <ext_y2> <int_x2> <int_y2> <weight2>
    ...
}
```

where

`ext_xN, ext_yN`: a point on the source/drain port, connected to the external net, of the switch transistor. Units are in um.

`int_xN, int_yN` : a point on the source/drain port, connected to the internal net, of the switch transistor.

`weightN` : the relative “weight” assigned to the specified transistor location for current distribution purposes (total weight for all points is 1)

This feature requires that there is only one sink layer for the switch cell, either by defining only a sink layer port in the LEF file, or using the keyword `CELL_PIN_FILE` to define the sink layer for a multi-layer port pin. If there are multiple definitions of `SWITCH_MODEL_XTR` for the same cell, then the last definition overwrites all the previous definitions.

Note that if the `SWITCH_MODEL_XTR_FILES` keyword is used, RedHawk uses the external/internal points defined in the file to determine the port geometries for those points landed, then pairs up the geometries as the points are paired in the file. Then RedHawk finds the nodes on the paired geometries and selects one from each node group on the paired geometries, and adds a transistor model between the two nodes for each pair. The relative weights are assigned to the transistors so that the simulator knows how to distribute the current on each transistor. The current distribution is calculated by the weight of the external/internal point, divided by sum of all the weights of all the parallel CONNECTED transistors of the switch cell. “CONNECTED transistors” means the transistors defined by the switch cell that are actually connected to the network. Note that RedHawk automatically picks up the `<cell>_adsgds.sw` file in `GDS_CELL` directory generated by GDS2RH and puts it in the `SWITCH_MODEL_XTR_FILES` list. *DMP compatible. Optional.*

Default: none.

Syntax:

```
SWITCH_MODEL_XTR_FILES {
    <file1>
    ...
}
```

SWITCH_RES_FILES

Specifies a Switch Resistance file defining RON values for each switch, and overrides values in the switch model file. The file format is as follows:

```
<switch_instance_name1> <RON1_Ohms>
...

```

Note that after starting a run, using the TCL command ‘`setup switchres <filename>`’ also specifies a Switch Resistance file overriding switch model file values. Then using ‘`setup switchres clear`’ causes the Switch Resistance file to be ignored, and returns Ron value control to the switch model file. *Optional. Default: none.*

Syntax:

```
SWITCH_RES_FILES {
    <file name>
}
```

USE_MF_SWITCH_MODEL

RedHawk can create pin instance node pairs between external and internal switch nets, assign a weight for each pair, and simulate separately the characterized switch model in parallel smaller switches with the proper weight, assuming equal weight for all the pairs (see keyword `SWITCH_MODEL_XTR_FILES`).

Automatic switch transistor recognition finds the minimum space between all the driver/receiver shapes, and assumes that the minimum space is the transistor length. Then all driver/receiver shape pairs that have a space equal to the minimum space found in the previous step are marked as switch transistor source/drain pairs. This keyword controls use of the multi-finger switch model, and removes group node shorting for switch cells if the multi-finger switch model is used.

When the design contains both multi-finger switch cells and regular switch cells, only cells with explicit multi-fingered switch information are treated as multi-fingered switch cells, and all other switch cells are treated as normal switch cells. However, you can force automatic multi-fingered switch cell treatment of all switch cells by setting this keyword to 1. *DMP Compatible. Optional. Default: 0.*

Syntax:

```
USE_MF_SWITCH_MODEL [ 0 | 1 ]
```

VP_CONTROL

Specifies the name of the Virtual Power Control file that describes the controlling pin information for switched IP macros. *Optional. Default: none.*

Syntax:

```
VP_CONTROL {
    <filename>
}
```

Example:

```
VP_CONTROL {
    vpcf_abcd4
}
```

Information in the VPC file has the following format (all times and delays in seconds):

```
<macro name> {
    POWERUP <block_ctrl_pin_1> <int_domain1> <pwrup_time> ?<daisy_chain_delay>?
    ...
    POWERUP <block_ctrl_pin_n> <int_domainN> <pwrup_time> ?<daisy_chain_delay>?
    POWERON <block ctrl pin_1> <int_domain1> ?<poweron_time>?
    ...
    POWERON <block ctrl pin_n> <int_domainN> ?<poweron_time>?
    SWITCH_RAMPUP_TIMING <sw_inst_name> <pwrup_time> ?<poweron_time>?
}
```

where

<daisy_chain_delay>: specifies the added delay for each successive switch in the form of a daisy chain. For distribution of the timing windows for daisy chain simulation to the internal domain switches, both the external and internal domains must be listed in the VP_CONTROL file.

Note: You must ensure that the timing windows for the specified pins appear in the timing file for RedHawk in order to execute a ramp-up simulation involving the switch IP block.

Three examples of using the VP_Control file are shown below.

Example 1 VPC file for global delay assignment:

```
Mem64x1024 {
    POWERUP PON VDDINT 100e-12
    POWERON PGOOD VDDINT 150e-12
```

}

The first line of the definition means that ‘PON’ is the name of controlling pin for the switches between the external net and the internal net ‘VDDINT’, and 100e-12 is the delay in seconds from PON to all the controlling pins of switches inside the memory macro for that virtual domain. The keyword ‘POWERUP’ means that PON is the charging pin and maps to the POWERUP state in the switch model. ‘POWERON’ means that ‘PGOOD’ is the controlling pin for the ON state in the switch model.

Example 2 PVC file for specifying daisy chain delay:

```
Mem64x1024 {
    POWERUP PON VDDEXT VDDINT 100e-12 20e-12
}
```

If there are three switch instances in Mem64x1024, namely adsU2, adsU3 and adsU4, then the controlling pin ‘PON’ of adsU2, adsU3 and adsU4 is assigned delay numbers of 100ps, 120ps and 140ps respectively, depending on the switch instance sequence in the block DEF file.

Example 3 VPC file for instance-based delay assignment:

```
Mem64x1024 {
    POWERUP PON VDDINT 50e-12
    SWITCH_RAMPUP_TIMING adsU2 100e-12
    SWITCH_RAMPUP_TIMING adsU3 120e-12
    SWITCH_RAMPUP_TIMING adsU4 140e-12
}
```

The ‘POWERUP’ statement must be specified. The SWITCH_RAMPUP_TIMING value overwrites the global delay assignment in POWERUP. For example, Mem64x1024 has switch instances adsU2, adsU3, adsU4 and adsU5. Switch adsU2, adsU3 and adsU4 are assigned delay numbers of 100ps, 120ps and 140ps respectively. Since SWITCH_RAMPUP_TIMING does not cover adsU5, a global delay of 50ps is assigned to adsU5.

ESD Keywords

ESD_CLAMP_FILE

Allows you to specify one or more ESD clamp cell definition files. When executing ‘perform esdcheck’, if a clamp file is not specified in the command, RedHawk picks up the clamp definition file from this keyword and executes ‘perform esdcheck’ on the clamp cells listed in the file.

The contents of the clamp files are saved into CMM models and loaded during the top run to provide ESD-aware macro models, based on and compatible with CMM (raw and binary). *Optional. Default: none.*

Syntax:

```
ESD_CLAMP_FILE <clamp_file>
```

or

```
ESD_CLAMP_FILE {
    <clamp_fileName1>
    <clamp_fileName2>
    ...
}
```

ESD_CLAMP_PIN_FILE

Specifies a clamp pin settings file describing the details for shorting clamp pin regions, and those with pseudo vias. The format of the clamp pin file is as follows (one cell per line):

```
<clamp_cellName> [all | shortByNet ] ? PIN {<pinName1> ... }? ? LAYER
{<layerName1> ... }? ? BBOX {<x1> <y1> <x2> <y2>}? ? MARGIN <distance_um>?
? INCLUDEALL [ 1| 2 ] ?  

...
where
```

all : shorts all nodes of the same pin

shortByNet : shorts all nodes of the same net for each instance of the cell

PIN : specifies a particular set of pins to be shorted

BBOX: specifies pins to be shorted in a particular BBOX

MARGIN : expands the bounding box by a distance in um for node shorting.

INCLUDEALL [1|2]: includes wire/via nodes in addition to pin nodes for shorting, where “1” means all wire/via nodes inside the specified shorting bounding box are to be shorted, and “2” means all wire/via nodes inside the expanded MARGIN shorting bounding box, but *not inside* the basic shorting bounding box, are to be shorted.

A sample ESD_CLAMP_PIN_FILE is shown below:

```
CellA PIN { VDD } LAYER { MET1 MET2 }
CellB all MARGIN { 0.3 } INCLUDEALL { 2 }
...

```

In the example, for CellA, all pins of VDD in MET1/MET2 are shorted together and pushed out. For CellB, all wire/via nodes inside the expanded (by 0.3 um) shorting bounding box, but not inside the basic shorting bounding box, are to be shorted.

The file also can be specified using the command 'pfs import clamp_pin <file_name>' to import the setting file. Alternatively, the same results can be achieved using the TCL command 'pfs add clamp_pin <cell_name>...' (see [section "pfs", page D-798](#)). *Optional. Default: none.*

Syntax:

```
ESD_CLAMP_PIN_FILES <file_name>
or
ESD_CLAMP_PIN_FILES {
    <file_name1>
    <file_name2>
    ...
}
```

ESD_CLAMP_PIN_NODE_DISTANCE

Specifies the pitch (um) at which to add more nodes on each partitioned finger segment. *Optional. Default: 0 (no nodes are added).*

Syntax:

```
ESD_CLAMP_PIN_NODE_DISTANCE <distance>
```

ESD_DEF_IGNORE_LAYER

Specifies layers to be ignored in DEF files that do not contain any clamp devices, to speed up ESD analysis. *Optional. Default: none.*

Syntax:

```
ESD_DEF_IGNORE_LAYER {
    <layer1>
    <layer2>
    ...
}
```

ESD_EXTRACT_CLAMP_NET

Specifies whether to automatically promote local nets between clamp devices that are not exposed to the top level and are not extracted by default, and also whether to add a ploc for the nets extracted. For clamp cells specified in the keyword ESD_CLAMP_PIN_FILE, this keyword (and the ESD_LOCAL_NETS keyword) control this behavior, as follows:

- 0 - the feature is turned off (default)
- 1 - adds a ploc and extracts nets that connect 2 or more clamp instances.
- 2 - adds a ploc and extracts all nets that are connected to any clamp instance.

Optional. Default: 0

Syntax:

```
ESD_EXTRACT_CLAMP_NET [ 0 | 1 | 2 ]
```

ESD_GSR_OPTIMIZE

When set, activates ESD capacity improvement feature. The optimized flow generates a new GSR file that selectively removes ESD-irrelevant cells/layers to reduce design size while keeping acceptable accuracy. There are three methods used to improve design capacity:

- Remove non-ESD standard-cell cells (for B2B/B2C/C2C/CD/R checks).
- Remove layers that reduce the number of ESD instances on each DEF file (for B2B/B2C/C2C/CD/R checks).
- Remove SPLIT_VIA_ARRAY or SPLIT_SPARSE_VIA_ARRAY settings (for non-CD checks).

When ESD_GSR_OPTIMIZE is set, the **Redhawk** session terminates at the beginning of the setup design stage and a new *esd.gsr* file is generated in *adsRpt/ESD*. You must then rerun **Redhawk** with *esd.gsr* to get an optimized design for ESD checking.

Note that since B2I/C2I/C2M/D2R checks need to access other instances that could be anywhere in the design, there is no design reduction for B2I/C2I/C2M/D2R checks. *Optional. Default: 0*.

1. When 'ESD_DEF_IGNORE_LAYERS' is set, the setting (as DEF_IGNORE_LAYERS) to DEF blocks that does not have clamp instances in/under it, to trim the lower layer wires/vias are applied.
2. When 'FAST_DEF_READ <N>' is set (<N> greater than 0), the flow will run all the way to the end w/ reduced data size.
3. When 'FAST_DEF_READ' is 0 (default), it is for the flow to run up to setup design, and it will dump out these 2 files for users' reference:
 - a. adsRpt/ESD/esd_block.rpt (this is always dumped out): Design hierarchy info, including DEF block and DEF file names, clamp instance, cell, type and layer info in/under every DEF blocks.
 - b. adsRpt/ESD/esd.gsr: Suggested GSR setting with IGNORE_CELLS and DEF_IGNORE_LAYERS in it.

Syntax:

```
ESD_GSR_OPTIMIZE [ 0 | 1 | 2 | 3 ]
```

where

- 0 (or off) : default.
- 1 (or r_check) : For B2B/B2C/C2C resistance check,
- 2 (or cd_check) : For ESD-CD, but ignore stdcells and decaps (so no detailed instance voltage reports).
- 3 (or rule_check): Examines ESD rule file(s) in ESD_RULE_FILE and then decide.

The functionality for 1 and 2 , ESD_CLAMP_FILE and/or ESD_CLAMP_PIN_FILE must be specified in GSR. For 3, ESD_RULE_FILE is also required in GSR.

ESD_LOCAL_NETS

Specifies which nets are subject to ploc insertion and are extracted, along with ESD_CLAMP_PIN_FILE and ESD_LOCAL_NETS keyword settings. *Optional.* Default: none.

Syntax:

```
ESD_LOCAL_NETS {  
    <netName1>  
    <netName2>  
    ...  
}
```

ESD_RULE_FILE

Allows you to assign ESD rule file names, so that when executing 'perform esdcheck', if a rule file is not specified in the command, RedHawk picks up the rule name specified in this keyword and executes 'perform esdcheck'. *Optional.* Default: none.

Syntax:

```
ESD_RULE_FILE <rule_filename>  
or  
ESD_RULE_FILE {  
    <rule_filename1>  
    <rule_filename2>  
    ...  
}
```

ESD_SHORT_CLAMP_PIN

To short ESD clamp pin nodes during extraction. *Optional.* Default: -1.

Syntax:

```
ESD_SHORT_CLAMP_PIN [ -1 | 0 | 1 | 2 ]
```

Where:

- 1: Do not short ESD clamp pin nodes during extraction if they are not specified in ESD_CLAMP_PIN_FILE.
- 0: Do not short ESD clamp pin nodes during extraction.
- 1: Short all ESD clamp pin nodes during extraction.

2: Short ESD clamp pin nodes when x/y/layer are specified in ESD_CLAMP_FILE during extraction

ESD_SIGNAL_NETS

Allows you to create a list of signal nets for ESD checking. When the net is in top block, cell name is optional and is required if the net is not in the top block. *Optional.* *Default:* none.

Syntax:

```
ESD_SIGNAL_NETS {  
    <netname1> <cell_name1>  
    <netname2> <cell_name2>  
    ...  
}
```

ESD_SIGNAL_NET_FILE

Allows you to specify a file containing a list of signal nets for ESD checking. When the net is in top block, cell name is optional and is required if the net is not in top block. *Optional.* *Default:* none.

Syntax:

```
ESD_SIGNAL_NET_FILE <filename>
```

where the file contains the names of signal nets, in the format:

```
<netname1> <cell_name1>  
...  
...
```

Name Mapping Keywords

DEFER_VIA_CREATION

For designs with a very large number of top DEF via models (over 60K), when this keyword is set RedHawk merges the excess via models with via rules having the same definitions in top DEF. To perform excess via model merging, RedHawk maps all via models with the same definition to a single via model name. The file *adsRpt/viaModelNameMapping.rpt* is generated that lists name mapping for all via models. The format of the file is:

```
<viaModel_name> <mapping name>
```

DMP compatible. *Optional.* *Default:* 0.

Syntax:

```
DEFER_VIA_CREATION [ 0 | 1 ]
```

NAME_CASE_SENSITIVE

Defines the name case sensitivity. If the value is set to 1, all *.lib*, *.lef/def*, *.spf/dspf*, *.vcd*, and STA filenames are assumed to be case-sensitive. *DMP compatible.* *Optional.* *Default:* 1.

Syntax:

```
NAME_CASE_SENSITIVE [ 0 | 1 ]
```

BUS_DELIMITER

Defines the characters used to delimit bus bits in the RedHawk database and GSR. *DMP compatible.* *Optional.* *Default:* []

Syntax:

BUS_DELIMITER <delim>

Example:

BUS_DELIMITER ()

PIN_DELIMITER

Defines a special character to separate net or instance names from pin names in RedHawk working files. Any character can be used that is not reserved in LEF or used in pin names. *DMP compatible. Optional. Default: :*

Syntax:

PIN_DELIMITER <delim>

Example:

PIN_DELIMITER @

BUS_DELIMITER_STA

Defines the characters used to delimit the bus bits in STA files. *DMP compatible. Optional. Default: []*

Syntax:

BUS_DELIMITER_STA <delim>

Example:

BUS_DELIMITER_STA ()

PIN_DELIMITER_STA

Defines the character between instance and pin names in STA files. *DMP compatible. Optional. Default: /*

Syntax:

PIN_DELIMITER_STA <delim>

Example:

PIN_DELIMITER_STA :

HIER_DIVIDER

Defines the character used to specify the hierarchy in the RedHawk database and GSR. *DMP compatible. Optional. Default: /*

Syntax:

HIER_DIVIDER <divider>

Example:

HIER_DIVIDER /

HIER_DIVIDER_STA

Defines the character used to specify the hierarchy in STA files. *DMP compatible. Optional. Default: /*

Syntax:

HIER_DIVIDER_STA <divider>

Example:

HIER_DIVIDER_STA /

Warning and Error Message Keywords

WARNING_COUNTS

Specifies the maximum number of Warning messages per type to be reported to *stdout* and *apache.log.xxxx*. *DMP compatible*. *Optional*. *Default: 3*.

Syntax:

```
WARNING_COUNTS [ <number_mess> | 0 | -1 ]
```

where

<number_mess> : specifies the maximum number of messages to output per type.

0 : no messages output

-1 : all messages output

Example:

```
WARNING_COUNTS 10
```

WARNING_LOG_COUNTS

Specifies the maximum number of Warning messages per type that are reported to *adsRpt/redhawk.warn*. *DMP compatible*. *Optional*. *Default: 100*.

Syntax:

```
WARNING_LOG_COUNTS [ <number_mess> | 0 | -1 ]
```

where

<number_mess> : specifies the maximum number of messages to report per type.

0 : no messages reported

-1 : all messages reported

Example:

```
WARNING_LOG_COUNTS 10
```

ERROR_COUNTS

Specifies the maximum number of Error messages per type that are reported to *stdout*. *DMP compatible*. *Optional*. *Default: 3*.

Syntax:

```
ERROR_COUNTS <number_mess>
```

Example:

```
ERROR_COUNTS 10
```

ERROR_LOG_COUNTS

Specifies the maximum number of Error messages per type that are reported to *adsRpt/redhawk.err*. *DMP compatible*. *Optional*. *Default: 100*.

Syntax:

```
ERROR_LOG_COUNTS [ <number_mess> | 0 | -1 ]
```

where

<number_mess> : specifies the maximum number of messages to report per type.

0 : no messages reported

-1 : all messages reported

Example:

```
ERROR_LOG_COUNTS 100
```

Ignore Function Keywords

IGNORE_CELLS

Specifies the list of cells to be ignored during importing DEF files (they are not imported into the design DB). For example, filler cells contribute no IR-drop and therefore can be ignored. If a DEF file path is *not* provided, cells specified are ignored in all hierarchies in the design, within the top cell and blocks. Cells specific to a block to be ignored should have the block's DEF file path provided. Wildcards are supported, with "?" representing a single character and "*" representing any number of characters. *DMP compatible. Optional. Default: none.*

Syntax:

```
IGNORE_CELLS
{
    <cell_name1> ? <DEF_file_path> ?
    ...
}
```

Example:

```
IGNORE_CELLS
{
    filler_cell12 /DEF/blockCDE.def
    filler_cellA17
}
```

IGNORE_CELLS_FILES

Specifies a file that contains the list of cells to be ignored by RedHawk during data input. The file containing the list referred to must have only one entry per line. When used in conjunction with IGNORE_CELLS above, both lists are read in. *DMP compatible. Optional. Default: none.*

Syntax:

```
IGNORE_CELLS_FILES <Exclude_cells_file_pathName>
```

Example:

```
IGNORE_CELLS_FILES /home/users/customerA/cells_exclude
```

IGNORE_DEF_ERROR

Normally if there are DEF data errors, undefined vias, or DEF blocks without power and ground pins, the RedHawk session is terminated with an error report. If the keyword IGNORE_DEF_ERROR is set to 1, RedHawk reports the DEF errors and continues. *DMP compatible. Optional. Default: 0 (perform check).*

Syntax:

```
IGNORE_DEF_ERROR
```

IGNORE_ERROR_POPUP

Eliminates unwanted pop-up Error dialog display. When set to 1, all error/warning pop-up windows are ignored, and the GUI is closed in the event of a crash, but command line control is still maintained, and the run continues, so that user command scripts can continue. This avoids delays in waiting for user response in a pop-up dialog box. *DMP compatible. Optional. Default: 0 (Off).*

Syntax:

```
IGNORE_ERROR_POPUP [ 0 | 1 ]
```

IGNORE_ESCAPE_CHAR

If IGNORE_ESCAPE_CHAR is set to true (default), the escape character (\) in names are ignored (removed). If the keyword is set to false, escape characters are kept in names. *DMP compatible. Optional. Default: true.*

Syntax:

```
IGNORE_ESCAPE_CHAR [ true | false ]
```

IGNORE_FILE_PREPARSE

Specifies whether the GSR file is pre-parsed prior to running RedHawk for syntax errors, missing information, and/or zero-byte-sized files that can potentially cause a hang or crash in later stages. *DMP compatible. Optional. Default: 0 (do not ignore preparse).*

Syntax:

```
IGNORE_FILE_PREPARSE [ 0 | 1 ]
```

IGNORE_FILLER_DECAP_CELL_REPORT

Ignores filler/decap cells in output reports. To speed processing of designs dominated by filler/decaps, this feature keeps filler/decap cells in extraction/simulation to avoid continuity issues, but ignores them in post-processing and reporting. *DMP compatible. Optional. Default: 0 (do not ignore)*

Syntax:

```
IGNORE_FILLER_DECAP_CELL_REPORT [ 0 | 1 ]
```

IGNORE_FLOATING_INSTANCE_MISSING_TW_CHECK

Allows you to identify inactive instances in a design, such as fillers, decaps, and tie-off instances, and remove them from the STA annotation reports, such as *adsRpt/apache.tw0*. To invoke this feature, set the keyword to 1. Details in the *adsRpt/redhawk.log* file are shown in the following example:

```
Checking instances in the design are not covered in the timing file(s).
Total number of FILLER floating instances (no input) = 102291
Total number of instances traced (CONST/STA/tracer) = 11702/161182
(6508/5194/0)
```

DMP compatible. Optional. Default: 0 (off).

Syntax

```
IGNORE_FLOATING_INSTANCE_MISSING_TW_CHECK [ 0 | 1 ]
```

IGNORE_FLOATING_PAD

To ignore floating plocs from both GUI and PG.ploc in the analysis. *Default : 0*

Syntax:

```
IGNORE_FLOATING_PAD [ 1 | 0 ]
```

IGNORE_GDSMEM_ERROR

Specifies whether parent *./ib/.lef* files for cells that have gone through *gds2def -m* or *gds2rh -m* are ignored if missing. *DMP compatible. Optional. Default: 0 (perform check).*

Syntax:

```
IGNORE_GDSMEM_ERROR [ 0 | 1 ]
```

IGNORE_GDS2DEF_UNCONNECTS

Specifies the handling of gds2rh-inserted sink pins in the *adsRpt/<design>_<net>.PinInst.unconnect* file. When set to 1, pin instances inserted by gds2rh are removed from the file. Note that only instances that are logically connected to analyzed nets, but physically disconnected, are listed in the file **.PinInst.unconnect*. *Optional. Default: 0*

Syntax:

```
IGNORE_GDS2DEF_UNCONNECTS [ 0 | 1 ]
```

IGNORE_INSTANCES

Specifies the list of instances to be ignored by RedHawk during data input (they are not imported into the design DB). For example, the filler instances contribute no IR-drop and can often be ignored. For both flattened and hierarchical designs the ignored instance names should be the exact names from the associated DEF_FILES keyword. If a DEF file path is *not* provided, instances specified are ignored in all hierarchies in the design, within the top cell and blocks. Instances specific to a block to be ignored should have the exact block DEF file path provided. *DMP compatible. Optional. Default: none*

Syntax:

```
IGNORE_INSTANCES {  
    <exact_DEF_instance_name1> ? <exact DEF_FILES path1> ?  
    ...  
}
```

Example:

```
IGNORE_INSTANCES {  
    filler_instance_name1 /DEF/blockA.def  
}
```

IGNORE_INSTANCES_FILES

Specifies a file that contains the list of instances to be ignored during building connectivity and reporting. The file containing the list referred to must have only one excluded instance per line. For both flattened and hierarchical designs the ignored instance names in the file should be the exact names from the associated DEF_FILES keyword. Note that comments are allowed in the specified file, as RedHawk ignores lines starting with "#". *DMP compatible. Optional. Default: none*.

Syntax:

```
IGNORE_INSTANCES_FILES <exclude_instances_file_pathname>
```

Example:

```
IGNORE_INSTANCES_FILES /home/users/customerA/nets_exclude
```

IGNORE_LEF_DEF_MISMATCH

Defines the DEF import operation. When set to 0, the DEF import operation stops when a pin instance name in the NETS section of a DEF file is not defined for the corresponding cell in the LEF file. *DMP compatible. Optional. Default: 0 (do not ignore).*

Syntax:

```
IGNORE_LEF_DEF_MISMATCH [ 0 | 1 ]
```

IGNORE_LEF_CELL_PIN_FILE

Specifies a file that contains the LEF cell pins to be ignored (that are not important in analysis) and therefore these pins are not reported in 'adsRpt/*.power.unused'. Wildcard(*) is supported for '<cellName>'.

Syntax:

```
IGNORE_LEF_CELL_PIN_FILE <file_name>
```

File format:

<cellName>	< pinName>
and2x	VBP
and2x	VBN
*	VDDR

IGNORE_LEF_PIN_DIRECTION

When set to 1, RedHawk does not change the OUT pin direction to IN. By default all P/G pins with direction OUT defined in LEF PIN are set to IN. *Optional. Default: 0.*

Syntax:

```
IGNORE_LEF_PIN_DIRECTION [ 0 | 1 ]
```

IGNORE_IO_POWER

Specifies that power associated with I/O cells will not be considered during power calculation. *DMP compatible. Optional. Default: 0 (do not ignore).*

Syntax:

```
IGNORE_IO_POWER [ 0 | 1 ]
```

IGNORE_LEF_MACRO

Specifies macros for which the LEF content is ignored (that is, the pin geometries), but still imports the DEF routing data. In the syntax below <macro1> is the DEF name for the block and <macro1_APACHECELL> is the LEF macroname for the same block's only child. *DMP compatible. Optional. Default: none.*

Syntax:

```
IGNORE_LEF_MACRO
{
    <macro1>
    <macro1_APACHECELL>
    ...
}
```

Example:

```
IGNORE_LEF_MACRO
    Cornerblock_LL
    Cornerblock_LL_APACHECELL
```

IGNORE_LIB_CHECK

Specifies whether a semantics check for unsupported attributes and groups in the .lib file is performed or not. *DMP compatible. Optional. Default: 0 (perform check).*

Syntax:

```
IGNORE_LIB_CHECK [ 0 | 1 ]
```

IGNORE_NETS

Specifies the list of nets defined in the SPECIAL_NETS or NETS sections of DEF files to be ignored during building connectivity (they are not imported into the design DB). For example, the ANALOG_VDD_NET net contributes no IR-drop to core VDD domain and can be ignored. If a DEF file path is *not* provided, nets specified are ignored in all hierarchies in the design, within the top cell and blocks. Nets specific to a block to be ignored should have the block's DEF file path provided. Also note that in Signal EM analysis mode, when a signal net is specified in IGNORE_NETS, it is also ignored for signal EM analysis. *DMP compatible. Optional. Default: none.*

Syntax:

```
IGNORE_NETS
{
    <net_name1> ? DEF_file_path ?
    ...
}
```

Example:

```
IGNORE_NETS
{
    ANALOG_VDD_NET1 /DEF/blockABC.def
}
```

IGNORE_NETS_FILES

Specifies the list of nets defined in the SPECIAL_NETS section of DEF files to be ignored during building connectivity. The file containing the list referred to must have only one excluded net per line. When used in conjunction with IGNORE_NETS, both lists are read in. *DMP compatible. Optional. Default: none.*

Syntax:

```
IGNORE_NETS_FILES <ignoreNets_file_pathName>
```

Example:

```
IGNORE_NETS_FILES /home/users/customerA/nets_exclude
```

IGNORE_PRECHECK_ERROR

Continues input data checking when a data pre-check error is found (data checking specified by the DYNAMIC_PRECHECK keyword). *Optional. Default: off.*

Syntax

```
IGNORE_PRECHECK_ERROR [ 0 | 1 ]
```

IGNORE_PRIMARY_LOAD_DECAP

When this keyword is turned on, the primary output load decaps (not on-die capacitance) are *not* considered in simulation. *Optional. Default: 0 (off).*

Syntax:

```
IGNORE_PRIMARY_LOAD_DECAP [ 0 | 1 ]
```

IGNORE_ROUTE

The IGNORE_ROUTE keyword allows skipping certain geometries when DEF is merged from GDS, so that for each block and the top (pointed to by <def_file_path_name>) the geometries of layer <layer_name> and above are ignored while importing data. *DMP compatible. Optional. Default: None.*

Syntax:

```
IGNORE_ROUTE {
    <def_file_path_name>  <layer_name>
}
```

Example:

```
IGNORE_ROUTE
{
    user_data/DEF/MBIST_INSTA6.def      V1
    user_data/DEF/MBIST_MACRO2.def      M3
    user_data/DEF/MBIST_INSTB3.def      V2
    user_data/DEF/INSTCS.def          V2
}
```

In the example, for block MBIST_INSTA6, geometries of layers V1 and above are ignored. (That is, only M1 and below are kept.) For the top, layers V2 and above are ignored. Note that by this setting, block MBIST_MACRO2 still keeps its own M2 and V2, even though its parent INSTCS's M2 and V2 are ignored.

IGNORE_SHORT

If set to true, ignores all shorts in the design. *DMP compatible. Optional. Default: 1 (on).*

Syntax:

```
IGNORE_SHORT [ 1 | 0 ]
```

IGNORE_SIMTIME_CHECK

If turned on, turns off the validation check that the specified simulation time in DYNAMIC_SIMULATION_TIME is reasonable and overrides the maximum value. *Optional. Default: 0 (off).*

Syntax:

```
IGNORE_SIMTIME_CHECK [ 1 | 0 ]
```

IGNORE_TECH_ERROR

Controls checking for dielectric layer data. When set to 1, the lack of dielectric data in the Tech file *does not cause a critical failure of the run; the run continues. DMP compatible. Optional. Default: 0 (stop for dielectric data failure)*

Syntax:

```
IGNORE_TECH_ERROR [ 0 | 1 ]
```

IGNORE_UNDEFINED_LAYER

If DEF files have layers that are not defined in the Tech file or in LEF, RedHawk exits at the end of DEF import. To proceed, either fix the Tech file layer data or set the keyword IGNORE_UNDEFINED_LAYER to true (1), which means wires/vias on the undefined layers are ignored. *DMP compatible. Optional. Default: 0 (off).*

Syntax:

```
IGNORE_UNDEFINED_LAYER [ 0 | 1 ]
```

IGNORE_UNPLACED_INSTANCE

If IGNORE_UNPLACED_INSTANCE is set, all instances with status 'UNPLACED' in DEF are ignored. *Optional. Default: 0 (off).*

Syntax:

```
IGNORE_UNPLACED_INSTANCE [ 0 | 1 ]
```

IGNORE_UPF_PGARC

When full P/G arc data is not available in the UPF, you should turn on this keyword, (default off). In this case UPF P/G arc detection is disabled, and file level P/G arcs defined in the custom LIB file are used. *DMP compatible. Optional. Default: 0 (off).*

Syntax:

```
IGNORE_UPF_PGARC [ 0 | 1 ]
```

To ignore/honor specific UPF libraries, include honor_upf_pgarc 0/1 in custom lib file. Note that this setting works with IGNORE_UPF_PGARC set to 1.

Syntax:

```
library <library_name> {  
    nomVoltage <voltage value>  
    nomTemperature <temp value>  
    honor_upf_pgarc [ 0 | 1 ]  
}
```

MISSING_VIA_CHECK_IGNORE_CELLS

Allows exclusion of specified instances in missing vias reports, rather than full GDS blocks. When this keyword is used, missing vias inside instances using the specified cellnames as masters are not reported. *Optional. Default: None.*

Syntax:

```
MISSING_VIA_CHECK_IGNORE_CELLS {  
    <cellname_1>  
    <cellname_2>  
    ...  
}
```

IGNORE_VP_CONTROL_ERROR

When set, allows RedHawk to continue when the VPC file has errors. When set and encountering VPC errors, for example when the file includes more cell types than are used in design, a warning (DSG-201) is displayed, but execution continues. *Optional. Default: Off.*

Syntax:

```
IGNORE_VP_CONTROL_ERROR [ 0 | 1 ]
```

GSR Macro Keywords

DEFINE

Defines a variable value, including a complete file path, to allow ease of changing between analysis runs, providing a convenient way of reusing and editing a file path. This keyword should be defined at the top of the GSR file.

The variable also can be defined in a TCL command file, before the 'import GSR' command, such as: 'gsr define <user_variable_name> <value>'. *Optional. Default: none.*

Syntax:

```
DEFINE <user_variable_name> <value>
```

Example

```
DEFINE AAA /my/path/design_data/lef
DEFINE BBB /my/path/design_data/lef
LEF_FILES {
    $AAA/a1.lef
    $AAA/a2.lef
    $BBB/b1.lef }
```

Or, using a command file definition:

```
run.tcl
gsr define AAA /my/path/design_data/lef
gsr define BBB /my/path/design_data/lef
```

INCLUDE

Used to include/parse contents of other GSR files as if they were included where the "INCLUDE" keyword is. *Optional. Default: none.*

Syntax:

```
INCLUDE <GSR_filename>
```

Example

If a GSR file has the contents:

```
INCLUDE <GSR_fileA>
<gsr_keyword_1> <val_1>
<gsr_keyword_2> <val_2>
...
...
```

and the <GSR_fileA> contents are:

```
<gsr_keywordA> <valueA>
```

This is the same as having the following GSR contents:

```
<gsr_keywordA> <valueA>
<gsr_keyword_1> <val_1>
<gsr_keyword_2> <val_2>
...
...
```

Pad, Power/Ground and I/O Definition Files

Unified Pad Input File Format

A single input file now can be used to specify all pad, power/ground and I/O input data previously provided in separate *.pcell, *.pad or *.ploc files. Separate sections of the file are identified for each of the three types of information with an '*' and a unique keyword, followed by standard data lines for that type of file, as follows:

```
*PLOC
<.ploc file data>
...
*PCELL
<.pcell file data>
```

```
...
*PAD
<.pad file data>
...
```

The unified pad input file can be imported into RedHawk using the ‘import pad’ command:

```
import pad unified_pad.txt
```

or using the GSR PAD_FILES keyword:

```
PAD_FILES {
    unified_pad.txt
}
```

Two other file formats, *ploc_pss* and *pad_pss*, can also be accepted in the unified file, but their data cannot be mixed with the other three types in the same file:

```
*PLOC_PSS
<pin_name> <x_loc> <y_loc> <layer> [POWER|GROUND]
    <PAD_R_ohm> <PAD_L_pH> <PAD_C_pF>

*PAD_PSS
<pad_inst name> <inst_pin name> <PAD_R_ohm>
    <PAD_L_pH> <PAD_C_pF> <pkg_node>
```

The PLOC_PSS keyword is used for the *.ploc* format with a package Spice subcircuit netlist. This is different from the typical *.ploc* format, where the last column has the SPICE node name that hooks up to the individual *.ploc* node.

The PAD_PSS keyword is used with the *.pad* format with a package Spice subcircuit netlist.

The following is an example of a unified file that includes all currently-supported *.ploc*, *.pcell*, and *.pad* input formats.

```
## Format for .ploc files:
*PLOC
# For original .ploc format with 5 items
# <pin_name> <x_loc> <y_loc> <layer> [POWER|GROUND]
    Vdd1      60.9     1214.67   M2 POWER
    Vss1      69.6     1214.67   M2 GROUND

# For original .ploc format with P/G_net name specified, 5 items
# <pin_name> <x_loc> <y_loc> <layer> <PG_net name>
    Vdd1      60.9     1214.67   M4 VDD1
    Vdd2      69.6     1214.67   M4 VDD2
    Vss       80       1214.67   M2 VSS

# For block pin voltage flow runs with 6 items
# <pin_name> <x_loc> <y_loc> <layer> [POWER|GROUND] <pin voltage>
    TVDD:0   2292.000 5110.110      M4      POWER  1.499
    TVDD:1   2292.000 5110.110      M2      POWER  1.499
    TVDD:2   2292.000 5110.210      M6      POWER  1.497
    TVSS:0   2292.000 10.210       M6      GROUND 0.045
```

```

# For specifying bump L, R, C data - 8 items
# <pin_name> <x_loc> <y_loc> <layer> <power|ground> <L_pH> <R_Ohms> <C_pF>
    Vdd_1    15.0    25.0    metal6   power   l=10      r=0.001 c=10
    Vss_1    1050.0  23.0    metal6   ground  l=20      r=0.005 c=5

## Format for .pcell files:
*PCELL

# Original .pcell file with 1 item
# <pad_cell macro name>
    T1VDDCOREA
    T1VSSA

# Formats that specify which pins are used in a pad macro - 2 items
# <pad_cell macro name> <macro pin name>
    VDD_PAD_MASTER vdd
    VDD_PAD_MASTER gnd
    VSS_PAD_MASTER vdd
    VSS_PAD_MASTER gnd

# Formats that specify connection locations and layers within
# a pad macro: 1-2 line format
# 1st : <pad_cell macro name>
# <x_macro_loc> <y_macro_loc> <layer> [ Power|Ground ]
    PVDD1DGZ
    18.0 2.4 METAL5 ground
    17.5 242.0 METAL3
    PVSS1DGZ
    18.0 3.4 METAL5 POWER
    17.5 242.0 METAL3

## Format for .pad files:
*PAD

# Original .pad file format - 1 item
# <pad_instance name>
    PAD_INST1
    PAD_INST2

# To specify which pins are used in a pad instance - 2 items
# <pad_inst name> <inst_pin name>
    VDD_PAD_INST1 vdd
    VDD_PAD_INST1 gnd
    VDD_PAD_INST8 vdd
    VDD_PAD_INST5 gnd

# To specify connection loc and layers for a pad instance: 1-2 lines
# 1st: <pad_inst_name> <x_macro_loc> <y_macro_loc> <layer> [ PWR|GRND]
# 2nd: <x_macro_loc> <y_macro_loc> <layer> [ PWR|GRND], if defined
# for the same pad instance name
    pvddl_io 17.50 -4.0 METAL6 POWER

```

```

18.0 -4.0 METAL6
pvss2_io 17.50 -4.0 METAL6 GROUND

## Format for .ploc files with a Package Spice Subckt (PSS)
# definition, in a separate input file
*PLOC_PSS
# Package-Spice-Subckt(PSS) with 6 items
# <pin_name> <x_loc> <y_loc> <layer> <PG_net name> <subckt node name>
Vdd1      60.9    1214.67 M4 VDD  n1
Vss1      69.6    1214.67 M4 VSS  n2
-----
```

To set up your package parameters so that you can easily look at the effects of different package designs on power integrity, after extraction use the TCL command ‘setup pss’, which defines PLOC and package subcircuit files to be used in simulation, using the follow syntax:

```
setup pss -pad_file <filename> -subckt <pss_ckt_name>
```

To import pad data in a file, use the **RedHawk** command:

```
import pad <filename>
```

Notes on using the unified pad file:

1. The same format keyword may appear multiple times in a file (but grouping entries with the same format type is recommended to facilitate debugging).
2. Empty lines are ignored.
3. All existing *.ploc/.pcell/.pad* files are accepted and parsed as before.
4. For package hookup to **RedHawk**, if there are dangling nets in the package netlist, both ends need to be tied to ground through a high R, such as 10 MOhms.
5. Any file name (even extensions *.ploc/.pcell/.pad*) is allowed for unified pad input, as long as one or more of the four keywords and acceptable data are found inside the file. Otherwise, **RedHawk** displays an error during pre-parsing if it finds no *.ploc/.pcell/.pad* files and no files with the keywords *PLOC, *PCELL, or *PAD.
6. Use of PSS format excludes other formats in the same file. That is, whenever a PSS is specified with a GSR PACKAGE_SPICE_SUBCKT_INFO keyword, other pad input formats must be in a different I/O file and use a separate import command.
7. Format for specifying a pad_cell macro or a pad_inst location within its bounding box is as follows: with all data on one line:

```
pvdd_master 17.50 -4.0 METAL6
```

However, if additional connection locations are required for a pad_cell or pad_inst, its name is required on the first line only, and the location and layer information can be put on a second line.

Individual Pad File Specification

For compatibility, individual **.pcell*, **.pad* or **.ploc* files are still acceptable. If you provide the *.pad* or *.pcell* file, **RedHawk** automatically identifies the VDD/GND pad locations by determining each pad location from the *.def* file. However, if you provide the *.ploc* file, **RedHawk** can directly read in the pad locations from the file. You can also manually add power and ground pads by using the **Edit> Add Pad** command.

Note that you can use **Edit -> Undo/Redo** on **Add/Delete Pad** operations, and multiple ‘**import pad *.pad/*.pcell**’ commands are allowed.

Pad Cellname File (*.pcell)

A. The following basic syntax can be used to describe pad cells and the associated pins in a *<design>.pcell* file:

```
<pad_cell_name_1> ?[<pin_name>| <pin_name> <layer_name>]?  

...  

<pad_cell_name_n> ?[<pin_name>| <pin_name> <layer_name>]?
```

Example

```
PADLZ55 VDD  

PADLZ55 VSS
```

Note that if only the pad cellname is given in the above syntax, the P/G pins of each instance of cell are checked against the DEF routing. Anywhere that P/G routing touches a pin of the correct type a connection to the P/G source net is made.

B. You can specify the exact location of voltage sources within the cells. This improves current modeling accuracy, especially if the pad cells are large.

The alternative format for a *.pcell* pad file is as follows:

```
<master cell name>  

<x source loc> <y source loc> <layer> <P/G pad type>  

...  

where
```

<master cell name>: specifies the master cell name (LEF macro name) indicating a pad cell (must be on a line by itself)

<x source loc> <y source loc> : x,y location of the master cell (referred to the macro's own coordinate system)

<layer> : source layer name

<P/G pad type> : the source P/G pad type

Example

```
PVDD1DGZ  

17.5 242.0 METAL6 POWER  

PVSS1DGZ  

17.5 -4.0 METAL6 GROUND
```

There may be more than one set of location, layer, and type entries following a master cell name. These all imply sources specific to this pad macro cell.

For each instantiation of a master cell, its source locations are transformed with respect to the instance location. A file called *adsRpt/<design_name>.NEW.ploc* reports all such transformed sources from all instantiations.

Pad Instance Name File (*.pad)

A. The following syntax can be used to describe pad cell instances (not macro names) and the pins used in a *<design>.pad* file.

```
<pad_instance_name> ?[<pin_name>| <pin_name> <layer_name>]?  

...  

<pad_instance_name> ? [<pin_name>| <pin_name> <layer_name>]?
```

The following is an example of a list of pad instance name specifications (from the DEF file) in a *<design>.pad* file:

```
padring_1/ring/pad_gndc120  

padring_1/ring/pad_gndc120a  

padring_1/ring/pad_gndc120b  

padring_1/ring/pad_vddc115
```

```
padring_1/ring/pad_vddc115a
padring_1/ring/pad_vddc115b
```

B. A *.pad* file also can use format ‘B’ as described in the previous **.pcell* file section, by replacing the master cell name with the pad instance name. A file *adsRpt/<design_name>.NEW.ploc* is also generated.

Pad Location File (*.ploc)

If VDD/GND pad locations are already determined, they can be specified in a *<design>.ploc* file, which is formatted as follows:

<i><pad_name></i>	<i><X></i>	<i><Y></i>	<i><metal name-DEF></i>	[POWER GROUND]	<i><L-pH></i>	<i><R-Ohms></i>	<i><C-pF></i>
VSS1	6484.0	3761.0	metal4	GROUND	l=10	R=0.001	C=2
VSS2	6484.0	5141.0	metal4	GROUND	l=10	R=0.001	C=2
VDD1	320.0	1599.0	metal6	POWER	l=20	R=0.003	C=5
VDD2	6484.0	4001.0	metal6	POWER	l=20	R=0.003	C=5
VDD3	6484.0	3401.0	metal6	POWER	l=20	R=0.003	C=10
VSS3	6484.0	7961.0	metal4	GROUND	l=10	R=0.001	C=2
VSS4	320.0	3039.0	metal4	GROUND	l=10	R=0.001	C=2
VDD4	320.0	3159.0	metal6	POWER	l=20	R=0.003	C=10
VSS10	320.0	1299.0	metal4	GROUND	l=10	R=0.001	C=2

where *<X>* and *<Y>* are x,y locations, and L, R, and C are specified pin/bump values, instead of general values from the ‘setup wirebond’ command. Unspecified pin/bump L/R/C values are taken from the ‘setup wirebond’ command values.

To facilitate PSS node hook-up with the PSS *.ploc* file, RedHawk creates a new file, *adsRpt/PG_simple.ploc*, in which each pad instance has just one ploc specified (instead of multiple plocs in *PG.ploc* file). You should manually change this *PG_simple.ploc* file, adding the PSS subckt nodes, and then re-run RedHawk with this modified *ploc* file. Note that the GSR keyword PRINT_ONE_PLOC_PER_PADINST must be set to 1 in order to create the special *adsRpt/PG_simple.ploc* file. RLC data can be displayed for each bump in the *adsRpt/PG_simple.ploc* file if the GSR keyword PGPLOC_DEBUG 1 is also set. A sample file, with RLC data and domain names, is shown below:

```
VDD10xyz 0.5 28.188 METAL5 VDD l=10000 r=0 c=0
VDD10xyz2 49.5 21.720 METAL5 VDD l=10000 r=0 c=0
VSSxyz 49.5 31.225 METAL5 VSS l=10000 r=0 c=0
```

Note that when using the DEF_SCALING_FACTOR and LEF_SCALING_FACTOR GSR keywords, you do not need to scale ploc (x,y) values manually. Use the original unscaled PLOC file and RedHawk automatically places the pads in the correct locations in the scaled design.

Dual RLC specifications in PLOC

When RLC values are specified in the PLOC file as well as in the ‘setup wirebond’ command, the PLOC file values override the ‘setup wirebond’ values, which is indicated in the log file. Following is an example PLOC file showing R values for each bump:

```
VDDC_1 561.40 3.64 M2 POWER R=100
VDDC_2 561.40 3.54 M2 POWER R=110
VSSC_1 561.40 1.40 M2 GROUND R=100
```

Following is an example of the ‘setup wirebond’ command:

```
setup pad -power -r 30 -ground -r 30
setup package -power -r 70 -ground -r 70
setup wirebond -power -r 50 -ground -r 50
```

For the above case, the log file contains the following messages:

```
** Pad/WireBond/Package RLC Setting Summary **
Pad Vdd Resistance = 30 ohm
Pad Gnd Resistance = 30 ohm
Wirebond Vdd Resistance = 50 ohm
Wirebond Gnd Resistance = 50 ohm
*PIN<VDDC_1> will use the following R, I, C values.
    Vdd Resistance = 100 ohm. Vdd Induct = 0 pH. Vdd Cap = 0 pF
*PIN<VDDC_2> will use the following R, I, C values.
    Vdd Resistance = 110 ohm. Vdd Induct = 0 pH. Vdd Cap = 0 pF
*PIN<VSSC_1> will use the following R, I, C values.
    Gnd Resistance = 100 ohm. Gnd Induct = 0 pH. Gnd Cap = 0 pF
Package Vdd Resistance = 70 ohms
Package Gnd Resistance = 70 ohms
```

When “-ignore” option is specified with TCL command “setup wirebond”, the RLC parameter defined in the ploc file is ignored. So the order of priority is as follows:

- setup wirebond -ignore
- RCL parameter defined in the ploc file
- setup wirebond -r -c -l

Pad PSS File

Hooking up distributed package SPICE subcircuit netlist nodes to the corresponding nodes on the chip can be complicated, especially if the x,y locations on the chip are unknown. In such cases, RedHawk can derive x,y locations from the .pcel file to use with the distributed package SPICE subcircuit netlist.

The unified pad file format described above can also include an enhanced format of pad cells (if no other types of pad definitions are included in the file). By listing the power/ground pin name inside the pad cell and the corresponding package subcircuit port name, the voltage sources can be hooked up appropriately. The keyword required within the unified pad file to recognize this format is *PAD_PSS. RedHawk automatically forms the connection without the requirement of XY location input. An example is shown below:

```
*PAD_PSS
#<pad inst name> <P/G pin name> <package subcircuit port name>
PVDDSSTT_1 VDDC POWER1
PVSSDDRR_4 VSS GROUND4
```

The above file can have any extension and must be instantiated within the PAD_FILES {} section of GSR.

As described above, you can use the GSR keyword

```
PRINT_ONE_PLOC_PER_PADINST 1
```

along with the .pcel file to create a simplified .ploc file that contains one ploc node for every instantiation of pad cell defined in .pcel file. RedHawk creates a *adsRpt/PG_simple.ploc* file in which each pad instance has just one ploc specified. This facilitates package SPICE sub-circuit node hook-up by manually changing the file and rerunning RedHawk.

Library Technology Files

The Library Exchange Format, or .lef file, defines the IC process technology and the associated library of cell models. Each .lef file must be specified on a separate line in the LEF_FILES GSR keyword with its absolute or relative path from the RedHawk run

directory. The first `.lef` file must contain technology layer and via information. The following is an example of a set of LEF files:

```
LEF(blocks.lef
LEF/abcd_13hd.lef
LEF/abcd_13hd_10_2a.lef
```

Design Netlist (DEF) Files

The Design Exchange Format, or `*.def` file, defines the elements of an IC design relevant to the physical layout, including the netlist and design constraints. Each `.def` file must be specified in the GSR keyword `DEF_FILES` on a separate line with its absolute or relative path from the **RedHawk** run directory, with the following format:

```
<path>/<blockname>.def block/top
...
```

where

`block/top`: indicates whether the block is a top-level block or sub-block of the top level block. The top-level block must be the last entry.

The following is an example of a set of DEF files:

```
DEF/abcd_13hd.def block
DEF/abcd_13hd_10_2a.def block
DEF/top_block.def top
```

Synopsys Library Files

The Synopsys Libraries files define the directory path for the set of Synopsys Library files (`.lib`), which are used for cell definition and power calculation.

LIB files and custom LIB files are specified using the APL configuration file keywords ‘`LIB_FILES`’, and ‘`CUSTOM_LIBS_FILE`’ and the GSR keyword ‘`LIB_FILES`’.

Custom LIB File Syntax

The following section shows the syntax and several examples of using a custom LIB file (no more than one may be used), which is used for overriding other LIB file data and also for providing supplementary information not included in a standard LIB file, such as definitions of P/G arcs in MVdd designs. The syntax for specifying custom library attributes in a file is as follows (all entries are optional):

```
pgarc {
    <vdd_pin_name> <gnd_pin_name>
    ...
}

pin <pin_name>
{
    capacitance <pF>
    direction [ input | output | inout ]
    function "<function_type>"
    type [ clock | scan ]
    vector "<inputPinName> <clockPinName> : <pinName> [0 | 1] ..."
}

cell <cell_name>
```

```

{
    type [ comb | latch | ff | seq | memory ]
    subtype [input_pad | output_pad | inout_pad | clockgating | mux ]
    pin <pin_name>
    {
        capacitance <pF>
        direction [ input | output | inout ]
        function "<function_type>"
        type [ clock | scan ]
        vector "<inputPinName> <clockPinName> : <pinName> [0 | 1] ..."
    }
}

library <library_name> {
    nomVoltage <volts>
    nomTemperature <temp>
    pin <pin_name> {
        ...
    }
}

```

Example: In library *.lib*, pins named “CK” are of type ‘clock’:

```

library lib
{
    pin CK
    {
        type clock
    }
}

```

Syntax for specifying cell attributes:

```

cell <cell_name>
{
    type <comb | latch | ff | seq | memory>
    subtype <input_pat | output_pad | inout_pad | clockgating | mux>
    pin <pin_name>
    {
        capacitance <pF> |
        direction <input | output | inout> |
        function <function_type> |
        type <clock | scan> |
        vector "<inputPinName> <clockPinName> : <pinName> [0 | 1]"
    }
}

```

Example: In cell “A”, a pin named “CK” is of type ‘clock’:

```

cell A
{
    pin CK
    {

```

```
        type clock
    }
}
```

Example: Define vector for function “A B” of pin C:

```
cell CEL
{
    pin C
    {
        vector "A NA : B 0"
    }
}
```

Syntax for specifying pin attributes:

```
pin <pin_name>
{
    capacitance <pF> |
    direction <input | output | inout> |
    function <function_type> |
    type <clock | scan> |
    vector "<inputPinName> <clockPinName> : <pinName> [0 | 1]"
}
```

Example: Pins named “CK” are of type ‘clock’:

```
pin CK
{
    type clock
}
```

Example: specifying P/G arcs in multi-Vdd/Vss designs.

```
pgarc {           <- this statement covers all cells
    VDD VSS
    VDDL VSSA
}
library LibABC {
    cell ram_mvdd { <- this statement is cell-specific
        pgarc {
            VDDPR VSS
            VDDAR VSS
            VDDNW VSS
        }
    }
}
```

Timing Data File

STA Compact Format Timing File

This ASCII file contains the following sections, in the order shown:

- Header
- Clocks used in the design
- Time Scale
- Name Map
- Instance Data
- Footer

The comment character is '#'.

Details on these sections are described in the following paragraphs

Header

The header section contains basic information such as file type, design name, and pin delimiter, with the following format:

```
##### Apache Design Solutions #####
# File Type : Timing Information
# Design Name : <design_name>
...
#####
# bus_naming_style <bus_open_close_chars>
# hierarchy_separator <separator_char>
# pin_delimiter <delimiter_char>
```

Clocks

The Clocks section lists each clock used in the design, in the following format:

```
CLOCK <rise> <fall> <period> <root> <idx>
```

where <idx> is an integer that starts at 0.

For a clock that has no waveform or period:

```
#CLOCK <clk_name> INVALID_PERIOD_OR_WAVEFORM
```

Time Scale

This section specifies the time scale used in the file:

```
#TIMESCALE 1e<digits>
```

For example:

```
#TIMESCALE 1e-9
```

which indicates that the time values in the file are in nanoseconds.

Name Map

This section contains a map of the hierarchy strings of leaf instances, in order to prevent repetition of long strings later in the file, as follows:

```
#NAMEMAP (total_entries)
<id> <instance_hierarchy>
...
#END NAMEMAP
```

where <id> is an integer starting from 1, and <instance_hierarchy> is the hierarchy of parent of leaf instance.

For example:

```
#NAMEMAP (2124)
1 hh_bscan
```

```

2 hh_bscan/bsc_nINTR_Out
3 hh_bscan/bsc_HIF2_nOutEn
4 hh_bscan/bsc_SYNC_Out
5 hh_bscan/bsc_HIF2_In
6 hh_bscan/bsc_HIF0_Out
7 hh_bscan/bsc_HIF1_I2C_In
...
2123 gl_hh_pads/gl_hh_pads
2124 hh_bscan/bsc_HIF0_CMOS_In
#END NAMEMAP

```

Instance Data

This section provides data on instances in the design, as follows:

```

#INSTANCE (total_instances)
I $<id>/<instance_name>
S <pin_name> <min_r_slew> <max_r_slew> <min_f_slew> <max_f_slew>
S ...
L <pin_name> <C1> <R> <C2> <type> <Cpin>
C <pin_name> <1|0>
...
T <pin_name> <is_clock> <min_r> <max_r> <min_f> <max_f> <clk_idx>
T ...
D <pin_name> <is_clock> <min_r> <max_r> <min_f> <max_f> <clk_idx>
...
#END INSTANCE

```

where

I : identifies the instance by using its name map id. Note that top level instances do not have hierarchy, and are not prefixed with \$<id>/.

S : specifies the slew information for an instance pin

L : represents signal load in the form of C1, R, C2 for an instance pin

<type> is an integer that describes the signal load, with these values:

0 : signal load is unknown

1 : signal load is based on SPEF file

2 : signal load is based on placement-based route estimation

3 : signal load is based on wireload model

4 : signal load is given by user

Cpin : the total pin capacitance included in the calculated signal load C1, R, C2

C : represents CONST instance pins

T : specifies the timing window information for an instance pin

D : specifies the timing window information for an input port

<is_clock= 1 or 0> : indicates whether the pin is the output of an instance along the clock tree or not.

<clk_idx> : specifies the index of the associated clock listed in the CLOCKS section

For example:

```

#INSTANCE (177763)
I rfClk_S5

```

```
S A 0.023 0.023 0.022 0.022
S Z 0.023 0.023 0.022 0.022
L Z 1.5e-15 452 4.5e-15 1 0.5e-15
T A 0 7.564 7.589 7.696 7.724 17
T Z 1 7.678 7.703 7.810 7.838 17
...
I $2124/BH_BUF30
S A 0.024 0.024 0.018 0.018
S Z 0.009 0.009 0.007 0.007
T A 0 4.565 4.565 4.576 4.576 31
T Z 0 4.643 4.643 4.655 4.655 31
#END INSTANCE
```

Footer

This section contains comments that summarize statistics about the file content, including the following items:

- Number of unique clock frequencies
- Various histograms on clocks and pin slew data
- Total processed pins
- Pins without timing windows
- Constant pins

Legacy Format Timing File

This ASCII file occupies 3-4 times more disk space than STA-compact format due to repetition of long strings. The information stored in this format is the same as in the STA-compact format, so it is recommended that the STA-compact format be used, which also provides a runtime benefit when RedHawk is reading the file, as the file I/O overhead is much less.

The legacy format contains the following sections, in the order shown:

- Header
- Clocks used in the design
- Signal load seen by driver pins
- Slew of instance pins
- Timing windows of instance pins
- Footer

The comment character is '#', and the time values are in seconds.

The content for header, clocks and footer sections are same as in the STA-compact format.

The other sections are described below.

Signal Load

This section lists the signal load for each driver pin, in the following format:

```
<pin_name> L <C1> <R> <C2> <type> <Cpin>
```

Slews

This section lists “slew” information (transition times, in seconds) for each instance pin, in the following format:

```
<pin_name> SL <min_r_slew> <max_r_slew> <min_f_slew> <max_f_slew>
```

Timing Windows

This section lists timing windows for instance pins, and also records pins that are constant, dangling, without a valid clock, or with no timing windows.

For pins with timing windows:

```
<pin_name> TW <is_clock> <min_r> <max_r> <min_f> <max_f> <clk_idx>
```

For input ports with timing windows:

```
<pin_name> IDEL <is_clock> <min_r> <max_r> <min_f> <max_f> <clk_idx>
```

For constant pins:

```
<pin_name> CONST [1|0]
```

For dangling pins:

```
#<pin_name> DANGLING
```

For pins with no timing window (except dangling pins):

```
#<pin_name> NO_TW {}
```

For pins that have no valid clock:

```
#<pin_name> NO_VALID_CLOCK
```

Result Files

The **RedHawk** log and result files for power, EM, and IR / dynamic voltage drop are written to the *adsPower* and *adsRpt* directories. Refer to [Chapter 6, "Reports"](#), for more details on files created during **RedHawk** analysis.

Appendix D

Command and GUI Reference

Introduction

This appendix describes the two user interfaces available for running **RedHawk**:

- TCL command line interface
- Graphical User Interface (GUI)

Invoking RedHawk

Invoke **RedHawk** from a UNIX command line in your working directory using the syntax:

```
redhawk [ -b <cmnd_file> ] [ -c <cmnd_file> ] [ -f <cmnd_file> ]
[ -i ?<cmnd_file>? ] [ -h ] [ -tclsh ] [ -lmhold <cmdfile> ]
[ -lmwait <wait_sec> ] [ -iconify ]
[ -style [ CDE | Motif | Plastique | Windows ] ]
[ -stack [ <stacksize_MB> | unlimited ] ]
[ clampviewer <IV_FileName> [<IV_Name(s)>] [<options>]
```

where

- b <cmnd_file> : specifies a TCL script batch command file to run, no GUI displayed. After the last command, **RedHawk** exits.
- c <cmnd_file>: specifies a TCL script command file, and performs pre-parsing checks in the command file for both built-in and **RedHawk**-specific TCL commands, without actually executing each command, and reports warnings or errors accordingly.
- f <cmnd_file> : specifies a TCL script command file to run, and GUI displayed
- i : provides a TCL command line for **RedHawk** command input, with no GUI display. After the last command, **RedHawk** only exits with a specific exit command.
- h : displays the **RedHawk** invocation syntax on screen
- tclsh : opens a full capability TCL shell and checks out a license based on your **RedHawk** package and the options specified, but does not start **RedHawk** until a 'redhawk' command is invoked. Allows rerunning/restarting an analysis run in the same session without releasing the license. When the run is exited, the license is held and the shell redisplayed. When only 'redhawk' is invoked, the GUI is displayed by default. You can hold a particular license using the 'license get' command:
 license get redhawk_vcd
- lmhold : opens a full capability TCL shell, gets a **RedHawk** license based on your **RedHawk** package and the options specified, and also starts a **RedHawk** process. When the run is exited, the license is held and the shell redisplayed.
- lmwait : allows a process to wait until a license is available, so that it does not exit due to lack of a license during the initialization process. With no time limit specified, the process is put into "sleep" mode indefinitely. When a wait limit is specified (seconds), if no license is obtained at the end of the specified time, **RedHawk** quits.
- style: allows you to change the GUI to a different window manager style (default: plastique)
- iconify : iconifies the **RedHawk** GUI at startup, which can be displayed by clicking on the icon. Because the GUI is created and iconified, you still must have a DISPLAY (be able to use 'setenv DISPLAY') for the option to work, and the environment must be able to run the **Redhawk** GUI. Also, the option has no effect if the '-b' option is used, which has no GUI display.

-stack [<stacksize_MB> | unlimited] : specifies the stack size in MB, or ‘unlimited’ (limited by the physical stack size available - default)

Terminating Processes

The keystroke combination ‘**Ctrl + c**’ allows you to terminate the current operation and most associated child processes, which can be useful if a **RedHawk** command is terminated abnormally, or is running too long, but the main process is not terminated. Note that pressing **Ctrl + c** on an X-term interrupts the main **RedHawk** process and essentially kills it. **Ctrl + c** in the GUI is not honored by a few processes, such as ‘setup design’ and extraction.

TCL / Script Commands

The TCL commands in **RedHawk** enable you to run a complete session and save the results for subsequent viewing and debugging in the GUI. The TCL script file can be created either manually or by executing the desired commands and using the **Playback** menu command. The TCL script command file serves as a replay file in the GUI.

TCL Syntax Conventions

The syntax conventions used for defining **RedHawk** commands, options and keywords are as follows:

<x>	x describes a variable or value to be specified
[a b c]	one of a, b, or c must be selected
? x ?	x is an optional parameter
a b c d	a, b, c, and d parameters all must be specified
<x> ...	elements of the same type as <x> can be added
{ {a b c ...} {j k l ...} ... }	similar sets of elements may be added. Note that if only one set of elements is included, two sets of brackets may still be required
+ - * /	standard arithmetic operators for add, subtract, multiply, divide

TCL Command Summary

This section describes TCL commands for running **RedHawk** from the command line, along with a summary of their syntax and a brief description of each option. To see the full syntax for any command, use the ‘help’ command on the TCL command line.

Auto-completion of TCL commands is supported in **RedHawk** GUI followed by next level suggestions. For example, to perform extraction just type pe followed by “tab” and it will autocomplete perform and tab again will show first level option which is either [analysis | extraction | pwrcalc etc.

Note: Pressing the TAB key returns the cursor to the TCL command line.

help

help

The ‘help’ command lists available **RedHawk** TCL commands:

>> Click on the command name in blue to jump to the command description.

cell swap	characterize	condition	config	decap
dump	eco	export	fao	generate
get	gsr	history	import	license get
marker	mesh	message	movie	perform
pfs	plot	print	probe	query
report	ring	route fix	save	select
setup	show	window	zoom	

The syntax and options available for any command or command group can be obtained by using the following invocation on the command line:

```
help <TCL_command_name>
```

The available RedHawk TCL commands are summarized in the following section.

cell swap

cell swap <type_of_cells> <cell_name_list>

The ‘cell swap’ command allows high power cells to be swapped with lower power cells to improve performance in areas with a high voltage drop.

See [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#), for details about using this command.

characterize

characterize apl ?<options> ?<configFileName>

The ‘characterize apl’ command runs Apache Power Library (APL) characterization to generate dynamic Vdd current waveforms and characterize decap cells in the design.

a. ‘characterize apl’ cell characterization

```
characterize apl ? -l <cellListFileName>?  
?-o <outputFileName>? <configFileName>
```

where

- l <cellistFileName> : specifies the file containing cells to characterize.
- o <outputFileName> : specifies the results file, with default <cell>.current.
- <confFileName> : specifies the configuration file

b. ‘characterize apl’ decap characterization

```
characterize apl ?[-c | -p <decapFileName>]? ?-l <cellistFileName>?  
?-o <outputFileName>? <configFileName>
```

where

- c : performs intrinsic / intentional decap characterization
- p <decapFileName> : specifies the file name containing intrinsic / intentional decap for characterization.
- l <cellistFileName> : specifies the cell list file
- o <outputFileName> : specifies the results file, with default <cell>.cdev.

condition

condition [set | get | unset] ?<options> ?

The ‘condition’ command sets, displays and unsets various conditions for post-DvD analysis (such as plot, print).

a. ‘condition set’ sets design parameters for post-DvD analysis.

```
condition set ?-xy <x1 y1 x2 y2>? ?-time <start end>?
?-type [ all |combinational| sequential| clock| memory]?
?-timestep <timestep>?
```

where

- xy <x1 y1 x2 y2> : specifies the lower left and upper right x,y corner coordinates for a rectangular region (microns).
 Note that if you specify ‘condition set -xy’, then a ‘Zoom In/Out’ command does *not change* the region, but if ‘condition set -xy’ is *not* specified, then a ‘Zoom In/Out’ command sets the region covered.
- time <start> <end> : specifies the simulation start and end times, defined as a sub-range of the entire simulation (ps)
- type [all |combinational |sequential| clock| memory] : selects the instance type to be used.
- timestep <timestep> : specifies the timestep in picoseconds and to set the Bin size of histograms. By default, 100 bins are used, so the bin size (or time-step) is thus the total simulation time divided by 100.

- b. ‘condition get’ examines conditions set previously using ‘condition set’.

```
condition get [-xy | -time | -type | -timestep ]
```

where

- xy: displays the selected rectangular region.
- time: displays the selected simulation time range,
- type: displays the selected instance type.
- timestep : displays the selected timestep

- c. ‘condition unset’ turns off design parameters set previously for post-DvD analysis.

