

StarRC™

User Guide and Command Reference

Version L-2016.06, June 2016

SYNOPSYS®

Copyright Notice and Proprietary Information

©2016 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

About This Manual	xxx
Customer Support	xxxi

Part I: StarRC User Guide

1. Introduction to StarRC

StarRC Features	1-2
StarRC Licensing	1-4
Tiered Licensing Checkout Policy	1-4
License Queuing	1-5
StarRC Documentation	1-6
StarRC Formal Messages	1-6
Computing Environment Warning Messages	1-7
StarRC Message Man Pages	1-7
Error Message Control	1-8

2. Running StarRC

StarRC Overview	2-2
The nxtgrd File	2-3
The StarXtract Command	2-4
The StarRC Command File	2-6
Distributed Processing	2-6
Manual Submission of Distributed Processing Jobs	2-7

Automatic Submission of Distributed Processing Jobs	2-7
LSF System	2-8
Oracle Grid Engine	2-8
General Network With a List of Machines	2-9
Runtime Design Automation System	2-9
Distributed Processing Reports	2-9
3. Design Databases	
Introduction to Physical Databases	3-2
The IC Compiler II NDM Database Flow	3-3
The IC Compiler Milkyway Database Flow	3-4
Antenna Diodes in Milkyway Designs	3-5
The LEF/DEF Database Flow	3-6
Merging Library GDSII Files in LEF/DEF Designs	3-7
Translation of Routing DEF Blockages	3-8
The Calibre Connectivity Interface Flow	3-9
Procedure Without an LVS Extraction Report	3-10
Procedure Using an LVS Extraction Report	3-11
Error Conditions in StarRC-Calibre Flows	3-12
Cross-Referenced Extraction in the Calibre Flow	3-13
Calibre Support of LVS Black Box Flows	3-13
The Hercules LVS Tool Flow	3-16
GDSII to XTR View Translation in Hercules Flows	3-17
Cross-Referenced Extraction in the Hercules Flow	3-18
Placement Information for the HSIM Reliability Flow	3-19
The IC Validator LVS Tool Flow	3-21
Steps in the IC Validator Extraction Flow	3-22
Examples of Script Files	3-22
Cross-Referenced Extraction in the IC Validator Tool	3-23
Error Conditions	3-23
Cross-Referencing in Transistor-Level Flows	3-24
XREF: NO	3-25
XREF: YES	3-25
XREF: COMPLETE	3-28
Cross-Referencing By Device Property	3-29

How the XREF Command Affects SPF Netlists.....	3-31
XREF: NO.....	3-31
XREF: YES.....	3-31
XREF: COMPLETE	3-32
Parameterized Cells	3-33
How StarRC Layer-Based Rules Affect Parameterized Cells	3-33
How LVS Tools Handle Parameterized Cells.....	3-33
Single Layout Device With No Container Cell	3-34
Multiple Layout Devices With No Container Cell	3-35
Layout Devices in a Schematic Container Cell	3-37
Extracting Parameterized Cells	3-38
Gray Box Handling	3-38
The IGNORE_CAPACITANCE Command	3-38
Extracting Coupling Capacitances	3-38
Retaining Coupling Capacitance Between Top and Skip Cell Levels.....	3-39
SKIP_PCELLS Netlist Behavior.....	3-40
Metal Fill	3-41
Emulated Metal Fill	3-42
Real Metal Fill	3-43
Handling Coupling Capacitance on Floating Metal Fills	3-44
Specifying Metal Fill in the Design Database	3-45
The Metal Fill Reuse Flow	3-46
4. StarRC Extraction and Output	
Simultaneous Multicorner Extraction	4-2
The Corners File	4-3
Operating Temperatures and Corner Types	4-4
Mapping Files	4-4
Via Coverage Option Files	4-4
The SELECTED_CORNERS Command.....	4-4
Simultaneous Multicorner Flow Examples.....	4-5
Corner Constraints.....	4-6
Writing Formulas to the Netlist.....	4-6
Writing Parasitic Resistor Temperature Coefficients to the Netlist.....	4-7
The Galaxy Parasitic Database	4-8
Using the Galaxy Parasitic Database	4-9
Creating a Netlist from a Galaxy Parasitic Database	4-10

The GPD Configuration File	4-10
Output Netlists	4-12
SPEF Netlist Example	4-12
SPF Netlist Example	4-13
NETNAME Netlist Example	4-14
NETLIST_IDEAL_SPICE_FILE Output Example	4-15
Netlists For HSIM Reliability Analysis	4-18
Creating the Simplified Command File	4-18
SPF Geometry Visualization in HSIM	4-19
Extraction For Electromigration Analysis	4-20
5. ECO Extraction	
ECO Extraction Overview	5-2
The Galaxy Standard ECO Flow	5-3
Identification of Nets Affected by an ECO	5-3
Galaxy Standard ECO Flow With Full-Chip Update Timing	5-4
The Galaxy Incremental ECO Flow	5-6
Galaxy Incremental ECO Flow With Incremental Update Timing	5-7
Galaxy Incremental ECO Flow With Full-Chip Update Timing	5-9
Changing the Working Directory Between ECO Iterations	5-11
6. The StarRC Field Solver	
Overview of Field Solver Extraction	6-2
Capacitance Types	6-2
Boundary Conditions	6-2
Conductor Types	6-3
Nets	6-3
Net Groups	6-3
Ground Nets	6-4
Fill Nets	6-4
Running the Field Solver	6-5
Output Files for the FSCOMPARE Flow	6-5
Controlling Field Solver Accuracy and Runtime	6-6
Specifying Convergence Goals	6-7
Specifying the Accuracy Goal	6-8

Specifying the Consistency of Results.....	6-8
Specifying Pattern Matching for Symmetric Nets.....	6-9
Distributed Processing for Field Solver Jobs	6-9
Setting up Distributed Processing	6-10
LSF System	6-10
Oracle Grid Engine.....	6-10
General Network With a List of Machines.....	6-11
Runtime Design Automation System	6-11
7. The Graphical User Interface	
Invoking the Graphical User Interface	7-2
Using the GUI to Run a Timing or Noise Analysis	7-2
Using the GUI to Run a SingleShot Analysis	7-5
StarRC Graphical User Interface Reference.....	7-6
Control Window	7-6
File Menu	7-7
Setup Menu	7-8
Timing Menu.....	7-11
Noise Menu.....	7-11
Viewer Menu.....	7-12
Entering Information in the StarRC GUI	7-13
Entry Fields	7-13
Editing a Mapping File in the StarRC GUI.....	7-15
8. Using StarRC With the Virtuoso Custom Design Platform	
Introduction to Virtuoso Integration.....	8-2
Setting Up Virtuoso Integration.....	8-3
Installation and Licensing	8-3
Environment Variables.....	8-4
Virtuoso Integration Flow Configuration and Related Files.....	8-5
The Device Mapping File.....	8-6
The Layer Mapping File.....	8-7
Customizing the LVS Run	8-10
Customizing the StarRC Extraction Run	8-10

View Generation	8-11
Net and Instance Name Conventions	8-11
Port and Terminal Connectivity Characteristics	8-13
Instance Property Annotation from the Schematic View	8-14
Controlling Instance Property Annotation	8-14
Property Mapping	8-16
Instance Name Matching Rule	8-17
Subnode Marker and Parasitic Device Visualization	8-18
User-Defined Callbacks	8-20
Pre-Extraction Callback	8-20
View Preprocessing Callback	8-21
View Postprocessing Callback	8-22
Instance Creation Callback	8-22
Callback Flow Example	8-24
StarRC Parasitic Generation Cockpit GUI	8-25
Populating the Cockpit Fields Automatically	8-28
Advanced Save and Load Mode	8-30
Functions in the StarRC Parasitic View Generation Dialog Box	8-30
Run Cockpit Tab	8-31
Device Extraction Tab	8-32
Extract Parasitics Tab	8-36
Output Parasitics Tab	8-38
Load Sharing Facility Job Submission	8-40
Selecting Nets for Reporting	8-42
Selecting and Customizing the Analysis Options	8-46
StarRC OA View Creation	8-48
The OpenAccess Flow	8-48
Linking OpenAccess Libraries	8-48
Linking StarRC OpenAccess Libraries	8-49
Support for Special StarRC Flows	8-49
Skip Cell Mapping	8-49
OpenAccess File Examples	8-50
OpenAccess Library Definition	8-50
OpenAccess Mapping Files	8-51
StarRC Commands for OpenAccess Parasitic Views	8-52
Parasitic Probing	8-53

StarRC Parasitic Prober	8-53
StarRC Parasitic Browser	8-56
StarRC Parasitic Netlist Browser	8-56
View Selection	8-57
Dynamic Flylines for Probing	8-57
Point-to-Point Resistance Probing	8-58
Double Highlighting of Point-to-Point Resistance Probe Results	8-58
Specifying and Saving the Probe Options	8-60
Highlighting or Blinking Probe Results	8-61
Probing a Single Bus Bit or All Bus Bits	8-61
Displaying Multiple Nets	8-63
Parasitic View Probing	8-63
Schematic View Probing	8-63
Probed Results Log and Cross-Probing	8-64
Prober File Input and Output	8-65
Schematic Annotation	8-65
Opens Debugger	8-67
Virtuoso Integration SKILL Procedures and Related Variables	8-71
GUI Integration with a Custom Interface	8-71
Batch Mode Procedures	8-72
RCGenParaViewBatch	8-72
RCCockpitRun	8-77
9. Transistor-Level Extraction	
Preparing Hercules Runsets	9-2
Runset Rules	9-2
Required Commands	9-4
Runset Example	9-6
Marker Generation in Hercules	9-10
Example of Text-Based Markers	9-10
Example of ID-Layer Markers	9-11
Example of Connection-Based Markers	9-11
Preparing IC Validator Runsets	9-12
Runset Rules	9-12
Required Functions	9-15
Hierarchy Options	9-16
Hierarchy Options Syntax	9-16

Runset Example	9-17
IC Validator Marker Generation	9-19
Text-Based Marker Layers	9-19
Multifinger Device Support.	9-21
Preparing Calibre Rule Files	9-21
Rule File Creation	9-21
Required Commands	9-24
Support for Calibre Preprocessor Directives and Include Statements	9-24
Rule File Example	9-25
Preparing the Mapping File	9-28
Mapping Rules	9-28
Mapping File Example	9-30
Running the Calibre Query Server for Output to StarRC	9-31
Stripping X-Cards in Source Names	9-33
Multifinger Device Support in the Calibre Connectivity Interface	9-34
Optional Layout Netlisting Query Commands	9-34
The Push Down Separate Properties (PDSP) Flow	9-34
The Push Down Back-Annotation (PDBA) Flow	9-35
Error Conditions for the PDSP and PDBA Flows	9-36
Preparing the StarRC Command File	9-37
Commands for Hercules Flows	9-37
Commands for IC Validator Flows	9-37
Commands for Calibre Connectivity Interface Flows	9-38
Other StarRC Commands	9-38
Interconnect Technology Format File	9-40
Preparing the ITF File	9-40
Limitations	9-40
ITF File Example	9-41
Transistor-Level Extraction Limitations	9-42
10. Special-Purpose Features	
Clock Net Inductance Extraction	10-2
Setting Up Clock Net Inductance Extraction	10-3
Feedthrough Nets	10-4
Feedthrough - First Issue.	10-5

Port Renaming	10-5
Feedthrough - Second Issue	10-6
Runset Requirements	10-6
Via Coverage	10-7
Determining the Coverage and Landing Areas for Rectangular Vias	10-8
Determining the Coverage and Landing Areas for Square Vias	10-10
Positive and Negative Check	10-10
Examples	10-11
Via Coverage Examples	10-14
Reading the Output Report	10-15
Long Ports	10-17

11. Hercules GENDEV Device Extraction and Netlist Generation

Introduction	11-2
Device Recognition and Syntax	11-3
Scheme Extraction Functions	11-5
Hercules LVS With GENDEV Devices	11-6
Scheme Property Comparison Functions	11-8
GENDEV Netlist Generation in StarRC	11-9

Part II: Process Modeling

12. Process Modeling Methodology

Flows for Process Characterization	12-2
The grdgexo Flow	12-2
The Direct ITF Flow	12-3
The Interconnect Technology Format File	12-3
Creating an ITF File	12-4
ITF Files for Transistor-Level Flows	12-6
ITF File Example	12-6
The Mapping File	12-7
The grdgexo Command	12-9
Syntax of the grdgexo Command	12-9

Partially Encrypting ITF Information	12-10
Incremental grdgenxo runs	12-11
Using the -res_update Option to Update an nxtgrd File	12-12
Generating TLUPlus Models	12-13
13. Process Characterization	
FinFET Modeling and Extraction	13-3
FinFET ITF Statement Guidelines	13-4
FinFET Capacitances	13-6
Through-Silicon Via Extraction	13-8
Microbump Modeling	13-9
3D-IC Flow with Cell-Defined TSVs and microbumps	13-9
GDS Flow	13-9
LEF/DEF and MilkyWay Flow	13-9
3D-IC Flow with Via-Defined TSVs and microbumps	13-10
GDS Flow	13-10
LEF/DEF and MilkyWay Flow	13-10
Double or Multiple Patterning Technology	13-11
Extraction for Double-Patterning Processes	13-11
Estimating Parasitics for Critical Nets Using the Precolor Flow	13-12
Conductor Layer Thickness Variation	13-12
Single-Box Method	13-13
Multiple-Box Method	13-14
Calculation of Effective Density	13-14
Width and Spacing-Dependent Thickness Variation	13-16
Bottom Conductor Thickness Variation	13-16
Conductor Sheet Zones	13-17
Tall Contact Via Modeling	13-20
Bridge Via Modeling	13-21
Gate-To-Diffusion Capacitance Extraction	13-22
Conformal Dielectrics	13-23
Conductor Cutting Through Dielectric	13-24
Covertical Conductors	13-25

Conductor Drop Factor	13-25
Drop Factor Error Conditions.....	13-27
Double-Polysilicon Process	13-29
Layer Etch.....	13-30
Spacing- and Width-Dependent Etch.....	13-30
Determining WMIN and SMIN Values	13-31
Overlapping Wells.....	13-31
Damage Modeling.....	13-33
Temperature Derating.....	13-34
Half-Node Scaling.....	13-34
Via Merging.....	13-36
Grouping Vias in an L-Shaped Array.....	13-38
Asymmetric Via Arrays	13-40
Rectilinear Via Arrays	13-43
Diffusion Resistance	13-44
User-Defined Diffusion Resistance.....	13-46

14. ITF Examples

Fully Planar Process ITF Example	14-2
Conformal Dielectric Process ITF Example	14-3
Gate Poly Process ITF Example.....	14-4
Local Interconnect Process ITF Example.....	14-5
Layer Etch Process ITF Example	14-6
Emulated Metal Fill Process ITF Example	14-7
Transistor-Level Process ITF Example.....	14-8
Through-Silicon Via Process ITF Example	14-9
Trench Contact Process ITF Example	14-11

Part III: StarRC Command Reference

15. StarRC Commands

3D_IC	15-2
3D_IC_FILTER_DEVICE	15-3
3D_IC_FLOATING_SUBSTRATE.....	15-4
3D_IC_SUBCKT_FILE	15-5
3D_IC_TSV_COUPLING_EXTRACTION.....	15-7
ANALOG_SYMMETRIC_NETS	15-9
AUTO_RUNSET	15-10
BLOCK	15-12
BLOCK_BOUNDARY	15-14
BUS_BIT	15-16
CALIBRE_LVS_DEVICE_TYPE_CAP	15-17
CALIBRE_LVS_DEVICE_TYPE_MOS.....	15-18
CALIBRE_LVS_DEVICE_TYPE_RES	15-19
CALIBRE_OPTIONAL_DEVICE_PIN_FILE	15-20
CALIBRE_PDBA_FILE	15-21
CALIBRE_QUERY_FILE	15-23
CALIBRE_RUNSET	15-25
CAPACITOR_TAIL_COMMENTS.....	15-26
CASE_SENSITIVE	15-28
CELL_TYPE	15-29
CHECK_SKIP_PCELL_LAYER_NAMES	15-30
CLOCK_NET_FREQUENCY	15-31
CLOCK_NET_INDUCTANCE.....	15-32
CLOCK_NET_INDUCTANCE_LAYERS.....	15-33
COMPARE_DIRECTORY.....	15-34
CONLY_NETS	15-35
CONVERT_DIODE_TO_PARASITIC_CAP	15-36

CONVERT_WARNING_TO_ERROR	15-37
CORNERS_FILE	15-38
CORNER_TYPE	15-40
COUPLE_NONCRITICAL_NETS	15-41
COUPLE_NONCRITICAL_NETS_PREFIX	15-43
COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX	15-44
COUPLE_TO_GROUND	15-45
COUPLE_TO_PCELL_PINS	15-47
COUPLING_ABS_THRESHOLD	15-48
COUPLING_MULTIPLIER	15-49
COUPLING_REL_THRESHOLD	15-50
COUPLING_REPORT_FILE	15-51
COUPLING_REPORT_NUMBER	15-52
COUPLING_THRESHOLD_OPERATION	15-53
DEBUG_MILKYWAY_DATABASE	15-54
DEF_ATTRIBUTE_FROM_LEF	15-57
DEF_USE_PINS	15-59
DENSITY_BASED_THICKNESS	15-60
DENSITY_OUTSIDE_BLOCK	15-61
DETECT_FUSE	15-62
DIELECTRIC_FILL_GDS_FILE	15-65
DIELECTRIC_FILL_GDS_LAYER_MAP_FILE	15-66
DIFFUSION_RES_MODE	15-69
DPT	15-71
DPT_COLOR_GDS_FILE	15-72
DPT_COLOR_GDS_LAYER_MAP_FILE	15-73
ECO_MODE	15-74
ECO_RESTORE_COMMAND	15-76

ECO_SAVE_COMMAND	15-77
EVACCESS_DIRECTORY	15-78
EXTRA_GEOMETRY_INFO	15-79
EXTRACTION	15-81
EXTRACT_RES_BODY_COUPLING	15-82
EXTRACT_VIA_CAPS	15-83
FSCOMPARE_COUPLING_RATIO	15-84
FSCOMPARE_COUPLING_THRESHOLD	15-85
FSCOMPARE_FILE_PREFIX	15-86
FSCOMPARE_OPTIONS	15-87
FSCOMPARE_THRESHOLD	15-91
FS_BOUNDARY_BOX	15-92
FS_BOUNDARY_CONDITION	15-94
FS_DP_STRING	15-96
FS_EXTRACT_NETS	15-98
FS_IGNORE_GATE_CHANNEL_CAPACITANCE	15-99
GDS_FILE	15-101
GDS_LAYER_MAP_FILE	15-102
GPD	15-105
GPD_DP_STRING	15-106
HIERARCHICAL_COUPLING_ABS_THRESHOLD	15-108
HIERARCHICAL_COUPLING_REPORT_NUMBER	15-109
HIERARCHICAL_COUPLING_REPORT_FILE	15-110
HIERARCHICAL_COUPLING_REL_THRESHOLD	15-112
HIERARCHICAL_SEPARATOR	15-113
HN_NETLIST_MODEL_NAME	15-114
HN_NETLIST_SPICE_TYPE	15-115
ICV_ANNOTATION_FILE	15-117

ICV_LVS_DEVICE_TYPE_CAP	15-119
ICV_LVS_DEVICE_TYPE_MOS	15-120
ICV_LVS_DEVICE_TYPE_RES	15-121
ICV_OPTIONAL_DEVICE_PIN_FILE	15-122
ICV_RUNSET_REPORT_FILE	15-123
IGNORE_CAPACITANCE	15-124
IGNORE_FIELDPOLY_DIFFUSION_COUPLING	15-128
IGNORE_GATE_CHANNEL_CAPACITANCE	15-129
IGNORE_SHARED莫斯_TERMINAL_CAP	15-130
INSTANCE_PORT	15-131
INSTANCE_PORT_OPEN_CONDUCTANCE	15-134
INTRANET_CAPS	15-135
ITF_FILE	15-136
KEEP_SUBCONT_MODELS	15-137
KEEP_VIA_NODES	15-138
LEF_FILE	15-139
LEF_USE_OBS	15-141
LPE_DEVICES	15-142
LPE_FLAGS_SETTING	15-143
LPE_PARAM	15-144
LVS_EXTRACTION_REPORT_FILE	15-146
MACRO_DEF_FILE	15-147
MAGNIFICATION_FACTOR	15-148
MAGNIFY_DEVICE_PARAMS	15-149
MAPPING_FILE	15-150
MARKER_GENERATION	15-151
MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER	15-152
MERGE_INSTANCE_PORTS	15-153

MERGE_PARALLEL_DEVICES	15-155
MERGE_VIAS_IN_ARRAY	15-157
MESSAGE_SUPPRESSION	15-159
MESSAGE_SUPPRESSION_LIMIT	15-161
METAL_FILL_BLOCK_NAME	15-162
METAL_FILL_GDS_BLOCK	15-163
METAL_FILL_GDS_FILE	15-164
METAL_FILL_GDS_FILE_NET_NAME	15-165
METAL_FILL_GDS_MAG	15-166
METAL_FILL_GDS_OFFSET	15-167
METAL_FILL_OASIS_FILE	15-168
METAL_FILL_OASIS_FILE_NET_NAME	15-169
METAL_FILL_OASIS_MAG	15-170
METAL_FILL_OASIS_OFFSET	15-171
METAL_FILL_POLYGON_HANDLING	15-172
METAL_SHEET_OVER_AREA	15-174
MILKYWAY_ADDITIONAL_VIEWS	15-175
MILKYWAY_CELL_VIEW	15-176
MILKYWAY_DATABASE	15-177
MILKYWAY_EXPAND_HIERARCHICAL_CELLS	15-178
MILKYWAY_EXTRACT_VIEW	15-179
MILKYWAY_REF_LIB_MODE	15-180
MILKYWAY_SHOW_CELL_INFO_DETAIL	15-182
MODE	15-183
MODEL_TYPE	15-184
MOS_GATE_CAPACITANCE	15-186
MOS_GATE_DELTA_RESISTANCE	15-187
MOS_GATE_DELTA_RESISTANCE_LAYERS	15-189

MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE	15-191
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE.....	15-193
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE	15-194
MULTIGATE_MODELS.....	15-195
NDM_DATABASE.....	15-196
NDM DESIGN VIEW.....	15-197
NDM_EXPAND_HIERARCHICAL_CELLS.....	15-199
NDM_SEARCH_PATH	15-200
NET_SEGMENT_CUT_LENGTH.....	15-201
NET_TYPE	15-203
NETLIST_CAPACITANCE_UNIT	15-204
NETLIST_COMMENTED_PARAMS.....	15-205
NETLIST_COMMENTS_FILE	15-206
NETLIST_COMPRESS	15-207
NETLIST_COMPRESS_COMMAND	15-208
NETLIST_CONNECT_OPENS.....	15-210
NETLIST_CONNECT_SECTION	15-211
NETLIST_COUPLE_UNSELECTED_NETS	15-212
NETLIST_DELIMITER	15-213
NETLIST_DEVICE_LOCATION_ORIENTATION	15-214
NETLIST_ECO_FILE	15-215
NETLIST_FILE	15-216
NETLIST_FORMAT	15-217
NETLIST_GROUND_NODE_NAME	15-219
NETLIST_HIER_PROBE_NODES.....	15-220
NETLIST_IDEAL_SPICE_FILE	15-221
NETLIST_IDEAL_SPICE_HIER	15-222
NETLIST_IDEAL_SPICE_TYPE	15-223

NETLIST_INCREMENTAL	15-224
NETLIST_INPUT_DRIVERS	15-225
NETLIST_INSTANCE_SECTION	15-226
NETLIST_LOGICAL_TYPE	15-228
NETLIST_MAX_LINE	15-229
NETLIST_MERGE_SHORTED_PORTS	15-230
NETLIST_MINCAP_THRESHOLD	15-231
NETLIST_MINRES_HANDLING	15-232
NETLIST_MINRES_THRESHOLD	15-233
NETLIST_NAME_MAP	15-234
NETLIST_NODE_SECTION	15-235
NETLIST_NODENAME_NETNAME	15-236
NETLIST_PARASITIC_RESISTOR_MODEL	15-238
NETLIST_PASSIVE_PARAMS	15-240
NETLIST_POSTPROCESS_COMMAND	15-242
NETLIST_POWER_FILE	15-244
NETLIST_PRECISION	15-245
NETLIST_PRINT_CC_TWICE	15-247
NETLIST_REMOVE_DANGLING_BRANCHES	15-249
NETLIST_RENAME_PORTS	15-250
NETLIST_RESISTANCE_UNIT	15-253
NETLIST_SELECT_NETS	15-254
NETLIST_SIM_OPTIONS	15-255
NETLIST_SMC_FORMULA	15-256
NETLIST_SUBCKT	15-257
NETLIST_TAIL_COMMENTS	15-258
NETLIST_TIME_UNIT	15-260
NETLIST_TOTALCAP_THRESHOLD	15-261

NETLIST_TYPE	15-262
NETLIST_UNSCALED_COORDINATES	15-263
NETLIST_USE_M_FACTOR	15-264
NETS	15-265
NETS_FILE	15-268
NONCRITICAL_COUPLING_REPORT_FILE	15-269
NUM_CORES	15-271
OA_BUS_BIT	15-272
OA_CDLOUT_RUNDIR	15-273
OA_CELL_NAME	15-274
OA_DEVICE_MAPPING_FILE	15-275
OA_LAYER_MAPPING_FILE	15-276
OA_LIB_DEF	15-277
OA_LIB_NAME	15-278
OA_MARKER_SIZE	15-279
OA_OVERWRITE_LOCKED_VIEW	15-280
OA_PORT_ANNOTATION_VIEW	15-281
OA_PROPERTY_ANNOTATION_VIEW	15-282
OA_PROPMAP_CASE_SENSITIVE	15-283
OA_REMOVE_DUPLICATE_PORTS	15-284
OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX	15-285
OA_REMOVE_SPICECARD_PREFIX	15-286
OA_SKIPCELL_MAPPING_FILE	15-287
OA_VIEW_NAME	15-288
OASIS_FILE	15-289
OASIS_LAYER_MAP_FILE	15-290
OBSERVATION_POINTS	15-293
OPERATING_FREQUENCY	15-294

OPERATING_TEMPERATURE	15-295
PIN_CUT_THRESHOLD	15-297
PIO_FILE	15-299
PLACEMENT_INFO_FILE	15-300
PLACEMENT_INFO_FILE_NAME	15-303
POWER_EXTRACT	15-304
POWER_NETS.....	15-307
POWER_PORTS	15-308
POWER_REDUCTION.....	15-309
PRINT_SILICON_INFO	15-310
PROBE_TEXT_FILE	15-312
REDUCTION	15-314
REDUCTION_MAX_DELAY_ERROR	15-315
REFERENCE_DIRECTION	15-316
REMOVE_DANGLING_NETS	15-317
REMOVE_DIFFUSION_GATE_OVERLAP	15-318
REMOVE_FLOATING_NETS.....	15-319
REMOVE_METAL_FILL_OVERLAP	15-320
REMOVE_NET_PROPERTY	15-321
REPORT_METAL_FILL_STATISTICS	15-322
REPORT_SMIN_VIOLATION.....	15-323
RES_UPDATE_FILE	15-325
RETAIN_CAPACITANCE_CAP_MODELS	15-327
RETAIN_GATE_CONTACT_COUPLING	15-329
RING_AROUND_THE_BLOCK	15-331
RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER	15-332
SELECTED_CORNERS	15-333
SHEET_COUPLE_TO_NET.....	15-334

SHEET_COUPLE_TO_NET_LEVEL	15-335
SHORT_EQUIV_NODES	15-336
SHORT_PINS.....	15-337
SHORT_PINS_IN_CELLS	15-341
SIMULTANEOUS_MULTI_CORNER	15-342
SKIP_CELL_AGF_FILE	15-344
SKIP_CELL_PORT_PROP_FILE.....	15-345
SKIP_CELLS	15-347
SKIP_CELLS_COUPLE_TO_NET.....	15-349
SKIP_CELLS_COUPLE_TO_NET_LEVEL	15-350
SKIP_CELLS_FILE.....	15-351
SKIP_INSTANCES	15-352
SKIP_PCELLS	15-353
SKIP_PCELL_LAYERS_FILE	15-355
SLEEP_TIME_AFTER_FINISH	15-360
SPICE_SUBCKT_FILE.....	15-361
STAR_DIRECTORY	15-362
STARRC_DP_STRING.....	15-363
STOP_EXTRACTION_ON_NUMEROUS_SHORTS	15-365
SUBSTRATE_EXTRACTION	15-366
SUMMARY_FILE	15-368
SUPPORT_DIFFERENT_PORTNAME_NETNAME	15-369
TARGET_PWRA.....	15-371
TCAD_GRD_FILE	15-374
TEMPERATURE_SENSITIVITY.....	15-375
TOP_DEF_FILE	15-376
TRANSLATE_DEF_BLOCKAGE	15-377
TRANSLATE_DRCFILL_AS_OBS	15-378

TRANSLATE_FLOATING_AS_FILL	15-379
TRANSLATE_NDM_BLOCKAGE	15-380
TRANSLATE_RETAIN_BULK_LAYERS	15-381
TRANSLATE_VIA_PINS	15-385
TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO	15-386
TSV_CELLS	15-387
USER_DEFINED_DIFFUSION_RES	15-388
VIA_COVERAGE	15-389
VIA_COVERAGE_OPTION_FILE	15-391
WIDE_DEVICE_TERM_RESISTANCE	15-396
XREF	15-398
XREF_FEEDTHRU_NETS	15-400
XREF_LAYOUT_INST_PREFIX	15-401
XREF_LAYOUT_NET_PREFIX	15-402
XREF_SWAP_MOS_SD_PROPERTY	15-403
XREF_USE_LAYOUT_DEVICE_NAME	15-404
ZONE_COUPLE_TO_NET	15-405
ZONE_COUPLE_TO_NET_LEVEL	15-406

16. ITF Statements

AREA	16-3
ASSOCIATED_CONDUCTOR	16-4
BACKGROUND_ER	16-6
BOTTOM_DIELECTRIC_ER	16-7
BOTTOM_DIELECTRIC_THICKNESS	16-8
BOTTOM_THICKNESS_VS_SI_WIDTH	16-11
BW_T	16-14
CAPACITIVE_ONLYETCH	16-16

CONDUCTOR.....	16-17
CRT_VS_AREA	16-20
CRT_VS_SI_WIDTH.....	16-22
CRT1, CRT2, and T0	16-24
DAMAGE_ER.....	16-26
DAMAGE_THICKNESS.....	16-27
DENSITY_BOX_WEIGHTING_FACTOR.....	16-28
DEVICE_TYPE.....	16-29
DIELECTRIC	16-31
DIELECTRIC_FILL_EMULATION_VS_SI_SPACING	16-33
DIELECTRIC_FILL_VS_SI_SPACING.....	16-35
DPT_MAX_SHIFT	16-38
DROP_FACTOR.....	16-39
DROP_FACTOR_LATERAL_SPACING	16-40
ER.....	16-41
ER_TABLE	16-42
ER_VS_SI_SPACING	16-43
ETCH	16-45
ETCH_VS_CONTACT_AND_GATE_SPACINGS.....	16-46
ETCH_VS_WIDTH_AND_LENGTH.....	16-51
ETCH_VS_WIDTH_AND_SPACING	16-54
EXTENSIONMIN	16-62
FILL_RATIO	16-63
FILL_SPACING.....	16-64
FILL_TYPE	16-65
FILL_WIDTH.....	16-66
FROM	16-67
GATE_TO_CONTACT_SMIN	16-68

GATE_TO_DIFFUSION_ADJUSTMENT_CAP	16-69
GATE_TO_DIFFUSION_CAP	16-70
GATE_TO_DIFFUSION_CHANNEL_CAP	16-73
GLOBAL_TEMPERATURE	16-75
HALF_NODE_SCALE_FACTOR	16-76
ILD_VS_WIDTH_AND_SPACING	16-78
IS_CONFORMAL	16-80
IS_PLANAR	16-82
LATERAL_CAP_SCALING_VS_SI_SPACING	16-83
LAYER_TYPE	16-86
LINKED_TO	16-88
MEASURED_FROM	16-89
MULTIGATE	16-91
POLYNOMIAL_BASED_THICKNESS_VARIATION	16-96
PROCESS_FOUNDRY	16-103
PROCESS_NODE	16-104
PROCESS_TYPE	16-105
PROCESS_VERSION	16-106
PROCESS_CORNER	16-107
RAISED_DIFFUSIONETCH	16-108
RAISED_DIFFUSIONETCH_TABLE	16-111
RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER	16-112
RAISED_DIFFUSION_THICKNESS	16-114
RAISED_DIFFUSION_TO_GATE_SMIN	16-116
RAISED_DIFFUSION_TO_GATE_SMIN_TABLE	16-117
REFERENCE_DIRECTION	16-118
RESISTIVE_ONLYETCH	16-119
RHO	16-120

RHO_VS_SI_WIDTH_AND_THICKNESS	16-121
RHO_VS_WIDTH_AND_SPACING	16-123
RPSQ	16-124
RPSQ_VS_SI_WIDTH	16-125
RPSQ_VS_SI_WIDTH_AND_LENGTH	16-127
RPSQ_VS_WIDTH_AND_SPACING	16-130
RPV	16-132
RPV_VS_AREA	16-133
RPV_VS_COVERAGE	16-136
RPV_VS_SI_COVERAGE	16-141
RPV_VS_SI_WIDTH_AND_LENGTH	16-144
RPV_VS_WIDTH_AND_LENGTH	16-148
SIDE_TANGENT	16-151
SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING	16-153
SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING	16-156
SMIN	16-157
SPACER_ER_VS_WIDTH_AND_SPACING	16-158
SW_T	16-160
TECHNOLOGY	16-161
THICKNESS	16-162
THICKNESS_VS_DENSITY	16-163
THICKNESS_VS_WIDTH_AND_SPACING	16-165
TO	16-167
TSV	16-168
TW_T	16-173
USER_DEFINED_DIFFUSION_RESISTANCE	16-174
USE_SI_DENSITY	16-178
VIA	16-179

WMIN	16-182
------------	--------

17. Mapping File Commands

conducting_layers.....	17-2
color_layers	17-6
via_layers	17-8
marker_layers.....	17-11
remove_layers	17-12
viewonly_layers	17-13
ignore_cap_layers	17-14

Preface

This preface includes the following sections:

- [About This Manual](#)
- [Customer Support](#)

About This Manual

This user guide describes how to use the StarRC tool to perform parasitic extraction and to use the grdgenxo tool to model advanced processes.

Audience

This manual is intended for circuit design and layout engineers who need to analyze the effects of parasitic capacitance and resistance on advanced circuit designs.

Related Publications

For additional information about the StarRC tool, see the documentation on the Synopsys SolvNet® online support site at the following address:

<https://solvnet.synopsys.com/DocsOnWeb>

You might also want to see the documentation for the following related Synopsys products:

- PrimeTime® Suite
- IC Compiler™
- IC Compiler II
- Custom Designer
- IC Validator
- Hercules™
- Raphael™

Release Notes

Information about new features, enhancements, changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *StarRC Release Notes* on the SolvNet site.

To see the *StarRC Release Notes*,

1. Go to the SolvNet Download Center located at the following address:
<https://solvnet.synopsys.com/DownloadCenter>
2. Select StarRC, then select a release in the list that appears.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates syntax, such as <code>write_file</code> .
<i>Courier italic</i>	Indicates a user-defined value in syntax, such as <code>write_file design_list</code> .
Courier bold	Indicates user input—text you type verbatim—in examples, such as <code>prompt> write_file top</code>
[]	Denotes optional arguments in syntax, such as <code>write_file [-format fmt]</code>
...	Indicates that arguments can be repeated as many times as needed, such as <code>pin1 pin2 ... pinN</code>
	Indicates a choice among alternatives, such as <code>low medium high</code>
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

The SolvNet site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNet site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNet site, go to the following address:

<https://solvnet.synopsys.com>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNet site, click HELP in the top-right menu bar.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a support case to your local support center online by signing in to the SolvNet site at <https://solvnet.synopsys.com>, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
 - E-mail support_center@synopsys.com from within North America.
 - Find other local support center e-mail addresses at
<http://www.synopsys.com/Support/GlobalSupportCenters/Pages>
- Telephone your local support center.
 - Call (800) 245-8005 from within North America.
 - Find other local support center telephone numbers at
<http://www.synopsys.com/Support/GlobalSupportCenters/Pages>

Part I: StarRC User Guide

1

Introduction to StarRC

The StarRC tool extracts parasitics such as resistors, capacitors, and inductors from databases that represent integrated circuit layout designs. You can use the StarRC tool to generate netlists for many types of analysis, such as timing, noise, and electromigration.

This chapter contains the following sections:

- [StarRC Features](#)
- [StarRC Licensing](#)
- [StarRC Documentation](#)

StarRC Features

The StarRC full-chip parasitic extraction tool is part of the Synopsys Galaxy™ sign-off analysis suite. StarRC provides a flexible framework that efficiently and accurately covers the entire extraction spectrum, whether analyzing a full chip containing millions of transistors or examining a few critical nets in three-dimensional field solver mode. StarRC works with a variety of input data formats and provides many options for customizing the output.

Some of the features of the StarRC tool are as follows:

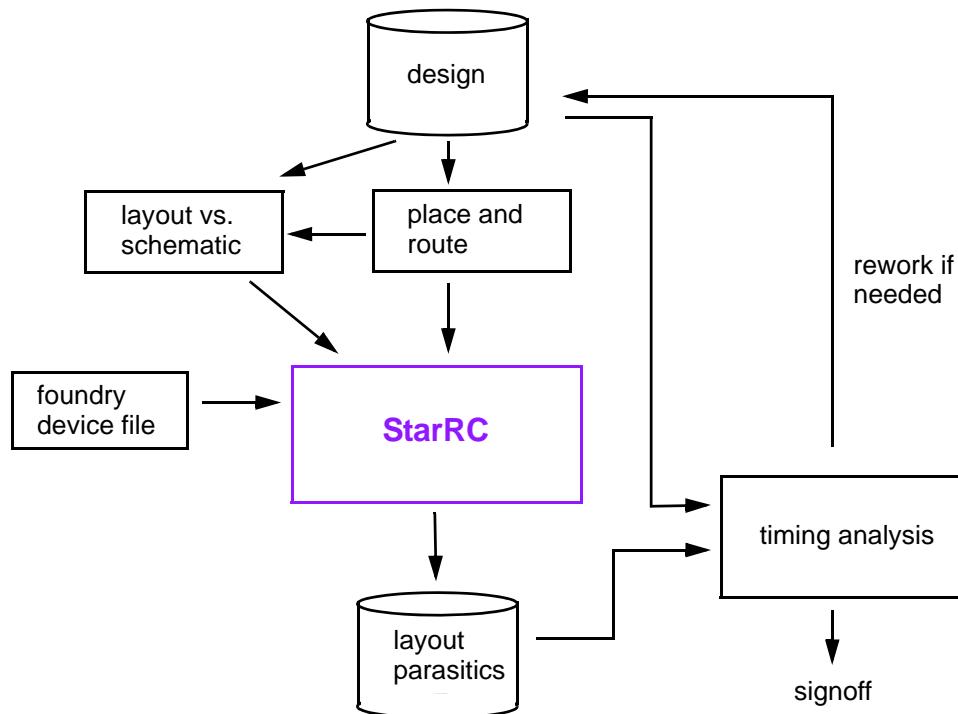
- Full-chip parasitic extraction for use with noise, electromigration, and timing verification tools
- Integrated field solver for accurate three-dimensional analysis of selected nets
- Ability to use characterization files provided by major foundries for process development kits
- Both gate-level and transistor-level extraction
- Both flat and hierarchical extraction
- Interoperability with many Synopsys and third-party layout-versus-extraction tools, custom layout tools, timing analysis tools, and simulation tools
- Ability to analyze early-stage designs that have opens, shorts, and other design rule violations

StarRC Ultra is an optional add-on product that offers advanced features, including:

- Efficient distributed processing and netlisting techniques to optimize runtime and memory usage for very large designs
- A fast Engineering Change Order flow that streamlines extraction by analyzing only those nets that have been changed from the reference design
- Analysis of FinFET and other advanced transistor designs
- Extraction for double or multiple patterning processes
- Ability to create customized characterization files using comprehensive process modeling for interconnect layers
- Simultaneous multicorner analysis, which saves runtime by examining related corners in parallel

Figure 1-1 shows how StarRC extraction is used in a typical design flow.

Figure 1-1 StarRC in the Design Flow



After a design has been completed and laid out, the circuit timing must be tested. Accurate timing analysis requires that all of the parasitic resistances and capacitances resulting from the manufacturing process are taken into account. The StarRC tool uses the chip layout along with the process description (usually obtained from a foundry) to extract millions of parasitic devices. If the design must be modified to fix timing violations, the extracted parasitics must also be updated.

Extracted parasitics are also important for other simulation tools such as circuit simulators and electromigration analysis tools.

StarRC Licensing

Three types of license keys govern StarRC tool use: StarRC, Custom, and Ultra.

Two options of the `StarXtract` command affect license usage:

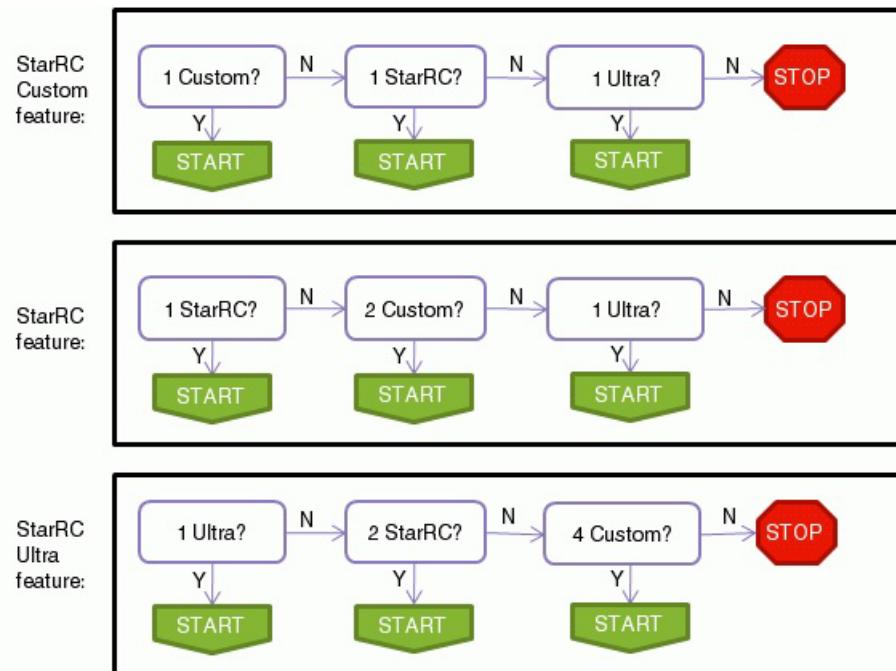
- The `-custom` option is designed for customers who have only one Custom license in their license server. With this option, multicore processing and upper-tier commands are not allowed.
- The `-ultra` option allows you to check out only an Ultra license key. By specifying this option, you can wait until an Ultra license is available, rather than checking out two StarRC licenses or four Custom licenses.

Tiered Licensing Checkout Policy

When all three types of license keys are available in the license pool, multiple lower-tier keys can be used to run upper-tier features, as shown in [Figure 1-2](#). For example, two StarRC keys or four Custom license keys can run a job with an Ultra feature. Similarly, two Custom license keys can run one job with a StarRC feature.

If the `-custom` option is specified, then StarRC runs only with an available Custom license. If the `-ultra` option is specified, then StarRC runs only with an available Ultra license.

Figure 1-2 Tiered Licensing Checkout Policy Without License Queuing



License Queuing

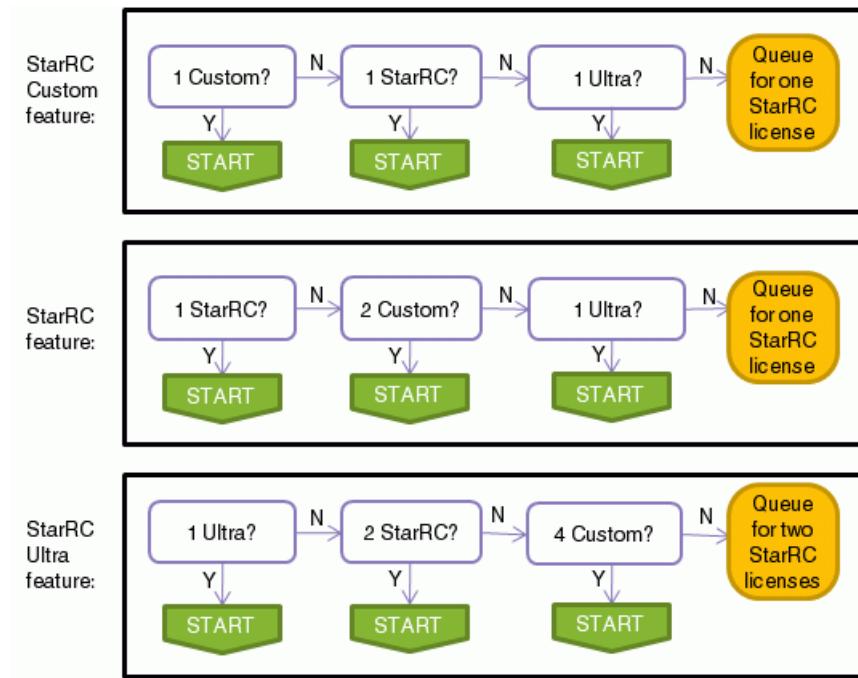
To queue a StarRC job when the appropriate licenses are not available, you can enable license queuing by setting the `STARRC_LICENSE_WAIT` environment variable as follows:

```
% setenv STARRC_LICENSE_WAIT yes
```

Figure 1-3 shows the tiered licensing checkout procedure with license queuing. When the appropriate licenses are not available, the StarRC tool queues the job according to the StarXtract command options and the features that are used:

- If the `-custom` option is specified, the job queues for one Custom license.
- If the `-ultra` option is specified, the job queues for one Ultra license.
- If neither option is specified, but the job uses an Ultra feature, the job queues for two StarRC licenses; if no Ultra feature is used, the job queues for one StarRC license.

Figure 1-3 Tiered Licensing Checkout Policy With License Queuing



License Queuing Daemon

To use StarRC license queuing, specify the license server as `port@host` or use the full path to the license file (e.g. a format such as `/top/Star-install/sys1/license_file`). Download and install the 10.9.2 (or higher) version of the `snpsslmd` combined vendor daemon.

StarRC Documentation

If you need help, information is available from the following resources:

- The *StarRC User Guide and Command Reference*
 - Message man pages displayed with the `starrc_man` command
 - SolvNet articles
-

StarRC Formal Messages

StarRC issues formal messages when a condition arises that requires user attention. The severity levels are as follows:

- Information: No action required if the condition is acceptable

For example, the SX-1429 message states that pin information from a DEF macro is used because there is no corresponding LEF macro. This scenario might be expected and acceptable, and the tool has enough information to proceed with extraction.

However, you should check informational messages to ensure that you understand their meaning.

- Warning: Unexpected condition, but execution does not stop

For example, the SX-1505 message notifies you that a layer specified in a skip cell layers file is not found in the design database. This scenario might be expected and acceptable, and the tool has enough information to proceed with extraction. However, the message might also indicate an oversight in the design or command file that should be investigated. Only you can determine if the condition is acceptable.

- Error: Serious condition, likely to be undesirable, but does not stop execution

For example, the SX-2780 message describes an issue with a Milkyway library.

- Severe: Serious condition, but execution continues until the current operation is complete to allow the reporting of all similar errors

For example, the SX-1837 message states that a database via layer is not mapped to an ITF via layer. Extraction cannot proceed, but the tool completes the analysis of the layer mapping file to find all similar errors before terminating the run.

- Fatal: Serious condition that halts execution immediately

For example, the SX-1748 message indicates that the schematic top block name is not specified. The extraction run stops immediately.

Computing Environment Warning Messages

If the StarRC tool detects computing environment issues, the tool issues an SX-3564 warning message, which is included in the standard summary file. In addition, a file named sml.sum is created that contains detailed SML warning messages for the conditions shown in [Table 2](#). For more information, see the warning message man pages.

Table 2 Warning Messages Related to the Computing Environment

Condition	Message ID	Description
Low memory availability	SML-001	System memory usage is over 98 percent
High CPU usage	SML-002	Host CPU load is over 95 percent, although the Synopsys tool process is using only 5 percent or less of the CPU capacity
Network congestion issue	SML-003	Packet retransmission rate is over 5 percent, indicating network congestion
Network connectivity issue	SML-004	Packet loss is over 5 percent, indicating poor network connectivity
High network latency	SML-005	Latency is over 300 ms from the host running the process to hosts specified in the message

StarRC Message Man Pages

Man pages for formal messages provide information that supplements the message that you see in your log file or interactive session. Each man page contains the text of the actual message, a description of the condition, and some suggestions for followup actions.

Use the `starrc_man` command at the operating system prompt to view a man page. For example,

```
% starrc_man SX-1748
```

The `starrc_man` command is installed in the same directory that contains the StarXtract executable. If you install the StarRC tool using the standard installation procedure, which includes setting the PATH environment variable, the `starrc_man` command is also installed.

Error Message Control

You can control how error messages are issued with the following StarRC commands:

- The `CONVERT_WARNING_TO_ERROR` command allows you to choose to halt extraction when StarRC encounters specified warning conditions.
- The `MESSAGE_SUPPRESSION` command limits the number of times that specific messages are issued. The limit applies only to messages whose IDs are listed in the command.
- The `MESSAGE_SUPPRESSION_LIMIT` command applies a limit to all warning, error, and informational messages.

In the following example, StarRC stops reporting most messages after the tenth occurrence, but reports EX-269 messages up to twenty times:

```
MESSAGE_SUPPRESSION_LIMIT: 10  
MESSAGE_SUPPRESSION: EX-269:20
```

2

Running StarRC

This chapter provides basic information about running StarRC.

This chapter contains the following sections:

- [StarRC Overview](#)
- [The StarXtract Command](#)
- [Distributed Processing](#)

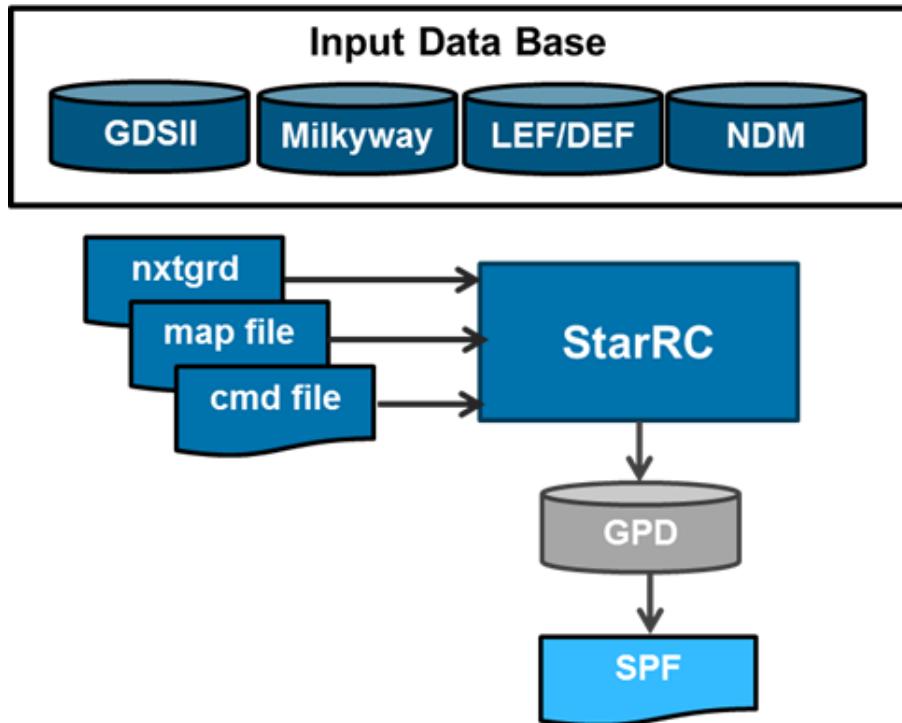
StarRC Overview

The StarRC tool analyzes the parasitic capacitances and resistances of advanced circuit designs. The tool uses pattern matching to analyze a design by comparing the layout to a set of primitive structures whose parasitics have been previously simulated. The reference parasitics are contained in a process characterization database called the nxtgrd file.

- StarRC accepts Milkyway designs, LEF/DEF designs, and design libraries created by the IC Compiler II tool in the New Data Model (NDM) format.
- For transistor-level extraction, StarRC can read output from the IC Validator, Hercules, and Mentor Calibre layout-versus-schematic (LVS) tools.
- The tool uses foundry-created nxtgrd files to represent the effects of the manufacturing process.
- StarRC saves the extracted parasitics in the binary Galaxy Parasitics Database (GPD). You can optionally create netlist files for use by other tools.

[Figure 2-1](#) illustrates the general StarRC flow.

Figure 2-1 StarRC Usage Flowchart



You can use StarRC in different ways to accomplish different goals. The primary StarRC flows are as follows:

- Gate-level (cell-level) extraction

In a gate-level flow, the design is composed of cells (macros). Gate-level extraction is commonly used to analyze the entire chip for timing analysis in a signoff loop. By default, cells are treated as skipped cells (or skip cells), which means that the capacitance of nets inside the cells is not analyzed.

- Transistor-level extraction

The transistor-level flow focuses on smaller portions of the design in greater detail. You can examine parasitics at the device level. Information from a layout-versus-schematic (LVS) tool is required for this flow.

- Special-purpose flows

- Clock net inductance analysis
- Field solver analysis

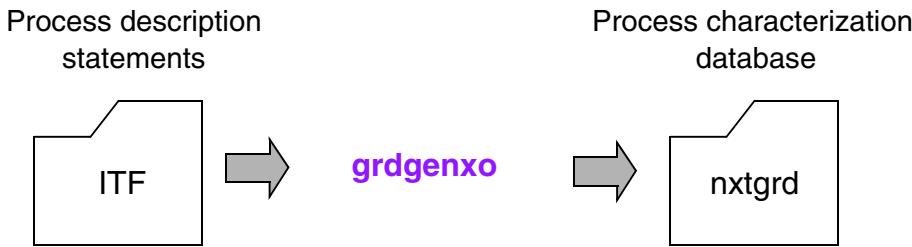
Some StarRC features or commands are restricted to a specific flow, as stated in the command reference pages or in feature descriptions within this user guide.

The nxtgrd File

The nxtgrd file contains the reference parasitics for the manufacturing process of the chip. Using an nxtgrd file obtained from a foundry is the most accurate way of analyzing the parasitics of a specific process technology. The nxtgrd file is specified in the StarRC command file with the `TCAD_GRD_FILE` command.

For process exploration, you can change some of the process parameters and create your own nxtgrd files. However, the results might not correlate with manufactured devices unless you use an nxtgrd file created specifically for the manufacturing process.

The nxtgrd file is created by the grdgexo tool, a StarRC utility program, as illustrated in [Figure 2-2](#). The grdgexo tool generates the nxtgrd file from a file written in a process description language called the Interconnect Technology Format (ITF). The grdgexo tool uses an internal field solver operating on an extensive set of primitive structures. The nxtgrd file contains capacitance, resistance, and layer information, along with the ITF statements.

Figure 2-2 Process Database File Generation

You can use ITF files in two ways:

- Run the grdgenxo tool to process the ITF file and create an nxtgrd file for use in the StarRC tool (specified with the `TCAD_GRD_FILE` command).
- Use the StarRC tool to read the ITF file directly by using the `ITF_FILE` command. The tool automatically uses the field solver to analyze the capacitance of the design. This operation is supported only for capacitance extraction in transistor-level flows.

For more information about the grdgenxo tool and process modeling, see [Chapter 12, “Process Modeling Methodology”](#) and [Chapter 13, “Process Characterization.”](#) For reference information about the ITF process description statements, see [Chapter 16, “ITF Statements.”](#)

The StarXtract Command

Invoke the StarRC tool by using the `StarXtract` command at the operating system prompt. [Table 2-1](#) provides brief descriptions of the command options and some links to find more information. The command syntax is as follows:

```
% StarXtract option1 option2 ...
star_cmd_file [cmd_file2] [cmd_file3] ...
```

Table 2-1 Command-Line Options for the StarXtract Command

Option	Description
<code>-cleanN</code>	Cleans netlisting process (for transistor-level extraction only)
<code>-Display</code>	Converts selected design data to Milkyway format for display; does not extract. For more information, see the DEBUG_MILKYWAY_DATABASE reference page.
<code>-Display_mf</code>	Converts selected metal fill design data to Milkyway format for display; does not extract. For more information, see the DEBUG_MILKYWAY_DATABASE reference page.

Table 2-1 Command-Line Options for the StarXtract Command (Continued)

Option	Description
<code>-convert_gpd_to_spef</code>	Creates a SPEF netlist from a GPD database
<code>-dump_gpd_config</code>	Generates an ASCII configuration file for a GPD database
<code>-set_gpd_config</code>	Applies a configuration file to a GPD database
<code>-reset_gpd</code>	Resets a GPD database to its original state
<code>-gui</code>	Invokes the StarRC graphical user interface (GUI). For more information, see Chapter 7, “The Graphical User Interface.”
<code>-ultra</code>	Uses the STAR-RC2_ULTRA_MANAGER license key only. For more information, see StarRC Licensing .
<code>-custom</code>	Uses the STAR-RC2_CUSTOM_MANAGER license key only. For more information, see StarRC Licensing .
<code>-cdnlicsvr</code>	Runs the Virtuoso® Integration License Server
<code>-tech_out</code>	Displays a list of StarRC command options and their defaults, if applicable
<code>-v</code>	Displays the StarRC version
<code>-h</code>	Displays the command usage report
<code>-iapinetmap</code>	Displays the net name or id mapping for IAPI
<code>-iapixindump</code>	Displays the ASCII xin output for IAPI
<code>-pio</code>	Dumps the Star-DC PIO file from Milkyway
<code>-skip</code>	Dumps the skip cells from Milkyway models
<code>-save_eco</code>	For ECO extraction, compresses the incremental extraction directory and saves it to a file. For more information, see Chapter 5, “ECO Extraction.”
<code>-restore_eco</code>	For ECO extraction, uncompresses a previously saved file and restores the incremental extraction directory. For more information, see Chapter 5, “ECO Extraction.”
<code>star_cmd_file</code>	A StarRC command file; at least one is required
<code>cmd_file2, ...</code>	Optional additional command files

The StarRC Command File

The StarRC command file is a list of commands that specify conditions for the extraction run. You can specify multiple command files with the `StarXtract` command. For example, you might want to include general setup commands in one command file and design-specific commands in a separate file.

Commands in separate command files are treated as if they were written in a single command file, in the order provided. The first command file is read first, followed by the second command file, and so on. If StarRC commands are duplicated, later instances of a command usually overwrite earlier instances. However, some commands that contain lists of objects are cumulative. For information about how the StarRC tool treats multiple instances of specific commands, see the reference pages in [Chapter 15, “StarRC Commands.”](#)

Distributed Processing

To invoke multiple jobs for distributed processing, you can

- Manually start multiple `StarXtract` jobs in the same directory by using the same StarRC command file
- Use the `NUM_CORES` command in your StarRC command file

In distributed processing, StarRC determines how to handle the design to achieve parasitic consistency among the allocated CPU cores. After complete extraction of the design, StarRC integrates the results of the netlists created in parallel on all the allocated cores onto a single core, and merges them into a final netlist.

The master CPU executes the `StarXtract` command, performs the initial translation, and partitions the run as uniformly as possible among all of the available CPUs. After extraction is finished on all CPUs, the master CPU compiles the results and generates the final netlist.

If one of the processing jobs terminates abnormally, the incomplete job is placed in the task queue for completion by an active CPU, even if the failed CPU was the master CPU. Subordinate CPUs check the task queue every 90 seconds, so there is a brief delay before an idle CPU picks up an incomplete job. The extraction results are not merged until all partitions have been processed.

To optimize performance, ensure that the directory specified by the `STAR_DIRECTORY` command is located on the machine that executes the first `StarXtract` command. Running the translation across a network greatly increases the runtime.

You must have a StarRC or StarRC Ultra license to run distributed multicore processing. The StarRC Custom package supports only single-core processing.

Manual Submission of Distributed Processing Jobs

Distributed processing for the `StarXtract` command requires that you begin a job on a single CPU and then use a remote login to execute jobs on other machines.

LSF System

A Load Sharing Facility (LSF) automatically distributes jobs to multiple CPUs. The environment setup might differ between LSF clusters, but the usage should be similar to that of the `bsub` command. The following script distributes a job across three CPUs:

```
bsub StarXtract star_cmd  
bsub StarXtract star_cmd  
bsub StarXtract star_cmd
```

Other Systems

If you are not using an LSF system, you must log into multiple remote machines to distribute the extraction. On each CPU, run StarRC with the following commands:

```
% xterm  
% cd /home/directory  
% StarXtract star_cmd  
% rlogin remote_machine  
% cd /home/directory  
% StarXtract star_cmd
```

You must execute the `StarXtract` command in the same directory for all machines.

Automatic Submission of Distributed Processing Jobs

With distributed processing, you can start a single run and let the StarRC tool automatically submit multiple jobs according to your specifications.

The job submission command can be specified either in an environment variable or as a command in the StarRC command file. If set in both places, the StarRC command file takes precedence. The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- [LSF System](#)
- [Oracle Grid Engine](#)
- [General Network With a List of Machines](#)
- [Runtime Design Automation System](#)

The license requirement for this feature is the same as that required by manual submission for the same number of jobs.

When you are using the Gridware system or the list of machines, a shell script named `_starrcdp.csh` is written to the current working directory, then invoked by a grid command (for a Gridware system) or the `rsh` command (for a list of machines).

LSF System

The submission command for LSF systems is `bsub`. You can specify it in either of the following forms:

- An environment variable before launching the tool. Enclose the entire submission command in single quotation marks because it might contain multiple arguments. For example,

```
% setenv STARRC_DP_STRING 'bsub -R "rusage[mem=5000]"'
```

- A statement in the StarRC command file. For example,

```
STARRC_DP_STRING: bsub -R "rusage[mem=5000]"
```

Oracle Grid Engine

The submission command for Oracle Grid Engine systems is `qsub`. You can specify it in either of the following forms:

- An environment variable before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example,

```
% setenv STARRC_DP_STRING 'qsub -P bnnormal -l "mem_free=1G" \
-v "STARRC_LICENSE_WAIT=YES"'
```

- A statement in the StarRC command file. For example,

```
STARRC_DP_STRING: qsub -P bnnormal -l "mem_free=1G" \
-v "STARRC_LICENSE_WAIT=YES"
```

To pass environment variables to the job, you must use the `-v` option. In this example, the `STARRC_LICENSE_WAIT` environment variable is set to allow license queuing.

General Network With a List of Machines

For a general network with a list of machines, the syntax is

```
STARRC_DP_STRING: list host1[:n1] [host2[:n2] ... hostm[:nm]]
```

where `host1:n1` means that you are submitting `n1` jobs (an integer number of jobs) to the machine with name `host1`. The default number of jobs per machine is 1. Do not use spaces inside the `host:n` syntax; use one or more spaces between host machines. Each machine must be available through a simple UNIX `rsh` command without a password. The keyword for the machine where the parent job is started is `localhost`. If you use `localhost`, system calls are used instead of `rsh` to submit the jobs.

Specify the `STARRC_DP_STRING` variable in either of the following forms:

- An environment variable before launching the tool. Enclose the list in single quotation marks because it might contain multiple arguments. For example,


```
% setenv STARRC_DP_STRING 'list localhost:1 alpha:2 beta:4 gamma'
```
- A statement in the StarRC command file. For example,


```
STARRC_DP_STRING: list localhost:1 alpha:2 beta:4 gamma
```

Runtime Design Automation System

The submission command for a Runtime Design Automation (RTDA) system is `nc run`. You can specify it in either of the following forms:

- An environment variable before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example,


```
% setenv STARRC_DP_STRING 'nc run -R "rusage[mem=5000]"'
```
- A statement in the StarRC command file. For example,


```
STARRC_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

Distributed Processing Reports

After the run is complete, the StarRC tool provides a distributed processing report in the summary file. The report includes the following information:

- Stage summary

Reports runtime, memory consumption, CPU or host on which the job was executed, and job completion timestamp.
- Distributed processing summary for distributed stages

Shows the maximum and average runtime for each CPU.

- Total runtime
 - Shows timestamps for the beginning and the end of the run, in addition to the total runtime.
- Pre-extraction and postextraction times
 - Reports the runtime information of pre-extraction and postextraction stages.

The following example shows the distributed process summary report based on cores:

CPU_03	*xTractDB	Elp=01:22:28	Cpu=01:22:25	Mem=2394.9	Feb 28 14:50:17	sys701
CPU_01	*xTractDB	Elp=01:23:13	Cpu=01:23:10	Mem=2332.2	Feb 28 14:51:00	sys701
CPU_02	*xTractDB	Elp=01:23:22	Cpu=01:23:19	Mem=2180.9	Feb 28 14:51:14	sys702
CPU_04	*xTractDB	Elp=01:23:42	Cpu=01:23:35	Mem=2431.6	Feb 28 14:51:23	sys702

3

Design Databases

This chapter describes how to manage the design and layout-versus-schematic databases that the StarRC tool uses as input.

This chapter contains the following sections:

- [Introduction to Physical Databases](#)
- [The IC Compiler II NDM Database Flow](#)
- [The IC Compiler Milkyway Database Flow](#)
- [The LEF/DEF Database Flow](#)
- [The Calibre Connectivity Interface Flow](#)
- [The Hercules LVS Tool Flow](#)
- [The IC Validator LVS Tool Flow](#)
- [Cross-Referencing in Transistor-Level Flows](#)
- [Parameterized Cells](#)
- [Metal Fill](#)

Introduction to Physical Databases

The StarRC tool calculates parasitic effects based on the physical layout of a design. The output of a layout-versus-schematic (LVS) tool is used to determine where the physical parasitic devices connect to the logical elements in the design. The supported databases are as follows:

- Design libraries in the New Data Model (NDM) format created by the IC Compiler II tool
- Design libraries in the Milkyway format created by the IC Compiler tool
- Design libraries in LEF/DEF format
- LVS output from the IC Validator tool
- LVS output from the Hercules tool
- LVS output from the Mentor Graphics® Calibre® tool

The StarRC tool can run on layouts containing opens or shorts. Special provisions are made to repair the netlist so that delay calculation can be performed even on the problem nets. Shorted nets are always netlisted independently and contain only parasitics for the polygons that really belong to them. Open nets can optionally be connected with the `NETLIST_CONNECT_OPENS` command in the technology file. Both opens and shorts are reported explicitly in detailed summary files. However, some StarRC results might be inaccurate in a design with shorts or opens.

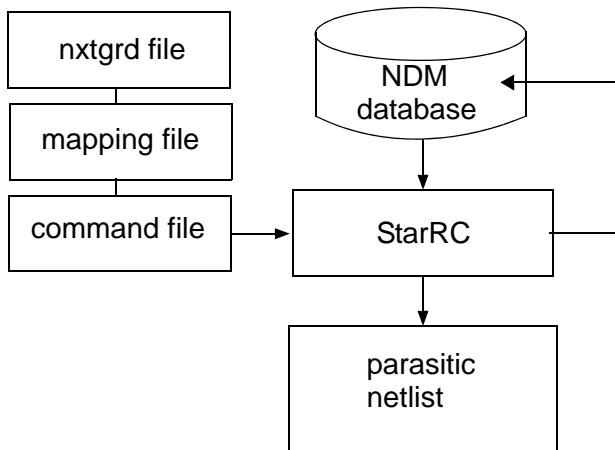
The StarRC tool integrates directly with the Synopsys IC Compiler tool for signoff-driven design closure. The tool also integrates smoothly with LEF/DEF-based place and route flows. You can read leaf cells in GDSII format directly to merge them during translation into the layout database.

The IC Compiler II NDM Database Flow

The IC Compiler II tool generates design libraries in the New Data Model (NDM) format. The StarRC tool can read these design libraries directly.

The extraction flow shown in [Figure 3-2](#) shows the IC Compiler II layout database containing all network annotation information. The layout does not need to be LVS-clean for completion of a successful extraction. The StarRC tool issues warnings when it finds open or shorted nets.

Figure 3-1 NDM Format Design Library Flow



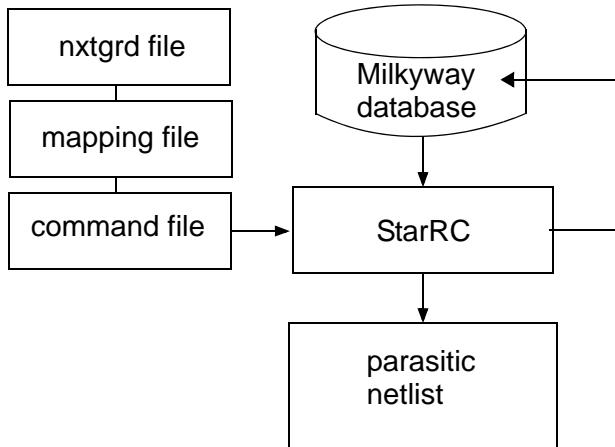
The following commands are used only for IC Compiler II design libraries. For more information, see the command reference pages.

- [NDM_DATABASE](#)
- [NDM DESIGN VIEW](#)
- [NDM_EXPAND_HIERARCHICAL_CELLS](#)
- [NDM_SEARCH_PATH](#)
- [TRANSLATE_NDM_BLOCKAGE](#)

The IC Compiler Milkyway Database Flow

The extraction flow shown in [Figure 3-2](#) shows the Milkyway layout database containing all network annotation information. Milkyway databases can be read directly by the StarRC tool. The layout does not need to be LVS-clean for completion of a successful extraction. The StarRC tool issues warnings when it finds open or shorted nets.

Figure 3-2 Milkyway StarRC Extraction Flow



The following commands are used only for Milkyway design libraries. For more information, see the command reference pages.

- [MILKYWAY_ADDITIONAL_VIEWS](#)
- [MILKYWAY_CELL_VIEW](#)
- [MILKYWAY_DATABASE](#)
- [MILKYWAY_EXPAND_HIERARCHICAL_CELLS](#)
- [MILKYWAY_EXTRACT_VIEW](#)
- [MILKYWAY_REF_LIB_MODE](#)

When determining the reference library status of a Milkyway database, StarRC uses the reference library mode stored within the main Milkyway database with the Milkyway dbSetLibRefCtrl FileMode function. This function specifies whether the reference library tree source is from the Internal Reference Mode or the Reference Control File Mode. See the Milkyway documentation for more information.

Antenna Diodes in Milkyway Designs

In the Milkyway database, when an antenna diode cell is inserted by place and route tools, the diode cell type is automatically set to a standard filler type. StarRC detects standard filler cell types automatically and ignores them during netlisting in the output parasitic file.

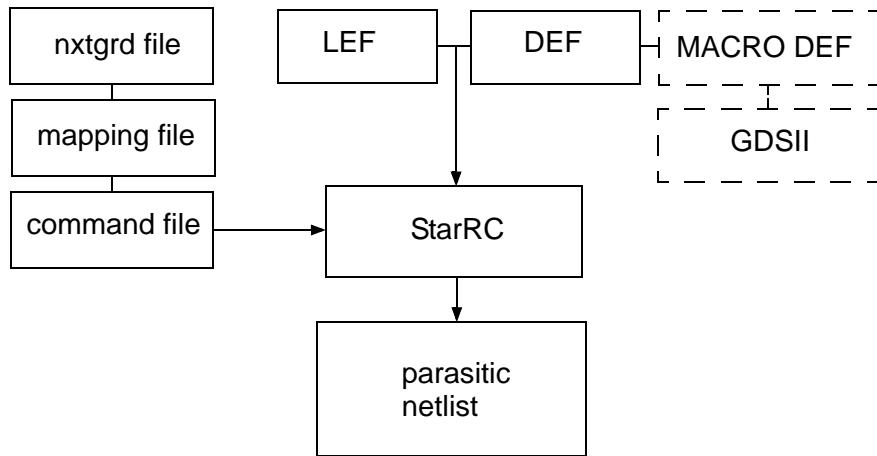
However, some antenna diode cells might be inserted manually with an incorrect cell type. This causes StarRC to extract and netlist the incorrectly defined diode cells. Diode cells should not be netlisted in the output parasitic file as they are not part of the original Verilog or SPICE netlist. They create parasitic back-annotation errors and warnings in the PrimeTime tool.

To correct a cell definition, query the diode cell instances to confirm the CELL FLAG property, set the antenna cell type to *standard filler*, and convert the pin type to DIODE-PIN type.

The LEF/DEF Database Flow

The Library Exchange Format/Design Exchange Format (LEF/DEF) layout description is read directly by the StarRC tool. The hierarchy is shown in [Figure 3-3](#). LEF/DEF file format versions 5.2 through 5.8 are supported.

Figure 3-3 LEF/DEF Extraction Flow



Most information about the design is read directly from the LEF/DEF database. The StarRC tool automatically identifies the following:

- Power nets
- Primary input and output ports
- Top-level block
- Skip cells

The following commands are used only for LEF/DEF design libraries. For more information, see the command reference pages.

- [DEF_ATTRIBUTE_FROM_LEF](#)
- [DEF_USE_PINS](#)
- [LEF_FILE](#)
- [LEF_USE_OBS](#)
- [MACRO_DEF_FILE](#)
- [TOP_DEF_FILE](#)

Hierarchical LEF/DEF designs are fully supported. You can specify multiple `MACRO_DEF_FILE` commands along with a single `TOP_DEF_FILE` command to extract a hierarchically routed design. If a macro DEF file is specified, all subcells referenced in the macro DEF must have corresponding MACRO definitions within the library LEF files.

You do not need to provide LEF descriptions for DEF macros. For DEF macros, the macro details are taken from the DEF file, not the LEF file. The DEF file is assumed to be more accurate than the LEF file.

The `DEF_USE_PINS` command controls the selection of pin information. If the command is set to `YES` (the default), pin information is taken preferentially from the DEF file. Pins not found in the DEF file are taken from the LEF file. Conversely, if the `DEF_USE_PINS` command is set to `NO`, the pin information is taken only from the LEF file.

In accordance with the LEF specification, the order in which the LEF files are specified is important. The technology LEF that contains layer definitions is required before any of the library LEF files are read. You can choose to obtain the routing width information for a specific DEF macro from any LEF file by using the `DEF_ATTRIBUTE_FROM_LEF` command.

Any parasitic capacitance information contained in the LEF files is ignored by StarRC. The layer resistance specifications are used only if resistance information is missing from both the `nxtgrd` file and the mapping file.

The LEF file must always contain via definitions, even if the vias are redefined in the DEF file. The DEF description takes precedence in the extraction whereas the LEF description is used for layer mapping and initial connectivity.

Merging Library GDSII Files in LEF/DEF Designs

GDSII information can be directly merged into the LEF description for library cells or macro blocks. When you use this feature, the StarRC tool uses only the pin shapes from the LEF MACRO cell definitions and discards the obstructions and other objects. The actual physical layout from the GDSII library is translated and used to represent the contents of skip cells during extraction.

GDSII layers for inclusion must be equated to a LEF database layer with the `GDS_LAYER_MAP_FILE` command. If any GDSII layer is not specified in the map file, it is not translated for extraction and does not contribute to the parasitics.

Indexes Warning in the Netlist Stage

If there is no port geometry for the pins of the cells in a LEF file, a warning is issued. For example:

```
MACRO inv
  PIN a
    DIRECTION INPUT ;
  END a
  PIN y
    DIRECTION OUTPUT ;
  END y
END inv
```

You must make the necessary correction in the LEF file.

Translation of Routing DEF Blockages

In a design flow, you can divide the chip into blocks and perform the routing of the blocks separately. These blocks are then integrated at the top level. While carrying out the routing at the block level, you can define routing blockages that represent the tracks in which the top-level routing resides.

While performing worst-corner (most pessimistic) extraction at the block or macro level, you can also consider the effect of the top-level signal net on the parasitics of the block-level nets. The top-level routing information might not be available during the block-level extraction, therefore the blockages defined for the macro serve as an approximation for the top-level routing. The StarRC tool must consider the effect of these routing blockages during extraction. StarRC, by default, does not read or translate DEF blockages.

The `TRANSLATE_DEF_BLOCKAGE` command translates only the routing DEF blockages from the file specified by the `TOP_DEF_FILE` command. DEF blockages from files specified by the `MACRO_DEF_FILE` command are ignored. The routing information corresponding to these blockages is already present in the top DEF file. Placement blockages in the top DEF file are ignored.

The following is an example of the blockage section in a DEF file:

```
[BLOCKAGES numBlockages ;
  [- LAYER layerName
    [+ COMPONENT compName | + SLOTS | + FILLS | + PUSHDOWN]
    [+ SPACING minSpacing | + DESIGNRULEWIDTH effectiveWidth]
    {RECT pt pt | POLYGON pt pt pt ...} ...
  ;] ...
  [- PLACEMENT
    [+ COMPONENT compName | + PUSHDOWN]
    {RECT pt pt} ...
  ;] ...
```

The Calibre Connectivity Interface Flow

The StarRC tool can read output files from the Calibre layout versus schematic (LVS) tool. This is achieved by using the Calibre Connectivity Interface, which generates layout, connectivity, port, and cross-referencing information from the Calibre LVS run in a form that is usable by the StarRC tool.

Using the Calibre Connectivity Interface flow, you can generate Detailed Standard Parasitic Format (DSPF) and Standard Parasitic Exchange Format (SPEF) netlists.

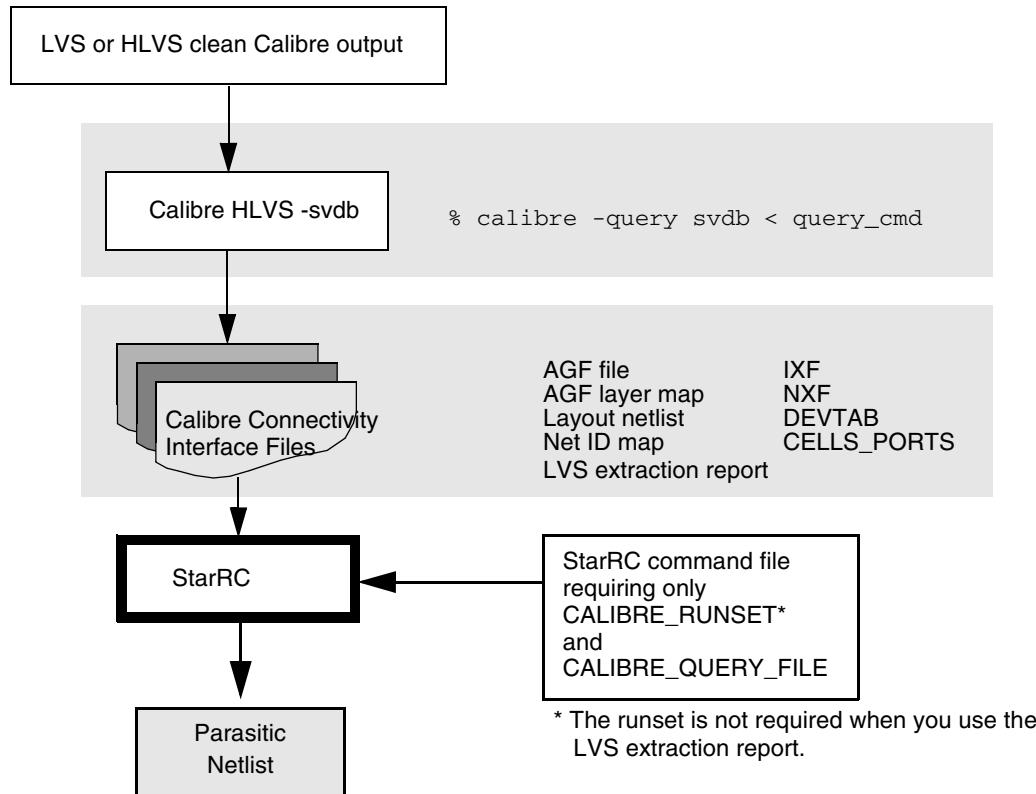
The Calibre tool generates multiple files with information about polygons, connectivity, text net names, device tables, and LVS cross-referencing tables. The StarRC tool reads the following Calibre Connectivity Interface information to construct the parasitic netlist:

- LVS extraction report file
 - When you include the LVS extraction report in the Calibre Connectivity Interface, the StarRC tool does not need to parse the Calibre rulefile.
- Annotated GDSII (AGF) file containing polygon and connectivity information
- Mapping file describing layer mapping between the AGF file and the Calibre runset
- Ideal layout netlist
- Layout net names file
- Calibre device table describing terminal layers for all possible devices in the runset
- Top-level ports file
- LVS net cross-referencing table
- LVS instance cross-referencing table
- Calibre runset to interpret layer connectivity

By reading the Calibre query command file directly, the StarRC tool can locate all Calibre Connectivity Interface generated subordinate files relating to the ideal layout netlist, the annotated GDSII file, and the cross-referencing information. Error checking on the Calibre query command file ensures that the Calibre Query Server was invoked with the required options for use with the StarRC tool. For more information, see [Running the Calibre Query Server for Output to StarRC](#).

[Figure 3-4](#) illustrates the basic Calibre Connectivity Interface flow. The procedure varies depending on whether or not you use an LVS extraction report.

Figure 3-4 StarRC Calibre Connectivity Interface Flow



Procedure Without an LVS Extraction Report

Follow these steps if your Calibre Connectivity Interface flow does not include an LVS extraction report file:

1. Generate an LVS or HLVS clean design.
2. Create a collection of Calibre Connectivity Interface input files with the query command file, named `query_cmd` in the following example:

```
% calibre -query svdb < query_cmd
```

The `svdb` file contains the design data from Calibre. The file named `query_cmd` contains the Calibre options that create the Calibre Connectivity Interface files.

3. Create a StarRC command file containing the CALIBRE_RUNSET and CALIBRE_QUERY_FILE commands. For example,

```
BLOCK: my_example
TCAD_GRD_FILE: my_example.nxtgrd
MAPPING_FILE: my_example.map
CALIBRE_RUNSET: calibre.runset
CALIBRE_QUERY_FILE: query_input
```

4. Run StarRC with the prepared command file.

The CALIBRE_RUNSET and CALIBRE_QUERY_FILE commands find the necessary Calibre file information and create the output file.

Procedure Using an LVS Extraction Report

If the Calibre query file uses the LVS SETTINGS REPORT WRITE command to write an extraction report, the StarRC CALIBRE_QUERY_FILE command automatically obtains the runset file and Push Down Back Annotation (PDBA) information from the query file. The CALIBRE_RUNSET and CALIBRE_PDBA_FILE commands are invalid in this case. This capability is valid only for StarRC version I-2013.12-1 (or later) and Calibre version 2013.4-15.12 (or later).

Note:

It is the responsibility of the Calibre tool to ensure that the information in the LVS extraction report is accurate and correctly handles directives and conditional statements.

Follow these steps if your Calibre Connectivity Interface flow includes an LVS extraction report file:

1. Generate an LVS or HLVS clean design.
 2. Create a collection of Calibre Connectivity Interface input files with the query command file, named query_cmd in the following example:
- ```
% calibre -query svdb < query_cmd
```
- The svdb file contains the design data from Calibre. The file named query\_cmd contains the Calibre options that create the Calibre Connectivity Interface files.
3. Write a StarRC command file containing the CALIBRE\_QUERY\_FILE command but no CALIBRE\_RUNSET or CALIBRE\_PDBA\_FILE commands.
  4. (Optional) If you want to use an LVS extraction report file that is different from the one specified in the Calibre LVS SETTINGS REPORT WRITE command, specify the with the LVS\_EXTRACTION\_REPORT\_FILE command in the StarRC command file.
  5. Run StarRC with the prepared command file.

---

## Error Conditions in StarRC-Calibre Flows

For information about the proper ordering and setup of Calibre query file commands for use with the StarRC tool, see [Running the Calibre Query Server for Output to StarRC](#).

Selected StarRC error conditions related to the Calibre Connectivity Interface are as follows:

- The GDS NETPROP NUMBER, GDS PLACEPROP NUMBER, or GDS DEVPROP NUMBER properties must be set in accordance with the Calibre Query File requirements.
- GDS SEED LAYER ORIGINAL, when specified, must appear before the GDS MAP command in the Calibre query command file.
- XREF:COMPLETE is not supported in the Calibre Connectivity Interface flow.
- Duplicate layer mappings in the Calibre AGF layer mapping file produce an error condition because data in the AGF might be corrupted as a result. For example, if two AGF layers are mapped to the same layer number, the following error message appears during the Translate DB stage:

```
ERROR:StarXtract
ERROR: different gds layers are mapped to the same gds
layer number:layer1, layer2, shared_layer_number
```

This condition can result if a partial layer mapping is provided in the Calibre query server command file. In general, you should not specify layer mappings (using GDS MAP commands) in the Calibre query server command file. For information about the query server command, see [Running the Calibre Query Server for Output to StarRC](#).

- Missing pin (x,y) information in the Calibre ideal netlist induces a warning message instructing you to include the relevant Calibre Connectivity Interface query commands in the command file.

For example, if the Calibre query server command LAYOUT NETLIST PIN LOCATIONS YES is not used during the query server run, StarRC issues the following warning:

```
WARNING: Unable to determine terminal locations for "0"
of device "n". This instance will not be netlisted.
```

- Composite STAMP commands in the rule file are not supported and result in an SX-1242 warning message. Simple STAMP commands are supported.

---

## Cross-Referenced Extraction in the Calibre Flow

In the Calibre Connectivity Interface flow, StarRC supports layout-based cross-referencing with the `XREF: YES` command. In this mode, all nets and devices that occur in the ideal layout netlist appear in the parasitic netlist, using schematic net and instance or device names wherever possible. Special naming techniques for handling various merging situations are described in the general discussion of the `XREF` command.

Schematic and layout cross-referencing in Calibre Connectivity Interface is purely layout based, meaning that the parasitic RC netlist mirrors the structure, connectivity, and quantities of nets, instances, and devices present in the actual layout. Calibre Connectivity Interface cross-referencing tables are used to annotate schematic names onto these nets, instances, and devices wherever matches are made during LVS.

For cross-referenced ideal devices, the model name specified with the Calibre NETLIST MODEL suboption is netlisted with `XREF: YES` whenever this suboption exists for a particular DEVICE command in the Calibre LVS rule file. Otherwise, the default Calibre model name is used. StarRC always uses the default model name with `XREF: NO`, because the layout netlist from Calibre uses that convention.

If a net does not exist in the Calibre NXF file, StarRC uses the layout net name with the prefix specified with the `XREF_LAYOUT_NET_PREFIX` command (default: `In_`). For example, if layout net A did not match in LVS and does not exist in the Calibre NXF file, StarRC assigns `In_A` in the parasitic netlist. Similarly, if an instance does not exist in the Calibre IXF file, StarRC uses the layout instance name with the prefix specified in the `XREF_LAYOUT_INST_PREFIX` command (default: `Id_`).

---

## Calibre Support of LVS Black Box Flows

To enable proper StarRC cross-referencing of Calibre LVS BOX cells, one additional command is required in the Calibre query server command file to output needed information in the Calibre Connectivity Interface NXF cross-referencing files. No additional StarRC commands are required.

Designers commonly perform LVS verification using incomplete subcell information by having the LVS tool not compare the functional contents of the subcell. In such cases the cell is called a black box, meaning that the LVS tool treats all instances of the cell as equivalent between schematic and layout without comparing the contents of the cell. The Calibre tool uses the LVS BOX syntax. The only constraint for a black-box cell is that cell ports must have correspondence between the schematic and layout.

To cross-reference LVS black-box cells, add the following Calibre query server command to the query command file:

```
NET XREF WRITE file_name LNXF [BOX]
```

The `NET XREF WRITE` command is required to enable cross-referencing using the `XREF` command in the StarRC tool. The `LNXF` keyword improves cross-referencing of nets that are not compared during LVS checking.

Note:

The LNXF construct of the `NET XREF WRITE` command requires Calibre version 2014.3 or later.

To enable Calibre LVS BOX capability, use the optional keyword `BOX`. Its effects on the NXF file are as follows:

- Within the NXF file, the Calibre tool lists hcells (or equivalence points) for all LVS BOX cells. Equivalence point lines begin with the percent character (%) in the IXF and NXF files. Note that equivalence points for LVS BOX cells are not listed in the NXF file if the keyword `BOX` is not used in the Calibre Query Server command file.
- Under the hcell (equivalence point) described in the NXF file, the Calibre tool lists all matched ports for the cell. This enables the StarRC tool to cross-reference all ports for LVS BOX cells.

An example of output from the Calibre query server `NET XREF WRITE LNXF BOX` command is as follows:

```
% invl 4 invl 6 BOX
0 vss 0 vss
0 vdd 0 vdd
0 b 0 b
0 a 0 a
```

Using this information, the StarRC tool

- Netlists cell instances for all LVS BOX cells using schematic instance names, whenever a match for the instance occurs in the Calibre IXF file.
- Netlists all port connections to LVS BOX instances using schematic port names, whenever a match for port names occurs in the Calibre NXF file.

Just as with non-LVS BOX cells, Calibre writes interconnect polygon information to the AGF file for LVS BOX cells. You should specify Calibre BOX cells as `SKIP_CELLS` in the StarRC command file. If you do not do this, the StarRC tool issues a warning and adds the BOX cell to the StarRC `SKIP_CELLS` list. Because these cells must be specified as StarRC skip cells, the StarRC tool treats the contents of LVS BOX cells in a gray-box manner by extracting capacitive interactions from extracted nets to polygons contained in such cells.

**Table 3-1** lists concepts, syntax, and files for the Calibre black box feature.

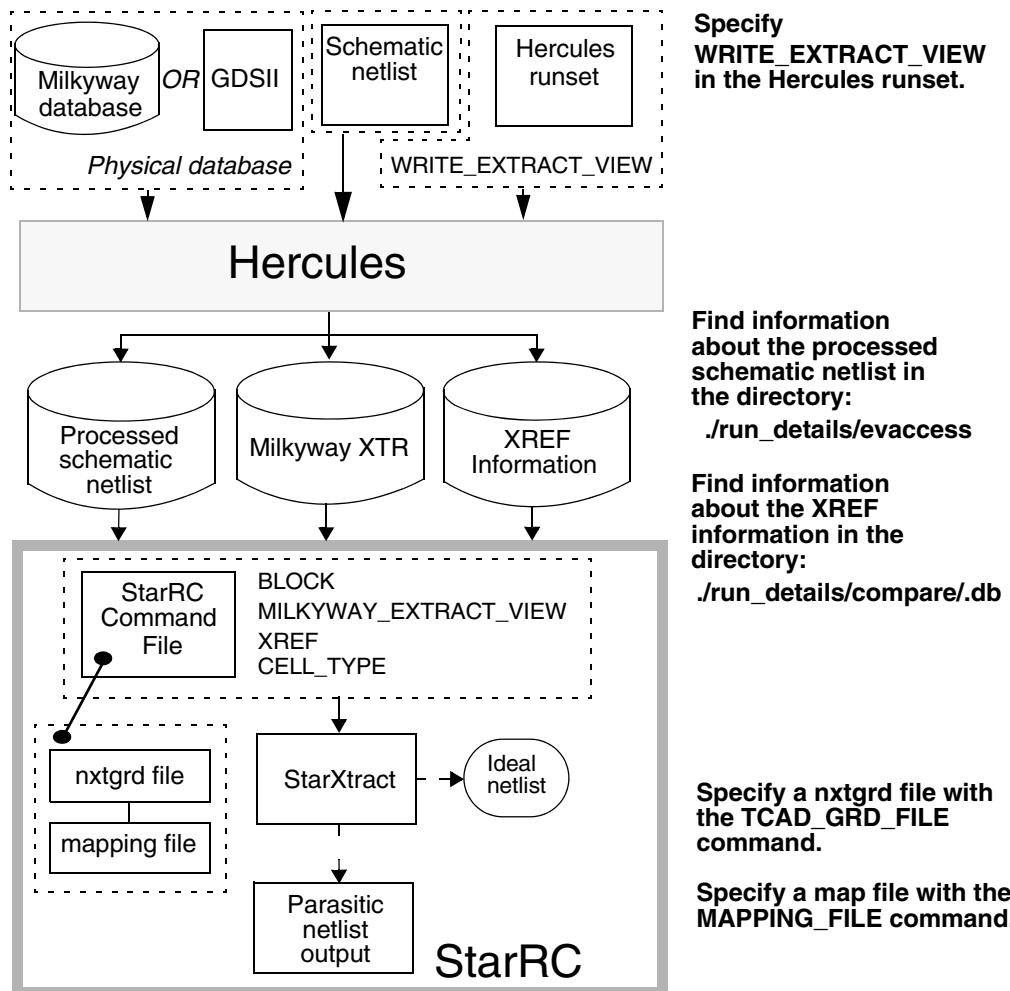
*Table 3-1 Calibre Black Box Feature*

| Term                            | Description                                                                                                                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LVS BOX                         | Calibre rule-file syntax for listing cells whose contents are not to be verified during LVS, but are LVS equivalent.                                                                          |
| hcell<br>(or equivalence point) | An expression of LVS equivalence between a schematic cell master and a layout cell master.                                                                                                    |
| IXF file                        | Calibre query server output file listing all instance (device and cell) matches between schematic and layout verified during LVS; this file is required by StarRC whenever XREF is activated. |
| NXF file                        | Calibre query server output file listing all net matches between schematic and layout verified during LVS; this file is required by StarRC whenever XREF is activated.                        |

## The Hercules LVS Tool Flow

To create a parasitic netlist output using the Hercules transistor-level extraction flow, you must provide a physical database, schematic netlist, and Hercules runset as shown in [Figure 3-5](#). To connect the Hercules runset to StarRC, include the `WRITE_EXTRACT_VIEW` Hercules command in the Hercules runset. In the StarRC command file, you must specify the `BLOCK`, `XREF`, `CELL_TYPE`, and `MILKYWAY_EXTRACT_VIEW` commands. Also, the path to the `nxtgrd` and mapping files must be specified in the StarRC command file. The connected Hercules database, or Milkyway XTR view, can be generated as a parasitic netlist or an ideal extraction of the layout from an original GDSII or Milkyway database. A net in the database that is not annotated with layout text is assigned a numerical net identifier by the Hercules tool.

*Figure 3-5 Hercules Extraction Flow*



---

## GDSII to XTR View Translation in Hercules Flows

A GDSII file contains no layer connectivity or ideal netlist information. The Hercules tool establishes layer connectivity and extracts an ideal layout netlist using rules defined in the Hercules runset. A connected version of the database is then stored in the form of a Milkyway XTR view database for input to the StarRC tool.

When generating the XTR view, follow these rules:

- The term `SUBSTRATE` is a reserved keyword in the Hercules runset and cannot be assigned as a `TEMP` or `PERM` layer for any Hercules command.
- In the case of a place-and-route Milkyway database, the Milkyway XTR view generated by Hercules should be written to a unique library because the technology information is different from that of the original library.
- Hercules runsets must be prepared in accordance with StarRC rules for device terminal and connectivity generation. For more information, see [Preparing Hercules Runsets](#).

The following Hercules runset example contains Hercules commands applicable to a cell-level GDS flow for use with StarRC:

```

header{
 layout_path = .
 inlib = library.gds
 format = stream
 block = top
}

assign_property {
 instance_name (4)
}

assign {
 poly (1) text(1;1)
 cont (2) text(1;1)
 metal1 (3) text(3;1)
 vial (4) text(1;1)
 metal2 (5) text(5;1)
 via2 (6) text(1;1)
 metal3 (7) text(7;1)
}

text_polygon metal3.text {
 cell_list = { top }
 size = 0.1
 text_list = { IN1 IN2 OUT }
} temp = io_pin

```

```

connect {
 poly metall1 by cont
 metall1 metall2 by vial
 metall2 metall3 by via2
 metall3 by io_pin
}

text {
 poly by poly
 metall1 by metall1
 metall2 by metall2
 metall3 by metall3
}

write_extract_view {
 library_name = TOP
 library_path = .
}

```

You must include a `WRITE_EXTRACT_VIEW` command as shown in the example to enable Hercules to write out a Milkyway XTR view. It is from this XTR view that StarRC derives the layout net annotation, layout device annotation, and layout connectivity.

The `TECHNOLOGY_OPTIONS` statement in the Hercules runset can have an impact on StarRC extraction runtime. A command-line option for Hercules, `-rcxt`, sets the `TECHNOLOGY_OPTIONS` statement for optimal StarRC performance; you should use the `-rcxt` option for transistor-level or hierarchical designs.

For more information about

- Marker generation, see [Marker Generation in Hercules](#).
- Runset creation, see [Preparing Hercules Runsets](#).

## Cross-Referenced Extraction in the Hercules Flow

Multi-equivalence points in the layout are supported for cross-referenced extraction. Multi-equivalence points are points in the layout where one or more layout cells equal to a single schematic cell. The `xy` coordinates of the instance ports and top-level ports are reported.

If the Hercules `IGNORE_CASE=TRUE` statement is specified in the runset, all schematic names are uppercase in the StarRC netlist. You must set `CASE_SENSITIVE=NO` in the command file if `IGNORE_CASE=TRUE` in the Hercules runset.

Note:

Check that the Hercules runset does not set `CREATE_VIEWS=FALSE` in the `EVACCESS_OPTIONS` section. Successful use of the StarRC `XREF` command requires that this option be either set to `TRUE` (the default) or left unspecified.

Certain memory designs might encounter errors in the XrefHN step of an XREF-enabled flow. For example,

```
ERROR: StarXtract
ERROR: Found layout instance SRAMblock258x532#448 of equiv
ERROR: SRAMblock258x532#448 which is not a valid equiv point
```

The SRAMblock instances are generated by the Hercules LVS engine when MEMORY\_ARRAY\_COMPARISON is set to TRUE (this is the default). The SRAMblock instances do not exist in the original schematic or layout netlists.

In general, the MEMORY\_ARRAY\_COMPARISON variable does not affect the LVS results in most memory designs. For nonmemory designs, you do not need to change this option. The StarRC XREF command options do not support the MEMORY\_ARRAY\_COMPARISON variable. You should set it to FALSE for memory designs that encounter this error when running LVS.

If StarRC finds an instance that is connected to the net "0" in the schematic netlist, the following error message is issued:

```
Build XrefHN
ERROR: StarXtract
ERROR: Schematic instance "MM15/SRC" is connected to ground "0".
ERROR: To prevent incorrect result, rename net "0" to another name.
Warnings: 0 Errors: 1 (See file xrefhn.sum)
```

To resolve this error, change the net name "0" to a different name in the schematic netlist and rerun Hercules LVS before starting a new StarRC run.

All StarRC XREF command modes support port ordering with the SPICE\_SUBCKT\_FILE command. The port list should match between the schematic cell definitions and the SPICE\_SUBCKT\_FILE definitions. StarRC generates net names in the *instance\_port* format for ports that are present in the SPICE\_SUBCKT\_FILE definitions but missing from the schematic or layout to ensure that the port count and order always match the SPICE\_SUBCKT\_FILE definitions.

When you specify a series merging in Hercules Compare, StarRC reports `1n_` for noncross-referenced internal nets.

## Placement Information for the HSIM Reliability Flow

StarRC interfaces to HSIM through a Detailed Standard Parasitic Format (DSPF) file for both hierarchical and flat post-layout simulation flows. The Synopsys HSIM tool can read hierarchical or flat DSPF files to perform hierarchical or flat timing and reliability analysis.

An important output of reliability analysis is a thermal map showing voltage (IR) drop and electromigration violations provided by HSIM. Because the HSIM product is netlist based, the reliability analysis thermal map is generated using node information (\*|S, \*|I, \*|P) provided by StarRC in the DSPF netlist. In a flat extraction, all nodes are shown with respect

to the origin of the top cell and a thermal map can be drawn without ambiguity. However, in a hierarchical flow, each node in a hierarchical cell's DSPF is shown with respect to its origin. To map these nodes to the next level of hierarchy, you must know the placement of the cell in the next level of hierarchy with rotation and flip attributes.

StarRC generates placement information when you specify the following options:

```
SKIP_CELL_PLACEMENT_INFO_FILE: YES | NO
SKIP_CELL_PLACEMENT_INFO_FILE_NAME: file_name
```

When the `PLACEMENT_INFO_FILE` command is set to `YES`, StarRC generates the `blockname.placement_info` file with the following information:

- Angle
- Reflection
- Cell location
- Cell name
- Cross-referenced instance name

The following example shows the output:

```
* PLACEMENT FILE
* VENDOR "Synopsys, Inc."
* PROGRAM "StarRC X-2005.06"
* DATE "Mon Oct 24 17:48:56 2005"
* UNIT " MICRONS"

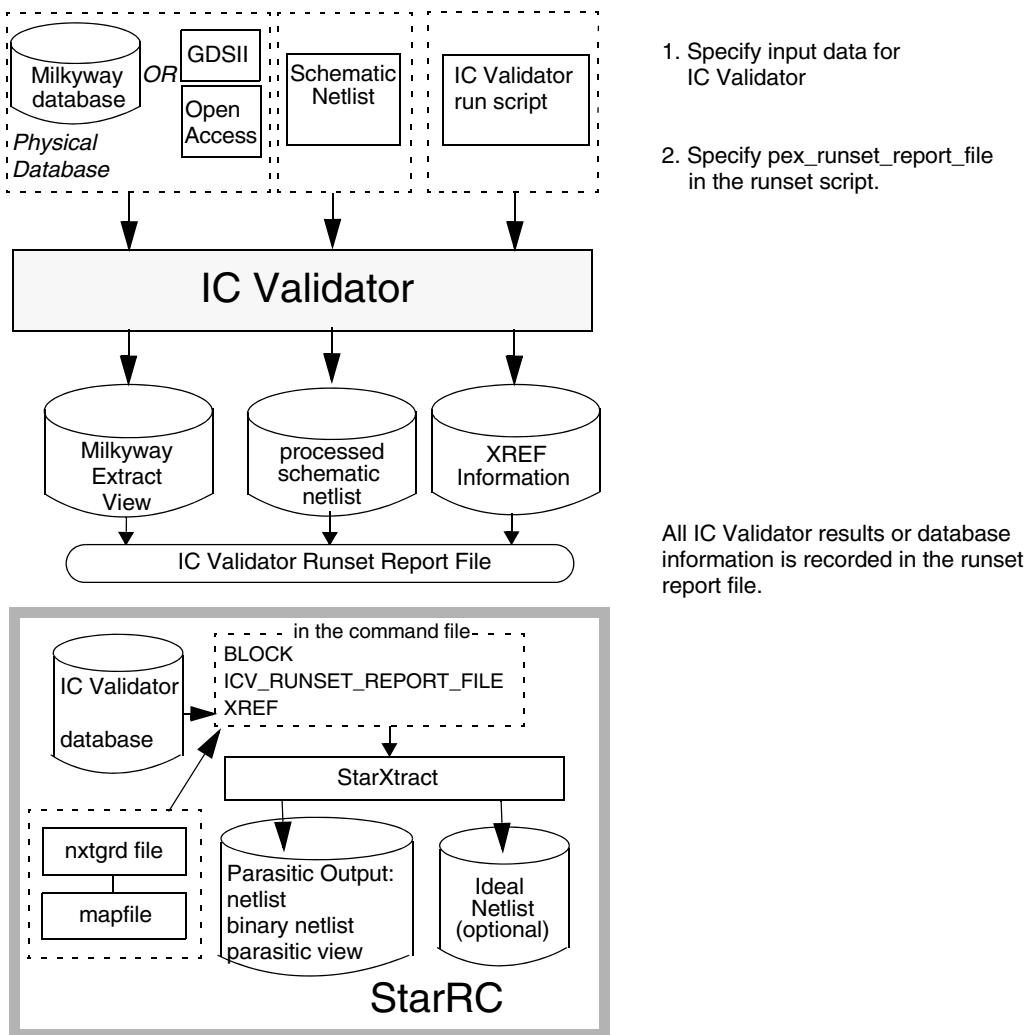
TOP_CELL = cell_name
instance_name cell_name x_coord y_coord angle reflection

XSI_0 INV1 49 132 0 NO
XSI_50 XOR2 484 132 180 NO
XSI_61 XOR2 124 312 180 YES
```

## The IC Validator LVS Tool Flow

To create a parasitic netlist with the IC Validator transistor-level extraction flow, you must provide a physical database, schematic netlist, and an IC Validator runset script to generate an IC Validator database, as shown in [Figure 3-6](#). The IC Validator database or the IC Validator runset report file can be used as input for the StarRC tool.

*Figure 3-6 IC Validator Flow*



---

## Steps in the IC Validator Extraction Flow

Follow these steps to use an IC Validator database with StarRC.

- In the IC Validator run script, specify the following command:

```
pex_runset_report_file
```

- In the StarRC command file, include these commands:

```
BLOCK, XREF, CELL_TYPE, and ICV_RUNSET_REPORT_FILE
```

- (Optional) In the IC Validator run script, specify the runset report file name in the `pex_report_handle` command. By default, the file name is specified as `pex_runset_report` in the `$ICV_HOME_DIR/includes/rcxt_public.rh` file.

The first two modifications are the required changes to run the IC Validator transistor extraction flow. The resulting IC Validator database or IC Validator runset report file can then be used to generate a parasitic netlist or an ideal extraction of the layout from an original GDSII database or Milkyway place and route database.

## Examples of Script Files

### IC Validator Runset Script

The following example shows the required commands in an IC Validator runset script with the default runset report file name.

```
pex_report_handle = pex_runset_report_file();
pex_generate_results(
 pex_matrix = pex_matrix,
 device_extraction_matrix = my_devices,
 device_db = device_db,
 layout_database = mw_handle,
 pex_process_map_file = pex_process_handle,
 pex_runset_report_file = pex_report_handle
);
```

To change the runset report file name, modify the `pex_report_handle` command specification, which is shown as `my_report` in the following example. All paths listed in the `pex_runset_report_file` command are absolute paths.

```
pex_report_handle = pex_runset_report_file("my_report");
```

### StarRC Command File

In the StarRC command file, add the following command:

```
ICV_RUNSET_REPORT_FILE : pex_runset_report
```

---

## Cross-Referenced Extraction in the IC Validator Tool

Multi-equivalence points in the layout are supported for cross-referenced extraction. Multi-equivalence points are points in the layout where one or more layout cells equal to a single schematic cell. The xy coordinates of instance ports and top-level ports are reported.

IC Validator can support a selective case-sensitive operation. The StarRC CASE\_SENSITIVE command might not cover all IC Validator case sensitivity combinations.

Specify the `pex_generate_results` function in the IC Validator runset script to run the IC Validator and StarRC flow, as shown in the following example.

```
pex_generate_results(
 pex_matrix = pex_matrix,
 device_extraction_matrix = my_devices,
 device_db = device_db,
 layout_database = mw_handle,
 pex_process_map_file = pex_process_handle,
 pex_runset_report_file = pex_report_handle
) ;
```

All StarRC cross-referencing modes support port ordering with the `SPICE_SUBCKT_FILE` command. The port count and port order in the schematic cell definitions and the `SPICE_SUBCKT_FILE` subcircuit definitions should always match. The StarRC tool generates net names in the *instance\_port* format for ports that are present in the `SPICE_SUBCKT_FILE` section but missing in the schematic or layout, depending on the options set in the `XREF` command.

## Error Conditions

### Memory Circuits

Certain memory designs might encounter errors in a cross-referenced flow. For example,

```
ERROR: StarXtract
ERROR: Found layout instance SRAMblock258x532#448 of equiv
ERROR: SRAMblock258x532#448 which is not a valid equiv point
```

The blocks named SRAMblock are generated by IC Validator when the `memory_array_compare` variable is set to TRUE (the default). Those blocks do not exist in the original schematic or layout netlists.

In general, the `memory_array_compare` variable does not affect the LVS results in most memory designs. For nonmemory designs, you do not need to change this option. The StarRC `XREF` options do not support the `memory_array_compare` variable. You should set it to FALSE for memory designs that encounter this error when running LVS.

## Ground Nets

If StarRC finds an instance that is connected to the net “0” in the schematic netlist, the following error message is issued:

```
Build XrefHN
ERROR: StarXtract
ERROR: Schematic instance "MM15/SRC" is connected to ground "0".
ERROR: To prevent incorrect result, rename net "0" to another name.
Warnings: 0 Errors: 1 (See file xrefhn.sum)
```

In this case, you must change the net name “0” to a different name in the schematic netlist and rerun LVS before starting a new StarRC run.

## Cross-Referencing in Transistor-Level Flows

The StarRC `XREF` command can be used with the Calibre, Hercules, and IC Validator flows.

[Table 3-2](#) lists definitions for terms related to cross-referencing.

*Table 3-2 Cross-Referencing Terms*

| Term                                                      | Definition                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XREF (also known as back-annotation or cross-referencing) | Generation of parasitic netlist containing layout parasitics with schematic-based net and instance names.                                                                                                                                                                                            |
| Schematic-based XREF                                      | Generation of parasitic netlist containing all devices and nets that occur in a schematic netlist used for LVS. Specified in the StarRC tool with the <code>XREF:COMPLETE</code> command.                                                                                                            |
| Layout-based XREF                                         | Generation of parasitic netlist containing all devices and nets that occur in the extracted layout netlist used for LVS. Specified in the StarRC tool with the <code>XREF:YES</code> command.                                                                                                        |
| Device merging                                            | Series or parallel combinations of multiple devices by the LVS tool for single electrically equivalent devices. Often necessary to successfully match schematic devices to corresponding layout devices that were implemented as electrically equivalent series or parallel combinations of devices. |
| Composite device                                          | Resulting device created by the LVS tool when individual layout or schematic devices are merged into series or parallel combinations. When devices are merged, only composite devices participate in the actual layout-to-schematic LVS matching.                                                    |

This section contains the following topics:

- [XREF: NO](#)
  - [XREF: YES](#)
  - [XREF: COMPLETE](#)
  - [Cross-Referencing By Device Property](#)
- 

## XREF: NO

When the `XREF` command is set to `NO`, the StarRC tool reports the layout net names generated during ideal layout extraction. These layout names are either generated or derived from the application of text. The layout cell instance names can either be the original GDSII instance name if the `ASSIGN_PROPERTY` statement in Hercules is used or they can be names generated by Hercules.

---

## XREF: YES

When the `XREF` command is set to `YES`, the StarRC tool does not use name prefixes for merged devices. The tool generates names that correspond to the premerged schematic names, making simulation easier and improving schematic-based simulation accuracy.

The `XREF: YES` command is layout-based; every layout device and net is reported. If the LVS tool successfully matches layout nets and devices to schematic nets and devices, the StarRC tool uses the schematic names in the parasitic netlist.

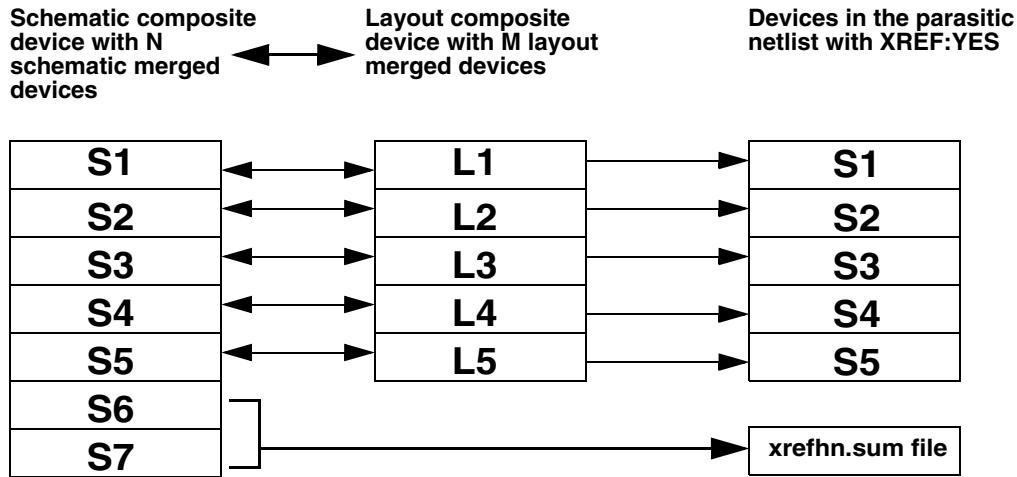
When the LVS tool establishes a one-to-one match between schematic net or device and layout net or device names, the StarRC tool netlists the layout nets and devices using their matching schematic names.

When the LVS tool creates a composite merged device on the schematic side consisting of  $N$  merged devices, and matches it to a composite merged device on the layout side consisting of  $M$  merged devices, the following rules govern the device names in the netlist generated by the `XREF: YES` command:

- $M$  individual devices are netlisted in the parasitic netlist, corresponding to the  $M$  layout devices.
- A one-to-one correspondence is established between the first  $X$  devices where  $X$  is the smaller of  $N$  or  $M$  within the schematic composite device and the layout composite device. The first  $X$  devices are netlisted in the parasitic netlist using their corresponding schematic device names.

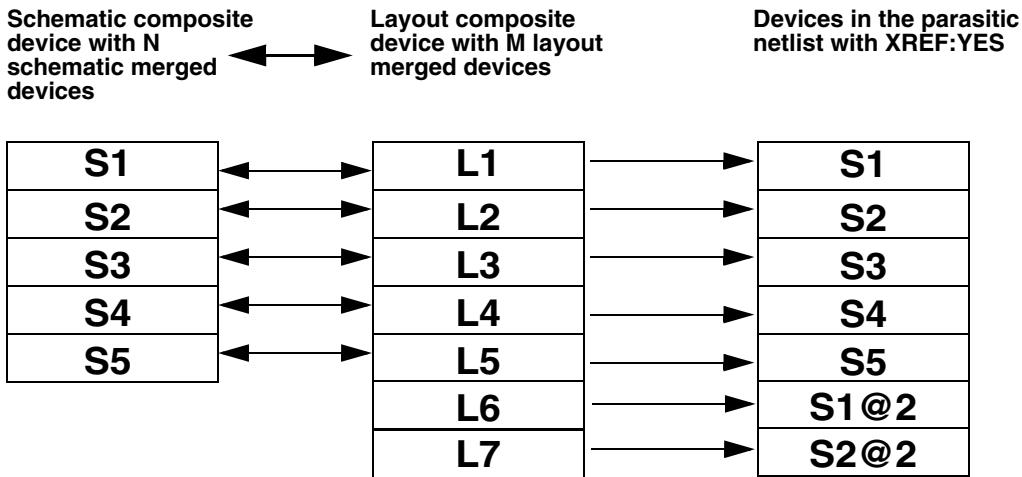
- If the number of schematic devices exceeds the number of layout devices ( $N > M$ ), the remaining ( $N - M$ ) schematic devices are not referenced in the schematic netlist, as shown in [Figure 3-7](#).

*Figure 3-7 Merged Device Handling: More Schematic Devices than Layout Devices*



- If the number of layout devices exceeds the number of schematic devices ( $N < M$ ), the remaining ( $M - N$ ) layout devices are mapped back to the top of the schematic device list, appended by @ [number] characters, as shown in [Figure 3-8](#).

*Figure 3-8 Merged Device Handling: Fewer Schematic Devices than Layout Devices*



**Table 3-3** shows the device naming conventions for devices participating in composite N:M merged devices with the XREF: YES setting. If a layout net is generated within a merged composite device, its name in the netlist contains the layout net name with a prefix specified by the XREF\_LAYOUT\_NET\_PREFIX command.

*Table 3-3 XREF: YES Device Naming Conventions*

| No. of devices<br>(schematic : layout) | Device instance names                                            | XREF-info<br>report file                         |
|----------------------------------------|------------------------------------------------------------------|--------------------------------------------------|
| 1:1                                    | S1                                                               |                                                  |
| 1:N                                    | S1<br>S1@2<br>S1@3 ...<br>S1@N                                   |                                                  |
| N:N                                    | S1<br>S2<br>S3 ...<br>SN                                         |                                                  |
| N:M<br>(N>M)                           | S1<br>S2<br>S3 ...<br>SM                                         | alias S(M+1) S1<br>alias S(M+2) S2<br>...<br>... |
| N:M<br>(N<M)                           | S1<br>S2<br>S3 ...<br>SN<br>S1@2<br>S2@2<br>S3@2 ...<br>SN@2 ... |                                                  |
| N:1                                    | S1                                                               | alias S2 S1<br>alias S3 S1 ...<br>alias SN S1    |
| 0:M (filtered schematic devices)       | <XREF_LAYOUT_INST_PREFIX>L1                                      |                                                  |

---

## XREF: COMPLETE

When the XREF command is set to COMPLETE, the StarRC tool reports every skip cell and device in the schematic. Parasitics appear in the netlist only for nets that have been successfully cross-referenced to schematic nets. Nets that do not cross-reference to a schematic net are treated as ideal connections. Schematic model names are reported for skip cells and primitive devices.

Internal nets of the SERIES or PATHS merged devices do not have \*|NET sections when the XREF: COMPLETE command is used; the netlist result is ideally included in the Instance Section.

**Table 3-4** describes how the StarRC tool computes properties for the Hercules and IC Validator flows.

*Table 3-4 Device Property Reporting*

| Schematic:<br>Layout  | Width                                                                                             | Length                                                       | AD, AS, PD, PS                                                                                                 |
|-----------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <b>MERGE_PARALLEL</b> |                                                                                                   |                                                              |                                                                                                                |
| 1:y                   | Summation of all the width values of the n MOS devices in the layout.                             | Smallest length values out of all of the layout MOS devices. | Summation of all the AD, AS, PD, and PS values of the NMOS devices in the layout.                              |
| x:1                   | Width of the layout MOS divided by x for each device.                                             | Same length of the layout MOS device for each device.        | AD, AS, PD, and PS values of the layout MOS device divided by x for each device                                |
| x:x                   | Corresponding width value from the layout. No calculation is performed.                           | Corresponding length value from the layout.                  | Corresponding AD, AS, PD, and PS values from the layout.                                                       |
| x:y                   | Summation of all the width values of the MOS devices in the layout, divided by x for each device. | Smallest length value of all of the layout MOS devices.      | Summation of all the AD, AS, PD, and PS values of the MOS devices in the layout, divided by x for each device. |

*Table 3-4 Device Property Reporting (Continued)*

| <b>Schematic:</b><br><b>Layout</b>                                                                                                                                                                                                                                                                                                                  | <b>Width</b>                                                        | <b>Length</b>                                                        | <b>AD, AS, PD, PS</b>                                                             |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|----------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| <b>MERGE_SERIES</b>                                                                                                                                                                                                                                                                                                                                 |                                                                     |                                                                      |                                                                                   |
| x:x                                                                                                                                                                                                                                                                                                                                                 | Corresponding width value from layout. No calculation is performed. | Corresponding length value from layout. No calculation is performed. | Corresponding AD, AS, PD, and PS values from layout. No calculation is performed. |
| <b>MERGE_SERIES_GATE</b>                                                                                                                                                                                                                                                                                                                            |                                                                     |                                                                      |                                                                                   |
| 1:y                                                                                                                                                                                                                                                                                                                                                 | Average width of the MOS devices in the layout.                     | Summation of all the length values of the MOS devices in the layout. | Summation of all the AD, AS, PD, and PS values of the MOS devices in the layout.  |
| x:1                                                                                                                                                                                                                                                                                                                                                 | Same width as the layout MOS device for each device.                | Length of the layout MOS divided by x for each device.               | AD, AS, PD, and PS values of the layout MOS device divided by x for each device.  |
| <b>MERGE_PATHS</b>                                                                                                                                                                                                                                                                                                                                  |                                                                     |                                                                      |                                                                                   |
| <p>MERGE_PATHS is a mixture of the MERGE_PARALLEL, MERGE_SERIES, and MERGE_SERIES_GATE cases.</p> <p>If there is a multiple-layout-cell-to-single-schematic-cell equivalency in the LVS, StarRC randomly chooses only one of the layout cells and uses the layout properties of that cell in when generating the netlist of all the XREF cells.</p> |                                                                     |                                                                      |                                                                                   |

---

## Cross-Referencing By Device Property

The StarRC and IC Validator tools provide a feature to improve the accuracy of cross-referencing parallel merged devices when the number of schematic devices is different from the number of layout devices. You must use the J-2014.06 or later versions of both tools to use this feature.

IC Validator can map devices into groups based on transistor width and provide this information in the cdb file. The feature is controlled by the following IC Validator environment variables:

```
XREF_PARALLEL_MAP
XREF_PARALLEL_MAP_MEMBER_LIMIT
XREF_PARALLEL_MAP_ITER_LIMIT
```

The StarRC tool uses the information in the cdb file automatically. The XREF command must be set to YES.

For example, assume the schematic parallel composite device contains these transistors:

```
M100 w=0.6u
M101 w=0.5u
```

The layout parallel composite device contains these transistors:

```
M1 w=0.3u
M2 w=0.3u
M3 w=0.5u
```

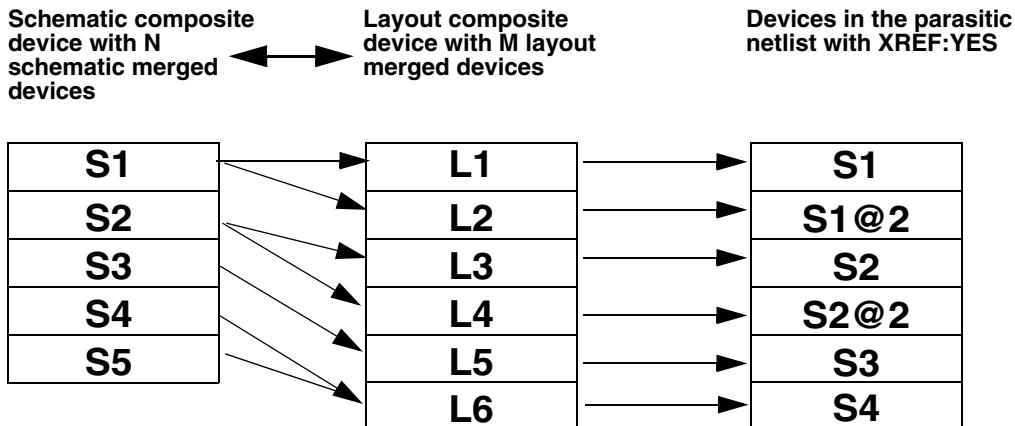
The final StarRC output (using the XREF: YES command) is as follows:

```
M1 -> M100
M2 -> M100@2
M3 -> M101
```

The combined width of the parallel-merged layout devices M1 and M2 is 0.6 um, equal to the width of schematic device M100. Layout device M3 is the same as schematic device M101.

[Figure 3-9](#) illustrates naming for scenarios where one schematic device maps to more than one layout device (S1 mapping to L1 and L2) and where more than one schematic device maps to one layout device (S4 and S5 mapping to L6).

*Figure 3-9 Merge By Device Property*



---

## How the XREF Command Affects SPF Netlists

The following examples show the effect of XREF command settings on an SPF netlist.

### XREF: NO

```
* | NET I_0/N_33 1.49e-02PF
* | I (I_0/I_39/M2:SRC I_0/I_39/M2 SRC B 0 -402.5 36.25)
* | I (I_0/I_39/M1:SRC I_0/I_39/M1 SRC B 0 -402.5 11)
* | I (I_0/I_41/M3:GATE I_0/I_41/M3 GATE I 1e-15 -419.5 36.25)
* | I (I_0/I_41/M1:GATE I_0/I_41/M1 GATE I 1e-15 -417 11)
Cg9 I_0/I_39/M2:SRC 0 4.82142e-15
Cg10 I_0/I_39/M1:SRC 0 6.20537e-15
Cg11 I_0/I_41/M3:GATE 0 1.62791e-15
Cg12 I_0/I_41/M1:GATE 0 2.25542e-15
R5 I_0/I_39/M2:SRC I_0/I_41/M3:GATE 141.086
R6 I_0/I_39/M2:SRC I_0/I_39/M1:SRC 1.41411
R7 I_0/I_39/M2:SRC I_0/I_41/M1:GATE 66.6508
R8 I_0/I_39/M1:SRC I_0/I_41/M3:GATE 95.2203
R9 I_0/I_39/M1:SRC I_0/I_41/M1:GATE 44.9831
R10 I_0/I_41/M3:GATE I_0/I_41/M1:GATE 625.714
```

### XREF: YES

```
* | NET B6 0.0223556PF
* | I (MM18@2:g MM18@2 g I 0.00425085 12.725 143.652)
* | I (MM18:g MM18 g I 0.00425085 11.875 143.652)
* | I (MM17@2:s MM17@2 s B 0 23.565 143.652)
* | I (MM17:s MM17@2 s B 0 23.565 143.652)
Cg30 MM18@2:g 0 2.0949e-15
Cg31 MM18:g 0 2.75462e-15
Cg32 MM17@2:s 0 2.03411e-15
Cg33 MM17:s 0 1.87562e-15
Cg34 B6:79 0 1.57134e-15
Cg35 B6:85 0 7.22334e-15
R7537 MM17:s B6:177 32.2773
R7538 MM17@2:s B6:173 48.3553
R7539 MM18:g B6:85 32.0359
R7540 MM18@2:g B6:79 32.2773
R7541 B6:85 B6:79 32.0359
```

## XREF: COMPLETE

```
* | NET x0/n33 1.49e-02PF
* | I (x0/x39/M2:SRC x0/x39/M2 SRC B 0 -402.5 36.25)
* | I (x0/x39/M1:SRC x0/x39/M1 SRC B 0 -402.5 11)
* | I (x0/x41/M3:GATE x0/x41/M3 GATE I 1e-15 -419.5 36.25)
* | I (x0/x41/M1:GATE x0/x41/M1 GATE I 1e-15 -417 11)
Cg1 x0/x39/M2:SRC 0 4.82142e-15
Cg2 x0/x39/M1:SRC 0 6.20537e-15
Cg3 x0/x41/M3:GATE 0 1.62791e-15
Cg4 x0/x41/M1:GATE 0 2.25542e-15
R1 x0/x39/M2:SRC x0/x41/M3:GATE 141.086
R2 x0/x39/M2:SRC x0/x39/M1:SRC 1.41411
R3 x0/x39/M2:SRC x0/x41/M1:GATE 66.6508
R4 x0/x39/M1:SRC x0/x41/M3:GATE 95.2203
R5 x0/x39/M1:SRC x0/x41/M1:GATE 44.9831
R6 x0/x41/M3:GATE x0/x41/M1:GATE 625.714
```

---

## Parameterized Cells

A parameterized cell (PCELL) is placed in the layout to represent a device. The cell contents are sized during placement to achieve certain geometries and functionality in the device. For simulation and analysis purposes, the entire contents of the cell are often characterized and modeled as a unit, including all conductor effects inside the cell boundary.

---

### How StarRC Layer-Based Rules Affect Parameterized Cells

By default, StarRC defines the boundary between interconnect parasitic effects and intradevice effects through layer-based rules for each of the following standard device types:

- Capacitance – These rules allow StarRC to discard all capacitance considered to be inside the device by ignoring capacitance between certain predetermined device terminal layer combinations.
- Resistance – StarRC permits the inclusion or exclusion of parasitic resistance on a layer-by-layer basis. However, because a parameterized cell is typically characterized as a complete unit, it becomes simpler and more accurate to use the cell boundary and not layer-based interactions to separate intradevice parasitic effects that are discarded from interconnect effects, but retained in the netlist.

StarRC uses gray box extraction techniques to handle parameterized cells, which means that the cell boundary is used to delineate the device boundary and knowledge of the layers within the cell is not required.

This extraction method allows you to extract PCELL devices without the need for device layer manipulation in the extraction rule file.

---

### How LVS Tools Handle Parameterized Cells

During ideal device extraction and layout versus schematic (LVS) checking, you can use standard device extraction commands to extract one or more logical devices from polygons inside a parameterized cell. The ideal layout netlist output then contains a hierarchy level representing the PCELL, referred to as the container cell. Inside this container cell is the extracted device, which represents the design down to the lowest level of hierarchy or transistor level.

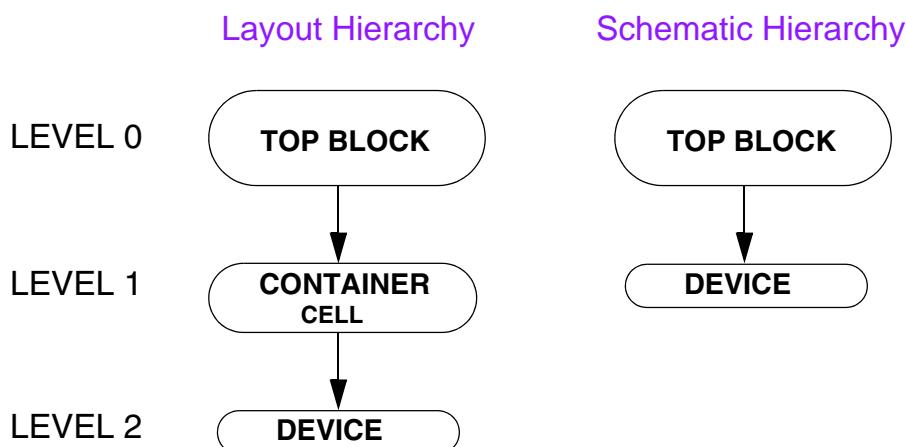
Conversely, the schematic netlist might or might not contain a level of cell hierarchy corresponding to the container cell. Possible scenarios are as follows:

- Single device in the layout with no container cell in the schematic
- Multiple devices in the layout with no container cell in the schematic
- Devices in the layout with container cell in the schematic

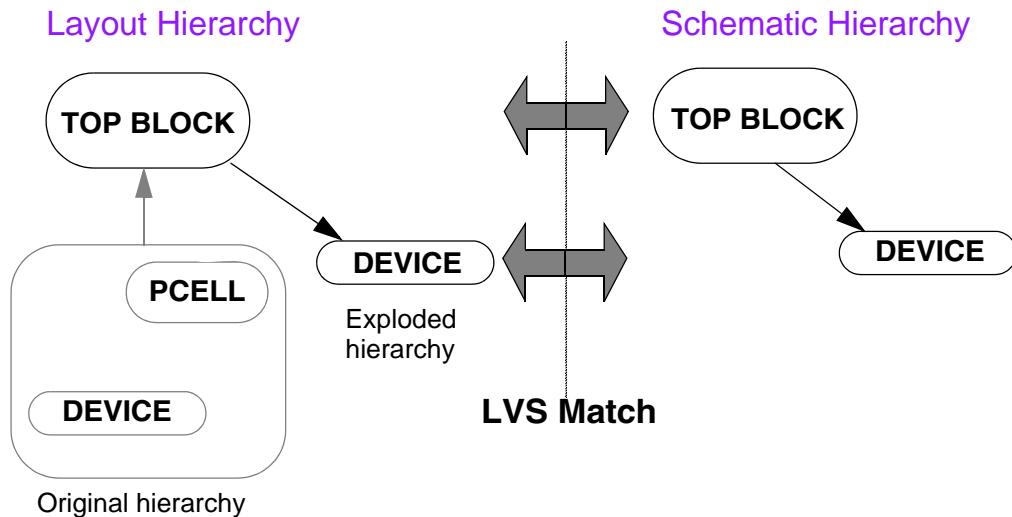
## Single Layout Device With No Container Cell

In one design scenario, the schematic netlist might contain only the device instance, not the container cell instance, because the parameterized cell is a physical but not logical object. This creates a nonuniform hierarchy between the layout and schematic, because the layout has an extra level of cell hierarchy not present in the schematic as shown in [Figure 3-10](#). To match the layout and schematic, LVS expands (explodes) the layout container cell in the layout netlist so that the extracted device appears in the layout netlist top block. This results in an equal hierarchy between the layout and schematic for complete LVS matching as shown in [Figure 3-11](#).

*Figure 3-10 PCELL Layout and Schematic Hierarchy; Single Device Inside Container*



*Figure 3-11 LVS Matching of PCELL Devices via Layout PCELL Explosion*

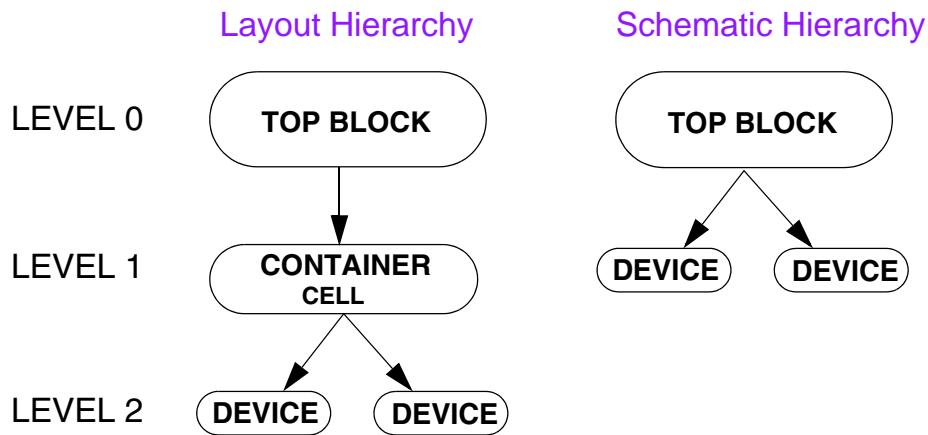


## Multiple Layout Devices With No Container Cell

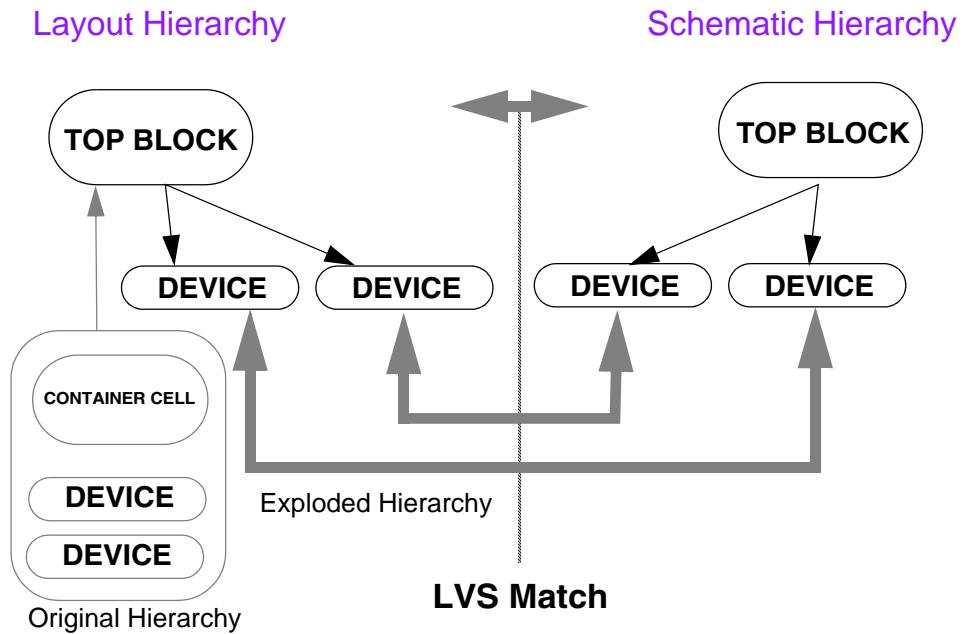
In another scenario, the schematic netlist might contain only the device instance, but not the schematic container cell instance. However, the layout container cell might contain multiple instances of device primitives, which might or might not be of the same device type. For example, a parameterized cell representing a MOSFET might have well diodes extracted inside the layout container cell as well as the MOS primitive itself as shown in [Figure 3-12](#).

In this case, LVS must explode the layout container cell again, as shown in [Figure 3-13](#), because no corresponding cell hierarchy exists in the schematic. Then, all primitives originally inside the layout container cell hierarchy are matched by LVS processing to primitives in the schematic.

*Figure 3-12 PCELL Layout and Schematic Hierarchy; Multiple Devices Inside Layout Container Cell*



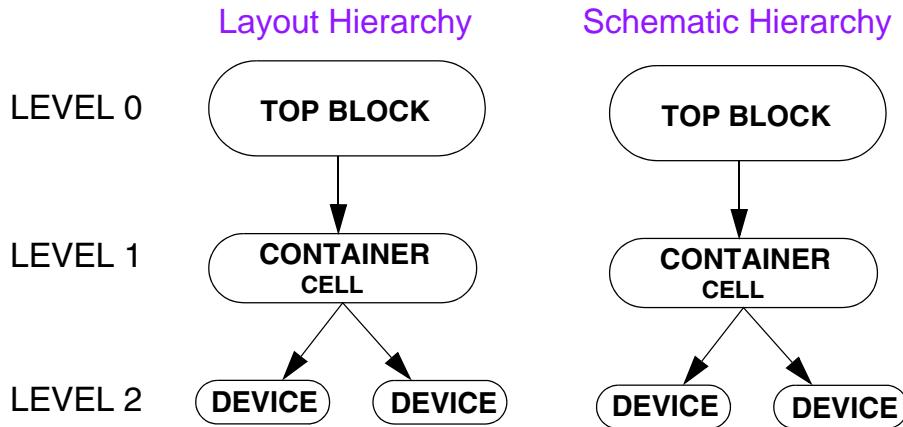
*Figure 3-13 LVS Matching of PCELL Devices via Layout PCELL Explosion: Multiple Devices Inside Layout Container Cell*



## Layout Devices in a Schematic Container Cell

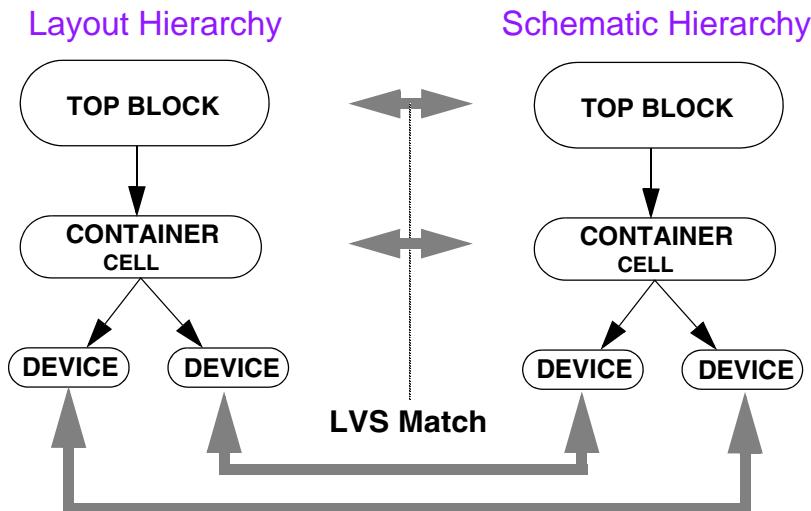
In this scenario, the schematic netlist might contain a device instance inside a schematic cell that has an EQUIV point (Hercules or IC Validator) / HCELL (Calibre) corresponding to the layout container cell. The layout container cell might contain one or more instances. In this case, LVS maintains the equivalent schematic and layout hierarchy to match the schematic device to the layout device as shown in [Figure 3-14](#).

*Figure 3-14 PCELL Layout and Schematic Hierarchy: Container Cell in Schematic*



In a design that has equivalence between the schematic and layout, no explosion is needed as shown in [Figure 3-15](#). In this case, LVS maintains the equivalent schematic and layout hierarchy to match the schematic device to the layout device.

*Figure 3-15 LVS Matching of PCELL Layout and Schematic Hierarchy*



---

## Extracting Parameterized Cells

To extract a parameterized cell (PCELL) as a fully characterized gray box cell unit during parasitic extraction, specify the `SKIP_CELLS` command in the StarRC command file. The following functions change for handling parameterized cells.

### Gray Box Handling

Parasitic resistance is extracted up to the instance port (x, y) location for each cell port. Capacitive interactions between top-level nets and the material inside the cell are extracted and compiled as ground capacitance in accordance with the StarRC standard gray box extraction method. Port capacitance is not included in the total capacitance for the net connecting to the port.

### The IGNORE\_CAPACITANCE Command

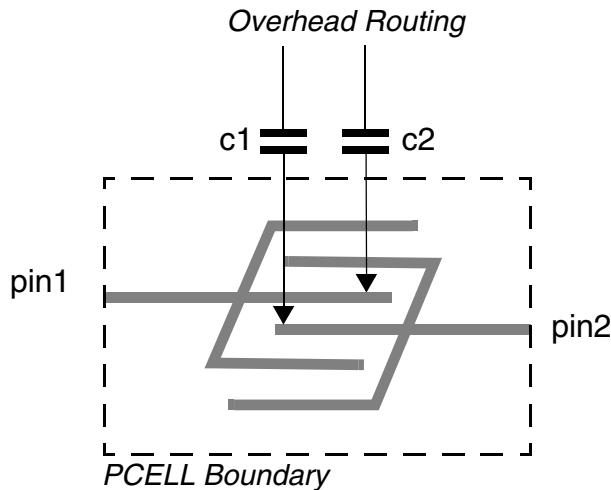
During extraction, the parameterized cell is treated as a gray-box skip cell. Functions related to the `IGNORE_CAPACITANCE` command are disabled for skip cells (but not for non-PCELL devices) because layer-based capacitance removal is not required.

### Extracting Coupling Capacitances

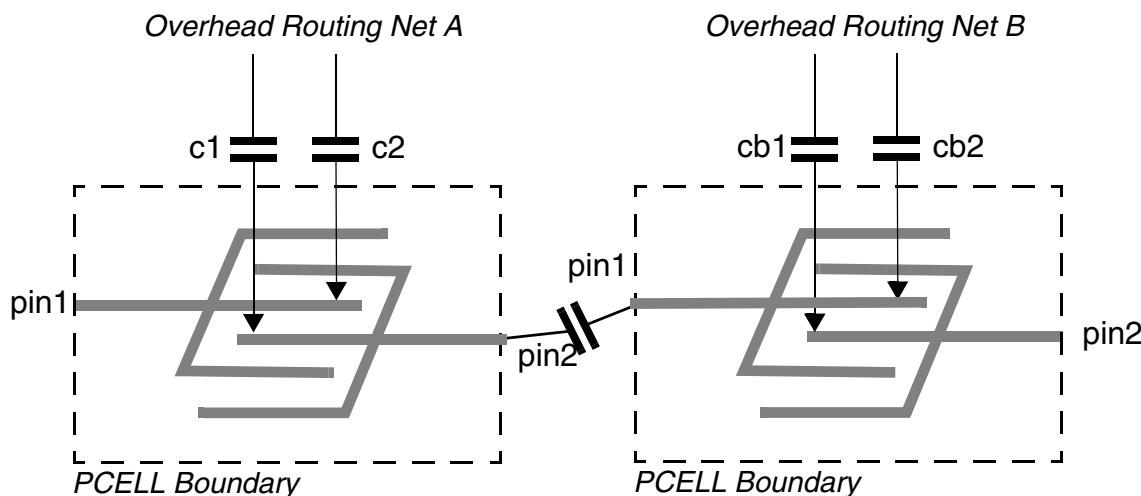
StarRC reports coupling capacitances for two additional conditions related to PCELL structures. You can report coupling capacitances between overhead nets and PCELL pins and coupling capacitances between different PCELL pins reported in the generated netlist. You can do this by specifying the `COUPLE_TO_PCELL_PINS` command.

With this command, the coupling capacitances between PCELL pins and overhead nets are reported or grounded, depending on the command. [Figure 3-16](#) shows the extraction of overhead nets to PCELL pins. [Figure 3-17](#) shows the extraction of overhead nets.

*Figure 3-16 Extraction of Overhead Nets*



*Figure 3-17 Extraction of Overhead Net*



## Retaining Coupling Capacitance Between Top and Skip Cell Levels

You can use the `COPUPLE_NONCRITICAL_NETS` command to retain coupling capacitances between top-level parent routing and `SKIP_CELLS` child net routing, where the fully routed child (DEF or CEL view) routing net names are used for coupling node names. This feature exists for the Milkyway flow using the SPEF netlist format.

To specify which noncritical nets are to be retained with an added prefix, use the `COUPLE_NONCRITICAL_NETS` and `COUPLE_NONCRITICAL_NETS_PREFIX` commands. Use the `COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX` command to add a subnode suffix to the noncritical nets. Use the `NONCRITICAL_COUPLING_REPORT_FILE` command to specify an output file containing all the capacitances coupled to the noncritical nets.

---

## SKIP\_PCELLS Netlist Behavior

For compilation purposes, the entire logical content of a PCELL generates a netlist. This means:

- All devices inside the PCELL container cell are generated in the DSPF instance section.
- All `* | I` lines in the DSPF file reflect connections to individual devices inside the container cell.

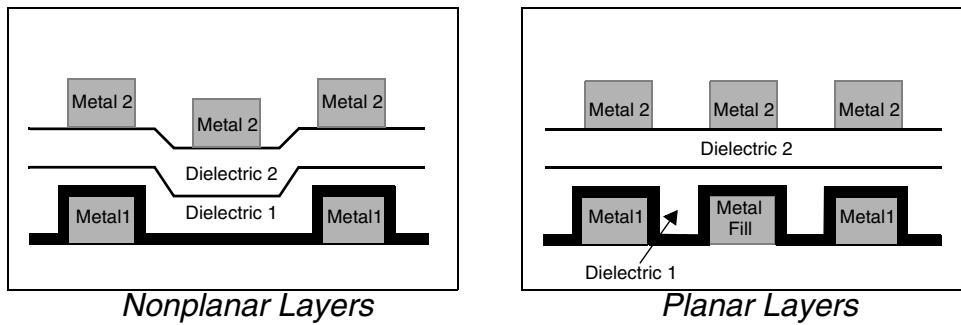
The logical content of the DSPF netlist (devices in the instance section, `* | I` lines) are identical to a generated netlist if no `SKIP_PCELLS` operation has been performed on the PCELL container. This allows StarRC to support different LVS configurations of PCELLs. The netlist result is as follows:

- The devices inside the PCELL container cell are generated with the same device names used when no `SKIP_PCELLS` operations are performed.
- All geometric device properties belonging to the devices inside the container cell are generated as normal in the DSPF instance section.
- If the PCELL container cell has a port that is not connected to a device inside the container cell, that port is ignored during netlist output.
- Any internal nodes inside the PCELL container cell (for example, nodes that are not instance ports of the container cell) are output as ideal nets in the DSPF.
- All specified `INSTANCE_PORT` commands for PCELLs are automatically set to `SUPERCONDUCTIVE`.

## Metal Fill

Metal fill is commonly included in designs that are manufactured with chemical-mechanical polishing (CMP) steps. CMP processes cause characteristic patterns of layer thickness or height nonuniformity. An example is shown in [Figure 3-18](#), in which the presence or absence of a metal line in an underlying layer affects the planarity of the top layer. To reduce these effects, dummy metal features are inserted into the design to improve process yield.

*Figure 3-18 Nonplanar Versus Planar Layers*



Metal fill features can be grounded or floating. Grounded metal fill is connected to power or ground by via connections. Floating metal fill has no connection to signal, power, or ground nets. Both types might exist in the same layout.

You can model metal fill in StarRC extraction in two ways:

- Emulated fill

A simple estimation of fill effects, used only in the early stages of the place-and-route flow

- Real (designed) fill

Accurate extraction of designs with metal fill polygons, read either directly from the design database (Milkyway or LEF/DEF) or from a separate GDSII file

To determine how many polygons were read from different layers, examine the metal fill statistics in the `readDB.sum` file located in the directory where StarRC deposits intermediate files. Metal fill reporting is off by default; to enable it, use the `REPORT_METAL_FILL_STATISTICS: YES` command.

## Emulated Metal Fill

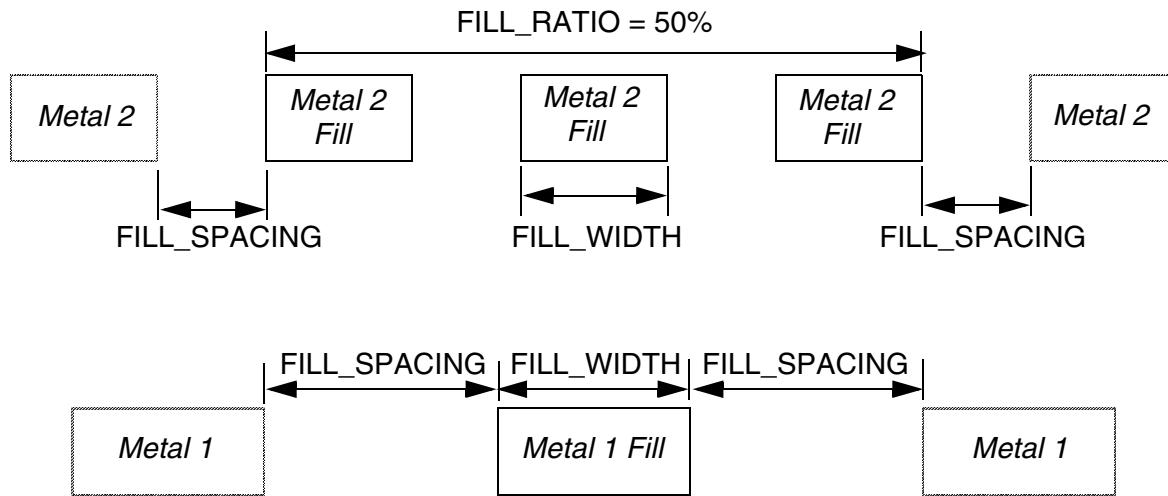
Emulated metal fill relies on ITF statements to model the fill parameters. No specific commands are required for the StarRC run, and no new data is required from the design database.

Four ITF statements describe metal fill: the `FILL_WIDTH` and `FILL_SPACING` statements for lateral capacitive effects, the `FILL_RATIO` statement for vertical capacitive effects, and the `FILL_TYPE` statement for specifying floating or grounded fill. [Figure 3-19](#) shows the parameter definitions. The parameters are valid only within a `CONDUCTOR` block in the ITF file.

When the `FILL_RATIO` statement is used without the `FILL_WIDTH` and `FILL_SPACING` statements, fill objects are placed only in areas where they do not generate any significant lateral capacitance with neighboring objects. Any empty space is modeled as though it were filled with floating metal of the same layer.

The `FILL_WIDTH` and `FILL_SPACING` statements must be specified together, along with the `FILL_RATIO` statement. Fill objects can be placed as close to existing features as the `FILL_SPACING` statement allows.

*Figure 3-19 Emulated Fill Example Spacing*



If an `nxtgrd` file was created with the emulated metal fill ITF statements, you cannot disable the emulated fill when using it for extraction. You must remove the ITF statements and generate a new `nxtgrd` file.

---

## Real Metal Fill

Real metal fill can exist in the design database as drawn shapes. The advantages and disadvantages of this approach are shown in [Table 3-5](#).

*Table 3-5 Advantages and Disadvantages of Real Metal Fill*

| Advantages      | Disadvantages                                                                                                   |
|-----------------|-----------------------------------------------------------------------------------------------------------------|
| Better accuracy | Creates more data<br>Requires longer runtimes<br>Requires you to regenerate the fill each time a change is made |

StarRC processes metal fills as floating or as grounded depending on their correct identification in the database. You identify fill in the database by using the `METAL_FILL_POLYGON_HANDLING` command options, as follows:

- `IGNORE`

This is the default. The command does not translate metal fill even if it is in the database.

- `FLOATING`

Processes all metal fill as floating even if the fill has been placed as grounded fill.

In this mode, capacitance is calculated between signal and fill polygons and between different fill polygons. After extraction, fill nodes are reduced on the fly and equivalent capacitance between signal nets and capacitance to ground for signal nets is calculated.

- `GROUNDED`

Processes all metal fill as grounded even if the fill has been placed as floating fill.

During signal net extraction, the fill polygons are treated just like a polygon belonging to a power and ground net. There is no special handling of these polygons during extraction.

- `AUTOMATIC`

Processes LEF/DEF and Milkyway fills based on the section in which they appear. These can be `LEF/DEF` or `ROUTE_TYPE` for Milkyway.

You can process a combination of floating and grounded metal fills. The grounded fills are created as a part of the power network by using text to identify them as part of the net. You can alternatively use the `METAL_FILL_GDS_FILE_NET_NAME` command to treat the fill as grounded in the design. The floating fills are created and used in the design either as `FILL` view in Milkyway format or `FILLS` in DEF format or GDS files.

StarRC reads design databases and translates metal fill polygons into the internal format before performing extraction. Translation depends on the setting of the `METAL_FILL_POLYGON_HANDLING` command or on the fill handling setting specified in the `GDS_LAYER_MAP_FILE` command.

The 3-D field solver can handle floating metal fill. The StarRC tool automatically prepares the appropriate 3-D field solver input deck based on the `METAL_FILL_POLYGON_HANDLING` command and the layer handling specified in the `GDS_LAYER_MAP_FILE` command.

## Handling Coupling Capacitance on Floating Metal Fills

The `METAL_FILL_POLYGON_HANDLING: FLOATING` and `REMOVE_FLOATING_NETS: YES` commands have similarities and differences.

The commands are similar in that floating polygons are not generated in the parasitic netlist. If either of these commands are used, any floating net or fill net or polygon does not have a `*D_NET` (SPEF format) or `*|NET` (standard parasitic format) section in the netlist, and no couplings are shown to these floating nets.

The different behavior of these commands becomes evident when you also use the `REMOVE_FLOATING_NETS: YES` command. StarRC treats floating nets as noncritical material. The tool finds the coupling capacitance that exists from the signal nets to this noncritical material and decouples these capacitors. This coupling capacitance is added to the total ground capacitance of the signal net.

If the `METAL_FILL_POLYGON_HANDLING` command is set to `FLOATING` and floating metal features are designated as fill (either with the `METAL_FILL_GDS_FILE`, or as fill material in the Milkyway or LEF/DEF database), the StarRC tool extracts coupling capacitance to floating fill polygons. However, the goal is to not generate these fill polygons for the netlist.

Instead of grounding the coupling capacitors to fill polygons, StarRC runs netlist reduction algorithms on the capacitors that connect to nodes on the fill. This allows StarRC to compute equivalent capacitances but eliminate the nodes on the floating fill polygons. Because the `METAL_FILL_POLYGON_HANDLING:FLOATING` command finds equivalent capacitance of capacitors coupling to fill, there are a few advantages:

- If nets couple to each other through fill polygons, then the netlist has a coupling capacitor between these two nets with `METAL_FILL_POLYGON_HANDLING:FLOATING`. When using `REMOVE_FLOATING_NETS: YES`, the coupling capacitance to the floating nets appears as additional ground capacitance.
- Nets that normally do not couple to each other can couple to each other after fill is added to the database. When `METAL_FILL_POLYGON_HANDLING:FLOATING` is specified, this effect is captured and a coupling capacitor between these nets shows up in the netlist. This increases the accuracy of signal integrity analysis because crosstalk effects induced by metal fill can be considered.

## Specifying Metal Fill in the Design Database

You can use Milkyway and LEF/DEF 5.4 (or later) databases containing real metal fill. No other flows are supported. Observe the following guidelines:

- Milkyway

The StarRC tool accepts metal fill polygons either in FILL view or CEL/FRAM view for a given block. This data can be also be created by IC Compiler.

- LEF/DEF

LEF/DEF versions 5.4 and later support the FILLS section for floating fill and fill wire for grounded fills only.

LEF/DEF has two different forms of syntax for specifying metal fill. Floating metal fill polygons are specified in the “FILLS” section of the DEF file. If the fill polygons are tied to power and ground nets, they are specified in the SPECIALNETS section (part of special Wiring with SHAPE defined as FILLWIRE) for the power and ground nets.

- GDS

The StarRC tool can read metal fill polygons from a separate GDSII file in addition to the design database. For this flow, the design database can be in Milkyway, LEF/DEF, or GDSII format (a Milkyway XTR view generated in Hercules or IC Validator or an AGF file generated in Calibre or IC Validator). The GDSII file must have metal fill polygons only, because all the polygons from the file are considered metal fill objects.

The handling mode for metal fill imported with the `METAL_FILL_GDS_FILE` command can be specified on either a global or layer-specific basis. For more information, see the command reference pages.

No floating fills should exist in the XTR view because StarRC cannot automatically identify these. You can attach (VSS) text to identify grounded fills in the physical layout and make them a part of the existing ground network.

The StarRC tool can accept a GDSII stream file containing only fill shapes and add them to the design. You can specify a flat GDSII file by using the `METAL_FILL_GDS_FILE` command. If you would like to attach all fills to a particular net, use the `METAL_FILL_GDS_FILE_NET_NAME` command.

To ensure that the fill structures are identified properly in the database, use the `GDS_LAYER_MAP_FILE` command.

## The Metal Fill Reuse Flow

The StarRC tool provides a method to improve runtime in an ECO loop by reusing metal fill. This feature is available for the LEF/DEF or Milkyway gate-level flows.

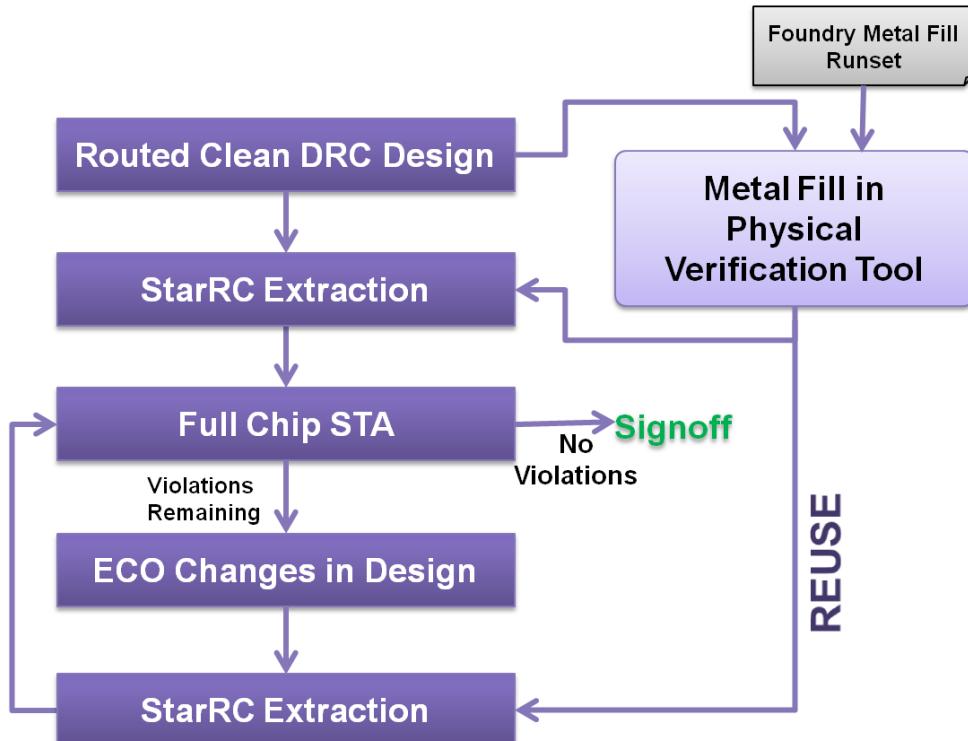
For timing closure, iterative design and analysis runs are frequently necessary to identify and fix problems. The design changes are known as engineering change orders (ECOs). ECO changes might include localized gate placement and sizing changes, net rerouting, and net additions or deletions. After the design is changed, parasitic extraction and timing analysis must be repeated to verify that the issues are resolved.

Metal fill insertion can be a significant part of the overall turnaround time of each ECO loop, even though only a small part of the design might have changed. The metal fill reuse flow provides a way to save time in the ECO loop by reusing metal fill features.

[Figure 3-20](#) shows the metal fill reuse flow. At the beginning of the design process, metal fill is generated for use in the first StarRC extraction run. After timing analysis, ECO changes might be necessary to fix timing violations. The modified design requires another extraction run, but now the original metal fill design is used, thereby saving runtime.

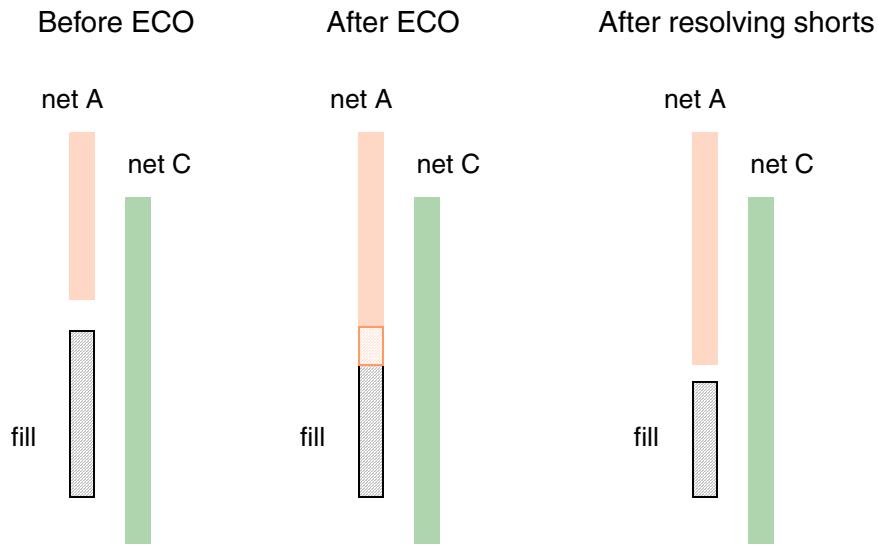
However, some of the old metal fill structures might short to the new design changes. The StarRC tool resolves these shorts by moving metal fill polygons away from signal nets.

*Figure 3-20 Metal Fill Reuse Flow*



The metal reuse flow has very little impact on accuracy because most of the original metal fill is the same throughout the signoff loop. Any metal fill structures that are moved to resolve shorts are moved in a manner consistent with the original fill polygon creation method.

*Figure 3-21 Resolving Shorts in the Metal Fill Reuse Flow*



The extraction runtime might increase by a small amount, but in most cases the decrease in turnaround time in the overall signoff loop is much larger.

To use the metal reuse flow, follow this procedure:

1. Perform a standard extraction on a MilkyWay or LEF/DEF design containing metal fill.
2. Perform timing analysis and fix timing violations.
3. Include the following command in the StarRC command file:  

```
REMOVE_METAL_FILL_OVERLAP: YES
```
4. (Optional) Specify the spacing to use when the StarRC tool resolves shorts between metal fill polygons and signal nets by inserting the value in the `conducting_layers` section of the mapping file as follows:  

```
conducting_layers
 overlap_fill_spacing=new_space
```

If you do not specify an explicit spacing value, the StarRC tool uses the `SMIN` value for the conductor layer in the ITF file.

5. Perform another extraction run.

6. Perform another timing analysis and fix timing violations.
7. Repeat the extraction, timing analysis, and design modification steps as needed.
8. If the number of ECO changes is large or the number of metal fill reuse iterations is large, generate new metal fill before proceeding.
9. For final signoff, perform full metal fill insertion followed by full-chip extraction and timing analysis to verify that there are no timing violations.

# 4

## StarRC Extraction and Output

---

This chapter describes fundamental aspects of the StarRC extraction process and the methods for reporting the extracted parasitics.

- [Simultaneous Multicorner Extraction](#)
- [The Galaxy Parasitic Database](#)
- [Output Netlists](#)

---

## Simultaneous Multicorner Extraction

Simultaneous multicorner (SMC) extraction optimizes the efficient extraction of multiple process and temperature corners for a design. The simultaneous multicorner flow runs a single extraction operation on a user-defined set of nxtgrd files and operating temperatures. This enables the generation of topologically equivalent individual corner netlists.

The simultaneous multicorner flow supports both gate-level and transistor-level extraction and can analyze up to 15 process corners (nxtgrd files) simultaneously. All extraction modes are supported, with the exception of inductance analysis.

For gate-level flows, the StarRC tool automatically detects process and design conditions for which SMC extraction would provide better runtime than single-corner extraction. If the StarRC commands are not set up to allow SMC extraction, the tool issues a warning message to recommend that you use SMC extraction.

Simultaneous multicorner extraction occurs whenever the `SIMULTANEOUS_MULTI_CORNER` command is either absent or set to `YES` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are both present. Other combinations are handled as follows:

- If the `SIMULTANEOUS_MULTI_CORNER` command is either absent or set to `NO` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are not present, StarRC runs single-corner extraction.  
For gate-level flows, the tool also issues a warning message to point out that simultaneous multicorner extraction would provide better runtime.
- If the `SIMULTANEOUS_MULTI_CORNER` command is set to `YES` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are not present, the tool issues an error message.
- If the `SIMULTANEOUS_MULTI_CORNER` command is set to `NO` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are present, the tool issues an error message.

To use the simultaneous multicorner flow, follow this general procedure. More information is provided in the subsequent sections.

1. Create a corners file that includes all of the defined corners (unique combinations of operating temperatures and nxtgrd files). The maximum number of corners is 15. The corners file must include a `TCAD_GRD_FILE` command for each corner.
2. Use the `SIMULTANEOUS_MULTI_CORNER: YES` command in the command file (or remove it from the file).
3. Add the `CORNERS_FILE: corner_file` command to the command file, where `corner_file` is the corners file name.
4. Add the `SELECTED_CORNERS: corner_list` command to the command file, where `corner_list` is a list of corners to be extracted. The selected corners must be a subset of the corners defined in the corners file.

5. (Optional) Use the `NETLIST_SMC_FORMULA` command to create a single netlist containing RC values written as formulas that use the corner names as variables. This option is valid only for transistor-level extraction.
6. (Optional) Use the `EXTRACTION: FSCOMPARE` command to use the FSCOMPARE flow during the simultaneous extraction.
7. (Optional) Use the `FS_EXTRACT_NETS` command to select specific nets for field solver extraction.

---

## The Corners File

A corner is a unique nxtgrd file and operating temperature pair. The `CORNERS_FILE` command specifies the file containing corner definitions. All corners that might be selected for extraction must appear in this file.

Use the following commands in the corners file to define each corner:

```
CORNER_NAME: name_of_corner
TCAD_GRD_FILE: path_to_nxtgrd_file
OPERATING_TEMPERATURE: temperature_in_Celsius
(optional) CORNER_TYPE: NOMINAL | OTHER
(optional) MAPPING_FILE: map_file
(optional) VIA_COVERAGE_OPTION_FILE: via_file
```

For simultaneous multicorner extraction, only the `TCAD_GRD_FILE` and `OPERATING_TEMPERATURE` commands within the corners file are used. Other instances of these commands in the StarRC command file are ignored.

You can optionally define some aspects of the extraction separately for each corner: the mapping file, the via coverage option file, and whether to output parasitic resistor temperature coefficients.

The `CORNERS_FILE` and `STAR_DIRECTORY` command arguments must follow these naming conventions:

- If the `STAR_DIRECTORY` command argument is a relative path, you can use either a relative path or an absolute path in the `CORNERS_FILE` command. For example:

```
STAR_DIRECTORY: star_work
CORNERS_FILE: smc_config
```

- If the `STAR_DIRECTORY` command argument is an absolute path, you must use an absolute path in the `CORNERS_FILE` command. For example:

```
STAR_DIRECTORY: /tmp/star
CORNERS_FILE: /remote/.../work_directory/smc_config
```

## Operating Temperatures and Corner Types

Use the `OPERATING_TEMPERATURE` command within a corner definition to provide the operating temperature to be used with the nxtgrd file for extraction of that corner.

## Mapping Files

You can optionally specify a mapping file for each corner by including `MAPPING_FILE` commands in the corner definitions. If a corner definition does not include a `MAPPING_FILE` command, a global mapping file must be defined in the command file. If every corner includes a mapping file, a global mapping file is not necessary; if one exists, it is ignored.

All mapping files for simultaneous multicorners flows must be consistent in terms of the number of database and ITF file layers and their mapping relationships. Each mapping file category, such as conductors or vias, should also be consistent. The only allowed variations are the RPSQ value for conductors and the RPV value for vias.

## Via Coverage Option Files

You can optionally specify a via coverage option file for each corner by including `VIA_COVERAGE_OPTION_FILE` commands in the corner definitions. If one corner includes such a file, every corner must include one. If the corners file contains via coverage option files, global `VIA_COVERAGE_OPTION_FILE` commands in the command file are ignored.

The only variations allowed are resistance per via (RPV) variations. If two corners have the same nxtgrd file, they must use the same via coverage option file. However, different nxtgrd files can use the same via coverage option file.

---

## The `SELECTED_CORNERS` Command

The `SELECTED_CORNERS` command lists the corner names to be extracted and netlisted. The constraints for the `SELECTED_CORNERS` command are:

- Each corner name listed in the `SELECTED_CORNERS` command must match a corner name in the corners file.
- Corner names must be separated by spaces.
- The nxtgrd files associated with the extracted corners must share similar construction.  
For more information, see [Corner Constraints](#).

---

## Simultaneous Multicorner Flow Examples

The following example uses one netlist per corner. The three corners are the nxtgrd file named nominal.nxtgrd analyzed at two temperatures (-25 and 125°C) and the nxtgrd file named rcmax.nxtgrd analyzed at one temperature (25°C).

The command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: corners.smc
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
```

The corners file contains the following commands:

```
CORNER_NAME: NOM_T1
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: -25

CORNER_NAME: NOM_T2
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: 125

CORNER_NAME: RCMAX_T3
TCAD_GRD_FILE: rcmax.nxtgrd
OPERATING_TEMPERATURE: 25
```

The resulting output files are star\_NOM\_T1.spf, star\_NOM\_T2.spf, and star\_RCMAX\_T3.spf.

The following example uses three corners and one netlist per corner with the FSCOMPARE flow. The three corners are the nxtgrd file named nominal.nxtgrd analyzed at two temperatures (-25 and 125°C) and the nxtgrd file named rcmax.nxtgrd analyzed at one temperature (25°C). The corners file is the same as in Example 1.

