cadence™

Voltus IC Power Integrity Solution User Guide

Product Version 21.17
March 2023

© 2013-2023 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 1-800-862-4522.

All other trademarks are the property of their respective holders.

Restricted Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

About This Manual . . . 17

How This Manual Is Organized . . . 17

Conventions Used in This Manual . . . 18

Related Documents . . . 19

Voltus Product Documentation . . . 19

Stylus Common UI Product Documentation . . . 19

Additional Learning Resources . . . 20

Online Trainings . . . 20

Videos . . . 20

1
Product and Licensing Information . . . 21

Overview . . . 22

About Voltus IC Power Integrity Solution Products and Product Options . . . 22

Voltus IC Power Integrity Solution . . . 22

Voltus IC Power Integrity Solution Product Options . . . 23

About Voltus IC Power Integrity Solution Licenses . . . 25

Licensing Terminology . . . 26

Checking Out Licenses for Product Options . . . 26

2
Getting Started . . . 29

Product and Installation Information . . . 30

Setting the Run-Time Environment . . . 30

Temporary File Locations . . . 31

Launching the Voltus Console . . . 31

Tab Completing Command Names, Parameter Names, and Enum Values . . . 32

Command-Line Editing . . . 33

© 2013-2023 All Rights Reserved.

Setting Preferences 34
Starting the Software 35
Accessing Documentation and Help 36

3
Data Preparation 41
Data Requirements for Power and IRDrop Analysis 41

4
Using the Graphical User Interface 45
Main Window 46
Menu Bar 47
Tabs 48
Toolbar Widgets 49
File Menu 50
View Menu 51
Power Rail Menu 52
Package Menu 54
Verify Menu 55
Tools Menu 55
Windows Menu 55
Flows Menu 55
Help Menu 55

5
Importing Design 57
Importing Innovus Database 58
Importing Third Party Design 60
Working with OpenAcess Databases 61
Importing a Large Hierarchical Design 62
Working with Hierarchical Databases 68
Importing CPF 71

© 2013-2023 All Rights Reserved.

6
Distributed Processing . . . 73

Overview . . . 74
Setting-up Distributed Processing . . . 74
Setting-up Power Analysis in Multi-CPU Mode . . . 75
Standalone Distributed Power Analysis . . . 76
Multi-size Heterogeneous Distributed Processing . . . 78
Setting-up Library Generation in Multi-CPU Mode . . . 80
Setting-up IR Drop Analysis in Multi-CPU Mode . . . 80

7
Extensively Parallel Processing . . . 1

Introduction to Voltus-XP . . . 2
Key Features and Concepts . . . 3
Cloud-based Design Environment . . . 3
Partitioned Database and Fully Distributed Flow . . . 4
Simple Use Model . . . 5
Sub-Flow Data Isolation . . . 5
Efficient Visualization of the Output Directory Structure . . . 6
Elastic Resource Usage . . . 7
Improved GUI Performance . . . 7
Enhanced Message Logging . . . 7
TCL Command Parameters for Performing Rail Analysis in the XP Mode . . . 9
Setting up and Running Dynamic Power and Rail Analysis . . . 10
Voltus-XP Output Directory Structure . . . 16
Distributed GUI . . . 18

8
Design Sanity Checks . . . 21

Check Design . . . 22
Check Timing Library . . . 25
Verify Connectivity . . . 26
Before you begin . . . 26
Types of connectivity violations reported . . . 26

© 2013-2023 All Rights Reserved.

Verify Power Via . . . 29
Verify Power and Ground Shorts . . . 31
Before you begin . . . 31
Types of short violations reported . . . 31
Violation Browser . . . 32
Violation Browser Features . . . 34
Clearing Violations . . . 35
Design Browser . . . 36

9
Early Power-Grid Analysis . . . 37
Overview . . . 38
Early Rail Analysis Key Features . . . 41
Prior to Running Early Rail Analysis . . . 42
Setting up and Running Early Rail Analysis . . . 43
Setting Up the GUI to Run ERA . . . 47
Viewing Early Rail Analysis Results . . . 50
TCL Command Examples . . . 51
Example 1: Region-based Power Specification . . . 51
Example 2: User specified Current Regions and Instance Power File for Total Power 51
Example 3: Auto generated Current Regions and Instance Power File from Total Power 51
Example 4: Power Engine Calculated Power . . . 52
Additional Examples . . . 52

10
Power-Grid Library Generation . . . 55
Overview . . . 57
Inputs . . . 58
Outputs . . . 59
Types of Power-Grid Libraries . . . 60
Power-Grid View Types . . . 60
Generating a Technology Library . . . 63
Generating Standard Cells Power-grid Library . . . 66
Creating a Power-grid View for a Macro Cell . . . 70

© 2013-2023 All Rights Reserved.

Characterizing Current Waveforms for Macro EM Views 75
Trigger File Format - for simulation vector-based files 86
Trigger File Format - for user-defined PWLs 87
Trigger File Format - FSDB 91
Trigger File Format - Datasheet 93
Trigger File Format - for user-specified Spice Deck 96
Trigger File Format - Dotlib 97
Sample Library Generation Output Report 100
Power-Grid View Generation Flow for Macros from PGDB/xDSPF 101
Multi-Mode Power-Grid View Generation 103
Additional Reports 107
Multi-Voltage Capacitance Characterization 109
Characterizing Decap, Filler, and Damping Decap Cells 111
Generating Power-Grid Library for a Flip Chip Design 112
Generating Open Access Based Library 115
Checking PowerGrid Library and Report 117
Viewing and Debugging PowerGrid Library 119
Validating a PowerGrid Library 126

11
Static Power, IRdrop and EM Analysis 127

Static Power Calculation 129
Static Power Calculation Overview 129
Vector-based Average Power Calculation 139
Propagation-based Average Power Calculation 140
Static Power Analysis Flow 144
Power Estimation of a Pre-CTS Netlist 150
Power Map File for Thermal Analysis 151
Voltus Thermal Model Generation for Thermal Analysis 158
Viewing Static Power Analysis Results 160
Interactive Queries of Power Data 173
Debugging Instance Power 174
Saving and Restoring Power Database 177
Generating Standalone Power Reports 178
Resistance Analysis 179

© 2013-2023 All Rights Reserved.

Setting up and Running Resistance Analysis . . . 180
Resistance Analysis Reports . . . 189
Viewing Resistance Analysis Results . . . 192
Additional TCL Commands . . . 194
Static Rail and EM Analysis . . . 195
Static IRDrop and EM Analysis Overview . . . 195
Static Rail Analysis Setup and Run . . . 226
Viewing Static Rail Analysis Results . . . 231
Generating Text Reports . . . 244
Power-Grid Integrity Reports . . . 247
Layer-Based IR Drop Report . . . 249
HTML Reports . . . 250
Constant Current Source Based IR Analysis . . . 254
Hotspot Debugger . . . 255

12
Dynamic Power and IRDrop Analysis . . . 257

Overview . . . 259
Dynamic Power Analysis . . . 260
Vectorless Methodology . . . 261
Probability-Based Vectorless Methodology . . . 261
State-Propagation-Based Vectorless Methodology . . . 262
Vector-Driven Methodology . . . 265
Vector Profiling . . . 266
Combining Vector Profiling and Power Analysis . . . 268
Power Density Aware Vector Profiling . . . 269
Dynamic Vectorless Power Analysis Flow . . . 271
Viewing Dynamic Current Waveforms . . . 276
Dynamic Vector-based Power Analysis Flow . . . 284
Viewing Profiling Database . . . 288
Additional Tcl Command Examples . . . 291
Scan Mode Analysis . . . 297
Generating User-defined Mode Based Current Waveform with Automatic PWL Scaling 301
Dynamic Rail Analysis Overview . . . 303
Setting up and running Dynamic Rail Analysis . . . 308

© 2013-2023 All Rights Reserved.

Run Dynamic Power Analysis and Rail Analysis in Parallel 311
Dynamic Rail Analysis Plots, Movies and Waveforms 313
Dynamic Rail Analysis for On-Chip Voltage Regulators 316
Using Characterized Current PGV for Dynamic Power and Rail Analysis 319
Dynamic Mixed Mode (Vector-based and Vectorless) Analysis 321
Noise Margin Calculation 324
Effective Instance Voltage Based Noise Margin Calculation 325
Performing Noise Margin Calculation 326

13
Hierarchical Power Integrity Analysis with Extreme Modeling
331

Overview 332
Using Extreme Modeling 333
xPGV Generation 334
xPGV Instantiation 335
Setting up and Running the Hierarchical PI Analysis Flow 336

14
ESD Analysis 341

Overview 342
Data Requirements for ESD Effective Resistance Checks 344
Specifying ESD Rules 346
ESD Rule File Examples 358
Performing ESD Analysis 363
Analyzing ESD Reports 367
Effective Resistance Check Reports 367
Current Density Check Report 373
Summary Report 376
Viewing Effective Resistance Check Results 382
CDM Based ESD Analysis 386

© 2013-2023 All Rights Reserved.

15
Package Analysis . . . 389
Overview . . . 390
Static and Dynamic Rail Analysis Including Package Model . . . 391
Simple Lumped RLC Pin Package Model . . . 391
Fully Distributed and Coupled RLCK Package SPICE Model . . . 393
RDL0 or Off-Chip Package Tracing . . . 397
Package to Die Mapping . . . 398
Voltus-Sigrity Package Analysis Flow . . . 400
Package Extraction . . . 400
Rail Analysis Including Package Model . . . 402
Package Model Analysis . . . 406
View Package Analysis Results . . . 408
Access Sigrity Tools from Voltus . . . 409
Example Script . . . 411
Generating die-model . . . 413
Need for die-model generation . . . 413
Die-model generation features . . . 414
Voltus flow to create die-model . . . 415
Time domain analysis of package and die-model . . . 417
Frequency domain analysis of package and die-model . . . 420

16
What-If Rail Analysis . . . 423
Scaling Resistance of the Power-Grid . . . 424
Scaling Static and Dynamic Currents . . . 426
Creating Static and Dynamic Current Regions . . . 428
Scaling Capacitance . . . 429
Creating Virtual Shapes . . . 431
What-If Analysis Flow . . . 433

17
Power Gate Analysis . . . 437
Overview . . . 437

© 2013-2023 All Rights Reserved.

Types of Power Gates . . . 438
Power Gate Analysis . . . 439
Power Gate Library Characterization . . . 440
Fine-grain memory characterization . . . 443
Steady state analysis . . . 446
Usage of Common Power Format (CPF) . . . 446
Steadystate Power Analysis . . . 448
Steadystate Rail Analysis . . . 449
Native Power-up Analysis . . . 450
Setup Native Powerup Analysis using GUI . . . 452

18
Through-Silicon Via and System-in-Package . . . 461
Overview to Through-Silicon Via . . . 461
Domain-based Through-Silicon Via Flow . . . 463
Overview to System-in-Package . . . 466
Domain-based System-in-Package Flow . . . 466
Mixed Approach - a Combination of TSV and SiP . . . 468

19
IR Drop Impact on Timing . . . 471
Introduction . . . 471
Basic Voltus/Tempus Flow . . . 472
IR Drop Effect on Delays . . . 474
Voltus Setup for Power and IR Drop . . . 475
Library Setup . . . 479
Reading Libraries Using read_lib Commands . . . 480
MMMC Setup . . . 480
read_design Configuration File . . . 481
Skew Analysis . . . 482
Jitter Analysis . . . 485
Performing Clock Jitter Analysis . . . 487
IR Drop Aware Critical Path Analysis Using Spice . . . 491
Running Spice Simulations with Constant IR Drop Voltage Per Instance . . . 491
Spice Comparison Reports . . . 492

© 2013-2023 All Rights Reserved.

IR Drop Aware Clock Jitter Analysis Using Spice 493
Example Flow Scripts 494
Non-MMMC Mode 494
Setting Up a Delay Corner in MMMC Mode with TriLibs and IR Drop Files Per Corner 495

20
Tempus Power Integrity Analysis 497
Tempus Power Integrity Analysis Overview 498
Licensing Requirements 501
Tempus Power Integrity Flow 502
Performing Tempus PI Analysis 504
Tempus PI Result Analysis and Reporting 511

21
IR-aware ECO Techniques 515
IR Drop Aware Placement 516
Overview 516
Data Requirements for IR Drop Aware Placement 516
IR Drop Aware Placement Flow 517
Debugging IR Aware Placement 523
Sample Run Script 525
Timing-aware IR Drop Fixing 527
IR-aware ECO Flow 528
Hotspot Debugger and ECO Analysis 530
Rechecking IR Drop and Result Analysis 531
Sample Run Script 534

22
Signal ElectroMigration 539
Analyze AC Signal ElectroMigration 540
Overview 540
Calculating Irms Waveform 540
Calculating Ipeak Waveform 541

© 2013-2023 All Rights Reserved.

Calculating Iavg Waveform . . . 543
Calculating Effective Frequency . . . 544
Computing Irms/Ipeak/Iavg for each Routing Segment . . . 545
Checking the AC Current Limits . . . 547
Prevention of AC Signal EM Violations . . . 552
Repair of AC Signal EM Violations . . . 552
Example Script for AC Signal EM Analysis . . . 553
Top Scope Signal EM Analysis . . . 555
Viewing Signal EM Analysis Results . . . 558
Signal EM Report Format . . . 564
Analyze DC Signal ElectroMigration . . . 566
Overview . . . 566
DC Signal EM Analysis using maxCap and maxTran Limits . . . 566
Frequency based Cell EM Limits in Liberty . . . 567
Repair of DC Signal EM Violations . . . 568
Example Script for DC Signal EM Analysis . . . 568

23
Self-Heating Effect Analysis . . . 571

Overview . . . 571
FEOL Self-Heat (delta T FEOL) . . . 572
BEOL Self-Heat (delta T RMS) . . . 572
Data Requirements for SHE Analysis . . . 573
Self-Heating Analysis Flow . . . 574
Sample Run Script . . . 575
View SHE Analysis Plots in Voltus . . . 576
Delta Temperature File Formats . . . 580

24
Statistical ElectroMigration Budgeting . . . 1

Introduction to Statistical ElectroMigration Budgeting . . . 2
Statistical ElectroMigration Budgeting Flow . . . 4
TCL Command Parameters for Running the SEB Flow . . . 5
Setting up and Running Power EM and Signal EM Analyses for FIT Calculation . . . 6
Output Reports . . . 12

© 2013-2023 All Rights Reserved.

Viewing FIT GUI Map 19

25
Body Bias Analysis 21

Overview 21
Library Generation 22
Power Analysis 24
Rail Analysis 26

26
Leakage Power Scaling using Liberate Library Files 29

Generating Liberty File Using Liberate 29
Voltus Use Model for Leakage Power Scaling 31

27
RTL Activity File Flow in Static Power Calculation 33

28
File Formats 37

Library Generation 38
Power Analysis 60
Rail Analysis 85

29
GDS2DEF Utility 113

Overview 113
Cell-Level and Full-Chip Level Implementation 114
Data Requirements for DEF Generation 115
ESD Analysis using convert_gds_to_def 116
Generating a DEF File 117

© 2013-2023 All Rights Reserved.

30
Customizing GDS Layermap and XTC Command File for TRIM Metals Layers 119

31
Glossary 123

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

About This Manual

This manual describes how to use the Voltus IC Power Integrity Solution tool to perform gate-level power-grid analysis on an Application-Specific Integrated Circuit (ASIC) design to determine whether the power grid will be adequate.

How This Manual Is Organized

This manual covers the following topics:

- Product and Licensing Information
- Getting Started
- Data Preparation
- Using the Graphical User Interface
- Importing the Design
- Distributed Processing
- Extensively Parallel Processing
- Design Sanity Checks
- Power Grid Library Generation
- Static Power, IRdrop and EM Analysis
- Dynamic Power and IRDrop Analysis
- Hierarchical Power Integrity Analysis with Extreme Modeling
- ESD Analysis
- Package Analysis
- What-If Rail Analysis
- Power Gate (Switch) Analysis
- Through-Silicon Via and System-in-Package

- IR Drop Impact on Timing
- IR Drop Aware Placement
- Signal ElectroMigration
- Self-Heating Effect Analysis
- Statistical ElectroMigration Budgeting
- File Formats

Conventions Used in This Manual

This document uses the following syntactical conventions:

- Commands and parameters used on the command line are given in `Courier` font:

  `command_name`

- Menu commands and the fields in dialog boxes in the graphical user interface are given in *Helvetica italic* font:

  File – Save

  An en dash (–) separates the menu name and the command name.

- Variables for which you are to substitute a value are given in *`Courier italic`* font:

  `filename`, *`cellname`*, *`layer_name`*

- File names are given in `Courier` font:

  `library.lib`

- Angle brackets enclose optional parameters:

  `<options>`

- Vertical bars (|) in commands indicate choices.

- Square brackets indicate that you must select one of the choices:

  `[a|b|c]`

- Square brackets followed by a plus sign mean that you can choose more than one of the choices, but you must select at least one:

  `[a|b|c]+`

- Use white space (tabs or spaces) to separate a command and its arguments.

Related Documents

For more information about the Voltus family of products, see the following documents. You can access these and other Cadence documents with the Cadence Help online documentation system.

Voltus Product Documentation

- *What's New in Voltus IC Power Integrity Solution*

  Provides information about new and changed features in this release of the Voltus IC Power Integrity Solution (Voltus).

- *Voltus Known Problems and Solutions*

  Describes important Cadence Change Requests (CCRs) for Voltus, including solutions for working around known problems.

- *Voltus Text Command Reference*

  Describes the Voltus text commands, including syntax and examples.

- *Voltus Menu Reference*

  Describes Voltus graphical user interface.

Stylus Common UI Product Documentation

- *What's New in Voltus Stylus Common UI*

  Provides information about new and changed features in this release of the Voltus Stylus Common UI.

- *Voltus Stylus Common UI Text Command Reference*

  Describes the Voltus Stylus Common UI text commands, including syntax and examples.

- *Voltus Stylus Common UI Migration Guide*

  Provides information on migrating from legacy to the Stylus Common UI version of the Voltus software.

- *Stylus Common UI Database Object Information*

Provides information about Stylus Common UI database objects.

For a complete list of documents provided with this release, see the Cadence Help System library.

Additional Learning Resources

Online Trainings

Cadence offers several training courses on Voltus. For a visual guide of the courses available in the Digital Design and Signoff space, click the links provided on our Learning Map page.

You can also write to `training_enroll@cadence.com`.

Videos

The Video Library on Cadence Online Support provides a comprehensive list of videos on various Cadence products. Click the link below to view a list of available Voltus videos:

- *Voltus Video Library*

1

Product and Licensing Information

- Overview on page 22
- About Voltus IC Power Integrity Solution Products and Product Options on page 22
  - Voltus IC Power Integrity Solution on page 22
  - Voltus IC Power Integrity Solution Product Options on page 23
- About Voltus IC Power Integrity Solution Licenses on page 25
  - Licensing Terminology on page 26
  - Checking Out Licenses for Product Options on page 26

© 2013-2023 All Rights Reserved.

Overview

Each Cadence® Voltus IC Power Integrity Solution product is sold as part of a product package. Product packages may also include product options. The options provide advanced features and capabilities.

Each product and product option has a corresponding license. The software uses licenses to determine the features that are available when the software runs.

You can run Voltus from Innovus and Tempus but this can only be done using Voltus licenses. However, you can run the following Voltus features from Innovus without the need of a Voltus license:

- Static power analysis
- Early Rail Analysis (ERA) with static power

ERA with dynamic power requires VTS-XL.

About Voltus IC Power Integrity Solution Products and Product Options

This release of the Voltus IC Power Integrity Solution software includes the following product package and option:

- Voltus IC Power Integrity Solution
- Voltus IC Power Integrity Solution Product Options

Voltus IC Power Integrity Solution

This package includes the products listed in **Table 1-1**. To start any of these products, type the following UNIX/Linux command:

```
voltus
```

Table 1-1 Voltus IC Power Integrity Solution Products

Name	Abbreviation	Prod. Num.	Description

© 2013-2023 All Rights Reserved.

Table 1-1 Voltus IC Power Integrity Solution Products

Voltus IC Power Integrity Solution - L	VTS-L	VTS100	Provides static and dynamic IR Drop and Electromigration analysis. Includes vector-driven and vectorless power and power distribution calculation, what-if analysis, low power optimization, and chip-package co-design. It allows tight integration with Innovus and Tempus. Voltus-L can only run on 1 CPU, that is, it does not support multi-CPU processing.
Voltus IC Power Integrity Solution - XL	VTS-XL	VTS200	Provides all Voltus-L functions plus Hierarchical Analysis and Hotspot Debugger functions. Voltus-XL license is enabled for multi-CPU usage and each license can run up to 8 CPUs.
Virtuoso Digital Signoff Power Solution	VDS-Power	VDS200	Provides all Voltus-L and Voltus-AA functions. 1 license runs up to 50K instances and can be stacked to 2 licenses for a total of 100K instances.

Voltus IC Power Integrity Solution Product Options

The product options provide the extensibility and cost-effective access to additional advanced technologies, such as electrostatic discharge analysis, on-chip voltage regulator, effective resistance calculator, and multi-die rail analysis.

The Voltus IC Power Integrity Solution includes the following product options:

Table 1-2 Voltus IC Power Integrity Solution Product Options

Name	Abbreviation	Prod. Num.	Description

© 2013-2023 All Rights Reserved.

Table 1-2 Voltus IC Power Integrity Solution Product Options

Voltus IC Power Integrity Solution Advanced Analysis GXL Option	VTS-AA Opt.	VTS201	Adds advanced features such as Statistical Leakage Power Analysis (SLPA), effective resistance calculator, on-chip voltage regulator support, clock and signal jitter analysis, qrcTechFile with features on 20nm or below process nodes, power network optimization, and Sigrity Package Analysis (SPA).
Voltus IC Power Integrity Solution - XM	VTS-XM Opt.	VTS202	The Voltus Extreme Modeling or Voltus-XM license enables modeling of large digital blocks (xPGV model generation), and use these models in the top-level full-chip analysis. It is an option to baseline license Voltus-XL. The licensing use model for Voltus-XM is: ■ One Voltus-XM license is required for generating one xPGV model, irrespective of the number of times the specific xPGV model is instantiated (xPGV with multiple instances). That is, if a CPU core's xPGV is instantiated four times, the Extreme Modeling flow would still require only 1 XM license. ■ The number of Voltus-XM licenses are limited to a maximum of 10 for a design with multiple blocks. For example, if a top-level run has to generate 12 xPGV models, you do not require 12 XM licenses. That is, 10 XM licenses will be able to run a design instantiating 12 xPGV models.
Voltus IC Power Integrity Solution - ESD	VTS-ESD Opt.	VTS203	The Voltus Electrostatic Discharge Analysis or Voltus-ESD license enables ESD analysis resistance checks and current density analysis. It is an option to baseline licenses, Voltus-L or Voltus-XL or VDS-P.

© 2013-2023 All Rights Reserved.

Table 1-2 Voltus IC Power Integrity Solution Product Options

Voltus IC Power Integrity Solution - MP (Massive Parallel Option to Voltus XL)	VTS-MP Opt.	VTS300	Adds 8 more CPUs per additional VTS-MP license. Voltus-MP is a license accelerator that only works with the base license of Voltus-XL. Voltus-L does not have multi-CPU capability and hence Voltus-MP does not work with it.
Voltus IC Power Integrity Solution - Sigrity Package Analysis	VTS-SPA Opt.	SIGR955	Enables Sigrity Package Analysis for Voltus. It provides integrated “Voltus + Sigrity” solutions for IC-centric Chip-PKG co-analysis. It is a product suite that contains both Sigrity XtractIM and Sigrity PowerDC but functions per “1-Product-1-License”. **Note**: The license requirement is for the SPA flow is Voltus-L/XL + Voltus-AA + Voltus-SPA.
Tempus Timing Signoff Solution - Power Integrity	TPS-PI Opt.	TPS600	The Tempus Power Integrity license enables identification of IR-sensitive critical paths. It is an option to the baseline licenses, Tempus-XL or Voltus-XL.

Important

Both option licenses (Voltus-MP and Voltus-AA) need a base license (Voltus-L or Voltus-XL). One base Voltus-XL license can run on 8 CPUs, incremental by 8 per license accelerator “VTS-MP” or “VTS-XL”. VTS-MP takes precedence over VTS-XL in license acceleration if both are available.

About Voltus IC Power Integrity Solution Licenses

When you run a command to invoke a product or product option, a license is checked out. Each product and product option has a unique license string (also called a license key).

The version of vendor daemon (`cdslmd`) and license daemon (`lmgrd`) should be 11.16.4.0 or higher. If the daemon versions have not been updated, Voltus may fail to checkout a license:

```
$ voltus
Checking out Voltus IC Power Integrity Solution license ...
```


© 2013-2023 All Rights Reserved.

```
Fail to find any Voltus IC Power Integrity Solution license...
Check your Voltus IC Power Integrity Solution license status.
```

To fix the above error, restart the license server using the latest `cdslmd` and `lmgrd` versions included in the release tree.

Licensing Terminology

The following terminology is useful in understanding licenses.

Base license

The license that is checked out when the software starts. Only a full-fledged product license can be used as a base license. You cannot use a product option license as a base license to start the software. For example, Voltus has VTS-L and VTS-XL as base licenses.

Dynamic license

A license for a product option that is not checked out until a feature provided by the product option is needed. You can check out more than one dynamic license per base license. For more information on dynamic licenses, see the following section "Checking Out Licenses for Product Options".

Multi-CPU license

A license that enables additional CPUs for multithreading or distributed processing. Multi-CPU licenses can be checked out after the base license is checked out. You can check out more than one multi-CPU license per base license. For base license of VTS-XL, you can also stack it up to access more CPUs.

Checking Out Licenses for Product Options

The following command specifies the product options to be checked out when you invoke the software. The product options specified with the `-lic_startup_options` parameter are checked out immediately and the product options specified with the `-lic_options` parameter are checked out dynamically.

```
voltus -lic_startup_options "option1 option2 …" -lic_options
"option1 option2 …"
```


© 2013-2023 All Rights Reserved.

If you do not want any product options to be checked out dynamically, use empty quotes with the `-lic_options` parameter, as follows:

```
voltus -lic_startup_options "option1 option2 …" -lic_options ""
```

The following command can be used to check out product options after you have invoked the software. The product option specified with the `-checkout` parameter is checked out immediately and the product options specified with the `-optionList` parameter are checked out dynamically.

```
set_license_check -checkout option -optionList "option1 option2 …"
```

With the `checkout` parameter, you can specify only one product option. If you do not want any product options to be checked out dynamically, use empty quotes with the `-optionList` parameter, as follows:

```
set_license_check -checkout option -optionList ""
```

You cannot check out a license for a product option if you have not checked out a base license.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

2

Getting Started

- Product and Installation Information on page 30
- Setting the Run-Time Environment on page 30
- Temporary File Locations on page 31
- Launching the Voltus Console on page 31
- Tab Completing Command Names, Parameter Names, and Enum Values on page 32
- Setting Preferences on page 34
- Starting the Software on page 35
- Accessing Documentation and Help on page 36

© 2013-2023 All Rights Reserved.

Product and Installation Information

For product, release, and installation information, see the README file at any of the following locations:

- downloads.cadence.com, where you can review the `README` before you download the software
- In the software installation, where it is also available when you are using or running the software

For information about Voltus IC Power Integrity Solution licenses, see "About Voltus IC Power Integrity Solution Licenses" in the Product and Licensing Information chapter.

Setting the Run-Time Environment

If `install_dir` is the location of your Voltus installation, you should set up your run-time environment like this:

- Add the `install_dir/bin` directory to your path. The `bin` directory has links to all the public executables in the install hierarchy.
- If you want the legacy Voltus man pages to be available from the Unix man command, you can add `install_dir/share/voltus/man` to your `MANPATH` envar.
- If you want the Common UI Voltus man pages to be available from the Unix man command, you can add `install_dir/share/voltus/stylus/man` to your `MANPATH` envar.
- If you want the Tcl man pages to be available from the Unix man command, you can add `install_dir/share/tcltools/man` to your `MANPATH` envar.

For example, you might add this to your startup shell script:

```
set install_dir = /tools/voltus16.2/lnx86
set path = ($install_dir/bin $path)
setenv MANPATH
$install_dir/share/voltus/man:$install_dir/share/tcltools/man:$MANPATH
```

Note: When Voltus launches, it automatically adds either the legacy or the Common UI man pages (depending on the `-stylus` option) and the Tcl man pages to the beginning of the current `MANPATH` inside Voltus. Therefore, from within Voltus, the `man` command will see both sets of man pages before any other man pages.

© 2013-2023 All Rights Reserved.

Supported and Compatible Platforms

The `README` file lists the supported and compatible platforms for this release.

Temporary File Locations

Each Voltus session creates its own temporary directory to store temporary files at the beginning of the run.

By default, `tmp_dir` is created in `/tmp`. If the Unix envar `TMPDIR` is set, then the `tmp_dir` directory is created inside `TMPDIR`.

The name of the `tmp_dir` directory will look like:

```
voltus_temp_[pid]_[hostname]_[user]_xxxxxx
```

Where `_xxxxxx` is a string added to make the directory unique. For example:

```
voltus_temp_10233_farm254_bob_nfp9ez
```

The temporary directory is automatically removed on exit or if the run terminated with a catchable signal (e.g. SIGSEGV).

Launching the Voltus Console

When you start Voltus in the non-GUI environment using the `-no_gui` parameter, the UNIX window (shell tool, xterm, and so on) where you start the Voltus session is called the `Voltus` console. This is where you enter all Voltus text commands and where the software displays messages. When a session is active, this console displays the following prompt:

```
voltus>
```

When you start Voltus in the GUI environment, the Voltus console is displayed within the main Voltus window.

If you use the console for other actions--for example, to use the vi editor--the session suspends until you finish the action.

If you suspend the session by typing `Control-z`, the `voltus>` prompt is no longer displayed. To return to the Voltus session, type `fg`, which brings the session to the foreground.

© 2013-2023 All Rights Reserved.

Tab Completing Command Names, Parameter Names, and Enum Values

You can use the `Tab` key within the software console to complete text command names.

After you type a partial text command name and press the `Tab` key, the software displays the exact command name that completes or matches the text you typed (if the string is unique to one text command) or all the commands that match the text you typed. For example, if you type `set_power_` and press the `Tab` key, the software displays the following commands:

```
voltus 3> set_power_<Tab>
set_power_analysis_temperature  set_power_data       set_power_def_files
set_power_include_file          set_power_lib_files  set_power_output_dir
set_power_pads
```

If you type `set_power_p<Tab>`, the software completes the command name as follows:

```
voltus 3> set_power_pads
```

This function supports all Tcl commands, and the `man` first argument also supports it (for example, `man set_power_p<Tab>`).

Tab Completing Parameter Names

The `Tab` completion capability is also available for parameter names starting with "`-`" for Voltus commands or for user commands registered with `define_proc`.

For example,

```
voltus 1> set_pg_library_mode -en
-enable_device_size_current_distribution  -enable_distributed_processing
-enable_multi_voltage_cap_generation
```

If you type `set_pg_library_mode -enable_m<Tab>`, the software completes the parameter name as follows:

```
voltus 1> set_pg_library_mode -enable_m
set_pg_library_mode -enable_multi_voltage_cap_generation
```

To view all parameters of a command, type the command name followed by `-` and press `Tab`.

For example:

```
voltus 1> rename_pg_library -
-cell_list_map_file  -merge_library_list_file  -output -output_library
```

Tab Completing Enum Type Values for Parameters

Tab completion can be used to view or complete enum values.

© 2013-2023 All Rights Reserved.

For example, if you type `set_pg_library_mode -cell_type` followed by a space and press the `Tab` key, the software displays the following:

```
voltus 3> set_pg_library_mode -cell_type
macros stdcells techonly
```

If you type `set_pg_library_mode -cell_type t<Tab>`, the software completes the value as follows:

```
voltus 5> set_pg_library_mode -cell_type techonly
```

Tab Completing Unix file and directory names

If you type a `<Tab>` in any other context, the Tcl shell will look for Unix file and directory names that match the string.

For example,

```
voltus 5> read_def test<Tab>
test.tcl test.def test.lef
```

Command-Line Editing

The Voltus software provides a GNU Emacs-like editing interface. You can edit a line before it is sent to the calling program by typing control characters. A control character, shown below as a caret followed by a letter, is typed by holding down the `Control` key when typing the character.

Most editing commands can be given a repeat count, *n*, where *n* is a number. To enter a repeat count, press the `Esc` key, the number, and then the command to execute. For example,`Esc 4 ^f` moves forward four characters. If a command can be given a repeat count, the text `[n]` is shown at the end of its description.

You can type an editing command anywhere on the line, not just at the beginning. You can press `Return` anywhere on the line, not just at the end.

Note: Editing commands are case sensitive: `Esc F` is not the same as `Esc f`.

Control (^) Characters

^A	Move to the beginning of the line
^B	Move left (backwards) [n]
^C	Exits from editing mode
^D	Delete character [n]

© 2013-2023 All Rights Reserved.

`^E`	Move to end of line
`^F`	Move right (forwards) [n]
`^G`	Ring the bell
`^H`	Delete character before cursor (backspace key) [n]
`^I`	Complete filename (Tab key); see below
`^J`	Done with line (Return key)
`^K`	Kill to end of line (or column [n])
`^L`	Redisplay line
`^M`	Done with line (alternate Return key)
`^N`	Get next line from history [n]
`^P`	Get previous line from history [n]
`^R`	Search backward (forward if [n]) through history for text; must start line if text begins with an up arrow
`^T`	Transpose characters
`^V`	Insert next character, even if it is an edit command
`^W`	Wipe to the mark
`^X^X`	Exchange current location and mark
`^Y`	Yank back last killed text
`^[`	Start an escape sequence (Esc key)
`^]c`	Move forward to next character c
`^?`	Delete character before cursor (Delete key) [n]

Setting Preferences

You set preferences at the beginning of a new design import. You can assign special characters for the design import parser for Verilog®, DEF, and PDEF files, and control the display of the Physical view window. You can also change the hierarchical delimiter character in the netlist before importing the design, and change the DEF hierarchical default character and the PDEF bus default delimiter before loading the file.

Note: If you change the default values for the DEF delimiter or PDEF bus delimiter, these changes become the default delimiters for the DEF and PDEF writers.

© 2013-2023 All Rights Reserved.

For information on setting design preferences, refer to the "*Set Preferences*" section in the **View Menu** chapter in *Voltus Menu Reference*.

The Voltus Power Integrity Solution software uses the following initialization files for setting preferences:

.ssvrc	Used for setting Tcl parameters or adding user-defined Tcl commands. If different versions of this file exist in the installation, home, or working directories, the file in the working directory takes precedence.
ssv.pref.tcl	Contains design preferences set using the Design and Console tabs in the Preferences form in the GUI. **Note:** By default, Voltus saves changes that you make to your preferences to the `ssv.pref.tcl` file in the working directory.
.ssv	Contains design preferences set using the Console tab in the Preferences form in the GUI.

The initialization files are read in the following sequence:

1. `.ssvrc` in the home directory
2. `.ssvrc` in the working directory
3. `ssv.pref.tcl` in the working directory
4. `.ssv` in the home directory

Note: If initialization files contain conflicting information, the last file read takes precedence.

Starting the Software

To start a Voltus session, type the following command with the appropriate parameters on the UNIX/Linux command line. If you type a command without parameters, the software starts in GUI mode and creates a log file and a command file. The system attempts to check out the license with the most functionality, then the license with the next most functionality, and so on.

`voltus` Starts one of the following products:

- Voltus IC Power Integrity Solution L

© 2013-2023 All Rights Reserved.

- Voltus IC Power Integrity Solution XL

For an overview of the products and product licensing, see Product and Licensing Information on page 21.

Accessing Documentation and Help

You can access the Voltus documentation and help system by using the following methods:

- Launching Cadence Help From the Command Prompt
- Accessing Documentation and Help From the GUI
- Using the man and help Commands on the Text Command Line
- Other Sources of Information

Launching Cadence Help From the Command Prompt

1. Change to the following directory:

   `installation_dir/tools/bin`

2. Enter the following command:

   `./cdnshelp`

After launching Cadence® Help, press `F1` or choose *Help* - *Contents* to display the help page for Cadence Help.

For more information see the *Cadence Help* manual.

Accessing Documentation and Help From the GUI

The software provides the following two methods to access documentation and help from the GUI:

Select Help on the Main Menu

Click *Help* on the main menu and then select *Documentation Library* to open the Cadence Help window. The Cadence Help window provides access to all the documentation shipped with the release.

© 2013-2023 All Rights Reserved.

Alternatively, select *Help*, and then click on any of the following options to open a document directly:

- *Text Command Reference*
  Opens the Table of Contents page of the text command reference.
- *User Guide*
  Opens the Table of Contents page of the user guide.
- *What's New*
  Opens the Table of Contents page of the what's new document

Select Help on a Voltus Form

Click the *Help* button in the bottom right corner of a form. Clicking the *Help* button opens the *Voltus Menu Reference* entry for the form in the Cadence Help window.

Using the man and help Commands on the Text Command Line

Using the help Command to View the Command Syntax

To see syntax information for a command, type the following command in the software console:

`help` *command_name*

For example, to see syntax information for the `read_lib` command, type the following command:

```
help set_power_include_file
```

The software displays the following text:

```
Command: set_power_include_file:
Usage: set_power_include_file [-help] [<file>] [-reset]
-help      # Prints out the command usage
<file>    # Specifies the powermeter include file which contains powermeter options
(string, optional)
-reset     # Specifies to reset and ignore all the previous powermeter include
files. (bool, optional)
```

To see the entire list of Voltus commands and their syntax, type the following command in the software console:

© 2013-2023 All Rights Reserved.

```
help
```

Using the man Command to View the Command Description

To see the complete set of information for a Voltus command, type the following command in the software console:

```
man command_name
```

For example, to see the complete set of information for the `set_power_include_file` command, type the following command:

```
man set_power_include_file
```

The software displays the following text:

Name

```
set_power_include_file - Specifies a power analysis include file
```

Syntax

```
set_power_include_file file [-reset]
```

Description

```
Specifies a power analysis include file. You can use the
set_power_include_file command to specify certain commands of the
dynamic engine which do not have a Voltus equivalent. For more informa-
tion on these commands, refer to the Power Analysis Options chapter.

The set_power_include_file command is required only if the
power_static_netlist attribute is set to def.

Note: The set_power_include_file command is used to pass the legacy
power analysis commands and report_power command options only in the
dynamic mode (vectorbased and vectorless). If the power_method
attribute is set to static, the set_power_include_file command is
ignored.
```

Parameters

```
file   Name of the include file. This file will include all commands
       that can not be translated to Voltus commands. In addition, you
       can also include the report_power command options to generate
       different power reports.
```


© 2013-2023 All Rights Reserved.

```
-reset Resets all options back to default values.
```

Examples

```
* The following command specifies a power include file of
  power.inc:
  set_power_include_file power.inc
```

Other Sources of Information

You can also get help on Cadence products by selecting *Customer Support* on the *Help* menu. The *Customer Support* submenu provides access to the following Cadence resources:

- Cadence Online Support

  Opens Cadence Online Support in your browser.

- Web Collaboration

  Opens SpaceCruiser in your browser.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

3

Data Preparation

Data Requirements for Power and IRDrop Analysis

The following table shows the required and optional inputs for power and IRdrop analysis.

Data	Mandatory?	Details
Timing Libraries	Yes	Synopsys libraries (dot libs) and PVT corner for analysis. Timing Window File (TWF) from STA for slews and vectorless dynamic analysis. If running within the Innovus platform, this file is optional as it will be generated on-the-fly. Foundry, process node, and process name.
Verilog	Yes	Verilog netlist
SDC	Yes	SDC timing constraints file
LEF	Yes	Technology LEF, standard cells, IOs, memories and IP LEF
DEF	Yes	Flattened DEF or Multiple DEFs for design top level and blocks
SPEF	Yes	Flattened or multiple SPEFs for top level and blocks
Spice Subckts	Yes	Spice netlist for all cells in the design along with the Spice models. It is recommended that: ■ the Spice netlist for macros/memories and IO cells have device X/Y location file. ■ the Spice netlist is extracted.

© 2013-2023 All Rights Reserved.

GDS	Yes	Design components for IOs, memories, and IPs
Layer map	Yes	For the GDS layer name, layer numbers for metal, vias, contact, poly and diffusion
Power pad location	Yes	Power and ground voltage source X Y location file or power pad cell names to use during analysis. All Power Domains to be analyzed and their voltages.
Extraction tech file	Yes	For Quantus or process file
CPF	No	Common Power Format file
Package model	No	Package RLC data to analyze package drop and impact of package inductance. Pin based R, L, and C values or spice subcircuit format is supported.
List of filler and decap cells	No	Needed for decap optimization.
VCD or TCF or FSDB	No	Full VCD file for vector-based analysis Boundary TCF or VCD file to annotate activity at primary inputs of the design. If not available, default % activity is used at the primary inputs. Global flop activity or partial TCF/VCD with activity on flops. Clock gating instances and activity at the output.
LVS rule file	No	Calibre, Hercules, or Assura for device recognition.
Design power	No	Expected static power. Power consumption file for custom cells and IPs that do not have internal and leakage power defined in dot libs.
ElectroMigration (EM) rules	No	Current limits per unit area for each process layer. Usually available through foundry.

Note:

- Design DEF is optional if the Innovus database is available. You need to provide LEF, Verilog, SDC, and timing libraries associated with the config file in the Innovus database.

© 2013-2023 All Rights Reserved.

- LVS rule file (Calibre, Hercules, Assura) for device recognition.

The following diagram shows the inputs and outputs for Voltus IC Power Integrity Solution:

© 2013-2023
All Rights Reserved.

© 2013-2023 All Rights Reserved.

4

Using the Graphical User Interface

- Main Window on page 46
  - Menu Bar on page 47
  - Tabs on page 48
  - Toolbar Widgets on page 49
- File Menu on page 50
- View Menu on page 51
- Power Rail Menu on page 52
- Package Menu on page 54
- Verify Menu on page 55
- Tools Menu on page 55
- Windows Menu on page 55
- Flows Menu on page 55
- Help Menu on page 55

© 2013-2023 All Rights Reserved.

Main Window

This section describes some of the basic features of the main window.

© 2013-2023
All Rights Reserved.

Menu Bar

The menu bar represents categories of commands. Each of the headings supports a pull-down menu of related commands. There are twelve pull-down menus:

- File
- View
- Power Rail
- Package
- ECO
- Clock
- Timing & SI
- Verify
- Tools
- Windows
- Flows
- Help

Important

The items in the *ECO*, *Clock*, and *Timing & SI* menu require Cadence® Tempus™ Timing Signoff Solution license. See the *Tempus Timing Signoff Solution Menu Reference* for details.

Click a menu bar category to display the group of represented commands. The menu names are enabled or disabled (grayed) depending on the stage of the design flow. With the drop-down menu visible, click on an enabled command to run it.

© 2013-2023 All Rights Reserved.

The drop-down menus support hotkey invocation for menu commands. The hotkey for each command name is shown with an underscore. For example, run the *Power & Rail – Run Power Analysis* command by clicking the *Power & Rail* menu and pressing `Shift+w`.

Tabs

In the display area, the *Layout* tab is open by default. This tab displays the design layout. You can click the **+** sign next to the *Layout* tab title to open any of the following in a new tab:

© 2013-2023 All Rights Reserved.

- *Layout*
- *Setup*
- *Schematics*
- *Design Browser*
- *Layout Viewer*
- *Violation Browser*

Toolbar Widgets

Refer to the section "Toolbar Widgets" for more information about each widget.

© 2013-2023 All Rights Reserved.

File Menu

The following table lists the menu options that are accessible from the *File* menu, and the command-line equivalent and link to the GUI form for each option.

Menu Option	Equivalent Command	Form Name
Read Design	read_design	*Restore Design*
Save Design	save_design	*Save Design*
Read SDC	read_sdc	*Load Timing Constraints*
Read SPEF	read_spef	*Load SPEF*
Read SDF	read_sdf (Tempus command)	*Load SDF File*
Load/Commit CPF	read_power_domain	*Load/Commit CPF*
Specify Operating Conditions	set_op_cond set_timing_library (Tempus commands)	*Specify Operating Condition*
Configure MMMC	create_library_set create_op_cond create_rc_corner create_constraint_mode create_delay_corner update_delay_corner create_analysis_view (Tempus commands)	*MMMC Browser*
Check - Check Design	check_design	*Check Design*
Check - Timing Library	check_library (Tempus command)	*Report Timing Library*

© 2013-2023 All Rights Reserved.

Menu Option	Equivalent Command	Form Name
`Check - Library`	`check_library` (Tempus command)	*Check Timing Library*
`Report - Summary`	-	*Summary Report*
`Report - Netlist Statistics`	-	-
`Hide GUI`	-	-
`Exit`	-	-

View Menu

Refer to the *Voltus Menu Reference* guide for descriptions of the *View* menu options.

© 2013-2023 All Rights Reserved.

Power Rail Menu

The following table lists the menu options that are accessible from the *Power & Rail* menu, and the command-line equivalent and link to the GUI form for each option.

Menu Option	Equivalent Command	Form Name
SetPowerGrid Library Mode	set_pg_library_mode set_advanced_pg_library_mode	*Set PG Library Mode*
Generate PowerGrid Library	generate_pg_library	*Generate PG Library*
Set Power Analysis Mode	set_power_analysis_mode	*Set Power Analysis Mode*
Run Power Analysis	report_power set_switching_activity set_default_switching_activity set_power	*Run Power Analysis*
Set Rail Analysis Mode	set_rail_analysis_mode	*Set Rail Analysis Mode*
Run Rail Analysis	analyze_rail scale_what_if_current scale_what_if_capacitance scale_what_if_resistance create_what_if_shape create_current_region	*Run Rail Analysis*
Run Resistance Analysis	analyze_resistance	*Run Resistance Analysis*
Library Cell Viewer	-	*Cell Viewer*

© 2013-2023 All Rights Reserved.

Menu Option	Equivalent Command	Form Name
Power Rail Plots	read_power_rail_results report_power_rail_results set_power_rail_display set_power_rail_layers_nets	*Power and Rail Plots*
Short	-	*Short*
ESD Results	analyze_esd_network	*ESD Summary*
Textual Reports - PowerGrid Library	check_pg_library	*Power Library Report*
Textual Reports - Power Analysis	report_power	*Report Power*
Histograms - Power Analysis	-	*Power Debug*
Dynamic Results - Waveforms	view_dynamic_waveform	*View Waveforms*
Dynamic Results - Movies	view_dynamic_movie	*View Movies*

© 2013-2023 All Rights Reserved.

Package Menu

The following table lists the menu options that are accessible from the *Package* menu, and the command-line equivalent and link to the GUI form for each option:

Menu Option	Equivalent Command	Form Name
Set Advanced Package Options	set_advanced_package_options	*Set Advanced Package Options*
Extract Package	extract_package	*Extract Package*
Analyze Package	analyze_package	*Analyze Package*
Package Plots	view_package_results	*Package Plots*
Sigrity - MCP Editor - Generate Mapping File	map_die_package	*MCP Editor*
Sigrity - MCP Editor - Merge Circuit Models	-	*Merge Circuit Models*
Sigrity - XtractIM	-	-
Sigrity - PowerDC	-	-

© 2013-2023 All Rights Reserved.

Verify Menu

The following table lists the menu options that are accessible from the *Verify* menu, and the command-line equivalent and link to the GUI form for each option.

Menu Option	Equivalent Command	Form Name
Connectivity	verify_connectivity	*Verify Connectivity*
Power Via	verify_power_via	*Verify Power Via*

Tools Menu

Refer to the *Voltus Menu Reference* guide for descriptions of the *Tools* menu options.

Windows Menu

Refer to the *Voltus Menu Reference* guide for descriptions of the *Windows* menu options.

Flows Menu

The following table lists the menu options that are accessible from the *Flows* menu, and the command-line equivalent and link to the GUI form for each option:

Menu Option	Equivalent Command	Form Name
Create Template	write_flow_template	*Save Foundation Flows Template*

Help Menu

Refer to the *Voltus Menu Reference* guide for descriptions of the *Help* menu options.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

5

Importing Design

Voltus allows flexible design import options.

- Importing Innovus Database on page 58
- Importing Third Party Design on page 60
- Working with OpenAcess Databases on page 61
- Importing a Large Hierarchical Design on page 62
  - Hierarchy Flattener on page 62
  - Saving a Database in Voltus and Innovus on page 63
  - Running Analysis without Creating a Physical Database on page 64
  - Running Analysis without Design Loading on page 65
  - Reading RC Parasitics for Power Calculation on page 68
- Working with Hierarchical Databases on page 68
- Importing CPF on page 71

© 2013-2023 All Rights Reserved.

Importing Innovus Database

This section describes how to load an Innovus database.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- SPEF

Procedure

To load a design into Voltus, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command: You can load the design database using the `read_design` command, as shown below:

   ```
   read_design -physical_data ../design/postRouteOpt.enc.dat
   super_filter
   ```

   Alternatively, you can read the design in fragments by loading libraries and netlists using commands, such as `read_lib`, `read_verilog`, and so on, as shown below:

   ```
   set lefs [list]
   lappend lefs ../data/lef/gsclib090_tech.lef
   lappend lefs ../data/lef/gsclib090_macro.lef
   lappend lefs ../data/lef/pll.lef
   lappend lefs ../data/lef/decap.lef
   read_lib -lef $lefs
   read_view_definition ../design/viewDefinition.tcl
   read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
   set_top_module super_filter -ignore_undefined_cell
   read_def ../design/super_filter.def.gz
   ```

2. Specify the CPF file for power domain information through the command line or through the GUI.

© 2013-2023 All Rights Reserved.

GUI: Select *File - Load/Commit CPF*.

Tcl command:

```
read_power_domain -cpf ../design/super_filter.cpf
```

Alternatively, you can define each power/ground net's voltage:

```
Using the following options to define each power/ground net's
voltage:
set_dc_sources VDD_AO -power -voltage 0.9 -force
set_dc_sources VDD_external -power -voltage 0.9 -force
set_dc_sources VDD_ring -power -voltage 0.9 -force
set_dc_sources VDD_column -power -voltage 0.9 -force
set_dc_sources VSS -ground
```

3. Specify the SPEF file for signal net parasitic information through the command line or through the GUI.

GUI: Select *File - Read SPEF*.

Tcl command:

```
read_spef -rc_corner  RC_wc_125 -decoupled\
../design/postRouteOpt_RC_wc_125.spef.gz
```


© 2013-2023 All Rights Reserved.

Importing Third Party Design

Voltus IC Power Integrity Solution allows you to load verilog, timing libraries, SDC, DEF(s), LEF(s) and SPEF(s). The design data can be loaded from the GUI using the *Setup* tab as shown in the following steps:

1. Select the *Setup* tab.
2. Fill out the fields or click *Load* to load a design setup file that has the appropriate Tcl commands.
3. Click *Apply*.

TCL Command:

The design data can be loaded using the following Tcl commands:

```
read_lib -lef tech.lef allcells.lef
read_lib -min stdcells_min.lib memories_min.lib
```


© 2013-2023 All Rights Reserved.

```
read_lib -max stdcells_max.lib memories_max.lib
read_verilog design.v
set_top_module design
read_sdc design.sdc
read_def design.def.gz
read_spef design.spef.gz
read_power_domain -cpf design.cpf
```

To improve the design import performance, the `read_def` command supports the `-skip_signal` parameter which skips signal nets when loading the DEF files.

```
read_def -skip_signal design.def
```

Working with OpenAcess Databases

Importing an OpenAccess Database

To load an OpenAccess database from the GUI, perform the following steps:

1. From the *File* menu, choose *Read Design*.
2. Choose the *OA* option.
3. Specify the following information:
   - *Library* - Specifies the OpenAccess database library.
   - *Cell* - Specifies the OpenAccess database cell.
   - *View* - Specifies the OpenAccess database view.
4. Choose the *Read Physical Data* option, and click *OK*.

You can use the following command to read physical information from the OpenAccess database:

```
read_design -cellview "designLib cell1 layout" -physical_data
```

You can use the following command parameter to read the LEF data directly from OpenAccess reference libraries:

```
read_lib -oaRef library_name
```


© 2013-2023 All Rights Reserved.

Saving an OpenAccess Database

Use the `save_design -cellview {lib cell view}` command to save an OpenAccess design. A Verilog based design can only be saved into OpenAccess if it uses OpenAccess reference libraries. It cannot be saved in OpenAccess if LEF files were used.

Restoring an OpenAccess Database

The OpenAccess database can be restored in Voltus by using the following Tcl command:

```
restore_oa_design designLib cell1 layout -physical_data
```

This command restores the OpenAccess design called `cell1`. The library is called `designLib` and the view is `layout`.

Importing a Large Hierarchical Design

Hierarchy Flattener

Voltus includes a hierarchical netlist flattener that can read the hierarchical verilog, DEFs, and SPEFs to create a flattened Innovus database in memory. It flattens all levels of hierarchical DEFs automatically.

It is recommended that you specify the `-skip_signal` parameter when reading multiple DEFs to achieve better performance.

© 2013-2023 All Rights Reserved.

You must specify `read_lib -lef` to load the LEFs. This command should never include the LEF definition of the block-level partitions. It should only include the LEF definition for primitive cells like standard cells, IOs, and memories.

TCL Commands

Voltus Load Hierarchical Design Tcl Script

```
read_lib -lef tech.lef allcells.lef
read_lib -min stdcells_min.lib memories_min.lib
read_lib -max stdcells_max.lib memories_max.lib
read_verilog top.v block1.v block2.v
set_top_module top
read_sdc top.sdc
read_def -skip_signal top.def block1.def block2.def subblock1.def
read_spef top.spef block1.spef block2.spef
```

Saving a Database in Voltus and Innovus

The `save_design` command in Voltus (`saveDesign` in Innovus) can be used to save timing graph, signal net parasitics, verilog, timing constraints, power constraints, and reference to design data.

The `save_design` command in Voltus (`saveDesign` in Innovus) saves a power constraints file which saves all the power constraints and modes in the database into a file named

© 2013-2023 All Rights Reserved.

<topcellname_power_constraints.tcl>. On restoring the database, this file be sourced to ensure that the same power constraint settings are used in the new session.

Sample Power Constraints File:

```
set_default_switching_activity -period 10.000000ns
read_activity_file -format VCD -scope adder_tb_e.u1
/../adder_netlist.vcd

set_power_analysis_mode -create_binary_db false -handle_glitch false
-handle_tri_state true -honor_negative_energy true -
ignore_control_signals true
```

Optionally, you can use Innovus to create a flattened database.

TCL commands

Voltus Load and Save Design Tcl Script

```
read_lib -lef tech.lef allcells.lef
read_lib -min stdcells_min.lib memories_min.lib
read_verilog dma.v
set_top_module dma
read_sdc top.sdc
read_def -skip_signal dma.def
read_spef dma.spef
save_design -rc dma.enc
```

In new Voltus session

```
read_design -physical_data dma.enc.dat top_cell_name
```

Running Analysis without Creating a Physical Database

Running the hierarchical flattener or restoring a physical database for very large designs (>10M components) can take a long time and significant resources.

Voltus allows you to pass the physical data of the design (DEFs) directly to the power and rail analysis engine without creating a physical database in the memory. The `specify_def` command allows you to pass DEF(s) directly to the power and rail analysis engines.

© 2013-2023 All Rights Reserved.

Once the analysis is complete, and you want to debug the result in the context of layout, a physical database can be read in the same session using the `read_def` command.

TCL Commands

Voltus Specify Hierarchical Design Tcl Script

```
read_lib -lef tech.lef allcells.lef
read_lib stdcells_min.lib memories_min.lib
read_verilog top.v block1.v block2.v
set_top_module top
read_sdc top.sdc
specify_def top.def block1.def block2.def
read_spef top.spef block1.spef block2.spef
read_power_domain -cpf design.cpf
```

Perform analysis

```
set_power_analysis_mode -method static ..
report_power
set_rail_analysis_mode -method static ..
analyze_rail -type domain ALL
```

Read physical data

```
read_def -skip_signal top.def block1.def block2.def
```

Note: In the `specify_def` mode, static power calculation does not use the specified DEF. Instead, it only uses the verilog netlist to calculate power. In case of a single power domain design, there is no impact on accuracy. In the case of an MSMV design with multiple voltage islands, you must provide the CPF file or logical connections of power pins to power nets using the `globalNetConnect` command to define logical connectivity. Optionally, you can choose to run dynamic power analysis which calculates static power as well. The dynamic power analysis uses DEFs specified by the `specify_def` command. The static and dynamic rail analysis engines use DEFs specified by the `specify_def` command as well.

Running Analysis without Design Loading

Voltus has the ability to run the dynamic power and rail analysis flow without loading a design, thereby, improving the the runtime and memory footprint for the whole flow. To run this flow, you must have an external TWF file as the timing engine will not be invoked from within Voltus. This flow is recommended for large designs (40M +) where front-end processing of SPEF/Verilog/timing is high.

© 2013-2023 All Rights Reserved.

Power Analysis

A sample script for running the dynamic vectorless power analysis flow is given below:

```
read_lib -min stdcells_min.lib memories_min.lib
read_lib -max stdcells_max.lib memories_max.lib
read_twf ./test.twf
specify_def./adder_routed.def
specify_spef ./adder_routed.spef
set_power_analysis_mode -method dynamic_vectorless -disable_static
false -power_grid_library …
set_power ...
set_default_switching_activity ...
set_switching_activity ...
read_activity_file ...
set_power_output_dir ...
set_power_include_file ...
set_dynamic_power_simulation ...
report_power
```

A sample script for running the dynamic vector-based power analysis flow is given below:

```
read_lib -min stdcells_min.lib memories_min.lib
read_lib -max stdcells_max.lib memories_max.lib
read_twf ./test.twf
specify_def./adder_routed.def
specify_spef ./adder_routed.spef
set_power_analysis_mode -method dynamic_vectorbased -disable_static
false -power_grid_library …
set_power ...
set_default_switching_activity ...
set_switching_activity ...
read_activity_file ...
set_power_output_dir ...
set_power_include_file ...
set_dynamic_power_simulation ...
report_power
```

For power analysis, it is recommended to provide either PGV or LEF for all the cells being analyzed. You can specify both LEF and PGV, but the software will only use one set of inputs. PGV has higher precedence over LEF due to the following reasons:

- Macro PGV with the dynamic/static current is required during power analysis so that the triggering event and/or current can be captured for rail analysis

© 2013-2023 All Rights Reserved.

- Powergate PGV contains information that allows power analysis to capture the switch net relationship as well as the powergate enable information for ramp-up analysis.

However, if there are other cells that are missing in PGV but available in LEF, you can specify both LEF and PGV along with the `set_power_analysis_mode -use_lef_for_missing_cells true` command parameter. This parameter specifies to use the LEF definition for cells that are missing in PGV. In case there are common cells between LEF and PGV, then PGV gets precedence.

The following commands and parameters are not supported in this flow:

- `set_power` and `set_switching_activity` will not accept a cell/instance/pin/net/portlist using the `get*` commands. You can use wild card support to give a list of design objects to these commands.
- The `-view` parameter of the `report_power` command.
- The `-analysis_view` parameter of the `set_power_analysis_mode` command are not supported.
- The `write_tcf` and `dump_unannotated_nets` commands are not supported in this flow.

Rail Analysis

Based on the static and dynamic current files generated in the power analysis flow, you can perform rail analysis as given below:

```
set_rail_analysis_mode -method dynamic/static -power_grid_library
...
set_pg_nets -force ...
set_rail_analysis_domain -name ...set_power_data -format ...
set_power_pads -net ...
set_package ...
set_net_group ...
set_advanced_rail_options ...
set_dynamic_rail_simulation ...
analyze_rail -type domain ...
```

The following parameter is not supported in this flow:

The `-analysis_view` parameter of the `set_rail_analysis_mode` command is not supported.

© 2013-2023 All Rights Reserved.

Reading RC Parasitics for Power Calculation

There are multiple options to read RC parasitics for power calculation. These are:

- `read_spef`

  This command reads RC parasitics in the memory and annotates it to the verilog and physical netlist. The SPEF must be read after DEF. Power calculation does not require coupling capacitance in the SPEF. Reading SPEF with coupling capacitance also takes a lot of memory. In such cases, use the `read_spef -decoupled` option. Multiple SPEFs can be read with this command.

- `write_rcdb` and `read_rcdb`

  You can write RC parasitics database in the binary format using the `write_rcdb` command and later read this database directly using `read_rcdb`. The RC database has better runtime and memory performance.

- `read_spef -rc_corner corner_namespef_file`

  When the CPF or MMMC views are loaded, SPEFs must be read after the `set_analysis_view` command.

  ```
  set_analysis_view -setup view1 -hold view1
  read_spef -rc_corner best design.spef
  ```

Working with Hierarchical Databases

Voltus provides a hierarchical database based GUI that allows you get both a flattened top view as well as a hierarchical view of a design. This feature allows you to view instance-level and design-level hierarchies. To create a hierarchical database (HDB), you need to specify a list of LEFs and hierarchical DEFs.

To create and load an HDB, you need to perform the following steps:

1. Start Voltus using the following command:

   ```
   voltus  -files input_file
   ```

2. Specify the `read_lib -lef` and `init_hier_design` commands to load the design (LEF files and hierarchical DEFs).

Note: You can read the hierarchical DEFs using the `read_def` or `init_hier_design` command.

© 2013-2023 All Rights Reserved.

3. Specify the `save_hier_design` command to save a snapshot of the design data, including the binary timing data, timing constraints file, netlist, and the timing analysis state.

4. Specify the `read_hier_design` command to restore all logical design information from a previous hierarchical design session saved in the software.

5. Start the GUI using the `start_gui` command.

   The top-level view is displayed, as shown below:

6. Click the icon on the toolbar to view the HDB menu, as shown blow:

   The menu allows you to select all hierarchical instances at the same time and change the view level. The following snapshots shows level hierarchy for the complete design:

© 2013-2023 All Rights Reserved.

Push down to the bottom level **Pull up to the top level**

7. Right-click on an instance to view the instance-level shortcut menu, as shown below:

The menu allows you to:

- select an instance and push down level by level, or to directly push down to the bottom level
- select an instance and pull up level by level, or to directly pull up to the top level
- set different view levels to different instances

The following snapshots shows level hierarchy for an instance:

© 2013-2023 All Rights Reserved.

Push down to the bottom level

Pull up to the top level

After loading the HDB database, you can load the rail directory and analyze the different rail plots.

TCL Commands:

The following are the Tcl commands for creating an HDB:

```
read_lib -lef tech.lef allcells.lef
init_hier_design -defs $designDir/Data/def/design1.def
$designDir/Data/def/Cpu.def $designDir/Data/def/design2.def
$designDir/Data/def/design3.def .. -skip_signal
save_hier_design saveDB
```

The following are the Tcl commands for restoring a saved HDB and loading the state directory:

```
read_hier_design saveDB Chip
read_power_rail_results -rail_directory ./rail/pd_25C_dynamic_1 -
instance_voltage_window {timing whole} -instance_voltage_method {avg
worst}
```

Importing CPF

Use the `read_power_domain` command to import a Common Power Format (CPF) file.

© 2013-2023 All Rights Reserved.

```
read_power_domain
file_name
[-cpf file_name]
```

The option `-cpf` specifies the name of the CPF file, which contains the power domain definitions. This file uses a `.cpf` extension. The CPF file can be created in Innovus.

For additional information on CPF, see the *CPF User Guide*.

In the absence of CPF, you can define nets, power domains, cell power domain, and its operating voltage using the following commands:

```
set_pg_nets
set_rail_analysis_domain
set_cell_power_domain
```


© 2013-2023 All Rights Reserved.

6

Distributed Processing

- Overview on page 74
- Setting-up Distributed Processing on page 74
- Setting-up Power Analysis in Multi-CPU Mode on page 75
  - Standalone Distributed Power Analysis on page 76
  - Multi-size Heterogeneous Distributed Processing on page 78
- Setting-up Library Generation in Multi-CPU Mode on page 80
- Setting-up IR Drop Analysis in Multi-CPU Mode on page 80

© 2013-2023 All Rights Reserved.

Overview

Multiple CPUs can be used to improve the overall turn-around time of power and IR drop analysis. The run-time improvement may vary depending on multi-CPU configuration, design size, and type. Generally, performance improvement will start to diminish beyond 8 CPUs.

Licensing Scheme

Multi-CPU execution for standalone Voltus is licensed as follows:

- First VTS-L license enables 1 CPU

  Parallel processing is not supported; that is, you cannot stack up two VTS-L licenses to run on 2 CPUs.

- First VTS-XL license enables 8 CPUs

  Each additional VTS-XL license enables 8 additional CPUs, and each additional VTS-MP license enables 8 additional CPUs. VTS-MP takes precedence over VTS-XL in license acceleration if both are available.

Setting-up Distributed Processing

The distributed processing is supported with two modes:

- Local Mode

  In this mode, you can specify the number of CPUs to use on local machine.

  ```
  set_distribute_host -local
  ```

  ```
  set_multi_cpu_usage -localCpu 4
  ```

- Distributed Mode

© 2013-2023 All Rights Reserved.

In this mode, you can specify one or more CPUs to use on network hosts for power-grid extraction. You can also specify the number of CPUs to use on local machine for power analysis and matrix solver.

```
set_distribute_host -rsh -add {pe-opt11 pe-opt12}
```

```
set_multi_cpu_usage -localCpu 4 -remoteHost 2 -cpuPerRemoteHost 2
```

You must have rlogin access to remote host machines. You can specify LSF and SGE queue or custom job submission script for the multi-CPU mode.

```
set_distribute_host
-lsf [-queue queue_name] [-resource resource_string] [-args lsf_arguments] |
-sge [-queue queueName] | -custom -custom_script script
```

Important

All remote hosts or distributed machines in the distributed processing mode should have the same path access to the run directory.

Setting-up Power Analysis in Multi-CPU Mode

Multi-CPU support is available in the static, dynamic vectorless, and vector-based power analysis. Both static and dynamic power analysis supports multi-CPU in the local and distributed modes.

Distributed processing allows faster runtime for hierarchical designs. In the distributed mode, you can use the `set_power_analysis_mode -distributed_setup` command parameter to customize the setup for running power analysis. The use model of this parameter is:

```
set_power_analysis_mode -distributed_setup file
```

The specified file has 4 columns:

```
INST/CELL <name> hier/block <power.inc>
```

where,

- `INST/CELL`: Keyword indicating cell or instance to be run in the distributed mode. Cell can be useful for repeated DEF blocks.
- `name`: Name of the hierarchical block or instance.

© 2013-2023 All Rights Reserved.

- `hier/block`: Keyword indicating hierarchy or block. hier implies that the full hierarchy will be preserved by including the DEFs from top to the block, and block implies that only the lower level DEF will be included
- `power.inc`: name of the power include file. This column is optional.

The unspecified instance/cell will be run with the top level as a single group. Each distributed block/instance should have corresponding TWF and SPEF to improve runtime.

Note: Each hierarchy or block will be run independently from other hierarchies/blocks, thus the external load or cell seen at the pin of the hierarchy/block can be different from flat analysis. If the partition boundary analysis is important, distributed processing is not recommended.

Standalone Distributed Power Analysis

Voltus supports standalone distributed power analysis in both the vector-based and vectorless dynamic power analysis flows. This feature allows you to perform dynamic power analysis and vector profiling in the multi-CPU mode without running rail analysis. The following parameters can be specified to perform standalone distributed power analysis:

- the `-distribute` parameter of the `report_power` command to enable standalone distributed power analysis.

  Note: To run vectoring profiling in the distributed mode, you must specify the `-vector_profile_mode event_based` parameter of the `set_power_analysis_mode` command, as shown below:

  ```
  set_power_analysis_mode -vector_profile_mode event_based

  report_power -distribute
  ```

- the `-partition_twf` parameter of the `set_power_analysis_mode` command to read the input TWF data (flat or hierarchical TWF files), and generate block-level TWFs. This parameter must be specified when the block/partition level TWF files are not available.

The standalone distributed power analysis flow works only on the DEF hierarchy. Before running this flow, you must set the partition (hierarchy or block) using the `set_power_analysis_mode -distributed_setup <file>` command parameter. Each partition is run like an independent block-level analysis on the specified distributed hosts.

© 2013-2023 All Rights Reserved.

TCL Script to Run Dynamic Vectorless Power Analysis in the Local Mode

```
set_distribute_host -local
set_multi_cpu_usage -localCpu 4
read_lib -lef <LEF>
read_lib <timing libraries>
read_verilog <verilog netlist>
read_sdc <timing constraints>
read_def -skip_signal <DEFs>
```

Use `specify_def` for better performance

```
read_spef <RC parasitic files>
```

Option (`-disable_static true`) is available to turn off static power calculation during dynamic power calculation

```
set_power_analysis_mode -reset
set_power_analysis_mode -method dynamic_vectorless -disable_static
true
```

Use 2 dominant cycles and ~50ps resolution

```
set_dynamic_power_simulation -period 8ns -resolution 50ps
set_default_switching_activity -input_activity 0.2 -seq_activity 0.1
-period 10.0
report_power
```

TCL Script to Run Dynamic Vectorless Power Analysis in the Distributed Mode

```
set_distribute_host –rsh –add “m1 m2”
```

##The following command specifies the number of remote hosts and number of CPUs per remote host

```
set_multi_cpu_usage -remoteHost 8 -cpuPerRemoteHost 2 -localCpu 16
```

##The following command specifies that all the blocks included in the `dp_setup.txt` file will be run in parallel

```
set_power_analysis_mode –method dynamic_vectorless -
distributed_setup dp_setup.txt
```

The following is the content of the dp_setup.txt file:

© 2013-2023 All Rights Reserved.

```
INST top.block1 block
INST top.block4 block
CELL block2 block
```

In the above example, 4 power distributed jobs are created for: top.block1, top.block4, hierarchical cell block2, and any other hierarchical instances not specified in `dp_setup.txt` that are run with the top-level.

TCL Script to Run Standalone Distributed Power Analysis

```
set_distribute_host -lsf -queue lnx64  -timeout 500000 -args {-P
resource:test -W 150:00 -o temp} -resource "lnx86_test"
set_multi_cpu_usage -remoteHost 4 -cpuPerRemoteHost 8 -localCpu 16
...
...
set_power_analysis_mode \
  -method dynamic_vectorless \
  -disable_static false \
  -power_grid_library "$DesignDir/../techonly.cl $DesignDir/../
stdcells.cl
  -distributed_setup power_control.txt  \
  -create_binary_db true \
  -partition_twf true \
set_db \
  power_method dynamic_vectorless \
  power_disable_static false \
  power_grid_libraries "$DesignDir/../techonly.cl $DesignDir/../
stdcells.cl
  power_distributed_setup power_control.txt  \
  power_write_db true \
  power_partition_twf true \

report_power -distribute -report_prefix report -output static_power
```

Multi-size Heterogeneous Distributed Processing

Voltus supports multi-size heterogeneous distributed processing using the `-custom_script_list` parameter of the `set_distribute_host` command. This parameter allows you to specify multiple launch strings to specify machines with different memory size in the order of priority, and a maximum allowed job count for each string.

© 2013-2023 All Rights Reserved.

The use model of this parameter is:

```
set_distribute_host -custom -custom_script_list {{{string1} count1}
{{string2} count2} ... {{stringN} countN}}
```

If you request for N number of jobs, the command starts submitting jobs starting from string1, until the corresponding count is reached, at which point it starts submitting the next string in the list. Therefore, the command submits the first "`count1`" jobs with `string1`, the next "`count2`" jobs with `string2`, and so on. It stops when the total submitted jobs has reached the requested number (N).

When running distributed power analysis with heterogeneous machines, Voltus uses the order of jobs specified in the distributed configuration file (`set_power_analysis_mode -distributed_setup file`) to allocate machines. You need to specify the correct order for optimal allocation of machine resources.

TCL Script to Run Multi-size Heterogeneous Distributed Processing

##The following command requests for 32 machines wherein two machines require 512GB memory and 30 machines require 256GB memory:

```
set_multi_cpu_usage -remoteHost 32

set_distribute_host -custom -custom_script_list {{{bsub -R
"OSNAME==Linux && OSREL==EE50 && mem > 500000"} 2}  {{bsub -R
"OSNAME==Linux && OSREL==EE50 && mem > 256000"}} 30}
```

##The following command specifies that all the blocks included in the `dp_setup.txt` file will be run in parallel

```
set_power_analysis_mode –method dynamic_vectorless -
distributed_setup dp_setup.txt
```

The following is the content of the `dp_setup.txt` file:

```
CELL block1 ./power.inc
CELL block2 ./power.inc
CELL graphics.block ./power.inc
CELL block3 ./power.inc
CELL block4 ./power.inc
CELL block5 ./power.inc
...
```

In the above example, block1 will be scheduled on the first 512GB machine, block2 will be scheduled on the second 512GB machine, and the remaining blocks will be scheduled on the 256GB machines.

© 2013-2023 All Rights Reserved.

Setting-up Library Generation in Multi-CPU Mode

The Library Generation flow supports multi-CPU processing to improve the overall run time for large designs. You can enable this flow using the `-enable_distributed_processing` parameter of the `set_pg_library_mode` command.

The flow requires separate Tcl scripts for the macro and standard cells. For macros, you need to specify the `-macros_config_file` parameter of the `set_pg_library_mode` command to provide configuration details for each macro cell per Library Generation run. This is an additional input for macros cell type wherein you can provide macro specific Spice and GDS. The format of the file is:

```
CELL <cell1>
GDS_FILE <gds_file1>
SUBCKT_FILE <subckt_file1>
TRIGGER_FILE <trigger_file_path>
POWER_GATE_PARAMS <parameters>

CELL <cell2>
GDS_FILE <gds_file1>
SUBCKT_FILE <subckt_file1>
...
```

`CELL` is the keyword to specify a new cell that is to be characterized. This is a mandatory keyword. This keyword is followed by optional keywords, like `GDS_FILE` and `SUBCKT_FILE`, that are specific to the cell.

The use model for Library Generation in the multi-CPU mode is:

```
set_distribute_host -local -timeout 240
set_multi_cpu_usage -localCpu 4
set_pg_library_mode -enable_distributed_processing true
-celltype macros|stdcells
-macros_config_file config.file #applicable only for macros
.....
generate_pg_library -output <directory_name>
```

Setting-up IR Drop Analysis in Multi-CPU Mode

Multi-CPU support is available in power-grid extraction and matrix solver to perform IR drop analysis. You can use multiple CPUs for static and dynamic analysis when performing domain-based analysis. Dynamic analysis has better runtime as compared to static analysis.

© 2013-2023 All Rights Reserved.

Distributed processing in both local and distributed (rsh) modes is supported for extraction and matrix solver. By default, matrix solver uses multi-CPUs on the local machine, that is, it relies on local CPUs on the machine where the job is executed. The number of CPUs used on the local machine is controlled by the `set_multi_cpu_usage -localCpu` command. For maximum performance, it is recommended that `set_multi_cpu_usage -remoteHost` is set to number of associated nets in the domain for better load balancing.

To enable distributed solver, you must specify the `-enable_distributed_processing_in_solver` parameter of the `set_rail_analysis_mode` command. You can set up solver to use multiple network hosts using the following distributed processing commands:

```
set_distribute_host -rsh -add {host1 host2 host2 host2 host3 host3
host3}

set_multi_cpu_usage -localCpu 4
```

Multiple specification of the same host using the `set_distribute_host` command will enable solver to use as many CPUs on that machine. You can enable distributed solver when peak memory of the analysis exceeds machine memory in the local multi-threaded mode. In the distributed mode, peak memory on the local machine will be reduced as child processes will be executed on network hosts. However, the master thread on the local machine will continue to consume majority of peak memory even in the distributed mode.

In domain based analysis, extraction jobs for associated nets are distributed across network hosts. The ideal setup for domain based analysis is to use network hosts with multi-CPU for parallel power-grid extraction and matrix solver. Using local mode for domain based analysis will increase memory requirement on the local machine as extraction for all nets will be performed in parallel, e.g. domain with a power and a ground net extracted in local machine in parallel will approximately take twice as much memory on local machine.

The net based analysis should always use local mode.

TCL Script for Domain Based IR Drop Analysis

Parallel extraction for each net using two remote hosts and 2 CPUs on each host

Matrix solver is run in parallel using 8 CPUs on local machine

```
set_distribute_host -rsh -add {pe-opt11 pe-opt12}
set_multi_cpu_usage -localCpu 8 -remoteHost 2 -cpuPerRemoteHost 2

read_lib -lef <LEF>
read_lib <timing libraries>
```


© 2013-2023 All Rights Reserved.

```
read_verilog <verilog netlist>
read_sdc <timing constraints>
read_def -skip_signal <DEFs>
```

Use `specify_def` for better performance

```
set_rail_analysis_mode \
-method dynamic \
-accuracy hd \
-temperature 25 \
-report_msmv_format true \
-save_voltage_waveforms true \
-generate_movies true \
-power_grid_library { \
../lab3_libraries/stdcells_accurate/accurate_stdcells.cl \
../lab3_libraries/memories_accurate/TSDN65LPA1024X32M8F.cl \
../lab3_libraries/memories_accurate/TSDN65LPA128X16M8F.cl}
```

Use `-reset` option to reset previously set power data

```
set_power_data -reset
set_power_data \
-format current \
-scale 1 \
{./dynamic_power_allOn/dynamic_VDD.ptiavg \
./dynamic_power_allOn/dynamic_VSS.ptiavg }

set_pg_nets -net VDD -voltage 1.08 -threshold 0.972 -tolerance 0.3
set_pg_nets -net VSS -voltage 0.0 -threshold 0.108 -tolerance 0.3
set_rail_analysis_domain -name core -pwrnets VDD -gndnets VSS
```

vsrc names must be unique for all nets in the domain,e.g. vdd1 vdd2,vss1 vss2

```
set_power_pads -net VDD -format xy -file ../DATA/dma_mac-VDD.ppl
set_power_pads -net VSS -format xy -file ../DATA/dma_mac-VSS.ppl
analyze_rail -type domain core
```


© 2013-2023 All Rights Reserved.

7

Extensively Parallel Processing

- Introduction to Voltus-XP on page 2
- Key Features and Concepts on page 3
- TCL Command Parameters for Performing Rail Analysis in the XP Mode on page 9
- Setting up and Running Dynamic Power and Rail Analysis on page 10
- Voltus-XP Output Directory Structure on page 16
- Distributed GUI on page 18

© 2013-2023 All Rights Reserved.

Introduction to Voltus-XP

The Voltus-Extensively Parallel (XP) mode employs distributed processing technology for power-grid sign-off at advanced-node process technologies. This mode provides performance improvements of up to 5X, ***works on giga-scale designs***, and enhances the Voltus solution's massively parallel execution with more efficient near-linear performance scalability on thousands of CPUs and hundreds of machines.

When Voltus is enabled in the XP mode, it distributes processing over a large number of machines effectively and provides a high-performance power sign-off solution with reduced processing time and silicon-proven accuracy. The following diagram illustrates the scalable system architecture for massive distribution of work:

The XP mode supports exponential increase in the number of nodes per chip, which requires large machines to run the various flows or to load the GUI. The XP solution is intended for large designs that have demanding requirements for parallel execution.

© 2013-2023 All Rights Reserved.

Key Features and Concepts

The following key features of Voltus-XP make it scalable, easier to use, and faster than the traditional single-machine or distributed processing mode:

- Cloud-based Design Environment
- Partitioned Database and Fully Distributed Flow
- Sub-Flow Data Isolation
- Efficient Visualization of the Output Directory Structure
- Elastic Resource Usage
- Improved GUI Performance
- Enhanced Message Logging

Cloud-based Design Environment

Voltus-XP lets you utilize the cloud platform from Amazon Web Services (AWS) to support cloud-based design flows. The cloud-based design environment allows easy deployment and integration of the entire flow because it is server-less, that is, enables dynamic allocation of machine resources without requiring dedicated resources or persistent storage. The XP mode leverages scalability in cloud computing to support unlimited design size, to quickly increase or decrease machine resources based on demand, and to shrink flow run-times.

© 2013-2023
All Rights Reserved.

Partitioned Database and Fully Distributed Flow

The traditional single-machine or distributed processing runs are not practical for very large designs (up to 500 million instances) because of the following reasons:

- The processing runs require a large amount of memory (~Terabytes)
- The processing runs take days to complete
- GUI loading becomes extremely slow

The XP mode deploys partition-based database to enable fully scalable and massively distributed analysis in every stage of the flow. The partition-based database is stored in the compressed format, therefore taking up less disk space and providing a reduced disk latency. Voltus in the XP mode is able to perform parallel execution in chip-power calculation, grid parasitic extraction, IR drop analysis, EM analysis, and GUI loading by running them on many small machines, as shown below:

© 2013-2023
All Rights Reserved.

Simple Use Model

Voltus XP solution is a major technology evolution compared to the distributed processing solution, but has the same analysis algorithm, use model, and report structure. The XP mode uses the same Tcl scripts (setup and analysis commands) as the distributed processing mode, except for a few additional command parameters that are specific to the XP mode.

Same scripts - no special setup required for running XP

```
set_distribute_host -local
set_multi_cpu_usage -localCpu 16
set_distribute_host -local
set_multi_cpu_usage -localCpu 16 -threadInfo 2

read_lib {/icd/libs/slow.lib_ecsm /icd/libs/pll.lib  /icd/libs/pso_header.lib /icd/libs/pso_ring.lib}
read_lib -lef /icd/lef/gsclib090_tech.lef /icd/lef/gsclib090_macro.lef /icd/lef/pll.lef /icd/lef/
pso_header.lef /icd/lef/pso_ring.lef
read_verilog /icd/design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter
set_dc_sources -power VDD_AO -voltage 0.9 -force
set_dc_sources -power VDD_column -voltage 0.9 -force
set_dc_sources -power VDD_ring -voltage 0.9 -force
set_dc_sources -power VDD_external -voltage 0.9 -force
set_dc_sources -ground VSS -voltage 0.0 -force
read_twf /icd/rak.twf
specify_def /icd/rak.def
specify_spef /icd/design/postRouteOpt_RC_wc_125.spef.gz
set_power -instance u0 -dynamic_switch_pattern {READ 0 READ}

set_distribute_host -lsf -env_script dp.env -queue voltus -timeout 500000 -lsf_args {-P VOLTUS} \
                    -resource {rusage[mem=2000] select[OSNAME==Linux && OSREL==EE60 && ncpus>=16]}
set_multi_cpu_usage -remoteHost 2 -cpuPerRemoteHost 4 -localCpu 16
set_power_pads -format xy -net VDD_AO -file /icd/design/super_filter_VDD_AO.pp
set_pg_nets -net VDD_AO -force -voltage 0.9 -threshold 0.85
set_power_pads -format xy -net VSS -file /icd/design/super_filter_VSS.pp
set_pg_nets -net VSS -force -voltage 0.0 -threshold 0.01
set_power_pads -format xy -net VDD_external -file /icd/design/super_filter_VDD_external.pp
set_pg_nets -net VDD_external -force -voltage 0.9 -threshold 0.85
set_rail_analysis_mode -accuracy hd -method dynamic -temperature 125  -extraction_tech_file /vols/
super_filter/qrcTechFile -enable_xp true
set_power_data -reset
set_power_data -format current -scale 1 {/vols/dynamic_VDD_external.ptiavg /vols/dynamic_VSS.ptiavg}
set_rail_analysis_domain -name core -pwrnets {VDD_AO VDD_external} -gndnets VSS
analyze_rail -type domain -results_directory dynamic_rail core
```

Same commands and options

Additional option to enable XP

Tcl | Tab Width: 8 | Ln 2, Col 1 | INS

Sub-Flow Data Isolation

Voltus-XP allows data isolation for each stage or sub-flow of the main flow, such as extraction, electromigration, and simulation, in the output directory. Each sub-flow has its own data storage location containing reports, logs, and results. It provides a fault tolerant architecture that enables re-use of the output of a sub-flow at a later stage in the flow, helps in quickly debugging and restarting a failed step due to NFS glitches, and resumes from the failed step.

© 2013-2023 All Rights Reserved.

For example, a failed sub-flow step due to low disk space error can be quickly resumed by using the output of the previous sub-flow, without having to re-run all the previous sub-flow steps.

Each sub-flow step has its own log file, which records detailed events during each sub-flow run. Monitoring these log files will help you to easily determine and troubleshoot errors.

Efficient Visualization of the Output Directory Structure

Voltus XP improves usability by recording files generated at each flow step in its corresponding directory. The new output (state directory) structure enhances the way the files are stored and presented when analyzing reports and GIFs for each step. For example, all the rail-related reports are saved in the `/Reports/rail` folder and all the electromigration-related reports are saved in the `/Reports/em` folder within the state directory.

© 2013-2023
All Rights Reserved.

Elastic Resource Usage

The XP mode supports elastic resource usage or elastic scaling by automatically matching your resource requirements to the available resources, and allowing to start rail/EM analysis with partial resources.

Improved GUI Performance

Voltus-XP GUI has improved by enabling GUI launch in the distributed processing mode and generating GUI preparation data (image cache). This has led to smaller memory footprint, faster start-up time, and enhanced rendering of the analysis plots.

Enhanced Message Logging

The message logging method has been advanced to effectively troubleshoot issues at a sub-flow level. The messages logged by Voltus-XP are written to the console and the log files, as depicted in the following diagram:

© 2013-2023
All Rights Reserved.

The following log files are generated during the XP run:

- `voltus.log` - a consolidated log file that contains the complete log details for the entire run. This file is generated in the current working directory.
- `voltus_xp.log` - a log file that is specific to the XP mode. This file helps to track all messages related to each XP sub-flow step. The `voltus_xp.log` file is located in the `/state_directory/logs` sub-directory.

  Each sub-flow step also has its own log file, which records detailed events during each sub-flow run. The sub-flow level log files are located in the corresponding folder (for example, `design_report` and `simulation`) within the `logs` sub-directory.
- `report_generation.log` - a log file that includes the circuit profile (nodes, resistors, current sources, and voltage sources) and rail analysis plot reports for all nets. This file is generated in the `/state_directory/reports/rail/domain` sub-directory.

© 2013-2023 All Rights Reserved.

TCL Command Parameters for Performing Rail Analysis in the XP Mode

The following parameters of the `set_rail_analysis_mode` command are used for performing rail analysis in the XP mode:

- `-enable_xp`
- `-xp_cpu_per_job_power`
- `-xp_cpu_per_job_simulation`
- `-xp_host_allocation_method`
- `-xp_purge`
- `-xp_resume`
- `-xp_reuse_extraction_directory`
- `-xp_reuse_power_directory`
- `-xp_simulation_cpu_timeout`
- `-xp_simulation_min_cpu`

For detailed information on these parameters, refer to *Voltus Text Command Reference*.

© 2013-2023 All Rights Reserved.

Setting up and Running Dynamic Power and Rail Analysis

This section describes how to set up the dynamic power and rail analysis mode and then run rail analysis in the XP mode.

Important

To run Voltus in the XP mode, the following are the mandatory settings:

- enable the XP mode (`set_rail_analysis_mode -enable_xp true`)
- specify the extraction technology file (`set_rail_analysis_mode -extraction_tech_file`) to remove dependency on the technology PGV.

Required Input Files

- Design data (LEF, netlist, DEF)
- SPEF
- Power-grid libraries (.cl)
- Vectorless or vector-based files created during dynamic power calculation
- Quantus extraction technology file

Procedure

To perform dynamic power and rail analysis, complete the following steps:

1. Specify the distributed processing settings. This flow requires at least 8 CPUs.

   Tcl command:

   ```
   set_multi_cpu_usage -localCpu 8 -remoteHost 2 -cpuPerRemoteHost 8
   set_distribute_host -custom -custom_script_list { {{bsub -m "sjf-dsgpe-07" -q hier -n 8  -K} 1} {{bsub -m "sjf-dsgpe-08" -q hier -n 8 -K} 1}}
   ```

   Note: By default, XP starts analysis with partial resources. For example, if you request for 16 hosts but the number of available hosts are only 8, the tool will start the run with 8 hosts.

2. Load the design data.

© 2013-2023 All Rights Reserved.

Tcl command:

```
read_lib -lef $lefs
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_twf ../design/super_filter.twf
specify_def ../design/super_filter.def.gz
```

3. Specify the SPEF file for signal net parasitic information.

   Tcl command:

```
specify_spef ../design/postRouteOpt_RC_wc_125.spef.gz
```

4. Specify the sub-directory that will contain the power results.

   Tcl command:

```
set_power_output_dir            dynVecLessPowerResults
```

5. Define the power analysis mode.

   Tcl command:

```
set_power_analysis_mode \
    -reset
set_power_analysis_mode \
    -disable_static             false \
    -write_static_currents      true \
    -binary_db_name             dynvectorlessPower.db \
    -create_binary_db           true \
    -method                     dynamic_vectorless
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl } \
    -enable_state_propagation true \
    -current_generation_method avg
    -distributed_setup dp_setup.txt
```


© 2013-2023 All Rights Reserved.

Note: It is recommended to specify the `-distributed_setup` parameter that allows you to give a user-defined file to control the number of partitions on which the power analysis run is performed.

6. Define the power of an instance or cell. For dynamic analysis, you can assign a piece-wise linear waveform to an instance.

   Tcl command:

```
set_power -instance u0 -dynamic_switch_pattern {READ 0 READ}
```

7. You can define default activities for the unassigned nets.

   Tcl command:

```
set_default_switching_activity -input_activity 0.3 \
    -period 4.0 -clock_gates_output_ratio   2.0
```

8. Optional. Define the simulation period and resolution.

   Tcl command:

```
set_dynamic_power_simulation -reset
set_dynamic_power_simulation -period 20 -resolution 50
```

   Note: If the resolution is high, the current peaks may be truncated, affecting the accuracy of the analysis. Alternatively, a small resolution may lead to increase in runtime.

9. Define the rail analysis mode.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_rail_analysis_mode\
    -method                     dynamic \
    -accuracy                   hd \
    -temperature                125 \
    -gif_resolution             0 \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    }
    -extraction_tech_file /design/super_filter/qrcTechFile
    -report_power_in_parallel true
    -enable_xp true
```

Important

You need to set the `-report_power_in_parallel true` parameter to run Voltus-XP power analysis in parallel with rail analysis to improve the overall turnaround time.

Note: The additional options listed in the TCL Command Parameters for Performing Rail Analysis in the XP Mode section can be specified in the XP mode.

10. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

Tcl command:

```
set_pg_nets -net VDD_AO     -voltage 0.9 -threshold 0.85
set_pg_nets -net VDD_external -voltage 0.9 -threshold 0.85
set_pg_nets -net VSS        -voltage 0.0 -threshold 0.05
set_rail_analysis_domain -name core -pwrnets {VDD_AO VDD_external}
-gndnets VSS
```

11. Define the location of the voltage sources for each power/ground net.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_power_pads -net VDD_AO -format xy -file
../design/super_filter_VDD_AO.pp
set_power_pads -net VDD_external -format xy -file
../design/super_filter_VDD_external.pp
set_power_pads -net VSS -format xy -file
../design/super_filter_VSS.pp
```

12. Run rail analysis.

Tcl command:

```
analyze_rail -type domain -output dynamic_rail core
```

The Voltus console will display rail analysis statistics, as shown in the following snippet:

```
Rail Analysis completed successfully.


Finished Rail Analysis at 11:56:53 08/20/2018 (cpu=0:00:20, real=0:00:34, peak
mem=1218.10MB)
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:03:56,
real=0:58:07, mem=1218.10
```


© 2013-2023 All Rights Reserved.

Output

The rail output directory will contain the following information:

Directory Name	Description
`<domainName>_<temperature>_dynamic_<X>`	For each run, the state directory is generated in the rail output directory. The directory has the default name `<domainName>_<temperature>_dynamic_<X>` for the dynamic analysis run, and `<domainName>_<temperature>_avg_<X>` for the static analysis run. `<domainName>` will be replaced with `ALL` if all domains are to be analyzed (`analyze_rail -output dir -type domain ALL`), or with `<netName>` if a power net is to be analyzed. The `rail` folder within `<output directory>`/`<state_directory>`/`reports` contains folders for each net, such as VDD and VSS. It also contains a folder called `domain` for domain-based reports. For more information on the rail output directory structure, refer to Voltus-XP Output Directory Structure. *Important* The rail output directory (`analyze_rail -output dir`) contains a symbolic link called `latest` that points to the latest state directory.

© 2013-2023 All Rights Reserved.

Voltus-XP Output Directory Structure

This section describes the rail analysis state directory structure and description of the individual high-level directories:

- **gui** - Stores GUI-related data files in the .json format. These files save information such as design name, layers, nets, partition number, layers mapping to each partition, nets, number of elements, number of nodes, number of instances, plots, and so on.
- **logs** - Stores detailed logs for various rail analysis steps, such as simulation, electromigration (em), and rail report generation (railrpt). It also includes a merged log file called `voltus_xp.log` that contains the complete log details of all the rail analysis steps.
- **misc** - Stores all the miscellaneous files generated by the various rail analysis steps. The miscellaneous files include temporary or application files generated by every step of the flow.
- **pgdb** - Stores all the electrical data used for analysis, such as nodes, elements, instances, tap nodes, voltage sources, tap caps, and what-if scales.

© 2013-2023 All Rights Reserved.

- **reports** - Stores all the report files generated by various steps of the design flow, such as `design`, `em`, and `rail`.
  - The `design` folder contains design check reports, namely, snap layer information for each net, incomplete nets, power-grid view table for each net, shorts, instance floating pins, instance disconnected pins, physically disconnected instances, and missing library models.
  - The `em` folder contains the current density analysis report (*`netname`*`.rj.avg.rpt`) for each net.
  - The `/reports/rail` directory contains the following:
    - net name (for example, VDD, VDD_external, VSS) directories for net-based reports. The net name folder contains the GIFs for the rail plots, and various rail reports (layer-based IR report, power-integrity report, powergate report, RLRP report, and so on).
    - a directory called `domain` for domain-based reports. The `domain` folder contains the IVD GIFs, EIV GIFs, DIV GIFs, and IV reports. For 3DIC designs, another layer of directory with the die name will be created between reports and partition directories.
    - a directory called `package` for package-related information, such as the package terminal mapping file and .ptmf file.
    - a directory called `waveforms` for storing current waveform files (.ptiavg) for each net.

```
[user@machine reports]$ ls
design em rail
[user@machine reports]$ ls rail/
domain     VDD    VSS    package    waveforms
```

- **results** - Stores all the result files generated by various steps, such as simulation, em, and rail. Each folder contains directory for each partition and a top-level directory.
- **voltus_data** - Contains the run directory for each rail analysis step. This folder contains the binary files generated and used throughout the XP run. It includes detailed information, such as reports, logs, and results, that can be used for debugging purpose.

© 2013-2023 All Rights Reserved.

Distributed GUI

Voltus XP in the GUI mode is called Distributed GUI or DGUI. DGUI allows you to load very large designs with billions of instances. The DGUI operations are distributed on multiple hosts to reduce execution time, reduce memory consumption per host, and faster loading and analysis of large designs.

To start Voltus XP in the GUI mode, run the software in the distributed mode using the following commands:

- `set_multi_cpu_usage`
- `set_distribute_host`

After specifying the distributed processing commands, load the design and read the XP output rail directory.

Important

You must specify the distributed processing commands again for DGUI before specifying the `read_power_rail_results` command, even if you have run Voltus-XP with the distributed processing commands in the non-GUI mode.

DGUI is supported for both the flat and hierarchical database (HDB). You can load the HDB and read the rail directory using the `read_power_rail_results` command. This method is recommended for very large designs.

For information on generating and loading a hierarchical database, refer to the *Working with Hierarchical Databases* section in the "**Importing Design**" chapter of *Voltus User Guide*.

Example for Running DGUI for a Flat Database:

Run Voltus-XP in the non-GUI mode

...
##Run Voltus-XP in the DGUI mode

```
set_distribute_host -lsf  -queue interactive -timeout 500000 -args {-P VOLTUS} -
resource {rusage[mem=20000] select[OSNAME==Linux && (OSREL==EE60) && ncpus>=2]}
set_multi_cpu_usage -remoteHost 4 -cpuPerRemoteHost 4 -localCpu 16
start_gui
read_power_rail_results -rail_directory ./dynamic_rail/latest/
```

Note: Currently, DGUI supports viewing only the rail analysis plots.

© 2013-2023 All Rights Reserved.

When the design is being loaded, the following “Caching images” message box appears indicating that GUI loading is in progress:

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

8

Design Sanity Checks

- Check Design on page 22
- Check Timing Library on page 25
- Verify Connectivity on page 26
  - Before you begin on page 26
  - Types of connectivity violations reported on page 26
- Verify Power Via on page 29
- Verify Power and Ground Shorts on page 31
  - Before you begin on page 31
  - Types of short violations reported on page 31
- Violation Browser on page 32
  - Violation Browser Features on page 34
  - Clearing Violations on page 35
- Design Browser on page 36

© 2013-2023 All Rights Reserved.

Check Design

You can check the design at various stages of the design process and can check for missing or inconsistent library and design data.

You can check for the following data:

- Physical library
- Timing library
- Netlist
- I/Os
- Tie-high and tie-low pins
- Power and ground pins

Cadence recommends that you check libraries and data as follows:

- Perform I/O checking at any time. I/O problems might not impede any tool, but they might add to design problems.
- Perform netlist checking at any time after the design has been loaded.
- Perform physical library checking before floorplanning.
- Perform power and ground checking before routing and extraction, and verifying geometry and connectivity.
- Perform timing library checking before any timing-related operation (for example, timing-driven placement or routing, timing optimization, clock-tree synthesis, and static timing analysis).
- Perform tie-high and tie-low checking before routing and extraction.

You can check the design using the following steps:

1. Select *File->Check->Check Design* menu in GUI.

© 2013-2023 All Rights Reserved.

The following form appears:

2. Select the appropriate *Check Design Options*.
3. Select the appropriate *Report Format*.
4. Specify the output directory.
5. Select *Apply* or *OK*.

TCL Command:

The TCL command for checking the design is as follows:

```
check_design -out_file checkDesign -type {power_intent timing
hierarchical pin_assign budget assign_statements place opt cts route
signoff all}
```


© 2013-2023
All Rights Reserved.

Sample Output

The following steps are needed to view check design report.

1. Generate the report using the `check_design` command or GUI.
2. Select *Tools* -> *Report*.
3. Load the report using *File* - *Open File* from the check design output directory.

Alternatively, you can look at the text file if the report was saved as text.

© 2013-2023 All Rights Reserved.

Check Timing Library

The Check Timing Library program reports timing library cells with missing internal or leakage power data.

The following steps are for creating a timing library cell report:

1. Select *File->Check->Timing Library*.

   The following form appears:

2. Specify the cells that you want to report on.
3. Specify *Timing Library Report* filename.
4. Click *OK*.

TCL Command:

The TCL command for checking a timing library is as follows:

```
check_library -outfile design.talib -checkpower -
reportMissingPowerOnly
```


© 2013-2023
All Rights Reserved.

Verify Connectivity

Verify the connectivity of your design to detect and report conditions such as opens, unconnected pins, dangling wires, loops, and partial routing. You can use the command to create violation markers in the design window and generate a violation report. There is no database impact from using this command unless you save the design, which saves the violation markers.

For regular wires, the software checks connectivity by using the center line of the wire segments and center of the vias. For special wires, the command checks the whole geometry. If a via or wire is slightly offset from where it should be, the software reports an error.

The software also detects connectivity loops based on the end points of a regular wire segment center line or the center of a via. It reports geometry loop violations.

Before you begin

Before you verify connectivity, the following conditions must be met:

- The design must be routed.
- The design must be loaded into the current Voltus session.

Types of connectivity violations reported

- Antennas - Unconnected wires (dangling wires).

  To avoid antenna violations, wires and nets must meet the following conditions:

  - Regular wires

    Must terminate on a pin or the center of a via. If a vertical wire intersects a horizontal wire along its axis, the end of the vertical wire must be covered by the horizontal wire. If the ends of a vertical wire and horizontal wire meet, they must meet at their end points.

  - Regular net vias

    Must be covered by a pin. The center of the via must be the start or end point of a wire. The center point of stacked vias must be coincident.

  - Special wires

    A point that is one-quarter of a wire width from the center of the ending edge of the special wire must be covered by a via, pin, or another wire on the same layer.

© 2013-2023 All Rights Reserved.

- Special net vias

  The metal rectangle of the special net via must overlap with a special wire or via of the same net.

- Opens

  Parts of nets, such as wires or pins, that are connected to each other but are missing a connection to the net as a whole. Marks each part of a net that is missing a connection as an open and displays a violation marker between the parts.
  Violation markers for opens are displayed as polygons that include all wires, pins, and via that connect to an island.

- Loops

- Unconnected pins

  Pins that are not connected to any other objects

The following steps are needed to check verification connectivity:

1. Select *Verify*-> *Connectivity* in the GUI.

© 2013-2023 All Rights Reserved.

The following form appears:

2. Specify the type of nets
3. Specify which nets
4. Specify the types of connectivity checks you want to perform
5. Specify the report output file name
6. Specify the maximum number of errors and warnings you want to report.
7. Click *Set Multiple CPU* to specify multi-threading settings for verifying connectivity.
8. After verifying connectivity, you can use the Violation Browser for interactive viewing and highlighting of violation markers.

© 2013-2023 All Rights Reserved.

TCL Command:

The TCL command for doing verification of connectivity is as follows:

```
verify_connectivity -type special -error 1000 -warning 50 -report
dma_mac.conn.rpt
```

Verify Power Via

Verify Power Via program does the following on the power vias:

- Checks for metal geometry overlap on PG nets
- Ignores metal fill geometries by default
- Supports stacked power via check
- Generates text report with location of missing via on PG net
- Highlights violations in the layout canvas
- Checks missing vias in a user-specified region
- Checks missing vias on non-orthogonal overlap
- Checks missing vias on wire-pin overlap
- Checks via utilization in metal overlap

Violations found by `verify_power_via` are highlighted in the layout canvas with cross-marks. They can be queried by selecting and pressing "Q" on the keyboard.

© 2013-2023 All Rights Reserved.

The *Verify Power Via* form can be executed in the GUI from the *Verify-> Power Via* menu.

For more information on the fields and descriptions, see *Voltus Menu Reference*.

TCL Command:

The TCL command examples are as follows:

- `verify_power_via -report power_via.report`

  Checks missing vias on all power nets and all metal overlaps.

- `verify_power_via -layer_range "M5 M8" -nets VDD`

  The `-layer_range "M5 M8"` option checks for the missing stacked vias between layer M5 and M8. Use the toolbar button called *Clear DRC Violations* in *Layout* tab to clear the violation markers.

- `verify_power_via -via_util 0.7 -report via_utilization.rpt`

  Checks whether via utilization is 70% on metal overlap.

© 2013-2023
All Rights Reserved.

Verify Power and Ground Shorts

You can verify power and ground shorts in your design and report shorts between two geometries belonging to different nets. Support for power-ground short detection lets you perform power and ground short checks between the following:

- PG and PG nets
- PG and signal nets
- PG and other special nets

Use the `verify_PG_short` command to create violation markers in the design window and generate a violation report. Then, use the *Violation Browser* for interactive viewing and highlighting of violation markers.

Before you begin

Before you verify shorts, you must restore the physical design using the following command:

```
read_design -physical_data directory topcell or read_def filename
```

Types of short violations reported

The following figures illustrate short violations and their description:

```
Pin of Cell tt1/sram1 & Blockage
```


© 2013-2023 All Rights Reserved.

```
False : No Layer : M1 Bounds (-14.140, -119.990) (-13.730, -119.990)
```

```
Special Wire of Net GND & Blockage

False : No Layer : M6 Bounds (-575.800, -2418.000) (-570.450, -
2413.945)
```

TCL Command:

The following TCL command checks short violations on the power net VDD:

```
verify_PG_short -net { VDD}
```

The Verify Power Ground Shorts feature supports multi-threading. To enable multi-threading, you must specify the `set_multi_cpu_usage -localCpu` command before specifying `verify_PG_short`, as shown below:

```
set_multi_cpu_usage -localCpu 4
verify_PG_short
```

The following TCL command checks short violations in the specified area:

```
verify_PG_short -area {-1045.174 -2848.612 -284.521 -2038.7 }
```

Violation Browser

Use *Tools* > *Violation Browser* form to view and highlight violation and lithography hotspot markers interactively.

© 2013-2023 All Rights Reserved.

The *Violation Browser* overwrites violation markers from the `verify_connectivity`, `verify_power_via` and `verify_AC_limit` commands if they are run more than once during a Voltus session. These commands are net-based, not area-based, so the browser does not make incremental updates for connectivity, process antennas, or AC limit. As a result of this behavior, the software does not keep the information from the first verification run.

For more information on the fields and descriptions, see the *Voltus Menu Reference*.

© 2013-2023 All Rights Reserved.

Violation Browser Features

- Click a violation on the violation list on the form to see a description of the violation. The description includes actual and target values for AC limit violations, process antenna violations, and geometry spacing violations.
  - An actual value is the current value
  - A target value is the value defined in the LEF file.
- Click the + or - sign to collapse or expand the listings of each violation type.
- Use the *First*, *Previous*, *Next*, *Last*, *Up*, and *Down* buttons to navigate through the list of violations.
- The browser displays the violations in the following hierarchical order:

  + tool

  + type

  + subtype

  Description

  where the tool, type, and subtype value correspond to the value you specify using the `createMarker` command.
- Use cross probing between the design display area and the *Violation Browser*.

To display the details of a violation in the *Violation Browser* form, double-click the violation marker in the design display area.

- If there are violation markers for overlapped objects, select the top-most marker in the design display area and press the space bar on your keyboard to navigate through the other markers. The type and name of the selected violation is displayed in the lower-left corner of the window. Use the q keyboard shortcut key to select a violation and highlight it in the *Violation Browser* form.
- Use the Zoom buttons to change the magnification level of a violation.
- Use any of the following buttons to change the display for a selected violation:
  - Highlight Color
  - Highlight Violations
  - De-Highlight Violations
  - Delete Violations

© 2013-2023 All Rights Reserved.

- ❑ Mark Violations as False
- ❑ Mark Violations as True

- Generate a report file by clicking the Save button.

The report file includes information on the violations shown in the violation browser.

- Limit the number of violations to display by using the Show Types panel in the Settings page of the form.
- Limit the area to display by using the Show Area panel in the Settings page of the form.
- Filter the violations to display by using the Other Filters panel in the Settings page of the form.

Clearing Violations

Use *Clear DRC Violations* widget in the toolbar to clear violations.

There is no GUI form for this command.

© 2013-2023
All Rights Reserved.

Design Browser

The design browser is available from GUI under *Tools->Design Browser* menu. See the **Design Browser** section in the *Voltus Menu Reference* for details.

© 2013-2023
All Rights Reserved.

9

Early Power-Grid Analysis

- Overview on page 38
- Early Rail Analysis Key Features on page 41
- Setting up and Running Early Rail Analysis on page 43
- Viewing Early Rail Analysis Results on page 50
- TCL Command Examples on page 51

© 2013-2023 All Rights Reserved.

Overview

The Early Rail Analysis (ERA) feature provides rail analysis at the early stage of design with the same use model as Signoff Rail Analysis. Static ERA can also be run with the Innovus base license. ERA is not intended for sign-off; if you want to include custom power-grid views or do dynamic analysis, then a Voltus license is required. ERA is intended for the early stage analysis and can be run on a power-grid at the floorplan stage, partially placed, and fully placed and routed designs.

ERA has the ability to analyze power-grid integrity early in the floorplanning stage, after placement, as well as postrouting. It helps fix power-grid problems early in the flow, rather than waiting for when the layout is mostly done and the problems are much more difficult to correct. ERA will take whatever blocks, macros, standard cells, and routing that is available to help improve the accuracy of early rail analysis. The ERA flow allows you to specify various power constraints interactively during the power planning stage and quickly perform rail analysis to estimate whether the design meets these constraints. ERA does not need fully routed power-grid network. It has the ability to create virtual power connections to the placed instances based on logical connectivity. This allows you to refine the placement for the macros to achieve better power distribution and achieve the IR drop goals for the design.

The following diagram illustrates the ERA flow:

© 2013-2023
All Rights Reserved.

The following steps describe the ERA flow:

1. **Grid Completion**: The ERA engine checks if the follow pin routing has been done or not in the design; if not, it creates virtual follow pins and drops the required virtual vias. Missing virtual vias between stripes are also dropped. Refer to Additional Examples for Tcl command script.

2. **Power Estimation and User-Defined Distribution**: During ERA, you have the flexibility to specify power for the various modules. The three common methods to specify power for a design are:

   - Region Based Power Specification - This method is used for a fully unplaced design. You can specify power based on the region and layer interactively in the GUI or command file. The specified power is distributed evenly on the specified layer in the given region, as shown below:

The region-based power specification can be used in conjunction with the other methods. Refer to Example 1: Region-based Power Specification for Tcl command script.

   - Total Power and Text Power File - This method is used for a partially placed design. You can specify the total power consumption for the design, and optionally you can specify the estimated (or known) power consumption for the cell or macros: In addition, you can always specify the distribution of the total power using region based power specification. This method can be implemented using the following two options:

     - Option 1: Specify user-defined current region and user-defined cell power file. Refer to Example 2: User specified Current Regions and Instance Power File for Total Power for Tcl command script.

© 2013-2023 All Rights Reserved.

- Option 2: Use tool-generated current region and tool-generated power file for placed instances. Refer to Example 3: Auto generated Current Regions and Instance Power File from Total Power for Tcl command script.

The following example illustrates the total power consumption for a design:

Here, power consumption for the known placed instances can be specified for better accuracy. In the unplaced cell region, current region is specified by interactively drawing regions in the GUI and specifying current on the desired layer.

- Power Engine Calculated Power and Instance-based Power - If a design is fully placed, you can calculate power by running the `report_power` command, or using GUI (*Power & Rail -> Run Power Analysis* menu). The power is computed using the synthesized verilog netlist so it will be computed for all the instances in the design. Refer to Example 4: Power Engine Calculated Power for Tcl command script.

3. **Static IRdrop and EM Analysis**: Run static IRdrop and electromigration analysis.
4. **Dynamic Analysis**: Run dynamic IRdrop analysis. For this, you need to specify explicit dynamic current region file or the dynamic instance current file.

© 2013-2023 All Rights Reserved.

ERA uses Voltus extractor and rail analysis engines. They can be used during power-grid prototyping to analyze power, IRdrop and power-grid integrity.

Early Rail Analysis Key Features

- Interactive current specification use-model to enable static and dynamic rail analysis at the floor-planning stage.
- Power and rail analysis during the placement stage using ERA driven virtual follow pin routing and virtual via for grid completion.
- Automatic current region generation accounting for unplaced instance in the design, and automatic area based distribution of power among placed instances in the design.
- PGV library is optional. If not provided, it is generated on the fly using the specified technology file.
- Static Power and Rail Analysis without PGV can also be run using the Innovus license.
- Power and rail analysis on a placed and routed database.
- Early power-switch analysis to refine power-switch placement.
- Support for multi-CPU in static/dynamic power and rail analysis, and static and dynamic power and rail analysis. For information on how to set up multi-CPU analysis, see the "Distributed Processing" chapter in *Voltus User Guide*.
- Support for the unplaced flow during static and dynamic analysis.
- What-if shape analysis to guide power-grid optimization.
- Support for native power-up analysis.

© 2013-2023 All Rights Reserved.

Prior to Running Early Rail Analysis

- The design must have power stripes for a net to be analyzed. If vias are not dropped among stripes, the ERA grid completion engine can create virtual connections.
- Instances must be logically connected to the PG net under consideration. Instances need not be placed.
- The design must be loaded into memory for virtual follow pin routing and virtual via completion.
- If the design has no instances, a single current region can be automatically created with all current attached to it. However, when assigning the single current region to a layer, the layer should have atleast a single tap/sink current for ERA to run the correct IR analysis.

© 2013-2023 All Rights Reserved.

Setting up and Running Early Rail Analysis

This section describes how to set up and run the early rail analysis mode and then run rail analysis.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- SPEF File
- Power-grid libraries (.cl). See Power-Grid Library Generation for details on generating power-grid libraries.
- Extraction Technology File

Procedure

To perform early rail analysis, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

```
read_lib -lef $lefs
read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz
```

2. Specify the CPF file for power domain information through the command line or through the GUI.

   GUI: Select *File - Load/Commit CPF*.

   Tcl command:

```
read_power_domain -cpf ../design/super_filter.cpf
```


© 2013-2023 All Rights Reserved.

3. Specify the SPEF file for signal net parasitic information through the command line or through the GUI.

   GUI: Select *File - Read SPEF*.

   Tcl command:

```
read_spef -rc_corner  RC_wc_125 -decoupled\
../design/postRouteOpt_RC_wc_125.spef.gz
```

4. Define the early rail analysis mode through the command line or through the GUI.

   GUI: Select *Power & Rail - Set Rail Analysis Mode*. For more information, refer to Setting Up the GUI to Run ERA.

   Tcl command:

```
set_rail_analysis_mode \
    -method                     era_static \
    -accuracy                   xd \
    -analysis_view              AV_wc_on \
    -extraction_tech_file      ../data/qrc/gpdk090_9l.tch \
    -temperature 125 \
    -era_current_region_file   ../tcl/VSS_1.curRegion
```

Note:

- The xd accuracy mode is used for the early implementation stage IR analysis.
- You have the option to use the extraction (Quantus) technology file without power-grid library, or use the power-grid library without the extraction technology file.
- The current region file specifies a list of regions and the amount of current (mA) to be distributed within a power-grid region. This parameter is to be used when a design is either fully unplaced or partially placed. Following is the format of the current region file:

```
#Format:
LABEL name NET netName AREA x1 y1 x2 y2 LAYER layername <CURRENT value |
PWL (t1 i1 t2 i2 …)> INTRINSIC_CAP <value> LOADING_CAP <value> [ADD_OVERLAP
| SUBTRACT_OVERLAP]
```

  Static Example:

```
label test1 net VDD area 100 200 400 500 layer M1 current 10
```

  Dynamic Example:

© 2013-2023 All Rights Reserved.

```
label test1 net VDD area 100 200 400 500 layer M1 pwl (0ns 0mA 1ns 0mA 1.9ns
0mA 2ns 10mA) intrinsic_cap 10 loading_cap 60
```

- You can specify the following parameters for virtual follow pin insertion:
  - `-era_insert_virtual_followpins`
  - `-era_insert_virtual_followpin_for_io`
- You can specify the following parameters for virtual via insertion:
  - `-era_skip_virtual_via_by_type`
  - `-era_insert_virtual_via_on_layers`
  - `-era_skip_virtual_via_on_layers`

For information on the ERA parameters, refer to the `set_rail_analyis_mode` command in the *Voltus Text Command Reference*.

5. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets.

   Tcl command:

```
set_pg_nets -net VSS        -voltage 0.0 -threshold 0.05
set_pg_nets -net VDD        -voltage 1.8 -threshold 1.62
set_rail_analysis_domain -name PD1 -pwrnets VDD -gndnets VSS
```

   Note: The `set_rail_analysis_domain` command is required for domain-based analysis. It is not required for the net-based analysis.

6. Define the location of the voltage sources for each power/ground net.

   Tcl command:

```
set_power_pads -net VSS -format xy \
-file ../design/super_filter_VSS.pp
```

7. Run early analysis through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Run Rail Analysis*.

   Tcl command:

```
analyze_rail -output ./era_run -type net VSS
```

The Voltus console will display rail analysis statistics, as shown in the following snippet:

© 2013-2023 All Rights Reserved.

```
Rail Analysis completed successfully.
Finished Rail Analysis at 10:22:02 09/27/2018 (cpu=0:00:09, real=0:00:22, peak
mem=1075.38MB)
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:01:32,
real=0:18:06, mem=1075.38MB)
```

Output

The early rail output directory will contain the following information:

Directory Name	Description
`<domainName>_<temperature>_avg_<X>`	For each run, an output directory is generated with the default name `<domainName>_<temperature>_avg_<X>`. `<domainName>` will be replaced with `ALL` if all domains are to be analyzed (`analyze_rail -output dir -type domain ALL`), or with `<netName>` if a power net is to be analyzed. The output directory contains state directories for each net, such as VDD and VSS. Each state directory contains the following: ■ `results` – a file that gives a summary of the rail analysis results for various plots, such as tc (tap current), rc (resistor current), and so on. ■ `Reports` – a folder that contains a list of useful report files that contain all power-grid integrity data. ■ Auto-generated GIF files for the rail analysis plots

© 2013-2023 All Rights Reserved.

Setting Up the GUI to Run ERA

To set up early rail analysis using GUI, perform the following steps.