```
condition unset ?-xy? ?-time? ?-type? ?-timestep? ?-all?
```

where

- xy : unsets the selected region.
- time :unsets the selected simulation time range.
- type : unsets the selected instance type.
- timestep :unsets the selected timestep.
- all : unsets all set conditions.

config

```
config [ colormap | keybind | stack | viewlayer | viewnet | viewpad | viewlegend ]
? <options> ?
```

The ‘config’ command sets GUI options for reviewing, analyzing and debugging of **RedHawk** results. The individual commands are described below.

- a. ‘config colormap’ specifies parameters for colormap displays

```
config colormap -map <colormap ID> -min <value> -max <value>
?[ -wire | -instance ]? ?[ -percent | -absolute]?
?-enable all ? ? -values {<list>} ? -disable {<list>} ?
?-gridsize {x y} ? ? -gridnum {x y} ? ? -refvdd <value> ?
?-refmax <reference max C or P value> ? ? -gndref [off | on]?
```

where

- map <colormap ID> : species any type of colormap. For a list of frequently-used colormap IDs, see the ‘dump gif’ command, [page D-761](#).
- min <value> -max <value> : selects the minimum and maximum values to be displayed, in whatever units are used in the type of colormap selected

-wire | -instance : for IR voltage colormaps only, selects map display for wires or instances
 -percent | -absolute : for IR voltage colormaps only, selects whether colors are displayed for absolute or percentage voltage drop values, relative to the selected reference VDD value
 -enable all : enables all color range settings
 -values {<list>}: allows specification of a list of color range values, either percentage or absolute voltage values
 -disable {<list>} – specifies the order of color range values to be disabled from the color map .
 Example:
 config colormap -map EM -values {0 0 0 75 80 85 90 95 100} -disable {0 1 2 3 }
 Displays EM map with color range greater than or equal to 80, disabling the first 4 values specified in “-values” option.
 -gridsize {x y} : Specifies grid size on command line
 -gridnum {x y} : Specifies grid num on command line.
 -refvdd <value> - specifies the reference VDD value from which to scale the color range settings
 -refmax : specifies the reference maximum value in the colormap display dialog, such as capacitance (for DD or IDD maps) or power (for PD or IPM maps). This allows changing the value scaling and the color display for the selected parameter.
 -gndref :

- b. The ‘config keybind’ command allows binding defined TCL procedures to key sequences and displaying defined keybindings.

config keybind <key_sequence> <TCL_procedure_name_or_definition>

Examples:

config keybind : displays all existing keybindings
 config keybind Ctrl+A : displays the TCL procedure of the “Ctrl+A” keybinding
 config keybind Ctrl+A export dbA dbB : defines the “Ctrl+A” keybinding, and replaces the previous one, if it existed

- c. ‘config stack(size)’ allows resetting stack size on the fly during a **RedHawk** run

config stack <size_in_MB>

where

<size_in_MB>: specifies the memory available

- d. ‘config viewlayer’ configures parameters for viewing layers.

config viewlayer -name [all | viaonly | metalonly | <layerName>]
 -style [invisible | fill | outline]

where

-name : selects a layers view, either via only, metal only, or a specified layer name. Default - all.

-style : selects the style for viewing layers. Defaults - via = invisible, metal = fill, instance = outline.

- e. ‘config viewnet’ controls display of all nets, including internal nets generally not of design interest. Using multiple commands, all nets can be turned off or on, and then specific nets can be turned on or off by net name as desired.

config viewnet -name [all | <netName>] -mode [off | on]

where

-name : selects one of the nets to view, either all or a specified net. Default - all.

-mode : selects viewing nets or not. Default - on.

- f. ‘config viewpad’ configures the parameters for viewing power/ground pads.

```
config viewpad -type [all | power | ground] -mode [on | off]
where
```

- type : selects one of the pad types to view, or all . Default - all.
- mode : selects display of pads or not. Default - on.

- g. ‘config viewlegend -fontsize’ allows you to change the legend font size between 7 and 32 in GIF images (default size is 10):

```
config viewlegend -fontsize <size>
```

decap

decap [advise | place | fill | remove | fix | report] ?<options>?

The ‘decap’ commands perform various types of modifications to decoupling capacitance to reduce dynamic voltage drop.

See [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#), for details about using these commands.

dump

dump [gif | fao | macro_pin_voltage | memory_via | mmx_pin_info | node_count | ptvoltage | res_network | sp_constraints | sptemplate | tsv_via | via_count | wire_count] ?<options> ?

- a. The ‘dump gif’ command writes out colormaps in GIF format. You may set the color mapping scale by using ‘config viewlayer’ and ‘config viewnet’ commands, or by using the ‘import guiconf’ command to load a previously saved GUI configuration file for a particular color map. Otherwise, it uses the system default settings. The command generally uses the same color map names as are on the GUI view and menu functions. This command is supported in both GUI mode and Batch mode. The GIF graphic saved is exactly as shown on the display, with the directory path at the bottom and key parameters about the display conditions on the right side, as shown in Figure D-1.

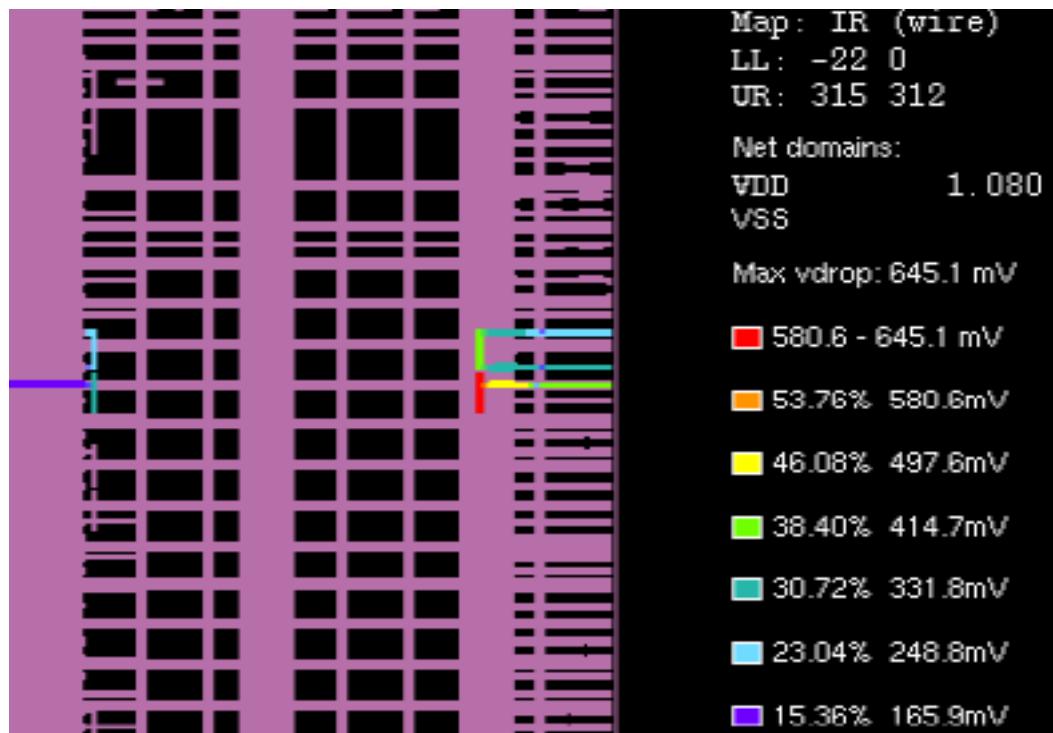


Figure D-1 Example dump of GIF file of RedHawk colormap

The command is:

```
dump gif -map <colorMapID> ?-o <outputFileName>? ?-nolegend?
```

where

-map <colorMapID> : specifies a color map to save to gif (and changes the view to that specified, if not presently displayed). Note that RedHawk makes no changes to the layer or map configuration settings. Desired changes to map configuration settings ('config colormap' and 'config viewlayer') should be made before executing the 'dump gif' command. In batch mode SA, IR, and IIR maps can be prepared in GIF format. In GUI mode the <colorMapID> can be one of the following types (case insensitive):

(Most are also available from the 'View Results' menu buttons:)

- SA: show all design features
- IR: IR voltage analysis (static)
- EM: electromagnetic analysis
- CUR: node current
- TP: thermal profile per layer
- PD: power density
- IPM: instance power
- CIP: clock instance power
- LPM: instance leakage power
- IFM: instance frequency
- DD: decap density
- DEV: device capacitance density
- LC: load cap density
- IDD: inserted decap density
- PG: power/ground cap density
- ISM: slack per instance
- IJK: instance K-factor
- IID: instance delta delay from DvD
- SICM: switch instance current color map
- SIVM: switch instance voltage color map

(From View menu commands:)

- DIM: highlight disconnected instances
(View > Connectivity > Highlight Disconnected Instances)
- DWM: highlight disconnected wires and vias
(View > Connectivity > Highlight Disconnected Wires/Vias)
- ITR: instance toggle rate
(View > Power Maps > Toggle Map of Instances)
- IPCM: instance peak current map
(View > Current Maps > Instance Peak Current Map)
- TRD: toggle rate density (**View > Power Maps > Toggle Density Map**)
- LPD: leakage power density
(View > Power Maps > Leakage Power Density)
- PADC: pad current map (**View > Power Maps > Pad Current Map**)
- PADV: pad voltage map (**View > Power Maps > Pad Voltage Map**)
- TRM: total resistance (**View > Resistance Maps**)
- PRM: power (Vdd) resistance (**View > Resistance Maps**)

GRM: ground (Vss) resistance ([View -> Resistance Maps](#))

WT : Wire Temperature map ([View->Map Configuration->View Button Customization -> Wire Temperature map](#))

IIR: Instance IR map

[\(View->Voltage Drop Maps ->Instance \)](#)

PIR: transistor pin minimum voltage for static or dynamic analysis

[\(View -> Transistor Pin Maps -> Voltage Drop Map\)](#)

PRES: transistor pin resistance

[\(View -> Transistor Pin Maps -> Resistance Map\)](#)

PCUR: transistor pin maximum current

[\(View -> Transistor Pin Maps -> Current Map\)](#)

PMIN: transistor pin minimum voltage over dynamic simulation time

[\(View -> Transistor Pin Maps -> Min voltage over simulation time\)](#)

PAVG: transistor pin average voltage over dynamic simulation time

[\(View -> Transistor Pin Maps -> Avg voltage over simulation time\)](#)

PMAX: transistor pin maximum voltage over dynamic simulation time

[\(View -> Transistor Pin Maps -> Max voltage over simulation time\)](#)

maxTW: maximum Vdd-Vss value over time window

[\(View -> Dynamic Instance DVD -> Max Vdd-Vss Over Time Window\)](#)

minTW: minimum Vdd-Vss value over time window

[\(View -> Dynamic Instance DVD -> Min Vdd-Vss Over Time Window\)](#)

avgTW: average Vdd-Vss value over time window

[\(View -> Dynamic Instance DVD -> Avg Vdd-Vss Over Time Window\)](#)

minCyc: minimum Vdd-Vss value over clock cycle

[\(View -> Dynamic Instance DVD\) -> Min Vdd-Vss Over Whole Cycle\)](#)

Decap: decap contributor ([View -> Decap Maps -> Decap Contributors](#))

RPJ: rising waveform period jitter ([View -> Clock Jitter Map](#))

FPJ: falling waveform period jitter ([View -> Clock Jitter Map](#))

RCCJ: rising cycle-to-cycle jitter ([View -> Clock Jitter Map](#))

FCCJ: falling cycle-to-cycle jitter ([View -> Clock Jitter Map](#))

PDEV: pin-based device capacitance

[\(View -> Transistor Pin Map -> PDEV\)](#)

The following colormapIDs provide peak-to-peak DvD waveforms that have no GUI menu equivalents:

VDDP2P : peak-to-peak DvD waveform of Vdd = max Vdd(t) - min Vdd(t)

GNDP2P : peak-to-peak DvD waveform of Gnd = max Gnd(t) - min Gnd(t)

PGP2P: peak-to-peak DvD waveform of difference (Vdd-Gnd) = max (Vdd-Gnd)(t) - min (Vdd-Gnd)(t)

Examples of these peak-to-peak waveforms are shown in Figure D-2.

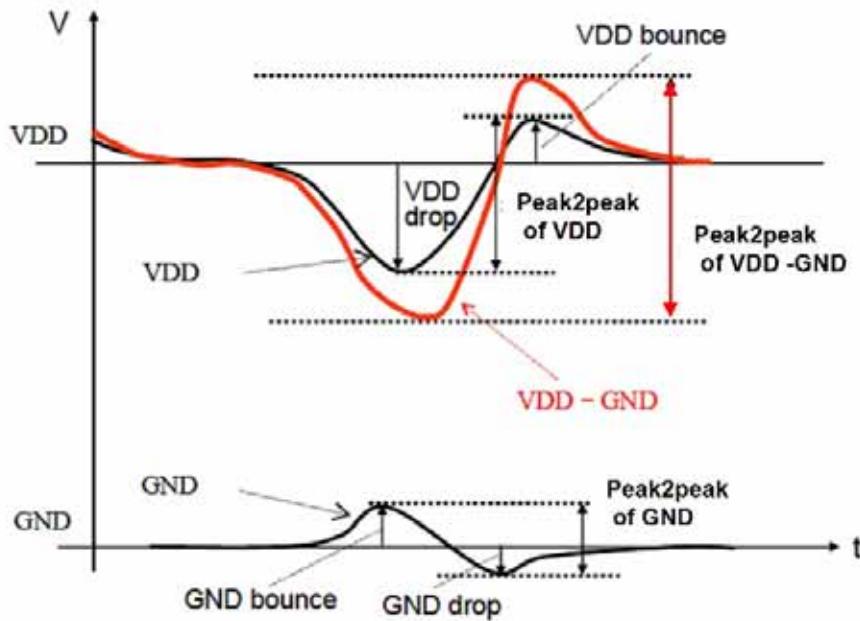


Figure D-2 Peak-to-peak DvD waveform definitions

- o <outputFileName> : specifies the GIF output file name.
- nolegend : switches off the printing of logo / legend. Default - prints logo / legend

Note: see the example TCL script at the end of the chapter using the ‘dump gif’ command, [page D-820](#).

- b. ‘dump gif -map ipcm -o <filename>

Displays the instance peak current distribution in the design in a colormap display (in microAmps)

- c. ‘dump fao’ <options>

Writes out Fix And Optimize-related TCL variable settings to screen or to a file, and either mesh or decap-related settings, or all FAO data (with no options).

```
dump fao ? [-mesh | -decap] ? ?-o <tcl_file>?
```

where

 - mesh | -decap : selects either mesh or decap-related settings to dump
 - o <tcl_file> : specifies the name of the TCL file output.

- d. The ‘dump macro_pin_voltage’ command reports voltages at the LEF macro pins, and the voltages at parent LEF pin locations of all instances of cells specified in GSR keyword GDS_CELLS. This command can be executed after static/dynamic analysis using the syntax :

```
dump macro_pin_voltage ?-inst? ?-gds [0|1]?
?-o <filename>?
```

All arguments are optional. The basic command reports voltages for all macros in the design. The options are:

 - inst : specifies the instance names for pin voltage reporting (default: all instances)
 - gds [0|1] : if set to 1, reports voltages on all GDS2DEF-created pins also (default: 0)
 - o : specifies an output filename (default file: .apache/.macropinvoltage)

- e. The ‘dump memory_via’ command reports via information to the file specified after simulation, including the value of total upper/lower metal area per via model.

```
dump -memory_via -o <file_name> -cells { <cell_list> }
where
```

-o <file_name> : specifies the output filename

-cells { <master_cell_list> } : specifies the master cell names to have instances listed

The format of the memory via information file is as follows:

```
<mem_master_name> <mem_inst1_name> <inst1_bbox>
<total_inst_current><via_model1_name> <num_of_vias> <total_cut_area>
<total_lower_metal_area> <total_upper_metal_area> <net_name>
<via_model2_name> ...
<mem_master_name> <mem_inst2_name> <inst2_bbox> ...
```

The above is repeated for each memory instance in the design as specified.

- f. The ‘dump mmx_pin_info’ command writes out transistor pin information within a selected x,y region.

```
dump mmx_pin_info ?-avg_volt_tw <tstart> <tend>? ?-o <output_file> ?
?-box <llx> <lly> <urx> <ury>? ?-report_disconnect? ? -wstpgarc ?
?-j <thread>? ?-inst_file <list>?
```

where

-avg_volt_tw <tstart> <tend>: specifies the timing window start and end times (ns)

-o <outfile> : specifies the name of the output file. If no output file is specified, a summary is displayed in the GUI and written to the log file.

-box <llx>, <lly>, <urx>, <ury>: specifies the lower left and upper right corner coordinates for the region of interest.

-report_disconnect: dumps out information if the transistor pin is disconnected from P/G grid

-wstpgarc: generates subckt-based worst case Vdd-Vss reports in MMX-based transistor analysis

-j <threads> : number of threads for multi-CPU machines. Default is 1.

-inst_file : specifies a list of leaf instances and subckts whose Vdd-Vss values are to be found. If not given, it finds all instances subckts.

The inst_file contents format is:

```
<leaf_instance_name> <subckt_name_pattern>
```

For example:

```
adsU1 X112.X3.X0*
adsU1 X614.X3.X0*
```

An example of a subckt dump command output:

```
#<InstName> <Subckt Name> <Power Domain> <Ground Domain>
    <Worst VDD-VSS> <Time(ns)> <VDD> <VSS> <VDD Node> <VSS Node>
adsU1 X112.X3.X0 VDD VSS 1.095690 1060.00000 1.138673 0.043021 7546 203793
adsU1 X112.X3.X0.X14 VDD VSS 1.03234 1060.00000 1.130435 0.09809 37273 201343
```

An output file format without the ‘-avg_volt_tw’ option is as follows:

```
#Power(Current) Summary within region LL: (1000.0000 250.0000) UR:
(2000.0000 1100.0000)
#Domain Name Total No.Pin      Total Power / Effective Current
#-----
#VDD          51118           6.1565e-03 W
#VSS          49004           -3.4196e-03 A
```

```
#<InstName:PinName> <Effective Current(A)> <PRatio> (<llx lly>) (<urx ury>)
<TransistorName>
```

```
adsU1:VDD.gds1 1.0321e-04 2103 (84.8250 62.6600) (85.0250 62.8600) X0.M5
adsU1:VDD.gds379 9.3990e-05 4263 (75.9250 60.4700) (76.1250 60.6700)
X0.X15.X31.M91
```

...

where '<llx lly>' and '<urx ury>' represent the lower left and upper right x,y corners of the pin geometry.

The output file format without the '-avg_volt_tw' option, but with the '-report_disconnect' option, is as follows:

```
#<InstName:PinName> <Effective Current(A)> <PRatio> (<llx lly>) (<urx ury>
<TransistorName> <Disconnected>
adsU1:VDD.gds1 1.0321e-04 2103 (84.8250 62.6600) (85.0250 62.8600) X0.M5 N
adsU1:VDD.gds379 9.3990e-05 4263 (75.9250 60.4700) (76.1250 60.6700)
X0.X15.X31.M91 N
...
```

The output file format using the '-avg_volt_tw' option is as follows:

```
#Time Unit: ps
#Time window start time: 0 ps
#Time window end time: 2000 ps
#<inst_name:pg_pin_name> <x_loc> <y_loc> <min_volt_tw@time>
<max_volt_tw@time> <avg_volt_tw> <xtor_name>
adsU1:VDD 82.8900 62.7600 1.7940@845 1.8000@0 1.7982 X0.M3
adsU1:VDD.gds1 84.9250 62.7600 1.7925@910 1.8000@0 1.7980 X0.M5
adsU1:VDD.gds2 11.8500 66.0000 1.7988@230 1.8000@0 1.7997 X0.X8.M3
adsU1:VDD.gds3 21.3250 18.7350 1.7960@350 1.8000@0 1.7990 X0.X9.X0.M22
...
where '<x_loc> <y_loc>' represent the center x,y coordinates of the pin.
```

In GUI mode, if the '-box' option is not specified, the region is based on the command 'condition set' (highest priority) and then the GUI selects the window region.

g. The 'dump node_count' command reports the node profile for a design, using the command:

```
dump node_count -o <output_file>
```

Note that layers with less than 1% of the wires are ignored by the command. A sample node count output report follows:

Layer Based Profile(for the fullchip):

```
#Layer      Nodes
METAL1    0.237805 M
METAL2    0.026096 M
METAL3    0.147819 M
METAL4    0.193270 M
-----
Total     0.604990 M

Total Number of Nodes      = 0.77 M
Die Area                  = 24.60 sqmm
Avg Number of Nodes per Sqmm = 0.03 M/sqmm
```

Master Cell based profile for GDS2DEF cells: Summary

```
#Cell-Name      Num_insts      Avg_nodes_per_inst      Tot_nodes
  cell_50        1            0.109984 M            0.109984 M
  cell_52        1            0.087503 M            0.087503 M
  cell_30        1            0.001403 M            0.001403 M

Master Cell based profile for block DEF cells: Summary
#Cell-Name      Num_insts      Avg_nodes_per_inst      Tot_nodes
  cell_35        1            0.109648 M            0.109648 M
  cell_31        1            0.103521 M            0.103521 M

Cell: cell_30
Number of instances = 1
-----
Layer Nodes_per_inst Total_nodes %of full-chip metal Node count
METAL3  0.000856 M    0.000856 M    0.11
METAL4  0.000547 M    0.000547 M    0.07
-----
Total   0.001403 M    0.001403 M    0.18
-----
```

- h. The ‘dump ptvoltage’ command displays, or writes out to a specified file, the PrimeTime TCL commands for each instance, for feedback to timing using PT. By default the average Vdd over the timing window (avgTW) is written for every instance.

```
dump ptvoltage -o <outFile> ? [ -aveTW | -maxTW | -minTW | -mincyc |
  -include_missing_TW [ minCyc | nominal ] ] ? ? -min_max ?
```

where

- o <outFile> : specifies name of output file
- aveTW : average Vdd over the timing window
- maxTW : maximum Vdd over the timing window
- minTW : minimum Vdd over the timing window
- mincyc : minimum Vdd over the clock cycle
- include_missing_TW : handles the case of missing timing windows, as follows:
 - mincyc - the “minimum voltage over whole cycle” value is used for all TWs
 - nominal - the nominal voltage for that cell (from the DB) is used.
 If '-include_missing_TW' is not used, the command works as usual, which means instances with no timing windows are ignored (not included).
- min_max : runs the ‘set_rail_voltage’ command for all cells for ‘min’ and ‘max’ conditions, and dumps minTW and maxTW voltage values as arguments to the “-min” and “-max” options

- i. The ‘dump res_network’ command creates a report on wires and vias in the design. The command syntax is as follows:

```
dump res_network -o <output_file> ? -compress ?
?-exclude_region <region_file>
?-box <llx> <lly> <urx> <ury> ?-nets <net1,net2,net3...>
?-layers <layer1,layer2,layer3...> ? -ignore_layers
<layer1,layer2,layer3...> ?-conf <config_file>
```

where

- o : specifies the output filename
- compress: saves the output in gzipped (*.gz) format

-exclude_region: specifies a file that defines regions of the network to be excluded in the output dump file. Multiple regions can be defined in a region input file. The format of the region file is:

```
#<llx> <llx> <urx> <ury> <topmost_metal_layer> ...
```

All metal in the region up to the given metal layer are excluded. For example:

```
<llx> <llx> <urx> <ury> METAL3
```

where the output excludes the region on METAL3 and below.

-box : specifies the lower left and upper right corner coordinates of a region of interest for the network dump. Data within the boundaries of the region are then reported in the output file. When “-box” is not defined, all wires and vias in the design are dumped.

-nets : specifies the nets for which report needs to be created.

-layers : specifies the layers for which report needs to be created.

-ignore_layers : specifies the layers to be ignored from 'dump res_network' output file.

-conf : specifies a configuration file that contains the fields in the full dump report that are requested. The configuration file should contain a list of keywords representing the fields desired in the report for 'Wires', 'WireSegments' and 'Vias', as a subset of the full syntax, as follows:

```
wire $WIRE_ID $LAYER $NET_NAME $LLX $LLY $LRX $LRY $URX $URY $ULX $ULY  
$WIRE_WIDTH $RESISTANCE $CURRENT_ORIENT $BLECH_LENGTH $R_FACTOR  
  
wiresegment $WIRESEG_ID $STARTX $STARTY $ENDX $ENDY $AVG_I $AVG_ILIMIT  
$AVG_EM $RMS_I $RMS_ILIMIT $RMS_EM $PEAK_I $PEAK_ILIMIT $PEAK_EM  
$LONG_PEAK_I $LONG_PEAK_ILIMIT $LONG_PEAK_EM $CURRENT_DIR $RESISTANCE  
$RES_ID $TEMPERATURE  
  
VIA $VIA_ID $LAYER $VIA_NAME $NET_NAME $CX $CY $CUT_NUM $CUT_WIDTH  
$CUT_HEIGHT $RESISTANCE $RES_ID $AVG_I $AVG_ILIMIT $AVG_EM $RMS_I  
$RMS_ILIMIT $RMS_EM $PEAK_I $PEAK_ILIMIT $PEAK_EM $CURRENT_DIR  
$VIA_RULE_NAME $LANDING_TYPE $COVERAGE_TYPE $CONNECTIONS  
$WIDTH ABOVE $WIDTH BELOW
```

The configuration file keywords are described below:

For wires:

\$WIRE_ID - unique (internal) integer identifier for each wire

\$LAYER - layer name that this wire is on

\$NET_NAME - net name that contains the wire

\$LLX \$LLY

\$LRX \$LRY - coordinates defining the wire geometry

\$URX \$URY LL - lower left, LR - lower right

\$ULX \$ULY UL - upper left, UR - upper right

\$WIRE_WIDTH - width of wire in um

\$RESISTANCE - wire resistance in ohms

\$CURRENT_ORIENT - wire orientation (based on current flow): h = horiz, v = vert

For wire segments:

\$WIRESEG_ID - unique identifier for wire segment of the form X_Y, where X is the WIRE_ID for the wire containing this segment, and Y is the index for the wire segment, starting from 0 for each wire.

\$LAYER - layer name that this wire segment is on

\$NET_NAME - net name that contains the wire segment

\$WIRE_WIDTH - width of the segment in um
 \$STARTX, \$STARTY \$ENDX, \$ENDY - coordinates of the wire segment (spine)
 \$AVG_I - current through this segment during EM_AVG analysis
 \$AVG_ILIMIT - current limit for this segment for EM_AVG analysis
 \$AVG_EM - EM percent for EM_AVG analysis (I/ILIMIT)
 \$RMS_I - current through this segment during EM_RMS analysis
 \$RMS_ILIMIT - current limit for this segment for EM_RMS analysis
 \$RMS_EM - EM percent for EM_RMS analysis (I/ILIMIT)
 \$PEAK_I - current through this segment during EM_PEAK analysis
 \$PEAK_ILIMIT - current limit for this segment for EM_PEAK analysis
 \$PEAK_EM - EM percent for EM_PEAK analysis (I/ILIMIT)
 \$RESISTANCE - wire segment resistance in ohms
 \$RES_ID - unique (internal) integer id for this segment's resistor
 \$CURRENT_DIR - actual current direction through this wire segment: u = up, d = down, l = left,
 r = right

For Vias:

\$VIA_ID - unique (internal) integer identifier for each via
 \$LAYER - cut layer name for this via
 \$VIA_NAME - via model name that applies to this via
 \$NET_NAME - net name that contains the via
 \$CX, \$CY - X,Y location of the via's center point
 \$CUT_NUM - number of via cuts grouped together
 \$CUT_WIDTH - width of the cut rectangle for this via
 \$CUT_HEIGHT - height of the cut rectangle for this via
 \$RESISTANCE - via resistance in ohms
 \$RES_ID - unique (internal) integer id for this via's resistor
 \$AVG_I - current through this via during EM_AVG analysis
 \$AVG_ILIMIT - current limit for this via for EM_AVG analysis
 \$AVG_EM - EM percent for EM_AVG analysis (I/ILIMIT)
 \$RMS_I - current through this via during EM_RMS analysis
 \$RMS_ILIMIT - current limit for this via for EM_RMS analysis
 \$RMS_EM - EM percent for EM_RMS analysis (I/ILIMIT)
 \$PEAK_I - current through this via during EM_PEAK analysis
 \$PEAK_ILIMIT - current limit for this via for EM_PEAK analysis
 \$PEAK_EM - EM percent for EM_PEAK analysis (I/ILIMIT)
 \$CURRENT_DIR - actual current direction through this via: u = up, d = down
 \$VIA_RULE_NAME - via landing rule name (if any) associated with this via
 \$LANDING_TYPE - via landing type (if any) of this via
 \$COVERAGE_TYPE - via cover type (if any) of this via
 \$CONNECTIONS - outputs a list of wire segment id's that are attached to this via
 \$WIDTH ABOVE - width of the wire attached to the top of this via
 \$WIDTH BELOW - width of the wire attached to the bottom of this via

An example output file header would be as follows:

```
# For Wires
# <ID> <layer> <net> < ll(x y) lr(x y) ur(x y) ul(x y) > <width>
# <resistance><current_dir h/v >
# For Wire-segments
# <ID> <layer> <net> <start(x y) end(x y) (um)> <direction(l|r|u|d)>
# <current(A)> <em_limit(A)> <em(%)> <resistance(ohm)>
# For Vias
# <ID> <layer> <via_name> <net> <coord(x y) (um)> <cut_#>
# <cut_width(um)><cut_height(um)> <resistance(ohm)> <current_dir(u|d)>
# <current(A)> <em_limit(A)> <em%> <rule-name> <Landing Type>
# <Wire-segid#1> <Wire-segid#2> ... <Wire-segid#N>
```

- j. The ‘dump sp_constraints’ command writes out a *state_propagation.pi* file containing state propagation constraints and also in-out primary nets.
- k. The ‘dump sptemplate’ command writes out the state propagation constraint file template. By default primary input pins and clock roots are included in the file.

dump sptemplate ? -all ?

where

-all : specifies that register outputs are also included in the constraint file

- l. The ‘dump tsv_via’ command dumps Sentinel TSV information into the file *adsRpt/<dieName>/<dieName>.tsv_via*, which allows the Sentinel TSV model file to be imported for each die, using the GSR keyword TSV_MODEL_FILE. The option -coating_thickness supports designs with multiple TSV diameters to generate the tsv array information file. The default thickness = 5% of minimum diameter. The syntax is:

```
dump tsv_via ?-coating_thickness <thickness value>?
-o <file_name>
```

Example: dump tsv_via -coating_thickness 0.33 -o topdie_viaray.tsv

- m. The ‘dump via_count’ command reports the number of vias in the design. Syntax:

dump via_count -o <filename>

- n. The ‘dump wire_count’ command reports the number of wires in the design. Syntax:

dump wire_count -o <filename>

eco

```
eco [ add [ clamp_cell | clamp_instance | decap | pad | strap | stackvia | via |
switch ] ] |
[ delete [ clamp_instance | pad | strap | switch | via ] ] ? <options> ?
```

The ‘eco’ command defines changes to the design database (GUI-based operation only), which is equivalent to GUI ‘what-if’ operation.

- a. ‘eco add clamp_cell’ adds or updates a clamp cell for ESD checking:

```
eco add clamp_cell <cell_name> -width <W> -height <H>
-vdd_pin <pinName_v1 pinName_v2 ...>
-gnd_pin <pinName_g1 pinName_g2 ...>
```

where

-width <W>: specifies the cell width

-height <H>: specifies the cell height

-vdd_pin : specifies power pin name(s)

-gnd_pin: specifies ground pin name(s)

If the specified <cell_name> is previously given, the cell's properties (width, height, vdd/gnd_pin) are updated accordingly.

Note that an existing cell in the original design cannot be updated.

- b. 'eco add clamp_instance' adds or updates a clamp instance for ESD checking.

```
eco add clamp_inst <inst_name> -cell <cell_name>
    -pin <p1 p2...> -net <net1 net2...> -origin <x y>
    ?-orient [N |S |E |W |FN |FS |FE |FW ]?
```

where

- cell <cell_name>: specifies the master cell name of the instance
- pin : specifies the pin name(s) to be assigned net(s)
- net : specifies the net name of each pin
- origin <x y>: specifies the location of the instance
- orient : specifies the orientation of the instance relative to the master cell orientation: North, South, East, West, Flipped North, Flipped South, Flipped East, Flipped West. The default is N.

If the specified <instance_name> is previously given, the instance's properties (cell, pin, net, origin, orientation) are updated accordingly.

Note that the master cell of the added instance can be an existing cell or an ECO-created clamp cell. An existing instance in the original design cannot be updated.

- c. 'eco add decap' adds decap cell as specified. Returns added decap name.

```
eco add decap -metal <metalLayer> -decap <cellName>
    -power <x y> -ground <x y>
```

where

- metal <metalLayer> : specifies the metal layer name.
- decap <cellName> : specifies the decap cell name.
- power <x y> : specifies the decap insertion point for power.
- ground <x y> : specifies the decap insertion point for ground.

- d. 'eco add pad' adds power/ground pad <metalLayer> at location x, y, and also can add pad RLC values when importing a PLOC file. If x,y locations outside of the wire geometry are specified, and the AUTO_PAD_CONNECTION_LAYERS GSR keyword is set, the x,y locations are snapped to existing geometry location and the pad is generated.

```
eco add pad -metal <metal_layer> -type [power | ground]
    -x <x_coord> -y <y_coord> ?-r <Res>? ?-l <Ind>? ?-c <Cap>?
    ?-use_net_name?
```

where

- metal <metal_layer> : specifies the metal layer name.
- type [power | ground] : specifies the power or ground pad to add.
- x <x_coord> : specifies the x location of the insertion in um.
- y <y_coord> : specifies the y location of the insertion in um.
- r <Res> : specifies the resistance of the pad
- c <Cap> : specifies the capacitance of the pad
- l <Ind> : specifies the inductance of the pad
- use_net_name: creates pad name using the net name.

Where the mesh is built after the design is loaded, the pad location can be connected after the mesh is created and the pads are then added with the 'eco add pad' command, which can use one or all three RLC arguments.

Example:

```
eco add pad -metal metal6 -type power -x 14 -y 2 -r 100 -l 1 -c 1
```

The GUI log window displays the pad instance name when it is added. The file *adsRpt/PG_simple.ploc* file shows the pad and RLC values assigned if the GSR keywords 'PRINT_ONE_PLOC_PER_PADINST' and 'PGPLOC_DEBUG' are set.

- e. `eco add strap'` adds strap (wire) as specified. Returns ID of added wire.

```
eco add strap -metal <metalLayer> -type [power | ground]
? -net <netName> ?
-[horizontal | vertical] -start <x y> -end <x y>
-width <width> ? -novias ?
? -toplayer <layerName>? ? -bottomlayer <layerName>?
```

where

- metal <metalLayer> : specifies the metal layer name.
- type [power | ground]: specifies the power or ground strap to add.
- net : specifies one power or ground net (-type then not required)
- [horizontal|vertical] : specifies the direction of the wire to add
- start : specifies the starting coordinates of the strap (lower left corner)
- end : specifies the end coordinates of the strap (upper right corner)
- width <width> : specifies the wire width
- novias : specifies no vias are placed associated with the strap
- toplayer <layerName>
- bottomlayer <layerName>

- f. `eco add stackvia | via'` adds a via between layers as specified.

```
eco add [ stackvia | via ] -toplayer "<netname> <topLayerName>" 
-bottomlayer "<netname> <bottomLayerName>" ? -no_adjust
-x <x> -y <y> ?-viamodel <viaName>?
```

where

- toplayer "<netname> <topLayerName>": specifies the net name and the top layer name for insertion.
- bottomlayer "<netname> <bottomLayerName>": specifies the net name and the bottom layer name for insertion.
- viamodel <viaName>: specifies the via model.
- x : specifies the x location for insertion.
- y : specifies the y location for insertion.
- no_adjust: by default, adds via/stackvia at the center of the overlap; when this option is specified adds via/stackvia at a customized location based on -x/-y settings.

- g. '`eco add switch'` adds a switch instance with the specified characteristics.

```
eco add switch <instName> -master <cellName>
-connect {<pin1 net1 pin2 net2 ...>} -x <x_loc> -y <y_loc>
? -count <xcount> <ycount> -pitch <x_pitch> <y_pitch> ?
? -orient [ N|S|E|W|FN|FS|FE|FW ]?
```

where

- <instName> : specifies the instance name of switch (for placing an array of switches using '-pitch' and '-count' options, see the naming format below)
- master <cellName> : specifies the name of master switch cell
- connect {<pin1 net1 pin2 net2 ...>} : specifies the associated pin and net connections for switches to be added
- x <x_loc> -y <y_loc> : identifies the lower left corner x,y location of the single switch or the first switch in the array

- count <xcount> <ycount> : defines an array of identical switch instances placed adjacent to each other in the x and/or y direction. Switches in the specified array automatically are assigned names of the form '<instName>_a_b', where a and b are a series of consecutive integers starting with the first switch named '<instName>_1_1' at location <x_loc> <y_loc>.
- pitch <x_pitch> <y_pitch> : for an array of switches placed using the '-count' option, specifies the x and y-direction pitch, which is the distance between the west edges (x direction) and south edges (y direction) of adjacent instances.
Note that a negative value for <x_pitch> causes the array to be placed to the left or west of the first instance at <x_loc> <y_loc>, and a negative value for <y_pitch> causes the array to be placed below or south of the first instance.
- orient [N|S|E|W|FN|FS|FE|FW] : specifies the desired switch instance orientation relative to the master cell orientation
- h. 'eco delete clamp_inst' deletes specified clamp instance(s)

```
eco delete clamp_inst { <inst_name> | -all -cell <cell_name> }
```

where
 - inst_name: deletes instance specified
 - all -cell : deletes all instances of the <cell_name> specified
Note that an instance in the original design cannot be deleted.
- i. 'eco delete pad' deletes a specified pad or all pads from the design.

```
eco delete pad <padName> ? -all ?
```
- j. 'eco delete strap' deletes a named strap (wire).

```
eco delete strap [<handle> | <metalLayer> -x <x> -y <y> ]
```

where
 - <handle> : specifies the "handle" for the deleted strap
 - <metalLayer> : specifies the metal layer name for the strap deletion
 - x <x> -y <y> : specifies the x,y coordinates for the strap to be deleted
- k. 'eco delete switch' deletes the specified switch instance or set of switch instances.

```
eco delete switch <instName> ? -index <from_aF:bG> <to_aM:bN> ?
```

where
 - <instName> : specifies the name of the instance to be deleted
 - index <from_aF:bG> <to_aM:bN> : specifies the range of identical switch instances to be deleted from an array, when the switches have been named with the format '<instName>_a_b' using the 'eco add switch -count' command.
For example, the delete command

```
eco delete switch sw -index 1:2 2:4
```

would delete array switch instances 'sw_1_2', 'sw_1_3', 'sw_1_4', 'sw_2_2', 'sw_2_3', and 'sw_2_4'.
- l. 'eco delete via' deletes a specified via or stackvia from the design

```
eco delete via [<handle> | <metalLayer> -x <x> -y <y> ]
```

where
 - [<handle> : specifies the "handle"
 - <metalLayer> : specifies the metal layer name for the via deletion
 - x <x> -y <y> : specifies the x,y coordinates for the via to be deleted

export

export [db | eco | gridcheck | guiconf | res_calc] ? <options> ?

The ‘export’ command exports RedHawk files in different formats for use by particular tools.

- a. ‘export db’ exports the RedHawk database to a specified hierarchical directory:

```
export db <dir1>/<dir2>/<DB_name>.db
```

- b. ‘export eco’ exports a RedHawk ECO file with changes to the database, as a result of FAO modifications to wires, vias, and decap instances, for example.

```
export eco <eco_filename> ? -hier ?
[ -cell {a b c ...} | -def ?-cell {a b c ...}? ?-def2pr?
 | ?-ploc <ploc_filename>? ? -gds_map <gds_layer>.map?
```

where

<eco_filename> : specifies the name of the exported ECO file for the top level design
 -hier : specifies that an exported ECO decap file includes an extra attribute indicating the hierarchical name of the closest instance for each decap.
 -def : in addition to creating the regular database ECO file, creates an additional DEF format file for the top level
 -cell {a b c ...} : in addition to creating the regular top level ECO file, creates files of the type [a / b / c /...].NORTH.eco for each of the sub-block cells (a, b, c ...) specified.

Note: If { a b c . } includes the top design name, the specified <eco_filename> will not include any decap cells included in sub blocks of the design.

-def -cell {a b c ...} : in addition to creating the regular ECO file, and [a / b / c /...].NORTH.eco files for each of the sub-block cells, it also creates block DEF files of the type [a / b / c /...].eco.def

-def -def2pr : in the additional DEF file, RedHawk adds the attribute “+ SOURCE DIST” to each new decap instance line

-ploc <ploc_filename> : specifies a *.ploc file to be generated resulting from an ‘Add Pad’ command, to provide the correct pad location, layer, pin name and net information. The package subckt information can then be manually added to the file.

-gds_map <gds_layer>.map : creates an extra tracing point file in gds2def/gds2rh configuration file format, adsRpt/gdsTracePt, and writes the information in the RedHawk log file and the GUI log window. For the example ECO file below:

```
## output eco file
#Redhawk_Eco_20
DESIGN my_io
UNIT 2000
# UNITS DISTANCE MICRONS 1000 ;
# DIEAREA ( 342000 0 ) ( 722500 1095000 ) ;
# @ 146.182 632.321 << saves computation of real coords
ADD pad VGG_1 GND M7 292365 1264642
# @ 331.253 623.508
ADD pad VGG_2 GND M7 662506 1247017
...
```

the output gdsmap file has the form (where M7 is GDS layer 57):

```
## gdsmap file
...
GND_NETS {
  VGG {
```

```

VGG_1 @ 57    488.182  623.508
#<-- 488.182 is 146.182, translated by 342
      VGG_2 @ 57    673.253  623.508
#<-- 673.253 is 331.253, translated by 342
}
}
...

```

- c. The ‘export gridcheck’ command creates a file containing the results of grid checking that can later be imported. The syntax is:

```
export gridcheck <filename>
```

- d. ‘export guiconf’ specifies the export of a configuration file that saves all GUI color map settings that have been changed and applied. Note that dismissing a GUI dialog using the dialog box ‘X’ (“destroy”) button returns the settings to their previous values, so they would *not* be exported by ‘guiconf’.

```
export guiconf <output_filename>
```

where

<output_filename> : specifies the name of the exported GUI settings configuration file

- e. The ‘export res_calc <filename> ’ command exports to the specified file the effective grid resistances computed from ‘perform res_calc’.

fao

fao add pad <options>

The 'fao add pad' command adds a set of power /ground pads over an entire mesh, or in a defined window. You must specify a window, a metal layer, and also a pitch in both x and y directions by which to distribute pads evenly over the specified region. **Undo/Redo** commands can be used. The command line syntax is:

```

fao add pad -window {llx lly urx ury} -metal <layer>
            -net <net_name> -pitchX <microns> -pitchY <microns>
            -r <resistance> -l <inductance> -c <capacitance>

```

where

-window { }: defines lower left and upper right x,y coordinates of the rectangular area in which pads are added (by default, the entire mesh).

-metal: defines the routing layer (required). Only one metal layer is allowed, since every metal has its own placement.

-net: defines the net name (required)

-pitchX -pitchY: defines the incremental distance between pads, starting at lower left corner of the rectangle specified by '-window { }' (required) .

For example, for “-window {a b c d}”, starting at {a b}, RedHawk adds pads at {a b}, {a+pitchX b}, {a+2*pitchX b} ..., {a b+pitchY}, {a+pitchX b+pitchY}, {a+2*pitchX b+pitchY} ... , over the entire window.

-r -l -c: specifies pad RLC parameters (units the same as in the PLOC file)

generate

generate [msdf | simulationdeck | lef2spice | pinmap | mycyc_stats] ? <options> ?

The ‘generate’ command generates specified files for review and/or future use.

- a. ‘generate msdf’ generates an MSDF (Modified Standard Delay File) containing SDF delay values derated for the effects of dynamic voltage drop.

```
generate msdf -i <sdfFileName> -o <msdfFileName> ?-isdf?
?-pvt <pvt_value>? ?[-min | -max | -typical ]?
?-dvd <dvd_threshold>?
?[-w "<logical_module_name>" "<physical_module_name>"]?
```

where

- i <sdfFileName>: specifies the input SDF filename.
- o <msdfFileName>: specifies the output MSDF filename.
- isdf: output MSDF file contains only delta delay values relative to SDF file delay values.
- pvt <pvt_value> if set to 1 (default), delay values are scaled if the **RedHawk** operating temperature value (in APL) matches any of the temperature definitions (Min, Max, Typical) specified in SDF. If the temperature definitions do not match, no derating is performed. If set to 0, all delay values are scaled regardless of **RedHawk** and SDF operating temperature specifications.
- min, -max, -typical: specifies which SDF values are used for performing voltage derating, If any are selected, the -pvt selection is ignored. If none are selected, derating is determined by the -pvt selection.
- dvd <dvd_threshold>: If selected, provides normal derating for delay values whose associated voltage is equal to or below the DvD threshold voltage (V_{nom} times the selected threshold ratio between 0.8 and 1.2). DvD voltages above the threshold value are derated as if they were DvD Threshold voltage.
- w "<logical_module_name>" "<physical_module_name>": allows users to specify logical and physical block names. This is useful, for example, when you want to evaluate the voltage drop impact on delay, and the top block and a sub-block are not defined in both SDF and in **RedHawk** hierarchy. In this case, the IP instance names do not match the hierarchical **RedHawk** names.

For example, in case the SDF is generated for a block called "block_1" at IP level and **RedHawk** has been run at top level, use:

```
generate msdf -w "" "top/block_1/" -i file.sdf -o file.msdf
```

Or, if SDF is generated for the top level and **RedHawk** is only run for a sub-block "block_2", use:

```
generate msdf -w "top/block_2/" "" -i file.sdf -o file.msdf
```

- b. 'generate simulationdeck' can be used to generates the **Spice** deck for DvD backannotation to the timing flow:

Note: there must be at least one block in the design that is modeled with MMX methodology for the command to generate a functional deck.

Syntax:

```
generate simulationdeck -mmx -aplmmx_cfg <filename>
-sim_time <time> ? -inst <inst_name>?
? -xtor_list <filename> ? ? -dir <output_dir>?
? -aplmmx_run_dir <run_dir_name>?
```

where

- mmx: specifies MMX **Spice** simulation deck
- aplmmx_cfg <filename>: specifies APLMMX configuration file
- sim_time <time>: specifies total simulation time for DvD back-annotated Spice simulation (ps)
- inst <inst_name>: specifies the MMX macro instance name (only required in SOC flow)
- xtor_list <filename>: specifies the file that lists the transistors for which you want to back-annotate DvD noise waveforms.
- dir <output_dir>: specifies the output directory to dump out **Spice** deck for back-annotation simulation. Default is 'aplmmx_ba'.

- aplmmx_run_dir <run_dir_name>: specifies the APLMMX run directory
- c. 'generate lef2spice' generates a **Spice** file from existing LEF design data.

```
generate lef2spice -c <cell list filename> -s <subckts filename>
    -l <lefs filename> ?-o <output filename>?
```

where

 - c <cell list filename> : specifies a file containing list of cells to be included
 - s <subckts filename> : specifies the SPICE subcircuit list file
 - l <lefs filename> : specifies the file containing the LEF descriptions of I/O cells
 - o <output filename> : specifies the output filename. Default: *lef2spice.map*
- d. 'generate pinmap' generates a pinmap file

```
generate pinmap -i <extracted p/g network> -s <subckts>
    ?-o <outputFileName>?
```

where

 - i <extracted_pg_network> : specifies the file of the extracted power/ground/signal network
 - s <subckt_file> : specifies the **Spice** subcircuit list file
 - o <output filename> : specifies the output file. Default: *pin.map*
- 'generate mycyc_stats' provides mcyc statistics based on cycle voltages annotated on Timing Paths and creates statistics on a per-path basis. Input is provided in terms of traces for each path. The syntax is:

```
generate mycyc_stats -in <inout_path_file>
```

get

```
get [ analysis_mode | build | cell | celltype | current | design | inst | instbynet | instofcell |
    master | net | orientation | pad | switch | viamodel | ver ] ?<options>?
    ? -out_file <file> ?
```

You can query the database for object attributes, and different types of reports on results, such as voltage drop measurements, and EM and power data. The 'get' command allows querying the database to find object names based on pattern matching to

- search and find design elements
- execute scripts
- allow deeper access to the design

For details on the 'get' commands, use the TCL 'help' command, which lists all available objects and options. To see a list of top level 'get' objects, use the command:

```
help get
```

which displays the objects available in the log window.

Lists of the following object types can be obtained by pattern matching: cell, celltype, inst, instbynet, instofcell, master, net, orientation, pad, switch, ver:

```
<object_type> <pattern> [-glob | -regexp | -exact ]
```

where '-exact' is the default matching.

The following objects have additional sub-options defining additional properties that can be requested: cell, inst, net, pad, design, viamodel, build. A list of additional properties available for each object can be obtained with a Help request of the form:

```
get cell -help
```

General 'get' command descriptions and examples follow. See additional details in [Appendix D1 - TCL 'get' Command Reference](#).

- a. 'get cell <options>'
 Returns a TCL list of cell names that match the options specified.,.
- a1. 'get cell <pattern> [-glob|-regexp|-exact] -type macro'
 Returns a list of cells of type 'macro'.
- a2. 'get cell <pattern> [-glob | -regexp | -exact] -design'
 Returns a list of cells referenced in the design. For example, for the regular expression 'cell12_*':

```
get cell cell12_* -glob
```
- a3. 'get cell * -bbox <x1> <y1> <x2> <y2> -type <cell_type>'
 Using '-type' and '-bbox' options together allows you to get all of the cell names in a given bbox that are of a particular cell type.
- b. 'get celltype' returns the cell type of a specific instance, such as inst241:

```
get celltype inst241
```
- c. 'get current' returns the sum of currents per via array in an output file, and displays to the sum of the via currents per domain. The syntax is:

```
get current -layer <via_cut_layername>
            -bbox <llx lly urx ury> -space <spacing_um> -o <output>
```

 where
 - layer: specifies the via cut layer name
 - bbox: specifies the lower left and upper right x,y corners of the query area
 - space : specifies a maximum distance between via cuts in um in order to consider the cuts as one, and sum their currents. If the spacing is larger than specified, RedHawk treats the two cut vias as separate and does not sum their currents. RedHawk uses this option to find the original via array. Default is 0 (no cuts are summed).
 - o: specifies an output file (default is .apache/.viacurrent), which has the format:

```
#Via_Layer Via_Name Via_Cut_Number Location Sum_Current
```
- d. 'get design <options>' returns the specified design information
 - d1. 'get design -worst_inst_drop -static ?-out_file <fname>
 -dvd_type <t> -net <> -limit <N>'
 Returns the worst instance name and worst static voltage drop for the specified net and DvD type in the design.
 where
 -limit <N>: returns the top N violations in the design.
 Output format:

```
<EM %> <X1> <Y1> <X2> <Y2> <layer> <netname>
```
 - d2. 'get design -layers'
 Returns all layers and vias in the design.
 - d3. 'get design -<type>_coverage -block <block name>'
 Reports block instance-specific results by coverage type. The option -<type> can be ipf, spef, sta, or vcd.
 - d4. 'get design -actual_bbox'
 to dump the actual rectilinear co-ordinates of the design (instead of rectangular) when die (top def) is actually rectilinear.
 - e. 'get inst' <options> returns a TCL list of instance names that match the class specified.
 See [Appendix D1 - TCL 'get' Command Reference](#) for details.

- f. “get instofcell’ <options>
 - Returns a TCL list of instance names for the given master cell, such as for cell112_8:
get instofcell cell112_8
- g. ‘get master’
 - Returns the master cell of the given instance, such as for inst241:
get master inst241
- h. ‘get pad <padName> -net’
 - Returns the name of the net connected to pad specified.
- i. ‘get switch <options>’
 - Returns the switches matching the class specified.
- i1. ‘get switch <switch_name> -int_voltage’
 - Returns the switch internal node voltage.
- i2. ‘get switch <switch_name> -voltage_drop -out_file<filename>’
 - Returns the voltage drop across the switch into the specified file.
- i3. ‘get switch <switch_name> -current’
 - Returns the current through the switch.
- i4. ‘get switch <switch_name> -net_pair’
 - Returns the external and internal net-pair for a switch instance, in format VDD_NET1:VDD_NET2.
- i5. ‘get switch <switch_name> -turn_on_time’
 - Returns the switch turn-on time.

gsr

```
gsr [ get <GSR keyword> | [ set | append ] <GSR keyword> {
  {<keywordValues>} ... } | dump ? -o <filename> ? ]
```

The ‘`gsr`’ command retrieves GSR variable values, sets GSR values, displays the contents, or sends the GSR contents to a file. The command only supports integer, double, Boolean, and string types.

- a. ‘`gsr get`’ retrieves the present value of the specified GSR keyword.
`gsr get <GSR keyword>`
- b. ‘`gsr set`’ sets the value of a GSR keyword (can be set to null), and deletes all previous values, as follows:

```
gsr set <GSR keyword> <keywordValue> |
```

Note that GSR keywords can have long compound values, lists, and more complicated structures, and a one-line continuous string is required (no new lines):

```
gsr set <GSR_keyword> {{<option1> <value11> <value12>...}
  {<option2> <value21> <value22>...} ... }
```

For example:

```
gsr set VCD_FILE {{top file.vcd} {FRONT_PATH "testbench/"}
  {SUBSTITUTE_PATH ""} {FRAME_SIZE 2000} {START_TIME 10000}
  {END_TIME 20000} {TRUE_TIME 1}}
```

1. Example for input data-related keywords:

```
import gsr <>.gsr
gsr set LIB_FILES { .lib CUSTOM }
gsr set DEF_FILES {{block1.def block} {block2.def block} {top.def top}}
setup design
```

The specified input data then are automatically imported during the ‘`setup design`’ step.

2. Using 'gsr set' and 'gsr get' commands for specifying and retrieving equivalent power/ground nets. The syntax is:

```
gsr set VDD_NETS {{VDD <vdd_value> { <equiv_Net> } }
                  {VDDQ <vddq_value> { <equiv_Net> } }}
```

Example:

```
gsr set VDD_NETS {{VDD 1.1 {VDD1CORE }} {VDDQ 3.3{VDDQ1 } }}
```

The output then would be:

```
VDD_NETS {
  VDD 1.1 {
    VDD1CORE
  }
  VDDQ 3.3 {
    VDDQ1
  }
}
```

- c. 'gsr append' adds specified new values to the existing values of a GSR keyword (no existing values are changed), using the same syntax as 'gsr set', as follows:

```
gsr append <GSR keyword> <new_keyword_option> <new_values>
```

- d. 'gsr dump' displays all intermediate level GSR keyword values on the screen, or sends all intermediate level GSR keywords to a specified file.

```
gsr dump ?-o <filename>?
```

where

<GSR keyword> : the specified keyword to operate on

<keywordValue> : specifies a new value for the keyword

<filename> : specifies the file in which to copy all GSR intermediate keywords and values

gui

gui

The 'gui' command opens a GUI from an interactive ('-i' option) non-GUI mode session. The syntax is:

```
gui ? <tcl_cmd_file>
```

history

The 'history' command displays all previous commands typed on the command line during the session.

The syntax is:

```
history
```

import

```
import [ apl | avm | db | def | eco | gsr | gsc | gridcheck | guiconf | keybinding | lef | lib | pad |
        power | res_calc | sdf | sta | tech | viamodel ] ? <options> ? <inputFileName>
```

The 'import' command imports a number of types of files for use in **RedHawk** analysis, such as APL, AVM, DB, DEF, ECO, GSR, guiconf, LEF, LIB, PAD, POWER, and TECH. The general syntax for the command is:

```
import <file_type> <file_name>
```

The import commands with more complicated syntax are described below.

- a. The 'import apl' command imports an APL -generated file

```
import apl ?-c? <inputFileName>
```

where

- c : specifies importing of an APL-generated device capacitance file.
- b. The ‘import avm <configFilename>’ command runs the AVM utility and imports the results into the **RedHawk** database.
- c. The ‘import db’ command imports a specified database with associated options.

```
import db <dir_name> ?-cache_mode [0|1]?
?-cache_dir <cacheTempDir>? ? -ignore_layout?
```

where

<dir_name>: specifies the path and directory name of the database to import
 -cache_mode [0|1]: when set to 1 enables adaptive disk caching for ‘import db’. This only overrides the global setting of the GSR keyword CACHE_MODE for the ‘import db’ operation. Make sure that there is local disk space to use for disk caching.
 -cache_dir <cacheTempDir>: specifies the directory where data is cached for ‘import db’. This only overrides the global setting of the GSR keyword CACHE_DIR for the ‘import db’ operation. For efficient operation, it should be on a local disk where **RedHawk** is running. Default: current working directory.
 -ignore_layout : does not load layout data from the DB/.MM directory, but repeats post-simulation processing to re-create all IR/EM maps. This option can be used as a work-around for ‘import db’ when the layout stored in the DB/.MM directory has been corrupted.

NOTE

The compatibility of **RedHawk** databases from different releases is as follows:

- If the database version and **RedHawk** version are of the same major release, but from different minor releases or patches:
 - A database generated by an older version of **RedHawk** can be loaded with a newer version of **RedHawk**, but new features in the newer versions of **RedHawk** are not available.
 - A database generated by a newer version of **RedHawk** cannot be loaded by an older version of **RedHawk**.
- If the database version and **RedHawk** version are from different major releases (for example, 2010.x and 2011.x), the database cannot be loaded across these versions.

*Apache recommends that the database version and the **RedHawk** version used to load the DB be the same. The “database version” is the version of the tool used to generate the database.*

- d. The ‘import def’ command imports a DEF file data into the design.

```
import def ?[-vddonly | -vssonly]? <inputFileName>
```

where

-vddonly | -vssonly : selects importation of Vdd or Vss data only
 <inputFileName> : specifies DEF input file name

- e. The ‘import esdcd’ command imports results from previously-run ESD or EM CD checks to create color map displays. The syntax is:

```
import esdcd {<rule_name1> ... } ?[-exact|-glob|-regexp]?
?-outDir <dir_name>? ? -cacheEM [0|3]? ? -EMScale <value>?
?-EMRuleSet <name>? ? -saveAs <results_filename>?
```

where

<rule_name1> ... : specifies use of results from one rule, or merges results from several rules specified.
 -exact/-glob/-regexp : provides filtering for rule names specified; default: exact
 -outDir: specifies the text report directory

- cacheEM : set to 3 to reuse the saved EM limit; default: 0 (no reuse)
- EMScale : sets scale factor for EM limit; default: 1.0
- EMRuleSet : specifies name of EM rule set to be used for EM checking
- f. The ‘import gridcheck’ command allows importing the results of a previously run gridcheck. The syntax is:


```
import gridcheck <filename>
```
- g. The ‘import guiconf’ command allows importing a GUI configuration file after the GUI is already opened. The syntax is: ‘import guiconf <gui_config_file>’

Note that the “import guiconf” command is not supported directly in batch mode, and the environment variable ‘APACHE_CONF_FILE’ does not work.
- h. The ‘import keybinding <configFilename>’ command imports a file defining user-specified bindkeys. For a description of the keybinding file format, see [section "Defining Bindkey Functions", page D-867](#).
- i. The ‘import pad’ command Imports pad cell (.pcell), pad instance (.pad), or pad location (.ploc) files. The ‘import pad’ command is disabled after setup design is completed.
- j. The ‘import res_calc <filename> ’ command imports effective grid resistances computed from ‘perform res_calc’ and exported
- k. The ‘import sdf’ command imports a specified SDF file with the syntax:


```
import sdf ?-check_only? ?[-use_min | -use_typ | -use_max]?
<sdfFileName>
```

where

 - check_only : only check syntax without importing into DB.
 - use_min : use minimum values.
 - use_typ : use typical values (default).
 - use_max : use maximum values

license get

license get <license_type>

The ‘license get’ command allows you to select and hold a particular type of RedHawk license, such as for VCD:

```
license get redhawk_vcd
```

marker

marker [add | addfile | clearall | delete] <options>

The ‘marker’ command adds or removes cross-hair marker(s) in the design, specified either as an x,y position or as an instance, or a file with list of instance or position markers. You cannot group markers having different colors and sizes together. When adding markers, specifying a name is recommended only if markers are to be manipulated as a group, or must have different attributes than other named groups of markers. Otherwise, assigning a name for markers is not necessary.

- a. ‘marker add -position’ generates a cross-hair marker with a specified color and size, and returns a marker name.

```
marker add -position <x> <y>
?-name [ <crosshair | star | diamond | rectangle | circle |
          uptriangle | downtriangle> ]?
?-color <color>? ?-size <size>?
```

where

<x> <y> : specifies the x,y location for the new marker

- name <marker_name>: all markers added with the same name form a group and have the same attributes. This helps you delete a group of markers together, and makes marker loading and deleting in the GUI faster, and memory use more efficient, irrespective of the size of the design. If '-name' is not specified, markers are assigned to a default group in which all markers have attributes as specified, or if not specified, they have the default attributes.
- color <color> : specifies the color of the new marker. Default: yellow
- size <size> : specifies a number indicating the relative size of the new marker. Size increases with number value, with no maximum. Default: 3 (small) .
- b. 'marker add -instance' generates a cross-hair marker for the specified instance.

```
marker add -instance <instance_name> ? -name <marker_name> ?
    ?-color <color>? ?-size <size>?
```

 where
 - instance : name of one instance to be marked
 - name: where a group of markers is to be manipulated, a marker name can be assigned to the group
- c. 'marker addfile <filename>' adds markers from a list in the specified file.

```
marker addfile <filename> [-instance | -position ]
    ? -name <marker_name> ? ?-color <>? ?-size <>?
```

 where
 - instance: the specified <filename> contains instance names, one per line
 - position : the specified <filename> contains <x, y> coordinates for the markers, one location per line.
 - name: where a group of markers is to be manipulated, a marker name can be assigned to the group

Note: the marker command puts a heavy demand on memory and CPU resources. Using more than about 10K markers makes drawing slow and could exhaust memory.
- d. 'marker [clearall | delete]' clears all markers or deletes markers as a group by name.

```
marker [ clearall | delete <marker_name> ]
```

 where
 - clearall: deletes all markers
 - delete : specifies the markers to be deleted as a group by marker name.

mesh

mesh [add | delete | fix | generate| optimize | sns calc | sub_grid| set_width | vias] ? <options> ?

The 'mesh' commands perform various types of modifications to power grids, including adding, deleting, and modifying widths and spacing, to reduce voltage drop.

See [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#), for details about this command.

message

message [disable | enable | info | list] ? <options> ?

The 'message' command displays Error, Info and Warning message information.

- a. 'message info' displays a detailed description of a specified message.

```
message info <message_id>
```

where

- <message_id> : unique identifier of message to be displayed
- b. ‘message list’ displays a list of all message IDs that are currently available.
- ```
message list
```
- c. ‘message disable’ disables display of selected message IDs or a class of messages.
- ```
message disable [all | errors | info | warnings | <list of ids>]
```
- where
- all : specifies all messages are disabled
 - errors: specifies all Error messages are disabled
 - info: specifies all Info messages are disabled
 - warnings: specifies all Warning messages are disabled
 - <list of ids>: specifies particular message IDs are disabled
- d. ‘message enable’ allows message display for the specified message IDs or a class of messages.
- ```
message enable [all | errors | info | warnings | <list of ids>]
```
- where
- all : specifies all messages are enabled
  - errors: specifies all Error messages are enabled
  - info: specifies all Info messages are enabled
  - warnings: specifies all Warning messages are enabled
  - <list of ids>: specifies particular message IDs are enabled

## movie

---

### **movie [ make | play ] ? <options> ?**

The ‘movie’ command sets up or plays an instance-based or transistor pin based “movie” of DvD performance, with the following options:

- a. ‘movie make’ saves data to create movie of dynamic voltage drop (DvD) changes sampled over the timing window.
- ```
movie make ?-transistor? ?-zoom? ?-time <t1> <t2>? ?-step <dt>?  
?-dir <path>? ? -nolegend ? -high_light_switching_instances ?  
-transistor: creates a movie based on transistor-level pin voltages. Default is an instance-based DvD movie.  
-zoom : makes the movie view the current display zoom level  
-time <t1> <t2>: specifies a start and end time interval within the simulation time for a shorter movie (ps by default).  
-step <time_step>: specifies a longer time step for frame sampling, in multiples of the time step specified in the GSR keyword DYNAMIC_TIME_STEP (ps units by default)  
-nolegend : eliminates voltage scale and time values that are annotated by default on the exported GIFs  
-high_light_switching_instances : highlights the switching instances in dynamic analysis
```
- b. ‘movie play’ displays frame-by-frame in the GUI the movie specified in the ‘movie make’ command.
- ```
movie play ?-exit? ?-dir <path>?
-exit: removes the movie display window when the last frame is played.
-dir <path>: specifies directory path for writing to or reading from the movie files.
```

**perform**

```
perform [analysis | clampcheck | cmmcheck | cpa | emcheck | esdcheck | extraction |
gridcheck | jitter | min_res_path | pwrcalc | powermodel | res_calc | thermalmodel] ?
<options> ?
```

The ‘perform’ command runs selected **RedHawk** analyses. The standard syntax is ‘**perform <cmd\_name>**’. Commands that have more options are described below.

- a. ‘**perform analysis**’ executes various forms of **RedHawk** analysis, along with special options.

```
perform analysis [-lowpower ?-alp3d ? | -static | -dynamic
| -vcd ? -t <start_time> <end_time>? | -v(ector)less | -signaleM
| -jitter [-vcd] ? -high_capacity 1? ? -clock ? ? -scan?
?-mcycle? ?-pwcap <pwcap_file>? ? -simtime <sim_time>? ? -ipwr_only?
?-early ? ? -nx <nx> -ny <ny>? ? -sir ?
```

where

- lowpower: runs ramp-up power analysis, as specified in the GSC file
- alp3d : accelerates ramp-up analysis about 3X, with an accuracy impact of only 10-15%. Note that the time step must be  $\leq 150$  ps.
- static: runs static (DC) power analysis
- dynamic : runs mixed mode power analysis, with a combination of vectorless and VCD methods on different sections of the design. VCD blocks follow the switching scenario in VCD and the remainder follow the vectorless flow (switching according to the toggle rate).
- vcd -t <start\_time> <end\_time>: specifies the simulation start and end times for vcd-based analysis, and overrides both the START\_TIME/END\_TIME defined in the VCD\_FILE GSR keyword and the most critical cycle selected by cycle selection. In mixed mode, VCD blocks follow the switching scenario in VCD and the remainder vectorless blocks do not toggle, unless specified in the GSC, since otherwise it is assumed that 'pure vcd' analysis is desired.
- vectorless (or “-dvd”): entire design (both VCD blocks and vectorless blocks) follows the vectorless flow (switching according to the toggle rate). For regular multi-cycle vectorless analysis, **RedHawk** generates a switching scenario using the vectorless algorithm for the first cycle. For the second cycle, the same instances switching in the first cycle transition in the opposite direction. The third cycle is then the same as the first cycle and the fourth cycle is the same as the second cycle, and this behavior continues for each subsequent cycle. To have a different vectorless switching scenario in each cycle, use the ‘-mcycle’ option. In mixed mode the whole design follows the vectorless flow (switching according to the toggle rate).
- signaleM: runs static signal EM analysis
- jitter : specifies jitter analysis, using either a -vcd or -vectorless switching scenario and an associated timing configuration file. The -clock option only reports waveforms for clock instances used for clock jitter or skew analysis. This option should be used for very long simulations to avoid large amounts of data for other non-clock instances being saved.
- scan : runs fast simulation with less accurate results than normal mode.
- high\_capacity 1: uses high-performance ASIM3D simulation engine to achieve faster run time and less memory usage. Supports Dynamic, Low power, Transient and CPM flows (On by default). Specifying -high\_capacity 0 will switch the solver to asim\_mixed.
- mcycle : for vectorless only, this option forces **RedHawk** to generate a new list of switching instances for each cycle in the simulation.
- pwcap <pwcap\_file> : specifies piece-wise capacitance file.
- simtime <sim\_time> : specifies the simulation time in picoseconds
- ipwr\_only: runs power analysis with aggressive node reduction for quick ipwr waveform generation.

- early : creates waveforms for demand current with accelerated simulation results (less aggressive node reduction than -ipwr\_only).
- nx -ny : runs power analysis on a grid basis, with <nx> and <ny> specifying the number of partitions or grid sections in the x- and y-directions. The demand current is captured for each partition in the files:
  - *adsRpt/Dynamic/<design>.partition.ipwr*
  - *adsRpt/Dynamic/<design>.partition.ignd*
- sir : automatically executes the ‘export db’ command before simulation and the ‘import db’ command after simulation is complete, to optimize memory use for designs requiring very large amounts of memory. Supports all power analysis commands except -signalEM and -jitter.

Note that runtime is increased in the SiR flow due to the additional export and import DB steps. However, SiR may allow you to complete a run that would otherwise fail due to insufficient memory using the normal flow.

b. ‘perform clampcheck’

reports a list of clamps with the specified attributes.