The command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: corners.smc
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
EXTRACTION: FSCOMPARE
```

The resulting output files are star.fs\_comptot.NOM\_T1, star.fs\_compcoup.NOM\_T1, star.fs\_comptot.NOM\_T2, star.fs\_compcoup.NOM\_T2, star.fs\_comptot.RCMAX\_T3 and star.fs\_compcoup.RCMAX\_T3.

---

## Corner Constraints

The corners to be simultaneously extracted must obey the following constraints with respect to the ITF commands used to generate the corner nxtgrd files:

- All corner ITF files must have the same number and ordering of conducting layers with the same names, WMIN and SMIN values, and T0 values. Conducting layers in all corner ITF files must have the same covertical configuration. For example, if two conductors are covertical in one corner ITF, they must be covertical in all corners.
- All corner ITF files must have the same number and ordering of via layers with the same names, FROM layers, TO layers, and T0 values.
- If any corner ITF file conductor or via definition has a process variation table defined, the corresponding conductor or via definition in all other corner ITF files must also have the same variation table. However, the contents of the tables can be different for each corner. Exceptions to this requirement are specified in the reference pages for individual ITF statements.
- All corners must have the same values for the following ITF commands:
  - GATE\_TO\_CONTACT\_SMIN (within a CONDUCTOR block)
  - DENSITY\_BOX\_WEIGHTING\_FACTOR (within a CONDUCTOR block)
  - HALF\_NODE\_SCALE\_FACTOR
  - GLOBAL\_TEMPERATURE
  - USE\_SI\_DENSITY
  - REFERENCE\_DIRECTION
- The DROP\_FACTOR and TSV statements are not supported.

---

## Writing Formulas to the Netlist

You can optionally write the RC values from all selected corners into one netlist by writing them as formulas that use the corner names as variables. A simulation tool that reads the netlist file can set the corner name variables to 1 or 0 to enable or disable the use of the corners.

To enable this option, use the NETLIST\_SMC\_FORMULA: YES command in the command file and select the corners with the SELECTED\_CORNERS command. The allowable netlist formats for this option are SPF, NETNAME, and OA.

Setting the NETLIST\_SMC\_FORMULA command option to NO (the default) generates netlists according to the format specifications in the SELECTED\_CORNERS command.

In the following example, the `NETLIST_SMC_FORMULA: YES` command is used in a flow in which three corners are defined (`NOM_T1`, `NOM_T2`, and `RCMAX_T3`). The SPF file output contains lines similar to the following example:

```
* | NET A '0.63*NOM_T1+0.65*NOM_T2+0.75*RCMAX_T3' PF
Cg1 A:1 0 3.6e-17*NOM_T1+4.07e-17*NOM_T2+4.09e-17*RCMAX_T3
R162 A:1 A:2 4.8*NOM_T1+4.9*NOM_T2+4.8*RCMAX_T3
```

---

## Writing Parasitic Resistor Temperature Coefficients to the Netlist

The `TEMPERATURE_SENSITIVITY` command writes the parasitic resistor temperature coefficients `TC1` (`CRT1`) and `TC2` (`CRT2`) to the netlist for use by simulation tools.

The `TEMPERATURE_SENSITIVITY` command can be used with the simultaneous multicorner flow. In this case, the maximum number of selected process corners is three and the `OPERATING_TEMPERATURE` settings in the corners file are ignored. If the selected corners contain redundant `nxtgrd` files, the StarRC tool discards the redundant corners, determines the temperature coefficients for the remaining corners, and issues a warning message.

## The Galaxy Parasitic Database

The Galaxy Parasitic Database (GPD) is a distributed and scalable binary database for gate-level extraction results. The GPD provides these benefits:

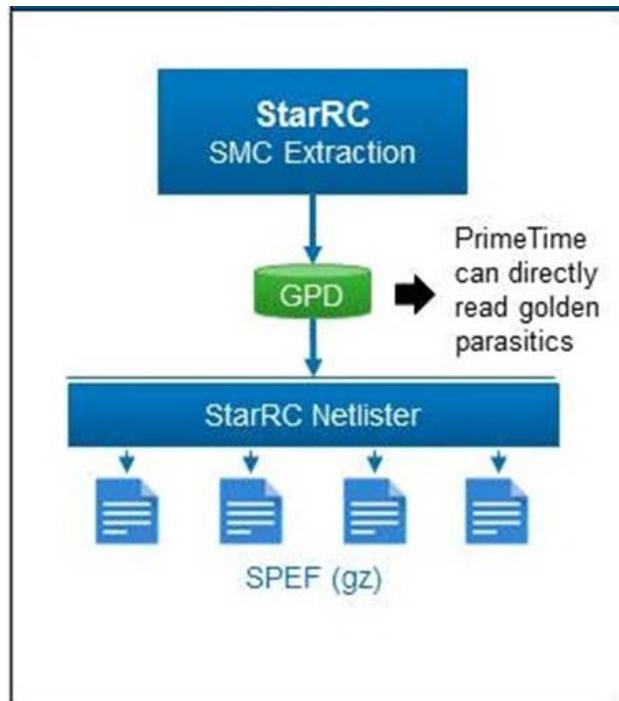
- Stores multicorner parasitic data
- Occupies less disk space than other parasitic data formats
- Enables faster data read and write times
- Reduces runtime by eliminating the translation and netlisting stages
- Can be read directly in the PrimeTime tool without generating a SPEF netlist

For gate-level flows, the StarRC tool saves extracted parasitic data in the GPD by default. Transistor-level flows do not use the GPD.

If you need a conventional netlist, you can create it directly from the parasitic database, either during the extraction run or at a later time.

[Figure 4-1](#) illustrates the relationship between the Galaxy Parasitic Database and the StarRC and PrimeTime tools.

*Figure 4-1 Galaxy Parasitic Database*



---

## Using the Galaxy Parasitic Database

The GPD flow is enabled by default for gate-level flows. You can optionally use the `GPD` command to specify the name of the directory in which to store the parasitic data. The default is to use the name specified by the `BLOCK` command.

Transistor-level, field solver, clock net inductance, and power extraction flows are not supported. In this case, the StarRC tool does not create a GPD directory.

### Ignored Commands

The following commands related to netlist generation are ignored.

- `NETLIST_MAX_LINE` (causes a warning message)
- `NETLIST_POSTPROCESS_COMMAND` (causes a warning message)

### Unsupported Commands

The following StarRC commands are not supported for the GPD flow. If these commands are used in the command file, the StarRC tool does not create a GPD directory.

- `EXTRACTION` settings other than `RC`
- `POWER_EXTRACT` settings other than `YES` or `NO`
- `INSTANCE_PORT` settings `NOT_CONDUCTIVE`, `MULTIPLE`, or `SUFFIXED`
- `CLOCK_NET_INDUCTANCE`
- `ECO` extraction commands
- `3D_IC*` commands
- `NETLIST_COUPLE_UNSELECTED_NETS`
- `NETLIST_RENAME_PORTS`
- `NETLIST_NAME_MAP`
- `NETLIST_SELECT_NETS`
- `NETLIST_TAIL_COMMENTS`
- `NETLIST_TYPE`
- `CONLY_NETS`
- `COUPLE_NONCRITICAL_NETS*` commands
- `SKIP_CELLS_COUPLE_TO_NET*` commands

- ZONE\_COUPLE\_TO\_NET\* commands
- SHEET\_COUPLE\_TO\_NET\* commands
- OBSERVATION\_POINTS
- PROBE\_POINTS
- PIO\_FILE
- POWER\_PORTS
- VIA\_COVERAGE\* commands

---

## Creating a Netlist from a Galaxy Parasitic Database

You can create a conventional netlist directly from the parasitic database, either during the extraction run or at a later time.

- To create a SPEF netlist during the extraction run, include the NETLIST\_FORMAT: SPEF and NETLIST\_FILE commands in the command file.
- To create a SPEF netlist from a saved GPD directory, use the following command, where *gpd\_dir* is the GPD directory and *spef\_file* is the name to use for the output SPEF file:

```
% StarXtract -convert_gpd_to_spef gpd_dir spef_file
```

If the StarRC command file used for the original extraction run contained commands for distributed processing, distributed processing is also used when creating a SPEF netlist from the GPD.

Note:

SPEF files created from a GPD database do not have routing\_conf or driver\_reduction sections to represent delay values for pi networks.

---

## The GPD Configuration File

After an extraction is complete, you can modify selected aspects of the stored data by using the GPD configuration file.

Follow this procedure:

1. Execute a StarRC extraction run that creates a GPD.
2. Generate the configuration file by executing the following command, where *gpd\_dir* is the name of the GPD directory:

```
% StarXtract -dump_gpd_config gpd_dir
```

3. Edit the configuration file.
4. Apply the edited configuration file to the GPD database:  
    % StarXtract -set\_gpd\_config *gpd\_dir config\_file*
5. Use the modified database to generate a new SPEF netlist:  
    % StarXtract -convert\_gpd\_to\_spef *gpd\_dir spef\_file*
6. (Optional) Reset the GPD to its original configuration:  
    % StarXtract -reset\_gpd *gpd\_dir*

### GPD Configuration File Commands

If the new configuration conflicts with the existing configuration, the tool issues an error message. For example, you cannot specify a coupling threshold that is smaller than the threshold used in the original extraction.

The following commands are allowed in a GPD configuration file:

- COUPLING\_ABS\_THRESHOLD
- COUPLING\_REL\_THRESHOLD
- COUPLING\_THRESHOLD\_OPERATION
- GPD\_DP\_STRING

This command is only valid in a GPD configuration file. If you are creating a SPEF netlist from a GPD, you can use this command in the configuration file to specify distributed processing conditions for netlist creation that are different from the distributed processing conditions used in the extraction run. Distributed processing in the extraction is controlled by the STARRC\_DP\_STRING command.

- NETLIST\_COMPRESS

This command is only valid in a GPD configuration file.

- NETLIST\_CONNECT\_SECTION
- NETLIST\_DELIMITER
- NETLIST\_PRECISION
- SELECTED\_CORNERS

The corners specified in the configuration file must be a subset of the corners extracted in the original run.

---

## Output Netlists

The StarRC tool can create several types of parasitics netlists for use as input to other tools. Each of them contains a slightly different structure and format. This section provides a simple example of each netlist type.

The available netlist types are as follows:

- Standard Parasitic Exchange Format (SPEF)

This is the format most commonly used for timing analysis.

- Standard Parasitic Format (SPF)

This format contains device-level information for input to simulation tools.

- NETNAME Format (transistor-level extraction only)

This is a SPICE-style output suitable for input to simulation tools.

- OpenAccess (OA) (transistor-level extraction only)

OpenAccess is an open standard data exchange format for circuit design.

---

## SPEF Netlist Example

```
*SPEF "IEEE 1481-1999"
*DESIGN "ALU"
*DATE "Wed April 17 19:50:14 2006"
*VENDOR "Synopsys"
*PROGRAM "StarRC"
*VERSION "K-2015.06"
*DESIGN_FLOW "PIN_CAP NONE" "NAME_SCOPE LOCAL"
*DIVIDER /
*DELIMITER :
*BUS_DELIMITER []
*T_UNIT 1 NS
*C_UNIT 1 FF
*R_UNIT 1 OHM
*L_UNIT 1 HENRY

*NAME_MAP
*5 net1
*10 insta/net2
*14 U1
*16 insta/U1
*17 insta/U2
```

```
*PORTS
*5 I *C 3.6 0

*D_NET *5 1.02e+00

*CONN
*P *5 I *C 3.6 0
*I *14:I I *C 6.76 8.94 *L 4 *D inv

*CAP
1 *5 0.60481
2 *14:I 0.413795

*RES
1 *5 *14:I 29.8492

*END

*D_NET *10 4.58e-01

*CONN
*I *16:ZN O *C 9.12 21.84
*I *17:I I *C 6.34 20.94 *L 4 *D inv

*CAP
1 *16:ZN 0.271701
2 *17:I 0.186

*RES
1 *16:ZN *17:I 16.8989

*END
```

---

## SPF Netlist Example

```
*
```

```
* | DSPF 1.0
* | DESIGN top
* | DATE "Wed April 17 13:34:43 2000"
* | VENDOR "Synopsys"
* | PROGRAM "StarRC"
* | VERSION "2003.2.0.0"
* | DIVIDER /
* | DELIMITER :
*
```

```
.SUBCKT top net1

* | GROUND_NET 0

* | NET net1 9.60e-04PF
* | P (net1 I 0 3.6 0)
* | I (U1:I U1 I I 4e-15 6.76 8.94)
R1 net1 U1:I 29.8492
Cg1 net1 0 5.8737e-16
Cg2 U1:I 0 3.72441e-16

* | NET insta/net2 5.10e-04PF
* | I (insta/U1:ZN insta/U1 ZN O 0 9.12 21.84)
* | I (insta/U2:I insta/U2 I I 4e-15 6.34 20.94)
R2 insta/U1:ZN insta/U2:I 16.8989
Cg3 insta/U1:ZN 0 2.96927e-16
Cg4 insta/U2:I 0 2.13328e-16
*
* Instance Section
*
Xinsta/U2 insta/U2:I in2 VDD VSS inv
Xinsta/U1 in1 insta/U1:ZN VDD VSS inv
XU1 U1:I net2 VDD VSS inv

.ENDS
```

## NETNAME Netlist Example

```
*
```

~~\* | DSPF 1.3~~
~~\* | DESIGN toprt~~
~~\* | DATE "Thu April 10 13:00:11 2001"~~
~~\* | VENDOR "Synopsys"~~
~~\* | PROGRAM "StarRC"~~
~~\* | VERSION "2003.2.0.0"~~
~~\* | DIVIDER /~~
~~\* | DELIMITER :~~
~~\*\*FORMAT STAR~~
~~\*~~
~~\* | GROUND\_NET 0~~
~~\* | NET min\_msb\_led[0] 1.06e-02PF~~
~~\* | P (min\_msb\_led[0] O 0 0 425.8)~~
~~\* | I (min\_msb\_led[0]:F485 min\_msb/conv\_blk1/U4 X 0 0 37 394)~~
~~Rmin\_msb\_led[0] min\_msb\_led[0] min\_msb\_led[0]:F1558 0.001~~
~~Cg1 min\_msb\_led[0]:F485 0 3.17002e-15~~
~~Cg2 min\_msb\_led[0]:F1558 0 7.44086e-15~~
~~R1 min\_msb\_led[0]:F485 min\_msb\_led[0]:F1558 12.105~~

```

* | NET min_lsb/cnt_blk1/n200 1.00e-02PF
* | I (min_lsb/cnt_blk1/n200:F191 min_lsb/cnt_blk1/U51 X O 0
63.9 49.8)
* | I (min_lsb/cnt_blk1/n200:F141 min_lsb/cnt_blk1/U53 C I 5e-
14 117 53)
Cg3 min_lsb/cnt_blk1/n200:F191 0 5.58775e-15
Cg4 min_lsb/cnt_blk1/n200:F141 0 4.42815e-15
R2 min_lsb/cnt_blk1/n200:F191 min_lsb/cnt_blk1/n200:F141
10.1364
*
* Instance Section
*
Xmin_lsb/cnt_blk1/U39 min_lsb/n100:F162 min_lsb/cnt_blk1/
n185:F164 VDD VSS INV2
Xmin_msb/cnt_blk1/U44 VDD min_msb/cnt_blk1/n216:F521 VDD
VSS INV2
Xmin_lsb/cnt_blk1/U45 min_lsb/n100:F235 min_lsb/cnt_blk1/
n195:F239 min_lsb/cnt_blk1/n190:F237 VDD VSS AND2
Xsec_msb/conv_blk1/U33 sec_msb/conv_blk1/n68:F1471
sec_msb_led[2]:F1475 sec_msb/conv_blk1/n67:F1473 VDD VSS
OR2
Xsec_msb/conv_blk1/U29 sec_msb/conv_blk1/n52:F1476
sec_msb/conv_blk1/n65:F1480 sec_msb/bcd[2]:F1478 VDD VSS
OR2

```

## NETLIST\_IDEAL\_SPICE\_FILE Output Example

```

* SPICE Netlist
* VENDOR "Synopsys, Inc."
* PROGRAM "StarRC"
* DATE "Thu April 16 16:26:00 2002"

**FORMAT SPICE

.SUBCKT AND2 B A OUT
XS1I1 B A S1N3 NAND2
XS1I2 OUT S1N3 INVA
.ENDS AND2

.SUBCKT AND3 C B A OUT
XS1I1 C B A S1N9 NAND3
XS1I2 OUT S1N9 INVA
.ENDS AND3

```

```

.SUBCKT CS_ADD1 SUM COUT C B A
XS1I2 B A S1N29 AND2
XS1I3 C S1N9 S1N31 AND2
XS1I4 S1N43 S1N35 S1N39 AND2
XS1I5 S1N29 S1N31 S1N35 NOR2
XS1I6 S1N41 S1N39 S1N37 NOR2
XS1I7 C B A S1N41 AND3
XS1I8 C B A S1N43 OR3
XS1I16 B A S1N9 OR2
XS1I33 COUT S1N35 INVA
XS1I34 SUM S1N37 INVA
.ENDS CS_ADD1

.SUBCKT INVA OUT IN
MS1I1 OUT IN GND GND N ad=39p as=39p l=1u pd=32u ps=32u w=13u
MS1I2 VDD IN OUT VDD P ad=61.5p as=61.5p l=1u pd=47u ps=47u
w=20.5u
.ENDS INVA

*.SUBCKT N D G S VBB
*.ENDS N

.SUBCKT NAND2 B A QN
MS1I1 S1N20 A GND GND N ad=13p as=39p l=1u pd=15u ps=32u w=13u
MS1I3 VDD B QN VDD P ad=61.5p as=46.125p l=1u pd=47u ps=25u
w=20.5u
MS1I4 VDD A QN VDD P ad=61.5p as=46.125p l=1u pd=47u ps=25u
w=20.5u
MS1I25 QN B S1N20 GND N ad=39p as=13p l=1u pd=32u ps=15u w=13u
.ENDS NAND2

.SUBCKT NAND3 C B A QN
MS1I1 S1N11 A GND GND N ad=19.5p as=39p l=1u pd=16u ps=32u
w=13u
MS1I4 VDD A QN VDD P ad=61.5p as=41p l=1u pd=47u ps=24.5u
w=20.5u
MS1I28 VDD B QN VDD P ad=35.875p as=41p l=1u pd=24u ps=24.5u
w=20.5u
MS1I29 VDD C QN VDD P ad=35.875p as=61.5p l=1u pd=24u ps=47u
w=20.5u
MS1I30 S1N32 B S1N11 GND N ad=19.5p as=19.5p l=1u pd=16u
ps=16u w=13u
MS1I31 QN C S1N32 GND N ad=39p as=19.5p l=1u pd=32u ps=16u
w=13u
.ENDS NAND3

```

```

.SUBCKT NOR2 B A QN
MS1I1 QN A GND GND N ad=29.25p as=39p l=1u pd=17.5u ps=32u
w=13u
MS1I2 QN B GND GND N ad=29.25p as=39p l=1u pd=17.5u ps=32u
w=13u
MS1I3 S1N5 B QN VDD P ad=20.5p as=112.75p l=1u pd=22.5u
ps=52u w=20.5u
MS1I4 VDD A S1N5 VDD P ad=61.5p as=20.5p l=1u pd=47u ps=22.5u
w=20.5u
.ENDS NOR2

.SUBCKT NOR3 C B A QN
MS1I1 QN A GND GND N ad=26p as=39p l=1u pd=17u ps=32u w=13u
MS1I2 QN B GND GND N ad=26p as=22.75p l=1u pd=17u ps=16.5u
w=13u
MS1I3 S1N5 B S1N25 VDD P ad=30.75p as=30.75p l=1u pd=23.5u
ps=23.5u w=20.5u
MS1I4 VDD A S1N5 VDD P ad=61.5p as=30.75p l=1u pd=47u ps=23.5u
w=20.5u
MS1I41 S1N25 C QN VDD P ad=30.75p as=82p l=1u pd=23.5u ps=49u
w=20.5u
MS1I42 QN C GND GND N ad=39p as=22.75p l=1u pd=32u ps=16.5u
w=13u
.ENDS NOR3

.SUBCKT OR2 B A OUT
XS1I1 B A S1N3 NOR2
XS1I2 OUT S1N3 INVA
.ENDS OR2

.SUBCKT OR3 C B A OUT
XS1I1 C B A S1N3 NOR3
XS1I2 OUT S1N3 INVA
.ENDS OR3

*.SUBCKT P D G S VBB
*.ENDS P

*.SUBCKT ADD4 S1N9 S1N7 S1N5 SUM3 SUM2 SUM1 SUM0 COUT CIN
B3 B2 B1 B0 A3 A2 A1 A0
XS1I1 SUM0 S1N5 CIN B0 A0 CS_ADD1
XS1I2 SUM1 S1N7 S1N5 B1 A1 CS_ADD1

XS1I3 SUM2 S1N9 S1N7 B2 A2 CS_ADD1
XS1I4 SUM3 COUT S1N9 B3 A3 CS_ADD1
*.ENDS ADD4

```

---

## Netlists For HSIM Reliability Analysis

Because you can set multiple commands in a StarRC command file when extracting a design for HSIM reliability analysis, often designers specify more design parameters than are needed. This leads to high memory use and large netlists. To remedy this, you can use the TARGET\_PWRA: YES command to generate a netlist relevant to the reliability analysis flow. You need only specify the power nets in the command file.

## Creating the Simplified Command File

To create the simplified command file, specify the commands as shown in the following example:

```
BLOCK:
MILKYWAY_DATABASE:
MILKYWAY_EXTRACT_VIEW:
TARGET_PWRA: YES
POWER_NETS: list_of_power_nets
TCAD_GRD_FILE:
NETLIST_INSTANCE_SECTION: YES | ALL
MAPPING_FILE:
XREF: YES
SKIP_CELLS:
```

Using the NETLIST\_INSTANCE\_SECTION command forces StarRC to netlist devices connected to an unextracted power net.

The TARGET\_PWRA command automatically includes the commands needed for power reliability analysis and overrides any commands of the same type that appear elsewhere in the command file. The only exception is the POWER\_REDUCTION command. The recommended option is the LAYER\_NO\_EXTRA\_LOOPS option; the TARGET\_PWRA command sets this option if no other instance of the POWER\_REDUCTION command appears in the command file. If you set the POWER\_REDUCTION command to YES in the command file, the TARGET\_PWRA command overrides it with the LAYER\_NO\_EXTRA\_LOOPS option. However, if you set the POWER\_REDUCTION command to the more conservative NO or LAYER options, those settings are not changed.

The TARGET\_PWRA: YES command causes the StarRC tool to generate two netlists. One netlist contains unreduced resistors for power nets. The other netlist contains reduced RC-coupled devices for signal nets. The signal netlist is useful for both reliability analysis and signal timing analysis.

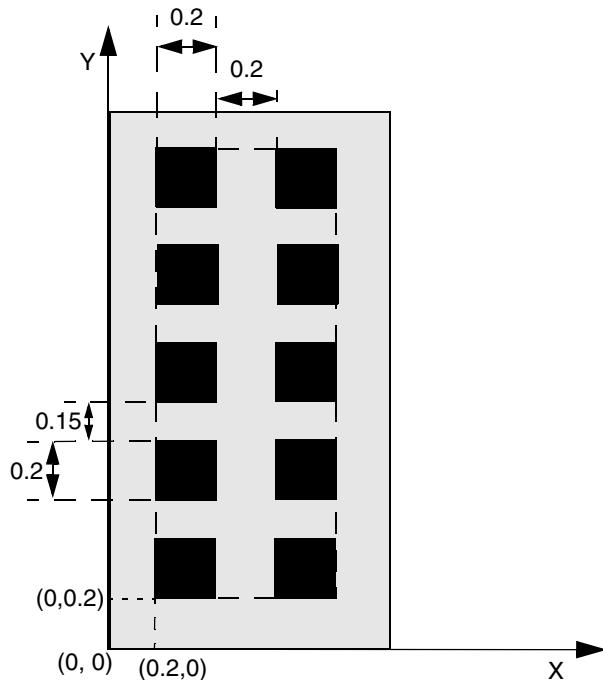
If you use the TARGET\_PWRA command, you must also use the REDUCTION: LAYER and KEEP\_VIA\_NODES: NO commands.

## SPF Geometry Visualization in HSIM

To perform SPF visualization of a merged vias with `MAX_VIA_ARRAY_SPACING` or `MAX_VIA_ARRAY_LENGTH` in the mapping file, specify the `NETLIST_TAIL_COMMENTS: YES`, `EXTRA_GEOMETRY_INFO: NODE`, and `KEEP_VIA_NODES: YES` commands in your command file.

[Figure 4-2](#) shows a 5-by-2 via array. Each via is 0.2-by-0.2-micron, vertical via-to-via spacing is 0.15 micron, and horizontal via-to-via spacing is 0.2 micron. If you specify `MAX_VIA_ARRAY_SPACING=0.3`, this via array is identified and extracted as one via resistor.

*Figure 4-2 Via Array*



If `NETLIST_TAIL_COMMENTS` is set to `YES`, the following is printed:

```
R45 29 32 0.2 $a=0.4 $lvl=5 $n=5x2 $p=4.4
```

Nodes 29 and 32 reside on the upper and lower layers connected by the via resistor. The via resistor value is calculated based on the total area of the vias represented by the `$a` parameter. The `$p` parameter represents the perimeter of the bounding box of the via array and is calculated as follows:

$$\text{perimeter} = (0.2 \times 2 + 0.2) \times 2 + (0.2 \times 5 + 0.15 \times 4) * 2 = 4.4$$

## Extraction For Electromigration Analysis

Electromigration refers to the detrimental physical effects of high current densities on integrated circuit features. As process geometries shrink, the effects of electromigration become more important.

StarRC extraction provides information for simulation tools that analyze electromigration risk. The location, resistance, and width of parasitic resistors are of primary importance.

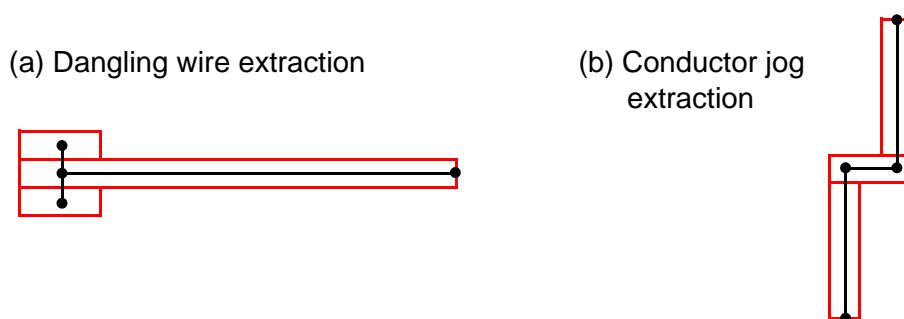
Use the following command settings in the StarRC command file if you plan to perform electromigration analysis in transistor-level flows:

- REDUCTION: NO
- EXTRA\_GEOMETRY\_INFO: NODE RES
- KEEP\_VIA\_NODES: YES
- SHORT\_PINS: NO
- NETLIST\_TAIL\_COMMENTS: YES
- NETLIST\_NODE\_SECTION: YES
- NETLIST\_CONNECT\_SECTION: YES

If you want to preserve a network node or subnode for individual vias, do not use via merging in the `via_layers` section of the layer map file. In other words, do not set the `MAX_VIA_ARRAY_SPACING` or `MAX_VIA_ARRAY_LENGTH` commands in the `via_layers` section.

Certain StarRC extraction methods are included for the purpose of providing accurate input for subsequent electromigration analysis. For example, the tool extracts the resistance for dangling wires. In [Figure 4-3\(a\)](#), a horizontal resistor is extracted for the dangling wire in addition to the two vertical resistors. The tool also accurately represents the current direction and resistance of conductors that have small jogs, as shown in [Figure 4-3\(b\)](#).

*Figure 4-3 Extraction Modifications for Electromigration Analysis*



# 5

## ECO Extraction

---

ECO extraction is the technique of performing extraction only on parts of a design that are different from a reference design. This capability allows efficient evaluation of engineering change orders, especially when coupled with timing analysis and place and route tools that work together.

This chapter includes the following sections:

- [ECO Extraction Overview](#)
- [The Galaxy Standard ECO Flow](#)
- [The Galaxy Incremental ECO Flow](#)
- [Changing the Working Directory Between ECO Iterations](#)

---

## ECO Extraction Overview

For timing closure and crosstalk analysis, iterative design and analysis runs are frequently necessary to identify and fix problems. The design changes are known as engineering change orders (ECOs). ECO changes might include localized gate placement and sizing changes, net rerouting, and net additions or deletions. After a design modification, parasitic extraction and timing analysis must be repeated to verify that the issues are resolved. Repeating the full-chip extraction and analysis cycle can be very time-consuming.

The StarRC tool provides several options to reduce turnaround time by performing extraction only on the ECO-affected nets, as follows:

- The Galaxy Standard ECO flow

The StarRC tool compares pre-ECO and post-ECO designs and performs extraction only on the affected nets, if a runtime benefit would result.

This flow supports all physical database formats supported in full-chip extraction mode, including IC Compiler II, Milkyway, and LEF/DEF designs. This flow generates full-chip parasitic netlists; therefore, no change is required for downstream timing analysis tools.

- The Galaxy Incremental ECO flow

The IC Compiler II, StarRC, and PrimeTime tools work together to share information about ECO design changes. This combined flow offers the largest overall turnaround time benefit.

ECO extraction is compatible with the following StarRC features:

- Distributed processing
- Simultaneous multicorner analysis

Limitations are as follows:

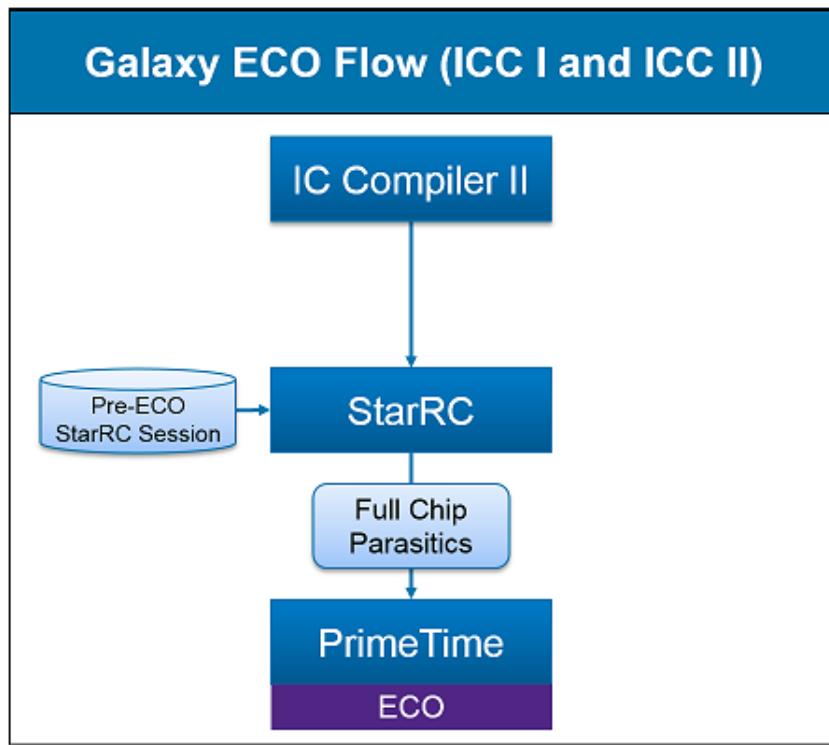
- Only SPEF netlists are supported.
- Field solver extraction is not supported.
- Only RC extraction is supported; the EXTRATION command must be set to RC.
- Power net extraction is not supported; the POWER\_EXTRACT command must be set to NO.
- The COUPLE\_TO\_GROUND command must be set to NO.

## The Galaxy Standard ECO Flow

In this flow, the StarRC tool compares pre-ECO and post-ECO designs and performs extraction only on the affected nets, if a runtime benefit would result. The PrimeTime tool reads full-chip parasitic netlists and performs full-chip update timing.

[Figure 5-1](#) illustrates the Galaxy ECO flow. This flow is suitable for use with IC Compiler II, Milkyway, and LEF/DEF designs.

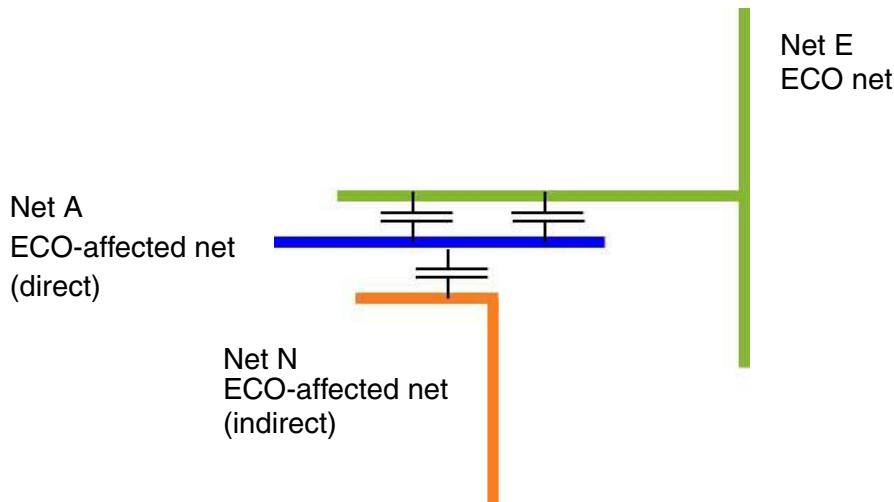
*Figure 5-1 ECO Extraction and Signoff Flow*



## Identification of Nets Affected by an ECO

[Figure 5-2](#) shows the types of nets that might be affected by a design change. Net E, the ECO net, is a net that is modified as part of a timing violation fix. Net A is coupled to net E; this type of net is a directly ECO-affected net. Net N is not coupled directly to net E, but is coupled to net A. This type of net is an indirectly ECO-affected net.

The StarRC tool includes net E and net A, but not net N, in the ECO netlist (the file specified by the `NETLIST_ECO_FILE` command). The coupling capacitance between net A and net N appears as a dangling capacitance under net A in the ECO netlist. The PrimeTime tool can read the ECO netlist and adjust the coupling and total capacitance of net N accordingly.

*Figure 5-2 ECO-Affected Nets*

If a cell is swapped with a footprint-compatible cell without changing the instance name and connected wires, the StarRC tool does not re-extract the connected wires or replace them in the netlist. The PrimeTime tool identifies the correct cell name for timing analysis based on the instance name in the ECO Verilog file.

## Galaxy Standard ECO Flow With Full-Chip Update Timing

This procedure is suitable for most design types. The StarRC tool detects the differences between the pre-ECO design and the post-ECO design and performs either a full-chip or incremental extraction to achieve the best results with the shortest runtime.

In this flow, the PrimeTime tool uses the full-chip netlist to perform full-chip update timing.

1. Include the following commands in the StarRC command file. The database and file names are examples; modify them as needed. This example is for the Milkyway flow.

```
MILKYWAY_DATABASE: CPU.mw
BLOCK: top_block_rev0
ECO_MODE: YES
STAR_DIRECTORY: star
NETLIST_FILE: eco_full_chip.spf
SUMMARY_FILE: eco.star_sum
```

2. Execute a baseline StarRC run.

The first run is a full-chip extraction and the resulting netlist is saved under the name specified by the `NETLIST_FILE` command.

3. If you are using the PrimeTime tool, modify the `read_parasitics` command in the script to include the netlist name:

```
read_parasitics -format spef -keep_capacitive_coupling \
./eco_full_chip.spef
```

4. Execute a baseline timing analysis run.
5. Implement fixes for timing violations and save the changes in the design database.
6. Modify the StarRC command file as follows:

- Edit the database name commands, if needed to point to the modified design.
- (Optional) Edit the `NETLIST_FILE` command to change the file name for the ECO extraction netlist. Otherwise, the previous netlist is overwritten.
- (Optional) Edit the `SUMMARY_FILE` command to change the file name. Otherwise, the previous summary file is overwritten.

7. Execute another StarRC run.

The extraction might be either a full-chip extraction or an ECO extraction, depending on the number of ECO changes. Informational messages indicate whether a full or ECO extraction is being performed. For ECO extractions, the percentage of ECO nets is also reported.

In either case, the full-chip netlist is saved in the file specified by the `NETLIST_FILE` command.

8. Perform the timing analysis again and check for timing violations. If necessary, modify the input script to point to the revised design and the updated parasitics file.
9. Repeat steps 5 to 8 until timing closure is achieved.
10. For chip signoff, perform a final full-chip extraction and timing analysis.

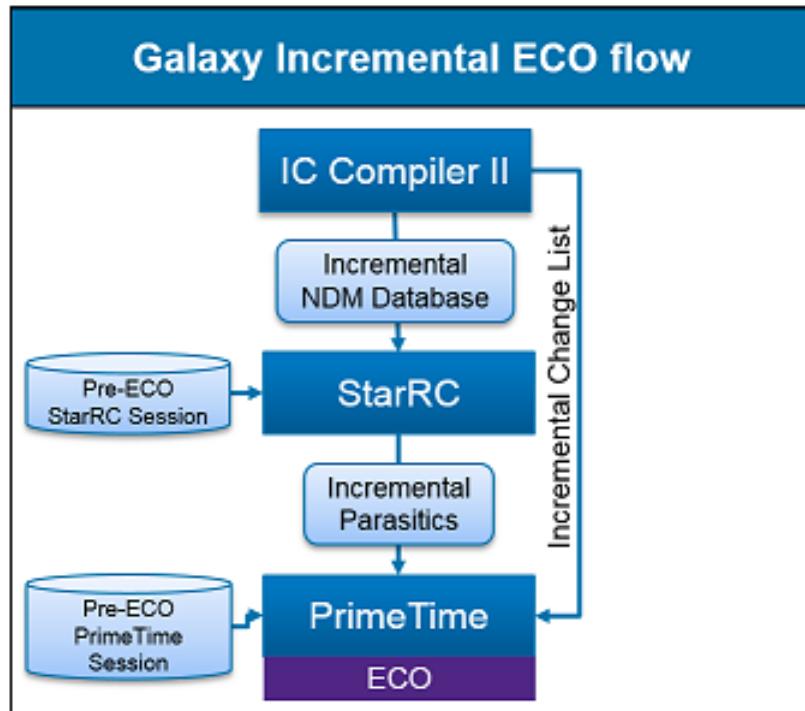
## The Galaxy Incremental ECO Flow

In this flow, the IC Compiler II, StarRC, and PrimeTime tools work together to share information about ECO design changes. The combined flow offers the largest overall turnaround time benefit.

For more information about procedures and commands in the IC Compiler II tool, see the *IC Compiler II Implementation User Guide*. For more information about procedures and commands in the PrimeTime tool, see the *PrimeTime User Guide*.

[Figure 5-3](#) illustrates the Galaxy incremental ECO flow. The IC Compiler II tool keeps track of physical and logical design changes during ECO iterations. The StarRC tool updates the extracted parasitics using the physical design changes. The PrimeTime tool updates the timing using the logical design changes and the updated parasitics.

*Figure 5-3 ECO Extraction and Signoff Flow*



---

## Galaxy Incremental ECO Flow With Incremental Update Timing

In this flow, the PrimeTime tool performs incremental update timing analysis.

1. Create an IC Compiler II design. Use the following command in the IC Compiler II tool to initialize the incremental database:

```
icc2_shell> record_signoff_eco_changes -init -def
```

2. Include the following commands in the StarRC command file. The database and file names are examples; modify them as needed. Note the `NETLIST_INCREMENTAL` command.

```
NDM_DATABASE: design1_pre_eco.ndm
ECO_MODE: YES
STAR_DIRECTORY: star
NETLIST_INCREMENTAL: YES
NETLIST_FILE: pre_eco_file.spf
SUMMARY_FILE: eco.star_sum
```

3. Execute a baseline StarRC run.

The resulting netlist is saved under the name specified by the `NETLIST_FILE` command.

4. Execute a baseline PrimeTime full update timing run, using the `read_parasitics` command with the parasitics netlist.

```
pt_shell> read_parasitics pre_eco_file.spf
pt_shell> update_timing -full
```

5. In the PrimeTime tool, specify ECO design changes to address timing violations and save the changes to a Tcl script file.

6. In the IC Compiler II tool, implement the specified ECO changes. Use the following command to record the incremental design database changes, where `Tcl_change_file` is the file created in step 5:

```
icc2_shell> record_signoff_eco_changes -start -input Tcl_change_file
```

To conclude the ECO operations, use the `record_signoff_eco_changes -stop -def` command. You might choose a different design or block name for this version of the design.

7. Modify the StarRC command file as follows:

- Edit the `NDM_DATABASE` and `BLOCK` commands, if needed to point to the modified design.
- Edit the `NETLIST_FILE` command to change the file name for the ECO extraction netlist.

- (Optional) Edit the `SUMMARY_FILE` command to change the file name. Otherwise, the previous summary file is overwritten.
8. Execute the StarRC ECO extraction run on the modified design.

The extraction run might be either a full-chip extraction or an ECO extraction, depending on the amount of design changes and the runtime benefit determined by the StarRC tool.
  9. Modify the PrimeTime script as follows:
    - Edit the `read_eco_changes` command to point to the modified design.
    - Edit the `read_parasitics` command to add the `-eco` option and to point to the parasitics file from the StarRC ECO extraction run.
    - Edit the `update_timing` command to remove the `-full` option.
  10. Execute the timing run and check for timing violations.
  11. Repeat steps 5 to 10 until timing closure is achieved.
  12. For chip signoff, perform a final full-chip extraction and timing analysis.

---

## Galaxy Incremental ECO Flow With Full-Chip Update Timing

In this flow, the PrimeTime tool always performs full-chip timing analysis.

1. Create an IC Compiler II design. Use the following command in the IC Compiler II tool to initialize the incremental database:

```
icc2_shell> record_signoff_eco_changes -init -def
```

2. Include the following commands in the StarRC command file. The database and file names are examples; modify them as needed.

```
NDM_DATABASE: design1_pre_eco.ndm
ECO_MODE: YES
STAR_DIRECTORY: star
NETLIST_FILE: pre_eco_file.spf
SUMMARY_FILE: eco.star_sum
```

3. Execute a baseline StarRC run.

The resulting netlist is saved under the name specified by the `NETLIST_FILE` command.

4. Execute a baseline PrimeTime timing analysis, using the `read_parasitics` command with the parasitics netlist.

```
pt_shell> read_parasitics pre_eco_file.spf
pt_shell> update_timing -full
```

5. In the PrimeTime tool, specify ECO design changes to address timing violations and save the changes to a Tcl script file.

6. In the IC Compiler II tool, implement the specified ECO changes. Use the following command to record the incremental design database changes, where `Tcl_change_file` is the file created in step 5:

```
icc2_shell> record_signoff_eco_changes -start -input Tcl_change_file
```

To conclude the ECO operations, use the `record_signoff_eco_changes -stop -def` command. You might choose a different design or block name for this version of the design.

7. Modify the StarRC command file as follows:

- Edit the `NDM_DATABASE` and `BLOCK` commands, if needed to point to the modified design.
- Edit the `NETLIST_FILE` command to change the file name for the ECO extraction netlist.
- (Optional) Edit the `SUMMARY_FILE` command to change the file name. Otherwise, the previous summary file is overwritten.

8. Execute the StarRC ECO extraction run on the modified design.  
The extraction run might be either a full-chip extraction or an ECO extraction, depending on the amount of design changes and the runtime benefit determined by the StarRC tool.
9. In the PrimeTime script, modify the `read_parasitics` command to point to the updated parasitics netlist.
10. Execute the PrimeTime update timing run and check for timing violations.
11. Repeat steps 5 to 10 until timing closure is achieved.
12. For chip signoff, perform a final full-chip extraction and timing analysis.

---

## Changing the Working Directory Between ECO Iterations

The recommended ECO flow is to maintain the same working directory, star directory, and file names for the entire ECO signoff loop. However, options for the `StarXtract` command allow you to save and restore the ECO database during the ECO loop. This procedure applies to all ECO flows.

The save and restore options are as follows:

- The `-save_eco` option

No extraction is performed. The StarRC tool packages all of the files related to the ECO extraction into one file, which is compressed and saved as `destination_file`. The command file specifies the star directory name to be processed. The syntax is as follows:

```
% StarXtract -save_eco destination_file command_file
```

- The `-restore_eco` option

No extraction is performed. The StarRC tool uncompresses the file named in the `source_file` argument and copies the resulting files to the star directory named in the command file. The syntax is as follows:

```
% StarXtract -restore_eco source_file command_file
```

The save and restore commands use the linux `tar` utility by default. You can specify the location of the executable in your computing environment by using the `ECO_SAVE_COMMAND` and `ECO_RESTORE_COMMAND` commands in the StarRC command file.

The save and restore operations can be performed in the background during the timing analysis and ECO implementation phases of the signoff loop.

A simple procedure to change the working directory between ECO cycles is as follows:

1. Select a working directory.

```
% cd work_dir_1
```

2. Run the first ECO extraction.

```
% StarXtract eco_1.cmd
```

3. Save the ECO session to a file. You can perform this step in the background while you use the extraction results to perform timing analysis or work on design modification.

```
% StarXtract -save_eco eco_data_1.tar.gz eco1.cmd
```

4. Select a different working directory.

```
% cd work_dir_2
```

5. Restore the saved ECO session. You can perform this step in the background while you use the extraction results to perform timing analysis or work on design modification.

The command file used in this step must be modified to point to the revised design database. You can optionally change the star directory name.

```
% StarXtract -restore_eco eco_data_1.tar.gz eco2.cmd
```

6. Run another ECO extraction with the same command file used for the restore operation (necessary to maintain the same star directory name).

```
% StarXtract eco_2.cmd
```

7. Save the second ECO session to a file. You can perform this step in the background while you use the extraction results to perform timing analysis or work on design modification.

```
% StarXtract -save_eco eco_data_2.tar.gz eco2.cmd
```

8. Repeat ECO cycles as needed.

# 6

## The StarRC Field Solver

---

The StarRC field solver is a 3-D capacitance extraction tool that solves electrostatic equations using statistical random-walk methods. The field solver is used to verify the accuracy of StarRC results or to provide enhanced extraction accuracy for selected nets.

The following sections describe the field solver:

- [Overview of Field Solver Extraction](#)
- [Running the Field Solver](#)
- [Controlling Field Solver Accuracy and Runtime](#)
- [Distributed Processing for Field Solver Jobs](#)

---

## Overview of Field Solver Extraction

The field solver extracts capacitance by solving the three-dimensional Laplace equations. By contrast, standard pattern-based extraction compares layout structures to a library of structures for which capacitances have already been determined. The field solver provides greater accuracy at the expense of runtime.

You can control the accuracy and runtime of the field solver. In addition, the tool can run on a single CPU, multiple threads within the same CPU, or in distributed processing mode.

---

## Capacitance Types

The field solver reports the following capacitances at a node:

- Total capacitance – The capacitance from the net to all other objects in the simulation.
- Coupling capacitance (xcap) – The capacitance between two nets that are not part of the same net group.

Each capacitance type is reported in a separate section of the field solver capacitance report.

---

## Boundary Conditions

Boundary conditions are applied at the six edges of the simulation domain. By default, the field solver uses a Dirichlet boundary condition of 0 volts. This is equivalent to placing a ground plane along each face of the simulation cube. These ground planes are connected to the field solver global ground net.

In general, the total node capacitance obtained with the Dirichlet boundary condition used in the field solver is higher than the total capacitance obtained with Neumann boundary conditions used in the RC2 and RC3 modes of the Raphael tool. As the boundary moves further from the simulated electrodes, the effect becomes smaller.

You can select Neumann boundary conditions in the field solver by adding the `-neuman_x` and `-neuman_y` options to the `FSCOMPARE_OPTIONS` statement in your StarRC command file.

The following example specifies the use of Neumann boundary conditions in both the x- and y-directions:

```
FSCOMPARE_OPTIONS: -neuman_x -neuman_y
```

You also can specify periodic boundary conditions on the x- or y-direction with the `-periodic_x` and `-periodic_y` options to the `FSCOMPARE_OPTIONS` statement in your StarRC command file. Periodic boundary conditions cause the random walks to wrap

around to the opposite side of the device. For example, a walk that exits the device at the positive x-side reenters at the negative x-side of the device. Periodic boundary conditions are useful for devices with repeated cells such as memory circuits.

The following example specifies the use of periodic boundary conditions in both x- and y-directions:

```
FSCOMPARE_OPTIONS: -periodic_x -periodic_y
```

---

## Conductor Types

Structures in the field solver are composed of conductors and dielectrics. Dielectrics are typically derived from the technology file. Conductors are composed of metal boxes or planar boxes.

The field solver uses the following conductor types:

- [Nets](#)
- [Net Groups](#)
- [Ground Nets](#)
- [Fill Nets](#)

Conductors are composed of collections of metal boxes. All the metal boxes in a conductor are connected and at the same potential (voltage). The following sections describe the types of conductors and their reporting rules.

### Nets

A net conductor is the most common general type of conductor. Random walks are started only from nets and net groups. The field solver reports capacitances between nets and from nets to all other types. Nets are created by placing metal boxes or planar boxes between a net statement and an end statement. Net statements are specified in the encrypted design file to describe and identify the nets.

### Net Groups

A net group is a collection of nets that are electrically connected and therefore have the same potential (voltage). The field solver extracts the capacitance between nets in different net groups but not between nets in the same net group. Net groups are specified in the design file.

## Ground Nets

If Dirichlet boundary conditions are used as the default, the edges of the bounding box are electrical ground planes that form a single net called ground. In addition, in the design file, some nets might be identified as ground. The field solver reports the capacitance from other nets to ground, but because analysis walks are not started from ground, the total capacitance of ground is not reported.

## Fill Nets

Fill nets are used to model fill metal polygons. Metal fill consists of boxes inserted into a metal layer to improve the planarity of the metal layer. A fill net is electrically floating—that is, not connected to any circuit elements in the netlist.

The electronic potential at fill nets is determined by setting the charge on the fill net set to zero. Even though fill nets are not electrically connected, they can introduce capacitive coupling effects between other nets. The charge on a fill net is calculated as shown in [Equation 6-1](#).

*Equation 6-1 Calculation of Charge on a Fill Net*

$$Q_1 = \frac{V_1 C_1 C_0 + (V_1 - V_2) C_1 C_2}{C_0 + C_1 + C_2}$$

The effective capacitance at net 1 is calculated as shown in [Equation 6-2](#).

*Equation 6-2 Calculation of Effective Capacitance Between Fill Nets*

$$C_{\text{eff}} = \frac{C_1 C_0 + C_1 C_2}{C_0 + C_1 + C_2}$$

The coupling capacitance between net 1 and net 2 is calculated as shown in [Equation 6-3](#).

*Equation 6-3 Calculation of Coupling Capacitance*

$$C_c = \frac{C_1 C_2}{C_0 + C_1 + C_2}$$

---

## Running the Field Solver

The StarRC tool includes two methods of using field solver extraction:

- The `FSCOMPARE` flow

Reports the differences between the pattern-based extraction results of the StarRC tool and the random-walk extraction results of the field solver

- The `FS_EXTRACT_NETS` flow

Uses the field solver to perform extraction on critical nets and standard pattern-based extraction on the remainder of the design

[Table 6-1](#) summarizes the differences between the flows.

You can customize the field solver extraction by using the `FSCOMPARE_OPTIONS` command, which applies to both of the field solver flows.

*Table 6-1 Comparison of Field Solver Flows*

|                                                                                       | <b>FSCOMPARE flow</b>                                                                                                                                                                                                             | <b>FS_EXTRACT_NETS flow</b>                                                                                                                               |
|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Command to invoke flow                                                                | <code>EXTRACTION: FSCOMPARE</code>                                                                                                                                                                                                | <code>FS_EXTRACT_NETS</code>                                                                                                                              |
| Default convergence goal set by the <code>FSCOMPARE_OPTIONS -perc_self</code> command | 0.5%                                                                                                                                                                                                                              | 1.5%                                                                                                                                                      |
| Output files                                                                          | Generates the <code>.fs_comptot</code> and <code>.fs_compcoup</code> output files, which report the differences between the pattern-based extraction results of StarRC and the random-walk extraction results of the field solver | Incorporates the extracted capacitances into the resulting netlist; does not generate <code>.fs_comptot</code> and <code>.fs_compcoup</code> report files |

---

## Output Files for the FSCOMPARE Flow

The field solver generates a report with the extension `.fs_comptot` for general capacitances and another report with the extension `.fs_compcoup` for coupling capacitances. The two file types contain data in the same format. The nets included in these reports are filtered by the values of the `FSCOMPARE_THRESHOLD` command for general capacitances and the `FSCOMPARE_COUPLING_THRESHOLD` command for coupling capacitances.

In the first section of the output files, the values in the FS column represent the capacitances calculated by the field solver, as shown in [Example 6-1](#). The percentages in parentheses

represent the corresponding statistical uncertainty in the field solver results. The values in the xTract column represent the capacitances calculated by the StarRC tool using its default pattern-based extraction. The values are reported in order of the absolute value of the per-net correlation reported in the first column, from largest to smallest. The first section of the report also contains overall statistics.

**Example 6-1 First Part of .fs\_comptot File (Unshorted Nets)**

| %Diff  | AbsError(fF) | FS(fF)         | xTract(fF) | NetName |
|--------|--------------|----------------|------------|---------|
| 0.55%  | 0.02676      | 4.89356 (1.2%) | 4.92032    | D23     |
| -0.49% | 0.0566781    | 11.5645 (0.5%) | 11.5078    | D7      |
| 0.24%  | 0.0278593    | 11.551 (0.5%)  | 11.5789    | S7      |
| -0.18% | 0.011445     | 6.49725 (1.0%) | 6.4858     | D22     |

If shorted nets are present, they are listed in the second section of the file, as shown in [Example 6-2](#). The second section includes per-net correlations but no overall statistics.

**Example 6-2 Second Part of .fs\_comptot File (Shorted Nets)**

| %Diff   | AbsError(fF) | FS(fF)         | xTract(fF) | ShortType | NetName |
|---------|--------------|----------------|------------|-----------|---------|
| 91.17%  | 0.417215     | 0.45763 (0.7%) | 0.87485    | Drawn     | UNSIG_2 |
| -19.54% | 0.238049     | 1.21858 (0.5%) | 0.980529   | Drawn     | d       |
| -12.68% | 0.373508     | 2.94479 (0.3%) | 2.57128    | Drawn     | vss     |

## Controlling Field Solver Accuracy and Runtime

In the field solver, the runtime has an inverse quadratic dependency on the accuracy. For example, reducing the accuracy tolerance by a factor of three (such as, from 3 percent to 1 percent) generally results in a ninefold increase in CPU time. The default accuracy tolerance in the field solver is 1 sigma at 0.5 percent for total capacitance. This means that the error incidence in total capacitance for 68 percent of the nets extracted is less than 0.5 percent. You can change the default 0.5 percent for total capacitance convergence to a different value using the `-perc_self` option with the `FSCOMPARE_OPTIONS` statement of your StarRC command file.

For example, to specify 1 percent as the requirement for total capacitance convergence, use the following command in the StarRC command file:

```
FSCOMPARE_OPTIONS: -perc_self 1
```

In general, the runtime is proportional to the number of nets extracted. The runtime is also dependent on the overall size of the layout (number of boxes or polygons). Increasing the number of dielectric layers can also cause the runtime to increase because the number of hops required for a walk to reach a conductor increases.

---

## Specifying Convergence Goals

A net is considered to be convergent if it satisfies the following relations:

$$Ec < \text{perc\_self} \times C \text{ & for each } C_{ij} \{ E_{ij} < \text{abs\_coup} + C_{ij} \times \text{perc\_coup} \}$$

In this equation,

- $Ec$  = estimated statistical error on the total net capacitance
- $C$  = total net capacitance
- $C_{ij}$  = coupling capacitance from net  $i$  to a neighbor net  $j$
- $E_{ij}$  = estimated statistical error on  $C_{ij}$
- Default of  $\text{abs\_coup} = C \times \text{coup\_cap\_thresh}$

Small coupling capacitors can be very slow to converge. To determine the value of a coupling capacitor between two electrodes A and B, a walk must start on electrode A and end on electrode B. The smaller the coupling capacitor, the smaller the probability that this happens, and a larger number of total walks is needed to obtain accurate statistics on the coupling capacitor.

For example, if the total capacitance at a node is 1e-15 farads and a coupling capacitor to the same node is 1e-18 farads, then 1000 more walks are needed to calculate the coupling capacitance value. Therefore, extracting the coupling capacitor to the same accuracy as the net total capacitance takes 1000 times longer. Usually the largest capacitances are the most important and converge the fastest in the field solver.

By default, the field solver does not check the percentage convergence of coupling capacitance. However, you can force the field solver to check the percentage convergence of coupling capacitance by using the `-perc_coup` option.

For example, to set the convergence of coupling capacitance to 10 percent, use the following command in the StarRC command file:

```
FSCOMPARE_OPTIONS: -perc_coup 10
```

By default, the field solver sets the threshold for convergence of coupling capacitance to 1 percent of the total net capacitance. To change the default, use the `-coup_cap_thresh` option.

For example, to set the absolute convergence of coupling capacitance to 2 percent of the total net capacitance, use the following command in the command file:

```
FSCOMPARE_OPTIONS: -coup_cap_thresh 2
```

Typically, it is not necessary to set coupling capacitance goals. If you do need to set them, make sure you fully understand the convergence criteria because incorrectly setting `-perc_coup` and `-abs_coup` can result in unnecessarily long runtimes.

There is no way to force the field solver to evaluate a specific coupling capacitor. Coupling capacitors are evaluated in random order, based on where the random walks start and end.

---

## Specifying the Accuracy Goal

You can specify the accuracy of the extracted total capacitance without having to calculate the convergence level in terms of the standard deviation. In this case, the tool automatically chooses the appropriate convergence tolerance.

To specify convergence in this manner, you can set the `-perc_accuracy` and the `-perc_accuracy_confidence` options. The default of the `-perc_accuracy_confidence` option is 99.7 percent. For example, if you set `-perc_accuracy` to 5 percent and leave `-perc_accuracy_confidence` at its default, the extracted total capacitance value is accurate to 5 percent with a confidence level of 99.7 percent. The 99.7 percent confidence interval about the mean of a Gaussian distribution is  $3\sigma$ . This is equivalent to setting `-perc_self 1.67` (that is, the standard deviation is  $5\% \div 3 = 1.67\%$ ).

---

## Specifying the Consistency of Results

In a random walk field solver, each net has a statistical error associated with the result. If a design contains 1000 identical nets, then the extracted values vary somewhat because of this error. The statistical error in the field solver follows a normal distribution. The standard deviation of the distribution sigma ( $\sigma$ ) is controlled by the `-perc_self` option.

For example, because the computed capacitance values follow the normal distribution, 68 percent of the nets lie within one sigma of the correct value ( $\mu$ ); that is, they occur between  $\mu-\sigma$  and  $\mu+\sigma$ , or within  $2\sigma$  of each other. Thus, the number of nets consistent to within  $2\sigma$  is 68 percent.

As an alternate way of setting the consistency, you can use the `-perc_consistency` and `-perc_of_nets_consistent` options. Use these options to calculate a value for the `-perc_self` option.

For example, to specify that 99.7 percent of the identical nets must have errors less than 1 percent, you can use either of the following equivalent commands:

- `FSCOMPARE_OPTIONS: -perc_consistency 1 -perc_of_nets_consistent 99.7`
- `FSCOMPARE_OPTIONS: -perc_self 0.167`

---

## Specifying Pattern Matching for Symmetric Nets

Memory designs often contain thousands of identical or symmetric nets with the same capacitance characteristics. The field solver uses a pattern matching process to identify these nets, analyze only a single representative net, and copy the capacitance characteristics to all matching nets. Using pattern matching can speed up the analysis of memory designs as well as ensure well-matched capacitance values for identical or symmetric nets.

To enable pattern matching, specify the `-match` option in the `FSCOMPARE_OPTIONS` command.

---

## Distributed Processing for Field Solver Jobs

During distributed processing, the field solver distributes the nets to be extracted among the available processors. One StarRC license enables up to four field solver distributed processing jobs

Specify the number of processors with the `FSCOMPARE_OPTIONS: -np` command and the number of threads per processor with the `FSCOMPARE_OPTIONS: -nt` command. If you specify both commands, the same number of threads is used on each processor.

For example, the following statement specifies to use four processors:

```
FSCOMPARE_OPTIONS: -np 4
```

The following statement specifies to use two threads on each of two processors:

```
FSCOMPARE_OPTIONS: -np 2 -nt 2
```

If the total number of specified threads (the product of the number of processors and the number of threads per processor) exceeds the number of nets to be extracted, the field solver uses only the same number of threads as nets to be extracted to avoid using unnecessary threads. Using distributed processing on small jobs that would only take a few minutes does not typically improve runtime because of the startup time for the machines.

Distributed processing of the field solver can tolerate machine failures. If a field solver job on one machine or thread is lost during a run, the lost job is taken over by the remaining available machines or threads to ensure that the run is successful.

---

## Setting up Distributed Processing

The job submission command can be specified either in an environment variable or as a command in the StarRC command file. If set in both places, the StarRC command file takes precedence. The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- [LSF System](#)
- [Oracle Grid Engine](#)
- [General Network With a List of Machines](#)
- [Runtime Design Automation System](#)

### LSF System

The submission command for LSF systems is `bsub`. You can specify it in either of the following forms:

- An environment variable before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example,  

```
% setenv FS_DP_STRING 'bsub -R "rusage[mem=5000]"'
```
- A statement in the StarRC command file. For example,  

```
FS_DP_STRING: bsub -R "rusage[mem=5000]"
```

### Oracle Grid Engine

The submission command for Oracle Grid Engine systems is `qsub`. You can specify it in either of the following forms:

- An environment variable before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example,  

```
% setenv STARRC_DP_STRING 'qsub -P bnnormal -l "mem_free=1G" \
-v "STARRC_LICENSE_WAIT=YES"'
```
- A statement in the StarRC command file. For example,  

```
STARRC_DP_STRING: qsub -P bnnormal -l "mem_free=1G" \
-v "STARRC_LICENSE_WAIT=YES"
```

To pass environment variables to the job, you must use the `-v` option. In this example, the `STARRC_LICENSE_WAIT` environment variable is set to allow license queuing.

## General Network With a List of Machines

For a general network with a list of machines, the syntax is

```
FS_DP_STRING: list host1[:n1] [host2[:n2] ... hostm[:nm]]
```

where `host1:n1` means that you are submitting `n1` jobs (an integer number of jobs) to the machine with name `host1`. The default number of jobs per machine is 1. Do not use spaces inside the `host:n` syntax; use one or more spaces between host machines. Each machine must be available through a simple UNIX `rsh` command without a password.

The keyword for the machine where the job is started is `localhost`. If you use `localhost`, system calls are used instead of `rsh` to submit the jobs.

You can specify the `FS_DP_STRING` variable in either of the following forms:

- An environment variable before launching the tool. Enclose the list in single quotation marks because it might contain multiple arguments. For example,

```
% setenv FS_DP_STRING 'list alpha beta gamma'
```

- A statement in the StarRC command file. For example,

```
FS_DP_STRING: list alpha:1 beta:4 gamma:2
```

## Runtime Design Automation System

The submission command for a Runtime Design Automation (RTDA) system is `nc run`. You can specify it in either of the following forms:

- An environment variable before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example,

```
% setenv FS_DP_STRING 'nc run -R "rusage[mem=5000]"'
```

- A statement in the StarRC command file. For example,

```
FS_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```



# 7

## The Graphical User Interface

---

The StarRC graphical user interface (GUI) provides an interactive environment for the extraction and analysis of physical designs. Any StarRC extraction can be configured and executed from the GUI. The GUI also provides post-layout verification and analysis tools. In addition, StarRC includes the Milkyway layout viewer.

This chapter covers running the StarRC GUI in the following sections:

- [Invoking the Graphical User Interface](#)
- [Using the GUI to Run a Timing or Noise Analysis](#)
- [Using the GUI to Run a SingleShot Analysis](#)
- [StarRC Graphical User Interface Reference](#)
- [Entering Information in the StarRC GUI](#)
- [Editing a Mapping File in the StarRC GUI](#)

---

## Invoking the Graphical User Interface

To start the GUI, enter the following command from the StarRC working directory:

```
% StarXtract -gui
```

The files needed to run the StarRC tool are listed in [Table 7-1](#). Specify the file locations in the StarRC command file.

*Table 7-1 Files Needed to Run StarRC*

| File                        | Definition                                                                                                                                               |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Design database             | Layout database in one of the following formats: Milkyway, LEF/DEF, Milkyway XTR, *.AGF (Calibre).                                                       |
| star_cmd                    | ASCII file containing StarRC commands that control extraction functions.                                                                                 |
| TCAD GRD file (nxtgrd file) | File containing the modeled layers of a design, specified with the <code>TCAD_GRD_FILE</code> command.                                                   |
| Mapping file                | File containing physical layer mapping information between the input database and the nxtgrd file, specified with the <code>MAPPING_FILE</code> command. |

All StarRC command options are global throughout the GUI. Even though not all options are visible on all Wizard forms, they still contain the values shown in the main window. Command settings that do not apply to the currently selected flow are ignored.

Upon startup, GUI entry fields display their defaults. Fields labeled with red lettering are required.

---

## Using the GUI to Run a Timing or Noise Analysis

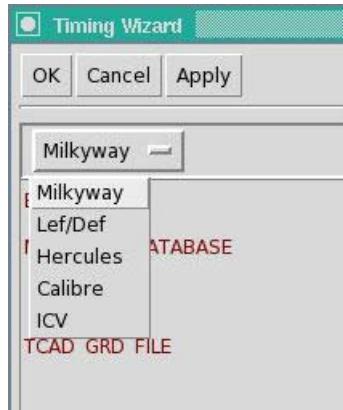
To start a StarRC timing or noise analysis, use the following procedure:

1. In the working directory, enter `StarXtract -gui` at the command line.
2. Select the type of analysis to run.
  - For timing analysis, choose Setup > Timing.
  - For noise analysis, choose Setup > Noise.

The Timing Wizard or Noise Wizard appears.

3. Select the input database type, as shown in [Figure 7-1](#).

*Figure 7-1 Database Type Selection*



4. Enter the command information as needed for the analysis.

The commands relevant for timing or noise analysis are displayed in the wizard dialog box. Different commands appear depending on the specified database type. Required commands are labeled in red.

5. After updating the command information, click OK.

6. Specify the layer mapping file.

Choose **Setup > Layer Mapping**.

Each analysis must contain a layer mapping file in the working directory. If you do not have a layer mapping file, see [Editing a Mapping File in the StarRC GUI](#).

7. Format the StarRC output file for a timing or noise run.

- For a timing analysis, choose **Timing > Netlist**. The **Timing Netlist** dialog box appears.
- For a noise analysis, choose **Noise > Report**. The **Noise Report** dialog box appears.

8. To save the settings, choose **File > Save**.

The **Save Tech File** window appears. In the **File name** field, enter a file name in which to save the command file for a future analysis.

9. To load a previously saved command file, choose **File > Load**.

Enter the file name in the **File name** field.

10. To run the analysis, choose File > Run. The Run dialog box appears, as shown in [Figure 7-2](#).

By default, the tool starts with the last unfinished module task. The Translate, xTractor, and Netlist stages are consecutive. When restarting an analysis, select the module that follows the last completed module, as shown in the log file.

*Figure 7-2 Run Form Dialog Box*



11. Click OK to run the analysis. Other options available in the Run Form Dialog Box are shown in [Table 7-2](#).

*Table 7-2 Options Available in the Run Form Dialog Box*

| Selection | Description                                                            |
|-----------|------------------------------------------------------------------------|
| OK        | Executes StarRC analysis and closes the Run Form dialog box.           |
| Cancel    | Cancels and closes the Run Form dialog box.                            |
| Apply     | Starts the analysis while leaving the Run Form dialog box open.        |
| Clean All | Removes previously specified file settings and starts StarRC analysis. |
| Translate | Starts the analysis in the Translate module.                           |
| xTractor  | Starts the analysis in the xTractor module.                            |
| Netlist   | Starts the analysis in the Netlist module.                             |

When the StarRC analysis is complete, “done” is written into the run log. You can open the view to view its complete contents.

## Using the GUI to Run a SingleShot Analysis

To run a SingleShot analysis, including both timing and noise, perform the following steps:

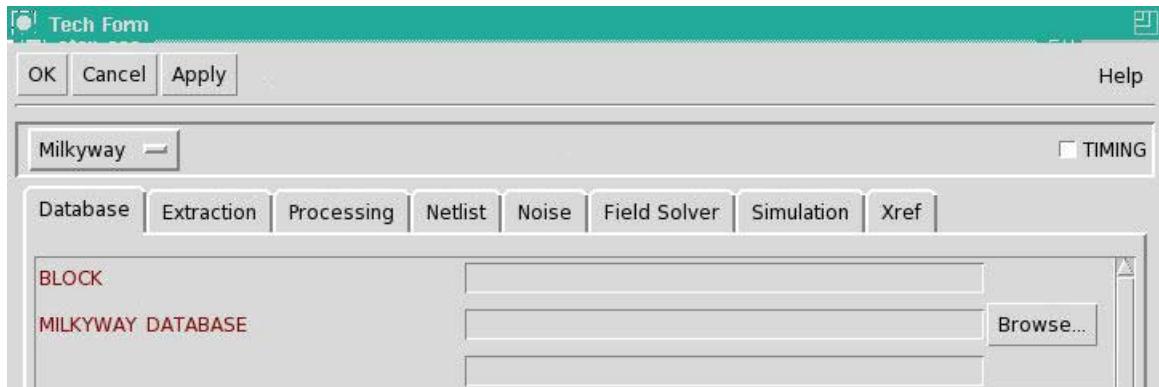
1. In the working directory, enter `StarXtract -gui` at the command line.
2. Choose Setup > SingleShot.

The Tech Form appears, as shown in [Figure 7-3](#).

3. Select the input database type from the Tech Form.
4. Specify the appropriate commands in each tab.

Each tab on the Tech Form contains commands appropriate for that operation. The Single Shot tabs are shown in [Figure 7-3](#).

*Figure 7-3 Tech Form Tabs*



Mandatory commands for each tab are labeled in red. The mandatory commands are different depending on the selected database type.

5. To save the command settings without running an analysis, choose File > Save. Provide a file name for saving the command file.
6. To start the SingleShot analysis, choose File > Run.

The Run Form appears.

By default, the tool starts with the last unfinished module task. The Translate, xTractor, and Netlist stages are consecutive. When restarting an analysis, select the module that follows the last completed module shown in the log file.

7. When the StarRC analysis is complete, the run log prints the word "done."

---

## StarRC Graphical User Interface Reference

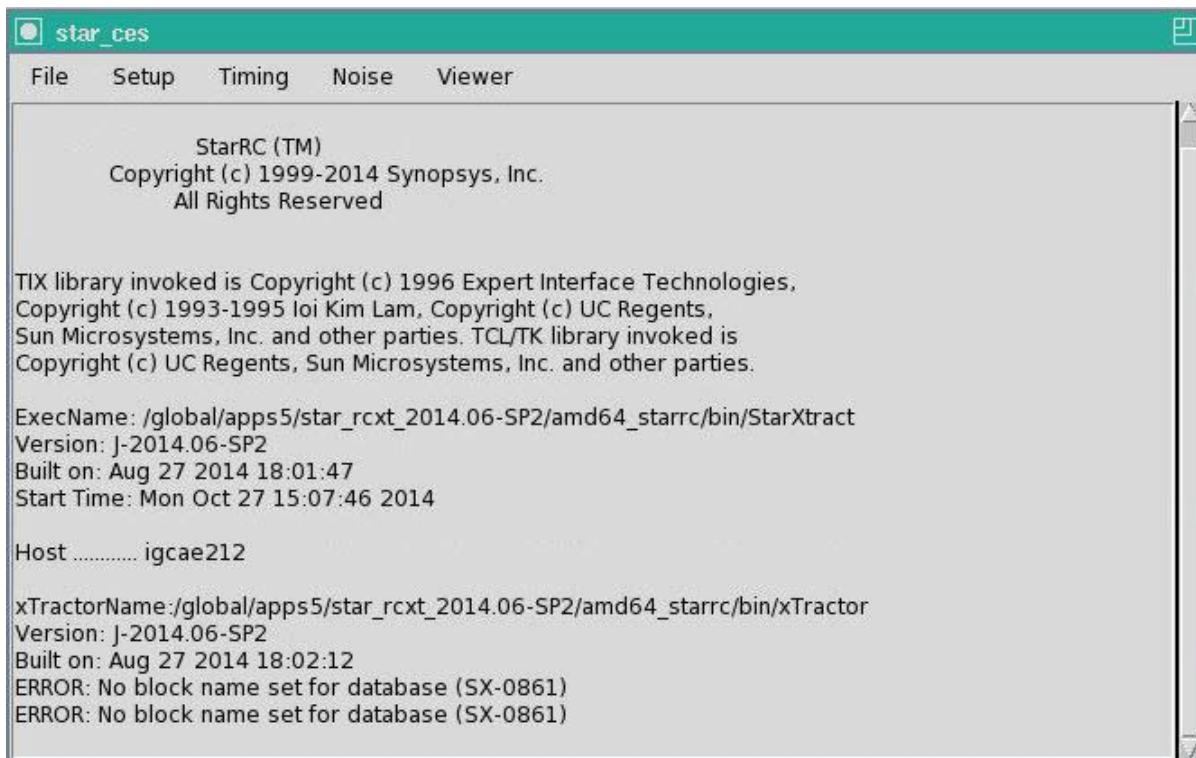
This section shows examples of the menus and dialog boxes available in the graphical user interface. The screens might be different for different StarRC versions.

---

### Control Window

The control window opens when you invoke the GUI. It contains the menus you use to configure and execute the StarRC analysis. The control window is shown in [Figure 7-4](#).

*Figure 7-4 GUI Control Window*



## File Menu

The control window contains a drop-down File menu, as shown in [Figure 7-5](#). The File menu commands enable you to load the command file and start the StarRC analysis. The options are described in [Table 7-3](#).

*Figure 7-5 File Menu*



*Table 7-3 File Menu Options*

| File menu command | Description                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Run               | Run StarRC analysis using the loaded file.                                                                                                                                                                                        |
| Cancel            | Cancel the analysis that is currently running. Intermediate files remain in the star directory. The next run can be started after you have used the intermediate files or specify a clean option to begin with unprocessed files. |
| Load              | Open the Load Tech File window.                                                                                                                                                                                                   |
| Clear             | Open the Clear Technology Options window. This resets every field to its default. It is helpful to use this command when you are building a command file with user-specific commands.                                             |
| Save              | Open the Save Technology File window. Specify the technology files to be saved in the file name field. All the commands you have entered are contained and stored in the file name. The file is saved in the working directory.   |
| Quit              | Exit the GUI. Any process currently running in the GUI is canceled                                                                                                                                                                |

## Setup Menu

The Setup menu pulldown list, shown in [Figure 7-6](#), is in the Control Window. Three types of StarRC runs are available (Timing, Noise, and SingleShot), as well as a selection for editing mapping files. Each selection opens a dialog box containing the relevant commands.

[Table 7-4](#) describes the options available in the Setup menu.

*Figure 7-6 Setup Menu*



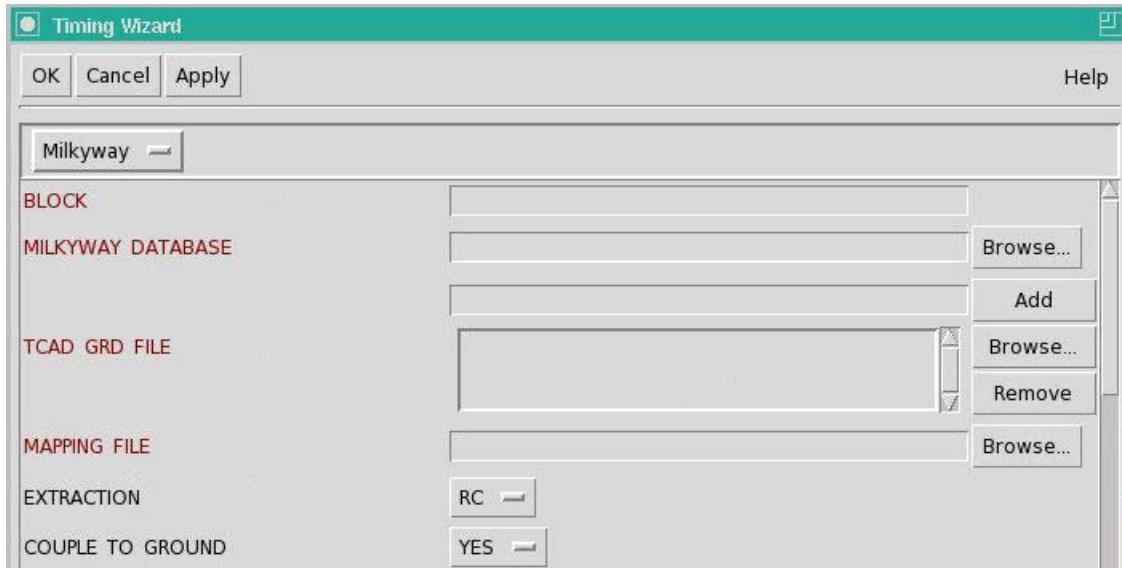
*Table 7-4 Setup Menu Options*

| Setup menu selection | Description                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Timing               | Open the Timing Wizard dialog box to specify options relevant to creating a netlist for timing analysis.                                                                                                                                                                                                                                                                                                    |
| Noise                | Open the Noise Wizard dialog box to specify options relevant to creating a netlist for a crosstalk or noise analysis.                                                                                                                                                                                                                                                                                       |
| SingleShot           | Open the Tech Form dialog box to specify options relevant to creation of a netlist for combined timing and noise analyses.<br><br>This option provides access to a larger set of StarRC commands than either the Timing or Noise analysis options. Multiple tabs let you set commands and options specific to Database, Extraction, Processing, Netlist, Noise, Field Solver, Simulation, and Xref choices. |
| Layer Mapping        | Open the specified mapping file so that you can alter its contents. A valid mapping file must have been specified in Timing, Noise, or Single Shot dialog box before it can be opened with this interface. Alternatively, you can create a mapping file by specifying a valid nxtgrd file in one of the analysis forms.                                                                                     |

## Setup > Timing

When you choose Setup > Timing, the Timing Wizard dialog box appears, as shown in [Figure 7-7](#). Use this dialog box to specify commands for generating timing analysis netlists.

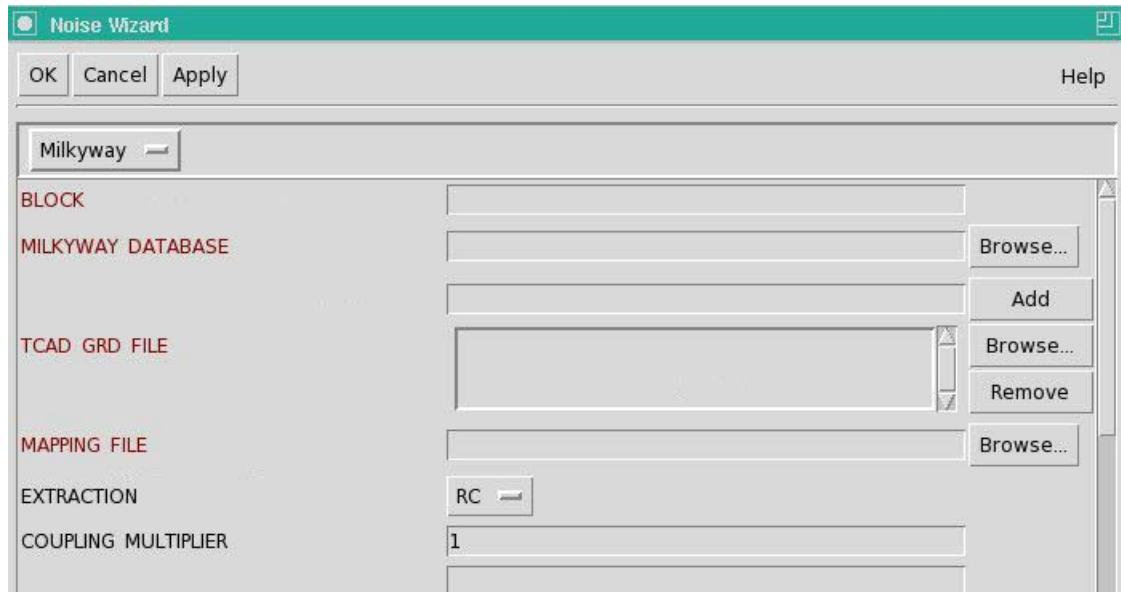
*Figure 7-7 Setup > Timing Wizard (Top Portion)*



## Setup > Noise

When you choose Setup > Noise, the Noise Wizard dialog box appears, as shown in [Figure 7-8](#). Use this dialog box to specify commands for generating noise analysis netlists.

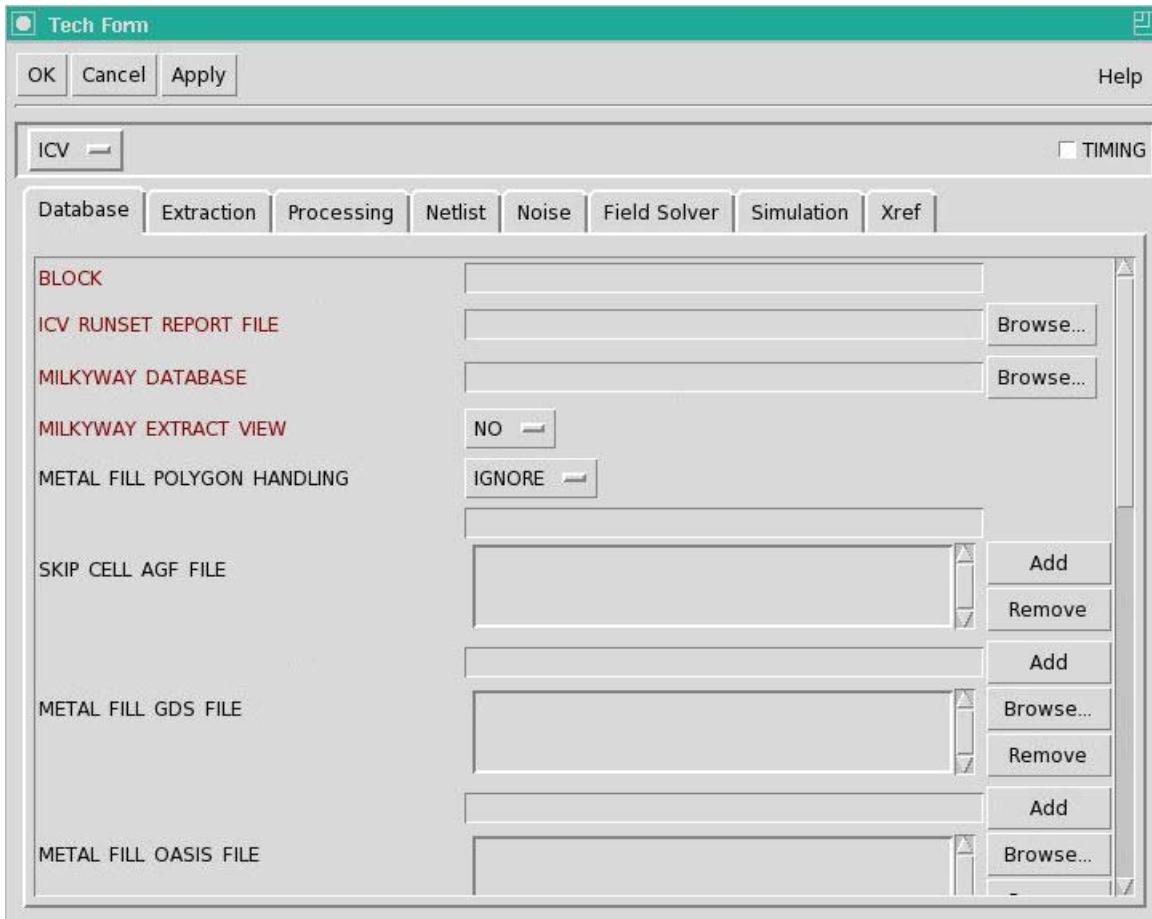
*Figure 7-8 Setup > Noise Wizard (Top Portion)*



## Setup > Single Shot

When you choose Setup > Single Shot, the Tech Form dialog box appears, as shown in [Figure 7-9](#). Use this dialog box to generate a netlist for both timing and noise analyses.

*Figure 7-9 Setup > Single Shot Tech Form*



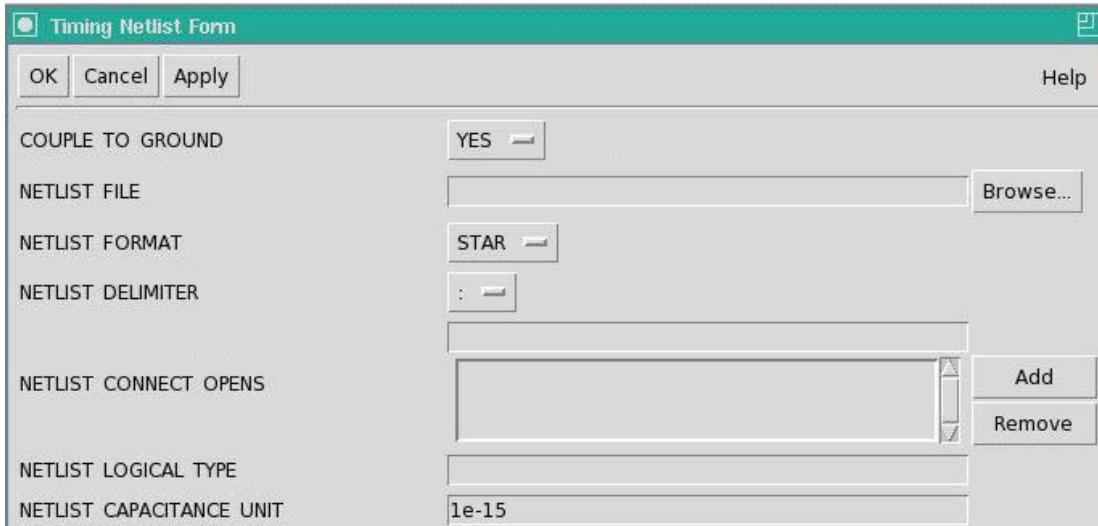
## Setup > Layer Mapping

When you choose Setup > Layer Mapping and a mapping file already exists, the Layer Mapping dialog box appears. To edit a mapping file, see [Editing a Mapping File in the StarRC GUI](#).

## Timing Menu

When you choose Timing > Netlist, the Timing Netlist dialog box appears, as shown in [Figure 7-10](#). This dialog box lists commands that affect the output netlist.

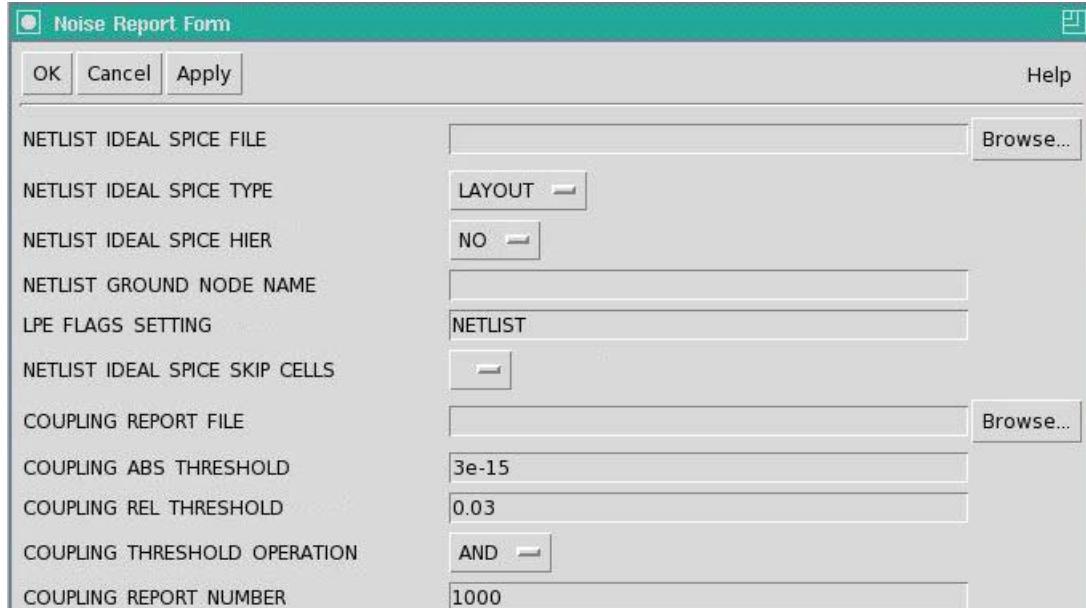
*Figure 7-10 Timing Netlist Form (Top Portion)*



## Noise Menu

When you choose Noise > Report, the Noise Report dialog box appears, as shown in [Figure 7-11](#). This dialog box lists commands that affect the output netlist.

*Figure 7-11 Noise Report Form*



## Viewer Menu

The Viewer menu, as shown in [Figure 7-12](#), features commands that let you access the Milkyway Viewer. [Table 7-5](#) describes the options.

*Figure 7-12 Viewer Menu*



*Table 7-5 Commands for Accessing the Milkyway Viewer With StarRC*

| Menu command        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prepare Viewer Data | Prepare the graphical layout data in the star directory from the details of the loaded command file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Launch Viewer       | Run an instance of Milkyway and open the specified block of layout design (read-only) from the loaded command file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Close Viewer        | Close the Milkyway viewer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Query Net           | Invoke a dialog box for querying nets. The filter provided accepts the wildcard asterisk (*) and question mark (?).<br><br>For each net, the details are shown in the right pane. The top edit box shows the net detail after the node information has been filtered out. The net's node names are listed in the bottom-left list box in the right pane.<br><br>The bottom-right list box provides the selected node detail. To show the node detail, double-click a node name or select a node name and click the Show node details button. The viewer zooms to the bounding box of the node. (You cannot do this from a Milkyway window. It must be done from the viewer.) |
| Query Device        | Invoke a dialog box for querying devices. The filter accepts the asterisk (*) and question mark (?) wildcards.<br><br>For each device, the details are shown in the right pane. The top edit box shows the device detail after the node information has been filtered out. Device details include port instances to which this device is connected.                                                                                                                                                                                                                                                                                                                          |
| Probe Text          | Open the Probe Text Window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

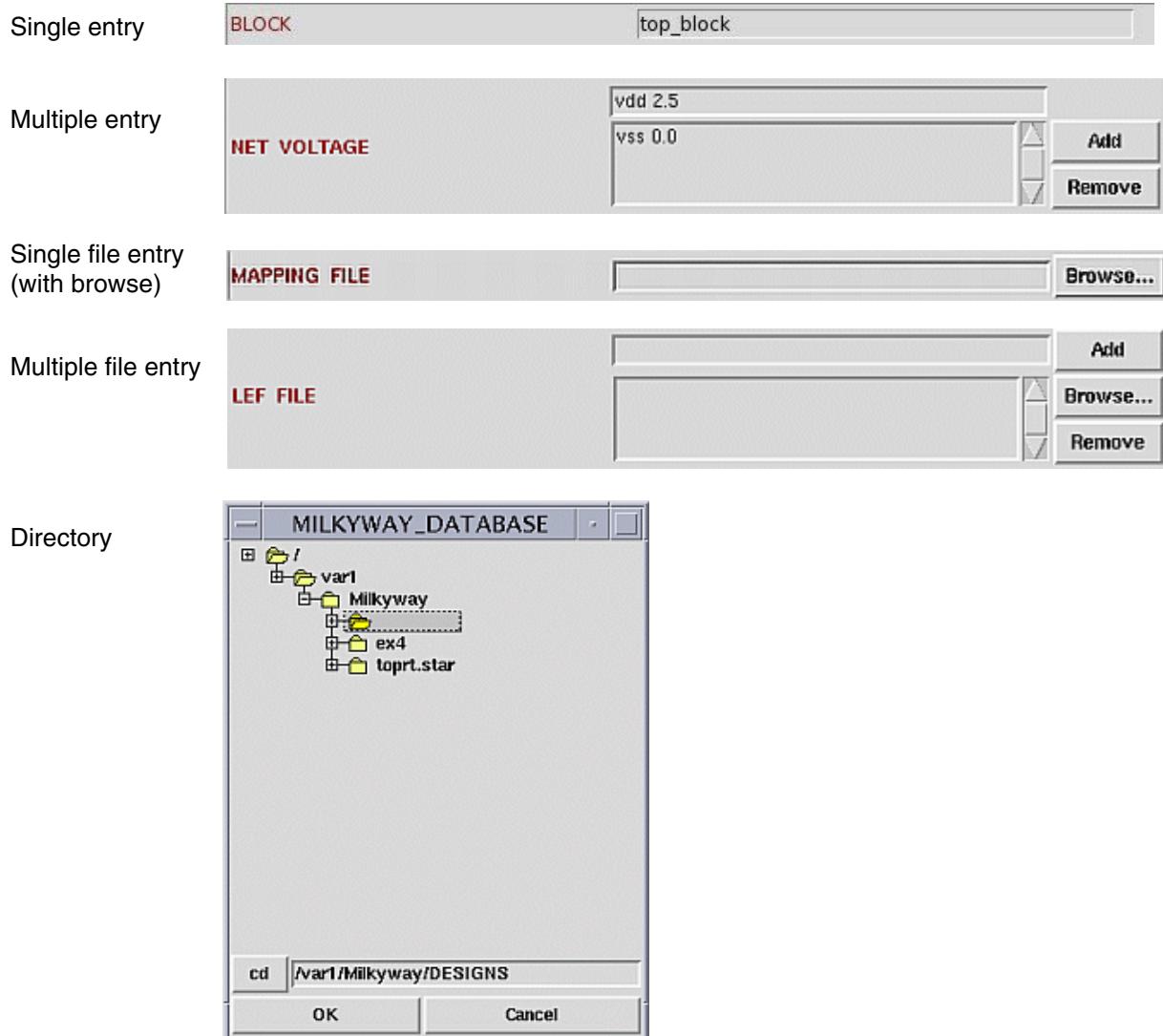
## Entering Information in the StarRC GUI

This section covers the entries and actions that the StarRC GUI enables. For information about specific commands in the forms, see [Chapter 15, “StarRC Commands.”](#)

### Entry Fields

You can set commands in the StarRC GUI by filling in the entry fields. The StarRC GUI has several types of entries, as shown in [Figure 7-13.](#)

*Figure 7-13 Available Entry Field Types*



**Table 7-6** explains the use of each type of field.

*Table 7-6 Field Types*

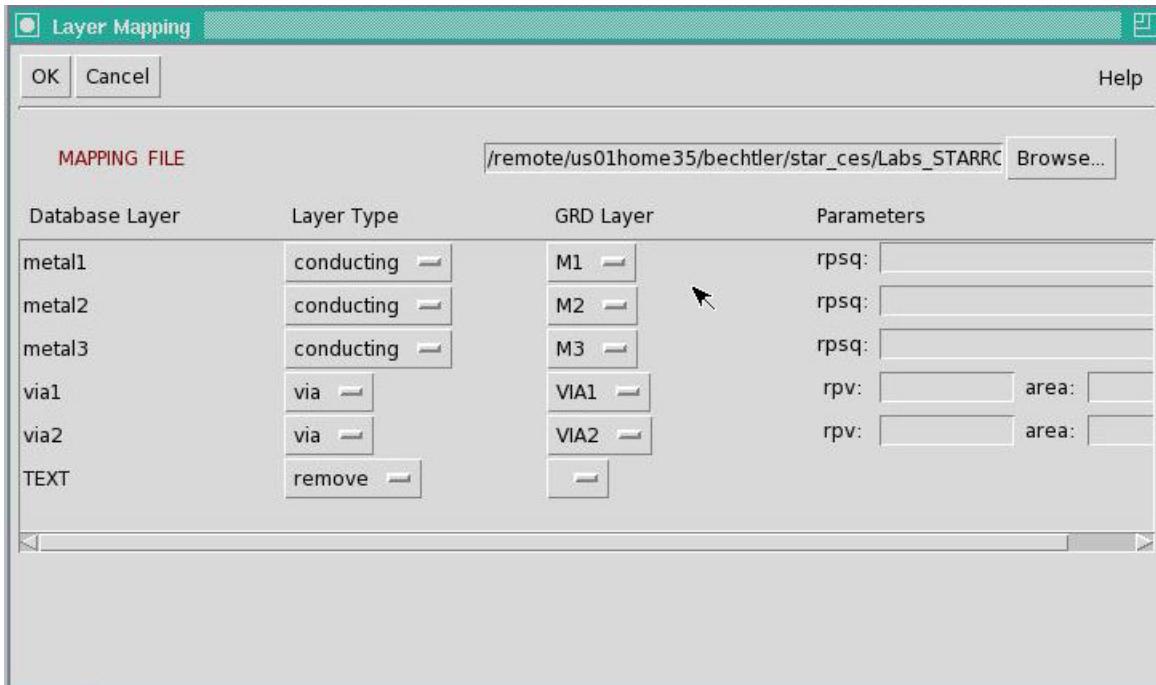
| Entry type       | Description                                                                                                                                                                                                            | Additional information                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Single entry     | Command options that are specified only one time. You can enter numbers, white-space delimited lists, and single words.                                                                                                | Enter text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Multiple entries | Command options that might be specified more than one time. You can enter multiple line commands.                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Single file      | Command options that can be specified only one time and require a single file path, for input or output files. An example is a mapping file.                                                                           | Clicking the Browse button opens a hierarchical file browser.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Multiple files   | Command options that can be specified more than one time and require a file path, for input files only.                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Directory        | Command options that require a directory path. The entry field for a directory can be filled manually, or you can click the Browse button to the right of the entry field to display a hierarchical directory browser. | <p>Clicking Browse opens a graphical file browser window. Select a file, then press return or click OK or Add in the file browser window to enter the file name in the list.</p> <p>To delete an entry from the list, select one of the files in the list and click Remove.</p> <p>Double-clicking a directory name collapses or expands the directory tree. Only directories are shown. Selecting a directory and then clicking OK places the directory name in the command entry field.</p> |

## Editing a Mapping File in the StarRC GUI

Choose Setup > Layer Mapping to edit mapping files in the StarRC Layer Mapping form.

The information in this form comes from the physical layout database and the nxtgrd file specified in the Setup windows. The mapping information appears as shown in [Figure 7-14](#). The form displays each database layer in a vertical list on the left side. The layer entries are organized by row; information in the fields to the right of a database layer entry applies to that layer.

*Figure 7-14 Layer Mapping Form*



Each database layer entry is associated with two pull-down menus. The menu in the Layer Type column contains the list of layer types, while the menu in the GRD Layer column contains the list of available layers in the nxtgrd file.

Both entries associated with a database layer must display a value before the layer mapping can be applied, unless the entry in the first column is removed.

The rightmost entry fields associated with a database layer allow you to override the resistance values specified in the nxtgrd file. Use this feature with care.

The contents of the Layer Mapping dialog box must be saved to a file before the extraction. Click OK to export the information to the file name entered in the MAPPING FILE field at the top of the dialog box.



# 8

## Using StarRC With the Virtuoso Custom Design Platform

---

This chapter describes the integration of StarRC with the Cadence Virtuoso custom design platform.

- [Introduction to Virtuoso Integration](#)
- [Setting Up Virtuoso Integration](#)
- [Virtuoso Integration Flow Configuration and Related Files](#)
- [View Generation](#)
- [StarRC Parasitic Generation Cockpit GUI](#)
- [StarRC OA View Creation](#)
- [Parasitic Probing](#)
- [Opens Debugger](#)
- [Virtuoso Integration SKILL Procedures and Related Variables](#)

---

## Introduction to Virtuoso Integration

The Virtuoso Integration (VI) interface allows you to use the StarRC tool to extract and investigate parasitics within the Virtuoso Analog Design Environment. The primary features are as follows:

- Parasitic View Generation

Set up and execute a StarRC extraction run within the Virtuoso environment. Read schematic and layout views, then generate parasitic views.

Parasitic view creation entails the generation of a Cadence DFII CDBA database (or OpenAccess) parasitic view that instantiates all ideal and parasitic devices extracted by the IC Validator, Hercules, or Calibre flows. This view is compatible with common netlist generation interfaces.

- Parasitic Prober

Examine the relationships between design elements and extracted parasitics.

Built-in highlighting capabilities enable you to highlight parasitic view nodes and polygons for previously probed resistances or capacitances. The parasitic prober also provides the ability to output probed parasitics to an ASCII report file and to annotate parasitic view total capacitance values to an associated schematic view.

- Opens Debugger

Investigate opens in the design.

The StarRC Cockpit is a graphical user interface that helps you to set up and execute most of the available operations. You can also use a settings file to store and load setup information.

---

## Setting Up Virtuoso Integration

The following sections describe settings and files for correct setup of the Virtuoso Integration environment:

- [Installation and Licensing](#)
  - [Environment Variables](#)
- 

### Installation and Licensing

The StarRC Virtuoso Integration functionality requires two StarRC license keys:

- STAR-RC2-NETLIST is checked out during parasitic view generation in cdba and is not necessary for the OpenAccess flow.
- STAR-RC2-PROBER is checked out during invocation of the parasitic prober.

Two components are necessary to run StarRC Virtuoso Integration:

- The `rcskill.cxt` Virtuoso binary context file
- The StarRC base executable package

Use the following installation procedure:

1. Ensure that StarRC (and the Hercules or IC Validator tools, for those flows) is contained in the operating system execution path before starting the Virtuoso tools.
2. Load the `rcskill` context file in the Command Interpreter Window (CIW) during Virtuoso tool invocation.  

```
loadContext("rcskill.cxt")
```
3. Initialize the context file `rcskill` in the Virtuoso Command Interpreter Window.  

```
callInitProc("rcskill")
```

The statements in steps 2 and 3 can be inserted into the `.cdsinit` file to be run automatically during Virtuoso startup.

Any relative path that you specify in the Cockpit must be relative to the run directory—the directory in which you invoke Virtuoso. Any relative path that you specify in the RCGenParaViewBatch SKILL procedures must be relative to the StarRC run directory.

---

## Environment Variables

You can specify global operating system conditions before starting a StarRC run by setting environment variables. To set an environment variable, use the `setenv` command at the operating system prompt with the variable name and the setting to be applied. For example:

```
% setenv RC_VI_SETTINGS_FILE starfile.txt
```

The following environment variables affect the operation of the StarRC interface with the Virtuoso application:

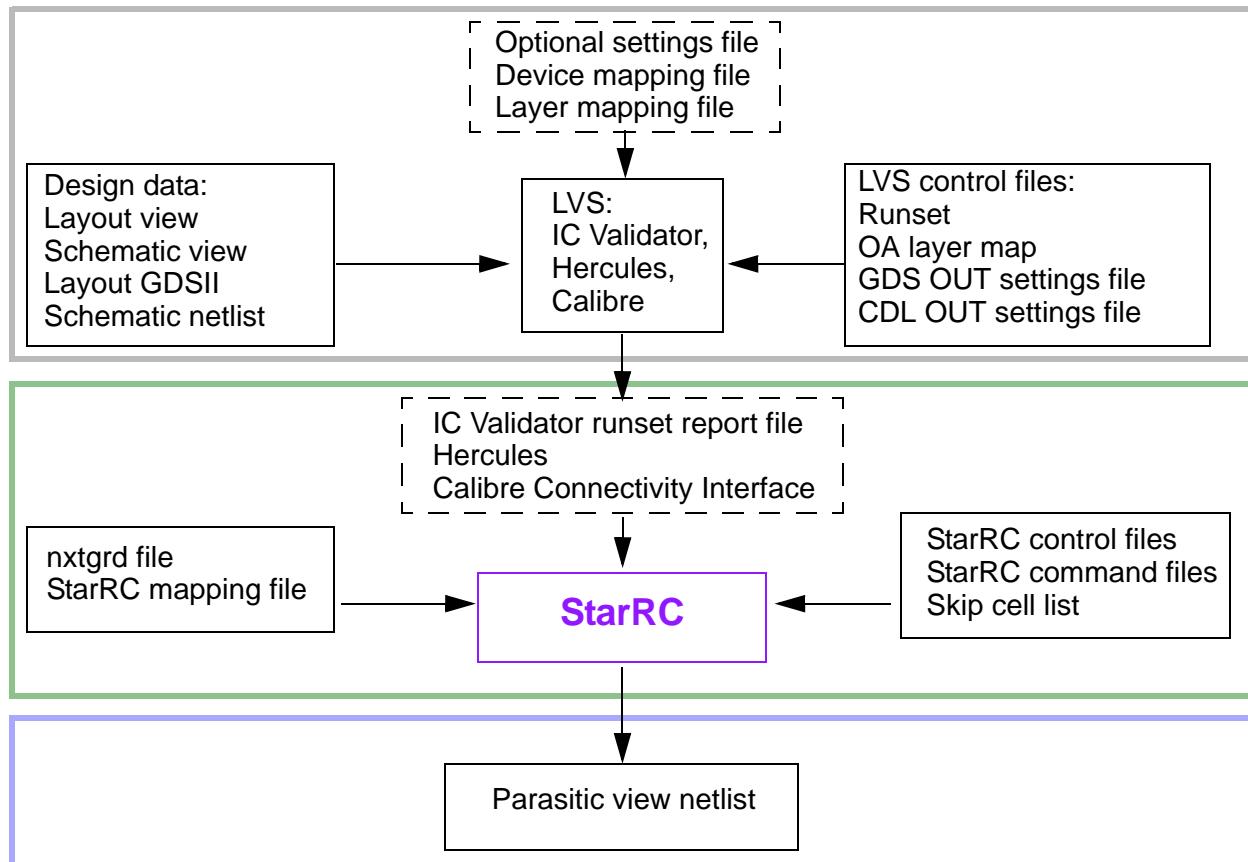
- `RC_VI_SETTINGS_FILE file_name`  
Specifies a Cockpit configuration file to prepopulate entry fields in the Cockpit user interface. For more information, see [Populating the Cockpit Fields Automatically](#).
- `ADVANCED_SAVE_LOAD YES | NO`  
Specifies to modify the file load and save button configuration in the Cockpit, as shown in [Advanced Save and Load Mode](#).
- `RC_SAVE_ALL YES | NO`  
Specifies whether to save the Output Lib and Output Cell names to the `.snps_settings` file. For more information, see [Output Parasitics Tab](#).
- `RC_ADD_MENU YES | NO`  
Specifies whether to display the StarRC pulldown menu in the Virtuoso layout and schematic windows.

## Virtuoso Integration Flow Configuration and Related Files

The Virtuoso Integration flow has three stages, as shown in [Figure 8-1](#):

- Layout versus schematic (LVS) stage, using the IC Validator, Hercules, or Calibre tools
- Extraction stage, in which you can adjust StarRC commands without a command file
- Output stage, in which you can select the output format and other options

*Figure 8-1 Parasitic Extraction Flow in the Virtuoso Flow*



The device mapping file, layer mapping file, and skip cell list are special files for Virtuoso Integration. All other files are standard input files for the LVS tools and the StarRC tool.

The Cockpit configuration file is a text file that contains entries for most of the settings that you can enter manually through the Cockpit graphical user interface. This file is also known as a settings file because it must have an extension of .snps\_settings. Loading a settings file is a convenient way to prepopulate Cockpit entry fields. Saving a settings file after a session enables you to re-create the session at a later time.

---

## The Device Mapping File

A device mapping file maps ideal and parasitic devices in the StarRC parasitic output to the corresponding device symbols in the Virtuoso symbol libraries. This file contains an entry for every ideal and parasitic device model that exists in the parasitic output. It also provides the ability to remap standard StarRC DSPF device property names to user-specified property names. The following syntax shows the entry format of a single line:

```
RC_model_name dfii_lib_name dfii_cell_name dfii_view_name
dfii_symbol_pin_1 dfii_symbol_pin_2 [dfii_symbol_pin_3] ...
[CALLBACK = procedureSymbol] [PROPMAP DSPFprop1 = ADEprop1...]
```

---

| Argument                    | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>RC_model_name</i>        | StarRC output model name; corresponds to the schematic device model name when XREF is activated. Note that keywords pres and pcap are used for parasitic resistors and capacitors.                                                                                                                                                                                                                                                                                                                                |
| <i>dfii_lib_name</i>        | Name of Virtuoso library containing the corresponding device symbol.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>dfii_cell_name</i>       | Cell name of Virtuoso device symbol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>dfii_view_name</i>       | View name of Virtuoso device symbol.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>dfii_symbol_pin_N</i>    | Name of the pin inside Virtuoso device symbol that corresponds to terminal #N in an analogous DSPF-based StarRC parasitic output. Note that the ordering of Virtuoso symbol pin names inside the device mapping file must match the StarRC netlist pin order for the device type of interest. The keyword <i>nil</i> can be specified for any <i>dfii_symbol_pin</i> to indicate that the terminal #N in the StarRC parasitic output should be ignored when connecting the corresponding Virtuoso library symbol. |
| <i>procedureSymbol</i>      | Symbol name of an optional user-defined callback procedure that is executed before instantiating a device of type <i>RC_model_name</i> .                                                                                                                                                                                                                                                                                                                                                                          |
| <i>DSPFpropN = ADEpropN</i> | Optional mapping of standard DSPF property name into a user-specified property name. Note that keyword PROPMAP is required before the first property name mapping entry. Setting <i>DSPFpropN = nil</i> prevents the listed property from being annotated to the device symbol.                                                                                                                                                                                                                                   |

---

Two slashes (//) serve as a comment delimiter in the device mapping file.

An example DFII symbol mapping file is illustrated as follows:

```
MNM devlib nmos4 ivpcell D G S B PROPMAP l=simL w=simW
MPM devlib pmos4 ivpcell D G S B PROPMAP l=simL w=simW
pres Lib presistor auLvs PLUS MINUS CALLBACK=insertPresProp
pcap Lib pcapacitor auLvs PLUS MINUS CALLBACK=insertPcapProp
pind analogLib pinductor symbol PLUS MINUS
pmind analogLib pmind symbol
```

Parasitic elements pres, pcap, pind, pmind should use the lib/cell/view from analogLib. For example,

```
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
pind analogLib pinductor symbol PLUS MINUS
pmind analogLib pmind symbol
```

However, if a parasitic element name conflicts with a user-defined device name, Virtuoso Integration provides the following parasitic element names:

- pres[starrc]
- pcap[starrc]
- pind[starrc]
- pmind[starrc]

When pres and pres[starrc] both appear in the device mapping file, pres[starrc] overrides pres as the parasitic element name.

For information about the CALLBACK keyword, see [Instance Creation Callback](#).

## The Layer Mapping File

The layer mapping file maps StarRC runset layers in the StarRC mapping file to the corresponding Virtuoso technology file layers. This allows polygons, ports, and subnodes from the parasitic extraction to be stored within the StarRC generated DFII parasitic view.

Each layer that you want to store in the parasitic view should be specified in the layer mapping file and should be mapped to an existing layer-purpose pair (LPP) from the Virtuoso technology library for the library being used. The mapping file optionally lets you specify DFII layer-purpose pairs for subnode markers generated by StarRC to represent parasitic top-level ports (\*|P), instance ports (\*|I), and subnodes (\*|S).

The format of the layer mapping file is as follows:

```
RC_MAPPING_FILE_layer
 dfii_polygon_layer_name dfii_polygon_purpose_name
 [dfii_port_layer_name dfii_port_purpose_name
 [dfii_subnode_layer_name dfii_subnode_purpose_name]]
```

```
RC_MAPPING_FILE_layer
 dfii_polygon_layer_name dfii_polygon_purpose_name
 [dfii_port_layer_name dfii_port_purpose_name
 [dfii_subnode_layer_name dfii_subnode_purpose_name]]
```

```
. . .
```

---

| Argument                                                           | Definition                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>RC_MAPPING_FILE_layer</i>                                       | Database layer name from the CONDUCTING_LAYERS, VIA_LAYERS, or MARKER_LAYERS sections of the mapping file specified by the MAPPING_FILE command.                                                                                                                                                                                                                                                                                                                                              |
| <i>dfii_polygon_layer_name</i><br><i>dfii_polygon_purpose_name</i> | Layer name and purpose name from the DFII technology library, which forms the layer-purpose pair to which <i>RC_MAPPING_FILE_layer</i> polygons should be written. If either entry is not specified or are specified as nil, polygons corresponding to the <i>RC_MAPPING_FILE_layer</i> is not generated within the parasitic view.                                                                                                                                                           |
| <i>dfii_port_layer_name</i><br><i>dfii_port_purpose_name</i>       | Layer name and purpose name from the DFII technology library, which forms the layer-purpose pair to which parasitic port markers corresponding to <i>RC_MAPPING_FILE_layer</i> interconnect should be written. Parasitic port markers are analogous to * P nodes and .SUBCKT header nodes that would appear in the DSPF output. If either entry is not specified or are specified as nil, ports corresponding to the <i>RC_MAPPING_FILE_layer</i> is not generated within the parasitic view. |
| <i>dfii_subnode_layer_name</i><br><i>dfii_subnode_purpose_name</i> | Layer name and purpose name from the DFII technology library, which forms the layer-purpose pair to which parasitic subnode markers corresponding to <i>RC_MAPPING_FILE_layer</i> should be written. Parasitic subnode markers are analogous to * I and * S nodes that would appear in a DSPF output. If not specified or if specified as nil, subnodes corresponding to the <i>RC_MAPPING_FILE_layer</i> is not generated within the parasitic view.                                         |

---

Use two slashes (//) to indicate comments in the layer mapping file.

Polygons are written to the parasitic view only if the Hercules, IC Validator, or Calibre database layer of the polygon is mapped to a valid Virtuoso layer-purpose pair in the DFII layer mapping file. If a Hercules, IC Validator or Calibre database layer is not listed in the mapping file, no polygons corresponding to that layer are stored in the parasitic view. If the file is not supplied at all, no graphical data is written.

Because all ports and subnodes correspond to specific database layers in standard StarRC outputs, separate layer-purpose pairs are used in the DFII layer mapping file for the generation of port and subnode markers relative to the generation of interconnect polygons. Because port and subnode markers enable point-to-point resistance probing with the StarRC parasitic probing utility, failure to include layer-purpose pairs for port and subnode markers prohibits the probing of point-to-point resistance between nodes lacking such markers.

No layer-purpose pair can be used both as a polygon LPP and a port and subnode LPP. If an LPP used as a polygon LPP is found in the mapping file, then that same LPP is disregarded if subsequently listed in the mapping file as a port or subnode LPP. Likewise, if an LPP used as a port or subnode LPP is found in the mapping file, then that same LPP is disregarded if subsequently listed in the mapping file as a polygon LPP.

The following example shows a DFII layer mapping file:

```
M1 metall1 net metall1 port metall1 node
fpoly poly net poly port poly node
ngate poly net poly port poly node
pgate poly net poly port poly node
diff_con cc net nil nil cc node
subtie pad drawing nil nil pad node
welltie pad drawing nil nil pad node
nsd nactive net nactive port nactive node
psd pactive net nactive port pactive node
m1_pio_marker nil nil metall1 port nil nil
m2_pio_marker nil nil metall2 port nil nil
m3_pio_marker nil nil metall3 port nil nil
poly_marker nil nil poly port nil nil
```

---

## Customizing the LVS Run

The Parasitic Generation Cockpit helps you to execute the layout versus schematic operation for the IC Validator, Hercules, or Calibre LVS tools. Choose the LVS tool from the Flow menu.

You must provide a runset for the selected LVS tool. For the Calibre Connectivity Interface flow, you must also supply a query command file. You can customize many items in the Device Extraction tab, including the following items:

- IC Validator (ICV): Regardless of the setting in the runset, the ICV runset report file name defaults to pex\_runset\_report\_file unless you provide a new name in the Cockpit field.
- Hercules: The location of the extract view is determined from the runset.  
If multiple WRITE\_EXTRACT\_VIEW commands exist in the runset, the last one is used regardless of the switch setting inside the runset.
- Calibre: The location of the svdb directory is determined from the runset.  
If multiple MASK SVDB DIRECTORY commands exist in the Calibre runset, the first one is used regardless of the switch setting inside the runset.

---

## Customizing the StarRC Extraction Run

You must provide a standard nxtgrd mapping file for Virtuoso Integration for parasitic extraction. To customize your StarRC job, you can

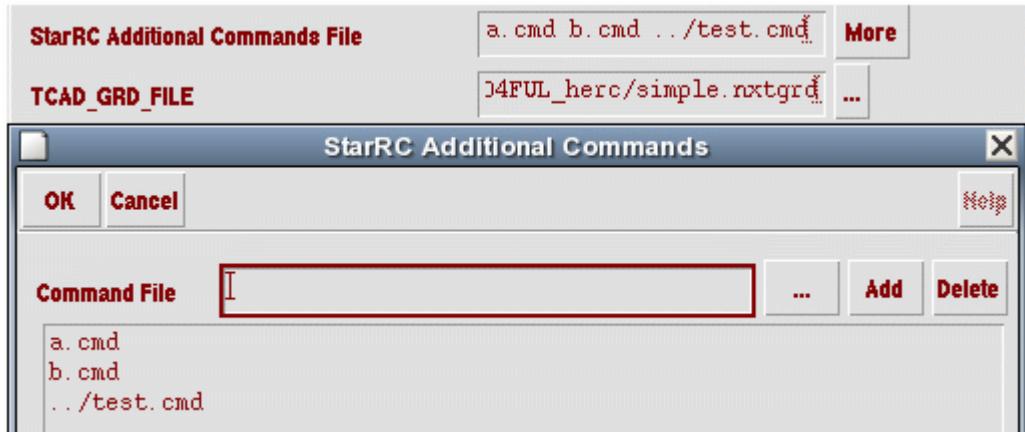
- Add a user-defined StarRC command file through the StarRC Additional Commands dialog box in the Extract Parasitics tab, as shown in [Figure 8-2](#).
- Use the Virtuoso Integration dialog box settings

The priority of commands is as follows, from highest to lowest:

1. Required native Virtuoso Integration commands
2. Additional StarRC command file commands
3. Additional settings in Virtuoso interface dialog boxes

Use the `view_cmd` or `oa_cmd` commands in the Virtuoso Integration run directory to list the StarRC commands that Virtuoso Integration has inserted into your job.

Figure 8-2 Adding or Deleting Additional StarRC Command Files



---

## View Generation

View generation is described in the following sections:

- [Net and Instance Name Conventions](#)
  - [Port and Terminal Connectivity Characteristics](#)
  - [Instance Property Annotation from the Schematic View](#)
  - [Subnode Marker and Parasitic Device Visualization](#)
  - [User-Defined Callbacks](#)
  - [Callback Flow Example](#)
- 

## Net and Instance Name Conventions

The StarRC parasitic view abides by naming conventions for nets and instances to enforce uniformity with DFII naming rules and naming conventions for schematic-based parasitic simulation analysis. These naming conventions are unique to the Virtuoso Integration flow and are different from standard StarRC DSPF netlist conventions.

The pipe character (|) serves as the default hierarchical delimiter for the parasitic view.

During schematic view netlisting for LVS and StarRC schematic-based cross-referencing (see the StarRC XREF command), the schematic view netlist generator can append prefixes to instance names according to the conventions of the netlist generator. These prefixes, commonly called *SPICE cards*, propagate into the StarRC parasitic netlist when the XREF

command is used. However, these extra instance prefixes are not present in the original schematic view and might impede the ability to perform full parasitic-view to schematic-view matching during simulation analysis. Therefore, the Virtuoso Integration flow removes these instance name prefixes as follows:

- Strips the initial SPICE card from ideal (not parasitic) instance names because StarRC adds this card directly.
- For hierarchical instance names, including those preceding internal net names:
  - Decomposes the name according to the hierarchical delimiter.
  - Strips the initial X character if a decomposed instance name begins with X anywhere in the decomposed instance list.
  - If the last name in the decomposed instance (often a primitive) begins with two occurrences of the same character, strips one of them.

*Table 8-1 Examples of Removal of Instance Name Prefixes*

| <b>StarRC DSPF instance name</b> | <b>Parasitic view instance name</b> |
|----------------------------------|-------------------------------------|
| XI0/XI5/M1                       | > I0/I5/1                           |
| XI0/XI5/MM1                      | > I0/I5/1                           |
| I0/I5/M1                         | > I0/I5/M1                          |
| I0/I5/MM1                        | > I0/I5/M1                          |
| XI0/X15/net1                     | > I0/I5/net1                        |
| I0/I5/net1                       | > I0/I5/net1                        |

- If the last name in the decomposed instance (often a primitive) begins with a SPICE card character such as X or M, strips that character.

The naming convention for input data makes the following assumptions:

- The schematic netlist generator always appends an X card to nonprimitive cell instances (for example, instances in the middle of a flattened hierarchical name).
- No instance name inside the schematic view begins with X.
- No instance name inside the schematic view begins with two occurrences of the same letter, such as the schematic view instance MM0.

Occurrences of bus bits (name<#>) are renamed if the bus bit is embedded within the middle of a hierarchical instance name. In such cases, the embedded string <#> is replaced with the embedded string (#). An example where this behavior can occur is an iterated schematic instance name embedded in a hierarchical name, for example,  
[0|I1|I2<3>|net4 becomes I0|I1|I2(3)|net4.

---

## Port and Terminal Connectivity Characteristics

StarRC reads the following information from a pre-existing view of the extracted cell and populates the same information within the parasitic view:

- *netExpression parameters*

StarRC parses all terminals and signals in the ports global nets view (default is layout) and records any netExpression parameters. If a terminal and a signal have the same name and both have individual netExpressions, then only the netExpression of the terminal is recorded. The netExpressions are then propagated into the new parasitic view as follows:

- If a terminal in the parasitic view has a name matching a terminal or signal name for which a netExpression was read from the existing cell, then the netExpression is added to that terminal.
- Otherwise, if a signal in the parasitic view has a name matching a cached terminal or signal name for which a netExpression was read from the existing view, then the netExpression is added to that signal.

Note:

Terminals in the parasitic view are dictated by the presence of ports in the StarRC parasitic output which are analogous to \*|P nodes or .SUBCKT headers in a DSPLF netlist. If no such nodes exist, then StarRC does not create any terminals in the parasitic view and no netExpression parameters are transferred.

- *Direction parameters for terminals*

StarRC parses all terminals in the pre-existing view and records any direction parameters listed on any terminals. If a terminal is found in the parasitic view bearing the same name as a terminal with a direction parameter in the pre-existing view, then StarRC attaches the same direction parameter string to the matching terminal in the parasitic view.

- *isGlobal parameters for signals*

StarRC parses all signals in the pre-existing view and records any isGlobal parameters listed on any signals. If a signal is found in the parasitic view bearing the same name as a signal with an isGlobal parameter in the pre-existing view, then StarRC attaches the same isGlobal parameter string to the matching terminal in the parasitic view. Note that isGlobal parameters are propagated only for signals to which no terminals bearing

netExpression parameters are attached. The pre-existing view from which the previous connectivity and terminal information is read is specified by the corresponding field in the StarRC parasitic generation cockpit or by the corresponding argument to RCGenParaViewBatch.

When updating a terminal view, StarRC gathers terminal information from a given symbolic view and parses all signals in the parasitic view to check for floating nets on device instance terminals. If a net name matches one terminal name on the symbol view, StarRC creates the name as the terminal name for the parasitic view.

- *Power net name for the ports*

StarRC creates the ports of the parasitic view as the input database top-block port names.

Sometimes, you might want to integrate the parasitic view to another circuit, some additional power pins are necessary for the parasitic view to connect to upper views. If additional power port names are necessary for the parasitic view, you can use the option on the Cockpit: Reading Pin from symbol, to assign an additional (symbol) view for StarRC to extract the additional power net names from the ports of the given view. Then, create new ports with the name for the parasitic view.

## Instance Property Annotation from the Schematic View

There are properties on the instance inside the parasitic views. The properties can be grouped into two basic groups: property names and values from the schematic view, and property names and values from the LVS tool. There might also be custom usages, such as to copy a property value to multiple property names, delete or make nil for a property, change a property name, or create a new property. StarRC needs to set the property name and values in the parasitic view for the simulator to work correctly. StarRC has developed strategies and syntax for the instance property annotation flow.

## Controlling Instance Property Annotation

Use the `XREF:YES` command when you want StarRC to generate a parasitic view constructed by the layout view and the extracted netlist from the LVS tool, with the schematic net name, instance name, or instance property attached.

The following options control the default behaviors of instance property annotation:

- In the `.snps_settings` file  
`CARRY_SCH_PROPERTY : [ YES ] | NO`
- In the `RCGenParaViewBatch` procedure  
`carry_sch_property : [ t ]/nil`

You might encounter the following issues:

- Some properties only exist in the schematic view.

Because LVS tools do not take schematic views as input, they use the CDL netlist generated from the schematic view. Some properties might not be netlisted in the CDL netlist procedure. Also, LVS tools report only the properties that have been defined in the LVS runset to LVS results database. Therefore, StarRC must carry those properties and their values from the schematic to the parasitic view.

- Some property names are used in both the schematic view and LVS results.

In this case, the layout view property values override the schematic view property values.

For example, a schematic view contains the following two instances:

```
I1: p1=s1 p2=s1 p3=s1 p4=s1
I2: p1=s2 p2=s2 p3=s2 p4=s2
```

The LVS tool extracts the following information:

```
I1: p3=11 p4=11 p5=11
I2: p3=12 p4=12 p5=12
```

In the DFII library, the cells I1 and I2 are used in the cell m.

```
m: p1=0 p2=0 p3=0 p4=0
```

When CARRY\_SCH\_PROPERTY: YES is specified, the parasitic view contains the following:

```
I1: p1=s1 p2=s1 p3=11 p4=11 p5=11
I2: p1=s2 p2=s2 p3=12 p4=12 p5=12
```

When CARRY\_SCH\_PROPERTY: NO is specified, the parasitic view contains the following:

```
I1: p1=0 p2=0 p3=11 p4=11 p5=11
I2: p1=0 p2=0 p3=12 p4=12 p5=12
```

## Property Mapping

This section describes syntax that you can use to adjust the property annotation behaviors. These behaviors are applied to all properties, whether they are in the schematic view or LVS results.

| Syntax   | Description                                                                |
|----------|----------------------------------------------------------------------------|
| dspfProp | Property name or value in the StarRC DSPF result, usually from LVS results |
| schProp  | Property name or value in the schematic view                               |
| cdfProp  | CDF defined property name                                                  |
| preProp  | all property name or value before StarRC parasitic view generation         |
| paraProp | property name or value in the final parasitic view                         |

### Property Mapping Behavior

A=B

In the instance of the parasitic view, paraProp B gets its value from preProp A. Also, paraProp A value comes from preProp A. This is equivalent to A=A and A=B.

A=B and A=C

Similar to A=B, but with multiple usage.

A>B

paraProp B gets its value from preProp A.

Also, paraProp A is removed.

If A is a cdfProp and can't be removed, the value is set to nil such that A=nil and A=B.

A="*constant*"

Assigns a constant to paraProp A, no matter what is its original value. If there is no preProp name A, then StarRC creates a paraProp A with the assigned value.

For example, A="10"

A=nil

Remove paraProp A

If A is a cdfProp and cannot be removed, the value of A is assigned to be nil.

## PROPMAP Case Sensitivity

You can specify the case sensitivity

- In the Cockpit dialog box with the PropMap Case Sensitive option, as shown in [Figure 8-5](#)
- In the .snps\_settings file

```
PROPMAP_CASE_SENSITIVE : YES | NO
```

- In the RCGenParaViewBatch procedure

```
?propmap_case_sensitive t|[nil]
```

This option controls the matching behavior of PROPMAP.

For example,

```
PROPMAP ABC=abc
```

When PROPMAP\_CASE\_SENSITIVE : YES

If there are 2 preProp names as ABC and abc, only ABC is copied to abc.

```
preProp : ABC=10 abc=100
paraProp: ABC=10 abc=10
```

When PROPMAP\_CASE\_SENSITIVE : NO

```
preProp: Abc=10
```

StarRC still uses the Abc value as ABC to write to abc.

```
paraProp: Abc=10 abc=10
```

## Instance Name Matching Rule

StarRC performs instance property annotation by finding the corresponding instance in schematic view to do the annotation. Because the instance name might be changed by netlist processing or LVS tool renaming behaviors, it is not easy to find the original schematic view instance name to determine schProp for annotation. StarRC adopts such a heuristic way to find instance in schematic view to annotate properties. (See [Net and Instance Name Conventions](#)) StarRC also performs SPICE card stripping to schematic view instances according the convention rule, then find the corresponding instance in schematic view to have its properties for annotation.

Note:

When Virtuoso Integration can't find a certain schematic view instance to annotate properties to parasitic view instances, StarRC creates the schematic\_info\_log file to report failed instances.

---

## Subnode Marker and Parasitic Device Visualization

The sunset layer to DFII layer mapping file provides the ability to write extracted polygon data to the parasitic view. Besides the storage of interconnect polygons, graphical data including subnode markers, port markers, and pres or pcap flylines are also written to the parasitic view. For more information about the layer mapping file, see [The Layer Mapping File](#).

A subnode is defined as any \*|I or \*|S node that would normally occur in a standard StarRC DSPF parasitic netlist. A port is analogous to any \*|P node or any entry in the .SUBCKT header line of a standard DSPF parasitic netlist. These nodes represent the electrical connection points for parasitic devices and ideal devices. Every subnode has unique xy coordinates as well as an extracted database layer on which the subnode lies. Using this information, it is possible to represent the subnodes in the parasitic view with small marker shapes placed at their corresponding xy coordinates.

The DFII layer-purpose pair to which a port or subnode marker is written is defined in the DFII layer mapping file. Only ports or subnodes whose corresponding database layers are listed in the DFII layer mapping file has markers written to the parasitic view. The default size of all subnode markers is 0.1 m x 0.1 m. This default size can be changed by adding an entry to the Cockpit configuration file as follows:

```
SUBNODE_SIZE: subnode_side_length_in_microns
```

Likewise, graphical data is also stored for parasitic resistors and coupling capacitors in the form of flylines between subnodes. Flylines are not stored for grounded capacitors because such capacitors by definition do not terminate at a finite xy coordinate on an interconnect polygon. These flylines are annotated with two properties: the parasitic instance name and the parasitic value. All flylines for parasitic resistors are written to a single Virtuoso layer-purpose pair. Likewise, all flylines for parasitic capacitors are written to a separate Cadence layer-purpose pair. These layer-purpose pairs can be listed in the DFII layer mapping file as follows, using the special tags \*pres and \*pcap.

```
// <*pres or *pcap> dfii_polygon_lyr dfii_polygon_purpose
*pres poly net
*pcap metall net
```

If no \*pres and \*pcap lines are defined in the DFII layer mapping file, StarRC uses the following default mapping:

```
*pres y0 drawing
*pcap y1 drawing
```

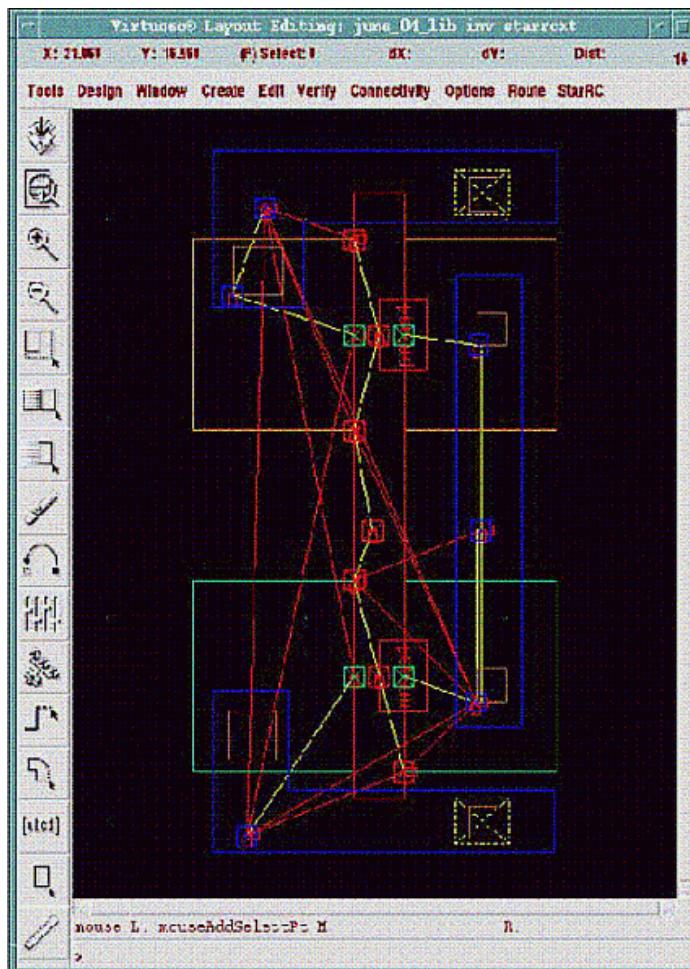
Note:

A flyline is stored in the parasitic view only if both of its nodes have markers stored in the parasitic view. Likewise, port and subnode markers are only stored for sunset layers that are mapped in the DFII layer mapping file. Therefore, the number of parasitic resistor

and capacitor flylines present in the parasitic view is a direct function of the runset layer to DFII layer mappings in the DFII layer mapping file.

An illustration of a StarRC parasitic view containing subnode markers and flylines is shown in [Figure 8-3](#).

*Figure 8-3 StarRC Parasitic View With Port and Subnode Markers and Pres or Pcap Flylines*



---

## User-Defined Callbacks

You can customize StarRC parasitic view generation through the use of user-defined callbacks. A callback is a method to invoke a procedure or command at a specific event in the flow. Four types of callbacks are available and are covered in the following sections:

- [Pre-Extraction Callback](#)
- [View Preprocessing Callback](#)
- [View Postprocessing Callback](#)
- [Instance Creation Callback](#)

Virtuoso tool versions 6.x and later uses an Open Access (OA) file writer to generate parasitic views. This flow supports the pre-extraction and postprocessing callbacks but does not support preprocessing or instance creation callbacks. For Virtuoso versions 6.x and later, you can disable the OA file writer and use the SKILL writer instead, which enables the use of all callback types. To disable the OA file writer, use one of these methods:

- Edit the `.snps_settings` file to include the following line:  
`OA_WRITER: NO`
- Edit the `RCGenParaViewBatch` procedure to include the following line:  
`oa_writer nil`

---

### Pre-Extraction Callback

The pre-extraction callback is a SKILL expression that is loaded and executed before the beginning of StarRC view generation. This callback interface enables you to perform customized operations on StarRC inputs before beginning the extraction. If the LVS and StarRC steps are used consecutively, pre-extraction callback is executed after LVS finishes and before StarRC starts. You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

`PRE_EXTRACTION_CALLBACK: cmd_string`

The *cmd\_string* argument can be any type of procedure call or SKILL expression. You can configure the pre-extraction callback to use existing LVS results by using the following predefined symbols:

| Symbol         | Definition                                                     |
|----------------|----------------------------------------------------------------|
| lvs_rundir     | LVS Run Directory                                              |
| starrc_rundir  | StarRC Extraction Directory                                    |
| cci_runset     | Calibre runset file for LVS                                    |
| cci_query_file | Calibre query file for Calibre Connectivity Interface database |
| herc_runset    | Hercules runset for LVS                                        |

The following example executes a call to procedure `UserPreExtractionCB`, which uses the `lvs_rundir` and `cci_query_file` symbols as arguments. These symbols are only valid if the LVS process is already included in the tasks.

```
PRE_EXTRACTION_CALLBACK: UserPreExtractionCB(lvs_rundir cci_query_file)
```

## View Preprocessing Callback

The view preprocessing callback is a SKILL expression that is loaded and executed after StarRC extraction and before parasitic view generation. You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

```
PREPROCESS_CALLBACK: cmd_string
```

The *cmd\_string* argument can be any type of procedure call or SKILL expression. The following symbols can be used in the argument string as variables or procedure arguments:

| Symbol   | Definition                                                                                             |
|----------|--------------------------------------------------------------------------------------------------------|
| cellview | A dbObject of new empty parasitic cell view before it is populated with parasitics and physical shapes |
| cmdfile  | String object representing the name of the StarRC command file                                         |
| usersym  | Generic symbol which you can set and then evaluate in downstream callback code                         |

The following example executes a call to procedure `UserPreProcCallback`, which uses the `cellview` and `cmdfile` symbols as arguments:

```
PREPROCESS_CALLBACK: UserPreProcCallback(cellview cmdfile)
```

## View Postprocessing Callback

The view postprocessing callback is a SKILL expression that is loaded and executed after the parasitic cell view is populated with parasitics and shapes but before it is saved and closed.

You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

```
POSTPROCESS_CALLBACK: cmd_string
```

The `cmd_string` parameter can be any type of procedure call or SKILL expression. The following symbols can be used in the argument string as variables or procedure arguments:

| Symbol                | Definition                                                                                               |
|-----------------------|----------------------------------------------------------------------------------------------------------|
| <code>cellview</code> | A dbObject of the parasitic cell view after it is populated with parasitics and physical shapes          |
| <code>cmdfile</code>  | String object representing the name of the StarRC command file                                           |
| <code>usersym</code>  | Generic symbol that is set by you in upstream code and then evaluated inside the postprocessing callback |

The following example executes a call to procedure `UserPostProcCallback`, which uses the `cellview` and `usersym` symbols as arguments:

```
POSTPROCESS_CALLBACK: UserPostProcCallback(cellview usersym)
```

## Instance Creation Callback

You can use an instance creation callback procedure to manipulate instance parameter lists, names, coordinate locations, and orientations before placing the instance. A procedure name can be specified on a model-by-model basis in the DFII\_DEVICE\_MAP file using the optional parameter `CALLBACK=procedureSymbol`. This procedure is invoked before creating an instance of the corresponding model type.

The user-defined procedure identified by *procedureSymbol* must be defined to accept two arguments as follows:

| <b>Argument</b>   | <b>Definition</b>                                                                                                                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>argument_1</i> | A defstruct instance that points to a property list of all properties specific to the instance.                                                                                                                                                                                                            |
| <i>argument_2</i> | A generic symbol that you can manipulate within the view-level preprocessing/postprocessing procedures and then call within the instance-level procedures; corresponds to the symbol <i>usersym</i> within the code scope in which the preprocessing/postprocessing/instance-level procedures are invoked. |

The defstruct property list for argument\_1 is as follows:

| <b>argument_1 property</b> | <b>Definition</b>                                                                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>inst</i>                | Instance name of device; type = string                                                                                                                               |
| <i>coordlist</i>           | XY coordinates of the instance; format is list (xcoord ycoord)                                                                                                       |
| <i>orientation</i>         | Orientation of instance (for example, R0, R90); type = string                                                                                                        |
| <i>propList</i>            | List of parameters that are being annotated to the instance; format is<br>list(list(t_propname1 t_propType1 g_value1)<br>list(t_propName2 t_propType2 g_value2) ...) |
| <i>instNodes</i>           | Read-only list of terminal node names; type=list. An example is:<br>list ("M1 DRN" "M1 GATE" "M1 DRN" "M1 BULK")                                                     |

---

## Callback Flow Example

The following example shows a user-defined callback procedure to instantiate a new user-defined property on each MPM device type, depending on a setting in the StarRC command file.

```
; set via_cap property on disembodied property list if
; EXTRACT_VIA_CAPS was used in the StarRC run
procedure(UserParseCmdFile(cmdfile usersym)
let((str stream fields)
 stream = infile(cmdfile)
 while(gets(str stream)
 fields = parseString(str,": \n")
 if(nth(0 fields) == "EXTRACT_VIA_CAPS" &&
 nth(1 fields) == "YES"
 then putprop(usersym t 'via_cap)
)
)
)
)

procedure(UserAddEVCProp(dev usersym)
if(usersym->via_cap
 then dev->propList =
 cons(list("viacap" "string" "TRUE") dev->propList)
)
```

Specify the instance creation callback within DFII\_DEVICE\_MAP as follows:

```
MPM devlib pmos4 ivpcell D G S B CALLBACK=UserAddEVCProp
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
```

Specify a preprocessing callback procedure call in the cockpit configuration file as follows:

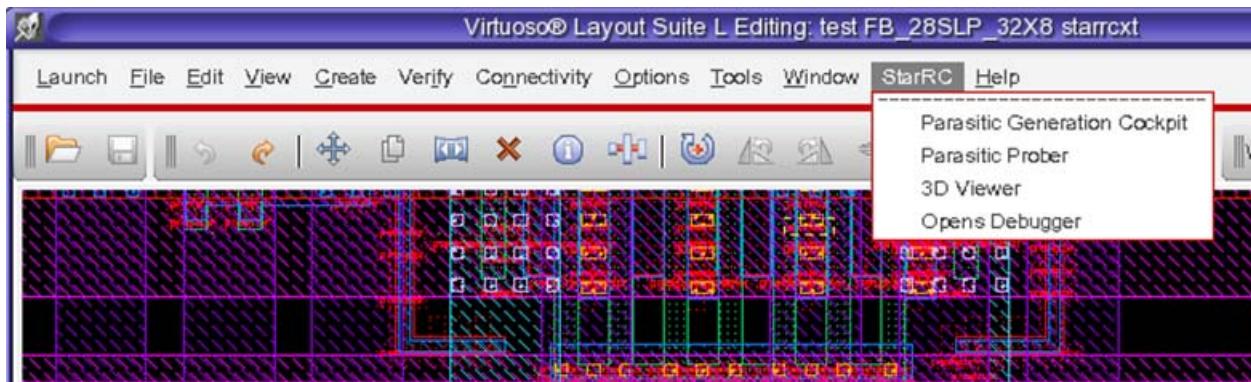
```
PREPROCESS_CALLBACK: UserParseCmdFile(cmdfile usersym)
```

The result of this setup is that devices of type MPM has a string property called viacap in the parasitic view if EXTRACT\_VIA\_CAPS: YES was set in the StarRC run.

## StarRC Parasitic Generation Cockpit GUI

Select the StarRC Parasitic Generation Cockpit by choosing StarRC > Parasitic Generation Cockpit from the Virtuoso menu bar, as shown in [Figure 8-4](#).

*Figure 8-4 Starting the StarRC Parasitic Generation Cockpit in Virtuoso*

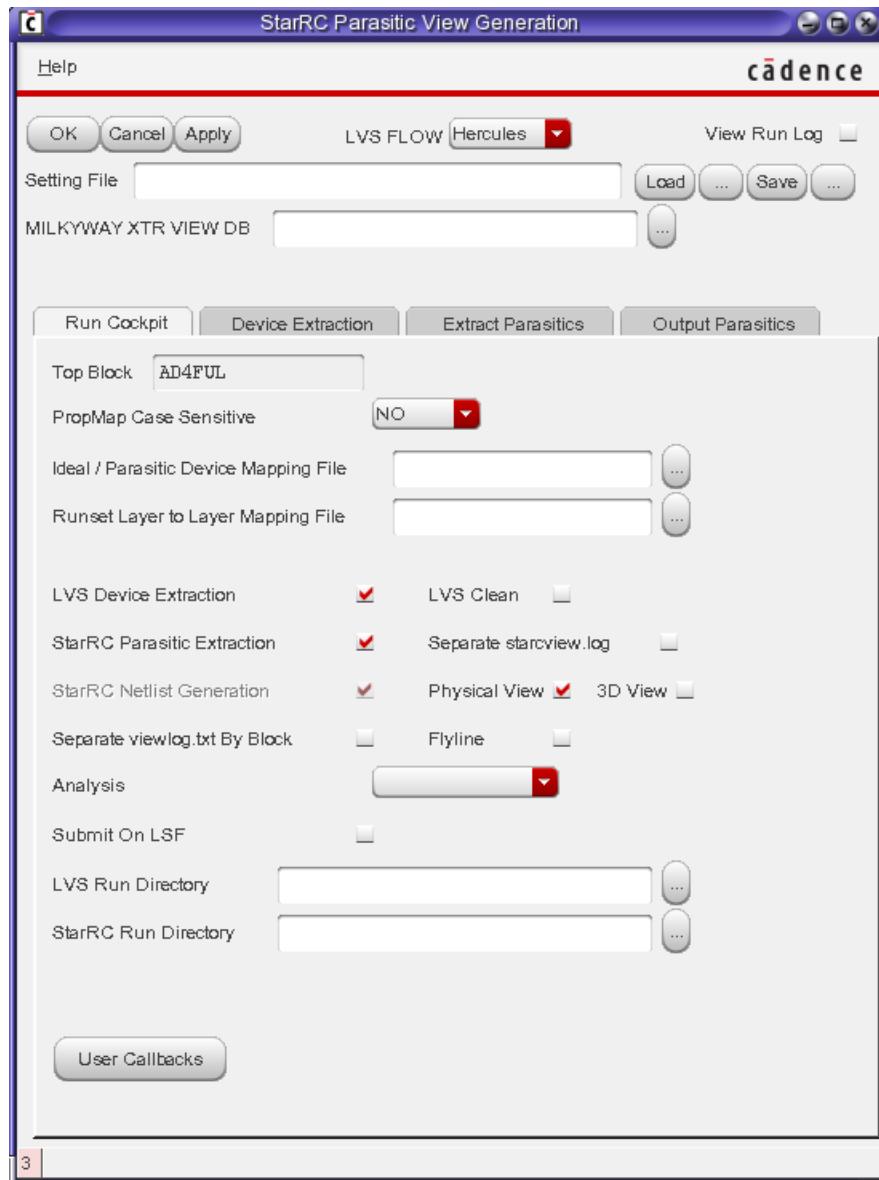


Each tab in the Cockpit represents a step in the flow; click a tab to see the options relevant for a specific step.

[Figure 8-5](#) shows the StarRC Parasitic View Generation dialog box, also called the Cockpit. From the Cockpit, you can execute the IC Validator to StarRC, Hercules to StarRC, or Calibre to StarRC flow either as a complete unit or incrementally in separate stages. You can also regenerate netlists or parasitic views if an extraction run has already been performed.

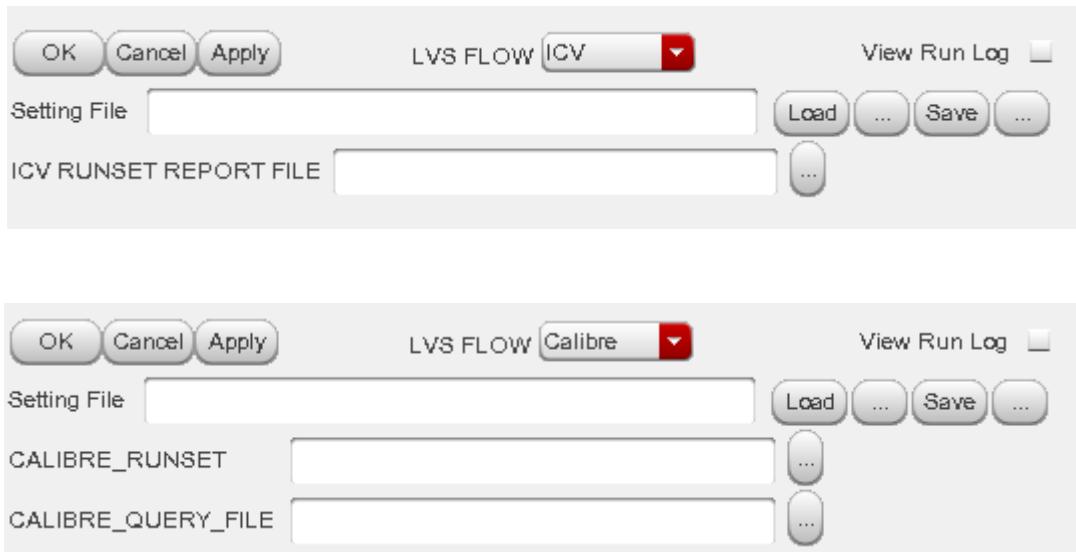
To see the real-time results of the LVS tool or StarRC run, check the View Run Log checkbox.

Figure 8-5 Run Cockpit Tab for the Hercules LVS tool



The top portion of the window is slightly different for different LVS tools, as shown in [Figure 8-6](#).

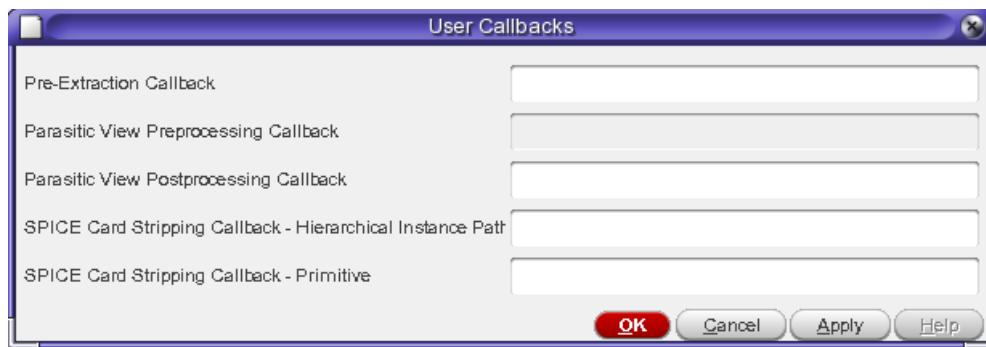
*Figure 8-6 Interface Options For the IC Validator and Calibre LVS Tools*



Some fields require file names or directory names. Use the button labeled “...” to open a file browser instead of manually entering the names.

The User Callbacks button, located near the bottom of the Run Cockpit tab, allows you to specify the names of callback procedures, as shown in [Figure 8-7](#). For more information about callback procedures, see [User-Defined Callbacks](#).

*Figure 8-7 User Callback Selection*



---

## Populating the Cockpit Fields Automatically

You can use a cockpit configuration file to automatically populate many of the fields in the Cockpit. The configuration file is also known as a settings file and must have an extension of .snps\_settings. You can specify a settings file as follows:

- Set the `RC_VI_SETTINGS_FILE` environment variable to point to the configuration file.
- If the `RC_VI_SETTINGS_FILE` variable is not set, the Cockpit looks for the `.snps_settings` file in the directory from which Virtuoso was invoked.

Use one of the following formats for each entry in the configuration file:

```
option_1 : value_1
CONSTANT option_2 : value_2
```

The `CONSTANT` keyword makes the entry for that option not editable in the user interface and can be inserted at the beginning of any line in the settings file.

Choose Load or Save to open a file browser and select a setting file. You can also type the file name in the Setting File box.

[Table 8-2](#) lists the Cockpit field options and values that can be automatically populated.

*Table 8-2 Cockpit Fields and Options That Can Be Prepopulated*

| Tab name                                 | <i>field_option : field_value</i>                                                                                                                                                                                                           |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Run Cockpit                              | <code>DFII_DEVICE_MAP: filepath</code><br><code>DFII_LAYER_MAP: filepath</code><br><code>FLOW: ICV   HERCULES   CALIBRE</code><br><code>PREPROCESS_CALLBACK: SKILL_expression</code><br><code>POSTPROCESS_CALLBACK: SKILL_expression</code> |
| Device Extraction<br>(IC Validator Form) | <code>ICV_RUNSET: filepath</code><br><code>ICV_RUNSET_REPORT_FILE: filepath</code>                                                                                                                                                          |
| Device Extraction<br>(Hercules Form)     | <code>HERCULES_RUNSET: filepath</code><br><code>HERCULES_COMMAND_LINE_OPTIONS: command_string</code>                                                                                                                                        |

*Table 8-2 Cockpit Fields and Options That Can Be Prepopulated(Continued)*

| <b>Tab name</b>         | <b><i>field_option : field_value</i></b>                                                                                                                                                                                                                               |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Extract Parasitics      | TCAD_GRD_FILE: <i>filepath</i><br>MAPPING_FILE: <i>filepath</i><br>CALIBRE_QUERY_FILE: <i>filepath</i> (Calibre flow)<br>CALIBRE_RUNSET: <i>filepath</i> (Calibre flow)<br>EXTRACTION: R   C   RCCOUPLE_TO_GROUND: YES   NO<br>NETLIST_GROUND_NODE_NAME: <i>string</i> |
| Additional Options Form | Any StarRC command file option listed on the Additional Options form                                                                                                                                                                                                   |
| Other                   | SUBNODE_SIZE: <i>subnode_side_length_in_microns</i><br>PORT_ANNOTATION: yes   no<br>PORT_ANNOTATION_VIEW: <i>lib_cel_view</i>                                                                                                                                          |

Use a semicolon (;) to indicate the beginning of a comment inside the .snps\_settings file. In the following example, the second line is ignored:

```
CONSTANT TCAD_GRD_FILE: simple.nxtgrd
; CONSTANT TCAD_GRD_FILE: simple2.nxtgrd
```

In this example, the `-hier` and `-spice` options are ignored:

```
CALIBRE_LVS_CMD_LINE_OPT : -lvs ; -hier -spice
```

By default, the StarRC commands specified in the Cockpit, such as the Additional Option dialog box, are saved to the .snps\_settings file. The GUI settings are also saved to the .snps\_settings file.

To create a new .snps\_settings file,

1. Open a blank Cockpit dialog box.
2. Edit the fields in the Cockpit.
3. Execute your run.
4. If your run is successful, save your settings to a new .snps\_settings file.

The new .snps\_settings file contains all required settings to reproduce a run.

## Advanced Save and Load Mode

A simplified interface for saving or loading a settings file is shown in [Figure 8-8](#).

*Figure 8-8 Advanced Save and Load Mode*



To enable this feature, set the `ADVANCED_SAVE_LOAD` environment variable to YES:

```
$ setenv ADVANCED_SAVE_LOAD YES
```

## Functions in the StarRC Parasitic View Generation Dialog Box

[Table 8-3](#) describes the commands and options in the top section of the StarRC Parasitic View Generation Dialog Box.

*Table 8-3 Commands and Options for StarRC Parasitic View Generation*

| Command or option      | Description                                                                                                              |
|------------------------|--------------------------------------------------------------------------------------------------------------------------|
| OK                     | Starts a Cockpit job and close Cockpit window                                                                            |
| Cancel                 | Closes the Cockpit window                                                                                                |
| Apply                  | Starts Cockpit job and keep the dialog box open                                                                          |
| Flow                   | Specifies one of three LVS tools and changes the options in the Device Extraction Tab accordingly for the specified flow |
| Setting File           | Points to a .snps_settings file                                                                                          |
| Select                 | Opens the file browser to display a setting file                                                                         |
| Load                   | Populates the Cockpit fields with values specified by the setting file                                                   |
| Save                   | Saves the current Cockpit values to the file in the Setting Files box                                                    |
| Milkyway XTR VIEW DB   | Specifies an LVS result database for StarRC to read                                                                      |
| ICV RUNSET REPORT FILE |                                                                                                                          |
| CALIBRE_RUNSET         |                                                                                                                          |
| CALIBRE_QUERY_FILE     |                                                                                                                          |

---

## Run Cockpit Tab

Specify the following settings in the Run Cockpit tab, shown in [Figure 8-5](#).

### LVS Clean

If you select LVS Clean, Virtuoso Integration Cockpit execution checks if the LVS job is compared or not. If not, Virtuoso Integration stops the job. StarRC obtains the LVS results from the following files:

- topblock.LVS\_ERRORS (in the IC Validator and Hercules flows)
- svdb database (in the Calibre flow)

### Physical View

The parasitic view consists of two parts, the physical view and the logical view. Use the physical view for browsing and probing; use the logical view for simulation and schematic view probing.

You can access the Physical View button from the Create button in the top-level Virtuoso menu, as shown in [Figure 8-4](#). This button controls the physical view generation. If it is not selected, no physical view is generated, thus runtime is saved for merging polygons and writing the physical parasitic view.

### Flyline

A flyline is a line that connects nodes on the same net. It helps you to probe point-to-point resistance. Although generating a flyline and storing it in the parasitic view consumes runtime and disk space, StarRC provides the option of flyline generation. By default, flyline generation is disabled.

### LVS Run Directory

The directory in which Virtuoso Integration executes an LVS job. All output files are written to the same directory.

### StarRC Run Directory

The directory in which Virtuoso Integration executes a StarRC job.

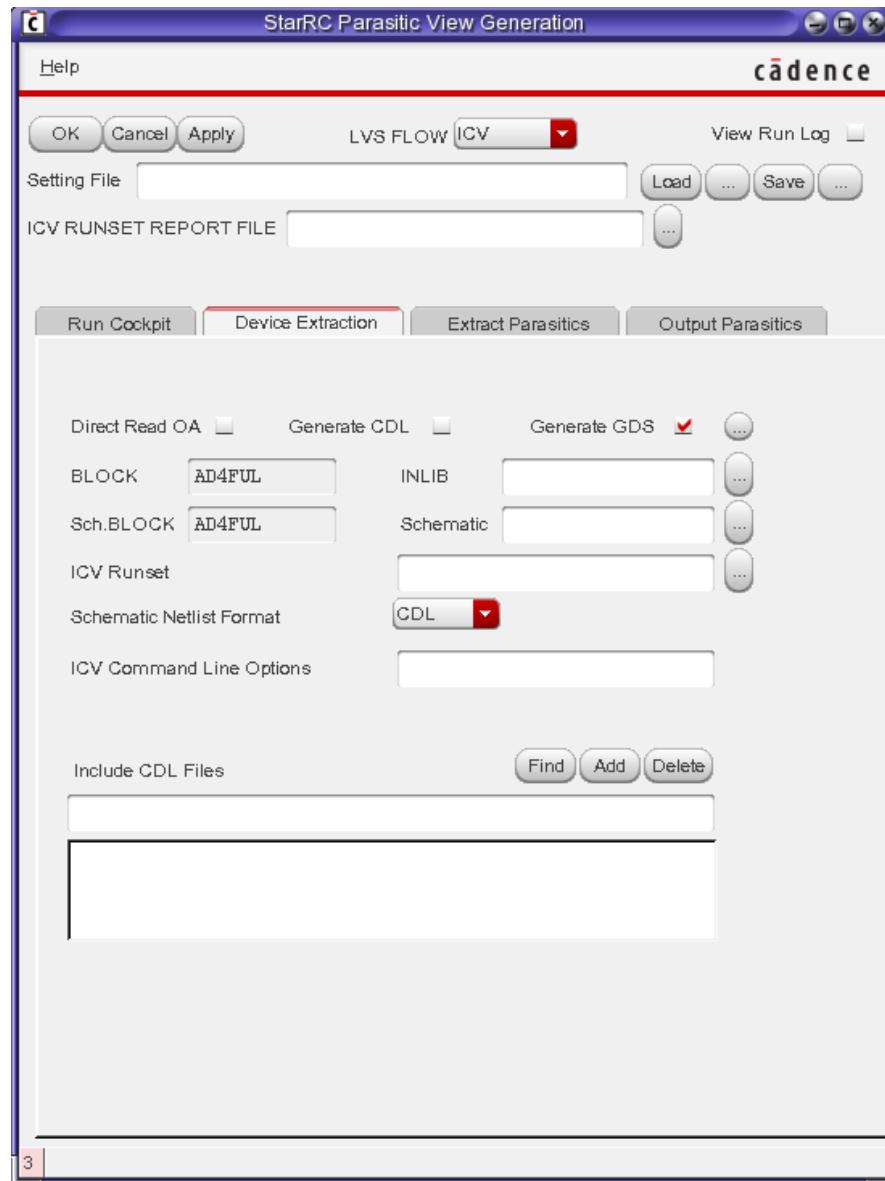
### User Callbacks

Five kinds of callbacks can be specified.

## Device Extraction Tab

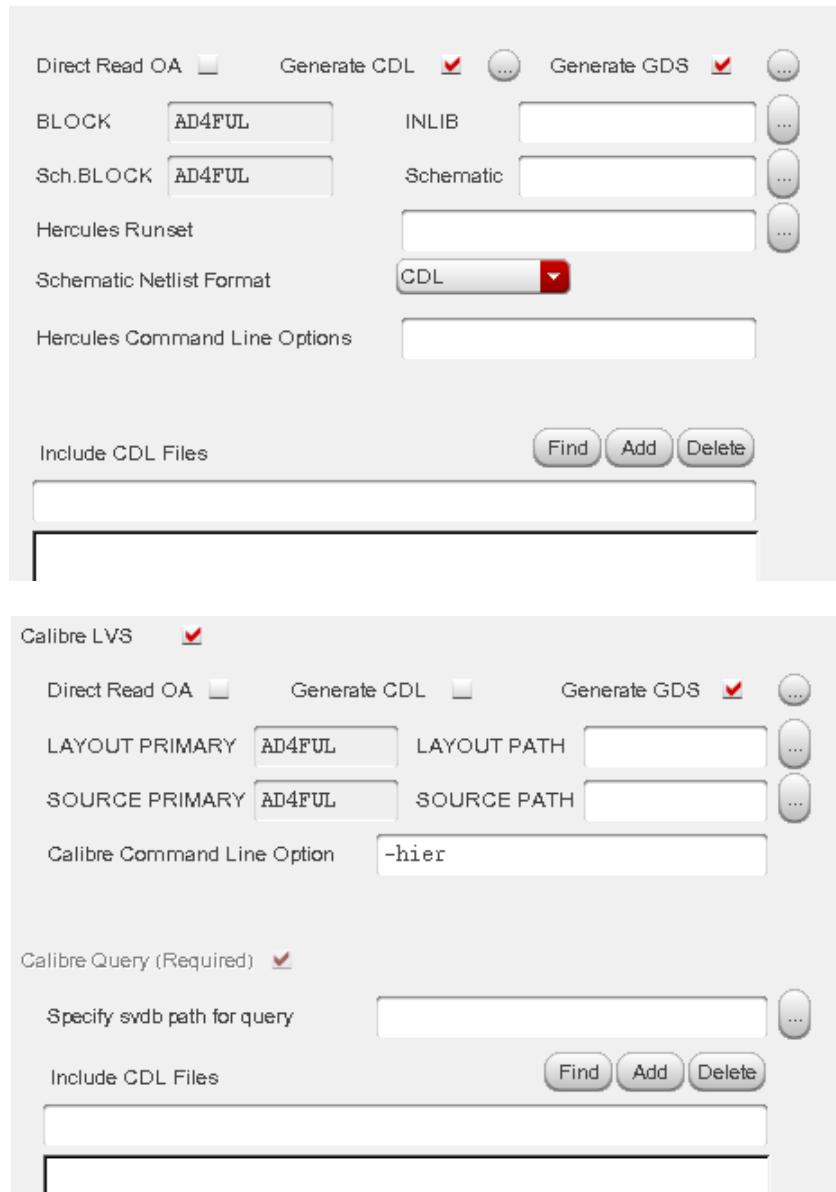
The Device Extraction tab is shown in [Figure 8-9](#) for the IC Validator flow.

*Figure 8-9 Device Extraction Tab for IC Validator Flow*



The Device Extraction tab appearance is slightly different for other LVS tools, as shown in [Figure 8-10](#).

*Figure 8-10 Device Extraction Tab for Hercules and Calibre Flows*



#### Direct OA READ

When you select this option, Virtuoso Integration forces the LVS tool to directly read the OA layout view. This button is only available in Virtuoso 6.0 and later versions. There are some more options for you to choose a view other than layout view or to add a

oa\_layer\_mapping file. For information about the oa\_layer\_mapping file usage and syntax, see the manual for your LVS tool.

#### Generate CDL

When you select this option, Virtuoso Integration writes the netlist to a CDL file that is passed to the LVS tool, instead of writing to a runset- or Cockpit-specified netlist file.

You can modify the streaming option in the CDL out settings form, as shown in [Figure 8-11](#).

#### Generate GDS

When you select this option, Virtuoso Integration streams out the layout view to a GDSII file that is passed to the LVS tool, instead of streaming the layout to the runset- or Cockpit-specified GDSII file.

You can modify the streaming option in the GDSII stream out settings form, as shown in [Figure 8-12](#).

#### Include CDL Files

To use multiple CDL files as input to the Virtuoso Integration LVS tools, specify the CDL files in the GUI.

The View Run Log checkbox near the top of the Device Extraction tab displays the StarRC log file in a separate window.

Figure 8-11 CDL Out Settings

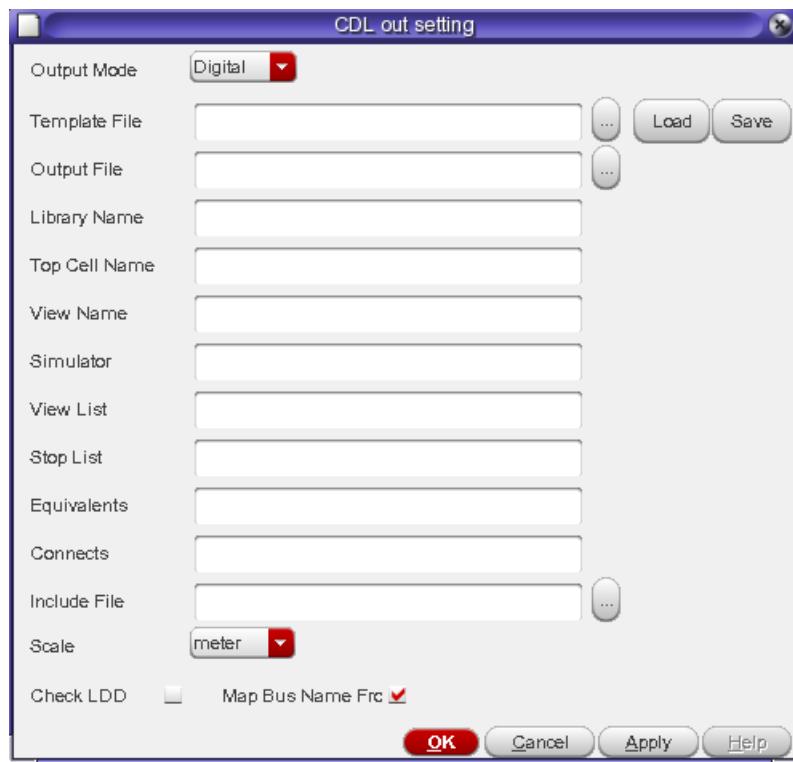
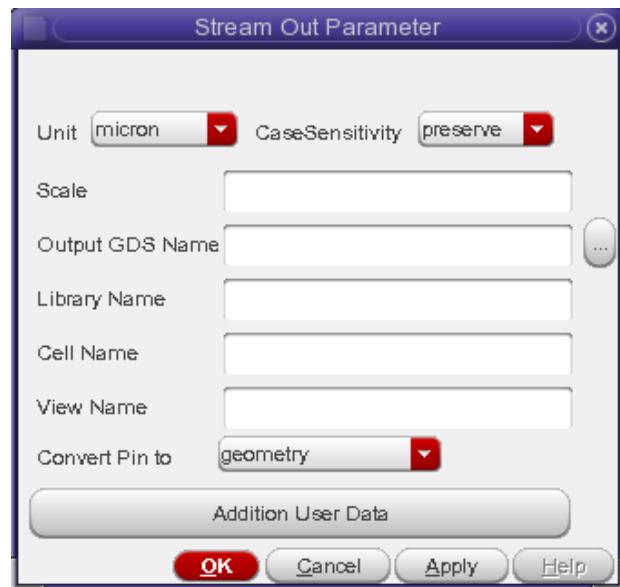


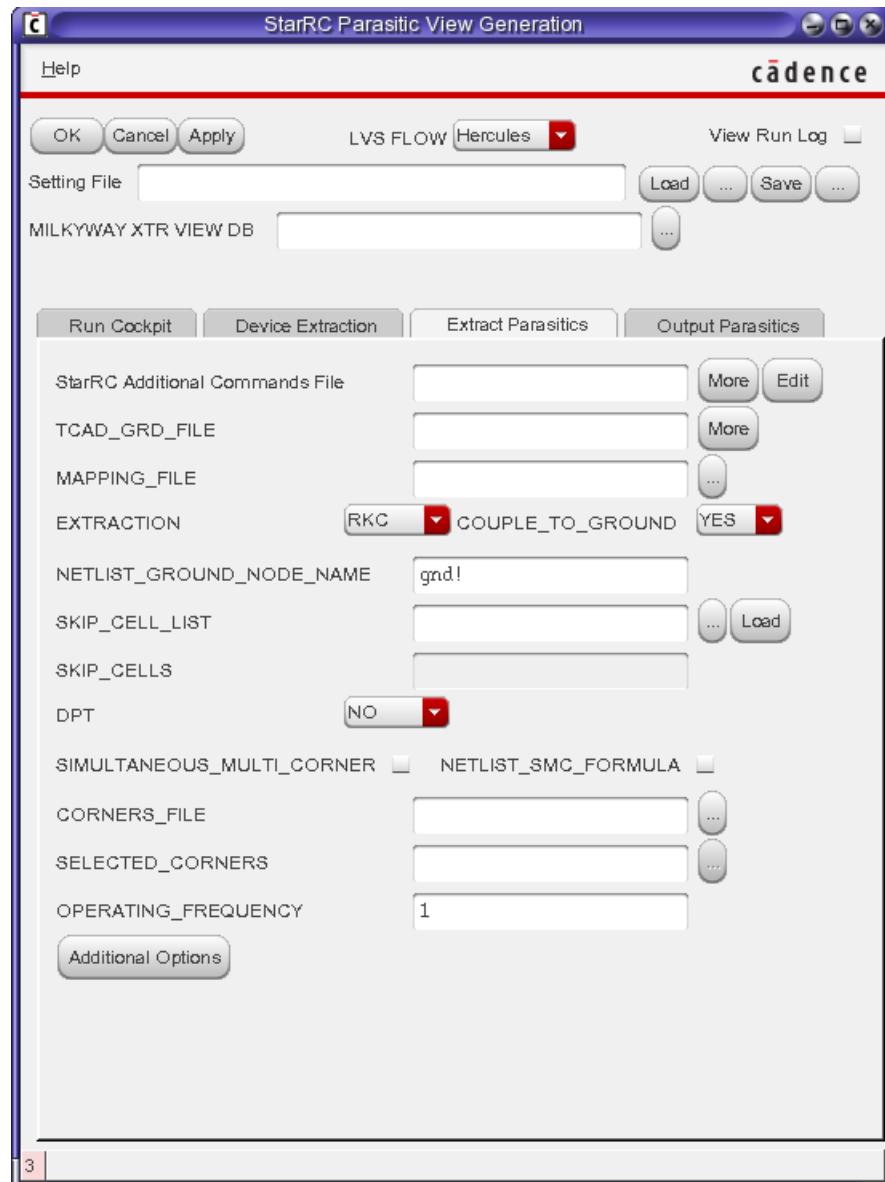
Figure 8-12 GDSII Stream Out Settings



## Extract Parasitics Tab

Figure 8-13 shows the Extract Parasitics tab.

Figure 8-13 Extract Parasitics Tab



### SKIP\_CELL\_LIST

The skip cell list is used for the Virtuoso Integration skip cell flow. The file format is the same as the device mapping file format, which is different from the standard StarRC skip

cell file format. For more information about the device mapping file and skip cell file format, see [The Device Mapping File](#). The following lines are examples of the correct file format:

```
skip_cell1 lib1 cell1 view1
skip_cell2 lib2 cell2 view2
```

#### Additional Command File and Additional Option Form

To assist you in creating the many option combinations needed by Virtuoso Integration for view creation, you can add a command for the Cockpit and also adjust some options with the Additional Options Form.

The extraction options are shown in [Table 8-4](#). If you select the RCC or RCG options, the COUPLE\_TO\_GROUND field shown in [Figure 8-13](#) does not appear because the extraction option determines its setting.

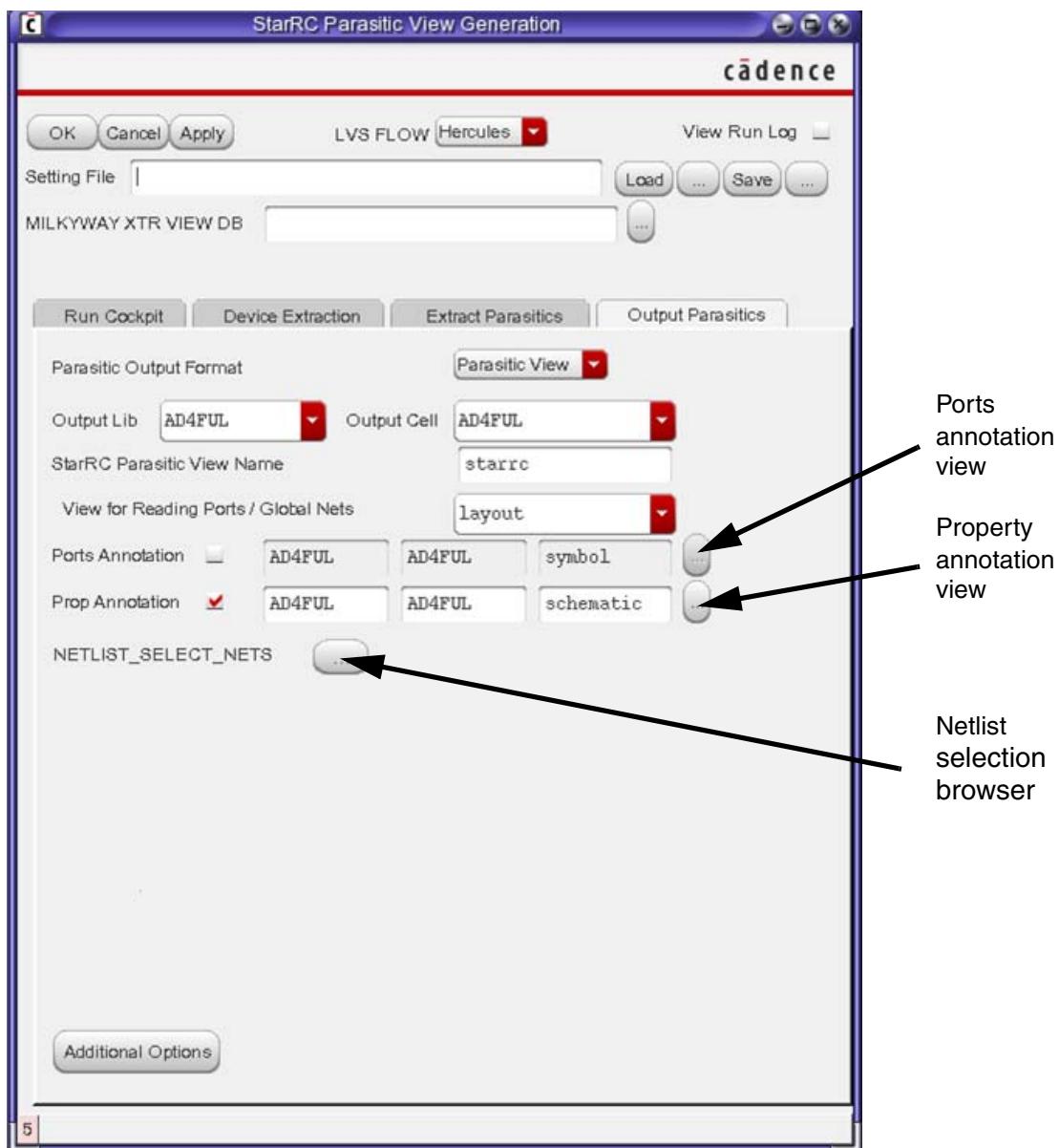
*Table 8-4 Extraction Options*

| RCC                  | RCG                   | NORC                          |
|----------------------|-----------------------|-------------------------------|
| EXTRACTION: RC       | EXTRACTION: RC        | NETLIST_SELECT_NETS: !*       |
| COUPLE_TO_GROUND: NO | COUPLE_TO_GROUND: YES | NETLIST_INSTANCE_SECTION: YES |

## Output Parasitics Tab

Figure 8-14 shows the Output Parasitics tab. You can use Virtuoso Integration as a GUI to configure the StarRC run.

Figure 8-14 Output Parasitics Tab



## Parasitic Output Format

Use this option to select the file format of the output file. Virtuoso Integration can generate not only a parasitic view but also several netlist file formats.

By default, you cannot change the Output Lib and Output Cell names in the Cockpit. This default behavior prevents accidentally writing a newly-generated view to a previous cell from which the .snps\_settings file was created.

To override this default behavior and save the Output Lib and Output Cell names to the .snps\_settings file, set the `RC_SAVE_ALL` environment variable:

```
$ setenv RC_SAVE_ALL NO
```

## Ports Annotation

Use this option when you are not generating a parasitic view as the top-block view, but want to integrate the parasitic view into other testbench circuits. Use the Ports Annotation View option to annotate the correct port and port direction list to the parasitic view. Then the parasitic view can be connected to the other view to form a complete circuit for simulation.

## Property Annotation

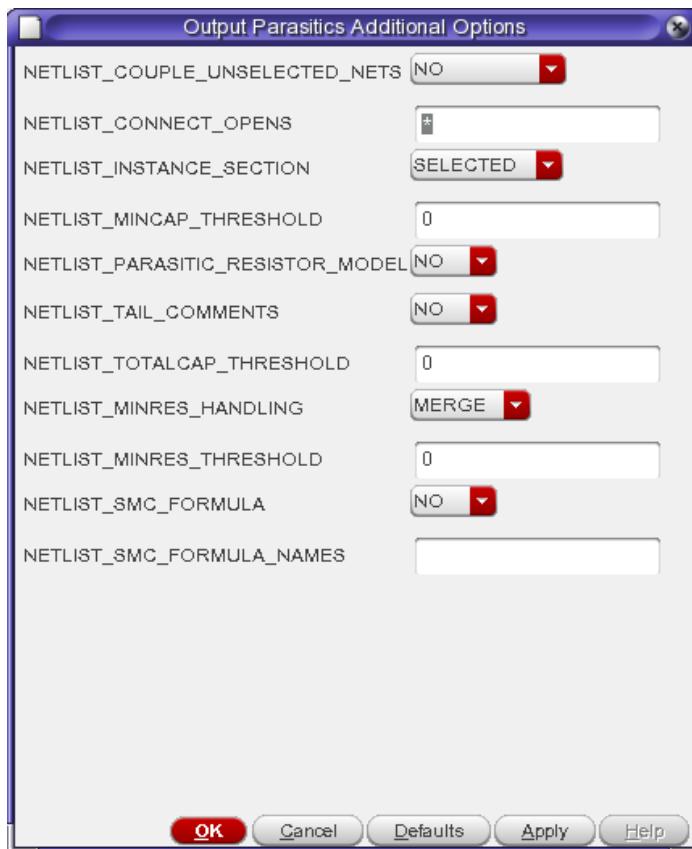
Use this option to specify the view from which to obtain property information for schematic property annotation. This option defaults to the schematic view in the same library or cell window that starts the Virtuoso Integration Cockpit window.

## Netlist Selection Browser

Use this option to select nets for netlisting.

The Additional Options button brings up the form shown in [Figure 8-15](#). These options affect the output netlist.

Figure 8-15 Output Parasitics Additional Options



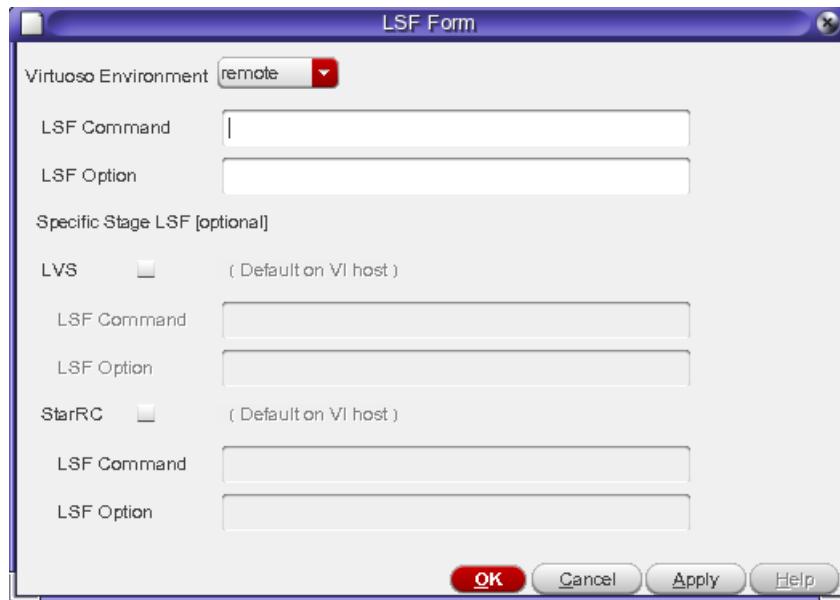
---

## Load Sharing Facility Job Submission

Load Sharing Facility (LSF) support in StarRC Virtuoso Integration gives you the flexibility to control jobs that are submitted to specified farms. By default, all jobs, including LVS and StarRC extraction, are performed on the same remote server. However, if you want to run LVS and StarRC on different farms, use the LSF Form to specify LSF settings for each LVS and extraction task.

Specify whether to use LSF in the Run Cockpit tab with the Submit On LSF checkbox. If you check the checkbox, a button for the LSF settings form appears. An example of the form is shown in [Figure 8-16](#). The LSF Form changes based on the flow selection.

Figure 8-16 LSF Form



If you want to source an environment file before calling StarRC and LVS jobs, such as setup license and path, you can create a wrapper to source these file first. The following is a simple example:

```
#!/bin/csh -fb
foreach arg ($*)
 if("$arg" == "-source") then
 set read_source = 1
 continue;
 endif

 if($read_source == 1) then
 set source_file = $arg
 set read_source = 0
 continue;
 endif

 set args = "$args $last_arg"
 set last_arg = $arg
end

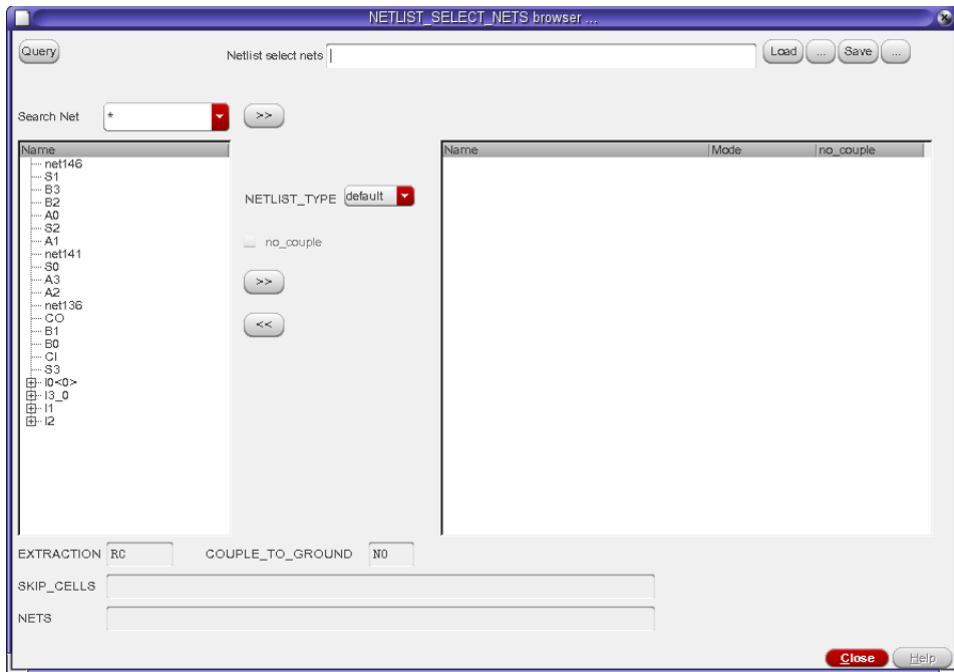
cat $source_file > tmp_file
echo $arg >> tmp_file

echo qsub $args tmp_file
$qsub $args $tmp_file
```

## Selecting Nets for Reporting

By default, the StarRC tool extracts and reports parasitics for all signal nets according to the settings of the EXTRACTION and COUPLE\_TO\_GROUND commands. However, you can limit the output to specific nets or specific types of parasitics by using the NETLIST\_SELECT\_NETS browser, shown in [Figure 8-17](#).