1. Select *Power & Rail - Set Rail Analysis Mode* menu.

   The *Set Rail Analysis Mode* form appears:

   Note: The content of the two tabs of this form change depending on the selection of the analysis method.

2. Set *Analysis Stage* as *Early*.

3. Set *Analysis Method* as *Static* or *Dynamic*.

   By default, *XD* is selected as the *Accuracy* mode.

4. Specify *Power-Grid Libraries* or *Extraction Tech File*. See Power-Grid Library Generation for details on generating power-grid libraries.

5. Click the *Advanced* tab of the *Set Rail Analysis Mode* form.

6. Select the *Specify Current Region* check box to specify a file that includes a list of regions and the amount of current to be distributed within them for the power-grid. When you select this check box, the *Create* button gets enabled that lets you create current regions for Static and Dynamic analysis.

7. Click the *Create* button.

   The *Create Current Region* form appears.

8. Specify the current region coordinates in the Create Current Region form.

   The steps for obtaining the coordinates for **domain-based** current regions are:

   a. Select the *Power Domain* check box.

   b. Select a power domain from the drop-down list.

© 2013-2023 All Rights Reserved.

c. Click *Get Coordinates* to pick the domain coordinates automatically. The form appears as:

The steps for obtaining the coordinates for **net-based** current regions are:

a. Select a net from the *Net* drop-down list.

b. Specify a label for the net in the *Label* field, and click *Draw*.

c. In the design layout, select an area.

© 2013-2023
All Rights Reserved.

This form will automatically display the coordinates of the selected area in the *x1*, *y1*, *x2*, *y2* fields. The form appears as:

9. Select a layer from the *Layer* drop-down list.
10. Enter the current for the region in the *Static Current (mA)* field.
11. Click the *Add* button to add the current region to *Current Region List*.
12. Click the *Save* button to save it to the current region file. For example, VSS_1.curRegion.
13. Repeat the steps 8-13 to create more current regions.
14. Click *OK* to close the Create Current Region form.
15. Click *OK* to save and apply the early rail analysis setup information.

© 2013-2023
All Rights Reserved.

The following fields in the *Advanced* tab are enabled only if you select *Early* analysis stage in the *Set Rail Analysis Mode - Basic tab*:

Viewing Early Rail Analysis Results

To view the early rail analysis results:

1. Open the *Power & Rail Setup* form, accessed by clicking the *DB Setup* button on the *Power & Rail Plots* form.
2. Load the early rail database by clicking the *Browse* button next to the *Rail Database* field.
3. Click *OK*.
4. Plot the results using the rail analysis plot type *ir* (`set_power_rail_display -plot ir`).

   You can use the *Power & Rail Plots* form to browse to other analysis runs and load them as well to view early analysis results and compare runs. You can specify the type of plot (Rail Analysis, Power Analysis, or Capacitance) and then select the specific plot type.

   For Early Rail Analysis, the viewing of *Power & Rail Plots* is the same as that used for Sign-off Analysis. For additional information on viewing the plots, see "Viewing Static Rail Analysis Results" in the *Voltus User Guide*.

© 2013-2023 All Rights Reserved.

TCL Command Examples

Example 1: Region-based Power Specification

An example TCL command for the Floorplan stage design grid analysis with current regions is as follows:

```
read_lib -lef  design/full.lef
read_verilog design/test.v.gz
set_top_module test
read_def design/full.def
```

**set_rail_analysis_mode -method era_static -accuracy xd **
-extraction_tech_file design/tech.tch -era_current_region_file design/current_region

```
set_pg_nets -net VDD -voltage 1.1 -threshold 1.067
set_power_pads -net VDD -format xy -file pads/vdd.pp
set_power_data -format area -power 1 -bias_voltage 1
analyze_rail -type net -output early_vdd VDD
```

Example 2: User specified Current Regions and Instance Power File for Total Power

An example TCL command for partially placed design grid analysis with user-defined macro power file is as follows:

```
set_pg_nets -net VSS -voltage 0.000 -threshold 0.020
set_power_pads -net VSS -format xy -file pads/VSS.pp
```
set_power_data -format area -bias_voltage 1 -power 1.5\ power/instance_power_file.txt

**set_rail_analysis_mode -method era_static -accuracy xd **
-extraction_tech_file design/tech.tch -era_current_region_file design/current_region

```
analyze_rail -type net -output user_power_analysis VSS
```

Example 3: Auto generated Current Regions and Instance Power File from Total Power

An example TCL command for partially placed design grid analysis with tool-generated power file is as follows:

set_power_data -format area -power 1.5 -bias_voltage 1

**set_rail_analysis_mode -extraction_tech_file tech.tch -accuracy xd -ignore_shorts true **
**-method era_static -era_current_distribution all -era_current_distribution_layer Metal1 **
-era_current_distribution_unplaced_area diearea

© 2013-2023 All Rights Reserved.

```
##When -era_current_distribution is set to all, the software will generate both the
instance power file (for placed instances) and the current region file (for unplaced
instances).
set_power_pads -net VDD_AO -format xy -file design/super_filter_VDD_AO.pp
set_power_pads -net VDD_SW -format xy -file design/super_filter_VDD_external.pp
set_power_pads -net VSS -format xy -file design/super_filter_VSS.pp
set_pg_nets -net VDD_AO -voltage 0.9 -threshold 0.85  -force
set_pg_nets -net VSS -voltage 0.0 -threshold 0.05  -force
set_rail_analysis_domain -name core -pwrnets { VDD_AO VDD_SW } -gndnets VSS
analyze_rail -type domain -output era_power_analysis core
```

Example 4: Power Engine Calculated Power

An example TCL command for fully placed design grid analysis is as follows:

```
set_power_analysis_mode -method static -write_static_currents true
set_default_switching_activity -global_activity 0
set_dynamic_power_simulation -period 70ns -resolution 50ps
report_power
set_power_data -format current -scale 1 "static_power/static_VDD_AO.ptiavg\
static_power/static_VDD_SW.ptiavg static_power/static_VSS.ptiavg"

set_rail_analysis_mode -power_grid_library tech.pgv -accuracy hd \
-ignore_shorts true -method era_static
set_power_pads -net VDD_AO -format xy -file design/super_filter_VDD_AO.pp
set_power_pads -net VSS -format xy -file design/VSS.pp
set_pg_nets -net VDD_AO        -voltage 0.9 -threshold 0.85  -force
set_pg_nets -net VSS           -voltage 0.0 -threshold 0.05  -force
set_rail_analysis_domain -name core -pwrnets { VDD_AO VDD_SW } -gndnets VSS
analyze_rail -type domain -output power_engine_analysis core
```

Additional Examples

- An example TCL command for Power Gate Design with auto area based power distribution for remaining placed and unplaced instances. Here user specified current regions and macro power is also specified for accounting for some instances of the design as follows:

```
read_design -physical_data design.dat CHIP
set_pg_nets -net VDD -voltage 1.1 -threshold 0.9
set_power_pads -net VDD -format xy -file design/vdd.pad
set_power_data -bias_voltage 1.2 -power 1.2 -format area\
power/macro_power_file.txt
set_rail_analysis_mode -method era_static -accuracy xd \
-extraction_tech_file design/tech.tech -era_power_gate_file design/power_gate_file\
-era_current_region_file design/user_current_regions -era_current_distribution all \
-era_current_distribution_layer M1
analyze_rail -type net -output mixed_power_analysis VDD
```


© 2013-2023 All Rights Reserved.

- An example TCL command for static analysis, where ERA is also doing follow pin routing and virtual via generation is as follows:

```
read_design -physical_data design.dat CHIP
```

**set_rail_analysis_mode -method era_static -accuracy xd -extraction_tech_file tech.tch \
-era_current_distribution_layer MET1 -era_current_distribution unplaced \
-era_current_region_file power/cur_region -era_insert_virtual_followpins standard \
-era_skip_virtual_via_by_type none**

```
set_pg_nets -net VDD -voltage 1.1 -threshold 1.067
set_power_data -format area -power 0.744 -bias_voltage 1.2\
power/cell_power.txt
set_power_pads -net VDD -format xy -file pads/vdd.pp
analyze_rail -type net -output ./early_vdd VDD
```

- An example TCL command for MSMV dynamic analysis is as follows:

```
read_verilog design/test.v.gz
set_top_module test
read_def design/full.def
set_pg_nets -net VSS -voltage 0 -threshold 0.18
set_pg_nets -net VDDm -voltage 0.84 -threshold 0.756
set_pg_nets -net VDD -voltage 0.84 -threshold 0.756
set_power_pads -net VDD -format xy -file design/vdd.pp
set_power_pads -net VDDm -format xy -file design/vddm.pp
set_power_pads -net VSS -format xy -file design/vss.pp
set_power_data -format current { instance_current_files/dynamic_VSS.ptiavg\
instance_current_files/dynamic_VDD.ptiavg\
instance_current_files/dynamic_VDDm.ptiavg\
instance_current_files/dynamic_VDDlu.ptiavg\
instance_current_files/dynamic_VDDau.ptiavg}
```

**set_rail_analysis_mode -method era_dynamic -accuracy xd \
-power_grid_library { stdcells_accurate/accurate_stdcells.cl \
lpcells_accurate/accurate_stdcells.cl memories_accurate/MEM.cl } -off_rails VDDau**

```
set_rail_analysis_domain -name PD -pwrnets {VDD VDDm} -gndnets VSS
analyze_rail -type domain PD
```


© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

10

Power-Grid Library Generation

- Overview on page 57
  - Inputs on page 58
  - Outputs on page 59
  - Types of Power-Grid Libraries on page 60
  - Power-Grid View Types on page 60
- Generating a Technology Library on page 63
- Generating Standard Cells Power-grid Library on page 66
- Creating a Power-grid View for a Macro Cell on page 70
- Characterizing Current Waveforms for Macro EM Views on page 75
  - Trigger File Format - for simulation vector-based files on page 86
  - Trigger File Format - for user-defined PWLs on page 87
  - Trigger File Format - FSDB on page 91
  - Trigger File Format - Datasheet on page 93
  - Trigger File Format - for user-specified Spice Deck on page 96
  - Trigger File Format - Dotlib on page 97
  - Sample Library Generation Output Report on page 100
  - Power-Grid View Generation Flow for Macros from PGDB/xDSPF on page 101
  - Multi-Mode Power-Grid View Generation on page 103
  - Additional Reports on page 107
- Multi-Voltage Capacitance Characterization on page 109
- Characterizing Decap, Filler, and Damping Decap Cells on page 111

© 2013-2023 All Rights Reserved.

- Generating Power-Grid Library for a Flip Chip Design on page 112
- Generating Open Access Based Library on page 115
- Checking PowerGrid Library and Report on page 117
- Viewing and Debugging PowerGrid Library on page 119
- Validating a PowerGrid Library on page 126

© 2013-2023 All Rights Reserved.

Overview

The Library Generation engine is used to create and modify a cell library database of power-grid views. In an integrated circuit design, it is common for many standard cells and megacells to be reused across multiple designs. Cadence flows require you to pre-process such cells, reducing the overall time required for extraction.

Important

Cadence strongly recommends that you preprocess standard cell layout separately from megacell layout. In most user environments, the standard cell layout rarely changes, but the megacell layout does. In addition, standard cells are more likely to be shared among groups and supported by a central CAD organization. Megacells may be unique to the particular design group without following a strict naming convention, so names may conflict within the same Library Generation library.

Library Generation takes LEF, GDSII, or a combination of LEF and GDSII as input and creates a binary power-grid view of the cell contents within the cell library database. Power-grid views contain the physical and electrical representation of all the cells and blocks in the design. They contain necessary information required for power and rail analysis:

- For rail analysis, it contains RC parasitics, tap currents, geometries
- For power analysis, it contains netlist and output capacitance loading

© 2013-2023 All Rights Reserved.

The following diagram illustrates the library generation flow:

Voltus IC Power Integrity Solution requires power-grid libraries for dynamic power calculation and static and dynamic IR drop analysis. The power-grid libraries are generated for standard cells, memories, IOs and custom macros in the design. Some of the library characterization features include the following:

- GUI driven and interactive TCL Command Interface
- Spectre-SIMI interface for accurate coupling capacitance characterization
- Automatic spawning of library generation jobs for macros (memories)
- Library Reports and Debugging
- Library viewer

Note: Hierarchical PGV generation is not supported by Voltus. Use the hierarchical power integrity analysis flow with Extreme Modeling to generate xPGV model for large digital blocks.

Inputs

The inputs to Library Generation are:

© 2013-2023 All Rights Reserved.

- Technology LEF (header.LEF)
- Cell LEF with power pins
- Cell GDS (for macro blocks)
- Extraction technology file for Quantus (qrcTechFile)
- Layer map file for GDS to qrcTechFile technology file
- Layer map file for LEF to qrcTechFile technology file
- SPICE device models (Spectre or Hspice)
- Subcircuit SPICE-format netlist, preferably with device X,Y coordinates

  This is required so that tap current is snapped to the nearest device and give most accurate load for the device. Without xy coordinates, it results in even distribution of tap current across the entire device.

Outputs

Library Generation outputs the following files:

- A cell library database that contains the following information:
  - LEF technology information, which includes the following:
    - Layer information for each layer: its name, its relationship to other layers, its routing width and direction
    - Via information for each via: its name, the layers that it connects, and a set of geometries accessible as polygons
  - Cell information, which may include the following:
    - Geometric views of the cell
    - Port information
    - Power-grid views
    - Process technology data
  - A bounding box for each cell
- A log file named `*Libgen_*/library_name/*.log*`
- Text report and summary files having detailed information about power-grid library

© 2013-2023 All Rights Reserved.

Types of Power-Grid Libraries

The two types of libraries are:

- Technology Library - The Technology Library contains the tech file, area capacitance specification, decap/filler/powergate cells, and Tech view for all the cells. A technology library is a minimum requirement to run rail analysis.

Important

The technology library is a mandatory requirement for rail analysis. However, to obtain accurate IR drop simulation, power-grid library should be characterized for standard cells and macro blocks. When specifying the list of PGVs to rail analysis using the `set_rail_analysis_mode` command, you must specify the technology library as the first PGV library in the list. If both technology and standard cell PGV libraries exist, the standard cell PGV is given a higher precedence.

- Power-Grid Library- The Power Grid Library contains three kind of power grid views (PGV), Early/IR/EM, for standard cell and macro/memory/io cells. The power-grid library does not contain the tech file, and the decoupling capacitance is derived from spice simulation.

  Note: Power Grid Libraries for standard cells are often generated by a third party and shipped to the customer with the corresponding process. In advanced process technologies (< 90nm) there may be multiple variants for the same process including different number of metal layers, different width and thickness combinations for top metal layers. Since standard cells usually only use the bottom couple of metal layers, these variants should not affect the standard cell library itself. A third party can ship technology libraries for each process variant to use with a single standard cell power grid library or can create a specific combination of corner and header.lef. Voltus will use the Quantus techfile embedded in the technology library to process all the Power Grid Libraries.

Power-Grid View Types

During rail analysis, different PGVs are used based on the accuracy mode selected. The PGV types correspond to the accuracy mode of rail analysis. The different power-grid view types are:

Tech-Only

This view is the basic view generated in the technology library for every cell. It only contains the current distribution factor and decoupling capacitance at the power ports of the cell and does not include any information inside the cell. If any cells do not have a Power Grid Library

© 2013-2023 All Rights Reserved.

defined, tech view will be used to run rail analysis, but it is recommended to generate Power Grid Library for all standard cells and macro blocks to obtain accurate rail analysis.

Early View (Standard cells/Macros)

This view only contains the current distribution factor and spice simulated decoupling capacitance information at the power ports of the cell and does not include the parasitics associated with the interconnect, which prevents IR drop within the block or cell from being analyzed. Early power-grid views are default PGV view for rail analysis XD accuracy mode, which is recommended for early design phase.

IR View (Macros)

IR View will perform reduction on extracted power-grid and merge small current value current taps together without significant impact on accuracy. The IR view was created for tool performance reasons. Current designs in many cases contain an embedded memory for boosting performance. The bit cells of the memory in general contribute to creation of a lot of current taps, however, they do not contribute to the IR much, but will affect the tool performance. This IR view suppresses those current taps which have a very small current value thus improving both memory usage and performance of the tool. There is limited visibility into the cell because the current taps for the bit-cells inside the memory could be collapsed. The IR view is the default for rail analysis HD mode, with the exception for static EM rail analysis where EM view will be applied.

EM View (Macros)

The EM view contains the extracted power-grid from the GDSII data of a cell. It will create current taps on the bottom conducting layer or at the locations of the vias layer above it. GDSII and XY spice netlist is required to generate accurate current distribution and decoupling capacitance, but you can manually force current values and capacitance to be

© 2013-2023 All Rights Reserved.

assigned to these taps. You can also set up a trigger file to manually define the dynamic current behavior or run Spice simulators to obtain current waveform.

Note: For standard cell, the IR/EM view is identical to the Early view due to the simplicity of the standard cells.

The following table illustrates the definition of PGV:

Cell Type	**tech**	**standard cell**	**macro**		
PGV	Tech	Early/IR/EM	Early	IR	EM
Capacitance	area based	spice simulated			
Current Taps	LEF Pins	LEF Pins	LEF/GDS Pins	Contact/VIA	Contact/VIA
Rail Extraction	No	No	No	Yes	Yes
Rail Reduction	not applicable	not applicable	not applicable	Yes	No

© 2013-2023
All Rights Reserved.

Generating a Technology Library

This section describes how to generate a technology library, and the various inputs and outputs of this flow.

Required Input Files

- technology LEF
- cell LEF
- Quantus technology file
- LEFDEF layermap - optional. If you do not specify this file, Voltus will automatically generate it.

Procedure

To create a technology power-grid view, complete the following steps:

1. Read the LEF files for all the cells to be characterized.

   Tcl command:

```
read_lib -lef
../data/lef/tech.lef \
../data/lef/cell_macro.lef \
../data/lef/pso_header.lef \
../data/lef/pso_ring.lef \
../data/lef/pll.lef \
../data/lef/decap.lef
```

2. Specify the settings for technology library characterization through the command line or through the GUI.

   GUI: Select *Power & Rail - Set PowerGrid Library Mode*.

   Note: In the GUI, the fields in the *Advanced* tab are disabled for the *Tech Only* cell type, except for the *Power Gate Cell Characterization* field. When generating technology library, if power gating cells are specified, the ron, idsat, and ileak parameters need to be specified as well. Power gating cells can be characterized using Spice netlist and model when generating standard cell power-grid library.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_pg_library_mode
-extraction_tech_file RCgen.tch
-lef_layermap lefdef.map
-celltype techonly
-power_pins { VDD 1.08 VDDO 1.08 VDDG 1.08 }
-ground_pins { VSS GND VSSG }
-temperature -40
-powergate_parameters { { cell1 VDDG VDD 100 0.003 0.00015 } { cell2
VDDG VDD 100 0.003 0.00015 } }
```

3. Generate the output for power-grid library generation through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Generate PowerGrid Library*.

   Tcl command:

```
generate_pg_library -output tech_pgv
```

The log file will generate the power-grid view generation statistics, as shown in the following log file snippet:

```
** INFO: (VOLTUS_LGEN-3606):
Power Grid View Generation Statistics:
# Total number of cells: 503
# TECH view created: 503 (100%).
```


© 2013-2023 All Rights Reserved.

Output

The output directory will contain the following information:

File Name	Description
`techonly.cl`	The default name of the technology library is `techonly.cl`. Tech views are LEF pin-based views. Included inside tech views: ■ User specified area based cell capacitance is distributed on power-pins evenly ■ Current taps (sinks) are added to the power-pins ■ Total current for the cell is distributed evenly between multiple current taps on each rail. ■ No visibility of current distribution and power-grid inside cell
`techonly.report`	A report for each cell's power/ground net containing: ■ name of power and ground net, voltage value for power pins ■ capacitance value which is area-based ■ number of taps ■ user-defined Idsat, Ileakage, and Ron for POWERGATE cell
`techonly.summary`	report containing: ■ cell types: STDCELL, FILLER, DECOUPLING CAP, POWER_GATE ■ pins, metal layers, number of current taps, etc.

© 2013-2023 All Rights Reserved.

Generating Standard Cells Power-grid Library

For standard cells, Voltus generates all three views: Early, EM, and IR, which are identical to each other. These views for standard cell libraries can improve the accuracy of the static and dynamic IRdrop and EM analysis. Voltus uses SPICE subcircuit and Spectre simulation to generate decoupled capacitance for standard cells. For power-gates, it characterizes Ron, Idsat and Ileakage.

Important

By default, the Spectre simulation is invoked with the `+spice` option when performing standard cell characterization.

This section describes how to generate a standard cell power-grid library, and the various inputs and outputs of this flow.

Required Input Files

- technology LEF
- cell LEF
- Quantus technology file
- LEFDEF layermap - optional. If you do not specify this file, Voltus will automatically generate it.
- Spice netlists for standard cells
- Spectre file that points to the Spectre models, or Spice models

Procedure

To create power-grid views for standard cells, complete the following steps:

1. Read the LEF files for all the cells to be characterized.

   Tcl command:

```
read_lib -lef
../data/lef/tech.lef \
../data/lef/buf_ao.lef \
../data/lef/pso_header.lef \
../data/lef/pso_ring.lef \
../data/lef/decap.lef
```


© 2013-2023 All Rights Reserved.

2. Specify the settings for standard cell library characterization through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Set PowerGrid Library Mode*.

   Note: You can perform standard cell PGV generation without the extraction technology file (qrcTechFile) and LEF layermap file. However, the LEF layer names should be consistent in all PGVs. When you merge a PGV generated without the technology file, with a PGV generated with the technology file and lefdef layermap files, the layermap would be automatically corrected in the merged PGV.

   Tcl command:

```
set_pg_library_mode \
-ground_pins VSS \
-power_pins {VDD 0.9 TVDD 0.9} \
-decap_cells {DECAP8 DECAP64 DECAP4 DECAP32 DECAP2 DECAP16 DECAP1}\
-filler_cells { FILL8 FILL64 FILL4 FILL32 FILL2 FILL16 FILL1}\
-celltype stdcells \
-cell_decap_file ../data/voltus/decap.cmd \
-cell_list_file ../data/voltus/cell.list \
-spice_subckts { \
../data/netlists/gsclib090.sp \
../data/netlists/pso_header.spi \
../data/netlists/pso_ring.spi } \
-lef_layermap ../data/voltus/lefdef.layermap \
-current_distribution propagation \
-spice_models ../data/netlists/spectre_load.sp \
-extraction_tech_file ../data/qrc/gpdk090_9l.tch \
-powergate_parameters { \
{RING_SWITCH TVDD VDD} \
{HEADER_SWITCH TVDD VDD} }
```

3. Generate the output for power-grid library generation through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Generate PowerGrid Library*.

   Tcl command:

```
generate_pg_library -output stdcell_pgv
```


© 2013-2023 All Rights Reserved.

The log file will generate the power-grid view generation statistics, as shown in the following log file snippet:

```
** INFO: (VOLTUS_LGEN-3265):
Power Grid View Generation Statistics:
# Total number of cells: 488
# EARLY view created: 488 (100%)
# IR view created: 488 (100%)
# EM view created: 488 (100%
```

Output

The output directory will contain the following information:

File Name	Description
`stdcells.cl`	The default name of the standard cell library is `stdcells.cl`. Included inside standard cells power-grid views: ■ Accurate coupling capacitance between power and ground pins is calculated from Spice simulation. ■ "Current Taps" are added to the power-pins.
`stdcells.report`	A report for each cell's power/ground net containing: ■ name of power and ground nets, voltage value for power pins ■ capacitance value which is simulation based ■ number of taps ■ Idsat, Ileakage and Ron for power gate cells calculated based on simulation ■ detailed information for power gate cells
`stdcells.summary`	report containing: ■ cell types: STDCELL, FILLER, DECOUPLING CAP, POWER_GATE ■ pins, metal layers, number of current taps, etc.

© 2013-2023 All Rights Reserved.

Note: In case of MSMV cell library characterization, specifying the input voltage range for decaps and power-gate cells is not mandatory. Library Generation automatically uses an internal voltage range to characterize coupling capacitance and other power-gate parameters for different voltages. These values are included in the library report file generated by the report library (`check_pg_library`). The Voltus Rail Analysis tool automatically selects appropriate capacitance based on the operating voltage of the net that is being analyzed. It generates a warning message when the cell library for decaps/power-gate cells is characterized outside the range of the operating voltage.

© 2013-2023 All Rights Reserved.

Creating a Power-grid View for a Macro Cell

Macro power-grid views are generated to accurately capture current and capacitance distribution inside the memory, macro, and IO cells. For macro/memory/IO, Voltus generates all three views: Early, EM, and IR.

The library generation requires GDS and spice subckt with device X-Y location. The library characterizer simulates the spice subckt and based on spice device models it calculates accurate capacitance and current distribution for the cell. The power net parasitics are extracted down to contact or up to the user-specified via layer, and current taps are snapped to the nearest via location.

If you have a spice netlist without device X-Y location, the accurate views will still capture the correct capacitance but it will distribute the total current evenly on the current taps. This could potentially mask the real IRdrop and EM problem inside the cell or result in false violations.

This section describes how to generate a macro cell power-grid library, and the various inputs and outputs of this flow.

Required Input Files

- technology LEF
- cell LEF
- Quantus technology file
- LEFDEF layermap - optional. If you do not specify this file, Voltus will automatically generate it.
- Spice netlist for the macro cell
- Spectre file that points to the Spectre models, or Spice models
- GDS file of the macro
- GDS layermap

Important

For macro PGV generation, the input files should contain only one macro definition. That is, each LEF, netlist (Spice or CDL), and GDS file should contain only one macro definition.

© 2013-2023 All Rights Reserved.

Procedure

To create power-grid views for a macro cell, complete the following steps:

1. Read the LEF file for the cell to be characterized.

   Tcl command:

```
read_lib -lef
../data/lef/tech.lef \
../data/lef/pll.lef
```

2. Specify the settings for macro cell library characterization through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Set PowerGrid Library Mode*.

   In the in the *Advanced* tab, you can specify the following:

   - *Stop@via*: Specifies to stop the extraction of the power-grid network at the specified via layer. `VIA_1` is the commonly used layer to create current tap. `CONT` diffusion contact layer is used when IP is not qualified for EM. Upper layers, such as `VIA_2` and `VIA_3`, are used when IP is qualified for EM and its effect is to be seen at the top level. Higher layer reduces the memory, disk footprint, and runtime due to reduced resistor and node count.
   - *Current Distribution*: Specifies the method of current distribution inside the cell.

   Tcl command:

```
set_pg_library_mode \
-gds_files ../data/gds/pll.gds \
-ground_pins VSS \
-cell_list_file ../data/voltus/macro.list \
-power_pins {VDD 0.9} \
-celltype  macros  \
-spice_subckts ../data/netlists/pll.sp \
-gds_layermap ../data/voltus/gds.layermap \
-lef_layermap ../data/voltus/lefdef.layermap \
-stop@via CONT \
-spice_models ../data/netlists/spectre_load.sp \
-current_distribution propagation \
-extraction_tech_file ../data/qrc/gpdk090_9l.tch
```


© 2013-2023 All Rights Reserved.

3. Generate the output for power-grid library generation through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Generate PowerGrid Library*.

   Tcl command:

```
generate_pg_library -output macro_pgv
```

The log file will generate the power-grid view generation statistics, as shown in the following log file snippet:

```
** INFO: (VOLTUS_LGEN-3265):
Power Grid View Generation Statistics:
# Total number of cells: 1
# EARLY view created: 1 (100%)
# IR view created: 1 (100%)
# EM view created: 1 (100%)
```


© 2013-2023 All Rights Reserved.

Output

The output directory will contain the following information:

File Name	Description
`macros_<cellname>.cl`	The default name of the macro power-grid library is `macros_<cellname>.cl`. Inside macro Early power-grid views: ■ Power net parasitic is extracted for power-pins only ■ Accurate coupling capacitance between power and ground pins is calculated from spice simulation. ■ "Current Taps" are added to the power-pins ■ No visibility of current distribution and power-grid inside macro cell. Inside macro IR power-grid view: ■ Reduced distributed RC for power net and intrinsic capacitance ■ Current Taps added to merged devices in the region with relatively small current consumption (e.g. bit cells region in the memory) ■ Complete visibility inside the cell for power-grid and transistor Inside macro EM power-grid view: ■ Distributed RC for power net and intrinsic capacitance ■ Current taps for each device's contact in the design ■ Complete visibility inside the cell for power-grid and transistor

© 2013-2023 All Rights Reserved.

File Name	Description
macros_<*cellname*>.report	A report for the cell's power/ground net containing: ■ name of power and ground nets, voltage value for power pins ■ capacitance value which is simulation based ■ number of taps for the Early/EM/IR views
macros_<*cellname*>.summary	report containing capacitance, the current of analyzed nets, resistors, and the number of current taps and layers for the Early/EM/IR views.

© 2013-2023 All Rights Reserved.

Characterizing Current Waveforms for Macro EM Views

Voltus supports characterization of the dynamic current behavior of custom macros. These current waveforms will be used during rail analysis instead of the default triangular waveform. To characterize current inside a macro PGV, the flow needs one of the following inputs:

- FSDB with Spice current waveforms
- Datasheet parameters
- User-defined PWL
- Simulation vectors
- Liberty-based vectors
- User-specified Spice deck

Note: For macro PGVs, the dynamic tap current port view is supported in both the early and EM views in the PWL-based flow.

A user-specified trigger file is used to add the signature of current for different mode conditions. The trigger file contains information on the input vectors for which it will toggle, waveform triggering input pin information, and current information. The trigger condition is used to identify the functional mode (read/write/idle) of the custom macro in the Power and Rail Analysis flow. During power analysis, if Macro PGVs with dynamic current is provided, the software will schedule the switching events for the macro blocks based on the activity ratio. The scheduling time will be based on the timing information on the trigger pin, and all the switching events will be stored into the trigger files under the power analysis output directory. Rail analysis will load the trigger files automatically when the current files (*.pti*) are loaded. The trigger file enables you to get the current signature of the memory for different current inputs during PGV generation.

To run this flow, you should also provide a spice netlist with x,y coordinates, and Library Characterization calculates average currents through each of them and then hooks up the currents to the nearest node on the contact layer preserving the current distribution. Extraction is done till the lowest contact layer or cut layer as defined by users.

© 2013-2023 All Rights Reserved.

When you specify a hierarchical netlist, the cell instances should have their coordinate information in the $T format. The $T format is given below:

```
$T=x,y,r,a

x - X-coordinate

y - Y-coordinate

r - reflection along x axis (0 or 1), 0 - No reflection, 1 -
Reflection is there along x-axis

a - rotation angle in degrees (0, 90, 180, 270)
```

The following diagram illustrates the current characterization flow:

© 2013-2023
All Rights Reserved.

Tcl command:

```
set_pg_library_mode \
  -power_pins {VDD 1 } \
  -ground_pins VSS \
  -celltype macros \
  -extraction_tech_file ../tech_worst.tch \
  -lef_layermap ../lef_layermap.txt \
  -cell_list_file ../16x32memory/cell.list \
  -spice_models {../model1.l ../model2.l  ../model3.l.special} \
  -spice_corners {{TT tt_hvt tt_lvt DIO TT_DIO} {TT_sr} {TT_sr}} \
  -current_distribution {dynamic_simulation trigger.txt}

set_advanced_pg_library_mode \
  -pgdb_list_file ../cell_32x64.hdr \
  -pgdb_layermap ../pgdb_layermap.txt \
  -libgen_command_file ../cell_32x64_1/lib.inc

generate_pg_library \
  -output .
```

The following is the sample content of the trigger.txt file:

```
CURRENT_CHARACTERIZATION_METHOD DOTLIB
DOTLIB_LIST test.lib
MODE_COUNT 3
SIM_STOP_TIME 10ns
SIM_STEP_SIZE 20ps
```

For the current characterization flow, you must specify the `-current_distribution { dynamic_simulation` *`trigger_file`* `}` parameter.

© 2013-2023 All Rights Reserved.

The format of the trigger file is:

Trigger File Format	Description
#Method of current characterization	
`CURRENT_CHARACTERIZATION_METHOD <method>` `<SIMULATION/DOTLIB/PWL/FSDB/DATASHEET/SPICE_DECK_BASED>`	■ `SIMULATION` - Vector-based flow ■ `DOTLIB` - Liberty based Flow ■ `PWL` - User-defined PWL flow ■ `FSDB` - Spice total current output captured in the FSDB format ■ `DATASHEET` - Datasheet details of any memory ■ `SPICE_DECK_BASED` - User-specified Spice deck flow
#Memory cell name	
`CELL <cellname1>`	Name of the memory cell
#Mode and conditional details for trigger	

© 2013-2023 All Rights Reserved.

`LEAKAGE_CURRENT { <PGPin1> <value1> … <PGPinN> <valueN> }`	Specifies the leakage current for both power and ground pins, or just power pins. Dynamic Power Analysis will process `LEAKAGE_CURRENT` specified in the PGV trigger file, and use it when a macro cell is idle or dynamic switch pattern is not defined (defined as "-").
	
	In the following example, the second cycle will use `LEAKAGE_CURRENT` defined in the PGV trigger file:
	
	`set_power -cell x -dynamic_switch_pattern { READ - READ }`
	
	You must specify the leakage current for all power pins. If leakage current is not defined for some pins, then those pins will get 0 leakage current when processed by Power Analysis.
	
	`LEAKAGE_CURRENT` should be specified in the trigger file as a global, and specified outside the mode specification, as shown below:
	
	`CURRENT_CHARACTERIZATION_METHOD PWL`
	`CELL cell1`
	
	`LEAKAGE_CURRENT { VSS 2p VDD 17n }`
	`MODE_NAME READ static`
	`...`
	
	`END_MODE`
	`END`

`MODE_NAME | MODE <mode_name> dynamic | static CONDITIONAL_INPUT <conditional input statement for cellname1>`	Mode name and conditional input statement for the cell. You can specify either the `MODE` or the `MODE_NAME` keyword. The optional `static` keyword with the `MODE` keyword indicates static current behavior and the `dynamic` keyword with the `MODE` keyword indicates dynamic current behavior. Conditional input of current for the given mode.
`CONDITIONAL_PIN <pin_name> (rise|fall|both)`	Conditional pin statement for the cell. Signal pin for which the tool should rise or fall.

© 2013-2023 All Rights Reserved.

`CONDITIONAL_PIN_REFERENCE_TIME <time>`	Specifies the reference time for conditional pins with respect to the specified waveform. The reference time information from the PGV is used later by Power Analysis to include the reference time with the actual arrival time in the trigger file generated by the Power Analysis engine, and by Rail Analysis to apply the reference time to the switching window. The Rail Analysis engine will use the value of reference time in the Power Analysis trigger file to calculate the waveform start time, that is equal to `power_analysis_trigger_time - (ref_time)`. *time* can be specified as a positive or negative value.
`CONDITIONAL_STIMULUS_FILE <usim1.txt>`	Simulation vector detail in the given mode.
`BIAS_VOLTAGE { pin1 voltage1 pin2 volatge2 .... }`	Specifies different bias conditions for capacitance characterization. This keyword allows you to simulate a macro's coupled capacitance for different modes with different bias conditions. You must specify this keyword for every mode even if the bias conditions are the same.
`CUSTOM_LABEL label_name`	A label or tag to indicate a sub-mode.
`VOLTAGE {net_name1 voltage1 net_name2 voltage2 ...}`	The voltage values at which simulation is done for each net.

© 2013-2023 All Rights Reserved.

CAPACITANCE { *cap_value* }	The load capacitance value for the specified sub-mode (CUSTOM_LABEL). *Important* Using the VOLTAGE and CAPACITANCE keywords, you can specify varying capacitance and voltage values for different sub-modes (CUSTOM_LABEL) in the PWL and FSDB-based current characterization flows. The Power Analysis engine uses the different capacitance PWLs for each mode, and scales the current based on the actual loading of the instance in the design. Rail Analysis uses the different voltage PWLs to derate the current waveform based on the voltage drop in the design. For each mode's capacitance and voltage variation, you need to specify the FSDB file (FSDB_FILE or FSDB_PATH) in the FSDB current characterization method, and specify the PWL file (USER_PWL_FILE) in the PWL current characterization method.
USER_PWL_FILE <pwl1.txt>	User-specified PWL file that contains information about the PWL waveforms (power/ground currents). The current for ground net should be specified as a negative value.
FSDB_START_TIME <time> FSDB_END_TIME <time> FSDB_FILE <filename.fsdb>	FSDB Simulation based mode details
CURRENT_FACTOR=X	Scaling factor for peak current. This keyword is specified inside a custom label in the PWL and FSDB-based current characterization flows.
TIME_FACTOR=Y	Scaling factor for time. This keyword is specified inside a custom label in the PWL and FSDB-based current characterization flows.

© 2013-2023 All Rights Reserved.

`NET=<netname>`	Specifies the net name. This keyword allows you to specify a different time and current scale factor for each power/ground net inside a custom label in the PWL and FSDB-based current characterization flows. If a net name is not specified, the same scale factor will be applied to all nets.
`SWITCHBIT_WINDOW { Tmin1 Tmax1 Tmin2 Tmax2 }`	Specifies the time windows for different modes/sub-modes. You can specify multiple `Tmin/Tmax` windows for the same PWL waveform. Thereafter, the Rail Analysis engine reads this switch-bit information from the PGV, and reports the elapsed and worst IVD when the switch-bit is 1. You can also specify negative switch-bit time. The Rail Analysis engine reads this negative switch-bit time from the PGV, and applies it from the waveform start time that was derived from the trigger file generated by the Power Analysis engine. Effective Instance Voltage (EIV) will be calculated in the switch-bit window.
`DATASHEET_PARAMETER < {pg_pin1 Capacitance Leakage} {pg_pin2 Capacitance Leakage}>`	Datasheet-based mode details
`END_MODE`	End mode section
#Datasheet clock required parameters	

© 2013-2023 All Rights Reserved.

`DATASHEET_INPUT_SLEW <time>`
`DATASHEET_OUTPUT_SLEW <time>`
`DATASHEET_DELAY <time>`
`DATASHEET_START_TIME <time>`
`DATASHEET_PARAMETER < {power_pin cap  leakage_pwr} >`
`PEAK_CURRENT {{ power_pin1 value1} { power_pin2 value2}...}`
	■ `DATASHEET_INPUT_SLEW` - Clock input slew of the conditional pin
	■ `DATASHEET_OUTPUT_SLEW` - Rising slew of the output pin Q
	■ `DATASHEET_DELAY` - Mode delay as the read_access and write_access time
	■ `DATASHEET_START_TIME` - Specifies the start time of the waveform
	■ `DATASHEET_PARAMETER` - Datasheet parameters include power pin voltage, load capacitance, and leakage power. Load capacitance is mode dependent.
	■ `PEAK_CURRENT` - Specifies peak current for the power pins.
	❑ When this parameter is specified, Library Generation constructs a triangular-shaped waveform with peak current matching the user-specified peak. If the area of the triangular waveform is less than the specified charge, another smaller peak is constructed before the main peak.
	❑ When this parameter is not specified, Library Generation constructs a trapezoid-shaped waveform whose area matches the given charge.

#FSDB required parameters	

© 2013-2023 All Rights Reserved.

`FSDB_PATH <path>`
`FSDB_NET_NAME <net>`
`[<generic_name>]`
`FSDB_START_TIME <time>`
`FSDB_END_TIME <time>`
`FSDB_HIER_CHAR <hier_char, for hier DB>`
	■ FSDB file path
	■ Power net name and its FSDB probe name
	■ FSDB start and end time
	■ Hierarchical separator for hierarchical database. The default separator is "."
	
	Single FSDB file having total current for ALL power/ground pins with all mode current in it

#PWL required parameters	
`PWL_START_TIME <time>` `PWL_END_TIME <time>`	Time ranges for different modes in the PWL-based current characterization flow. These keywords allow you to use a single PWL input file for multiple modes and specify different time ranges for these modes in which the dynamic PGV is being created.
#User-specified Spice deck required parameters	

`DECK_FILE <filename>`
`SIM_OPTIONS <option_name>`
`FSDB_FILE <filename>`
	■ Specify a single deck file for all modes, or a unique deck file for each mode.
	■ User-specified Spectre simulator options. The default value of this keyword is `+aps`.
	■ Voltus executes the Spice deck and reads the current waveform generated by simulation. After simulation of the deck file, an FSDB file is generated. You must specify the FSDB file name with the `FSDB_FILE` keyword.

#<Other mode independent parameters>	

© 2013-2023 All Rights Reserved.

`SIMULATOR <type of simulator>`
`DOTLIB_LIST {a list of memory dotlibs files}`
`MODE_COUNT`
`MODE_CONFIG_FILE <file_path>`
`SIM_START_TIME <time>`
`SIM_STOP_TIME <time>`
`SIM_STEP_SIZE <step_size>`
`STIMULUS_FILE <file_path>`
`INCLUDE_FILE <file_path>`
	■ `SIMULATOR` - Specifies the simulator to be used for characterizing current waveforms for Macro EM views. You can specify the following simulators: Accelerated Parallel Simulator (APS), SPECTRE, and UltraSim. APS is the default simulator.
	■ `DOTLIB_LIST` - Specifies a list of memory dotlibs.
	■ `MODE_COUNT` - Specifies the maximum number of modes to be generated. Valid values are either a positive number or `` `all' ``. `` `all' `` means stimulus for all power groups are to be generated. The default value is `4`. If you set this parameter to all, it may result in high runtime as it may generate too many modes.
	■ `MODE_CONFIG_FILE` - Specifies the file containing extra information for memory cell. This file contains the following information - signal slew, signal active value, and memory write information.
	■ `SIM_START_TIME` - Specifies the simulation start time. The default unit is in nanoseconds (ns).
	■ `SIM_STOP_TIME` - Specifies the simulation stop time. The default unit is in nanoseconds (ns).

© 2013-2023 All Rights Reserved.

	■ `SIM_STEP_SIZE` - Specifies the step size for power-grid analysis. The default value of step size is 20 picoseconds (ps). This parameter supports other units such as, ns and fs. This parameter supports unit-prefixes (pico, femto, nano) combined with units of measure (ps, fs, ns), but does not support units itself ("second", "Ohm", "Farad") without unit-prefix. ■ `STIMULUS_FILE` - Specifies the stimulus file containing the trigger information for all input ports. ■ `INCLUDE_FILE` - Specifies the name of the simulation include file.
#End cell section	
`END`	End cell section

Trigger File Format - for simulation vector-based files

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD SIMULATION
SIMULATOR APS
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
CONDITIONAL_STIMULUS_FILE <usim1.txt>
END_MODE
END
```

where,

"!" denotes low state of the signal

"&" denotes boolean AND operation

"|" denotes boolean OR operation

© 2013-2023 All Rights Reserved.

For each mode, you need to specify separate stimulus files (for example usim1.txt and usim2.txt). The stimulus files contain the PWL definitions that will be used by UltraSim for simulating the specific mode.

Example 1 - Simulation

```
CURRENT_CHARACTERIZATION_METHOD SIMULATION
SIMULATOR APS
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = (! ATPGM & EZ &! SCAN &! TEZ & TM &! TPAGE &! TWTZ
&! TWZ)
CONDITIONAL_PIN CLK {RISE}
CONDITIONAL_STIMULUS_FILE usim.txt
END_MODE
END
```

Trigger File Format - for user-defined PWLs

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
USER_PWL_FILE pwl.txt
PWL_START_TIME <time>
PWL_END_TIME <time>
END_MODE
END
```

The two formats of the PWL file are:

Format 1: Single PWL for Each Net

```
UNIT CURRENT <current unit> (default is A, Supported values are A,
mA, uA & nA)
UNIT TIME <time unit> (default is s, Supported values are s, ms, us,
ns, ps)
NET <net_name1>
REGION {t1 i1 t2 i2 t3 i3.. } | PWLFILE net1_pwl.txt
```


© 2013-2023 All Rights Reserved.

```
NET <net_name2>
REGION {t4 i4 t5 i5 t6 i6.. } | PWLFILE net2_pwl.txt
```

Here, the time and current units are as defined using `UNIT CURRENT` and `UNIT TIME` syntax. `NET` specifies that the `REGION` command following it applies to the specified net name. This stays in effect until another `NET` or `CELL` specification is encountered. `REGION` specifies the power up piecewise linear (PWL) stimulus in (time, current) pair format. Instead of the `REGION` keyword, you can use the `PWLFILE` keyword to include a supply net (power or ground) or a switched-net-specific PWL file.

The following is an example of the file specified with the `PWLFILE` keyword:

```
0, 3
1.0e-9, 5
1.2e-9, 3
...
```

Here, each line contains the time and current pairs.

Both the `REGION` and `PWLFILE` keywords can be used, but the `PWLFILE` keyword is recommended for long waveforms so that you can easily edit the PWL values, insert some new points, and comment out certain points that are to be ignored.

Format 2: Multiple PWLs for Each Net (Specify Multiple PWLs for Different Regions)

```
UNIT CURRENT mA
UNIT TIME ns
NET VDD
REGION { 0 1.0 0.100 1.0 ... } RECT {llx1 lly1 urx1 ury1}
REGION { 0 0.5 0.100 2.5 ... } RECT {llx2 lly2 urx2 ury2}...

NET VSS
REGION { 0 -1.0 0.100 -1.0 ... } RECT {llx1 lly1 urx1 ury1}
REGION { 0 -0.5 0.100 -2.5 ... } RECT {llx2 lly2 urx2 ury2}
```

This format allows you to assign multiple PWLs to different regions inside a macro. This format has multiple occurrences of the `REGION` keyword in the `NET` section. Each occurrence of the `REGION` keyword specifies the entire PWL waveform in the time-current pair format as a single line. This is followed by the `RECT` keyword that specifies the coordinates of an area inside the macro. This PWL file format allows you to model the dynamic current waveform for the taps in different areas accurately.

© 2013-2023 All Rights Reserved.

Example 1: User-defined PWL

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL mem1
MODE_NAME  READ
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ
& !TWZ )
CONDITIONAL_PIN  CLK { RISE }
USER_PWL_FILE pwl.txt
END_MODE
END
```

The following is the sample content of the `pwl.txt` file:

```
UNIT CURRENT A
UNIT TIME s
NET VSS
REGION { 0.0000 -430.6500e-9 20.0000e-12 -43.4421e-6 40.0000e-12 -
23.2054e-6 }
NET VDDP
REGION { 0.0000 -69.8031e-9 20.0000e-12 1.5876e-6 40.0000e-12
3.1345e-6 }
NET VDDPR
REGION { 0.0000 -2.8611e-9 20.0000e-12 4.5883e-9 40.0000e-12
162.9478e-9}
NET VDDAR
REGION { 0.0000 -1.5191e-09 20.0000e-12 871.1007e-09 40.0000e-12
8.7003e-09 }
```

The following is the sample content of the `pwl.txt` file that specifies multiple PWLs for different regions:

```
UNIT CURRENT mA
UNIT TIME ns
NET VDDCE
REGION { 0 10.0000 0.4000 20.0000 0.4800 200.40000 0.5384 0 0.9496 0
} RECT { 0.000 0.000 66.000 100.000 }
REGION { 0 2.0000 0.4000 4.0000 0.4800 41.40000 0.5384 0 0.9496 0 }
RECT { 0.000 100.001 66.000 202.000 }
END VDDCE
```


© 2013-2023 All Rights Reserved.

Example 2: User-defined PWL

Following is an example of the trigger file specifying time ranges for different modes using a single PWL input file:

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL MEM1

MODE_NAME READ
CONDITIONAL_INPUT !WE
CONDITIONAL_PIN CLK { RISE }
USER_PWL_FILE pwl.txt
PWL_START_TIME 0ns
PWL_END_TIME 5ns
END_MODE

MODE_NAME WRITE
CONDITIONAL_INPUT WE
CONDITIONAL_PIN { RISE }
USER_PWL_FILE pwl.txt
PWL_START_TIME 5ns
PWL_END_TIME 10ns
END_MODE
END
```

Example 3: User-defined PWL

Following is an example of the trigger file specifying different sub-modes (custom-labels):

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL MEM1

MODE_NAME READ dynamic
CONDITIONAL_INPUT =  !CEN
CONDITIONAL_PIN CLK { RISE }

CUSTOM_LABEL A_1
VOLTAGE { VDD 0.9 VDD_LP 0.9 VSS 0 }
USER_PWL_FILE pwl1.txt

CUSTOM_LABEL A_2
VOLTAGE { VDD 0.7 VDD_LP 0.7 VSS 0 }
USER_PWL_FILE pwl1.txt
```


© 2013-2023 All Rights Reserved.

```
END_MODE
...
END
```

Example 4: User-defined PWL

Following is an example for specifying time and current scale factor for each net in the trigger file:

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL cell2
LEAKAGE_CURRENT { VSS 2p VDD 17n }

MODE_NAME  READ static
CONDITIONAL_INPUT =  !CEN
CONDITIONAL_PIN CLK { RISE }
CONDITIONAL_PIN_REFERENCE_TIME 0.0

CUSTOM_LABEL A1
VOLTAGE { VDD 0.9 VDD_LP 0.9 VSS 0 }
CAPACITANCE 10
USER_PWL_FILE /../pwl_1.txt TIME_FACTOR=2 CURRENT_FACTOR=2 NET=VDD
TIME_FACTOR=1.400 CURRENT_FACTOR=1.200 NET=VSS
STATIC_INTERNAL_POWER { VSS 1p VDD 17n VDD_LP 1 }
CELL_DECAP_FILE cell_decap_read.txt
SWITCHBIT_WINDOW {-3 4} VSS
SWITCHBIT_WINDOW {4 8} VDD

...

END_MODE

END
```

Trigger File Format - FSDB

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
```


© 2013-2023 All Rights Reserved.

```
FDSB_START_TIME <time>
FSDB_END_TIME <time>
END_MODE
FSDB_PATH <path>
FSDB_NET_NAME <net> [<generic_name>]
END
```

Example 1: FSDB

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ
& !TWZ )
CONDITIONAL_PIN CLK { RISE }
FDSB_START_TIME 0ns
FSDB_END_TIME 10ns
END_MODE
FSDB_PATH “new.fsdb"
FSDB_NET_NAME "i1(vvddpr)" "VDDPR"
FSDB_NET_NAME "i1(vvddar)" "VDDAR"
FSDB_NET_NAME "i1(vvddp)" "VDDP"
FSDB_NET_NAME "i1(vvss)" "VSS"
END
```

Example 2: FSDB

You can read multiple FSDBs for different modes, and specify different start and end times for each mode. The following example shows that 2 FSDB files are read, wherein `read.fsdb` is read for the `READ` mode, and `write.fsdb` is read for the `WRITE` mode:

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL BLULP00256032040
MODE_NAME READ
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ
& !TWZ )
CONDITIONAL_PIN CLK { RISE }
FDSB_START_TIME 0ns
FSDB_END_TIME    10ns
FSDB_PATH "read.fsdb"
FSDB_NET_NAME "i1(vvddpr)"  "VDDPR"
FSDB_NET_NAME "i1(vvddar)"  "VDDAR"
END MODE
```


© 2013-2023 All Rights Reserved.

```
MODE_NAME WRITE
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ
& !TWZ )
CONDITIONAL_PIN CLK { RISE }
FDSB_START_TIME 0ns
FSDB_END_TIME    15ns
FSDB_PATH "write.fsdb"
FSDB_NET_NAME "i1(vvddpr)"  "VDDPR"
FSDB_NET_NAME "i1(vvddar)"  "VDDAR"
END MODE
END
```

Example 3: FSDB

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL cellA
MODE_NAME WRITE
CONDITIONAL_INPUT = !CEN
CONDITIONAL_PIN CLK { FALL }
BIAS_VOLTAGE { A[1] 0.9 A[2] 0.9 }
FSDB_START_TIME 19ns
FSDB_END_TIME 33ns
FSDB_PATH "./tran.fsdb"
END_MODE

MODE_NAME READ
CONDITIONAL_INPUT = !CEN
CONDITIONAL_PIN CLK { RISE }
BIAS_VOLTAGE { A[1] 0.9 A[2] 0.0 }
FSDB_START_TIME 10ns
FSDB_END_TIME 40ns
FSDB_PATH "./tran.fsdb"
END_MODE
```

Trigger File Format - Datasheet

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD DATASHEET
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
```


© 2013-2023 All Rights Reserved.

```
DATASHEET_DELAY <time>
DATASHEET_PARAMETER < {power_pin cap  leakage_pwr} >
PEAK_CURRENT {power_pin peak_current}
END_MODE
DATASHEET_INPUT_SLEW <time>
DATASHEET_OUTPUT_SLEW <time>
END
```

Example 1: Datasheet

```
CURRENT_CHARACTERIZATION_METHOD DATASHEET
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = ( WZ & EZ )
CONDITIONAL_PIN  CLK { RISE }
DATASHEET_DELAY  7.6913ns
DATASHEET_PARAMETER {{ VDDAR 0.1715pf 2.496e-05mW }{ VDDPR 18.8985pf
3.63e-06mW }{ VBBNW 0.1780pf 2.601e-06mW  }}
END_MODE

MODE_NAME WRITE
CONDITIONAL_INPUT = ( !WZ & !WTZ & EZ  )
CONDITIONAL_PIN  CLK { RISE }
DATASHEET_DELAY  8.216ns
DATASHEET_PARAMETER {{ VDDAR 0.1715pf  1.355e-06mW }{ VDDPR  0.1715pf
1.81e-05mW } { VBBNW 0.1715pf 2.614e-06mW }}
END_MODE

MODE_NAME STANDBY
CONDITIONAL_INPUT = ( !EZ )
CONDITIONAL_PIN  CLK { RISE }
DATASHEET_DELAY  0.6913ns
DATASHEET_PARAMETER {{ VDDAR 0.1715pf 2.902e-05mW }{ VDDPR  0.2024pf
1.81e-05mW } { VBBNW 0.1723pf 6.92e-07mW }}
END_MODE

DATASHEET_INPUT_SLEW 0.6ns
DATASHEET_OUTPUT_SLEW 0.3682ns
END
```


© 2013-2023 All Rights Reserved.

Example 2: Datasheet

```
CURRENT_CHARACTERIZATION_METHOD DATASHEET
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = ( WZ & EZ )
CONDITIONAL_PIN CLK { RISE }
DATASHEET_DELAY 7.6913ns
DATASHEET_PARAMETER {{ VDD 0.1715pf 2.5e-05W }{ VDDPR 18.8985pf
3.63e-06W }{ VBBNW 0.1780pf 2.601e-06W }}
PEAK_CURRENT {{ VDD 0.15mA} { VDDPR 0.1mA} { VBBNW 0.25mA}}
END_MODE

DATASHEET_INPUT_SLEW 0.6ns
DATASHEET_OUTPUT_SLEW 0.8ns
END
```

Example 3: Datasheet

```
CURRENT_CHARACTERIZATION_METHOD DATASHEET
CELL COMPILER_MEM
MODE_NAME READ
CONDITIONAL_INPUT = (ME & !WE &!LS & !DS & !SD)
CONDITIONAL_PIN CLK {RISE}
DATASHEET_DELAY 1.2ns
DATASHEET_START_TIME 100e-12
DATASHEET_PARAMETER {{VDD 0.1193e-09 0.0025mW} {VDDM 0.00146e-09
0mW}}
DATASHEET_PEAK_CURRENT {{VDD 68mA} {VDDM 0mA}}
END_MODE

MODE_NAME WRITE
CONDITIONAL_INPUT = (ME & !WE &!LS & !DS & !SD)
CONDITIONAL_PIN CLK {RISE}
DATASHEET_DELAY 1.2ns
DATASHEET_START_TIME 100e-12
DATASHEET_PARAMETER {{VDD 0.1813e-09 0.0025mW} {VDDM 0.00146e-09
0mW}}
DATASHEET_PEAK_CURRENT {{VDD 82mA} {VDDM 0mA}}
END_MODE

DATASHEET_INPUT_SLEW 5.508481e-03ns
DATASHEET_OUTPUT-SLEW 3.693163e-03ns
END
```


© 2013-2023 All Rights Reserved.

Trigger File Format - for user-specified Spice Deck

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD SPICE_DECK_BASED
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
FDSB_START_TIME <time>
FSDB_END_TIME <time>
END_MODE
DECK_FILE <filename>
FSDB_NET_NAME <net> [<generic_name>]
FSDB_FILE <filename>
SIM_OPTIONS +aps
END
```

Example 1: User-specified Spice Deck

```
CURRENT_CHARACTERIZATION_METHOD SPICE_DECK_BASED
CELL cellA
MODE READ
CONDITIONAL_INPUT = ( !CEN & !WEN )
CONDITIONAL_PIN CLK { RISE }
FSDB_START_TIME 0ns
FSDB_END_TIME  0.5ns
END_MODE

MODE WRITE
CONDITIONAL_INPUT = ( !CEN & !WEN )
CONDITIONAL_PIN CLK { RISE }
FSDB_START_TIME 0.6ns
FSDB_END_TIME  2ns
END_MODE

DECK_FILE deck.cir
FSDB_NET_NAME i1(vvdd)  VDD
FSDB_NET_NAME i1(vvss) VSS
FSDB_FILE usimCktFile.fsdb.gz
SIM_OPTIONS +aps

END
```


© 2013-2023 All Rights Reserved.

Trigger File Format - Dotlib

For dotlib-based current characterization, the trigger and stimulus files are automatically generated from an existing dotlib. This flow is applicable only for memory cells. Voltus generates stimuli for the worst power groups (rise_power, fall_power or power) in the specified dotlib. It reads only the internal power groups having the "when" condition and constructs stimulus vectors. Each of the generated stimulus belongs to one of the following groups:

- pin power based (this group is associated with input pin)
- arc-based (this group is associated with output pin)

Note: arc-based group without related_pin will be ignored.

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD DOTLIB
DOTLIB_LIST {a list of memory dotlibs files}
MODE_COUNT <specifies the maximum number of modes to be generated.
Valid values are either a positive number or `all'. The default value
is 4>
MODE_CONFIG_FILE <file_path>
SIM_STOP_TIME <time>
SIM_STEP_SIZE <step_size>
```

If the dotlib contains arc-based power groups, you must specify `MODE_CONFIG_FILE` and include the `memory_write` condition in the memory description file. Otherwise, arc-based power groups will be skipped. The memory description file contains the following information:

- **Signals slew**: specify slew for signals. These slew values will be used in case rise or fall is to be applied on signal. The slew value specified using this construct takes priority over the slew value saved for the group. The default slew unit in the slews construct is ns. The syntax is:

  ```
  slews {
  <signal_1> <slew_value_1>
  <signal_2> <slew_value_2>
  ....
  }
  ```

- **Signals**: specify signals' active values irrespective of `memory_write`. The syntax is:

  ```
  signals {
  <sig_1_name> <active_value>
  <sig_2_name> <active_value>
  ....
  ```


© 2013-2023 All Rights Reserved.

```
    }
```

where `<active value>` can be `H` or `L`

- **Memory write information**: specify information required to write data to memory through `memory_write` block.

```
memory_write {
clock <clock_name> <type>, where <type> can be R, F, H or L
data_in <data_bus_name>
address <address bus name>
signal <sig_1 name> <active_value>, where <active value> can be H
or L
signal <sig_2 name> <active_value>, where <active value> can be H
or L
....
}
```

A sample memory description file is given below:

```
cell memory1
{
memory_write
{
  address A
  data_in D
  clock CLK R
  signal WEN L
  signal CEN L
}
    slews

{
  CLK 0.1
}
  signals

{
   D[1] H
}
}
```

Example 1: Dotlib

```
CURRENT_CHARACTERIZATION_METHOD DOTLIB
DOTLIB_LIST {}
```


© 2013-2023 All Rights Reserved.

```
MODE_COUNT all
SIM_STOP_TIME 10ns
SIM_STEP_SIZE 100ps
```


© 2013-2023 All Rights Reserved.

Sample Library Generation Output Report

The following is a report generated from a power-grid library using a dotlib-based trigger file:

```
=================================================================
Cell    PowerNets  Capacitance    PowerGrid Views
-----------------------------------------------------------------|
Cell1
      VSS(0.0000)   5.3e-12    EARLY(255 taps) EM(5279 devices) IR(4890 devices)
      VDD(0.9000)   5.2e-12    EARLY(193 taps) EM(3371 devices) IR(2326 devices)

Cell_Status: PASS
INFO: Dynamic Tap Current views were created for this cell. MODE1 has Conditional
input = "( !CEN & !WEN ) " and Conditional pin = "CLK RISE".
INFO: Dynamic Tap Current views were created for this cell. MODE2 has Conditional
input = "( CEN & WEN ) " and Conditional pin = "CLK RISE".
-----------------------------------------------------------------|
```

Tip

The PGV report containing the switch-bit information will have the information about the Tmin/Tmax windows, as shown below:

```
INFO: Dynamic Tap Current views were created for this cell.
read has Conditional input = "!control2 " and Conditional pin
= "ref_clk rise" and SWITCHBIT_WINDOW = "{0.04 0.085}".
```

The `.summary` file generated during the characterization of current waveforms for custom macros includes the functional modes or custom modes (if given by the user) and the area information for each mode.

Contents of Trigger File Based Memory/Macro Cell Library

Inside the accurate memory/macro power-grid view:

- Distributed RC for power net and intrinsic capacitance
- PWL current taps for each device (transistor) in the design
- Complete visibility inside the cell for power-grid and transistor
- Macro trigger information is used later by Power Analysis and Rail Analysis tools to automatically use current waveforms as per the trigger information. The Power Analysis software identifies the trigger condition saved in the EM power-grid view during vectorless or vector-based analysis and generates trigger files. The Rail Analysis

© 2013-2023 All Rights Reserved.

software processes the trigger files generated by Power Analysis and the EM power library of the custom macro generated by the Library Characterization engine, and performs full-chip dynamic IR drop analysis with custom macros operating under different functional modes.

The following is a sample of the log information generated during Power Analysis:

```
** INFO:  (VOLTUS_POWR-2061): Writing ASCII trigger file
DYNAMIC_POWER/trigger.txt
```

The trigger file contains instance names and when they will be triggered during simulation. The following is a sample auto-generated trigger file in the power analysis directory:

```
INST1 2.1995e-07 1 READ
```

The following is a sample of the log information generated during Rail Analysis:

```
INFO (VOLTUS_RAIL-5023): Found a dynamic tap current view for instance "INST1"
whose waveform is triggered at time 1e-07s.
```

- Optimized collapsed views are incrementally generated by performing RC reduction and collapsing (merging) the devices in the region with relatively small current consumption, (e.g. bit cells region in the memory). This helps improve IR drop analysis performance without sacrificing accuracy of any significant order. The optimized views can be used during IR drop analysis by selecting *Fast: Accurate* accuracy mode.

Power-Grid View Generation Flow for Macros from PGDB/xDSPF

You can use PGDB and xDSPF as input to library characterization for generating power-grid libraries in advanced node designs. This feature supports third-party extractors that create an extracted view of the design in the xDSPF format. In addition, library characterization supports the PGDB format netlist generated using the Quantus extractor. When you specify xDSPF or PGDB as inputs, GDS files are not required.

Tcl command for PGDB input:

```
set_pg_library_mode ...
set_advanced_pg_library_mode ...
    -pgdb_layermap ../pgdb_layermap.txt
  -pgdb_list_file ../voltus_pgdb_dynamic_1/cell1.hdr
```

#Specify the `-pgdb_layermap` and `-pgdb_list_file` parameters to generate PGV from an external PGDB.

The syntax of the PGDB header file is:

© 2013-2023 All Rights Reserved.

```
CELL cell_name
PGDB pgdb_directory
```

The PGDB directory contains the PGDB for all the power and ground nets of the cell. The PGDB directory has `cellname_Netname.*` files.

The syntax of the PGDB layermap file is:

#LibGen type	technology file layername	keyword	PGDB layername
metal	METAL_1	pgdb	metal1
via	VIA_1	pgdb	via1
metal	METAL_2	pgdb	metal2
via	VIA_2	pgdb	via2

Important

When generating PGV from Voltus-Fi, some of the layers are not required for digital analysis, but must be preserved to maintain connectivity. Voltus supports mapping of these layers to a special layer “connectivity” in the layermap file.

Example:

```
diff    connectivity    pgdb    tn_diff
diff    connectivity    pgdb    tp_diff
metal   metal1          pgdb    metal1
```

Tcl command for xDSPF input:

```
set_pg_library_mode ...
set_advanced_pg_library_mode ...
   -import_xdspf_list_file ../voltus_xdspf_dynamic_1/cell2.xdspf
   -xdspf_layermap ../voltus_xdspf_dynamic_1/layermap_dspf
```

#Specify the `-import_xdspf_list_file` and `-xdspf_layermap` parameters to generate PGV from an xDSPF netlist.

The syntax of the xDSPF layermap file is:

```
#-----------------------------
# layer      process
# type       layername
# ----------- ---------------
metal       METAL3
```


© 2013-2023 All Rights Reserved.

```
via         VIA3
metal       METAL2
via         VIA2
```

You must define layers in the stack from the top most layers to the bottom most layers. With reference to this layermap stack, the library characterizer will create the PGDB for the layers as defined in the xDSPF layermap file.

Multi-Mode Power-Grid View Generation

In the multi-mode PGV generation flow, the software allows you to save static or dynamic currents for multiple voltages and frequency per each functional mode in a single PGV. The software applies current from the PGV based on the specified mode, voltage, and frequency for different instances of the cell. This flow can be used in scenarios where multiple instances of the same cell are existing in a design, and each instance is operating at a different voltage and/or clock frequency.

To run this flow, specify the following keywords in the trigger file:

- `MODE <mode_name> static | dynamic` - the `static` keyword with the `MODE` keyword indicates static current behavior and the `dynamic` keyword with the `MODE` keyword indicates dynamic current behavior.

  Note: In the power analysis flow, you can specify the `set_power -static_mode` parameter to define the mode name or the custom label name associated with a cell or instance for computing instance current from PGV with static multi-voltage multi-frequency (MvMf) modes.

- `VOLTAGE` - specifies the voltage values at which simulation is done for each net.
- `FREQ` - specifies the frequency value at which simulation is done.
- `CUSTOM_LABEL` - specifies a label or tag to distinguish various voltage and frequency pairs across all modes.
- `SIM_DIR` - specifies the directory containing the simulation results corresponding to the specified voltage and frequency. The simulation directory (`SIM_DIR`) can point to the static or dynamic current files. If you specify the dynamic simulation data for the static mode, the software will derive the average current for each device and save it in the static PGV.
- `RON_FILE` - specifies the file containing the powergate RON values corresponding to the specified voltage and frequency. If `RON_FILE` is not specified, then the RON values are read from `SIM_DIR`.
- `RON_METHOD` – `RON_FILE` contains 3 possible RON values: `min`, `max`, `avg`. This keyword can be used to select one RON value. The default is `max`.

© 2013-2023 All Rights Reserved.

- `RON_THRESHOLD` – specifies the RON threshold value. The software compares the powergate RON value with the RON threshold to determine whether the powergate is ON or OFF.
- `OFF_NETS` - specifies the list of switched-nets inside the cell that are defined as off nets. If a net is specified as the off net, the net is dropped when the PGV is loaded for top-level rail analysis. This is referred to as the user-specified off nets. However, if the RON of all the powergates is greater than `RON_THRESHOLD`, the switched net is treated as off net. This is referred to as the derived off nets.
- `CELL_DECAP_FILE <file>` - specifies different cell decoupling capacitance values for multiple voltages in each mode. The format of the cell decap file is:

  ```
  UNIT CAP  <Cap unit>  (default is fF)
  UNIT RESISTANCE <Res unit> (default is Ohm)
  CELL <cellname>
  NET <netname1> <capacitance> <resistance>
  ```

- `STATIC_INTERNAL_POWER {<NET_NAME> <VALUE> ….}` - specifies to store user-defined net-based internal power in the static PGV. The value is in A/MHz.

  If `STATIC_INTERNAL_POWER` is defined in the PGV, the power for macro is computed as below:

  - **Leakage**: obtained from PGV. If not defined in PGV, it is obtained from .lib.
  - **Internal**: STATIC_INTERNAL_POWER * frequency
  - **Switching**: sum of all output cap * transition density

Example 1: Dynamic PGV Generation

Following is an example of the trigger file for dynamic PGV generation:

```
CELL cell1
RON_THRESHOLD = 5
MODE READ dynamic
CONDITIONAL_INPUT = ( !CEN & !WEN )
CONDITIONAL_PIN clk4 { RISE }

CUSTOM_LABEL One
VOLTAGE { VDDG 0.6 }
FREQ 100
RON_FILE ./ronfile1.txt
RON_METHOD min
OFF_NETS { VVDDG }
```


© 2013-2023 All Rights Reserved.

```
CUSTOM_LABEL two
VOLTAGE { VDDG 0.9 }
FREQ 100
RON_FILE ./ronfile1.txt
END_MODE
.....
END
```

The file specified with the `RON_FILE` keyword contains multiple on-resistance (Ron) values, that is minimum, maximum, and average, for power gates. The following is a snippet of the RON file:

```
D-Term-Net     S-Term_Net      Avg      Max      Min      Gate

 VDD            VDDG            1        2        3      X2_unmatched

 VDD            VDDG            1        2        3      X3_unmatched

 VDD            VDDG            1        2        3      X5_unmatched
```

The following is a snippet of the additional information related to the multi-mode PGV generation that is added in the library report:

```
RON_THRESHOLD = <value>
MODE = <mode/sub_mode>
RON_METHOD = <min | avg | max>
<Net>  <#Powergates > <#Off Powergates> <Status>
```

The following is a snippet of the library report:

```
...
...
MODE = READ
CUSTOM_LABEL = one
RON_METHOD = min
Net               #Powergates         #Off Powergates          Status
VDD                3                    3                      OFF (Derived)
VDDG               3                    2                      ON

MODE = READ
CUSTOM_LABEL = two
RON_METHOD = avg
Net               #Powergates         #Off Powergates          Status
VDD                3                    2                      OFF (Specified)
VDDG               3                    2                      ON
……..
```

This trigger file information is used later by Power Analysis and Rail Analysis to automatically use current waveforms as per the trigger information. During power analysis, you can schedule the desired mode in the following two ways

© 2013-2023 All Rights Reserved.

- Specify the custom label corresponding to a voltage and frequency pair.
- Specify the mode name. In this case, the software selects the closest voltage and frequency pair for the instance based on the instance voltage and fastest frequency.

The Power Analysis engine supports multiple modes/custom labels in the switch-pattern if they are generated without RON files and off nets. However, if a switch-pattern has a mode generated with RON file or off nets, then the software ignores such modes and only uses the previous modes.

In the following example, the multi-voltage multi-frequency PGV has the following 3 modes, that is, two modes without a RON file and off nets, and the third mode with a RON file and off nets:

```
Read  [ No off nets and RON files in the trigger file ]
   CUSTOM_LABEL R1
   CUSTOM_LABEL R2

Write [  No off nets and RON files in the trigger file ]
       CUSTOM_LABEL W1
       CUSTOM_LABEL W2

HV_IVD [ With off nets and RON files in the trigger file ]
       CUSTOM_LABEL H1
       CUSTOM_LABEL H2
```

The following commands describe the use model for mixed multiple modes (with/without RON file/off nets):

- `set power -instance -dynamic_switch_pattern { R1 H1 W1 }`

  Uses the first pattern “`R1`”.

- `set power -instance -dynamic_switch_pattern { H1 R1 W1 }`

  Uses the first pattern “`H1`”.

- `set power -instance -dynamic_switch_pattern { R1 W1 H1}`

  Uses “`R1`” for the 1st cycle and “`W1`” for the 2nd cycle

The log information generated during Power Analysis reports the label and the corresponding voltage/frequency pair for the specific instance, as shown below:

```
Instance 'I1' is using mode 'M1'
```

During Rail Analysis, the specified mode is triggered, and the off nets are removed from the power-grid.

© 2013-2023 All Rights Reserved.

Example 2: Static PGV Generation

Following is an example of the trigger file for static PGV generation:

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL cell2
LEAKAGE_CURRENT { VSS 2p VDD 17n }

MODE_NAME  READ static
CONDITIONAL_INPUT =  !CEN
CONDITIONAL_PIN CLK { RISE }
CONDITIONAL_PIN_REFERENCE_TIME 0.0

CUSTOM_LABEL A1
VOLTAGE { VDD 0.9 VDD_LP 0.9 VSS 0 }
CAPACITANCE 10
USER_PWL_FILE /../pwl_1.txt TIME_FACTOR=2 CURRENT_FACTOR=2
STATIC_INTERNAL_POWER { VSS 1p VDD 17n VDD_LP 1 }
CELL_DECAP_FILE cell_decap_read.txt
SWITCHBIT_WINDOW {-3 4} VSS
SWITCHBIT_WINDOW {4 8} VDD

CUSTOM_LABEL A2
VOLTAGE { VDD 0.7 VDD_LP 0.7 VSS 0 }
CAPACITANCE 50
USER_PWL_FILE /../pwl_1.txt
STATIC_INTERNAL_POWER { VSS 2p VDD 27n VDD_LP 1 }
CELL_DECAP_FILE cell_decap_read.txt
END_MODE

MODE_NAME  WRITE  static
...
END_MODE
END
```

Additional Reports

Tap Current Report

The macro PGV generation flow generates a report called as `tapSnapReport.txt` that gives detailed information about current assigned to taps/devices. The report allows you to

© 2013-2023 All Rights Reserved.

verify the current and capacitance values assigned to each device, and the tap current distribution for each interface node.

The format of the file is:

```
Cell Name:
Net Name:
Layer Name:
NODE    DEVICE(X,Y)   NODE(X,Y)   CURRENT   CAPVALUE  TAP NAME
SWITCH_NODE(X,Y)
....
nodeId : NODE % of taps connected
```

Here,

- `Cell Name` is the macro cell name
- `Net Name` is the power/ground net that supplies power to the macro cell
- `Layer Name` is the via layer name. The current taps will be attached to the layer that is below the specified via or contact layer.
- `NODE` is the unique node ID
- `DEVICE(X,Y)` is the X,Y co-ordinate of the device specified in the netlist file
- `NODE(X,Y)` is the X,Y location of the node in GDS. This location is relative to the LEF coordinate space.
- `CURRENT` is the current value for the node
- `CAPVALUE` is the capacitance value for the node
- `TAP NAME` is the name of the device
- `NODE %` of taps connected is the % of taps connected to a node. This can be used to determine the list of nodes that are heavily connected.
- `SWITCH_NODE(X,Y)` is the X,Y co-ordinate of the switched net connected to the power/ground (PG) net. This column is displayed only if the PG net is connected to a switched net. The switched net will have a separate section for those taps where it is not connected to the PG net.

© 2013-2023 All Rights Reserved.

Multi-Voltage Capacitance Characterization

Voltus generates PGVs that contain voltage-dependent capacitance table for the specified cell. This feature allows you to use accurate cell capacitance at different supply voltages in power-up analysis, and in different voltage domains in dynamic analysis.

To enable this feature, specify the following parameters during PGV generation:

- `set_pg_library_mode -enable_multi_voltage_cap_generation {true | false}` - computes capacitance values over a range of input voltages, including the nominal voltage. It generates 10 values from 0.1 to 1.3 times the vdd voltage.

  Note: You can use the `set_advanced_pg_library_mode -stdcell_characterization_voltage_value {val1 val2 val3 ...}` parameter to specify capacitance characterization voltages when generating standard cell PGVs containing voltage-dependent capacitance table for the specified cell. This parameter allows you to compute capacitance based on the user-defined voltage values. This parameter must be specified with the `-enable_multi_voltage_cap_generation true` parameter. If this parameter is not specified with `-enable_multi_voltage_cap_generation`, capacitance is calculated based on the software-derived voltage range. When the `-stdcell_characterization_voltage_value` parameter is specified, the library generation engine prints the capacitance values and corresponding voltages in the report file, and the rail analysis engine prints the capacitance used for each instance.

  The following example illustrates the use model of user-defined cap characterization:

```
set_pg_library_mode  -ground_pins VSS -power_pins  {VDD 0.9 TVDD 0.9 } \
-celltype stdcells -enable_multi_voltage_cap_generation true
set_advanced_pg_library_mode -stdcell_characterization_voltage_value {1.0 1.2
1.5 2.0 3.3}
```


© 2013-2023 All Rights Reserved.

The following is a snippet of the standard cell report file:

```
==========================================================================
Cell                PowerNets       Capacitance       Load Capacitance      PowerGrid Views
--------------------------------------------------------------------------------------------------------------------------

AND2X8
                    VDD(0.9000)     1.8e-14           0.0e+00          EARLY(1 taps) EM(1 taps) IR(1 taps)
                    VDD(1.0000)     1.747e-14
                    VDD(1.2000)     1.742e-14
                    VDD(1.5000)     1.744e-14
                    VDD(2.0000)     1.804e-14
                    VDD(3.3000)     2.075e-14
                    VSS(0.0000)     1.8e-14           0.0e+00          EARLY(1 taps) EM(1 taps) IR(1 taps)
                    VSS(1.0000)     1.747e-14
                    VSS(1.2000)     1.742e-14
                    VSS(1.5000)     1.744e-14
                    VSS(2.0000)     1.804e-14
                    VSS(3.3000)     2.075e-14
 Cell Type = STDCELL                                    Cell Status: PASS
INFO: PGV contains voltage dependent cap table for this cell

--------------------------------------------------------------------------------------------------------------------------
```

Command Example:

```
set_pg_library_mode -celltype stdcells -cell_list_file cellfile -
extraction_tech_file ./../qrcTechFile -lef_layermap ./../lefdef.layermap -
temperature 125 -power_pins { vcc 1.26 vccpg 1.26 vccr 1.26 } -ground_pins {
vss } -powergate_parameters {{powercell_pg1 vcc vccpg} {powercell_pg1 vcc vccr}
{powercell_pg2 vcc vccpg} {powercell_pg2 vcc vccr}} -spice_corners { tt } -
spice_models spice_model/libspice -spice_subckts { powercell_pg.net
powercell_pg2.net } -enable_multi_voltage_cap_generation true
```

The PGV report includes the following information:

```
INFO: PGV contains voltage dependent cap table for this cell
```

- `check_pg_library -report_detail_multiple_voltage_cap` *`filename`* - generates a capacitance table that contains the detailed capacitance values of the given power-grid library.

Command Example:

```
check_pg_library -report_detail_multiple_voltage_cap ../powergate_stdcells.cl
```


© 2013-2023 All Rights Reserved.

The following is a snippet of the capacitance table file (`powergate_stdcells.cap_table`):

```
==============================================================================
CellName                        NetName          Voltage       Capacitance
------------------------------------------------------------------------------
powercell_pg_m_r                vccm            0.1000          4.586e-14
powercell_pg_m_r                vccm            0.2825          4.102e-14
powercell_pg_m_r                vccm            0.4650          3.794e-14
powercell_pg_m_r                vccm            0.6475          3.577e-14
powercell_pg_m_r                vccm            0.8300          3.414e-14
powercell_pg_m_r                vccm            1.0125          3.284e-14
powercell_pg_m_r                vccm            1.1950          3.177e-14
powercell_pg_m_r                vccm            1.3775          3.087e-14
powercell_pg_m_r                vccm            1.5600          3.010e-14
powercell_pg_m_r                vccm            1.7425          2.942e-14
powercell_pg_m_r                vccaux          0.1000          3.229e-14
...
...
```

Characterizing Decap, Filler, and Damping Decap Cells

Decoupling capacitance on power-grid helps in reducing dynamic IR drop. When creating the power-grid library, you must specify the list of decap and filler cells for decap optimization:

- **Decap cells** are cells that have explicit decoupling capacitance for use in optimization. These cells are nearly equivalent to feedthrough cells, except that there is typically an active device between the power and ground nets to provide extra decoupling capacitance. The coupling capacitance is stored in the power-grid library, and is used in top-level IRdrop analysis.
- **Filler cells** are cells that have no GDS data. The geometrical data contained in the LEF files for these cells is considered complete for the purposes of determining the promoted views. The data contained in the filler cells is extracted at the top level.

You can specify the list of decap and filler cells using the following methods:

- *Set PG Library Mode - Basic* form
- `set_pg_library_mode -decap_cells <cell_name> -filler_cells <cell_name>`

For Early/IR/EM view generation, the cap value for all decap cells is automatically calculated using the SPICE netlist. You can also specify a value using the external LibGen include file (`set_advanced_pg_library_mode -libgen_command_file`). For Tech views, you

© 2013-2023 All Rights Reserved.

can specify the default capacitance value for all cells (`set_pg_library_mode -default_area_cap`).

During standard cell cap characterization, the tool does not take into account series resistance that connects to device coupling capacitance between the power and ground nets. The reason being that this resistance is too small to affect IR drop analysis. It only calculates coupling capacitance for each tap present in a decap cell. However, there are also some special damping decap cells which are used in the design to control resonance frequency. These decap cells have high series resistance which needs to be modeled. You can specify the following command to model this resistance, that is, to write an additional series resistance value with the coupling capacitance value for each VDD/VSS pin:

- `set_advanced_pg_library_mode -damping_decap_cell_list { cell1 cell2 ..} -damping_decap_frequency value`

The cells in the damping decap list will be characterized at the damping decap frequency. Damping decap cells are characterized at a very low frequency. Default is 200K Hz.

Generating Power-Grid Library for a Flip Chip Design

When running power and rail analysis for a flip chip design, you can create voltage sources for the flip chip to be used in the Rail Analysis flow. The bumps are modeled with several points in a resistor network, and hence avoid false EM violations.

The LEF file of the flip chip bump can be used as a reference. Bumps are usually octagonal, although sometimes they are represented as squares. The following is an example of the bump LEF file:

```
VERSION 5.6 ;
BUSBITCHARS "[]" ;
DIVIDERCHAR "/" ;
UNITS
 DATABASE MICRONS 1000 ;
 END UNITS
MACRO BUMP
 CLASS COVER BUMP ;
 FOREIGN BUMP -49.45 -49.45 ;
 ORIGIN 49.45 49.45 ;
 SIZE 98.9 BY 98.9 ;
 PIN PAD
    DIRECTION INOUT ;
    USE SIGNAL ;
    PORT
      LAYER AP ;
        #RECT -49.45 -49.45 49.45 49.45 ;
        POLYGON -20.49 -49.45 20.49 -49.45 49.45 -20.49 49.45
        20.49 20.49 49.45 -20.49 49.45 -49.45    20.49 -49.45 -20.49 ;
    END
  END PAD
```


© 2013-2023 All Rights Reserved.

```
 END BUMP
END LIBRARY
```

Important

In the LEF file, a polygon shape is used to create an octagonal bump, but the corresponding coordinates that would have been used for a square are commented out.

The following are the steps for PGV characterization of a flip-chip design:

1. Create a file called `bump.padfile`. This file contains one line, the MACRO name of the bump from the LEF, as shown below:

   ```
   BUMP
   ```

2. Create a file called `bump.srcfile`, as shown below:

   ```
   CELL BUMP
        NET PAD
            PORT {
           AP -49.45 -49.45 49.45 49.45
         }
   ```

   The `CELL` and `NET` names in the voltage source file must match that of the bump LEF file. The `NET` name is the `PIN` name from the LEF. Similarly, the port layer name in the voltage source file (AP) is the same layer from the LEF. The commented-out square coordinates in the LEF example can be used to specify the coordinates of the PORT shape which should be a square that encloses the octagonal bump.

3. Create the bump power-grid view. You can create a technology-based bump PGV or a standard cell based PGV. When generating a standard cell based PGV, you can:

   - specify separate pad names for power and ground pins
   - specify a common pad name for power and ground pins
   - specify x, y locations of the voltage sources to be connected to the pad cells

4. Run Rail analysis.

   When running rail analysis, use the `bump.padfile` in the `set_power_pads` command. The same padfile can be used for any rail, as shown below:

   ```
   set_power_pads \
     -net VDD \
     -format padcell \
     -file bump.padfile
   set_power_pads \
     -net VSS \
   ```


© 2013-2023 All Rights Reserved.

```
-format padcell \
-file bump.padfile
```

All bumps will be recognized as voltage sources, with multiple points inside the bump shape.

The following table illustrates the Tcl script for generating different types of bump PGV:

Tech PGV	Standard Cell PGV - separate pad names for P/G pins	Standard Cell PGV - common pad name for P/G pins (without Spice netlist of bump)	Standard Cell PGV - pad x, y locations
set_pg_library_mode -celltype techonly\ -extraction_tech_file\ cworst.qrcTechFile\ -lef_layermap layer.map\ -power_pins {VDD 0.81}\ -ground_pins {VSS}\ -cell_list_file bump.list\	set_pg_library_mode \ -celltype stdcells\ -extraction_tech_file\ cworst.qrcTechFile\ -lef_layermap layer.map\ -power_pins\ {VDD 0.81 PAD_VDD}\ -ground_pins\ {VSS PAD_VSS}\ -cell_list_file\ bump.list\ -spice_models\ ../model.lib\ -spice_corners SS\ -spice_subckts\ {Io_pad.spice}	set_pg_library_mode \ -celltype stdcells\ -extraction_tech_file\ cworst.qrcTechFile\ -lef_layermap layer.map\ -power_pins {VDD 0.81}\ -ground_pins {VSS}\ -cell_list_file\ bump.list\	set_pg_library_mode \ -celltype stdcells\ -extraction_tech_file\ cworst.qrcTechFile\ -lef_layermap layer.map\ -power_pins {VDD 0.81}\ -ground_pins {VSS}\ -cell_list_file\ bump.list\
set_advanced_pg_library_mode\ -common_supply_pins {PAD}		set_advanced_pg_library_mode\ -common_supply_pins { PAD }	set_advanced_pg_library_mode\ -source_location_file\ bump.srcfile\ -libgen_command_file\ libgen.inc\
generate_pg_library \ -output techonly.pgv	generate_pg_library \ -output stdcell.pgv	generate_pg_library \ -output stdcell.pgv	generate_pg_library \ -output stdcell.pgv

© 2013-2023 All Rights Reserved.

Tcl command:

Following is a sample script, called `create_bump_pwrgrid.ss0p81v.tcl`:

```
read_lib -lef tech.lef \
         BUMP.lef

set_pg_library_mode \
        -celltype techonly \
        -extraction_tech_file cworst.qrcTechFile \
        -lef_layermap lef_layer.map \
        -power_pins {VDD 0.81} \
        -ground_pins {VSS} \
        -cell_list_file bump.list

set_advanced_pg_library_mode \
        -source_location_file bump.srcfile \
        -libgen_command_file libgen.inc

generate_pg_library \
        -output fast_bump.ss_0p81v
```

Here, the `bump.list` file just contains the bump name and is the same as the `bump.padfile`. The following is a sample `libgen.inc` file (the cell and net name should match the bump LEF):
`cell_common_supply_names cell BUMP nets {PAD}`

Generating Open Access Based Library

Voltus allows you to specify an OpenAccess (OA) database for Library Generation, instead of converting the design to the GDSII database format. When you specify an OA database with the layout data, the software generates a GDS file for the specified OA library. This GDS file is used to generate the cell library. The layermap file for GDS will be automatically created from the specified OA file. The automatic layermap file will contain arbitrary GDS layer numbers corresponding to layer names in OA. For OA setup, you must specify the first OA library as the technology library.

Note that the LEF files will be required to run this flow if the macro OA database does not have an abstract view (view containing LEF ports information).

To run this flow:

1. Specify the `read_lib -oaRef {technology_library_name cell_library_name}` command.
2. Save a `cds.lib` file in the home directory. This file contains the OA database for technology and cell library.

Note: You do not need to specify a GDS layermap.

© 2013-2023 All Rights Reserved.

Tcl command:

Following is a sample script, called `create_bump_pwrgrid.ss0p81v.tcl`:

```
read_lib -oaRef { tech_lib1  cell1 }

set_pg_library_mode \
-ground_pins VSS \
-cell_list_file /../cell.list \
-power_pins {VDD 0.9 VDD_LP 0.9} \
-celltype macros \
-spice_corners {tt tt_sr} \
-spice_subckts /../cell1.net \
-lef_layermap /../lef_layermap.txt \
-stop@via CONT \
-current_distribution {dynamic_simulation /../input_trigger.txt} \
-spice_models {/../spice90.l /../spice_90g.l.special} \
-extraction_tech_file /../c_worst.tch

set_advanced_pg_library_mode \
-techgen_dir /../tech \
generate_pg_library \
-output .
```


© 2013-2023 All Rights Reserved.

Checking PowerGrid Library and Report

You can check the power library and generate a report. For details, see `check_pg_library` in the *Voltus Text Command Reference Manual*.

The below report identifies the Cell Name, associated power pins and their voltage, capacitance, generated views and PASS/FAIL cell status. The power gate cells are identified and characterized Ron, Idsat and Ileakage values are printed. The cells without any power-grid views or power nets/pins are tagged with FAILED status.

Examples:

Standard Cells Power-Grid Library Report and its description

```
==============================================================================
Library path name: .
Library name: stdcells
Hierarchical library: NO
User login: tom
Date: 2013-Sep-24 00:26:43 (2013-Sep-24 07:26:43 GMT)
ToolIdentifier: LIBGEN
Application version: voltus_libGen Version v13.20-b048_1 (09/18/2008 05:42:15)
Number of cells in library: 2
==============================================================================
Cell      PowerNets    Capacitance   PowerGrid Views
------------------------------------------------------------------------------|

HDRSID0
          VSS(0.0000)   2.9e-15    Early(2 taps)  EM(2 devices) IR(2 devices)
          VDD(0.8400)   1.8e-15    Early(6 taps)  EM(6 devices) IR(6 devices)
          TVDD(0.8400)  2.9e-15    Early(12 taps) EM(12 devices) IR(12 devices)

 Cell Type = STDCELL
 Total power-net series-resistance = 0
 Total ground-net series-resistance = 0                        Cell_Status: PASS
------------------------------------------------------------------------------|

RSDFCD1
          VSS(0.0000)   5.9e-15     Early(2 taps) EM(2 devices) IR(2 devices)
          VDD(0.8400)   4.8e-15     Early(1 taps) EM(1 devices) IR(1 devices)
          TVDD(0.8400)  1.1e-15     Early(1 taps) EM(1 devices) IR(1 devices)


Cell Type = STDCELL
Total power-net series-resistance = 10206
Total ground-net series-resistance = 8000                     Cell_Status: PASS
------------------------------------------------------------------------------|
```

The following steps are needed to view a power-grid library cell:

1. Select *Power & Rail - Textual Reports - PowerGrid Library*.

© 2013-2023 All Rights Reserved.

The following form appears:

2. Click the browse button next to the *Power Library* field, and select a library from the PowerGrid Library browser window.
3. Select the *View Power Library* option
4. Select *All Cells* to view all cells, or select *Cell* and specify the cells to view in the library report.
5. Click *Apply* or *OK*.

   The power-grid view library report appears in the *Console* tab.

© 2013-2023 All Rights Reserved.

Viewing and Debugging PowerGrid Library

You can use the Cell Viewer form to display the content of power-grid cell libraries. The Cell Viewer allows you to view and debug multiple power-grid libraries.

Cell Viewer

The default views for different power-grid view types are:

Tech

- Displays the basic port information of a cell.

Early

- Geometrical View - displays the LEF of a cell. The types of geometrical views are: PORT_GEOM, OBS_GEOM, PROMOTED_GEOM and OPTIMIZED_GEOM views.
- Port View - displays more accurate port and tap information of a cell when . This information is available in the PORT_POWER view of the library.

© 2013-2023
All Rights Reserved.

- Capacitance Views - displays the grid and device capacitance information of a cell. The types of capacitance views are: GRID_CAP and DEVICE_CAP views. These views are net specific.

EM

- Detailed View - displays the detailed power-grid information for all nets. This information is available in the DETAILED_POWER view of the library.
- Tap Current View - displays tap information of a cell. All the tap currents are shown in the viewer drawn in 8 colors for 8 linear filters according to the min max value. This information is available in the TAP_CURRENT view of the library.
- Dynamic Tap Current View - displays dynamic tap currents for each interval. The total current plot is net based, and supports multiple trigger view in a single PGV. This information is available in the DYN_TAP_CURRENT view of the library.
- Powergate View - displays the power switch information of a power-gate cell. This information is available in the POWERGATE view of the library.

IR

- Collapsed View - displays the collapsed current taps for the bit-cells inside the memory. This information is available in the TAPCOLLAPSE_POWER view of the library.

You can navigate to a cell using the Library Browser, which is on the left side of the Cell Viewer. The details of the cell are displayed in the Display Area, which is on the right side of the Cell Viewer. You can click the + button to expand a cell and view the default cell views. Each view has a check box for displaying the content specific to the selected view. You can view either one or multiple views overlapped over each other.

To view the tap current of a cell:

1. Select Power & Rail - Library Cell Viewer, and click on the PGV tab.

   The Cell Viewer form appears.

2. Select *Open Power Grid Library* from the *File* menu to open a cell library.
3. Right-click on the *TAP_CURRENT* view of a cell and choose *Current Filter* from the context menu.

© 2013-2023 All Rights Reserved.

The Tap Current Filter form appears.

4. Select a net from *Tap Current Nets*. The *Min* and *Max* values can be entered to change this from the default. Any values falling outside this range will not be viewed. After the range is set, selecting the *Linear* button will update the filters with the new values. Selecting the *Auto* button will set the filters back to the default.

5. Click *Plot*.

Tip

You can customize the colors for the layers by using the *IR Drop & EM Color Selection* form. To access this form, click the *All Colors* button on the main window to open the *Color Preferences* form. Then, select the *View-Only* tab, and select *IR Drop & EM*, in the *Multi-Color Layers* section.

© 2013-2023
All Rights Reserved.

To view the dynamic tap current of a cell:

1. Select *Power & Rail - Library Cell Viewer*, and click on the *PGV* tab.

   The Cell Viewer form appears.

2. Select *Open Power Grid Library* from the *File* menu to open a cell library.

3. Right-click on the *DYN_TAP_CURRENT* view of a cell and choose *Show Waveform* from the context menu.

   The View Waveform form appears.

4. Select a net/trigger condition and click *Plot* to view the dynamic current waveforms using SimVision waveform viewer interface.

To view the power-grid for a single net:

1. Select *Power & Rail - Library Cell Viewer*, and click on the *PGV* tab.

   The Cell Viewer form appears.

2. Select *Open Power Grid Library* from the *File* menu to open a cell library.

© 2013-2023 All Rights Reserved.

3. On the *Options* menu, click *Net Based View - Show Popup Automatically*. If this menu item is selected, clicking on a view displays the following dialog box automatically. By default, this menu item is not selected.

Note: Alternatively, you can view this dialog box by right-clicking on a view to see the short-cut menu. Then, click the view name option on the shortcut menu. The dialog box displays the power, ground, and signal nets. Select a net and click *Plot* to view a single net.

4. Click on the cell name to return to the whole view mode, and display the selected views for the cell name.

To view the grid and device capacitance information of a cell:

1. Select *Power & Rail - Library Cell Viewer*, and click on the *PGV* tab.

   The Cell Viewer form appears.

2. Select *Open Power Grid Library* from the *File* menu to open a cell library.

© 2013-2023
All Rights Reserved.

3. Right-click on the *GRID_CAP* or *DEVICE_CAP* view of a cell and choose *Grid Cap Filter* or *Device Cap Filter* from the context menu.

   The Grid Cap Filter or Device Cap Filter form appears.

4. Select a net from *Power Ground Nets*. The *Min* and *Max* values can be entered to change this from the default. Any values falling outside this range will not be viewed. After the range is set, selecting the *Linear* button will update the filters with the new values. Selecting the *Auto* button will set the filters back to the default.

5. Click *Plot*.

Tip

You can customize the colors for the layers by using the *IR Drop & EM Color Selection* form. To access this form, click the *All Colors* button on the main window to open the *Color Preferences* form. Then, select the *View-Only* tab, and select *IR Drop & EM*, in the *Multi-Color Layers* section.

To view the power switch information of a power-gate cell:

1. Select *Power & Rail - Library Cell Viewer*, and click on the *PGV* tab.

   The Cell Viewer form appears.

2. Select *Open Power Grid Library* from the *File* menu to open a cell library.

3. Right-click on the *POWERGATE* view of a cell and choose *Power Gate* from the context menu.

   The Power Gate List form appears.

Gate Name	Enable Pin	Ron	Idsat	Ileakage	Voltage
PG_SWCIN_0	SWCIN	66.1622	0.00465682	3.05472e-08	0.600000
PG_SWCIN_0	SWCIN	12.5824	0.0284678	6.71994e-08	0.840000
PG_SWCIN_0	SWCIN	6.41622	0.0671994	1.44892e-07	1.080000
PG_SWCIN_0	SWCIN	5.36081	0.0888	2.12084e-07	1.200000
PG_SWCIN_0	SWCIN	4.30811	0.13394	4.55766e-07	1.440000
PG_SWCIN_0	SWCIN	3.81622	0.18093	9.916e-07	1.680000

© 2013-2023 All Rights Reserved.

This form displays the properties of the power switch.

© 2013-2023 All Rights Reserved.

Validating a PowerGrid Library

Voltus allows you to validate a macro PGV, and determine if there are any issues in the PGV. You can use the `validate_pg_library` command that takes a macro PGV and Liberty file as input, and performs IR Drop based validation by running static and dynamic power and rail analysis. In this flow, Library Generation will read the input PGV and generate a dummy DEF file that has an instance of the macro (one-cell design). It will then run static/dynamic power and rail analysis on this design. You can analyze the power and rail results to check for hotspots, connectivity, and current distribution inside the PGV.

Tcl command:

```
validate_pg_library \
    -cell pll \
    -liberty ../data/libs/pll.lib\
    -power_grid_library_path
             ../data/pgv_dir/macro_pgv/macros_pll.cl \
    -extraction_tech_file ../data/qrc/gpdk090_9l.tch \
    -tech_lef_file ../data/lef/gsclib090_tech.lef \
    -output validate_pg_output_static \
    -rail_analysis_type static
```

The output directory contains the following sub-folders:

- `design_output` - This folder contains the design files, such as `.def`, `.lef`, `.sdc`, and verilog, that are required to run power and IRdrop analysis.
- `power` - This folder contains the output files related to power analysis, such as power.rpt and current files (`.ptiavg`).
- `rail` - This folder contains the output files related to rail analysis, such as state directories for each net, and auto-generated GIF files and rail summary reports in each net directory.

© 2013-2023 All Rights Reserved.

11

Static Power, IRdrop and EM Analysis

- Static Power Calculation on page 129
  - Static Power Calculation Overview on page 129
  - Vector-based Average Power Calculation on page 139
  - Propagation-based Average Power Calculation on page 140
  - Static Power Analysis Flow on page 144
  - Power Map File for Thermal Analysis on page 151
  - Voltus Thermal Model Generation for Thermal Analysis on page 158
  - Viewing Static Power Analysis Results on page 160
  - Interactive Queries of Power Data on page 173
  - Debugging Instance Power on page 174
  - Saving and Restoring Power Database on page 177
- Resistance Analysis on page 179
- Static Rail and EM Analysis on page 195
  - Static IRDrop and EM Analysis Overview on page 195
    - EM Modeling using Voltus Format on page 212
    - EM Modeling using QRC Technology File on page 219
  - Static Rail Analysis Setup and Run on page 226
  - Viewing Static Rail Analysis Results on page 231
  - Generating Text Reports on page 244
  - Power-Grid Integrity Reports on page 247
  - Layer-Based IR Drop Report on page 249

- HTML Reports on page 250
- Constant Current Source Based IR Analysis on page 254
- Hotspot Debugger on page 255

Voltus IC Power Integrity Solution offers static power, IRdrop and electromigration analysis. Some of the features include the following:

- Signoff quality power-grid extraction to take into account various manufacturing effects like erosion, wire edge enlargement, temperature, dishing, slotting, cladding etc.
- Support for RDL and full-chip GDS for power-grid extraction
- XD and HD accuracy modes for analysis
- Various interactive power-grid analysis plots like IRdrop, current density, current across resistors, device tap currents, drop across package, instance based voltage, instance based effective resistance, etc.
- CPF Support
  - Power domain definition
  - Power domain operating conditions
  - Off power-switch nets
- Highlight the least effective resistance path for the instance
- Result browser report
- Interactive query
- HTML reports
- GUI and TCL command interface

Static Power Calculation

Static Power Calculation Overview

Type of power (internal, leakage, switching)

Static Average Power is consumed in three basic ways in integrated circuits:

- **Switching power**, which is the power consumed in the charging and discharging of interconnect capacitances. In most cases, this type of power consumption dominates because of large drivers having to drive large capacitive loads.

  $P = 0.5*C_L V^2 F*A$
  where C_L is the output capacitive loading, V is the voltage, F is frequency, and A is the average switching activity either from VCD or computed.

- **Internal Power**, which is the power consumed in charging and discharging of interconnect and device capacitances internal to cell. Internal power can be divided into two parts:

  - Pin Power
  - Arc Power

  Internal power is calculated by using the internal power tables provided in the .lib, which capture the characterized internal power over a range of input slew rates and external loading. The tables reflect the combination of both the internal switching and internal feedthrough power. Tables are generated as a result of spice simulation during library characterization. If k-factor power scaling parameters (for process, temperature, and voltage) are specified in the .lib file, the power engine will take them into consideration when calculating internal power (Note: timing related scaling factors are not handled by the power engine).

- **Leakage power**, which is the power consumed by devices when they are not switching. It includes state-dependent leakage, which is leakage that depends on the state of the gate, that is, whether a transistor is on or off. This value comes from the .lib file if it exists. If k-factor power scaling parameters (for process, temperature, and voltage) are specified in the .lib file, the power engine will take them into consideration when calculating leakage power (Note: timing related scaling factors are not handled by the power engine).

Definition of activity, duty cycle, and transition density

Activity means the probability of a particular net in design switching from 0 ->1 or 1 -> 0 in one clock cycle. For instance, if an activity of net or instance is 0.1 then the power engine assumes that net or instance will switch from 0->1 or 1->0 once every ten clock cycles.

For the above diagram,
`Activity = (Number of (0->1 or 1->0) transitions / Number of clock cycles) = 2/5 = 0.4`

Duty Cycle means the probability that a signal net has the value of 1. For instance, if signal of a net is 1 for 2ns in the total simulation time of 10ns, then duty cycle of the net is 0.2. The duty cycle of the signal in the previous diagram is 0.5 (2.5/5). However, if a signal is Z or X for some time and 0 for the rest of time, then duty cycle of signal is 0.

Transition Density means number of times signal toggle from 0->1 or 1->0 in 1 second. For the previous diagram and assuming one clock cycle is 4ns, then Transition Density = 1e+08 (2/20ns)

How PM calculates internal, leakage and switching power including state dependency

The power calculation methods employed inside the power engine are split into 4 components:

- State dependent internal power associated with input pins
- State dependent internal power associated with output pins
- State dependent leakage power
- Switching Power due to charging or discharging the net loading on the output pins.

When looking at the output power numbers generated by the power engine, the following information is reported in the power file for each instance in the design:

- Instance Name
- Internal Power = sum of (1) and (2) above.
- Switching Power = (4) above.
- Total Power
- Leakage Power = (3) above.
- Cell-type Name

The ordering was chosen to be consistent with previous tools used in the industry. The following sections describe how the power engine calculates the power for each instance based on each of the above 4 components.

State Dependent Internal Power (input pins)

Inputs can have several sets of power table pairs, each associated with a `when' clause that specifies the logical condition of inputs that the tables apply to. A call to the table lookup function utilizes a procedure that returns a weighted sum of the energies based on the `when' clause functions and the signal activity.The weighting of energies is similar to the propagation of static probabilities (duty cycles):

$$y = f(x_0, x_1, \ldots, x_n)$$
$$P(y) = P(x_i) * P(y|_{x_i=1}) + (1 - P(x_i)) * P(y|_{x_i=0})$$

The procedure for the weight calculation is similar. However one complication of the energy weighting is that the coverage of the `when' clauses might not be complete. For example, one set of state dependent tables includes only two `when' clauses:

```
when : "A & !B";

when : "!A & B";
```

This set of clauses does not account for cases where A and B are either both high or both low. In normal operation, neither of these conditions may appear, so the incomplete coverage may not matter. However, the transition density data is static and lacks signal correlation; the conditions not included in the `when' clauses should be accounted for, or else the internal

power will be underestimated. Scaling can resolve this. But it turns out that the most common situation for incomplete clauses is in memories, where assuming the energy to be 0 is the more correct thing to do. As a result, the power engine assumes energy of 0 for missing clauses.

Implementation for state-based internal power on the inputs is straightforward. The implementation for the internal power contribution from a single input port is:

$$inputEnergy = \frac{1}{2}\left(port \rightarrow riseEnergy() + port \rightarrow fallEnergy()\right)$$

For multiple input ports, each port has a set of energy tables, instead of just one pair, and each pair includes a `when' clause, from which a probability can be calculated:

$$inputEnergy = \frac{\sum_i prob(port \rightarrow when(i)) * (port \rightarrow when(i) \rightarrow riseEnergy() + port \rightarrow when(i) \rightarrow fallEnergy())}{2\sum_i prob(port \rightarrow when(i))}$$

As an example, consider a port with the following data, and with P(A) = 0.25 and P(B) = 0.50:

Index	When	Rise Energy	Fall Energy
0	!A&B	3.0nJ	4.1nJ
1	A&!B	3.2nJ	5.0nJ
2	!A&!B	7.0nJ	9.0nJ

The calculation of energy for a single transition on this input would be:

$$inputEnergy = \frac{prob(!A\,\&\,B) * (3.0nJ + 4.1nJ) + prob(A\&!B) * (3.2nJ + 5.0nJ) + prob(!A\&!B) * (7.0nJ + 9.0nJ)}{2(prob(!A\,\&\,B) + prob(A\&!B) + prob(!A\&!B)}$$
$$= \frac{0.375(7.1nJ) + 0.125 * (8.2nJ) + 0.375(16nJ)}{2(0.375 + 0.125 + 0.375)} = \frac{9.6875}{1.75} = 5.5357nJ$$

State Dependent Arc-based Internal Power (output pins)

Output internal energy is a weighted sum of values extracted from internal power tables that are associated with timing arcs. These tables are indexed by input transition time and output load capacitance. The values for each arc are weighted by the transition density of the corresponding inputs. The resulting energy value is multiplied by the transition density of the output pin to calculate the power.

When state dependent arc-based (output) internal power tables are present, if two or more tables apply to the same arc, they are weighted by the `when' clauses in the same manner as described in the previous section for "state dependent internal power (input pins)". As an example, we will calculate the output internal power for an AND gate with inputs A and B and output Y. The goal is to calculate the internal power contributed by the output Y. This example does not include any state-dependency.

The first step is to determine the output load capacitance on Y. The output load is the sum of the parasitic net capacitance on the net connected to the Y pin of the instance, plus the pin capacitances of all of the input pins that the net drives. In the example shown, this value is 0.0085pF + 0.03pF = 0.0385pF. Convert this value as required to the units specified in the .lib file for capacitive loads:

```
capacitive_load_unit (1,pf);
```

The next step is to convert the input transition time for each input. The transition time provided is assumed to be extrapolated to 0-100. To convert it, find the bounds in the .lib file:

```
slew_lower_threshold_pct_fall : 10.0;
slew_upper_threshold_pct_fall : 90.0;
slew_lower_threshold_pct_rise : 10.0;
slew_upper_threshold_pct_rise : 90.0;
```

For this case, the time needs to be converted to 10-90. Assuming that in our example that the given transition time for both of the inputs is 0.5ns, the result would be:

$$tt' = \frac{rise - fall}{100} * tt = \frac{90 - 10}{100} * 0.5ns = 0.4ns$$

If there are no slew threshold options set in the .lib file, or they are commented out, the default is 20-80.

The resulting time should be converted, if necessary, to the time units provided in the .lib:

```
time_unit : "1ns";
```

The next step is to associate the timing arcs with the tables in the .lib file. An AND gate has eight arcs, each corresponding to a change in output logic level in response to a change in an input logic level:

$$\begin{array}{l} Y\uparrow - A\uparrow \\ Y\uparrow - A\downarrow \\ Y\uparrow - B\uparrow \\ Y\uparrow - B\downarrow \\ Y\downarrow - A\uparrow \\ Y\downarrow - A\downarrow \\ Y\downarrow - B\uparrow \\ Y\downarrow - B\downarrow \end{array}$$

Normally, these eight arcs are represented by four tables in the .lib file. This is the correspondence.

The energy for a transition has two components, rise (EY,rise) and fall (EY,fall). These two values are averaged, because output Y rises as often as it falls, for the total energy for one transition. Note that in the following equations, the rise and fall contributions are averaged as well for the same reason.

$$E_{Y,rise} = \frac{D(A) * \frac{1}{2}(E(Y\uparrow A\uparrow) + E(Y\uparrow A\downarrow)) + D(B) * \frac{1}{2}(E(Y\uparrow B\uparrow) + E(Y\uparrow B\downarrow))}{D(A) + D(B)}$$

$$E_{Y,fall} = \frac{D(A) * \frac{1}{2}(E(Y\downarrow A\uparrow) + E(Y\downarrow A\downarrow)) + D(B) * \frac{1}{2}(E(Y\downarrow B\uparrow) + E(Y\downarrow B\downarrow))}{D(A) + D(B)}$$

In the above equations, D(A) is the transition density on input A, and E(Y*A*) is the energy value looked up from the corresponding table as described above.

For this example, the energy is:

$$E_{Y,rise} = \frac{10000s^{-1} * \frac{1}{2}(0.0134pJ + 0.0134pJ) + 2000s^{-1} * \frac{1}{2}(0.0135pJ + 0.0135pJ)}{10000s^{-1} + 2000s^{-1}}$$
$$= \frac{161pJs^{-1}}{12000s^{-1}} = 0.0134167\,pJ$$
$$E_{Y,fall} = \frac{10000s^{-1} * \frac{1}{2}(0.0244pJ + 0.0244pJ) + 2000s^{-1} * \frac{1}{2}(0.0267pJ + 0.0267pJ)}{10000s^{-1} + 2000s^{-1}}$$
$$= \frac{297.4pJs^{-1}}{12000s^{-1}} = 0.0247833pJ$$

The power is:

$$P = \frac{1}{2}(E_{Y,rise} + E_{Y,fall})D(Y)$$
$$= 0.5(0.0134167\,pJ + 0.0247833\,pJ)3500s^{-1}$$
$$= 66.85\,pW$$

State Dependent Leakage Power

As the leakage component of power dissipation increases, accurate estimation of it becomes more and more critical. Originally, the leakage component of a cell's power dissipation was modeled in Liberty libraries as a single number. However, nowadays it is more common to associate different leakage power values with different input combination ("state-dependence").

The method to compute leakage power is as follows. Extract the state-dependent leakage data from a cell's Liberty .lib description and compute a weighted sum of the leakage values based on the instance's input probabilities.

The state-dependence is expressed as a set of logical functions describing various input conditions. This set of functions may or may not be complete (e.g. covering all possible input combinations). In addition, a generic leakage power value may also be provided. The power engine covers all of the possible combinations of available data.

State-dependent leakage data appears in a Liberty library in the following form:

```
cell_leakage_power : 14.335 ;
leakage_power(){
when : "!A1 !A2" ;
```

```
value : 9.120 ;
}
leakage_power(){
when : "!A1 A2" ;
value : 16.467 ;
}
leakage_power(){
when : "A1 !A2" ;
value : 12.364 ;
}
leakage_power(){
when : "A1 A2" ;
value : 19.390 ;
}
```

Note that the set of `when' clauses are complete; all possible combinations of the inputs A1 and A2 are accounted for. Also note that there is a generic leakage power value that is not associated with any condition.

We assume the following combinations of input data for our approach:

1. Complete clause set with or without generic leakage value
2. Incomplete clause set with generic leakage value
3. Incomplete clause set without generic leakage value
4. Over-complete clause set with generic leakage value (error condition)
5. Over-complete clause set without generic leakage value (error condition)

If the clause set is complete, we expect the sum of the probabilities of all of the clauses to add up to 1.0. Verification of a complete clause set requires logical analysis of the statements. As extensive library verification is not within the functional requirements of the power engine, we base our assessment of the clause set's completeness by the sum of the probabilities. If this sum is equal to 1.0, the set is complete. If the sum is less than 1.0, we will assume the set is incomplete. If the sum is greater than 1.0 (which would indicate over-coverage), we assume that there is an error in the library data. In order to decide which of the above five conditions we have, we need two pieces of information: whether or not we have a generic leakage value and the sum of the clause probabilities.

We find the probability sum as follows:

$$probTotal = \sum_{i} prob(cell \rightarrow when(i))$$

Then we use the following equations and procedures to detect and calculate the five input conditions:

1. Complete clause set with or without generic leakage value

   This condition is assumed if probTotal is equal to 1.0. For this case we calculate the weighted sum of all available clauses:

$$leakSum = \sum_i cell \rightarrow when(i) \rightarrow value() * prob(cell \rightarrow when(i))$$

$$leakagePower = leakSum$$

2. Incomplete clause set with generic leakage value

   For this case, we use the generic leakage value to "fill in" the missing clauses. We do this by weighting the generic leakage value with 1.0 minus probTotal.