```
perform clampcheck
? -o <out_file>? ?-instConn? ? -clamp <rule_file> ?
? -bump <name1, name2, name3....>? ? -isolatedBump?
? -cell <cell_name>? ?-celltype <type>?
? -inst <inst_name>? ? -volt <voltage>? ?-net <net_name>?
? -netConn <net1> <net2> ...? ? -allNetConn ?
? -bumpConn <bump_name> ? ? -allBumpConn ? ? -loop?
? -loopLength <stage_num>? ? -b2bLoopLength <stage_num>?
? -roff <R_thresh> ? ? -rptDisConn ? ? -rule <ruleName> ?
? -esdStage <stage_num>?
? -detail? ?-append? ? -summary
```

where

- o : specifies the output filename
- clamp : verifies that the clamp pin location specified in the rule file is on the physical pin location of the clamp. Detailed messages are displayed in the *esd\_info.rpt* file.
- bump : to report ESD pin pairs connected to the bump(soo') specified.
- instConn: reports clamp instances that have missing connections to the P/G grid.
- isolatedBump: reports bumps that do not connect to any clamp
- cell <cell\_name>: reports information on specified clamp cell(s)
- celltype <type>: reports information on clamp cells of a particular clamp type
- inst <inst\_name>: reports information on specified clamp instance(s).
- volt <voltage>: reports a list of clamps connected to a particular node voltage.
- net <net\_name>: reports information on all clamp nodes connected to specified net(s), both directions.
- netConn: checks connectivity of each net pair (both directions), so for “-netConn {netX netY}”, the connectivity of both netX->netY and netY->netX is checked. If “-netConn netX” is specified, the connectivity from netX to all other nets and from all other nets to netX is checked. For each net pair reported, a symbol “->” or “-<” is added between the two nets to indicate the connectivity direction between them.
- allNetConn: reports clamp connectivity to all nets and domains, in both directions
- bumpConn: reports clamp connectivity to all bump connectivity from the specified bump
- allBumpConn: reports clamp connectivity of all bumps
- loop: lists the clamp instances of all B2B loops for queried net pairs.

-loopLength/ -b2bLoopLength : sets the EXACT stage number(s) to be reported for queried net pairs. Note that “-esdStage” sets the RANGE of stage numbers, while “-loopLength” sets the EXACT stage numbers. -loopLength can specify more than 2 numbers.

-roff: sets the resistance threshold value at which clamp devices are considered “OFF”. The default threshold value is 1e6 Ohms.

-rptDisConn: reports net-pairs with no clamps between. Note that when the following command options are used:

```
perform clampcheck -allNetConn -rptDisConn
then
```

- clamps with only one pin are reported as unconnected.
- clamp pins not specified in ESD\_PIN\_PAIR are reported.

-rule: specifies the name of the rule file for checking. Multiple rule files can be represented in two formats: {rule1 rule2 ...}, or rule1,rule2, ... (no spaces in this form).

-esdStage: specifies the number of clamp stages for net/domain pair checks

-detail: reports detailed information on net/domain pair.

-append: appends results to existing results in the output file

-summary: generates a bump/clamp summary text report, with format as follows:

For the bump summary, the format is:

```
<Net> <Bump#> <Pass#> <Fail#> <OutRadius#> <Isolated#>
```

For the clamp summary, the format is:

```
<Net> <Pin#> <Connected#> <Unconnected#>
```

- c. ‘perform cmmcheck’ provides information on a particular CMM database and CMM cells. The command can be invoked at any time. If the GSR is read in the ‘-cell’ option can be used, otherwise, the full model path using the ‘-path’ option must be specified to query CMM information:

```
perform cmmcheck ? -version? ?-o <output_filename>?
?-path <cmm_db_path>? ?-statistics? ?-cell <cell_name>?
```

- d. ‘perform cpa thermal’ executes thermal simulation, with the following steps:

- Saves project files to <project\_path>.
- Performs DRC checking.
- Generates a package thermal profile.

- e. ‘perform emcheck’ can be used to perform power EM analysis for specific modes and nets from the command window, using the command:

```
perform emcheck ?-mode [AVG | RMS | PEAK | all]?
?-net [<net_name> | <net_filename>]?
?-skipcmm <cell_list> ?
```

where

-mode : specifies the type of emcheck to run

-net : specifies either a net name or a file listing one or more nets to be checked. The file format accepts one net name per line; leading and trailing white spaces in the file are ignored. Sample contents of a nets file:

```
Net1
```

```
Net2
```

```
...
```

-skipcmm <cell.list>: skips EM checks for specified or all CMM blocks to avoid false EM violations on CMM block geometries. <cell\_list> specifies one CMM cell name per line. If a <cell\_list> is not provided, emcheck skips all CMM blocks in the design. If you have already performed emcheck on the whole design, or on a CMM block, and later use the

“perform emcheck –skipcmm” command, EM still retains the old data, but the text reports are updated.

Results are dumped into the file *adsRpt/SignalEM/*.

Note that each new report of ‘perform emcheck’ results overwrites the previous perform command results.

Skip the -mode and -net options to perform analysis for all modes and all nets in the design.

- f. ‘perform esdcheck’ invokes the ESD checking command, with syntax

```
perform esdcheck -rule <rules_file>
 -clamp <clamp_file1> <clamp_file2> ...
? -statusCheck <timeInterval>?
? -ruleType <rule_type1> <rule_type2> ...?
? -ruleName <rule_name> ? ? -thread <num_threads>?
? -optimize <mode>? ? -detail ? ? -append ? ? -excel?
? -outDir <results_dir> ? ? -ignoreError ?
? -saveDBDir <path> ? ? -jobCount <num_jobs> ?
? -collate ? ? -jobFile <job_defin_file> ? ?-incremental ?
?-from {<x> <y> <layer><net>} -to {<x> <y> <layer> <net>}
[{-zapI <I> | -zapV <V> -zapR <R>}]?
? -extendClampConn ?
```

where

- rule <rules\_file> : specifies the name of the rules file containing the rules to be checked. At least one rule must be specified. If the ESD check for a rule has an error, the default behavior is to stop execution. The ‘-ignoreError’ option allows checking to continue past any errors, if possible, and attempt to finish all rules in the rule file.
  - clamp <clamp\_filename> : specifies the clamp filename. Clamp statements can be combined in the rule file or separately defined in a clamp file. See the Clamp file section for a description of the clamp file syntax and an example clamp file.
  - statusCheck: dumps out a progress report of the run. By default the report is dumped into the file *adsRpt/ESD/esd\_progress.rpt*. The unit for <timeInterval> is hours (can be a fraction), which works with or without the ‘-jobCount’ option. When -jobCount is specified, *adsRpt/ESD/esd\_progress.rpt* reports the status for all jobs, and the file *adsESD<N>/adsRpt/ESD/esd\_progress.rpt* reports the sub-jobs.
  - ruleType <rule\_type> : checks all rules that match the specified type. This allows flexibility to construct a large “template” file with many rules defined in it. Then when only -type is specified in the rule file, all rules that match the specified type are executed.
  - ruleName <rule\_name>: performs checks for the user-specified rule name. When only ‘-ruleName’ is specified, all rules that match the specified name are executed (regardless of the type). When both ‘-type’ and ‘-name’ are specified, the rules in the rule file that match both the specified rule type and rule name are executed. When ESD rule file has multiple CD rules, if ‘-type cd’ is given along with ‘-name <rule1>, <rule2> ...’, the rule names <rule1>, <rule2> etc of ESD checks are submitted in sequence for prioritizing the execution. Wildcard(\*) in rule names for ‘-name’ option is supported.
  - thread <num\_threads>: specifies the maximum number of threads/processes to be used, to speed up long simulation resistance calculations. All available CPUs may be used to support the multiple threads unless ‘-mcore’ is specified. Default: 2.
  - optimize <mode>: as the mode specified increases from 0 to 3, more optimization is done, for greater performance gains.
- Mode values 0 and 1 are only recommended to work around issues that have caused resistance computation to fail:
- 0: all optimization off

1: performs only basic netlist optimizations (up to 2X performance gain)

Mode values 2 and 3 are recommended for normal use.

2: performs aggressive netlist optimization for greater performance gain (up to 20X possible). This is the default.

3. In addition to optimization of mode 2, performs additional optimization for greater coverage (can double the performance as with 2). This setting would be effective only for C2I, C2M, and B2I resistance checks.

**-detail** : provides detailed reports on B2B and multi-stage checks, including data on x,y locations, netnames, metal layers and the cell names of the clamp instances, so you do not have to refer to any other files to get details.

**-append** : appends results to the output files, instead of overwriting the previous output file

**-excel** : creates a consolidated ESD report file in Excel format, *adsRpt/ESD/esd\_excel.rpt*, for resistance and CD checks, so that the information can be easily sorted and processed using Excel.

**-outDir <results\_dir>**: specifies the directory for the results files. Default: *adsRpt*.

**-ignoreError** : where the <rules\_file> contains multiple rules to be checked, if there are data issues or error conditions such that the ESD check for a rule has an error, the default behavior is to stop execution. '**-ignoreError**' allows checking to continue past any errors, if possible, and attempt to finish all rules in the rule file.

**-saveDBDir**: saves the DB into the specified directory. You can then execute esdcheck in the same run used for exporting the DB.

**-jobCount**: specifies the maximum number of jobs to execute at a time, particularly when the rule file has multiple rules in it, or when an ESD current density check rule has many 'from/to' point pairs to be analyzed. This option partitions the rule check into multiple sub-tasks, which are executed in parallel when the host machine has multiple CPUs. One job is performed by the parent PF session and the rest of the jobs are executed in different sessions in *adsESD1*, *adsESD2*, ... as the working directory.

The recommended usage is to start the session by importing a previously- exported DB, then to run ESD checking with the **-jobCount** option specified.

This option provides a status report on CD checking, including rule name, rule type and the number of tests, in the *esd\_info.rpt* file. Example file contents:

```
ESD Job Assignment Report
#<JOB> <WORKING_DIR> <RULE_NAME> <RULE_TYPE> <NUM_TESTS>
Job1 ./adsESD1 C2I_RULE CLAMP2INST 1
Job1 ./adsESD1 G2S_RULE BUMP2BUMP 1
Job1 ./adsESD1 SIGDC CURRENT_DENSITY 16 (shotgun)
Job2 ./adsESD2 C2M_RULE CLAMP2MACRO 1
Job2 ./adsESD2 S2P_RULE BUMP2BUMP 1
Job2 ./adsESD2 GPDC CURRENT_DENSITY 25
```

**-collate**: concatenates ESD text reports for the esdcheck jobs specified with **-jobCount** into the directory specified by **-outDir** option or the default master *adsRpt/ESD*.

**-jobFile** : specifies the job definition rules file

**-incremental** : provides incremental resistance check coverage when you have already run with one radius and want to rerun with an increased radius (uses previous B2C results). default 0.

#### Options for current density checking:

**-from / -to** : specifies the x,y coordinates, the layer and net for from and to zapping. Finds the closest nodes that match the specified points. Note that other specialized commands are also available for “-from/-to” checking:

```
-fromBump <bumpName> -toBump <bumpName>
-fromClamp {<instName> <locId>} -toClamp {<instName> <locId>}
-fromInst {<instName> <netName>} -toInst {<instName> <netName>}
-fromInst {<instName> <pinName>} -toInst {<instName> <pinName>}
```

Note: For the specified instance, a node in the specified net connected to the pins of the instance is selected. More than one "-from/-to" check can be specified, where all "-from" points are shorted and connected to the positive terminal of the zapping source, and all the "-to" points are connected to the ground terminal of the zapping source. All points to be shorted should be in the same net (or connected by power gating switches), and also physically connected in the layout. Otherwise, checking is not performed.

```
-zapI <I> : zapping current (default 1A)
-zapV <V> -zapR <R> : zapping voltage and zapping source resistance. (The default is to use '-zapI')
-extendClampConn: uses the discharge paths through clamp devices, even for cases when from/to zap points are part of same net.
```

- g. 'perform extraction' builds network connectivity for the specified type of net and extracts RLC values as specified. Default - build power network and extract resistance. Extraction for a particular type of network is not repeated even if specified.

```
perform extraction [-power | -ground | -clock] ?-spice? ?-l? ?-c?
? -nets <nets_file>?
```

where

- power : builds power network connectivity.
- ground : builds ground network connectivity.
- clock : builds network containing only clock nets
- spice : builds power/ground network connectivity for clock tree analysis.
- l : extracts inductance for specified nets.
- c : extracts capacitance for specified nets.
- nets: specifies a file that defines a limited list of nets to be extracted, in both power EM and signal EM flows. Can be used in a hierarchical flow in which you want to analyze the top level nets only, without sub-block level nets.

- h. 'perform gridcheck' generates a list of the relative power and ground circuit resistance for every instance (sorted by decreasing total relative resistance), or for standard cells and macro blocks separately, allowing an assessment of relative grid weakness for all nodes.

```
perform gridcheck ? -o <output_filename>? ?-limit <max_lines>?
? -perArc ? ? -stdcell [ave | min | max | all | none]?
? -macro [ave | min | max | all | none]? ? -package ?
? -lumpSwitch? [-allDie | -die <die name>] ?-append? ?-force ?
? -box <bbox> ? ? -printPin ?
? -excludeCellFile <file_Name> ? ? -excludeInstFile <file_Name> ?
? -excludeDecap ? ? -ignoreInterPloc ?
```

where

- o <output\_filename> : specifies output filename. Default name: *adsRpt/apache.gridcheck*
- ignore\_floating : unconnected pin instances are not included in the output P/G Weakness report
- limit <max\_lines> : if the '-perArc' option is not selected, the higher resistance arcs are reported up to the limit and lower resistance arcs have fewer reports than the limit. If -perArc is specified, the maximum number of instances are reported for all P/G arcs.
- Default : 5000 per arc

-perArc : gridcheck reports about the same number of instances for each P/G arc in the design, rather than reporting more P/G arcs with higher resistances.

-stdcell / -macro : selects the type of node resistance report, for standard cells or macro blocks [ ave | min | max | all | none ]: 'ave' is the average resistance for all nodes in each instance selected, and is the default. For the 'min' and 'max' options, the node with the minimum or the maximum resistance value for the instance is reported, instead of the average of all nodes in the instance. The 'none' option is used to eliminate reporting on all standard cell or macro instances. For 'all', the largest 5000 node resistances in the design are reported.

-package : includes package resistance in the resistance calculation

-lumpSwitch' : reports the lumped resistance of multi-finger switches, instead of single pin pair resistance

-all\_die : runs gridcheck for all dies in a 3D-IC design

-die : specifies the die name

-append : appends new results to the previous output file

-box <bbox> : reports the P/G arc-based instance gridcheck resistances in the same format as 'perform gridcheck'. Use the syntax  
 perform gridcheck -box <bbox> -fullchip'  
 to report the gridcheck resistances for all nodes in the box.

-printPin : reports instance-specific power and ground pins in 'apache.gridcheck' file.

-force : reruns gridcheck from scratch, rather than using previous results

-ignoreInterPloc : Shorts the internal pads and extends the tracing to external pads.

- i. 'perform jitter' executes the PJX jitter analysis, either in high-capacity fullchip level ATE-based jitter analysis, or using sign-off accurate Spice-based jitter analysis.

```
perform jitter -fullchip -signoff -config <config_file>
? -mode [effvdd | ideal]?
```

If CLOCK\_SOURCES specification is not present in *psiw.cfg*, then fullchip jitter analysis is performed, if it was not already done, and then Sign-Off analysis is performed for the set of clocks with the worst jitter, and the leafs with the worst jitter. The 'ideal' option enables jitter simulation with ideal P/G waveforms (default is effective voltage).

- j. 'perform min\_res\_path' allows you to trace the minimum resistance path (SPT - "shortest path tracing") between P/G pins of an instance or region and the pad connected to it, to identify weakly-connected instances, or between two specified nodes. Selects the node with the Worst Voltage Drop as the starting point for minimum resistance path tracing. Can be run prior to simulation if extraction has been completed. Note that in this case the IR drop column in the output report is blank, as IR analysis has not been performed. Reports voltage drop data on wires, vias and stacked vias in the report file *adsRpt/res\_path.rpt*. The command also highlights the path in the GUI and creates a resistance bottleneck report that summarizes voltage drops and resistance along different segments of the path.

```
perform min_res_path -inst [<inst_name> |<region_name>]
?-pin <pin name> ? ? -pad {<padname1> <padname2> ... } ?
? -clear? ?-fromClamp? ? -logicPin ? ? -viaInst <inst> ?
? -viaSwitch <switch_name>? ? -lumpSwitch?
? -from {<x1 y1> <layerName1> ?<netName>? }
-to {<x2 y2> <layerName2> ?<netName>?} ? -o <*.rpt>?
? -addNode ? ? -width ? ? -total_res ? ? -total_vol ?
? -simplified_report ? ? -append <filename> ?
? -esdFail <esd_type> <N>? ? -multipath [?-gui?]?
? -nodeSel [minres |maxres | wstdrop]? -consolidatedR ?
? -set effR [On|Off]? ? -effR ?
```

```
? -set nodeSel [minres |maxres | wstdrop]?
? -set fromClamp ?
```

where

- inst : specifies the instance name or the block power assignment region name of interest. The minimum resistance paths are highlighted, connecting power pads and ground pads from the P/G pins of the specified instance or region
- pin: performs path tracing only for the specified pin of the instance. Other P/G pins are ignored in the GUI highlighting and text report.
- pad: specifies the pad or pads to which minimum resistance paths are found and highlighted from the P/G pins of the instance
- clear: clears all GUI highlights
- fromClamp : all path tracings are from nodes until -fromClamp turned off
- logicPin : reports the minimum resistance path to selected pin geometries of each of the logic pins. If two different pins are connected to the same net, it reports a path for each of the pins. The selection of pin geometry for the logic pin is determined by the “-nodeSel” option
- viaInst <: allows reporting SPT paths through a user-specified instance to the bumps, or from <inst1> through <inst2>, and then to the bumps, using syntax such as:  
`perform min_res_path -inst <inst1> -viaInst <inst2>`  
or reports SPT paths using the 'to' option, with syntax such as  
`perform min_res_path -to <x,y,layer,net> -viaInst <inst2>`
- viaSwitch : traces minimum resistance paths through user-specified switch instance.
- lumpSwitch : reports the lumped resistance of multi-finger switches, instead of single pin pair resistance
- from/-to : allows tracing a resistance path from one node at or near <x1 y1> on layer <layerName1> to another node at or near <x2 y2> on layer <layerName2> (“point-to-point SPT”). Specifying the associated net names is optional. The minimum resistance path connecting nodes at <x1 y1> and <x2 y2> is identified and highlighted. The output file gives the physical length of the calculated least-resistance path, the resistance value and the voltage drop for each segment of the path.
- o: redirects the output to the file specified, instead of the default file *adsRpt/res\_path.rpt*
- addNode : creates a node for accurate resistance calculation. When specified, together with “-from x,y,layer,net” and/or “-to x,y,layer,net” (layer/net is optional, but highly recommended), a node is created at that location provided there is no node within a 1uM radius; and there are wires at the specified location. Note the following in using this feature:
  - The node is created at the center of the wire, so it may not be the exact coordinate specified. You can also use option '-searchDist <u>' to change the 1uM (default) maximum radius as desired.
  - When a new node is created in this way, all previous SPT/gridcheck/res\_calc/ PF-S db data become invalid and are removed.
- width : an additional column, Width, is displayed in the output report, showing the width of the metal wire segments.
- total\_res: reports the total resistance of the minimum resistance path
- total\_vol: reports the total voltage drop of the minimum resistance path
- simplified\_report: specifies that a smaller simplified report file is to be generated besides the main report file. In the simplified file, each line represents one path, in the following format:  
`# instance masterCell pin net totalLength(um) totalResistance(Ohm)`  
The generated simplified file filename has a “\_Simplified” suffix.

- append : the results of successive SPT runs are appended to the specified file; if it does not exist, RedHawk creates one. The default filename is *adsRpt/res\_path.rpt*. If specified, this must be the last option in ‘perform min\_res\_path’ command.
- esdFail <esd\_type> <N>: generates an SPT report for the top 'N' violations in ESD checking, for the specified rule type B2B, B2C, B2I, C2I, or C2M.
- multipath : the top N shortest paths to all pads are dumped into the file *adsRpt/multipath.rpt*.
- multipath -gui : the top N shortest paths to all pads are displayed on a multipath output page dialog. The multipath dialog allows users to sort the results by the value of each column and also to select either single line or multiple lines for display.
- nodeSel : specifies which node (pin port) is the first on which to do path tracing inside an instance, where
  - minres - selects the node with the minimum grid check resistance as the first node (default)
  - maxres - selects the node with the maximum grid check resistance as the first node.
  - wstdrop - selects the node with worst voltage drop as the first node, if simulation is completed.
- effR : reports effective resistance of nodes.
- consolidatedR : To perform more accurate min res path tracing at wide wire junctions where the R-network is made of multiple fanout connections with higher resistances. Can be used only if “SAVE\_CONSOLIDATED\_R 1” in set in GSR
- set effR [On | Off] : when On, sets the default value of effR for the session, calculating the effective resistance for all nodes (or until turned Off), particularly when using the GUI.
- set nodeSel <options> : when set, sets the default option value of nodeSel for the session, or until reset.
- set fromClamp : all path tracings are from nodes until fromClamp turned off

k. ‘perform pwrcalc’ executes power calculation for generic (default), static, or dynamic analysis.

```
perform pwrcalc ?[-static | -dynamic]?
```

l. ‘perform powermodel’

creates a chip power model for system power integrity design.

```
perform powermodel [-wirebond | <flip chip partitions>| -cdie| -static
| -lowpower | -esd] ? -esd_clamp <file> ? ? -parasitic ? ?-vcd ?
?<no model option-default>? ? -pincurrent ? ? -rleak ? ? -rleak_par?
?-solver mor? ?-plocname? ? -ind? ? -no_afs? ? -passive ?
? [-noglobal_gnd | -global_gnd] ? ? -high_capacity 1 ?
? -repeat_current [<start_time> | presim | best]? -probe ?
? -internal_node -cell_file <cell_list_filename> ? ? -reportcap ?
? -o <output_filename> ? ? -reuse ? ? -io ?
```

where

-wirebond : specifies a wirebond package

<flip chip partitions> : -nx <num\_x\_partitions> -ny <num\_y\_partitions>, specifies the number of partitions in the x and y directions. For -nx 1, -ny 1, a Cdie/Rdie report is generated in the *adsRpt/CPM/apache.Cdie* file.

-cdie : sets nx=1 and ny=1, and connects all power nets together and all ground nets together to obtain a single-port solution to obtain the equivalent Cdie and Rdie values for the chip. No current waveform is generated. A Cdie/Rdie report is generated in the *adsRpt/CPM/apache.Cdie* file.

Note that the -cdie option just provides faster run time by not calculating the current waveforms. For all other purposes it is equivalent to using ‘-nx1 -ny1’.

- static : creates static analysis chip power model, using DC conditions, a resistance-based circuit, and average current.
- lowpower: CPM generation support for ramp-up analysis.
- esd: creates a CPM that includes clamp device models/characteristics, along with the PDN model Spice deck. Clamp connections are modeled as ports in the CPM model, and subcircuits for clamp devices are included in the CPM model. You must also specify the clamp files in the GSR. If the clamp files are not present in the GSR, use the “–esd\_clamp” option to specify the clamp file.
- parasitic: generates only the passive part of the CPM, without performing transient simulation, to generate the current signatures of the CPM ports, an extension of -cdie option for multi-partition CPMs. This can save time in transient simulation. However, the CPM model generated with -parasitic can only be used for DC and AC analysis (not usable for transient analysis).
- vcd : uses VCD file as basis for determining the worst case switching scenario
- pincurrent: specifies that CPM generate the model without current conservation (balanced current between Vdd and Vss) to achieve better correlation with **RedHawk** dynamic simulation results. In general, CPM enforces current conservation. However, in cases when **RedHawk** does not produce balanced VDD and VSS currents, this option can be used.
- leak: causes the leak resistance to be added between ports on the VDD (power) net and the reference port on the VSS net.
- leak\_par: inserts leakage resistance between the VDD and VSS ports of each defined partition. Note that this option only works with partitioned CPM models. See the following section for more details on usage.
- solver mor: turns off the default ‘solver ac’ function, which is an accurate frequency-based linear solver AC solution with passivity enforcement.
- plocname: specifies pad/group names for CPM port names. If the ‘-wirebond’ option is used, the subcircuit terminal names are taken from the pad names (if no grouping is specified), or the group names from the 6th column of the .ploc file. These group names are also known as SPICE node names, as this mechanism is used to connect a package using the keyword ‘PACKAGE\_SPICE\_SUBCKT’. If this option is used with ‘-nx # -ny #' options, the node names are generated in the following form: PAR\_0\_0\_VDD1, PAR\_0\_0\_VSS2, ... For wirebond CPMs without any grouping, the subckt terminal name is the ploc name.
- ind: accounts for on-chip inductance, if the option ‘-l’ of the **RedHawk** ‘perform extraction’ command has been used

**Note:** If you specify the ‘-l’ option of the ‘perform extraction’ command, but not ‘-ind’, inductance is ignored. Not specifying the ‘-l’ option and using ‘-ind’ is an error.

- no\_afs : turns off the default AFS function. Adaptive Frequency Sweep for AC mode execution intelligently selects seven to nine frequencies (enough to reach convergence) to perform AC analysis, as opposed to 26 frequency samples that are performed by default in ‘solver ac’ mode. This option is recommended for design sizes exceeding 50 ports. Port count can be determined by computing  $N * M * P$ , where N = number of X partitions in the CPM (-nx option), M = number of Y partitions in the CPM (-ny option), and P = number of power and ground domains.
- passive : only effective with the MOR function, which uses MOR to generate a passivity-enforced **SPICE** model, which can reduce accuracy. This mode can be useful if the **SPICE** simulation of the package/PCB CPM has convergence issues. The **SPICE** netlist is significantly smaller than using the default mode. But CPM is a compact model, so complexity is not a concern for cases with, for example, 100 bond pads, or up to 10x10 partitions for a flip-chip design.
- noglobal\_gnd | -global\_gnd: specifies the type of parasitic modeling in CPM, either (a) without Spice Node 0, using option ‘-noglobal\_gnd’ (the default), where there is a direct

connection between the power and ground ports without going through Spice node 0, or (b) using Spice Node 0, using option '-global\_gnd', in which the RLC parasitics from power and ground are connected to Spice global ground (node 0).

- high\_capacity: specifies use of the ASIM3D solver for CPM generation (default)
- repeat\_current: specifies that the CPM current signature is repeated starting from the specified time point. Either a <start\_time> in ns, the 'presim' time, or 'best' (chosen to cause the best continuity at the repeat point), can be chosen as the starting point of the repeating waveform. A warning is issued for incorrect values (such as a negative value or value greater than the maximum time of the PWL source). For non-zero values the closest time in the PWL definition is used. For example, if a '-repeat\_current 1n' option is specified, and PWL time values ..., 900, 930, 960, 990, 1020, ... ps are defined, 'R=990 ps' is used. This is required for **SPICE** to accept the netlist. For the 'best' option to work well, the presim time and the transient simulation time need to be set to "n\*T", where n is a positive integer, and T is the period of the clock frequency. With this option, the repeat time may be different for each individual PWL current source. Usage examples:

```
perform powermodel -nx 2 -ny 2 -repeat_current 0
```

which repeats starting from the beginning, t=0. Or,

```
perform powermodel -nx 2 -ny 2 -repeat_current 2n
```

which repeats starting from the time value in the PWL source definition closest to the 2ns time specified.

- probe: invokes the iCPM utility that enables the visibility of sensitive P/G connections in the design, and allows you to probe device locations inside the chip. You must set the PROBE\_NODE\_FILE GSR keyword, as described in [section "iCPM- Internal Node Probing", page 14-405](#).

- internal\_node: specifies additional ports located at P/ G pins on the same net that are to be shorted together to form one internal port. These nodes are named with the format '<instance name>\_<netname>' in the CPM, such as "inst\_1\_VDD". Note that the option "-internal\_node" is not supported in static analysis.

Note that the options '-internal\_node' and '-cell\_file' are required to execute this feature, and the '-pincurrent' option should be used to keep the CPM currents at the correct value without further modifying the port currents, so that the sum of all port currents is zero. This feature also allows you to include/exclude the instance current profile and device capacitance from CPM creation, using the EXCLUDE option in the GSC file. To exclude instance current profiles and device capacitance, use the GSC syntax: '<instance name> EXCLUDE'.

- cell\_file <cell\_list\_filename>: specifies a file containing a list of the instances whose internal P/G pins are to be exposed.

- reportcap: creates a log file report of all capacitance components included in the CPM generation. Example output:

```
Capacitance components -
 Intentional Decap - 0.000000e+00 pF
 Intrinsic Decap - 2.404236e+02 pF
 Load Decap - 1.945542e+02 pF
 Power grid Decap - 3.260729e+00 pF
 Well Decap - 0.000000e+00 pF
 Total - 4.382385e+02 pF
```

- o <output\_filename> : specifies output filename (default - *PowerModel.sp*, with the passive part in the file *PowerModel.sp.inc*)

- reuse : allows reuse of the generated current waveforms after one CPM run, if chip power models with different partitioning schemes are desired

- io : specifies that the I/O cells are to be included in the chip power model

- m. 'perform res\_calc' calculates the effective P/G grid resistance from all pads to selected instances, nets or locations, and provides an absolute resistance value in Ohms (whereas 'perform gridcheck' provides the minimum path resistance normalized to the maximum path resistance between points). The default invocation, without any options, creates a resistance report for the 1000 worst instances, as indicated by quick estimation. The syntax is:

```
perform res_calc
? [-instance <name_list>]? ?[-instFile <filename>]?
? [-cell <name_list>]? ?-cellFile <cell_filename>?
? -box <llx lly urx ury>? ? -net <name_list> ? ?-layer <name_list>?
? -thread <num>? ? -mcore <num>? ?-all_point? ? -loopmode ?
? -incremental? ?-append? ?-verbose? ?-limit <num_lines>? ?-o <file>?
? -constrFile <res_Const_file> ? ? -optimize <mode> ?
? -package ?
? -excludeCell <cell_Name(s)> ? ? -excludeInst <inst_Name(s)> ?
? -excludeCellFile <file_Name> ? ? -excludeInstFile <file_Name> ?
? -excludeDecap ?
```

**Note: the following additional options are for MMX designs:**

```
? -xtor <name_list>? ?-pin <name_list>? ?-allPin? ? -guardring ?
```

**Note: the following additional options require a PathFinder license:**

```
-from {<pt1>} -from {<pt2>} ... -to {<pt3>} -to {<pt4>} ...
-shortAll ? -addNode ?
-fromFile <file1> -toFile <file2> [-pairwise |-straightPair]
-fullchip
```

where

-instance : displays a report of the worst resistance location for the specified instance(s). The report contains worst case resistance for one location per domain per instance.

<name\_list> : can be represented in two formats:

{<name1> <name2> ... }, or <name1>,<name2>, ... (no spaces in this form)

-instFile : same as the -instance option, except that you can specify a file containing a list of instances. The report contains worst case resistance for one location per domain per instance.

-cell : creates a resistance report of the worst resistance locations of all instance(s) for the specified cell masters. The report has one location per domain for each instance.

-cellFile : specifies a text file that contains the cell names, one per line, for res\_calc to compute the equivalent grid resistances.

-box <llx lly urx ury> : creates a resistance report of the points within the box specified. By default, the report has a maximum 1000 locations within the defined box.

-net : specifies a list of nets for performing res\_calc

-layer : reports the resistance values for specified layers.

-thread: specifies the number of threads/processes to be used for speeding up the calculation.  
 All available CPUs may be used to support the multiple threads unless -mcore is specified.

-mcore: specifies the number of CPUs to be used for each process

-allPoint: specifies that calculations are done for all points on specified instances

-loopmode : enables instance VDD + VSS resistance reporting. Note that when -loopmode is used, the value for '-limit' specifies the maximum number of instances, not nodes. A sample output report follows:

```
Resistances from all pads to the listed points
```

```
LOOP_R VDD_R GND_R VDD(X Y LAYER NET) GND(X Y LAYER NET) INSTANCE
80.9858 67.3984 13.5873 (4459.77 443.855 METAL3 VDD) (2095.97 561.905 METAL3
 VSS) inst_129747/adsU1
4.41046 2.19395 2.21651 (4600.23 764.24 METALL1 VDD) 4600 767.93 METALL1 VSS)
 inst_509611
3.86578 1.86783 1.99796 (843.18 911.84 METALL1 VDD) 843.18 908.15 METALL1 VSS)
 inst_129995
2.59412 1.34763 1.24649 (2323.54 4074.99 METALL1 VDD (2321.08 4078.68 METALL1
 VSS) inst_129228/inst_376373
2.58131 1.22812 1.35319 (2900.5 2230.53 METAL3 VDD) 2898.08 2204.01 METAL3
 VSS) inst_129424/inst_92357
 -incremental : checks the new res_calc database and computes the 'next N worst' instances,
 based on the gridcheck report.
 -append: appends results to the output file.
 -verbose: displays the full resistance report in the log window.
 -limit <num_lines>: specifies a maximum number of lines in the resistance report. Default is
 1000. When -1 is specified, the program determines the limit based on the size of the
 design.
 -o <file>: saves the report to specified output file (default - adsRpt/<design_name>.res_calc).
 The general output format is shown below:
Ohms Location <x y> Layer Net Pin Instance
2.58129 3316.37 3948.71 METALL1 VSS VSS instance1
 -constrFile : specifies a file defining bump-to-instance pin resistance checks to be performed
 based on user-specified constraints . The constraint file format is shown below:
```

---

```
IP_RESISTANCE
 CHECKBUMP [RES_TOP|RES_BOT|DIS_TOP|DIS_BOT | SHORT |
 ALL | RES_MAX | RES_MIN | DIS_MAX | DIS_MIN] <num_bumps>
 CELL <cell_name>
 PIN <Pin_name> <Res_threshold> ?<x> <y> ?<layer>? ?
 ...
 END CELL
 INSTANCE <inst_name>
 PIN <Pin_name> <Res_threshold> ?<x> <y> ?<layer>? ?
 ...
 END INSTANCE
 END IP_RESISTANCE
```

where CHECKBUMP specifies the type of check and the number of bumps to be checked, which applies to all pins in the constraint file, and:

RES\_TOP - checks nodes with the highest resistance, up to <num\_bumps>  
 RES\_BOT - checks nodes with the lowest resistance, up to <num\_bumps>  
 DIS\_TOP - checks nodes at longest distance, up to <num\_bumps>  
 DIS\_BOT - checks nodes at shortest distance, up to <num\_bumps>  
 SHORT: checks and reports the effective resistance from IP pins to all associated bumps shorted for calculation purposes.  
 ALL: checks and reports point-to-point effective resistance from each pin to all associated bumps.  
 RES\_MAX : checks and reports the maximum point-to-point effective resistance among all pin-bump pairs.

**RES\_MIN** : checks and reports the minimum point-to-point effective resistance among all pin-bump pairs.

**DIS\_MAX** : checks and reports the maximum geometric distance among all pin-bump pairs.

**DIS\_MIN** : checks and reports the minimum geometric distance among all pin-bump pairs.

---

**-optimize <mode>**: as the mode specified increases from 0 to 3, more optimization is done, for greater performance gains.

Mode values 0 and 1 are only recommended to work around issues that have caused resistance computation to fail:

0: all optimization off

1: performs only basic netlist optimizations (up to 2X performance gain)

Mode values 2 and 3 are recommended for normal use:

2: performs aggressive netlist optimization for greater performance gain (up to 20X possible). This is the default.

3. In addition to optimization of mode 2, performs additional optimization for greater coverage (can double the performance as with 2).

**-package** : includes package resistance in the resistance calculation

**-excludeCell <cell\_Name(s)>** : excludes specified cells from effective resistance computation.

**-excludeInst <inst\_Name(s)>** : excludes specified instances from effective resistance computation

**-excludeCellFile <file\_Name>** : excludes specified cells in the Cell file from effective resistance computation

**-excludeInstFile <file\_Name>** : excludes specified instances in the instance file from effective resistance computation

**-excludeDecap** : excludes decaps from effective resistance computation

**-xtor**: calculates resistance for points on pins in transistors specified. Names should be separated by commas. (for MMX instances only)

**-pin**: calculates resistance for points on specified pins. Names should be separated by commas.

**-allPin**: calculates resistance for all pins on all MMX instances

**-guardring**: performs resistance calculation on the substrate guard ring

**-from {<ptn>}/ -to {<ptm>}** : performs resistance calculation from one or more nodes at or near <ptn> to one or more other nodes at or near <ptm>. The syntax for specifying <ptn> is { <xn> <yn> ?<layer>? ?<net>? }. Specifying the associated layer name or net name is optional.

**-addNode** : creates a node for accurate resistance calculation. When specified, together with “-from x,y,layer,net” and/or “-to x,y,layer,net” (layer/net is optional, but highly recommended), a node is created at that location provided there is no node within a 1uM radius; and there are wires at the specified location. Note the following in using this feature:

- The node is created at the center of the wire, so it may not be the exact coordinate specified. You can also use option '-searchDist <u>' to change the 1uM (default) maximum radius as desired.
- When a new node is created in this way, all previous SPT/gridcheck/res\_calc/ PF-S db data become invalid and are removed.

**-shortAll**: shorts all points specified by a '-to option

-fromFile/-toFile: performs resistance calculation between points listed in two files in one of two ways: “pairwise” calculates p2p resistances for all pairs of points in the two files, while the “straightPair” option only calculates and reports p2p resistances for “straight pairs”— that is, between the 1st point in the -fromFile and the 1st point in the -toFile, and between the 2nd points in each file, and so on. One requirement of the ‘straightPair’ option is that the -fromFile and -toFile must have the same number of points. Also, the content of the specified files has the following format:

```
<x> <y> <layer> <net>
...

```

where <layer> and <net> are optional, but highly recommended to reduce unneeded data.

-fullchip: specifies that effective resistances for all nodes in the design, including those for wires/vias (not just the pin nodes of the instances), are to be computed and reported.

- n. The ‘perform thermalmodel’ command creates a thermal model of the chip as specified:

```
perform thermalmodel -layer ?-self_heat? ?-inst?
?-o <output_file>? ?-d <Thermal dir> ? ? -ctmfilecheck ? ? -mt ?
```

where

- layer: uses layer-specific temperatures from the thermal profile.
- self\_heat: distributes power/ground heat to individual layers. If -self\_heat is not specified, all power is assigned to the device layer.
- inst : creates an *ins\_power\_loc* file that shows which instances are under the hot spots outside RedHawk. Including this file in CTM generation enables this feature in Sentinel-TI package analysis.
- d : a <Thermal dir>.tar.gz file is generated in the thermal directory. If no -d <Thermal dir> is specified, an *adsThermal.tar.gz* file is generated in addition to the *adsThermal* directory
- ctmfilecheck : generates the text format ctm file 'adsThermal/chip.ctm.txt'.
- mt : enables multi\_threading mode. With MULTI\_THREAD keyword in GSR, we can control the number of temperatures to be put in one parallel batch. If the keyword is not specified, thread number will be set to CPU core number.

## pfs

---

### **pfs [ add | delete | export | import | show ] ? <options>?**

The ‘pfs’ command provides a number of functions to support ESD checking operations.

- a. The ‘pfs add clamp\_pin’ command creates shorted parallel clamp pin regions. Note that extraction must be run after adding or deleting shorted regions for the actions to take place.

```
pfs add clamp_pin <clamp_cell_name> ?-pin {<pin1> ... }?
?-pinpair {<pin_name1> <pin_name2>}? ? -type <tname>? -layer {<layer1>
... }?
?-region <x1> <y1> <x2> <y2>? ? -locID <loc_ID>? ? -inst <instName> ?
?-margin <distance(um)> ? -includeAll [1|2]?
?-finX/-finY ?<num_fingers>? ? ?-par <partition_size>?
?-dist <um> ? > ? -multiPair ?
?-fwdR <forward_res> -bwdR <backward_res>?
?-perFwdR <per_pair_fwdR> -perBwdR <per_pair_bkwdR>?
?-ivname <iv_name>?
```

where

- clamp\_pin <cell\_names> : specifies cell names for shorting
- pin <pin\_name> : specifies pin names for shorting
- pinpair <pinpair\_name> : specifies pin pair names for shorting.
- type <tname> : specifies clamp type names for shorting

- layer <layer\_names> : specifies the layer name of the pins for shorting
- region <x1> <y1> <x2> <y2> : specifies the lower left and upper right corner x,y instance coordinates for the region for shorting
- locID : locID is assigned on a pin, so that user can specify multiple regions in the same pin but treat each region independently (in finger formation, shorting etc)
- inst <instName> : get the coordinates of instances and then do the transformation to change the coordinates back to the cell-level.
- margin: expands the clamp bounding box by distance in um for node shorting considerations.
- includeAll: includes wire/via nodes in addition to pin nodes for shorting, where "1" means all wires/vias nodes inside the specified shorting bounding box are to be shorted, and "2" means all wires/vias nodes inside the expanded MARGIN shorting bounding box, but *not* inside the basic shorting bounding box, are to be shorted.
- finX/-finY <num\_fingers>: enables automatic clamp finger pairing in horizontal or vertical direction for multiple finger devices. The fingers of the same pin within the group are shorted together. Specifying a different number of fingers causes different finger pairing, as follows:
  - finX/-finY : if <num\_fingers> is not specified, each finger is defined by the PIN clamp file keyword. Each adjacent pin pair of different nets is recognized and defined by the ESD\_PIN\_PAIR clamp file keyword.
  - finX/-finY 1: each finger is defined by the PIN keyword. Each adjacent pin pair of different nets is recognized and defined by the ESD\_PIN\_PAIR keyword. Also, all nodes of the same finger are shorted together.  
When '-finX 1' or '-finY 1' option is used, the '-par' (number of partitions) value is automatically reset to 1 to avoid creating electronically redundant ESD\_PIN\_PAIRs.
  - finX/-finY N (N > 1) : every N fingers are clustered as a group. All nodes of the same pin within the same group are shorted together. Pin pairs are formed between different pins of the same group. This speeds up ESD checking by reducing the number of clamp devices (ESD\_PIN\_PAIRs).
- par <partition\_size>: specifies partitioning of each finger into a specified number of pin segments for finer control of clamp pin modeling. By default each partitioned finger segment has at least one node for pin pairing.
- dist <um>: instead of controlling the value of ESD\_PIN\_PAIR with the '-par <count>' option, -dist allows you to create ESD\_PIN\_PAIRs on fingers per clamp instance based on a specified distance apart (spacing).
- multiPair: when "ESD\_CLAMP\_PIN\_NODE\_DISTANCE <um>" is set in the GSR, -multiPair creates ESD\_PIN\_PAIRs on all clamp fingers according to the node distance specified in "ESD\_CLAMP\_PIN\_NODE\_DISTANCE", and you do not have to specify '-par/-dist' with a value when 'pfs add clamp\_pin' is executed.
- fwdR <forward\_res>: specifies the equivalent forward resistance (Ohms) of the group of pin pairs defined in the ESD\_PIN\_PAIR clamp file keyword (equivalent options are forwardR/-ron)
- bwdR <backward\_res>: specifies the equivalent backward resistance of the group of pin pairs defined in the ESD\_PIN\_PAIR keyword (equivalent options are -backwardR/-revR/-reverseR/-roff)
- perFwdR /-perBwdR : specify forward and backward resistance (Ohms) for each ESD\_PIN\_PAIR generated in the cell template, instead of an equivalent distributed resistance as with "-fwdR" and "-bwdR" options.
- ivname<iv\_name>: specifies the I-V model of the group of pin pairs defined by ESD\_PIN\_PAIR clamp file keyword

Note that the values of fwdR/bwdR are adjusted properly to all the pin pairs within the same region, so that the parallel combination of them equals the specified fwdR/bwdR.

- b. The ‘pfs delete’ command deletes previous esdcheck results from the database, and also deletes previously created shorted clamp pin regions.

- b1. To delete ESD results, the syntax is:

```
pfs delete ?-all? ?-type <rule_type>? ?-name <rule_name>?
```

where

-all: deletes all available ESD checking results

-type : deletes all ESD check results of the specified ESD type.

-name : deletes ESD check result of specified rule name. Note that the ‘-type’ option is required when the ‘-name’ option is specified.

Sample command to delete ESD results for rule ‘esd\_b2b\_rule’ of type b2b:

```
pfs delete -type b2b -name esd_b2b_rule
```

- b2. To delete previously-created shorted clamp pin regions created with the ‘pfs add’ command, the syntax is:

```
pfs delete clamp_pin ?-all? ?-cell <cell_name>? ?-pin <pin_name>?

 ? -pinPair {<pin_name1> <pin_name2>}?
```

where

-all : Delete clamp pin settings of all cells

-cell : Delete clamp pin settings of specified cell

-pin : Delete pin setting in the specified cell

-pinpair : Delete pin pair setting in the specified cell

- c. The ‘pfs export’ command exports descriptions of shorting clamp region into the specified text file list of TCL commands that can re-imported.

```
pfs export clamp_pin <output_file_name> ?-mergeDB?

 ? -esdCell <existing clamp_file>?

 ?-setupClamp? ?-shortNode [0 | 1 | 2]?
```

where

clamp\_pin <output\_file\_name>: specifies the output filename

-mergeDB : merges an existing clamp DB with clamp pin settings into the specified output file

-setupClamp : imports and saves the clamp info into the DB. Note that when -setupClamp is used, -esdCell is optional/not required.

-esdCell' : merges an <existing clamp file> with clamp pin settings into an output “template” file in a format that is similar to the clamp cell definition syntax, as well as include particular data options, if specified. The file syntax is:

-shortNode: Specifies the node shorting options. The available options are:

0 (off): Do not short ESD clamp pin nodes during extraction.

1 (all): Short all ESD clamp pin nodes during extraction.

2 (loc): Short ESD clamp pin nodes when x/y/layer are specified in ESD\_CLAMP\_FILE during extraction.

```
pfs export clamp_pin <file_name> -esdCell ?-fwdR? ?<fwdR>?

 ?-bwdR <bwdR> ? ? -ivname <I-V_model>?
```

where the file syntax keywords are:

-fwdR <forward\_res>: specifies the equivalent forward resistance of the group of fingers defined in the ESD\_PIN\_PAIR clamp file keyword (equivalent options are forwardR/-ron)

-bwdR <backward\_res>: specifies the equivalent backward resistance of the group of fingers defined in the ESD\_PIN\_PAIR clamp file keyword (equivalent options are -backwardR/-revR/-reverseR/-roff)

- ivname<iv\_name>: specifies the I-V model of the group of pin pairs defined by ESD\_PIN\_PAIR
- d. The 'pfs import' command is used to import the specified shorting clamp description file that has been previously exported using the 'pfs export' command.
 

```
pfs import clamp_pin <file_name>
```
- e. The 'pfs show' command displays clamp pin shorting and clamp I-V information.

```
pfs show
?clamp_pin? ?-cell <cell_name>? ?-pin <pin_name>?
?-shorting? ?-layer <layer_name>? ? -pinPair ?
? clamp_iv <clampDeviceName> ?-plot? ?-x <min,max>?
?-y <min,max>? ?-o <file>?
```

where

clamp\_pin : displays plot of clamp pin shorting as specified by options  
 -cell, -pin, and -layer options select the object or group of objects displayed.  
 -shorting : specifies that all shorted clamp pin points matching the specified filtering options are displayed in white flight lines (after extraction). Otherwise, only the boundary of the shorting region is displayed.  
 -pinPair: displays paired-up pairs of clamp pins (yellow flight lines)  
 clamp\_iv <clampDeviceName> - name of clamp device whose characteristics are to be plotted  
 -x <min,max> , -y <min,max> - specifies the range of plotted values for the X-axis (voltage in Volts) and Y-axis (current in Amps). The default for <min, max> is -10, +10. Instead of '-x' and '-y', the following options can also be used, with the same units: '-v' , '-i' or '-vol' , '-cur'.  
 -o <filename – text file describing the specified plot. By default the command plots the specified clamp I-V curve. When '-o' is specified, an ASCII text file is dumped out that can be read by xgraph or sv. When -o is used and '-plot' is not used, no curve is plotted.

User can export clamp\_pin with -esdCell (and/or -setupClamp), then use 'pfs show clamp\_pin -shorting -pinpair' to see the effect without waiting till after extraction.

## plot

---

```
plot [charge | current | rect | voltage | scatter | switching | line | analysis] ?<options> ? ?-o
<outputFileName>? ?-sv ? ?-nograph ?
```

The 'plot' command generates graphical plots of simulation waveforms based on specified conditions. The definitions for the options that are available for several 'plot' commands are not repeated.

- a. 'plot charge' creates a histogram of charge versus switching time.
 

```
plot charge ?-o <outputFileName>?
```

where

-o <outputFileName> : specifies output file name. Default - none.
- b. 'plot current' plots current waveforms for power, ground or individual nets or pads. The waveforms are extracted on the fly if not already created. Both Vdd and Vss currents are plotted. The output text file contains the working directory name and also the command used to create the text report in the header section. The general syntax is shown below; see the following examples for more specific usage and the option descriptions after the last one.

```
plot current [-net ?-power|-ground? | -switch | -pad]
? -virtual? ?-name <name>?
? -instance {instname pinname}
? -range <lowerNum> <upperNum>?] ?-overlay?
? -ldo -name <ldo_inst>? ?-o <out_file> ?
```

---

```
? -sv ? ?-nograph? [-png | -gif]
? -region <x1 y1 x2 y2>?
? [-fft ?-start_time <time>? ?-stop_time <time>?]?
?-npts <num>? ?-fs <freq>? ?-window?
?-probe -name <probe_name>? ?{<state1> ... <stateN>}?
```

where

- net -power |-ground : specifies plot of either total power or ground current
- switch : specifies plot of current for named switch instance
- pad : specifies plot of current for either all or of named pad instances
- virtual : plots current for specified low power virtual domain
- name <name>: specifies the individual net or switch name, or list of pads, for which to plot current.
- instance: to plot current per instance pin if DYNAMIC\_SAVE\_WAVEFORM 2 is set in gsr
- range <lowerNum> <upperNum> : specifies the current plots desired as ordered in the *pad.current* file
- overlay : adds a plot of total pad current to the total power or ground current plot
- ldo: displays load current waveforms for specified LDO instances.
- sv : specifies that the waveform data in whatever native format is to be plotted using the Apache 'sv' program. By default the waveforms are extracted and rendered in xgraph or sv format. In general, waveform types supported by the 'sv' program are: \*.tr0, \*.ta0, \*.ac0, \*.sw0, \*.hout, \*.fsdb, \*.wdb, and \*.pwl.
- nograph : plot data is sent to the output file only, with no GUI display
- png/-gif : specifies output waveform format, either png or gif.
- region : specifies corner coordinates for a rectangular region for which current is to be plotted, including presim switching current.
- fft: computes fast Fourier transform and plot frequency-magnitude result.
- start\_time <time>, stop\_time <time>: specifies start/stop time for FFT. The time unit can be sec, ms, us, ns, ps. Default is s.
- npts <number>: number of sampling points.
- fs <frequency>: sampling frequency. Units can be THz | GHz | MHz | KHz | Hz, default is Hz.

Note that -npts and -fs are used to calculate start and stop time using "stop\_time - start\_time = npts/fs". If neither start time or stop time are given, start time is assumed to be 0.

-window: specifies the use of a Hamming window  $w(k)=0.54-0.46\cos(2\pi k/(N-1))$ ,  $k=0$   $N-1$ .

- b1. plot current -net [-power | -ground] ?-o out? ?-sv?
 

Plots the total current drawn from the power or ground nets.
- b2. plot current -net [-power | -ground] -pad ?-o out? ?-sv?
 

Plots the total current for all power/ground nets from the *pad.current* file.
- b3. plot current -net [-power | -ground] -pad -overlay ?-sv?
 

The -overlay option adds the total pad current plot to the total power or ground current plot.
- b4. plot current -net -name <net\_name> -pad ?-o out? ?-sv?
 

Plots the current from all pads that belong to the specified net (from the *pad.current* file). The -pad option must be specified when -net and -name options are used.
- b5. plot current -region <x1 y1 x2 y2>
  - ? -inst\_list <Mem\_inst\_name> ?
  - ? -net -name <external\_PG\_net\_name> ?
  - ? -o <xgraph\_waveform\_filename> ?

Plots region-based current waveforms for all Vdd and Vss nets present in the specified region in the design. If the region is smaller than 50x50um, the command scales the region to 50x50um around the region specified. If '-inst\_list <>' is specified, operates on a specified memory instance. To plot current for a specific domain in a region, add the option '-net -name <net\_name>'.

- b6. `plot current -region <x1 y1 x2 y2> ? -file/-inst_file <instfile> ? ? -o <dir> ? ? -single ? ?-fft ?`  
 Plots all domains' fft information corresponding to each instance within the specified output directory. The -file/-inst\_file <> specifies filename with the list of instances in the specified region. The -single option outputs instance-based fft files under the directory specified after option -o.

- b7. The command 'plot current -switch' plots the current for the specified switch instance, or list of instances. The syntax is:

```
plot current -switch [-name <inst_name>] ?-netname <net_name>?

 ?-inst_list <inst_list_file>? ?-o out? ?-sv?
```

where

- name <instance name> specifies single switch instance
- netname <net\_name> provides the output sum of all switch currents associated with a specified net
- inst\_list <inst\_list\_file> provides the output sum of all switch currents from the specified list
- netname <net\_name> -inst\_list <inst\_list\_file> provides the output sum of all switch currents in the list connected to the specified net

- b8. `plot current -pad ?-o out?`

Plots the current for all pads listed in the *pad.current* file. The maximum number of plots displayed is 30.

- b9. `plot current -pad -name <list_pad_names> ?-o out? ?-sv?`

Plots the individual current waveforms of pads specified from *pad.current*.

- b10. `plot current -pad -range <lowerNum> <upperNum> ?-o out?`

Plots the current of pads in the specified number range, as listed in the *pad.current* file. The maximum number of plots displayed is 30.

- b11. 'plot current -pad -fft' displays an xgraph or sv plot of the pad current in the frequency domain.

- b12. The command 'plot current -probe -name <probe\_name> {<state1> ... <stateN>}' displays the current waveform in the Waveform Viewer at the defined probe locations for the states specified. For multi-cycle multi-state signal EM analysis, if the "-state" option is not specified, current waveforms for all states are displayed.

- b13. `plot current -instance {instname pinname}`

Plots the to the current plot on the screen.

- b14. `plot current -instance {{inst1 pin1} {inst2 pin2} ...} ?-o outdirectory?`

to put plot files in "outdiritory" for each of the requested instance pins. If -o is not specified, waveform will be generated by default to .wave directory. Also, a file called "cross\_ref" is included in this directory, which maps instance and pin names to a file in the outdirectory.

- c. 'plot rect' draws a specified rectangle on the design. The syntax is

```
plot rect -position <ll_x> <ll_y> <ur_x> <ur_y>

 ?-fill ? ? -color <color_name> ?
```

where

- position : specifies the lower left and upper right x,y coordinates of the rectangle
  - fill : fills the rectangle with same color as the rectangle lines
  - color : specifies the color for the rectangle line, and fill if specified (default white).
- d. 'plot voltage' <options>
- Creates a Vdd, Gnd or Pad waveform plot for a specified instance. For low power designs, use the -ext or -int options for switch instance voltages. A cross reference file named *cross\_ref* is created to indicate which waveform data corresponds to which instance or pad, using actual net names. Note that there is no display when the '-file' option is used to request a large number of plots. The output text file contains the working directory name and also the command used to create the text report in the header section.
- If the number of nodes is greater than 5M (default threshold), when the plot voltage command is executed the first time, the Vdd file is split into multiple Vdd files with 1M nodes in each file. From the second time the command is executed onwards, the 'plot voltage' command reads the split Vdd files. The 5M default threshold value can be adjusted using the GSR keyword 'SPLIT\_VDD\_EXTRACT\_LP <value>'. The number of nodes in each split Vdd file can be adjusted using GSR keyword 'SPLIT\_VDD\_EXTRACT\_LP\_FSIZE <value>' (default 1M).
- ```
'plot voltage [ -name <inst_name> | -xtorname <xtor_name> }
| -file <filename> ] ? [ -pad | -vdd | -gnd ]? ?[-ext | -int]?
? -virtual ? ? -sv ? [ -png | -gif ]
? -netname <list of net names>? ?-pinname <list of pin names>?
? -ldo -name <ldo_inst>? ? -o <output_file>? ? -nograph ?
? [-fft ?-start_time <time>? ?-stop_time <time>? ]?
?-npts <num>? ?-fs <freq>? ?-window?
```
- where
- name <inst_name> : specifies the instance name or pad name for which the waveform is desired
 - xtorname <xtor_name>: specifies the transistor name, or regular expression pattern, for the transistor waveforms to plot
 - file <filename> : specifies a file that contains a list of pads, transistors, or instances to have waveforms plotted, formatted one line per item
 - pad | -vdd | -gnd : specifies that PAD, VDD, and/or GND voltage waveforms are to be plotted. The '-fft' option displays an xgraph or sv plot of the voltage in the frequency domain.
 - ext | -int : for switch instances in low power circuits, selects which of the header or footer switch voltages are desired, internal or external.
 - virtual: plots voltage for the specified internal low power voltage domain
 - sv : specifies that the waveform data in whatever native format is to be plotted using the Apache 'sv' program. By default the waveforms are extracted and rendered in xgraph or sv format. In general, waveform types supported by the 'sv' program are: *.tr0, *.ta0, *.ac0, *.sw0, *.hout, *.fsdb, *.wdb, and *.pwl.
 - png/-gif : specifies output waveform format, either png or gif.
 - netname <list of net names>: specifies the instance net names for the waveforms desired.
 - pinname <list of pin names>: specifies the instance pin names for the waveforms desired.
 - ldo: displays output voltage waveforms for specified LDO instances.
 - o <output_file> : specifies either the output filename, or the directory name when used with the '-file' option. All the waveform data and cross file are dumped into that directory. Contains the working directory and the command used to create the text report in the header section. The default directory is /wave/.
 - fft: computes fast Fourier transform and plot frequency-magnitude result.
 - start_time <time>, stop_time <time>: specifies start/stop time for FFT. The time unit can be sec, ms, us, ns, ps. Default is s.

-npts <number>: number of sampling points.

-fs <frequency>: sampling frequency. Units can be THz | GHz | MHz | KHz | Hz, default is Hz.

Note that -npts and -fs are used to calculate start and stop time using “stop_time-start_time = npts/fs”. If neither start time or stop time are given, start time is assumed to be 0.

-window: specifies the use of Hamming window $w(k)=0.54-0.46\cos(2\pi k/(N-1))$, $k=0$. $N-1$.

e. ‘plot scatter ?-o <outputFileName>?’

Creates a scatter plot of switching instances on chip X,Y axes.

f. ‘plot switching’ ?-o <outputFileName>?

Creates a histogram of switching instances versus switching time.

g. ‘plot line’ <options>

Changes the appearance in the **RedHawk** GUI of a highlighted critical path or a clock tree path generated by timing analysis, or of a highlighted net associated with a selected pin in a timing path.

```
plot line [ -path | -net | -reset | -clearall | -instance
           <list of instance> | -position <start_x start_y end_x end_y> ]
           ?-color <color>? ?-width <width>?
```

where

-path : changes the appearance of a selected critical path or clock tree path highlighted

-net : changes the appearance of a net connected to a pin selected in either a ‘Path Detailed Information’ table for critical paths or a ‘Clock Tree Details’ table for clock trees.

-reset : resets all path and net display lines to default appearance

-clearall: clears all plot lines in the **RedHawk** GUI

-instance <list of instance names>: draws lines to connect specified instances in the order of the instance in the list.

-position <start_x start_y end_x end_y>: draws a line between specified x,y start and end points. The coordinates are design coordinates (microns).

-color <color> : specifies a new color for the specified timing and clock paths or nets. Choices are black, blue, brown, green, grey, orange, purple, red, white and yellow. Default color for path lines is white, and for net lines is ivory4.

-width <width> : specifies a new integer line width for specified timing and clock paths or nets. The default line width is 1 (1 pixel).

h. ‘plot analysis <options>’

Generates histogram results for all types of analyses. All GUI menu options for histogram analyses have a corresponding TCL command, as listed below:

```
plot analysis -type [ StaticIR | StaticEM | DvD | esd |
                      DynEM | signalem ] ? -emmode [ ave | rms | peak ] ?
                      ?-layer [<layer_name> | ALL]? ?-net [<netname> | ALL]?
                      ?-instance [avgTW | minTW | maxTW | minCyc |
                      instDrop | vddDrop | vssBounce]? ?-lowerLim <lower>?
                      ?-upperLim <upper>? ?-o <filename>?
                      ? -ruleType <ruleType> ? ? -ruleName <rule_name> ?
                      ?[-binsize <size_mv> | -binnumber <number_bins>]? ?-nograph?
```

where

-type [StaticIR | StaticEM | DvD| esd|DynEM |signalem] : specifies the type of analysis histogram to be created, in units of mv for voltage and % of limit for EM.

-emode [ave | rms | peak]: allows selection of the type of EM limit to plot.

-layer [<layer_name> | ALL] : for wire IR drop, specifies plot of specific metal layer name or plot for all metal layers

-net [<netname> | ALL] : for wire IR drop, specifies plotting Vdd drop or Vss bounce for a specific domain net name or ALL power nets.

-instance [avgTW | minTW | maxTW | minCyc | instDrop | vddDrop | vssBounce] : for instance voltage drop, specifies the parameter to use when generating the histogram:

- avgTW : average Vdd-Vss difference over timing window
- minTW : minimum Vdd-Vss difference over timing window
- maxTW : maximum Vdd-Vss difference over timing window
- minCyc : minimum Vdd-Vss difference over whole simulation
- instDrop : Vdd minus Vss for static analysis
- vddDrop : worst-case dynamic Vdd drop (minimum)
- vssBounce : worst-case dynamic Vss bounce (maximum)

-lowerLim <lower> : specifies the lower limit of voltage shown in histogram.

-upperLim <upper> : specifies the upper limit of voltage shown in histogram.

-o <filename> : specifies output filename.

-ruleType : for ESD data plots, specifies the type of rule data to be plotted, which can be : BUMP2BUMP (b2b), BUMP2CLAMP (b2c), BUMP2INST (b2i), CLAMP2CLAMP (c2c), CLAMP2INST (c2i), CLAMP2MACRO (c2m), PIN2CLAMP (p2c), PIN2INST (p2i), PIN2PIN (p2p)

-ruleName : specifies the ESD ruleName to be plotted

[-binsize <size> | -binnumber <number>] : specifies the step size, or the number of steps (in voltage) when generating the histogram.

-nograph : specifies no histogram is displayed in GUI window.

print

print [charge | decap | dvd | instance | inst_data | memory | mmx_decap_report | statistics | switching | type] ?<options> ?

The ‘print’ command prints text-based reports for specified conditions.

- a. ‘print charge’ prints charge statistics for each cell type. Apply ‘condition set -xy’.
- ```
print charge ?-o <outputFileName>?
```
- where
- o <outputFileName> : specifies output file. Default - output displayed in the terminal window where **RedHawk** was launched
- b. ‘print decap’ displays decap estimates for each cell type, and optionally the total decap for a specified net.
- Note: to define only a region to be reported by print decap, use the command ‘condition set -xy <ll\_x> <ll\_y> <up\_x> <ur\_y>’.

```
print decap ?-used? ?-o <outputFileName>?
```

where

- used : excludes decap of floating pins for multi-vdd decap cells, and reports only those decaps or parts of decaps that are connected to analyzed nets.
- o <outputFileName> : specifies output filename (no default file)

The command returns estimates of:

‘Non-switching Cap’ - the summation of cdev and C's in the SPEF and pin cap from all non-switching instances.

‘DevCap’ - the summation of the intrinsic cap from APL cdev values for all instances in the design.

‘LoadCap’ - the cap information in the SPEF files and pin cap from *.lib*.

‘Total PG Cap’ - from **RedHawk** analysis

‘Intentional Cap’ - cells tagged as “DECAP” in the DECAP\_CELLSSR keyword.

Note that more accurate decap values are available by running CPM ([Chapter 14, "Chip Power Modeling \(CPM\)"](#)).

- c. ‘print dvd’ prints a DvD analysis report in N number sections of “timesteps” specified in the condition command. Default: 500ps. Report contains instances in the order of decreasing charge.

To define a smaller region to be covered by print dvd, use the command

```
‘condition set -xy <ll_x> <ll_y> <ur_x> <ur_y>’
```

```
print dvd ?-limit [<number> | all]? -o <outputFileName>
```

where

–limit [<number> | all] : specifies the number of switching instances to be reported, or all instances. Default: 1000.

- d. ‘print instance’ prints a list of instances and their switching status (preceded by 1 = switching, 0 = not switching). To limit coverage of command , use ‘condition set -xy <xy\_coords> -time -type’.

```
print instance ?-o <outputFileName>?
```

- e. ‘print inst\_data’ prints information about the specified instance.

```
print inst_data <instance_name> ?-o <outputFileName>?
```

where

<instance\_name> : specifies an instance for printing information.

- f. ‘print memory’ prints a list of memory instances. To limit coverage of command , use ‘condition set -xy <xy\_coords> -time’.

```
print memory ?-o <outputFileName>?
```

- g. ‘print mmx decap’ prints information on MMX decaps. Default output is in file *decaps\_mmx.rpt*.

```
print mmx_decap_report ?-o <out>?
```

- h. ‘print statistics’ prints a list of key statistics about the design, as in the following example:

|                        |                     |
|------------------------|---------------------|
| Name of top cell:      | GENERIC             |
| Size of chip:          | 0 0 4920.62 5000.36 |
| Number of power nets:  | 2                   |
| Number of ground nets: | 1                   |
| Number of P/G pads:    | 80                  |
| Number of instances:   | 471853              |
| Number of memory/ip:   | 6                   |
| Number of sub blocks:  | 0                   |
| Number of clock inst:  | 6022                |
| Number of decaps:      | 133229              |
| Number of nodes:       | 1018591             |
| Number of resistors:   | 1149938             |

- i. ‘print switching’ returns a Tcl list of switching instance names. (The list may be very long if not restricted by condition. Apply “condition set -xy -time -type”.)

- j. ‘print type’ prints a list of the number of instances of each cell type in the design. To limit coverage of command , use ‘condition set -xy <xy\_coords> -time’.

```
print type ?-o <outputFileName>?
```

A sample ‘print type’ list:

```
Cell Type Statistics:
80878 of 476317 instances selected. 395439 ignored.
Rectangular region: Full chip
```

| Type          | Switch | Total | Switch % |
|---------------|--------|-------|----------|
| <hr/>         |        |       |          |
| Combinational | 6543   | 57900 | 11.30%   |
| Sequential    | 784    | 16918 | 4.63%    |
| Clock         | 3586   | 6060  | 59.17%   |
| Special       | 0      | 0     | 0.00%    |
| Memory/IP     | 0      | 0     | 0.00%    |
| Total         | 10913  | 80878 | 13.49%   |

## probe

---

**probe [ add | delete | get ] ? <options>?**

The ‘probe’ command allows selection and de-selection of particular nodes prior to extraction for which simulation output data are needed. No changes can be made to node selection after extraction; that is, if changes are made with the ‘probe’ command, extraction must be rerun. Note that probes also can be specified with the PROBE\_NODE\_FILE GSR keyword.

- a. The ‘probe add’ command identifies nodes for which simulation output data is to be created.

```
probe add <x> <y> <layer_name> <node_name>
```

For example:

```
probe add 1023.2 3457.1 Met1 nodeABC
```

To view current/voltage data for the nodes, use the following commands:

```
plot current -probe -name <node_name>
plot voltage -probe -name <node_name>
```

- b. The ‘probe delete’ command deletes specified previously selected probes, or all previously selected probes.

```
probe delete [<node_name> | all]
```

- c. The ‘probe get’ command displays information requested about the specified probe by name:

- Displays the probe location with layer information (-location).
- After extraction and simulation, displays current flowing through the named probe node (-current).
- In Static analysis, displays the current through and voltage at each probe node. For dynamic analysis, use the ‘plot’ commands.

```
probe get <probe_name> [-location | -current]
```

- c1. In Static analysis only :

- Display current flowing through the probe node. Syntax:

```
probe get <probe-name> -current
```

Example:

```
probe get nodetest -current
Current : 0.804813 mA
```

- Display the probe location with layer information. Syntax:

```
probe get <probe-name> -location
```

Example:

```
probe get test1 -location
```

4870.45 456.78 METAL4)

- Display voltage at each probe node. The syntax is:

```
probe get <probe-name> -voltage
```

## query

---

The ‘query’ command displays information and analysis parameters on selected objects in the design

```
query [top | inst | simulation_time | location] ?<options> ?
```

- d. ‘query top’ returns the bbox or name of the Ttop level design.

```
query top [-bbox | -name]
```

- e. The ‘query inst’ command returns the bbox of given instances in one of the three forms:

- a TCL list of instances
- a pattern for name matching
- a filename that contains the instance names.