*Figure 8-17 NETLIST\_SELECT\_NETS Browser*



The NETLIST\_SELECT\_NETS browser reads nets from the schematic. If the XREF command is set to NO during extraction, the NETLIST\_SELECT\_NETS browser is disabled. In this case, use the NETLIST\_SELECT\_NETS command instead by including it in an additional StarRC command file specified in the VI Cockpit.

Features of the NETLIST\_SELECT\_NETS browser are as follows:

- Extracted nets appear in the left window; instances appear as subtrees. The list of extracted nets is affected by the settings of the NETS and SKIP\_CELLS commands that were used during extraction. The values are displayed at the bottom of the browser window for reference only.
- You can filter the list of nets by entering a string in the NETS field at the bottom of the browser window. Wildcards \*, ?, and ! are acceptable. In [Figure 8-18](#) part (a), the list of nets is restricted to nets beginning with the letter A.
- You can list all nets in the design by selecting “Show all nets.” In this case, the NETS field is ignored. [Figure 8-18](#) part (b) illustrates the effect.

Entering \* in the NETS field or leaving it blank both mean to show all nets. In these cases, the “Show all nets” checkbox and label are hidden.

- Select specific nets by moving them from the left window to the right window using the button labeled “>>.”

Before moving the selected nets, you can choose which parasitics to report by using the NETLIST\_TYPE pulldown list; the choices are limited by the extraction that was performed. If the COUPLE\_TO\_GROUND command was set to NO during extraction and the selected netlist type is R, CG, or RCG, the checkbox labeled “no\_couple” is available.

- You can also select nets by using the Search Net field; use spaces to separate multiple specifications. Wildcards \*, ?, and ! are acceptable. Click the button labeled “>>” to select the nets displayed in the search results list. [Figure 8-19](#) illustrates this feature.

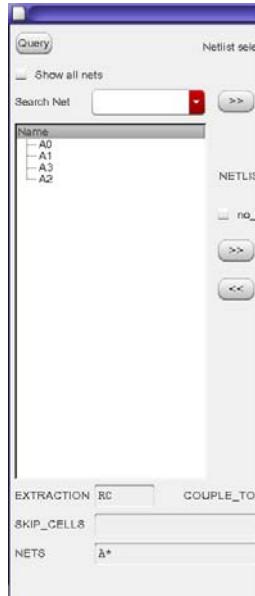
Note:

The drop-down list displays only schematic nets. However, the search function also returns layout-only nets. Using the “>>” button to select nets after a search might also select layout-only nets that meet the search specifications.

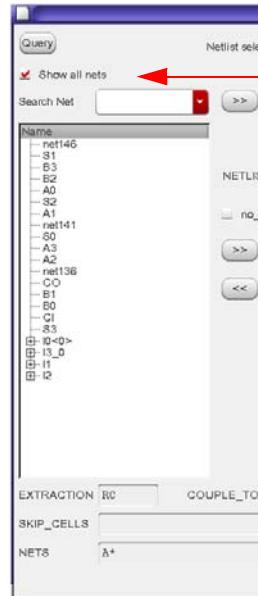
- If you try to select unextracted nets for output, an error message appears.
- If the same net is selected by multiple actions, the last action overrides all previous selection operations for that net.
- Use the shift+click key sequence to select more than one net, as illustrated in [Figure 8-20](#).
- Query the schematic view by using the button in the top left corner. The queried net is highlighted in the left window.
- Load or save a file containing selected nets by using the buttons in the top right corner.

**Figure 8-18 Net Filtering**

(a) Net filtering



(b) Selecting all nets

**Figure 8-19 Net Searching**

Net search for net names that begin with A or end in SEL

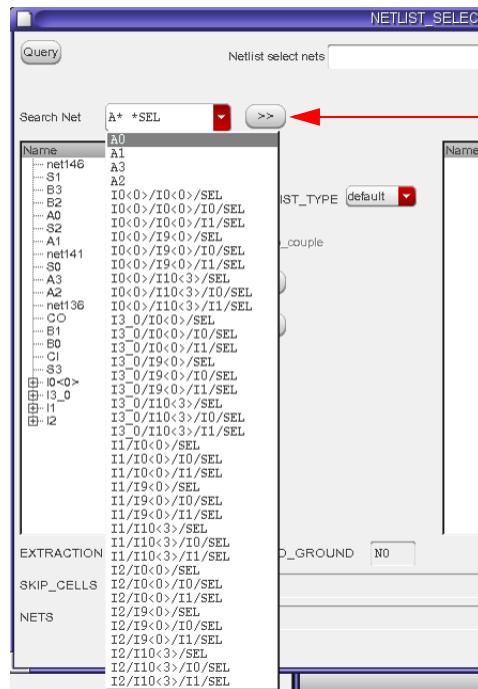
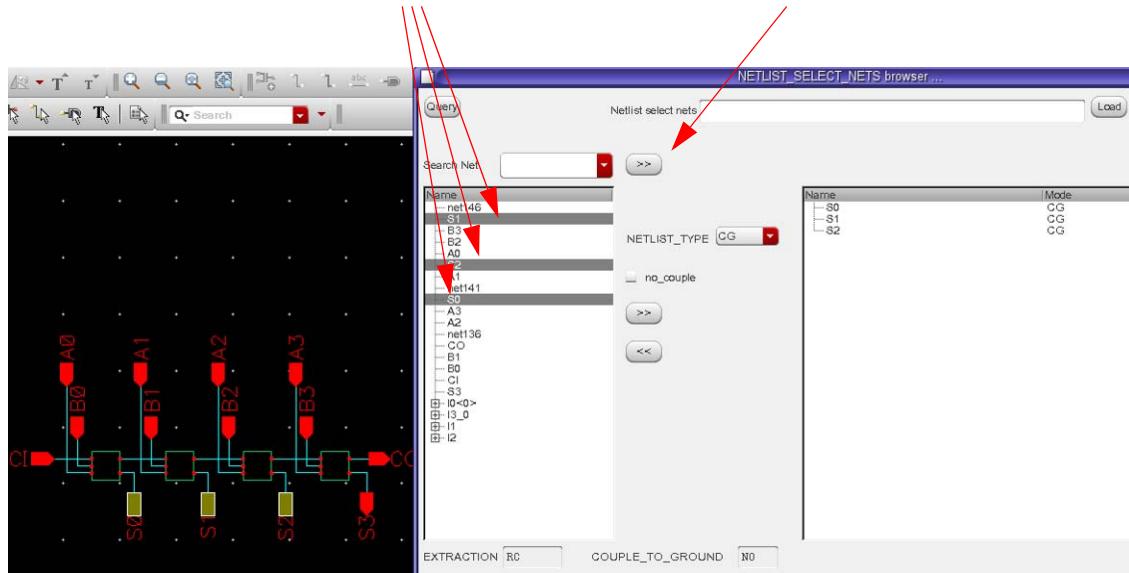


Figure 8-20 Multiple Net Selection

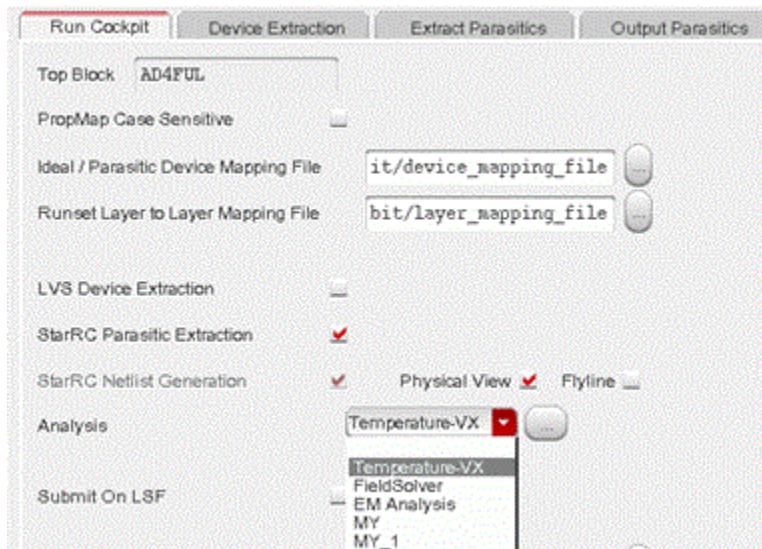
Multiple nets selected with Ctrl + shift + click followed by >> button



## Selecting and Customizing the Analysis Options

In the Run Cockpit dialog box, you can select Analysis options such as Temperature-VX, FieldSolver, electromigration analysis, or customized settings, as shown in [Figure 8-21](#).

*Figure 8-21 Analysis Options in Run Cockpit Dialog Box*



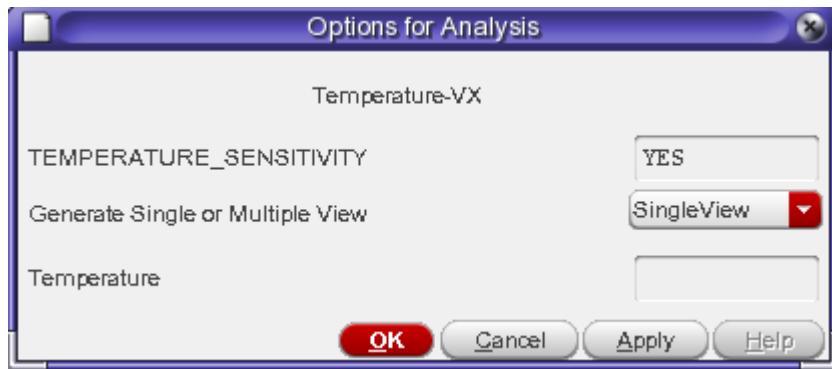
[Table 8-5](#) describes the four predefined Analysis options.

*Table 8-5 Predefined Analysis Options*

| Analysis option | Function                                                                                            |
|-----------------|-----------------------------------------------------------------------------------------------------|
| (blank)         | Uses your current settings; this is the default setting                                             |
| Temperature-VX  | Allows you to output a single view with a selected corner                                           |
| FieldSolver     | Specifies the StarRC FS_EXTRACT_NETS command                                                        |
| EM Analysis     | Specifies the StarRC REDUCTION: NO, EXTRA_GEOMETRY_INFO: NODE RES, and POWER_REDUCTION: NO commands |

To store or edit the StarRC settings for the extraction run, select the analysis option in the Run Cockpit tab and click the "..." button. The Options for Analysis dialog box appears. An example is shown in [Figure 8-22](#) for temperature sensitivity analysis.

*Figure 8-22 Options for Analysis Dialog Box*



To customize the StarRC settings for one or more analysis options,

1. Add the following statement to the .snps\_settings file:

```
ANALYSIS_SETTING: analysis_settings_file_name
```

2. In the analysis settings file, specify the ANALYSIS statement for each analysis setting followed by its corresponding StarRC commands. Use the following syntax:

```
ANALYSIS: [" "] | Simulation | EM Analysis | FieldSolver |
Temperature-VX | custom_setting_name
StarRC_Command_1
[StarRC_Command_2]
...
```

To customize the default settings, specify

```
ANALYSIS: " "
```

The following example defines custom settings named ANALYSIS\_CC and ANALYSIS(CG):

```
ANALYSIS: ANALYSIS_CC
EXTRACTION: RC
COUPLE_TO_GROUND: NO

ANALYSIS: ANALYSIS(CG)
EXTRACTION: C
COUPLE_TO_GROUND: YES
```

## StarRC OA View Creation

The OpenAccess view is a parasitic view that can be used directly in the Custom Designer and Virtuoso design environments. This view includes all parasitic components and network connections along with physical polygons.

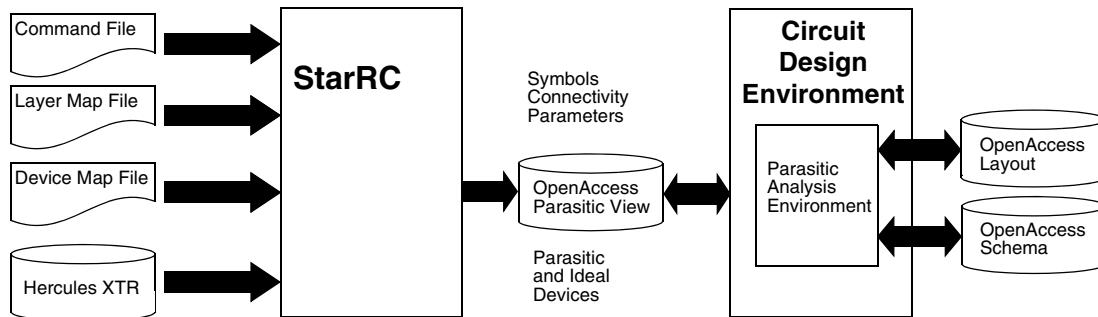
StarRC can execute OpenAccess format parasitic view generation outside Virtuoso. StarRC has many options to control the OA view generation. If using the Virtuoso Integration Cockpit GUI or the Custom Designer GUI, the GUI is helpful for setting up many options.

### The OpenAccess Flow

StarRC provides seamless integration with the Virtuoso Design Environment as shown in [Figure 8-23](#). You can run extraction and generate a parasitic view within the design environment for efficient post-layout simulation. The current parasitic view is generated using the available SKILL-based functions and provides you with an accurate representation of parasitics that can be used to optimize designs.

Accurate layout representation requires netlist connectivity information to instantiate each extracted parasitic (RC) and device in the design. The parasitic view provides you with an efficient and detailed analysis tool in the physical domain. The parasitic view must contain schematic symbols that can be netlisted for simulation as well as used as a graphical tool for identifying parasitic devices.

*Figure 8-23 OpenAccess Flow*



### Linking OpenAccess Libraries

The OpenAccess libraries are dynamic libraries. The standard OpenAccess libraries are located in the lib/std\_oa directory of the standard StarRC installation. These OpenAccess libraries can be downloaded from the following address:

<http://www.si2.org>

## Linking StarRC OpenAccess Libraries

StarRC calls a library called liboaStar-O.so located in the lib directory of the standard StarRC installation.

## Support for Special StarRC Flows

The StarRC simultaneous multicorner flow is supported for OpenAccess views. For more information, see [Simultaneous Multicorner Extraction](#).

The temperature sensitivity flow is also supported for the creation of multiple OA views. For more information, see the reference page for the TEMPERATURE\_SENSITIVITY command.

---

## Skip Cell Mapping

If you use skip cells in the Virtuoso Integration flow, the number of ports and the port names in the OpenAccess symbol view of the skip cell must match the number of ports and the port names created from the layout during layout versus schematic (LVS) checking done before parasitic extraction.

If the number of ports and the port names match, use the OA\_SKIPCELL\_MAPPING\_FILE command to specify a file that defines which cell master to use for the skip cells defined in the SKIP\_CELLS command. The syntax is as follows:

```
INV1 myLib INV symbol
```

If there is a mismatch in either the number of ports or the port names, use the following guidelines to ensure that skip cells are properly connected in the OpenAccess view created by the StarRC tool:

- The number of ports is different, but port names match

Use the SPICE\_SUBCKT\_FILE command to specify SPICE files that contain .subckt definitions for the skip cells. The StarRC tool uses the schematic ports found in the SPICE files instead of the LVS port definitions.

In addition, use the OA\_SKIPCELL\_MAPPING\_FILE command to specify a file that defines which cell master to use for the skip cells defined in the SKIP\_CELLS command.

- The number of ports matches, but some or all port names are different

Use the OA\_DEVICE\_MAPPING\_FILE command to specify a file that contains skip cell mapping definitions. An example of the syntax is as follows:

```
mimcap myLib mimcap symbol PLUS MINUS SHIELD1 SHIELD2
```

- The number of ports and the port names are both different

You can define both the number of ports and the port names in the device mapping file (specified with the `OA_DEVICE_MAPPING_FILE` command).

Alternatively, you can define the number of ports with SPICE .subckt files (specified with the `SPICE_SUBCKT_FILE` command) and the port names with a device mapping file (specified with the `OA_DEVICE_MAPPING_FILE` command).

---

## OpenAccess File Examples

The following sections provide examples for the OpenAccess library definition and the layer and device mapping files. For more information about mapping files, see [The Device Mapping File](#) and [The Layer Mapping File](#).

### OpenAccess Library Definition

The following is an example of the OpenAccess library specified by the `OA_LIB_DEF` command in the StarRC command file:

```
DEFINE w_xxb_fifo1 /remote/cae933/VI/OA/w_xxb_fifo1
DEFINE N901o /testcases/misc/star_virtuoso/N901o
DEFINE cdsDefTechLib /global/apps/ic_61/linux/tools/dfII/etc/
cdsDefTechLib
DEFINE analogLib /global/apps/ic_61/linux/tools/dfII/etc/cdslib/artist/
analogLib
DEFINE US_8ths /global/apps/ic_61/linux/tools/dfII/etc/cdslib/sheets/
US_8ths
DEFINE basic /global/apps/ic_61/linux/tools/dfII/etc/cdslib/basic
```

## OpenAccess Mapping Files

The following is an example of the OpenAccess layer mapping file specified by the StarRC `OA_LAYER_MAPPING_FILE` command:

```
poly PO drawing PO pin PO dummy
metal1 M1 drawing M1 pin M1 dummy
metal2 M2 drawing M2 pin M2 dummy
metal3 M3 drawing M3 pin M3 dummy
metal4 M4 drawing M4 pin M4 dummy
metal5 M5 drawing M5 pin M5 dummy
metal6 M6 drawing M6 pin M6 dummy
metal7 M7 drawing M7 pin M7 dummy
Cont CO drawing nil nil nil nil
pl3co CO drawing nil nil nil nil
VIA1 VIA1 drawing nil nil nil nil
VIA2 VIA2 drawing nil nil nil nil
VIA3 VIA3 drawing nil nil nil nil
VIA4 VIA4 drawing nil nil nil nil
VIA5 VIA5 drawing nil nil nil nil
VIA6 VIA6 drawing nil nil nil nil
```

The following is an example of the OpenAccess device mapping file specified by the StarRC `OA_DEVICE_MAPPING_FILE` command:

```
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
nch N90lo nch auLvs G S D B
pch N90lo pch auLvs G S D B
nch_18 N90lo nch_18 auLvs G S D B
```

---

## StarRC Commands for OpenAccess Parasitic Views

[Table 8-6](#) lists StarRC commands that affect the OpenAccess flow.

*Table 8-6 Commands for OpenAccess*

| Command                                  | Type              | Default            | Description                                                                                                                                                                 |
|------------------------------------------|-------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NETLIST_FORMAT                           | string            | NETNAME            | Set to OA; this command is required                                                                                                                                         |
| OA_BUS_BIT                               | string            | Same as<br>BUS_BIT | Specifies the bus bit delimiter                                                                                                                                             |
| OA_CDLOUT_RUNDIR                         | directory<br>name | none               | Directory containing ihnl subdirectory and mapping files                                                                                                                    |
| OA_CELL_NAME                             | string            | none               | OpenAccess cell name                                                                                                                                                        |
| OA_DEVICE_MAPPING_FILE                   | file name         | none               | File containing device mapping                                                                                                                                              |
| OA_LAYER_MAPPING_FILE                    | file name         | none               | File containing layer mapping                                                                                                                                               |
| OA_LIB_DEF                               | string            | lib.defs           | OpenAccess library definition file; optional                                                                                                                                |
| OA_LIB_NAME                              | string            |                    | OpenAccess library name                                                                                                                                                     |
| OA_MARKER_SIZE                           | float             | 0.1                | Port or subnode marker size (microns); optional                                                                                                                             |
| OA_OVERWRITE_LOCKED_VIEW                 | Boolean           | NO                 | Allows StarRC to overwrite a locked parasitic view                                                                                                                          |
| OA_PORT_ANNOTATION_VIEW                  | string            | null string        | Enables the simulation of a parasitic view; generated by the OpenAccess writer                                                                                              |
| OA_PROPERTY_ANNOTATION_VIEW              | string            | none               | Specifies which schematic library, cell, or view is used to check against ideal devices for schematic-only properties and to attach them into the OpenAccess parasitic view |
| OA_PROPMAP_CASE_SENSITIVE                | Boolean           | NO                 | Case sensitivity of property mapping                                                                                                                                        |
| OA_REMOVE_DUPLICATE_PORTS                | Boolean           | NO                 | Prevents duplication of port names                                                                                                                                          |
| OA_REMOVE_PRIMITIVE_<br>SPICECARD_PREFIX | Boolean           | YES                | Specifies whether to remove SPICE card prefix characters before primitives in ideal instance names                                                                          |

*Table 8-6 Commands for OpenAccess (Continued)*

| <b>Command</b>             | <b>Type</b> | <b>Default</b> | <b>Description</b>                                                                |
|----------------------------|-------------|----------------|-----------------------------------------------------------------------------------|
| OA_REMOVE_SPICECARD_PREFIX | Boolean     | YES            | Specifies whether to remove SPICE card prefix characters from instance name paths |
| OA_SKIPCELL_MAPPING_FILE   | string      | none           | Specifies cell master to use for skip cells in the parasitic view                 |
| OA_VIEW_NAME               | string      | starrc         | Parasitic view name                                                               |

---

## Parasitic Probing

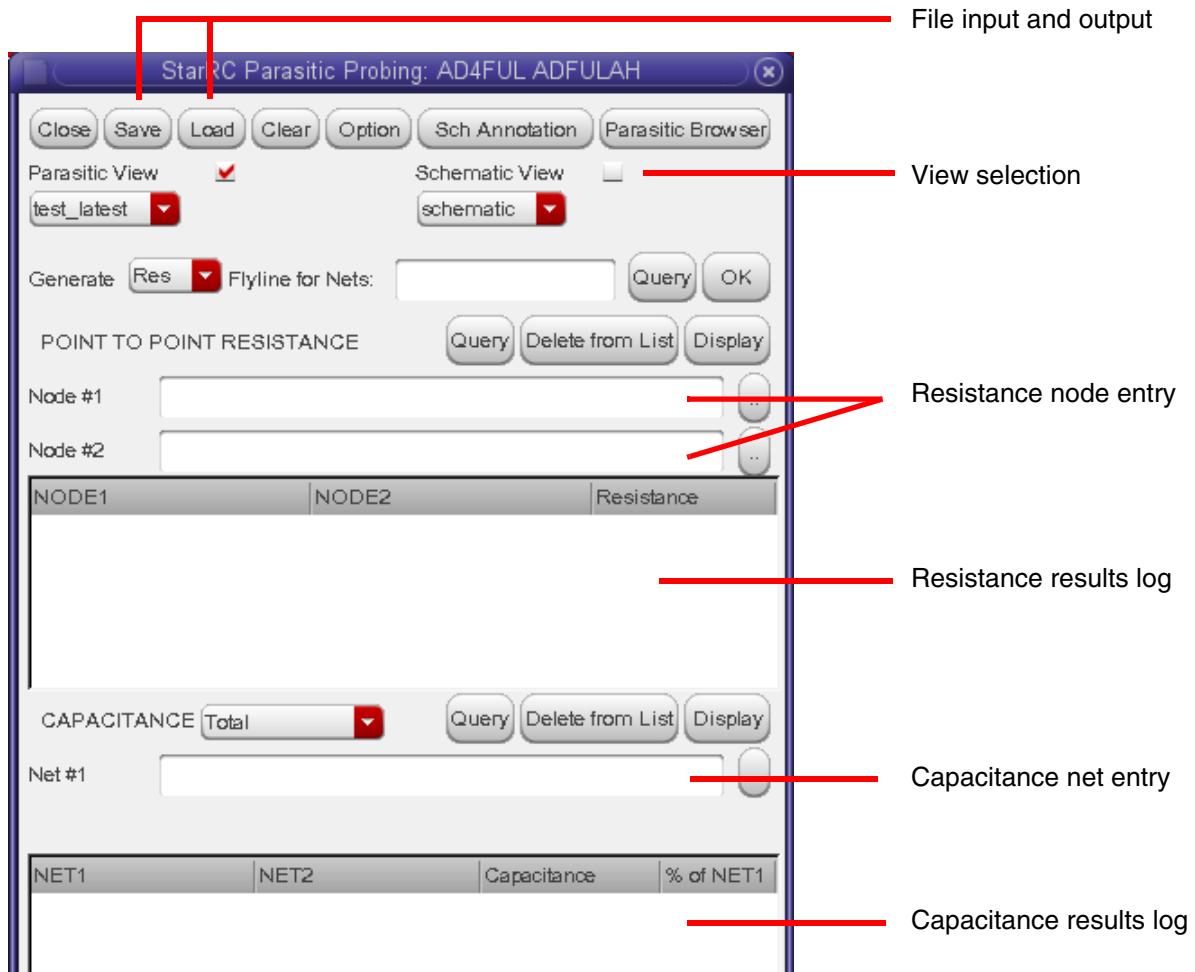
After the parasitic view is generated, you can probe the parasitic view or the schematic view to understand the StarRC extraction results.

---

### StarRC Parasitic Prober

You can access the StarRC parasitic prober through the Parasitic Prober entry of the StarRC pulldown menu within any parasitic or schematic view window. Use the StarRC Parasitic Probing dialog box, shown in [Figure 8-24](#), to probe parasitics within the parasitic view or the corresponding schematic view.

Figure 8-24 StarRC Parasitic Probing Dialog Box



Choose one of the following probing modes:

- Parasitic View
  - Probes port and subnode markers for point-to-point resistance between any two same-net points.
  - Probes interconnect polygons for total net capacitance, with or without couplings to constituent nets.
  - Probes interconnect polygons for total coupling capacitance between two nets.

- Schematic View
  - Probes schematic instance terminals for point- to-point resistance between any two same-net terminals, at any level of hierarchy contained within the schematic cell matching the extracted cell.
  - Probes schematic nets for total net capacitance, with or without couplings to constituent nets, at any level of hierarchy contained within the schematic cell matching the extracted cell.
  - Probes schematic nets for total coupling capacitance between two nets, at any level of hierarchy contained within the schematic cell matching the extracted cell.

The prober also provides the following additional features:

- Highlighting and zooming of parasitic view interconnect polygons corresponding to a previously probed total capacitance or point-to-point resistance result.
- Annotation of total capacitance results to a corresponding schematic view window
- Sorting of logged resistance and capacitance results based on net name or parasitic value
- Output of probed parasitic results to an ASCII report file, as well as input of parasitic results from a previously output ASCII report file
- Activation or deactivation of flylines representing individual parasitic resistors and capacitors

---

## StarRC Parasitic Browser

From the Parasitic Browser, you can Input, search, or query a net name. All the node connections and parasitic devices on this net are listed in the form. You can review this information, select the node, resistor, or capacitor to be deleted or display information about the StarRC view.

Type in a name, then click Search, and the Parasitic Browser displays all the net names contained the input pattern. When the required name is shown, click Done. The Parasitic Browser parses the net to show all the physical nodes in the Connection field.

*Table 8-7 Parasitic Browser Button and Field Functions*

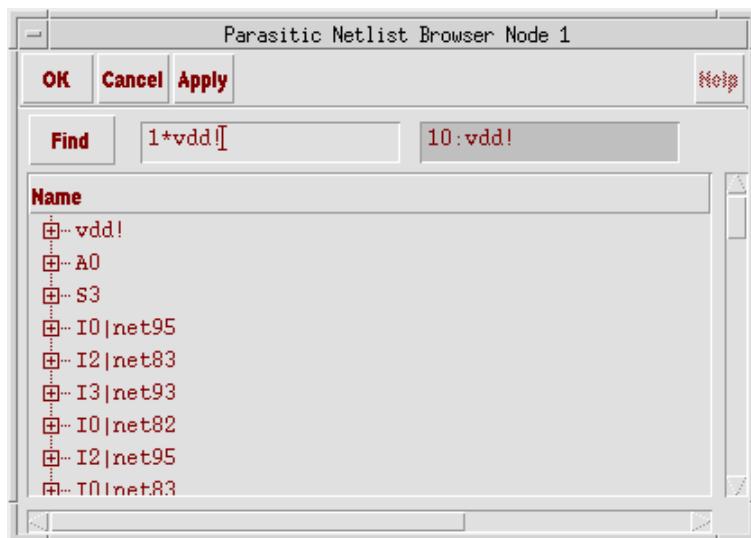
| Form button or field | Function                                                                                                    |
|----------------------|-------------------------------------------------------------------------------------------------------------|
| Query                | Chooses a net and sends it to the Connection field by querying a name from schematic view or parasitic view |
| DISPLAY              | Displays the selected node to be highlighted in the parasitic view                                          |
| DELETE               | Deletes the selected node                                                                                   |

---

## StarRC Parasitic Netlist Browser

The Parasitic Netlist Browser, as shown in [Figure 8-25](#), helps you find a specific node. Click the plus sign to the left of a net name to expand the net and show the signal group. Click the minus sign to collapse the signal group into a net.

*Figure 8-25 Parasitic Netlist Browser*



Use the Parasitic Browser to browse, search, and select a net name to send to the prober. You can enter a net name containing a wildcard in the Find box. When you click Find, the matching net name appears. When you click Apply, the pattern is sent to the node or net field.

Note:

Only one wildcard is allowed in the search string. A string containing more than one wildcard or the question mark (?) character might not return the expected results.

---

## View Selection

Parasitic view probing is done either within the parasitic view or the schematic view. The Parasitic View and Schematic View radio buttons at the top of the prober form enable probing for either the selected parasitic view or the selected schematic view. Only one probing mode is selectable at a time, but the mode can be changed at any time using these radio buttons.

The menus beneath the Parasitic View and Schematic View radio buttons enable you to select the specific view names to be used for each mode. The parasitic view name or schematic view name can be changed at any time. Note that when parasitic view probing is in effect, the selected schematic view is not relevant and is ignored. However, when schematic view probing is in effect, the selected parasitic view specifies the view from which resistance and capacitance parasitics is read.

---

## Dynamic Flylines for Probing

When you want to probe point-to-point resistance or capacitance, flylines can help you select the right node pair. Virtuoso Integration has the capability to generate flylines dynamically only for certain nets.

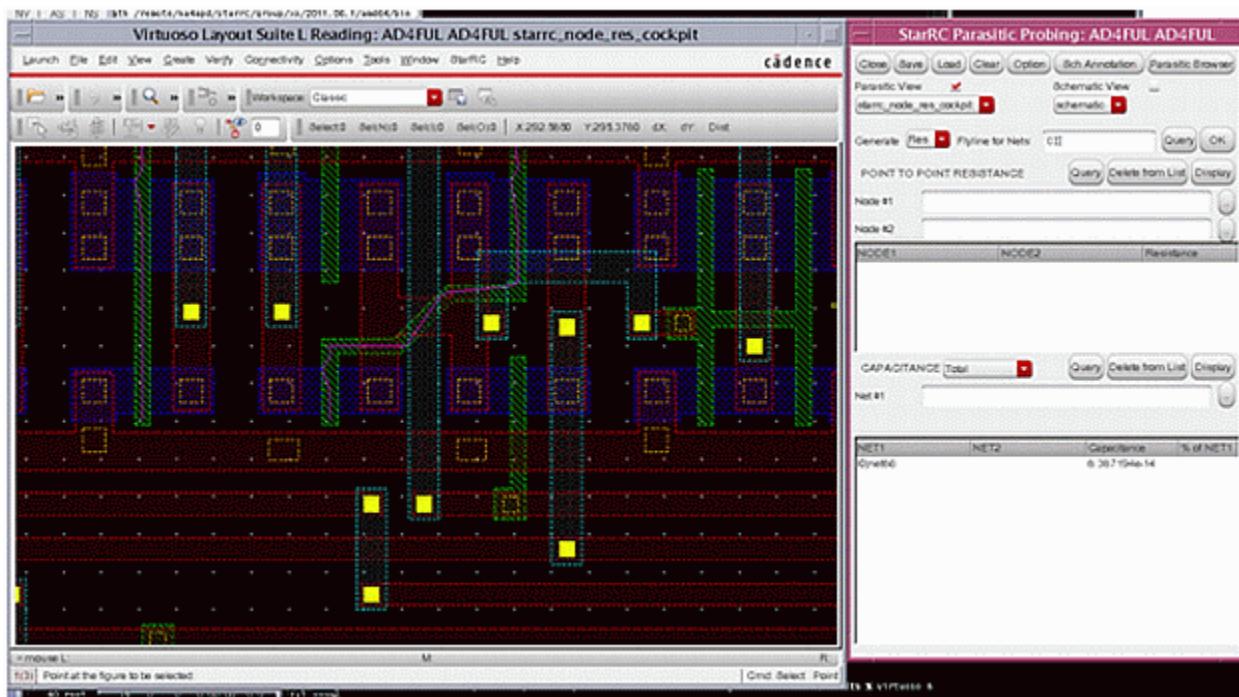
In the Flyline for Nets field in the Prober GUI, shown in [Figure 8-26](#), enter the net names or click Query to start a search. When you click OK, Virtuoso Integration generates the flylines for the specified nets.

*Figure 8-26 Specify Nets to Generate Dynamic Flyline*



The flylines can assist you in point-to-point probing. [Figure 8-27](#) shows the flyline generated for net C1.

*Figure 8-27 Dynamic Flyline Probing*



## Point-to-Point Resistance Probing

Point-to-point (P2P) resistance probing allows you to query two same-net nodes from the selected view and derive the corresponding parasitic path resistance between these two nodes. This calculation uses resistance network reduction techniques to reduce the network down to the two selected nodes and report the equivalent resistance between the two nodes.

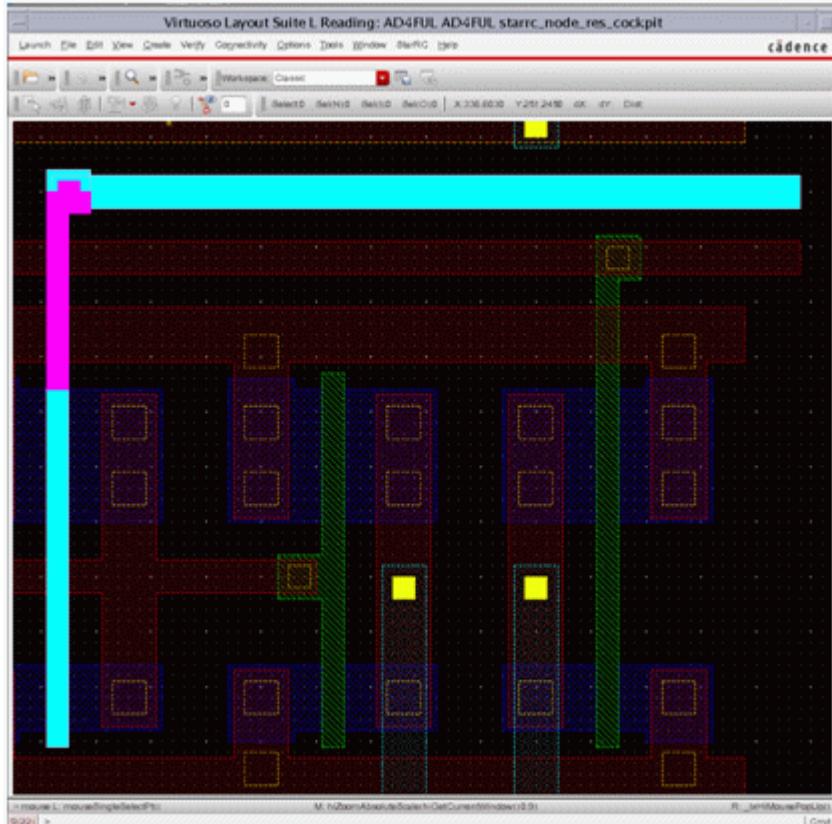
In addition to probing nodes, you can also manually enter the node names into the corresponding text boxes on the prober form to compute equivalent point-to-point resistance.

## Double Highlighting of Point-to-Point Resistance Probe Results

Virtuoso Integration can display double highlighting for the probe results of a point-to-point resistance network. This feature can help you visualize the parasitic extractions results more easily.

**Figure 8-28** shows an example of double highlighting. The entire net is highlighted in cyan. The point-to-point resistance probe results are highlighted in magenta.

*Figure 8-28 Double Highlighting of a Point-to-Point Resistance Network*



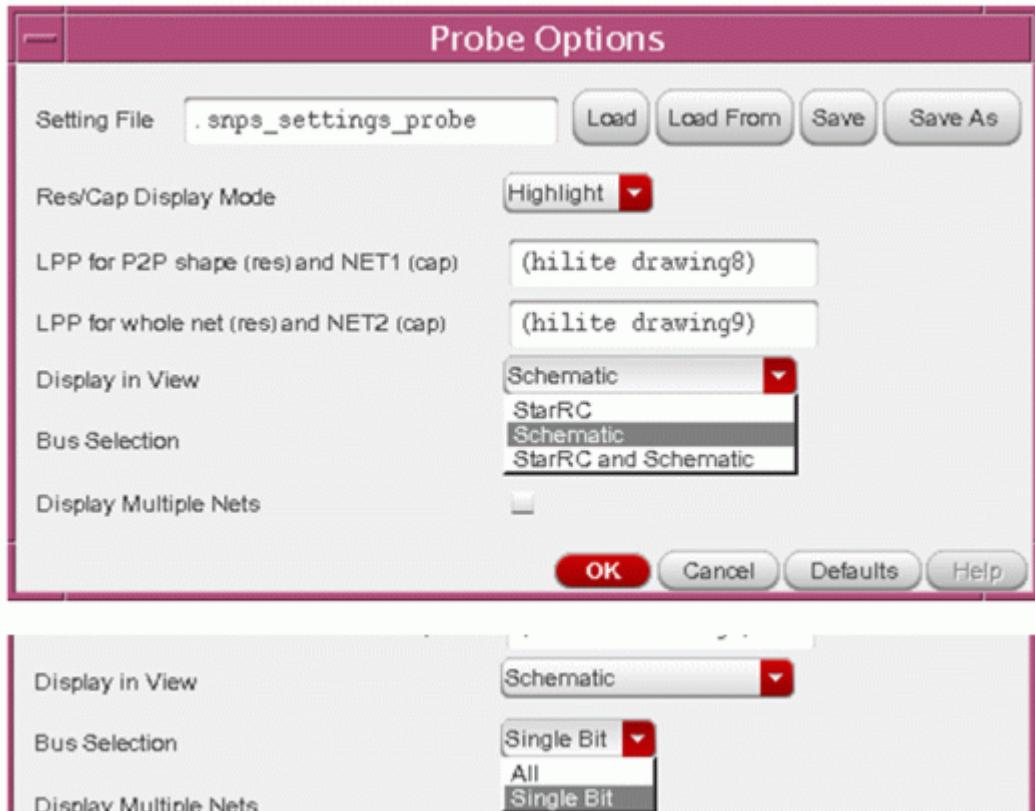
To enable double highlighting, add the following commands to your StarRC command file:

```
REDUCTION : NO
EXTRA_GEOMETRY_INFO: NODE RES
```

## Specifying and Saving the Probe Options

To adjust the highlighting properties, click the Option button. The dialog box shown in [Figure 8-29](#) appears.

*Figure 8-29 Probe Options Dialog Box*



To save your configuration for future sessions, specify the following settings in the .snps\_settings\_probe file:

- DISPLAY\_IN\_VIEW: STARRC | SCHEMATIC | STARRC\_AND\_SCHEMATIC

The default is STARRC. Specify SCHEMATIC or STARRC\_AND\_SCHEMATIC to send highlights to the schematic view.

- BUS\_SELECTION: ALL | SINGLE\_BIT

The default is ALL, which selects the entire bus in a node or wire. If you specify the SINGLE\_BIT option, Virtuoso Integration opens the Select Bus Bit dialog box for you to specify the name.

- DISPLAY\_MULTI\_NETS: NO | YES

The default is NO; which specifies that a new highlight clears a previous highlight.

## Highlighting or Blinking Probe Results

The Res/Cap Display Mode gives you two choices: Highlight and Blinking.

You can specify the color and fill of the highlights in the LPP settings boxes. The color and fill is picked up from your Virtuoso display resource.

If you set the Res/Cap Display Mode to Blinking, the Prober displays the probe results as blinking highlights. Note the following limitations to the blinking highlights:

- Only the highlight for the whole net blinks. The highlight of the P2P partial net does not blink.
- In blinking mode, the Prober changes the display resource to have blinking LPP. A new packet—with “B” appended to the original name—is created and used for this LPP.
- The prober tries to add temporary shapes to the view for blinking effects.

If you save the view, Virtuoso Integration asks you to confirm saving the changes made by the blinking mode. If you do not want to save the changes made by Virtuoso Integration, choose Cancel.

---

## Probing a Single Bus Bit or All Bus Bits

You can choose to probe a single bus bit or all bits in a bus, as shown in [Figure 8-30](#).

*Figure 8-30 Bus Bit Selection in the Probe Options Dialog Box*



[Figure 8-31](#) shows an example of probing results when you choose to probe all bus bits.

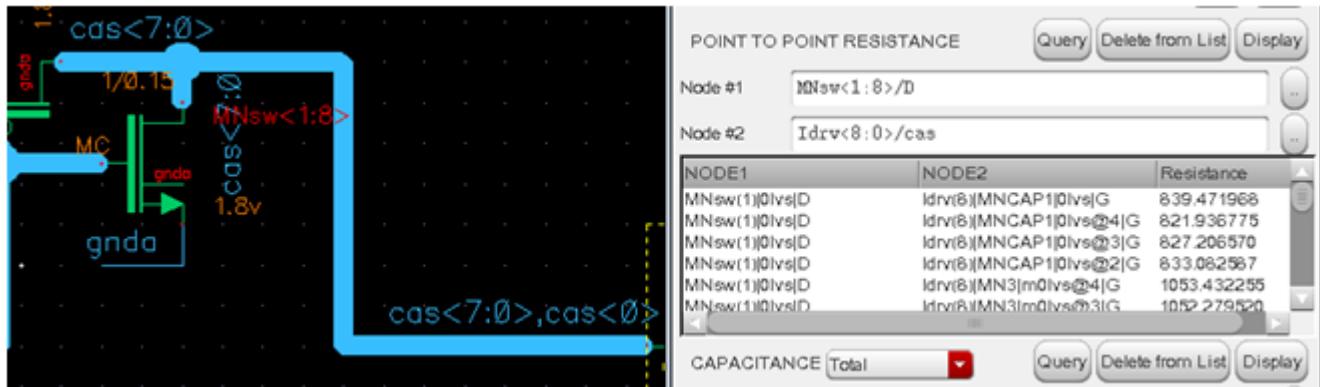
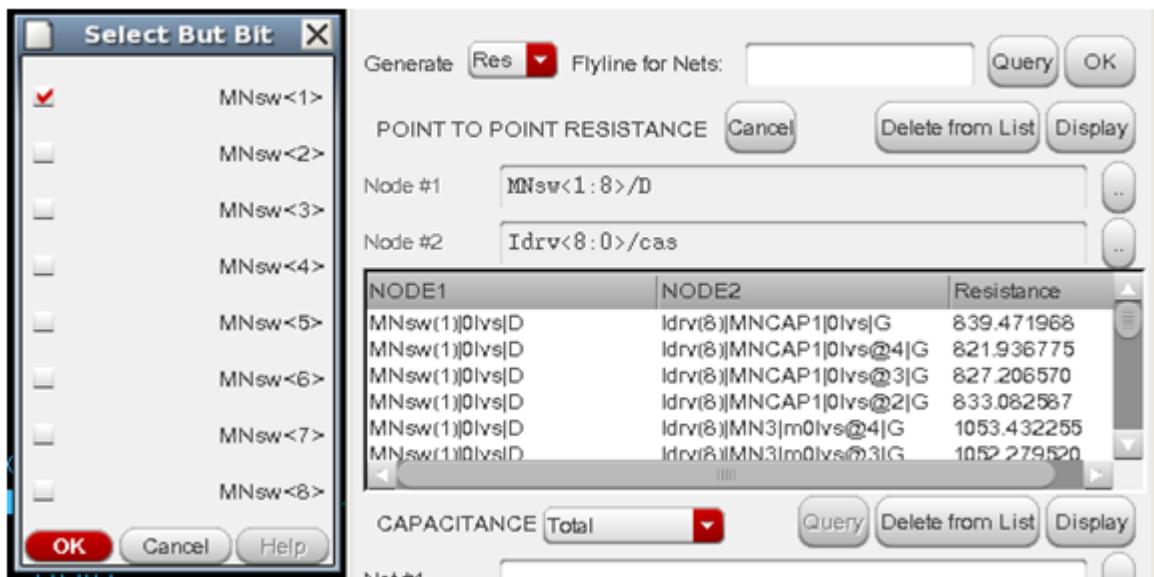
**Figure 8-31 Probing All Bus Bits**

Figure 8-32 shows an example of probing results when you choose to probe a single bus bit.

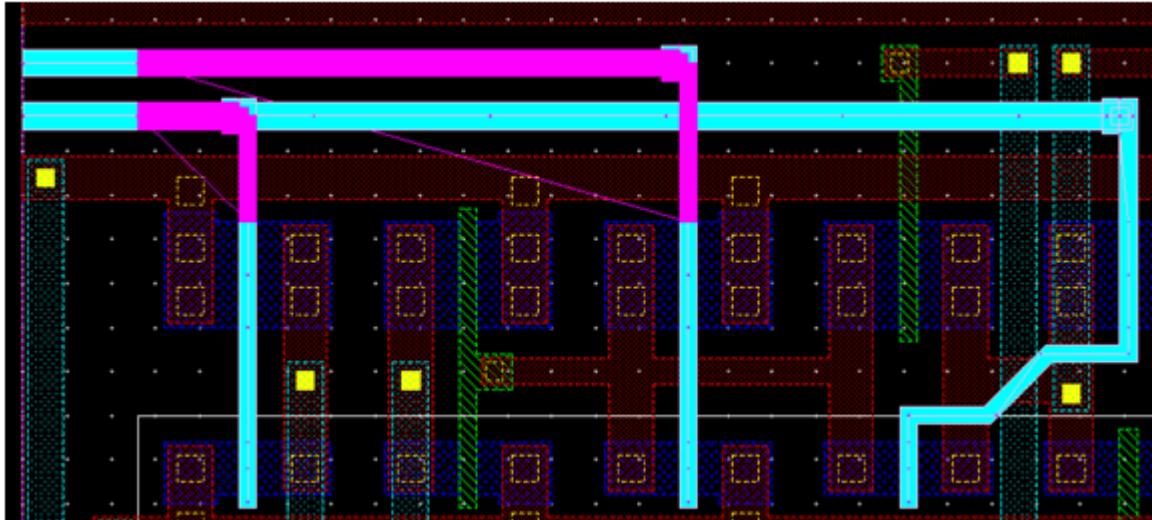
**Figure 8-32 Probing a Single Bus Bit**

---

## Displaying Multiple Nets

To display multiple nets, select Display Multiple Nets in the Probe Options dialog box, as shown in [Figure 8-30](#). Virtuoso Integration displays each net in different color, as shown in [Figure 8-33](#).

*Figure 8-33 Displaying Multiple Nets*



---

## Parasitic View Probing

When you are probing point-to-point resistance inside the parasitic view, it is only possible to query polygons listed as node (that is, port or subnode) layer-purpose pairs in the Runset Layer to DFII Layer mapping file. If no node LPPs were specified in that file, point-to-point parasitic probing is deactivated in the probing utility.

---

## Schematic View Probing

Schematic-based probing of parasitic view resistance is available both when probing the top-level schematic corresponding to the parasitic view block and when probing out-of-context by descending the schematic view hierarchy. With this feature, you can probe instance terminals within the schematic view, and corresponding nodes are located within the selected parasitic view. It is possible to probe a hierarchical schematic and achieve resistance results even if the underlying parasitic extraction flattened all hierarchy (that is, `SKIP_CELLS: ! *`). If you do not specify the `OBSERVATION_POINTS` command, StarRC adds it to the Virtuoso Integration run.

The matching of instance terminals from a hierarchical schematic to parasitic subnodes from a flattened extraction is accomplished as follows:

- The StarRC `OBSERVATION_POINTS` command generates parasitic subnodes corresponding to hierarchical interactions of cells that are not skip cells. Therefore it is possible to probe the terminal of a cell instance in schematic that does not correspond to a skip cell and still have the parasitic prober find a directly corresponding node in the flat parasitic extraction.
- There are several situations where it is not possible to match a probed schematic instance terminal to a subnode in the parasitic view.

The `OBSERVATION_POINTS` command only generates nodes when net material in the parent cell interacts with port material in the child cell in layout. If, for example, a top-level net in layout connects through a level-1 instance down to port material inside a level-2 instance, with no port material existing specifically in the level-1 block, no observation point node is generated for the level-0 to level-1 interaction.

Nodes are only generated for parent- and child-cell interactions when the child cell is listed as a successfully matched EQUIV point (Hercules or IC Validator flow) or HCELL (Calibre flow) in the LVS output.

When no direct match is found, the parasitic prober searches for all valid parasitic terminal nodes which (a) connect to the same top-level net as does the probed schematic instance terminal, and (b) are located inside the specific instance probed in schematic. All matching parasitic nodes that meet these criteria are used when reporting point-to-point resistance. Therefore, if you probe one schematic terminal that matches M parasitic terminal nodes, and a second schematic terminal that matches N parasitic terminal nodes, a total of  $M * N$  point-to-point resistance results are reported.

---

## Probed Results Log and Cross-Probing

As point-to-point resistances are probed within either the schematic view or the parasitic view, results are stored within the results logs (see [Figure 8-24](#)). If a window containing the parasitic view is open, you can select a previously probed resistance entry in the results log and click Display to highlight or zoom-in to the node shapes between which the resistance value applies. A flyline also appears between the two node shapes. In this manner, you can probe point-to-point resistance results in the schematic view and then select the result in the prober to see it highlighted in the parasitic view.

Note that the highlighting and zooming functions work properly only if node markers are contained inside the parasitic view for the two nodes listed in the selected resistance result. It is possible to probe the schematic and build resistance results in the results log, but be unable to highlight and zoom in to the results due the fact that node shapes were not generated during parasitic view generation. For more information about parasitic view node shapes, see [The Layer Mapping File](#).

## Prober File Input and Output

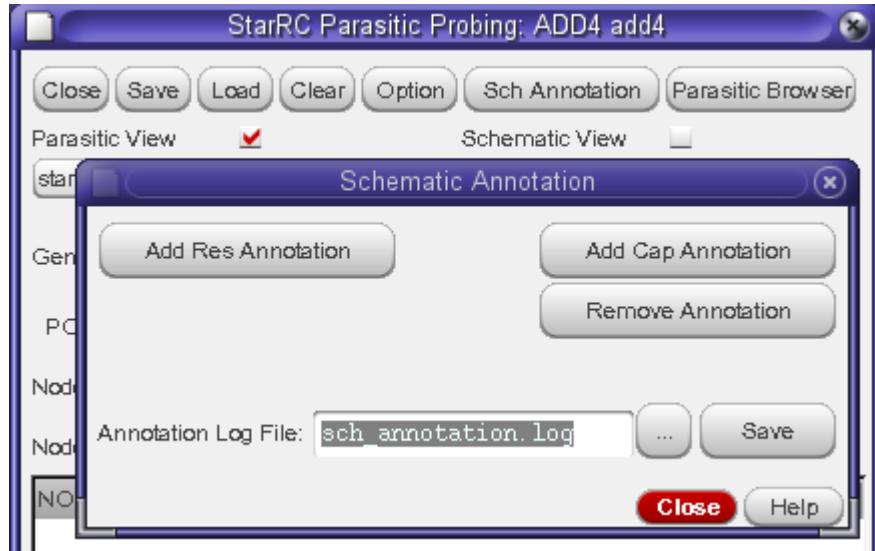
The toolbar of the parasitic prober contains two buttons used to store and load parasitic results between the prober form and a text file.

| Button         | Description                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------|
| Save to File   | Saves all results contained in the resistance results log and capacitance results log to a specified text file. |
| Load From File | Loads capacitance and resistance results from a specified text file into the prober form results logs.          |

## Schematic Annotation

The Parasitic Prober provides the ability to annotate schematic net names with total capacitance or total resistance information from the parasitic view specified in the prober form. Click the “Sch Annotation” button on the Parasitic Probing form to activate the Schematic Annotation form, as shown in [Figure 8-34](#).

*Figure 8-34 Schematic Annotation Form*



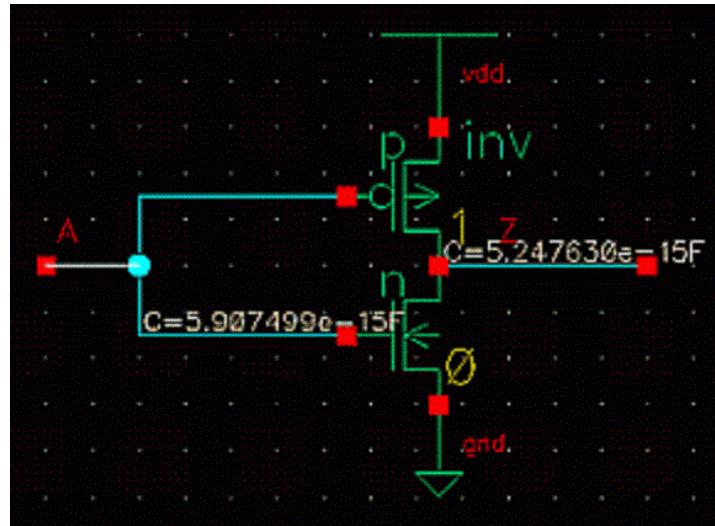
You can save the annotation to a file; the default file name is sch\_annotation.log. In the Parasitic Probing form, the Load button allows you to load saved annotation files.

The annotations are highlight labels instantiated on layer-purpose pairs:

```
("annotate" "drawing8")
```

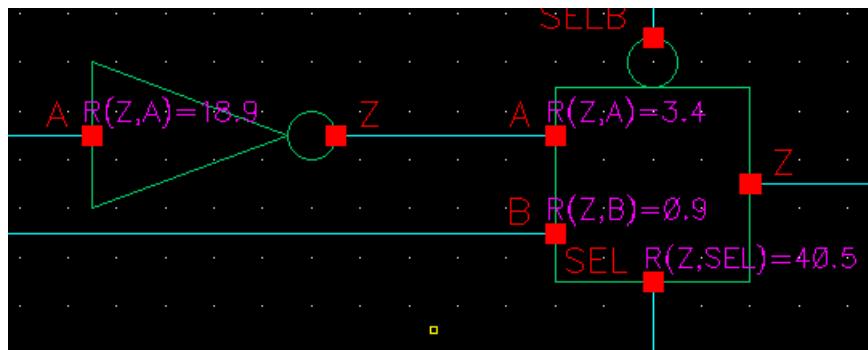
An example of a simple inverter schematic annotated with parasitic capacitance is shown in [Figure 8-35](#).

*Figure 8-35 Example of Parasitic Capacitance Annotation*



An example of parasitic resistance annotation is shown in [Figure 8-36](#). The label takes the form  $R(x,y)$  where  $x$  is the driver pin and  $y$  is the load pin.

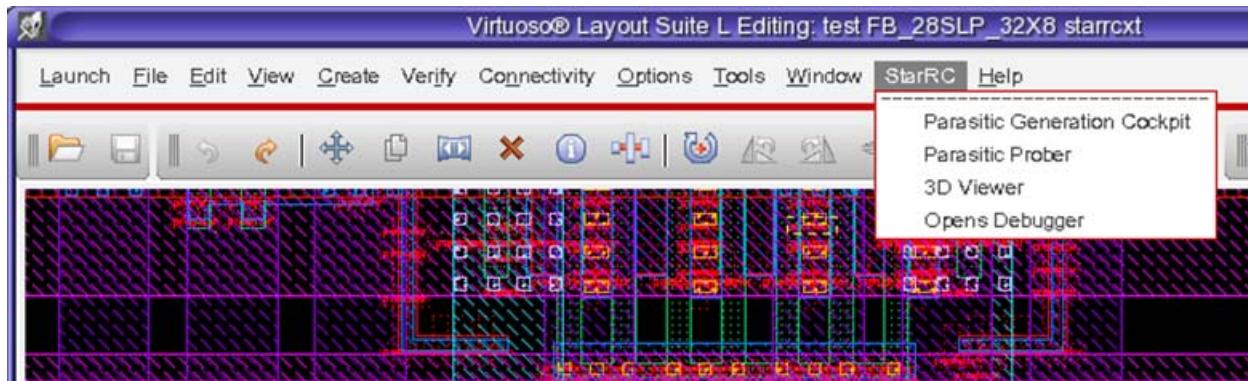
*Figure 8-36 Example of Parasitic Resistance Annotation*



## Opens Debugger

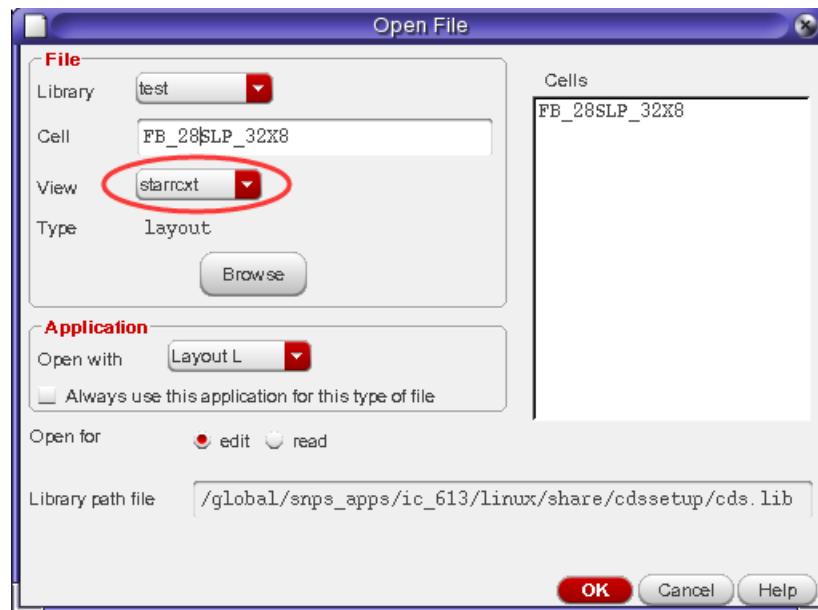
The StarRC tool provides a utility for debugging opens. Select it in the top-level Virtuoso menu bar, as shown in [Figure 8-37](#).

*Figure 8-37 Selecting the Opens Debugger*



Before launching, make sure that the view type is starrcxt, as shown in [Figure 8-38](#). The physical view should be generated with the `REDUCTION: NO` and `EXTRA_GEOMETRY_INFO: NODE RES` commands. Otherwise, an error message is issued.

*Figure 8-38 Selecting the Opens Debugger*

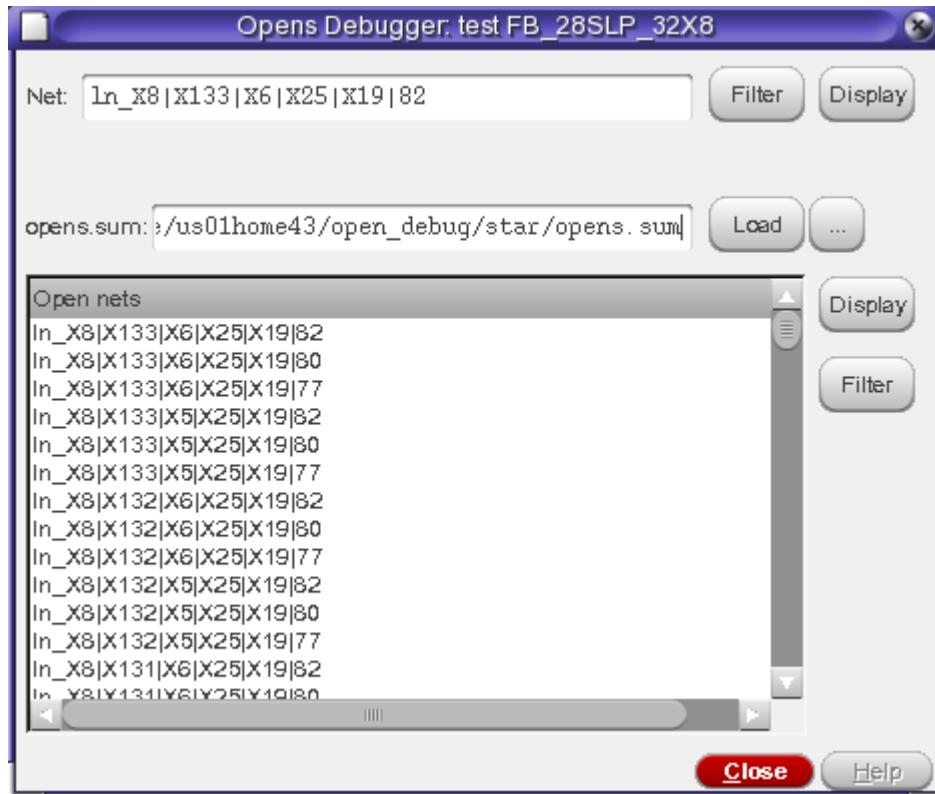


The Opens Debugger form appears, as shown in [Figure 8-39](#).

To choose a net to display, use one of the following methods:

- Type a net name into the Net field at the top of the form.
- Load an opens.sum file in the middle of the form, then select a net from the resulting list.

*Figure 8-39 Opens Debugger Form*



After specifying a net, use the Filter button to the right of the net name to select Resistively Coupled Groups, as shown in [Figure 8-40](#).

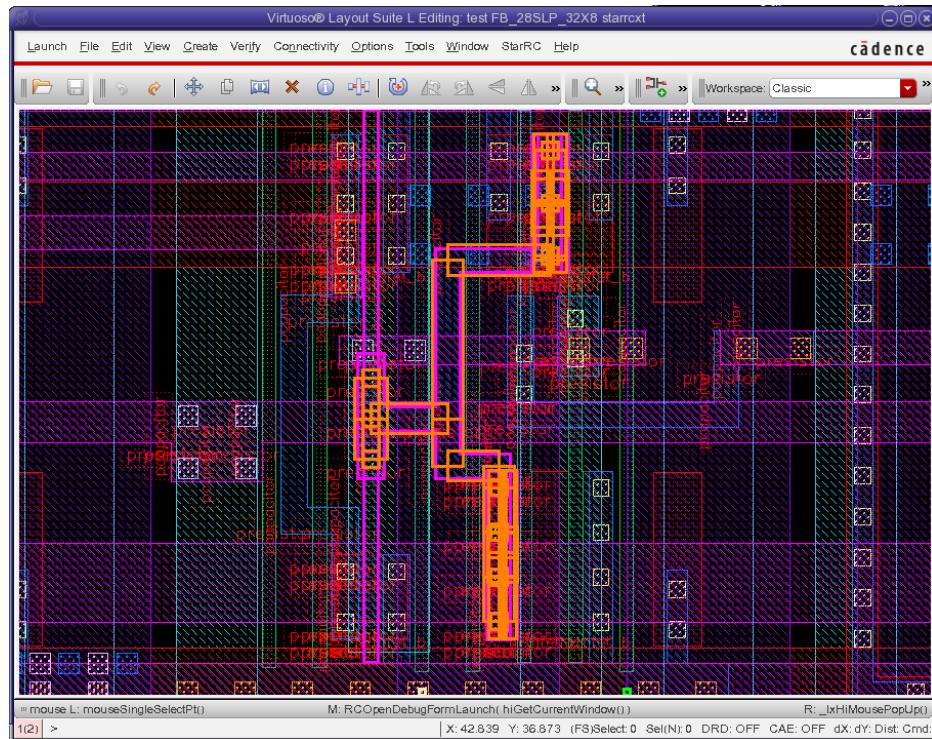
Figure 8-40 Opens Debugger RCG Selection



A resistively connected group (RCG) is a set of nets that have the same name or ID but have no physical connection. The StarRC tool uses the available layout and schematic information to create related groups of nets to assist the debugging process. Select or deselect the RCG options presented in the window to highlight different groups.

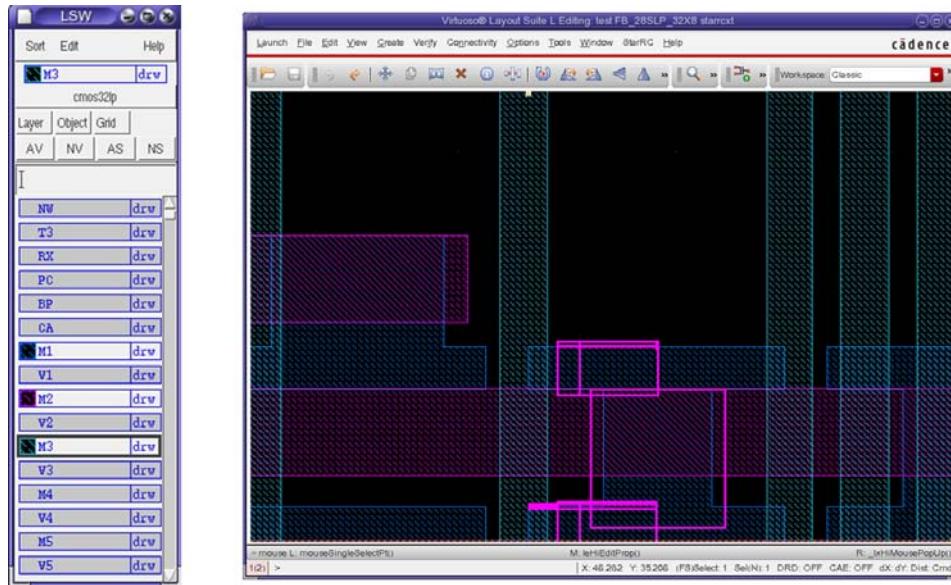
[Figure 8-41](#) shows an example of a highlighted resistively coupled group.

**Figure 8-41** Opens Debugger RCG Display



You can choose the layers to display, as shown in [Figure 8-42](#).

**Figure 8-42** Opens Debugger RCG Display



---

## Virtuoso Integration SKILL Procedures and Related Variables

As you load the rcskill.cxt file, you can use two types of SKILL procedures:

- [GUI Integration with a Custom Interface](#)
  - [Batch Mode Procedures](#)
- 

### GUI Integration with a Custom Interface

You can customize your user interface with the following features:

- `RC_ADD_MENU` environment variable

The `RC_ADD_MENU` environment variable controls the StarRC pulldown menu in the layout and schematic windows.

| Variable                       | Definition                                                         |
|--------------------------------|--------------------------------------------------------------------|
| <code>RC_ADD_MENU = YES</code> | StarRC pulldown menu is shown in the layout and schematic windows  |
| <code>RC_ADD_MENU = NO</code>  | StarRC pulldown menu not shown in the layout and schematic windows |

---

Note:

This variable must be set before invoking `callInitProc` to load the StarRC Virtuoso Integration feature; this setting remains in effect for the entire Virtuoso session.

- `RCPProbeFormLaunch` SKILL procedure

If you set `RC_ADD_MENU=NO` to disable the StarRC pulldown menu, you can use the following SKILL procedure to enable the Parasitic Prober to be launched from your custom user interface:

```
RCPProbeFormLaunch(hiGetCurrentWindow())
```

- `SetupAllInOneForm` SKILL procedure

If you set `RC_ADD_MENU=NO` to disable the StarRC pulldown menu, you can use the following SKILL procedure to enable the Cockpit GUI to be launched from your custom user interface:

```
SetupAllInOneForm(hiGetCurrentWindow())
```

---

## Batch Mode Procedures

You can access batch mode parasitic view generation with the following procedures:

- [RCGenParaViewBatch](#)
- [RCCockpitRun](#)

### RCGenParaViewBatch

The RCGenParaViewBatch procedure accepts only key-based input.

An example of the procedure call is as follows:

```
RCGenParaViewBatch(?dfiiInputLib "AD4FUL"
 ?dfiiInputCell "AD4FUL"
 ?dfiiOutputView "starrc"
 ?cmdFile "star_cmd"
 ?dfiiDeviceMap "/design/dfii_devlist.txt"
 ?propmap_case_sensitive "NO"
 ?dfiiLayerMap "/design/dfii_layermap.txt"
 ?extract t
 ?genPhyPolygn t
 ?genFlyline nil
 ?skip_cell_list ""
 ?carry_sch_prop t
 ?sch_view_name "schematic"
 ?lsf_command nil
 ?separate_starrcviewlog nil
 ?vi_bus_bit ""
)

```

This example includes only the minimum necessary set of keys. Other available keys are summarized in [Table 8-8](#).

An alternative method for using the RCGenParaViewBatch procedure is to write a script procedure to invoke it. The enclosing procedure can fix some of the key values while allowing others to be entered on the command line. An example of this method is as follows:

```
procedure(
 RCGenParaViewBatch(
 @key dfiiInputLib dfiiInputCell dfiiOutputView cmdFile
 dfiiDeviceMap dfiiLayerMap extract blockMode runDir
 (subnodeSize "0.1")
 preprocessCallback postprocessCallback
 spiceCardStripInstpathCallback spiceCardStripPrimitiveCallback
 dfiiOutputLib dfiiOutputCell (genPhyPolygn t) (genFlyline nil)
 (skip_cell_list "") (oa_writer t) (oa_lib_def "") (carry_sch_prop t)
 (sch_view_name "schematic") (lsf_command nil)
 (propmap_case_sensitive nil)
 (port_annotation_view "")
)
)
```

*Table 8-8 Keys Used in the RCGenParaViewBatch Procedure*

| Key name       | Data type | Definition [default, if any]                                                           |
|----------------|-----------|----------------------------------------------------------------------------------------|
| dfiiInputLib   | string    | Input library name                                                                     |
| dfiiInputCell  | string    | Input cell name                                                                        |
| dfiiOutputView | string    | Output view name                                                                       |
| cmdFile        | string    | StarRC command file name                                                               |
| dfiiDeviceMap  | string    | Device mapping file name                                                               |
| dfiiLayerMap   | string    | Layer mapping file name                                                                |
| extract        | Boolean   | Whether to run extraction<br>[Default is <code>t</code> , which means true or yes.]    |
| blockMode      | Boolean   | Whether to run StarRC in block mode<br>[Default is <code>nil</code> , which means no.] |
| runDir         | string    | StarRC run directory<br>[Default is the current working directory.]                    |
| subnodeSize    | float     | Pin or subnode size in physical view<br>[Default is 0.1.]                              |

*Table 8-8 Keys Used in the RCGenParaViewBatch Procedure(Continued)*

| <b>Key name</b>                                       | <b>Data type</b> | <b>Definition [default, if any]</b>                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| preprocessCallback                                    | string           | Callback to call before view generation, for SKILL writer only                                                                                                                                                                                                                                                                                                                                                                                    |
| postprocessCallback                                   | string           | Callback to call after view generation                                                                                                                                                                                                                                                                                                                                                                                                            |
| spiceCardStripInstpath<br>Callback<br>(all one word)  | string           | Whether to strip the SPICE cards for instance paths. Accepts the following mutually exclusive arguments: t (do not strip), nil (strip), or a string procedural call to a SKILL expression for SPICE-card stripping within the instance path.<br><br>[Default is nil, which means to strip.]                                                                                                                                                       |
| spiceCardStripPrimitiveCa<br>llback<br>(all one word) | string           | Whether to strip the SPICE cards for primitive devices. Accepts the following mutually exclusive arguments: t (do not strip), nil (strip), a string procedural call to a SKILL expression for SPICE-card stripping of the primitive at the end of the instance path, or the file name of a cdlinclude file that contains the SUBCKT definitions for which the primitive X card should be stripped.<br><br>[Default is nil, which means to strip.] |
| dfiiOutputLib                                         | string           | Output lib name                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| dfiiOutputCell                                        | string           | Output cell name                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| genPhyPolygn                                          | Boolean          | Whether to generate physical polygons<br><br>[Default is t.]                                                                                                                                                                                                                                                                                                                                                                                      |
| genFlyline                                            | Boolean          | Whether to generate flylines<br><br>[Default is nil.]                                                                                                                                                                                                                                                                                                                                                                                             |
| skip_cell_list                                        | string           | Skip cell mapping file to instantiate skip cells                                                                                                                                                                                                                                                                                                                                                                                                  |
| oa_lib_def                                            | string           | User-specified lib.defs file name<br><br>[Default is "", which uses the Virtuoso lib.defs file]                                                                                                                                                                                                                                                                                                                                                   |
| carry_sch_prop                                        | Boolean          | Pass schematic property or not<br><br>[Default is t.]                                                                                                                                                                                                                                                                                                                                                                                             |

*Table 8-8 Keys Used in the RCGenParaViewBatch Procedure(Continued)*

| <b>Key name</b>                              | <b>Data type</b> | <b>Definition [default, if any]</b>                                                   |
|----------------------------------------------|------------------|---------------------------------------------------------------------------------------|
| sch_view_name                                | string           | View name to pass schematic property<br>[Default is “schematic”.]                     |
| lsf_command                                  | string           | Command to submit an LSF job                                                          |
| propmap_case_sensitive                       | Boolean          | Whether the schematic view property annotation is case sensitive<br>[Default is nil.] |
| port_annotation_view                         | string           | View name to annotate pin info                                                        |
| set_isglobal_to_ground                       | Boolean          | Whether to set isGlobal flag to ground node<br>[Default is t.]                        |
| temperature                                  | string           | Temperatures for TemperatureVX flow, separated by spaces                              |
| skip_lowerlevel_inheritedterm (all one word) | Boolean          | Whether to skip lower level inherited term reading<br>[Default is nil.]               |
| separate_starrcviewlog                       | Boolean          | Whether to generate a separate starrcview.log file for each run<br>[Default is nil.]  |
| vi_bus_bit                                   | string           | Bus bit delimiting characters<br>[Default is “<>”]                                    |
| overwrite_locked_view                        | Boolean          | Whether to allow overwriting of locked parasitic views<br>[Default is nil.]           |

Some of the keys correspond to StarRC commands, as shown in [Table 8-9](#).

*Table 8-9 Correspondence to StarRC Commands*

| Key                                 | StarRC Command                              |
|-------------------------------------|---------------------------------------------|
| dfiiInputLib                        | OA_LIB_NAME (if dfiiOutputLib is not set)   |
| dfiiInputCell                       | OA_CELL_NAME (if dfiiOutputCell is not set) |
| dfiiOutputView                      | OA_VIEW_NAME                                |
| dfiiDeviceMap                       | OA_DEVICE_MAPPING_FILE                      |
| dfiiLayerMap                        | OA_LAYER_MAPPING_FILE                       |
| subnodeSize                         | OA_MARKER_SIZE                              |
| spiceCardStripInstpathCallback nil  | OA_REMOVE_SPICECARD_PREFIX YES              |
| spiceCardStripInstpathCallback t    | OA_REMOVE_SPICECARD_PREFIX NO               |
| spiceCardStripPrimitiveCallback nil | OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX YES    |
| spiceCardStripPrimitiveCallback T   | OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX NO     |
| dfiiOutputLib                       | OA_LIB_NAME                                 |
| dfiiOutputCell                      | OA_CELL_NAME                                |
| genPhyPolygn                        | controlled by OA_LAYER_MAPPING_FILE         |
| skip_cell_list                      | OA_SKIPCELL_MAPPING_FILE                    |
| oa_lib_def                          | OA_LIB_DEF                                  |
| carry_sch_prop                      | controlled by OA_PROPERTY_ANNOTATION_VIEW   |
| sch_view_name                       | OA_PROPERTY_ANNOTATION_VIEW                 |
| propmap_case_sensitive              | OA_PROPMAP_CASE_SENSITIVE                   |
| port_annotation_view                | OA_PORT_ANNOTATION_VIEW                     |
| vi_bus_bit                          | OA_BUS_BIT                                  |
| overwrite_locked_view               | OA_OVERWRITE_LOCKED_VIEW                    |

## RCCockpitRun

You can run a StarRC extraction within the Virtuoso environment by loading a batch file with the SKILL interface or by using the StarRC generation cockpit. In either case, a command file named `cockpit_cmd` is saved in the run directory. You can use this file as the input to an RCCockpitRun batch mode run.

After a successful StarRC run within the Virtuoso environment, several automatically created files remain in the run directory:

- `oa_cmd`  
A StarRC command file created when the output from the StarRC tool is an OpenAccess format netlist.
- `gui_cmd`  
A StarRC command file generated from the “additional options” form in the StarRC cockpit.
- `view_cmd`  
A StarRC command file containing all of the commands necessary for view generation or output netlist generation from the GUI.
- `starrcview.log.time_stamp`  
A file created only during an OpenAccess output extraction run. The file contains a header, the list of StarRC commands used in the run, and the final status (pass or fail). A time stamp with the date and time is automatically appended to the file name.
- `viewlog.block_name.txt`  
A log file showing commands and status. Including the block name in the file name is optional but might be useful if you are running multiple extractions. To include the block name, select the checkbox labeled Separate viewlog.txt By Block in the “Run Cockpit” tab. If you do not select this option, the log file is named `viewlog.txt`.

You can also create an optional command file to perform additional actions. The following is the command that you would execute from the unix command line, where `user_cmd` is the user-created command file:

```
% StarXtract gui_cmd user_cmd view_cmd oa_cmd
```

The order in which the files appear on the command line is important. In this example, the user-created command file overrides the `gui_cmd` file, but the `view_cmd` and `oa_cmd` files override the user-created command file.

An RCCockpitRun Virtuoso Integration job can be rerun if the `cockpit_cmd`, `gui_cmd`, and `view_cmd` files are kept. To replicate a StarRC run within the Virtuoso environment, enter `RCCockpitRun( "cockpit_cmd" )` in the Cadence CIW window.



# 9

## Transistor-Level Extraction

---

This chapter contains information for transistor-level extractions. There is also information for modifying a Calibre rule file for use in the StarRC Calibre Connectivity Interface flow. Included are rules and examples for developing Hercules, IC Validator, and Calibre transistor-level ideal layout extraction runsets, mapping files, and command files.

The chapter contains the following sections:

- [Preparing Hercules Runsets](#)
- [Preparing IC Validator Runsets](#)
- [Preparing Calibre Rule Files](#)
- [Preparing the Mapping File](#)
- [Running the Calibre Query Server for Output to StarRC](#)
- [Preparing the StarRC Command File](#)
- [Interconnect Technology Format File](#)
- [Transistor-Level Extraction Limitations](#)

## Preparing Hercules Runsets

The following information explains how to prepare Hercules runsets. Runset rules, required commands, and a runset example are included.

EQUATE statements do not need to be included in the original Hercules runset; Hercules-generated EQUATE statements in the lvsflow/lvs\_include.ev file are sufficient.

Command-line overrides can be used in the Hercules layout extraction and LVS compare.

Runset layer: Any layer specified in the CONNECT section of the Hercules runset.

Physical layer: Any layer specified as a CONDUCTOR or VIA in the ITF file.

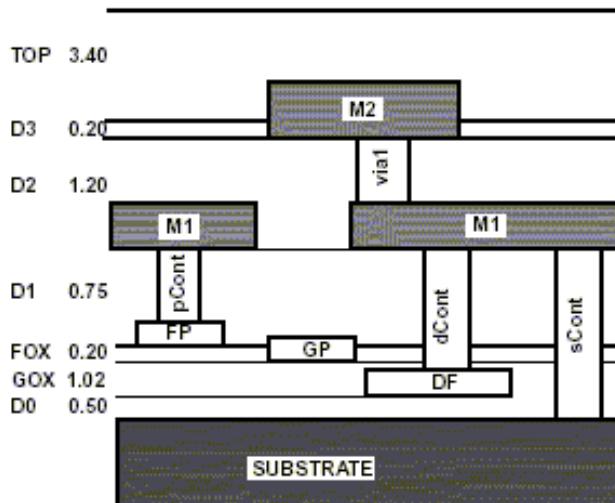
## Runset Rules

Follow these rules when preparing the Hercules runset:

- A runset layer via can connect only two physical layers.

It is important to separate metal1 to diffusion contacts (dCont) and metal1 to poly contacts (pCont), because these two contacts connect metal1 to two different physical layers at different levels in the ITF file. In SOI and STI processes where diffusion and substrate exist on two different physical levels in the ITF file, it might also be necessary to distinguish metal1 to substrate contacts (sCont). See [Figure 9-1](#). Metal1 to substrate contacts (where diffusion and substrate exist on separate physical levels in the ITF file) are shown.

*Figure 9-1 Separate Metal1 to Diffusion Contacts*



For most runsets, following the correct convention is simple. In general, the runset should have specific contacts for connecting the following physical layers:

- Metal 2 and Metal 1
- Metal 1 and poly
- Metal 1 and diffusion
- Metal 1 and substrate

**Incorrect:**

```
CONNECT m1 poly BY cont
CONNECT m1 nsd psd BY cont
```

**Correct:**

```
BOOLEAN cont AND poly {} TEMP = polyCont
BOOLEAN cont NOT polyCont {} TEMP = subCont

CONNECT m1 poly BY polyCont
CONNECT m1 nsd psd BY subCont
```

- Device layers overlap with routing layers should be eliminated by using the NOT operation.

Removing portions of routing layers that overlap device layers allows StarRC to correctly ignore device capacitances. This is required for MOS gate, source, and drain terminals as well as for CAP terminals.

**Incorrect:**

```
BOOLEAN poly AND ndiff {} TEMP = ngate
BOOLEAN ndiff NOT poly {} TEMP = nsd
NMOS n ngate nsd nsd SUBSTRATE {} TEMP = ndev

BOOLEAN m2 AND cap_recog {} TEMP = cap_top
BOOLEAN m1 AND cap_recog {} TEMP = cap_bot
BOOLEAN cap_top AND cap_bot {} TEMP = cap_dev
CAP m1m2cap cap_dev cap_top cap_bot {} TEMP = cdev

CONNECT poly BY ngate
CONNECT cap_top BY m2
CONNECT cap_bot BY m1
```

StarRC does not ignore the gate cap correctly, and the designed capacitance is double-counted in this example. Other devices might not have these issues.

Correct:

```

BOOLEAN poly AND ndiff {} TEMP = ngate
BOOLEAN ndiff NOT poly {} TEMP = nsd
BOOLEAN poly NOT ngate {} TEMP = poly
NMOS n ngate nsd nsd SUBSTRATE {} TEMP = ndev

BOOLEAN m2 AND cap_recog {} TEMP = cap_top
BOOLEAN m1 AND cap_recog {} TEMP = cap_bot
BOOLEAN cap_top AND cap_bot {} TEMP = cap_dev
BOOLEAN m2 NOT cap_top {} TEMP = m2
BOOLEAN m1 NOT cap_bot {} TEMP = m1
CAP m1m2cap cap_dev cap_top cap_bot {} TEMP = cdev

CONNECT poly BY ngate
CONNECT cap_top BY m2
CONNECT cap_bot BY m1

```

The Boolean NOT for the gate and field poly is required for planar processes (where gate poly and field poly layers are both mapped to a single POLY layer in the nxtgrd file) as well as nonplanar processes (where gate poly and field poly layers are mapped to separate covertical layers in the nxtgrd file), because IGNORE\_CAPACITANCE works in both cases.

- A physical layer cannot directly connect to another physical layer without using a via, unless the two physical layers are covertical.

In [Figure 9-1](#), metal1 connects to metal2 by via1. However, there is no need for a contact between physical layers FP and GP because they overlap on the vertical profile. These two conductors are called covertical.

In the previous runset example, connecting poly and ngate layers is allowed because the two runset layers map to the covertical physical layers FP and GP. Likewise, connecting cap\_top to m2 is allowed because both are mapped to the same physical layer M2. The same also applies for cap\_bot and m1.

## Required Commands

This section describes commands that are required in your Hercules runset to make the StarRC tool run correctly.

### **WRITE\_EXTRACT\_VIEW**

```

WRITE_EXTRACT_VIEW {
 LIBRARY_NAME = LIB_NAME
 LIBRARY_PATH = .
}

```

StarRC uses Milkyway XTR views instead of the CHECK\_POINT for input of the layout database directory which was used for StarRC. After Hercules is finished, the XTR view is

written into the *LIB\_NAME* directory, which is placed in your Hercules working directory. In writing this XTR view, Hercules creates its own Milkyway technology file called hx2mw.tf.

The hx2mw.tf file has all the layer information of the XTR view, which is different from that in the ASSIGN section of the runset. This is because the Milkyway XTR view database contains only derived runset layers that contribute to connectivity or device formation. It does not contain ASSIGN section layers unless those layers are directly used in CONNECT statements or device extraction commands.

If the design originates from a Milkyway library, you should not write the XTR view into the existing library. The technology file of the original Milkyway database is different from the one made by Hercules. The specified *LIBRARY\_NAME* option should be a unique library name into which Hercules dumps the XTR view results.

You can open the XTR view from any Milkyway viewer.

## TECHNOLOGY\_OPTIONS

```
TECHNOLOGY_OPTIONS {
 INCLUDE TECHNOLOGY_DEFAULTS = FALSE
 ALLOW_EXPLODE_WITH_TEXT = TRUE
 EXPLODE_AREFS = TRUE /* Default: FALSE */
 EXPLODE_1XN_AREFS = TRUE
 EXPLODE_AUTO_FLATTEN_LIMIT=100
 EXPLODE_CELL_SIZE_PERCENT=70
 EXPLODE_CELL_SIZE_PERCENT_OF_TOP=70
 EXPLODE_BIG_SPARSE_CELL=TRUE
 POST_VCELL_EXPLODE_CELL_SIZE <= 10
 VIA_AUTO_EXPLODE=TRUE
 SUBLEAF_AUTO_EXPLODE=6
 CELL_SIZE_AUTO_EXPLODE <= 10
 EXPLODE_DATA_CELL_LIMIT = 1
 POST_VCELL_EXPLODE_DATA_CELL_LIMIT = 12
 EXPLODE_HOLDING_CELL_LIMIT = 1
 EXPLODE_PLACEMENT_LIMIT = 1
 VCELL_PASS {
 MIN_CELL_OVERLAP = 60
 RATIO=10000.000
 MIN_COUNT = 40
 MIN_LEAF = 40
 STYLE = EXPLODE
 }
}
```

You can use TECHNOLOGY\_OPTIONS to optimize the hierarchy in a Hercules run, thereby speeding up the run and reducing memory usage. In addition, these options optimize the output hierarchy in the XTR library. This enhances StarRC performance significantly if the design is very hierarchical.

However, in XREF flows, some of the defaults, such as VCELL\_PASS {STYLE = PAIRS} and VCELL\_PASS {STYLE = SETS}, can cause unwanted explodes of EQUIV points and

possible mismatches. As a result, you should not use all of the defaults for an LVS-enabled Hercules run. This is because XREF: YES cross-references down to the EQUIV points and all the EQUIV points should be preserved.

You should therefore include the TECHNOLOGY\_OPTIONS section in your Hercules runset, rather than using the defaults. You can use the Hercules command-line option `-rcxt` to create the section. If you use the `-rcxt` option, make sure that your runset does not have any TECHNOLOGY\_OPTIONS. If the `INCLUDE_TECHNOLOGY_DEFAULTS=FALSE` statement is set in the Hercules runset, the Hercules `-rcxt` option does not work.

## NETLIST

```
NETLIST { }
```

NETLIST does not do anything in generating XTR, but it is required for Hercules LVS.

## Compare and Evaccess Directories

These directories are generated by Hercules. In XREF runs, the StarRC tool gets the cross-reference information from these directories. Do not remove them before the StarRC run is complete. Their locations are detected automatically if they have not been moved since the Hercules run. Otherwise, use the `EVACCESS_DIRECTORY` and `COMPARE_DIRECTORY` commands in the StarRC command file to specify paths to the directories.

## Commands That Are No Longer Required

The following commands are not required in your runset:

- `CHECK_POINT` – The `CHECK_POINT` directory is not used.
- `SAVE_NETLIST_DATABASE` – The `WRITE_EXTRACT_VIEW` command does not get any information from the `SAVE_NETLIST_DATABASE` command.
- `SPICE` – The StarRC tool does not look at the *block.sp* file for cross-referencing.
- `GRAPHICS_NETLIST` – Because graphical output exists in the XTR format, you do not need to create another layout database in LTL format.

## Runset Example

The following is a typical Hercules runset for StarRC transistor-level extraction.

```
HEADER {
 LAYOUT_PATH =
 INLIB = ADD4_CUSTOM.GDS
 OUTLIB = EV_OUT
 FORMAT = GDSII
 BLOCK = add4
 SCHEMATIC = ADD4.sch
}
```

```

OPTIONS {
 SCHEMATIC_GLOBAL = {VDD GND}
 SCHEMATIC_GROUND = {GND}
 SCHEMATIC_POWER = {VDD}
 LAYOUT_GLOBAL = {VDD GND}
 LAYOUT_POWER = {VDD}
 LAYOUT_GROUND = {GND}
 EXPLODE = {VIA *con *tie DEV_*}
 NET_PREFIX = N_
 EDTEXT = ADD4.text
}
TEXT_OPTIONS {
 ATTACH_TEXT = ALL
 CONNECT_BY_NAME = MIXED_MODE
}

ASSIGN {
 Ndiff (1)
 Pdiff (2)
 Nwell (3)
 Poly (5)
 Cont (6)
 Metal1 (8) TEXT (28)
 Vial (9)
 Metal2 (10) TEXT (30)
 PnAnode (51)
 PnCathode (52)
 NpCathode (61)
 NpAnode (62)
 NpnCollector (71)
 NpnBase (72)
 NpnEmitter (73)
 Cap1 (11)
 Cap2 (12)
 PolyRes (20)
}
TEXT_POLYGON Metal2.text {
 SIZE = 0.01
 CELL_LIST = {add4}
 TEXT_LIST = {* !*VDD* !*VSS* }
} TEMP = Metal2_Mark

/** NPN BJT */
BOOLEAN Cont AND NpnEmitter {} TEMP = NpnEmitterCont
BOOLEAN Cont NOT NpnEmitterCont {} TEMP = Cont
BOOLEAN Cont AND NpnBase {} TEMP = NpnBaseCont
BOOLEAN Cont NOT NpnBaseCont {} TEMP = Cont
BOOLEAN Cont AND NpnCollector {} TEMP = NpnCollectorCont
BOOLEAN Cont NOT NpnCollectorCont {} TEMP = Cont
/** Without these, all the three terminals of this Npn device will be
shorted and the device will be filtered out. ***/

```

```

NPN NpnDev NpnEmitter NpnBase NpnCollector SUBSTRATE {} TEMP = NpnDev

/*** PN DIODE ***/
BOOLEAN PnCathode NOT PnAnode {} TEMP = PnCathodeTerm
BOOLEAN PnAnode AND PnCathode {} TEMP = PnDevice
COPY PnAnode {} TEMP = PnAnodeTerm

DIODE PnDev PnDevice PnAnodeTerm PnCathodeTerm {
 DIODE_TYPE = PN
 DIODE_RECOGNITION_LAYER_USED = TRUE
} TEMP = PnDev

/*** MOS ***/
BOOLEAN Pdiff AND Nwell {} TEMP = pdev
BOOLEAN Ndifff NOT Nwell {} TEMP = ndev
BOOLEAN Pdiff NOT Nwell {} TEMP = subtie
BOOLEAN Ndifff AND Nwell {} TEMP = welltie
BOOLEAN Poly AND ndev {} TEMP = ngate
BOOLEAN Poly AND pdev {} TEMP = pgate
BOOLEAN ndev NOT ngate {} TEMP = nsd
BOOLEAN pdev NOT pgate {} TEMP = psd
BOOLEAN Poly NOT ngate {} TEMP = Poly
BOOLEAN Poly NOT pgate {} TEMP = Poly
/*** These 2 Boolean expressions are mandatory for StarXtract. ***/

NMOS n ngate nsd nsd SUBSTRATE {} TEMP = ndevice
PMOS p pgate psd psd Nwell {} TEMP = pdevice

/*** Contact separation ***/
BOOLEAN Cont AND Poly {} TEMP = PolyCont
BOOLEAN Cont NOT PolyCont {} TEMP = SubCont
/*** Contact separation is a must for StarXtract. ***/

/*** CAPACITOR ***/
BOOLEAN Cap1 AND Metall1 {} TEMP = Cap1
BOOLEAN Cap2 AND Poly {} TEMP = Cap2
BOOLEAN Metall1 NOT Cap1 {} TEMP = Metall1
BOOLEAN Poly NOT Cap2 {} TEMP = Poly
BOOLEAN Cap1 AND Cap2 {} TEMP = CapRec

CAP CapDev CapRec Cap1 Cap2 {
 EV_AREA_CAPVAL = 1.0e-15;
 CAP_RECOGNITION_LAYER_USED = TRUE;
} TEMP = CapDev

/*** Poly RESISTOR ***/
BOOLEAN Poly AND PolyRes {} TEMP = pres_body
BOOLEAN Poly NOT pres_body {} TEMP = field_poly
/*** Routing layers can be used as resistor terminals. ***/

```

```

RES ResDev pres_body field_poly field_poly {
 EV_RESVAL = 3.5;
} TEMP = ResDev

CONNECT {
 Metal2 Metall BY Vial
 Metall BY Cap1
 Metall field_poly BY PolyCont
 field_poly BY Cap2
 Metall nsd psd welltie subtie BY SubCont
 Metall PnCathodeTerm BY SubCont
 Metall PnAnodeTerm BY SubCont
 Metall NpnCollector BY NpnCollectorCont
 Metall NpnBase BY NpnBaseCont
 Metall NpnEmitter BY NpnEmitterCont
 ngate pgate BY field_poly
 Nwell BY welltie
 SUBSTRATE BY subtie
 Metal2_Mark BY Metal2
}

TEXT {
 Metall1 BY Metall1.text
 Metal2 BY Metal2.text
}

NETLIST {}
WRITE_EXTRACT_VIEW {
 LIBRARY_NAME = ADDER
 LIBRARY_PATH = .
}
COMPARE {}

```

## Power Port Netlist Generation

By default, StarRC netlists only ports that are connected to extracted nets. Nets specified in the `POWER_NETS` command are not extracted by default and therefore are not netlisted. There are two ways to have these power ports netlisted:

- Exclude the power nets from the `POWER_NETS` command so that these nets are not identified by StarRC as power nets and extracted.
- Include a SPICE file with the StarRC `SPICE_SUBCKT_FILE` command. StarRC uses this SPICE file to duplicate the port list and ordering. This file can be generated from Hercules by using the SPICE command in the Hercules runset. After the SPICE file is generated and the power ports are netlisted in this .sp file, it can be specified in the StarRC command file as the following:

```
SPICE_SUBCKT_FILE; .sp
```

The power ports are then netlisted in the parasitic netlist as they are in the .sp file, even though they might still be included in the `POWER_NETS` command.

---

## Marker Generation in Hercules

The following sections show how you can generate text or layer markers.

### Example of Text-Based Markers

The Hercules TEXT\_POLYGON command is used for this flow; it generates a shape surrounding selected text in the layout. It should be very small, because it is included in the CONNECT sequence and can create shorts.

In the Hercules runset (runset.ev):

```
...
ASSIGN metall1 (3;0) text(3;4)
...
TEXT_POLYGON metall1.text {
 text_list = { * }
 cell_list = { TOP }
 size = 0.1
} temp = metall1_pio
TEXT_POLYGON metall1.text {
 text_list = { * }
 cell_list = { * !TOP }
 size = 0.1
} temp = metall1_pin

...
SELECT metall1 INTERACT metall1_pin { CELL_LEVEL } temp =
metall1_tmp
BOOLEAN metall1_tmp OR metall1_pin { } temp = metall1_pin
...
CONNECT metall1 BY metall1_pio
CONNECT metall1 BY metall1_pin
...
TEXT metall1 BY metall1
TEXT metall1_pio BY metall1
TEXT metall1_pin BY metall1
```

In the StarRC mapping file:

```
conducting_layers
 metall1
...
marker_layers
 metall1_pio
 metall1_pin
```

## Example of ID-Layer Markers

The simplest approach to manual marker generation, this technique uses a pre-existing input layer (ASSIGN) to identify marker shapes.

In the Hercules runset (runset.ev):

```
...
ASSIGN metall (3;0) text (3;4)
ASSIGN metall_pin (33;0)
ASSIGN PAD (32;0)
...
BOOLEAN metall AND PAD { } temp = metall_pio
BOOLEAN metall_pio OR metall_pin = metall_markers
...
CONNECT metall by metall_markers
TEXT metall by metall
TEXT metall_markers by metall
```

In the StarRC mapping file:

```
conducting_layers
 metall
...
marker_layers
 metall_markers
```

## Example of Connection-Based Markers

This approach uses hierarchical geometric interactions to identify exact connection points into subcells. This is valid only for an all-Hercules flow. This technique uses either `BOOLEAN` or `CONNECTION_POINTS` commands to generate an output layer that represents the hierarchical interactions of conducting layers. In this example, the `REMOVE_OVERLAP` command is used to eliminate redundancy in the layout for overlay ports. For more information, see the *Hercules LVS User Guide*.

In the Hercules runset (runset.ev):

```
ASSIGN cont (2;0)
ASSIGN metall (3;0) text (3;4)
ASSIGN vial (4;0)
...
REMOVE_OVERLAP metall { } temp = metall_tmp
CONNECTION_POINTS metall_tmp metall_tmp cont vial {} temp=metall_pin
CONNECT metall by metall_pin
TEXT metall by metall
TEXT metall_pin by metall
```

In the StarRC mapping file:

```
conducting_layers
 metall1
...
marker_layers
 metall1_pin
```

## Preparing IC Validator Runsets

The following section explains the rules and the required functions in the IC Validator runsets.

### Runset Rules

When creating an IC Validator runset, you must follow certain rules in data manipulation.

The following layer definitions are accepted by IC Validator:

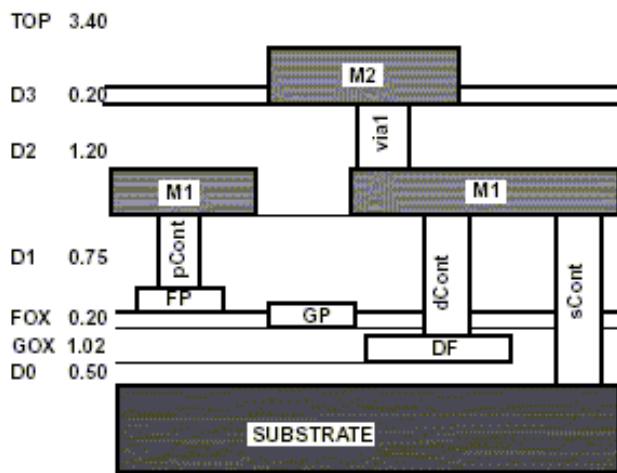
- Runset layer – Any layer specified in the connect section of the IC Validator runset.
- Physical layer – Any layer specified as a CONDUCTOR or VIA in the ITF file.

#### A runset layer via can connect only two physical layers

```
input_cdb = connect(
 connect_items = {
 {{M1, poly}, cont},
 {{M1, nsd, psd}, cont}
 ...
};
```

Separating metal1 to diffusion contacts (dCont) and metal1 to poly contacts (pCont) is necessary because these two contacts connect metal1 to two different physical layers at different levels in the ITF file. In SOI or STI processes where diffusion and substrate exist on two different physical levels in the ITF file, distinguishing the metal1 to substrate contacts (sCont) might also be necessary. Metal1 to substrate contacts, where diffusion and substrate exist on separate physical levels in the ITF file, are shown in [Figure 9-2](#).

*Figure 9-2 Separate Metal1 to Diffusion Contacts*



For most runsets, following the correct convention is straightforward. In general, the runset should have specific contacts for connecting the following physical layers:

```
...
metal2 - metal1
metal1 - poly
metal1 - diffusion and SUBSTRATE

CORRECT:
polyCont = cont and poly;
subCont = cont not polyCont;
input_cdb = connect(
 connect_items = {
 {{m1, poly}, polyCont},
 {{m1, nsd, psd}, subCont}
 }
);
...
```

### All device runset layers should be negated with the routing runset layers

Removing portions of routing layers that overlap device layers allows StarRC to ignore device capacitances correctly, and is necessary for MOS gate, source, and drain terminals as well as for capacitance terminals.

**Incorrect:**

```

ngate = poly and ndiff;
nsd = ndiff not poly;
...
nmos(my_devices, "n", nsd, ngate, nsd, {{psub}});
...
cap_top = m2 and cap_recog;
cap_bot = m1 not cap_bot;
cap_dev = cap_top and cap_bot;
...
capacitor(my_devices, "m1m2cap", cap_dev, cap_top, cap_term,
area_capval=1e-15);
.....
input_cdb = connect(
 connect_items = {
 {{poly}}, ngate},
 {{cap_top}}, m2},
 {{cap_bot}}, m1},
 ...
);

```

StarRC does not ignore the gate capacitance correctly and the designed capacitance is double-counted in this example. Other devices might not have these issues.

**Correct:**

```

naget = poly and ndiff;
nsd = ndiff not poly;
poly = poly not ngate;
my_devices = init_device_matrix(netlist_cdb);
nmos(my_devices, "n", nsd, ngate, nsd, {{SUBSTRATE}});
cap_top = m2 and cap_recog;
cap_bot = m1 and cap_recog;
m2 = m2 not cap_top;
m1 = m1 not cap_bot;
capacitor(my_devices, "m1m2cap", cap_dev, cap_top, cap_bot,
area_capval=1e-15);
input_cdb = connect(
 connect_items = {
 {{poly}}, ngate},
 {{cap_top}}, m2},
 {{cap_bot}}, m1},
 ...
);

```

The `not()` function performed on the gate and field poly is required for the planar and nonplanar processes because the `IGNORE_CAPACITANCE` capability behaves the same in both cases. A planar process implies that the gate poly and the field poly layers are both

mapped to a single POLY layer, while a nonplanar process implies that the gate poly and the field poly are mapped to separate covirtual layers in the nxtgrd file.

**A physical layer cannot directly connect to another physical layer without using a via, unless the two physical layers are covirtual**

In [Figure 9-1](#) metal1 connects to metal2 by via1. However, placing a contact between physical layers FP and GP is not necessary because they overlap on the vertical profile. These two conductors are called covirtual. In the previous runset example, connecting poly and ngate layers is allowed because the two runset layers map to the covirtual physical layers FP and GP. Likewise, connecting cap\_top to m2 is allowed because both are mapped to the same physical layer M2. The same also applies for cap\_bot and m1.

## Required Functions

Add the following required functions to your IC Validator runset script to ensure that StarRC runs correctly:

```
pex_generate_results(
 pex_matrix = pex_matrix,
 device_extraction_matrix = my_devices,
 device_db = device_db,
 layout_database = mw_handle,
 pex_process_map_file = pex_process_handle,
 pex_runset_report_file = pex_report_handle
);
```

In addition, a tag name can be used to assign a layer name to runset layers. In [Example 9-1](#) “SUBSTRATE” and “via2” are tag names shown in the runset report file and extract view layer name.

*Example 9-1 Tag Name Use in IC Validator Run Script*

```
pex_conducting_layer_map(pex_matrix, psub, "SUBSTRATE");
pex_via_layer_map(pex_matrix, V2, "via2");
```

StarRC uses the IC Validator runset report file to obtain all IC Validator LVS results and perform parasitic extraction. The key argument is `pex_runset_report_file` in the `pex_generate_result()` function. The default for `pex_runset_report_file` is `./pex_runset_report`.

After an IC Validator run is completed, the XTR view is written into the LIB\_NAME directory under the default directory path, `$working_directory/XTROUT`.

To change the defaults for variables related to IC Validator, examine the \$ICV\_HOME\_DIR/include/rcxt\_public.rh file.

The content of this file is similar to the following example:

```
pex_generate_results : published function (
 pex_matrix : pex_layer_matrix,
 device_extraction_matrix : device_matrix,
 device_db : device_database,
 layout_database : in_out milkyway_library_handle,
 pex_process_map_file : pex_process_map_file_handle,
 pex_runset_report_file : pex_runset_report_file_handle,
 pex_lpp_map_file : pex_lpp_map_file_handle =
 NULL_LPP_MAP_FILE,
 pex_tagname_required : boolean = true,
 precision : integer = 3
) returning void;
```

## Hierarchy Options

To specify hierarchical options, use the `hierarchy_options()` function in IC Validator. This is the same as the `technology_options` function in Hercules. The `hierarchy_options()` option controls the hierarchy optimization.

Note:

When using the `hierarchy_options` option, some of the vcell options create new cells that do not exist in the schematic and hinder the XREF:YES or XREF:COMPLETE flows in StarRC. In this case, set the `iterate_max` to 0 in both pairs and sets of stages.

## Hierarchy Options Syntax

The syntax used to specify hierarchy options is shown in the following example:

```
hierarchy_options (
 pairs = {{pair_cells = {"string", ...},
 iterate_max = integer,
 explode_into_vcell = ON | OFF | AUTO,
 x_pairing = true | false,
 y_pairing = true | false},
 ... }, //optional
 sets = {{base_cells = {"string", ...},
 programming_cells = {"string", ...},
 iterate_max = integer,
 explode_into_vcell = ON | OFF | AUTO,
 flatten_sets = true | false,
 min_cell_overlap = integer,
 share_base_cells = true | false},
 ...}, //optional
);
);
```

---

## Runset Example

The following example is a typical IC Validator runset for a transistor-level extraction.

```
#include "primeyield.rh"

library(
 format = GDSII,
 cell = "add4",
 library_name = "ADD4.GDS"
);

schematic_netlist_db = schematic(
 schematic_file = {"ADD4.sp", SPICE}
);

#include "equiv.rs"

NDIFF = assign({{1}});
PDIFF = assign({{2}});
NWELL = assign({{3}});
POLY = assign({{5}});
POLY_TEXT = assign_text({{25}});
CONT = assign({{6}});
M1 = assign({{8}});
M1_TEXT = assign_text({{28}});
V1 = assign({{9}});
M2 = assign({{10}});
M2_TEXT = assign_text({{30}});
V2 = assign({{44}});
M3 = assign({{45}});
M3_TEXT = assign_text({{32}});

sub1 = cell_extent(
 cell_list = {"*"}
);

psub = sub1 not NWELL;
pactive = PDIFF and NWELL;
nactive = NDIFF not NWELL;
subtie = PDIFF not NWELL;
welltie = NDIFF and NWELL;
pactive = PDIFF not subtie;
nactive = NDIFF not welltie;
ngate = POLY and nactive;
pgate = POLY and pactive;
fpoly = POLY not (ngate or pgate);
nsd = nactive not ngate;
psd = pactive not pgate;

input_cdb = connect(
 connect_items = {
```

```

 { {M3, M2}, V2 },
 { {M2, M1}, V1 },
 { {M1, fpoly}, poly_con },
 { {ngate, pgate}, fpoly },
 { {M1, nsd, psd}, diff_con },
 { {M1, welltie, subtie}, diff_con },
 { {NWELL}, welltie },
 { {psub}, subtie }
 }
);

netlist_cdb = text_net(
 connect_sequence = input_cdb,
 text_layer_items = {
 { M3, M3_TEXT },
 { M2, M2_TEXT },
 { M1, M1_TEXT },
 { fpoly, POLY_TEXT }
 },
 attach_text = ALL
);

my_devices = init_device_matrix(netlist_cdb);

nmos(my_devices, "n", nsd, ngate, nsd, {{psub}});
pmos(my_devices, "p", psd, pgate, psd, {{NWELL}});

device_db = extract_devices(my_devices);

layout_netlist_db = netlist(device_db);

compare_settings = init_compare_matrix();

compare(compare_settings, schematic_netlist_db, layout_netlist_db,
 generate_equiv = {FULL_NAME}, write_equiv_netlists=ALL);

pex_matrix = init_pex_layer_matrix(my_devices);
pex_conducting_layer_map(pex_matrix, M3, "metal3");
pex_conducting_layer_map(pex_matrix, M2, "metal2");
pex_conducting_layer_map(pex_matrix, M1, "metal1");
pex_conducting_layer_map(pex_matrix, fpoly, "poly");
pex_conducting_layer_map(pex_matrix, ngate, "poly");
pex_conducting_layer_map(pex_matrix, pgate, "poly");
pex_conducting_layer_map(pex_matrix, nsd, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, psd, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, welltie, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, subtie, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, NWELL, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, psub, "SUBSTRATE");

pex_via_layer_map(pex_matrix, V2, "via2");
pex_via_layer_map(pex_matrix, V1, "vial");
pex_via_layer_map(pex_matrix, poly_con, "polyCont");

```

```

pex_via_layer_map(pex_matrix, diff_con, "diffCont");

pex_process_handle = pex_process_map_file();
pex_report_handle = pex_runset_report_file();
mw_handle = milkyway_library("XTROUT");

pex_generate_results(
 pex_matrix = pex_matrix,
 device_extraction_matrix = my_devices,
 device_db = device_db,
 layout_database = mw_handle,
 pex_process_map_file = pex_process_handle,
 pex_runset_report_file = pex_report_handle
);

```

## IC Validator Marker Generation

The following sections describe text and layer markers in the IC Validator tool.

### Text-Based Marker Layers

The `text_origin()` function is used for the IC Validator flow. It generates a shape surrounding the selected text in the layout. The shape should be very small, because it is included in the CONNECT sequence and can create shorts.

In the IC Validator runset (runset.rs):

```

metal1 = assign({{6}});
metal1_text = assign_text({{20}, {30}});

```

In the previous example, `markers = text_origin(metal1_text, shape_size=0.01, cells = {"XT_DEVICES"}, text = {"*", "VDD", "VSS"}); m1_markers = markers interacting metal1.`

```

input_cdb = connect(
 connect_items = {
 ...
 {{metal1}}, m1_markers},
 ...
);
netlist_cdb = text_net(
 connect_sequence = input_cdb,
 text_layer_items = {
 { metal1, metal1_text },
 ...
 },
 attach_text = ALL
);
...

```

In the StarRC mapping file:

```
conducting_layers
metall
...
marker_layers
m1_markers
```

### Example of ID-Layer Markers

The simplest approach to manual marker generation, this technique uses a pre-existing input layer (`assign()`) to identify marker shapes.

The following is defined In the IC Validator runset:

```
metall = assign({{6}});
metall_text = assign_text({{20}, {30}});
...
metall_pio = metall and PAD
metall_markers = metall_pio or metall_pin
input_cdb = connect(
 connect_items = {
 ...
 {{metall}}, m1_markers},
 ...
 }
);
netlist_cdb = text_net(
 connect_sequence = input_cdb,
 text_layer_items = {
 {metall, metall_text},
 },
 attach_text = ALL
);
```

In the StarRC mapping file:

```
conducting_layers
metall
...
marker_layers
metall_markers
```

---

## Multifinger Device Support

Multifinger devices have multiple gate, source, and drain terminals. However, in the XTR view of the Milkyway database, a device can only have a single gate, source, or drain terminal. The XTR view writer in the IC Validator tool stores the extra terminal information inside the terminal properties, which is parsed by the StarRC tool.

When a device function is declared in the IC Validator tool with the option settings `recognition_layer=true` and `merge_parallel=true`, the individual coordinates of each polygon in terminals with multiple polygons are reported.

The StarRC tool reads each set of coordinates from the XTR view for correct resistance network calculation.

---

## Preparing Calibre Rule Files

The following sections describe the preparation of a rule file for the Calibre Connectivity Interface flow:

- [Rule File Creation](#)
- [Required Commands](#)
- [Support for Calibre Preprocessor Directives and Include Statements](#)
- [Rule File Example](#)

---

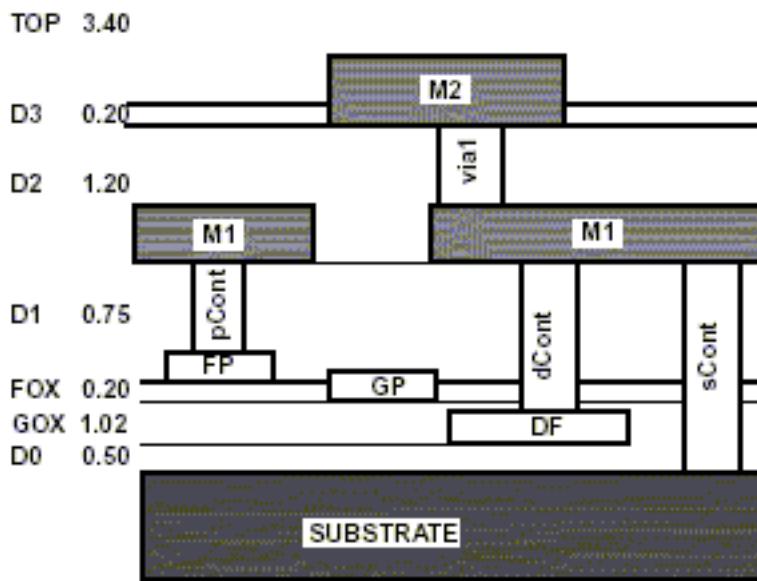
## Rule File Creation

Use the following guidelines when creating a Calibre runset:

- A runset layer can connect only two physical layers.

In the Calibre rule file, it is important to separate the metal1 to diffusion contact from the metal1 to poly contact. LVS conventions typically allow both contacts to exist on the same rule file layer (“cont” in this example). However, physically these are separate contacts, so it is important for them to be separated for the StarRC extraction engine. See [Figure 9-3](#). Moreover, these contacts also have different resistivity.

*Figure 9-3 Separate Calibre Runset Layers*



Incorrect:

```
CONNECT m1 poly BY cont
CONNECT m1 nsd psd BY cont
```

Correct:

```
polyCont = cont AND poly
subCont = cont NOT poly
```

```
CONNECT m1 poly BY polyCont
CONNECT m1 nsd psd BY subCont
```

- Device layers overlap with routing layers should be eliminated by using the NOT operation.

It is important to remove from routing layers any overlap that exists with device terminal layers. The StarRC IGNORE\_CAPACITANCE functionality is based on device terminal layers. If there are locations where routing layers overlap terminal layers where the two are mapped to the same physical layer, StarRC ignores capacitance to the terminal layer but retains capacitance to the routing layer. Thus, any such overlap must be removed from the routing layer to prevent the inadvertent inclusion of MOS device capacitances when such capacitances are to be ignored, and to prevent the inclusion of parasitic capacitance between the terminals of a designed capacitor device.

**Incorrect:**

```

ngate = poly AND ndiff
nsd = ndiff NOT poly
DEVICE MN(n) ngate nsd (G) nsd (S) nsd (D) psub (B)

cap_top = m2 AND cap_recog
cap_bot = m1 AND cap_recog
cap_dev = cap_top AND cap_bot
DEVICE C(m1m2cap) cap_dev cap_top (POS) cap_bot (NEG)

CONNECT poly ngate
CONNECT cap_top m2
CONNECT cap_bot m1

```

**Correct:**

```

ngate = poly AND ndiff
nsd = ndiff NOT poly
poly_route = poly NOT ngate
DEVICE MN(n) ngate nsd (S) nsd (D) psub (B)

cap_top = m2 AND cap_recog
cap_bot = m1 AND cap_recog
cap_dev = cap_top AND cap_bot
m2_route = m2 NOT cap_top
m1_route = m1 NOT cap_bot
DEVICE C(m1m2cap) cap_dev cap_top (POS) cap_bot (NEG)

CONNECT poly_route ngate
CONNECT cap_top m2_route
CONNECT cap_bot m1_route

```

- A physical layer cannot connect to another physical layer without using a via, unless the two physical layers are covertical.

In the rule file example, the poly\_route and ngate layers are allowed to directly connect without a via because the two runset layers map either to the same physical layer (if field and gate poly are not separated in the ITF file) or to covertical physical layers (if field and gate poly are separated in the ITF file). Similarly, the cap\_top layer is allowed to directly connect to the m2\_route layer because both are mapped to the same physical layer M2. The same is true for the cap\_bot and m1\_route layers.

- Simple STAMP commands in the Calibre rule file are supported. However, the StarRC tool ignores composite STAMP commands and issues an SX-1242 warning message.

---

## Required Commands

The following commands are required in the Calibre rule file to ensure the Calibre Connectivity Interface query server can properly generate StarRC input files:

- MASK SVDB DIRECTORY "svdb" CCI

The Calibre Connectivity Interface flag ensures that Calibre writes all required information to the svdb directory such that the Calibre query server can generate all Calibre Connectivity Interface outputs required for StarRC.

- PORT LAYER TEXT

Top-level ports are identified by text layers designated as PORT LAYER TEXT in the Calibre rule file. This information in turn is used by the Calibre Connectivity Interface query server to generate the CALIBRE\_CELLS\_PORTS file, which StarRC uses to netlist top-level ports (\*|P or \*P) in the parasitic netlist. Thus, when top-level ports are required in the parasitic netlist, it is essential that PORT LAYER TEXT be used in the Calibre rule file.

---

## Support for Calibre Preprocessor Directives and Include Statements

StarRC can correctly read the Calibre rule file preprocessor directives #DEFINE, #UNDEFINE, #IFDEF, #IFNDEF, and #ENDIF. In interpreting #IFDEF and #IFNDEF statements, StarRC requires that conditional names be defined with #DEFINE directives directly within the rule file.

StarRC does not support names defined outside the rule file as shell environment variables. Such names are typically prefixed by a dollar sign (\$) within #IFDEF and #IFNDEF directives. If StarRC encounters a conditional within the rule file that uses a shell variable prefixed by \$, and that \$-prefixed variable does not occur previously in the rule file within an explicit #DEFINE directive, StarRC interprets the variable as undefined and issues the following warning:

```
WARNING: StarXtract
WARNING: CALIBRE_RUNSET preprocessor directives which refer
WARNING: to shell or ENV variables are not supported.
WARNING: Conditional expressions which refer to external
WARNING: variables will evaluate as though the variable is undefined.
WARNING: See layers.sum for a full list of affected directives.
```

StarRC also reads Calibre rule file INCLUDE statements that allow one rule file to instantiate another rule file. In cases where preprocessor directives occur across rule files instantiated with INCLUDE, StarRC correctly interprets the directives and applies them to all conditionals across the rule file hierarchy.

---

## Rule File Example

The following is a typical Calibre Connectivity Interface rule file for StarRC transistor-level extraction.

```
LAYOUT SYSTEM GDSII
LAYOUT PATH "XT_DEVICES.GDS"
LAYOUT PRIMARY "XT_DEVICES"
LAYOUT ERROR ON INPUT YES

LVS POWER NAME VDD
LVS GROUND NAME VSS
MASK SVDB DIRECTORY "svdb" CCI
TEXT DEPTH PRIMARY
PRECISION 100

// GDSII Input Layer Mapping
LAYER NWELL 1
LAYER ACTIVE 2
LAYER POLY 3
LAYER BORON 4
LAYER CONTACT 5
LAYER METAL1 6
LAYER VIA 7
LAYER METAL2 8
LAYER PWELL 10
LAYER CAP_REG 12
LAYER POLYRES 40
LAYER BJT_REG 13
LAYER DIODE 14

// GDSII Input Text Layer Mapping
TEXT LAYER 20
LAYER MAP 20 TEXTTYPE >= 0 20
TEXT LAYER 30
LAYER MAP 30 TEXTTYPE >= 0 30

// Layer Derivations
psub1 = EXTENT
diode_active = INTERACT ACTIVE DIODE
bjt_active = INTERACT ACTIVE BJT_REG
active1 = (ACTIVE NOT diode_active) NOT bjt_active
bjt_nwell = INTERACT NWELL BJT_REG
nwell1 = NWELL NOT bjt_nwell
psub = psub1 NOT nwell1
poly_cont = POLY AND CONTACT
diff_cont = CONTACT NOT poly_cont
res_poly = INTERACT POLY POLYRES
poly1 = POLY NOT res_poly
res_body = res_poly AND POLYRES
res_term = res_poly NOT res_body
pactive = BORON AND active1
```

```

nactive = active1 NOT pactive
diff = pactive OR nactive
pgate = poly1 AND pactive
5tngate = INTERACT (poly1 AND nactive) PWELL
ngate = NOT INTERACT (poly1 AND nactive) PWELL
psd1 = pactive NOT pgate
nsd1 = (nactive NOT ngate) NOT 5tngate
subtap = psd1 NOT nwell1
weltap = (nsd1 AND nwell1) NOT PWELL
psd = psd1 NOT subtap
pplus = diode_active AND BORON
nplus = diode_active NOT BORON
emitter = BORON AND bjt_active
bjt_nwell_ntap = active1 AND bjt_nwell
5tnsd = INTERACT nsd1 5tngate
nsd = (nsd1 NOT weltap) NOT 5tnsd
poly_cap_term = (poly1 AND METAL1) AND CAP_REG
fpoly = (poly1 NOT poly_cap_term) NOT diff
metall1_cap_term = (METAL1 AND poly1) AND CAP_REG
m1 = METAL1 NOT metall1_cap_term
dpn_nwell_term = INTERACT NWELL pplus
dnp_psub_term = INTERACT psub nplus

// Text Attachments
ATTACH 20 m1
PORT LAYER TEXT 20
ATTACH 30 METAL2
PORT LAYER TEXT 30

// Layer Connectivity
CONNECT m1 metall1_cap_term
CONNECT METAL2 m1 metall1_cap_term BY VIA
CONNECT m1 psd nsd weltap subtap 5tnsd BY diff_cont
CONNECT m1 nplus pplus emitter bjt_nwell_ntap BY diff_cont
CONNECT metall1_cap_term psd nsd weltap subtap 5tnsd BY
diff_cont
CONNECT metall1_cap_term nplus pplus emitter bjt_nwell_ntap
BY diff_cont
CONNECT m1 fpoly res_term poly_cap_term BY
poly_cont
CONNECT metall1_cap_term fpoly res_term poly_cap_term BY
poly_cont
CONNECT poly_cap_term fpoly
CONNECT pgate ngate 5tngate fpoly
CONNECT NWELL dpn_nwell_term weltap
CONNECT psub dnp_psub_term subtap PWELL
CONNECT bjt_nwell bjt_nwell_ntap

// Device Definitions
DEVICE D(DPN) pplus pplus (POS) dpn_nwell_term (NEG)
DEVICE D(DNP) nplus dnp_psub_term (POS) nplus (NEG)
DEVICE MP(p) pgate pgate (G) psd (S) psd (D) NWELL (B)<diff>

```

```

[
 property L, W, AD, PD, AS, PS
 W = (perimeter_coincide(pgate,psd) / 2)
 A = AREA(pgate)
 L = A/W
 AD = area(D) * W / perimeter_inside(D, diff)
 AS = area(S) * W / perimeter_inside(S, diff)
 PD = perimeter(D) * W / perimeter_inside(D, diff)
 PS = perimeter(S) * W / perimeter_inside(S, diff)
]
DEVICE MN(n) ngate ngate (G) nsd (S) nsd (D) psub (B) <diff>
[
 property L, W, AD, PD, AS, PS
 W = (perimeter_coincide/ngate,nsd) / 2)
 A = AREA(ngate)
 L = A/W
 AD = area(D) * W / perimeter_inside(D, diff)
 AS = area(S) * W / perimeter_inside(S, diff)
 PD = perimeter(D) * W / perimeter_inside(D, diff)
 PS = perimeter(S) * W / perimeter_inside(S, diff)
]
DEVICE MN(5tn) 5tnGate 5tnGate (G) 5tnsd (S) 5tnsd (D)
PWELL (B) NWELL (B2) <diff>
[
 property L, W, AD, PD, AS, PS
 W = (perimeter_coincide(5tnGate,5tnsd) / 2)
 A = AREA(5tnGate)
 L = A/W
 AD = area(D) * W / perimeter_inside(D, diff)
 AS = area(S) * W / perimeter_inside(S, diff)
 PD = perimeter(D) * W / perimeter_inside(D, diff)
 PS = perimeter(S) * W / perimeter_inside(S, diff)
]
DEVICE R(poly_res) res_body res_term (POS) res_term (NEG)
DEVICE C(poly_cap) CAP_REG poly_cap_term (POS)
metall_cap_term (NEG)
DEVICE Q(pbjt) psub psub (C) bjt_nwell (B) emitter (E)

// COMPARE section
SOURCE SYSTEM SPICE
SOURCE CASE YES
SOURCE PATH "XT_DEVICES.sp"
SOURCE PRIMARY "XT_DEVICES"
LAYOUT CASE YES

LVS REPORT report.lvs
LVS REPORT OPTION A B C D
LVS REPORT MAXIMUM 100
LVS IGNORE PORTS NO
LVS REDUCE SPLIT GATES YES
LVS RECOGNIZE GATES ALL

```

```
LVS COMPARE CASE NAMES TYPES
LVS CHECK PORT NAMES NO
LVS REDUCE PARALLEL MOS YES
LVS NL PIN LOCATIONS YES
LVS COMPONENT SUBTYPE PROPERTY mode
```

---

## Preparing the Mapping File

This information explains the rules for preparing mapping files for use with the Hercules, IC Validator, and Calibre tools. A mapping file example is included.

---

### Mapping Rules

The mapping file maps the layers in CONNECT sequences to physical layers in the ITF file.

#### No ideal layer Is allowed

Every layer in the CONNECT section of the Hercules or IC Validator runset or Calibre rule file needs to be mapped to a conductor or via in the ITF file. Similarly, every device terminal layer needs to be mapped to an ITF conductor layer to ensure devices are properly connected to the parasitic mesh in the parasitic netlist. Nonterminal device recognition layers should not be mapped, with one exception: RES device body layers should be mapped either as conducting layers (to include the impact of the RES body on surrounding nets) or remove\_layers (to ignore the impact of the RES body on surrounding nets).

If a runset layer CONNECT or device terminal layer cannot be mapped to any physical layer, it should be listed under the remove\_layers section of the mapping file.

#### Bulk layer mapping must be considered

A bulk layer is any layer that serves as the bulk terminal connection in any device extraction command. Bulk layers mapped in the `conducting_layers` section of the mapping file are used during parasitic extraction only if the `TRANSLATE_RETAIN_BULK_LAYERS` command is used with options other than the default.

The mapping of bulk and nonbulk layers is advisable under the following circumstances:

- Power net extractions where capacitance to well layers should be generated in the netlist as coupling capacitance
- Extractions containing well devices (for example, well resistors, diodes, BJTs) whose database terminal layers consist solely of bulk layers
- Scenarios in which the bulk terminals of designed devices should be generated in the netlist as instance port subnodes instead of ideal nodes

The StarRC tool makes a distinction between true bulk layers and other layers that are physically part of the substrate, such as wells. These are referred to as nonbulk layers.

If there are any other layers in any CONNECT sequences that connect solely to bulk layers, these layers should be mapped in the same manner as the bulk layers to which they connect. This is because StarRC only allows a runset layer to be designated as a conducting layer if the layer has connectivity to other translated runset layers in the design.

### **Map port layers as marker layers to obtain top-level ports**

Top-level ports in the Hercules and IC Validator flows are identified by the existence of polygons mapped as marker layers. In MARKER\_GENERATION: AUTOMATIC mode, such layers are generated by the use of TEXT\_Polygon commands in the Hercules runset.

No such mapping is required in the Calibre Connectivity Interface flow, because top-level markers are identified by listings in the CALIBRE\_CELLS\_PORTS file output by the Calibre Connectivity Interface query server.

---

## Mapping File Example

In the following mapping file example, the asterisk (\*) indicates comments.

```
conducting_layers
 * Routing layers
 Metal2 metal2
 Metal1 metal2
 field_poly poly
 Nwell SUBSTRATE
 welltie SUBSTRATE
 subtie SUBSTRATE
 * Device layers connected to routing layers
 ngate gate
 pgate gate
 Cap1 metall1
 Cap2 poly
 pres_body poly
 * Device layers built into the SUBSTRATE
 nsd SUBSTRATE
 psd SUBSTRATE
 PnAnodeTerm SUBSTRATE
 PnCathodeTerm SUBSTRATE
 NpnEmitter SUBSTRATE
 NpnCollector SUBSTRATE
 NpnBase SUBSTRATE

via_layers
 * Via layers
 Vial vial
 PolyCont poly_con
 SubCont sub_con
 NpnEmitterCont sub_con
 NpnCollectorCont sub_con
 NpnBaseCont sub_con

marker_layers (only required in Hercules flow)
 * Marker layer
 Metal2_Mark

remove_layers
```

---

## Running the Calibre Query Server for Output to StarRC

This section discusses the Calibre query server commands required to generate proper Calibre Connectivity Interface files. The commands listed in the query server command file shown in [Example 9-2](#) should be provided to the `calibre -query svdb` command.

Note:

Calibre query commands are executed sequentially. Changing the order of the commands might affect the results.

### *Example 9-2 Calibre Query Command File*

```
LAYOUT NETLIST DEVICE LOCATION CENTER
Instructs query server to write net ID to seed polygons
GDS SEED PROPERTY ORIGINAL

Instructs query server to output further information about the
device recognition layer to the CCI database.
GDS SEED PROPERTY DEVICE ORIGINAL

#Generates the GDS_MAP layer file.
RESPONSE FILE GDS_MAP
GDS MAP
RESPONSE DIRECT

The following lines define the property numbers for net names and
instance names. StarRC expects the NETPROP number to be 5, the
PLACEPROP number to #be 6, and INSTPROP number to be 7. Do not change.
GDS NETPROP NUMBER 5
GDS PLACEPROP NUMBER 6
GDS DEVPROP NUMBER 7

#Outputs Calibre AGF file for StarRC.
GDS WRITE agf_file

These commands ensure pin coordinates and proper hierarchy
in the ideal layout netlist written out by Calibre.
AGF is the only allowed keyword for LAYOUT NETLIST HIERARCHY.
LAYOUT NETLIST DEVICE LOCATION CENTER
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST PIN LOCATIONS YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE netlist_file
```

```

Outputs Calibre ideal layout name map for StarRC.
The EXPAND_CELLS keyword for LAYOUT NAMETABLE WRITE ensures that the
lnn file and the netlist have the same hierarchy. The EXPAND_CELLS
keyword requires Calibre version 2014.3 or later.
LAYOUT NAMETABLE WRITE lnn_file EXPAND_CELLS

Outputs Calibre net cross-referencing table file; required to run XREF.
The LNXF keyword requires Calibre version 2014.3 or later.
NET XREF WRITE nxf_file LNXF

Outputs Calibre instance cross-referencing table for StarRC
This is required to run XREF.
INSTANCE XREF WRITE ixf_file

Outputs Calibre cell extents file for StarRC
CELL EXTENTS WRITE extents.txt

Outputs Calibre top-level ports file for StarRC.
PORT TABLE CELLS WRITE cells_ports_file

Outputs Calibre device table report file for StarRC.
RESPONSE FILE devtab_file
DEVICE TABLE
RESPONSE DIRECT

```

When creating your query server command file, keep the following points in mind:

- The path of the LVS extraction file must be included in the Calibre query file.

```
LVS SETTINGS REPORT WRITE ./CCI_query/extraction_report_file
```

When you include the LVS extraction report file in the Calibre Connectivity Interface, StarRC does not need to parse the Calibre rulefile, so the Calibre runset is not required. It is the responsibility of Mentor to ensure that the information in the LVS extraction report is accurate and handles directives and conditional statements.

- Additional GDS MAP layer mapping commands to specify specific GDS layer numbers are not required for the query server to output relevant connectivity and device terminal layers to the AGF file. They simply change the AGF output layer number. Using GDS MAP commands in this way is problematic, because they can cause the query server to output layers to a layer number that is already internally mapped to another layer by Calibre. Only the following commands are essential for correct layer mapping in StarRC:

```
RESPONSE FILE GDS_MAP
GDS_MAP
RESPONSE DIRECT
```

- The following commands must be specified to identify the properties that StarRC should obtain from the AGF file:

```
GDS NETPROP NUMBER
GDS PLACEPROP NUMBER
GDS DEVPROP NUMBER
```

- For Hercules flows, the following command directs Calibre to report the device center as the device location instead of the default vertex:

```
LAYOUT NETLIST DEVICE LOCATION CENTER
```

- If you use Virtuoso Integration with the hierarchical Calibre Connectivity Interface, add the following command to the query server command file:

```
HIERARCHY SEPARATOR |
```

This forces Calibre to use the pipe character (|) as the hierarchical separator for compatibility with Virtuoso Integration.

## Stripping X-Cards in Source Names

The following command directs Calibre to strip X-cards in source names:

```
XREF XNAME SOURCE OFF
```

This command strips X-card prefixes at each hierarchical level of source net and instance names in the NXF and IXF tables. Such functionality is useful in StarRC gate-level extractions based on hierarchical Calibre LVS runs, in which the StarRC parasitic netlist must backannotate to an original Verilog netlist that does not contain X-card prefixes in each level of hierarchy of a hierarchical net or instance name.

The effect on the StarRC output parasitic netlist of using this Calibre query server command is illustrated by the following SPF example:

A StarRC netlist without the `XREF XNAME SOURCE OFF` command:

```
* | NET XA/XB/XC/net0 0.225231PF
* | I (XA/XB/XC/MM1:D XA/XB/XC/MM1 D B 0 13 175)
R1 XA/XB/XC/MM1:D XA/XB/XC/net0:4 1.2335
Cg1 XA/XB/XC/net0:4 1.63099e-16
```

A StarRC netlist with the `XREF XNAME SOURCE OFF` command:

```
* | NET A/B/C/net0 0.225231PF
* | I (A/B/C/MM1:D A/B/C/MM1 D B 0 13 175)
R1 A/B/C/MM1:D A/B/C/net0:4 1.2335
Cg1 A/B/C/net0:4 1.63099e-16
```

**Note:**

The XREF XNAME SOURCE OFF command should appear in the query server command file before the NET XREF WRITE and INSTANCE XREF WRITE commands.

The corresponding Calibre query server XREF XNAME LAYOUT OFF command should not be used with StarRC. This is because the layout netlist generated by Calibre always contains X-cards regardless of the setting of XREF XNAME LAYOUT, and StarRC requires consistency between the NXF and IXF layout names and the Calibre-generated layout netlist.

This command is not applicable to and produces no differences for StarRC runs based on Calibre flat LVS runs.

See the Calibre documentation for further details.

---

## Multifinger Device Support in the Calibre Connectivity Interface

The Calibre Connectivity Interface provides multifinger access resistance calculation by creating pins for each finger. To use this feature, add the following Calibre query command:

```
GDS SEED PROPERTY DEVICE ORIGINAL
```

With this command in the query command file, StarRC initiates automatic pin detection to obtain the pin location for each terminal of the device. When there is more than one polygon for one terminal, the automatic pin detection code generates a pin location for each terminal polygon. With this information, StarRC can provide more accurate resistance extraction results.

---

## Optional Layout Netlisting Query Commands

### The Push Down Separate Properties (PDSP) Flow

The PDSP flow is the recommended backannotation flow. In the PDSP flow, the separate properties are written to a file during query output generation, not during the LVS run. The StarRC tool automatically reads the file from the query output.

To use the PDSP flow, insert these commands into the query command file:

```
LAYOUT NETLIST DEVICE LOCATION CENTER
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST SEPARATED PROPERTIES NO
LAYOUT NETLIST PIN LOCATIONS NO
LAYOUT NETLIST DEVICE TEMPLATES YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE nl
LAYOUT SEPARATED PROPERTIES WRITE CCI/pdsp
```

## The Push Down Back-Annotation (PDBA) Flow

The PDBA flow is an older backannotation flow. In the PDBA flow, the separate properties are written to a file during the LVS run. The file must be referenced in the StarRC command file by using the CALIBRE\_PDBA\_FILE command.

To use the PDBA flow, insert these commands into the query command file:

```
LAYOUT NETLIST DEVICE LOCATION CENTER
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST SEPARATED PROPERTIES NO
LAYOUT NETLIST PIN LOCATIONS NO
LAYOUT NETLIST DEVICE TEMPLATES YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE nl
```

## Error Conditions for the PDSP and PDBA Flows

StarRC issues warning or error messages if the tool finds unsupported commands in the Calibre query command file. For error messages, extraction also stops. The affected Calibre query commands are shown in [Table 9-1](#).

*Table 9-1 Unsupported Calibre Query Commands*

| Command                                                                                                  | Setting                               | StarRC result                      |
|----------------------------------------------------------------------------------------------------------|---------------------------------------|------------------------------------|
| LAYOUT NETLIST PIN LOCATIONS<br>LAYOUT NETLIST DEVICE TEMPLATES<br>(used together in PDSP or PDBA flows) | NO (default)<br>YES                   | allowed                            |
| LAYOUT NETLIST PIN LOCATIONS<br>LAYOUT NETLIST DEVICE TEMPLATES<br>(used together in PDSP or PDBA flows) | NO<br>NO (default)                    | warning                            |
| LAYOUT NETLIST PIN LOCATIONS<br>LAYOUT NETLIST DEVICE TEMPLATES<br>(used together in PDSP or PDBA flows) | YES<br>YES                            | warning                            |
| LAYOUT NETLIST PIN LOCATIONS<br>LAYOUT NETLIST DEVICE TEMPLATES<br>(used together in PDSP or PDBA flows) | YES<br>NO                             | warning                            |
| LAYOUT NETLIST SEPARATED PROPERTIES<br>(used in PDSP or PDBA flows)                                      | NO (default)<br>YES                   | allowed<br>error                   |
| LAYOUT NETLIST HIERARCHY<br>(all flows including PDSP and PDBA flows)                                    | AGF<br>ALL (default)<br>FLAT<br>HCELL | allowed<br>error<br>error<br>error |

---

## Preparing the StarRC Command File

Instructions for preparing a StarRC command file are found in this section. Options to be included in this file are explained.

---

### Commands for Hercules Flows

To run the transistor-level Hercules flow, you must use the following commands:

```
MILKYWAY_DATABASE: LIB_NAME
MILKYWAY_EXTRACT_VIEW: YES
BLOCK: BLOCK
SKIP_CELLS: !*
TCAD_GRD_FILE: technology.nxtgrd
MAPPING_FILE: mapping_file_name
```

MILKYWAY\_DATABASE: *LIB\_NAME* is the path to the directory having the XTR view generated by Hercules. It should be the same as the *LIB\_NAME* in the WRITE\_EXTRACT\_VIEW command in the Hercules runset. However, you are allowed to set a new path if you have moved the directory to another place. Note that the default for SKIP\_CELLS is “\*” and you must make sure that you set it as “!\*” to extract down to the transistor level.

---

### Commands for IC Validator Flows

To run the transistor-level IC Validator flow, you must use the following commands:

```
ICV_RUNSET_REPORT_FILE: file_name
BLOCK: BLOCK
SKIP_CELLS: !*
TCAD_GRD_FILE: technology.nxtgrd
MAPPING_FILE: mapping_file_name
```

StarRC obtains information such as connection information, the location of LVS COMPARE output, device pin and properties information, and the location of the extract view from the runset report file specified by the ICV\_RUNSET\_REPORT\_FILE command. The default for SKIP\_CELLS is “\*” and you must make sure that you set it as “!\*” to extract down to the transistor level.

---

## Commands for Calibre Connectivity Interface Flows

To run the transistor-level Calibre Connectivity Interface flow, you must specify the following options:

```
BLOCK: block_name
SKIP_CELLS: !*
TCAD_GRD_FILE: technology.nxtgrd
MAPPING_FILE: mapping_file_name
CALIBRE_QUERY_FILE: calibre_query_command_file
```

Calibre options are explained in [The Calibre Connectivity Interface Flow](#).

---

## Other StarRC Commands

The following commands are also important for StarRC command file preparation.

### **NETLIST\_FORMAT**

The **NETLIST\_FORMAT** command defines the structure and format of the output parasitic netlist:

```
NETLIST_FORMAT: SPF | SPEF | NETNAME | OA
```

For transistor-level extraction, StarRC supports four output netlist formats: **SPF**, **SPEF**, **OA**, and **NETNAME** formats. You can use the **NETLIST\_CONNECT\_SECTION: NO** and **NETLIST\_NODE\_SECTION: NO** commands to suppress \*|I and \*|S sections.

### **IGNORE\_CAPACITANCE**

This command disallows certain types of device-level capacitances from being extracted and netlisted:

```
IGNORE_CAPACITANCE: ALL | DIFF | NONE
```

You can ignore both gate and diffusion capacitance with the **IGNORE\_CAPACITANCE** command.

If you set **IGNORE\_CAPACITANCE: ALL**, both overlap and sidewall capacitances between gate and **SUBSTRATE** is ignored. All other coupling effects to the gate poly node from other conducting layers are considered. If you set **IGNORE\_CAPACITANCE** to **ALL** or **DIFF**, you can ignore the junction capacitance between diffusion and **SUBSTRATE**.

The **IGNORE\_CAPACITANCE** command is runset-layer-based. This means that each layer in the runset (even if multiple runset layers are mapped to a single ITF layer) is processed individually. If the runset defines a PMOS device that uses **NWELL** as the **BULK** layer, **NWELL** must be the only layer under the device for the **IGNORE\_CAPACITANCE** command to work.

If the runset contains a connected copy of NWELL that is also present under every PMOS device, the `IGNORE_CAPACITANCE` command has no effect (because the NWELL copy is not tagged as a MOS BULK layer). The runset should not contain connected copies of BULK.

## XREF

This command determines which set of names to report for StarRC netlist generation and analysis flows and which devices and nets to retain if both layout names and cross-referenced schematic names are present.

`XREF: NO | YES | COMPLETE`

The XREF command enables cross-referencing of the parasitic netlist for LVS-based extraction flows. Nets, devices, and cell instances are output in the parasitic netlist using schematic names, according to the functionality of the two available modes `YES` and `COMPLETE`.

Note:

`XREF: NO | YES | COMPLETE` is available in the Hercules flow, while `XREF: NO | YES` is available in the Calibre Connectivity Interface flow. For details about the use of `XREF` in Calibre Connectivity Interface flows, see [The Calibre Connectivity Interface Flow](#).

If you set `XREF:COMPLETE`, all the nets that are not cross-referenced are ignored. Only successfully matched nets and instances are netlisted in the output file.

If you want to use the `SKIP_CELLS` command in XREF-enabled flows, be sure that the `SKIP_CELLS` are EQUIV points (in the Hercules or IC Validator flows) or hcells (in the Calibre flow) that were successfully matched during LVS. Those cells that are not matched during LVS cannot be properly skipped.

Because StarRC gets cross-reference information from the evaccess and compare directories, you should not remove them (in the Hercules or IC Validator flow) before the StarRC run is over.

## COMPARE\_DIRECTORY and EVACCESS\_DIRECTORY

The `COMPARE_DIRECTORY` command specifies the location of the Hercules LVS COMPARE output. The `EVACCESS_DIRECTORY` command specifies the location of the Hercules LVS EvAccess database.

`COMPARE_DIRECTORY: PATH_TO_compare_DIRECTORY`  
`EVACCESS_DIRECTORY: PATH_TO_evaccess_DIRECTORY`

In `XREF: YES` or `COMPLETE` runs, StarRC gets the cross-reference information and schematic netlist information from the compare and evaccess directories. If the paths to the original compare and evaccess directories have changed, you have to set the `COMPARE_DIRECTORY` and `EVACCESS_DIRECTORY` commands with the new paths. These options can be skipped if they are the same as the Hercules run.

---

## Interconnect Technology Format File

This section explains how to prepare the Interconnect Technology Format (ITF) file. For more information, see [Flows for Process Characterization](#).

---

### Preparing the ITF File

Preparing an ITF file for transistor-level extraction is similar to cell-level extraction; the difference is that transistor-level ITF files have the connectivity information down to the diffusion or SUBSTRATE layers. Both planar as well as nonplanar process files can be created.

The transistor-level ITF file can be either planar, using a single poly layer for both field and gate poly, or nonplanar, using separate field and gate poly layers. Using a planar process for transistor-level extraction allows for faster nxtgrd file generation time relative to the nonplanar process, because one less CONDUCTOR layer exists in a planar ITF file. However, the choice is optional and should be dictated by the parameters of the actual physical process.

An ITF file contains via layer information and you need to be careful when defining them. Because a via layer can connect only two conducting layers, you must separate a poly contact from a SUBSTRATE contact (and diffusion contact, if any).

---

### Limitations

Coverting layers (such as gate and field poly) should be vertically overlapping. Thus if the bottom of the field poly layer is higher than the top of the gate poly layer (as happens when the field oxide is too thick or the gate poly is too thin), they are not connected to each other by mere touch. In this case, you need to modify your runset or rule file, mapping file, and ITF file to make a dummy via layer connecting the gate and field poly.

---

## ITF File Example

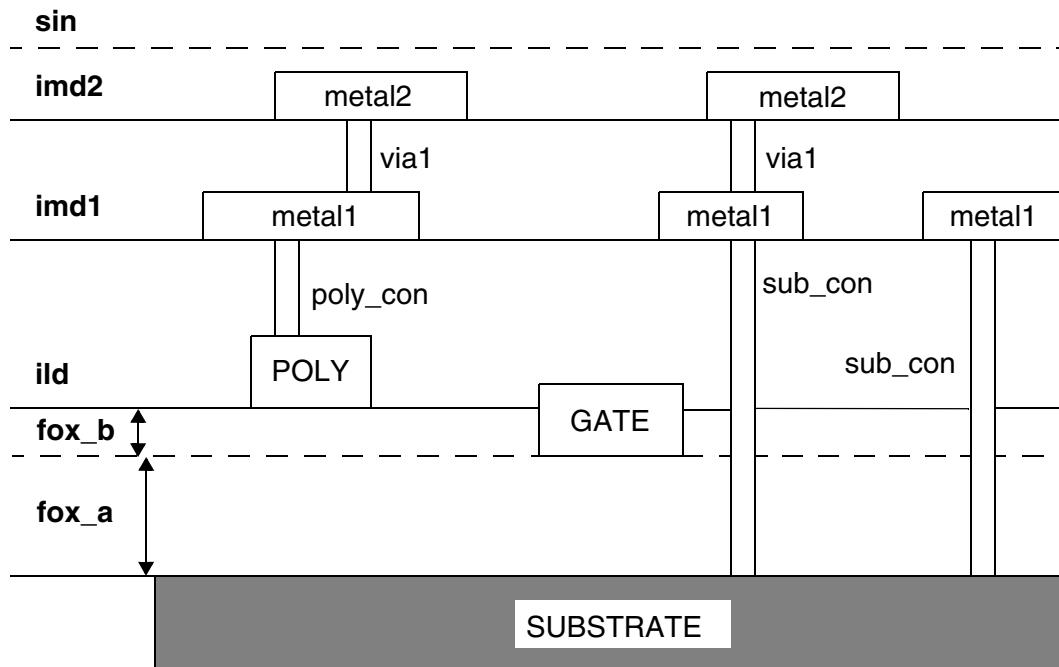
The following example shows an ITF file that can be used with the two earlier file examples provided in the [Runset Example](#) and [Interconnect Technology Format File](#) sections. The topology view is shown in [Figure 9-4](#).

```
TECHNOLOGY=add4_2m1p

DIELECTRIC sin {THICKNESS=0.70 ER=7.9}
DIELECTRIC imd2 {THICKNESS=1.00 ER=4.2}
CONDUCTOR metal2 {THICKNESS=0.50 WMIN=0.35 SMIN=0.35 RPSQ=0.07}
DIELECTRIC imd1 {THICKNESS=1.05 ER=4.2}
CONDUCTOR metal1 {THICKNESS=0.35 WMIN=0.30 SMIN=0.30 RPSQ=0.08}
DIELECTRIC i1d {THICKNESS=0.70 ER=4.1}
CONDUCTOR poly {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_b {THICKNESS=0.10 ER=3.9}
CONDUCTOR gate {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_a {THICKNESS=0.25 ER=3.9}

VIA via1 {FROM=metal1 TO=metal2 AREA=0.09 RPV=1}
VIA poly_con {FROM=poly TO=metal1 AREA=0.04 RPV=12}
VIA sub_con {FROM=SUBSTRATE TO=metal1 AREA=0.04 RPV=16}
```

*Figure 9-4 Topology of File Example*



---

## Transistor-Level Extraction Limitations

The following general limitations apply:

- Covertical layers should be overlapping vertically.
- Only `XREF: YES` is supported in the Calibre Connectivity Interface flow.
- In the Calibre Connectivity Interface flow, each device extraction command listed in the rule file must have a unique model name, to allow StarRC to properly distinguish devices.

The following limitations apply to cross-referencing flows:

- `DETECT_PERMUTABLE_PORTS` is not supported.
- `PUSH_DOWN_DEVICES (Hercules)` is not handled correctly.
- `PUSH_DOWN_PINS: FALSE (Hercules)` should be set when possible to achieve best results with `XREF`.
- In `XREF:COMPLETE` flows in which  $m(\text{schematic}) : n(\text{layout})$  or  $m:1$  device merging occurs, StarRC netlists nets connecting to device terminals as ideal nets. This is because no method exists by which to distribute parasitics from a single layout net across the multiple schematic nets that are netlisted in the `XREF:COMPLETE` netlist.
- In `XREF:COMPLETE` netlists for which multistage  $m:n$  merging occurs for designed passive devices, `NETLIST_PASSIVE_PARAMS` parameters might not appear in the parasitic netlist. No method exists by which to annotate layout device properties from  $N$  layout devices onto  $M$  schematic devices, because no direct one-to-one correlation can be established among the layout and schematic devices.

# 10

## Special-Purpose Features

---

This chapter describes additional features available in the StarRC tool.

- [Clock Net Inductance Extraction](#)
- [Feedthrough Nets](#)
- [Via Coverage](#)
- [Long Ports](#)

## Clock Net Inductance Extraction

Inductance is important at high operating frequencies for long nets, such as clock nets. Inductance effects include signal overshoot and undershoot, bumps in the waveform, and increased clock skew.

Clock net inductance extraction is available for use in targeted StarRC runs, separate from standard full-chip or sign-off extraction runs. The output is a DSPF netlist containing full RLC information for the selected nets. This netlist is typically used as input for HSPICE or other circuit simulation tools.

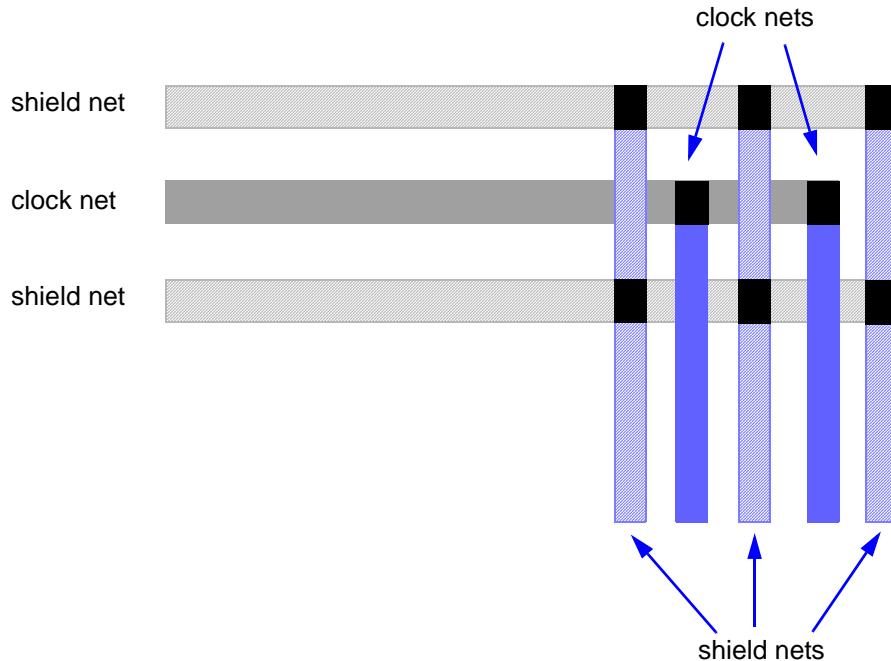
You can use a standard foundry-provided nxtgrd file for StarRC inductance extraction. All process variation effects modeled within the nxtgrd file are acceptable for inductance analysis.

Limitations are as follows:

- Only DSPF netlists are supported for output.
- The design must be a gate-level Milkyway or LEF/DEF design.

An example of a shielded clock net design is shown in [Figure 10-1](#), in which shield nets appear on both sides of the clock net. A design might contain shielded clock nets on more than one layer.

*Figure 10-1 Clock Nets With Shielding*



Shielding is modeled as follows:

- The clock net must be shielded by power or ground nets on the same layer. Signal nets cannot act as shielding nets.
- The current return is assumed to occur exclusively through the power or ground shield nets.
- Inductive coupling between a clock net and its shield nets is ignored.
- Inductive coupling between clock domains is ignored because the clock nets are assumed to be fully shielded.
- Fill polygons between the clock net and its shield nets are allowed.
- Shield nets are modeled to be no farther than five microns away from the clock net, in terms of the spacing between the edges of the clock net and shield net polygons. In other words, the modeled shield distance is either the actual spacing or five microns, whichever is smaller. In some designs, the shield distance might vary along the length of the clock net.

## Setting Up Clock Net Inductance Extraction

The following command file settings are required for an inductance extraction:

```
EXTRACTION: RC
NETS: list_of_cnets
CLOCK_NET_INDUCTANCE: YES
REDUCTION: YES | NO
NETLIST_FORMAT: SPF
(optional) CLOCK_NET_FREQUENCY: cnet_freq
(optional) CLOCK_NET_INDUCTANCE_LAYERS: cnet_layers
```

Only the `RC` setting of the `EXTRACTION` command is allowed. If the command is not explicitly defined, the default is `RC`.

Only the `YES` and `NO` settings of the `REDUCTION` command are allowed. If the command is not explicitly defined, the default is `YES`.

If the `CLOCK_NET_INDUCTANCE_LAYERS` command is used, nets listed in the `NETS` command are extracted only if they are found on the specified layers.

Limitations for the command file are as follows:

- Simultaneous multicorner extraction cannot be used.
- The `POWER_EXTRACT` command must be set to `NO` (the default).

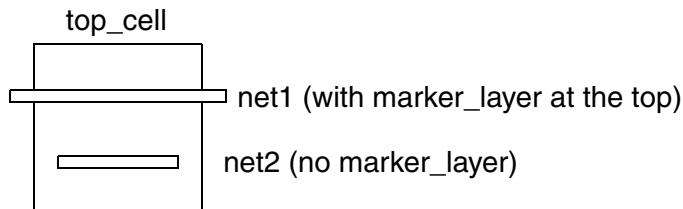
## Feedthrough Nets

A feedthrough net is a net that has one connection to a higher level in the hierarchy. Feedthroughs do not typically exist in a schematic. StarRC handles the following feedthrough cases:

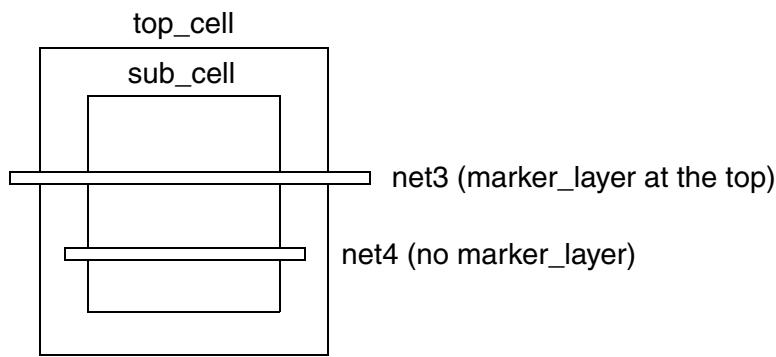
- Extracting lower-level cells that are later used as `SKIP_CELLS`, as shown in [Figure 10-2](#).
- Ensuring proper naming for the feedthrough ports of a skip cell. [Figure 10-3](#) shows an example of extracting the TOP block by skipping the cell with feedthrough ports.

Both issues should be taken care of by default in `XREF:YES` because it is layout-based and the feedthrough exists in the layout.

*Figure 10-2 First Feedthrough Case*



*Figure 10-3 Second Feedthrough Case*



To handle these cases with `XREF:COMPLETE`, use the `XREF_FEEDTHRU_NETS` command, for which the default is `NO`. This command controls feedthrough net handling for the two previous cases when `XREF:COMPLETE` is used.

Note:

Both of these issues can happen for a single cell. For example, if a cell has a `SKIP_CELLS` with feedthrough ports and has a feedthrough port for a bigger cell containing the cell itself, both `*|I` and `*|P` are correctly netlisted.

---

## Feedthrough - First Issue

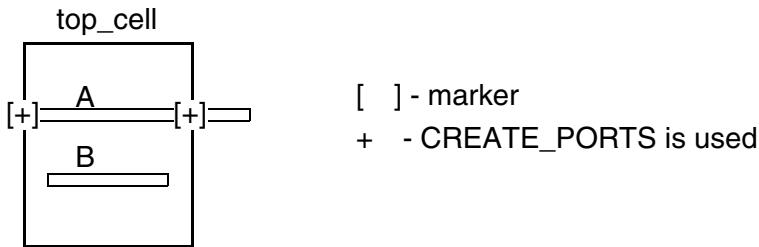
The first feedthrough problem is cross-referencing \*|P for the feedthrough ports of a subblock when extracting the subblock.

If nets or ports at the top cell of the layout are not compared, the netlist of those nets appears with the “In\_” prefix attached to the net names from the layout. This is done by default for XREF : YES, but for XREF : COMPLETE, the XREF\_FEEDTHRU\_NETS command must be set to YES (the default is NO). The prefix itself can be changed with the XREF\_LAYOUT\_NET\_PREFIX command.

The XREF : COMPLETE flow always prints out the prefix for the feedthrough net name, because the net name is always based on the layout.

In the Hercules runset, the CREATE\_PORTS command must be used for feedthrough ports in the XREF : COMPLETE flow and included in the netlist.

*Figure 10-4 XREF:YES Example*



The output is:

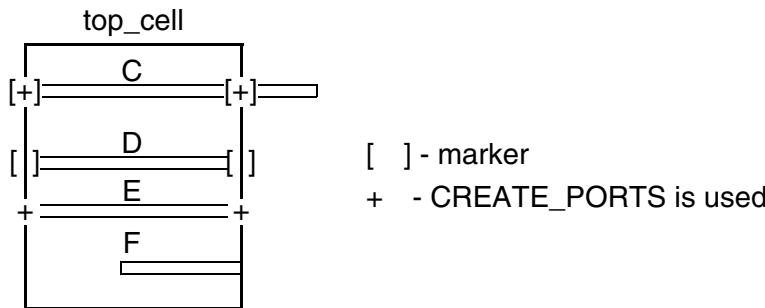
```
...
* |NET ln_A
* |P ln_A
...
* |NET ln_B
...
```

## Port Renaming

To rename feedthrough ports, you can use the SHORT\_PINS : NO option. For example, if SHORT\_PINS : NO, the output would be

```
...
* |NET ln_A
* |P ln_A_1
* |P ln_A_2
...
* |NET ln_B
```

*Figure 10-5 XREF:COMPLETE and XREF\_FEEDTHRU\_NETS:YES Example*



The output is:

```
...
* | NET ln_C
* | P ln_C
...
* | NET ln_E
...
```

Note that nets D and F are not netlisted, because CREATE\_PORTS (Hercules) was not used.

## Feedthrough - Second Issue

When you are cross-referencing \*|| for SKIP\_CELLS feedthrough ports (when extracting the top block with SKIP\_CELLS having feedthrough, as in [Figure 10-3](#)), if XREF\_FEEDTHRU\_NETS is set to YES, the XREF:COMPLETE command has the same behavior as XREF:YES, even though the schematic does not have the feedthrough port connection with the SKIP\_CELLS.

When XREF\_FEEDTHRU\_NETS is NO with XREF:COMPLETE, StarRC ignores the \*|| and the instance section also skips the connection.

If a SPICE\_SUBCKT\_FILE is provided for the SKIP\_CELLS, then the order and content from the SPICE\_SUBCKT\_FILE is maintained in the instance section of the netlist.

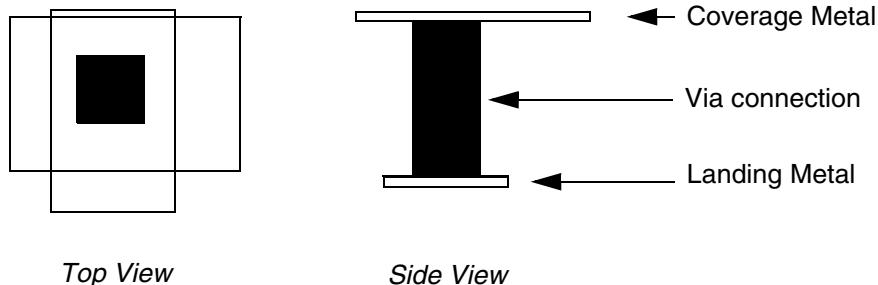
## Runset Requirements

- Hercules LVS for the SKIP\_CELLS having the feedthrough ports must be done with EQUIV statements. Using the Hercules command BLACK\_BOX in the equiv file is not permitted.
- The Hercules CREATE\_PORTS command must be used to properly netlist feedthroughs.

## Via Coverage

The StarRC tool provides a way to report via metal coverage by comparing the coverage metal and landing metal, as shown in [Figure 10-6](#).

*Figure 10-6 Coverage and Landing Metal Connecting a Via*



The `VIA_COVERAGE_OPTION_FILE` command checks rectangular vias; the `VIA_COVERAGE` command checks square vias.

### Via Coverage Commands

To report the via coverage, specify either the `VIA_COVERAGE` or `VIA_COVERAGE_OPTION_FILE` command in the StarRC command file.

Vias specified in a `VIA_COVERAGE` or `VIA_COVERAGE_OPTION_FILE` command must be defined in the ITF file.

Because the coverage and landing areas of vias are used to classify how different types of vias are handled in reliability analysis, you must specify coverage and landing values (in nanometers) for each type to be classified.

Those values are

- Full coverage
- Quarter coverage
- Semicoverage
- Full landing
- Quarter landing
- Semilanding

The via coverage refers to the metal layer above the via and the via landing refers to the metal layer below the via.

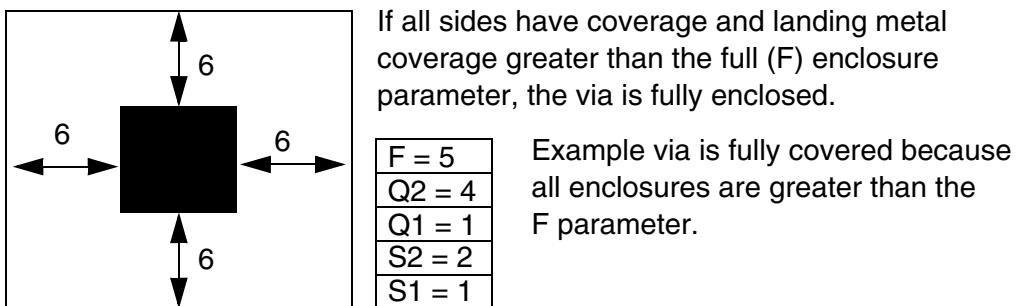
## Determining the Coverage and Landing Areas for Rectangular Vias

Use the `VIA_COVERAGE_OPTION_FILE` command to specify a file containing checking rules for rectangular vias. The following section explains how the rectangular via coverage and landing areas are defined. The *via coverage* refers to the metal layer above the via, and the *via landing* refers to the metal layer below the via.

The following conditions determine via coverage and landing:

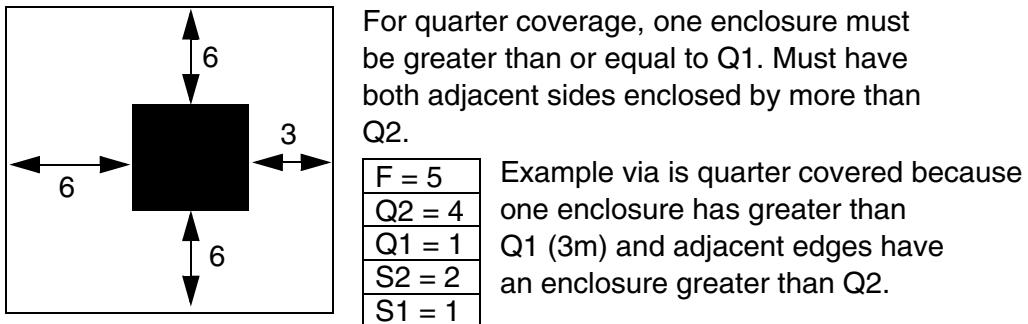
- If all edges are enclosed by the full- coverage parameter, the via is fully covered. See [Figure 10-7](#).

*Figure 10-7 Via Rules for Verifying Full Coverage*



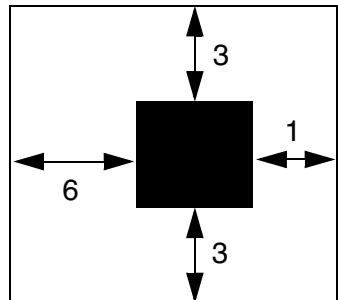
- If the via is not fully covered, it might be quarter covered. If one edge has enclosure greater than or equal to Q1, and BOTH adjacent edges have enclosure greater than Q2, the via is quarter covered. The opposite edge must also have an enclosure greater than or equal to Q1. See [Figure 10-8](#).

*Figure 10-8 Via Rules for Verifying Quarter Coverage*



- If the via is not quarter covered, it might be semicovered. If one edge has enclosure greater than or equal to S1, and BOTH adjacent edges have enclosure greater than S2, the via is semicovered. The opposite edge must also have enclosure greater than or equal to S1. See [Figure 10-9](#).

*Figure 10-9 Via Rules for Verifying Semi Coverage*



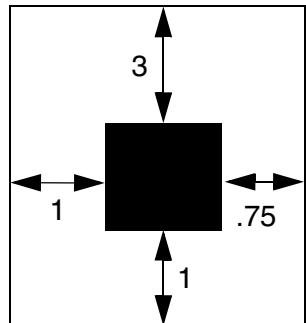
For semicoverage, one enclosure must be greater than or equal to S1. Both adjacent sides must be enclosed by more than S2.

|        |
|--------|
| F = 5  |
| Q2 = 4 |
| Q1 = 1 |
| S2 = 2 |
| S1 = 1 |

Example via is semicovered because one edge has an enclosure greater than or equal to S1 (1m), both adjacent edges have an enclosure greater than S2, and adjacent sides are enclosed by less than Q2.

- If none of the preceding conditions is met, the via is partially covered as shown in [Figure 10-10](#).

*Figure 10-10 Via Rules for Verifying Partial Coverage*



For partial coverage, no enclosure meets the conditions of full, quarter, or semicovered.

|        |
|--------|
| F = 5  |
| Q2 = 4 |
| Q1 = 1 |
| S2 = 2 |
| S1 = 1 |

Example via is partially covered because no edge meets the full, quarter, or semicovered requirements

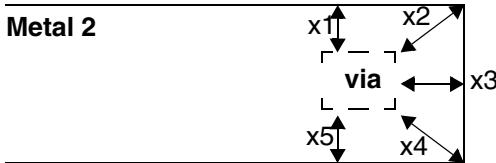
- In instances where a via appears to satisfy both the quarter coverage and semicoverage, the via is considered to be quarter covered.

---

## Determining the Coverage and Landing Areas for Square Vias

Use the `VIA_COVERAGE` command to define the via coverage and landing areas for square vias. The via *coverage* refers to the metal layer above the via, and the via *landing* refers to the metal layer below the via.

*Figure 10-11 VIA\_COVERAGE Behavior*



Coverage for the via in [Figure 10-11](#) is determined by examining the minimum distance, considering the values x1 through x5. [Table 10-1](#) shows the definitions.

*Table 10-1 Via Coverage Definitions*

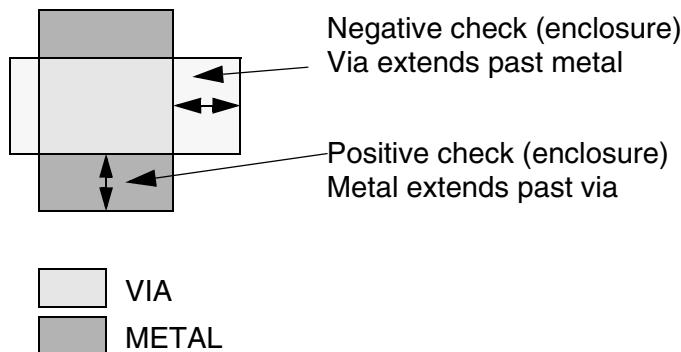
| Minimum Distance                                                      | Coverage              |
|-----------------------------------------------------------------------|-----------------------|
| Greater than or equal to full coverage                                | (F) Fully covered     |
| Greater than or equal to quarter coverage and less than full coverage | (Q) Quarter covered   |
| Greater than or equal to semi coverage and less than quarter coverage | (S) Semi covered      |
| Less than semi coverage                                               | (P) Partially covered |

---

## Positive and Negative Check

Positive and negative checks using `VIA_COVERAGE` are described in the following section. This concept is shown in [Figure 10-12](#).

- Positive check  
Metal edges extend beyond the via edges and metal encloses the via.
- Negative check  
Via extends beyond the edges of the metal polygons.

*Figure 10-12 Positive and Negative Checks*

Of the two commands supporting via coverage capabilities, `VIA_COVERAGE` and `VIA_COVERAGE_OPTION_FILE`, the negative check is only supported in the `VIA_COVERAGE_OPTION_FILE` command.

For the negative check, you specify negative parameters in the `VIA_COVERAGE_OPTION_FILE` command. For metal parameters, StarRC performs the negative check with greater than or equal to values of the negative parameter. If you specify a negative value in a `VIA_COVERAGE` command, StarRC issues an error message. The values of check parameters you specify in the `VIA_COVERAGE_OPTION_FILE` command can be zero. Any coverage larger or equal to zero (for example nonnegative) satisfies the zero coverage and landing check.

## Examples

The examples in this section describe the negative check. [Table 10-2](#) defines the F1, F2, Q1, Q2, S1, and S2 parameters.

*Table 10-2 Negative Check*

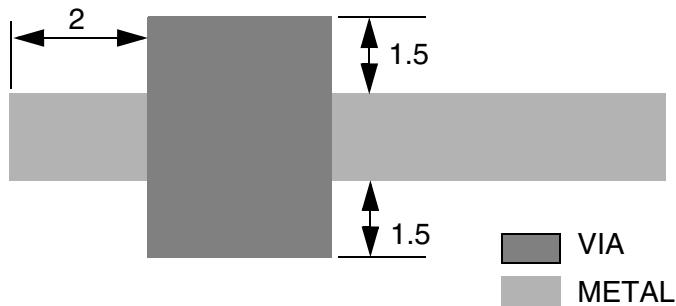
| Parameter | Definition                                                 |
|-----------|------------------------------------------------------------|
| F1        | First landing or coverage parameter for a full check       |
| F2        | Second landing or coverage parameter for a full check      |
| Q1        | First landing or coverage parameter for a quarter check    |
| Q2        | Second landing or coverage parameter for a quarter check   |
| S1        | First first landing or coverage parameter for a semi check |
| S2        | Second landing or coverage parameter for a semi check      |

### Example 1

- Case 1:  $Q1=-1$  and  $Q2=1$

The geometries do not meet  $Q1=-1$  and  $Q2 = 1$  because via edges extend beyond metal by 1.5 or more than 1. Via coverage equals -1.5 which is smaller than -1. This is shown in [Figure 10-13](#).

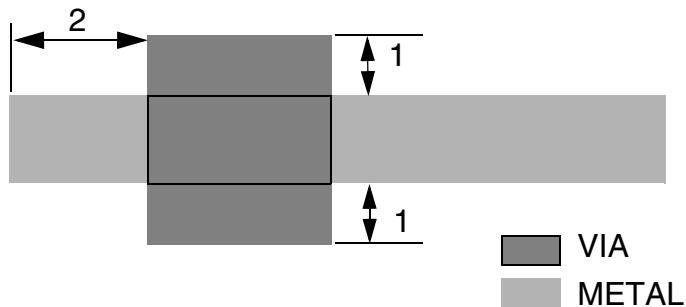
*Figure 10-13 Example 1, Case 1*



- Case 2:  $F2= 5$ ,  $F1= -1$ ;  $Q2= 3$ ,  $Q1= -1$ ;  $S2= 3$ ,  $S1= -2$

The geometries meet  $Q1=-1$  and  $Q2=1$ . Because via edges extend beyond the metal edge by 1, satisfying  $Q1=-1$ ; and the other adjacent opposite metal edges enclose the via by distances larger or equal to 2. This is shown in [Figure 10-14](#).

*Figure 10-14 Example 1, Case 2*

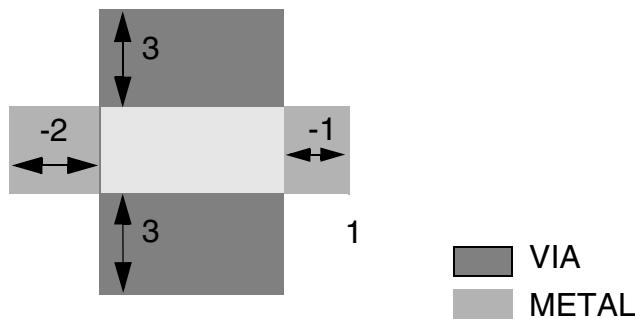


## Example 2

The parameters are  $F2=5$ ,  $F1=-1$ ;  $Q2=3$ ,  $Q1=-1$ ;  $S2=3$ ,  $S1=-2$ . These are shown in [Figure 10-15](#).

- The geometries do not meet  $F2=5$  and  $F1=-1$  because via extends past metal by more than 1 and the adjacent enclosures are less than 5.
- The geometries do not meet  $Q2=3$  and  $Q1=-1$  because one via edge extension past metal is -2 which is less than -1 although another opposite enclosure is equal to -1 and the adjacent enclosures are equal to 3.
- The geometries meet the  $S2=3$  and  $S1=-2$  because the via extension past metal is greater than or equal to -2 and the adjacent enclosures are equal to 3.

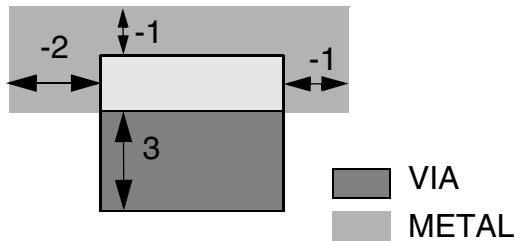
*Figure 10-15 Example 2*



## Example 3

The parameters are  $F2=5$ ,  $F1=-1$ ;  $Q2=3$ ,  $Q1=-1$ ,  $S2=1$ ,  $S1=-1$ , as shown in [Figure 10-16](#).

- The geometries do not meet  $F2=5$  and  $F1=-1$  because no two opposite enclosures are greater than or equal to 5.
- The geometries do not meet  $Q2=3$  and  $Q1=-1$  because no two opposite enclosures are greater than or equal to 3.
- The geometries do not meet  $S2=1$  and  $S1=-1$  because no two opposite enclosures are greater than or equal to 1.

**Figure 10-16 Example 3**

## **Output**

The via coverage output for a negative check does not have an impact on the output format.

## **Via Coverage Examples**

The `VIA_COVERAGE` command supports the checking of square vias and the `VIA_COVERAGE_OPTION_FILE` command supports the checking of rectangular vias.

### **VIA\_COVERAGE Syntax With Semicoverage**

You can specify the `VIA_COVERAGE` command with the semicoverage function as follows.

```
VIA_COVERAGE: via_layer_name Lf Lq [Ls] Cf Cq [Cs]
NETLIST_FORMAT: SPEF
NETLIST_TAIL_COMMENTS: YES
VIA_COVERAGE: VIA1 100 80 40 100 80 40
VIA_COVERAGE: VIA2 100 80 40 100 80 40
```

### **VIA\_COVERAGE Syntax Without Semicoverage**

For backward compatibility, you can specify the `VIA_COVERAGE` command without the semicoverage function.

```
VIA_COVERAGE: via_layer_name Lf Lq Cf Cq
NETLIST_FORMAT: SPF
NETLIST_TAIL_COMMENTS: YES
VIA_COVERAGE: VIA1 100 80 100 80
VIA_COVERAGE: VIA2 100 80 100 80
```

**VIA\_COVERAGE\_OPTION\_FILE - Syntax With Semicoverage**

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing =
FL,QL1,QL2,[SL1,SL2];Coverage=FC,
QC1,QC2,[SC1,SC2]}
(Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;
Landing=FL,QL1,QL2,[SL1,SL2];Coverage=FC,QC1,QC2,[SC1,SC2])
```

**VIA\_COVERAGE\_OPTION\_FILE - Syntax Without Semicoverage**

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing =
FL,QL1,QL2;Coverage=FC,QC1,QC2}
(Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing =
FL,QL1,QL2;Coverage=FC,QC1,QC2)
```

---

**Reading the Output Report**

The results from the `VIA_COVERAGE` functions appear under the heading of `VIA_COVERAGE_CODES` in the netlist and as a comment in the resistor element.