```
leakagePower = leakSum + (cell_when_generic_value () * (1.0 -
probTotal))
```

3. Incomplete clause set without generic leakage value

   For this case we simply scale up the leakSum value to accommodate the missing clauses. We accomplish this by dividing it by probTotal.

$$leakagePower = \frac{leakSum}{probTotal}$$

4. Over-complete clause set with generic leakage value (error condition)

   This case occurs when probTotal is greater than one, and there is a generic leakage value. For this case, we simply revert to the generic value (a warning is also printed to alert the user that there is a possible problem with the library).

$$leakagePower = cell \rightarrow when_generic_value()$$

5. Over-complete clause set without generic leakage value (error condition)

   Without a generic leakage value, we must use the incorrect leakage data as best as we can. We do this by calculating leakSum and scaling it down. It uses the same equation as condition 3. (A warning is also be printed).

Switching Power

Switching power is calculated using the basic equation:

`SwitchingPower` = CAV^2F

Where:

C = Loading net capacitance (SPEF/DSPF or a default load value)
V = Voltage
A = Nodal activity
F = Operating frequency

The product of "A*F" is the transition density (D) calculated during the activity propagation inside the power engine. Since the calculated transition density includes both rising and falling transitions, the equation is modified for a given power rail as:

`SwitchingPower` = $1/2CV^2D$

Where a net is driven by multiple outputs, a good example is a clock mesh driven by parallel clock drivers, the capacitance is split or divided amongst the output drivers.

Vector-based Average Power Calculation

The vector-driven approach uses the VCD or TCF output of a logic simulator to obtain the number of transitions for each net. PM requires gate-level VCD or TCF with good functional coverage for accurate power calculation results.

You can use VCD or TCF information to calculate accurate power consumption figures, if the following conditions apply:

- Gate-level simulation is possible at the full-chip level.
- Gate-level simulation provides sufficient functional coverage for the design.
- The vectors include those that cause the highest power consumption.

The power engine calculates the number of transitions from 0<->1, 0/1<->X and 0/1<->Z. The 0<->1 transition is counted as 1, 0/1 <-> X transitions are counted as 0.5 by default and 0/1<->Z transitions are counted as 0.25 by default. You can use `-x_transition_factor` and `-z_transition_factor` options of `set_power_analysis_mode` command for changing the default value of 0/1<->X or 0/1 <->Z transitions. The power engine also calculates the duty cycle of each net for state dependent internal or leakage power calculation as described previously. The power engine also takes clock definition from VCD or TCF and gives higher

priority to clock definition from VCD or TCF if there is discrepancy between clock frequency in SDC or TWF and VCD or TCF.

Propagation-based Average Power Calculation

The power engine calculates the switching probability, as well as static state probability, of each net in the design. The propagation based approach is vector-independent and provides coverage for all nets in a design.

However, the accuracy depends on good starting values, that is, information about the switching probabilities at the primary inputs in a design. Simple examples are clock and reset or enable inputs. Obtaining an accurate prediction without information about the switching probabilities of these special inputs is difficult, and in most cases an inaccurate prediction causes an over estimation of the power consumption.

Activity Propagation in the power engine

Activity propagation inside the power engine can be divided into following categories.

Activity propagation through combinational cells

Activity propagation through combinational cells is easier. The power engine gets function of combinational cell from .lib and uses the function to propagate activity through combinational cells. The power engine also propagates duty cycle through combinational cells. The only tricky part is when there are combinational loops inside design. In this case the power engine seeds activity at input to break combinational loop. The seeded activity is based upon internal heuristic of power engine and takes into account activity of other neighboring pins.

Activity propagation through sequential Cells

Activity Propagation through sequential cell is based upon activity of input pin, set or reset pin, and scan enable pin. However most of sequential cells are in sequential loops like state machines which make activity propagation through sequential cell based on heuristics by seeding activity at input of sequential cell. Therefore it is recommended to provide activity at outputs of sequential cells. With the power engine you can use either the `set_default_switching_activity` command to specify the average activity on sequential cells or use RTL, VCD, or TCF for seeding activity at sequential cells.

As seen in the example below, the activity at the output of the sequential cell in the loop can not be resolved using propagation. In this case, iterating the loop to determine the activity at the output Q of the sequential cell will result in diminishing activity towards 0. It proves that

using heuristic method to compute activity in sequential loop is an intractable problem for propagation based power calculation. Therefore, in order to get good average power numbers for the design under test, you must always specify the average activity at the output of the sequential cell using the above mentioned command. In addition, you can override this default activity using VCD or TCF.

Activity propagation through macros

The major component of power in macros is internal power. Internal power of macros is highly sensitive to activity on read and write signals. Small change in activity of the read or write signal can cause large change in internal power numbers. Therefore, it is recommended that users specify activity at the read and write signals of macros.

Activity propagation through clock network and clock gates

For accurate propagation through clock network it is important that users specify the TWF file which has clock frequency of generated clocks as well. The activity propagation of clock through clock gating cells depends upon activity of clock enable signal. Since clock enable is a signal net, it will generally have low activity unless specified, which will cause lot of optimism in power calculation. Therefore, it is recommended that users should specify activity at enable of clock gating cells for proper propagation through clock network. You can use the `set_default_switching_activity` command in Voltus for specifying average activity at enable signal of all clock gating cells.

Recommended methodology for activity propagation

The above diagram explains the recommended methodology for activity propagation. The accuracy of results increase as you specify more inputs to the power engine. At the very least users should specify the average sequential activity, which is MUST for accurate activity propagation.

Here are some of examples which depict how well the above methodology works:

Design	Input TCF (from VCD)	Input TCF and Avg seq activity (from VCD)	VCD
Testcase1	75.83	96.90	70.75
Testcase2	148.80	230.0	199
Testcase3	279.46	258.45	195
Testcase4	243.48	243.24	170

The table clearly shows that if you specify just activity on inputs then the result vary from -25% from +45% whereas if you specify average sequential activity of design then results are pessimistic by 25-30%, which is expected because combinational activity propagation is meant to be pessimistic.

Here are another two examples which show how well results correlate after specifying activity at Macros and clock gating cells.

Testcase 5:

- Expected power = ~375mW
- Power after specifying default input activity = 225mW
- Power after specifying default input activity + macro activity = 254mW
- Power after specifying default input activity + macro activity + clock gating activity = 370mW

Testcase 6:

- Expected power = 10W
- Power after specifying default input activity = 5.5W
- Power after specifying clock gating activity + Macro activity = 10.8W

Static Power Analysis Flow

This section describes how to set up the static power analysis mode and then run power analysis.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- SPEF

Procedure

To perform static power analysis, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

```
read_lib -lef $lefs

read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz
```

2. Specify the CPF file for power domain information through the command line or through the GUI.

   GUI: Select *File - Load/Commit CPF*.

   Tcl command:

```
read_power_domain -cpf ../design/super_filter.cpf
```

3. Specify the SPEF file for signal net parasitic information through the command line or through the GUI.

   GUI: Select *File - Read SPEF*.

Tcl command:

```
read_spef -rc_corner  RC_wc_125 -decoupled\
../design/postRouteOpt_RC_wc_125.spef.gz
```

4. Define the power analysis mode through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Set Power Analysis Mode*.

   Tcl command:

```
set_power_analysis_mode \
    -reset
set_power_analysis_mode \
    -analysis_view              AV_wc_on \
    -write_static_currents      true \
    -binary_db_name             staticPower.db \
    -create_binary_db           true \
    -method                     static
```

5. Optional. Define the power of an instance or cell, if you do not specify .lib.

   Tcl command:

```
set_power -reset
set_power -cell pll -pin VDD 0.5
set_power -instance u0 -pin VDD 0.5
```

   You can also specify power-grid views using the `set_power_analysis_mode -power_grid_library` parameter, instead of specifying .lib or defining a power value with `set_power`.

6. Specify the toggle rates on specific instances, nets, and primary inputs, if they are not defined through user attributes, TCF, VCD, or clock constraints.

Tcl command:

```
set_switching_activity -reset
set_switching_activity -input_port rst -activity 0.25 -duty 0.30
set_switching_activity \
   -pin [ get_pins -of_objects [ get_cells RC_CGIC* -hierarchical] \
        -filter "@direction == out" ] \
    -activity 0.1
propagate_activity
```

You can specify the `propagate_activity` command to propagate activities in the design and check for activity annotation before running power calculation. This is an optional step because the `report_power` command will also propagate activities, and then do power calculation based on the propagated activities.

7. You can check whether the activities were applied properly.

   Tcl command:

```
get_activity -port  rst
get_activity -pin [ get_pins -of_objects [ get_cells RC_CGIC* -
hierarchical] \
        -filter "@direction == out" ]
```

8. You can define default activities for the unassigned nets.

   Tcl command:

```
set_default_switching_activity -input_activity 0.3 \
    -period 4.0 -clock_gates_output_ratio   0.5
```

9. Specify the directory where the power results will be stored.

   Tcl command:

```
set_power_output_dir staticPowerResults
```

10. Run power analysis through the command line or through the GUI.

    GUI:

    - Select *Power & Rail* - *Run Power Analysis* to access the *Run Power Analysis* form that can be used to specify cell/instance power (`set_power`), toggle rates (`set_switching_activity` and `set_default_switching_activity`),

power result directory (`set_power_output_dir`), and then run power analysis (`report_power`).

- Select *Power & Rail - Textual Reports - Power Analysis* to access the *Report Power* form that can be used to generate incremental power reports.

Tcl command:

```
report_power -outfile static.rpt
#Generate additional report formats
report_power -instances {u0}    -outfile u0.rpt
report_power -instances {*}     -outfile all.rpt
report_power -instances {ring}  -outfile ring.rpt
```

The Voltus console will display power statistics, as shown in the following snippet:

```
Total Power
-------------------------------------------------------------------------------
Total Internal Power:        2.30917507    64.6507%
Total Switching Power:       1.03379946    28.9436%
Total Leakage Power:         0.22879451     6.4056%
Total Power:                 3.57176907
-------------------------------------------------------------------------------
Ended Static Power Report Generation: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=750.07MB/1877.45MB/750.07MB)

Begin Creating Binary Database
Ended Creating Binary Database: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=750.34MB/1877.45MB/750.34MB)
```

Output

The power output directory will contain the following information:

File Name	Description
`<binary_database>.db`	The name of the binary database is specified using the `set_power_analysis_mode -binary_db_name` parameter. The database can be loaded in the Voltus GUI for viewing and analyzing different types of power plots, and GUI interactive debugging. It cross references the instance names and locations with the power results.
`staticPower.db.cnstr.tcl`	This TCL file stores the commands used to set up power analysis.
`<power_report>.rpt`	The name of the power report is specified using the `report_power -outfile` parameter. The power report summarizes power consumption, by groups (sequential cells, macros, clocks), power net, and clock domain.
`static_<net_name>.ptiavg`	This is the current file for each net. The `set_power_analysis_mode -write_static_currents` parameter will generate the current files that are required for rail analysis.

You can use the `set_dc_sources` command to specify the operating voltages of the power/ground nets. This command is used to specify a global voltage or voltages for specific power nets in the design.

Note: If you do not have a SPEF file, you can use wire load models for load specification which can then be used for static power analysis. The following commands are used to specify wire load models:

- `auto_wire_load_selection`
- `enable_wire_load_model_support`
- `report_wire_load`
- `reset_wire_load_mode`
- `reset_wire_load_model`
- `reset_wire_load_selection_group`
- `set_wire_load_mode`

- `set_wire_load_model`
- `set_wire_load_selection_group`

For more information about these commands, see the *Tempus Text Command Reference*.

Tip

If you have the following commands

```
set_power_output_dir static_power_max
report_power -outfile design.rpt
```

then `design.rpt` will be dumped to `static_power_max` directory. If you explicitly specify a directory in the `-outfile` option such as below,

```
report_power -outfile my_dir/design.rpt
```

then `design.rpt` is saved to `my_dir` directory and the path setting by `set_power_output_dir` will be ignored.

M**MMC Mode Default Views**

If set to the MMMC mode, it is recommended to specify the view by one of the below commands,

`set_power_analysis_mode -analysis_view` *`viewname`*

or

`report_power -view` *`viewname`*

If you do not, then the default power view is the first setup or hold view depending on the `checkType` of the `set_analysis_mode` command.

Power analysis runs on only one view at a time, and therefore, the argument to the `-view` parameter must specify only one view. If you give multiple view names to the `-view` parameter of `report_power` or you do not activate the view using the `set_analysis_mode` command, a warning message will be displayed.

For multiple views, you need to have multiple runs, one for each view. Also, you need to make the view active in order to do power analysis.

If specified as shown below,

```
set_analysis_view -setup {view1 view2} -hold {view3 view4}
read_spef -rc_corner best design.spef
set_analysis_mode -checkType setup
report_power
```

then the power view in this case is the first setup view which is `view1`.The default for `-checkType` is `setup`. See `set_analysis_mode` in the *Tempus Text Command Reference* for details of this command.

If specified as shown in the next example,

```
set_analysis_view -setup {view1 view2} -hold {view3 view4}
read_spef -rc_corner best design.spef
set_analysis_mode -checkType hold
report_power
```

then the power view in this case is the first hold view which is `view3`.

Note: When CPF or MMMC views are loaded, SPEFs must be read after the `set_analysis_view` command as shown in the above examples.

If multiple command types specify the view, the priority is given as follows (from highest to lowest):
`report_power`, `set_power_analysis_mode`, `set_analysis_mode`

For additional MMMC details, see *Performing Multi-Mode Multi-Corner Timing Analysis* in the *Tempus User Guide*.

Power Estimation of a Pre-CTS Netlist

You can run the `set_virtual_clock_network_parameters` command to analyze and provide fast power estimation of the clock tree power of a pre-CTS netlist. When this command is specified, Voltus constructs a virtual clock tree and estimates power of the clock network. You can estimate power selectively for specific clocks in the design.

The example TCL command for power estimation of a pre-CTS netlist is as follows:

```
read_verilog design.v
read_lib "$libraryDir/.../.../library1.lib $libraryDir/.../.../library1.lib"
set_top_module design

read_sdc $testcaseRoot/new.sdc
set_power_analysis_mode -create_binary_db true -honor_negative_energy true -
ignore_control_signals false -write_static_currents true -power_grid_library "
$testcaseRoot/all.cl "
set_default_switching_activity -input_activity 0.2 -period 1.87512 -duty 0.5
set_virtual_clock_network_parameters -cell {BUFX12} -library typical -max_fanout 2
report_power -rail_analysis_format VS -outfile ./instance.rpt
report_power -outfile power.rpt
report_power -clock_network all -outfile clock.rpt
exit
```

Power Map File for Thermal Analysis

Voltus supports generation of a tile-based power map file to perform accurate thermal analysis. This power map file is later used by PowerDC, the thermal analysis tool for packages and boards.

To generate the power mapping file, you need to specify the following parameters of the `report_power` command:

- `-thermal_power_map_file` *`file_name`* - generates the power map file with the specified name in the output directory.
- `-thermal_power_map_tile` *`{Xint Yint}`* - defines the tile number in the X and Y direction. By default `10*10` will be used.
- `-thermal_leakage_temp` *`{temp1 temp2 temp3 …}`* - defines the temperature to calculate leakage for the power map file in a simple format.
- `-thermal_power_map_format {simple | stack}` - generates a tile-based power map file in the `simple` or `stack` format to perform accurate thermal analysis. The `simple` format generates a 2D power map file and the `stack` format generates a 3D power map file. You must use the `stack` format to support multiple dies in a package for thermal analysis. The default format is `simple`.
- `-thermal_conductivity_inputs` *`file_name`* - specifies a file containing the thermal conductivity value for layers. You can use this parameter to include the thermal conductivity value of the following layers in the power map file:
  - Silicon
  - Metal
  - Dielectric
  - Micro Bump (optional for 3DIC design)
  - Fill material in micro bump layer (optional for 3DIC design)

The format of the thermal conductivity input file is:

```
Unit <unit_definition> (optional, by default unit is mW/(um * C))
Metal  <thermal_conductivity_value>
Silicon <thermal_conductivity_value>
Dielectric <thermal_conductivity_value>
Bump  <thermal_conductivity_value> (optional)
Fill_Material  <thermal_conductivity_value> (optional)
```

Simple Format

The format of the power map file is divided into two parts, `TRANSIENT POWER MAP` and `LEAKAGE POWER MAP`, as shown below:

Power Unit : mW **Time Unit : ms**			
TRANSIENT POWER MAP			
nx	ny	...	
P(0,ny-1)	P(1,ny-1)	...	P(nx-1,ny-1)
P(0,ny-2)	P(1,ny-2)	...	P(nx-1,ny-2)
:			
P(0,0)	P(1,0)		P(nx-1,0)
END TRANSIENT POWER MAP			
LEAKAGE POWER MAP			
T0	nx	ny	
	P0(0,ny-1)	P0(1,ny-1)	... P0(nx-1,ny-1)
	P0(0,ny-2)	P0(1,ny-2)	... P0(nx-1,ny-2)
:			
	P0(0,0)	P0(1,0)	... P0(nx-1,0)
T1	nx	ny	
	P1(0,ny-1)	P1(1,ny-1)	... P1(nx-1,ny-1)
	P1(0,ny-2)	P1(1,ny-2)	... P1(nx-1,ny-2)
:			
	P1(0,0)	P1(1,0)	P1(nx-1,0)
T2	nx	ny	
:			

where,

- `nx` and `ny` is the tile number in the power map file

- `P(0,0)` …. is the power value assigned to a tile at a specific coordinate

The following diagram illustrates the location of each tile:

The power value of each tile is the total power of instances located in the tile. If one instance overlaps with more than one tile, its power is distributed to each of the tile according to the area overlaps. The package thermal analysis tool uses tile names in the power map file to get the coordinates of each tile.

For the **tri-lib flow**, Voltus writes out the `LEAKAGE POWER MAP` part to the power map file to include the temperature dependent leakage power of each tile. The temperature value in the `LEAKAGE POWER MAP` part come from the values defined using the `-thermal_leakage_temp` parameter. If the `-thermal_leakage_temp` parameter is not used; the temperature values defined in the timing library is used for the `LEAKAGE POWER MAP` part. If the `LEAKAGE POWER MAP` part is included in the power map file, the leakage power is not included in the `TRANSIENT POWER MAP` part. The power value in the `TRANSIENT POWER MAP` part is the sum of the internal power and switching power calculated using the temperature defined by the `set_power_calc_temperature` command.

For the **non tri-lib flow**, the `LEAKAGE POWER MAP` is not included, and only the `TRANSIENT POWER MAP` part is included in the power map file. The power value in the `TRANSIENT POWER MAP` part is the sum of the internal power, switching power, and leakage power.

Stack Format

The stack format of the power map file is divided into two parts, `DIE STACK POWER MAP` and `LEAKAGE POWER MAP`, as shown below:

Length Unit : um **Power Unit : mW** **Thermal_conductivity Unit : ((mW/(um * C)))** **Time Unit : ms**					
DIE_STACK_POWER_MAP					
DIE_NAME	die_inst_name				
DIE_AREA	xmin(um)	ymin(um)	xmax(um)	ymax(um)	nx(int) ny(int)
NUMBER_OF_LAYERS	nLayers(int)				
LAYER	layer_name0(string) layer_thickness0(um) metal_thermal_conductivity(mW/(um * C)) silicon_thermal_conductivity(mW/(um * C)) XCOOR x0(0) x0(1) x0(2)... ...x0(nx-1) YCOOR y0(0) y0(1) y0(2)... ...y0(ny-1)				
BEGIN_POWER_SEQUENCE					
POWER	<step1_start>	<step1_end>			
P0(0,ny-1)	P0(1,ny-1)	...	P0(nx-1,ny-1)		
P0(0,ny-2)	P0(1,ny-2)	...	P0(nx-1,ny-2)		
:					
P0(0,0)	P0(1,0)		P0(nx-1,0)		
END_POWER					
POWER	<step2_start>	<step2_end>			
...					
END_POWER					
END_POWER_SEQUENCE					
METAL_DENSITY					
K0(0,ny-1)	K0(1,ny-1)		K0(nx-1,ny-1)		
K0(0,ny-2)	K0(1,ny-2)		K0(nx-1,ny-2)		
K0(0,0)	K0(1,0)		K0(nx-1,0)		

END_METAL_DENSITY			
LEAKAGE_POWER_MAP			
T0	nx	ny	
	P0(0,ny-1)	P0(1,ny-1)	... P0(nx-1,ny-1)
	P0(0,ny-2)	P0(1,ny-2)	... P0(nx-1,ny-2)
:			
	P0(0,0)	P0(1,0)	... P0(nx-1,0)
T1	nx	ny	
	P1(0,ny-1)	P1(1,ny-1)	... P1(nx-1,ny-1)
	P1(0,ny-2)	P1(1,ny-2)	... P1(nx-1,ny-2)
:			
	P1(0,0)	P1(1,0)	P1(nx-1,0)
T2	nx	ny	
:			
END_LEAKAGE_POWER_MAP			
END_LAYER			
LAYER	layer_name1	layer_thickness1	
:			
END_LAYER			
:			
END_DIE_STACK_POWER_MAP			

where,

- `DIE_NAME` is the name is the die instance
- `DIE_AREA` specifies the coordinates of the die area
- `NUMBER_OF_LAYERS` is the number of layers in each die instance
- `LAYER` specifies the LEF layer name, layer thickness, and coordinates of each layer
- `BEGIN_POWER_SEQUENCE` section reports the transient power for all tiles at each time step.
- `nx` and `ny` is the tile number in the power map file. This is specified using the `-thermal_power_map_tile {Xint Yint}` parameter to divide the power map in tiles.

- `P(0,0)` …. is the power value assigned to a tile at a specific coordinate
- `P(x,y)` is total power value of all the instances falling in this tile.
- the thermal conductivity value for layers are included in the power map file as:
  - `metal_thermal_conductivity(mW/(um * C))`
  - `silicon_thermal_conductivity(mW/(um * C))`
  - `bump_thermal_conductivity(mW/(um * C))`
  - `fill_material_thermal_conductivity(mW/(um * C))`
  - `dielectric_thermal_conductivity(mW/(um * C)`

The following diagram illustrates that the die is always facing up, and also shows the location of each tile:

- If the `stack` format is chosen, and no power grid library is provided using the `set_power_analysis_mode -power_grid_library` parameter, only one layer called `SUBSTRATE` will be generated in the power map file with distributed power information. As there is no thickness information, `NULL` will be used for thickness of `SUBSTRATE` layer for you to manually edit it in PowerDC.
- If the `stack` format is chosen, and power grid library is provided using the `set_power_analysis_mode -power_grid_library` parameter, the thickness of each layer is obtained from the QRC technology file stored in the techonly power grid library. For most of the die with no definition for the substrate thickness, `NULL` will be the thickness of the `SUBSTRATE` layer for you to manually edit it in PowerDC. If there is

thickness definition in the QRC technology file, write the value defined in the QRC technology file to the power map file.

TCL Script for Generating a Simple Power Map File

```
set_power_analysis_mode -method static
set_power_calc_temperature 25
report_power -rail_analysis_format VS -outfile powermeter.txt -
thermal_power_map_file powermap.txt -thermal_power_map_tile { 4 6 } -
thermal_power_map_format simple
```

TCL Script for Generating a Stack Power Map File

```
set_power_analysis_mode -method vector_profile -vector_profile_mode
{power_density} ...
read_activity_file -format SHM  /shm/ADDER.trn -scope adder_tb_e/u1 -start 0ns -
end 60ns
set_power_calc_temperature 25
report_power -outfile powermeter.txt -thermal_power_map_tile { 2 2} -
thermal_leakage_temp { 25 35 50} -thermal_power_map_format stack -
power_density_tiles_row_col { 2 2} -time_based_report
```

The following is a sample simple power map file for a non tri-lib flow:

```
Power Unit : mW
Time Unit : ms
TRANSIENT POWER MAP
5    5
0.0250229  0.00773986   0.0079822    0.00643712   0.0233794
0.0123914  0.0020813    0.00297774   0.00175598   0.071479
0.0474979  0.00203936   2.91455e-05  0.00093574   0.0717009
0.0238089  0.000668492  2.91455e-05  0.000654917  0.00218322
0.0587654  0.0645726    0.010563     0.0288181    0.0250317
END TRANSIENT POWER MAP

LEAKAGE POWER MAP

25 2 2

1.00556e-06 2.07938e-06

4.88284e-07 3.31147e-07
```

Voltus Thermal Model Generation for Thermal Analysis

Voltus supports generation of a thermal model that divides the design into equal sized tiles, and includes data on the power and thermal material properties of each tile. The flow uses the `specify_def` command instead of the `read_def` command, therefore allows you to handle very large designs without loading design data into memory.

The flow is illustrated in the following diagram:

The flow has the following main steps:

1. Extracts metal density information using the `set_metal_density_options` and `extract_metal_density` commands. This step generates the Voltus Thermal Model (VTM) metal density file for Celsius Thermal Solver (thermal analysis tool) without performing power calculation.

2. Generates the power database using the `set_power_analysis_mode -save_bbox` parameter and `report_power` command. This command saves the power data with the instance boundary box information.

3. Combines the metal density file and power database using the `create_thermal_model` command for VTM generation.

   Note: Optionally, you can use the `merge_vtm` command to import the top-level and block-level VTMs to generate a merged uniform single VTM.

TCL Script for VTM Generation

The TCL script for running the VTM flow are as follows:

```
set_top_module …
specify_def data/design.def
.....(related power cal commands)
set_power_output_dir powerdb_test
set_power_analysis_mode
-method  static \
-save_bbox true \
-create_binary_db true \
-enable_xp    true \
-static_netlist  def \
-power_grid_library   ../pgv/techonly.cl \
-extraction_tech_file ../data/Qrctech/qrcTechFile

report_power
set_metal_density_options \
-power_grid_library       { ../pgv_qrc_test/techonly.cl } \
-tech_file ../data/Qrctech/qrcTechFile \
-vtm_path ./MDdb_test \
-thermal_conductivity_file ../misc/conductivity_file.txt \
-vtm_densitytable_tile {10 10}
extract_metal_density -mode true

create_thermal_model \
-power_db $powerdbinput \
-metal_density_file $metalfile \
-vtm_output_dir ./vtm_output \
-vtm_file  stack_vtm.txt \
-vtm_tile {5 5} \
-vtm_format stack \
-vtm_SI_unit false  \
-vtm_header_include_file ../misc/user_header.txt
```

Viewing Static Power Analysis Results

The calculated static power can be debugged in the context of physical layout using interactive power analysis plots. Voltus allows you to visualize power consumption in the design using the following methods:

- Power Results Viewer
- Power Debugger

Required Input Files

- Design data (LEF, netlist, DEF)
- Power database generated during static power calculation

Procedure (Power Results Viewer)

To visualize power results using power results viewer, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

```
read_lib -lef $lefs

read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz
```

2. Load the power data through the command line or through the GUI.

   GUI:

   a. Select *Power & Rail - Power & Rail Plots*.

The Power & Rail Plots form appears. This form is the main window for displaying the power analysis results.

b. Click the *DB Setup* button.

The Power & Rail Setup form appears.

c. You can enter the power database or click *Browse* next to the *Power Database* field, and select a location from the Binary DB File browser window.

d. Click *OK*.

Tcl command:

```
read_power_rail_results -power_db staticPowerResults/staticPower.db
```

3. In the Power & Rail Plots form, select the *Power* radio button and select a power analysis type from the drop-down list.

The selected plot will be overlaid on the design.

Tcl command:

```
#Displays the specified plot
set_power_rail_display -plot ip -filter_max 0.0038 \
                       -filter_min 1.00001e-05

#Displays the specified plot and generates a report
(power_rail_<plot_type>.rpt) in the current working directory
report_power_rail_results -plot  freq

#Clear the display
set_power_rail_display -plot none
```

4. Click the Range Max/Min icon to specify the minimum and maximum value to include in the report/display. Use the scissor icon to further filter out the data within the range specified. “Filter Max” will hide any data point greater than the filtered value, and vice-

versa for “Filter Min”.

- You can select the *Auto Apply for Color Scale* checkbox to automatically redraw the layout after modifying the range values. If you do not select this checkbox, you can click the *Apply* button to manually redraw the layout.
- You can click the *Basic*/*Advanced* toggle button to alternate between the color gradient (*Basic*) and color range (*Advanced*) mode. By default, the *Basic* button and the color gradient based filter are displayed. When you click the *Basic* button, the button text changes to *Advanced*, and the form shows the color range based filter. The color gradient based filter allows you to modify the minimum and maximum values, and the color range based filter allows you to modify 8 equal filters between the minimum and maximum values.

5. Select the *Result Browser* option to display the Results Browser form. This form allows you to check the power distribution of each instance and identify areas of the design that have high current. The Result Browser form has the following two tabs:

- *Summary* - provides summary information about the plot displayed. The data points are divided into buckets based on the range specified. The summary information provides the statistic of each bucket, and displays the power analysis results in the histogram form.

Range	Total Count	Displayed / Scissored	Percentage
4.57e-41 ~ -2.11e-29 mW	0	0 / 0	0.00%
-2.11e-29 ~ 4.57e-41 mW	0	0 / 0	0.00%
> 100.00 mW	0	0 / 0	0.00%
100.00 ~ 2.15 mW	0	0 / 0	0.00%
2.15 ~ 46.42m mW	0	0 / 0	0.00%
46.42m ~ 1.00m mW	0	0 / 0	0.00%
1.00m ~ 21.54u mW	2052	2052 / 0	76.77%
21.54u ~ 464.16n mW	621	621 / 0	23.23%
464.16n ~ 10.00n mW	0	0 / 0	0.00%
< 10.00n mW	0	0 / 0	0.00%

- *Detail* - provides details of the individual data points of the plots displayed. By default, the first 1000 data points are displayed. It lists the instances from the highest total instance power to the lowest instance power.

ip_l (mW)	Cell	Instance
0.000366986	CLKBUFX20	column/clk__FENCE_MSV_Fence_I_N0__L2_I0
0.000366986	CLKBUFX20	column/clk__FENCE_MSV_Fence_I_I0
0.000366986	CLKBUFX20	column/clk__FENCE_MSV_Fence_I_N0__L2_I2
0.000366986	CLKBUFX20	AO1/clk__Fence_I1
0.000366986	CLKBUFX20	AO2/clk__Fence_N0__L1_I0
0.000366986	CLKBUFX20	AO2/clk__Fence_N0__L1_I1
0.000366986	CLKBUFX20	AO1/rc_gclk__Fence_I1
0.000366986	CLKBUFX20	AO2/rc_gclk_135__Fence_I0
0.000366986	CLKBUFX20	external/clk__FENCE_MSV_Fence_I_I0
0.000366986	CLKBUFX20	AO2/rc_gclk__Fence_I0
0.000366986	CLKBUFX20	ref_clk__L1_I0
0.000366986	CLKBUFX20	external/rc_gclk__FENCE_MSV_Fence_I_I0
0.000366986	CLKBUFX20	ring/clk__FENCE_MSV_Fence_I_I0
0.000366986	CLKBUFX20	AO1/rc_gclk_135__Fence_I1
0.000366986	CLKBUFX20	column/clk__FENCE_MSV_Fence_I_N0__L2_I1
0.000366986	CLKBUFX20	AO2/clk__Fence_I0
0.000366986	CLKBUFX20	AO2/clk__Fence_N0__L1_I2
0.000366986	CLKBUFX20	test_clk__L1_I0
0.000366986	CLKBUFX20	external/rc_gclk_135__FENCE_MSV_Fence_I_I0
0.000366986	CLKBUFX20	AO2/clk__Fence_N0__L1_I3

6. Select the *Auto Select* check box to automatically zoom in to the selected instance, as shown below:

7. Click *Close* to close the Result Browser form.
8. Experiment with the remaining power plots, as listed in the following table:

Plot Name	Description
Instance Total Power - ip	Total power plots show power distribution of all the instances in the design. It can be used to debug regions of high IRdrop in the design.
Instance Internal Power - ip_i	Internal power is a component of total power and displayed for all instances in the design. It can be used to debug instances that consume unexpectedly a large total power. The internal power is derived from the.lib file.
Instance Switching Power - ip_s	Switching power is a component of total power and displayed for all instances in the design. The switching power plot can be used in conjunction with transition density and loading capacitance plots to understand the switching profile for the design (i.e. high activity and loading capacitance for the instance translates to high switching power).

Plot Name	Description
Instance Leakage Power - ip_l	Leakage power is a component of total power and displayed for all instances in the design. The leakage power plot can be used to debug high leakage power instances.
Instance Clock Total Power - ip_clk	Displays power of clock instances in the design.
Instance Total Power Density - ipd	Displays instance-based total power density to determine total power consumed by area.
Instance Internal Power Density - ipd_i	Displays instance-based internal power density to determine internal power consumed by area.
Instance Switching Power Density - ipd_s	Displays instance-based switching power density to determine switching power consumed by area.
Instance Leakage Power Density - ipd_l	Displays instance-based leakage power density to determine leakage power consumed by area.
Frequency Domain - freq	The frequency domain plot displays the operating frequency of all instances in the design. The instances operating with multiple clock domains are displayed using the fastest clock frequency for the instance. The power is directly proportional to the frequency, so this plot can be used to debug instances with high power consumption.
Instance Slack - slack	Slack for the instance is derived by Common Timing Engine or external Timing Window File. This plot is primarily used to identify time critical instances and can be useful when performing timing aware decap optimization and IRdrop aware timing analysis.
Transition Density - td	Transition density is the product of frequency and activity. It is plotted for all instances in the design. This plot can be used to debug instances with high power consumption and regions of high activity. Transition density is plotted for the off domains in power-gated design. However, switching power plot correctly displays that no power is calculated for the off domains.
Tile Power Density - tpd	Displays power density map with user-specified grid size.
Instance Activity - activity	Displays the worst activity for the output and inout pins of every instance.

Plot Name	Description
Instance Toggle Rate - ip_tr	Displays toggle rates for all instances. The toggle rate for an instance is calculated as the average of toggle rates of all outputs of the instance. This plot allows you to determine the design area that toggles more than the others.

Procedure (Power Debugger)

To visualize power results using power debugger, complete the following steps:

1. Load the design data through the command line or through the GUI.
2. Load the power data through the command line or through the GUI.
3. Select *Power & Rail* - *Histograms* - *Power Analysis*.

The Power Debug form appears that describes how power is distributed between the levels of hierarchy:

4. Double-click on any part of the pie chart to descend the hierarchy. Click the *Up Level* button to go up one level.

   The power pie chart provides an easy way to access any part of the design hierarchy and examine the power results. The *Power Histogram* section of the form displays how power is distributed between the internal, leakage, and switching power for the current level of hierarchy.

5. Click the *Histograms* button to bring up the various histograms.

   The Power Details form appears:

6. You can select the various tabs that help visualize how the power is distributed.

Tab Name	Description
Cell Type	Shows how power is distributed among sequential, standard cells, IOs, and macros.
Clock Network	Shows power distribution by clock domain
Power Domain	Shows power distribution by power domain
Power Rail	Shows power distribution by power net
Net	Shows power distribution by nets with highest power
Net Toggle	Shows power distribution by nets with highest toggle rate

Tab Name	Description
Net Probability	Shows power distribution by nets with highest toggle probability

You can search for net(s) in the *Net*, *Net Toggle*, and *Net Probability* tabs using wildcard entries (*) and filter any net(s) to display in the histogram when debugging static power. This displays 50 nets at a time, starting with the first 50 nets.

The net that you select in the histogram gets highlighted in the table, and similarly, the net that you select in the table gets highlighted in the histogram.

Information about the selected net is displayed across all the three tabs - *By Net*, *Net Toggle*, and *Net Probability*.

Interactive Queries of Power Data

The power data can be queried interactively in GUI by selecting an instance and pressing "q" on keyboard. The *Attribute Viewer* displays total, internal, switching and leakage power for the instance. In addition, it lists frequency domain, transition density and associated power domains of the instance.

You can also query the power data using the `get_metric` Tcl command that allows to query predefined metrics directly after the `report_power` command is run, without any need for log parsing. The `report_power` command generates the following metrics:

Metric	Description	Example
`power.total`	Returns total power consumption	`power.total 11.8963` `Total power consumption is 11.8963 mW.`
`power.total.leakage`		

	Returns total leakage power	`power.total.leakage 0.0411615` `Total leakage power is .0411615 mW.`
`power.total.internal`		
	Returns total internal power consumption	`power.total.internal 11.4083` `Total internal power consumption is 11.4083 mW.`
`power.total.switching`		
	Returns total switching power	`power.total.switching 0.446901` `Total switching power is .446901 mW.`
`power.total.clock`		
	Returns total power consumed by clocks	`power.total.clock 0.0272253` `Total power consumed by clocks is .0272253 mW.`

The following set of commands returns metrics for the `report_power` command:

```
report_power
get_metric power.*
```

The final output is as follows:

```
12.0688 power.total
 0.042129 power.total.leakage
 11.4966 power.total.internal
 0.530001 power.total.switching
 0.0489306 power.total.clock
```

Debugging Instance Power

The calculated instance power can be debugged using the `report_instance_power` command. This command generates a text based report for the specified instance. The command enables you to determine how the different components of static power are calculated. You can use the report to debug after performing power analysis (`report_power`).

This command is available through both the static and dynamic power engines. This command is supported in the dynamic power engine by including this command in the power include file through the command `set_power_include_file`.

Note: For dynamic power engine, the `report_instance_power` command does not generate text reports incrementally. If the dynamic run has been performed and you need the debug information for a static power number, you should re-run power analysis.

The format of the report is given below:

#This section gives a breakup of the total Internal, Switching, and Leakage power of the instance:

```
Instance: i_24
Cell: XNOR2X1
Liberty file: typical.lib
Internal power: 0.000178732mW
Switching power: 7.72426e-05mW
Leakage power: 3.162207e-05mW;
Total power: 0.000256006mW;
```

#This section shows the calculation of switching power of each net associated with the instance.

```
Net  Pin  Dir  Voltage  Duty  Density       Cap      RiseSlew  FallSlew   Power
n_20  Y  Output  1.2   0.4961 3.90625e+07 0.0027464  0.1378    0.0841 7.72426e-05
n_25  B  Input   1.2   0.5313 3.125e+07   0.00595432 0.0916    0.0789      0
n_26  A  Input   1.2   0.4375 2.5e+07     0.00501434 0.236     0.0775      0
```

#It includes:

Column Name	Description
`Net`	net name
`Pin`	pin name
`Direction`	direction of the pin
`Voltage (V)`	voltage of the power rail
`Duty`	duty cycle of the pin
`Density`	transition density
`Cap (pf)`	total cap (net cap + pin cap)
`rise slew (ns)`	rise slew
`fall slew (ns)`	fall slew from the .lib
`power (mW)`	power for the output pin (calculated using $1/2CV^2*TD$)

#This section shows the calculation of `Leakage power` for state dependent leakage. For different states, the power value is multiplied with the duty cycle. Finally, all the state dependent power values are added to get the `Total Leakage power` reported in the first section.

```
Leakage power
```

```
When                  Duty       Power
((A) & (B))           0.23      0.23*4.7799e-08
((A) & (!(B)))        0.30      0.30*2.88e-08
((!(A)) & (B))        0.21      0.21*2.9958e-08
((!(A)) & (!(B)))     0.26      0.26*2.1912e-08
```

#This section shows how the internal power is calculated for each arc.

```
Internal power
From   To   when    activity/ns  duty      energy      power(mW)
A ^ -> Y ^ : B :    0.0390625    0.5    3.84665e-15   0.00015026
A v -> Y v : B :    0.0390625    0.5    5.60604e-15   0.000218986
A v -> Y ^ : !(B) : 0.0390625    0.5    4.16763e-15   0.000162798
...
...
```

For example, the first row signifies to calculate power for 1st arc when `pin A` is rising and `Y` is rising and `B` is high. Activity from either the activity file or calculated by power engine is `0.0390625/1e-09`. Energy number from the .lib is `3.84665e-15`. Power for the arc is calculated as activity * energy `0.0390625/1e-09*3.84665e-15 = 0.00015026mw`. Similarly, internal power is calculated for each arc.

The total internal power is calculated as shown below:

step 1) calculate power: A -> Y

step 2) calculate power: B -> Y

then, output power at Y is:

```
Power(Y) = [ Power(A -> Y) * TD(A) + Power(B -> Y) * TD(B) ] / [TD(A)+TD(B)] (#1)
Details: (in mW) Note: TD(A) =2.5*e+07; TD(B) = 3.125*e+07;

1) Power (A -> Y)
a) when = B
Power (A -> Y) = [ power (A->Y ^) + power (A -> Y v) ] /2 = [avg (rise + fall)]
               ~ [ 15.030*e-05 + 21.9*e-05 ] / 2 = 18.46*e-05

b) when = !B
Power (A -> Y) = [ power (A->Y ^) + power (A -> Y v) ] /2 = [avg (rise + fall)]
               ~ [ 16.28*e-05 + 4.01*e-05 ] / 2 = 10.145*e-05

c) final: Power (A -> Y) = Pow (A->Y) * Duty(B) + Pow (A->Y) * Duty(!B)
                         = [18.46*(0.513) + 10.145*(.487)] * e-05
                         = 14.41 * e-05

2) Power (B->Y)
a) when : A
Pow(B->Y) = 20.6 * e-05 = [avg (rise + fall)]

b) when : !A
Power(B->Y) = 20.457 * e-05 =[ avg (rise + fall)]

c) final Pow(B->Y) = Pow(B->Y) * duty(A) + Power(B->Y)*duty(!A)
                  =[ 20.6 * .4375 + 20.457 * .5625] * e-05
                   = 20.5125 * e -05
```

See (#1)

```
Power (Y) = [ (14.41 * 2.5 * e+07) + ( 20.5125 * 3.125 * e+07) ] * e-05 / [[(2.5 +
3.125)]*e+07]
          = ( 9.0125 + 11.5 ) * e-05 / ( 5.625)
          = 17.8 * e-05 mW
```

Saving and Restoring Power Database

You can save power databases for incremental power report generation (including TCF output), view power histograms and graphical power plots, without having to re-run power analysis during both the static and dynamic runs. This is applicable for flat or hierarchical designs.

TCL Script for Generating Power Database

```
read_lib -lef <LEF>
read_lib <timing libraries>
read_verilog <verilog netlist>
read_sdc <timing constraints>
read_def <DEFs>
```

Use `specify_def` for better performance

```
read_spef <RC parasitic files>
```

#You can choose the method to be dynamic vector-based or static

#The power database will not be generated if `set_power_analysis_mode -disable_static` is set to `true` when running either of the dynamic power analyses

```
set_power_analysis_mode -method dynamic_vectorless -create_binary_db true -
binary_db_name Dynamic.db
```

#Power database name is optional. The default name is `power.db`

```
set_dynamic_power_simulation -period 8ns -resolution 50ps
set_default_switching_activity -input_activity 0.2 -seq_activity 0.1 -period 10.0
report_power
```

TCL Script for Restoring Power Database

```
read_design -physical_data new.dat top
restore_power_database -in_file Dynamic.db
```

You will be able to view the power histograms once the power database has been restored

Incremental report generation can be done without re-doing power analysis. The power db can be restored in the same session also. It is important that you should not change any power mode settings because that will re-trigger power analysis.

```
report_power -outfile restore.rep
report_power -clock_network { all } -outfile Clock.rep
report_power -cell_type { all } -outfile Cell.rep
write_tcf TOP.tcf
```

Generating Standalone Power Reports

Voltus supports incremental and interactive standalone report generation from an existing binary power database stored in the power output directory. This feature allows you to quickly and periodically generate power reports for specific cells/instances from a previous power analysis run.

The use model for standalone power report generation is:

`report_power -power_db_directory <database_dir> -outfile <report_name>`

The following parameters of the `set_power_analysis_mode` command are required to perform standalone report generation in the Extensively Parallel (XP) mode:

- `-enable_xp {true | false}`
- `-extraction_tech_file filename`

The power database is located in the power output directory (for example, `dynamic_power/power.db`) for single-machine multi-threading (local) mode. In case of the distributed and XP modes, the power database is located in the distributed host/partition sub-directory within the power output directory (for example, `dynamic_power/dist_X/power.db or dynamic_power/part_X/power.db`).

Note: The power-grid library is a mandatory requirement for standalone XP power report generation flow.

TCL Script for Generating Standalone Power Reports in the XP Mode

```
set_power_analysis_mode -method dynamic_vectorless \
-enable_state_propagation true \
-power_grid_library {  /pgv_dir/tech_pgv/techonly.cl
/pgv_dir/stdcell_pgv/stdcells.cl  /pgv_dir/macro_pgv/macros_pll.cl  } \
-extraction_tech_file /qrcTechFile -enable_xp true

...
report_power
```

Resistance Analysis

The Effective Resistance Analysis feature allows you to perform effective resistance calculation for all instances in a design in both the net-based and domain-based analysis modes. When you perform domain-based resistance analysis, Voltus gives the sum effective resistance values (Rvdd + Rvss) for all nets in the domain.

You can use the `analyze_resistance` command to perform effective resistance analysis. Use the `analyze_resistance -net` parameter for net-based analysis, and `analyze_resistance -domain` parameter for domain-based analysis.

Important

The effective resistance analysis flow requires a VTS-AA (Voltus IC Power Integrity Solution Advanced Analysis GXL Option) license.

The following resistance analysis methods enable you to determine the type of analysis to be performed:

- **Full-Chip Analysis** - calculates the effective resistance between the specified nodes or instances to all voltage sources/bumps in a design. This type of analysis can be done using one of the followings inputs:
  - specifying a list of instances (`-instance_list`/`-instance_list_file`) - Reports effective resistance between the specified user-defined instances and all voltage sources in the design. By default, the tool will use the instance pin associated with the net being analyzed. You can also specify a pin when two different pins of an instance are connected to the same net. When an instance has multiple pins, the effective resistance at all pins are computed.
  - specifying a master cell name (`-cell`) - Reports effective resistance between all instances of the specified master cell name and all voltage sources in the design. This method allows you to perform effective resistance calculation for all instances of a particular cell. You can specify multiple cells for performing effective resistance analysis of multiple cells simultaneously.
  - specifying a list of nodes (`-node_list`/`-node_list_file`) - Reports effective resistance between the specified user-defined nodes and all voltage sources in the design. You can add a list of nodes in a region on a given layer.
- **Point-to-Point Analysis** - calculate the effective resistance between the selected node pairs or instance pairs. The point-to-point resistance analysis flow ignores all voltage sources. This type of analysis can be done using one of the following inputs:

- specifying instance pairs (`-instance_pair_list`/`-instance_pair_list_file`) - Reports effective resistance between two instances in a design. You can specify multiple instance pairs at a time.

  If the pin name of an instance is specified, the tool will short all nodes only of the specified pin. If an instance pair is specified without their pin names, all nodes of all pins of the instance pair are shorted.

- specifying node pairs (`-node_pair_list`/`-node_pair_list_file`) - Reports effective resistance between two nodes on the power-grid. You can specify multiple node pairs at a time.

Note: Full-chip and point-to-point analyses are mutually exclusive. This means that you can specify `-instance_list` and `-instance_list_file` together but you cannot specify `-instance_list` and `-instance_pair_list` together.

The `analyze_resistance` command can be run without performing power calculation and IR drop analysis. The resistance analysis flow requires you to load the design, set the rail analysis mode, set power nets, set power pads, and run resistance analysis.

Important

You can use the `scale_what_if_resistance` command to scale effective resistance for all analysis methods in the resistance analysis flow, that is, between two nodes on the power-grid, or from a node to all the voltage sources in the design, or from an instance to all the voltage sources in the design, or between two instances. You can use the what-if capability to scale the resistance of the power-grid without changing the design.

Setting up and Running Resistance Analysis

This section describes how to set up and run the following resistance analysis flows:

- domain-based resistance analysis
- net-based node to pad resistance analysis
- net-based node to node resistance analysis

Required Input Files

- Design data (LEF, netlist, DEF)

Procedure - Domain-based Resistance Analysis

To perform domain-based resistance analysis, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

```
read_lib -lef $lefs

read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz
```

2. Define the rail analysis mode through the command line or through the GUI.

   GUI: Select *Power & Rail - Set Rail Analysis Mode*.

   Tcl command:

```
set_rail_analysis_mode \
    -method                     static \
    -accuracy                   xd \
    -analysis_view              AV_wc_on \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } \
    -verbosity true \
    -temperature 125
```

3. Set up domain-based effective resistance analysis through the command line or through the GUI.

   GUI: Select *Power & Rail - Run Resistance Analysis*.

   a. Select the *Domain Based* option.

   b. Select a domain name from the *Domain Name* drop-down list. *Domain Name* is populated automatically if using CPF.

c. Specify the power and ground nets that are in the domain for which resistance analysis will be performed on, in the *Power Net(s)* and *Ground Net(s)* fields. For each net, specify the voltage and threshold values in the *Voltage(s)* and *Threshold* fields.

d. Select the method to generate voltage sources by selecting *Power Pads* and *XY File*.

You can use any one of the following methods to generate voltage sources: *Def Pins*, *Pad File*, *XY File*, *Boundary File*. You can click the *Create* button to fetch all voltage sources of a power or ground net and place it in a text file.

e. Click the *File Name* browse button to specify the power pad file containing the voltage source information.

f. Specify the name of the power net for which the power pad file has been added in the *Net Name* field, and click the *Add* button to add the net to the list. The *Remove* button removes the net from the list.

Note: You must specify power pad information for every PG net that is a part of the domain.

g. Specify the directory name where the resistance analysis results will be stored in the *Output Directory* field.

h. Select the *Effr to Power Pads* option.

i. Click *OK*.

The filled up form appears as shown below:

Tcl command:

```
set_power_pads -reset
set_power_pads -net VSS -format xy \
               -file ../design/super_filter_VSS.pp
set_power_pads -net VDD_external -format xy \
               -file ../design/super_filter_VDD_external.pp
set_power_pads -net VDD_AO -format xy \
               -file ../design/super_filter_VDD_AO.pp
analyze_resistance -domain ALL -output_dir  Reff_domain
```

Procedure - Net-based Node to Power Pad Resistance Analysis (Node to Voltage Source)

To perform net-based resistance analysis, complete the following steps:

Note: You do not need to set up the rail analysis mode again if you have already specified it once for the domain-based resistance analysis.

1. Set up the net-based effective resistance analysis through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Run Resistance Analysis*.

   a. Select the *Net Based* option.

   b. Select a net name from the *Net Based* drop-down list.

   c. Specify the power or ground net for which resistance analysis will be performed on, in the *Power Net(s)* or *Ground Net(s)* field. For the net, specify the voltage and threshold values in the *Voltage(s)* and *Threshold* fields.

   d. Select the method to generate voltage sources by selecting *Power Pads* and *XY File.*

   You can use any one of the following methods to generate voltage sources: *Def Pins*, *Pad File*, *XY File*, *Boundary File*. You can click the *Create* button to fetch all voltage sources of a power or ground net and place it in a text file.

   e. Click the *File Name* browse button to specify the power pad file containing the voltage source information.

   f. Specify the name of the power net for which the power pad file has been added in the *Net Name* field, and click the *Add* button.

   g. Specify the directory name where the resistance analysis results will be stored in the *Output Directory* field.

   h. Select the *Effr to Power Pads* option.

   i. Add the list of nodes using the following fields:

   - In the *Node1* section, click the *Get Coordinate* button and select a region from the *Layout* tab to get the x y coordinates of any node in the design.

     The selected nodes will be displayed in the *x1* and *y1* fields.

   - Select the layer name from the *Layer1* field.

   - Specify a node name in the *Label* field.

   - Click the *Add* button to add the node to the node list.

   - Continue adding nodes to calculate effective resistance between the specified user-defined nodes and all voltage sources.

2. Click *OK*.

   The filled up form appears as shown below:

Tcl command:

```
set_pg_nets -net VSS -voltage 0 -threshold 0.5
set_power_pads  -reset
set_power_pads -net VSS -format xy \
               -file ../design/super_filter_VSS.pp
analyze_resistance -net VSS -output_dir Reff_VSS_node \
                   -node_list {{ 15.968 287.7695 Metal1 n1 } \
                   { 137.117 214.088 Metal5 n2 } \
                   { 99.568 182.207 Metal5 n3 }}
```

Procedure - Net-based Node to Node Resistance Analysis

To perform net-based node to node resistance analysis, complete the following steps:

1. Repeat the steps described in the "Procedure - Net-based Node to Power Pad Resistance Analysis (Node to Voltage Source)" section to specify the net details, power pad file, and output directory.

2. Select the *Point-To-Point Effr* option.
3. In the *Node1* section, click the *Get Coordinate* button and select a region from the *Layout* tab to get the x y coordinates of the first node.

   The selected nodes will be displayed in the *x1* and *y1* fields.
4. In the *Node2* section, click the *Get Coordinate* button and select a region from the *Layout* tab to get the x y coordinates of the second node.

   The selected nodes will be displayed in the *x2* and *y2* fields.
5. Select the layer name from the *Layer1* and *Layer2* fields.
6. Specify a node name in the *Label* field.
7. Click the *Add* button to add the two nodes.
8. Click *OK*.

   The filled up form appears as shown below:

Tcl command:

```
set_pg_nets -net VSS -voltage 0 -threshold 0.5
set_power_pads  -reset
set_power_pads -net VSS -format xy \
               -file ../design/super_filter_VSS.pp
analyze_resistance -net VSS -output_dir Reff_VSS_node_to_node \
-node_pair_list {{ 75.4795 318.9425 Metal1 61.31 272.183 Metal4 n1 }}
```

You can perform effective resistance calculation at the node or instance level using the following fields:

- for a list of nodes using *Node1* field or *Node File*.
- for a list of instances using *Instance1* field or *Instance File*.
- for node pairs between which effective resistance needs to be calculated using *Node1* and *Node2*, or *Node-Pair File*.

- for instance pairs between which effective resistance needs to be calculated using *Instance1* and *Instance2*, or *Instance-Pair File*.

Output

The output directory will contain the following information:

File Name	Description

`effr.rpt`
`domain_effr.rpt`
	`effr.rpt` for net-based analysis and `domain_effr.rpt` for domain-based analysis.
	
	The file reports the total effective resistance values (Rvdd + Rvss) for all the nets in the domain, and the percentage of each net. These files are located in the state directory under the specified resistance analysis directory. For example, the report file for all nets in the domain is located at:
	
	■ domain-based report:
	`Reff_domain/ALL_25C_reff_1/domain_effr.rpt`
	
	You can also view the effective resistance reports for each net. For example, the report file for VSS is located at:
	
	■ net-based report:
	`Reff_domain/ALL_25C_reff_1/VSS/REFF/effr.rpt`

`effr.gif`	This file is generated only for net-based analysis.

Resistance Analysis Reports

This section gives snippets of the report formats for the various net-based resistance analysis methods (using net VDD with one voltage source):

The report format for node pair list is given below:

Node Pair List

```
# Layer Representation: LEF
#       Coordinate Unit: um
#       Resistance Unit: Ohm

NET: VDD

# Threshold: N/A

# Effective resistance for given pair of nodes
# PASS/FAIL        REFF      LABEL        X1        Y1     LAYER1      X2       Y2     LAYER2
  -               1.65432    FILL2      662.760   677.800    M1      661.220  677.800    M1
```

The report format for instance list is given below:

Instance List

```
# Layer Representation: LEF
#       Coordinate Unit: um
#       Resistance Unit: Ohm

NET: VDD

# Threshold: N/A
# Effective resistance for instances sorted from high to low
# *Note:"D/C" means the instance is disconnected from the net.
# PASS/FAIL        REFF                INSTANCE              PIN
  -               1.6543            MFILL_FILL2_773         VDD
  -                1e-08            MFILL_FILL2_770         VDD
```

The report format for instances of a cell is given below:

Cell Instance List

```
# Layer Representation: LEF
#       Coordinate Unit: um
#       Resistance Unit: Ohm

NET: VDD

# Threshold: N/A
# Effective resistance for instances sorted from high to low
# *Note:"D/C" means the instance is disconnected from the net.
# PASS/FAIL          REFF      INSTANCE                                PIN
  -                369.46              MFILL_FILL2_960                VDD
  -                369.35              MFILL_FILL2_199                VDD
  -                368.93              MFILL_FILL2_184                VDD
  -                368.93              MFILL_FILL2_139                VDD
  -                368.92              MFILL_FILL2_148                VDD
  -                368.87              MFILL_FILL2_212                VDD
  -                368.83              MFILL_FILL2_971                VDD
...
...
  -                3.9813              MFILL_FILL2_606                VDD
  -                3.5988              MFILL_FILL2_763                VDD
  -                1.6543              MFILL_FILL2_773                VDD
  -                 1.638              MFILL_FILL2_560                VDD
  -                 1e-08              MFILL_FILL2_770                VDD
```

The report format for instance pair list is given below:

Instance Pair List

```
# Layer Representation: LEF
#       Coordinate Unit: um
#       Resistance Unit: Ohm

NET: VDD

# Threshold: N/A

# Effective resistance for instance pair sorted from high to low
# PASS/FAIL     REFF          INSTANCE1                    INSTANCE2            PIN1    PIN2
  -            1.6543      MFILL_FILL2_770              MFILL_FILL2_773         VDD     VDD
```

The report format for all instances is given below:

```
# Layer Representation: LEF
#      Coordinate Unit: um
#      Resistance Unit: Ohm

NET: VDD

# Threshold: N/A
# Effective resistance for instances sorted from high to low
# *Note:"D/C" means the instance is disconnected from the net.
# PASS/FAIL         REFF              INSTANCE                          PIN
  -               220.74        MFILL_FILL3 _104                       VDD
  -               220.59        MFILL_FILL2 _265                       VDD
  -               220.46        Data8_Pad/inst_140                     VDD
 ...
 ...
  -               1.9615        MFILL_FILL3_ 559                       VDD
  -               1.8999        MFILL_DCAP8_ 767                       VDD
  -               1.6543        MFILL_FILL2_ 773                       VDD
  -                1.638        MFILL_FILL2_ 560                       VDD
  -               1.3489        MFILL_FILL3_ 772                       VDD
  -                1e-08        MFILL_FILL2_ 770                       VDD
```

All Instances

Viewing Resistance Analysis Results

To view the resistance analysis results:

1. Open the *Power & Rail Setup* form, accessed by clicking the *DB Setup* button on the *Power & Rail Plots* form.
2. Load the rail database and effective resistance report (`effr.rpt`) by clicking the *Browse* button next to the *Rail Database* and *Effective Res* fields.
3. Click *OK*.
4. Plot the results using the rail analysis plot type *effr* (`set_power_rail_display -plot effr`).

   The following is a snapshot of an effective resistance plot:

You can analyze the `effr` and `node_reff` (node-based effective resistance) plots in the GUI by specifying the `effr.rpt` for each net in the *Power & Rail Setup* form.

Additional TCL Commands

The following are example TCL commands for running different types of resistance analysis:

- The following command performs effective resistance calculation for the list of nodes and instances:

```
analyze_resistance -net VSS -node_list { 231 1061.6 M1 } -instance_list_file
instance.list
```

- The following command specifying a list of user-defined nodes and a node list file for effective resistance calculation:

```
analyze_resistance -net VDD -node_list {{ 213.556 133.469 M1  I123 }} -
node_list_file /../node_list.txt.1 -output reff_rpt
```

- The following is an example of specifying a list of instances for effective resistance calculation:

```
analyze_resistance -instance_list {{INV11 VDD} {INV21 VDD}} -net VDD
```

- The following is an example of specifying a list of user-defined nodes for effective resistance calculation:

```
analyze_resistance -net VDD -node_list {{800 11.88 M4} {999.768 11.88 M4}
{1000.296 11.880 M4}} -output out.txt
```

- The following is an example of specifying node pair list for effective resistance calculation:

```
analyze_resistance -net VDD -node_pair_list {195.728 133.369 M1 213.556 133.469
M1} -output out_nodelist.txt
```

- The following is an example of specifying instance pair list for effective resistance calculation:

```
analyze_resistance -net GND -instance_pair_list {inst1 inst2 GND GND} -output
outfile -threshold 100
```

- The following command performs effective resistance calculation for full-chip analysis:

```
analyze_resistance -threshold 13.0  -domain_threshold 25.0 -domain ALL
```

- The following command performs effective resistance calculation for instance list analysis:

```
analyze_resistance -domain ALL -threshold 13.0 -domain_threshold 25.0 -
instance_list_file inst.list
```

- The following command reports reff for all instances of the cell type CKBD8:

```
analyze_resistance -net VDD -cell CKBD8
```

Static Rail and EM Analysis

Static IRDrop and EM Analysis Overview

IR Drop and Ground Bounce

IR drop and ground bounce refer to the variations in voltage caused by current flowing through a resistive power network.

IR Drop

IR drop is a reduction in voltage that occurs on power supply networks (VDD) in integrated circuits.

Integrated-circuit design usually assumes the availability of an ideal power supply that can instantly deliver any amount of current to maintain the specified voltage throughout the chip. In reality, however, the combination of increasing current per-unit area on the die and narrower metal line widths (which causes an increase in the power-grid resistance) causes localized voltage drops within the power grid, leading to decreased power supply voltage at cells and transistors.

These localized drops in the power supply voltage decrease the operating voltage of the chip, potentially causing timing problems and functional failures.

Ground Bounce

Ground bounce is an increase in voltage that occurs on ground networks (VSS or GND) in integrated circuits.

The current that is sourced onto the ground network combined with a finite resistance of the ground network leads to localized increases in the ground voltages around the chip.

As with IR drop, these increases in the ground voltage also decrease the operating voltage of the chip, resulting in the same potential timing problems and functional failures.

Concepts in IR Drop and Ground Bounce

The following figure illustrates a power-grid network that helps to explain the concept of IR drop and ground bounce. It shows a chip power supply connected to the chip pads. The power-grid network is illustrated by the R11-R14 resistors for VDD and R21-R24 resistors for

VSS. These resistors represent the resistance of the interconnect from the pads to the cells or transistors. G1-G4 are cells connected to VDD and VSS.

Example Power-Grid Network

When you perform transistor- or gate-level simulation, the voltages (V1- V4) are typically assumed to be equal. It is further assumed that all power-grid resistances are zero ohms and that all chip components receive ideal power supply voltages.

In reality, the power-grid resistances of a chip cannot be ignored. For example, cell G4 never has an ideal VDD voltage at its VDD pin when it is active; it has a lower voltage because the current flowing from the VDD pad to G4 must flow through resistances R11-R14. Ohm's law states that a current, I, flowing through an effective resistance, R, introduces a voltage drop, as given by the equation V = IR.

Note: The term "IR drop" is derived from "IR."

Similarly, cell G4 never has an ideal VSS at its ground pin, because the current flowing from G4 back to the VSS pad must also travel through the ground network resistances R21-R24. In this case, cell G4 has an increased voltage at its ground pin.

The power-grid network diagram also illustrates the complexity of power grids and IR drop. Assume that gate G4 consumes I4 mA and that no other cell is drawing current. This current flows from the power supply through the resistors of the power grid to G4. The IR drop at cell G4 is given by the following equation:

$$IR_{G4} = I4\ (R11+R12+R13+R14)$$

In addition, cell G2 experiences an IR drop of I4(R11+R12) because of the current drawn by cell G4.

Clearly, the current associated with each cell or transistor in a design can influence the IR drop or ground bounce for all other devices in the design.

Average and Peak Current

If the cells on a single metal VDD line switch together, the instantaneous IR drop can be large. Given simultaneous currents I1-I4 for the G1-G4 cells (as shown in Example Power-Grid Network), the IR drop at G4 would be given by the following equation:

$$I_{IR_{G4}} = I1(R11) + I2(R11+R12) + I3(R11+R12+R13) + I4(R11+R12+R13+R14)$$

If this peak current occurred over one fourth of the clock cycle, the average IR drop at cell G4 over the entire clock cycle would be one quarter of that value.

The peak IR drop at any location in a chip can therefore be much larger than the IR drop averaged over a clock cycle. As the size of a block increases, the ratio of the peak IR drop to the average decreases towards 1.0, because the probability of many devices switching at once decreases significantly.

Static IRdrop

For 130-nanometer designs and above, static power-rail analysis is still widely used to validate that the basic power network is sufficiently robust. Static analysis accurately highlights IR drop and electromigration problems caused by issues such as open circuits in the power rails, missing vias or via arrays, high current and high current densities, lack of power straps, and insufficient power routing widths. Static analysis is the preferred approach for power electromigration verification because it emulates the average impact of a design operating over a long period of time. It therefore calculates the average voltage drop. It is a good approach to use when high decoupling capacitance is available. As noted earlier, decoupling capacitors are added to mitigate voltage drops in all available locations.

Static analysis is best used early in the flow and validation process to detect and correct major supply grid problems. Cadence recommends that you first perform static analysis to verify that the power rails are sufficiently robust before you perform dynamic analysis. You should immediately address issues with excessive static IR drop until the IR drop is within user-specified limits (each design team will have their own limits, but a typical goal is 2 to 5 percent). You can establish a baseline for IR drop from the static analysis, then use dynamic analysis to show additional transient IR drop. If you add sufficient decoupling capacitance to these areas, the results from dynamic analysis should be close to the results from static analysis because the transient IR drops will be effectively smoothed.

The following diagram shows the static analysis approach:

Static Analysis Approach

Global and Local IR Drop and Ground Bounce

IR drop and ground bounce can be both local and global phenomena.

- They can be local phenomena when a number of cells in close proximity switch simultaneously, causing IR drop or ground bounce in that localized area. A higher power-grid resistance to a specific portion of the chip can also cause localized IR drop or ground bounce.
- They can be global phenomena when activity in one region of a chip causes effects in other regions. For example, block B2 in the following diagram suffers from IR drop because of the current drawn by block B1.

IR Drop as a Global Phenomenon

In an ideal power grid with equally distributed current and peripheral power pads, the power grid has a set of equapotential surfaces that form concentric circles centered in the middle of the chip, as shown in the following diagram:

Equally distributed current resulting in concentric rings of IR Drop or Ground Bounce

The center of these concentric circles has the largest IR drop or ground bounce, and the amount of the effect decreases toward the pads. Discontinuities in these circles typically identify regions containing discontinuities in the power grid or unequally distributed current flow.

Simultaneous Switching

Large amounts of IR drop or ground bounce can be caused by simultaneous switching of cells and transistors. If all cells and transistors of a design switched at once, the local or global IR drop or ground bounce on a chip would be extremely large. Luckily, the logic itself typically prevents this from happening.

Potential causes of simultaneous switching are the following:

- Synchronized clock networks
- Bus drivers
- Memory word line drivers
- Signal I/O circuitry
- Any groups of large drivers designed to switch together

Symptoms of IR Drop and Ground Bounce

Because the number of failures resulting from poor power-grid networks has become significant only recently, many designers do not look at power distribution as a potential source of chip failure.

Symptoms of IR drop and ground bounce problems include the following:

- Non-functional chips

  If the global IR drop or ground bounce is too high when a chip operates, logic gates malfunction. The failure resembles a logical functional failure or manufacturing problem, although simulation indicates that the design is logically correct. One way of diagnosing this symptom is to increase the power supply voltage and see if the chip works.

- Intermittent or data-dependent functional failures

  Local power-grid problems are sensitized, or forced to occur, by specific operations in close proximity, such as all bits of a bus switching at once. In normal operation, the specific sensitization might not occur. However, a specific data input activates the problem. The symptom appears as a logic functional failure of that portion of the chip.

- Intermittent or data-dependent timing failures

  Like intermittent functional failures, specific data inputs can cause IR drop or ground bounce that appears to be a timing failure. An intermittent timing failure of this type is also symptomatic of a signal net cross-coupling capacitance problem. One way of diagnosing this symptom is to decrease the clock frequency and see if the chip works.

- Hard timing failures

  When the power-grid problems are not high enough to cause complete logic failure, the symptom can be a timing failure of the chip. In this case, the IR drop or ground bounce slows down the speed of the circuitry, causing a hard timing failure.

- Clock signal jitter

  Experiments have shown that a 5 percent IR drop on a clock buffer can slow down its speed by up to 15 percent. The reduction in speed is due to two mechanisms, as shown in the following diagram:

Impact of IR Drop on speed

First, the IR drop on the power input of inverter1 slows the charging rate of the output. Explained simply, the output voltage of inverter1 rises only up to VDD minus the IR drop. In turn, this drop reduces the logical high input voltage (VIH) to inverter2, reducing its switching speed as well.

Two problems have resulted from the traditional approach to working around potential IR drop problems:

- Timing libraries are constructed on the basis of assumptions about a certain amount of IR drop, such as 10 percent. These assumptions add to the timing margins applied to the design. The added margins are becoming quite significant and impede timing closure. As a result, the final chip timing is increasingly unpredictable. Chips can fail because the IR drop was actually higher, or they can operate much faster than the specified frequency.

- Power grids are heavily over designed. These grids remove a significant amount of chip area from use for signal routing and may result in larger chips. In addition, designers frequently over design the power grid but by an unknown amount.

Manufacturing Effects in Power-grid Extraction

The extractor embedded in Voltus IC Power Integrity Solution can include the effects of manufacturing on resistance, capacitance, or both in the extraction results. In copper processes and processes at or below 130 nanometers, the manufacturing steps of resist, lithography, etching, and chemical mechanical polishing (CMP) result in differences between the layout drawn by designers and the manufactured design. Earlier processes exhibited minor effective changes in a wire's characteristics, such as a change of wire width resulting from a trapezoidal wire shape. Copper processes also exhibit a trapezoidal effect, although more pronounced. In addition, copper processes also exhibit more significant variations in wire width and thickness with respect to the drawn layout.

Accurate prediction of circuit behavior in deep submicron silicon requires the modeling of these manufacturing effects. In the past, it was difficult to obtain measurements and specifications of these manufacturing effects, so they were generally represented by process variations. As consistent sources of these variations have been identified, extraction and analysis tools have been developed that include not only accuracy, capacity, and performance, but models for these deep submicron effects. With each new process generation, the number of such models is expected to increase.

The manufacturing effects on a wire that cause variation from the drawn wire are wire-edge enlargement, resistivity as a function of wire width and erosion.

- Wire-edge enlargement refers to the effective change to a wire's width as a function of its width and spacing to its neighbors, because of its optical proximity to neighboring wires.
- Resistivity as a function of a wire's width refers to the effective variations in a wire's resistivity because of dishing, slotting, and cladding thickness.
- Erosion refers to the change in a wire's thickness as a function of metal density in a portion of a chip. The Fire & Ice ZX extractor computes the manufactured dimensions of a wire (width and thickness) and applies parasitic models to the resulting wire.

These effects are described in detail in the following sections.

Wire-Edge Enlargement

Before manufacturing, reticle enhancement technologies (RET), such as optical proximity correction (OPC) and phase shift masks (PSM), are employed to help create the correct dimensions on silicon from the drawn layout. Even with the use of these advanced techniques, the manufacturing process contains optical effects that result in systematic wire width variations that must be accounted for in nanometer designs.

Basically, the coherent light source that is used to expose a mask onto the wafer creates diffraction patterns around the edges of the mask geometries. These patterns cause a variation in the exposure, which eventually results in a variation of the spacing, and therefore the width, of the wires.

Foundries usually focus the lithographic equipment to allow an increase in the width of wires only to avoid a reduction in wire width that may cause fatal open circuits. The size of the width increase is influenced by the width of the wire in question, its proximity to neighboring wires, and the size and spacing of the neighboring wires. The following diagram illustrates the results of this wire-edge enlargement:

Wire Edge Enlargement

The spacing and width of the neighboring wire may enlarge each wire edge by a different amount.

Resistivity as a Function of Width ($R_s = f(W_s)$)

Variation in resistivity is caused by a number of phenomena. Copper is a softer material than the dielectric in which it is embedded. As a result, the polishing of the wafer during the CMP process has a tendency to remove a little extra copper from the top of the wire. This effect is called dishing because of the shape of the resulting wire top. The effect becomes more pronounced as the wire width increases. This effect is shown in exaggerated form in (A) in the following figure:

Dishing with Erosion and Without Erosion

To reduce the effect of dishing on wide wires, small holes, or slots, can be inserted at regular intervals in wide wires. These slots insert a form of hard "posts" in the wire so that the CMP process removes less copper. This technique reduces the dishing, as well as the effective resistivity.

Another contributor to resistivity variation is the cladding in copper wires. Cladding is the material grown around the sides and bottom of copper wires to protect them from chemical reactions with the dielectric material. This cladding is illustrated in gray in the above diagram. The thickness of the cladding on the sides and bottoms of wires also varies with the width of a wire. Because cladding has a much higher resistance than copper, it impacts the effective resistivity of copper wires. This effect is more pronounced in the narrowest wires.

The combination of the effects of dishing, slotting, and cladding thickness is modeled by the wire resistivity as a function of the wire width in silicon. The drawn width of a wire is not representative of the silicon width of a wire after wire-edge enlargement effects are considered.

Erosion

Erosion is the wearing away of the top of the metal by the CMP process. Dishing is limited to a single wire, but erosion impacts a large area of the wafer. It is caused by the varying density of metal across the die or wafer.

Variations in metal density cause the CMP process to over-polish areas of high metal density compared to areas of low metal density, because the amount of harder dielectric varies with the metal density. High metal density is associated with low dielectric density, and vice-versa, and this affects the general hardness of an area. The impact of metal density on erosion is

illustrated in an exaggerated form in Dishing with Erosion and Without Erosion. (A) shows dishing without erosion, and (B) shows dishing with erosion.

The impact of erosion on a design can be quite significant. Measurements have shown that high metal-density areas can suffer up to a 30 percent reduction in metal thickness.

Dishing with Erosion and Without Erosion also illustrates an important difference between dishing and erosion. Erosion has a greater impact on capacitance values than dishing. The sides of a wire with dishing effects are of the desired thickness. But erosion affects the thickness of the wire edges, causing a decrease in lateral coupling capacitance.

In order to minimize erosion, most manufacturers have established metal density rules that force a more consistent density of metal (and therefore of dielectric) throughout a design. Although these rules help to minimize the impact of erosion, extraction tools must model erosion for nanometer designs.

Metal Fill

As noted previously, the thickness of wires after CMP varies as a function of the density of metal in an area. In the ideal case, all layers of the entire chip would have a uniform density of about 50 percent (effectively, minimum width and spaced wires), but this is not possible in practice. Extreme ranges of density may have highly non-linear wire thickness. Therefore, thickness variation is managed to some extent through design rules that require all areas of a chip to have densities greater than a specified amount, such as 20 percent, and less than another amount, such as 80 percent. It is common for portions of a design on a specific layer to have a very low density of metal in wires. To increase the overall density of metal in these areas, pieces of metal are inserted in the layout that have no logical function in the circuit but serve only to increase the density of metal. These pieces of metal are called metal fill. The use of metal fill can vary in terms of the size, shape, spacing of the pieces, and whether they are connected to electrical nets in the design, such as VDD or VSS.

In the past, metal fill was inserted at the manufacturing stage of a tapeout. However, since inserted metal fill can influence the resistance and capacitance parasitics of wires in the design, it is now necessary to generate metal fill in design tools and to model the impact of metal fill in parasitic extraction tools. In a typical densely routed design, metal fill has little impact on wire capacitance. In the worst case, however, metal fill can vary wire capacitance by more than 100 percent. Floating metal fill, which refers to those pieces not connected to electrical nets of the design, contributes less to overall net capacitance but can increase the magnitude of coupled capacitance to other nets. The implementation style of the metal fill can therefore have an impact on timing and signal integrity, so it must be modeled in extraction.

You can choose two types of metal fill:

- Grounded, which has a fixed electrical charge. Typically, grounded metal has a higher capacitance.
- Floating, which has no fixed electrical charge

Metal fill is usually specified in the input DEF file. If the metal fill in the DEF file is incomplete or otherwise unsatisfactory, you can import the metal fill from an external GDSII file, frequently obtained from a foundry. When you use this GDSII file, Fire & Ice ZX ignores the metal fill polygons in the input DEF file.

Electromigration (EM)

Electromigration is used as a general term to describe failure mechanisms in the metal wires of a computer chip caused by the movement of metal atoms in a wire because of high current stress. As electrons move through a metal wire, they collide with the atoms in that wire. These collisions cause wires to become heated, and if enough electrons collide with a metal atom over a period of time, the metal atom may move in the direction of the electron flow, causing two failure mechanisms:

- First, if enough atoms are moved, the wire effectively breaks and becomes an open circuit.
- Second, if enough atoms move to the same location, a short to an adjacent metal wire can be created. This phenomenon is commonly known as fusing.

Either of these mechanisms changes the functionality of the chip.

Two distinct physical phenomena cause electromigration: wearout and Joule heating.

Wearout

Electromigration in a specific sense is a long-term wearout mechanism of chip interconnect wires.

As metal atoms are moved and the wire becomes narrower (in the location from which the atoms are moving), the resistance of the wire increases. If atom movement continues, the wire eventually breaks.

Because wearout can take a significant amount of time to be observed, failure due to wearout is measured as a statistical process and is characterized by a wire's mean time to failure (MTTF).

To reduce the impact of electromigration, chip interconnect wires on a given metal layer are built from several metal materials in a sandwich fashion. The extra metal layers in the wire,

generally the top and the bottom layers in the sandwich structure, are more resistant to electromigration, so they help prevent total wire failure because of breaks. Electromigration failures are therefore defined as an unacceptable increase in the resistance of the wire rather than an outright break in the wire. Because electromigration is the result of long-term wearout, the risk to a wire is measured in terms of the long-term average current flowing through the wire.

There remains some debate in the reliability community about the proper model for wires exposed to alternating current flow. One approach indicates that when the average current is 0 A, no electromigration occurs. Another approach says that the recovery from the reversed current is not full, so the average should be the peak in one direction minus a recovery factor times the reverse current. A recovery factor of zero indicates no recovery, and a factor of 1.0 indicates full recovery.

Joule Heating

Joule heating refers to the excessive heating of a specific segment of a wire because of high alternating currents. Wires are heated as alternating current moves through them. This heating occurs regardless of the direction of the current. If the heating of the wire becomes significant, thermal expansion and temperature-induced electromigration occurs. Since the risk of Joule heating increases with temperature, and increasing frequency increases heating effects, electromigration due to Joule heating is associated with high-frequency wires.

Because the root cause of Joule heating failure is different from electromigration, different measures are applied in verification.

Fusing

Fusing is a specific failure mechanism related to electromigration. It refers to the instant failure of a wire segment because of excessive instantaneous current. To address this failure mechanism, different measures are applied in verification. Some reliability researchers classify this failure mechanism as a subset of Joule heating.

Symptoms of Electromigration Problems

Because electromigration is a wearout phenomenon, its symptoms are primarily a change in either the timing or functionality of a chip over time. Wires that provide unique connectivity in the circuit cause total functional failure if they fail because of electromigration. Wires that are inherently redundant, such as a meshed power grid, exhibit symptoms of higher IR drop or ground bounce after electromigration failure results in disconnected sections of the power grid. If electromigration causes a short between wires on a chip, total functional failure occurs.

Advanced Processing Techniques

Using metals such as titanium nitride and copper in chips can also impact electromigration.

Titanium Nitride

To reduce the impact of electromigration, chip interconnect wires on a given metal layer are built from several metal materials in a sandwich fashion, as shown in the following diagram:

Multiple Materials Used to Build a Single Wire

Generally the top and the bottom layers in the sandwich structure are constructed from titanium nitride (TiN), which is more resistant to electromigration. This structure helps to prevent total open circuit failures; however, because the resistivity of the TiN layer is greater than aluminum, electromigration failures are defined as an unacceptable increase in the resistance of the wire rather than an outright break in the wire.

Copper Interconnect

State-of-the-art process technologies use copper as chip interconnect material because copper offers not only lower resistivity but also higher resistance to electromigration wearout. However, copper interconnect structures are not as deep as aluminium structures, so the cross-section area is smaller and therefore the current densities are higher. As a result, electromigration-based failures still occur in copper structures.

In contrast to aluminum-based processes, vias built with copper increase the likelihood of electromigration wearout and require more detailed verification.

Evaluating Electromigration Risk

Electromigration occurs in metal wires that conduct high current densities. Conducting electrons collide with diffusing metal atoms and impel the atoms in the direction of electron flow. This collision produces a mass flux opposite to that of the current. Divergences in this

mass flux can result in damage to the conductor in the form of voids or hillocks. You can minimize this problem by limiting the current density in metal conductor lines, vias, and contacts.

Electromigration resistance is primarily a function of processing parameters. Many of the variations observed in the lifetime behavior of conductors because of electromigration cannot be attributed to a specific cause. In addition, material contains significant lot-to-lot variations. To fully understand electromigration within the chip, you must be familiar with the specific properties of your own material and process.

In some cases, complete data is not available. The following suggestions may help you in making an evaluation. The figures quoted here are from relevant literature over the past 25 years; you should consider them non-definitive values and not necessarily representative of your material or process. Cadence provides these figures only as background information.

In recent evaluations of electromigration performance in the industry, it has become commonplace to observe that failures in lines connected to bond pads obey $1/j^2$ kinetics and that lines connected to vias and contacts usually obey 1/j kinetics. Therefore, these are the default values used in this simulation. If you have evidence that your process provides other kinetics, you can make suitable changes in the modeling.

The following table shows the pre-exponential and activation energy for line and via electromigration failure for wide and narrow lines for a variety of alloys. The values chosen are conservative and represent the latest data available from leading industrial laboratories. Al/Si figures are estimated from the expectation that lifetimes without copper alloying are about 10 percent of those with copper. No recent data on this assumption is available, and earlier data is questionable in view of recent advances in electromigration testing procedures.

Table 8-1 Pre-exponential and activation energy for line and Via electromigration failure

Metal Configuration	n	A	Ea
Al/Si wide line	2	130 to 20,000	0.5 eV
Al/Si narrow line or Al/Cu wide line, with or without barrier shunt layer	2	1,300 to 80,000	0.6 eV
Al/Cu narrow line without barrier shunt layer	2	1,300 to 200,000	0.8 eV
Al/Cu narrow line with barrier shunt layer	2	13,000 to 800,000	0.8 eV
Al/Si via	1	.00001	0.8 eV

Al/Si via	2	7.5	0.5 eV
Al/Cu via without barrier	1	.0001	0.8 eV
Al/Cu via without barrier	2	75	0.8 eV
Al/Cu via with barrier	1	.0004 to .008	0.8 eV
Al/Cu via with barrier	2	300 to 16,000	0.8 eV

Note: These values are not independent. If you choose a different value for the activation energy, it is necessary to determine another value for *A*.

- *n* is either 2 for nucleation-dominated failure or 1 for growth-dominated failure.
- *A* is the technology- and structure-dependent pre-factor. The units of *A* are the following:
  - Amp-hours per cm^2 degree for *n*=1
  - Amp^2-hours per cm^4 degrees for *n*=2
- *Ea* is the activation energy for the electromigration mechanism.

For vias, the current density is defined as the current density in the wire immediately contacting the via or the contact to silicon. This density is assumed to be for a line with a width equal to the via. Al and Al/Si are assumed to be equivalent, and all Al/Cu alloys are assumed to be created equal.

Modeling electromigration risk

Electromigration risk analysis requires the use of failure models. The choice of failure model depends on the particular manufacturing process and design rules. The failure kinetics vary according to the metal line width. They must be determined from the test structures. Model parameters such as *A_n*, *A_w*, and *n* (the current density coefficient) are used to describe the predominant failure kinetics.

On the basis of the calculated average current density, Cadence analysis tools use Black's Equation to determine the mean time to failure (MTTF):

$$t_{50} = \frac{AT^m}{j^n}\exp\left(\frac{Ea}{kT}\right)$$

- *n* is either 2 for nucleation-dominated failure or 1 for growth-dominated failure.

- `A` is the technology- and structure-dependent pre-factor.
- `T` is the temperature.
- `m` is the temperature coefficient.
- `j` is the average current density.
- `Ea` is the activation energy for the electromigration mechanism.
- `k` is Boltzmann's constant.

The MTTF is then used to calculate the probability of failure for a line segment:

$$P_{fi}(t) = \frac{\sigma}{\sqrt{2\pi}\,\ln(t_{50}/t)} \exp - \left(\frac{\ln(t_{50}/t)}{\sqrt{2\sigma}} \right)^2$$

- `t` is the expected lifetime of the design, which you can specify by using the `Lifetime` environment variable.
- σ is the lognormal standard deviation. It is process-dependent and must be determined by using test structures. It is a model parameter that you can specify layer by layer.

After Cadence analysis tools calculate the failure probability for each wire segment, they use the following calculation to determine the failure probability for the full chip:

$$p_f(n, t) = 1 - \prod_{i=1}^{n} (1 - P_{fi}(t))$$

This approach assumes a lognormal failure distribution, which is analogous to the weakest link in a chain.

Cadence analysis tools report the overall probability of failure after they finish the analysis. They report the inverse of the failure probability for each wire segment as electromigration risk.

You must specify several processing parameters for each layer to obtain good results from electromigration risk analysis.

EM Modeling using Voltus Format

The command `set_rail_analysis_mode -em_models` file is used to point to the file that includes the models needed for EM analysis.

Important

The `-process_techgen_em_rules` and `-em_models` parameters of the `set_rail_analysis_mode` command are mutually exclusive. If none of these parameters are specified, `rj` analysis will not be performed.

The use model for temperature effects on resistance is given below:

- Sources of TC1/TC2/TREF will follow the same use model as of EM rules. Voltus will read both TC1/TC2/TREF and EM rules from one of the following three sources: `em model file`, `.cl`, or extraction technology file. These sources are mutually exclusive.
  - When `set_rail_analysis_mode -process_techgen_em_rules` is `true` and `-extraction_tech_file` is specified, the extraction technology file settings will be used.
  - When `-em_models` is provided, em model file will be used.
  - In other cases, `.cl` file will be used.
- If the temperature coefficients are not set in the em model file, they will be reset to default settings (0 for `TC1` and `TC2`, 25 for `Tref`) and a warning message will be displayed.
- In the em model file, `Tref` needs to be defined on the layer "`ALL`".

Normal Modeling

```
model [layer|all] parameter1 = value1[parameter2 = value2...parametern =
     valuen]
```

PWL Modeling

```
model [layer|all] parameter = "value_1 width_1 value_2 width_area_2 [value_3
     width_area_3 [...value_n width_area_n]]"

model metal_layer jmax_parameter = "jmax_value_1 width_1 jmax_value_2 width_2
     [jmax_value_3 width_3 [...jmax_value_n width_n]]"<via_range=number>

model via_layer jmax_parameter = "jmax_value_1 area_1 jmax_value_2 area_2
     [jmax_value_3 area_3 [...jmax_value_n area_n]]"<via_range = number>
     <current_direction = [up | down | both]>
```

```
model layer_name jmax_length_factor = "factor_1 length _1 ... factor_n
     length_n"
model jmax_lifetime [jmax_dc_avg | jmax_dc_peak | jmax_dc_rms] layer_name
     "scale1 Lifetime1 scale2 Lifetime2 ... scalen Lifetimen"
```

The `model` command lists or changes resistor model parameters, which makes it possible for Voltus to perform electromigration verification. The model parameters provide the following information:

- What layers are to be modeled, such as metal layers and via layers
- Layer dimensions
- Current density information

Parameter

layer	Specifies the name of the layer for which the software is to set parameters. This name should be the same as the layer name defined in the RCgen and XTC command file.
all	Indicates all existing layers. These layer names should be the same as the layer names defined in the RCgen and XTC command file.
parameter1 = value1 area1 *parameter2 = value2 area2* . . *parametern = valuen arean*	Specifies the `model` parameter and its new value. If you do not include this argument, the software lists the current model settings. This is for normal modeling. **Note:** There must be spaces before and after the equals sign.

`parameter = "value_1 width_area_1 value_2 width_area_2 value_2 width_area_2 ...[value_n width_area_n]"`	Specifies the points of the piecewise linear interpolation. Value is one of the following parameters: ■ *Ea* ■ *A* ■ *Jmax_dc_avg* ■ *Jmax_dc_rms* ■ *Jmax_dc_peak* If you specify piecewise linear current density limits for a metal layer, *width_area_n* must be the width of the wire, in meters. If you specify the piecewise linear current density limits for a via layer, *width_area_n* is the area of the via, in meters squared. You must define at least two points in a PWL model.
jmax_parameter	jmax_parameter is one of the dc parameters (i.e. `jmax_dc_avg`, `jmax_dc_rms`, `jmax_dc_peak`).
`via_range=` *number*	The permitted current on a metal layer can be specified depending on the number of vias. This is done using the `via_layer` option. If this option is not used, all currents defined will pertain, independent of the number of vias. "`model` *metal_layer via_range*" specifies the number of parallel vias encountered when going from the referenced layer to the top metal layer. Multiple `model via_range` commands are used to specify the different currents for the different ranges in number of vias. The order of these multiple commands are not important. Any via ranges not specified will apply the closest one. An example with three model commands is shown below.
`current_direction =` *[up | down | both]*	Provides the ability to specify EM model parameters based on direction. The default is `both`.

jmax_length_factor	Specifies the length dependent adjustment factor as a PWL function. The length values are specified in meters.
jmax_lifetime	Specifies the current density limit for different lifetimes. Example: `jmax_lifetime jmax_dc_avg metal1 "4.3 5 1 10 0.4 15"`
jmax_dc_avg jmax_dc_rms jmax_dc_peak	Specifies dc `jmax_factor` for average, peak, or rms currents.
scale1 scale2 ... scalen	Specifies the scaling factor used to calculate the current density limit.
Lifetime1 Lifetime2 ... Lifetimen	Specifies the lifetime value. (in hours)
TC1 = "value_1 width_1 value_2 width_2 ...value_n width_n"	Specifies the model parameter TC1/TC2, its new value, and the width corresponding to the new value. The width values are specified in meters.

Example

- The following is an example of using the `via_range` option.

  ```
  model via_layer jmax_dc_avg = "jmax_value_1 area_1 jmax_value_n area_n"
  via_range=2 ;
  ```

  Where, `via_range=2` refers to number of vias from 1 to 2.

  ```
  model via_layer jmax_dc_avg = "jmax_value_1 area_1 jmax_value_n area_n"
  via_range=4 ;
  ```

  Where, `via_range=4` refers to number of vias from 3 to 4, since 1 to 2 was covered by prior command.

  ```
  model via_layer jmax_dc_avg = "jmax_value_1 area_1 jmax_value_n area_n"
  via_range=8 ;
  ```

  Where, `via_range=8` refers to number of vias from 5 to 8, since 1 to 4 was covered by the prior two commands. For any cases that have more than 8 vias, this model will also be applied.

  The `via_range` rules on metal layers are only applied to resistors that belong to a segment with a via connection to the upper layer. If there is no via connection (i.e top

metal layer), then the default rule (without `via_range`, like command shown below) will be used.

```
model via_layer jmax_dc_avg = "jmax_value_1 area_1 jmax_value_n area_n"
```

If wire width and number of vias rules are both specified, the number of vias rule will be applied first.

The "`info r`", "`which r`", and "`print rj`" commands will list the number of via cuts to be used in EM modeling. All of these commands require a previous EM analysis with `via_range` models performed, since only then will it run the connectivity retracing to account for via connections.

Temperature effects on resistance are available. The model parameters for each layer include fields for TC1, TC2, and TREF. The resistance of a resistor is calculated by using the following formula:

$$R_{effective} = R*(1 + TC1 * delta_t + TC2 * delta_t^2)$$

This syntax contains the following parameters:

- `delta_t` - Specifies the default temperature minus the nominal model temperature (`TREF`).
- `TC1` - Specifies the linear temperature coefficient. The default is 0.0 in 1/degrees.
- `TC2` - Specifies the quadratic temperature coefficient. The default is 0.0 in 1/degrees2.
- `TREF` - Specifies the nominal model temperature, in degrees Celsius. The default is 25 degrees Celsius.
- `TC1` and `TC2` come from the foundry. `TREF` refers to the temperature at which the technology file was generated. The temperature variable sets the operating temperature at which you want to run the analysis. The tool then recalculates the resistance from the extracted `TREF` at the analysis temperature that you want.

Below is an example of the contents of the EM model file:

EM model file example

```
##############################################################
# Example foundry specification current density based EM check
# Generated for 110C
# METAL_1 => Imax = 0.36*2.0*(w-0.02)
# METAL_2 to METAL_6 => Imax = 0.36*2.6*(w-0.02)
# METAL_7 and METAL_8 => Imax = 0.36*4.48*(w-0.02)
# where w is width.
# Contact => 0.294mA/Cont
```

```
# VIA_1 to VIA_6 => 0.189mA/via
# VIA_7 and VIA_8 => 1.452mA/via
##############################################################
# Jmax Calculation: e.g. METAL_1
# Imax = 0.36*2.6*(w-0.02)
# For w= 0.12u, Imax = 0.36*2.0*(mA/u)*(0.1u) = 0.09mA
# Jmax = Imax/(Thickness*width) = 0.09 e-3 (Amp)/(0.24e-4*0.12e-4) (cm^2) = 0.3e+5
A/cm^2
#
# Thickness for the layer can be found in ICT file.
# For different widths, you can generate the PWL limits and Voltus will interpolate
between these values based on width of the resistor, e.g.
# model METAL_1 jmax_dc_avg  =   "0.25e+5 0.1e-6 0.3e+5 0.12e-6 0.38e+5 0.14e-6
1.25e+5 0.16e-6 1.61e+5 0.2e-6 2.51e+5 0.3e-6 5.21e+5 .6e-6 8.81e+5 1e-6 35.8e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
# where first value is calculated Jmax and second is width. "Jmax width1 Jmax width2
Jmax width3...."
## METAL_8
model METAL_8 TREF           =   110
model METAL_8 Tc1            =   0.0000
model METAL_8 Tc2            =   0.0000
model METAL_8 thickness      =   0.7744e-06; #Meters
model METAL_8 jmax_dc_avg  =   "0.74e+5 0.1e-6 0.93e+5 0.12e-6 1.11e+5 0.14e-6
1.3e+5 0.16e-6 1.67e+5 0.2e-6 2.6e+5 0.3e-6 3.72e+5 .42e-6 5.39e+5 0.6e-6 9.11e+5
1e-6 37e+5 4e-6";  #[A/cm^2]  pwl limits according to widths.
## VIA_7
model VIA_7 TREF           =   110
model VIA_7 Tc1            =   0.0000
model VIA_7 Tc2            =   0.0000
model VIA_7 vcwidth        =   0.36e-06;  #Meters
model VIA_7 jmax_dc_avg_w  =   1.452e-03;  #[A/Via]
## METAL_7
model METAL_7 TREF           =   110
model METAL_7 Tc1            =   0.0000
model METAL_7 Tc2            =   0.0000
model METAL_7 thickness      =   0.2986e-06; #Meters
model METAL_7 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
## VIA_6
model VIA_6 TREF           =   110
model VIA_6 Tc1            =   0.0000
model VIA_6 Tc2            =   0.0000
model VIA_6 vcwidth        =   0.13e-06;  #Meters
model VIA_6 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]
## METAL_6
model METAL_6 TREF           =   110
model METAL_6 Tc1            =   0.0000
model METAL_6 Tc2            =   0.0000
model METAL_6 thickness      =   0.2986e-06; #Meters
model METAL_6 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
## VIA_5
model VIA_5 TREF           =   110
model VIA_5 Tc1            =   0.0000
```

```
model VIA_5 Tc2            =   0.0000
model VIA_5 vcwidth        =   0.13e-06;  #Meters
model VIA_5 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]

## METAL_5
model METAL_5 TREF           =   110
model METAL_5 Tc1            =   0.0000
model METAL_5 Tc2            =   0.0000
model METAL_5 thickness      =   0.2986e-06; #Meters
model METAL_5 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.

## VIA_4
model VIA_4 TREF           =   110
model VIA_4 Tc1            =   0.0000
model VIA_4 Tc2            =   0.0000
model VIA_4 vcwidth        =   0.13e-06;  #Meters
model VIA_4 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]

## METAL_4
model METAL_4 TREF           =   110
model METAL_4 Tc1            =   0.0000
model METAL_4 Tc2            =   0.0000
model METAL_4 thickness      =   0.2986e-06; #Meters
model METAL_4 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.

## VIA_3
model VIA_3 TREF           =   110
model VIA_3 Tc1            =   0.0000
model VIA_3 Tc2            =   0.0000
model VIA_3 vcwidth        =   0.13e-06;  #Meters
model VIA_3 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]
#model VIA_3 jmax_dc_peak  =   0.713e-03;  #[A]

## METAL_3

model METAL_3 TREF           =   110
model METAL_3 Tc1            =   0.0000
model METAL_3 Tc2            =   0.0000
model METAL_3 thickness      =   0.2986e-06; #Meters
model METAL_3 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.

## VIA_2
model VIA_2 TREF           =   110
model VIA_2 Tc1            =   0.0000
model VIA_2 Tc2            =   0.0000
model VIA_2 vcwidth        =   0.13e-06;  #Meters
model VIA_2 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]

## METAL_2
model METAL_2 TREF           =   110
model METAL_2 Tc1            =   0.0000
model METAL_2 Tc2            =   0.0000
model METAL_2 thickness      =   0.2986e-06; #Meters
model METAL_2 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_2 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]
```

```
## VIA_1
model VIA_1 TREF           =   110
model VIA_1 Tc1            =   0.0000
model VIA_1 Tc2            =   0.0000

# vcwidth is the default via width.
model VIA_1 vcwidth        =   0.13e-06;  #Meters
model VIA_1 jmax_dc_avg_w  =   0.189e-3;  #[A/Via]

## METAL_1
model METAL_1 TREF           =   110
model METAL_1 Tc1            =   0.0000
model METAL_1 Tc2            =   0.0000
model METAL_1 thickness      =   0.24e-06; #Meters
model METAL_1 jmax_dc_avg  =   "0.25e+5 0.1e-6 0.3e+5 0.12e-6 0.38e+5 0.14e-6
0.42e+5 0.16e-6 0.54e+5 0.2e-6 0.83e+5 0.3e-6 1.75e+5 0.6e-6 2.96e+5 1e-6 11.96e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.

# CONT
model CONT TREF    =   110
model CONT Tc1            =   0.0000
model CONT Tc2            =   0.0000
model CONT vcwidth        =   0.12e-06;  #Meters
model CONT jmax_dc_avg_w  =   0.294e-3;  #[A/Cont]
```

EM Modeling using QRC Technology File

In addition to the `set_rail_analysis_mode -em_models` file command parameter, you can use the QRC Techfile for EM model specification. The `set_rail_analysis_mode -process_techgen_em_rules true` parameter specifies to process the EM models in the ICT section of the qrcTechFile.

Important

The `-process_techgen_em_rules` and `-em_models` parameters of the `set_rail_analysis_mode` command are mutually exclusive. If none of these parameters are specified, rj analysis will not be performed. When -

`process_techgen_em_rules` is specified with `-extraction_tech_file`, the extraction technology file will be processed. If `-process_techgen_em_rules` is specified without `-extraction_tech_file`, the .cl model will be processed.

Note: You need to use Techgen to update the qrcTechFile with the EM model information. Techgen can be found under the bin directory of the Quantus QRC installation hierarchy. Use the following Techgen command to include the EM rules from the specified .ircx file or ICT file into the qrcTechFile for all the process nodes:

```
Techgen -update_em techfile
iRCX_file_with_EM_models/ict_file_with_EM models
```

The EM models are available in the specified IRCX or ICT-EM (ICT with only the `em_model` section) file. For more information, refer to the "**Techgen Commands**" chapter in *Quantus Techgen Reference Manual*.

Instead of updating the qrcTechFile, you can also use the following command to directly specify the ICT-EM file in Voltus to calculate the EM limits for power net EM analysis:

```
set_rail_analysis_mode -process_techgen_em_rules true -ict_em_models
ICT-EM file
```

For detailed information on the ICT-EM file format, refer to the “EM Only ICT File” section in the **File Formats** chapter of the *Voltus User Guide*.

If you are defining an `em_model` to be used for limit based analysis, specify the `em_jmax_ac_avg` parameter that specifies the maximum AC average current density which can either be defined as a single value, a PWL table, or an equation (EQU). The following examples illustrate the limits supported:

- Single value - use a single number limit for current density.

  ```
  em_jmax_dc_avg 12.15
  ```

- PWL table - use piecewise linear (PWL) interpolation in the `em_model` to accurately determine current density using interpolation of specified points on a graph. It allows you to specify Jmax value for each width/area of metal/via in the design to support width dependency on limits.

  - Conductor: `em_jmax_dc_avg = "jmax_value_1 width_1 … jmax_value_n width_n"`
  - Via: `em_jmax_dc_avg = "jmax_value_1 area_1 … jmax_value_n area_n"`

  ```
  em_jmax_dc_avg PWL 0.1672 0.002025 0.3344 0.005265
  ```

- Equation - use an equation to calculate current density. The EQU keyword denotes the definition of an equation for current density. It supports width dependencies.

  ```
  em_jmax_dc_avg EQU 1 * 0.996 * ( w - 0.003 )
  ```

Topology-Dependent EM Rule Check

Permitted current on the metal routing layer will change depending on the number of vias. When many vias are placed, the permitted current of metal layer on EM rule will be large.

When a small number of vias are placed, the permitted current of metal layer on EM rule will be small.

The permitted current (Jmax) value can be defined based on the metal width and the number of vias. `via_range` is an optional parameter which can be used with the via layers. The value of `via_range` denotes number of vias for which the calculated limits are valid.The following is an example of using the via_range option:

```
em_jmax_dc_avg 12.15 via_range 9 …
em_jmax_dc_avg 5.4 via_range 4 …
```

`via_range` means the number of vias in the array. It could be 3x3 or 1x9, 2x2 or 1x4, respectively. It allows you to handle current density limit on a via based on number of vias.

Temperature Dependent Current Density Limits

You can handle temperature dependent current density limits by adjusting the Jmax value by a user defined co-efficient value which considers the temperature and life-time effect.

- `jmax_factor` is the optional scaling factor to use at different temperatures compared to the reference temperature.

  ```
  jmax_factor <temp1> <scale1> [<temp2> <scale2>...]
  ```

  `em_jmax_dc_avg 12.15` **jmax_factor** `105 1.434 110 1.000 115 0.704 120 0.500 125 0.358`

- `jmax_lifetime` provides the ability to set the scaling factor that applies to the current density limits for different lifetimes.

```
jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>...]
```

```
em_jmax_dc_avg EQU 3 * (w - 0.00485) jmax_lifetime 40000 2.05 50000 1.73 75000
1.26 100000 1.0 110000 0.924 150000 0.711 200000 0.556
```

Length and Width Dependency

The `L` and `W` parameters are used to specify the length and width-based jmax values in microns for both metal and vias. These parameters are used when the dimensions to be measured are the same for the metal above and below a via.

```
em_jmax_dc_avg EQU 1 * 1.227 * ( w - 0.008 ) jmax_factor 105 1.434 110 1.000 115
0.704 120 0.500 125 0.358 L > 10 W < 0.5
```

```
em_jmax_dc_avg EQU 1.5 * 1.227 * ( w - 0.008 ) jmax_factor 105 1.434 110 1.000 115
0.704 120 0.500 125 0.358 L > 5
```

For long metal interconnect segments, current density limits are given as a constant. As wires become shorter (i.e. below 20um), there is a point where electromigration becomes less and less of a problem. There is also a lower limit for the length of the interconnect known as "Blech length", and any wire that has a length below this will not fail by electromigration. The "Blech length" is modeled using higher current density limits for shorter wires.

Important

The following is an example of EM rules:

```
em_jmax_dc_avg EQU 1 * 1.06 * ( w - 0.003 ) jmax_factor 100 1.45 105 1.00 110
0.70 jmax_life 87600 1.0 L > 3 W < 0.04
```

```
em_jmax_dc_avg EQU 2 * 1.06 * ( w - 0.003 ) jmax_factor 100 1.45 105 1.00 110
0.70 jmax_life 87600 1.0 L > 3 W >= 0.04
```

where, `W` is the wire width and `L` is the Blech length (or the path length in the current density -rj report). Blech length (Path length/ Short length) is the longest and continuous centerline path from edge to edge among the connected wire shapes on the same metal layer. After the Blech length is determined, resistors on these connected wire shapes on the same metal layer use the same Blech length to get the EM limit.

When certain constraints (“**L>3W<0.4**”) are met, its EM limit can be 2X, 3X, or multiple X of the base EM limit. These multiples are called Benefit Factor (red-highlighted in the above example - “1”, "2").

Voltus supports metal length/width rules above and below vias wherein the dimensions to be checked for the rules are different for the metal above and the metal below. In this case, `Lu/Wu` refers to dimensions of the metal above the via and `Lb/Wb` refers to dimensions of the metal below the via.

The following diagram illustrates a scenario where dimensions are different for the metal above and below a via (Lu > 5, Wu >= 0.8, Lb > 5, and Wb >= 0.4):

The following example illustrates the EM model syntax for different rules above and below:

```
em_jmax_dc_avg PWL 4 0.04 L <= 5
em_jmax_dc_avg PWL 2 0.0324 Lu > 5 Wu >= 0.8 Lb > 5 Wb >= 0.4
em_jmax_dc_avg PWL 2 0.04 Lu <= 5 Lb > 5 Wb >= 0.4
em_jmax_dc_avg PWL 2 0.04 Lu > 5 Wu >= 0.8 Lb <= 5
```

The following length and width dependent EM rules are supported:

- Differentiate between single square cut via and square via-array using the keyword `single`. The following example shows how to use the `single` keyword:

  ```
  em_jmax_dc_avg PWL 0.0276 0.001024 jmax_factor 105 1.434 110 1.000 115
  0.704 120 0.500 125 0.358 130 0.258 140 0.138 150 0.076 current_direction up
  Lb > 4 Wb >= 0.064 current_direction up single
  ```

- Modes to calculate length as centerline or outer edge of a wire segment. The process definition parameter `em_segment_length` is used to determine how to calculate the length of a wire segment. This parameter determines whether the segment length is of the type `perimeter` or `centerline`. The difference between the `perimeter` and `centerline` segment length types is as follows:

  - `perimeter` - specifies that the length calculation is along the wire outer edge boundary. This value is used for 28-nm process nodes.
  - `centerline` - specifies that the length is extended to the edge of the segment along the centerline path. This value is used for 20-nm process nodes.

  The following example shows how to use the `em_segment_length` parameter:

  ```
  em_segment_length centerline
  ```

- Via current direction dependency rule. The syntax of current direction is:

  ```
  current_direction [ up | down | both ]
  ```

- `up` - means the current is from down to top and specifies to use the length and width of the metal below `Lb/Wb` for the VIA Jmax factor
- `down` - means the current is from top to down and specifies to use the length and width of the metal above `Lu/Wu` for VIA Jmax factor
- `both` - This rule is only applicable when current direction is uncertain. This rule is unlikely to be applied since the software calculates the current direction.

Note: The EM rules for 20-nm process node require a VTS-AA license.

Static Rail Analysis Setup and Run

Voltus rail analysis has two accuracy modes, XD and HD.

- `XD` (accelerated definition) – use for early implementation stage IR/EM analysis
- `HD` (high definition) – use for final verification stage IR/EM analysis

The following table illustrates the default PGV selected for static and dynamic analysis based on the accuracy mode (XD/HD):

	XD		HD	
Analysis	Static	Dynamic	Static	Dynamic
PGV	Early	Early	IR/EM	IR
DFM Effects	No	No	No	No
Via Clustering (Above via1)	25x25	25x25	4x4	25x25

The EM power-grid view is enabled for static HD accuracy mode when `-em_models` or `-process_techgen_em_rules true` is specified. By default, the DFM effects and fillers are disabled. To change the PGV selection or enable DFM effects, use the following parameters of the `set_rail_analysis_mode` command:

- `-use_early_view_list` *`filename`*
- `-use_ir_view_list` *`filename`*
- `-use_em_view_list` *`filename`*
- `-enable_manufacturing_effects {true|false}`

This section describes how to set up the static rail analysis mode and then run rail analysis.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- Power-grid libraries (.cl). See Power-Grid Library Generation for details on generating power-grid libraries.

- Power database generated during static power calculation
- Binary current files created during static power calculation

Procedure

To perform static rail analysis, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

```
read_lib -lef $lefs
read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
```

2. Specify the CPF file for power domain information through the command line or through the GUI.

   GUI: Select *File - Load/Commit CPF*.

   Tcl command:

```
read_power_domain -cpf ../design/super_filter.cpf
```

3. Define the rail analysis mode through the command line or through the GUI.

   GUI: Select *Power & Rail - Set Rail Analysis Mode*.

Tcl command:

```
set_rail_analysis_mode \
    -method                     static \
    -accuracy                   xd \
    -analysis_view              AV_wc_on \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } \
    -use_em_view_list           ../data/voltus/em_view.list \
    -enable_rlrp_analysis       true \
    -verbosity true \
    -temperature 125
```

Note:

- *XD* uses LEF based views, and *HD* uses GDS based accurate views.
- The technology library is a mandatory requirement for rail analysis. However, to obtain accurate IR drop simulation, power-grid library should be characterized for standard cells and macro blocks. When specifying the list of PGVs to rail analysis using the `set_rail_analysis_mode` command or the Set Rail Analysis Mode form, you must specify the technology library as the first PGV library in the list. If both technology and standard cell PGV libraries exist, the standard cell PGV is given a higher precedence.
- You need to specify the EM analysis models if analyzing Electromigration. If you do not specify an Electromigration model file, the software will select Electromigration models from the Quantus technology file. You can use the command `set_rail_analysis_mode -em_models` file to override these settings.

4. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

Tcl command:

```
set_pg_nets -net VDD_AO     -voltage 0.9 -threshold 0.85
set_pg_nets -net VDD_ring   -voltage 0.9 -threshold 0.85
set_pg_nets -net VSS        -voltage 0.0 -threshold 0.05
set_rail_analysis_domain -name PD_ring -pwrnets VDD_AO  -gndnets VSS
-threshold 0.10
set_rail_analysis_domain -name PD_AO -pwrnets VDD_AO  -gndnets VSS
-threshold 0.10
```

5. Define the location of the voltage sources for each power/ground net.

   Tcl command:

```
set_power_pads -net VDD_AO -format xy -file
../design/super_filter_VDD_AO.pp
set_power_pads -net VSS -format xy -file
../design/super_filter_VSS.pp
```

6. Specify the location of the binary current files generated through power analysis.

   Tcl command:

```
set_power_data -format current {
staticPowerResults/static_VDD_AO.ptiavg
staticPowerResults/static_VDD_ring.ptiavg
staticPowerResults/static_VSS.ptiavg}
```

7. Run rail analysis through the command line or through the GUI.

   GUI: Select *Power & Rail* - *Run Rail Analysis*.

   Tcl command:

```
#Analyze all domains
analyze_rail -output ./staticRailResults -type domain ALL
#Analyze only one domain
analyze_rail -output ./staticRailResults -type domain PD_AO
#Analyze a power net if the design did not have CPF and power domains
analyze_rail -output ./staticRailResults -type net VDD_AO
```

The Voltus console will display rail analysis statistics, as shown in the following snippet:

```
Rail Analysis completed successfully.
```

```
Finished Rail Analysis at 11:56:53 08/20/2018 (cpu=0:00:20, real=0:00:34, peak
mem=1218.10MB)
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:03:56,
real=0:58:07, mem=1218.10
```

Output

The rail output directory will contain the following information:

Directory Name	Description
<*domainName*>_<temperature>_avg_<*X*>	For each run, an output directory is generated with the default name <*domainName*>_<*temperature*>_avg_<*X*>. <*domainName*> will be replaced with ALL if all domains are to be analyzed (analyze_rail -output *dir* -type domain ALL), or with <*netName*> if a power net is to be analyzed. The output directory contains state directories for each net, such as VDD and VSS. Each state directory contains the following: ■ results – a file that gives a summary of the rail analysis results for various plots, such as tc (tap current), rc (resistor current), and so on. ■ Reports – a folder that contains a list of useful report files that contain all power-grid integrity data. **Note:** You can specify the set_rail_analysis_mode -def_based_hierarchical_reports true parameter to generate separate rail reports (EIV, RLRP, and EFFR) for each DEF instance. Reports will be generated for every net in each DEF hierarchy using the following directory structure: <output_dir>/<state_dir>/Reports/HIER/hier_<HIER_ID>/<domain_name | net_name>.<report_type> ■ Auto-generated GIF files for the rail analysis plots

Viewing Static Rail Analysis Results

The rail analysis results can be debugged in the context of physical layout using interactive rail analysis plots. Voltus allows you to visualize rail analysis results in the design using the Rail Results Viewer.

Required Input Files

- Design data (LEF, netlist, DEF)
- Power database generated during static power calculation
- Rail analysis output directory

Procedure (Rail Results Viewer)

To visualize rail results using rail results viewer, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

   ```
   read_lib -lef $lefs
   read_view_definition ../design/viewDefinition.tcl
   read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
   set_top_module super_filter -ignore_undefined_cell
   ```

2. Load the rail data through the command line or through the GUI.

   GUI:

   a. Select *Power & Rail - Power & Rail Plots*.

The Power & Rail Plots form appears. This form is the main window for displaying the rail analysis results.

b. Click the *DB Setup* button.

The Power & Rail Setup form appears.

c. You can enter the power database or click *Browse* next to the *Power Database* field, and select a location from the `Binary DB File` browser window.

d. You can enter the rail output directory or click *Browse* next to the *Rail Database* field, and select a location from the `Rail State Directory` browser window.

e. In the *Selective Loading* section, you can select the following fields to reduce the amount of the rail analysis data being loaded into the GUI:

- *Nets* - Specify whether all nets (*All*) or selected nets (*Partial*) are to be loaded into the GUI. If you select *Partial*, click the *Select* button to select the required power and ground nets.
  The selected nets will appear in the *Nets* tab of the Power & Rail Layers/Nets form.
- *IV Window* - Load the instance voltage data for the selected EIV window.
- *IV Method Type* - Load the instance voltage data for the specified EIV computing method.

f. Click *OK*.

Tcl command:

```
read_power_rail_results -power_db staticPowerResults/staticPower.db
-rail_directory staticRailResults/ALL_125C_avg_1
```

3. Optional. You can select the nets or layers to be viewed through the command line or through the GUI.

GUI:

a. Click the *Layers/Nets* button on the Power & Rail Plots form.

The Power & Rail Layers/Nets form appears.

b. In the *Layers* tab, select or clear the check boxes adjacent to the specific layers whose visibility you want to turn on or off in the *Visible* column, and to control the processing of individual layers in the *Report* column. Then, use the slider control to control the transparency of the rail analysis output and the design database.

c. In the *Nets* tab, select or clear the check boxes adjacent to the power or ground nets to modify the display of nets. The *Nets* tab allows you to control the display of the switch and always-on nets in the GUI. When you load an always-on net state directory, the *Power Nets* section shows both the always-on and switch nets as a sub-tree under each net, as shown below:

d. Click *OK*.

Tcl command:

```
set_power_rail_layers_nets -disable_visible_layers {Metal7 Via5
Via4 Via3} -disable_report_layers {Metal7 Via5 Via4 Via3} -
disable_visible_sem_layers {} -disable_nets VDD_AO
```

4. In the Power & Rail Plots form, select the *Rail* radio button and select a rail analysis type from the drop-down list.

The selected plot will be overlaid on the design.

Tcl command:

```
#Displays the specified plot
set_power_rail_display -plot ir -filter_max 0.0038 \
                       -filter_min 1.00001e-05

#Displays the specified plot and generates a report
(power_rail_<plot_type>.rpt) in the current working directory
report_power_rail_results -plot  tc

#Clear the display
set_power_rail_display -plot none
```

5. Set the filter options, as required. When a plot type is selected, the filter values automatically get filled in.

   Note: You can select the *Auto Apply for Color Scale* checkbox to automatically redraw the layout after modifying the range values. If you do not select this checkbox, you can

click the *Apply* button to manually redraw the layout.

6. Select the *Result Browser* option to display the Results Browser form. Based on the plot selected, this form allows you to check the resistor or instance information.

7. Select the *Voltage Source* check box to view nets with their voltage sources. If you want to view voltage sources of specific nets, you can clear the check boxes adjacent to the remaining power/ground nets in the *Nets* tab of the Power & Rail Layers/Nets form.

8. Click *Measure* and draw an area in the main window to view the tap current and capacitance of the specified region. The following is an example of the information that will be displayed in the Voltus console:

```
Net: VDD_AO
Region: (20.842, 100.715) (47.0065, 115.105)
Tap Current: 1.16205e-05 A
Tap Current On Unconnected Grids: 0 A
Worst IR: 0.884208 V
Grid Capacitance: 0 F
Intrinsic Capacitance(C1): 0 F
Load Capacitance(C2): 0 F
Decap Required: 0 F
```

9. For the *rlrp* rail plot, if you select an object (instance or resistor) and click the *Trace* button, Voltus highlights the least effective resistance path from the object to the nearest voltage source, and will open the Resistance Path form. This form will list the break down of all resistor segments in the rlrp path. You can select a resistor in the Resistance Path form, single click to select/zoom, and double click to query. Click the *Clear* button to clear the RLRP path. The RLRP path will be retained until you click the *Clear* button. If you select/deselect a resistor on the RLRP path to view its attributes, the RLRP path will continue to be displayed.

   The Resistance Path form has the following fields, in addition to information on the layer/via name, and node coordinates (location for each segment in the least resistance path):

   - *Power Gate* - specifies if a resistor is identified as a power-gate Ron resistor for easy identification.
   - *Resistance (Cumulative)* - displays each resistance and cumulative resistance (in bracket) from a voltage source to the specific resistor.
   - *Voltage Drop (Cumulative)* - displays voltage drop of each resistance (based on the worst IR drop that occurred between two connected nodes), and the cumulative voltage drop (in bracket) from a voltage source to the specific resistor.

- *Auto Zoom Level* - enables you to specify the appropriate zoom level and improve visibility. The zoom levels are “+2, +1, 0, 1, 2”. 0 is selected by default. The prefix “+” means zoom-in whereas “-” means zoom-out.
- *Auto Select* - allows you to highlight any resistor on the list and the resistor will be auto-zoomed.
- *Show Single Path* - allows you to select and view a single RLRP path. For an instance with multiple RLRP paths, the single path display feature allows you to debug the desired path easily.
- *Enable Multi Segments Selection* - allows you to select multiple resistors in the RLRP path. When you select the *Enable Multi Segments Selection* check box and double-click on a row in the Resistance Path form, the Attribute Viewer form appears, as shown below:

The form shows multiple objects/resistors. You can click the *Prev/Next* button, or select a resistor from the drop-down list, to view the details of each resistor.

The rlrp report at the command prompt contains 2 parts: a detailed report showing each rlrp segment information, and a summary report showing information for each layer.

Note: The rail analysis must include RLRP analysis (`set_rail_analysis_mode -enable_rlrp_analysis true`) in order to have the required data for this functionality.

10. Click the *Apply* button on the Power & Rail Plots form.

Selecting *Apply* will accept the changes, and *Reset* will set it to default values. By default, the range does not auto update. You can click the *Reset* button to auto-scale.

11. Experiment with the remaining rail plots, as listed in the following table:

Plot Name	Description
IR Drop - ir	The voltage drop (IRdrop) plot is displayed on all the power-grid segments of the analyzed net. The IRdrop data is saved inside the state directory of the net. The worst case voltage can be displayed by selecting the *Result Browser* option. The segments with high IRdrop can be debugged using the rail analysis plots in conjunction with power plots.
Grid Resistance - res	The grid resistance plot displays the resistance of each power-grid segment. This plot does not consider the power distribution when computing resistance thus highlighting power-grid weakness and resistance hot spots in the design irrespective of activity vector coverage. This analysis plot allows designers to check power-grid robustness during early stages of the power-grid design when worst case vectors that produce high power and IRdrop are unknown. The plot can be used during decap optimization (dynamic) as well because adding decaps along the path of high resistance has no effect on localized dynamic IRdrop.

Plot Name	Description
Instance Resistance of LRP - rlrp	The instance effective resistance plot is similar to grid resistance plot except it is plotted for the instances connected to the power-grid. The resistance for the instance pin is calculated as total resistance along the least resistance path. If an instance has multiple power pins connected to the power-grid, instance based rlrp plot uses the pin with the worst (highest) resistance value to plot the instance based data. Like grid resistance plot, instance based resistance plot is also independent of power vectors. This allows you to identify weakly connected instances in the design during early stages of power planning. It can be used during the decap optimization as well because adding decaps along the path of high resistance has no effect on localized dynamic IRdrop.
Resistor Current - rc	After observing IR drop plots, the next question is how the current flows in the design to create the IR drop plot generated. It is important to examine these plots because they also exhibit behaviors that are not obvious from examining the IR drop plots. These plots show current-flow trends that you might not have expected or current from several power pins reconverging in the middle of the chip to create high current through some wires.

Plot Name	Description
Tap Current - tc	The tap current plot shows the current distribution inside the cell. The power-grid library generation program characterizes the current distribution inside the cell and replaces devices connected to the power-grid with current taps (sink). The relative current distribution inside the cell power-grid view library is based on device width and length. The instance based power calculated during power analysis is fed to rail analysis during IRdrop analysis. The supplied instance power is then distributed inside the cell and can be viewed using tap current plot. If accurate power-grid view is used during rail analysis for the memories, the tap current plot clearly highlights the low power (current) distribution in bit cells region and high power distribution in sense amplifier and decoder circuit of the memory. The tap current plot can be used to debug high IR drop caused by high power density in the region.
Current Density - rj	Once you understand how current flows through your design, you may want to know where the high current densities occur. Use the Current Density analysis type to examine them. Current density reporting does not plot the current density of the segment but it is based on the ratio of wire current density over the current density limit that is allowed by the process - (J/Jmax). You therefore want all wires to have a ratio less than 1.0 and debug or fix the current density violations that have greater that 1.0 value. This ratio is easier to work with because its value is more consistent across designs.
Instance Voltage Drop (Net-based) - ivdn	The IRdrop plot displays the net-based instance voltage drop. ivdn shows the worst instance voltage drop for each net. To see only one net, you can disable all the nets except one.

Plot Name	Description
Instance Voltage Drop (Domain-based) - ivdd	ivdd will show the worst instance voltage drop for each power-ground pair. When at least one power and one ground net is selected, the plot will show the worst instance drop value. For dynamic analysis, you can also select the type of IV method (best, worst, avg, worst avg) and/or the IV window (switching/timing window, whole window) in the *Layers* tab of the Power and Rail Plots - Layers/Nets form.
Power-Gate Current (I/Idsat) - pi	This plot is only applicable to the power-gated design and analysis. The power-gate current (I/Idsat) plot displays the ratio of current through the power-gate and the saturation current (Idsat).This plot can be used to analyze and debug whether the current through power-gates exceeds the saturation current. The saturation current and on-resistance of the power-switches are characterized and stored inside the power-grid library of the power-gate cell. A ratio of greater than 1 means that the current requirement of the power-gated block can not be met with placed power-switch instance in that region. Either the power-switch needs to be upsized or more power-gate instances need to be placed. In addition to the plot, rail analysis creates the text report (pi.report) and binary ECO file (`vs2fe_pg.rpt`) for Innovus Implementation System (Innovus) inside the state directory. The ECO file can be read inside Innovus using "`optPowerSwitch -vsdgInFile`" command to optimize power-gates.
Power-Gate Voltage - pv	The power-gate voltage plot displays the IRdrop across power-switch instances. This plot can be used to analyze and debug the IRdrop inside the power-gated block (e.g. if the IRdrop across power gate is already high, the IRdrop inside the power-gated block will be much higher. In this case, the power-gate placement will have to be refined or more power-gates will have to be added in order to resolve the IRdrop problem).

Plot Name	Description
Resistor Sensitivity Analysis - rs	Resistor sensitivity is calculated by the software as: `S = delta (Worst Voltage)/ delta (Resistance) in V/Ohm` Resistor sensitivity analysis is supported only in static rail analysis and can be enabled using the "`set_rail_analysis_mode -enable_sensitivity_analysis true`" option. By default it is `false`. The resistor sensitivity for power-grid can be viewed by plotting "rs" analysis type in GUI. The sensitivity analysis can be used as guidance for power-grid optimization. Sensitivity is calculated for all power-grid segments and the resistors with higher sensitivity when optimized have better chance to improve overall IRdrop profile of the power-grid. The optimization in this context means reducing overall resistance of the power-grid, e.g. wire widening. Although Voltus stand-alone product does not allow design and netlist changes, it can be used inside the Innovus platform which has wide range of power-grid editing and routing features. Optionally, you can perform what-if analysis (What-If Rail Analysis) inside the Voltus environment which allows you to scale resistance of power-grid segments on-the-fly to analyze and estimate the optimization efforts to improve IRdrop profile of the power-grid. There are two additional rs plots called rs (wire) and rs (wire group) to display resistor sensitivity on a wire or wire-group basis. Instead of displaying sensitivity of an individual resistor in a wire, these plots display highest sensitivity of a collection of resistors that form a wire or a wire group. These plots are only available when you invoke Voltus or ERA analysis in Innovus where the original power planning was done. The power planning commands define the wire or wire-groups that form the power-grid of the design, and Voltus or ERA is used to analyze sensitivity of these wires or wire-groups and optimize the power-grid using a power-planning ECO file.

The following table lists the static analysis plot types and the corresponding controls to enable them:

Type	Control
power gate current I/Idsat (pi) and power gate voltage (pv)	enabled automatically when power-gate(s) are connected to the analyzed net.
voltage drop across package model (vu)	enabled automatically when package data is provided (spice model, in power pin location file, or through default_package_* commands)
instance effective resistance (rlrp)	enabled if `set_rail_analysis_mode -enable_rlrp_analysis true`
voltage across via (vv)	enabled if `set_rail_analysis_mode -enable_voltage_across_vias true`
resistor current density (rj)	enabled if EM models are present with `set_rail_analysis_mode -process_techgen_em_rules true` or `-em_models`
resistor sensitivity (rs)	enabled if `set_rail_analysis_mode -enable_sensitivity_analysis true`

Generating Text Reports

In addition to auto-generated HTML reports, Voltus can incrementally generate text reports for all analysis types, post analysis. These reports can be generated using the `report_power_rail_results` command and report option. You are required to load state directory which has rail analysis results. The power reports can be generated after restoring the power database.

Example TCL Flow

Import design

```
read_design …
```

Load rail analysis state directory

```
read_power_rail_results -rail_directory ALL_25C_avg_1/VSS
```

IRdrop report

```
report_power_rail_results -plot ir -filename VSS.irdrop.report
```

Resistance analysis report

```
report_power_rail_results -plot rlrp -filename VSS.resistance.report
```

Current density (EM) violation report

```
report_power_rail_results -plot rj -filename VSS.currentdensity.report
```

Sensitivity analysis report

```
report_power_rail_results -plot rs -filename VSS.sensitivity.report
```

Instance-based least resistance path report without violation report

```
report_power_rail_results -plot rlrp -filename rlrp.report -limit 0 -rlrp_inst
A/B/C
```

Note: The parameter `-limit 0` effectively turns off the violation report.

There are two ways to generate static power reports in Voltus:

- Using `report_power` after restoring power database
- Using `report_power_rail_results` after restoring power database

Both methods generate report in different format. See Viewing Static Power Analysis Results to view `report_power` options. In general, `report_power` has more detailed options to generate power distribution reports, e.g. for design hierarchy, cell type, instance, clock domain, power domain etc. In contrast, `report_power_rail_results` can only generate global power, activity and frequency reports to highlight worst-case values.

Example TCL flow

```
# Restore power database
```

```
restore_power_database -file power.db
report_power -clock_network all -outfile clock.rpt
report_power -instances inst1
report_power -cell CKBUF
report_power -hierarchy dma_dut -outfile dma_dut.pwr
report_power -pg_net VDD -outfile VDD.pwr
```

OR

```
report_power_rail_results -plot ip -filename total_power.rpt
report_power_rail_results -plot ip_s -filename switching_power.rpt
report_power_rail_results -plot ip_i -filename internal_power.rpt
report_power_rail_results -plot ip_l -filename leakage_power.rpt
report_power_rail_results -plot freq -filename freq.rpt
report_power_rail_results -plot td -filename transition_density.rpt
report_power_rail_results -plot load -filename load_cap.rpt
```

Example IR Drop Report

```
##############################################################
# Generated by: Cadence Voltus IC Power Integrity Solution 09.10-b057_1
# OS: Linux x86_64(Host ID tfopvsv4004)
# Generated on: Fri Aug 28 22:08:39 2009
# Command: read_power_rail_results -rail_directory irdrop_results/...
##############################################################
VALUE (V) LOCATION (nm) LAYER
1.05664587 {662200 1342000 667000 1342000} M1
1.05672038 {657400 1342000 662200 1342000} M1
1.05686557 {652600 1342000 657400 1342000} M1
1.05711162 {667500 1342000 673300 1342000} M1
1.05747104 {647900 1342000 652500 1342000} M1
1.0579921 {536160 2018800 537840 2018800} M1
1.05799508 {538520 2018800 538840 2018800} M1
1.05799508 {538680 2018975 538680 2019325} M1
1.05799568 {538560 2018800 539440 2018800} M1
…..
```

Example Effective Instance Voltage Report

```
##############################################################
# Generated by: Cadence Voltus IC Power Integrity Solution 09.10-b057_1
# OS: Linux x86_64(Host ID tfopvsv4004)
# Generated on: Fri Aug 28 22:08:21 2009
# Command: read_power_rail_results -rail_directory irdrop_results/...
##############################################################
VALUE (V) INSTANCE
1.02425003 clk__L13_I3
1.02438998 dma_dut/dmamaster_dut/internal_memory/RAM4096x32/g977
1.02464008 clk__L13_I4
1.02498007 dma_dut/dmamaster_dut/internal_memory/RAM4096x32/g978
1.02550006 clk__L11_I2
1.02680993 clk__L13_I2
1.02904999 clk__L12_I2
1.02925003 clk__L15_I4
--------
```

Example Current Density Report

```
##############################################################
# Generated by: Cadence Voltus IC Power Integrity Solution 09.10-b057_1
# OS: Linux x86_64(Host ID tfopvsv4004)
# Generated on: Fri Aug 28 22:08:29 2009
# Command: report_power_rail_results -plot rj -filename ./irdrop_result...
##############################################################
VALUE () LOCATION (nm) LAYER
7.08138275 {157570 819200 157570 819200} M7
7.04982233 {157570 2000800 157570 2000800} M7
6.83572531 {157570 1819200 157570 1819200} M7
6.51245785 {157570 2019200 157570 2019200} M7
6.26489735 {157570 1200800 157570 1200800} M7
6.13188887 {157570 2001720 157570 2001720} M7
5.97691965 {157570 2018280 157570 2018280} M7
5.88610029 {157570 1019200 157570 1019200} M7
```

In addition to text reports, you can measure the worst IR drop, total current and capacitance values in the region interactively using "*Draw*" and "*Measure*" buttons in Power & Rail Results form. This feature is also supported through TCL command as follows,

```
read_power_rail_results -rail_directory ALL_25C_dynamic_1/VDD
report_power_rail_results -region {0 0 100 100}
```

```
Region: (38.0275, 1144.72) (53.2615, 1245.33)
Tap Current: 7.33398e-05 A
Worst IR: 0.96455 V
Grid Capacitance: 7.8342e-17 F
Intrinsic Capacitance(C1): 6.97428e-16 F
Load Capacitance(C2): 7.56275e-14 F
Decap Required: 0 F
```

Power-Grid Integrity Reports

Rail Analysis generates a single consolidated report containing all power-grid integrity data in the `.rpt` and `.html` formats (`net_name.pg_integrity.rpt` and `net_name.pg_integrity.html`). This report lists all power-grid integrity related problems, such as instances without voltage-source connectivity in PGDB, disconnected PGVs in PGDB, missing cells, physically disconnected cells, and weakly connected power segments.

The format of the PG integrity report in the DP and XP modes is given below.

Distributed Processing Mode

The consolidated report is generated for each net in the state directory at: `../state_dir/<net>/Reports.`

The report format is:

*** Missing Cell Power-Grid Views** - these leaf-level cells are found to be connected to the power net being analyzed but cell power-grid views for these cells are missing.