```
query inst -bbox [<tcl-list> | <pattern> | <filename>]
?-o <file> ?
```

- f. ‘query simulation time’ returns the simulation time according to the option requested.

```
query simulation_time [-power ?-detail? | -gsr]
```

- f1. ‘query simulation\_time -power’ returns a display of the type:

2500ps

Values are obtained from the power summary file.

- f2. ‘query simulation\_time -power -detail’ returns a display of the type:

Recommend 2500ps to include 96.7841% total power.

Values are obtained from the power summary file.

- f3. ‘query simulation\_time -gsr’ returns a display of the type

0.0 5e-09

Value is from the GSR keyword DYNAMIC\_SIMULATION\_TIME.

- f4. ‘query location’ displays information about the object at a given location and layer. If layer name is not specified, all layer information at that location is displayed.

```
query location -xy ?-layer?
```

## report

---

The ‘report’ command creates a number of reports on RedHawk and clock network analysis results.

```
report [block_power_assignment | clocknetwork | cmmpin | dvd | esdcheck | ir | leafinst | net |
power | res_calc | resistance | result] ?<options> ?
```

- a. ‘report block\_power\_assignment’ or ‘report bpa’ creates BPA summary reports in the *adsRpt/apache.blockPowerAssign* file (default). With the ‘-node’ option it also reports nodes used for power/current assignment for the BPA instances:

```
report bpa ? -o <file>? ?-node?
```

- b. ‘report clocknetwork’ creates reports on clock network analysis

```
report clocknetwork ?-o <output_file>? ?-root <trace_start_pin>? ?-
leaf <trace_end_pin>? ?-block <block_inst_name>? ?-dumpLeaf? ?-
detailArc ?
```

where

-o <output\_file>: writes a clock instance list into the specified output file.

- root <trace\_start\_pin>: writes a clock instance list under the specified starting pin into the output file.
- leaf <trace\_end\_pin> : writes a clock instance list driving the ending pin into the output file.
- block <block\_inst\_name> : writes a clock instance list for the specified block into the output file
- dumpLeaf : writes leaf information into the output file.
- detailArc : writes detail pin-to-pin arc information into the output file.

- c. ‘report cmmppin’ reports CMM pin static voltages, including boundary pin voltages, to insure that the interface of CMM blocks are at correct voltage. Static analysis must have been performed. A file is produced, *adsRpt/cmm\_pin\_voltages.rpt* by default, reporting the worst voltage for each pin of each CMM instance. The format of the report file is as follows.

```
Report Format
<inst_name> <CMM_cell_name> <x_loc_of_pin> <y_loc_of_pin>
<layer_name> <net> <real_voltage> <ideal_voltage> <pin_name>
```

- d. ‘report ir/dvd’ commands provide reports of static and dynamic voltage drop values.

```
report [ir | dvd ?-via?] ?[-routing | -instance | -mmxpin]?
 ? -net [<net1> ... <netN>] ? ? -bbox ?
 ? -layer [<layer1> ... <layerN>]? ? -o <outfile_path> ?
 ? -threshold <volt_drop>? ?-limit <output_limit>?
 ? -precision <# digits>?
```

where

ir : creates a static voltage drop report

dvd : creates a dynamic voltage drop report. If ‘-via’ is selected, a DvD via report is created, as long as the ‘VIA\_IR\_REPORT’ GSR keyword is set to 1.

-routing : generates a routing wire-based voltage drop report

-instance : generates an instance-based voltage drop report

-net : specifies the nets to be reported. All nets are reported by default

-bbox : only reports the IR or DVD results for the bbox region specified.

-layer : specifies the layers to be reported. All layers are reported by default.

-o : specifies the path of output file

-threshold : specifies a voltage drop threshold for reporting.

Note that ‘-threshold’ is applicable only for the ‘-routing’ option, and not with the ‘-instance’ or ‘-limit’ options.

-limit : specifies the maximum number of output lines to be reported. When the option ‘-net’ is used, -limit refers to the maximum number of report lines per layer on which the net exists.

When the -limit value ‘-1’ is used with -instance option, RedHawk generates reports for all instances. Default is 1000 lines.

-precision : specifies the number of digits to report after the decimal point. Default: 3 digits, but some variation per type of report.

The output file has the following format:

```
#Report voltage information at each location (x, y)
voltage ideal_volt net x_y_location layer_name
 0.0273 0.0000 VSS (811.480, 3446.870) METAL1
 0.0275 0.0000 VSS (821.100, 3446.870) METAL1
 0.0275 0.0000 VSS (824.780, 3446.870) METAL1
 0.0276 0.0000 VSS (828.460, 3446.870) METAL1
```

- e. ‘report esdcheck’ specifies the type of reports to be created for the ESD checks. The command is:

```
report esdcheck -type <ruleType> ? -name <rulename>
 ? -b2cName <b2c_rule_name>? ?-[glob |-regexp] ?
```

```
? -outDir <output_dir>? ? -reportB2C ? ? -reportC2B ?
? -arcR <max_R>? ? -loopR <loop_threshold>?
? -parallelR <parallel_threshold>? ? -append? ? -excel ?
? -cell { <cell1> ... <cellN> }?
? -failedOnly? ? -detail? ? -netPairVth <Vth_value>
?,<net1>, ..., <netN>? ? -peakVolt <peak_V>?
?-diffVolt <diff_V>?
? -summary ? -sort [name|em|I|R]? ? ? -o <filename>?
```

where

- type: reports results for the specified type of ESD checks. Default is all types.
- name: specifies the name of the rule tested. If multiple checks have been specified within a single rule-- for example, two bump pairs are specified in the same rule-- PathFinder internally splits them into sub-rules, each with a single bump pair. Then it appends '(\_\d+)+' to the base rule name. So when using 'report esdcheck', you can specify this derived name to look at the particular results desired.
- b2cName <b2c\_rule\_name>: user-specified B2C rule to be used for checking
- glob|regexp: specifies the type of wildcard expressions used. PathFinder reports the results for all sub-rules that match the specification; 'glob' supports regular wildcard expressions and 'regexp' supports normal wild cards and also extended character strings. For example:

```
report esdcheck -detail -perNet -type B2C -name abc* -glob
report esdcheck -type B2C -name {M_test_[3-5]} -regexp
```

- outDir : specifies the report files directory. Default is *adsRpt/ESD*
- reportB2C/-reportC2B: prepares reports on B2C/C2B CD checks in reports that have either bump-centric or clamp-centric results. By default, without either '-reportC2B' or '-reportB2C' options, 'report esdcheck' reports in the mode in which the esdcheck was performed.
- arcR: specifies the acceptable resistance threshold for arc-R checking; required for B2C, C2C, C2I, C2M. The format of the report is:

```
<ARC_R> <x,y loc> <layer> <domain> <pin> <inst> <clamp> <x,y loc> <layer>
-loopR: specifies the threshold for loop-R checking; required for C2I
-parallelR: specifies the threshold for parallel-R checking; required for B2B
-append: appends results to existing results in the output directory.
-excel : creates a consolidated ESD report file in Excel format, adsRpt/ESD/esd_excel.rpt, for
resistance and CD checks, so that the information can be easily sorted and processed
using Excel.
-cell : reports CD results for specified cell names
-failedOnly: only reports failed results.
-detail : provides more detailed report results for B2B rules
-netPairVth: reports driver-receiver voltage pair threshold violations for current density (CD)
checking rules. Optionally, one or more nets with the same VTH value can be specified,
separated by commas.
-peakVolt: specifies the peak voltage to be used for CD checks
-diffVolt : specifies the differential voltage to be used for CD checks
-summary : prepares a summary esdcheck text report, primarily for CD checks, but some data
is also presented for other rule types, if selected.
-sort : arranges the summary report items in the selected sorting order based on either rule
name, worst EM % value, maximum clamp current, equivalent resistance between points.
Default sorting order is by rule name.
```

- o : creates an output text report in the specified filename. By default, summary data is displayed to the screen only.
- f. 'report leafinst' reports output adsU1 or adsCopy instance/cell P/G pin voltages for GDS blocks in the 'report leafinst' output file. Reports not only pins of gds2def/gds2rh, but also Black Box pins. *DMP compatible*. The command syntax is:

```
report leafinst -pg_volt -o <filename> ?-f <input_file>?
```

where

- pg\_volt: specifies pin voltages be reported. If '-pgvolt' only is used, the report is in the latest format, either with or without the -f option
- o <filename> : specifies the output filename
- f <input\_file>: only reports instances/cells specified in this input file

The output report format is as follows:

```
<inst_name> <cell_name> <inst_x> <inst_y> <layer_name>
<domain> <eff_voltage> <source_voltage> <source_name>
```

where

- <eff\_voltage> is the actual voltage
- <source\_voltage> is the ideal voltage

A sample file is shown below:

```
core/mem/roms2/adsBoundary RODSH_ADSBOUNDARY 774.617 849.315 METAL4 VDD
 1.19579 1.2 VDD
core/mem/roms2/adsBoundary RODSH_ADSBOUNDARY 774.617 678.330 METAL4 VDD
 1.19561 1.2 VDD
core/mem/roms2/adsBoundary RODSH_ADSBOUNDARY 848.440 675.830 METAL4 VDD
 1.19562 1.2 VDD
core/mem/roms2/adsBoundary RODSH_ADSBOUNDARY 700.795 675.830 METAL4 VDD
 1.19549 1.2 VDD
core/mem/roms2/adsBoundary RODSH_ADSBOUNDARY 848.440 763.822 METAL3 VDD
 1.1957 1.2 VDD
```

- g. 'report power' provides reports on design power data. The default is to report a summary of the whole design for all nets. The TCL command syntax is:

```
report power ?-inst <inst_name>? ?-region <llx lly, urx, ury>?
?-region_namee>? ?-xtorlist <xtorlist_file>?
?-net <net_name>? ?-mmx_only? ?-o <filename>? ?-all?
```

where

- inst: specifies the instance name (wildcards are supported)
- region <corner locations>: specifies the bounding box
- region\_name: specifies the bounding box name for -region
- xtorlist: specifies a file containing device names (wildcards are supported)
- net: specifies the net names. Default is to report all nets.
- mmx\_only: only reports power for MMX instances.
- o: dumps report to <filename>
- all: generates report file *adsRpt/\$designname.power.rpt*

- h. 'report res\_calc' produces a consolidated text report on the results of all res\_calc operations performed:

```
report res_calc ?-o <file>? ?-loop?
```

where

- o <file>: specifies the output filename for res\_calc results; the default filename is *adsRpt/ <design>.res\_calc*

- loop: reports the total of vdd+ gnd resistance for instances, as computed in the -loopMode option
- i. ‘report resistance’ calculates and reports the resistance of a wire to verify the resistance calculation of **RedHawk**.
 

```
report resistance -layer <name> -length <> -width <> -density <>
 -spacel <l_space> -spacer <r_space>
 -colorw <> -colorl <> -colorr <>
```

 where
  - spacel and spacer are left and right wire spacing
  - colorw, colorl and colorr are the Mask Colors for the current wire, left wire and right wire respectively. All dimensions are in microns.
- j. ‘report result’ creates a full set of **RedHawk** analysis reports and color maps as specified in the design constraint file.
 

```
report result ?-constraint <constraint_filename>?
 ?-o <output_file>?
```

 where
  - constraint <constraint\_filename> : specifies constraint file name
  - o <output\_file> : specifies output filename

## **ring**

---

The ‘ring’ command adds and deletes specified power grid rings from the design.

- a. ring add <arguments>

See [section "Mesh and ring commands - FAO functions to optimize and fix grids."](#), page 7-154, for details on the ‘ring add’ command.

- b. ring delete <ring\_name>

See [section "Mesh and ring commands - FAO functions to optimize and fix grids."](#), page 7-154, for details on the ‘ring delete’ command.

## **route fix**

---

**route fix [ -from { x1 y1 x2 y2 <layer>} | -from\_pt { xf yf <layer>} ]**

**[ -to {x1 y1 x2 y2 <layer>} | -to\_pt { xt yt <layer>} ] -width <float>**

The ‘route fix’ command routes new power/ground wires and adds associated vias to the **RedHawk** design database to reduce excessive voltage drop.

See [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#), for details about using this command.

## **select**

---

**select [add | addfile | clear | clearall | get] ? <options> ?**

The ‘select’ command selects and highlights objects in the GUI.

- a. ‘select add’ adds the specified object to the selection.

**Note:** When a lot of objects are being selected, you can stop the selection process by pressing the keys **CTRL + Break** simultaneously, and the ‘select add’ operation stops.

```
select add <object> ?-linewidth <width>? ?-color <color>?
 ?-name <groupname>? ?[-exact | -glob | -regexp]?
```

where

<object> : specifies a single instance or a TCL list of instance names. The names can include patterns (wildcard, expressions).

- linewidth <width> : specifies linewidth for the line used to highlight the selected object.
- color <color> : specifies the color used to highlight the selected object, from Xlib *rgb.txt* or their hex value.
- name <groupname> : adds a defined group of objects to highlight
- [-exact | -glob | -regexp] : specifies object name matching criteria. Default - exact. '-glob' allows use of wildcards, '-regexp' allows use of regular expressions.
- b. 'select addfile' adds the objects specified in the file to the selection.  
`select addfile <FileName> ?-linewidth <width>? ?-color <color>?`  
 where  
`<FileName>` : specifies the filename containing a list of object.
- c. 'select clear' removes the objects from the selection.  
`select clear <object> ?[-exact | -glob | -regexp]?`
- d. 'select clearall' removes selected types of objects, or all objects from the selection set.  
`select clearall ?[-instances | -markers | -lines | -ruler]?`
- e. 'select get' returns a list of object names that are currently selected.  
`select get`

## setup

---

**setup [analysis\_mode | .apache | apl | cpa | design | fsdb | package | pad | pss | switchres | wirebond ] ? <options> ?**

The 'setup' command sets up data and conditions required for performing **RedHawk** analysis.

- a. 'setup analysis\_mode' sets up data and conditions specifically for the selected type of analysis prior to the 'perform' command, to save on CPU and memory.  
`setup analysis_mode [ static | dynamic | jitter | lowpower | signalEM | esd ] ? -dmp <config_file> ?`  
 where  
`-dmp` : invokes distributed multi-processing for computation.
- b. 'setup .apache <dir>' allows pointing the *.apache* directory to a user-specified local drive before loading the DB or the GSR file.
- c. 'setup apl' generates setup data for an external APL run.  
`setup apl ?-dir <output_directory>?`  
 where  
`-dir <output_directory>` : specifies the directory for the APL setup file.
- d. 'setup cpa' sets up a cpa run and designates the desired thermal, ploc and scaling parameters  
`setup cpa component -name <componentName>`  
`[ -thermal -power xx (W) | -ctm ctmfile | -mhs mhs.file ]`  
`ploc -auto -ploc <file> -scale <scale> -tolerance <tolerance>`  
 where
  - thermal: used only for die components in thermal simulation
  - power: assigns constant power for a die component in BGA packages.
  - ctm: specifies CTM path for a die component.
  - mhs: specifies multi-heat source file for a die component`d1) setup cpa part -name <partname_list> ?-type`  
`[die|resistor|inductor|capacitor|bga|other]? ?-neverflip [true|false]?`

To turn flip on or off for a part specified, where  
`-neverflip`: specifies NeverFlip is on(true) or off(false)

d2) `setup cpa unplocnet -net <net1, net2...> ?-float? ? -volt <voltage_value>?`

To command to float or short to voltage source for ploc unconnected die domain net, where:

-net: specifies a net or nets list

-float: floating the ploc unconnected die domain nets

-volt: specifies the voltage value

- e. ‘`setup design`’ sets up design database for analysis, after importing design files.

`setup design ? <gsrFilename> ?`

where

`<gsrFilename>` : name of GSR file

- f. ‘`setup package`’ defines the package parasitic parameters R, C, and L for package equivalent circuits. RLC elements are considered a P/G loop, even if specified separately.

`setup package [-c <c_val>] [-ignore | -power | -ground]  
 [-r <r_val> | -l <l_val>]`

where

-c <C\_pF> : specifies equivalent package capacitance in picoFarads. Default: 0.

-ignore : specifies package RLC parameters are to be ignored

-r <R\_Ohms> : specifies equivalent package resistance in Ohms. Default: 0.

-l <L\_pH> : specifies equivalent package inductance in picoHenrys. Default: 0.

- g. ‘`setup pad`’ defines the pad parasitics parameters.

`setup pad [-ignore | -power | -ground] [-r <r_value> | -c <c_value> ]`

where

-ignore : specifies pad parameters are to be ignored

-power : specifies the use of power RC parasitics.

-ground : specifies the use of ground RC parasitics.

-c <c\_value> : specifies capacitance used for pad parasitics (pF). Default: 0.

-r <r\_value> : specifies resistance used for pad parasitics (Ohms). Default: 0.

- h. ‘`setup pss`’ defines the PLOC and package subcircuit files to be used in simulation

`setup pss -pad_file <filename> -subckt <pss_ckt_name>`

where

-pad\_file <filename> : specifies a PLOC pad definition file that must have a .ploc extension and be in PSS PLOC format.

-subckt <pss\_ckt\_name> : specifies the Spice package subcircuit file

For example:

`setup pss -pad_file test_pss.ploc -subckt pkgA.sp`

- i. ‘`setup switchres`’ defines or clears a switch resistance file specifying RON values, overriding the values of the switch model file:

`setup switchres ?<file>? ?-clear? ?-Ron <Ohm>? ?-cmmOnly?`

where

<file> : a file containing user-specific switch instance name and resistance value (equivalent to the GSR option SWITCH\_RES\_FILES)

-clear : to clear user-specific values

-Ron : specifies resistance value which is applied to all switches.

-cmmOnly : to apply -Ron to switch instances inside CMM only.

- j. ‘`setup wirebond`’ defines wirebond or flip-chip parasitics

`setup wirebond [-ignore | -power | -ground]`

[ -r <r\_value> | -l <l\_value> | -c <c\_value> ]

where

- ignore : specifies wirebond parameters are to be ignored
- power : specifies the use of power RLC parasitics.
- ground : specifies the use of ground RLC parasitics.
- c <c\_value> : specifies capacitance used for wirebond parasitics (pF). Default: 0.
- l <l\_value> : specifies inductance used for wirebond parasitics (pH). Default: 0.
- r <r\_value> : specifies resistance used for wirebond parasitics (Ohms). Default: 0.

## show

---

### show <map\_type>

The ‘show’ command displays a colormap of specified **RedHawk** results in the GUI.

The colormap results that can be displayed by the ‘show’ command are the same as can be saved as a GIF file by the ‘dump GIF -map <map\_type>’ command. Most of these colormaps can also be accessed from the **View** menu. See the list of colormap types and descriptions in [section "dump", page D-760](#).

- a. The ‘show sicm’ and ‘show sivm’ commands display colormaps of color-coded current and voltage for the power switches in a design. The color range dialog box can be used to change the colors indicating particular voltage and current ranges as desired.

## window

---

The ‘window geometry’ command allows you to change the **RedHawk** window geometry so that the “dump gif” command can produce a higher resolution color map.

```
window ? geometry ?<x> <y>? ?<width> <height>? ? cmd?
 ? title <title name>? ?-absolute? ? snapshot <-o ?outputfile>
```

where

- window geometry: returns the current x,y window location (upper left corner) relative to the upper left corner of the screen (pixels)
- window geometry <width> <height> : changes the window width and height to the specified dimensions (in pixels)
- window geometry <x> <y> <width> <height>: changes the window geometry to the specified width and height, with upper left corner at the specified x and y location, in pixels
- window cmd : to launch a command browser window containing history of all previously used tcl commands, which can then be executed from this window directly
- window title : provides user-specified window title, of the format “<specified\_title>: <top cell name>”
- window title -absolute : the title is exactly what you specify
- window snapshot <-o outPutName> : to dump snapshot

## zoom

---

The ‘zoom’ command controls the GUI size display function, primarily for TCL script execution.

### zoom [ in | out | top | rect <x1 y1 x2 y2> ]

where

- in : increases the size of the objects in the GUI display by
- out : decreases the size of the objects in the GUI display by
- top : displays the full design

rect <x1 y1 x2 y2> : displays a rectangular portion of the design with corner coordinates x1,y1 and x2,y2

## Running RedHawk in the TCL Script Mode

---

The following command runs **RedHawk** in TCL script mode by executing the commands in a specified TCL script command file. In this mode you do not have access to the GUI or to the interactive command line.

```
% redhawk -b <script_command_file>
```

The results from standard output can be redirected to a log file, as shown below.

```
% redhawk -b <script_command_file> >& <log_file>
```

The -b option exits **RedHawk** after the last command in the command file, or when a command in the file is not completed successfully.

The following commands run **RedHawk** in interactive command mode by executing the commands in a specified command file. In this method you have no access to the GUI, but can interactively issue TCL commands.

```
% redhawk -i
% Redhawk > source <script_command_file>
```

```
% redhawk -i
% Redhawk > type in tcl commands one by one
```

```
% redhawk -i
% RedHawk > exit
```

The following command runs **RedHawk** from the GUI in various modes. Run **RedHawk** from the GUI by executing the commands in the command\_file.

```
% redhawk -f <command_file>
```

**Note:** If you have a file named **.redhawkrc** in your home directory or in the current working directory that has one or more standard RedHawk TCL commands in it, RedHawk executes that file before it executes any other RedHawk command.

Run **RedHawk** from the GUI by using the “playback” of the commands in the script\_command file.

```
% redhawk
```

Click on menu: **File -> Playback -> <TCL\_script\_command\_file>**

Run **RedHawk** from the GUI by issuing the TCL command:

```
'source <TCL_script_command_file>'
```

Use the window at the bottom of the GUI for entering TCL commands one at a time.

```
% redhawk
```

In TCL command window type: `source <TCL_script_command_file>`

In TCL command window type TCL commands one by one.

## Starting the GUI at a designated step in batch mode

You can start to run **RedHawk** in non-GUI (batch) mode initially, with a command prompt, and then display the GUI at a specified step for design examination. You must add the command 'gui' in a command file after the command step where you want the GUI to be displayed, such as creating a command file *run.tcl*, as follows:

```
Import data
import gsr GENERIC.gsr
```

```
setup design
Calculate power
perform pwrcalc
Power grid extraction
perform extraction -power -ground
To display the GUI
gui
```

Then start **RedHawk** using:

```
redhawk -i run.tcl
```

When the ‘gui’ step is reached, the **RedHawk** GUI and the design are displayed. Now in interactive mode you can explore key aspects of the design and continue running commands from the GUI prompt.

## TCL Script Execution Examples

---

The following TCL script command file performs DB exports at various stages of a normal **RedHawk** run.

```
import gsr user_data/design.gsr
setup design
export db Post_Setup_DB_Dir
perform pwrcalc
perform extraction -power -ground
export db Post_Extr_DB_Dir
setup pad -power -ground
setup wirebond -power -r <Res> -l <Induct> -c <Cap>
 -ground -r <Res> -l <Induct> -c <Cap>
setup package
perform analysis -static
export db Post_Simu_DB_Dir
```

DB directories *Post\_Setup\_DB\_Dir*, *Post\_Extr\_DB\_Dir*, and *Post\_Simu\_DB\_Dir* are created. Each contains a snapshot of a **RedHawk** run at various stages of command execution.

The following script, using TCL commands, first loads the previously saved snapshot *Post\_Setup\_DB\_Dir*, then continues with power calculation, extractions, and simulations.

```
import db Post_Setup_DB_Dir
perform pwrcalc
perform extraction -power -ground
setup pad -power -r 0.1 -l 0.1 -c 0.2
 -ground -r 0 -l 0 -c 0
setup wirebond -power -r 0.01 -l 0.1 -c 0.1
 -ground -r 0 -l 0 -c 0
setup package -r 0.002 -l 50 -c 5
perform analysis -static
```

The following execution script using TCL commands bypasses power calculation and extraction by loading the previously saved snapshot *Post\_Extr\_DB\_Dir*:

```
import db Post_Extr_DB_Dir
setup pad -power -ground
setup wirebond -power -r <Res> -l <Induct> -c <Cap>
 -ground -r <Res> -l <Induct> -c <Cap>
setup package
perform analysis -static
```

## Sample TCL Scripts

---

### Static IR Drop Analysis Example

There are three ways to import design information into **RedHawk**:

- using the ‘import gsr’ command
- using the ‘setup design’ command
- using the `import gsr` command, and then specifying different design files to override those specified in the GSR file.

Following is an example of the three methods for importing design information for static voltage drop analysis using the TCL command format. Comment lines are preceded by “#” or “\*” characters.

```
importing data, method 1
import gsr design_static.gsr
setup design

importing data, method 2
setup design design_static.gsr

importing data, method 3 (not recommended)
import gsr design_static.gsr
import tech /work/6LM013um.tech
import def xyz.def
setup design
```

The TCL commands for running static voltage drop analysis are:

```
begin static analysis
perform pwrcalc
```

### perform extraction -power

---

```
setup pad -power -r 0.010000 -c 0.5
setup pad -ground -r 0.010000 -c 0.5
setup wirebond -power -r 0.245000 -l 1420 -c 5
setup wirebond -ground -r 0.245000 -l 1420 -c 5
setup package -r 0.002 -l 10 -c 3
perform analysis -static
export db design_static.results
```

In the above example, the pad cell `design.pcell` and pad location `design.ploc` files define the voltage and ground sources.

After power calculation is performed, power results are automatically stored in the `adsPower` directory. If you have previously calculated power results, you must rename the existing `adsPower` directory, otherwise the new data overwrites the existing information.

Power-net-only analysis and resistance extraction are defined by

```
perform extraction -power
```

The example ignores both capacitance and inductance extraction, as they are not required for static analysis.

The **RedHawk** database is exported to the `design_static.results` directory after static IR drop analysis is completed.

## Dynamic Voltage Drop Analysis Example

Below is an example of a typical command file for DvD analysis using the TCL command format.

```
import gsr design_dynamic.gsr
import apl <cell>.current
import apl -c <cell>.cdev
setup design

#import power and begin dynamic analysis
import power adsPower_gold
perform extraction -power -ground -c
setup pad -power -r 0.010000 -c 0.5
setup pad -ground -r 0.010000 -c 0.5
setup wirebond -power -r 0.245000 -l 1420 -c 5
setup wirebond -ground -r 0.245000 -l 1420 -c 5
setup package -r 0.002 -l 10 -c 1
perform analysis [-vectorless | -vcd]
export db design_dynamic.results
```

The example shows importing of the pre-characterized decoupling capacitance file *<cell>.cdev*.

The power results are imported from the *adsPower\_gold* directory. This assumes that the latest power results are already available in *adsPower\_gold*. Optionally, you may re-calculate the power, in which case the power results are written to *adsPower* directory.

Simultaneous power and ground net analysis and resistance/capacitance extraction are defined by 'perform extraction -power -ground -c'. The example ignores inductance extraction.

The **RedHawk** database is exported to the *design\_dynamic.results* directory after dynamic voltage drop analysis is completed .

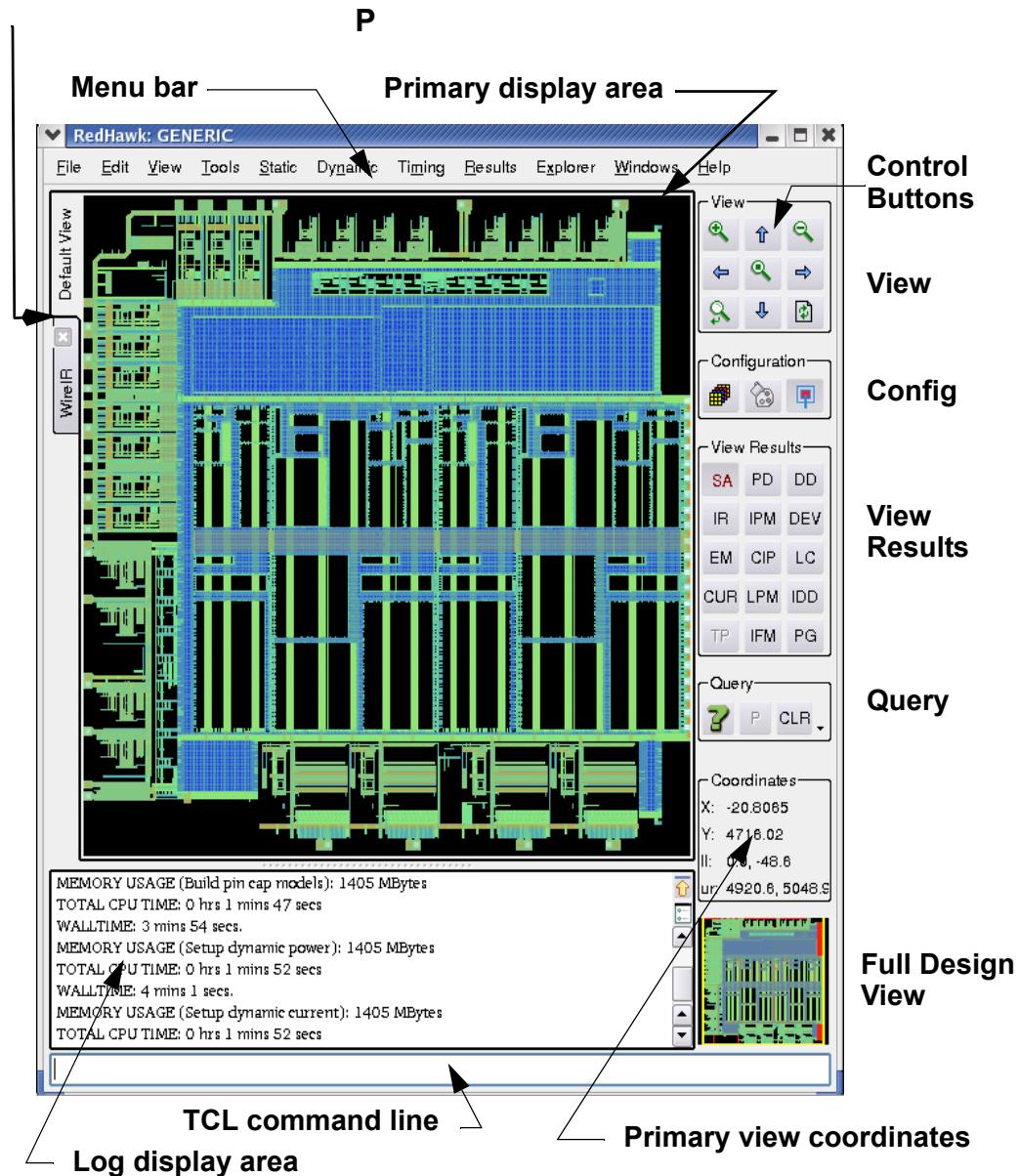
## Automated Color Map Generation

The following TCL script creates six VDD voltage drop and six GND bounce color maps with values between 5% to 30% of nominal VDD value (max):

```
set i 5
set up 31
while {$i < $up} {
 config colormap -percent -min 0 -max $i -wire
 config viewlayer -name metalonly -style fill
or for Layer1: config viewlayer -name Layer1 -style fill
for VDD
 config viewnet -name VDD -mode on
 config viewnet -name GND -mode off
 dump gif -map IR -o IR_VDD_$i.gif
for GND
 config viewnet -name VDD -mode off
 config viewnet -name GND -mode on
 dump gif -map IR -o IR_GND_$i.gif
 incr i +5
}
```

## RedHawk Graphic User Interface Description

The **RedHawk** GUI functional areas are shown in [Figure D-3](#) below. See the section "Graphical User Interface", page 3-10, for an overview of the key functional areas in the interface.



**Figure D-3** RedHawk Graphical User Interface display

### Mouse Function

The mouse provides the standard method of manipulating the pointer and making selections in the GUI. In addition, the left and right mouse buttons provide enhanced graphic selection functions as follows:

## Left Mouse Button Object Selection, Highlighting and Query

To display the properties of objects in a design you can left button click on them in the layout view (**SA** button). The object selected is highlighted, and its properties are shown in the log display area. Note that the objects desired must have their display properties turned on, such as instances, pins or vias, and the appropriate layer must be displayed (see the Layer dialog description and Figure D-6).

The **Show Power Pad** button must be pressed to display the pins/pads. The properties displayed include: pin name, layer, connected top-level net, rectangular coordinates, and center location. If static simulation has been performed, the average pin voltage and current values are also shown in the log display. Peak voltage is displayed after dynamic simulation.

In general, clicking on a point in the design highlights the selected object in the lowest layer first (metal1). Successive clicks on the same point highlights objects on the next lower layer, if there is an object at that point on the layer. This selection process cycles with each successive click at the same point.

**Note:** you may have to zoom in several times before a small object can be identified and highlighted for querying.

Also, to display pads, the appropriate layer display must be selected.

Clicking on a point on a wire after completion of dynamic analysis displays a list of the wire parameters in the **RedHawk** log window:

- net name
- layer ID
- wire width and length in microns
- wire segment corner coordinates
- resistance (Ohms)
- query point x,y coordinates
- voltage at query point
- current at query point
- EM mode
- EM rule
- EM percentage (parallel and perpendicular) and current limits in Amps

Note that depending on the circuit topology, there may be current in both the wire direction and perpendicular to the wire direction. If present, both current elements are reported to assist in EM investigations. If there is current in only one direction, the query report is of the form:

```
Current parallel to wire direction at query position: 0.000705042 A
v
```

If orthogonal current exists, then the query report looks as follows:

```
Current perpendicular to wire direction at query position:
0.000705042 A < 0.001 ^
```

The character appearing after the current value (Amps) indicates its direction (<, >, ^, v).

Also, external voltage sources connected to the grid can be queried. Click the pad to see layer, voltage and geometry information.

## Right Mouse Button Zoom

- Zoom IN - depressing the right mouse button and dragging from *left to right* creates a rectangular area that is zoomed in.
- Zoom OUT - depressing the right mouse button and dragging from *right to left* creates a rectangular area that is zoomed out. The view is zoomed out by the ratio of the layout view size to the drawn rectangle size.

## Mouse Wheel Zooming

Using the wheel on the mouse, you can zoom in rapidly pushing the wheel up, and zoom out rapidly pushing the wheel down.

## Using GUI Dialog Box Settings in RedHawk

---

Many buttons and menu functions bring up dialog boxes to allow configuration and display options to be changed. If changes are made to the display configuration and the **Apply** or **OK** buttons in the dialog are executed, the changes to the configuration are saved. Even if **Cancel** is pressed to dismiss the dialog, the GUI configuration settings are retained as they were when they were set or saved. However, if the **X** ("destroy") button in the upper right-hand corner of a dialog box is used to dismiss a dialog, any changes to the display are lost and the settings revert to their default values thereafter.

For colormaps, each of the ten color settings that can be applied to GUI parameter values can be turned on or off independently. Also, to easily return to the original color display values, click **Default** to reset the values of all fields.

Color legend data can be displayed beside the primary display area (off by default). To turn on this function, use the menu command , then check the box **Color Maps-> Show Legend** in the dialog. Then all color maps and views have their corresponding color legend displayed. The 'dump gif' command with this feature On captures the display area that includes color scale legend by default.

To save a particular set of new GUI configuration settings to a file, use the TCL command '`export guiconf <filename>`'. The settings for any color maps changed and saved are exported to the specified file.

## GUI Control Buttons

---

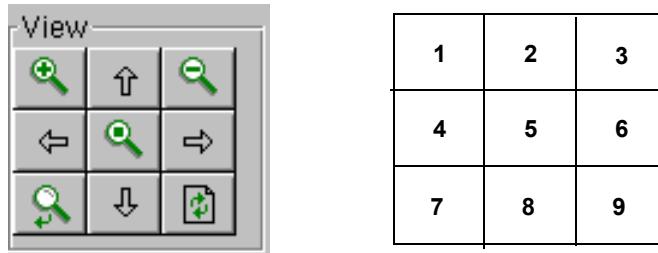
The control buttons on the right side of the GUI window invoke changes in the display or activate **RedHawk** commands. The name/function of each button is displayed when the cursor is placed over it. The control buttons are described in the following sections. Note that the entire GUI Control button panel can be detached using the menu function at the left side of the menu bar: **-> Windows -> Detach View Bar** and reattached using the same menu command.

### 'View' buttons

The 'View' buttons have the following functions that modify the image in the primary display area (see the key below, [Figure D-4](#)):

- 1 - "Zoom In" - makes image approximately 100% larger while maintaining the same center
- 2 - "Move Up" - moves center of view up at same magnification (image moves down)
- 3 - "Zoom Out" - makes image about one third smaller while maintaining the same center
- 4 - "Move Left" - moves view left at same magnification (image moves right)
- 5 - "Zoom to Top" - shows full design view
- 6 - "Move Right" - moves view right at same magnification (image moves left)
- 7 - "Zoom Back One Level" - goes back to previous view
- 8 - "Move Down" - moves center of view down at same magnification (image moves up)

9 - “Refresh” - updates display with the same view characteristics



**Figure D-4 ‘View’ Button Palette**

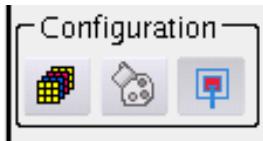
The **RedHawk** GUI functionality has the ability to change the Shift/Zoom factors that modify the rate of Zoom or the rate of vertical/horizontal Shift. The following menu command brings up a dialog to allow changes to the Zoom and Shift control functions:

#### **Windows->Preferences->Advanced**

The Zoom In/Out factor changes the rate of Zooming when the Zoom buttons are clicked on the View palette (upper left and right corner buttons). The Shift factor changes the amount the image shifts when the vertical and horizontal arrow buttons are clicked on the View palette. Note that these factors do not change the right mouse button Zoom function.

#### **Configuration’ buttons**

The configuration buttons (see [Figure D-5](#)) are described in the follow section.



**Figure D-5 Configuration buttons**

- “View Layers” (left button in the row) - displays the ‘Layer Select’ dialog box, as shown in Figure D-6, to allow modifying the display of objects on metal and via layers in the design, as well as pin instances. For each layer in the design you can choose whether to display its objects or not (**Invisible** button), change their display color (**Color** column), choose whether to display the objects in solid colors (**Fill** button) or in **Outline**. In locations in which there are objects on many layers at the same point, you can easily select an object on a lower layer by making only its layer selectable with the **Selectable** button. Using the **Pin inst** display buttons allows pins to be displayed by layer, independent of other objects. Because there are so many pins, pin instance display is Off by default to avoid interfering with the display of other objects. Zoom in to the instance before turning on pin display, in order to easily find the desired pin. For designs with more than one via model per layer, each via model is listed and can be displayed separately using the dialog. Buttons at the bottom of several selection columns allows **Net** or **Via** layers to be turned off and on as a group. Block pins also can be independently turned off and on using the **Show Block Pins** button at the bottom of the dialog.  
The **Base Layers** button allows you to choose which layers are to be marked as base layers. Once the base layers are marked, the Layer Selection Window removes them from the list, making layer selection much easier. To reassign the remaining layers from blue to red, select the checkbox **'Reassign colors'**.

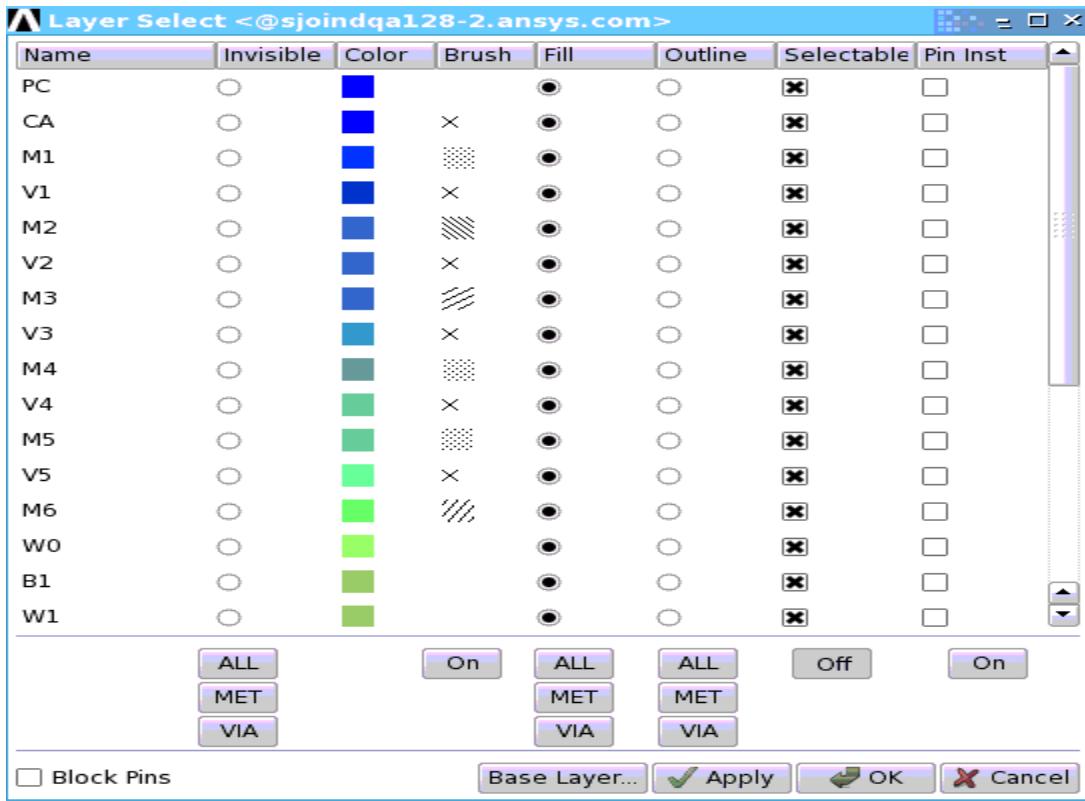


Figure D-6 Layer selection and object display dialog

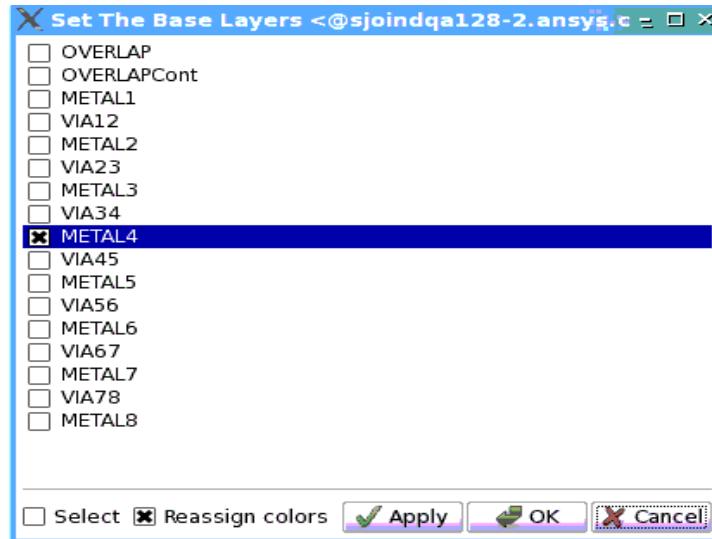


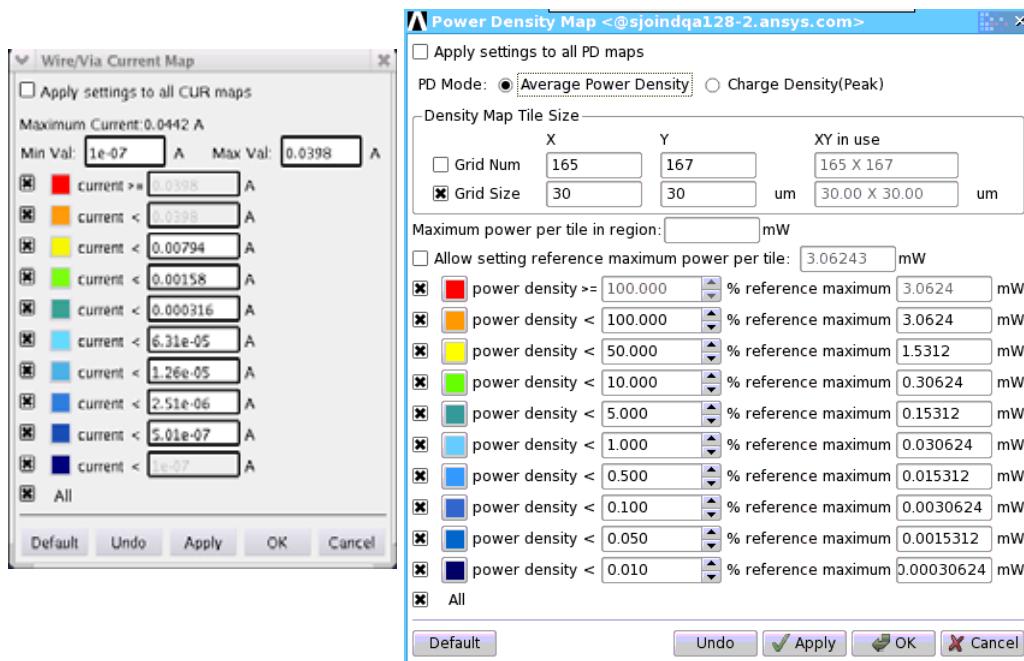
Figure D-7 Base Layers dialog

- “Set Color Range” (center button in row) - displays a Color Map dialog box based on the ‘View Results’ or menu selection. Figure D-8 shows examples of color maps for current and power density.

Note: Voltage Drop Color Map displays for multiple Vdd designs indicate the worst % drop of all nets selected in the View Nets dialog box. So multi-Vdd instances have colors indicating the domain with the worst % voltage drop.

The right-hand ‘mv’ column in the Voltage Drop Color Map dialog indicates only the % of the ideal voltage selected in the ‘VDD Domain’ drop down box. Hence selecting a different VDD Domain value changes the values in the ‘mv’ column, but does not change the worst % voltage drop color map.

Color maps may display either objects, a high parameter value, or a parameter density by tiled grid. Density displays use a grid composed of “tiles” that are 30 microns square by default. The size/number of the tiles can be changed using the ‘Grid Num’ or ‘Grid Size’ boxes on the dialog.



**Figure D-8 Current and power density color map dialogs**

- Colors can be assigned to specific ranges of parameter values, such as static or dynamic voltage drop, power, EM severity, current, frequency, or capacitance. Display colors can be changed by clicking on the color button and then selecting a new color from the ‘Pick a Color’ dialog box that appears. Ranges of parameter values also can be turned on or off independently with check boxes. The **Default** button allows easy return to the original color display values.
- “Show Power Pad” (right button in row) - toggles the display of power pads on and off (on by default). VDD pads are shown in orange and VSS pads in white.

### ‘View Results’ buttons

In general the specific color map dialog box for each of the following functions in the ‘View Results’ panel can be displayed by clicking on the ‘Set Color Range’ Configuration button described above.

#### First column ‘View Results’ buttons

- **SA** - “Show all” of layout - displays some or all objects in the design layout (individual layer grids, instances, vias and pins per layer), based on the selections in the ‘Layers’ dialog box. The dialog allows a choice of colors for displaying each type of object, as well as showing objects in

either solid or outline representations. By default pins are turned off, layer and instance objects are displayed solid.

You can view P/G nets in different colors for each domain. The color of the nets can be changed using the color range dialog box. To enable this feature, select the checkbox “Layout maps: Show net based color map” in the dialog box.

- **IR** - “IR Map” - displays the latest analysis results, static IR voltage drop or dynamic voltage drop analysis, based on view selections in the ‘Layers’ dialog box and the color ranges selected in the ‘Voltage Color Map’ dialog. Instances are shown in Fill representation by default, while layers and vias are always shown in Fill representation, in the color chosen to indicate their voltage drop category. At high magnifications high voltage drop values on individual instances can be observed. By default the ranges assigned to each color are equally divided in six steps between the ‘Maximum’ voltage drop and ‘Minimum’ voltage drop values specified.
- **EM** - “ElectroMigration Map” - displays results of electromigration analysis for static or dynamic current conditions, based on object view selections in the ‘Layers’ dialog box and the color ranges selected in the ‘ElectroMigration Color Map’ dialog. (Instances are turned off by default, and are shown in selected object colors when turned on.)  
The value displayed is the EM current tolerance, which is the computed fraction of current density in the wire relative to the maximum acceptable EM current density specified in the ‘metal’ keyword (‘EM’ parameter) in the \*.tech file. This value must be set for either static or dynamic condition of interest.
- **CUR** - “Current Map” - displays results of maximum current by node for static or dynamic conditions, based on the color ranges selected in the ‘Current Color Map’ dialog. Individual colors represent a selected percent of maximum current in the design. Instances are not displayed by default. If displayed, instances are shown in their selected object color.
- **TP** - “Thermal Profile” - displays a color-coded map of regional temperature for the layer specified.

#### **Second column ‘View Results’ buttons**

- **PD** - “Power density Map” - displays per instance power density by grid. Clicking on the ‘Set Color Range’ button (middle ‘Configuration’ button) displays the ‘Power Density Distribution Color Map’ dialog that allows you to define the tile/grid size, the power ranges to display and select a color for each. Grid-based results are calculated from Instance Power Map data (see below). User can switch between Power Density map and Charge density map by selecting the PD mode in the ‘Set Color Range’ ‘Configuration’ button. Charge Density map displays the peak charge density in units fc.
- **IPM** - “Instance Power Map” - displays results of power analysis for all instances. Instance power includes leakage power and short circuit internal power from the LIB files, plus the calculated switching power. Clicking on the ‘Set Color Range’ ‘Configuration’ button displays the ‘Instance Power Color Map’ dialog that allows you to define the instance power ranges to display and select a color for each . Static results are displayed for average power, and dynamic results are displayed for peak power. Before simulation, power map displays data from power calculation. Data for all instances are displayed; map data cannot be filtered by nets or P/G arcs.
- **CIP** - “Clock Instance Power Map” - displays results of power analysis for clock instances. Clicking on the ‘Set Color Range’ ‘Configuration’ button displays the ‘Instance Power Color Map’ dialog that allows you to define the clock instance power ranges to display and select a color for each.
- **LPM** - “Leakage Power Map” - displays leakage power for all instances. Clicking on the ‘Set Color Range’ ‘Configuration’ button displays the ‘Instance Power Color Map’ dialog that allows you to define the instance leakage power ranges to display and select a color for each.
- **IFM** - “Instance Frequency Map” - displays the frequency of each instance from STA file information. If there is no information in STA file on the frequency of the instance, it is assigned a frequency of zero. Clicking on the ‘Set Color Range’ ‘Configuration’ button displays the

'Frequency Color Map' dialog that allows you to select a color for each unique frequency and also displays a List of Clock Domain Frequencies.

#### Third column 'View Results' buttons - Capacitance display

Decap distribution (density in capacitance per tile, which can be defined in square microns or a defined number per design) of various types can be viewed using the capacitance display buttons described in the following section. Clicking on the 'Set Color Range' configuration button displays a 'Cap Distribution Color Map' dialog that allows you to select a color for each range of decaps for each capacitance display. By default, each capacitance colormap is scaled according to its own maximum value. Clicking on the button 'Allow setting max cap per tile' in the 'Cap Distribution Color Map' dialog and then selecting the desired maximum value to be used in all cap displays allows good discrimination of decap ranges and comparable color displays in all decap views.

- **DD** - "Decap Density Map" - displays usable decap density, which includes intrinsic decap and load capacitance, but not intentional decap (displayed by **IDD** button).
- **DEV** - "Device Decap Density Map" - displays original intrinsic device decap density.
- **LC** - "Load Cap Density Map" - displays load capacitance density.
- **IDD** - "Inserted Decap Density Map" - displays inserted intentional decap density for cells defined in the GSR.
- **PG** - "P/G Cap Density Map" - displays power/ground capacitance density.

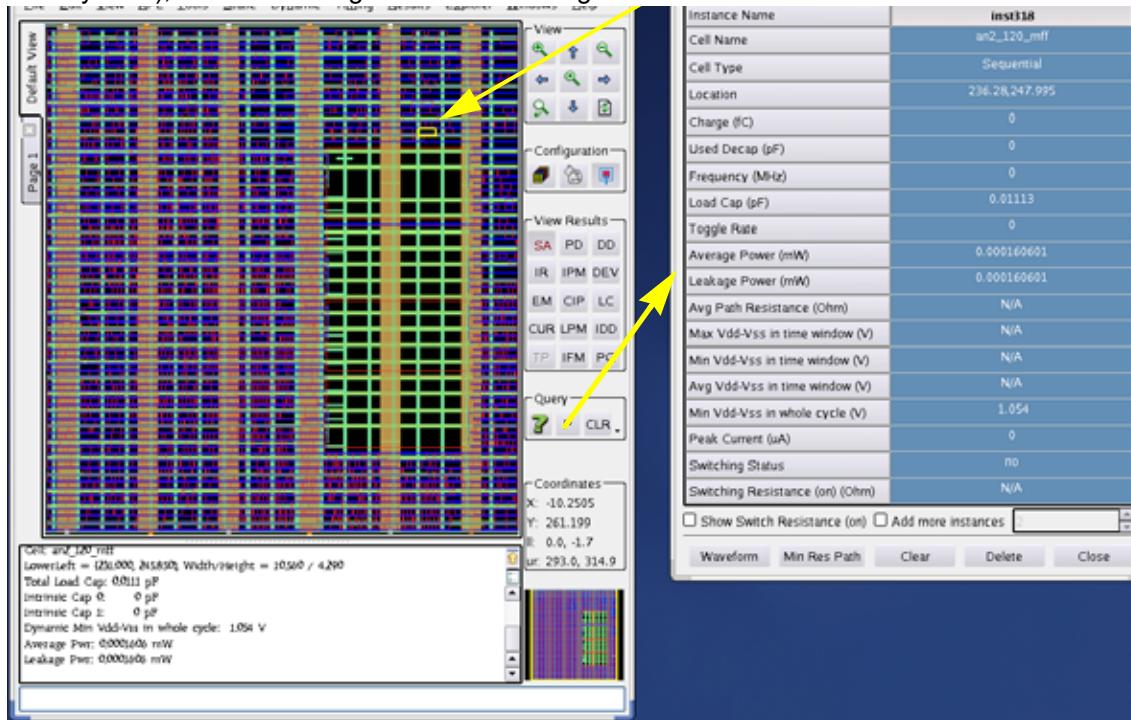
#### 'Query' buttons

- **?** - displays a 'Query Objects' dialog box that allows searching for design objects, specified either by name or by x,y location. Also allows highlighting instances for a specified cell, or multiple instances of objects that are listed in a specified file. The highlight color for objects found also can be selected.

To zoom in to show all instances of the same cell in the design, enter the cell name in the **Highlight Instances** for cell field in the Query dialog box, and then check **Multiple Instance Zoom** box. RedHawk zooms in to the smallest area that includes all instances of the cell

specified.

- **P** - displays an ‘Instance Property’ dialog, listing the properties of the selected instance (in yellow), as shown in Figure D-9. The dialog contents are described below.



**Figure D-9     Instance property display**

### Instance Property dialog contents for a selected instance

- Instance Name - name of instance
- Cell Name - instance lib/model name
- Cell Type - instance cell type, such as combinational, generic\_sequential, decap
- Location - instance x, y location coordinates
- Charge - charge consumed by the instance in the first cycle of switching, in femto Coulomb
- Intrinsic cap - Intrinsic (0/1) Cap value, to be consistent with capacitance values reported in the message window. For a decap cell, this denotes the decap capacitance values (pF).
- Frequency - clock frequency of the instance (MHz)
- Load Cap - instance output signal load cap in pF (from the largest Cload, if there are multiple output pins)
- Toggle Rate - the average switching rate in a cycle for the instance, where the maximum value is 1 for instances not in the clock network and 2 for instances in the clock network
- Average Power - total power consumed by the instance as calculated in power calculation (mW)
- Leakage Power - leakage power of instance taken from ./ib if POWER\_MODE LIB| MIXED GSR keyword settings are used, or from APL if POWER\_MODE APL setting is used (mW)
- Avg Path Resistance - average path resistance from all power and ground pins of the instance, as computed by gridcheck (Ohms)
- Max Vdd-Vss in time window - the maximum dynamic Vdd - Vss differential over the time window (V)

- Min Vdd-Vss in time window - the minimum dynamic Vdd - Vss differential over the time window (V)
- Avg Vdd-Vss in time window - the average dynamic Vdd - Vss differential over the time window (V)
- Min Vdd-Vss in whole cycle - the minimum dynamic Vdd - Vss differential over the whole cycle (V)
- Peak Current - the peak current from the switching current waveform of the instance (uA)
- Switching status - indicates whether instance is switching or not
- Switching Resistance (on) - for power gating instances, displays the On resistance value. Note that for instances that are not power gates, a value of 0 is shown (Ohms)
- Show Switch Resistance (on)** check box - when checked, displays On switching resistance for the selected power gate instance.
- Add more instances** button - Adds more columns to the Instance Property table for listing data and comparing additional selected instances.
- Waveform** button - displays xgraph or sv Vdd and Vss waveforms for the selected instance
- Min Res Path** button - runs gridcheck and displays the minimum resistance path. The Avg Path Resistance value also now can be displayed.
- **CLR** - clears all highlights, selections and rulers in the display (default function). This button also can be modified to execute two other functions, if selected with the drop-down menu:
  - **GIF** - brings up a Save File dialog that allows a GIF image of the design area to be saved to a selected directory and file.
  - **SNP** - brings up a Save File dialog that allows a GIF image of the entire window to be saved to a selected directory and file.
  - **RLR** - invokes the **Draw Ruler** function, the same as under the **Edit** menu.

### Coordinates readout area

- **X,Y** - describes the x,y coordinates of the cursor location
- **ll,ur** - describes the x,y coordinates of the lower left and upper right corners of the view displayed

### Full design view area

The small lower right corner image always displays the full design, with a highlight showing the primary display view.

### User-defined Shortcuts for GUI Buttons

RedHawk allows setting up user-defined key bindings for GUI buttons, such as Query, Property, and Clear. For example, to use “Ctrl + d” as a shortcut to the Query button; “Ctrl + p” as a shortcut to the Property button; and “Ctrl + c” as the Clear combo button, the following steps can be followed:

1. Enter the shortcut details, in the following format, in a keybinding definition file, such as *keybind.txt*.
 

```
ctrl<key>d Button/Ob
ctrl<key>p Button/Property
ctrl<key>c Button/QueryCombo
```
2. Then on the RedHawk GUI command line, enter the TCL command
 