Coordinate locations of the reported vias are found in the SPF and SPEF outputs.

**Report Differences**

The report generated by StarRC output looks different depending on which via coverage commands you specify. Commands that include the semicoverage check generate more results.

**When Checking Semicoverage**

You can read the results of the semicoverage check in the netlist file. Reading the lines horizontally, find the specific check in the index line. The first letter indicates the via landing and the second letter indicates the via cover. As shown in [Example 10-1](#), the SS column indicates semivia landing and semivia coverage. Similarly, PP indicates partial via landing and partial via coverage and so on.

**Example 10-1 Semicoverage Codes for a Square or Rectangular Check With Semi Results**

```
// *index FF FQ FS FP QF QQ QS QP SF SQ SS SP PF PQ PS PP X_by_Y
// * 'FQ' stands for Full landing and Quarter coverage
// *0 0
// *1 0 1 1x1
// *2 0 1x1
// *3 0 1x1
// *4 0 0 0 0 1 0 1x1
// *5 0 1 0 1x1
// *6 0 1x1
// *7 0 0 0 0 0 0 0 0 0 1 0 1x1
// *8 0 0 0 1 0 1x1
// *9 0 1 0 0 1x1
// *10 0 1x1
// *11 0 0 0 0 0 0 0 0 0 1 0 1x1
// *12 0 1 0 1x1
// *13 0 1 0 0 1x1
// *14 0 1x1
// *15 0 0 0 0 0 0 0 0 0 1 0 1x1
// *16 1 0 1x1
```

The via coverage code shown in [Example 10-1](#) is for the following net:

```
*|NET FF OPF
*|S (FF:1 0.05 0.05) // $llx=-0.1 $lly=-0.1 $urx=0.2 $ury=0.2 $lvl=1
*|S (FF:2 0.05 0.05) // $llx=-0.1 $lly=-0.1 $urx=0.2 $ury=0.2 $lvl=2
R16 FF:1 FF:2 23.67 $vc=16 $a=0.01 $lvl=3
```

The coverage code table output when semi is not checked is similar, except the columns indicating semi via landing or semi via coverage are not included, as shown in [Example 10-2](#).

**Example 10-2 Semicoverage Codes for a Square or Rectangular Check With Semi Results**

```
// *index FF FQ FS FP QF QQ QS QP SF SQ SS SP PF PQ PS PP X_by_Y
// * 'FQ' stands for Full landing and Quarter coverage
// *0 0x0
// *1 0 1 1x1
// *2 0 1x1
// *3 0 0 1 0 1x1
// *4 0 1x1
// *5 0 1x1
// *6 0 1 0 1x1
// *7 0 1x1
// *8 0 0 0 1 0 1x1
// *9 1 0 1x1
```

The via coverage code shown in [Example 10-2](#) is for the following example net:

```
*|NET PP OPF
*|S (PP:1 1.85 -1.45) // $llx=1.78 $lly=-1.52 $urx=1.92 $ury=-1.38 $lvl=1
*|S (PP:2 1.85 -1.45) // $llx=1.78 $lly=-1.52 $urx=1.92 $ury=-1.38 $lvl=2
R1 PP:1 PP:2 23.67 $vc=1 $a=0.01 $lvl=3
```

In an SPF file, a parameter is added to the tail comment: \$vc. This is an integer that identifies the *coverage code*. The key for the codes is located in STAR\_DIRECTORY/via\_coverage\_codes.

StarRC supports both SPF and SPEF formats.

The X\_by\_Y column is written for via arrays. In [Example 10-2](#), the two resistors in the netlist are the same 2-by-5 via array, with the last one rotated 90 degrees.

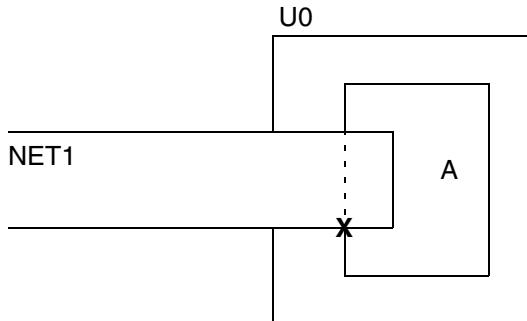
Net0 (noncritical) polygons are not used in the VIA\_COVERAGE calculation and should not be considered. All relevant neighbor polygons are considered when the context of the VIA is being analyzed.

## Long Ports

This feature is used to force \*|I reported port locations in both top level and cell-level (for INSTANCE\_PORT:NOT\_CONDUCTIVE) coordinate systems to the coordinates of the physical overlap of the top-level route with the instance port shape.

In the simplest case, the xy coordinate netlisted for the \*|I port simply the lower-left corner of the overlapping region. These regions are called PortUpContacts. See [Figure 10-17](#).

*Figure 10-17 PortUpContact Example*



The x in [Figure 10-17](#) represents the intersection coordinate for the NET1->U0/A connection. It is called a PortUpContact node.

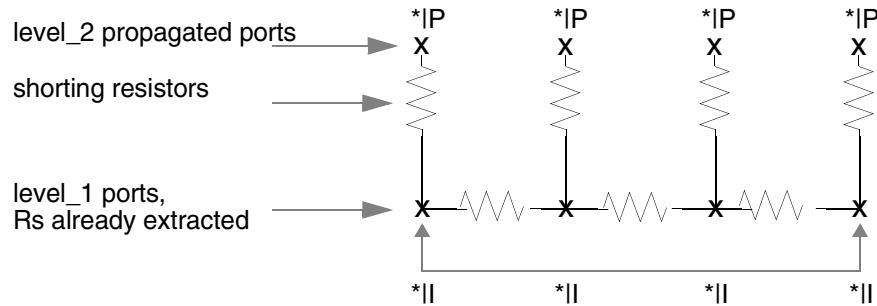
For longer “overlay”-type connections, multiple PortUpContact nodes are extracted. These nodes are located in the same way that subnodes on a net are normally extracted.

For multiple separated connections of a single net to a single port (multiple intersection), multiple PortUpContact nodes are extracted, one for each overlapping area.

When a propagated port is extracted, it is shorted to the lower level by a shorting resistor, but resistance extraction does not occur at that level. It is assumed that the extraction has already occurred at the lower level.

In [Figure 10-18](#), the level\_2 propagated port is netlisted with multiple \*|P, with global coordinates. The level\_1 ports are netlisted with multiple \*|I, with local coordinates for the level\_1 instance, and they are shorted to the level\_2 with small shorting resistors.

*Figure 10-18 Propagated Ports*



# 11

## Hercules GENDEV Device Extraction and Netlist Generation

---

This chapter describes how to use the Hercules GENDEV functionality in StarRC to extract and netlist designed nonstandard devices and nonstandard properties of standard devices.

This chapter contains the following sections:

- [Introduction](#)
- [Device Recognition and Syntax](#)
- [Scheme Extraction Functions](#)
- [Hercules LVS With GENDEV Devices](#)
- [Scheme Property Comparison Functions](#)
- [GENDEV Netlist Generation in StarRC](#)

---

## Introduction

In a transistor-level Hercules flow, StarRC is capable of creating a netlist for designed standard devices and standard device properties from the layout, as shown in the following table:

| Device                                  | Properties                                                                                                      |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| NMOS or PMOS<br>(3-, 4-, or 5-terminal) | length (l)<br>width (w)<br>source area (as)<br>drain area (ad)<br>source perimeter (ps)<br>drain perimeter (pd) |
| RESISTOR                                | length (l)<br>width (w)<br>resistance (r)                                                                       |
| CAPACITOR                               | length (l)<br>width (w)<br>capacitance (c)                                                                      |
| DIODE                                   | area (a)<br>junction perimeter (pj)                                                                             |
| INDUCTOR                                | inductance (l)                                                                                                  |
| NPN or PNP                              | area (a)                                                                                                        |

If you want to extract from the layout a nonstandard designed device not supported by a Hercules standard device extraction command, you can use the Hercules Generic Device command `GENDEV`. Also, if you want to extract a nonstandard device property for a standard device, (such as, DIODE length and width), you can use `GENDEV` instead of the normal Hercules device extraction command to define and extract this standard device with nonstandard properties.

---

## Device Recognition and Syntax

Generally speaking, most GENDEV devices can be extracted with GENDEV AUTO mode operation, wherein a device is recognized whenever exactly one polygon from each specified terminal layer interacts (through overlap, line touch, or point touch) with the specified device recognition layer. The Hercules GENDEV command has the following syntax:

```
GENDEV device_name term_layer1 ... [term_layerN]
{
 initialization_func = init_func
 extraction_func = extract_func
 recognition_layer = { layer_name }
 [processing_layers = { processing_layer1 ...
 processing_layerN }]
 [gendev_devtype = MOS | RES | CAP | IND | NPNODE |
 PNDIODE | NPNBJT | PNPNBT]
 [gendev_dev_port_name = TRUE | FALSE]
 [gendev_hier_proc = TRUE | FALSE]
 [ev_netlist_func = ev_netlist_func]
 [spice_netlist_func = spice_netlist_func]
} Output [Error_Output]
```

**Table 11-1** lists the parameters relevant to StarRC flows.

*Table 11-1 GENDEV Parameter Definitions*

| Parameter                   | Definition                                                                                                                                                                                                                                   |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>term_layer#</i>          | Layers that define the device terminals and touch or overlap the <i>recognition_layer</i>                                                                                                                                                    |
| <i>initialization_func</i>  | Scheme function that lists the properties to be extracted for the device                                                                                                                                                                     |
| <i>extraction_func</i>      | Scheme function wherein property values are calculated for each property defined in the initialization function                                                                                                                              |
| <i>recognition_layer</i>    | Layer with which all terminal layers must interact in the layout to define the device                                                                                                                                                        |
| <i>processing_layers</i>    | Layers that are neither terminal nor recognition layers but are used for calculating property values in the extraction function                                                                                                              |
| <i>gendev_devtype</i>       | Standard device type that the GENDEV device is intended to emulate; used for cases in which GENDEV is employed to extract nonstandard device properties for standard devices                                                                 |
| <i>gendev_dev_port_name</i> | Makes extracted device pin names consistent with the standard device defined by <i>gendev_devtype</i> ; set to <code>TRUE</code> if <i>gendev_devtype</i> is specified; if it is set to <code>FALSE</code> , pin names are T1, T2, and so on |

The following example shows a GENDEV command used to implement a standard designed resistor:

```
GENDEV customres resterm resterm {
 initialization_func = res-init
 extraction_func = res-extract
 recognition_layer = { resbody }
 gendev_devtype = RES
 gendev_dev_port_name = TRUE
} TEMP = res_layer
```

Hercules extracts a resistor device at any location in the layout where two polygons of layer *resterm* interact with one polygon of layer *resbody*.

For more information about GENDEV options, see the *Hercules Reference Manual*.

---

## Scheme Extraction Functions

For GENDEV extraction, Scheme functions serve two purposes. The initialization function specified by *initialization\_func* specifies the names of device properties that are associated with the generic device. An example of an initialization function that delineates length, width, area, and perimeter properties for a GENDEV resistor follows:

```
(define res-init (lambda ()
 (ev-dev-property-type-set "l" 'DOUBLE)
 (ev-dev-property-type-set "w" 'DOUBLE)
 (ev-dev-property-type-set "a" 'DOUBLE)
 (ev-dev-property-type-set "p" 'DOUBLE)
 (ev-dev-property-scale-factor-set "l" 'U)
 (ev-dev-property-scale-factor-set "w" 'U)
 (ev-dev-property-scale-factor-set "a" 'P)
 (ev-dev-property-scale-factor-set "p" 'U)
))
```

By default, Hercules calculates one-dimensional geometric properties in units of microns. However, StarRC netlists these properties in units of meters, according to standard SPICE conventions. Because of the inherently generic and nonstandard nature of GENDEV device properties, there is no clear way for either tool to know the appropriate geometric unit for a newly-defined property, unless you explicitly specify the unit.

The ev-dev-property-scale-factor Scheme function lets you resolve this by specifying an appropriate scale factor for converting each property value to the appropriate units for purposes of StarRC netlist generation. The syntax for the function is

```
(ev-dev-property-scale-factor-set prop_name prop_scale)
```

- prop\_name  
The name of property, enclosed in quotation marks.
- prop\_scale  
'U (1e-6) for one-dimensional geometric properties (l, w)  
'P (1e-12) for two-dimensional geometric properties (area)

A call to ev-dev-property-scale-factor should be specified for every geometric property defined in the initialization function, and these calls should be placed after the calls to ev-dev-property-type-set as in the previous example.

For GENDEV AUTO mode, the extraction function specified by extraction\_func is used purely to describe how values are to be calculated for each user-defined property delineated in the initialization function. Within this function, you calculate property values by utilizing Hercules-provided ev-measure Scheme routines to measure the geometric properties of the terminal, recognition, and processing layers. You then attach the calculated values to the

property names defined in the initialization function. An example of an extraction function that calculates values for resistor length, width, area, and perimeter properties follows:

```
(define res-init (lambda ()
 (let
 (
 (L 0)
 (W 0)
 (body-perim)
 (body-area)
)
 ; compute length and width
 (set! body-perim (ev-measure1 'PERIM resbody))
 (set! W (/ (ev-measure2 resbody 'ABUT_PERIM resterm) 2))
 (set! L (/ (- body-perim (* 2 W)) 2))
 (set! body-area (ev-measure1 'AREA resbody))

 ; assign values to the device properties
 (ev-dev-property-value-set customres "l" L)
 (ev-dev-property-value-set customres "w" W)
 (ev-dev-property-value-set customres "a" body-area)
 (ev-dev-property-value-set customres "p" body-perim)
)
))
```

## Hercules LVS With GENDEV Devices

Hercules GENDEV devices can be equated as standard devices, depending on the setting of the *gendev\_devtype* option. If *gendev\_devtype* is set to one of the Hercules standard device types, the EQUATE command for the device can be specified with a corresponding EQUATE standard device type. In the preceding example, the device would be equated with the following command:

```
EQUATE RES customres=customres A B { }
```

If the GENDEV device of interest is emulating a nonstandard device and no *gendev\_devtype* is specified, the device should be equated in the EQUATE command as a DEV device. (See the EQUATE command in the *Hercules Reference Manual*.)

When a standard device type (CAP, RES, DIODE, NMOS, PMOS, NPN, PNP, IND) is used in the EQUATE statement for the GENDEV device, Hercules LVS can perform device filtering and merged device property comparisons for standard properties in the normal manner. However, if device merging is enabled via the MERGE\_SERIES, MERGE\_PARALLEL, or MERGE\_PATHS options, and if nonstandard properties exist for the device, then you must provide Scheme routines that dictate how merged nonstandard properties should be calculated and compared during LVS.

You specify which nonstandard properties are to be compared by using the `check_user_properties` EQUATE option:

```
check_user_properties = { nonstandard_property_list }
```

If device merging is enabled, each nonstandard property to be compared must be accompanied by a corresponding Scheme routine that dictates how to calculate the merged property and to perform the property comparison. This is specified inside the EQUATE command with the `comp_prop_func` option:

```
comp_prop_func[non_standard_prop] = scheme_routine
```

If the GENDEV device is emulating a standard device, you can use the filtering options for the standard device by configuring the EQUATE command accordingly. However, if the GENDEV device is not emulating a standard device (in which case EQUATE DEV is used), no standard filtering functionality is available. Again, you can provide a Scheme routine to specify how device filtering should be performed for such an instance. This routine is specified within the EQUATE command as

```
filter_func = scheme_routine
```

Detailed information about writing a Scheme filtering function is available in the *Hercules Reference Manual*.

The following GENDEV custom resistor example enables series and parallel device merging during LVS and compares the merged property `a` during LVS. This requires you to supply a Scheme routine to calculate the nonstandard `a` property for the merged device. Because the example emulates a standard RES device, no Scheme filtering routines are necessary. A complete EQUATE statement for the GENDEV custom resistor example would look like this:

```
EQUATE RES customres=customres A B BULK {
 merge_series = true
 merge_parallel = true
 check_properties = {l, w}
 check_user_properties = {a}
 comp_prop_func[a] = res-area-comp
}
```

Note that in this example, “`l`” and “`w`” are standard Hercules-recognized RES properties whereas “`a`” is not. Thus, Scheme routines are necessary to instruct Hercules how to compare merged property `a` during LVS. Note also that `COMPARE_PROPERTIES = TRUE` must be set in the Hercules `COMPARE` command for property comparisons to occur at all.

---

## Scheme Property Comparison Functions

When you require nonstandard device properties for devices that are merged during LVS, you must supply a Scheme routine to perform the merged property comparison. The body of the routine is used to achieve a single merged value for the schematic merged device and the layout merged device and to then compare these two values after application of the specified tolerance bounds. The routine should be coded to return a value of #t when it considers the property comparison to have passed, and a value of #f when the property comparison failed. Hercules lsh reads this return value and then reports that the property comparison either passed or failed.

The following is a Scheme routine that performs the property comparison for the GENDEV *customres* device property *a* follows. More information about writing a Scheme merged property comparison function is available in the *Hercules Reference Manual*.

```
(define res-area-comp (lambda (sch-id lay-id prop)
 (let (
 (sch-type #f) (lay-type #f)

 ;; Get values for the specified property
 (sch-prop (lvs-inst-prop-get sch-id prop))
 (lay-prop (lvs-inst-prop-get lay-id prop))
 (sch-name (lvs-inst-name-get sch-id))
 (lay-name (lvs-inst-name-get lay-id))
 (is-sch-merged (lvs-inst-is-member sch-id))
 (is-lay-merged (lvs-inst-is-member lay-id))

 (pos-tolerance (car (lvs-inst-tolerance-get lay-id prop)))
 (neg-tolerance (cadr (lvs-inst-tolerance-get lay-id prop)))
 (abs-tolerance (lvs-inst-tolerance-is-absolute lay-id prop))
 (sch-merged-area 0)
 (lay-merged-area 0)
)

 ;; Make sure property values exist for this property
 ;; on both devices
 (if (not sch-prop)
 (return (format "~s: No schematic property '~s' found"
 sch-name prop))
)

 (if (not lay-prop)
 (return (format "~s: No layout property '~s' found"
 lay-name prop))
)
 ;; Check whether merging took place in schematic
 (if (not is-sch-merged)
 (set! sch-merged-area (car sch-prop))
 (set! sch-merged-area (sum-list (sch-prop)))
)
```

```

;; Check whether merging took place in layout
(if (not is-lay-merged)
 (set! lay-merged-area (car lay-prop))
 (set! lay-merged-area (sum-list lay-prop)))
)

;; Calculate tolerance boundaries
(if (not abs-tolerance)
 (set! pos-tolerance (* pos-tolerance sch-merged-area))
)

(if (not abs-tolerance)
 (set! neg-tolerance (* neg-tolerance sch-merged-area))
)

(if (> (- lay-merged-area sch-merged-area) pos-tolerance)
 (format
 "\t~a=~a: Property ~a mismatch. ~s != ~s"
 sch-name lay-name prop sch-merged-area lay-merged-area)
)

(if (<= (- sch-merged-area lay-merged-area) neg-tolerance)
 #t
 (format
 "\t~a=~a: Property ~a mismatch. ~s != ~s"
 sch-name lay-name prop sch-merged-area lay-merged-area)
)
)
)
)
```

# GENDEV Netlist Generation in StarRC

StarRC netlists all property names and values defined by the initialization and extraction functions for GENDEV devices. In SPF/STAR/NETNAME netlist format, these devices are netlisted in the instance section by use of standard SPICE cards (R for resistor, C for capacitor, and so on) according to the setting of `gendev_devtype` in the Hercules netlist. If no `gendev_devtype` is specified, the instance is netlisted with an x-card, indicating a call to a separate .SUBCKT circuit definition.

The instance name is always followed by the device node names, the model name for the device, and the list of properties defined for the device. For example, if gendev\_devtype=RES is used, the *customres* resistor occurs in the StarRC SPF netlist as follows:

RR1 R1:A R1:B customres l=2.5u w=1u a=2.5p p=7u

If `gendev_devtype` and `gendev_dev_port_name` are not specified in the Hercules runset, the device is netlisted by StarRC as follows:

XG1 G1:T1 G1:T2 customres l=2.5u w=1u a=2.5p p=7u

The latter example requires a .SUBCKT *customres* definition to define the contents of the device for simulation.

One limitation is that if XREF:COMPLETE is specified, StarRC only netlists device properties defined for the device in the schematic netlist, unless the properties are standard properties by Hercules and StarRC for the emulated standard device.

Note:

To ensure that all such standard properties are netlisted for Hercules standard devices, use the following StarRC option:

```
NETLIST_PASSIVE_PARAMS: YES | NO
```

## Part II: Process Modeling

---

---



# 12

## Process Modeling Methodology

The nxtgrd file describes the relationship between the design layout and the manufacturing process, which is essential information for calculating accurate parasitics. You can model your own processes and create an nxtgrd file using the provided grdgenxo tool, or you can obtain an nxtgrd file from a foundry.

This chapter contains the following sections:

- [Flows for Process Characterization](#)
- [The Interconnect Technology Format File](#)
- [The Mapping File](#)
- [The grdgenxo Command](#)

---

## Flows for Process Characterization

Accurately determining the capacitance and resistance of a circuit requires detailed modeling of the process technology. The StarRC tool uses pattern matching to analyze a design by comparing the layout to a set of primitive structures whose parasitics have been previously simulated. Those reference parasitics are contained in a process characterization database called the New Xtraction Generic Regression Database or nxtgrd file. The file is also known as a TCAD GRD file because of the StarRC command that references it.

Foundries often develop nxtgrd files for their processes and distribute them in encrypted form.

A process description language known as the Interconnect Technology Format (ITF) is used to describe the fabrication process. ITF statements must be contained in a file called the ITF file. You can use the ITF file in two ways:

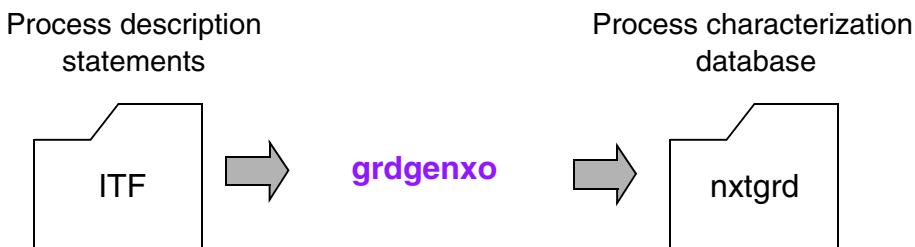
- Run the grdgenxo tool to process the ITF file and create a database file for later use in the StarRC tool.
- Use the StarRC tool to read the ITF file directly. This operation is supported only for capacitance extraction in transistor-level flows.

---

### The grdgenxo Flow

The grdgenxo tool generates the process database file from the ITF file, as illustrated in [Figure 12-1](#).

*Figure 12-1 Process Database File Generation*



The grdgenxo tool uses an internal field solver operating on an extensive set of primitive structures to generate the nxtgrd file. The nxtgrd file contains capacitance, resistance, and layer information. The ITF statements are also included in the file.

To use an nxtgrd file in extraction, specify the file name in the StarRC command file with the `TCAD_GRD_FILE` command.

---

## The Direct ITF Flow

To experiment with process changes in transistor-level flows, you can specify an ITF file directly by using the `ITF_FILE` command in the StarRC command file. The StarRC tool automatically uses a field solver to analyze the capacitance of the process structures.

The direct ITF flow does not use the `grdgenxo` tool and does not generate or use an `nxtgrd` file. This operation is supported only for capacitance extraction in transistor-level flows.

In the direct ITF flow, you cannot use the following commands in the StarRC command file:

- `TCAD_GRD_FILE`
- `FS_EXTRACT_NETS`
- `3D_IC`
- `NETLIST_POWER_FILE`
- `TEMPERATURE_SENSITIVITY`
- `EXTRACTION` (except when used with the `c` option)
- `EXTRA_GEOMETRY_INFO` (except when used with the `NONE` option)

---

## The Interconnect Technology Format File

An ITF file defines a cross section profile of the process. The file contains an ordered list of conductor and dielectric layer definition statements. The layers are defined from the topmost dielectric layer to the bottommost dielectric layer, excluding the substrate. The ITF parameters are specified layer by layer in a way that is consistent with the physical process.

The lowest layer in the ITF cross section must be a dielectric layer. The `SUBSTRATE` keyword refers to a special conductor whose top plane is at 0 height, underneath the lowest dielectric layer. Do not define `SUBSTRATE` in the ITF file.

The via layers are defined relative to valid conducting layers.

The heights of the conductors and dielectrics are determined by the order in which they are specified and by the thicknesses of the lower layers. When you are specifying a new conductor or dielectric layer, the bottom plane of that layer is exactly the top plane of the lowest dielectric layer unless a `MEASURED_FROM` statement is included to explicitly specify the location of the bottom plane. The lowest dielectric—the lowest physical layer—listed in the ITF file is automatically measured from the `SUBSTRATE` layer. A fully planar process, in which the process cross section does not contain any vertically intersecting conductors at different heights, is the simplest model. For an example, see [Fully Planar Process ITF Example](#).

Using the ITF file to describe a process technology eliminates the need to check parameter sets for consistency. If the sheet resistance parameters for a process must be altered, it is best to regenerate the nxtgrd file. Regenerating the file with updated sheet resistances is very fast, because the grdgenxo tool uses the capacitance solution from the original run.

## Creating an ITF File

The following procedure provides an overview of ITF file creation. For more information about specific ITF statements, see [Chapter 16, “ITF Statements.”](#)

1. Specify the TECHNOLOGY statement:

```
TECHNOLOGY = process_name
```

The TECHNOLOGY statement is mandatory and should precede all other statements, but it does not need to be the first line. The *process\_name* argument becomes the file name of the nxtgrd file.

2. (Optional) Specify process description information. The keywords, listed in [Table 12-1](#), have names that suggest their intended uses. You must follow the rules for the values allowed.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

*Table 12-1 Process Description Keywords*

| Keyword         | Type   | Description                    |
|-----------------|--------|--------------------------------|
| PROCESS_FOUNDRY | String | Identifies the foundry         |
| PROCESS_NODE    | Float  | Represents the process node    |
| PROCESS_TYPE    | String | Identifies the process type    |
| PROCESS_VERSION | Float  | Represents the process version |
| PROCESS_CORNER  | String | Names the process corner       |

3. (Optional) Specify the global temperature and the relative permittivity of the background dielectric:

```
GLOBAL_TEMPERATURE = temp_value
BACKGROUND_ER = relative_permittivity
```

The background permittivity fills the cross section with material of the given dielectric constant to an infinite height. Layer-specific permittivities specified in the ITF file override the global background dielectric. The default for the background dielectric is 1.0.

4. Define the basic layer characteristics for all of the conductor, dielectric, and via layers.

You must specify the following layer characteristics:

- Thickness of each conductor and dielectric layer
- Minimum width and spacing of each conductor layer (design rule spacing)
- Resistivity of each conductor layer
- Permittivity of each dielectric layer
- Resistivity information of each via layer
- Connectivity information of each via layer

5. Specify additional ITF statements to model process effects.

The following example shows a basic ITF file:

```
TECHNOLOGY = SIMPLE
DIELECTRIC TOP { THICKNESS = 3.600 ER = 3.9 }
CONDUCTOR M2 {
 THICKNESS = 0.250 WMIN = 0.5
 SMIN = 0.5 RPSQ = 0.05 }
DIELECTRIC D3 { THICKNESS = 0.300 ER = 3.9 }
CONDUCTOR M1 {
 THICKNESS = 0.212 WMIN = 0.5
 SMIN = 0.5 RPSQ = 0.05 }
DIELECTRIC D2 { THICKNESS = 0.200 ER = 4.2 }
CONDUCTOR POLY{
 THICKNESS = 0.100 WMIN = 0.3
 SMIN = 0.3 RPSQ = 10.0}
DIELECTRIC D1 { THICKNESS = 0.300 ER = 3.9 }
VIA via1 { FROM=M1 TO=M2 RHO=0.263 }
VIA polyCont { FROM=POLY TO=M1 RHO=0.352 }
VIA subCont { FROM=SUBSTRATE TO=M1 RHO=0.500 }
```

---

## ITF Files for Transistor-Level Flows

Transistor-level ITF files differ from gate-level ITF files because transistor-level extraction requires connectivity information down to the diffusion or substrate layers.

A transistor-level ITF file can describe a planar process, in which a single polysilicon (poly) layer is used for both the gate and field poly structures. Alternatively, the process might be nonplanar, using separate gate and field poly layers.

Via layers can be used only to connect two conducting layers. Therefore you must separate a poly contact from a substrate contact or a diffusion contact.

Covertical layers (such as gate and field poly layers) should be vertically overlapping. If the bottom of the field poly layer is higher than the top of the gate poly layer, the two poly layers are not connected to each other. In this case you must create a dummy via layer to connect the two poly layers.

---

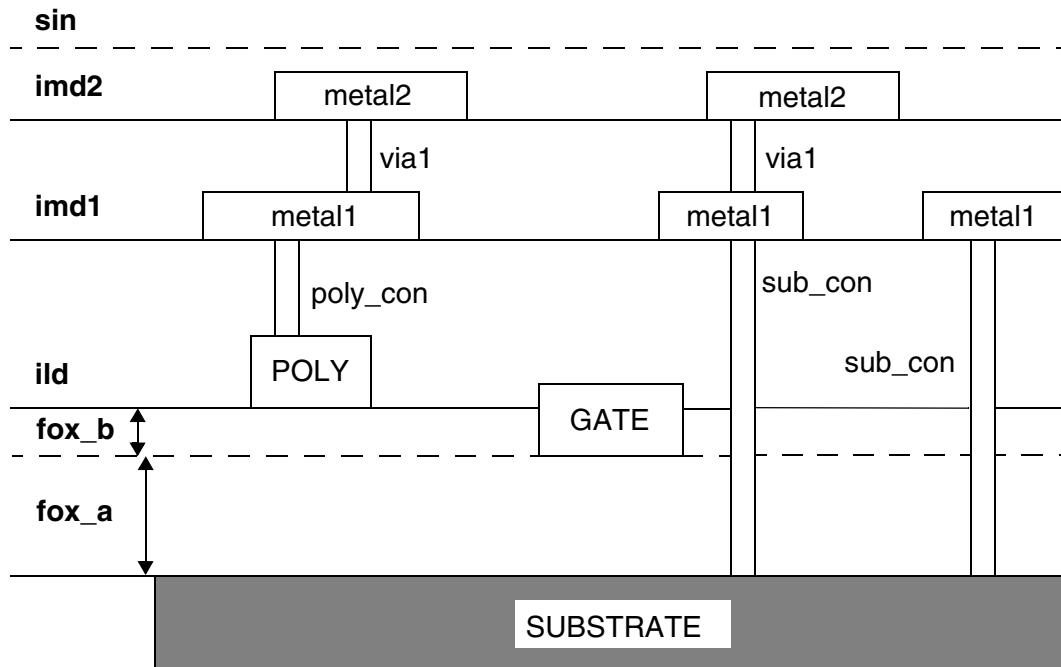
## ITF File Example

The following example shows the ITF statements that describe the cross section shown in [Figure 12-2](#). For more examples, see [Chapter 14, “ITF Examples.”](#)

```
TECHNOLOGY=add4_2m1p

DIELECTRIC sin {THICKNESS=0.70 ER=7.9}
DIELECTRIC imd2 {THICKNESS=1.00 ER=4.2}
CONDUCTOR metal2 {THICKNESS=0.50 WMIN=0.35 SMIN=0.35 RPSQ=0.07}
DIELECTRIC imd1 {THICKNESS=1.05 ER=4.2}
CONDUCTOR metall1 {THICKNESS=0.35 WMIN=0.30 SMIN=0.30 RPSQ=0.08}
DIELECTRIC i1d {THICKNESS=0.70 ER=4.1}
CONDUCTOR poly {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_b {THICKNESS=0.10 ER=3.9}
CONDUCTOR gate {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_a {THICKNESS=0.25 ER=3.9}

VIA via1 {FROM=metall1 TO=metal2 AREA=0.09 RPV=1}
VIA poly_con {FROM=poly TO=metall1 AREA=0.04 RPV=12}
VIA sub_con {FROM=SUBSTRATE TO=metall1 AREA=0.04 RPV=16}
```

*Figure 12-2 Process Cross Section*

## The Mapping File

The mapping file maps layers in the design database to layers in the nxtgrd file and is required for every StarRC run. You can create the mapping file manually or through the StarRC GUI.

The following guidelines apply to mapping files:

- A database layer can map to only one process layer.
- Multiple database layers can map to the same process layer.
- Database layers that map to the same process layer should not contain overlapping structures. Modify the LVS runset to use Boolean operations to avoid overlap and remove unused layers from the StarRC mapping file.
- Every logically connected database layer must be mapped in this file, even if the layer is derived or used only for intermediate connections with no real physical significance.
- In LEF/DEF flows, every layer defined in the technology LEF file—including vias—must be mapped.

Syntax conventions for mapping files are as follows:

- Commands are not case-sensitive. The convention in this user guide is to use lowercase for mapping file commands.
- Layer names are case-sensitive.
- Characters after an asterisk are considered to be part of a comment.

For more information about mapping files, see [Chapter 17, “Mapping File Commands.”](#) The following is an example of a mapping file:

```

conducting_layers * Note the use of a single poly layer for all gates
 fpoly Poly
 Bulk SUBSTRATE
 Poly Poly
 Rpoly Poly
 Rndiff Od
 Ndif Od
 Nwell SUBSTRATE
 ngate1 Poly
 Ngate Poly
 Pgate Poly
 Nsd Od
 Psd Od
 Pdif Od
 Met4 Metal4
 Met3 Metal3
 Met2 Metal2
 met1 Metal1
via_layers
 Diffcnt Odcont
 Plycnt polyCont
 diffcnt odCont
 m3via via3
 m2via via2
 mvia via1
marker_layers
 met1_pin
 met2_pin
 met3_pin
remove_layers
 cont
 diffcnt
ignore_cap_layers
 dngate ngate ngate1
 pdif psd
 ndif rndiff nsd
 Ngate Nsd L=0.1
 Pgate Psd L=0.1
 Nsd SUBSTRATE L=0.1
 Psd SUBSTRATE L=0.1

```

---

## The grdgenxo Command

The `grdgenxo` command is used at the linux command line to run the `grdgenxo` tool. The primary use of the tool is to create an `nxtgrd` file from an ITF process description file. You can also create TLUPlus files and perform a limited set of updating operations on existing `nxtgrd` files.

---

### Syntax of the `grdgenxo` Command

```
grdgenxo
 [-add_sf]
 [-encrypt]
 [-format format_file]
 [-help]
 -input input_file
 [-inc]
 [-itf2itf]
 [-itf2TLUPlus]
 [-link_device_models]
 [-nxtgrd2itf]
 [-jpg jpg_file]
 -output output_file
 [-parse]
 [-res_update]
 [-usage]
 [-Version]
```

---

| Argument                                | Description                                                                                                                                                                                                                       |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-add_sf</code>                    | Reprocesses an existing <code>nxtgrd</code> file with a new half-node scaling factor.                                                                                                                                             |
| <code>-encrypt</code>                   | Encrypts the ITF statements and includes them in the <code>nxtgrd</code> file.                                                                                                                                                    |
| <code>-format <i>format_file</i></code> | Uses the specified format file when creating TLUPlus models.                                                                                                                                                                      |
| <code>-h(elp)</code>                    | Shows command-line options.                                                                                                                                                                                                       |
| <code>-i(nput) <i>input_file</i></code> | Specifies the input file, which can be an ITF file for <code>nxtgrd</code> file generation, or an <code>nxtgrd</code> file for rescaling with the <code>-add_sf</code> option.                                                    |
| <code>-inc</code>                       | Performs an incremental <code>nxtgrd</code> file update. The <code>grdgenxo</code> tool recognizes the modified layers in the ITF file during successive runs and regenerates the capacitance models for the changed layers only. |
| <code>-itf2itf</code>                   | Encrypts selected fields in an ITF file.                                                                                                                                                                                          |

| Argument                               | Description                                                                                                                                                                    |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -itf2TLUplus                           | Generates a TLUPlus model from an ITF file. If used, this argument must be specified first.                                                                                    |
| -link_device_models                    | Reuses device models when they are composed of different conductors with identical capacitive properties.                                                                      |
| -nxtgrd2itf                            | Encrypts selected fields in an ITF file.                                                                                                                                       |
| -o( <i>output</i> ) <i>output_file</i> | Specifies an output file name.                                                                                                                                                 |
| -parse                                 | Parses the ITF file to check for syntax errors, without starting an nxtgrd file generation. If no errors are found, the grdgenxo tool issues a confirmation message and stops. |
| -res_update                            | Allows updating of resistance parameters from commands such as the RPSQ and RHO_VS_WIDTH_AND_SPACING commands without processing through the field solver.                     |
| -usage                                 | Shows command-line options.                                                                                                                                                    |
| -Version                               | Includes the version number in the output file.                                                                                                                                |

---

## Partially Encrypting ITF Information

To encrypt selected fields in an ITF file, use the `-itf2itf` and `-nxtgrd2itf` options with the `grdgenxo` command. Select a field to encrypt by prefixing the field with the pound sign (#). In this way, you can protect sensitive process data while modifying nonprocess fields such as metal fill ratios. The `grdgenxo` tool issues a parsing error if the encrypted fields are modified.

**Note:**

StarRC versions earlier than G-2012.06 cannot read partially-encrypted nxtgrd files.

In the following example of an ITF file, the # character indicates that the THICKNESS, WMIN, SMIN, and RPSQ values are to be encrypted.

```
CONDUCTOR M3 {
 THICKNESS= #0.81
 WMIN= #0.61
 SMIN= #0.51
 RPSQ= #0.661
 FILL_RATIO=0.5
 FILL_WIDTH=0.5
 FILL_SPACING=0.25
}
```

The following example shows the appearance of the ITF statements in the output nxtgrd file. Some are encrypted and some are unencrypted.

```
CONDUCTOR M3 {
 THICKNESS= @46e54570d7fccee0c7
 WMIN= @74fdf83f98c5d8d288
 SMIN= @22d884f1d2b9fe84c2
 RPSQ= @507cd4ceede9ddf1ece7
 FILL_RATIO=0.5
 FILL_WIDTH=0.5
 FILL_SPACING=0.25
}
```

The presence of encrypted fields in the ITF file suppresses technology information and layer data in the nxtgrd header and produces a simple layer list instead of a full ITF command list in the TLUPlus header.

## Incremental grdgenxo runs

The incremental option of the `grdgenxo` command lets you update an nxtgrd database using a modified ITF file. You supply the latest ITF file in the working directory and specify the `-inc` option. The `-inc` option distinguishes the run as an incremental generation and updates only the differences found between the original ITF file and the new ITF file.

You can run the `grdgenxo` tool incrementally on a previous run directory only when using the same version of the `grdgenxo` tool, and only if the changes you have made to the ITF file do not affect the capacitance tables. Changes that modify conductor resistance values are allowed.

In a distributed processing environment, the `grdgenxo` tool performs additional checking to verify that all processors are using the same ITF file.

## Limitations

The following conducting layer options have only localized effect on capacitance values. Therefore, changes can be incrementally updated into an existing nxtgrd database. Changes to the following keywords in the `CONDUCTOR` statement are supported:

```
CAPACITIVE_ONLYETCH
ETCH
ETCH_VS_WIDTH_AND_SPACING
ETCH_VS_WIDTH_AND_SPACING: CAPACITIVE_ONLY
ETCH_VS_WIDTH_AND_SPACING: ETCH_FROM_TOP
MEASURED_FROM
SIDE_TANGENT
SMIN
THICKNESS
WMIN
```

Global layer options have extensive effects on capacitance values. Changes to the following keywords in the `CONDUCTOR` statement are not supported:

```
DROP_FACTOR
FILL_RATIO
FILL_SPACING
FILL_WIDTH
FILL_TYPE
GATE_TO_CONTACT_SMIN
```

Any changes to dielectric layers have large effects on capacitance values. Do not use the incremental option for the following types of changes:

- Changes to the total number of conductors or dielectrics
- Removal of a conductor

Inappropriate `WMIN` and `SMIN` values specified in the ITF file might cause unwanted opens or shorts of the neighboring layers by applying the etch values provided in the table. A message is issued during the `grdgenxo` run for `WMIN` violations. Reporting of `SMIN` violations is off by default; to enable it, use the `REPORT_SMIN_VIOLATION: YES` command.

---

## Using the `-res_update` Option to Update an `nxtgrd` File

You can update the resistance parameters of an `nxtgrd` file by using the `-res_update` option of the `grdgenxo` command in conjunction with a new ITF file. An example of the command is as follows:

```
grdgenxo -res_update new_itf_file -i old_nxtgrd_file -o new_nxtgrd_file
```

This method might save substantial computing time. The version number and time stamp is not updated; the original values are kept.

The following limitations apply:

- You cannot use the `-res_update` option with `grdgenxo` command options other than the `-encrypt`, `-input` and `-output` options.
- You can update existing resistance parameters, but you cannot introduce new resistance parameters into an `nxtgrd` file.
- You cannot change any StarRC commands that affect capacitance.

---

## Generating TLUPlus Models

TLUPlus models describe advanced process effects that can be used by the parasitic extractor in the Synopsys place-and-route tool for modeling. The TLUPlus model file is in a binary format containing an ASCII header section that lists the input files used.

You can obtain a TLUPlus model file from the foundry. Alternatively, you can create one using the grdgenxo tool. An example of the command is as follows:

```
% grdgenxo -itf2TLUPlus -input ITF_file -output TLU_file [-format ffile]
```

The grdgenxo tool searches for the STAR-RC2-TCAD license to enable TLUPlus output. If a STAR-RC2-TCAD license is found, a Galaxy-Common license is not necessary. If a STAR-RC2-TCAD license is not found, the grdgenxo tool looks for a Galaxy-Common license. If neither license is found, an error message is issued.

A directory called *technology\_name.TLUPlus* contains the temporary results. The input ITF file and the format file are saved in that directory with the .itf and .format extensions, respectively. Subsequent grdgenxo runs on the same database check the input files for changes from previous runs and issue an error if there are any differences.

The following ITF keywords are not supported for creating TLUPlus models:

- DROP\_FACTOR
- DROP\_FACTOR\_LATERAL\_SPACING
- ETCH\_VS\_CONTACT\_AND\_GATE\_SPACINGS

The grdgenxo tool uses the following defaults for TLUPlus models:

- Units of fF, ohm, and micron for the Milkyway technology file (the place-and-route tools convert units as necessary)
- Nominal operating conditions for capacitance tables names
- Capacitance tables dimension of 5x16 for width versus spacing
- Capacitance tables grid points that are multiples of minimum width and spacing values

You need to change the defaults specified in the TLUPlus files if

- You want to create TLUPlus tables for minimum or maximum operating conditions.
- You want to change the dimension of the capacitance table. Larger tables do not necessarily give you greater accuracy; smaller tables reduce runtime.
- You want to use nonuniform width and spacing values. This might improve accuracy by reducing the need for interpolation or extrapolation.

Dimensions of resistance tables and width points, if applicable, are determined automatically based on the information in the ITF file.

To change the defaults, you can create a format file as shown in [Table 12-2](#). You do not have to specify parameters for every layer, nor do you need to specify the `WIDTH` and `SPACING`

values if you want to change only the capacitance table dimensions while keeping the default width and spacing intervals.

*Table 12-2 Format File Example*

| File content          | Definition                                                                                                 |
|-----------------------|------------------------------------------------------------------------------------------------------------|
| CAP_UNIT 1e-15        | Capacitance units, in femtofarads. Do not change.                                                          |
| RES_UNIT 1            | Resistance units in ohms. Do not change.                                                                   |
| OPCOND NOM            | Operating condition; can be MIN, NOM, or MAX                                                               |
| LAYER poly            | Layer name from the ITF file (for example, poly, metal1, metal2) to which the subsequent statements apply. |
| NUMWIDTH 3            | Number of capacitance table width points for the named layer; can be 2 to 16 inclusive.                    |
| NUMSPACING 3          | Number of capacitance table spacing points for the named layer; can be 2 to 16 inclusive.                  |
| WIDTH 0.15 0.3 0.45   | Values of the capacitance table width points.                                                              |
| SPACING 0.15 0.3 0.45 | Values of the capacitance table spacing points.                                                            |
| LAYER metal2          | Layer name to which the subsequent statements apply.                                                       |
| NUMWIDTH 5            | Number of capacitance table width points for the named layer.                                              |
| NUMSPACING 16         | Number of capacitance table spacing points for the named layer.                                            |



# 13

## Process Characterization

---

This chapter describes how to model specific process effects using StarRC commands and ITF statements.

This chapter contains the following sections:

- FinFET Modeling and Extraction
- Through-Silicon Via Extraction
- Double or Multiple Patterning Technology
- Conductor Layer Thickness Variation
- Bottom Conductor Thickness Variation
- Conductor Sheet Zones
- Tall Contact Via Modeling
- Bridge Via Modeling
- Gate-To-Diffusion Capacitance Extraction
- Conformal Dielectrics
- Conductor Cutting Through Dielectric
- Covertical Conductors
- Conductor Drop Factor

- Double-Polysilicon Process
- Layer Etch
- Overlapping Wells
- Damage Modeling
- Temperature Derating
- Half-Node Scaling
- Via Merging
- Diffusion Resistance
- User-Defined Diffusion Resistance

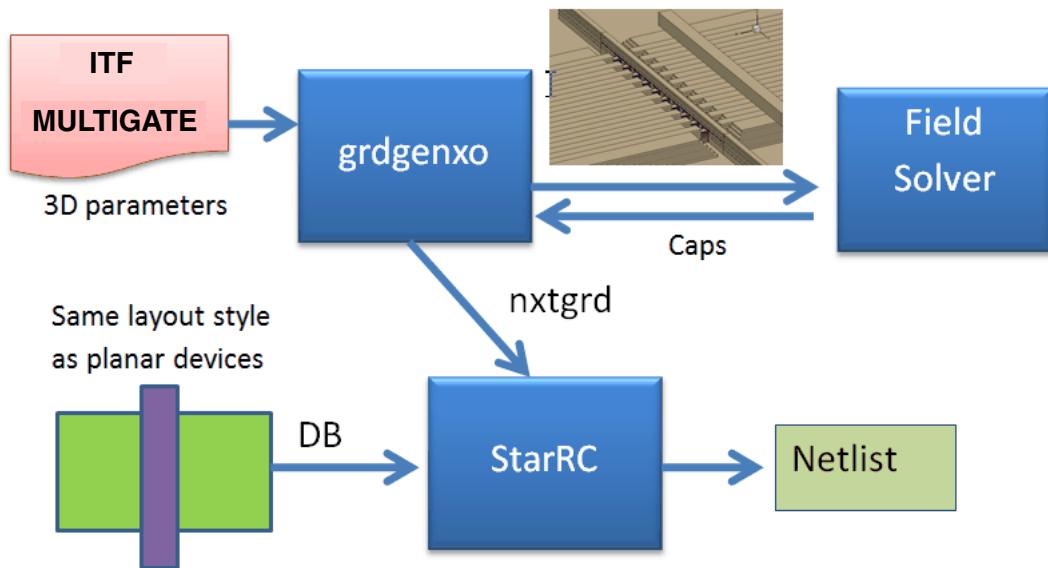
## FinFET Modeling and Extraction

StarRC extraction supports FinFET devices. FinFET device layout is similar to the planar device layout; in both cases the device layout includes the intersection of a diffusion layer with polysilicon.

For FinFET modeling, the ITF file contains a `MULTIGATE` block which describes how to map the device layout to 3-D geometries. The `grdgenxo` tool creates models of FinFET devices based on the `MULTIGATE` statement in the ITF file. These models are characterized using a 3-D field solver, and parameterized capacitance tables are built and stored in the `nxtgrd` file. The StarRC tool uses the capacitance tables during scanband extraction and field-solver extraction. [Figure 13-1](#) shows the FinFET extraction flow.

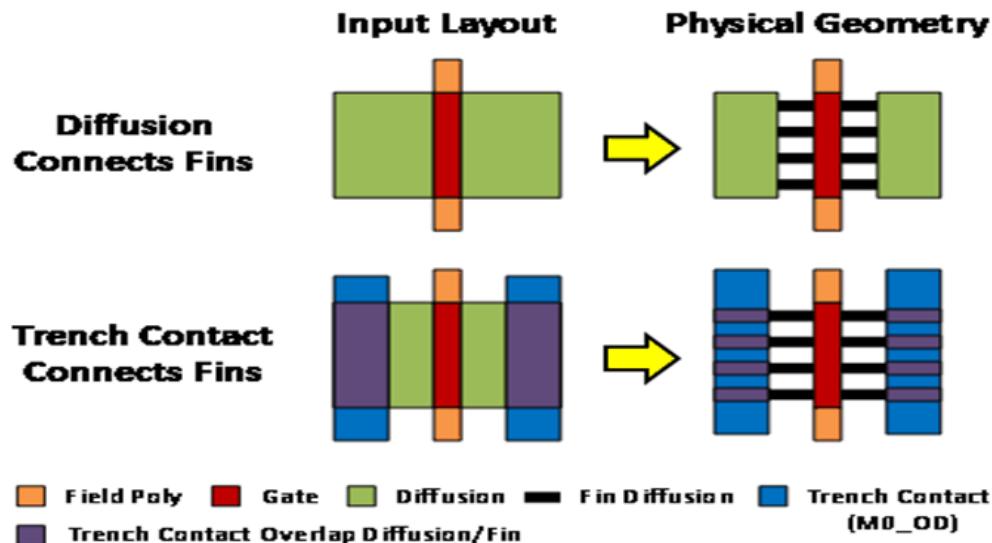
To enable the FinFET extraction flow, add `MULTIGATE_MODELS: YES` to the StarRC command file.

*Figure 13-1 FinFET Extraction Flow*



The StarRC tool recognizes FinFET devices and converts them to realistic FinFET physical geometries, as shown in [Figure 13-2](#).

*Figure 13-2 FinFET Layout to Physical Geometry*



## FinFET ITF Statement Guidelines

### Design Guidelines

- Set RAISED\_DIFFUSION\_GROWTH to a value greater than zero.
- When using RAISED\_DIFFUSION\_GROWTH, set the raised diffusion region in the diffusion layer.
- Set RAISED\_DIFFUSION\_GROWTH to less than one half the FIN\_SPACING value.
- Use trench contacts with FinFETs. Via-style diffusion contacts are not supported.
- Represent FinFET poly and diffusion as noncovertical. To understand this requirement, see [Process Description Requirements](#).

Some FinFET processes have highly nonlinear gate resistances due to the complex 3-D geometry of the gate conductors. To model this nonlinearity, provide the resistance per square (RPSQ) values in the RPSQ\_VS\_SI\_WIDTH\_AND\_LENGTH table.

## Process Description Requirements

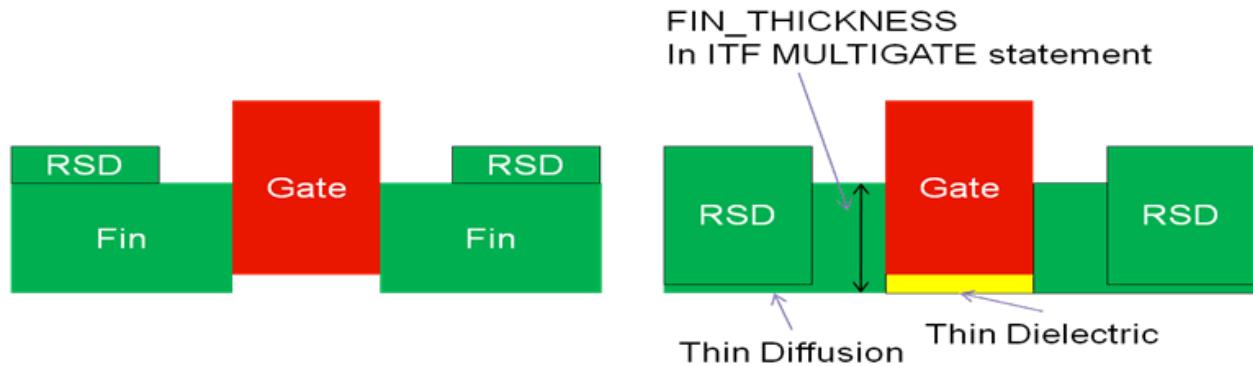
FinFET processes intrinsically require covertical poly and diffusion layers because the gate extends down the sides of the transistor channel. However, the StarRC tool assumes that poly and diffusion are not covertical. Therefore, you must describe the FinFET poly and diffusion as non covertical. Use `RAISED_DIFFUSION_THICKNESS` to specify the full height of the diffusion conductor and to model the capacitive effects.

To model a FinFET process using `RAISED_DIFFUSION_THICKNESS`, describe a very thin diffusion layer with a large `RAISED_DIFFUSION_THICKNESS` value, as shown in [Figure 13-3](#). Specify a thin dielectric extending just above the thin diffusion. Set the thickness of the gate poly to its height minus the thickness of the thin dielectric.

Using `RAISED_DIFFUSION_THICKNESS` in this way makes it impossible to separately model etch effects on the raised diffusion (`RAISED_DIFFUSIONETCH`) as shown in [Figure 13-4](#). If these effects are significant, accuracy is reduced. To minimize these effects, model raised diffusion etch approximately, using an intermediate etch value applied to the full height of the diffusion conductor.

Using `RAISED_DIFFUSION_THICKNESS` affects conformal dielectrics. Specify any conformal dielectric to be in contact with its associated conductor. If the diffusion is made much thinner, any associated conformal dielectrics must begin at a correspondingly lower level, which might affect some processes.

*Figure 13-3 FinFET Vertical Cross Section and Layer Descriptions*



*Figure 13-4 With RAISED\_DIFFUSIONETCH: FinFET Vertical Cross Section and Layer Descriptions*



## FinFET Capacitances

Figure 13-5 and Figure 13-6 show the capacitance components of a FinFET device.

Figure 13-5 FinFET Capacitance Components, Side View

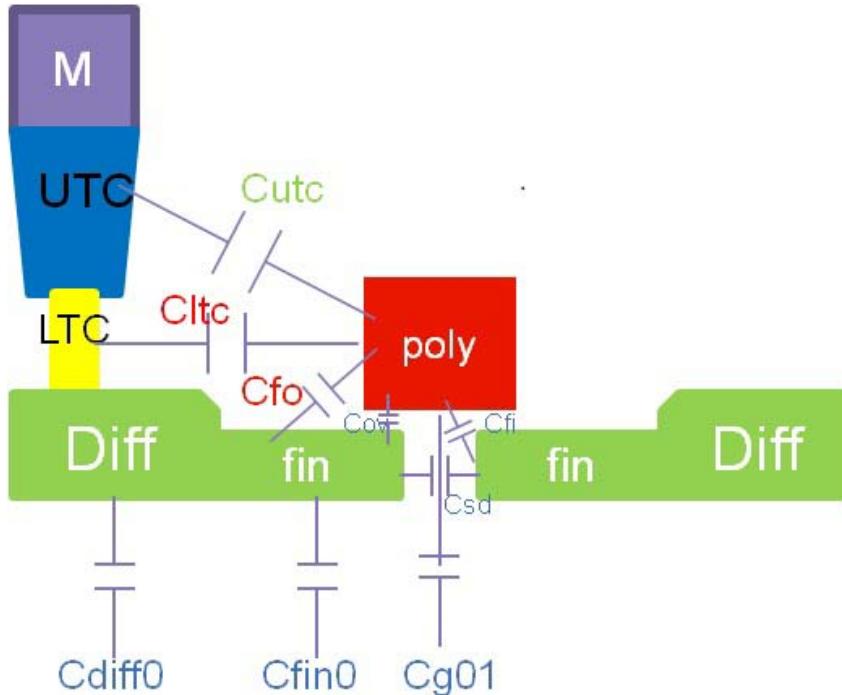
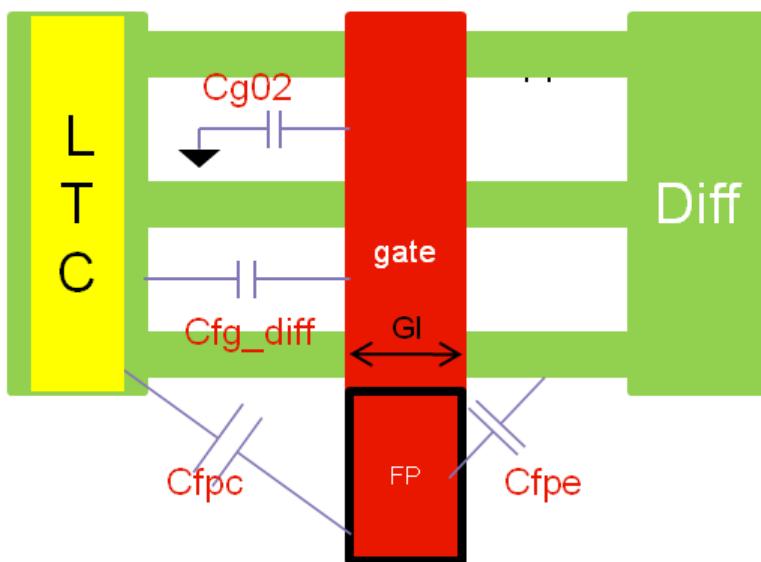


Figure 13-6 FinFET Capacitance Components, Top View



The grdgenxo tool uses an internal 3-D field solver to build parameterized tables for each of the capacitance components. The tool uses the ITF `MULTIGATE` statement to construct the fin geometry, and assumes the `FIN_LENGTH`, `FIN_WIDTH`, and `FIN_SPACING` parameters are constant for each device type.

Four components are characterized for the capacitance tables:

- $C_{gd}$  (gate to diffusion)

This includes the capacitance from the gate to the top of the diffusion ( $C_{fo}$  in [Figure 13-5](#)) and from the gate to the fin and diffusion sidewalls ( $C_{fg\_diff}$  in [Figure 13-6](#)). It does not include the overlap capacitance between the gate and the diffusion ( $C_{ov}$  in [Figure 13-5](#)) or the capacitance from the gate to the diffusion through the channel ( $C_{fi}$  in [Figure 13-5](#)).

- $C_{g0}$  (gate to substrate)

This includes the gate to substrate capacitance ( $C_{go2}$  in [Figure 13-6](#)) but not the gate to substrate capacitance through the channel ( $C_{go1}$  in [Figure 13-5](#)).

- $C_{gc}$  (gate to contact)

This is the same as the gate to lower trench contact capacitance  $C_{ltc}$  in [Figure 13-5](#). The upper trench contact capacitance  $C_{utc}$  is extracted outside of the grdgenxo tool.

- $C_{fpe}$  (field poly to diffusion)

This is labeled as  $C_{fpe}$  in [Figure 13-6](#).

You can provide a table to represent gate-to-diffusion capacitance inside the channel ( $C_{fi}$ ) by using the `GATE_TO_DIFFUSION_CHANNEL_CAP` statement in the ITF file.

If the StarRC command file includes the `IGNORE_GATE_CHANNEL_CAPACITANCE: NO` command, the total gate-to-diffusion capacitance ( $C_f$ ) is extracted by the field solver and no  $C_{fi}$  capacitance subtraction occurs, regardless of whether a  $C_{fi}$  table exists.

Alternatively, you can calculate  $C_{fo}$  by first having the field solver extract the total capacitance  $C_f$ , then subtracting the  $C_{fi}$  value (interpolated from the  $C_{fi}$  table) from  $C_f$ . To enable this calculation method, use the `IGNORE_GATE_CHANNEL_CAPACITANCE: YES` command in the StarRC command file. If a  $C_{fi}$  table does not exist, the command has no effect.

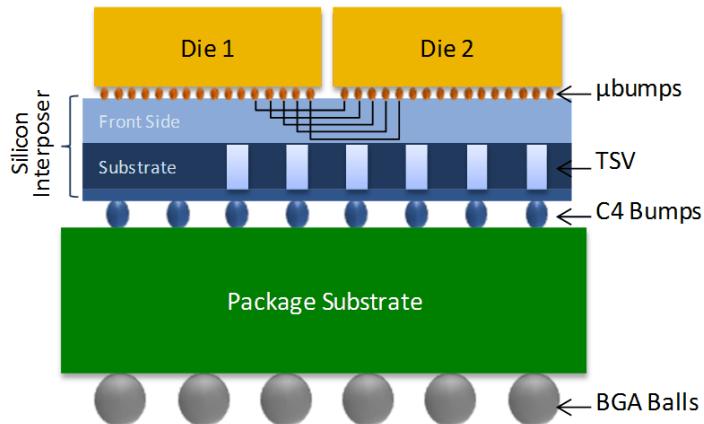
The `IGNORE_GATE_CHANNEL_CAPACITANCE` command requires nxtgrd files generated by StarRC version H-2013.06-SP1 or later.

## Through-Silicon Via Extraction

Wafer-level integration refers to a packaging technique in which integrated circuits are connected to each other and to a package through a thinned silicon substrate called an interposer. A via that extends completely through the interposer layer is called a through-silicon via (TSV) and connects the die to the package.

**Figure 13-7** shows a silicon interposer attached to two die and a package. Conductive bumps on the die surfaces connect to routing metal wires on the front side of the interposer. The TSVs connect the interposer front side to the backside metal layers, which are in turn connected to the package through metal bumps (labeled as C4 bumps in the figure).

*Figure 13-7 Interposer Structure Showing TSVs and Microbumps*



You can extract TSVs that are defined as either cells or vias with the GDS, LEF/DEF, and Milkyway flows. You can also extract microbumps in this manner.

The following StarRC commands and ITF statements are related to TSV extraction:

- [3D\\_IC](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [TSV\\_ CELLS](#)
- [OPERATING\\_FREQUENCY](#)
- [TSV](#)

---

## Microbump Modeling

The microbump is a type of via layer. The microbump connects to the top metal layer and a thin (1 nm) pseudo-metal layer at the top of the stack. The pseudo-metal layer is defined with `LAYER_TYPE=BUMP`.

Define a microbump in the ITF file as shown in the following example:

```
VIA <ubump_name> {
 FROM=<RDL Metal Layer>
 TO=<Pseudo Metal layer>
 RPV=<R>
}
CONDUCTOR <name> {
 THICKNESS= 0.001
 WMIN=<value> SMIN=<value>
 RPSQ=0.00000001
 LAYER_TYPE=BUMP
}
```

---

## 3D-IC Flow with Cell-Defined TSVs and microbumps

You can extract TSVs and microbumps defined as cells in the GDS, LEF/DEF, and Milkyway flows.

### GDS Flow

You can extract TSV and microbumps as devices in the GDS flow if you define them as cells in the runset.

When you set `SKIP_PCELLS` on TSV and microbump devices, StarRC does not extract anything inside these devices. The parasitics inside these devices are part of the device characteristics.

The StarRC tool extracts only to the pins of the TSV or microbump cells. These pins appear as instance ports in the netlist file and the corresponding cell instances are listed in the Instance Section.

### LEF/DEF and MilkyWay Flow

The TSV cell flow is similar to the `SKIP_CELLS` flow; the TSV cell source is defined by the database and the `TSV_CELLS` command. StarRC supports net-to-port TSV cells from the IC Compiler tool. The StarRC tool supports merging the same net in the front and back. To keep the one net to two ports output in the netlist, use the `SHORT_PINS:MIXED` command.

If you do not specify the `3D_IC_SUBCKT_FILE` for a TSV cell, that TSV cell is translated as a skip cell and listed in the instance section. StarRC outputs netlists for skip cells. The StarRC tool determines the parasitics from the `nxtgrd` file.

If you specify the `3D_IC_SUBCKT_FILE` for a TSV cell, StarRC extraction replaces the TSV cells with the specified subcircuit file and outputs a single netlist. The microbump cells are processed the same as the TSV cells.

---

## 3D-IC Flow with Via-Defined TSVs and microbumps

You can extract TSVs and microbumps defined as vias in the GDS, LEF/DEF, and Milkyway flows.

### GDS Flow

Because the interposer does not contain devices, nets have different top-level pins attached to them, which causes short errors in the LVS report. To fix the shorts, add a pseudo-resistor device at the ports to separate them from the net in the interposer. To remove the pseudo-resistor device, specify the device in the `3D_IC_FILTER_DEVICE` command, which flattens it in the netlist.

If you do not specify the `3D_IC_SUBCKT_FILE` for TSV vias, StarRC extracts a two-port model and outputs a single netlist. The parasitics are determined by the factor in the `nxtgrd` file.

If you specify the `3D_IC_SUBCKT_FILE` for TSV vias, StarRC replaces the TSV vias with the subcircuit file and outputs a single netlist.

Follow the same process for microbump vias.

### LEF/DEF and MilkyWay Flow

The TSV and microbump vias can be defined by the database. StarRC extraction merges the nets in the front and back. To keep the same net and the two ports in the netlist output, set `SHORT_PINS:MIXED`.

If the `3D_IC_SUBCKT_FILE` is not provided for a TSV via, StarRC extracts the two-port model and outputs a single netlist. The parasitics are determined by the factor in the `nxtgrd` file.

If the `3D_IC_SUBCKT_FILE` is provided for a TSV via, StarRC replaces the TSV vias with the subcircuit file and outputs a single netlist.

The microbump via can be replaced by using the `3D_IC_SUBCKT_FILE` command.

---

## Double or Multiple Patterning Technology

Advanced processes sometimes require a double or multiple patterning lithography process, which uses two or more masks to print closely spaced patterns. Layers that do not meet the minimum spacing requirements are split into separate masks, referred to as colors. The exposures from the masks are overlaid to print the single layer. The StarRC tool can extract parasitics created by multiple patterning. The terms double and multiple patterning are used interchangeably in this user guide.

StarRC double-patterning extraction provides the following features:

- Exactly simulates the mask color misalignment and reports the final parasitic results
- Supports simultaneous multicorner technology
- Supports both gate-level and transistor-level designs
- Creates the nxtgrd database using the ITF process technology file along with statistically determined shift-impact measurements to compensate for misalignment shifts
- Reads color information for specified nets and reports parasitics for these nets, to allow scenario evaluation before colorizing the entire design (precolor flow)

The following sections describe how to use StarRC to extract parasitics from designs using double-patterning technology:

- [Extraction for Double-Patterning Processes](#)
- [Estimating Parasitics for Critical Nets Using the Precolor Flow](#)

---

## Extraction for Double-Patterning Processes

In StarRC, you model double-patterning misalignment effects by using dielectric constant changes, which are controlled by the following two commands:

- ER\_VS\_SI\_SPACING
  - Add the `ER_VS_SI_SPACING` layer definition statement to the ITF process technology file.
- DPT
  - Add `DPT: YES` to the StarRC command file.

For layout-versus-schematic (LVS) tools, color information resides as layers in the design database. For example, you define a layer as `M1_color1` and `M1_color2` in a Hercules or IC Validator XTR view. The LVS deck can include these color layers in the output. The interface between LVS tools and StarRC enables StarRC to recognize the color layers.

For Milkyway tools, use the following commands to map color layers:

- DPT\_COLOR\_GDS\_FILE: *file\_name.GDS*  
This command specifies the name of the GDSII file containing the color layer polygons.
- DPT\_COLOR\_GDS\_LAYER\_MAP\_FILE: *file\_name*  
This command specifies the name of the file that maps the color layers in the design database to the GDSII layers.

To map the color layers in the design physical database to an nxtgrd file modeled layer:

1. Create a section named `color_layers` in the mapping file. For more information, see the [color\\_layers](#) description in [Chapter 17, “Mapping File Commands.”](#)
2. Add each database layer name in the `DPT_COLOR_GDS_LAYER_MAP_FILE` to the `color_layers` section in the mapping file.

---

## Estimating Parasitics for Critical Nets Using the Precolor Flow

Use the precolor flow to determine parasitics for different layout options. Upstream tools create the `DPT_COLOR_GDS_FILE` using a precolor attribute in the design database.

The precolor flow is built around the simultaneous multcorner analysis feature. To enable a precolor flow with multiple corners,

1. Define a typical nxtgrd corner in the corners file using the `NOMINAL` keyword.
2. Use the `NOMINAL` corner when extracting the same color polygons across all the corners.
3. Create multiple nxtgrd corners for conductors on different masks or unspecified colors.

---

## Conductor Layer Thickness Variation

Conductor thickness variation can change circuit behavior dramatically. The extent of the variation depends on the technology node, foundry, and fabrication processes. For example, chemical-mechanical polishing (CMP) is a process technique in which a wafer is physically polished to partially remove deposited layers. CMP processes produce surfaces that are globally flat but that exhibit complex local thickness variations.

The dishing effect is a phenomenon in which the center of a feature is thinner than the edges. The change in conductor cross section affects the resistance and capacitance of the structure. The amount of dishing is dependent on the density of a layer, the spacing from a feature to its neighbors, and the feature width.

StarRC performs extraction on designs with thickness variation by first determining the thickness variation for a feature, then computing resistances and capacitances based on the thickness variation.

You can model the process variation by specifying one or more of the following effects in the ITF file:

- The variation of thickness with density
- The weighting factors for different density boxes
- The variation of thickness with width and spacing of conductors
- The orders of density and width for modeling thickness variation using a polynomial equation and the coefficients of the polynomial equation

---

## Single-Box Method

In this method, you choose a single box size and specify the variation of thickness of the conductor in a table. The box is always a square. The maximum size of the box is 500 microns. This method is simple and does not require an exhaustive characterization like the multiple box method. If specified alone for a conductor in the process file, then it does not model local density thickness variation. For linear table modeling, specify a multipoint thickness variation versus density table in the process file.

The `THICKNESS_VS_DENSITY` statement uses the following syntax:

```
THICKNESS_VS_DENSITY [RESISTIVE_ONLY | CAPACITIVE_ONLY]
 {(D1 R1) (D2 R2) (D3 R3) (D4 R4) ... }
```

*D<sub>1</sub>, D<sub>2</sub>, D<sub>3</sub>, D<sub>4</sub>* represent the density values. *R<sub>1</sub>, R<sub>2</sub>, R<sub>3</sub>, R<sub>4</sub>* represent the relative change in thickness. Negative R indicates a decrease in thickness and vice versa. Even though R can be any number between -1 and 1, a number close to 1 or -1 is unrealistic. R cannot be -1.

The `THICKNESS_VS_DENSITY` variation affects both resistance and capacitance. However, if the coefficients for resistance and capacitance are different, then use the `RESISTIVE_ONLY` and `CAPACITIVE_ONLY` options.

If no `DENSITY_BOX_WEIGHTING_FACTOR` is specified, the default density box size of 50 microns is used with a weighting factor of unity.

The following combinations are not supported:

- A `THICKNESS_VS_DENSITY` table cannot be combined with `THICKNESS_VS_DENSITY RESISTIVE_ONLY` or `THICKNESS_VS_DENSITY CAPACITIVE_ONLY`.
- The `DENSITY_BOX_WEIGHTING_FACTOR` statement cannot be used without a `THICKNESS_VS_DENSITY` table.

The following example illustrates the use of the THICKNESS\_VS\_DENSITY statement:

```
CONDUCTOR metal3 {
 THICKNESS_VS_DENSITY {
 (0.1, -0.1) (0.2 0.1) (0.3 0.2) (0.4 0.3) } THICKNESS=0.5
 SMIN=0.2 WMIN=0.22 RPSQ=0.06 }
```

## Multiple-Box Method

In this method you specify the multiple-box size and its weighting factor for effective density calculation. This method requires that you characterize the wafer in greater detail than the previous method. This method is preferred when the single-box method does not reflect the process behavior. The density box is a square. The maximum size of the density box is 50 microns and the maximum number of boxes is 5.

The following specification is an example that uses four density boxes:

```
THICKNESS_VS_DENSITY { (D1 R1) (D2 R2) (D3 R3) (D4 R4) }
DENSITY_BOX_WEIGHTING_FACTOR { (S1 W1) (S2 W2) (S3 W3) (S4 W4) }
```

The parameters are as follows:

- S1 to S4 are integers that represent the sizes of the five density boxes in microns. The values must be larger than zero and less than 500 and must be ordered from smallest to largest.
- W1 to W4 are weighting factors. If W is set to 0, then the pair (S W) is ignored. The values must fall between -10 and +10.
- R1 to R4 are the relative thickness values; negative R indicates a decrease in thickness and vice versa.
- D1 to D4 are the density values, or the fraction of the box area occupied by the features. The values must be between 0 and 1.

## Calculation of Effective Density

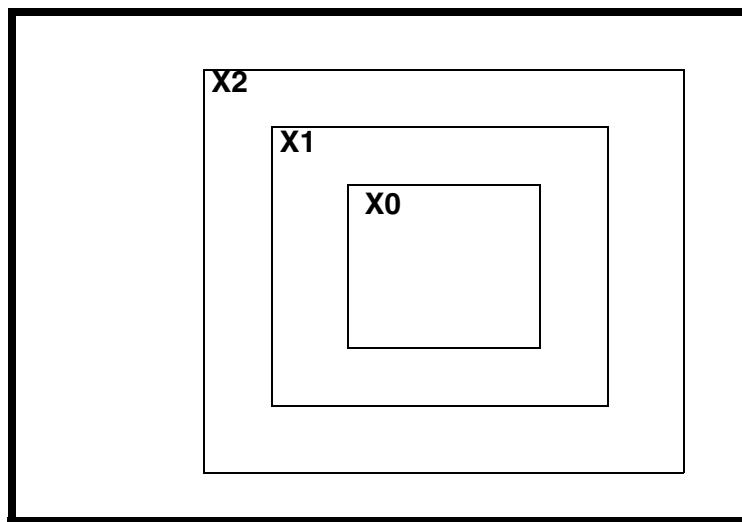
All density calculation is based on drawn width and spacing. When multiple density boxes are specified, the effective density is calculated as shown in [Figure 13-8](#).

*Figure 13-8 Effective Density Calculation*

$$D_{\text{eff}} = \sum_{i=0 \text{ to } i=5} D(X_i) * W(X_i)$$

D(X<sub>i</sub>) - Density of box X<sub>i</sub>

W(X<sub>i</sub>) - Weighting factor of box X<sub>i</sub>



[Figure 13-8](#) shows three boxes: X0, X1, and X2. StarRC calculates the density for each box for a given segment. The effective density is computed as shown in the equation in [Figure 13-8](#). After computing the effective density, the variation in thickness is computed based on the THICKNESS\_VS\_DENSITY table. Both resistance and capacitance for a given segment are calculated after thickness modification is taken into account.

Make sure to choose a weighting factor in such a way that calculated effective density is less than unity.

If the computed density exceeds the limit of the density table, the closest density value is picked to calculate the thickness variation.

The following is an example:

```
CONDUCTOR METAL3 {
 THICKNESS = 0.35
 WMIN = 0.2
 SMIN = 0.21
 DENSITY_BOX_WEIGHTING_FACTOR {
 (10, 1) (30, 0.23) (20, 0.29)
 (40, 0.18) (50, -0.12) }
 THICKNESS_VS_DENSITY { (0.1, -0.1) (0.2 0.1) }
}
```

---

## Width and Spacing-Dependent Thickness Variation

In this method, the variation of thickness as a function of the width of a conductor and the relative spacing to its neighbor is modeled. This thickness variation can be either negative or positive. As can be noted, this is a very local phenomenon and is independent of the density box. If specified with either single or multiple boxes, this thickness variation is computed independently of the density box.

The effective thickness is calculated with the following equation:

$$T = T_{nom} \times (1 + RTf(Deff) + RTf(W, S) + RTf(SiW))$$

where

- $T_{nom}$  is the nominal thickness specified in the ITF file.
- $RTf(Deff)$  is the relative change in thickness based on density.
- $RTf(W,S)$  is the relative change in thickness based on width and spacing.
- $RTf(SiW)$  is the relative change in thickness based on silicon width.

The resistance and capacitance is computed after effective thickness is computed. You can model this variation in a process file with the `THICKNESS_VS_WIDTH_AND_SPACING` statement for a conductor.

The `THICKNESS_VS_WIDTH_AND_SPACING` variation affects both resistance and capacitance. However, if the coefficients for resistance and capacitance are different, use the `RESISTIVE_ONLY` and `CAPACITIVE_ONLY` options.

---

## Bottom Conductor Thickness Variation

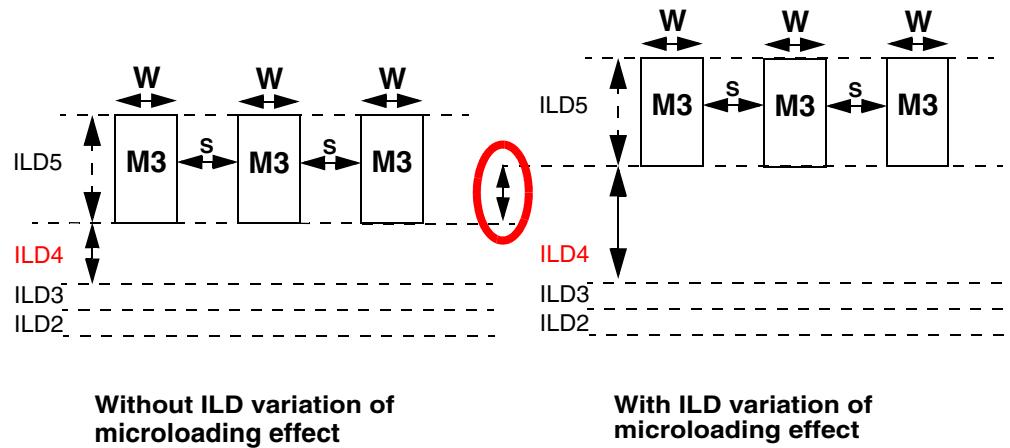
The bottom conductor thickness variation, or microloading effect, is caused by variation in the trench depth etching process for thin lines. Microloading effects increase as wires become thinner and more closely spaced. Trench depth variation affects the thickness of the interconnect and the separation between metal layers, so it affects both resistance and capacitance.

Modeling of the microloading effect can be done in different ways, depending on the data available from foundries.

The `BOTTOM_THICKNESS_VS_SI_WIDTH` statement is used to model microloading as a function of silicon width after etch. The `ILD_VS_WIDTH_AND_SPACING` statement is different from the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement because it models intralayer dielectric (ILD) variation as a function of drawn conductor width and spacing, as opposed to fabricated width and spacing.

[Figure 13-9](#) shows the ILD4 layer thickness varying as a result of microloading on the metal3 conductor.

*Figure 13-9 Model Microloading in the Form of ILD*



When the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement is used, the thickness of the conductor can change (increase or decrease). In contrast, when the `ILD_VS_WIDTH_AND_SPACING` statement is used, the conductor thickness remains constant, but the location of the conductor moves up or down, depending on the direction of dielectric variation. This difference might have a significant effect on coupling capacitance to neighboring conductors.

The following restrictions apply to the ILD variation function. Errors are reported to standard output on the terminal screen if any of the following conditions occur:

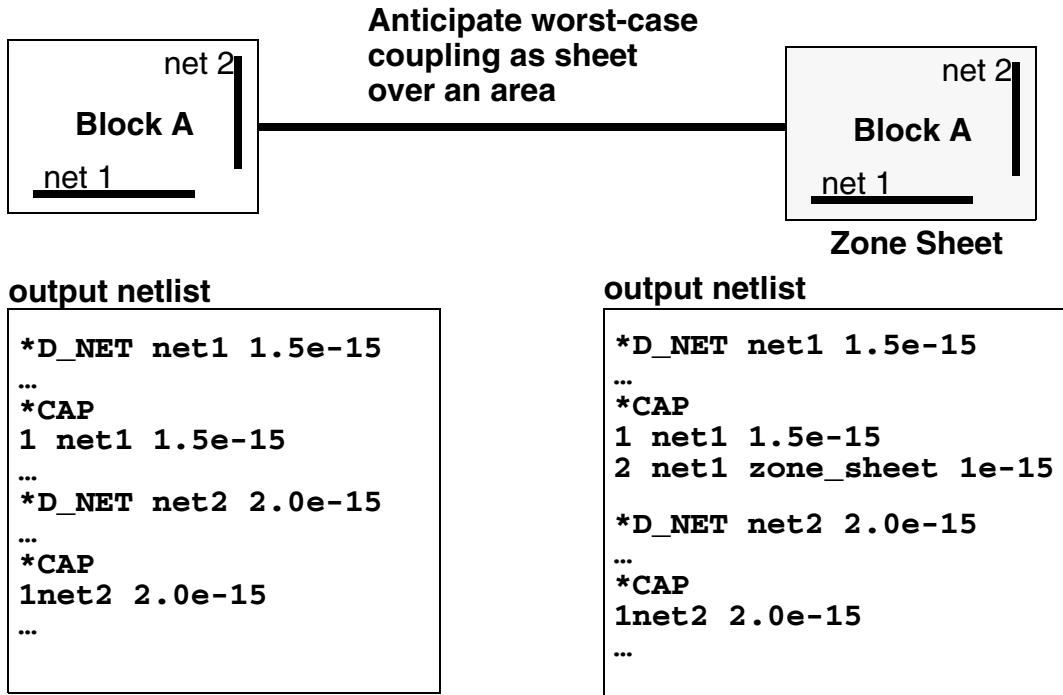
- The ILD variation is specified for a dielectric layer that does not exist directly below a conductor.
- The ILD variation specified is greater than 0.2 or less than -0.2.
- The ILD variation table is specified in the same `CONDUCTOR` block as a `BOTTOM_THICKNESS_VS_SI_WIDTH` table.

## Conductor Sheet Zones

A sheet zone is a location in which you model the insertion of a metal sheet over a specific area as shown in [Figure 13-10](#). This allows you to measure the coupling capacitance of a given metal sheet, which has a user-defined net name. You can also provide a suffix to the netname. By using sheet zone modeling, you can either specify one sheet or many thin strips of metal with this same command interface.

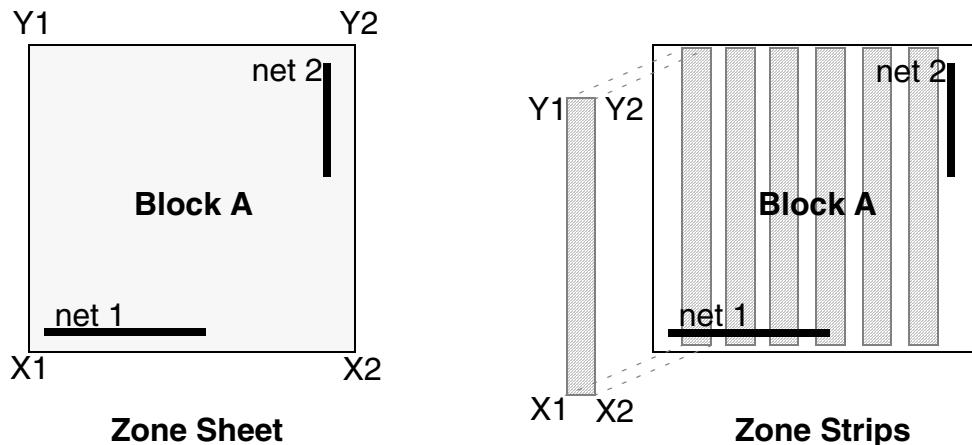
You must ensure that any added sheet zone resides in an area that does not cause metal shorts.

*Figure 13-10 Sheet Zone Modeling*



Specify the `METAL_SHEET_OVER_AREA` command in the command file followed by the metal layer name and four coordinates. These coordinates pinpoint the x- and y-locations of a single sheet as shown in [Figure 13-11](#). Then specify the `SHEET_COUPLE_TO_NET` command to designate a unique net name or name prefix. You have the option to specify the `SHEET_COUPLE_TO_NET_LEVEL` command, which enables a layer-level number to be output as the net name suffix in the output netlist.

*Figure 13-11 Specifying a Sheet Zone or Sheet Strips*



The following example shows the order of the commands for a single zone sheet:

```
METAL_SHEET_OVER_AREA METAL2 0 0 100 100
METAL_SHEET_OVER_AREA METAL2 200 200 400 400
METAL_SHEET_OVER_AREA METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

The following example shows the order of the commands for several sheet strips:

```
METAL_SHEET_OVER_AREA METAL2 0 5 10 10
METAL_SHEET_OVER_AREA METAL2 8 13 10 10
METAL_SHEET_OVER_AREA METAL2 16 21 10 10
METAL_SHEET_OVER_AREA METAL2 23 28 10 10
METAL_SHEET_OVER_AREA METAL2 31 36 10 10
METAL_SHEET_OVER_AREA METAL2 38 43 10 10
SHEET_COUPLE_TO_NET: zone_strips
SHEET_COUPLE_TO_NET: YES
```

The following limitations accompany the metal sheet capability:

- You must verify that the metal sheet zones you specify do not cause a short.
- The prefix or root net name specified with the `SHEET_COUPLE_TO_NET` command must be unique.

## Tall Contact Via Modeling

StarRC supports processes that include tall contact vias (vias with high aspect ratio) that connect device-level layers (such as diffusion or polysilicon layers) to higher-level metal layers. Memory processes often include tall vias.

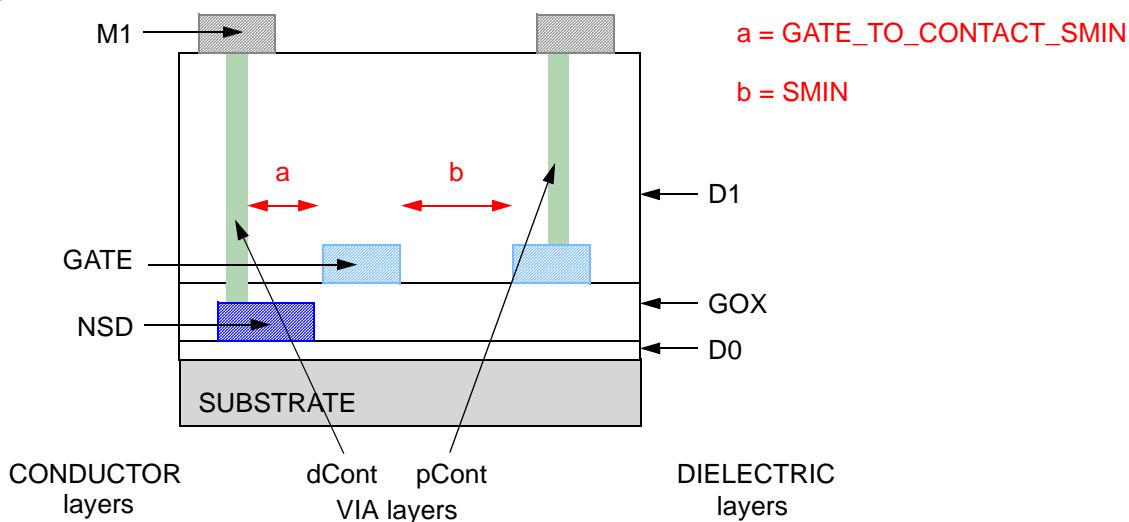
Guidelines for modeling tall vias in the ITF file are as follows:

- Define a tall contact via within a `VIA` block.
- Specify the `LAYER_TYPE=GATE` option in the `CONDUCTOR` block that defines a polysilicon layer in close proximity to the via.
- Use the `GATE_TO_CONTACT_SMIN` statement in the `CONDUCTOR` block for the polysilicon layer.

In the following ITF example, illustrated in [Figure 13-12](#), vias pCont and dCont are tall contacts. The `CONDUCTOR` statement that defines the GATE conductor contains both the `SMIN` and `GATE_TO_CONTACT_SMIN` statements as well as the `LAYER_TYPE=GATE` specification.

```
TECHNOLOGY = xtor
CONDUCTOR M1 {THICKNESS=0.50 WMIN=0.50 SMIN=0.45 RPSQ=0.062}
DIELECTRIC D1 {THICKNESS=1.8 ER=3.9}
CONDUCTOR GATE {THICKNESS=0.25 LAYER_TYPE=GATE WMIN=0.35
 GATE_TO_CONTACT_SMIN=0.40 SMIN=0.90 RPSQ=3.200}
DIELECTRIC GOX {THICKNESS=0.48 ER=5.0}
CONDUCTOR NSD {THICKNESS=0.40 WMIN=1.00 SMIN=0.35 RPSQ=10.00}
DIELECTRIC D0 {THICKNESS=0.50 ER=3.9}
VIA pCont {FROM=GATE TO=M1 RHO=0.352}
VIA dCont {FROM=NSD TO=M1 RHO=0.500}
```

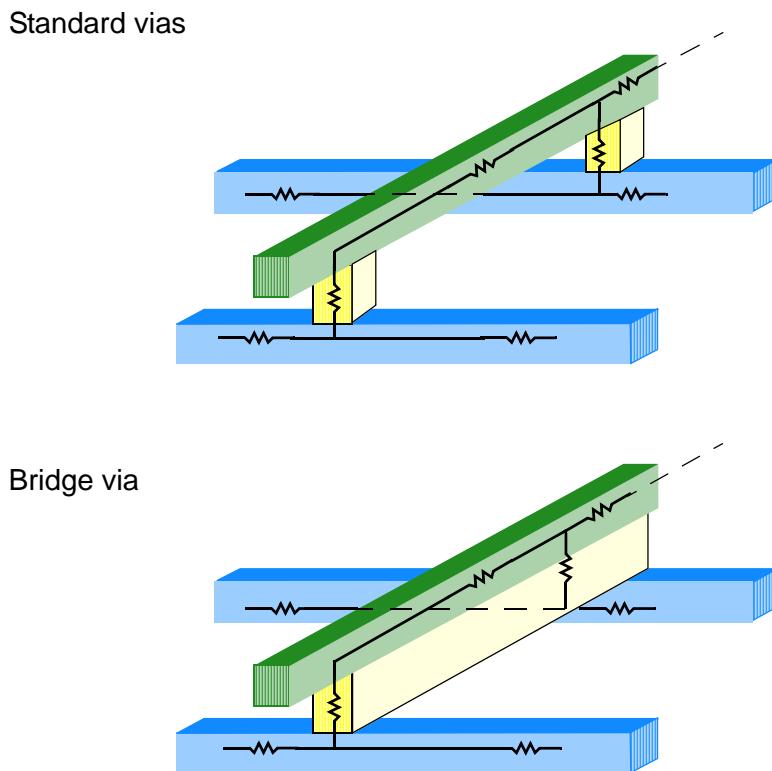
*Figure 13-12 Tall Contact Via Structure*



## Bridge Via Modeling

Conventional vias consist of simple connections between two layers. In advanced processing technologies, the via layer sometimes crosses multiple metal lines in either the top or bottom conductor layers. The top diagram in [Figure 13-13](#) illustrates conventional vias between one metal line in the upper layer and two metal lines in the lower level. The bottom diagram shows a bridge via that connects the same metal lines.

*Figure 13-13 Bridge Via Extraction*



The StarRC tool extracts bridge vias without any changes to the command file. The tool models a bridge via as a set of single vias, as shown by the resistor symbols in [Figure 13-13](#). The following design requirements apply:

- A bridge via can cross two metal lines in either the upper or lower conductor layer. The via cannot cross multiple lines in both layers.
- A bridge via cannot be used only to connect multiple lines on the same conductor layer. The via must connect an upper layer to a lower layer.

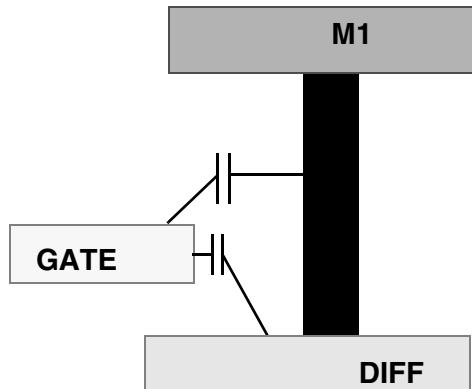
StarRC commands affect the analysis and reporting of bridge vias as follows:

- If the command file contains the `RPV_VS_AREA` command, the entire bridge via area is used to determine the total via resistance. The resistance is divided by the number of connected metal lines to determine the value assigned to each parasitic resistor.
- If the command file contains the `NETLIST_TAIL_COMMENTS: YES` command, the reported via area is the original bridge via area. Via resistors from the same bridge via have the same via layer.
- If the command file contains the `EXTRA_GEOMETRY_INFO: RES` command, the bounding box of the via resistor is the same as the original bridge via.

## Gate-To-Diffusion Capacitance Extraction

Parasitic extraction and circuit simulation tools must avoid double counting or elimination of layout-dependent device parasitics, such as gate-to-contact and gate-to-diffusion capacitance, as shown in [Figure 13-14](#). As process nodes shrink, it is common practice to remove the constant, spatially independent, device-level parasitics from SPICE models in favor of allowing parasitic tools StarRC to extract these components.

*Figure 13-14 Layout Dependent Parasitics*



This section describes the extraction of the gate-to-diffusion capacitance when the `IGNORE_CAPACITANCE: ALL` command is specified. The gate-to-diffusion intradevice capacitance is of interest for parasitic extraction tools because of its strong layout dependency. The gate-to-contact capacitance is extracted by using the `EXTRACT_VIA_CAPS: YES` command in the StarRC command file.

To retain the gate-to-diffusion ( $C_f$ ) capacitance during extraction, use the following command:

```
IGNORE_CAPACITANCE:ALL RETAIN_GATE_DIFFUSION_COUPLING
```

For `IGNORE_CAPACITANCE` settings such as `DIFF` or `NONE`, in which the gate-to-diffusion capacitance is retained by default, StarRC extracts this component as requested.

When you specify this option, StarRC uses the following methods to extract the gate-to-diffusion component:

- Based on precharacterized models, similar to other capacitances extracted by StarRC
- Based on a 2-D capacitance table look-up dependent on layout parameters

---

## Conformal Dielectrics

The `MEASURED_FROM` statement provides the ability to customize the model to account for such process characteristics as conformal dielectrics, mixed conformal and planar dielectrics, and covirtual conductors. When used with a `DIELECTRIC` layer definition, the `MEASURED_FROM` keyword can either refer to a lower dielectric or can have the value `TOP_OF_CHIP`. When used with a `CONDUCTOR` layer definition, the `MEASURED_FROM` keyword can refer only to a lower `PLANAR` dielectric.

The `TOP_OF_CHIP` keyword facilitates the creation of conformal dielectrics. It creates the bottom plane from the layers already present below the new layer and mimics the topology of the existing base layer, copying any existing nonplanarities to the new layer.

The `TOP_OF_CHIP` keyword is required only if you are creating a conformal layer whose topology is based on the top of the chip. If you want to create a conformal layer that is on top of an existing conformal dielectric, you can measure either from `TOP_OF_CHIP` or the existing conformal layer.

A `MEASURED_FROM` statement in `CONDUCTOR` definitions must always refer to a planar dielectric.

If you create a layer in a `MEASURED_FROM` command that refers to a planar layer, the new layer is also planar, regardless of whether or not you define `TW_T` and `SW_T`.

To regain layer planarity after a conformal dielectric has been defined, take the following steps when defining the new planarized layer:

1. Use the `MEASURED_FROM` statement to reference a planar dielectric somewhere lower in the process cross section.
2. Adjust the thickness for the new layer so it is equal to its actual physical thickness plus the thickness of any layer on top of the `MEASURED_FROM` layer.

If you place another dielectric layer on top of the conformal layer without using the `MEASURED_FROM` statement to regain planarity, use the `SW_T` and `TW_T` keywords to set the sidewall and top wall thickness, because the new layer is also conformal.

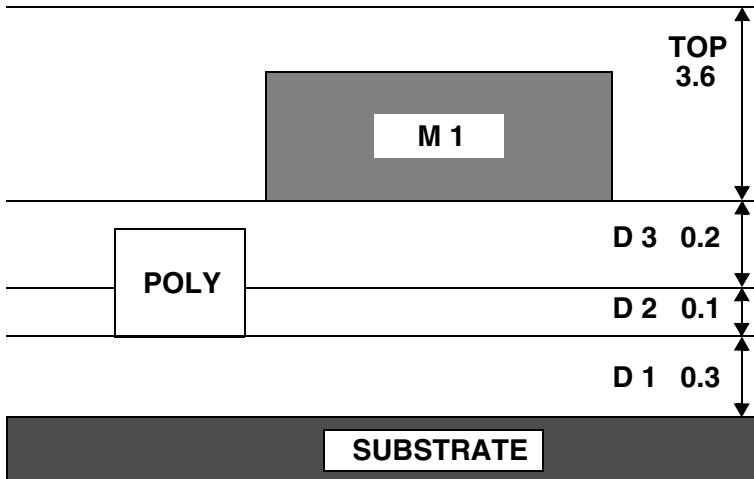
## Conductor Cutting Through Dielectric

If you use the `MEASURED_FROM` statement with a conductor and that conductor layer is measured from a dielectric layer that is placed below another dielectric layer, the conductor might cut through the intermediate dielectric. Consider the following example:

```
TECHNOLOGY = SIMPLE
DIELECTRIC TOP { THICKNESS = 3.600 ER = 3.9 }
CONDUCTOR M1 { THICKNESS = 0.600 WMIN = 0.5
 SMIN = 0.5 RPSQ = 0.05 }
DIELECTRIC D3 { THICKNESS = 0.300 ER = 3.9 }
CONDUCTOR POLY{ THICKNESS = 0.200 WMIN = 0.3
 SMIN = 0.3 RPSQ = 10.0
 MEASURED_FROM = D1 }
DIELECTRIC D2 { THICKNESS = 0.100 ER = 4.2 }
DIELECTRIC D1 { THICKNESS = 0.300 ER = 3.9 }
```

The process cross section is shown in [Figure 13-15](#), where `POLY` cuts through dielectric `D2`.

*Figure 13-15 Conductor Cuts an Intermediate Dielectric*



---

## Covertical Conductors

StarRC supports covertical (vertically overlapping) conductors. See the gate poly and local interconnect processes in [Chapter 14, “ITF Examples”](#) for examples that illustrate the handling of covertical conductors that might have unique thicknesses, as in the case of local poly interconnect.

In this case, the layout database should be modified for the covertical layers, so that those layers (gate and field poly, or poly and local interconnect) do not overlap each other. This can be done in the Hercules runset by use of `BOOLEAN` operations:

```
BOOLEAN POLY NOT LI {} TEMP=POLY
```

or

```
BOOLEAN POLY AND LI {} TEMP=LI_OVERLAP
BOOLEAN POLY NOT LI_OVERLAP {} TEMP=POLY
BOOLEAN LI NOT LI_OVERLAP {} TEMP=LI
```

In the latter case, both `LI` and `LI_OVERLAP` are mapped to the local interconnect layer in the `nxtgrd` file, and the `CONNECT` sequence in the Hercules runset must be modified accordingly.

Another use for covertical conductors is to handle “metal cheeseing” (also known as wide metal slotting); it creates two metal layers and gives them different sheet resistances, which can be done in the mapping file without changing anything in the ITF file, as follows:

```
conducting_layers
M5 metal5 RPSQ=10
M5_cheese metal5 RPSQ=100
```

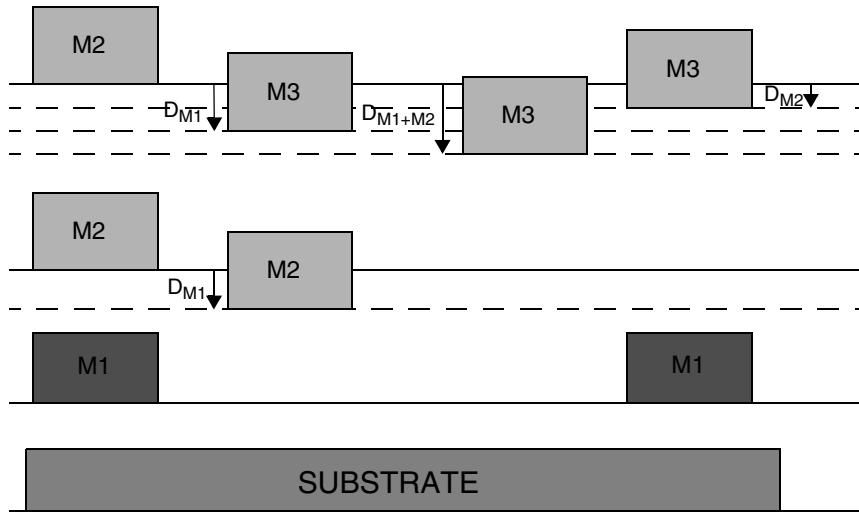
Note that making separate layers in the ITF file for covertical conductors is suitable only for capacitive modeling; you should not use it for modeling resistance differences.

---

## Conductor Drop Factor

The drop factor handles the case in which a conducting layer is at different heights because of the absence of a lower conducting layer. For example, if Metal2 runs over Metal1, Metal2 is uniform at a certain height above Metal1. If Metal2 is layered over a location where there is no Metal1, Metal2 is layered at a lower height. The drop factor considers the differences between the conducting layer heights and calculates the area and lateral capacitance correctly. An illustration of the process is shown in [Figure 13-16](#).

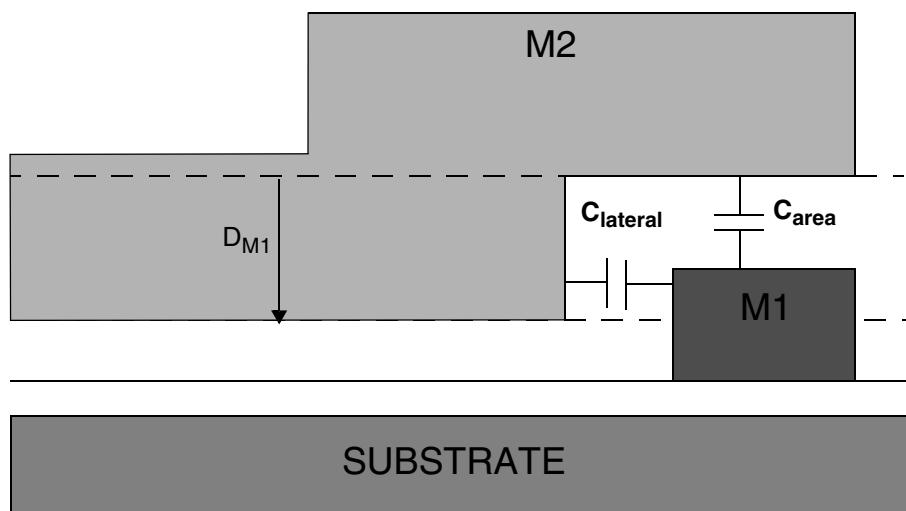
*Figure 13-16 Nonplanar Process With Varying Conductor Heights*



Nonplanar conductor modeling is typically required for legacy processes at process nodes above 0.18 micron with smaller numbers of metal conducting layers, for these reasons:

- Such processes typically contain three metals or less.
- Nonplanarities can be introduced by any missing layer in the physical cross section.
- Both area and lateral capacitance effects are relevant between adjacent metal layers. Depending on the degree of drop of an upper conductor, the drop could introduce a covertical overlap between consecutive conductors that would introduce a potentially significant lateral capacitance effect. For example, see [Figure 13-17](#).

*Figure 13-17 Lateral and Area Capacitance Effects Introduced by Large Drop Factor Values*



## Drop Factor Error Conditions

StarRC might find the following error conditions in your design:

- Specifying the `DROP_FACTOR` ITF statement should not cause different horizontally consecutive levels of the same conductor to become noncovertical with each other. In other words, if a piece of conductor routing undergoes a different cumulative drop factor as the number of lower conductors vary along the length of the route, the conductor should never drop such that it can no longer abut with itself. Horizontally adjacent pieces of a conductor can fail to be covertical because of an excessive cumulative drop factor. See [Figure 13-18](#).
- No conductor can be modeled at a height below conductors represented at lower levels in the ITF cross sectional description. If this is the case, the `grdgenxo` tool issues an error. See [Figure 13-19](#).
- The drop factor is not supported with processes that have these features:
  - Covertical layers such as gate and field polysilicon or polysilicon and local interconnect
  - Metal fill emulation using `FILL_SPACING`, `FILL_WIDTH`, and `FILL_RATIO`
  - Conformal dielectrics
- You can specify the `DROP_FACTOR` keyword for no more than four conductors.

*Figure 13-18 Drop Factor Error Condition 1*

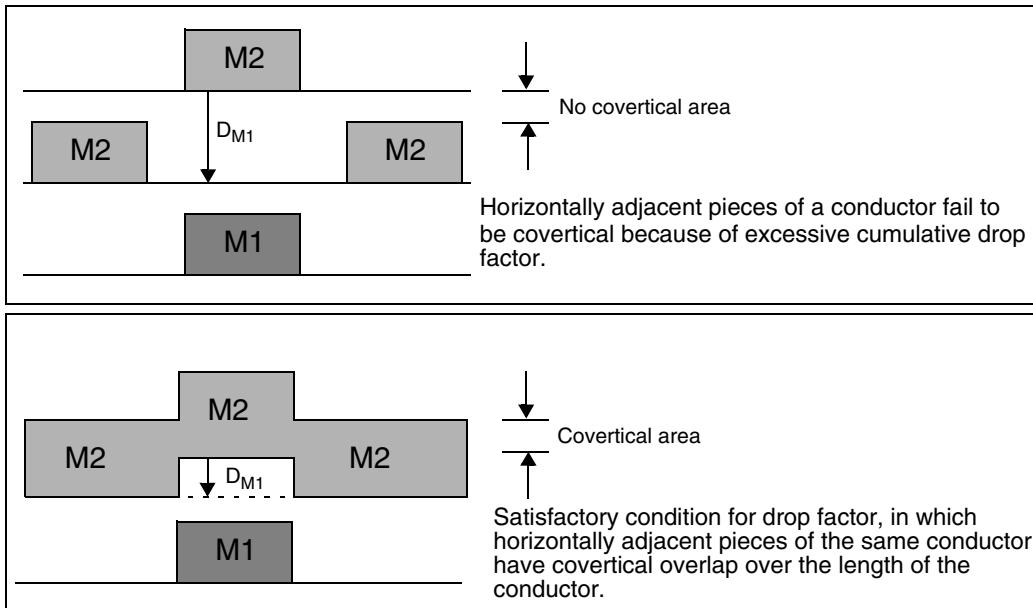
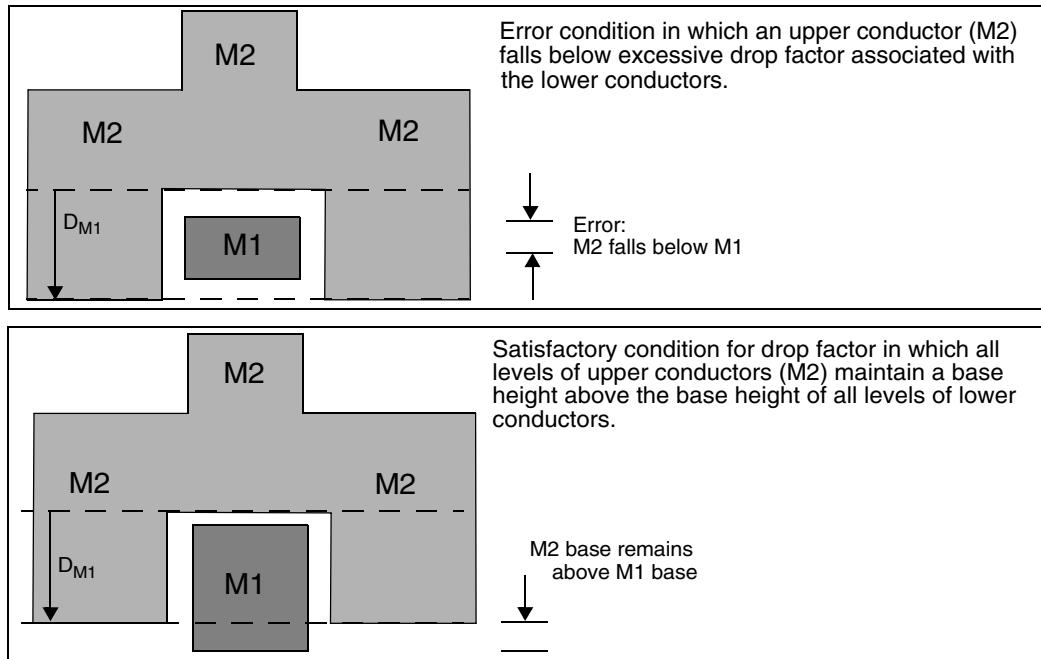


Figure 13-19 Drop Factor Error Condition 2

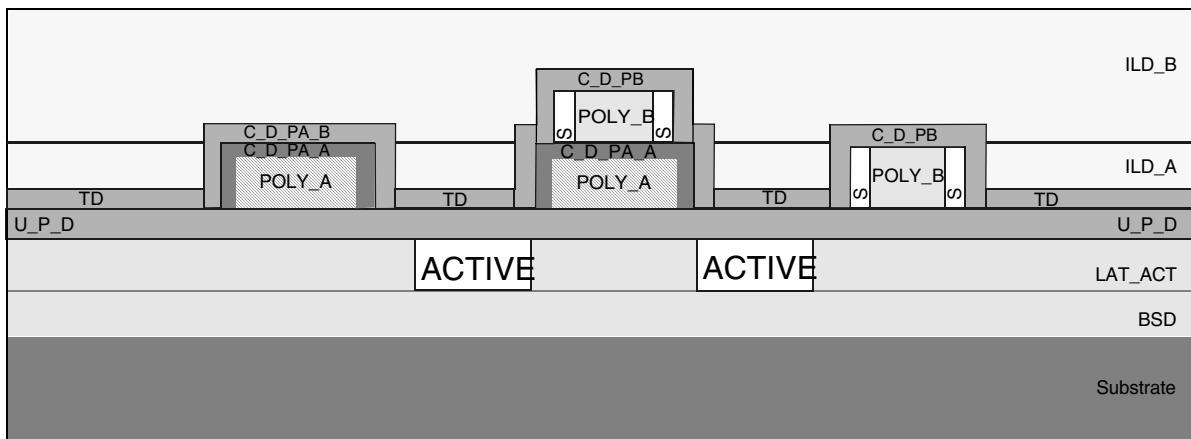


## Double-Polysilicon Process

To model a double-polysilicon process, set the `IS_CONFORMAL` and `ASSOCIATED_CONDUCTOR` options in the `DIELECTRIC` layer block of the ITF file. The `IS_PLANAR` option is necessary in this case to make the metals above the poly layers planar.

See an example of this cross section in [Figure 13-20](#).

*Figure 13-20 Conformal Dielectric and Poly Layers*



The following ITF statements model the cross section shown in [Figure 13-20](#).

```

DIELECTRIC ILD_B { THICKNESS=0.3 MEASURED_FROM=ILD_A ER=4.2 }
DIELECTRIC C_D_PB { IS_CONFORMAL ER=7 SW_T=0.03 TW_T=0.03
 ASSOCIATED_CONDUCTOR=POLY_B }
DIELECTRIC S { IS_CONFORMAL ER=6 SW_T=0.055 TW=0.0
 ASSOCIATED_CONDUCTOR=POLY_B }
CONDUCTOR POLY_B { THICKNESS=0.15 WMIN=0.08 SMIN=0.07 RPSQ=10.0 }
DIELECTRIC ILD_A { THICKNESS=0.10 MEASURED_FROM=TD ER=4.2 }
DIELECTRIC TD { ER=7 MEASURED_FROM=U_P_D THICKNESS=0.03 }
DIELECTRIC C_D_PA_B { IS_CONFORMAL ER=7 SW_T=0.03 TW_T=0.03
 ASSOCIATED_CONDUCTOR=POLY_A }
DIELECTRIC C_D_PA_A { IS_CONFORMAL ER=3.9 SW_T=0.04 TW_T=0.01
 ASSOCIATED_CONDUCTOR=POLY_A }
CONDUCTOR POLY_A { THICKNESS=0.12 WMIN=0.05 SMIN=0.05 RQSP=849
 DROP_FACTOR=0.13 }
DIELECTRIC U_P_D { THICKNESS=0.05 ER=3.9 }
DIELECTRIC LAT_ACT { THICKNESS=0.19 ER=4 }
CONDUCTOR ACTIVE { THICKNESS=0.19 WMIN=0.1 SMIN=0.14 RPSQ=0.0001 }
DIELECTRIC BSD { THICKNESS=0.19 ER=4 }

```

## Layer Etch

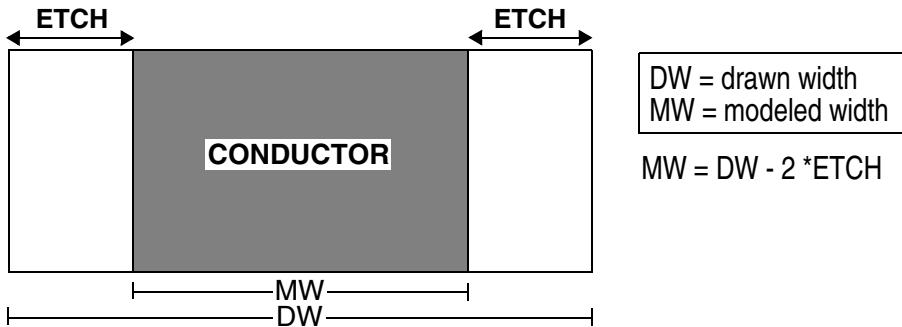
You can make an adjustment in the ITF process file for layer etch effects that cause the manufactured line width of a conductor to be different from its drawn width. The `ETCH` statement can be specified as a part of any `CONDUCTOR` definition.

Both conductor sidewalls retreat or expand by the value specified in the `ETCH` statement, resulting in a net width change of twice the etch value, as illustrated in [Figure 13-21](#). A positive etch value shrinks the conductor width, and a negative `ETCH` value increases the conductor width.

`WMIN` and `SMIN` values are not affected by the `ETCH` statement because StarRC makes the etch adjustments internally.

See [Layer Etch Process ITF Example](#) for an example of a process with layer etch.

*Figure 13-21 Process Using Layer Etch Adjustment*



## Spacing- and Width-Dependent Etch

Spacing- and width-dependent etch can be implemented in the `nxtgrd` with the `ETCH_VS_WIDTH_AND_SPACING` option within a `CONDUCTOR` block of the ITF file.

With this feature, StarRC can consider the actual fabricated patterns in extracting parasitic components. This is important, because optical proximity correction (OPC) cannot fix all proximity effects and the actual patterns might be different from the drawn mask patterns.

If you add `ETCH_VS_WIDTH_AND_SPACING` to an existing ITF file, you must rerun the `grdgenxo` tool after removing the working directory.

The `ETCH_VS_WIDTH_AND_SPACING` statement can be used with the `ETCH` statement. If these statements are used together, the `ETCH_VS_WIDTH_AND_SPACING` calculation is applied before the `ETCH` adjustment, then the `RPSQ_VS_SI_WIDTH` value is calculated.

The `ETCH_VS_WIDTH_AND_SPACING` statement affects both capacitance and resistance by default. You can use the `CAPACITIVE_ONLY` or `RESISTIVE_ONLY` options to restrict the effect to capacitance or resistance, respectively.

These two options can be used together within a given `CONDUCTOR` statement but cannot be used in conjunction with normal `ETCH_VS_WIDTH_AND_SPACING` tables. The `RESISTIVE_ONLY` and `CAPACITIVE_ONLY` options can each be defined only one time within any given `CONDUCTOR` statement.

---

## Determining WMIN and SMIN Values

It is important to have a correct set of `WMIN` and `SMIN` values for the `CONDUCTOR` object that contains the `ETCH_VS_WIDTH_AND_SPACING` statement.

The `WMIN` and `SMIN` values of the conductor described by the `ETCH_VS_WIDTH_AND_SPACING` statement can be the same as the smallest value (or the first entry) in the `WIDTHS` and `SPACINGS` tables, respectively.

Inappropriate `WMIN` and `SMIN` values might cause unwanted opens or shorts of the neighboring layers by applying the etch values provided in the table. In such a case, a message is printed during the run. For the entries corresponding to the `WMIN` in the `WIDTHS` table, if positive etch values are greater than or equal to half of the `WMIN` value, an open warning is issued.

For the entries corresponding to the `SMIN` value in the `SPACINGS` table, if absolute values of the negative etch are greater than or equal to half of the `SMIN` value, a potential short condition exists. However, reporting of this condition is optional because most such errors should be caught during design rule checking. To enable `SMIN` violation reporting, use the `REPORT_SMIN_VIOLATION: YES` command in the StarRC command file (the default is `NO`).

---

## Overlapping Wells

You can set the vertical profile of the substrate. To set the vertical precedence for layers mapped to the substrate, use the following mapping file syntax:

```
conducting_layers
db_layer1 SUBSTRATE precedence=pos_integer
db_layer2 SUBSTRATE precedence=pos_integer
...
...
```

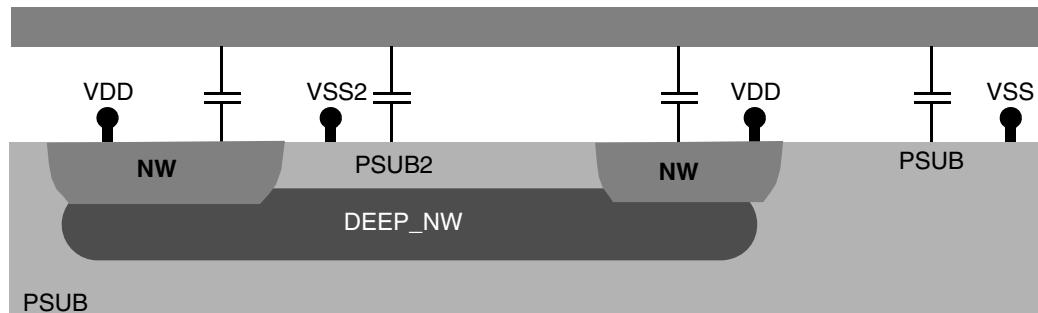
Any layer mapped to the substrate, and only layers mapped to the substrate, accepts a positive integer precedence value that establishes the layer's position in the vertical substrate profile. Larger numbers denote higher vertical precedence, while smaller numbers denote lower vertical precedence. It is not necessary for values to be sequential.

If two layers have the same precedence value, and polygons from those two layers overlap in the layout, StarRC chooses the topmost layer for the purpose of coupling capacitance attachment and `IGNORE_CAPACITANCE` command functionality. SUBSTRATE-mapped layers for which precedence is not specified have a precedence value of zero, meaning that their precedence is lower than all other layers.

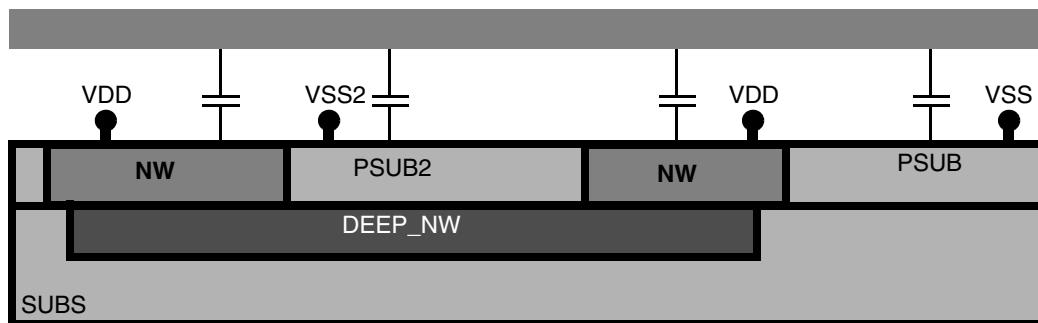
The following mapping file example shows a mapping file used to establish vertical precedence for a buried well profile. [Figure 13-22](#) shows the profile of a physical well for a buried well process and a profile for a discrete well.

```
conducting_layers
 SUBS SUBSTRATE precedence=1
 DEEP_NW SUBSTRATE precedence=2
 NW SUBSTRATE precedence=3
 PSUB2 SUBSTRATE precedence=3
 PSUB SUBSTRATE precedence=3
```

*Figure 13-22 Physical Well and Discrete Buried Well Profile*



Physical well profile for buried well process



Discrete buried well profile for parasitic extraction

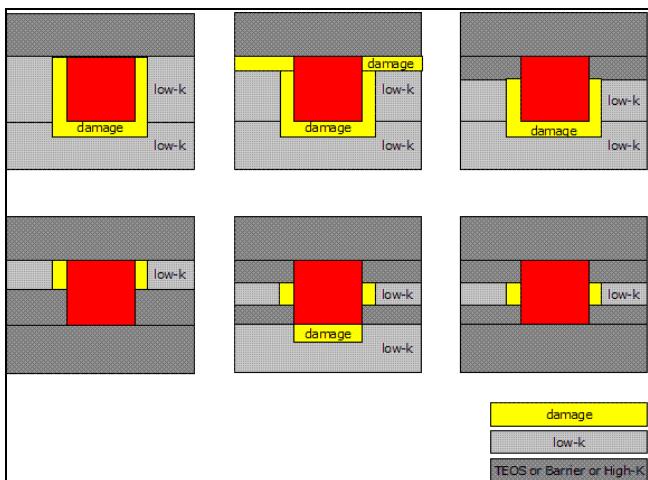
## Damage Modeling

For advanced process nodes, sidewall and bottom wall damage as a consequence of processing low-k dielectrics might need to be considered.

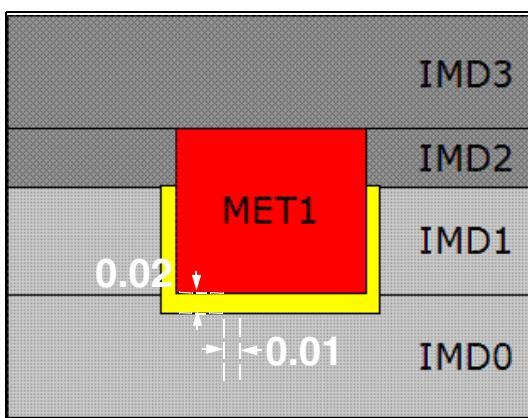
The `DAMAGE_THICKNESS` ITF statement defines the thickness of the damage region on the surface of the layer, while the `DAMAGE_ER` ITF statement specifies the equivalent permittivity of the damage region.

[Figure 13-23](#) shows the damage modeling cross sections that are modeled with the `DAMAGE_ER` and `DAMAGE_THICKNESS` statements.

*Figure 13-23 Damage Modeling Cross Sections*



*Figure 13-24 Low-K Damage*



In [Figure 13-24](#), the damage defined for `IMD1` models the side wall damage because `IMD1` is the intrametal dielectric for `MET1`, whereas `IMD0` models the bottom wall damage because `MET1` is on top of the dielectric layer `IMD0`.

The following example corresponds to [Figure 13-24](#).

```
DIELECTRIC IMD3 { THICKNESS=0.09 ER=2.9 }
DIELECTRIC IMD2 { THICKNESS=0.07 ER=4.5 }
DIELECTRIC IMD1 { THICKNESS=0.13 ER=2.9
 DAMAGE_THICKNESS=0.01 DAMAGE_ER=5.1 }
 CONDUCTOR MET1 { THICKNESS 0.20 SMIN=0.1 WMIN=0.1 }
DIELECTRIC IMD0 { THICKNESS=0.10 ER=2.9
 DAMAGE_THICKNESS=0.02 DAMAGE_ER=5.1 }
```

## Temperature Derating

The StarRC tool can model temperature-dependent resistance for conducting layers and vias. The resistances are modeled in the same way as they are in SPICE, by using the following equation:

$$R = R_0 * [1 + CRT1 * (T - T_0) + CRT2 * (T - T_0)^2]$$

$R_0$  is the resistance value at the nominal temperature  $T_0$ ,  $CRT1$  and  $CRT2$  are linear and quadratic temperature coefficients, and  $R$  is the modeled resistance at the operating temperature  $T$ . Note that the modeled resistance  $R$  exactly equals the nominal resistance ( $R_0$ ) if  $T=T_0$ , or if both  $CRT1=0$  and  $CRT2=0$ .

The statement options  $CRT1$ ,  $CRT2$ , and  $T_0$  are specified on a per-layer basis. If either  $CRT1$  or  $CRT2$  is nonzero for a layer (vias included), then a nominal temperature is required for that layer.

The defaults for  $CRT1$  and  $CRT2$  are zero. The default nominal temperature for the layers can be set globally with the `GLOBAL_TEMPERATURE` statement at the beginning of the ITF file. If the temperature is set both globally and at a layer, the layer nominal temperature overrides the global setting.

```
GLOBAL_TEMPERATURE = temp_in_Celsius
```

## Half-Node Scaling

The half-node scaling function allows you to scale the input design data uniformly across all layers without affecting the connectivity of the layout network.

The `HALF_NODE_SCALE_FACTOR` statement allows for a transparent shrink flow. The flow requires downstream tools to interpret this option. This flow produces modified electrical properties, for example resistance and capacitance, but retains the original unshrunk design coordinates for the final netlist. The `HALF_NODE_SCALE_FACTOR` function does not require scaling options to be set in other tools. The technology files supplied to the designer (from the foundry or CAD design group) contain the scaling factor for the particular design flow.

## How the Function Works

Set the `HALF_NODE_SCALE_FACTOR` option in the ITF file as shown in the following example:

```
TECHNOLOGY = 65nm_example
GLOBAL_TEMPERATURE = 25
HALF_NODE_SCALE_FACTOR = 0.9
DIELECTRIC PASS2 {THICKNESS=0.800000 ER=6.9}
$DIELECTRIC PASS1 {THICKNESS=0.700000 ER=4.0}
```

If a half-node scale factor exists in the nxtgrd file, the StarRC tool automatically sets the `NETLIST_UNSCALED_COORDINATES` command to `YES` and the `MAGNIFY_DEVICE_PARAMS` command to `NO` and issues a warning message to notify you of the changes.

*Table 13-1 Half-Node Scale Factor Effect on Commands*

| Command                                               | Function with <code>HALF_NODE_SCALE_FACTOR</code>                                                                                                                      |
|-------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>NETLIST_UNSCALED_COORDINATES: YES</code>        | Writes all coordinate information in the netlist at full node (automatically set).                                                                                     |
| <code>MAGNIFY_DEVICE_PARAMS: NO</code>                | Writes standard device properties ( <code>\$w</code> , <code>\$l</code> , <code>\$area</code> , and so on) at full node in transistor-level flows (automatically set). |
| <code>NETLIST_DEVICE_LOCATION_ORIENTATION: YES</code> | Writes full node (original GDSII) coordinates in the netlist for flows requiring device locations.                                                                     |

## How Scaling Affects The Netlist

The following is an example of coordinates that are affected in a SPICE-like netlist:

```
*|I (ZN:F12 M1 SRC B 0 0.48 0.64)
*|P (ZN B 0 0.695 1.115)
*|S (ZN:16 0.53 1.545)
```

If the `NETLIST_TAIL_COMMENTS` command is used to write the physical properties of parasitic resistors (used for reliability analysis flows), the properties are the full-node size:

```
R1 F9 F8 0.588229 $l=9.9 $w=2 $lvl=1
```

The `HALF_NODE_SCALE_FACTOR` option does not change the function of the `MAGNIFICATION_FACTOR` option. However, you cannot use the `MAGNIFICATION_FACTOR` option with the `HALF_NODE_SCALE_FACTOR` option.

## Changing a Scale Factor Value in the nxtgrd File

If you generated the nxtgrd file without setting a scale factor, or you would like to change the scale factor, run the grdgexo tool to generate an updated nxtgrd file. The following example sets the scale factor to 0.9:

```
% grdgexo -add_sf 0.9 -i noshrink.nxtgrd -o shrink.nxtgrd
```

If you generated a StarRC nxtgrd file with a HALF\_NODE\_SCALE\_FACTOR value and you would like to run extraction without the shrink, remove the scaling factor from the nxtgrd file by setting the factor to 1, as follows:

```
% grdgexo -add_sf 1 -i noshrink.nxtgrd -o shrink.nxtgrd
```

Note:

You can set the MAGNIFICATION\_FACTOR command in the StarRC command file after changing the value in the nxtgrd file. You cannot delete the HALF\_NODE\_SCALE\_FACTOR line from the nxtgrd file as this causes the nxtgrd file to be corrupt.

## Interface to Reliability Flows

Reliability tools read the netlist output from the StarRC. Because the netlist from StarRC represents full node coordinates, the reliability tool's electromigration rules are supported at the full node.

The half node scale factor is written as a comment in the final netlist for downstream analysis tools to compute the physical width for estimation. For example:

```
//COMMENTS
//TCAD_GRD_FILE /remote/na4apd/starrc/group/option_2/tcad/grd
//TCAD_TIME_STAMP Sun Feb 18 12:08:22 2007
//TCADGRD_VERSION 56
//HALF_NODE_SCALE_FACTOR 0.9
```

## Via Merging

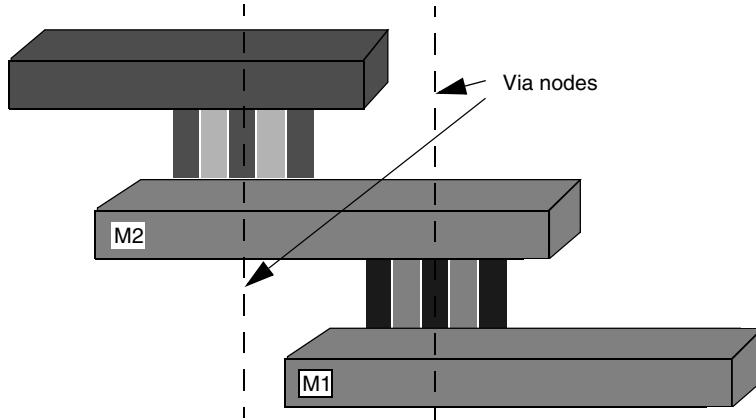
The merging of vias is controlled by the MAX\_VIA\_ARRAY\_LENGTH mapping file command. The command determines how many vias along one side can be merged together. Large via arrays are split into smaller sets of via arrays for merging, thus reducing the netlist size. Without this option, a via array with via spacing less than or equal to the MAX\_VIA\_ARRAY\_SPACING value is reduced to one via and eventually to one resistor.

The output netlist contains one subnode (\*|S) for every resultant via array. The resistors are listed with an effective resistance value including a summation of area for all individual vias in the group. [Figure 13-25](#) illustrates the effective resistances of three conductors connected by merged via arrays.

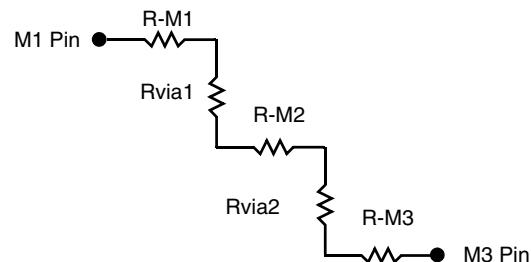
The mapping file syntax for this function is as follows:

```
VIA GRD_VIA RPV = value
AREA = value
MAX_VIA_ARRAY_SPACING = value
MAX_VIA_ARRAY_LENGTH = value
```

*Figure 13-25 R Network for a Multilayer Net*



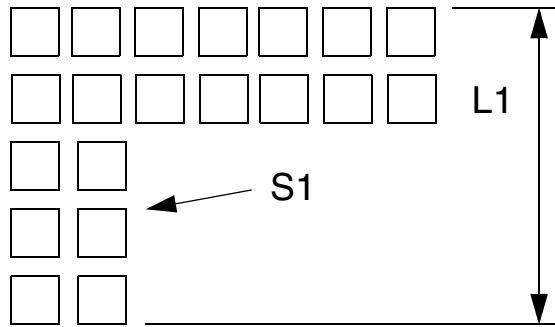
The expected R network is as follows:



## Grouping Vias in an L-Shaped Array

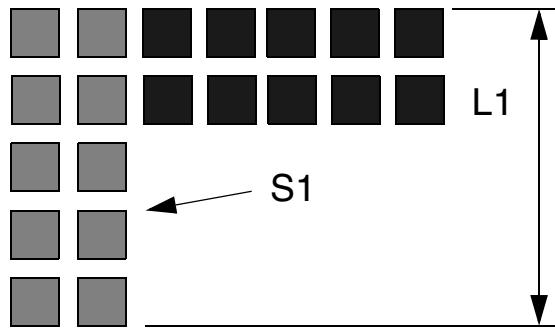
In an L-shaped via array, StarRC groups the vias into multiple sets in an arbitrary manner as shown in [Figure 13-26](#).

*Figure 13-26 Case 1 - Before Merge*



If you specify `MAX_VIA_ARRAY_SPACING: S1` and `MAX_VIA_ARRAY_LENGTH: L1`, the via array can be divided into two groups.

*Figure 13-27 Case 1 - After Merge*

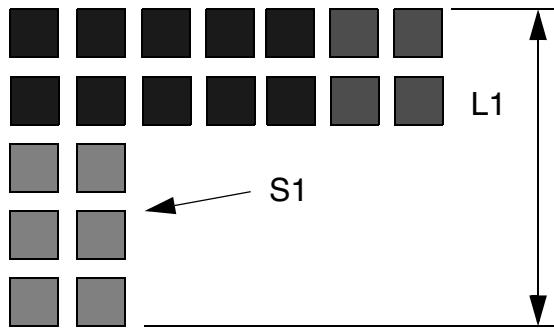


The merged result is shown in [Figure 13-27](#). The output netlist for this case is as follows:

```
R1 no:1 no:2 (1/10)*via_res $area=10*via area $lvl= num
R2 no:3 no:4 (1/10)*via_res $area=10*via area $lvl=num
R3 no:2 no:3 value $w =num $l=num $lvl =num
```

Another possibility is that StarRC might divide the vias into three groups as shown in [Figure 13-28](#).

*Figure 13-28 Case 1 - Dividing Into Three Groups*



The output netlist for three groups is as follows:

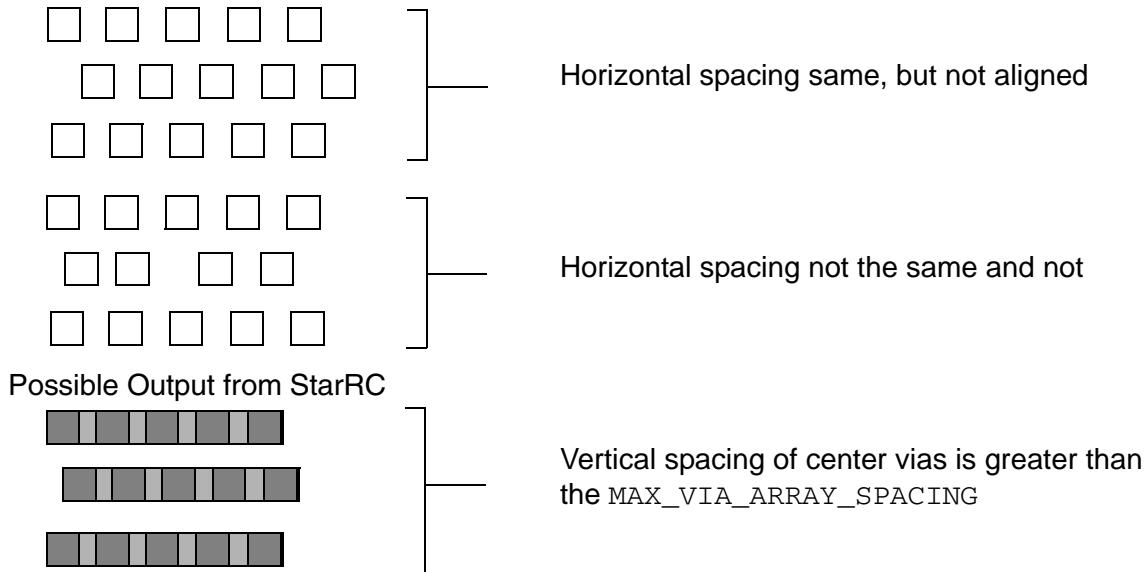
```
R1 no:1 no:2 (1/6)*via_res $area=6*via_area
R2 no:3 no:4 (1/10)*via_res $area=10*via_area
R3 no:5 no:6 (1/4)*via_res $area=4*via_area
R4 no:2 no:3 value $w =num $l=num $lvl =num
R5 no:4 no:5 value $w =num $l=num $lvl =num
```

---

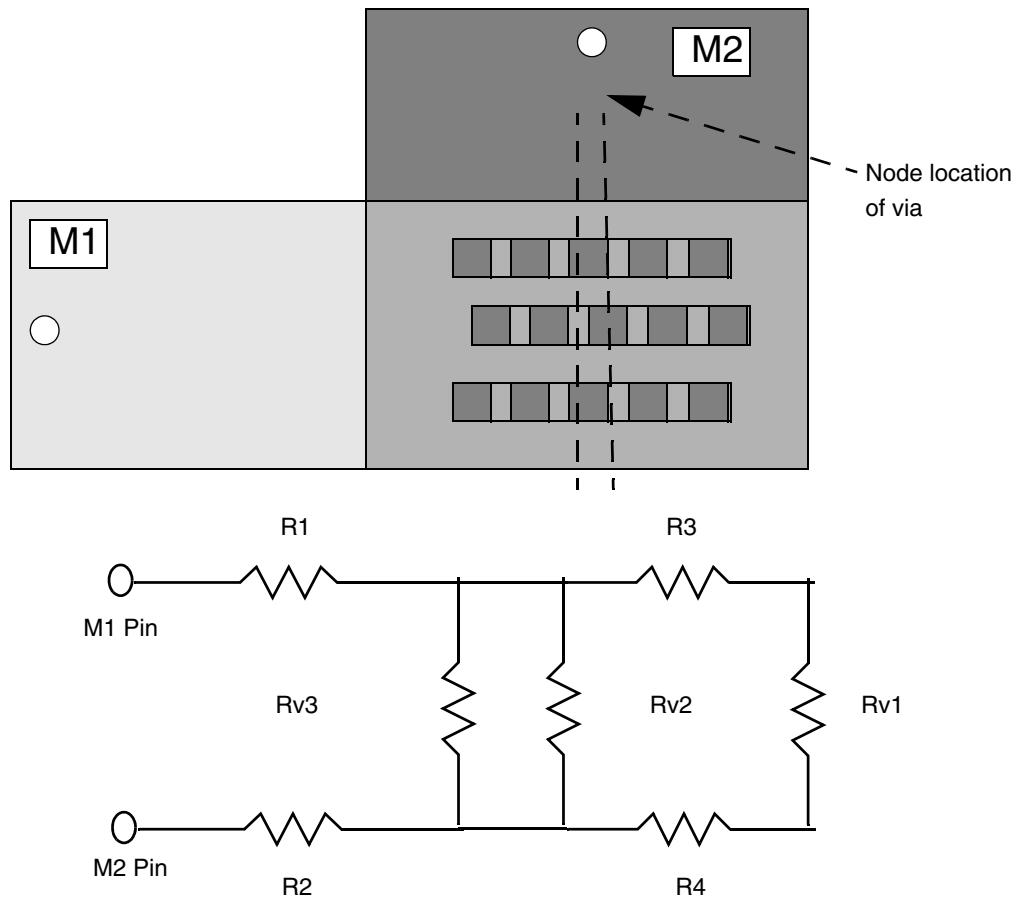
## Asymmetric Via Arrays

If the design has asymmetric via arrays with different pitch, then StarRC groups them arbitrarily based on the specified `MAX_VIA_ARRAY_SPACING` and `MAX_VIA_ARRAY_LENGTH` commands, as shown in [Figure 13-29](#).

*Figure 13-29 Case 2 - Irregular Horizontal Spacing*

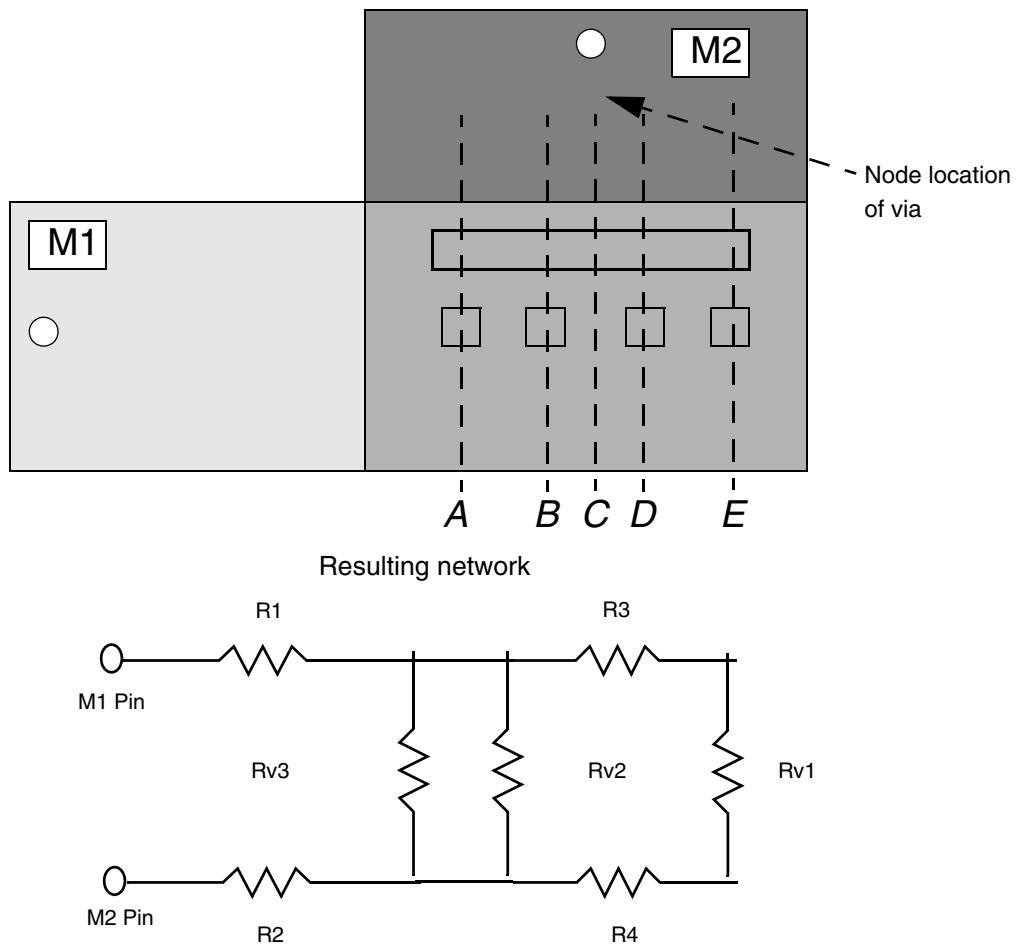


*Figure 13-30 View from Under the Network*



In [Figure 13-30](#), you can possibly get below the network. However, if the distance between two via node locations is small, it can be merged. This means R3 and R4 can be shorted.

*Figure 13-31 View from Under the Network - Multiple Nodes*

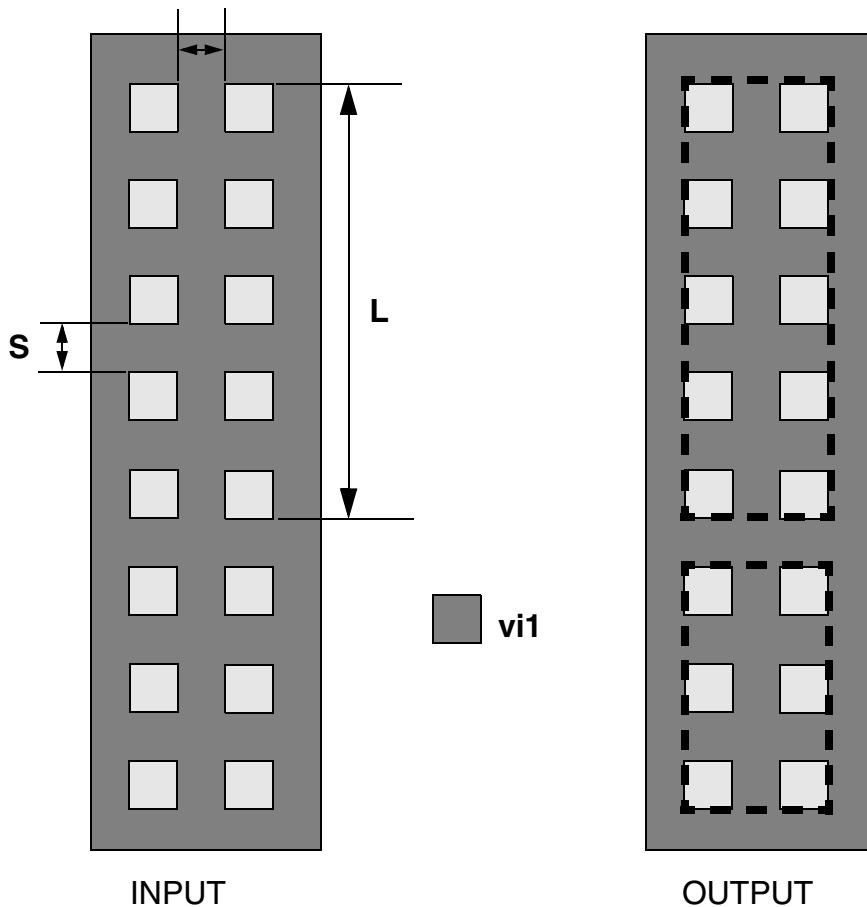


In [Figure 13-31](#), StarRC creates multiple nodes for the vias. It chooses the center for the via-merged polygon (node C as the merged polygons are aligned). Nodes A, B, D, and E are created for the individual vias. Because the distance between nodes B, C, and D is small, the metal resistance is shorted out and the resulting network is shown in the lower portion of the figure.

# Rectilinear Via Arrays

In a simple rectilinear via array, as shown in Figure 13-32, StarRC merges the vias based on the settings of the `MAX_VIA_ARRAY_SPACING` and `MAX_VIA_ARRAY_LENGTH` commands.

*Figure 13-32 Simple Rectilinear Via Array*



MAX\_VIA\_ARRAY\_SPACING: S

MAX\_VIA\_ARRAY\_LENGTH: L

The output netlist for the arrays shown in Figure 13-32 is as follows:

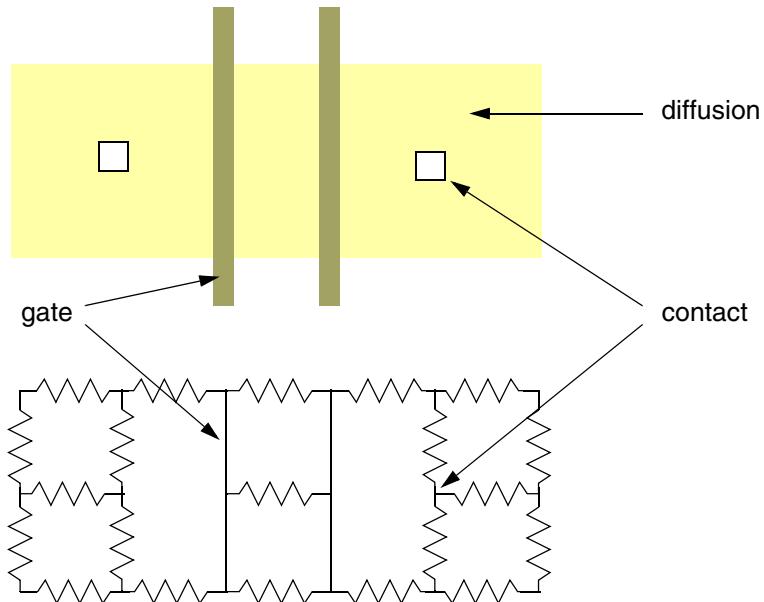
```
R1 no:1 no:2 (1/10)*via_res $area=10*via_area
R2 no:3 no:4 (1/6)* via_res $area=6* via_area
```

## Diffusion Resistance

In the ITF file, diffusion is defined as a conductor for a standard shallow trench isolation (STI) process. By default, if diffusion is not defined in the ITF file, no resistance is extracted.

Diffusion resistance is extracted as a mesh by default. The gate and diffusion overlap is located at the equipotential surface (line node), as illustrated in [Figure 13-33](#).

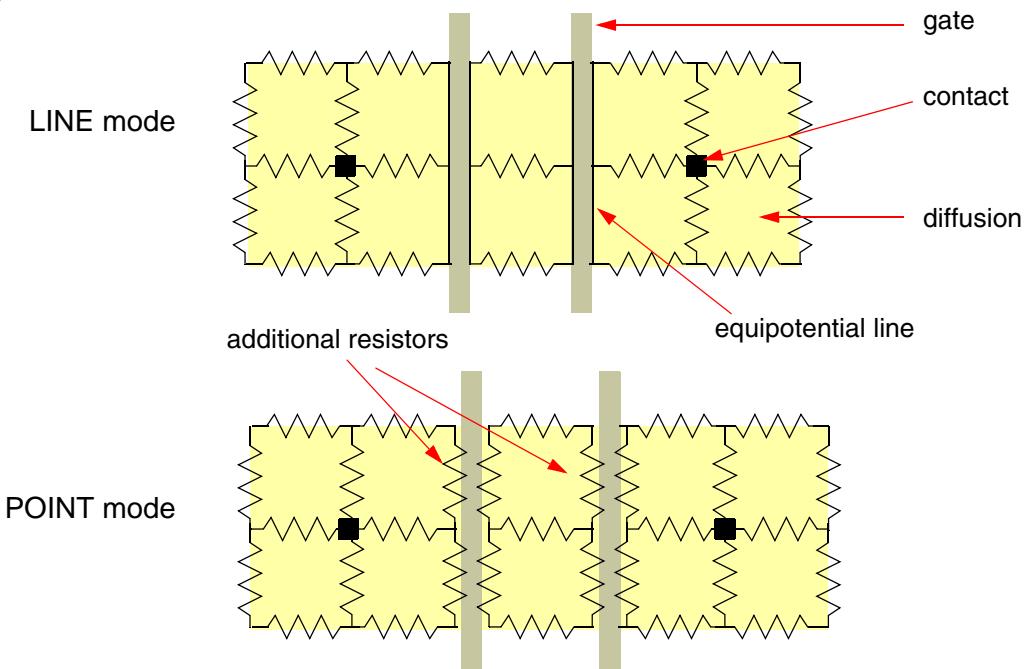
*Figure 13-33 Resistance Mesh in Source and Drain Diffusion Region (LINE mode)*



For advanced process nodes and transistors with large widths, the aspect ratio of the diffusion might be very large. You can model the effect with additional resistors along the source and drain contact area by setting the `DIFFUSION_RES_MODE` command to `POINT` (the default is `LINE`).

Figure 13-34 illustrates the two models.

Figure 13-34 Diffusion Resistance Modes



## User-Defined Diffusion Resistance

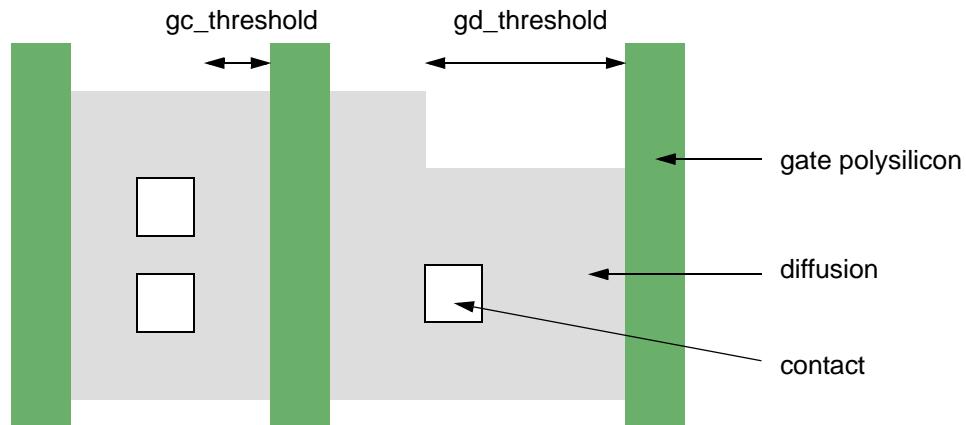
For advanced processing nodes, you can take factors such as contact location and diffusion layout into account by applying a user-defined model to specific diffusion patterns.

To use this analysis, perform the following steps:

1. Set the `USER_DEFINED_DIFFUSION_RESISTANCE` command to `YES` in the StarRC command file. (This step is optional because the command defaults to `YES`.)
2. Include the `USER_DEFINED_DIFFUSION_RESISTANCE` command within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
3. Set the `LAYER_TYPE` keyword to `GATE` within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
4. Specify a model name by using the `diffusion_res_model` keyword in the `conducting_layers` command in the mapping file.

Resistances calculated by the user-defined diffusion resistance method are inserted into the circuit before reduction and are included in reduction operations. To use this method, the layout dimensions must be smaller than the threshold values specified by the `GATE_TO_CONT_THRESHOLD` parameter (labeled `gc_threshold` in [Figure 13-35](#)) and the `GATE_TO_DIFF_BEND_THRESHOLD` parameter (labeled `gd_threshold`).

*Figure 13-35 User-Defined Diffusion Resistance Parameters*



You can use the `grdgenxo -res_update` command to update the `nxtgrd` file with new user-defined diffusion resistance parameters.

Simultaneous multicorner extraction is supported. However, the `GATE_TO_CONT_THRESHOLD` and `GATE_TO_DIFF_BEND_THRESHOLD` values must be identical between corners. Temperature sensitivity analysis is not supported.

# 14

## ITF Examples

---

This appendix contains examples of ITF files for several process technologies.

Each of the following sections contains an ITF file example and a diagram of the process cross section:

- [Fully Planar Process ITF Example](#)
- [Conformal Dielectric Process ITF Example](#)
- [Gate Poly Process ITF Example](#)
- [Local Interconnect Process ITF Example](#)
- [Layer Etch Process ITF Example](#)
- [Emulated Metal Fill Process ITF Example](#)
- [Transistor-Level Process ITF Example](#)
- [Through-Silicon Via Process ITF Example](#)
- [Trench Contact Process ITF Example](#)

## Fully Planar Process ITF Example

The following example ITF file describes the fully planar process shown in [Figure 14-1](#).

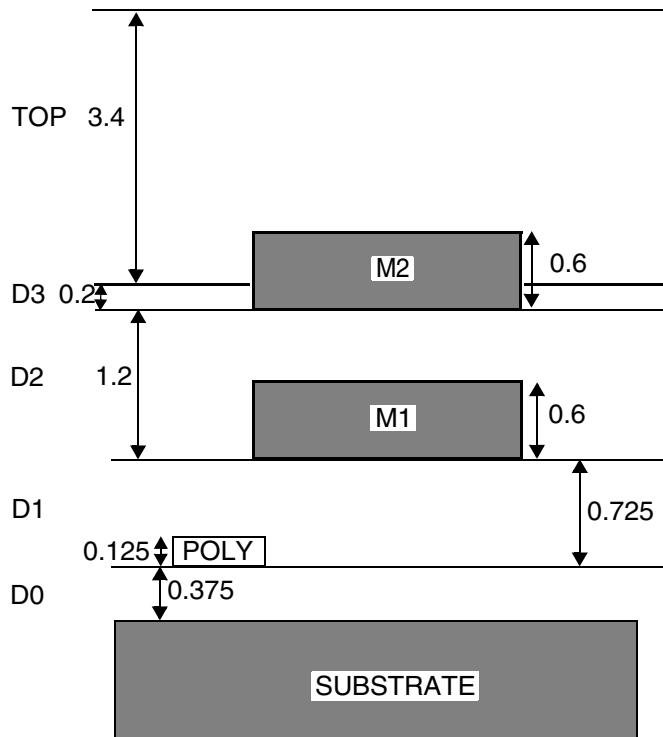
```

TECHNOLOGY = planar
DIELECTRIC TOP { THICKNESS=3.4 ER=3.9 }
DIELECTRIC D3 { THICKNESS=0.2 ER=4.7 }
CONDUCTOR M2 { THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05 }
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 { THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY{ THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0{ THICKNESS=0.375 ER=3.9 }

VIA sub_tie { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA poly_cont { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA via { FROM=M1 TO=M2 AREA=0.36 RPV=4 }

```

*Figure 14-1 Fully Planar Process*



## Conformal Dielectric Process ITF Example

The following example ITF file describes the conformal dielectric process shown in [Figure 14-2](#).

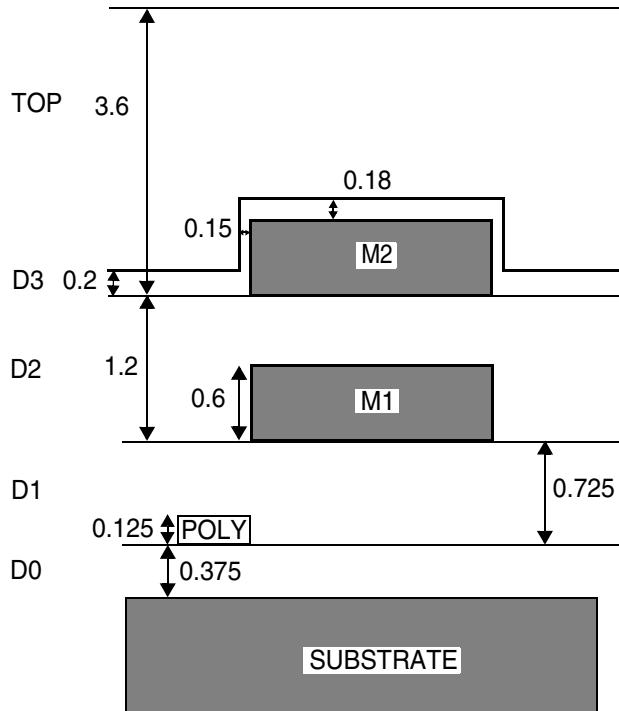
```

TECHNOLOGY = conformal
$ TOP is planarized by measuring from D2
DIELECTRIC TOP { THICKNESS=3.6 MEASURED_FROM=D2 ER=3.9 }
$ D3 is a conformal dielectric
DIELECTRIC D3 {
 THICKNESS=0.2 MEASURED_FROM=TOP_OF_CHIP
 SW_T=0.15 TW_T=0.18 ER=5.9 }
CONDUCTOR M2 { THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05 }
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 { THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY { THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
 }
DIELECTRIC D0 { THICKNESS=0.375 ER=3.9 }

VIA DIFF_CONT { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA POLY_CONT { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA V1 { FROM=M1 TO=M2 AREA=0.36 RPV=4 }

```

*Figure 14-2 Conformal Dielectric Process*

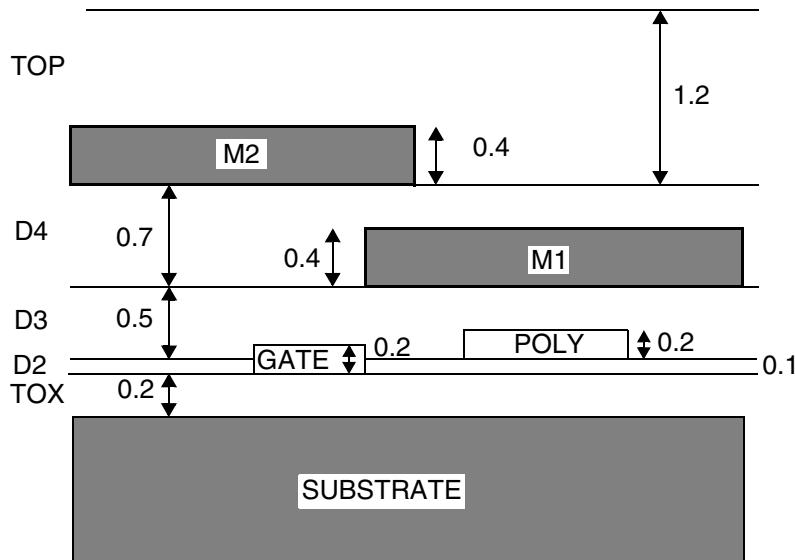


## Gate Poly Process ITF Example

The following example ITF file describes the gate poly process shown in [Figure 14-3](#).

```
TECHNOLOGY = polygate
DIELECTRIC TOP { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M2 { THICKNESS=0.4 WMIN=0.5 SMIN=0.5 RPSQ=0.05
DIELECTRIC D4 { THICKNESS=0.7 ER=3.9}
CONDUCTOR M1 { THICKNESS=0.4 WMIN=0.4 SMIN=0.4 RPSQ=0.05 }
DIELECTRIC D3 { THICKNESS=0.5 ER=3.9 }
CONDUCTOR POLY { THICKNESS=0.2 WMIN=0.3 SMIN=0.3 RPSQ=10.0 }
DIELECTRIC D2 { THICKNESS=0.1 ER=3.9 }
CONDUCTOR GATE { THICKNESS=0.2 WMIN=0.2 SMIN=0.2 RPSQ=8.0 }
DIELECTRIC TOX { THICKNESS=0.2 ER=3.9 }
VIA DIFF_CONT { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA POLY_CONT { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA V1 { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

*Figure 14-3 Gate Poly Process*



## Local Interconnect Process ITF Example

The following example ITF file describes the local interconnect process shown in [Figure 14-4](#).

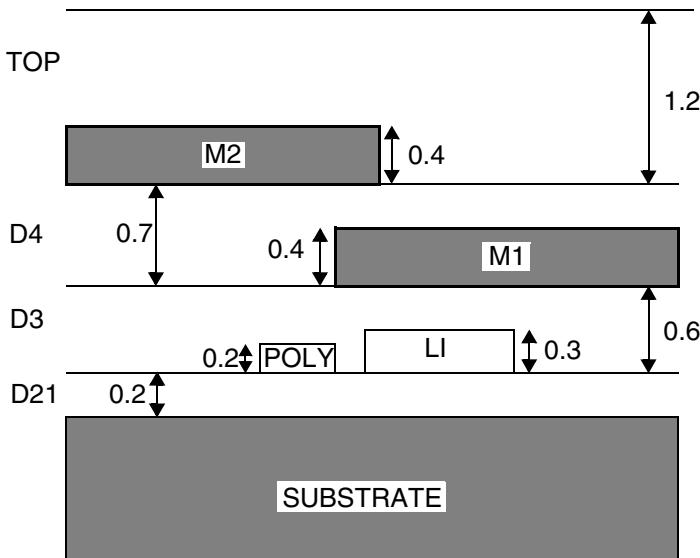
```
TECHNOLOGY = polyli
DIELECTRIC TOP { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M2 { THICKNESS=0.4 WMIN=0.5 SMIN=0.5 RPSQ=0.05
DIELECTRIC D4 { THICKNESS=0.7 ER=3.9 }
$ D4 thickness measured from D3
CONDUCTOR M1 { THICKNESS=0.4 WMIN=0.4 SMIN=0.4 RPSQ=0.05 }
DIELECTRIC D3 { THICKNESS=0.6 ER=3.9 }
$ D3 thickness measured from D21
CONDUCTOR LI { THICKNESS=0.3 WMIN=0.4 SMIN=0.4 RPSQ=1 }
$ LI thickness measured from top of D21
CONDUCTOR POLY { THICKNESS=0.2 WMIN=0.2 SMIN=0.2 RPSQ=10.0 }

$ POLY thickness measured from top of D21

DIELECTRIC D21 { THICKNESS=0.2 ER=3.9 }

VIA LI_SUB { FROM=SUBSTRATE TO=LI AREA=0.25 RPV = 4 }
VIA CONT { FROM=LI TO=M1 AREA=0.25 RPV=5 }
VIA V1 { FROM=M1 TO=M2 AREA=0.25 RPV=4 }
```

*Figure 14-4 Local Interconnect*



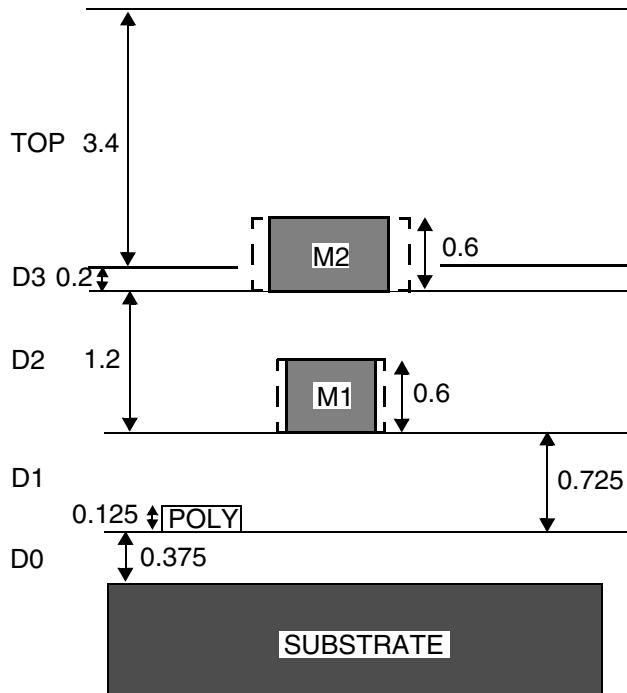
## Layer Etch Process ITF Example

The following example ITF file describes the layer etch process shown in [Figure 14-5](#).

```
TECHNOLOGY = etch
DIELECTRIC TOP { THICKNESS=3.4 ER=3.9 }
DIELECTRIC D3 { THICKNESS=0.2 ER=3.9 }
CONDUCTOR M2 {
 THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05
 ETCH=0.1 }
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05
 ETCH=0.05 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY{ THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0 { THICKNESS=0.375 ER=3.9 }

VIA sub_tie { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA poly_cont { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA via { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

*Figure 14-5 Layer Etch Process*



## Emulated Metal Fill Process ITF Example

The following example ITF file describes the metal fill process shown in [Figure 14-6](#).

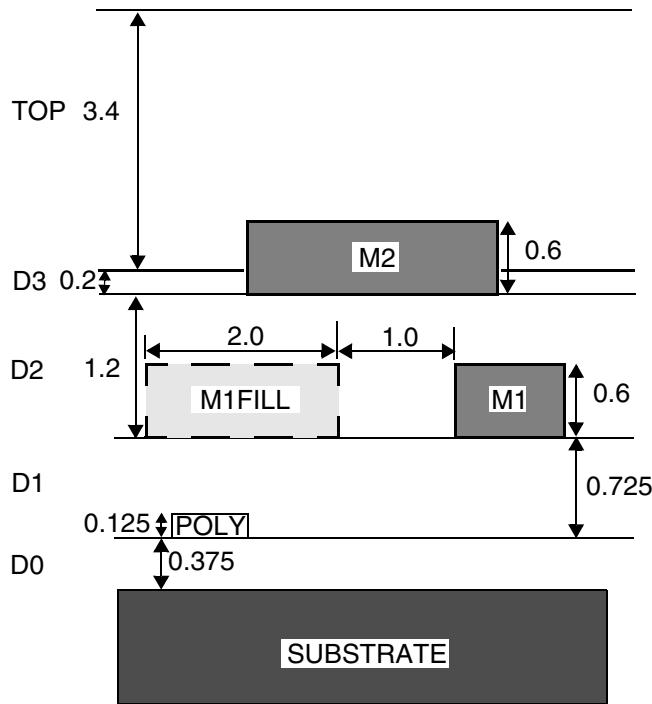
```

TECHNOLOGY = metal_fill
DIELECTRIC TOP { THICKNESS=3.4 ER=3.9 }
DIELECTRIC D3 { THICKNESS=0.2 ER=3.9 }
CONDUCTOR M2 {
 THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05
}
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05
 FILL_RATIO=0.4 FILL_SPACING=1.0 FILL_WIDTH=2.0
}
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY{ THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0 {THICKNESS = 0.375 ER = 5.2}

VIA sub_tie { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA poly_cont { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA via { FROM=M1 TO=M2 AREA=0.36 RPV=4 }

```

*Figure 14-6 Metal Fill Process (Emulated)*



## Transistor-Level Process ITF Example

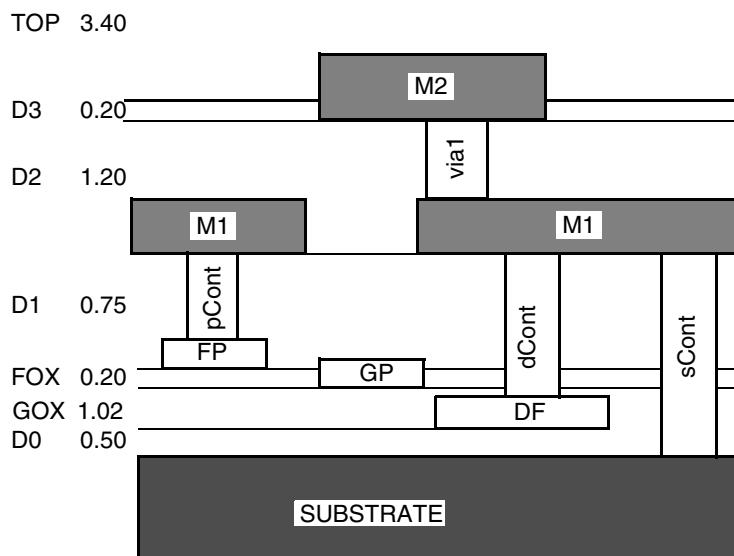
The following example ITF file describes the transistor-level process shown in [Figure 14-7](#).

```
TECHNOLOGY=xtor
DIELECTRIC TOP { THICKNESS=3.40 ER=4.3 }
DIELECTRIC D3 { THICKNESS=0.20 ER=3.9 }
CONDUCTOR M2 { THICKNESS=0.60 WMIN=0.55 SMIN=0.50 RPSQ=0.062
}
DIELECTRIC D2 { THICKNESS=1.20 ER=3.9 }
CONDUCTOR M1 { THICKNESS=0.60 WMIN=0.50 SMIN=0.45 RPSQ=0.062
}

DIELECTRIC D1 { THICKNESS=0.75 ER=3.9 }
CONDUCTOR FP{ THICKNESS=0.30 WMIN=0.35 SMIN=0.40 RPSQ=3.200
}
DIELECTRIC FOX { THICKNESS=0.20 ER=3.9 }
CONDUCTOR GP{ THICKNESS=0.30 WMIN=0.35 SMIN=0.40
RPSQ=3.200 }
DIELECTRIC GOX { THICKNESS=1.02 ER=5.0 }
CONDUCTOR DF{ THICKNESS=1.00 WMIN=1.00 SMIN=0.35
RPSQ=10.00 }
DIELECTRIC D0 { THICKNESS=0.50 ER=3.9 }

VIA via1 { FROM=M1 TO=M2 RHO=0.263 }
VIA pCont { FROM=FP TO=M1 RHO=0.352 }
VIA dCont { FROM=DF TO=M1 RHO=0.500 }
VIA sCont { FROM=SUBSTRATE TO=M1 RHO=0.600 }
```

*Figure 14-7 Transistor-Level Process*



---

## Through-Silicon Via Process ITF Example

The following example ITF file describes the through-silicon via process shown in [Figure 14-8](#). The TSV statement must be placed after the frontside ITF statements and before the backside ITF statements.

```

TECHNOLOGY = tsv_process
GLOBAL_TEMPERATURE = 25.0

$ Frontside ITF statements
CONDUCTOR M1 {
 THICKNESS=0.3 WMIN=0.12 SMIN=0.12 SIDE_TANGENT=0.2
 CRT1=7.0e-03 CRT2=-8.0e-07
}
DIELECTRIC ILD_B { ER=5.5 THICKNESS=0.5 }
DIELECTRIC GATE_OXIDE { ER=4.3 THICKNESS=0.03 }
DIELECTRIC FOXIDE_A { ER=5.0 THICKNESS=0.1 }
CONDUCTOR DIFFUSION {
 THICKNESS=0.1 WMIN=0.1 SMIN=0.06
 CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
}
DIELECTRIC FOXIDE { ER=5.0 THICKNESS=0.2 }
VIA vial { FROM=M1 TO=M2 AREA=0.03 RPV=2.5 CRT1=3.0e-04 CRT2=-6.0e-06 }

$ TSV statement
TSV tsv {
 FROM=M1 TO=M1b RHO=0.05 AREA=49.0 THICKNESS=20.0
 INSULATION_THICKNESS = 0.4 INSULATION_ER = 5.5
 CRT1=7.0e-03 CRT2=-3.0e-08
}

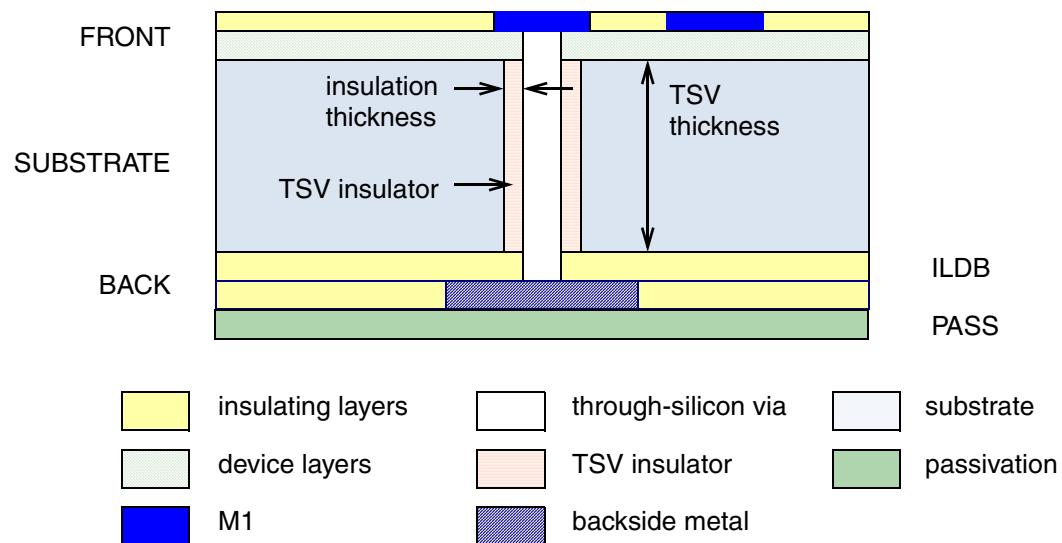
$ Backside ITF statements
DIELECTRIC PASS { ER=9.0 THICKNESS=2.0 }
DIELECTRIC DIELb { ER=5.0 THICKNESS=1.0 }
CONDUCTOR M1b {
 THICKNESS=2.0 WMIN=4.0 SMIN=4.0 SIDE_TANGENT=-0.2
 CRT1=1.0e-03 CRT2=-7.0e-07
}
DIELECTRIC ILDB{ ER=5.2 THICKNESS=1.0 }

```

**Note:**

The locations of the backside dielectric layers ILDB and PASS with respect to the substrate are shown in [Figure 14-8](#).

Figure 14-8 Cross Section of Through-Silicon Via



---

## Trench Contact Process ITF Example

The following example shows a trench contact process definition in the ITF file.

*Example 14-1 Trench Contact Process Definition in the ITF File*

```
CONDUCTOR TC {
 THICKNESS=0.05 WMIN=0.025 SMIN=0.060 RPSQ=1.0
 GATE_TO_CONTACT_SMIN=0.02
 LAYER_TYPE=TRENCH_CONTACT
}

DIELECTRIC D4 {THICKNESS=0.015 ER=4.5 MEASURED_FROM=D3 }

DIELECTRIC DC_POLY {
 THICKNESS=0.0 ER=7.0
 IS_CONFORMAL ASSOCIATED_CONDUCTOR=POLY
 SW_T=0.01 TW_T=0.005
}

CONDUCTOR POLY {
 THICKNESS=0.03 WMIN=0.045 SMIN=0.045 RPSQ=1.0
 GATE_TO_CONTACT_SMIN=0.02
 LAYER_TYPE=GATE
}
DIELECTRIC D3 {THICKNESS=0.001 ER=2.8 }
DIELECTRIC D2 {THICKNESS=0.05 ER =3.4 }

CONDUCTOR DIFF {
 THICKNESS=0.05 SMIN=0.045 WMIN=0.06 RPSQ=1.0
 RAISED_DIFFUSION_THICKNESS = 0.015
 RAISED_DIFFUSION_TO_GATE_SMIN = 0.014
 RAISED_DIFFUSIONETCH = 0.010
 LAYER_TYPE=DIFFUSION
}

DIELECTRIC D1 { THICKNESS=0.1 ER=6.8 }
```



## **Part III: StarRC Command Reference**

---

---



# 15

## StarRC Commands

---

This chapter describes the commands that you can use in the StarRC command file.

---

## 3D\_IC

Enables the 3-D IC flow.

### Syntax

3D\_IC: YES | NO

### Arguments

| Argument     | Description              |
|--------------|--------------------------|
| YES          | Enables the 3-D IC flow  |
| NO (default) | Disables the 3-D IC flow |

### Description

The 3D\_IC command enables the 3-D IC flow.

### See Also

- [TSV](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [TSV\\_ CELLS](#)
- [Through-Silicon Via Extraction](#)

---

## 3D\_IC\_FILTER\_DEVICE

Specifies the device to be flattened in the netlist during 3-D IC extraction.

### Syntax

3D\_IC\_FILTER\_DEVICE: *device\_name*

### Arguments

| Argument    | Description                                                                  |
|-------------|------------------------------------------------------------------------------|
| device_name | Specifies the device to be flattened in the netlist during 3-D IC extraction |

### Description

The 3D\_IC\_FILTER\_DEVICE command specifies the device that should be flattened in the netlist during 3-D IC extraction.

For 3-D IC extraction, run LVS on each die separately. If you run LVS on several die together, add a pseudo-resistor device at the ports to separate them from the net in interposer and prevent shorts. There is no device in the interposer.

### Example

The following syntax specifies the name of the device which should be flattened in the netlist during 3-D IC extraction.

3D\_IC\_FILTER\_DEVICE: DEVICE\_1

### See Also

- [TSV](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [TSV\\_ CELLS](#)
- [Through-Silicon Via Extraction](#)

---

## 3D\_IC\_FLOATING\_SUBSTRATE

Specifies the TSV substrate as a floating conductor.

### Syntax

3D\_IC\_FLOATING\_SUBSTRATE: YES | NO

### Arguments

| Argument     | Description                                                  |
|--------------|--------------------------------------------------------------|
| YES          | Specifies the TSV substrate as a floating conductor          |
| NO (default) | Specifies that the TSV substrate is not a floating conductor |

### Description

The 3D\_IC\_FLOATING\_SUBSTRATE command enables StarRC to model the TSV substrate in the interposer as a floating conductor. The grdgenxo tool ignores its effect on modeling. You do not need to specify the substrate layer in the ITF file; its thickness is determined by the HEIGHT parameter in the TSV statement.

StarRC uses name substitution for this floating ground. When 3D\_IC\_FLOATING\_SUBSTRATE is YES, the NETLIST\_GROUND\_NODE\_NAME is internally set to fsub\_starrc and all zero ground nets in the netlist are replaced by fsub\_starrc.

This feature is supported in SPEF netlists because the SPEF does not netlist ground nets explicitly.

### Example

The following syntax specifies the TSV substrate as a floating conductor.

3D\_IC\_FLOATING\_SUBSTRATE: YES

### See Also

- [TSV](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [TSV\\_ CELLS](#)

---

## 3D\_IC\_SUBCKT\_FILE

Specifies the subcircuit used to replace TSV and microbump extraction results.

### Syntax

3D\_IC\_SUBCKT\_FILE: *filename1 filename2 ...*

### Arguments

| Argument             | Description                                                                              |
|----------------------|------------------------------------------------------------------------------------------|
| <i>filename1 ...</i> | The files containing the subcircuit used to replace TSV and microbump extraction results |

### Description

The 3D\_IC\_SUBCKT\_FILE command specifies the files containing the subcircuit used to replace TSV and microbump extraction results.

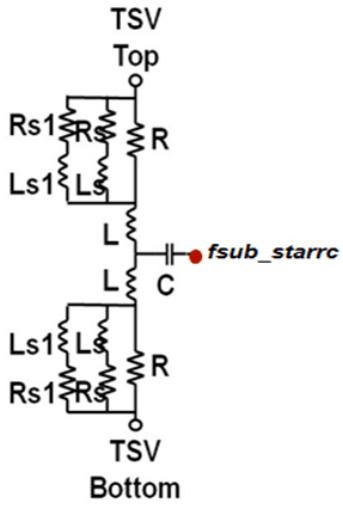
The StarRC tool supports 1R, 2R, and 2R1C models. When using these models, connect the TSV subcircuit netlist from the front side to the backside metals. This modeling also applies to microbump replacements. Specify the via names or cell names in the subcircuit file. The following example shows a 2R1C subcircuit model:

```
subckt via_name or cell_name top bottom
r1 top n1 0.0126
c1 n1 0 5.678e-3
r2 n1 bottom 0.0126
.ends
```

### RLC Model Replacement

The tool supports RLC models for TSV model replacement in the TSV-as-VIA flows; microbumps are not supported. When modeling, connect the TSV subcircuit netlist from the front side to the backside metals. The RLC model must be symmetric.

[Figure 15-1](#) show an RLC subcircuit model.

*Figure 15-1 RLC Subcircuit Model*

```
.subckt TSV top bottom
 rs1_tsv top n11 0.0123
 rs1_tsv top n1 0.0711
 r1_tsv top n2 0.1234
 ls11_tsv n11 n2 1.1111E-11
 ls1_tsv n1 n2 4.2300E-12
 l1_tsv n2 n3 1.5678E-11
 C_tsv n3 0 155E-15
 l2_tsv n3 n4 1.2578E-11
 ls2_tsv n4 n5 4.2300E-12
 ls21_tsv n4 n6 1.1111E-11
 r2_tsv n4 bottom 0.1234
 rs2_tsv n5 bottom 0.0711
 rs21_tsv n6 bottom 0.0123
.ends
```

## Example

The following syntax specifies the TSV substrate as a floating conductor.