```
Cell Name
------------
cell1
cell2
```

*** Physically Disconnected Instances** - all instances listed below are disconnected from the power-grid.

```
Instance Name     | Cell Name   | Root Cause
------------------------------------------------------------
top/A/inst1       | Cell1       | Missing PGV
top/A/inst1       | Cell2       | Physically Disconnected
```

*** Instances with Floating Power Pins** - instances in this list are physically connected to at least one power pin but one or more power pins are disconnected from the power net.

```
Instance Name  |Cell Name |Disconnected Nodes/Total Nodes  |Location (um)
-------------------------------------------------------------------------------
Top/A/inst1    |cell1     | 2/10                           |Pin VDD: X1 Y1 X2 Y2
```

*** Disconnected Wire Segments** - lists wire segments are disconnected on top-level and inside the cell which are not connected to voltage source. View unconnected plot and gif file in state directory for graphical representation.

```
Layer           | Internal to Cell      | Location (um)
-------------------------------------------------------------------
M1              | Yes/No                | X1 Y1 X2 Y2
```

*** Weakly Connected Metal Segments** - lists wire segments that have no connections to the upper metal layers through via.

```
Layer      | Location (um)
----------------------------
M1         | X1 Y1 X2 Y2
```

*** Dropped Voltage Sources** - lists voltage sources that could not be inserted on to the power-grid either due to incorrect location or missing power-grid geometry.

```
Voltage Source Name     | Layer     | XY Location (Micron)
-----------------------------------------------------------------
VDD_vsrc1               | M8        | x1 y1 x2 y2
```

Extensively Parallel (XP) Mode

The consolidated report is generated for each net in the state directory at: `../state_dir/reports/rail/<net>`.

The report format is:

*** Missing Cell Power-Grid Views** - these leaf-level cells are found to be connected to the power net being analyzed but cell power-grid views for these cells are missing.

```
Cell Name
------------
cell1
cell2
```

*** Unconnected Instance Pins** - all instance pins listed below are disconnected from the power-grid.

```
X        |Y        |Pin_Name   |Cell_Name    |Inst_Name
-------------------------------------------------------------
19.430   |11.890   |VDD        |DECAP1       |AO1/FILLER_PD_AO_11
19.430   |14.530   |VDD        |DECAP1       |AO1/FILLER_PD_AO_17
```

*** Unconnected Nodes and Resistors** - all nodes and resistors listed below are not connected to a voltage source.

```
Layer     |#Discon Pwr Grid Res    |#Discon PGV Res   |#Discon Node
-----------------------------------------------------------------------
METAL_1   |182                     |0                  |256
Total     |182                     |0                  |256
```

*** Dropped Voltage Sources** - lists voltage sources that could not be inserted on to the power-grid either due to incorrect location or missing power-grid geometry.

```
Source Name        | Layer     | X              |Y
-------------------------------------------------------------
VDD_vsrc1          | M8        | 19.250          |11.890
```

*** Unmapped Package Terminals** - lists package terminals that are not mapped to the grid locations or the top-level pins.

```
Unmapped Terminals
----------------------------
DIE_VDD_4
DIE_VDD_8
DIE_VSS_3
```

*** Unconnected/Weakly Connected Pin Nodes** - lists pin nodes that are physically connected to instances, but are disconnected from the power net.

```
X        |Y        |Layer      |Pin_Name     |Cell_Name    |Inst_Name
-----------------------------------------------------------------------
19.430   |11.890   |M4         |vddpe1       |cell1        |inst1
```

Layer-Based IR Drop Report

By default, Voltus generates a layer-based IR drop report for each net in the static and dynamic rail analysis flows. The name of the report is <*net_name*>.layerbased_ir.rpt, and it is located within the <*output directory*>/<*state_directory*>/Reports/<*net_name*>/ directory. This report prints the highest and the lowest IR drop on each metal layer. The IR drop report can be used to debug the IR drop contribution from each layer.

The following is the format of the report:

```
Layer          |  IR drop (volt) |  IR drop range (volt) |  Elements
---------------|-----------------|-----------------------|---------
```

Here,

- Layer is the metal layer name.
- IR drop (volt) is the difference between the maximum IR drop and the minimum IR drop.

- IR drop range is the node voltage range, that is, max(node voltage) –> min(node voltage).
- Elements is the number of nodes.

The following is a snippet from the report:

```
Layer          |  IR drop (volt) |  IR drop range (volt) |  Elements
---------------|-----------------|-----------------------|---------
METAL_6        |  0.215          |  0.899   -> 0.685     |  57
METAL_5        |  0.215          |  0.9     -> 0.685     |  67
METAL_4        |  0.188          |  0.872   -> 0.684     |  284
METAL_3        |  0.183          |  0.866   -> 0.683     |  343
METAL_2        |  0.182          |  0.865   -> 0.682     |  321
METAL_1        |  0.325          |  0.863   -> 0.538     |  5704
```

HTML Reports

HTML reports are automatically generated and can be viewed by selecting the *Report* menu item from the *Tools* menu. Select *File* - *Open File* to load an html file. The main HTML file that includes links to other reports will be in a directory call *Reports* in the state directory (i.e. *.../PDcore_25C_avg_1/Reports/dma_mac.main.html*).

Once the main html file is loaded, you can scroll down through the various reports. Some of these have additional links that will bring up other sub-reports, as shown below:

In the following form, you can see there is an IR Drop Analysis Report, which has a link to a Layer Based IR Drop Report:

Total Resistor Elements	116064
Top-level Resistor Elements	116064
Cell Library Resistor Elements	0
Total Number of Current Taps	43048

IR DROP ANALYSIS

Layer based IR Drop Report	METAL_1
Minimum, Average, Maximum IR drop	0.902V 1.138V 1.200V
Dynamic Worst Case Interval	62 (3.100ns)
Peak Dynamic Current Specified	0.377A
Peak Dynamic Current Loaded	0.507A
Number of Violations	0

EFFECTIVE RESISTANCE ANALYSIS

Minimum, Average, Maximum	0.720 5.572

This link will display the layer-based IR drop report, as shown below:

Layer	IR drop (volt)	IR drop range (volt)	Elements
METAL_6	0.0896	1.2 -> 1.11	1099
METAL_5	0.0896	1.2 -> 1.11	2382
METAL_4	0.28	1.18 -> 0.904	10311
METAL_3	0.282	1.18 -> 0.903	9808
METAL_2	0.282	1.18 -> 0.903	9727
METAL_1	0.282	1.18 -> 0.902	71006

Constant Current Source Based IR Analysis

Voltus allows you to perform constant current source based IR drop analysis, where power pads are connected with current sources instead of voltage sources. Using this flow, you can perform accurate analysis in design scenarios where voltage regulators are used to ensure constant current supply at power pads. In this mode, there could be higher voltages for some instances, and the hotspot would impact a larger portion of the design.

To run this flow, you must specify:

- a power pad file in the `xy` format (`set_power_pads -format xy`) to define all the pad locations (voltage/current source locations).
- a power pad file in the `xyiv` format using the `set_power_pads -format` parameter. The `xyiv` format indicates whether the voltage sources specified in the xy power pad location file are current source or reference voltage source. There must be at least one reference voltage source defined in the xyiv file. The format is the same as the xy file, except the additional first and last columns to indicate the current/voltage source and current/voltage value. The name specified in the xyiv file must match the xy file.

  The format of the xyiv file is:

  ```
  <Notation - I/V> <Current/Voltage Source Name> <X Y Coordinates in micron>
  <Layer Name> <Value (I in milliampere and V in volts)>
  ```

  The following is an example of power pad file:

  ```
  I ISR1 12.5 0 M3 100
  V VSRC1 16.401 18.432 M1 0.8
  ```

  where, I is the current source and V is the reference voltage

- the `-enable_ccs_analysis` parameter of the `set_rail_analysis_mode` command to enable constant current source analysis. The default value is `false`.

The following is a sample Tcl script to run this flow:

```
read_lib -lef $lefs
read_verilog test.v.gz
set_top_module test
read_def test.def
read_sdc test.sdc

set_dc_sources VDD -power -voltage 0.825
set_default_switching_activity -input_activity 0.2
set_power_analysis_mode -method static  -power_grid_library { pgv/../stdcells.cl }
-write_static_currents true
```

```
report_power -outfile power.rpt
set_power_data -format current POWER/static_VDD.ptiavg
set_pg_nets -net VDD -voltage 0.825 -threshold 0.80
set_rail_analysis_domain -name PD -pwrnets { VDD } -gndnets VSS
set_power_pads -net VDD -format xy -file pad.vdd
set_power_pads -net VDD -format xyiv -file pad.vdd.ccs
set_rail_analysis_mode -method static -accuracy hd -power_grid_library {
pgv/../techonly.cl pgv/../stdcells.cl } -enable_ccs_analysis true
analyze_rail -type net -output CCS_IRDROP VDD
```

Hotspot Debugger

Hotspot Debugger enables you to identify and fix the hotspots in a design. Hotspot Debugger can automatically identify the region with worst IRdrops and EM violations, and provide necessary information to determine and fix the root cause. Using this feature, you can identify whether the hotspot is caused by one or more of the following:

- Data error
- Weak power-grids or missing vias
- High average power density, simultaneous switching
- Low decap density
- Lack of power-gates
- Lack of power bumps
- Package effects

Hotspot Debugger will identify the top violating instances in each region and report the following information:

- Hotspot Region Coordinate
- Timing Library Analysis
- Power-Grid View Analysis
- Power-Grid Integrity Analysis
- IR-Drop Composition
- Root Cause Analysis

Use the `debug_irdrop` command to identify hotspots. You must specify this command after running the `analyze_rail` command. For more information on the command and Hot-Spot Debugger report, refer to `debug_irdrop` in *Voltus Text Command Reference*.

Note: The Hotspot Debugger feature requires a VTS-XL license.

12

Dynamic Power and IRDrop Analysis

- Overview on page 259
- Dynamic Power Analysis on page 260
  - Vectorless Methodology on page 261
    - Probability-Based Vectorless Methodology on page 261
    - State-Propagation-Based Vectorless Methodology on page 262
  - Vector-Driven Methodology on page 265
    - Vector Profiling on page 266
    - Combining Vector Profiling and Power Analysis on page 268
    - Power Density Aware Vector Profiling on page 269
  - Dynamic Vectorless Power Analysis Flow on page 271
  - Viewing Dynamic Current Waveforms on page 276
  - Dynamic Vector-based Power Analysis Flow on page 284
  - Viewing Profiling Database on page 288
  - Additional Tcl Command Examples on page 291
  - Scan Mode Analysis on page 297
  - Generating User-defined Mode Based Current Waveform with Automatic PWL Scaling on page 301
- Dynamic Rail Analysis Overview on page 303
  - Setting up and running Dynamic Rail Analysis on page 308
  - Run Dynamic Power Analysis and Rail Analysis in Parallel on page 311
  - Dynamic Rail Analysis Plots, Movies and Waveforms on page 313

- Dynamic Rail Analysis for On-Chip Voltage Regulators on page 316
- Using Characterized Current PGV for Dynamic Power and Rail Analysis on page 319
- Dynamic Mixed Mode (Vector-based and Vectorless) Analysis on page 321
- Noise Margin Calculation on page 324

Overview

Voltus IC Power Integrity Solution offers dynamic Power and Rail Analysis on power-grid network. Some of the features include,

- Support for RDL and full-chip GDS for power-grid extraction
- XD and HD accuracy modes for analysis
- Various interactive power-grid analysis plots like IRdrop, current across resistors, device tap currents, IRdrop across package, instance based voltage, grid capacitance, etc.
- CPF Support
  - Power domain definition
  - Power domain operating conditions
  - Off power-switch nets
- Package analysis and die-model generation to take into account the on-chip dynamic switching effects during package design
- Dynamic IRdrop analysis on power-switch design during
  - Steady-state on and off mode
  - Impact of rush current on global power-grid during power-up mode
  - Power-switch ECO for Innovus (optPowerSwitch)
- Dynamic IRdrop waveform with SimVision waveform viewer interface
- Dynamic IRdrop movies
- Dynamic IRdrop waveform for critical path analysis in Tempus Timing Signoff
- Violation browser
- Interactive query
- HTML reports
- GUI and TCL command interface

Voltus offers sign-off quality dynamic power and IRdrop analysis.

Dynamic Power Analysis

Voltus IC Power Integrity Solution requires the use of dynamic power-consumption data for each instance and is output by the power calculator when you perform dynamic power-grid analysis. The current is required to calculate the IR drop. Dynamic, or transient, analysis at the gate level is a way to detect power integrity problems, determine the optimal number and placement of decoupling capacitors to minimize power leakage, provide insight into the effect of simultaneously switching outputs, and assess the impact of resistance, capacitance, and inductance due to packaging, bond wire, and C4 bumps on transient IR drop.

Dynamic power-consumption calculation is a method of analyzing a circuit to obtain operating currents and voltages. It is time-based, analyzing the circuit netlist over a specified period, such as a clock cycle.

The power calculator uses two methods to calculate the dynamic power consumption:

- The vector-driven approach uses the full VCD output of a logic simulator to identify the instances that are switching and when they switch. This method is described in detail in Vector-Driven Methodology.
- The vectorless approach uses timing arrival window information from a static timing analysis tool to determine when instances switch. A timing arrival window describes when a signal can change within a clock cycle. An additional algorithm then determines the instances that switch. This method is called the vectorless, timing-windows-based, or pseudo-dynamic approach. This method is described in detail in Vectorless Methodology.

The following table summarizes where the power calculator obtains signal switching information for both vector-driven and vectorless power-consumption calculation.

Table 9-1 Where the Power Calculator Obtains Signal Switching Information

	Vector-Driven Power-Consumption Calculation	Vectorless Power-Consumption Calculation
What signals switch?	VCD file	User specified activity and activity propagation
When do the signals switch?	VCD file	TWF file
How do the signals switch?	`.lib` + characterization	`.lib` + characterization

Vectorless Methodology

The vectorless methodology can be classified into the following two types:

- **Probability-Based** - Annotates switching probability to all the cells in the datapath.
- **State-Propagation-Based** - Annotates logic events (rise/fall) to the cells on the data path by propagating the rise/fall logic events from the flip-flops' outputs to the downstream logic cones.

The following diagram illustrates probability and state propagation based vectorless methodology:

Probability-Based Vectorless Methodology

In the vectorless, probability-based approach to dynamic power-consumption calculation, the power calculator generates a virtual worst-case power vector using design information and activity and switching constraints that you supply. The challenge posed by vectorless methods of dynamic power-consumption calculation is to obtain information on what instances to consider for switching and when to schedule them to produce a worst-case vector without using actual simulation. The power calculator's vectorless approach creates a realistic, but not too pessimistic, dynamic profile for the design so that you can analyze worst-case dynamic IRdrop in the design.

To determine the time of switching, the power calculator uses the timing database or timing windows file (TWF) generated by a static timing analysis program like Tempus Timing Signoff. In addition to clock domains, slews, slack and constant, it also includes arrival times for each instance pin. The window of arrival times is not an exact time of transition but it is a union of several arrival times combined into one clock cycle, possibly through different combinational

paths, at different clock periods. The power calculator uses Monte Carlo algorithm to schedule (instance's exact switching time) inside this timing window.

To determine how often an instance is switching, the power calculator uses activity propagation along with any user specified activity. The known activity data on primary inputs, sequential output, and clock gates further helps generation of accurate and realistic worst-case dynamic profile. See Activity propagation through clock network and clock gates to see how having some partial VCD information can help with power calculation accuracy.

Once the activity and time of switching is determined, the program uses output load and slew to create a weighted average current profile for the instance based on power arcs defined in .lib for the cell. The calculated dynamic currents are saved for all power nets associated with the instance. These current waveforms can be viewed using SimVision waveform viewer interface. The calculated current waveforms are then fed to dynamic rail analysis engine to perform dynamic IRdrop analysis.

Using a TWF file is less accurate than using a VCD file because it contains ranges of information rather than precise arrival times, however, constructing a worst-case dynamic profile offers great improvement in run time over a vector-based solution which is not scaleable for full-chip. Using a TWF file is also faster than using a VCD file, because it requires less preparation. With a VCD file, you must understand how to prepare stimulus to prepare worst case vector that could produce worst-case power and IRdrop.

State-Propagation-Based Vectorless Methodology

The vectorless, state-propagation-based approach can be done using the following two methods:

- **User-defined activity** - the software will schedule the clock sources, primary inputs, sequential outputs, and macro outputs based on the user-defined activity settings, along with the cells in the fanout cone of such instances. This flow requires the following optional inputs: flip-flop activity, clock network activity (clock gating factor), primary input activity, and macro activity.

  Each clock cycle simulates a different set of cells to switch based on the following configurable parameters:

  - clock domain frequency: Prioritizes scheduling cells associated with faster clocks
  - fanout: Prioritizes scheduling cells with higher fanout
  - user-defined activity: Prioritizes scheduling cells with higher activity

This flow is enabled by the following command:
`set_power_analysis_mode -method dynamic_vectorless -enable_state_propagation true`

The state-propagation-based dynamic analysis generates a switching scenario file (`voltus_power.stateprop.switchsrc`) that contains the switching information of flip-flops.

```
The following is a snippet of the report:
0.0973 DFF11/DFF Q r
0.0987 DFF31/DFF Q f
0.1064 DFF11/DFF QN f
0.1067 DFF31/DFF QN r
....
```

This report contains the following columns:

- first column: time values when the flip-flops toggle
- second column: flip-flop name
- third column: output pin of the flip-flop
- fourth column: present logic of the pin, that is, r indicates "rise" and f indicates "fall"

For every flip-flop, toggles at both Q and QN outputs are reported in this file. For information on the TCL commands for this flow, see State-Propagation-Based Vectorless Flow with User-Defined Activity.

- **User-defined power target** - the software will utilize the user-defined power targets for full-chip power analysis in the XP mode. You can specify the power targets for full-chip, physical block, logical block, and power/ground net. When these targets are specified, Voltus will automatically scale the toggle rates of clock gate ratio, input, macro, and sequential activities to meet the specified power targets. The tool will run multiple iterations of state propagation with a different activity value (pre-defined upper and lower bounds) for each iteration till the power targets are achieved.

  This flow is enabled by the following command:
  `set_power_analysis_mode -method dynamic_vectorless -enable_state_propagation true -enable_power_target_flow true`In addition, the flow has the following steps:

  - Specify the power target values using the `set_power` *`value`* `[-instance/-cell` *`blockname`*`] [-pg_net` *`netname`*`]` command.
  - Specify whether to keep the activities fixed or automatically adjust the activity values for each iteration. This can be specified using the following parameters:
    - `-keep_clock_gate_ratio_in_iterations true | false`

- `-adjust_input_activity_in_iterations true|false`
- `-adjust_macro_activity_in_iterations true|false`

When values are set to fixed for either CGO, input or macro activity, the user-defined values for these parameters (specified with the `set_default_switching_activity` command) will be honored. For information on the TCL commands for this flow, see State-Propagation-Based Vectorless Flow with User-Defined Power Targets.

Vector-Driven Methodology

The VCD file is the output from a gate-level simulation of many vectors and contains all the net switching activity information for the design. The VCD simulation approach to dynamic power-consumption calculation uses the same VCD file as that used for static power-consumption calculation, but unlike the static VCD approach, it uses the information on the switching instances and when they switch. Using output load and slew, it creates current profile for the instance based on power arcs defined in .lib for the cell. The calculated dynamic currents are saved for all power nets associated with the instance. These current waveforms can be viewed using SimVision waveform viewer interface. The accuracy of the current waveform depends on the power tables in the liberty (.lib) file.

Generally the VCD file can be very large in terms of total simulation duration. While static power calculation can process such full VCD file and derive average activity to calculate static power, running dynamic simulation on full VCD is unrealistic due to performance and resource limitation. Hence it is recommended that you identify the power hungry cycles or cycles with high switching activity up-front to perform dynamic power calculation. For information on the VCD profiling feature, see Vector Profiling.

It is recommended that VCD window (start and stop time) for dynamic power calculation does not exceed more than five cycles of dominant clock in the design.

The vector-driven approach for dynamic power-consumption calculation offers several advantages:

- The data that it yields is exact for the given vectors. For a given vector, you know exactly what is switching and when.
- You can use it to validate the vectorless, timing-windows based simulation.
- It is a well-understood methodology, if power-hungry cycles are known and is therefore easy to develop.

Vector Profiling

The Vector profiling feature enables you to identify windows with maximum activity and power which can then be used to drive dynamic vector-based power analysis. Voltus supports event-based vector profiling, wherein the vector profiler computes power profile of every event on each net. Event-based peak power profiling enables you to accurately capture vectors that could produce peak power using a very small resolution. The recommended step size for event-based profiling is two times the fastest clock, and 100 steps are created by default.

The following example illustrates how internal power is calculated in event-based power profiling:

In the 1st period, pin A has a fall transition and pin B is stable at 2.5ns, pin B has a fall transition and pin A is stable at 5ns, and pin A has a rise transition at 7.5ns. Vector profiling calculates power as and when an event happens. Here,

- For the 1st period, power will be calculated as summation of events happening at 2.5, 5.0, 7.5, and 10ns.
- For the 2nd period, power will be calculated as summation of events happening at 12.5 and 17.5ns.

In this example, internal power using event-based power profiling is calculated as:

```
Internal Power (1st Period) = (FallEnergy for !A&B + FallEnergy for !A&!B +
RiseEnergy for A&!B)/10ns
```

The Rise and Fall Energy values are derived from the Liberty file.

To perform Vector profiling, you must provide basic data such as the verilog netlist, SDC file, Liberty files (.libs), SPEF file, DEF file, and the vector files (VCD, TCF, FSDB, SAIF, PHY, or SHM). Use the `read_activity_file -format` command to specify the file format.

Tip

By default, the `report_power -outfile` command parameter generates a summarized vector profile report. You can use the `-time_based_report` parameter to generate a detailed vector profile report with profiling results for all steps.

The Power Analysis engine generates a text based profiling report and profiling database, which can be viewed using the SimVision interface, based on the options defined with the `report_power` and `set_power_analysis_mode` commands.

The Vector profiling flow can be described as follows:

Combining Vector Profiling and Power Analysis

You can combine Vector Profiling and Power Analysis in a single run by first running Vector Profiling, and then running Static and Dynamic power analysis on the worst case window identified by Vector Profiler.

This is done by storing the worst case window from the last profiler run as variables `$worst_power_window_start` and `$worst_power_window_end`. You can specify these variables to set the `-start` and `-end` parameters of the `read_activity_file` command. When specified, the software takes the worst case start and end times from the vector profiling run.

The use model of this flow is:

```
set_power_analysis_mode -method vector_profile
read_activity_file ...
report_power...

set_power_analysis_mode -reset
set_power_analysis_mode -method dynamic_vectorbased
read_activity_file -reset
read_activity_file ...-start ${worst_power_window_start} -end
${worst_power_window_end}
report_power
```

You can perform dynamic power analysis and vector profiling for multiple windows by using the `set_power_analysis_mode -worst_window_count` parameter. This parameter allows you to specify the number of worst windows from the last vector profile run to be used for dynamic power analysis. If this parameter is not specified, power analysis is performed on the single worst window.

Power Density Aware Vector Profiling

The Vector Profiler flow identifies worst activity and worst average power cycles in the VCD/FSDB that are then used for detailed dynamic power and IRdrop analysis. However, if you need to take into account core regions of high power density that could produce high dynamic switching current and IRdrop, use tile dimension for power density calculation. In this flow, you can specify the number of physical tiles, and the tool will:

- Compute power density in each tile at every time step
- Track which tiles have worst power density during VCD profile
- Sort vectors based on worst-case power density

To run this flow, you must specify the following parameters:

```
set_power_analysis_mode -vector_profile_mode {power_density}
report_power -power_density_tiles
```

The use model of this flow is:

```
read_activity_file -format VCD -block {} -scope {top} -start_time_shift {} -start
{} -end {} ieee.vcd
set_power_analysis_mode ... -vector_profile_mode {power_density} -worst_step_size
20ns
report_power -outfile tiling -nworst 5 -time_based_report -pg_net {VDD VSS} -
power_density_tiles 25
```

The format of the vector profile report is:

```
# Header
# Tile Description
Tile 1 X1 Y1 X2 Y2
Tile 2 X1 Y1 X2 Y2
Tile n X1 Y1 X2 Y2
# Worst-Case Vectors <sorted based on max power density tile>
Tile No.    Power Density (mW/tile area)  # Vector Time Stamp
# Worst Tile Profile For Each Step
Tile No.    PPower Density (mW/tile area um^2)    time stamp
# Worst Time Stamp Profile For Each Tile
Tile No.    PPower Density (mW/tile area um^2)    time stamp
# Detailed Tile Profile For Each Step
Step(ns)    Instance    Power(mW)    Internal Power(mW)    Switching Power(mW)
Leakage Power(mW)    Activity
```

The following is a sample report:

```
*-------------------------------------------------------------------------------
--------
*    Voltus IC Power Integrity Solution 13.20 (64bit) 07/03/2013 01:28 (Linux 2.6)
*     Date & Time:    2013-Jul-10 14:56:39 (2013-Jul-10 21:56:39 GMT)
*-------------------------------------------------------------------------------
--------
```

```
Tile 0   0      0  213.192  222.264
Tile 1  213.192 0  426.384  222.264
Tile 2  426.384 0  639.576  222.264
...............
...............
Worst-Case Vectors
Tile No.    Power Density (mW/tile area um^2)      time stamp ns
0            7.46e-10           500.000
164          7.46e-10           500.000
100             0               500.000
...............
...............
Worst Tile Profile For Each Step
Tile No.    Power Density  (mW/tile area)      time stamp
0            7.46e-10           500.000
Worst Time Stamp Profile For Each Tile
Tile No.    Power Density   (mW/tile area)      time stamp
0           7.46e-10           500.000
1             0                500.000
...............
.............

Detailed Tile Profile For Each Step
Step(ns)          Instance    Power(mW)    Internal Power(mW)    Switching
Power(mW)     Leakage Power(mW)    Activity
---------------------------------------------------------------------------
0.000 to 1.000     Top         2.408             2.384              0.02222
              0.001835            7
...
```

Dynamic Vectorless Power Analysis Flow

This section describes how to set up the dynamic vectorless power analysis mode and then run power analysis.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- Timing/Power Libraries (.lib)
- SDC or TWF
- SPEF

Important

Static Power Analysis should be run first and all static problems should be resolved prior to running dynamic analysis.

Procedure

To perform dynamic vectorless power analysis, complete the following steps:

1. Load the design data through the command line or through the GUI.

   GUI: Select *File - Read Design*.

   Tcl command:

```
read_lib -lef $lefs

read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz
```

2. Specify the CPF file for power domain information through the command line or through the GUI.

   GUI: Select *File - Load/Commit CPF*.

Tcl command:

```
read_power_domain -cpf ../design/super_filter.cpf
```

3. Specify the SPEF file for signal net parasitic information through the command line or through the GUI.

   GUI: Select *File - Read SPEF*.

   Tcl command:

```
read_spef -rc_corner  RC_wc_125 -decoupled\
../design/postRouteOpt_RC_wc_125.spef.gz
```

4. Specify the sub-directory that will contain the power results.

   Tcl command:

```
set_power_output_dir            dynVecLessPowerResults
```

5. Define the power analysis mode through the command line or through the GUI.

   GUI: Select *Power Rail - Set Power Analysis Mode*.

   Tcl command:

```
set_power_analysis_mode \
    -reset
set_power_analysis_mode \
    -analysis_view              AV_wc_on \
    -disable_static             false \
    -write_static_currents      true \
    -binary_db_name             dynvectorlessPower.db \
    -create_binary_db           true \
    -method                     dynamic_vectorless
```

   Note: This Tcl command is the same as for the static power analysis, except that `-method` must be `dynamic_vectorless`.

6. Define the power of an instance or cell. For dynamic analysis, you can assign a piece-wise linear waveform to an instance.

Tcl command:

```
set_power -reset
set_power -pg_net VDD_AO -pwl -instance instA { 0ns 0mA 0.075ns 0mA
0.175ns 45mA 0.225ns 15mA 0.425ns  0mA } -sticky
```

- You can also specify power-grid views using the `set_power_analysis_mode -power_grid_library` parameter, instead of specifying .lib or defining a power value with `set_power`.
- Time and current units must always be in ns and mA.
- `-sticky` will apply the waveform as specified in the simulation. If `-sticky` is not specified for `-pwl`, the waveform will be applied each time the output toggles.
- If you are using a VCD file and start the simulation at some point other than `t=0`, the simulator will consider that start time `t=0`.

7. Optional. Define the simulation period and resolution.

Tcl command:

```
set_dynamic_power_simulation -reset
set_dynamic_power_simulation -resolution 50ps
```

Note: If the resolution is high, the current peaks may be truncated, affecting the accuracy of the analysis. Alternatively, a small resolution may lead to increase in runtime.

8. Run power analysis through the command line or through the GUI.

GUI:

- Select *Power Rail* - *Run Power Analysis* to access the *Run Power Analysis* form that can be used to specify cell/instance power (`set_power`), toggle rates (`set_switching_activity` and `set_default_switching_activity`), power result directory (`set_power_output_dir`), and then run power analysis (`report_power`).
- Select *Power Rail* - *Textual Reports* - *Power Analysis* to access the *Report Power* form that can be used to generate incremental power reports.

Tcl command:

```
report_power -outfile dyn.rpt
#Generate additional report formats
report_power -instances {u0}    -outfile u0.rpt
report_power -instances {*}     -outfile all.rpt
report_power -instances {ring}  -outfile ring.rpt
```

The Voltus console will display power statistics, as shown in the following snippet:

```
Total Power
-----------------------------------------------------------------------------
Total Internal Power:        2.30917507    64.6507%
Total Switching Power:       1.03379946    28.9436%
Total Leakage Power:         0.22879451     6.4056%
Total Power:                 3.57176907
-----------------------------------------------------------------------------
Ended Static Power Report Generation: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=750.07MB/1877.45MB/750.07MB)

Begin Creating Binary Database
Ended Creating Binary Database: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=750.34MB/1877.45MB/750.34MB)
```

9. View the peak current for a particular instance, at the level of hierarchy or the whole design. For more information, refer to Viewing Dynamic Current Waveforms.

Output

The power output directory will contain the following information:

File Name	Description
<binary_database>.db	The name of the binary database is specified using the `set_power_analysis_mode -binary_db_name` parameter.. The database can be loaded in the Voltus GUI for viewing and analyzing different types of power plots, and GUI interactive debugging. It cross references the instance names and locations with the power results.

File Name	Description
`<power_report>.rpt`	The name of the power report is specified using the `report_power -outfile` parameter. The power report summarizes power consumption, by groups (sequential cells, macros, clocks), power net, and clock domain.
`static_<net_name>.ptiavg`	This is the static current file for each net. The `set_power_analysis_mode -write_static_currents` parameterwill generate the current files that are required for rail analysis.
`dynamic_<net_name>.ptiavg`	This is the dynamic current file for each net. This file allows you to view the peak current for a particular instance, at the level of hierarchy or the whole design.

Viewing Dynamic Current Waveforms

Voltus IC Power Integrity Solution has embedded SimVision waveform viewer to analyze the generated dynamic current waveforms. You can view the peak current for a particular instance, at the level of hierarchy, or the whole design.

Required Input Files

- Waveform files
- Power database generated during power calculation

Procedure

To view dynamic power (current) waveforms in GUI, the following steps are needed:

1. Select *Power Rail* - *Dynamic Results* - *Waveforms*.

The following form appears:

2. Select the *Power (Current) Waveforms* option.
3. To plot an instance current waveform:
   a. In the *Waveform Files* section, click *Add* and select a dynamic current waveform (`dynamic_`*`netname`*`.ptiavg`) file from the *Select a File* browser window.

      You can query the waveforms of all the instances of the selected power net.

   b. Perform one of the following steps to select instances:

❍ Select an instance from the list box, as shown below:

- Filter instances using the filter pattern. To filter instances, enter a pattern (`ref_clk*`) in the *Pattern* field, and click *Instances*. The required instance (`ref_clk__L1_I0`) appears in the list box, as shown below:

You can click the "*X*" button in the *Pattern* field to clear the text in the field.

- Select instances from the Layout GUI, and click the *Select From GUI* button, as shown below:

c. Click the *Plot* button.

The SimVision plot appears.

4. To plot current for the entire design:
   a. In the *Waveform Files* section, click *Add* and select a dynamic current waveform (`dynamic_netname.ptiavg`) file from the *Select a File* browser window.
   b. Select *Total Current* from the *Composite Waveform* drop-down list.
   c. Click the *Plot* button.
5. To plot a composite dynamic current profile for a clock domain:
   a. In the *Waveform Files* section, click *Add* and select a dynamic current waveform (`dynamic_netname.ptiavg`) file from the *Select a File* browser window.
   b. Click the browse button next to the *Power Database* field, and select a database file from the *Open Power DB File* browser window.
   c. Select *Clock* from the *Composite Waveform* drop-down list.
   d. Select a clock net to produce composite dynamic waveform. Optionally, select the "*all*" option to produce total composite dynamic waveform for all clock domains.
   e. Click the *Plot* button.
6. To plot composite dynamic waveform for a hierarchical module:

a. In the *Waveform Files* section, click *Add* and select a dynamic current waveform (`dynamic_`*`netname`*`.ptiavg`) file from the *Select a File* browser window.

b. Select *Hierarchy* from the *Composite Waveform* drop-down list.

c. Specify a hierarchy name to generate its total dynamic current profile in the text field.

d. Click the *Plot* button.

Important

The View Waveforms form supports multi-threading to enable the search of instances in parallel, and thereby improve the search time in large designs. Multi-threading is supported for instance search using both the *Instances* and the *Select From GUI* buttons.For this feature, you must set up multi-CPU usage before accessing the View Waveforms form, as shown below:

```
set_multi_cpu_usage -localCpu number_of_cpus
```

TCL Commands for Viewing Waveforms

```
##TCL command for viewing instance current waveform:
view_dynamic_waveform \
    -waveform_type current \
    -waveform_files dynamic_vss.ptiavg \
    -instance inst1


##TCL command for viewing composite total current waveform is as
follows:
view_dynamic_waveform \
    -waveform_type current \
    -waveform_files dynamic_vss.ptiavg \
    -composite_waveform_type total_current


##TCL Command for viewing composite clock waveform is as follows:
view_dynamic_waveform \
    -waveform_type current \
    -waveform_files dynamic_vss.ptiavg \
    -power_db power.db \
    -composite_waveform_type clock \
    -composite_waveform_name clk


##TCL Command for viewing hierarchical waveform is as follows:
view_dynamic_waveform \
    -waveform_type current \
    -waveform_files dynamic_vss.ptiavg \
    -composite_waveform_type hierarchy \
    -composite_waveform_name ethernet_mac_2
```

Dynamic Vector-based Power Analysis Flow

This section describes how to combine vector profiling and power analysis for the worst activity windows.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- SPEF
- Activity file

Procedure

To perform vector profiling, complete the following steps:

1. Repeat the steps 1-3 given in the Dynamic Vectorless Power Analysis Flow section to load the design data, CPF, and the SPEF file.
2. Specify the sub-directory that will contain the power results.

   Tcl command:

   ```
   set_power_output_dir            dynVectorbasedPowerResults
   ```

3. Specify the activity file.

   Tcl command:

   ```
   set_power_analysis_mode -report_missing_nets true
   read_activity_file -reset
   read_activity_file -format VCD -scope FIR_TB.Unit -start {} -end {}
   -block {} ../design/ncsim.vcd
   ```

   - The `-report_missing_nets` parameter gives a list of nets in the design that are not annotated by VCD. This is useful when pipe-cleaning the flow. The default value is `false`.
   - Voltus can read activity files in the following formats: VCD, TCF, SAIF, FSDB, PHY, and SHM.

4. Run vector profiling by defining the vector profile power analysis mode to identify where the maximum activity is taking place.

   Tcl command:

```
Event-based Vector Profiling
set_power_analysis_mode \
    -reset
set_power_analysis_mode \
    -analysis_view              AV_wc_on \
    -disable_static             false \
    -write_static_currents      true \
    -binary_db_name             dynvectorbasedPower.db \
    -create_binary_db           true \
    -method                     vector_profile \
    -worst_step_size            2ns \
    -write_profiling_db         true \
    -worst_window_count 2
report_power -nworst 5 -outfile eventbased_profiler.rpt -
time_based_report
The following are options required for the different types of profiling:
Activity Profiling
set_power_analysis_mode -vector_profile_mode {activity}
report_power -outfile activity_profiler.rpt
Power Density Profiling
set_power_analysis_mode -vector_profile_mode {power_density}
report_power -outfile tiling -nworst 5 -time_based_report -pg_net
{VDD VSS} -power_density_tiles_row_col { 5 5 }
```

   Note:

   - This Tcl command is the same as for the static power analysis or dynamic vectorless flow, except that `-method` must be `vector_profile`.
   - Use the `-worst_window_count` parameter to perform dynamic power analysis and vector profiling for multiple windows.

5. Optional. Define the simulation period and resolution.

   Tcl command:

```
set_dynamic_power_simulation -reset
set_dynamic_power_simulation -resolution 50ps
```

Note: If the resolution is high, the current peaks may be truncated, affecting the accuracy of the analysis. Alternatively, a small resolution may lead to increase in runtime.

6. Run dynamic power analysis.

   Tcl command:

```
set_power_analysis_mode -reset
set_power_analysis_mode -method dynamic_vectorbased
read_activity_file -reset
read_activity_file -format SHM ADDER.trn -scope adder_tb_e/u1 -start
${worst_power_window_start} -end ${worst_power_window_end}
report_power
```

   Note: The worst start/end activity windows are stored in memory through the Tcl variables `$worst_power_window_start` and `$worst_power_window_end` specified using the `read_activity_file` command.

The Voltus console will display vector profiling statistics, as shown in the following snippet:

```
User did not set the step size. The step size is being configured as 39.9998 ns.
Finished Reading VCD toggles
2018-Oct-03 10:01:46 (2018-Oct-03 04:31:46 GMT)

  Filename (activity)                    : ../design/ncsim.vcd
  Names in file that matched to design   : 3165/4472
  Annotation coverage for this file      : 3165/3170 = 99.8423%

With this vcd command,  83472 value changes and 3.99998e-06 second simulation time
were counted for power consumption calculation.
Begin Creating Binary Database
Ended Creating Binary Database: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=330.03MB/2571.58MB/1042.52MB)
Finished Power Analysis at 10:01:46 10/03/2018 (cpu=0:00:03, real=0:00:09, peak
mem=1042.53MB)
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:01:28,
real=0:18:05, mem=1042.53MB)
```

Output

The power output directory will contain the following information:

File Name	Description
<binary_database>.db	The name of the binary database is specified using the `set_power_analysis_mode -binary_db_name` parameter. The database can be loaded in the Voltus GUI for viewing and analyzing different types of power plots, and GUI interactive debugging. It cross references the instance names and locations with the power results.
<vector_profile_report>.rpt	The name of the vector profile report is specified using the `report_power -outfile` parameter. ■ **Average Vector Profiling**: The report details the average power of all events in the user-specified window. This report includes a summary based on the power components, cell type, power rail, and clock network. The worst window report is captured in a file specified with the `-outfile` parameter, and the report is suffixed by '`.avgpower`'. When `-outfile` is not specified, the `vectroprofile.report.avgpower` report file is created. ■ **Activity Vector Profiling**: The report generates only the worst activity report, and does not generate the average power report. The name of the file is as specified using the `-outfile` parameter. ■ **Power Density Profiling**: The report generates power for each tile of the design, where design can be considered to be physically divided into equal area tiles. By default, a design is divided in 10x10 tiles.
<profiling_database>.trn	The name of the profiling database is specified using the `set_power_analysis_mode -write_profiling_db` parameter. The profiling database is a binary database of the profile data and can be viewed with SimVision. It allows you to visualize the activity of the design.

Viewing Profiling Database

Voltus uses the SimVision waveform viewer to visualize the activity of the design.

Required Input Files

- Binary database of the profile data (.dsn and .trn files)

Procedure

To view the activity of the design in the GUI, the following steps are needed.

1. Select *Power Rail* - *Dynamic Results* - *Waveforms*.

   The *View Waveforms* form appears.

2. Select the *Profiling Histograms* option.

3. In the *Waveform Files* section, click *Add* and select a database file (`file.trn`) from the *Select a File* browser window.

4. Click the *Plot* button.

   The SimVision window appears.

5. In the *Design Browser* part of the window, click on the database folder in the upper section.

The folder content will appear in the lower section, as shown below:

6. Click on a node in the lower section, and the waveform will appear in the waveform window.
7. Select *View->Zoom->Full X* and *Full Y* to see the full waveform, as shown below:

8. You can zoom in and view the activity over time.

TCL Commands for Viewing Profiling Database

```
##TCL command for viewing profiling database:
view_dynamic_waveform -type profile \
-waveform_files {vectorprofile.report.trn}
```

Additional Tcl Command Examples

Vector-Based Flow - Multiple VCD/FSDB Files

The TCL commands for dynamic vector-based power analysis are shown below.

```
set_power_analysis_mode -method dynamic_vectorbased -corner max \
    -power_grid_library {fast_allcells.cl} -disable_static true
```

```
read_activity_file -format vcd -start 10ns -end 20ns -scope top/block design.vcd
set_switching_activity -activity 0.1
set_power -cell BUFX2MTL -pwl {0 0 1 0 2 1 3 0}
set_power_output_dir dynamic_vectorbased_max
report_power
```

You can specify multiple VCD/FSDB files with different windows for the vector-based dynamic analysis/vector profiling flow and write out a single dynamic current file/profiling report. The TCL commands for dynamic vectorbased power analysis using multiple VCD files are shown below:

```
set_power_analysis_mode \
-start_time_alignment true -disable_static false \
-power_grid_library " $testcaseRoot/library_pv.cl"\
-method vector_profile -x_transition_factor 1 -z_transition_factor 1 \
-vector_profile_mode {activity} -worst_step_size 2ns
```

By default, the software aligns all signals at t=0 (or min start time). You can disable the start time alignment by setting `set_power_analysis_mode -start_time_alignment` to `false`.

```
read_twf $testcaseRoot/pt.tw_ckt2
read_activity_file -format VCD -scope top -start 3 -end 10 top.vcd
read_activity_file -format VCD -block block1 -start 5 -end 12 block1.vcd
read_activity_file -format VCD -block block2 -start 4 -end 15 block2.vcd
report_power -outfile activity.rpt
```

#Simulation period is max(endtime - starttime), that is, 11ns.

#Start times for all VCDs will be aligned at t=0

You can specify the `read_activity_file` command multiple times to read multiple VCD files. When you specify the `-start_time_shift` parameter of this command, the software first aligns all the signals of the specified activity files at time zero, and then shifts the start time of the activity file specified with the `-start_time_shift` parameter. A sample Tcl command to illustrate this is given below:

```
set_power_analysis_mode -method dynamic_vectorbased
read_activity_file -format VCD -scope top -start 3 -end 10 top.vcd
read_activity_file -format VCD -block block1 -start 5 -end 12 block1.vcd -
start_time_shift 2ns
read_activity_file -format VCD -block block2 -start 4 -end 15 block2.vcd
report_power
```

#Simulation period is max(endtime - starttime), that is, 11ns.

#Start times for the top + block2 VCD will be aligned at t=0 whereas for block1.vcd it will start at 2ns.

Probability-Based Vectorless Flow

The TCL commands for dynamic probability-based vectorless power analysis are shown below.

```
set_power_analysis_mode -method dynamic_vectorless -corner max \
                        -power_grid_library {fast_allcells.cl}
set_default_switching_activity -input_activity 0.2 -seq_activity 0.1 \
                               -hier top/block/module1
read_activity_file -format vcd -start 10ns -end 20ns -scope top/block design.vcd
set_switching_activity -activity 0.1
set_power -cell BUFX2MTL -pwl {0 0 1 0 2 1 3 0}
set_dc_sources {VDD VDD1} -power -voltage 1.2
set_power_output_dir dynamic_vectorless_max
report_power
```

You can use the `set_dc_sources` command to specify the operating voltages of the power/ground nets. This command is used to specify a global voltage or voltages for specific power nets in the design.

To enable multi-threading in dynamic vectorless flow

```
set_multi_cpu_usage -localCpu 4
set_power_analysis_mode -method dynamic_vectorless -corner max \
 -power_grid_library {fast_allcells.cl} -disable_static true
 set_default_switching_activity -input_activity 0.2 -seq_activity 0.1 \
                               -hier top/block/module1
read_activity_file -format vcd -start 10ns -end 65ns -scope top test.vcd
set_switching_activity -activity 0.1
set_power -cell BUFX2MTL -pwl {0 0 1 0 2 1 3 0}
set_power_output_dir dynamic_vectorless_max
report_power
```

State-Propagation-Based Vectorless Flow with User-Defined Activity

The TCL commands for dynamic state-propagation-based vectorless power analysis are shown below.

```
set_multi_cpu_usage -localCpu 4
set_default_switching_activity -input_activity 0.2 -seq_activity 0.1 \
                               -clock_gates_output 2.0
set_power_analysis_mode -method dynamic_vectorless -enable_state_propagation
true -power_grid_library {fast_allcells.cl}
```

```
set_dynamic_power_simulation -period 20ns -resolution 50ps
report_power -output dynamic_power
```

Voltus generates a time varying switching profile report (`voltus_power.togglestats`) in the output directory for both the vector-based and state-propagation-based vectorless dynamic power analysis flows. The report includes the number of output toggles with the corresponding simulation time.

The following is a snippet of the report:

```
Toggle Statistics
*____________________________
*Time(s)     Toggle Count
*____________________________
0.0000e+00    3
2.0000e-11    0
…
```

Following is a sample of a detailed vector profile report:

```
The 10 intervals have the maximum power (mW):
               1.000    to                2.000   0.03452
              47.000    to               48.000    0.0315
              48.000    to               49.000   0.03035
              46.000    to               47.000   0.02716
               5.000    to                6.000   0.02574
              15.000    to               16.000   0.02398
              25.000    to               26.000   0.02398
              35.000    to               36.000   0.02398
              45.000    to               46.000   0.02398
              55.000    to               56.000   0.02398
The 10 intervals have the maximum activity:
              45.000    to               46.000        11
              35.000    to               36.000         9
              46.000    to               47.000         9
               1.000    to                2.000       8.5
              47.000    to               48.000         7
               5.000    to                6.000         6
              15.000    to               16.000         6
              25.000    to               26.000         6
              48.000    to               49.000         4
               2.000    to                3.000       3.5

The 10 intervals having maximum dp/dt (mW):
               1.000    to                2.000   0.03439
              49.000    to               50.000  -0.03022
               5.000    to                6.000   0.02561
```

```
               6.000    to                7.000  -0.02561
              15.000    to               16.000   0.02384
              25.000    to               26.000   0.02384
              35.000    to               36.000   0.02384
              36.000    to               37.000  -0.02384
              45.000    to               46.000   0.02384
              55.000    to               56.000   0.02384
The 10 intervals have the maximum da/dt:
              45.000    to               46.000        11
              35.000    to               36.000         9
              36.000    to               37.000        -9
               1.000    to                2.000       8.5
               5.000    to                6.000         6
               6.000    to                7.000        -6
              15.000    to               16.000         6
              25.000    to               26.000         6
               2.000    to                3.000        -5
              16.000    to               17.000        -4

Step(ns)    Instance Power(mW) Internal   Switching    Leakage        Activity
                               Power(mW)  Power(mW)    Power(mW
-----------------------------------------------------------------------
0.000 to 1.000 Top   0.0001358     0       0           0.0001358       0
1.000 to 2.000 Top   0.03452      0.02459  0.009792    0.0001358       8.5
...
```

State-Propagation-Based Vectorless Flow with User-Defined Power Targets

The TCL commands for state-propagation-based vectorless power analysis with user-defined power targets are shown below.

Example1: Enable the power target flow with fixed CGO

```
set_power_analysis_mode \
...
-method dynamic_vectorless \
-enable_state_propagation true \
-enable_power_target_flow true \
-keep_clock_gate_ratio_in_iterations true
...
```

```
##Set power target at the top-level
set_power 3mW
...
##User-specified CGO
set_default_switching_activity \
-clock_gates_output 0.4
```

In this case, the clock gate output is fixed, while the values for input, macro, and sequential activities will be adjusted to meet the power target of 3mW.

```
target_0: (cg_ratio 0.2 seq_activity 0 input_activity 0 macro_activity 0)
target_0: (cg_ratio 0.2 seq_activity 1 input_activity 1 macro_activity 1)
target_0: (cg_ratio 0.2 seq_activity 0.54 input_activity 0.54 macro_activity 0.54)
Achieved total power 0.0031013 in 2 iterations
```

Example2: Enable the power target flow with a single power target of 3mW

```
set_power_analysis_mode \
...
-method dynamic_vectorless \
-enable_state_propagation true \
-enable_power_target_flow true \
...
##Set power target at the top-level
set_power 3mW
...
```

In this case, the values for input, macro, clock gate output, and sequential activities will be adjusted to meet the power target of 3mW.

```
target_0: (cg_ratio 0 seq_activity 0 input_activity 0 macro_activity 0)
target_0: (cg_ratio 0 seq_activity 1 input_activity 1 macro_activity 1)
target_0: (cg_ratio 0.44 seq_activity 0.44 input_activity 0.44 macro_activity 0.44)
Achieved total power 0.00310119 in 2 iterations
```

Example3: Enable the power target flow with two power targets

```
set_power_analysis_mode \
...
-method dynamic_vectorless \
-enable_state_propagation true \
-enable_power_target_flow true \
...
```

```
##Set 2 power targets (top and physical/logical hierarchy)
set_power 3mW
set_power 1mW -instance ring
...
```

In this case, the values for input, macro, clock gate output, and sequential activities will be adjusted to meet the two power targets.

```
target_0: (cg_ratio 0 seq_activity 0 input_activity 0 macro_activity 0)
target_1: (cg_ratio 0 seq_activity 0 input_activity 0 macro_activity 0)


target_0: (cg_ratio 1 seq_activity 1 input_activity 1 macro_activity 1)
target_1: (cg_ratio 1 seq_activity 1 input_activity 1 macro_activity 1)


target_0: (cg_ratio 0.44 seq_activity 0.44 input_activity 0.44 macro_activity 0.44)
target_1: (cg_ratio 0.79 seq_activity 0.79 input_activity 0.79 macro_activity 0.79)


target_0: (cg_ratio 0.38 seq_activity 0.38 input_activity 0.38 macro_activity 0.38)
target_1: (cg_ratio 0.79 seq_activity 0.79 input_activity 0.79 macro_activity 0.79)


target_0: (cg_ratio 0.35 seq_activity 0.35 input_activity 0.35 macro_activity 0.35)
target_1: (cg_ratio 0.79 seq_activity 0.79 input_activity 0.79 macro_activity 0.79)


Achieved total power 0.00306517 in 4 iterations
```

Scan Mode Analysis

Scan Mode Analysis is a method used in design for testing (DFT) to build scan-chains that are used to detect stuck-at faults in a circuit. In scan-design, the daisy chained scan-flops are connected in series to form a scan-chain which are populated (shifted) with test vectors at scan clock frequency when the `scan_enable` pin is high. The daisy-chained scan flops are bigger flops with a multiplexer.The following diagram illustrates scan chain configuration:

"`Scan In`" pin is connected to the input of the first flop in the chain to send a series of patterns (1's and 0's) into the chain. "`Scan Out`" is the output pin where the shifted pattern comes out. "Scan Enable" pin is the input pin used to enable the scan mode. "`Scan Clock`" is the input pin used to apply the clock signal to do the shifting of patterns. Scan mode analysis supports the following two modes:

- **Scan shift mode**- You need to provide the initial state of all scan flops and scan-chain order. In this mode, Voltus will:

  a. "shift" the state across scan-chain at scan clock freq (SDC) depending upon number of cycles specified for simulation

  b. make "state" assumption for combinational logic which is driven from PI

  c. perform "state/logic" propagation of flop states to the logic cone.

   d. capture dynamic current profile, and run Irdrop analysis w/ package.

- **Vector-less mode**- You need to provide scan-chain order and % activity [global | each scan-chain]. In this mode, Voltus will:

   a. generate scan pattern based on activity and internal vector-less heuristic to decide which flops will toggle in each cycle

   b. use “state/logic” propagation or “vector-less” propagation for combination logic cone

   c. capture dynamic current profile, and run Irdrop analysis w/ package.

To run this flow, you must specify the `-scan_control_file` parameter of the `set_power_analysis_mode` command. The use model for this flow is:

```
set_power_analysis_mode -method dynamic_vectorbased -scan_control_file filename
```

The scan control file contains details about the scan analysis mode used for testing.

Note: You can use the `set_power_analysis_mode -scan_chain_name` {*design_name chain_name pattern*} parameter to automatically generate a scan-chain control file from a scan-chain based DEF file for the shift mode scan-chain analysis. The generated scan-chain control file is saved in the power output directory.

Scan Control File Format

The format of the scan control file is:

```
MODE = [ SHIFT | VECTORLESS]
VECTORLESS_GLOBAL_ACTIVITY = <factor> [Between 0 - 1]
SCAN_PATTERN = <digital_pattern> <chain_name>
<chain_name> <flop_inst> <initial_state> <master_cell_name> <scale> <mode>
<scan_in_pin> <scan_out_pin> <scan_en_pin>
```

Note that `<initial_state>` is optional in the vector-less mode, `<master_cell_name>`, `<scale>`, `<mode>`, `<scan_in_pin>`, `<scan_out_pin>`, `<scan_en_pin>` are optional and only needed for the first internal scan flop inside one macro.

Where,

- `Mode` specifies the mode used for scan design analysis. The two modes are: `SHIFT` and `VECTORLESS`.
- `VECTORLESS_GLOBAL_ACTIVITY` specifies the activity specification used in the `VECTORLESS` mode.
- `SCAN_PATTERN` specifies the digital 0/1 pattern to be used in the scan chain repeatedly, and this pattern will be shifted afterwards through the scan chain. You can specify the scan pattern only in the `SHIFT` mode. The initial state of the scan chain follows the

SCAN_PATTERN from the start of the scan chain to the end of the scan chain, as shown in the following diagram:

You must specify the scan chain name in the SCAN_PATTERN line for each scan chain to assign a unique digital pattern to the specified scan chain. In the following scan control file example, scan pattern has been specified for two scan chains:

```
MODE = SHIFT
SCAN_PATTERN = 10101 AutoChain_1
AutoChain_1 inst_i0
AutoChain_1 inst_i1
...
MODE = SHIFT
SCAN_PATTERN = 11111 AutoChain_2
AutoChain_2 inst_i5
AutoChain_2 inst_i6
...
```

You can also assign the same digital pattern to all the chains present in the design (global specification) by not specifying the scan chain name in the SCAN_PATTERN line, as shown in the following example:

```
MODE = SHIFT
SCAN_PATTERN = 1000
SC1 SC1/DFF1
SC1 SC1/DFF2
SC1 SC1/DFF3
SC1 SC1/DFF4

SC2 SC2/DFF1
SC2 SC2/DFF2
SC2 SC2/DFF3
SC2 SC2/DFF4
```

here,

- SC1 chain and SC2 chain will have the same digital pattern, that is, the pattern 1000 will be passed through its flop instances.

- `chain_name` is the name of the scan chain.
- `initial_state` is the initial state of all scan flops.
- `master_cell_name` is the name of the macro.
- `scale` is the scaling factor applied to the current.
- `mode` is mode name of the scan mode in the PGV view.
- `scan_in_pin` is the macro's scan_in pin name.
- `scan_out_pin` is the macro's scan_out pin name.
- `scan_en_pin` is the macro's scan_enable pin name.

The following is a sample scan control file for the vectorless mode:

```
MODE = VECTORLESS
VECTORLESS_GLOBAL_ACTIVITY = 0.5
chain_0  ../../ Idly_ff0  1  ampslv 1 SCAN_MODE scan_in scan_out tst_amp_scanEn
```

The following is a sample scan control file for the shift mode:

```
MODE = SHIFT
SCAN_PATTERN = 0101 scan_chain_0
scan_chain_0 inst_i0
scan_chain_0 inst_i1
scan_chain_0 inst_i2
scan_chain_0 inst_i3
scan_chain_0 inst_i4
scan_chain_0 inst_i5
```

You can use the `set_power_analysis_mode -report_scan_chain_stats` parameter to generate a report file called `voltus_power.scanchain.stat` during scan mode analysis. The report gives the statistics for the scan chains in the design.

Generating User-defined Mode Based Current Waveform with Automatic PWL Scaling

Given one master cell PWL waveform, Voltus can automatically generate scaled PWL waveforms of the other companion cells defined in a trigger file proportional to the master cell internal power. It keeps the waveform shape and scales the waveform magnitude based on the mode and net-dependent relative ratio of the NLPM internal power energy table value to the master cell model. This feature decouples the mode-based current waveform from PGV, and allows power analysis to directly take the PWL format from the user-defined trigger file. You do not need a dynamic PGV for Voltus power and rail analysis.

Note: The companion cells must have the same configuration as the master cell, and they can only be in different sizes.

Use the `-custom_macro_pwl` parameter of the `set_power` command to specify a custom trigger file that allows you to apply instance or cell PWL waveforms.

- In the vector-based analysis, the user-defined PWL model overrides the .lib model. When instance input pins' state matches with the `CONDITONAL_INPUT` and `CONDITIONAL_PIN` description, the PWL waveform is applied.
- In the vectorless mode, the `-custom_macro_pwl` parameter must be used with `set_power -dynamic_switch_pattern` in the vectorless flow.

The use model of the `-custom_macro_pwl` parameter is:

```
set_power -custom_macro_pwl { ./cell.trigger } -dynamic_switch_pattern -cell
SRAM_A1 { MODE_0 MODE_0 MODE_0}
```

Example of a Custom Macro Power Trigger File (cell.trigger)

```
MASTER_CELL        SRAM_A1
COMPANION_CELLS    { SRAM_A2, SRAM_A3, SRAM_A4, SRAM_A5 }
      MODE_NAME WRITE
               CONDITIONAL_INPUT = ( !SD & !SLP & !CEB & !WEB )
               CONDITIONAL_PIN = CLK { RISE }
               USER_PWL_FILE write_rise.txt
      END_MODE
      MODE_NAME READ
               CONDITIONAL_INPUT = ( !SD & !SLP & !CEB & WEB )
               CONDITIONAL_PIN = CLK { RISE }
               USER_PWL_FILE read_rise.txt
      END_MODE
      ...
END
```

For information on the trigger file format, see the **Trigger File Format** section in the `set_power` section of *Voltus Text Command Reference*.

The following information is displayed in the log file:

```
-----------------------------------------------------------------
INFO : Macro cell <Name of the cell> used user-defined current waveform.
-----------------------------------------------------------------
```

Voltus power saves the waveform scaling factor of each companion cell in an individual text file in the output directory. The file name is `scale_factor_report_<master_cell_name>.txt`. The following is an example of the text file:

```
# CellName    NET     MODE      Scale-Factor
  SRAM_A2     VDD     WRITE      1.0
  SRAM_A2     VDDM    WRITE      1.0
  SRAM_A2     VSS     WRITE      1.0
```

Dynamic Rail Analysis Overview

In addition to static analysis, Voltus offers you the option of running dynamic gate-level analysis. You can use the dynamic approach to optimize the power network, minimize IR drop transients, and reduce leakage associated with added decoupling capacitance. Specifically, dynamic power-rail analysis can:

- Detect power integrity problems.
- Provide insight into the effect of local simultaneously switching high-power drivers.
- Quantify instantaneous IR drop.
- Assess the impact of resistance, capacitance, and inductance due to packaging, bond wire, and C4 bumps on transient IR drop. See Package Analysis.

Rather than averaging currents as static analysis does, dynamic analysis uses peak currents as the basis for the analysis, enabling you to see the finer time variation of currents over a clock cycle. Voltus's dynamic, or transient, approach to power-rail analysis uses a dynamic power or current waveform that you can generate either by performing logic simulation or by using timing window information. The analysis also includes the impact of package inductance.

Power integrity

Power is consumed in three basic ways in integrated circuits:

- Switching power, which is the power consumed in the charging and discharging of interconnect capacitances. In most cases, this type of power consumption dominates because of large drivers having to drive large capacitive loads.
- Internal power, which is the power consumed in the charging and discharging of the internal capacitances within gates and the capacitances in driving the load. It also includes feedthrough current from VDD to VSS, which occurs for a short time when both NMOS and PMOS devices are on. Feedthrough current is also known as crossbar, crowbar, or short-circuit current. The faster the slew rate, the less feedthrough current there is.
- Leakage power, which is the power consumed by devices when they are not switching. It includes state-dependent leakage, which is leakage that depends on the state of the gate, that is, whether a transistor is on or off.

For additional details, see Type of power (internal, leakage, switching).

Source of Decoupling Capacitance

There are five kinds of decoupling capacitance that can impact dynamic power rail analysis:

- Parasitic capacitance between power and ground
- Device decoupling (diffusion and gate capacitance)
- Discrete decoupling, added to the power rails
- Natural decoupling capacitance from signal net capacitances through gates
- Well capacitance

The well capacitance component is not currently modeled by Voltus, because the impact of this component is less than 10 percent for common process and design styles, when compared to the other four components.All other components of decoupling capacitance are accurately extracted using the embedded three-dimensional parasitic extraction engine, either during the processing of the main design input files or when generating the power grid views. A design can include all five forms. Decoupling capacitors operate as charge reservoirs to stabilize the power and ground grid and to minimize transients. Before the advent of 90-nanometer designs, design teams would typically add a significant number of decoupling capacitors between the power and ground rails to help smooth any voltage transients on the rails and to minimize transient voltage spikes caused by the switching of large buffers. As a result, the noise on the power grid was minimized.For 90-nanometer designs and below, however, the increased leakage from decoupling capacitors begins to add to the power consumption, so you can no longer liberally add them to minimize transients. For low-power and 90-nanometer designs, analyzing where to add and remove decoupling capacitors is of critical importance. Dynamic power-rail analysis gives you the necessary information to understand how to maximize the efficiency of the added decoupling capacitors to minimize the power consumption. Dynamic power-rail analysis helps you understand the optimal size of local decoupling capacitance, the optimal location in which to place the decoupling capacitors, and the impact of inductance on power rail transients. For example, consider the situation shown in the following figure:

Dynamic IR Drop

In the area shown in the circle, simultaneously switching outputs have caused more power to be used, resulting in voltage dropping below 1.15 volts.

You could do one of three things to correct this problem:

- Re-position the existing decoupling capacitors to be more effective.
- Add local decoupling capacitors to reduce the IR drop transients.
- Import transient IR drop data into the timing flow to understand the impact on delay.

Voltus's dynamic analysis reports regions with too few decoupling capacitors, enabling you to add more decoupling capacitors to limit IR drop transients, and regions with too many decoupling capacitors, enabling you to save power and space. It also includes the impact of package inductance.Dynamic analysis reports IR drop transients on the power networks, typically caused by localized simultaneous switching of devices, which can be smoothed by the addition of decoupling capacitance. Voltus also reports both the density and effectiveness of the decoupling capacitors, which can be used to direct where decoupling capacitance can be optimized to either reduce IR drop transients or lessen leakage.The following figure shows how adding decoupled capacitance smooths the transient drops on the VDD line.

Effect of added Decoupling Capacitance

Adding decoupling capacitance is performed by replacing filler cells with filler cells containing decoupling capacitance (decaps). Filler cells are cells that do not have signal pins and do not consume any power. Decoupling capacitance cells are cells that do not have signal pins, but have an internal capacitance (mostly in the form of MOS gate capacitance) implemented. Because of the internal capacitance, they cause leakage current.

Simultaneous switching

Groups of high-power drivers located in the same power region that switch synchronously drain power from the power rail and cause local transient IR drop in places, which must be included in the timing flow. Static analysis provides only limited insight into this problem. When your design has several simultaneously switching events and little decoupling capacitance, dynamic analysis enables you to identify optimal placement of decoupling capacitors to minimize transients.IR drop due to simultaneous switching is known as transient IR drop.The following figure shows a group of local high-power drivers that switch synchronously:

High-power drivers in local area switching synchronously

Instantaneous IR Drop

IR drop is an instantaneous phenomenon that is time-dependent and therefore only measurable by dynamic transient techniques. In addition, obtaining just the measurement of the IR drop is often insufficient. It is the shape and the magnitude of the IR drop that has the highest impact on timing. A large, very fast spike is typically recoverable and may have minimal impact on timing, but a spike of lesser magnitude but with a much longer duration can disrupt timing much more.Static analysis only gives a qualitative assessment of the design while dynamic analysis gives quantitative assessment. Static analysis give visibility into the quality of the grid, but not in absolute numbers over time as is done in dynamic analysis.

Packaging parasitics

For designs 130 nanometers and smaller, parasitic resistance, capacitance, and inductance due to packaging, bond wire, and C4 bumps can have a significant impact on transient IR drop at modest frequencies. At higher frequencies, the resonance can interfere with the on-chip clocks. Inductance in particular becomes a problem outside the chip because of the effects of the bond wire and the packaging routing.Voltus accepts package RLC modeling in the form of SPICE subcircuits generated from different packaging modeling products such as Allegro. You can specify the name of a file containing a SPICE subcircuit describing a package model.See Package Analysis for additional details.

Setting up and running Dynamic Rail Analysis

This section describes how to set up the dynamic rail analysis mode and then run rail analysis.

Required Input Files

- Power-grid libraries (.cl). See Power-Grid Library Generation for details on generating power-grid libraries.
- Vectorless or vector-based files created during dynamic power calculation

Procedure

To perform dynamic rail analysis, complete the following steps:

1. Define the rail analysis mode through the command line or through the GUI.

   GUI: Select *Power Rail* - *Set Rail Analysis Mode*.

   Tcl command:

```
set_rail_analysis_mode \
    -method                     dynamic \
    -analysis_view              AV_wc_on \
    -generate_movies            true \
    -save_voltage_waveforms     true \
    -accuracy                   hd \
    -temperature                125 \
    -gif_resolution             0 \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    }
```

   Note:

   - *XD* uses LEF based views, and *HD* uses GDS based accurate views.
   - The technology library is a mandatory requirement for rail analysis. However, to obtain accurate IR drop simulation, power-grid library should be characterized for standard cells and macro blocks. When specifying the list of PGVs to rail analysis

using the `set_rail_analyis_mode` command or the Set Rail Analysis Mode form, you must specify the technology library as the first PGV library in the list.

- ❑ You need to specify the EM analysis models if analyzing Electromigration. If you do not specify an Electromigration model file, the software will select Electromigration models from the Quantus technology file. You can use the command `set_rail_analyis_mode -em_models` file to override these settings.

2. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

   Tcl command:

```
set_pg_nets -net VDD_AO     -voltage 0.9 -threshold 0.85
set_pg_nets -net VDD_ring   -voltage 0.9 -threshold 0.85
set_pg_nets -net VSS        -voltage 0.0 -threshold 0.05
set_rail_analysis_domain -name PD_ring -pwrnets VDD_AO  -gndnets VSS
-threshold 0.10
set_rail_analysis_domain -name PD_AO -pwrnets VDD_AO  -gndnets VSS
-threshold 0.10
```

3. Define the location of the voltage sources for each power/ground net.

   Tcl command:

```
set_power_pads -net VDD_AO -format xy -file
../design/super_filter_VDD_AO.pp
set_power_pads -net VSS -format xy -file
../design/super_filter_VSS.pp
```

4. Specify the location of the current files generated during dynamic power analysis.

Tcl command:

```
set_power_data -reset
set_dynamic_rail_simulation -reset
set_dynamic_rail_simulation -resolution 50ps
set_power_data \
    -format                     current \
    { \
        dynVecLessPowerResults/dynamic_VDD_AO.ptiavg \
        dynVecLessPowerResults/dynamic_VDD_column.ptiavg \
        dynVecLessPowerResults/dynamic_VDD_ring.ptiavg \
        dynVecLessPowerResults/dynamic_VDD_external.ptiavg \
        dynVecLessPowerResults/dynamic_VSS.ptiavg \
    }
```

5. Run rail analysis through the command line or through the GUI.

   GUI: Select *Power Rail* - *Run Rail Analysis*.

   Tcl command:

```
analyze_rail \
    -output ./dynVecLessRailResults \
    -type                       domain \
```

The Voltus console will display rail analysis statistics, as shown in the following snippet:

```
Rail Analysis completed successfully.


Finished Rail Analysis at 11:56:53 08/20/2018 (cpu=0:00:20, real=0:00:34, peak
mem=1218.10MB)
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:03:56,
real=0:58:07, mem=1218.10
```

Output

The rail output directory will contain the following information:

Directory Name	Description
`<domainName>_<temperature>_dynamic_<X>`	For each run, an output directory is generated with the default name `<domainName>_<temperature>_dynamic_<X>`. `<domainName>` will be replaced with `ALL` if all domains are to be analyzed (`analyze_rail -output dir -type domain ALL`), or with `<netName>` if a power net is to be analyzed. The output directory contains state directories for each net, such as VDD and VSS. Each state directory contains the following: ■ `results` – a file that gives a summary of the rail analysis results for various plots, such as tc (tap current), rc (resistor current), and so on. ■ `Reports` – a folder that contains a list of useful report files that contain all power-grid integrity data. ■ Auto-generated GIF files for the rail analysis plots

TCL Command:

TCL command for dynamic rail analysis

```
set_rail_analysis_mode -method dynamic \
    -generate_movies true -save_voltage_waveforms false \
    -accuracy hd -power_grid_library \
        {accurate_stdcells.cl memories.cl}
set_rail_analysis_domain -name PDcore -pwrnets VDD -gndnets VSS
set_pg_nets -net VDD -voltage 1.08 -threshold 0.972 -tolerance 0.3
set_pg_nets -net VSS -voltage 0.0 -threshold 0.108 -tolerance 0.3
set_power_data -format current {dynamic_VDD.ptiavg dynamic_VSS.ptiavg}
set_power_pads -net VDD -format xy -file VDD.ppl
set_power_pads -net VSS -format xy -file VSS.ppl
analyze_rail -type domain -output ./ PDcore
```

Run Dynamic Power Analysis and Rail Analysis in Parallel

You can run dynamic power analysis in parallel with rail analysis to improve the overall turnaround time. To run this flow, you must specify the following parameter:

```
set_rail_analysis_mode -report_power_in_parallel
```

In this usage model, you do not need to modify the power analysis setup. Remove the `report_power` command and set the `-report_power_in_parallel` parameter of `set_rail_analyis_mode` to `true`. The `analyze_rail` command will run both dynamic power and rail analysis in parallel. The usage model is given below:

#Power Analysis Setup

```
set_power_analysis_mode...
set_switching_activity ...
read_activity_file ...
set_power ...
set_power_output_dir <dirname>
```

Important

If `set_power_output_dir` is not specified, it defaults to the current working directory to look for the required current files.

#Rail Analysis Setup

```
set_rail_analysis_mode -report_power_in_parallel <true|false> ..(default: false)
set_pg_nets ...
set_rail_analysis_domain ...
set_power_pads ...
```

Important

In this parallel flow, `set_power_data <power_file>` will be set internally to the auto-generated power data file in the directory specified by the `set_power_output_dir` command. Therefore, `set_power_data <power_file>` is not required when `-report_power_in_parallel` is set to `true`. If `set_power_data <power_file>` is still specified, the tool will error out. To run the `set_power_data` parameters (scale/offset/etc.) on the dynamic power current files, specify it with the parameter `-power_output_directory`, as shown below:

```
set_power_data -power_output_directory <dirname> ...
```

#Run Dynamic Power and Rail Analysis

```
analyze_rail
```

Dynamic Rail Analysis Plots, Movies and Waveforms

Dynamic rail analysis plots interface is available through Power Rail - Power Rail Results form. The interface is same as static rail analysis plots but in addition dynamic rail analysis offers various capacitance plots like grid capacitance, available capacitance, filler cell density, etc. Refer to Viewing Static Rail Analysis Results for more information on GUI and TCL interface.

The following table lists the dynamic analysis plot types and the corresponding controls to enable them:

Type	Control
power gate current I/Idsat (pi), power gate voltage (pv)	enabled automatically when power-gate(s) are connected to the analyzed net.
voltage across package model (vu)	enabled automatically when package data is provided (spice model, in power pin location file, or through `default_package_*` commands)
instance effective resistance (rlrp)	enabled if `set_rail_analyis_mode -enable_rlrp_analysis true`
resistor current (rc)	enabled if `set_rail_analyis_mode -enable_rc_analysis true`
voltage drop across vias (vv)	enabled if `set_rail_analyis_mode -enable_voltage_across_vias true`

In addition to dynamic plots, dynamic IRdrop and Tap Current movies are generated that can be viewed using *Power Rail - Dynamic Results - Waveforms* form. The movies are also available using the TCL command: `view_dynamic_movie -type ir -movies_directory VDD/movies_directory`. The dynamic IRdrop waveforms are created for all instances in state directory as `net.ptiavg` file. The current waveforms through voltage sources are saved inside `voltus_rail_pad.tran.ptiavg` file to analyze impact of on-chip capacitance. The dynamic IRdrop waveforms can be plotted using *Power Rail - Dynamic Results - Waveforms* form.

To plot rail current, voltage, and power current waveforms:

1. Select *Power Rail -> Dynamic Results -> Waveforms*.

The following form appears:

2. In the *Waveform Files* section, click *Add* to add the current and voltage waveform (.ptiavg) and profiling (.trn) files.

   The list of nets/instances for the selected waveform files are displayed.

3. Select `Rail Current`, `Voltage`, and/or `Power Current` to filter the instances based on the waveform type.

   - `Rail Current` displays waveform instance names with prefix `TC_`.
   - `Voltage` displays waveform instance names with suffix `:ir`.

- ❑ `Power Current` displays waveform instance names without a prefix or suffix.

4. Filter instances using the filter pattern. To filter instances, enter a pattern in the *Pattern* field, and click *Instances*.
5. Select the required instances and click the *Plot* button.

   The following form appears:

TCL Commands:

TCL Command for viewing dynamic voltage waveforms

```
view_dynamic_waveform  -type  { rail_current voltage} -instance_name I1_1
-waveform_files { PD_25C_dynamic_1/Reports/Waveforms/tran_VSS.ptiavg
dynamic_power/part_0/dynamic_VSS.ptiavg}
```

Dynamic Rail Analysis for On-Chip Voltage Regulators

System-on-a-chip (SoC) voltage regulators (Vreg) have been widely used in designs to improve device power efficiency as well as device stability. Voltus IC Power Integrity Solution supports analysis of on-chip Vreg in SoC to stabilize voltage supplies and to handle Dynamic Voltage and Frequency Scaling (DVFS). This analysis has been included in the IR Drop analysis flow to understand how Vreg affects the on-chip voltage drop, chip package co-design, and how it interacts with the IC/package/board system. This feature allows you to accurately model Vreg modules for rail analysis.In the following diagram, the chip configuration has a Vreg that is connected to an input and output domain along with a common ground:

Chip Configuration

Vreg Setup in Voltus

During Vreg analysis, the tool captures the noise at the output of Vreg caused due to loading current and RC effects, and uses this time varying voltage waveform for rail analysis. When you perform Vreg analysis, a reduced grid is created for both input and output, and reduced ground resulting in a much concise RC netlist to do simulations of vectors with these loadings. The noise at the output of Vreg is captured and used for rail analysis.

Note: Designs with on-chip Vreg have a non-linear circuitry of Vreg in the midst of large linear networks of power-grid. Therefore, the tool simulates the Vreg module using Spectre and the linear networks of power-grid using Voltus solver.

Use the `set_voltage_regulator_module` command to enable the Voltage Regulator analysis flow. You must specify this command multiple times for multiple Vreg instances. This command is used to specify all required information for defining a Vreg module, such as Vreg module name, SPICE netlist, power pin mapping for input/output/ground domains, component list (DEF instances working as a voltage regulator), and top-level Vreg subcircuit. The Spice netlist of the Vreg cell should encapsulate the testbench required for the Vreg module to work.When Voltus loads this netlist, it traces the connectivity inside the netlist and maps the top-level pin to instance pin, if the VRM is constructed by multiple sub-instances. After identifying all physical pin locations for Vreg input and output, the physical pins for Vreg

output is used as voltage source. The location and placement of voltage sources on VRM cells is done by the tool based on the LEF port definitions of the power pin of the VRM cells.

Example TCL script for Vreg analysis

Load the design# Perform Power Analysis# Rail Analysis

```
set_pg_nets -net VDD_out -voltage 1.35 -threshold 1.21 -tolerance 0.3 -force
set_pg_nets -net VDD_in -voltage 3.00 -threshold 2.70 -tolerance 0.3 -force
set_pg_nets -net VSS -voltage 0.00 -threshold 0.13 -tolerance 0.3 -force
set_rail_analysis_domain -name PD -pwrnets {VDD_out VDD_in} -gndnets VSS
```

Important

For running this flow, you must analyze the input domain, output domain, and ground domain in rail analysis.

```
set_rail_analysis_mode \
    -method dynamic \
    -accuracy xd\
    -power_grid_library ../accurate_stdcells.cl

set_power_data \
    -format current { \
    dynamic_power/dynamic_VDD_out.ptiavg \
    dynamic_power/dynamic_VDD_in.ptiavg \
    dynamic_power/dynamic_VSS.ptiavg \
    }

set_power_pads -net VDD_in -format xy -file ./VDD_in.pp
set_power_pads -net VSS -format xy -file ./vs/VSS.pp
set_voltage_regulator_module \
    -name vrm1 \
    -netlist ./TEST.spi \
    -input_pwr_pin_mapping {{ VDD_in VDD_in }} \
    -output_pwr_pin_mapping {{ VDD_out VDD_out }} \
    -ground_pin_mapping {{ VSS VSS }} \
    -subckt_name VRM_MODEL \
    -component_list { reg_1 }

set_voltage_regulator_module \
    -name vrm2 \
    -netlist ./TEST.spi \
    -input_pwr_pin_mapping {{ VDD_in VDD_in }} \
    -output_pwr_pin_mapping {{ VDD_out VDD_out }} \
    -ground_pin_mapping {{ VSS VSS }} \
    -subckt_name VRM_MODEL \
    -component_list { reg_2 }

analyze_rail \
    -type domain \
    -output ./dynamic_rail \
    PD
```

After performing rail analysis, you can view the time varying waveform created at the output domain of the voltage regulator. The current waveforms through voltage sources are saved

inside the `voltus_rail.tran.ptiavg` file to analyze voltage variation. The dynamic IRdrop waveforms can be plotted using *Power Rail - Report - Dynamic Waveforms* form. The following is a sample waveform that illustrates the voltage variation at the output domain:

Using Characterized Current PGV for Dynamic Power and Rail Analysis

Power-grid view library with dynamic data is supported in both the vector-based and vectorless Voltus power calculation. By default, Voltus computes dynamic power from .lib files. When there are pre-defined currents stored in PGV, Voltus dynamic power calculation automatically identifies the dynamic data and trigger information inside the power-grid view library.

- In the vector-based analysis, Voltus looks at the trigger condition in VCD and schedules the current modes with the matching condition.
- In the vectorless mode, you can control the trigger scenario by using the `set_power -dynamic_switching_pattern` parameter. This can be performed in both the instance-based and cell-based switching.

Example of instance-based switching:

```
set_power -instance mem1 -dynamic_switch_pattern { MODE1 0 MODE2 0 }
```

This command specifies to trigger MODE1 @ first clock edge and MODE2 @ 3rd clock edge.

Example of cell-based switching:

```
set_power -dynamic_switch_pattern -cell SRAM_A1 { MODE_0 MODE_0 MODE_0}
```

This command specifies to trigger MODE_0 @ 1st, 2nd, and 3rd clock edges.

Following is an example of a trigger.txt file generated in the output directory at the end of the run that shows the various modes and trigger times:

```
mem1 1.95e-09 1 MODE1
mem1 3.195e-08 1 MODE1
mem1 4.195e-08 1 MODE1
mem1 2.205e-08 1 MODE2
mem1 3.205e-08 1 MODE2
```

Voltus dynamic rail analysis checks availability of the detailed dynamic data in the power-grid views from a given list of cell libraries. Rail analysis triggers the waveform stored inside a power-grid view based on the trigger time from the power analysis run. The following is a snapshot of the log file:

```
------------------
INFO Found a dynamic tap current view for instance "mem1" whose waveform is
triggered at time 1.95e-09s
-----------------
```

If the power-grid view library does not have any associated trigger time, then the dynamic current stored inside this library starts at t = 0. To shift this waveform, you can use the `set_rail_analyis_mode -dynamic_trigger_file <filename>` parameter. The syntax of dynamic trigger file is as follows:

```
INSTANCE <name> <time> <scale_factor> <mode_name>
CELL <name> <time> <scale_factor> <mode_name>
```

You can capture the current/voltage waveforms using the `set_rail_analyis_mode -probe_waveform_list` parameter.

Example TCL script for Dynamic Switch Pattern in the Vectorless Mode

```
set_power_analysis_mode \
  -power_grid_library {dynamic_allcells.cl} -method dynamic_vectorless

set_default_switching_activity ...
set_dynamic_power_simulation …
set_power -instance u0 -dynamic_switch_pattern {READ 0 READ}
set_power_include_file …
report_power …
set_rail_analysis_mode \
  -power_grid_library {dynamic_allcells.cl} ...
set_power_data ...
set_power_pads ...
set_pg_nets ...
set_rail_analysis_domain ...
analyze_rail …
```

Dynamic Mixed Mode (Vector-based and Vectorless) Analysis

In addition to the regular methods (`static`, `dynamic_vectorless`, `dynamic_vectorbased`) of power analysis, you can use the `dynamic_mixed_mode` method to combine the vector-based and state-propagation-based vectorless dynamic analysis. This flow can be used in the following cases:

Cases	When `dynamic_mixed_mode` is specified
`Case 1: vectors are not available for every block of a full-chip design`	■ The software uses the state-propagation-based vectorless methodology for blocks with incomplete vectors
`Case 2: blocks with vectors having unannotated combinational logic`	■ The software performs state propagation for the missing combinational logic ■ The software uses the vectorless methodology for a logic cone that does not have the primary inputs or flops annotated from the vector, and then performs state propagation

The two cases are illustrated in the following diagram:

To enable the dynamic mixed mode analysis flow, you must specify the following commands:

```
set_default_switching_activity -clock_gates_output 2.0 -input_activity <activity>\
                               -seq_activity <activity>
set_power_analysis_mode -method dynamic_mixed_mode
read_activity_file -scope <> -block <> -> (for available blocks)
```

In this flow:

- the specified global activity will be used for selection of unannotated flops and primary inputs in each physical tile.
- if there is a mismatch between the TWF and vector files in terms of clock frequency for the unannotated flops/primary inputs, the TWF frequency will take precedence and will be used for scheduling.
- when multiple vectors are provided for different blocks, they are required to be of the same vector type (post-simulation/zero-delay/rtl) and the same format (VCD, FSDB, PHY, SHM). Mixing of the vector types and formats of the different blocks is not supported.
- the simulation period for different windows in the vector-based analysis is derived from the `max(endtime - starttime)` specified using the `read_activity_file -start` and `-end` command parameters.

Refer to the Dynamic Vectorless Power Analysis Flow and Dynamic Vector-based Power Analysis Flow section for the required input files and the detailed steps to perform dynamic analysis.

In addition to the dynamic current files (`dynamic_<net_name>.ptiavg`) and power report, this flow generates the following output files:

- a switching scenario file (`voltus_power.stateprop.switchsrc`) that contains the switching information of flip-flops.
- a report file (`voltus_power.stateprop.stats`) that contains the instance toggle coverage, that is, number of primary input, flop, combinational, macro, and idle instances selected at different clock frequencies.

Tcl command:

```
read_lib ../DesignData//LIB/TYP_DRIVE.lib
../DesignData//LIB/TYP_PRIM.lib ../DesignData//LIB/TYP_SPECIAL.lib

read_lib -lef ../DesignData//LEF/lef.r_mod

read_verilog ../DesignData//VERILOG/my_chip1.v
../DesignData//VERILOG/my_chip2.v

set_top_module my_chip

specify_def { ../DesignData//DEF/def1.def
../DesignData//DEF/def2.def}

specify_spef { ../DesignData//SPEF/core.spef
../DesignData//SPEF/signal.spef}

read_twf ../DesignData//TWF/my_chip.twf

read_twf ../DesignData//TWF/mixed.twf -scope inst_A1

read_twf ../DesignData//TWF/mixed.twf -scope inst_B1

read_twf ../DesignData//TWF/mixed.twf -scope inst_C1

read_twf ../DesignData//TWF/mixed.twf -scope inst_D1

read_activity_file -format VCD -scope top/shell/soin -start 600ns -
end 700ns ../mixed_vcd/blocka.vcd -block inst_A1

read_activity_file -format VCD -scope top/shell/soin -start 600ns -
end 700ns ../vcd_hier/DesignData/VCD/blockb.vcd -block inst_B1

read_activity_file -format VCD -scope top/shell/soin -start 600ns -
end 700ns ../vcd_hier/DesignData/VCD/blockc.vcd -block inst_C1

read_activity_file -format VCD -scope top/shell/soin -start 600ns -
end 700ns ../mixed_vcd/blockd.vcd -block inst_D1

set_power_analysis_mode -method dynamic_mixed_mode -disable_static
false -power_grid_library ../DesignData//PGV/library.cl

set_default_switching_activity -input_activity 0.2 \
                               -seq_activity 0.1

report_power
```

Noise Margin Calculation

Noise margin calculation allows you to determine the allowable noise voltage on the input of a gate so that the output will not be affected. It is the amount of noise a circuit can resist without affecting its operation. These values are determined to ensure that the false noise signals are not causing any undesired operation. The noise values can be compared with the noise margins and if they are within the margins provided, it indicates the ability of the circuit to operate in the presence of noise.

The following diagram illustrates a chain of two inverters:

Here,

V_{oh} (minimum output high voltage) = `VDD_driver - VSS_receiver`

V_{ol} (maximum output low voltage) = `VSS_driver - VSS_receiver`

V_{ih} (minimum input high voltage) = `(1-noise_limit)*(VDD_receiver - VSS_receiver)`

V_{il} (maximum input low voltage) = `noise_limit*(VDD_receiver - VSS_receiver)`

The following is the criteria for noise margin calculation:

`Margin_High` = `NMH` $(V_{oh} - V_{ih})$ - `signal noise` - `guardband` > 0

`Margin_Low` = `NML` $(V_{il} - V_{ol})$ - `signal noise` - `guardband` > 0

Effective Instance Voltage Based Noise Margin Calculation

Voltus uses the Effective Instance Voltage method for noise margin calculation. In this method, noise margins are calculated with worst voltages seen on the driver-receiver pins. Noise margins will be calculated for only the driver-receiver pairs where flop is a receiver, as shown below:

This method uses the elapsed worst case voltages on the power and ground pins of the driver and receiver instances. This method has the following stages:

- **Power Analysis -** The Power Analysis engine will generate the driver-receiver pairs. This will be limited to the state-elements (flops) as noise margins will be calculated only for the driver-receiver pairs where flop is a receiver.

  You need to specify the `set_power_analysis_mode -create_driver_db true` command parameter to generate the driver-receiver database.

- **Rail Analysis -** The Rail Analysis engine automatically detects that this is for EIV-based noise margin calculation based on the availability of the driver-receiver database.

- **Calculate Noise Margin -** Specify the `calculate_noise_margin` command to calculate noise margin (NMH/NML). To run this command, you need to specify the default or cell-specific noise limit, signal noise, guardband, rail directory, and power directory.

  The format of the noise limit file is:

  ```
  default_noise_limit <value>
  default_signal_noise <value>
  default_guard_band <value>
  <cell_name> <noise_limit> <signal_noise> <guardband>
  ```

Performing Noise Margin Calculation

This section describes the various inputs and outputs of the noise margin calculation flow. Noise margin calculation can be done in both the static and dynamic power/rail analysis flow.

Required Input Files

- Design data (LEF, netlist, DEF)
- Common Power Format (CPF) file
- SPEF
- Power-grid libraries (.cl)

Procedure

To calculate noise margin, complete the following steps:

1. Load the design data.

   Tcl command:

   ```
   read_lib -lef $lefs
   read_view_definition ../design/viewDefinition.tcl
   read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
   set_top_module super_filter -ignore_undefined_cell
   read_def ../design/super_filter.def.gz
   ```

2. Specify the CPF file for power domain information.

   Tcl command:

   ```
   read_power_domain -cpf ../design/super_filter.cpf
   ```

3. Specify the SPEF file for signal net parasitic information.

   Tcl command:

   ```
   read_spef -rc_corner  RC_wc_125 -decoupled\
   ../design/postRouteOpt_RC_wc_125.spef.gz
   ```

4. Specify the sub-directory that will contain the power results.

Tcl command:

```
set_power_output_dir            dynVecLessPowerResults
```

5. Define the dynamic power analysis mode.

Tcl command:

```
set_power_analysis_mode \
    -reset
set_power_analysis_mode \
    -analysis_view              AV_wc_on \
    -write_static_currents      true \
    -binary_db_name             dynvectorlessPower.db \
    -create_binary_db           true \
    -method                     dynamic_vectorless
    -create_driver_db true
```

6. Optional. Define the simulation period and resolution.

Tcl command:

```
set_dynamic_power_simulation -reset
set_dynamic_power_simulation -resolution 50ps
```

7. Run power analysis.

Tcl command:

```
report_power -outfile  dyn.rpt
```

8. Define the rail analysis mode.

Tcl command:

```
set_rail_analysis_mode \
    -method                     dynamic \
    -analysis_view              AV_wc_on \
    -save_voltage_waveforms     true \
    -accuracy                   hd \
    -temperature                125 \
    -gif_resolution             0 \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    }
```

9. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

Tcl command:

```
set_pg_nets -net VDD_AO     -voltage 0.9 -threshold 0.85
set_pg_nets -net VDD_ring   -voltage 0.9 -threshold 0.85
set_pg_nets -net VSS        -voltage 0.0 -threshold 0.05
set_rail_analysis_domain -name PD_ring -pwrnets VDD_AO  -gndnets VSS
-threshold 0.10
set_rail_analysis_domain -name PD_AO -pwrnets VDD_AO  -gndnets VSS
-threshold 0.10
```

10. Define the location of the voltage sources for each power/ground net.

Tcl command:

```
set_power_pads -net VDD_AO -format xy -file
../design/super_filter_VDD_AO.pp
set_power_pads -net VSS -format xy -file
../design/super_filter_VSS.pp
```

11. Specify the location of the current files generated during dynamic power analysis.

Tcl command:

```
set_dynamic_rail_simulation -resolution 50ps
set_power_data \
    -format                     current \
    { \
        dynVecLessPowerResults/dynamic_VDD_AO.ptiavg \
        dynVecLessPowerResults/dynamic_VDD_column.ptiavg \
        dynVecLessPowerResults/dynamic_VDD_ring.ptiavg \
        dynVecLessPowerResults/dynamic_VDD_external.ptiavg \
        dynVecLessPowerResults/dynamic_VSS.ptiavg \
    }
```

12. Run rail analysis.

Tcl command:

```
analyze_rail \
    -results_directory          ./dynVecLessRailResults \
    -type                       domain corecurrent \
```

13. Calculate noise margin.

Tcl command:

```
calculate_noise_margin -noise_limit_file ./noiselimit \
-state_directory        dynVecLessRailResults/core_125C_dynamic_1\
-power_directory        dynVecLessPowerResults
```

Output

The state directory will contain the following file:

File Name	Description
powernetname_ground netname.noise_margin	A report that gives a summary of the noise margin values for the driver-receiver pins. # **File Format for EIV-based method**: `<Margin_High> <Margin_Low> <NMh> <NMl> <Voh> <Vol> <Vih> <Vil> <Signal_Noise> <Guardband> <Dr Inst/pin> <Rx Inst/pin>`

13

Hierarchical Power Integrity Analysis with Extreme Modeling

- Overview on page 332
- Using Extreme Modeling on page 333
- xPGV Generation on page 334
- xPGV Instantiation on page 335
- Setting up and Running the Hierarchical PI Analysis Flow on page 336

Overview

Voltus can perform hierarchical power integrity (PI) analysis of large designs using the Extreme Modeling (XM) technology. Voltus-XM allows you to create xPGV models for the IP blocks which accurately capture the demand current and electrical parasitics of the block. These xPGV models are smaller compared to the fully extracted block and when used in the chip-level analysis, it can help significantly reduce runtime and memory. The generated IP models can be reused in different designs or for multiple instantiations within a design. The modeling methodology used in Voltus-XM ensures minimal result difference relative to a fully flat analysis.

Voltus-XM (also known as xPGV generation) is an advanced PGV modeling flow that has been introduced for generating power-grid view (xPGV) models for large digital blocks. The hierarchical analysis flow with XM is intended primarily for reducing the compute resource requirements for the analysis of large designs. It can be used when there are multiple DEF files composing a design, and the blocks represented by the component DEF files can be analyzed independently.

Important

The xPGV view is not intended for use on small macro blocks representing power switches which feed other blocks. The modeling internal to the xPGV views is based on current sources, and this can lead to inaccurate results if current flows out of the xPGV block. Avoid using this flow if there are LEF pins defined for switched nets on the block which are intended to behave as outputs.
If a block contains power switches, but the switched nets are completely contained within the block, then the xPGV flow can be used. The switched net data will not appear in the view, but the current consumed by the switched nets is included in the via currents of the always-on nets that supply power to the switched nets.

Using Extreme Modeling

The hierarchical analysis flow with XM has two steps, as shown in the following diagram:

- **xPGV generation (Block Analysis)** - perform chip-level or block-level rail analysis to obtain only block-level instance current data and generate xPGV model.
- **xPGV instantiation (Top-Level Analysis)** - instantiate the generated xPGVs and use them to perform top-level rail analysis.

This flow requires the Voltus-XM license. It is an option to the baseline license Voltus-XL. The licensing use model for Voltus-XM is:

- **xPGV generation** - One Voltus-XM license is required for generating one xPGV model.
- **xPGV usage** - One Voltus-XM license is required for each xPGV model that is used in the analysis, up to a maximum of 10 XM licenses. Usage of more than 10 xPGV models does not require more than 10 licenses. For example, 10 XM licenses will suffice for the use of 12 xPGV models. License requirement is based on number of models used instead of instantiations. If a CPU block's xPGV model is instantiated four times, it will require only one XM license.

xPGV Generation

The block xPGV can be generated either after the signoff static/dynamic IR drop analysis or after the static/dynamic Early Rail Analysis (ERA) of the block. In both cases, the xPGV is generated using the R-C grid of the block and currents captured at the via layer during the analysis. The xPGV generation step creates a context-independent model that does not require a chip-level context for building the xPGV model. Therefore, there is no need to simulate the isolated partitions (surrounding blocks) of large designs during block-level analysis. It has the ability to dynamically create xPGV models for only those IP blocks of the design that you want to analyze.

The following diagram illustrates xPGV model generation:

In the xPGV model generation flow:

- the via cut layer for the upper layers is defined by users.
- top layers above the cut layer are preserved.
- tap currents at the device pin layers are preserved.
- massively reduced RC network is created between the specified via cut layer and current taps.
- the outside connection of the model is defined by LEF.

The reduced RC model is created between the instance taps (instance LEF pins) and the user-specified via cut layer, as shown in the following diagram:

The grid below the via layer, and any grid portion above the via layer which has become disconnected due to the removal of the lower layers are discarded. Any LEF ports below the via layer specified for cutting the grid are also discarded. This reduces the size of the view significantly, because the majority of the resistors in most blocks are on the lower metal layers. This also means that you will not see the lower layers of the block and switched supply nets when using xPGV views.

For block xPGV generation, you need to follow the usual steps of data preparation, power calculation, and rail analysis on the block-level DEF. The xPGV view is created from the results of the block-level rail analysis. To create the xPGV model, rail analysis keeps track of the currents flowing through the user-specified via cut layer in the block.

xPGV Instantiation

The block-level xPGV view can then be used to perform rail analysis of the DEFs at the next level up in the hierarchy or for top-level full-chip analysis. The next level rail analysis will run faster with the block-level xPGVs because these runs use the power-grid that is modeled inside the xPGV.

Setting up and Running the Hierarchical PI Analysis Flow

For the hierarchical analysis flow, you need two Tcl scripts (first for xPGV generation and second for xPGV instantiation).

Required Input Files

- Design data (LEF, netlist, DEF)
- Quantus technology file
- Power-grid libraries

Procedure

Following are the steps for running this flow:

1. In the first Tcl script, load the block-level design data.

   Tcl command:

   ```
   read_lib -lef $lefs
   specify_def {/Data//def/top.def /Data//def/hier_block.def}
   specify_spef {Data//spefs/top.spef.gz \
   /Data//spefs/hier_block.spef.gz}
   ```

2. Specify the sub-directory that will contain the power results.

   Tcl command:

   ```
   set_power_output_dir            dynVecLessPowerResults
   ```

3. Specify the power analysis commands.

Tcl command:

```
set_power_analysis_mode \
    -method                     dynamic_vectorless \
    -disable_static             false \
    -binary_db_name             dynvectorlessPower.db \
    -create_binary_db           false \
    -enable_mt_in_vectorbasedflow       true \
    -power_grid_library        /hierpgv_dyn_accurate/All_merge.cl\
    -enable_state_propagation   true
set_default_switching_activity -input_activity 0.2 \
  -seq_activity 0.1 -clock_gates_output 2.0 \
set_dynamic_power_simulation -period 20 -resolution 20
```

4. Specify the rail analysis commands.

Tcl command:

```
set_rail_analysis_mode \
    -accuracy hd \
    -method dynamic \
    -temperature 125 \
    -ignore_shorts true \
    -limit_number_of_steps false  \
    -use_em_view_list /Data//RAIL/submegacells.list \
    -snap_layer_for_current_taps \
    /Data//RAIL/megacell.collapse_tap_current_layer \
    -power_grid_library $pgv_list \
    -report_power_in_parallel true \
    -xpgv_config_file xpgv.conf
set_power_pads -format defpin -net VDD1
set_pg_nets -net VDD1 -force -voltage 1.21 -threshold 1.08
set_power_pads -format defpin -net VSS
set_pg_nets -net VSS -force -voltage 0.0 -threshold 0.12
set_rail_analysis_domain -name core -pwrnets VDD1 -gndnets VSS
analyze_rail -type domain -output dynamic_rail core
```

Note: The following parameter is specific to the xPGV generation flow:

- `-xpgv_config_file` *`file`* - specifies block details, such as, block name for which xPGV is generated, via cut layer name that indicates to preserve the layers

about the cut layer, and block LEF file names. This parameter is specified during block-level analysis.

Configuration File Format:

```
CELL <block_name>
CUT_LAYER <cut_via_layer_name>
Lef <tech_lef_name   block_lef_name>
END.
```

Example

```
CELL hier_block
CUT_LAYER V4
Lef {/Data/lefs/tech.lef /Data/lefs/hier_block.lef}
END
```

5. Generate the block-level xPGVs.

   Tcl command:

```
generate_xpgv -cell hier_block -output_dir xpgv -state_directory
dynamic_rail/core_125C_dynamic_1
```

   The log file will generate the power-grid view generation statistics, as shown in the following log file snippet:

```
** INFO: (VOLTUS_LGEN-3265):
Power Grid View Generation Statistics:
        # Total number of cells: 1
        # EARLY view created: 1 (100%)
        # IR view created: 1 (100%)
        # EM view created: 1 (100%)
Finished PGV Library Generator at 13:08:20 02/15/2019 (cpu=0:00:01,
real=0:01:01, peak mem=511.77MB)
```

6. In the second Tcl script, specify the design loading (top-level design data), power analysis, and rail analysis commands. Following is a snippet of the top-level rail analysis commands using the generated xPGV model:

Tcl command:

```
set_rail_analysis_mode \
    -accuracy hd \
    -method dynamic \
    -temperature 125 \
    -ignore_shorts true \
    -limit_number_of_steps false  \
    -use_em_view_list /Data//RAIL/submegacells.list \
    -snap_layer_for_current_taps \
    /Data//RAIL/megacell.collapse_tap_current_layer \
    -power_grid_library {/Data/hierpgv_dyn_accurate/All_merge.cl \
    ../xpgv/hier_block/hier_block.cl} \
    -report_power_in_parallel true \
    -flatten_xpgv_block_instances  use_def.txt
set_power_pads -format defpin -net VDD1
set_pg_nets -net VDD1 -force -voltage 1.21 -threshold 1.08
set_power_pads -format defpin -net VSS
set_pg_nets -net VSS -force -voltage 0.0 -threshold 0.12
set_rail_analysis_domain -name core -pwrnets VDD1 -gndnets VSS
analyze_rail -type domain -output dynamic_rail core
```

The following parameters are specific to the xPGV instantiation flow:

- `-power_grid_library` *`libraries`* - specifies the generated xPGV library and other PGV libraries being used in the flow.
- `-flatten_xpgv_block_instances` *`file`* - specifies a file that includes a list of block instances that use DEF instead of xPGV for power and rail analysis, that is, specify the instances that are not to be used as xPGV. This parameter allows you exclude DEFs for xPGV blocks. You can specify a few instances of the same block that will use DEF, and the others that will use xPGV at the top-level. If you do not specify this parameter, and if both DEF and xPGV are given for a block, xPGV takes precedence over DEF.

  Instance List File Format:

  `<block_name>  <block_instance_name|All|None>`

  Example

  For a design with three blocks, if you want to use the DEF and xPGV as following:

  Block_1 - > A1 (xPGV) & A2 (DEF)

  Block_2 - > B1 & B2 (xPGV for both instances)

Block_3 - > C1 & C2 (DEF for both instances)

The specified file will contain the following information:

```
Block_1 A2
Block_2 None
Block_3 All
```

Output

The xPGV output directory will contain the following information:

File Name	Description
`<cellname>.cl`	The name of the macro power-grid library is `<cellname>.cl`. It contains the physical and electrical representation of the hierarchical block that is required for rail analysis.
`<cellname>.report`	A report for the cell's power/ground net containing: ■ name of power and ground nets, voltage value for power pins ■ number of taps for the Early/EM/IR views
`<cellname>.summary`	A report containing capacitance, the current of analyzed nets, resistors, and the number of current taps and layers for the Early/EM/IR views.

The top-level analysis will generate the following:

- Static power reports and plots
- Dynamic currents and IR-drop plots

14

ESD Analysis

- Overview on page 342
- Data Requirements for ESD Effective Resistance Checks on page 344
- Specifying ESD Rules on page 346
  - Rule for computing clamp-to-clamp resistance on page 358
  - Rule for computing bump-to-clamp resistance on page 359
  - Rule for computing bump-to-clamp resistance (shorted clamps) on page 359
  - Rule for computing bump-to-bump resistance (shorted bumps) on page 359
  - Rule for computing bump-to-bump resistance (unshorted bumps) on page 359
  - Rule for computing bump-to-instance resistance on page 360
  - Rule for computing clamp-to-instance resistance on page 361
  - Rule for reporting isolated bump/clamps on page 361
  - EM Rule for all nets with zaps and sinks with all bumps shorted on respective nets on page 361
  - EM rule for zaps and sinks on specific bumps on specific nets on page 361
- Performing ESD Analysis on page 363
- Analyzing ESD Reports on page 367
- Viewing Effective Resistance Check Results on page 382
- CDM Based ESD Analysis on page 386

© 2013-2023 All Rights Reserved.

Overview

Electrostatic Discharge (ESD) is defined as the transient electric current between two objects at different electrostatic potentials causing an ESD event. Due to shrinking geometries of nanometer designs, devices are at a higher risk to ESD exposure and are prone to damage.

To prevent an ESD event, you can place ESD devices at power/ground pads to provide an efficient discharge path between one bump to another.

Calculation of effective resistance between a large pair of nodes is a difficult engineering problem with high computational complexity. Although there are graph-based search algorithms, they only identify the least resistance paths but not the effective resistance between power/ground bumps and ESD devices. Because of the presence of multiple alternate paths between bumps and ESD cells, the resistance of least resistive path will be considerably higher than the effective resistance.

The ESD Analysis (`analyze_esd_network`) flow in Voltus allows you to perform different types of effective resistance checks, such as bump to clamp, bump to bump, and clamp to clamp, and current density analysis during an ESD zap event. This command checks whether every power or ground bump has an ESD device, and the device is placed in such a way that it does not violate the effective resistance or current density limit. The objective of ESD Checker is to easily identify the violations caused by high effective resistance or the current density above the specified threshold value. It also performs isolation checks, that is, if any bump does not have a path to an ESD cell (indicating existence of isolated bumps), or if a clamp is not connected to any bump (indicating existence of isolated clamps).

In addition, you can perform the Charged Device Model (CDM) based ESD analysis (`analyze_esd_voltage`) that allows you to report violations where potential difference across driver/receiver is greater than the given threshold.

ESD analysis supports the following features:

- **bump to clamp effective resistance check** - Effective resistance is computed between each bump to all ESD devices in the design, or from each ESD device to all bumps in the design (if the report method is reverse). It also supports ESD analysis from shorted bumps to each clamp. Therefore, it allows you to perform two types of checks,

© 2013-2023 All Rights Reserved.

one is shorted bumps to all clamps and second is each bump to each clamp. Both checks will be done with a different threshold value.

- **bump to bump effective resistance check** - Effective resistance is computed for all paths - from a power bump to an ESD device to a ground bump. It calculates and reports the effective least resistance path for each bump of a power net to each bump of a ground net through clamps. This rule works only for domain-based or multiple domain-based analysis since it calculates the least effective resistance path between power and ground nets taking into account multiple clamps in the path.

  The ESD analysis flow also supports bump to bump ESD checks over a user-specified radius. The software can calculate bump to bump resistance only for bumps within the specified radius. This feature allows you to:

  - select and check bumps only in the given radius. This reduces the number of bumps and ESD output file size, and improves the runtime.
  - check for weak ESD cells within radius, and skip/fix them.
  - ignore uninterested large paths.

- **clamp to clamp effective resistance check** - Effective resistance is computed between all clamps for each net.
- **bump to instance effective resistance check** - Effective resistance is computed between every bump to instance, and will check if the resistance value is less than that of the same bump to clamp cell. If the resistance value of bump to clamp is more than the resistance value of bump to instance, then a fail status is given.
- **clamp to instance effective resistance check** - Effective resistance is computed between every instance to nearest clamp. If the resistance value of instance to clamp is more than the threshold value, then a fail status is given. This report is important to perform CDM check.
- **Isolated bump/clamp check** - Checks if there are any bumps that are not protected by an ESD cell, or checks if there are any clamps that are not connected to any bumps.
- **Current density/EM analysis** - Checks whether current densities are not exceeding limits that cause metal melting or fusion.

Important

You can perform ESD analysis using the `read_def` or `specify_def` flow. For information on running `analyze_esd_network` with `specify_def`, refer to the ESD Analysis with specify_def section in this chapter.

© 2013-2023 All Rights Reserved.

Data Requirements for ESD Effective Resistance Checks

The following table lists the required inputs for performing ESD resistance checks in Voltus:

Input	Description
`ESD cell PGVs`	ESD cells are tagged using the `-esd_cells` or `-esd_cells_list_file` parameter of the `set_advanced_pg_library_mode` command, and ESD cell pins are tagged using the `-esd_pin_list` parameter of the `set_advanced_pg_library_mode` command. The ESD cell PGV contains cell pin information and ESD device electrical characteristics.
`ESD Rule File`	A text file to specify a set of rules that define all the ESD checks to be performed along with the effective resistance threshold limit. Some of the characteristics of the rule file are: ■ All rules should be in a single file ■ A single rule specifies the type of check and the pass/fail criteria ■ The various rule types are: ❑ `clamp2clamp` - computes clamp to clamp resistance ❑ `bump2clamp` - computes bump to clamp resistance ❑ `bump2bump` - computes bump to bump resistance ❑ `bump2instance` - computes bump to instance resistance ❑ `clamp2instance` - computes clamp to instance resistance ❑ `em` - checks for current density ❑ `connectivity` - checks for isolated bumps or clamps

© 2013-2023 All Rights Reserved.

Input	Description
ESD EM Technology File	This file has the same format as the ICT EM file. It contains the current density check limits for each metal layer.
Technology PGV	The technology library contains the extraction technology file, area capacitance specification, decap/filler/powergate cells, and tech view for all the cells. ESD analysis requires only LEF-based PGVs.
Design DEFs	The DEF files contain information about cell placement, power/ground grid, and signal routing.

© 2013-2023 All Rights Reserved.

Specifying ESD Rules

Voltus allows you to specify all the rules for computing effective resistance in a single rule file. The rule file is specified using the `-config_file` parameter of the `analyze_esd_network` command.

The rule syntax is case insensitive. The syntax of the rule file is:

Syntax for Resistance Checks

```
rule rule_name
type {clamp2clamp | bump2bump | bump2clamp | connectivity | bump2instance |
clamp2instance}
power -nets net1 net2 net3 ... -short_bumps {true | false}
ground -nets net1 net2 net3 ... -short_bumps {true | false}
nets <list_of_nets|all>
nets -from <list_of_nets|all> -to <list_of_nets|all>
bumps <list_of_bumps>
bumps –from <list_of_bumps> –to <list_of_bumps>
clamp_types -from type1 type2 type3 ... -to type1 type2 type3 ...
clamp_cells -from cell1 cell2 cell3 -to cell1 cell2 cell3
current value
em_results {true | false}
reff_threshold value
reporting {reverse | both}
check_isolated_bumps {true | false}
check_isolated_clamps {true | false}
lerp_threshold value
skip_domains net1:net2
instance_list_file inst_filename [-include|exclude]
cell_list_file cell_filename [-include|exclude]
esd_device_check {true | false}
effective_calculation {true | false}
parallel_clamp_threshold value
reff_report_threshold value
report_line_threshold value
end rule
```

Syntax for EM Checks

```
rule rule_name
type em
```


© 2013-2023 All Rights Reserved.