```
import keybind keybind.txt
```

## GUI Menu

---

The **RedHawk** GUI menu functions are summarized in the following sections.

### File Menu

#### File -> Import Design Data

Displays an ‘Import Design Data’ dialog box to specify the design data files to be imported to perform Database setup. Check boxes for **Signal EM mode** and **ESD mode** are also available. The dialog box has fields for specifying GSR files for the design. If you choose to browse directories for the GSR files, use the **Select File Name** box. Selecting ‘OK’ then loads all specified design files.

#### File -> Import Database

Displays an ‘Import a DB’ dialog box to specify the database file to be imported. The dialog box also has fields for browsing directories and specifying database files, and also buttons for directory and file management, such as ‘Create Dir’, ‘Delete File’, and ‘Rename File’.

#### File -> Export Database

Displays an ‘Export ADB’ dialog box to specify the database file to be exported. The dialog box also has fields for browsing directories and specifying database files, and also buttons for directory and file management, such as ‘Create Dir’, ‘Delete File’, and ‘Rename File’.

#### File -> Import ESD DB

Displays an ‘Import ESD DB’ dialog to allow selection of a design to import for ESD analysis. Any time after design extraction an esdcheck database can be imported from a particular directory and ESD check dB file.

#### File -> Export ESD DB

Displays an ‘Export ESD DB’ dialog. The design can be exported to an ESD database file and directory any time after the esdcheck command has been performed.

#### File -> Import ECO

Displays an ‘Import ECO’ dialog box to specify the ECO file to be imported. The dialog box also has fields for browsing directories and specifying database files, and also buttons for directory and file management, such as ‘Create Dir’, ‘Delete File’, and ‘Rename File’.

#### File -> Export ECO

Displays an ‘Export ECO’ dialog box to specify the ECO file to be exported. The dialog also has fields for browsing directories and specifying database files.

#### File -> Import GUI Config

Displays an ‘Import GUI Config’ dialog box to specify the configuration file to be imported. The dialog box also has fields for browsing directories and specifying database files.

#### File -> Export GUI Config

Displays an ‘Export GUI Config’ dialog box to specify the configuration file to be exported. The dialog box also has fields for browsing directories and specifying database files.

#### File -> Playback

Displays a ‘Playback’ dialog box to specify the **RedHawk** session playback file to be loaded (of the form *redhawk\_<time\_stamp>.replay*), either a TCL or GUI script file. The dialog box also has buttons for selecting waiting time, ‘As is’, or a number of seconds. The Show Script Debugger Window can be displayed and the replay is then displayed in that window. You must know the date and time of the session you want to replay.

## File -> Exit

Exits RedHawk session and saves all associated files.

## Edit Menu

### Edit -> Undo

Reverses the action performed by the previous command invoked from the **Edit** menu. If the **Undo** command cannot be performed, a message is displayed.

### Edit -> Redo

Reverses the last **Undo** command performed.

### Edit -> Add Pad

Displays an ‘Add Power/Ground Pads’ dialog box to specify new power and ground pads. You must select a Power or Ground pad type, the appropriate layer, and click on one or more legal pad locations to add pads.

### Edit -> Delete Pad

Displays a ‘Delete Pad’ dialog box with a list of existing power and ground pads, their location and their layer. Specify which power or ground pads on the list should be deleted, and then click on ‘Delete Pad’.

### Edit -> Add Power Strap

Displays a ‘Power Strap’ dialog box to specify new power or ground straps. You can click on a tab for the type of net (‘Power Nets’, ‘Ground Nets’), and the name of the net in the list. Then in the ‘Strap Information’ box choose ‘Horizontal’ or ‘Vertical’ orientation, the x,y locations for Strap start and Strap end points, and the strap width (default 0.5 um). You can add multiple straps by x,y range and pitch, and then specify the appropriate metal layers and associated vias.

### Edit -> Edit Power Strap

Displays an ‘Edit Power Strap’ dialog box to specify modifications to existing power or ground straps. After clicking on the strap to be modified, information about it appears in the dialog box: ‘Net’, ‘Wire’ (layer), ‘Length’, ‘Width’, ‘Strap left lower x’ (x location at lower left end of strap), ‘Strap left lower y’ (y location at lower left end of strap) and ‘Resistance’. When the new description of the strap to be edited is displayed, the strap can be modified using the ‘Delete’ or ‘Change’ buttons.

### Edit -> Add Via

Displays an ‘Add Via’ dialog box to specify new vias to be added. After clicking on the location for a new via, information about the layers at that location appears in the ‘Select Metal Layers’ panel. After selecting the appropriate layers for the via, the available vias for the selected two layers appear in the ‘Available Via Type Lists’ panel. Select the appropriate via type and then execute the addition.

### Edit -> Delete Via

Displays a ‘Delete Via’ dialog box to specify existing vias to be removed from the design. Displaying vias on particular layers using the ‘View Layers’ button helps select the vias desired. After selecting the via to be deleted, its Net and Via Name are identified in the dialog box. If this is the desired via to be deleted, click on the **Delete** button to execute the command.

### Edit -> Add Decap Cell

Displays an ‘Add Decap Cell’ dialog box to specify new decoupling capacitors to be added. After selecting the power or ground net, click on the location for a new decap, and information about the layers at that location appears in the ‘Select Vdd’ or ‘Select Gnd’ panel. The available decap cells appear in the ‘Available Decap Cells List’ panel. Select the appropriate decap cell type and then execute the addition using the ‘Commit Adding’ button.

## Edit -> Delete Decap Cell

Displays an ‘Delete Decap Cell’ dialog box to specify decoupling capacitors to be deleted. After selecting decap cell to be deleted, its Name and Cell Type are identified in the dialog box. If this is the desired decap cell to be deleted, click on the **Delete** button to execute the command.

## Edit -> Edit Probe

Displays an ‘Edit Probe’ dialog that allows Adding and Deleting probe locations to get waveforms at locations not already available. To create a new probe, click on the desired location, select the layer name and type in a probe name in the dialog, and select ‘Add’. To Delete a probe, check the box next to its name and then ‘Delete’.

## Edit -> ESD Clamp ECO

### Clamp Cell

Displays an ESD Clamp Cell dialog box in which a cell can be selected and parameters specified for editing. Provides a method of performing the TCL command ‘eco add/delete clamp\_cell’ in the GUI interface.

### Clamp Instance

Displays an ESD Clamp Instance dialog box in which an instance can be selected and parameters specified for editing. Provides a method of performing the TCL command ‘eco add/delete clamp\_instance’ in the GUI interface.

## Edit -> Chip Partition

Displays an ‘Edit Chip Partition’ dialog box to allow the mouse to be used to create a non-uniform partitioning of the design for chip power modeling. Select either ‘Add Line’ or ‘Delete Line’ functions and select horizontal or vertical partitioning. You can click on the button ‘Clear partition lines when exit edit mode’ to remove partitioning upon terminating editing mode. Use the **Ctrl** key and right mouse button to select vertical and horizontal partition lines. Click on ‘Create Partitions’ button when the desired new partitions have been specified.

## Edit -> Ruler ->

### Draw Ruler

After turning on the ruler function with the check box, click the left mouse button on an x,y location on the design and drag to a second point on the design creates an x,y axis with lengths in microns. Continuing to move the cursor and click with the left mouse button defines additional measurement distances between points. The last point to be measured is determined by clicking on the right mouse button.

### Clear Ruler

All drawn rulers remain displayed until the ‘Clear ruler’ button is selected.

## Edit -> Single Key

This check box enables single key binding and operations if a key binding definition file has been imported. Note that the TCL command line is disabled when single key binding is in effect. You can import a bindkey definition file at the beginning of a **RedHawk** session using the TCL command:  
`import keybinding <bindkey_file>`.

## View Menu

### View -> Chip Layout Map

Performs the same “show layout view” operation as the **SA** button in the “View Results’ buttons’ panel. Does not change the area of the design viewed.

### **View -> Nets**

Displays a ‘Nets’ dialog box with lists nets by type selected by tab -- Power, Ground, Clock, or Signal (if extracted). The dialog allows you to select and highlight particular groups of nets by name or by type, and identifies internal P/G nets corresponding to any external net when users specify the external net name in the Nets dialog. Due to the large number of nets in a design, by default the signal and clock nets are not shown at all in the dialog.

In the Enabling Nets box, you can use pattern-based matching to get a list of signal or clock net names. Only the first 1000 matches are displayed. The global select/deselect button can be used for selecting/deselecting all nets in the design.

### **View -> Connectivity ->**

#### **Highlight Disconnected Instances**

This check box turns on highlighting of disconnected instances in the **SA** view, which is off by default. The outline colors of the instances indicate the status of their power/ground connections, as follows:

- yellow outline: both VDD and GND are unconnected
- light blue outline: GND is unconnected
- green outline: VDD is unconnected

The highlights can be cleared using the same menu command, the **CLR** menu button, or the TCL command 'select clearall'.

#### **Highlight Disconnected Wires/Vias**

This check box turns on highlighting of disconnected wires and vias in the **SA** view, which is off by default. The highlights can be cleared using the same menu command, the **CLR** menu button, or the TCL command 'select clearall'.

#### **List of Disconnected Instances**

Provides a list of instances that are considered to be disconnected from power or ground.

Disconnected instances can be selected in the table and then zoomed to in the layout display. All disconnected instances (either logically or physically disconnected) in the design are displayed in the GUI. Note that only instances that are logically connected to analyzed nets, but physically disconnected, are listed in the file *\*.PinInst.unconnect*.

You can select the disconnected instance locations in the list and click on the **Marker** button at the bottom of the window, which adds green markers at the problem locations in the GUI. Select the check-boxes “Export All Pages” and “Export Current page” to export all or current page of disconnected instances. The additional options in the marker button are:

- Export C: Export checked instances onto a file.
- Export L: Export entire list of instances onto a file
- Import: Import a list of instances into the GUI window

#### **List of Disconnected Wires/Vias**

You can browse through unconnected pins/wires/vias, similar to how you can browse through a list of shorts. From the list you can directly zoom to the location of a selected disconnected object. You can select the disconnected wire and via locations in the list and click on the **Marker** button at the bottom of the window, which adds green markers at the problem locations in the GUI. Disconnected wires and vias can also be listed for a specified region of the design by giving the corner coordinates of the desired region.

#### **Disconnected Transistor Pins**

To capture potential design problems in MMX blocks, provides a list of all transistors disconnected from the power and ground grid.

### **List of PG Weaknesses -> Instances**

Displays a ‘Report of PG Weakness Instances’ dialog box, which includes a sorted list of instances with the worst relative resistance as calculated by gridcheck, with the following information on each instance:

- Total Resistance: the total normalized P/G resistance seen by the instance, relative to the highest normalized P/G resistance  $(R_{inst} - R_{min})/(R_{max} - R_{min})$ , where  $R_{inst}$  is the sum of the effective VDD and VSS resistances for the instance, and  $R_{min}$  and  $R_{max}$  are the minimum and maximum  $R_{inst}$  values in the design
- VDD%: the VDD percentage of the normalized instance resistance ( $R_{vdd} * 100 / R_{inst}$ )
- VSS%: the VSS percentage of the normalized instance resistance ( $R_{vss} * 100 / R_{inst}$ )
- X,Y Location: the x,y location of the instance
- ArcID: identification of weak arc, if there are multiple domains
- Instance Name: the name of the instance

You can also highlight and zoom to any of the critical instances by selecting an instance in the list and then clicking on the **Zoom** button. You can also sort the table by Total Resistance, Vdd, or Vss order criteria. The next 1000 highest resistance instances on the list can be viewed by using the ‘Next’ button.

### **List of PG Weaknesses -> Transistor Pins**

Displays a ‘Report of Gridcheck Weakness for Transistor Pins’ dialog box, displaying a sorted list of normalized grid resistance by pin, with following information on each:

- Resistance: the total normalized P/G resistance seen by the pin, relative to the highest normalized P/G resistance  $(R_{inst} - R_{min})/(R_{max} - R_{min})$ , where  $R_{inst}$  is the sum of the effective VDD and VSS resistances for the pin, and  $R_{min}$  and  $R_{max}$  are the minimum and maximum  $R_{inst}$  values in the design.
- X,Y Location: the x,y location of the instance
- Instance Name: the name of the instance of the pin
- Pin Name: name of the pin

You can also select and zoom to any of the critical pins by selecting an instance in the list and then clicking on the **Zoom** button. The next 1000 highest resistance transistor pins on the list can be viewed by using the ‘Next’ button.

### **Missing Vias**

Displays a ‘Missing Vias’ dialog listing all P/G mesh intersections that represent possible missing via locations, including those on 45-degree geometry, compiled by surveying all grid intersections on all layers and noting the intersections that are not connected by vias. It also lists the voltage difference between upper and lower metal layers at potential via sites. The dialog allows you to step up or down the list and select a missing via location. The layers to be included in the list and the Threshold voltage difference can also be used to filter the list. Clicking on ‘Include Stack Via’ or ‘Include GDS cell’ check boxes allows you include those items. Clicking on the **Zoom** button highlights the missing via location for closer inspection. The missing vias list is sorted by voltage gradient---the higher the voltage difference between the metal layers above and below the via, the more impact adding this via would have on improving the voltage drop.

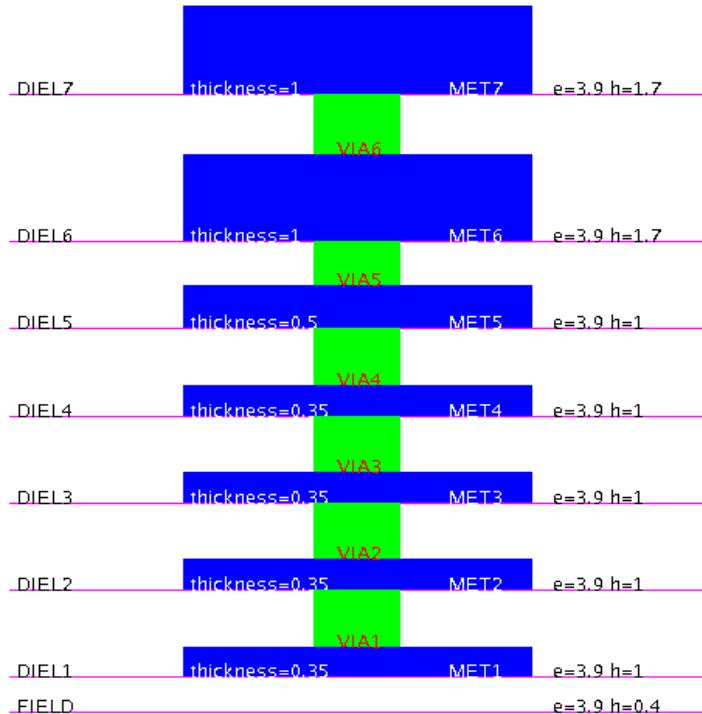
### **Shorts**

Displays a ‘Shorts’ dialog listing all P/G wires that may be improperly shorted. The dialog allows you to step up or down the list and select a short location. Clicking on the **Zoom** button highlights the short for closer inspection. You can select the shorts locations in the list and click on the **Marker** button at the bottom of the window, which adds green markers at the problem locations in the GUI. Click

**Synchronize** button to automatically fill the actual zoom coordinates (condition get -xy) as a filter in GUI. Shorts can also be listed for a specified region of the design by giving the corner coordinates of the desired region.

#### View -> Technology Layers

Displays a graphic showing the layers in the design and the key characteristics of each, including dielectric constant (e), metal thickness and full layer thickness (h), as shown in Figure D-10. If you use the right mouse button to drag a vertical line over one or more layers of particular interest, the view zooms to include just those layers. Clicking on the 'Full View' button then returns the view to the full view of all layers.



**Figure D-10 Technology File Viewer**

#### View -> Hierarchy Level

Displays a 'Hierarchy Level' dialog box that allows you to select the number of hierarchy levels for which design details are displayed. Hierarchy levels below that specified are shown as a "black box."

#### View -> EM Mode

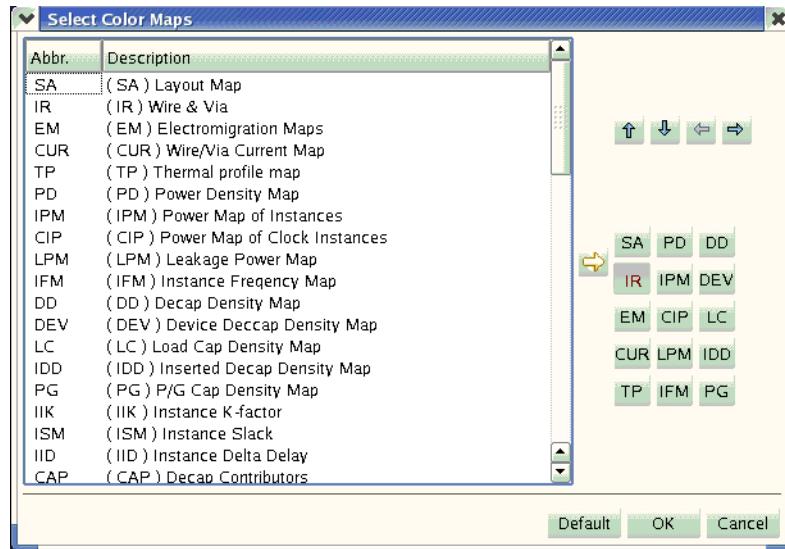
After invoking the 'perform emcheck' command for dynamic analysis, allows selection of the type of EM values to be displayed in the GUI, either DC, RMS, or Peak. For Signal EM analysis, **RMS** is default, and for Power analysis, **Peak** is default. Only **DC** data are available for Static analysis.

#### View -> Map Configuration ->

Note that for all color maps except the Layers Color map, a **Color Entries** box is displayed if the **Windows -> Preference -> Color Maps -> Allow More Color Spectrum** feature is selected. This allows increasing or decreasing the number of color steps used for the map display. Individual colors and ranges can be turned on and off independently.

#### View Button Customization

Allows you to change the type of colormaps displayed by the fifteen **View Results** buttons and also change the locations of the buttons, as shown in [Figure D-11](#). To change a **View Results** button to a new function, select the button icon in the “Select Color Maps” dialog, select the new function from the list displayed, and then click **OK**. You can also change the location of the buttons by using the arrows at the top of the dialog to move a selected button icon. The **Default** button returns the button functions to the way they were when the session was started.



**Figure D-11 View button customization dialog (Select Color Maps)**

### Layers Color Map

Displays a ‘Layers’ dialog box, the same as the ‘View Layers’ button in the ‘Configuration’ control panel. The dialog allows you to turn on/off and select color and outline or fill characteristics for displaying all metal and via layers, and the associated pin instances in the design. See [Figure D-6](#).

### IR Drop Color Map

Displays a static IR ‘Voltage Drop Color Map’ dialog box that allows you to select colors for a set of static voltage drop ranges in the design, as shown in [Figure D-12](#). You can choose to use the ‘maximum instance voltage drop’ in the design by clicking on the button. The default display divides the ranges equally between the specified minimum and maximum voltage drop settings. Individual colors and ranges can be turned on and off independently.

For DvD displays the title of the Voltage Drop Color Map dialog indicates the type of DvD displayed, such as “avgTW”, average effective voltage ( $V_{dd}-V_{ss}$ ) over the timing window, “minTW”, minimum effective voltage over the timing window, “maxTW”, maximum effective voltage over the timing window, or “minCYC”, minimum effective voltage over the clock cycle.

If the “Show absolute voltage drop map” box is checked, the map colors are keyed to absolute voltage drop values rather than percentage voltage drop values. **Note that the absolute voltage drop values are only valid for nodes associated with the selected reference VDD value, and are not valid for other domains with different VDD values.**

### Current Color Map

Displays a ‘Current Color Map’ dialog box that allows you to select colors for a set of current values from minimum to maximum in the design. The default display divides the ranges equally between the minimum and maximum current. Individual colors and ranges can be turned on and off independently.

### Power Density Color Map

Displays a ‘Power Density Distribution Color Map’ dialog box that allows you to select colors for a set of power density values as a percentage of maximum power density. Individual colors and ranges can be turned on and off independently. You can change the number/size of the tiles using the Grid Num and Grid Size windows to get more or less resolution in the density information.

### Leakage Power Density Color Map

Displays a ‘Leakage Power Density Color Map’ dialog box that allows you to select colors for a range of leakage power density values as a percentage of maximum leakage power density. Individual colors and ranges can be turned on and off independently. You can change the number/size of the tiles using the Grid Num and Grid Size windows to get more or less resolution in the density information.

### Instance Power Color Map

Displays an ‘Instance Power Density Color Map’ dialog box that allows you to select colors for a range of instance power values as a percentage of maximum instance power. Individual colors and ranges can be turned on and off independently.

### EM Color Map

After EM analysis is run, displays the ‘ElectroMigration Maps’ dialog box that allows you to select colors for a range of EM values as a percentage of EM maximum tolerance.

### Frequency Color Map

Displays the ‘Instance Frequency Map’ dialog box, which lists all clock domain frequencies in the design and allows you to select a color to display the nets driven by each frequency.

### Cap Density Color Map

Displays the ‘Device Decap Density Map’ dialog box, which allows you to select colors for a range of capacitance values as a percentage of maximum capacitance. You can change the number/size of the tiles using the ‘Grid Num’ and ‘Grid Size’ windows to get more or less resolution in the density information. Individual colors and ranges can be turned on and off independently. You also can make the maximum cap value per tile the same in all capacitance density displays by clicking on the ‘max cap value per tile’ button to make comparison between displays easier (**DD**, **DEV**, **LC**, **IDD**, **PG**).

### Toggle Density Color Map

Displays the ‘Toggle Density Map’ dialog box, which allows you to select colors for a range of average toggle rates per tile as a percent of maximum toggle density. Individual colors and ranges can be turned on and off independently. You can change the number/size of the tiles using the ‘Grid Num’ and ‘Grid Size’ windows to get more or less resolution in the density information.

### Instance Toggle Rate Color Map

Displays the ‘Toggle Map of Instances’ dialog, which allows you to select colors to display toggle rates per instance. Individual colors and ranges can be turned on and off independently.

### Instance Slack Color Map

Displays the ‘Instance Slack’ dialog box, which allows you to select colors for a range of slack values in ps using linear scaling from ‘Best Value’ to ‘Worst Value’.

### Instance K-factor Color Map

Displays the ‘Instance K-factor’ color map dialog box, which allows you to select colors for a range of delay values using linear scaling from ‘Minimum’ to ‘Maximum’ values. A button allows you to select maximum timing values from simulation to display.

### Instance Delta Delay Color Map

Displays an ‘Impact on Instance Delta Delay Color Map’ dialog box, which that allows you to select colors for a range of delay values. A button allows you to select default values to display. Individual colors and ranges can be turned on and off independently.

#### **Instance Peak Current Color Map**

Displays a ‘Instance Peak Current Map’ dialog box, which allows you to select colors for a range of current values. The maximum and minimum current values are shown. Individual colors and ranges can be turned on and off independently. A button allows you to select default values to display.

#### **Thermal Color Map**

After thermal analysis is run, displays a ‘Thermal Color Map’ dialog box, which that allows you to select the layer to be displayed and colors for a range of temperatures as a percent of the high temperature value. Changing the High and Low temperature values allows you to change the resolution of the temperatures displayed. Individual colors and ranges can be turned on and off independently.

### **View -> Power Maps ->**

Note: The Layer Map dialog does not control map displays. Since instances are the only objects displayed it can only turn instance display on and off.

Also, you can change the colors, scaling and tile grid size of the display of the following map commands by using the **View -> Map Configurations** command for the display.

#### **Power Density Map**

Performs the same operation as the **PD** button in the ‘View Results’ buttons’ panel.

#### **Power Map of Instances**

Performs the same operation as the **IPM** button in the ‘View Results’ buttons’ panel.

#### **Power Map of Clock Instances**

Performs the same operation as the **CIP** button in the ‘View Results’ buttons’ panel.

#### **Leakage Power Map**

Performs the same operation as the **LPM** button in the ‘View Results’ buttons’ panel.

#### **Leakage Power Density Map**

Displays a color map of leakage power density if leakage power data is available.

#### **Toggle Density Map**

Displays a color map indicating the average toggle rate per tile area in the design.

#### **Toggle Map of Instances**

Displays a color map indicating the toggle rate by instance in the design.

### **View -> Resistance Maps**

#### **Total Path Resistance**

Displays a color map indicating the relative strength of the Vdd and Vss grid connections to each instance, computed as a normalized “gradient”. The Total Resistance gradient for an instance is calculated as  $(R_{inst} - R_{min})/(R_{max} - R_{min})$ , where  $R_{min}$  and  $R_{max}$  indicate the best and worst grid connections among all instances in the design, and  $R_{inst}$  is the sum of the Vdd and Vss resistances connected to the instance. For instances with multiple Vdd/Vss connections, the sum of the highest

Vdd resistance plus the highest Vss resistance connected to the instance is selected as Rinst. In multiple Vdd/Vss designs use the **View-> Nets** command to select the nets to be included in the display. Note that turning off a net that does not represent the maximum Rinst value does not cause a change in the color display. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Total Path Resistance’ color map dialog, which allows you to select colors for the range of grid strengths as a percentage of Rmax-Rmin. For more details on the Gridcheck report, see [section "Examining Power/Ground Grid Weakness", page 4-53.](#)

#### **VDD Path Resistance**

Displays a color map indicating the relative strength of the VDD grids connected to each instance, as the instance VDD resistance gradient. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Resistance Color Map’ dialog, which allows you to select colors for a range of resistance gradient values as a percentage of the global Rmax - Rmin value.

#### **GND Path Resistance**

Displays a color map indicating the relative strength of the VSS grids connected to each instance, as the instance VSS resistance gradient. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘GND Path Resistance’ color map dialog, which allows you to select colors for a range of resistance gradient values as a percentage of the global Rmax - Rmin.

#### **Pin Path Resistance**

Displays a gridcheck pin map of P/G resistance for each pin geometry for instances. Pin geometries are color coded with their average resistance value. This feature is useful for analyzing the resistance maps of macro instances having multiple pin geometries per P/G pin. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Pin Path Resistance’ color map dialog, which allows you to select colors for a range of resistance gradient values as a percentage of the global Rmax - Rmin. If you check the Enable Layer Filtering box, and then click on the Layer Selection button, in the Layer Select dialog you can turn Pin Instance display on or off by individual layers.

#### **Wire/Via Path Resistance**

After gridcheck has been run, displays a color map of wire/via resistance. If gridcheck has not already run, executes gridcheck and then displays the resistance color map. Extraction must have been run before running gridcheck. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Wire/Via Path Resistance’ color map dialog, which allows you to select colors for the range of resistance values.

#### **Wire/Via Extractor Resistance**

Displays a color map of wire/via resistance based on extraction. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Wire/Via Extractor Resistance’ color map dialog, which allows you to select colors for the range of extraction resistance values. The Linear Scaling Initialization button divides all color steps into equal resistance percentage ranges.

#### **Effective Instance Resistance**

After resistance calculation has been run, displays a color map of effective (parallel) instance resistance from P/G grid to PLOC. If resistance calculation has not already been run, a reduced number of instances are selected based on the highest gridcheck resistance values to compute effective resistance, and then a color map of the computed effective resistances is displayed. Click on the ‘Set Color Range’ button in the Configuration group to obtain an ‘Effective Instance Resistance’ color map dialog, which allows you to select colors for the range of effective instance resistance values. The Linear Scaling Initialization button divides all color steps into equal resistance percentage ranges.

#### **Effective Wire/Via Resistance**

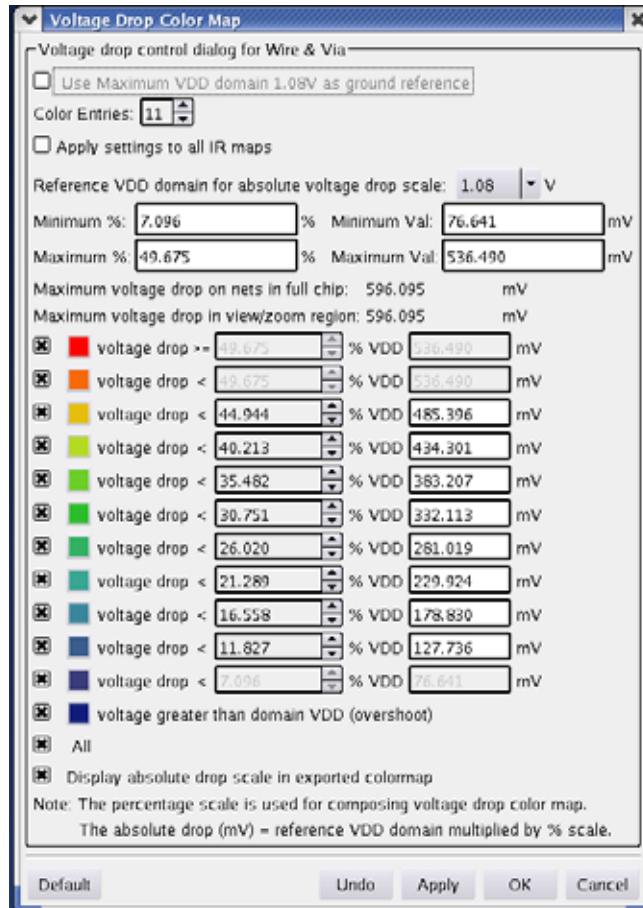
After resistance calculation has been run, displays a color map of effective (parallel) node resistance to PLOC. If res calc has not already run, a reduced number of nodes are selected based on the highest gridcheck resistance values to compute effective node resistance, and then a color map of the computed effective resistances is displayed. Click on the ‘Set Color Range’ button in the Configuration group to obtain an ‘Effective Wire/Via Resistance’ color map dialog, which allows you to select colors for the range of resistance values.

**Note:** Because of memory use considerations, if this color map is displayed after simulation, the color maps related to IR, EM and Current are no longer available unless simulation is rerun.

### View -> Voltage Drop Maps ->

#### Wire & Via

Displays the latest results from static IR voltage drop or dynamic voltage drop analysis for wires and vias. Performs the same operation as the **IR** button in the ‘View Results’ buttons’ panel. Click on the ‘View Layers’ button in ‘Configuration’ panel to select specific layers and objects from the ‘Layers’ dialog displayed. Click on the ‘Set Color Range’ button in the Configuration group to obtain an ‘Voltage Drop color map’ dialog, which allows you to select colors for each range of voltage drop values (see Figure D-12).



**Figure D-12    Voltage Drop color map dialog**

The Voltage Drop Color Map dialog allows you to modify the GUI display in several ways:

- Use Maximum Vdd domain - for multiple Vdd designs, you can select the domain to use as ground reference

- Color Entries - you can expand or reduce the number of color steps used to display voltage drop (default is 8 steps)
- Apply settings to all IR maps - check to synchronizes all IR color map settings
- Reference VDD domain for absolute voltage drop scale - for multiple Vdd designs, you can select the domain voltage used for calculating absolute voltage drop in mv
- Minimum % and Minimum Val - allows selection of minimum percentage drop or minimum voltage drop to be displayed
- Maximum % and Maximum Val - allows selection of maximum percentage drop or maximum voltage drop to be displayed
- voltage drop ranges - you can choose to display or not the voltage drop in each range, and also change the color used to display each one, and the actual percentage or absolute drop represented by each color
- voltage greater than domain VDD (overshoot) - voltage overshoot has a separate display color that can be selected. Note that this is only valid for wire voltage displays.

### Instance

Displays the latest results for static IR voltage drop or dynamic voltage drop analysis for instances only. The display and the associated color map are the same as described for 'Wire and Via' previously, except that the voltages are "effective" voltage values, Vdd-Vss.

### Pad Voltage Map

Displays a color map indicating the voltage on the pads in the design. The pad colors represent the average pad voltage for static IR analysis and the maximum voltage drop relative to nominal for dynamic analysis. A white pad outline indicates a ground pad and an orange pad outline represents a power pad. Click on the 'Set Color Range' configuration button to display the Pad Voltage Color Map dialog that indicates the pad voltage values, and allows you to change the colors, and also scale the size of the pads for better viewing.

### View -> Current Maps ->

#### Wire/Via Current Map

Displays a color map indicating the wire and via currents in the design, the same as using the **CUR** button in the 'View Results' buttons' panel.

#### Pad Current Map

Displays a color map indicating the pad currents in the design. The pad colors represent the absolute average current for static IR analysis and absolute peak current for dynamic analysis. A white pad outline indicates a ground pad and an orange pad outline represents a power pad. Click on the 'Set Color Range' configuration button to display the Pad Current Map dialog that indicates the pad current values and allows you to change the colors for each range. The Pad Size Scaling Factor panel allows you to scale (multiply) the size of the pads for better viewing. The Color Entries panel allows you to expand or reduce the number of color steps used to display pad current (default is 10 steps). You can also use the check box to Show Pad Current Value in Text in the primary GUI display.

#### Instance Peak Current Map

Displays the instance peak current distribution in the design in microAmps. Click on the 'Set Color Range' configuration button to display the Instance Peak Current Map dialog that indicates the peak current values and allows you to change the colors for each range. The Color Entries panel allows you to expand or reduce the number of color steps used to display pad current (default is 10 steps).

#### Supply Current Waveforms

Brings up a ‘Supply Current Waveforms’ dialog to choose current waveforms to be displayed. Both “Demand Current” (consumed current) and “Battery Current” (source current) can be plotted for user-selected nets. The View Nets button displays a Nets dialog that allows particular nets to be selected by type (Power, Ground, Clock, Signal).

- If only **Demand Current** is selected :
  - If all power nets are selected, total demand current is displayed.
  - If one or more of the power nets, but not all, are selected, then net-specific demand current waveforms are displayed. Ground net selections are treated the same.
- If only **Battery Current** is selected:
  - If all power nets are selected, total battery current is displayed.
  - If one or more of the power nets, but not all, are selected, then net-specific battery current waveforms are displayed. Ground net selections are treated the same.
- If both **Battery Current** and **Demand Current** are selected:
  - If all power nets are selected, total demand current and total battery current are displayed. Ground net selections are treated the same.
  - If one or more of the power nets, but not all, are selected, then net-specific battery and demand current waveforms are displayed. Ground net selections are treated the same.

The presimulation time period also can be displayed if the **Show Presim** button is clicked. A filename can be specified to write the selected waveforms into. The selected supply current is displayed on a color-coded X-graph plot when the Apply button is clicked.

### **View -> Electromigration Maps**

After EM analysis is run, displays results of electromigration analysis for static or dynamic current conditions, the same operation as the **EM** button in the ‘View Results’ buttons’ panel. Click on the ‘Set Color Range’ configuration button to display the Electromigration Maps dialog that indicates the EM tolerance values and allows you to change the colors for each range. The Color Entries panel allows you to expand or reduce the number of color steps used to display EM tolerance (default is 10 steps).

### **View -> Transistor Pin Maps ->**

#### **Transistor Pin Avg Voltage Over Simulation Time**

#### **Transistor Pin Max Voltage Over Simulation Time**

#### **Transistor Pin Min Voltage Over Simulation Time**

Displays the average, maximum, or minimum transistor pin voltage over the simulation time, in colors identifying their voltage values. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Transistor Pin Voltage Color Map’ dialog, which allows you to select colors for a range of voltage drop values as a percentage of maximum DvD, set the maximum and minimum DvD percentages, and also increase the size of the pins for ease of viewing using the ‘Pin Size Scale’ window.

#### **Voltage Drop Map**

Displays a map with pin colors indicating the worst-case transistor pin voltages. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Transistor Pin Voltage Drop Color Map’ dialog, which allows you to select colors for a range of voltage values as a percentage of the maximum DvD, set the maximum and minimum percentages, and also increase the size of the pins for ease of viewing using the ‘Pin Size Scale’ window.

#### **Transistor Pin Actual Resistance Map**

After ‘perform extraction’ is completed, this command generates a color map of actual resistance for all transistor pins in the MMX analysis. The resistance report is generated in the file *adsRpt/MMX/ <design>.mmx.res*, displaying transistor pins in colors identifying their actual resistance values. Click

on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Transistor Pin Actual Resistance Color Map’ dialog, which allows you to select colors for a range of resistance values as a percentage of the resistance gradient, set the maximum and minimum resistance gradient percentages, and also increase the size of the pins for ease of viewing using the ‘Pin Size Scale’ window. For run-time reasons only the highest 10,000 pin resistances are calculated, based on P/G grid weakness computations. To change the maximum number calculated, use the GSR keyword ‘MMX\_RES\_MAP\_LIMIT <pin\_limit>’.

### Transistor Pin PG Weakness Map

Displays a P/G pin weakness ranking colormap, in colors indicating the relative resistance of all transistor pins from highest to lowest. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Transistor Pin PG Weakness Color Map’ dialog, which allows you to select colors for a range of resistance values as a percentage of the resistance gradient, set the maximum and minimum gradient percentages, and also increase the size of the pins for ease of viewing using the ‘Pin Size Scale’ window.

### Transistor Pin Current Map

Displays a map with pin colors indicating average transistor pin currents. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Transistor Pin Current Color Map’ dialog, which allows you to select colors for a range of current values as a percentage of the maximum pin current, set the current range for the specified color, and also increase the size of the pins for ease of viewing using the ‘Pin Size Scale’ window.

### Transistor Pin Decap Map

Displays a map indicating the locations of transistor pin decaps.

## View -> Dynamic Instance DvD ->

Note: to understand the color maps associated with this menu group, see [Figure D-12](#) and the detailed explanation of the color map dialog for all menu commands in this group.

### Max Vdd-Vss Over Time Window

Displays the maximum dynamic Vdd - Vss differential over the time window, for the nets selected in **View -> Nets**. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Voltage Drop Color Map’ dialog, which allows you to select colors for a range of voltage drop values as a percentage of the maximum voltage. Instances are shown in Fill style in the colors chosen to indicate their voltage drop category. At high magnifications the maximum Vdd - Vss difference values for individual instances can be observed.

### Min Vdd-Vss Over Time Window

Displays the minimum dynamic Vdd - Vss differential over the time window, for the nets selected in **View -> Nets**. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Voltage Drop Color Map’ dialog, which allows you to select colors for a range of voltage drop values as a percentage of the maximum voltage. Instances are shown in Fill style in the colors chosen to indicate their voltage drop category. At high magnifications the minimum Vdd - Vss difference values for individual instances can be observed.

### Avg Vdd-Vss Over Time Window

Displays the average dynamic Vdd - Vss differential over the time window, for the nets selected in **View -> Nets**. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Voltage Drop Color Map’ dialog, which allows you to select colors for a range of voltage drop values as a percentage of the maximum voltage. Instances are shown in Fill style in the colors chosen to indicate

their voltage drop category. At high magnifications the average Vdd - Vss difference values for individual instances can be observed.

#### **Min Vdd-Vss Over Whole Cycle**

Displays the minimum dynamic Vdd - Vss difference over the whole cycle, for nets selected in **View -> Nets**. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Voltage Drop Color Map’ dialog, which allows you to select colors for a range of voltage drop values as a percentage of the maximum voltage. Instances are shown in Fill style in the colors indicating their voltage drop category. At high magnifications the whole cycle average Vdd - Vss difference values for individual instances can be observed.

#### **View -> Decap Maps ->**

Note: for all menu commands in this group you can change the colors, scaling and tile grid size of the display of the following map commands by using the **View -> Map Configurations -> Cap Density Color Map** command for the display, or click on the ‘Set Color Range’ Configuration button. The Color Entries panel allows you to expand or reduce the number of color steps used to display cap value (default is 10 steps).

#### **Decap Density Map**

Displays usable decap density, which includes intrinsic decap and load capacitance, but not intentional decap (displayed by **IDD** button, the same operation as the **DD** button in the “**View Results**” buttons’ panel. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Decap Density Map’ dialog, which allows you to select colors for a range of capacitance values as a percentage of the maximum cap value.

#### **Device Decap Density Map**

Displays original intrinsic device decap density, the same operation as the **DEV** button in the “**View Results**” buttons’ panel. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Device Decap Density Map’ dialog, which allows you to select colors for a range of capacitance values as a percentage of the maximum cap value.

#### **Load Cap Density Map**

Displays load capacitance density, the same operation as the **LC** button in the “**View Results**” buttons’ panel. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Load Cap Density Map’ dialog, which allows you to select colors for a range of capacitance values as a percentage of the maximum cap value.

#### **P/G Cap Density Map**

Displays power/ground capacitance density, the same operation as the **PG** button in the “**View Results**” buttons’ panel. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘P/G Cap Density Map’ dialog, which allows you to select colors for a range of capacitance values as a percentage of the maximum cap value.

#### **Inserted Decap Density Map**

Displays inserted intentional decap density for cells defined in the GSR, the same operation as the **IDD** button in the “**View Results**” buttons’ panel. Click on the ‘Set Color Range’ button in the Configuration group to obtain a ‘Inserted Decap Density Map’ dialog, which allows you to select colors for a range of capacitance values as a percentage of the maximum cap value.

#### **Decap Contributors**

Displays a ‘Decap Contributors’ dialog box that allows you to choose which type, or combination of types, of decoupling capacitance to display:

- ‘Intrinsic capacitance available during simulation’ - native decaps of regular cells that *do not switch* during the simulation period.

- ‘Intrinsic cap not available during simulation’ - native decaps of regular cells that *switch during the simulation period* and do not perform as decap cells.
- ‘Intentional cap’ - decaps of special cells defined in the DECAP\_CELL GSR keyword
- ‘Power and ground routing cap’ - decaps of power and ground routing grids
- ‘Load cap available during simulation’ - capacitance of load (wire and gate capacitance) for cells that are not switching and are at high output state (logic state 1 and stable). They are connected to VDD through an open P device, contributing to total decap.

**View -> ESD Resistance Lists ->**

Displays a dialog listing resistance and other key parameters for bumps included in completed ESD resistance checks. The dialogs displayed by these commands list key bump and clamp information, such as Resistance, Bump/clamp ID, Net, x,y location, Layer, and Direction, as well as Rule Name, Loop Count, for the following types of lists:

[List of Clamp-to-instance Arc](#)

[List of Clamp-to-instance Loop](#)

[List of Bump-to-Instance](#)

[List of Bump-to-Bump Para](#)

[List of Bump-to-Clamp](#)

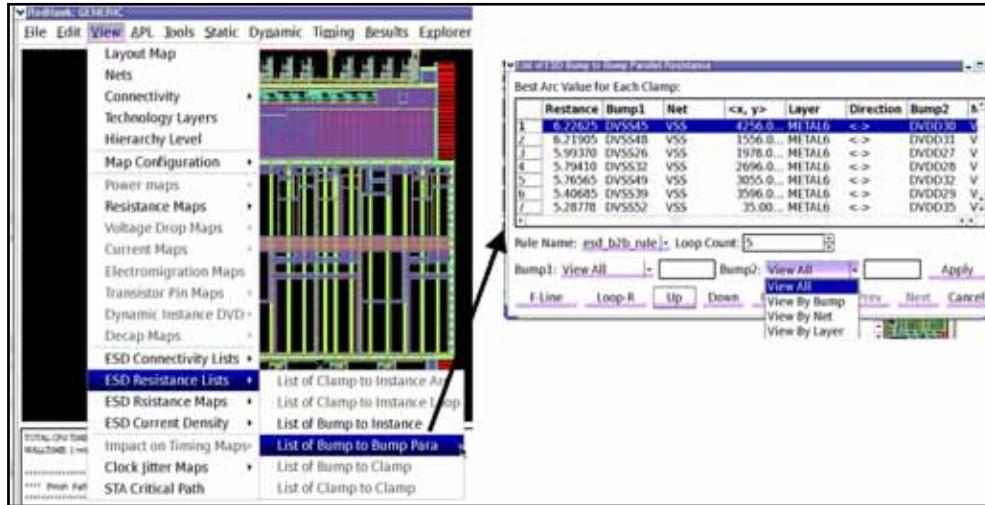
[List of Clamp-to-Clamp](#)

Select the desired path to highlight the resistance path between selected bumps or clamps. Click on the **F-line** button to draw flight lines between the bump or clamp and the associated instance in the GUI. You can also zoom to the instance to see the associated resistance values.

In addition, you can change the value of the resistance threshold and color configuration using the color map **Configuration** button. The color map dialog allows easily changing the threshold of various ESD checks on the fly and analyzing the corresponding ESD maps.

You can also easily trace the minimum resistance ESD paths to facilitate detailed analyses on the routing of failed ESD paths by clicking the **SPT** button at the bottom of the resistance list window. Some resistance lists are two-level (such as bump-to-bump), in which case the **SPT** button is available only in the second-level list. For example, the SPT button is in the second-level table for B2B parallel resistances. Choose **View -> ESD Resistance List -> List of Bump-to-Bump Para**, as shown in Figure D-13. Then select a parallel resistance path, and click the **Loop-R** button. The second-level window is displayed, with all the loops for this bump-to-bump pair. Choose a loop in the loop

resistance list and click the **SPT** button to view the minimum resistance route traced between the ESD paths.



**Figure D-13 List of B2B parallel resistance checking**

The selected tracings are accumulative. To clear them, click **CLR** on the right side of the **RedHawk** GUI.

#### List of P2P-R

Displays a P2P-R List dialog providing the Rule Name, EffR and the From/To test points for each Test Name run.

#### View -> ESD Clamp Lists

Displays a List of Clamps dialog table with a list of clamp pins by instance and a list of clamp pin pairs by instance.

#### View -> ESD Resistance Maps ->

Displays color maps showing ESD characteristics, integrated with the list of ESD resistances, so the map varies according to the test selected. In addition, **RedHawk** optionally draws flight lines and traces the minimum resistance path for the ESD path selected. You can also change the threshold value of the ESD checks on the fly and then analyze the corresponding ESD map. For each type of color map click on the 'Set Color Range' configuration button to display the associated ESD Color Map dialog, which specifies and allows you to change the colors, the resistance ranges, and the resistance thresholds. The loop count can also be changed, which specifies the maximum number of clamps to traverse between bump pairs.

#### Clamp to Inst Arc Resistance

Displays a color map and flight lines indicating the resistance for clamp to instance arcs, and indicates by color those paths that are less than or exceed the specified resistance threshold (green and red by default).

#### Clamp to Inst Loop Resistance

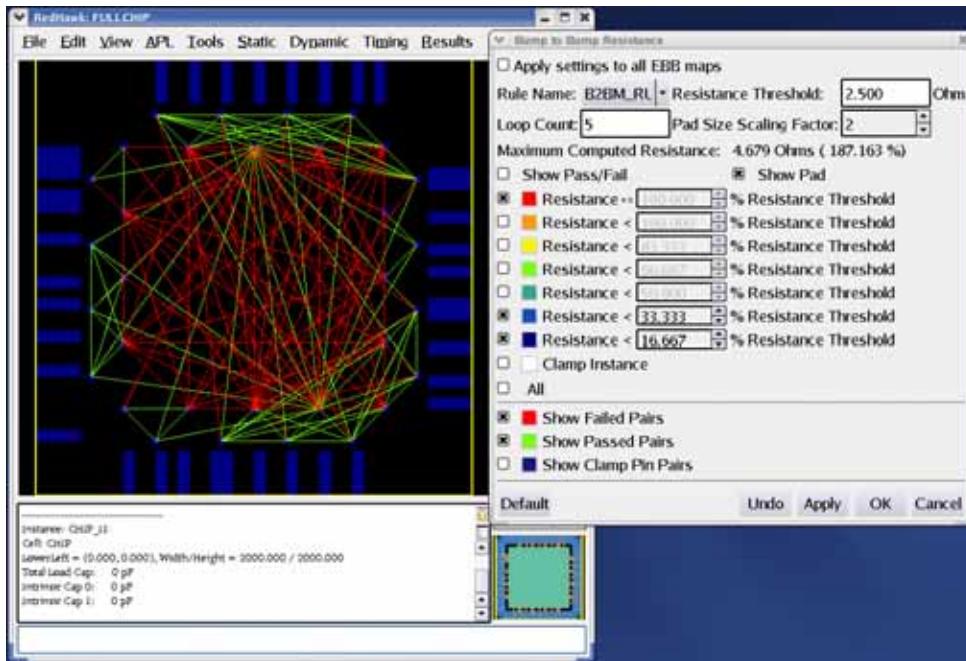
Displays a color map and flight lines indicating the resistance for clamp to instance loop paths, and indicates by color those paths that are less than or exceed the specified resistance threshold (green and red by default).

### Bump to Inst Resistance

Displays a color map and flight lines indicating the resistance for bump to instance paths, and indicates by color those paths that are less than or exceed the specified resistance threshold (green and red by default).

### Bump to Bump Resistance

Displays a color map and flight lines indicating the resistance between bump pairs, and indicates by color those paths that are less than or exceed the specified resistance threshold (green or red by default). See Figure D-14, which shows a B2B resistance color map and the associated color map dialog that allows you to change the way the B2B resistance check information is displayed.



**Figure D-14 B2B resistance color map**

### Bump to Clamp Resistance

Displays a color map and flight lines indicating the resistance for bump to clamp paths, and indicates by color those paths that are less than or exceed the specified resistance threshold (green and red by default).

### Clamp to Clamp Resistance

Displays a color map and flight lines indicating the resistance for clamp to clamp paths, and indicates by color those paths that are less than or exceed the specified resistance threshold (green and red by default).

See also [section "Resistance Checking for B2B, B2C, and C2C Rules", page 16-448](#), for more details on these checks.

**View -> ESD Current Density->**

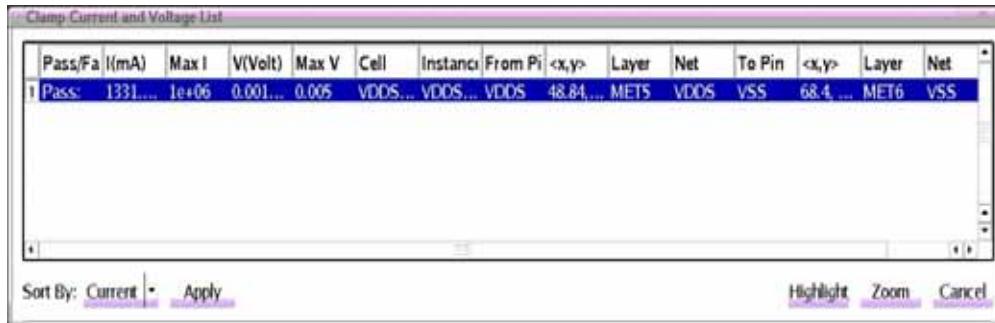
### Current Density Test List

Displays a dialog box with a list of completed CD tests by rule Name and the key results, such as Worst EM %, Max Clamp current, Effective resistance and From/To connections. You can load voltage

and current maps for any previously-performed ESD CD check by Rule name, and set EM caching and the EM scale factor to test against. You can Highlight a selected test flight line.

### Clamp Current/Voltage List

Displays a list of clamp current and voltage values as specified in the rule file or clamp file, including cell and instance names, layers and From/To connections, as shown in Figure D-15.

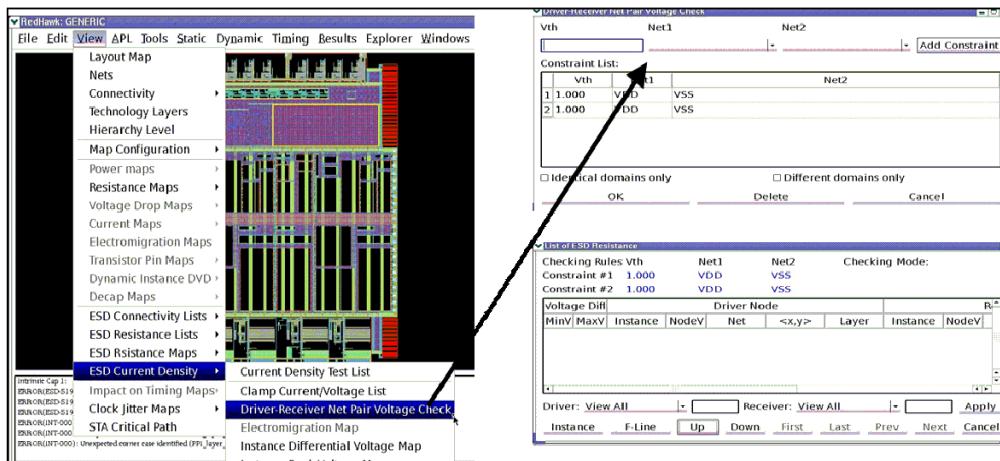


The dialog box is titled "Clamp Current and Voltage List". It contains a table with columns: Pass/Fail, I(mA), Max I, V(Volt), Max V, Cell, Instance, From Pin, <x,y>, Layer, Net, To Pin, <x,y>, Layer, Net. One row is highlighted in blue, showing: Pass, 1331..., 1e+06, 0.001..., 0.005, VDD..., VDD..., VDD, 48.84..., MET5, VDD, VSS, 68.4..., MET6, VSS. At the bottom, there are buttons for "Sort By: Current", "Apply", "Highlight", "Zoom", and "Cancel".

**Figure D-15** Clamp current and voltage failure list

### Driver-Receiver Net Pair Voltage Check

Checks and displays a list of driver/receiver net pairs that exceed voltage thresholds checks for ESD current density tests, as shown in The dialog allows specifying the maximum allowed voltage difference and the nets to be checked. Node pairs that exceed the specified voltage threshold f are listed. You can then highlight the instances of selected node pairs and draw flight lines between nodes.



**Figure D-16** Driver-receiver net pair voltage checking

### Electromigration Map

Displays EM results in the GUI as a color coded map of % EM tolerance. To see the EM color values, click on the Color map Configuration button to display the Electromigration Color Map dialog. You can change the color display and the EM tolerance ranges with this dialog. Individual color ranges can be turned off using the left side check boxes.

### Instance Differential Voltage Map

Displays the Differential voltage values for instances in the GUI as a color coded map. To see the Differential voltage color values, click on the Color map Configuration button to display the ESD Instance Differential Voltage Color Map dialog. You can change the color display and the voltage tolerance ranges with this dialog. Individual color ranges can be turned off using the left side check boxes.

#### **Instance Peak Voltage Map**

Displays the Peak voltage values for instances in the GUI as a color coded map. To see the Peak voltage color values, click on the Color map Configuration button to display the ESD Instance Peak Voltage Color Map dialog. You can change the color display and the voltage tolerance ranges with this dialog. Individual color ranges can be turned off using the left side check boxes.

#### **Pad Current Map**

Displays the Pad current values the GUI as a color coded map. Note that the Show Pad Power Configuration button (right) must be clicked to display the power pads. To see the Pad current color values, click on the Color map Configuration button to display the ESD Pad Current Color Map dialog. You can change the color display and the current tolerance ranges with this dialog. Individual color ranges can be turned off using the left side check boxes.

#### **Wire/Via Current Map**

Displays the Wire and Via current values in the GUI as a color coded map. To see the Wire and Via current values, click on the Color map Configuration button to display the Current Color Map dialog. You can change the color display and the current tolerance ranges with this dialog. Individual color ranges can be turned off using the left side check boxes.

#### **Wire/Via Voltage Map**

Displays the Wire and Via voltage values in the GUI as a color coded map. To see the Wire and Via voltage values, click on the Color map Configuration button (middle) to display the ESD DC Wire and Via Voltage Color Map dialog. You can change the color display and the voltage tolerance ranges with this dialog. Individual color ranges can be turned off using the left side check boxes.

### **View -> Impact on Timing Maps ->**

#### **Instance Slack**

Displays a map of slack by instance, same function as **ISM** ‘Impact on Timing’ button. Using the Colormap button you can adjust the color display of the slack

#### **Instance K-factor**

Displays a map of instance K-factor, the same function as **IIK** ‘Impact on Timing’ button. K-factor is the localized linear rate of change of delay relative to DvD.

#### **Instance Delta Delay**

Displays a map of instance delta delay, the same function as **IID** ‘Impact on Timing’ button.

### **View -> Clock Jitter Maps (PJX)**

#### **Rise Period Map**

Displays a map of period jitter for rising waveforms of clock instances in the design.

#### **Fall Period Map**

Displays a map of period jitter for falling waveforms of clock instances in the design.

#### **Rise C-to-C Jitter**

Displays a map of cycle-to-cycle jitter for rising waveforms of clock instances in the design.

### Fall C-to-C Jitter

Displays a map of cycle-to-cycle jitter for falling waveforms of clock instances in the design.

#### **View -> Clock Jitter Maps ->**

##### Rise Period Map

Displays a map of period jitter for rising waveforms of clock instances in the design.

##### Fall Period Map

Displays a map of period jitter for falling waveforms of clock instances in the design.

##### Rise C-to-C Jitter

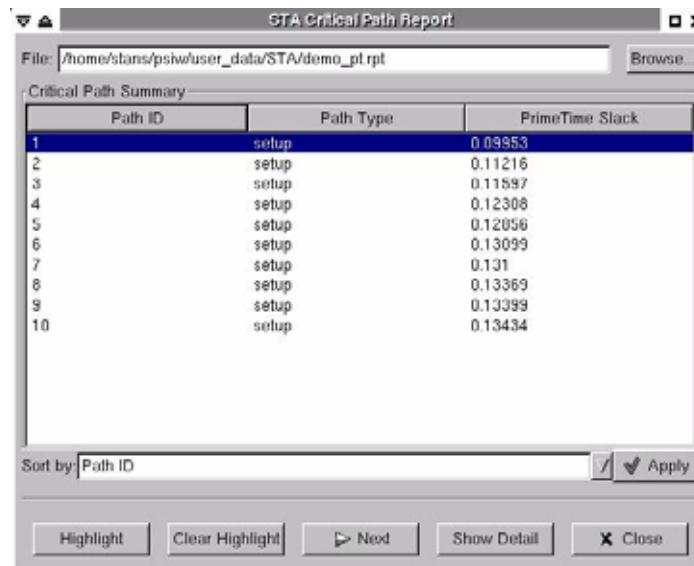
Displays a map of cycle-to-cycle jitter for rising waveforms of clock instances in the design.

##### Fall C-to-C Jitter

Displays a map of cycle-to-cycle jitter for falling waveforms of clock instances in the design.

#### **View -> STA Critical Path**

Displays a dialog box to specify an STA critical path report to import, allowing you to review the top critical timing paths in the design. One or more STA paths from the imported STA file can be selected to highlight in the layout view, as shown in Figure D-17.



**Figure D-17 Imported list from STA critical path report**

### Tools Menu

#### **Tools -> Lowpower ->**

##### Power -> Calculate

Executes power calculation for low power analysis flow.

##### Power -> Import

Displays an ‘Open Directory’ dialog box to specify a previously-executed low power calculation directory to be imported.

#### **Network Extraction**

Displays a ‘Power Ground Extraction’ dialog box to specify the core R,L,C parameters to extract for specified VDD, GND, CLK and SIGNAL nets.

#### **Pad Wirebond Package Constraints**

Displays a ‘Pad Wirebond Package Constraints’ dialog box to specify the key R, L, C parameters for pad and wirebond/bump power and ground, and package subcircuit models. Any of the elements can be specified to be ignored in the model.

#### **Ramp Up Analysis**

Displays a ‘Low Power Ramp Up Analysis’ dialog box that allows selection of the desired simulation parameters ‘Piecewise Cap Model’ file and Simulation Time’ for ramp-up analysis.

#### **EM Check**

Executes the ‘perform emcheck’ command.

#### **Tools -> Signal EM ->**

##### **Power -> Calculate**

Executes power calculation for Signal EM dynamic voltage drop analysis flow.

##### **Power -> Import**

Displays an ‘Import’ dialog box to specify a previously-executed power calculation directory to be imported.

#### **Network Extraction**

Executes the “perform extraction -signal”.

*Note that for this command to be valid, the ‘setup analysis\_mode signalEM’ command must be in the command script in the setup phase.*

#### **EM Analysis**

Executes the ‘perform analysis -signalEM’ command.

#### **EM Check**

Executes the ‘perform emcheck’ command.

#### **Tools -> Chip Power Model ->**

##### **Power -> Calculate**

Executes power calculation for the CPM dynamic voltage drop analysis flow.

##### **Power -> Import**

Displays an ‘Open Directory’ dialog box to specify a previously executed power calculation directory to be imported.

#### **Network Extraction**

Displays a ‘Power Ground Extraction’ dialog box to specify the core R,L,C parameters to extract for VDD and GND nets.

### Chip Power Modeling

Displays a ‘Generate Chip Power Model’ dialog box to specify the Output filename, the Flip Chip or Wirebond design type, and the number of partitions desired in the X and Y directions. Click on the Model Option ‘Reuse’ button to reuse the model. Select either Vectorless or VCD type flow.

### Tools -> PJX Timing Paths ( was Fullchip Critical Paths PPX )

Displays slack analysis results from the fullchip critical path command, the same as the TCL command “perform timing\_check -slack” (beta).

### Tools -> Clock Jitter ->

#### Clock Network Summary

After setting up and running Clock Tree Jitter Analysis (see [section "PJX Clock Tree Jitter Sign-Off Analysis", page 8-177](#), for the procedure), a summary of key parameters of the clock network is displayed in the ‘Clock Network Summary’ window, such as Clock Root, Frequency, Leafs, Gates, Max Level, Min Level, and Average Effective Vdd. Key details of any list entry can be displayed by selecting the entry and clicking on the Show Details button at the bottom of the window. A Clock Tree Details dialog is displayed, listing the Pin Name, Cell Type and Level for elements of the selected tree. You can search for a particular Pin Name in the list, or Zoom to the instance selected on the list.

#### Fullchip Jitter -> Analyze

Invokes fullchip jitter analysis, the same as using the TCL command “perform jitter -fullchip”.

#### Fullchip Jitter -> Long Term Jitter Report

Displays Long Term Jitter analysis results (maximum variation in clock arrival time across the simulation). Results are organized by clock domain.

#### Fullchip Jitter -> Long Term Cycle-2-Cycle Jitter Report

Displays Long Term Cycle 2 Cycle Jitter results (maximum variation at clock arrival time in consecutive cycles ). Results are organized by clock domain.

#### Fullchip Jitter -> Long Term Jitter Bottleneck Report

Displays a report of worst jitter weakness locations across all clock domains.

#### Sign-off Jitter -> Simulate

Displays a Clock Jitter Analysis dialog to select invocation parameters for **RedHawk** dynamic analysis and timing analysis.

**Dynamic Analysis.** Allows a choice of either VCD-based True Time or vectorless dynamic analysis, as well as the ability to Store Clock Instance Waveform Only, and the choice of simulation Start Time, End Time, Presim Time, and Presim Time Step multiplication Factor.

**Timing Impact Analysis.** extracts the multiple-cycle power ground-waveforms and performs a multiple cycle Spice simulation to obtain clock jitter results. The dialog allows the ability to Skip Dynamic Analysis if it has been run previously, and to specify the Configuration File Name and directory. After selecting the desired parameters, select the OK button to start execution of dynamic analysis, if needed, and timing Analysis.

#### Sign-off Jitter -> Clock Jitter Report

After running clock jitter analysis, use this command to check and analyze the sign-off clock jitter results and waveforms through the GUI interface. See [section "Clock Tree Jitter Results", page 8-187](#).

#### Sign-off Jitter -> Clock Jitter Bottleneck Report

Displays a Clock Jitter Bottleneck Report dialog with a jitter bottleneck ranking list for determining the worst jitter weakness locations. Fixing jitter on pins with higher rankings maximizes the benefit in reducing the overall jitter problem.

## Tools -> PathFinder SOC

### Network Extraction

Displays a ‘Power Ground Extraction’ dialog box to specify the core R parameters to extract for VDD and/or GND nets.

### Resistance & Current Density Check

*Note: To activate this command you must have a ‘setup analysis\_mode esd’ command in your command file, which requires a PathFinder license.*

Displays a PathFinder ESD Analysis dialog box to specify parameters for executing ESD resistance and current density checks.

**ESD Rule File (required):** describes the type of the checking, the thresholds, and other settings related to rule selected.

**Select File:** select a rule or clamp file specified in GSR

**Clamp File(optional):** describes the information related to clamps

**Output directory (optional) :** default is *adsRpt/ESD*

**Number of threads :** number of parallel jobs to be run.

**Job Count:** specifies number of jobs (default 1, maximum 4)

**Append Mode** button: if enabled, appends results to the output files instead of overwriting the previous output

**Incremental Mode** button: if enabled, performs incremental instance and macro point selection,

**Save Clamp DB:** saves clamp database

### P2P Resistance & Current Density Check

*Note: To activate this command you must have a ‘setup analysis\_mode esd’ command in your command file, which requires a PathFinder license.*

Displays a dialog box to execute three point-to-point resistance and current density checks (see Figure D-18):

- perform esdcheck (current density). See the TCL command and options description, page D-787.
- perform res\_calc (effective resistance). See the TCL command and options description, page D-795.
- perform min\_res\_path (SPT). See the TCL command and options description, page D-790.

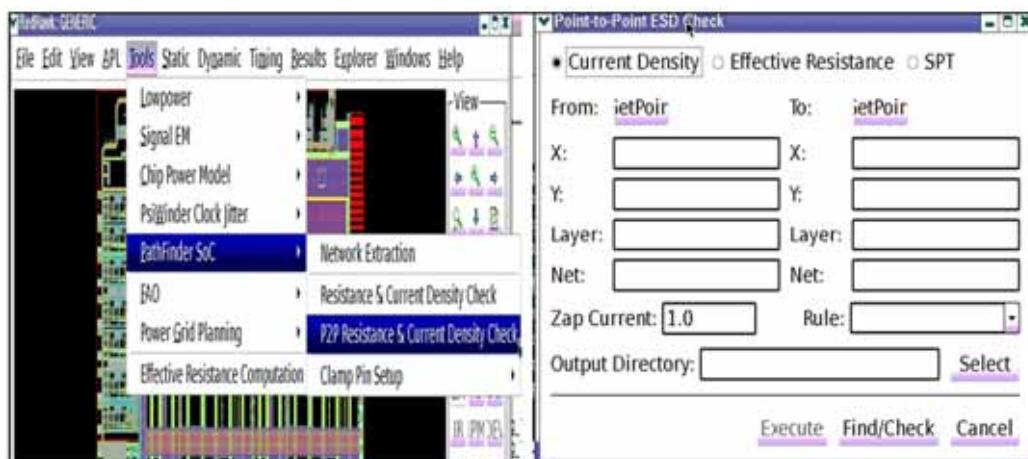


Figure D-18 P2P Resistance & current density check

### Clamp Pin Setup -> Add

Displays a Clamp Pin Setup dialog box to allow selection of options for the 'pfs add clamp\_pin' command. The 'pfs add' command creates shorted parallel clamp pin regions. Note that extraction must be run after adding or deleting shorted regions for the actions to take place. See the TCL command and options description, page D-798.

### Clamp Pin Setup -> Delete

Displays a Clamp Pin Setup dialog box to allow selection of options for the 'pfs delete clamp\_pin' command. The 'pfs delete' command deletes previous esdcheck results from the database, and also deletes previously-created shorted clamp pin regions. See the TCL command and options description, page D-800.

### Clamp Pin Setup -> Highlight

Displays a Clamp Pin Setup dialog box to allow selection of options for the 'pfs show clamp\_pin' command. The 'pfs show' command displays clamp pin shorting information. See the TCL command and options description, page D-801.

### Clamp Pin Setup -> Import

Displays a Clamp Pin Setup dialog box to allow selection of options for the 'pfs import clamp\_pin' command. The 'pfs import' command is used to import the specified shorting clamp description file that has been previously exported using the 'pfs export' command. See the TCL command and options description, page D-801.

### Clamp Pin Setup -> Export

Displays a Clamp Pin Setup dialog box to allow selection of options for the 'pfs export clamp\_pin' command. See the TCL command and options description, page D-800.

### Clamp Connectivity Check -> Unconnected Clamp Devices

Displays an Unconnected Clamp Devices dialog box listing clamp devices with missing connections to the P/G grid, including clamp cell-instance name, x,y location, layer, and net association. The clamp device list can be filtered by a specified cell, pin, layer, or instance. Clicking on the **Zoom** button zooms in to the selected and highlighted clamp device in the display.

### Clamp Connectivity Check -> Clamp Connectivity Filtering

Display a Clamp Connectivity Filtering dialog box to allow selection of various filtering options to list connected clamp devices. Clicking "Apply" button to display a clamp connectivity dialog box listing clamp devices matching the filtering criteria, including clamp cell-instance name, x,y location, layer, and net association. Clicking on the **Zoom** button zooms in to the selected and highlighted clamp device in the display.

### Clamp Connectivity Check -> Net Connectivity

Displays a Net Connectivity dialog box to allow selection of "-netConn/-allNetConn/-rptDisconn" options for the "perform clampcheck" command (see the TCL command and option description, 'perform clampcheck', page D-785).

### Clamp Connectivity Check -> Isolated Bump

Displays an Isolated Bump List dialog box listing bumps with no clamp connections, including bump ID, x,y location, layer, and net association. The bump list can be filtered by specified bump name, layer, or net. Clicking on the **Zoom** button zooms in to the selected and highlighted isolated bump in the display.

### Tools -> Fixing and Optimization (FAO) ->

Displays “Fixing and Optimization (FAO)” dialog that can be used to specify FAO commands in the areas of GSR setting, Missing Vias, Via operations, Mesh Operations, and Decap Operations, as described in the following paragraphs.

Also see [Chapter 7, "Fixing and Optimizing Grid and Power Performance"](#), for detailed descriptions of FAO command functionality using related TCL commands.

#### **Global GSR Setting button -> General Global GSR Setting tab**

Selecting this section of the dialog allows specifying coordinates of an FAO region, and set values for FAO GSR keywords FAO\_DYNAMIC\_MODE, NUM\_HOTINST, CAP\_LIMIT, LEAKAGE\_LIMIT, NOISE\_LIMIT, NOISE\_REDUCTION, FIX\_WINDOW Width and Height, FAO\_NETS, FAO\_LAYERS and FAO\_ROW\_VDD\_SITE. See [section "FAO General Keywords", page C-711](#), for descriptions of the use of these keywords.

#### **Global GSR Setting button -> Advanced Global GSR Setting tab**

Selecting this section of the dialog allows specifying FAO GSR keywords FAO\_VERBOSE, FAO\_PRECISE\_VOLTAGE, FAO\_ADD\_STACK\_VIA, FAO\_TURBO\_MODE, FAO\_DRC, FAO\_DECAP\_OVERLAP, and FAO\_MAX\_SHIFT. See [section "FAO General Keywords", page C-711](#), for descriptions of the use of these keywords.

#### **Missing Via Report button**

Selecting this section of the dialog allows specifying coordinates for an Operation Window for FAO\_REGION, specifying Top Layer and Bottom Layer to be included the missing via report, and additional options for sorting by Hot Instance, Excluding Stack Vias, Include GDS Cells, and setting Voltage Threshold, Pitch, Minimum Width, Via Model, Output Report File, as well as specifying Nets, Cells and Instances by name.

#### **Via Operation button -> Add Vias tab**

Selecting this section of the dialog allows specifying coordinates for an Operation Window for FAO\_REGION, specifying Top Layer and Bottom Layer to be included the via addition, and additional options to Report Shorts or Replace vias, as well as specifying particular Via Coordinates File and Nets, Instances, Via Rules, Via Models by name. Top Widths, Bottom Widths and Electrically Equivalent Nets can also be specified.

#### **Via Operation button -> Delete Vias tab**

Selecting this section of the dialog allows specifying coordinates for an Operation Window for FAO\_REGION to be included the via deletion, and also specifying Cut Layer and Via Models.

#### **Mesh Operation button -> Add Mesh tab**

Selecting this section of the dialog allows specifying coordinates for an Operation Window for FAO\_REGION to be included mesh addition, and also specifying Mesh Geometry parameters such as Layer, Direction, Offset, Space, Pitch, and Width. Other options Extend Layers and Auto Add Vias, Nets and Clip Cells by name are also available.

#### **Mesh Operation -> Delete Mesh tab**

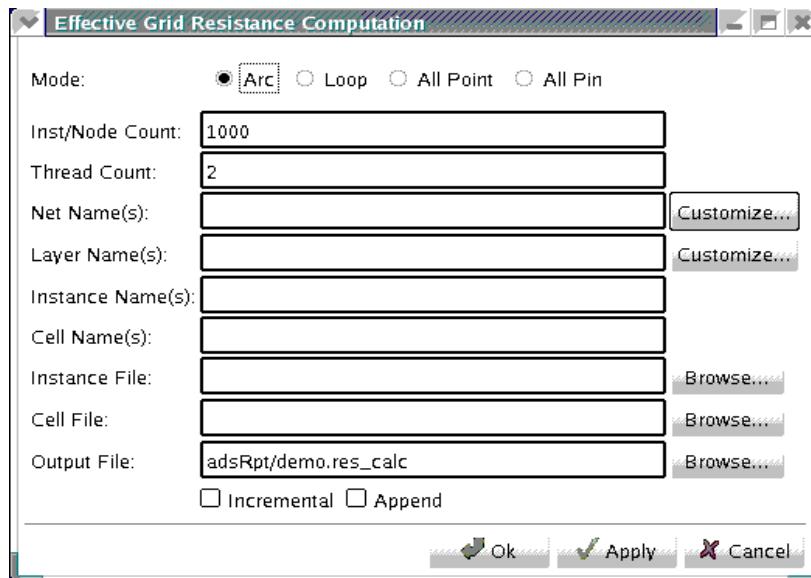
Selecting this section of the dialog allows specifying coordinates for an Operation Window for FAO\_REGION to be included the mesh deletion, and also specifying Mesh Geometry parameters such as Layer, Direction, and Width or Length Constraints. Other options Keep Via, Delete All, and Partial Grid Deletion are available, as well as specifying Cells and Exclude Regions by name.

#### **Decap Operation button -> Decap Fill tab**

Selecting this section of the dialog allows specifying characteristics of the Operation Window, Filter Cells, DvD Criteria for Hot Instance Selection, and Cdecap Cell Placement.

### Tools > Effective Resistance Computation

Displays an ‘Effective Grid Resistance Computation’ dialog, as shown in Figure D-19, to select options for executing the grid resistance calculation command.



**Figure D-19 Effective Grid Resistance Computation**

The computation Modes available are

Arc - one worst-case node per net included (default)

Loop - one node per net, but includes both power and ground nodes

All Points - includes all nodes

All Pins - for MMX designs, also includes all pins internal to transistors (but only one node per pin)

Input data includes

Inst/Node Count - maximum number of worst-case instances or nodes to be included (default 1000)

Thread Count - number of threads used for parallel R computation

Net Names - allows selection of a limited group of nets to be included. The Customize button allows selection of nets from a dialog box list.

Layer Names - allows selection of a limited group of Layers to be included. The Customize button allows selection of layers from a dialog box list.

Instance Names - allows selection of a limited group of instances to be included.

Cell Names - allows selection of a limited group of cells to be included.

Instance File - specifies a file that contains a list of instances to be included in the computation. The Browse button allows file selection from a dialog box list.

Cell File - specifies a file that contains the list of cells to be included in the computation. The Browse button allows file selection from a dialog box list.

### Static Menu

#### Static > Power ->

#### BPA -> Import BPA

Brings up an **Import BPA** dialog to allow you to browse directories and select a file from which to import the BLOCK\_POWER\_MASTER\_CELL and BLOCK\_POWER\_ASSIGNMENT descriptions are copied into and replace the descriptions in the existing GSR file for the session.

#### BPA -> Export BPA

Brings up an **Export BPA** dialog to allow you to copy BLOCK\_POWER\_MASTER\_CELL and BLOCK\_POWER\_ASSIGNMENT descriptions from the session GSR file into a specified text file.

#### BPA -> BPA in GSR

Brings up a **BPA in GSR** dialog to allow you to view and edit elements of BLOCK\_POWER\_ASSIGNMENT keyword statements that are defined for the design using the **Add**, **Edit**, and **Delete** buttons. The selected BPA option is highlighted in the **RedHawk** GUI and a visual display of the region whose values are changing is displayed.

#### BPA -> Instance List

Brings up an **BPA Instances** dialog showing the power of each instance, and allowing you to select and highlight the instances in the design.

#### BPA -> Highlight Exclude Area

When the check box is clicked, areas that are defined in the EXCLUDE option of the BLOCK\_POWER\_ASSIGNMENT keyword are highlighted.

#### BPA -> Power Map of Instances

Displays a color map of the design based on the power value of each instance. Click on the Set Color Range button to display a **Power Map of Instances** dialog, showing the power range represented by each color. You can change the colors and power ranges displayed as desired. Use the check box “Allow setting reference max instance power for color scaling” to change the maximum instance power to clarify the differences between power values of key instances.

#### Calculate

Executes power calculation for the static IR voltage drop analysis flow.

#### Import

Displays an ‘Import’ dialog box to specify a previously-executed power calculation directory to be imported.

#### Static -> Network Extraction

Displays ‘Power Ground Extraction’ dialog box to specify the core R parameters to extract for VDD and GND nets.

#### Static -> Pad Wirebond Package Constraints

Displays a ‘Pad Wirebond Package Constraints’ dialog box to specify the key R,L,C parameters for Pads, Wirebond/bumps, and Package power and ground subcircuit models. Check boxes allow ignoring any of the three elements.

#### Static -> Static Voltage Drop Analysis

Executes static IR voltage drop analysis.

#### Static -> EM Check

Executes EM analysis.

#### Dynamic Menu

## Dynamic -> Power ->

### Calculate

Executes power calculation for the dynamic voltage drop analysis flow.

### Import

Displays an ‘Import’ dialog box to specify a previously-executed power calculation directory to be imported.

## Dynamic -> Network Extraction

Displays a ‘Power Ground Extraction’ dialog box to specify the core R, L, C parameters to extract for VDD and GND nets.

## Dynamic -> Pad Wirebond Package Constraints

Displays a ‘Pad Wirebond Package Constraints’ dialog box to specify the key R,L,C parameters for Pads, Wirebond/bumps, and Package power and ground subcircuit models. Check boxes allow ignoring any of the three elements.

For clock jitter analysis you need to provide accurate pad, wirebond and package constraints in order to evaluate package LC resonance noise and its damping effect. You can also ignore the pad, wirebond/bump or package effects by using one or more of the checkboxes.

## Dynamic -> Dynamic Voltage Drop Analysis

Executes mixed vectorless and VCD voltage drop and current analysis, the same as the TCL command ‘perform analysis -dynamic’.

## Dynamic -> Vectorless Only Analysis

Executes only vectorless voltage drop and current analysis, the same as the TCL command ‘perform analysis -vectorless’.

## Dynamic -> VCD Only Analysis

Executes dynamic voltage drop and current analysis using the specified VCD file, the same as the TCL command ‘perform analysis -vcd’.

## Dynamic -> EM Check

Executes EM analysis.

## Timing Menu

### Timing -> Sign-off Clock Tree ->

#### Analysis

Displays a ‘Clock Tree Setup’ dialog box to specify the timing config file that contains operational controls for running the clock tree analysis program. See [Chapter 8, "Analysis of DvD and Cross-coupling Noise Impacts on Timing"](#) for more details on running the program. After specifying the configuration file path, click the ‘Run’ button to run clock tree analysis.

#### Clock Tree Report

Displays a ‘Clock Tree Report’ window, providing links to show sorted analysis results for each clock tree, displays of path topology from root to selected leafs, plots of DvD waveforms of clock tree instances, highlighted clock tree instances in the layout window, and displaying a summary table for all clock trees, including total leaf count, level count, skew, and delay information.

### Timing -> Sign-off Critical Path ->

### DvD Filter

Brings up a DVD Filter Setup dialog box that allows you to specify a DvD filter rule file, Create a sample DvD filter rule file, Run the DvD filter, and View Results.

### Analysis

Displays a ‘Critical Path Setup’ dialog box to specify the timing configuration file that contains operational controls for running the critical path analysis program, and also specify the DvD Filter Report File. See [Chapter 8, "Analysis of DvD and Cross-coupling Noise Impacts on Timing"](#) for more details on running the program. After specifying the configuration file path, click the ‘Run’ button to run critical path analysis.

### Critical Path Report

Displays a ‘Critical Path Report’ window, providing links to show detailed analysis results of each critical path, highlighting critical paths, plotting DvD waveforms of critical instances, and displaying a slack summary for all critical paths, including a comparison of the slack times calculated by [PrimeTime](#).

### Timing -> Generate MSDF

Displays a ‘Generate MSDF’ dialog box to specify the functionality for generating an MSDF file with derated delay values:

- names for the Input SDF file and for the Output Modified SDF file
- Triplet Match: specifies which SDF values are used for performing voltage derating, Minimum, Maximum, or Typical. If any are selected, the ‘Consider PVT Corner’ button selection is ignored. If none are selected, derating is determined by ‘Consider PVT Corner’ button selection.
- Consider PVT Corner: if set to 1 (default), delay values are scaled if the [RedHawk](#) operating temperature value (in APL) matches any of the temperature definitions (Min, Max, Typical) specified in SDF. If the temperature definitions do not match, no derating is performed. If set to 0, all delay values are scaled regardless of [RedHawk](#) and SDF operating temperature specifications.
- Incremental SDF: output MSDF file contains only delta delay values relative to SDF file delay values.
- DvD Threshold: If selected, provides normal derating for delay values whose associated voltage is equal to or below the DvD threshold voltage ( $V_{nom}$  times the selected threshold ratio between 0.8 and 1.2). DvD voltages above the threshold value are derated as if they were DvD Threshold voltage.

## Results Menu

### Results -> Log Message Viewer

Provides access to four different report viewers described in the following paragraphs: ‘Errors and Warnings Summary’ (default), ‘CPU/Memory Usage’, ‘Setup Design’, ‘Power’ and ‘Results’.

#### Errors/Warnings Summary

Displays a dual window showing a summary of the total messages generated during the analysis by type (Error, Warning, Info) in the upper window and a selected individual message in the lower window.

#### CPU/Memory and Usage

Displays a table showing maximum Memory use, Total CPU Time, and Wall Time for each design step from importing files through dynamic analysis. Note that the CPU time for the simulation step is *not* displayed as cumulative for convenience; the following step and all other steps are shown as cumulative CPU time.

#### Setup Design

Displays a list of files and their paths that have been imported in the design.

### Power

Displays a dual window showing a summary of design power information by frequency and by domain in the upper window, and power related warning messages by instance in the lower window.

### Results

Displays a simulation results report, such as Worst Dynamic Voltage Drop summary report for dynamic analysis.

#### Results -> List of Effective Grid Resistances

After the 'perform res\_calc' command has been executed to compute effective resistances in the design, this command displays the effective resistance per pin in a dialog list. The **Select** button allows ordering the list by layer, net, or both, as specified in the "View Layers" configuration button dialog and the **View-> Nets** menu command dialog.

#### Results -> List of Effective Instance Resistances

After the 'perform res\_calc -loopMode' command has been executed to compute effective resistances for instances in the design, this command displays the resistance for VDD, VSS and the total resistance. The list can be organized by VDD, VSS, or total resistance using the **Sort by** button.

#### Results -> List of Worst EM

For EM analysis, displays an ordered list of the 1000 worst-case power electromigration locations, with the wire x,y starting and ending locations for each occurrence, the layer, net and percent of EM limit. You can also zoom to any of the critical EM locations and highlight it by selecting the location and using the 'Go to Location' button, and also step through the various locations using the 'Prev' and 'Next' buttons. Clicking on the 'Report Setting' button brings up a dialog that allows the list of worst EM locations to be reduced by selecting only certain voltage domains or layers, only certain metal segment sizes, or only those locations that have a selected EM threshold or higher. The 'Export List' button dumps the present 'List of Worst EM' table into a specified text file. In Signal EM mode, a column showing 'Current Direction' is also displayed in the table. The EM results can be sorted by Layer type by clicking on the 'Layer' Header. Also, user can selectively specify which layer violations or violations for particular metal constraints needs to be displayed on the EM results window.

#### Results -> List of Worst IR for Wires and Vias

Displays a 'List of Worst IR for Wires and Vias' dialog box, with an ordered table of worst-case IR drops at Power and Ground nodes, including the x, y Location and Layer of each. By default a 5% IR drop threshold is used to select the worst nodes. The list can be shortened by selecting a larger % drop threshold in the 'Threshold' box, such as 10% or 20%. You can also zoom to any of the critical nodes and highlight it by using the 'Zoom to Location' button.

#### Results -> List of Highest Power Instances

Displays a 'Highest Power Instance Report' dialog box, with an ordered table of worst-case high static power consumption instances, with the x,y location and Instance Name of each occurrence. You can also highlight and zoom to any of the critical instances by using the 'Zoom to Location' button.

#### Results -> List of Metal-Via EM Ratio

This command is foundry-specific. Contact your Apache AE for use information.

#### Results -> List of Worst IR Instances (Static)

Displays an ordered list of worst-case high IR drop instances, with following information on each instance: Vdd-Vss, Vdd drop, Vss bounce, Ideal Vdd, and the x,y Location of the node. You can select the domain to show for multiple-domain designs. You can also highlight and zoom to any of the critical instances by using the 'Zoom to Location' button.

#### Results -> List of Worst Instance DVD

Displays a ‘List of Worst Instance DVD’ dialog box, with an ordered table of worst-case high DvD instances, with following information on each instance: Ideal (nominal) Vdd, Ave DV (average dynamic voltage), Max DV, Min DV, Min DV WC (worst case condition), and the x,y Location and instance Name of each. Using the ‘Sort By’ box at the bottom of the dialog, you can sort the list of critical instances by Average, Maximum or Minimum VDD minus VSS differential. You can also zoom to any of the critical instances and highlight them by selecting an instance in the list and clicking on the Zoom button. The ‘DvD Plot’ button displays an xgraph or sv voltage drop waveform of a selected instance. By default a 5% DvD threshold is used to select the worst instances, based on the criteria selected (‘Average Vdd-Vss’ by default). The list can be shortened by selecting a larger % drop threshold in the ‘Threshold’ box, such as 10% or 20%. You can navigate a large list of instances using the ‘First’, ‘Last’, ‘Prev 1000’, and ‘Next 1000’ buttons at the bottom of the dialog. The **Export List** button allows dumping the displayed instance list to a file. There is also a multi-selection capability to mark the selected instances with the **Zoom** button.

### **Results -> List of Worst Transistor Pin Voltages**

Displays a dialog listing the voltage domains in the left side. After selecting a voltage domain, the Report of Worst Transistor Pin Voltages is displayed, giving the Min Volt’, ‘Max Volt’ and ‘Avg Volt’, pin x,y location, pin name, and transistor name. You can click on the ‘Min Volt’, ‘Max Volt’ and ‘Avg Volt’ column headers to sort the worst transistor pin voltage entries. You can also select and zoom to any of the critical pins by using the ‘Zoom to Location’ button, and view an xgraph or sv plot of the voltage using the ‘Voltage Plot’ button.

### **Results -> Analysis Histograms**

Displays an ‘Analysis Histogram’ dialog box to specify what parameters to display in the histogram: Dynamic Voltage Drop, Static IR, Static EM, or Dynamic EM. Histograms can be constructed ‘By Layer’ selection, or ‘By Instance’ for criteria

- Average/Max/Min DvD over Timing Window,
- Minimum DvD in Simulation Cycle,
- Max Vdd Drop, or
- Max Vss Bounce

For multiple Vdd/Vss domain designs, histograms can be displayed by domain Net Name.

‘Limits’ boxes provide automatic selection of ‘Lower’ and ‘Upper’ mv limits, as well as choice of bin size or number, or values for all of these display parameters can be selected.

To display analysis data for elements only within a specified bounding box area, use the TCL command ‘condition set -xy <x1 y1 x2 y2>’, where x1,y1 and x2,y2 are the lower left and upper right corners of the desired bounding box to be analyzed.

### **Results -> Movie ->**

#### **Make**

Displays a dialog box to select the file for storing the simulation movie, and also chose whether you want a Full Chip Movie or a Partial Chip at Current Zoom.

To create a DvD movie for selected P/G nets, first use the menu command **View -> Nets**, to select or deselect the nets to be included in the ‘Nets’ dialog box that is displayed. After the nets have been selected, use the **Make** command. If movie making is Canceled, all intermediate files are removed in order to save disk space.

To highlight the switching instances, select the checkbox “Highlight Switching Instances” in DvD Movie make dialog box.

#### **Show**

Displays a window to show the color-coded results of dynamic simulation in steps, showing the step number and simulation time for each step. You can use the normal video controls, such as Power (dismisses window), Start, Stop and Pause, or you can change the frames per second speed, or manually step through the simulation.

## Explorer Menu

### Explorer -> Generate

Brings up a ‘Generate’ dialog to select Explorer reports of three types:

- **Data Integrity**
- **Design Weakness**
- **Hot Spots**

and also provides buttons to:

- **Edit Constraints** - brings up a dialog to allow editing parameters for APL, LIB, LEF, STA, SPEF, GDS2DEF/GDS2RH, and VCD checks
- **Run Explorer** - generates reports of the selected type. Note that whatever results have been generated are saved in the design database, but if Explorer is run again with different results selected, any previously-generated reports are overwritten.

More details on using Explorer are provided in the Training documents on the RHE support site, accessed via the link:

[http://www.apache-da.com/%7Eapacheda/system/files/privateapacheda/RHE\\_Detailed\\_Trainingv10.2.pdf](http://www.apache-da.com/%7Eapacheda/system/files/privateapacheda/RHE_Detailed_Trainingv10.2.pdf).

After logging in, go to the “RedHawk Explorer Detailed Training” link on the support site.

### Explorer -> View Results

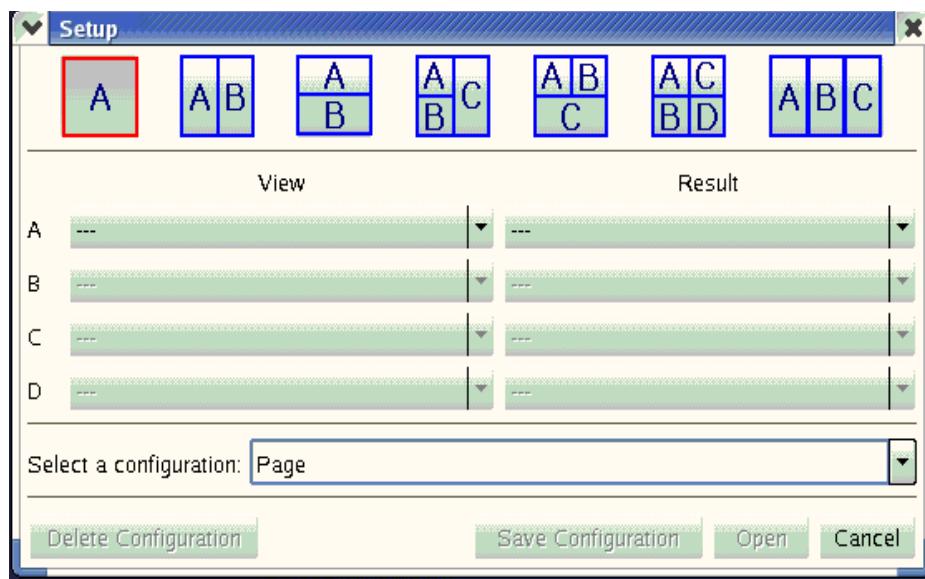
When Explorer is run, results are automatically displayed upon completion. However, if the results window has been closed, or after results are saved to the DB, they can be redisplayed with this command.

## Windows Menu

### Windows -> Multiple Pages

#### Setup

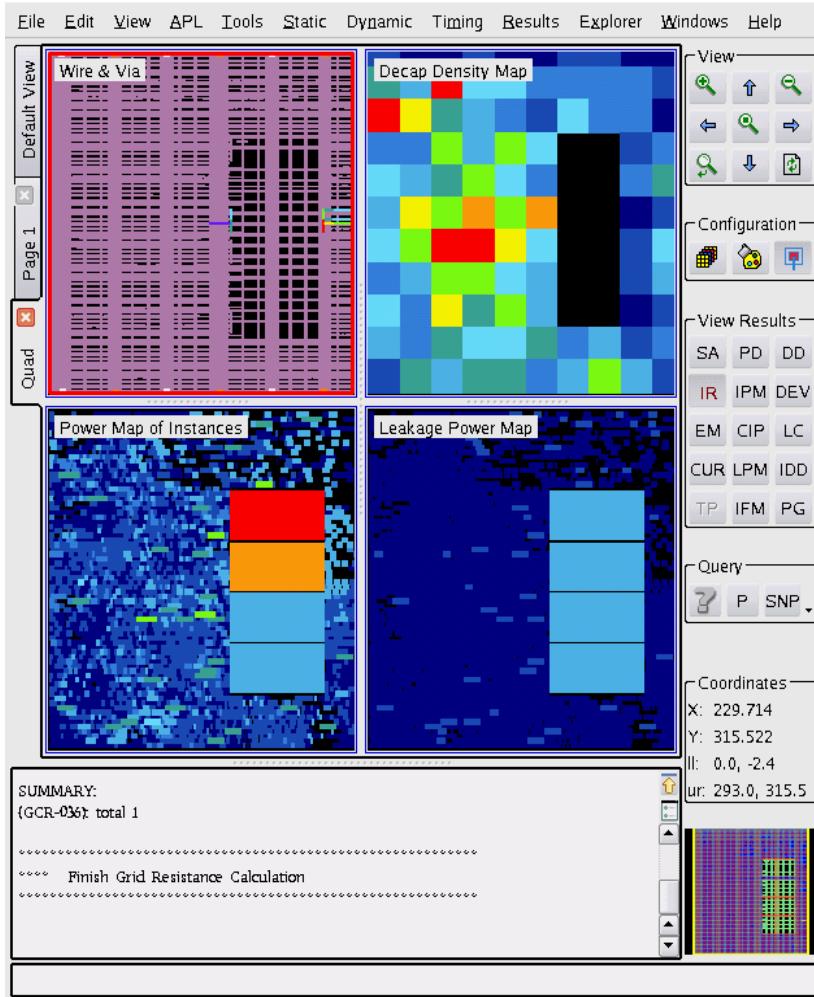
Displays a Setup dialog that allows configuring two, three or four simultaneous design views in the window, as shown in Figure D-20. A different color map can be chosen to be displayed for each of the A, B, C, D panels using the drop-down selection menu. Each multiple page view can be named and opened. A tab with the chosen view configuration is displayed at the left side of the window. The single page Default View is always available from the page view tab.



**Figure D-20 Windows setup dialog**

Several pre-configured views are available from the **Windows-Multiple Pages** drop-down menu: **WireEM**, **WireEM2**, and **WireIR**.

A sample quad view is shown in Figure D-21.



**Figure D-21 Sample multiple page quad view**

#### Windows > Preferences

Displays a Preferences dialog to select behavior of multiple page window. The window parameters to configure are:

##### Basic tab

- GUI Font - allows selection of alternate font for the GUI legend from drop down menu, or default font (Sans Serif 10 point)
- Text Message Font - allows selection of text message font from drop down menu, or return to the default font (Bitstream Charter-9-Normal)
- Map Legend Font - allows selection of map legend font from drop down menu, or return to the default font (Helvetica-8-Normal).
- Mouse Button - allows customization of mouse button functions using the Customization button, which brings up a dialog to choose customization of all three mouse buttons.  
 Also provides a check box to 'Mouse Click Query Leaf Instance First', in which leaf instances are selected on the first mouse click; and the block instantiating that instance is selected on the second mouse click. Similarly for routing layers, the lowest turned on layer is selected first, then the second one on the next click, and so on.

- Color Maps - allows more color steps to be displayed in color maps without going to a scroll bar. The number of color steps can be selected by a Color Entries box displayed at the top of the color map dialog. The default number of steps is 8. The Show Legend box displays color map data in the primary display area. Also can select to show or not show the map legend using the check box (off by default).  
The “Disable GDS Instance in Color Map” box allows disabling the instance IR map of `GDS_CELLS` to view the leaf instances underneath.
- Layout maps - allows showing net-based color maps, such as viewing P/G nets (in SA view) in different colors for each domain. To enable this feature, select the checkbox “Show net based color map”.
- Dialog Properties - saves dialog properties if checked. Also allows Main window overlap for non-modal dialogs, when the box is checked (default).
- Cancel Button - specifies that the Cancel button either Destroys windows or Hides windows when executed.
- Save - allows saving the Preference selections in the Current Directory (default) or the Home Directory.
- Signal Viewer (sv) - allows loading waveforms into the same instance of SV. Supports loading gzipped version of FSDB files. Also supports FSDB hierarchy filter. To take in user-specified labels and units for the waveform's x-axis and y-axis, invoke SV using the following command:

```
sv -xaxis "<x_label>" -yaxis "<y_label>" <file_name>
```

### Advanced tab

- Image Format - choice of GIF (default) or PNG design graphic formats
- Ruler Font - allows selection of ruler font from drop down menu, or return to the default font (Courier-9-Normal)
- Ruler Mode - check box allows showing ticks (intermediate dimension marks - default) or not. The Precision box allows setting the number of digits displayed right of the decimal place to 3. Two digit precision is the default value.
- Voltage Drop Maps - allows settings on IR maps to be synchronized (have the same voltage scaling), or be different, based on the check box:
  - No synchronization of maps
  - Sync all instance based IR map settings, such as IIR and the maps under **View -> Dynamic instance DvD** (default)
  - Sync all wire and via IR map settings
- Highlight Color - allows selection of object highlight color. Default is yellow.
- Highlight Style - allows selection of alternative highlight styles, which are all off by default:
  - Blinking - highlighted objects blink
  - Preselect - objects pointed to by the cursor are highlighted without clicking
  - Cursor - a small highlight rectangle indicates what the cursor is pointing to
  - Highlight invisible layers - allows highlighting instances that are set to 'invisible' in the 'layer select' option in the display.
  - Exclude Duplicate Instance - excludes duplicate instance names when using the "select get" TCL command
- Highlight Sync - reproduces highlights on all views on one page or in all views. By default synchronization of highlights between multiple views is off.
- Zoom Sync - reproduces zoom level on all views on one page or in all views. By default synchronization of zoom level between multiple views is off.

- Movie Show In - selects the view in which to show the selected movie, Pop-up window or Current canvas.
- View Nets Window - allows showing Power/Ground Nets in one tab (default)
- Layer Color Setting - allows choosing High Contrast Color Scheme I or II, which have in-built colors for different cut and routing layers, making identification of different layers easier and assists in debugging issues in the GUI.

### Windows -> Detach View Bar

Detaches the right-hand menu panel from the primary display view.

### Windows -> Detach Message Window

Detaches the log message window from the bottom of the design view. The window can be re-attached using the button on the right side border of the detached message window.

## Help Menu

### Help -> About

Displays information about the **RedHawk** software release being utilized.

### Help -> Manual

Displays the latest “RedHawk User Manual” in PDF format.

## Defining Bindkey Functions

---

### Multiple-key Functions

**RedHawk** can bind some GUI menu commands and functions with defined keystrokes. A template file, *keybind.txt*, in the release *\*/doc* directory, contains a set of pre-defined bindkey functions, a portion of which are shown below:

```
Ctrl-d invokes File->Import DB dialog box
ctrl<key>d File/ImportDB
Ctrl-Left pans to the left
ctrl<key>Left Button/Left
F4 switches to IR (voltage drop) view
<key>F4 Button/IR
...
```

The rules for creating a new bindkey definition file are:

3. Copy the sample keybinding text file, *keybind.txt*, found in the release *doc*/directory, into your working directory with the name you choose, such as *bindkeyABC.txt*. The sample file contains a list of all functions available for definition in the right-hand column, which cannot be edited.
4. Edit and save the key definitions in the left column of the bindkey file as desired.
5. Except for the function keys **F1** to **F12**, all other keystroke combinations must start with a **Ctrl** or **Ctrl-Shift**.
6. Leave the “*<key>*” keyword in the lefthand line when editing. It serves as a delimiter between key definitions.
7. You can import a bindkey definition file at the beginning of a **RedHawk** session using the **TCL** command:

```
import keybinding <bindkey_file_N>
```

8. Using the UNIX environmental variable ‘APACHE\_KEYBIND’, you can automatically import a set of key bindings in a definition file using the following command in your *.cshrc* file:  

```
setenv APACHE_KEYBIND <bindkey_file_N>
```
9. Note that you can define a set of bindkeys only once during a **RedHawk** session, and once defined, they cannot be modified during that session.

## Direct Keybinding

**RedHawk** allows direct binding of often-used TCL procedures to key sequences, and also displaying defined keybindings, using the following commands:

```
config keybind <key_sequence>
```

which displays all defined keybindings, and

```
config keybind <key_sequence> <TCL_procedure_name_or_defin>
```

Example:

```
config keybind Ctrl+A
```

which displays the defined TCL procedure of the “Ctrl+A” keybinding

Example:

```
config keybind Ctrl+A export dbA dbB
```

which defines a “Ctrl+A” keybinding, and replaces the previous one, if it existed.

## Single-key Functions

The GUI also supports user definition of single-key shortcuts. Follow the instructions below to enable single-button shortcut keys. Note that while this function is invoked, the TCL command line is disabled.

1. Copy the sample keybinding text file, *keybinding.txt*, found in the release *doc/* directory, into your working directory with the name you choose, such as *keybindingABC.txt*. The sample file contains a list of all functions available for definition in the right-hand column, which cannot be edited.
2. Edit and save the key definitions in the left column of the keybinding file as desired, as shown in the Figure D-22 example. The “<key>” keyword must precede the name of the key being defined.

Note: This feature is limited to the use of keyboard keys A-Z, a-z, and 0-9. Also, key binding cannot be modified in the same session; that is, keybinding cannot be set twice in the same session.

|        |                   |
|--------|-------------------|
| <key>x | Button/ZoomIn     |
| <key>z | Button/ZoomOut    |
| <key>f | Button/ZoomToTop  |
| <key>k | Button/ZoomBack   |
| <key>r | Button/Refresh    |
| <key>l | Button/ViewLayers |

Figure D-22 Sample keybinding.txt file

3. Import the keybinding file into **RedHawk**:

```
import keybinding keybindingABC.txt
```

4. Enable the specified single keybinding shortcuts by clicking on the **Edit->Single Keybinding** menu command, or use the **Ctrl-T** shortcut. The TCL command line is now disabled and your defined single-key shortcuts are active.
5. After finishing use of the shortcut keys, enable command line input again by clicking on the **Edit->Single Keybinding** menu command, or use the **Ctrl-T** shortcut.

# Appendix E

# Utility Programs

## Introduction

---

This appendix describes key utility programs available in RedHawk:

- [vcctrans](#) : Generates a net toggle file from a Value Change Dump (VCD) file.
- [vcdscan](#) : Generates per-cycle power information from a VCD file for VCD-based dynamic voltage drop analysis.
- [fsdbtrans](#) : Generates a net toggle file from a Fast Signal Database (FSDB) file.
- [rhtech](#): Generates a RedHawk technology file from standard *nxtgrd* or *itf* vendor technology files.
- [ircx2tech](#)
- [gds2rh/gds2def](#) : Generates DEF and LEF files from GDSII data
- [gds2rh -m](#) and [gds2def -m](#) : Generates a detailed view of memory models from GDSII.
- [pt2timing](#): PrimeTime TCL interface to generate the slew (transition times) and timing window for each instance.
- [sim2iprof](#): Generates current profiles from simulation outputs. Extracts “read”, “write”, and “standby” waveforms based on \*.lib formula.
- [aplreader](#): Converts data from a given current profile to standard output.
- [aplcdev2pwc](#): Checks the validity of cdev data, and convert the cdev data to pwcdev data.
- [aplchk](#): Performs cap and pwc validations.
- [clampviewer](#). Linux executable utility to preview ESD clamp I-V data.

## vcctrans

---

The vcctrans utility generates the number of toggles for each net in a VCD file over the whole simulation time, or over frames of specified time. The input to the utility is a VCD file and the outputs are individual block toggle files called *<block>.toggle*. The recommended file extension for the toggle file is *\*.toggle*.

Syntax:

```
vcctrans <input_file_name> -o <output_file_name> -c
 -s <start_time_in_ps> -e <end_time_in_ps>
 -w "<logical_module_name>" "<physical_module_name>"
```

where

- <input\_file\_name> : specifies the VCD file.
- o <output\_file\_name> : specifies the toggle file.
- c : enables case-insensitive checking.
- s <start\_time\_in\_ps> : specifies the start time, an integer with a default of t=0.
- e <end\_time\_in\_ps> : specifies the end time, an integer with default of full simulation time
- w "<logical\_module\_name>" : defines the logical module name from VCD
- "<physical\_module\_name>" : defines the physical module name from DEF.

vcדtrans can be run on a global VCD file to generate the global toggle file.

Note: If the specified time value of vcדtrans or vcדscan options -s and -e are smaller than 0.1, RedHawk assumes you want to use seconds as the time unit. Otherwise, ps is used as the time unit.

## vcדscan

---

You can run VCD-based dynamic voltage drop analysis if you have a known-valid VCD file for your design. First, make sure that you have instance power calculation files generated by RedHawk, by default located in the *adsPower* directory. The vcדscan utility computes the peak power and per-frame power scenario. The default simulation frame size is 1/freq (1 cycle).

Syntax:

```
vcדscan <VCD_file_name> -f <cycle_time_in_ps>
 -w "<logical_module_name>" "<physical_module_name>"
 -msg <msg_file_name> -o <output_file_name>
 -d <input_file_dir_path> -cmd <command_file>
 -s <start_time> -e <end_time>
 -a <sim_frame_start_time> <sim_frame_end_time> -tt
```

where

- <VCD\_file\_name> : specifies the VCD file from Verilog simulation.
- f <cycle\_time\_in\_ps> : specifies the frame size or cycle time in picoseconds.
- s <start\_time> -e <end\_time> : defines the start and end times for a power calculation scan sequence, in ps
- a <sim\_frame\_start\_time> <sim\_frame\_end\_time> : defines the start and end time for the complete simulation frame, in ps
- w "<logical\_module\_name>" : specifies the logical module name from VCD
- "<physical\_module\_name>" : specifies the physical module name from DEF.
- msg <message\_file\_name> : defines where the messages are saved. The default is print to screen.
- o <output\_file\_name> : defines where the output file containing frame-by-frame power data are saved. The default is <design>.pwr.
- d <input\_file\_dir\_path> : defines the relative or absolute path to the input files. The default is the *adsPower* dir.
- cmd <command file> : defines a command file containing the vcדscan command options to be executed.
- tt : selects use of true timing results from vcd file for simulation

Example:

```
vcדscan VCD/design.VCD -f 3320 -msg vcדscan.msg
 -w "test/top" "" -o design.pwr
```

The output file *design.pwr* from the vcדscan utility contains the average power of each cycle, as well as the top five peak power cycles and their corresponding time steps.

---

**NOTE:** Remember the peak power times (<FROM>, <TO>) and the worst-case periods, as this information is used later in RedHawk during simulation.

---

To run VCD vector-based dynamic voltage drop analysis, invoke RedHawk by using the **Dynamic > VCD-based Dynamic IR-drop Analysis** command. The command prompts for simulation parameters, such as the peak power periods (<FROM>, <TO>) and the total simulation cycle time. Enter the simulation parameters and click **Start Simulation** to run VCD-based dynamic voltage drop analysis. If you already have peak power

information from either intimate design knowledge or from a third-party tool, you can skip the vcd utilities and proceed directly to running **Dynamic > VCD-based Dynamic IR-drop Analysis** in RedHawk.

## fsdbtrans

---

The fsdbtrans utility generates the number of toggles for each net in an FSDB file over the whole simulation time, or over frames of specified time. The input to the utility is a FSDB file and the outputs are individual block toggle files with names *<block>.toggle*. The recommended file extension for the toggle file is *.toggle*.

Syntax:

```
fsdbtrans <input_file_name> -o <output_file_name> -c
 -s <start_time_in_ps> -e <end_time_in_ps>
 -w "<logical_module_name>" "<physical_module_name>"
```

where

- <input\_file\_name> : specifies the FSDB file.
- o <output\_file\_name> : specifies the toggle file.
- c : enables case-insensitive checking.
- s <start\_time> : defines the start of scan time sequence in ps
- e <end\_time> : defines the end of scan time sequence in ps
- w "<logical\_module\_name>" : specifies the logical module name from FSDB
- "<physical\_module\_name>" : specifies the physical module name from DEF.

The fsdbtrans utility can be run on a global FSDB file to generate the global toggle file.

## ircx2tech

---

### ircx2tech <options>

The ‘ircx2tech’ command converts an input iRCX file to an appropriate Apache tech file with the syntax:

```
ircx2tech -i <input_file> -o <output_file> -v [min|typ|max]
 ?-m <layer_mapping_file>? ?-g captab ?
```

where

- i: specifies the input iRCX file
- o: specifies the name for the output tech file
- v: specifies a via resistance table with a different RC corner iRCX file
  - a. “-v min” for R\_best/C\_best corner
  - b. “-v max” for R\_worst/C\_worst corner
  - c. “-v typ” for typical corner
- m: specifies the name of the layer mapping file that contains the layer mapping table from the iRCX file to the tech file
- g captab: if specified, RedHawk calls \$APACHEROOT/bin/captab to generate the corresponding capacitance lookup table file

## rhtech

---

The **RedHawk** Technology file specifies all required technology information in text format. The rhtech utility takes technology information provided in standard vendor \*.nxtgrd or \*.itf formats and translates the data into **RedHawk** readable tech file format. The syntax for invoking the rhtech utility from a UNIX command line is given below:

```
rhtech -i <input_file> [-o <output_file>]
[-m <layer_mapping_file>] [-e <EM_file>] [-pe <EM_file>]
[-t <temp_file>]
```

where

- i <input\_file> : specifies required vendor technology input file, in *nxtgrd* or *itf* format
- o <output\_file> : specifies optional output file; default is '<input\_file>-tech'. Note that the output file contains several comment lines defining what input tech file generated the output, as shown in the example:

```
Apache Redhawk Technology File
generated by 'rhtech' program from file- 'in/tddmy09m7t.itf'
-m <layer_mapping_file> : specifies optional layer name mapping file. No default.
-e <EM_file> : specifies basic EM current density limit file.
-pe <EM_file>: specifies special polynomial-based EM rule file. This will
incorporate EM-Rule file section of GLOBAL_LIFE_TIME. The syntax of the
polynomial EM rule file for the '-pe' option is:
```

```
CONDUCTOR metal1 {
 POLYNOMIAL_BASED_EM_DC {
 LENGTH_RANGES { 3.402823e+38 }
 EM_POLYNOMIAL { 4.14 * w }
 }
 POLYNOMIAL_BASED_EM_RMS {
 LENGTH_RANGES { 3.402823e+38 }
 EM_POLYNOMIAL { 3.64 * w * (sqrt (1 + (2.46 / (w + .08)))) }
 }
 POLYNOMIAL_BASED_EM_PEAK {
 LENGTH_RANGES { 3.402823e+38 }
 EM_POLYNOMIAL { 34.4 * w }
 }
 ...
}
VIA via1 {
 ...
}
```

-t <temp\_file> : specifies optional temperature conditions file. No default.

The following provides more detail on input data to the utility:

**-m** : If needed, maps \*.nxtgrd file layer names to the Tech file design layer names (not the same as layers specified in GDS2DEF or GDS2RH). Also specifies removal layers, such that if the ITF or \*.nxtgrd file contains layers below metal1 such as poly, active and contact, these layers can be removed from the output file by adding them to the 'remove\_layers' section of the map file. The format of the map file is as follows:

#### Map File Syntax

```
conducting_layers
 <RH_layer_name> <input_techfile_layer_name>
via_layers
```

```
<RH_layer_name> <input_techfile_layer_name>
remove_layers
<input_techfile_layer_name>
```

Example:

```
conducting_layers
 METAL1 metal1
 METAL2 metal2
via_layers
 VIA1 vial
remove_layers
 TEXT
 Poly
 Active
```

-e : Specifies the file providing the maximum acceptable EM current density. No EM limits are specified in the vendor technology files, so it must be specified here if required. The default value in tech file is 1.0 mA per micron wire width. The syntax of the specified EM file is:

```
<metal_layer_name> <EM_value> [<EM_adjust_value>]
<via_layer_name> <via_max_current>
```

where

<metal\_layer\_name>: tech file metal layer name

<EM\_value>: 'EM' parameter value for specified metal layer in tech file (see section "METAL", page C-547)

<EM\_adjust\_value>: 'EM\_adjust' option value for specified metal layer in tech file. If specified, the wire width used to calculate current density is biased as follows: calculated current density = true current in wire / (width of wire - EM\_adjust).

<via\_layer\_name> : name of via layer

<via\_max\_current>: maximum allowable current for a via (mA) whose area is specified by the via Area keyword in the ITF file for that layer. The value is scaled for vias with a different area.

For example, if you have the following design rule, with an EM adjust of .01:

| Metal  | I <sub>max</sub> (mA) |
|--------|-----------------------|
| -----  |                       |
| Metal1 | 1.0 x (w-0.01)        |
| Metal2 | 2.0 x (w-0.01)        |
| ...    |                       |
| -----  |                       |
| Via    | I <sub>max</sub> (mA) |
| -----  |                       |
| Via1   | 0.189 per via         |
| Via2   | 0.210 per via         |

then the *rtech* EM input file should look like:

```
Metal1 1.0 0.01
Metal2 2.0 0.01
...
Vial 0.189
Via2 0.210
```

**-t** : Specifies the file providing the temperature at which the *nxtgrd* parameters are created, the final target RH temperature, and the first and second order temperature coefficients. The syntax of the specified temperature file is:

```
<layer_name> <nominal_T> <RH_actual_T> <Tc1> <Tc2>
```

where

<nominal\_T> <RH\_actual\_T> : nominal and RedHawk actual temperature conditions (°C)

<Tc1> <Tc2> : first and second order temperature coefficients.

For example, if the input resistance parameters are at 25°C, and the RH analysis is to be performed at 125°C, then in the temperature file you need to specify temperature data as follows for each layer:

```
<layer_name> 25 125 <Tc1> <Tc2>
```

...

Also note that for accuracy RedHawk can handle width-spacing-based sheet resistance data; rhtech can extract width-spacing based tables from the *nxtgrd* and *itf* files directly to the Apache technology file (as specified with the 'RPSQ\_VS\_WIDTH\_AND\_SPACING' keyword).

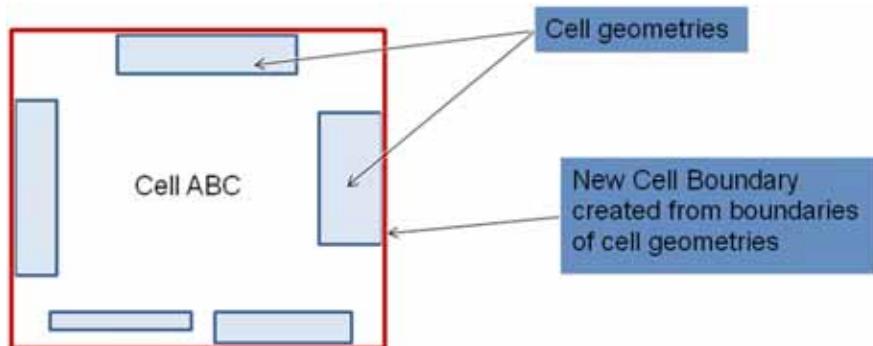
See [section "Apache Technology File \(\\*.tech\)", page C-544](#), for details on the syntax and definitions of keywords in the technology file.

## gds2rh/gds2def

Certain data, such as memories, redistribution layers, flip-chip bump layer descriptions, and analog blocks, may only be available in GDSII format. This type of data must be converted to DEF/LEF format for integration into the RedHawk database. The gds2rh/gds2def utilities generate a .def file from one GDSII file or a set of GDSII files. They are used to convert memory grids, flip-chip layers, extract a portion of the full-chip GDSII (used in RedHawk's IO-SSO flow), and other applications where the block consumes negligible power.

Depending on the specifications, the program outputs a DEF file, or both DEF and LEF files, using the name of the cell, such as *<cellname>.def* and *<cellname>\_adsgds.lef*. The created files contain the metal geometries of all the extracted nets. When these DEF/LEF files are used in RedHawk for analysis, the power distribution of these blocks are uniform and the current sinks are along the pins of the block.

If the GDS prboundary layer is missing for the cell, gds2rh/gds2def picks the cell bounding bbox from the cell geometries, and outputs it in LEF/DEF, as shown in Figure E-1



**Figure E-1 Automatic cell bounding box determination**

**NOTE:** The gds2rh/gds2def utility can output smooth 45-degree geometries in DEF for power/ground routing, including I/O rings and RDL. Designs that specify 45° power routes using DEF 5.6 format are read in and handled smoothly in the native 45° form. Other non-Manhattan shapes are not modeled.

This feature is turned off by default. It can be turned on by specifying the following in the gds2rh/gds2def configuration file: LEFDEF56 1.

GDS2RH/GDS2DEF also supports the MAG magnification keyword in the GDSII file that allows changing the size of dimensions of an instance relative to the master cell.

To convert GDSII to a `.def` file, first create the gds2rh/gds2def configuration file and GDSII layer map file. Then convert the GDSII file to `*.def` using gds2rh/gds2def.

### Comparison of GDS2DEF and GDS2RH Use

GDS2RH has an improved flow relative to the original GDS2DEF flow, in that it has:

- better run time and memory usage
- APM2 support (geometry-based optimization to reduce node count during the RedHawk run)
- the same usage model as gds2def/gds2def-m
- Note that neither the Spice-based flow nor the contact-based pin creation flow are supported in GDS2RH. You should use the **Totem** transistor-level flow.

The two flows are compared in Figure E-2 and Figure E-3.

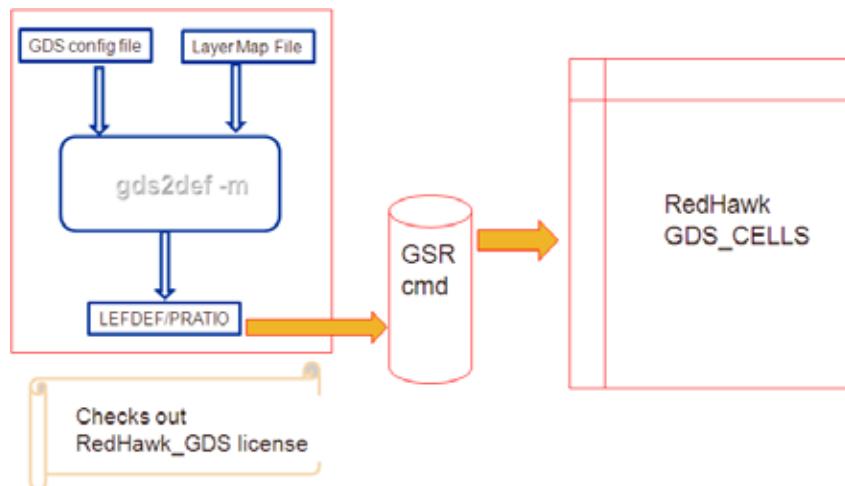
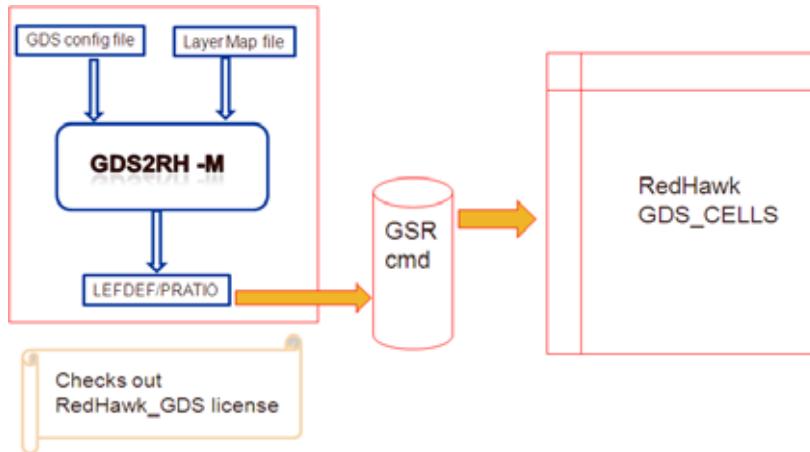


Figure E-2     GDS2DEF flow



**Figure E-3 GDS2RH flow**

To use the GDS2RH command, type :

```

gds2rh [-m] <gds_config_file>
 -dmp <lsv_config_file> <gds2rh_config_file>
 ? [-out_dir | -o | -output_directory] <dir_name>?
 ? -log' <log_directory> ?

```

A RedHawk\_GS license is checked out. The default output directory is the current run directory. The \*.config file creates a \*.def file in the output directory. You should not use the same output directory for gds2rh -m commands.

The log file *adsRpt/GDS/.lefcell* is removed after a normal GDS2RH + SPEF flow run, in order to save disk space.

If the VALIDATE\_MODEL configuration file keyword is used, then a RedHawk\_static license is checked out. Log, Error, and Warning files are written to:

- *adsRpt/gds.<log,err,warn>*
- *adsRpt/redhawk.<log,err,warn>*

### Distributed Processing of GDS2RH Runs

For multiple top cell designs, every top cell is processed in parallel when the -dmp option is specified. LSF configuration file keywords NUMBER\_OF\_JOBS, GRID\_TYPE and CANDIDATE\_HOST\_LIST must be used, as described below.

- CANDIDATE\_HOST\_LIST {
  - <machine1>
  - <machine2>
  - ...
}
 Specifies the machines that gds2rh DMP is to run on.
- GRID\_TYPE SSH
 Specifies the SSH distribution system.
- NUMBER\_OF\_JOBS <num\_jobs>
 Specifies the maximum number of jobs that can be run simultaneously.

## Creating the GDS2RH/GDS2DEF Configuration File

Create the GDSII configuration file `<gds2rh_filename>.config` or `<gds2def_filename>.config`. There are four required sections in the GDSII configuration file:

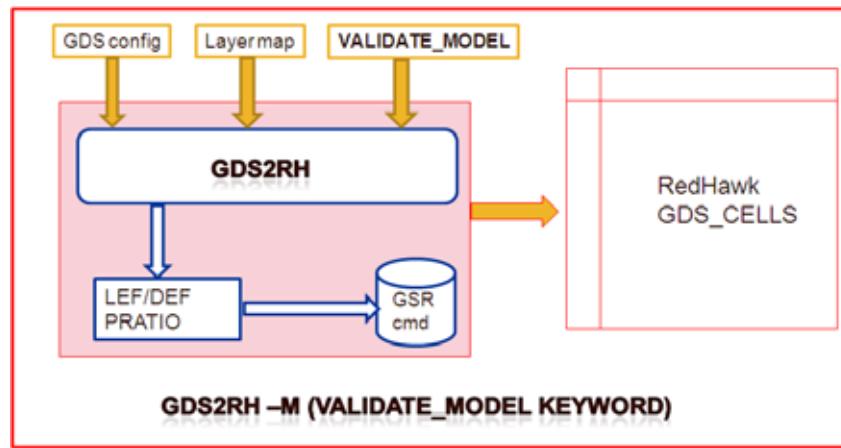
- GDSII file: specifies a set of GDSII files for converting.
- Top cells definition: the name of the top cell structure(s) in GDSII.
- Nets definition: identifies the power, ground and signal nets that are labeled in text in the GDSII.
- Layer map definition: specifies the map file that contains information for mapping the GDSII layer numbers to the corresponding layers defined in LEF/DEF.

Note that for memories, information about the configuration file is in [section "gds2rh -m and gds2def -m", page E-915](#).

- For GDS2RH only, to validate the models the `VALIDATE_MODEL` keyword should be set in the configuration file, with syntax as follows:

```
VALIDATE_MODEL {
 TECH_FILE <tech_file_path>
 LIB_FILES {
 <lib_files>
 ...
 }
 PAD_FILES {
 <pad_files>
 ...
 }
 ADD_PLOC_FROM_TOP_DEF 1
}
```

The validation flow is shown in Figure E-4.



**Figure E-4 Cell validation for GDS2RH flow**

There are a number of optional sections in the GDS2RH/GDS2DEF configuration file:

- Input LEF files
- Geometry extraction

- Selective cell hierarchy handling
- Auto pad location generation
- DSPF/SPEF-based standard cell flow
- Switch cell handling
- Other keywords

The keywords for these parts of the configuration file are described in the following sections.

### GDS2RH Sub-configuration Files

RedHawk supports GDS2RH sub-configuration files for flows with multiple top cells, in which the VDD\_NETS or GND\_NETS start points can be explicitly specified in the sub-configuration files separately. The same nets should be specified in the main configuration file also, as in the following example:

```

main.config:

TOP_CELLS {
A a.conf
B b.conf

}

VDD_NETS {
VDD
}

GND_NETS {
VSS
}

a.conf :

VDD_NETS{
VDD {
- @ 10 10 10 10 10
}
}

GND_NETS{
VSS {
- @ 20 20 20 20 20
}
}
```

### GDSII Files Keywords

#### **GDS\_FILE**

If more than one GDSII file must be specified for conversion, all the file names should be specified in GDSII file section. Required. The syntax is as follows:

```
GDS_FILE {
<GDSII file name>
```

```
 ...
}
```

### **GDS\_MAP\_FILE**

Specifies the GDSII mapping file, which is passed down to the automatically generated GDS2DB template configuration file, and maps layer numbers to layer names.

Syntax:

```
GDS_MAP_FILE <GDSII_layer_map_file>
```

## **Top Cell Definition Keywords**

### **CELL\_NAME\_MAP**

Can be used where the top cell name in GDS is different from that in the netlist. Then while executing the GDS2RH flow, you can rename the top cell name in the generated LEF/DEF views to match the top cell name in the netlist. Any desired name can be given, and the name is retained in the LEF/DEF.

Syntax:

```
CELL_NAME_MAP {
 <gds_name> <desired name>
}
```

### **FLATTEN\_TOP**

When set, the hierarchy of the top cell GDS of the design is flattened (default 0).

Syntax:

```
FLATTEN_TOP [0 | 1]
```

### **FLATTEN\_TOP\_PROPERTIES**

When the OPTIMIZED\_TRACING option is set, tracing is performed in a faster optimized manner. When set to 1, no feed-through geometries are traced through box cell hierarchy. When set to 2, feed-through geometries are traced through box cell hierarchy to the top level. The default is 0, in which no optimized tracing is performed, and which requires longer runtimes.

Syntax:

```
FLATTEN_TOP_PROPERTIES {
 OPTIMIZED_TRACING [0 | 1 | 2]
}
```

### **GENERATE\_TOP\_LEF**

When set, creates top cells LEF in ‘reduced’ and ‘extended’ modes, as defined in the GSR GDSII\_FILE <file>. This is useful for stitching a cell properly at the next (parent) level in the hierarchical flow (box\_cell flow). Note the following behavior:

- no parent LEFs are created when there is a parent LEF already specified in the GDS configuration file.
- if a parent LEF already exists in the output directory, the file is not overwritten, but instead a new file \*\_1.lef is generated.

Default: 0.

Syntax:

```
GENERATE_TOP_LEF [0 | 1]
```

**TOP\_CELL****TOP\_CELLS**

TOP\_CELL and TOP\_CELLS keywords specify the name of the top cell or the list of top cells to be extracted as defined in the GDSII file. RedHawk supports multiple top cells in gds2rh/gds2def to reduce the time needed to read in multiple top cells in large GDSII files.

```
TOP_CELL {
 <TopCell_name> ?<sub-configuration filename>?
}
OR: TOP_CELLS {
 <TopCell_name> ?<sub-configuration filename>?
 ...
 <TopCell_nameN>
}
```

where

<TopCell\_name>: the top cell name to be extracted.

<TopCell\_nameN>: additional cell names to be extracted

<sub-configuration filename> : optional definition file. Several cells can share one sub-configuration file. Keywords specified in the sub-configuration file override the value set in the top cell configuration file. If no sub-configuration file is specified for a cell, the value set in the top configuration file is used. Only a few keywords are allowed in the sub-configuration file:

MEMORY\_CORE\_REGIONS, SPICE\_NETLIST, MEMORY\_CELL,  
WORD\_LINE\_DIMENSION, and OUTPUT\_DIRECTORY.

There is no keyword to switch between 'gds2rh/gds2def' and 'gds2def -m/gds2rh -m'. If -m is specified on the command line, all cells specified in TOP\_CELLS are treated as part of the memory characterization process.

**TOP\_CELL\_MAP**

This keyword is used along with TOP\_CELL (which finds the block/cell name in the GDSII data). The TOP\_CELL\_MAP keyword specifies the names of cells inside the generated LEF/DEF files, particularly when the macro cell name in LEF (input parent) and cellname in GDSII data do not match.

```
TOP_CELL_MAP <def_top_cell_name>
or TOP_CELL_MAP {
 <def_top_cell_name1>
 ...
 <def_top_cell_nameN>
}
```

**TOP\_CELLS\_MAP**

In there is a master cell name mismatch between LEF and GDS, use the TOP\_CELLS\_MAP keyword to rename the master cell in the output LEF/DEF. GDS2RH/GDS2DEF then extracts geometries from TOP\_CELLS and renames them to the mapped name as specified by TOP\_CELLS\_MAP.

```
TOP_CELLS_MAP {
 <cell_name1> <cell_name1_map>
 <cell_name2> <cell_name2_map>
```

```
...
```

A sample GDS configuration file would look as follows:

```
TOP_CELLS {
 VDD_ST_TF3V3_LIN_50u
}
LEF_FILE cell.lef
VDD_NETS {
 vdde3v3
}
GND_NETS {
 gnde
}
GDS_FILE design.gds
GDS_MAP_FILE gds.map
TOP_CELLS_MAP {
 VDD_3V3_LIN_50u VDD_3V3_LIN
}
```

### **DEF\_FILE\_DEFINITIONS**

Defines default resolution units (default 2000) and die size and location. DIE\_AREA specifies the opposite die corners, in microns. OFFSET specifies the coordinate offsets, in microns.

#### **Syntax:**

```
DEF_FILE_DEFINITIONS {
 UNITS <DEF resolution unit>
 DIEAREA <llx> <lly> <urx> <ury>
 OFFSET <x_off> <y_off>
}
```

### **Nets Definition Keywords**

#### **CHECK\_TRACING**

Enables checking of any two nets that are shorted. The default is 0 (do not check for shorts).

```
CHECK_TRACING [0 | 1]
```

#### **DUMMY\_NET\_OUTPUT\_PINS\_FILE**

To support the required dummy net generation in the GDS + SPEF flow, GDS2DEF extracts the data of the NETS section in DEF from the SPEF file. Specify a dummy net file containing standard/memory cell output pin names to enable dummy net generation in NETS if an empty SPEF file is provided. The input dummy net file format is :

```
Q
Out
Dout[0]
Dout[1]
```

#### **Syntax:**

```
DUMMY_NET_OUTPUT_PINS_FILE <filename>
```

Sample output of NETS in DEF :

```
NETS 1 ;
- dummy_net (*Q)(*Qout) ...
```

### **EXTRACT\_ALL\_LEF\_PIN\_SIGNALS**

When set, GDS2RH/GDS2DEF preserves the signal names in the output DEF file, instead of renaming the signals “adsN1\*”, “adsN2\*” ... , which is the default behavior.

```
EXTRACT_ALL_LEF_PIN_SIGNALS [0 | 1]
```

### **GND\_NETS / VDD\_NETS**

To define the ground net mapping section, use the GND\_NETS keyword, which has the following syntax, the same as for VDD nets specification following. The configuration file also supports keywords VDD\_NETS and GND\_NETS in cell-specific sub-configuration files.

```
GND_NETS
{
<ground_net_name_in_output_DEF> {
 <ground_net_name> @ <gds_layer_number>?:<data_type>?
 <gds_x_position> <gds_y_position>
 ...
 <ground_pin_name_in_DEF>
 <ground_net_name_in_GDS>
 ...
}
*<cell_name> @ <layer_name>
<NetName> <cell_name> @ <layer_name>
}
```

The syntax of the power net mapping section, using the VDD\_NETS keyword, is shown below:

```
VDD_NETS
{
<power_net_name_in_output_DEF> {
 <power_net_name> @ <gds_layer_number>?:<data_type>?
 <gds_x_position> <gds_y_position>
 ...
 <power_pin_name_in_DEF>
 <power_net_name_in_GDS>
 ...
}
*<cell_name> @ <layer_name>
<NetName> <cell_name> @ <layer_name>
}
```

where

<power/ground\_net\_name\_in\_output\_DEF> : specifies net names in DEF for groups of power nets. Net naming can use wild cards, such as “VDD\*” for “VDD3” and “VDD:”. There is also support for bit select definitions, such as “VDD(0)” or “VDD[0]”, and part select, such as “VDD[1:3]”, in Net Names.

@ <gds\_layer\_num> <gds\_x\_position> <gds\_y\_position> : specifies the GDS layer number, and optionally data type, and x,y position. The coordinate format is just to provide a tracing starting point; the name label is not used for text label searching.

<power/ground\_pin\_name\_in\_DEF> : specifies power pin name in DEF

<power/ground\_net\_name\_in\_GDS> : specifies power net name in GDS or Spice

\*<cell\_name> @ <layer\_name> : specifies all isolated geometries/domains traced using geometries from the specified GDS layer inside cell\_name, create nets with internal names such as cell\_name\_Net1, cell\_name\_Net2

<NetName> <cell\_name> @ <layer\_name>: specifies all isolated geometries/domains traced using geometries from the specified GDS layer inside cell\_name, create nets with names prefixed with NetName such as NetName1, NetName2

The GDS2RH/GDS2DEF conversion utility allows multiple power net tracing. It searches the GDSII file for nets that are labeled with the same names as in the power net section, trace the geometries that belong to the net and generate the .def description with those geometries.

The power net and ground net can be specified by simply using a net name also. For example, if a power net only has one entry and the power net name to be used in output DEF is the same as the net name in the entry, then a simplified format can be used, as shown below:

```
VDD_NETS {
 <power net1>
 <power net2>
 ...
}
```

An example power net section is shown below:

```
VDD_NETS {
 VDD {
 VDD
 VDD1_core
 VDD2_core
 VDD3_core
 }
}
```

The power net in the output .def has the given name in the power net section. In this example, VDD, VDD1\_core, VDD2\_core, and VDD3\_core is included under VDD in the generated .def file. Note that the power net names are case sensitive.

### IGNORE\_LEF\_PG\_PIN\_GEOMETRY

When set, GDS2RH/GDS2DEF does not merge LEF P/G pins and GDSII in the generated DEF and preserves them independently. Based on the GDS2RH/GDS2DEF command, results are as follows:

- for gds2rh/gds2def (no -m option)

The LEF P/G pins go into generated LEF, irrespective of any keyword.

- for gds2rh -m /gds2def -m

The LEF P/G pins do not go into generated LEF by default. If 'IGNORE\_LEF\_PG\_PIN\_GEOMETRY 1' is set, the LEF P/G pins go into LEF.

For the DEF PINS section, when IGNORE\_LEF\_PG\_PIN\_GEOMETRY is set to 1, no input LEF P/G pin geometries go into the DEF PINS section; when the keyword is set to 0, they do.

### MULTI\_TASKS

When set to 1, each net tracing process uses a separate CPU, to speed up GDS2DEF run time. Default 0.

```
MULTI_TASKS [0 | 1]
```

### NET\_NAME\_CASE\_SENSITIVE

By default, text label name searching is case sensitive. Specifying NET\_NAME\_CASE\_SENSITIVE to 'no'/'0'/'false' turns the search of text-layer case insensitive.

### SIGNAL\_NETS

The syntax of the signal net mapping section is specified with the SIGNAL\_NETS keyword, as follows:

```
SIGNAL_NETS {
 <DEF_signal_net_name-1>
{
 <signal net-a>
 <signal net-b>
 ...
}
...
{
<DEF_signal_net_name-N>
{
 <signal net-Na>
 ...
*<cell_name> @ <layer_name>
<NetName> <cell_name> @ <layer_name>
 ...
}
```

where

- <DEF\_signal\_net\_name-N> : net name in DEF for group of signal nets
- <signal net-a> : signal net name in GDS or Spice
- \*<cell\_name> @ <layer\_name> : specifies all isolated geometries/domains traced using geometries from the specified GDS layer inside cell\_name, create nets with internal names such as cell\_name\_Net1, cell\_name\_Net2.
- <NetName> <cell\_name> @ <layer\_name>: specifies all isolated geometries/domains traced using geometries from the specified GDS layer inside cell\_name, create nets with names prefixed with NetName such as NetName1, NetName2.

To trace and extract all signal nets from their corresponding I/O pins in the selected region, use the keyword:

```
EXTRACT_ALL_LEF_PIN_SIGNALS 1
```

### **USE\_TEXT\_FROM\_HIERARCHY\_LEVEL**

Describes hierarchy levels from which the text labels should be taken for net tracing. The syntax is:

```
USE_TEXT_FROM_HIERARCHY_LEVEL <hierarchy_level_descripton>
```

Examples:

```
USE_TEXT_FROM_HIERARCHY_LEVEL 0 2 5 ... # 0 is top-level
USE_TEXT_FROM_HIERARCHY_LEVEL 2-5
USE_TEXT_FROM_HIERARCHY_LEVEL 2-MAX
USE_TEXT_FROM_HIERARCHY_LEVEL 2 - MAX
USE_TEXT_FROM_HIERARCHY_LEVEL 0-5
```

### **USE\_TEXT\_FROM\_CELLS**

Describes cells from which the text labels are taken for net tracing. The syntax is:

```
USE_TEXT_FROM_CELLS
{
<cell_1>
<cell_2>
...
}
```

Example:

```
USE_TEXT_FROM_CELLS
{
<cellABC>
<cellXYZ>
Top_ab
}
```

### **USE\_TEXT\_FROM\_HIERARCHY\_BLOCK**

Describes the hierarchies from which the text labels should be taken for net tracing. The syntax is:

```
USE_TEXT_FROM_HIERARCHY_BLOCK {
<hierarchy_path1>
...
}
```

Example :

```
USE_TEXT_FROM_HIERARCHY_BLOCK {
TOP/A
TOP/A/B
TOP
}
```

In this example cell B is inside cell A and A is inside the TOP cell.

### **USE\_TOP\_LEVEL\_TEXT\_ONLY**

When set to 1, only the text in the top cell is extracted and used for power/ground net tracing purposes. The syntax is:

```
USE_TOP_LEVEL_TEXT_ONLY [0|1]
```

Names must be defined in VDD\_NETS and GND\_NETS keywords. The default is 0.

## Layer Map Definition Keywords

The Layer Map file contains information regarding the metal layer numbers and text layer numbers associated with the metal layers, as well as via specifications. The format of the GDS2 layermap file allows multiple layer numbers to be specified, either as a layer number or text layer number. The layermap file syntax is

```
<Layer_name> <Layer_type> <Layer_num> <Text_layer_num> <Text_device_label>
[<Via_name> v <Layer_number> - (<upper_margin> <lower_margin>)]
```

where

Layer\_name : specifies the name of the layer

Layer\_type: can be m = metal, v = via, c = metal cap, text, temp

Layer\_num and Text\_layer\_num : can be specified by several layer-datatype pairs and layer numbers, separated by a ';' (semi-colon). Layer and datatype are separated by a ':' (colon). Text\_layer\_number can be ignored by inserting a dash '-'.

Text\_device\_label: text label for device

Via\_name : specifies the via name. Note that multiple via layers may be specified between two metal layers.

<upper\_margin> <lower\_margin> : specifies the margin around the via on the upper layer and lower layer, respectively (default - 20 nm)

The following is an example of the layermap file format:

| #Layer_name | Layer_type | Layer_num  | Text_lay_num          | Text dev label |
|-------------|------------|------------|-----------------------|----------------|
| METALL1     | m          | 1:5;10:0   | 11:1;13               |                |
| VIA1        | v          | 2;12       | - (10 10)             |                |
| METAL2      | m          | 13:0;15:50 | -                     |                |
| VIA2        | v          | 3:12;3:43  | -                     |                |
| dio_txt     | text       | 10:1;10:2  | pw_diode_t;nw_diode_t |                |

## Using Boolean Expressions in the Layermap file

Boolean expressions can be used to define logical layer operations in the GDS2DB layerMap file. GDS logical layer definitions support the following Boolean expressions:

(A&B) - intersection of the geometries of A and B

(A|B) - either geometries of A or geometries of B.

(A-B) - inclusion of only the geometries of A that do not intersect/overlap with any of the geometries in B

(!A&B) - inclusion of only the geometries of B that do not intersect/overlap with any of the geometries in A (same as B-A).

(A>B) - inclusion of only the geometries of A whose center lie within B

(A<B) - inclusion of only the geometries of B whose center lie within A .

(A>>B) - inclusion of only the geometries of A which touch B irrespective of the center.

(A<<B) - inclusion of only the geometries of B which touch A irrespective of the center.

The operators can be used to derive the following types of layers:

- device (pdifff/ndifff/poly/cm)
- contacts
- bumpvias
- marker layers

The syntax for defining logical layers is:

```
BOOLEAN <logic_layername> <layer_type> <Boolean_definition>
```

Note that the BOOLEAN keyword works on a temp type layer only, and the resultant layer cannot be metal (m), or via (v) or throughVia (tv). Nested temp layers can be defined.

GDS2DB also does some special Boolean operations independent of the layerMap file, such as:

- Metal on metal handling.
- Resistor handling (poly/diff/cm)
- Metal resistor handling

The following are GDS configuration file keywords relating to the Layermap file.

### **GDS\_MAP\_FILE**

Specifies the name of the GDS layer map file. Specify DESC if layers are listed in descending order (from top metal to bottom). Required.

```
GDS_MAP_FILE <layer_map_file> ? DESC ?
```

### **DATATYPE**

Sub-layers can be defined optionally using the following keyword:

```
DATATYPE {
 <valid sub-layer numbers>
}
```

The following are optional sections of the GDS2DB/GDS2RH configuration file.

## **Input LEF Keywords**

### **IGNORE\_LEF\_ORIGIN\_FOR\_START\_PT**

When set to '1', the ORIGIN from the start points imported from LEF file for net tracing are ignored. This becomes active when USE\_LEF\_PINS\_FOR\_TRACING is set. Default is 0.

```
IGNORE_LEF_ORIGIN_FOR_START_PT [0 | 1]
```

### **LEF\_FILE**

The .lef files for GDSII can be specified using the LEF\_FILE keyword. The syntax is as follows:

```
LEF_FILE {
 <LEF file name>
 ...
}
```

If the .lef files for GDSII are specified, the GDS2RH/GDS2DEF utility parses these files and search for the top cell macro information. If the FOREIGN keyword is specified for the macro, the GDS2RH/GDS2DEF utility converts the structure whose name is the same as macro foreign name, instead of the specified top cell name.

For the most accurate approach to obtaining the die area, the GDS2RH/GDS2DEF utility takes DIEAREA specified by the SIZE of the MACRO keywords in the .lef file. If the .lef is not available, then the GDS2RH/GDS2DEF utility estimates the area based on the bounding box of all geometries in the GDSII file.

### BOX\_CELLS\_DEBUG

In the LEF\_VIEW flow BOX\_CELLS\_DEBUG dumps connectivity information between the top level net geometries and block level pin geometries. This report can be used for debugging mismatches between LEF file port geometries and GDS geometries in the LEF\_VIEW flow. The syntax is:

```
BOX_CELLS_DEBUG {
 <cell1>
 ...
 <celln>
}
```

A report is generated in the *adsRpt/GDS/boxcells\_debug.rpt* file, in the following format (brackets indicate data values ):

```
Instance (Master) : <Instance_Name>(<MasterCell_Name>)
Pin <pinName> geometry <px1 py1> <px2 py2> of <layer_name> layer touches top
net <net_name> geometry <nx1 ny1> <nx2 ny2> of <layer_name> layer
```

All coordinates are with respect to top level and in um. Only those pins that have intersections with the top level net geometry are reported.

### LEF\_FOREIGN\_OFFSET\_HANDLE

When set to '1', shifts the input LEF pin geometries of the top-cell such that it matches the GDS of the top-cell.. Default is 0.

```
LEF_FOREIGN_OFFSET_HANDLE [0 | 1]
```

### USE\_LEF\_FOR\_LOGICAL\_CONNECTION

In the GDS + SPEF flow, use this keyword to control which master cell instance logical connections should be created in the generated DEF. The syntax is:

```
USE_LEF_FOR_LOGICAL_CONNECTION {
 <cellname1>
 <cellname2>
 ...
}
```

### USE\_LEF\_PINS\_FOR\_TRACING

Where LEF pins defined as VDD\_NETS and GND\_NETS are used for tracing, uses geometries defined in the PINS section of the LEF file of the block being extracted to trace its power and ground nets during GDS translation. No text or label based extraction is performed. However, user-specified coordinates are honored. Default: 0 (off, no LEF pin based pin tracing).