```
3D_IC_SUBCKT_FILE: SUBCKT1 SUBCKT2
```

## See Also

- [TSV](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [TSV\\_ CELLS](#)
- [Through-Silicon Via Extraction](#)

---

## 3D\_IC\_TSV\_COUPLING\_EXTRACTION

Considers substrate effects during high frequency extractions.

### Syntax

`3D_IC_TSV_COUPLING_EXTRACTION: RCSUB | CEFF | NONE`

### Arguments

| Argument       | Description                                                            |
|----------------|------------------------------------------------------------------------|
| RCSUB          | Uses $R_{sub}$ and $C_{sub}$ for SPICE simulation                      |
| CEFF           | Enables $C_{eff}$ model for static timing analysis simulation          |
| NONE (default) | Does not consider substrate effects during high frequency extractions. |

### Description

This command specifies the output model when extracting high frequency substrate effects.

When you want to use  $R_{sub}$  and  $C_{sub}$  for SPICE simulation, set

`3D_IC_TSV_COUPLING_EXTRACTION: RCSUB` and specify a file with the  
`3D_IC_SUBCKT_FILE` command.

When you want to use  $C_{eff}$  for static timing analysis simulation, use the

`3D_IC_TSV_COUPLING_EXTRACTION: CEFF`, `OPERATING_FREQUENCY`, and  
`3D_IC_SUBCKT_FILE` commands.

The `3D_IC_TSV_COUPLING_EXTRACTION: CEFF` command should not be used with the  
`NETLIST_FORMAT: SPEF` command.

### TSV Coupling Extraction

`3D_IC_TSV_COUPLING_EXTRACTION: RCSUB`

StarRC evaluates the spacing between two TSV pairs and determines the  $R_{sub}$  and  $C_{sub}$  values from the ITF table. The netlist is a parallel  $R_{sub}$  and  $C_{sub}$  network connecting the two TSVs. The netlist is used for SPICE simulation.

`3D_IC_TSV_COUPLING_EXTRACTION: CEFF`

When using the `CEFF` setting, use the `OPERATING_FREQUENCY` command to specify the operating frequency. The StarRC tool evaluates the spacing between the two TSV pairs and determines the  $C_{eff}$  value based on the frequency from the ITF table. The netlist has a single

Ceff value and connects between two TSVs. You can use this netlist with SPICE and static timing analysis tools.

## Examples

The following example enables static timing analysis simulation.

```
3D_IC_TSV_COUPLING_EXTRACTION: CEFF
3D_IC_SUBCKT_FILE = STA.txt
```

## See Also

- [TSV](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [TSV\\_ CELLS](#)
- [Through-Silicon Via Extraction](#)

---

## **ANALOG\_SYMMETRIC\_NETS**

Enables the analog symmetric nets extraction mode.

### **Syntax**

`ANALOG_SYMMETRIC_NETS: NO | YES`

### **Arguments**

| Argument     | Description                                        |
|--------------|----------------------------------------------------|
| NO (default) | Disables the analog symmetric nets extraction mode |
| YES          | Enables the analog symmetric nets extraction mode  |

### **Description**

The `ANALOG_SYMMETRIC_NETS` command enables a special extraction mode for analog symmetric nets, such as differential signals. This extraction mode is supported only for transistor-level extraction, not for the gate-level Milkyway or LEF/DEF flows.

In this extraction mode, the StarRC tool extracts symmetric total and coupling capacitance values that are independent of design orientation (rotation and flip). the StarRC tool identifies the width, spacing, and density for each polygon individually and uses similar capacitance models to ensure that the extraction results of the symmetric nets are very close to each other.

Many advanced process node designs, including FinFET and trench contact designs, no longer require the `ANALOG_SYMMETRIC_NETS` command. StarRC issues warning messages when the `ANALOG_SYMMETRIC_NETS` command is present but not needed in the StarRC command file for a specific design.

### **Example**

Use the following command when you need consistent total and coupling capacitances for the symmetric nets in your design:

```
ANALOG_SYMMETRIC_NETS: YES
```

---

## AUTO\_RUNSET

Automatically performs the necessary modifications internally for the separation of device layers based on device definitions in the LVS rule file.

### Syntax

AUTO\_RUNSET: NO | YES

### Arguments

| Argument     | Description                                                                                                                                                                                                            |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO (default) | Processes device layers directly from the LVS database without any detection or modification of layer separation                                                                                                       |
| YES          | Enables automatic device layer separation for necessary devices such as MOS devices and capacitors. Generates the corresponding statements for such devices by separating layers based on IGNORE_CAPACITANCE settings. |

### Description

Normally, StarRC uses device extraction data from LVS flows such as IC Validator, Hercules, and Calibre for device-level parasitic extraction. To ensure accurate extraction results, a rule file for StarRC parasitic extraction flows as well as device extraction and LVS flows must abide by the following conventions:

- MOS device and capacitor terminal layers must be distinct from the routing layers that form those terminal layers, and no polygon overlap should occur between those layers. This ensures that StarRC properly excludes all intradevice capacitances that are represented by device models during simulation.
- All contacts must connect exactly two physical layers to ensure uniformity with the physical processes as defined in the StarRC ITF file.

To satisfy these conventions required by the StarRC device-level extraction flow, the LVS rule files must be updated. If you specify the AUTO\_RUNSET command, StarRC internally performs the necessary modifications to your LVS rule file automatically, based on device definitions in the LVS rule file.

## Example

The following syntax shows a typical Calibre rule file that uses the polysilicon interconnect layer directly as the MOS gate terminal layer:

```
gate = poly AND diff
gatenw = gate NOT nwell
ngate = gatenw AND nplus
DEVICE MN(n) ngate poly (G) ndiff (S) ndiff (D) psub (B)
```

In this example, the interconnect layer poly itself serves as the gate terminal layer, while ngate serves as the unconnected recognition (seed) layer. Since the terminal layer derivations violate the tool's assumption that the gate terminal layer represents only the gate area, StarRC erroneously ignores the capacitance for portions of poly that do not serve as part of the gate, that is, all capacitance between the poly interconnect and the device diffusions. StarRC then generates the following instructions for the extraction engine to ignore intradevice capacitance:

```
IGNORE_CAPACITANCE poly ndiff
IGNORE_CAPACITANCE poly psub
```

You can solve this problem by using the AUTO\_RUNSET command, which performs internal layer separation operations to differentiate the poly serving as the gate terminal from the poly remaining as the routing layer.

### Note:

The AUTO\_RUNSET command can be used only if the recognition layer, which is ngate in this example, exists in the input physical database.

## See Also

- [IGNORE\\_CAPACITANCE](#)

---

## BLOCK

Defines the layout block name that is to be extracted.

### Syntax

`BLOCK: block_name`

### Arguments

| Argument                       | Description                                                   |
|--------------------------------|---------------------------------------------------------------|
| <code><i>block_name</i></code> | The layout name of the block to be extracted<br>Default: none |

### Description

For Milkyway gate-level place and route designs, this option is mandatory. Valid arguments are top-level blocks or fully placed and routed core blocks that exist in the Milkyway library.

For LEF/DEF gate-level designs, this option is invalid. The block to be extracted is determined by the DESIGN keyword in the DEF file specified by the `TOP_DEF_FILE` command.

For Hercules gate-level or device-level flows, this option is mandatory. Valid arguments for this flow are dependent on the settings of the `XREF` and `CELL_TYPE` commands. With `CELL_TYPE: LAYOUT`, which is the default, valid arguments are any cell that exists in the `WRITE_EXTRACT_VIEW` library from Hercules. With `CELL_TYPE: SCHEMATIC` and `XREF: YES` or `XREF: COMPLETE`, valid arguments are any valid equivalence point from Hercules compare.

For Calibre gate-level or device-level flows, this option is mandatory. Valid arguments for this flow are dependent on the settings the `XREF` and `CELL_TYPE` commands. With `CELL_TYPE: LAYOUT`, which is the default, valid arguments are any cell that exists in the annotated GDSII file and layout netlist file (.nl). With `CELL_TYPE: SCHEMATIC` and `XREF: YES`, valid arguments are any valid HCELL from Calibre compare (.ixf).

### Example

This example specifies INT4 as the block to be extracted:

`BLOCK: INT4`

To override the `BLOCK` statement for a particular run, you can specify the `StarXtract` command line option `-b`.

`% StarXtract -b SMALLARRAY`

As shown in the previous example, specifying the `-b` option causes StarRC to use `SMALLARRAY` as the block to extract rather than the block that is defined in the command file. The command file is not changed, and the override exists for that run only.

## See Also

- [CELL\\_TYPE](#)
- [MILKYWAY\\_DATABASE](#)
- [MILKYWAY\\_EXTRACT\\_VIEW](#)
- [TOP\\_DEF\\_FILE](#)
- [XREF](#)

---

## BLOCK\_BOUNDARY

Defines a boundary for the block specified by the `BLOCK` command.

### Syntax

`BLOCK_BOUNDARY: x1 y1 x2 y2 [... xn yn]`

### Arguments

| Argument        | Description                               |
|-----------------|-------------------------------------------|
| <code>x1</code> | The first x-coordinate in database units  |
| <code>y1</code> | The first y-coordinate in database units  |
| <code>x2</code> | The second x-coordinate in database units |
| <code>y2</code> | The second y-coordinate in database units |

### Description

The `BLOCK_BOUNDARY` command defines a boundary for the block specified by the `BLOCK` command when you use the `RING_AROUND_THE_BLOCK` command for in-context approximation. The `BLOCK_BOUNDARY` command is required when the `RING_AROUND_THE_BLOCK` command is specified for a LEF/DEF flow. The `BLOCK_BOUNDARY` information is used by StarRC to build the in-context routing approximation rings for preplacement block noise analysis.

Specify the coordinates in database units, not microns. You can specify an arbitrary number of boundary points. Two points (`x1, y1`)`(x2, y2)` define a rectangular block boundary. If you specify more than two points, this specifies a rectilinear, or nonrectangular, block boundary. The points you specify must be in counterclockwise order around the boundary of the block. You can specify a closing point, but it is not required. This command and coordinates can be specified only one time in the StarRC command file.

It is not necessary to use `BLOCK_BOUNDARY` when the database is Milkyway, because the boundary information can be read directly. However, if specified in a Milkyway flow, the `BLOCK_BOUNDARY` command overrides the internal database value.

### Example

The following example shows how to specify a block boundary with four points:

```
BLOCK_BOUNDARY: 0 0 269.6 0 269.6 137.6 0 137.6
```

If the following error message appears, you should provide at least four coordinates of the block boundary irrespective of the shape in counterclockwise direction from the DEF file in database units to the `BLOCK_BOUNDARY` command:

```
ERROR: BLOCK_BOUNDARY must have at least 4 points
```

### See Also

- [BLOCK](#)
- [RING\\_AROUND\\_THE\\_BLOCK](#)

---

## BUS\_BIT

Specifies the character or pair of characters used as the bus bit delimiter during extraction and netlist creation.

### Syntax

`BUS_BIT: {} | [] | () | <> | : | .`

### Arguments

| Argument              | Description                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <code>{}</code>       | Characters used as the bus bit delimiters. Do not insert spaces between the characters in the string |
| <code>[]</code>       | .                                                                                                    |
| <code>()</code>       | .                                                                                                    |
| <code>&lt;&gt;</code> | Default: Database BUSBIT value                                                                       |
| <code>:</code>        | .                                                                                                    |
| <code>.</code>        | .                                                                                                    |

### Description

The `BUS_BIT` command specifies the character or pair of characters used as the bus bit delimiter during extraction and netlist creation.

The value of this option affects net selection and the Standard Parasitic Exchange Format (SPEF) `*BUS_DELIMITER` header statement that is read by follow-on tools. Any literal occurrence of these characters in a net or instance name should be escaped during net selection; attempting to use an illegal delimiter results in an error.

For a LEF/DEF database, the `BUS_BIT` characters are defined by the LEF/DEF default specification or database definition of the `BUSBITCHARS` keyword in the LEF/DEF database.

This command does not do character substitution in the output netlist; it affects only selection and escaping.

The `BUS_BIT` command does not define the bus bit character in the final netlist. To make this change in the netlist, you must change the input file or post process the netlist with a script.

### See Also

- [NETS](#)
- [OA\\_BUS\\_BIT](#)

---

## CALIBRE\_LVS\_DEVICE\_TYPE\_CAP

Identifies user-defined capacitor devices for the Calibre LVS flow.

### Syntax

CALIBRE\_LVS\_DEVICE\_TYPE\_CAP: *list\_of\_C\_devices*

### Arguments

| Argument                 | Description                      |
|--------------------------|----------------------------------|
| <i>list_of_C_devices</i> | List of user-defined CAP devices |

### Description

This command identifies user-defined capacitors as capacitor devices.

The *list\_of\_C\_devices* argument follows the case-sensitivity set by the CASE\_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts.

### Example

CALIBRE\_LVS\_DEVICE\_TYPE\_CAP: cap\_ss

### See Also

- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_MOS](#)
- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_RES](#)
- [CALIBRE\\_OPTIONAL\\_DEVICE\\_PIN\\_FILE](#)

---

## CALIBRE\_LVS\_DEVICE\_TYPE\_MOS

Identifies user-defined MOS devices for the Calibre LVS flow.

### Syntax

CALIBRE\_LVS\_DEVICE\_TYPE\_MOS: *list\_of\_M\_devices*

### Arguments

| Argument                 | Description                      |
|--------------------------|----------------------------------|
| <i>list_of_M_devices</i> | List of user-defined MOS devices |

### Description

This command identifies user-defined MOS devices.

The *list\_of\_M\_devices* argument follows the case-sensitivity set by the CASE\_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts in certain situations.

### Example

CALIBRE\_LVS\_DEVICE\_TYPE\_MOS: nmos\_ss

### See Also

- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_CAP](#)
- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_RES](#)
- [CALIBRE\\_OPTIONAL\\_DEVICE\\_PIN\\_FILE](#)

---

## CALIBRE\_LVS\_DEVICE\_TYPE\_RES

Identifies user-defined resistors as resistor devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command. Valid for the Calibre LVS flow.

### Syntax

`CALIBRE_LVS_DEVICE_TYPE_RES: list_of_R_devices`

### Arguments

---

| Argument                       | Description                      |
|--------------------------------|----------------------------------|
| <code>list_of_R_devices</code> | List of user-defined RES devices |

---

### Description

The `CALIBRE_LVS_DEVICE_TYPE_RES` statement identifies user-defined resistors as resistor devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command.

### Example

In the following example, the `rp_sio` and `pwr_rm1` devices defined in the rule file are identified as resistors:

`CALIBRE_LVS_DEVICE_TYPE_RES: rp_sio pwr_rm1`

### See Also

- [CALIBRE\\_OPTIONAL\\_DEVICE\\_PIN\\_FILE](#)
- [WIDE\\_DEVICE\\_TERM\\_RESISTANCE](#)

---

## CALIBRE\_OPTIONAL\_DEVICE\_PIN\_FILE

Assigns nonstandard device terminals by name for the Calibre LVS flow.

### Syntax

`CALIBRE_OPTIONAL_DEVICE_PIN_FILE: file_name`

### Arguments

| Argument               | Description                 |
|------------------------|-----------------------------|
| <code>file_name</code> | Name of the device pin file |

### Description

By default, StarRC assigns nonstandard device terminals by position. However, if you specify the `CALIBRE_OPTIONAL_DEVICE_PIN_FILE` command, StarRC assigns the device terminal by the name in the specified file.

The syntax of each line in the file is as follows:

```
device_type model_name[seed_layer] pin_layer_1 (pin name 1) \
 pin_layer_2 (pin name 2) pin_layer_3 (pin name 3) ...
```

Valid values for the `device_type` parameter are M (MOS device), C (capacitor), and R (resistor).

The seed layer is optional, but it must be used when multiple devices have the same model name.

### Example

In the following example, a file named `devpin_file` contains the list of device terminal names:

```
CALIBRE_OPTIONAL_DEVICE_PIN_FILE: devpin_file
```

Example lines within a device pin file are as follows:

```
M MOS_DEV NDIFSI_D (D) POLY (G) NDIFSI_S (S) NWELL (B)
C CAP_DEV[met1] CAP_TOP (PLUS) CAP_M1 (MINUS)
C CAP_DEV[met2] CAP_TOP (PLUS) CAP_M2 (MINUS)
```

### See Also

- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_CAP](#)
- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_MOS](#)
- [CALIBRE\\_LVS\\_DEVICE\\_TYPE\\_RES](#)

---

## CALIBRE\_PDBA\_FILE

Specifies the use of data from a Calibre Push Down Back-Annotation (PDBA) file.

### Syntax

`CALIBRE_PDBA_FILE: file_name`

### Arguments

| Argument         | Description                                                    |
|------------------|----------------------------------------------------------------|
| <i>file_name</i> | File containing devices and device properties<br>Default: none |

### Description

The `CALIBRE_PDBA_FILE` command reads a Calibre PDBA file. This file is generated by the Calibre tool; it lists devices and device properties. Use the `CALIBRE_PDBA_FILE` command to retrieve the separated properties for a final parasitic netlist with complete device information.

The `CALIBRE_PDBA_FILE` command might not be necessary, depending on the tool versions and the flow.

- If you are using the StarRC-CCI Push Down Back-annotation (PDBA) flow with StarRC version 2013.12-1 (or later) and Calibre version 2013.4-15.12 (or later), StarRC can obtain the PDBA information from the query file and the LVS extraction report file. To use this capability, include the following command in the Calibre query file:

```
LVS SETTINGS REPORT WRITE cci_file
```

Include the following commands in the Calibre runset:

```
LVS PUSH DEVICES YES
LVS PUSH DEVICES SEPARATE PROPERTIES "pdba.data" AGF
```

You do not need to include the `CALIBRE_PDBA_FILE` command in the StarRC command file because the query file finds it automatically. If the command file includes a `CALIBRE_PDBA_FILE` command, the file specified by the command takes precedence over the PDBA file from the query output.

- If you are using the StarRC-CCI Push Down Back-annotation (PDBA) flow with earlier versions of either the StarRC or Calibre tools, include the `CALIBRE_PDBA_FILE` command in the StarRC command file and the following commands in the Calibre runset:

```
LVS PUSH DEVICES YES
LVS PUSH DEVICES SEPARATE PROPERTIES "pdba.data" AGF
```

- If you are using the StarRC-CCI Push Down Separated Properties (PDSP) flow, do not include the `CALIBRE_PDBA_FILE` command in the StarRC command file and include the following commands in the Calibre runset:

```
LVS PUSH DEVICES YES
LVS PUSH DEVICES SEPARATE PROPERTIES YES
```

The StarRC tool automatically uses the PDSP file from the Calibre query output. If the StarRC command file includes a `CALIBRE_PDBA_FILE` command, the file specified by the command takes precedence over the PDSP file from the query output.

If the `CALIBRE_PDBA_FILE` command is used, the `NETLIST_IDEAL_SPICE_HIER` command is automatically set to `NO` because the layout netlist cannot be maintained as hierarchical when the StarRC tool integrates DFM properties into the final parasitic netlist.

When you use the `CALIBRE_PDBA_FILE` command, the `SKIP_CELLS` command might fail because the DFM properties annotation can promote the instance hierarchy.

### Example

`CALIBRE_PDBA_FILE: ./pdः.data`

### See Also

- [SKIP\\_CELLS](#)
- [LVS\\_EXTRACTION\\_REPORT\\_FILE](#)

---

## CALIBRE\_QUERY\_FILE

Specifies the location of Calibre Connectivity Interface input files.

### Syntax

CALIBRE\_QUERY\_FILE: *query\_script\_file*

### Arguments

| Argument                 | Description                                                                           |
|--------------------------|---------------------------------------------------------------------------------------|
| <i>query_script_file</i> | The command file used with the Calibre Connectivity Interface server<br>Default: none |

### Description

To have the Calibre Connectivity Interface generate all the files needed for a StarRC extraction, all the necessary query commands must be in the query command file specified by the CALIBRE\_QUERY\_FILE command.

For details about the Calibre Connectivity Interface, see [The Calibre Connectivity Interface Flow](#). For details about Calibre query commands, see [Running the Calibre Query Server for Output to StarRC](#).

### Example

Use the command as shown here. An example of a Calibre query file follows.

CALIBRE\_QUERY\_FILE: query.cmd

### Note:

The LNXF construct of the NET XREF WRITE command and the EXPAND\_CELLS of the LAYOUT NAMETABLE WRITE command require Calibre version 2014.3 or later.

The following example shows a Calibre query command file for use with StarRC.

**Example 15-1 Calibre Query File**

```
Define property numbers in annotated GDSII
GDS NETPROP NUMBER 5
GDS PLACEPROP NUMBER 6
GDS DEVPROP NUMBER 7

Output seed polygon with net ID
GDS SEED PROPERTY ORIGINAL

Output annotated GDSII mapping file for StarRC
RESPONSE FILE GDS_MAP
GDS MAP
RESPONSE DIRECT

Output annotated GDSII file for StarRC
GDS WRITE agf

Output device table file containing device layer descriptions
RESPONSE FILE devtab_file
DEVICE TABLE
RESPONSE DIRECT

Output layout net name and net ID mapping table for StarRC
The EXPAND_CELLS keyword for LAYOUT NAMETABLE WRITE ensures that the
lnn file and the netlist have the same hierarchy
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NAMETABLE WRITE lnn_file EXPAND_CELLS

Output ideal layout netlist for StarRC
AGF is the only allowed keyword for LAYOUT NETLIST HIERARCHY
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST PIN LOCATIONS YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE netlist_file

Output net or device instance cross referencing tables for StarRC
NET XREF WRITE nxf_file LNXF
INSTANCE XREF WRITE ixf_file

Output ports file for StarRC
PORT TABLE CELLS WRITE ports_cells_file

Output cell extents file for StarRC
CELL EXTENTS WRITE extents.txt
```

**See Also**

- [CALIBRE\\_RUNSET](#)

---

## CALIBRE\_RUNSET

Specifies the LVS command file used for the Calibre run.

### Syntax

CALIBRE\_RUNSET: *lvs\_command\_file*

### Arguments

| Argument                | Description                                                    |
|-------------------------|----------------------------------------------------------------|
| <i>lvs_command_file</i> | The LVS command file used for the Calibre run<br>Default: none |

### Description

The CALIBRE\_RUNSET command specifies the LVS command file used for a Calibre run.

An LVS rule deck contains data creation commands, device creation commands, device property calculations, layer connect sequences, and LVS comparison options. The StarRC tool parses the layer connect sequence from the Calibre runset, including derived layer connectivity, to understand the runset layer connectivity.

The CALIBRE\_RUNSET command usage requirements are as follows:

- The command is required if you are using StarRC versions earlier than 2013.12-1.
- The command is required if you are using Calibre versions earlier than 2013.4-15.12.
- If you are using StarRC version 2013.12-1 (or later) and Calibre version 2013.4-15.12 (or later), the StarRC CALIBRE\_QUERY\_FILE command obtains the necessary information from the query file and the LVS extraction report file. The Calibre query file must also use the LVS SETTINGS REPORT WRITE command to write an extraction report.

In this case, the StarRC tool ignores the CALIBRE\_RUNSET and CALIBRE\_PDBA\_FILE commands.

You can read a specific Calibre extraction report instead of the automatically determined report by using the StarRC LVS\_EXTRACTION\_REPORT\_FILE command.

### See Also

- [LVS\\_EXTRACTION\\_REPORT\\_FILE](#)
- [CALIBRE\\_QUERY\\_FILE](#)
- [MAPPING\\_FILE](#)

---

## CAPACITOR\_TAIL\_COMMENTS

Writes geometric information about parasitic capacitors as comments in the netlist. Valid in transistor-level flows only.

### Syntax

CAPACITOR\_TAIL\_COMMENTS: YES | NO

### Arguments

| Argument     | Description                                                       |
|--------------|-------------------------------------------------------------------|
| YES          | Writes capacitor geometric information as comments in the netlist |
| NO (default) | Disables capacitor tail comments                                  |

### Description

The CAPACITOR\_TAIL\_COMMENTS command controls whether geometric information about parasitic capacitors is added to the netlist output. This option is available for the SPF, NETNAME, and STAR netlist formats.

The additional information is as follows:

- Layer numbers (one layer for grounded capacitors, two layers for coupling capacitors)  
The \$lvl argument is a number that appears in the LAYER\_MAP section of the netlist, just after the header. The number is associated with a retained mapping file layer name.
- Instance pin declarations  
For RC extraction, instance pin declarations always appear in the netlist. For capacitance-only extraction, instance pin declarations are not included in the netlist by default. However, if you set the CAPACITOR\_TAIL\_COMMENTS command to YES, the StarRC tool performs RC extraction regardless of the setting of the EXTRACTION command. This is necessary to obtain instance pin information for the capacitors.

If you specify capacitance-only extraction, resistor information is not included in the netlist even if it is calculated in the background as a result of enabling the capacitor tail comments feature. In addition, the runtime and star directory size for capacitance-only extraction are not smaller than the corresponding values for RC extraction.

You can set the CAPACITOR\_TAIL\_COMMENTS and NETLIST\_TAIL\_COMMENTS commands independently. However, the two commands require different restrictions on the REDUCTION command.

When the `CAPACITOR_TAIL_COMMENTS` command is set to YES, the following restrictions apply:

- For capacitance-only extraction, the `REDUCTION` command is automatically set to `LAYER` regardless of any settings in the command file.
- For RC extraction, the allowed `REDUCTION` command settings are `LAYER`, `NO`, and `LAYER_NO_EXTRA_LOOPS`.
- The `EXTRA_GEOMETRY_INFO` command is automatically set to include the `NODE` option. If the command was previously set to `RES`, the new setting is `NODE RES`.
- The `FS_EXTRACT_NETS` command cannot be used.

### Example

The following example shows an SPF netlist for RC extraction with the capacitor tail comments feature enabled, highlighting the information that would not otherwise appear:

```
*|NETZCB 0.000282157PF
*|I(F82 M12 SRC B 0 0.9 1.63) // $llx=0.9 $lly=1.485 $urx=1.12 $ury=1.775 $lvl=37
*|I(F58 M6 SRC B 0 0.9 0.55) // $llx=0.9 $lly=0.55 $urx=1.12 $ury=0.55 $lvl=32
*|S(213 1.025 0.5075) // $llx-0.975 $lly=0.465 $urx=1.075 $ury=0.55 $lvl=89
C29_69 F82 444 1e-18 $lvl1=37 $lvl2=59
C29_70 F58 444 1e-18 $lvl1=32 $lvl2=59
Cg29_78 F58 0 1e-17 $lvl=32
R29_97 F82 213 10 $l=0.485 $2=0.1 $lvl=89
R29_102 213 F58 20 $a=0.0081 $lvl=171
```

The following example shows the same netlist for capacitance-only extraction with the capacitor tail comments feature enabled, highlighting the information that would not otherwise appear:

```
*|NETZCB 0.000282157PF
*|I(F82 M12 SRC B 0 0.9 1.63) // $llx=0.9 $lly=1.485 $urx=1.12 $ury=1.775 $lvl=37
*|I(F58 M6 SRC B 0 0.9 0.55) // $llx=0.9 $lly=0.55 $urx=1.12 $ury=0.55 $lvl=32
*|S(213 1.025 0.5075) // $llx-0.975 $lly=0.465 $urx=1.075 $ury=0.55 $lvl=89
C29_69 F82 444 1e-18 $lvl1=37 $lvl2=59
C29_70 F58 444 1e-18 $lvl1=32 $lvl2=59
Cg29_78 F58 0 1e-17 $lvl=32
```

### See Also

- [NETLIST\\_FILE](#)
- [NETLIST\\_FORMAT](#)
- [EXTRA\\_GEOMETRY\\_INFO](#)
- [REDUCTION](#)
- [NETLIST\\_TAIL\\_COMMENTS](#)

---

## CASE\_SENSITIVE

Specifies the case-sensitivity of net and cell names during selection.

### Syntax

CASE\_SENSITIVE: YES | NO

### Arguments

| Argument      | Description                                                           |
|---------------|-----------------------------------------------------------------------|
| YES (default) | Specifies that cell and net names are case-sensitive during selection |
| NO            | Specifies that cell and net names are not case-sensitive              |

### Description

The CASE\_SENSITIVE command specifies whether net and cell names are case-sensitive during selection. StarRC always retains the case sensitivity of the input database for netlist creation.

If IGNORE\_CASE is set to TRUE in your Hercules runset, then you must specify CASE\_SENSITIVE: NO in your StarRC command file.

### Example

The following syntax specifies that all net selection and cell selection are not case-sensitive.

CASE\_SENSITIVE: NO

### See Also

- [ONLY\\_NETS](#)
- [INSTANCE\\_PORT](#)
- [NETLIST\\_SELECT\\_NETS](#)
- [NETLIST\\_TYPE](#)
- [NETS](#)
- [POWER\\_NETS](#)
- [SKIP\\_CELLS](#)

---

## CELL\_TYPE

Specifies whether layout or schematic cell names are used for data selection.

### Syntax

CELL\_TYPE: LAYOUT | SCHEMATIC

### Arguments

| Argument         | Description                                                              |
|------------------|--------------------------------------------------------------------------|
| LAYOUT (default) | Uses layout cell names from the Milkyway XTR or Calibre layout databases |
| SCHEMATIC        | Uses schematic cell names matched during LVS                             |

### Description

The CELL\_TYPE command specifies whether layout or schematic cell names are used for blocks and skip cells during data selection.

This command is ignored if XREF: NO is specified.

Note:

CELL\_TYPE identifies only the source of cell names for block and skip cell selection. It does not affect the cell names reported by StarRC.

### Example

CELL\_TYPE: LAYOUT

### See Also

- [BLOCK](#)
- [MILKYWAY\\_EXTRACT\\_VIEW](#)
- [NET\\_TYPE](#)
- [SKIP\\_CELLS](#)
- [XREF](#)

---

## CHECK\_SKIP\_PCELL\_LAYER\_NAMES

Enables checking the validity of PCELL layer names in a Calibre Connectivity Interface flow.

### Syntax

CHECK\_SKIP\_PCELL\_LAYER\_NAMES: YES | NO

### Arguments

| Argument     | Description                   |
|--------------|-------------------------------|
| YES          | Checks layer name validity    |
| NO (default) | Disables the layer name check |

### Description

The `CHECK_SKIP_PCELL_LAYER_NAMES` command enables existence checking for blocking layers listed in the file specified by the `SKIP_PCELL_LAYERS_FILE` command. If the `CHECK_SKIP_PCELL_LAYER_NAMES` command is set to YES, the StarRC tool checks to see if the blocking layer names are defined in the layout-versus-schematic (LVS) tool database.

### Errors

The StarRC tool issues a warning message if all of the following conditions are true:

- The `CHECK_SKIP_PCELL_LAYER_NAMES` command is set to YES.
- A layer name appears in the `BLOCKING_LAYERS` section of the file specified by the `SKIP_PCELL_LAYERS_FILE` command.
- The layer name does not appear in the LVS database file.

### See Also

- [SKIP\\_PCELLS](#)
- [SKIP\\_PCELL\\_LAYERS\\_FILE](#)

---

## CLOCK\_NET\_FREQUENCY

Specifies the analysis frequency for clock net inductance extraction.

### Syntax

`CLOCK_NET_FREQUENCY: freq`

### Arguments

| Argument    | Description                                                               |
|-------------|---------------------------------------------------------------------------|
| <i>freq</i> | Clock frequency<br>Units: GHz<br>Default: 1<br>Allowable range: 0.1 to 20 |

### Description

The `CLOCK_NET_FREQUENCY` command specifies the frequency used for inductance calculations. This command has an effect only if the `CLOCK_NET_INDUCTANCE` command is set to YES.

### See Also

- [CLOCK\\_NET\\_INDUCTANCE](#)
- [CLOCK\\_NET\\_INDUCTANCE\\_LAYERS](#)
- [Clock Net Inductance Extraction](#)

---

## CLOCK\_NET\_INDUCTANCE

Enables clock net inductance extraction.

### Syntax

CLOCK\_NET\_INDUCTANCE: YES | NO

### Arguments

| Argument     | Description                            |
|--------------|----------------------------------------|
| YES          | Enables clock net inductance analysis  |
| NO (default) | Disables clock net inductance analysis |

### Description

Set the `CLOCK_NET_INDUCTANCE` command to `YES` to enable clock net inductance extraction. You must also set the `EXTRACTION` command to `RC`.

In typical usage, you use the `NETS` command to specify the clock nets of interest. You can optionally specify the frequency of analysis by using the `CLOCK_NET_FREQUENCY` command. In addition, you can specify which database layers to consider by using the `CLOCK_NET_INDUCTANCE_LAYERS` command. A clock net and its return path must reside on the same database layer, but different clock nets can reside on different layers.

Clock net inductance extraction assumes that clock nets are fully shielded by power or ground nets and that the current return path is exclusively through the shielding nets.

Limitations are as follows:

- Clock net inductance extraction is valid only for gate-level Milkyway or LEF/DEF flows.
- Simultaneous multicorner extraction cannot be used.
- The `POWER_EXTRACT` command must be set to `NO` (the default).
- Only DSPF netlists are supported for output.

### See Also

- [CLOCK\\_NET\\_INDUCTANCE\\_LAYERS](#)
- [CLOCK\\_NET\\_FREQUENCY](#)
- [Clock Net Inductance Extraction](#)

---

## CLOCK\_NET\_INDUCTANCE\_LAYERS

Specifies the design database layers to consider for the inductance return paths in clock net inductance extraction.

### Syntax

`CLOCK_NET_INDUCTANCE_LAYERS: layer_list`

### Arguments

| Argument                | Description                                                                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>layer_list</code> | Space-delimited list of design database layers to consider for the return paths. Not case-sensitive; no wildcards allowed.<br>Default: all layers |

### Description

If the `CLOCK_NET_INDUCTANCE_LAYERS` command is used, only the specified design database layers are used for clock net inductance extraction. A clock net and its return path must reside on the same database layer, but different clock nets can reside on different layers.

### See Also

- [CLOCK\\_NET\\_INDUCTANCE](#)
- [CLOCK\\_NET\\_FREQUENCY](#)
- [Clock Net Inductance Extraction](#)

---

## COMPARE\_DIRECTORY

Specifies the location of the Hercules LVS COMPARE output.

### Syntax

COMPARE\_DIRECTORY: *path*

### Arguments

| Argument    | Description                                                                        |
|-------------|------------------------------------------------------------------------------------|
| <i>path</i> | The path to the location of the Hercules LVS COMPARE output files<br>Default: none |

### Description

The COMPARE\_DIRECTORY command specifies the location of the Hercules LVS COMPARE output. This path is specified in the Hercules sunset HEADER section, with the COMPARE\_DIRECTORY option. The Hercules default for this option is ./run\_details/compare.

This command is optional; however, if this path is not specified and XREF: YES or XREF: COMPLETE is specified, the tool attempts to read the compare directory location from the XTR (Milkyway Extract) view.

### See Also

- [XREF](#)

---

## CONLY\_NETS

Reports cross-capacitance to noncritical net neighbors.

### Syntax

CONLY\_NETS: *list\_of\_nets*

### Arguments

| Argument            | Description                                             |
|---------------------|---------------------------------------------------------|
| <i>list_of_nets</i> | List of nets; wildcards can be used.<br>(default: none) |

### Description

The CONLY\_NETS command reports cross-capacitance to noncritical net neighbors. If the COUPLE\_TO\_GROUND: YES command is used, CONLY\_NETS has no effect. If the COUPLE\_TO\_GROUND: NO command is used, the NETS command controls the behavior.

### Example

In the following example, CONLY\_NETS has no effect and all nets are netlisted:

```
COUPLE_TO_GROUND: NO
NETS: *
```

In the following example, net\_a is extracted and netlisted with coupling to net\_b:

```
COUPLE_TO_GROUND: NO
NETS: net_a
CONLY_NETS: net_b
```

The previous example results in the following output:

```
* | NET net_a
...
*CAP
1 net_a:23 net_b 1.32
2 net_a:3433 net_b 12.46
```

### See Also

- [COUPLE\\_TO\\_GROUND](#)
- [NETLIST\\_COUPLE\\_UNSELECTED\\_NETS](#)
- [NETS](#)

---

## CONVERT\_DIODE\_TO\_PARASITIC\_CAP

Specifies the extraction of parasitic properties of antenna diode structures.

### Syntax

CONVERT\_DIODE\_TO\_PARASITIC\_CAP: *model\_name* *area\_coeff* *perimeter\_coeff*

### Arguments

| Argument               | Description                                                              |
|------------------------|--------------------------------------------------------------------------|
| <i>model_name</i>      | Antenna diode model name<br>Default: none                                |
| <i>area_coeff</i>      | Area capacitance coefficient<br>Units: F/m <sup>2</sup><br>Default: none |
| <i>perimeter_coeff</i> | Perimeter capacitance coefficient<br>Units: F/m<br>Default: none         |

---

### Description

Use the CONVERT\_DIODE\_TO\_PARASITIC\_CAP command to extract parasitic properties of antenna diode structures.

### Errors

Error messages are issued when

- The model name is not found
- The capacitance coefficients are not realistic values such as negative numbers
- Device properties are not found in the input data

### Example

CONVERT\_DIODE\_TO\_PARASITIC\_CAP: NpParaDiode 1e-15 1e-16

### See Also

- [EXTRACTION](#)

---

## CONVERT\_WARNING\_TO\_ERROR

Specifies warning messages for which StarRC should halt execution instead of continuing.

### Syntax

CONVERT\_WARNING\_TO\_ERROR: *ID\_1* *ID\_2* ...

### Arguments

| Argument                        | Description                                                         |
|---------------------------------|---------------------------------------------------------------------|
| <i>ID_1</i> , <i>ID_2</i> , ... | Affected message ID numbers, separated by spaces<br>(default: none) |

### Description

The CONVERT\_WARNING\_TO\_ERROR command allows you to choose to halt extraction when StarRC encounters specified warning conditions.

### Example

The following command causes StarRC to stop if either a SX-2549 or EX-269 warning occurs:

CONVERT\_WARNING\_TO\_ERROR: SX-2549 EX-269

---

## CORNERS\_FILE

Specifies the file containing the definitions of corners available for extraction in the simultaneous multicorner flow.

### Syntax

`CORNERS_FILE: corner_file_name`

### Arguments

| Argument                      | Description                                    |
|-------------------------------|------------------------------------------------|
| <code>corner_file_name</code> | Name of the file containing corner definitions |

### Description

The `CORNERS_FILE` command specifies the file containing corner definitions of all corners that are available to be extracted in the simultaneous multicorner (SMC) flow. To specify which of the defined corners to extract, use the `SELECTED_CORNERS` command in the StarRC command file. These commands have an effect only if the SMC flow is enabled.

The `CORNERS_FILE` and `STAR_DIRECTORY` command arguments must follow these naming conventions:

- If the `STAR_DIRECTORY` command argument is a relative path, you can use either a relative path or an absolute path in the `CORNERS_FILE` command. For example:

`STAR_DIRECTORY: star_work`  
`CORNERS_FILE: smc_config`

- If the `STAR_DIRECTORY` command argument is an absolute path, you must use an absolute path in the `CORNERS_FILE` command. For example:

`STAR_DIRECTORY: /tmp/star`  
`CORNERS_FILE: /remote/.../work_directory/smc_config`

Use the following commands in the corners file to define a corner:

```
CORNER_NAME: name_of_Corner
TCAD_GRD_FILE: nxtgrd_path_and_file_name
OPERATING_TEMPERATURE: temperature_in_Celsius
(optional) CORNER_TYPE: NOMINAL | OTHER
(optional) MAPPING_FILE: map_file_name
(optional) VIA_COVERAGE_OPTION_FILE: via_file
```

You can optionally use the `MAPPING_FILE` and `VIA_COVERAGE_OPTION_FILE` commands to specify conditions for each corner.

## Examples

### Standard netlist output

The corners file CF0.txt contains the following commands:

```
CORNER_NAME: NOM_T1
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: -25
```

```
CORNER_NAME: NOM_T2
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: 125
```

```
CORNER_NAME: RCMAX_T3
TCAD_GRD_FILE: rcmax.nxtgrd
OPERATING_TEMPERATURE: 25
```

The command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: CF0.txt
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
```

This extraction generates three output netlists whose names are star\_NOM\_T2.spf, star\_NOM\_T2.spf, and star\_RCMAX\_T3.spf. Each netlist is a standard netlist (subject to other netlist options such as including tail comments or selecting the format).

### See Also

- [SELECTED\\_CORNERS](#)
- [CORNER\\_TYPE](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)
- [OPERATING\\_TEMPERATURE](#)
- [MAPPING\\_FILE](#)
- [VIA\\_COVERAGE\\_OPTION\\_FILE](#)
- [Simultaneous Multicorner Extraction](#)

---

## CORNER\_TYPE

This command is valid only within a corners file.

For precolor DPT simultaneous multicorner flows, the CORNER\_TYPE command specifies the corner to use for calculating capacitance between same-color conductors. For other simultaneous multicorner flows, the command specifies whether a corner provides temperature coefficients for the output netlist.

### Syntax

CORNER\_TYPE: NOMINAL | OTHER

### Arguments

| Argument        | Description                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------|
| NOMINAL         | Precolor DPT SMC flow: Specifies the corner to use for capacitance calculation between same-color conductors. |
| OTHER (default) | Precolor DPT SMC flow: Specifies that a corner is not a color-related corner.                                 |

### Description

In a precolor DPT simultaneous multicorner flow, the CORNER\_TYPE command is used in the corners file to specify the corner used for same-color conductors. When you use the CORNER\_TYPE: NOMINAL command, the StarRC tool uses the associated TCAD\_GRD\_FILE for capacitance extraction between polygons of the same color. Use only one NOMINAL corner in a corners file.

### See Also

- [DPT](#)
- [DPT\\_COLOR\\_GDS\\_FILE](#)
- [DPT\\_COLOR\\_GDS\\_LAYER\\_MAP\\_FILE](#)
- [DPT\\_MAX\\_SHIFT](#)
- [ER\\_VS\\_SI\\_SPACING](#)
- [Double or Multiple Patterning Technology](#)

---

## COUPLE\_NONCRITICAL\_NETS

Reports the actual net names when coupling to material outside of the primary extraction cell.

### Syntax

COUPLE\_NONCRITICAL\_NETS: *cell\_list*

### Arguments

| Argument         | Description                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>cell_list</i> | A cell or a list of cells for reporting coupling capacitance outside the primary extraction cell<br>Default: !* |

### Description

Reports the actual net names when coupling to material outside of the primary extraction cell. This command affects the child cells of BLOCK and the parent, sibling, and child cells of macros.

If you add a child cell to this list (that is, down from the primary cell), primary cell nets can couple to the real hierarchical net names in the child. The child cells are not included in the netlist and are floating nodes in the output SPEF. If you add a parent or a sibling cell to this list, a special naming scheme is used to identify the coupling nodes. Instead of using full hierarchical names from the macro perspective, the noncritical coupling nodes are named *cell\_name/local\_net\_name*.

This command overrides the ZONE\_COUPLE\_TO\_NET and SKIP\_CELLS\_COUPLE\_TO\_NET commands for the selected cells.

The \*, ? and ! wildcards are acceptable. This command can be specified multiple times.

Note:

You should explicitly specify the cells in this list instead of using the asterisk (\*) wildcard. Using a wildcard slows down runtime unnecessarily.

### Example

This example extracts an instance XU0 of cell MacroA in the context of the BLOCK TopBlock. MacroB is a sibling of MacroA, and SubMacroC is a child of MacroA. The following example shows how to configure the related options to achieve the following result:

- Parent TopBlock and siblings MacroA and MacroB are coupled with special block and net names.

- Other siblings of MacroA, such as standard cells, are coupled with the net name ZONE.
- Child SubMacroC is coupled with real hierarchical net names, from the perspective of MacroA.
- Other subcells of MacroA are coupled with the net name LUMP.

```
BLOCK: TopBlock
MACRO: XU0
COUPLE_NONCRITICAL_NETS: TopBlock MacroA MacroB SubMacroC
SKIP_CELLS_COUPLE_TO_NET: LUMP
ZONE_COUPLE_TO_NET: ZONE
```

The resulting SPEF might have capacitors like this:

```
*CAP
...
11 net1 TopBlock/clk 1.2e-13
12 net2 MacroA/net1 1e-16 // can couple to
 identical // sibling
13 net3 MacroB/net2 1.3e-18
14 net1 instA/net1 8.2e-17 // couple to SubMacroC
 cell
...
*END
```

## See Also

- [BLOCK](#)
- [COUPLE\\_NONCRITICAL\\_NETS\\_PREFIX](#)
- [COUPLE\\_TO\\_GROUND](#)
- [NETLIST\\_FORMAT](#)
- [SKIP\\_CELLS\\_COUPLE\\_TO\\_NET](#)
- [ZONE\\_COUPLE\\_TO\\_NET](#)

---

## COUPLE\_NONCRITICAL\_NETS\_PREFIX

Specifies a prefix for the nets output by the COUPLE\_NONCRITICAL\_NETS command.

### Syntax

COUPLE\_NONCRITICAL\_NETS\_PREFIX: *prefix*

### Arguments

| Argument      | Description                                   |
|---------------|-----------------------------------------------|
| <i>prefix</i> | Prefix string<br>Default: SYNOPSYS_INCONTEXT_ |

### Description

Changes the prefix used by the COUPLE\_NONCRITICAL\_NETS command flow for nets, which must be made unique to preserve independent names.

From the specified `BLOCK` down the hierarchy, this command applies the prefix to interconnect or port nets of selected COUPLE\_NONCRITICAL\_NETS and SKIP\_CELLS. From a specified `macro` up the hierarchy, the prefix is applied to all names in the external environment. For example, *instance/prefix\_netname* is applied for all noncritical nets.

If you do not specify any value, the default is SYNOPSYS\_INCONTEXT\_. If you specify NONE (not case-sensitive), an empty prefix is used such that the coupling netname is *instance/netname*.

This command is ignored if you do not specify the COUPLE\_NONCRITICAL\_NETS command.

### See Also

- [COUPLE\\_NONCRITICAL\\_NETS](#)

---

## COUPLE\_NONCRITICAL\_NETS\_SUBNODE\_SUFFIX

Specifies a netlist delimiter between the netname and suffix.

### Syntax

COUPLE\_NONCRITICAL\_NETS\_SUBNODE\_SUFFIX: *netlistDelimiter*

### Arguments

| Argument                | Description                                                                                 |
|-------------------------|---------------------------------------------------------------------------------------------|
| <i>netlistDelimiter</i> | Netlist delimiter to be inserted between the netname and suffix<br>Default: an empty string |

### Description

The COUPLE\_NONCRITICAL\_NETS\_SUBNODE\_SUFFIX command specifies a netlist delimiter between the netname and suffix. For example,

*instance/prefix\_netname\_netlistDelimiter\_suffix*

This command only works for cell-level extraction.

Retaining coupling capacitances between the top-level parent routing and SKIP\_CELLS child net routing exists for the Milkyway flow using the SPEF netlist format.

### Example

MY\_SUB\_GROUP\_1/SYNOPSYS\_INCONTEXT\_n192:1

### See Also

- [COUPLE\\_NONCRITICAL\\_NETS](#)
- [NETLIST\\_FORMAT](#)
- [RING\\_AROUND\\_THE\\_BLOCK](#)
- [SKIP\\_CELLS\\_COUPLE\\_TO\\_NET](#)
- [ZONE\\_COUPLE\\_TO\\_NET](#)

---

## COUPLE\_TO\_GROUND

Specifies whether coupling capacitances are lumped to ground during extraction and netlist generation.

### Syntax

```
COUPLE_TO_GROUND: YES | NO | YES RETAIN_GATE_CONTACT_COUPLING
 | YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING
```

### Arguments

| Argument                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES (default)                                  | Grounds coupling capacitors                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| NO                                             | Retains coupling capacitors                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| YES RETAIN_GATE_CONTACT_COUPLING               | Retains the gate-to-via or the gate-to-trench contact coupling capacitance in the netlist. Gate, trench contact, and diffusion layers are identified by the LAYER_TYPE declaration in the ITF file.<br><br>When using this option, you must set EXTRACT_VIA_CAPS: YES to retain the gate-to-contact capacitance, even if the coupling capacitances are below the capacitance thresholds specified in the COUPLING_ABS_THRESHOLD or COUPLING_REL_THRESHOLD commands.<br><br>Used only in the Hercules, IC Validator, and Calibre flows.                                                                                                                                                                       |
| YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING | Retains both the gate-to-via or the gate-to-trench contact coupling capacitance and the gate-to-diffusion coupling capacitance in the netlist. Gate, trench contact, and diffusion layers are identified by the LAYER_TYPE declaration in the ITF file.<br><br>When using this option, you must set EXTRACT_VIA_CAPS: YES to retain the gate-to-contact capacitance, even if the coupling capacitances are below the capacitance threshold value specified in the COUPLING_ABS_THRESHOLD or COUPLING_REL_THRESHOLD commands.<br><br>You must also set the IGNORE_CAPACITANCE command to NONE, DIFF or ALL RETAIN_GATE_DIFFUSION_COUPLING.<br><br>Used only in the Hercules, IC Validator, and Calibre flows. |

### Description

The COUPLE\_TO\_GROUND command specifies whether parasitic coupling capacitances are lumped to ground during extraction and netlist generation. The related

**COUPLING\_MULTIPLIER** command defines a scaling factor required to account for crosstalk effects during decoupling.

## Examples

COUPLE\_TO\_GROUND: YES

## See Also

- [COUPLING\\_MULTIPLIER](#)
- [COUPLING\\_ABS\\_THRESHOLD](#)
- [COUPLING\\_REL\\_THRESHOLD](#)
- [EXTRACT\\_VIA\\_CAPS](#)
- [IGNORE\\_CAPACITANCE](#)

---

## COUPLE\_TO\_PCELL\_PINS

Specifies the coupling of parameterized cell (PCELL) pins to overhead nets.

### Syntax

```
COUPLE_TO_PCELL_PINS: NO | YES | YES KEEP(CG) | YES IGNORE(CG)
 | YES AUTOMATIC(CG HANDLING)
```

### Arguments

| Argument                      | Description                                                                                                                                                                                                     |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO (default)                  | Ignores the coupling of PCELL pins to all material outside the PCELL                                                                                                                                            |
| YES                           | Extracts the coupling of PCELL pins to adjacent PCELL pins and to overhead metal routing. Keeps coupling between the PCELL pins and ground planes on SUBSTRATE layers.                                          |
| YES KEEP(CG)<br>(same as YES) | Extracts the coupling of PCELL pins to adjacent PCELL pins and to overhead metal routing. Keeps coupling between the PCELL pins and ground planes on SUBSTRATE layers.                                          |
| YES IGNORE(CG)                | Extracts the coupling of PCELL pins to adjacent PCELL pins and to overhead metal routing. Ignores couplings between PCELL pins and ground planes on SUBSTRATE layers.                                           |
| YES AUTOMATIC_(CG HANDLING)   | Extracts coupling of PCELL pins to adjacent PCELL pins and to overhead routing. Extracts coupling between PCELL pins and ground planes on SUBSTRATE layers only if SUBSTRATE polygons are not inside the PCELL. |

### Description

The COUPLE\_TO\_PCELL\_PINS command controls whether StarRC extracts PCELL pin coupling to adjacent PCELL pins and ground.

### Example

In the following example, StarRC extracts coupling between adjacent PCELL pins and couplings between PCELL pins and ground:

```
COUPLE_TO_PCELL_PINS: YES KEEP(CG)
```

### See Also

- [SKIP\\_PCELLS](#)

---

## COUPLING\_ABS\_THRESHOLD

Specifies an absolute threshold for grounding coupling capacitors.

### Syntax

COUPLING\_ABS\_THRESHOLD: *threshold*

### Arguments

| Argument         | Description                                                                                 |
|------------------|---------------------------------------------------------------------------------------------|
| <i>threshold</i> | Absolute threshold for grounding coupling capacitors<br>Units: farads (F)<br>Default: 3e-15 |

### Description

Specifies an absolute threshold for grounding coupling capacitors. Any coupling capacitor less than this value that also meets the relative threshold specified by the COUPLING\_REL\_THRESHOLD command is grounded.

### See Also

- [COUPLING\\_REL\\_THRESHOLD](#)
- [COUPLING\\_THRESHOLD\\_OPERATION](#)

---

## COUPLING\_MULTIPLIER

Specifies a design- and process-dependent factor to be applied for transferring coupling capacitances to ground.

### Syntax

COUPLING\_MULTIPLIER: *value*

### Arguments

| Argument     | Description                                               |
|--------------|-----------------------------------------------------------|
| <i>value</i> | A floating-point number greater than zero<br>Default: 1.0 |

### Description

Applies a design- and process-dependent factor when transferring coupling capacitances to ground. This command is used primarily to scale parasitic capacitances for crosstalk effects.

### Example

COUPLING\_MULTIPLIER: 6

### See Also

- [COUPLE\\_TO\\_GROUND](#)

---

## COUPLING\_REL\_THRESHOLD

Specifies the ratio of coupling to total capacitance.

### Syntax

COUPLING\_REL\_THRESHOLD: *threshold*

### Arguments

| Argument         | Description                                            |
|------------------|--------------------------------------------------------|
| <i>threshold</i> | Floating-point number between 0 and 1<br>Default: 0.03 |

### Description

Specifies the ratio of coupling to total capacitance, which defines nets to be grounded. Two nets are decoupled when the ratio of coupling capacitance to each total net capacitance is less than this value, if the coupling capacitance meets absolute threshold specified by the COUPLING\_ABS\_THRESHOLD command.

### See Also

- [COUPLING\\_ABS\\_THRESHOLD](#)
- [COUPLING\\_THRESHOLD\\_OPERATION](#)

---

## COUPLING\_REPORT\_FILE

Generates a report listing the coupling capacitance by net.

### Syntax

COUPLING\_REPORT\_FILE: *file*

### Arguments

| Argument    | Description                               |
|-------------|-------------------------------------------|
| <i>file</i> | File name for the report<br>Default: none |

### Description

Generates a report listing the coupling capacitance by net after smart decoupling. The report is sorted by the percentage of coupling capacitance to total capacitance for the net. The report uses the following format:

*Cc/Ct \*100 Cc victim\_net aggressor\_net*

The report contains the number of entries indicated by the COUPLING\_REPORT\_NUMBER command.

The total net capacitance used for the coupling percentage calculation is the net capacitance that is used for smart decoupling. It does not include loading pin capacitors and intranet coupling capacitors (same net coupling).

### Example

```
* 1000 worst couplings in descending order
* ratio(%) coupling victim aggressor
30.00 3e-15 net1 net2
20.00 2e-15 net3 net2
10.00 3e-15 net2 net1
```

### See Also

- [COUPLING\\_REPORT\\_NUMBER](#)

---

## COUPLING\_REPORT\_NUMBER

Specifies the number of nets reported by COUPLING\_REPORT\_FILE.

### Syntax

COUPLING\_REPORT\_NUMBER: *no\_of\_nets*

### Arguments

| Argument          | Description                                                                     |
|-------------------|---------------------------------------------------------------------------------|
| <i>no_of_nets</i> | Integer number of nets for which to report coupling capacitors<br>Default: 1000 |

### Description

Controls the size of the coupling capacitance report file by limiting the number of nets in the report.

### Example

COUPLING\_REPORT\_NUMBER: 5

### See Also

- [COUPLING\\_REPORT\\_FILE](#)

---

## COUPLING\_THRESHOLD\_OPERATION

Specifies the use of AND filtering or OR filtering of coupling thresholds.

### Syntax

COUPLING\_THRESHOLD\_OPERATION: AND | OR

### Arguments

| Argument      | Description   |
|---------------|---------------|
| AND (default) | AND filtering |
| OR            | OR filtering  |

### Description

The following conditions are checked to determine whether a coupling capacitance,  $Cc(\text{net1}-\text{net2})$  should be decoupled:

Condition1:  $Cc(\text{net1}-\text{net2}) < \text{COUPLING\_ABS\_THRESHOLD}$

Condition2:  $Cc(\text{net1}-\text{net2}) < \text{COUPLING\_REL\_THRESHOLD} * \text{TCAP}_{\text{net1}}$

Condition3:  $Cc(\text{net1}-\text{net2}) < \text{COUPLING\_REL\_THRESHOLD} * \text{TCAP}_{\text{net2}}$

The COUPLING\_THRESHOLD\_OPERATION command specifies the use of AND filtering or OR filtering of coupling capacitances.

- When AND filtering is specified, a coupling capacitance is decoupled if the following operation is true:  
*Condition1 AND (Condition2 AND Condition3)*
- When OR filtering is specified, a coupling capacitance is decoupled if the following operation is true:  
*Condition1 OR (Condition2 AND Condition3)*

### See Also

- [COUPLING\\_ABS\\_THRESHOLD](#)
- [COUPLING\\_REL\\_THRESHOLD](#)

---

## DEBUG\_MILKYWAY\_DATABASE

Specifies the name of a Milkyway database to use for debugging purposes.

### Syntax

DEBUG\_MILKYWAY\_DATABASE: *mw\_lib*

### Arguments

| Argument      | Description                                 |
|---------------|---------------------------------------------|
| <i>mw_lib</i> | The Milkyway database name<br>Default: none |

### Description

The DEBUG\_MILKYWAY\_DATABASE command provides the name of a Milkyway database into which the StarRC tool can write data for debugging purposes. If a database with this name already exists, it is overwritten.

The DEBUG\_MILKYWAY\_DATABASE command is used only in a short StarRC command file that converts xin data to Milkyway format for the purpose of examining the data in a viewer.

You can save two types of data for debugging:

- Signal net opens and shorts
- Metal fill shorts

This feature is available for Milkyway and LEF/DEF flows that do not use the field solver. Nets are displayed in mask layout dimensions after the application of any half-node scale factors.

### Signal Net Opens and Shorts

To investigate signal net opens and shorts that the StarRC tool finds during an extraction run, follow this procedure:

1. Create a simple StarRC command file for the purpose of visualizing opens and shorts.  
The following example is a command file named star\_cmd\_debug. Use the NETS command to select specific nets to view; selecting all nets is not recommended.

```
DEBUG_MILKYWAY_DATABASE: my_library
STAR_DIRECTORY: star
NETS: net1 net2 net3
```

2. Invoke the StarRC tool with the following command:

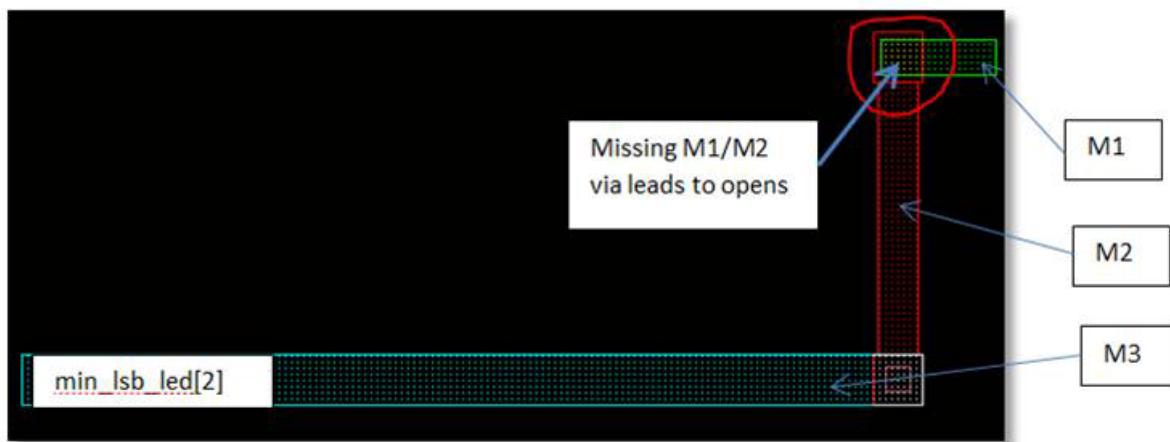
```
% StarXtract -Display star_cmd_debug
```

The `-Display` option directs the StarRC tool to use information in the star directory to identify the nets of interest and write them into a Milkyway format database. The tool does not perform any extraction or netlisting operations.

3. Open a Milkyway viewer to examine the nets.

[Figure 15-2](#) is an example of a net identified as an open by the StarRC tool. Examination reveals that a via is missing between the M1 and M2 layers.

*Figure 15-2 Example View of Open in Milkyway Database*



### Metal Fill Shorts

To investigate metal fill shorts, follow this procedure:

1. Create a simple StarRC command file for the purpose of visualizing metal fill shorts. The following example is a command file named `star_mf`. Use the `NETS` command to select specific nets to view; selecting all nets is not recommended.

```
DEBUG_MILKYWAY_DATABASE: mf_debug
STAR_DIRECTORY: star
NETS: net1 net2 net3
```

2. Invoke the StarRC tool with the following command:

```
% StarXtract -Display_mf star_mf
```

The `-Display_mf` option directs the StarRC tool to use information in the star directory to identify the metal fill polygons of interest and write them into a Milkyway format database. The tool does not perform any extraction or netlisting operations.

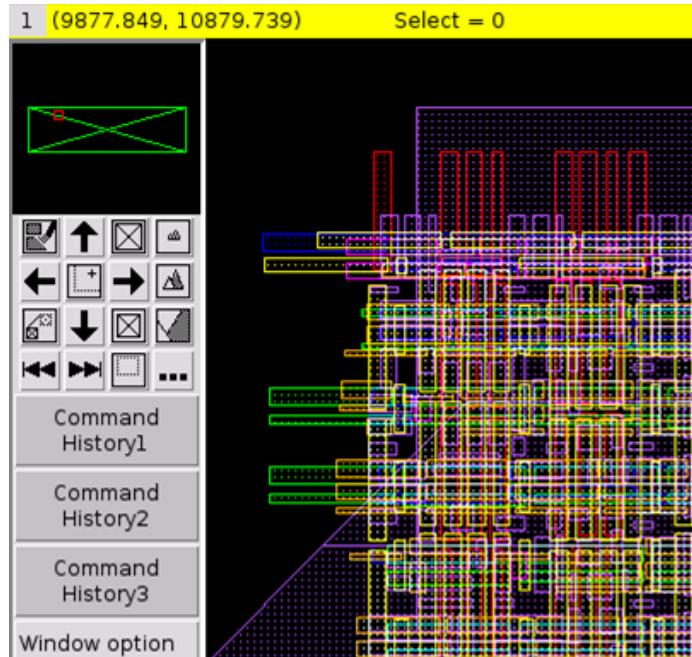
You can optionally include a set of coordinates (in microns) to restrict the region from which to export the metal fill polygons. For example,

```
% StarXtract -Display_mf star_mf 9880 10880 9890 10890
```

3. Open a Milkyway viewer to examine the nets.

[Figure 15-3](#) is an example of the resulting Milkyway display.

*Figure 15-3 Example View of Metal Fill Polygons*



## See Also

- [STAR\\_DIRECTORY](#)
- [NETS](#)

---

## DEF\_ATTRIBUTE\_FROM\_LEF

Assigns a property from a specific LEF file to a specific DEF macro.

### Syntax

```
DEF_ATTRIBUTE_FROM_LEF: WIDTH def_macro_name lef_file_name
```

### Arguments

| Argument              | Description                                                       |
|-----------------------|-------------------------------------------------------------------|
| WIDTH                 | The routing width of a layer in the LEF file                      |
| <i>def_macro_name</i> | The DEF macro name                                                |
| <i>lef_file_name</i>  | The LEF file name from which the property information is obtained |

### Description

By default, the StarRC tool uses LEF properties from the default LEF file for DEF macros. The `DEF_ATTRIBUTE_FROM_LEF` command overwrites the LEF property from the specified LEF file only for the specified DEF macro. You must specify each DEF macro separately to use the LEF property from a LEF file.

The routing width property (WIDTH) is the only property available from the LEF file.

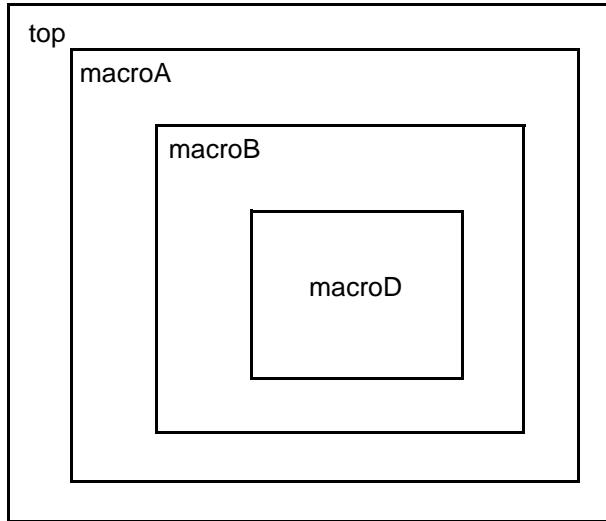
### Example

The following syntax assigns the routing width from different LEF files to specific DEF macros. [Figure 15-4](#) illustrates the relationship between the macros.

```
LEF_FILE: tech.lef
TOP_DEF_FILE: top.def
MACRO_DEF_FILE: macroA.def
MACRO_DEF_FILE: macroB.def
MACRO_DEF_FILE: macroD.def

DEF_ATTRIBUTE_FROM_LEF: WIDTH macroA tech1.lef
DEF_ATTRIBUTE_FROM_LEF: WIDTH macroD tech5.lef
```

*Figure 15-4 Macros in the top.def File of a Hierarchical Design*



The macros named top and macroB use the routing width from the default LEF file (tech.lef). MacroA and macroD use routing width values from LEF files tech1 and tech5, respectively, due to the use of the `DEF_ATTRIBUTE_FROM_LEF` command.

---

## DEF\_USE\_PINS

Specifies whether to obtain pin information from DEF or LEF files.

### Syntax

DEF\_USE\_PINS: YES | NO

### Arguments

| Argument      | Description                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------|
| YES (default) | Takes pin information first from the DEF file and then from the LEF file, if not present in the DEF file |
| NO            | Takes pin information from the LEF file                                                                  |

### Description

For a LEF/DEF flow, the `DEF_USE_PINS` command specifies whether to obtain pin information from the DEF file or the LEF file.

If the command is set to `YES` (the default), pin information is taken preferentially from the DEF file. Pins not found in the DEF file are taken from the LEF file. Conversely, if the `DEF_USE_PINS` command is set to `NO`, the pin information is taken only from the LEF file.

### See Also

- [LEF\\_FILE](#)
- [TOP\\_DEF\\_FILE](#)
- [MACRO\\_DEF\\_FILE](#)
- [The LEF/DEF Database Flow](#)

---

## DENSITY\_BASED\_THICKNESS

Enables the calculation of density and thickness variation during extraction.

### Syntax

DENSITY\_BASED\_THICKNESS: YES | NO

### Arguments

| Argument      | Description                                                                      |
|---------------|----------------------------------------------------------------------------------|
| YES (default) | Considers density-based thickness variation options as specified in the ITF file |
| NO            | Does not consider density-based thickness options                                |

### Description

This command enables the calculation of density and thickness variation using the THICKNESS\_VS\_DENSITY or the POLYNOMIAL\_BASED\_THICKNESS\_VARIATION commands in the ITF file.

No warning is issued if thickness variation commands are not specified in the ITF file.

### Example

DENSITY\_BASED\_THICKNESS: YES

### See Also

- [NETS](#)
- [USE\\_SI\\_DENSITY](#)
- [POLYNOMIAL\\_BASED\\_THICKNESS\\_VARIATION](#)
- [THICKNESS\\_VS\\_DENSITY](#)

---

## DENSITY\_OUTSIDE\_BLOCK

Specifies the pattern density outside the block, which affects the thickness variation and parasitic RC values.

### Syntax

DENSITY\_OUTSIDE\_BLOCK: *density\_value*

### Arguments

| Argument             | Description                                                              |
|----------------------|--------------------------------------------------------------------------|
| <i>density_value</i> | Pattern cell density; a floating-point number from 0.0 to 1.0, inclusive |

### Description

The DENSITY\_OUTSIDE\_BLOCK command defines the pattern density outside the block. The specified density is applied to all layers on which StarRC performs density calculation.

For calculating the density of a polygon, the tool considers a 50-micron square window. If the polygon of interest is located near the edge of the block, StarRC calculates the final density using a weighted calculation that takes both the actual inside density and the outside density specified with the DENSITY\_OUTSIDE\_BLOCK command into account.

When this option is not set, StarRC extends the density inside the block to outside the block, under the assumption that each layer has the same density distribution inside and outside the block.

This command is effective only when you specify density-based thickness variation in the ITF file and include the DENSITY\_BASED\_THICKNESS: YES command in the StarRC command file.

### Example

The following example specifies a pattern density outside the block of 40 percent:

```
DENSITY_OUTSIDE_BLOCK: 0.40
```

### See Also

- [DENSITY\\_BASED\\_THICKNESS](#)

---

## DETECT\_FUSE

Enables the detection and reporting of abutting metal polygons and other types of jog patterns.

### Syntax

DETECT\_FUSE: YES | NO

### Arguments

| Argument     | Description                             |
|--------------|-----------------------------------------|
| YES          | Enables metal polygon overlap analysis  |
| NO (default) | Disables metal polygon overlap analysis |

### Description

The DETECT\_FUSE command detects instances of abutting metal polygons or conductor jog patterns that might pose electromigration risks. A fuse is a metal pattern in which an open circuit can be intentionally created by forcing current through a region of reduced width. Fuse-like conditions might occur unintentionally when the width of a metal conductor along the direction of current flow is reduced.

The StarRC tool detects fuse conditions when the DETECT\_FUSE command is set to YES. You must also specify the following commands:

- NETLIST\_TAIL\_COMMENTS: YES (to write the fuse information in the tail comments)
- REDUCTION: NO (to prevent the merging of fuse resistors)

Each detected fuse is represented in the netlist as a shorting resistor with a resistance value of 0.01 ohms, a length of 0, and a width. The reported width for abutting polygons is the length of the abutment region; the reported width of an overlap region is the length of the diagonal of the overlap region.

In the network, the fuse resistor is added to the wider of the two polygons because that is where the most narrowing along the direction of current flow occurs. You can report the fuse node location in the netlist by setting the EXTRA\_GEOMETRY\_INFO command to NODE and setting the NETLIST\_NODE\_SECTION command to YES.

### Examples

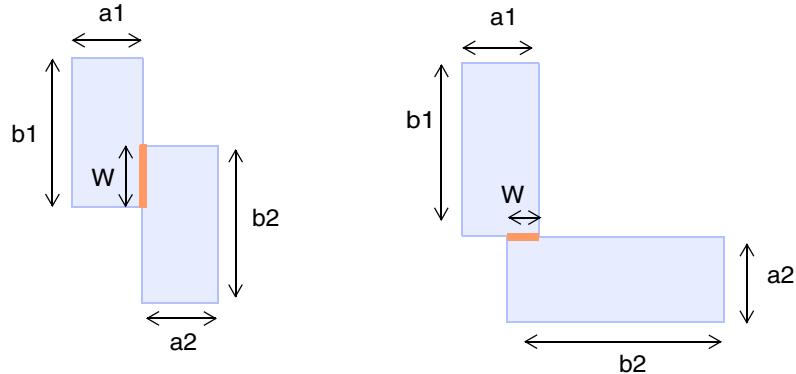
[Figure 15-6](#) and [Figure 15-5](#) show fuses created from overlapping or abutting polygons. [Figure 15-7](#) shows a tunnel pattern, in which a narrow polygon connects two wider

polygons. L-shaped conductor configurations can be reduced to these cases. The rules also apply to versions of these patterns rotated by 90 degrees.

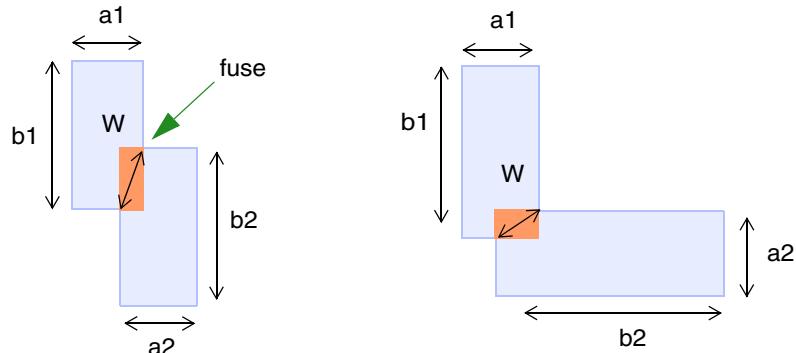
In these figures, dimensions  $b_1$  and  $b_2$  represent the longer sides of the polygons; dimensions  $a_1$  and  $a_2$  are the shorter sides of the corresponding polygons.

The StarRC tool detects a fuse if dimension  $W$  is smaller than both the  $a_1$  and  $a_2$  dimensions. The fuse resistor width in the netlist is equal to dimension  $W$ .

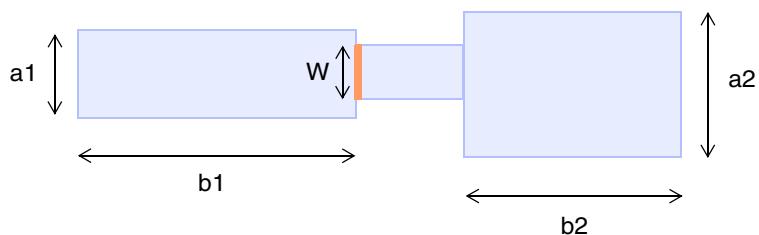
*Figure 15-5 Fuse Geometry With Abutting Polygons*



*Figure 15-6 Fuse Geometry With Jog Patterns*

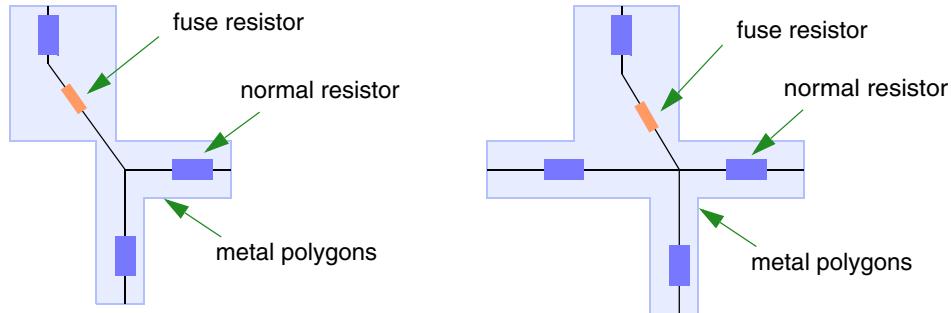


*Figure 15-7 Fuse Geometry With Tunnel Pattern*



The fuse resistor location is shown in [Figure 15-8](#). The greatest width narrowing occurs in the y-direction; therefore the fuse resistor is placed between the parasitic resistor for the widest net and the node where the metal polygons meet.

*Figure 15-8 Fuse Resistor Location*



An example netlist is as follows:

```
* | NET T1P2_12_050 0PF
* | P (T1P2_12_050 B 0 -229.71 160.863) // $llx=-230.705 $lly=160.858
$urx=-228.705 $ury=160.868 $lvl=4
* | P (T1P2_12_050_1 B 0 -229.71 170.847) // $llx=-231.205 $lly=170.847
$urx=-230.205 $ury=170.863 $lvl=4
* | S (67 -229.955 165.863) // $llx=-231.205 $lly=165.863 $urx=-228.705
$ury=165.863 $lvl=2
* | S (61 -229.955 165.863) // $llx=-230.705 $lly=165.863 $urx=-230.205
$ury=165.863 $lvl=2
* | S (F393 -230.71 170.847) // $llx=-231.205 $lly=170.847 $urx=-230.205
$ury=170.863 $lvl=4
* | S (F392 -229.71 160.863) // $llx=-230.705 $lly=160.858 $urx=-228.705
$ury=160.868 $lvl=4
RxT1P2_12_050_1 T1P2_12_050_1 F393 0.001
RxT1P2_12_050 T1P2_12_050 F392 0.001
R13 F392 67 1.647236 $l=5 $w=1 $lvl=2
R14 67 61 0.01 $l=0 $w=0.5 $lvl=2
R15 61 F393 0.825000 $l=4.984 $w=2 $lvl=2
```

In this example, R14 is a fuse resistor with width 0.5. Subnode 61 is the fuse node. Its location is \$llx=-230.705 \$lly=165.863 \$urx=-230.205 \$ury=165.863 \$lvl=2.

## See Also

- [NETLIST\\_TAIL\\_COMMENTS](#)
- [REDUCTION](#)
- [EXTRA\\_GEOMETRY\\_INFO](#)
- [NETLIST\\_NODE\\_SECTION](#)

---

## DIELECTRIC\_FILL\_GDS\_FILE

Specifies the GDSII file containing metal fill data.

### Syntax

DIELECTRIC\_FILL\_GDS\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                 |
|------------------|-------------------------------------------------------------|
| <i>file_name</i> | GDSII file containing dielectric fill data<br>Default: none |

### Description

The DIELECTRIC\_FILL\_GDS\_FILE command supports either hierarchical or flat GDSII files.

All shapes on layers mapped by the file specified by the DIELECTRIC\_FILL\_GDS\_LAYER\_MAP\_FILE command within the master definition of `BLOCK` and its child cells are treated as dielectric fill objects for extraction. All other data not referenced by the master definition of `BLOCK` is ignored.

The DIELECTRIC\_FILL\_GDS\_FILE and DIELECTRIC\_FILL\_GDS\_LAYER\_MAP\_FILE commands must both be specified.

You can specify gzip and compressed GDS files for this command.

### See Also

- [GDS\\_FILE](#)
- [GDS\\_LAYER\\_MAP\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE\\_NET\\_NAME](#)

---

## DIELECTRIC\_FILL\_GDS\_LAYER\_MAP\_FILE

Specifies the mapping between the GDSII layer number and layer name in the design database for dielectric fill features.

### Syntax

DIELECTRIC\_FILL\_GDS\_LAYER\_MAP\_FILE: *file\_name*

### Arguments

| Argument         | Description                                           |
|------------------|-------------------------------------------------------|
| <i>file_name</i> | Name of the GDSII layer mapping file<br>Default: none |

### Description

The DIELECTRIC\_FILL\_GDS\_LAYER\_MAP\_FILE command specifies the mapping between the GDSII layer number and layer name in the design database whenever the DIELECTRIC\_FILL\_GDS\_FILE command is used to import GDSII data into the design database.

All translated GDSII layers must have an entry in the file specified by the DIELECTRIC\_FILL\_GDS\_LAYER\_MAP\_FILE command and must have a definition in the layout database.

If you use more than one of the DIELECTRIC\_FILL\_GDS\_FILE, METAL\_FILL\_GDS\_FILE, and GDS\_FILE commands in a single run, use a unified layer mapping file for the GDSII files.

The conductor layer must contain the DIELECTRIC\_FILL\_VS\_SI\_SPACING command in the ITF file.

The layer map file uses the following syntax:

```
database_layer gdsii_layer_number gdsii_datatype
 [FLOATING | GROUNDED | IGNORE]
```

| Argument                  | Description                                                                                                                                                                        |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>database_layer</i>     | The database layer name                                                                                                                                                            |
| <i>gdsii_layer_number</i> | The GDSII layer number                                                                                                                                                             |
| <i>gdsii_datatype</i>     | The GDSII data type. If a GDSII data type is not specified, then all data types on a given layer are read.                                                                         |
| FLOATING                  | Specifies that the corresponding fill layer is to be treated as floating. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored. |
| GROUNDED                  | Specifies that the corresponding fill layer is to be treated as grounded. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored. |
| IGNORE                    | Specifies that the corresponding fill layer is to be ignored. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.             |

The layer-specific fill-handling keyword allows you to decide how individual metal fill layers are handled during parasitic extraction. This handling is considered only when the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command is set in the StarRC command file. If the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command appears in the StarRC command file, but a fill-handling mode is not specified for a particular layer inside the GDS layer map file, the default setting is FLOATING for that layer. If another setting of the METAL\_FILL\_POLYGON\_HANDLING command (other than AUTOMATIC) is specified, that setting governs the handling of all layers, and any layer-specific mode specifications inside the GDS layer map file are ignored.

In the example, handling mode GROUNDED is specified for layer DIFF. If the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command is also specified in the StarRC command file, DIFF is treated as GROUNDED while all other layers are treated as FLOATING. If METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC is not specified in the command file, the layer-specific mode specification for DIFF is disregarded, and the global mode set by the METAL\_FILL\_POLYGON\_HANDLING command takes precedence.

## Examples

The following example shows how the DIFF layer is assigned to GDSII layer 2 and GDSII datatype 0. This example maps layers from a metal fill GDS file and specifies layer-specific fill handling for the DIFF layer.

### *Example 15-2 Layer-Specific Fill Handling*

```
DIFF 2 0 GROUNDED
POLY 7 0
CONT 4 0
METAL1 10 0
METAL1 10 1
METAL1 76 0
VIA1 11 0
METAL2 12 0
```

In the following example, the `METAL_FILL_POLYGON_HANDLING: AUTOMATIC` command is set in the command file.

### *Example 15-3 Automatic Fill Handling*

```
*layer treated as grounded
DIFF 2 0 GROUNDED
*layer treated as floating
POLY 7 0 FLOATING
*layer governed by default floating mode since mode is unspecified.
METAL1 4 0
```

## See Also

- [GDS\\_FILE](#)
- [LEF\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE](#)

---

## DIFFUSION\_RES\_MODE

Specifies the model to use for mesh source and drain diffusions.

### Syntax

DIFFUSION\_RES\_MODE: LINE | POINT

### Arguments

| Argument       | Description                                                          |
|----------------|----------------------------------------------------------------------|
| LINE (default) | Models the gate to diffusion overlap region as an equipotential line |
| POINT          | Models the gate to diffusion overlap region with resistors           |

### Description

The StarRC tool extracts diffusion resistance as a resistor mesh structure. By default, the StarRC tool models the source and drain terminals of a transistor as an equipotential line, as shown in [Figure 15-9](#).

For advanced process nodes and transistors with large widths, the aspect ratio of the diffusion might be very large. The `POINT` mode inserts additional resistance along the source and drain contact area. [Figure 15-9](#) and [Figure 15-10](#) illustrate the two modes.

*Figure 15-9 Diffusion Resistance LINE Mode*

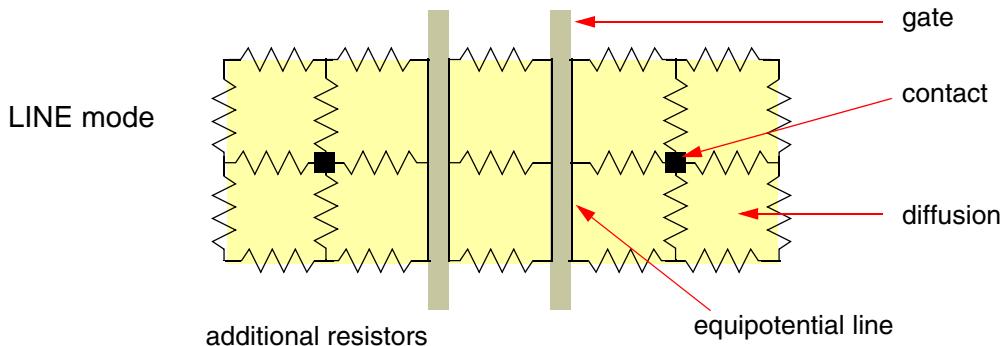
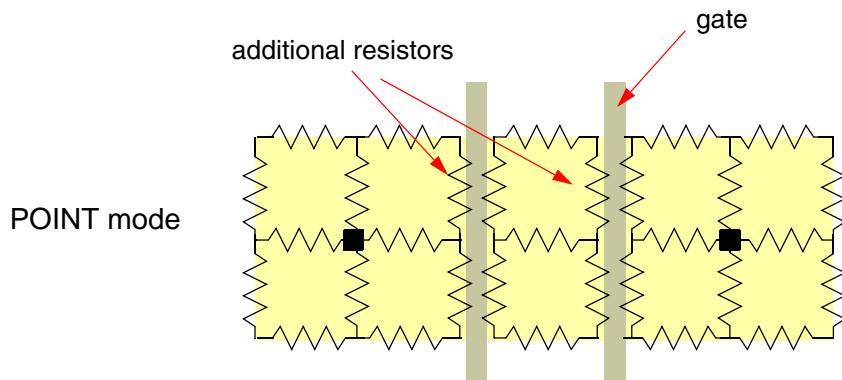


Figure 15-10 Diffusion Resistance POINT mode



### See Also

- [Diffusion Resistance](#)

---

## DPT

Enables extraction for double or multiple patterning processes.

### Syntax

DPT: YES | NO | SHIFT

### Arguments

| Argument     | Description                                                                             |
|--------------|-----------------------------------------------------------------------------------------|
| YES          | Uses dielectric constant changes                                                        |
| SHIFT        | Uses the exact misalignment data                                                        |
| NO (default) | Performs regular extraction; does not extract parasitics for multiple patterned layers. |

### Description

Specifies the misalignment effects model for double or multiple patterning technologies.

### Example

The following command specifies the use of dielectric constant changes to model double patterning effects:

DPT: YES

### See Also

- [DPT\\_COLOR\\_GDS\\_FILE](#)
- [DPT\\_COLOR\\_GDS\\_LAYER\\_MAP\\_FILE](#)
- [DPT\\_MAX\\_SHIFT](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [Double or Multiple Patterning Technology](#)

---

## DPT\_COLOR\_GDS\_FILE

Specifies the GDSII file containing the color-layer polygon definitions.

### Syntax

DPT\_COLOR\_GDS\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                   |
|------------------|---------------------------------------------------------------|
| <i>file_name</i> | The GDSII file containing the color-layer polygon definitions |

### Description

The DPT\_COLOR\_GDS\_FILE specifies the GDSII file containing the color-layer polygon definitions. This file is usually created by the place-and-route tool.

### Example

In the following example, the file named Decomposed\_m1\_m2.GDS contains the color-layer polygon definitions:

DPT\_COLOR\_GDS\_FILE: Decomposed\_m1\_m2.GDS

### See Also

- [DPT](#)
- [DPT\\_COLOR\\_GDS\\_LAYER\\_MAP\\_FILE](#)
- [DPT\\_MAX\\_SHIFT](#)
- [ER\\_VS\\_SI\\_SPACING](#)
- [SELECTED\\_CORNERS](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)
- [Double or Multiple Patterning Technology](#)

---

## DPT\_COLOR\_GDS\_LAYER\_MAP\_FILE

Maps color layers in the design database to GDSII layers.

### Syntax

DPT\_COLOR\_GDS\_LAYER\_MAP\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                                      |
|------------------|----------------------------------------------------------------------------------|
| <i>file_name</i> | Mapping file for mapping color layers in the design database to the GDSII layers |

### Description

The DPT\_COLOR\_GDS\_LAYER\_MAP\_FILE command specifies the file that maps the color layers in the design database to the GDSII layers. This file is created by the place-and-route tool and uses a pound sign (#) for comments.

An example of the file is as follows:

| #DBLayerName | #GDS_LayerID | #GDS_LayerDataType |
|--------------|--------------|--------------------|
| M1_color1    | 21           | 1                  |
| M1_color2    | 21           | 2                  |
| M2_color1    | 22           | 1                  |
| M2_color2    | 22           | 2                  |

All layer names in the specified file must exist in the StarRC mapping file under the color\_layers section.

### Example

In the following example, the DPT\_COLOR\_GDS\_LAYER\_MAP\_FILE command specifies a file named gmap as the file that contains color layer mapping information.

DPT\_COLOR\_GDS\_LAYER\_MAP\_FILE: gmap

### See Also

- [DPT](#)
- [DPT\\_COLOR\\_GDS\\_FILE](#)
- [DPT\\_MAX\\_SHIFT](#)
- [Double or Multiple Patterning Technology](#)

---

## ECO\_MODE

Sets conditions for ECO extraction.

### Syntax

`ECO_MODE: YES | NO | RESET`

### Arguments

| Argument     | Description                                                                             |
|--------------|-----------------------------------------------------------------------------------------|
| YES          | Specifies that ECO extraction should be performed if conditions are met                 |
| NO (default) | Specifies extraction of all nets with standard output (not suitable for ECO extraction) |
| RESET        | Specifies extraction of all nets with output suitable for ECO extraction                |

### Description

ECO (engineering change order) extraction is the technique of performing extraction only on parts of a design that are different from a reference design. This capability allows efficient evaluation of engineering change orders, especially when coupled with a timing analysis tool that can also operate on ECO netlists.

The `ECO_MODE` command controls ECO extraction. The options are `YES`, `RESET`, and `NO` (the default).

- The `YES` option allows the StarRC tool to perform the most appropriate extraction for the state of the ECO cycle. An ECO extraction is performed unless one of the following conditions applies:
  - If the design database does not have any logical or physical changes since the previous extraction, the StarRC tool does not perform any extraction and does not update the netlists.
  - If the star directory is missing, the tool performs full-chip extraction due to the absence of a reference run.
  - If the ECO design changes are extensive and therefore the number of nets selected for re-extraction is large, the tool performs a full-chip extraction because there is little runtime benefit from an ECO extraction. This full-chip run becomes the reference for subsequent ECO runs.

- The `RESET` option causes the StarRC tool to perform a full-chip extraction in a manner compatible with subsequent ECO extractions. This option is a method to force a full-chip extraction during a cycle of ECO extractions.
- The `NO` option disables ECO extraction, which is equivalent to not using the command. In this case, the StarRC tool performs full-chip extraction, and the resulting star directory is not compatible with ECO extraction. If you plan to use the ECO flow, you should not use this option.

The first StarRC run is always a full-chip extraction because a reference run must exist for ECO extractions. Subsequent StarRC runs might be either full-chip or ECO extractions, depending on the number of ECO-affected nets compared to the size of the design.

You cannot make changes in the command file that affect the extraction behavior between full-chip and ECO extractions or between successive ECO extractions. The commands can remain in the StarRC command file, but they cannot be modified between runs.

## See Also

- [NETLIST\\_ECO\\_FILE](#)
- [NETLIST\\_INCREMENTAL](#)
- [Chapter 5, “ECO Extraction”](#)

---

## ECO\_RESTORE\_COMMAND

Specifies the command to be used for restoring the incremental extraction directory during ECO extraction.

### Syntax

ECO\_RESTORE\_COMMAND: *os\_cmd*

### Arguments

| Argument      | Description                                             |
|---------------|---------------------------------------------------------|
| <i>os_cmd</i> | An operating system command string<br>Default: tar -xzf |

### Description

For portability of the working directory during ECO extraction, the `StarXtract` command provides the `-save_eco` and `-restore_eco` options. The `-save_eco` option compresses the directory containing the ECO extraction results into a single file and writes that file to a specified location. The `-restore_eco` option uncompresses the file and restores the directory in preparation for performing another extraction cycle.

You can specify the operating system command to use for restoring a saved ECO extraction directory by setting the `ECO_RESTORE_COMMAND` command argument.

### Examples

The following command specifies where to find the linux `tar` command and the options to use with it:

ECO\_RESTORE\_COMMAND: /depot/local/bin/tar -xzf

### See Also

- [ECO\\_MODE](#)
- [ECO\\_SAVE\\_COMMAND](#)
- [Chapter 5, “ECO Extraction”](#)

---

## ECO\_SAVE\_COMMAND

Specifies the command to be used for saving the incremental extraction directory during ECO extraction.

### Syntax

ECO\_SAVE\_COMMAND: *os\_cmd*

### Arguments

| Argument      | Description                                             |
|---------------|---------------------------------------------------------|
| <i>os_cmd</i> | An operating system command string<br>Default: tar -czf |

### Description

For portability of the working directory during ECO extraction, the `StarXtract` command provides the `-save_eco` and `-restore_eco` options. The `-save_eco` option compresses the directory containing the ECO extraction results into a single file and writes that file to a specified location. The `-restore_eco` option uncompresses the file and restores the directory in preparation for performing another extraction cycle.

You can specify the operating system command to use for saving an ECO extraction directory by setting the `ECO_SAVE_COMMAND` command argument.

### Example

The following command specifies where to find the linux `tar` command and the compression options to use with it:

ECO\_SAVE\_COMMAND: /depot/local/bin/tar -czf

### See Also

- [ECO\\_MODE](#)
- [ECO\\_RESTORE\\_COMMAND](#)
- [Chapter 5, “ECO Extraction”](#)

---

## EVACCESS\_DIRECTORY

Specifies the location of the Hercules LVS EvAccess database.

### Syntax

EVACCESS\_DIRECTORY: *path*

### Arguments

| Argument    | Description                                                 |
|-------------|-------------------------------------------------------------|
| <i>path</i> | Path to the Hercules LVS EvAccess database<br>Default: none |

### Description

Specifies the location of the Hercules LVS EvAccess database. This path can also be specified in the Hercules runset EVACCESS\_OPTIONS section, with the PATH option. The Hercules default for this option is ./evaccess.

If this path is not specified and the XREF command is set to YES or COMPLETE, the tool attempts to read the directory location from the XTR (Milkyway Extract) view.

The EVACCESS\_DIRECTORY command is not used in the IC Validator flow.

### See Also

- [XREF](#)

---

## **EXTRA\_GEOMETRY\_INFO**

Reports the internal node bounding box information for resistors.

### **Syntax**

`EXTRA_GEOMETRY_INFO: RES | NODE | RES NODE | NODE RES | NONE`

### **Arguments**

| <b>Argument</b>             | <b>Description</b>                                 |
|-----------------------------|----------------------------------------------------|
| <code>RES</code>            | Reports bounding boxes for metal and via resistors |
| <code>NODE</code>           | Reports bounding boxes for nodes                   |
| <code>RES NODE</code>       | Reports both resistor and node information         |
| <code>NODE RES</code>       | Identical to <code>RES NODE</code>                 |
| <code>NONE</code> (default) | Does not report extra geometry information         |

### **Description**

The `EXTRA_GEOMETRY_INFO` command reports the internal node bounding box information for a resistor, either as a tail comment in the node section of the netlist, a node property in the Milkyway parasitic database, or both. The bounding box dimensions are always as drawn and are not affected by the `NETLIST_UNSCALED_COORDINATES` command.

If you use the `RES` setting,

- The `NETLIST_TAIL_COMMENTS` command must be set to `YES` to write the extra resistor information to the netlist.
- The `REDUCTION` and `POWER_REDUCTION` commands must be set to `NO` to preserve the original layout topology.
- The `KEEP_VIA_NODES` command must be set to `YES` if you want to preserve via resistors.

If you use the `NODE` setting,

- All reduction modes are supported.
- The `KEEP_VIA_NODES` command must be set to `YES` if you want to preserve via resistors.

If you are performing power rail analysis, set the `TARGET_PWRA` command to `YES`. This command automatically sets StarRC commands for optimal analysis, including setting the `EXTRA_GEOMETRY_INFO` command to `RES NODE`.

If the `EXTRA_GEOMETRY_INFO` command appears more than one time in a StarRC command file, only the last setting is used.

## Example

Bounding boxes are reported by adding four coordinates to the tail comments, as follows:

- `$llx` is the lower-left x-coordinate (all coordinates are in microns)
- `$lly` is the lower-left y-coordinate
- `$urx` is the upper-right x-coordinate
- `$ury` is the upper-right y-coordinate
- `$dir` is the direction of current flow (0 for horizontal, 1 for vertical, 2 for non-Manhattan)

This example shows part of a netlist with the `EXTRA_GEOMETRY_INFO: NODE RES` setting:

```
*D_NET I20|N9 0.0869015

*CONN
*I I20|I44.X O *C 151.19 11.75 *D NAN2 $llx=150.35 $lly=11.75 \
$urx=151.19 $ury=12.73 $lvl=1
*I I20|I45.A I *C 149.16 11.75 *L 2 *D NOR2 $llx=149.16 $lly=11.75 \
$urx=149.16 $ury=12.73 $lvl=1
*N I20|N9.220 *C 155.04 17.42 // $llx=154.83 $lly=17.42 $urx=155.04 \
$ury=17.42 $lvl=1
*N I20|N9.235 *C 154.83 18.68 // $llx=154.83 $lly=18.68 $urx=155.04 \
$ury=18.68 $lvl=1
*N I20|N9.234 *C 153.57 18.68 // $llx=153.57 $lly=18.68 $urx=153.57 \
$ury=18.68 $lvl=1

1: *84.1718 *84.1705 0.646554 // $l=0.152000 $w=0.114000 $lvl=4 \
$llx=102.490 $lly=64.576 $urx=102.605 $ury=64.823 $dir=0
```

When you run an extraction using `EXTRA_GEOMETRY_INFO`, the `LAYER_MAP` section of the netlist can also contain generated layer names. Extra layers are formed in the case of device-level extraction when there are database layers at the diffusion level or below that share a contact. For instance, if the runset contains the line shown in the following example, then the `LAYER_MAP` section contains an extra layer called `nsd:psd` or `psd:nsd`, which becomes the lower terminal level of `diffCont` via resistors.

```
CONNECT metall nsd psd BY diffCont
```

## See Also

- [CAPACITOR\\_TAIL\\_COMMENTS](#)
- [NETLIST\\_TAIL\\_COMMENTS](#)
- [NETLIST\\_UNSCALED\\_COORDINATES](#)

---

## EXTRACTION

Specifies the type of extraction and the scope of the generated netlist.

### Syntax

EXTRACTION: RC | C | R | FSCOMPARE

### Arguments

| Argument     | Description                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RC (default) | Extracts both parasitic resistor and capacitor devices and merges them into the original database network to produce a consolidated RC network description of the layout in the specified format.                                                                                                                                                                                                                     |
| C            | Extracts only parasitic capacitor devices and produces a merged parasitic layout network description as a SPICE file. The NETLIST_FORMAT command is ignored for capacitance-only extractions.                                                                                                                                                                                                                         |
| R            | Extracts only parasitic resistor devices and produces a merged parasitic layout network description in the specified format.                                                                                                                                                                                                                                                                                          |
| FSCOMPARE    | Provides a comparison report of a merged layout network description containing only parasitic capacitors, executes a field solver analysis of the layout, and produces report files that describe the accuracy in a comparison of the two results.<br>When this option is specified, the .fscomptot and .fs_compcoup output comparison files always use the layout net names, regardless of the XREF command setting. |

### Description

The extraction of parasitic devices is performed only on that portion of the layout network defined by the NETS command, terminating each net at the boundary of a skip cell.

### See Also

- [FSCOMPARE\\_OPTIONS](#)
- [FS\\_EXTRACT\\_NETS](#)
- [NETLIST\\_FORMAT](#)
- [REDUCTION](#)

---

## EXTRACT\_RES\_BODY\_COUPLING

Specifies the extraction of coupling capacitances between resistor body elements and interconnect features or ground.

### Syntax

EXTRACT\_RES\_BODY\_COUPLING: YES | NO

### Arguments

| Argument     | Description                                 |
|--------------|---------------------------------------------|
| YES          | Enables resistor body capacitor extraction  |
| NO (default) | Disables resistor body capacitor extraction |

### Description

Specifies the extraction of coupling capacitances between resistor body elements and interconnect features or ground. The coupling between a resistor body and interconnect layers is distributed between the two terminals of the resistor.

### Errors

The StarRC tool issues a warning message and ignores the following commands if they are used with the EXTRACT\_RES\_BODY\_COUPLING: YES command:

- COUPLE\_TO\_GROUND: YES RETAIN\_GATE\_CONTACT\_COUPLING
- COUPLE\_TO\_GROUND: YES RETAIN\_GATE\_CONTACT\_AND\_DIFFUSION\_COUPLING

### Example

EXTRACT\_RES\_BODY\_COUPLING: YES

### See Also

- [COUPLE\\_TO\\_GROUND](#)

---

## **EXTRACT\_VIA\_CAPS**

Performs a detailed via capacitance extraction.

### **Syntax**

`EXTRACT_VIA_CAPS: NO | YES [IGNORE_GATE_CONTACT_COUPLING]`

### **Arguments**

| Argument                     | Description                                                                                                                                                                                                                                                  |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO (default)                 | Ignores capacitive effect of vias and contacts. In general this setting uses less runtime and reports fewer capacitances. This is best used while developing designs and for technology nodes of 130 nm and above.                                           |
| YES                          | Consider the capacitive effect of vias and contacts. In general, this setting uses more runtime and reports the accuracy of the total net capacitance improvement. This is best used for signoff designs and those with technology nodes of 90 nm and below. |
| IGNORE_GATE_CONTACT_COUPLING | Ignores the gate contact coupling if SPICE models include gate-to-contact capacitance. Syntax required as follows:<br><code>EXTRACT_VIA_CAPS: YES IGNORE_GATE_CONTACT_COUPLING</code>                                                                        |

### **Description**

This command is used in conjunction with the [EXTRACTION](#) command.

If `EXTRACT_VIA_CAPS` is set to YES, StarRC considers the capacitive effects of all kinds of via and contact layers when it extracts parasitics. You do not need to prepare a new nxtgrd file to extract via capacitance in general; StarRC takes into account the via capacitance with a normal nxtgrd file.

The ITF option `GATE_TO_CONTACT_SMIN` should be used in addition to `SMIN` inside the ITF CONDUCTOR definition for polysilicon, for flows in which MOS gate-to-contact capacitance accuracy is relevant. This allows StarRC to use the actual gate-to-contact spacing when extracting contact capacitance because this spacing is typically smaller than the standard polysilicon `SMIN` spacing.

### **See Also**

- [EXTRACTION](#)

---

## FSCOMPARE\_COUPLING\_RATIO

Specifies the minimum ratio of the coupling capacitance to the total capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction.

### Syntax

`FSCOMPARE_COUPLING_RATIO: value`

### Arguments

| Argument           | Description                                                                               |
|--------------------|-------------------------------------------------------------------------------------------|
| <code>value</code> | Coupling capacitance ratio; a floating-point number between zero and one<br>Default: 0.10 |

### Description

The FSCOMPARE\_COUPLING\_RATIO command specifies the minimum ratio of the coupling capacitance to the total capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction. The results are filtered by the field solver results.

The specified value is applied to the capacitance values calculated by the field solver.

This command does not apply to field solver extractions specified by the FS\_EXTRACT\_NETS command.

### Example

`FSCOMPARE_COUPLING_RATIO: 0.2`

### See Also

- [EXTRACTION: FSCOMPARE](#)
- [FSCOMPARE\\_THRESHOLD](#)
- [FSCOMPARE\\_COUPLING\\_THRESHOLD](#)

---

## FSCOMPARE\_COUPLING\_THRESHOLD

Specifies the minimum coupling capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction.

### Syntax

FSCOMPARE\_COUPLING\_THRESHOLD: *value*

### Arguments

| Argument     | Description                                        |
|--------------|----------------------------------------------------|
| <i>value</i> | Coupling capacitance threshold<br>Default: 1.0e-15 |

### Description

The FSCOMPARE\_COUPLING\_THRESHOLD command specifies the minimum coupling capacitance required on a net to make it eligible for comparison in an FSCOMPARE extraction. The specified threshold is applied to the capacitance values calculated by the field solver.

The FSCOMPARE\_COUPLING\_THRESHOLD command applies only to coupling capacitances. Use the FSCOMPARE\_THRESHOLD command to filter total capacitances.

This command does not apply to field solver extractions specified by the FS\_EXTRACT\_NETS command.

### Example

Set the FSCOMPARE\_COUPLING\_THRESHOLD command to 0 to specify that no nets are eliminated based on coupling capacitance values.

```
FSCOMPARE_COUPLING_THRESHOLD: 0
```

### See Also

- [EXTRACTION: FSCOMPARE](#)
- [FSCOMPARE\\_COUPLING\\_RATIO](#)
- [FSCOMPARE\\_THRESHOLD](#)

---

## FSCOMPARE\_FILE\_PREFIX

Specifies the prefix to be attached to files generated by the EXTRACTION: FSCOMPARE command.

### Syntax

FSCOMPARE\_FILE\_PREFIX: *prefix*

### Arguments

| Argument | Description                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------|
| prefix   | A prefix to be attached to the beginning of file names<br>Default: block name specified in the StarRC command file |

### Description

This command specifies the prefix to be attached to files generated by the EXTRACTION: FSCOMPARE command.

This command does not apply to field solver extractions specified by the FS\_EXTRACT\_NETS command.

### Example

FSCOMPARE\_FILE\_PREFIX: myprefix  
myprefix.fs-compcoup

### See Also

- [EXTRACTION](#)

---

## FSCOMPARE\_OPTIONS

Specifies field solver options such as convergence goal and multiprocessing.

### Syntax

`FSCOMPARE_OPTIONS: option_1 [option_2 ...]`

### Arguments

| Argument                             | Description                                               |
|--------------------------------------|-----------------------------------------------------------|
| <code>option_1 [option_2 ...]</code> | Field solver options listed in <a href="#">Table 15-1</a> |

### Description

[Table 15-1](#) lists the options for the `FSCOMPARE_OPTIONS` command. The options apply to all field solver extraction regardless of whether it was invoked with the `FS_EXTRACT_NETS` command or the `EXTRACTION:FSCOMPARE` command.

*Table 15-1 Field Solver Options in the FSCOMPARE\_OPTIONS Command*

| Argument                           | Description                                                                                                                                                                                                               |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-f input_files</code>        | Specifies the input files. Specify the technology file before the design file.                                                                                                                                            |
| <code>-e list_of_nets</code>       | Specifies the list of nets to extract.<br>Default: all nonground nets                                                                                                                                                     |
| <code>-v</code>                    | Prints the program version.                                                                                                                                                                                               |
| <code>-np number_processors</code> | For distributed processing, sets the number of processors to use.<br>Default: 1                                                                                                                                           |
| <code>-nt number_threads</code>    | For multithreaded processing, sets the number of threads to use.<br>This number of threads is used on each processor if more than one processor is enabled with the <code>-np</code> option.<br>Default: 1<br>Maximum: 32 |
| <code>-time_out wait_time</code>   | For distributed processing, sets the maximum time that the master process waits for the subordinate machines to start.<br>Units: minutes<br>Default: 1440                                                                 |

**Table 15-1 Field Solver Options in the FSCOMPARE\_OPTIONS Command(Continued)**

| <b>Argument</b>                            | <b>Description</b>                                                                                                                                                                                                                                                            |
|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-mach_term retry_time</code>         | For distributed processing, sets the time to keep trying to contact a machine that has stopped responding. If the machine does not respond within the specified time, the job terminates.<br>Units: minutes<br>Default: 10                                                    |
| <code>-min_per_net minutes_per_net</code>  | Sets the maximum extraction time per net<br>Units: minutes<br>Default = 20                                                                                                                                                                                                    |
| <code>-l file_name</code>                  | Specifies the name of a file containing a list of client machines. For each client, specify a row with the following: <i>machine_name arch</i> . If <code>-l</code> is given, then LSF is not used for the clients.                                                           |
| <code>-perc_self self_cap_conv_goal</code> | Sets the self-capacitance convergence goal at one standard deviation as a percentage<br>Units: percent<br>Default: 0.5 for the FSCOMPARE flow<br>1.5 for the FS_EXTRACT_NETS flow                                                                                             |
| <code>-perc_coup coup_cap_conv_goal</code> | Sets the coupling capacitance convergence goal at one standard deviation as a percentage<br>Default: not checking                                                                                                                                                             |
| <code>-abs_coup abs_cap_conv_goal</code>   | Sets the absolute coupling capacitance convergence goal<br>The dynamic value of <code>abs_cap_conv_goal</code> is determined by multiplying the total net capacitance by the percentage set by the <code>-coup_cap_thresh</code> option.<br>Units: farads<br>Default: dynamic |
| <code>-coup_cap_thresh number</code>       | Sets the percentage of total capacitance at which to start checking coupling<br>Units: percent<br>Default: 1                                                                                                                                                                  |

**Table 15-1 Field Solver Options in the FSCOMPARE\_OPTIONS Command(Continued)**

| <b>Argument</b>                               | <b>Description</b>                                                                                                                                                                                                                        |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-perc_consistency max_dev</code>        | Sets the maximum deviation between identical nets, expressed as a percentage of the total capacitance<br>Units: percent<br>Default: none                                                                                                  |
| <code>-perc_of_nets_consistent number</code>  | Sets the percentage of identical nets that deviate from each other by the level specified by the <code>-perc_consistency</code> option<br>Units: percent<br>Default: 97                                                                   |
| <code>-perc_accuracy number</code>            | Sets the accuracy of the extracted capacitance value, expressed as a percentage of the capacitance value.<br>Note: if this parameter is specified, the <code>-perc_self</code> should not be specified.<br>Units: percent<br>Default: 1.5 |
| <code>-perc_accuracy_confidence number</code> | Sets the confidence level for the estimated capacitance value, extracted at the accuracy level set by the <code>-perc_accuracy</code> option, expressed as a percentage<br>Units: percent<br>Default: 99.7                                |
| <code>-min_cap cap_value</code>               | Sets the minimum capacitance value to report in the output file<br>Units: farads<br>Default: 1.0e-20                                                                                                                                      |
| <code>-seed rand_num_seed</code>              | Sets the random number seed<br>Default: 12345                                                                                                                                                                                             |
| <code>-bb xl yl xh yh</code>                  | Sets the bounding box<br>Units: grid units<br>Default: 100 microns larger than design                                                                                                                                                     |
| <code>-neuman_x</code>                        | Uses Neumann boundary conditions on x-boundaries<br>Default: false                                                                                                                                                                        |

*Table 15-1 Field Solver Options in the FSCOMPARE\_OPTIONS Command(Continued)*

| <b>Argument</b> | <b>Description</b>                                                                                                       |
|-----------------|--------------------------------------------------------------------------------------------------------------------------|
| -neuman_y       | Uses Neumann boundary conditions on y-boundaries<br>Default: false                                                       |
| -periodic_x     | Uses periodic boundary conditions on x-boundaries<br>Default: false                                                      |
| -periodic_y     | Uses periodic boundary conditions on y-boundaries<br>Default: false                                                      |
| -match          | Enables pattern matching and improves runtime and accuracy for symmetric or identical net extraction in the field solver |

To obtain a complete list of the field solver options, enter `fieldsolver -help` on the command line.

## Examples

To run two clients with a one percent self-capacitance goal, use the following command:

```
FSCOMPARE_OPTIONS: -np 2 -perc_self 1
```

To run four clients with a 10 percent coupling capacitance and one percent self-capacitance goal, use the following command:

```
FSCOMPARE_OPTIONS: -np 4 -perc_coup 10 -perc_self 1
```

## See Also

- [EXTRACTION: FSCOMPARE](#)
- [FS\\_EXTRACT\\_NETS](#)
- [FSCOMPARE\\_COUPLING\\_RATIO](#)
- [FSCOMPARE\\_COUPLING\\_THRESHOLD](#)
- [FSCOMPARE\\_THRESHOLD](#)

---

## FSCOMPARE\_THRESHOLD

Specifies the minimum total capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction.

### Syntax

`FSCOMPARE_THRESHOLD: value`

### Arguments

| Argument           | Description                               |
|--------------------|-------------------------------------------|
| <code>value</code> | Capacitance threshold<br>Default: 3.0e-15 |

### Description

`FSCOMPARE_THRESHOLD` specifies the minimum total capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction. The specified threshold is applied to the total capacitance values calculated by the field solver.

The `FSCOMPARE_THRESHOLD` command does not apply to coupling capacitances. Use the `FSCOMPARE_COUPLING_THRESHOLD` command to filter coupling capacitances.

### Example

Setting `FSCOMPARE_THRESHOLD` to 0 ensures that no nets are eliminated based on capacitance values.

```
FSCOMPARE_THRESHOLD: 0
```

### See Also

- [EXTRACTION: FSCOMPARE](#)
- [FS\\_EXTRACT\\_NETS](#)
- [FSCOMPARE\\_COUPLING\\_RATIO](#)
- [FSCOMPARE\\_COUPLING\\_THRESHOLD](#)
- [FSCOMPARE\\_OPTIONS](#)

---

## FS\_BOUNDARY\_BOX

Overrides the default boundary box for the field solver.

### Syntax

`FS_BOUNDARY_BOX: x_min y_min z_min x_max y_max z_max`

### Arguments

| Argument                                         | Description                                |
|--------------------------------------------------|--------------------------------------------|
| <code>x_min y_min z_min x_max y_max z_max</code> | Boundary box coordinates<br>Units: microns |

### Description

The `FS_BOUNDARY_BOX` command overrides the default boundary box for the field solver.

StarRC scales the x- and y- coordinates, but not the z-coordinates, of the boundary box when you specify either of the following:

- The `MAGNIFICATION_FACTOR` command in the StarRC command file
- The `HALF_NODE_SCALE_FACTOR` statement in the ITF file

Note:

Although you must specify `z_min` and `z_max` values, StarRC ignores them.

### Errors

If you do not specify the `FS_BOUNDARY_BOX` and `FS_BOUNDARY_CONDITION` commands together, StarRC issues an error.

### Example

In the following example, the design data is expressed as 10000 grids per user unit:

`FS_BOUNDARY_BOX: 1.0 0.16 0.1 17.50 1.45 7.40`

This command translates to the following setting, as reported in the “Invoking the Field Solver with Command” section of the StarRC summary file:

`FSCOMPARE_OPTIONS: -bb 10000 1600 175000 14500`

Alternatively, if you specify `MAGNIFICATION_FACTOR: 2.0`, then the previous `FS_BOUNDARY_BOX` command translates to the following setting:

`FSCOMPARE_OPTIONS: -bb 20000 3200 350000 29000`

## See Also

- [FS\\_BOUNDARY\\_CONDITION](#)
- [FSCOMPARE\\_OPTIONS](#)
- [HALF\\_NODE\\_SCALE\\_FACTOR](#)
- [MAGNIFICATION\\_FACTOR](#)

---

## FS\_BOUNDARY\_CONDITION

Overrides the default boundary conditions used in the field solver.

### Syntax

FS\_BOUNDARY\_CONDITION:

[ -bc\_xn N | P ] [-bc\_xp N | P] [-bc\_yn N | P] [-bc\_yp N | P]

### Arguments

| Argument | Description                                     |
|----------|-------------------------------------------------|
| -bc_xn   | Boundary condition for the negative x-direction |
| -bc_xp   | Boundary condition for the positive x-direction |
| -bc_yn   | Boundary condition for the negative y-direction |
| -bc_yp   | Boundary condition for the positive y-direction |
| N        | Neumann boundary conditions                     |
| P        | Periodic boundary conditions                    |

### Description

The FS\_BOUNDARY\_CONDITION command overrides the default boundary conditions used in the field solver.

You must specify

- The same boundary condition for the negative and positive x-directions
- The same boundary condition for the negative and positive y-directions

If you specify different boundary conditions for the positive and negative directions, the Neumann boundary condition overrides the periodic boundary condition.

### Errors

If you do not specify the FS\_BOUNDARY\_BOX and FS\_BOUNDARY\_CONDITION commands together, StarRC issues an error.

## Example

The following examples specifies the use of Neumann boundary conditions on x-boundaries and periodic boundary conditions on y-boundaries:

```
FS_BOUNDARY_CONDITION: -bc_xn N -bc_xp N -bc_yn P -bc_yp P
```

This command translates to the following setting, as reported in the “Invoking the Field Solver with Commands” section of the StarRC summary file:

```
FSCOMPARE_OPTIONS: -neuman_x -periodic_y
```

## See Also

- [FS\\_BOUNDARY\\_BOX](#)
- [FSCOMPARE\\_OPTIONS](#)

---

## FS\_DP\_STRING

Specifies the distributed processing method and job control parameters for field solver extraction runs.

### Syntax

```
FS_DP_STRING:
 bsub lsf_arguments
 | qsub gridware_arguments
 | list list_of_machines
 | nc sub rtida_arguments
```

### Arguments

| Argument                  | Description                                      |
|---------------------------|--------------------------------------------------|
| <i>lsf_arguments</i>      | Arguments for an LSF system                      |
| <i>gridware_arguments</i> | Arguments for a Gridware system                  |
| <i>list_of_machines</i>   | List of machines on a general network            |
| <i>rtida_arguments</i>    | Arguments for a Runtime Design Automation system |

### Description

The job submission command can be specified either in an environment variable or as a command in the StarRC command file. If set in both places, the StarRC command file takes precedence. The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Specify the number of processors and number of threads per processor to use for field solver distributed processing with the `FSCOMPARE_OPTIONS` command.

Distributed processing is available for the following computing environments:

- LSF system
- Gridware system
- General network with a list of machines
- Runtime Design Automation (RTDA) system

## Examples

On an LSF system:

```
FS_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
FS_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

On a general network with a list of machines:

```
FS_DP_STRING: list alpha beta gamma
```

On an RTDA system:

```
FS_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

## See Also

- [FSCOMPARE\\_OPTIONS](#)
- [Distributed Processing for Field Solver Jobs](#)

---

## FS\_EXTRACT\_NETS

Specifies the nets to be extracted by the field solver.

### Syntax

FS\_EXTRACT\_NETS: *net\_names*

### Arguments

| Argument         | Description                                                      |
|------------------|------------------------------------------------------------------|
| <i>net_names</i> | List of nets for field solver extraction. Wildcards can be used. |

### Description

The FS\_EXTRACT\_NETS command specifies a list of nets for which field solver extraction should be used. The resulting netlist combines the nets extracted by StarRC and the field solver.

StarRC creates comparison tables for both total and coupling capacitances with the FS\_EXTRACT\_NETS command. This enables you to generate an accurate parasitic netlist along with a validation report. This is available for all flows.

Distributed processing is supported by setting options with the FSCOMPARE\_OPTIONS command.

The NET\_TYPE:SCHEMATIC command is supported when used with the FS\_EXTRACT\_NETS command in the Calibre flow.

### Example

In the following example, StarRC extracts all nets, but the field solver is used to extract the three nets specified by the FS\_EXTRACT\_NETS command:

```
NETS: *
FS_EXTRACT_NETS: net1 net2 net3
NET_TYPE: SCHEMATIC
```

### See Also

- [FSCOMPARE\\_OPTIONS](#)
- [NET\\_TYPE](#)

---

## FS\_IGNORE\_GATE\_CHANNEL\_CAPACITANCE

For field solver analysis, restricts extraction of certain capacitances of a MOS device.

### Syntax

`FS_IGNORE_GATE_CHANNEL_CAPACITANCE: YES | NO`

### Arguments

| Argument     | Description                                      |
|--------------|--------------------------------------------------|
| YES          | Enables only outer fringe capacitance extraction |
| NO (default) | Performs standard extraction                     |

### Description

The `FS_IGNORE_GATE_CHANNEL_CAPACITANCE` command provides control over the extraction of gate to diffusion capacitances. The gate to diffusion capacitance usually includes both the capacitance between the gate conductor and the top surface of the diffusion region (outer fringe capacitance) and the capacitance between the gate conductor and the side of the diffusion region (inner fringe capacitance). Setting the command to `YES` specifies to extract only the outer fringe capacitance.

This command interacts with statements in the ITF file and other commands in the command file, as follows:

- If the ITF file contains a `GATE_TO_DIFFUSION_CAP` section, the `FS_IGNORE_GATE_CHANNEL_CAPACITANCE` command is overridden.  
The `IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING` command is also required.
- If the ITF file does not contain a `GATE_TO_DIFFUSION_CAP` section and the `FS_IGNORE_GATE_CHANNEL_CAPACITANCE` command is set to `YES`, only the outer fringe capacitance is extracted.  
The gate conductor must be identified as layer type `GATE` in the ITF file. The `IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING` command is also required.
- If the ITF file does not contain a `GATE_TO_DIFFUSION_CAP` section and the `FS_IGNORE_GATE_CHANNEL_CAPACITANCE` command is set to `NO` (or does not appear in the command file), the extracted gate to diffusion capacitance includes both the outer and inner fringe components.

**Example**

```
FS_IGNORE_GATE_CHANNEL_CAPACITANCE: YES
```

**See Also**

- [EXTRACTION](#): FSCOMPARE
- [FS\\_EXTRACT\\_NETS](#)
- [FSCOMPARE\\_OPTIONS](#)
- [GATE\\_TO\\_DIFFUSION\\_CAP](#)
- [IGNORE\\_CAPACITANCE](#)

---

## GDS\_FILE

Specifies GDSII format files to represent part of the physical layout.

### Syntax

`GDS_FILE: file1 [file2] ...`

### Arguments

| Argument                       | Description                                                                  |
|--------------------------------|------------------------------------------------------------------------------|
| <code>file1 [file2] ...</code> | Names of GDSII files containing physical layout information<br>Default: none |

### Description

The `GDS_FILE` command specifies GDSII format files to represent part of the physical layout. You can specify gzip and compressed GDSII files for this command.

In the Milkyway flow, the `GDS_FILE` command merges GDSII data into FRAM and CEL views for skip cells to provide a full physical layout representation. StarRC uses only the pin shapes in the FRAM view for the cells that are defined both by the FRAM view and the GDSII file. StarRC replaces obstructions with material defined within GDS cells of the same name. If no matching cell name is found within the GDSII file for a particular FRAM cell, the FRAM obstruction is used for that cell.

In the LEF/DEF flow, this command merges GDSII data into LEF MACRO definitions for skip cells to provide a full physical layout representation. StarRC uses only the pin shapes from the LEF MACRO for the cells that are defined by both the LEF MACRO and the GDS file. Any material defined within the OBS section of the LEF MACRO is overwritten and replaced by material defined within the GDS cell of the same name. If no matching cell name is found, the OBS section of the corresponding LEF MACRO is used for that cell.

The `GDS_FILE` command can be specified multiple times. It must be used with the `GDS_LAYER_MAP_FILE` command, but cannot be used with the `OASIS_FILE` command.

### See Also

- [GDS\\_LAYER\\_MAP\\_FILE](#)
- [SKIP\\_CELLS](#)

---

## GDS\_LAYER\_MAP\_FILE

Specifies the mapping between the GDSII layer number and layer name in the design database.

### Syntax

`GDS_LAYER_MAP_FILE: file_name`

### Arguments

| Argument                      | Description                                   |
|-------------------------------|-----------------------------------------------|
| <code><i>file_name</i></code> | The GDSII layer mapping file<br>Default: none |

### Description

The `GDS_LAYER_MAP_FILE` command specifies the mapping between the GDSII layer number and layer name in the design database whenever the `GDS_FILE` or `METAL_FILL_GDS_FILE` command is used to import GDSII data into the design database.

#### Note:

You cannot use the `GDS_LAYER_MAP_FILE` command with the `OASIS_LAYER_MAP_FILE` command.

All translated GDSII layers must have an entry in the file specified by the `GDS_LAYER_MAP_FILE` command and must have a definition in the layout database. Use the `GDS_LAYER_MAP_FILE` command to import GDSII cell material into a Milkyway or LEF/DEF database or to import metal fill polygons into a Milkyway, LEF/DEF, or Calibre Connectivity Interface database. If you use the `METAL_FILL_GDS_FILE` and `GDS_FILE` commands in a single Milkyway or LEF/DEF run, use a unified layer mapping file for the GDSII files.

An error occurs if any layer is specified in the file does not have a corresponding layer in the layout database.

The GDS layer map file uses the following syntax:

```
database_layer gdsii_layer_number gdsii_datatype
 [FLOATING | GROUNDED | IGNORE]
```

| Argument                  | Description                                                                                                                                                                        |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>database_layer</i>     | The database layer name                                                                                                                                                            |
| <i>gdsii_layer_number</i> | The GDSII layer number                                                                                                                                                             |
| <i>gdsii_datatype</i>     | The GDSII data type. If a GDSII data type is not specified, then all data types on a given layer are read.                                                                         |
| FLOATING                  | Specifies that the corresponding fill layer is to be treated as floating. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored. |
| GROUNDED                  | Specifies that the corresponding fill layer is to be treated as grounded. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored. |
| IGNORE                    | Specifies that the corresponding fill layer is to be ignored. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.             |

The layer-specific fill-handling keyword allows you to decide how individual metal fill layers are handled during parasitic extraction. This handling is considered only if the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command is set in the StarRC command file.

If the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command appears in the StarRC command file but a fill-handling mode is not specified for a layer in the GDS layer map file, the default setting is FLOATING for that layer. If another setting of the METAL\_FILL\_POLYGON\_HANDLING command (other than AUTOMATIC) is specified, that setting governs the handling of all layers, and any layer-specific mode specifications inside the GDS layer map file are ignored.

In the example, handling mode GROUNDED is specified for layer DIFF. If the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command is also specified in the StarRC command file, DIFF is treated as GROUNDED while all other layers are treated as FLOATING. If METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC is not specified in the command file, the layer-specific mode specification for DIFF is ignored, and the global mode set by the METAL\_FILL\_POLYGON\_HANDLING command takes precedence.

## Examples

The following example shows how the DIFF layer is assigned to GDSII layer 2 and GDSII datatype 0. This example maps layers from a metal fill GDS file and specifies layer-specific fill handling for the DIFF layer.

### Example 15-4 Layer-Specific Fill Handling

```
DIFF 2 0 GROUNDED
POLY 7 0
CONT 4 0
METAL1 10 0
METAL1 10 1
METAL1 76 0
VIA1 11 0
METAL2 12 0
```

In the following example, the `METAL_FILL_POLYGON_HANDLING: AUTOMATIC` command is set in the command file.

### Example 15-5 Automatic Fill Handling

```
*layer treated as grounded
DIFF 2 0 GROUNDED
*layer treated as floating
POLY 7 0 FLOATING
*layer governed by default floating mode since mode is unspecified.
METAL1 4 0
```

## See Also

- [GDS\\_FILE](#)
- [LEF\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE](#)

---

## GPD

Specifies the name of the directory in which to store binary parasitic data for gate-level extraction.

### Syntax

GPD: *parasitics\_dir*

### Arguments

| Argument              | Description                                                                                                                                                               |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parasitics_dir</i> | The name of the parasitics database directory<br>Default: <i>block.gpd</i> , where <i>block</i> is the top-level design block specified by the <code>BLOCK</code> command |

### Description

The Galaxy Parasitic Database (GPD) is a distributed and scalable parasitic database that is designed for efficient communication between the StarRC tool and the PrimeTime tool.

The GPD flow is enabled by default for most gate-level flows. You can optionally use the `GPD` StarRC command to specify the name of the directory in which to store the parasitic data. The default is to use the name specified by the `BLOCK` command.

Transistor-level, field solver, clock net inductance, and power extraction flows are not supported. In this case, the StarRC tool does not create a GPD directory.

### See Also

- [GPD\\_DP\\_STRING](#)
- [The Galaxy Parasitic Database](#)

---

## GPD\_DP\_STRING

Specifies distributed processing conditions for use in a Galaxy Parasitic Database configuration file. Valid only in a GPD configuration file.

### Syntax

```
GPD_DP_STRING:
 bsub lsf_arguments
 | qsub gridware_arguments
 | list list_of_machines
 | list localhost num_processes
 | nc sub rtida_arguments
```

### Arguments

| Argument                  | Description                                      |
|---------------------------|--------------------------------------------------|
| <i>lsf_arguments</i>      | Arguments for an LSF system                      |
| <i>gridware_arguments</i> | Arguments for a Gridware system                  |
| <i>list_of_machines</i>   | List of machines on a general network            |
| <i>num_processes</i>      | Number of processes on the local host            |
| <i>rtida_arguments</i>    | Arguments for a Runtime Design Automation system |

### Description

Use this command in a Galaxy Parasitic Database (GPD) configuration file to specify distributed processing conditions for netlist creation that are different from the distributed processing conditions used in the original extraction run. Distributed processing in the extraction run is controlled by the STARRC\_DP\_STRING command.

The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- LSF system
- Gridware system
- General network with a list of machines
- Runtime Design Automation (RTDA) system

## Examples

On an LSF system:

```
GPD_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
GPD_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

On a general network with a list of machines:

```
GPD_DP_STRING: list alpha beta gamma
```

On an RTDA system:

```
GPD_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

## See Also

- [GPD](#)
- [The Galaxy Parasitic Database](#)
- [STARRC\\_DP\\_STRING](#)
- [Distributed Processing](#)

---

## HIERARCHICAL\_COUPLING\_ABS\_THRESHOLD

Specifies an absolute threshold for reporting hierarchical coupling capacitances.

### Syntax

`HIERARCHICAL_COUPLING_ABS_THRESHOLD: threshold`

### Arguments

| Argument         | Description                                               |
|------------------|-----------------------------------------------------------|
| <i>threshold</i> | Absolute threshold<br>Units: farads (F)<br>Default: 3e-15 |

### Description

Hierarchical coupling capacitance occurs between extracted signal nets and skip cell instances. The hierarchical coupling capacitance includes the capacitance to all signal nets inside the instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The `HIERARCHICAL_COUPLING_REPORT_FILE` command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the `HIERARCHICAL_COUPLING_REL_THRESHOLD` command.
- The number of reported capacitances is smaller than the value specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

### See Also

- [HIERARCHICAL\\_COUPLING\\_REPORT\\_NUMBER](#)
- [HIERARCHICAL\\_COUPLING\\_REPORT\\_FILE](#)
- [HIERARCHICAL\\_COUPLING\\_REL\\_THRESHOLD](#)

---

## HIERARCHICAL\_COUPLING\_REPORT\_NUMBER

Specifies the number of nets to be reported in hierarchical coupling capacitance extraction.

### Syntax

`HIERARCHICAL_COUPLING_REPORT_NUMBER: no_of_nets`

### Arguments

| Argument                | Description                                                                                  |
|-------------------------|----------------------------------------------------------------------------------------------|
| <code>no_of_nets</code> | Integer number of nets for which to report hierarchical coupling capacitors<br>Default: 1000 |

### Description

Hierarchical coupling capacitance occurs between extracted signal nets and skip cell instances. The hierarchical coupling capacitance includes the capacitance to all signal nets inside the instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The `HIERARCHICAL_COUPLING_REPORT_FILE` command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the `HIERARCHICAL_COUPLING_REL_THRESHOLD` command.
- The number of reported capacitances is smaller than the value specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

### See Also

- [HIERARCHICAL\\_COUPLING\\_ABS\\_THRESHOLD](#)
- [HIERARCHICAL\\_COUPLING\\_REPORT\\_FILE](#)
- [HIERARCHICAL\\_COUPLING\\_REL\\_THRESHOLD](#)

## HIERARCHICAL\_COUPLING\_REPORT\_FILE

Enables hierarchical coupling capacitance analysis for gate-level extraction and specifies a file name for the report.

### Syntax

HIERARCHICAL\_COUPLING\_REPORT\_FILE: *file\_name*

### Arguments

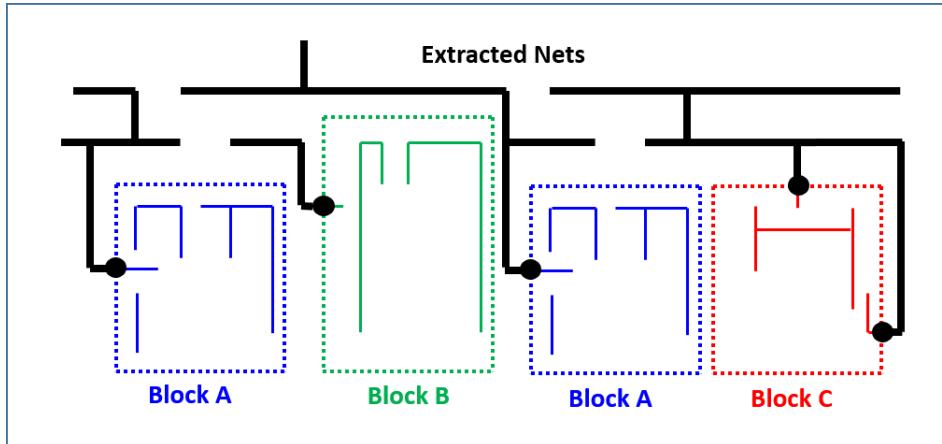
| Argument         | Description                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>file_name</i> | Enables the hierarchical coupling capacitance feature and specifies a file name for the report<br>Default: none |

### Description

The HIERARCHICAL\_COUPLING\_REPORT\_FILE command enables the extraction of coupling capacitance between extracted signal nets and skipped cells and provides a name for the report.

Figure 15-11 illustrates a design that contains four skip cells. By default, features in skip cells are treated as ground during extraction. However, the coupling capacitance between extracted signal nets and skip cells can be significant and might affect timing analysis. Analyzing hierarchical coupling capacitance provides insight into this occurrence.

Figure 15-11 Relationship Between Extracted Nets and Skip Cell Nets



The hierarchical coupling capacitance includes the capacitance to all signal nets inside the skip cell instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

The hierarchical coupling capacitance report presents coupling capacitances in decreasing order of the relative percentage of coupling capacitance to total capacitance for any extracted signal net. The report contains the number of entries specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Capacitances are filtered by the values of the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` and `HIERARCHICAL_COUPLING_REL_THRESHOLD` commands.

In a simultaneous multicorner flow, the coupling report is generated only for the primary corner.

## Example

The following example shows the contents of a report file:

```
* 637 worst coupling capacitances listed in descending order:
* relative weight(%) coupling capacitance Extracted net name instance name
 23.2 1.386e-15 I237/B77 I20 (adder1)
 21.8 2.733e-15 T336/W22/I90 I31 (INV2)
 16.4 5.33e-14 PACKAGE_3 I20 (adder1)
 15.1 3.773e-15 GB_32772 I17 (mux2d4)
 ...
```

In this example, critical signal net I237/B77 has 23.2 percent of its total capacitance coupling to signal nets in skip cell instance I20. The magnitude of the coupling capacitance is 1.386e-15 Farads.

## See Also

- [HIERARCHICAL\\_COUPLING\\_ABS\\_THRESHOLD](#)
- [HIERARCHICAL\\_COUPLING\\_REPORT\\_NUMBER](#)
- [HIERARCHICAL\\_COUPLING\\_REL\\_THRESHOLD](#)

---

## HIERARCHICAL\_COUPLING\_REL\_THRESHOLD

Specifies the ratio of coupling to total capacitance.

### Syntax

`HIERARCHICAL_REL_THRESHOLD: threshold`

### Arguments

| Argument               | Description                                                                              |
|------------------------|------------------------------------------------------------------------------------------|
| <code>threshold</code> | Relative capacitance threshold; a floating point number between 0 and 1<br>Default: 0.03 |

### Description

Hierarchical coupling capacitance occurs between extracted signal nets and skip cell instances. The hierarchical coupling capacitance includes the capacitance to all signal nets inside the instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The `HIERARCHICAL_COUPLING_REPORT_FILE` command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the `HIERARCHICAL_COUPLING_REL_THRESHOLD` command.
- The number of reported capacitances is smaller than the value specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

### See Also

- [HIERARCHICAL\\_COUPLING\\_ABS\\_THRESHOLD](#)
- [HIERARCHICAL\\_COUPLING\\_REPORT\\_NUMBER](#)
- [HIERARCHICAL\\_COUPLING\\_REPORT\\_FILE](#)

---

## HIERARCHICAL\_SEPARATOR

Specifies the character used as the hierarchical delimiter during extraction and netlist creation.

### Syntax

HIERARCHICAL\_SEPARATOR: | | / | . | :

### Arguments

| Argument    | Description          |
|-------------|----------------------|
|             | Pipe ( ) character   |
| / (default) | Slash (/) character  |
| .           | Period (.) character |
| :           | Colon (:) character  |

### Description

The HIERARCHICAL\_SEPARATOR command specifies the character used as the hierarchical delimiter during extraction and netlist creation. If hierarchical nets are specified with the NETS command, this character must be used to derive flattened names for the selection. Do not use the hierarchical separator character in a net or instance name; the tool returns an error message for this condition.

### Example

This example sets the hierarchical separator to the period (.).

```
HIERARCHICAL_SEPARATOR: .
```

### See Also

- [BUS\\_BIT](#)
- [NETS](#)

---

## HN\_NETLIST\_MODEL\_NAME

Writes a simulation model name instead of the StarRC model name in the parasitic netlist.

### Syntax

`HN_NETLIST_MODEL_NAME: ref_model_name SPICE_model_name`

### Arguments

| Argument                      | Description                                                                                        |
|-------------------------------|----------------------------------------------------------------------------------------------------|
| <code>ref_model_name</code>   | The model name in the schematic or layout; no wildcards allowed<br>Default: none                   |
| <code>SPICE_model_name</code> | The internal database is updated with this SPICE model name; no wildcards allowed<br>Default: none |

### Description

This command updates the internal database with the model name provided in the argument list. The command also controls the model names of devices in an ideal netlist output by StarRC with the `NETLIST_IDEAL_SPICE_FILE` command. The `MODEL_TYPE` command determines whether StarRC looks for a reference model in the layout or schematic netlist.

If the specified model is not present, StarRC issues a warning message. If the same model name is specified more than one time, the last command overrides earlier commands.

You can map multiple reference models to a single simulation model.

If you specify the `XREF_USE_LAYOUT_DEVICE_NAME` and `HN_NETLIST_MODEL_NAME` command in the same command file, the `HN_NETLIST_MODEL_NAME` setting overrides the `XREF_USE_LAYOUT_DEVICE_NAME` setting.

### Example

If you have multiple device models, specify the command as follows:

```
HN_NETLIST_MODEL_NAME: p_dev pmos
HN_NETLIST_MODEL_NAME: pdev2 pmos-2
```

### See Also

- [MODEL\\_TYPE](#)
- [NETLIST\\_IDEAL\\_SPICE\\_FILE](#)

---

## **HN\_NETLIST\_SPICE\_TYPE**

Specifies StarRC netlist standard devices as SPICE .SUBCKT calls beginning with X.

### **Syntax**

`HN_NETLIST_SPICE_TYPE: device_model_name X`

### **Arguments**

| <b>Argument</b>                | <b>Description</b>                                                                                                                                          |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>device_model_name</code> | For a flow based on Hercules, any layout model name specified in a device extraction command, or any schematic model name specified in an EQUATE statement. |
|                                | For a flow based on Calibre, any model name or netlist model name specified in a DEVICE command.                                                            |
|                                | Default: none                                                                                                                                               |

### **Description**

Specifies StarRC netlist standard devices as SPICE .SUBCKT calls beginning with X. When you specify this command for a standard, nonuser-defined ideal device, the SPICE device card is replaced with an X card in the netlist.

For cases in which multiple device extraction commands match a single `device_model_name` specified with the `HN_NETLIST_SPICE_TYPE` command, X device names are included in the netlist for devices from all matching commands.

In Hercules flows, the SPICE device name can be set directly in the Hercules runset using the `DEVICE_PREFIX` option. This setting is automatically propagated into the StarRC output netlist even if you do not use the `HN_NETLIST_SPICE_TYPE` command. The Hercules `DEVICE_PREFIX` option is defined using a string argument. StarRC only includes the first character of that string argument in the netlist. See the Hercules documentation for more details about the option `DEVICE_PREFIX`.

In the Calibre flow, the `device_model_name` is automatically read from block.devtab for model name and model card information:

```
DEVICE {element_name [(model_name)]} device_layer...
[NETLIST MODEL netlist_model_name] [NETLIST ELEMENT
netlist_element_name]
...
[[property_specification]]
```

StarRC treats an element or model name as a layout name and a netlist element name as a schematic name.

**Example**

HN\_NETLIST\_SPICE\_TYPE: p X

**See Also**

- [MODEL\\_TYPE](#)

---

## **ICV\_ANNOTATION\_FILE**

Specifies the IC Validator annotation file.

### **Syntax**

`ICV_ANNOTATION_FILE: file_name`

### **Arguments**

| <b>Argument</b>        | <b>Description</b>                                                      |
|------------------------|-------------------------------------------------------------------------|
| <code>file_name</code> | Name of the gzip-format IC Validator annotation file<br>Default: adp.gz |

### **Description**

The `ICV_ANNOTATION_FILE` command specifies the IC Validator annotation file. This annotation file is produced by IC Validator to extract advanced device properties efficiently such as the well-proximity effect (WPE) and the length of diffusion (LOD). This command also triggers the IC Validator Dual Hierarchy Extraction (DHE) flow. You do not need the `ICV_ANNOTATION_FILE` command in the StarRC command file if the `annotation_file` statement exists in your IC Validator runset report file.

The IC Validator annotation file must be compressed into the gzip format.

In this annotation file, a line starting with the asterisk (\*) character is considered a comment and ignored. The annotation file contains device property data in the SPICE format.

### **Example**

`ICV_ANNOTATION_FILE: ./my_icv_adp.gz`

#### *Example 15-6 Syntax of the IC Validator Annotation File*

```
- Define File
property_annotation_file (
 file = "string" //optional
);
- Initiate DHE Flow
init_device_matrix(dual_hierarchy_extraction=true)
- Write Annotation file
write_annotation_file (
 device_db = device_database,
 output_file = property_annotation_file_handle,
 precision = integer //optional
);
```

**Example 15-7 Example of an IC Validator Annotation File**

```
.SUBCKT mc_co_stld_4x8
M0 sa =0.11u sal=1.1e-07 sa2=1.1e-07 sa3=1.1e-07 sb=0.29u sb1=2.9e-07
sb2=2.9e-07 sb3=2.9e-07
M1 sa=0.11u sal=1.1e-07 sa2=1.1e-07 sa3=1.1e-07 sb=0.11u sb1=1.1e-07
sb2=1.1e-07 sb3=1.1e-07
.ENDS

.SUBCKT blkcontrolc
M3 sa=0.29u sal=2.9e-07 sa2=2.9e-07 sa3=2.9e-07 sb=1.19u sb1=1.19e-06
sb2=1.19e-06 sb3=1.19e-06
M4 sa=1.01u sal=1.01e-06 sa2=1.01e-06 sa3=1.01e-06 sb=0.47u sb1=4.7e-07
sb2=4.7e-07 sb3=4.7e-07 sca=1.19602 scb=1.91168e-06 scc=2.37977e-12
spa=1.4e-07 spa1=1.4e-07 spa2=1.4e-07 spba=1e-06
X1/M2 sa=1.01u sal=1.01e-06 sa2=1.01e-06 sa3=1.01e-06 sb=0.47u
sb1=4.7e-07 sb2=4.7e-07 sb3=4.7e-07 sca=1.19602 scb=1.91168e-06
scc=2.37977e-12 spa=1.4e-07 spa1=1.4e-07 spa2=1.4e-07 spba=1e-06
.ENDS
```

**See Also**

- [ICV\\_RUNSET\\_REPORT\\_FILE](#)

---

## ICV\_LVS\_DEVICE\_TYPE\_CAP

Identifies user-defined capacitance devices for the IC Validator flow.

### Syntax

ICV\_LVS\_DEVICE\_TYPE\_CAP: *list\_of\_C\_devices*

### Arguments

| Argument                 | Description                              |
|--------------------------|------------------------------------------|
| <i>list_of_C_devices</i> | List of user-defined capacitance devices |

### Description

This command identifies user-defined capacitors as CAP devices.

The *list\_of\_C\_devices* argument follows the case-sensitivity set by the CASE\_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts in certain situations.

### Example

ICV\_LVS\_DEVICE\_TYPE\_CAP: cap\_ss

### See Also

- [ICV\\_LVS\\_DEVICE\\_TYPE\\_MOS](#)
- [ICV\\_LVS\\_DEVICE\\_TYPE\\_RES](#)
- [ICV\\_OPTIONAL\\_DEVICE\\_PIN\\_FILE](#)

---

## ICV\_LVS\_DEVICE\_TYPE\_MOS

Identifies user-defined MOS devices for the IC Validator flow.

### Syntax

ICV\_LVS\_DEVICE\_TYPE\_MOS: *list\_of\_M\_devices*

### Arguments

| Argument                 | Description                      |
|--------------------------|----------------------------------|
| <i>list_of_M_devices</i> | List of user-defined MOS devices |

### Description

This command identifies user-defined MOS devices.

The *list\_of\_M\_devices* argument follows the case-sensitivity set by the CASE\_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts in certain situations.

### Example

ICV\_LVS\_DEVICE\_TYPE\_MOS: nmos\_ss

### See Also

- [ICV\\_LVS\\_DEVICE\\_TYPE\\_CAP](#)
- [ICV\\_LVS\\_DEVICE\\_TYPE\\_RES](#)
- [ICV\\_OPTIONAL\\_DEVICE\\_PIN\\_FILE](#)

---

## ICV\_LVS\_DEVICE\_TYPE\_RES

Identifies user-defined resistors as RES devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command. Valid for the IC Validator flow.

### Syntax

`ICV_LVS_DEVICE_TYPE_RES: list_of_R_devices`

### Arguments

| Argument                       | Description                      |
|--------------------------------|----------------------------------|
| <code>list_of_R_devices</code> | List of user-defined RES devices |

### Description

The `ICV_LVS_DEVICE_TYPE_RES` statement identifies user-defined resistors as RES devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command.

### Example

In the following example, the `rp_sio` and `pwr_rm1` devices defined in the rule file are identified as resistors:

```
ICV_LVS_DEVICE_TYPE_RES: rp_sio pwr_rm1
```

### See Also

- [ICV\\_LVS\\_DEVICE\\_TYPE\\_CAP](#)
- [ICV\\_LVS\\_DEVICE\\_TYPE\\_MOS](#)
- [ICV\\_OPTIONAL\\_DEVICE\\_PIN\\_FILE](#)
- [WIDE\\_DEVICE\\_TERM\\_RESISTANCE](#)

---

## ICV\_OPTIONAL\_DEVICE\_PIN\_FILE

Assigns nonstandard device terminals by name for the IC Validator flow.

### Syntax

`ICV_OPTIONAL_DEVICE_PIN_FILE: file_name`

### Arguments

| Argument               | Description                 |
|------------------------|-----------------------------|
| <code>file_name</code> | Name of the device pin file |

### Description

By default, StarRC assigns nonstandard device terminals by position. However, if you specify the `ICV_OPTIONAL_DEVICE_PIN_FILE` command, StarRC assigns the device terminal by the name in the specified file.

The syntax of each line in the file is as follows:

```
device_type model_name[seed_layer] pin_layer_1 (pin name 1) \
 pin_layer_2 (pin name 2) pin_layer_3 (pin name 3) ...
```

Valid values for the `device_type` parameter are M (MOS device), C (capacitor), and R (resistor).

The seed layer is optional, but it must be used when multiple devices have the same model name.

### Example

In the following example, `devpin_file` contains the list of device terminal names:

```
ICV_OPTIONAL_DEVICE_PIN_FILE: devpin_file
```

The following lines show an example of a device pin file:

```
M MOS_DEV NDIFSI_D (D) POLY (G) NDIFSI_S (S) NWELL (B)
C CAP_DEV CAP_TOP (PLUS)CAP_BOT (MINUS)
```

### See Also

- [ICV\\_LVS\\_DEVICE\\_TYPE\\_CAP](#)
- [ICV\\_LVS\\_DEVICE\\_TYPE\\_MOS](#)
- [ICV\\_LVS\\_DEVICE\\_TYPE\\_RES](#)

---

## ICV\_RUNSET\_REPORT\_FILE

Specifies the IC Validator runset report file and activates the IC Validator flow.

### Syntax

`ICV_RUNSET_REPORT_FILE: file_name`

### Arguments

| Argument               | Description                                                        |
|------------------------|--------------------------------------------------------------------|
| <code>file_name</code> | IC Validator runset report file name<br>Default: pex_runset_report |

### Description

The IC Validator runset report file contains information that the IC Validator tool provides to the StarRC tool for RC extraction, such as connection information, the location of LVS COMPARE output, device pin and properties information, and location of the extract view.

When the `ICV_RUNSET_REPORT_FILE` command is used, the `MILKYWAY_EXTRACT_VIEW` command is set to YES. In addition, the `MILKYWAY_DATABASE`, `MAPPING_FILE`, `COMPARE_DIRECTORY`, and `OA_LAYER_MAPPING_FILE` commands become optional. If these optional commands are specified in the StarRC command file, then the values in the StarRC command file override the values in the IC Validator runset report file.

StarRC automatically reads different formats of IC Validator data. The IC Validator tool provides information about the data format in the runset report file.

### Example

`ICV_RUNSET_REPORT_FILE: my_icv_rrf`

### See Also

- [COMPARE\\_DIRECTORY](#)
- [MAPPING\\_FILE](#)
- [MILKYWAY\\_DATABASE](#)
- [MILKYWAY\\_EXTRACT\\_VIEW](#)
- [OA\\_LAYER\\_MAPPING\\_FILE](#)

---

## **IGNORE\_CAPACITANCE**

Prevents certain types of device-level capacitances from being extracted and netlisted.

### **Syntax**

`IGNORE_CAPACITANCE: ALL [RETAIN_GATE_DIFFUSION_COUPLING] | DIFF | NONE`

### **Arguments**

| Argument                       | Description                                                                                                                                                                                 |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALL (default)                  | Ignores capacitance between diffusion regions and the substrate, as well as between the transistor gates and diffusion regions or substrate. Both overlap and sidewall effects are ignored. |
| RETAIN_GATE_DIFFUSION_COUPLING | Retains gate-to-diffusion coupling capacitance                                                                                                                                              |
| DIFF                           | Ignores only the junction capacitance. The gate capacitance is included.                                                                                                                    |
| NONE                           | Includes all parasitic capacitance                                                                                                                                                          |

### **Description**

The `IGNORE_CAPACITANCE` command prevents certain types of device-level capacitances from being extracted and netlisted. This is desirable to prevent double counting when the primitive device models already account for those capacitances.

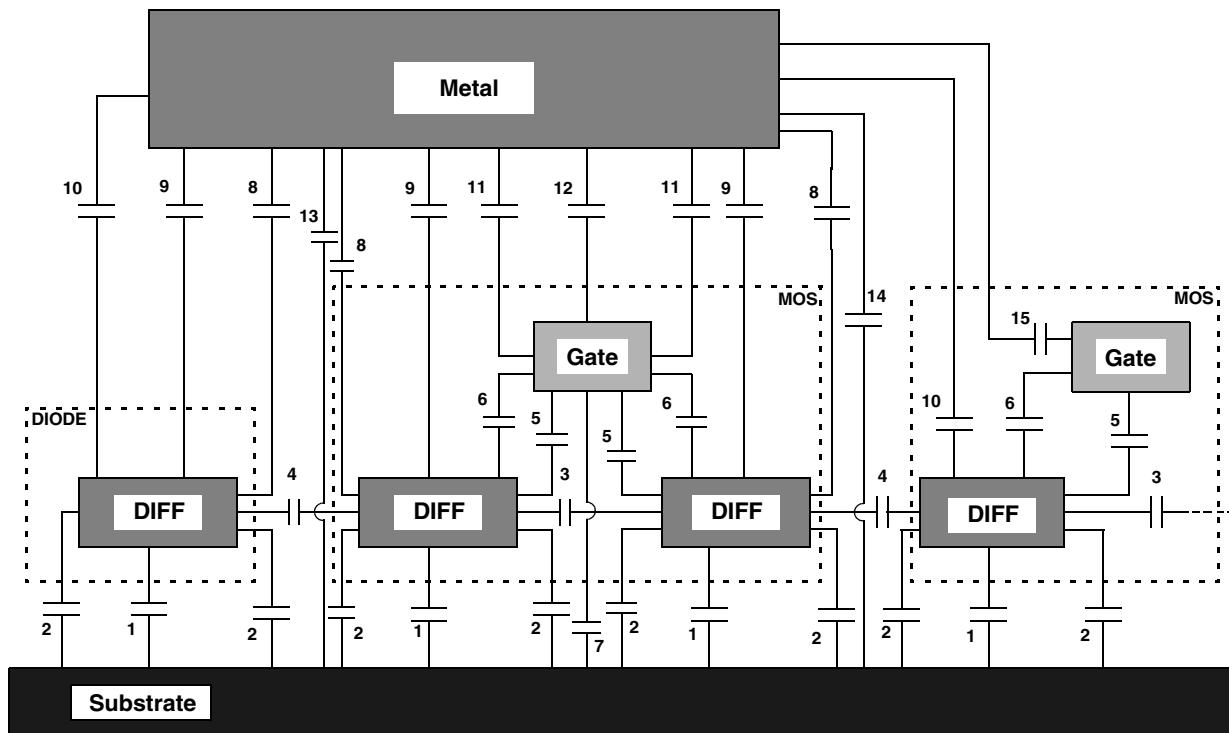
#### **Note:**

`IGNORE_CAPACITANCE` is a device-level command and affects only parasitic output related to transistor devices.

The `IGNORE_CAPACITANCE` command applies to capacitive effects internal to each individual device in the runset. For the highest accuracy, `IGNORE_CAPACITANCE: ALL | DIFF` does not exclude capacitive effects between a device and its neighbors because these interdevice effects are not accounted for in the device model.

[Figure 15-12](#) shows the capacitances within a transistor and between transistors.

Figure 15-12 Capacitance Components of the IGNORE\_CAPACITANCE Command



*Table 15-2 Capacitances Extracted for Each Argument*

| Argument | FinFET devices with MULTIGATE_MODELS: YES | FinFET devices without MULTIGATE_MODELS: YES | Other devices |
|----------|-------------------------------------------|----------------------------------------------|---------------|
| ALL      | 4; 8 to 15                                | 4; 8 to 15                                   | 4; 8 to 15    |
| DIFF     | 4; 8 to 15                                | 4; 8 to 15                                   | 4 to 15       |
| NONE     | 4; 8 to 15                                | 1 to 4; 8 to 15                              | 1 to 15       |

The StarRC field solver does not extract capacitances 1 to 3 and 5 to 7 for FinFET devices during FinFET process characterization. Therefore, the DIFF and NONE arguments are the same for these devices.

FinFET models that do not use the MULTIGATE\_MODELS: YES statement already account for capacitances 1 to 3. For these models, using DIFF instead of NONE prevents double counting of those capacitances.

The following example contains two MOS devices:

```
NMOS N ngate nsd nsd SUBSTRATE { } TEMP=ndev_out
PMOS P pgate psd psd NWELL { } TEMP=pdev_out
```

**Table 15-3** lists the command interactions for the previous example.

*Table 15-3 Command Interactions for IGNORE\_CAPACITANCE: ALL*

| StarRC ignores the following interactions<br>for IGNORE_CAPACITANCE: ALL | StarRC retains the following interactions<br>for IGNORE_CAPACITANCE: ALL |
|--------------------------------------------------------------------------|--------------------------------------------------------------------------|
| ngate-SUBSTRATE                                                          | ngate-pgate                                                              |
| nsd-nsd                                                                  | nsd-psd                                                                  |
| ngate-nsd                                                                | ngate-psd                                                                |
| nsd-SUBSTRATE                                                            | pgate-nsd                                                                |
| pgate-NWELL                                                              | ngate-NWELL                                                              |
| psd-psd                                                                  | nsd-NWELL                                                                |
| pgate-psd                                                                |                                                                          |
| psd-NWELL                                                                |                                                                          |

This means that interdevice capacitive effects that are not accounted for in the device model are included in the parasitic netlist. For IGNORE\_CAPACITANCE to be most effective and accurate, it is very important that device layers in the runset be separated from other connected layers that conflict with them (e.g. in the CONNECT sequence of the runset).

For example, if nsd and psd are derived from input layers DIFF, PPLUS and NPLUS, and DIFF is also a final CONNECT layer, the following Boolean sequence is required:

- BOOLEAN DIFF not nsd { } TEMP=DIFF
- BOOLEAN DIFF not psd { } TEMP=DIFF

This is critical because the DIFF layer is not used in a device and therefore it is not identifiable as a diffusion layer. Therefore, its parasitic contribution to both N and P devices cannot be ignored by StarRC. In other words, DIFF is not an N or P device layer, so it must be part of the unmodeled environment.

When the calculation is done for netlist reduction to reduce the nodes with error control, small coupling capacitors are moved around their individual nodes to attach to one of the neighboring nodes. In such a situation, some device terminal nodes get coupling capacitances even if the coupling capacitance is not associated with them irrespective of the IGNORE\_CAPACITANCE setting.

## Retaining Gate-to-Diffusion Coupling Capacitance

To retain the gate-to-diffusion (C<sub>f</sub>) coupling component when `IGNORE_CAPACITANCE:ALL` is specified, use the `RETAIN_GATE_DIFFUSION_COUPLING` suffix:

```
IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING
```

When this suffix is specified, StarRC has two methods for extracting the gate-to-diffusion component:

- Based on precharacterized models, similar to other capacitances extracted by StarRC
- Based on a 2-D capacitance table lookup dependent on layout parameters

## FinFET Capacitance

[Table 15-4](#) lists the FinFET capacitance components retained when you use the `IGNORE_CAPACITANCE` command with `MULTIGATE` devices. Note that diffusion-to-substrate capacitance is not extracted for `MULTIGATE` devices, so the `DIFF` keyword is equivalent to `NONE`.

*Table 15-4 FinFET Capacitance Components Retained*

| IGNORE_CAPACITANCE Value           | FinFET Capacitance Components Retained                                                    |
|------------------------------------|-------------------------------------------------------------------------------------------|
| ALL (default)                      | C <sub>gc</sub> + C <sub>fpe</sub> + C <sub>fpc</sub>                                     |
| ALL RETAIN_GATE_DIFFUSION_COUPLING | C <sub>gc</sub> + C <sub>fpe</sub> + C <sub>fpc</sub> + C <sub>gd</sub>                   |
| NONE or DIFF                       | C <sub>gc</sub> + C <sub>fpe</sub> + C <sub>fpc</sub> + C <sub>gd</sub> + C <sub>g0</sub> |

## **IGNORE\_FIELDPOLY\_DIFFUSION\_COUPLING**

Extracts the field poly to diffusion coupling.

### **Syntax**

IGNORE\_FIELDPOLY\_DIFFUSION\_COUPLING: YES | NO

### **Arguments**

| Argument     | Description                                 |
|--------------|---------------------------------------------|
| NO (default) | Retains field poly to diffusion capacitance |
| YES          | Excludes field poly coupling capacitance    |

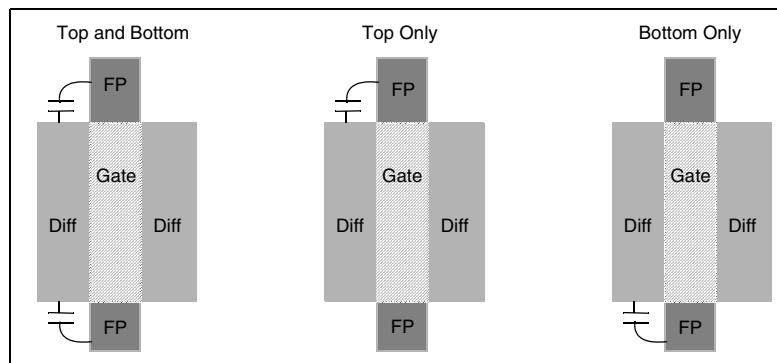
### **Description**

This command extracts the field poly to diffusion coupling. Although the capacitance effect could be small from orthogonal coupling capacitance as shown in [Figure 15-13](#), at lower technology nodes it might have a capacitance impact. Given the large number of field poly connections in a design, this can have a large impact on the circuit performance.

This command allows you the flexibility to ignore fieldpoly diffusion coupling when this coupling effect is already included in the SPICE model.

This command supports all transistor-level flows using Hercules, IC Validator, or Calibre.

*Figure 15-13 Field Poly to Diffusion Ignored by Default*



### **See Also**

- [IGNORE\\_CAPACITANCE](#)

---

## IGNORE\_GATE\_CHANNEL\_CAPACITANCE

Specifies the method of calculating gate-to-diffusion capacitance for FinFET devices.

### Syntax

IGNORE\_GATE\_CHANNEL\_CAPACITANCE: YES | NO

### Arguments

| Argument     | Description                                                                         |
|--------------|-------------------------------------------------------------------------------------|
| NO (default) | Disables use of Cfi table to calculate Cfo                                          |
| YES          | Enables calculation of Cfo by subtracting Cfi table values from extracted Cf values |

### Description

The total gate-to-diffusion capacitance (Cf) is the sum of the gate-to-diffusion capacitance inside the channel (Cfi) and the gate-to-diffusion capacitance outside the channel (Cfo).

You can provide a table to represent gate-to-diffusion capacitance inside the channel (Cfi) by using the ITF `GATE_TO_DIFFUSION_CHANNEL_CAP` statement.

When the `IGNORE_GATE_CHANNEL_CAPACITANCE: NO` command is used, the total gate-to-diffusion capacitance (Cf) is extracted by the field solver and no Cfi capacitance subtraction occurs, regardless of whether a Cfi table exists.

Alternatively, you can calculate Cfo by first having the field solver extract the total capacitance Cf, then subtracting the Cfi value (interpolated from the Cfi table) from Cf. To enable this calculation method, use the `IGNORE_GATE_CHANNEL_CAPACITANCE: YES` command. If a Cfi table does not exist, the command has no effect.

The `IGNORE_GATE_CHANNEL_CAPACITANCE` command requires nxtgrd files generated by StarRC version H-2013.06-SP1 or later.

### See Also

- [GATE\\_TO\\_DIFFUSION\\_CHANNEL\\_CAP](#)
- [IGNORE\\_CAPACITANCE](#)

---

## IGNORE\_SHARED\_MOS\_TERMINAL\_CAP

Ignores capacitance on shared terminals.

### Syntax

IGNORE\_SHARED\_MOS\_TERMINAL\_CAP: YES | NO

### Arguments

| Argument     | Description                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Ignores capacitance on a terminal layer shared by a MOS device terminal and a capacitor device terminal or on a terminal layer shared by a MOS device terminal and the resistance body of a resistor device  |
| NO (default) | Extracts capacitance on a terminal layer shared by a MOS device terminal and a capacitor device terminal or on a terminal layer shared by a MOS device terminal and the resistance body of a resistor device |

### Description

By default, StarRC extracts the capacitance between the following layers if a gate, diffusion, or bulk terminal is used as a capacitor terminal:

- The substrate and the gate, diffusion, and bulk layers
- The bulk layers and the gate and diffusion layers

To ignore this capacitance, specify IGNORE\_SHARED\_MOS\_TERMINAL\_CAP: YES.

### Example

In the following example, StarRC ignores the capacitance on shared terminals.

```
IGNORE_SHARED_MOS_TERMINAL_CAP: YES
```

### See Also

- [IGNORE\\_CAPACITANCE](#)

---

## INSTANCE\_PORT

Applies different parasitic resistance extraction modes to different groups of instance ports in a single extraction run.

### Syntax

```
INSTANCE_PORT: CONDUCTIVE [SUFFIXED | SUFFIXED MULTIPLE | CAPACITIVE]
| NOT_CONDUCTIVE | SUPERCONDUCTIVE | EXPLODE
[CELL cell_name] [INST inst_name]
[PORt port_name]
```

### Arguments

| Argument              | Description                                                                                                                                                                                                                                                                  |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONDUCTIVE (default)  | Enables resistance extraction for the ports of skip cell instances. Therefore, feedthrough resistance of selected instance ports are preserved.<br>The supplemental modes available are SUFFIXED or CAPACITIVE. The SUFFIXED mode can be specified with or without MULTIPLE. |
| NOT_CONDUCTIVE        | Similar to the CONDUCTIVE MULTIPLE SUFFIXED option, with the exception of the port resistance not being netlisted. If the top-level material overlaps with the instance port, the conductance of the overlapping part of the top-level material is not reported, either.     |
| SUPERCONDUCTIVE       | When specified for a SKIP_CELLS port, the port is extracted as an ideal node with zero resistivity.                                                                                                                                                                          |
| EXPLODE               | Promotes all selected instance port material to the top level, and marks it as part of the top-level net connecting it. No port nodes are generated for instance ports selected for EXPLODE extraction.                                                                      |
| CELL <i>cell_name</i> | Layout cell name.                                                                                                                                                                                                                                                            |
| INST <i>inst_name</i> | Layout instance name.                                                                                                                                                                                                                                                        |
| PORt <i>port_name</i> | Layout port name.                                                                                                                                                                                                                                                            |

### Description

The INSTANCE\_PORT command applies different parasitic resistance extraction modes to different groups of instance ports in a single extraction run. You can specify the INSTANCE\_PORT command multiple times. Each use is cumulative, but if any instance ports match more than one command, the last definition overrides any previous ones.

You can apply the `INSTANCE_PORT` command to specific cells, instances, or ports. Specify only layout names in these arguments, not schematic names. The default for all three options is the asterisk wildcard (\*).

For the `CONDUCTIVE MULTIPLE` and `NOT_CONDUCTIVE` modes, the number of port nodes is determined by the top-level resistive interaction regions. For the `CONDUCTIVE` mode, only one port node is created.

### **CONDUCTIVE Option**

By default, one port (\*|I) node per instance port is generated and no capacitance is reported for the port. The location of the node is a point of contact between the port and the top-level material. This default mode should be sufficient for most applications.

There are three supplemental modes that can be used with `INSTANCE_PORT: CONDUCTIVE`. They are `SUFFIXED`, `MULTIPLE`, and `CAPACITIVE`.

The `CAPACITIVE` option netlists the capacitance of the selected instance ports.

The `SUFFIXED` option modifies the port node instance name with a suffix according to the `NETLIST_RENAME_PORTS` command. Its effect depends on whether you also use the `MULTIPLE` option. Use of the `MULTIPLE` option without the `SUFFIXED` option is not supported.

- If the `MULTIPLE` option is not used, exactly one connection point is generated, regardless of the number of interactions between the port and the top-level material. In the case of no interactions, the location of the node is random within the port. In the case of multiple connections to the top-level material, only one is reported as a port node.
- If the `MULTIPLE` option is used, then all of the connection points are reported as port nodes. In the case of no interactions, no port nodes are generated. In the case of a large overlap with the top-level material, multiple connection points are reported.

### **NOT\_CONDUCTIVE Option**

If the port resistors are not included in the netlist, there might be disconnected networks. However, no open warnings are issued, because the net is known to be connected by feedthrough. Port nodes for a given instance port are generated at every top-level interaction point. If there is more than one interaction point, multiple port nodes are netlisted. In the case of no interaction with top-level material, no port node is generated.

### **SUPERCONDUCTIVE Option**

StarRC assumes ideal (zero) resistivity when extracting port shapes inside the skip cell. The port shapes act as shorts and no resistance is extracted or netlisted for the port shapes.

## EXPLODE Option

StarRC promotes a selected instance port to the top level and marks it as part of the top-level net connecting it. No port nodes are generated for instance ports selected for EXPLODE extraction and no \*I or \*|I statements are generated.

### Examples

- To select all instance ports as CONDUCTIVE:

```
INSTANCE_PORT: CONDUCTIVE
```

- To select all ports in cell ANTENNA for EXPLODE (to the top level), leaving all other instance ports set to CONDUCTIVE:

```
INSTANCE_PORT: EXPLODE CELL ANTENNA
```

- To select ports VDD and VSS in all cells to be netlisted with suffixes and multiple times if there is more than one connection point:

```
INSTANCE_PORT: CONDUCTIVE MULTIPLE SUFFIXED CELL * PORT VDD VSS
```

This causes ports VSS and VDD in ANTENNA to be retained, but all other ports in ANTENNA are exploded.

- To select all instance ports as CONDUCTIVE SUFFIXED, except for instance ports with CELL names matching A\*, which are set to NOT\_CONDUCTIVE:

```
INSTANCE_PORT: CONDUCTIVE SUFFIXED
INSTANCE_PORT: NOT_CONDUCTIVE CELL A*
```

- To set all instance ports matching CELL B\* (but not AB\*) PORT VDD\* to CONDUCTIVE

```
INSTANCE_PORT: NOT_CONDUCTIVE
INSTANCE_PORT: CONDUCTIVE SUFFIXED CELL B* !AB* PORT VDD*
```

### See Also

- [NETLIST\\_RENAME\\_PORTS](#)
- [SKIP\\_CELLS](#)

---

## INSTANCE\_PORT\_OPEN\_CONDUCTANCE

Defines a fixed conductance value for the resistors used to connect members of a port that are not resistively connected inside of the defined skip cell.

### Syntax

INSTANCE\_PORT\_OPEN\_CONDUCTANCE: *cond\_value*

### Arguments

| Argument          | Description                                                                                                                                                                              |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>cond_value</i> | A fixed conductance value (floating point value) for resistors used to connect members of a port that are not electrically connected inside the skip cell<br>Units: mho<br>Default: 10.0 |

### Description

This command defines a fixed conductance value for the resistors used to connect members of a port that are not resistively connected inside of the defined skip cell. This command affects only CONDUCTIVE instance ports.

This case can happen frequently when the instance port material is not completely ported. Often, small routing targets at the edges, instead of the entire port, are used as the port definition. In this case, StarRC inserts resistors with user-defined conductance to prevent opens in the output netlist. If you are also using the NETLIST\_TAIL\_COMMENTS command, the resistor width is reported as nine microns to facilitate their identification.

---

## **INTRANET\_CAPS**

Specifies whether to extract intranet capacitances.

### **Syntax**

INTRANET\_CAPS: YES | NO

### **Arguments**

| Argument     | Description                                            |
|--------------|--------------------------------------------------------|
| YES          | Retains the intranet capacitances during extraction    |
| NO (default) | Eliminates the intranet capacitances during extraction |

### **Description**

This command specifies whether to extract intranet capacitances, which are coupling capacitances that have nodes that belong to the same net.

---

## ITF\_FILE

Specifies a file containing ITF commands for the direct ITF flow. Valid only for transistor-level flows.

### Syntax

ITF\_FILE: *file\_name*

### Arguments

| Argument         | Description                        |
|------------------|------------------------------------|
| <i>file_name</i> | The ITF file name<br>Default: none |

### Description

This command allows you to experiment with process changes by specifying an ITF file directly in the StarRC command file.

The direct ITF flow does not use the grdgexo program and does not generate or use an nxtgrd file. This command is supported only for capacitance extraction in transistor-level flows.

### See Also

- [The Direct ITF Flow](#)

---

## KEEP\_SUBCONT\_MODELS

Specifies a list of models for which to keep all substrate contacts. Valid only for transistor-level flows.

### Syntax

KEEP\_SUBCONT\_MODELS: *model\_list*

### Arguments

| Argument          | Description               |
|-------------------|---------------------------|
| <i>model_list</i> | Model list<br>Default: !* |

### Description

The KEEP\_SUBCONT\_MODELS command allows you to specify how the StarRC tool handles substrate contacts during transistor-level extraction. This command has an effect only when the TRANSLATE\_RETAIN\_BULK\_LAYERS command is set to CONLY.

By default, the tool retains only one substrate contact (subCont instance) for each substrate polygon. However, you can keep all substrate contacts for specific features by including the KEEP\_SUBCONT\_MODELS command and specifying the models of interest.

### See Also

- [TRANSLATE\\_RETAIN\\_BULK\\_LAYERS](#)

---

## KEEP\_VIA\_NODES

Defines via nodes as the original nodes that a via resistor connects.

### Syntax

KEEP\_VIA\_NODES: YES | NO | MAPPING\_FILE

### Arguments

| Argument     | Description                                            |
|--------------|--------------------------------------------------------|
| YES          | Preserves original nodes that a via resistor connects  |
| NO (default) | Does not necessarily preserve original nodes           |
| MAPPING_FILE | Follows via reduction instructions in the mapping file |

### Description

The `KEEP_VIA_NODES` command defines via nodes as the original nodes that a via resistor connects. The original nodes might be reduced or merged with other nodes, depending on the conductor configuration and reduction mode. Setting the `KEEP_VIA_NODES` command to `YES` preserves such nodes.

You can specify how reduction is applied to vias by setting the `KEEP_VIA_NODES` command to `MAPPING_FILE` and including the `REDUCE_VIA` option in the `via_layers` section in the mapping file. The reduction options are as follows:

- `yes` (reduce vias)
- `no` (do not reduce vias)
- `power` (reduce vias only if they belong to power nets)
- `signal` (reduce vias only if they belong to signal nets)

### See Also

- [REDUCTION](#)

---

## **LEF\_FILE**

Creates a white-space-delimited list of LEF format files containing complete library cell descriptions.