```
setup [-ahdl_include Verilog-A_file] [-spice_include SPICE_file] [-iv_lib {true
| false}] [-cross_domain {true | false}]
source -type {power | ground} -nets net1 net2 net3... -short_bumps {true | false}
-report_threshold value
sink -type {power | ground} -nets net1 net2 net3... -short_bumps {true | false}
-report_threshold value
clamp_cells cell1 cell2 cell3
clamp_types type1 type2 type3...
nets -from <list_of_nets|all> -to <list_of_nets|all>
bumps -from <list_of_bumps> -to <list_of_bumps>
current value
end rule
```

Parameters

Parameter Name	Description
`arc_reff_threshold value`	Allows to filter clamps, that is, only clamps with an effective resistance to bump value less than the specified arc threshold value will be considered for the bump-to-bump resistance check.
`bump_region_xy_size x y`	Specifies the length (x) and width (y) of the bump region in µm for radius-based bump-to-bump ESD checks.
`bumps -from <list_of_bumps> -to <list_of_bumps>`	
`bumps <list_of_bumps>`	
	`-reporting {original | reverse | both}`
	`-report_threshold value`

© 2013-2023 All Rights Reserved.

Parameter Name	Description

	Allows you to specify the bumps for which the rule is run. bumps is mutually exclusive with the `power`, `ground`, `nets` parameters.
	
	The following are the two use models:
	
	■ specify the bump names for the `bump2clamp` and `clamp2clamp` rule types:
	
	`bumps <list_of_bumps>`
	
	■ specify the bump names for the `em` and `bump2bump` rule types:
	
	`bumps -from <list_of_bumps> -to <list_of_bumps>`
	
	For `bumps`, you can specify the following parameters:
	
	■ `-from` and `-to` parameters to indicate the list of starting and ending bumps. You must specify the bumps within the same net. This parameter is applicable to the `em` and `bump2bump` rule types.
	
	■ `-reporting {original | reverse | both}` specifies the reporting format for sorting of the rule. The default value is `original`.
	
	■ `-report_threshold value` specifies to report EM violations above the threshold value. The value is the ratio of current (avg) to the respective EM limit. The default value is `0.9`. This parameter is applicable to the `em` and `bump2bump` rule types.

`cell_list_file cell_filename [-include|exclude]`

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	Excludes or includes the list of cells given in the specified file. If this parameter is not specified, ESD analysis is performed on all cells in the design. The specified cell list text file contains a list of cells. Each cell name must reside on a separate line in the file. The following is the format of the specified file: *cell1* *cell2*
`check_isolated_bumps {true | false}`	
	Specifies to check isolated bumps. The default value is `false`. This parameter is applicable only to the `connectivity` rule type.
`check_isolated_clamps {true | false}`	
	Specifies to check isolated clamps. The default value is `false`. This parameter is applicable only to the `connectivity` rule type.
`clamp_cells -from` *`cell1 cell2 cell3`* `-to` *`cell1 cell2 cell3`*	
	Specifies the clamp cells for which the rule is run. `clamp_cells` and `clamp_types` are mutually exclusive. For `clamp2clamp`, you need to specify the `-from` and `-to` parameters to specify the starting and ending clamp cells. For the other types of checks, you need to specify individual clamp cells. The `-from` and `-to` parameters are not required.
`clamp_types -from` *`type1 type2 type3`* ... `-to` *`type1 type2 type3`* ...	

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	Specifies the type of clamp for which the rule is run. The different clamp types are stored in PGV. `clamp_cells` and `clamp_types` are mutually exclusive. For `clamp2clamp`, you need to specify the `-from` and `-to` parameters to specify the starting and ending clamp types. For the other types of checks, you need to specify individual clamp cell types. The `-from` and `-to` parameters are not required.
`current` *`value`*	Specifies the zap current value. This parameter is applicable to the `em` and `bump2bump` rule types. For `bump2bump`, this parameter must be specified when the IV curve PGV models are used to perform non-linear ESD analysis.
`effective_calculation {true | false}`	
	Enables shorting of clamps on the specified nets and performs effective resistance check from each bump to the shorted clamps. It also reports the number of shorted ESD clamps that are connected in parallel to each bump. This parameter is applicable to the `bump2clamp` type.
`em_results {true | false}`	
	EM analysis is performed when IV curve PGV based ESD analysis is performed. You can use this option to report those EM results if required. It is recommended to set `em_results` to `false` for saving disk space. The default value is `false`. This parameter is applicable to the `bump2bump` rule type.
`esd_device_check {true | false}`	
	Specifies to do bump2bump analysis till taps. For this analysis, it is recommended to use ESD PGV generated using Voltus-Fi, and tag the ESD devices inside the PGV. The default value is `false`. This parameter is applicable to the `bump2bump` type.

© 2013-2023 All Rights Reserved.

Parameter Name	Description
`ground -nets {net1 net2 net3... | all} -short_bumps {true | false}`	Specifies the list of ground nets to be analyzed. You can specify `all` to include all ground nets. For `bump2clamp`, you can specify `-short_bumps true` to do a resistance check from a shorted bump to each clamp. If `-short_bumps` is set to `false`, resistance check is done for every bump to every clamp. You can have two rules for `bump2clamp`, one with `-short_bumps` set to `true` and the other with `-short_bumps` set to `false`. Both these rules will have a different value for `reff_threshold`.
`instance_list_file inst_filename [-include|exclude]`	Excludes or includes the list of instances given in the specified file. If this parameter is not specified, ESD analysis is performed on all instances in the design. The specified instance list text file contains a list of instances. Each instance name must reside on a separate line in the file. The following is the format of the specified file: *instance1* *instance2*
`lerp_threshold value`	Specifies the threshold for least effective resistance path between bump2bump, beyond which failure will be reported. If a threshold value is not specified, all rules will be passed. This parameter is applicable to the `bump2bump` type.
`nets <list_of_nets|all>`	
`nets -from <list_of_nets|all> -to <list_of_nets|all>`	

© 2013-2023 All Rights Reserved.

Parameter Name	Description

	`-reporting {original | reverse | both}`
	
	`-short_bumps {true | false}`
	
	`-report_threshold value`
	
	`-short_net_bumps list_of_nets`
	
	Allows you to specify the nets for which ESD analysis is to be performed. nets is mutually exclusive with the `power`, `ground`, `bumps` parameters. This parameter gives you the flexibility to specify any nets for the `-from` and `-to` options, and does not confine to the power nets for the `-from` list and the ground nets for the `-to` list.
	
	The following are the two use models:
	
	■ specify the net names for the `bump2clamp` and `clamp2clamp` rule types:
	
	`nets <list_of_nets|all>`
	
	■ specify the net names for the `em` and `bump2bump` rule types:
	
	`nets -from <list_of_nets|all> -to <list_of_nets|all>`

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	For `nets`, you can specify the following options: ■ `-from` and `-to` parameters to indicate the list of starting and ending nets. You can either specify the list of nets or `all` to include all nets. ■ `-short_net_bumps` specifies to short bumps of the specified nets only. ■ `-short_bumps` specifies to short all bumps for each net. The default value is `false`. **Note:** `-short_bumps` and `-short_net_bumps` are mutually exclusive. ■ `-reporting {original | reverse | both}` specifies the reporting format for sorting of the rule. The default value is `original`. ■ `-report_threshold` *`value`* specifies to report EM violations above the threshold value. The value is the ratio of current (avg) to the respective EM limit. The default value is `0.9`. This parameter is applicable to the `em` and `bump2bump` rule types.
`parallel_clamp_threshold` *`value`*	Specifies the minimum number of shorted clamps between bumps. The number of shorted clamps should be greater than or equal to the given threshold value, beyond which failure will be reported. This is an optional parameter that is used with the `effective_calculation` parameter. This parameter is applicable to the `bump2clamp` type.
`parallel_reff_threshold` *`value`*	Specifies the resistance threshold to report the equivalent parallel resistance through multiple clamps in the bump-to-bump path. The equivalent parallel resistance is calculated by considering only those clamps with effective resistance less than the specified arc_reff_threshold.
`pass_criteria {all | one}`	

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	Specifies the Pass/Fail criteria for bumps/clamps/nets. ■ `all`: criterion for a bump/clamp/net to pass is that all paths from that bump/clamp/net passes. ■ `one`: criterion for a bump/clamp/net to pass is if atleast one path from that bump/clamp/net passes. The default value is `all`.
`path_reff_threshold` *`value`*	Specifies the resistance threshold for the bump-to-bump path's effective resistance to pass. A pass status is given when the effective resistance is below the specified threshold value, and a fail status is given when the effective resistance is above the specified threshold value.
`power -nets {`*`net1 net2 net3...`*` | all} -short_bumps {true | false}`	Specifies the list of power nets to be analyzed. You can specify `all` to include all power nets. For `bump2clamp`, you can specify `-short_bumps true` to do a resistance check from a shorted bump to each clamp. If `-short_bumps` is set to `false`, resistance check is done for every bump to every clamp. You can have two rules for `bump2clamp`, one with `-short_bumps` set to `true` and the other with `-short_bumps` set to `false`. Both these rules will have a different value for `reff_threshold`.
`reff_report_threshold` *`value`*	Specifies the minimum allowed threshold for the resistance check information written to the report file. The default value is `0`. This parameter is applicable to all effective resistance check rule types.
`report_only_pg_pairs {true | false}`	

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	Reports bump-to-bump resistance for the power-to-ground pair. When this parameter is set to `false`, reports power-to-power and ground-to-ground bump-to-bump resistance with the power-to-ground pair bump-to-bump resistance. The default value is `true`.
`report_line_threshold` *`value`*	
	Controls the size of the effective resistance check report by limiting the number of lines written to the report file. This parameter is useful in limiting the report size when you specify a very low resistance check threshold. The default value is `500`. This parameter is applicable to all effective resistance check rule types.
`reporting {original | reverse | both}`	

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	Specifies the reporting format that can be used for sorting of the rule. This parameter gives you the flexibility to report the effective resistance in a different sequence. By default, it reports as per the rule type. The options of this parameter are: ■ `reverse` - reports effective resistance from clamp to bump. It checks if any clamp is failing from clamp to bump. ■ `original` - reports effective resistance from bump to clamp. It checks if any bump is failing from bump to clamp. This is the default value. ■ `both` - reports effective resistance both from bump to clamp and clamp to bump. Example for rule type `bump2clamp`: ■ When reporting is set to `original`, it reports effective resistance from bump to clamp. ■ When reporting is set to `reverse`, it reports effective resistance from clamp to bump. ■ When reporting is set to `both`, it reports effective resistance from bump to clamp and clamp to bump.
`rule` *`rule_name`*	Specifies the name of the rule.
`reff_threshold` *`value`*	Specifies the resistance threshold for the rule to pass (rule pass criteria). A pass status is given when the effective resistance is below the specified threshold value, and a fail status is given when the effective resistance is above the specified threshold value. This parameter is applicable to `clamp2clamp`, `bump2bump`, and `bump2clamp` types.
`sep_short {true | false}`	Specifies to short only the pins of the clamp pair for which the effective resistance is being calculated. By default, the tool shorts pins for all clamps. The default value is `false`.

© 2013-2023 All Rights Reserved.

Parameter Name	Description
`setup [-ahdl_include Verilog-A_file] [-spice_include SPICE_file] [-iv_lib {true | false}] [-cross_domain {true | false}]`	The `setup` parameter is used to enable zap analysis on power or ground nodes of a cross-domain pair, wherein the driver-receiver pair have different power or ground domains. It generates current density report for each resistor on the grid for cross-domains with multiple nets between the source and sink bumps. ■ `-ahdl_include`, `-spice_include`, and `-iv_lib` are mutually exclusive. Use one of these parameters to provide the non-linear connections (ESD models) between the power and ground pins, instead of directly providing ESD Ron/Roff resistor values. ■ `-iv_lib` means inherit the model created in PGV run. ■ `-cross_domain true` indicates that all the domain nets that have ESD connections with the source and sink net will be taken into account during zap analysis. The default value of `false` indicates that the tool simulates only the network of the source and sink net.
`skip_reff {true | false}`	Specifies to calculate only the least effective resistance path (LERP). When set to `true`, the effective resistance values will not be reported. The default value is `false`.
`sink -type {power | ground} -nets {net1 net2 net3... | all}-short_bumps {true | false} -report_threshold value`	
`source -type {power | ground} -nets {net1 net2 net3... | all} -short_bumps {true | false} -report_threshold value`	

© 2013-2023 All Rights Reserved.

Parameter Name	Description
	For `em` check, source (the point where current is injected), and sink (the point from where the current comes out) details are provided. For both source and sink, you need to specify the type of net (power/ground), the nets to be analyzed, and whether bumps are to be shorted. You can specify EM rules for all nets or specific nets with zaps and sinks on bumps. If `-short_bumps` is set to `true`, current is injected in the shorted node (vdd), and comes out of the shorted node (vss). In the `em` rule check, the software runs a static simulation and checks current through each metal line against the ESD EM techfile.
`skip_domains` *`net1`*`:`*`net2`*	Specifies a pair of nets separated by ":" that are skipped during bump2bump analysis. This parameter is applicable to the `bump2bump` type.
`type {clamp2clamp | bump2bump | bump2clamp | bump2instance | clamp2instance | connectivity | em}`	Specifies the rule type.

ESD Rule File Examples

The following are examples of different rule types.

Rule for computing clamp-to-clamp resistance

```
Rule c2c_rule1
type clamp2clamp
power -nets VDD
ground -nets VSS
clamp_types -from "CLAMPTYPE1" -to  "CLAMPTYPE2"
OR clamp_cells -from "CLAMPCELL1" -to "CLAMPCELL2"
reff_threshold 3
end rule
```


© 2013-2023 All Rights Reserved.

Rule for computing bump-to-clamp resistance

```
Rule b2c_rule2
type bump2clamp
clamp_cells "DIODE1 CLAMP1"
power -nets all -short_bumps true/false
ground -nets all -short_bumps true/false
reporting both
reff_threshold 2.5
report_line_threshold 500
reff_report_threshold 10
end rule
```

Rule for computing bump-to-clamp resistance (shorted clamps)

```
Rule b2c_rule
type bump2clamp
clamp_cells "ESDCLAMP"
ground -nets NET1 -short_bumps false
reporting both
reff_threshold 3.5
pass_criteria all
effective_calculation true
parallel_clamp_threshold 2
end rule
```

Rule for computing bump-to-bump resistance (shorted bumps)

```
Rule rule_shorted_bump_device
type bump2bump
power -nets all -short_bumps true
ground -nets all -short_bumps true
lerp_threshold 10
skip_domains VSS0:VDD
esd_device_check true
end rule
```

Rule for computing bump-to-bump resistance (unshorted bumps)

```
Rule rule_unshorted_bump_device
type bump2bump
power -nets all -short_bumps false
ground -nets all -short_bumps false
```


© 2013-2023 All Rights Reserved.

```
lerp_threshold 10
skip_domains VDD0:VSS
esd_device_check true
end rule
```

Rule for computing radius-based bump-to-bump resistance check

```
Rule b2b_rule_region
type bump2bump
power -nets all
ground -nets all
parallel_reff_threshold 10
arc_reff_threshold 5
lerp_threshold 5
path_reff_threshold 10
bump_region_xy_size 200 200
end Rule
```

Rule for computing bump-to-bump resistance check for IV curve PGV models

```
Rule b2b_rule_non_linear_all_net
type bump2bump
nets -from all -to all -reporting both -report_threshold 0.8
setup -ahdl_include /DATA/models.va
current 1.5
em_results true
end rule
```

Rule for computing bump-to-instance resistance

```
Rule b2i_rule1
type bump2instance
power -nets VDD
ground -nets VSS
instance_list_file inst_list -include
cell_list_file cell_list -include
end Rule
```

The following is the content of the `cell_list` file:

```
NAND2
NAND4
```

The following is the content of the `inst_list` file:

© 2013-2023 All Rights Reserved.

```
INST/i_127
INST/i_131
```

Rule for computing clamp-to-instance resistance

```
Rule c2i_rule1
type clamp2instance
power -nets VDD
ground -nets VSS
clamp_cells "cell_32x64"
reff_threshold 20
pass_criteria  all
end Rule
```

Rule for reporting isolated bump/clamps

```
Rule connectivity
type connectivity
clamp_cells CLAMP_A CLAMP_B
power -nets all
ground -nets all
check_isolated_bumps true
check_isolated_clamps true
end rule
```

EM Rule for all nets with zaps and sinks with all bumps shorted on respective nets

```
Rule em_rule_all_nets
type em
source -type power -nets all -short_bumps true -report_threshold 2
sink -type ground -nets all -short_bumps true -report_threshold 1
clamp_types <types of ESD cells>
current 1.5
end rule
```

EM rule for zaps and sinks on specific bumps on specific nets

```
Rule em_rule_multi_bumps
type em
source -type power -nets VDD -short_bumps false -bumps VDDvsrc2
sink -type ground -nets VSS -short_bumps false -bumps "VSSvsrc1"
clamp_cells ESD_CELL1
```


© 2013-2023 All Rights Reserved.

```
current 1.5
end rule
```

EM rule for zaps and sinks for cross-domain analysis with multiple nets between the source and sink bumps

```
Rule em_rule_non_linear
type em
setup -ahdl_include ./DATA/models.va -cross_domain true
source -type power -nets all -short_bumps true -report_threshold 1
sink -type ground -nets all -short_bumps true -report_threshold 1
clamp_cells clamp1 clamp2 clamp3 clamp4 clamp5 clamp6
current -6
end rule
```


© 2013-2023 All Rights Reserved.

Performing ESD Analysis

This section describes the various inputs and outputs of the ESD analysis flow.

Required Input Files

- ESD Cell PGVs
- Design data (LEF, netlist, DEF)
- ESD Rule File
- ESD EM technology file
- Technology PGV

Procedure

To perform ESD analysis, complete the following steps:

1. Create PGVs for ESD cells.

   Tcl command:

```
read_lib -lef
../../design/BONDPAD52.lef ../../design/giolib045_modified.lef
set_pg_library_mode -celltype techonly -ground_pins { VSS } -
power_pins { VDD 1 } -lef_layermap ./lefdef.map -
extraction_tech_file ./qrcTechFile
set_advanced_pg_library_mode -esd_cells_list_file {esd_cell_list} -
esd_parameters_file param_file.txt -esd_pin_list {VDD VSS VDDESD
VSSESD}
generate_pg_library -output esd_pgv
```

   Here, the following parameters are specific to the ESD analysis flow:

   - `-esd_cells_list_file` specifies the ESD clamp cells that are to be analyzed.
   - `-esd_pin_list` specifies the ESD clamp pins that are to be analyzed. Effective resistance analysis will be done only if the nets are connected to these pins.
   - `-esd_parameters_file` specifies the electrical characteristics of the clamp cells.

   The format of the `-esd_cells_list_file` file is:

```
<clamp cell name>  <clamp type>
esd_decap_vdd       CLAMPIO #clamp type can be used to group clamps
esd_decap_core      CLAMPCORE
```


© 2013-2023 All Rights Reserved.

The format of the `-esd_parameters_file` file is:

```
Cell <cell name>  <Ron> <Roff> <power_pin> <ground_pin> <voltage>
Cell esd_decap_vdd 0.9  10000    vdd_esd     vss_esd      0.4
```

2. Load the design data.

   Tcl command:

```
read_lib -lef $lefs
specify_def {/data/def/top_final.def  /data/def/dmu.def}
```

3. Define the rail analysis mode.

   Tcl command:

```
set_rail_analysis_mode \
    -method                     static \
    -accuracy                   hd \
    -power_grid_library       ./data/pgv_dir/esd_pgv/techonly.cl \
    -ignore_shorts            true\
    -enable_rlrp_analysis       true \
    -ict_em_models ./data/ESD_em.ict \
```

4. Specify the power/ground net voltages, and define the location of the voltage sources for each power/ground net.

   Tcl command: .

```
set_pg_nets -net VDD -voltage 0.9 -threshold 0.8 -force
set_pg_nets -net VSS -voltage 0 -threshold 0.1 -force
set_power_pads -format xy -net VDD -file ./data/top_VDD.pp
set_power_pads -format xy -net VSS -file ./data/top_VSS.pp
```

5. Specify the rail analysis power domain.

   Tcl command:

```
 set_rail_analysis_domain -name PD -pwrnets { VDD } -gndnets { VSS }
```

6. Run ESD analysis.

© 2013-2023 All Rights Reserved.

Tcl command:

```
analyze_esd_network PD -config_file rule1.txt -output ESD_analysis
-type domain -use_power_pad true
```

Instead of shorting all nodes of the lowest ESD cell pin layer, you can specify the –`clamp_pin_short_file` <*`pin_shorting_file`*> parameter to specify the list of pin layers to be shorted for bump-to-clamp and clamp-to-clamp resistance checks.

The format of the `-clamp_pin_short_file` file is:

```
#Cell specification
CELL <cell name> <pin> <layer>

#Instance specification
INST <instance name> <pin> <layer>
```

The log file will generate the ESD analysis statistics, as shown in the following log file snippet:

```
Begin Processing Rule esd_isolation
Running rule esd_isolation...
Processing net VDD...
Processing net VSS...
Generating reports for rule esd_isolation...
Ended Processing Rule esd_isolation: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=326.27MB/2086.20MB/822.55MB)
Generating summary of all rules...
Ended Running ESD Rule Analysis: (cpu=0:00:00, real=0:00:00,
mem(process/total/peak)=326.28MB/2086.20MB/822.55MB)
Rail Analysis Statistics:
Warning messages:      0
Error messages:        0
Rail Analysis completed successfully.
Finished Rail Analysis at 00:26:44 11/21/2018 (cpu=0:00:12, real=0:00:22, peak
mem=822.55MB)
```

The `analyze_esd_network` command creates a state directory inside the <*`output_directory`*> specified with the command. The `ESD` directory is created inside this state directory. The `ESD` directory contains a detailed report for each rule along with a summary report for all the rules. Following is an example of the directory structure:

`ESD_analysis/top_25C_esd_1/ESD`

The `ESD` folder contains the following files:

- b2c_rule_CORE.reff

© 2013-2023 All Rights Reserved.

- b2c_rule_all_clamps.reff
- c2c_rule_core.reff
- esd_isolation.reff
- em_rule_all_nets_VDD_short_VSS_short
- summary.rpt

Output

The `ESD` directory will contain the following files:

File Name	Description
`summary.rpt`	A report that gives a summary of all rules and their pass/fail status.
`<rule_name>.reff`	`<rule_name>.reff` files that contain detailed reports of each type of rule.
`<em_rule_name>_<domain>/<net>`	This folder contains current density detailed results.

© 2013-2023 All Rights Reserved.

Analyzing ESD Reports

The following reports are generated for ESD Analysis in Voltus:

- Effective Resistance Check Reports
- Current Density Check Report

Effective Resistance Check Reports

The following reports are generated for effective resistance checks:

- Bump-to-Clamp Report
- Clamp-to-Clamp Report
- Bump-to-Bump Report
- Clamp-to-Instance Report
- Bump-to-Instance Report
- Isolated Bumps or Clamps Report
- Summary Report

The report formats and samples are provided below.

Report of Bump-to-Clamp Type Rule

Sample filename: b2c_rule1.reff

Format:

```
#<bump> <x y (um)> <layer>
#- <pass/fail> <R ohm> <threshold ohm> <clamp cell> <x y (um)> <pin> <layer> <clamp instance>
```

Format Description:

`bump` specifies the bump name.

`x y` specifies the location of the bump.

`layer` specifies the metal layer on which the bump is located.

© 2013-2023 All Rights Reserved.

`pass/fail` specifies the status of the clamp instance - a pass status is given when the bump to clamp effective resistance is below the specified threshold value and a fail status is given when the effective resistance is above the specified threshold value.

`R ohm` specifies the bump to clamp effective resistance.

`threshold` specifies the minimum allowed effective resistance between a power or ground bump and an ESD device pair.

`clamp cell` specifies the name of the ESD clamp cell.

`x y` specifies the location of the clamp cell pin node. If there are multiple nodes on the pin and are shorted, this is the location of the shorted pin.

`pin` specifies the name of the ESD pin.

`layer` specifies the metal layer of the pin whose x,y location is given.

`clamp instance` specifies the name of the ESD clamp instance.

Sample Report:

```
-  #NET VSS
-  #Reporting from bump to clamp
-  #<bump> <x y (um)> <layer>
-  #- <pass/fail> <R ohm> <threshold ohm> <clamp cell> <x y (um)> <pin> <layer> <clamp instance>
-  VSSvsrc1  (  485.42999 668.21997  ) METAL8
-  - PASS 1.3283726 2.5000000 INVXL (  676.50000 470.95999  ) VSS METAL1 DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_10645
-  - PASS 1.3836780 2.5000000 INVXL (  678.47998 561.67999  ) VSS METAL1 DTMF_INST/ULAW_LIN_CONV_INST/i_9748
-  - PASS 1.4263393 2.5000000 INVXL (  677.82001 470.95999  ) VSS METAL1 INST/TDSP_CORE_INST/TDSP_CORE_GLUE_INST/i_9801
-  - PASS 1.4321177 2.5000000 INVXL (  677.15997 420.56000  ) VSS METAL1 INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9664
-  - PASS 1.5168804 2.5000000 INVXL (  674.52002 460.88000  ) VSS METAL1 INST/TDSP_CORE_INST/TDSP_CORE_GLUE_INST/i_9805
-  - FAIL 4.2026470 2.5000000 INVXL (  425.04001 531.44000  ) VSS METAL1 INST/TDSP_CORE_INST/DATA_BUS_MACH_INST/i_9559
-  - FAIL 4.2825665 2.5000000 INVXL (  943.79999 370.16000  ) VSS METAL1 INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/i_9752
```

Format (bump to shorted clamp**)**:

```
#<bump> <x y (um)> <layer>
#- <pass/fail> <R ohm> <threshold ohm> <parallel clamps> <pin>
```

Format Description:

`bump` specifies the bump name.

`x y` specifies the location of the bump.

`layer` specifies the metal layer on which the bump is located.

© 2013-2023 All Rights Reserved.

`pass/fail` specifies the status of the shorted clamp instance - a pass status is given when the bump to shorted clamp effective resistance is below the specified threshold value (`reff_threshold`) and the number of shorted clamps is greater than or equal to the specified threshold value (`parallel_clamp_threshold`). Similarly, a fail status is given when the bump to shorted clamp effective resistance is above the specified threshold value (`reff_threshold`) or the number of shorted clamps is less than the specified threshold value (`parallel_clamp_threshold`).

`R ohm` specifies the bump to clamp effective resistance.

`threshold` specifies the minimum allowed effective resistance between a power or ground bump and an ESD device pair.

`parallel clamps` specifies the number of shorted clamps.

`pin` specifies the name of the ESD pin.

Report of Clamp-to-Clamp Type Rule

Sample filename: c2c_rule1.reff

Format:

```
#<clamp cell> <x y (um)> <pin> <layer> <clamp instance>
#- <pass/fail> <R ohm> <threshold ohm> <clamp cell> <x y (um)> <pin> <layer> <clamp instance>
```

Format Description:

`clamp cell` specifies the from-clamp cell name.

`x y` specifies the location of the from-clamp cell.

`pin` specifies the name of the ESD pin of the from-clamp cell.

`layer` specifies the metal layer of the ESD pin of the from-clamp cell.

`clamp instance` specifies the name of the clamp instance.

`pass/fail` specifies the status of the to-clamp instance - a pass status is given when the clamp to clamp effective resistance is below the specified threshold value and a fail status is given when the effective resistance is above the specified threshold value.

`R ohm` specifies the clamp to clamp effective resistance.

© 2013-2023 All Rights Reserved.

`threshold` specifies the minimum allowed effective resistance between the from-clamp and to-clamp cells.

`clamp cell` specifies the to-clamp cell name.

`x y` specifies the location of the to-clamp cell.

`pin` specifies the name of the ESD pin of the to-clamp cell.

`layer` specifies the metal layer of the ESD pin of the to-clamp cell.

`clamp instance` specifies the name of the to-clamp instance.

Sample Report:

```
-  #NET VDD
-  #<clamp cell> <x y (um)> <pin> <layer> <clamp instance>
-  #- <pass/fail> <R ohm> <threshold ohm> <clamp cell> <x y (um)> <pin> <layer> <clamp instance>
-  ESD_CELL1 (  289.73999 72.800003  ) VDD METAL1 DTMF_INST/DIGIT_REG_INST/i_10629
-  - PASS 1.1898331 3.0000000 INVXL (  385.44000 92.959999  ) VDD METAL1 DTMF_INST/RESULTS_CONV_INST/i_10636
-  - PASS 1.4248223 3.0000000 INVXL (  388.73999 92.959999  ) VDD METAL1 DTMF_INST/RESULTS_CONV_INST/i_10635
-  - PASS 2.0081453 3.0000000 INVXL (  168.30000 72.800003  ) VDD METAL1 DTMF_INST/DIGIT_REG_INST/i_10630
-  - FAIL 4.4588715 3.0000000 INVXL (  948.41998 496.16000  ) VDD METAL1 INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/i_9616
-  - FAIL 4.5367170 3.0000000 INVXL (  956.34003 566.71997  ) VDD METAL1 INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/i_9604
```

Report of Clamp to Instance Rule

Sample filename: `c2i_core.reff`

Format:

```
#Failed instance-ESD resistance above Threshold sorted from large to small
#<clamp cell> <x y (um)> <pin> <layer> <clamp instance>
#- <pass/fail> <R ohm> <threshold ohm> <cell> <x y (um)> <pin> <layer> <instance>
```

Format Description:

`clamp cell` specifies the clamp cell name.

`x y` specifies the location of the clamp cell.

`pin` specifies the name of the ESD pin of the clamp cell.

`layer` specifies the metal layer on which the pin is located.

© 2013-2023 All Rights Reserved.

`clamp instance` specifies the name of the clamp instance.

`pass/fail` specifies the status of the instance - a pass status is given when the clamp to instance effective resistance is below the specified threshold value and a fail status is given when the effective resistance is above the specified threshold value.

`R ohm` specifies the clamp to instance effective resistance.

`threshold` specifies the minimum allowed effective resistance between the clamp and instance.

`cell` specifies the cell name.

`x y` specifies the location of the cell.

`pin` specifies the name of the cell pin.

`layer` specifies the metal layer on which the pin is located.

`instance` specifies the name of the instance.

Sample Report:

```
#NET VDD

#Failed instance-ESD resistance above Threshold=4.5 sorted from large to small

#<clamp cell> <x y (um)> <pin> <layer> <clamp instance>
#- <pass/fail> <R ohm> <threshold ohm> <cell> <x y (um)> <pin> <layer> <instance>
clamp_celll (  2186.1951 1017.8250  ) VDD P_SOURCE_DRAIN inst0
- FAIL 54.118763 4.5000000 NAND2XLTH (  2805.8799 2570.2400  ) VDD M1 inst1830
- FAIL 54.118580 4.5000000 NAND2XLTH (  2807.5601 2570.2400  ) VDD M1 inst1763
- FAIL 54.118032 4.5000000 NAND3XLTH (  2804.0601 2570.2400  ) VDD M1 inst1731
```

Report of Bump to Instance Rule

Sample filename: `b2i_core.reff`

Format:

```
#NET VDD
#Failed instance with closest B2C > closest B2I
#<clamp cell> <x y (um)> <pin> <layer> <clamp instance> <bump> <x y (um)> <layer>
#- <pass/fail> <B2C ohm> <B2I ohm> <cell> <x y (um)> <pin> <layer> <instance>
```

Format Description:

© 2013-2023 All Rights Reserved.

`clamp cell` specifies the clamp cell name.

`x y` specifies the location of the clamp cell.

`pin` specifies the name of the ESD pin of the clamp cell.

`layer` specifies the metal layer on which the clamp cell is located.

`clamp instance` specifies the name of the clamp instance.

`bump` specifies the bump name.

`x y` specifies the location of the bump.

`layer` specifies the metal layer on which the bump is located.

`pass/fail` specifies the status of the instance - a pass status is given when the bump to clamp is below the resistance value of bump to instance, and a fail status is given when the resistance value of bump to clamp is more than the resistance value of bump to instance.

`B2C ohm` specifies the bump to clamp effective resistance.

`B2I ohm` specifies the bump to instance effective resistance.

`cell` specifies the cell name.

`x y` specifies the location of the cell.

`pin` specifies the name of the cell pin.

`layer` specifies the metal layer on which the cell is located.

`instance` specifies the name of the instance.

Sample Report:

```
#NET VDD

#Failed instance with closest B2C > closest B2I

#<clamp cell> <x y (um)> <pin> <layer> <clamp instance> <bump> <x y (um)> <layer>
#- <pass/fail> <B2C ohm> <B2I ohm> <cell> <x y (um)> <pin> <layer> <instance>


clamp_celll (  2186.1951 1017.8250  ) VDD P_SOURCE_DRAIN inst01 VDDvsrc21 (  2331.5000 212.29800  ) I
- FAIL 0.40287828 0.39962601 celll (  2138.3650 726.12000  ) VDD M3 inst12
```


© 2013-2023 All Rights Reserved.

Report of Isolated Bump/Clamp Rule

Sample filename: `esd_isolation.reff`

Format:

```
#NET vdd
#Isolated Total Bumps : value
list_of_bumps

#Isolated Total Clamps : value
list_of_clamps
```

Format Description:

`Isolated Total Bumps` specifies the total number of isolated bumps and lists the names of the isolated bumps.

`Isolated Total Clamps` specifies the total number of isolated clamps and lists the names of the isolated clamps.

Sample Report:

```
- #NET DVDD_D1
- #Isolated Total Bumps : 0
-

- #Isolated Total Clamps : 2
- PVDD_08_08_NT_DR_H_I7_RING1 PVDD_08_08_NT_DR_H
- PVSS_08_08_NT_DR_H_I6_RING1 PVSS_08_08_NT_DR_H
```

Current Density Check Report

Sample filename: VDD.rj.avg.rpt

Format:

```
I/Ilimit   I(mA)   Ilimit(mA)   Layer   Location {X1 Y1 X2 Y2}   ShortLength
(um)   Width (um)   ViaArea(nm^2) Via Cuts   Fix Width/Via   Current
Direction   Benefit Factor   J(A/cm^2)   Jmax(A/cm^2)   ResistorID
```

Format Description:

© 2013-2023 All Rights Reserved.

`I(mA)` specifies the current value that flows through the Metal/Via resistor that is calculated by the tool.

`Ilimit(mA)` specifies the current limit value that is provided by the foundry depending upon its width and thickness.

`Layer` specifies the metal/via layer name on which the resistor is located.

`Location` specifies the coordinates of the resistor.

`ShortLength` is also known as "Blech length" or "Path length". It can be explained as "the longest and continuous centerline path from edge to edge among the connected wire shapes on the same metal layer".

`ShortLength` = Drawn length of longest continuous same metal * Layout scale factor (from qrctechFile).

`Width` specifies the width of the resistor.

`Via Area` is the drawn dimension via area in square microns.

`Via Cuts` is the number of via cuts.

`Fix Width/ Via` The “Fix Width/Via” is recalculated by modifying the existing via width to eliminate EM violation. If `I/Ilimit` is greater than 1, which is a violation, Fix Width/Via will be:
`violation * via width`

`Current Direction` is applicable only to via layers where 'D" denotes down, "U" denotes up.

`Benefit Factor` When certain constraints (blue-highlighted - “L>3W<0.4”) are met, its EM limit can be 2X, 3X, or multiple X of the base EM limit. These multiples are called Benefit Factor (red-highlighted in the above example - “1”, "2").

```
m_jmax_dc_avg EQU 1 * 1.059 * ( w - 0.003 )  jmax_factor 100 1.448 105 1.000 110
0.697 115 0.491 120 0.349 125 0.250 130 0.180 140 0.096 150 0.053 jmax_life 87600
1.000 L > 3 W < 0.04
```

```
em_jmax_dc_avg EQU 2 * 1.059 * ( w - 0.003 )  jmax_factor 100 1.448 105 1.000 110
0.697 115 0.491 120 0.349 125 0.250 130 0.180 140 0.096 150 0.053 jmax_life 87600
1.000 L> 3 W >= 0.04
```

Here, W is the wire width and L is the Blech length.

`J(A/cm^2)` is the current density value.

`Jmax(A/cm^2)` is the current density limit.

© 2013-2023 All Rights Reserved.

`ResistorID` is the unique ID assigned to a resistor.

Sample Report:

# I/Ilimit	I(mA)	Ilimit(mA)	Layer	Location (X1 Y1 X2 Y2)	ShortLength (um)	Width (um)	ViaArea (nm^2)	Via Cuts	Fix Width/ Via	Current Direction	Benefit Factor	J(A/cm^2)	Jmax(A/cm^2)	ResistorID
#														
#														
5257.01	2.602220e+02	4.950000e-02	M1	{12.0475 14.3350 12.0475 14.4650}	N/A	0.0495	N/A	N/A	260.222	N/A	1.00	5.257010e+08	1.000000e+05	69
5114.2	2.531527e+02	4.950000e-02	M1	{14.4775 17.1350 14.4775 17.2650}	N/A	0.0495	N/A	N/A	253.153	N/A	1.00	5.114196e+08	1.000000e+05	83
4772.82	2.362546e+02	4.950000e-02	M1	{15.2875 17.1350 15.2875 17.2650}	N/A	0.0495	N/A	N/A	236.255	N/A	1.00	4.772821e+08	1.000000e+05	77
4257.56	2.107491e+02	4.950000e-02	M1	{16.0975 14.3350 16.0975 14.4650}	N/A	0.0495	N/A	N/A	210.749	N/A	1.00	4.257557e+08	1.000000e+05	57

© 2013-2023 All Rights Reserved.

Summary Report

The Summary report summarizes the PASS/FAIL status of each rule with rule details. This report allows you to assess the ESD impact on the design before referring to the individual detailed reports.

Sample filename: `summary.rpt`

Sample Report:

The following are snippets of the summary report:

- Clamp-to-Clamp Rule

```
Rule c2c_rule_core
type clamp2clamp
Reff_threshold 3
power -nets VDD
ground -nets VSS
clamp_types -from C1 C2 -to C3
End Rule


Rule Status FAIL

Net                 Type        FromClamps  ToClamps    TotalPaths  PassClamps  FailClamps  Status      MinRes      MaxRes
-------------------------------------------------------------------------------------------------------------------------------

VDD                 power       12          2           24          0           12          FAIL        1.61987     21.1111
VSS                 ground      12          2           24          3           9           FAIL        0           17.979

-------------------------------------------------------------------------------------------------------------------------------
```

`Rule Status`: Specifies the `PASS`/`FAIL` status of the rule. It will have the status `PASS` if all the nets under this rule pass, and will have the status `FAIL` if even one net fails.

`Net`: Nets analyzed under this rule

`Type`: Specifies the type of the net being analyzed

`FromClamps`: Number of clamps analyzed under the from section of the rule on each net

`ToClamps`: Number of clamps analyzed under the to section of the rule on each net

`Total Paths`: Number of paths analyzed. If from and to clamps are of the same clamp type, total paths = `#FromClamps * (#Toclamps - 1)`. If from and to are not of same clamp type, total paths = `#Fromclamps * #Toclamps`.

© 2013-2023 All Rights Reserved.

`PassClamps`: Number of pass clamps. Criterion for a clamp to pass is if all paths from that clamp passes.

`FailClamps`: Number of fail clamps.

`Status`: Status of each net. Status is set to `Pass` if `FailClamps` = `0`, and is set to `Fail` if `FailClamps` > `0`.

`MinRes`: The minimum resistance value for each net.

`MaxRes`: The maximum resistance value for each net.

- Bump-to-Clamp Rule

```
Rule b2c_rule_core
type bump2clamp
Reff_threshold 2.5
Reporting both
power -nets all -short_bumps true
ground -nets all -short_bumps true
clamp_types C1 C2 C3
End Rule


B2C Status FAIL

B2C Summary

Net                 Type        Clamps      Bumps       Short       TotalPaths  PassPaths   FailPaths   Status      MinRes      MaxRes
-------------------------------------------------------------------------------------------------------------------------------------------

VDD                 power       14          1           TRUE        14          0           14          FAIL        12.783      14.4931
VSS                 ground      14          1           TRUE        14          0           14          FAIL        11.4422     12.8037
-------------------------------------------------------------------------------------------------------------------------------------------



C2B Status FAIL

C2B Summary

Net                 Type        Clamps      Bumps       Short       TotalPaths  PassPaths   FailPaths   Status      MinRes      MaxRes
-------------------------------------------------------------------------------------------------------------------------------------------

VDD                 power       14          1           TRUE        14          0           14          FAIL        12.783      14.4931
VSS                 ground      14          1           TRUE        14          0           14          FAIL        11.4422     12.8037
-------------------------------------------------------------------------------------------------------------------------------------------
```

`Rule Status`: Specifies the `PASS`/`FAIL` status of the rule. It will have the status `PASS` if all the nets under this rule pass, and will have the status `FAIL` if even one net fails.

`Net`: Nets analyzed under this rule

`Type`: Specifies the type of the net being analyzed

`Clamps`: Clamp cells on each net

`Bumps`: Bumps on each net

`Short`: Specifies if bumps were shorted

`Total Paths`: Number of paths analyzed. If bumps are shorted, total paths is equal to the number of clamps. If bumps are not shorted, total paths is equal to `#Bumps` * `#Clamps`

© 2013-2023 All Rights Reserved.

`PassPaths`: Number of pass bumps in the B2C Summary, or number of pass clamps in the C2B Summary. Criterion for a bump/clamp to pass is if all paths from that bump/clamp passes.

`FailPaths`: Number of fail bumps/clamps.

`Status`: Status of each net. Status is set to `Pass` if `FailPaths` = `0`, and is set to `Fail` if `FailPaths` > `0`.

`MinRes`: The minimum resistance value for each net

`MaxRes`: The maximum resistance value for each net

- Isolated Bumps and ESD Clamps

```
Rule esd_isolation
type connectivity
power -nets VDD
ground -nets VSS
clamp_types C1 C2 C3
Check_isolated_bumps true
Check_isolated_clamps true
End Rule


Rule Status PASS

Net                 Type        Clamps      Bumps       IsolatedClamps IsolatedBumps  Status
-------------------------------------------------------------------------------------------------

VDD                 power       14          1           0              0              PASS
VSS                 ground      14          1           0              0              PASS

-------------------------------------------------------------------------------------------------
```

`Rule Status`: Specifies the `PASS`/`FAIL` status of the rule. It will have the status `PASS` if all the nets under this rule pass, and will have the status `FAIL` if even one net fails.

`Net`: Nets analyzed under this rule

`Type`: Specifies the type of the net being analyzed

`Clamps`: Number of clamps on each net

`Bumps`: Number of bumps on each net

`Isolated Clamps`: Number of isolated clamps on each net

`Isolated Bumps`: Number of isolated bumps on each net

© 2013-2023 All Rights Reserved.

`Status`: Status of each net. Status is set to `Pass` if `Isolated Clamps/Isolated Bumps` = `0`, and is set to `Fail` if `Isolated Clamps/Isolated Bumps` > `0`.

- Bump-to-Bump Rule

```
Rule b2b_rule_core
type bump2bump
power -nets all -short_bumps true
ground -nets all -short_bumps true
lerp_threshold 10
End Rule


Rule Status FAIL

FromNet             ToNet               FromBumps   ToBumps     TotalPaths  FailedPaths Status      MinLoop     MaxLoop
-------------------------------------------------------------------------------------------------------------------------

VDD                 VSS                 1           1           1           1           FAIL        25.7732     25.7732
VSS                 VDD                 1           1           1           1           FAIL        1.00002e+06 1.00002e+

-------------------------------------------------------------------------------------------------------------------------
```

`Rule Status`: Specifies the `PASS`/`FAIL` status of the rule. It will have the status `PASS` if all the nets under this rule pass, and will have the status `FAIL` if even one net fails.

`FromNet`: Specifies the from-power/ground net name

`ToNet`: Specifies the to-power/ground net name

`FromBumps`: Number of bumps defined under the from section of the rule on each net

`ToBumps`: Number of bumps defined under the to section of the rule on each net

`TotalPaths`: Number of paths analyzed

`FailedPaths`: Number of failed paths

`Status`: Status of each net to net loop. Status is set to `Pass` if `FailedPaths` = `0`, and is set to `Fail` if `FailedPaths` > `0`.

`MinLoop`: The minimum loop resistance value for each net

`MaxLoop`: The maximum loop resistance value for each net

© 2013-2023 All Rights Reserved.

- Bump-to-Instance Rule

```
Rule b2i_core
type bump2instance
Reff_threshold 2.5
power -nets all -short_bumps false
ground -nets all -short_bumps false
End Rule


Rule Status FAIL

Net                 Type        Clamps      Bump        Instances   TotalPaths  FailInstances  Status      MinRes      MaxRes
-------------------------------------------------------------------------------------------------------------------------------

VDD                 power       2           44          500         22000       41             FAIL        0.381985    1.07071
VSS                 ground      2           147         500         73500       73             FAIL        0.280479    1.01224

-------------------------------------------------------------------------------------------------------------------------------
```

`Rule Status`: Specifies the `PASS`/`FAIL` status of the rule. It will have the status `PASS` if all the nets under this rule pass, and will have the status `FAIL` if even one net fails.

`Net`: Nets analyzed under this rule

`Type`: Specifies the type of the net being analyzed

`Clamps`: Clamp cells on each net

`Bump`: Bumps on each net

`Instances`: Instances on each net

`TotalPaths`: Number of paths analyzed

`FailInstances`: Number of failed instances

`Status`: Status of each net. Status is set to `Pass` if `FailInstances` = `0`, and is set to `Fail` if `FailInstances` > `0`.

`MinRes`: The minimum resistance value for each net

`MaxRes`: The maximum resistance value for each net

© 2013-2023 All Rights Reserved.

- Clamp-to-Instance Rule

```
Rule c2i_rule_core
type clamp2instance
Reff_threshold 4.5
power -nets all
ground -nets all
End Rule

Rule Status FAIL

Net                 Type        Clamps      Instances   TotalPaths  FailPaths   Status      MinRes      MaxRes
-------------------------------------------------------------------------------------------------------------------

VDD                 power       2           500         1000        1000        FAIL        15.4136     54.1211
VSS                 ground      2           500         1000        1000        FAIL        15.5375     79.3505

-------------------------------------------------------------------------------------------------------------------
```

`Rule Status`: Specifies the `PASS`/`FAIL` status of the rule. It will have the status `PASS` if all the nets under this rule pass, and will have the status `FAIL` if even one net fails.

`Net`: Nets analyzed under this rule

`Type`: Specifies the type of the net being analyzed

`Clamps`: Clamp cells on each net

`Instances`: Instances on each net

`TotalPaths`: Number of paths analyzed

`FailPaths`: Number of failed paths

`Status`: Status of each net. Status is set to `Pass` if `FailPaths` = `0`, and is set to `Fail` if `FailPaths` > `0`.

`MinRes`: The minimum resistance value for each net

`MaxRes`: The maximum resistance value for each net

© 2013-2023 All Rights Reserved.

Viewing Effective Resistance Check Results

Voltus GUI supports the visual display of ESD resistance check and EM results. To view the ESD analysis results:

1. Select *Power Rail - ESD Results*.

   The ESD Summary window appears. This window displays the list of rules that were analyzed.

2. Click the browse button to access the ESD directory under the state directory to load results, and click *Load*.

3. Click on a specific rule to view a summary of the rule.

© 2013-2023
All Rights Reserved.

The Rule Detail window appears. This window displays details of the rule selected in the ESD Summary window (top section), and the status summary for that rule (bottom section).

For the `em` rule type, the bottom section of the Rule Detail window shows the multiple nets associated with a domain.

4. Select a net (resistance check) in the lower section of the form to see the corresponding bumps/clamps and ESD cells.

© 2013-2023
All Rights Reserved.

Note: For the `em` rule type, select a domain in the lower section to see the net-based current density information. The Domain window displays multiple tabs based on the nets in the domain. Each tab will include the EM ratio, current values for the wire segment, current limit values, layer name of the segment, location of the segment, and width of the segment.

The Net window appears. This window displays the From and To sections for the net selected in the Rule Detail window.

Clamp	PIN	Layer	X(um)	Y(um)	Instance
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3277.889893	6321.200195	ESDCLAMP_1V2_INH_2M4PINS_inst_106
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3546.576904	731.770020	ESDCLAMP_1V2_INH_2M4PINS_inst_108
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3546.576904	1045.405029	ESDCLAMP_1V2_INH_2M4PINS_inst_109
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	1382.977051	2927.215088	ESDCLAMP_1V2_INH_2M4PINS_inst_11
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3546.576904	1359.040039	ESDCLAMP_1V2_INH_2M4PINS_inst_110
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3546.576904	1672.675049	ESDCLAMP_1V2_INH_2M4PINS_inst_111
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3549.221924	1986.310059	ESDCLAMP_1V2_INH_2M4PINS_inst_112
ESDCLAMP_1V2_INH_2M4P...	VDDCORE...	M4	3548.340088	2299.945068	ESDCLAMP_1V2_INH_2M4PINS_inst_113

From Clamp	To Clamp	To Clamp Cell Type	X(um)	Y(um)	PIN	Layer
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_16	ESDCLAMP_1V2_...	1656.953003	2598.580078	VDDCORE...	M4
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_22	ESDCLAMP_1V2_...	1927.402954	2509.580078	VDDCORE...	M4
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_11	ESDCLAMP_1V2_...	1382.977051	2927.215088	VDDCORE...	M4
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_17	ESDCLAMP_1V2_...	1653.427002	2927.215088	VDDCORE...	M4
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_15	ESDCLAMP_1V2_...	1653.427002	2299.945068	VDDCORE...	M4
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_9	ESDCLAMP_1V2_...	1385.621948	2298.945068	VDDCORE...	M4
ESDCLAMP_1V2_INH_2M4PINS_inst_10	ESDCLAMP_1V2_INH_2M4PINS_inst_23	ESDCLAMP_1V2_...	1923.876953	2927.215088	VDDCORE...	M4

The top section displays the *From Clamps/Bumps*. You can select a row from the top section to display the corresponding *To Clamps/Bumps* in the bottom section. In

© 2013-2023 All Rights Reserved.

addition, the GUI will highlight the fly-lines between from clamps/bumps and to clamps/bumps, as shown below:

You can select the *Show Plot* option to show or hide flight-lines when viewing ESD resistance-check results in the GUI.

ESD Analysis with specify_def

The `analyze_esd_network` command supports the `specify_def` flow where the design DEFs are not loaded at the outset which leads to performance gain.

Following is an example command list illustrating how to run `analyze_esd_network` with `specify_def`, including the PGV generation step:

```
set_multi_cpu_usage -localCpu 16
read_lib -lef  data/file.lef
set_advanced_pg_library_mode -common_supply_pins IO  -esd_cells { ESD1 ESD2 }

set_pg_library_mode -celltype techonly -filler_cells *FILL* -decap_cells *DECAP* -
extraction_tech_file data/qrcTechFile -lef_layermap data/lefdef.mapfile -
power_pins { VDD 1.0  } -ground_pins { GND } -stop@via V1 -current_distribution
propagation

generate_pg_library -output ./PGV
specify_def ./data/design.def

set_rail_analysis_mode -method static -temperature 125 -accuracy hd -
power_grid_library { \./PGV/techonly.cl \} -vsrc_search_distance 75 -
ignore_shorts  false -enable_manufacturing_effects false -
report_via_current_direction  false -enable_rlrp_analysis false
```


© 2013-2023 All Rights Reserved.

```
analyze_esd_network -config_file ./esd_new_rule.txt -type domain -output
ESD_analysis
```

CDM Based ESD Analysis

Charged-device model (CDM) based ESD events occur when a chip is charged during manufacturing, for example, during board assembly, and it comes in contact with a conductive surface. Even with CDM robustness of advanced mixed signal ICs, CDM failures on gate oxides can occur on driver and receiver pairs within the same power and ground domain. A large voltage drop between the local ground of drivers and receivers can result in a high voltage drop on the receiver gate, resulting in oxide damage. Although, CDM ESD damage often occurs at power domain signal crossings, but it can also occur for high resistive ground connections between driver and receiver.

The CDM based ESD Analysis flow allows you to report the driver-receiver violation pairs and perform static IR Drop analysis to check the voltage drop over a given limit between all driver and receiver pairs. This flow helps to analyze the driver/receiver pairs for CDM failure.

The following inputs are required for CDM based ESD analysis:

- Net name
- Driver/receiver file of the net
- ESD pin location file
- Current distribution layer
- Total ESD current to be distributed

The output of the analysis is a text report (.rpt), which contains the driver/receiver voltage drop information for the specified net.

The following steps are performed to run the CDM based ESD Analysis flow:

1. Load the design data or the Innovus database.
2. Perform power calculation to generate the driver/receiver pairs.

   You must specify the `-create_driver_db` parameter of the `set_power_analysis_mode` command attribute to determine all the drivers/receivers on the ground net.
3. Set up rail analysis for extraction.
4. Perform IR Drop analysis for every driver/receiver pair and report voltage drop above the given limit. Use the `analyze_esd_voltage` command to provide the current injection

© 2013-2023 All Rights Reserved.

points (sinks), the current distribution layer (source), and the total ESD current to be distributed.

The command generates a report containing all the violations where potential difference across driver/receiver is greater than the given threshold.

The following is a sample Tcl script to run this flow:

```
##Load the design
read_lib -lef { tech.lef all_cell.lef }
read_lib {all_cell_lib}
read_verilog chip_top.v.gz
set_top_module chip_top
specify_def { input_data/def/chip_top.def ...}
read_twf ../design.twf.gz
specify_spef { input_data/spef/chip_top.cmax_hot.spef ....}

##Run power calculation
set_dc_sources vdd -power -voltage 1.32
set_dc_sources gnd -ground -voltage 0.0
set_default_switching_activity -input_activity 0.2
set_power_analysis_mode -method dynamic_vectorless  -write_static_currents true -
disable_static false -create_driver_db true
report_power -outfile power.rpt

##Set up rail analysis
set_pg_nets -net gnd -voltage 0 -threshold 0.01
set_pg_nets -net vdd -voltage 1.32 -threshold 1.2
set_rail_analysis_domain -name PD -pwrnets vdd -gndnets gnd
set_power_pads -net gnd -format xy -file vss.pad.check
set_rail_analysis_mode -method static -accuracy hd -power_grid_library {
pgv/pgv_output/techonly.cl all_cells_pgv.cl} } -temperature 125 -ignore_fillers
false -ignore_decaps false  -cluster_via1_ports false

##Perform IR drop analysis
analyze_esd_voltage -net gnd -driver_receiver_file ./POWER/driverdb_railgnd  -
esd_pin_location_file ./epinfile -current_distribution_layer m1 -total_current 8 -
report FULL_new_gnd_esd_cdm.rpt
```

The following is the sample ESD failure report:

```
##############################################################################
File Info    : Driver/Receiver Voltage drop with ESD pin peak current = 8A
# File Columns : 1. Differential voltage(V) 2. Driver name (node location(um) ) 3.
Receiver name (node location(um)) 4. ESD pin name (location(nm))
# Date Created : 2016-Jan-17 22:22:35 (2016-Jan-18 06:22:35 GMT)
# User login   : user
# Created on   : sjfsb088
##############################################################################

14.1324 bot/a_rcvr_55 ( 2050.875 -2760.085 ap ) io_toplevel/padring/I_ESDCLAMP_39
( 1428.200 2884.250 m4 ) gnd_pin_22 ( 1458.200 2898.050 m4 )
13.9963 bot/a_rcvr_55 ( 2050.875 -2760.085 ap ) io_toplevel/padring/I_PAD_18 (
1529.710 2860.000 m4 ) gnd_pin_99 ( 1525.000 2898.050 m4 )
...
...
```


© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

15

Package Analysis

- Overview on page 390
- Static and Dynamic Rail Analysis Including Package Model on page 391
  - Simple Lumped RLC Pin Package Model on page 391
  - Fully Distributed and Coupled RLCK Package SPICE Model on page 393
  - RDL0 or Off-Chip Package Tracing on page 397
  - Package to Die Mapping on page 398
- Voltus-Sigrity Package Analysis Flow on page 400
  - Package Extraction on page 400
  - Rail Analysis Including Package Model on page 402
  - Package Model Analysis on page 406
  - View Package Analysis Results on page 408
  - Access Sigrity Tools from Voltus on page 409
  - Example Script on page 411
- Generating die-model on page 413
  - Need for die-model generation on page 413
  - Die-model generation features on page 414
  - Voltus flow to create die-model on page 415
  - Time domain analysis of package and die-model on page 417
  - Frequency domain analysis of package and die-model on page 420

© 2013-2023 All Rights Reserved.

Overview

For today's system-on-chip (SoC) designs, it is essential that the IC power analysis includes the package effects for accurate simulation results from the early design stage to the signoff stage. To address this need, Voltus IC Power Integrity Solution supports the following two types of package models for both the static and dynamic rail analysis:

- Simple lumped resistance, capacitance, and inductance (RLC) pin model for the block-level or early design analysis
- Fully distributed and coupled RLCK SPICE model for full-chip signoff. This model can be generated from different IC packaging modeling products such as, Sigrity XtractIM.

To improve the chip-package co-simulation turn-around time and flow usability, Volltus IC Power Integrity Solution also provides a feature called as: Voltus-Sigrity Package Analysis (SPA). It allows the IC designers to access Sigrity products from the Voltus environment and seamlessly perform:

- Package model extraction for IC rail analysis
- Package-aware on-chip static and/or dynamic rail analysis
- IC-aware package IR drop analysis for validation and what-if analysis
- Chip-package result co-display and cross reference for debugging and what-if analysis

The license requirement for SPA is Voltus-L/XL + Voltus-AA + Voltus-SPA. For details on licensing, refer to Product and Licensing Information. You must install Sigrity version 2015 QIR3 or later. For details on Sigrity installation, refer to *Cadence Sigrity 2015 Release Installation Guide*.

Furthermore, to enable system-centric chip-package-PCB co-simulation, Voltus IC Power Integrity Solution is able to generate one of the following die models:

- An n-port reduced R model with static current profile after static rail analysis
- An n-port reduced RC model with dynamic switching current profile after dynamic rail analysis

These die models can be used with package designs, as well as PCB designs in the Sigrity Package Analysis products to verify whether or not the entire power network across chip, package, and PCB meets the targeted static and dynamic voltage noise margins and targeted impedances.

This chapter covers the following topics:

- Static and dynamic rail analysis including package model

© 2013-2023 All Rights Reserved.

- Voltus-SPA flow
- Die-model generation

Static and Dynamic Rail Analysis Including Package Model

Package model information can be specified using the Voltus GUI or the TCL commands. When loading the RLC package pin models or SPICE package models, a confirmation of the values used or package model files loaded are written to the log file.

During static analysis, Voltus considers only the resistive elements. It ignores the capacitive and inductive effects by modeling them as opens and shorts respectively. During dynamic analysis, Voltus considers resistive, capacitive, and inductive effects. It shorts voltage sources of 0 volts and ignores resistance values above 1 megohm.

Simple Lumped RLC Pin Package Model

Default package parameters

You can set the default package R (Resistance), L (Inductance), and C (Capacitance) in a Tcl command as below:

```
set_rail_analysis_mode \
[-default_package_resistor value]
[-default_package_capacitor value]
[-default_package_inductor value]
```

The default package parameter values are applied to each power pad defined using the `set_power_pads` command.

Package parameters embedded inside power pad location file

You can also set the unique package R (Resistance), L (Inductance), and C (Capacitance) for each power pad using the `set_power_pads` command.

```
set_power_pads -net <power_net_name> -format xy -file <power_net>.pp
set_power_pads -net <ground_net_name> -format xy -file <ground_net>.pp
```

Example of the <power_net>.pp file:

```
*-------------------------------------------------------------------
* <vsrc name> x(um) y(um) <layer> Package R(ohm) L(H) C(F)
* vsrc name must be unique
```


© 2013-2023 All Rights Reserved.

```
* vsrc name across different net name in domain should be unique
*-----------------------------------------------------------------
VDDvsrc100 5.000 793.500 M7 R=25e-03 L=125e-12 C=1.80e-12
VDDvsrc101 1187.835 293.500 M7 R=15e-03 L=225e-12 C=2.80e-12
VDDvsrc102 695.000 5.000 M6 R=35e-03 L=325e-12 C=3.80e-12
VDDvsrc103 1187.835 493.500 M7 R=45e-03 L=425e-12 C=4.80e-12
```

Default package parameters in GUI

You can also set the default package R (resistance), L (Inductance), and C (Capacitance) for all power pins in the Power & Rail - Set Rail Analysis Mode form, as shown below:

© 2013-2023
All Rights Reserved.

Package parameters embedded inside power pad location file in GUI

In the GUI, the power pad location file can be supplied in the *Advanced* tab of the Power & Rail - Run Rail Analysis form. If you do not have a power pad location file, you can create it by clicking the *XY File* option, and then clicking the *Create* button.

Fully Distributed and Coupled RLCK Package SPICE Model

The package SPICE models used in Voltus can be from different package modeling tools. However, the models generated using Sigrity XtractIM is better integrated with Voltus.

Set SPICE Package Subckt in Tcl

You can specify the SPICE package model in the Tcl command as below:

```
set_package -spice <file> [-mapping <file>] [-offset {x y}] [-rotate
<rotation_angle_in_degress>] [-flip {true | false}]
```

where `[-mapping <file>]`, `[-offset {x y}]`, `[-rotate <rotation_angle_in_degress>]`, and `[-flip {true | false}]` are optional parameters. They are not required if the package model is extracted by Sigrity XtractIM and if it contains the die placement information. If the die placement information is not available in the XtractIM package model, or the package model is from third-party tools, then you can

© 2013-2023 All Rights Reserved.

use either the MCP Editor utility to create a mapping file, or some of these parameters to instruct rail analysis how to connect the package model properly. Refer to Package to Die Mapping for details.

Set SPICE Package Subckt in GUI

You can also specify a package model and mapping file in Power & Rail - Run Rail Analysis form, as shown below:

SPICE package model format

A SPICE package model generated by Sigrity XtractIM consists of:

- Subcircuit node definition
- Sigrity Model Connection Protocol (MCP) header
- RLCKEFGH elements

Example of a package model file

```
.SUBCKT C1_Voltus_4L_C1_Voltus_4L_Final_PinBaseSPICE
+              CD1_BMP_9_1_1             FPBGA144_A9
```


© 2013-2023 All Rights Reserved.

```
+             CD1_BMP_10_3_1
+             CD1_BMP_11_5_1
+             CD1_BMP_13_9_1
+             CD1_BMP_14_11_1
+             CD1_BMP_15_13_1
+             CD1_BMP_16_15_1
+             CD1_BMP_25_1_3
……
*Cadence SpeedXP Suite XtractIM Version 15.0.3.07171 S004
*
*The following is the Cadence MCP(model connection protocol) Section
************************************
*[MCP Begin]
*[MCP Ver] 1.1
*[Structure Type] PKG
*[MCP Source] Cadence Design Systems, Inc. XtractIM 15.0.3.07171 S004 7/30/2015
*
***********************************
*
*[REM]The following is the info for component connection CD1
*[REM]***********************************
*[Connection] CD1 CD1 200
*[Connection Type] DIE
*[Power Nets]
*BMP_9_1_1      CD1_BMP_9_1_1   VDD     0.0016632       -0.0005292
*BMP_10_3_1     CD1_BMP_10_3_1  VDD     0.0016632       -0.0003780
*BMP_11_5_1     CD1_BMP_11_5_1  VDD     0.0016632       -0.0002268
*BMP_13_9_1     CD1_BMP_13_9_1  VDD     0.0016632       0.0000756
*BMP_14_11_1    CD1_BMP_14_11_1 VDD     0.0016632       0.0002268
*BMP_15_13_1    CD1_BMP_15_13_1 VDD     0.0016632       0.0003780
……
*
*[MCP End]
*This concludes the MCP section
…….
* Begin Chip Package Protocol
* Start Version Info
* CPP_version 1.1
* Generator_Program XtractIM  15.0.3.07171 S004
* End Version Info
* Start Design Property
* DesignType Package
* End Design Property
* Start Package Property
* PkgType flipchip dieup
* End Package Property
*
* Start Units
* Length um
* End Units
*
* Start Power Ground Ports
* CD1-BMP_9_1_1:(1663.2   -529.2):CD1_BMP_9_1_1=VDD:VDD_Group0_0_0:DIE
* CD1-BMP_10_3_1:(1663.2   -378):CD1_BMP_10_3_1=VDD:VDD_Group0_0_1:DIE
* CD1-BMP_11_5_1:(1663.2   -226.8):CD1_BMP_11_5_1=VDD:VDD_Group0_0_2:DIE
…….
*
LD1 CD1_BMP_9_1_1 DrN1 1.3719e-10
LD2 CD1_BMP_10_3_1 DrN2 1.37633e-10
LD3 CD1_BMP_11_5_1 DrN3 1.39643e-10
LD4 CD1_BMP_13_9_1 DrN4 1.38742e-10
```


© 2013-2023 All Rights Reserved.

```
LD5 CD1_BMP_14_11_1 DrN5 1.35669e-10
…
Kd188_190 LD188 LD190 0.244828
Kd189_190 LD189 LD190 0.229717
C1_1 FPBGA144_A9 Node__GND 8.04931e-13
RG1_1 FPBGA144_A9 Node__GND 113345
C2_2 FPBGA144_A1 Node__GND 7.5236e-13
RG2_2 FPBGA144_A1 Node__GND 119953
….
.ends
```

Controlled and Dependent Sources

In addition to the RLCK elements, Voltus supports the following "Controlled" or "Dependent" sources in the package model:

- Voltage controlled voltage source (VCVS)
- Current controlled current source (CCCS)
- Voltage controlled current source (VCCS)
- Current controlled voltage source (CCVS)

The format of these sources are given below:

```
Exxx n+ n- <VCVS> in+ in- gain
Fxxx n+ n- <CCCS> vn1 gain
Gxxx n+ n- <VCCS> in+ in- transconductance
Hxxx n+ n- <CCVS> vn1 transresistance
```

where,

n+ and `n-` are the positive and negative terminals of the dependent source, respectively.

`in+` and `in-` are the positive and negative terminals of the controlling voltage source, respectively.

`vn1` is the zero value voltage source used to measure the controlling current.

Example

```
.SUBCKT INV_5_5
+ VDDIN_1 VDD
R1 VDDIN_1 2 3
R2 2 0 5
R3 2 3 3
R4 3 0 5
R5 3 4 3
Evx 4 VDD VCVS 2 0 0.1
Evx1 4 VDD 2 0 0.2
```


© 2013-2023 All Rights Reserved.

```
Gvx 4 VDD VCCS 2 0 0.3
Gvx1 4 VDD 2 0 0.4
.ENDS
```

RDL0 or Off-Chip Package Tracing

Package Analysis lets you include RDL0 (off-chip package trace) resistance effects by tracing internal switched net bumps during static and dynamic rail analysis. A design will have dummy bumps on the switched nets to define chip terminals that are used for package to hook up RDL0 netlist to the power-grid, as shown below:

Separate RDL0 subcircuit **RDL0 subcircuit inside package netlist**

There are two types of RDL0 runs:

- **Run without package** - a separate RDL0 subcircuit is created and hooked to the power-grid.
- **Run with package** - an RDL0 subcircuit is provided inside the package netlist that is hooked to the power-grid.

The off-chip package tracing feature allows you to analyze the switched net bumps that are marked as RDL0 bumps. The inputs required to use this feature are:

- trace model names (names of switched net bumps)
- switched net bump location files
- SPICE subcircuit (RDL0 netlist or full-chip package netlist)
- mapping file containing mapping of the terminals of the SPICE subcircuit (SPICE circuit port name) in the package model file to the locations at the power-grid (voltage source name in switch net's)

© 2013-2023 All Rights Reserved.

To enable off-chip package tracing, specify the `set_offchip_package_trace` command. When this command is specified, the software automatically generates switched net bump current and bump voltage waveforms in the `voltus_rail_pad.tran.ptiavg` file created in the switch net directory located under the rail state directory.

Tcl Command Example:

```
set_offchip_package_trace -name VDDau  -ploc_file_list  ploc_file_list.txt -spice
pkg.ckt -mapping pkg_mapping.file
```

The format of the `-ploc_file_list` *`file`* is:

```
SWITCHED_NET1 plocfile1
SWITCHED_NET2 plocfile2
…..
```

The format of *`plocfile`* (power pad) is:

```
<voltage_source_name> <x> <y> <metal_layer>
```

The format of the `-mapping` file is:

```
<spice_circuit_port_name> pin <vsrcName_in_switch_net's_ploc>
```

The following is a snippet of `ploc_file_list.txt`:

```
VDDau /design/data/switch_rdl_VDDau.pp
```

The following is a snippet of `pkg_mapping.file`:

```
1402 pin VDDausrc1
0402 pin VDDausrc1
```

Package to Die Mapping

Voltus can perform automatic mapping between package and die in two different ways, as explained below:

- **Package with die placement information**: This information describes how a chip is placed on a package and is specified as die coordinates, flip, and rotation related to the package system. It is usually saved into a package design database, such as, .MCM or .SiP, and then gets carried over into the Model Connection Protocol (MCP) section of the Sigrity XtractIM package model via the keywords: [BBOX], [Flip] and [Rotation]. Using the die placement information, Voltus automatically determines the die shrink factor and creates a mapping between package and die.

  Following is an example of the die placement information in a package model:

© 2013-2023 All Rights Reserved.

```
*[REM] [Rotation] 0
*[REM] [BBOX] -0.0039258 -0.003195 0.0039258 0.003195
*[REM] [Flip] No
```

- **MCPEditor Utility**: If the package model does not contain the BBOX information, Voltus can still perform automatic mapping using the *MCP Editor* utility. For more information, refer to Rail Analysis Including Package Model.

Voltus also allows the user-specified package die connections with the `set_package` parameters: `[-mapping <file>] [-offset {x y}] [-rotate <rotation_angle_in_degress>] [-flip {true | false}]`

Mapping File Format

Mapping File specifies the name of the package terminal mapping file, which contains the mapping of the terminals of the SPICE subcircuit in the package model file to the locations at the grid or top-level pins. The mapping file formats supported by Voltus are:

- Model Connection Protocol (MCP)
- Voltus-specific Format

Note: If the MCP file is embedded in the package model file, you do not need to specify a mapping file.

For more information on the mapping file formats, refer to the **Package Model and Mapping File** section in the "*File Formats*" chapter.

Format of Voltus-specific terminal mapping file

```
<Pkg_die_pin_name> pin <die_vsrc_pin_name>
<pkg_bga_power_pin_name> power_pin <net_name>
<pkg_bga_gnd_pin_name> ground_pin <net_name>
```

Example of Voltus-specific terminal mapping file

```
VDD_in power_pin VDD_CPU
VSS_in ground_pin VSS
VDD_out pin  VDD_CPUvsrc1
VDD_out pin  VDD_CPUvsrc2
VDD_out pin  VDD_CPUvsrc3
```


© 2013-2023 All Rights Reserved.

Voltus-Sigrity Package Analysis Flow

Voltus-Sigrity Package Analysis (SPA), the IC-centric Chip-Package co-simulation flow, has Sigrity embedded in Voltus to provide an accurate and easy-to-use power signoff solution for a highly-coupled Power Delivery Network (PDN).

It allows you to access the Sigrity package analysis tools from the Voltus environment for:

- Package model generation
- Package-aware static and dynamic rail analysis
- Package IR drop analysis
- Chip-package result co-display and cross reference
- Access to Sigrity MCP editor, if required

Note: This feature requires the Voltus-SPA license and Sigrity version 2015 QIR3 or later.

Package Extraction

Voltus generates a package model for user-specified power nets by calling Sigrity XtractIM in the batch mode. XtractIM is a tool that provides a complete model extraction environment focused specifically on IC package applications.

To create a package model using GUI:

1. Select *Set Advanced Package Options* from the *Package* menu.

   The following form appears:

2. Specify the full path to the bin folder of the package extraction and analysis tools (XtractIM and PowerDC) in the *Sigrity Tool Path* field.

3. Specify the name of the two column mapping file that contains the mapping between the net names of package with the net names of die in the *Net Mapping File* field. Following is a sample format:

© 2013-2023 All Rights Reserved.

```
vdd_ao VDD
vdd_external VDD
```

4. Click *Save* to save the settings as a `.conf` file (configuration file), and reuse it next time by loading it.

5. Click *OK*.

   After specifying the tool path, access the Extract Package form to start package model extraction.

6. Select *Extract Package* from the *Package* menu.

   The following form appears:

7. Specify the work space name (package model database) of the package extraction tool (XtractIM) in the *Workspace* field. The file name extension is `.ximx`.

8. Specify the domain name in the *Rail Analysis Domain* field.

9. Define the die circuit name in the *Package Die Name* field. When specified, the software automatically creates a mapping between the die name in the package extraction tool and the die name in the software.

10. Define the ball grid array (BGA) circuit name in the *Package BGA Name* field. It is the PCB component in the package model, which eventually connects to the PCB model.

11. You can specify the output/result directory in the *Output Folder* field. By default, the package model will be created in the workspace. However, if you specify an output directory, then a copy of the result will be made in the new directory.

12. Specify the ground net that is selected as reference net for pin-based model generation in the *Reference Net* field.

13. Click *OK*.

    The result of the package model extraction, which is done by calling XtractIM from within Voltus, is a Spice model that will be stored in the specified output directory. The naming format of the file is: `<xtract_im workspace>_<spd_filename>_Final_PinBaseSPICE.ckt`.

© 2013-2023 All Rights Reserved.

The TCL command for the Set Advanced Package Options form is `set_advanced_package_options`, and for the Extract Package form is `extract_package`.

Rail Analysis Including Package Model

Rail analysis is performed with the package model included. Refer to the "Static and Dynamic Rail Analysis Including Package Model" section for details. In the SPA flow, rail analysis will automatically pick the package model generated using the `extract_package` command. Hence, you do not need to specify the package model file via the `set_package` command. You can specify whether or not to perform the auto chip-package connection by using the following command:

```
set_rail_analysis_mode -unconnected_die_pkg_pins {ignore | error | edit}
```

- If the `-unconnected_die_pkg_pins` parameter is set to "`ignore`", then Voltus will continue simulation without pausing or stopping. This means that there are some unconnected pins between package and die, and you would like to continue with whatever connections are made.
- If the `-unconnected_die_pkg_pins` parameter is set to "`error`", then Voltus stops the simulation by erroring out. This means that there are some unconnected pins between package and die, and you do not want to ignore them, but want to start the MCP Editor GUI to make manual connections.
- If the `-unconnected_die_pkg_pins` parameter is set to "`edit`", then the Voltus simulation will pause, and the MCP Editor GUI will appear to allow you to make manual connections between package and die pins. This means that there are unconnected pins, and you want to start the MCP Editor GUI to try manual connections.

After performing manual connections, the simulation will continue irrespective of whether there is a 100% match or not.

To perform automatic mapping using the MCP Editor utility:

1. Select the *Generate Mapping File* option from the *Package* -> *Sigrity* -> *MCP Editor* menu.

© 2013-2023 All Rights Reserved.

The MCP Editor form appears:

2. You can select one of the following options:

   When Voltage Source Files is selected:

   a. Specify the net name and source file in the *Net* and *Source File* fields.

   b. Click *Add* to add details of the specific net and the corresponding source file to the power/ground nets table.

   c. Browse and select a package model in the *Package Model* field.

   d. Browse and select the output directory in the *Output Folder* field.

   When DIE MCP Header is selected:

   a. Browse and select a circuit file with the MCP header in the *DIE MCP Header* field.

   b. Browse and select a package model in the *Package Model* field.

   c. Browse and select the output directory in the *Output Folder* field.

3. Click *Launch*.

© 2013-2023
All Rights Reserved.

The following form appears:

4. Click *Auto Connect*.

© 2013-2023 All Rights Reserved.

The *MCP Auto Connection* form appears.

5. Specify an option to match the die and package pins by selecting one of the following radio buttons:

- ❑ *Auto match*
- ❑ *Pin name match*
- ❑ *Ckt node name match*
- ❑ *Net name match*
- ❑ *Coord match*

In the above sample form, the disconnected pins are matched by first selecting the *Pin name match* radio button, then specifying the left and right pins in the respective fields, and clicking the *Match* button.

© 2013-2023 All Rights Reserved.

The Pin Name Match information will be displayed at the bottom of the *MCP Auto Connection* form.

6. Click *OK* to close the *MCP Auto Connection* form and return to the previous form.
7. Click *OK* to close the *MCP Editor* form.

   The command prompt displays the following message:

   ```
   Successfully mapped die package model with the result mapping file
   ```

The TCL command for the MCP Editor form is `map_die_package`

The outputs for the Rail Analysis flow are:

- When the MCP Editor utility is internally invoked to establish a connection between die and package, a ploc file (`*ploc.ckt`) is generated that is available in the `<state_dir>/Reports` folder. This file is a dummy die model which contains the die pins and the MCP header.
- A two column mapping file (`*.map`) is generated, by default, which is in the `<state_dir>/Reports` folder. This file has the die pin names and the corresponding package pin names.
- If the MCP Editor utility is invoked, a mapping file will be saved in the `<state_dir>` which can be used as is in the `set_package -spice <> -mapping <mapping_file>` command.
- A die pin current voltage file (`*_iv.rpt`) is generated, by default, which is used in PowerDC to emulate the die model. The file is in the `<state_dir>/Reports` folder. The die pin current voltage file has three columns:
  `<pkg_pin_name>  <voltage_value>  <current_value>`
- The final results of Rail Analysis is in the `<state_dir>`.

Package Model Analysis

Voltus enables and runs package IR drop analysis by calling PowerDC from within the Voltus environment. The Sigrity PowerDC environment provides fast and accurate DC analysis for IC packages and PCBs.

To perform package model analysis in GUI:

1. Select *Analyze Package* from the *Package* menu.

© 2013-2023 All Rights Reserved.

The following form appears:

2. Specify the workspace name for package analysis (powerDC) in the *Workspace* field. The file name extension is `.pdcx`.
3. Specify the domain name in the *Rail Analysis Domain* field.
4. Define the die circuit name in the *Package Die Name* field. When specified, the software automatically creates a mapping between the die name in the package analysis tool and the die name in the software.
5. Define the BGA circuit name in the *Package BGA Name* field.
6. You can specify the package analysis results directory in the *Output Folder* field. By default, the package model results are created in the workspace.
7. Specify the name of the `.xml` result file in the *Result Name* field.
8. Specify the name of the generated pin current file from rail analysis that contains the package pin names and the current values in the *Pin Current File* field. It is automatically generated after the package model is created, and IR drop analysis is performed using the package model.
9. Click *OK*.

   The PowerDC GUI appears. Using this GUI, you can view the Electrical-field distribution once the simulation has completed. The 2D/3D E-Distributions are the static IR drop results in 2D (X/Y) or 3D (X/Y/Z). If you select 2D distribution, you can see the voltage/current on one layer at a time. Alternatively, if you select 3D distribution, you can see the X/Y (layer) information as well as the Z direction information.

The TCL command for the Analyze Package form is `analyze_package`.

© 2013-2023 All Rights Reserved.

View Package Analysis Results

You can use the Package Plots form to view the package IR drop analysis results in the PowerDC GUI which will be invoked from within Voltus.

To view package analysis results in GUI:

1. Select *Package Plots* from the *Package* menu.

   The following form appears:

2. Specify the workspace name for package analysis (powerDC) in the *Workspace* field. The file name extension is .pdcx.

3. Specify the name of the `.xml` result file in the *Result Name* field.

4. Click *OK* to view the package IR drop analysis results in the PowerDC GUI, as shown below:

© 2013-2023
All Rights Reserved.

The TCL command for the Package Plots form is `view_package_results`.

Access Sigrity Tools from Voltus

You can access the following Sigrity tools from within Voltus using the *Package -> Sigrity* menu:

- MCP Editor
- XtractIM
- PowerDC

These are additional options that allow you to access the package extraction (XtractIM) and analysis (PowerDC) tools to have more control over package modeling and to easily make modifications (if needed), and then proceed with extraction or package analysis.

Cross Probing between Voltus and Sigrity Results in GUI

When you run the Voltus-SPA flow, the Package IR drop results are displayed in the Sigrity (PowerDC) GUI, and the die results are displayed in the Voltus GUI. The software allows cross probing between the Voltus and Sigrity results, while viewing them in their respective GUIs, in the following ways:

- **Zoom Synchronization between Voltus and PowerDC**: When you zoom into a portion of the design in Voltus, the corresponding design area in PowerDC will be selected and zoomed in. Similarly, when you zoom into a portion of the design in PowerDC, the corresponding design area in Voltus will be selected and zoomed in.
- **Bump Synchronization between Voltus and PowerDC**: When you select a bump in Voltus, the corresponding bump in PowerDC will be selected and zoomed in. Similarly, when you select a bump in PowerDC, the corresponding bump in Voltus will be selected and zoomed in.
- **Net Enabled Synchronization between Voltus and PowerDC**: When you enable/disable nets in the Power & Rail Layers/Nets form in Voltus and click *Apply*, the corresponding nets will be enabled/disabled in PowerDC automatically. Similarly, when you enable/disable nets in PowerDC, the corresponding nets will be enabled/disabled in the Power & Rail Layers/Nets form automatically.

© 2013-2023 All Rights Reserved.

The following is a snapshot of bump synchronization:

© 2013-2023
All Rights Reserved.

The following is a snapshot of net synchronization:

Example Script

The following example illustrates a code snippet for package extraction and analysis:

```
#-----------------------------------------------------------------
# Static Rail Analysis Setup
#-----------------------------------------------------------------

set_rail_analysis_mode \
   -method                      static \
   -accuracy                    xd \
   -analysis_view              AV_wc_on \
   -power_grid_library          { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } \
    -use_em_view_list            ../data/voltus/em_view.list \
    -enable_rlrp_analysis       true \
    -verbosity true \
    -unconnected_die_pkg_pins error  \
    -temperature 125
```


© 2013-2023
All Rights Reserved.

```
#-----------------------------------------------------------------
# Package Extraction
#-----------------------------------------------------------------

set_advanced_package_options \
    -tool_path /home/sigrity/tools/bin/
    -net_mapping_file net_mapping.txt

extract_package ALL \
    -workspace ../rak_xim_pkg_model/C1_Voltus_4L.ximx \
    -pkg_bga_name FPBGA144 \
    -pkg_die_name CD1 \
    -output rak_xim_pkg_model_voltus_cpf \
    -reference_net VSS


#-----------------------------------------------------------------
# Rail Analysis
#-----------------------------------------------------------------

analyze_rail \
    -output          ./spa_cpf \
    -type                        domain \
                                    ALL


#-----------------------------------------------------------------
# Package Analysis
#-----------------------------------------------------------------

analyze_package ALL \
    -type dc \
    -workspace /../sigrity/rak_pdc_irdrop/test.pdcx \
    -pkg_bga_name FPBGA144_Signal02 \
    -pkg_die_name CD1_Signal01 \
    -output rak_pdc_irdrop_voltus_cpf \
    -result_name rak_pdc_irdrop_voltus \
    -pkg_pin_current_file
     /../spa_cpf/ALL_125C_avg_1/Reports/super_filter_pkg_iv.rpt

view_package_results \
    -type dc
```


© 2013-2023 All Rights Reserved.

Generating die-model

This section covers the following topics:

- Need for die-model generation
- Die-model generation features
- Voltus flow to create die-model
- Time domain analysis of package and die-model
- Frequency domain analysis of package and die-model

Need for die-model generation

An n-port reduced RC chip model with dynamic switching profile can be created during analysis. This die-model can be used with the package model in Sigrity products to verify whether package and on-chip dynamic profile meets the target impedance across wide frequency spectrum.

- Resistance in package routing results in voltage loss at chip level
  - Inductance in package routing results in dynamic power changes in die
  - The impact can be significant especially in low power designs
- Die model generation enables:
  - Chip-package resonance analysis
  - package decap estimate
  - System in Package (SiP) design and analysis

A cross section of a typical chip package is shown in the following diagram:

© 2013-2023 All Rights Reserved.

Die, package, and board

Die-model generation features

- Supports n-port die-model generation
- Supports die-model generation for TSV/SiP designs
- Eliminates iterations of long full-chip dynamic simulations with package
- Enables efficient resonance analysis of the design with different package and die parameters
- Die-model can be fed to Sigrity PowerSI to do freq domain analysis using SpeedSim
- Die-model and package can be simulated in Spectre to do freq domain analysis
- Allows package designer to optimize package in context of die to achieve target impedance at wide range of frequencies
- Allows circuit designers to optimize die (on-chip decaps and dynamic transient currents) in context of package to achieve the target impedance
- Additionally die + package model is generated in SPICE format to perform frequency and time domain analysis using Spectre to perform resonance analysis and verify accuracy of die-model

Die + package model

© 2013-2023
All Rights Reserved.

Voltus flow to create die-model

Die model generation flow is shown in the following diagram:

Die model generation flow in Voltus

Tcl command to generate die-model

```
create_die_model \
-state_directory irdrop_allOn/allDomains_25C_dynamic_1 \
-rail_analysis_domain allDomains \
-output_directory n_port_diemodel
```

Example Tcl Flow

```
# Import design
read_design ….
```

```
# Power calculation
set_power_analysis_mode….
report_power
```

```
# Rail analysis
set_rail_analysis_mode ….
set_power_data …
```


© 2013-2023 All Rights Reserved.

```
set_pg_nets ….
set_rail_analysis_domain ….
set_power_pads ….
set_package …
analyze_rail ….
# n-port die-model generation
create_die_model \
-state_directory allDomains_25C_dynamic_1 \
-rail_analysis_domain allDomains \
-output_directory n_port_diemodel \
-repeat 50
```

Note that `create_die_model` command is incremental and you can generate die-model in a new session without re-analyzing dynamic IR drop. As long as state directory containing dynamic IRdrop analysis result is available the `create_die_model` will use it to generate die-model. You will have to re-define power nets and domain using `set_pg_nets` and `set_rail_analysis_domain` again. Die Model generation can use multiple CPUs that have been set earlier in the script using the `set_multi_cpu_usage -localCpu` command.

TCL command to generate die-model for a TSV/SiP design

A TSV/SiP design has multiple die in the design. After performing dynamic IR drop analysis for the design, you can run the die-model generation flow to create a complete die model. As a result, you are able to use the complete die model for system simulation. The Tcl command to generate die-model is as follows:

```
create_die_model \
-state_directory irdrop_allOn/allDomains_25C_dynamic_1 \
-rail_analysis_domain allDomains \
-output_directory n_port_diemodel \
-multi_die true
```

Example Tcl Flow

```
# Rail analysis
set_rail_analysis_mode …-die_mode multi-die -die_instance_name die1
set_rail_analysis_mode …-die_mode multi-die -die_instance_name die2
set_power_data …-die_instance_name die1
set_power_data …-die_instance_name die2
set_pg_nets …-die_instance_name die1
set_pg_nets …-die_instance_name die2
set_power_pads … -die_instance_name die1
set_power_pads … -die_instance_name die2
specify_def …. -die_instance_name die1
specify_def …. -die_instance_name die2
set_rail_analysis_domain ….
set_multi_die_analysis_mode ….
set_package …
```


© 2013-2023 All Rights Reserved.

```
analyze_rail ….
```

Note: For information on TSV/SiP analysis, refer to the "**Domain-based Through-Silicon Via Flow**" section in the Through-Silicon Via and System-in-Package chapter.

```
# n-port die-model generation
create_die_model \
-state_directory allDomains_25C_dynamic_1 \
-rail_analysis_domain allDomains \
-output_directory n_port_diemodel \
-repeat 50 \
-multi_die true
```

Note: To generate a die model for a TSV/SiP design, you must specify the `-multi_die` parameter of the `create_die_model` command.

Time domain analysis of package and die-model

The following figure shows how the package is mapped to the die to perform full-chip dynamic transient analysis using Voltus.

Package + full-chip transient simulation

However, this transient analysis is expensive and limited to only a few clock cycles. In order to simulate the package + die for longer duration to analyze the impact of package inductance (Ldi/dt) beyond initial clock cycles, you can run transient analysis on package + die-model, as shown in the following figure:

© 2013-2023 All Rights Reserved.

Package + die-model transient simulation

The package + die-model transient simulation setup is captured in .sp file in the output directory specified by `create_die_model` command. This setup can be simulated in SPICE. Although original dynamic IR drop simulation is performed over limited number of cycles, it may not fully capture on-chip current and voltage oscillation due to package. You can choose to repeat the original dynamic current profile of the chip (captured as .isrc file in output directory) using the `-repeat` <time in ns> option in `create_die_model` command. This enables you to perform transient simulation of package and die-model for longer periods of time:

```
* Voltage sources *
*******************************
VVSS VSS_in 0 DC 0
VVDDm VDDm_in 0 DC 0.84
VVDD VDD_in 0 DC 1.08
******************************
* User specified package file *
******************************
.INCLUDE "./c2_399_msmv.ckt"
**********************************************
* Sub-circuit file created for the representing die
**********************************************
.SUBCKT ALL N001 N002 N003
** Die Model Resistors
R_VSS00 N001 N004 0.00794167
R_VSS01 N004 N007 0.00794167
R_VSS02 N007 N010 0.00794167
R_VDDm00 N002 N005 0.573108
R_VDDm01 N005 N008 0.573108
R_VDDm02 N008 N011 0.573108
R_VDD00 N003 N006 0.0413616
R_VDD01 N006 N009 0.0413616
R_VDD02 N009 N012 0.0413616
** Die model Capacitors
C_VSS00 N004 0 3.71397e-09
```


© 2013-2023 All Rights Reserved.

```
C_VSS_VDDm_01 N007 N008 2.27204e-24
C_VSS_VDD_03 N007 N009 2.4652e-18
C_VDDm00 N005 0 3.63009e-10
C_VDD00 N006 0 6.03553e-10
** Die Model Stimulus
.INCLUDE "ALL.sp.isrc"
.ENDS ALL
******************************
* Instance of package model *
******************************
Xpkg1 VSS_in PK_O2 VDD_in PK_O4 VDDm_in PK_O6 c2_399
******************************
* Instance of die model *
******************************
Xdie1 PK_O2 PK_O6 PK_O4 ALL

* Transient analysis
*.TRAN 10p 14n
*.PROBE TRAN I(*) V(*)
*.PRINT TRAN I(xdie1.R_VSS00) I(xdie1.R_VDDm00) I(xdie1.R_VDD00)
.END
```

You can also measure the die-model accuracy by probing the current through power-pad on die form SPICE simulation and correlating i t with VC_SUM signal generated by Voltus transient simulation. This comparison is shown below:

Current measurement at power pads

© 2013-2023
All Rights Reserved.

Frequency domain analysis of package and die-model

- At resonant frequency, on-chip capacitance and package inductance forms a circuit with high admittance that results in power supply oscillation beyond permissible threshold (+- 5-10%)
- The on-chip capacitance provides decoupling at high frequencies (low impedance because capacitor impedance in inversely proportional to frequency). However, components operating at low frequencies do not benefit from on-chip capacitance due to high impedance and hence current to these components primarily comes from package which is inductive in nature. So designs producing high transient current at low frequencies tend to oscillate the power supply and it will resonate at

  ```
  F0 = 1/2*pi*sqrt(LC) = resonance frequency
  ```

- Target impedance must be met at wide range of frequencies (1Hz->10GHz) to avoid power supply oscillation beyond threshold (+- 5-10%)

  ```
  Target Impedance = Ztarget = 10%VDD/Ipeak
  ```

  e.g. `VDD=1v` and `Ipeak = 0.1Amp;Ztarget = 1Ohm`

The resonance analysis with die-model can be done in the Sigrity environment using SpeedSim or using stand-alone Spectre circuit simulator. See results in the following form:

© 2013-2023 All Rights Reserved.

Chip package frequency response

The package + die-model frequency domain simulation setup is captured in `.sp` file in the output directory specified by `create_die_model` command. This setup can be simulated in SPICE.

```
*******************************
* Voltage sources *
*******************************
VVSS VSS_in 0 DC 0
VVDDm VDDm_in 0 DC 0.84 ac=1,0
VVDD VDD_in 0 DC 1.08 ac=1,0
*******************************
* User specified package file *
*******************************
.INCLUDE "./c2_399_msmv.ckt"
*********************************************
* Sub-circuit file create for representing die
*********************************************
.SUBCKT ALL N001 N002 N003
```


© 2013-2023 All Rights Reserved.

```
** Die Model Resistors
R_VSS00 N001 N004 0.00794167
R_VSS01 N004 N007 0.00794167
R_VSS02 N007 N010 0.00794167
R_VDDm00 N002 N005 0.573108
R_VDDm01 N005 N008 0.573108
R_VDDm02 N008 N011 0.573108
R_VDD00 N003 N006 0.0413616
R_VDD01 N006 N009 0.0413616
R_VDD02 N009 N012 0.0413616
** Die model Capacitors
C_VSS00 N004 0 3.71397e-09
C_VSS_VDDm_01 N007 N008 2.27204e-24
C_VSS_VDD_03 N007 N009 2.4652e-18
C_VDDm00 N005 0 3.63009e-10
C_VDD00 N006 0 6.03553e-10
** Die Model Stimulus
.INCLUDE "ALL.sp.isrc"
.ENDS ALL
*******************************
* Instance of package model *
*******************************
Xpkg1 VSS_in PK_O2 VDD_in PK_O4 VDDm_in PK_O6 c2_399
*******************************
* Instance of die model *
*******************************
Xdie1 PK_O2 PK_O6 PK_O4 ALL
* AC analysis
*.AC LIN 300 10Meg 10000Meg
*.PROBE AC I(*) V(*)
*.PRINT AC im(xdie1.R_VSS00) im(xdie1.R_VDDm00) im(xdie1.R_VDD00)
.END
```

Note that you can also feed only the die-model (`domain_25C_dyanamic_1/die_model/domain_without_package.sp`) to the Sigrity products to perform the analysis.

© 2013-2023 All Rights Reserved.

16

What-If Rail Analysis

- Scaling Resistance of the Power-Grid on page 424
- Scaling Static and Dynamic Currents on page 426
- Creating Static and Dynamic Current Regions on page 428
- Scaling Capacitance on page 429
- Creating Virtual Shapes on page 431

Voltus IC Power Integrity Solution offers wide range of what-if analysis capabilities to guide power-grid optimization. This analysis is used primarily to change electrical parameters of the power-grid on-the-fly to estimate eventual optimization efforts to satisfy IRdrop and ElectroMigration limits.

The following power-grid what-if analysis features are available in Voltus:

- Scaling Resistance of the Power-Grid
- Scaling Static and Dynamic Currents
- Creating Static and Dynamic Current Regions
- Scaling Capacitance

What-if analysis can be driven from the GUI as well as TCL commands. In the GUI, it is invoked from the *Run Rail Analysis - Advanced* form, as shown below:

© 2013-2023 All Rights Reserved.

What-If Analysis Setup Button

What-if analysis is also available through TCL commands:

- `scale_what_if_current`
- `scale_what_if_capacitance`
- `scale_what_if_resistance`
- `create_current_region`

Scaling Resistance of the Power-Grid

During what-if analysis, you can enable sensitivity analysis (rs) (see Viewing Static Rail Analysis Results in the Static Power, IRdrop and EM Analysis chapter). It highlights power-grid segments which when optimized, improves the IRdrop of the design. Based on the IRdrop - ir, the Resistor Sensitivity Analysis - rs, Current Density - rj plots, and Generating Text Reports, you can scale resistance of a given LEF layer on-the-fly to see whether changing the wire resistance improves the IRdrop. You can scale resistance of a layer in a region or globally for the design. Using the "*What-If Analysis Setup - Resistance*" form,

© 2013-2023 All Rights Reserved.

you can interactively draw a box in the layout canvas in which scaling is to be applied. If multiple nets are being analyzed, you can specify the net name for the same.

What-If Resistance Analysis Form

IRdrop problems inside a power-gated domain can be resolved by possibly increasing the drive strength of power-gates or adding more power-gates rather than optimizing power-grid mesh of the power-gated net. You can quickly perform what-if analysis by scaling the on-resistance of the already placed power-gates.

Example TCL flow

```
# Import design
read_design ….

# Power calculation

set_power_analysis_mode ….
report_power

# Rail analysis
set_rail_analysis_mode ….
set_power_data …
set_pg_nets ….
set_power_pads ….
analyze_rail ….
```


© 2013-2023 All Rights Reserved.

```
# View results
read_power_rail_results...
report_power_rail_results...

# What-if analysis
scale_what_if_resistance -reset
scale_what_if_resistance -net VSS -global -layer M7 -scale 0.5
scale_what_if_resistance -net VDD -region {100 100 200 200} -layer M7 -scale 0.5
scale_what_if_resistance -net VDD -region {400 400 600 600} -scale 0.5

analyze_rail …

read_power_rail_results...
report_power_rail_results...
```

Scaling Static and Dynamic Currents

When power-grid optimization is not feasible in the regions of IRdrop and electromigration violations, you can choose to scale static or dynamic currents on-the-fly to do what-if analysis. This analysis can help determine whether the designer should reduce activity, simultaneous switching and placement in the design to meet IRdrop and electromigration threshold. You can scale static or dynamic currents using the "*What-If Analysis Setup - Current*" form. You have the option to scale current inside the design hierarchy and regions. You can interactively draw regions on the layout canvas. Additionally, there is an option to not scale currents for instances associated with clock network, however, this option is only available when scaling currents for design hierarchy. The current will be scaled for all components inside region.

© 2013-2023 All Rights Reserved.

What-If Current Analysis Form

Example TCL flow

```
# Import design
read_design ….

# Power calculation
set_power_analysis_mode ….
report_power

# Rail analysis
set_rail_analysis_mode ….

set_power_data …

set_pg_nets ….

set_power_pads ….

analyze_rail ….

# View results
read_power_rail_results...
report_power_rail_results...

# What-if analysis
scale_what_if_current -reset
scale_what_if_current -hierarchy ethernet_mac_1 -scale 0.5
scale_what_if_current -hierarchy ethernet_mac_2 -scale 0.5
scale_what_if_current -region {0 0 100 100} -scale 2.0

analyze_rail …
```


© 2013-2023 All Rights Reserved.

```
read_power_rail_results...
report_power_rail_results...
```

Creating Static and Dynamic Current Regions

You can specify current regions on the routed power-grid layer to analyze the impact of unplaced or placed custom macros (without accurate power-grid view) on the global power-grid. Both static and dynamic current regions can be specified using the "*What-If Analysis Setup - Regions*" form by interactively drawing box in the layout canvas. The dynamic current regions require PWL current format with time (ns) and current (mA) pairs. In addition, dynamic current regions require specification of internal and loading capacitance to be used for the region.

What-If Current Region Form

Example TCL Flow

```
# Import design
read_design ….
# Power calculation
set_power_analysis_mode ….
report_power
# Rail analysis
```


© 2013-2023 All Rights Reserved.

```
set_rail_analysis_mode ….
set_power_data …
set_pg_nets ….
set_power_pads ….
analyze_rail ….

# View results
read_power_rail_results...
report_power_rail_results...

# What-if analysis
create_current_region -region {40 663 573 861}  -layer M1 -current 10
create_current_region -region {403 873 573 1244}-layer M1 -current 5

analyze_rail …

read_power_rail_results...
report_power_rail_results...
```

Scaling Capacitance

During dynamic analysis, you can perform what-if analysis to scale capacitance to see its impact on dynamic IRdrop caused by simultaneous switching currents. This feature can be used to perform what-if decap analysis by scaling intrinsic capacitance in the region to estimate decap optimization efforts. Additionally, you have the option to scale loading and power-grid capacitance. The capacitance scaling can be performed in GUI using the "*What-If Analysis Setup - Capacitance*" form.

© 2013-2023 All Rights Reserved.

What-If Capacitance Analysis Form

Example TCL Flow

```
# Import design
read_design ….

# Power calculation
set_power_analysis_mode ….
report_power

# Rail analysis
set_rail_analysis_mode ….

set_power_data …

set_pg_nets ….

set_power_pads ….

analyze_rail ….

# View results
read_power_rail_results...
report_power_rail_results...

# What-if analysis
scale_what_if_capacitance -intrinsic_cap -region {0 0 100 100} 2.0
scale_what_if_capacitance -intrinsic_cap -region {200 300 400 500} 2.0

analyze_rail …

read_power_rail_results...
report_power_rail_results...
```


© 2013-2023 All Rights Reserved.

Creating Virtual Shapes

You can create what-if wires/vias to fix voltage drop and analyze the new grid for various type of existing analyses. The analysis with what-if wires/vias can be done in both early and signoff mode rail analysis. In the sign-off mode what-if analysis, only possible virtual vias between what-if wires and design original wires are automatically generated. You can also skip the generation of these automatic “whatif” vias using the `-era_skip_virtual_via_by_type` parameter of the `set_rail_analysis_mode` command. The virtual vias are dropped using the default layer cut size and cut spacing.

Virtual shapes can be created in GUI using the "*What-If Analysis Setup - Virtual Shape*" form.

What-If Virtual Shape Creation Form

Example TCL Flow

```
# Import design
read_design ….
# Power calculation
set_power_analysis_mode ….
report_power
```


© 2013-2023
All Rights Reserved.

```
# Rail analysis
set_rail_analysis_mode ….
set_power_data …
set_pg_nets ….
set_power_pads ….
analyze_rail ….

# View results
read_power_rail_results...
report_power_rail_results...

# What-if analysis

create_what_if_shape -type wire -net VDD -area {391.000000 663.000000 407.000000
672.000000} -layer METTOP -add
create_what_if_shape -type via -net VDD -area {513.000000 497.700000 517.000000
499.100000} -layer {MET5 METTOP} -add –method auto

#optional step, the generated file can be used with set_rail_analysis_mode ….-
what_if_shapes_file
create_what_if_shape -save whatif.shapes

#enable what-if shapes analysis using set_rail_analysis_mode
set_rail_analysis_mode ….-import_what_if_shapes ..

or

set_rail_analysis_mode ….-what_if_shapes_file whatif.shapes

analyze_rail …

read_power_rail_results...
report_power_rail_results...
```


© 2013-2023 All Rights Reserved.

What-If Analysis Flow

This section describes how to set up the what-if analysis mode and then run rail analysis.

Required Input Files

- Design data (LEF, netlist, DEF)
- Power database generated during power calculation
- State directory created during rail analysis

Procedure

To perform what-if analysis, complete the following steps:

1. Load the design data and then run rail analysis. Refer to Static Power Analysis Flow.
2. Select *Power & Rail* - *Run Rail Analysis*.

   The Run Rail Analysis form appears.
3. Click the *Advanced* tab.
4. Click the *What-If Analysis Setup* button.

   The What-If Analysis Setup form appears.
5. Specify the resistance scaling options through the command line or through the GUI.

   GUI:

   a. Click the *Resistance* tab.

   b. Select the *Scale Resistance* check box.

   c. Select the *Auto Scale Adjacent Via Layers* check box.

   d. Select a layer from the *Layer Name* drop-down list.

   e. Specify the scale value in the *Scale* field.

   f. Specify the net name in the *Net Name* field.

   g. Click the *Global* radio button to scale all the rings and stripes of the selected layer.

   h. Click the *Add* button to save the changes.

© 2013-2023 All Rights Reserved.

Tcl command:

```
scale_what_if_resistance -global -net VDD_AO -layer Metal4 -scale 4
-auto_scale_adjacent_via_layers true
```

6. Specify the current scaling options through the command line or through the GUI.

   GUI:

   a. Click the *Current* tab.

   b. Select the *Scale Hierarchy Currents* check box.

   c. Select the *Scale Global Current* check box.

   d. Specify the scale value in the *Scale* field.

   e. Click the *Add* button to save the changes.

   Tcl command:

```
scale_what_if_current -global -scale 10
```

7. Click *OK* in the What-If Analysis Setup form.
8. In the Voltus console, re-run rail analysis.

   Tcl command:

```
analyze_rail -output ./staticRailResults/whatif -type domain ALL
```

9. Select *Power & Rail - Power & Rail Plots* to open Power & Rail Plots form (Rail Results Viewer).
10. Click the *DB Setup* button to open the Power & Rail Setup form.
11. Click *Browse* next to the *Rail Database* field, and select the latest state directory from the `Rail State Directory` browser window.
12. Click *OK*.
13. In the Power & Rail Plots form, select the *Rail* radio button and select the *ir* rail analysis type from the drop-down list.
14. Review the analysis results in the layout window.

    Similarly, you can create current regions or change the resistance or current scale factors, and rerun rail analysis.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

17

Power Gate Analysis

- Overview on page 437
  - Types of Power Gates on page 438
  - Power Gate Analysis on page 439
- Power Gate Library Characterization on page 440
  - Fine-grain memory characterization on page 443
- Steady state analysis on page 446
  - Usage of Common Power Format (CPF) on page 446
  - Steadystate Power Analysis on page 448
  - Steadystate Rail Analysis on page 449
- Native Power-up Analysis on page 450
  - Setup Native Powerup Analysis using GUI on page 452

Overview

In deep sub-micron designs, power has become a major concern as cell leakage increases with shrinking dimensions. The leakage power has increased from roughly 10% of total power in 130nm to 35% in 90nm to 55% in 65nm designs. As a way to address power consumption, designers use different methodologies to minimize power.

In modern low power IC designs, one common method of conserving power is to shut off portions of the chip when they are not being used. One of the ways to do this is power gating. In this design style, the portion of the chip to be powered down is driven by a local power supply rail. This local rail is then connected to a global power supply rail through one or more power gates (also known as power switches). Power gates are typically large transistors which can be turned off when the block is not being used. By shutting off the power supply to

© 2013-2023 All Rights Reserved.

the block, the leakage current of the block is significantly reduced. Since the block is shut off, the switching current of the block will be zero.

Types of Power Gates

There are two methodologies of using Power Gating using sleep transistors to control power consumptions.

- Coarse-grain sleep transistors are used to control the power to a large section of logic.
- Fine-grain sleep transistors are used to control the power of specific instances.

These two methods are shown below:

Two ways to reduce power consumption using sleep transistors

To implement coarse-grain sleep transistors in a standard place and route environment, power gates (containing sleep transistors) are inserted between the power rails of a block of logic and the main power rails of the design. This method offers the following benefits and drawbacks:

- Less area needed for implementation
- Greater control of leakage
- Degraded noise margins

© 2013-2023 All Rights Reserved.

- More difficult to guarantee performance
- Requires more custom design

Fine-grain sleep transistors are integrated as part of the circuit design of the cell and therefore provides greater control over each instance that contains a sleep transistor. This method offers the following benefits and drawbacks:

- Increased area for implementation
- Less control of leakage
- Guaranteed performance
- Requires a standard cell library with sleep transistor logic.

In all cases where sleep transistors are used to power down sections of logic, there is a need to return to the last known state when the power is re-applied. For coarse-grain sleep transistors, this can be achieved by using input latch cells that are outside the power down region. For fine-grain sleep transistors, this is typically achieved using specially designed state retention power gating (SRPG) cells for registers and latches.

Power Gate Analysis

When considering rail analysis of a design that includes power-gates, following analysis should be performed:

- Analysis of how the insertion of the power-gates impacts power, IR drop and electromigration when the power-gates are turned completely on (steady state analysis)
- Analysis of the total power savings and chip leakage when power-gates are turned off (steady state analysis)
- Analyze the time required to turn on all power-gates (powerup analysis) from its powered down state to its nominal voltage. The functional logic inside the power-gated block can not be activated until this time.
- Analyze the impact of rush current generated by the powering up logic on global (always-on) power-grid voltage (rush current analysis).

Voltus IC Power Integrity Solution addresses the power-gate analysis requirements using two approaches:

- A steady state analysis, where the state of the power-gates is not dynamically changing, instead, all power-gates are either on or off. It is used to understand how the insertion of the power gates impacts IR drop and electromigration when all power-gates are on and

© 2013-2023 All Rights Reserved.

power-savings when all power-gates are off. The static or dynamic power can be used to do steady state analysis.

- A rush current analysis, to understand the impact of turning on the power-gates dynamically to measure turn on time and analyze the impact of rush current on the IRdrop of the always-on power-grid.

Power Gate Library Characterization

During library generation of the power-gate cell, Voltus characterizes the power gate's on-resistance (Ron), the leakage current (Ileakage) when the power-gate is off, and the saturation current (Idsat) of the power gate. The information is then stored in the cell library, which is then used by Voltus during IR-drop analysis. Optionally you have the ability to override the Ron, leakage and saturation currents for the power-gates if desired.

The library generation of power-gate cells requires:

- LEF
- Extraction tech file
- Layermap of LEF and extraction tech file
- SPICE netlist and SPICE device models

  For power gate cell PGV characterization, the SPICE netlist with device x, y coordinates is recommended to be used to generate the detail EM view having the exact Ron resistors in the specific locations to represent the power gate transistors. It is important to have a clear and accurate resistance network of power gate cells so that the overall IR-drop and EM analysis of the low-power design is accurate.
- Power-gate cell's always on and switched pin

If SPICE netlist is not available, Voltus can not characterize Ron, Idsat and Ileakage parameters for the power-gate. In this case, you must specify estimated parameters for each power-gate cell manually.

To perform power-gate cell characterization:

1. Select *Power & Rail - Set PowerGrid Library Mode*.

© 2013-2023 All Rights Reserved.

The following form appears:

2. Set *Cell Type* to *Std Cells* to use SPICE subcircuit for view generation.
3. Specify *Filler Cell Names* and *Decap Cell Names* (optional) to perform decap optimization during rail analysis.
4. Select *Simulation*, and specify the netlist in the *Spice subckt netlist* field to characterize cell capacitance and current for the power-gate cell.
5. Specify *Spice Model* and *Corners* for device current and capacitance characterization.
6. Specify *Extraction Tech File* and *LEF Layermap*.
7. Specify *Power Pins*, *Voltages*, and *Ground Pins*.
   Ensure that the always-on and switched pin voltages are same.
8. Click the *Advanced* tab.

© 2013-2023
All Rights Reserved.

The following form appears:

9. *Cell List File* and *Cell Decap File* are optional fields.
10. Select *Power Gate Cell Characterization*.
11. Specify power-gate cell name
12. Specify always-on power pin name
13. Specify switch power pin name
14. If spice subckts are provided, Ron, Idsat and Ileakage will be disabled.
15. Click *Add* to add the power-gate cell information to the listbox.
16. Similarly add all power-gate cell information to the listbox.
17. Click *OK* to characterize the power-gate library.

Below is an example TCL script to generate power-gate cell library.

```
set_pg_library_mode
```


© 2013-2023
All Rights Reserved.

```
-extraction_tech_file RCgen.tch
-lef_layermap lefdef.map
-celltype stdcells
-power_pins { VDD 1.08 VDDO 1.08 VDDG 1.08 }
-ground_pins { VSS GND VSSG }
-cell_list_file built_cells_list
-temperature -40
-spice_models models/v1d1_usage.l
-spice_corners { ss_lib }
-spice_subckts { typical_max_m40.spice }
-powergate_parameters { { cell1 VDDG VDD } { cell2 VDDG VDD } }
generate_pg_library -output temp
```

Fine-grain memory characterization

For fine-grain power gate support, each cell has an internal switched rail. To use fine-grain power gates, the following needs to be done:

- Specify the `set_pg_library_mode -powergate_finegrain_simulation true` and `-powergate_parameters {{cell supply switched [Ron Idsat Ileak]}+}`
- Add a label for the switched rail manually using the `set_advanced_pg_library_mode -add_port_labels` *`file_name`* parameter. This file contains details about the label of a switched rail. Use this parameter if the switched rail is not labeled.

Below is an example TCL script to generate a power-gate cell library for fine grain cells:

```
set_pg_library_mode
-celltype macros
-spice_subckts { typical_max_m40.spice }
-spice_corners { ss_lib }
-spice_models models/v1d1_usage.l
-gds_files {mem1.gds.gz mem2.gds.gz} \
-gds_layermap gds.layermap \
-generate_port_using_gds defmbb
-extraction_tech_file RCgen.tch
-lef_layermap lefdef.map
-power_pins { VDD 1.08 VDDO 1.08 VDDG 1.08 }
-ground_pins { VSS GND VSSG }
-current_distribution {dynamic_simulation dynsim_trigger.txt}
-stop@via C01L
-cell_list_file built_cells_list
-powergate_parameters { { cell1 VDDG VDD } { cell2 VDDG VDD } }
-powergate_finegrain_simulation true
```


© 2013-2023 All Rights Reserved.