```
USE_LEF_PINS_FOR_TRACING 0/1
```

### USE\_LEF\_FOR\_TOP\_BBOX

By default (1), forces extraction to pick up the cell bounding box from the LEF file. The syntax is:

```
USE_LEF_FOR_TOP_BBOX [0 | 1]
```

### LEF\_PIN\_POWER

Enables current to be distributed to vias connected to the LEF pins. When LEF\_PIN\_POWER is On, GDS2RH/GDS2DEF traces only vias under power pin

geometry in the input LEF file and outputs the corresponding pins under those vias in the output LEF file. Default: 0 (off)

LEF\_PIN\_POWER [0|1]

#### GENERATE\_SEPARATE\_PG\_PINS\_LEF

Allows static +/-GDS2RH to keep the original LEF by setting this keyword to 1, which creates another LEF macro with power/ground nets and their geometries only, and creates an instance of this macro in the generated top cell DEF file. The default is off (0).

GENERATE\_SEPARATE\_PG\_PINS\_LEF [0|1]

#### IGNORE\_LEF\_FOREIGN\_OFFSET

Many designs that consider the LEF ‘FOREIGN offset keyword in the original LEF file generate GDS2RH/GDS2DEF output DEF with misplaced geometric elements, which results in shorts in the RedHawk run. The keyword ‘IGNORE\_LEF\_FOREIGN\_OFFSET’ can be used to ignore the offset values. The default is 0 or off, which means the FOREIGN offset value is used when GDS2RH/GDS2DEF writes out DEF geometry. If ‘IGNORE\_LEF\_FOREIGN\_OFFSET’ is set to 1, GDS2RH/GDS2DEF sets both x and y offsets to 0.

IGNORE\_LEF\_FOREIGN\_OFFSET [0 | 1]

Currently, there is no guideline to help you decide whether or not set this keyword. It depends on how the LEF file is prepared.

Note: Even when this keyword is set to 1, GDS2RH/GDS2DEF still uses the LEF foreign name for GDS structure name mapping.

### Geometry Extraction Keywords

#### APACHE\_PHYSICAL\_MODEL

When set, optimizes geometry and reduces the node count for extraction and provides significant speed-up in network processing, with a minimal reduction in accuracy. To reduce unneeded geometries, below the specified APM layer only those geometries that are necessary to keep connectivity between the pins on the APM layer are retained. All geometries (wires/vias) are kept on or above the specified APM metal layer.

For current sources/sinks:

- pins are created on the specified APM metal layer.
- the bounding-box for a group of vias above the APM layer are formed, based on the intersection of wires at APM layer and the metal-layer above APM layer. Pins are created using this bounding-box, instead of creating a single pin below each via. Pratio is scaled, based on the number of vias used to create the pin.
- in memory core regions, pins are created with a 0 pratio value at the core-extraction-starting-layer. Core-extraction-starting layer should be the same or above APM layer.

For ESD analysis, APM supports expanded ESD clamp cell modeling by extracting detailed views inside clamps and a higher level of abstraction outside the ESD cells. If APACHE\_PHYSICAL\_MODEL is set and clamp-cells are auto-detected (using the ESD marker layer in the layermap), APM ESD flow is automatically invoked.

- for P/G nets, only geometries (wires/vias) on or above the specified metal layer are retained outside clamp instances, and all geometries are retained inside clamp instances. Also, no current sources/sinks are created outside clamp-instances, and current sources/sinks are created only inside clamp-instances.
- for signal nets, there is no change in modeling geometries and current sources/sinks.

This provides faster resistance and current density checks, without compromising accuracy.

If there are switch-cells in the design, during reduction of the wire-via network below APM layer, RedHawk retains the wires/vias that are necessary to maintain connectivity to switch-instance pins, to ensure that switches are not disconnected.

Default: none. For GDS2RH only.

Syntax:

```
APACHE_PHYSICAL_MODEL <APM_metal_layer>
```

### ADJUST\_Polygon

If set to 1, removes any residual polygons with a dimension less than 0.01 micron that can occur from conversion of non-rectangular areas. Default is 0.

Syntax:

```
ADJUST_POLYGON [0 | 1]
```

### AUTOFIT\_DIFFUSION\_WITH\_POLY

When set to 1, shrinks diffusion to poly height for some advanced technology nodes in which diffusion can be extending beyond poly-cut height

Syntax:

```
AUTOFIT_DIFFUSION_WITH_POLY [0 | 1]
```

### ENABLE\_POLY\_VIA\_MERGE

When set, large circular vias are merged into a single via, rather than split into segments when set to 0. Default: Off (0).

Syntax:

```
ENABLE_POLY_VIA_MERGE [0 | 1]
```

### EXTRACTION\_STARTING\_LAYER

Sets the lowest starting layer for extraction and tracing. Then GDS2RH/GDS2DEF does not include excluded layers from the parent LEF, which are ignored. If the option ‘Traceall’ is also specified, GDS2RH/GDS2DEF traces all layers, but only extracts starting from the lowest layer specified. The default is to trace and extract all the layers specified in the layermap file. To define the starting metal layer name for power/ground net extraction in the GDSII file, use the following specification:

```
EXTRACTION_STARTING_LAYER <lowest_metal_layer> ? traceall ?
```

Both ‘GDS2RH/GDS2DEF’ and ‘GDS2DEF -m/GDS2RH -m’ utilities honor this specification.

If layers lower than the starting layer have been used to connect to higher layers, use the ‘traceall’ option, which considers connected elements on all metal layers during network tracing. However, if the option ‘traceall’ is specified, note that GDS2RH has different behavior than GDS2DEF. GDS2DEF, after tracing through all metal layers, discards all geometries below the specified EXTRACTION\_STARTING\_LAYER. GDS2RH internally invokes the APACHE\_PHYSICAL\_MODEL approach, which does not discard geometries at all layers below the EXTRACTION\_STARTING\_LAYER, but keeps connections between the pins (current sources/sinks) created at the ESL layer, through lower metal/via layers, for better voltage-drop modeling of the block

### EXTRACT\_ISOLATED\_SUBDOMAINS

When set, creates isolated sub-domains for a net: When a block contains multiple islands for the same voltage domain that is connected to different net domains at the top level,

this keyword must be set to create different islands as part of separate net domains. A name suffix is created, '\_gdsNet\*', to differentiate the different subnets-- for example, 'VDD\_gdsNet2'. The syntax is:

```
EXTRACT_ISOLATED_SUBDOMAINS [0 | 1]
```

### **IMPORT\_REGIONS**

IMPORT\_REGIONS is used to specify regions in the top cell for extraction, using corner coordinates, as follows:

```
IMPORT_REGIONS {
 <low_left_x> <low_left_y> <up_right_x> <up_right_y>
 ...
}
```

### **MERGE\_WIRE\_NETWISE**

After net tracing, when set, merges all wire geometries by net. This keyword is 1 (On) by default for gds2rh -m design flows, and 0 by default for other flows.

```
MERGE_WIRE_NETWISE [0 | 1]
```

### **NEW\_IMPORT\_REGION**

When set, imports only the required portion of the cover cell. Otherwise the polygon geometries that are partially inside the imported region are left "as-is" (without cutting) inside the hierarchy. Default Off.

```
NEW_IMPORT_REGION [0 | 1]
```

### **NO\_METAL\_IN\_VIA\_MODEL**

When turned on, generated via models have a minimum size (1x1 nm ) on top and bottom metal layers. By default (0), via models generated are the same size as cut rectangle geometries of top and bottom metal layers. The syntax is:

```
NO_METAL_IN_VIA_MODEL [0 | 1]
```

### **NON\_RECT\_DIEAREA**

When set, supports non-rectangular DIEAREA generation in DEF, based on the non-rectangular pr boundary layer geometry in the top cell. No special keyword is needed for the GDS2RH + SPEF flow. The syntax is:

```
NON_RECT_DIEAREA [0 | 1]
```

### **OUTPUT\_VIA\_ARRAY**

Setting this keyword allows grouping adjacent vias so that you can select an entire group of arrays with a single click.

```
OUTPUT_VIA_ARRAY [0 | 1]
```

### **START\_POINT\_TOLERANCE**

When location-based tracing is used, allows control over the GDS2DEF algorithm that uses DSPF x,y locations to trace and extract signal nets from the GDSII file in the signal EM flow. The default behavior of the algorithm attempts to automatically correct any imprecision between input data files, and optimizes the extraction of nets for most designs. But occasionally some x,y locations cannot be mapped precisely, which leads to shorts or missed traces, and errors later in the analysis. Under these conditions, this keyword gives you control of the mapping error correction, using the specified fixed radius to snap DSPF x,y locations to the next polygon found. Specifies the maximum distance in microns from the starting point specified in the GDS configuration file that can be used

to start tracing. The unit for the radius is the design unit from the tech file. *Optional.*  
*Default:* None

Syntax:

```
START_POINT_TOLERANCE <search_radius>
```

Example:

```
START_POINT_TOLERANCE 0.05
```

### VIA\_NAME\_PREFIX

Normal via naming creates very long names for via models, resulting in very large output DEF files. When 'VIA\_NAME\_PREFIX' is set to 0, the typical "GDSVia\_<topcell>" via name prefix is removed in order to shorten it, and reduce file size. The syntax is:

```
VIA_NAME_PREFIX [0 | 1]
```

For example, with VIA\_NAME\_PREFIX set to 1 (default), a via model name could be *GDSVia\_logic\_die\_ABC\_0\_M7\_RV\_AP\_0*. With VIA\_NAME\_PREFIX set to 0, the via model name would be *0\_M7\_RV\_AP\_0*.

## Selective Cell Hierarchy Handling Keywords

### WHITE\_BOX\_CELLS

This keyword (previously called OUTPUT\_CELLS) specifies the cells that create separate instances of the cells in the output DEF file, and retain all geometries inside cell-instances. Current sinks are created inside the box. If the keyword 'CREATE\_LEF\_MACRO\_FOR\_BOX\_CELLS' is set to 1, a LEF file is also generated for the cell. Including all cells of interest in the WHITE\_BOX\_CELL list is recommended. Wildcards are supported in specifying cell names. A report *boxcellWildcardMatch.txt* is generated in the *adsRpt/GDS* directory, which contains a list of the cells found in GDS.

For GDS2RH, all traced internal geometries are brought to the top level, and internal geometries at all metal layers are used for pin creation (unless over-ridden by PIN\_CREATION\_LAYERS property in BOX\_CELLS\_PROPERTIES).

The syntax is:

```
WHITE_BOX_CELLS {
 <cell_name1>
 ...
}
```

Note: If two instances of the same cell have a pin in one instance connected to power and the same pin in another instance of the cell connected to ground, the pin type for each pin of the WHITE\_BOX\_CELLS description in <topcell>\_adsgds1.lef is designated 'SIGNAL'. The pin names in <topcell>\_adsgds1.lef are of the form 'ADS\_PIN1', 'ADS\_PIN2', and so on. In DEF, the connectivity is handled in the SPECIALNETS section correctly.

### BLACK\_BOX\_CELLS

This GDS2RH/GDS2DEF keyword (previously called IGNORE\_CELLS) is used like the keyword WHITE\_BOX\_CELLS to control the handling of sub-block hierarchy. All cells listed in BLACK\_BOX\_CELLS are ignored when READING in GDS2 files. Wildcards are supported in specifying cell names. A report *boxcellWildcardMatch.txt* is generated in the *adsRpt/GDS* directory, which contains a list of the cells found in GDS.

Handling in GDS2RH is the same as in GDS2DEF. These cells are completely ignored; that is, their internal geometries are not considered when tracing, and instances for these cells are not created.

The syntax is:

```
BLACK_BOX_CELLS {
 <cell_name1>
 ...
}
```

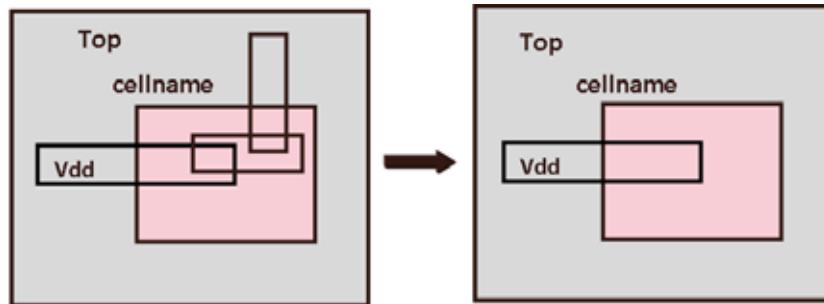
### GRAY\_BOX\_CELLS

The GRAY\_BOX\_CELLS keyword defines separate instances of cells in the output DEF file. The gray box cell specification [ 0| 1] is used in GDS2RH/GDS2DEF to control this behavior. Note that internal geometries of all metal layers are used for pin creation (unless over-written by PIN\_CREATION\_LAYERS property in BOX\_CELLS\_PROPERTIES).

Wildcards are supported in specifying cell names. A report *boxcell\_wildcard\_match.txt* is generated in the *adsRpt/GDS* directory, which contains a list of the cells found in GDS.

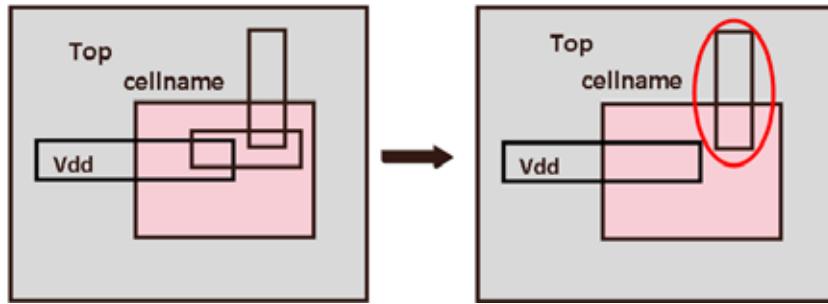
For a hierarchical design flow, if this keyword is set to 0 for a listed cell, only cell-instances are created, with no feed-through geometries converted. If set to 1, boundary boxes and feed-through geometries crossing top-level and sub-block boundaries are preserved for use in the top level view. See Figure E-5 and Figure E-6 showing the behavior of the two settings. This can be useful in analog blocks with a digital sub-block flow, by instantiating and placing sub-blocks within the block. It can also be used to resolve capacity issues, such as memory and run-time, and avoid extraction of the sub-block geometries, while properly placing bounding boxes of sub-blocks in the master cell. You use the CREATE\_LEF\_MACRO\_FOR\_BOX\_CELLS keyword, or must provide sub-block LEF files, in which case the LEF files are created automatically. If the keyword CREATE\_LEF\_MACRO\_FOR\_BOX\_CELLS is defined, a cell LEF macro is generated for gray box cells in the file <top>\_adsgds1.lef. Default: 0. The syntax is:

```
GRAY_BOX_CELLS {
 <cellname1> [0|1]
 ...
}
```



**Figure E-5    Gray\_Box\_Cells 0**

When set to 0 Gray\_Box\_Cells discards the geometries of the 'cellname' block before tracing the net. So any geometries at a higher level of hierarchy that can be traced only through the 'cellname' block are *not* retained.



**Figure E-6    Gray\_Box\_Cells 1**

When set to 1 Gray\_Box\_Cells discards the geometries of the ‘cellname’ block after tracing the net. So any geometries at a higher level of hierarchy that can be traced only through the ‘cellname’ block *are* retained.

#### **BOX\_CELLS\_PROPERTIES**

BOX\_CELLS\_PROPERTIES is used in the GDS2RH flow to streamline all box cell-related options and functionality. Note that only directly-connected metal segments are used to create box cell pins. The syntax is:

```
BOX_CELLS_PROPERTIES {
 PIN_CREATION_LAYERS [ALL | <layer1> <layer2> ...]
 UNIQUIFY [0 | 1]
 MERGE_INTERNAL_NETS [0 | 1]
 PIN_CREATION_LEVEL [min|max]
 TRACE_FROM_PIN [0|1]
 PIN_NAME_TEXT_LABELS <text_label1> <text_label2> ...
 MEMORY_SAVING_MODE [0|1]
}
```

The PIN\_CREATION\_LAYERS option specifies the layers within the box cell on which the box cell pins are created, which can be specified as follows:

ALL - uses the metal geometries from all layers within the box cell to create the switch cell pins. This is the default behavior when BOX\_CELLS\_PROPERTIES is not specified.

<layer1> <layer2> - uses only geometries from the specified metal layers for pin creation. Combinations of device layers and metal layers may also be specified, including diffusion or covertical base layers, such as

```
PIN_CREATION_LAYERS ndiff pdiff
PIN_CREATION_LAYERS ndiff met3
```

The UNIQUIFY option invokes the uniquification feature for box cells when ‘UNIQUIFY 1’ is specified. Note that MERGE\_INTERNAL\_NETS is automatically enabled when ‘UNIQUIFY 1’ is set (not available in GDS2DEF).

The MERGE\_INTERNAL\_NETS option specifies the internal nets of the box cell that have the same connectivity to top level nets across all instances of a particular cell, and are merged together if ‘MERGE\_INTERNAL\_NETS 1’ is specified.

The PIN\_CREATION\_LEVEL option, when set to “min” considers the text directly inside block, and when set to “max” considers text from all hierarchy inside block .

The TRACE\_FROM\_PIN option, when set to “0” creates pins with geometries that are connected to the text-label at the corresponding layer (should be specified in PIN\_CREATION\_LAYERS), and when set to “1” creates pins with geometries that are

connected to text label at any layer. For example, if there is text on M2 geometry inside the block and the M2 geometry is connected to M3 geometry through v23 geometry (inside the block), then PIN\_CREATION\_LAYER is specified as M3. If 'TRACE\_FROM\_PIN 1' is set, RedHawk uses M3 geometry for pin creation, even though the text label is on M2.

Using the option PIN\_NAME\_TEXT\_LABELS, GDS2RH can create pins on box cells based on text labels specified (regular expressions are accepted). A pin is created on the metal geometries touching the same layer as the text label. So if a text labels is on M1, then only M1 geometries touching the text label layer are part of the pin geometry. This feature allows creation of standard hierarchical data structures (similar to LEF/DEF). This keyword also honors text labels at lower hierarchies within box cells, as well as text labels at box cell hierarchy level. One application in which it can be used is the GDS-based ESD analysis flow, which allows standardizing pin names and locations on clamp cells (extracted as box cells), easing subsequent clamp info file creation. This option also honors text labels at lower hierarchies within box cells, as well as text labels at box cell hierarchy level. Note that only directly-connected metal segments are used to create pins, and not those connected through upper layers.

The option MEMORY\_SAVING\_MODE reduces memory usage for box cell modeling.

#### **CREATE\_LEF\_MACRO\_FOR\_BOX\_CELLS**

When specified along with OUTPUT\_CELLS, automatically generates a LEF file called <original\_top\_level\_cell>\_adsgds1.lef containing power and ground pins for the specified output cells. (This keyword was previously called OUTPUT\_CELLS\_TO\_LEF.) Default is 0 (no LEF file).

CREATE\_LEF\_MACRO\_FOR\_BOX\_CELLS [ 1 | gray | white | 0 ]

where

1: creates LEF for both gray and white box cells

gray: creates LEF for gray box cells only

white: creates LEF for white box cells only

#### **UNIQUIFY\_BOX\_CELLS**

Creates unique LEF cells when instances of the cell have different connections to Vdd and Vss, based on the external net connectivity of the cell. The keyword is used for the unquification of box cells specified using either the GRAY\_BOX\_CELLS or WHITE\_BOX\_CELLS keywords. Using the recommended and default '2' value assigns a LEF pin name based on the parent cell (top) net connection. If specifically desired, the previous functionality can be invoked using the keyword setting '1'. With this setting, if the same box cell pin connects to both power and ground (parent) nets in different instances, the created LEF pins are assigned names that do not relate to their parent cell connections. This keyword is only used for GDS2DEF, since the UNIQUIFY option in BOX\_CELLS\_PROPERTIES for GDS2RH provides the same function.

UNIQUIFY\_BOX\_CELLS [ 2 | 1 ]

### **Auto Pad Location Generation Keywords**

#### **GENERATE\_PLOC**

When specified, GDS2RH creates pads in the PINS section of the DEF file it creates. The locations where these pads are created depends on the option specified. During analysis you must set the GSR keyword 'ADD\_PLOC\_FROM\_TOP\_DEF 1' to allow GDS2RH to create pads as ploc points.

GENERATE\_PLOC [ USE\_TEXT\_LABEL | USE\_USER\_START\_PT |

```
USE_INPUT_LEF_PIN | USE_PIN_LAYERS]
USE_NIF_FILE <nif_file_path>]
```

where

**USE\_TEXT\_LABEL:** generates plocs at the location of text labels. Refer to the following GDS configuration file keywords to determine the selection of the text labels:

- USE\_TEXT\_FROM\_HIERARCHY\_LEVEL
- USE\_TEXT\_FROM\_CELLS
- USE\_TEXT\_FROM\_HIERARCHY\_BLOCK

**USE\_USER\_START\_PT:** if you have specified custom start points for net tracing, then those locations are used for PLOC creation when this keyword is specified. This keyword also may be used either independently or in combination with any of the above specified options. Can be used with **USE\_TEXT\_LABEL**.

**USE\_INPUT\_LEF\_PIN:** generates plocs that have the same size and position as all pin geometries that are present in the input LEF. To have finer control over which layer and area the plocs are created, the keyword **GENERATE\_PLOC\_FILTER** may be used.

**USE\_PIN\_LAYERS:** generates plocs that have the same size and location as all metal geometries. To have finer control over which layer and area the plocs are created, the keyword **GENERATE\_PLOC\_FILTER** may be used.

**USE\_NIF\_FILE <file>:** creates PLOCs for all ports in power and ground domains occurring in the NIF file, and generates the PINS section in the generated DEF (GDS2DB) based on the “port” section of the network information file (NIF), which is only valid in the Calibre LVS flow. The generated DEF uses the “port” section from the NIF file for the PINS section in DEF. The file name in the configuration file keyword 'NIF\_FILE <filename>' must be specified as well.

Note: The **GENERATE\_PLOC\_FILTER** keyword following can be used for controlling the plocs generated by ALL of the above keyword options, and have the same effect in each case.

### **GENERATE\_PLOC\_FILTER**

**GENERATE\_PLOC\_FILTER** maybe be used with any of the **GENERATE\_PLOC** options (**USE\_INPUT\_LEF\_PIN**, **USE\_TEXT\_LABEL**, **USE\_PIN\_LAYERS**) to fine tune pad location creation, and have the same effect in each case.

```
GENERATE_PLOC_FILTER {
 [<metal_layer_name> | ALL] [<x1 y1 x2 y2> | ALL]
 ...
}
```

where (x1, y1) and (x2, y2) represent the bottom left and top right x,y coordinates, respectively, of the region bounding box.

The layer and region combinations for the filter may be specified in three ways:

1. Combination of LAYER and REGION. For example:

```
GENERATE_PLOC_FILTER {
 M1 10 20 15 40
 M3 120 200 240 300
}
```

Here only those geometries that are inside the specific bounding boxes for M1 and M3 are created as DEF pins.

**2. LAYER** only. For example:

```
GENERATE_PLOC_FILTER {
 M1 ALL
 M3 120 200 240 300
}
```

Here, ALL geometries in M1 go to the DEF pin section, while only those geometries that are inside the specific bounding box for M3 are created as DEF pins.

**3. REGION** only. For example,

```
GENERATE_PLOC_FILTER {
 ALL 120 200 240 300
 ALL 400 60 700 250
}
```

In this case for ALL metal layers, only those geometries in the specified regions are in the DEF pin section.

## DSPF/SPEF-based standard cell flow Keywords

### **DSPF\_FILE**

Specifies the filename for the DSPF file.

```
DSPF_FILE {
 <filename>
}
```

### **SPEF\_FILE**

Specifies the filename for the SPEF file. Currently RedHawk does not support SPEF files that do not contain coordinate information.

```
SPEF_FILE {
 <filename>
}
```

### **SPEF\_CELLS**

Specifies a list of SPEF cell names.

```
SPEF_CELLS {
 <cell name1>
 ...
}
```

### **DEF\_FILE**

In the SPEF-based GDS2RH flow, you can use DEF files directly, and avoid conversion of block DEFs into LEF files

```
DEF_FILE {
 <block.defs>
 ...
}
```

## GRAY\_BOX\_CELLS

In the SPEF-based GDS2RH flow, you can use DEF files directly, as shown in the syntax below. Mixing and matching keywords SPEF\_CELLS, GRAY\_BOX\_CELLS and USE\_LEF\_FOR\_LOGICAL\_CONNECTION with different views is allowed and supported.

```
GRAY_BOX_CELLS {
 <block1> <def_view>
 <block2> <def_view>
 ...
 <blockN> <def_view>
}
```

## DSPF\_CELLS\_PREFIX

### SPEF\_CELLS\_PREFIX

These keywords remove any standard cell name prefix from blocks for the GDS master-cell name, and rename the instance of master cells with the corresponding DSPF instance name, if there is a match of GDS instance bbox to DSPF-instance bbox. The syntax is:

```
DSPF_CELLS_PREFIX {
 <Block_prefix1>
 ...
}
```

For example, suppose ‘Top’ has three blocks, BlockA, BlockB and BlockC, and standard cell buffer “BUFX1” is in all three blocks. Then the LEF includes “MACRO BUFX1”, and GDS includes the following:

```
TOP : Master Cell BUFX1
BlockA : Master Cell BlockA_BUFX1
BlockB : Master Cell BlockB_BUFX1
BlockC : Master Cell BlockC_BUFX1
```

## NO\_DEF\_LOGICAL\_CONNECTION

If set, logical connectivity data is removed from the generated DEF for standard cells (such as \*VDD and \*VSS). RedHawk builds the correct logical connectivity by keeping the explicit logical connections of the cells as specified in SPEF\_CELLS/DSPF\_CELLS/GRAY\_BOX\_CELLS/WHITE\_BOX\_CELLS in the output DEF file.

```
NO_DEF_LOGICAL_CONNECTION [0 | 1]
```

## DSPF\_CELL\_INSTANCE\_MATCH\_TOLERANCE

Specifies the tolerance in um by which the GDS instances to be matched with the DSPF/ SPEF instances, are virtually expanded with this tolerance to obtain a match. This allows pins present in the DSPF that lie slightly outside of the GDS cell boundary to be mapped. The default tolerance is 0.001 um.

```
DSPF_CELL_INSTANCE_MATCH_TOLERANCE <distance_microns>
```

## SPEF\_MAGNIFICATION\_FACTOR

Specifies a multiplication factor for coordinates in SPEF, so original SPEF coordinates of ‘A’ would now have coordinate values of ‘<factor>\*A’. For example, if the SPEF\_MAGNIFICATION\_FACTOR is specified as 0.9 in the configuration file, GDS2DEF

reverts back to the coordinates of the SPEF from 'Y' to 'Y/0.9' and then uses the value 'Y/0.9' for processing. Default is 1.0.

```
SPEF_MAGNIFICATION_FACTOR <factor>
```

### **USE\_LEF\_FOR\_LOGICAL\_CONNECTION**

Specifies that LEF is used to create logical connections between TOP and the boxcells based on the LEF pins. The LEF file provides the pins and their corresponding port geometries for the box cells. A report file *logicalconnections.txt* is generated.

```
USE_LEF_FOR_LOGICAL_CONNECTION [0 | 1]
```

## **Automated Switch Cell Handling Keywords**

### **DEFINE\_SWITCH\_CELLS**

Specifies the switch name, type, and the LEF pin name mapping. This is a mandatory keyword regardless of mode. Must be used with EXTRACT\_SWITCH\_CELLS and IDENTIFY\_INTERNAL\_PG\_NETS.

```
DEFINE_SWITCH_CELLS {
 <switch_name> [HEADER|FOOTER] <input/ext LEF pin name>
 <output/int LEF pin name> <control_pin1> <control_pin2> ...
 ...
}
```

#### Example

```
DEFINE_SWITCH_CELLS {
 sw1 HEADER VDD_EXT VDD_INT EN
 sw2 HEADER VDD_EXT VDD_INT EN
}
```

### **EXTRACT\_SWITCH\_CELLS**

Specifies the switch name, type and the internal and external nets that it connects to. This is a mandatory keyword regardless of mode. Must be used with DEFINE\_SWITCH\_CELLS.

```
EXTRACT_SWITCH_CELLS {
 <switch_name> [HEADER | FOOTER] <ext_net_name> <int_net_name>
 ...
}
```

#### Example

```
EXTRACT_SWITCH_CELLS {
 sw1 HEADER ext_vdd1 int_vdd1
 sw1 HEADER ext_vdd2 int_vdd2
}
```

### **SWITCH\_CELLS\_PROPERTIES**

This keyword controls switch cells pin creation, and automatically creates the power switch hierarchy and a switch model file to handle cases in which the switch hierarchy is flattened in GDS layout.

After you run APLMMX with keyword SWITCH\_MODEL\_TABLE to dump out the power switch transistor information in the \*.gdsmmx file, then you use this keyword for the embedded power switch hierarchy and power switch model file creation:

```
SWITCH_CELLS_PROPERTIES {
```

```
CREATE_HIERARCHY {
 USE_SWITCH_XTOR
 XTOR_FINGER_PITCH
 XTOR_FINGER_WIDTH
 EXTEND_SWITCH_BBOX
 MAX_SWITCH_WIDTH
 MAX_SWITCH_HEIGHT
}
VIRTUAL_NET_PAIRS {
 <extNet_name> <intNet_name>
}
SMASH_GEOMETRIES [1|0]
USE_SWITCH_INT_PIN_FOR_TRACING [1|0]
PIN_CREATION_LAYERS [ALL| <MET2>| <MET1>| <cont_lay>]
}
```

where

- USE\_SWITCH\_XTOR - keyword to enable embedded power switch handling
- XTOR\_FINGER\_PITCH - poly-gate perpendicular direction grouping threshold, [8\*L], based on max L of all switches
- XTOR\_FINGER\_WIDTH - poly-gate parallel direction grouping threshold, [1.5\*W], based on max W of all switches
- EXTEND\_SWITCH\_BBOX - to extend identified switch finger bounding box, default =0
- MAX\_SWITCH\_WIDTH - to restrict the maximum width of the switch, [2\*XTOR\_FINGER\_WIDTH]
- MAX\_SWITCH\_HEIGHT - to restrict the maximum height of the switch, [20\*XTOR\_FINGER\_PITCH]
- VIRTUAL\_NET\_PAIR - defines the internal and external net pairs of the switch cells
- SMASH\_GEOMETRIES: when set to 1 (default), all switch cell internal geometries are pushed to the top level, which makes it similar to a white box cell approach. When set to 0, none of the switch cell internal geometries are pushed to the top level, which makes it like a gray box cell.
- USE\_SWITCH\_INT\_PIN\_FOR\_TRACING: when set, forces GDS2DB to use text labels for internal pins inside the switch cell hierarchy to trace the internal P/G nets. This keyword can only be used when the switch cell hierarchy inside the GDS file has text labels corresponding to <internal pin>, as defined in the keyword DEFINE\_SWITCH\_CELLS. This eliminates the need for you to specify coordinates for tracing internal P/G nets when text labels for them are not brought out at top level hierarchy.
- PIN\_CREATION\_LAYERS: specifies layers on which pins are created. When a contact layer (name from layer map file) is specified, the pins are created on MET1 based on the contact via top MET1 geometries.

The following is a sample switch cell GDS2DB configuration file:

```
SWITCH_CELLS_PROPERTIES {
 PIN_CREATION_LAYERS ALL
 CREATE_HIERARCHY {
 USE_SWITCH_XTOR
 XTOR_FINGER_PITCH 0.1 # Switch grouping keyword
 }
}
```

```
XTOR_FINGER_WIDTH 0.6 # Switch grouping keyword
EXTEND_SWITCH_BBOX 10 # Switch grouping keyword
MAX_SWITCH_WIDTH 4 # Switch grouping keyword
MAX_SWITCH_HEIGHT 4 # Switch grouping keyword
}
VIRTUAL_NET_PAIRS { # External/Internal net pair defin
 VSS vssc # External/Internal net pairs defin
 VDD vddpi # External/Internal net pairs defin
}
} # External/Internal net pairs defin
}
```

### **SWITCH\_MODEL\_TABLE**

Specifies a switch model table file that has a list of device models with the following information:

- On-state resistance of the power switch
- On-state saturation current of the switch IDS
- Off-state leakage current of the power gate

A sample *switch\_model.tbl* file is shown below:

```
DeviceModel Width(um) Length(um) Ron(Ohms) IdSat(Amps) Ileak(Amps)
* 0.8 0.04 60 1.2e-3 0.1e-8
pch 0.8 0.04 60 1.2e-3 0.1e-8
nch 0.8 0.04 34 2.2e-3 0.2e-8
...
...
```

Note that only one width and length combination per device model must be specified in the lookup table. The syntax is:

```
SWITCH_MODEL_TABLE <switch_table.mdl>
```

### **Other Keywords**

#### **GENERATE\_GSR\_ONLY**

When set, GDS2RH only creates the GSR file *.gds2rh/.gds2rh.v.gsr*, and does not run RedHawk. The syntax is:

```
GENERATE_GSR_ONLY [0 | 1]
```

#### **OUTPUT\_DIRECTORY**

To write out the DEF/LEF files to a different directory than the current run directory, use the following specification:

```
OUTPUT_DIRECTORY <output_directory>
or use the GDS2RH/GDS2DEF command options '-out_dir <dir_name>' or '-log <dir_name>'.
```

#### **COMPRESS\_DEF**

This GDS2RH/GDS2DEF keyword performs a gzip on the output DEF file. When this keyword is turned on, the output DEF file is gzipped and given the filename *<top\_cell>.def.gz*. The default value is '0' (not gzipped).

```
COMPRESS_DEF [0 | 1]
```

**BUSBITCHARS**

Defines the characters used for bus bits in the generated DEF/LEF file. If not defined, GDS2RH/GDS2DEF uses either BUSBITCHARS defined in the original LEF file, or the default value “[ ]”.

```
BUSBITCHARS <char>
```

**DEFAULT\_NET\_CURRENT\_PIN\_CREATION**

When using the command ‘gds2rh’ and the “Extended” option of the GSR keyword GDSII\_FILE, by default current pins are created on the EXTRACTION\_STARTING\_LAYER and/or CORE\_EXTRACTION\_STARTING\_LAYER based on the vias above that layer. However, when a net does not have any geometry on the lowest via layer (that is, if the EXTRACTION\_STARTING\_LAYER is M1), then pins are created on VIA12, or the net does not have any vias traced, the default current pins creation methodology for GDS2RH “extended” does not create any pins for such nets. However, if you set 'DEFAULT\_NET\_CURRENT\_PIN\_CREATION 1', pins are created on the lowest metal layer. Default: 0.

```
DEFAULT_NET_CURRENT_PIN_CREATION [0 | 1]
```

**ESD\_CELLS**

When this keyword is specified, GDS2DB can create hierarchy for clamp cells from a flattened GDS for GDS-based ESD static analysis. A unique virtual cell is created from the specified ESD components with all the geometries in all the layers from all the hierarchies in that region. The geometries that extend outside of the specified ESD regions are cut and made part of the ESD cell. Syntax:

```
ESD_CELLS {
 CREATE_HIERARCHY {
 [USE_CELL_NAMES | USE_ID_LAYER | USE_REGION]]
 RESTRICT_ID_LAYER_MERGE_ACROSS_HIERARCHY [1|0]
 TRACE_THROUGH_SERIAL_DIODE [1|0]
 TRACE_THROUGH_STACKED_DIODE [1|0]
 POLARITY_BASED_CLAMP_INFO [0|1]
 DUMMY_DIODE_NAMES {
 <dummy_diode_name1>
 <dummy_diode_name2>
 ...
 }
 CELL_NAMES {
 <cell_1>
 ...
 }
 REGION {
 <llx> <llx> <urx> <ury>
 ...
 }
 ? TOLERANCE <global_tolerance> ?
 ? TOLERANCE_PER_ID {
 <id_layer1> <tolerance1>
 <id_layer2> <tolerance2>
 ...
 }?
 }
}
```

where

**USE\_CELL\_NAMES/ CELL\_NAMES:** uses the specified CELL\_NAMES to find the region of the overlapping instances of the cell. All instances of this cell are identified and the overlapping instances, or those meeting the specified criteria, are considered together to form a virtual ESD cell. The created virtual ESD cell covers all geometries inside all of these instances and is instantiated under the TOP cell. The instances of the cell specified in CELL\_NAMES section are not retained in the database or in the generated LEF/DEF files. Usually ESD cells are multi-finger devices formed by small parameterized cells called pcells. These pcell names should be specified in the CELL\_NAMES section.

**USE\_ID\_LAYER:** uses the GDS ID layer specified in the layer map file to find the region enclosing the geometries from these layers.

With overlapping layers, all the ID layer geometries are found from all the hierarchies, and the ID layers are then flattened to the top. The overlapping ID layers are grouped together to form a virtual ESD cell.

With non-overlapping layers, the cell is created by considering the bounding box covering the region of ID layer geometries within the maximum distance specified.

For example, in the GDS layer map file:

```
ESD_diode ESD 122:0 -
```

This keyword can create Clamp Cells hierarchy independent of the GDS hierarchy if the RESTRICT\_ID\_LAYER\_MERGE\_\* option is set to 0.

**RESTRICT\_ID\_LAYER\_MERGE\_ACROSS\_HIERARCHY :** the default value of 1 means that the Clamp Cell Hierarchy is created only within the same GDS cell or hierarchy.

**USE\_REGION/ REGION:** when specified, uses the defined REGION to create a virtual ESD cell that covers all the geometries in the region, and has all the geometries from all layers in the region. Geometries extending outside of the region are cut and made part of the ESD cell. This cell is instantiated under the TOP cell.

**TRACE\_THROUGH\_SERIAL\_DIODE :** when specified, traces internal nets connecting any serial diodes created by the CREATE\_HIERARCHY option, which are not traced if this option is not used. Top-level nets are created for these internal trace-through nets. This keyword is used with the CREATE\_HIERARCHY option. A file, *clamp.ppi*, is generated for this feature that has the pin and layer information. Example portion of PPI file:

```
serial_diode_D1_1 PIN { pad } LAYER { PDIFF NDIFF } BBOX { 0.0000 0.0000
0.8150 1.1330 }
serial_diode_D2_1 PIN { pad vdd } LAYER { PDIFF NDIFF } BBOX { 0.0250 0.0000
0.8400 1.1460 }
serial_diode_D2_1 PIN { Net1 } LAYER { PDIFF NDIFF } BBOX { 0.0000 2.9170
0.8660 4.0630 }
```

**TRACE\_THROUGH\_STACKED\_DEVICE:** when specified, traces internal nets connecting any stacked devices created by the CREATE\_HIERARCHY option, which are not traced if this option is not used.

**POLARITY\_BASED\_CLAMP\_INFO** - when set, identifies the actual cathode/anode of diode devices and creates pin pairs with the correct polarity.

**DUMMY\_DIODE\_NAMES :** specifies the diode cell names that can be shorted to its own terminal. This kind of shorted diode areas could be present inside the custom hierarchy and are not considered for pin creation when this keyword is used.

**TOLERANCE:** optionally specifies the maximum distance between regions based on the GDS ID layer, or between instances defined in CELL\_NAMES, to consider merging them (microns).

**TOLERANCE\_PER\_ID:** optionally specifies the maximum distance between regions based on the GDS ID layer, or between instances defined in CELL\_NAMES, to merging them (microns).

Note that even if two regions are overlapping, or within the tolerance limit, if they belong to different hierarchies, they are retained as different cells.

### **EXTRACT\_METAL\_FILLER**

Metal fills are segments of metal added due to DRC rules to increase the metal density. These metals are floating and not connected to any power domain. Although as conductors they are meaningless, they are very important in calculating the metal density. Metal density in turn is used in calculating metal strip resistance. If this keyword is used, GDS2RH/GDS2DEF extracts all metal fills in each layer to get the correct metal density calculation. Default: 0.

```
EXTRACT_METAL_FILLER [0 | 1]
```

An example of the output in the DEF file is as follows:

```
- MFILL
+ ROUTED M1 10400 (501120 -10) (* 88250)
 NEW M1 460 (672290 800050) (675670 *)
 NEW M1 1000 (687220 829780) (697940 *)
 NEW M1 240 (701920 786750) (705010 *)
```

### **GDS\_INSTANCE\_NAME\_KEY**

Allows reading user-specified instance annotations in GDS. If the GDS has instance name annotation in custom properties, you can provide a key-value pair to the GDS translation engine to retain the instance names in the generated LEF/DEF files using this keyword.

```
GDS_INSTANCE_NAME_KEY <key_number>
```

### **INTERNAL\_DBUNIT**

Specifies the internal GDS2RH/GDS2DEF dimensional unit resolution (multiplier for actual user unit, such as microns). Default is 2000. If ‘UNITS’ keyword in DEF\_FILE\_DEFINITIONS is changed to 1000, this keyword value must match.

```
INTERNAL_DBUNIT [1000 | 2000]
```

### **MAGIC\_COMPATIBLE**

When MAGIC\_COMPATIBLE is set to 1, GDS2RH/GDS2DEF treats reference array (AREF) in the GDS file in a manner compatible with the Magic tool. The default is 0. The syntax is:

```
MAGIC_COMPATIBLE [0 | 1]
```

### **MERGE\_DUP\_STRUCTURE**

When turned on, merges multiple GDS files with the same name. By default, the most recently processed structure replaces the previously-processed structure, when there are multiple GDS files with the same name. For merging, set the value to “yes”/“1”/“true”.

```
MERGE_DUP_STRUCTURE <value>
```

**MEMORY\_SAVING\_MODE**

The MEMORY\_SAVING\_MODE keyword enables reduced peak memory use when more than 16 GB may be needed, and reduces memory swapping (gds2def only). The default is 1 (On).

```
MEMORY_SAVING_MODE [0 | 1]
```

**OUTPUT\_APACHECELL**

When OUTPUT\_APACHECELL is set to 1, “\_APACHECELL” is appended to original top cell name. When set to 0, cells are generated in COMPONENTS session with the original top cell name. The default is 1.

```
OUTPUT_APACHECELL [0 | 1]
```

**OUTPUT\_DETAILED\_DEF\_PIN**

The DEF output produced by GDS2RH/GDS2DEF does not have physical geometries for the power and ground pins. By turning on OUTPUT\_DETAILED\_DEF\_PIN, the physical geometries of power and ground pins in the LEF file are included in the DEF file. The size and location of these pins are obtained from the user-provided LEF file of the macro cell. Default: 1.

```
OUTPUT_DETAILED_DEF_PIN [0 | 1]
```

**FULL\_GDSII\_INFO**

By default, GDS2RH generates the imported GDS data summary to the *adsRpt/GDS/gdsii.info* file. It has information about the first 500 GDS structures (blocks) that have highest number of geometries. In order to see all the GDS structures (blocks), use this keyword. Default 0.

```
FULL_GDSII_INFO [0 | 1]
```

**PREFLATTEN\_CELLS**

Specifies that GDS structures (blocks) in GDS2RH are flattened before hierarchical net tracing.

```
PREFLATTEN_CELLS {
 Cell1
 Cell2
}
```

This keyword can also be used to flatten the top cell as well, as follows:

```
TOP_CELL top_cell
PREFLATTEN_CELLS {
 top_cell
}
```

**Note:** pre-flattening the top cell has a run time impact.

**VALIDATE\_MODEL**

When set, creates command file and GSR from specified inputs, loads the output LEF and DEF files, and runs RedHawk. When the GENERATE\_GSR\_ONLY configuration file keyword is set, GDS2RH only creates the GSR file *.gds2rh/.gds2rh.v.gsr*, and does not run RedHawk. You can use this keyword to configure RedHawk runs the way you wish. *Optional; default: none.*

Syntax :

```
VALIDATE_MODEL {
 TECH_FILE <tech_file_path>
 LIB_FILES {
 <lib_files>
 ...
 }
 PAD_FILES {
 <pad_files>
 ...
 }
 ADD_PLOC_FROM_TOP_DEF 1
}
```

## Running gds2rh or gds2def

---

To run gds2rh or gds2def, use the following invocation syntax:

Syntax:

```
[gds2drh | gds2def] <config_file> ? <GDSII_filename> ?
? -out_dir <output_dir> ? -log <log_dir> ?
where
<config_file> : specifies the gds2def/gds2def configuration file
<GDSII_filename> : specifies the GDS filename
-out_dir <output_dir>: specifies the output results directory, the same as the
 OUTPUT_DIRECTORY keyword in the GDS configuration file, where all files
 (including all LEF/DEF, and pratio files) are written. If OUTPUT_DIRECTORY
 and '-out_dir' are both specified, the '-out_dir' specification takes
 precedence.
-log <log_dir>: specifies the directory into which all log files are redirected. By
 default log files are written to ./adsRpt/gds.log, gds.warn, and gds.err. If you
 specify the -log option, the log files go into that directory. Also allows absolute
 log directory path definitions.
```

The input to GDS2RH/GDS2DEF utility is the GDSII configuration file and GDSII file. The output is top-level .def and .lef files, as well as the log and error/warning files.

## Special Applications

---

### Boundary Layer Definition

GDS2RH can create PR boundaries based on “b” type layer definitions in GDS. When 'RETAIN\_PRBOUNDARY\_FOR\_TOP\_BBOX 1' is set in the GDS2RH/GDS2DEF configuration file (Default: 0), the PR boundary layer in the top-cell is used as the bounding-box for the top-cell, even if there are geometries in the top cell extending outside the PR boundary. For example, in the GDS layer map file, you can define a “b” type entry as follows:

```
...
BDRY b 0:0 -
```

Syntax:

```
RETAIN_PRBOUNDARY_FOR_TOP_BBOX [1 | 0]
```

When you use 'RETAIN\_PRBOUNDARY\_FOR\_BOX\_CELL\_BBOX 1', GDS2DB honors the sub-block PRBOUNDARY keyword to get the sub-block bounding box in the hierarchical flow. This enables you to provide LEF for sub-blocks (generated again with PRBOUNDARY in layer map), which can be stitched up with the top-level seamlessly.

Syntax:

```
RETAIN_PRBOUNDARY_FOR_BOX_CELL_BBOX [1 | 0]
```

### Specific metal resistor support using the marker layer

By default GDS traces all metal layers. If you want to break GDS tracing, a metal resistor on the marker layer can be used. For specific metal-resistor situations in which two nets, for example netA and netB, are connected through a metal resistor, the metal resistor can be drawn in layout with the metal layer and a marker layer inside GDS. The metal resistor must be treated as "open", so that netA and netB are not shorted. This type of metal resistor treatment is common to separate two nets in the layout. The layer map entry would look as follows:

```
<metal_resistor_name> metalres <resistor layer num> <metal layer num>
```

For this type of example, the GDS layer map file would be as follows:

```
m1 m 31:0 - # normal metal definition
...
m3 m 33:0 - # normal metal definition
mr1 metalres 110:11 31:0 # marker layer
mr2 metalres 110:12 32:0 # marker layer
mr3 metalres 110:13 33:0 # marker layer
```

### Modeling through-via support

Native through-vias can be modeled as vias connecting two non-consecutive metal layers, using the keyword 'ENABLE\_NATIVE\_THROUGH\_VIAS 1'. If this keyword is not specified, or is set to 0, through-vias are modeled as stacked-vias from bottom to top layer, instead of as a single through-via from a bottom to a top metal layer; in other words, all metal bounding boxes from upper layer to lower metal layers appear in the viamodel definition in this case. The format for specifying through-vias in the layer map file is as follows:

```
tv -
THROUGHVIA
```

Example layer map file:

```
TSV tv 10 -
BM m 11 -
VIA_DUM v 999 -
M1 m 31 131
MCAP c 2 - MCAPM2VIA
VIA1 v 51 -
M2 m 32 132
```

```
THROUGHVIA TSV BM M2
```

where

TSV: the through-via layer name. Via names (for the through-via geometries) in the output DEF file also start with this name.

BM: specifies the lower metal layer of the "TSV" via layer.

M2: specifies the upper metal layer of the "TSV" via layer.

## Bump Via Support

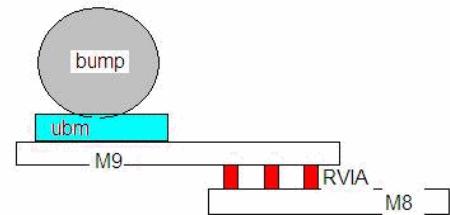
Bump vias are special vias between two metal layers in addition to the regular vias between these metal layers, as shown in Figure E-7. Bump vias have the following properties:

- their electrical properties differ from normal vias between the same metal layers, so they need to be modeled as a separate layer.
- they are generally defined as logical operations between two other GDS layers.

As with all layer specifications, the Bump Via layer has to be specified in the GDS layer map file, and similarity with “THROUGHVIA” support is maintained. The format for specifying bump vias in the GDS map file is as follows:

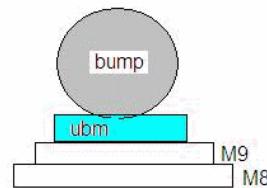
```
BUVPVIA <bump_via_layer_name> <logical_layer_operation>
 <associated_via_layer_name>
```

- Normal connection from M8 to RDL using RV via layer



The second type (No Vias – direct connection):

- Direct connection from M8 to RDL and bump



**Figure E-7      Bump via modeling**

A sample layer map file with bump vias would look like:

```
M8 m 38 138
RV v 85 -
M9 m 74 126
CB2 temp 86 -
CBD temp 169 -
BUVPVIA BV (CB2&CBD) RV
```

where

BV: the bump via layer name. The via names (for the bump via geometries) in the output DEF file also start with this name.

(CB2&CBD): specifies the logical layer operation that defines the bump via geometries. Any valid regular expression is supported in the GDSMMX logical layer operations for device layers. As is the case of device layer logical layer operations, all layers that are part of the expression MUST already be defined as “temp” layers in the layer map file.

RV: specifies the associated via layer name--that is, the regular via layer name at whose level the Bump Via is created. So the bump via has the same upper layer and lower layer as the ‘RV’ via layer.

## DSPF/SPEF file-based GDS2DEF Standard Cell Flow

### Input requirements

- Standard cell LEF macros
- Schematic information in the form of either SPEF or DSPF files
- All GDSII-related data
- Layer map
- Special RedHawk license
- GDS2DEF configuration file with required keywords (see [section "DSPF/SPEF-based standard cell flow Keywords", page E-898](#))
- Note that the schematic connectivity data must be completely synchronized with layout (GDSII) for the flow to work properly, particularly instance name matching.

### Flow Steps

1. Standard LEF cells:
  - a. Read standard cell LEF macro names and treat them as GRAY\_BOX\_CELLS --that is, standard cell content is not extracted. The size of these cells is taken from the LEF file, and the origin is obtained from the GDS.
  - b. No LEF file is generated from GDS2DEF.
2. Box Cells Creation:
  - a. Read the DSPF/SPEF file and read the instance names, pin information (location, name, type), and net information.
  - b. Run GDS2DEF on the provided GDSII data and create the standard cell hierarchy defined in the LEF\_FILE keyword.
  - c. Form a Bounding Box with pin location information from the DSPF/SPEF file. The Bounding Box for cells specified in GRAY\_BOX\_CELLS or DSPF/SPEF\_CELLS are obtained from the GDS. The generated instance names are of the form “adsU\*”.
3. Instance Name Matching:
  - a. Iterate through the GDS instances and find whether the GDS instance Bbox overlaps any DSPF/SPEF instance Bbox.
  - b. If only one overlapping instance is found, the GDS instance is mapped to that DSPF/SPEF instance.
  - c. Based on the Bounding Boxes of the adsU\* instances, find the instPins completely inside the BB. If found, use the same name for the adsU\*.
  - d. Rename the adsUx with an actual DSPF/SPEF instance-name.
  - e. If multiple overlaps are found (supported only in SPEF currently), post process them.
4. Add Nets from SPEF or DSPF file.
5. Use LEF inputs for logical connections with keyword USE\_LEF\_FOR\_LOGICAL\_CONNECTION

6. When the DSPF/SPEF\_CELL\_INSTANCE\_MATCH\_TOLERANCE configuration file keyword is specified, the GDS instance bounding box is virtually expanded by this tolerance (um) to allow mapping pins within the tolerance distance. This expansion is performed only when overlaps to any DSPF/SPEF instances are found, which does not change the actual Bbox of the instance in the output DEF.

### Test case illustration

- Available Standard LEF cells (to be treated as Gray\_Box\_Cells):

```
PPL_ANA_LOC_65
IVHVY4
IP1_ANA_LOC_65
VDDIOCO_5V_ANA_LOC_58_1
```

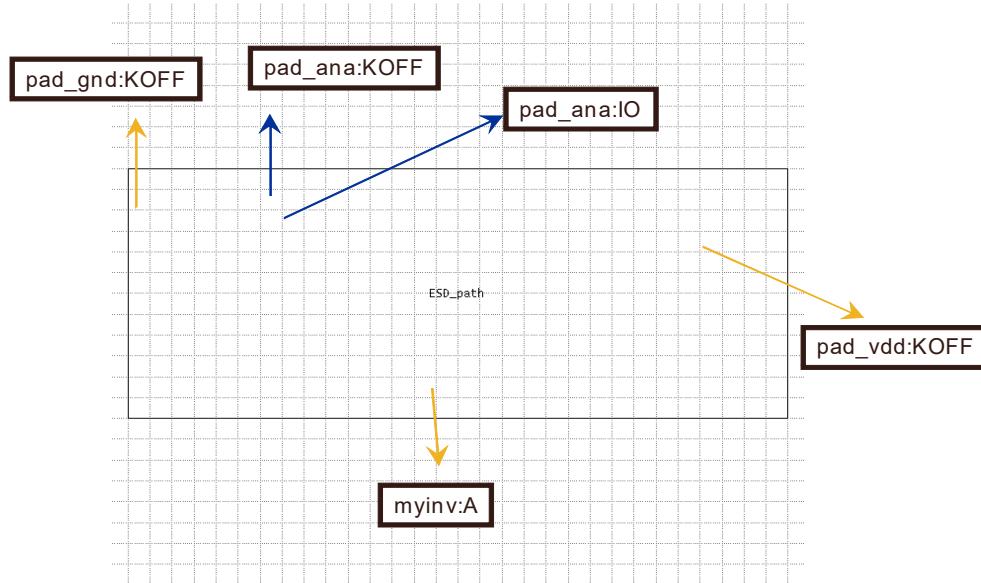
- Defined DSPF pins:

```
* | NET PAD 1.92567e-15
* | I (pad_ana:IO pad_ana IO O 0 825.71 0)
* | I (myinv:A myinv A I 0 1316.71 -877.26)
* | NET KOFF 1.03934e-12
* | I (pad_gnd:KOFF pad_gnd KOFF B 0 65.60 86.65)
* | I (pad_ana:KOFF pad_ana KOFF B 0 764.78 86.65)
* | I (pad_vdd:KOFF pad_vdd KOFF B 0 2696.97 86.65)
```

- Additional Nets from DSPF file:

Net PAD, KOFF at the top level

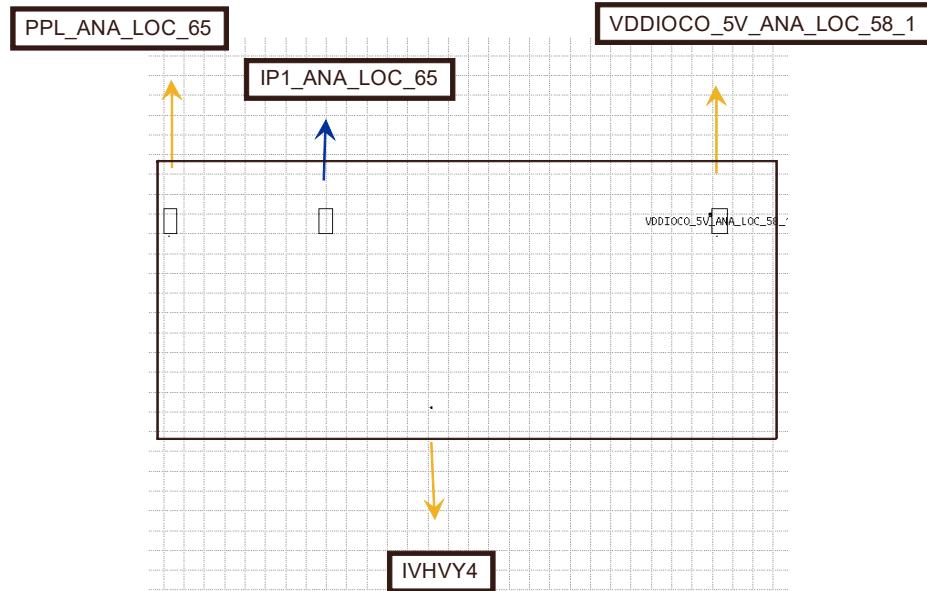
There are five DSPF instPins available, as shown in Figure E-8. In this example, clearly *pad\_ana:KOFF* and *pad\_ana:IO* belong to one instance, *pad\_ana*.



**Figure E-8     DSPF flow example**

Also in this example, there are four standard cells placed, as shown in Figure E-9.

After doing the name matching based on instPin, the name for the adsU\* blocks are identified.



**Figure E-9     Example GDSII layout showing standard cells**

### Generated DEF

The DEF generated from the example is shown in

| <u>Existing GDS2DEF</u>                                                                                                                                                                                                                                                                                 | <u>SPEF/DSPF based GDS2DEF</u>                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> NETS  COMPONENTS 5; - adsU1 ESD_path_APACHECELL + PLACED (0 -881650) N ; - adsU2 VDDIOCO_5V_ANA_LOC_58_1 + PLACED (26959700) S ; - adsU3 IP1_ANA_LOC_65 + PLACED (7541200) S ; - adsU4 PPL_ANA_LOC_65 + PLACED (-106600) S ; - adsU5 IVHVY4 + PLACED (1315690 -880350) FN ; END COMPONENTS </pre> | <pre> NETS  PAD (pad_ana IO) (myinv A) KOFF(pad_gnd KOFF) (pad_vdd KOFF)  COMPONENTS 5; - adsU1 ESD_path_APACHECELL + PLACED (0 -881650) N ; - pad_vdd VDDIOCO_5V_ANA_LOC_58_1 + PLACED (26959700) S ; - pad_ana IP1_ANA_LOC_65 + PLACED (7541200) S ; - pad_gnd PPL_ANA_LOC_65 + PLACED (-106600) S ; - myinv IVHVY4 + PLACED (1315690 -880350) FN ; END COMPONENTS </pre> |

**Figure E-10    DEF file generated from example DSPF flow**

### Reports Generated

- Multiple instance mismatch occurrences are recorded in files *Gds\_Spef\_Mismatch.txt* and *Gds\_DSPF\_Mismatch.txt*, which are used to extract mapping information. The format of these reports is shown below:

\*\*\*\* Cells Under Top Cell Hierarchy for which Input Lef is not specified

```
<Cell Name> <No of Instantiations>

***** SPEF CELLS NOT HAVING ANY INSTANTIATIONS IN TOP CELL HIERARCHY
<Cell Name>

***** GDS Instance Bbox having overlap with SPEF Instance Bbox
GDS Instance adsUx (cell name:) at loc LL: (xx, yy) UR: (xx, yy) is
matching to SPEF instance(s) (cell name: ...) Inst1 (cell name: ...) Inst2

SPEF instance Inst1 and GDS instance adsUx (cell name: ...) are matching,
mapping adsUx to Inst1

***** Spef Instances having no overlap with any GDS cells
Instance "Inst1" has no overlap with any cell in GDS.
BBOX of which from SPEF File is :LL(xx yy) UR(xx yy)
```

- Block-centric connectivity report, *logicalconnections.txt*. The report shows for each cell and for each instance, which TOP domains the internal pins are connected to.

### Restrictions on the DSPF/SPEF flow

- Using both DSPF keywords and SPEF keywords is not allowed. A check for this is performed, and errors displayed for violations identified.
- Commands 'gds2def-m' and 'gdsmmx' should not be used with DSPF/SPEF flows. A check for this is performed, and errors displayed for violations identified.

### Select Cell Hierarchy Modeling

The “box cell” concept is useful in manually creating instance hierarchy inside the top cell to model sub-blocks for a top GDS cell in a hierarchical design. You may process a sub-block (hierarchical block) of the design separately and use the XX\_BOX\_CELLS configuration file keywords to create hierarchy. There are three types of box cell models:

- White box sub-block - creates a hierarchy and also all the sub-block geometries, including current sinks. Define using the keyword WHITE\_BOX\_CELLS.
- Black box sub-block - ignores the sub-block, including internal geometries. Define using the keyword BLACK\_BOX\_CELLS.
- Gray box sub-block - creates an instance of the sub-block without any geometries. Using Gray Box Cells, GDS2RH/GDS2DEF and gdsmmx can handle large designs using manual hierarchy creation. Define using the keyword GRAY\_BOX\_CELLS.

See the detailed syntax and usage of these keywords in [section "Selective Cell Hierarchy Handling Keywords", page E-893](#), and [section "Creating the GDS2RH/GDS2DEF Configuration File", page E-878](#), for a full description of the configuration file.

Several cases of using the Gray Box cells concept are described in the following section.

### Case 1 - reducing memory use and run-time

This procedure resolves capacity issues, avoids extraction of geometries of sub-blocks, yet properly places bounding boxes of sub-blocks in the master. If the LEF file for the sub-block is already available from another tool, or by running GDS2RH/GDS2DEF for the sub-block first, use the LEF file to properly place the sub-blocks in DEF COMPONENTS

section. The LEF must have the correct ORIGIN and SIZE for the sub-block. In this case put into the GDS2RH/GDS2DEF configuration file:

```
LEF_FILES {
 ./cellabc.lef
}
GRAY_BOX_CELLS {
 cellname 0
}
```

### Case 2 - no LEF, but need bbox and pins at top level

This procedure is useful if you do not have a LEF file for the sub-block, but still want only the bounding box and pins of sub-block for top-level analysis. In this case put into the GDS2RH/GDS2DEF configuration file:

```
GRAY_BOX_CELLS {
 cellabc 0
}
CREATE_LEF_MACRO_FOR_BOX_CELLS 1
```

Internally, geometries are extracted for the sub-block to get the boundary box, so this has more memory and run-time use than in case 1.

### Case 3 - no LEF, need bbox and pins, and retain P/G routing

This procedure is useful if you do not have a LEF file for the sub-block, but still want only the bounding box and pins of sub-block during top-level analysis. Also, you need to retain the P/G routing of block/top, which are only through the sub-block. In this case put into the GDS2RH/GDS2DEF configuration file:

```
GRAY_BOX_CELLS {
 cellabc 1
}
CREATE_LEF_MACRO_FOR_BOX_CELLS 1
```

Internally, geometries are extracted for the sub-block to get the boundary box. Tracing is enabled through the sub-block.

### Case 4 - have LEF, need to retain P/G routing

LEF file for the sub-block is available, but need to retain the P/G routes of block/top, which are only through the sub-block. In this case put into the GDS2RH/GDS2DEF configuration file:

```
LEF_FILES {
 ./cellabc.lef
}
GRAY_BOX_CELLS {
 cellabc 1
}
```

Internally, geometries are extracted for the sub-block to enable tracing through the sub-block.

## Clamp Cell identification based on marker layer

ESD clamp cells can be automatically identified and extracted from GDS by specifying marker layers in the layer map. A sample layer map is shown below:

```
<device_name> ESD <cell_ID_layer> <pin_ID_layer1> <pin_ID_layer2>
```

```
CLAMP ESD 215:20 215:1 215:2
D1 ESD 215:21 215:1 215:2
D2 ESD 215:22 215:1 215:2
D3 ESD 215:23 215:1 215:2
D4 ESD 215:24 215:1 215:2
D5 ESD 215:26 215:1 215:2
D6 ESD 215:27 215:1 215:2
```

The presence of a marker layer inside the cells causes GDS2RH/GDS2DEF to classify them as ESD devices and output them in LEF/DEF format.

## gds2rh -m and gds2def -m

---

The primary use of gds2rh -m or gds2def -m is to convert GDSII files of memory cells, but it can also be used on blocks such as analog cells, I/O cells and other IP for which DEF is not available. The pin-based memory model supports the multi-Vdd concept and contact layers natively.

**Note:** there is a difference between the outputs of gds2rh and gds2rh -m commands:

- the *gds2rh \*.config* file creates a *\*.def* file
- the *gds2rh -m \*.config* file creates a *\*\_adsgds.def* file

Therefore you should not create both DEFs in the same directory, since both DEFs have the same design name for the same block.

## Pin-based Memory Modeling

---

Pin-based gds2rh -m/gds2def -m commands generate LEF, DEF, and *\*.pratio* files. The *\*.pratio* files specify weighting factors for current distribution among power/ground pins in the memory. You must use the GDS\_CELLS GSR keyword to import gds2def -m/ gds2rh -m cells, so this change is transparent.

The ‘gds2def -m’ utility traces the power/ground network down to the contact layer. The ‘gds2rh -m’ utility traces the power/ground network only up to metal1 layer. Power and ground pins are defined at locations where transistors’ drains/sources are connected to the grid. Pins are identified only outside of core regions whose boundaries can be either automatically detected (by default) or defined by the user.

Gds2rh/gds2def distributes the total current among all these pins based on device sizes (if a Spice netlist is provided) or the area of contact layer shapes. During simulation, the current is sourced or sunk at these pin locations, and the distribution of the current at each pin recorded in the *\*.pratio* file, which is used by RedHawk in simulation. The pin-based modeling makes simulation current distribution more accurate and eliminates false violations on tie-high/tie-low wires. It also is more efficient in terms of memory use and runtime.

The program creates corresponding DEF, LEF and PRATIO files named after the cell, such as *<cellname>.def*, *<cellname>.lef*, *<cellname>.pratio*. A current distribution that is more characteristic to the cell can be obtained from this process. If used with alternatives such as a SPICE netlist with the X,Y locations of the transistors, the current sinks can be appropriately scaled. The gds2def -m/gds2rh -m utility can handle 45-degree geometries in the GDSII. Other non-Manhattan shapes are not modeled. The 45-degree geometries in DEF version 5.6 are also supported.

## Creating the gds2rh -m/ gds2def -m Characterization Configuration File

---

There are four sections in the gds2rh -m/gds2def -m characterization configuration file.

- Layermap file: specifies the map file that contains the information mapping the GDSII layer numbers to the corresponding layers defined in the LEF/DEF.

The GSS2DEF layermap file must contain a line specifying the contact (diffusion) layer indexes, where 'contact' is a predefined keyword.

### Example:

```
#Layer Type Numbers Text-layer-Number
#-----
- contact 3;4
M1 m 30;31;32;33;34;37;38;39;150;158;35;36
...
```

**Note:** If contact information is not found in a layermap file, GDS2RH/GDS2DEF creates pins on M1, where vias between M1 and M2 are located.

**Note:** To fully utilize the precision of contact-driven pin placement, extraction must start from the lowest metal layer, M1.

- GDSII file: specifies the layout of the memory. The power and ground nets must be labeled in text in GDSII.
- SPICE file: specifies the layout extracted netlist. The netlist must contain placement information for each transistor and instance. The supported formats are Mentor Calibre and Synopsys Hercules.
- LEF file: specifies the description of the memory. The LEF is required to extract the pin information for the memory.

## **gds2rh -m/ gds2def -m Characterization Configuration File Keywords**

In addition to the basic configuration file keywords described in the previous GDS2RH/GDS2DEF section, this section describes the additional keywords used in the characterization configuration file for memories.

### **AUTOGROUP**

To reduce the number of pins, this keyword invokes grouping of vias for each pin based on the specified grid square size, and distributes pratio values to the vias/pins. RedHawk considers the lower metal layer geometry and also the via in each square together, and uses the median via geometry to choose the pin geometry, then assigns the pratio to the created pin. If the grid size is set to 0, no auto grouping of vias is performed. Default grid size: 5  $\mu\text{m}$ .

```
AUTOGROUP <grid_size_um>
```

### **CHECK\_MOS\_SUBCKT**

For instances with names of the type "X<M\_name>", when this keyword is set to 1, 'gds2def -m/gds2rh -m' checks the name of the subckt model to see if it is one of the MOS model names defined in the keywords NMOS\_MODEL\_NAME or PMOS\_MODEL\_NAME. If it is and has four nodes it is treated as the MOS element, disregarding the "X" in the name. Default: 0.

```
CHECK_MOS_SUBCKT [0 | 1]
```

### **CORE\_EXTRACTION\_STARTING\_LAYER**

The metal layer name for starting power/ground nets extraction for the memory\_bit\_cell core region. By default the metal layer used is two layers higher than the metal layer

specified in EXTRACTION\_STARTING\_LAYER. Optional. Default: Extraction\_Start\_Layer + 2.

```
CORE_EXTRACTION_STARTING_LAYER <layer_name>
```

### **GENERATE\_DECAP**

Determines whether decap cells are generated in the output DEF and .LIB file. Default is 1 (generate decap).

```
GENERATE_DECAP [0 | 1]
```

### **LIMIT\_CURRENT\_SOURCE\_HOOKUP**

Specifies whether or not the current source is connected to *only* the lowest layer of metal (as defined by either the layer map file or the Extraction\_Start\_Layer keyword. Default is 0 (off).

```
LIMIT_CURRENT_SOURCE_HOOKUP [0 | 1]
```

### **MEMORY\_CELL**

The name of the memory cell sub-circuit name. Optional. Default: Off.

```
MEMORY_CELL [auto_detect | Off | <memory_cell_name>]
```

### **MEMORY\_BIT\_CELL**

The name of the memory bit cell in the core array. Optional. Default: auto\_detect if 'gds2rh-m' command is used. Default is Off without -m option.

```
MEMORY_BIT_CELL [auto_detect | Off |
<memory_bit_cell_name>]
```

### **MEMORY\_BIT\_CELL\_SANITY\_CHECK**

When set, insures that small via cells are not incorrectly identified and set as memory bit cells when using the auto-detection algorithm. Default: 0.

```
MEMORY_BIT_CELL_SANITY_CHECK [0 | 1]
```

### **MEMORY\_CORE\_REGIONS**

Specifies the file describing the bounding box of memory bit areas, on each line as <x1><y1><x2><y2> (microns), that have virtually no power; the power is distributed to the rest of the memory. You can use any layout viewing tool to define the memory bit rectangle area. Decap pins are created inside core regions. This keyword overrides either MEMORY\_BIT\_CELL or MEMORY\_CELL specifications.

```
MEMORY_CORE_REGIONS <memory_coordinate_filename>
```

### **NMOS\_MODEL\_NAME**

The name(s) of the NMOS transistors in the SPICE\_NETLIST. Required only if SPICE\_NETLIST is used. Accepts wild cards for pattern matching model names.

```
NMOS_MODEL_NAME <nmos_model_name>
```

### **OUTPUT\_DETAILED\_DEF\_PIN**

The DEF output produced by GDS2RH/GDS2DEF does not have physical geometries for the power and ground pins. By turning on OUTPUT\_DETAILED\_DEF\_PIN, the physical geometries of power and ground pins in the LEF file are included in the DEF file. The size

and location of these pins are obtained from the user-provided LEF file of the macro cell.  
Default: 1.