### **Syntax**

```
LEF_FILE: technology_lef library_lef
LEF_FILE: library_lef macro_lef
LEF_FILE: macro_lef custom_lef
```

### **Arguments**

| <b>Argument</b>       | <b>Description</b>                                                   |
|-----------------------|----------------------------------------------------------------------|
| <i>technology_lef</i> | LEF file with the technology information<br>Default: none            |
| <i>library_lef</i>    | LEF file with the cell library information<br>Default: none          |
| <i>macro_lef</i>      | LEF file with the core cell information<br>Default: none             |
| <i>custom_lef</i>     | LEF file with the custom cell and block information<br>Default: none |

### **Description**

This command, which is mandatory for LEF/DEF flows, can be specified multiple times in a single command file. The order in which the LEF files are specified is very important.

There are three types of LEF files: the technology LEF, the standard cell LEF, and the macro LEF. The technology LEF must be listed first in the command file or in the graphic user interface. Every layer defined in the technology LEF must be mapped to an nxtgrd file layer by a mapping file entry. Undefined layers that exist in the LEF file cause StarRC to issue an error and exit. If any of the subsequently specified LEF files contain additional technology information, it is ignored.

The standard cell LEF and the macro LEF can be listed in any order after the technology LEF. You do not need to provide LEF descriptions for DEF macros specified in a MACRO\_DEF\_FILE command. Either a LEF or DEF description is required for every member of the list specified by the SKIP\_CELLS command.

## See Also

- [GDS\\_LAYER\\_MAP\\_FILE](#)
- [MACRO\\_DEF\\_FILE](#)
- [MAPPING\\_FILE](#)
- [SKIP\\_CELLS](#)

---

## LEF\_USE\_OBS

Enables the use of OBS shapes in the LEF MACRO.

### Syntax

LEF\_USE\_OBS: YES | NO

### Arguments

| Argument      | Description                                                                                            |
|---------------|--------------------------------------------------------------------------------------------------------|
| YES (default) | Includes the OBS shapes defined in the LEF MACRO and allows StarRC to extract coupling to these shapes |
| NO            | Uses only the PIN shapes translated from the LEF MACRO                                                 |

### Description

When NO is set, removes all MACRO blockage material from the extraction. LEF\_USE\_OBS applies to all SKIP\_CELLS commands in your design.

Most LEF MACRO definitions contain groups of shapes under the heading OBS. These are blockage layers that restrict the top-level router and typically are a close approximation of the actual cell layout.

This command has no effect on MACRO definitions for which supplemental GDS descriptions have been provided. A GDS description for LEF MACRO is optional and can be imported with the GDS\_FILE command.

### See Also

- [GDS\\_FILE](#)
- [SKIP\\_CELLS](#)

---

## LPE\_DEVICES

Specifies the device model names that follow the LPE\_PARAM rules.

### Syntax

LPE\_DEVICES: *param\_name* *device\_model\_1* [*device\_model\_2* ...]

### Arguments

| Argument            | Description                                                                                                                          |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <i>param_name</i>   | A user-defined LPE parameter name. The specified parameter name should match the parameter names specified by the LPE_PARAM command. |
| <i>device_model</i> | Device models to which the LPE parameter should be applied                                                                           |

### Description

The LPE\_DEVICES specifies the device model names that follow the LPE\_PARAM rules. This command must be used in conjunction with the LPE\_PARAM command.

If the list of device models is very long, you can use multiple LPE\_DEVICES commands for the same parameter. The lists of device models are concatenated.

### Example

```
LPE_DEVICES: pre_layout nfet pfet
LPE_DEVICES: nores ncap
```

### See Also

- [LPE\\_PARAM](#)
- [LPE\\_FLAGS\\_SETTING](#)

---

## LPE\_FLAGS\_SETTING

Specifies how to apply parameters specified in the `LPE_PARAM` command.

### Syntax

`LPE_FLAGS_SETTING: NETLIST | EXTRACTION`

### Arguments

| Argument          | Description                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------|
| NETLIST (default) | Applies parameters based on the extraction status of nets connected to terminals of user-defined instances as in the final netlist |
| EXTRACTION        | Applies parameters on a global level based on the <code>EXTRACTION</code> command setting                                          |

### Description

The `LPE_DEVICES` and `LPE_PARAM` commands allow you to control which LPE parameters are used for user-defined instances. Specifying an appropriate LPE parameter for each user-defined instance results in the best accuracy for overall simulation. The default setting `NETLIST` for the `LPE_FLAGS_SETTING` command assumes this approach.

Alternatively, you can simplify the application of LPE parameters by setting the `LPE_FLAGS_SETTING` command to `EXTRACTION`. In this case, the extraction mode set in the `EXTRACTION` command (`R`, `C`, or `RC`) is used for all LPE parameters of user-defined instances. This setting tends to produce pessimistic results.

### Example

In this example, even though the `LPE_PARAM` command contains flags for all modes for device type `nfet`, the `LPE_FLAGS_SETTING` command forces the use of the capacitance-mode flag (as set in the `EXTRACTION` command) for all user-defined instances.

```
EXTRACTION: C
LPE_DEVICES: pre_layout nfet pfet
LPE_FLAGS_SETTING: EXTRACTION
LPE_PARAM: nfet NORC 1 C 3 R 2 RC 0 PINEXCEPT 4
```

### See Also

- [LPE\\_PARAM](#)
- [LPE\\_DEVICES](#)
- [EXTRACTION](#)

---

## LPE\_PARAM

Defines a netlist parameter for user-defined instances.

### Syntax

```
LPE_PARAM: param_name mode1 value1 [mode2 value2...]
 [PINEXCEPT pinIds]
 [PINOPERATION AND|OR]
```

### Arguments

| Argument                                  | Description                                                                                                                                                                                            |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>param_name</i>                         | The name of the LPE parameter                                                                                                                                                                          |
| <i>mode</i>                               | The extraction mode used to netlist the device. Valid modes are R, C, RC, and NORC.                                                                                                                    |
| <i>value</i>                              | Corresponds to flags in the device simulation SPICE model. The value can be customized based on flows.                                                                                                 |
| PINEXCEPT <i>pin_IDs</i>                  | The list of pin indexes to be ignored while computing the extraction mode for the device. Pin indexes start at 0 for the first pin of a device.                                                        |
| PINOPERATION AND   OR<br>(default is AND) | The method used to determine the value of the gate-to-contact coupling LPE parameter. When you set PINOPERATION to OR, the tool uses an OR operation to evaluate the extraction mode of the terminals. |

### Description

The `LPE_PARAM` command defines a netlist parameter for user-defined instances. This post-layout parameter is set based on the extraction mode of the nets connected to the user-defined instance. You can optionally use the `LPE_FLAGS_SETTING` and `EXTRACTION` commands to specify one extraction mode to use for all instances.

The command can be used to control, based on a user-defined parameter and value, which parasitics are taken from simulation SPICE models and which parasitics are extracted by the StarRC tool.

This command must be used in conjunction with the `LPE_DEVICES` command.

The `PINEXCEPT` option causes the list of pin indexes to be ignored while computing the extraction mode for the device. For example, it could be used to ignore the bulk in a MOS

device (PINEXCEPT 3), so that only the net connected to drain, source and gate would be taken into account for computing an LPE parameter value.

There can be only one `LPE_PARAM` definition for each parameter. If multiple `LPE_PARAM` definitions exist for a single parameter, then the last definition overrides the previous definitions.

For completeness, all extraction modes should be specified in the `LPE_PARAM` command. However, if the extraction mode is not specified in the `LPE_PARAM` command, the tool tries to compute the value.

If RC is not described for a parameter, then capacitance extraction has no impact on its value. If capacitance extraction is performed, the tool must use the NOR value, which is the default.

### Example

The following defines three parameters for a user-defined instance along with the extraction mode and the corresponding flag setting in the SPICE model.

```
LPE_PARAM: pre_layout_local NORC 1 C 3 R 2 RC 0 PINEXCEPT 4
LPE_PARAM: setres NORC -2 C -2 R 0 RC 0
LPE_PARAM: setcap NORC -1 C 0 R -1 RC 0
```

The following table lists four user-defined instances and the extraction mode setting for pin A and pin B of each instance. The table shows the value of the LPE parameter when `PINOPERATION` is set to an OR operation.

|   | Pin A | Pin B | AND Operation | OR Operation |
|---|-------|-------|---------------|--------------|
| 1 | RC    | RC    | RC            | RC           |
| 2 | NO RC | RC    | NO RC         | RC           |
| 3 | RC    | NO RC | NO RC         | RC           |
| 4 | NO RC | NO RC | NO RC         | NO RC        |

### See Also

- [LPE\\_DEVICES](#)
- [LPE\\_FLAGS\\_SETTING](#)

---

## LVS\_EXTRACTION\_REPORT\_FILE

Specifies the Calibre

### Syntax

LVS\_EXTRACTION\_REPORT\_FILE: *lvs\_rpt*

### Arguments

| Argument       | Description                                          |
|----------------|------------------------------------------------------|
| <i>lvs_rpt</i> | Calibre extraction report file name<br>Default: none |

### Description

If the Calibre query file uses the LVS SETTINGS REPORT WRITE command to write an extraction report, the StarRC CALIBRE\_QUERY\_FILE command automatically obtains the runset file and PDBA information from the query file. The CALIBRE\_RUNSET and CALIBRE\_PDBA\_FILE commands are invalid in this case.

Instead of reading the automatically determined extraction report, you can specify a different Calibre extraction report by using the LVS\_EXTRACTION\_REPORT\_FILE command in the StarRC command file. The file specified by the StarRC LVS\_EXTRACTION\_REPORT\_FILE command takes precedence over the file named in the Calibre LVS SETTINGS REPORT WRITE command.

### Example

LVS\_EXTRACTION\_REPORT\_FILE: cci\_rpt

### See Also

- [CALIBRE\\_PDBA\\_FILE](#)
- [CALIBRE\\_QUERY\\_FILE](#)
- [CALIBRE\\_RUNSET](#)
- [MILKYWAY\\_DATABASE](#)

---

## MACRO\_DEF\_FILE

Creates a white-space-delimited list of DEF files that define any macros referenced in the file specified by the `TOP_DEF_FILE` command.

### Syntax

`MACRO_DEF_FILE: macro_def_file1 macro_def_file2 ...`

### Arguments

| Argument                    | Description                                                              |
|-----------------------------|--------------------------------------------------------------------------|
| <code>macro_def_file</code> | File that contains any macro referenced in the <code>TOP_DEF_FILE</code> |

### Description

This command can be specified multiple times in a single command file. The order of the files in the list is not important. There is no need to provide LEF descriptions with the LEF file for macros on this list because the DEF descriptions are used instead.

The cells described within a file specified in a `MACRO_DEF_FILE` command might be on the list of skip cells included in the file specified by the `SKIP_CELLS` command. For example, you might go into these macro cells to produce a full-chip flat netlist, or you might skip them if you are doing hierarchical extraction or analysis. StarRC does not require any preprocessing to flatten a hierarchical DEF file; the hierarchical DEF is flattened internally.

You can specify gzip or compressed DEF files with the `MACRO_DEF_FILE` command.

### See Also

- [LEF\\_FILE](#)
- [SKIP\\_CELLS](#)
- [TOP\\_DEF\\_FILE](#)

---

## MAGNIFICATION\_FACTOR

Scales input data uniformly for all layers.

### Syntax

`MAGNIFICATION_FACTOR: value`

### Arguments

| Argument           | Description                                                      |
|--------------------|------------------------------------------------------------------|
| <code>value</code> | Scaling factor; a positive floating-point number<br>Default: 1.0 |

### Description

The `MAGNIFICATION_FACTOR` command performs scaling of layout dimensions for all layers. Scaling does not affect the connectivity of the layout network.

Specifying a value less than 1 indicates a shrink; a value greater than 1 indicates expansion. The scaling operation is performed on a database only while reading it.

The nxtgrd file must contain rules for minimum width, spacing, and tables associated with the shrunk or enlarged database.

You cannot use the `MAGNIFICATION_FACTOR` command with the `HALF_NODE_SCALE_FACTOR` ITF command.

### See Also

- [HALF\\_NODE\\_SCALE\\_FACTOR](#)
- [MAGNIFY\\_DEVICE\\_PARAMS](#)

---

## MAGNIFY\_DEVICE\_PARAMS

Controls the behavior of the `MAGNIFICATION_FACTOR` command.

### Syntax

`MAGNIFY_DEVICE_PARAMS: YES | NO`

### Arguments

| Argument      | Description                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------|
| YES (default) | Applies the value specified in the <code>MAGNIFICATION_FACTOR</code> command to SPICE device parameters |
| NO            | Does not apply the value of the <code>MAGNIFICATION_FACTOR</code> to the SPICE parameters               |

### Description

The `MAGNIFY_DEVICE_PARAMS` command controls the behavior of the `MAGNIFICATION_FACTOR` command. When you specify `MAGNIFY_DEVICE_PARAMS: YES`, all designed device parameters in the SPICE netlist are multiplied by the factor set by the `MAGNIFICATION_FACTOR` command. [Table 15-5](#) lists the parameters affected by the `MAGNIFY_DEVICE_PARAMS` command.

*Table 15-5 Device Parameters Affected By MAGNIFY\_DEVICE\_PARAMS*

| Device type | Magnified parameters             |
|-------------|----------------------------------|
| Diode       | area, pj                         |
| Resistor    | l, w                             |
| Capacitor   | l, w                             |
| MOS         | ad, as, pd, l, w, sa ,sb, and sc |
| BJT         | area                             |

### See Also

- [MAGNIFICATION\\_FACTOR](#)

---

## MAPPING\_FILE

Specifies the file containing physical layer mapping information between the input database and the specified nxtgrd file.

### Syntax

`MAPPING_FILE: file_name`

### Arguments

| Argument                      | Description                            |
|-------------------------------|----------------------------------------|
| <code><i>file_name</i></code> | The mapping file name<br>Default: none |

### Description

This command is required for all flows. Every connected layer must be mapped.

You can elect to override sheet resistance values specified in the nxtgrd file by specifying new values in the mapping file specified by the `MAPPING_FILE` command.

For simultaneous multicorners flows, you can specify a mapping file for each corner by including mapping file commands in the corner definitions in the corners file. If a corner definition does not include a `MAPPING_FILE` command, a global mapping file must be defined in the top-level command file. If every corner includes a mapping file, a global mapping file is not necessary; if one exists, it is ignored.

All mapping files for simultaneous multicorners flows must be consistent in terms of the number of database and ITF file layers and their mapping relationships. Each mapping file category, such as conductors or vias, should also be consistent. The only allowed variations are the RPSQ value for conductors and the RPV and AREA values for vias.

For more information about mapping files, see [Chapter 17, “Mapping File Commands.”](#)

### See Also

- [TCAD\\_GRD\\_FILE](#)
- [CORNERS\\_FILE](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)

---

## **MARKER\_GENERATION**

Specifies how to generate pin shapes for a database.

### **Syntax**

`MARKER_GENERATION: AUTOMATIC | USER`

### **Arguments**

| Argument            | Description                                |
|---------------------|--------------------------------------------|
| AUTOMATIC (default) | Generates pin shapes based on connectivity |
| USER                | User generates the pin shapes              |

### **Description**

The StarRC tool requires pin shapes to define the instance ports. Pin shapes are also referred to as markers.

There are two ways to create a pin shape. The first is to identify a special output layer that generates the pin shape, and the second is to have StarRC automatically generate pin shapes, based on connectivity.

For most purposes, the `AUTOMATIC` setting is preferred because it is more robust. The `USER` setting allows more flexibility but requires great care when creating marker shapes and a rigorous knowledge of the routing methodology to avoid creating opens and shorts in the extraction.

When you specify the `MARKER_GENERATION:USER` command, you must

- Use one of the following techniques to generate a pin shape: text-based markers, ID-layer markers, or connection-based markers.
- Specify marker layers in the mapping file with the `marker_layers` statement.

### **See Also**

- [MAPPING\\_FILE](#)

---

## MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER

Specifies the maximum number of virtual via segments in a trench contact process.

### Syntax

MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER: *number\_of\_segments*

### Arguments

| Argument                  | Description                                                                   |
|---------------------------|-------------------------------------------------------------------------------|
| <i>number_of_segments</i> | Maximum number of via segments; an integer<br>Units: none<br>Default: 1000000 |

### Description

The MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER command specifies the maximum number of virtual via segments in a trench contact process.

### Errors

You can use the TRENCH\_CONTACT\_VIRTUAL\_VIA\_SEGMENTATION\_RATIO command without the MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER command. However, if you use the MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER command alone, the StarRC tool issues an error message.

### Example

MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER: 4

### See Also

- [TRENCH\\_CONTACT\\_VIRTUAL\\_VIA\\_SEGMENTATION\\_RATIO](#)

---

## MERGE\_INSTANCE\_PORTS

Decreases the effective resistance connecting the schematic port to the rest of the net when set to YES.

### Syntax

MERGE\_INSTANCE\_PORTS: YES | NO

### Arguments

| Argument     | Description                                                                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Electrically merges all ports, which causes the branches to be treated as parallel-connected resistive paths during extraction, netlist reduction, and netlist generation |
| NO (default) | Matches one of the layout ports to the single schematic port, leaving the other branches as dangling in the parasitic netlist                                             |

### Description

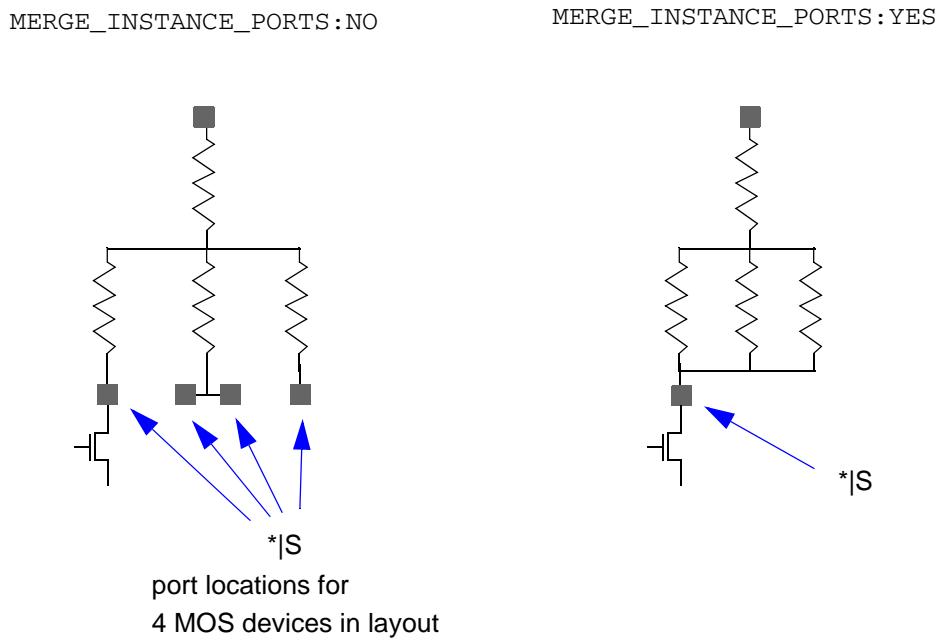
This command is valid only when the XREF:COMPLETE command is also used.

For parallel 1:N (schematic:layout) merging, the command electrically merges all of the N layout instance ports into a single port for netlist generation on each of the source, drain, and gate nets.

For parallel M:N merging where N > M, the command electrically merges each of the extra unmatched (N-M) layout ports with one of the matched M layout ports. The final parasitic netlist contains M ports for each of the source, drain, and gate nets.

[Figure 15-14](#) illustrates the effect of the command.

Figure 15-14 XREF:COMPLETE Parasitic Netlist With MERGE\_INSTANCE\_PORTS



### See Also

- [XREF:COMPLETE](#)

---

## MERGE\_PARALLEL\_DEVICES

Enables merging of parallel devices identified by the IC Validator tool.

### Syntax

MERGE\_PARALLEL\_DEVICES: YES | NO

### Arguments

| Argument     | Description                      |
|--------------|----------------------------------|
| YES          | Enables parallel device merging  |
| NO (default) | Disables parallel device merging |

### Description

The MERGE\_PARALLEL\_DEVICES command enables the StarRC tool to use information provided by the IC Validator tool about merged parallel devices. This capability provides netlist reduction for the instance section of the output netlist. The command is valid for the IC Validator flow only.

During layout versus schematic checking, the IC Validator tool identifies parallel layout devices based on option settings in the tool. You must verify the correct use of the IC Validator options. Merging is typically used for devices with identical properties. The StarRC tool uses the IC Validator binary cdb file directly and does not check the validity of the merging operation.

When parallel device merging occurs, one device instance (known as the master instance) from a set of parallel devices is selected to represent the set. The device without the at sign (@) delimiter is retained as the master instance. The number of merged devices is added to the device properties using the format  $m=N$ , where N is the number of merged devices.

Merging parallel devices might not preserve resistances that exist on the paths between the parallel devices.

The MERGE\_PARALLEL\_DEVICES command has an effect only if the netlist instance section is enabled with the NETLIST\_INSTANCE\_SECTION command and the XREF command is set to YES.

For example, consider parallel (and identical) device instances as follows:

```
XA/T2 A/T2:DRN A/T2:GATE A/T2:SRC A/T2:BULK pmos l=0.02u
ps=4.8e-07 as=1.19e-14 pd=2.3e-07 ad=5.1e-15 w=0.3u
XA/T2@2 A/T2@2:DRN A/T2@2:GATE A/T2@2:SRC A/T2:BULK pmos l=0.02u
ps=4.8e-07 as=1.19e-14 pd=2.3e-07 ad=5.1e-15 w=0.3u
```

After merging, only one instance (the master instance) is reported, with its original properties and the additional `m` property set to 2, indicating that two of these devices appear in parallel in the layout:

```
XA/T2 A/T2:DRN A/T2:GATE A/T2:SRC A/T2:BULK pmos l=0.02u
ps=4.8e-07 as=1.19e-14 pd=2.3e-07 ad=5.1e-15 w=0.3u m=2
```

---

## MERGE\_VIAS\_IN\_ARRAY

Specifies whether vias in an array are merged.

### Syntax

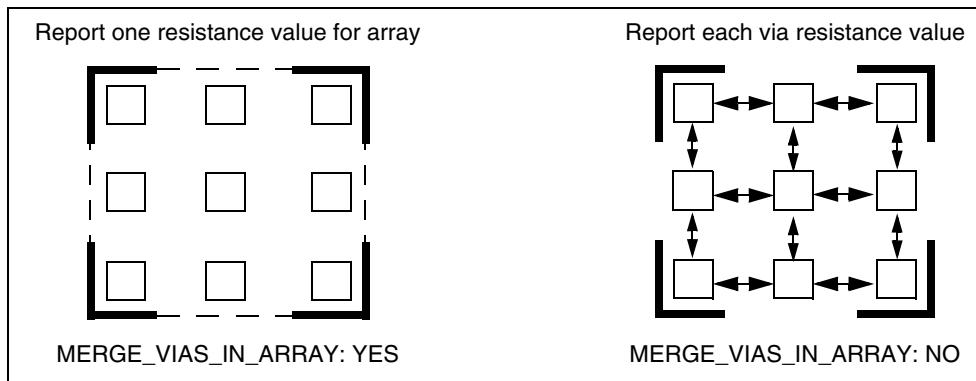
MERGE\_VIAS\_IN\_ARRAY: YES | NO

### Arguments

| Argument     | Description                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Merges resistance values for a via array and reports one resistance value for the array in the netlist                                     |
| NO (default) | Netlists each via in an array structure and reports parasitic resistors between vias in the netlist output with separate resistance values |

### Description

Resistance values between vias can be netlisted by specifying NO. By default, this command is set to YES to merge vias into one reported resistance.



### Examples

#### Example 1

MERGE\_VIAS\_IN\_ARRAY: YES

The result of this example is the following output:

```
13 min_lsb_led[4]:45min_lsb_led[4]:46 0.72000 // $a=2.00000 $lvl=5
```

## Example 2

```
MERGE_VIAS_IN_ARRAY:NO
VIA_COVERAGE_OPTION_FILE: via_file_name
```

Content of the *via\_file\_name* file:

```
VIA1 {Xrange=100,100;Yrange=100,100;
Landing=100,80,10,40,10;Coverage=100,80,10,40,10}
```

The result of this example is the following output:

```
14 min_lsb_led[4]:45 min_lsb_led[4]:46 1.44000 // $a=1.00000 $lvl=5
15 min_lsb_led[4]:54 min_lsb_led[4]:55 1.44000 // $a=1.00000 $lvl=5
```

## See Also

- [KEEP\\_VIA\\_NODES](#)
- [VIA\\_COVERAGE\\_OPTION\\_FILE](#)

---

## MESSAGE\_SUPPRESSION

Limits the number of times a specific warning or error message is reported.

### Syntax

MESSAGE\_SUPPRESSION: *ID\_1:limit1 ID\_2:limit2 ...*

### Arguments

| Argument              | Description                                             |
|-----------------------|---------------------------------------------------------|
| <i>ID_1, ID_2</i>     | Affected message ID                                     |
| <i>limit1, limit2</i> | Maximum number of times to report the specified message |

### Description

In cases where StarRC generates a large number of warning, error, or informational messages, you might want to limit the number of times that similar messages (messages that have the same ID) are issued.

The MESSAGE\_SUPPRESSION command limits the number of times that specific messages are issued. The limit applies only to messages whose IDs are listed in the command. If the same message ID appears in more than one MESSAGE\_SUPPRESSION command, the limit specified in the last statement takes precedence.

The similar MESSAGE\_SUPPRESSION\_LIMIT command applies a limit to all warning, error, and informational messages and does not require you to name specific message IDs.

Limits for the following messages are fixed at 1000 and are not affected by the MESSAGE\_SUPPRESSION or MESSAGE\_SUPPRESSION\_LIMIT commands:

- Shorts message EX-503
- Fill shorts message EX-505
- SMIN violation messages EX-792 and EX-356
- Via violation message EX-714

### Errors

By default, StarRC issues up to 100 messages of the same type. You can increase or decrease this limit. However, if you set the limit to a value greater than 1000, runtime might be increased and the tool issues an SX-3253 warning message. Values greater than 100,000 are reset to 100,000 and the tool issues an SX-3252 warning message.

## Example

The following command causes StarRC to stop reporting SX-2549 messages after the fifth occurrence and EX-269 messages after the twentieth occurrence:

```
MESSAGE_SUPPRESSION: SX-2549:5 EX-269:20
```

## See Also

- [MESSAGE\\_SUPPRESSION\\_LIMIT](#)

---

## MESSAGE\_SUPPRESSION\_LIMIT

Limits the number of times any single warning, error, or informational message is reported.

### Syntax

MESSAGE\_SUPPRESSION\_LIMIT: *count\_value*

### Arguments

| Argument           | Description                                                          |
|--------------------|----------------------------------------------------------------------|
| <i>count_value</i> | Maximum number of times to report any single message<br>Default: 100 |

### Description

In cases where StarRC generates a large number of warning, error, or informational messages, you might want to limit the number of times that similar messages are issued. The MESSAGE\_SUPPRESSION\_LIMIT command limits the number of times that any message with the same ID is reported. The limit applies to all messages, with the following exceptions:

- Messages specified in a MESSAGE\_SUPPRESSION command
- The following messages, whose limits are fixed at 1000:
  - Shorts message EX-503
  - Fill shorts message EX-505
  - SMIN violation messages EX-792 and EX-356
  - Via violation message EX-714

If you specify a value greater than 1000, the tool issues an SX-3253 warning. Values greater than 100,000 are reset to 100,000 and the tool issues an SX-3252 warning.

### Example

The following commands cause StarRC to stop reporting most messages after the tenth occurrence, but to report EX-269 messages up to twenty times:

```
MESSAGE_SUPPRESSION_LIMIT: 10
MESSAGE_SUPPRESSION: EX-269:20
```

### See Also

- [MESSAGE\\_SUPPRESSION](#)

---

## METAL\_FILL\_BLOCK\_NAME

Changes the top-level block name in the metal-fill database to match the corresponding block name in the main design database.

### Syntax

METAL\_FILL\_BLOCK\_NAME: *top\_block\_name*

### Arguments

| Argument              | Description                                    |
|-----------------------|------------------------------------------------|
| <i>top_block_name</i> | The top-block name in the fill design database |

### Description

The METAL\_FILL\_BLOCK\_NAME command specifies the metal fill top-level block name. The METAL\_FILL\_BLOCK\_NAME command is valid for both Oasis and GDSII data formats. The METAL\_FILL\_BLOCK\_NAME command takes precedence over the METAL\_FILL\_GDS\_BLOCK command if both are present in the command file.

By default, the StarRC tool assumes the metal fill top-level block name is the same as the top-level block name of the design. Use this option when the metal fill top-level block name is not the same as the top-level block name of the design.

When the names are inconsistent, StarRC terminates and issues an error message.

### Example

The following example specifies the top-level block name in the fill design.

METAL\_FILL\_BLOCK\_NAME: top\_fill\_block1

### See Also

- [METAL\\_FILL\\_GDS\\_BLOCK](#)
- [METAL\\_FILL\\_GDS\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE\\_NET\\_NAME](#)

---

## METAL\_FILL\_GDS\_BLOCK

Changes the top-level block name in the metal-fill GDSII database to match the corresponding block name in the main design database.

### Syntax

METAL\_FILL\_GDS\_BLOCK: *top\_block\_name*

### Arguments

| Argument              | Description                                    |
|-----------------------|------------------------------------------------|
| <i>top_block_name</i> | The top-block name in the main design database |

### Description

The METAL\_FILL\_GDS\_BLOCK command specifies a metal fill GDS top-level block name that is different from the top-level block name of the design.

By default, the StarRC tool assumes the metal fill GDS top-level block name is the same as the top-level block name of the design. Use this option when the metal fill GDS top-level block name is not the same as the top-level block name of the design.

When the names are inconsistent, StarRC terminates and issues an error message.

### Example

The following example specifies the top-level block name in the GDSII database name in the design. StarRC uses this name to replace the corresponding top-level block name in the GDSII database, if the names do not match.

METAL\_FILL\_GDS\_BLOCK: top\_block1

### See Also

- [METAL\\_FILL\\_GDS\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE\\_NET\\_NAME](#)

---

## METAL\_FILL\_GDS\_FILE

Specifies the GDSII file containing metal fill data.

### Syntax

METAL\_FILL\_GDS\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                    |
|------------------|----------------------------------------------------------------|
| <i>file_name</i> | Name of GDSII file containing metal fill data<br>Default: none |

### Description

The METAL\_FILL\_GDS\_FILE command supports either hierarchical or flat GDSII files. All shapes on layers mapped by GDS\_LAYER\_MAP\_FILE within the master definition of BLOCK in METAL\_FILL\_GDS\_FILE and its child cells are treated as metal fill objects for extraction. All other data not referenced by the master definition of BLOCK is ignored.

METAL\_FILL\_POLYGON\_HANDLING applies to all shapes imported through the METAL\_FILL\_GDS\_FILE interface.

#### The METAL\_FILL\_GDS\_FILE command

- Must be specified together with the GDS\_LAYER\_MAP\_FILE command
- Cannot be used with the METAL\_FILL\_OASIS\_FILE command

#### Note:

StarRC does not support a flow with metal fill polygons connected to more than one power net when metal fill polygons are present in a separate GDSII file from the design database.

When metal fill is embedded in a Milkyway or LEF/DEF database, StarRC reads the data when you specify a METAL\_FILL\_GDS\_FILE command.

You can specify gzip and compressed GDS files for this command.

### See Also

- [GDS\\_FILE](#)
- [GDS\\_LAYER\\_MAP\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE\\_NET\\_NAME](#)

---

## METAL\_FILL\_GDS\_FILE\_NET\_NAME

Ties metal fill polygons to a power or ground net.

### Syntax

METAL\_FILL\_GDS\_FILE\_NET\_NAME: *net\_name*

### Arguments

| Argument        | Description                          |
|-----------------|--------------------------------------|
| <i>net_name</i> | The layout net name<br>Default: none |

### Description

You can use the METAL\_FILL\_GDS\_FILE\_NET\_NAME command to tie metal fill polygons to a power or ground net.

The METAL\_FILL\_GDS\_FILE\_NET\_NAME command

- Must be specified together with METAL\_FILL\_GDS\_FILE and METAL\_FILL\_POLYGON\_HANDLING: GROUNDED
- Cannot be used with the METAL\_FILL\_OASIS\_FILE\_NET\_NAME command

### See Also

- [GDS\\_LAYER\\_MAP\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE](#)
- [METAL\\_FILL\\_POLYGON\\_HANDLING](#)

---

## METAL\_FILL\_GDS\_MAG

Specifies the scaling factor that is applied to the GDSII metal fill data.

### Syntax

METAL\_FILL\_GDS\_MAG: *factor*

### Arguments

| Argument      | Description                          |
|---------------|--------------------------------------|
| <i>factor</i> | Magnification factor<br>Default: 1.0 |

### Description

The METAL\_FILL\_GDS\_MAG command specifies the scaling factor that is applied to the GDSII metal fill data.

The metal fill offset specified by the METAL\_FILL\_GDS\_OFFSET command is not multiplied by the scaling factor.

Note:

The METAL\_FILL\_GDS\_MAG command cannot be used with the METAL\_FILL\_OASIS\_MAG command.

### Example

In the following example, the GDSII metal fill data length and width are multiplied by a factor of 0.8:

METAL\_FILL\_GDS\_MAG: 0.8

The total entire area of the design would therefore be scaled by a factor of 0.64.

### See Also

- [METAL\\_FILL\\_GDS\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_OFFSET](#)

---

## METAL\_FILL\_GDS\_OFFSET

Specifies the origin of the metal fill GDSII file.

### Syntax

METAL\_FILL\_GDS\_OFFSET: *x\_coordinate y\_coordinate*

### Arguments

| Argument            | Description                                    |
|---------------------|------------------------------------------------|
| <i>x_coordinate</i> | X-coordinate<br>Units: microns<br>Default: 0.0 |
| <i>y_coordinate</i> | Y-coordinate<br>Units: microns<br>Default: 0.0 |

### Description

The METAL\_FILL\_GDS\_OFFSET command specifies the coordinates of the origin of the metal fill GDSII file. This command does not affect the magnification factor behavior.

#### Note:

The METAL\_FILL\_GDS\_OFFSET command cannot be used with the METAL\_FILL\_OASIS\_OFFSET command.

### Example

METAL\_FILL\_GDS\_OFFSET: 10.8 5.3

### See Also

- [METAL\\_FILL\\_GDS\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE\\_NET\\_NAME](#)
- [METAL\\_FILL\\_POLYGON\\_HANDLING](#)

---

## METAL\_FILL\_OASIS\_FILE

Specifies the OASIS file containing metal fill data.

### Syntax

METAL\_FILL\_OASIS\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                    |
|------------------|----------------------------------------------------------------|
| <i>file_name</i> | Name of OASIS file containing metal fill data<br>Default: none |

### Description

The METAL\_FILL\_OASIS\_FILE command supports either hierarchical or flat OASIS files. All shapes on layers mapped by the OASIS\_LAYER\_MAP\_FILE file within the master definition of the block in the METAL\_FILL\_OASIS\_FILE file and its child cells are treated as metal fill objects for extraction. All other data not referenced by the master definition of the block is ignored. The METAL\_FILL\_POLYGON\_HANDLING command applies to all shapes imported through the METAL\_FILL\_OASIS\_FILE command.

#### The METAL\_FILL\_OASIS\_FILE command

- Must be specified together with the OASIS\_LAYER\_MAP\_FILE command
- Cannot be used with the METAL\_FILL\_GDS\_FILE command

#### Note:

StarRC does not support a flow with metal fill polygons connected to more than one power net when metal fill polygons are present in a separate OASIS file from the design database.

When metal fill is embedded in a Milkyway or LEF/DEF database, StarRC reads the data when you specify a METAL\_FILL\_OASIS\_FILE command.

You can specify gzip and compressed OASIS files for this command.

### See Also

- [METAL\\_FILL\\_OASIS\\_FILE\\_NET\\_NAME](#)
- [OASIS\\_FILE](#)
- [OASIS\\_LAYER\\_MAP\\_FILE](#)

---

## METAL\_FILL\_OASIS\_FILE\_NET\_NAME

Ties metal fill polygons to a power or ground net.

### Syntax

METAL\_FILL\_OASIS\_FILE\_NET\_NAME: *net\_name*

### Arguments

| Argument        | Description                          |
|-----------------|--------------------------------------|
| <i>net_name</i> | The layout net name<br>Default: none |

### Description

You can use the METAL\_FILL\_OASIS\_FILE\_NET\_NAME command to tie metal fill polygons to a power or ground net.

The METAL\_FILL\_OASIS\_FILE\_NET\_NAME command

- Must be specified together with METAL\_FILL\_OASIS\_FILE and METAL\_FILL\_POLYGON\_HANDLING: GROUNDED
- Cannot be used with the METAL\_FILL\_GDS\_FILE\_NET\_NAME command

### See Also

- [METAL\\_FILL\\_OASIS\\_FILE](#)
- [METAL\\_FILL\\_POLYGON\\_HANDLING](#)
- [OASIS\\_LAYER\\_MAP\\_FILE](#)

---

## METAL\_FILL\_OASIS\_MAG

Specifies the scaling factor that is applied to the OASIS metal fill data.

### Syntax

METAL\_FILL\_OASIS\_MAG: *factor*

### Arguments

| Argument      | Description                          |
|---------------|--------------------------------------|
| <i>factor</i> | Magnification factor<br>Default: 1.0 |

### Description

The METAL\_FILL\_OASIS\_MAG command specifies the scaling factor that is applied to the OASIS metal fill data.

The metal fill offset specified by the METAL\_FILL\_GDS\_OFFSET command is not multiplied by the scaling factor.

Note:

The METAL\_FILL\_OASIS\_MAG command cannot be used with the METAL\_FILL\_GDS\_MAG command.

### Example

In the following example, the OASIS metal fill data length and width are multiplied by a factor of 0.8:

METAL\_FILL\_OASIS\_MAG: 0.8

The total entire area of the design would therefore be scaled by a factor of 0.64.

### See Also

- [METAL\\_FILL\\_OASIS\\_FILE](#)
- [METAL\\_FILL\\_OASIS\\_OFFSET](#)

---

## METAL\_FILL\_OASIS\_OFFSET

Specifies the origin of the metal fill OASIS file.

### Syntax

METAL\_FILL\_OASIS\_OFFSET: *x\_coordinate y\_coordinate*

### Arguments

| Argument            | Description                                    |
|---------------------|------------------------------------------------|
| <i>x_coordinate</i> | X-coordinate<br>Units: microns<br>Default: 0.0 |
| <i>y_coordinate</i> | Y-coordinate<br>Units: microns<br>Default: 0.0 |

---

### Description

The METAL\_FILL\_OASIS\_OFFSET command specifies the coordinates of the origin of the metal fill OASIS file. This command does not affect the magnification factor behavior.

The METAL\_FILL\_OASIS\_OFFSET command cannot be used with the METAL\_FILL\_GDS\_OFFSET command.

### Example

METAL\_FILL\_OASIS\_OFFSET: 10.8 5.3

### See Also

- [METAL\\_FILL\\_OASIS\\_FILE](#)
- [METAL\\_FILL\\_OASIS\\_FILE\\_NET\\_NAME](#)
- [METAL\\_FILL\\_POLYGON\\_HANDLING](#)

---

## METAL\_FILL\_POLYGON\_HANDLING

Translates metal fill polygons into the internal format before performing extraction.

### Syntax

`METAL_FILL_POLYGON_HANDLING: IGNORE | GROUNDED | FLOATING | AUTOMATIC`

### Arguments

| Argument            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IGNORE<br>(default) | Performs resistance and capacitance extraction without considering the effect of metal fill polygons. In Milkyway, metal fill polygons are ignored only if they have the correct <code>ROUTE_TYPE</code> .                                                                                                                                                                                                                                                                                                                                                                                                              |
| GROUNDED            | Performs extraction by treating metal fill as grounded. By default, StarRC treats metal fill polygons as connected to ground during extraction, unless the Milkyway or LEF/DEF design database ties the metal fill polygons to a specific power or ground net or the <code>METAL_FILL_GDS_FILE_NET_NAME</code> command is specified.                                                                                                                                                                                                                                                                                    |
| FLOATING            | Performs extraction by treating the metal fill as floating. This argument overrides the type of metal fill polygons provided in the input database. This means that even though the input database has grounded fill polygons, the <code>METAL_FILL_POLYGON_HANDLING</code> command extracts the capacitance as if the fill polygons were floating and ignores the existence of fill objects in the database.                                                                                                                                                                                                           |
| AUTOMATIC           | Performs automatic extraction by treating metal fill as floating or grounded.<br>For LEF/DEF designs, this argument parses the DEF file and translates fill polygons based on the section in which they appear in the DEF file.<br>For Milkyway designs, this argument translates fills based on the <code>ROUTE_TYPE</code> attached to a net ID or not having a net ID (floating). Both floating and grounded fills are allowed in the same design, if they are identified as such correctly.<br>To ensure that fills are treated properly, use the <code>insert_metal_filler</code> command in the IC Compiler tool. |

### Description

The `METAL_FILL_POLYGON_HANDLING` command translates metal fill polygons into the internal format before performing extraction. This process depends on the setting of this command. If metal fill polygons are written into the FILL view in the Milkyway database, you need to also set the `MILKYWAY_ADDITIONAL_VIEWS` command in StarRC.

**Table 15-6** explains the usage of commands related to the METAL\_FILL\_POLYGON\_HANDLING command.

*Table 15-6 Usage of Commands Related to the METAL\_FILL\_POLYGON\_HANDLING Command*

| Related command              | Usage of related command                                                                                                                                                                                                                                        |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| METAL_FILL_GDS_FILE_NET_NAME | The METAL_FILL_GDS_FILE_NET_NAME command is required when you want to tie metal fill polygons to a power or ground net. The net name should match the layout net name. This command works only when the METAL_FILL_POLYGON_HANDLING command is set to GROUNDED. |
| METAL_FILL_GDS_FILE          | The METAL_FILL_GDS_FILE command supports either hierarchical or flat GDSII files. The cell name of the GDS file must be the same as the BLOCK name or top cell name of the design.                                                                              |
| GDS_LAYER_MAP_FILE           | The GDS_LAYER_MAP_FILE command specifies the file that contains information about the mapping between the GDSII layer number and the layer name in the design database.                                                                                         |
| GDS_FILE                     | If the GDS_FILE command is specified for a LEF/DEF database, a single unified layer mapping file should be used for both GDSII files.                                                                                                                           |

## Example

This command treats the metal fill polygons as grounded:

```
METAL_FILL_POLYGON_HANDLING: GROUNDED
```

## See Also

- [GDS\\_FILE](#)
- [GDS\\_LAYER\\_MAP\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE](#)
- [METAL\\_FILL\\_GDS\\_FILE\\_NET\\_NAME](#)
- [MILKYWAY\\_ADDITIONAL\\_VIEWS](#)

---

## METAL\_SHEET\_OVER\_AREA

Specifies an area for sheet metal coupling capacitance measurement.

### Syntax

METAL\_SHEET\_OVER\_AREA: *layer\_name* *X1* *Y1* *X2* *Y2*

### Arguments

| Argument            | Description                                           |
|---------------------|-------------------------------------------------------|
| <i>layer_name</i>   | Metal layer name<br>Default: none                     |
| <i>X1</i> <i>Y1</i> | Lower left coordinates of the area<br>Units: microns  |
| <i>X2</i> <i>Y2</i> | Upper right coordinates of the area<br>Units: microns |

---

### Description

Use this command to associate a sheet of metal to a user-defined net name and output suffix. You can use the command multiple times to specify multiple metal sheets. You can optionally specify the SHEET\_COUPLE\_TO\_NET\_LEVEL command to enable a net name suffix.

You must verify that the sheet metal areas do not cause metal shorts. The StarRC tool does not check for areas of metal that overlay each other. The tool checks that the specified layer is a metal layer and that the bounding box coordinates are correct.

### Example

```
METAL_SHEET_OVER_AREA: METAL2 0 0 100 100
METAL_SHEET_OVER_AREA: METAL2 200 200 400 400
METAL_SHEET_OVER_AREA: METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

### See Also

- [SHEET\\_COUPLE\\_TO\\_NET](#)
- [SHEET\\_COUPLE\\_TO\\_NET\\_LEVEL](#)

---

## MILKYWAY\_ADDITIONAL\_VIEWS

Specifies a Milkyway view.

### Syntax

MILKYWAY\_ADDITIONAL\_VIEWS: *view\_name*

### Arguments

| Argument         | Description                                  |
|------------------|----------------------------------------------|
| <i>view_name</i> | Name of the additional view<br>Default: none |

### Description

Milkyway stores design data in different files called *views* inside a generated Milkyway library. Use this command to read views other than CEL, FRAM, TIM, or PWR views. The previously listed views are automatically read by StarRC. This command causes StarRC to read an additional view.

See the Milkyway documentation for a complete list of output views.

### Example

MILKYWAY\_ADDITIONAL\_VIEWS: FILL

### See Also

- [METAL\\_FILL\\_POLYGON\\_HANDLING](#)

---

## MILKYWAY\_CELL\_VIEW

Creates a white-space-delimited list specifying cells that StarRC uses as the layout cell view.

### Syntax

MILKYWAY\_CELL\_VIEW: *list\_of\_cells*

### Arguments

| Argument             | Description                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------|
| <i>list_of_cells</i> | List of cells for which StarRC uses the layout cell view during extraction, if available<br>Default: none |

### Description

This command creates a white-space-delimited list specifying cells that StarRC uses as the layout cell view (Milkyway CEL view) during extraction if it is available. You can specify this command multiple times in a single command file. The asterisk (\*) and question mark (?) wildcard characters are acceptable.

#### Note:

This command applies to skip cells and their child cells only; the CEL view is always used for non skipped cell masters.

For skip cells not on this list, the Milkyway FRAM view represents the cell contents during extraction. The FRAM view typically contains all pin shapes and obstructions.

StarRC attempts to translate the Milkyway cell view for each skip cell on this list. The cell view contains the actual physical layout, including nonroute layers. Specifying a cell in this list automatically includes all child cells. If a cell view cannot be found for a cell contained in this list, StarRC issues a warning and reverts to the FRAM view for that cell and all child cells.

### Example

```
MILKYWAY_CELL_VIEW: cell1 cell2 cell3
MILKYWAY_CELL_VIEW: cellA cellB cell? *C
MILKYWAY_CELL_VIEW: *
```

### See Also

- [SKIP\\_CELLS](#)

---

## MILKYWAY\_DATABASE

Specifies the location of the input Milkyway layout database.

### Syntax

MILKYWAY\_DATABASE: *layout\_library*

### Arguments

| Argument              | Description                                                                |
|-----------------------|----------------------------------------------------------------------------|
| <i>layout_library</i> | The name of the layout library from the Milkyway database<br>Default: none |

### Description

The MILKYWAY\_DATABASE command is mandatory for a Milkyway extraction flow.

You must specify the block for extraction with the BLOCK command.

### See Also

- [BLOCK](#)

---

## MILKYWAY\_EXPAND\_HIERARCHICAL\_CELLS

Flattens any routed cell instance that has the cell type or property *macro*.

### Syntax

MILKYWAY\_EXPAND\_HIERARCHICAL\_CELLS: YES | NO

### Arguments

| Argument     | Description                                                                            |
|--------------|----------------------------------------------------------------------------------------|
| YES          | Flattens the cell types that are routed in the Milkyway database and to run extraction |
| NO (default) | Does not flatten macro or routed cells                                                 |

### Description

For this command, “routed” means any cell that contains one or more nets or more than one instance placement. Any other cell type remains a member of the list of skip cells specified by the SKIP\_CELLS command.

The MILKYWAY\_EXPAND\_HIERARCHICAL\_CELLS command automatically configures the skip cells list and takes precedence over the SKIP\_CELLS command.

### See Also

- [SKIP\\_CELLS](#)

---

## MILKYWAY\_EXTRACT\_VIEW

Selects the XTR (Milkyway extract view) layout description as the input for your StarRC extraction.

### Syntax

MILKYWAY\_EXTRACT\_VIEW: YES | NO

### Arguments

| Argument     | Description                                |
|--------------|--------------------------------------------|
| YES          | StarRC reads the Milkyway XTR view         |
| NO (default) | StarRC does not read the Milkyway XTR view |

### Description

This command selects the XTR (Milkyway extract view) layout description as the input for your StarRC extraction. This command is mandatory for a Hercules flow.

### Errors

For StarRC to read in data correctly from Hercules output, you must specify the MILKYWAY\_EXTRACT\_VIEW: YES command. If the MILKYWAY\_EXTRACT\_VIEW command is not set, StarRC treats the Milkyway library that was generated by Hercules as if it were a library generated by Synopsys physical design tools and displays the following message:

WARNING: cannot open milkyway cell top:CEL!

### See Also

- [BLOCK](#)
- [MILKYWAY\\_DATABASE](#)
- [POWER\\_NETS](#)

---

## MILKYWAY\_REF\_LIB\_MODE

Specifies the order in which libraries are searched for a cell master.

### Syntax

MILKYWAY\_REF\_LIB\_MODE: NONE | HIER | FILE

### Arguments

| Argument       | Description                                                                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NONE (default) | The cell is first searched for in the reference library. The search order for the reference library is the same as it is in the main design library.        |
| HIER           | The child cell is searched for in the same library as its parent library. If the child cell is not found, the default mode is used.                         |
| FILE           | A reference control file in the main library specifies which reference library is checked first. The specified order is followed to find and open the cell. |

### Description

When extracting Milkyway databases, the MILKYWAY\_REF\_LIB\_MODE command controls the search preference among libraries and reference libraries for the cell master.

#### Note:

If you use the IC Compiler tool for place and route, you should specify the MILKYWAY\_REF\_LIB\_MODE: HIER command to follow the same search sequence as the IC Compiler tool.

In the HIER mode, the CEL views are uniquified by appending special characters to the cell names so that they are hierarchically separated from each other in case of name collisions across different libraries. This unification might affect the names used in skip cells when exploding a specific cell. For example, if a cell named macroA needs to be exploded or flattened, you must use the SKIP\_CELLS: macroA\* command to explode the cell across different libraries.

Other tools use different commands or options to specify the library search order. See the documentation for those tools for more details to ensure that you specify a consistent search order throughout your entire design flow.

## Examples

In the library structure shown in [Example 15-8](#), if you specify MILKYWAY\_REF\_LIB\_MODE: NONE, StarRC uses Cell A in ref lib 1. If you specify MILKYWAY\_REF\_LIB\_MODE: HIER, StarRC uses Cell A in the main library.

### Example 15-8

```
[Top/top lib] --[A1/(instantiated from cell A)]
| - Cell A
|-----[lib1/ref lib] - cell A
```

In the library structure shown in [Example 15-9](#), StarRC uses ref lib/lib1 cell A for instance A1.

### Example 15-9

```
[Top/top lib] --[A1/(instantiated from cell A)]
|-----[lib1/ref lib] - cell A
|-----[lib2/ref lib] - cell A
```

In the library structure shown in [Example 15-10](#), StarRC uses ref lib/lib 1.

### Example 15-10

```
[Top/top lib] --[A1/(instantiated from cell A)]
|-----[lib1/ref lib] - cell A
|-----[lib2/ref lib] - cell A
```

## See Also

- [MILKYWAY\\_DATABASE](#)

---

## MILKYWAY\_SHOW\_CELL\_INFO\_DETAIL

Writes information about every translated cell into a log file.

### Syntax

MILKYWAY\_SHOW\_CELL\_INFO\_DETAIL: YES | NO

### Arguments

| Argument     | Description                                                    |
|--------------|----------------------------------------------------------------|
| YES          | Writes information about every translated cell into a log file |
| NO (default) | Does not report cell details                                   |

### Description

Milkyway designs contain data in different files known as views, such as the CEL and FRAM views. One cell might have many different views. In addition, different design libraries might use the same cell name. Many commands in the StarRC command file affect which libraries, cells, and views are translated. Therefore, it might be difficult to know exactly which design files are used in an extraction run.

To obtain detailed information about every translated cell, set the MILKYWAY\_SHOW\_CELL\_INFO\_DETAIL command to YES.

If the input data contains GDSII or OASIS data, the StarRC tool writes the information into the summary/cells.sum file. Otherwise, the tool writes the information into the star/cell\_info.detail file.

### Example

The following lines are examples of the information that might be included in the cell\_info.detail file. Each line contains the cell name, the view name (highlighted in color in this example), and the library name.

```
Cell Info: FILL2BWP16P90ULVT FRAM /slowfs/dept5242t/MW/tcbn16ff11bwp 16p
Cell Info: mimcap_unitcell FRAM /slowfs/dept52425t/MW_w_power
Cell Info: toprt CEL /na4apd/starrc/mw/xtdesign
Cell Info: toprt FILL /na4apd/starrc/mw/xtdesign
```

### See Also

- [MILKYWAY\\_DATABASE](#)

---

## MODE

Sets the level of extraction accuracy versus speed.

### Syntax

MODE: 200 | 400

### Arguments

| Argument      | Description                                                                                                                                    |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 200 (default) | Provides a balance between runtime performance and accuracy compared to the field solver. Use this mode for faster extraction turnaround time. |
| 400           | Provides better accuracy than MODE: 200, using field solver results as a reference. Use this mode for higher extraction accuracy.              |

### Description

The `MODE` command specifies the level of accuracy versus speed and is used only in transistor-level flows for established process nodes. Many advanced process node designs, including FinFET and trench contact designs, no longer require the `MODE` command. StarRC issues warning messages when the `MODE` command is present but not needed in the StarRC command file for a specific design.

For gate-level designs, such as Milkyway and LEF/DEF flows, the `MODE` command is not needed because StarRC automatically optimizes extraction accuracy and speed.

If the command file contains the `MODE` command for a gate-level design, the StarRC tool issues the following warning message:

MODE is no longer required for Milkyway and LEF/DEF flows. Ignoring MODE in command file. Please refer to StarRC H-2012.12 release notes for more information.

### Example

The following command sets the mode to 400:

MODE: 400

### See Also

- [EXTRACTION](#)

---

## MODEL\_TYPE

Specifies whether the reference model comes from a layout or schematic.

### Syntax

MODEL\_TYPE: LAYOUT | SCHEMATIC

### Arguments

| Argument         | Description                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------------------|
| LAYOUT (default) | Specifies that the reference model has been generated from a layout                                                            |
| SCHEMATIC        | Specifies that the reference model has been generated from a schematic. This setting is not allowed with the XREF: NO command. |

### Description

This command specifies whether the reference model in the HN\_NETLIST\_MODEL\_NAME, RETAIN\_CAPACITANCE\_CAP\_MODELS, or HN\_NETLIST\_SPICE\_TYPE commands comes from a layout or schematic.

If MODEL\_TYPE is not specified in the command file, the default is LAYOUT.

StarRC reports the layout net names generated by Hercules, IC Validator, or Calibre during ideal layout extraction.

| XREF command setting                                                   | Behavior                                                  |
|------------------------------------------------------------------------|-----------------------------------------------------------|
| XREF: YES   COMPLETE<br>(for schematic model name output)<br>(default) | Prints schematic model name in the parasitic netlist      |
| XREF: NO                                                               | Prints the layout model name (default)                    |
| XREF: YES<br>(for layout model name output)                            | Sets XREF_USE_LAYOUT_DEVICE_NAME: YES in the command file |

### Example

```
MODEL_TYPE: SCHEMATIC
HN_NETLIST_MODEL_NAME: myrcxtmodel mysim_modelname
```

## See Also

- [XREF](#)
- [HN\\_NETLIST\\_MODEL\\_NAME](#)
- [HN\\_NETLIST\\_SPICE\\_TYPE](#)
- [RETAIN\\_CAPACITANCE\\_CAP\\_MODELS](#)

---

## MOS\_GATE\_CAPACITANCE

Specifies a global loading capacitance per unit area.

### Syntax

MOS\_GATE\_CAPACITANCE: *load\_cap*

### Arguments

| Argument        | Description                                                                                                                             |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>load_cap</i> | MOS gate capacitance<br>Units: farads per square micron<br>Default: 1e-15, but defaults to zero for advanced device property extraction |

### Description

Specifies a global loading capacitance per unit area (in square microns) for MOS gate terminals in the Detailed Standard Parasitic Format (DSPF) and SPEF connectivity sections (\*|I and \*I, respectively) of the output parasitic netlist. Only devices generated by the Hercules commands NMOS and PMOS are assigned this capacitance. In addition, all MOS gates are netlisted with direction "I".

## MOS\_GATE\_DELTA\_RESISTANCE

Specifies whether to extract the gate resistance of MOS devices as a delta network.

### Syntax

`MOS_GATE_DELTA_RESISTANCE: YES | NO`

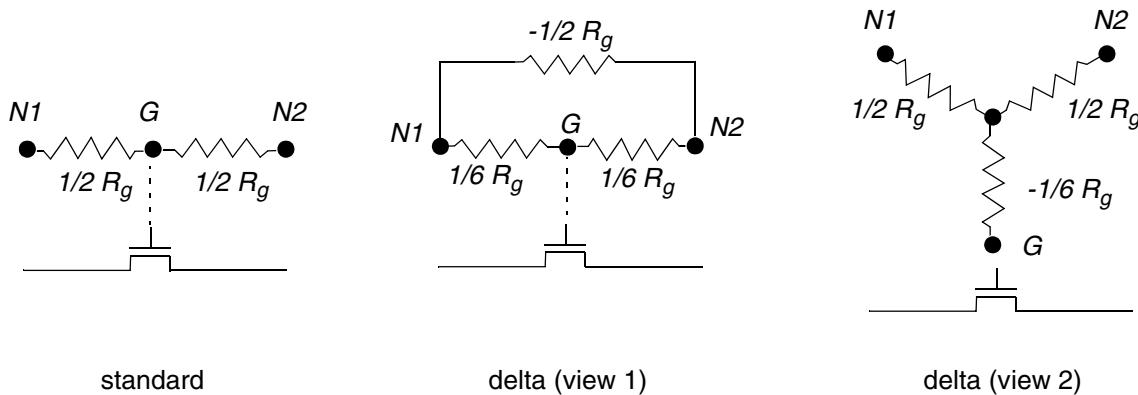
### Arguments

| Argument     | Description                                                                        |
|--------------|------------------------------------------------------------------------------------|
| YES          | Uses the delta network model for MOS gate resistance. Does not apply to X devices. |
| NO (default) | Uses the standard gate resistance model                                            |

### Description

This command changes the gate resistance model for MOS devices to a delta network. [Figure 15-15](#) shows the standard model and two equivalent views of the delta model. Nodes N1 and N2 represent the ends of the gate polygon, while node G represents the ideal gate terminal location.

*Figure 15-15 Standard Gate Resistance and Two Views of Delta Resistance Network*



In the standard model, the resistance from node N1 to node N2 is  $R_g$  and the resistance from either node to node G is  $1/2 R_g$ . The value of  $1/2 R_g$  appears in the parasitic netlist.

If the `MOS_GATE_DELTA_RESISTANCE` command is set to `YES`, a delta network resistance model is used. The two views in [Figure 15-15](#) are equivalent views of the same network. In

the delta model, the resistance from node N1 to node N2 is still  $R_g$ , but the resistance from either node to node G is  $1/3 R_g$ . The value of  $1/3 R_g$  appears in the parasitic netlist.

This model requires negative resistance to achieve the desired result. Some simulators cannot handle negative resistance. In this case, set the `MOS_GATE_DELTA_RESISTANCE` command to `NO` and use the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command to use an alternate model that does not require negative resistance.

The MOS devices affected by this commands includes devices defined by the `ICV_LVS_DEVICE_TYPE_MOS` and `CALIBRE_LVS_DEVICE_TYPE_MOS` commands.

The `MOS_GATE_DELTA_RESISTANCE` command does not apply to X devices. To specify the gate resistance model for X devices, use the `MOS_GATE_DELTA_RESISTANCE_LAYERS` command.

## See Also

- [MOS\\_GATE\\_DELTA\\_RESISTANCE\\_LAYERS](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE\\_LOOP\\_SCALE](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE\\_SINGLE\\_SCALE](#)

---

## MOS\_GATE\_DELTA\_RESISTANCE\_LAYERS

Specifies the database gate layers of MOS and X devices to extract using the delta model for gate resistance.

### Syntax

`MOS_GATE_DELTA_RESISTANCE_LAYERS: layer_list`

### Arguments

| Argument                | Description                                                                                                                                                                                                                                                                                        |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>layer_list</code> | The delta gate resistance model is used for all MOS and X devices whose gates are formed in these layers. Each layer must map to an ITF layer with <code>LAYER_TYPE</code> set to <code>GATE</code> .<br>Space-delimited list of layers. Wildcards * and ! are allowed.<br>Default: !* (no layers) |

### Description

Use the `MOS_GATE_DELTA_RESISTANCE_LAYERS` command in conjunction with the `MOS_GATE_DELTA_RESISTANCE` command to define whether StarRC models MOS devices or X devices with the standard resistance model or the delta network resistance model.

[Figure 15-15](#) illustrates the two resistance models.

Use these two commands in the StarRC command file as follows:

- To use the delta model only for specific MOS or X devices, list the gate terminal layers for those devices in the `MOS_GATE_DELTA_RESISTANCE_LAYERS` command. All other MOS and X devices use the standard resistance model.

```
MOS_GATE_DELTA_RESISTANCE: NO
MOS_GATE_DELTA_RESISTANCE_LAYERS: layer1 layer2 ...
```

- To use the delta model for all MOS devices plus specific X devices, list the gate terminal layers for the desired X devices in the `MOS_GATE_DELTA_RESISTANCE_LAYERS` command and set the `MOS_GATE_DELTA_RESISTANCE` command to YES. (The `MOS_GATE_DELTA_RESISTANCE` command does not apply to X devices.)

```
MOS_GATE_DELTA_RESISTANCE: YES
MOS_GATE_DELTA_RESISTANCE_LAYERS: layer1 layer2 ...
```

The MOS devices affected by these commands includes devices defined by the `ICV_LVS_DEVICE_TYPE_MOS` and `CALIBRE_LVS_DEVICE_TYPE_MOS` commands.

To apply delta network models to multifingered devices, the gate polysilicon must connect only to field polysilicon.

### See Also

- [MOS\\_GATE\\_DELTA\\_RESISTANCE](#)

---

## MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE

Extracts the gate resistance of MOS devices as an adjustable network without negative resistance.

### Syntax

MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE: YES | NO

### Arguments

| Argument     | Description                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------|
| YES          | Extracts gate resistance using a model that does not include negative resistance and allows scaling |
| NO (default) | Does not use this model                                                                             |

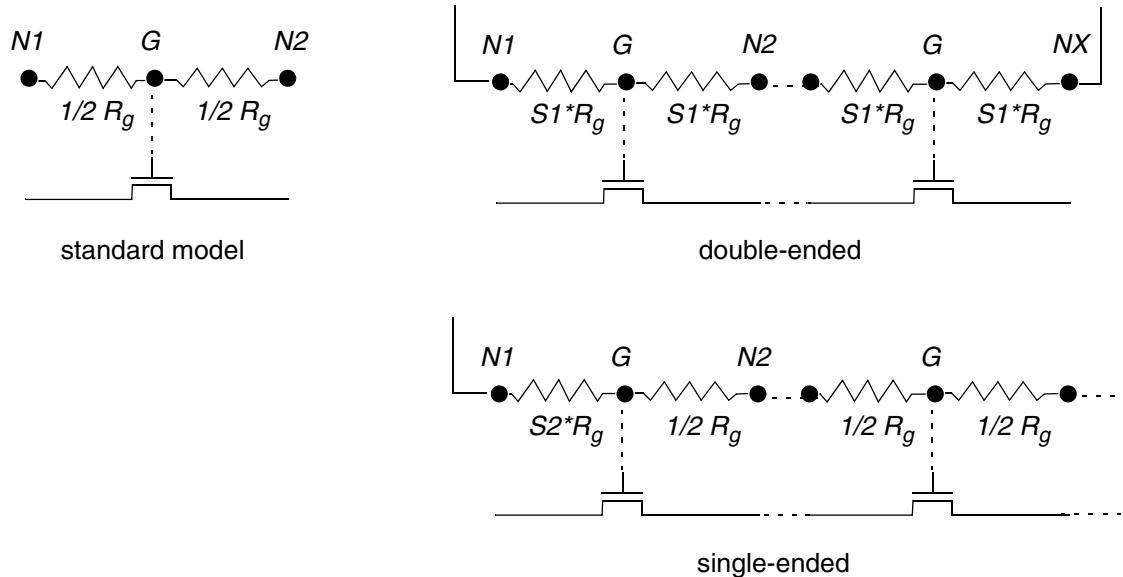
### Description

This command changes the gate resistance model for MOS devices to an adjustable model without negative resistance. [Figure 15-16](#) shows the standard model and the two modes of the alternative model. Nodes N1 and N2 represent the ends of the gate polygon, while node G represents the ideal gate terminal location.

If devices are connected in series, node NX represents the end of the gate polygon of the last device. This alternative model has two design modes: double-ended (looped) mode in which nodes N1 and NX are both connected to a driver net, and single-ended mode in which only node N1 is connected to a driver net.

In the standard model, the resistance from node N1 to node N2 is  $R_g$  and the resistance from either node to node G is  $1/2 R_g$ . The value of  $1/2 R_g$  appears in the parasitic netlist.

**Figure 15-16 Nonnegative Gate Resistance Network Model**



If the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command is set to `YES`, the network resistance model shown in [Figure 15-16](#) is used. S1 and S2 are scale factors as follows:

- Scale factor S1 is for the double-ended mode. S1 is a multiplication factor applied to every resistance between node N1 and node G and between node N2 and node G for every transistor.

Set this factor with the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE` command. The default is 0.333.

- Scale factor S2 is for the single-ended mode. S2 is a multiplication factor applied only to the resistance between node N1 (connected to the driver net) and node G. The resistance between node G and node N2 of this device and the node-to-gate resistance of all other devices in series are fixed at  $1/2 R_g$ .

Set this factor with the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE` command. The default is 0.333.

## See Also

- [MOS\\_GATE\\_DELTA\\_RESISTANCE](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE\\_LOOP\\_SCALE](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE\\_SINGLE\\_SCALE](#)

---

## MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE\_LOOP\_SCALE

Scale factor for use with an alternate gate resistance model.

### Syntax

MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE\_LOOP\_SCALE: *factor\_S1*

### Arguments

| Argument         | Description                                                                        |
|------------------|------------------------------------------------------------------------------------|
| <i>factor_S1</i> | Scale factor for the loop mode of the alternate resistance model<br>Default: 0.333 |

### Description

Specifies a scale factor to use for the loop mode (double-ended mode) of the alternate gate resistance model enabled with the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command.

### See Also

- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE\\_SINGLE\\_SCALE](#)
- [MOS\\_GATE\\_DELTA\\_RESISTANCE](#)

---

## MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE\_SINGLE\_SCALE

Scale factor for use with an alternate gate resistance model.

### Syntax

MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE\_SINGLE\_SCALE: *factor\_S2*

### Arguments

| Argument         | Description                                                                        |
|------------------|------------------------------------------------------------------------------------|
| <i>factor_S2</i> | Scale factor for the loop mode of the alternate resistance model<br>Default: 0.333 |

### Description

Specifies a scale factor to use for the single-ended mode of the alternate gate resistance model enabled with the MOS\_GATE\_NON\_NEGATIVE\_DELTA\_RESISTANCE command.

### See Also

- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE](#)
- [MOS\\_GATE\\_NON\\_NEGATIVE\\_DELTA\\_RESISTANCE\\_LOOP\\_SCALE](#)
- [MOS\\_GATE\\_DELTA\\_RESISTANCE](#)

---

## MULTIGATE\_MODELS

Enables FinFET modeling.

### Syntax

MULTIGATE\_MODELS: YES | NO

### Arguments

| Argument     | Description              |
|--------------|--------------------------|
| YES          | Enables FinFET modeling  |
| NO (default) | Disables FinFET modeling |

### Description

To use FinFET models in the nxtgrd file, you must specify MULTIGATE\_MODELS: YES in the StarRC command file. FinFET extraction requires an Ultra license. An Information message is generated if FinFET models are detected in the nxtgrd file but the MULTIGATE\_MODELS command is not set to YES.

### Example

To use FinFET models, add the following command to the StarRC command file:

MULTIGATE\_MODELS: YES

### See Also

- [MULTIGATE](#)
- [RPSQ\\_VS\\_SI\\_WIDTH\\_AND\\_LENGTH](#)
- [FinFET Modeling and Extraction](#)

---

## NDM\_DATABASE

Specifies the name of a design library created by the IC Compiler II tool.

### Syntax

NDM\_DATABASE: *design\_library*

### Arguments

| Argument              | Description                                     |
|-----------------------|-------------------------------------------------|
| <i>design_library</i> | The name of the design library<br>Default: none |

### Description

The NDM\_DATABASE command specifies the name of the IC Compiler II design library. This is the name that you use with the `open_lib` command in the IC Compiler II tool.

The NDM\_DATABASE command is mandatory for extraction based on an IC Compiler II flow. You must also use the StarRC BLOCK command to specify the top-level block name used for the `open_block` command in the IC Compiler II tool.

Only gate-level designs are supported.

### See Also

- [BLOCK](#)
- [NDM DESIGN VIEW](#)
- [NDM\\_EXPAND\\_HIERARCHICAL\\_CELLS](#)
- [NDM\\_SEARCH\\_PATH](#)
- [TRANSLATE\\_NDM\\_BLOCKAGE](#)

---

## NDM\_DESIGN\_VIEW

A list of cells for which to use the DESIGN view instead of the FRAME view.

### Syntax

NDM\_DESIGN\_VIEW: *list\_of\_cells*

### Arguments

| Argument             | Description                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------|
| <i>list_of_cells</i> | List of cells for which StarRC uses the DESIGN view during extraction, if available<br>Default: none |

### Description

The argument of the NDM\_DESIGN\_VIEW command is a white-space-delimited list specifying cells for which the StarRC tool uses the DESIGN view instead of the FRAME view during extraction, if it is available.

You can specify this command multiple times in a single command file. The asterisk (\*) and question mark (?) wildcard characters are acceptable.

#### Note:

This command applies to skip cells and their child cells only; the DESIGN view is always used for non-skip-cell masters.

For skip cells not on this list, the FRAME view represents the cell contents during extraction. The FRAME view typically contains all pin shapes and obstructions.

StarRC attempts to translate the DESIGN view for each skip cell on this list. This view contains the actual physical layout, including nonroute layers. Specifying a cell with the NDM\_DESIGN\_VIEW command automatically includes all child cells. If a DESIGN view cannot be found for a cell contained in this list, the StarRC tool issues a warning and reverts to the FRAME view for that cell and all child cells.

### Example

```
NDM_DESIGN_VIEW: cell1 cell2 cell3
NDM_DESIGN_VIEW: cellA cellB cell? *C
NDM_DESIGN_VIEW: *
```

### See Also

- [NDM\\_DATABASE](#)

- [NDM\\_EXPAND\\_HIERARCHICAL\\_CELLS](#)
- [NDM\\_SEARCH\\_PATH](#)
- [NDM DESIGN VIEW](#)

---

## NDM\_EXPAND\_HIERARCHICAL\_CELLS

Enables flattening of cells in an IC Compiler II design library.

### Syntax

NDM\_EXPAND\_HIERARCHICAL\_CELLS: YES | NO

### Arguments

| Argument     | Description                                  |
|--------------|----------------------------------------------|
| YES          | Flattens cells of type NDM_DESIGN_TYPE_MACRO |
| NO (default) | Does not flatten cells                       |

### Description

If the NDM\_EXPAND\_HIERARCHICAL\_CELLS command is set to YES, cells in an IC Compiler II design library whose type is NDM\_DESIGN\_TYPE\_MACRO are flattened. Any other cell type remains a member of the list of skip cells specified by the SKIP\_CELLS command.

The NDM\_EXPAND\_HIERARCHICAL\_CELLS command automatically configures the skip cells list and takes precedence over the SKIP\_CELLS command.

### See Also

- [NDM\\_DATABASE](#)
- [NDM DESIGN VIEW](#)
- [NDM\\_SEARCH\\_PATH](#)
- [TRANSLATE\\_NDM\\_BLOCKAGE](#)

---

## NDM\_SEARCH\_PATH

Paths to search for the reference libraries of an IC Compiler II design library.

### Syntax

NDM\_SEARCH\_PATH: *design\_path*

### Arguments

| Argument           | Description                                            |
|--------------------|--------------------------------------------------------|
| <i>design_path</i> | The search path of the design library<br>Default: none |

### Description

The NDM\_SEARCH\_PATH command is an optional command that specifies a list of paths to search for the reference libraries of the IC Compiler II design library named in the NDM\_DATABASE command.

This command is the equivalent of the `search_path` command in the IC Compiler II tool.

### See Also

- [NDM\\_DATABASE](#)
- [NDM DESIGN VIEW](#)
- [NDM\\_EXPAND\\_HIERARCHICAL\\_CELLS](#)
- [NDM DESIGN VIEW](#)

---

## NET\_SEGMENT\_CUT\_LENGTH

Specifies the length of a cut segment. Valid only for transistor-level flows.

### Syntax

NET\_SEGMENT\_CUT\_LENGTH: *cut\_length*

### Arguments

| Argument          | Description                                            |
|-------------------|--------------------------------------------------------|
| <i>cut_length</i> | Length of cut segment<br>Units: microns<br>Default: 20 |

### Description

The StarRC tool cuts polygons of straight paths along their lengths. You can use the NET\_SEGMENT\_CUT\_LENGTH command to specify the default segment length.

The length of each resulting segment must be at least twice the width of the path. A cut is not made if it would result in a segment that is shorter than twice the width of the path.

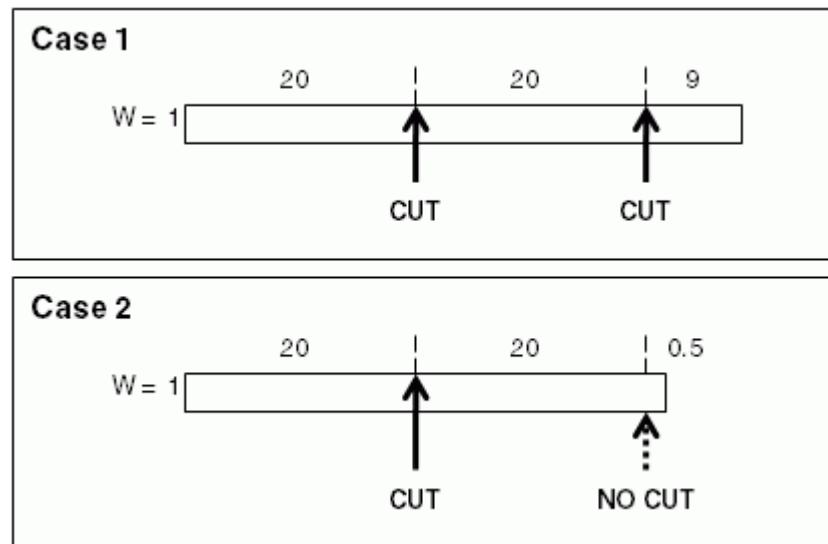
If you enable netlist reduction with the REDUCTION command, then the additional nodes created by the NET\_SEGMENT\_CUT\_LENGTH command are merged based on error control.

Alternatively, you can make layer-based definitions by using the NET\_SEGMENT\_CUT\_LENGTH keyword in the conducting\_layers section of the mapping file. If the cut length is defined in both the StarRC command file and the mapping file, the definition in the mapping file takes precedence.

[Figure 15-17](#) shows two paths that are one micron wide. The default segment cut length is 20 microns. In Case 1, the length of the last segment is 9 microns, which is acceptable because it is more than twice the width of the path. In Case 2, the length of the last segment

is 0.5 microns, which is less than twice the width. Therefore, the second cut is not made and the length of the last segment is 20.5 microns.

Figure 15-17 Cut Segments for NET\_SEGMENT\_CUT\_LENGTH: 20



### See Also

- [REDUCTION](#)
- [conducting\\_layers](#)

---

## NET\_TYPE

Specifies the use of layout or schematic names during data selection.

### Syntax

NET\_TYPE: LAYOUT | SCHEMATIC

### Arguments

| Argument         | Description                                                                     |
|------------------|---------------------------------------------------------------------------------|
| LAYOUT (default) | Specifies that the net names in a NETS: command are referenced to the layout    |
| SCHEMATIC        | Specifies that the net names in a NETS: command are referenced to the schematic |

### Description

Milkyway XTR (extraction) databases contain both layout names and cross-referenced schematic names. This command determines which set of names to use when looking up NETS and POWER\_NETS during data selection.

This command is ignored if the MILKYWAY\_EXTRACT\_VIEW: NO (Hercules flow) or XREF: NO command is specified.

Note:

The NET\_TYPE command identifies only the source of net names for selection in the NETS command. Reported net names are not affected.

### See Also

- [XREF](#)
- [CELL\\_TYPE](#)
- [MILKYWAY\\_EXTRACT\\_VIEW](#)
- [NETS](#)
- [POWER\\_NETS](#)

---

## NETLIST\_CAPACITANCE\_UNIT

Specifies the units used for reporting capacitance values.

### Syntax

NETLIST\_CAPACITANCE\_UNIT: *cap\_unit*

### Arguments

| Argument        | Description                                                                |
|-----------------|----------------------------------------------------------------------------|
| <i>cap_unit</i> | The unit of capacitance for SPEF format<br>Units: farads<br>Default: 1e-15 |

### Description

This command alters the units used for reporting capacitance values in both the header and the body of the output netlist. Applicable only for SPEF netlists.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_COMMENTED\_PARAMS

Lists instance parameters in the netlist as comments.

### Syntax

NETLIST\_COMMENTED\_PARAMS: YES | NO

### Arguments

| Argument     | Description                                                                  |
|--------------|------------------------------------------------------------------------------|
| YES          | Lists instance parameters beginning with a '\$' SPICE comment in the netlist |
| NO (default) | Does not list instance parameters in the netlist                             |

### Description

Specifies whether to generate instance parameters in the netlist beginning with a '\$' SPICE comment. Extra terminals (\$BULK) and \$.model are always included in the netlist.

---

## NETLIST\_COMMENTS\_FILE

Inserts the contents of specified files into the parasitic netlist as comments.

### Syntax

NETLIST\_COMMENTS\_FILE: *file1 file2 ...*

### Arguments

| Argument                 | Description                                                             |
|--------------------------|-------------------------------------------------------------------------|
| <i>file1, file2, ...</i> | File names whose contents are to be appended to the output netlist file |

### Description

Inserts the contents of specified files into the parasitic netlist as comments. This section begins after the netlist HEADER is printed. Each line from the file is inserted as is, prefixed by a comment string (// in SPEF format, \*\* in all other formats). Empty lines are not included.

### See Also

- [NETLIST\\_FILE](#)
- [NETLIST\\_FORMAT](#)

---

## NETLIST\_COMPRESS

Specifies whether to compress a netlist generated from the Galaxy Parasitic Database.  
Valid only in a GPD configuration file.

### Syntax

NETLIST\_COMPRESS: YES | NO

### Arguments

| Argument | Description                   |
|----------|-------------------------------|
| YES      | Compresses the netlist        |
| NO       | Does not compress the netlist |

### Description

Use this command in a Galaxy Parasitic Database (GPD) configuration file to specify whether to compress a netlist created from the GPD.

### See Also

- [GPD](#)
- [The Galaxy Parasitic Database](#)

---

## NETLIST\_COMPRESS\_COMMAND

Pipes the parasitic netlist to an executable or script for compression or postprocessing.

### Syntax

NETLIST\_COMPRESS\_COMMAND: *utility* [*options*] | *script\_file* | *os\_cmd*

### Arguments

| Argument           | Description                                                 |
|--------------------|-------------------------------------------------------------|
| <i>utility</i>     | Executable file to be run on the netlist                    |
| <i>script_file</i> | Script file to be run on the netlist                        |
| <i>os_cmd</i>      | An operating system command string to be run on the netlist |

### Description

The NETLIST\_COMPRESS\_COMMAND command is primarily intended to provide a command for compressing the netlist. For distributed processing runs, file compression is performed on a partition by partition basis. A partition is the portion of the job that StarRC runs on one core.

The command also allows you to specify a script or operating system command. The action is performed on a line by line basis for each partition before the netlist is saved. It is not possible to perform any operations on the entire netlist after it is saved.

OA netlists cannot be postprocessed with this command. For OA netlists, the StarRC tool ignores the command and issues a warning message.

The argument must be one of the following:

- An executable (utility program)

No suffix is appended to the output netlist file (in other words, the file name is not changed to \*.gz or other extensions). If the specified utility is not in a directory specified by the PATH variable, you must specify the full path to the executable.

- The name of a script file, such as a Tcl or Perl script

The StarRC tool does not check for the existence of script files called by the command.

- An operating system command such as awk or sed

The StarRC tool does not check the syntax or validity of operating system commands.

This command can be used with the NETLIST\_POSTPROCESS\_COMMAND command to perform two operations on a netlist. Only one instance of each command is allowed in the StarRC

command file. If both commands are present, the NETLIST\_POSTPROCESS\_COMMAND operation is performed first regardless of the order in the StarRC command file.

For example, you might want to modify some labels within a netlist file by using a custom script, then save the netlist in compressed form. In this case, you would use the NETLIST\_POSTPROCESS\_COMMAND command to execute a script file, followed by the NETLIST\_COMPRESS\_COMMAND command to call a compression utility such as gzip.

## Examples

To compress the netlist with gzip, use the following command:

```
NETLIST_COMPRESS_COMMAND: /usr/local/bin/gzip
```

To postprocess the netlist to remove backslash characters (to make the netlist file compatible with subsequent tools), process the netlist with a UNIX sed command as follows:

```
NETLIST_COMPRESS_COMMAND: sed 's/\\\\</</g' | sed 's/\\\\>/>/g'
```

## See Also

- [NETLIST\\_POSTPROCESS\\_COMMAND](#)

---

## NETLIST\_CONNECT\_OPENS

Creates a white-space-delimited list of nets for StarRC to connect if found open in the input database.

### Syntax

NETLIST\_CONNECT\_OPENS: *netnames*

### Arguments

| Argument        | Description                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>netnames</i> | Specifies the net names connected by a small resistor to be listed if found open during extraction<br>Default: all nets (*) |

### Description

The NETLIST\_CONNECT\_OPENS command creates a white-space-delimited list of nets for StarRC to connect if found open in the input database. You can specify this command multiple times in a single command file. The asterisk (\*), question mark (?), and exclamation mark (!) wildcards are acceptable.

A small resistor is inserted wherever a physical open is found on any net belonging to this list. This function makes it possible for most timing analyzers to calculate delay, even though the net is not actually connected.

### Example

The following example shows how you can explicitly state that a shorting resistor of a particular value can be used to connect resistively connected groups (RCGs). In this example, the resistor is denoted by R=0.01 ohms and width = 100.

```
NETLIST_CONNECT_OPENS: * !pwr* !gnd*
NETLIST_CONNECT_OPENS: net1 net2 net3 ... netn

*RES
742 6:425 6:970 0.0100000 // $l=174.000 $w=100.000 $lvl = 0
743 6:970 6:1445 0.0100000 // $l=1.37 $w=100.000 $lvl = 0
```

---

## NETLIST\_CONNECT\_SECTION

Specifies whether the \*I, \*P, or \*CONN sections are written to the output file.

### Syntax

NETLIST\_CONNECT\_SECTION: YES | NO

### Arguments

| Argument      | Description                                                     |
|---------------|-----------------------------------------------------------------|
| YES (default) | Writes the *I, *P, or *CONN sections to the output file         |
| NO            | Does not write the *I, *P, or *CONN sections to the output file |

### Description

Applicable for all noncapacitor-only formats, including NETNAME. Setting this command to NO disables the generation of the information normally contained in the \*|I and \*|P or \*CONN sections. This can reduce the netlist size significantly, but most delay calculators and static timing analysis tools require this information.

### See Also

- [NETLIST\\_FORMAT](#)

---

## **NETLIST\_COUPLE\_UNSELECTED\_NETS**

Specifies the netlisting of coupling capacitances of unselected nets.

### **Syntax**

NETLIST\_COUPLE\_UNSELECTED\_NETS: IDEAL | COMPLETE | NO

### **Arguments**

| <b>Argument</b> | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IDEAL           | Coupling capacitances are netlisted from nets specified in a NETLIST_SELECT_NETS command to other nets, but no * NET lines appear for unselected nets.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| COMPLETE        | Coupling capacitances from nets specified by the NETLIST_SELECT_NETS command to other nets are netlisted, and those from other nets have * NET parasitic sections.<br><br>The net type for this option is specified using wildcards with the NETLIST_TYPE command. Coupling capacitances to unselected nets are netlisted if the NETLIST_TYPE: no_couple command is not set for the unselected nets to which the couplings exist. The NETLIST_TYPE: no_couple command option overrides the NETLIST_COUPLE_UNSELECTED_NETS for any unselected nets described in the NETLIST_TYPE command. If NETLIST_TYPE: no_couple is set for certain unselected nets that are coupled to selected nets, neither the unselected nets nor their couplings to selected nets are netlisted. |
| NO (default)    | Couplings to unselected nets are not netlisted                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

### **Description**

Specifies that StarRC creates a netlist from all unselected nets (nets not specified by the NETLIST\_SELECT\_NETS command) or those that are coupled to selected nets (nets specified by the NETLIST\_SELECT\_NETS command).

This is a netlist command. However, for coupled nets to be present in the output, the extraction must be coupled by using the EXTRACTION:RC (or EXTRACTION:C) and COUPLE\_TO\_GROUND: NO commands.

### **See Also**

- [NETLIST\\_SELECT\\_NETS](#)
- [NETLIST\\_TYPE](#)

---

## NETLIST\_DELIMITER

Specifies the instance pin delimiter.

### Syntax

NETLIST\_DELIMITER: : | | . | / | #

### Arguments

| Argument | Description                      |
|----------|----------------------------------|
| :        | (default) Colon (:) character    |
|          | Pipe ( ) character               |
| /        | Slash (/) character              |
| .        | Period (.) character             |
| #        | Pound sign or hash (#) character |

### Description

Sets the instance pin delimiter to be printed in the output parasitic netlist.

---

## **NETLIST\_DEVICE\_LOCATION\_ORIENTATION**

Writes device location information to the netlist.

### **Syntax**

NETLIST\_DEVICE\_LOCATION\_ORIENTATION: YES | NO | COMMENT

### **Arguments**

| Argument     | Description                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------|
| YES          | Device x-location, y-location, and orientation (angle) values are written in the instance section of the netlist           |
| NO (default) | Location and orientation information is not written in the netlist                                                         |
| COMMENT      | The location and orientation information is written to the netlist with a dollar sign (\$) prefix for the parameter labels |

### **Description**

In transistor-level flows, you can set the NETLIST\_DEVICE\_LOCATION\_ORIENTATION command to YES to write device location information into an SPF or NETNAME netlist. The extra information consists of the x and y locations and the angle. The angle is nonzero only for MOS devices.

### **Examples**

The following example shows the netlist appearance when the command is set to NO:

```
MM1 10753:F40289 97802:F40290 10755:F40291 vgnd:F40288 MN ad=5.4p as=9.6p
pd=20.54u ps=40.96u l=0.18u w=20u
```

The following example shows the netlist appearance when the command is set to YES:

```
MM1 10753:F40289 97802:F40290 10755:F40291 vgnd:F40288 MN ad=5.4p as=9.6p
pd=20.54u ps=40.96u l=0.18u w=20u x=-1873.77 y=1789.68 angle=0
```

The following example shows the netlist appearance when the command is set to COMMENT:

```
MM1 10753:F40289 97802:F40290 10755:F40291 vgnd:F40288 MN ad=5.4p as=9.6p
pd=20.54u ps=40.96u l=0.18u w=20u $x=-1873.77 $y=1789.68 $angle=0
```

### **See Also**

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_ECO\_FILE

Specifies the name of the incremental SPEF netlist generated during ECO extraction.

### Syntax

NETLIST\_ECO\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                                                                                                                      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>file_name</i> | The incremental netlist file name<br>Default: <i>block_name_incr.spf</i> , where <i>block_name</i> is the block specified by the <a href="#">BLOCK</a> statement |

---

### Description

During some ECO extraction flows, the StarRC tool can maintain two netlists. The netlist from the most recent full-chip extraction is written to the file specified by the NETLIST\_FILE command. The netlist from the most recent ECO (incremental) extraction is written to the file specified by the NETLIST\_ECO\_FILE command.

This command has an effect only if ECO extraction is enabled by setting the [ECO\\_MODE](#) command to YES and incremental netlist generation is enabled by setting the NETLIST\_INCREMENTAL command to YES.

ECO netlists contain parasitics only for ECO-affected nets and related coupling capacitances. They can be used only by tools that have ECO analysis capability (such as the PrimeTime tool).

Only SPEF files are supported.

### Example

NETLIST\_ECO\_FILE: top\_block.spf

### See Also

- [ECO\\_MODE](#)
- [NETLIST\\_INCREMENTAL](#)
- [Chapter 5, “ECO Extraction”](#)

---

## NETLIST\_FILE

Specifies the name of the file to which the output parasitic netlist is written.

### Syntax

NETLIST\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                                                                                          |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <i>file_name</i> | The generated output file.<br>Default: <i>block_name</i> .spf, where <i>block_name</i> is the block specified by the BLOCK statement |

### Description

If the NETLIST\_FILE command is present, a netlist is generated from the parasitic database and stored in the named file.

If the command is not present, a netlist is not generated.

### Example

NETLIST\_FILE: top\_block.spf

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_FORMAT

Defines the format of the output parasitic netlist.

### Syntax

`NETLIST_FORMAT: SPF | SPEF | OA | NETNAME`

### Arguments

| Argument | Description                                                                                                                                           |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPF      | Supports only EXTRACTION: RC with COUPLE_TO_GROUND: YES   NO.<br>Supports coupling capacitors.                                                        |
| SPEF     | Flexible and compact. All names are mapped internally, reducing netlist size.<br>SPEF prints the D_NET (detailed parasitics) net type in the netlist. |
| OA       | (Transistor-level extraction only) OpenAccess database format                                                                                         |
| NETNAME  | (Transistor-level extraction only) Formats internal node names as <code>netname:0</code> ,<br><code>netname:1</code> , and so on                      |

### Description

The `NETLIST_FORMAT` command specifies the format for the output netlist.

Usage of the `NETLIST_FORMAT` command depends on the type of extraction, as follows:

- Gate-level extraction

The StarRC tool saves parasitic data into the Galaxy Parasitic Database (GPD) by default. The tool does not create an ASCII netlist at the time of the extraction run unless you include the `NETLIST_FORMAT` and `NETLIST_FILE` commands in the command file.

You can create a SPEF netlist from an existing GPD directory by using the `StarXtract -convert_gpd_to_spef` command.

- Transistor-level extraction

Parasitic data from transistor-level extraction is available only in an ASCII netlist.

You can create a new netlist from a completed transistor-level extraction run by using the `StarXtract -cleanN` command. If the initial run generated a netlist in the SPF, OA, or NETNAME formats, you can create a new netlist in any format. However, if the initial run generated a SPEF netlist or did not generate any netlist, you can create only SPEF netlists with the `StarXtract -cleanN` command.

## See Also

- [NETLIST\\_FILE](#)
- [GPD](#)

---

## NETLIST\_GROUND\_NODE\_NAME

Defines the net name used when reporting the capacitance with respect either to noncritical material or to an ITF defined SUBSTRATE.

### Syntax

NETLIST\_GROUND\_NODE\_NAME: *net\_name*

### Arguments

| Argument        | Description                                                                |
|-----------------|----------------------------------------------------------------------------|
| <i>net_name</i> | The name of the net to which the capacitance is to be lumped<br>Default: 0 |

### Description

This command is not valid with SPEF format netlists, because the ground node is not output in SPEF.

If you find a reference to node 0 in your output netlist, it is the location where all noncritical extracted materials are lumped. This includes coupling to ideal ground, or SUBSTRATE in StarRC. The entry for node 0 is the SPICE ground.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_HIER\_PROBE\_NODES

Specifies whether the net hierarchy must be reported in the RC netlist.

### Syntax

NETLIST\_HIER\_PROBE\_NODES: YES | NO

### Arguments

| Argument     | Description                                                                            |
|--------------|----------------------------------------------------------------------------------------|
| NO (default) | Does not report the hierarchy in the output                                            |
| YES          | Reports the hierarchy information <i>cell_inst:text_label</i> in the RC netlist output |

### Description

This command specifies whether the net hierarchy must be reported in the RC netlist.

### Example

```
**|OI (cell_inst : text_label cell_inst text_label
Z 0 x_coord y_coord
*| NET SUM0 0.0128485PF
**|OI ($1I1:ProbeA1 $1I1 ProbeA1 Z 0 459.5 34.5)
R16 $1I1:ProbeA1 SUM0 1.19335 $1 = 38.495 $w = 2 $lvl = 1
```

The text, \$1I1:ProbeA1 is inserted into the output when NETLIST\_HIER\_PROBE\_NODES is set to YES.

---

## NETLIST\_IDEAL\_SPICE\_FILE

Generates an ideal SPICE netlist for use with simulation.

### Syntax

NETLIST\_IDEAL\_SPICE\_FILE: *file\_name*

### Arguments

| Argument         | Description              |
|------------------|--------------------------|
| <i>file_name</i> | Ideal SPICE netlist file |

### Description

This command creates an ideal SPICE netlist for use with simulation.

The NETLIST\_IDEAL\_SPICE\_FILE stops at SKIP\_CELLS boundaries. SKIP\_CELLS and device .SUBCKT statements are included in the netlist as comments to indicate port ordering. For SKIP\_CELLS extractions, the ideal device-level netlists can be provided with SPICE\_SUBCKT\_FILE to order the .SUBCKT ports in the NETLIST\_IDEAL\_SPICE\_FILE. Then the SPICE\_SUBCKT\_FILE can be provided to a simulation tool in combination with the NETLIST\_IDEAL\_SPICE\_FILE for a complete device-level ideal SPICE netlist.

The content of the file depends on the settings of the NETLIST\_PASSIVE\_PARAMS, NETLIST\_MAX\_LINE, NETLIST\_IDEAL\_SPICE\_TYPE, NETLIST\_IDEAL\_SPICE\_HIER, and SUPPORT\_DIFFERENT\_PORTNAME\_NETNAME commands.

If the CALIBRE\_PDBA\_FILE command is used, the NETLIST\_IDEAL\_SPICE\_HIER command is automatically set to NO because the layout netlist cannot be maintained as hierarchical when the StarRC tool integrates DFM properties into the final parasitic netlist.

### See Also

- [NETLIST\\_IDEAL\\_SPICE\\_HIER](#)
- [NETLIST\\_IDEAL\\_SPICE\\_TYPE](#)
- [NETLIST\\_MAX\\_LINE](#)
- [NETLIST\\_PASSIVE\\_PARAMS](#)
- [SPICE\\_SUBCKT\\_FILE](#)
- [SUPPORT\\_DIFFERENT\\_PORTNAME\\_NETNAME](#)

---

## NETLIST\_IDEAL\_SPICE\_HIER

Specifies whether to preserve the original hierarchy when generating the ideal SPICE netlist.

### Syntax

NETLIST\_IDEAL\_SPICE\_HIER: YES | NO

### Arguments

| Argument     | Description                                        |
|--------------|----------------------------------------------------|
| YES          | Preserves the original netlist or layout hierarchy |
| NO (default) | Flattens the ideal netlist                         |

### Description

Specifies whether to preserve the original hierarchy when generating the NETLIST\_IDEAL\_SPICE\_FILE.

When the CALIBRE\_PDBA\_FILE or ICV\_ANNOTATION\_FILE commands are used with the NETLIST\_IDEAL\_SPICE\_HIER command, the StarRC output does not include the DFM properties in a separate device property file.

If the CALIBRE\_PDBA\_FILE command is used, the NETLIST\_IDEAL\_SPICE\_HIER command is automatically set to NO because the layout netlist cannot be maintained as hierarchical when the StarRC tool integrates DFM properties into the final parasitic netlist.

### See Also

- [NETLIST\\_IDEAL\\_SPICE\\_FILE](#)
- [NETLIST\\_IDEAL\\_SPICE\\_TYPE](#)

---

## NETLIST\_IDEAL\_SPICE\_TYPE

Specifies whether to netlist a layout- or schematic-based NETLIST\_IDEAL\_SPICE\_FILE command.

### Syntax

NETLIST\_IDEAL\_SPICE\_TYPE: SCHEMATIC | LAYOUT

### Arguments

| Argument  | Description              |
|-----------|--------------------------|
| SCHEMATIC | Uses schematic net names |
| LAYOUT    | Uses layout net names    |

### Description

The default for XREF:NO is LAYOUT. The default for all other XREF values is SCHEMATIC.

### See Also

- [NETLIST\\_IDEAL\\_SPICE\\_FILE](#)

---

## NETLIST\_INCREMENTAL

Specifies whether to perform full-chip or incremental extraction in the ECO flow. Valid only for IC Compiler II designs.

### Syntax

NETLIST\_INCREMENTAL: YES | NO

### Arguments

| Argument     | Description                                                                              |
|--------------|------------------------------------------------------------------------------------------|
| YES          | Performs incremental extraction on ECO-affected nets, if runtime is improved by doing so |
| NO (default) | Performs full-chip extraction regardless of the number of ECO changes detected           |

### Description

ECO extraction is the technique of performing extraction only on the parts of a design that are different from a reference design. This capability allows efficient evaluation of engineering change orders. During ECO extraction, StarRC evaluates the ECO-affected nets and extracts only those nets if runtime would be reduced by doing so.

By default, the StarRC tool generates a full-chip parasitics netlist regardless of the extent of the ECO changes during that cycle.

If you set the NETLIST\_INCREMENTAL command to YES, the StarRC tool generates an incremental parasitics netlist that includes only the extraction results from the ECO-affected nets.

This feature is available only for IC Compiler II designs.

### See Also

- [ECO\\_MODE](#)
- [NETLIST\\_ECO\\_FILE](#)
- [Chapter 5, “ECO Extraction”](#)

---

## NETLIST\_INPUT\_DRIVERS

Identifies the driving cell models in the SPEF netlist format for each instance pin with the optional \*D statement.

### Syntax

NETLIST\_INPUT\_DRIVERS: YES | NO

### Arguments

| Argument     | Description                                       |
|--------------|---------------------------------------------------|
| YES          | Prints the driving cell model name in the netlist |
| NO (default) | Does not enable the command function              |

### Description

Many static timing tools do not require this information for the inputs, because the loading cap for the net is provided. StarRC does not print the \*D statements for the inputs by default. Use this option to print the models for the input instance pins.

This command is ignored unless you specify the NETLIST\_FORMAT: SPEF command.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_INSTANCE\_SECTION

Specifies whether the instance section is included in the output netlist.

### Syntax

NETLIST\_INSTANCE\_SECTION: YES | NO | SELECTED

### Arguments

| Argument           | Description                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| YES                | Includes all instances                                                                                                            |
| NO                 | Does not generate the instance section                                                                                            |
| SELECTED (default) | Generates the instance section from the connected nets group that is the combination of the NETS and NETLIST_SELECT_NETS commands |

### Description

The NETLIST\_INSTANCE\_SECTION command is valid only when the NETLIST\_FORMAT command is set to SPF, NETNAME, or OA.

Because .SUBCKT statements must be consistent with the instance section, they are also written to the netlist. The .SUBCKT line lists all ports, including ports that are not attached to selected nets.

### Examples

The following examples describe the effects of different combinations of the NETS, XREF, NETLIST\_INSTANCE\_SECTION, and NETLIST\_SELECT\_NETS commands.

#### Example 1

```
NETS: selected
XREF: NO | YES | NETS
NETLIST_SELECT_NETS: *
```

If the NETLIST\_INSTANCE\_SECTION command is set to NO, the netlist does not contain an instance section.

If the NETLIST\_INSTANCE\_SECTION command is set to YES or SELECTED, the instance section contains devices attached to extracted nets.

## Example 2

```
NETS: selected
XREF:COMPLETE
NETLIST_SELECT_NETS:*
```

If the NETLIST\_INSTANCE\_SECTION command is set to NO, the netlist does not contain an instance section.

If the NETLIST\_INSTANCE\_SECTION command is set to YES or SELECTED, all schematic devices are included in the instance section. Nets that are not extracted are ideal nets.

## Example 3

```
NETS: *
XREF:NO|YES|NETS
NETLIST_SELECT_NETS:selected
```

If the NETLIST\_INSTANCE\_SECTION command is set to NO, the netlist does not contain an instance section.

If the NETLIST\_INSTANCE\_SECTION command is set to YES, all devices are included.

If the NETLIST\_INSTANCE\_SECTION command is set to SELECTED, only devices attached to selected nets are included.

## See Also

- [NETLIST\\_FORMAT](#)
- [NETLIST\\_SELECT\\_NETS](#)
- [NETLIST\\_COUPLE\\_UNSELECTED\\_NETS](#)
- [NETS](#)
- [POWER\\_NETS](#)

---

## NETLIST\_LOGICAL\_TYPE

Sets a value to be printed in the `SPEF DESIGN_FLOW NETLIST_TYPE val` header card.

### Syntax

`NETLIST_LOGICAL_TYPE: VERILOG | VHDL87 | VHDL93 | EDIF`

### Arguments

| Argument | Description                                               |
|----------|-----------------------------------------------------------|
| VERILOG  | Verilog is the naming convention used in the SPEF netlist |
| VHDL87   | VHDL87 is the naming convention used in the SPEF netlist  |
| VHDL93   | VHDL93 is the naming convention used in the SPEF netlist  |
| EDIF     | EDIF is the naming convention used in the SPEF netlist    |

### Description

This command specifies which naming convention is used in the creation of the SPEF netlist. It is not required by StarRC but might be necessary for some follow-on tools.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_MAX\_LINE

Maximum number of characters to write on each netlist line.

### Syntax

NETLIST\_MAX\_LINE: *no\_of\_chars*

### Arguments

| Argument           | Description                                                                |
|--------------------|----------------------------------------------------------------------------|
| <i>no_of_chars</i> | Maximum number of characters allowed on each netlist line<br>Default: none |

### Description

This command applies to SPF netlists. The default is to place no limit on the number of characters in a line.

When writing a netlist, the StarRC tool limits line length in the file to the specified number of characters. If needed, the tool continues the text on the following line and writes the “+” continuation tag at the beginning of the second and subsequent lines.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_MERGE\_SHORTED\_PORTS

Removes 0.001-ohm node-sharing resistors and merges node names in the netlist to reduce the file size.

### Syntax

NETLIST\_MERGE\_SHORTED\_PORTS: YES | NO

### Arguments

| Argument     | Description                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------|
| YES          | Instance ports connected by node sharing resistors are replaced by one node in the group                                  |
| NO (default) | Instance nodes are unique and might be connected by node sharing resistors (if there are no physical parasitic resistors) |

### Description

If the NETLIST\_MERGE\_SHORTED\_PORTS command is set to YES, whenever multiple port nodes for a net are connected together by node-sharing shorting resistors, StarRC chooses one node randomly from the group to represent all nodes. StarRC uses this node to replace every node in the group for every electrical element in the netlist including parasitic elements, elements in the instance section, and \* | I occurrences in DSPF.

### Example

NETLIST\_MERGE\_SHORTED\_PORTS: YES

---

## NETLIST\_MINCAP\_THRESHOLD

Sets the minimum capacitance allowed in the RC section of the netlist.

### Syntax

NETLIST\_MINCAP\_THRESHOLD: *capacitance\_value*

### Arguments

| Argument                 | Description                                                                                                         |
|--------------------------|---------------------------------------------------------------------------------------------------------------------|
| <i>capacitance_value</i> | The smallest capacitance to be allowed without merging with another capacitance<br>Units: farad (F)<br>Default: 0.0 |

### Description

Any capacitance below this threshold is merged with another smaller capacitance or larger capacitance in a given net. This is applicable for both coupling and grounded capacitance. The capacitance value cannot be less than 0 (zero).

#### Note:

Capacitance that is below the threshold can remain in the netlist.

When the NETLIST\_MINCAP\_THRESHOLD and COUPLING\_ABS\_THRESHOLD commands are both specified, StarRC applies the COUPLING\_ABS\_THRESHOLD command first.

This command is supported only for transistor-level extraction.

### Example

This sets the threshold level at 1 fF.

NETLIST\_MINCAP\_THRESHOLD: 1e-15

### See Also

- [COUPLING\\_ABS\\_THRESHOLD](#)
- [NETLIST\\_TOTALCAP\\_THRESHOLD](#)

---

## NETLIST\_MINRES\_HANDLING

Specifies how a resistor is handled if it is less than or equal to the specified threshold.

### Syntax

NETLIST\_MINRES\_HANDLING: SHORT | MERGE

### Arguments

| Argument        | Description                                                                                          |
|-----------------|------------------------------------------------------------------------------------------------------|
| SHORT           | Does not preserve point-to-point resistance                                                          |
| MERGE (default) | Merges the resistor with a neighboring resistor if it is in series and is smaller than the threshold |

### Description

This command specifies how a resistor is handled if it is less than or equal to the specified threshold in the NETLIST\_MINRES\_THRESHOLD command. This command is supported only for transistor-level extraction.

- If there is only one resistor in the net, it is not merged or shorted.
- If a resistor is attached to a probe node (or \*P or \*I), that terminal is attached to “keep nodes” and should not be affected.
- The NETLIST\_MINRES\_HANDLING command does not preserve point-to-point resistance.
- The NETLIST\_MINRES\_HANDLING command ensures that no resistors exist with their terminals shorted.
- You can specify either SHORT or MERGE in the NETLIST\_MINRES\_HANDLING command, but not both. If you specify both, the second one overrides the first one.
- The NETLIST\_MINRES\_HANDLING:MERGE command does not work on resistor nodes with more than 3 branches. In other words, only series merging is supported. However, the command works on all resistor nodes and shorts the nodes appropriately.
- When the NETLIST\_MINRES\_HANDLING:MERGE command is specified, the capacitance on the reduced node is moved to the smaller resistor.

### See Also

- [NETLIST\\_MINRES\\_THRESHOLD](#)

---

## NETLIST\_MINRES\_THRESHOLD

Merges or shorts all resistances in the netlist less than or equal to the specified threshold.

### Syntax

NETLIST\_MINRES\_THRESHOLD: *threshold\_value*

### Arguments

| Argument               | Description                                                                                                   |
|------------------------|---------------------------------------------------------------------------------------------------------------|
| <i>threshold_value</i> | Threshold at which all resistances are merged or shorted if less than or equal to this value<br>Default: none |

### Description

This command merges or shorts all resistances in the netlist less than or equal to the specified threshold. This command is supported only for transistor-level extraction.

- You cannot specify a value less than zero.
- This option is governed by the [NETLIST\\_MINRES\\_HANDLING: SHORT | MERGE](#) command.

### Example

NETLIST\_MINRES\_THRESHOLD: 0.001

### See Also

- [NETLIST\\_MINRES\\_HANDLING](#)

---

## NETLIST\_NAME\_MAP

Controls name mapping for SPEF netlists.

### Syntax

NETLIST\_NAME\_MAP: YES | NO

### Arguments

| Argument      | Description                            |
|---------------|----------------------------------------|
| YES (default) | Enables name mapping in SPEF netlists  |
| NO            | Disables name mapping in SPEF netlists |

### Description

The NETLIST\_NAME\_MAP command enables or disables name mapping in SPEF netlists. Name mapping reduces the netlist size for big designs by replacing long strings with short codes to indicate nets.

If name mapping is enabled, the netlist file contains a section that lists the name codes, as follows:

```
*NAME_MAP
*3 VDD
*8 X1
*9 X2
...
```

Disabling name mapping greatly increases the netlist size.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_NODE\_SECTION

Generates the \*|S or \*N statements in the output netlist.

### Syntax

NETLIST\_NODE\_SECTION: YES | NO

### Arguments

| Argument     | Description                                                  |
|--------------|--------------------------------------------------------------|
| YES          | Generates * S or *N statements in the output netlist         |
| NO (default) | Does not generate * S or *N statements in the output netlist |

### Description

This command generates the \*|S or \*N statements in the output netlist.

Using the default setting of NO greatly reduces the netlist size and is useful for most postextraction flows.

Specify this command with netlist formats SPF or SPEF.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_NODENAME\_NETNAME

Retains a net name for one of the subnodes of a nonport net.

### Syntax

NETLIST\_NODENAME\_NETNAME : YES | NO

### Arguments

| Argument     | Description                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Retains net names for one of the subnodes of a nonport net. Names the subnodes with the net name without the pin delimiter and positive integer |
| NO (default) | Does not retain subnode names                                                                                                                   |

### Description

The NETLIST\_NODENAME\_NETNAME command retains a net name for one of the subnodes of a nonport net. StarRC chooses one subnode from a nonport (internal) net and converts it to a net name.

This command is valid for all settings of the NETLIST\_FORMAT command and is particularly useful for the standard parasitic format. However, parasitic netlist reading tools that adhere strictly to the SPEF standard might issue errors. To avoid SPEF netlist reading errors, set the NETLIST\_NODENAME\_NETNAME command to NO.

#### Note:

Do not use a probe name specified in a probe text file that is the same as a net name. In that case, the two nodes are electrically shorted.

If a net has a top-level port node, for example, \*|P (DSPF) or \*P (SPEF), the NETLIST\_NODENAME\_NETNAME : YES command does not rename or generate a node for that net.

When a net has at least one \*|S node (DSPF) or \*N node (SPEF), one of those \*|S or \*N is renamed to match the \*|NET or \*D\_NET net name.

A node is never a candidate for renaming if it is from one of the following categories:

---

### **Node**

---

|                  |                             |
|------------------|-----------------------------|
| Instance port    | * I (DSPF) or *I (SPEF)     |
| Top-level port   | * P (DSPF) or *P (SPEF)     |
| Observation port | ** O (DSPF) or **O (SPEF)   |
| Probe text       | ** OP (DSPF) or **OP (SPEF) |

---

If a net contains no \*|S or \*N subnodes but has at least two \*|P or \*|I nodes, then a new node is generated whose name matches the net name.

If a net is floating (no \*|P or \*|I nodes) or dangling (only one \*|P or \*|I node), then no resistor is generated and no node is renamed.

### **Example**

```
NETLIST_NODENAME_NETNAME: NO
* | NET x0.x38.n15 0.000900241PF
* | I (x0.x38.M2|DRN x0.x38.M2 DRN B 0 -492.5 11) //
$lx=-492.5 $ly=4.5 $urx=-489.5 $ury=17.5 $lvl=7
* | I (x0.x38.M1|DRN x0.x38.M1 DRN B 0 -489.5 11)//
$lx=-489.5 $ly=11 $urx=-489.5 $ury=11 $lvl=7
Cg1 x0.x38.M2|DRN 0 2.85306e-16
R1 x0.x38.M2|DRN x0.x38.M1|DRN 0.001 $l=3 $w=10 $lvl=7
```

Example continues ...

```
NETLIST_NODENAME_NETNAME: YES
* | NET x0.x38.n15 0.000900241PF
* | I (x0.x38.M2|DRN x0.x38.M2 DRN B 0 -492.5 11) //
$lx=-492.5 $ly=4.5 $urx=-489.5 $ury=17.5 $lvl=7
* | I (x0.x38.M1|DRN x0.x38.M1 DRN B 0 -489.5 11) //
$lx=-489.5 $ly=11 $urx=-489.5 $ury=11 $lvl=7
* | S (x0.x38.n15 -492.5 11)//
$lx=-492.5 $ly=4.5 $urx=-489.5 $ury=17.5 $lvl=7
Cg1 x0.x38.M2|DRN 0 2.85306e-16
R1 x0.x38.n15 x0.x38.M1|DRN 0.001 $l=3 $w=10 $lvl=7
R2 x0.x38.n15 x0.x38.M2|DRN 0.001 $l=3 $w=10 $lvl=7
```

### **See Also**

- [NETLIST\\_FORMAT](#)

---

## **NETLIST\_PARASITIC\_RESISTOR\_MODEL**

Writes parasitic resistor model names into the parasitic netlist.

### **Syntax**

NETLIST\_PARASITIC\_RESISTOR\_MODEL: YES | NO

### **Arguments**

| Argument     | Description                                              |
|--------------|----------------------------------------------------------|
| YES          | Writes model information to the parasitic netlist        |
| NO (default) | Does not write layer or model information to the netlist |

### **Description**

This command writes resistor models into the parasitic netlist for use by simulators.

The model name is based on the database layer from which the resistor was extracted. If you want to use the same model for more than one nxtgrd file layer, map the corresponding database layers to the same model in the mapping file.

If a nonphysical resistor is introduced into the netlist, it is not generated in the netlist.

If the NETLIST\_PARASITIC\_RESISTOR\_MODEL command is set to YES and you do not specify a resistor model in the mapping file for a layer, no model name is written into the netlist for resistors in that layer and the StarRC tool issues warning messages.

### **Examples**

The mapping file uses the following syntax:

```
Conducting Layers
 database_layer GRD_layer RPSQ = value MODEL = model_name
 database_layer GRD_layer MODEL = model_name
 database_layer GRD_layer MODEL = model_name RPSQ = value
Via_layers
 database_layer GRD_layer RPV=value AREA=value MODEL=model_name
 database_layer GRD_layer MODEL=model_name
 database_layer GRD_layer MODEL=model_name RPV=value
 AREA=value
```

## Example 1

```
TCAD_GRD_FILE: process.nxtgrd
NETLIST_PARASITIC_RESISTOR_MODEL: YES
NETLIST_TAIL_COMMENTS: YES

conducting_layers
metal2 m2 rpsq=0.02 model=M2R
metal1 m1 rpsq=0.02 model=M1R
poly PO1 rpsq=10 model=PR
pgate PO1 rpsq=10 model=PR
ngate PO1 rpsq=10 model=PR
```

## Example 1 Final Netlist

```
* | NET IN2 0.0253958PF
* | I (xSD_11/M1:GATE xSD_11/M1 GATE I 2e-14 -5.1 29.8)
* | P (IN2 B 0 -30.8 7)
* | I (xSD_11/M6:GATE xSD_11/M6 GATE I 2e-14 -5.1 -7.2)
* | I (xSD_11/M2:GATE xSD_11/M2 GATE I 2e-14 -12.1 29.8)
* | I (xSD_11/M5:GATE xSD_11/M5 GATE I 2e-14 -12.1 -7.2)
Cg3 xSD_11/M1:GATE 0 1.0243e-15
C4 IN2:1 xSD_11/SN_22:1 6.78185e-17
Cg5 IN2:1 0 6.15523e-15
Cg6 IN2 0 3.06582e-15
C7 xSD_11/M6:GATE xSD_11/SN_22:1 2.54972e-16
Cg8 xSD_11/M6:GATE 0 3.78517e-16
Cg9 xSD_11/M2:GATE 0 8.35609e-16
C10 xSD_11/M5:GATE xSD_11/M6:DRN 2.46994e-16
Cg11 xSD_11/M5:GATE 0 6.85312e-16
Cg12 IN2:3 0 1.05002e-14
R3 xSD_11/M1:GATE IN2:1 PR r=130.419 $l=21.8 $w=1 $lvl=3
R4 IN2:1 IN2:2 M2R r=0.0701772 $l=7 $w=2.56 $lvl=1
R5 IN2:1 xSD_11/M6:GATE M1R r=121.333 $l=15.2 $w=1 $lvl=2
R6 IN2 IN2:2 M2R r=0.102702 $l=15.2 $w=3.6 $lvl=1
R7 IN2:2 IN2:3 VIAR r=0.0237957 $a=2.56 $lvl=10
R8 xSD_11/M2:GATE IN2:3 PR r=130.419 $l=21.8 $w=1 $lvl=3
R9 xSD_11/M5:GATE IN2:3 PR r=121.333 $l=15.2 $w=1 $lvl=3
```

## See Also

- [NETLIST\\_TAIL\\_COMMENTS](#)
- [REDUCTION](#)

---

## **NETLIST\_PASSIVE\_PARAMS**

Specifies the generation of passive device parameters in the parasitic or ideal netlist.

### **Syntax**

NETLIST\_PASSIVE\_PARAMS: YES | NO

### **Arguments**

| Argument     | Description                                                    |
|--------------|----------------------------------------------------------------|
| YES          | Generates the passive device parameters in the netlist         |
| NO (default) | Does not generate the passive device parameters in the netlist |

### **Description**

Selects a format for the designed R, L, or C devices in the parasitic netlist instance section (NETLIST\_FILE) and the ideal netlist (NETLIST\_IDEAL\_SPICE\_FILE).

The following is an example of the default format for an RLC instance netlist. This format does not require a SPICE model for these devices for simulation:

- R11 R11:A R11:B 1000 \$.model=Rdev \$BULK=VDD  
C12 C12:A C12:B 1e-12 \$.model=Cdev  
L13 L13:A L14:B 10 \$.model=Ldev

If the NETLIST\_PASSIVE\_PARAMS: YES command is set, the format changes to include any properties you calculated in the Hercules or IC Validator runset as well as the standard device extraction properties always calculated by Hercules or IC Validator.

Nonstandard properties such as capacitor length and width are always generated as comments in the netlist, because they are not SPICE-model-compatible. Similarly, the BULK terminals on ideal RLC devices are always generated as comments in the netlist.

### **Example**

The following is an example of the NETLIST\_PASSIVE\_PARAMS: YES format:

```
R11 R11:A R11:B Rdev r=1000 l=10u w=1000um
 $BULK=VDD
C12 C12:A C12:B Cdev c=1e-12 area=100p pj=40u
 $l=10u $w=10u
L13 L13:A L13:B Ldev l=10
```

## See Also

- [NETLIST\\_FILE](#)
- [NETLIST\\_INSTANCE\\_SECTION](#)
- [NETLIST\\_IDEAL\\_SPICE\\_FILE](#)

---

## NETLIST\_POSTPROCESS\_COMMAND

Specifies a method to modify the netlist.

### Syntax

NETLIST\_POSTPROCESS\_COMMAND: *script\_file* | *os\_cmd*

### Arguments

| Argument           | Description                                                 |
|--------------------|-------------------------------------------------------------|
| <i>script_file</i> | Script file to be run on the netlist                        |
| <i>os_cmd</i>      | An operating system command string to be run on the netlist |

### Description

The NETLIST\_POSTPROCESS\_COMMAND command gives you the flexibility to modify an ASCII netlist by operating on it with a script or operating system command. The action is performed on a line by line basis before the netlist is saved.

For distributed processing runs, actions specified by the NETLIST\_POSTPROCESS\_COMMAND command occur on a line by line basis separately for each partition. A partition is a portion of the job that StarRC runs on one core. When the run is complete, the netlist portions are combined with no further processing. It is not possible to perform any operations on the entire netlist after it is saved.

OA netlists cannot be postprocessed with this command. For OA netlists, the StarRC tool ignores the command and issues a warning message.

The argument must be one of the following:

- The name of a script file, such as a Tcl or Perl script
  - The StarRC tool does not check for the existence of script files called by the command.
- An operating system command such as `awk` or `sed`
  - The StarRC tool does not check the syntax or validity of operating system commands.

This command can be used with the NETLIST\_COMPRESS\_COMMAND command to perform two operations on a netlist. Only one instance of each command is allowed in the command file. If both commands are present, the NETLIST\_POSTPROCESS\_COMMAND operation is performed first regardless of the command order.

For example, you might want to modify some labels within a netlist file by using a custom script and then save the netlist in compressed form. In this case, you would use the NETLIST\_POSTPROCESS\_COMMAND command to execute a script file followed by the NETLIST\_COMPRESS\_COMMAND command to call a compression utility such as gzip.

## Example

### Operating System Command Example

The following command edits the netlist to replace all instances of the string Cg at the beginning of a line with Cgnd:

```
NETLIST_POSTPROCESS_COMMAND: sed 's/^Cg/Cgnd/g'
```

### See Also

- [NETLIST\\_COMPRESS\\_COMMAND](#)
- [Distributed Processing](#)

---

## NETLIST\_POWER\_FILE

Controls the file name given to the file created by `POWER_EXTRACT:RONLY`, which contains the power rail resistor values.

### Syntax

`NETLIST_POWER_FILE: file_name`

### Arguments

| Argument               | Description                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| <code>file_name</code> | The file name of a netlist (for power) with resistors only<br>Default: none |

### Description

This command should be used only with the `POWER_EXTRACT:RONLY` command.

### See Also

- [POWER\\_EXTRACT](#)

---

## NETLIST\_PRECISION

Specifies the number of significant digits reported for resistance values, capacitance values, and node coordinates in StarRC output netlists and in files generated by the PLACEMENT\_INFO\_FILE command.

### Syntax

NETLIST\_PRECISION: *sig\_digits*

### Arguments

| Argument          | Description                                              |
|-------------------|----------------------------------------------------------|
| <i>sig_digits</i> | The number of significant digits to report<br>Default: 6 |

### Description

The NETLIST\_PRECISION command sets the number of significant digits to use for reporting certain values.

The setting applies as follows:

- The affected output values are resistances, capacitances, and node coordinates.
- The affected output files are StarRC output netlists and files generated by the PLACEMENT\_INFO\_FILE command.

Trailing zeros are not displayed. Node coordinate precision might be limited by the information available in the design database.

### Examples

The following netlist excerpt shows output with the default of 6:

```
* | NET x0/n36 0.0115668PF
* | I (x0/x37:A x0/x37 A B 0 -479 23.5)
Cg3 x0/n36:1 0 2.60786e-15
Cg4 x0/x37:A 0 6.68525e-15
R4 x0/n36:1 x0/n36:2 12.5053
R5 x0/n36:1 x0/n36:4 0.001
```

The NETLIST\_PRECISION: 10 command results in the following output:

```
* | NET x0/n36 0.01156682055PF
* | I (x0/x37:A x0/x37 A B 0 -479 23.5)
Cg3 x0/n36:1 0 2.607864308e-15
Cg4 x0/x37:A 0 6.6852465e-15
R4 x0/n36:1 x0/n36:2 12.50529289
R5 x0/n36:1 x0/n36:4 0.001000000047
```

## See Also

- [NETLIST\\_FORMAT](#)
- [PLACEMENT\\_INFO\\_FILE](#)

---

## NETLIST\_PRINT\_CC\_TWICE

Specifies whether to generate the reciprocal coupling capacitor twice in the netlist.

### Syntax

NETLIST\_PRINT\_CC\_TWICE: YES | NO

### Arguments

| Argument     | Description                                                              |
|--------------|--------------------------------------------------------------------------|
| YES          | Generates the reciprocal coupling capacitor twice in the netlist         |
| NO (default) | Does not generate the reciprocal coupling capacitor twice in the netlist |

### Description

The second capacitor is reported as a comment so that the netlist remains accurate for standard simulation and timing analysis. This command is used when the netlist format is specified as STAR or NETNAME.

## Example

```
NETLIST_PRINT_CC_TWICE: NO
*|NET NETA 0.0010000PF
*I (NETA:F1 I0 A I 0 485.5 11)
*I (NETA:F2 I1 Z O 0 483.5 11)
*I (NETA:F2 I1 Z O 0 483.5 11)
R1 NETA:F1 NETA:F2 12.43
C1 NETA:F1 0 6e-15
C2 NETA:F2 0 3.5e-15
C3 NETA:F1 NETB:F1 5e-16
*|NET NETB 0.007000PF
*|P (NETB B 0 32.5 8.3)
*I (NETB:F1 I32 B I 0 554.3 12)
RNETB NETB:F1 1032
C4 NETB 0 5e-15
C5 NETB:F1 0 1.5e-15
NETLIST_PRINT_CC_TWICE: YES
*|NET NETA 0.0010000PF
*I (NETA:F1 I0 A I 0 485.5 11)
*I (NETA:F2 I1 Z O 0 483.5 11)
R1 NETA:F1 NETA:F2 12.43
C1 NETA:F1 0 6e-15
C2 NETA:F2 0 3.5e-15
C3 NETA:F1 NETB:F1 5e-16
*|NET NETB 0.007000PF
*|P (NETB B 0 32.5 8.3)
*I (NETB:F1 I32 B I 0 554.3 12)
RNETB NETB:F1 1032
C4 NETB 0 5e-15
C5 NETB:F1 0 1.5e-15
*C6 NETB:F1 NETA:F1 5e-16
```

## See Also

- [COUPLE\\_TO\\_GROUND](#)
- [NETLIST\\_FORMAT](#)

---

## NETLIST\_REMOVE\_DANGLING\_BRANCHES

Removes dangling RC branches in the netlist.

### Syntax

NETLIST\_REMOVE\_DANGLING\_BRANCHES: YES | NO

### Arguments

| Argument     | Description                       |
|--------------|-----------------------------------|
| YES          | Removes dangling branches         |
| NO (default) | Does not remove dangling branches |

### Description

The NETLIST\_REMOVE\_DANGLING\_BRANCHES command removes dangling RC branches in the netlist. While doing so, it moves the capacitors—both Cg and Cc—on dangling branches to an appropriate location in the RC network.

The NETLIST\_REMOVE\_DANGLING\_BRANCHES command

- Cannot be used for SPEF netlists
- Does not affect point-to-point resistance between \*Ps or \*Is, because the removed RC branch would be dangling
- Does not work on open nets that have multiple RCgs

Note:

The possibility of having a dangling RC branch when the REDUCTION command is set to YES, HIGH, or NO\_EXTRA\_LOOPS is low, because the reduction operation eliminates them.

### See Also

- [NETLIST\\_MINRES\\_HANDLING](#)
- [NETLIST\\_MINRES\\_THRESHOLD](#)
- [REDUCTION](#)

---

## NETLIST\_RENAME\_PORTS

Specifies the global renaming scheme for ports.

### Syntax

NETLIST\_RENAME\_PORTS: XY | *delimiter*

### Arguments

| Argument                   | Description                                                                                   |
|----------------------------|-----------------------------------------------------------------------------------------------|
| XY                         | Adds a suffix based on the instance port cell location                                        |
| XYI                        | Adds a suffix based on the instance port cell location and a randomly generated short integer |
| <i>delimiter</i> (default) | Specifies a delimiter string to use for naming<br>Default: _ (single underscore character)    |

### Description

The NETLIST\_RENAME\_PORTS command specifies the global renaming scheme for multiple port instances, which can exist when any of the following commands are used:

- SHORT\_PINS: NO
- INSTANCE\_PORT: NOT\_CONDUCTIVE
- INSTANCE\_PORT: CONDUCTIVE SUFFIXED MULTIPLE

The default scheme to name multiple ports is a single underscore character with sequential numbering. You can replace the default delimiter with a different character or string. The XY mode names each port with the cell local coordinates of the instance port interaction point, in nanometers. The XYI mode also uses the coordinates, but adds a short integer to avoid name duplication when the port has an identical location on different layers.

### Examples

The examples in this section use the same test case with different invocations of the NETLIST\_RENAME\_PORTS command. In all cases, the following commands are also used in the command file:

```
NETLIST_FORMAT: SPEF
INSTANCE_PORT: CONDUCTIVE MULTIPLE SUFFIXED
NETLIST_NAME_MAP: NO
```

The INSTANCE\_PORT command preserves multiple instances for the same port. The NETLIST\_NAME\_MAP:NO command is used for clarity in these examples but does not affect the operation of the NETLIST\_RENAME\_PORTS command.

### **NETLIST\_RENAME\_PORTS: (with no argument or not used at all)**

```
*PORTS
VDD O
///*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_1 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD_2 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD_1 B *C 29.5200 13.9400 *D INV_Q// $llx-29.5200 ...
```

### **NETLIST\_RENAME\_PORTS:#**

```
*PORTS
VDD O
///*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD#1 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD#2 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD#1 B *C 29.5200 13.9400 *D INV_Q// $llx-29.5200 ...
```

### **NETLIST\_RENAME\_PORTS:\_snpsindex\_**

```
*PORTS
VDD O
///*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_snpsindex_1 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD_snpsindex_2 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD_snpsindex_1 B *C 29.5200 13.9400 *D INV_Q// $llx-29.5200 ...
```

**NETLIST\_RENAME\_PORTS: XY**

```
*PORTS
VDD_x200y13910 O
/*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_x10000y10500 B *C 19.1800 13.9400 *D INV_Q// $1lx=19.1800 ...
*I X1:VDD_x0y10500 B *C 9.18000 13.9400 *D INV_Q// $1lx=9.18000 ...
*P VDD_x200y13910 O *C 0.200000 13.9100 // $1lx=13.4400 ...
*I X2:VDD_x0y10500 B *C 29.5200 13.9400 *D INV_Q// $1lx-29.5200 ...
```

**NETLIST\_RENAME\_PORTS: XYI**

```
*PORTS
VDD_x200y13910_7302 O
/*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_x10000y10500_46410 B *C 19.1800 13.9400 *D INV_Q// $1lx=19.1800
*I X1:VDD_x0y10500_55973 B *C 9.18000 13.9400 *D INV_Q// $1lx=9.18000 ...
*P VDD_x200y13910_7302 O *C 0.200000 13.9100 // $1lx=13.4400 ...
*I X2:VDD_x0y10500_5375 B *C 29.5200 13.9400 *D INV_Q// $1lx-29.5200 ...
```

**See Also**

- [INSTANCE\\_PORT](#)
- [SHORT\\_PINS](#)
- [NETLIST\\_NAME\\_MAP](#)

---

## NETLIST\_RESISTANCE\_UNIT

Specifies the units used for reporting resistance values in both the header and the body of the output netlist.

### Syntax

NETLIST\_RESISTANCE\_UNIT: *res\_unit*

### Arguments

| Argument        | Description                                                      |
|-----------------|------------------------------------------------------------------|
| <i>res_unit</i> | Resistance unit in an SPEF netlist<br>Units: ohm<br>Default: 1.0 |

### Description

This command specifies the units used for reporting resistance values in both the header and the body of SPEF netlists.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_SELECT\_NETS

Specifies a subset of extracted nets to netlist.

### Syntax

NETLIST\_SELECT\_NETS: *net\_names*

### Arguments

| Argument         | Description                                           |
|------------------|-------------------------------------------------------|
| <i>net_names</i> | List of nets to be netlisted<br>Default: all nets (*) |

### Description

This command specifies a subset of the extracted nets to include in the output netlist. Wildcards “\*”, “!”, and “?” are supported. The same selection rules detailed in the NETS command reference page also apply to this command.

The StarRC tool issues a warning message if a net marked for netlisting with the NETLIST\_SELECT\_NETS command was not extracted because it was not specified by the NETS command or because it does not exist.

### See Also

- [NETLIST\\_TYPE](#)
- [NETS](#)

---

## NETLIST\_SIM\_OPTIONS

Enables you to specify simulation commands for downstream tools.

### Syntax

NETLIST\_SIM\_OPTIONS: *text\_line*

### Arguments

| Argument         | Description                                          |
|------------------|------------------------------------------------------|
| <i>text_line</i> | A line of text to be written into the output netlist |

### Description

This command allows you to specify lines of text to be written directly into the output netlist. Each occurrence of the NETLIST\_SIM\_OPTIONS command writes a line of text, in the order specified. Typical usage of this feature is to set simulation options to be interpreted by downstream analysis tools.

### Example

The following commands add three lines to the output netlist:

```
NETLIST_SIM_OPTIONS: .param cflag=0
NETLIST_SIM_OPTIONS: .param rflag=0
NETLIST_SIM_OPTIONS: .option scale=0.9
```

The output netlist might look something like this. Note the customized lines in the file:

```
** TCAD_GRD_FILE process.nxtgrd
** TCAD_TIME_STAMP Tue Nov 27 15:29:49 2007
** TCADGRD_VERSION 62
...
.param cflag=0
.param rflag=0
.option scale=0.9
*| GROUND_NET 0
*| NET BOUNDBUF_N_838 0.0735652PF
*| I (cg/p0849A134:Z cg/p0849A134 Z O O -314.774 -248.309)
...
```

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_SMC\_FORMULA

This command specifies the use of formulas for writing RC values in the output netlist in a simultaneous multicorner flow. Valid only for transistor-level flows.

### Syntax

NETLIST\_SMC\_FORMULA: YES | NO

### Arguments

| Argument     | Description                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------|
| YES          | A single final netlist is generated that contains RC values written as formulas using the corner names as variables |
| NO (default) | One or more final netlist files are generated as defined by the SELECTED_CORNERS command                            |

### Description

In a simultaneous multicorner flow, the NETLIST\_SMC\_FORMULA command specifies that the output should be a single netlist containing RC values written as formulas that use the corner names as variables. Any simulation tool that reads the netlist file can set the corner name variables to 1 or 0 to calculate the RC values for a specific corner. The allowable netlist formats for this option are SPF, NETNAME, and OA.

### Example

When the NETLIST\_SMC\_FORMULA: YES command is used in a flow in which three corners are defined (NOM\_T1, NOM\_T2, and RCMAX\_T3), the SPF file output contains lines similar to the following example:

```
* | NET A '0.63*NOM_T1+0.65*NOM_T2+0.75*RCMAX_T3'PF
Cg1 A:1 0 3.6e-17*NOM_T1+4.07e-17*NOM_T2+4.09e-17*RCMAX_T3
R162 A:1 A:2 4.8*NOM_T1+4.9*NOM_T2+4.8*RCMAX_T3
```

### See Also

- [SIMULTANEOUS\\_MULTI\\_CORNER](#)
- [SELECTED\\_CORNERS](#)
- [NETLIST\\_FORMAT](#)

---

## NETLIST\_SUBCKT

Specifies whether to write the .SUBCKT and .ENDS statements in an SPF netlist.

### Syntax

NETLIST\_SUBCKT: YES | NO

### Arguments

| Argument      | Description                                     |
|---------------|-------------------------------------------------|
| YES (default) | Writes the .SUBCKT and .ENDS statements         |
| NO            | Does not write the .SUBCKT and .ENDS statements |

### Description

Specifies whether to write the .SUBCKT and .ENDS statements in standard parasitic format (SPF) netlists.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_TAIL\_COMMENTS

Writes geometric information about parasitic resistors as comments in the netlist.

### Syntax

NETLIST\_TAIL\_COMMENTS: YES | NO

### Arguments

| Argument     | Description                                                      |
|--------------|------------------------------------------------------------------|
| YES          | Writes resistor geometric information as comments in the netlist |
| NO (default) | Disables resistor tail comments in the netlist                   |

### Description

The NETLIST\_TAIL\_COMMENTS command controls whether geometric information about parasitic resistors is added to the netlist output.

If you set the NETLIST\_TAIL\_COMMENTS command to YES, the allowed settings for the REDUCTION and POWER\_REDUCTION commands are NO, LAYER, LAYER\_NO\_EXTRA\_LOOPS, and TOPOLOGICAL.

### Layer Maps in the Netlist

For device-level extraction, the netlist contains a LAYER\_MAP section, which contains both the design database layer names and the ITF layers names. Multiple design layer names can map to the same ITF layer.

An example of a LAYER\_MAP section is as follows:

```
*LAYER_MAP
*0 SUBSTRATE
*1 M1 ITF=M1
*2 M2 ITF=METAL_2
*3 M3 ITF=METAL_3
*4 POLY1 ITF=POLY
*5 POLY2 ITF=POLY
...
```

Layer names might be generated names. Extra layers are formed when there are database layers at the diffusion level or below that share a contact. For example, if the runset contains the line shown in the example, then the LAYER\_MAP section contains an extra layer called nsd:psd or psd:nsd, which becomes the lower terminal layer of the diffCon via resistors.

## Example

```
CONNECT metall nsd psd BY diffCon
```

The SPF and NETNAME netlist formats present geometric information as follows, for the example of a conductor resistor R3 and a via resistor R4:

```
R3 n1:3 n1:43 132.4 $l=12.3 $w=0.45 $lvl=3
R4 n1:3 n1:4 1.2 $a=0.6332 $lvl=14 $vdx=0.05 $vdy=0.05
R5 M2M1_22_24:1 M2M1_22_24 22.37 $l=0.5 $w=0.22 $lvl=4 $m=2
```

The SPEF netlist format presents geometric information as follows:

```
3 n1:3 n1:43 132.4 // $l=12.3 $w=0.45 $lvl=3
4 n1:3 n1:4 1.2 // $a=0.6332 $lvl=14 $vdx=0.05 $vdy=0.05
```

Reported parameters include the following:

- The \$lvl parameter is a number that appears in the LAYER\_MAP section of the netlist, just after the header. The number is associated with a retained mapping file layer name.
- The \$vdx and \$vdy parameters are via x and y dimensions before merge.
- The \$m parameter is the mask number, if mask-aware extraction is performed. The mask number can come from the design database or from the layer mapping file.
- The \$w and \$l parameters are the resistor dimensions.

The \$w and \$l parameters reported in tail comments for MOS transistor source and drain regions are not meaningful because these diffusions are extracted in a complex mesh structure.

## See Also

- [NETLIST\\_FILE](#)
- [NETLIST\\_FORMAT](#)
- [POWER\\_REDUCTION](#)
- [REDUCTION](#)
- [CAPACITOR\\_TAIL\\_COMMENTS](#)

---

## NETLIST\_TIME\_UNIT

Specifies the units for reporting delay values in the SPEF netlist.

### Syntax

NETLIST\_TIME\_UNIT: *time\_unit*

### Arguments

| Argument         | Description                                                                          |
|------------------|--------------------------------------------------------------------------------------|
| <i>time_unit</i> | Specifies the unit for delay in the SPEF netlist<br>Units: seconds<br>Default: 1e-09 |

### Description

This command specifies the units for reporting delay values in the header and body of the netlist. You can use this command only with a SPEF netlist.

### See Also

- [NETLIST\\_FORMAT](#)

---

## NETLIST\_TOTALCAP\_THRESHOLD

Specifies the capacitance threshold that determines whether a net is reported in the netlist.

### Syntax

NETLIST\_TOTALCAP\_THRESHOLD: *capacitance\_value*

### Arguments

| Argument                 | Description                                                                                                                                                                                       |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>capacitance_value</i> | Threshold below which the net is treated as ideal and there is not any D_NET or * NET in the parasitic netlist. The capacitance value cannot be less than 0.<br>Units: farads (F)<br>Default: 0.0 |

### Description

This command specifies the capacitance threshold that determines whether a net is reported in the netlist. If the total capacitance is below the specified threshold, then the net is treated as “ideal” and there is no D\_NET or \*|NET in the parasitic netlist.

This command is supported only for transistor-level extraction.

### Example

The following example sets the ideal threshold at 0.5 fF:

```
NETLIST_TOTALCAP_THRESHOLD: 0.5e-15
```

---

## NETLIST\_TYPE

Specifies parasitics to be generated for nets specified in the NETLIST\_SELECT\_NETS command.

### Syntax

NETLIST\_TYPE: [no\_couple] [R | CG | CC | RCG | RCC] *wildcard\_net\_names*

### Arguments

| Argument                  | Description                                                                                                                                                                                                                                                                                                                  |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| no_couple                 | No coupling capacitance is netlisted from other Cc/RCc nets to the listed <i>wildcard_net_names</i> nets. When unspecified, any extracted coupling capacitance is netlisted from other Cc/Rcc/RLc nets to the listed <i>wildcard_net_names</i> nets. This argument is valid only when used with NETLIST_TYPE: R, Cg, or RCg. |
| R                         | Resistance only                                                                                                                                                                                                                                                                                                              |
| CG                        | Lumped capacitance to ground only                                                                                                                                                                                                                                                                                            |
| CC                        | Coupled capacitance                                                                                                                                                                                                                                                                                                          |
| RCG                       | Resistance plus lumped capacitance to ground                                                                                                                                                                                                                                                                                 |
| RCC (default)             | Resistance plus coupled capacitance                                                                                                                                                                                                                                                                                          |
| <i>wildcard_net_names</i> | Name or a list of names to which the specified type of netlist is to be applied. Wildcards are allowed in this list.                                                                                                                                                                                                         |

### Description

The last NETLIST\_TYPE specified for a particular net or net group overrides previous NETLIST\_TYPE specifications for the same net. When NETLIST\_TYPE is not specified, parasitics are generated according to the EXTRACTION and COUPLE\_TO\_GROUND settings.

If a net is specified with NETLIST\_TYPE CC or RCC, couplings between that net and all other selected nets are included in the netlist regardless of the NETLIST\_TYPE of those other selected nets unless no\_couple is specified for those other selected nets.

### See Also

- [NETLIST\\_COUPLE\\_UNSELECTED\\_NETS](#)
- [NETLIST\\_SELECT\\_NETS](#)

---

## NETLIST\_UNSCALED\_COORDINATES

Specifies the output of scaled or unscaled coordinates.

### Syntax

NETLIST\_UNSCALED\_COORDINATES: YES | NO

### Arguments

| Argument     | Description                           |
|--------------|---------------------------------------|
| YES          | Writes unscaled values in the netlist |
| NO (default) | Writes scaled values in the netlist   |

### Description

When this command is set to YES, the value specified in the MAGNIFICATION\_FACTOR command is used to unscale the layout coordinates for \*|I, \*|P, \*|S, and INSTANCE\_PORT:NOT\_CONDUCTIVE port names.

### See Also

- [INSTANCE\\_PORT](#)
- [MAGNIFICATION\\_FACTOR](#)

---

## NETLIST\_USE\_M\_FACTOR

Specifies whether to use the magnification factor from the schematic netlist to calculate device properties.

### Syntax

NETLIST\_USE\_M\_FACTOR: YES | NO

### Arguments

| Argument      | Description                                                      |
|---------------|------------------------------------------------------------------|
| YES (default) | Calculates device properties with a magnification factor         |
| NO            | Does not calculate device properties with a magnification factor |

### Description

When this command is set to YES, it uses the magnification factor from the schematic netlist to calculate device properties.

#### Note:

The NETLIST\_USE\_M\_FACTOR command has an effect only when the XREF:COMPLETE command is also specified. The value is ignored for any other setting.

### See Also

- [XREF](#)

---

## NETS

Specifies a list of nets for StarRC to extract.

### Syntax

`NETS: net1 net2 ...`

### Arguments

| Argument                   | Description                                                        |
|----------------------------|--------------------------------------------------------------------|
| <code>net1 net2 ...</code> | Nets to be extracted, separated by spaces<br>Default: all nets (*) |

### Description

By default, the StarRC tool extracts all nets. This is the behavior when the `NETS` command is not used in an extraction run.

However, if you use the `NETS` command even one time, the tool only extracts nets that are specified in a `NETS` command, regardless of whether the command is included in the StarRC command file or is written in a separate file. If the `NETS` command appears multiple times, the list of nets to extract is a cumulative list based on the contents of all of the `NETS` commands.

You can also use the `NETS_FILE` command to specify a file that contains a list of `NETS` commands. The effect of the `NETS` command is the same whether it appears directly in the command file or in a separate file.

The following usage notes apply:

- Wildcards are supported: asterisk (\*) for all values, question mark (?) for a single character, and exclamation mark (!) for negation.
- The !\* argument is invalid at all times, whether used alone or with other arguments.
- If a `NETS:*` command appears anywhere within the StarRC command file or in a file specified with the `NETS_FILE` command, and other `NETS` commands are also included, all nets are extracted and the other `NETS` commands are ignored.

If the names originate from a hierarchical netlist, they must be fully flattened with the appropriate hierarchical separator, as defined by the `HIERARCHICAL_SEPARATOR` command. Additionally, any reserved character from the input database must be included in this list to tag the literal use of special characters such as the `BUS_BIT` delimiter. Names must be case-sensitive only in accordance with the value of the `CASE_SENSITIVE` command. The

input database case is always preserved in the output netlist, regardless of the case used in this list.

StarRC does not alter the net information in the input database except to flatten names as required. It is important to understand how the place and route tool handles names. If possible, extract names directly from the input database.

## Examples

The following commands extract nets that begin with A or B:

```
NETS: A*
NETS: B*
```

The following commands extract nets that begin with A, but no nets beginning with AD unless they begin with ADQ:

```
NETS: A*
NETS: !AD*
NETS: ADQ*
```

## Verilog Example

The following example demonstrates the extraction of names from a hierarchical Verilog and assumes that the place and route tool remembered to handle special characters in the Verilog identifiers.

```
module low (A , Y) ;
 input A ;
 output Y ;
 wire n1 ;
 INVX1 X0 (.A(A) , .Y(n1)) ;
 INVX1 X1 (.A(n1) , .Y(Y));
endmodule

module mid (IN , OUT) ;
 input IN ;
 output OUT ;
 wire \instA/din[1] , net2 ;
 low U0/instA (.A(IN) , .Y(\instA/din[1])) ;
 INVX1 U1 (.A(\instA/din[1]) , .Y(net2)) ;
 INVX1 U2 (.A(net2) , .Y(OUT)) ;
endmodule

module top (SIG, SAME) ;
 input SIG ;
 output SAME ;
 wire routel ;
 mid U10 (.IN(SIG) , .OUT(routel));
 mid U11 (.IN(routel) , .OUT(SAME));
endmodule
```

Assume that the `HIERARCHICAL_SEPARATOR:/` and `BUS_BIT: []` commands are used. To extract `instA/din[1]` from instance `U10` in block `top`, specify the following:

```
NETS: U10/instA\ /din\[1\]
```

To extract `n1` from `U0/instA` of `U10` in block `top`, specify the following:

```
NETS: U10/U0\ /instA/n1
```

## See Also

- [BUS\\_BIT](#)
- [CASE\\_SENSITIVE](#)
- [HIERARCHICAL\\_SEPARATOR](#)
- [NET\\_TYPE](#)
- [NETS\\_FILE](#)

---

## NETS\_FILE

Specifies one or more files containing a series of NETS commands.

### Syntax

NETS\_FILE: *file1* [*file2... fileN*]

### Arguments

| Argument                               | Description                             |
|----------------------------------------|-----------------------------------------|
| <i>file1</i> [ <i>file2... fileN</i> ] | Files containing the NETS command lines |

### Description

This command specifies one or more files containing a series of NETS commands.

### Example

In this example, the myNets file contains the following lines:

```
NETS: neta1 neta2
NETS: netb1 netb2 netb3
NETS: clock1
```

The following command specifies that the myNets file contains the series of NETS commands:

```
NETS_FILE: myNets
```

### See Also

- [BUS\\_BIT](#)
- [CASE\\_SENSITIVE](#)
- [HIERARCHICAL\\_SEPARATOR](#)
- [NETS](#)

---

## NONCRITICAL\_COUPLING\_REPORT\_FILE

Specifies the file name for the noncritical coupling report.

### Syntax

NONCRITICAL\_COUPLING\_REPORT\_FILE: *output\_file*

### Arguments

| Argument           | Description                                                         |
|--------------------|---------------------------------------------------------------------|
| <i>output_file</i> | File name for the noncritical coupling report file<br>Default: none |

### Description

Report file generation is supported in the Milkyway flow. The format of the file includes the following:

- A comment at the top of the file refers to the corresponding SPEF file name, *prefix* command, and *suffix* command.
- The report lists all coupling capacitances from noncritical nets to critical nodes, in reverse order from the .spef output. For example, if the SPEF lists the first line shown in the following, the report output lists what is shown on the second line.

SPEF:  
14 A B/SYNOPSYS\_INCONTEXT\_b 1.0e-15

Report file:  
14 B/SYNOPSYS\_INCONTEXT\_b A 1.0e-15

- The command works with NETLIST\_NAME\_MAP:YES | NO for net name mapping of noncritical net names.

Do not specify the same name for the report file name with either the NETLIST\_FILE or COUPLING\_REPORT\_FILE commands. Otherwise, StarRC generates an error message and stops.

Retaining coupling capacitances between the top-level parent routing and the skip cell child net routing exists for the Milkyway flow using the SPEF netlist format.

Only SPEF format is supported. If the user-specified netlist is not SPEF, StarRC issues a warning message.

## See Also

- [NETLIST\\_FORMAT: SPEF](#)
- [COUPLE\\_NONCRITICAL\\_NETS](#)
- [COUPLE\\_NONCRITICAL\\_NETS\\_PREFIX](#)
- [RING\\_AROUND\\_THE\\_BLOCK](#)
- [SKIP\\_CELLS\\_COUPLE\\_TO\\_NET](#)
- [ZONE\\_COUPLE\\_TO\\_NET](#)

---

## NUM\_CORES

Specifies the number of cores used for distributed processing.

### Syntax

NUM\_CORES: *number\_of\_cores*

### Arguments

| Argument               | Description                       |
|------------------------|-----------------------------------|
| <i>number_of_cores</i> | The number of cores<br>Default: 1 |

### Description

The NUM\_CORES command enables distributed processing for extraction and specifies the number of cores to use.

If you specify a value greater than the number of cores available, the StarRC tool issues a warning and uses as many cores as available.

### See Also

- [STARRC\\_DP\\_STRING](#)

---

## OA\_BUS\_BIT

Specifies the delimiters used during OpenAccess view creation for bus bits or iterated instances.

### Syntax

OA\_BUS\_BIT: {} | [] | () | <>

### Arguments

| Argument | Description                                                                                                               |
|----------|---------------------------------------------------------------------------------------------------------------------------|
| { }      | Characters used as the bus bit or iterated instance delimiters; do not insert spaces between the characters in the string |
| [ ]      |                                                                                                                           |
| ( )      | Default: the delimiters specified by the BUS_BIT command                                                                  |
| <>       |                                                                                                                           |

### Description

The OA\_BUS\_BIT command specifies the delimiters used during OA view creation for bus bits or iterated instances. Use this command if you need different bus bit delimiters than those specified by the BUS\_BIT command for the original database and ASCII flow. The OA\_BUS\_BIT command applies to view creation, port annotation, and schematic view annotation.

You must use the BUS\_BIT command to specify the bus bit delimiters for the original database before using the OA\_BUS\_BIT command. If you use the OA\_BUS\_BIT command alone, the StarRC tool issues a warning message and does not change the bus bit delimiters in the OpenAccess view.

### Example

In the following example, bus name a(2) becomes a<2> in the OA view:

```
BUS_BIT: ()
OA_BUS_BIT: <>
```

### See Also

- [BUS\\_BIT](#)

---

## OA\_CDLOUT\_RUNDIR

Specifies the path to the CDL out directory created by the Virtuoso tool.

### Syntax

OA\_CDLOUT\_RUNDIR: *dir\_path*

### Arguments

| Argument        | Description                                    |
|-----------------|------------------------------------------------|
| <i>dir_path</i> | Path to the CDL out directory<br>Default: none |

### Description

This command specifies the path to the CDL out directory created by the Virtuoso tool. The StarRC tool reads files in the ihnl subdirectory of the CDL out directory.

The goal is to obtain the original schematic names of nets in the OpenAccess view, because some names might have changed during the netlisting process. The StarRC tool reads information in the files related to schematic names and the design hierarchy; all other information is ignored.

The argument can be either a relative path or absolute path. Ending the path with a slash character (/) is optional.

### Example

OA\_CDLOUT\_RUNDIR: /path/

### See Also

- [NETLIST\\_FORMAT: OA](#)

---

## OA\_CELL\_NAME

Specifies the name of the OpenAccess (OA) cell for which the parasitic view is written.

### Syntax

OA\_CELL\_NAME: *cell\_name*

### Arguments

| Argument         | Description                                                   |
|------------------|---------------------------------------------------------------|
| <i>cell_name</i> | Cell name<br>Default: the name specified in the BLOCK command |

### Description

The OA\_CELL\_NAME command specifies the name of the OpenAccess (OA) cell for which the parasitic view is written by StarRC.

### Example

OA\_CELL\_NAME: vco

### See Also

- [OA\\_LIB\\_NAME](#)

---

## OA\_DEVICE\_MAPPING\_FILE

Specifies the mapping file to link database and OA device names.

### Syntax

OA\_DEVICE\_MAPPING\_FILE: *file\_path*

### Arguments

| Argument         | Description                                      |
|------------------|--------------------------------------------------|
| <i>file_path</i> | OA device mapping file location<br>Default: none |

### Description

The OA\_DEVICE\_MAPPING\_FILE command specifies a device mapping file that maps ideal and parasitic devices in the StarRC parasitic output to the corresponding device symbols in the OpenAccess symbol libraries. This file contains an entry for every ideal and parasitic device model that exists in the parasitic output. It also provides the ability to remap standard StarRC DSPF device property names to user-specified property names.

For an example of the OpenAccess device mapping file, see [OpenAccess Mapping Files](#).

### Example

OA\_DEVICE\_MAPPING\_FILE: /path/oa\_device\_map

### See Also

- [NETLIST\\_FORMAT: OA](#)
- [OA\\_LAYER\\_MAPPING\\_FILE](#)

---

## OA\_LAYER\_MAPPING\_FILE

Specifies the mapping file to link database and OA layer names.

### Syntax

OA\_LAYER\_MAPPING\_FILE: *file\_path*

### Arguments

| Argument         | Description                                     |
|------------------|-------------------------------------------------|
| <i>file_path</i> | OA layer mapping file location<br>Default: none |

### Description

The OA\_LAYER\_MAPPING\_FILE command specifies a layer mapping file that maps StarRC runset layers from the StarRC mapping file to the corresponding OpenAccess technology file layers. This allows polygons, ports, and subnodes from the parasitic extraction to be stored within the StarRC generated OA parasitic view.

For an example of the OpenAccess layer mapping file, see [OpenAccess Mapping Files](#).

### Example

OA\_LAYER\_MAPPING\_FILE: /path/oa\_layer\_map

### See Also

- [NETLIST\\_FORMAT](#): OA
- [OA\\_DEVICE\\_MAPPING\\_FILE](#)

---

## OA\_LIB\_DEF

Specifies the path to a file that defines libraries referenced in OA\_DEVICE\_MAPPING\_FILE and OA\_LAYER\_MAPPING\_FILE.

### Syntax

OA\_LIB\_DEF: *file\_path*

### Arguments

| Argument         | Description                                              |
|------------------|----------------------------------------------------------|
| <i>file_path</i> | Path to the library definition file<br>Default: lib.defs |

### Description

The OA\_LIB\_DEF command specifies the file name, including the full path, that defines libraries referenced by any of the following commands:

- OA\_DEVICE\_MAPPING\_FILE
- OA\_LAYER\_MAPPING\_FILE
- OA\_LIB\_NAME
- OA\_SKIPCELL\_MAPPING\_FILE
- OA\_PORT\_ANNOTATION\_VIEW
- OA\_PROPERTY\_ANNOTATION\_VIEW

### Example

OA\_LIB\_DEF: /path/lib.def

### See Also

- [OA\\_DEVICE\\_MAPPING\\_FILE](#)
- [OA\\_LAYER\\_MAPPING\\_FILE](#)
- [OA\\_LIB\\_NAME](#)
- [OA\\_SKIPCELL\\_MAPPING\\_FILE](#)
- [OA\\_PORT\\_ANNOTATION\\_VIEW](#)
- [OA\\_PROPERTY\\_ANNOTATION\\_VIEW](#)

---

## OA\_LIB\_NAME

Specifies the name of the OpenAccess (OA) library written by StarRC.

### Syntax

OA\_LIB\_NAME: *library\_name*

### Arguments

| Argument            | Description                   |
|---------------------|-------------------------------|
| <i>library_name</i> | Library name<br>Default: none |

### Description

The OA\_LIB\_NAME command specifies the name of the OpenAccess (OA) library written by StarRC. The path to the library can be specified in the OA\_LIB\_DEF file.

### Example

OA\_LIB\_NAME: PLL

### See Also

- [NETLIST\\_FORMAT:OA](#)
- [OA\\_DEVICE\\_MAPPING\\_FILE](#)
- [OA\\_LAYER\\_MAPPING\\_FILE](#)

---

## OA\_MARKER\_SIZE

Specifies the port or subnode marker size.

### Syntax

OA\_MARKER\_SIZE: *value*

### Arguments

| Argument     | Description                                                       |
|--------------|-------------------------------------------------------------------|
| <i>value</i> | The port or subnode marker size<br>Units: microns<br>Default: 0.1 |

### Description

This command is optional.

### Example

OA\_MARKER\_SIZE: 0.4

### See Also

- [NETLIST\\_FORMAT:OA](#)

---

## OA\_OVERWRITE\_LOCKED\_VIEW

Specifies whether StarRC can overwrite a locked parasitic view in the Virtuoso Integration flow.

### Syntax

OA\_OVERWRITE\_LOCKED\_VIEW: YES | NO

### Arguments

| Argument     | Description                |
|--------------|----------------------------|
| YES          | Overwriting is allowed     |
| NO (default) | Overwriting is not allowed |

### Description

The `OA_OVERWRITE_LOCKED_VIEW` command specifies whether StarRC can overwrite a locked OpenAccess parasitic view in the Virtuoso Integration interface. A parasitic view is locked if it is opened for editing while StarRC is running.

If the command is set to `NO` (the default), StarRC issues an error message if you try to save a parasitic view that is already locked.

You can also enable this feature by using the `overwrite_locked_view` key in the `RCGenParaViewBatch` procedure.

### See Also

- [RCGenParaViewBatch](#)

---

## OA\_PORT\_ANNOTATION\_VIEW

Enables the simulation of a parasitic view generated by the OpenAccess writer.

### Syntax

OA\_PORT\_ANNOTATION\_VIEW: *lib\_name* *cell\_name* *view\_name*

### Arguments

| Argument         | Description                                            |
|------------------|--------------------------------------------------------|
| <i>lib_name</i>  | Library name containing the top-level port information |
| <i>cell_name</i> | Cell name containing the top-level port information    |
| <i>view_name</i> | View name containing the top-level port information    |

### Description

The function accepts a library name, cell name, or view name containing the pins or ports of interest. The specified library, cell, or view name must correspond to the top-level block. If the OpenAccess writer cannot find the named string, a warning is issued and the annotation view is not generated.

The command functions in the following way:

- For each port in the OA\_PORT\_ANNOTATION\_VIEW, there must be a port on the net in the parasitic view with the same name. If the net does not have that same port, a port is created on that net.
- If the port has been created after extraction, there is no annotation for that port.

### Example

This example shows how the schematic view from the specified alib and acell arguments is checked for schematic-only properties. The schematic-only properties are attached from the OpenAccess parasitic view.

```
OA_PORT_ANNOTATION_VIEW: alib acell symbol
```

### See Also

- [NETLIST\\_FORMAT: OA](#)

---

## OA\_PROPERTY\_ANNOTATION\_VIEW

Specifies which schematic library, cell, or view is used to check against ideal devices for schematic-only properties and to attach them into the OpenAccess parasitic view.

### Syntax

`OA_PROPERTY_ANNOTATION_VIEW: [lib] [cell] view_name`

### Arguments

| Argument               | Description                                                                                                                                                                                                                                                                                                                   |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lib</code>       | Checks ideal devices for schematic-only properties in this library                                                                                                                                                                                                                                                            |
| <code>cell</code>      | Checks ideal devices for schematic-only properties in this cell                                                                                                                                                                                                                                                               |
| <code>view_name</code> | A view name in the current extraction library. It can be a library name, cell name, or a view name. A warning is issued <ul style="list-style-type: none"> <li>- If the specified lib, cell, or view cannot be found (cannot be opened).</li> <li>- If you specify more than the allowed names (an invalid value).</li> </ul> |

### Description

Specifies which schematic library, cell, or view is used to check against ideal devices for schematic-only properties and to attach them into the OpenAccess parasitic view.

If some ideal instances cannot be correctly cross-referenced, for example `I0|I1|ld_M21`, the OpenAccess writer does not do a schematic annotation for it, while issuing an internal warning for it as in the following text: "Warning: Instance `I0|I1|ld_M21` can't get property from schematic view as not-XREF-ed"

Only `XREF:YES` and `XREF:COMPLETE` are supported in the schematic view annotation.

### Example

This example shows the specified `alib` and `acell` arguments that are checked for schematic-only properties. The OpenAccess parasitic view is *schematic*.

`OA_PROPERTY_ANNOTATION_VIEW: alib acell schematic`

### See Also

- [XREF](#)
- [NETLIST\\_FORMAT: OA](#)

---

## OA\_PROPMAP\_CASE\_SENSITIVE

Specifies whether the schematic view property annotation is case-sensitive.

### Syntax

```
OA_PROPMAP_CASE_SENSITIVE : YES | NO | MIXED
```

### Arguments

| Argument     | Description                                                          |
|--------------|----------------------------------------------------------------------|
| YES          | Case-sensitive                                                       |
| NO (default) | Not case-sensitive                                                   |
| MIXED        | Lowercase for specific properties; not case-sensitive for all others |

### Description

The `OA_PROPMAP_CASE_SENSITIVE` command affects schematic property annotation with the StarRC OpenAccess file writer.

If you specify the `MIXED` argument, most properties are not case-sensitive. However, the following default properties are forced to be lowercase: `l`, `w`, `as`, `ad`, `ps`, `pd`, `nd`, `nr`, `m`, `area`, and `pj`.

The `OA_PROPMAP_CASE_SENSITIVE` command is similar to the `PROPMAP_CASE_SENSITIVE` option in the `.snps_settings` file. While the `PROPMAP_CASE_SENSITIVE` option is used for the Virtuoso Integration flow, the `OA_PROPMAP_CASE_SENSITIVE` command is used for running StarRC as a standalone tool.

### See Also

- [PROPMAP Case Sensitivity](#)

---

## OA\_REMOVE\_DUPLICATE\_PORTS

Prevents duplication when annotating both the vector-form port names and the expanded port names.

### Syntax

OA\_REMOVE\_DUPLICATE\_PORTS: YES | NO

### Arguments

| Argument     | Description                            |
|--------------|----------------------------------------|
| YES          | Avoids duplicate ports in the OA view  |
| NO (default) | Annotates all the ports in the OA view |

### Description

When your design has nonexpanded port names in the schematic view and expanded BUS port names in the layout, StarRC annotates both port names by default. If you do not want this duplication, set the OA\_REMOVE\_DUPLICATE\_PORTS command to YES.

### Example

If you have OTP<0:1> for the nonexpanded port names in the schematic view and then have OTP<0> and OTP<1> in the layout view, by default, the tool outputs all three ports in the final OA parasitic view:

OTP<0:1>, OTP<0>, OTP<1>

When you set the OA\_REMOVE\_DUPLICATE\_PORTS command to YES, StarRC only outputs the OTP<0> and OTP<1> ports in the netlist.

### See Also

- [OA\\_PORT\\_ANNOTATION\\_VIEW](#)

---

## OA\_REMOVE\_PRIMITIVE\_SPICECARD\_PREFIX

Specifies whether SPICE card prefix characters appearing before the primitives of instance names should be removed.

### Syntax

`OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX: YES | NO | cdl_file`

### Arguments

| Argument        | Description                                                                                                               |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| YES (default)   | Removes the SPICE card prefix characters from the primitives of instance names                                            |
| NO              | Preserves the SPICE card prefix characters in the primitives of instance names                                            |
| <i>cdl_file</i> | The file name of a cdlinclude file that contains the SUBCKT definitions for which the primitive X card should be stripped |

### Description

The `OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX` command specifies whether to remove SPICE card prefix characters before the primitives in ideal instance names. Prefix characters are C, D, M, R, Q, or X.

You can optionally provide the name of a cdlinclude file that contains SUBCKT definitions.

A `?spiceCardStripPrimitiveCallBack` key in a SKILL procedure takes precedence over the `OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX` command.

### Example

If the layout-versus-schematic tool provides instance names as follows:

```
XXI18/XI0/M1
XXI18/XI1/R1
```

After SPICE card prefix removal, the instance names become:

```
XXI18|XI0|1
XXI18|XI1|1
```

### See Also

- [OA\\_REMOVE\\_SPICECARD\\_PREFIX](#)

---

## OA\_REMOVE\_SPICECARD\_PREFIX

For Virtuoso Integration flows, specifies whether the SPICE card prefix in the paths of ideal instance names and net names should be removed by StarRC.

### Syntax

OA\_REMOVE\_SPICECARD\_PREFIX: YES | NO

### Arguments

| Argument      | Description                                                                  |
|---------------|------------------------------------------------------------------------------|
| YES (default) | Removes the prefix characters from the paths of instance names and net names |
| NO            | Preserves the prefix characters in the paths of instance names and net names |

### Description

The OA\_REMOVE\_SPICECARD\_PREFIX command specifies whether to remove SPICE card prefix characters from the paths of ideal instance names. Prefix characters are C, D, M, R, Q, or X.

A ?spiceCardStripinstpathCallBack key in a SKILL procedure takes precedence over the OA\_REMOVE\_SPICECARD\_PREFIX command.

### Example

If the layout-versus-schematic tool provides instance names as follows:

XXI18/XI0/M1  
XXI18/XI1/R1

After SPICE card prefix removal, the instance names become:

I18|I0|M1  
I18|I1|R1

### See Also

- [OA\\_REMOVE\\_PRIMITIVE\\_SPICECARD\\_PREFIX](#)

---

## OA\_SKIPCELL\_MAPPING\_FILE

Specifies a file to define skip cell extraction in an OpenAccess flow.

### Syntax

OA\_SKIPCELL\_MAPPING\_FILE: *oa\_skip\_file*

### Arguments

| Argument            | Description                                                                  |
|---------------------|------------------------------------------------------------------------------|
| <i>oa_skip_file</i> | Name of the cell to skip at a particular hierarchical level<br>Default: none |

### Description

For hierarchical designs, you might only want to extract the top-level design for a parasitic view without extracting the lower-level block to reduce the view generation time. In the ASCII DSPF flow, the **SKIP\_CELLS** command is typically used for pre-extracted blocks. In a DSPF flow, those skip cells specified in a **SKIP\_CELLS** command are listed in the "Instance" section for simulation purposes.

To enable the skipping operation in an OpenAccess parasitic view, you must specify which cell master to use for the skip cell instantiation in the parasitic view. Each specified skip cell has corresponding mapping information for cell instantiation in the parasitic view.

### Example

```
SKIP_CELLS: INV1 INV2
OA_SKIPCELL_MAPPING_FILE: skip_file
```

File *skip\_file* contains the following:

```
INV1 analogLib INV symbol
INV2 analogLib INV symbol
INV3 analogLib INV symbol
```

Even though there might be an *INV3* in the top block, it is not treated as a skip cell because it is not listed in the **SKIP\_CELLS** command.

### See Also

- [NETLIST\\_FORMAT: OA](#)
- [SKIP\\_CELLS](#)

---

## OA\_VIEW\_NAME

Specifies the name of the OpenAccess parasitic view generated by StarRC.

### Syntax

OA\_VIEW\_NAME: *view\_name*

### Arguments

| Argument         | Description                                  |
|------------------|----------------------------------------------|
| <i>view_name</i> | Name of OA parasitic view<br>Default: starrc |

### Description

You can specify the name of the OA parasitic view generated by StarRC. By default, the OA parasitic view name is starrc. This command is optional.

### Example

OA\_VIEW\_NAME: extract\_view

### See Also

- [NETLIST\\_FORMAT:OA](#)

---

## OASIS\_FILE

Specifies OASIS format files to represent part of the physical layout.

### Syntax

OASIS\_FILE: *file1* [*file2*] ...

### Arguments

| Argument                          | Description                       |
|-----------------------------------|-----------------------------------|
| <i>file1</i> [ <i>file2</i> ] ... | OASIS file names<br>Default: none |

### Description

The `OASIS_FILE` command specifies OASIS format files to represent part of the physical layout. You can specify gzip and compressed OASIS files for this command.

For Milkyway designs, this command merges OASIS data into FRAM or CEL views for skip cells to provide a full physical layout representation. For the cells that are defined by both the OASIS file and the FRAM or CEL views, StarRC uses only the pin shapes from the FRAM or CEL view. StarRC replaces any cells in the obstruction section of the FRAM or CEL view with cells of the same name in the OASIS file. If StarRC does not find a matching cell name in the OASIS file, then the FRAM or CEL view continues to be used for that cell.

For LEF/DEF designs, this command merges OASIS data into LEF MACRO definitions for skip cells to provide a full physical layout representation. For the cells that are defined by both the OASIS file and the LEF macro, StarRC uses only the pin shapes from the LEF macro. StarRC replaces any cells in the OBS section of the LEF MACRO with cells of the same name in the OASIS file. If StarRC does not find a matching cell name in the OASIS file for a particular DEF cell placement, then the OBS section of the LEF MACRO continues to be used for that cell.

The `OASIS_FILE` command can be specified multiple times. It must be used with the `OASIS_LAYER_MAP_FILE` command and cannot be used with the `GDS_FILE` command.

### See Also

- [OASIS\\_LAYER\\_MAP\\_FILE](#)
- [SKIP\\_CELLS](#)

---

## OASIS\_LAYER\_MAP\_FILE

Specifies the mapping between the OASIS layer number and layer name in the design database.

### Syntax

`OASIS_LAYER_MAP_FILE: file_name`

### Arguments

| Argument               | Description                                    |
|------------------------|------------------------------------------------|
| <code>file_name</code> | OASIS layer mapping file name<br>Default: none |

### Description

The `OASIS_LAYER_MAP_FILE` command specifies the mapping between the OASIS layer number and layer name in the design database whenever `OASIS_FILE` or `METAL_FILL_OASIS_FILE` is used to import OASIS data into the design database.

#### Note:

You cannot use the `OASIS_LAYER_MAP_FILE` command together with the `GDS_LAYER_MAP_FILE` command.

### OASIS Layer Map File Syntax

The `OASIS` layer map file uses the following syntax:

```
database_layer oasis_layer_number oasis_datatype
 {FLOATING | GROUNDED | IGNORE}
```

*Table 15-7 OASIS Layer Map File Syntax*

| Argument                        | Description                                                                                     |
|---------------------------------|-------------------------------------------------------------------------------------------------|
| <code>database_layer</code>     | The database layer name                                                                         |
| <code>oasis_layer_number</code> | The OASIS layer number                                                                          |
| <code>oasis_datatype</code>     | The OASIS data type. If a data type is not specified, all data types on a given layer are read. |

*Table 15-7 OASIS Layer Map File Syntax (Continued)*

| Argument | Description                                                                                                                                                                        |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FLOATING | Specifies that the corresponding fill layer is to be treated as floating. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored. |
| GROUNDED | Specifies that the corresponding fill layer is to be treated as grounded. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored. |
| IGNORE   | Specifies that the corresponding fill layer is to be ignored. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.             |

All OASIS layers for translation must have an entry in this file and must have a definition in the layout database.

Use the OASIS\_LAYER\_MAP\_FILE command with the following commands:

- OASIS\_FILE, when importing OASIS cells into a Milkyway or LEF/DEF database
- METAL\_FILL\_OASIS\_FILE, when importing metal fill polygons into a Milkyway, LEF/DEF, or Calibre Connectivity Interface database

If you use both the METAL\_FILL\_OASIS\_FILE and OASIS\_FILE commands in a single Milkyway or LEF/DEF run, you should use a single unified layer mapping file for both the OASIS files.

If a layer in the mapping file does not have a corresponding layer in the layout database, StarRC issues an error.

The additional specification of a layer-specific fill-handling keyword is available to allow users to selectively decide how individual metal fill layers are handled during parasitic extraction. This handling is considered only when METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC is set in the StarRC command file. If METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC is set in the StarRC command file, but a fill-handling mode is not specified for a particular layer inside OASIS\_LAYER\_MAP\_FILE, StarRC defaults to FLOATING for that layer. If another setting of METAL\_FILL\_POLYGON\_HANDLING (other than AUTOMATIC) is specified in the StarRC command file, that setting governs the handling of all layers, and any layer-specific mode specifications inside OASIS\_LAYER\_MAP\_FILE are disregarded.

Note that in the given example, handling mode GROUNDED is specified for layer DIFF. If METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC is also specified in the StarRC command file, DIFF is treated as GROUNDED while all other layers are treated as FLOATING. If METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC is not specified, the layer-specific mode

specification for DIFF is disregarded, and the global mode set by METAL\_FILL\_POLYGON\_HANDLING takes precedence.

## Examples

The following example shows how the DIFF layer is assigned to OASIS layer 2 and OASIS datatype 0. Note that this example maps layers from a METAL\_FILL\_OASIS\_FILE and specifies layer-specific fill handling for the DIFF layer.

### *Example 15-11 Layer-Specific Fill Handling*

```
DIFF 2 0 GROUNDED
POLY 7 0
CONT 4 0
METAL1 10 0
METAL1 10 1
METAL1 76 0
VIA1 11 0
METAL2 12 0
```

In the following example, the METAL\_FILL\_POLYGON\_HANDLING: AUTOMATIC command is set in the command file.

### *Example 15-12 Automatic Fill Handling*

```
*layer treated as grounded
DIFF 2 0 GROUNDED
*layer treated as floating
POLY 7 0 FLOATING
*layer governed by default floating mode since mode is unspecified.
METAL1 4 0
```

## See Also

- [OASIS\\_FILE](#)
- [LEF\\_FILE](#)
- [METAL\\_FILL\\_OASIS\\_FILE](#)

---

## OBSERVATION\_POINTS

Specifies cells that are not skipped for reporting observation nodes in the output netlist.

### Syntax

OBSERVATION\_POINTS: *wildcard\_cell\_list*

### Arguments

| Argument                  | Description                                    |
|---------------------------|------------------------------------------------|
| <i>wildcard_cell_list</i> | List of nonskipped cell names<br>Default: none |

### Description

This command generates nodes at nonskipped hierarchical interactions. Observation nodes are reported in the parasitics section of the netlist, allowing these locations to be used during simulation. This command is not supported for LEF/DEF input.

Observation nodes are netlisted as **\*\*O** statements in the SPEF-format netlist and as **\*\*|O** statements in the **DSPF**, **NETNAME**, and **STAR** format netlists. The observation point statement has formatting identical to the **\*I** and **\*|I** statements.

The **OBSERVATION\_POINTS** command works with the **XREF:NO**, **YES**, and **COMPLETE** commands. Only EQUIV points can be selected cells with the **OBSERVATION\_POINTS** command. Schematic names are used with observation node netlisting when the **XREF** command is used. The **CELL\_TYPE** command determines which cell names are applied for selection.

Observation nodes are formed from marker layers. If there are no marker shapes in a selected level of hierarchy, there are no **\*\*O** statements. With the default automatic marker generation, there must be a hierarchical interaction of the routing layers to get a marker shape for **\*\*O** at that level of the hierarchy. You can also control **OBSERVATION\_POINTS** by generating marker shapes with the **MARKER\_GENERATION: USER** command.

The marker shape has nothing to do with either **TEXT\_POLYGON** commands in Hercules runset or marker layers in the mapping file when **MARKER\_GENERATION:** is set to **AUTOMATIC**.

### See Also

- [CELL\\_TYPE](#)
- [MARKER\\_GENERATION](#)
- [XREF](#)

---

## OPERATING\_FREQUENCY

Specifies the frequency used when including substrate effects during high frequency extractions.

### Syntax

OPERATING\_FREQUENCY: *freq*

### Arguments

| Argument    | Description                                                                                                          |
|-------------|----------------------------------------------------------------------------------------------------------------------|
| <i>freq</i> | The frequency used when including substrate effects during high-frequency extractions<br>Units: GHz<br>Default: none |

### Description

The OPERATING\_FREQUENCY command specifies the frequency used when including substrate effects during high-frequency extractions.

### See Also

- [TSV](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [TSV\\_ CELLS](#)
- [Through-Silicon Via Extraction](#)

---

## OPERATING\_TEMPERATURE

Specifies the operating temperature either in the StarRC command file or in a corners file.

### Syntax

OPERATING\_TEMPERATURE: *temperature*

### Arguments

| Argument           | Description                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------|
| <i>temperature</i> | Operating temperature in degrees Celsius<br>Default: none<br>Valid range: -200 to +300 degrees |

### Description

The OPERATING\_TEMPERATURE command can be used in either the StarRC command file or in a corners file.

Use the command as follows:

- In the StarRC command file

The OPERATING\_TEMPERATURE: *temperature* command must be used in the command file if you want to use derating information contained in the nxtgrd file. If the resistance of a layer is changed by the mapping file, the resistance value in the mapping file is used for derating and the ITF value is ignored. The operating temperature is documented in the DESIGN\_FLOW TEMPERATURE header card in the SPEF file.

- In a corners file

For simultaneous multicorner extraction, you can set different operating temperatures for different corners by including an OPERATING\_TEMPERATURE: *temperature* command in each corner definition in the corners file.

### Examples

In this example, rcworst.nxtgrd is extracted at 125° C.