```
set_advanced_pg_library_mode -add_port_labels label
generate_pg_library -output temp
```

Note: You can also use the `-powergate_parameters` parameter to specify multiple switched nets. The following example specifies that the cell block1 with vdd as always-on net and vdds1 and vdds2 as switched nets:

```
-powergate_parameters {{block1 vdd vdds1} {block1 vdd vdds2}}
```

The power-gate library report can be generated from GUI or using TCL command as follows:

```
check_pg_library -output report -report /../pg.cl
```

The generated report lists power-gate cell, power pins, voltage, coupled capacitance, and characterized power-gate parameters like Ron, Idsat and Ileakage for each enable pins. The example report is shown below:

```
=====================================================================
Cell   PowerNets        Capacitance   PowerGrid Views
---------------------------------------------------------------------

cell1

   VDDG2(1.1000)     4.0e-13    EARLY(38 taps) EM(116 devices) IR(103 devices)
   VSS(0.0000)       1.7e-11    EARLY(328 taps)EM (648 devices)IR(637 devices)
   VDDG(1.1000)      5.5e-13    EARLY(79 taps) EM(170 devices) IR(158 devices)
   VDDP(1.1000)      1.5e-11    EARLY(206 taps)EM(1429 devices)IR(1329 devices)

INFO: This is a POWERGATE cell. The EM view of this cell.

Following Powergates are driven by enable Pin PD

Powergate M3175 has Idsat = 0.0001578A, Ileakage = 7.692e-07A, Ron = 2002 Ohm at
Voltage = 0.55
        Idsat = 0.0002834A, Ileakage = 2.168e-06A, Ron = 1428 Ohm at Voltage = 0.77
        Idsat = 0.0004235A, Ileakage = 5.335e-06A, Ron = 1151 Ohm at Voltage = 0.99
        Idsat = 0.0004977A, Ileakage = 8.054e-06A, Ron = 1058 Ohm at Voltage = 1.1
        Idsat = 0.0006528A, Ileakage = 1.72e-05A, Ron = 922.1 Ohm at Voltage = 1.32
        Idsat = 0.0008153A, Ileakage = 3.41e-05A, Ron = 826.8 Ohm at Voltage = 1.54

Powergate M3191 has Idsat = 0.002753A, Ileakage = 4.921e-05A, Ron = 111.4 Ohm at
Voltage = 0.55
        Idsat = 0.004798A, Ileakage = 0.0001291A, Ron = 82.66 Ohm at Voltage = 0.77
        Idsat = 0.00707A, Ileakage = 0.0002904A, Ron = 67.83 Ohm at Voltage = 0.99
        Idsat = 0.00827A, Ileakage = 0.0004162A, Ron = 62.74 Ohm at Voltage = 1.1
        Idsat = 0.01077A, Ileakage = 0.0007908A, Ron = 55.16 Ohm at Voltage = 1.32
        Idsat = 0.01339A, Ileakage = 0.001375A, Ron = 49.75 Ohm at Voltage = 1.54

    ............
    ............

INFO: This is a POWERGATE cell. The EARLY power grid view of this cell.

Following Powergates are driven by enable Pin PD

Powergate PG_PD_1 has Idsat = 0.0116432A, Ileakage = 0.000199917A, Ron = 26.382 Ohm
at Voltage = 0.55
    Idsat = 0.0203256A, Ileakage = 0.000525072A, Ron = 19.5343 Ohm at Voltage = 0.77
    Idsat = 0.029974A,  Ileakage = 0.00118294A, Ron = 16.0138 Ohm at Voltage = 0.99
    Idsat = 0.0350708A, Ileakage = 0.00169702A, Ron = 14.8069 Ohm at Voltage = 1.1
```


© 2013-2023 All Rights Reserved.

```
    Idsat = 0.0456912A, Ileakage = 0.003232A, Ron = 13.0116 Ohm at Voltage = 1.32
    Idsat = 0.0568212A, Ileakage = 0.0056364A, Ron = 11.7316 Ohm at Voltage = 1.54

Cell_Status: PASS
--------------------------------------------------------------------------------
```


© 2013-2023 All Rights Reserved.

Steady state analysis

For steady state analysis of power-gates, Voltus models the power-gate transistor as a linear resistance. The value of the resistance is determined from the device model during power grid library generation of the power gate. Voltus can analyze the impact of this resistance during static or dynamic power and rail analysis to ascertain whether sufficient power-gates are placed in the design to satisfy the current requirement of the switched block.

Voltus determines whether each power-gate cell is operating within the non-saturated (linear resistance) region, and reports when it is not. Power-gates operating outside of the non-saturation region are considered to be drawing too much current to be reliable. In addition to graphical plots, detailed power-gate text report and ECO file for Innovus is generated after the analysis.

Voltus can also be used to calculate the power savings and leakage power for the design when the power-gated block is powered down.

Usage of Common Power Format (CPF)

The power-gate implementation is part of the low-power MSMV design methodology. As power and voltage constraint become more aggressive, the low-power designs are becoming more complex with multiple voltage islands and power-gated domains to save power. The Common Power Format (CPF) allows single specification of power intent used throughout the design implementation and analysis. It is an ASCII file that captures various power domains, power logic and power modes of the low-power design. More information about CPF and how to create this specification file can be found in *Common Power Format User Guide* shipped with Innovus.

Using CPF enables power to be captured in the specification and the design process. This enables all the tools in the chain to utilize the same power description, facilitating easy exploration of different power architectures with each tool automatically seeing the same description of the power architecture. Also, by separating this information from the RTL, it enables easy re-use of IP or blocks that have the same functionality but different power profiles and this CPF can be handed off to physical implementation so that physical implementation and analysis team uses the same representation as the logic design team. Finally, power-aware DFT can be automated through this, enabling higher quality silicon and less power consumed on the tester.

Following is an example of a low-power design that can be described using CPF:

© 2013-2023 All Rights Reserved.

Example of low-power design

- RTL Design
  - DMA and DMA Bridge
  - Two MACs w/ identical RTL
- Define power domains
  - PDCore
  - PDMac1
  - PDMac2
- Define state retention
  - MAC1 optimized
  - MAC2 stores all regs
- Define power controls
- Define isolation logic
- Power modes:
- Technology data
  - Library cells
  - Operating conditions

Mode #	Mac1	Mac2
1	ON	OFF
2	OFF	ON
3	OFF	OFF
4	ON	ON

It is a small design with two instances of the MAC - for instance, one MAC could be for ethernet and another WLAN. We can use CPF to define the power domains VDDm (0.84v) and VDD (1.08v). We can also specify power-gated domains VDDlu and VDDau which are powering MAC I and MAC II. Notice how the two instances of the MAC are in different power domains - this is because typically both will not be used at the same time. We also use different state retention policies, where MAC2 retains state for all registers for fast recovery from shutdown, while MAC1 only retains state on a few registers to save power and area. If you were to specify this in the RTL, you would need to maintain two different versions of the MAC RTL, which also makes what-if analysis more complicated.

In addition to state retention, power controls and isolation logic are defined in CPF. The power modes are defined here, with different combinations of MAC1 and MAC2 being on and off. Finally the technology information is specified by associating different operating conditions, liberty files, and SDC to different power modes.

Voltus supports CPF to drive power and rail analysis on low-power design. You can load CPF using `read_power_domain -cpf` command described in chapter Importing Design.

© 2013-2023
All Rights Reserved.

Based on the power-mode (analysis view) specification in CPF, Voltus power and rail analysis associates timing libraries, PVT corner, SDC and state of the power domains, as depicted in the figure below:

CPF Power and Rail Analysis flow for Power Gates

Steadystate Power Analysis

Steady state power analysis can be done on a power-gated design to analyze total power savings and leakage power when the power-gated block is powered down. The powered down domain does not consume any switching power. The leakage power for the powered down block is only the leakage of the power-gates.

Static and dynamic power can be computed during steadystate analysis and fed to steadystate rail analysis to analyze the impact on IRdrop. The GUI and TCL instructions to run static and dynamic power analysis are same as in Static Power, IRdrop and EM Analysis and Dynamic Power and IRDrop Analysis.

If CPF is loaded, you can specify `set_power_analysis_mode -analysis_view` *`view`* to drive power analysis. It will automatically pick up associated timing libraries, SDC, power

© 2013-2023 All Rights Reserved.

net voltages and off domains to do power analysis. Note that if you want to specify "analysis view" using the `set_power_analysis_mode` command, the specified analysis view must be active during the session. This can be done by executing the following command in the Console tab in GUI, or TCL script:

```
set_analysis_view -setup view -hold view
```

Also the signal parasitic file at the desired RC corner must be read after activating the analysis view.

```
set_analysis_view -setup view -hold view
read_spef -rc_corner best design.spef
set_power_analysis_mode -method static -analysis_view view
set_default_switching_activity -input_activity 0.2
report_power
```

If CPF is not available, you can turn off power domain(s) manually by specifying the switched net using `set_power_analysis_mode -off_pg_nets` *`net_names`*.

Use `report_power -leakage` command to report leakage when the power domain is shut off.

Steadystate Rail Analysis

Steadystate rail analysis can be done on power-gated design to analyze whether placed power-gates are sufficient to accommodate the power need of the power-gated block. The GUI and TCL instructions to run steadystate rail analysis on power-gated design is same as described in Static Power, IRdrop and EM Analysis and Dynamic Power and IRDrop Analysis. The static or dynamic power can be fed in the form of current files of all associated nets to do this analysis.

If CPF is used, you can specify the analysis view using `set_rail_analysis_mode -analysis_view` *`view`*. This will automatically derive all power domains defined in CPF and voltage of all power nets in the domain.

```
set_rail_analysis_mode -analysis_view view
# following commands are executed automatically by the program
# set_pg_nets
# set_rail_analysis_domain
```

If a switched net is off in an analysis view, this information is also automatically derived from CPF. However, if CPF is not loaded, you must specify off power nets manually using `set_rail_analysis_mode -off_rails` *`net_name`*. In addition, you must specify all power nets, its voltages and domain using the `set_pg_nets` and `set_rail_analysis_domain` commands.

It is important to note that when analyzing power-gated domain, you do not have to specify a switched net in the domain definition (`set_rail_analysis_domain`). This definition

© 2013-2023 All Rights Reserved.

should only contain always-on net names. The switched net is automatically traced by the program and the state of this switched net is either driven by `set_rail_analysis_mode -off_rails` or directly from CPF.

In non-CPF steadystate analysis of power-gated domain,

```
set_rail_analysis_mode -off_rails VDDlu
```

```
set_pg_nets -net VDDau -voltage 0.84 -threshold 0.756 -tolerance 0.3
set_pg_nets -net VDDlu -voltage 0.84 -threshold 0.756 -tolerance 0.3
set_pg_nets -net VDDm -voltage 0.84 -threshold 0.756 -tolerance 0.3
set_pg_nets -net VDD -voltage 1.08 -threshold 0.972 -tolerance 0.3
set_pg_nets -net VSS -voltage 0.0 -threshold 0.084 -tolerance 0.3
```

```
set_rail_analysis_domain -name PDmac1 -pwrnets VDDm -gndnets VSS
set_rail_analysis_domain -name PDmac2 -pwrnets VDDm -gndnets VSS
set_rail_analysis_domain -name PDcore -pwrnets VDD -gndnets VSS
set_rail_analysis_domain -name ALL -pwrnets {VDDm VDD} -gndnets VSS
```

Once the analysis is complete, load the results using `report_power_rail_results` command and view the graphical plots "`power-gate current I/Idsat - pi`" and "`power-gate voltage -pv`". Both plots are explained in chapter Static Power, IRdrop and EM Analysis.

In addition, review the `pi.report` text file generated by the analysis.

Native Power-up Analysis

Voltus IC Power Integrity Solution supports native power-up analysis for performing rush current analysis without having to invoke UltraSim. The UltraSim-based power-up flow has the following disadvantages:

- Long simulation time for UltraSim
- Inability to take effect of the always-on net parasitics while doing power-up analysis
- Customizing the Spice deck and UltraSim options is cumbersome
- Need for a separate UltraSim license

The power-grid library generation engine of Voltus supports the native power-up flow, where the power-gates are modelled as voltage controlled current sources (VCCS). There is no use model change during library generation of the power-gate cell. The same data inputs and commands, as previously used, are required to generate the power-gate library (PGV).

© 2013-2023 All Rights Reserved.

The following diagram illustrates the modelling of a power-gate cell as VCCS:

The Power Analysis engine in Voltus writes out the arrival (rise/fall) times at the enable pin of the power-gate cells along with the transition times (rise/fall) for all the power-gate cells in the design. This information is then further used by the Rail Analysis engine downstream to do rush current analysis. At the power analysis stage, it is assumed that you will provide an accurate Liberty file for the power-gate cells along with proper constraints for the enable circuitry in the SDC file. The inbuilt timer will then use this information to calculate the correct arrival and transition times for the power-gate cells in the context of the whole design.

The Rail Analysis engine reads and uses the VCCS models for power switches along with the enable pin stimulus generated during the power analysis run. You can perform rush current analysis using the following methods:

- powering up the block of interest without taking into account the always-on net, that is, assuming an ideal always-on net. This flow only allows you to analyze one powering up net at a time.
- powering up the block(s) of interest taking into account the parastitcs of the always-on net, that is, assuming a non-ideal always-on net. This flow allows you to analyze more than one power-up net at a time. Therefore, you can simultaneously analyze the IR drop on the always-on net and power up one or more switched nets in the same run.

At the end of the analysis, Voltus reports the turn-on time of the switched net, peak current and a text based report having firing time of each switch. This report is similar to the text based report written out by the UltraSim-based flow. Also, a binary file is generated which has the current profile of the powering-up net. This can be viewed using the SimVision waveform viewer.

© 2013-2023 All Rights Reserved.

In native power-up analysis, multiple powering up rails are found in the `set_rail_analysis_mode` command. Each powering up rail requires a dedicated CPU and all powering up rails are analyzed in parallel. If you specify only one powering up rail, you do not require a multi-CPU license. You can use the `set_multi_cpu_usage` command to specify the number of CPUs required.

Note: Voltus does not support power-up analysis of ground switched nets.

Example

```
set_multi_cpu_usage -localCpu 2 ...
set_rail_analysis_mode -powering_up_rails {VDD1 VDD2} ...
```

Note: In the UltraSim based solution, power gates firing sequence is decided by forward propagating (simulating) PWL waveforms applied at the root pin. In the native power-up solution, power gates firing sequence is decided by SDC and timing analysis. You should specify an SDC (`set_input_delay` or `create_clock`) constraint at the root pin. Power gates fire at the arrival time on enable pin of each power gating cell.

Setup Native Powerup Analysis using GUI

To set up native power-up analysis, perform the following steps:

1. Perform library generation of power-gate cells. See Power-Grid Library Generation.
2. Set up and run Dynamic Power Analysis. See Dynamic Vector-based Power Analysis Flow.
3. Select *Power & Rail - Set Rail Analysis Mode*.

© 2013-2023 All Rights Reserved.

The following form appears:

4. Set *Analysis Method* as *Dynamic*.
5. Set *Accuracy*.

   XD uses LEF based views, and *HD* uses GDS based accurate views.
6. Specify Power-grid library. See Power-Grid Library Generation for details on generating power-grid libraries.
7. If CPF is loaded, select *Analysis View*.
8. Select *Switched Off Nets* to specify a list of nets to be excluded from analysis.

© 2013-2023
All Rights Reserved.

Note: *Switched Off Nets* and *Block Power Up Net* are mutually exclusive.

9. Select *Powering Up Nets*. This option takes into account the IR drop effect of always-on nets. You can specify a list of powering up nets. You can select *Block Power Up Net* to specify a single net for block power-up analysis.

   Note: *Powering Up Nets* and *Block Power Up Net* are mutually exclusive.

10. Specify *EM Analysis Models* if analyzing Electromigration

    Note: If you do not specify an Electromigration model file, the software will select Electromigration models from the QRC technology file. You can use the command `set_rail_analysis_mode -em_models` file to override these settings.

11. Select *Generate Movies*. This will provide the ability to view plots over simulation time.
12. Select the *Advanced* tab of the Set Rail Analysis Mode form.
13. Select *Generate Boundary Voltage File* if using Hierarchical view for the block
14. Specify GDS for *Flip-chip RDL* or *Full-chip GDS* if needed.
15. Fill in the decap fields as needed.
16. Click *OK* or *Apply*.

Example TCL script for native power-up analysis (powering-up nets)

#There is no change in the use model of power-gate library characterization and dynamic power analysis. See Power-Grid Library Generation to view the TCL script to generate a power-gate cell library. See Dynamic Vector-based Power Analysis Flow to view the TCL script for dynamic vectorbased or vectorless power analysis.

Note: You must ensure that you have an accurate Liberty file for the power-gate cell.

#This section covers the example TCL script to drive the native power-up analysis.

```
set_distribute_host -local -timeout 300
set_multi_cpu_usage -localCpu 2
read_lib -lef <LEF>
read_lib -max <maximum timing analysis libraries>
read_lib -min <minimum timing analysis libraries>
read_verilog <verilog netlist>
set_top_module <design name>
read_sdc <timing constraints>
read_def <DEF>
read_spef <RC parasitic files>
set_dynamic_power_simulation -period 60ns
report_power
```

```
set_rail_analysis_mode -method dynamic -accuracy hd -generate_movies false -
power_grid_library {Powergate/HEADBUF16MA10TR.cl Powergate/HEADBUF32MA10TR.cl
```


© 2013-2023 All Rights Reserved.

```
Powergate/memory.cl Powergate/accurate_stdcells.cl} -powering_up_rails {VDD_SW1
VDD_SW2}
```

#Specify names of powering-up rails.

```
set_power_data -format current ./Dynamic_vectorlessDir/dynamic_VDD.ptiavg
set_power_data -format current ./Dynamic_vectorlessDir/dynamic_VSS.ptiavg
set_power_data -format current ./Dynamic_vectorlessDir/dynamic_VDD_SW1.ptiavg
set_power_data -format current ./Dynamic_vectorlessDir/dynamic_VDD_SW2.ptiavg
```

In addition to current files for always-on nets, specify current files for powering-up rails.

```
set_pg_nets -net VSS -voltage 0 -threshold 0.02 -tolerance 0.3 -force
set_pg_nets -net VDD -voltage 1.1 -threshold 1.08 -tolerance 0.3 -force
set_rail_analysis_domain -name ALL -pwrnets VDD -gndnets VSS -threshold 0.3
set_power_pads -net VDD -format xy -file new_VDD_sources.ppl
set_power_pads -net VSS -format xy -file new_VSS_sources.ppl
analyze_rail -type domain -output Dynamic_rail ALL
```

View Rush Currents (powering-up nets)

```
view_dynamic_waveform -type current -instance_name total_current_VDD_SW1\
 -waveform_files dynamic_powerup_VDD_SW1_VDD.ptiavg
```

Important

You must specify the `view_dynamic_waveform -instance_name` parameter to view the rush current value for a specific instance. The `.pti` file contains the instance level current and the total current (`total_current_<powering_up_rail>`) waveforms.

Upon successful completion of rail analysis for rush current analysis, Voltus generates a power-up rush current file and a power-up text based report for the net under the `<net_name>` directory in the state directory of the power nets.

The software generates a text report file called `<powerup_net>_powering_up.report` in the `<net_name>` directory with measurement of turn on time and peak rush current. It highlights the wake-up time of individual power-gates in the design. If any power-gate is not turned on during simulation, it is also highlighted in the report. An example power-up report is shown below:

#This is a Power-up Report File.

```
Summary
=======


Simulation time = 1e-07s
Threshold (Vt) = 0.9V
Measured maximum rush current = 0.362636A
Measured wake-up time for switched net = 1.225e-08s
Number of power switches turned on in this simulation = 287 [of total 334]

Last power switch to turn-on in this simulation is 'HEADBUFS30RHT_390_4661_384:E'
at time 9.92015e-08s.
```


© 2013-2023 All Rights Reserved.

```
Detailed Report
===============
A. Reporting all the power gate instances which are turned on.


ORDER      TURN-ON TIME PEAK CURRENT INSTANCES
1         6.45e-10       0.0427877      HEADBUFS30RHT_406_5443_1:E
2         1.18e-09       0.0397882      HEADBUFS30RHT_422_5443_2:E
3         1.503e-09      0.0391968      HEADBUFS30RHT_438_5443_3:E
4         1.8075e-09     0.037844       HEADBUFS30RHT_454_5443_4:E
5         2.143e-09      0.036158       HEADBUFS30RHT_470_5443_5:E
.........................................
```

Voltus generates a current waveform called `dynamic_powerup_<powerup net>_<always-on net>.ptiavg` in the `<net_name>` directory.

To view peak rush current after analysis:

1. Open *Power & Rail* > *Report* > `Dynamic Waveforms` form and select *Power Waveforms*.
2. Load the current waveform file named as `dynamic_powerup_<powerup net>_<always-on net>.ptiavg`. For example, `dynamic_powerup_VDD_SW1_VDD.ptiavg`.
3. Select *Total Current* from the *Composite Waveform* drop-down listbox.

To view the probed waveforms on each instance, load the `dynamic_powerup_<powerup net>_<always-on net>.ptiavg` file in SimVision waveform viewer using *File* > *Open Database* form.

© 2013-2023 All Rights Reserved.

Rush Current - Powering-up Nets

Example TCL script for native power-up analysis (block power-up net)

#There is no change in the use model of power-gate library characterization. See Power-Grid Library Generation. For power analysis, you need to set the power analysis mode. You do not require to run power analysis. See Setup Native Powerup Analysis using GUI.

Note: You must ensure that you have an accurate Liberty file for the power-gate cell.

#This section covers the example TCL script to drive the dynamic IRdrop analysis.

```
read_lib -lef <LEF>
read_lib -typ <typical timing analysis libraries>
read_verilog <verilog netlist>
set_top_module <design name>
read_sdc <timing constraints>
read_def <DEF>

set_dc_sources <power_net_name> -power -voltage 1
set_dc_sources <ground_net_name> -ground
read_spef <RC parasitic files>

set_rail_analysis_mode -method dynamic -accuracy hd -generate_movies false -
power_grid_library {Powergate/HEADBUF16MA10TR.cl Powergate/HEADBUF32MA10TR.cl
```


© 2013-2023 All Rights Reserved.

```
Powergate/memory.cl Powergate/accurate_stdcells.cl} -block_powerup_rail
VCC_SWITCHED -powerup_sequence_file <file_name>
set_power_data -format current -repeat 100n
dynamic_result_native_powerup/dynamic_VCC_SWITCHED.ptiavg
```

#Only one power-up rail is supported at a time.

```
set_dynamic_rail_simulation -start 0ns -stop 100ns
set_pg_nets -net VCC_SWITCHED -voltage 1.0 -threshold 0.95 -tolerance 0.3 -force
set_power_pads -net VCC_SWITCHED -format xy -file /dev/null
analyze_rail -type net -output BLOCKPOWERUP VCC_SWITCHED
```

Note: In the native power-up analysis flow for a block power-up net, rail analysis of the power-up net is not done.

View Rush Currents (block power-up net)

```
view_dynamic_waveform -type current -instance_name total_current_VCC_SWITCHED
 -waveform_files dynamic_powerup_VCC_SWITCHED_VDD.ptiavg
```

You must specify the `view_dynamic_waveform -instance_name` parameter to view the rush current value for a specific instance. The `.pti` file contains the instance level current and the total current (`total_current_<powering_up_rail>`) waveforms.

Upon successful completion of rail analysis for rush current analysis, Voltus generates a power-up rush current file and a power-up text based report for the net under the `<net_name>` directory in the state directory of the power nets.

The software generates a text report file called `<powerup_net>_powering_up.report` in the `<net_name>` directory with measurement of turn on time and peak rush current. It highlights the wake-up time of individual power-gates in the design. If any power-gate is not turned on during simulation, it is also highlighted in the report. An example power-up report is shown below:

#This is a Power-up Report File.

```
Summary
=======
Simulation time = 1e-07s
Threshold (Vt) = 0.9V
Measured maximum rush current = 0.362267A
Measured wake-up time for switched net = 1.525e-08s
Number of power switches turned on in this simulation = 181 [of total 334]
Last power switch to turn-on in this simulation is
'dumy_decode_0/PSW_DSPARC_psoI_PD_DSPARC_1_HEADBUFS30RHT_1106_3671_230:E' at time
9.99953e-08s.
Detailed Report
===============


A. Reporting all the power gate instances which are turned on.
```


© 2013-2023 All Rights Reserved.

```
ORDER      TURN-ON TIME PEAK CURRENT INSTANCES
1         6.45e-10       0.0427877      HEADBUFS30RHT_406_5443_1:E
2         1.18e-09       0.0397882      HEADBUFS30RHT_422_5443_2:E
3         1.503e-09      0.0391968      HEADBUFS30RHT_438_5443_3:E
4         1.8075e-09     0.037844       HEADBUFS30RHT_454_5443_4:E
5         2.143e-09      0.036158       HEADBUFS30RHT_470_5443_5:E
.........................................
```

Voltus generates a current waveform called `dynamic_powerup_<powerup net>_<always-on net>.ptiavg` in the `<net_name>` directory. You can view the peak rush current using SimVision waveform viewer.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

18

Through-Silicon Via and System-in-Package

- Overview to Through-Silicon Via on page 461
  - Domain-based Through-Silicon Via Flow on page 463
- Overview to System-in-Package on page 466
  - Domain-based System-in-Package Flow on page 466
- Mixed Approach - a Combination of TSV and SiP on page 468

Overview to Through-Silicon Via

Through-Silicon Vias (TSVs) is a method of multi-die analysis, wherein, two or more chips are stacked vertically and are connected using via objects running through the stack. The via objects take signals/power/ground from the back-side of a die through its substrate to the main routing areas above the substrate. Back side metal routes enable routing on the back-side of the die to make better alignment for contact to a package. The TSV connects between the highest of the back-side metals through the substrate to the traditional Metal1.

Voltus IC Power Integrity Solution supports the following two types of TSV stacked die configuration:

- **faceDown Configuration (top metal to top metal)** - this configuration utilizes TSV only in the lower die with the upper die connecting through the top level regular metals. For example, Metal7 is using landing pads (like bump cells) on the highest metal aligned to connect between the 2 die.

© 2013-2023 All Rights Reserved.

- **faceUp Configuration (top metal to the back-side metal)** - this configuration utilizes TSV on both die. Signals power and ground go from the lower die's upper metal through back-side metal at TSVs of the upper die to reach its regular interconnect layers.

In both types of configuration, one die is the slave of the other die relying on the connections through the master die to draw current for power and ground from the package. The slave does not have direct connections to the package. Current is drawn from the package through the master die. As a result, the master die sees the slave as additional loading from a power/ground network perspective and sensitivity to IR drop is increased. Designing the power/ground network of the lower master die in order to satisfy the current demands of both itself and then also supplying or feeding current through to the upper slave die must be taken into account during the design phase. Analyzing and verifying the overall power and ground network is even more essential to ensure satisfactory power delivery and working die.

The following diagram illustrates a die containing TSV:

The following diagrams illustrate the stacked die configuration in detail:

© 2013-2023
All Rights Reserved.

Domain-based Through-Silicon Via Flow

An example of TCL commands for setting up and running domain-based TSV analysis is as follows:

```
set_rail_analysis_mode
-method static \
-die_mode multi-die \
-die_instance_name mother_die \
-accuracy hd\
-power_grid_library {./data/libgen/accurate_io_tier1/accurate_ios.cl \
./data/libgen/accurate_std_tier1/accurate_stdcells.cl} \
-report_via_current_direction false
-read_thermal_map map1.file
set_rail_analysis_mode
-method static \
-die_mode multi-die \
-die_instance_name daughter_die \
-accuracy hd \
-power_grid_library {./data/libgen/accurate_std_tier2/accurate_stdcells.cl \
./data/libgen/fast_block_tier2/fast_allcells.cl} \
-report_via_current_direction false
-read_thermal_map map2.file
```

Note: The `-die_mode` parameter specifies to perform multi-die analysis and the `-die_instance_name` parameter specifies the die instance name. You can use the `-read_thermal_map` parameter to use the distribution temperature defined in the specified thermal map file of each die generated from thermal analysis to do resistance scaling. The `set_rail_analysis_mode` command parameters are specified for both master and slave die.

```
set_pg_nets -net VDD -voltage 1.08 -threshold 0.9 -tolerance 0.3 -die_instance_name
mother_die -force
set_pg_nets -net VDD -voltage 1.08 -threshold 0.9 -tolerance 0.3 -die_instance_name
daughter_die -force
set_pg_nets -net VSS -voltage 0 -threshold 0.12 -tolerance 0.3 -die_instance_name
mother_die -force
set_pg_nets -net GND -voltage 0 -threshold 0.12 -tolerance 0.3 -die_instance_name
daughter_die -force
set_power_data -format current -scale 1
{./data/tier1_static_power/static_VDD.ptiavg} -die_instance_name mother_die
set_power_data -format current -scale 1
{./data/tier2_static_power/static_VDD.ptiavg} -die_instance_name daughter_die
set_power_data -format current -scale 1
{./data/tier1_static_power/static_VSS.ptiavg} -die_instance_name mother_die
set_power_data -format current -scale 1
{./data/tier2_static_power/static_GND.ptiavg} -die_instance_name daughter_die
set_power_pads -net VDD -format xy -file ./data/VDD_t1.pp -die_instance_name
mother_die
set_power_pads -net VDD -format xy -file ./data/VDD_t2.pp -die_instance_name
daughter_die
set_power_pads -net VSS -format xy -file ./data/VSS_t1.pp -die_instance_name
mother_die
set_power_pads -net GND -format xy -file ./data/GND_t2.pp -die_instance_name
daughter_die
```


© 2013-2023 All Rights Reserved.

```
c ./data/mother_die.def -die_instance_name mother_die
specify_def ./data/daughter_die.def -die_instance_name daughter_die
```

Note: Specify these commands for each die instance by specifying the instance name using the `-die_instance_name` parameter. For example, you can specify the power grid nets, power data, power pad file, and DEF file for each die instance.

```
set_rail_analysis_domain -name pd -pwrnets "mother_die/VDD daughter_die/VDD" -
gndnets "mother_die/VSS daughter_die/GND"
```

Note: Use the `set_rail_analysis_domain` command to specify the die name/ net name pairs for power and ground nets in the specified domain.

```
set_multi_die_analysis_mode -name tsv -die_list {mother_die daughter_die} \
-stacked_die_mapping ./data/pwr.mapping
```

The pad location file and the bump mapping file support embedded bumps in the hierarchical TSV design flow.

Note: Specify the `set_multi_die_analysis_mode` command to specify the multi-die analysis type (tsv), list of die instances, and the mapping file containing the connections between the two stacked die.

```
analyze_rail -type domain -output . pd
exit
```

Format of the Mapping File

The mapping file from Innovus is a virtual mapping that provides the connections between any 2 stacked die with communication directly between the die (master/slave configuration). Depending on the stacking die configuration, Innovus generates one or more mapping files. For example, if you are stacking 3 die, one on top of each other, Innovus generates 2 mapping files, die1-die2 and die2-die3. Alternatively, if you are stacking two smaller die on top of one large die, Innovus generates a single mapping file, die1-die2/die3.

The following diagram illustrates these two cases:

The format of the mapping file is as follows:

© 2013-2023
All Rights Reserved.

```
POWER {
<mapping_pair_1>
<mapping_pair_2>
…
<mapping_pair_i>
}
GROUND {
<mapping_pair_1>
<mapping_pair_2>
…
<mapping_pair_j>
}


SIGNAL {
<mapping_pair_1>
<mapping_pair_2>
…
<mapping_pair_k>
}
```

where, `<mapping_pair_*>` for power means:

```
<design1>          <net_in_design1>          <Vsrc_in_design1>
<design2>          <net_in_design2>          <Vsrc_in_design12>
```

The components are separated with a white space or tab. One line has one mapping pair.

`<design1> = instName1(cellName1)` for one of the die

`<net_in_design1>` is the name of the power net connected to the power bump in `design1`

`<Vsrc_in_design1>` is the name of the power bump in `design1`

Example

```
SIGNAL {
(tsv_top) A[0] A[0] mother_die(mother_die) A[0] A[0]
(tsv_top) A[1] A[1] mother_die(mother_die) A[1] A[1]
(tsv_top) A[2] A[2] mother_die(mother_die) A[2] A[2]
(tsv_top) A[3] A[3] mother_die(mother_die) A[3] A[3]
.......................................................
daughter_die(daughter_die) test_Q[8] test_Q[8] mother_die(mother_die) test_Q[8]
test_Q[8]
daughter_die(daughter_die) test_Q[9] test_Q[9] mother_die(mother_die) test_Q[9]
test_Q[9]
}
POWER {
daughter_die(daughter_die) VDD frontBump_1 mother_die(mother_die) VDD backBump_320
daughter_die(daughter_die) VDD frontBump_10 mother_die(mother_die) VDD
backBump_311
daughter_die(daughter_die) VDD frontBump_11 mother_die(mother_die) VDD
backBump_310
```


© 2013-2023 All Rights Reserved.

```
.....................................................
daughter_die(daughter_die) VDD frontBump_9 mother_die(mother_die) VDD backBump_312
}
GROUND {
daughter_die(daughter_die) GND frontBump_26 mother_die(mother_die) VSS
backBump_295
daughter_die(daughter_die) GND frontBump_27 mother_die(mother_die) VSS
backBump_294
.....................................................
daughter_die(daughter_die) GND frontBump_50 mother_die(mother_die) VSS
backBump_271
}
```

Overview to System-in-Package

System-in-Package (SiP) is similar to TSV multi-die analysis as it performs simultaneous analysis of two die. In case of SIP, the dies connect only through the package and there are no virtual connections between each die. In Voltus, to accurately simulate the interactive effect between each die, all the dies could be combined to perform IR drop analysis simultaneously and do not need to simulate one die at a time.

The following diagram illustrates the flow:

SiP can generate a single package model containing ports to multiple die. The naming of ports is driven by DEFs read by SiP from Innovus.

Domain-based System-in-Package Flow

An example of TCL commands for setting up and running domain-based SiP analysis is as follows:

```
set_multi_cpu_usage -localCpu 4
```


© 2013-2023 All Rights Reserved.

```
set_rail_analysis_mode -method dynamic \
-die_mode multi-die \
-die_instance_name EngineX4 \
-accuracy xd \
-power_grid_library { \
../data/EngineX4/libgen_bump/bump.cl \
../data/EngineX4/libgen_libs/tcb013ghp.cl \
../data/EngineX4/libgen_libs/tpz013g2.cl} \
-report_via_current_direction false
-read_thermal_map map1.file
```

```
set_rail_analysis_mode -method dynamic \
-die_mode multi-die \
-die_instance_name dtmf_chip \
-accuracy xd\
-power_grid_library { \
../data/dtmf/PGV/io_wc/fast_allcells.cl \
../data/dtmf/PGV/lp_wc/fast_allcells.cl \
../data/dtmf/PGV/macro_level_wc/fast_allcells.cl \
../data/dtmf/PGV/std_wc/accurate_stdcells.cl \
../data/dtmf/PGV/tech_only/techonly_library.cl} \
-report_via_current_direction false
-read_thermal_map map2.file
```

Note: The `-die_mode` parameter specifies to perform multi-die analysis, and the `-die_instance_name` parameter specifies the die instance name. You can use the `-read_thermal_map` parameter to use the distribution temperature defined in the specified thermal map file of each die generated from thermal analysis to do resistance scaling. The `set_rail_analysis_mode` command parameters are specified for both master and slave die.

```
set_pg_nets -net VDD -voltage 1.2 -threshold 1.08 -tolerance 0.3 -die_instance_name
EngineX4
set_pg_nets -net VSS -voltage 0 -threshold 0.12 -tolerance 0.3 -die_instance_name
EngineX4
set_pg_nets -net VDD_IO -voltage 0.81 -threshold 0.729 -tolerance 0.3 -
die_instance_name dtmf_chip
set_pg_nets -net Avdd -voltage 0.81 -threshold 0.729 -tolerance 0.3 -
die_instance_name dtmf_chip
set_pg_nets -net VDDL -voltage 0.72 -threshold 0.648 -tolerance 0.3 -
die_instance_name dtmf_chip
set_pg_nets -net VDDL_sw -voltage 0.72 -threshold 0.648 -tolerance 0.3 -
die_instance_name dtmf_chip
set_pg_nets -net VDD -voltage 1.2 -threshold 1.08 -tolerance 0.7 -die_instance_name
dtmf_chip
set_pg_nets -net VDD_sw -voltage 1.2 -threshold 1.08 -tolerance 0.7 -
die_instance_name dtmf_chip
set_pg_nets -net VSS -voltage 0 -threshold 0.072 -tolerance 0.3 -die_instance_name
dtmf_chip
set_pg_nets -net Avss -voltage 0 -threshold 0.072 -tolerance 0.3 -die_instance_name
dtmf_chip
```

```
set_power_data -format current -scale 1
{../data/EngineX4/voltus/dynamic_VDD.ptiavg
../data/EngineX4/voltus/dynamic_VSS.ptiavg} -die_instance_name EngineX4
set_power_data -format current -scale 1 {../data/dtmf/voltus/dynamic_VDD.ptiavg
../data/dtmf/voltus/dynamic_VDD_sw.ptiavg ../data/dtmf/voltus/dynamic_VDDL.ptiavg
```


© 2013-2023 All Rights Reserved.

```
../data/dtmf/voltus/dynamic_VDDL_sw.ptiavg
../data/dtmf/voltus/dynamic_VSS.ptiavg} -die_instance_name dtmf_chip
```

```
set_power_pads -net VDD -format padcell -file ../data/EngineX4/data/pad_cell.VDD -
die_instance_name EngineX4
set_power_pads -net VSS -format padcell -file ../data/EngineX4/data/pad_cell.VSS -
die_instance_name EngineX4
set_power_pads -net VDD -format padcell -file ../data/dtmf/data/pad_cell.vdd -
die_instance_name dtmf_chip
set_power_pads -net VDDL -format padcell -file ../data/dtmf/data/pad_cell.vdd -
die_instance_name dtmf_chip
set_power_pads -net VSS -format padcell -file ../data/dtmf/data/pad_cell.vss -
die_instance_name dtmf_chip
```

```
localCpu ../data/EngineX4/data/routed1_5.5via.def.gz -die_instance_name EngineX4
localCpu ../data/dtmf/data/test2.def.gz -die_instance_name dtmf_chip
```

Note: Specify these commands for each die instance by specifying the instance name using the `-die_instance_name` parameter. For example, you can specify the power grid nets, power data, power pad file, and DEF file for each die instance.

```
set_package -spice ../data/package/paksi_interconn.ckt -mapping
../data/package/paksi_interconn.ptmf
```

Note: You must specify this command as the two die connect through the package. This command is optional for TSV analysis.

```
set_multi_die_analysis_mode -name sip -die_list {EngineX4 dtmf_chip}
```

Note: Specify the `set_multi_die_analysis_mode` command to specify the multi-die analysis type (sip) and the list of die instances. SiP analysis does not require a mapping file as there are no connections between the two stacked die.

```
set_rail_analysis_domain -name ALL -pwrnets {dtmf_chip/VDD dtmf_chip/VDDL
EngineX4/VDD} -gndnets {dtmf_chip/VSS EngineX4/VSS};
```

Note: Use the `set_rail_analysis_domain` command to specify the die name/ net name pairs for power and ground nets in the specified domain.

```
analyze_rail -type domain -output ./ ALL
exit
```

Mixed Approach - a Combination of TSV and SiP

Voltus supports a mix of traditional SiP (multi-die connected through package) combined with stacked die using TSVs to enable die-die communication. The following diagram illustrates an

© 2013-2023 All Rights Reserved.

example of a mixed approach using traditional SiP techniques (both 3D-IC and side-by-side) as well as 3D-IC using TSV to enable communication between the stacked die:

In this example, the package connects to 4 die, Logic1, Logic2, Logic3, and RF1. RF1 is connected to the package using wire-bonding. The others are connected to the package using bumps. Logic1 (master) is connected to Mem1 (slave). Logic2 (master) is connected to Mem2 (slave). Mem1 and Mem2 are two instances of the same design. Logic3 and RF1 communicate only with the package and any inter-die communication occurs through the package. Mem1 communicates 100% through Logic1 and Mem2 communicates 100% through Logic2. Any communication between Mem1 and Mem2 to Logic3, RF1, or the board, occurs through the master chips Logic1 and Logic2, respectively.

The following diagram illustrates how electrically the interfaces can be handled using both stacked 3D-IC TSV die and traditional SiP stacked die:

In this example, the package connects to 3 die, namely Logic1, SOC1, and RF1. RF1 is connected to the package using wire-bonding. The others are connected to the package using bumps. The package model describes the electrical circuit between the die with a mapping file to connect between the die.

Here, one pair of stacked 3D-IC chips is utilizing TSV. Logic1 (master) is connected to Mem1 (slave). The mapping file from Innovus describes the virtual mapping that provides the connections between any two stacked die with communication directly between the die

© 2013-2023 All Rights Reserved.

(master/slave configuration). The RLC for microbump electrical characteristics is added in the voltage source file for the slave die.

© 2013-2023 All Rights Reserved.

19

IR Drop Impact on Timing

- Introduction on page 471
- Basic Voltus/Tempus Flow on page 472
- IR Drop Effect on Delays on page 474
- Voltus Setup for Power and IR Drop on page 475
- Library Setup on page 479
- Skew Analysis on page 482
- Jitter Analysis on page 485
- IR Drop Aware Critical Path Analysis Using Spice on page 491
- IR Drop Aware Clock Jitter Analysis Using Spice on page 493
- Example Flow Scripts on page 494

Introduction

IR drop on power rails can significantly affect cell and net delays. IR drop transitions dynamically vary with varying switching activities in the design.

© 2013-2023
All Rights Reserved.

The variation can span from within a clock cycle or clock cycle to clock cycle. The delay variations due to IR drop affects both cell and net delays in clock and data paths.

The delay variance in data paths and clock paths account in setup and hold timing checks. Tempus supports skew analysis to check IR drop effects on clock arrival times at different register pins. It also supports jitter analysis to check late and early arrival times at a particular register.

Basic Voltus/Tempus Flow

Voltus performs min and max IR drop analysis based on low and high switching activities and derives effective min/max operating voltages for every instance in the design. These instance

© 2013-2023
All Rights Reserved.

specific effective operating voltages are written into files which are referred as min or max IR drop files. The min/max IR drop files can be read in Tempus using the `update_delay_corner` command. These instance based min and max voltages are used during delay computation for respective corners and for performing variance checks using these delays.

Tempus also runs Spice simulations for the given timing path and compares delays against delays computed using the cell-level delay models. It creates a constant voltage source for each instance with voltage specified in the IR drop files. On the other hand, Voltus generates detailed power and ground voltage waveforms for each instance, which can be input into Tempus. Tempus uses these waveforms to create piece-wise linear voltage waveform for each instance.

© 2013-2023 All Rights Reserved.

IR Drop Effect on Delays

IR drop can affect both net delays and cell delays. Since the peak voltage is reduced, IR drop voltage swing range will be smaller than the nominal voltage, which affects the net delay and input slew at the receiver.

In the following figure, assume that the driver is operating at the nominal voltage V_{dd} and receiver is operating at V_{IRdrop} voltage.

Also assume the following trip points set in the timing library:

- Lower slew threshold: 20%
- Upper slew threshold: 80%
- Input threshold: 50%
- output threshold: 50%

The net delay with nominal voltage swing is smaller than the delay computed with IR drop voltage as the signal waveform propagates across the net. This example also shows the variance in input slew at the receiver. As the cell delay is the function of input slew and output load, the cell delay will also be affected.

Apart from input slew, the cell delays derived from the library which is characterized for nominal voltage will be derated to IR drop voltages. For accurate delay modeling, it is

© 2013-2023 All Rights Reserved.

recommended to use multiple libraries characterized with different operating voltages. These libraries are referred as `triLibs` in this document.

Tip

IR drop files created in Voltus includes IR drop voltages for all the instances in a design. However, it is important to specify nominal voltage for the design using `create_op_cond` and `set_op_cond` commands. Otherwise, I/O ports assumed to be at same voltage as the first cell in their fan-out cone causes inconsistency with the spice netlist created in Tempus.

Voltus Setup for Power and IR Drop

Clock jitter analysis requires minimum and maximum voltage for each instance in the design. To achieve this, IR drop analysis is run with minimum and maximum activity which translates to min and max power for the design. This power is fed to IR drop analysis to generate min and max voltage for each instance in the design, as shown in the following figure:

The min activity during power analysis is defined by turning off activity of combinational logic and toggling only clock network. This is achieved by setting negligible activity (1%) on primary inputs and output of sequential logic. If the design has gated clock, it should be enabled. Refer to the `set_default_switching_activity` command in the script below. Note that

© 2013-2023 All Rights Reserved.

specifying 0% activity does not work at this time, as power analysis will revert to default activity of 20% in this case.

The max activity during power analysis is defined as expected average activity on primary inputs and output of sequential logic. In addition, gated clocks are disabled.

Alternately, if VCD is available with low and high activity cycles, it can be specified in Voltus using `read_activity_file` command with start and stop times. Optionally, you can find high or low activity cycles by profiling VCD file. For information on vector profiling, refer to Dynamic Vector-based Power Analysis Flow.

Example Voltus script for the flow is shown below.

```
#######################################################
# Voltus Script
# Purpose: Dynamic analysis at min and max activity
# Run
>voltus -files script.tcl
#######################################################
# Import design
read_lib -lef <lef files>
read_lib -min <min libraries>
read_lib -max <max libraries>
read_verilog <verilog netlist>
set_top_module <top cell>
read_sdc <timing constraints>
read_def <physical netlist>
read_spef <signal parasitics>

# Power analysis
set_power_analysis_mode \
 -method dynamic_vectorless \
 -corner max \
 -create_binary_db true \
 -write_static_currents true
# dynamic power @low activity
set_default_switching_activity \
 -input_activity 0.01 \
 -seq_activity 0.01 \
 -clock_gates 1.0 \
 -period 4.0
set_power_output_dir ./dynamic_power_min
report_power
# dynamic power @high activity
set_default_switching_activity \
 -input_activity 0.25 \
 -seq_activity 0.25 \
 -clock_gates 0.0 \
 -period 4.0
set_power_output_dir ./dynamic_power_max
report_power
```


© 2013-2023 All Rights Reserved.

```
# IR drop analysis
set_rail_analysis_mode \
 -method dynamic \
 -accuracy hd \
 -generate_movies true \
 -power_grid_library {stdcells.cl memories.cl}

# Power nets, domain and IR drop threshold
set_pg_nets \
 -net VSS \
 -voltage 0.0 \
 -threshold 0.05 \
 -tolerance 0.3

set_pg_nets \
 -net VDD \
 -voltage 1.0 \
 -threshold 0.95 \
 -tolerance 0.3

set_rail_analysis_domain \
 -name core \
 -pwrnets VDD \
 -gndnets VSS

# Power pad location
set_power_pads \
 -net VDD \
 -format xy \
 -file ../DATA/powerpads/VDD.pp

set_power_pads \
 -net VSS \
 -format xy \
 -file ../DATA/powerpads/VSS.pp

# IR drop anlaysis @min power
set_power_data \
 -format current \
 -scale 1 \
 { dynamic_power_min/dynamic_VDD.ptiavg \
 dynamic_power_min/dynamic_VSS.ptiavg }

analyze_rail \
 -type domain \
 -output ./dynamic_rail_min \
 core

# IR drop analysis @max power
set_power_data -reset
set_power_data \
 -format current \
 -scale 1 \
 { dynamic_power_max/dynamic_VDD.ptiavg \
 dynamic_power_max/dynamic_VSS.ptiavg }

analyze_rail \
 -type domain \
 -output ./dynamic_rail_max core

##################### END #######################################
```

The instance voltage file has `.iv` extension. It is saved under the IR drop results directory, for example, "`core_25C_dynamic_1/Reports/VDD_VSS.iv`". The `.iv` and `.iv.gif` files are located in the `Reports` folder under the state directory. The instance voltage is

© 2013-2023 All Rights Reserved.

determined using power and ground IR drop waveforms inside the switching window of the instance, as shown below. It is also referred to as effective instance voltage.

This file can be read in Tempus/Voltus using `update_delay_corner` command as discussed in the subsequent sections of clock jitter and skew analysis. The instance voltage can be displayed in the context of layout using the `set_power_rail_display` command as well.

Example:

```
read_power_rail_results -rail_directory core_25C_dynamic_1/VDD
set_power_rail_display -plot ivdn
```

In addition to instance voltage, IR drop waveforms are also stored inside the IR drop results directory, for example, `core_25C_dynamic_1/VDD/VDD.ptiavg` and `VSS/VSS.ptiavg`. These voltage waveforms can be used for critical path analysis using SPICE. If dynamic power and IR drop analysis is done using the vector-based method, you may be required to shift these waveforms during critical path analysis using the `create_spice_deck` command. This will be required to align IR drop waveform with the clock edges of SDC. The IR drop waveform for instance can be viewed using `view_dynamic_waveform` TCL command in Voltus, as shown below:

```
view_dynamic_waveform \
-type voltage \
-waveform_files \
{core_25C_dynamic_1/VDD/VDD.ptiavg core_25C_dynamic_1/VSS/VSS.ptiavg} \
-effective_voltage_waveform \
-instance_name inst1
```


© 2013-2023 All Rights Reserved.

You can use the `set_rail_analysis_mode -eiv_method {best worst}` command to generate min and max instance voltage files within the timing window file in Voltus. The min/max IR drop files are fed into Static Timing Analyzer (STA) to perform clock jitter analysis in Tempus.

To ensure compatibility with Tempus to run jitter analysis, the following command should not be used:

```
set_rail_analysis_mode -report_voltage_drop -eiv_detail_report -
eiv_threshold -eiv_max_instances
```

Library Setup

Multiple libraries characterized with different operating voltages (`triLibs`) can be read into Tempus. Delay calculator computes an instance delay by interpolating delays between two libraries that are close to IR drop voltage of that particular instance.

For example, if you load libraries characterized for 1.0 volts, 0.9 volts, and 0.8 volts, the delay calculator uses1.0v and 0.9v libraries for all instances with IR drop < 1.0v and > 0.9v.

The `triLibs` can be loaded into Tempus in the following three ways

© 2013-2023
All Rights Reserved.

Reading Libraries Using read_lib Commands

```
read_lib [list [list $libDir/max/scmetro_cmos10lp_hvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_hvt_ss_0p9v_125c.lib] \
$libDir/max/scmetro_cmos10lp_hvt_ss_1p08v_125c.lib] \
[list $libDir/max/scmetro_cmos10lp_lvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_lvt_ss_0p9v_125c.lib \
$libDir/max/scmetro_cmos10lp_lvt_ss_1p08v_125c.lib] \
[list $libDir/max/scmetro_cmos10lp_rvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_rvt_ss_0p9v_125c.lib \
$libDir/max/scmetro_cmos10lp_rvt_ss_1p08v_125c.lib]\
$libDir/max/pllclk_slow.lib \
$libDir/max/iometroil_cmos10lp_hvt_ff_1p1v_2p7v_2p5v_m40c.lib \
$libDir/max/ram_256x16A_ss_1p08v_125c_syn.lib \
$libDir/max/rom_512x16A_ss_1p08v_125c_syn.lib]
```

Libraries with same cells characterized for different voltages should be grouped into separate list. From the above example, libraries `scmetro_cmos10lp_hvt_ss_*`, `scmetro_cmos10lp_lvt_ss_*` and `scmetro_cmos10lp_rvt_ss_*` which has three libraries characterized for 1.0v, 0.9v and 0.8v are grouped into a list. The list can be mixed with other libraries like `pllclk_slow.lib`, which is not characterized with multiple operating voltages.

MMMC Setup

`triLibs` can be specified while creating library sets for particular corner.

```
create_library_set -name max_corner \
[list [list $libDir/max/scmetro_cmos10lp_hvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_hvt_ss_0p9v_125c.lib] \
$libDir/max/scmetro_cmos10lp_hvt_ss_1p08v_125c.lib] \
[list $libDir/max/scmetro_cmos10lp_lvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_lvt_ss_0p9v_125c.lib \
$libDir/max/scmetro_cmos10lp_lvt_ss_1p08v_125c.lib] \
[list $libDir/max/scmetro_cmos10lp_rvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_rvt_ss_0p9v_125c.lib \
$libDir/max/scmetro_cmos10lp_rvt_ss_1p08v_125c.lib]\
$libDir/max/pllclk_slow.lib \
$libDir/max/iometroil_cmos10lp_hvt_ff_1p1v_2p7v_2p5v_m40c.lib \
$libDir/max/ram_256x16A_ss_1p08v_125c_syn.lib \
$libDir/max/rom_512x16A_ss_1p08_125c_syn.lib]
```


© 2013-2023 All Rights Reserved.

read_design Configuration File

Libraries can be loaded into Tempus via configuration file which exists in the saved database. The following syntax should be used in the configuration file to load `triLibs` via `read_design`

```
set rda_Input (ui_timelib)
[list [list $libDir/max scmetro_cmos10lp_hvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_hvt_ss_0p9v_125c.lib] \
$libDir/max/scmetro_cmos10lp_hvt_ss_1p08v_125c.lib] \
[list $libDir/max/scmetro_cmos10lp_lvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_lvt_ss_0p9v_125c.lib \
$libDir/max/scmetro_cmos10lp_lvt_ss_1p08v_125c.lib] \
[list $libDir/max/scmetro_cmos10lp_rvt_ss_0p8v_125c.lib \
$libDir/max/scmetro_cmos10lp_rvt_ss_0p9v_125c.lib \
$libDir/max/scmetro_cmos10lp_rvt_ss_1p08v_125c.lib] \
$libDir/max/pllclk_slow.lib \
$libDir/max/iometroil_cmos10lp_hvt_ff_1p1v_m40c.lib \
$libDir/max/ram_256x16A_ss_1p08v_125c_syn.lib \
$libDir/max/rom_512x16A_ss_1p08v_125c_syn.lib]
```

It is recommended that for better accuracy, the difference between two adjacent libraries should not be more than 20%. Also better delay accuracy can be achieved by reducing this difference at lower operating voltages.

Tip

The voltage range of characterized libraries should cover min and max IR drop voltages. If IR drop voltage of an instance falls out side this range, extrapolation is not supported currently, instead Tempus adjusts the voltage to the closest value. For example, if libraries characterized for 0.8, 0.9 and 1.0 are read-in and a particular instance IR drop voltage is 1.1volt, Tempus computes the delays for this instance based on 1.0 volt.

© 2013-2023 All Rights Reserved.

Skew Analysis

Skew is the difference in arrival times of clock at launch and capture registers. In the on-chip variation mode, the launch flop uses late paths and capture flop uses early paths. Late path delays are computed based on IR drop observed during high switching activity and Early path delays are computed based on IR drop observed during low switching activity.

High switching activity results into higher voltage drop means lower operating voltages. Lower operating voltages result in larger net/cell delays. Low switching activity results into lower voltage drop means higher operating voltages (but less than nominal voltage). Higher operating voltages result in smaller net/cell delays.

Clock skew analysis can be performed in Tempus using the `report_clock_timing -type skew` command.

Clock Skew Report for a Register Pair

Without IR Drop Delays

```
Skew Latency Clock Pin
-----------------------------------------------------------------------------
      1.079   rise DTMF_INST/RESULTS_CONV_INST/upper1336/\dout_reg[7]/CK
0.802 0.278   rise DTMF_INST/RESULTS_CONV_INST/LP_OBS_DFT_tpi_flop/CK
```


© 2013-2023
All Rights Reserved.

With IR Drop Delays

```
Skew Latency Clock Pin
---------------------------------------------------------------------------
      1.124   rise DTMF_INST/RESULTS_CONV_INST/upper1336/\dout_reg[7] /CK
0.824 0.300   rise DTMF_INST/RESULTS_CONV_INST/LP_OBS_DFT_tpi_flop/CK
```

`report_clock_timing` command has the `-verbose` option, which generates detailed reports.

Detailed Report without IR Drop Delays

```
Clock: m_rcc_clk
Skew Latency Clock Pin
---------------------------------------------------------------------------
      1.079   ri   DTMF_INST/RESULTS_CONV_INST/upper1336/\dout_reg[7] /CK
0.802 0.278   ri   DTMF_INST/RESULTS_CONV_INST/LP_OBS_DFT_tpi_flop/CK


Launch Clock latency Path
Clock Fall Edge 8.000
= Beginpoint Arrival Time 8.000
Other End Path:
-------------------------------------------------------------------------------
Instance Arc Cell Delay Clock Latency
-------------------------------------------------------------------------------
DTMF_INST/TEST_CONTROL_INST/g40 Y v 0.000
DTMF_INST/RESULTS_CONV_INST/g6001 A v -> Y ^ INVX10MTH 0.277 0.277
DTMF_INST/RESULTS_CONV_INST/RC_CGIC_INST CK ^ -> ECK ^ TLATNTSCAX12MTH 0.800 1.077
DTMF_INST/RESULTS_CONV_INST/upper1336/\dout_reg[7] CK ^ SDFFQNX1MTH 0.002 1.079
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

Capture Clock latency Path
Clock Fall Edge 8.000
= Beginpoint Arrival Time 8.000
Other End Path:
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Instance Arc Cell Delay Clock Latency
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

DTMF_INST/TEST_CONTROL_INST/g40 Y v 0.000
DTMF_INST/RESULTS_CONV_INST/g6001 A v -> Y ^ INVX10MTH 0.277 0.277
DTMF_INST/RESULTS_CONV_INST/LP_OBS_DFT_tpi_flop CK ^ SDFFQNX1MTH 0.001 0.278
-------------------------------------------------------------------------------
```

Detailed Report with IR Drop Delays

```
Clock: m_rcc_clk
Skew Latency Clock Pin
---------------------------------------------------------------------------
      1.124   ri   DTMF_INST/RESULTS_CONV_INST/upper1336/\dout_reg[7] /CK
0.824 0.300   ri   DTMF_INST/RESULTS_CONV_INST/LP_OBS_DFT_tpi_flop/CK
```


© 2013-2023 All Rights Reserved.

```
Launch Clock latency Path
Clock Fall Edge 8.000
= Beginpoint Arrival Time 8.000
Other End Path:
-------------------------------------------------------------------------------
Instance Arc Cell Delay Clock Latency
-------------------------------------------------------------------------------

DTMF_INST/TEST_CONTROL_INST/g40 Y v 0.000
DTMF_INST/RESULTS_CONV_INST/g6001 A v -> Y ^ INVX10MTH 0.298 0.298
DTMF_INST/RESULTS_CONV_INST/RC_CGIC_INST CK ^ -> ECK ^ TLATNTSCAX12MTH 0.823 1.121
DTMF_INST/RESULTS_CONV_INST/upper1336/\dout_reg[7] CK ^ SDFFQNX1MTH 0.003 1.124
-------------------------------------------------------------------------------

Capture Clock latency Path
Clock Fall Edge 8.000
= Beginpoint Arrival Time 8.000
Other End Path:
-------------------------------------------------------------------------------

Instance Arc Cell Delay Clock Latency
-------------------------------------------------------------------------------
DTMF_INST/TEST_CONTROL_INST/g40 Y v 0.000
DTMF_INST/RESULTS_CONV_INST/g6001 A v -> Y ^ INVX10MTH 0.284 0.284
DTMF_INST/RESULTS_CONV_INST/LP_OBS_DFT_tpi_flop CK ^ SDFFQNX1MTH 0.016 0.300
-------------------------------------------------------------------------------
```

Delay reported for instance `DTMF_INST/RESULTS_CONV_INST/g6001` without IR drop is 0.277ns for both late and early paths. When min/max IR drop files are provided, IR drop caused the instance `DTMF_INST/RESULTS_CONV_INST/g6001` delay to increase from 0.277 to 0.298ns for late path (using max delays) and 0.284ns for early path (using min delays).

Overall there is 22ps timing jump due to IR drop.

© 2013-2023 All Rights Reserved.

Jitter Analysis

Jitter is the difference in the late and early arrival times of clock phase at a register clock pin. Latest arrival time can be calculated based on the cell and net delays of longest clock path to the register clock pin during the high switching activity. Earliest arrival time can be calculated based on the cell and net delays of shortest clock path to the register clock pin during the low switching activity.

The Voltus Clock Jitter Analysis flow leverages the power and rail capabilities of Voltus, and integrates it with Tempus static timing analysis and Spectre simulation capabilities to perform the full analysis using a single script. It enables you to compute clock jitter for all the clock end points or user-defined end points. This flow includes profiling the cycle-to-cycle EIV values, applying the worst-case cycle-to-cycle EIV values for two simulations, measuring the delta delays for each of the clock tree cells, and computing the effective jitter for the clock path.

© 2013-2023 All Rights Reserved.

The following diagram shows cycle-to-cycle EIV profiling in Voltus:

This flow includes the following steps:

1. Creates the power profile using the vectorless or vector-based dynamic power calculation method.
2. Generates cycle-to-cycle EIV report file. The following command is used to generate the cycle-to-cycle EIV file:

   ```
   set_rail_analysis_mode -eiv_average_per_window_list
   clock_network.insts
   ```

   The following is a snippet of the `clock_network.insts` input file:

   ```
   INST CLK_L2_I1
   INST CLK_L2_I13
   ...
   ```

   The output EIV format appears as follows:

   ```
   #INST_NAME        START_TIME:END_TIME                  EIV_AVG
   clk_L2_I1  1.32e-09:1.78e-09 1.06354 6.38e-09:6.8e-09  1.06217
   clk_L2_I13 1.32e-09:1.68e-09 1.06273 6.38e-09:6.72e-09 1.06854
   ```
3. Reads the IR drop generated by Voltus, profiles and identifies the cycle-to-cycle worst IR drop, and uses the worst two cycles to do the jitter analysis.
4. Calculate jitter for all the clock end points or user-defined end points.

© 2013-2023 All Rights Reserved.

Performing Clock Jitter Analysis

This section describes the various inputs and outputs of the clock jitter analysis flow.

Required Input Files

- Design data (LEF, netlist, DEF)
- Technology PGV
- EIV File (generated from IR drop analysis)
- Clocks or Path Collection
- Spice netlist and model file
- LSF job details (queue resources for simulation)

Procedure

To perform jitter analysis, complete the following steps:

1. Load the design data.

   Tcl command:

```
read_lib -lef $lefs
read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz
read_sdc ../design/timing.sdc
```

2. Specify the SPEF file for signal net parasitic information.

   Tcl command:

```
read_spef -rc_corner  RC_wc_125 -decoupled\
../design/postRouteOpt_RC_wc_125.spef.gz
```

3. Specify the directory where the power results will be stored.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_power_output_dir dynVecLessPowerResults
```

4. Define the power analysis mode.

Tcl command:

```
set_power_analysis_mode \
    -method                      dynamic_vectorless \
    -enable_state_propagation    true \
    -create_binary_db false \
    -report_instance_switching_info all \
    -power_grid_library         { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } -corner max
```

5. Specify jitter frame and simulation time.

Tcl command:

```
set_default_switching_activity -name low -input_activity 0.05 \
      -seq_activity 0.1 -clock_gates_output 0.2 \
set_default_switching_activity -name high -input_activity 0.1 \
      -seq_activity 0.2 -clock_gates_output 2.0 \
set_dynamic_power_simulation -period 15ns -resolution 50ps \
      -activity_pattern {0 high 5ns low 10ns high}
```

6. Define the rail analysis mode.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_rail_analysis_mode \
    -method                     dynamic \
    -accuracy                   hd \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } \
    -eiv_method { worst best avg } \
    -work_directory_name VOLTUS_OUTPUT/extraction_output \
    -report_power_in_parallel true  \
    -eiv_average_per_window_list all_cell.list
```

7. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

   Tcl command:

```
set_pg_nets -net VDD -voltage 0.9 -threshold 0.810 -force
set_pg_nets -net VSS -voltage 0.0 -threshold 0.010 -force
set_rail_analysis_domain -name PD -pwrnets VDD -gndnets VSS
```

8. Define the location of the voltage sources for each power/ground net.

   Tcl command:

```
set_power_pads -net VDD -format xy -file ../design/VDD.pp
set_power_pads -net VSS -format xy -file ../design/VSS.pp
```

9. Run rail analysis.

   Tcl command:

```
analyze_rail -type domain -output OUTPUT/IRDROP PD
```

10. Perform Tempus delay calculation.

© 2013-2023 All Rights Reserved.

Tcl command:

```
report_timing
report_timing -net
```

11. Run jitter analysis.

Tcl command:

```
analyze_jitter -eiv_vdd_vss {VDD VSS} \
 -eiv_file /IRDROP/PD_25C_dynamic_1/Reports/VDD_VSS.win_avg.rpt \
 -model_file /../include.ss -subckt_file /../worst.spi \
 -spectre /../tools.lnx86/bin/spectre \
 -no_hard_stop_at_to_pins true -outdir jitter_output \
 -from i_REC_CLK -spectre_cmd_options {+aps +spice +mt=8 \
 +postlayout +dcopt  } -lsf_job_string {/grid/sfi/farm/bin/bsub \
 -q rnd -P VOLTUS:16.1:PV:regTest -W 140:00 -R \
 "OSNAME==Linux && OSBIT==64 && (OSREL==EE60||OSREL==EE50)" -Is } \
 -run_simulation \
 -levels_combine 2 \
 -max_spectre_lic 100
```

The Voltus console will display jitter analysis statistics, as shown in the following snippet:

```
Spectre simulation finishes
-----------------------------------------------------------------------------------------------------------------------
Start-point-> [through-point-> through-point] -> End-point                Cycle-to-Cycle Jitter Value (ps)
-----------------------------------------------------------------------------------------------------------------------
i_REC_CLK -> i_REC_CLK_user_added_diode/I                                                  0.002
-----------------------------------------------------------------------------------------------------------------------
Details:  Cell-input-pin (R|F)          Cell-output-pin (R|F)    EIV_C1(V)  Delay_C1(ps)  EIV_C2(V)  Delay_C2(ps)
-----------------------------------------------------------------------------------------------------------------------
Stage1    i_REC_CLK (R) ->            i_REC_CLK_user_added_      0.810      0.331         0.810      0.329
                                            diode/I (R)
```


© 2013-2023 All Rights Reserved.

Output

The jitter analysis directory will contain the following files:

File Name	Description
`summary.jitter`	A report that gives a list of all the start and end point pair clock branches and the corresponding cycle-to-cycle jitter value in picosecond. If the -through point is used, it is also given in the report.
`clock_branch1.jitter` `clock_branch2.jitter`	A detailed report for each start and end point pair branch. For each branch, it includes stage by stage details as follows: ■ Cycle-to-Cycle Jitter Value (ps) ■ Cell input pin name and output pin name with rise/fall transition ■ EIV value used for cycle 1 and cycle 2 ■ Simulations measured cell delay for cycle 1 and cycle 2

IR Drop Aware Critical Path Analysis Using Spice

Tempus supports IR drop aware spice analysis of timing paths. It writes out sensitized spice netlist for given timing path and invokes Spectre simulator to do spice simulations. Upon completion of spice simulations, Tempus reads back the spice report and generates delay comparison table between spice delays and Tempus internal delay calculator.

When IR drop files are read into Tempus, Tempus automatically creates a constant voltage source for each instance in the spice netlist with voltage specified in the IR drop file for that particular instance.

Running Spice Simulations with Constant IR Drop Voltage Per Instance

Spice deck created using `create_spice_deck` command after reading IR drop files automatically creates constant voltage source per each instance.

Here is the syntax of `create_spice_deck` command:

© 2013-2023 All Rights Reserved.

```
create_spice_deck [-report_timing {report_timing_options} \
 -power {power nets} -ground {ground nets} \
 -model_file <spice model files> \
 -subckt_file <spice sub circuit file> \
 -outdir <output directory for spice run>\
 -run_path_simulation
```

Example

```
create_spice_deck [-report_timing {-from i_5/CK -path_type full_clock}\
 -power {VDD VNW} -ground {VSS VPW} \
 -model_file "spmodels.sp" \
 -subckt_file "cmos10lphvt_m.cdl" \
 -outdir path_spice_deck_max \
 -run_path_simulation
```

To be able to run Spectre simulator, it should be in your search path or use `-spectre` option to specify full path to Spectre.

Spice Comparison Reports

```
+------------------+---------------+
 | Report | report_timing |
 +---------------------+---------------+

| Module | top |
| Timing | LATE |
| Slew Propagation | WORST |
| Operating Condition | nomi |
| PVT Mode | max |
| Tree Type | worst_case |
| Process | 1.000 |
| Voltage | 0.850 |
| Temperature | 125.000 |
| time unit | 1.000 ns |
| capacitance unit | 1.000 pF |
| resistance unit | 1.000 kOhm |

+---------------------------------+----------------------------+

Path 1 : MET Setup Check with Pin i_7/CK
Endpoint : i_7/D (v) checked with leading edge of 'CLK'
Beginpoint : i_5/Q (v) triggered by leading edge of 'CLK'

+----------------------------------------+

Other End Arrival Time 0.414
 - Setup 0.428
 + Phase Shift 10.000
 = Required Time 9.986

+----------------------------------------+

- Arrival Time 1.821
 = Slack Time 8.166

+----------------------------------------+

Clock Rise Edge 0.000
 = Beginpoint Arrival Time 0.000

+----------------------------------------+
```


© 2013-2023 All Rights Reserved.

```
Table : timing on the data path
+------+----------+-------------+-------+------+---------+-------+----------+
| net | instance | cell | pin | edge | arrival | delay | sArrival |sDelay |
+------+----------+-------------+-------+------+---------+-------+----------++
| clk1 | | | clk1 | ^ | 0.000 | | 0.000 | |
| n_0 | i_0 | CLKBUFX1MTH | i_0/Y | ^ | 0.135 | 0.135 | 0.132 | 0.132 |
| n_1 | i_1 | CLKBUFX1MTH | i_1/Y | ^ | 0.415 | 0.280 | 0.389 | 0.256 |
| n_2 | i_2 | CLKBUFX1MTH | i_2/Y | ^ | 0.642 | 0.227 | 0.647 | 0.259 |
| n_3 | i_3 | CLKBUFX1MTH | i_3/Y | ^ | 0.809 | 0.167 | 0.799 | 0.152 |
| n_4 | i_4 | CLKBUFX1MTH | i_4/Y | ^ | 1.047 | 0.238 | 1.002 | 0.203 |
| n_5 | i_5 | DFFSRX1MTH | i_5/Q | v | 1.600 | 0.554 | 1.639 | 0.636 |
| n_6 | i_6 | BUFX5MTH | i_6/Y | v | 1.821 | 0.220 | 1.862 | 0.223 |
| | i_7 | DFFSRX1MTH | i_7/D | v | 1.821 | 0.000 | 1.862 | 0.000 |
+------+----------+-------------+-------+------+---------+-------+----------+
Table : timing on the clock path
+------+----------+-------------+--------+------+---------+-------+---------+
| net | instance | cell | pin | edge | arrival | delay | sArrival |sDelay |
+------+----------+-------------+--------+------+---------+-------+----------+
| clk1 | | | clk1 | ^ | 0.000 | | 0.000 | |
| n_0 | i_0 | CLKBUFX1MTH | i_0/Y | ^ | 0.135 | 0.135 | 0.132 | 0.132 |
| n_1 | i_1 | CLKBUFX1MTH | i_1/Y | ^ | 0.415 | 0.280 | 0.389 | 0.256 |
| | i_7 | DFFSRX1MTH | i_7/CK | ^ | 0.415 | 0.000 | 0.389 | 0.000 |
+------+----------+-------------+--------+------+---------+-------+-----------+
```

IR Drop Aware Clock Jitter Analysis Using Spice

For considering instances that are common in the longest and shortest clock paths (such as instance U1 in the following figure), spice netlist has to be created separately for longest and shortest paths with irDrops calculated from high switching and low switching activities respectively.

These spice decks can be created using the following command:

© 2013-2023 All Rights Reserved.

```
create_spice_deck -report_timing {-from U1/CK -late -path_type full_clock} ...
create_spice_deck -report_timing {-from U1/CK -early -path_type full_clock} ...
```

During setup timing analysis launch clock path traverses through longest path, and hold analysis launch path traverses through shortest path. Tempus runs spice simulations on these paths and reports arrival times at the register pin computed by timing analysis engine and spice. Difference between the arrival times reported by these two simulations gives clock-jitter.

Example Flow Scripts

Non-MMMC Mode

```
#Load the design
read_design simple.conf

#Set analysis mode to on chip variations
set_analysis_mode -analysisType onChipVariation

# Read parasitics information
read_spef simple.spef

#Set nominal voltage.
# if required, you can create operation condition with required PVT values
#create_op_cond -name triLib_nominal -library_file dtmf_hvt_10v.lib -P 1 -V 1.0 -
T 125
set_op_cond typical -library elc_10v

# input max and min IR drop files
update_delay_corner -late_irdrop_file simple_max.iv
update_delay_corner -early_irdrop_file simple_min.iv

#Analyze setup and hold timing of critical paths
report_timing -late -path_type full_clock
report_timing -early -path_type full_clock

#Analyze clock skew
report_clock_timing -type skew -nworst 10 -verbose

#Analyze clock jitter
report_clock_timing -type jitter -nworst 10

#check data path delays from spice with orioinal STA report
create_spice_deck -report_timing {-from i_5/CK -late -path_type full_clock} \
 -outdir path_spice_deck_max -power {VDD VNW} -ground {VSS VPW} \
 -model_file "spmodels.sp" -subckt_file cmos10lphvt_m.cdl \
 -run_path_simulation

#check clock path delays from spice with original STA report
create_spice_deck -report_timing {-from i_5/CK -early -path_type full_clock} \
 -outdir path_spice_deck_min -power {VDD VNW} -ground {VSS VPW} \
 -model_file "spmodels.sp" -subckt_file $ file cmos10lphvt_m.cdl \
 -run_path_simulation
```


© 2013-2023 All Rights Reserved.

Setting Up a Delay Corner in MMMC Mode with TriLibs and IR Drop Files Per Corner

```
create_library_set -name max_corner \
 [list [list $libDir/max/scmetro_cmos10lp_hvt_ss_0p8v_125c.lib \
 $libDir/max/scmetro_cmos10lp_hvt_ss_0p9v_125c.lib] \
 $libDir/max/scmetro_cmos10lp_hvt_ss_1p08v_125c.lib] \
 [list $libDir/max/scmetro_cmos10lp_lvt_ss_0p8v_125c.lib \
 $libDir/max/scmetro_cmos10lp_lvt_ss_0p9v_125c.lib \
 $libDir/max/scmetro_cmos10lp_lvt_ss_1p08v_125c.lib] \
 $libDir/max/pllclk_slow.lib ]
```

```
create_rc_corner -name rc-cworst -cap_table myTech.4_bc.CapTbl -T 50
```

```
create_delay_corner -name dcWCCOM \
-rc_corner rc-cworst \
-late_irdrop_file high_active.iv\
-early_irdrop_file low_active.iv
```


© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

20

Tempus Power Integrity Analysis

- Tempus Power Integrity Analysis Overview on page 498
- Licensing Requirements on page 501
- Tempus Power Integrity Flow on page 502
- Performing Tempus PI Analysis on page 504
- Tempus PI Result Analysis and Reporting on page 511

© 2013-2023 All Rights Reserved.

Tempus Power Integrity Analysis Overview

The Tempus Power Integrity Analysis (Tempus PI) flow is a combination of the Tempus static timing analysis and Voltus power and IR drop technologies. Tempus and Voltus are built on a common database with common runtime and memory model, and are fully integrated for better accuracy. Therefore, the IR drop and static timing analysis information can easily be shared between the two tools.

The primary objective of the Tempus PI flow is to identify IR-sensitive timing paths, to accurately predict IR drop on these paths with vectorless analysis, and to enable automated fixing of these critical paths using Innovus and Tempus IR-ECO. The IR-sensitive timing paths are paths that have sufficient positive slack but degrade significantly with IR drop. This timing/sensitivity-driven IR analysis is multi-cycle (transient IR simulation), functional, and vectorless. It is not only functional for the paths under analysis but it also creates a worst-case but feasible switching scenario for neighbors around it. The IR drop and timing events are natively aligned within the tool because Tempus and Voltus share all information, such as IR drop, timing windows, and switching within the timing windows.

The following diagram illustrates the Tempus-Voltus integration in the Tempus PI flow:

© 2013-2023
All Rights Reserved.

The following are the key features of the Tempus PI flow:

- Identify IR-sensitive paths - the critical paths are timed using the traditional VDD margins on cell delays using max IR drop. The IR-sensitive paths are not timing critical paths but can be critical in the IR conditions. These IR-sensitive paths are identified among many paths, such as the resistance sensitive paths, power sensitive paths, timing sensitive paths, IR-sensitive paths for neighboring cells, and user-defined paths. The IR-sensitive critical paths are scheduled to switch in all cycles of the transient IR simulation.
- Create functional path switching scenarios - it creates realistic worst-case switching scenarios for the path and for the environment around it. In this flow, the software is aware of the delays along the path, and also the delays of the paths in the neighborhood. It creates switching scenarios that are timing slack and timing path aware.
- Identify proximity aggressors - it finds proximity aggressors around voltage and time-critical instances based on power-grid resistance and timing sensitivity. These proximity aggressors are modeled to switch at different times within the timing windows from cycle to cycle. The other aggressors are modeled using the vectorless algorithm to meet activity target and package effects.

© 2013-2023
All Rights Reserved.

- Calculate impact of delay/timing on voltage and time critical paths using cycle-cycle voltage statistic.

Following is a comparative analysis of a slack report (histogram of timing and number of paths/endpoints for the given design) from the traditional timing signoff flow and a slack report from the Tempus PI flow:

Traditional Timing Signoff (With IR Derate)		**Tempus Power Integrity**	
Slack Range	**Paths**	**Slack Range**	**Paths**
-50ps : -40ps	0	-50ps : -40ps	0
-40ps : -30ps	0	-40ps : -30ps	1
-30ps : -20ps	0	-30ps : -20ps	1
-20ps : -10ps	0	-20ps : -10ps	1
-10ps : 0ps	0	-10ps : 0ps	68
0ps : 10ps	0	0ps : 10ps	231
10ps : 20ps	6	10ps : 20ps	961
20ps : 30ps	99	20ps : 30ps	824
30ps : 40ps	446	30ps : 40ps	3
40ps : 50ps	1516	40ps : 50ps	2
50ps : 60ps	25	50ps : 60ps	0
60ps+	0	60ps+	0
Total Paths Identified	2092	Total Paths Identified	2092

Here, you can observe the following:

- There are a total of 2092 paths.
- In the case of the traditional timing signoff flow, there are no negative slacks, and the slack bin starts at the +50ps to +60ps range. The worst path is in the +10 to +20ps slack range.
- In the case of the Tempus PI true signoff flow, 71 paths are failing timing with negative slack values. The worst path is in the -30ps to -40ps range.

© 2013-2023 All Rights Reserved.

- Using the Tempus PI generated vectors with a more aggressive path analysis in the neighborhood, slack has moved to a negative range. This indicates a major shift in the overall delay.

  These 71 failing paths that can affect performance on silicon can be fixed using the Tempus PI ECO feature.

Licensing Requirements

Tempus PI can be invoked from either a Tempus or a Voltus session. This feature requires 3 licenses: Tempus XL (TPS200), Voltus XL (VTS200), and the Tempus Power Integrity Option (TPS600). The following table describes the licensing use model:

Tool Used	Base License	Command Used	Additional Licenses
Tempus	Tempus XL	-enable_tempus_pi true	Tempus PI + Voltus XL
Voltus	Voltus XL	-enable_tempus_pi true	Tempus PI + Tempus XL

© 2013-2023 All Rights Reserved.

Tempus Power Integrity Flow

The following diagram illustrates the Tempus PI flow:

To perform Tempus PI analysis, perform the following steps:

1. DB Signoff: Load the input design database.
2. Critical Path Analysis - Identify the critical paths that have timing violations in the Tempus STA stage.
3. IR drop Analysis - Report IR drop hotspots (Effective Instance Voltage - EIV) and identify worst violation victims and their aggressors in the Voltus IR Drop analysis stage.
4. Tempus ECO Optimization - Perform both downsizing of cells or aggressors that cause IR drop and ECO timing optimization in the Tempus ECO Fixing stage.

© 2013-2023 All Rights Reserved.

5. IR Fix Validation - Run Tempus PI analysis again to re-check the result after Tempus ECO.

This flow generates a database with better timing and potentially better IR drop.

© 2013-2023 All Rights Reserved.

Performing Tempus PI Analysis

This section describes the various inputs and outputs of the Tempus PI analysis flow.

Required Input Files

- Design data (LEF, timing libraries, netlist, SDC, DEF, SPEF)
- Technology PGV

Procedure

To perform Tempus PI analysis, complete the following steps:

Step 1: Tempus STA Analysis Using IR Derates

1. Load the design data.

   Tcl command:

```
read_lib -lef $lefs
read_view_definition ../design/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter
read_def ../design/super_filter.def.gz
```


© 2013-2023 All Rights Reserved.

Tcl command - Snippet of viewDefinition.tcl:

```
create_library_set -name LS_wc_1p00 -timing
./standard_cells_1p00_125C.lib

create_library_set -name LS_wc_0p90 -timing
./standard_cells_0p90_125C.lib ;# This library set will be used to
improve interpolation later in the flow

create_library_set -name LS_wc_0p85 -timing
./standard_cells_0p85_125C.lib

create_rc_corner -name RC_wc_125 -T 125


create_delay_corner -name DC_wc_1p00 -rc_corner RC_wc_125 -
library_set LS_wc_1p00

create_delay_corner -name DC_wc_0p85 -rc_corner RC_wc_125 -
library_set LS_wc_0p85

create_constraint_mode -name CM_wc -sdc_files ./design.sdc


create_analysis_view -name AV_wc_1p00 -constraint_mode CM_wc -
delay_corner DC_wc_1p00 ;# This is the non-derated/non-margined view
at 1.0v for IR drop analysis

create_analysis_view -name AV_wc_0p85 -constraint_mode CM_wc -
delay_corner DC_wc_0p85 ;# This is the traditional STA view at 0.85V
with voltage margined/derated for IR drop analysis
```

2. Specify the SPEF file for signal net parasitic information.

   Tcl command:

```
read_spef -rc_corner  RC_wc_125
../design/postRouteOpt_RC_wc_125.spef.gz
```

3. Define the analysis views to be used for setup and hold analysis, and update timing for the design.

   Tcl command:

```
set_analysis_view -setup {AV_wc_1p00 AV_wc_0p85} -hold {AV_wc_1p00
AV_wc_0p85}
```

Step 2: Voltus Power and IR Drop Analysis (EIV DB Generation)

1. Specify the directory where the power results will be stored.

   Tcl command:

```
set_power_output_dir dynVecLessPowerResults
```


© 2013-2023 All Rights Reserved.

2. Define the power analysis mode. The Tempus PI analysis flow is applicable only to the dynamic vectorless power calculation method.

   Tcl command:

```
set_power_analysis_mode \
    -method                      dynamic_vectorless \
    -disable_static true \
    -enable_state_propagation    true \
    -honor_combinational_logic_on_clock_net false \
    -report_instance_switching_info all \
    -power_grid_library         { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } \
    -enable_tempus_pi true \
    -ir_derated_timing_view AV_wc_0p85 \
    -enable_pba_for_tempus_pi true
```

   Note: The following are the parameters specific to the Tempus PI analysis flow:

   - `-enable_tempus_pi true` - specifies to enable the Tempus PI analysis flow.
   - `-ir_derated_timing_view` *`view_name`* - specifies the name of the timing view used to calculate timing slack and to rank the flops. This parameter must be specified if there are multiple views.

     The `-ir_derated_timing_view` parameter defines the timing view used to detect the IR-sensitive paths using a lower voltage than the actual analysis. This lower voltage simulates the IR drop for each instance in the design. The software uses this view to prioritize the end-points that need to be prioritized for switching.
   - `-enable_pba_for_tempus_pi true` - specifies to enable the Path-Based Analysis (PBA) for slack calculation. By default (`false`), the Graph-Based Analysis (GBA) is used.

3. Optional. Define default activities for various points in the design.

   Tcl command:

```
set_default_switching_activity -input_activity 0 \
    -period 4.0 -clock_gates_output_ratio   0.5 -seq_activity 0.2
```


© 2013-2023 All Rights Reserved.

The default switching activity for sequential cells (`-seq_activity`) in percentage (ratio 0.0 to 1.0) specifies the number of fully expanded data paths that will be scheduled to switch in the Tempus PI flow.

4. Optional, but recommended. Define the power simulation period and resolution.

   Tcl command:

```
set_dynamic_power_simulation -resolution 50ps -period 320ns
```

   The dynamic power simulation period defines the number of power analysis cycles to be run.

   Note: The user-defined sequential activity per cycle (`set_default_switching_activity -seq_activity`) and the total power analysis simulation period (`set_dynamic_power_simulation -period`) control the number of critical path end-points that will be covered per simulation cycle and the total simulation time. Allowing a larger power simulation period will allow Tempus PI to cover all the end-points.

5. Run power analysis.

   Tcl command:

```
report_power
```

6. Define the rail analysis mode.

   Tcl command:

```
set_rail_analysis_mode \
    -method                     dynamic \
    -accuracy                   hd \
    -analysis_view              AV_wc_1p00 \
    -power_grid_library { \
        ../data/pgv_dir/tech_pgv/techonly.cl \
        ../data/pgv_dir/stdcell_pgv/stdcells.cl \
        ../data/pgv_dir/macro_pgv/macros_pll.cl \
    } \
    -enable_rlrp_analysis       true \
    -verbosity true \
    -temperature 125 \
    -eiv_eval_window switching \
    -eiv_method {bestavg worstavg}
```


© 2013-2023 All Rights Reserved.

7. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

   Tcl command:

```
set_pg_nets -net VDD_AO -voltage 1 -threshold 0.95 -force
set_pg_nets -net VDD_column -voltage 1 -threshold 0.95 -force
set_pg_nets -net VDD_ring -voltage 1 -threshold 0.95 -force
set_pg_nets -net VDD_external -voltage 1 -threshold 0.95 -force
set_pg_nets -net VSS -voltage 0.0 -threshold 0.05 -force
set_rail_analysis_domain -name ALL -pwrnets { VDD_AO VDD_external }
-gndnets VSS
```

8. Define the location of the voltage sources for each power/ground net.

   Tcl command:

```
set_power_pads -net VDD_AO -format xy -file
../design/super_filter_VDD_AO.pp
set_power_pads -net VDD_external -format xy -file
../design/super_filter_VDD_external.pp
set_power_pads -net VSS -format xy -file
../design/super_filter_VSS.pp
```

9. Define the rail simulation resolution, and specify the location of the binary current files generated through dynamic power analysis (step 5).

   Tcl command:

```
set_dynamic_rail_simulation -resolution 50ps
set_power_data -scale 1 -format current [glob
dynVecLessPowerResults/*.ptiavg]
```

10. Run rail analysis.

   Tcl command:

```
analyze_rail -output ./dynVecLessRailResults -type domain ALL
```

   The Voltus console will display rail analysis statistics, as shown in the following snippet:

```
Rail Analysis completed successfully.
Finished Rail Analysis at 19:32:45 02/10/2019 (cpu=0:01:12, real=0:02:33, peak
mem=1697.00MB)
```


© 2013-2023 All Rights Reserved.

```
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:09:48,
real=1:06:51, mem=1697.00MB)


***Info: Found 70 instances with IR drop violations in design based on
eiv_method: Worst and eiv_eval_window: elapse.
```

Step 3: Tempus IR-Aware STA Using EIV Database (any IR Derates Removed)

1. Calculate IR-aware slack by specifying the IR analysis output as input to the Tempus run:

   Tcl command:

```
set_delay_cal_mode -early_irdrop_data_type best_average \
             -late_irdrop_data_type worst_average
update_delay_corner -name DC_wc_1p00 \
             -library_set {LS_wc_1p00 LS_wc_0p90 LS_wc_0p85} \
             -irdrop_data ./dyn_rail_phaseIII/ALL_25C_dynamic_1
set_analysis_view -setup {AV_wc_1p00} -hold {AV_wc_1p00}
update_timing -full
report_timing -max_paths 200000 -retime path_slew_propagation -
retime_mode exhaustive > pi_timing.pba.rpt
```

   The final analysis within Tempus PI is done at the non-derated voltage corner (that is, 1.0V in the `viewDefinition.tcl` example). Therefore, the `AV_wc_1p00` view is selected for analysis, and the corresponding delay corner (`DC_wc_1p00`) is updated to enable voltage tri-lib interpolation by specifying multiple library sets (`-library_set`)at various voltages (1.00V, 0.90V and 0.85V).

   Note: The following are the parameters specific to the Tempus PI analysis flow:

   - `-irdrop_data` *`directory_name`* - specifies the IR drop files or directory of files to apply to both the early and late delay calculation for the specified delay corner object.

Step 4: Tempus ECO Fixing

Important

In the 19.11 release, the Tempus PI feature can only identify the IR-sensitive timing critical paths and report timing with IR drop voltages. The ECO fixing of timing violations will be supported in a future release.

© 2013-2023 All Rights Reserved.

Output

The rail output directory will contain the following information:

Directory Name	Description
`<domainName>_<temperature>_dynamic_<X>`	For each run, an output directory is generated with the default name `<domainName>_<temperature>_dynamic_<X>`. `<domainName>` will be replaced with `ALL` if all domains are to be analyzed (`analyze_rail -output dir -type domain ALL`), or with `<netName>` if a power net is to be analyzed. The output directory contains state directories for each net, such as VDD and VSS. Each state directory contains the following: ■ `results` – a file that gives a summary of the rail analysis results for various plots, such as tc (tap current), rc (resistor current), and so on. ■ `Reports` – a folder that contains a list of useful report files that contain all power-grid integrity data. ■ Auto-generated GIF files of the rail analysis plots

The current work directory contains the timing report:

Directory Name	Description
`<timing_report>.rpt`	A timing report that provides information about the various paths in the design. The `report_timing > <timing_report>.rpt` command is used to generate the report and write it to the specified file name. For more information on Tempus commands, refer to *Tempus Text Command Reference*.

© 2013-2023 All Rights Reserved.

Tempus PI Result Analysis and Reporting

You can analyze the timing reports generated from the traditional timing signoff and the Tempus PI flows to determine the instances in the high IR drop region. When you compare the timing reports in the following snapshots, you will observe that the timing path that had positive slack in the traditional analysis flow has negative slack in the Tempus PI analysis flow:

Worst-case Path Analysis (Traditional)

```
#############################################################
#  Generated by:      Cadence Voltus IC Power Integrity Solution 19.11-e091_1
#  OS:                Linux x86_64(Host ID sjfdsl26)
#  Generated on:      Wed Jun 26 02:43:49 2019
#  Design:            super_filter
#  Command:           report_timing -retime path_slew_propagation -path_group pll_clk  > timing.rpt
#############################################################
Path 1: MET Setup Check with Pin AO1/SUM_reg_3_15/CK
Endpoint:   AO1/SUM_reg_3_15/D (^) checked with  leading edge of 'pll_clk'
Beginpoint: bypass             (^) triggered by  leading edge of 'io_clk'
Path Groups: {pll_clk}
Retime Analysis { Data Path-Slew  }
Other End Arrival Time         109.300 (-3.00S)  |  109.300 |  0.000
+ Other End Clock Edge          0.000
- Setup                         2.300 (+3.00S)  |  2.300 |  0.000
+ Phase Shift                  380.000
+ CPPR Adjustment               2.600 (+3.00S)  |  2.600 |  0.000
- Uncertainty                  20.000
= Required Time                469.600 (-3.00S)  |  469.600 |  0.000
- Arrival Time                 387.200 (+3.00S)  |  387.200 |  0.000
= Slack Time                   82.400 (-3.00S)  |  82.400 |  0.000
= Slack Time(original)         77.100 (-3.00S)  |  77.100 |  0.000
    Clock Rise Edge                    0.000
    + Clock Network Latency (Prop)   111.200 (+3.00S)  |  111.200 |  0.000
    = Beginpoint Arrival Time        111.200 (+3.00S)  |  111.200 |  0.000
     ----------------------------------------------------------------------------------------------------------------------
     Cell                                  Arc           Edge  Fanout  Load    Retime  Retime  Retime  Arrival  Voltage  Pin
                                                                           Slew    Incr    Delay   Time
                                                                                   Delay
     ----------------------------------------------------------------------------------------------------------------------
     -                                     CP ^          ^     6       8.409   23.000  -       -       111.200  0.898    pinA/CP
     FE_OFC1_bypass                        CP ^ -> Q ^   ^     -       4.224   13.400  0.000   32.100  143.300  0.898    pinA/Q
     ring/g1131                            -             ^     4       4.224   13.600  0.000   1.200   144.500  0.898    pinB/I
     ring/g1131                            I ^ -> Z ^    ^     -       23.220  66.800  0.000   31.500  176.000  0.898    pinB/Z
     column/g1131                          -             ^     12      23.220  72.700  9.000   25.700  201.700  0.898    pinC/I
     column/g1131                          I ^ -> ZN v   v     -       9.353   43.700  0.000   30.700  232.400  0.898    pinC/ZN
     ISO_RULE_ISO_HIER_INST_4/g1           -             v     11      9.353   46.100  1.500   8.700   241.100  0.898    pinD/B1
     ISO_RULE_ISO_HIER_INST_4/g1           B1 v -> ZN ^  ^     -       6.728   63.900  0.000   44.800  285.900  0.898    pinD/ZN
     AO1/add_94_37_I1/g70                  -             ^     1       6.728   64.100  8.100   10.400  296.300  0.898    pinE/I
     AO1/add_94_37_I1/g70                  I ^ -> Z ^    ^     -       29.605  84.600  0.000   42.000  338.300  0.898    pinE/Z
     AO1/add_94_37_I1/g702                 -             ^     1       29.605  98.400  20.000  48.900  387.200  0.898    pinF/D4 ->
     ----------------------------------------------------------------------------------------------------------------------
```


© 2013-2023 All Rights Reserved.

Worst-case path analysis (Tempus PI)

```
#################################################################
#  Generated by:      Cadence Voltus IC Power Integrity Solution 19.11-e091_1
#  OS:                Linux x86_64(Host ID sjfds126)
#  Generated on:      Wed Jun 26 02:43:49 2019
#  Design:            super_filter
#  Command:           report_timing -retime path_slew_propagation -path_group pll_clk  > timing.rpt
#################################################################
Path 1: MET Setup Check with Pin AO1/SUM_reg_3_15/CK
Endpoint:   AO1/SUM_reg_3_15/D (^) checked with  leading edge of 'pll_clk'
Beginpoint: bypass             (^) triggered by  leading edge of 'io_clk'
Path Groups: {pll_clk}
Retime Analysis { Data Path-Slew  }
Other End Arrival Time         107.700 (-3.00S)  |  107.700 |  0.000
+ Other End Clock Edge          0.000
- Setup                        11.300 (+3.00S)  |  11.300 |  0.000
+ Phase Shift                  380.000
+ CPPR Adjustment               3.300 (+3.00S)  |  3.300 |  0.000
- Uncertainty                  20.000
= Required Time                459.700 (-3.00S)  |  459.700 |  0.000
- Arrival Time                 485.100 (+3.00S)  |  485.100 |  0.000
= Slack Time                  -25.400 (-3.00S)  | -25.400 |  0.000
= Slack Time(original)        -46.750 (-3.00S)  | -46.750 |  0.000
     Clock Rise Edge                  0.000
     + Clock Network Latency (Prop)   112.100 (+3.00S)  |  112.100 |  0.000
     = Beginpoint Arrival Time        112.100 (+3.00S)  |  112.100 |  0.000
     ------------------------------------------------------------------------------------------------------------------------------
     Cell                                  Arc           Edge  Fanout  Load    Retime  Retime  Retime   Arrival  Voltage  Pin
                                                                           Slew    Incr    Delay    Time
                                                                                   Delay
     ------------------------------------------------------------------------------------------------------------------------------
     -                                     CP ^          ^     6       8.409   24.300  -       -        112.100  0.904    pinA/CP
     FE_OFC1_bypass                        CP ^ -> Q ^   ^     -       4.224   13.400  0.000   31.700   143.800  0.904    pinA/Q
     ring/g1131                            -             ^     4       4.224   13.000  0.100   1.100    144.900  0.880    pinB/I
     ring/g1131                            I ^ -> Z ^    ^     -       23.220  67.100  0.000   32.100   177.000  0.880    pinB/Z
     column/g1131                          -             ^     12      23.220  58.800  8.400   21.000   198.000  0.773    pinC/I
     column/g1131                          I ^ -> ZN v   v     -       9.353   46.400  0.000   42.900   240.900  0.773    pinC/ZN
     ISO_RULE_ISO_HIER_INST_4/g1           -             v     11      9.353   51.800  1.700   7.200    248.100  0.831    pinD/B1
     ISO_RULE_ISO_HIER_INST_4/g1           B1 v -> ZN ^  ^     -       6.728   69.100  0.000   49.500   297.600  0.831    pinD/ZN
     AO1/add_94_37_I1/g70                  -             ^     1       6.728   52.400  8.500   3.700    301.300  0.664    pinE/I
     AO1/add_94_37_I1/g70                  I ^ -> Z ^    ^     -       29.605  119.600 0.000   81.000   382.300  0.664    pinE/Z
     AO1/add_94_37_I1/g702                 -             ^     1       29.605  334.500 42.200  102.800  485.100  0.896    pinF/D4 ->
     ------------------------------------------------------------------------------------------------------------------------------
```


© 2013-2023 All Rights Reserved.

The following snapshot shows the top five register2register critical paths (*Timing SI* -> *Debug Timing* menu option) and the corresponding IR drop heat map (*Power Rail* -> *Power Rail Plots* menu option):

Following is the Tempus PI performance data for a design with 5.05M standard cell instances:

	36-CPUs @ 18cores*36-CPUs E5-2699 v3 @2.3GHz 46080KB	
Flow Steps	**Real Time**	**Peak Memory**
Update Timing (pre)	0h 37m 41s	31.2G
Report Power	2h 13m 51s	142.8G
Analyze Rail	1h 24m 35s	57.3G
Update Timing (post)	2h 14m 35s	57.7G
Total	6h 30m 42s	

© 2013-2023
All Rights Reserved.

© 2013-2023 All Rights Reserved.

21

IR-aware ECO Techniques

- IR Drop Aware Placement on page 516
  - Overview on page 516
  - Data Requirements for IR Drop Aware Placement on page 516
    - Hotspot Debugger on page 518
    - IR Aware Placement on page 520
    - Rechecking IR Drop and Performing Iterations on page 521
    - Rerouting the Signal Nets and Saving the Design on page 522
  - IR Drop Aware Placement Flow on page 517
  - Debugging IR Aware Placement on page 523
  - Sample Run Script on page 525
- Timing-aware IR Drop Fixing on page 527
  - IR-aware ECO Flow on page 528
  - Hotspot Debugger and ECO Analysis on page 530
  - Rechecking IR Drop and Result Analysis on page 531
  - Sample Run Script on page 534

© 2013-2023 All Rights Reserved.

IR Drop Aware Placement

Overview

Due to resistive power grid or simultaneous switching of close instances, there is reduction in voltage on VDD nets and increase in voltage on VSS nets. These deviation in voltage compared to the original values is IR Drop (on VDD nets) or Ground Bounce (on VSS nets). The reason IR drop analysis is so important is that the resistance increases with advanced nodes, and power increases with high operating frequencies, both resulting in IR drop violations. IR drop may cause timing issues if they are higher than 5%-10% of effective supply voltage, or cause functional failures if they are more than 20%-30% of effective supply voltage.

The following are some of the existing methods to fix IR Drop:

- Add power pads to the hot spot regions. This method is restricted by the package design.
- Add decap cells near aggressors. However, effective series resistance increases in case of advanced nodes, which makes the decap cells less effective.
- Add power straps, but this can only be done if sufficient routing resources are available.

This chapter covers the IR aware placement method for fixing IR drop issues. In this method, the aggressors are moved to a different location. This reduces current requirements at the hot spot regions, and therefore decreasing IR drop.

Data Requirements for IR Drop Aware Placement

The table below lists the inputs that are required for IR Drop Aware Placement in Voltus and Innovus:

Data
Files for loading the design
Post-CTS or post-routed Innovus database
Timing Libraries
Verilog
SDC

© 2013-2023 All Rights Reserved.

LEF + DEF
Files for the Rail Analysis stage
DEF
Power-grid Views
Dynamic Power Files
Power Pad Location Files
Power Domain and its associated Power Nets
Files for the Placement stage
Reports generated by Hotspot Debugger after Rail Analysis

IR Drop Aware Placement Flow

The following diagram illustrates the IR Drop Aware Placement analysis flow:

The flow has three fundamental steps:

1. Restore the post CTS or post-routed database using the following command:

   ```
   restoreDesign
   ```


© 2013-2023 All Rights Reserved.

2. Set up power and rail analysis modes and environment using the following commands:

```
set_power_output_dir dir_name
set_power_analysis_mode ...
set_default_switching_activity ...
set_dynamic_power_simulation ...
set_rail_analysis_mode ...
set_pg_nets ...
set_rail_analysis_domain ...
set_power_pads ...
```

3. Perform IR aware placement analysis using the following command:

```
setPlaceMode ...
refinePlace ...
ecoRoute
```

In this flow, the `refinePlace` command performs power and rail analysis, identifies hotspots, and refines the placement of instances. After checking for IR Drop violations, if there are still some critical violations, a second iteration of IR aware placement is done which is followed by IR Drop analysis.

Hotspot Debugger

In the IR Drop Aware Placement flow, you need to identify the hotspots with aggressors (instances) that demand high currents. These instances in the hotspot regions are reported by the `refinePlace` command, and the report is then used by Innovus to perform IR Aware Placement. The command divides the design into regions with the hotspots, and identifies aggressors (worst instances) in those hotspots, as shown in the following diagram:

The values of the following two parameters of the `debug_irdrop` command are key to get the best results for IR Drop improvement with lesser affect to timing:

© 2013-2023 All Rights Reserved.

- The `-nregion` *`value`* (default: `3`) parameter specifies the number of regions that the design needs to be divided into. The value of `-nregion` will depend on the number of hotspot regions that can be seen in the design. The higher the value of `-nregion`, the more likely the hotspot regions will be concentrated resulting in efficient results.
- The `-nworst_instances` *`value`* (default: `10`) are the number of instances that are reported in each regions. For `-nworst_instances`, a higher value will cause more movement of the instances and a smaller value will cause lesser movement of the instances.

Based on the values of `-nregion` and `-nworst_instances`, you can control the movement of instances and the number of instances that are being moved.

The hotspot regions are reported in the main Hotspot Debugger report (`report.rpt`), located in the `VoltusDebugIRdropResults` directory, generated after running the `refinePlace` command. The following is a snippet of the main report:

```
HOT SPOT DEBUGGER REPORT

Region   | XY Location (um)                    |Timing Library|PG View |PG Integrity|IR-Drop Composition| Root Cause Analysis
-------------------------------------------------------------------------------------------------------------------------------------
Region 1 | 129.340 61.480 169.940 66.700       |N/A           |  PASS  |  PASS      | 0.113(V)          | see the region_1.rpt file
Region 2 | 38.280 61.480 64.380 66.700         |N/A           |  PASS  |  PASS      | 0.094(V)          | see the region_2.rpt file
Region 3 | 99.180 61.480 129.340 66.700        |N/A           |  PASS  |  PASS      | 0.096(V)          | see the region_3.rpt file
Region 4 | 80.620 61.480 100.050 66.700        |N/A           |  PASS  |  FAIL      | 0.094(V)          | see the region_4.rpt file
Region 5 | 234.900 173.710 253.460 178.930     |N/A           |  PASS  |  PASS      | 0.092(V)          | see the region_5.rpt file
Region 6 | 61.190 61.480 79.170 66.700         |N/A           |  PASS  |  PASS      | 0.093(V)          | see the region_6.rpt file
Region 7 | 26.680 61.480 43.210 66.700         |N/A           |  PASS  |  PASS      | 0.094(V)          | see the region_7.rpt file
```

For each region, there are region-based reports (red-highlighted in the above diagram) that have instances with the worst IR drops and their layer-based and net-based IR drop values.

© 2013-2023 All Rights Reserved.

IR Aware Placement

The aim of IR Aware Placement analysis is to spread the high IR drop instances away from hotspots to improve IR drop. The high IR drop instances are vertically moved high as it is more efficient than horizontal movement.

This step is performed in Innovus. IR Aware placement requires the following commands and variable:

- `setPlaceMode -place_detail_irdrop_aware_effort{none | low | medium high}` - Specifies the IR drop optimization effort level to be used during `refinePlace`. The possible arguments are:
  - `none`: Disables the feature. (Default)
  - `low`: Moves the aggressor instances to nearby rows, and the aggressor has the same priority as the other instances during legalization.
  - `medium`: Moves the aggressor instances to nearby rows, and the aggressor has higher priority than the other instances during legalization.
  - `high`: Moves the aggressor instances to nearby rows, and the aggressor has higher priority than the other instances during legalization. `high` will treat the rule more strictly during placement legalization.
- `refinePlace` - Performs the placement of instances based on the reports generated by Hotspot Debugger. The `refinePlace` command performs power and rail analysis, identifies hotspots, and refines the placement of instances. Therefore, you do not have

© 2013-2023 All Rights Reserved.

to specify the `report_power`, `analyze_rail`, and `debug_irdrop` standalone commands. The use model of this command is given below.

- For Post-CTS Database: `refinePlace`
- For Post-routed Database:
  `setPlaceMode -place_detail_preserve_routing true`
  `refinePlace`

In case of the post-routed database, you need to re-route the signal nets using the `ecoRoute` command after all the optimization iterations are complete.

Tip

For information on the timing analysis and placement commands used in the Innovus software, refer to *Innovus Text Command Reference*.

The following 4 directories are generated in the current directory after `refinePlace`:

- `VoltusPowerAnalysisResults`
- `VoltusRailAnalysisResults`
- `VoltusDebugIRdropResults`
- `work`

Rechecking IR Drop and Performing Iterations

As part of the first iteration after `refinePlace`, you need to recheck the IR drop in the design. For rechecking IR Drop, specify the `refinePlace` command multiple times:

```
setPlaceMode -place_detail_preserve_routing true
refinePlace
mv VoltusDebugIRdropResults VoltusDebugIRdropResults_1
setPlaceMode -place_detail_preserve_routing true
refinePlace
mv VoltusDebugIRdropResults VoltusDebugIRdropResults_2
setPlaceMode -place_detail_preserve_routing true
refinePlace
mv VoltusDebugIRdropResults VoltusDebugIRdropResults_3
analyze_rail -output ./VoltusRailAnalysisResults -type domain ALL
```

You must run the `analyze_rail` command after the final iteration using the `refinePlace` command. That is, you need to run rail analysis after the final placement of instances.

© 2013-2023 All Rights Reserved.

Rerouting the Signal Nets and Saving the Design

Rerouting needs to be done in the post-routed database after `refinePlace` as the instances are moved to a different location, and the signal nets need to be rerouted. For rerouting, use the following command:

```
ecoRoute
```

You can do `ecoRoute` after every `refinePlace` iteration but this will increase the turnaround time. It is therefore recommended to do `ecoRoute` after all the iterations are done.

Important

If you are checking timing after optimization using `timeDesign`, it will not be accurate unless you do `ecoRoute`.

At every stage, it is recommended to save the design after `refinePlace` as there will be changes in the design that needs to be stored. For saving the design, use the following command:

```
saveDesign design_name.enc
```


© 2013-2023 All Rights Reserved.

Debugging IR Aware Placement

After the first Power and Rail Analysis, you will be able to see the hotspots in the GUI. The following image shows that some of the instances, such as clock buffer cells and flip flops, are switching simultaneously causing these highlighted hotspot regions:

You need to use Hotspot Debugger to recognize the hotspots. The following is a snapshot of the main Hotspot Debugger report:

```
HOT SPOT DEBUGGER REPORT

Region   | XY Location (um)                |Timing Library|PG View |PG Integrity|IR-Drop Composition| Root Cause Analysis
-------------------------------------------------------------------------------------------------------------------------------
Region 1 | 129.340 61.480 169.940 66.700   |N/A           |  PASS  |  PASS      | 0.113(V)          | see the region_1.rpt file
Region 2 | 38.280 61.480 64.380 66.700     |N/A           |  PASS  |  PASS      | 0.094(V)          | see the region_2.rpt file
Region 3 | 99.180 61.480 129.340 66.700    |N/A           |  PASS  |  PASS      | 0.096(V)          | see the region_3.rpt file
Region 4 | 80.620 61.480 100.050 66.700    |N/A           |  PASS  |  FAIL      | 0.094(V)          | see the region_4.rpt file
Region 5 | 234.900 173.710 253.460 178.930 |N/A           |  PASS  |  PASS      | 0.092(V)          | see the region_5.rpt file
Region 6 | 61.190 61.480 79.170 66.700     |N/A           |  PASS  |  PASS      | 0.093(V)          | see the region_6.rpt file
Region 7 | 26.680 61.480 43.210 66.700     |N/A           |  PASS  |  PASS      | 0.094(V)          | see the region_7.rpt file
```


© 2013-2023 All Rights Reserved.

Here, the design is divided into seven regions. The report states the location of these regions, timing libraries, power-grid views and power-grid integrity checks, IR drop consumption, and report for each region with root cause analysis. Based on the XY location of each region, you can plot boxes to mark the regions that are reported. For that you can use the following command:

```
add_gui_shape -rect {x1 y1 x2 y2} -layer layer_name  -width value
```

This will plot rectangles as shown in the following image:

© 2013-2023
All Rights Reserved.

Sample Run Script

```
#####################################
## IR Drop Analysis - Initial
#####################################
setMultiCpuUsage -localCpu 4
restoreDesign ../design/postRouteOpt_iraware.enc.dat/super_filter
```

Power Analysis Environment Setup

```
set_power_output_dir ./power/power1
set_power_analysis_mode -reset

set_power_analysis_mode \
-analysis_view AV_wc_on \
-disable_static false \
-write_static_currents true \
-binary_db_name dynPower.db \
-create_binary_db true \
-method dynamic_vectorless \
-enable_state_propagation true

set_default_switching_activity -reset
set_default_switching_activity -seq_activity 0.2 -input_activity 0.2 -
clock_gates_output 2
set_dynamic_power_simulation -reset
set_dynamic_power_simulation -period 160ns -resolution 20ps
```

Dynamic Rail Analysis Environment Setup

```
set_rail_analysis_mode \
-method dynamic \
-analysis_view AV_wc_on \
-accuracy hd -gif_resolution 0 \
-limit_number_of_steps false \
-power_grid_library ../data/pgv_dir/tech_pgv/techonly.cl \
-enable_rlrp_analysis true

set_power_data -reset
set_power_data -format current [ ../*dynamic*.ptiavg ]
set_pg_nets -net VDD_AO -voltage 0.9 -threshold 0.855
set_pg_nets -net VDD_column -voltage 0.9 -threshold 0.855
set_pg_nets -net VDD_ring -voltage 0.9 -threshold 0.855
set_pg_nets -net VDD_external  -voltage 0.9 -threshold 0.855
set_pg_nets -net VSS -voltage 0 -threshold 0.045
set_rail_analysis_domain -name ALL -pwrnets { VDD_AO VDD_external } -gndnets VSS
```

Define Voltage Source Location

```
set_power_pads -reset
set_power_pads -net VDD_AO -format xy -file ../power_pads/super_filter_VDD_AO.pp
set_power_pads -net VDD_external -format xy -file
../power_pads/super_filter_VDD_external.pp
set_power_pads -net VSS -format xy -file ../power_pads/super_filter_VSS.pp
```

Initial Timing Analysis

```
timeDesign -postRoute -outDir ./timing/timing_initial
```


© 2013-2023 All Rights Reserved.

#----------
Running RefinePlace - multiple iterations

#----------
Use `deleteFiller` to delete fillers if there are any fillers in the design to make space for instances to move

```
setPlaceMode -place_detail_irdrop_aware_effort high -
place_detail_preserve_routing true
refinePlace
mv VoltusDebugIRdropResults VoltusDebugIRdropResults_1
setPlaceMode -place_detail_preserve_routing true
refinePlace
mv VoltusDebugIRdropResults VoltusDebugIRdropResults_2
setPlaceMode -place_detail_preserve_routing true
refinePlace
mv VoltusDebugIRdropResults VoltusDebugIRdropResults_3
analyze_rail -output ./VoltusRailAnalysisResults -type domain ALL
# Rerouting broken signal nets
ecoRoute
```

Final Timing Analysis

```
timeDesign -postRoute -outDir timing/final
timeDesign -postRoute -outDir timing/final_hold -hold
saveDesign databases/final_database.enc
```


© 2013-2023 All Rights Reserved.

Timing-aware IR Drop Fixing

The Timing-aware IR drop fixing flow allows you to fix the IR drop violations using the Tempus ECO feature. Some of the challenges related to IR drop analysis and fixing are:

- Power-grid is consuming an increasing amount of chip area
- Libraries that are characterized at given voltages are getting unusable
- Huge time is spent in fixing of IR drops through traditional means

The manual techniques used to fix the IR drop hotspots are time consuming and iterative because finding the right manual ECO can be challenging when designers do not have the complete timing and physical information.

Tempus-ECO is a physical-aware optimization tool that can work on the sign-off timing analysis data (Graph Base Analysis or Path Base Analysis) and can provide very accurate slack information. This can make the IR drop fixing more predictive, avoiding several iterations which are otherwise required in manual techniques. In this flow, the IR drop analysis is run using Voltus Rail Analysis, which provides feedback on hotspots to Tempus-ECO to make the ECO changes and to reduce the IR drop. Rail Analysis will output a binary database containing the IR drop information, as shown below:

The IR drop hotspots will be fixed by sizing logic cells in the timing-aware mode, as shown below:

© 2013-2023
All Rights Reserved.

IR-aware ECO Flow

The following diagram illustrates the combined setup and steps to perform the Tempus IR-aware ECO flow:

Important

Voltus checks out the Tempus TSO license option when this flow is invoked with the `eco_opt_design` command.

The flow has the following steps:

1. Perform static timing analysis.
2. Set up power and rail analysis modes and environment using the following commands:

```
set_power_output_dir dir_name
set_power_analysis_mode ...
set_default_switching_activity ...
set_dynamic_power_simulation ...
set_rail_analysis_mode ...
set_pg_nets ...
set_rail_analysis_domain ...
set_power_pads ...
```


© 2013-2023 All Rights Reserved.

3. Perform power and rail analysis using the following commands:

   ```
   report_power
   analyze_rail
   ```

4. Run Hotspot Debugger to recognize hotspots using the following command:

   ```
   debug_irdrop
   ```

5. Perform Tempus ECO analysis using the following commands:

   ```
   set_eco_opt_mode
   eco_opt_design
   ```

   The ECO flow generates the `eco_innovus.tcl` file.

6. Run rail analysis again to re-check the result after Tempus ECO.

   To perform post-ECO rail analysis:

   a. Source the `eco_innovus.tcl` file in Innovus.

   b. Run rail analysis in Innovus.

   OR

   Run rail analysis in Voltus. In this case, use the DEF file from Innovus to run post-ECO rail analysis within Voltus.

© 2013-2023 All Rights Reserved.

Hotspot Debugger and ECO Analysis

Hotspot Debugger (`debug_irdrop`) identifies the hotspots with aggressors that demand high currents. These instances in the hotspot regions are reported by the `debug_irdrop` command, and later used by Tempus to perform Tempus ECO. This command divides all the hotspots in the design into the specified number of regions, and reports the worst instances in that region. The values of the `-nregion` and `-nworst_instances` parameters of the `debug_irdrop` command are key to get the best results for IR Drop improvement with lesser affect to timing.

The following two commands are used in the Tempus ECO analysis part of the flow:

- `set_eco_opt_mode` is a command to set the global parameters for ECO. You need to specify these parameters before running the `eco_opt_design` command.
- `eco_opt_design` automatically uses these global parameters to run ECO optimization.

The following ECO command parameters can be specified in this flow:

- `set_eco_opt_mode -fix_ir_drop {true | false}` - fixes IR drop violations.
- `set_eco_opt_mode -max_slack <> -max_paths <> -nworst <>` - these options are configurable based on need, and are active only when the -retime parameter is set.
- `set_eco_opt_mode -load_irdrop_db` *`dir`* - specifies the directory path where previously generated debug_irdrop files are kept. This directory must be same as specified with the debug_irdrop -output_directory parameter.
- `set_eco_opt_mode -allow_multiple_incremental {true | false}` - allows to run multiple types of optimization in the incremental mode.
- `eco_opt_design -drv` - triggers ECO.

Depending on the `set_eco_opt_mode` settings, the software will either auto-generate the ECO Timing DB, or load the existing one. Then, it will proceed to the timing fixing phase, and finally output an ECO file and detailed diagnostics in the log file.

© 2013-2023 All Rights Reserved.

Rechecking IR Drop and Result Analysis

You need to recheck the IR drop in the design after Tempus ECO to see improvement in IR drop violations. The same activity file and switching scenario that was used during the first iteration should be used in the second iteration.

There are multiple ways to analyze the quality of results before and after Tempus ECO, such as using voltage drop map, checking the number of IR drop violations, and analyzing the timing report.

Check the ivdd map to see hotspots

To view the ivdd map in the GUI:

1. Select *Power & Rail Plots* option from the *Power & Rail* menu.

   The Power & Rail Plots form appears.
2. Open the *Power & Rail Setup* form, accessed by clicking the *DB Setup* button.
3. Click the *Browse* button next to the *Rail Database* field to load the rail data.
4. Click *OK*.
5. In the Power & Rail Plots form, select the *Rail* radio button, and select the *ivdd* rail analysis plot type from the drop-down list to view the ivdd map.

© 2013-2023 All Rights Reserved.

The following two snapshots show the ivdd map before and after running Tempus ECO:

ivdd map Pre-Tempus ECO

ivdd map Post-Tempus ECO

Check the IR drop violations reported in the log file

You can open the log file to check the number of violations reported after running the `debug_irdrop` command.