```
OUTPUT_DETAILED_DEF_PIN [0 | 1]
```

### **PMOS\_MODEL\_NAME**

The name(s) of the PMOS transistors in the SPICE\_NETLIST. Required only if SPICE\_NETLIST is used. Accepts wild cards for pattern matching model names.

```
PMOS_MODEL_NAME <pmos_model_name>
```

### **SPICE\_NETLIST**

The layout extracted SPICE netlist with placement information. Optional

```
SPICE_NETLIST <Spice_netlist_filename>
```

### **SPICE\_XY\_SCALE**

Used to scale the X and Y coordinates in SPICE\_NETLIST. The default value is 1 (see note). Required only if SPICE\_NETLIST is used.

```
SPICE_XY_SCALE <scaling_number>
```

---

**NOTE:** If the X and Y coordinates in the SPICE netlist are in microns, set SPICE\_XY\_SCALE to 1000. The units in Hercules netlists are typically specified in microns, so SPICE\_XY\_SCALE is set to 1000. The units in Calibre netlists is in microns \*1000, so SPICE\_XY\_SCALE is set to the default value of 1.

---

The characterization configuration file used by gds2def -m/gds2rh -m varies based on whether the SPICE netlist with X,Y locations are available or not. The following describes the use of keywords for both scenarios.

### **WORD\_LINE\_DIMENSION**

The width of the word line in the memory. Required only if SPICE\_NETLIST is used.

```
WORD_LINE_DIMENSION <word_line_width>
```

### **Spice Netlist with X,Y Locations**

Use a defined SPICE netlist containing the X,Y locations, as follows:

```
SPICE_NETLIST my_cell.spice
```

An example SPICE netlist is shown below:

```
.SUBCKT my_cell 1 2 3 4 5 6 11
 ** N=14 EP=7 IP=53 FDC=8
 M0 3 14 2 3 nlv L=1.3e-07 W=1e-06 $X=790 $Y=5490
 M1 12 4 3 3 nlv L=1.3e-07 W=1e-06 $X=1360 $Y=5490
 M2 13 5 12 3 nlv L=1.3e-07 W=1e-06 $X=1930 $Y=5490
 M3 14 6 13 3 nlv L=1.3e-07 W=1e-06 $X=2500 $Y=5490
 M4 1 14 2 11 plv L=1.3e-07 W=1e-06 $X=790 $Y=1590
 M5 14 4 1 11 plv L=1.3e-07 W=1e-06 $X=1360 $Y=1590
 M6 1 5 14 11 plv L=1.3e-07 W=1e-06 $X=1930 $Y=1590
 M7 14 6 1 11 plv L=1.3e-07 W=1e-06 $X=2500 $Y=1590
.ENDS
```

Use SPICE\_XY\_SCALE to set the scale based on the units used for the X and Y coordinates in the SPICE netlist.

Define the NMOS, PMOS, MEMORY\_BIT\_CELL, MEMORY\_CELL, and WORD\_LINE\_DIMENSION, as follows:

```
NMOS_MODEL_NAME {
 nlv
}
PMOS_MODEL_NAME {
 plv
}
MEMORY_BIT_CELL <bit_cell_name>
MEMORY_CELL <bit_cell_subckt_name>
WORD_LINE_DIMENSION 128
```

## Spice Netlist without X,Y Locations

Use the defined MEMORY\_CORE\_REGIONS to specify a bounding box that avoids placing any current sinks, as follows:

```
MEMORY_CORE_REGIONS my_cell.core
```

where the file *my\_cell.core* contains the bounding boxes for the core regions in the following:

```
10 10 50 50
10 60 50 100
```

Each line in the file represents a rectangular region with the lower left and upper right corner X,Y coordinates specified in microns.

An example of a gds2def -m/gds2rh -m characterization configuration file is shown below:

```
#INPUT FILES
TOP_CELL sram1024x8
GDS_FILE sram1024x8.gds2
GDS_MAP_FILE ram.map
SPICE_NETLIST sram1024x8.sp
LEF_FILE sram1024x8.lef

Cell specific information
VDD_NETS {
 VDD
 VDD:P
}
GND_NETS {
 VSS
 VSS:G
}
NMOS_MODEL_NAME {
 N
}
PMOS_MODEL_NAME {
 P
}
WORD_LINE_DIMENSION 1024
MEMORY_CELL sram1024x8_corecell
```

sram1024x8.sp has units of microns in the x,y declarations, so set scale to 1000:

```
SPICE_XY_SCALE 1000
```

If the Spice netlist has no X,Y locations, use the following option:

```
MEMORY_CORE_REGIONS sram1024x8_memory_bit_X_Y_coordinate_file
#OUTPUT
OUTPUT_DIRECTORY mem_dir
```

The GDSII map file, *<gds\_filename>.map* is used to map GDSII layers to layer names. The syntax of the layer map file is described in [section "Layer Map Definition Keywords", page E-887](#).

More keywords, related to switch cells.

```
LEF_FILES {
 <IP LEF file>
 <switch_cell_1>_adsgds.lef
 <switch_cell_2>_adsgds.lef
 ...
}
```

Defines input block and switch LEF files provided by user or generated from previous step

```
SWITCH_CELLS {
 SWITCH_CELLS {
 <sw1 cellname> <out_pin_name1>_ADS <in_pin_name1>_ADS
 <sw2 cellname> <out_pin_name2>_ADS <in_pin_name2>_ADS
 }
}
```

Maps cellnames to output\_pin\_names and input\_pin\_names, based on pin names in the switch LEF files, and generates a file called *<macro\_name>.swinfo*, which is used in the follow-on step to extract the switch subcircuit for characterization.

```
VP_PAIRS
 VP_PAIRS {
 <vdd1_int> <vdd1_ext> <vdd2_int> <vdd2_ext>
 }

```

Maps internal\_net\_name to external\_net\_name.

## Defining Memories in RedHawk

---

Outputs of the gds2rh -m/gds2def -m utility are written to the directory defined using the '-out\_dir' option, or using the OUTPUT\_DIRECTORY keyword in the configuration file. The memory characterization output files are described as follows:

- the LEF file, *<top\_cell>\_adsgds.lef*, contains the cell abstractions (such as sense amps and decoding logic) inside the memory
- the DEF file, *<top\_cell>\_adsgds.def*, contains the placement of the cell and the power and ground routing geometries
- the Synopsys LIB file, *<top\_cell>\_adsgds.pratio*, contains the relative power dissipation for each of the cells in the memory

where *<top\_cell>* is the memory top cell name defined by the TOP\_CELL keyword in the configuration file. In addition, three log files, *gds.log*, *gds.warn* and *gds.err*, are created.

Before running **RedHawk**, you must make sure that the memories and the path are defined using the GDS\_CELLS keyword in the GSR file.

## Running gds2rh -m and gds2def -m

---

The gds2rh/gds2def -m utility runs memory characterization for the memory specified in the characterization configuration file, using the following syntax:

```
[gds2rh | gds2def] -m <memory_config_file> ?<GDSII_filename>?
```

```
-log <log_dir_name> -out_dir <dir_name>
```

where

<memory\_config\_file> : specifies the memory configuration file

<GDSII\_filename> : specifies the GDS filename

-log <log\_dir\_name>: specifies the directory into which all log files are written

-out\_dir <dir\_name>: specifies the directory into which all output files are written.

Note that this takes precedence over the OUTPUT\_DIRECTORY keyword.

## pt2timing

---

RedHawk's PrimeTime TCL interface pt2timing can be used to obtain timing information such as slew (transition times) and timing windows.

Note: see [Chapter 19, "Timing File Creation Using Apache Timing Engine \(ATE\)"](#) for an introduction to STA file creation and an improved method of generating the STA output file.

Timing information is used by RedHawk to improve the accuracy of both static IR and Vectorless Dynamic voltage drop analysis. For static IR drop analysis, the slew information is used to calculate the average power. In addition, for dynamic voltage drop analysis, the timing window information is used to calculate the peak power and voltage drop.

The following steps are performed in the PrimeTime (Synopsys) environment:

1. Invoke PrimeTime.

```
pt_shell
```

2. Read in the design.

If a fully annotated PrimeTime database is available, use the `read_db` command:

```
pt_shell> read_db <path>/<design_db_name>
```

If a fully-annotated database is not available, then the Verilog netlist and library must be read using the `read_verilog` command:

```
pt_shell> read_verilog <path>/<verilog_netlist>
```

3. Link the design.

Ensure that all link paths are set correctly before linking the design.

```
pt_shell> link_design
```

4. Read in the parasitic information.

The parasitic information yields accurate timing information in the design:

```
pt_shell> read_parasitics <path>/<parasitic_netlist>
```

5. Set up case analysis and constraints.

The appropriate case analysis and constraints must be set up prior to calculating timing information. Case analysis is required to get the correct operating conditions for clock gating or muxing. Source the appropriate PrimeTime constraint files for your design.

Note that STA tools such as PrimeTime use BC/WC analysis as default, where the worst-case corner conditions are used for the Setup time check and the best-case

corner conditions are used for the Hold time check on a clock path. If the worst pin timing is used for min\_arrival and the best pin timing is used for max\_arrival, then the min\_arrival can be greater than max\_arrival. To avoid this situation, make sure that the following conditions are met prior to running PrimeTime timing analysis:

- the SPEF file is generated for worst-case corner conditions
  - the .lib and .db files contain worst-case corner conditions
  - For PrimeTime script, use set\_analysis\_type single and set\_operating\_condition or set\_min\_library to be 'worst'.
  - Use a propagated clock.
6. After loading the design in Primetime, source the *pt2timing.tcl* script. The script extracts the slew (transition times) and timing window information. The default units for these values are nanoseconds (e-9). Note that if ILMs (Interface Logic Models) are used for the full-chip timing, then the ILM must be disabled prior to running the *pt2timing.tcl* script.
7. Write out the timing file using the command 'getSTA'. A typical PT command file for generating the STA file looks as follows:

```
set search_path ". ./dbs/"
set link_path "* lib1.db"
read_verilog generic.v
current_design generic
read_parasitics generic.spef
read_sdc generic.sdc
update_timing
source <path>/pt2timing.tcl
getSTA *
```

Note that by default no explicit timing updates are performed prior to timing window generation, which can reduce runtime by up to 25%. A full timing update can be invoked prior to timing window generation by using the command 'getSTA \* -full'. Incremental timing updates can be invoked prior to timing window generation by using the command 'getSTA \* -inc'.

8. Primetime exits after generating the timing file. The output file is saved as *<design>.timing* in the working directory.
9. The *pt2timing* script internally sets false paths on all clock domains before generating the timing windows. It has no impact on timing window accuracy, but greatly improves timing file generation runtime. So, after executing getSTA script it is not possible to report timing and slack. If you do not want to exit from Primetime after timing window generation, you can use '-noexit' switch
- ```
getSTA * -noexit
```
10. If you are running *pt2timing* on blocks inside the top level design, you should use the -b switch. This generates timing windows for primary input/output ports also.
- ```
getSTA * -b
```

## sim2iprof

---

The sim2iprof utility uses third-party simulation output files, such as *.fsdb*, *.ta0*, *.tr0-.trX*, *.out*, *.wdb* or *.pwl* format files as inputs to obtain Read/Write/Standy mode data for memories, and generates current profiles in a *<cell>.current* file for RedHawk power analysis. Hence the utility supports NSPice, HSpice, NanoSim, Eldo and HSIM outputs in one or more of the previously noted formats.

You can use one of three sim2iprof configuration file keywords to define memory timing:

- MCF\_FILE - the file `.apache/apache.mcf`, which is supplied by user, defines the clock and data timing. When the MCF\_FILE keyword and this file is specified, the tool automatically parses the `.lib` formula and generates the output.
- SIM\_FUNC - use SIM\_FUNC to describe the Boolean timing equation for READ, WRITE, and STANDBY functions explicitly, together with the clock name. The equations are available in the corresponding `.lib` file,
- SIM\_TIME - you can specify directly capture times for READ, WRITE, STANDBY-H, and STANDBY-L if you have knowledge about the waveforms. You must specify the exact starting times for each Read/ Write/ Standby cycle.

The Vdd current waveform is extracted and converted to RedHawk `<cell>.current` format. To get decap information, you can specify the '`-avm`' option and an avm config file, which runs the AVM utility (see the section "AVM Datasheet Characterization", page 9-271) to obtain decap values, equivalent power circuit resistance and capacitance, and leakage power data, and write the data to the `<cell>.cdev` file. Alternatively, if you have already obtained decap data by running other tools, such as ACE, you can directly enter the information in the config file, and sim2iprof then writes them into the `<cell>.cdev` file.

## Running sim2iprof

---

To run the sim2iprof utility, use the following invocation:

```
sim2iprof <config_file> ? -log <log_file> ?
?-o <current_output_file>? ?-avm <avm_config>?
```

where

- <config\_file> : specifies the name of the configuration file
- log <log\_file> : allows specifying a log file for each SIM2IPROF run
- o <current\_output\_file> : specifies optional output filename. The default name is `<cell>.current` when the keyword SIM\_FILE is used in the config file, and `<cellname>.spiprof` when the keyword CELL is used.
- avm <avm\_config> : specifies optional AVM configuration filename to allow AVM to generate decap data. The default is the log file, `sim2iprof.log`, which contains Information, Warning and Error messages.

Note: The specified simulation result file used as a data source must contain both P/G pin current waveforms and related signal voltage waveform data.

### sim2iprof Configuration File

Required keywords for sim2iprof are either SIM\_FILE or CELL, as well as VDD\_PIN and one of MCF\_FILE, SIM\_FUNC, or SIM\_TIME keywords.

Note: If CUSTOM\_STATE\_FILE is specified in the config file, then the keywords CUSTOM\_STATE\_SIM\_TIME, CUSTOM\_STATE\_SIM\_FUNC, SIM\_TIME, and SIM\_FUNC is ignored.

Among keywords CUSTOM\_STATE\_SIM\_TIME, CUSTOM\_STATE\_SIM\_FUNC, SIM\_TIME, and SIM\_FUNC, if you specify more than one of them, the first one in the file is used, and the rest are ignored.

Note: SIM\_FILE reads Vdd value from simulation result file (that is, the ".vdd" line in the .hout file), which may not reflect the true Vdd value used in the simulation, and can cause one simulation result file to be dropped if the Vdd values are the same among two simulation result files. Therefore, for multiple simulation result files with different Vdd values, using the FILENAME option of the keyword CELL is recommended, since then you can directly set the Vdd values for each result file.

Following is a summary of SIM2IPOROF configuration file keywords and the options used to specify the inputs for current profile generation, in alphabetical order:

**ACCURATE\_MODE [ 0 | 1 ]**

To achieve greater accuracy during characterization of PWL inputs, set this keyword to 1. The default value is 0. However, file size is increased in ACCURATE mode, so the size of the output file for big test cases can be reduced by using the default value.

**ALIGN\_VSS\_CURRENT [ 0 | 1 ]**

Controls the window range for VSS\_PIN. For the default of 1, VSS\_PIN uses the same window as VDD\_PIN. You should set this keyword to 0 when the window range for VSS\_PIN is not the same as the VDD\_PIN window range.

**APL\_RUN\_MODE [ DI | DD ]**

Sets APL run mode. When set to DI, RedHawk interpolates between load values and generate sim2iprof waveforms for load values that are not present in the input file. By default, in Design Dependent mode, there is no interpolation, since actual values are used.

**CDEV\_FILE <filename>**

Specifies the device capacitance filename. SIM2IPROF reads this file and generates a new cdev file *<cellname>.cdev*. If the CDEV\_FILE name specified is *<cellname>.cdev*, a random number is added to the filename, such as *<cellname>\_XXXX.cdev*. If neither CDEV\_FILE nor the CDEV section in CELL is specified in the config file, the *<cellname>.cdev* file is not created, and the extracted leakage data is only printed in the log file. A warning is issued that the leakage data is lost.

**CDEV**

Sim2iprof supports multiple nominal voltage cdev characterization as specified in this keyword for the same multiple nominal voltage corner specified for sim2iprof input files.

**Syntax:**

```
CDEV <vdd_pin1>=<Volt_a1> <vdd_pin2>=<Volt_b1> {
 <vdd_gnd_arcl> {
 C0 = <value>
 C1 = <value>
 R0 = <value>
 R1 = <value>
 leak0 = <value>
 leak1 = <value>
 }
 <vdd_gnd_arc2> {
 ...
 }
}
CDEV <vdd_pin1>=<Volt_a1> <vdd_pin2>=<Volt_b1> {
 ...
 ...
}
```

**CELL <cellname> { <cell\_parameters> }**

For each cell, specifies the result file, Vdd values, transition times, loads, and leakage values. **Either** SIM\_FILE or CELL is required; both cannot be used in the same config file. However, using the CELL keyword is recommended, since it provides more data.

For the CELL keyword, two APL files are created in the current directory:

- <cellname>.spiprof - cell switching current file
- <cellname>.cdev - cell cdev file (if CDEV is specified)

If multiple Vdd values are specified (nominal value and degraded values), the switching waveforms are sorted in descending order of Vdd value.

```
CELL <cellname> {
 FILENAME {
 <sim_result_file_1> <vdd_pin1>=<Volt_a> <vdd_pin2>=<Volt_b> ...
 <sim_result_file_2> ...
 ...
 }
 SLEW {
 <trans_time_sec>
 }
 LOAD {
 <load_cap_F>
 }
 CDEV {
 <vdd_pin> <gnd_pin> {
 C0 = <cap_low_F>
 C1 = <cap_hi_F>
 R0 = <R_low_Ohms>
 R1 = <R_hi_Ohms>
 ? LEAK0 = <leak_low_A>
 LEAK1 = <leak_hi_A> ?
 CK_PULSE
 }
 ...
 }
}
```

where

<sim\_result\_file\_n>: specifies one simulation result file per line, which can be in \*.fsdb, \*.hout, \*.tr0-\*.\*.trX, \*.ta0-\*.\*.taX, wdb, or \*.pwl format. Each file contains simulation results for a different set of Vdd values. At least one <sim\_result\_file> must be specified. The Vdd pin order should be consistent with that specified using the keyword VDD\_PIN. If no Vdd value can be read from simulation results, and no Vdd value is specified, it is assumed to be 1.0 V. Supports separate FSDB files for different states (Read, Write, Standby, ...) for memories.

For a \*.pwl file, the file format for defining pin voltage/current is as follows:

```
<Pin_name> PWL (
 + <time_1> <value_1>
 + <time_2> <value_2>
 + ...
)
```

where <Pin\_name> is the pin name for a signal, and <time\_n> <value\_n> are the associated time and voltage or current values, in seconds and Volts or Amps, respectively. The <Pin\_name> needs to match the pin name specified in the configuration file.

Note also that if current is to be specified for a pin, the pin name **must have an initial “I” to be correctly identified** and have current units.

Example:

```
Ivdd PWL (
+ 0.0 0.0
+ 1.0e-9 1e-3
+ 5.0e-9 1e-3
+ 6.0e-9 1e-6
+ 1.0e-8 1e-6
)
```

Note the use of “+” as a continuation character.

<vdd\_pin1>=<Volt\_a> <vdd\_pin2>=<Volt\_b>: optionally, you can also specify each Vdd domain's nominal voltage value.

SLEW <trans\_time> : input transition time (sec)

LOAD <load\_cap\_F> : output capacitance (Farad)

Note: normally for SIM2IPROF applications, which are capturing switching currents for large macros such as memory and I/Os, there is no need to specify SLEW and LOAD values, since they have little impact on the overall switching current. However, if you want to specify your own sample values for a macro, these two keywords can be used.

CDEV : specifies pwr/gnd arcs, device capacitance, “ESR” equivalent power resistance, and leakage current, which is subtracted from switching current. If CDEV\_FILE, LEAKAGE\_VALUE and the CDEV section in CELL are all specified, CDEV section has higher priority. If the CDEV section are specified, but leakage in the CDEV section is not specified, SIM2IPROF performs extraction from the simulation results and puts that data into the .cdev file. If neither the CDEV section nor CDEV\_FILE is specified in the config file, the <cellname>.cdev file is not created, and the extracted leakage data is only printed in the log file. A warning is issued that the leakage data is lost.

<vdd\_pin> <gnd\_pin> : specifies power/ground arcs, one per line for multiple Vdd/Vss pins. Note that if you specify Vdd pin values, you must do it for *ALL* Vdd domains/pins. A partial pin list generates an error. If no Vdd values are specified, all are taken from simulation. If there are multiple Vdd values (derated Vdd values), add them in new lines within the FILENAME section.

C0 = <cap\_low\_F> : equivalent power circuit capacitance (Farad), cell output low

C1 = <cap\_hi\_F> : equivalent power circuit capacitance (Farad), cell output high

R0 = <R\_low\_Ohms> : equivalent power circuit resistance, cell output low

R1 = <R\_hi\_Ohms> : equivalent power circuit resistance, cell output high

LEAK0 = <leak\_low\_A> : leakage current (Amps), cell output low (optional)

LEAK1 = <leak\_hi\_A> : leakage current (Amps), cell output high (optional)

### **CHECK\_BOOLEAN\_INPUT**

When set, Sim2iprof checks and errors out when a Boolean equation or active input pin is missing in the configuration file for keywords CUSTOM\_STATE\_SIM\_TIME, CUSTOM\_STATE\_SIM\_FUNC and COMPOSITE\_STATE. Default : 0 (Off). The syntax is:

```
CHECK_BOOLEAN_INPUT [0 | 1]
```

### **COMPOSITE\_STATE**

Supports summation and scaling of individual current profiles, which can generate multi-state current profiles for user-defined input/output pin scenarios for clock toggle, data pin toggle, and clock and output toggle. Multi-bit sequential cell handling also can include individual edge-triggered events, where all input/output pins do not have to trigger off the

same event. When sim2iprof detects a composite switching state name defined in the COMPOSITE\_STATE keyword, it calculates the state algebraic expression. The Boolean expression and active input pin are optional. The syntax is:

```
COMPOSITE_STATE {
 <state_name> "<Boolean_eq>" "<active_input_pin>"
 "<state_algebraic_expression>"
 ...
}
```

where

- <state\_name>: specifies the user-defined state name for each composite state
- <Boolean\_eq>: specifies the Boolean expression to be used in VCD mode analysis to determine when the specific state is On (active).
- <active\_input\_pin> : specifies the input pin that switches
- <state\_algebraic\_expression>: defines a linear combination of multi-state currents using Boolean definitions. Note that a composite state cannot be used to define another composite state.

Example 1:

```
COMPOSITE_STATE {
 OUT1 "a&b" "clk" "10 * RISE_OUT"
 OUT2 "" "" "RISE_OUT * 3"
}
```

Example 2:

```
COMPOSITE_STATE {
 DATA_IN_TOGGLE_POS "data1" "data1" "DATA_IN_RISE"
 DATA_OUT_TOGGLE_POS "data_out_pin" "data_out_pin" "REF_WAVEFORM1 - CLOCK"
 CLOCK_TRIG_POS_VCD "(en1|en2)&clk" "clk" "CLOCK_RISE_TOGGLE"
 CLOCK_TRIG_NEG_VCD "(en1|en2)&!clk" "!clk" "CLOCK_FALL_TOGGLE"
}
```

Note that all base states specified in COMPOSITE\_STATE must be defined in either CUSTOM\_STATE\_SIM\_TIME or CUSTOM\_STATE\_SIM\_FUNC, such as:

```
CUSTOM_STATE_SIM_TIME {
 # Base state definitions
 # <state_name> <Boolean_express> <active_input_pin> <start_time>
 <end_time>
 CLOCK_TRIGGER "" clk 5.0e-9 6.0e-9
 INPUT_TOGGLE "" "" 8.0e-9 9.0e-9
 OUTPUT_TOGGLE "" clk 5.0e-9 6.0e-9
}
COMPOSITE_STATE {
 # composite switching state definitions
 # <state_name> <Boolean_eq> <active_input_pin>
 <state_algebraic_expression>
 OUTPUT_STATES1 "" "" "1*OUTPUT_TOGGLE"
 OUTPUT_STATES2 "a & b" clk "2*OUTPUT_TOGGLE -
 1*CLOCK_TRIGGER"
 OUTPUT_STATES3 "" "" "32*OUTPUT_TOGGLE -
 31*CLOCK_TRIGGER"
 STANDBY_STATE1 "c & d" clk "1*INPUT_TOGGLE"
```

```
}
```

Also, a composite state cannot be used by another composite state. For example, specifying the following is **NOT** allowed:

```
COMPOSITE_STATE {
 OUTPUT_STATES1 " " " " "1*OUTPUT_TOGGLE"
 OUTPUT_STATES2 " " " " "2*OUTPUT_TOGGLE"
}
```

### CURRENT\_INVERT\_FLAG [ 0 | 1 ]

Controls inversion of the current waveform by SIM2IPROF, depending on the keyword setting and total charge and the peak of the waveform. The keyword usage is as follows:

- When the keyword is not specified, the default behavior is SIM2IPROF inverts the current if the total charge and the peak are both negative.
- When 'CURRENT\_INVERT\_FLAG 0' : no inverting occurs. The output waveform is consistent with the FSDB data.
- When 'CURRENT\_INVERT\_FLAG 1' : all current waveforms are inverted.

### CURRENT\_SCALING\_FACTOR <l\_factor>

Multiplies all calculated output current values at every time point in `<cell>.current` by the `<l_factor>` (default: 1.0).

Where different current scaling is needed for each custom state, use a separate configuration file for each custom state and a different CURRENT\_SCALING\_FACTOR in each config file. For example, on the command line specify:

```
sim2iprof state1.config state2.config
```

Then in the `state1.config` file, custom state 1 and the associated current scaling factor could be defined as follows:

```
CELL SC9_CB_64X42 {
 ...
 CUSTOM_STATE_SIM_FUNC {
 M0 clk&cam_x_alloc_en_n clk
 }
 CURRENT_SCALING_FACTOR 2
 ...
}
```

And in the `state2.config` file, custom state 2 and the associated current scaling factor could be defined as follows:

```
CELL SC9_CB_64X42 {
 ...
 CUSTOM_STATE_SIM_FUNC {
 M1 clk&en_n clk
 }
 CURRENT_SCALING_FACTOR 3
 ...
}
```

In this way the configuration files provide individual current scaling for each custom state.

### CUSTOM\_STATE\_FILE { <filename> }

Specifies a file containing cell state and timing information, such as the Boolean timing equation for each custom state. Note that multi-state mode supports only memory/IP

cells, and not combination and sequential cells. Each line is a state definition (evaluated in order). The syntax of the state file is as follows:

```
CELL <mem_cell> {
 <state_name> "<Boolean_eq>" "<active_pin/clock_pin>" ?<output_pin>"?
 ...
}
```

where

<mem\_cell>: specifies the name of memory cell

<state\_name> : required user state name composed of a string of letters, numbers, and/or '\_' used to define state of a multi-state instance in the GSC. Note that there are two **required** reserved state names, as follows:

STANDBY\_TRIG: captures i(Vdd) and i(Vss) when only the clock pin is switching in standby mode, from low to high for positive-triggered cells and from high to low for negative-triggered cells.

STANDBY\_NTRIG: captures i(Vdd) and i(Vss) when only the clock pin is switching in standby mode, from high to low for positive-triggered cells and from low to high for negative-triggered cells.

<Boolean\_eq> : required Boolean expression defining the trigger condition for the cell. For example, “A&B&!C” refers to A=1, B=1, and C=0. A state definition is a string composed of pin names and Boolean operators: ! (negation), & (and), and space (optional). Each pin name should appear only once to avoid ambiguity. Also, the state definition string must be enclosed in double-quotes. For sequential cells, clock pin(s) also need to be included in the state definition string. “CLK” refers to positive edge trigger and “!CLK” refers to negative edge trigger.

**Note:** The Boolean equation must be conditioned by the trigger expression, with a character limit of 1024 characters (the trigger expression can have non-clock inputs).

<active\_pin/clock\_pin>: specifies required pin that is switching for APL characterization. For example, for a defined state of “A&B&!C” and active pin “B”, specifies that A=1, B switches from 0 to 1, and C=0. Multiple active pins are allowed if the pins switch simultaneously, and each should be in double quotes. If there is only one active pin, double quotes are not required. For positive-triggered cells, only the clock pin is required. For negative-triggered cells, an active pin ‘!’ is needed (such as '!clk').

<output\_pin>: specifies the output pin

---

**NOTE:** When creating multiple-state models, you must capture current profiles for reserved states ‘standby\_trig’ and ‘standby\_ntrig’. During dynamic analysis (vectorless/VCD), if a memory *does not switch* in one of the other custom states (read/write/compare, and so on), then they are assumed to be in standby mode and a current profile corresponding to a standby state is used. If standby states are not captured, RedHawk treats the instances not switching in other custom states as decaps and they do not appear in the output *adsRpt/\*dvd\** reports and in instance DvD colormaps.

---

### CUSTOM\_STATE\_SIM\_FUNC

```
CUSTOM_STATE_SIM_FUNC {
 <state_name> <Boolean_eq> <clk_name> ?<alias_state_name>?
 ...
}
```

Specifies custom functional states for simulation, where

- <state\_name> : user name for custom state
- <Boolean\_eq> : Boolean equation defining switching
- <clk\_name>: specifies the clock name
- <alias\_state\_name>: specifies an acceptable alternative Boolean equation and current profile if there are no current profiles in the simulation output data for the particular state.

### **CUSTOM\_STATE\_SIM\_TIME**

```
CUSTOM_STATE_SIM_TIME {
 <state_name> <Boolean_eq> <active_input_pin> <start_time>
 <end_time> <iscale=<value>> <outpin="active_output_pin">
 ...
}
```

Supports custom multiple-state Boolean expressions in LIB files. Each line is a state definition, where

<state\_name>: user name for custom state. Note that there are two reserved state names, as follows:

STANDBY\_TRIG: captures i(Vdd) and i(vss) when only the clock pin is switching in standby mode, from low to high for positive-triggered cells and from high to low for negative-triggered cells.

STANDBY\_NTRIG: captures i(Vdd) and i(Vss) when only the clock pin is switching in standby mode, from high to low for positive-triggered cells and from low to high for negative-triggered cells.

<Boolean\_eq> and <active\_input\_pin> : used to report in cell current file only.

<start\_time>: start of capture times for the state.

<end\_time> : optional end time for the state. If specified, the duration is computed as (<end\_time> - <start\_time>), or else the value specified in the DURATION keyword is used.

iscale=<value> : scales the current by the specified value

outpin="active\_output\_pin" : specifies the names of the active output pins and displays them in the output file

For example:

```
CUSTOM_STATE_SIM_TIME {
 M0 cena&cenb clka 22.4e-9 25.2e-9
 M1 cena clka 2.4e-9
 M2 !cena clka 7.4e-9
 M3 cena&cenc clka 12.4e-9
 M4 cenb clka 17.4e-9
}
DURATION 5e-9
```

### **CYCLE\_SELECTION [ peak | avg ]**

Specifies cycle selection criteria when there are multiple cycles that meet SIM\_FUNC formula definition. ‘peak’ selects the cycle with the highest peak current value and ‘avg’ selects cycle that is the summation of current values within the cycle. Optional - default =peak.

**CYCLE\_SKIP <cycle0> < cycle1> ...**

Specifies skipping extraction of undesired cycles, such as an initial spike cycle. For example, the first clock cycle may include a large amount of noise that is not to be considered. Cycle count starts from 0. Optional.

**DELAY\_TIME <delay\_time\_in\_sec>**

Specifies delay after the clock triggering edge to start capture period. Optional; default =0.

**DURATION <time\_sec>**

All transactions (READ/WRITE/STANDBY) are normally defined to end at the point at which the current drops below 10% of the maximum value. If this keyword is specified, sim2iprof uses the DURATION value rather than the 10% point to determine the transaction end times.

**DURATION\_PEAK\_FLAG [ 0 | 1 ]**

When set to 1, captures peak current in duration width beginning from the trigger edge when the duration is less than the cycle width of clock pin

**IGNORE\_INPUT\_CDEV\_ERROR**

Overrides different cell names in the *mcap* and *cdev* files. If set to 1 and cell names are different in the *mcap* and *cdev* files, cell names in *mcap* are replaced by *cdev* names in the configuration file.

**IGNORE\_UNRECOGNIZED\_KEYWORD [ 0 | 1 ]**

Controls the response to errors found in checking syntax and spelling of configuration file keywords. By default (1) unrecognized keywords are identified in a Warning message, but processing continues. If IGNORE\_UNRECOGNIZED\_KEYWORD is set to 0, SIM2IPROF errors out for any unrecognized keyword. Be sure to spell the keyword correctly when using it to insure that it is in effect.

**IPROF\_SAMPLING\_MODE**

```
IPROF_SAMPLING_MODE
[<number_samples> | Default (50) |
 Extended (100) |
 Accurate (400)]
```

Specifies the number of sampling points for generating APL current profiles. More sampling points provide finer waveform resolution in the final <cell>.current file.

**LEAKAGE [1 | 0]**

Specifies whether leakage currents are subtracted from switching current values or all leakage currents are set to zero. If 'LEAKAGE 0' is specified, sim2iprof turns off leakage subtraction and sets all pin leakage currents to 0. Default value is '1': leakage currents are considered.

**LEAKAGE\_VALUE**

```
LEAKAGE_VALUE {
 <vdd_pin> <gnd_pin> {
 LEAKAGE0 = <val>
 LEAKAGE1 = <val>
 }
}
```

Specifies pin names <vdd\_pin> and <gnd\_pin>, and leakage values in Amps for output low and output high conditions LEAKAGE0 and LEAKAGE1, respectively. If both CDEV\_FILE and LEAKAGE\_VALUE are given, SIM2IPROF uses values from LEAKAGE\_VALUE when creating <cellname>.cdev. If only LEAKAGE\_VALUE is

specified, this leakage is subtracted from switching current. A warning is issued that the leakage data is lost. Supports multiple nominal voltage cdev characterization.

---

**NOTE:** APL switching current data (the `<cell>.current` file) should not contain any leakage data, in order to avoid double reporting of leakage in both `<cell>.current` and `<cell>.cdev`. If you specify leakage in the config file in the CDEV section, that leakage data is written into the `<cell>.cdev` file. If you do not specify leakage in the CDEV section in the config file, sim2iprof performs extraction from the simulation results and puts that data into the `<cell>.cdev` file. If the CDEV section is not specified in the config file, the `<cell>.cdev` file is not created, and the extracted leakage data is only printed in the log file. In all cases, leakage data is removed from the `<cell>.current` file.

---

#### **MCF\_FILE <mcf\_file>**

Specifies the READ/WRITE/STANDBY Boolean formulas for waveform capture. The MCF file `adsPower/apache.mcf` is generated when performing power calculation in RedHawk. If the MCF\_FILE keyword is used, SIM\_FUNC and SIM\_TIME cannot be used. You can create an MCF file using the following format:

```
cell <cell_name> <number of clock pins> <clock_pin_name1> ...
C01 "<C01_Boolean_function>"
C10 "<C10_Boolean_function>"
C00 "<C00_Boolean_function>"
C11 "<C11_Boolean_function>"
```

Example:

```
Cell memory64x8 1 clk
C01 "(!write & read & clk)"
C10 "(write & !read & clk)"
C00 "(!write & !read & clk)"
C11 "(!write & !read & !clk)"
```

Note: The states must be listed in the order as shown (C01, C10, C00, C11). Also, C00 and C11 are typically reserved for non-active states, such as standby for memories.

#### **MINTIMESTEP**

Sim2iprof can handle corner cases with large signal waveforms and simulation times greater than 1 us, and also small time steps on the order of 0.1 ps, while providing accurate results. Use this keyword to configure the desired time step or resolution. Data points for which the time step is less than the specified value are skipped.

Syntax:

```
MINTIMESTEP <time_in_seconds>
```

#### **NODE\_PLOT <node\_name1> <node\_name2> ...**

Specifies signal names to extract X,Y coordinates, which can be viewed by the waveform visualization tool (i.e., xgraph). Each specified node generates a `node_name.xy` file. Note: Vdd current value is stored as a negative value. Scaling factors must be properly specified when viewing voltage and current waveforms together. Optional.

#### **PIN\_MAP**

```
PIN_MAP {<pin_name_.lib file> <pin_name_result file>}
```

Maps the pin names if those used in SIM\_FUNC (from `.lib`) are not the same as those used in the simulation result file. Optional.

**PROCESS [ SS | TT | FF ]**

Specifies a process type, where SS is slow, TT is typical, and FF is fast. Required.

**RATIO <percentage>**

Specifies a ratio of the peak value for determining the end of the capture window. The last occurrence of the ratio value in the event is considered the end of the capture. Can be used to filter out noise. Optional - default =0.1 ( 10%)

**SCALE\_FACTORS**

Specifies TSF/CSF per Power/GND pin and per mode for better accuracy at APL model generation for compiler through interpolation.

Current Scale Factor (CSF) = target peak / ref peak

Time Scale Factor (TSF) = (target area / ref area) \* 1/CSF)

Syntax in sim2iprof config file-

```
SCALE_FACTORS {
 CELL <old_cell_name> <new_cell_name> {
 <state1> <pinname1> <tsf1> <csf1>
 <state1> <pinname1> <tsf2> <csf2>
 <state1> <pinname1> <tsf1> <csf1>
 <state2> <pinname1> <tsf2> <csf2>

 }
}
```

**SCALE\_LEAKAGE [0 |1]**

If 'CURRENT\_SCALING\_FACTOR' is specified and SCALE\_LEAKAGE is set to 1 (default), both switching current and leakage current are scaled. If 'SCALE\_LEAKAGE 0' is set, then the leakage current is not scaled.

**SETUP\_TIME <setup\_time\_in\_sec>**

Specifies the setup time before the triggering edge of the clock to start capture period. Optional - default =0.

**SIM\_FILE**

```
SIM_FILE { <cellname> <filename> ? <trans_time_ps>?
?<load_fF>?
... }
```

Specifies the list of cell and file names, optional rise and fall transition times in ps, and unit load capacitance in fF units.

*Either* SIM\_FILE or CELL is required; both cannot be used in the same config file.

**SIM\_FUNC**

```
SIM_FUNC {
 READ <Boolean_eq> <clk_Name>
 WRITE <Boolean_eq> <clk_Name>
 STANDBY <Boolean_eq> <clk_Name>
}
```

Specifies the Boolean timing equation for start of READ/ WRITE/ STANDBY signals from .lib file, and the associated clock name. The time at which signals fall to 10% of peak value is “end” time. If SIM\_FUNC is used, then MCF\_FILE and SIM\_TIME keywords cannot be used. Note “!” identifies a negative value.

**SIM\_TIME**

```
 SIM_TIME {
 READ <Start_time>
 WRITE <Start_time>
 STANDBY-H <Start_time>
 STANDBY-L <Start_time>
 }
```

Specifies the starting times for READ/ WRITE/ STANDBY-H / STANDBY-L signals, in seconds. Can be used instead of SIM\_FUNC or MCF\_FILE.

Note: For I/O blocks, because of the difficulty in specifying STANDBY-H and STANDBY-L values from Spice simulation, if the Standby values are not specified, sim2iprof assigns zero current by default for those states.

**SLEW\_THRESHOLD <low\_V> <hi\_V>**

Specifies low and high fraction of Vdd for determining slew value.

**SPICE2LEF\_PIN\_MAPPING**

```
 SPICE2LEF_PIN_MAPPING {
 <subckt_pin_name> <LEF_pin_name>
 ...
 }
```

Maps pin names between simulation result files and LEF data. When writing pin names to *<cell>.current* and *<cell>.cdev* files, LEF pin names are used instead of Spice pin names.

**TEMPERATURE <degree\_C>**

Specifies operating temp for current profile generation.

**VDD\_PIN <vdd\_pin\_name> ...**

Specifies Vdd pin names that have a current waveform, including multiple power domain pins. **Required**.

**VSS\_PIN <vss\_pin\_name> ...**

Specifies Vss pin names that have a current waveform, including multiple power domain pins. **Required**.

**WORST\_IPEAK**

By default sim2iprof picks the time window with the worst peak current value if multiple time windows satisfy the Boolean expression. When you specify 'WORST\_IPEAK 0' sim2iprof picks the *first* time window that satisfies the Boolean expression.

**ERROR\_MISSING\_STANDBY\_STATE [ 0|1 ]**

When set, RedHawk errors out if standby states are missed in the configuration file, as follows:

- when set to 0 (default), issues a warning and proceeds on missed standby states
- when set to 1, errors out on missed standby states

## Configuration File Example

The following is an example of a sim2iprof configuration file using the 'CELL' keyword. The file includes keywords typically used to automatically generate optimal results. Other optional keywords may be used to tune the results to meet special needs. RedHawk checks whether the specified Vdd values and Vdd pin names for multi-rail macros match.

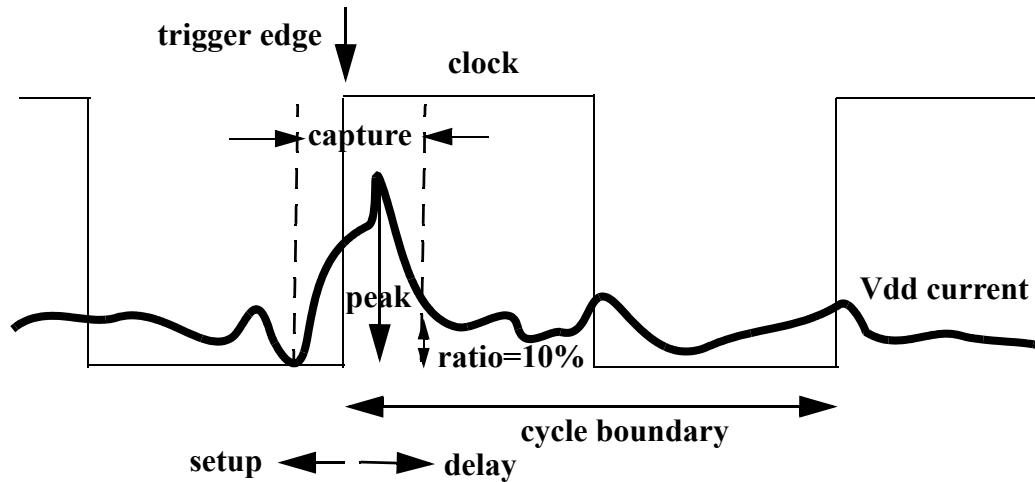
If the pin names in the VDD\_PIN section and FILENAME section do not match, it reports an error.

```
!-- sim2iprof configuration file generated by
sim2iprof_setup.pl on Thu Feb 1 15:23:41 PST 2007

CELL (sram128x32) {
 FILENAME {
 # <filename> [<vdd1>=<vdd1_v2> <vdd2>=<vdd2_v2>...]
 dram.s0.fsdb vdd=1.15 vddo=1.05
 dram.s1.fsdb vdd=1.1 vddo=1.0
 }
 SLEW {
 11ps
 }
 LOAD {
 15ff
 }
 CDEV {
 vpwr vgnd {
 C0 = 1.0p
 C1 = 2.0p
 R0 = 400.21
 R1 = 395
 LEAK0 = 1.0e-6
 LEAK1 = 4.2uA
 }
 ivdd vgnd {
 C0 = 3.0p
 C1 = 5.0p
 R0 = 621
 R1 = 385
 LEAK0 = 2.0e-6
 LEAK1 = 4.2uA
 }
 } // end of CDEV
} // end of cell

SIM_TIME {
 READ 1e-12
 WRITE 1e-12
 STANDBY-H 1e-12
 STANDBY-L 2e-12
}
VDD_PIN vpwr ivdd
VSS_PIN vgnd
DURATION 500e-12
```

Some of the key sim2iprof functions and terminology are illustrated in Figure E-11.



**Figure E-11    Sample sim2iprof waveform and terminology**

## Output

By default, sim2iprof generates a `<cell>.current` file and a `sim2iprof.log` file. The `sim2iprof.log` reports the time window, peak current, area, and leakage for pins, and also the voltage with which the extracted values are associated. A sample output report is given below.

```
Info: Printing 'nb_write' of pin 'vvdd_wb' summary at vvdd_wb=1.21 ...
Info: window range = (196.12, 200.12) ns
Info: peak value = 55769.6 uA
Info: area value = 25.7529 pc
Info: leak value = 0 uA
Info: Printing 'nb_write' of pin 'vvss_wb' summary at vvdd_wb=1.21 ...
Info: window range = (196.12, 200.12) ns
Info: peak value = 91668.8 uA
Info: area value = 38.6493 pc
Info: leak value = 0 uA
```

Check the log file for Error or Warning messages if it does not produce the expected results. RedHawk uses the `<cell>.current` file for further analysis.

## aplreader

The aplreader utility extracts data from the APL waveform files such as `<cell>.current`, `vmemory.current` or `*.cdev`, and displays the information in the terminal window where it is invoked.

### Running aplreader

To run the aplreader utility, use the following Linux/Solaris invocation syntax :

```
aplreader <input_dir/file> [-l <cell_list_file>]
 -timing <apl_file_name>
```

where

`<input_dir/file>` : specifies the input file directory and filename

```
-l <cell_list_file> : specifies an optional cell list file for which only cells in the file
are processed
-timing : prints timing info such as delay/slew/firing in the specified file
```

## aplreader Output

---

### Current Outputs

The aplreader utility writes to the display the following current-related parameters:

- “cell”: name of cell
- “C1”, “C2” - load capacitances in Farads
- domain name and nominal voltage in Volts
- “R” - load resistance in Ohms
- “Slew1”, “Slew2” - input transition times in seconds
- “state” - state name
- “vector” - Boolean definition of vector for the state
- “active input”, “active output” - names of active input and active output pins
- “pin” - power/ground pin name
- “peak” - peak cell current in Amperes
- “area” - charge under the waveform in Coulombs
- “width” - duration of waveform in seconds

The format of the aplreader output for current waveform files, *\*.current*, is as follows:

```

cell : <cellname>
vdd1 = <vdd value 1> ; vdd2 = <vdd value 2>; ... vddN = <vdd value
N> ; C1 = <Cload1 value> ; R = <load resistance> ; C2 = <Cload2
value>; Slew1 = <rising transition time>; Slew2 = <falling
transition time> ;
state = <state name> ; vector = <Boolean_eq> ; active_input =
<active_input name> ; active_output = <active_output name> ;
pin peak area width
<pin name> <peak I - A> <Total charge - C> <Wave duration - S>

(... repeat previous syntax for all Vdd and Vss pins)
(... repeat previous syntax for all toggles)
(... repeat previous syntax for all Vdd values)
(... repeat previous syntax for all cells)

```

### aplreader Current Output Examples

The following is a sample aplreader output for a standard cell APL characterization:

```

cell : NAND2
vdd1 = 1.3915 V ; C1 = 0 F ; R = 0 Ohm ; C2 = 2.80552e-15 F ;
Slew1 = 1.7046e-11 S ; Slew2 = 1.0228e-11 S ;

state = output_rise ; vector = a&b ; active_input = a ; active_output = z ;
pin peak area width
```

---

```
vddfx 9.00e-05 A 5.58e-15 C 1.44e-10 S
vssfx 9.40e-05 A 5.72e-15 C 1.44e-10 S
state = output_fall ; vector = a&b ; active_input = a ; active_output = z ;
pin peak area width
vddfx 8.10e-05 A 1.94e-15 C 9.60e-11 S
vssfx 3.80e-05 A 1.31e-15 C 9.50e-11 S


```

The following is a sample *apreader* output for a memory cell characterized by sim2iprof and avm:

```

cell : memory_a
vdd1 = 1.188 ; vdd2 = 1.08 ; C1 = 3.9e-14 ; R = 180.9 ;
C2 = 1.8e-13 ; Slew1 = 1.2e-10 ; Slew2 = 9.1e-11 ;

state = READ ; vector = ram_en*re ; active_input = clk ;
active_output = N/A ;
pin peak area width
vdd1 4.6e-2 A 3.7e-12 C 1.7e-9 S
vdd2 3.6e-3 A 5.3e-12 C 1.7e-9 S
vss 6.8e-2 A 6.3e-12 C 5.7e-10 S

state = WRITE ; vector = ram_en*we ; active_input = clk ;
active_output = N/A ;
pin peak area width
vdd1 2.3e-2 A 2.3e-12 C 1.7e-9 S
vdd2 2.6e-4 A 2.2e-12 C 1.7e-9 S
vss 4.9e-3 A 9.2e-13 C 6.3e-10 S

state = STANDBY ; vector = ram_en!*re*!we ; active_input = clk ;
active_output = N/A ;
pin peak area width
vdd1 3.5e-5 A 1.7e-13 C 1.7e-9 S
vdd2 2.6e-5 A 4.7e-13 C 1.7e-9 S
vss 4.9e-5 A 9.9e-13 C 6.3e-10 S
##--NOTE: use "output_rise" for C01, "output_fall" for C10,
"output_nochange_input_trigger" for C00,
"output_nochange_input_nontrigger" for C11.

```

## Equivalent Device Capacitance and Resistance Outputs

### **apreader Device C and R Output Examples - Version 10.1 and later**

Sample *apreader* output for equivalent device capacitance and resistance characterization:

```
Temperature = 125.00000 C; State = ADS_DEFAULT_STATE_LOW; vector = i;
active_input = i; active_output = o;
VDD = 1.200000 V; VSS = 0.000000 V;
pin = VDD, esc = 1.67321e-15 F, esr = 630.892 ohm
pin = VSS, esc = 1.67321e-15 F, esr = 630.892 ohm
pin = VDD, leak = 5.16966e-10 A
```

```
pin = VSS, leak = 5.16966e-10 A
Temperature = 125.00000 C; State = ADS_DEFAULT_STATE_HIGH; vector = i;
active_input = i; active_output = o;
VDD = 1.200000 V; VSS = 0.000000 V;
pin = VDD, esc = 1.80674e-15 F, esr = 797.314 ohm
pin = VSS, esc = 1.80674e-15 F, esr = 797.314 ohm
pin = VDD, leak = 4.01521e-10 A
pin = VSS, leak = 4.00058e-10 A
```

## Piecewise Linear Capacitance and Resistance Outputs

The format of apreader output for power-up equivalent capacitance and resistance characterization files, \*.pwcdev, is as follows:

```
Info: cell=<cellname>
Temperature = ; State = ; vector = ; active_input = ;
active_output = ;
vdd = value; vss = value;
arc=<P/G domain>, esc = <capacitance>, esr= <resistance>,
pin= p/g pin; leak = <leakage_current>
(... repeat for all Vdd values up to three. If there are more than
three Vdd values, apreader takes the smallest, middle, and
largest Vdd values)
(... repeat for all states)
(... repeat previous syntax for all cells)
```

## apreader PWL Device C and R Output Example

The following is a sample apreader output for a power-up equivalent capacitance and resistance characterization:

```

Info: cell=test_cell
Temperature = 125.000000 C; State = ADS_DEFAULT_STATE_LOW; vector = A;
active_input = A; active_output = X;
VDD = 0.090000 V; VSS = 0.000000 V;
arc = (VDD VSS), esc = 4.04313e-15 F, esr = 761.765 ohm
pin = VDD, leak = 1.33983e-08 A
VDD = 0.540000 V; VSS = 0.000000 V;
arc = (VDD VSS), esc = 8.11297e-15 F, esr = 1138.18 ohm
pin = VDD, leak = 5.60882e-08 A
VDD = 0.990000 V; VSS = 0.000000 V;
arc = (VDD VSS), esc = 1.01403e-14 F, esr = 161.96 ohm
pin = VDD, leak = 2.91051e-07 A
Temperature = 125.000000 C; State = ADS_DEFAULT_STATE_HIGH; vector = A;
active_input = A; active_output = X;
VDD = 0.090000 V; VSS = 0.000000 V;
arc = (VDD VSS), esc = 4.04353e-15 F, esr = 1370.52 ohm
pin = VDD, leak = 1.33983e-08 A
VDD = 0.540000 V; VSS = 0.000000 V;
arc = (VDD VSS), esc = 1.03437e-14 F, esr = 459.626 ohm
pin = VDD, leak = 6.42139e-08 A
VDD = 0.990000 V; VSS = 0.000000 V;
arc = (VDD VSS), esc = 1.33618e-14 F, esr = 124.924 ohm
```

```
pin = VDD, leak = 1.85495e-07 A

```

## aplcdev2pwc

---

The aplcdev2pwc utility is invoked from a Linux/Solaris command line to check the validity of cdev data, and it then converts the cdev data to pwcdev data. The input cdev file can be any version, but the output pwcdev file can only be a 5v3 version or older.

When converting cdev data to pwcap data, the Cap/Res/Leakage values for nominal VDD value cdev data are scaled with following formulas:

$$\begin{aligned} C_{\text{pwcap}}(\text{VDD}) &= C_{\text{cdev}} * (\text{VDD} / \text{nominal\_VDD})^{\text{cf}}; \\ R_{\text{pwcap}}(\text{VDD}) &= R_{\text{cdev}} * (\text{VDD} / \text{nominal\_VDD})^{\text{rf}}; \\ \text{Leak\_pwcap}(\text{VDD}) &= \text{Leak\_cdev} * (\text{VDD} / \text{nominal\_VDD})^{\text{lf}}; \end{aligned}$$

The syntax of the command is:

```
aplcdev2pwc [-cf <cap_factor>] [-rf <res_factor>] [-lf
<leakage_factor>] [-o <output_pwcdev_file>]
<input_APL_config_file> <input_cdev_file>
```

where

- cf: specifies the capacitance factor (default 0.5)
- rf: specifies the resistance factor (default -0.5)
- lf: specifies the leakage current factor (default 1.5)
- o: specifies the output pwcdev filename

Example invocation:

```
aplcdev2pwc apl.ABCconfig cellMNO.cdev -o cell.pwcdev
```

## aplcdev2pwc Configuration File

---

The input configuration file has many of the same keywords as that for APL cdev/pwcap characterization. Only the following keywords are needed in this conversion:

PROCESS <process name> - specifies a process type, such as TT, SS, or FF  
VDD <value> - specifies the voltage  
VDD\_PIN\_NAME <name> - specifies the VDD pin name  
GND\_PIN\_NAME <name> - specifies the ground pin name  
PRIMARY\_VDD\_PIN <name> - in low power designs, specifies the associated power pin name  
PRIMARY\_GND\_PIN <name> - in low power designs, specifies the associated ground pin name  
SWEEPVALUE <value1> [<value2> ...] - specifies a list of voltage samples in pwcdev  
LEAKAGE\_SCALING <leakage\_scaling\_factor\_for\_vdd1> [<leakage\_scaling\_factor\_for\_vdd2> ... ] - specifies custom scaling factors for leakage for each voltage in SWEEPVALUE. For example:  
SWEEPVALUE 1.0 0.9 0.8 0.7  
LEAKAGE\_SCALING 2 1.9 1.8 1.7

In above example, leakage is scaled by a factor 2 for 1.0V, 1.9 for 0.9V and so on. If LEAKAGE\_SCALING is specified, SWEEPVALUE should also be there, otherwise the process errors out. There also should be an equal number of values.

## aplchk

---

The ‘aplchk’ utility performs several types of APL checks from the UNIX command line. The -pwc -spice’ options perform pwcap validation, similar to cap validation. The syntax is:

```
aplchk -pwc -l <cell_list> -spice <apl_config_file>
```

The output file format is:

```
CELL <cellname> <power/ground pin> <state> <voltage value index>
<model peak-to-peak current value> <cell peak-to-peak current value>
<diff = abs(model pp. value - cell pp. value) / max(cell pp. value, model pp.
value) * 100%>
```

## clampviewer

---

Linux executable utility '\$APACHEROOT/bin/clampviewer' previews ESD clamp I-V data. The 'clampviewer' command takes the PF-S clamp file as input, and if there is any clamp I-V data defined in the file, it xgraph plots the I-Vs, similar to the tcl command 'pfs show clamp\_iv <ivName>'. The syntax is:

```
clampviewer <IV_FileName> [<IV_Name>] [<options>]
```

The available options are :

IV\_Name : name(s) of I-V plots specified in the file to plot (for example, 'iv1, iv2'),  
By default all I-Vs in the file are plotted.  
-h : displays specified help message  
-i <min,max> : plot range for current (default: [-10,10])  
-v <min,max> : plot range for voltage (default: [-10,10])  
-marker : draw marker  
-oneview : put all I-Vs into one plot  
-tlp : also plot TLP data (when I-V is derived from TLP\_FILE)  
-fit <value> : TLP fitting factor (default: 1.0)  
-di <value> : di = 'delta I', for specifying TLP\_I\_RESOLUTION value in clamp IV file

An example I-V output is shown in Figure E-12.

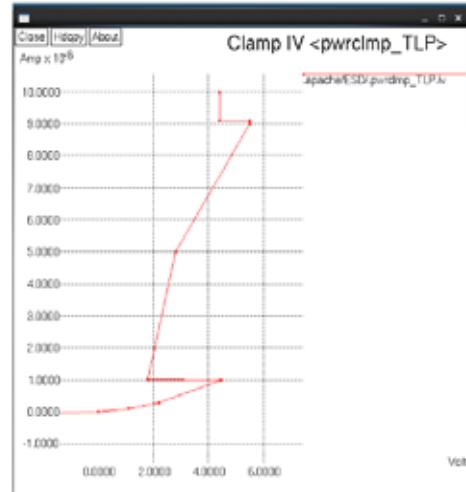
```
clampviewer all_clamps_IV.info pwrcimp_TLP -i -1e-6,1e-5 -v -1,7 -m
```

```
file: all_clamps_IV.info

BEGIN_CLAMP_IV
NAME pwrcimp_TLP
TLP_FILE tlp.txt
END_CLAMP_IV

```

| Vout | Iin      |
|------|----------|
| -0.5 | 1E-8     |
| 0.0  | 0        |
| 1.1  | 1E-07    |
| 2.2  | 3E-07    |
| 3.3  | 6E-07    |
| 4.4  | 1E-06    |
| 1.8  | 1.01E-06 |
| 2.8  | 5E-06    |
| 5.5  | 9E-06    |
| 4.4  | 9.2E-06  |



**Figure E-12** clampviewer IV plot

# Appendix F

## Third-Party Software Licenses

This product contains licensed software which requires reproduction of the following notices:

---

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

### 1. DEFINITIONS

"Contribution" means:

- a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
- b) in the case of each subsequent Contributor:
  - i) changes to the Program, and
  - ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

### 2. GRANT OF RIGHTS

- a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.
- b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.
- c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other

entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

### 3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

- a) it complies with the terms and conditions of this Agreement; and
- b) its license agreement:
  - i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
  - ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
  - iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and
  - iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

- a) it must be made available under this Agreement; and
- b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

### 4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

## 5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

## 6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## 7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions

(including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

---

#### The MIT License (MIT)

Copyright (c) 2017 ZHU Kaidi

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-

---

INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

Copyright (C) 1996-2002 by Jorn Lind-Nielsen All rights reserved

Permission is hereby granted, without written agreement and without license or royalty fees, to use, reproduce, prepare derivative works, distribute, and display this software and its documentation for any purpose, provided that (1) the above copyright notice and the following two paragraphs appear in all copies of the source code and (2) redistributions, including without limitation binaries, reproduce these notices in the supporting documentation. Substantial modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated in all files where they apply.

IN NO EVENT SHALL JORN LIND-NIELSEN, OR DISTRIBUTORS OF THIS SOFTWARE BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHORS OR ANY OF THE ABOVE PARTIES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JORN LIND-NIELSEN SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

---

The GAlib source code is not in the public domain, but it is available at no cost. As a work developed by Matthew Wall using MIT resources and MIT funding, the original GAlib source code copyright is owned by the Massachusetts Institute of Technology. Later portions were added by Matthew Wall without MIT funding. All rights are reserved.

#### GAlib License Agreement

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

You may copy and distribute the source code and/or library/executable code for GAlib in any medium provided that you conspicuously and appropriately give credit to the author and keep intact all copyright and disclaimer notices in the library.

Any publications of work based upon experiments that use GAlib must include a suitable acknowledgement of GAlib. A suggested acknowledgement is: "The software for this work used the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology."

The author of GAlib and MIT assume absolutely no responsibility for the use or misuse of GAlib. In no event shall the author of GAlib or MIT be liable for any damages resulting from use or performance of GAlib.

---

The GIFLIB distribution is Copyright (c) 1997 Eric S. Raymond

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

Copyright (c) 1992-2013 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved. Copyright (c) 2000-2013 The University of California Berkeley. All rights reserved. Copyright (c) 2006-2013 The University of Colorado Denver. All rights reserved.

\$COPYRIGHT\$

Additional copyrights may follow

\$HEADER\$

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR

---

---

BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright (c) 2003-2009 Tim Kientzle All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer in this position and unchanged. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR(S) ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR(S) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright (C) 2002-2011 Carsten Haitzler and various contributors (see AUTHORS)  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

---

Copyright 1990 - 1997 by AT&T, Lucent Technologies and Bellcore.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the names of AT&T, Bell Laboratories, Lucent or Bellcore or any of their entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

AT&T, Lucent and Bellcore disclaim all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall AT&T, Lucent or Bellcore be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

---

Llibigl is licensed under the Mozilla Public License (MPL) Version 2.0, the text of which can be found at: <https://www.mozilla.org/media/MPL/2.0/index.815ca599c9df.txt>. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the Llibigl source code.

---

This software is based in part on the work of the Independent JPEG Group.

---

Copyright 1995-2013, Regents of the University of Minnesota

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at  
<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

---

Copyright (C) 2002 Michael Ringgaard. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright (c) 2005 Nominet UK ([www.nic.uk](http://www.nic.uk))  
All rights reserved.

Contributors: Ben Laurie, Rachel Willmer. The Contributors have asserted their moral rights under the UK Copyright Design and Patents Act 1988 to be recorded as the authors of this copyright work.

This is a BSD-style Open Source licence.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Nominet UK or the contributors may not be used to endorse or promote products derived from this software without specific prior written permission;  
and provided that the user accepts the terms of the following disclaimer:

THIS SOFTWARE IS PROVIDED BY NOMINET UK AND CONTRIBUTORS "AS IS"  
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

---

THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL NOMINET UK OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [openssl-core@openssl.org](mailto:openssl-core@openssl.org).
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).  
This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License -----

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, Ihash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the rouines from the library being used are not cryptographic related :-. 4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE

---

USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

---

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright (c) 2009, Krasnoshchekov Petr  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

---

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the <organization> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY Krasnoshchekov Petr "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL Krasnoshchekov Petr BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright [Copyright (c) 1994-1996 The Regents of the Univ. of California. All rights reserved.]

Permission is hereby granted, without written agreement and without license or royalty fees, to use, copy, modify, and distribute this software and its documentation for any purpose, provided that the above copyright notice and the following two paragraphs appear in all copies of this software.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.]

---

This program is based in part on the work of the Qwt project (<http://qwt.sf.net>).

---

Copyright (c) 1985,86,87,88 by Kenneth S. Kundert and the University of California.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the copyright notices appear in all copies and supporting documentation and that the authors and the University of California are properly credited. The authors and the University of California make no representations as to the suitability of this software for any purpose. It is provided 'as is', without express or implied warranty.

\$Date: 88/06/18 11:13:29 \$ \$Revision: 1.2 \$

---

Copyright (c) 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy)

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. (3) Neither the name of Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright (C) 1996, Makoto Matsumoto All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Copyright, and License:

Copyright (c) 2005 by Timothy A. Davis, University of Florida, davis@cise.ufl.edu. All Rights Reserved.

UMFPACK License:

Your use or distribution of UMFPACK or any derivative code implies that you agree to this License.

THIS MATERIAL IS PROVIDED AS IS, WITH ABSOLUTELY NO WARRANTY EXPRESSED OR IMPLIED. ANY USE IS AT YOUR OWN RISK.

Permission is hereby granted to use or copy this program, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses UMFPACK or any modified version of UMFPACK code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. This software was developed with support from the National Science Foundation, and is provided to you free of charge.

---

Most files in this release are marked with the copyrights of the organizations who have edited them. The copyrights below are in no particular order and generally reflect members of the Open MPI core team who have contributed code to this release. The copyrights for code used under license from other parties are included in the corresponding files.

Copyright (c) 2004-2010 The Trustees of Indiana University and Indiana University Research and Technology Corporation. All rights reserved.

Copyright (c) 2004-2010 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright (c) 2004-2010 High Performance Computing Center Stuttgart, University of Stuttgart. All rights reserved.

Copyright (c) 2004-2008 The Regents of the University of California. All rights reserved.

Copyright (c) 2006-2010 Los Alamos National Security, LLC. All rights reserved.

Copyright (c) 2006-2010 Cisco Systems, Inc. All rights reserved.

Copyright (c) 2006-2010 Voltaire, Inc. All rights reserved.

Copyright (c) 2006-2011 Sandia National Laboratories. All rights reserved.

Copyright (c) 2006-2010 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.

Copyright (c) 2006-2010 The University of Houston. All rights reserved.

Copyright (c) 2006-2009 Myricom, Inc. All rights reserved.

Copyright (c) 2007-2008 UT-Battelle, LLC. All rights reserved.

Copyright (c) 2007-2010 IBM Corporation. All rights reserved.

Copyright (c) 1998-2005 Forschungszentrum Juelich, Juelich Supercomputing Centre, Federal Republic of Germany

Copyright (c) 2005-2008 ZIH, TU Dresden, Federal Republic of Germany

Copyright (c) 2007 Evergrid, Inc. All rights reserved.

Copyright (c) 2008 Chelsio, Inc. All rights reserved.

Copyright (c) 2008-2009 Institut National de Recherche en Informatique. All rights reserved.

Copyright (c) 2007 Lawrence Livermore National Security, LLC. All rights reserved.

Copyright (c) 2007-2009 Mellanox Technologies. All rights reserved.

Copyright (c) 2006-2010 QLogic Corporation. All rights reserved.

Copyright (c) 2008-2010 Oak Ridge National Labs. All rights reserved.

Copyright (c) 2006-2010 Oracle and/or its affiliates. All rights reserved.

Copyright (c) 2009 Bull SAS. All rights reserved.

Copyright (c) 2010 ARM Ltd. All rights reserved.

Copyright (c) 2010-2011 Alex Brick <bricka [a] ccs.neu.edu>. All rights reserved.

Copyright (c) 2012 The University of Wisconsin-La Crosse. All rights reserved.

\$COPYRIGHT\$

Additional copyrights may follow

\$HEADER\$

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.

Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and other parties. The following terms apply to all files associated with the software, unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

**GOVERNMENT USE:** If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

---

This file is subject to the following copyright notice, which is different from the notice used elsewhere in Tcl, but roughly equivalent in meaning.

Copyright (c) 1992,1993,1995,1996, Jens-Uwe Mager, Helios Software GmbH

Not derived from licensed software.

Permission is granted to freely use, copy, modify, and redistribute this software, provided that the author is not construed to be liable for any results of using the software, alterations are clearly marked as such, and this notice is not modified.

---

Copyright (c) 1987-1994 The Regents of the University of California.

Copyright (c) 1994-1995 Sun Microsystems, Inc.

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

**RESTRICTED RIGHTS:** Use, duplication or disclosure by the government is subject to the restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software Clause as DFARS 252.227-7013 and FAR 52.227-19.

Copyright (c) 1993-1998 Lucent Technologies, Inc.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and warranty disclaimer appear in supporting documentation, and that the names of Lucent Technologies any of their entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

Lucent disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall Lucent be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

---

Gifsicle is copyright (C) 1997-2017 Eddie Kohler and licensed under GNU General Public License (GPL) Version 2. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the Gifsicle source code.

G Wave is copyright (C) Steve Tell and licensed under GNU General Public License (GPL) Version 2. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the GNU Readline source code.

GNU Readline is copyright (C) Free Software Foundation and licensed under GNU General Public License (GPL) Version 2. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the GNU Readline source code.

#### GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

#### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too,

receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

#### TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License.

(Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above, provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sub license, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sub license or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative

works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

## NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

---

CSPARSE is Copyright (C) 2006, Timothy Davis and licensed under GNU Library General Public License (LGPL) Version 2.1. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the CSPARSE source code.

GDK, GTK+, and GLib are Copyright (C) The GNOME Project and licensed under GNU Library General Public License (LGPL) Version 2.1. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the GDK or GLib source code.

Libart is Copyright (C) Raph Levien and licensed under GNU Library General Public License (LGPL) Version 2.1. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the Libart source code.

The Qt GUI Toolkit is Copyright (C) 2015 The Qt Company Ltd. and licensed under GNU Library General Public License (LGPL) Version 2.1. Please contact [open.source@ansys.com](mailto:open.source@ansys.com) for a copy of the Qt GUI Toolkit source code.

#### GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

---

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many

---

libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

#### GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices. Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy. This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the

requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License.

Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law. If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.) Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sub license, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sub license, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**END OF TERMS AND CONDITIONS**