```
OPERATING_TEMPERATURE: 125
TCAD_GRD_FILE: rcworst.nxtgrd
```

### See Also

- [NETLIST\\_FORMAT](#)

- [TCAD\\_GRD\\_FILE](#)
- [CORNERS\\_FILE](#)
- [CORNER\\_TYPE](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)
- [Simultaneous Multicorner Extraction](#)

---

## PIN\_CUT\_THRESHOLD

Takes a long port and splits it into multiple \*|P nodes.

### Syntax

`PIN_CUT_THRESHOLD: distance_in_nm`

### Arguments

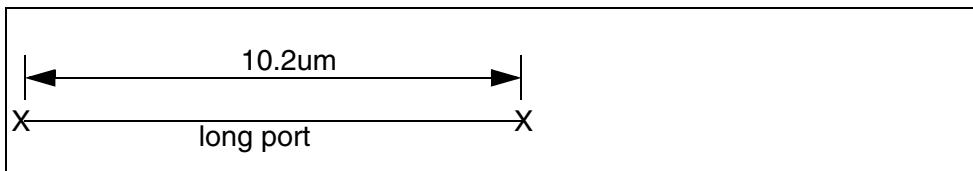
| Argument                           | Description                                                        |
|------------------------------------|--------------------------------------------------------------------|
| <code><i>distance_in_nm</i></code> | Distance at which to cut a long port<br>Units: nm<br>Default: none |

### Description

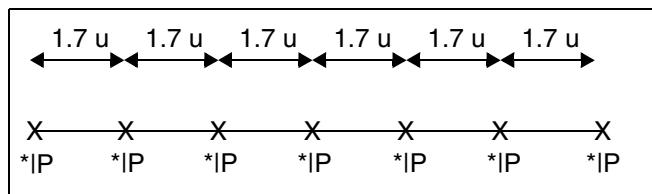
The `PIN_CUT_THRESHOLD` command splits a long port into multiple \*|P nodes. If a port has a continuous Manhattan length less than the `PIN_CUT_THRESHOLD`, StarRC creates one \*|P node for that segment. If a long port is not evenly divisible by the `PIN_CUT_THRESHOLD` value, StarRC breaks the port into symmetric segments, as shown in [Figure 15-18](#)

This command supports only Manhattan port shapes and LEF/DEF, Milkyway, IC Validator, and Hercules flows. It does not support the Calibre Connectivity Interface flow.

*Figure 15-18 Symmetric Division*



With a `PIN_CUT_THRESHOLD` value of 2,000 nm, this 10.2m long port would be cut as follows:



The output in the SPEF netlist would be as follows:

### Example

PIN\_CUT\_THRESHOLD: 500

```
...
*CCONN
*P NET1_x40000y105000 B *C 40.00 105.00
*P NET1_x45000y115500 B *C 45.00 115.50
*P NET1_x50000y115500 B *C 50.00 115.50
*P NET1_x55000y115500 B *C 55.00 115.50
....
*I Level_1/NET_A:PIN_A_x10000y10000 *C 10.00 10.00
*I Level_1/NET_A:PIN_A_x15000y20500 *C 15.00 20.50
*I Level_1/NET_A:PIN_A_x20000y20500 *C 20.00 20.50
*I Level_1/NET_A:PIN_A_x25000y20500 *C 25.00 20.50
```

### See Also

- [INSTANCE\\_PORT](#)

---

## **PIO\_FILE**

Specifies a file containing primary pin direction and loading capacitance.

### **Syntax**

`PIO_FILE: file1 file2 ... fileN`

### **Arguments**

| <b>Argument</b>    | <b>Description</b>                                    |
|--------------------|-------------------------------------------------------|
| <code>fileN</code> | File containing the pin descriptions<br>Default: none |

### **Description**

This command specifies a file containing primary pin direction and loading capacitance. Only applicable for top-level ports. Information contained in `PIO_FILE` is applied to the output netlist. Format is white-space-delimited with one entry per line:

`pin_name IN | OUT | BIDIR [cap cap_value]`

### **Example**

```
IN1 IN
CLK IN
OUT OUT cap 5pf
IN2 IN
OUT2 OUT cap 1e-12
```

### **See Also**

- [NETLIST\\_FORMAT](#)

---

## PLACEMENT\_INFO\_FILE

Specifies the generation of output placement information.

### Syntax

PLACEMENT\_INFO\_FILE: YES | NO

### Arguments

| Argument     | Description                            |
|--------------|----------------------------------------|
| YES          | Creates a placement information file   |
| NO (default) | Does not provide placement information |

### Description

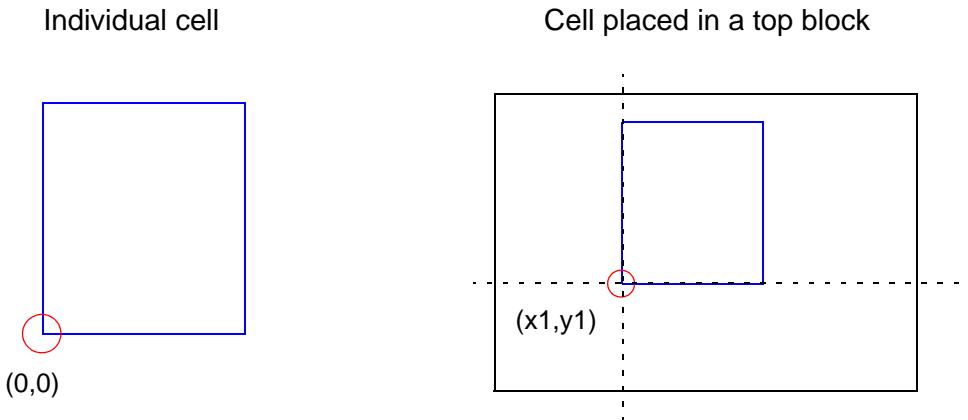
DSPF netlists can be either flat or hierarchical. In a flat extraction, all nodes are shown with respect to origin of the top cell. However, in a hierarchical flow, each node in a hierarchical cell's DSPF is shown with respect to its origin. To map these nodes to the next level of hierarchy, a downstream analysis tool must know the placement of the cell in the next level of hierarchy with rotation and flip attributes.

When you set the `PLACEMENT_INFO_FILE` command to `YES`, the StarRC tool creates a file that contains the following information:

- Angle
- Reflection
- Location of the cell (X and Y coordinates)
- Cell name
- Cross-referenced instance name

You can specify a file name with the `PLACEMENT_INFO_FILE_NAME` command. You can also control the precision of the resistance and capacitance values and the location coordinates in the placement information file with the `NETLIST_PRECISION` command.

The cell location written to the file is relative to the cell origin unless the cell is instantiated in a top design block, in which case the location is relative to the top block origin.

**Figure 15-19 Cell Location**

## Examples

The following is an example of a transistor-level placement file.

```
* PLACEMENT FILE
* VENDOR: "Synopsys Inc."
* PROGRAM: "StarRC"
...
TOP_CELL = STBASE
inst_name cell_name X_Coord Y_Coord Angle reflection
xSD_8/M1 P 8.5 54 0 NO
xSD_8/M2 N 8.5 17 0 NO
xSD_9/M1 P 8.5 54 0 NO
xSD_9/M2 N 8.5 17 0 NO
xSD_10/M1 P 8.5 54 0 NO
xSD_10/M2 N 8.5 17 0 NO
xSD_11/M1 P 15.5 54 0 NO
xSD_11/M2 P 8.5 54 0 NO
xSD_11/M3 P 29.5 54 0 NO
xSD_11/M4 N 29.5 17 0 NO
xSD_11/M5 N 8.5 17 0 NO
xSD_11/M6 N 15.5 17 0 NO
* End of File
```

This is an output example from a placement file. A transistor-level placement file example follows the argument list.

```
* PLACEMENT FILE
* VENDOR "Synopsys, Inc."
* PROGRAM: "StarRC"
...
TOP_CELL = SMALLARRAY
inst_name cell_name X_Coord Y_Coord Angle reflection
xSD_0 STBASE -1925.8 -379 0 NO
xSD_1 STBASE -1797.2 -379 0 NO
xSD_2 STBASE -1668.6 -379 0 NO
xSD_3 STBASE -1540 -379 0 NO
xSD_4 STBASE -1411.4 -379 0 NO
xSD_5 STBASE -1282.8 -379 0 NO
xSD_6 STBASE -1154.2 -379 0 NO
xSD_7 STBASE -1025.6 -379 0 NO
```

## See Also

- [NETLIST\\_PRECISION](#)
- [PLACEMENT\\_INFO\\_FILE\\_NAME](#)

---

## PLACEMENT\_INFO\_FILE\_NAME

Specifies the name of the placement info file.

### Syntax

PLACEMENT\_INFO\_FILE\_NAME: *info\_file*

### Arguments

| Argument         | Description                                                    |
|------------------|----------------------------------------------------------------|
| <i>info_file</i> | The generated placement information file name<br>Default: none |

### Description

Use this command to specify a file name when you create a placement info file by setting the PLACEMENT\_INFO\_FILE command to YES.

### Example

PLACEMENT\_INFO\_FILE\_NAME: top\_block.place

### See Also

- [PLACEMENT\\_INFO\\_FILE](#)

---

## POWER\_EXTRACT

Specifies the extraction of power nets.

### Syntax

```
POWER_EXTRACT: YES | NO | RONLY | DEVICE_LAYERS [NON_DEVICE_LAYERS_ONLY]
```

### Arguments

| Argument                                    | Description                                                                                                                                                                                                                                                                   |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES                                         | Extracts the power nets                                                                                                                                                                                                                                                       |
| NO (default)                                | Does not extract the power nets                                                                                                                                                                                                                                               |
| ROONLY                                      | Extracts the power nets for resistance only. Creates an additional resistor-only netlist when the NETLIST_POWER_FILE command is also used.                                                                                                                                    |
| DEVICE_LAYERS                               | Extracts the resistance and capacitance for the power nets whose layers are specified with the device_layer keyword in a conducting_layers statement in the mapping file. Valid for the Hercules, IC Validator, and Calibre flows.                                            |
| DEVICE_LAYERS<br>NON_DEVICE_LAYERS<br>_ONLY | Extracts the capacitance for the power nets on all layers. Extracts the resistance for the power nets whose layers are specified with the device_layer keyword in a conducting_layers statement in the mapping file. Valid for the Hercules, IC Validator, and Calibre flows. |

### Description

By default, the StarRC tool does not extract the power nets. The power nets are taken into consideration when the signal nets are extracted. However, they are not netlisted and the resulting effect on the signal nets is reported as a grounded capacitance. You can extract power nets by setting the POWER\_EXTRACT command to YES.

The power nets are identified as follows:

- Power nets are identified implicitly in a routed Milkyway database or a LEF/DEF layout description.
- For transistor-level flows, you can specify the power nets to be extracted by using the POWER\_NETS command. If the POWER\_NETS command is not used, the power net definition from the LVS tool is inherited.

The coupling capacitances between the signal and power nets are kept and grounded. The coupling capacitances between power nets are not extracted, and the total capacitance reported for power nets is zero.

Special-purpose options are available, as follows:

- The `RONLY` option

Using the `POWER_EXTRACT:RONLY` command creates an additional resistor-only netlist when the `NETLIST_POWER_FILE` command is used. The signal netlist (the standard netlist) contains resistance and capacitance parasitics of the signal nets. The power netlist contains resistance parasitics of the power nets.

- The `DEVICE_LAYERS` option (transistor-level flows only)

StarRC extracts the resistance and capacitance only from the power nets that contain the `device_layer` keyword within a `conducting_layers` or `via_layers` statement in the mapping file.

This option specifies a special flow in which layers on power nets other than device layers are treated as superconductive layers. During extraction, the StarRC tool connects nodes from superconductor layers to top-level power ports by using shorting resistors. The result in the netlist is one `*|P` entry that connects to millions of device-level interaction nodes.

To simplify processing and reduce the number of resistors on a node, the tool connects the superconductive nodes within a partition, then makes a single connection to a top-level port. You cannot control how the superconductive nodes and ports are connected.

Note the following:

- In your design, power nets on metal layers (superconductive layers) must be directly connected to ports, not connected indirectly through device layers.
- If multiple isolated power straps connect to a top-level port, the node processing procedures might cause the isolated regions to be connected.
- If physical opens exist on power nets in non-device layers, the tool does not recognize them because the node processing procedures mark these layers with the same net ID. As a result, the opens reported for the `DEVICE_LAYERS` option might be different from the opens reported for the `YES` option.
- The `DEVICE_LAYERS NON_DEVICE_LAYERS_ONLY` option (transistor-level flows only)

StarRC extracts the capacitance from power nets on all layers, and both resistance and capacitance from the power nets that contain the `device_layer` keyword within a `conducting_layers` or `via_layers` statement in the mapping file.

## Examples

In the following example, StarRC extracts contact, gate, and diffusion resistance for VDD and VSS power nets based on the `device_layer` keyword in the mapping file. The instance section netlist lists all the devices connected to these power nets along with the signal nets.

```
NETS: (any definition)
POWER_NETS: VDD VSS
POWER_EXTRACT: DEVICE_LAYERS
```

## See Also

- [NETLIST\\_POWER\\_FILE](#)
- [NETS](#)
- [POWER\\_NETS](#)
- [POWER\\_REDUCTION](#)
- [conducting\\_layers](#)
- [via\\_layers](#)

---

## POWER\_NETS

Specifies power nets for special treatment during extraction.

### Syntax

POWER\_NETS: *netnames*

### Arguments

| Argument        | Description                                               |
|-----------------|-----------------------------------------------------------|
| <i>netnames</i> | List of net names to identify power nets<br>Default: none |

### Description

The power nets are implicitly defined in place and route Milkyway and LEF/DEF databases and do not need to be specified in the StarRC command file. If this command is not specified for a Hercules database, StarRC attempts to read the list of power nets from the Hercules runset. An optional command for a Hercules flow extraction.

The \*, ?, and ! wildcards are acceptable in this white-space- delimited list, and case sensitivity is determined by the value of the CASE\_SENSITIVE command.

### See Also

- [CASE\\_SENSITIVE](#)
- [POWER\\_EXTRACT](#)
- [NET\\_TYPE](#)
- [XREF](#)

---

## POWER\_PORTS

Specifies a list of patterns for identifying POWER/GROUND-type ports for SKIP\_CELLS if their types are not explicitly defined in the database and their names are different from the top level POWER\_NETS.

### Syntax

POWER\_PORTS: *wildcard\_list*

### Arguments

| Argument             | Description                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------|
| <i>wildcard_list</i> | List of port names to identify to the power nets<br>Default: List specified by the POWER_NETS statement |

### Description

In LEF/DEF input, the USE POWER and USE GROUND keywords in LEF MACRO PIN definitions identify the usage.

In Milkyway place and route input, the port-type tables defined at stream-in and during BLOCKAGE\_PIN\_VIA\_EXTRACTION determine the usage. Querying the PIN shapes in the FRAM views indicates their usage type. For Hercules or IC Validator XTR view input, none of the ports are marked.

By default, POWER\_PORTS inherits the POWER\_NETS list.

### See Also

- [NETLIST\\_POWER\\_FILE](#)
- [POWER\\_NETS](#)

---

## POWER\_REDUCTION

Determines whether the reduction of extracted power nets occurs during extraction.

### Syntax

POWER\_REDUCTION: NO | LAYER | LAYER\_NO\_EXTRA\_LOOPS | YES

### Arguments

| Argument             | Description                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO                   | Similar to the REDUCTION: NO command, except that the reduction is applied to power nets rather than signal nets                                                                                          |
| LAYER                | Similar to REDUCTION: LAYER command, except that the reduction is applied to the specified power nets rather than signal nets. Reduction is not applied across different conductor layers.                |
| LAYER_NO_EXTRA_LOOPS | Similar to REDUCTION: LAYER_NO_EXTRA_LOOPS command, except that the reduction is applied to the specified power nets rather than signal nets. Reduction is not applied across different conductor layers. |
| YES (default)        | Similar to REDUCTION: YES command, except that the reduction is applied to the specified power nets rather than signal nets                                                                               |

### Description

The POWER\_REDUCTION command does not apply when the POWER\_EXTRACT command is set to YES. The POWER\_REDUCTION command settings take effect only when the POWER\_EXTRACT command is set to RONLY. When the POWER\_EXTRACT: YES command is specified, the power nets are extracted and netlisted in the same manner as normal signals.

### See Also

- [REDUCTION](#)

---

## PRINT\_SILICON\_INFO

Prints resistor silicon width or area in the netlist tail comments.

### Syntax

PRINT\_SILICON\_INFO: YES | NO

### Arguments

| Argument     | Description                                                                   |
|--------------|-------------------------------------------------------------------------------|
| YES          | Prints silicon width \$si_w for conductor resistors and nonphysical resistors |
| NO (default) | Does not print silicon width                                                  |

### Description

Downstream simulation tools sometimes require the resistor silicon width value, which is different from the drawn width. The `PRINT_SILICON_INFO` command prints the silicon width in addition to the drawn width. The extra information is printed in the netlist tail comments, therefore the `NETLIST_TAIL_COMMENTS` command must also be set to `YES`.

StarRC prints the silicon width `$si_w` for conductor resistors and nonphysical resistors in addition to the existing length, width, and process layer values (`$l`, `$w` and `$lvl`). For via resistors, the silicon area `$si_a` is provided after the existing area and process layer values (`$a` and `$lvl`).

The silicon width might be smaller or larger than the drawn width. ITF commands such as the `ETCH` and `ETCH_VS_WIDTH_AND_SPACING` commands affect the silicon width computation. The silicon width is the value used in resistance calculations.

The silicon area `$si_a` for a via resistor is always the same as the area `$a` because via etch is not supported in resistance calculation.

### Errors

The silicon width might be inaccurate when the `RPSQ_VS_WIDTH_AND_SPACING` or `RHO_VS_WIDTH_AND_SPACING` ITF commands are specified. A warning is issued when the `PRINT_SILICON_INFO` command is used with these ITF commands. When this occurs, replace the `RPSQ_VS_WIDTH_AND_SPACING` and `RHO_VS_WIDTH_AND_SPACING` commands with the `RPSQ_VS_SI_WIDTH` and `ETCH_VS_WIDTH_AND_SPACING` command.

The `RPSQ_VS_WIDTH_AND_SPACING` command should not be used to model a scenario where a conductor has different spacing on the left side compared to the right side.

## Example

The following portion of a netlist shows the silicon width value at the end of the tail comments for conductor resistor R41 and the silicon area value at the end of the tail comments for via resistor R42:

```
R41 M15:SRC Y:15 11.6946 $l=0.105 $w=0.153 $lvl=100 $si_w=0.149
R42 Y:12 Y:54 30 $a=0.0081 $lvl=134 $si_a=0.0081
```

The simultaneous multicorner flow can generate either individual netlists for each corner or a single merged netlist. The silicon width is reported in the SMC flow as follows:

- For an individual netlist, the netlist tail comments values come from that single corner.
- For a standard merged netlist, the netlist tail comments values come from the primary corner, which is the first corner in the colon-delimited list in the `SELECTED_CORNERS` command.
- For a merged netlist with resistor temperature coefficient output, the netlist tail comment values come from the corner that contains the `CORNER_TYPE: NOMINAL` designation.

## See Also

- [NETLIST\\_TAIL\\_COMMENTS](#)

---

## PROBE\_TEXT\_FILE

Specifies a file that contains simulation probe points to appear in the StarRC parasitic netlist.

### Syntax

PROBE\_TEXT\_FILE: *file\_name*

### Arguments

| Argument         | Description                                                                                            |
|------------------|--------------------------------------------------------------------------------------------------------|
| <i>file_name</i> | The file containing defined (x, y location) probe points<br>Coordinate units: microns<br>Default: none |

### Description

A probe point is a specific node point in the parasitic node mesh for a particular net.

The coordinates in the probe text file are in microns. A coordinate is an absolute value and is not related to precision. The HALF\_NODE\_SCALE\_FACTOR and the MAGNIFICATION\_FACTOR commands are applied.

A PROBE\_TEXT\_FILE entry is translated and netlisted if the following is true:

- The specified cell reference name is a valid extracted cell. If XREF is in the command file, the cell name must correspond to a valid LVS equivalence point. The CELL\_TYPE command regulates whether the schematic or layout name is used.
- A polygon at the specified layer name exists at the specified coordinate location.
- The probe point falls on a net for which parasitics are included in the netlist.
- The probe point falls on a cross-referenced net within a cross-referenced instance in the XREF: COMPLETE command.

Probe points take the following formats in the output netlist:

- Instance level: \*\*OI

For probe text defined at the instance level, regardless of whether the probe text overlaps any routing or port polygons

- Top level: \*\*OP

For probe text defined at the top level, regardless of whether the probe text overlaps any routing or port polygons

## Examples

[Example 15-13](#) shows the syntax of the probe text file.

### Example 15-13 Probe Text File Syntax

```
CELL cell_name
textstring local_x local_y layername
textstring local_x local_y layername
textstring local_x local_y layername
textstring local_x local_y layername
```

---

| Parameter         | Definition                                                                                                                                                                     |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>textstring</i> | Identifying layout text label for the probe point by which the point is identified in the parasitic netlist.                                                                   |
| <i>local_x</i>    | X-coordinate for the probe point location. The specified coordinate is interpreted local to the specified cell.                                                                |
| <i>local_y</i>    | Y-coordinate for the probe point location. The specified coordinate is interpreted local to the specified cell.                                                                |
| <i>layername</i>  | Database layer name to which the probe point corresponds. The name must correspond to a layer mapped in the <code>conducting_layers</code> section of the StarRC mapping file. |
| <i>cell_name</i>  | Name of the cell master in which the probe point is specified. The <code>CELL_TYPE</code> command regulates whether a layout cell or schematic cell is used.                   |

---

An example of a probe text file is as follows:

```
CELL ADD4_TOP
 PROBE1 10 10 M3
CELL INVX$
 GATE_1 16.3 14.5 GPOLY
 GATE_2 15.3 1.1 GPOLY
```

An example of observation points in a netlist is as follows:

```
**|OP (PROBE1 Z 0 10 10)
**|OI (I1:GATE_1 I1 GATE_1 Z 0 26.3 19.5)
**|OI (I1:GATE_2 I1 GATE_2 Z 0 25.3 6.1)
```

## See Also

- [CELL\\_TYPE](#)

---

## REDUCTION

Specifies parasitic netlist reduction options.

### Syntax

`REDUCTION:HIGH | YES | NO | LAYER | NO_EXTRA_LOOPS | LAYER_NO_EXTRA_LOOPS`

### Arguments

| Argument             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HIGH                 | Performs maximum netlist reduction for device-level parasitic extraction to reduce the runtime of SPICE-level simulators. Parasitic netlists produced with the <code>HIGH</code> setting are 2X to 10X smaller than netlists produced with the <code>YES</code> setting. Runtime is 10 to 20 percent larger relative to the <code>YES</code> setting.                                                                                                                                                                                                           |
| YES (default)        | Performs standard netlist reduction                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| NO                   | Does not perform netlist reduction                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| LAYER                | Similar to <code>YES</code> , but performs reduction on the same layer only                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| NO_EXTRA_LOOPS       | Performs netlist reduction, but ensures that no resistive loops are introduced, for use with delay calculators that cannot interpret resistive loops. The netlist is 10 to 20 percent larger than with the <code>YES</code> setting.<br><br>This option retains loops in the parasitic netlist that result from the layout topology. For example, overlapping metals connected by parallel vias can produce meshes in the parasitic netlist even with the <code>NO_EXTRA_LOOPS</code> setting, because such meshes reflect the physical topology of the layout. |
| LAYER_NO_EXTRA_LOOPS | Similar to <code>NO_EXTRA_LOOPS</code> , but performs reduction on the same layer only                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

### Description

This command specifies the reduction of extracted parasitic devices. The degree of reduction is controlled by the `REDUCTION_MAX_DELAY_ERROR` command. The reduction algorithm preserves point-to-point resistance and total net capacitance.

If you use the PrimeTime tool for timing analysis, use the `NO_EXTRA_LOOPS` setting to simplify the resistor network.

### See Also

- [REDUCTION\\_MAX\\_DELAY\\_ERROR](#)

---

## REDUCTION\_MAX\_DELAY\_ERROR

Specifies the acceptable net delay error tolerance when StarRC performs reduction.

### Syntax

REDUCTION\_MAX\_DELAY\_ERROR: *threshold*

### Arguments

| Argument         | Description                                                               |
|------------------|---------------------------------------------------------------------------|
| <i>threshold</i> | The absolute delay error threshold<br>Units: seconds<br>Default: 1.0 e-12 |

### Description

The absolute delay error between the original and reduced netlists cannot be greater than this threshold.

#### Note:

This option should not be combined with REDUCTION:TOPOLOGICAL because the TOPOLOGICAL mode does not have timing error control.

### See Also

- [REDUCTION](#)

---

## REFERENCE\_DIRECTION

Specifies the reference direction of the process technology.

### Syntax

REFERENCE\_DIRECTION: VERTICAL | HORIZONTAL | NONE | GATE

### Arguments

| Argument       | Description                                            |
|----------------|--------------------------------------------------------|
| VERTICAL       | Reference direction is vertical                        |
| HORIZONTAL     | Reference direction is horizontal                      |
| NONE (default) | Reference direction should be taken from the ITF file  |
| GATE           | Reference direction is determined by device pin layout |

### Description

The `REFERENCE_DIRECTION` statement defines the reference direction for the application of orientation-dependent etch defined by the `ETCH_VS_WIDTH_AND_SPACING` statement in the ITF file.

You can specify the reference direction in the StarRC command file or the ITF file. If the reference direction is specified in both files, the setting in the StarRC command file takes precedence.

### Example

The following example specifies that the reference direction is horizontal:

```
REFERENCE_DIRECTION: HORIZONTAL
```

### See Also

- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [REFERENCE\\_DIRECTION \(ITF command\)](#)

---

## REMOVE\_DANGLING\_NETS

Specifies whether to identify dangling nets and reassign them to ground (effectively removing them).

### Syntax

REMOVE\_DANGLING\_NETS: YES | NO

### Arguments

| Argument     | Description                                                 |
|--------------|-------------------------------------------------------------|
| YES          | Specifies that the netlist should not contain dangling nets |
| NO (default) | Specifies that the netlist should retain dangling nets      |

### Description

Dangling nets are defined as:

- Unrouted database nets (for Milkyway, LEF/DEF, and Calibre Connectivity Interface)
- Nets that have only one connection (for Milkyway, LEF/DEF, Calibre Connectivity Interface, Hercules, and IC Validator flows).

Nets that are connected to a pin port (\*|P) are not eligible for removal. In Hercules flows, ports must be generated during Hercules device connectivity extraction using the CREATE\_PORTS runset command. Otherwise, the StarRC tool does not consider the port connection and the net is removed. Using the REMOVE\_DANGLING\_NETS command does not remove layout material; the objects are assigned to ground.

### See Also

- [REMOVE\\_FLOATING\\_NETS](#)

---

## REMOVE\_DIFFUSION\_GATE\_OVERLAP

Specifies the removal of the gate-diffusion overlap.

### Syntax

REMOVE\_DIFFUSION\_GATE\_OVERLAP: YES | NO

### Arguments

| Argument     | Description                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Removes the gate-diffusion overlap by aligning the diffusion with the gate edges                                                          |
| NO (default) | Keeps the gate-diffusion overlap unchanged. This option is only effective for a trench contact process with raised source and drain etch. |

### Description

In an actual device, the gate polysilicon might overlap with the diffusion layer, and the real diffusion layer is round or diamond-shaped in the corner next to the gate. Process modeling uses a simple effective profile for the device region where the edge of the diffusion shape is exactly aligned with the gate polysilicon shape.

The StarRC field solver implements this model for better accuracy of the capacitance between the sides and top of the gate to the diffusion ( $C_{fi}$ ) and the total gate-to-diffusion capacitance ( $C_f$ ) with the foundry reference data. The `REMOVE_DIFFUSION_GATE_OVERLAP` command specifies whether the gate-diffusion overlap is removed.

---

## REMOVE\_FLOATING\_NETS

Specifies whether to removes nets that have no connection by grounding them.

### Syntax

REMOVE\_FLOATING\_NETS: YES | NO

### Arguments

| Argument     | Description                                                        |
|--------------|--------------------------------------------------------------------|
| YES          | Specifies that the output netlist should not contain floating nets |
| NO (default) | Specifies that the output netlist should retain floating nets      |

### Description

When the REMOVE\_FLOATING\_NETS command is set to YES, the StarRC tool does not completely disregard polygons on floating nets. The goal is to eliminate floating nets from the parasitic netlist while accounting for effects these nets have on real signal nets in the design.

When the REMOVE\_FLOATING\_NETS command is set to YES, StarRC treats the floating nets as noncritical material. In other words, the tool finds the coupling capacitance from signal nets to the noncritical material and then decouples these capacitors. The decoupled capacitance is then added to the ground capacitance of the signal net. The total capacitance of the signal nets accounts for the effects of coupling to floating nets. The floating nets are not shown in the parasitic netlist.

### See Also

- [REMOVE\\_DANGLING\\_NETS](#)

---

## REMOVE\_METAL\_FILL\_OVERLAP

Enables the metal fill reuse flow.

### Syntax

REMOVE\_METAL\_FILL\_OVERLAP: YES | NO

### Arguments

| Argument     | Description                        |
|--------------|------------------------------------|
| YES          | Enables the metal fill reuse flow  |
| NO (default) | Disables the metal fill reuse flow |

### Description

The REMOVE\_METAL\_FILL\_OVERLAP command allows you to reuse metal fill structures in a timing signoff loop to save overall turnaround time.

When extraction is run with reused metal fill, the StarRC tool resolves shorts between metal fill polygons and signal nets by moving the metal fill structures away from the signal nets. The default spacing for the moved metal fills is the SMIN value of the conductor layer in the ITF file. You can optionally define a different spacing by setting the `overlap_fill_spacing` value in the `conducting_layers` statement in the mapping file.

### Example

The following

```
REMOVE_METAL_FILL_OVERLAP: YES
```

### See Also

- [conducting\\_layers](#)
- [The Metal Fill Reuse Flow](#)

---

## REMOVE\_NET\_PROPERTY

Specifies a single property name to indicate to the Milkyway layout database which objects should not be extracted as signal nets. Valid only for Milkyway flows.

### Syntax

REMOVE\_NET\_PROPERTY: *property\_name*

### Arguments

| Argument             | Description                                                                             |
|----------------------|-----------------------------------------------------------------------------------------|
| <i>property_name</i> | Specifies that nets with this property are not included in the netlist<br>Default: none |

### Description

These objects are treated as ground during the extraction, and the influence of these objects is considered when you are extracting the capacitance of neighboring signal nets. See the *Milkyway Environment Data Preparation User Guide* for information about setting object properties in the Milkyway database.

### See Also

- [MILKYWAY\\_DATABASE](#)

---

## REPORT\_METAL\_FILL\_STATISTICS

Specifies whether to report metal fill statistics.

### Syntax

REPORT\_METAL\_FILL\_STATISTICS: YES | NO

### Arguments

| Argument     | Description                           |
|--------------|---------------------------------------|
| YES          | Reports metal fill statistics         |
| NO (default) | Does not report metal fill statistics |

### Description

StarRC can read metal fill information from GDSII files. You can determine how many polygons were read from different layers by enabling the reporting of metal fill statistics. If enabled, the statistics are included in the /star/summary/mf\_statistics.sum file.

### See Also

- [METAL\\_FILL\\_GDS\\_FILE](#)

---

## REPORT\_SMIN\_VIOLATION

Specifies whether to report SMIN violations.

### Syntax

REPORT\_SMIN\_VIOLATION: YES | NO

### Arguments

| Argument     | Description                     |
|--------------|---------------------------------|
| YES          | Reports SMIN violations         |
| NO (default) | Does not report SMIN violations |

### Description

It is important to have a correct set of `WMIN` and `SMIN` values for any `CONDUCTOR` statement that contains an `ETCH_VS_WIDTH_AND_SPACING` command. The `WMIN` and `SMIN` values of the conductor should be at least as large as the smallest value (the first entry) in the `WIDTHS` and `SPACINGS` tables, respectively.

Inappropriate `WMIN` and `SMIN` values might cause unwanted opens or shorts of the neighboring layers by applying the etch values provided in the table. In this case, a warning message is issued. For the entries corresponding to the `WMIN` in the `WIDTHS` table, if positive etch values are greater than or equal to half of the `WMIN` value, a warning message is issued about possible opens.

For the entries corresponding to the `SMIN` value in the `SPACINGS` table, if the absolute value of the negative etch is greater than or equal to half of the `SMIN` value, a potential short condition exists. However, reporting of this condition is optional because most such errors should be caught during design rule checking. To enable `SMIN` violation reporting, use the `REPORT_SMIN_VIOLATION: YES` command.

An `SMIN` violation between two neighboring polygons occurs only if all of the following conditions are met:

- The drawn distance between the polygons is less than the `SMIN` value.
- The polygons belong to different nets.
- The polygons are not floating fill polygons or via clones.
- No more than one of the polygons is a grounded fill polygon.
- The distance over which the `SMIN` spacing is in violation is at least 10X the `WMIN` value.

## See Also

- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [SMIN](#)
- [WMIN](#)

---

## RES\_UPDATE\_FILE

Provides a method to modify resistance models without regenerating an nxtgrd file.

### Syntax

`RES_UPDATE_FILE: filename`

### Arguments

| Argument              | Description                                                                   |
|-----------------------|-------------------------------------------------------------------------------|
| <code>filename</code> | File that contains the resistance update data for metal layers and via layers |

### Description

Use the `RES_UPDATE_FILE` command to specify a file that contains `RPSQ_VS_SI_WIDTH` and `RPSQ_VS_SI_WIDTH_AND_LENGTH` tables for `CONDUCTOR` layers and `RPV_VS_AREA` values for `VIA` layers. These specifications take precedence over the mapping file and nxtgrd file settings for the corresponding ITF layers.

Each layer can be defined only one time in the update file. Multiple layer resistance definitions are treated as errors.

The layers defined in the file specified by the `RES_UPDATE_FILE` command are not .db layers. They are ITF layers that can apply to several .db layers in the layout according to the mapping file.

### Example

The following example shows a file named res.itf that specifies resistance update data for metal layers M1, M2, M3, and via layers via1 and via2.

The res.itf file contains the `RPSQ_VS_SI_WIDTH` and `RPV_VS_AREA` tables for the layers that need to be modified as follows:

```
RES_UPDATE_FILE: res.itf

CONDUCTOR M1 {
 RPSQ_VS_SI_WIDTH {
 (0.34, 0.075) (0.40, 0.062)
 (0.823, 0.0817)(2.0, 0.0321)
 (6.0, 0.0173)
 }
}
```

```
CONDUCTOR M2 {
 RPSQ_VS_SI_WIDTH {
 (0.4, 0.075) (0.43, 0.062)
 (0.8, 0.0817)(2.0, 0.0321)
 (8.0, 0.0173)
 }
}
CONDUCTOR M3 {
 RPSQ_VS_SI_WIDTH {
 (0.3, 0.075) (0.36, 0.062)
 (0.70, 0.0817)(2.2, 0.0321)
 (6.9, 0.0173)
 }
}
CONDUCTOR GATE {
 RPSQ_VS_SI_WIDTH_AND_LENGTH {
 LENGTHS {0.05 0.1 0.15}
 WIDTHS {0.02 0.022 0.024}
 VALUES {0.1 0.23 0.45
 0.12 0.26 0.47
 0.18 0.28 0.53 }
 }
}
VIA via1 {
 RPV_VS_AREA {
 (350, .5)
 (600, .25)
 (200, .5)
 }
}
VIA via2 {
 RPV_VS_AREA {
 (350, .5)
 (600, .25)
 (200, .5)
 }
}
```

## See Also

- [RPSQ\\_VS\\_SI\\_WIDTH](#)
- [RPSQ\\_VS\\_SI\\_WIDTH\\_AND\\_LENGTH](#)
- [RPV\\_VS\\_AREA](#)

---

## RETAIN\_CAPACITANCE\_CAP\_MODELS

Specifies model-specific plate-to-plate capacitance retention for capacitor devices.

### Syntax

RETAIN\_CAPACITANCE\_CAP\_MODELS : *model\_list*

### Arguments

| Argument          | Description                                                                                                                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>model_list</i> | List of capacitor devices for which plate-to-plate capacitance is to be retained.<br>Accepted model names in the list include either the schematic or layout names,<br>regardless of the XREF command setting.<br>Default: none |

### Description

In certain applications, it is advantageous to retain parasitic capacitances within the parasitic netlist for capacitor devices, particularly when you want to simulate devices using parasitic capacitances instead of device simulation models. To do this, you can selectively retain plate-to-plate capacitance for designed capacitor devices. Devices whose capacitances are to be retained are specified by model name.

Specify a list of model names of capacitor devices for which IGNORE\_CAPACITANCE statements should not be generated between terminal layers and for which plate-to-plate capacitance should not be ignored by the StarRC extraction engine.

If a specified model name matches the schematic model name of a capacitor device in the design, the RETAIN\_CAPACITANCE\_CAP\_MODELS functionality is propagated to all layout capacitor devices matching that schematic model name. If a specified model name matches the layout model name of a capacitor device in the design, the RETAIN\_CAPACITANCE\_CAP\_MODELS functionality is propagated only to layout capacitor devices matching that layout model name. All of this occurs independent of the XREF command in the command file.

#### Note:

If the MODEL\_TYPE command is not specified in the command file, the default setting is LAYOUT.

### Errors

A warning is issued when a model name exists in the RETAIN\_CAPACITANCE\_CAP\_MODELS command for which no corresponding capacitor exists.

## Example

The following command retains capacitance for device cm1m2:

```
RETAIN_CAPACITANCE_CAP_MODELS: cm1m2
```

## See Also

- [MODEL\\_TYPE](#)

---

## RETAIN\_GATE\_CONTACT\_COUPLING

Retains gate-to-contact coupling capacitance.

### Syntax

RETAIN\_GATE\_CONTACT\_COUPLING: YES | NO

### Arguments

| Argument     | Description                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------|
| YES          | Retains the gate-to-contact capacitance when COUPLE_TO_GROUND: YES is set in the StarRC command file |
| NO (default) | All couplings are retained if COUPLE_TO_GROUND: NO is set in the StarRC command file.                |

### Description

When this command is set in conjunction with EXTRACT\_VIA\_CAPS: YES, the gate-to-contact capacitance is retained with COUPLE\_TO\_GROUND: YES, even if the coupling capacitances are below the capacitance threshold value specified for COUPLING\_ABS\_THRESHOLD or COUPLING\_REL\_THRESHOLD.

When COUPLE\_TO\_GROUND: YES is set in the StarRC command file, coupling is retained to the contact. Coupling thresholds are not used for the gate-to-contact coupling. When COUPLE\_TO\_GROUND: NO is set in the StarRC command file, the coupling to contact is retained only if the coupling is above the coupling thresholds specified.

**Note:**

The RETAIN\_GATE\_CONTACT\_COUPLING command is supported only in the Hercules, IC Validator, and Calibre flows.

**Table 15-8** describes the behavior of the `RETAIN_GATE_CONTACT_COUPLING` command when specified with other commands.

*Table 15-8 RETAIN\_GATE\_CONTACT\_COUPLING Interaction With Commands*

| Command                                                                                                | COUPLE_TO_GROUND:YES                                                                                                                                     | COUPLE_TO_GROUND:NO                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NETS : *                                                                                               | Retain the gate-to-contact capacitance for all signal nets even if they are less than the coupling threshold values.                                     | All coupling capacitances for signal nets including gate-to-contact are retained and netlisted. Coupling thresholds apply to all coupling capacitances, including gate-to-contact. |
| EXTRACT_VIA_CAPS: YES<br>RETAIN_GATE_CONTACT_COUPLING: YES<br>POWER_EXTRACT: NO                        |                                                                                                                                                          |                                                                                                                                                                                    |
| NETS : *<br>EXTRACT_VIA_CAPS: YES<br>RETAIN_GATE_CONTACT_COUPLING: YES<br>POWER_EXTRACT: DEVICE_LAYERS | Retain the gate-to-contact capacitance for all signal and power nets, only if the appropriate device layers are set (that is, contacted) for power nets. | All coupling capacitances for signal and power nets are retained, only if the appropriate device layers are set (that is, contacted) for power nets.                               |
| NETS : *<br>EXTRACT_VIA_CAPS: YES<br>RETAIN_GATE_CONTACT_COUPLING: YES<br>POWER_EXTRACT: YES           | Retain the gate-to-contact capacitance for all signal nets and power nets.                                                                               | All coupling capacitances for signal and power nets are retained and netlisted.                                                                                                    |

## Errors

The `RETAIN_GATE_CONTACT_COUPLING` command results in the following error message when it is used in a LEF/DEF or Milkyway flow:

Error Message: `RETAIN_GATE_CONTACT_CAPACITANCE` is only supported in the Hercules or Calibre flow.

## See Also

- [NETS](#)
- [NETS\\_FILE](#)

---

## RING\_AROUND\_THE\_BLOCK

Generates a virtual ring on every routing layer around the block for extraction.

### Syntax

RING\_AROUND\_THE\_BLOCK: YES | NO

### Arguments

| Argument     | Description                               |
|--------------|-------------------------------------------|
| YES          | Generates a ring around the block         |
| NO (default) | Does not generate a ring around the block |

### Description

The RING\_AROUND\_THE\_BLOCK command generates a virtual ring around the block for extraction. The capacitance between the block material and the imaginary ring is calculated as though the block is surrounded by solid metal.

Specify the block for extraction with the BLOCK command.

Specify the block boundary with the BLOCK\_BOUNDARY command and a list of points, which can be read directly from the Milkyway database. It must be provided for LEF/DEF designs. Block material that lies outside the specified boundary does not create shorts with or attach capacitance from the imaginary rings, even if they overlap. Overlaps should be avoided, because shielding of nets occurs.

You can specify the spacing between the block boundary and the ring with the RING\_AROUND\_THE\_BLOCK\_SMIN\_MULTIPLIER command.

The rings are grounded by default but can be given a special name with the ZONE\_COUPLE\_TO\_NET command.

### See Also

- [BLOCK](#)
- [BLOCK\\_BOUNDARY](#)
- [RING\\_AROUND\\_THE\\_BLOCK\\_SMIN\\_MULTIPLIER](#)
- [ZONE\\_COUPLE\\_TO\\_NET](#)

---

## RING\_AROUND\_THE\_BLOCK\_SMIN\_MULTIPLIER

Specifies the spacing between the block boundary and the ring around the block in hierarchical analysis.

### Syntax

`RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER: multiplier`

### Arguments

| Argument          | Description                                                                                   |
|-------------------|-----------------------------------------------------------------------------------------------|
| <i>multiplier</i> | Multiplier; floating-point number greater than or equal to 0.5<br>Units: none<br>Default: 0.5 |

### Description

The `RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER` command specifies the spacing between the block boundary and the ring around the block. The spacing is equal to the product of the multiplier and the SMIN value of the nxtgrd layer. The SMIN value might be different from layer to layer, resulting in different ring spacing.

### See Also

- [BLOCK](#)
- [BLOCK\\_BOUNDARY](#)
- [RING\\_AROUND\\_THE\\_BLOCK](#)

---

## SELECTED\_CORNERS

Specifies the corners to be extracted and netlisted in the simultaneous multicorner flow.

### Syntax

SELECTED\_CORNERS: *name\_list*

### Arguments

| Argument         | Description                                                    |
|------------------|----------------------------------------------------------------|
| <i>name_list</i> | The corners to be extracted and netlisted, separated by spaces |

### Description

This command specifies the corners to be extracted in the simultaneous multicorner (SMC) flow. The command has an effect only if the SMC flow is enabled.

Every corner listed in the SELECTED\_CORNERS command must be defined in the corners file specified by the CORNERS\_FILE command.

To create a SPEF netlist containing more than one corner, use a colon (:) between corner names. The first corner listed is considered the primary corner for output netlist reduction. This feature is valid only for transistor-level flows.

The name of the netlist file for a given corner is the base file name specified in the NETLIST\_FILE command, with the corner name appended to it.

### Example

Consider the following definition of selected corners:

SELECTED\_CORNERS: C1 C3 C4

In this case, three netlists are created. The first netlist contains parasitic data from C1, the second netlist contains data from C3, and the third netlist contains data from C4.

### See Also

- [CORNERS\\_FILE](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)
- [Simultaneous Multicorner Extraction](#)

---

## SHEET\_COUPLE\_TO\_NET

Specifies a prefix net name for the sheet zone extraction capability. This user-defined name appears in the generated output netlist.

### Syntax

SHEET\_COUPLE\_TO\_NET: *prefix\_name*

### Arguments

| Argument           | Description                                                     |
|--------------------|-----------------------------------------------------------------|
| <i>prefix_name</i> | Net prefix name reported in the output netlist<br>Default: none |

### Description

If this command is not specified, StarRC automatically sets the prefix name to ZONE\_SHEET.

### Example

```
METAL_SHEET_OVER_AREA: METAL2 0 0 100 100
METAL_SHEET_OVER_AREA: METAL2 200 200 400 400
METAL_SHEET_OVER_AREA: METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

### See Also

- [METAL\\_SHEET\\_OVER\\_AREA](#)
- [SHEET\\_COUPLE\\_TO\\_NET\\_LEVEL](#)

---

## SHEET\_COUPLE\_TO\_NET\_LEVEL

Enables or disables a net name suffix using the layer-level number for the sheet zone netlist capability.

### Syntax

SHEET\_COUPLE\_TO\_NET\_LEVEL: NO | YES

### Arguments

| Argument     | Description                                                     |
|--------------|-----------------------------------------------------------------|
| NO (default) | Disables net name suffix reporting in sheet zone netlist output |
| YES          | Enables net name suffix reporting in sheet zone netlist output  |

### Description

Enables or disables a net name suffix using the layer-level number for the sheet zone netlist capability.

### Example

```
METAL_SHEET_OVER_AREA: METAL2 0 0 100 100
METAL_SHEET_OVER_AREA: METAL2 200 200 400 400
METAL_SHEET_OVER_AREA: METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

### See Also

- [METAL\\_SHEET\\_OVER\\_AREA](#)
- [SHEET\\_COUPLE\\_TO\\_NET](#)

---

## SHORT\_EQUIV\_NODES

Specifies whether StarRC adds a short resistor between short equivalent nodes.

### Syntax

SHORT\_EQUIV\_NODES: YES / NO

### Arguments

| Argument     | Description                     |
|--------------|---------------------------------|
| YES          | Shorts equivalent nodes         |
| NO (default) | Does not short equivalent nodes |

### Description

The SHORT\_EQUIV\_NODES command merges equivalent nets into one single net. A short resistor is added during netlisting to merge the nets.

You must specify the XREF: YES command to use the SHORT\_EQUIV\_NODES command.

#### Note:

The resistance value is set to 0.01 ohm for shorting opens and 0.001 ohm for shared nodes.

You can use this command when using IC Validator or Hercules with StarRC. You cannot use this command with the Calibre Connectivity Interface because this database does not contain information about equivalent nodes.

### Example

When you use the following command in the StarRC command file, StarRC adds a short resistor between short equivalent nodes:

```
SHORT_EQUIV_NODES : YES
```

### See Also

- [XREF](#)

---

## SHORT\_PINS

Specifies whether to short top-level pin ports that have multiple placements.

### Syntax

SHORT\_PINS: YES | NO | MIXED

### Arguments

| Argument      | Description                                                                                                                                                                                                                                                                                                                  |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES (default) | Reports all of the placements of the pin port as a single node (a single * P)                                                                                                                                                                                                                                                |
| NO            | Each Hercules or IC Validator marker group, or Milkyway (LEF/DEF flow) PIN, at the top level is uniquely netlisted with a suffix defined by NETLIST_RENAME_PORTS.                                                                                                                                                            |
| MIXED         | Ensures correct back-annotation of a parasitic netlist. This option is only available in the LEF/DEF flow. In a Milkyway flow, the SHORT_PINS:MIXED command is the same as the SHORT_PINS:YES command. In a Hercules, IC Validator, or Calibre flow, the pin names are the same as net names, so this option does not apply. |

### Description

A unique pin instance is defined as a physically connected group of objects marked as interface material: PIN section in DEF, PIN type in Milkyway, MARKER\_LAYER in Hercules. Each separated group of connected objects is a pin instance. When you specify the SHORT\_PINS:YES command, the name of the \*P connected to a net always inherits the name of the \*D\_NET in case there are multiple names.

If the layout database has unique names for each pin instance associated with a single port, that information is used to netlist the \*P, instead of the NETLIST\_RENAME\_PORTS method being used. The NETLIST\_RENAME\_PORTS delimiter is used only if no naming information exists in the input layout database.

## Examples

Input database contains unique pin names:

```
PINS 2 ;
- C.1 + NET C + LAYER METAL2 (0 0) (1000 1600)
 + PLACED (263800 136000) N ;
- C.2 + NET C + LAYER METAL2 (0 0) (1000 1600)
 + PLACED (264800 136000) N ;
END PINS

SHORT_PINS: NO
* |NET C 0.0295887PF
* |P (C.2 B 0 264.8 136)
* |P (C.1 B 0 263.8 136)

SHORT_PINS: YES
* |NET C 0.0295887PF
* |P (C B 0 264.8 136)
```

Input database does not contain unique pin names:

```
PINS 2 ;
- C + NET C + LAYER METAL2 (0 0) (1000 1600)
 + PLACED (263800 136000) N ;
- C + NET C + LAYER METAL2 (0 0) (1000 1600)
 + PLACED (264800 136000) N ;
END PINS

SHORT_PINS: NO, NETLIST_RENAME_PORTS: _
* |NET C 0.0295887PF
* |P (C_2 B 0 264.8 136)
* |P (C_1 B 0 263.8 136)

SHORT_PINS: YES
* |NET C 0.0295887PF
* |P (C B 0 264.8 136)
```

Input database contains partial unique naming information:

```
PINS 3 ;
- C.1 + NET C + LAYER METAL2 (0 0) (1000 700)
 + PLACED (263800 136000) N ;
- C.1 + NET C + LAYER METAL2 (0 0) (1000 700)
 + PLACED (263800 137000) N ;
- C.2 + NET C + LAYER METAL2 (0 0) (1000 1600)
 + PLACED (264800 136000) N ;
END PINS

SHORT_PINS: NO, NETLIST_RENAME_PORTS: -
*|NET C 0.0295974PF
*|P (C.1 B 0 263.8 136)
*|P (C.2 B 0 264.8 136)
*|P (C.1_1 B 0 263.8 137)

SHORT_PINS: YES
*|NET C 0.0295887PF
*|P (C B 0 264.8 136)
```

Input database contains the mixed pin names shown in the following table:

| Net name | Logical pins | Physical pins                        | SHORT_PINS: MIXED |
|----------|--------------|--------------------------------------|-------------------|
| Net 1    | Net 1        | Net 1                                | Net 1             |
| Net 1    | net 1        | Net 1 (Multiple)                     | Net 1             |
| Net 1    | Pin 1        | Pin 1                                | Pin 1             |
| Net 1    | Pin 1 Net 1  | Pin 1 Net 1                          | Pin1 Net 1        |
| Net 1    | Pin 1 Net 1  | Pin 1 Net 1 (Multiple)               | Pin1 Net 1        |
| Net 1    | Pin 1 Net 1  | Pin 1 (Multiple) Net 1<br>(Multiple) | Pin1 Net 1        |

### Fixing Redundant Ports

Redundant ports that are unnecessary for an HSIM or NanoSim parasitic back-annotation flow can be removed. The port name postfix “\_1” in an extracted file is generated because the input database contained partial unique naming information, such as:

```
SUBCKT v_ctl VCI_P0 VCI_P0_1 GND_P0 GND_P0_1 VDD_P0 VDD_P0_1
```

The redundant ports “VCI\_P0\_1”, “GND\_P0\_1” and “VDD\_P0\_1” can be removed by changing the StarRC command SHORT\_PINS:NO to SHORT\_PINS:YES.

The result is as follows:

```
SUBCKT v_ctl VCI_P0 GND_P0 VDD_P0
```

The extracted standard parasitic format file can then be used for HSIM or NanoSim parasitic back-annotation.

## See Also

- [NETLIST\\_RENAME\\_PORTS](#)

---

## SHORT\_PINS\_IN\_CELLS

Merges all electrically equivalent pins into a single port for the specified list of skip cells.

### Syntax

SHORT\_PINS\_IN\_CELLS: *list*

### Arguments

| Argument    | Description                               |
|-------------|-------------------------------------------|
| <i>list</i> | The list of skip cell names<br>Default: * |

### Description

Place and route Milkyway databases sometimes contain ports of that are electrically equivalent. These ports are logically equivalent but can have a wide geographic distribution throughout the design. StarRC merges all electrically equivalent pins into a single port for every SKIP\_CELLS on the SHORT\_PINS\_IN\_CELLS list.

Occasionally, designs with electrically equivalent ports are instantiated as macros. If a macro with electrically equivalent ports is on the SKIP\_CELLS list, by default, StarRC reports a single connection for the electrically equivalent port in the netlist, using the first electrically equivalent pin location. Because the electrically equivalent pins can be so widely distributed in the layout, it can be desirable to treat each of the electrically equivalent pins as a unique port for simulation. Negating a cell from the SHORT\_PINS\_IN\_CELLS list means that StarRC treats each electrically equivalent pin as a unique port and uses the original database port suffix to report it.

Wildcards “\*”, “?”, and “!” are acceptable. This command can be specified multiple times. SHORT\_PINS always overrides SHORT\_PINS\_IN\_CELLS.

### See Also

- [CASE\\_SENSITIVE](#)
- [SHORT\\_PINS](#)
- [SKIP\\_CELLS](#)

---

## **SIMULTANEOUS\_MULTI\_CORNER**

Enables the simultaneous multicorner (SMC) flow.

### **Syntax**

`SIMULTANEOUS_MULTI_CORNER: YES | NO`

### **Arguments**

| <b>Argument</b> | <b>Description</b>                        |
|-----------------|-------------------------------------------|
| YES (default)   | Enables the simultaneous multicorner flow |
| NO              | Uses the single-corner extraction flow    |

### **Description**

Simultaneous multicorner (SMC) extraction optimizes the efficient extraction of multiple process and temperature corners for a single design.

Simultaneous multicorner extraction requires that the `CORNERS_FILE` and `SELECTED_CORNERS` commands also be set. When the simultaneous multicorner flow is enabled,

- The StarRC tool uses the `nxtgrd` files specified in the corners file and ignores any other `TCAD_GRD_FILE` commands in the command file.
- The tool uses the operating temperatures specified in the corners file and ignores any other `OPERATING_TEMPERATURE` commands in the command file.

Simultaneous multicorner extraction occurs whenever the `SIMULTANEOUS_MULTI_CORNER` command is either absent or set to `YES` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are both present. Other combinations are handled as follows:

- If the `SIMULTANEOUS_MULTI_CORNER` command is either absent or set to `NO` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are not present, StarRC runs single-corner extraction. For gate-level flows, the tool also issues a warning message to point out that simultaneous multicorner extraction would provide better runtime.
- If the `SIMULTANEOUS_MULTI_CORNER` command is set to `YES` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are not present, the tool issues an error message.
- If the `SIMULTANEOUS_MULTI_CORNER` command is set to `NO` and the `CORNERS_FILE` and `SELECTED_CORNERS` commands are present, the tool issues an error message.

## Using the Field Solver Flows With Simultaneous Multicorner Extraction

If you use the SMC flow with the EXTRATION: FSCOMPARE command, StarRC generates the total and coupling capacitance report files for each corner with a unique nxtgrd file specified by the SELECTED\_CORNERS command. If multiple corners with the same nxtgrd file are selected, StarRC generates a report only for the first corner because capacitance does not depend on the operating temperature.

You can use the FS\_EXTRACT\_NETS command in the simultaneous multicorner flow. If you specify nets with the SELECTED\_CORNERS and FS\_EXTRACT\_NETS commands, the field solver generates netlist files for the different corners.

### Example

The following example uses one netlist per corner in a simultaneous multicorner flow. The three corners are the nxtgrd file named nominal.nxtgrd analyzed at two temperatures (-25 and 125°C) and the nxtgrd file named rcmax.nxtgrd analyzed at one temperature (25°C).

The command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: corners.smc
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
```

The corners file contains the following commands:

```
CORNER_NAME: NOM_T1
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: -25

CORNER_NAME: NOM_T2
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: 125

CORNER_NAME: RCMAX_T3
TCAD_GRD_FILE: rcmax.nxtgrd
OPERATING_TEMPERATURE: 25
```

The resulting output files are star\_NOM\_T1.spf, star\_NOM\_T2.spf, and star\_RCMAX\_T3.spf.

### See Also

- [CORNERS\\_FILE](#)
- [SELECTED\\_CORNERS](#)
- [Simultaneous Multicorner Extraction](#)

---

## **SKIP\_CELL\_AGF\_FILE**

Imports Hercules annotated GDSII (AGF) files as the detail view for skip cells.

### **Syntax**

`SKIP_CELL_AGF_FILE: cell agf_file herc_ideal_netlist`

### **Arguments**

| <b>Argument</b>                        | <b>Description</b>          |
|----------------------------------------|-----------------------------|
| <code><i>cell</i></code>               | AGF cell name               |
| <code><i>agf_file</i></code>           | AGF file name               |
| <code><i>herc_ideal_netlist</i></code> | Hercules ideal netlist name |

### **Description**

Use this command in the StarRC command file to import Hercules annotated GDSII (AGF) files as the detail view for skip cells. Specify the command one time for each skip cell that has an AGF description.

- A `SKIP_CELL_AGF_FILE` command cannot be specified as a macro.
- All `SKIP_CELL_AGF_FILE` commands specified must use the same layer mapping as defined in the `GDS_LAYER_MAP_FILE` command.
- A layer mapping file must also be shared with the `GDS_FILE` (if it is used).

If the list of cells in a `COUPLE_NONCRITICAL_NETS` command contains a `SKIP_CELL_AGF_FILE` cell (or descendant), the layout names are used to identify its contents. If the list of cells in a `COUPLE_NONCRITICAL_NETS` command does not contain a `SKIP_CELL_AGF_FILE` cell, its contents are annotated as noncritical (ground potential).

Child cells of a `COUPLE_NONCRITICAL_NETS` command must be specified in the `COUPLE_NONCRITICAL_NETS` command as well; they are not automatically included.

### **Example**

`SKIP_CELL_AGF_FILE: INV_BUF buf.agf buf.sp`

### **See Also**

- [COUPLE\\_NONCRITICAL\\_NETS](#)
- [GDS\\_LAYER\\_MAP\\_FILE](#)

---

## SKIP\_CELL\_PORT\_PROP\_FILE

Specifies files containing pin or port information for skip cell properties.

### Syntax

SKIP\_CELL\_PORT\_PROP\_FILE: *file1* [*file2* ... *fileN*]

### Arguments

| Argument     | Description                |
|--------------|----------------------------|
| <i>fileN</i> | File name<br>Default: none |

### Description

Use the SKIP\_CELL\_PORT\_PROP\_FILE command to specify files that contain pin information (such as direction or capacitance) for LEF/DEF and checkpoint databases (or to override values from a Milkyway database) for netlist purposes.

The description of all port properties of a cell should be in one CELL block and should be uniquely defined. For example, do not specify multiple descriptions of a cell in the same or another (reference) library. Otherwise, the result is unpredictable. Ensure that reference libraries you specify contain unique definitions of cells.

Create a port property file by specifying the keyword CELL followed by the cell name. Follow this by specifying the named cell's pin name, pin direction, and pin capacitance. All three parameters are required.

| Argument         | Description                                                                                                                                   |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| CELL             | Keyword specifying the beginning of the cell definition                                                                                       |
| <i>cell_name</i> | Name of the cell whose pin characteristics are defined                                                                                        |
| <i>pin_name</i>  | String of the pin or port name                                                                                                                |
| <i>pin_io</i>    | Pin or port direction. Valid values are I, O, or B representing input, output, and bidirectional                                              |
| <i>pin_cap</i>   | The capacitance value of the corresponding pin or port. It can include a unit of the capacitance. The valid units are: FF, PF, NF, uF and mF. |

## Example

The following example shows the port property file format:

```
CELL cell1name
pin1name pin1io pin_cap
pin2name pin2io pin_cap

CELL cell2name
pin1name pin1io pin_cap
pin2name pin2io pin_cap
```

---

## SKIP\_CELLS

Creates a white-space-delimited list of cells for StarRC to skip during extraction.

### Syntax

SKIP\_CELLS: *cell1 cell2 ... cellN*

### Arguments

| Argument                     | Description                                                                                        |
|------------------------------|----------------------------------------------------------------------------------------------------|
| <i>cell1 cell2 ... cellN</i> | A list of cells to be skipped during extraction. Wildcard characters are acceptable.<br>Default: * |

### Description

The SKIP\_CELLS command creates a white-space-delimited list of cells for StarRC to skip during extraction. This command can be specified multiple times in a single command file. The asterisk (\*), exclamation mark (!), and question mark (?) wildcard characters are acceptable. Case sensitivity for selection purposes is determined by the value of the CASE\_SENSITIVE command, but the netlist always retains the case of the original input database.

Skip cells are typically cells with their own timing models, which can later be applied, along with the StarRC parasitic netlist, to perform a timing analysis. Skip cells should contain labeled or otherwise annotated pin shapes for proper parasitic netlisting.

The default for all extraction flows is to skip all lower-level blocks in the input database, so any macro blocks without timing models must be negated from the list.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a SKIP\_CELLS or SKIP\_PCELLS command.

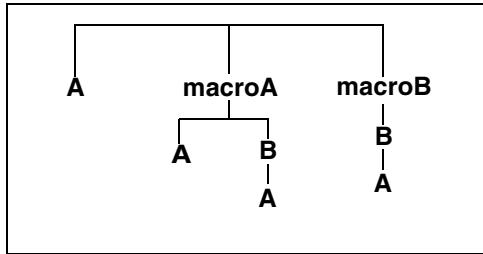
#### Note:

When you use the CALIBRE\_PDBA\_FILE command, the SKIP\_CELLS command might fail because the DFM properties annotation can promote the instance hierarchy.

### Example

```
SKIP_CELLS: cellA !cellB cell? *C ...
SKIP_CELLS: * !macroA !macroB ...
```

Figure 15-20 Example Using SKIP\_CELLS



Using the hierarchy shown in Figure 15-20, only cell A is skipped in the following example:

```
SKIP_CELLS: A
```

Cell macroA is not skipped. The resulting netlist contains 4 instances of A.

If you use the following command, cells A and B is skipped:

```
SKIP_CELLS: A B
```

Cells macroA and macroB are not skipped. The resulting netlist contains 2 instances of A and 2 instances of B.

In following example, all cells are skipped:

```
SKIP_CELLS: *
```

The following example specifies that all cells except macroA and macroB are skipped:

```
SKIP_CELLS: * !macro?
```

In following example, no cells are skipped:

```
SKIP_CELLS: !*
```

## See Also

- [CASE\\_SENSITIVE](#)
- [CELL\\_TYPE](#)

---

## SKIP\_CELLS\_COUPLE\_TO\_NET

Specifies a lump net for all coupling capacitance between top-level routes and noncritical skip cell material.

### Syntax

SKIP\_CELLS\_COUPLE\_TO\_NET: *net\_name*

### Arguments

| Argument        | Description                                                               |
|-----------------|---------------------------------------------------------------------------|
| <i>net_name</i> | The net name to the noncritical skip cell material<br>Default: ground net |

### Description

The default is to use ground as the noncritical skip cell material potential.

You must set the NETLIST\_FORMAT: SPEF and COUPLE\_TO\_GROUND: NO commands to use this option.

Use this option in conjunction with the SKIP\_CELLS\_COUPLE\_TO\_NET\_LEVEL command to append the net name specified in the SKIP\_CELLS\_COUPLE\_TO\_NET command to the actual skip cell object layer number.

### Example

SKIP\_CELLS\_COUPLE\_TO\_NET: LUMP

### See Also

- [COUPLE\\_TO\\_GROUND](#)
- [NETLIST\\_FORMAT](#)
- [SKIP\\_CELLS\\_COUPLE\\_TO\\_NET\\_LEVEL](#)

---

## SKIP\_CELLS\_COUPLE\_TO\_NET\_LEVEL

Appends the SKIP\_CELLS\_COUPLE\_TO\_NET name to the real layer number for the SKIP\_CELLS material.

### Syntax

SKIP\_CELLS\_COUPLE\_TO\_NET\_LEVEL: YES | NO

### Arguments

| Argument     | Description                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------|
| YES          | Specifies that the layer number of the SKIP_CELLS material is appended to the assigned net name     |
| NO (default) | Specifies that the layer number of the SKIP_CELLS material is not appended to the assigned net name |

### Description

This command appends the SKIP\_CELLS\_COUPLE\_TO\_NET name to the real layer number for the SKIP\_CELLS material.

This command is ignored unless the SKIP\_CELLS\_COUPLE\_TO\_NET command is specified.

### Example

If the net name is LUMP and this command is set to YES, the resulting coupling capacitor between top-level net1 and a metal1 object inside a SKIP\_CELLS might be as follows:

```
*CAP
...
12 net1 LUMP_1 1.2e-15
...
*END
```

### See Also

- [SKIP\\_CELLS\\_COUPLE\\_TO\\_NET](#)

---

## SKIP\_CELLS\_FILE

Specifies a file containing SKIP\_CELLS commands.

### Syntax

SKIP\_CELLS\_FILE: *file1 file2 ...*

### Arguments

| Argument                | Description                                              |
|-------------------------|----------------------------------------------------------|
| <i>file1, file2 ...</i> | Files containing the defined skip cells<br>Default: none |

### Description

This command specifies a file containing SKIP\_CELLS commands. See the SKIP\_CELLS command for more details.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a SKIP\_CELLS or SKIP\_PCELLS command.

### Example

The following lines appear in a file named cells\_to\_skip.dat.

```
SKIP_CELLS: cellA cellB cellX
SKIP_CELLS: cellC
```

The following command appears in the StarRC command file:

```
SKIP_CELLS_FILE: cells_to_skip.dat
```

As a result, cellA, cellB, cellC, and cellX are all skipped during extraction, and the cell names are written into the summary/translate.sum file.

### See Also

- [CASE\\_SENSITIVE](#)
- [SKIP\\_CELLS](#)

---

## SKIP\_INSTANCES

Specifies layout instance instances to skip.

### Syntax

`SKIP_INSTANCES instance_list`

### Arguments

| Argument             | Description                                                |
|----------------------|------------------------------------------------------------|
| <i>instance_list</i> | The list of layout instance names to skip<br>Default: none |

### Description

The `SKIP_INSTANCES` command lists layout instance names that should be treated as skip cells. Wildcards “\*”, “!”, and “?” are accepted, but cannot be used as global arguments (for example, the `SKIP_INSTANCES: *` command is not allowed).

Instances that you might want to skip are those that already have timing model or parasitic netlists elsewhere in the library representing the cell master.

The default is not to skip any specific instances, so all cell instances are skipped or flattened depending on the `SKIP_CELLS` command setting.

### Example

In this example, all macroA instances are flattened, except macroA\_instance1. All macroB instances are flattened, except macroB\_instance1.

```
SKIP_CELLS: * !macroA !macroB
SKIP_INSTANCES: macroA_instance1 macroB_instance1
```

### See Also

- [SKIP\\_CELLS](#)

---

## SKIP\_PCELLS

Extracts a parameterized cell (PCELL) as a fully characterized gray-box cell unit during parasitic extraction and creates the entity in the DSPF netlist with all layout properties extracted for the PCELL.

### Syntax

SKIP\_PCELLS: *pcell\_name*

### Arguments

| Argument          | Description                                                                                                                  |
|-------------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>pcell_name</i> | Name of parameterized cell. Only the exclamation mark (!) and asterisk (*) wildcard characters can be used.<br>Default: none |

### Description

The SKIP\_PCELLS command extracts a parameterized cell (PCELL) as a fully characterized gray-box cell unit during parasitic extraction and creates the entity in the DSPF netlist with all layout properties extracted for the PCELL. StarRC defines a PCELL as a container cell, within which one or more devices (including hierarchical cells) are extracted by the ideal device extraction tool. With this command, StarRC treats the container cell as a gray-box for parasitic extraction purposes but creates an entry in the DSPF Instance section listing all geometric properties of the ideal device extracted inside the container cell.

In this flow, the PCELL placed in layout is assumed to be a fully characterized unit, for which the layout's PCELL container cell boundary defines the perimeter between the intradevice effects inside the cell and the interconnect effects outside the cell. Runset terminal layer manipulation is not required to isolate intradevice effects because the PCELL cell boundary serves this role. Using the cell boundary eliminates the need to perform runset terminal layer manipulation for PCELLs while retaining device properties in the netlist. This is the functional benefit of this command.

When you specify the SKIP\_PCELLS command, StarRC

- Creates the entity in the DSPF instance section as a device with all layout-extracted device properties; the instantiation name must be consistent with the ideal devices inside the container cell
- Treats the container cell as a gray box (analogous to the SKIP\_PCELLS command functionality), which means that parasitic effects are extracted up to the cell boundary

The StarRC tool handles exploded shapes in PCELLS by supporting the following scenarios:

- PCELL shapes that are exploded to the upper level
- Flattened designs with some premodeled areas in which the PCELL is not preserved

Both scenarios require you to specify the PCELL or blocking layer, as well as the layers that are exploded, in a file specified by the `SKIP_PCELL_LAYERS_FILE` command.

The `translate.sum` file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a `SKIP_CELLS` or `SKIP_PCELLS` command.

### See Also

- [SKIP\\_CELLS](#)
- [SKIP\\_PCELL\\_LAYERS\\_FILE](#)

---

## SKIP\_PCELL\_LAYERS\_FILE

Specifies a file that contains information about PCELL layers.

### Syntax

SKIP\_PCELL\_LAYERS\_FILE: *file\_name*

### Arguments

| Argument         | Description                |
|------------------|----------------------------|
| <i>file_name</i> | File name<br>Default: none |

### Description

The SKIP\_PCELL\_LAYERS\_FILE command specifies a file that lists parameterized cell (PCELL) layers for the purpose of blocking extraction. The blocked region still contributes to coupling capacitance. You can use the SKIP\_PCELL\_LAYERS\_FILE command for both conducting layers and via layers.

Skipping parameterized cells is a strategy that uses cells in the design to block parasitic extraction from other portions of the design. An alternative strategy uses polygons on a blocking layer to block extraction. The blocking layer approach works with any design hierarchy.

The PCELL names must be preserved in the hierarchy of the layout-versus-schematic tool results database, including the Calibre Connectivity Interface or Milkyway XTR view database, although some shapes are exploded.

The file uses the following syntax:

```
PCELL_LAYERS
pcell_name1 layer1 layer2 ...
pcell_name2 layer3 layer4 ...
pcell_* layer5 layer6 ...

BLOCKING_LAYERS
block_layer1 [SIZE svalue | SCALE factor] [TOUCH layerX] layer1 layer2 ...
block_layer2 [SIZE svalue | SCALE factor] [TOUCH layerY] layer3 layer4 ...
```

To include comments, use a pound sign (#) at the beginning of the comment. The StarRC tool ignores all characters on the same line after the pound sign.

Requirements for the PCELL\_LAYERS section are as follows:

- The first field specifies the PCELL name; the asterisk (\*) wildcard is allowed. The remaining fields specify the list of exploded layers. Only the found polygons in the blocking layer are retained for use in the blocking layer.
- The PCELLs must be specified in the SKIP\_PCELLS command.
- The layers must be specified in the StarRC mapping file.

Requirements for the BLOCKING\_LAYERS section are as follows:

- The blocked layer must contain instance pin, port, or probe texts. You cannot use a blocking layer to block parasitics under a marker layer or to block floating nets.
- The first field specifies the blocking layer name.
- The SIZE parameter (in microns) expands or shrinks the blocking layer by the specified amount. A positive value specifies expansion; a negative value indicates shrinkage.
- The SCALE parameter expands or shrinks the blocking layer by the specified factor.
- The (R) flag, when appended to a blocking layer name without any spaces, blocks only resistance extraction on these polygons. The (C) flag appended to the layer name blocks only capacitance extraction. If the layer name appears alone, both resistance and capacitance are blocked. The parentheses are part of the syntax. For example, poly\_23(R) blocks only resistance extraction for layer poly\_23.
- The TOUCH keyword indicates that a blocking layer polygon is valid only when it touches a polygon in one of the specified TOUCH layers. See the example section for more information.

If you specify the blocking layers section, you do not need to have PCELLs in the design; the design can be completely flattened. When using this approach,

- The blocking layers must be specified in the Calibre Connectivity Interface or Milkyway XTR view.
- Avoid using both blocking layers and PCELLs for the same area.

You can specify an ITF layer in both the PCELL\_LAYERS and BLOCKING\_LAYERS sections. If multiple database layers map to the same ITF layer, specify the ITF layer instead of the database layers at that level. All database layers mapped to the ITF layer are blocking layers. To specify an ITF layer, use the following syntax, with no spaces before or after the double colon:

`ITF::itfLayerName`

For example,

```
BLOCKING_LAYERS
SIZE_ID_MN3V3 PWELLC_FINAL SUBS_FINAL ITF::ACTIVE
```

ITF layer names are case-sensitive except when specifying a substrate layer. You can use either the `ITF::SUBSTRATE` or `ITF::substrate` syntax.

## Errors

If a layer specified in the `BLOCKING_LAYERS` section does not block any texts, the StarRC tool issues a warning message.

If the `COPPLE_TO_PCELL_PINS` command is set to `YES`, the StarRC tool cannot calculate the coupling capacitance to the specified PCELLs, possibly resulting in capacitance underestimation. Additional information is provided in the `pcelllayers.sum` file in the summary directory.

## Examples

### Use of the `PCELL_LAYERS` Section

[Table 15-9](#) illustrates the use of the `PCELL_LAYERS` section.

*Table 15-9 Example of PCELL\_LAYERS Usage*

|                                                      |                                                                                                                                                                                 |
|------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StarRC command to identify PCELLS                    | <code>SKIP_PCELLS: Cell_1 Cell_2</code>                                                                                                                                         |
| StarRC command to specify the skip PCELL layers file | <code>SKIP_PCELL_LAYERS_FILE: skip_file1</code>                                                                                                                                 |
| Contents of file <code>skip_file1</code>             | <pre>PCELL_LAYERS Cell_1 m1 v1 m2 Cell_3 m3 v3 m4</pre>                                                                                                                         |
| Behavior                                             | <p><code>Cell_1</code>: normal PCELL + m1, v1, m2 shapes handling<br/> <code>Cell_2</code>: normal PCELL<br/> <code>Cell_3</code>: not considered for PCELL_LAYERS approach</p> |

## The `TOUCH` Keyword

The `TOUCH` keyword provides a way to recognize polygons in layers other than the named blocking layer as blocking polygons. The StarRC tool first finds polygons in the blocking layer that overlap polygons in the touch layer, then uses the overlapped polygons in the touched layer to block the device layer.

The effective region should contain pins (`*|I`, `*|P`, or probe text instances). An isolated effective region without any pins might cause incorrect results.

If the `TOUCH` and `SIZE` keywords are both used, the StarRC tool first analyzes the relationships defined by the `TOUCH` keyword, then performs the polygon resizing.

For example, [Figure 15-21](#) shows two device layers fabricated within an N-well layer. The following line in a blocking layers section applies to this layout:

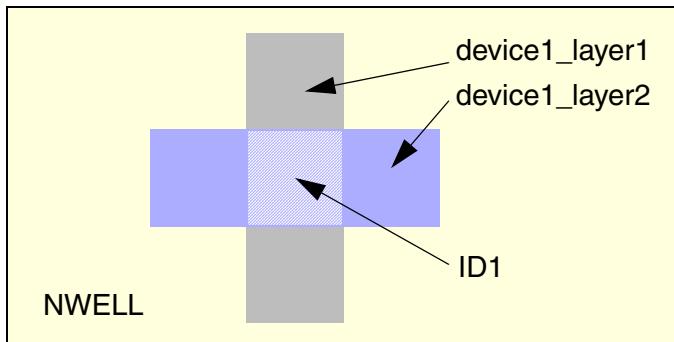
```
NWELL [TOUCH ID1] device1_layer1 device1_layer2
```

Described in terms of design rule checking, the final effective portion of device1\_layer1 for RC extraction is as follows (where `not` and `interact` are operations, not layer names):

```
device1_layer1 not (NWELL interact ID1)
```

In other words, extraction for layers device1\_layer1 and device1\_layer2 is blocked in the region defined by layer ID1, as long as layer NWELL completely overlaps the other layers.

*Figure 15-21 Device1 Blocking Example*



The following line in a blocking layers section applies to the layout in [Figure 15-22](#):

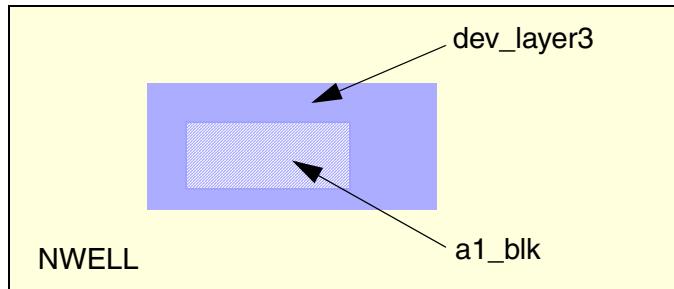
```
NWELL [TOUCH a1_blk] dev_layer3
```

Described in terms of design rule checking, the final effective portion of dev\_layer3 for RC extraction is as follows:

```
dev_layer3 not (NWELL interact a1_blk1)
```

In other words, extraction for layer dev\_layer3 is blocked in the region defined by layer a1\_blk, provided that layer NWELL completely overlaps the other layers.

*Figure 15-22 Device2 Blocking Example*



## See Also

- [SKIP\\_CELLS](#)
- [SKIP\\_PCELLS](#)
- [COUPLE\\_TO\\_PCELL\\_PINS](#)

---

## SLEEP\_TIME\_AFTER\_FINISH

Specifies the CPU wait time before between runs.

### Syntax

SLEEP\_TIME\_AFTER\_FINISH: *number\_of\_seconds*

### Arguments

---

| Argument                 | Description                                                               |
|--------------------------|---------------------------------------------------------------------------|
| <i>number_of_seconds</i> | Specifies the CPU wait time<br>Units: seconds<br>Default: 2<br>Maximum: 2 |

---

### Description

The SLEEP\_TIME\_AFTER\_FINISH command specifies the CPU wait time between runs.

By default, SLEEP\_TIME\_AFTER\_FINISH is set to 2 seconds. The maximum value allowed is 2 seconds.

---

## SPICE\_SUBCKT\_FILE

Specifies the SPICE file containing .subckt definitions for all of the skip cells.

### Syntax

SPICE\_SUBCKT\_FILE: *file1* [... *fileN*]

### Arguments

| Argument                         | Description                                                                                |
|----------------------------------|--------------------------------------------------------------------------------------------|
| <i>file1</i> [... <i>fileN</i> ] | Files containing the .subckt definitions for all skip cells in the design<br>Default: none |

### Description

StarRC reads the files specified by the SPICE\_SUBCKT\_FILE command to obtain port ordering information. The files control the port ordering of the top cell as well. The port order and the port list members read from the .subckt for a skip cell are preserved in the output netlist.

This command does not support the .include statement in the SPICE files.

### See Also

- [SKIP\\_CELLS](#)

---

## STAR\_DIRECTORY

Sets the working directory for the StarRC tool.

### Syntax

STAR\_DIRECTORY: *directory*

### Arguments

| Argument         | Description                                   |
|------------------|-----------------------------------------------|
| <i>directory</i> | The StarRC working directory<br>Default: star |

### Description

The STAR\_DIRECTORY command sets the working directory for StarRC with the following restrictions:

- Absolute or relative paths are permitted.
- A relative path to a directory can only be specified when the directory resides below the working directory in the path.

```
% star_dir/working_dir/other_dir (incorrect)
% working_dir/other_dir/star_dir (correct)
```

The CORNERS\_FILE and STAR\_DIRECTORY command arguments must follow these naming conventions:

- If the STAR\_DIRECTORY command argument is a relative path, you can use either a relative path or an absolute path in the CORNERS\_FILE command. For example:

```
STAR_DIRECTORY: star_work
CORNERS_FILE: smc_config
```

- If the STAR\_DIRECTORY command argument is an absolute path, you must use an absolute path in the CORNERS\_FILE command. For example:

```
STAR_DIRECTORY: /tmp/star
CORNERS_FILE: /remote/.../work_directory/smc_config
```

### See Also

- [CORNERS\\_FILE](#)

---

## STARRC\_DP\_STRING

Enables automatic submission of distributed processing jobs.

### Syntax

```
STARRC_DP_STRING:
 bsub lsf_arguments
 | qsub gridware_arguments
 | list list_of_machines
 | list localhost num_processes
 | nc sub rtida_arguments
```

### Arguments

| Argument                  | Description                                      |
|---------------------------|--------------------------------------------------|
| <i>lsf_arguments</i>      | Arguments for an LSF system                      |
| <i>gridware_arguments</i> | Arguments for a Gridware system                  |
| <i>list_of_machines</i>   | List of machines on a general network            |
| <i>num_processes</i>      | Number of processes on the local host            |
| <i>rtida_arguments</i>    | Arguments for a Runtime Design Automation system |

### Description

This command enables you to start a single run and let the StarRC tool automatically submit multiple jobs. The job submission command can be specified either in an environment variable or as a command in the StarRC command file. If set in both places, the StarRC command file takes precedence. The control parameters in the submission command are site-specific; contact your system administrator for assistance.

The license requirement for this feature is the same as that required by manual submission for the same number of jobs.

Distributed processing is available for the following computing environments:

- LSF system
- Gridware system
- General network with a list of machines
- Runtime Design Automation (RTDA) system

## Examples

On an LSF system:

```
STARRC_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
STARRC_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

On a general network with a list of machines:

```
STARRC_DP_STRING: list alpha beta gamma
```

On an RTDA system:

```
STARRC_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

## See Also

- [NUM\\_CORES](#)
- [Distributed Processing](#)

---

## STOP\_EXTRACTION\_ON\_NUMEROUS\_SHORTS

Specifies whether to stop extraction if a large number of shorts is detected.

### Syntax

STOP\_EXTRACTION\_ON\_NUMEROUS\_SHORTS: YES | NO

### Arguments

| Argument      | Description                                             |
|---------------|---------------------------------------------------------|
| YES (default) | Stops extraction if too many shorts are detected        |
| NO            | Continues extraction even when many shorts are detected |

### Description

The `STOP_EXTRACTION_ON_NUMEROUS_SHORTS` command allows you to choose whether to continue extraction in the presence of shorts.

Runtime can be very long for a design that contains a large number of shorts. This condition might result from an unknown problem in the design data, in which case the appropriate course of action is to stop the run and resolve the problem.

In other cases, extraction data might be desired for an incomplete design, in which case the shorts are expected and can be ignored in the output data. In this case, the appropriate course of action is to continue processing despite the presence of shorts.

By default, StarRC halts execution if both of the following conditions are true:

- The number of shorts is greater than the number of polygons in the partition.
- The number of shorts is greater than 25000.

Set the `STOP_EXTRACTION_ON_NUMEROUS_SHORTS` command to `NO` to continue execution.

---

## SUBSTRATE\_EXTRACTION

Extracts resistance from layers mapped to substrate in the mapping file.

### Syntax

SUBSTRATE\_EXTRACTION: YES | NO

### Arguments

| Argument     | Description                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Performs substrate extraction on layers designated as substrate layers in a <code>conducting_layers</code> section of the mapping file |
| NO (default) | Does not perform substrate extraction                                                                                                  |

### Description

The `SUBSTRATE_EXTRACTION` command enables substrate extraction. You must also set the `TRANSLATE_RETAIN_BULK_LAYERS` command to a value different from the default of `NO` in the StarRC command file.

You must specify each substrate layer in the mapping file by using the keyword `substrate` in a `conducting_layers` section. At least one substrate layer must be defined in the mapping file. You can include `RPSQ` values for each layer, and the values can be different for each layer. Substrate layers without `RPSQ` values are treated as ideal.

Since bulk layers are large and highly resistive, StarRC performs mesh extraction for these layers, resulting in an increased parasitic netlist size.

Use the `SUBSTRATE_EXTRACTION` command to obtain substrate resistance information for the following purposes:

- To prevent shorting of multiple electrical nodes that exist within a common substrate well for the purpose of analyzing power nets having multiple electrical taps to a common well
- For parasitic viewing applications, to enable resistive probing between nodes that share a common well
- To allow the extraction of resistance between a the bulk terminal and the source and drain terminals of a MOS device for the purpose of simulating back-biasing effects

This feature is not intended to facilitate substrate extraction for purposes of substrate noise modeling.

## Example

The following example shows a mapping file:

```
(with substrate extraction)
conducting_layers
 nwell SUBSTRATE RPSQ=1000
 psub SUBSTRATE RPSQ=2000
```

```
(with no substrate extraction)
conducting layers
 nwell SUBSTRATE
 psub SUBSTRATE
```

## See Also

- [RPSQ](#)
- [TRANSLATE\\_RETAIN\\_BULK\\_LAYERS](#)

---

## SUMMARY\_FILE

Specifies the name of the summary file.

### Syntax

`SUMMARY_FILE: file_name`

### Arguments

| Argument               | Description                                                               |
|------------------------|---------------------------------------------------------------------------|
| <code>file_name</code> | The name of the summary file<br>Default: <code>block_name.star_sum</code> |

### Description

The `SUMMARY_FILE` command specifies the name of the summary file generated by StarRC.

By default, the summary file is located in the run directory and has the name `block_name.star_sum`, where `block_name` is the block specified by the `BLOCK` command.

You can use the `SUMMARY_FILE` command to change the name and location of the summary file. This command accepts a path relative to the run directory. However, an absolute path is not permitted because of the possibility of a change in the network file system.

### Example

To create a summary file `my_summary.log` in the `results` subdirectory, use the following syntax in the StarRC command file:

`SUMMARY_FILE: ./results/my_summary.log`

### See Also

- [BLOCK](#)

---

## SUPPORT\_DIFFERENT\_PORTNAME\_NETNAME

Specifies whether to allow different names for ports and the nets connected to those ports.

### Syntax

SUPPORT\_DIFFERENT\_PORTNAME\_NETNAME : YES | NO

### Arguments

| Argument     | Description                                                                        |
|--------------|------------------------------------------------------------------------------------|
| YES          | Supports different port names and net names                                        |
| NO (default) | Does not support a different port name on a net. Different port names are dropped. |

### Description

By default, the StarRC tool assumes that a net connected to a port has the same name as the port. If the names are different, the tool issues a warning message, keeps the net, and does not include the port in the netlist.

For the Calibre Connectivity Interface transistor-level flow, you can set the SUPPORT\_DIFFERENT\_PORTNAME\_NETNAME command to YES to allow ports and the nets connected to them to have different names. In addition, one net can connect to multiple ports, all with different names.

If the NETLIST\_IDEAL\_SPICE\_FILE command is used to create an ideal SPICE file, the new ports are included in the .SUBCKT definitions.

**Table 15-10** shows the effects of the command setting.

*Table 15-10 Different Port Name and Net Name Behavior*

| Port name | Net name | Behavior if NO (default)                                                   | Behavior if YES                                    |
|-----------|----------|----------------------------------------------------------------------------|----------------------------------------------------|
| VSS       | VSS      | .SUBCKT BLOCK VSS ...<br>* NET VSS<br>* P VSS                              | .SUBCKT BLOCK VSS ...<br>* NET VSS<br>* P VSS      |
| VSS       | VSS_1    | .SUBCKT BLOCK VSS_1 ...<br>* NET VSS_1<br>(no * P created; warning issued) | .SUBCKT BLOCK VSS ...<br>* NET VSS_1<br>* P VSS    |
| VSSA      | VSS_1    | .SUBCKT BLOCK VSS_1 ...                                                    | .SUBCKT BLOCK VSSA VSSB VSSC...                    |
| VSSB      |          | * NET VSS_1                                                                | * NET VSS_1                                        |
| VSSC      |          | (no * P created; warning issued)                                           | * P (VSSA ...)<br>* P (VSSB ...)<br>* P (VSSC ...) |

A different port name overrides port names specified in the `SPICE_SUBCKT_FILE` and `PIO_FILE` commands.

## See Also

- [NETLIST\\_IDEAL\\_SPICE\\_FILE](#)
- [SPICE\\_SUBCKT\\_FILE](#)
- [PIO\\_FILE](#)

---

## TARGET\_PWRA

Generates the StarRC commands needed for power reliability analysis.

### Syntax

TARGET\_PWRA: YES | NO

### Arguments

| Argument     | Description                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------|
| YES          | Generates an optimized set of commands for reliability analysis using the StarRC or HSIM flow |
| NO (default) | Does not generate the reliability analysis commands                                           |

### Description

The TARGET\_PWRA command automatically generates StarRC commands for RC extraction in a power reliability analysis flow. With this command, you must also use the POWER\_NETS command to specify a list of power nets.

The TARGET\_PWRA: YES command causes the StarRC tool to generate two netlists. One netlist contains unreduced resistors for power nets. The second netlist contains reduced RC-coupled devices for signal nets. The signal netlist is useful for both reliability analysis and signal timing analysis.

The TARGET\_PWRA: YES command generates the following commands and overrides any commands of the same type that appear elsewhere in the command file:

- POWER\_EXTRACT: RONLY  
Creates an additional resistor-only netlist when the NETLIST\_POWER\_FILE and POWER\_EXTRACT: YES commands are used.
- POWER\_REDUCTION: LAYER\_NO\_EXTRA\_LOOPS  
Specifies that reduction is applied to the specified power nets rather than signal nets.  
This option is specified if no other instance of the POWER\_REDUCTION command appears in the command file or if you set the POWER\_REDUCTION command to YES elsewhere in the command file. However, if you set the POWER\_REDUCTION command to the more conservative NO or LAYER options, those settings are not changed.
- NETLIST\_CONNECT\_SECTION: YES  
Generates \*|I, \*|P, and \*CONN sections in the output file.

- **NETLIST\_NODE\_SECTION: YES**  
Generates \*|S and \*N statements in the output file.
- **EXTRA\_GEOMETRY\_INFO: RES NODE**  
Writes node and resistor geometry information as comments in the netlist.
- **NETLIST\_TAIL\_COMMENTS: YES**  
Enables writing the extra geometry information in the netlist.
- **NETLIST\_FORMAT: SPF**  
Specifies the SPF netlist format.
- **NETLIST\_POWER\_FILE: *block\_name.pwr.spf***  
Specifies the name for the output netlist that contains power rail resistor values.
- **SHORT\_PINS: NO**  
Specifies not to short top-level pin ports that have multiple placements.

You do not need to set the `EXTRA_GEOMETRY_INFO: NODE` command because \*|S is the center of the bounding box node.

If the `TARGET_PWRA: YES` command is used and the `MAX_VIA_ARRAY_SPACING` option of the `via_layers` mapping file command is not set, the StarRC tool sets the value of the `MAX_VIA_ARRAY_SPACING` option as follows:

- If the `AREA` option of the `via_layers` mapping file is set, the `MAX_VIA_ARRAY_SPACING` option value is set to the square root of the `AREA` option value.
- If the `AREA` option is not set but an `RPV_VS_AREA` table exists in the `VIA` statement, the `MAX_VIA_ARRAY_SPACING` parameter is set to the square root of the first area value in the `RPV_VS_AREA` table.
- `AREA` values specified in a mapping file take precedence over `RPV_VS_AREA` values specified in an ITF file.

## Example

```
BLOCK: blockname
MILKYWAY_DATABASE: mw_file
MILKYWAY_EXTRACT_VIEW
TARGET_PWRA:YES
POWER_NETS: list_of_power_nets
TCAD_GRD_FILE: nxt_file
MAPPING_FILE: map_file
XREF: YES
SKIP_CELLS: list_of_cells
NETLIST_FILE: blockname_signal_nets.spf
```

## See Also

- [EXTRA\\_GEOMETRY\\_INFO: NODE](#)
- [KEEP\\_VIA\\_NODES: NO](#)
- [via\\_layers](#)
- [NETLIST\\_CONNECT\\_SECTION](#)
- [NETLIST\\_FORMAT](#)
- [NETLIST\\_POWER\\_FILE](#)
- [NETLIST\\_NODE\\_SECTION](#)
- [NETLIST\\_TAIL\\_COMMENTS](#)
- [POWER\\_EXTRACT](#)
- [POWER\\_NETS](#)
- [POWER\\_REDUCTION](#)
- [REDUCTION](#)
- [SHORT\\_PINS](#)

---

## TCAD\_GRD\_FILE

Specifies the name of the nxtgrd file.

### Syntax

```
TCAD_GRD_FILE: file
TCAD_GRD_FILE: nxtgrd_file1 nxtgrd_file2 nxtgrd_file3
```

### Arguments

| Argument    | Description                               |
|-------------|-------------------------------------------|
| <i>file</i> | Name of the nxtgrd model<br>Default: none |

### Description

This command is mandatory for all extraction flows. It specifies the name of the nxtgrd file containing conductor sheet resistances and models for 3-D parasitic capacitance calculation.

The [MAPPING\\_FILE](#) command specifies a file that maps every process layer in the nxtgrd file to a corresponding layout database layer. For more information about creating the mapping file, see [The Mapping File](#). For information about creating the nxtgrd file, see [Flows for Process Characterization](#).

### See Also

- [MAPPING\\_FILE](#)

---

## TEMPERATURE\_SENSITIVITY

Enables the temperature sensitivity flow for transistor-level extraction.

### Syntax

TEMPERATURE\_SENSITIVITY: YES | NO

### Arguments

| Argument     | Description                               |
|--------------|-------------------------------------------|
| YES          | Enables temperature sensitivity analysis  |
| NO (default) | Disables temperature sensitivity analysis |

### Description

The TEMPERATURE\_SENSITIVITY command writes the parasitic resistor temperature coefficients TC1 (CRT1) and TC2 (CRT2) to the netlist for use by simulation tools.

Temperature coefficients are not reported if they are not set or if they are equal to 0. If the absolute value of TC1 is less than 1e-06, it is reported as 1e-06. If the absolute value of TC2 is less than 1e-09, it is reported as 1e-09.

The format of the coefficients depends on the netlist type, as follows:

- In a DSPF netlist, temperature coefficients are labeled TC1 and TC2.

```
R1 n1:1 n2:2 resStar R=78 TC1=0.0012556 TC2=-1.95301e-07
```

- In a SPEF netlist, temperature coefficients are written using the sensitivity format. The field definitions are written in the \*VARIATION\_PARAMETERS section of the netlist. The index number of the TC1 and TC2 values might vary in different netlists.

```
*VARIATION_PARAMETERS
6 CRT1
7 CRT2 25.0000
*RES
1 p1:A p3:Z 2.50093 *SC 4:0.900 5:0.531 6:0.00321 7:-.000021
```

The TEMPERATURE\_SENSITIVITY command can be used with the simultaneous multicorner flow. In this case, the maximum number of selected process corners is three and the OPERATING\_TEMPERATURE settings in the corners file are ignored. If the selected corners contain redundant nxtgrd files, the StarRC tool discards the redundant corners, determines the temperature coefficients for the remaining corners, and issues a warning message.

All reduction modes are supported.

---

## TOP\_DEF\_FILE

Specifies the top-level block design file in DEF format.

### Syntax

TOP\_DEF\_FILE: *def\_file*

### Arguments

| Argument        | Description                                              |
|-----------------|----------------------------------------------------------|
| <i>def_file</i> | The top block design file in DEF format<br>Default: none |

### Description

This command defines the top-level block for extraction and is mandatory for LEF/DEF flows.

This DEF file can reference macros defined in separate files with the MACRO\_DEF\_FILE command. The standard cell and routing layer definitions should be defined in the accompanying LEF file. Macro blocks appearing in the DEF file specified by the TOP\_DEF\_FILE command are skipped by default.

You can specify gzip files with the TOP\_DEF\_FILE command.

### See Also

- [LEF\\_FILE](#)
- [MACRO\\_DEF\\_FILE](#)

---

## TRANSLATE\_DEF\_BLOCKAGE

Translates the routing DEF blockages from a TOP\_DEF\_FILE.

### Syntax

TRANSLATE\_DEF\_BLOCKAGE: YES | NO

### Arguments

| Argument     | Description                                                         |
|--------------|---------------------------------------------------------------------|
| YES          | Translates the routing DEF BLOCKAGES from a designated top DEF file |
| NO (default) | Does not translate DEF blockages                                    |

### Description

This command translates the routing DEF blockages from a TOP\_DEF\_FILE only. It ignores DEF blockages from the MACRO\_DEF\_FILE. This is because the routing information corresponding to these blockages is already present in the TOP\_DEF\_FILE. Moreover, placement blockages in the TOP\_DEF\_FILE are ignored by this option.

### Example

```
[BLOCKAGES numBlockages ;
 [- LAYER layerName
 [+ COMPONENT compName | + SLOTS | + FILLS | + PUSHDOWN]
 [+ SPACING minSpacing | + DESIGNRULEWIDTH effectiveWidth]
 {RECT pt pt | POLYGON pt pt pt ...} ...
 ;] ...
 [- PLACEMENT
 [+ COMPONENT compName | + PUSHDOWN]
 {RECT pt pt} ...
 ;] ...
```

### See Also

- [MACRO\\_DEF\\_FILE](#)
- [TOP\\_DEF\\_FILE](#)

---

## TRANSLATE\_DRCFILL\_AS\_OBS

Translates DRCFILL polygons in the SPECIALNETS section of a DEF file as if they were OBS polygons.

### Syntax

TRANSLATE\_DRCFILL\_AS\_OBS: YES | NO

### Arguments

| Argument      | Description                                                          |
|---------------|----------------------------------------------------------------------|
| YES (default) | Translates DRCFILL polygons in the SPECIALNETS section of a DEF file |
| NO            | Does not translate DRCFILL polygons                                  |

### Description

Designs for double patterning processes might contain extra polygons to enable the advanced lithography. For designs created in the NDM format by the IC Compiler II tool, the extension polygons are marked internally according to whether they belong to normal nets, unconnected nets, or obstruction (OBS) geometries.

The StarRC tool can read and translate all such polygons when the IC Compiler II logic library is read directly.

However, if a logic library from the IC Compiler II tool is saved in a LEF/DEF format, the polygons from the normal nets are written in the NETS section of the DEF file while the polygons from unconnected or OBS nets are written in the SPECIALNETS section. Using the LEF/DEF version of the design as input to the StarRC tool requires that the extension polygons in the SPECIALNETS section be recognized.

The TRANSLATE\_DRCFILL\_AS\_OBS command instructs the StarRC tool to translate all of the DRCFILL polygons in a DEF file as if they were OBS polygons.

### See Also

- [NDM\\_DATABASE](#)

---

## TRANSLATE\_FLOATING\_AS\_FILL

Considers disconnected floating polygons as fill polygons or connected to ground.

### Syntax

TRANSLATE\_FLOATING\_AS\_FILL: YES | NO

### Arguments

| Argument     | Description                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------|
| YES          | Treats disconnected floating polygons as fill polygons. Their capacitive interaction is accounted as metal fill polygons. |
| NO (default) | Treats disconnected floating polygons as simple ideal ground                                                              |

### Description

StarRC handles floating and grounded metal fill through a separate metal fill GDSII file interface for transistor-level flows. For these flows, StarRC determines the name of a net based on pin-marker definitions from layout versus schematic (LVS) tools. For nets that do not have pin-marker layers or text, LVS tools generally assign a random net ID to these layers. These polygons are considered disconnected or floating. Since these polygons are present in the input database, the StarRC tool must take the polygons into account for capacitive interaction. Resistance is not a concern because there are no pins present and thus no current flowing through these polygons.

The TRANSLATE\_FLOATING\_AS\_FILL command is independent of the METAL\_FILL\_POLYGON\_HANDLING command.

Note:

If you set the TRANSLATE\_FLOATING\_AS\_FILL command to YES, you cannot use an emulated metal fill nxtgrd file.

### See Also

- [METAL\\_FILL\\_GDS\\_FILE](#)

---

## TRANSLATE\_NDM\_BLOCKAGE

Translates the routing blockages from an IC Compiler II design library.

### Syntax

TRANSLATE\_NDM\_BLOCKAGE : YES | NO

### Arguments

| Argument     | Description                                                           |
|--------------|-----------------------------------------------------------------------|
| YES          | Enables translation of the routing blockages from a designated design |
| NO (default) | Disables translation of routing blockages                             |

### Description

The TRANSLATE\_NDM\_BLOCKAGE command enables the translation of the routing blockages from an IC Compiler II design library.

### See Also

- [NDM\\_DATABASE](#)
- [NDM DESIGN VIEW](#)

---

## TRANSLATE\_RETAIN\_BULK\_LAYERS

Specifies how mapped bulk layers are treated during transistor-level extraction.

### Syntax

TRANSLATE\_RETAIN\_BULK\_LAYERS: YES | CONLY [ ALL\_TO\_CLOSEST\_CONTACT ]

### Arguments

| Argument                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES                          | Passes all mapped bulk layers to StarRC for extraction                                                                                                                                                                                                                                                                                                                                                                                                |
| CONLY                        | Passes all mapped bulk layers to StarRC for capacitance-only extraction. Connects device bulk terminals ideally to the port with the same name as the connected net name in the instance section of the netlist, but does not report the terminals in the detailed NET section.<br><br>Also connects nonbulk terminals on SUBSTRATE layers (such as terminals of NWELL resistors or ESD protection diodes) to the closest substrate contacts ideally. |
| CONLY ALL_TO_CLOSEST_CONTACT | Passes all mapped bulk layers to StarRC for capacitance-only extraction, connected to the closest tap. Bulk connections are netlisted as *II instances, allowing back-annotation to bulk nodes.                                                                                                                                                                                                                                                       |

### Description

The TRANSLATE\_RETAIN\_BULK\_LAYERS command specifies how StarRC handles bulk layers during transistor-level extraction. This command has an effect only if the bulk layers are mapped in the mapping file.

A bulk layer is defined as any database layer that is used as the bulk terminal layer for any of the following devices: NMOS, PMOS, resistor, diode, or bipolar junction transistor. A nonbulk layer is any other layer that is physically part of the substrate, such as a well.

Note:

In Calibre flows, bulk layer recognition is applicable to devices that have a device type of MOS in the CALIBRE\_DEVTAB file (including MN, MP, MD, and ME element names) and also to devices that have the element name 'M'. Bulk layer recognition is not applicable to lightly-doped-drain (LDD) devices, because those source and drain pins are not swappable. It is also not applicable to most devices bearing the USER tag in the DEVICE\_TYPE field of the CALIBRE\_DEVTAB file, including inductor devices.

For accurate capacitance extraction, specify the layer precedence in the mapping file. Specifying the precedence is required if more than one database layer is mapped to the substrate and the `TRANSLATE_RETAIN_BULK_LAYERS` command is set to any value other than `NO`.

The command options affect extraction as follows:

- `YES`

StarRC extracts the coupling capacitance to device bulk layers that serve as device terminal layers. In this mode, the tool outputs an instance port subnode for the bulk terminal of the devices in the detailed NET section of the netlist.

- `CONLY`

StarRC extracts the coupling capacitance to device bulk layers that serve as device terminal layers. In this mode, the tool connects device bulk terminals ideally to the port object with the same name as the connected net name in the instance section of the netlist output, but does not report the terminals in the detailed NET section of the netlist.

In addition, device nonbulk terminals on `SUBSTRATE` layers are connected to the closest substrate contacts ideally with a shorting resistor. The tool outputs an instance port subnode for these nonbulk terminals in the detailed NET section of the netlist.

If a net does not have a top level pin (a `*|P` entry), the bulk terminals are connected to the closest contact.

Note:

If the `SHORT_PINS` command is set to `YES` (the default), an additional virtual connection is assumed to exist between multiple same-text pins. Different groups of substrate contacts might have a connection path through the virtual connection.

Using the `TRANSLATE_RETAIN_BULK_LAYERS:CONLY` command might cause different results depending on whether the `SHORT_PINS` command is set to `YES` or `NO`, due to the processing needed to avoid shorting effects.

- `CONLY_ALL_TO_CLOSEST_CONTACT`

StarRC extracts the coupling capacitance to device bulk layers that serve as device terminal layers. In this mode, the bulk terminal is connected to the closest tap (contact between the substrate and a low-level metal layer). As a result, the bulk connection is reported as a `*||I` instance in the netlist, allowing back-annotation on bulk nodes by simulation tools.

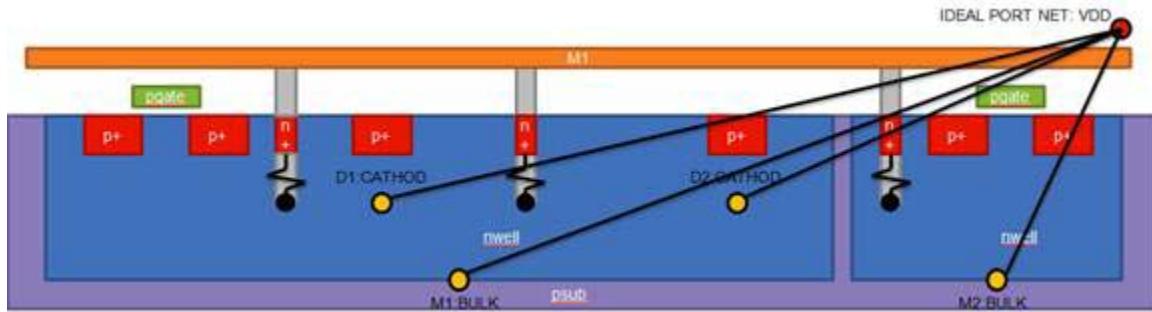
If multiple floating taps connect to a bulk layer, only one tap is retained for every isolated connection.

[Figure 15-23](#) and [Figure 15-24](#) illustrate the ways that the bulk nodes can be connected. The `TRANSLATE_RETAIN_BULK_LAYERS:CONLY` command corresponds to [Figure 15-23](#). The bulk layers are extracted for capacitance and connected to an ideal net.

**Note:**

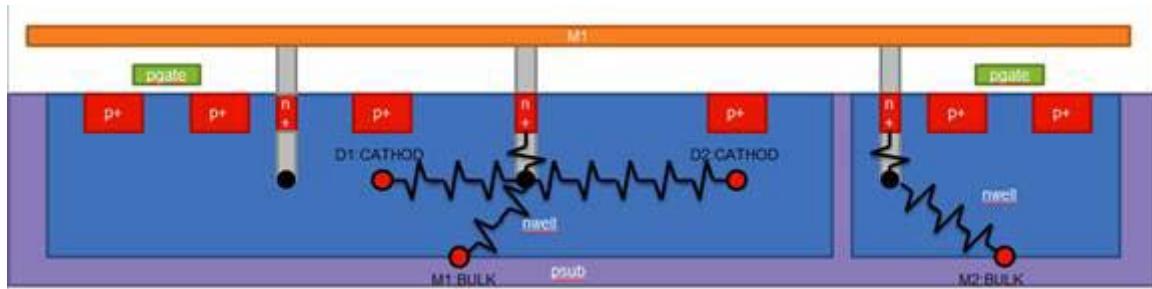
If you want to retain all of the substrate contacts (instead of the default of one contact) for certain models when using the `TRANSLATE_RETAIN_BULK_LAYERS:CONLY` command, use the `KEEP_SUBCONT_MODELS` command to specify the models of interest.

*Figure 15-23 Effect of the CONLY Option*



The `TRANSLATE_RETAIN_BULK_LAYERS:CONLY ALL_TO_CLOSEST_CONTACT` command corresponds to [Figure 15-24](#). The bulk layers are extracted for capacitance and connected to the closest tap.

*Figure 15-24 Effect of the CONLY ALL\_TO\_CLOSEST\_CONTACT Option*



## Examples

The following lines show an excerpt from a netlist generated using the `TRANSLATE_RETAIN_BULK_LAYERS: CONLY` command:

```
* | NET VDD 0.00463746PF
* | I (M2:SRC M2 SRC B 0 5 35.75)
Cg6 VDD:1 GND 1.55208e-15
...
*
* Instance Section
*
MM2 M2:DRN M2:GATE M2:SRC VDD nch ...
```

The following lines show an excerpt from a netlist generated using the TRANSLATE\_RETAIN\_BULK\_LAYERS: CONLY ALL\_TO\_CLOSEST\_CONTACT command:

```
* | NET VDD 0.00463746PF
* | I (M2:BULK M2 BULK B 0 5.5 35.75)
* | I (M2:SRC M2 SRC B 0 5 35.75)
Cg6 VDD:1 GND 1.55208e-15
...
*
*Instance Section
*
MM2 M2:DRN M2:GATE M2:SRC M2:BULK nch ...
```

## See Also

- [COUPLE\\_TO\\_GROUND](#)
- [SHORT\\_PINS](#)
- [KEEP\\_SUBCONT\\_MODELS](#)

---

## TRANSLATE\_VIA\_PINS

Specifies whether StarRC translates incomplete via pins to real vias.

### Syntax

TRANSLATE\_VIA\_PINS: YES | NO

### Arguments

| Argument      | Description                                                                   |
|---------------|-------------------------------------------------------------------------------|
| YES (default) | Translates via pins to real vias                                              |
| NO            | Does not translate via pins to real vias. There is no impact on connectivity. |

### Description

The TRANSLATE\_VIA\_PINS command specifies whether StarRC translates via pins to real vias in the Milkyway flow.

### Example

When you use the following command in the StarRC command file, StarRC does not translate incomplete vias in the FRAM view. An incomplete via is a via with a missing upper-layer net name or lower-layer net name.

TRANSLATE\_VIA\_PINS: NO

---

## TRENCH\_CONTACT\_VIRTUAL\_VIA\_SEGMENTATION\_RATIO

Specifies the segmentation ratio of virtual vias in a trench contact process.

### Syntax

TRENCH\_CONTACT\_VIRTUAL\_VIA\_SEGMENTATION\_RATIO: *ratio*

### Arguments

| Argument     | Description                                                                              |
|--------------|------------------------------------------------------------------------------------------|
| <i>ratio</i> | Segmentation ratio represented as a floating-point number<br>Units: none<br>Default: 2.0 |

### Description

Trench contacts can have tall covertial layers that are not connected by physical vias. To extract vertical resistance, virtual vias are inserted between the trench contact conductors. The StarRC tool segments these vias automatically to create a distributed resistance network.

The segmentation ratio is a floating-point number calculated as follows:

$$\text{ratio} = (\text{via length}) / (\text{via width})$$

The height of the virtual via must be less than or equal to 2 nm.

### Errors

You can use the TRENCH\_CONTACT\_VIRTUAL\_VIA\_SEGMENTATION\_RATIO command without the MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER command. However, if you use the MAX\_VIRTUAL\_VIA\_SEGMENTATION\_NUMBER command alone, the StarRC tool issues an error message.

### Example

TRENCH\_CONTACT\_VIRTUAL\_VIA\_SEGMENTATION\_RATIO: 2.5

### See Also

- [MAX\\_VIRTUAL\\_VIA\\_SEGMENTATION\\_NUMBER](#)

---

## TSV\_ CELLS

Specifies the TSV cells used for subcircuit replacement in the 3-D IC flow.

### Syntax

TSV\_ CELLS: *tsvcell\_1* *tsvcell\_2* *tsvcell\_3* ...

### Arguments

| Argument             | Description                                                  |
|----------------------|--------------------------------------------------------------|
| <i>tsvcell_1</i> ... | TSV cells used for subcircuit replacement in the 3-D IC flow |

### Description

The TSV\_CELLS command specifies the TSV cells used for subcircuit replacement in the 3-D IC flow.

### Example

The following syntax specifies the TSV cells used for subcircuit replacement.

TSV\_ CELLS: *tsvcell\_1* *tsvcell\_2*

### See Also

- [TSV](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [Through-Silicon Via Extraction](#)

---

## USER\_DEFINED\_DIFFUSION\_RES

Enables user-defined diffusion resistance calculation.

### Syntax

USER\_DEFINED\_DIFFUSION\_RES: YES | NO

### Arguments

| Argument      | Description                                           |
|---------------|-------------------------------------------------------|
| YES (default) | Enables user-defined diffusion resistance calculation |
| NO            | Disables all use of user-defined diffusion resistance |

### Description

The `USER_DEFINED_DIFFUSION_RES` command enables the calculation of diffusion resistance by a method that is more accurate for advanced process nodes than the standard mesh resistance calculation.

#### Note:

This implementation of user-defined diffusion resistance is not compatible with the feature of the same name available in earlier StarRC versions. To use this feature, you must use an nxtgrd file created in StarRC version L-2016.06 or later.

To use this analysis, perform the following steps:

1. Set the `USER_DEFINED_DIFFUSION_RES` command to `YES` in the StarRC command file. (This step is optional because the command defaults to `YES`.)
2. Include the `USER_DEFINED_DIFFUSION_RESISTANCE` command within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
3. Set the `LAYER_TYPE` keyword to `GATE` within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
4. Specify a model name by using the `diffusion_res_model` keyword in the `conducting_layers` command in the mapping file

### See Also

- [USER\\_DEFINED\\_DIFFUSION\\_RESISTANCE](#)
- [conducting\\_layers](#)
- [User-Defined Diffusion Resistance](#)

---

## VIA\_COVERAGE

Specifies six values from the via edge to outer metal edge for coverage and landing measurement.

### Syntax

VIA\_COVERAGE: *vial Lf Lq [Ls] Cf Cq [Cs]*

### Arguments

| Argument  | Description                                                            |
|-----------|------------------------------------------------------------------------|
| <i>Lf</i> | Maximum number for a landing full measurement<br>Units: nanometers     |
| <i>Lq</i> | Maximum number for a landing quarter measurement<br>Units: nanometers  |
| <i>Ls</i> | Maximum number for a landing semi measurement<br>Units: nanometers     |
| <i>Cf</i> | Maximum number for a coverage full measurement<br>Units: nanometers    |
| <i>Cq</i> | Maximum number for a coverage quarter measurement<br>Units: nanometers |
| <i>Cs</i> | Maximum number for a coverage semi measurement<br>Units: nanometers    |

---

### Description

Each number corresponds to a coverage or landing measurement.

The VIA\_COVERAGE command checks square vias; the VIA\_COVERAGE\_OPTION\_FILE command checks rectangular vias.

Note:

The NETLIST\_TAIL\_COMMENTS:YES command is required; the NETLIST\_FORMAT:SPEF command is not required. The VIA\_COVERAGE command does not require a text file.

All netlist formats are accepted by this command.

Each `VIA_COVERAGE` command must have all six entries to include the semicoverage capability. You can specify four numbers to get results without the semicoverage capability. Both are reported in the netlist under the heading “`VIA_COVERAGE_CODES`.”

The coverage and landing units are nanometers and represent as-drawn dimensions, before the application of any `MAGNIFICATION_FACTOR` command.

The full coverage and landing values must be greater than the semicoverage and landing values. All vias specified for this feature must also be defined in the ITF file.

## Example

This example specifies the measurement of via coverage including semicoverage values.

```
NETLIST_FORMAT: SPF
NETLIST_TAIL_COMMENTS: YES
VIA_COVERAGE: via1 100 80 100 80
VIA_COVERAGE: via2 100 80 100 80
```

## See Also

- [MERGE\\_VIAS\\_IN\\_ARRAY](#)
- [NETLIST\\_FORMAT](#)
- [NETLIST\\_TAIL\\_COMMENTS](#)
- [VIA\\_COVERAGE\\_OPTION\\_FILE](#)

---

## VIA\_COVERAGE\_OPTION\_FILE

Specifies a file that contains via checking rules for rectangular vias.

### Syntax

VIA\_COVERAGE\_OPTION\_FILE: *file\_name*

### Arguments

| Argument         | Description                  |
|------------------|------------------------------|
| <i>file_name</i> | The via coverage option file |

### Description

The VIA\_COVERAGE\_OPTION\_FILE command provides the name of a file that contains rules for checking rectangular vias. The command can be used in the StarRC command file or optionally within a corners file for a simultaneous multicorner flow.

The VIA\_COVERAGE\_OPTION\_FILE command checks rectangular vias; the VIA\_COVERAGE command checks square vias. The file is a text file containing via check ranges.

The via coverage report in the netlist contains a table that shows the detail of the via coverage results. These results are determined by rules inside of StarRC that read and analyze your via coverage data. The via coverage results are shown as full coverage, quarter coverage, semicoverage, and partial coverage.

A via satisfies the area check of a drawn box if the box area in the figure is filled with metal polygons. This applies to both coverage and landing. For coverage, the metal layer refers to the metal layer above the via layer (for example, metal2 for via12) and for the landing it refers to the metal layer below the via layer (for example metal1 for via 12).

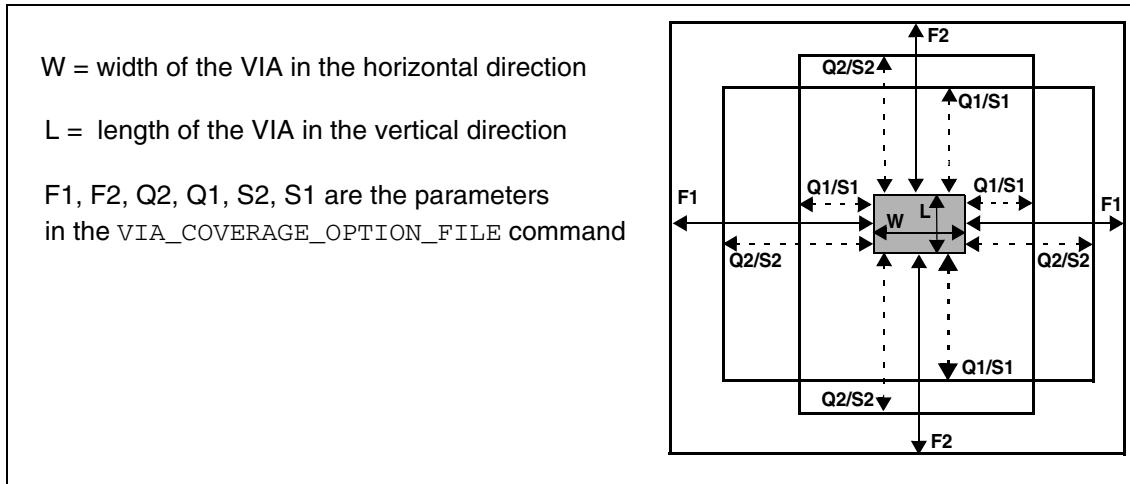
The default dbunit for the values in the via coverage option file is 1000 (i.e. the units are nanometers).

The output netlist contains the following via classifications:

- FULL means all sides of the via are covered because all enclosures are greater than the F parameter (as shown in [Figure 15-25](#)).
- QUARTER means one enclosure must be greater than or equal to Q1 and must have both adjacent sides enclosed by greater than Q2.
- SEMI means one enclosure must be greater than or equal to S1 and both adjacent sides must be enclosed by more than S2.
- PARTIAL means no enclosures meet the full, quarter, or semicoverage requirements.

Specify three numbers for coverage and three numbers for landing when you are not checking semicoverage and landing. Specify five numbers for coverage and five numbers for landing when you are checking semicoverage and landing.

*Figure 15-25 Via Coverage*



The `VIA_COVERAGE_OPTION_FILE` command rules are as follows:

- Non-Manhattan shapes are not supported.
- For via arrays, all inside vias (those not on the perimeter) are considered fully covered.
- The horizontal direction is equal to the direction of the x-axis of coordinates used by the StarRC extraction.
- The vertical direction is equal to the direction of the y-axis coordinates used by StarRC extraction.
- Via coverage parameters must meet the following conditions, which apply to both coverage and landing parameters and are checked in StarRC:
  - $F$  must be greater than or equal to  $Q2$ .
  - $F$  must be greater than or equal to  $S2$ .
  - $Q2$  must be greater than  $S2$ .
  - $Q2$  must be greater than  $Q1$ .
  - $S2$  must be greater than or equal to  $S1$ .

The following table lists rules for the parameters.

| Keyword | Description                                                    |
|---------|----------------------------------------------------------------|
| Xrange  | Width of the via contact                                       |
| Xmin    | Minimum width value of the via contact                         |
| Xmax    | Maximum width value of the via contact                         |
| Yrange  | Length of the via contact                                      |
| Ymin    | Minimum length value of the via contact                        |
| Ymax    | Maximum length value of the via contact                        |
| FL1     | Full coverage y value for via landing                          |
| FL2     | Full coverage x value for via landing                          |
| FC1     | Full coverage y value for via cover                            |
| FC2     | Full coverage x value for via cover                            |
| QL1     | Quarter coverage value for landing (small enclosure value)     |
| QC1     | Quarter coverage value for via cover (small enclosure value)   |
| QL2     | Quarter coverage value for via landing (large enclosure value) |
| QC2     | Quarter coverage value for via cover (large enclosure value)   |
| SL1     | Semicoverage value for via landing (small enclosure value)     |
| SC1     | Semicoverage value for via cover (small enclosure value)       |
| SL2     | Semicoverage value for via landing (large enclosure value)     |
| SC2     | Semicoverage value for via cover (large enclosure value)       |

When the `VIA_COVERAGE_OPTION_FILE` command is used in a corners file for a simultaneous multicorner extraction, the following rules apply:

- If two corners have the same nxtgrd file, they must use the same via coverage option file. However, different nxtgrd files can use the same via coverage option file.
- The only variations allowed within the via coverage option files are RPV variations.

- If any corner has a via coverage option file, all corners must have such a file.
- If via coverage option files are defined in the corners file, a `VIA_COVERAGE_OPTION_FILE` command in the global command file is ignored.
- If no via coverage option files are defined in the corners file, a `VIA_COVERAGE_OPTION_FILE` command in the global command file is applied.
- Via resistance is calculated based on the RPV for each process corner. If temperature deration is defined for the associated via layer, the deration is applied to each corner-dependent RPV value.

## Examples

In the following example files, Xrange and Yrange represent the as-drawn length in X and Y dimension of the via (before any scale factor has been applied). The ranges must not be overlapping, and, each via size should be map-able to exactly one of the data sets in the list. The range specification for a via landing and via coverage is the same. These requirements are checked in the tool. No interpolation or extrapolation is needed. If -during a via coverage mode extraction, a via is found that cannot be mapped to one of the specified ranges, then StarRC issues a warning and the via has an “invalid” via coverage code, which is 0. There is no limit to the number of data set ranges.

### Text File Syntax Single Parameter

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing = FL,QL1,QL2,[SL1,SL2];
Coverage = FC,QC1,QC2,[SC1,SC2]}
(Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;
Landing = FL,QL1,QL2,[SL1,SL2];Coverage = FC,QC1,QC2,[SC1,SC2])
```

### Text File Syntax With Two Parameters

Without semicoverage extraction

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing = FL1,FL2,QL1,QL2;
Coverage = FC1,FC12,QC1,QC2}
```

With semicoverage extraction

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;
Landing = FL1,FL2,QL1,QL2,[SL1,SL2];Coverage = FC1,FC2,QC1,QC2,[SC1,SC2]}
```

### File Format Example 1 - With Semicoverage

```
VIA1 { Xrange= 100,100; Yrange = 100,100;
Landing = 100,80,10,40,10;Coverage = 100,80,10,40,10}
```

### File Format Example 1 - Without Semicoverage

```
VIA1 { Xrange= 100,100; Yrange = 100,100;
Landing = 100,80,10;Coverage = 100,80,10}
```

## See Also

- [NETLIST\\_FORMAT](#)
- [NETLIST\\_TAIL\\_COMMENTS](#)
- [REDUCTION](#)
- [VIA\\_COVERAGE](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)

---

## WIDE\_DEVICE\_TERM\_RESISTANCE

Activates equipotential line node handling for RES devices.

### Syntax

WIDE\_DEVICE\_TERM\_RESISTANCE: RES

### Arguments

| Argument       | Description                                                |
|----------------|------------------------------------------------------------|
| RES            | Activates equipotential line node handling for RES devices |
| NONE (default) | Disables this feature                                      |

### Description

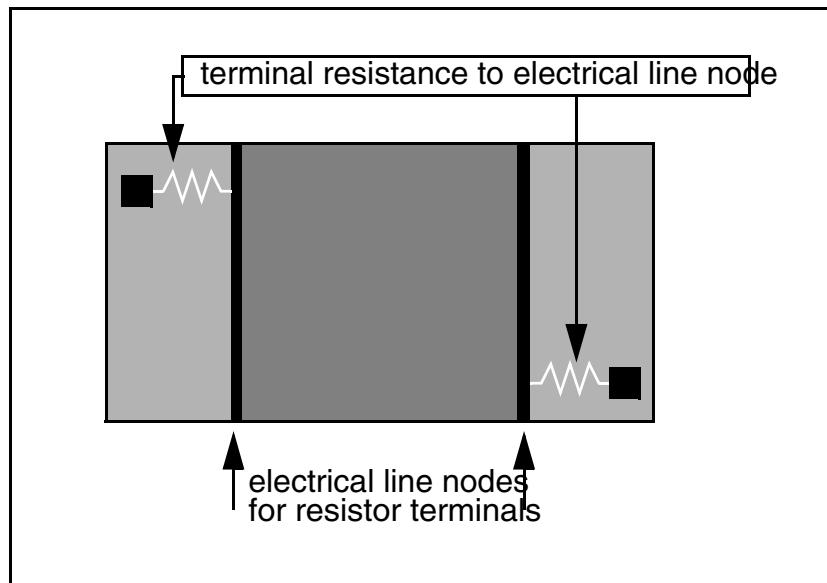
This command activates equipotential line node handling for RES devices. For example, devices extracted with a RES command in Hercules or devices containing an element\_name = R in Calibre. Specifically, this command forces StarRC to always treat the A and B terminal nodes as electrical line nodes (as opposed to electrical point nodes), which is the default behavior. With this treatment, StarRC identifies the terminal or body boundary line and extracts parasitic resistance orthogonal to that line.

For a resistor device, an electrical point node is a physical approximation that assumes all current is concentrated at a single point instead of being distributed along the width of the material. For a device whose width is small relative to its length, this approximation is appropriate for default extraction flows. However, when the width of the device is large relative to its length, as shown in [Figure 15-26](#), the impact of current distribution along the entire terminal or body boundary must be considered during the parasitic resistance extraction.

If RES is specified, StarRC first identifies the A and B terminal layers for all the RES devices extracted by LVS tools. Since LVS tools always put a RES device instance port on the butting edge of the terminal shape and the RES device body shape, StarRC can determine the current direction for the terminal shape to be perpendicular to the butting edge when extracting the resistance of the terminal shapes. It is assumed that the RES terminal contains one shape with its RES instance port lying on one edge. If the terminal contains multiple shapes, only the shape touched by the instance port is treated. The other shapes resume normal processing.

This command does not apply to bulk terminal layers of RES devices.

Figure 15-26 Line Nodes for Resistor Terminals



---

## XREF

Specifies the set of names used for netlist generation and analysis flows.

### Syntax

`XREF: NO | YES | COMPLETE`

### Arguments

| Argument     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO (default) | Reports the layout net names generated by Hercules or Calibre during ideal layout extraction. These layout names are either generated or derived from the application of text. The layout instance names can either be the original GDSII instance name (if the ASSIGN_PROPERTY construct in Hercules is used), or names generated by Hercules.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| YES          | Reports all layout nets and devices occurring in the ideal layout netlist using schematic, net, and instance names wherever possible. When nets or devices exist that were not successfully matched during LVS, layout names are used. When a successful match did occur during LVS, StarRC uses schematic names. StarRC handles composite device merging by using schematic instance names with modifiers attached whenever layout devices outnumber schematic devices within a particular composite device pairing.                                                                                                                                                                                                                                                                                                                                           |
| COMPLETE     | <p>Reports every SKIP_CELLS/device in the schematic. XREF : COMPLETE is only valid in the Hercules flow. The netlist includes all instances. If you do not want this, make sure that any net selected with the NETS command is also included in NETLIST_SELECT_NETS.</p> <p>Parasitics are netlisted only for nets that were successfully cross-referenced to schematic nets. Nets that do not cross-reference to a schematic net are treated as ideal connections by StarRC. Schematic model names are reported for SKIP_CELLS and primitive devices.</p> <p>Internal nets of the SERIES or PATHS merged devices do not have * NET sections in XREF : COMPLETE; they are netlisted ideally in the instance section.</p> <p>Layout property merging for XREF : COMPLETE applies only to standard SPICE properties. Nonstandard properties cannot be merged.</p> |

### Description

A GDS-based device-level extraction database contains both layout names and cross-referenced schematic names if a layout versus schematic verification was performed. The `XREF` command specifies which set of names to provide to StarRC for netlist creation and analysis flows. It also determines which devices and nets to retain.

**See Also**

- [MILKYWAY\\_EXTRACT\\_VIEW](#)
- [Cross-Referencing in Transistor-Level Flows](#)

---

## XREF\_FEEDTHRU\_NETS

Specifies whether to output pure layout feed-through nets and ports in the XREF: COMPLETE netlist.

### Syntax

XREF\_FEEDTHRU\_NETS: YES | NO

### Arguments

| Argument     | Description                                                                    |
|--------------|--------------------------------------------------------------------------------|
| YES          | Generates pure layout feed-through nets and ports in the XREF:COMPLETE netlist |
| NO (default) | Does not generate feed-through nets and ports                                  |

### Description

This command specifies whether to output pure layout feed-through nets and ports in the XREF: COMPLETE netlist. These are routes that cross a hierarchical boundary but whose ports were excluded from LVS, because they have no correspondence in the schematic netlist.

This command has no effect on XREF arguments other than COMPLETE. Pure layout feed-through nets are always netlisted in the other XREF modes, because the other modes are layout-based.

#### Note:

The CREATE\_PORTS option must be enabled in the Hercules runset for XREF\_FEEDTHRU\_NETS: YES.

### See Also

- [XREF](#)

---

## XREF\_LAYOUT\_INST\_PREFIX

Specifies a prefix for layout device instance names that were not cross-referenced to a schematic device by the LVS tool.

### Syntax

XREF\_LAYOUT\_INST\_PREFIX: *prefix*

### Arguments

| Argument      | Description                                        |
|---------------|----------------------------------------------------|
| <i>prefix</i> | Prefix used for netlist generation<br>Default: ld_ |

### Description

This prefix is applicable only for layout-based XREF mode YES, which generates a netlist for every layout element whether or not it was cross-referenced. Device instances that have no LVS comparison information, such as filtered layout instances, are netlisted with this prefix followed by the layout instance name.

### See Also

- [XREF\\_LAYOUT\\_NET\\_PREFIX](#)
- [XREF](#)

---

## XREF\_LAYOUT\_NET\_PREFIX

Sets a prefix for layout net names that were not cross-referenced to a schematic net by the LVS tool.

### Syntax

XREF\_LAYOUT\_NET\_PREFIX: *prefix*

### Arguments

| Argument      | Description                                                     |
|---------------|-----------------------------------------------------------------|
| <i>prefix</i> | Prefix used for net names in netlist generation<br>Default: ln_ |

### Description

This prefix is applicable only for the layout-based XREF mode YES , which generates a netlist for every layout element whether or not it was cross-referenced. Nets that have no LVS comparison information such as dangling and floating nets are output with this prefix followed by the layout net name.

### See Also

- [XREF\\_LAYOUT\\_INST\\_PREFIX](#)
- [XREF](#)

---

## XREF\_SWAP\_MOS\_SD\_PROPERTY

Specifies a pair of MOS properties to be swapped.

### Syntax

XREF\_SWAP\_MOS\_SD\_PROPERTY: *prop1 prop2 [model\_name]*

### Arguments

| Argument           | Description                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------|
| <i>prop1 prop2</i> | A pair of properties to be swapped                                                                |
| <i>model_name</i>  | Specifies that <i>prop1</i> and <i>prop2</i> are swapped only for <i>model_name</i> device models |

### Description

This command specifies a pair of MOS properties to be swapped. The pair of properties has the same swapping behavior as generic MOS source and drain properties such as, ad, ps, and pd.

StarRC automatically swaps the following properties:

(as, ad) (ps, pd) (nrs, nrd) (rsc, rdc)  
(asej, adej) (psej, pdej) (aseo, adeo) (pseo, pdeo)

The XREF\_SWAP\_MOS\_SD\_PROPERTY statement affects SPICE, standard parasitic format, and any other netlist format that has instances with properties. It does not affect SPEF files, which do not include device parameters.

### Example

To restrict the property swapping to a specific device model, specify the model name in the XREF\_SWAP\_MOS\_SD\_PROPERTY statement as shown in the following example:

XREF\_SWAP\_MOS\_SD\_PROPERTY: as5 ad5 nch\_mac5

### See Also

- [MODEL\\_TYPE](#)
- [XREF](#)

---

## XREF\_USE\_LAYOUT\_DEVICE\_NAME

Specifies whether to use layout device names when the XREF command is set to YES.

### Syntax

XREF\_USE\_LAYOUT\_DEVICE\_NAME: YES | NO

### Arguments

| Argument     | Description                                                                                                                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YES          | Netlist layout model names for all devices with XREF: YES. A layout model name is a primary device model name used in a device extraction command in a Hercules, IC Validator, or Calibre rule file.                                                                                                   |
| NO (default) | Netlist schematic model names for all devices with XREF: YES; continue to use layout model names for XREF: NO. A schematic model name is a device model name that occurs in a schematic netlist used for an LVS (Hercules flow) or that occurs in NETLIST MODEL or TEXT MODEL commands (Calibre flow). |

### Description

Device model names recognized by StarRC can originate from one of two places:

- Schematic model name from a Hercules flow or NETLIST MODEL rule-file command from a Calibre flow.
- Layout model name specified in device extraction command

### See Also

- [XREF](#)

---

## ZONE\_COUPLE\_TO\_NET

Couples noncritical material outside a defined macro to the specified net.

### Syntax

ZONE\_COUPLE\_TO\_NET: *net\_name*

### Arguments

| Argument        | Description                                                                         |
|-----------------|-------------------------------------------------------------------------------------|
| <i>net_name</i> | The net name to the noncritical material outside the defined macro<br>Default: none |

### Description

This command is analogous to the SKIP\_CELLS\_COUPLE\_TO\_NET command, except that it applies to overhead or adjacent material when you are doing the in-context extraction of a macro or using the RING\_AROUND\_THE\_BLOCK in-context approximation.

You must specify the NETLIST\_FORMAT: SPEF and COUPLE\_TO\_GROUND: NO commands to use this command.

### See Also

- [ZONE\\_COUPLE\\_TO\\_NET\\_LEVEL](#)

---

## ZONE\_COUPLE\_TO\_NET\_LEVEL

Specifies whether to append the layer number of the material outside the macro.

### Syntax

ZONE\_COUPLE\_TO\_NET\_LEVEL: YES | NO

### Arguments

| Argument     | Description                                                       |
|--------------|-------------------------------------------------------------------|
| YES          | Appends the layer number of the material outside a macro.         |
| NO (default) | Does not append the layer number of the material outside a macro. |

### Description

This command appends the ZONE\_COUPLE\_TO\_NET name to the real layer number for the SKIP\_CELLS material.

This command is ignored unless the ZONE\_COUPLE\_TO\_NET command is specified.

### See Also

- [ZONE\\_COUPLE\\_TO\\_NET](#)

# 16

## ITF Statements

---

The Interconnect Technology Format (ITF) file defines a cross section profile of the process. The ITF file consists of an ordered list of conductor and dielectric layer definition statements. The layers are defined from the topmost dielectric layer to the bottommost dielectric layer, excluding the substrate, in a way that is consistent with the physical manufacturing process.

This chapter describes the ITF statements and their options. Use the ITF statements to create an ITF file with the following structure:

```
TECHNOLOGY = process_name
[GLOBAL_TEMPERATURE = temp_value]
[BACKGROUND_ER = value]
[HALF_NODE_SCALE_FACTOR = scale_factor]
[USE_SI_DENSITY = YES | NO]
[DROP_FACTOR_LATERAL_SPACING = value]
[REFERENCE_DIRECTION = VERTICAL | HORIZONTAL | NONE | GATE]

DIELECTRIC top_dielectric_name ...
CONDUCTOR top_conductor_name ...
...
[DIELECTRIC bottom_dielectric_name ...]

VIA top_via_name ...
...
VIA bottom_via_name ...
```

## Nomenclature

The terms statement, command, option, and keyword are synonymous. The term block refers to a group of related statements.

### Delimiters in Lists

In lists and matrixes, both commas and spaces are accepted as delimiters between values. Spaces are the preferred delimiters. Line feeds, carriage returns, tabs, and other nonprinting characters are ignored.

### Comments in an ITF File

In an ITF file, a dollar sign (\$) at the beginning of a line indicates a comment that is not interpreted by the extraction tool.

### Restrictions for Layer Names

Layer names must obey the following restrictions:

- Layer names are case-sensitive.
- Names must contain only alphanumeric characters and underscores (\_) unless otherwise noted.
- Names must begin with an alphabetic character.

---

## AREA

Specifies the default area of a via. Valid within a `VIA` or `TSV` block.

### Syntax

`AREA = via_area`

### Arguments

| Argument              | Description                                                     |
|-----------------------|-----------------------------------------------------------------|
| <code>via_area</code> | Area of default via<br>Units: square microns<br>Default: 1.0e-6 |

### Description

Within a `VIA` block, the resistive properties of a standard via layer (not a trench contact via layer) must be specified with one of three mutually exclusive methods:

- Using the `RHO` keyword
- Using the `RPV` and `AREA` keywords (these keywords cannot be used alone in a `VIA` block)
- Using an `RPV_VS_AREA` table to specify resistance values for different via areas

Within a `TSV` block, the `AREA` keyword is a required keyword.

### Example

```
VIA vial { FROM=m1 TO=m2 AREA=4 RPV=0.25 }
```

### See Also

- [RHO](#)
- [RPV](#)
- [RPV\\_VS\\_AREA](#)
- [VIA](#)
- [TSV](#)

---

## ASSOCIATED\_CONDUCTOR

When used within a `DIELECTRIC` block, names a conductor layer associated with a conformal dielectric layer. When used within a `CONDUCTOR` block, specifies which ITF layer is used to create trench contact extensions.

### Syntax

```
ASSOCIATED_CONDUCTOR = layer_name
```

### Arguments

| Argument                | Description                     |
|-------------------------|---------------------------------|
| <code>layer_name</code> | Associated conductor layer name |

### Description

When used within a `DIELECTRIC` block, the `ASSOCIATED_CONDUCTOR` keyword names a specific conductor layer to which a dielectric layer conforms. The following usage notes apply:

- Only one `ASSOCIATED_CONDUCTOR` keyword is allowed within a `DIELECTRIC` block.
- The `ASSOCIATED_CONDUCTOR` keyword can only be used with the `IS_CONFORMAL` keyword. However, the `IS_CONFORMAL` keyword can be used without the `ASSOCIATED_CONDUCTOR` keyword. When no `ASSOCIATED_CONDUCTOR` keyword is specified for an `IS_CONFORMAL` layer, the default is to measure from the top layer.
- Using the `IS_CONFORMAL` and `ASSOCIATED_CONDUCTOR` keywords is mutually exclusive with using the `MEASURED_FROM` keyword within the same `DIELECTRIC` block.
- When an `ASSOCIATED_CONDUCTOR` layer drops because of a `DROP_FACTOR` defined for a layer below it, the related `IS_CONFORMAL` dielectric layers also drop.
- The conductor named in the `ASSOCIATED_CONDUCTOR` keyword cannot be higher than the dielectric named in the `IS_CONFORMAL` keyword.
- If a conductor above a conformal dielectric layer overlaps with the dielectric layer top wall thickness, the conductor cuts into the dielectric layer.

When used within a `CONDUCTOR` block, the `ASSOCIATED_CONDUCTOR` keyword names a layer to use for the trench contact extensions. The following usage notes apply:

- In multigate models, trench contact is created only for a conductor layer that has the `LAYER_TYPE` keyword set to `TRENCH_CONTACT` and that connects to the diffusion layer in the `GATE_DIFFUSION_LAYER_PAIR` list in the `MULTIGATE` block.

- If the trench contact conductor block has an ASSOCIATED\_CONDUCTOR keyword, the named associated layer is used for the trench contact extension.
- If the ASSOCIATED\_CONDUCTOR keyword is missing, the trench contact extension is created using the same layer as the trench contact.

### Example

```
DIELECTRIC D1 {
 IS_CONFORMAL
 ASSOCIATED_CONDUCTOR=met1
 SW_T=0.1 TW_T=0.1 ER=2.5
}
```

### See Also

- [IS\\_CONFORMAL](#)
- [DIELECTRIC](#)
- [LAYER\\_TYPE](#)
- [CONDUCTOR](#)

---

## BACKGROUND\_ER

Specifies the relative permittivity (dielectric constant) of the background dielectric.

### Syntax

```
BACKGROUND_ER = relative_permittivity
```

### Arguments

| Argument                     | Description                           |
|------------------------------|---------------------------------------|
| <i>relative_permittivity</i> | Relative permittivity<br>Default: 1.0 |

### Description

The BACKGROUND\_ER statement is an optional statement that can be included in the global parameters section of the ITF file. If the BACKGROUND\_ER statement is not specified, the relative permittivity of the background dielectric is set to 1.0, the relative permittivity of air.

The background dielectric globally fills the cross section to an infinite height, effectively replacing air as the operating medium for the chip.

Relative permittivity settings within individual DIELECTRIC blocks override the global background value.

### Example

```
TECHNOLOGY = example_tech
GLOBAL_TEMPERATURE = 31.0
BACKGROUND_ER = 4.1
```

---

## BOTTOM\_DIELECTRIC\_ER

Specifies the relative permittivity of the dielectric region below a conductor. Valid within a CONDUCTOR block.

### Syntax

`BOTTOM_DIELECTRIC_ER = permittivity`

### Arguments

| Argument                         | Description                             |
|----------------------------------|-----------------------------------------|
| <code><i>permittivity</i></code> | Relative permittivity of the dielectric |

### Description

The `BOTTOM_DIELECTRIC_ER` keyword must be specified along with the `BOTTOM_DIELECTRIC_THICKNESS` keyword within a CONDUCTOR block. If the specified conductor layer does not appear in a certain model, the bottom dielectric layer does not appear in the model either.

### Example

`BOTTOM_DIELECTRIC_ER = 4.0`

### See Also

- [BOTTOM\\_DIELECTRIC\\_THICKNESS](#)
- [CONDUCTOR](#)

---

## BOTTOM\_DIELECTRIC\_THICKNESS

Specifies the thickness of the dielectric region below a conductor. Valid within a CONDUCTOR block.

### Syntax

`BOTTOM_DIELECTRIC_THICKNESS = thickness`

### Arguments

| Argument                      | Description                                   |
|-------------------------------|-----------------------------------------------|
| <code><i>thickness</i></code> | Thickness of the dielectric<br>Units: microns |

### Description

The BOTTOM\_DIELECTRIC\_THICKNESS keyword must be specified along with the BOTTOM\_DIELECTRIC\_ER option within a CONDUCTOR statement. If the specified conductor layer does not appear in a certain model, the bottom dielectric layer does not appear in the model either.

If a conductor with a bottom dielectric also has conformal dielectrics on the sides of the conductor, the side conformal dielectric layers extend to the lowest surface of the bottom conformal dielectric, as shown in [Figure 16-1](#). When placing a conductor with bottom dielectric in the dielectric stack, the lowest surface of the bottom dielectric layer sits on the top surface of the planar dielectric layer defined below the conductor.

To specify the thickness of the conductor with the bottom dielectric layer, use the THICKNESS option in the CONDUCTOR statement. The top of a conductor that includes a bottom dielectric is placed above the planar dielectric at a height equal to the sum of the conductor thickness and the bottom dielectric thickness.

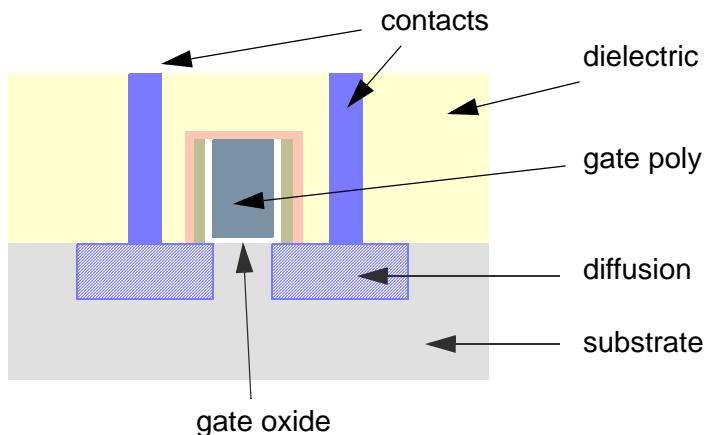
## Examples

### High-Permittivity Gate Oxide

The following statements model the high-permittivity dielectric under the gate shown in [Figure 16-1](#):

```
CONDUCTOR gpoly {
 THICKNESS = 0.06
 WMIN = 0.03
 SMIN = 0.03
 BOTTOM_DIELECTRIC_THICKNESS = 0.002
 BOTTOM_DIELECTRIC_ER = 10.0
}
```

*Figure 16-1 Modeling High-Permittivity Dielectric Under the Gate*

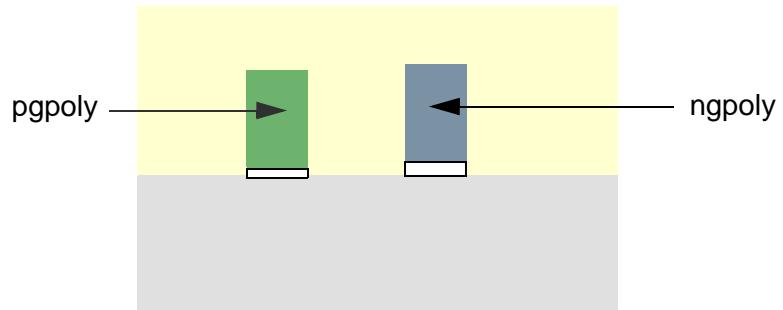


### Independent Bottom Dielectric Regions in Covertical Conducting Layers

You can define independent bottom dielectric regions in covertical conducting layers. For example, the following statements model pgpoly and ngpoly conductors with different bottom dielectric regions, as shown in [Figure 16-2](#):

```
CONDUCTOR pgpoly {
 THICKNESS = 0.06 WMIN = 0.03 SMIN = 0.03
 BOTTOM_DIELECTRIC_THICKNESS = 0.002
 BOTTOM_DIELECTRIC_ER = 10.0
}
CONDUCTOR ngpoly {
 THICKNESS = 0.06 WMIN = 0.03 SMIN = 0.03
 BOTTOM_DIELECTRIC_THICKNESS = 0.004
 BOTTOM_DIELECTRIC_ER = 12.0
}
```

Figure 16-2 Bottom Dielectric Layer With Covertical Layers



### See Also

- [BOTTOM\\_DIELECTRIC\\_ER](#)
- [CONDUCTOR](#)

---

## BOTTOM\_THICKNESS\_VS\_SI\_WIDTH

Specifies the bottom thickness of a conductor layer at different widths. Valid within a CONDUCTOR block.

### Syntax

```
BOTTOM_THICKNESS_VS_SI_WIDTH [RESISTIVE_ONLY | CAPACITIVE_ONLY]
{ (s1, r1) (s2, r2) ... (sn, rn) }
```

### Arguments

| Argument        | Description                                                                                                                                                                                                                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RESISTIVE_ONLY  | Applies thickness adjustment to resistance only                                                                                                                                                                                                                                                                                    |
| CAPACITIVE_ONLY | Applies thickness adjustment to capacitance only                                                                                                                                                                                                                                                                                   |
| s1 ... sn       | Silicon widths in ascending order. The first entry should be the smallest possible silicon width of the layer coming from the drawn WMIN value.                                                                                                                                                                                    |
| r1 ... rn       | Relative changes in bottom thickness. This is the absolute thickness change from the bottom divided by the nominal thickness of the layer. A bottom thickness or BTi is positive if the thickness becomes larger in the nominal value. A bottom thickness BTi is negative if the thickness becomes smaller than the nominal value. |

### Description

The BOTTOM\_THICKNESS\_VS\_SI\_WIDTH option models the bottom thickness of a conductor.

In a damascene process, deposited metal fills previously etched trenches. The variation in the trench etch depth affects the thickness of the interconnect as well as the vertical distance between metal interconnects. Both parasitic resistance and capacitance can be affected by these variations.

The grdgenxo tool does not check the validity of the silicon width values. The StarRC tool performs linear interpolation of the thickness for wires whose widths fall between entries in the table. The StarRC tool does not extrapolate beyond the table limits.

When the StarRC tool modifies a conductor thickness based on the BOTTOM\_THICKNESS\_VS\_SI\_WIDTH statement, the tool modifies the conductor sheet resistance values accordingly. The modifications apply to values specified in the RPSQ, RPSQ\_VS\_SI\_WIDTH, and RPSQ\_VS\_WIDTH\_AND\_SPACING statements.

The following limitations apply:

- The grdgenxo tool automatically processes trapezoidal conductor cross sections. This means that at a given thickness coming from a change at the bottom or top, the specified ETCH\_VS\_WIDTH\_AND\_SPACING, ETCH\_FROM\_TOP, or SIDE\_TANGENT value is automatically applied for the whole cross section when calculating the sensitivity.
- You can specify the BOTTOM\_THICKNESS\_VS\_SI\_WIDTH option along with the thickness variation from the top of the conductor by using the following options: THICKNESS\_VS\_DENSITY, THICKNESS\_VS\_WIDTH\_AND\_SPACING, POLYNOMIAL\_BASED\_THICKNESS\_VARIATION.
- The BOTTOM\_THICKNESS\_VS\_SI\_WIDTH option cannot be specified in the same CONDUCTOR statement as the MEASURED\_FROM option.
- The StarRC tool allows the BOTTOM\_THICKNESS\_VS\_SI\_WIDTH option to be used with multiple ETCH\_VS\_WIDTH\_AND\_SPACING tables. However, this combination might result in many sources of change for a metal line, making the behavior difficult to understand. This combination of ITF statements is not recommended.

### **Effective Thickness Calculation**

The effective thickness is calculated as follows:

$$T = T_{nom} \times (1 + RTf(Deff) + RTf(W, S) + RTf(SiW))$$

where

- $T_{nom}$  is the nominal thickness specified in the ITF file.
- $RTf(Deff)$  is the relative thickness change due to density.
- $RTf(W,S)$  is the relative change in thickness due to width and spacing.
- $RTf(SiW)$  is the relative change in thickness due to silicon width.

The resistance and capacitance are computed after the effective thickness is computed.

### **Errors**

An error occurs if the BOTTOM\_THICKNESS\_VS\_SI\_WIDTH option changes the relative thickness by more than 50 percent because the accuracy is compromised if the thickness changes greatly.

An error occurs if either the RESISTIVE\_ONLY or CAPACITIVE\_ONLY option is used with another BOTTOM\_THICKNESS\_VS\_SI\_WIDTH option without any qualifiers. The tool expects either a common BOTTOM\_THICKNESS\_VS\_SI\_WIDTH table or separate tables for the RESISTIVE\_ONLY and CAPACITIVE\_ONLY scenarios.

## See Also

- [CONDUCTOR](#)
- [RPSQ](#)
- [RPSQ\\_VS\\_SI\\_WIDTH](#)
- [RPSQ\\_VS\\_SI\\_WIDTH\\_AND\\_LENGTH](#)
- [RPSQ\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [SIDE\\_TANGENT](#)
- [THICKNESS\\_VS\\_DENSITY](#)
- [THICKNESS\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [POLYNOMIAL\\_BASED\\_THICKNESS\\_VARIATION](#)
- [MEASURED\\_FROM](#)

---

## BW\_T

Specifies the extension distance (bottom wall thickness) of the conformal dielectric below the conductor. Valid within a `DIELECTRIC` block.

### Syntax

`BW_T = thickness`

### Arguments

| Argument               | Description                                                                                         |
|------------------------|-----------------------------------------------------------------------------------------------------|
| <code>thickness</code> | The bottom conformal dielectric thickness<br>Units: microns<br>Default: thickness of the dielectric |

### Description

The `BW_T` option works with the `RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER` option, which specifies the dielectric constant of the associated silicon dielectric area between the raised diffusion and the gate processes. The `BW_T` value specifies the extension distance of the conformal dielectric below the conductor.

You can associate multiple conformal dielectric layers with `BW_T` values specified for the same conductor to produce multiple bottom conformal dielectrics. The `BW_T` thicknesses of multiple conformal dielectrics are additive. The `BW_T` dielectric listed first in the ITF file is the topmost `BW_T` dielectric associated with a particular conductor, that is, the closest bottom conformal dielectric to the conductor.

Use the following guidelines for the `BW_T` option:

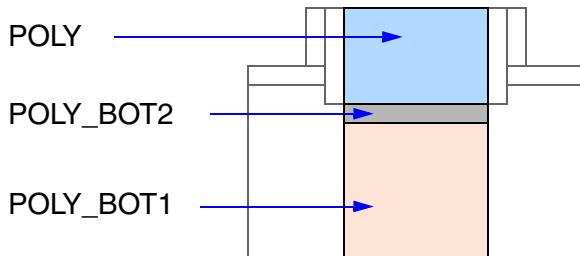
- The `BW_T` keyword can only be used in dielectric statements that also contain `IS_CONFORMAL` and `ASSOCIATED_CONDUCTOR` keywords.
- The `DAMAGE_THICKNESS` and `DAMAGE_ER` keywords cannot be simultaneously specified with the `BW_T` keyword.
- The `BOTTOM_DIELECTRIC_THICKNESS` and `BOTTOM_DIELECTRIC_ER` keywords cannot be specified in a conductor that is associated with a conformal layer that has a `BW_T` keyword.
- For conformal layers without a `BW_T` specification, set the `BW_T` value to zero to ensure backward compatibility with existing ITF files.
- If you define a `BW_T` value for a dielectric layer, you must set the `SW_T` and `TW_T` values to zero.

## Example

The following example uses the `BW_T` keyword to define the bottom dielectrics shown in Figure 16-3.

```
$ Gate oxide bottom conformal dielectric (closest to poly) DIELECTRIC POLY_BOT2 {
 THICKNESS=0.0 IS_CONFORMAL ASSOCIATED_CONDUCTOR=POLY SW_T=0 TW_T=0 BW_T=0.005 }
$ Silicon dielectric under gate oxide (farthest from poly) DIELECTRIC POLY_BOT1 {
 THICKNESS=0.0 IS_CONFORMAL ASSOCIATED_CONDUCTOR=POLY SW_T=0 TW_T=0 BW_T=0.3 ...}
```

Figure 16-3 Example of `BW_T` Option Usage



## See Also

- [DIELECTRIC](#)
- [MEASURED\\_FROM](#)
- [SW\\_T](#)
- [RAISED\\_DIFFUSION\\_GATE\\_SIDE\\_CONFORMAL\\_ER](#)
- [THICKNESS](#)

---

## CAPACITIVE\_ONLYETCH

Identical to the `ETCH` option, except that only capacitance is affected. Valid within a `CONDUCTOR` block.

### Syntax

`CAPACITIVE_ONLYETCH = etch_value`

### Arguments

| Argument                | Description                                                                                                                                        |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>etch_value</code> | Absolute width adjustment for one sidewall. A positive value shrinks the conductor; a negative value expands it.<br>Units: microns<br>Default: 0.0 |

### Description

The `CAPACITIVE_ONLYETCH` option applies an etch value to the sidewalls of a conductor. A positive value denotes conductor shrink; a negative value denotes conductor expansion. The adjusted conductor width is equal to the drawn width minus twice the etch value.

Use this option instead of the `ETCH` option to specify that an etch operation is to be used only for capacitance calculations.

If you use one of the `ETCH` options in addition to one or more `ETCH_VS_WIDTH_AND_SPACING` tables, the `ETCH_VS_WIDTH_AND_SPACING` operations are applied first, followed by the `ETCH` operation.

This option is not the same as `ETCH_VS_WIDTH_AND_SPACING CAPACITIVE_ONLY`.

### Example

```
CONDUCTOR metal1 {
 CAPACITIVE_ONLYETCH = 0.05
 THICKNESS=0.66 WMIN=0.15 SMIN=0.15 RPSQ=0.078
}
```

### See Also

- [ETCH](#)
- [RESISTIVE\\_ONLYETCH](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## CONDUCTOR

Describes the properties of a conductor layer.

### Syntax

```
CONDUCTOR conductor_name {
 [LAYER_TYPE = GATE | FIELD_POLY | DIFFUSION | TRENCH_CONTACT | BUMP
 | ROUTING_VIA]
 [ASSOCIATED_CONDUCTOR = associated_layer]
 [LINKED_TO = linked_layer]
 SMIN = min_spacing
 WMIN = min_width
 [EXTENSIONMIN = min_extension]
 [DPT_MAX_SHIFT = value]
 THICKNESS = cond_thk
 [DIELECTRIC_FILL_VS_SI_SPACING { ... }]
 [DIELECTRIC_FILL_EMULATION_VS_SI_SPACING { ... }]
 [BOTTOM_DIELECTRIC_THICKNESS = b_diel_thk
 BOTTOM_DIELECTRIC_ER = b_diel_er]
 [BOTTOM_THICKNESS_VS_SI_WIDTH ... { ... }]
 [T0 = nominal_temp]
 [CRT1 = lin_coeff
 | CRT1 = quad_coeff
 | CRT1 = lin_coeff CRT2 = quad_coeff
 | CRT_VS_SI_WIDTH { ... }]
 [DENSITY_BOX_WEIGHTING_FACTOR { ... }]
 [THICKNESS_VS_DENSITY ... { ... }]
 [DEVICE_TYPE { ... }]
 [DROP_FACTOR = value]
 [ETCH = value
 | CAPACITIVE_ONLYETCH = value
 | RESISTIVE_ONLYETCH = value]
 [ETCH_VS_WIDTH_AND_SPACING ... { ... }]
 [FILL_RATIO = fill_ratio_value
 FILL_WIDTH = fill_width_value
 FILL_SPACING = fill_spacing_value
 FILL_TYPE = GROUNDED | FLOATING]
 [GATE_TO_CONTACT_SMIN = value]
 [GATE_TO_DIFFUSION_CAP { ... }]
 [GATE_TO_DIFFUSION_ADJUSTMENT_CAP { ... }]
 [ILD_VS_WIDTH_AND_SPACING { ... }]
 [IS_PLANAR]
 [MEASURED_FROM = dielectric_name | TOP_OF_CHIP]
 [POLYNOMIAL_BASED_THICKNESS_VARIATION { ... }]
 [RAISED_DIFFUSIONETCH = rd_distance
 [RAISED_DIFFUSIONETCH_TABLE { ... }]
 RAISED_DIFFUSION_THICKNESS = rd_thickness
 RAISED_DIFFUSION_TO_GATE_SMIN = rd_spacing
 [RAISED_DIFFUSION_TO_GATE_SMIN_TABLE { ... }]
 [RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER = rd_er]]
```

```

[RPSQ = rpsq_value
| RHO = rho_value
| RPSQ_VS_SI_WIDTH { ... }
| RPSQ_VS_WIDTH_AND_SPACING { ... }
| RPSQ_VS_SI_WIDTH_AND_LENGTH { ... }
| RHO_VS_SI_WIDTH_AND_THICKNESS { ... }
| RHO_VS_WIDTH_AND_SPACING { ... }]
[SIDE_TANGENT = value
| SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING { ... }
| SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING { ... }]
[THICKNESS_VS_WIDTH_AND_SPACING ... { ... }]
[USER_DEFINED_DIFFUSION_RESISTANCE ... { ... }]
}

```

## Arguments

| Argument                                                          | Description                                                                                    |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| <i>conductor_name</i>                                             | Name of the conductor layer                                                                    |
| <i>dielectric_name</i>                                            | Name of a dielectric layer whose top surface is a reference height                             |
| ASSOCIATED_CONDUCTOR=<br><i>associated_layer</i>                  | For trench contact layers only, the layer used for trench contact extensions.                  |
| LINKED_TO= <i>linked_layer</i>                                    | Names of other conductor layers considered to be identical                                     |
| <i>SMIN</i> = <i>min_spacing</i>                                  | Minimum spacing between two geometries on this layer<br>Units: microns                         |
| <i>WMIN</i> = <i>min_width</i>                                    | Minimum width of a geometry on this layer<br>Units: microns                                    |
| <i>EXTENSIONMIN</i> = <i>min_width</i>                            | Minimum allowable extension of the field poly layer beyond the gate polygon.<br>Units: microns |
| <i>THICKNESS</i> = <i>cond_thk</i>                                | Thickness of the layer (minimum value is 0.001 micron)<br>Units: microns                       |
| <i>BOTTOM_DIELECTRIC_THICKN</i><br><i>ESS</i> = <i>b_diel_thk</i> | Thickness of the bottom dielectric layer<br>Units: microns                                     |
| <i>BOTTOM_DIELECTRIC_ER</i> =<br><i>b_diel_er</i>                 | Relative permittivity of the bottom dielectric layer                                           |

| <b>Argument</b>                                | <b>Description</b>                                                                                    |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <code>T0 = nominal_temp</code>                 | Nominal temperature<br>Units: degrees Celsius<br>Default: temperature specified by GLOBAL_TEMPERATURE |
| <code>CRT1 = lin_coeff</code>                  | Layer-specific linear temperature coefficient<br>Default: 0                                           |
| <code>CRT2 = quad_coeff</code>                 | Layer-specific quadratic temperature coefficient<br>Default: 0                                        |
| <code>FILL_RATIO = fill_ratio_value</code>     | Ratio of metal fill coverage                                                                          |
| <code>FILL_WIDTH = fill_width_value</code>     | Average width of metal fill objects<br>Units: microns                                                 |
| <code>FILL_SPACING = fill_spacing_value</code> | Average lateral spacing between signal nets and metal fill objects<br>Units: microns                  |
| <code>RPSQ = rpsq_value</code>                 | Resistance per square of the conducting layer<br>Units: ohms/square<br>Default: 0                     |
| <code>RHO = rho_value</code>                   | Bulk resistivity of the conductor layer<br>Units: ohms-micron                                         |

## Description

The CONDUCTOR statement describes the properties of a conductor layer such as minimum width, minimum spacing, thickness, resistivity, and process variation.

StarRC allows a maximum of 50 CONDUCTOR statements in the ITF file.

## Example

The following example shows a simple CONDUCTOR statement for a metal layer.

### Example 16-1 CONDUCTOR Statement for a Metal Layer

```
CONDUCTOR M1 { THICKNESS=0.8 WMIN=0.5 SMIN=0.45 RPSQ=0.041 }
```

---

## CRT\_VS\_AREA

Specifies the temperature coefficients of resistance as a function of via area. Valid within a `VIA` block.

### Syntax

```
CRT_VS_AREA {
 (area_1, crt1_1, crt2_2)
 (area_2, crt1_2, crt2_2)
 ...
 (area_n, crt1_n, crt2_n)
}
```

### Arguments

| Argument                       | Description                                                      |
|--------------------------------|------------------------------------------------------------------|
| <code>area_1 ... area_n</code> | Via areas specified in increasing order<br>Units: square microns |
| <code>crt1_1 ... crt1_n</code> | Linear temperature coefficients for corresponding via sizes      |
| <code>crt2_1 ... crt2_n</code> | Quadratic temperature coefficients for corresponding via sizes   |

### Description

Use the `CRT_VS_AREA` option within a `VIA` block to specify the temperature coefficients of resistance as function of via area. There is no limit to the number of entries. You cannot specify the `CRT_VS_AREA` option with either the `CRT1` or `CRT2` options in the same `VIA` block.

When the actual via size does not exactly equal any of the area entries in the `CRT_VS_AREA` table, `CRT1` and `CRT2` are determined by the following methods:

- If the actual via size is less than the smallest area entry in the `CRT_VS_AREA` table, the `CRT` values are set to the corresponding `CRT1` and `CRT2` entries of the smallest area entry; no extrapolation is performed.
- If the actual via size falls between two area entries in the `CRT_VS_AREA` table, `CRT1` and `CRT2` are calculated by linear interpolation.
- If the actual area is greater than the largest area entry in the `CRT_VS_AREA` table, the `CRT` values are set to the corresponding `CRT1` and `CRT2` entries of the largest area entry; no extrapolation is performed.

`CRT` and `AREA` values specified in a mapping file take precedence over the `CRT_VS_AREA` values specified in an ITF file.

## Errors

The grdgenxo tool issues a warning message if any of the following conditions is true:

- Area values in the table are not specified in increasing order. The tool internally reorders the table entries with increasing area values.
- The absolute value of CRT1 specified in the table is greater than 0.02.
- The value of CRT2 specified in the table is less than -0.002.

The grdgenxo tool issues an error message and stops the run if any of the following conditions is true:

- You specify both the `CRT_VS_AREA` option and the `CRT` option in the same `VIA` statement.
- You specify neither the nominal temperature for the via layer nor the global temperature.
- The `CRT_VS_AREA` table contains fewer than two rows. This same requirement applies to the `RPV_VS_AREA` table.
- The area values in the table are zero or negative.
- The `CRT_VS_AREA` option contains duplicate area values.
- You use the `-res_update` option with the `StarXtract` command while adding or removing a `CRT_VS_AREA` table.

## Example

```
CRT_VS_AREA {
 (0.002025, 9.04E-04, 4.74E-07)
 (0.005265, 1.18E-03, 8.02E-07)
}
```

## See Also

- [CRT1, CRT2, and T0](#)
- [VIA](#)

---

## CRT\_VS\_SI\_WIDTH

Specifies CRT-based temperature derating for different conductor widths. Valid within a CONDUCTOR block.

### Syntax

```
CRT_VS_SI_WIDTH {
 (siw_1, crt1_1, crt2_1)
 (siw_2, crt1_2, crt2_2)
 ...
 (siw_n, crt1_n, crt2_n)
}
```

### Arguments

| Argument                        | Description                                                               |
|---------------------------------|---------------------------------------------------------------------------|
| <i>siw_1</i> ... <i>siw_n</i>   | Conductor silicon (post-etch) widths<br>Units: microns                    |
| <i>crt1_1</i> ... <i>crt1_n</i> | Linear temperature coefficients for the corresponding conductor widths    |
| <i>crt2_1</i> ... <i>crt2_n</i> | Quadratic temperature coefficients for the corresponding conductor widths |

### Description

Use a CRT\_VS\_SI\_WIDTH table within a CONDUCTOR block to define CRT-based temperature derating for different conductor widths. There is no limit to the number of entries you can specify.

When the actual conductor width does not exactly equal any of the *siw* values in the CRT\_VS\_SI\_WIDTH table, then CRT1 and CRT2 are determined by the following methods:

- If the actual conductor width is less than the smallest *siw* value in the CRT\_VS\_SI\_WIDTH table, the CRT values are set to the corresponding *crt1* and *crt2* entries of the smallest *siw* entry; no extrapolation is performed.
- If the actual conductor width falls between two *siw* values in the CRT\_VS\_SI\_WIDTH table, CRT1 and CRT2 are calculated by linear interpolation.
- If the actual conductor width is greater than the largest *siw* value in the CRT\_VS\_SI\_WIDTH table, the CRT values are set to the corresponding *crt1* and *crt2* entries of the largest *siw* entry; no extrapolation is performed.

If both the `CRT_VS_SI_WIDTH` and `RPSQ_VS_SI_WIDTH` statements are specified for the same conductor, the width index should be the same for both statements.

### Example

```
CONDUCTOR MET1 {
 THICKNESS=0.6 WMIN=0.34 SMIN=0.40
 CRT_VS_SI_WIDTH {
 (0.34, 0.001, 0.000)
 (0.40, 0.001, 0.001)
 (0.823, 0.002, 0.001)
 (2.0, 0.003, 0.001)
 }
}
```

### See Also

- [CONDUCTOR](#)
- [CRT1, CRT2, and T0](#)
- [RPSQ\\_VS\\_SI\\_WIDTH](#)

---

## CRT1, CRT2, and T0

Defines parameters for temperature-dependent resistance models. Valid in CONDUCTOR, VIA, and TSV blocks.

### Syntax

```
CRT1 = lin_coeff
CRT2 = quad_coeff
T0 = nominal_temp
```

### Arguments

| Argument                 | Description                                                                                                                      |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| CRT1 = <i>lin_coeff</i>  | Linear temperature coefficient for the layer. Specified on a per-layer basis.<br>Default: 0                                      |
| CRT2 = <i>quad_coeff</i> | Quadratic temperature coefficient for the layer. Specified on a per-layer basis.<br>Default: 0                                   |
| T0 = <i>nominal_temp</i> | Nominal temperature for the layer<br>Units: degrees Celsius<br>Default: Temperature specified by the GLOBAL_TEMPERATURE keyword. |

### Description

The CRT1, CRT2, and T0 options define temperature-dependent resistance models for conducting layers and vias. The resistances are modeled similar to the way they are modeled in SPICE, by using the following equation:

$$R = R_0 \times [CRT1 \times (T - T_0) + CRT2 \times (T - T_0)^2 + 1]$$

In this equation,

- R is the modeled resistance at the operating temperature T.
- R0 is the resistance value at the nominal temperature T0.
- CRT1 and CRT2 are the linear and quadratic temperature coefficients.

The modeled resistance R exactly equals the nominal resistance (R0) if T=T0 or if CRT1 and CRT2 both equal 0.

If either CRT1 or CRT2 is nonzero for a layer, a nominal temperature specification is required for that layer. Set a global value for nominal temperature with the GLOBAL\_TEMPERATURE option at the beginning of the ITF file. If you set a nominal temperature both globally and for an individual layer, the layer nominal temperature overrides the global setting.

The OPERATING\_TEMPERATURE command must also be set in the StarRC command file to use the derating information in the nxtgrd file. If the resistance of a layer is changed by the mapping file, and if that layer has temperature derating in the ITF file, specifying the OPERATING\_TEMPERATURE command uses the temperature derating coefficients for that layer from the ITF file.

### Example

```
TECHNOLOGY = example_tech
GLOBAL_TEMPERATURE = 31.0
DIELECTRIC IMD2 { THICKNESS=2.0 ER=3.9 }
CONDUCTOR metal2 {
 CRT1=3.00e-3 CRT2=2.0e-7
 THICKNESS = 0.6 SMIN=0.5 WMIN=0.5 RPSQ = 0.06
}
DIELECTRIC IMD1 { THICKNESS=1.9 ER=4.9 }
CONDUCTOR metal1 {
 CRT1=3.50e-3 CRT2=2.5e-7
 THICKNESS = 0.5 SMIN = 0.4 WMIN=0.4 RPSQ = 0.08
}
DIELECTRIC FOX { THICKNESS=1.0 ER=3.9 }
VIA via1 {
 FROM=metal1 TO=metal2 AREA=1 RPV=1
 CRT1=2.5e-3 CRT2=1e-6 T0=29
}
```

### See Also

- [GLOBAL\\_TEMPERATURE](#)
- [OPERATING\\_TEMPERATURE](#)

---

## DAMAGE\_ER

Defines the equivalent permittivity of a damage layer. Valid within a [DIELECTRIC](#) block.

### Syntax

`DAMAGE_ER = relative_permittivity`

### Arguments

| Argument                           | Description           |
|------------------------------------|-----------------------|
| <code>relative_permittivity</code> | Relative permittivity |

### Description

Use the `DAMAGE_ER` and `DAMAGE_THICKNESS` options together to specify the relative permittivity of a damage layer. These options cannot be used for conformal dielectric layers.

### See Also

- [DAMAGE\\_THICKNESS](#)
- [DIELECTRIC](#)

---

## DAMAGE\_THICKNESS

Defines the thickness of a dielectric damage layer. Valid within a `DIELECTRIC` block.

### Syntax

`DAMAGE_THICKNESS = thickness`

### Arguments

| Argument               | Description                                                                      |
|------------------------|----------------------------------------------------------------------------------|
| <code>thickness</code> | Thickness of the damage layer on the surface of the dielectric<br>Units: microns |

### Description

Use the `DAMAGE_ER` and `DAMAGE_THICKNESS` options together to specify the equivalent permittivity of a damage layer. These options cannot be used for conformal dielectric layers.

### See Also

- [DAMAGE\\_ER](#)
- [DIELECTRIC](#)

---

## DENSITY\_BOX\_WEIGHTING\_FACTOR

Specifies density box weighting factors. Valid within a CONDUCTOR block.

### Syntax

```
DENSITY_BOX_WEIGHTING_FACTOR { (S1 W1) (S2 W2) (S3 W3) (S4 W4) (S5 W5) }
```

### Arguments

| Argument             | Description                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>S<sub>n</sub></i> | The size of the density box. Up to five entries are allowed.<br>S1, S2, S3, S4, and S5 are integers between 0 and 500, ordered from smallest to largest. The density box size is SxS.<br>Units: microns |
| <i>W<sub>n</sub></i> | The weighting factor specified as a floating-point number. Must be within the range of -10 < W < 10. If W is set to 0, that pair (S <sub>n</sub> W <sub>n</sub> ) is ignored.                           |

### Description

This option is to be used with the THICKNESS\_VS\_DENSITY option when specifying a single or multiple box method for effective density calculation. The weighting factor is multiplied by the density to arrive at the effective density.

If you do not specify a DENSITY\_BOX\_WEIGHTING\_FACTOR option, the default density box size of 50 µm is used with the weighting factor of unity. In other words, not specifying this option has the same effect as the following statement:

```
DENSITY_BOX_WEIGHTING_FACTOR { (50 1) }
```

### Example

```
CONDUCTOR metal13 {
 SMIN= 0.35 WMIN=0.42 THICKNESS=0.53 RPSQ=0.061
 THICKNESS_VS_DENSITY { (0.1 -0.1)(0.2 0.1)(0.3 0.15)(0.5 0.3) }
 DENSITY_BOX_WEIGHTING_FACTOR
 { (10 1)(20 0.23)(30 0.29)(40 0.08)(50 0.12) }
}
```

### See Also

- [THICKNESS\\_VS\\_DENSITY](#)
- [DENSITY\\_BASED\\_THICKNESS](#)

---

## DEVICE\_TYPE

Identifies conductor layers that compose a specified device. Valid within a CONDUCTOR block.

### Syntax

```
DEVICE_TYPE { [device_type_name_1] ... [device_type_name_2] | ALL | NONE }
```

### Arguments

| Argument                | Description                                              |
|-------------------------|----------------------------------------------------------|
| <i>device_type_name</i> | Applies the conductor layer to the specified device type |
| ALL                     | Applies the conductor layer to all device types          |
| NONE                    | Does not apply the conductor layer any device types      |

### Description

To improve the performance of the grdgexo tool when creating device models, the ITF DEVICE\_TYPE keyword identifies conductor layers that compose a specified device.

The *device\_type\_name* arguments are user-defined names. To include the conductor in all device types, specify the ALL keyword. Conversely, if the conductor does not exist in or near any device in the process, specify the NONE keyword; this is useful for a conductor corresponding to a capacitor or resistor that is never in or near a device.

The DEVICE\_TYPE option has the following constraints:

- The DEVICE\_TYPE designation can only be included in conductors with GATE, FIELD\_POLY, TRENCH\_CONTACT, or DIFFUSION layer types.
- Only one DEVICE\_TYPE option is allowed for each conductor.
- A device type name must adhere to the same syntax rules as a conductor name, via name, or dielectric name in the ITF file.
- Exactly one conductor with a LAYER\_TYPE=GATE statement and one conductor with a LAYER\_TYPE=DIFFUSION statement must be specified for each defined device type name.
- Conductors that have no specified device type but have a LAYER\_TYPE=GATE statement are applied to all device types, therefore no other conductors with a LAYER\_TYPE=GATE statement are allowed to exist in the process if a DEVICE\_TYPE statement appears anywhere in the ITF file. An equivalent rule is applied to conductors that have no device type but do have a LAYER\_TYPE=DIFFUSION statement.

- Any number of conductors that have a `LAYER_TYPE=FIELD_POLY` or `LAYER_TYPE=TRENCH_CONTACT` statement can be associated with a single device type name.
- `DEVICE_TYPE { NONE }` can only be applied to conductors with `LAYER_TYPE=FIELD_POLY` or `LAYER_TYPE=TRENCH_CONTACT` statements.

If any of these constraints are violated, the `grdgenxo` tool issues an error message.

### Example

The following example shows part of an ITF file that uses the `DEVICE_TYPE` option.

#### *Example 16-2 ITF File With DEVICE\_TYPE Option*

```
CONDUCTOR TC_RSD { DEVICE_TYPE { N_RSD P_RSD } ... }
CONDUCTOR TC { DEVICE_TYPE { N_NO_RSD P_NO_RSD } ... }
...
CONDUCTOR FPOLY_N { DEVICE_TYPE { N_RSD N_NO_RSD } ... }
CONDUCTOR FPOLY_P { DEVICE_TYPE { P_RSD P_NO_RSD } ... }
CONDUCTOR GPOLY_N { DEVICE_TYPE { N_RSD N_NO_RSD } ... }
CONDUCTOR GPOLY_P { DEVICE_TYPE { P_RSD P_NO_RSD } ... }
...
CONDUCTOR DIFF_NO_RSD { DEVICE_TYPE { N_NO_RSD P_NO_RSD } ... }
CONDUCTOR DIFF_N_RSD { DEVICE_TYPE { N_RSD } ... }
CONDUCTOR DIFF_P_RSD { DEVICE_TYPE { P_RSD } ... }
```

### See Also

- [CONDUCTOR](#)
- [LAYER\\_TYPE](#)

---

## DIELECTRIC

Describes the properties of a dielectric layer.

### Syntax

```
DIELECTRIC dielectric_name {
 er_value
 [ER_VS_SI_SPACING {
 (SI_SPACING1, ER1)
 (SI_SPACING2, ER2)
 ...
 (SI_SPACINGn, ERn) }]
 [ER_TABLE { ... }]
 THICKNESS = diel_thickness
 [MEASURED_FROM = meas_layer | TOP_OF_CHIP
 [SW_T = sw_thick] [TW_T = tw_thick]] [BW_T = bw_thick]
 [IS_CONFORMAL [ASSOCIATED_CONDUCTOR = conductor_name
 [SPACER_ER_VS_WIDTH_AND_SPACING {...}]]]
 [DAMAGE_THICKNESS = damg_thick DAMAGE_ER = damg_er]
 }
}
```

### Arguments

| Argument               | Description                              |
|------------------------|------------------------------------------|
| <i>dielectric_name</i> | Name of the dielectric layer             |
| <i>er_value</i>        | Relative permittivity of the layer       |
| <i>diel_thickness</i>  | Thickness of the layer<br>Units: microns |
| <i>meas_layer</i>      | Name of the reference dielectric layer   |
| <i>sw_thick</i>        | Sidewall thickness<br>Units: microns     |
| <i>tw_thick</i>        | Top wall thickness<br>Units: microns     |
| <i>bw_thick</i>        | Bottom wall thickness<br>Units: microns  |
| <i>conductor_name</i>  | Name of the associated conductor         |

| Argument          | Description                               |
|-------------------|-------------------------------------------|
| <i>damg_thick</i> | Damage layer thickness<br>Units: microns  |
| <i>damg_er</i>    | Relative permittivity of the damage layer |

## Description

The `DIELECTRIC` statement describes a dielectric layer above or below a conductor layer.

## Errors

The following error messages are issued when limitations for `THICKNESS`, `TW_T`, and `SW_T` are not observed:

```
ERROR: (908) ITF**
ERROR: Too thin SW_T value of 0.001 is specified for layer local1;
0 < SW_T < 0.005 is not allowed

ERROR: (910) ITF**
ERROR: Too thin TW_T value of 0.001 is specified for layer local1;
0 < TW_T < 0.005 is not allowed

ERROR: (906) ITF**
Too thin THICKNESS value of 0.0007 is specified for layer thin;
0<THICKNESS<0.001 is not allowed (THICKNESS=0 is allowed for conformal
dielectrics)
```

## Example

The following example describes a dielectric layer with a thickness of 0.3 µm and a relative permittivity of 3.9.