```
[07/10 12:55:23     35s] Rail Analysis completed successfully.
[07/10 12:55:23     35s]
[07/10 12:55:23     35s] Finished Rail Analysis at 12:55:23 07/10/2017 (cpu=0:00:10, real=0:00:15, peak mem=0.00MB)
[07/10 12:55:23     35s] Current Voltus resource usage: (total cpu=0:00:00, real=0:00:00, mem=0.00MB)
[07/10 12:55:23     40s] <CMD> debug_irdrop -state_directory rail/ALL_125C_dynamic_1/ -domain ALL -output_directory
e-eco_latest -nworst_instances 1000000 -eco_report -nregion 8
[07/10 12:55:41     41s]
[07/10 12:55:41     41s]
[07/10 12:55:41     41s] ***Info: Found 2431 instances with IR-drop violation in design.
```

The following snippet shows IR drop violations before the Tempus ECO flow:

```
[07/10 12:57:02     74s] Rail Analysis completed successfully.
[07/10 12:57:02     74s]
[07/10 12:57:02     74s] Finished Rail Analysis at 12:57:02 07/10/2017 (cpu=0:00:13, real=0:00:15, peak mem=0.00
[07/10 12:57:02     74s] Current Voltus resource usage: (total cpu=0:00:00, real=0:00:00, mem=0.00MB)
[07/10 12:57:03     81s] <CMD> debug_irdrop -state_directory railPostECO/ALL_125C_dynamic_1/ -domain ALL -output
tory post-eco_latest -nworst_instances 1000000 -eco_report -nregion 8
[07/10 12:57:19     81s]
[07/10 12:57:19     81s]
[07/10 12:57:19     81s] ***Info: Found 2069 instances with IR-drop violation in design.
[07/10 12:57:19     81s]
```


© 2013-2023
All Rights Reserved.

From the above two snippets, it is observed that this flow has fixed a total 362 IR drop violations.

Check the timing reported in the log file

You can open the log file to check timing quality of results before and after running Tempus ECO. The following snippet shows IR drop violations before and after the Tempus ECO flow:

```
-------------------------------------
          Initial Hold Summary
-------------------------------------

+--------------------+-----------+
|     Hold mode      |    all    |
+--------------------+-----------+
|           WNS (ns):|     0.280 |
|           TNS (ns):|     0.000 |
|    Violating Paths:|         0 |
+--------------------+-----------+

-------------------------------------

-------------------------------------
         Initial Setup Summary
-------------------------------------

+--------------------+-----------+
|     Setup mode     |    all    |
+--------------------+-----------+
|           WNS (ns):|     1.991 |
|           TNS (ns):|     0.000 |
|    Violating Paths:|         0 |
+--------------------+-----------+
```

Timing QoR - Pre-Tempus ECO

```
-------------------------------------
          Final Hold Summary
-------------------------------------

+--------------------+-----------+
|     Hold mode      |    all    |
+--------------------+-----------+
|           WNS (ns):|     0.280 |
|           TNS (ns):|     0.000 |
|    Violating Paths:|         0 |
+--------------------+-----------+

-------------------------------------

-------------------------------------
         Final Setup Summary
-------------------------------------

+--------------------+-----------+
|     Setup mode     |    all    |
+--------------------+-----------+
|           WNS (ns):|     1.884 |
|           TNS (ns):|     0.000 |
|    Violating Paths:|         0 |
+--------------------+-----------+
```

Timing QoR - Post-Tempus ECO

These two snippets indicate that timing is not degraded.

Check the eco.ird file

You can check the eco.ird file before and after the Tempus ECO flow to compare the IR drop numbers of each instance and to see an impact of the Tempus ECO flow. The eco.ird file is located at <debug_irdrop_output_dir>/eco.ird.

© 2013-2023 All Rights Reserved.

Sample Run Script

```
set_multi_cpu_usage -localCpu 4 -remoteHost 4 -cpuPerRemoteHost 1
```

 # Load Design Data

```
set lefs [list]
lappend lefs ../data/lef/gsclib090_tech.lef
lappend lefs ../data/lef/gsclib090_macro.lef
lappend lefs ../data/lef/decap.lef
lappend lefs ../data/lef/pll.lef
lappend lefs ../data/lef/pso_header.lef
lappend lefs ../data/lef/pso_ring.lef
lappend lefs ../data/lef/buf_ao.lef
read_lib -lef $lefs
read_view_definition ../design/postRouteOpt.enc.dat/viewDefinition.tcl
read_verilog ../design/postRouteOpt.enc.dat/super_filter.v.gz
set_top_module super_filter -ignore_undefined_cell
read_def ../design/super_filter.def.gz

source ../data/voltus/tempus_cmd.tcl

read_spef -decoupled -rc_corner RC_wc_125 ../design/postRouteOpt_RC_wc_125.spef.gz
```

 # Set up Power Analysis Mode Options

```
set_power_output_dir power
set_power_analysis_mode \
-analysis_view AV_wc_on \
-write_static_currents false \
-binary_db_name dynPower.db \
-create_binary_db true \
-method dynamic_vectorless \
-disable_static false \
-enable_mt_libreading true \
-honor_negative_energy true \
-ignore_control_signals false \
-ignore_nets_without_timing true \
-report_missing_input true \
-report_missing_nets true \
-use_encounter_db false \
-enable_state_propagation true \
-current_generation_method peak \
```


© 2013-2023 All Rights Reserved.

```
-report_instance_switching_info output_logic \
-honor_combinational_logic_on_clock_net false

set_default_switching_activity -input_activity 0.9 -seq_activity 0.9 -
clock_gates_output_ratio 2.0
set_dynamic_power_simulation -resolution 50ps
set_power_include_file ../data/voltus/power_reg.inc

set timing_library_read_without_power false
```

#Run Power Analysis

```
report_power
write_twf pre_eco.twf
```

#Set up Rail Analysis Mode Options

```
set_rail_analysis_mode \
-eiv_eval_window switching \
-eiv_method worstavg \
-method dynamic \
-limit_number_of_steps false \
-analysis_view AV_wc_on \
-enable_rlrp_analysis true \
-verbosity true \
-cluster_via1_ports false \
-optimize_LEF_ports false \
-optimize_stdcells_library false \
-ignore_fillers false \
-ignore_decaps false \
-enable_manufacturing_effects true \
-accuracy hd \
-temperature 125 \
-gif_resolution 0 \
-power_grid_library {  ../data/pgv_dir/tech_pgv/techonly.cl
../data/pgv_dir/stdcell_pgv/stdcells.cl  ../data/pgv_dir/macro_pgv/macros_pll.cl
}
```

#Setup Voltage Source Locations

```
set_power_pads -net VDD_AO -format xy -file ../design/super_filter_VDD_AO.pp
set_power_pads -net VDD_external -format xy -file
../design/super_filter_VDD_external.pp
set_power_pads -net VSS -format xy -file ../design/super_filter_VSS.pp
```


© 2013-2023 All Rights Reserved.

#Setup PG Nets, Supply Voltage and Threshold

```
set_pg_nets -net VDD_AO -voltage 0.9 -threshold 0.85 -force
set_pg_nets -net VDD_column -voltage 0.9 -threshold 0.85 -force
set_pg_nets -net VDD_ring -voltage 0.9 -threshold 0.85 -force
set_pg_nets -net VDD_external -voltage 0.9 -threshold 0.85 -force
set_pg_nets -net VSS -voltage 0.0 -threshold 0.05 -force
```

#Set up Rail Analysis Domain and Power Data Files to Run Rail Analysis

```
set_power_data -format current {  ./power/dynamic_VDD_AO.ptipeak
./power/dynamic_VDD_column.ptipeak  ./power/dynamic_VDD_ring.ptipeak
./power/dynamic_VDD_external.ptipeak  ./power/dynamic_VSS.ptipeak  }
set_rail_analysis_domain -name ALL -pwrnets { VDD_AO VDD_external } -gndnets VSS
```

#Run Rail Analysis

```
analyze_rail -results_directory ./rail -type domain ALL
```

#HotSpot Debugger

```
debug_irdrop \
    -state_directory rail/ALL_125C_dynamic_1/ \
    -domain ALL \
    -output_directory pre-eco_latest \
    -nworst_instances 1000000 \
    -eco_report \
    -nregion 8

#######################
set_eco_opt_mode -retime path_slew_propagation
set_eco_opt_mode -max_slack 5.0 -max_paths 10000000 -nworst 50
set_eco_opt_mode -allow_multiple_incremental true
set_eco_opt_mode  -fix_max_tran false -fix_max_cap false -fix_ir_drop true
set_eco_opt_mode -load_irdrop_db pre-eco_latest
eco_opt_design -drv

write_twf post_eco.twf
##########################################
### Post-ECO
##########################################
set_power_output_dir powerPostECO
read_activity_file -format switchsrc ./power/voltus_power.statprop.switchsrc
report_power
```


© 2013-2023 All Rights Reserved.

```
set_power_data -format current {  ./powerPostECO/dynamic_VDD_AO.ptipeak
./powerPostECO/dynamic_VDD_column.ptipeak
./powerPostECO/dynamic_VDD_ring.ptipeak
./powerPostECO/dynamic_VDD_external.ptipeak  ./powerPostECO/dynamic_VSS.ptipeak  }

analyze_rail -results_directory ./railPostECO -type domain ALL

debug_irdrop \
    -state_directory railPostECO/ALL_125C_dynamic_1/ \
    -domain ALL \
    -output_directory post-eco_latest \
    -nworst_instances 1000000 \
    -eco_report \
    -nregion 8

save_design -def super_filter_voltus_tempus_eco -overwrite
```


© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

22

Signal ElectroMigration

- Analyze AC Signal ElectroMigration on page 540
  - Overview on page 540
  - Calculating Irms Waveform on page 540
  - Calculating Ipeak Waveform on page 541
  - Calculating Iavg Waveform on page 543
  - Calculating Effective Frequency on page 544
  - Computing Irms/Ipeak/Iavg for each Routing Segment on page 545
  - Checking the AC Current Limits on page 547
  - Prevention of AC Signal EM Violations on page 552
  - Repair of AC Signal EM Violations on page 552
  - Example Script for AC Signal EM Analysis on page 553
  - Top Scope Signal EM Analysis on page 555
  - Viewing Signal EM Analysis Results on page 558
  - Signal EM Report Format on page 564
- Analyze DC Signal ElectroMigration on page 566
  - Overview on page 566
  - DC Signal EM Analysis using maxCap and maxTran Limits on page 566
  - Frequency based Cell EM Limits in Liberty on page 567
  - Repair of DC Signal EM Violations on page 568
  - Example Script for DC Signal EM Analysis on page 568

Analyze AC Signal ElectroMigration

Overview

The charged particles of a conductor in an electric field give up kinetic energy when they collide with atomic ions. The increase in this kinetic energy of the ions manifests itself as heat and a rise in temperature. Wire self-heating or Joule Heating describes a phenomenon where high AC currents flowing through the resistive interconnects causes extreme temperature. The following diagram illustrates signal net experiencing AC current as load capacitance is charged and discharged:

In order to prevent wire self-heating or AC signal electromigration, signal interconnects should be analyzed for their AC current carrying capacity and measured against the AC current limits specified by foundry.

Earlier, the AC current density limits could only be defined in the Technology LEF file, and only root-mean-square (RMS) current limit check was supported. Now, the software supports RMS current limit, peak AC current, and average AC current limit checks. However, for peak AC current and average AC current limits, these limits must be defined in the QRC Technology file. For RMS current limits, you can define them in both the Technology LEF and QRC Technology files. To achieve more accurate results, it is recommended to use the ECSM timing library.

Calculating I_{rms} Waveform

The default Advanced Analysis Engine (AAE) delay calculator uses the internal integration method to calculate I_{rms}.

The following diagram shows the I(t) waveform for a given net:

The equation is:

$$\text{Irms} = \sqrt{\frac{S}{T_{sw}} \int_0^{T_{sw}} I^2(t)dt}$$

Where, S = Switching Factor

T_{sw} = Period of one switching cycle (rising + falling)

$S/T_{sw} = F_{eff}$, which means the effective frequency.

Calculating I_{peak} Waveform

The software estimates the peak AC current waveform from the default AAE delay calculator. The peak current is checked against the peak limit (`I_peak_ac_limit`) provided by the foundry as below:

```
Ipeak_ac_limit = Ipeak_dc_limit/sqrt(r)
```

To check peak EM violation, the tool compares "`Ipeak_measured`" against "`Ipeak_ac_limit`". An equivalent alternative is to compare "`Ipeak_measured * sqrt(r)`" against "`Ipeak_ac_limit * sqrt(r) = Ipeak_dc_limit`". By default, the

tool uses the alternative and reports "`Ipeak_measured * sqrt(r)`" as peak current, and "`Ipeak_ac_limit * sqrt(r)`" as peak limit.

ICT EM Model (Old Format Example):

```
em_jmax_ac_peak EQU 123.12*(w-0.002)
```

In this old ICT EM file format, "`123.12*(w-0.002)`" is in fact `Ipeak_dc_limit`.

ICT EM Model (New Format Example):

```
em_jmax_ac_peak EQU 123.12*(w-0.002)/( r ) ^ 0.5
```

In this new ICT EM file format, "`123.12*(w-0.002)/( r ) ^ 0.5`" is `Ipeak_ac_limit`, and "`em_peak_with_duty_ratio explicit`" is present in the process header.

The following diagram shows the I(t) waveform for a given net:

T_d calculation equation is given below:

$$\mathrm{Td} = \frac{\int_0^{T_{sw}} |I(t)|dt}{\mathrm{I_{peak_measured}}}$$

Where,

`Tsw` = Period of one switching cycle (rising + falling)

$I_{peak_measured}$ = `abs.max(Ipeak(rising), Ipeak(falling))`

According to the DRM from foundry, only signal nets with effective frequency equal or larger than the minimum effective frequency (1MHz by default) will be checked for their peak AC current. If the signal nets have an effective frequency lower than the minimum frequency, the software will issue a warning message stating that the net will be ignored for peak current check. You can use the `verify_AC_limit -minPeakFreq` parameter to change the minimum effective frequency. If the duty ratio `r` for a signal net is equal or lower than the minimum duty ratio (0.05 by default), then the minimum duty ratio will be used. You can use the `verify_AC_limit -minPeakDutyRatio` parameter to change the default value.

Calculating I_{avg} Waveform

The software uses the default AAE delay calculator to calculate the average current considering the recovery factor. For a rising (positive) and falling (negative) waveform, I_{avg} is calculated using the following equation:

I_{avg} = abs. max (rising, falling) - EM Recovery factor * abs. min (rising, falling)

where, the EM recovery factor is defined in the QRC Technology file. You can overwrite this value. The default value of the EM recovery factor is 1, that is, I_{avg} will be zero. If you do not specify the EM recovery factor, the software will issue a warning message when avg analysis is selected and no EM recovery factor is defined.

For digital CMOS circuits that are charging and discharging a fixed load, by definition, abs.Iavg(rising) = abs.Iavg(falling).

$$I_{avg(falling)} = \frac{S}{T_{sw}} \int_0^{T_{sw}} I_{(falling)}(t)dt$$

$$I_{avg(rising)} = \frac{S}{T_{sw}} \int_0^{T_{sw}} I_{(rising)}(t)dt$$

where, S = Switching Factor

T_{sw} = Period of one switching cycle (rising + falling)

S/T_{sw} = Feff, which means the effective frequency

Note: To improve the current waveform accuracy when calculating I_{avg} especially at the advanced nodes, such as 40nm and below, the Miller-Effect is taken into account in $I_{rising(t)}$ and $I_{falling(t)}$.

Example:

In this example

- $I_{avg(rising)}$ = 2mA
- $I_{avg(falling)}$ = -2mA
- em_recovery_factor = 0.5

Based on the equation, I_{avg} calculation is as follows:

I_{avg} = 2 - 0.5*2 = 1mA

Calculating Effective Frequency

For clock nets, T_{sw} is already given in the timing constraints, and S, the switching factor, is always 2.0, so the effective frequency is, Feff = $1/T_{sw}$.

For signal nets, there are two choices:

- You can specify the switching factor (S) for the signal nets using the verify_AC_limit -toggle parameter. The specified switching factor will be applied to all signal nets and multiplied with the switching frequency (1/Tsw) to calculate the effective frequency of the signal net. The default switching factor of 1.0 (100%) signifies that the signal net will switch once at every period of the related clock cycle (rising + falling). The switching factor of the clock net will not change with the verify_AC_limit -toggle parameter as it always equals to *2.0*. The switching frequency ($1/T_{sw}$) for the net is

derived from the frequency of the associated clock (fastest clock will be used if more than one is associated) using the integrated static timing analysis engine when running this software.

default switching factor of 100% on data network

```
verify_AC_limit \
-method rms \
-detailed \
-toggle 1.0 \
-report AC.1.0.rpt
```

- If using the global activity on all nets is too pessimistic or optimistic, you can set the effective frequency (`frequency * activity`) using the TCF, VCD, or FSDB file generated by netlist simulators. Optionally, you can set input and flop activities and propagate them to generate a TCF file as follows:

```
set_default_switching_activity \
-input_activity 0.2 \
-seq_activity 0.15 \
-clock_gates_output 2.0 \
-period 7.0

propagate_activity

write_tcf <design>.tcf

read_activity_file \
-format TCF \
-set_net_freq true dma_mac.tcf
verify_AC_limit \
-method avg \
-detailed \
-avgRecovery 0.5 \
-useQRCTech \
-use_db_freq \
-report AVG.tcf.rpt
```

Computing Irms/Ipeak/Iavg for each Routing Segment

For a routed net, the wire widths can taper as they reach the sink pins, and the routing can branch to multiple sinks. In order to avoid false violations, a local $I_{rms}/I_{peak}/I_{avg}$ value must be computed for each wire segment of the net.

The $I_{rms}/I_{peak}/I_{avg}$ value of each wire segment should only be the current required to charge the capacitance of the "downstream" wire segments and sinks. The following diagram illustrates $I_{rms}/I_{peak}/I_{avg}$ calculation on "downstream" wire segments:

In this diagram, a) shows that the wire segment s1 must carry current to charge/discharge the entire net, so the $I_{rms}/I_{peak}/I_{avg}$ at the driver output is the correct value to check for s1. In b), the wire segment s2 only carries current to charge/discharge s2, s4, s5 and the pin-caps of i2 and i3. In c), wire segment s5 only carries current for s5 and the pin-cap of i3.

The total wire capacitance (Cwire) comes from extraction (SPEF), and the pin-capacitance for each instance pin from the timing libraries. The total $I_{rms}/I_{peak}/I_{avg}$ will be calculated using the total wire capacitance (Cwire). The software estimates the "downstream capacitance" that must be charged/discharged through individual wire segments by using the total area of the "downstream wire segments" relative to the total area of the wire plus the pin-caps of the sinks.

In example b) for I_{rms}, the result is:

`Cwire(total net)` is already known from extraction

`Cnet(total net) = Cwire(total net) + Cpin` for all the sinks (driver's Cpin is not included)

I_{rms}(total net) is computed from `Cnet`, the slew rate and frequency is as described earlier

```
Area(total net) = Area (s1 + s2 + s3 + s4 + s5)
Cwire(downstream of s2) = Cwire(total net) * [Area(s2 + s4 + s5) / Area(total net)]
Cnet(downstream of s2) = Cwire(downstream of s2) + Cpin(i2) + Cpin(i3)
```

I_{rms} is proportional to `Cnet`, so we can estimate `Irms(s2)` as:

```
Irms(s2) = Irms(total net) * Cnet(downstream of s2) / Cnet(total net)
```

Checking the AC Current Limits

You can verify AC current violations on signal nets by using the `verify_AC_limit` command. `verify_AC_limit` checks for the following types of AC current violations on signal nets:

- Root-mean-square (RMS) current limit violations (I_{rms} violations)
- Peak current limit violations (I_{peak} violations)

In addition, `verify_AC_limit` can be used for average current limit (Iavg) analysis.

AC current limit violations are sometimes also referred as wire self-heat violations. Design Rule Check (DRC) manuals have these current limits to avoid over-heating a signal-net wire with too much AC current. To prevent wire self-heating or AC signal electromigration, signal interconnects should be analyzed for their AC current carrying capacity and measured against the AC current limits specified by foundry.

Use the `verify_AC_limit -method` parameter to specify the waveform calculation method to be used (rms, peak, or avg).

Note: SignalStorm can still be used for I_{rms} calculations, but it does not support I_{peak} or I_{avg} calculations.

RMS/Peak/Avg Current Limit Violations

By default, `verify_AC_limit` only checks for I_{rms} violations. The software calculates the I_{rms} at the driver output and compares it to the `ACCURRENTDENSITY` tables in the LEF file, which contains the I_{rms} limits for each routing layer.

If the computed Irms for a net exceeds the `ACCURRENTDENSITY` limit defined in LEF for any of the layers/widths used by the net, an error occurs and a violation marker is attached to the net. Each routing segment is checked separately. If more than one driver exists, the check starts at an arbitrary driver. The layers and widths of pins are not checked for AC current limits.

Note: You can use `verify_AC_limit -useQRCTech` to force I_{rms} checks to use the EM rules defined in the QRC technology file instead of the technology LEF file. If either I_{peak} or I_{avg} current limit checks are also turned on, then all checks, including I_{rms}, will use the EM rules defined in the QRC technology file.

The software support checks the RMS table for all layers. To get access to the width-dependent syntax, we must add an arbitrary single frequency value. Therefore, a typical table might look like:

```
LAYER met1
```

```
    ...
    #RMS AC current limits for met1, at 100 C, allowing max temp change of 20 C
    ACCURRENTDENSITY RMS
    FREQUENCY 1 ;            #syntax needs one frequency value
    WIDTH
    0.15 0.3 0.6 1.2 15 ;         #values in microns from min to max width
    TABLEENTRIES
    10.0 9.2 8.8 8.7 8.6 ;         #mA/um for each width
END met1 ;
```

The tables are interpreted as piece-wise-linear tables indexed by width and frequency. In the example above, a wire of width 0.15um can carry `10.0 mA/um * .15um = 1.5mA` of current.

The I_{rms} limits in the table can also be scaled for other factors like temperature. For example, the I_{rms} limits table should be computed for a specific "maximum allowed temperature change" (`maxTchange`). The I_{rms} limits are normally proportional to the square-root of the `maxTchange`. Therefore, if the table used a `maxTchange` of 20 degrees, and you want a `maxTchange` of 10 degrees, you would use a scale value of `sqrt(10/20) = 0.71`.

The software checks the `ACCURRENTDENSITY` tables for the following conditions:

- If more than one frequency is given, `verify_AC_limit` will warn that it is only using width values from the first frequency in the table.
- There must be at least two width values, otherwise `verify_AC_limit` gives an error message and quits.
- The widths must be increasing in value, and the current limits must be increasing in value, otherwise the software gives an error message and quits.
- If no table exists for a routing layer, a warning message is given, and the limit is assumed to be infinite for that layer.

`verify_AC_limit` also checks the `ACCURRENTDENSITY` tables for the following conditions and takes the following actions:

- If there is no table for a routing layer, the software gives a warning and assumes an infinite limit for the layer.
- If `PEAK` and `AVERAGE` tables are present, the software ignores them.

For `Irms`, signal EM analysis not only supports the rules defined in the technology LEF file, but also the rules defined in the QRC Technology file. But for `Ipeak` and `Iavg`, only the rules defined in the QRC Technology file are supported. If no EM rule is defined in the QRC Technology file, the software will stop checking and give an error message.

These rules defined in the QRC Technology file are represented in the form of an equation, as opposed to PWL, which when processed for each wire segment individually results in a more accurate EM limit calculation.

To perform this analysis, you must load the QRC tech file in Voltus using the `-qx_tech_file` parameter of `create_rc_corner` or `update_rc_corner`. `verify_AC_limit` processes the limits defined as polynomials inside the QRC tech file to determine layer-based AC current density limits. The QRC technology file attached to the current timing view (current RC corner) is used.

The QRC tech file is generated from an ASCII input file called the ICT file. For each layer, the ICT file includes `em_model` construct that defines AC and DC current density polynomials. Here's a sample `em_model` construct for a metal layer with current limits as an equation (`EQU`):

```
em_model {
em_jmax_dc_avg EQU 1 * 1.594 * ( w - 0.003 ) jmax_factor 105 1.434 110 1.000 115
0.704 120 0.500 125 0.358 L > 5 W < 0.21
em_jmax_dc_avg EQU 2 * 1.594 * ( w - 0.003 ) jmax_factor 105 1.434 110 1.000 115
0.704 120 0.500 125 0.358 L > 5 W >= 0.21
em_jmax_dc_avg EQU 4 * 1.594 * ( w - 0.003 ) jmax_factor 105 1.434 110 1.000 115
0.704 120 0.500 125 0.358 L <= 5


em_jmax_ac_peak EQU 18.04 * ( w - 0.003 )
em_jmax_ac_rms EQU sqrt( 12.66 * deltaT * ( w - 0.003 )^2 * ( w - 0.003 + 0.264 )
/ ( w - 0.003 + 0.0443 ) )
em_jmax_ac_avg EQU 2.17 (w - 0.003)
em_recover 0.5
}
```

The peak, RMS, and average current limits for this layer are computed using the equations for the following three `em_jmax_ac` models.

- `em_jmax_ac_peak`: Peak current limits for this layer.
- `em_jmax_ac_rms`: RMS current limits for this layer.
- `em_jmax_ac_avg`: Average current limits for this layer.

Important

Signal AC EM limit depends on transition time (`Td` - `time between half peaks in current waveforms`) and duty ratio (`r`) in advanced process nodes. Additionally, foundries require to apply a specific duty ratio value regardless of the calculated duty ratio.
To support this, the following AC peak EM rule should be used:

```
em_jmax_ac_peak EQU 58.10 * ( w - 0.02 ) w >= 0.1 Td >= 0.5
apply r = 1
em_jmax_ac_peak EQU 58.10 * ( w - 0.02 ) w >= 0.1 Td >= 0.001
Td < 0.5
```

In conditional statements, always use the qualifier before numerical values. For example, use `Td >= 0.5`, instead of `0.5 <= Td`.

The syntax of `EQU` for these three `em_jmax_ac` equations is a numeric expression ending in a newline. The expression follows normal expression syntax with normal precedence rules:

1. exponentiation (^)
2. * or /
3. + or -

The following reserved names (case-insensitive) can be used in the equations to pass in parameters.

- `w`: Width of the wire being checked (required for metal layers that should be checked)
- `deltaT`: Delta temperature allowed. Normally only used for RMS limits to specify the max change in temperature allowed (optional).
- `jmax_factor`: For temperature based derating (optional).
- `jmax_lifetime`: Maximum years/hours of operation for lifetime derating (optional)

The following built-in functions are supported:

- ^ (exponentiation)
- * / (multiply, divide)
- + - (addition, subtraction)
- ( ) (grouping of expressions for precedence)
- sqrt
- log

Specification of derating for lifetime or hours of operation in ICT:

`jmax_lifetime` *`lifetime1scale1 [lifetime2scale2.....]`*

`jmax_lifetime` provides the ability to set the scaling factor that applies to the current density limits for different lifetimes. Based on the lifetime value that you specify (using the

`em_lifetime_units` parameter defined in the `process` section of the ICT file), the appropriate multiplication factor is applied to the nominal current density limit. The unit should be specified in accordance with the setting defined by the `em_lifetime_units` parameter defined in the `process` definition of the ICT file.

Sample extract from the ICT file:

```
process "name" {
[em_lifetime_units hours | years]
}
conductor "M1" {
em_model {
em_jmax_ac_avg EQU 1 * 1.594 * ( w - 0.003 ) jmax_lifetime 5 4.3 10 1 15 0.4
}
}
```

Specification for temperature based derating:

`jmax_factor` *`temp1scale1`* `[`*`temp2scale2 ...`*`]`

`jmax_factor` is an optional scaling factor that can be used at different temperatures compared to the reference temperature (defined by `em_tref` in the `process` definition). The temperature for `jmax_factor` should be specified in degrees Celsius. Scaling factor is a positive integer: `>1` to scale up, `<1` to scale down, or `1` for no scale effect.

Note: When setting scale factors for multiple temperatures, they should be specified in ascending sequence to enable interpolation.

Sample extract from the ICT file:

```
process "name" {
[em_tref value_in_degrees_Celsius]
}
conductor "M1" {
em_model {
em_jmax_ac_avg EQU 1 * 1.594 * ( w - 0.003 ) jmax_factor 105 1.434 110 1.000 115
0.704 120 0.500 125 0.358
}
}
```

Area dependent parameters for via layers:

Area-based EM limits for vias can be defined using a PWL pair of area value. The `PWL` keyword in the syntax below signifies the use of area based EM limits.

```
em_jmax_ac_peak [ value | PWL value1area1 | EQU fn(E) ]
```

Sample ICT file for a via layer:

```
via "via1" {
em_model {
em_vcwidth 0.32400
em_jmax_ac_avg PWL 12.308 0.104976
}
}
```

Units for EM limits:

You can control the EM units in the ICT file using the following parameters:

```
process "name" {

# EM_Model parameters

[em_tref value]
[em_output_wlt silicon | drawn]
[em_variables variable1 [variable2 [ ...]]]
[em_conductor_unit A/cm^2 | mA/um ]
[em_via_unit A | mA ]
[em_via_area_unit A/cm^2 | mA/um^2 ]
[em_lifetime_units hours | years]
}
```

The default units are highlighted in bold.

For more information on the ICT file, see the *EM_Model* section in the "*Creating the ICT File*" chapter of the *Quantus Techgen Reference Manual*.

For an MMMC design, if you want to choose one view to perform $I_{rms}/I_{peak}/I_{avg}$ check, you need to use the `set_default_view` command to specify the view name or else the first defined view will be used by default.

Prevention of AC Signal EM Violations

Clock nets are generally the first nets to fail the AC current density limits because they have the highest frequency and net loads. It is impractical to "repair" them later without affecting the clock-skew, so clock tree synthesis should avoid creating AC current limit violation. Therefore, CTS inside Innovus has an optional max_cap value to the CTS specification file. CTS will avoid creating any clock net that exceeds the max_cap limit and therefore avoid any possible AC current limit violation.

Repair of AC Signal EM Violations

In Innovus, a `fixACLimitViolation` command exists that will repair any violations found by the `verify_AC_limit` command. It will do this initially by adding a buffer to split up the

net load to any net marked with a violation by `verify_AC_limit`. Note that most users avoid fixing violations on clock network as it will impact clock skew.

Note: The `fixACLimitViolation` command is not available in Voltus.

Example Script for AC Signal EM Analysis

read design

```
read_lib -lef <lefs>
read_lib <timing libs>
read_verilog <verilog netlist>
set_top_module <design name>
read_def <def netlist>
```

create analysis view

```
create_library_set -name <library set name> -timing <timing libs>
create_rc_corner -name <rc corner name> -qx_tech_file <qrc Tech>
create_delay_corner -name <delay corner name> -rc_corner <rc corner name>
create_constraint_mode -name <constraint name> -sdc_files <timing constraints>
create_analysis_view -name <view name> -constraint_mode <constraint name> \
-delay_corner <delay corner name>
set_analysis_view -setup <view name> -hold <view name>
read_spef -rc_corner <rc corner name> <signal parasitics>
set_default_view -hold <view name>
```

Note: If you use the Quantus-generated extended SPEF file format in the signal EM flow with `-effort_level medium` or `high`, you must specify the `-extended` and `-starN` parameters with the `read_spef` command.

Switching factor of 100% on data network

```
verify_AC_limit \
-detailed \
-toggle 1.0 \
-report AC.1.0.rpt
```

Switching factor of 50% on data network

```
verify_AC_limit \
-detailed \
-toggle 0.5 \
-report AC.0.5.rpt
```

set input activity and propagate it to get realistic effective frequency

```
set_default_switching_activity \
-input_activity 0.2 \
```

```
-seq_activity 0.15 \
-clock_gates_output 2.0 \
-period 7.0

propagate_activity

write_tcf dma_mac.tcf

read_activity_file \
-format TCF \
-set_net_freq true dma_mac.tcf


verify_AC_limit \
-method {rms avg peak}
-detailed \
-use_db_freq \
-useQRCTech \
-avgRecovery 0.5 \
-report AC.tcf.rpt
```

Top Scope Signal EM Analysis

During iterative design refinement, Electromigration (EM) analysis is performed on the modules and the top-level logic for a large hierarchical design. For top-level analysis, use the `create_top_scope` command to perform abstraction of the top-level hierarchical instances for analysis of only the top-level logic instead of running the EM analysis of the fullchip netlist. The command allows you to perform abstraction of all the hierarchical instances in the top-level, or specify the top-level hierarchical instances that should be abstracted. When a hierarchical instance is abstracted, all the internal logic is ignored while retaining the interface paths.

The following diagram illustrates the top scope signal EM flow:

The `create_top_scope` command must be specified before the `read_sdc` command. The following is an example script for the top scope signal EM flow:

Import Design

```
read_lib ..
read_verilog …
set_top_module …
```

Create Top Scope

```
create_top_scope
read_sdc...
read_def...
read_spef...
set_dc_sources...
```

Set the process node before EM analysis

```
set_design_mode...
verify_AC_limit...
```

When the `create_top_scope` command is specified, Voltus generates current waveforms for the top-level nets, and nets interfacing with the hierarchical blocks. All the other nets are ignored from the current generation process. The clock network is traced for the design to calculate accurate slews for the nets being driven from inside the block.

Boundary Model Based Signal EM Analysis

The boundary model for a block provides sufficient details of its interface paths to accurately analyze these paths when loading the model into the signal EM analysis of the top-level module. These details include interface netlist, SPEF, and side inputs. Aggressors that are not on the boundary paths are not included in the model, and any coupling capacitance to such nets is grounded. To create the cell boundary model, use the `create_cell_signal_em_model` command. After the boundary model is generated, it is automatically loaded to the higher-level module.

The following figure shows the top scope with the block boundary model:

Here,

- Red nets indicate signal EM analysis is performed
- Black nets indicate skipped from signal EM analysis

- **Yellow logic** is removed during boundary model generation
- Full clock propagation such that slew and clock marking are done correctly

In the top-level signal EM run, the boundary model is loaded instead of the full block contents. Boundary models can be loaded for multiple blocks. If you specify a Verilog for a block that already has a boundary model, the specified Verilog will override the netlist in the boundary model. The boundary model should not be overridden with a Verilog definition.

You can specify the `create_cell_signal_em_model` command before or after running the `set_top_module` command. The details of the two use models are given below:

- If the `create_cell_signal_em_model` command is specified before `set_top_module`, the command will generate the boundary model for a sub-cell and load it. For this use model, the `-cell`, `-dir`, `-spef`, and `-verilog` parameters must be specified.
- If the `create_cell_signal_em_model` command is specified after `set_top_module`, the command will generate the model for the top cell for use in a subsequent top-level run. In this case, the command is issued in a block-level run, and the SPEF should be loaded prior to the `create_cell_signal_em_model` command. For this use model, the `-dir` parameter must be specified. Since the design is already loaded, the relevant information is obtained from the current session.

The following is an example script for the boundary model based top scope signal EM flow:

Import Design

```
read_lib -lef ...
read_lib ..
```

Read top-level Verilog

```
read_verilog …
```

BM generation for blocks

```
create_cell_signal_em_model -cell <cell> ...
```

The following is the command for creating a boundary model and loading it into the current session:

```
create_cell_signal_em_model -dir bm_dir  -cell cella  -verilog cella.v  -spef
cella.spef.gz  -overwrite -tech_lef tech.lef  -tech_file 65lp_typical.tch -
tech_file_map  leflayermap
```

The following is the command for creating a boundary model after running `set_top_module` in a block-level session:

```
create_cell_signal_em_model -dir bm_dir -overwrite
```

Create Top Scope

```
create_top_scope -include_side_loads -summary -thread_count 8 -cell_list
$cell_list
```

Boundary model Verilog files to be read for blocks “automatically”

```
set_top_module <top>
read_def -skip_pg -keepPinGeometry -
top_scope_ignore_block_internal_nets_on_boundary_path
```

Boundary model SPEF files to be read for blocks “automatically”

```
read_spef -extended -starN $spef_list
read_sdc $sdc_list
```

Set the process node before EM analysis

```
set_design_mode...
verify_AC_limit...
```

Viewing Signal EM Analysis Results

The signal EM results can be debugged in the context of physical layout using the interactive signal EM analysis plots. To view the signal EM plots:

1. Select Power & Rail - Power & Rail Plots.

The Power & Rail Plots form appears. This form is the main window for displaying the signal EM analysis results.

Note: To enable the signal EM GUI display, you must specify the `verify_AC_limit -report_db` command parameter. This parameter writes the signal EM data into a database file (`filename.db`) after performing signal EM analysis. This file contains the information required for loading and displaying signal EM data in GUI.

2. Click the *DB Setup* button on the Power & Rail Plots form.

The Power & Rail Setup form appears.

3. Click the *Browse* button next to the *SEM Database* field to load the signal EM database file. You need to load the database file generated by the `verify_AC_limit -report_db` command parameter.
4. Click *OK*.
5. To set the visibility of the signal EM layers, click the *Layers/Nets* button.

   The Power & Rail Layers/Nets form appears.

6. In the *SemLayers* tab, select or clear the check boxes adjacent to the signal EM layers to modify the display of layers.
7. Click *OK*.

8. Select the *Sem* radio button and select a signal EM analysis plot type from the drop-down list.

The *sem_rms*, *sem_avg*, and *sem_peak* plots are enabled based on the `verify_AC_limit -method` command parameter. If you specify "`-method {rms peak avg}`", then all the three plots are enabled. If you specify "`-method {rms avg}`", then only I_{rms} and I_{avg} can be plotted in the GUI.

The following image displays the *sem_avg* plot:

The color of each wire segment represents its average current density that was determined after running the `verify_AC_limit` command.

9. Similarly, select the other types of signal EM plots and analyze the signal EM analysis results.

10. You can query signal wire segment related data, such as net information or current density value, by double-clicking a wire segment.

    The Attribute Viewer form appears:

    The Attribute Viewer form can also be displayed by selecting a wire segment and pressing `q`.

11. Click *Close*.

Example Script for Displaying the Signal EM Plot in GUI

read design

```
read_lib -lef <lefs>
read_lib <timing libs>
read_verilog <verilog netlist>
set_top_module <design name>
read_def <def netlist>
```

write the signal EM data into a database file. The `-method {rms peak avg}` parameter enables the *sem_rms*, *sem_avg*, and *sem_peak* plots in the *Power & Rail Plots* form

```
verify_AC_limit -method {rms peak avg} -detailed -report_db -report dma_design.rpt
```

load the signal EM data. When loading the signal EM data, the rail directory (`-rail_directory`) must be specified with the signal EM database (`-sem_db`).

```
read_power_rail_results -rail_directory dynamic_rail/core_25C_dynamic_1 -sem_db
dma_design.db -nets { VSS VDD} -instance_voltage_method { worst avg } -
instance_voltage_window {timing}
```

display the RMS current density plot

```
set_power_rail_display -plot sem_rms
```

Signal EM Report Format

Sample filename: signal.rpt

Format:

```
Irms/Limit   Irms   Limit   Ipeak/Limit   Ipeak   Limit   Iavg/Limit   Iavg   Lim
it   Width/CutArea   FixWidth/FixCutNum   Cap   Layer   (X1 Y1 X2
Y2)   PathLength   BlechLength   Current_Direction   RMS_V2VFactor
```

Format Description:

`Irms/Limit` specifies the RMS current value that flows through the metal/via resistor that is calculated by the tool divided by the current limit value provided by the foundry.

`Irms` specifies the actual RMS current value that flows through the metal/via resistor that is calculated by the tool.

`Limit(mA)` specifies the current limit value that is provided by the foundry depending upon its width and thickness.

`Ipeak/Limit` specifies the peak current value that flows through the metal/via resistor that is calculated by the tool divided by the current limit value provided by the foundry.

`Ipeak` specifies the actual peak current value that flows through the metal/via resistor that is calculated by the tool.

`Limit(mA)` specifies the current limit value that is provided by the foundry depending upon its width and thickness.

`Iavg/Limit` specifies the average current value that flows through the metal/via resistor that is calculated by the tool divided by the current limit value provided by the foundry.

`Iavg` specifies the actual average current value that flows through the metal/via resistor that is calculated by the tool.

`Limit(mA)` specifies the current limit value that is provided by the foundry depending upon its width and thickness.

`Width/CutArea` If the wire segment is metal, it is calculated as the actual width perpendicular to the current flow of the wire segment. If the wire segment is via, it is calculated as the area of the cut in the via.

`FixWidth/FixCutNum` is the width (for metal segment) or cutNum (for via) required to fix the violation. When there is a violation, it is the new width of the metal wire or the new number of cuts of via (drawn area).

`Cap` is the capacitance of the wire segment.

`Layer` specifies the metal/via layer name on which the resistor is located.

`(X1 Y1 X2 Y2)` specifies the coordinates of the resistor.

`PathLength` of a wire segment is the distance traced from the first pin shape of driver to the center of the wire segment.

`Blechlength` is the longest and continuous centerline path from edge to edge among the connected wire shapes on the same metal layer.

`Current Direction` is applicable only to via layers where 'D" denotes down, "U" denotes up.

`RMS_V2VFactor` is the EM benefit from the via-to-via spacing rule for metal layers.

Analyze DC Signal ElectroMigration

Overview

Hot-carrier injection or DC signal electromigration can occur whenever a transistor switches. Whenever there are Ids (drain-source current), there is some probability that the carriers get enough kinetic energy to be injected into the gate oxide and cause a Vth shift. The calculation of this probability is a complex function of the Vgs (gate-source voltage) and the Vds (drain-source voltage) and Ids.

The most common problem occurs when there is a slow output transition with a large output capacitance which would mean a high Vds for a long time with a high Vgs and Ids. This type of operation causes a higher probability of hot-carrier injection, and therefore, Vth shifts and could result in reliability problems. The following diagram illustrates that within a cell, some segments only experience DC current:

The current discharged through wire A and B is a pulsed DC current that is proportional to the frequency of the signal net and the load capacitance:

```
Idc = frequency x Cout
```

Since the wire inside the cell has a fixed width, it also has a fixed maximum DC current electromigration limit. A high-frequency output needs a smaller Cout to keep Idc under the DC electromigration limit, while a low-frequency output can tolerate a higher Cout.

The most useful way to express this limit is as a maxCap versus frequency limit for the cell.

DC Signal EM Analysis using maxCap and maxTran Limits

The software manages DC electromigration using maximum capacitance limit based on switching frequency. This is driven by `maxCapPerFreq` data in liberty. In addition, it also

supports `maxCapPerTran` and `maxCapPerFreqTran`. The violations can be reported using the `report_design_rule -cap` and `report_design_rule -tran` commands.

Optionally these constraints can be set for cell pins manually, as follows:

```
set_max_cap_per_freq \
-force \
-freq {0 100e6 200e6} \
-cap {2.0 2.0 1.0} \
-pins {*_16X *}


set_max_tran_per_freq \
-force \
-lib tcbn65lplvtwc \
-freq {50e6 70e6 100e6 200e6} \
-tran {5.0 4.0 3.0 1.5} \
-pins {* *}
```

Frequency based Cell EM Limits in Liberty

The electromigration "group" in `liberty` supports 1-D and 2-D look-up tables that lists frequency based electromigration limits based on input transition time and output loading capacitance.

- Input Pin: 1-D table listing frequency limits based on input transition time
- Output Pin: 2-D table listing frequency limits based on input transition time and output load capacitance

The following figure displays cell-based DC electromigration limit templates in `liberty`:

Important

You can also use Liberate to generate frequency-based cell electromigration limits in Liberty. Liberate is an accurate, highly efficient and easy-to-use library characterizer that creates electrical views (timing, power and signal integrity) in industry standard formats such as Liberty (.lib) format. For more information on generating Liberty EM models using Liberate, refer to the "*Electromigration Models*" chapter in the Liberate Reference Manual & User Guide.

The `report_freq_violation` command invokes static timing analysis and computes effective frequency (switching factor * clock domain freq) on each instance pin. The switching factor is determined by either propagating the activities or by specifying a TCF, VCD, or FSDB file. The clock domain frequency for the instance pin is derived from static timing analysis. The software will then compute frequency limit at the output pin from `liberty` based on slew on the related input pin and output load to determine frequency violation.

Repair of DC Signal EM Violations

In Innovus, the `optDesign` command can be used to upsize cell instances and minimize load capacitance to fix DC signal EM violations. This command is not available inside Voltus. The auto-fixing is not available for violations reported by the `report_freq_violation` command at this time.

Example Script for DC Signal EM Analysis

read design

```
read_lib -lef <lefs>
read_lib -max <timing libs>
read_lib -min <timing libs>
read_verilog <verilog netlist>
set_top_module <design name>
read_sdc <timing constraints>
read_def <def netlist>
read_spef <signal parasitics>
```

check max transition violations

```
report_design_rule \
-tran \
-max \
-outfile max.tran.rpt
```

check max capacitance violations

```
report_design_rule \
-cap \
-max \
-outfile max.cap.rpt
```

check max cap violations based on frequency

manually set max cap limits per freq in liberty using `set_max_cap_per_freq` command

note that lib name is library name inside dotlib, e.g. library (tcbn65lplvtwc)

read activity file for frequency annotation

```
set_default_switching_activity \
-input_activity 0.2 \
-seq_activity 0.15 \
-clock_gates_output 2.0 \
-period 7.0


propagate_activity

write_tcf dma_mac.tcf

read_activity_file \
-format TCF \
-set_net_freq true dma_mac.tcf

set_max_cap_per_freq \
-force \
-lib tcbn65lplvtwc \
-freq {50e6 70e6 100e6 200e6} \
-cap {5.0 4.0 3.0 1.5} \
-pins {* *}

report_design_rule \
-cap \
-max \
-outfile max.capperfreq.rpt
```

check max tran violations based on frequency

manually set max tran limits per freq in liberty using `set_max_tran_per_freq` command

```
set_max_tran_per_freq \
-force \
-lib tcbn65lplvtwc \
-freq {50e6 70e6 100e6 200e6} \
-tran {5.0 4.0 3.0 1.5} \
-pins {* *}

report_design_rule \
-tran \
-max \
-outfile max.tranperfreq.rpt
```

check frequency violation

requires dotlib with em templates

```
report_freq_violation \
-detailed \
-outfile freq.rpt
```

23

Self-Heating Effect Analysis

- Overview on page 571
  - FEOL Self-Heat (delta T FEOL) on page 572
  - BEOL Self-Heat (delta T RMS) on page 572
- Data Requirements for SHE Analysis on page 573
- Self-Heating Analysis Flow on page 574
  - Sample Run Script on page 575
- View SHE Analysis Plots in Voltus on page 576
- Delta Temperature File Formats on page 580

Overview

The 16nm FinFET process node is rapidly becoming the preferred choice for advanced Integrated Circuit (IC) designs. The self-heating effects (SHE) of FinFET has become very significant and need to be taken into account in device modeling and circuit simulation. Conventional self-heating compact models, such as those that use auxiliary thermal resistance and capacitance RC circuits, are inefficient in simulation because of the additional nodes introduced in RC thermal network and therefore increased circuit matrix size. In addition, FinFET compact models also become very expensive to evaluate because large amount of temperature-dependency terms and derivatives must be computed.

Self-Heating Effect is driven by front-end-of-line (FEOL) self-heat and back-end- of-line (BEOL) self-heat.

- FEOL is the first portion of IC fabrication where the individual devices (transistors, capacitors, resistors, etc.) are patterned in the semiconductor. FEOL generally covers everything up to (but not including) the deposition of metal interconnects layers.

© 2013-2023 All Rights Reserved.

- BEOL is the second portion of IC fabrication where the individual devices (transistors, capacitors, resistors, etc.) get interconnected with wiring on the wafer. Both contribute towards the delta T for self-heating.

FEOL Self-Heat (delta T FEOL)

Temperature increase is a function of # of fins and # of fingers, given the same FinFET structure. Also, temperature increase is linear to power. Both contribute towards self-heating. To calculate delta T FEOL:

- it must be calculated based on Instance
- depending on instance, power and cell-based fin # and finger # is calculated from SHF
- thermal resistance depends on cell type, core versus IO

BEOL Self-Heat (delta T RMS)

Temperature increase is a function of Irms through metal and the metal width. Interconnect wires create the self-heating effect. To calculate delta T BEOL, you need to:

PG Net Analysis:

1. Perform dynamic power and IR drop analysis.
2. Calculate resistor I_{rms} current.
3. Calculate delta T on each PG resistor (fractured wire segment)

For Signal Net Analysis:

1. Perform signal EM analysis based on same switching activities
2. Get I_{rms} on each wire segment
3. Calculate delta T on each signal wire segment

To calculate final delta T on BEOL, the software considers the RMS current on BEOL and interaction effect from FEOL.

© 2013-2023 All Rights Reserved.

Data Requirements for SHE Analysis

The table below lists the inputs that are required for SHE analysis in Voltus:

Data	Mandatory	Details
Timing Libraries	Yes	Synopsys libraries (dot libs) and PVT corner for analysis.
Verilog	Yes	Verilog netlist
SDC	Yes	SDC timing constraints file
VCD or TCF	Yes	Full VCD file for vector-based analysis
LEF	Yes	Technology LEF, standard cells, IOs, memories and IP LEF.
DEF	Yes	DEF or Multiple DEFs for design top level and blocks
SPEF	Yes	Flattened or multiple SPEFs for top level and blocks
Spice Subckts	Yes	Spice netlist for all cells in the design along with the Spice models. It is recommended that the supplied Spice netlist for macros/memories and IO cells have device X/Y location file.
GDS	Yes	Design components for IOs, memories, and IPs
Extraction tech file	Yes	For Quantus or process file
ElectroMigration (EM) rules	Yes	Current limits per unit area for each process layer. Usually available through foundry. It should contain the I_{RMS} equation.
Alpha parameter	Yes	Need Alpha parameter for all layers. It is the metal layer effect.
Beta Parameter	Yes	Thermal Coupling Effect coefficient value
Instance Power File	Yes	A file for Instance and its static power. Use half of total power for SHE table look-up. It is a simple 2 column format of <instance_name> and <total_power>

© 2013-2023 All Rights Reserved.

Thermal resistance File	Yes	A file containing instance name, finger number, and fin number. It is to derive the FOEL delta T calculation. Characterization on <finger_num> and <fin_num> is based on PMOS only, thus only half of total power should be used.

Self-Heating Analysis Flow

The following diagram illustrates the self-heating analysis flow:

The following key tasks are performed for the SHE flow:

1. Load the design (LEF/DEF/Verilog/Timing Libs/SDC)
2. Specify activity for the design (TCF/VCD/FSDB)

© 2013-2023 All Rights Reserved.

3. Run `report_power` to calculate instance-based static power and dynamic currents on power network.
4. Run the `analyze_self_heat` command to specify the number of tiles, layer-based α coefficients, β coefficients, and list of cells with number of fins/fingers. This command will use the static power data to compute delta T FEOL for all instances in the design. This command:
   - computes the RMS currents on signal wires
   - computes the RMS currents on power network wires
   - generates Detail Delta Temperature File (ddt), Instance Delta Temperature File (idt), and Tile Delta Temperature File (tdt)

Tip

Instead of using the `analyze_self_heat` command, you can use the `analyze_joule_heat` command if you need to analyze only the BEOL self-heat (delta T RMS) without analyzing the FEOL self-heat and any coupling effects from FEOL to BEOL.

Sample Run Script

##Setup Design

```
read_lib -lef  {  TECH/file.lef  }
read_lib -typ LIB/library1.lib
read_verilog VERILOG/top.v
set_top_module test
read_def  DEF/design.def
read_sdc SDC/top.sdc
create_library_set -name libs_min -timing library2.lib
create_rc_corner -name nominal -qx_tech_file qrcTechFile
create_delay_corner -name corner_min -rc_corner nominal
create_constraint_mode -name setup_func_mode -sdc_file SDC/top.sdc
create_analysis_view -constraint_mode setup_func_mode -name setup_func -
delay_corner corner_min
create_analysis_view -constraint_mode setup_func_mode -name hold_func -
delay_corner corner_min
set_analysis_view -setup [list setup_func] -hold [list hold_func]
set_analysis_mode -analysisType onChipVariation -checkType hold
read_spef -rc_corner nominal  SPEF/top.spef
set_design_mode -process 16
```

Power Analysis

```
set_default_switching_activity -input_activity 0.2 -seq_activity 0.1\
                               -clock_gates_output_ratio 2.0
propagate_activity
set_power_analysis_mode -reset
set_power_analysis_mode -method static
```


© 2013-2023 All Rights Reserved.

```
set_power_output_dir static_db
report_power -rail_analysis_format VS -outfile static_power.rpt
```

Rail Analysis

```
set_pg_nets -net VDD -voltage 0.80 -threshold 0.72 -tolerance 0.2 -force
set_pg_nets -net VSS -voltage 0.00 -threshold 0.08 -tolerance 0.2 -force
set_rail_analysis_domain -name ALL -pwrnets {VDD} -gndnets VSS
set_rail_analysis_mode -verbosity true -method dynamic -accuracy hd -
vsrc_search_distance 100 -power_grid_library
{./pgv/std_sign_off_voltus/techonly.cl} -process_techgen_em_rules true -
temperature 110
set_power_data -format current -scale 1 -bias_voltage 1.0
./dynamic_db/dynamic_VDD.ptiavg
set_power_pads -net VDD -file vdd.ploc -format xy
set_power_pads -net VSS -file vss.ploc -format xy
```

##Analyze Self-heat

```
set_advanced_rail_options -reset
analyze_self_heat -domain ALL \
  -alpha_parameters {{M1 0.47 0.58} {M2 0.40 0.58} {M3 0.34 0.58} {M4 0.28 0.58}
{M5 0.24 0.58} {M6 0.20 0.58} {M7 0.20 0.58} {M8 0.20 0.58} {M9 0.20 0.58} {M10
0.20 0.58} {M11 0.20 0.58}}\
  -beta_parameters {0.0048 -0.0025 0.6496}\
  -cell_thermal_resistance_file thermal_resistence_file\
-instance_power_file static_power.rpt \
  -tiles {100 100} \
  -scale_rms_limit 1.32 \
  -instance_delta_temperature_file sh_itm.txt \
  -tile_delta_temperature_file sh_ttm.txt \
  -detail_delta_temperature_file power_detail.rpt \
  -report_conn_pin_wire self_heat.rpt
read_power_rail_results -detail_delta_temperature_file power_detail.rpt
set_power_rail_display -plot ddt
```

The following is an example for the `analyze_joule_heat` command:

```
analyze_joule_heat -domain ALL -tile_delta_temperature_file
joule_ttm.txt -tiles {10 10} -detail_delta_temperature_file
joule_heat_detail.rpt -report_conn_pin_wire joule_heat.rpt
```

Important

Self-heat analysis (`analyze_self_heat`/`analyze_joule_heat`) should be performed for the entire design, that is, you must use domain analysis with all PG nets (`-domain ALL`). The `-net` *<pgnet>* parameter should be used only for preliminary checking.

View SHE Analysis Plots in Voltus

To view the SHE analysis plots using command line:

© 2013-2023 All Rights Reserved.

1. Load the detail/instance/tile delta T files using the following parameters of the `read_power_rail_results` command:
   - ❑ `-detail_delta_temperature_file`
   - ❑ `-instance_delta_temperature_file`
   - ❑ `-tile_delta_temperature_file`
2. Specify the `ddt`/`idt`/`tdt` plot type using the following command:

   ```
   set_power_rail_display -plot ddt
   ```

To view the SHE analysis plots using GUI:

1. Select *Power & Rail - Power & Rail Plots*.

   The following form appears:

2. Click *DB Setup*.

© 2013-2023
All Rights Reserved.

The following form appears:

3. Click the *Browse* button next to the *Rail Database* field to load the rail data.
4. Load either detail, instance, or tile delta T files in the corresponding field in the *Instance Files* section.
5. Click *OK*.
6. In the Power & Rail Plots form, click the *Rail* radio button.
7. Select the plot type (*ddt/idt/tdt*) corresponding to the loaded delta T file from the drop-down list.
8. Click *Apply*.

   The temperature map appears in the *Layout* tab.

The following types of SHE analysis plots are available:

© 2013-2023 All Rights Reserved.

- Detail Delta Temperature File (ddt) - Reports the increase in temperature (deltaT) for all metal layers in the entire design. This plot provides a detailed view of the temperature variations on the design.

- Instance Delta Temperature File (idt) - Reports the increase in temperature (deltaT) for all instances in the entire design.

© 2013-2023 All Rights Reserved.

- Tile Delta Temperature File (tdt) - Reports the increase in temperature (deltaT) per tile based on the number of tiles, specified using the `analyze_self_heat -tiles` parameter. The design on the layout is divided into sections based on the number of tiles that are specified, for example, 100x100. On the layout, the tiles are colored on the basis of the temperature increase in each tile. This is shown in the figure below:

Delta Temperature File Formats

Detail Delta Temperature File (ddt) - Self-Heat Analysis

© 2013-2023 All Rights Reserved.

```
# Report Format :
# LayerName  I_rms(mA)  Alpha Beta T_FEOL T_RMS T_BEOL  ll_x ll_y  ur_x  ur_y
# Net: IN01
# Net: IN02
M2  0.0002  0.8000  0.8429  0.2363  0.0000  0.1594 (4939.380 1177.008) (4939.397
1177.040)
M2  0.0002  0.5500  0.8429  0.2092  0.0000  0.0970 (4939.315 1177.008) (4939.380
1177.040)
```

Detail Delta Temperature File (ddt) - Joule Heat Analysis

```
# Report Format :
# LayerName I_rms(mA) dT_RMS  ll_x     ll_y       ur_x     ur_y
# Net: IN01
 M2       0.0380    0.0511  (144.684  118.310) (144.720 118.330)
 M3       0.0380    0.1023  (144.708  118.320) (144.732 120.040)
 ...
```

Instance Delta Temperature File (idt)

```
# deltaT_FEOL
instance_name
2.174284e-03       inst1
2.157267e-03       inst2
2.158814e-03       inst3
...
```

Tile Delta Temperature File (tdt)

DIE_STACK_TEMPERATURE_MAP	
DIE_AREA	xmin(um) ymin(um) xmax(um) ymax(um) nx(int) ny(int)
NUMBER_OF_LAYERS	nLayers(int)
LAYER	layer_name

© 2013-2023 All Rights Reserved.

```
Temperature
XCOR 0 ...
YCOR 0 ...
T0(0,ny-1)  T0(1,ny-1)  ...    T0(nx-1,ny-1)
T0(0,ny-2)  T0(1,ny-2)  ...    T0(nx-1,ny-2)
:
T0(0,0)     P0(1,0)            P0(nx-1,0)
END_TEMPERATURE
END_LAYER

LAYER                  layer_name1
:
END_LAYER
:
END_DIE_STACK_TEMPERATURE_MAP
```

where,

- `DIE_AREA` specifies the coordinates of the die area. `nx` and `ny` is the tile number in the temperature map file.
- `NUMBER_OF_LAYERS` is the number of layers in each die instance.
- `LAYER` specifies the LEF layer name.
- `XCOR` and `YCOR` are coordinates of each layer
- `T(0,0)` …. is the temperature value assigned to a tile at a specific coordinate.

Following is an example of the tile-based delta T file:

```
DIE_STACK_TEMPERATURE_MAP
DIE_AREA 0 0 4999.86 1213.06 10 10
NUMBER_OF_LAYERS 12
LAYER FEOL
TEMPERATURE
XCOR 0 499.986 999.972 1499.96 1999.94 2499.93 2999.92 3499.9 3999.89 4499.87
4999.86
YCOR 0 121.305 242.61 363.915 485.22 606.525 727.83 849.135 970.44 1091.74 1213.06
17.1648 17.1648 17.1648 17.1648 3.79183 1.60646 17.1648 17.1648 17.1648 17.1648
2.25187 2.25187 2.25187 2.25187 0.883812 0.883812 2.25187 2.25187 2.25187 2.25187
```


© 2013-2023 All Rights Reserved.

```
0.185081 0 0 0 0 0 0 0 0 0.0172782
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 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
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
END_TEMPERATURE
END_LAYER


LAYER M1
TEMPERATURE
XCOR 0 499.986 999.972 1499.96 1999.94 2499.93 2999.92 3499.9 3999.89 4499.87
4999.86
...
END_TEMPERATURE
END_LAYER
END_DIE_STACK_TEMPERATURE_MAP
```


© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

24

Statistical ElectroMigration Budgeting

- Introduction to Statistical ElectroMigration Budgeting on page 2
- Statistical ElectroMigration Budgeting Flow on page 4
- TCL Command Parameters for Running the SEB Flow on page 5
- Setting up and Running Power EM and Signal EM Analyses for FIT Calculation on page 6
- Output Reports on page 12
- Viewing FIT GUI Map on page 19

© 2013-2023 All Rights Reserved.

Introduction to Statistical ElectroMigration Budgeting

Statistical ElectroMigration budgeting (SEB) enables you to meet the ElectroMigration (EM) reliability targets by performing a statistical interconnect EM analysis based on the current modeling and thermal analysis of the design. This feature leverages the statistical nature of EM reliability to compute the level of the EM risk for all design rule violations. SEB analysis requires fixing of only those violations that add significant risk. This reduces the design verification time and improves performance while ensuring that the design meets its chip-level reliability goals.

The Voltus SEB flow involves:

- Measuring the total chip failure rate by EM analysis
- Checking if the total failure rate meets the product requirement

Failures in Time (FIT) is a factor to represent the chip failure rate. Following is the definition of FIT:

1 FIT = One failure per 10^9 product hours

= One failure in 1000 products through 10^6 hours

= One failure in 10^5 products through 10^4 hours

= or any other combination between products and hours as long as the product of the two is 10^9

Voltus will calculate the FIT value for each resistor of metal and via. The tool will follow the foundry EM definition to calculate FIT.

The ElectroMigration FIT equation is given below:

$$FIT = -\frac{10^9}{LT} \times \ln\left(\frac{1}{2} - \frac{1}{2} erf\left(\frac{\ln\left(\left(\frac{I_{DC}}{I_{\text{limit}}}\right)^n \times \left(\frac{LT}{MTF(T_{new})}\right)\right)}{\sigma\sqrt{2}}\right)\right)$$

© 2013-2023 All Rights Reserved.

The following is the detailed description of the Median Time to Failure (MTF) equation:

$$MTF(T_{new}) = MTF(T_{ambient}) \times \exp\left(\frac{Ea}{k}\left(\frac{1}{T_{new}} - \frac{1}{T_{ambient}}\right)\right)$$

$$T_{new} = T_{ambient} + \Delta T$$

- LT = lifetime of product in hour
- erf() = error function
- S_{DC} = DC EM severity ratio, I_{DC}/I_{limit}
  - I_{DC}: DC current of design
  - I_{limit}: DC EM limit from foundry design rule manual and calculated at $T_{ambient}$
- $MTF(T_{ambient})$ = median time to failure in hour (from foundry)
- σ= sigma, spread in lifetime distribution (from foundry)
- n = current density exponent (from foundry)
- Ea = activate energy (from foundry)
- k = Boltzmann's constant
- T = absolute temperature in K

© 2013-2023 All Rights Reserved.

Statistical ElectroMigration Budgeting Flow

The following diagram illustrates the SEB analysis flow:

The following key tasks are performed for the SEB flow:

1. Load the design (LEF/DEF/Verilog/Timing Libs/SDC) and apply the signal EM settings.
2. Perform static and/or dynamic power analysis. The static power analysis step is optional and is not required if you have an instance power file.
3. Perform self-heating analysis to evaluate the BEOL and FEOL self-heat and any coupling effects from FEOL to BEOL, or just Joule-heating analysis (`delta_T_RMS`), to calculate the final BEOL temperature rise effect.

   Note: Since the VIA layer does not have the RMS EM rules, `delta_T_RMS` cannot be calculated for the VIA layer. VIA's `delta_T` is always decided by the max `delta_T` of the connecting upper and lower metal.
4. Perform SEB FIT calculation based on the final BEOL temperature rise to generate the power and signal SEB report.

© 2013-2023
All Rights Reserved.

TCL Command Parameters for Running the SEB Flow

The following parameters are specified using the `set_rail_analysis_mode` command for the PG SEB flow or the `set_signal_em_analysis_mode` command for the signal SEB flow:

- `-check_thermal_aware_em`
- `-enable_seb`
- `-env_temperature`
- `-read_detail_delta_temperature_file`
- `-seb_lifetime`
- `-seb_table`
- `-seb_temperature`
- `-use_rms_delta_t`

For detailed information on these parameters, refer to *Voltus Text Command Reference*.

© 2013-2023 All Rights Reserved.

Setting up and Running Power EM and Signal EM Analyses for FIT Calculation

This section describes how to set up the power EM and signal EM analysis modes, and then calculate FIT.

Required Input Files

- Design data (LEF, netlist, DEF, SPEF, SDC)
- Power-grid libraries (.cl)
- Current files created during static/dynamic power calculation

In addition to these generic inputs for the EM analysis flow, the following specific inputs are required (**provided by foundry**):

- ICT-EM file - this file contains the EM model/rules to calculate the EM limits for power net EM analysis.
- Cell thermal resistance (Rth) file - this file contains the list of cells and their corresponding thermal resistance value that is used to calculate FEOL delta T.
- Thermal coupling coefficients - these parameters specify the thermal coupling factor between devices and metal layers.

Procedure

To perform power EM and signal EM analyses, complete the following steps:

Section 1: Self-Heat Analysis

1. Load the design data.

   Tcl command:

```
read_lib -lef $lefs
read_lib $lib
read_verilog $verilog
set_top_module $top_module
read_def $def
read_sdc $sdc
```


© 2013-2023 All Rights Reserved.

2. Specify the MMMC mode settings to apply the power EM settings.

   Tcl command:

```
create_library_set -name libs_min -timing $lib
create_rc_corner -name nominal -qx_tech_file $qrcTechfile
create_delay_corner -name corner_min -library_set libs_min -
rc_corner nominal
create_constraint_mode -name setup_func_mode -sdc_files $sdc
create_analysis_view -constraint_mode setup_func_mode -name
setup_func -delay_corner corner_min
create_analysis_view -constraint_mode setup_func_mode -name
hold_func -delay_corner corner_min
set_analysis_view -setup [list setup_func] -hold [list hold_func]
set_analysis_mode -analysisType onChipVariation -checkType hold
read_spef -rc_corner nominal  SPEF/top.spef
set_design_mode -process 7
set_delay_cal_mode
```

3. Optional. Define the static power analysis mode, and specify the sub-directory that will contain the power results. If you have the instance power file, the static power analysis step is not required.

   Tcl command:

```
set_power_analysis_mode -method static
set_power_output_dir      static_db
report_power -outfile static_power.rpt
```

4. Define the dynamic power analysis mode, and specify the sub-directory that will contain the power results.

   Tcl command:

```
set_power_analysis_mode \
    -disable_static             false \
    -write_static_currents      true \
    -binary_db_name             dynvectorlessPower.db \
    -create_binary_db           true \
    -method                     dynamic_vectorless \
set_power_output_dir            dynamic_db
```

5. You can define default activities for the unassigned nets.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_default_switching_activity -global_activity 0.15 \
                               -clock_gates_output_ratio 2.0
```

6. Specify the activity file.

   Tcl command:

```
propagate_activity -set_net_freq true
write_tcf out.tcf
```

   Note: `write_tcf` is an optional command that is specified to perform sanity checks.

7. Run power analysis.

   Tcl command:

```
report_power
```

8. Optional. If you do not load the CPF file, use the `set_pg_nets` command to define the power nets in the design, and assign attributes to these nets. Then, use `set_rail_analysis_domain` to define the power domains.

   Tcl command:

```
set_pg_nets -net VDD -voltage 0.9 -threshold 0.72
set_pg_nets -net VSS -voltage 0.0 -threshold 0.08
set_rail_analysis_domain -name ALL -pwrnets {VDD} -gndnets VSS
```

9. Specify the location of the current files generated during dynamic power analysis.

   Tcl command:

```
set_power_data -format current -scale 1 \
                               -bias_voltage 1.0\
                               $dynamic_current_list
```

10. Define the location of the voltage sources for each power/ground net.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_power_pads -net VDD -format xy -file ../PLOC/VDD.ploc
set_power_pads -net VSS -format xy -file ../PLOC/VSS.ploc
```

11. Specify the parameters for power RMS analysis:

Tcl command:

```
set_rail_analysis_mode \
    -method dynamic \
    -accuracy hd \
    -power_grid_library $pgv\
    -temperature 125\
    -ict_em_models $ictem
```

Note: Dynamic and RMS analyses are required to calculate Joule heat.

12. Perform self-heat analysis.

Tcl command:

```
analyze_self_heat -domain ALL \
    -alpha_parameters {{M1 $alpha_factor1 $alpha_factor2}...} \
    -beta_parameters {$beta_factor1 $beta_factor2 $beta_factor3}\
    -cell_thermal_resistance_file TRF.txt\
    -instance_power_file   ./static_db/static_power.rpt \
    -instance_delta_temperature_file Inst_Delta_Temp_Report.txt \
    -tile_delta_temperature_file sh_ttm.txt \
    -tiles {10 10} \
    -detail_delta_temperature_file power_detail.rpt \
    -report_conn_pin_wire self_heat.rpt
```

Section 2: Power FIT Analysis

1. Define the rail analysis mode with the SEB parameters.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_rail_analysis_mode\
    -method                     static \
    -accuracy                   hd \
    -temperature                125 \
    -power_grid_library $pgv \
    -enable_seb true \
    -seb_table seb_table.txt \
    -env_temperature 85 \
    -em_threshold 0.0 \
    -check_thermal_aware_em true \
    -read_detail_delta_temperature_file power_detail.rpt \
    -ict_em_models $ictem
```

2. Specify the power nets, power domain, location of the voltage sources, and static current files created during power analysis:.

Tcl command:

```
set_pg_nets -net VDD -voltage 0.9 -threshold 0.7
set_pg_nets -net VSS -voltage 0.0 -threshold 0.05
set_rail_analysis_domain -name ALL -pwrnets {VDD} -gndnets {VSS}
set_power_data -format current { \
     static_db/static_VDD.ptiavg \
    static_db/static_VSS.ptiavg \
    }
set_power_pads -net VDD -format xy -file ../PLOC/VDD.ploc
set_power_pads -net VSS -format xy -file ../PLOC/VSS.ploc
```

3. Run rail analysis.

Tcl command:

```
analyze_rail -type domain -output static_rail core
```

The Voltus console will display rail analysis statistics, as shown in the following snippet:

```
Rail Analysis completed successfully.


Finished Rail Analysis at 11:56:53 08/20/2018 (cpu=0:00:20, real=0:00:34, peak
mem=1218.10MB)
```


© 2013-2023 All Rights Reserved.

```
Current Voltus IC Power Integrity Solution resource usage: (total cpu=0:03:56,
real=0:58:07, mem=1218.10
```

Section 3: Signal FIT Analysis

1. Run signal EM analysis.

   Tcl command:

```
set_signal_em_analysis_mode
    -method {avg} \
    -avgRecovery 0 \
    -detailed \
    -useQrcTech \
    -use_db_freq \
    -report seb_reduce.rpt \
    -enable_seb \
    -seb_table seb_table.txt \
    -env_temperature 85 \
    -seb_lifetime 43800 \
    -check_thermal_aware_em \
    -read_detail_delta_temperature_file power_detail.rpt \
    -ict_em_models $ictem
verify_AC_limit
```

   The Voltus console will display the number of violations, as shown in the following snippet:

```
Num Violations: 72

**************End VerifyACLimit****************
   (CPU Time =0:03:56 MEM=23.020M)
```


© 2013-2023 All Rights Reserved.

Output Reports

Power SEB Report

The rail output directory contains the power SEB report:

Directory Name	Description
*<domainName>*_<temperature>_dynamic_*<X>*	For each run, the state directory is generated in the rail output directory. The directory has the default name *<domainName>*_*<temperature>*_dynamic_*<X>* for the dynamic analysis run, and *<domainName>*_*<temperature>*_avg_*<X>* for the static analysis run. *<domainName>* will be replaced with ALL if all domains are to be analyzed (analyze_rail -output *dir* -type domain ALL), or with *<netName>* if a power net is to be analyzed. Following is an example of the report path: ALL_125C_avg_1/Reports/VDD/**VDD.rj.avg.rpt**

Sample filename: VDD.rj.avg.rpt

Format:

I/Ilimit I(mA) Ilimit(mA) Layer Location {X1 Y1 X2 Y2} ShortLength (um) Width (um) ViaArea(nm^2) Via Cuts Fix Width/Via Current Direction Benefit Factor J(A/cm^2) Jmax(A/cm^2) ResistorID **DT_FEOL** **DT_RMS** **EM_Temperature** **SEB_Temperature** **Sdc** **MTF** **FIT** **Reference_FIT_R**

Format Description:

I/Ilimit specifies the current value that flows through the metal/via resistor that is calculated by the tool divided by the current limit value provided by the foundry.

I(mA) specifies the current value that flows through the metal/via resistor that is calculated by the tool.

Ilimit(mA) specifies the current limit value based on DC EM rules that is provided by the foundry. It depends upon the width and thickness of the wire segment.

Layer specifies the metal/via layer name on which the resistor is located.

© 2013-2023 All Rights Reserved.

`Location {X1 Y1 X2 Y2}` specifies the coordinates of the resistor.

`ShortLength` is also known as "Blech length" or "Path length". It can be explained as "the longest and continuous centerline path from edge to edge among the connected wire shapes on the same metal layer".

`ShortLength` is the drawn length of the longest continuous same metal * layout scale factor (from qrctechFile).

`Width` specifies the width of the resistor.

`Via Area` is the drawn dimension via area in square microns.

`Via Cuts` is the number of via cuts.

`Fix Width/ Via` The “Fix Width/Via” is recalculated by modifying the existing via width to eliminate the EM violation. If `I/Ilimit` is greater than 1, which is a violation, Fix Width/Via will be:
`violation * via width`

`Current Direction` is applicable only to via layers where "D" denotes down and "U" denotes up.

`Benefit Factor` When certain constraints (blue-highlighted - “L>3W<0.4”) are met, its EM limit can be 2X, 3X, or multiple X of the base EM limit. These multiples are called Benefit Factor (red-highlighted in the following example - “1”, "2").

```
em_jmax_dc_avg EQU 1 * 10 * ( w - 0.3 ) jmax_factor 95 1.58 100 1.000 105 0.97
jmax_life 87600 1.000 L>3 W<0.04
```

```
em_jmax_dc_avg EQU 2 * 10 * ( w - 0.3 ) jmax_factor 95 1.58 100 1.000 105 0.97
jmax_life 87600 1.000 L>3 W>=0.04
```

Here, W is the wire width and L is the Blech length.

`J(A/cm^2)` is the current density value.

`Jmax(A/cm^2)` is the current density limit.

`ResistorID` is the unique ID assigned to a resistor.

`DT_FEOL` is the FEOL self-heat value derived from the thermal resistance of cells.

`DT_RMS` is the Joule-heat value derived from the resistor I_{rms} current.

`EM_Temperature` is the temperature within `delta_T` used to calculate the I_{limit}.

© 2013-2023 All Rights Reserved.

`SEB_Temperature` is the temperature used to update MTF with respect to the ambient temperature.

`Sdc` is the DC EM severity ratio calculated for the resistor. This value is based on the user-specified `env_temperature`.

`MTF` is the median time to failure that is calculated at `seb_temperature` by considering `delta_T` used for EM analysis.

$T_{(seb_temperature)} = T_{(env_temperature)} + delta_T$

- without self-heating

  `delta_T = delta_T_RMS`

- with self-heating

  `delta_T = delta_T_RMS + delta_T_SHE`

Note: The via layer does not have the RMS EM rules, therefore, it cannot calculate `delta_T_RMS`. Via's `delta_T` is based on the max `delta_T` of the connecting upper and lower metal.

`FIT` is the failure in time calculated for each resistor of metal and via.

`Reference_FIT_R` is the resistor that is used as a reference.

Sample Report: The snapshot of the report has been split into 2 parts, Part A and Part B.

Part A of the report

```
Report format:
I/Ilimit  I(mA)      Ilimit(mA)    Layer         Location            ShortLength    Width  ViaArea  Via   Fix Width/  Current    Benefit
ion  Reference_FIT_R
                                                 {X1 Y1 X2 Y2}       (um)           (um)   (nm^2)   Cuts  Via         Direction  Factor

198963  7.767793e-01  3.904148e+00  M10       {8.3670  87.9360  10.1400  87.9360}  258.2790  0.424  N/A    N/A  N/A    N/A    1.00
1891  7.382752e-01  3.904148e+00  M10       {8.3670  92.2200  10.1400  92.2200}  258.2790  0.424  N/A    N/A  N/A    N/A    1.00
185068  7.225341e-01  3.904148e+00  M10       {8.3670  96.5040  10.1400  96.5040}  258.2790  0.424  N/A    N/A  N/A    N/A    1.00
```

Part B of the same report

```
  J(A/cm^2)    Jmax(A/cm^2)  ResistorID  DT_FEOL DT_RMS EM_Temperature    SEB_Temperature  Sdc  MTF  FIT


  1.263467e+06  6.350273e+06    5406  1.63  0.42 85.42  85.42  0.198963  83516.1  -0.000000e+00  -0.000000e+00
1.200838e+06  6.350273e+06    5512  0.20  0.38 85.38  85.38  0.1891  83802  -0.000000e+00  -0.000000e+00
  1.175234e+06  6.350273e+06    5618  1.04  0.36 85.36  85.36  0.185068  83958.4  -0.000000e+00  -0.000000e+00
```


© 2013-2023 All Rights Reserved.

The total FIT value is displayed in the header of `VDD.rj.avg.rpt` as following:

```
# Total FIT: 2.20000e-01
```


© 2013-2023 All Rights Reserved.

Signal SEB Report

By default, the current working directory contains the violations report file:

Directory Name	Description
`<filename>.rpt`	This file contains all the violations arranged in the descending order of the ratio of current calculated to the current limit. The violations are marked by '* ' at the end of the calculated current values. When the result shows that the current calculated is higher than the limit, a violation can be seen for that net. If there are no violations, there will be no entry in the report.
`<filename>.detailed.rpt`	The file contains the detailed report for all the nets, and is generated when the `-detailed` parameter is specified.

Sample filename: `seb_signal.detailed.rpt`

Format:

```
Iavg/Limit   Iavg   Limit   Width/CutArea   FixWidth/FixCutNum   Cap   Layer
(X1 Y1 X2 Y2)   PathLength   BlechLength   Current_Direction FIT
 ID  Ref_ID  SdcSEB  TdcSEB  dTrmsSEB  MtfNew  Tdc_dTrms
```

Format Description:

`Iavg/Limit` specifies the current value that flows through the metal/via resistor that is calculated by the tool divided by the current limit value provided by the foundry.

`Iavg` specifies the actual current value that flows through the metal/via resistor that is calculated by the tool.

`Limit(mA)` specifies the current limit value that is provided by the foundry depending upon its width and thickness.

`Width/CutArea` If the wire segment is metal, it is calculated as the actual width perpendicular to the current flow of the wire segment. If the wire segment is via, it is calculated as the area of the cut in the via.

`FixWidth/FixCutNum` is the width (for metal segment) or cutNum (for via) required to fix the violation. When there is a violation, it is the new width of the metal wire or the new number of cuts of via (drawn area).

© 2013-2023 All Rights Reserved.

`Cap` is the capacitance of the wire segment.

`Layer` specifies the metal/via layer name on which the resistor is located.

`(X1 Y1 X2 Y2)` specifies the coordinates of the resistor.

`PathLength` of a wire segment is the distance traced from the first pin shape of the driver to the center of the wire segment.

`Blechlength` is the longest and continuous centerline path from edge to edge among the connected wire shapes on the same metal layer.

`Current Direction` is applicable only to via layers where "D" denotes down and "U" denotes up.

`FIT` is the failure in time calculated for each resistor of metal and via.

`ID` is the original resistor ID.

`Ref_ID` is the ID of the resistor that is used as a reference.

`SdcSEB` is the DC EM severity ratio calculated for the resistor. This value is based on user-specified `env_temperature`.

`TdcSEB` is the temperature used to update MTF with respect to the ambient temperature.

`dTrmsSEB` is the Joule-heat value derived from the resistor I_{rms} current for SEB temperature.

`MtfNew` is the median time to failure that is calculated at `seb_temperature` by considering `delta_T` used for EM analysis.

$T_{(seb_temperature)} = T_{(env_temperature)} + delta_T$

- without self-heating

  `delta_T = delta_T_RMS`

- with self-heating

  `delta_T = delta_T_RMS + delta_T_SHE`

`Tdc` is the temperature within `delta_T` used to calculate the I_{limit}.

`dTrms` is the Joule-heat value derived from the resistor I_{rms} current.

Sample Report: The snapshot of the report has been split into 2 parts, Part A and Part B.

© 2013-2023 All Rights Reserved.

Part A of the report

Iavg/Limit	Iavg	Limit	Width/CutArea	FixWidth/FixCutNum	Cap	Layer	(x1	y1	x2	y2)	PathLength	BlechLength
0.01	0.005787	0.442655	4.00e-04	/	0.029150	VIA1	(144.674	118.310	144.694	118.330)	0.000	0.320
0.02	0.005783	0.331991	4.00e-04	/	0.029127	VIA2	(144.710	118.310	144.730	118.330)	0.036	0.219
0.01	0.005737	0.422459	4.00e-04	/	0.028900	VIA3	(144.710	120.030	144.730	120.050)	1.751	1.800
0.01	0.005707	0.853949	0.001444	/	0.028748	VIA4	(144.057	120.021	144.095	120.059)	2.395	0.758

Part B of the same report

Current_Direction	Fit	ID	Ref_ID	SdcSEB	TdcSEB	dTrmsSEB	MtfNew	Tdc	dTrms
Up	0.0	1	1	0.013074	85.051102	0.051100	8.60e+04	85.051102	0.051100
Up	0.0	2	2	0.017418	85.102303	0.102300	8.56e+04	85.102303	0.102300
Up	0.0	3	3	0.013581	85.102303	0.102300	8.57e+04	85.102303	0.102300
Up	0.0	4	4	0.006683	85.014099	0.014100	8.63e+04	85.014099	0.014100

The total FIT value is displayed at the end of the `seb_signal_detailed.rpt` file as the following:

```
EM Violation Summary(Net Analyzed):
#                    RMS          PEAK          AVG          ALL        DESIGN
Nets                   0             0            1            1        204638
Clock Nets             0             0            1            1          1523
Data Nets              0             0            0            0        203115
Wire Segments          0             0            1            1       4522985
Total Fit:                                                1.113992e-05
~
~
```

The major portion of the total FIT at the design level is from the power and ground nets. The portion from the signal nets is usually negligible because of the small average current due to the behavior of the rise and fall currents canceling each other.

© 2013-2023 All Rights Reserved.

Viewing FIT GUI Map

To visualize FIT values using the rail results viewer, complete the following steps:

1. Load the design data through the command line or through the GUI.

   Tcl command:

   ```
   read_lib -lef $lefs
   read_lib $lib
   read_verilog $verilog
   set_top_module $top_module
   read_def $def
   ```

   GUI: Select *File - Read Design*.

   Note: This step is required only if FIT analysis is running in a different Voltus session. If the analysis is running in the same session, perform step 2.

2. Load the rail data through the command line or through the GUI.

   Tcl command:

   ```
   read_power_rail_results -rail_directory VDD_125C_avg_1
   ```

   GUI:

   a. Select *Power Rail - Power Rail Plots*.

   The Power Rail Plots form appears. This form is the main window for displaying the rail analysis results.

   b. Click the *DB Setup* button.

   The Power & Rail Setup form appears.

   c. You can enter the rail output directory or click *Browse* next to the *Rail Database* field, and select a location from the `Rail State Directory` browser window.

   d. Click *OK*.

3. In the Power Rail Plots form, select the *Rail* radio button and select the ***fit*** rail analysis plot type from the drop-down list.

© 2013-2023 All Rights Reserved.

Tcl command:

```
set_power_rail_display -plot fit
```

The ***fit*** plot will be overlaid on the design.

© 2013-2023
All Rights Reserved.

25

Body Bias Analysis

- Overview on page 21
- Library Generation on page 22
- Power Analysis on page 24
- Rail Analysis on page 26

Overview

Body biasing is used at advanced nodes for the following reasons:

- Avoid latch-up
- Optimize sub-threshold leakage
- Optimize performance

Generally, WELL contacts in a design are tied to core VDD and VSS. In order to optimize sub-threshold leakage power consumption, body voltage is applied to control `Vth` of the transistor using dedicated body bias supply rails. Reverse body bias is applied to low `Vth` devices during idle mode to reduce leakage, and forward body bias is applied to high `Vth` devices during active mode to achieve higher performance. The body voltage is generally much lower than the voltage of the core supply rails because high reverse body bias will produce higher leakage, and high forward body bias can result in excessive current through device.

© 2013-2023 All Rights Reserved.

Example of back-bias supply with NMOS is shown below.

When `VPW < Source`, `Vth` increases, `Id` and `leakage` decreases.

Following is a pictorial representation of well tap cells and body bias connectivity:

Library Generation

For body bias analysis, cells with body bias pins needs to be included during power-grid view library generation. The body bias pins should be defined in the cell LEF. Following is an

© 2013-2023
All Rights Reserved.

example of a standard cell (CORE) LEF, where `WELLPIN` has no connections within the cell, and would be an additional physical-only LEF pin:

```
MACRO STDCELL
...
PIN VDD
     PORT
          LAYER M1
               RECT ...
               ...
PIN NWELLPIN
     PORT
          LAYER NWELL
               RECT ...
               ...
          ...
     ...
...
PIN PWELLPIN
...
END STDCELL
```

Following is an example of a tap cell with body bias pin:

```
MACRO VBIASCELL
...
PIN VDD
     PORT
          LAYER M1
               RECT ...
               ...
PIN VBIAS
     PORT
          LAYER M1
               RECT ...
               ...
          LAYER NWELL
               RECT ...
               ...
          ...
     ...
...
PIN PWELLPIN(same as standard cell, unless
there is a n-channel bias in the triple well
case) * Could be part if the VSS pin as well.
END PWELLPIN
END VBIASCELL
```

Since the WELL layers are not processed during power-grid view library generation, you will have to map these layers to the diffusion layer in order to extract connectivity. This mapping can be specified using LEF-Technology layer map file specified using the following command:

© 2013-2023 All Rights Reserved.

```
set_pg_library_mode -lef_layermap <LEF:QRC technology mapping file>
```

An example of LEF layer map is given below:

```
####
diff NDIFF  lefdef NWELL
diff PDIFF  lefdef PWELL
diff NDIFF  lefdef NDIFF
diff PDIFF  lefdef PDIFF
via   CONT  lefdef CONT
metal MET1  lefdef MET_1
via   V1    lefdef V_12
…
```

You need to define bias pins in the library using the following command:

```
set_pg_library_mode -power_pins {pin1 voltage1} -ground_pins {pin}
```

By default, library generation ignores ports on the diffusion layer. In order to process body bias pin's ports on the diffusion layer for electrical analysis on body bias network, set the following command parameter to `true` during library generation.

```
set_advanced_pg_library_mode -process_bulk_diffusion_ports true
```

Once library generation is completed, check `<library>.report` to ensure that body bias pins are present for these cells and current taps are generated.

Power Analysis

The bias pin definitions are read from the liberty (dotlibs) libraries. Generally, no additional commands or parameters are needed to calculate power for designs with body bias as long as the liberty files are properly characterized with body bias power domain definitions.

Example of a liberty file with body bias pin definitions is as follows:

```
Library ( ) {
voltage_map (VDD, 1.08);
voltage_map (VNW, 0.3); // Body bias pin
voltage_map (VPW, -0.2); // Body bias pin
voltage_map (VSS, 0);
Cell () {
          pg_pin (VDD) {
            pg_type : primary_power;
            voltage_name : VDD;
            related_bias_pin : VNW;
          }
          pg_pin (VNW) {
            pg_type : nwell;
            voltage_name : VNW;
```


© 2013-2023 All Rights Reserved.

```
            physical_connection : device_layer;
          }
          pg_pin (VPW) {
            pg_type : pwell;
            voltage_name : VPW;
            physical_connection : device_layer;
          }
          pg_pin (VSS) {
            pg_type : primary_ground;
            voltage_name : VSS;
            related_bias_pin : VPW;
          }
 leakage_power () {
      value : 47.3572;
      when : "(A * !(B))";
      related_pg_pin : VDD;
    }
    leakage_power () {
      value : 0.0501967;
      when : "(A * !(B))";
      related_pg_pin : VNW;
    }
    leakage_power () {
      value : 0.0112439;
      when : "(A * !(B))";
      related_pg_pin : VPW;
    }
    leakage_power () {
      value : 0;
      when : "(A * !(B))";
      related_pg_pin : VSS;
    }
 internal_power () {
        related_pin : "A";
        related_pg_pin : VNW;
        rise_power (power_template_3x3) {..}
        fall_power (power_template_3x3) {..}
}
internal_power () {
        related_pin : "A";
        related_pg_pin : VPW;
        rise_power (power_template_3x3) { ..}
        fall_power (power_template_3x3) {..}
}
```

With body bias pin definitions in liberty, the power and ground pins with type pwell and nwell will be regarded as body bias pins. The absolute voltage of these bias pins will be used as reference voltage to calculate leakage and internal power for these pins.

If body bias pins are defined in LEF, but missing in the liberty file, power analysis cannot rely on LEF to infer if a PG pin belongs to a primary power domain or a body bias domain. In this case, power analysis calculates power for the cell and splits it between primary domain

© 2013-2023 All Rights Reserved.

(VDD:VSS) and body bias domain (VNW:VPW) which is not desirable. To avoid this, you can specify body bias pins using the following command parameter during power analysis:

```
set_power_analysis_mode –bulk_pins { VNW VPW }
```

In this case, if a liberty file does not have body bias definition, power analysis does not distribute power to the body bias domain.

Rail Analysis

Once the power-grid view library is generated for cells with body bias pins as described above and power analysis is run to compute power dissipation on these pins, you can run static and dynamic IRdrop analysis on body bias network. Through library generation, current taps (sinks) are added on diffusion ports.

In order to extract body bias PG network connectivity to diffusion ports of cell, and process current taps on body bias pins, specify the following command parameters during rail analysis:

- `set_rail_analysis_mode -process_bulk_pins_for_body_bias true`

  Default: `false`

  This parameter processes current taps assigned to bulk pins in power-grid view library. It is used during body bias analysis where current taps are assigned to bulk pins.

- `set_rail_analysis_mode -extractor_include ./mybbias.cmd`

  In the `mybbias.cmd` file, specify the following variable:

  ```
  setvar enable_toplevel_diff_extract true
  ```

  Default: `false`

  Extracts ports defined on the diffusion layer in power-grid view library. This variable is used during body bias analysis to extract connectivity in the diffusion layer.

Following is an example script for rail analysis:

```
set_power_pads -net bVDD -file BVDD.vsrc -format xy
set_power_pads -net bVSS -file BVSS.vsrc -format xy
set_pg_nets -net BVDD - -threshold 0.25  -voltage 0.3
set_pg_nets -net BVSS –threshold -0.25 -voltage -0.3
set_rail_analysis_mode -accuracy hd -method static \
    -power_grid_library ./mylibrary.cl \
    -process_bulk_pins_for_body_bias true \
    -extractor_include ./mybbias.cmd
```


© 2013-2023 All Rights Reserved.

```
set_power_data -reset
set_power_data -format current {static_BVDD.ptiavg static_BVSS.ptiavg}
analyze_rail -output ./result -type net BVDD
analyze_rail -output ./result -type net BVSS
```

Example of R network of a body bias power net:

© 2013-2023
All Rights Reserved.

© 2013-2023 All Rights Reserved.

26

Leakage Power Scaling using Liberate Library Files

At lower technology nodes, leakage power is a substantial component of the total power in a design. Therefore, it is important to accurately calculate the leakage power at different voltage and temperature corners. This section focuses on the use model for running power calculation for leakage power scaling in Voltus using Liberate generated liberty files.

Generating Liberty File Using Liberate

The flow for generating the liberty files using Liberate to enable leakage power scaling is given below:

1. Obtain cell leakage power from liberty files using one of the following methods:
   - If the cell group has the `cell_leakage_power` attribute, its value will be taken as the cell leakage power in one PVT corner.
   - If the cell group does not have the `cell_leakage_power` attribute, Liberate will take the value of the value attribute from the first `leakage_power` group that does not have the when attribute as the cell leakage power.
   - Otherwise, Liberate will skip generating scaling factors group for this cell with a warning message *Warning* (merge_leakage) : cannot determine the `leakage_power` value for the cell.

```
cell (AN2DP) {
  cell_leakage_power : 0.000491;
  area : 10467;
  cell_footprint : "an2d2";
  leakage_power () {
  value : 36.34567;
  related_pg_pin : VDD;
}
  leakage_power () {
  value : 20.623454;
  when  : "!A1 !A2 !Z";
```


© 2013-2023 All Rights Reserved.

```
  related_pg_pin : VDD;
}
```

2. Library characterization - there are two use models for performing library characterization. These are:

- ❑ when liberty files are present for all corners, use the following use model to generate the merged library file with the leakage scale factor table:

  a. read all P*V*T library files using the `read_library` command

  b. generate leakage scale factor table with the PVT information using the `merge_leakage` command

  A sample script is shown below:

```
read_library { \
         SS_1.0_25.lib  SS_1.0_85.lib  SS_1.0_125.lib \
         SS_1.2_25.lib  SS_1.2_85.lib  SS_1.2_125.lib \
         TT_1.0_25.lib  TT_1.0_85.lib  TT_1.0_125.lib \
         TT_1.2_25.lib  TT_1.2_85.lib  TT_1.2_125.lib \
}
merge_leakage  -process {SS TT} \
         -voltage {1.0 1.2 } \
         -temp {25 85 125 } \
         -base {TT 1.0 25}
write_library -ecsm ecsm.lib
```

- ❑ when liberty files are not present for all corners, use the following use model to generate the merged library file with the leakage scale factor table:

  a. Generate scaling library file for each process separately based on V*T

  b. Merge these scaling library files using the `merge_library` command

© 2013-2023 All Rights Reserved.

A sample script is shown below:

```
#
# 1. base_ff.tcl
# generate scaling library file for 'FF' process
#
...
read_library $flist
merge_leakage -process {1} \
              -voltage {0.99} \
              -temp {-40 0 125} \
              -base {1 0.99 0}

write_library -overwrite ff_base.lib

#
# 2. base_ffg.tcl
# generate scaling library file for 'FFG' process
#
# 3. base_tt.tcl
# generate scaling library file for 'TT' process
#
...
#
# 4. base_ss.tcl
# generate scaling library file for 'SS' process
#
...
# 5. merge.tcl
# merge library files together to generate the final scaling
library file
#
set libs {ff_base.lib ffg_base.lib tt_base.lib ss_base.lib}

merge_library -libname "base_corner" ff_base.lib
{ffg_base.lib tt_base.lib ss_base.lib}
```

Voltus Use Model for Leakage Power Scaling

Run the Voltus flow with the Liberate liberty files, as shown in the following script:

##Specify Liberate generated liberty files for leakage power scaling##

```
read_lib "merge.base.lib merge.hvt.lib"
read_verilog verilog_file
set_top_module top
read_def def_file
read_sdc sdc_file
read_spef spef_file
```

##Comment: Specify P/V/T parameters to use for each liberty file ##

```
create_op_cond -P 1 -T 100 -V 0.99 -name OP -library_file merge.base.lib
set_op_cond OP -library merge.base.lib
create_op_cond -P 1 -T 100 -V 0.99 -name OP1 -library_file merge.hvt.lib
set_op_cond OP1 -library merge.hvt.lib
report_power -leakage
```


© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

27

RTL Activity File Flow in Static Power Calculation

RTL Activity File Flow in Static Power Calculation

The toggle data from RTL simulations are applied to gate-level netlists in Voltus for power analysis. You can use the `map_activity_file` command of Voltus to specify instance name mapping between RTL netlist and GATE level netlist. The mapping file used for instance name mapping is generated by Conformal.

The following diagram illustrates the flow:

Conformal flattens and models the Golden and Revised designs and automatically maps the key points. In Conformal, the `REPORT MAPPED POINTS` command is used to display the mapped points that were automatically identified or added with the Conformal software. The mapping file generated using the `report mapped points` command is used in Voltus to

© 2013-2023 All Rights Reserved.

increase the annotation of the RTL activity file. This mapping is required to annotate waveform from the RTL switching activity (that is, VCD), to do power calculation on a post-layout netlist. RTL level VCD or TCF are obtained from simulation on the RTL Verilog netlist. The mapping file has information for each mapped point from the Golden and Revised design, and a summary of all Golden and Revised mapped points. The tool assumes that the Golden (G) refers to the RTL netlist and Revised (R) refers to the GATE level netlist.

By default, Conformal uses the primitive instance name of the D flip-flops (DFF) during reporting. For example:

```
LEC> report mapped point
...
34-th mapped points:
  (G) + 16  DFF  /x_reg[1]
  (R) + 16  DFF  /CDN_MBIT_a_reg[1]_MB_a_reg[2]_MB_a_reg[3]_MB_a_reg[4]/U$1
35-th mapped points:
  (G) + 17  DFF  /x_reg[2]
  (R) + 17  DFF  /CDN_MBIT_a_reg[1]_MB_a_reg[2]_MB_a_reg[3]_MB_a_reg[4]/U$2
...
```

Names like `U$1` are pin names of the DFF cells referred by Conformal when creating mapping. Sometimes, it is useful to know which library pin a particular DFF instance is referencing, especially when dealing with multibit DFFs. You can use the `set gate report -USE_LIBRARY_PIN_NAME` command option before generating a report. With this option, Conformal uses the library pin name instead of the primitive name in a gate report, if the gate comes from the library.

```
LEC> set gate report -USE_LIBRARY_PIN_NAME
LEC> report mapped point -long
...
34-th mapped points:
  (G) + 16  DFF  /x_reg[1]
  (R) + 16  DFF  /CDN_MBIT_a_reg[1]_MB_a_reg[2]_MB_a_reg[3]_MB_a_reg[4]/Q1
35-th mapped points:
  (G) + 17  DFF  /x_reg[2]
  (R) + 17  DFF  /CDN_MBIT_a_reg[1]_MB_a_reg[2]_MB_a_reg[3]_MB_a_reg[4]/Q2
...
```

In the TCL mode, the command option will be:

```
set_gate_report -USE_LIBRARY_PIN_NAME
```

Note the following when using the `-USE_LIBRARY_PIN_NAME` option:

- The option only affects the gate netlist and not the RTL because there are no library pins in the RTL to report.
- This option affects all the report commands. For example, `report mapped point`, `report gate`, `report path`, `write map point`, and so on.

© 2013-2023 All Rights Reserved.

- Use this option before saving a mapping file (`report mapped points -long`) for use in Voltus because the mapping file will contain the library pin name. However, this mapping file cannot be read back in Conformal.
- This command option is available in Conformal version 15.10-s180 or later.

Conformal makes use of the `SET NAMING RULE` command to specify special text handling rules, such as hierarchical separators, tristate naming rules, register naming rules, array delimiters, instance names, or variable names. This command makes the mapping more runtime efficient and reliable. However, these name changes can make the mapping file unusable.

To resolve the issues with the mapping file, add the comment syntax to block of the code related to the `SET NAMING RULE` command in the Conformal script (and force LEC to map renamed signals by function, and not by names). Although this produces a useable mapping file, it leads to higher runtime for Conformal.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

28

File Formats

- Library Generation on page 38
  - Current Region File on page 38
  - Cell Pin Net Map File on page 39
  - Cell Accura Data File on page 40
  - Cell Decap File on page 42
  - Cell Well Cap File on page 43
  - Circuit Include File on page 43
  - GDS Layermap File on page 44
  - LEF-DEF Layermap File on page 45
  - Pad Voltage Source File on page 46
  - Port File on page 49
  - Trigger File on page 50
- Power Analysis on page 60
  - Instance ASCII Power File on page 60
  - Instance Temperature File on page 61
  - TCF File on page 61
  - TWF File on page 63
- Rail Analysis on page 85
  - Dynamic Trigger File During Rail Analysis on page 85
  - Dynamic Trigger File During Rail Analysis on page 85
  - EM Models File on page 86

© 2013-2023 All Rights Reserved.

- EM Only ICT File on page 100
- Package Model and Mapping File on page 107
- Power-Up Sequence File on page 111

Library Generation

This section lists the LibGen file formats in alphabetical order.

Current Region File

A current region file is the definition of current for an area bounded by x,y coordinates. It includes a list of regions and the amount of current to be distributed within them for the power-grid. This will override any values calculated by Voltus library simulator. The dynamic current region file is used for dynamic power-grid view generation.

Sample file name: `current_region`

Command: `set_pg_library_mode {-current_distribution` *`current_region region_file`*`}`

Format:

```
UNIT CURRENT default_unit_in_amps
UNIT CAPACITANCE default_unit_in_farads
CELL cell_name
<PROP [FLAT | HIER]>
NET net_name
LAYER layer_name
[ REGION x1 y1 x2 y2 current_value <cap_value> | REGION current_value <cap_value> ]
```

Format Description:

`UNIT` defines the default scale to be used with the current and capacitance values read from the current region file. These scales apply to any value that is not followed by an explicit unit specification. Valid current values are: A (Amps), mA (miliAmps), uA (microAmps), and nA (nanoAmps). Valid capacitance values are: F (Farads), mF (miliFarads), uF (microfarads), nF (nanofarads), pF (picoFarads), and fF (femtoFarads).

`CELL` specifies the name of the cell. All the following `NET`, `LAYER`, and `REGION` commands will apply to this `CELL` until another `CELL` specification is encountered. Wildcards can be used in the `cell_name`.

`PROP [FLAT | HIER]` If the keyword `FLAT` is applied, LibGen will not perform hierarchical tracing for the given net in this cell. If the keyword `HIER` is presented, LibGen will perform

© 2013-2023 All Rights Reserved.

hierarchical tracing for the given net in this cell. This provides the ability for LibGen to automatically specify a current value to all instances of a cell in the design hierarchy. The assumption is that all of these instances of the cell will consume the same power. The default is `HIER`.

`NET` specifies that all the `LAYER` and `REGION` commands following it applies to the `net_name`. This stays in effect until another `NET` or `CELL` specification is encountered. Wildcards can be used in the `net_name`.

`LAYER` specifies that all the `REGION` commands following it apply to the `layer_name`. This stays in effect until another `LAYER`, `NET`, or `CELL` command is encountered. The `layer_name` must be a library layer name. If the layer is not found, LibGen will issue an error message and exit. You must provide a valid layer name for LibGen to run. If you specified a valid layer, but LibGen does not find any via in the assigned current region, LibGen will issue a warning message, but the process will still continue and exit successfully. In this case, you may get a 0 current value for the specified cell.

`REGION` specifies that the current value and optionally the capacitance value to be associated with the rectangular region specified by the bounding box (*`x1 y1 x2 y2`*). Current taps will be created at the locations of the vias or contacts specified by the most recent `LAYER` command. These current taps will be attached to the layer that is below the specified via or contact layer. Coordinate values are assumed to be in the GDSII coordinate units and will be interpreted according to the scale and precision settings contained in the GDSII file (or spice netlist with device XY location) that contains the cell. The value of the current drawn by each current tap will be 1/n of the value specified for the current region, where n is the number of taps within the current region. The second variation of the `REGION` command without the bounding box is valid only if there is only one current region. In this case, LibGen can automatically use the bounding box of the cell for the region.

Example:

```
UNIT CURRENT uA
UNIT CAPACITANCE fF
CELL an4_80
NET vcc
LAYER CONT
REGION 0 14000 28900 17000 3.0 5.0
NET gnd
LAYER CONT
REGION 1.5e-2mA
```

Cell Pin Net Map File

A file containing the mappings between a cell's LEF pin names and the internal names of the net to which those pins are attached. This file is used when the LEF file contains multiple unique power pins that are connected to the same power net.

© 2013-2023 All Rights Reserved.

Sample file name: `pin_map.txt`

Command: `set_advanced_pg_library_mode -cell_pinnet_map_file` *`filename`*

Format:

```
cellname net_name pin_name
```

Format Description:

`cellname` specifies the name of the cell.
`net_name` specifies the internal name of the net.
`pin_name` specifies the name of the LEF pin.

Example:

```
block_a VDD vdd_1
block_a VDD vdd_2
block_a VDD vdd_3
```

Cell Accura Data File

The Cell Accura Data file contains data about the electrical characteristics of a cell. It supports include statements for hierarchical descriptions. Specifying clock, instance power, and toggle count data is optional. Give only one specification for each type of cell. If you give more than one specification, LibGen issues a warning message and uses the last specification given.

Sample file name: `cell_char.file`

Command: `set_advanced_pg_library_mode -cell_accura_data_file` *`filename`*

Format:

```
CELL cellname
{
<FREQ=value>
<FA_VALUE=value>
<AVG_PWR=value>
<CLK_FILE=filename>
<FA_FILE=filename>
<AVG_PWR_FILE=filename>
}
```

Format Description:

`CELL` *cellname* specifies the name of the cell.

© 2013-2023 All Rights Reserved.

`FREQ=value` specifies the clock frequency for the cell used to compute the average currents. The default unit is hertz.

`CLK_FILE=filename` specifies the name of the file containing the clock nodes, subcircuit definitions, and subcircuit instance names and their frequencies. You can specify multiple file names.

`FA_VALUE=value` specifies the default nodal activity to be used instead of Accura. This value is used at each node unless it is overridden by a value in the file specified by `FA_FILE`. It is given as an activity factor. A value of 0.20 would be 20% of the default frequency.

`FA_FILE=filename` specifies the name of the file containing nodes and their activity. It is given as an activity factor. A value of 0.20 would be 20% of the default frequency. You can specify multiple file names.

`AVG_PWR=value` specifies the total average power dissipation of the cell connected to the voltage source. The default unit is watts.

`AVG_PWR_FILE=filename` specifies the name of the file containing the subcircuit definitions, subcircuit instance names, and average power consumption. You can specify multiple file names.

Example:

```
CELL XYZ
{
<FREQ=300>
<FA_VALUE=0.1>
<AVG_PWR=30>
<CLK_FILE=cell1.clk>
<FA_FILE=cell1.fa>
<AVG_PWR_FILE=cell1.power>
}
```

Clock File Format

The clock file has the format shown in the following example:

```
clk clk_internal 150M
```

`CLK` is a keyword.

- If you use the `CLK` keyword, the node name is treated as the root of a clock running at the given frequency.
- If you use the `INST` keyword, all input and output nodes--that is, node names on the instance card--are set to toggle at the specified frequency. The activity of the nodes internal to the subcircuit is computed by using Accura. If you give an optional activity value, Accura is disabled, and the activities of all the nodes are set to the specified value;

© 2013-2023 All Rights Reserved.

all nodes toggle at the time represented by the product of the specified frequency and the activity.

- If you use the `CELL` keyword, all instances of the cell are processed instead of a single instance.

`CLK` has precedence over `INST`, and `INST` overrides `CELL` information to resolve redundant specifications.

Frequency Activity File Format

The frequency activity file has the format shown in the following example. The column on the left shows the net names, and the column on the right shows the activity.

```
a100007    .134399e-01
a100014    .663391e-01
a100024    .705505e-01
......
```

Average Power File Format

The average power file has the format shown in the following example:

```
CELL C_MUX2_TG 2.0e-4
Xxev|xeu|xem|wpy|sdh 1.0e-3
```

The `CELL` keyword specifies that the name following it is a subcircuit definition and that all instances of the cell dissipate the same average power. If the file does not contain the `CELL` keyword, the name is an instance of a subcircuit dissipating the specified average power.

Cell Decap File

A file containing a list of cells and the decoupling capacitance value for each.

Sample file name: `decaps`

Command: `set_pg_library_mode -cell_decap_file` *`filename`*

Format:

```
UNIT unit_in_farads
CELL cell_name cap_value
NET net_name cap_value
```

Format Description:

`UNIT` is optional. If no `UNIT` statement is present at the beginning of the file, all capacitance values not followed by a unit specification are assumed to be in femtofarads.

© 2013-2023 All Rights Reserved.

`CELL` specifies the total decoupling capacitance for a cell. The `CELL` specification stays in effect until another `CELL` specification is encountered.

`NET` specifies the decoupling capacitance for a net.

Example:

```
UNIT     1.0e-15
CELL     Cell_A     100.00
NET      VDD         55.00
NET      VDD1        45.00
```

Cell Well Cap File

A file containing a list of cells and the well capacitance value for each.

Sample file name: wellcaps

Command: `set_advanced_pg_library_mode -well_cap_file` *`filename`*

Format:

```
UNIT unit_in_farads
CELL cell_name cap_value
```

Format Description:

`UNIT` is optional. If no `UNIT` statement is present at the beginning of the file, all capacitance values not followed by a unit specification are assumed to be in femtofarads.

`CELL` specifies the well capacitance value for a cell. The `CELL` specification stays in effect until another `CELL` specification is encountered.

Example:

```
UNIT     1.0e-15
CELL     Cell_A     100.00
```

Circuit Include File

Specifies the name of the Voltus library simulator input file. This file is used for current estimation and contains information about the cells in the design, additional MODEL cards for passive devices like resistors, and additional DC voltages.

Sample file name: `simulator.ckt`

Command: `set_advanced_pg_library_mode -circuit_include_file` *`filename`*

© 2013-2023 All Rights Reserved.

Format:

Spice format

Example:

```
V1 VDD1 0 1.2
V2 VDD2 0 1.8
.lib `log013.1' TT
.lib `log013.1' MID
.lib `log013.1' DIO
.include some_model
```

GDS Layermap File

A layer mapping file is required for the Rail Analysis library generation flow. The GDS layermap file contains the mapping information to map the GDS layer number to the layer names in the technology file. This file is required when library generation reads a GDS file. This file format is also used by the GDS2DEF utility (`convert_gds_to_def`).

Sample file name: `gds.layermap`

Command: `set_pg_library_mode -gds_layermap` *`filename`*

Format:

```
<LibGen Type>    <Library Layer Name>    <gds>    <GDSII Layer Number>    <GDS
Datatype> <PORT> <TEXT_ONLY|NO_TEXT>
```

Format Description:

`LibGen Type` specifies the type of library layer, for example, metal or polycide. The various layer types are:

- `diff` - diffusion layer
- `poly` - polycide layer
- `via` - via layer
- `metal` - metal layer
- `layer` - layer name used in the technology file

  Use this for layers that are not diffusion, metal, poly, or via layers. You can use it a number of times for each layer name to assign a different GDSII layer number to a cell library layer.

© 2013-2023 All Rights Reserved.

`Library Layer Name` specifies the name of the library layer to map to the GDSII layer number. The library layers are technology layers which can be dumped using the `Techgen -techinfo` command.

`Keyword` specifies the keyword for the GDS layer mapping.

`GDSII Layer Number` specifies the layer number of the GDSII layer.

`GDS Datatype` specifies the optional number of the subclass for the GDSII layer. If you do not specify a datatype, the software uses all datatypes for that GDSII layer. If GDS datatype is not defined, the software considers it as a regular layout routing layer where all the shapes will be extracted.

The GDS layer may have additional information/attribute assigned using the following keywords:

- `PORT`: Converts text labels in this layer to ports.
- `TEXT_ONLY`: Includes only the text in the named layer and ignores all geometries.
- `NO_TEXT`: Excludes text in the named layer.

Note: The layers are not required to be in any specific order.

Example:

```
via polycont gds 5 1
poly poly gds 4 1
metal met1 gds 6 1
metal met1 gds 103 1 PORT TEXT_ONLY
via via1 gds 7 1
metal met2 gds 8 1
metal met2 gds 104 1 PORT TEXT_ONLY
layer BOUNDARY gds 96 1
```

LEF-DEF Layermap File

A layer mapping file is required for the Rail Analysis library generation flow. The LEF-DEF layermap file contains the mapping information to map the layer names in the LEF file to the layer names in the technology file. This file is required when LibGen reads a LEF file.

Sample file name: `lefdef.layermap`

Command: `set_pg_library_mode -lef_layermap` *`filename`*

Format:

```
LibGen Type     Library Layer Name     Keyword     LEFDEF Layer Name
```


© 2013-2023 All Rights Reserved.

Format Description:

`LibGen Type` specifies the type of library layer, for example, metal or polycide.

`Library Layer Name` specifies the name of the library layer to map to the LEF/DEF layer.

`Keyword` specifies the keyword for the LEF-DEF layer mapping.

`LEFDEF Layer Name` specifies the name of the LEF/DEF layer to map to the library layer.

Example:

```
# LibGen library key lefdef
# type layername word layername
# ----------- --------------- ------ ---------
poly POLYCIDE lefdef PO
via CONT lefdef CO
metal METAL_1 lefdef M1
metal METAL_2 lefdef M2
   ...
```

Pad Voltage Source File

A file containing the locations of the voltage sources to be connected to the pad cells.

Sample file name: `voltsrc_533a.txt`

Command: `set_advanced_pg_library_mode -source_location_file` *`file`*

Format:

```
<MIN_SPACING spacing_value>
<DENSITY density_value>
CELL cell_name
<[PORT|DETAILED] {>
    NET net_name
    <[PORT|DETAILED] {>
        lef_layer_name <PORT|DETAILED> x y [x y]
        lef_layer_name <PORT|DETAILED> x y [x y]
    <}>
<}>
```

Format Description:

This source location file syntax includes the following parameters:

- `MIN_SPACING` *`spacing_value`*

  `DENSITY` *`density_value`*

  These values are used when you specify the source location as a rectangle. LibGen attempts to generate a grid of voltage sources with the dimensions `density_value x`

© 2013-2023 All Rights Reserved.

`density_value y` in the rectangle. However, if this grid would place the voltage sources closer together than the value set by `MIN_SPACING`, LibGen generates fewer voltage sources to avoid violating the `MIN_SPACING` setting.

Since these variables are global to the source location file, you can specify them anywhere in the file. If you specify them more than once, LibGen uses the last setting. The default value for `MIN_SPACING` is 1 micron. The default value for `DENSITY` is 10.

- `CELL` *`cell_name`*

  `NET` *`net_name`*

  These specifications apply to the voltage source locations that follow until LibGen encounters a new `CELL` or `NET` line. You can specify voltage sources as single points or as rectangles. If you specify them as rectangles, LibGen generates a grid of voltage sources inside these rectangles.

- `PORT|DETAILED`

  The source location file can specify voltage sources to be used with port or detailed views. If you do not use the `PORT` and `DETAILED` keywords, LibGen uses the voltage source data for detailed views. LibGen issues a warning instructing you to update the format of your source location file to include the `PORT` and `DETAILED` keywords.

  You can specify the `PORT` and `DETAILED` keywords on the cell level, on the net level, or on an individual voltage source location definition. LibGen uses specifications at the cell level unless they are overridden at the net level. It uses specifications at the net level unless they are overridden by the individual voltage source specifications.

- *`lef_layer_name`* `<PORT|DETAILED>` *`x y`* `[`*`x y`*`]`

  Lines beginning with a LEF layer name specify the actual location of the voltage sources. Usually you can specify this location as a single point by providing one set of values for x and y. These x and y coordinates are specified in microns and use the same coordinate system as that used for LEF ports in the LEF file.

  Alternatively, you can specify the voltage source locations as a rectangle by giving two sets of *`x`* and *`y`* locations. In this case, LibGen attempts to generate a 10-by-10 grid of voltage sources inside this area. From a rectangle specification, it does not create voltage sources closer than 1 micron apart, so if the rectangle is less than about 10 microns on a side, it generates fewer voltage sources.

  In the source location file, the location of the voltage sources is specified by using a LEF layer name and location in LEF coordinate space. However, because of the way LibGen uses this data internally, no LEF geometries are actually required to exist at the location where the voltage sources are to be placed. LibGen uses this data internally with the layout of the block (that is, the GDS data) during detailed power-grid view generation, so

© 2013-2023 All Rights Reserved.

it only requires that there be layout (GDS) geometries at the location of the voltage sources. Similarly, the LEF layer name used in the source location file does not actually have to be included in any of the input LEF files; it must be defined only in the layer map used by LibGen.

Example:

- Following is an example of the contents of the voltage source location file:

```
CELL an2_120
NET vcc
  MET1 11.0 16.0
    MET1 0.0 15.0 22.1 17.5
NET gnd
   MET1 11.0 1.0
   MET1 0 -0.5 22.1 2.0

CELL xor
NET vcc
   MET1 10.0 5.0
NET gnd
  MET1 10.0 1.0
```

- Following is an example file that uses only cell-level specifications to define voltage sources for the port view of the NAND1 cell:

```
CELL NAND1
PORT
  NET VDD
    MET1 5 95
    MET2 5 95
  NET VSS
    MET1 5 5
    MET2 5 5
```

- Following is an example file that uses net-level specifications to define voltage sources for both the port and detailed views of the NAND1 cell:

```
CELL NAND1
  NET VDD
    PORT
    MET1 5 95
    MET1 DETAILED 5 92
    MET1 DETAILED 5 98
  NET VSS
    PORT
    MET1 5 5
    MET1 DETAILED 5 2
    MET1 DETAILED5 8
```

An alternate way to achieve the same results as the last example is by mixing port- and net- level specifications, or port-level and individual source specifications:

```
CELL NAND1
  PORT
    NET VDD
      MET1 5 95
```


© 2013-2023 All Rights Reserved.

```
      DETAILED
        MET1 5 92
        MET1 5 98
    NET VSS
      MET1 5 5
      MET1 5 2
      MET1 5 8
```

- Following is an example file using `MIN_SPACING` and `DENSITY` to create a 10 x 10 grid of voltage sources on the VSS net in the detailed view of the NAND1 cell:

```
DENSITY 10
MIN_SPACING 1
CELL NAND1
  DETAILED
  NET VSS
```

- Following is an example using `DENSITY`:

```
DENSITY 10
CELL PVDD_33
NET VDD
M6 DETAILED 15 83.23
M6 PORT 15 83.21
```

Port File

A file containing the details about the label of a switched rail. It contains all information on ports that you want to create and load.

Sample file name: `fire_ports`

Command: `set_advanced_pg_library_mode -add_port_labels` *`filename`*

Format:

The format of the file is as follows:

```
MACRO <cellname>
PIN <pinname> <direction> <x-coord in microns> <y-coord in microns> <xtc layer>
```

Format Description:

`MACRO` The `MACRO` statement is followed by a cell name.

`PIN` The `PIN` statement is followed by a pin name and a direction. These are the valid directions:

- B--Bidirectional
- I--Input
- O--Output

© 2013-2023 All Rights Reserved.

Example:

```
MACRO foo
PIN VSS I 4.95 -392.91 METAL_1
PIN VDD O 4.21 421.1 METAL_2
.
```

Trigger File

A file containing the trigger conditions for each cell. The trigger file contains information on the input vectors and waveform triggering input pin information. The trigger condition is used to identify the functional mode (read/write/idle) of the custom macro in the Power and Rail Analysis flow.

Sample file name: `trig.cell`

Command: `set_pg_library_mode -current_distribution {dynamic_simulation trigger_file}`

Format:

The format of the trigger file is:

Trigger File Format	Description
#Method of current characterization	
`CURRENT_CHARACTERIZATION_METHOD <method>` `<SIMULATION/DOTLIB/PWL/FSDB/DATASHEET/SPICE_DECK_BASED>`	■ `SIMULATION` -> Vector-based flow ■ `DOTLIB` -> Liberty based Flow ■ `PWL` - User-defined PWL flow ■ `FSDB` - Spice total current output captured in the FSDB format ■ `DATASHEET` - Datasheet details of any memory ■ `SPICE_DECK_BASED` - User-specified Spice deck flow
#Memory cell name	
`CELL <cellname1>`	Name of the memory cell
#Mode and conditional details for trigger	

© 2013-2023 All Rights Reserved.

MODE_NAME <*mode1*> CONDITIONAL_INPUT <*conditional input statement for cellname1*>	Mode name and conditional input statement for the cell. Conditional input of current for the given mode.
CONDITIONAL_PIN <*pin_name*> (rise|fall|both)	Conditional pin statement for the cell. Signal pin for which the tool should rise or fall.
CONDITIONAL_STIMULUS_FILE <*usim1.txt*>	Simulation vector detail in the given mode.
USER_PWL_FILE <*pwl1.txt*>	User-specified PWL file that contains information about the PWL waveforms (power/ground currents)
FSDB_START_TIME <*time*> FSDB_END_TIME <*time*>	FSDB Simulation based mode details
DATASHEET_PARAMETER < {*pg_pin1 Capacitance Leakage*} {*pg_pin2 Capacitance Leakage*}>	Datasheet-based mode details
END_MODE	End mode section
#Datasheet clock required parameters	
DATASHEET_INPUT_SLEW <*time*> DATASHEET_OUTPUT_SLEW <*time*> DATASHEET_DELAY <*time*> DATASHEET_PARAMETER < {*power_pin cap leakage_pwr*} >	■ DATASHEET_INPUT_SLEW - Clock input slew of the conditional pin ■ DATASHEET_OUTPUT_SLEW - Rising slew of the output pin Q ■ DATASHEET_DELAY - Mode delay as the read_access and write_access time ■ DATASHEET_PARAMETER - Datasheet parameters include power pin voltage, load capacitance, and leakage power. Load capacitance is mode dependent.
#FSDB required parameters	
FSDB_PATH <*path*> FSDB_NET_NAME <net> [<*generic_name*>] FSDB_START_TIME <*time*> FSDB_END_TIME <*time*> FSDB_HIER_CHAR <*hier_char, for hier DB*>	■ FSDB file path ■ Power net name and its FSDB probe name ■ FSDB start and end time ■ Hierarchical separator for hierarchical database. The default separator is "**.**" Single FSDB file having total current for ALL power/ground pins with all mode current in it

© 2013-2023 All Rights Reserved.

#PWL required parameters	
`PWL_START_TIME <time>` `PWL_END_TIME <time>`	Time ranges for different modes in the PWL-based current characterization flow. These keywords allow you to use a single PWL input file for multiple modes and specify different time ranges for these modes in which the dynamic PGV is being created.
#User-specified Spice deck required parameters	

`DECK_FILE <filename>`
`SIM_OPTIONS <option_name>`
`FSDB_FILE <filename>`
	■ Specify a single deck file for all modes, or a unique deck file for each mode.
	■ User-specified Spectre simulator options. The default value of this keyword is `+aps`.
	■ Voltus executes the Spice deck and reads the current waveform generated by simulation. After simulation of the deck file, an FSDB file is generated. You must specify the FSDB file name with the `FSDB_FILE` keyword.

#<Other mode independent parameters>	

© 2013-2023 All Rights Reserved.

`SIMULATOR <type of simulator>`
`DOTLIB_LIST {a list of memory dotlibs files}`
`MODE_COUNT`
`MODE_CONFIG_FILE <file_path>`
`SIM_START_TIME <time>`
`SIM_STOP_TIME <time>`
`SIM_STEP_SIZE <step_size>`
`STIMULUS_FILE <file_path>`
`INCLUDE_FILE <file_path>`
	■ `SIMULATOR` - Specifies the simulator to be used for characterizing current waveforms for Macro EM views. You can specify the following simulators: Accelerated Parallel Simulator (APS), SPECTRE, and UltraSim. APS is the default simulator.
	■ `DOTLIB_LIST` - Specifies a list of memory dotlibs.
	■ `MODE_COUNT` - Specifies the maximum number of modes to be generated. Valid values are either a positive number or `all'. `all' means stimulus for all power groups are to be generated. The default value is 4. If you set this parameter to all, it may result in high runtime as it may generate too many modes.
	■ `MODE_CONFIG_FILE` - Specifies the file containing extra information for memory cell. This file contains the following information - signal slew, signal active value and memory write information.
	■ `SIM_START_TIME` - Specifies the simulation start time. The default unit is in nanoseconds (ns).
	■ `SIM_STOP_TIME` - Specifies the simulation stop time. The default unit is in nanoseconds (ns).

© 2013-2023 All Rights Reserved.

	`SIM_STEP_SIZE` - Specifies the step size for power-grid analysis. The default value of step size is 20 picoseconds (ps). This parameter supports other units such as, ns and fs. This parameter supports unit-prefixes (pico, femto, nano) combined with units of measure (ps, fs, ns), but does not support units itself ("second", "Ohm", "Farad") without unit-prefix. `STIMULUS_FILE` - Specifies the stimulus file containing the trigger information for all input ports. `INCLUDE_FILE` - Specifies the name of the simulation include file.
#End cell section	
`END`	End cell section

Trigger File Format - for simulation vector-based files

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD SIMULATION
SIMULATOR APS
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
CONDITIONAL_STIMULUS_FILE <usim1.txt>
END_MODE
END
```

where,

"`!`" denotes low state of the signal

"`&`" denotes boolean AND operation

"`|`" denotes boolean OR operation

For each mode, you need to specify separate stimulus files (for example `usim1.txt` and `usim2.txt`). The stimulus files contain the PWL definitions that will be used by UltraSim for simulating the specific mode.

Example

© 2013-2023 All Rights Reserved.

```
CURRENT_CHARACTERIZATION_METHOD SIMULATION
SIMULATOR APS
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = (! ATPGM & EZ &! SCAN &! TEZ & TM &! TPAGE &! TWTZ &! TWZ)
CONDITIONAL_PIN CLK {RISE}
CONDITIONAL_STIMULUS_FILE usim.txt
END_MODE
END
```

Trigger File Format - for user-defined PWLs

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
USER_PWL_FILE pwl1.txt
END_MODE
END
```

The format of the PWL file is:

```
UNIT CURRENT <current unit> (default is A, Supported values are A, mA, uA & nA)
UNIT TIME <time unit> (default is s, Supported values are s, ms, us, ns, ps)
NET <net_name1>
REGION {t1 v1 t2 v2 t3 v3.. }
NET <net_name2>
REGION {t4 v4 t5 v25 t6 v6.. }
```

Here, the time and current units are as defined using `UNIT CURRENT` and `UNIT TIME` syntax. `NET` specifies that the `REGION` command following it applies to the specified net name. This stays in effect until another `NET` or `CELL` specification is encountered. `REGION` specifies the power up piecewise linear (PWL) stimulus in (time, voltage) pair format.

Example

```
CURRENT_CHARACTERIZATION_METHOD PWL
CELL mem1
MODE_NAME  READ
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ & !TWZ )
CONDITIONAL_PIN  CLK { RISE }
USER_PWL_FILE pwl1.txt
END_MODE
END
```

Trigger File Format - FSDB

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL <cellname1>
MODE_NAME <name1>
```


© 2013-2023 All Rights Reserved.

```
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
FDSB_START_TIME <time>
FSDB_END_TIME <time>
END_MODE
FSDB_PATH <path>
FSDB_NET_NAME <net> [<generic_name>]
END
```

Example

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ & !TWZ )
CONDITIONAL_PIN CLK { RISE }
FDSB_START_TIME 0ns
FSDB_END_TIME 10ns
END_MODE
FSDB_PATH “new.fsdb"
FSDB_NET_NAME "i1(vvddpr)" "VDDPR"
FSDB_NET_NAME "i1(vvddar)" "VDDAR"
FSDB_NET_NAME "i1(vvddp)" "VDDP"
FSDB_NET_NAME "i1(vvss)" "VSS"
END
```

You can read multiple FSDBs for different modes, and specify different start and end times for each mode. The following example shows that 2 FSDB files are read, wherein `read.fsdb` is read for the `READ` mode, and `write.fsdb` is read for the `WRITE` mode:

```
CURRENT_CHARACTERIZATION_METHOD FSDB
CELL BLULP00256032040
MODE_NAME READ
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ & !TWZ )
CONDITIONAL_PIN CLK { RISE }
FDSB_START_TIME 0ns
FSDB_END_TIME    10ns
FSDB_PATH "read.fsdb"
FSDB_NET_NAME "i1(vvddpr)"  "VDDPR"
FSDB_NET_NAME "i1(vvddar)"  "VDDAR"
END MODE

MODE_NAME WRITE
CONDITIONAL_INPUT = ( !ATPGM & EZ & !SCAN & !TEZ & TM & !TPAGE & !TWTZ & !TWZ )
CONDITIONAL_PIN CLK { RISE }
FDSB_START_TIME 0ns
FSDB_END_TIME    15ns
FSDB_PATH "write.fsdb"
FSDB_NET_NAME "i1(vvddpr)"  "VDDPR"
FSDB_NET_NAME "i1(vvddar)"  "VDDAR"
END MODE
END
```

Trigger File Format - Datasheet

The file format of the trigger file is:

© 2013-2023 All Rights Reserved.

```
CURRENT_CHARACTERIZATION_METHOD DATASHEET
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
DATASHEET_DELAY <time>
DATASHEET_PARAMETER < {net cap pwr} >
END_MODE
DATASHEET_INPUT_SLEW <time>
DATASHEET_OUTPUT_SLEW <time>
END
```

Example

```
CURRENT_CHARACTERIZATION_METHOD DATASHEET
CELL mem1
MODE_NAME READ
CONDITIONAL_INPUT = ( WZ & EZ )
CONDITIONAL_PIN  CLK { RISE }
DATASHEET_DELAY  7.6913ns
DATASHEET_PARAMETER {{ VDDAR 0.1715pf 2.496e-05mW }{ VDDPR 18.8985pf 3.63e-06mW }{
VBBNW 0.1780pf 2.601e-06mW  }}
END_MODE

MODE_NAME WRITE
CONDITIONAL_INPUT = ( !WZ & !WTZ & EZ  )
CONDITIONAL_PIN  CLK { RISE }
DATASHEET_DELAY  8.216ns
DATASHEET_PARAMETER {{ VDDAR 0.1715pf  1.355e-06mW }{ VDDPR  0.1715pf 1.81e-05mW }
{ VBBNW 0.1715pf 2.614e-06mW }}
END_MODE

MODE_NAME STANDBY
CONDITIONAL_INPUT = ( !EZ )
CONDITIONAL_PIN  CLK { RISE }
DATASHEET_DELAY  0.6913ns
DATASHEET_PARAMETER {{ VDDAR 0.1715pf 2.902e-05mW }{ VDDPR  0.2024pf 1.81e-05mW }
{ VBBNW 0.1723pf 6.92e-07mW }}
END_MODE

DATASHEET_INPUT_SLEW 0.6ns
DATASHEET_OUTPUT_SLEW 0.3682ns
END
```

Trigger File Format - for user-specified Spice deck

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD SPICE_DECK_BASED
CELL <cellname1>
MODE_NAME <name1>
CONDITIONAL_INPUT <conditional input statement for cellname1>
CONDITIONAL_PIN <pin_name> (rise|fall|both)
FDSB_START_TIME <time>
FSDB_END_TIME <time>
END_MODE
DECK_FILE <filename>
FSDB_NET_NAME <net> [<generic_name>]
FSDB_FILE <filename>
```


© 2013-2023 All Rights Reserved.

```
SIM_OPTIONS +aps
END
```

Example

```
CURRENT_CHARACTERIZATION_METHOD SPICE_DECK_BASED
CELL cellA
MODE READ
CONDITIONAL_INPUT = ( !CEN & !WEN )
CONDITIONAL_PIN CLK { RISE }
FSDB_START_TIME 0ns
FSDB_END_TIME  0.5ns
END_MODE

MODE WRITE
CONDITIONAL_INPUT = ( !CEN & !WEN )
CONDITIONAL_PIN CLK { RISE }
FSDB_START_TIME 0.6ns
FSDB_END_TIME  2ns
END_MODE

DECK_FILE deck.cir
FSDB_NET_NAME i1(vvdd)  VDD
FSDB_NET_NAME i1(vvss) VSS
FSDB_FILE usimCktFile.fsdb.gz
SIM_OPTIONS +aps

END
```

Trigger File Format - Dotlib

For dotlib-based detailed dynamic view characterization, the trigger and stimulus files are automatically generated from an existing dotlib. This flow is applicable only for memory cells. Voltus generates stimuli for the worst power groups (rise_power, fall_power or power) in the specified dotlib. It reads only the internal power groups having the "`when`" condition and constructs stimulus vectors. Each of the generated stimulus belongs to one of the following groups:

- pin power based (this group is associated with input pin)
- arc-based (this group is associated with output pin)

Note: arc-based group without `related_pin` will be ignored

The file format of the trigger file is:

```
CURRENT_CHARACTERIZATION_METHOD DOTLIB
DOTLIB_LIST {a list of memory dotlibs files}
MODE_COUNT <specifies the maximum number of modes to be generated. Valid values are
either a positive number or `all'. The default value is 4>
MODE_CONFIG_FILE <file_path>
SIM_STOP_TIME <time>
SIM_STEP_SIZE <step_size>
```


© 2013-2023 All Rights Reserved.

If the dotlib contains arc-based power groups, you must specify `MODE_CONFIG_FILE` and include the `memory_write` condition in the memory description file. Otherwise, arc-based power groups will be skipped. The memory description file contains the following information:

- Signals slew: specify slew for signals. These slew values will be used in case rise or fall is to be applied on signal. The slew value specified using this construct takes priority over the slew value saved for the group. The default slew unit in the `slews` construct is ns. The syntax is:

```
slews {
<signal_1> <slew_value_1>
<signal_2> <slew_value_2>
....
}
```

- Signals: specify signals' active values irrespective of `memory_write`. The syntax is:

```
signals {
<sig_1_name> <active_value>
<sig_2_name> <active_value>
....
    }
where <active value> can be H or L
```

- Memory write information: specify information required to write data to memory through `memory_write` block.

```
memory_write {
clock <clock_name> <type>, where <type> can be R, F, H or L
data_in <data_bus_name>
address <address bus name>
signal <sig_1 name> <active_value>, where <active value> can be H or L
signal <sig_2 name> <active_value>, where <active value> can be H or L
....
}
```

A sample memory description file is given below:

```
cell memory1
{
memory_write
{
  address A
  data_in D
  clock CLK R
  signal WEN L
  signal CEN L
}
    slews

{
  CLK 0.1
}
  signals

{
   D[1] H
```


© 2013-2023 All Rights Reserved.

```
}
}
```

Example

```
CURRENT_CHARACTERIZATION_METHOD DOTLIB
DOTLIB_LIST {}
MODE_COUNT all
SIM_STOP_TIME 10ns
SIM_STEP_SIZE 100ps
```

Power Analysis

This section lists the Power Analysis file formats in alphabetical order.

Instance ASCII Power File

A file containing the peak or average power-consumption values for each cell type. The values are specified in watts. It enables you to scale the cell peak power values up or down when you want to limit the peak power that an instance can draw at any time. This file is optional for both static and dynamic analysis.

Sample file name: `pg22e2_cellpwr.txt`

Command: `set_power_data -format ascii vdd_vdd.ptipeak`

Format:

```
<instancename1> <power_value1> <power_pin_name1>
<instancename2> <power_value2> <power_pin_name2>
.
.
```

Cell names cannot contain wildcards.

Format Description:

`instancename` specifies the instance to apply the power value to.

`power_value` specifies the power value in watts.

`power_pin_name` specifies the power pin name of the instance.

Example:

```
INVX1 1e-06 vdd
MEM25 2.5e-03 vdd1
NFD2QX4 1.5e-05 vdd3
```


© 2013-2023 All Rights Reserved.

Instance Temperature File

A temperature file is used to support thermal aware leakage. You can use this file to perform temperate aware leakage and internal power calculation. This file is generated by the thermal engine.

Sample file name: `temperaturefile`

Command: `set_inst_temperature_file` *`filename`*

Format:

```
Default_temperature <temperature>
- <instance_name> temperature;
- <instance_name> temperature;
END ;
```

Format Description:

`Default_temperature` specifies the default temperature, in degrees.

`instance_name` specifies the instance to apply the temperature to.

`temperature` is the temperature of the specified instance.

Example:

```
Default_temperature 25
- bridge/i_12866 31.351 ;
- bridge/i_12865 31.343 ;
- bridge/i_36209 31.27 ;
- bridge/dataf_cmdt_3_reg 31.348 ;
- bridge/dataf_cmdt_2_reg 31.343 ;
- bridge/dataf_cmdt_4_reg 31.335 ;
- bridge/extern_m_in_78to74_feedback_reg_4 30.023 ;
- bridge/i_35699 30.954 ;
- bridge/extern_s2_out_reg_75 30.956 ;
END ;
```

TCF File

A toggle count format (TCF) file contains switching activity information that is needed for power optimization and power analysis in low-power synthesis. The switching activity includes the toggle count information and the probability of the net or pin being in the logic 1 state.

A TCF file contains the switching activity information in a hierarchical representation. In a hierarchical representation, the net and pin names are specified within the instance scope.

© 2013-2023 All Rights Reserved.

This file is used to provide activity information during power calculation and can be read in using the `read_activity_file` command.

Sample file name: `design.tcf`

Command: `read_activity_file -format TCF` *`design.tcf`*

Format:

```
tcffile () {
    tcfversion : " ";
    generator : " ";
    genversion : " ";
    date : " ";
    duration : " ";
    unit : " ";
    instance () {
        pin () {
            " " : " ";
            ...
            }
        or
        net () {
            " " : " ";
            ...
            }
        }
}
```

Format Description:

`tcfversion` identifies the TCF version.

`generator` identifies the utility that generated this TCF file. Possible values are "`BGPower Verilog PLI`" and "`BGPower Verilog VHDL PLI`." This parameter is optional.

`genversion` identifies the version of the utility that generated the TCF file.

`date` specifies the date and time when the data in the TCF file was created.

`duration` specifies the simulation capture period.

`unit` specifies the time unit used for the simulation duration.

`instance` is identified through a string. For each `pin` and `net`, the TCF file lists two strings:

- The first string represents the name (hierarchical name or name with respect to the instance).
- The second string contains a real number representing the probability of that net or pin being logic 1, and an integer representing the toggle count.

© 2013-2023 All Rights Reserved.

Note: The following special characters are supported in the TCF file: + = . , / [ ] ! \ $ * @ ~

Example:

Following is an example of a TCF file with hierarchical net names:

```
tcffile () {
  tcfversion : "1.0";
  generator : "Tempus Timing Signoff Solution 15.12-s063_1"
  duration : "1.0";
  unit : "s";
  instance () {
    net () {
      "net_1" : "0.500000 400000000";
      "net_2" : "0.500000 400000000";
      "net_3" : "0.500000 400000000";
      "net_4" : "0.500000 400000000";
      "net_5" : "0.500000 400000000";

...
      "I220/net_10" : "0.499900 800000000";
      "I221/net_11" : "0.702900 897744000";
      "I113/net_12" : "0.990000 121991952";
    }
  }
}
```

TWF File

The timing windows file (TWF) contains timing window and slew data from timing analysis of the design that is used to constrain noise analysis for more realistic results based on the timing dependency of switching signals.

TWF is required when you perform a vectorless dynamic power consumption calculation. For each net or pin, TWF contains the ranges of signal arrival times and transition times derived from the static timing analysis output by Tempus. It compresses all vectors into one result. This file is in ASCII format.

Sample file name: `dtmf.twf`

Command: This file can be generated using the Tempus command `write_twf` and can be read in using the `read_twf` command.

Format:

```
timing_windows_file
    ::= ( TIMING_WINDOWS header body_statement+ )
header
    ::= ( HEADER version header_info+ )
version
```


© 2013-2023 All Rights Reserved.

```
    ::= ( VERSION QSTRING )
header_info
    ::= design_name
    ||= date
    ||= program
    ||= delimiters
    ||= time_scale
    ||= cap_scale
    ||= res_scale
    ||= voltage_scale
    ||= voltage_threshold
    ||= lifetime
    ||= voltage
    ||= temperature
    ||= operating_conditions
    ||= default_input_slew
design_name
    ::= ( DESIGN QSTRING )
date
    ::= ( DATE QSTRING )
program
    ::= ( PROGRAM program_name program_version program_company )
program_name
    ::= QSTRING
program_version
    ::= QSTRING
program_company
    ::= QSTRING
delimiters
    ::= ( DELIMITERS QSTRING )
time_scale
    ::= ( TIME_SCALE multiplier )
cap_scale
    ::= ( CAPACITANCE_SCALE multiplier )
res_scale
    ::= ( RES_SCALE multiplier )
voltage_scale
    ::= ( VOLTAGE_SCALE multiplier )
voltage_threshold
    ::= ( VOLTAGE_THRESHOLD min_thres max_thres )
lifetime
```


© 2013-2023 All Rights Reserved.

```
    ::= ( LIFETIME NUMBER )
process
    ::= ( PROCESS single_or_triple_number )
voltage
    ::= ( VOLTAGE single_or_triple_number )
temperature
    ::= ( TEMPERATURE single_or_triple_number )
operating_conditions
    ::= ( OPERATING_CONDITIONS single_or_triple_a_string )
default_input_slew
    ::= ( DEFAULT_INPUT_SLEW a_min_rise a_min_fall a_max_rise a_max_fall )
multiplier
    ::= NUMBER
body_statement
    ::= ext_ctlf_files
    ||= waveform_entry+
    ||= clock_entry+
    ||= clock_group_entry+
    ||= driver_cell_entry+
    ||= driver_strength_entry+
    ||= external_load_entry+
    ||= constant_entry+
    ||= input_slew_entry+
    ||= caused_by_net+
    ||= caused_by_null+
ext_ctlf_files
    ::= ( EXTENSION "CTLF_FILES" (file_name+))
waveform_entry
    ::= ( WAVEFORM "waveform_name" period
    ( POSEDGE edge_position ) ( NEGEDGE edge_position ) )
    edge_position
    ::= NUMBER
clock_entry
    ::= ( CLOCK "waveform_name" port )
clock_group_entry
    ::= ( CLOCK_GROUP "group_name" "waveform_name"+ )
driver_cell_entry
    ::= ( DRIVER_CELL
    ( ( CELL_TYPE "cell_name" library
    ( pos_neg_edge src_pin ) sink_pin )
    ( su_hd? pos_neg_edge? ( INPUT_SLEW slew ) )
```


© 2013-2023 All Rights Reserved.

```
    port )
slew
    ::= NUMBER
driver_strength_entry
    ::= ( DRIVER_STRENGTH driver_strength_val port )
driver_strength_val
    ::= opt_qualifier? resistance
    ||= min_rise min_fall max_rise max_fall
opt_qualifier
    ::= ( su_hd? pos_neg_edge? )
su_hd
    ::= SETUP
    ||= HOLD
pos_neg_edge
    ::= POSEDGE
    ||= NEGEDGE
external_load_entry
    ::= ( EXTERNAL_LOAD single_or_triple_a_number port )
constant_entry
    ::= ( constant_kw net_pin "port" const_val )
    ::= ( constant_kw net_pin "port" const_val rise_k fall_k)
input_slew_entry
    ::= ( INPUT_SLEW a_min_rise a_min_fall a_max_rise a_max_fall port)
caused_by_net
    ::= ( CAUSED_BY "port" rise_fall window_statement+)
caused_by_null
    ::= ( CAUSED_BY NULL window_statement+)
window_statement
    ::= ( net_pin "port"
rise_at rise_time rise_k fall_at fall_time fall_k )
    ||=
    ( net_pin "port" rise_at rise_time rise_k fall_at fall_time fall_k
    clock_or_data)
    ||=
    ( net_pin "port" rise_at rise_time rise_k rise_slack fall_at fall_time
    fall_k fall_slack)
    ||=
    ( net_pin "port" rise_at rise_time rise_k rise_slack fall_at fall_time
    fall_k fall_slack clock_or_data)
constant_kw
    ::= CONSTANT
```


© 2013-2023 All Rights Reserved.

```
    ||= USER_CONSTANT
net_pin
    ::= NET
    ||= PIN
const_val
    ::= 1
    ||= 0
rise_fall
    ::= RISE
    ||= FALL
file_name
    ::= STRING
single_or_triple_number
    ::= NUMBER
    ||= min typ max
single_or_triple_a_string
    ::= STRING
    ||= min_a_string typ_a_string max_a_string
single_or_triple_a_number
    ::= a_number
    ||= a_min a_typ a_max
a_number
    ::= *
    ||= NUMBER
rise_at
    ::= triplet
rise_time
    ::= triplet
rise_k
    ::= triplet
rise_slack
    ::= triplet
fall_at
    ::= triplet
fall_time
    ::= triplet
fall_k
    ::= triplet
fall_slack
    ::= triplet
clock_or_data
```


© 2013-2023 All Rights Reserved.

```
    ::= C
    ||= D
    ||= U
    ||= Z
triplet
    ::= *
    ||= NUMBER
    ||= NUMBER:NUMBER
    ||= NUMBER:NUMBER:NUMBER
a_min
    ::= a_number
a_typ
    ::= a_number
a_max
    ::= a_number
a_min_rise
    ::= a_number
a_min_fall
    ::= a_number
a_max_rise
    ::= a_number
a_max_fall
    ::= a_number
min_thres
    ::= NUMBER
max_thres
    ::= NUMBER
min
    ::= NUMBER
typ
    ::= NUMBER
max
    ::= NUMBER
min_rise
    ::= NUMBER
min_fall
    ::= NUMBER
max_rise
    ::= NUMBER
max_fall
    ::= NUMBER
```


© 2013-2023 All Rights Reserved.

```
period
    ::= NUMBER
resistance
    ::= NUMBER
min_a_string
    ::= a_string
typ_a_string
    ::= a_string
max_a_string
    ::= a_string
a_string
    ::= *
    ||= STRING
waveform_name
    ::= STRING
port
    ::= STRING
group_name
    ::= STRING
cell_name
    ::= STRING
library
    ::= STRING
src_pin
    ::= STRING
sink_pin
    ::= NUMBER
```

Format Description:

Header Section

The header section of a TWF file contains information relevant to the entire file, including the design name, the tool used to generate the TWF, and scaling factors for the values in the file.

The design name, date, and program entries are for documentation purposes and do not affect the meaning of the data in the rest of the file. The version, delimiters, and scaling factors *do* affect how the data in the file is interpreted.

© 2013-2023 All Rights Reserved.

Syntax

```
header
     ::= ( HEADER version header_info+ )

header_info
     ::= design_name
     ||= date
     ||= program
     ||= delimiters
     ||= time_scale
     ||= cap_scale
     ||= res_scale
     ||= voltage_scale
     ||= voltage_threshold
     ||= lifetime
     ||= voltage
     ||= temperature
     ||= operating_conditions
     ||= default_input_slew
```

TWF Version

The `VERSION` construct identifies the version of the TWF specification to which the file conforms.

Syntax

```
version
     ::= ( VERSION QSTRING )
```

Example

```
(VERSION "0.1")
```

Design Name

The `DESIGN` construct specifies the name of the design to which the TWF file applies.

Syntax

```
design_name
     ::= ( DESIGN QSTRING )
```

Example

```
(DESIGN "D97000000")
```


© 2013-2023 All Rights Reserved.

Date

The DATE construct indicates the date and time when the data was generated.

Syntax

```
date
     ::= ( DATE QSTRING )
```

Example

```
(DATE "Tue Jan 22 18:02:05 2002")
```

Program

The PROGRAM construct indicates the name of the program that created or last modified the TWF file, the version number of that program, and the company that produced the program.

Syntax

```
program
     ::= ( PROGRAM program_name program_version program_company )

program_name
     ::= QSTRING

program_version
     ::= QSTRING

program_company
     ::= QSTRING
```

Example

```
(PROGRAM "Pearl" "5.1-b047" "Cadence")
```

Delimiters

The DELIMITERS construct specifies the characters that are used as delimiters in design names for hierarchy delimiter character, left index character and right index character.

Syntax

```
delimiters
     ::= ( DELIMITERS QSTRING )
```


© 2013-2023 All Rights Reserved.

Example

```
(DELIMITERS "/[]")
```

Scaling Factors

Scaling factor constructs specify the multiplier to use to scale the values for the specified physical property.

Syntax

```
time_scale
     ::= ( TIME_SCALE multiplier )

cap_scale
     ::= ( CAP_SCALE multiplier )

res_scale
     ::= ( RESISTANCE_SCALE multiplier )

voltage_scale
     ::= ( VOLTAGE_SCALE multiplier )

multiplier
     ::= NUMBER
```

The default time scale is 1 second. If `TIME_SCALE` is specified, all delay numbers in the TWF are multiplied by the specified value. For example, `1.0E-12` corresponds to delay values in picoseconds.

The default capacitance scale is 1 Farad. If `CAP_SCALE` is specified, all capacitance numbers in the TWF are multiplied by the specified value. For example, `1.0E-12` corresponds to capacitance values in picofarads.

The default resistance scale is 1 ohm. If `RES_SCALE` is specified, all resistance numbers in the TWF are multiplied by the specified value. For example, `1.0E-3` corresponds to resistance values in milli-ohms.

The default voltage scale is 1 volt. If `VOLTAGE_SCALE` is specified, all voltage numbers in the TWF are multiplied by the specified value. For example, `1.0E-3` corresponds to voltage values in millivolts.

Voltage Threshold

The `VOLTAGE_THRESHOLD` construct specifies the two measurement points on a waveform that were used in calculating the slews in the TWF file. These measurement points are defined as a percentage of the change in voltage from the start of the transition to the end of the transition.

© 2013-2023 All Rights Reserved.

Syntax

```
voltage_threshold
     ::= ( VOLTAGE_THRESHOLD min_thres max_thres )

min_thres
     ::= NUMBER

max_thres
     ::= NUMBER
```

Example

```
(VOLTAGE_THRESHOLD 20 80)
```

Lifetime

The `LIFETIME` construct specifies the required operating lifetime for the design in years. This value is not scaled by the `TIME_SCALE` statement.

Syntax

```
lifetime
     ::= ( LIFETIME NUMBER )
```

Example

```
(LIFETIME 3)
```

Operating Conditions

The operating conditions constructs specify environment corners and related process, voltage, and temperature derating points.

Syntax

```
operating_conditions
     ::= ( OPERATING_CONDITIONS single_or_triple_a_string )

process
     ::= ( PROCESS single_or_triple_number )

voltage
     ::= ( VOLTAGE single_or_triple_number )

temperature
     ::= ( TEMPERATURE single_or_triple_number )
```


© 2013-2023 All Rights Reserved.

```
single_or_triple_a_string
     ::= typ_a_string
     ||= min_a_string typ_a_string max_a_string

min_a_string
     ::= a_string

typ_a_string
     ::= a_string

max_a_string
     ::= a_string

single_or_triple_number
     ::= NUMBER
     ||= min typ max

min
     ::= NUMBER

typ
     ::= NUMBER

max
     ::= NUMBER
```

One string (representing typical) or three strings (representing minimum, typical, and maximum) or asterisks (representing placeholders) can be specified for `OPERATING CONDITIONS` *`a_string`* values .

One or three values can be specified for the *`single_or_triple_number`* parameters.

- When one value is specified, it applies to the typical value.
- When three values are specified, the first represents the minimum value, the second represents the typical value, and the third represents the maximum value.

Example

```
(OPERATING_CONDITIONS MIN_COND TYP_COND MAX_COND)
(PROCESS 1 1 1)
(VOLTAGE 3.3 3.3 3.3)
(TEMPERATURE 25 25 25)
```

Default Input Slew

The `DEFAULT_INPUT_SLEW` construct specifies the default minimum rise, minimum fall, maximum rise, and maximum fall times for input slews for all primary input and bidirectional pins of the design.

© 2013-2023 All Rights Reserved.

Syntax

```
default_input_slew
     ::= ( DEFAULT_INPUT_SLEW a_min_rise a_min_fall a_max_rise a_max_fall )

a_min_rise
     ::=  a_number

a_min_fall
     ::=  a_number

a_max_rise
     ::=  a_number

a_max_fall
     ::=  a_number
```

The slews are time values in the units defined by the `TIME_SCALE`. Each slew value may be specified as a numeric value or an asterisk, which serves as a placeholder.

Body Section

Statements in the body section specify timing data and constraints for the design.

Syntax

```
body_statement
     ::= ext_ctlf_files
     ||= waveform_entry+
     ||= clock_entry+
     ||= clock_group_entry+
     ||= driver_cell_entry+
     ||= driver_strength_entry+
     ||= external_load_entry+
     ||= constant_entry+
     ||= input_slew_entry+
     ||= caused_by_net+
     ||= caused_by_null+
```

CTLF Files

The locations of the Timing Library Format (TLF) or compiled TLF (CTLF) files for the design are specified in an *`ext_ctlf_files`* statement.

Syntax

```
ext_ctlf_files
     ::=  ( EXTENSION "CTLF_FILES" ( file_name+  ) )
```


© 2013-2023 All Rights Reserved.

```
file_name
     ::=  STRING
```

Example

```
(EXTENSION "CTLF_FILES"
    (
     /data/tlf/pad/timing.tlf
     /data/tlf/MAX/timing.tlf
     )
)
```

Waveform

The `WAVEFORM` construct defines an abstract periodic waveform.

Syntax

```
waveform_entry
     ::= ( WAVEFORM "waveform_name" period
           ( POSEDGE edge_position ) ( NEGEDGE edge_position ) )

waveform_name
     ::=  STRING

period
     ::= NUMBER

edge_position
     ::= NUMBER
```

The name of the waveform must be unique. The *`period`* describes the interval at which the waveform repeats. The *`edge_position`* gives the time at which the transition takes place relative to the start of each period. *`edge_position`* and *`period`* are represented in units defined by `TIME_SCALE` in the header section.

Example

```
(WAVEFORM "100MHz" 2.8e-08 (POSEDGE 0) (NEGEDGE 1.4e-08))
```

Clock

Each clock is described by relating a previously defined waveform to a *`port`* (the source of that waveform). These ports are usually the roots of a clock network.

© 2013-2023 All Rights Reserved.

Syntax

```
clock_entry
     ::=  ( CLOCK "waveform_name" port )
```

Example

```
(CLOCK "100Mhz" clk_buff/N01)
```

Clock Groups

Clocks within a group are identified by their waveform names. The definitions of the waveforms must precede the clock group specification.

Syntax

```
clock_group_entry
     ::=  ( CLOCK_GROUP "group_name" "waveform_name"+ )
```

Example

```
(CLOCK "IDL_CLK" IDL_CLK)
(CLOCK "clk_derived" clk)
(CLOCK_GROUP "GROUP1"
    "clk_derived"
    "IDL_CLK"
)
```

Driver Cell Section

The `DRIVER_CELL` section contains information about the cells driving the input ports of the design, specifying the cell type and the slews at the driver output pins.

Syntax

```
driver_cell_entry
     ::=  ( DRIVER_CELL
     ( ( CELL_TYPE "cell_name" library )
     ( pos_neg_edge src_pin ) sink_pin  )
     ( su_hd? pos_neg_edge? ( INPUT_SLEW number ) )
     port  )

pos_neg_edge
     ::=  POSEDGE
     ||=  NEGEDGE
```


© 2013-2023 All Rights Reserved.

```
su_hd
     ::=  SETUP
     ||=  HOLD

cell_name
     ::=  STRING

library
     ::=  STRING

src_pin
     ::=  STRING

sink_pin
     ::=  STRING

port
     ::=  STRING
```

The *library* name indicates the library that contains the driver cell. The *pos_neg_edge* item indicates whether the sink pin transitions on the rising (POSEDGE) or falling (NEGEDGE) of the source pin.

The input slew must be specified in the units defined by the TIME_SCALE.

If	Input slew applies to:
HOLD	*min_rise* and *min_fall*
HOLD NEGEDGE	*min_fall*
HOLD POSEDGE	*min_rise*
NEGEDGE	*min_fall* and *max_fall*
POSEDGE	*min_rise* and *max_rise*
SETUP	*max_rise* and *max_fall*
SETUP NEGEDGE	*max_fall*
SETUP POSEDGE	*max_rise*
none	*min_rise*, *min_fall*, *max_rise*, and *max_fall*

port specifies the input or bidirectional port being driven.

Example

```
(DRIVER_CELL ((CELL_TYPE "INBUF" vixen)
              (NEGEDGE BONDING) COREPIN)
```


© 2013-2023 All Rights Reserved.

```
(NEGEDGE (INPUT_SLEW 0))
cpurdwr)
```

This example specifies the `INBUF` cell from the `vixen` library. Sink pin `COREPIN` transitions on the falling edge of source pin `BONDING`. An input slew of 0 applies to the falling edge of the driver output.

Driver Strength Section

The `DRIVER_STRENGTH` section specifies the holding resistance of the cells driving the input ports.

Syntax

```
driver_strength_entry
     ::= ( DRIVER_STRENGTH driver_strength_val port )

driver_strength_val
     ::=  opt_qualifier? resistance
     ||=  min_rise min_fall max_rise max_fall

opt_qualifier
     ::= ( su_hd? pos_neg_edge? )

su_hd
     ::=  SETUP
     ||=  HOLD

pos_neg_edge
     ::=  POSEDGE
     ||=  NEGEDGE

min_rise
     ::=  NUMBER

min_fall
     ::=  NUMBER

max_rise
     ::=  NUMBER

max_fall
     ::=  NUMBER

port
     ::=  STRING
```


© 2013-2023 All Rights Reserved.

The driver strength is a resistance value and must be specified in the units defined by the `RES_SCALE`.

If	Resistance applies to:
`HOLD`	*`min_rise`* and *`min_fall`*
`HOLD NEGEDGE`	*`min_fall`*
`HOLD POSEDGE`	*`min_rise`*
`NEGEDGE`	*`min_fall`* and *`max_fall`*
`POSEDGE`	*`min_rise`* and *`max_rise`*
`SETUP`	*`max_rise`* and *`max_fall`*
`SETUP NEGEDGE`	*`max_fall`*
`SETUP POSEDGE`	*`max_rise`*
none	*`min_rise`*, *`min_fall`*, *`max_rise`*, and *`max_fall`*

External Load Construct

The `EXTERNAL_LOAD` construct specifies the external capacitance, or load, at the specified *port* (pin).

Syntax

```
external_load_entry
     ::=  ( EXTERNAL_LOAD single_or_triple_a_number port )

single_or_triple_a_number
     ::=  a_number
     ||=  a_min a_typ a_max

a_min
     ::= a_number

a_typ
     ::= a_number

a_max
     ::= a_number

port
     ::=  STRING
```


© 2013-2023 All Rights Reserved.

The external load is a capacitance value in the units defined by the `RESISTANCE_SCALE` construct. Each *`a_number`* value may be specified as a number or an asterisk placeholder.

Example

```
(EXTERNAL_LOAD 8.4e-13 usb_bus_out[0])
```

Constant Section

The `Constant` section specifies all non-switching nets and their values.

Syntax

```
constant_entry
     ::=  ( constant_kw net_pin "port" const_val )
     ||=  ( constant_kw net_pin "port" const_val rise_k fall_k)

constant_kw
     ::=  CONSTANT
     ||=  USER_CONSTANT

net_pin
     ::=  NET
     ||=  PIN

port
     ::=  string

const_val
     ::=  1
     ||=  0

rise_k
     ::=  triplet

fall_k
     ::=  triplet
```

Assigns a logical constraint (0 or 1) to the net(s) or pin(s) connected to *`port`*. The *`rise_k`* value represents the rising signal source resistance and the *`fall_k`* represents the falling signal source resistance.

Example

```
(CONSTANT NET "usb_bus_out[21]" 1)
```

This example sets all nets connected to `usb_bus_out[21]` to a logical value of `1`.

© 2013-2023 All Rights Reserved.

Input Slew Section

The INPUT_SLEW section specifies slews at the input ports of the design.

Syntax

```
input_slew_entry
     ::= ( INPUT_SLEW a_min_rise a_min_fall a_max_rise a_max_fall port)

a_min_rise
     ::=  a_number

a_min_fall
     ::=  a_number

a_max_rise
     ::=  a_number

a_max_fall
     ::=  a_number

port
     ::=  string
```

Caused By Section

Nets and pins that are timing constrained have arrival times (timing windows) caused by clock pins.

- All nets/pins constrained by the same clock pin are grouped under a CAUSED_BY section for that *port*.
- A time-constrained net/pin can be contained under multiple CAUSED_BY sections.

Nets and pins that are not timing constrained are grouped under the CAUSED_BY NULL section.

Syntax

```
caused_by_net
     ::=  ( CAUSED_BY "port" rise_fall window_statement+ )

caused_by_null
     ::=  ( CAUSED_BY NULL window_statement+ )

window_statement
     ::= ( net_pin "port" rise_at rise_time rise_k fall_at fall_time fall_k )
     ||=
     ( net_pin "port" rise_at rise_time rise_k fall_at fall_time fall_k
     clock_or_data)
```


© 2013-2023 All Rights Reserved.

```
     ||=
     ( net_pin "port" rise_at rise_time rise_k rise_slack fall_at fall_time
     fall_k fall_slack)
     ||=
     ( net_pin "port" rise_at rise_time rise_k rise_slack fall_at fall_time
     fall_k fall_slack clock_or_data)

net_pin
     ::=  NET
     ||=  PIN

rise_fall
     ::=  RISE
     ||=  FALL

rise_at
     ::=  triplet

rise_time
     ::=  triplet

rise_k
     ::=  triplet

rise_slack
     ::=  triplet

fall_at
     ::=  triplet

fall_time
     ::=  triplet

fall_k
     ::=  triplet

fall_slack
     ::=  triplet

port
     ::=  string

clock_or_data
     ::= C
     ||= D
     ||= U
     ||= Z
```

The following definitions apply to the `CAUSED_BY` parameters:

rise_at	Rising signal arrival time. For `CAUSED_BY NULL` statements, this value defaults to *.
rise_time	Rising signal transition time.

© 2013-2023 All Rights Reserved.

rise_k	Rising signal source resistance.
rise_slack	Rising slack value.
fall_at	Falling signal arrival time. For `CAUSED_BY NULL` statements, this value defaults to *.
fall_time	Falling signal transition time.
fall_k	Falling signal source resistance.
fall_slack	Falling slack value.
clock_or_data	Value that signifies whether the net or pin is part of the clock network. (`C` = Clock; `D` = Data; `U` = Undefined; `Z` = Tristate)

Example

```
( CAUSED_BY "ClkA" RISE
    (NET "NetA" 2.8 0.05 * 1.4 3.0 0.03 * 1.5)
    (NET "NetB" 2.9 0.05 * 1.5 3.0 0.03 * 1.6)
)
( CAUSED_BY "ClkB" RISE
    (NET "NetB" 2.9 0.05 * 1.5 3.0 0.03 * 1.6)
    (NET "NetC" 2.8 0.05 * 1.4 3.0 0.03 * 1.5)
)
```

In this example, `NetA` and `NetB` have their arrival times (timing windows) caused by `ClkA`, and `NetB` and `NetC` are caused by `ClkB`. `NetB` is constrained by multiple clocks and occurs in both `CAUSED_BY` sections.

Example

```
(CAUSED_BY NULL (NET "NetD" * 100 100 * 100 200))
```

In this example, net `NETD` is not time constrained.

Example:

```
(TIMING_WINDOWS
 (HEADER
  (VERSION "0.1")
  (DATE "Mon Nov 17 17:13:22 2003")
  (DESIGN "AIO")
  (DELIMITERS "/[]")
  (VOLTAGE_THRESHOLD 30 70)
  (TIME_SCALE 1e-9)
```


© 2013-2023 All Rights Reserved.

```
  (PROGRAM "pt_to_celtic.tcl" "v3.3 (Aug 14 2003 18:11:07)" "Cadence")
 )
 (WAVEFORM "clocki"   11.250 (POSEDGE    0.000) (NEGEDGE    5.625))
 (WAVEFORM "clocko"   11.250 (POSEDGE    0.000) (NEGEDGE    5.625))
 (WAVEFORM "clocks2"   13.500 (POSEDGE    0.000) (NEGEDGE    6.750))
 (WAVEFORM "clocksi"   13.500 (POSEDGE    0.000) (NEGEDGE    6.750))
 (WAVEFORM "clockso"   13.500 (POSEDGE    0.000) (NEGEDGE    6.750))
 (WAVEFORM "clock"    4.500 (POSEDGE    0.000) (NEGEDGE    2.250))
 (CONSTANT NET "Logic0/output" 0)
 (CONSTANT NET "Logic1/output" 1)
 (CAUSED_BY NULL
  (NET "U40/I" * 1.084:1.084 * 0.242:6.029 * 1.078:1.078 * 0.216:6.101)
 )
. . .
(CAUSED_BY "clocko" FALL
  (NET "U63/I" 7.213:7.213 0.145:0.145 * * 7.244:7.244 0.126:0.126 * *)
 )
 (CONSTANT NET "U74/ZN" 0)
 (CONSTANT NET "U59ASTttcInst689/Z" 1)
 (CONSTANT NET "U75/ZN" 0)
 (CONSTANT NET "U86/ZN" 0)
 (CAUSED_BY "clocki" FALL
  (NET "U61/ZN" 7.226:7.226 0.146:0.146 * * 7.201:7.201 0.136:0.136 * *)
 )
. . .
```

Rail Analysis

This section lists the Rail Analysis file formats in alphabetical order.

Dynamic Trigger File During Rail Analysis

A file that contains the trigger conditions for a detailed dynamic view block. When a detailed dynamic view block is used in a hierarchical design, it needs to be determined when this block is triggered. The dynamic trigger file is used to determine this. At the trigger time, the dynamic current waveforms of the detailed dynamic view block will be applied.

Sample file name: `trigger.txt`

Command: `set_rail_analysis_mode -dynamic_trigger_file` *`filename`*

© 2013-2023 All Rights Reserved.

Format:

```
INSTANCE <name> <trigger_time> <scale_factor> <mode_name>
CELL <name> <trigger_time> <scale_factor> <mode_name>
```

Format Description:

`INSTANCE` and `CELL` keywords should be used to identify whether the trigger should be applied to specific instance or all the instances of the cell name. By default it uses `s` (for example, "`top/insta 2e-9`").

Example:

```
INSTANCE INST_1 1.0 1 MODE1
```

EM Models File

The EM Models file specifies the models needed for Electromigration analysis. If you do not not specify an Electromigration model file, the software will select Electromigration models from the QRC technology file.

Sample file name: `em_models.txt`

Command: `set_rail_analyis_mode -em_models` *`file`* and `set_rail_analyis_mode -process_techgen_em_rules {true | false}`

Format:

Limit-based EM Model Syntax

```
em_model {
      # Width Independent parameters are not affected by width or area
      [em_vcwidth <value>]
      [em_W_n <value>]
      [em_recover <value>]

      # Parameters applying specifically to narrow or wide wires
      [em_jmax_ac_avg_n  <value>]
           [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
      [em_jmax_ac_avg_w  <value>]
           [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
      [em_jmax_ac_peak_n  <value>]
           [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
      [em_jmax_ac_peak_w  <value>]
           [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
      [em_jmax_ac_rms_n  <value>]
           [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
```


© 2013-2023 All Rights Reserved.

```
[em_jmax_ac_rms_w  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_ac_rec_n  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_ac_rec_w  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_dc_avg_n  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_dc_avg_w  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_dc_peak_n  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_dc_peak_w  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_dc_rms_n  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
[em_jmax_dc_rms_w  <value>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
```

 # Width dependent parameters for a conductor or diffusion layer

```
[em_jmax_ac_avg <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [via_range <value>]
[em_jmax_ac_peak <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [via_range <value>]
[em_jmax_ac_rms <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [via_range <value>]
[em_jmax_ac_rec <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [via_range <value>]
[em_jmax_dc_avg <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>....]
     [via_range <value>] [L  < | <= | > | >=   <value in microns>]
     [W < | <= | > | >=   <value in microns>]
[em_jmax_dc_peak <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [via_range <value>] [L  < | <= | > | >=   <value in microns>]
     [W < | <= | > | >=   <value in microns>]
[em_jmax_dc_rms <value> | <value_1> <width_1> [...] | EQU <fn(E)> ]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>....]
     [via_range <value>] [L  < | <= | > | >=   <value in microns>]
     [W < | <= | > | >=   <value in microns>]
```


© 2013-2023 All Rights Reserved.

```
# Area dependent parameters for a via layer
[em_jmax_ac_avg <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [current_direction up|down|both] [via_range <value>]
[em_jmax_ac_peak <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [current_direction up|down|both] [via_range <value>]
[em_jmax_ac_rms <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [current_direction up|down|both] [via_range <value>]
[em_jmax_ac_rec <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.....]
     [current_direction up|down|both] [via_range <value>]
[em_jmax_dc_avg <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>....]
     [current_direction up|down|both] [via_range <value>]
     [L  < | <= | > | >=   <value in microns>]
     [W < | <= | > | >=   <value in microns>]
[em_jmax_dc_peak <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>....]
     [current_direction up|down|both] [via_range <value>]
     [L  < | <= | > | >=   <value in microns>]
     [W < | <= | > | >=   <value in microns>]
[em_jmax_dc_rms <value> | <value_1> <area_1> [...] | EQU <fn(E)>]
     [jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]]
     [jmax_lifetime <lifetime1> <scale1> [<lifetime2> <scale2>.......]
     [current_direction up|down|both] [via_range <value>]
     [L  < | <= | > | >=   <value in microns>]
     [W < | <= | > | >=   <value in microns>]
}
```

Risk-based EM Model Syntax

```
em_model {
# Width Independent parameters are not affected by width
[em_vcwidth <value>]
[em_W_n <value>]
[em_m <value>]
[em_n <value>]
[em_sigma <value>]

# Parameters applying specifically to narrow or wide wires
[em_A_n  <value>]
[em_A_w <value>]
```


© 2013-2023 All Rights Reserved.

```
[em_Ea_n <value>]
[em_Ea_w <value>]

# Width/Area PWL parameters
[em_A <value>]
[em_Ea <value>]
}
```

Format Description:

Parameters for both Risk-based and Limit-based Analysis

`em_vcwidth`	Optional parameter for `Via` layers that defines the width of a standard via in microns. The analysis tool assumes that the via is square, so it squares a side to obtain the area. Do not specify this parameter for conductor or diffusion layers. If PWL value/area pairs or an equation is defined then vcwidth is not required to calculate current density limits and it is ignored. **Default:** 0 for metal layers and 0.5 for cut layers **Range:** Any integer or floating point number `vcwidth` is specified as microns instead of meters to be consistent with the rest of the ICT file.
`em_W_n`	Optional parameter that specifies the wire width in microns below which a line is considered narrow. **Note:** all lines are considered wide unless you specify otherwise using this parameter. **Default:** 0 **Range:** Any integer or floating point number.

Parameters for Risk-based Analysis

© 2013-2023 All Rights Reserved.

`em_A`

Optional parameter that specifies the value of A when current densities are defined in PWL format. Its unit is specified as: A^2h/kcm^4.

Default: 700
Range: A positive integer or floating point number

`em_A_n`

Specifies the value of A to use for narrow lines. Its unit is specified as: A^2h/kcm^4.

Default: 700
Range: A positive integer or floating point number

`em_A_w`

Specifies the value of A to use for wide lines. Its unit is specified as: A^2h/kcm^4.

Default: 45000
Range: A positive integer or floating point number

`em_Ea`

Optional parameter that specifies the activation energy value to use for PWL linear interpolation. Its unit is in electron volts.

Default: 0.8 ev
Range: A positive integer or floating point number

`em_Ea_n`

Specifies the activation energy value to use for narrow lines. Its unit is specified in electron volts.

Default: 0.8 ev
Range: A positive integer or floating point number

`em_Ea_w`

Specifies the activation energy value to use for wide lines. Its unit is specified in electron volts.

Default: 0.6 ev
Range: A positive integer or floating point number

`em_m`

Optional parameter that specifies the temperature coefficient used by Black's equation for electromigration risk assessment.

Default: 1
Range: A positive integer or floating point number

© 2013-2023 All Rights Reserved.

`em_n`

Optional parameter that specifies the current exponent used by Black's equation for electromigration risk assessment.

Default: 2.0 for metal layers; 1.0 for Via layers
Range: A positive integer or floating point number

`em_sigma`

Optional parameter that specifies the value to be used in the lognormal failure rate model for predicting the lifetime of a circuit.

Default: 0.7
Range: A positive integer or floating point number

Parameters for Limit-based Analysis:

`em_jmax_ac_avg`

Optional parameter that specifies the maximum AC average current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: None

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

`em_jmax_ac_avg_n`

This optional parameter is essentially the same as `em_jmax_ac_avg`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_ac_avg_w`

This optional parameter is essentially the same as `em_jmax_ac_avg`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

© 2013-2023 All Rights Reserved.

`em_jmax_ac_peak`

Optional parameter that specifies the maximum AC peak current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: None

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

`em_jmax_ac_peak_n`

This optional parameter is essentially the same as `em_jmax_ac_peak`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_ac_peak_w`

This optional parameter is essentially the same as `em_jmax_ac_peak`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_ac_rms`

Optional parameter that specifies the maximum AC RMS current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: None

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

© 2013-2023 All Rights Reserved.

`em_jmax_ac_rms_n` This optional parameter is essentially the same as `em_jmax_ac_rms`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_ac_rms_w` This optional parameter is essentially the same as `em_jmax_ac_rms`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_ac_rec` Optional parameter that specifies the maximum recovery AC current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: None

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

`em_jmax_ac_rec_n` This optional parameter is essentially the same as `em_jmax_ac_rec`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_ac_rec_w` This optional parameter is essentially the same as `em_jmax_ac_rec`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

© 2013-2023 All Rights Reserved.

`em_jmax_dc_avg`

Optional parameter that specifies the maximum DC average current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: 1.0e+5 A/cm^2 for conductor layers, 0.001A/via for via layers.

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

`em_jmax_dc_avg_n`

This optional parameter is essentially the same as `em_jmax_dc_avg`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_dc_avg_w`

This optional parameter is essentially the same as `em_jmax_dc_avg`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

© 2013-2023 All Rights Reserved.

`em_jmax_dc_peak`

em_jmax_dc_peak is the maximum DC peak current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: 1.0e+7 A/cm^2 for conductor layers, 0.001A/via for via layers.

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

`em_jmax_dc_peak_n`

This optional parameter is essentially the same as `em_jmax_dc_peak`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_dc_peak_w`

This optional parameter is essentially the same as `em_jmax_dc_peak`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

© 2013-2023 All Rights Reserved.

`em_jmax_dc_rms`

em_jmax_dc_rms is the maximum DC RMS current density which can either be defined as a single value, a PWL table, or an equation (`EQU`).

For a conductor or diffusion layer the PWL table is defined as the current density limit value based on the wire width. For via layers it is defined as the current density limit value based on the via area.

Default: 1.0e+6 A/cm^2 for conductor layers, 0.001A/via for via layers.

Range: A positive integer or floating point number, a PWL table, or an equation (`EQU`). The value should be specified before any qualifiers (`jmax_factor`, `current_direction`, and `via_range`). The qualifiers themselves are not order dependent and may be specified as needed.

`em_jmax_dc_rms_n`

This optional parameter is essentially the same as `em_jmax_dc_rms`, with the exception that it is specified for narrow wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_jmax_dc_rms_w`

This optional parameter is essentially the same as `em_jmax_dc_rms`, with the exception that it is specified for wide wires (as defined by `em_W_n`). In addition, this parameter only accepts a single value, not a PWL table or an equation (EQU).

`em_recover`

Optional parameter that specifies the recovery factor used for calculating the average current density with recovery. You can set the recovery factor to -1.0 when you want to account for the average of the absolute current (avg(abs(current))) so that the negative current does not cancel the positive current values in arriving at the average current result.

Default: 1

Range: A positive integer or floating point number

Example:

```
# Jmax Calculation: e.g. METAL_1
# Imax = 0.36*2.6*(w-0.02)
# For w= 0.12u, Imax = 0.36*2.0*(mA/u)*(0.1u) = 0.09mA
```


© 2013-2023 All Rights Reserved.

```
# Here (0.36*2.6) multiplying factor is the current density limit J.
# Now, Jmax = Imax/(Thickness*width) = 0.09 e-3 (Amp)/(0.24e-4*0.12e-4) (cm^2) =
0.3e+5 A/cm^2
# Thickness for the layer can be found in icecaps.ict file.
# For different width, you can generate the PWL limits and the software will
# interpolate between these values based on width of the resistor.
# e.g.
# model METAL_1 jmax_dc_avg  =   "0.25e+5 0.1e-6 0.3e+5 0.12e-6 0.38e+5 0.14e-6
1.25e+5 0.16e-6 1.61e+5 0.2e-6 2.51e+5 0.3e-6 5.21e+5 .6e-6 8.81e+5 1e-6 35.8e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.


# Where first value is calculated Jmax and second is width. "Jmax width1 Jmax width2
Jmax width3....


## METAL_8
model METAL_8 TREF           =   110
model METAL_8 Tc1            =   0.0000
model METAL_8 Tc2            =   0.0000
model METAL_8 thickness      =   0.7744e-06; #Meters
model METAL_8 jmax_dc_avg  =   "0.74e+5 0.1e-6 0.93e+5 0.12e-6 1.11e+5 0.14e-6
1.3e+5 0.16e-6 1.67e+5 0.2e-6 2.6e+5 0.3e-6 3.72e+5 .42e-6 5.39e+5 0.6e-6 9.11e+5
1e-6 37e+5 4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_8 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]


## VIA_7
model VIA_7 TREF   =   110
model VIA_7 Tc1            =   0.0000
model VIA_7 Tc2            =   0.0000
model VIA_7 vcwidth        =   0.36e-06;  #Meters
model VIA_7 jmax_dc_avg_w  =   1.452e-03;  #[A/Via]
#model VIA_7 jmax_dc_peak  =   0.713e-03;  #[A]


## METAL_7
model METAL_7 TREF           =   110
model METAL_7 Tc1            =   0.0000
model METAL_7 Tc2            =   0.0000
model METAL_7 thickness      =   0.2986e-06; #Meters
model METAL_7 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_7 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]


## VIA_6
```


© 2013-2023 All Rights Reserved.

```
model VIA_6 TREF   =   110
model VIA_6 Tc1            =   0.0000
model VIA_6 Tc2            =   0.0000
model VIA_6 vcwidth        =   0.13e-06;  #Meters
model VIA_6 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]
#model VIA_6 jmax_dc_peak  =   0.713e-03;  #[A]

## METAL_8
model METAL_8 TREF           =   110
model METAL_8 Tc1            =   0.0000
model METAL_8 Tc2            =   0.0000
model METAL_8 thickness      =   0.7744e-06; #Meters
model METAL_8 jmax_dc_avg  =   "0.74e+5 0.1e-6 0.93e+5 0.12e-6 1.11e+5 0.14e-6
1.3e+5 0.16e-6 1.67e+5 0.2e-6 2.6e+5 0.3e-6 3.72e+5 .42e-6 5.39e+5 0.6e-6 9.11e+5
1e-6 37e+5 4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_8 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]

## VIA_7
model VIA_7 TREF   =   110
model VIA_7 Tc1            =   0.0000
model VIA_7 Tc2            =   0.0000
model VIA_7 vcwidth        =   0.36e-06;  #Meters
model VIA_7 jmax_dc_avg_w  =   1.452e-03;  #[A/Via]
#model VIA_7 jmax_dc_peak  =   0.713e-03;  #[A]

## METAL_7
model METAL_7 TREF           =   110
model METAL_7 Tc1            =   0.0000
model METAL_7 Tc2            =   0.0000
model METAL_7 thickness      =   0.2986e-06; #Meters
model METAL_7 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_7 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]

## VIA_6
model VIA_6 TREF   =   110
model VIA_6 Tc1            =   0.0000
model VIA_6 Tc2            =   0.0000
model VIA_6 vcwidth        =   0.13e-06;  #Meters
model VIA_6 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]
#model VIA_6 jmax_dc_peak  =   0.713e-03;  #[A]
```


© 2013-2023 All Rights Reserved.

```
## VIA_3
model VIA_3 TREF   =   110
model VIA_3 Tc1            =   0.0000
model VIA_3 Tc2            =   0.0000
model VIA_3 vcwidth        =   0.13e-06;  #Meters
model VIA_3 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]
#model VIA_3 jmax_dc_peak  =   0.713e-03;  #[A]

## METAL_3
model METAL_3 TREF           =   110
model METAL_3 Tc1            =   0.0000
model METAL_3 Tc2            =   0.0000
model METAL_3 thickness      =   0.2986e-06; #Meters
model METAL_3 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_3 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]

## VIA_2
model VIA_2 TREF   =   110
model VIA_2 Tc1            =   0.0000
model VIA_2 Tc2            =   0.0000
model VIA_2 vcwidth        =   0.13e-06;  #Meters
model VIA_2 jmax_dc_avg_w  =   0.189e-03;  #[A/Via]
#model VIA_2 jmax_dc_peak  =   0.713e-03;  #[A]

## METAL_2
model METAL_2 TREF           =   110
model METAL_2 Tc1            =   0.0000
model METAL_2 Tc2            =   0.0000
model METAL_2 thickness      =   0.2986e-06; #Meters
model METAL_2 jmax_dc_avg  =   "0.69e+5 0.1e-6 0.87e+5 0.12e-6 1.04e+5 0.14e-6
1.21e+5 0.16e-6 1.56e+5 0.2e-6 2.43e+5 0.3e-6 5.05e+5 .6e-6 8.53e+5 1e-6 34.65e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_2 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]

## VIA_1
model VIA_1 TREF   =   110
model VIA_1 Tc1            =   0.0000
model VIA_1 Tc2            =   0.0000
# vcwidth is the default via width.
```


© 2013-2023 All Rights Reserved.

```
model VIA_1 vcwidth        =   0.13e-06;  #Meters
model VIA_1 jmax_dc_avg_w  =   0.189e-3;  #[A/Via]
#model VIA_1 jmax_dc_peak  =   0.713e-03;  #[A]

## METAL_1
model METAL_1 TREF           =   110
model METAL_1 Tc1            =   0.0000
model METAL_1 Tc2            =   0.0000
model METAL_1 thickness      =   0.24e-06; #Meters
model METAL_1 jmax_dc_avg  =   "0.25e+5 0.1e-6 0.3e+5 0.12e-6 0.38e+5 0.14e-6
0.42e+5 0.16e-6 0.54e+5 0.2e-6 0.83e+5 0.3e-6 1.75e+5 0.6e-6 2.96e+5 1e-6 11.96e+5
4e-6";  #[A/cm^2]  pwl limits according to widths.
#model METAL_1 jmax_dc_peak  =   4.950e+05;  #[A/cm^2]

# CONT
model CONT TREF   =   110
model CONT Tc1            =   0.0000
model CONT Tc2            =   0.0000
model CONT vcwidth        =   0.12e-06;  #Meters
model CONT jmax_dc_avg_w  =   0.294e-3;  #[A/Cont]
#model CONT jmax_dc_peak  =   0.713e-03;  #[A]
```

EM Only ICT File

The EM only ICT file provides the process and EM model information for EM analysis. This is a text-based file, which means you can use any text editor to enter information in this file.

Sample file name: `file.ict`

Command: `set_rail_analysis_mode -process_techgen_em_rules true -ict_em_models` *`ICT-EM file`*

Format:

```
process "name" {
  background_dielectric_constant value
  layout_scale value
  temp_reference value
# EM_Model parameters
  [em_tref <value>]
  [em_output_wlt <silicon|drawn>]
  [em_variables <variable1> [<variable2> [ ...]]]
  [em_conductor_unit mA]
  [em_via_unit mA ]
  [em_via_area_unit mA ]
  [em_segment_length <perimeter | centerline>]
}
```


© 2013-2023 All Rights Reserved.

```
conductor "name" {
em_model {
 ...
 }
}
```

Format Description:

`background_dielectric_constant value` - Specifies the dielectric constant for the region above the top or last dielectric layer. This field is required.

`layout_scale value` - Specifies the ratio of the scaled dimensions divided by the drawn dimensions. The ICT file is defined based on the scaled dimensions.

`temp_reference value` - Specifies the reference temperature in Celsius. This is the nominal temperature used when the tech file resistance information is calculated.

`em_tref <value>` - Specifies the nominal temperature for EM analysis in Celsius. The default value is 25C, and can be specified as any integer value.

`em_output_wlt [ silicon | drawn ]` - This determines whether the width and length parameters are written out as drawn dimensions or silicon dimensions, and whether erosion tables for thickness variation are output. The default setting is drawn.

`em_variables` - This parameter is used to globally declare variables that are used within the various em_models. w and delta_T are used normally.

- `w`: width of the conductor
- `delta_T`: ΔT(°C) is the temperature rise due to Joule heating. delta_T is an input for the following command parameters:
  - For P/G EM, `set_rail_analysis_mode -rms_em_delta_T <value>`
  - For signal EM, `verify_AC_limit -deltaTemp <value>`

Note: `l` (length) is not supported in `em_variables`. That is, `l` is not supported in `EQU` or `PWL`.

`em_conductor_unit mA` - Specifies the units to use for current density limits on conductor layers. Only mA is supported.

`em_via_unit mA` - Specifies the units to use for current density limits on via layers on a simple per via basis. Only mA is supported.

`em_via_area_unit mA` - Specifies the units to use for piecewise linear (PWL) current density limits for via layers using density/area value pairs. Only mA is supported.

© 2013-2023 All Rights Reserved.

`em_segment_length [perimeter | centerline]` - Determines how to calculate the length of a wire segment. This parameter determines whether the segment length is of the type `perimeter` or `centerline`. The difference between the `perimeter` and `centerline` segment length types is as follows:

- `perimeter` - specifies that the length calculation is along the wire outer edge boundary. This value is used for 28-nm process nodes.
- `centerline` - specifies that the length is extended to the edge of the segment along the centerline path. This value is used for 20-nm process nodes.

Rules for EM model in Conductor layer

■ Equations in EM Model

The conductor layer usually uses equation (EQU) to describe the EM model. EQU is a function of the conductor/metal width, fn(w). The operators used in EQU are: +, -, *, /, ^, sqrt, (). You must specify the EQU statements in the same fn(w) form with different beneficial factors, and the EQU statements must be listed in the ascending order of beneficial factors.

```
em_model {
   em_jmax_dc_avg EQU 1 * 0.123 * (w – 0.001) jmax_factor 100 2.0 105 1.5 110
1.0 115 0.8 120 0.6 125 0.4 W <= 2
   em_jmax_dc_avg EQU 2 * 0.123 * (w – 0.001) jmax_factor 100 2.0 105 1.5 110
1.0 115 0.8 120 0.6 125 0.4 W > 2 W <= 3
   em_jmax_dc_avg EQU 3 * 0.123 * (w – 0.001) jmax_factor 100 2.0 105 1.5 110
1.0 115 0.8 120 0.6 125 0.4 W > 3
         }
```

In the above EQU statements:

`fn(w) =` **0.123 * (w – 0.001)**

Beneficial factors are **1**, **2**, and **3**, respectively. The beneficial factor must be preceded to fn(w).

■ jmax_factor

```
jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]
```

EQU may have the temperature scaling table, `jmax_factor`. `jmax_factor` is the optional scaling factor to use at different temperatures compared to the reference temperature (defined by `em_tref` in the process definition). The temperature for `jmax_factor` should be specified in degrees Celsius. Scaling factor is a positive integer: >1 to scale up, <1 to scale

© 2013-2023 All Rights Reserved.

down, or 1 for no scale effect. Normally, the lower temperature has a bigger scale factor, while the higher temperature has a smaller scale factor for the current limit.

```
em_model {
    em_jmax_dc_avg EQU 1 * 0.123 * (w - 0.001) jmax_factor 100 2.0 105
1.5 110 1.0 115 0.8 120 0.6 125 0.4
}
```

Note: When setting scale factors for multiple temperatures, they should be specified in an ascending sequence to enable interpolation.

The temperature in `jmax_factor` for EM calculation is the input by `set_rail_analysis_mode -em_temperature <value>`.

■ Length or Width Condition

EQU may have the width (W) and/or length (L) of conductor/metal as the condition. The L parameter specifies the length-based jmax values in microns. The W parameter specifies the width-based jmax values in microns. L and W are applicable for both metals and vias.

```
em_model {
    em_jmax_dc_avg EQU 1 * 0.123 * (w - 0.001) jmax_factor 100 2.0 105
1.5 110 1.0 115 0.8 120 0.6 125 0.4 L > 4 W < 1
    em_jmax_dc_avg EQU 2 * 0.123 * (w - 0.001) jmax_factor 100 2.0 105
1.5 110 1.0 115 0.8 120 0.6 125 0.4 L >= 2 L <= 4
}
```

The qualifiers of W or L must be preceded to inequality symbols.

W >= 2 W <= 4; acceptable
2 <= W <= 4; not acceptable

■ delta_T Variable

EQU may have the variable, `delta_T`, which is the input by `set_rail_analysis_mode -rms_em_delta_T <value>`. `delta_T` is the maximum temperature rise permitted in Celsius degree. This value is used for root-mean-square (rms) Electronmigration current limit analysis.

© 2013-2023 All Rights Reserved.

```
em_model {
    em_jmax_ac_rms EQU sqrt( 1.23 * delta_T * ( w - 0.001 )^2 )
}
```

■ Implicit /sqrt(r) in em_jmax_ac_peak

For AC peak current limit, it may have EQU as shown below:

```
em_model {
    em_jmax_ac_peak EQU 123.12 * (w - 0.002)
}
```

When the AC peak current limit is calculated, the formulation of "`123.12*(w-0.002)/sqrt(r)`" is used inside the tool. That is /sqrt(r) is always implicit in the EQU of `em_jmax_ac_peak`. "r" is duty ratio.

Rules for EM model in Via layer

■ Piecewise Linear Interpolation

The via layer usually uses Piecewise Linear (PWL) to describe the EM model. The keyword, PWL, must be defined explicitly before the piecewise linear table. In the PWL statements, there must be a base PWL table without any conditions. In the following PWL statements, the blue line is the base PWL table without any conditions. Generally, it should have the lowest current limit, as others have the current limit with the multiplier of beneficial factors (>1.0).

```
em_model {
    em_vcwidth 0.040
    em_jmax_dc_avg PWL 0.666 0.0032 0.999 0.0064 jmax_factor 100 2.0 105
1.5 110 1.0 115 0.8 120 0.6 125 0.4 Lu < 2 current_direction down
    em_jmax_dc_avg PWL 0.222 0.0016 0.444 0.0032 0.666 0.0064 jmax_factor
100 2.0 105 1.5 110 1.0 115 0.8 120 0.6 125 0.4 Lu >= 2 Lu < 4
current_direction down
    em_jmax_dc_avg PWL 0.111 0.0016 0.222 0.0032 0.333 0.0064 jmax_factor 100 2.0
105 1.5 110 1.0 115 0.8 120 0.6 125 0.4
}
```


© 2013-2023 All Rights Reserved.

The base PWL table should contain all the possible cut areas as the index. For example, in the blue line, it is 0.0016 (Vx), 0.0032 (VxBAR), 0.0064 (VxLRG). A PWL table's index is the area of cut. The unit is µm2. The PWL table syntax is:

```
PWL value1 index1 [value2 index2 …]
```

Normally, a via layer has 1x square via, rectangular via, and 2x square via. Therefore, the area index will follow area of 1x square via, area of rectangular via, and area of 2x square via. The area index must be listed in the ascending order. Ensure that all the areas of the possible cut shapes are listed in the table; interpolation will not be performed for calculation.

For the PWL tables with conditions (not the base table), the current limits of different cut areas must have the identical beneficial factor against the respective values in the base table. For example, in the first table, the beneficial factor is 3=0.666/0.222=0.999/0.333; in the second table, the beneficial factor is 2=0.222/0.111=0.444/0.222=0.666/0.333.

The parameter, `em_vcwidth`, is only used for a via/cut layer, and not used for a conductor or diffusion layer. It is the cut width of the square via. You must define this parameter explicitly for a via layer.

■ jmax_factor

```
jmax_factor <temp1> <scale1> [<temp2> <scale2> ...]
```

PWL table may have the temperature scaling table, `jmax_factor`. `jmax_factor` is the optional scaling factor to use at different temperatures compared to the reference temperature (defined by `em_tref` in the `process` definition). The temperature for `jmax_factor` should be specified in degrees Celsius. Scaling factor is a positive integer: >1 to scale up, <1 to scale down, or 1 for no scale effect. Normally, the lower temperature has a bigger scale factor, while the higher temperature has a smaller scale factor for the current limit.

```
em_model {
    em_jmax_dc_avg EQU 1 * 0.123 * (w - 0.001) jmax_factor 100 2.0 105
1.5 110 1.0 115 0.8 120 0.6 125 0.4
}
```

Note: When setting scale factors for multiple temperatures, they should be specified in an ascending sequence to enable interpolation.

The temperature in *jmax_factor* for EM calculation is the input by `set_rail_analysis_mode -em_temperature <value>`.

■ Conditions

© 2013-2023 All Rights Reserved.

PWL table may have the width (W) and/or length (L) of of the upper/bottom metal of the via as the condition.

```
em_model {
    em_vcwidth 0.040
    em_jmax_dc_avg PWL 0.111 0.0016 0.222 0.0032 0.333 0.0064 jmax_factor
100 2.0 105 1.5 110 1.0 115 0.8 120 0.6 125 0.4 Lu > 2 Lu <= 4 Lb <= 4
}
```

The various conditions are:

- The `L` parameter specifies the length-based jmax values in microns.
- The `W` parameter specifies the width-based jmax values in microns.
- `Lu`, `Lb`, `Wu`, and `Wb` are applicable for vias only.

Where,
`L` = length
`W` = Width
`Lu` = Upper metal length of the via
`Lb` = Bottom metal length of the via
`Wu` = Upper metal width of the via
`Wb` = Lower metal width of the via

- **Current Direction**

PWL table may have different current limits if the current through the cut has a specific direction. The `current_direction` syntax is:

```
current_direction <up|down>
```

```
 em_model {
    em_vcwidth 0.040
    em_jmax_dc_avg PWL 0.123 0.0016 0.222 0.0032 0.333 0.0064 jmax_factor
100 2.0 105 1.5 110 1.0 115 0.8 120 0.6 125 0.4 Lb > 4 Wb >= 0.1
current_direction up
    em_jmax_dc_avg PWL 0.321 0.0016 0.222 0.0032 0.333 0.0064 jmax_factor
100 2.0 105 1.5 110 1.0 115 0.8 120 0.6 125 0.4 Lu > 4 Wu >= 0.1
current_direction down
}
```

`up` - means the current flows from the bottom metal to the upper metal, and specifies to use the length and width of the metal below Lb/Wb for the VIA Jmax factor

© 2013-2023 All Rights Reserved.

`down` - means the current flows from the upper metal to the bottom metal, and specifies to use the length and width of the metal above Lu/Wu for VIA Jmax factor

The current direction has the converse direction of electrons flow.

■ single Keyword for Single Square Cut on Metal Layers

PWL table may have a specific current limit for single square cut via (not via array).

```
em_model {
    em_vcwidth 0.040
    em_jmax_dc_avg PWL 0.222 0.0016 jmax_factor 100 2.0 105 1.5 110 1.0
115 0.8 120 0.6 125 0.4 Lu > 2 Lu <= 4 Lb <= 4 current_direction up single
 }
```

Example:

```
process "gpdk090" {
  background_dielectric_constant 1.000000
  temp_reference 25
  layout_scale 0.9
  #! use_silicon_density true
  em_tref 110
  em_output_wlt drawn
  em_variables w delta_T
  em_conductor_unit mA
  em_via_area_unit mA
  em_via_unit mA
  }
conductor "mt1" {
em_model {
em_jmax_dc_avg EQU 100 * 1* ( w - 0.008 ) jmax_factor 100 2.077 105 1.434 110 1.000
115 0.704 120 0.500 125 0.358 L > 9 W < 0.45
em_jmax_dc_avg EQU 150 * 1.227 * ( w - 0.008 ) jmax_factor 100 2.077 105 1.434 110
1.000 115 0.704 120 0.500 125 0.358 L <= 9 L > 4.5 W < 0.45
em_jmax_dc_avg EQU 200 * 1.227 * ( w - 0.008 ) jmax_factor 100 2.077 105 1.434 110
1.000 115 0.704 120 0.500 125 0.358 L > 9 W >= 0.45
em_jmax_dc_avg EQU 200 * 1.227 * ( w - 0.008 ) jmax_factor 100 2.077 105 1.434 110
1.000 115 0.704 120 0.500 125 0.358 L <= 9 L > 4.5 W >= 0.45
em_jmax_dc_avg EQU 400 * 1.227 * ( w - 0.008 ) jmax_factor 100 2.077 105 1.434 110
1.000 115 0.704 120 0.500 125 0.358 L <= 4.5
em_jmax_dc_peak EQU 25.0 * ( w - 0.008 ) em_jmax_ac_peak EQU 25.0 * ( w - 0.008 )
em_jmax_ac_rms EQU sqrt( 18.58 * delta_T * ( w - 0.008 )^2 * ( w - 0.008 + 0.246 )
/ ( w - 0.008 + 0.0443 ) )
    }
}
```

Package Model and Mapping File

The Package Model file contains a SPICE subcircuit describing a package model. It enables Voltus to use package models during full-chip static and dynamic power-grid analysis.

© 2013-2023 All Rights Reserved.

The Mapping file specifies the name of the package terminal mapping file, which contains the mapping of the terminals of the SPICE subcircuit in the package model file to the locations at the grid or top-level pins. The mapping file formats supported by Voltus are:

- Voltus-specific Format
- Model Connection Protocol (MCP)

Command: `set_package -spice` *`model_file`* `-mapping` *`mapping_file`*

Voltus-specific Format

The format of the package terminal mapping file is as follows:

```
subckt_terminal_name
subckt_terminal_name pin toplevel_pin_name
subckt_terminal_name cell pad_cell_instance_name
subckt_terminal_name x,y
subckt_terminal_name [power_pin|ground_pin] net_name
```

Format Description:

`subckt_terminal_name` specifies the name of the terminals of the SPICE subcircuit in the package model file to be mapped to the grid locations or the top-level pins. subckt_terminal_name without mapping information indicates that the terminal should be kept floating. subckt_terminal_name specified with the power or ground keyword indicates that the subcircuit terminal should be connected to a virtual power or ground node during power-grid analysis.

`pin toplevel_pin_name` specifies the name of the top-level pin to which to connect the SPICE subcircuit terminal.

`cell pad_cell_instance_name` specifies the name of the pad cell instance to which to connect the SPICE subcircuit terminal.

`x,y` specifies the location, in microns, where the specific SPICE subcircuit terminal should be attached.

`power_pin net_name` specifies the power pin name and net name to which to connect the SPICE subcircuit terminal.

`ground_pin net_name` specifies the ground pin name and net name to which to connect the SPICE subcircuit terminal.

Example:

© 2013-2023 All Rights Reserved.

The following is an example of a terminal mapping file:

```
VDD_in power_pin VDD_CPU
VSS_in ground_pin VSS
VDD_out pin  VDD_CPUvsrc1
VDD_out pin  VDD_CPUvsrc2
VDD_out pin  VDD_CPUvsrc3
```

Model Connection Protocol (MCP)

The MCP mapping file format can be used for location based pin mapping for package models. MCP is a block that is inserted into a host file as the comment section. It uses the host file's comment leading strings as its leading string.

The format of the MCP package terminal mapping file is as follows:

```
* [MCP Begin]
* [MCP Ver] 1.1
* [REM] comment till the end of the line
* [Structure Type] PKG
* [MCP Source] source
* [Coordinate Unit] unit
* [Connection] compName modelName totalNumberOfPins
* [Connection Type] DIE

* [Power Nets]
* pin cktnode net x y
* …
* pin cktnode net x y

* [Ground Nets]
* pin cktnode net x y
* …
* pin cktnode net x y

* [Signal Nets]
* pin cktnode net x y
* …
* pin cktnode net x y
* [MCP End]
```

Format Description:

`[MCP Begin]` and `[MCP End]` mark the MCP block.

The contents of the MCP block are described below.

`[MCP Ver] 1.1` is a one-line item. It specifies the version of MCP. It is a required item.

`[REM]` comment is the MCP comment. It is a one-line item. The comment can also be in the form from the host, except in the net list section. In the net list sections, the comment must be in the form of MCP comments.

© 2013-2023 All Rights Reserved.

`[Structure Type]` PKG specifies the type of the extracted circuit model. It is a one-line item. It is optional. It can be a combination of the following (delimited by spaces) DIE, PKG, PCB.

`[MCP Source]` source specifies the origin of the circuit model. It is a one-line item. It can include the tool name, the tool version, and/or the date the model was created. This item is optional.

`[Coordinate Unit]` unit defines the unit used by the coordinates. unit can be one of the following, m, cm, mm, um, nm, mil. The default is m. This item is optional.

`[Connection] compName modelName totalNumberOfPins` starts a component connection section in the MCP block. It is a one-line item. This item is required. An MCP block should contain at least one component connection section. In the item, compName is the name of the component connection (or circuit name in SPD file, or reference designator in common layout tools); modelName is the model definition name of the component connection (or part number in common layout tools); and the totalNumberOfPins is the number of pins in this component connection. A new [Connection] line ends the previous component connection section and starts a new component connection section.

`[Connection Type] DIE` specifies the type of the component connection. It is a one-line item. It is optional. It can be a combination of the following (delimited by spaces) DIE, PKG, PCB, VRM, or SINK.

`[Power Nets]` starts the power pin section of the component. It is a one-line item. It is optional.

`pin cktnode net x y` lists the information for a pin. pin is the name of the pin. cktnode is the circuit node name used in the extracted circuit corresponding to the pin. net is the net of the pin. x y are optional and are the relative coordinates of the pin relative to the component origin. The five fields should be in one line.

`[Ground Nets]` starts the ground pin section of the component. It is a one-line item. It is optional.

`[Signal Nets]` starts the signal pin section (ends the [Power Nets] or [Ground Nets] section if exists) of the component connection. It is a one-line item. It is optional.

In the Power/Ground/Signal pin sections, comments must be in the MCP comment form as [REM] comment.

Example:

The following is an example of an MCP file:

© 2013-2023 All Rights Reserved.

```
.subckt SigrityModel list_of_external_nodes

* The following is the MCP Section

****************************************
* [MCP Begin]

* [MCP Ver] 1.1

* [MCP Source] Sigrity XtractIM 1.3.1 12/31/2007

* [Coordinate Unit] um

*
****************************************
* The following is the info for component connection U1

****************************************

* [Connection] U1 Si_fc_die_3000 3000
* [Connection Type] DIE
* [REM]
* [REM] -----Power Pin List of U1----------
* [REM]

* [Power Nets]
* VDD01 CKT01 NET1 0 0
* VDD02 CKT02 NET1 200 0

* [REM]
* [REM] -----Ground Pin List of U1----------
* [REM]

* [Ground Nets]
* [REM] Pins of net NET2
* VSS01 CKT03 NET2 100 0
* VSS01 CKT04 NET2 300 0
* …

* [REM]
* [REM] -------------Signal Pin List of U1----------
* [REM]

* [Signal Nets]
* SIG01 CKT05 NET3 500 300
* SIG01 CKT06 NET3 300 500
* …
*
* [MCP End]
* This concludes the MCP section
* Contents of the subckt
.ends
```

Power-Up Sequence File

A file containing the firing time for each power gate instance.

Sample file name: `sequence.txt`

© 2013-2023 All Rights Reserved.

Command: `set_rail_analysis_mode -powerup_sequence_file` *`filename`*

Format:

```
<inst_name> <pin_name> <always_on_net> <switch_net> <rise_time> <rise_slew>
<fall_time> <fall_slew>
```

Format Description:

`inst_name` specifies the name of the full hierarchical power gate instance.

`pin_name` specifies the name of the enable pin.

`always_on_net` specifies the name of the always-on net.

`switch_net` specifies the name of the switched net.

`rise_time` specifies the rise time at the enable pin.

`rise_slew` specifies the rise slew at the enable pin.

`fall_time` specifies the fall time at the enable pin.

`fall_slew` specifies the fall slew at the enable pin.

Note: The `fall_time` and `fall_slew` are optional.

Example:

```
inst_name E VCC_NORMAL VCC_SWITCHED 5.50E-09 2.90E-10 5.50E-09 2.90E-10
inst_name E VCC_NORMAL VCC_SWITCHED 5.93E-09 7.23E-10 5.93E-09 7.23E-10
```


© 2013-2023 All Rights Reserved.

29

GDS2DEF Utility

- Overview on page 113
- Data Requirements for DEF Generation on page 115
- ESD Analysis using convert_gds_to_def on page 116
- Generating a DEF File on page 117

Overview

`convert_gds_to_def` is a GDS2DEF utility that converts a given GDSII file to a DEF file for power/rail signoff analysis. For some cells that only have a GDS format, it is not possible to show the design data inside the cells in Voltus GUI. In addition, it is not feasible to generate a desired DEF using the existing translators of `Str2oa` and `oa2def` because of missing

© 2013-2023 All Rights Reserved.

connectivity information. Thus, it is necessary to have a GDS2DEF utility to generate a desired DEF file for rail analysis.

This utility enables you to perform rail analysis when the DEF file is not accessible. For example, certain data, such as the redistribution layer (RDL) flip-chip bump layer descriptions, is only available in the GDSII format. This type of data must be converted to the DEF/LEF format to be integrated into the DB data.

Cell-Level and Full-Chip Level Implementation

The utility can convert both cell-level GDS and full-chip level GDS to DEF. The generated DEF contains all the geometries for the P/G routing, the specified P/G pin information, component and logical connectivity information.
Signal routing in not considered in this utility, that is, the signal nets will be filtered out in the

© 2013-2023 All Rights Reserved.

DEF file. As an illustration of comparison between the new and original DEFs shown in the following figure, the signal nets visible in the image on the right side are not present on the left.

DEF by GDS2DEF utility Original DEF

Data Requirements for DEF Generation

The table below lists the inputs that are required for the `convert_gds_to_def` utility:

Input	Description
GDSII file	Top cell name in the GDSII file is the design name in the generated DEF.
Layer mapping file	The layer mapping file between GDS and tech layers.
Net pin information	This file contains a list of coordinates and layer names as the starting point to trace.

© 2013-2023
All Rights Reserved.

Tech and Macro information	For cells that need to be treated as an instance in the generated DEF and do not require to extract the P/G routing details, the Macro LEF information of these cells have to be provided. The utility traces the net connectivity using pin shape information in LEF and does not check the geometry inside the cell in GDS. If no LEF macros are provided, the utility flattens all the cell instances. Tech file is also required for the utility to get the LEF layers. The .cl library contains the tech information.

ESD Analysis using convert_gds_to_def

You can enable Electrostatic Discharge Analysis (ESD) analysis for macros through the `convert_gds_to_def` utility. The following parameters of the `convert_gds_to_def` utility can be used to identify the bounding box of the ESD cells in the GDS file:

- `-esd_marker_layer_file <filename>` - specifies a file that includes a list of ESD cell/instance names and their corresponding GDS marker layers that represent the bounding box of the ESD cells. The format of the file is:

  ```
  <cellname> <marker layer> [top layer] [bottom layer]
  ```

- `-esd_cell_bbox_file <filename>` - specifies a file that includes the bounding box for each ESD cell. The format of the file is:

  ```
  <esd inst name> <x1 y1 x2 y2> [top layer] [bottom layer]
  ```

  You can use the optional top layer and bottom layer arguments to specify the top and bottom technology layers of the ESD cell pin shape. This provides you with the flexibility to specify the lowest layers to be used for creating pins in the LEF.

When either of these parameters are specified, the software identifies the bounding box of the ESD cells in the GDS file, and collects all the GDS shapes inside the bounding box on layers between the specified top layer and bottom layer and considers them as ESD cell LEF pin shapes. If the top and bottom layers are not specified in these files, the shapes inside the bounding box of all layers are considered as ESD cell shapes.

`convert_gds_to_def` utility generates a LEF file for each ESD cell to enable PGV generation. The output DEF includes the user-defined nets and the ESD instances based on the specified marker layer or bounding box.

© 2013-2023 All Rights Reserved.

Generating a DEF File

The use model of the `convert_gds_to_def` utility is given below:

```
convert_gds_to_def
  [-help]
  [-black_box_cell_file filename]
  [-gds_file filename]
  [-gds_layermap filename]
  [-inst_name_file filename]
  [-net_location_file filename]
  [-output_components {ALL | NONE | CONN}]
  [-output_def_file filename]
  [-output_lef_macros filename]
  [-power_grid_library libraryname]
  [-top_cell cellname]
  [-use_gds_label {ALL | TOP | <tech_layer_name> | <cell_name>:<tech_layer_name> |
<cell_name>:ALL}]
  [-verilog_file filename]
  [-spef_file filename]
  [-esd_marker_layer_file filename]
  [-esd_cell_bbox_file filename]
```

The following command converts the specified GDS file `new.gds` to a DEF file `top.def`.

```
convert_gds_to_def -net_location_file net.loc -gds_file new.gds -
power_grid_library ../pv_library.cl -gds_layermap lmap -top_cell cella -
output_def_file top.def
```

For detailed information on the command parameters, refer to the `convert_gds_to_def` command in the *Voltus Text Command Reference*.

© 2013-2023 All Rights Reserved.

© 2013-2023 All Rights Reserved.

30

Customizing GDS Layermap and XTC Command File for TRIM Metals Layers

The purpose of the CUT/TRIM layer is to cut the metal layers so as to separate the different supplies. The following diagram illustrates how the CUT layers separate the power and ground supplies:

To ensure that Voltus is able to identify the CUT layers, you need to manually modify these layers in the GDS layermap file. After modifying the layermap file, you need to customize the XTC command file for supporting these layers during power-grid library generation.

This chapter gives examples for simple (without CUT layers) and advanced (with CUT layers) GDS layermap files, and the corresponding XTC command files. It gives sample XTC command files for stop@via layer 0, 1, and 2.

Simple GDS Layer Map File

```
metal    M0    gds    180  250
via      VIA0  gds    159  250
metal    M1    gds    31   250
via      VIA1  gds    51   250
metal    M2    gds    32   250
via      VIA2  gds    52   250
metal    M3    gds    33   270
```

Simple XTC Command File

```
CONNECT    M0    VIA0    M1    VIA1    M2    VIA2    M3
```


© 2013-2023 All Rights Reserved.

Actual Layer Map with Mask Layers (Not Supported in Voltus)

```
metal    M0      gds     180  250
metal    M0      gds     180  255
metal    M0      gds     180  256
metal    M0      gds     202  180  port  textonly
metal    M0      gds     180  109  =>Mask Metal Layer
metal    M0      gds     180  110  =>Mask Metal Layer
metal    M0      gds     180  112  =>Mask Metal Layer
via      VIA0    gds     159  250
metal    M1      gds     31   250
metal    M1      gds     31   255
metal    M1      gds     31   256
metal    M1      gds     20   231  port  textonly
metal    M1      gds     31   109  =>Mask Metal Layer
metal    M1      gds     31   110  =>Mask Metal Layer
via      VIA1    gds     51   250
via      VIA1     gds     51    255
via      VIA1     gds     51    256
metal    M2       gds     32    250
metal    M2       gds     32    255
metal    M2       gds     32    256
metal    M2       gds     202   32  port  textonly
metal    M2       gds     32    110  =>Mask Metal Layer
via      VIA2     gds     52    250
via      VIA2     gds     52    255
via      VIA2     gds     52    256
metal    M3       gds     33    270
metal    M3       gds     33    275
metal    M3       gds     33    276
metal    M3       gds     20    233  port  textonly
via      VIA3     gds     53    330
metal    M4       gds     54    330
metal    M4       gds     303   34   port  textonly
```

Modified Layer Map with Renamed Metal (M0, M1, M2) and Mask Layers

```
metal    M0      gds     180  250
metal    M0_0    gds     180  255
metal    M0_1    gds     180  256
metal    M0      gds     202  180  port  textonly
metal    CM0_0   gds     180  109  =>Mask Metal Layer
```


© 2013-2023 All Rights Reserved.

```
metal     CM0_1    gds     180  110   =>Mask Metal Layer
metal     CM0_2    gds     180  112   =>Mask Metal Layer
via       VIA0     gds     159  250
metal     M1       gds     31   250
metal     M1_0     gds     31   255
metal     M1_1     gds     31   256
metal     M1       gds     20   231  port  textonly
metal     CM1_0    gds     31   109   =>Mask Metal Layer
metal     CM1_1    gds     31   110   =>Mask Metal Layer
via       VIA1     gds     51   250
via       VIA1     gds     51    255
via       VIA1     gds     51    256
metal     M2       gds     32    250
metal     M2_0     gds     32    255
metal     M2_1     gds     32    256
metal     M2       gds     202   32  port  textonly
metal     CM2      gds     32    110   =>Mask Metal Layer
via       VIA2     gds     52    250
via       VIA2     gds     52    255
via       VIA2     gds     52    256
metal     M3       gds     33    270
metal     M3       gds     33    275
metal     M3       gds     33    276
metal     M3       gds     20    233  port  textonly
via       VIA3     gds     53    330
metal     M4       gds     54    330
metal     M4       gds     303   34   port  textonly
```

XTC Command File with STOP@VIA0

```
COPY    M0_0      M0
COPY    M0_1      M0
AND     M0_0      CM0_0      CM0_0_phy
AND     M0_1      CM0_1      CM0_1_phy
NOT     M0        CM0_0_phy  M0_0_phy
NOT     M0_0_phy  CM0_1_phy  M0_1_phy
DELETE  M0
COPY    M0_1_phy  M0
COPY    M1_0      M1
COPY    M1_1      M1
AND     M1_0      CM1_0      CM1_0_phy
```


© 2013-2023 All Rights Reserved.

```
AND     M1_1      CM1_1      CM1_1_phy
NOT     M1        CM1_0_phy  M1_0_phy
NOT     M1_0_phy  CM1_1_phy  M1_OUT
DELETE  M1
COPY    M1_OUT    M1
COPY    M2_0      M2
COPY    M2_1      M2
AND     M2_1      CM2        CM2_phy
NOT     M2        CM2_phy    M2_phy
DELETE  M2
COPY    M2_phy    M2
CONNECT     M0    VIA0    M1    VIA1    M2    VIA2    M3
```

XTC Command File with STOP@VIA1

```
COPY    M1_0      M1
COPY   M1_1      M1
AND    M1_0      CM1_0     CM1_0_phy
AND    M1_1      CM1_1     CM1_1_phy
NOT    M1        CM1_0_phy M1_0_phy
NOT    M1_0_phy  CM1_1_phy M1_OUT
DELETE M1
COPY   M1_OUT    M1
COPY   M2_0      M2
COPY   M2_1      M2
AND    M2_1      CM2       CM2_phy
NOT    M2        CM2_phy   M2_phy
DELETE M2
COPY   M2_phy    M2
CONNECT     M1    VIA1    M2    VIA2    M3
```

XTC Command File with STOP@VIA2

```
COPY    M2_0     M2
COPY    M2_1     M2
AND     M2_1     CM2         CM2_phy
NOT     M2       CM2_phy     M2_phy
DELETE  M2
COPY    M2_phy   M2
CONNECT     M2     VIA2     M3
```


© 2013-2023 All Rights Reserved.

31

Glossary

AC

AC is an abbreviation for "alternating current," which is current that reverses direction at a regular rate. The current flow in signal wires is generally AC.

abstract power-grid view

An abstract power-grid view is a simplified form of the detailed power-grid view in which a block's or macro's internal current loading has been replaced by a mathematically equivalent impedance value assigned to the block's or macro's port pins.

activity

Number of signal transitions per clock period.

angled resistor

An angled resistor is any resistor that is not horizontal or vertical.

ASCII format

ASCII format is a human-readable format called American Standard Code for Information Interchange.

average current

Average current is the asymptotic average current flowing through each tap point.

© 2013-2023 All Rights Reserved.

block

A block describes a higher-level cell that can contain other blocks, cells, or transistors.

cell

A cell is a group of transistors that implements a primitive logic function. Typical examples of cells are NAND gates, NOR gates, XOR gates, and flip-flops. Typically, a cell does not contain placements of other cells.

chemical mechanical polishing (CMP)

Chemical mechanical polishing (CMP) is the process of removing surface irregularities and obtaining uniform planarization of a silicon wafer surface during the manufacturing of semiconductor devices below 0.35 microns.

cladding

Cladding is a barrier layer between the copper interconnect and low-k dielectric material to protect the interconnect from chemical reactions with the dielectric material.

clock signal jitter

Clock signal jitter is the variations in the delay, skew, and slew rates caused by supply voltage fluctuations.

clock skew

Clock skew is the difference between the time required for the clock signal to travel through the longest path from the clock origin to a latch and the time required for the clock signal to travel through the shortest path from the clock origin to a latch.

clock slew

Clock slew is the time required for the signal voltage at a buffer or receiver to rise from a lower voltage threshold to a higher voltage threshold (rising slew), or the reverse when measuring the falling slew rate. Typically, the lower voltage threshold is defined as 10 percent of the power supply voltage, and the higher voltage threshold is defined as 90 percent of the power supply voltage.

© 2013-2023 All Rights Reserved.

conductor layer

A conductor layer is a layer of material that carries electric current in an integrated circuit.

core

A core is a block that contains the main functionality of a design.

DC

DC is an abbreviation for "direct current," which is current that flows in only one direction. The current in a power grid is generally DC. DC is also a standard representation of a waveform that is typically used as a voltage source in circuit simulation. It displays a flat waveform.

decoupled capacitance

Decoupled capacitance is lumped capacitance that is used in place of coupled capacitance, typically to support static timing analysis.

decoupling capacitance

Added capacitance used to smooth the power rail. It is usually between power and ground.

decoupling cells

Decoupling cells are cells containing decoupling capacitance.

delay

Delay is the time that it takes a signal to propagate from a given input to a given output.

detailed power-grid view

A detailed power-grid view is a detailed electrical model of a cell's or macro's internal power routing extracted from the GDSII data. It provides resistance and current loading information.

© 2013-2023 All Rights Reserved.

dielectric layer

The dielectric layer is any layer deposited below the highest passivation layer.

diffusion layer

The diffusion layer is any layer that is below the surface of the substrate. It includes active areas, such as n-channel and p-channel areas, but not polysilicon, which is a conductor layer.

dishing

Dishing is the reduction of a cross-sectional area of wider interconnect lines because of chemical mechanical polishing (CMP). Dishing occurs when the copper recedes below or protrudes above the level of the adjacent dielectric. It is limited to a single wire.

DSPF

DSPF is an acronym for Detailed Standard Parasitic Format, which is a form of SPF that enables the file-based transfer of parasitic resistance and capacitance from physical design tools to timing analysis and other downstream analysis tools. DSPF cannot represent true coupling capacitance but models it as decoupled to ground.

duty cycle

The input duty cycle is a value between 0 and 1, representing the fraction of time that the signal is 1.

dynamic analysis

Dynamic analysis is a method of analyzing a circuit to obtain operating currents and voltages. It is time-based, analyzing the circuit netlist over a specified period, such as a clock cycle.

effective current source model (ECSM)

The effective current source model (ECSM) is a highly accurate, non-linear driver model used for delay calculation.

© 2013-2023 All Rights Reserved.

effective instance voltage (EIV)

The rail-to-rail voltage obtained during rail analysis. EIV is obtained by processing the voltage waveforms and finding the worse-case effective voltage between the power and ground pins during a specific window. EIV is measured over all the timing/switching windows (default), or the entire analysis (elapse). There are four possible windows to evaluate EIV:

- `switching` – any switching activity (current is non-leakage) reported in power analysis
- `timing` – timing window information from TWF or embedded timing engine
- `both` (Default) – both switching or timing window
- `elapse` – entire rail simulation, and not filtered to a specific window

The EIV windows are illustrated in the following diagram:

You can specify EIV windows using the following command:

```
set_rail_analysis_mode -eiv_eval_window { switching | timing | both | elapse}
```

effective instance voltage (EIV) method

There are different methods of computing EIV during rail analysis: `worst` (default), `best`, `average`, or `worst-average`.

Using the following, you can write out the best, worst, or average effective instance voltage between power and ground nets:

```
set_rail_analysis_mode  -eiv_method { worst best avg worstavg}
```


© 2013-2023 All Rights Reserved.

The `worstavg` EIV method computes the average instance voltage during each evaluation window defined by the `-eiv_eval_window` parameter, and reports the worst average instance voltage out of all the windows, as shown in the following diagram:

In this case:

- `Best`: Best drop/bounce inside the window
- `Worst` (default): Worst drop/bounce value inside the window
- `Average`: Average drop/bounce for all the windows
- `Worst-average`: Average of each window, and then report the worst average

If `-eiv_method worstavg` and `-eiv_eval_window elapse` (i.e. 1 evaluation window) are specified, then this method will report average voltage across all time steps in simulation, as shown in the following diagram:

In this case:

© 2013-2023
All Rights Reserved.

- `Best`: Best drop/bounce for the entire simulation
- `Worst` (default): Worst drop/bounce value for the entire simulation
- `Average`: Average drop/bounce for the entire simulation
- `Worst-average`: same as average since there is only 1 window (whole simulation)

electromigration

Electromigration is the mass transport of metal resulting from the transfer of momentum between conducting electrons and diffusing metal atoms. It occurs in metal wires that conduct high current densities. Conducting electrons collide with diffusing metal atoms and impel the atoms in the direction of electron flow. This collision produces a mass flux opposite to that of the current. Divergences in this mass flux can result in damage to the conductor in the form of voids or hillocks. This problem can be minimized by limiting the current density in metal conductor lines, vias, and contacts.

embedded block

An embedded block is a large custom block created for a specific design.

erosion

Erosion is the reduction of a cross-sectional area of interconnect lines and a localized thinning of the dielectric.

feedthrough

A feedthrough is a signal net within a cell that may be connected to the top-level routing at multiple locations simultaneously. It is commonly used in cells intended to be assembled by abutment to form a continuous net passing through the cells. The feedthrough may or may not connect to the circuitry in the cell.

In extraction, parasitics are not normally computed for cell ports, so long wires created by abutted feedthroughs can be modeled incorrectly if the resistance of the feedthroughs are not extracted.

© 2013-2023 All Rights Reserved.

The following figure shows a feedthrough in a single cell and a feedthrough in four abutting cell instances.

filler cell

A filler cell is a standard cell that only fills space rather than implementing any logical functionality. Filler cells are used to fill gaps in the power and ground rails in flows where the power grid does not include them or to act as placeholders in case functionality must be inserted at a late stage in the design process. They often include the ties that connect the wells to power or ground. Many flows require them to be inserted at intervals to make sure that the wells are tapped.

flat extraction

Flat extraction is the process of extracting the interconnect parasitic resistance and capacitance down to the transistor level.

flip chip

Flip chip refers to Direct Chip Attach (DCA), a type of microelectronic assembly in which an integrated circuit die is attached to the package by eutectic solder bumps of electrically conducting material placed on the chip bond pads across the top layer (called a redistribution layer, or RDL) of the chip. The chip is flipped face down on the substrate or board and the bumps attached to the package with reflow soldering. This type of packaging is an alternative to traditional packaging that uses perimeter bonding wires to attach the chip to the package. Flip-chip packaging offers improved signal integrity to processors, DSPs, SRAMs, chip sets, and other types of integrated circuits.

fusing

Fusing is the instant failure of a wire segment because of excessive instantaneous current. This failure mechanism is related to electromigration and is sometimes classified as a subset of Joule heating.

© 2013-2023 All Rights Reserved.

GDSII

GDSII is a binary file in graphical design system version 2.0 Stream format.

ground

Ground is a conducting body used as the electrical current return. It is also referred to as VSS or GND.

ground bounce

Ground bounce is an increase in voltage that occurs on the ground network of an integrated circuit because of the current flowing through the resistance of the ground network itself (V = I x R).

hierarchical design

A hierarchical design is a way of structuring logic at more than one level of abstraction; that is, a symbol at one level of abstraction represents the internal contents of a cell at a higher level of hierarchy.

hold time

The hold time is the period after the clocking edge for which the input data must be held constant.

hold-time violation

A hold-time violation is a change of the input signal earlier than the specified hold time.

horizontal resistor

A horizontal resistor is any resistor with an angle smaller than 30 degrees with respect to the X axis.

HSPICE

HSPICE is a specific commercial implementation of the SPICE simulator that has become an industry-standard tool for performing circuit simulation. It was developed by Meta Software.

© 2013-2023 All Rights Reserved.

input duty cycle

The input duty cycle is a value between 0 and 1, representing the fraction of time that the signal is 1.

interconnect

Interconnect refers to the physical wires and vias between the gates and cells in an integrated circuit.

ipeak analysis

Ipeak analysis is a static power-grid analysis based on peak transistor saturation currents.

IR drop

IR drop is a reduction in voltage that occurs on the VDD network of an integrated circuit because of the current flowing through the resistance of the VDD network itself (V = I x R).

Itaputil

Itaputil is a Cadence utility that either reports characteristics of tap current files or extracts portions of the data and places them into a new tap current file.

Joule heating

Joule heating is a cause of electromigration in which a specific segment of a wire becomes excessively heated because of high currents.

LEF/DEF

LEF/DEF is an acronym for Library Exchange Format/Design Exchange Format, which is a layout format widely used by place-and-route tools. LEF format stores the physical, timing, power, and related information of the cells used in a design. It can contain definitions at the polygon level or a pointer to a GDSII Stream file for data in the cell. DEF format contains all the instantiations of the cells and wiring, as determined by the router, and the port names and pin locations on the cells. LEF/DEF databases include node and instance names that match those of the schematic and gate-level netlists.

© 2013-2023 All Rights Reserved.

macro

A macro is a cell referenced in the LEF file. The geometric data for this cell is stored in either DEF or GDSII. If the cell is stored in GDSII, it may contain references to subcells.

Miller capacitance

Miller capacitance is coupling capacitance between nets associated with coupling between the gate and the source/drain of a transistor.

Moore's Law

Moore's Law refers to an observation made by Gordon Moore, former CEO of Intel, that the number of transistors on silicon integrated circuits has approximately doubled once every 18 months since 1962, the year in which the technology was invented.

netlist

A netlist is an ASCII or binary description of all the components in a design and their interconnections.

noise injection

Noise injection is the appearance of unexpected voltage values on a net resulting from transitioning nets or circuitry and coupling through interconnect or substrate parasitics.

optical proximity correction (OPC)

Optical proximity correction is the process of automatically generating mask features that compensate for subwavelength lithography effects.

panning

Panning is the process of obtaining a panoramic view of the design by selecting one point as the center of the view without changing the scale of the design.

© 2013-2023 All Rights Reserved.

parasitic capacitance

Parasitic capacitance is any capacitance causing unwanted effects between conductors or in devices.

passivation layer

The passivation layer is any layer deposited above the highest conductor of the process.

peak current

Peak current is the maximum value of a current waveform.

piecewise linear (PWL)

Piecewise linear is a standard representation of a waveform that is typically used as a voltage source in circuit simulation. The waveform displays no recurring pattern.

planar dielectric

Planar dielectric refers to a flat dielectric layer, as opposed to a conformal dielectric layer. See also "conformal dielectric."

port

A port is a named geometry that marks a location in certain output files. It can be a single point, marked by a text label on a special layer. It can be a collection of polygons on a special layer marked by text labels on that special layer. It can be a fully extracted node in the cell marked as a port by a text label on a special layer.

port power-grid view

A port power-grid view is a simple electrical view of a cell or macro containing no detail about the internal circuitry and power grid. Creation of the port power-grid view does not require the use of extraction.

power

Power is the voltage supply. It is also referred to as VDD or VCC.

© 2013-2023 All Rights Reserved.

power grid

A power grid is a large collection of interconnected wires on a chip that distributes power from the pins to all the gates and transistors on the chip. It is similar in function to a power distribution system that delivers power from a power station to residential and commercial facilities.

power-grid signoff (PGS)

Power-grid signoff (PGS) is the process of requiring a chip to pass static power-grid analysis before tapeout.

power-grid view

A power-grid view is a model of a cell's or block's power grid that contains power-port information providing the current loading to the power-grid networks.

promoted geometries

Promoted geometries are geometries within a cell or block that must be considered at the full-chip level to obtain accurate results when the power grid is extracted.

pulse

Pulse is a standard representation of a waveform that is typically used as a voltage source in circuit simulation. The waveform displays a recurring pattern.

setup time

The setup time is the period before the clocking edge for which the data must be held constant.

setup-time violation

A setup-time violation is a change of the input signal later than the specified setup time.

signal skew

Signal skew is the difference in signal arrival times across a design.

© 2013-2023 All Rights Reserved.

simultaneously switching output (SSO)

To switch the outputs at the same time. This is done to see what effect it has on the power grid.

skin effect

A skin effect is the concentration of a time-varying current near the surface of a conductor as opposed to a steady-state current flowing through a conductor that is distributed uniformly throughout the cross-section of the conductor. The penetration depth of the current is called the skin depth and is inversely proportional to the frequency of the time-varying current.

slotting

Slotting is the insertion of small holes, or slots, at regular intervals in wide wires to reduce the effect of dishing on wide wires. These slots insert a form of hard "posts" in the wire so that the CMP process removes less copper. This technique reduces the dishing, as well as the effective resistivity.

SPEF

SPEF is an acronym for Standard Parasitic Exchange Format, which is a form of SPF that allows true coupling capacitance to be carried between nodes and passed to downstream analysis tools.

SPICE

SPICE (Simulation Program with Integrated Circuit Emphasis) is a simulation program originally developed at UC Berkeley as a transistor-level circuit simulator. SPICE is now often used as a generic term for any transistor-level circuit simulator.

static analysis

Static analysis is a method of analyzing a circuit without using time-based simulation.

steady-state analysis

Steady-state analysis is an analysis of a system that assumes that the components have steady-state values.

© 2013-2023 All Rights Reserved.

Synopsys design constraints (SDC) file

The Synopsys design constraints (SDC) file is a set of data based on the tool command language (TCL) that specifies the timing and area constraints for a design. It can be submitted to Synopsys' static timing analysis tools for timing analysis.

substrate

The substrate is the supporting material on or within which an integrated circuit is fabricated or to which an integrated circuit is attached.

tap currents

Tap currents are currents arising from the connection of devices such as transistors to the power grid.

tapeout

Tapeout is the process of placing the final chip layout design files on media for delivery to a mask-making vendor.

toggle count format (TCF) file

The toggle count format file contains switching activity information that includes the toggle count information and the probability of the net or pin being in the logic 1 state.

transient analysis

Transient analysis is a time-based analysis of a system.

transition density

Number of signal transitions (up and down) per second.

transition time

The transition time is the time interval during which a signal switches from low to high or high to low.

© 2013-2023 All Rights Reserved.

value change dump (VCD) file

The value change dump (VCD) file is a standard Verilog file described in OVI documents and IEEE 1364 that contains the test vectors and switching activity information for all the nets in the design.

VDD

VDD is the common name of the power node, which is the voltage supply. It is the same as VCC.

vertical resistor

A vertical resistor is any resistor with an angle smaller than 30 degrees with respect to the Y axis.

via

A via is a connection point between two adjacent metal routing layers defined by a pad in each layer.

VSS

VSS is the common name of the ground node, which is a conducting body used as the electrical current return. It is the same as GND.

wire-edge enlargement

Wire-edge enlargement is the effective change to a wire's width and spacing because of its proximity to neighboring wires. This is an optical effect.

zooming

Zooming is the process of magnifying or shrinking the view of a selected area of the chip.

© 2013-2023 All Rights Reserved.