```
DIELECTRIC FOX { THICKNESS=0.3 ER=3.9 }
```

---

## DIELECTRIC\_FILL\_EMULATION\_VS\_SI\_SPACING

Specifies dielectric fill for use in TLUPlus files; does not affect StarRC extraction results.  
Valid within a CONDUCTOR block, for routing layers only.

### Syntax

```
DIELECTRIC_FILL_EMULATION_VS_SI_SPACING {
 SI_SPACINGS { s1 s2 ... sm }
 LENGTHS { L1 L2 ... Ln }
 VALUES { er(s1,L1) er(s2,L1) ... er(sm,L1)
 er(s1,L2) er(s2,L2) ... er(sm,L2)
 ...
 er(s1,Ln) er(s2,Ln) ... er(sm,Ln)
 }
}
```

### Arguments

| Argument                       | Description                                                         |
|--------------------------------|---------------------------------------------------------------------|
| <i>s1 s2 ... sm</i>            | Conductor spacings specified in ascending order<br>Units: microns   |
| <i>L1 L2 ... Ln</i>            | Conductor lengths specified in ascending order<br>Units: microns    |
| <i>er(s1,L1) ... er(sm,Ln)</i> | Relative permittivity for corresponding index values<br>Units: none |

### Description

TLUPlus models are used by the parasitic extractor in Synopsys place-and-route tools to model advanced process effects.

The DIELECTRIC\_FILL\_EMULATION\_VS\_SI\_SPACING option provides a method for estimating the parasitic capacitance of routing layers for processes that use dielectric fill (the insertion of low-permittivity dielectric shapes to reduce capacitance).

Use this option as follows:

1. Insert the DIELECTRIC\_FILL\_EMULATION\_VS\_SI\_SPACING option into the ITF file, for conductor routing layers only.
2. Use the grdgenxo tool to create a TLUPlus file from the ITF file.
3. Use the TLUPlus file in the IC Compiler or IC Compiler II tools.

If the `DIELECTRIC_FILL_EMULATION_VS_SI_SPACING` option is present in an ITF file or in an `nxtgrd` file, the StarRC tool ignores it during extraction.

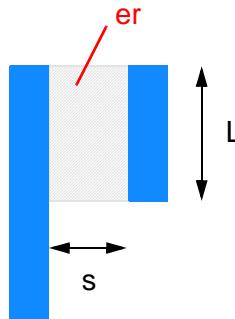
This option cannot be used with the following options in the same CONDUCTOR layer:

- `LATERAL_CAP_SCALING_VS_SI_SPACING`
- `ER_VS_SI_SPACING`
- `AIR_GAP_VS_SPACING`

The conductor spacing and length values are used as indexes for the two-dimensional table of relative permittivity values. Each combination of spacing  $s$  and length  $L$  has a corresponding relative permittivity value  $er(s, L)$ . [Figure 16-4](#) shows the conductor spacing and length values required for the `LENGTHS` and `SPACINGS` lists. If the conductors have different lengths, the length of the shortest conductor is used for the table lookup.

If the actual spacing or length value does not match any of the values supplied in the table, but is within the specified range, linear interpolation is used. If the actual spacing or length value falls outside of the specified range, the nearest value in the table is used.

*Figure 16-4 Application of Emulated Dielectric Fill*



### Example

In the following example, when the spacing is 0.1 microns and the length is 5 microns, the relative permittivity is 3.0. For spacing of 0.2 microns and length of 10 microns, the relative permittivity is 2.7. If the length is 30 microns, the values for a length of 20 microns are used.

```
DIELECTRIC_FILL_EMULATION_VS_SI_SPACING {
 SPACINGS { 0.1 0.2 }
 LENGTHS { 5.0 10.0 20.0 }
 VALUES { 3.0 2.5 2.0 2.8 2.7 2.5 }
}
```

### See Also

- [LATERAL\\_CAP\\_SCALING\\_VS\\_SI\\_SPACING](#)

---

## DIELECTRIC\_FILL\_VS\_SI\_SPACING

Defines parameters of dielectric fill features. Valid within a CONDUCTOR block.

### Syntax

```
DIELECTRIC_FILL_VS_SI_SPACING {
 SPACINGS { s1 s2 .. sn }
 WIDTHS { w(s1) w(s2) ... w(sn) }
 THICKNESSES { t(s1) t(s2) ... t(sn) }
 BOTTOM_HEIGHTS { h(s1) h(s2) ... h(sn) }
 SIDE_TANGENTS { a(s1) a(s2) ... a(sn) }
 ERS { e(s1) e(s2) ... e(sn) }
}
```

### Arguments

| Argument                     | Description                                                                                                                                                                                                                                        |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>s1 s2 ... sn</i>          | Spacing values between the two conductors, in ascending order<br>Units: microns                                                                                                                                                                    |
| <i>w(s1) w(s2) ... w(sn)</i> | Width of the dielectric fill region<br>Units: microns                                                                                                                                                                                              |
| <i>t(s1) t(s2) ... t(sn)</i> | Thickness of the dielectric fill region<br>Units: microns                                                                                                                                                                                          |
| <i>h(s1) h(s2) ... h(sn)</i> | Bottom height: offset of the bottom of the dielectric fill region with respect to the drawn bottom height of the conductor. A negative value indicates that the dielectric fill region is lower than the conductor bottom height<br>Units: microns |
| <i>a(s1) a(s2) ... a(sn)</i> | Side tangent of the dielectric fill region (representing the sidewall angle)<br>Units: none                                                                                                                                                        |
| <i>e(s1) e(s2) ... e(sn)</i> | Relative dielectric constant of the dielectric fill region<br>Units: none                                                                                                                                                                          |

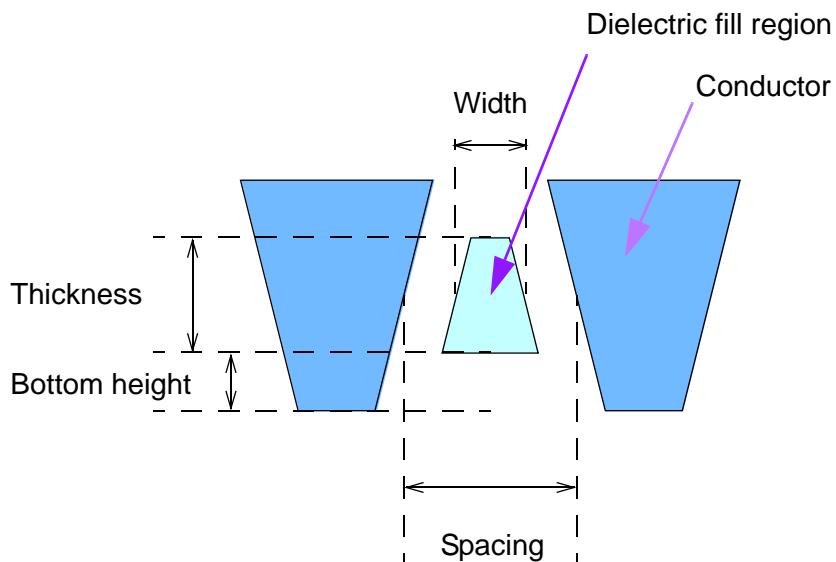
### Description

The DIELECTRIC\_FILL\_VS\_SI\_SPACING command defines the parameters of dielectric fill. Dielectric fill features typically have very low dielectric constants and are inserted to reduce

the coupling capacitance between upper-layer conductors. [Figure 16-5](#) shows the geometry of a dielectric fill region.

The side tangent value represents the angular shift from vertical of the side of the dielectric fill feature. The use of side tangent values is similar to the `SIDE_TANGENT` option used at the top level of the `CONDUCTOR` definition. A positive side tangent results in a top width that is larger than the center width, while a negative value results in a top width that is smaller than the center width. The feature in [Figure 16-5](#) results from a negative side tangent value.

*Figure 16-5 Dielectric Fill Geometry and Parameters*



Dielectric fill is inserted whenever the spacing between conductor features falls within the range specified in the `SPACINGS` list. The minimum allowed spacing value is the `SMIN` value of the conducting layer and the maximum value is 5 times the `SMIN` value.

The geometry of a specific dielectric fill feature is determined as follows:

- If the actual conductor spacing is smaller than the smallest value in the `SPACINGS` list, no dielectric fill is inserted.
- If the actual conductor spacing is larger than the largest value in the `SPACINGS` list, dielectric fill is inserted using the parameters for the largest spacing value in the `SPACINGS` list.
- If the actual conductor spacing falls between two spacing values in the `SPACINGS` list, linear interpolation is applied to calculate the width, thickness, and bottom height.

- If the actual conductor spacing falls between two spacing values in the SPACINGS list, the dielectric constant is not interpolated, but instead is set equal to the ERS value that corresponds to the smaller of the two spacing values.

For each conductor spacing value in the SPACINGS list, you must provide one value of each of the other parameters. In other words, the number of entries in each of the parameter lists must be equal. Lists of values are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

### Example

In the following example, dielectric fill is inserted between features in conductor layer M1 whenever the spacing between conductor features is greater than 0.035 microns.

```
CONDUCTOR M1 {
 THICKNESS=0.08
 CRT1=1E-03 CRT2=2E-07
 SIDE_TANGENT=0.09
 POLYNOMIAL_BASED_THICKNESS_VARIATION { ... }
 ETCH_VS_WIDTH_AND_SPACING { ... }
 DIELECTRIC_FILL_VS_SI_SPACING {
 SPACINGS { 0.035 .07 .24 }
 WIDTHS { 0.03 0.06 0.12 }
 THICKNESSES { 0.22 0.44 0.88 }
 BOTTOM_HEIGHTS { 0.008 0.016 0.032 }
 SIDE_TANGENTS { 0.09 0.095 0.15 }
 ERS { 2.6 2.7 2.8 }
 }
}
```

### See Also

- [CONDUCTOR](#)
- [SIDE\\_TANGENT](#)
- [SMIN](#)

---

## DPT\_MAX\_SHIFT

Specifies the maximum process misalignment shift for conductors using double or multiple patterning technology. Valid within a CONDUCTOR block.

### Syntax

```
DPT_MAX_SHIFT = shift_value
```

### Arguments

| Argument           | Description                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <i>shift_value</i> | The maximum misalignment shift in the X and Y direction for conductors using double or multiple patterning technology.<br>Units: microns |

### Description

StarRC provides two methods for modeling misalignment effects caused by double or multiple patterning technologies. In one method, you provide dielectric constant changes; in the other method, you provide the shift amount.

When you calculate the shift, use the DPT\_MAX\_SHIFT layer definition statement in the ITF file to specify the maximum layer shift.

### Example

```
CONDUCTOR m1 {
 WMIN=0.03 SMIN=0.03 DPT_MAX_SHIFT= 0.004 ...
}
```

### See Also

- [DPT](#)
- [DPT\\_COLOR\\_GDS\\_FILE](#)
- [DPT\\_COLOR\\_GDS\\_LAYER\\_MAP\\_FILE](#)
- [ER\\_VS\\_SI\\_SPACING](#)
- [Double or Multiple Patterning Technology](#)

## DROP\_FACTOR

Specifies the decrease in base height of all upper conductors when the bottom conductor is not present in the given layout area. Valid within a CONDUCTOR block.

### Syntax

`DROP_FACTOR = drop_value`

### Arguments

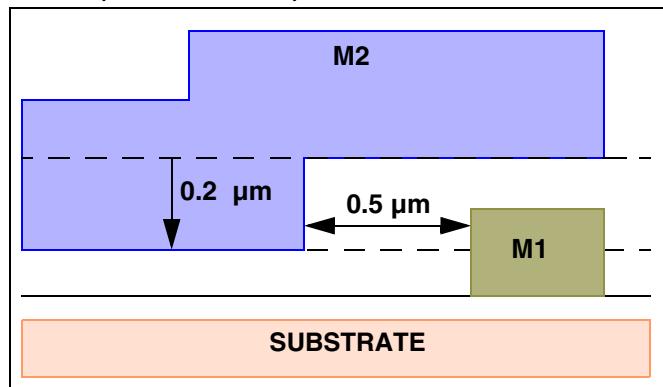
| Argument                | Description                                                                           |
|-------------------------|---------------------------------------------------------------------------------------|
| <code>drop_value</code> | Conductor base height decrease due to missing lower-level conductor<br>Units: microns |

### Description

The `DROP_FACTOR` option specifies the decrease in base height of upper-level conductors when a lower-level conductor is not present. Use the `DROP_FACTOR` option in the `CONDUCTOR` block for the lower-level conductor. You can specify a drop factor for no more than four conductors.

Figure 16-6 shows the effect of setting the drop factor to 0.2 microns for lower-level conductor M1. The base height of upper-level conductor M2 drops by 0.2 microns when M1 is not present. A lateral gap is maintained between the dropped part of the upper-level conductor and the lower-level conductor. This lateral gap is 0.5 um by default, but can be modified by using the `DROP_FACTOR_LATERAL_SPACING` statement.

Figure 16-6 Drop Factor Example



### See Also

- [DROP\\_FACTOR\\_LATERAL\\_SPACING](#)

---

## DROP\_FACTOR\_LATERAL\_SPACING

Specifies a constant lateral spacing value to use in conjunction with conductor drop factors.

### Syntax

```
DROP_FACTOR_LATERAL_SPACING = drop_lateral
```

### Arguments

| Argument            | Description                                                          |
|---------------------|----------------------------------------------------------------------|
| <i>drop_lateral</i> | Constant value between 0.5 and 4.0<br>Units: microns<br>Default: 0.5 |

### Description

The `DROP_FACTOR_LATERAL_SPACING` statement specifies a constant lateral spacing value used between a lower-level conductor that has a `DROP_FACTOR` statement and the dropped part of an upper-level conductor.

Specify the `DROP_FACTOR_LATERAL_SPACING` statement in the global parameters section after the `TECHNOLOGY` statement.

### Example

This example specifies a lateral spacing of 0.6  $\mu\text{m}$  for all conductors when dropped.

```
TECHNOLOGY = example_tech
DROP_FACTOR_LATERAL_SPACING = 0.6
```

### See Also

- [DROP\\_FACTOR](#)

---

## ER

Specifies the relative permittivity of a dielectric.

### Syntax

`ER = permittivity`

### Arguments

| Argument                         | Description                                 |
|----------------------------------|---------------------------------------------|
| <code><i>permittivity</i></code> | Relative permittivity of a dielectric layer |

### Description

The `ER` parameter specifies the relative permittivity, or dielectric constant, of a dielectric layer.

The `ER` parameter must be specified within the `DIELECTRIC` statement.

### Example

`DIELECTRIC D2 {THICKNESS = 1.2 ER = 3.9}`

### See Also

- [DIELECTRIC](#)

---

## ER\_TABLE

Specifies the device-dependent relative permittivity of a dielectric.

### Syntax

```
ER_TABLE {
 (device_type1 permittivity1)
 (device_type2 permittivity2)
 ...
}
```

### Arguments

| Argument            | Description                                 |
|---------------------|---------------------------------------------|
| <i>device_type</i>  | Device type                                 |
| <i>permittivity</i> | Relative permittivity of a dielectric layer |

### Description

The `ER_TABLE` option specifies the device-dependent relative permittivity, or dielectric constant, of a dielectric layer.

Specify the `ER_TABLE` option within the `DIELECTRIC` statement for the gate oxide under the polysilicon layer.

In the mapping file, you must provide the mapping information for the different device types.

### Example

The following example shows the use of the `ER_TABLE` option:

```
DIELECTRIC D3_BOT1 {
 THICKNESS=0.0 IS_CONFORMAL
 ER=7.55 ASSOCIATED_CONDUCTOR=POLY
 SW_T=0 TW_T=0 BW_T=0.001
 ER_TABLE { (G_1D5VIO_PMOS 7.0) (G_CORE_PMOS 8.0) }
}
```

### See Also

- [DIELECTRIC](#)
- [ER](#)

---

## ER\_VS\_SI\_SPACING

Specifies dielectric relative permittivity values as a function of conductor spacing width.  
Valid within a `DIELECTRIC` block.

### Syntax

```
ER_VS_SI_SPACING {
 (si_spacing_1, er_1)
 (si_spacing_2, er_2)
 ...
 (si_spacing_n, er_n)
}
```

### Arguments

| Argument                    | Description                                                |
|-----------------------------|------------------------------------------------------------|
| <i>si_spacing_1</i> ...     | Spacing values specified in ascending order                |
| <i>si_spacing_n</i>         | Units: microns                                             |
| <i>er_1</i> ... <i>er_n</i> | Relative permittivity for the corresponding spacing values |
|                             | Units: none                                                |

### Description

The `ER_VS_SI_SPACING` option models variable dielectric constants in the ITF file. One application is for modeling of capacitance variation due to double patterning processes. If two conductors move closer together or farther apart, the change in capacitance can be modeled by varying the dielectric constant.

To enable double patterning extraction, set the `DPT` command in the StarRC command file to `YES` and specify the `ER_VS_SI_SPACING` option within a `DIELECTRIC` block in the ITF file.

### Example

A dielectric layer might have the following relative permittivity definition:

```
ER = 5.3
ER_VS_SI_SPACING {
 (0.020, 7.2)
 (0.050, 6.4)
 (0.070, 5.3)
}
```

The resulting permittivity values are shown in [Table 16-1](#).

*Table 16-1 Relative Permittivity for Different Spacing Values*

| Spacing After Etch | Relative Permittivity             |
|--------------------|-----------------------------------|
| 0 to 0.02          | 7.2                               |
| 0.02 to 0.05       | Interpolated; between 7.2 and 6.4 |
| 0.05 to 0.07       | Interpolated; between 6.4 and 5.3 |
| greater than 0.07  | 5.3                               |

### See Also

- [DIELECTRIC](#)
- [DPT](#)
- [ER](#)

---

## ETCH

Specifies a width adjustment to model layer etch effects. Valid within a CONDUCTOR block.

### Syntax

ETCH = *etch\_value*

### Arguments

| Argument          | Description                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>etch_value</i> | Absolute width adjustment for one sidewall. A positive value shrinks the conductor; a negative value expands it.<br>Units: microns |

### Description

The ETCH option applies an etch value to each of the sidewalls of a conductor. A positive argument denotes conductor shrink; a negative value denotes conductor expansion. The adjusted conductor width is equal to the drawn width minus twice the etch value. The new conductor width is determined before resistance is calculated (for example by applying the RPSQ keyword).

You can specify that an etch operation is to be used for only capacitance calculations by using the CAPACITIVE\_ONLYETCH option instead of the ETCH option. Similarly, you can specify that an etch operation is to be used for only resistance calculations by using the RESISTIVE\_ONLYETCH option instead of the ETCH option.

If you use one of the ETCH options in addition to one or more ETCH\_VS\_WIDTH\_AND\_SPACING tables, the ETCH\_VS\_WIDTH\_AND\_SPACING operations are applied first, followed by the ETCH operation.

### Example

```
CONDUCTOR M1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05 ETCH=0.05
}
```

### See Also

- [CAPACITIVE\\_ONLYETCH](#)
- [RESISTIVE\\_ONLYETCH](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## ETCH\_VS\_CONTACT\_AND\_GATE\_SPACINGS

Specifies via etch as a function of spacing between vias and spacing between the gate and via. Valid within a `VIA` block.

### Syntax

```
ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {
 [NUMBER_OF_TABLES = num_of_tables]
 name_of_table1 {
 CONTACT_TO_CONTACT_SPACINGS { c1 c2 c3 ... }
 GATE_TO_CONTACT_SPACINGS { s1 s2 s3 ... }
 VALUES { v(c1,s1) v(c2,s1) ...
 v(c1,s2) v(c2,s2) ...
 ...
 }
 name_of_table2 {
 CONTACT_TO_CONTACT_SPACINGS { c1 c2 c3 ... }
 GATE_TO_CONTACT_SPACINGS { s1 s2 s3 ... }
 VALUES { v(c1,s1) v(c2,s1) ...
 v(c1,s2) v(c2,s2) ...
 ...
 }
 ...
 }
 }
}
```

### Arguments

| Argument                     | Description                                                                      |
|------------------------------|----------------------------------------------------------------------------------|
| CAPACITIVE_ONLY              | Applies etch effect to capacitance only                                          |
| <i>num_of_tables</i>         | The number of supplied tables                                                    |
| <i>nam_of_tableX</i>         | The name of the subsequent table; allowed only if multiple tables are specified  |
| <i>c1 c2 c3 ...</i>          | Contact to contact spacing values specified in ascending order<br>Units: microns |
| <i>s1 s2 s3 ...</i>          | Gate to contact spacing values specified in ascending order<br>Units: microns    |
| <i>v(c1,s1) v(c2,s2) ...</i> | Etch values of corresponding contact and gate spacing pairs<br>Units: microns    |

## Description

The actual size of contacts in silicon might be different than the sizes drawn in layout. To account for this difference during parasitic extraction, the `VIA` statement allows a contact bias (etch) to be specified as a function of contact-to-contact and gate-to-contact spacing. The etch affects only the estimated capacitance.

A positive etch value models a shrinking contact; a negative etch value models an expanding contact.

## Errors

Common error and warning conditions are as follows:

- From the `grdgenxo` tool
  - If the `NUMBER_OF_TABLES` keyword is not specified, an error occurs if there are multiple tables or if a table has a table name.
  - If the number of tables is different from the number specified with the `NUMBER_OF_TABLES` keyword, the `grdgenxo` tool stops and issues an error message.
  - If the old ITF format (no `NUMBER_OF_TABLES`, no table name) and the new format exist at the same time in the ITF file, with each format used for contact-etch and Cf tables, the `grdgenxo` tool stops and issues an error message.
- From the StarRC tool
  - If the `table_name` option in a mapping file does not match any table name in the ITF file, the StarRC tool issues a warning message.
  - If the `table_name` option is used in the mapping file for a layer other than a gate layer, the StarRC tool issues a warning message.
  - If contact-etch or gate-diffusion capacitance tables with a table name are present in the ITF file, but a gate-level layer in the mapping file does not have the `table_name` option, the StarRC tool issues a warning message.

## Examples

The following `VIA` definition does not have a contact bias:

```
VIA DiffCont {
 FROM=diff TO=metall1 AREA=0.003 RPV=15
 CRT1=6.56e-04 CRT2=-5.643e-0
}
```

The via definition can be extended to include a contact bias 2-D table, or etch table, as a function of contact-to-contact and gate-to-contact spacings. For example,

```
VIA DiffCont {
 FROM=diff TO=metall1 AREA=0.003 RPV=15
 CRT1=6.56e-04 CRT2=-5.643e-0
 ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {
 CONTACT_TO_CONTACT_SPACINGS {0.1 0.2 0.3}
 GATE_TO_CONTACT_SPACINGS {0.05 0.1 0.15}
 VALUES {0.005 0.006 0.007
 0.004 0.005 0.006
 0.003 0.004 0.005}
 }
}
```

You can specify multiple table definitions in the `ETCH_VS_CONTACT_AND_GATE_SPACINGS` option. Each table should have a name that associates with a gate database layer through a mapping file. You must use the same name for a contact etch table and a gate diffusion table for each kind of device. A keyword can, however, associate with only one contact-etch table or gate-diffusion capacitance table.

The following is an example of an ITF file with multiple contact etch tables defined:

```
VIA diffCont {
 FROM=diff TO=metall1 AREA=0.0036 RPV=55
 CRT1=9.56e-04 CRT2=-3.68e-07
 ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {
 NUMBER_OF_TABLES=2
 NMOS {
 CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
 GATE_TO_CONTACT_SPACINGS {0.04 0.08}
 VALUES {0.008 0.009 0.003 0.005}
 }
 PMOS {
 CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
 GATE_TO_CONTACT_SPACINGS {0.04}
 VALUES {0.004 0.002}
 }
 }
}
```

## Mapping File Syntax

Use the following syntax for the mapping file:

```
conducting_layers gate_database_layer gate_grd_layer [table_name=keyword]
```

If the `table_name` keyword is defined for a gate database layer, the StarRC tool finds and uses the contact etch table and gate-diffusion capacitance table of the same name in the ITF file.

If the `table_name` keyword is not specified for a gate database layer, no contact etch or gate-diffusion capacitance calculation is applied to the device with the gate of that data layer. For those devices, the contact etch is zero, and the gate-diffusion capacitance is the extracted value if the `IGNORE_CAPACITANCE` command is set to extract gate-diffusion coupling.

### Mapping File for Multiple Contact Etch Table Support

When multiple contact etch tables are defined in the ITF file, the device gate layer that maps to the corresponding table name must be specified in the StarRC mapping file. Use the following mapping file syntax to look up the appropriate gate-to-diffusion table:

```
conducting_layers
 gate_database_layer1 gate_grd_layer1 [table_name=name1]
 gate_database_layer2 gate_grd_layer2 [table_name=name2]
```

If `table_name` is defined for a gate, the StarRC tool finds and uses the corresponding gate-diffusion capacitance table with same name in the ITF file. If `table_name` is not defined for a gate database layer, no gate-diffusion capacitance calculation is done for the device with the gate of that database layer.

The following example shows a mapping file to look up the corresponding gate-to-diffusion capacitance tables based on the tables specified in the previous example:

```
conducting_layers
 ngate gpoly table_name=NMOS
 pgate gpoly table_name=PMOS
 tngate gpoly
 tpgate gpoly
```

With this mapping file definition, devices with `ngate` as the gate polysilicon use the NMOS contact etch table, and devices with `pgate` as the gate polysilicon use the PMOS table in the ITF file. No gate-to-contact etch is applied to the devices with `tngate` or `tpgate` gates.

### Backward Compatibility

If you use an `nxtgrd` file from a previous version of the StarRC tool, the tool behaves as it did in the previous version. The older format supports a single table for each contact-etch and Cf specification and does not have the `table_name` attribute in the ITF file.

StarRC treats the given table as a default table for all gate layers as it did in older versions. Even when the mapping file has `table_name` attributes for gate layers,

- The StarRC tool ignores the table name in the mapping file because there is no match in the `nxtgrd` file.
- The tool uses the unnamed table in the `nxtgrd` file for all cases.

**See Also**

- [CAPACITIVE\\_ONLYETCH](#)
- [GATE\\_TO\\_DIFFUSION\\_CAP](#)

---

## **ETCH\_VS\_WIDTH\_AND\_LENGTH**

Specifies different via etch values for the length and width sides of nonsquare vias.

### **Syntax**

```
ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {
 LENGTHS { l1 l2 ... lm }
 WIDTHS { w1 w2 ... wn }
 VALUES { (v_l1,v_w1) (v_l2,v_w1) ... (v_lm,v_w1)
 (v_l1,v_w2) (v_l2,v_w2) ... (v_lm,v_w2)
 ...
 (v_l1,v_wn) (v_l2,v_wn) ... (v_lm,v_wn)
 }
}
```

### **Arguments**

| Argument                                          | Description                                                           |
|---------------------------------------------------|-----------------------------------------------------------------------|
| CAPACITIVE_ONLY                                   | Applies etch effect to capacitance only                               |
| <i>l1 l2 ... lm</i>                               | Via lengths specified in ascending order<br>Units: microns            |
| <i>w1 w2 ... wn</i>                               | Via widths specified in ascending order<br>Units: microns             |
| ( <i>v_l1,v_wn</i> )<br>... ( <i>v_lm, v_wn</i> ) | Etch values of corresponding via lengths and widths<br>Units: microns |

### **Description**

Specify the `ETCH_VS_WIDTH_AND_LENGTH` option within a `VIA` statement to apply different via etch values to the length and width sides of nonsquare vias.

The specified length and width values are used as indexes for the two-dimensional table of etch values. Each combination of length *lm* and width *wn* has a corresponding pair of etch values (*v\_lm, v\_wn*), where *v\_lm* represents the etch value for the length, and *v\_wn* represents the etch value for the width.

The numbers in the `VALUES` field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Note the following restrictions when using an `ETCH_VS_WIDTH_AND_LENGTH` table:

- When the width is greater than the length, the corresponding etch values are n.
- Do not use the `ETCH_VS_WIDTH_AND_LENGTH` option to describe nonsquare diffusion contacts.
- Do not specify the `ETCH_VS_WIDTH_AND_LENGTH` option together with the `ETCH_VS_CONTACT_AND_GATE_SPACINGS` option in the same `VIA` statement.
- You can use `ETCH_VS_WIDTH_AND_LENGTH` together with the constant etch `CAPACITIVE_ONLYETCH` option.

## Errors

If the `ETCH_VS_WIDTH_AND_LENGTH` option is specified along with the constant etch `CAPACITIVE_ONLYETCH` option in a `VIA` definition, the two etch values are added together to apply the via etch.

The `grdgenxo` tool issues a warning message if

- A width and length combination results in an aspect ratio greater than one. The tool then forces their corresponding etch values to be zero.
- A width and length combination results in an aspect ratio equal to one, but their corresponding etch values are not the same.

The `grdgenxo` tool errors out if

- The `CAPACITIVE_ONLY` keyword is not specified.
- The `ETCH_VS_WIDTH_AND_LENGTH` option and the `ETCH_VS_CONTACT_AND_GATE_SPACINGS` option both exist in the same `VIA` definition.
- $VL \geq 0.5 \times LENGTH$  or  $VW \geq 0.5 \times WIDTH$ .

Note:

If you specify both `ETCH_VS_WIDTH_AND_LENGTH` and `CAPACITIVE_ONLYETCH` within the same `VIA` definition, then the combined value of the two etches is subject to the preceding restriction.

## Example

In the following example, the etch entry for length=0.045 and width=0.115 is set to (VL,VW)=(0.000, 0.000) because the parameter combination is invalid when the length is less than the width.

```
VIA vial{ FROM=M8 TO=M9 AREA=0.005 RPV=5
 ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {
 LENGTHS { 0.045 0.115 0.200 }
 WIDTHS { 0.045 0.115 }
 VALUES { (0.015, 0.015) (0.002, 0.002) (0.003 0.001)
 (0.000, 0.000) (0.015, 0.015) (0.005 0.002)
 }
 }
}
```

## See Also

- [ETCH\\_VS\\_CONTACT\\_AND\\_GATE\\_SPACINGS](#)
- [VIA](#)

---

## ETCH\_VS\_WIDTH\_AND\_SPACING

Specifies conductor etch values with respect to conductor width and spacing. Valid within a CONDUCTOR block.

### Syntax

```
ETCH_VS_WIDTH_AND_SPACING {
 [RESISTIVE_ONLY | CAPACITIVE_ONLY | ETCH_FROM_TOP]
 [PARALLEL_TO_REFERENCE | PERPENDICULAR_TO_REFERENCE | PARALLEL_TO_GATE]
 [NUMBER_OF_MASKS = num_masks]
 [MASKS(a,b) [(c,d)] {
 SPACINGS { s1 s2 ... sm }
 WIDTHS { w1 w2 ... wn }
 VALUES { v(s1,w1) v(s2,w1) ... v(sm,w1)
 v(s1,w2) v(s2,w2) ... v(sm,w2)
 ...
 v(s1,wn) v(s2,wn) ... v(sm,wn)
 }
 }]
 [MASKS(i,j) [(k,l)] {
 ...
 }]
}
```

### Arguments

| Argument                    | Description                                                                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RESISTIVE_ONLY              | Applies etch effect to resistance only                                                                                                                        |
| CAPACITIVE_ONLY             | Applies etch effect to capacitance only                                                                                                                       |
| ETCH_FROM_TOP               | Specifies a trapezoidal cross section (allows the etch to affect sidewall angle)                                                                              |
| PARALLEL_TO_REFERENCE       | Applies etch effect to wires that are parallel to the reference direction                                                                                     |
| PERPENDICULAR_TO_REFERENCE  | Applies etch effect to wires that are perpendicular to the reference direction                                                                                |
| PARALLEL_TO_GATE            | Applies etch on border edges that are parallel to the device gate. Use this option only for conductor layers where the LAYER_TYPE option is set to DIFFUSION. |
| NUMBER_OF_MASKS = num_masks | In a multiple mask patterning flow, the number of different mask colors                                                                                       |

| Argument                                                                                                  | Description                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MASKS ( <i>i, j</i> )                                                                                     | In a multiple mask patterning flow, the pairs of mask numbers to which the VALUES table applies                                                                                                                                                                                                                                   |
| <i>s<sub>1</sub>, s<sub>2</sub>, ... s<sub>m</sub></i>                                                    | Spacing values specified in ascending order; the number of spacings can be different for each set of mask pairs<br>Units: microns                                                                                                                                                                                                 |
| <i>w<sub>1</sub>, w<sub>2</sub>, ... w<sub>n</sub></i>                                                    | Width values; the number of width values can be different for each set of mask pairs<br>Units: microns                                                                                                                                                                                                                            |
| <i>v(s<sub>1</sub>,w<sub>1</sub>), v(s<sub>1</sub>,w<sub>2</sub>), ... v(s<sub>m</sub>,w<sub>n</sub>)</i> | Etch values. The notation <i>v(s<sub>1</sub>,w<sub>1</sub>)</i> denotes the etch value corresponding to spacing <i>s<sub>1</sub></i> and width <i>w<sub>1</sub></i> .<br>Units: microns<br>The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters. |

## Description

The `ETCH_VS_WIDTH_AND_SPACING` option helps you to consider fabricated pattern shapes in the extraction. Etch and lithography processing effects can cause the amount of etch to vary with the width of the conductor and its spacing to neighboring features. The etch effect might be different on two edges of the same conductor if the spacing to the nearest neighbor is different on the two sides. This option must be used within a `CONDUCTOR` section.

To model processes with multiple etch steps, use multiple `ETCH_VS_WIDTH_AND_SPACING` options. Specify them in the same order as the corresponding etch processes. Note that each occurrence of the `ETCH_VS_WIDTH_AND_SPACING` option might include more than one table if a multicolor mask flow is involved.

If you use the `ETCH` option in addition to one or more `ETCH_VS_WIDTH_AND_SPACING` tables, the `ETCH_VS_WIDTH_AND_SPACING` operations are applied first, followed by the `ETCH` operation.

Note:

If you use multiple `ETCH_VS_WIDTH_AND_SPACING` tables in addition to `RPSQ_VS_WIDTH_AND_SPACING` or `RHO_VS_WIDTH_AND_SPACING` tables, the StarRC tool issues a warning message. This combination results in many sources of variation for the same conductor and might lead to extraction inaccuracy.

For each `ETCH_VS_WIDTH_AND_SPACING` table, you must specify the `SPACINGS` and `WIDTHS` index vectors and the `VALUES` matrix; these three items can appear in any order. Positive

etch values cause structure widths to decrease, while negative etch values cause structure widths to increase.

You can apply the `ETCH_VS_WIDTH_AND_SPACING` option to both capacitance and resistance.

## Examples

### Single and Multiple Tables

The following example uses a single table:

```
CONDUCTOR metal2 {
 THICKNESS=0.65 WMIN=0.65 SMIN=0.50 RPSQ=0.62 ETCH=0.05
 ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY {
 SPACINGS { 0.5 0.67 0.8 }
 WIDTHS { 0.65 0.9 }
 VALUES { 0.1 0.05 -0.05
 0.15 0.10 -0.10 }
 }
}
```

The following example uses multiple tables:

```
CONDUCTOR M2 {
 THICKNESS=0.132 WMIN=0.050 SMIN=0.050 SIDE_TANGENT=0.06992
 ETCH_VS_WIDTH_AND_SPACING {
 *First etch_vs_width_and_spacing table (pre ADI table)
 SPACINGS { 0.050 0.100 }
 WIDTHS { 0.050 1.0 }
 VALUES { -0.0027 0.0034
 0.0003 0.0052 }
 }
 ETCH_VS_WIDTH_AND_SPACING {
 *Second etch_vs_width_and_spacing table (post ADI table)
 SPACINGS { 0.045 0.150 }
 WIDTHS { 0.045 2.0 }
 VALUES { -0.002 0.0014
 0.004 -0.003 }
 }
}
```

### Etch From Top

The `ETCH_FROM_TOP` keyword affects how the etch is applied to the structure sidewalls. Using this keyword results in a trapezoidal cross section. [Figure 16-7](#) shows the effects of using negative and positive etch values along with the `ETCH_FROM_TOP` keyword.

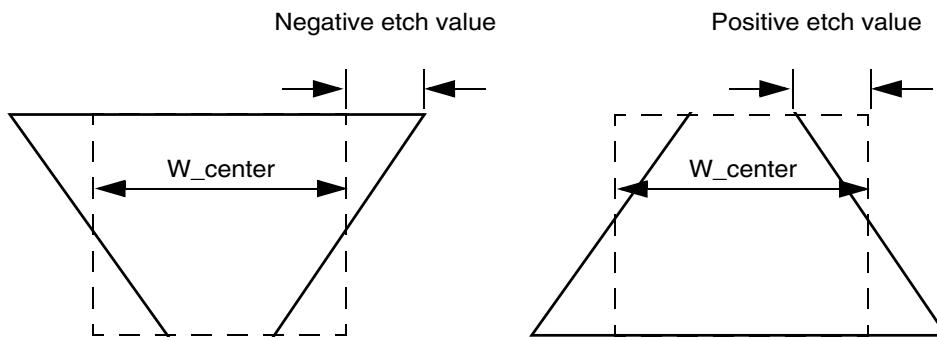
Because the amount of etch specified by the `ETCH_VS_WIDTH_AND_SPACING` option is based on the spacing to a neighboring structure, different sides of a conductor can be exposed to

different spacings and therefore can end up with different sidewall angles after the etch. However, the center width remains constant.

The `ETCH_FROM_TOP` keyword takes precedence over any `SIDE_TANGENT`, `SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING`, and `SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING` keywords used in a conductor definition.

The resistance of a conductor is not changed by application of the `ETCH_FROM_TOP` keyword, because the cross sectional area of the conductor does not change. Therefore, the `ETCH_FROM_TOP` keyword affects only capacitance by default. You cannot use the `RESISTIVE_ONLY` or `CAPACITIVE_ONLY` keywords with the `ETCH_FROM_TOP` keyword.

*Figure 16-7 Effect of the ETCH\_FROM\_TOP Keyword*



In the following example, a conductor of width 1 um spaced 0.5 um away from its neighbor on the left receives an etch of +0.05 um on the left edge. If that conductor is spaced 3 um away from another structure on the right side, the right edge receives an etch of -0.2 um.

```
CONDUCTOR Metal5 {
 THICKNESS=1.2 WMIN=0.3 SMIN=0.3 RPSQ = 0.62
 ETCH_VS_WIDTH_AND_SPACING ETCH_FROM_TOP {
 SPACINGS { 0.5 3 }
 WIDTHS { 1 2 }
 VALUES { 0.05 -0.2
 0.1 -0.17 }
 }
}
```

### Orientation-Dependent Width

Some etch processes affect lines differently depending on the orientation of the feature. To apply orientation-dependent width variation, you must specify the reference direction in either the ITF file or the StarRC command file. If the reference direction is specified in both files, the setting in the StarRC command file takes precedence.

The command formats are as follows:

- In the ITF file

```
REFERENCE_DIRECTION = VERTICAL | HORIZONTAL | GATE
```

- In the StarRC command file

```
REFERENCE_DIRECTION: VERTICAL | HORIZONTAL | GATE
```

One or more pairs of `ETCH_VS_WIDTH_AND_SPACING` tables can be included in each metal conductor layer. In each pair, one table must contain the `PARALLEL_TO_REFERENCE` keyword and specify the etch values for wires that are parallel to the reference direction; the other table must contain the `PERPENDICULAR_TO_REFERENCE` keyword and specify the etch values for wires that are perpendicular to the reference direction.

Note:

You cannot use the `ETCH_FROM_TOP` keyword with the `PARALLEL_TO_REFERENCE` or `PERPENDICULAR_TO_REFERENCE` keyword.

[Figure 16-8](#) shows an example of orientation-dependent width variation in which the reference direction is vertical. The etch orientation is either parallel or perpendicular to the reference direction.

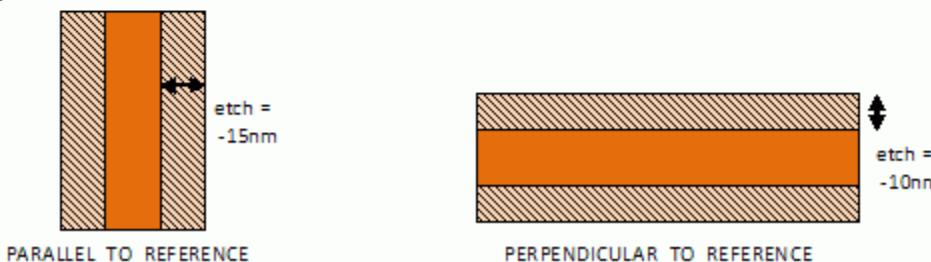
*Figure 16-8 Example of Orientation-Dependent Width Variation*

`REFERENCE_DIRECTION = VERTICAL`

#### Layer M2



#### Layer M3



The following example shows the corresponding settings in the ITF file.

```
REFERENCE_DIRECTION = VERTICAL
...
CONDUCTOR M2 {
 THICKNESS = ...
 ETCH_VS_WIDTH_AND_SPACING
 PARALLEL_TO_REFERENCE {
 WIDTHS {0.03} SPACINGS {0.03} VALUES {-0.015}
 }
 ETCH_VS_WIDTH_AND_SPACING
 PERPENDICULAR_TO_REFERENCE {
 WIDTHS {0.03} SPACINGS {0.03} VALUES {-0.005}
 }
}
CONDUCTOR M3 {
 THICKNESS = ...
 ETCH_VS_WIDTH_AND_SPACING
 PARALLEL_TO_REFERENCE {
 WIDTHS {0.03} SPACINGS {0.03} VALUES {-0.015}
 }
 ETCH_VS_WIDTH_AND_SPACING
 PERPENDICULAR_TO_REFERENCE {
 WIDTHS {0.03} SPACINGS {0.03} VALUES {-0.010} }
}
```

## Multiple Masks

A double patterning process splits features from a single layer into two separate masks. Printing the masks separately can achieve smaller features and spaces than a single mask. This practice affects extraction results if one of the masks shifts with respect to the other mask, because some features on the same layer move closer together and others move farther apart. If more than two masks are used for a single layer, the relationships are even further complicated.

You can use `ETCH_VS_WIDTH_AND_SPACING` tables to modify etch behavior based on mask shifts. The `NUMBER_OF_MASKS` keyword specifies the number of masks used to pattern the layer. The `MASKS` keyword names one or more mask pairs to which an etch table applies. A mask pair format is (self mask, neighbor mask); more than one pair can be specified for the same table. A mask ID of 0 indicates a colorless mask.

For a multiple mask process, all combinations of mask relationships must be specified. For a two-mask process, the table for mask pair (1,1) is used when adjacent features are both patterned by mask 1. Similarly, the table for mask pair (2,2) is used when adjacent features are both patterned by mask 2. The table for mask pair (1,2) is used for a feature on mask 1 when its neighbor is patterned by mask 2.

For two mask pairs with the same masks in different orders, such as mask pairs (1,2) and (2,1), the width and spacing indexes must be the same but the values can be different. If the values are also the same, you can specify one table for both pairs.

The following example uses a two-mask process. In this example, the same etch values are used for mask pair (1,2) and mask pair (2,1).

```
ETCH_VS_WIDTH_AND_SPACING
 NUMBER_OF_MASKS = 2
 MASKS (1,1) {
 SPACINGS { 0.5 0.67 0.8 }
 WIDTHS { 0.65 0.9 }
 VALUES { 0.1 0.05 -0.05
 0.15 0.10 -0.10 }
 }
 MASKS (1,2) (2,1) {
 SPACINGS { 0.05 0.10 }
 WIDTHS { 0.05 0.10 }
 VALUES { -0.002 0.0014
 0.004 -0.003 }
 }
 MASKS (2,2) {
 SPACINGS { 0.045 0.15 }
 WIDTHS { 0.045 2.0 }
 VALUES { -0.002 0.0014
 0.004 -0.003}
 }
```

The following example is a table for mask ID 0 that can be optionally added to the two-mask ETCH\_VS\_WIDTH\_AND\_SPACING block to define the behavior of noncolored geometries:

```
MASKS (0,0)(0,1)(0,2)(1,0)(2,0) {
 SPACINGS { 0.05 0.10 }
 WIDTHS { 0.005 1.0 }
 VALUES { -0.001 0.002
 0.004 0.003}
```

Specification of tables for mask 0 is optional. If you do not define the tables, the StarRC tool derives a pessimistic evaluation from the other mask tables, as follows:

- (0,0) is set to the average of (1,2) and (2,1)
- (0,1) is set to (2,1)
- (1,0) is set to (1,2)
- (0,2) is set to (1,2)
- (2,0) is set to (2,1)
- For mask pairs between mask 0 and a mask ID bigger than 2, the tables for mask 1 and the other mask are used.

If multiple ETCH\_VS\_WIDTH\_AND\_SPACING commands are used for the same layer, the number of masks must be the same for each ETCH\_VS\_WIDTH\_AND\_SPACING command of

the same type (RESISTIVE\_ONLY or CAPACITIVE\_ONLY). For example, you could have two CAPACITIVE\_ONLY tables, each specifying two masks, and one RESISTIVE\_ONLY table that does not specify any masks.

## See Also

- [CONDUCTOR](#)
- [REFERENCE\\_DIRECTION](#) (ITF command)
- [REFERENCE\\_DIRECTION](#) (StarRC command)

---

## EXTENSIONMIN

Specifies the minimum allowable extension of field poly beyond the gate polygon. Valid within a CONDUCTOR block.

### Syntax

EXTENSIONMIN = *ext\_value*

### Arguments

| Argument         | Description                               |
|------------------|-------------------------------------------|
| <i>ext_value</i> | Minimum extension value<br>Units: microns |

### Description

The EXTENSIONMIN option specifies the minimum allowable extension of field poly beyond the gate polygon.

Specify this option within a CONDUCTOR block. If the EXTENSIONMIN option is not specified, the WMIN value is used.

### Example

```
CONDUCTOR m1 {
 THICKNESS=1.00 WMIN=0.13 SMIN=0.15 RPSQ=0.015 EXTENSIONMIN = 0.01
}
```

### See Also

- [CONDUCTOR](#)
- [WMIN](#)

---

## FILL\_RATIO

Specifies the ratio of metal fill coverage for a specified conductor. Valid within a CONDUCTOR block.

### Syntax

`FILL_RATIO = ratio`

### Arguments

| Argument                  | Description                           |
|---------------------------|---------------------------------------|
| <code><i>ratio</i></code> | Ratio of metal fill for a given layer |

### Description

The FILL\_RATIO option specifies the ratio of metal fill coverage for a specified conductor. For example, if the density of the fill is 50 percent, specify `FILL_RATIO = 0.5`.

You must specify the FILL\_RATIO option when you specify the FILL\_SPACING and FILL\_WIDTH options.

### Example

```
CONDUCTOR m1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3
 FILL_RATIO = 0.4 FILL_SPACING = 1.0 FILL_WIDTH = 2.0
}
```

### See Also

- [FILL\\_SPACING](#)
- [FILL\\_TYPE](#)
- [FILL\\_WIDTH](#)
- [Emulated Metal Fill](#)

---

## FILL\_SPACING

Specifies the average lateral space separating signal nets and metal fill objects. Valid within a CONDUCTOR block.

### Syntax

`FILL_SPACING = spacing_value`

### Arguments

| Argument                          | Description                                   |
|-----------------------------------|-----------------------------------------------|
| <code><i>spacing_value</i></code> | The average lateral spacing<br>Units: microns |

### Description

The FILL\_SPACING option specifies the average lateral space separating signal nets and metal fill objects in microns. It is required if FILL\_RATIO is specified.

### Example

```
CONDUCTOR m1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3
 FILL_RATIO = 0.4 FILL_SPACING = 1.0 FILL_WIDTH = 2.0
}
```

### See Also

- [FILL\\_RATIO](#)
- [FILL\\_TYPE](#)
- [FILL\\_WIDTH](#)
- [Emulated Metal Fill](#)

---

## FILL\_TYPE

Specifies grounded or floating processing of lateral metal fill emulation. Valid within a CONDUCTOR block.

### Syntax

```
FILL_TYPE = GROUNDED | FLOATING
```

### Arguments

| Argument           | Description                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------------------|
| GROUNDED (default) | Treats lateral metal fill patterns defined by the FILL_WIDTH and FILL_SPACING commands as if they are tied to ground. |
| FLOATING           | Treats lateral metal fill patterns defined by the FILL_WIDTH and FILL_SPACING commands as if they are floating.       |

### Description

The FILL\_TYPE command selects floating or grounded processing of lateral metal fill emulation as defined by the FILL\_WIDTH and FILL\_SPACING commands.

### Example

```
CONDUCTOR m1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3 FILL_RATIO = 0.4
 FILL_SPACING = 1.0 FILL_WIDTH = 2.0 FILL_TYPE=FLOATING
}
```

### See Also

- [FILL\\_RATIO](#)
- [FILL\\_SPACING](#)
- [FILL\\_WIDTH](#)
- [Emulated Metal Fill](#)

---

## FILL\_WIDTH

Specifies the average size of metal fill objects. Valid within a CONDUCTOR block.

### Syntax

`FILL_WIDTH = value`

### Arguments

| Argument           | Description                                           |
|--------------------|-------------------------------------------------------|
| <code>value</code> | Average width of metal fill objects<br>Units: microns |

### Description

The `FILL_WIDTH` option specifies the average size of metal fill objects, in microns. `FILL_WIDTH` is required if `FILL_SPACING` or `FILL_RATIO` is specified.

### Example

```
CONDUCTOR m1 {
 THICKNESS=0.6 WMIN=0.3 SMIN=0.3
 FILL_RATIO = 0.4 FILL_SPACING = 1.0 FILL_WIDTH = 2.0
}
```

### See Also

- [FILL\\_RATIO](#)
- [FILL\\_SPACING](#)
- [FILL\\_TYPE](#)
- [Emulated Metal Fill](#)

---

## FROM

Specifies a conductor layer connected by a via. Valid within a [VIA](#) block.

### Syntax

FROM = *layer*

### Arguments

| Argument     | Description                          |
|--------------|--------------------------------------|
| <i>layer</i> | Conductor layer connected by the via |

### Description

The **FROM** option specifies the upper or lower layer (which must be a defined CONDUCTOR layer) connected by the via.

### Example

```
VIA sub_tie {
 FROM = SUBSTRATE
 TO = M1
 AREA = 0.25
 RPV = 5
}
```

### See Also

- [TO](#)
- [VIA](#)

## GATE\_TO\_CONTACT\_SMIN

Specifies the minimum spacing value between the polysilicon gate and the conductor-to-diffusion via layer. Valid within a CONDUCTOR block.

### Syntax

```
GATE_TO_CONTACT_SMIN = spacing
```

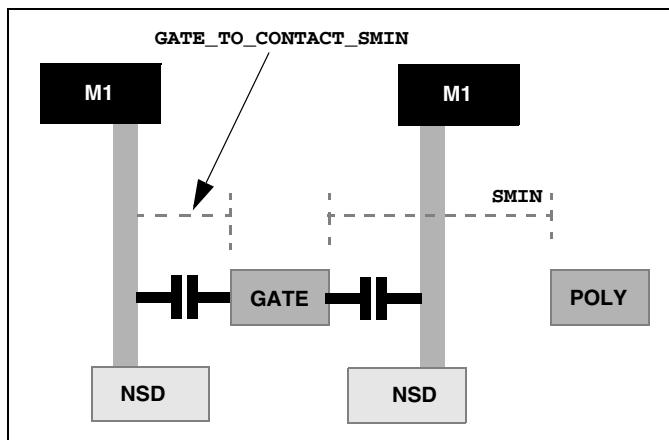
### Arguments

| Argument       | Description                                                                                                                                |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <i>spacing</i> | Minimum spacing between the polysilicon gate layer and the via layer between the diffusion and conductor layers<br>Default: the SMIN value |

### Description

The GATE\_TO\_CONTACT\_SMIN option specifies the minimum spacing between the polysilicon gate and the M1-to-diffusion via layer, as shown in [Figure 16-9](#). Use this option with the StarRC EXTRACT\_VIA\_CAPS command. For the polysilicon conductors, specify both the GATE\_TO\_CONTACT\_SMIN and SMIN values. Because gate-to-contact spacing is typically less than the SMIN value for polysilicon, specifying both values improves accuracy for EXTRACT\_VIA\_CAPS flows.

*Figure 16-9 SMIN and GATE\_TO\_CONTACT\_SMIN Definitions*



### Example

```
CONDUCTOR poly {
 THICKNESS=1.00 WMIN=0.13 SMIN=0.15
 RPSQ=0.015 GATE_TO_CONTACT_SMIN=0.08
}
```

---

## GATE\_TO\_DIFFUSION\_ADJUSTMENT\_CAP

Models dielectric constant variations around FinFET fins. Valid in a CONDUCTOR block for gate conductor layers only.

### Syntax

```
GATE_TO_DIFFUSION_ADJUSTMENT_CAP
 CHANNEL_LENGTHS {L1 L2 ... Lm}
 CHANNEL_WIDTHS {w1 w2 w3 ... wn}
 CAPS_PER_MICRON {
 v(L1,w1) v(L2,w1) ... v(Lm,w1)
 v(L1,w2) v(L2,w2) ... v(Lm,w2)
 ...
 v(L1,w3) v(L2,w3) ... v(Lm,wn)
 }
```

### Arguments

| Argument              | Description                                 |
|-----------------------|---------------------------------------------|
| CHANNEL_LENGTHS {...} | Channel lengths<br>Units: microns           |
| CHANNEL_WIDTHS {...}  | Channel widths<br>Units: microns            |
| CAPS_PER_MICRON {...} | Capacitance adjustments<br>Units: fF/micron |

### Description

Specify this option to model the effect of variable dielectric permittivity of the spacer conformal dielectric deposited on FinFET gate polysilicon layers. The dielectric properties might vary depending on the total number of fins in the device.

The GATE\_TO\_DIFFUSION\_ADJUSTMENT\_CAP option can only be used in conductor layers for which the LAYER\_TYPE option is set to GATE.

---

## GATE\_TO\_DIFFUSION\_CAP

Models the gate-to-diffusion capacitance within a CONDUCTOR statement.

### Syntax

```

GATE_TO_DIFFUSION_CAP {
 NUMBER_OF_TABLES = num_of_tables
 model_name1 {
 CONTACT_TO_CONTACT_SPACINGS {c1 c2 c3 ... cm}
 GATE_TO_CONTACT_SPACINGS {s1 s2 s3 ... sn}
 CAPS_PER_MICRON {
 v(c1,s1) v(c2,s1) ... v(cm,s1)
 v(c1,s2) v(c2,s2) ... v(cm,s2)
 ...
 v(c1,sn) v(c2,sn) ... v(cm,sn)
 }
 GATE_WIDTHS { g1 g2 g3 ... gx }
 GATE_TO_CONTACT_SPACINGS { s1 s2 s3 ... sn }
 CAPS_PER_MICRON {
 v(g1,s1) v(g2,s1) ... v(gx,s1)
 v(g1,s2) v(g2,s2) ... v(gx,s2)
 ...
 v(g1,sn) v(g2,sn) ... v(gx,sn)
 }
 }
 ...
 model_name2 {
 ...
 }
}

```

### Arguments

---

| Argument                    | Description                                             |
|-----------------------------|---------------------------------------------------------|
| <i>num_of_tables</i>        | Number of tables                                        |
| <i>model_name</i>           | Model name                                              |
| <i>c1 c2 c3 ...</i>         | Nearest contact-to-contact spacing<br>Units: microns    |
| <i>s1 s2 s3 ...</i>         | Gate-to-contact spacings<br>Units: microns              |
| <i>v(c1,s1) v(c2,s1)...</i> | Capacitance per micron<br>Units: femtofarads per micron |

| Argument                  | Description                   |
|---------------------------|-------------------------------|
| <code>g1 g2 g3 ...</code> | Gate widths<br>Units: microns |

### Description

The StarRC tool can extract the gate-to-diffusion capacitance component when the `IGNORE_CAPACITANCE: ALL` setting is specified. The gate-to-diffusion intradevice capacitance is of interest for parasitic extraction tools because of the strong layout dependency of this capacitance component. The gate-to-contact capacitance is extracted using the `EXTRACT_VIA_CAPS: YES` command in the StarRC command file.

To retain the gate-to-diffusion ( $C_f$ ) coupling component when `IGNORE_CAPACITANCE: ALL` is specified during a StarRC parasitic extraction, a command must be specified to include this capacitance in the netlist.

The capacitance table is included as part of the gate polysilicon definition in the ITF file. The three layout-dependent parameters used for the capacitance value in the table are:

- `CONTACT_TO_CONTACT_SPACINGS`
- `GATE_TO_CONTACT_SPACINGS`
- `GATE_WIDTHS`

Notes:

- A contact etch table and gate-diffusion cap table cannot be individually selected for a gate layer or for a device. They are always selected as a set. If you need another combination of two tables for a specific type of device, they can be defined in a table set with a new keyword in the ITF file and a new database layer for the corresponding gate in the mapping file.
- If the `GATE_TO_DIFFUSION_CAP` tables are not specified in the ITF file, the tool extracts the gate-to-diffusion capacitance based on its own grdgenxo models.
- In the case where contacts do not exist, as for shared source and drain regions, the largest spacing value is taken from the specified  $C_f$  table.

The numbers in the `CAPS_PER_MICRON` field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

## Examples

The following example shows a simple gate-to-diffusion capacitance table:

```
GATE_TO_DIFFUSION_CAP {
 CONTACT_TO_CONTACT_SPACINGS { c1 c2 c3 ... c_m }
 GATE_TO_CONTACT_SPACINGS { s1 s2 s3 ... s_n }
 CAPS_PER_MICRON {
 v(c1,s1) v(c2,s1) ... v(c_m,s1)
 v(c1,s2) v(c2,s2) ... v(c_m,s2)
 ...
 v(c1,s_n) v(c2,s_n) ... v(c_m,s_n)
 }
}
```

## Device-Dependent Gate-to-Diffusion Capacitance Table

You can specify a Cf table based on device type. [Example 16-3](#) shows multiple gate-to-diffusion capacitance tables defined in the ITF file, one for an n-type and another for a p-type device. Note that the number of tables and the table name must be specified when multiple gate-to-diffusion tables are specified.

A contact etch table and a gate-diffusion capacitance table for the same type of device should have the same table name.

### *Example 16-3 Device-Dependent Gate-to-Diffusion Capacitance Table*

```
CONDUCTOR gpoly {
 THICKNESS= 0.080000 WMIN= 0.040 SMIN= 0.100 RPSQ=12.000
 GATE_TO_CONTACT_SMIN=0.040 CRT1=1.924e-03 CRT2=-8.751e-07
 GATE_TO_DIFFUSION_CAP {
 NUMBER_OF_TABLES=2
 NMOS{
 CONTACT_TO_CONTACT_SPACINGS { 0.08 0.12}
 GATE_TO_CONTACT_SPACINGS { 0.04 0.08}
 CAPS_PER_MICRON { 0.062 0.088 0.080 0.096}
 }
 PMOS {
 CONTACT_TO_CONTACT_SPACINGS { 0.08 0.12}
 GATE_TO_CONTACT_SPACINGS { 0.04 0.08}
 CAPS_PER_MICRON { 0.088 0.1200.108 0.128}
 }
 }
}
```

## See Also

- [CAPACITIVE\\_ONLY\\_ETCH](#)
- [ETCH\\_VS\\_CONTACT\\_AND\\_GATE\\_SPACINGS](#)
- [IGNORE\\_CAPACITANCE](#): ALL RETAIN\_GATE\_DIFFUSION\_COUPLING

---

## GATE\_TO\_DIFFUSION\_CHANNEL\_CAP

Models the device-dependent gate-to-diffusion-channel capacitance within a CONDUCTOR statement.

### Syntax

```
CONDUCTOR gate_layer {
 LAYER_TYPE=GATE THICKNESS= ...
 GATE_TO_DIFFUSION_CHANNEL_CAP {
 NUMBER_OF_TABLES=num_of_tables
 name_of_table1 {
 CHANNEL_LENGTH { l1 l2 l3 ... }
 CHANNEL_WIDTH { w1 w2 w3 ... }
 CAPS_PER_MICRON { c(l1, w1) c(l2, w1) c(l3, w1) ...
 c(l1, w2) c(l2, w2) c(l3, w2) ...
 c(l1, w3) c(l2, w3) c(l3, w3) ... }
 }
 name_of_table2 {
 CHANNEL_LENGTH { l1 l2 l3 ... }
 CHANNEL_WIDTH { w1 w2 w3 ... }
 CAPS_PER_MICRON { c(l1, w1) c(l2, w1) c(l3, w1) ...
 c(l1, w2) c(l2, w2) c(l3, w2) ...
 c(l1, w3) c(l2, w3) c(l3, w3) ... }
 }
 ...
 }
}
```

### Arguments

| Argument             | Description                      |
|----------------------|----------------------------------|
| <i>gate_layer</i>    | Gate layer name                  |
| <i>num_of_tables</i> | Number of tables                 |
| <i>name_of_table</i> | Name of tables                   |
| <i>l1 l2 l3 ...</i>  | Channel length<br>Units: microns |
| <i>w1 w2 w3 ...</i>  | Channel width<br>Units: microns  |

| Argument                             | Description                                             |
|--------------------------------------|---------------------------------------------------------|
| <code>c(11, w1) c(12, w1) ...</code> | Capacitance per micron<br>Units: femtofarads per micron |

## Description

The `GATE_TO_DIFFUSION_CHANNEL_CAP` keyword specifies the device-dependent gate-to-diffusion-channel capacitance ( $C_{fi}$ ) table. While the `GATE_TO_DIFFUSION_CAP` parameter depends on the `GATE_TO_CONTACT_SPACINGS` parameter, the `GATE_TO_DIFFUSION_CHANNEL_CAP` parameter depends on the `CHANNEL_LENGTH` and `CHANNEL_WIDTH` parameters.

Specify the `GATE_TO_DIFFUSION_CHANNEL_CAP` keyword within a `CONDUCTOR` statement. In the mapping file, you must provide the mapping information for the different device types.

The numbers in the `CAPS_PER_MICRON` field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

## Example

```
GATE_TO_DIFFUSION_CHANNEL_CAP {
 NUMBER_OF_TABLES = 2
 G_1D5VIO_NMOS {
 CHANNEL_LENGTH {0.02 0.04 0.06}
 CHANNEL_WIDTH {0.02}
 CAPS_PER_MICRON{0.02 0.04 0.06}
 }
 G_CORE_NMOS {
 CHANNEL_LENGTH {0.02 0.04 0.06}
 CHANNEL_WIDTH {0.02}
 CAPS_PER_MICRON{0.02 0.04 0.06}
 }
}
```

## See Also

- [CONDUCTOR](#)
- [GATE\\_TO\\_DIFFUSION\\_CAP](#)
- [IGNORE\\_GATE\\_CHANNEL\\_CAPACITANCE](#)

---

## GLOBAL\_TEMPERATURE

Specifies the default nominal global temperature.

### Syntax

```
GLOBAL_TEMPERATURE = temp_value
```

### Arguments

| Argument          | Description                                                 |
|-------------------|-------------------------------------------------------------|
| <i>temp_value</i> | Global temperature<br>Units: degrees Celsius<br>Default: 25 |

### Description

The GLOBAL\_TEMPERATURE statement is optional. If the GLOBAL\_TEMPERATURE statement is not specified, then the nominal global temperature defaults to 25 degrees Celsius.

The nominal layer temperature overrides the global temperature when both are specified.

### Example

```
TECHNOLOGY = example_tech
GLOBAL_TEMPERATURE = 31.0
```

---

## **HALF\_NODE\_SCALE\_FACTOR**

Shrinks the design database before extraction begins.

### **Syntax**

`HALF_NODE_SCALE_FACTOR = scale_factor`

### **Arguments**

| <b>Argument</b>                  | <b>Description</b>                               |
|----------------------------------|--------------------------------------------------|
| <code><i>scale_factor</i></code> | A positive, nonzero scale factor<br>Default: 1.0 |

### **Description**

The `HALF_NODE_SCALE_FACTOR` statement can optionally be included in the global parameters section of the ITF file. The `HALF_NODE_SCALE_FACTOR` statement directs the extraction tool to shrink the design database by the specified value before extraction begins, which is useful if you are using a half-node process technology.

When the `HALF_NODE_SCALE_FACTOR` value is used, the StarRC tool sets the `MAGNIFY_DEVICE_PARAMS` command to `NO` and the `NETLIST_UNSCALED_COORDINATES` command to `YES` to ensure that the final netlist contains the full node coordinates.

- The `MAGNIFY_DEVICE_PARAMS: NO` command ensures that the standard device properties (`$w`, `$l`, `$area` and so on) in the netlist are the full-node values. For example, the physical properties of parasitic resistors generated by the `NETLIST_TAIL_COMMENTS: YES` command represent the full-node dimensions.
- The `NETLIST_UNSCALED_COORDINATES: YES` command ensures that all of the coordinates in the netlist are full-node values.

If you change the `HALF_NODE_SCALE_FACTOR` value, you must change the `WMIN` and `SMIN` values accordingly. The StarRC tool does not modify the `WMIN` or `SMIN` values automatically.

You can set the magnification factor in the StarRC command file after removing the value from the nxtgrd file by using the `grdgenxo` command, as shown in the examples. You cannot, however, delete the `HALF_NODE_SCALE_FACTOR` line from the nxtgrd file, because this causes the nxtgrd file to be corrupt.

## Errors

If the MAGNIFICATION\_FACTOR command appears in the StarRC command file and the HALF\_NODE\_SCALE\_FACTOR command appears in the nxtgrd file, the tool issues an error message.

If the MAGNIFY\_DEVICE\_PARAMS : YES or NETLIST\_UNSCALED\_COORDINATES : NO command is set in the StarRC command file for a run that uses a nxtgrd file containing the HALF\_NODE\_SCALE\_FACTOR option, the tool issues a warning message stating the new value for the options.

## Examples

The following is an example of an ITF header using the HALF\_NODE\_SCALE\_FACTOR option:

```
TECHNOLOGY = 65nm_example
GLOBAL_TEMPERATURE = 25
HALF_NODE_SCALE_FACTOR = 0.9

DIELECTRIC PASS2 {THICKNESS=0.800 ER=6.9}
DIELECTRIC PASS1 {THICKNESS=0.700 ER=4.0}
```

The HALF\_NODE\_SCALE\_FACTOR option interacts with the -add\_sf option of the grdgexo command as follows:

- If you generated the nxtgrd file without setting the HALF\_NODE\_SCALE\_FACTOR value, or you would like to change the value to 0.9, you can run the grdgexo tool and generate an updated nxtgrd file by using the following command:

```
% grdgexo -add_sf 0.9 -i noshrink.nxtgrd -o shrink.nxtgrd
```

- If you generated a StarRC nxtgrd file with a HALF\_NODE\_SCALE\_FACTOR value and you would like to run StarRC without scaling, you can reset the scale factor in the nxtgrd file by using the following command:

```
% grdgexo -add_sf 1 -i noshrink.nxtgrd -o shrink.nxtgrd
```

## See Also

- [SMIN](#)
- [WMIN](#)

---

## ILD\_VS\_WIDTH\_AND\_SPACING

Models the microloading effect or bottom conductor thickness variation. Valid within a CONDUCTOR block.

### Syntax

```
ILD_VS_WIDTH_AND_SPACING {
 DIELECTRIC = ILD_layer_name
 WIDTHS {w1 w2 w3 ...}
 SPACINGS {s1 s2 s3 ...}
 THICKNESS_CHANGES {
 v(s1,w1) v(s2,w1) ...
 v(s1,w2) v(s2,w2) ...
 ...
 }
}
```

### Arguments

| Argument                  | Description                                                                                                                                                                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ILD_layer_name</i>     | The name of the dielectric layer below the conductor corresponding to the thickness variation                                                                                                                                              |
| WIDTHS { ... }            | Width in drawn dimensions                                                                                                                                                                                                                  |
| SPACINGS { ... }          | Spacing in drawn dimensions                                                                                                                                                                                                                |
| THICKNESS_CHANGES { ... } | Absolute thickness variation of the ILD layer specified. A positive variation indicates an increase of thickness, and a decrease of thickness is represented by a negative variation. The maximum value is 0.2. The minimum value is -0.2. |

### Description

The `ILD_VS_WIDTH_AND_SPACING` option models the microloading effect or bottom conductor thickness variation.

`WIDTH` and `SPACING` are drawn dimensions. The `DIELECTRIC` keyword specifies the dielectric layer below the conductor that exhibits the thickness variation. Each entry in the `THICKNESS_CHANGES` field specifies the absolute thickness variation of the specified dielectric layer. A positive value indicates a thickness increase, while a negative value represents a thickness decrease. The `SPACING` option can be specified before the `WIDTH` option; however, the mapping of the values remains the same regardless of the order of specification of the two options.

Maximum values specified in the table must not be greater than 0.2 and must not be less than -0.2.

The `ILD_VS_WIDTH_AND_SPACING` option cannot be used in the same `CONDUCTOR` statement as the `BOTTOM_THICKNESS_VS_SI_WIDTH` option or the `MEASURED_FROM` option.

The numbers in the `THICKNESS_CHANGES` field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

### Handling ILD Variation in Flows

The StarRC tool does not combine multiple bottom thickness variation sources. You cannot specify both the `BOTTOM_THICKNESS_VS_SI_WIDTH` and `ILD_VS_WIDTH_AND_SPACING` statements for the same conductor.

### Restrictions

The StarRC tool detects the following error conditions:

- Variation is specified for a dielectric layer that does not exist directly below a conductor.
- ILD variation is specified for a conductor layer for which the `POLYNOMIAL_BASED_THICKNESS_VARIATION` command is not specified.
- The ILD variation specified is greater than 0.2 or less than -0.2.
- The ILD variation table is specified within the same `CONDUCTOR` statement as a `BOTTOM_THICKNESS_VS_SI_WIDTH` table.

### Example

```
ILD_VS_WIDTH_AND_SPACING {
 DIELECTRIC = ILD3
 WIDTHS {0.1 0.2 0.3 0.4}
 SPACINGS {0.11 0.22 0.33 0.44}
 THICKNESS_CHANGES {0.130 0.134 0.138 0.140
 0.135 0.138 0.139 0.142
 0.138 0.139 0.139 0.143
 0.140 0.142 0.144 0.146
 }
}
```

The `DIELECTRIC` statement specifies the dielectric layer below which the ILD variation is to be accounted for. The `VALUES` specified are absolute ILD variation values.

### See Also

- [BOTTOM\\_THICKNESS\\_VS\\_SI\\_WIDTH](#)
- [MEASURED\\_FROM](#)
- [POLYNOMIAL\\_BASED\\_THICKNESS\\_VARIATION](#)

---

## IS\_CONFORMAL

Defines the material the conductor layer is deposited around and allows conformal layers to be associated with a specified conductor layer. Valid within a **DIELECTRIC** block.

### Syntax

`IS_CONFORMAL`

### Arguments

The `IS_CONFORMAL` option does not have any arguments.

### Description

The `IS_CONFORMAL` option is specified with an `ASSOCIATED_CONDUCTOR` option. It defines the material the conductor layer is deposited around and allows conformal layers to be associated with a specified conductor layer.

- A **DIELECTRIC** layer specified with an `IS_CONFORMAL` layer tag is a conformal layer.
- An `IS_CONFORMAL` layer can have `SW_T` or `TW_T` around the `CONDUCTOR`. If `TW_T` or `SW_T` is not specified, the default is 0.0.
- When an `ASSOCIATED_CONDUCTOR` material drops with a `DROP_FACTOR` defined for a `CONDUCTOR` below it, `IS_CONFORMAL` layers also drop.
- If a `CONDUCTOR` above overlaps with the `TW_T` of an `IS_CONFORMAL` layer, the `CONDUCTOR` cuts into the `IS_CONFORMAL` layer.
- An `IS_CONFORMAL` layer can be specified without an `ASSOCIATED_CONDUCTOR`. See also `ASSOCIATED_CONDUCTOR`

### Errors

The following error and warning messages can be issued in the noted conditions:

- If thickness is defined for an `IS_CONFORMAL` layer, an error message is issued:  
`ERROR: LAYER xxx is set to be IS_CONFORMAL; it must have a thickness of`
- If `ASSOCIATED_CONDUCTOR` is defined for a layer that is not an `IS_CONFORMAL` layer:  
`ERROR: LAYER xxx must be set to IS_CONFORMAL if it has an ASSOCIATED_CONDUCTOR.`
- If the conductor defined for `ASSOCIATED_CONDUCTOR` is higher than the `IS_CONFORMAL` layer, the following message appears:  
`ERROR: ASSOCIATED_CONFORMAL layer xxx for layer xxx cannot be higher than the layer itself.`

**Example**

```
DIELECTRIC D1 {
 IS_CONFORMAL ASSOCIATED_CONDUCTOR=met1
 SW_T=0.1 TW_T=0.1 ER=2.5
}
```

**See Also**

- [ASSOCIATED\\_CONDUCTOR](#)
- [DROP\\_FACTOR](#)

---

## IS\_PLANAR

Specifies planar layers.

### Syntax

IS\_PLANAR

### Arguments

The IS\_PLANAR keyword does not have arguments.

### Description

Specifies that from this conductor and above, the layers do not drop because the DROP\_FACTOR option is specified for the lower layers.

### Example

```
CONDUCTOR ELEC1 {
 THICKNESS = 0.010
 WMIN = 0.180
 SMIN = 0.100
 RPSQ = 0.00001
 CAPACITIVE_ONLYETCH = 0
 IS_PLANAR
}
```

### See Also

- [DROP\\_FACTOR](#)

---

## LATERAL\_CAP\_SCALING\_VS\_SI\_SPACING

Specifies a capacitance scaling factor as a function of mask level and lateral spacing. Valid within a `DIELECTRIC` block.

### Syntax

```
LATERAL_CAP_SCALING_VS_SI_SPACING {
 [NUMBER_OF_MASKS = num_masks]
 [MASKS(a,b) [(c,d)] {
 (si_spacing_1, factor_1)
 (si_spacing_2, factor_2)
 ...
 (si_spacing_n, factor_n)
 }
 [MASKS(i,j) [(k,l)] {
 ...
 }]
}
```

### Arguments

---

| Argument                                                     | Description                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>NUMBER_OF_MASKS = <i>num_masks</i></code>              | In a multiple mask patterning flow, the number of different mask colors.                                                                                                                                                                                                         |
| <code>MASKS (<i>a,b</i>)</code>                              | In a multiple mask patterning flow, the pairs of mask numbers to which the scaling factors apply. Pair specifications ( <i>a,b</i> ) and ( <i>b,a</i> ) are equivalent.<br>Valid entries: 0 to <i>num_masks</i> , inclusive                                                      |
| <code><i>si_spacing_1</i> ...<br/><i>si_spacing_n</i></code> | Spacing values specified in ascending order; the number of spacing-permittivity pairs can be different for each set of mask pairs.<br>Units: microns<br>Valid entries: Positive numbers greater than 0 and less than 5 times the <code>SMIN</code> value of the conducting layer |
| <code><i>factor_1</i> ... <i>factor_n</i></code>             | Scaling factor that corresponds to the lateral spacing value in the same pair.<br>Units: none<br>Valid entries: Positive numbers from 0.6 to 1.4; the last entry must be no larger than 1.0                                                                                      |

---

## Description

The `LATERAL_CAP_SCALING_VS_SI_SPACING` statement specifies capacitance scaling factors as a function of lateral spacing for the purpose of modeling capacitance variation due to multiple patterning processes.

Use `LATERAL_CAP_SCALING_VS_SI_SPACING` statements in the ITF file according to the following conditions:

- No conductor mask coloring; single dielectric layer

Use one `LATERAL_CAP_SCALING_VS_SI_SPACING` statement without the `NUMBER_OF_MASKS` and `MASKS` keywords.

Alternatively, use the `ER_VS_SI_SPACING` statement, which does not take mask colors into account. You cannot specify both the `LATERAL_CAP_SCALING_VS_SI_SPACING` and `ER_VS_SI_SPACING` statements for the same conducting layer.

- Conductor with mask coloring and a single dielectric layer

Use one `LATERAL_CAP_SCALING_VS_SI_SPACING` statement with the `NUMBER_OF_MASKS` and `MASKS` keywords.

- Conductor with mask coloring and multiple dielectric layers

For multiple intrametal dielectric (IMD) layers that share the same conducting layer, a `LATERAL_CAP_SCALING_VS_SI_SPACING` table can be specified on some or all of the IMD layers.

All `LATERAL_CAP_SCALING_VS_SI_SPACING` tables for dielectrics associated with the same conductor must have the same spacing values for a specific mask pair.

All of the tables associated with a single conductor must use the same `NUMBER_OF_MASKS` value. In addition, if a mask-based `ETCH_VS_WIDTH_AND_SPACING` statement is used for the same conductor layer, its `NUMBER_OF_MASKS` value must be the same.

Spacing values are silicon (post-etch) spacing values. If the actual spacing falls within two values in the table, linear interpolation is used to calculate the scaling factor.

The `NUMBER_OF_MASKS` keyword specifies the number of masks used to pattern the layer. The `MASKS` keyword names one or more mask pairs to which an etch table applies. A mask pair format is (self mask, neighbor mask); more than one pair can be specified for the same table. A mask ID of 0 indicates a colorless mask.

When the `NUMBER_OF_MASKS` keyword is used, at least one colored mask combination must be specified with the `MASKS` keyword. For mask pairs that require no scaling, do not specify them; in this case, StarRC sets the factor to 1.

Specifying mask pairs that include mask 0 is optional. If they are missing, the tool maps mask pairs (0,1), (1,0), (0,2), and (2,0) to mask pair (1,2). For larger mask numbers, the tool

maps mask pairs (0,n) and (n,0) map to mask pair (1,n). If mask pair (1,n) is not specified, the tool uses a scaling factor of 1. You cannot specify the same mask pair more than one time.

The `DAMAGE_THICKNESS` option, if present, has higher priority.

## Errors

The `NUMBER_OF_MASKS` and `MASKS` keywords must be used if the design database contains colored mask information; the keywords cannot be used if the masks are uncolored. If a mismatch occurs, the StarRC tool issues an error message and exits the run.

The `LATERAL_CAP_SCALING_VS_SI_SPACING` statement is not valid for the following layers:

- Conformal dielectric layers
- Dielectric layers whose associated conductor layer contains a `DROP_FACTOR` specification

## Examples

The following example provides a single table for uncolored masks:

```
LATERAL_CAP_VS_SI_SPACING {
 (0.1,0.7) (0.12,0.8) (0.15,0.95) (0.2,1.0) }
```

The following example uses different values for different mask pairs. .

```
LATERAL_CAP_VS_SI_SPACING {
 NUMBER_OF_MASKS = 2
 MASKS (1,2) {
 (0.05 0.7) (0.08 0.8) (0.1 1.0)
 }
 MASKS (1,1) (2,2) {
 (0.05 1.25)
 (0.08 1.17)
 (0.1 1.0)
 }
}
```

## See Also

- [ER\\_VS\\_SI\\_SPACING](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## LAYER\_TYPE

Specifies the layer type within a CONDUCTOR block.

### Syntax

```
LAYER_TYPE = GATE | FIELD_POLY | DIFFUSION | TRENCH_CONTACT |
 | BUMP | CAPACITOR | ROUTING_VIA
```

### Arguments

| Argument       | Description                                                                                                                                                                                                        |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GATE           | A conducting layer that forms the gate of a device. If separate ITF conducting layers are not specified for gate and field polysilicon, specify the combined gate-field polysilicon layer as<br>LAYER_TYPE = GATE. |
| FIELD_POLY     | A field polysilicon layer outside of the device region                                                                                                                                                             |
| DIFFUSION      | A layer used for source or drain diffusion regions                                                                                                                                                                 |
| TRENCH_CONTACT | A conducting layer used for trench contacts. This includes both M1-to-diffusion trench contacts and M1-to-polysilicon trench contacts that can be used both inside and outside of the device region.               |
| BUMP           | A pseudo-metal via layer, used to connect microbumps to the top metal layer                                                                                                                                        |
| CAPACITOR      | A special-purpose layer for interleaved metal-finger capacitor structures                                                                                                                                          |
| ROUTING_VIA    | A layer that connects a routing conductor layer and a device conductor layer (such as a field poly or trench contact layer). Extraction conditions are determined by the layers being connected.                   |

### Description

For advanced process technologies, information in the ITF file about the function of the conducting layers improves capacitance extraction accuracy. You can optionally include a LAYER\_TYPE keyword within a CONDUCTOR section to guide the analysis. If a conducting layer does not have a specified layer type, it is considered to be a standard routing layer.

Note the following constraints on the relative vertical position of conductors in the interconnect stack:

- Conductors with `LAYER_TYPE = GATE` and `LAYER_TYPE = FIELD_POLY` must be covertical.
- Conductors with `LAYER_TYPE = TRENCH_CONTACT` must be covertical with or above conductors with `LAYER_TYPE = GATE` or `LAYER_TYPE = FIELD_POLY`.
- Conductors with `LAYER_TYPE = DIFFUSION` must be below the conductors with `LAYER_TYPE = GATE`.

## Errors

If the constraints on the layer type are not satisfied, the `grdgenxo` tool issues an error message.

## Example

The following example uses the `LAYER_TYPE` option to identify gate and diffusion layers:

```
CONDUCTOR PS {
 THICKNESS = 0.04
 WMIN = 0.04
 SMIN = 0.04
 GATE_TO_CONTACT_SMIN = 0.02
 LAYER_TYPE = GATE
 ...
}
DIELECTRIC DP1 {
 THICKNESS = 0.001
 ...
}
DIELECTRIC D_DIFF {
 THICKNESS = 0.04
 ...
}
CONDUCTOR DIFF {
 THICKNESS = 0.04
 WMIN=0.04
 SMIN=0.04
 LAYER_TYPE = DIFFUSION
 ...
}
```

## See Also

- [CONDUCTOR](#)
- [VIA](#)

---

## LINKED\_TO

Specifies the names of conductor layers to be considered identical.

### Syntax

`LINKED_TO = layer_name`

### Arguments

| Argument                       | Description           |
|--------------------------------|-----------------------|
| <code><i>layer_name</i></code> | Conductor layer names |

### Description

The `LINKED_TO` option specifies the names of other conductor layers to be considered identical with this layer. The `grdgenxo` tool automatically identifies conductors that have identical capacitive properties and attempts to reuse results previously generated with matching layers. If any `LINKED_TO` statements appear in the ITF file, the automatic linking is disabled and only explicitly linked conductors are considered to be identical.

Specify this option within a `CONDUCTOR` statement.

### Example

```
CONDUCTOR m1 {
 LINKED_TO = m3
}
```

### See Also

- [CONDUCTOR](#)

---

## MEASURED\_FROM

Modifies the reference location where thickness is measured.

### Syntax

```
MEASURED_FROM = dielectric_layer | TOP_OF_CHIP
```

### Arguments

| Argument                | Description                                                                                                                                                                                                       |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>dielectric_layer</i> | The name of the dielectric layer from which the measurement is made. This layer name must be defined in the ITF file.<br>The default is the dielectric layer immediately below.                                   |
| TOP_OF_CHIP             | Creates the bottom plane from the layers already present below the new layer and mimics the topology of the existing base, copying any existing nonplanarities to the new layer, which has a conformal thickness. |

### Description

Modifies the reference location where thickness is measured. The default is the dielectric layer immediately below it; be careful when using this feature for a conducting layer, as it is possible to create a conductor that cuts a dielectric, which might not be the wanted effect.

The MEASURED\_FROM option provides the ability to customize the model to account for such process characteristics as conformal dielectrics, mixed conformal and planar dielectrics, and covertical conductors. When used with a DIELECTRIC layer definition, the MEASURED\_FROM keyword can refer to a lower dielectric or can have the value TOP\_OF\_CHIP. When used with a CONDUCTOR layer definition, the MEASURED\_FROM keyword can refer only to a lower PLANAR dielectric.

The heights of the conductors and dielectrics are determined exclusively by the order in which they are specified and by the thicknesses of the lower layers. When you are specifying a new conductor or dielectric layer, the bottom plane of that layer is exactly the top plane of the dielectric layer immediately below it unless a MEASURED\_FROM option is included (to explicitly specify the location of the bottom plane).

Specify the MEASURED\_FROM option within a CONDUCTOR or DIELECTRIC statement.

## Examples

In the following example, TOP is planarized by measuring from D2:

```
DIELECTRIC TOP {
 THICKNESS = 3.6
 MEASURED_FROM = D2
 ER = 3.9
}
```

In the following example, D3 is a conformal dielectric:

```
DIELECTRIC D3 {
 THICKNESS=0.2
 MEASURED_FROM = TOP_OF_CHIP
 ER=3.9
}
```

## See Also

- [CONDUCTOR](#)
- [DIELECTRIC](#)
- [SW\\_T](#)
- [TW\\_T](#)

---

## MULTIGATE

Describes FinFET devices.

### Syntax

```
MULTIGATE fin {
 FIN_SPACING = fin_spacing
 FIN_WIDTH = fin_width
 FIN_LENGTH = fin_length (optional)
 FIN_THICKNESS = fin_thickness
 RAISED_DIFFUSION_GROWTH = raised_diffusion_growth (optional)
 GATE_POLY_EXTENSION = gate_poly_extension (optional)
 GATE_OXIDE_TOP_T = gate_oxide_top_thickness
 GATE_OXIDE_SIDE_T = gate_oxide_side_thickness
 GATE_OXIDE_ER = gate_oxide_permittivity
 GATE_POLY_TOP_T = gate_poly_top_thickness
 GATE_POLY_SIDE_T = gate_poly_side_thickness
 CHANNEL_ER = channel_permittivity
 GATE_DIFFUSION_LAYER_PAIR { (PGATE PDIFF) (NGATE NDIFF) ... }
}
```

### Arguments

| Argument                | Description                                                                                                              |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------|
| FIN_SPACING             | Fin spacing; the distance between adjacent fin sidewalls<br>Units: microns                                               |
| FIN_WIDTH               | Fin width<br>Units: microns                                                                                              |
| FIN_LENGTH              | Fin length (optional); the distance between the gate sidewall and the raised diffusion region<br>Units: microns          |
| FIN_THICKNESS           | Fin thickness<br>Units: microns                                                                                          |
| RAISED_DIFFUSION_GROWTH | Raised diffusion growth (optional); distance that raised diffusion extends laterally from fin sidewall<br>Units: microns |
| GATE_POLY_EXTENSION     | Gate poly extension (optional); distance that gate poly extends laterally outside the diffusion region<br>Units: microns |

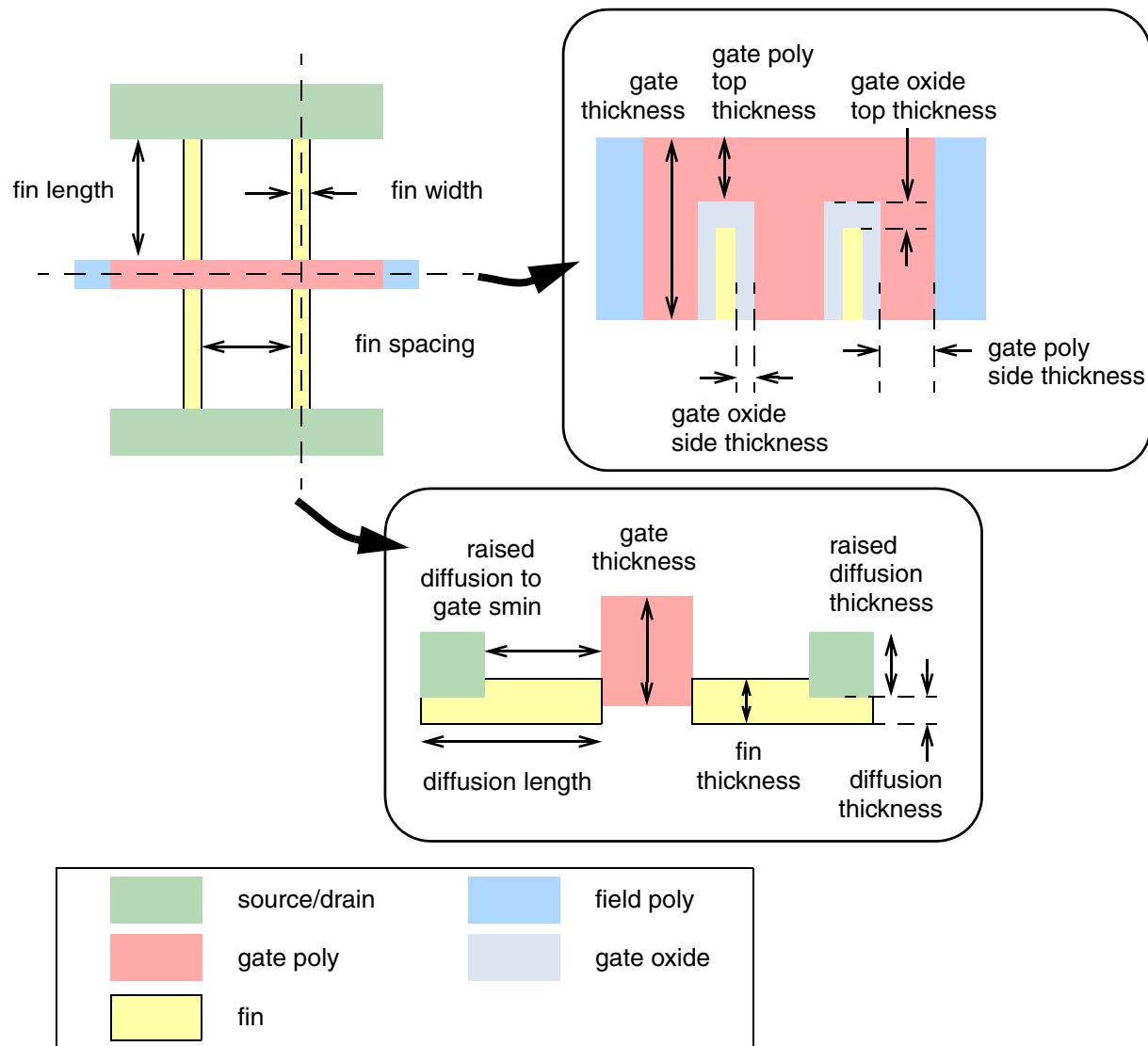
| Argument                  | Description                                                                       |
|---------------------------|-----------------------------------------------------------------------------------|
| GATE_OXIDE_TOP_T          | Gate oxide top thickness<br>Units: microns                                        |
| GATE_OXIDE_SIDE_T         | Gate oxide side thickness<br>Units: microns<br>Default: value of GATE_OXIDE_TOP_T |
| GATE_OXIDE_ER             | Gate oxide relative permittivity<br>Units: none (ratio)                           |
| GATE_POLY_TOP_T           | Gate poly top thickness<br>Units: microns                                         |
| GATE_POLY_SIDE_T          | Gate poly side thickness<br>Units: microns<br>Default: value of GATE_POLY_TOP_T   |
| CHANNEL_ER                | Channel relative permittivity<br>Units: none (ratio)                              |
| GATE_DIFFUSION_LAYER_PAIR | Identifies the gate diffusion layer pair                                          |

### Description

A block starting with the `MULTIGATE` keyword represents FinFET or other advanced devices. Inside the block, you specify parameters that describe a three-dimensional structure. The units for all length parameters are microns. [Figure 16-10](#) and [Figure 16-11](#) illustrate transistor parameters and cross sections.

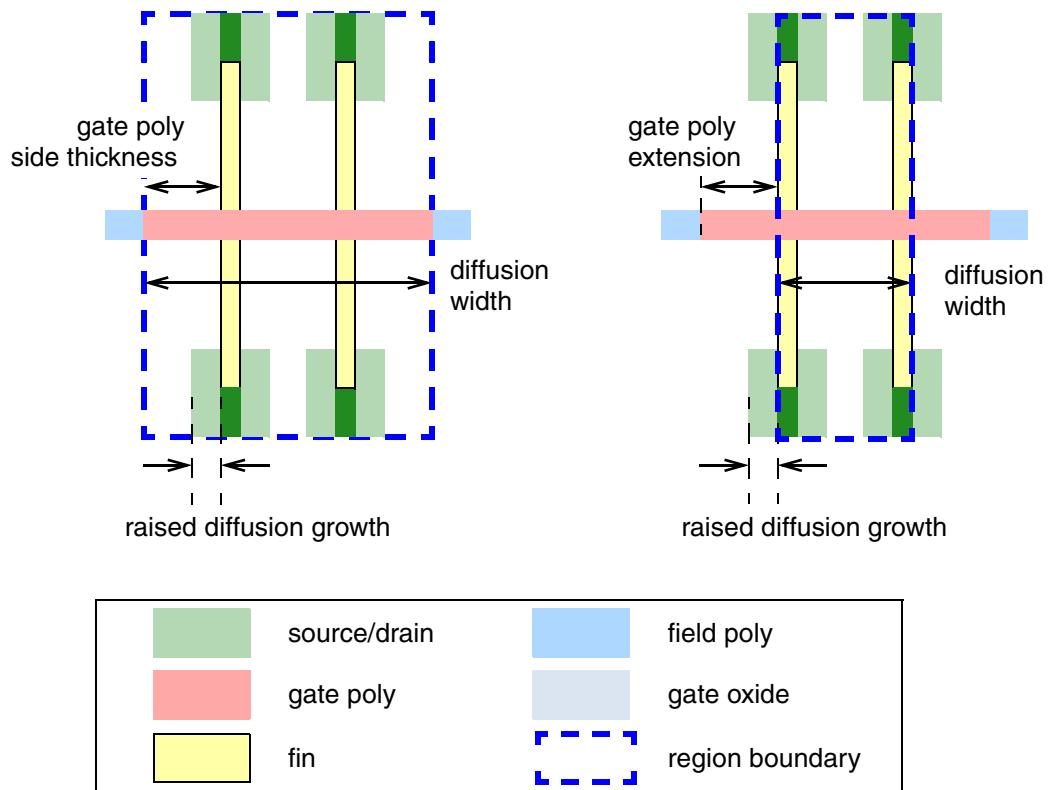
The `FIN_LENGTH` parameter is optional. If the parameter is not specified, the final fin length is the distance from the gate edge to the far edge of the diffusion region, labeled as diffusion length in [Figure 16-10](#).

*Figure 16-10 FinFET Top View and Cross Section Along Gate and Along Fin*



The RAISED\_DIFFUSION\_GROWTH parameter is optional; it represents the distance that a raised source/drain region extends laterally from the fin sidewall. If the value is larger than half the spacing between the fins, the raised diffusion regions merge, as shown by the large green boxes in [Figure 16-10](#). If the value is less than half the fin spacing, the raised diffusion regions remain separate, as shown by the smaller green boxes in [Figure 16-11](#).

*Figure 16-11 FinFET Top View With Different Diffusion Bounding Boxes*



If the RAISED\_DIFFUSION\_GROWTH parameter is not specified, a merged region is assumed. If the parameter is specified, the dimensions of the unmerged raised diffusion box are:

- Length (parallel to the fin length), calculated as:  
 $\text{DIFFUSION\_LENGTH} - \text{RAISED\_DIFFUSION\_TO\_GATE\_SMIN}$
  - Width (perpendicular to the fin length), calculated as:  
 $\text{FIN\_WIDTH} + 2 * \text{RAISED\_DIFFUSION\_GROWTH}$
  - Height or thickness, equal to the RAISED\_DIFFUSION\_THICKNESS parameter
- StarRC uses the drawn diffusion polygon boundary and the information in the MULTIGATE block to model finFET devices. Two styles of diffusion regions are supported, as shown by the dashed lines in [Figure 16-11](#).
- If the GATE\_POLY\_SIDE\_T parameter is specified and the GATE\_POLY\_EXTENSION parameter is undefined, the diffusion region boundary extends beyond the outermost fins by the distance specified in the GATE\_POLY\_SIDE\_T parameter.

- If the GATE\_POLY\_EXTENSION parameter is defined, the diffusion region boundary coincides with the outer sidewall of the outermost fins; the gate poly extends beyond the diffusion region boundary by that distance.

## Examples

The following two MULTIGATE blocks model NMOS and PMOS FinFET transistors. In the presence of a MULTIGATE statement, the DEVICE\_TYPE keyword must be used to identify the conductor layers that compose a specified device if there is more than one layer with LAYER\_TYPE=FIELD\_POLY in the ITF file.

### *Example 16-4 NMOS And PMOS FinFET Transistor Models*

```
MULTIGATE Nmos_fin {
 Fin_spacing=0.040
 Fin_width=0.02
 Fin_length=0.06
 Fin_thickness=0.05
 Raised_diffusion_growth=0.01
 Gate_oxide_top_t=0.003
 Gate_oxide_side_t=0.003
 Gate_oxide_er=8.0
 Gate_poly_top_t=0.03
 Gate_poly_side_t=0.04
 Channel_er=10.0
 Gate_diffusion_layer_pair {(Ngate Ndifff) }
}
MULTIGATE Pmos_fin {
 Fin_spacing=0.045
 Fin_width=0.025
 Fin_length=0.065
 Fin_thickness=0.05
 Raised_diffusion_growth=0.01
 Gate_oxide_top_t=0.0025
 Gate_oxide_side_t=0.0025
 Gate_oxide_er=8.0
 Gate_poly_top_t=0.035
 Gate_poly_side_t=0.045
 Channel_er=10.0
 Gate_diffusion_layer_pair {(Pgatet Pdiff) }
}
```

If two device types have the same parameters, you can define them with a single MULTIGATE block as follows:

```
MULTIGATE MOS_FIN {
 ...
 GATE_DIFFUSION_LAYER_PAIR { (NGATE NDIFF) (PGATE PDIFF) }
}
```

---

## POLYNOMIAL\_BASED\_THICKNESS\_VARIATION

Models conductor thickness variation as a function of feature density and width in a polynomial format. Valid within a CONDUCTOR block.

### Syntax

```

POLYNOMIAL_BASED_THICKNESS_VARIATION {
 [SI_]DENSITY_POLYNOMIAL_ORDERS = { do(n) do(n-1) ... do(0) }
 [SI_]WIDTH_POLYNOMIAL_ORDERS = { wo(m) wo(m-1) ... wo(0) }
 WIDTH_RANGES = { wt0 wt1 ... }
 POLYNOMIAL_COEFFICIENTS = {
 a(n,m) a(n,m-1) ... a(n,0)
 a(n-1,m) a(n-1,m-1) ... a(n-1,0)
 ...
 a(0,m) a(0,m-1) ... a(0,0)
 }
 POLYNOMIAL_COEFFICIENTS = {
 b(n,m) b(n,m-1) ... b(n,0)
 b(n-1,m) b(n-1,m-1) ... b(n-1,0)
 ...
 b(0,m) b(0,m-1) ... b(0,0)
 }
 POLYNOMIAL_COEFFICIENTS = {
 c(n,m) c(n,m-1) ... c(n,0)
 c(n-1,m) c(n-1,m-1) ... c(n-1,0)
 ...
 c(0,m) c(0,m-1) ... c(0,0)
 }
 ...
 [DENSITY_BOUNDS_VS_WIDTH {
 (wd1 dmin_wd1 dmax_wd1)
 (wd2 dmin_wd2 dmax_wd2)
 ...
 (wdn dmin_wdn dmax_wdn)
 }]
 [THICKNESS_BOUNDS {
 tmin tmax
 }]
}

```

**SYNTAX NOTES:** If the WIDTH\_RANGES keyword is not used, there must be exactly one POLYNOMIAL\_COEFFICIENTS table. If the WIDTH\_RANGES keyword is used, the number of POLYNOMIAL\_COEFFICIENTS tables must be one more than the number of widths in the WIDTH\_RANGES argument list.

## Arguments

| Argument                | Description                                                                                                         |
|-------------------------|---------------------------------------------------------------------------------------------------------------------|
| $do(n) \dots do(0)$     | Density exponents<br>Format: integer                                                                                |
| $wo(n) \dots wo(0)$     | Width exponents<br>Format: integer                                                                                  |
| $wt0, wt1, \dots$       | Conductor widths that determine which coefficient table to use;<br>must appear in ascending order<br>Units: microns |
| $a(n,m), b(n,m), \dots$ | Polynomial coefficients                                                                                             |
| $wd1, wd2, \dots$       | Conductor widths that define density bounds; must appear in<br>ascending order<br>Units: microns                    |
| $dmin_wd1, dmax_wd1$    | Minimum and maximum density for the corresponding width                                                             |
| $tmin$                  | Minimum absolute thickness; must be smaller than conductor<br>nominal thickness<br>Units: microns                   |
| $tmax$                  | Maximum absolute thickness; must be larger than conductor<br>nominal thickness<br>Units: microns                    |

## Description

The `POLYNOMIAL_BASED_THICKNESS_VARIATION` command uses a polynomial to model conductor thickness variation as a function of density and width. You can use either the as-drawn or the post-etch values of density and width in the calculations.

The command contains one or more tables of polynomial coefficients. A specific table applies to a range of conductor widths defined by the `WIDTH_RANGES` keyword. The coefficients table is selected as follows:

- If the `WIDTH_RANGES` keyword is not used, one coefficients table is used for all conductor widths. Only one table should be provided.
- If the `WIDTH_RANGES` keyword has an argument list with one width value, there must be exactly two `POLYNOMIAL_COEFFICIENTS` tables. The first coefficients table applies to

conductor widths less than or equal to the listed width, while the second table applies to conductor widths larger than the listed width.

- If the `WIDTH_RANGES` keyword has an argument list with more than one width value, the widths define a series of ranges. The first coefficients table applies to conductor widths less than or equal to the first width in the list. The second table applies to widths greater than the first width in the list and less than or equal to the second width, and so on. The last table applies to conductor widths larger than the last width in the list.

Thickness variation is calculated as follows, where  $W$  is the conductor width and  $D$  is the density:

$$\frac{dT}{T} = \left[ D^{do(n)} D^{do(n-1)} \dots D^{do(1)} D^{do(0)} \right] \begin{bmatrix} a(n, m) & a(n, m-1) & \dots & a(n, 1) & a(n, 0) \\ a(n-1, m) & \dots & \dots & \dots & a(n-1, 0) \\ \dots & \dots & \dots & \dots & \dots \\ a(1, m) & \dots & \dots & \dots & a(1, 0) \\ a(0, m) & a(0, m-1) & \dots & a(0, 1) & a(0, 0) \end{bmatrix} \begin{bmatrix} W^{wo(m)} \\ W^{wo(m-1)} \\ \dots \\ W^{wo(1)} \\ W^{wo(0)} \end{bmatrix}$$

The following example illustrates the calculation by using symbols to represent the coefficients. In this example, one coefficients table is used for all conductor widths because no `WIDTH_RANGES` keyword is included:

```
POLYNOMIAL_BASED_THICKNESS_VARIATION {
 DENSITY_POLYNOMIAL_ORDERS = { 2 1 0 }
 WIDTH_POLYNOMIAL_ORDERS = { 4 2 0 }
 POLYNOMIAL_COEFFICIENTS = {
 a b c
 d e f
 g h i
 }
}
```

The resulting thickness variation equation is as follows:

$$\frac{dT}{T} = D^2 \cdot (aW^4 + bW^2 + cW^0) + D^1 \cdot (dW^4 + eW^2 + fW^0) + D^0 \cdot (gW^4 + hW^2 + iW^0)$$

## Specifying As-Drawn or Silicon Values for Width and Density

You can use either the as-drawn layout dimensions or post-etch (silicon) dimensions in the calculations, as follows:

- Conductor width
  - If you use the `WIDTH_POLYNOMIAL_ORDERS` keyword, conductor widths used in calculations are based on the as-drawn layout dimensions. This includes the widths used in the `WIDTH_RANGES` keyword and the widths (but not the densities) used in the `DENSITY_BOUNDS_VS_WIDTH` keyword.
  - If you use the `SI_WIDTH_POLYNOMIAL_ORDERS` keyword, conductor widths are based on the post-etch (silicon) dimensions.
- Conductor density
  - If you use the `DENSITY_POLYNOMIAL_ORDERS` keyword, conductor densities used in calculations are based on the as-drawn layout dimensions. This includes the densities used in the `DENSITY_BOUNDS_VS_WIDTH` keyword.
  - If you use the `SI_DENSITY_POLYNOMIAL_ORDERS` keyword, conductor densities are based on the post-etch (silicon) dimensions.

Using the global `USE_SI_DENSITY` command in the ITF file might conflict with these keywords. [Table 16-2](#) describes the allowed usages.

*Table 16-2 Effect of USE\_SI\_DENSITY Command Settings*

| POLYNOMIAL_BASED_THICKNESS_VARIATION<br>Keyword | USE_SI_DENSITY<br>= YES | USE_SI_DENSITY<br>= NO (or not used) |
|-------------------------------------------------|-------------------------|--------------------------------------|
| <code>WIDTH_POLYNOMIAL_ORDERS</code>            | drawn width             | drawn width                          |
| <code>DENSITY_POLYNOMIAL_ORDERS</code>          | silicon density         | drawn density                        |
| <code>SI_WIDTH_POLYNOMIAL_ORDERS</code>         | error                   | silicon width                        |
| <code>SI_DENSITY_POLYNOMIAL_ORDERS</code>       | error                   | silicon density                      |

## Density Bounds

Use the optional `DENSITY_BOUNDS_VS_WIDTH` keyword to limit the conductor densities to be used in the calculations. Each entry in the argument list consists of one width value and two density values enclosed in parentheses; you can provide more than one such entry.

Whether the width and density values in the `DENSITY_BOUNDS_VS_WIDTH` keyword use as-drawn or post-etch (silicon) values is determined by the presence or absence of the `SI_` prefix in the `WIDTH_POLYNOMIAL_ORDERS` and `DENSITY_POLYNOMIAL_ORDERS` keywords.

The density bounds in the `DENSITY_BOUNDS_VS_WIDTH` keyword are applied as follows:

- If the density is outside the range defined by the minimum and maximum densities, the nearest density bound is used in the calculation.
- If there is only one entry (in other words, one triplet of values) in the `DENSITY_BOUNDS_VS_WIDTH` argument list, the minimum and maximum densities apply to all conductor widths regardless of the width specified in the triplet.
- If there are multiple triples in the keyword argument list, the behavior is as follows:
  - If the conductor width is smaller than the smallest width specified, the minimum and maximum densities for the smallest width are used.
  - If the conductor width is larger than the largest width specified, the minimum and maximum densities for the largest width are used.
  - If the conductor width is between two specified widths, the minimum and maximum densities are determined by interpolating between the enclosing density values.

In the following example, only density values between 0.05 and 1.00 are used when calculating thickness variation for conductor widths up to and including 0.1 microns. Density values from 0.07 to 0.50 are used for conductor widths greater than 0.2 microns. If the conductor width is 0.15 microns, the allowable range of density values is 0.06 to 0.75, as determined by linear interpolation.

```
DENSITY_BOUNDS_VS_WIDTH {
 (0.1 0.05 1.00)
 (0.2 0.07 0.50)
}
```

If you do not use the `DENSITY_BOUNDS_VS_WIDTH` keyword, the density bounds are calculated using the minimum and maximum width and spacing values in the `ETCH_VS_WIDTH_AND_SPACING` command defined for the same conductor. This method is available for backward compatibility, but might result in unreasonable values.

In this method, the minimum and maximum densities are calculated as follows, where D is density, W is width, and S is spacing:

$$D_{\min} = \frac{W_{\min}}{(W_{\min} + S_{\max})}$$

$$D_{\max} = \frac{W_{\max}}{(W_{\max} + S_{\min})}$$

## Thickness Bounds

Use the optional `THICKNESS_BOUNDS` keyword to apply limits to the calculated conductor thickness values, as follows:

- If the conductor thickness calculated from the polynomial function is smaller than the minimum thickness in the `THICKNESS_BOUNDS` argument list, the minimum thickness value is used instead of the calculated thickness.
- If the conductor thickness calculated from the polynomial function is larger than the maximum thickness in the `THICKNESS_BOUNDS` argument list, the maximum thickness value is used instead of the calculated thickness.

### Example

In the following example, the first coefficients table is used for widths less than or equal to 0.27 microns, while the second table is used for widths greater than 0.27 microns. All widths and densities are calculated using post-etch (silicon) dimensions.

```
CONDUCTOR M1 { THICKNESS=0.18 SIDE_TANGENT = 0.0556
 POLYNOMIAL_BASED_THICKNESS_VARIATION {
 SI_DENSITY_POLYNOMIAL_ORDERS = { 3 2 1 0 }
 SI_WIDTH_POLYNOMIAL_ORDERS = { 4 3 2 1 0 }
 WIDTH_RANGES = { 0.27 }
 $ Coefficients for width <= 0.27
 POLYNOMIAL_COEFFICIENTS = {
 0 1.656E+03 -9.488E+02 1.731E+02 -1.041E+01
 0 -1.212E+03 6.935E+02 -1.262E+02 7.666E+00
 0 2.314E+02 -1.320E+02 2.400E+01 -1.580E+00
 0 -5.211E+00 3.417E+00 -6.853E-01 1.131E-01
 }
 $ Coefficients for width > 0.27
 POLYNOMIAL_COEFFICIENTS = {
 1.027E-03 -2.006E-02 8.996E-02 -5.189E-02 -1.814E-01
 -2.805E-03 5.795E-02 -3.084E-01 4.211E-01 1.152E-01
 2.097E-03 -4.375E-02 2.394E-01 -3.662E-01 -2.697E-02
 -4.866E-04 1.001E-02 -5.416E-02 1.012E-01 4.308E-02
 }
 DENSITY_BOUNDS_VS_WIDTH {
 (0.1 0.05 0.97)
 (0.2 0.07 0.49)
 (0.3 0.10 0.52)
 (1.0 0.50 0.97)
 }
 THICKNESS_BOUNDS { 0.30 0.45 }
 }
}
```

## Error Conditions

Several ITF commands provide alternative methods for modeling conductor thickness variation. Therefore, they cannot be specified for the same conducting layer when the POLYNOMIAL\_BASED\_THICKNESS\_VARIATION command is used. These commands are as follows:

- THICKNESS\_VS\_DENSITY
- THICKNESS\_VS\_WIDTH\_AND\_SPACING
- DENSITY\_BOX\_WEIGHTING\_FACTOR

## See Also

- [THICKNESS\\_VS\\_DENSITY](#)
- [THICKNESS\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [DENSITY\\_BOX\\_WEIGHTING\\_FACTOR](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## PROCESS\_FOUNDRY

Specifies an optional foundry identifier as part of a process information keyword group.

### Syntax

PROCESS\_FOUNDRY = *foundry\_id*

### Arguments

| Argument          | Description                                       |
|-------------------|---------------------------------------------------|
| <i>foundry_id</i> | One-word string, intended to identify the foundry |

### Description

The PROCESS\_FOUNDRY keyword is one of the five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS\_FOUNDRY keyword are as follows:

- This keyword is a one-word string; spaces are not allowed.
- Case does not matter; FAB\_1802c is equivalent to fab\_1802C.
- The special characters @, #, and \$ are not allowed.

### Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

---

## PROCESS\_NODE

Specifies an optional foundry identifier as part of a process information keyword group.

### Syntax

PROCESS\_NODE = *node\_number*

### Arguments

| Argument           | Description                                                     |
|--------------------|-----------------------------------------------------------------|
| <i>node_number</i> | Floating-point number, intended to identify the technology node |

### Description

The PROCESS\_NODE keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS\_NODE keyword are as follows:

- The keyword is a nonnegative floating-point number.
- The value is intended to represent a technology node, but there are no requirements for the value.

### Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

---

## PROCESS\_TYPE

Specifies an optional foundry identifier as part of a process information keyword group.

### Syntax

PROCESS\_TYPE = *type\_id*

### Arguments

| Argument       | Description                                       |
|----------------|---------------------------------------------------|
| <i>type_id</i> | One-word string, intended to identify the process |

### Description

The PROCESS\_TYPE keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS\_TYPE keyword are as follows:

- This keyword is a one-word string; spaces are not allowed.
- Case does not matter; SOI is equivalent to soi.
- The special characters @, #, and \$ are not allowed.

### Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

---

## PROCESS\_VERSION

Specifies an optional foundry identifier as part of a process information keyword group.

### Syntax

```
PROCESS_VERSION = version_number
```

### Arguments

| Argument              | Description                                                     |
|-----------------------|-----------------------------------------------------------------|
| <i>version_number</i> | Floating-point number, intended to identify the process version |

### Description

The `PROCESS_VERSION` keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the `TECHNOLOGY` statement
- Specify the keywords in the correct order

Rules for the `PROCESS_VERSION` keyword are as follows:

- The keyword is a nonnegative floating-point number.
- The value is intended to represent a process version such as 1.0 or 2.0, but there are no requirements for the value.

### Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

---

## PROCESS\_CORNER

Specifies an optional foundry identifier as part of a process information keyword group.

### Syntax

PROCESS\_CORNER = *corner\_id*

### Arguments

| Argument         | Description                                              |
|------------------|----------------------------------------------------------|
| <i>corner_id</i> | One-word string, intended to identify the process corner |

### Description

The PROCESS\_CORNER keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS\_CORNER keyword are as follows:

- The keyword is a one-word string; spaces are not allowed.
- Case does not matter; TYPICAL is equivalent to typical.
- The special characters @, #, and \$ are not allowed.

### Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

## RAISED\_DIFFUSIONETCH

Specifies the etch distance of the raised diffusion conductor.

### Syntax

```
RAISED_DIFFUSIONETCH = distance
```

### Arguments

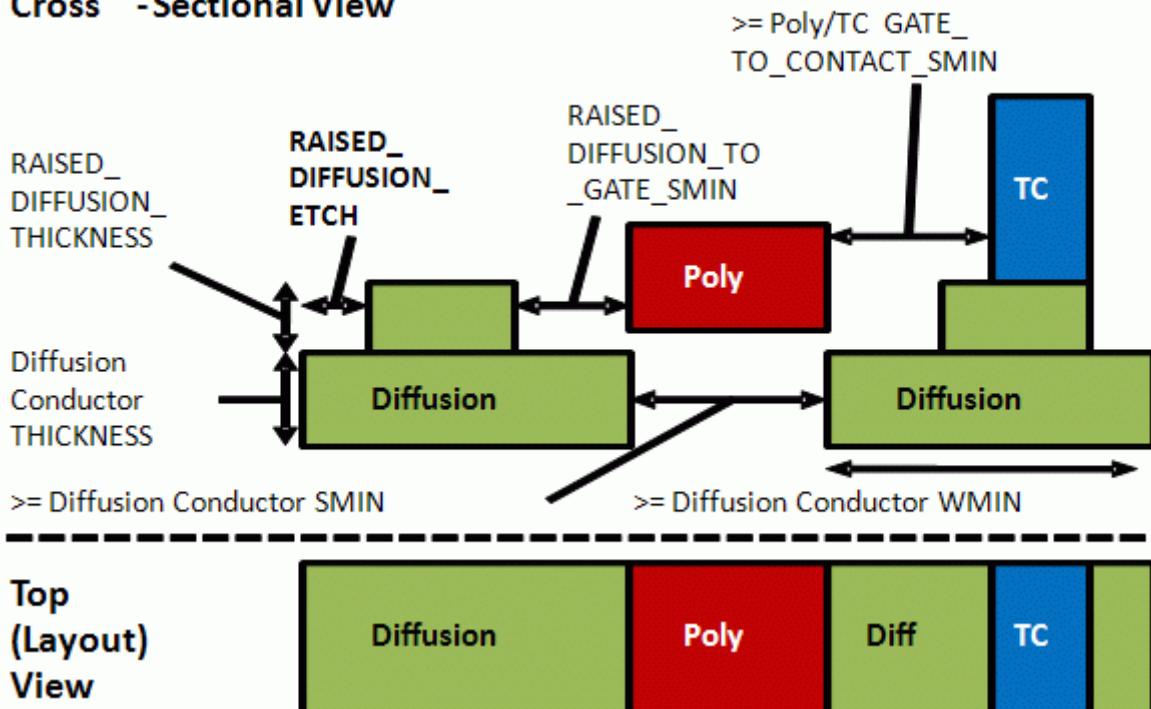
| Argument        | Description                                                       |
|-----------------|-------------------------------------------------------------------|
| <i>distance</i> | Etch distance of the raised diffusion conductor<br>Units: microns |

### Description

The RAISED\_DIFFUSIONETCH option specifies the etch distance of the raised diffusion conductor, on the sides of the diffusion conductor that are not adjacent to a gate or field polysilicon conductor, as shown in [Figure 16-12](#).

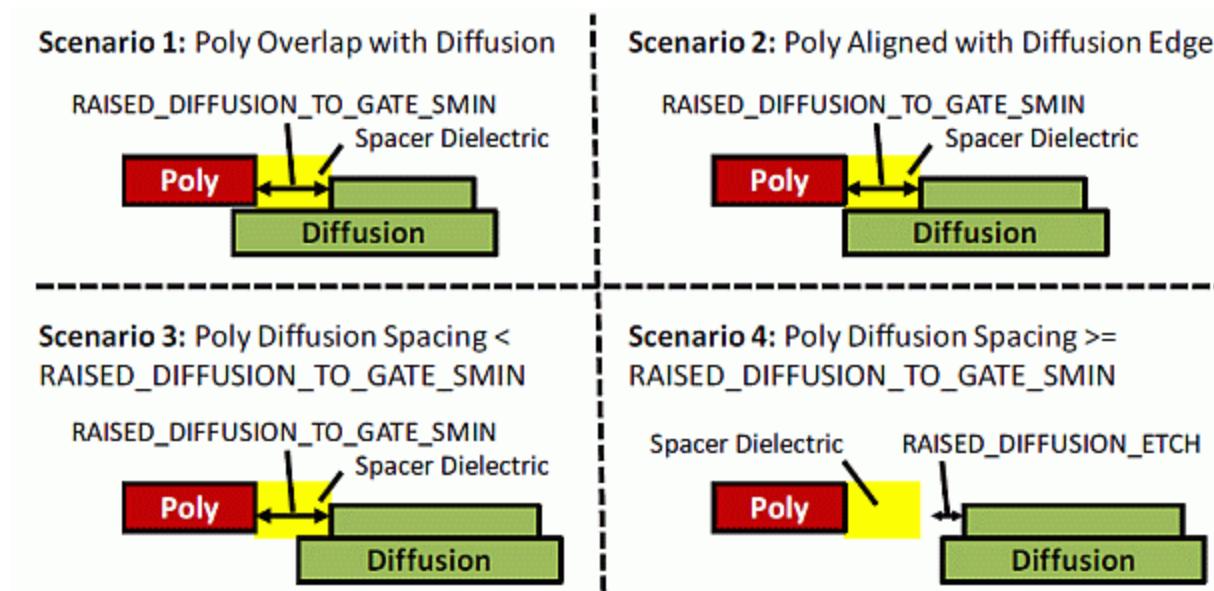
*Figure 16-12 Cross Sectional and Top Views of the Trench Contact Process*

#### Cross -Sectional View



**Figure 16-13** shows the specific scenarios in which the RAISED\_DIFFUSIONETCH and RAISED\_DIFFUSION\_TO\_GATE\_SMIN options are applied. In a typical process, raised diffusion growth is affected by the location of the polysilicon spacer dielectric. The RAISED\_DIFFUSION\_TO\_GATE\_SMIN option models this process effect. The RAISED\_DIFFUSIONETCH option models a different process effect in regions that do not overlap with the polysilicon spacer dielectric. Note that the raised diffusion edge geometry is solely determined by the RAISED\_DIFFUSION\_THICKNESS, RAISED\_DIFFUSION\_TO\_GATE\_SMIN, and RAISED\_DIFFUSIONETCH options. Therefore, the RAISED\_DIFFUSION\_TO\_GATE\_SMIN option is independent of the polysilicon conductor's conformal dielectrics. The RAISED\_DIFFUSIONETCH and RAISED\_DIFFUSION\_TO\_GATE\_SMIN options are applied based on the postetch silicon dimensions of the polysilicon and diffusion conductors.

*Figure 16-13 Polysilicon Spacing From Raised Diffusion Regions*



The RAISED\_DIFFUSIONETCH option has the following constraints:

- RAISED\_DIFFUSIONETCH must be greater than or equal to 0.0.
- RAISED\_DIFFUSIONETCH must be specified with both RAISED\_DIFFUSION\_THICKNESS and RAISED\_DIFFUSION\_TO\_GATE\_SMIN.
- RAISED\_DIFFUSIONETCH must be specified in a conductor layer with LAYER\_TYPE = DIFFUSION.
- To ensure sufficiently wide raised source and drain regions in minimum-size devices, the xy dimension of the raised diffusion must be 5 nm or greater after

`RAISED_DIFFUSION_TO_GATE_SMIN` and `RAISED_DIFFUSIONETCH` are applied. This is equivalent to the following conditions:

- Diffusion WMIN - `RAISED_DIFFUSION_TO_GATE_SMIN` - `RAISED_DIFFUSIONETCH`  $\geq$  5 nm
- Diffusion WMIN -  $2 \cdot \text{RAISED\_DIFFUSIONETCH}$   $\geq$  5 nm
- Diffusion WMIN -  $2 \cdot \text{RAISED\_DIFFUSION\_TO\_GATE\_SMIN}$   $\geq$  5 nm

If any of these constraints are violated in a given ITF file, the `grdgenxo` tool issues an error message.

### Example

```
CONDUCTOR DIFF {
 THICKNESS = 0.05
 LAYER_TYPE = DIFFUSION
 RAISED_DIFFUSIONETCH = 0.005
 RAISED_DIFFUSIONTHICKNESS = 0.015
 RAISED_DIFFUSION_TO_GATE_SMIN = 0.01
 ...
}
```

### See Also

- [CONDUCTOR](#)
- [LAYER\\_TYPE](#)
- [RAISED\\_DIFFUSIONTHICKNESS](#)
- [RAISED\\_DIFFUSION\\_TO\\_GATE\\_SMIN](#)

---

## RAISED\_DIFFUSIONETCH\_TABLE

Specifies the device-dependent etch distance of the raised diffusion conductor.

### Syntax

```
RAISED_DIFFUSIONETCH_TABLE {
 (device_type1 distance1)
 (device_type2 distance2)
 ...
}
```

### Arguments

| Argument           | Description                                                       |
|--------------------|-------------------------------------------------------------------|
| <i>device_type</i> | Device type                                                       |
| <i>distance</i>    | Etch distance of the raised diffusion conductor<br>Units: microns |

### Description

The RAISED\_DIFFUSIONETCH\_TABLE option specifies the device-dependent etch distance of the raised diffusion conductor on the sides of the diffusion conductor that are not adjacent to a gate conductor. The RAISED\_DIFFUSIONETCH keyword specifies the default that is used if the device type is not found in the table.

The device type of the raised source and drain is determined by the surrounding gate device type. The spacing corresponding to the resulting device type is used for the source and drain.

### See Also

- [CONDUCTOR](#)
- [DEVICE\\_TYPE](#)
- [LAYER\\_TYPE](#)
- [RAISED\\_DIFFUSION\\_THICKNESS](#)
- [RAISED\\_DIFFUSION\\_TO\\_GATE\\_SMIN\\_TABLE](#)

---

## RAISED\_DIFFUSION\_GATE\_SIDE\_CONFORMAL\_ER

Specifies the dielectric constant of the area between the raised diffusion and the gate processes.

### Syntax

```
RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER = er_value
```

### Arguments

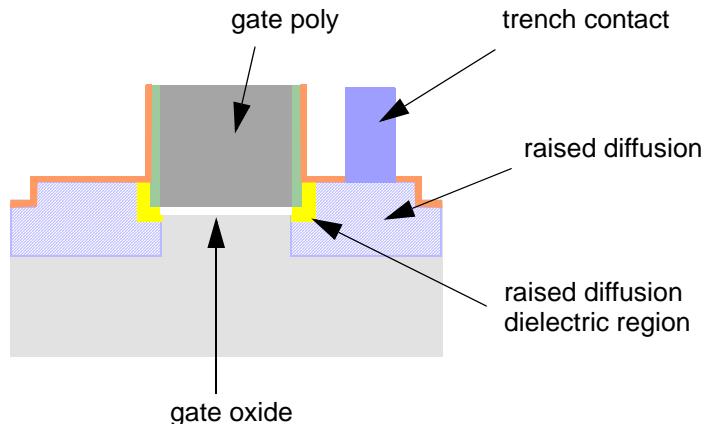
| Argument        | Description                                                                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>er_value</i> | The dielectric constant of the area between the raised diffusion and the gate processes<br>Units: microns<br>Default: Dielectric constant of the raised diffusion area. |

### Description

The RAISED\_DIFFUSION\_GATE\_SIDE\_CONFORMAL\_ER option specifies the dielectric constant of the dielectric region between the raised diffusion and the gate processes when you model silicon dielectrics underneath the gate and diffusion conductors. This option works with the BW\_T option, which models the associated silicon dielectrics.

Place the RAISED\_DIFFUSION\_GATE\_SIDE\_CONFORMAL\_ER option within the raised diffusion CONDUCTOR statement. This option can only be placed in conductors containing RAISED\_DIFFUSION\_THICKNESS and RAISED\_DIFFUSION\_TO\_GATE\_SMIN definitions.

Figure 16-14 Area Between the Raised Diffusion and the Gate Processes



### Example

The following example uses the `RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER` option to specify the dielectric constant of the raised diffusion dielectric region, shown in Figure 16-14.

```
CONDUCTOR DIFF { THICKNESS=0.1
RAISED_DIFFUSION_THICKNESS=0.02
RAISED_DIFFUSION_TO_GATE_SMIN=0.03
RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER=5.0 ... }
```

### See Also

- [MEASURED\\_FROM](#)
- [SW\\_T](#)
- [BW\\_T](#)
- [THICKNESS](#)

---

## RAISED\_DIFFUSION\_THICKNESS

Specifies additional diffusion thickness in raised source and drain regions.

### Syntax

```
RAISED_DIFFUSION_THICKNESS = thickness
```

### Arguments

| Argument         | Description                                                                         |
|------------------|-------------------------------------------------------------------------------------|
| <i>thickness</i> | Additional diffusion thickness in raised source and drain regions<br>Units: microns |

### Description

The RAISED\_DIFFUSION\_THICKNESS option specifies additional diffusion thickness in raised source and drain regions, as shown in [Figure 16-12](#).

Specify the RAISED\_DIFFUSION\_THICKNESS option within a CONDUCTOR statement with LAYER\_TYPE = DIFFUSION. The raised portion of the diffusion is not applied to regions that are closer than the specified RAISED\_DIFFUSION\_TO\_GATE\_SMIN value to conductors with the GATE or FIELD\_POLY layer type.

### Example

In the following example, the raised diffusion region exists in all diffusion conductors at a spacing of 10 nm from adjacent polysilicon conductors. The raised diffusion region extends 11 nm above the nominal diffusion height. Therefore, the raised diffusion region is covertical with the bottom 10 nm of the polysilicon conductor. The DIFF\_NO\_RSD layer is a standard diffusion layer without raised source and drain regions, which can also exist in the process.

*Example 16-5 Raised Diffusion Definition*

```
CONDUCTOR PS {
 THICKNESS=0.04
 WMIN=0.04
 SMIN=0.04
 GATE_TO_CONTACT_SMIN=0.02
 LAYER_TYPE=GATE
 ...
}
DIELECTRIC DP1 {
 THICKNESS=0.001
 ...
}
DIELECTRIC D_DIFF {
 THICKNESS=0.04
 ...
}
CONDUCTOR DIFF {
 THICKNESS=0.04
 WMIN=0.04
 SMIN=0.04
 RAISED_DIFFUSION_THICKNESS=0.011
 RAISED_DIFFUSION_TO_GATE_SMIN=0.01
 LAYER_TYPE=DIFFUSION
 ...
}
CONDUCTOR DIFF_NO_RSD {
 THICKNESS=0.04
 WMIN=0.04
 SMIN=0.04
 LAYER_TYPE=DIFFUSION
 ...
}
```

**See Also**

- [CONDUCTOR](#)
- [LAYER\\_TYPE](#)
- [RAISED\\_DIFFUSIONETCH](#)
- [RAISED\\_DIFFUSION\\_TO\\_GATE\\_SMIN](#)

---

## RAISED\_DIFFUSION\_TO\_GATE\_SMIN

Specifies the minimum lateral spacing between raised source and drain regions and gate or field polysilicon conductors.

### Syntax

`RAISED_DIFFUSION_TO_GATE_SMIN = spacing`

### Arguments

| Argument       | Description                                                                       |
|----------------|-----------------------------------------------------------------------------------|
| <i>spacing</i> | Minimum lateral spacing between raised source and drain regions<br>Units: microns |

### Description

The `RAISED_DIFFUSION_TO_GATE_SMIN` option specifies the minimum lateral spacing between raised source and drain regions and gate or field polysilicon conductors, as shown in [Figure 16-12](#).

Specify the `RAISED_DIFFUSION_TO_GATE_SMIN` option within a `CONDUCTOR` block that contains a `LAYER_TYPE = DIFFUSION` statement.

### See Also

- [CONDUCTOR](#)
- [LAYER\\_TYPE](#)
- [RAISED\\_DIFFUSIONETCH](#)
- [RAISED\\_DIFFUSION\\_THICKNESS](#)

---

## RAISED\_DIFFUSION\_TO\_GATE\_SMIN\_TABLE

Specifies the device-dependent minimum lateral spacing between raised source and drain regions and gate conductors.

### Syntax

```
RAISED_DIFFUSION_TO_GATE_SMIN_TABLE {
 (device_type1 spacing1)
 (device_type2 spacing2)
 ...
}
```

### Arguments

| Argument           | Description                                                                       |
|--------------------|-----------------------------------------------------------------------------------|
| <i>device_type</i> | Device type                                                                       |
| <i>spacing</i>     | Minimum lateral spacing between raised source and drain regions<br>Units: microns |

### Description

The RAISED\_DIFFUSION\_TO\_GATE\_SMIN\_TABLE option specifies the device-dependent minimum lateral spacing between raised source and drain regions and gate conductors. The RAISED\_DIFFUSION\_TO\_GATE\_SMIN keyword specifies the default that is used if the device type is not found in the table.

Specify the RAISED\_DIFFUSION\_TO\_GATE\_SMIN\_TABLE option within a CONDUCTOR statement that contains the LAYER\_TYPE = DIFFUSION statement.

The device type of the raised source and drain is determined by the surrounding gate device type. The spacing corresponding to that device type is used for the source and drain.

### Example

```
RAISED_DIFFUSION_TO_GATE_SMIN_TABLE
 {(G_1D5VIO_PMOS 0.02) (G_CORE_PMOS 0.015)}
```

### See Also

- [CONDUCTOR](#)
- [LAYER\\_TYPE](#)
- [RAISED\\_DIFFUSION\\_TO\\_GATE\\_SMIN](#)

---

## REFERENCE\_DIRECTION

Specifies the reference direction of the process technology.

### Syntax

```
REFERENCE_DIRECTION = VERTICAL | HORIZONTAL | NONE | GATE
```

### Arguments

| Argument       | Description                                            |
|----------------|--------------------------------------------------------|
| VERTICAL       | Reference direction is vertical                        |
| HORIZONTAL     | Reference direction is horizontal                      |
| NONE (default) | Directional etch tables are ignored                    |
| GATE           | Reference direction is determined by device pin layout |

### Description

The `REFERENCE_DIRECTION` statement defines the reference direction for the application of orientation-dependent etch defined by the `ETCH_VS_WIDTH_AND_SPACING` statement. The `REFERENCE_DIRECTION` statement appears in the header of an ITF file with other global statements such as the `TECHNOLOGY` statement.

You can specify the reference direction in the StarRC command file or the ITF file. If the reference direction is specified in both files, the StarRC command file takes precedence.

If you set the reference direction to `GATE`, the StarRC tool uses the locations of the gate, drain, and source pins of the first encountered transistor to determine whether the reference direction should be `VERTICAL` or `HORIZONTAL`. The same reference direction is used for all devices in the design.

### Example

The following example specifies that the reference direction is horizontal:

```
REFERENCE_DIRECTION = HORIZONTAL
```

### See Also

- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [REFERENCE\\_DIRECTION \(StarRC command\)](#)

---

## RESISTIVE\_ONLYETCH

Identical to the `ETCH` option, except that only resistance is affected. Valid within a `CONDUCTOR` block.

### Syntax

`RESISTIVE_ONLYETCH = etch_value`

### Arguments

| Argument                | Description                                                                                                                        |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <code>etch_value</code> | Absolute width adjustment for one sidewall. A positive value shrinks the conductor; a negative value expands it.<br>Units: microns |

### Description

The `RESISTIVE_ONLYETCH` option applies an etch value to the sidewalls of a conductor. A positive value denotes conductor shrink; a negative value denotes conductor expansion. The adjusted conductor width is equal to the drawn width minus twice the etch value.

Use this option instead of the `ETCH` option to specify that an etch operation is to be used only for resistance calculations.

If you use one of the `ETCH` options in addition to one or more `ETCH_VS_WIDTH_AND_SPACING` tables, the `ETCH_VS_WIDTH_AND_SPACING` operations are applied first, followed by the `ETCH` operation.

This option is not the same as `ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY`.

This option works only with `EXTRACTION:RC`. Otherwise resistance extraction does not have etching effects.

### Example

```
CONDUCTOR metall1 {
 RESISTIVE_ONLYETCH = 0.05
 THICKNESS=0.66
 WMIN=0.15 SMIN=0.15 RPSQ=0.078
}
```

### See Also

- [CAPACITIVE\\_ONLYETCH](#)
- [ETCH](#)

---

## RHO

Defines the bulk resistivity of a VIA or CONDUCTOR layer.

### Syntax

RHO = *rho\_value*

### Arguments

| Argument         | Description                                                                                                                                |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <i>rho_value</i> | Bulk resistivity of the via or conductor layer<br>Units: ohms-micron<br>Default: 0.0 for conductors<br>RPV x AREA equal to 1.0e-6 for vias |

### Description

Via layer resistance properties must be specified in one of three mutually exclusive ways:  
RHO, RPV and AREA, or RPV\_VS\_AREA.

Conductor resistance properties must be specified in one of two mutually exclusive ways:  
RHO or RPSQ.

### Example

```
VIA vial {FROM=M1 TO=M2 RHO=0.263}
CONDUCTOR M1 {THICKNESS=0.4 SMIN=0.15 WMIN=0.18 RHO=0.8}
```

### See Also

- [RHO\\_VS\\_SI\\_WIDTH\\_AND\\_THICKNESS](#)
- [RHO\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [RPV](#)
- [RPV\\_VS\\_AREA](#)
- [RPV\\_VS\\_COVERAGE](#)
- [RPV\\_VS\\_WIDTH\\_AND\\_LENGTH](#)

---

## RHO\_VS\_SI\_WIDTH\_AND\_THICKNESS

Models resistivity as a function of silicon (post-etch) width and thickness. Valid within a CONDUCTOR block.

### Syntax

```
RHO_VS_SI_WIDTH_AND_THICKNESS {
 WIDTH { w1 w2 w3 ... }
 THICKNESS { t1 t2 t3 ... }
 VALUES { v(w1,t1) v(w2,t1) ...
 v(w1,t2) v(w2,t2) ...
 ...
 }
}
```

### Arguments

| Argument                       | Description                                                     |
|--------------------------------|-----------------------------------------------------------------|
| w1 w2 w3 ...                   | Silicon widths of the conductor<br>Units: microns               |
| t1 t2 t3 ...                   | Silicon thicknesses of the conductor<br>Units: microns          |
| v(w1,t1) v(w2,t1) v(w1,t2) ... | Resistivity values for the corresponding widths and thicknesses |

### Description

The RHO\_VS\_SI\_WIDTH\_AND\_THICKNESS option models resistivity as a function of silicon width and thickness.

You can define the THICKNESS values before the WIDTH values; however, the mapping of the resistivity values remains the same regardless of the order of the WIDTH and THICKNESS definitions.

The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

When using the `RHO_VS_SI_WIDTH_AND_THICKNESS` option, StarRC issues a warning if

- You do not specify one of the following ITF options for the conductor:
  - `POLYNOMIAL_BASED_THICKNESS_VARIATION`
  - `THICKNESS_VS_DENSITY`
  - `THICKNESS_VS_WIDTH_AND_SPACING`
- You do not specify one of the following ITF options for the conductor:
  - `ETCH`
  - `ETCH_VS_WIDTH_AND_SPACING (RESISTIVE_ONLY)`

### **Example**

```
RHO_VS_SI_WIDTH_AND_THICKNESS {
 WIDTH {0.1 0.2 0.3 0.4}
 THICKNESS {0.11 0.22 0.33}
 VALUES { 0.304 0.410 0.518 0.640
 0.210 0.340 0.438 0.560
 0.504 0.530 0.618 0.720
 }
}
```

### **See Also**

- [POLYNOMIAL\\_BASED\\_THICKNESS\\_VARIATION](#)
- [THICKNESS\\_VS\\_DENSITY](#)
- [THICKNESS\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [ETCH](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## RHO\_VS\_WIDTH\_AND\_SPACING

Models resistivity variation with respect to width and spacing. Valid within a CONDUCTOR block.

### Syntax

```
RHO_VS_WIDTH_AND_SPACING
 SPACINGS {s1 s2}
 WIDTHS {w1 w2 w3}
 VALUES {v(s1 w1) v(s2 w1)
 v(s1 w2) v(s2 w2)
 v(s1 w3) v(s2 w3) }
```

### Arguments

| Argument       | Description                                                                                                                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPACINGS {...} | Spacing to the nearest conductor<br>Units: microns                                                                                                                                                                   |
| WIDTHS {...}   | Conductor widths<br>Units: microns                                                                                                                                                                                   |
| VALUES {...}   | The RHO values for the corresponding width and spacing. The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.<br>Units: ohm-microns |

### Description

Specify this option to model conductor resistivity variation with respect to width and spacing.

The RHO\_VS\_WIDTH\_AND\_SPACING option cannot be used in conjunction with the RHO, RPSQ, and RPSQ\_VS\_WIDTH\_AND\_SPACING options.

#### Note:

If multiple ETCH\_VS\_WIDTH\_AND\_SPACING tables are used in addition to RPSQ\_VS\_WIDTH\_AND\_SPACING or RHO\_VS\_WIDTH\_AND\_SPACING tables, the StarRC tool issues a warning message. This combination results in many sources of variation for the same conductor and might lead to extraction inaccuracy.

---

## RPSQ

Specifies the resistance per square (RPSQ) of a conductor layer.

### Syntax

`RPSQ = rpsq_value`

### Arguments

| Argument                | Description                                                                  |
|-------------------------|------------------------------------------------------------------------------|
| <code>rpsq_value</code> | Sheet resistance of the conducting layer<br>Default: 0<br>Units: ohms/square |

### Description

RPSQ is the sheet resistance (in units of ohms per square) of a conductor. You can specify the resistive properties of CONDUCTOR layers using either the RPSQ or the RHO values, but only one is required.

You can also specify the RPSQ value in the mapping file within the `conducting_layers` statement. The RPSQ value specified in the mapping file overrides any RPSQ, RPSQ\_VS\_WIDTH\_AND\_SPACING, or RPSQ\_VS\_SI\_WIDTH statements in the ITF file.

The RPSQ value is modified by the conductor thickness variation specified by the BOTTOM\_THICKNESS\_VS\_SI\_WIDTH statement unless the CAPACITANCE\_ONLY flag is used.

### Example

```
CONDUCTOR metall1 {
 RESISTIVE_ONLYETCH=0.05 THICKNESS=0.66
 WMIN=0.15 SMIN=0.15 RPSQ=0.078
}
```

### See Also

- [RHO](#)
- [RPSQ\\_VS\\_SI\\_WIDTH](#)
- [RPSQ\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## RPSQ\_VS\_SI\_WIDTH

Defines the conductor sheet resistance as a function of the conductor width. Valid in a CONDUCTOR block.

### Syntax

```
RPSQ_VS_SI_WIDTH {
 (SIW1, R1)
 (SIW2, R2)
 ...
 (SIWn, Rn)
}
```

### Arguments

| Argument    | Description                                                            |
|-------------|------------------------------------------------------------------------|
| <i>SIWn</i> | Post-etch conductor width<br>Units: microns                            |
| <i>Rn</i>   | Conductor sheet resistance for width <i>SIWn</i><br>Units: ohms/square |

### Description

The `RPSQ_VS_SI_WIDTH` table specifies conductor sheet resistance (RPSQ) as a function of conductor width. Resistance might vary with line width due to effects such as cladding and dishing.

The conductor width in this table is the post-etch width. ITF commands that

The first entry of `SIW` does not have to be the same as `WMIN`.

You can also specify the RPSQ value in the mapping file within the `conducting_layers` statement. The RPSQ value specified in the mapping file overrides any `RPSQ`, `RPSQ_VS_WIDTH_AND_SPACING`, or `RPSQ_VS_SI_WIDTH` statements in the ITF file.

The RPSQ value is modified by the conductor thickness variation specified by the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement unless the `CAPACITANCE_ONLY` flag is used.

### Errors

The following ITF options are mutually exclusive methods for specifying conductor resistance: `RPSQ`, `RPSQ_VS_WIDTH_AND_SPACING`, `RPSQ_VS_SI_WIDTH`, `RHO`, `RHO_VS_SI_WIDTH_AND_THICKNESS`, and `RHO_VS_WIDTH_AND_SPACING`.

## Example

This example specifies a varying RPSQ value with respect to the conductor width.

```
CONDUCTOR MET1 {
 THICKNESS=0.6 WMIN=0.34 SMIN=0.40
 RPSQ_VS_SI_WIDTH {
 (0.34, 0.075) (0.40, 0.062)
 (0.823, 0.0817)(2.0, 0.0321)
 (6.0,0.0173)
 }
}
```

## See Also

- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [RPSQ](#)
- [RPSQ\\_VS\\_WIDTH\\_AND\\_SPACING](#)

---

## RPSQ\_VS\_SI\_WIDTH\_AND\_LENGTH

Models nonlinear gate resistances. Valid within a CONDUCTOR block.

### Syntax

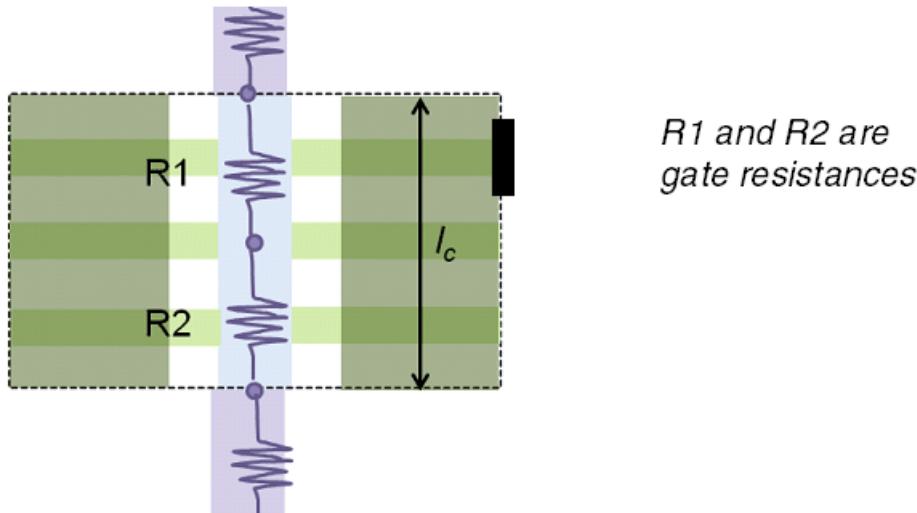
```
RPSQ_VS_SI_WIDTH_AND_LENGTH {
 WIDTHS { w1 w2 ... }
 LENGTHS { l1 l2 ... }
 VALUES {RPSQ(w1,l1) RPSQ(w1,l2) ...
 RPSQ(w2,l1) RPSQ(w2,l2) ...
 ... }
}
```

### Arguments

| Argument    | Description                                                                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| w1, w2, ... | Gate width values, in ascending order<br>Units: microns                                                                                                                                                           |
| l1, l2, ... | Gate length values, in ascending order<br>Units: microns                                                                                                                                                          |
| RPSQ(w,l)   | RPSQ values as a function of length and width<br>Units: ohms per square<br>The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters. |

### Description

The RPSQ\_VS\_SI\_WIDTH\_AND\_LENGTH statement models nonlinear gate resistance resulting from the geometry of gate conductors. This statement must be specified with either LAYER\_TYPE=GATE and LAYER\_TYPE=FIELD\_POLY.

*Figure 16-15 FinFET Top View*

The `grdgenxo` tool stores the resistance in the `nxtgrd` file. StarRC extraction calculates, using linear interpolation if necessary, the resistance as

$$R(W_m, L_n, H) = RPSQ_{n,m} * L_n / W_m$$

If a conductor is longer than the maximum gate width in the table, the maximum gate width value for `RPSQ` is used. If a conductor is shorter than the minimum gate width in the table, the minimum gate width value for `RPSQ` is used.

If a conductor is wider than the maximum gate length in the table, the maximum gate length value for `RPSQ` is used. If a conductor is narrower than the minimum gate length in the table, the minimum gate length value for `RPSQ` is used.

The `RPSQ` value is modified by the conductor thickness variation specified by the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement unless the `CAPACITANCE_ONLY` flag is used.

## Example

The following example specifies a varying RPSQ value for a conductor layer named GATE:

*Table 16-3*

| RPSQ [Ohm/sq]                            | LENGTHS in gate width [microns] |      |      |
|------------------------------------------|---------------------------------|------|------|
| WIDTHS<br>wm<br>gate length<br>[microns] | 0.05                            | 0.1  | 0.15 |
|                                          | 0.02                            | 0.1  | 0.23 |
|                                          | 0.022                           | 0.12 | 0.26 |
|                                          | 0.024                           | 0.18 | 0.28 |
|                                          |                                 |      | 0.53 |

```

CONDUCTOR GATE {
 THICKNESS = 0.6 WMIN = 0.3 SMIN = 0.15
 LAYER_TYPE=GATE
 RPSQ_VS_SI_WIDTH_AND_LENGTH {
 LENGTHS {0.05 0.1 0.15}
 WIDTHS {0.02 0.022 0.024}
 VALUES {0.1 0.23 0.45
 0.12 0.26 0.47
 0.18 0.28 0.53 }
 }
}

```

## See Also

- [RPSQ\\_VS\\_SI\\_WIDTH](#)
- [RPSQ](#)
- [MULTIGATE](#)

---

## RPSQ\_VS\_WIDTH\_AND\_SPACING

Specifies the resistance per square (RPSQ) for different conductor widths and spacings.  
Valid within a CONDUCTOR block.

### Syntax

```
RPSQ_VS_WIDTH_AND_SPACING {
 SPACINGS {s1 s2 s3 ... }
 WIDTHS {w1 w2 w3 ... }
 VALUES {v(s1,w1) v(s2,w1) ...
 v(s1,w2) v(s2,w2) ...
 }
}
```

### Arguments

| Argument       | Description                                                                                                                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPACINGS {...} | Spacings to the nearest conductor<br>Units: microns                                                                                                                                                                                   |
| WIDTHS {...}   | Conductor widths<br>Units: microns                                                                                                                                                                                                    |
| VALUES {...}   | The RPSQ values for the corresponding width and spacing values<br>Units: ohms per square<br>The numbers in the VALUES field are interpreted on a sequential basis,<br>independent of any carriage returns or other hidden characters. |

### Description

The RPSQ\_VS\_WIDTH\_AND\_SPACING table models the effect of different conductor widths and spacings on RPSQ. Use this table to model process effects such as conductor cladding or dishing. The grdgenxo tool analyzes the RPSQ values for the different widths and spaces and stores them in the nxtgrd file. If RPSQ has only a dependency on width, SPACINGS can be skipped and vice versa. The first entry of SPACINGS and WIDTHS should be the same as SMIN and WMIN, respectively.

The RPSQ\_VS\_WIDTH\_AND\_SPACING option cannot be used with the RHO, RPSQ, and RHO\_VS\_WIDTH\_AND\_SPACING options.

#### Note:

If multiple ETCH\_VS\_WIDTH\_AND\_SPACING tables are used in addition to RPSQ\_VS\_WIDTH\_AND\_SPACING or RHO\_VS\_WIDTH\_AND\_SPACING tables, the StarRC tool

issues a warning message. This combination results in many sources of variation for the same conductor and might lead to extraction inaccuracy.

The RPSQ value is modified by the conductor thickness variation specified by the BOTTOM\_THICKNESS\_VS\_SI\_WIDTH statement unless the CAPACITANCE\_ONLY flag is used.

You can also specify the RPSQ value in the mapping file within the conducting\_layers statement. The RPSQ value specified in the mapping file overrides any RPSQ, RPSQ\_VS\_WIDTH\_AND\_SPACING, or RPSQ\_VS\_SI\_WIDTH statements in the ITF file.

### Example

```
CONDUCTOR m1 {
 THICKNESS = 0.6 WMIN = 0.25 SMIN = 0.25
 RPSQ_VS_WIDTH_AND_SPACING {
 SPACINGS {0.25 0.3}
 WIDTHS {0.25 0.3}
 VALUES {0.1 0.05 0.05 0.01} }
 }
}
```

### See Also

- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [RPSQ](#)
- [RPSQ\\_VS\\_SI\\_WIDTH](#)

---

## RPV

Specifies the resistive properties of a via layer. Valid within a VIA block.

### Syntax

RPV = *rpv\_value*

### Arguments

| Argument         | Description                       |
|------------------|-----------------------------------|
| <i>rpv_value</i> | Resistance per via<br>Units: ohms |

### Description

The resistive properties of the via layer must be specified. The via resistance can be specified in one of three mutually exclusive ways: RHO, RPV and AREA, or RPV\_VS\_AREA.

If RPV is specified, AREA is also required.

The default of RPV is such that RPV x AREA = 1.0e-6.

### Example

```
VIA via1 {
 FROM=m1 TO=m2 AREA=0.5 RPV=4
}
```

### See Also

- [RHO](#)
- [RPV\\_VS\\_AREA](#)

---

## RPV\_VS\_AREA

Specifies the resistance per via (RPV) as a function of via area. Valid within a VIA block.

### Syntax

```
RPV_VS_AREA {
 (area1, rpv1)
 (area2, rpv2)
 (area3, rpv3)
 ...
}
```

### Arguments

| Argument | Description                       |
|----------|-----------------------------------|
| area     | Via area<br>Units: square microns |
| rpv      | Resistance per via<br>Units: ohms |

### Description

Use the RPV\_VS\_AREA table in a VIA statement to specify the via resistance as a function of via size. The RPV\_VS\_AREA cannot be used with RPV or RHO in the same VIA statement.

A via definition must use one of the following mutually exclusive methods to define via resistance:

- Specify the bulk resistivity with the RHO statement. The resistance of a via is calculated as

$$R = \frac{\rho \times t}{l \times w}$$

where  $\rho$  is the bulk resistivity,  $t$  is the via thickness (height), and  $l$  and  $w$  are the lateral via length and width.

- Specify the resistance per via with the RPV statement and the via area with the AREA statement. The resistance of a via having an arbitrary area is calculated as

$$R = \frac{RPV \times AREA}{l \times w}$$

where RPV and AREA are the values specified by the ITF statements and  $l$  and  $w$  are the lateral via length and width of an arbitrary via.

- Specify a table of values with the `RPV_VS_AREA` statement.

RPV and AREA values specified in a mapping file take precedence over the `RPV_VS_AREA` values specified in an ITF file.

### Interpolation and Extrapolation

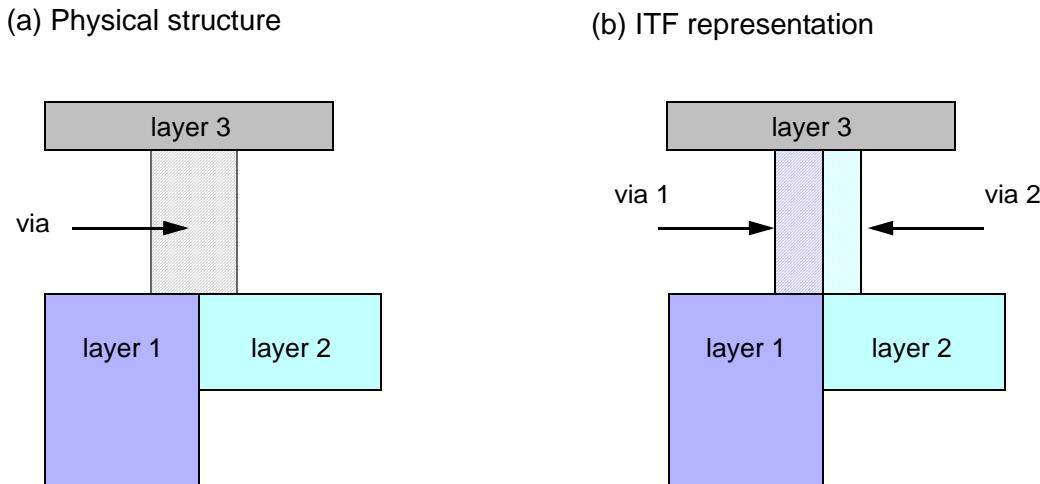
Interpolation and extrapolation are handled as follows:

- If the via area falls between two area entries in the `RPV_VS_AREA` table, the tool converts resistance to conductance, performs linear interpolation in the conductance domain, then converts conductance to resistance.
- If the via area is less than the smallest area entry in the `RPV_VS_AREA` table, StarRC determines the conductance of the via with the smallest area entry and uses that value to calculate the via resistance for any vias with smaller areas.
- If the via area is greater than the largest area entry in the `RPV_VS_AREA` table, StarRC determines the conductance of the via with the largest area entry and uses that value to calculate the via resistance for any vias with larger areas.

### Abutting Vias

Physically, a single via might land on the boundary between two conductor layers. However, ITF process description rules require that a via be defined between one upper layer and one lower layer. Therefore the nxtgrd file represents the physical via as two abutting vias, with one via for each conductor layer.

*Figure 16-16 Abutting Vias*



The StarRC tool handles abutting vias in the presence of `RPV_VS_AREA` tables as follows:

- Adds the areas of the two vias
- Uses the merged area to calculate the total resistance using the `RPV_VS_AREA` table
- Distributes the total resistance between the original vias based on the original via areas

Guidelines:

- You should ensure that the `RPV_VS_AREA` tables in the separate `VIA` blocks are identical. The StarRC tool does not check for equivalency.
- This procedure does not apply to trench contact fake vias or to vias that are overlapping instead of abutting.
- This procedure applies only to transistor-level extraction.

### **Example**

```
VIA vial {
 FROM=m1 TO=m2
 RPV_VS_AREA { (200, 0.5) (350, 0.5) (600, 0.25) }
}
```

### **See Also**

- [VIA](#)

---

## RPV\_VS\_COVERAGE

Specifies resistance per via (RPV) values with respect to via size, upper layer coverage, and lower layer coverage. Valid within a **VIA** block.

### Syntax

```
RPV_VS_COVERAGE [UPPER_LAYER_ONLY] {
 VIA_SIZE {sx1 sy1}
 COV_X {cx1, cx2, ... cxm}
 COV_Y {cy1, cy2, ... cyn}
 VALUES {r(cx1,cy1) r(cx1,cy2) ... r(cx1,cyn)
 r(cx2,cy1) r(cx2,cy2) ... r(cx2,cyn)
 ...
 r(cxm,cy1) r(cxm,cy2) ... r(cxm,cyn)}
 }
 [VIA_SIZE {sx2 sy2}]
 COV_X {dx1, dx2, ... dxm}
 COV_Y {dy1, dy2, ... dyn}
 VALUES {r(dx1,dy1) r(dx1,dy2) ... r(dx1,dyn)
 r(dx2,dy1) r(dx2,dy2) ... r(dx2,dyn)
 ...
 r(dxm,dy1) r(dxm,dy2) ... r(dxm,dyn)}
 }
 ...
}
```

### Arguments

| Argument         | Description                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------|
| UPPER_LAYER_ONLY | Optional keyword that restricts the coverage check to the upper layer                                         |
| sx1 sy1          | Via size in the x- and y-dimensions (as-drawn dimensions)<br>Units: microns                                   |
| cx1, cx2, ...    | X-direction coverage values, in ascending order; coverage values for different via sizes do not have to match |
| dx1, dx2, ...    | Units: microns                                                                                                |
| cy1, cy2, ...    | Y-direction coverage values, in ascending order; coverage values for different via sizes do not have to match |
| dy1, dy2, ...    | Units: microns                                                                                                |
| r(cx1,cy1)       | Resistance per via (RPV) for the corresponding coverage values                                                |
| r(dx1,dy1)       | Units: ohms                                                                                                   |

## Description

The `RPV_VS_COVERAGE` command specifies resistance per via (RPV) values based on the via size and the amount of upper and lower conductor layer coverage.

### Note:

The `RPV_VS_COVERAGE` command uses as-drawn dimensions for the via and for the upper and lower layer coverage features. You can use after-etch dimensions for the upper layer coverage analysis by using the `RPV_VS_SI_COVERAGE` command instead.

The `RPV_VS_COVERAGE` command contains one or more tables of RPV values. Each table applies to a specific via size based on the drawn dimensions. The via dimensions must match exactly; however, the x- and y-dimensions can be rotated by 90 degrees. When rotation is required to match the via size, the table indexes and values are also rotated.

If a via does not match any of the via sizes specified in the `RPV_VS_COVERAGE` command, the StarRC tool uses the `RPV_VS_AREA` command to determine the via resistance.

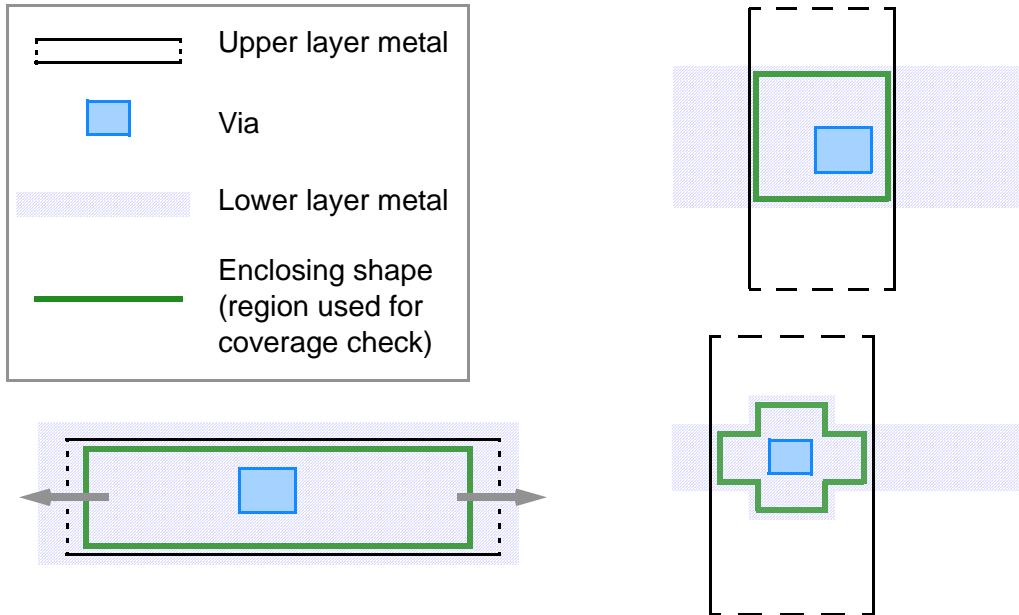
If a via matches a via size specified by a `VIA_SIZE` keyword in the `RPV_VS_COVERAGE` command, the StarRC tool determines the via resistance using the following procedure:

1. Find the enclosing shape that covers the via.

The tool performs a Boolean AND operation of the upper conductor layer and lower conductor layer to get the enclosing shape that covers the via, as shown in [Figure 16-17](#).

You can optionally use only the upper layer for the coverage check by specifying the `UPPER_LAYER_ONLY` keyword immediately after the `RPV_VS_COVERAGE` command.

*Figure 16-17 Via Coverage Enclosing Shape*



2. Decompose the enclosing shape into rectangles.

A complex enclosing shape can be viewed as a set of overlapping boxes, as shown in [Figure 16-18](#).

3. Evaluate the available coverage boxes.

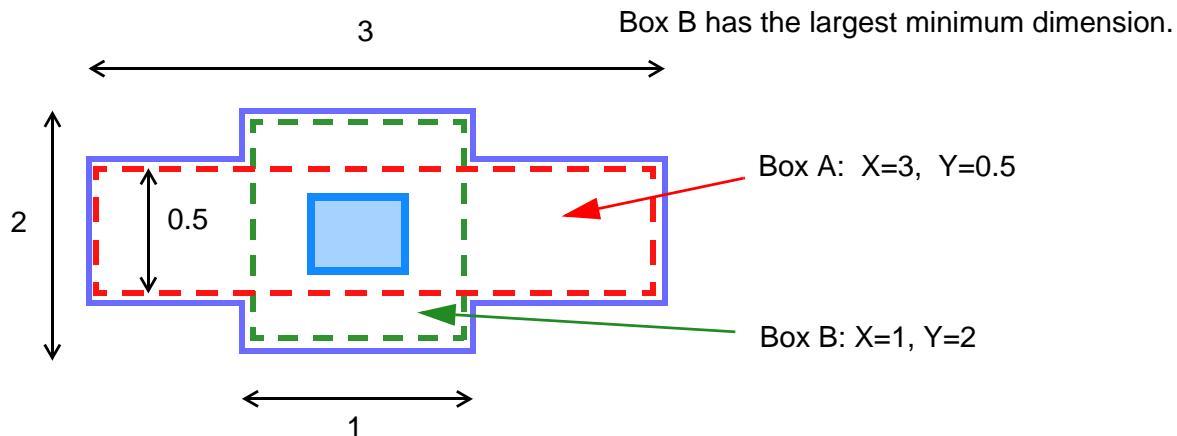
The `cov_x` and `cov_y` values in the `RPV_VS_SI_COVERAGE` command define a set of coverage boxes with those X and Y dimensions. (The number of available boxes is the product of the number of `cov_x` values and the number of `cov_y` values.)

The StarRC tool determines the largest coverage box that, when centered over the via, fits within the enclosing shape.

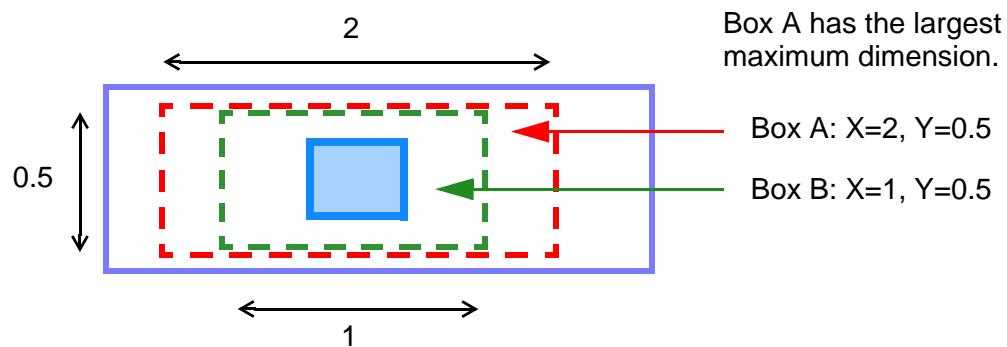
If multiple coverage boxes fit within the enclosing shape, the box with the largest minimum dimension is selected, as shown in [Figure 16-18](#).

If multiple coverage boxes have the same minimum dimension values, the box with the largest maximum dimension is selected. An example is shown in [Figure 16-19](#).

*Figure 16-18 Coverage Rule Selection Example 1*



*Figure 16-19 Coverage Rule Selection Example 2*



#### 4. Look up or calculate the resistance per via.

The possible configurations are as follows, assuming that the coverage box is centered over the via.

- If all of the coverage boxes are too large to fit within the enclosing shape, the tool uses the RPV value of the smallest coverage box.
- If all of the coverage boxes fit within the enclosing shape, but the enclosing shape extends beyond the largest coverage box in both X and Y dimensions, the tool uses the RPV value of the largest coverage box.
- If an edge of the enclosing shape falls between the edges of two coverage boxes, linear interpolation is used to calculate the RPV value. Interpolation in both the X and Y dimensions might be necessary.

### Examples

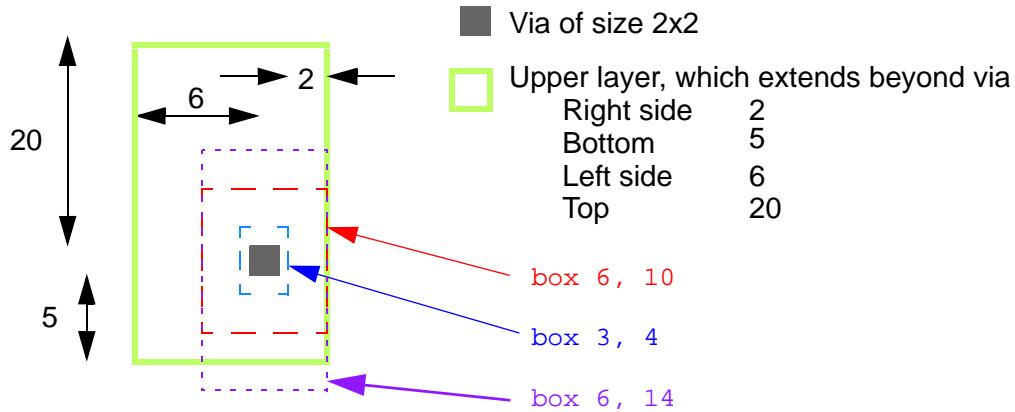
#### Example of Coverage Analysis

The following command provides one table for vias of size 2 x 2. The structure in [Figure 16-20](#) illustrates how this table might be applied.

```
RPV_VS_SI_COVERAGE {
 VIA_SIZE { 2 2 }
 COV_X { 3 6 8 }
 COV_Y { 4 10 14 }
 VALUES { 18 20 25 21 26 30 27 32 35 }
}
```

The table defines nine coverage boxes based on the `COV_X` and `COV_Y` values. The RPV values in the `VALUES` list correspond to the X,Y pairs in this order: (3,4) (3,10) (3,14) (6,4) (6,10) (6,14) (8,4) (8,10) (8,14).

In [Figure 16-20](#), the via is positioned near the lower right corner of the upper level feature. The coverage boxes with X dimension 6 exactly match the nearest X edge of the upper layer feature, therefore boxes with other X dimensions are eliminated from consideration.

**Figure 16-20 Coverage Box Usage**

The nearest Y edge of the upper layer feature falls between the edges of the (6,10) and (6,14) coverage boxes. Linear interpolation is used between the RPV values of these two coverage boxes to calculate the RPV value for this case.

If the (6,14) coverage box were not included in the table, the RPV value of the (6,10) coverage box would be used. In other words, no extrapolation is performed.

### Example With Multiple Tables

The following command provides tables for two via sizes:

```
RPV_VS_SI_COVERAGE {
 VIA_SIZE { 0.032 0.032 }
 COV_X { 0.064, 0.096, 0.2 }
 COV_Y { 0.064, 0.096, 0.2 }
 VALUES { 18 20 25 21 26 30 27 32 35 }

 VIA_SIZE { 0.064 0.64 }
 COV_X { 0.064, 0.096, 0.2 }
 COV_Y { 0.064, 0.096, 0.2 }
 VALUES { 6 8 13 9 14 18 15 20 23 }
}
```

### See Also

- [RPV\\_VS\\_AREA](#)
- [RPV\\_VS\\_SI\\_COVERAGE](#)

---

## RPV\_VS\_SI\_COVERAGE

Specifies resistance per via (RPV) values with respect to via size and upper layer coverage, taking etch effects into account for the upper layer. Valid within a `VIA` block.

### Syntax

```
RPV_VS_SI_COVERAGE UPPER_LAYER_ONLY {
 VIA_SIZE {sx1 sy1}
 COV_X {cx1, cx2, ... cxm}
 COV_Y {cy1, cy2, ... cyn}
 VALUES {r(cx1,cy1) r(cx1,cy2) ... r(cx1,cyn)
 r(cx2,cy1) r(cx2,cy2) ... r(cx2,cyn)
 ...
 r(cxm,cy1) r(cxm,cy2) ... r(cxm,cyn)}
 }
 [VIA_SIZE {sx2 sy2}]
 COV_X {dx1, dx2, ... dxm}
 COV_Y {dy1, dy2, ... dyn}
 VALUES {r(dx1,dy1) r(dx1,dy2) ... r(dx1,dyn)
 r(dx2,dy1) r(dx2,dy2) ... r(dx2,dyn)
 ...
 r(dxm,dy1) r(dxm,dy2) ... r(dxm,dyn)}
 }
 ...
}
```

### Arguments

| Argument         | Description                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------|
| UPPER_LAYER_ONLY | Required keyword that restricts the coverage check to the upper layer                                         |
| sx1 sy1          | Via size in the x- and y-dimensions (as-drawn dimensions)<br>Units: microns                                   |
| cx1, cx2, ...    | X-direction coverage values, in ascending order; coverage values for different via sizes do not have to match |
| dx1, dx2, ...    | Units: microns                                                                                                |
| cy1, cy2, ...    | Y-direction coverage values, in ascending order; coverage values for different via sizes do not have to match |
| dy1, dy2, ...    | Units: microns                                                                                                |
| r(cx1,cy1)       | Resistance per via (RPV) for the corresponding coverage values                                                |
| r(dx1,dy1)       | Units: ohms                                                                                                   |

## Description

The `RPV_VS_SI_COVERAGE` command specifies resistance per via (RPV) values based on the via size and the amount of upper conductor layer coverage.

Note:

The `RPV_VS_SI_COVERAGE` command uses post-etch dimensions for the upper layer coverage features. You can use as-drawn dimensions by using the `RPV_VS_COVERAGE` command instead.

The `RPV_VS_SI_COVERAGE` command contains one or more tables of RPV values. Each table applies to a specific via size based on the drawn dimensions of the via. The via dimensions must match exactly; however, the x- and y-dimensions can be rotated by 90 degrees. When rotation is required to match the via size, the table indexes and values are also rotated.

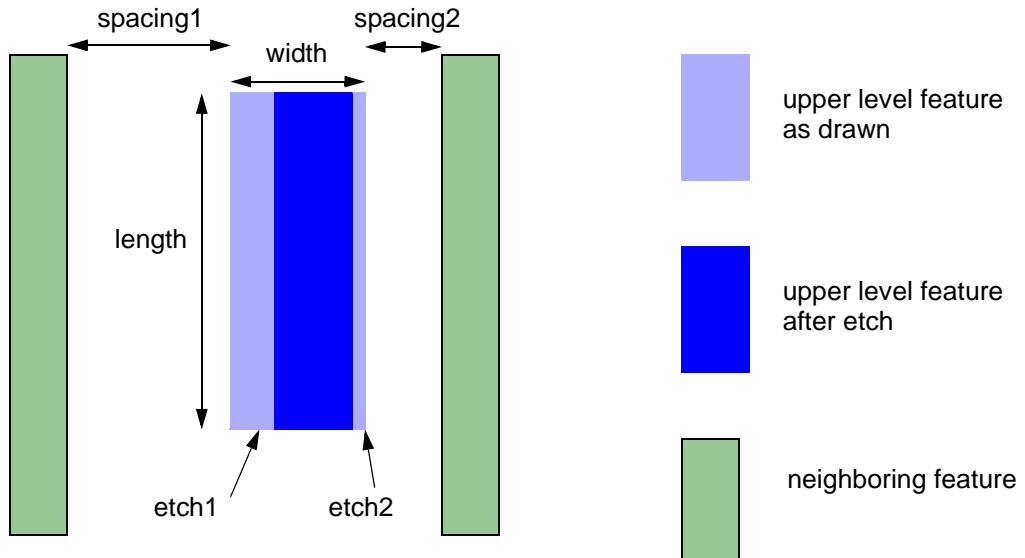
If a via does not match any of the via sizes specified in the `RPV_VS_SI_COVERAGE` command, the StarRC tool uses the `RPV_VS_AREA` command to determine the via resistance.

If a via matches a via size specified by a `VIA_SIZE` keyword in the `RPV_VS_SI_COVERAGE` command, the tool calculates the via resistance according to the following procedure.

1. Apply etch values to the upper layer feature that covers the via.

[Figure 16-21](#) illustrates the etch effect. The upper level feature covering the via is shown in light blue for the as-drawn dimensions and dark blue for the post-etch dimensions. A neighboring feature is present on each side, with different spacings on the two sides. If the `ETCH_VS_WIDTH_AND_SPACING` command is used to specify the etch conditions, the amount of etch applied to the upper level feature might be different on each side based on the spacing to the closest neighboring feature.

As a simplification, the StarRC tool calculates the average of the actual etch values on the two sides of the upper layer feature and applies the average etch to both sides.

**Figure 16-21 Example of Feature Etch**

## 2. Evaluate the available coverage boxes.

The `COV_X` and `COV_Y` values in the `RPV_VS_SI_COVERAGE` command define a set of coverage boxes with those X and Y dimensions. (The number of available boxes is the product of the number of `COV_X` values and the number of `COV_Y` values.)

The StarRC tool determines the largest coverage box that, when centered over the via, fits within the etched upper layer feature.

## 3. Look up or calculate the resistance per via.

The possible configurations are as follows, assuming that the coverage box is centered over the via.

- If all of the coverage boxes are too large to fit within the etched upper layer feature, the tool uses the RPV value of the smallest coverage box.
- If all of the coverage boxes fit within the upper layer feature, but the upper layer extends beyond the largest coverage box in both X and Y dimensions, the tool uses the RPV value of the largest coverage box.
- If an edge of the upper layer feature falls between the edges of two coverage boxes, linear interpolation is used to calculate the RPV value. Interpolation in both the X and Y dimensions might be necessary.

## See Also

- [RPV\\_VS\\_AREA](#)
- [RPV\\_VS\\_COVERAGE](#)

---

## RPV\_VS\_SI\_WIDTH\_AND\_LENGTH

Models the resistivity of a trench contact virtual via according to the silicon width and length of one of the associated trench contact conductor layers. Valid within a `VIA` block.

### Syntax

```
ETCH_ASSOCIATED_LAYER = tc_layer
RPV_VS_SI_WIDTH_AND_LENGTH {
 LENGTHS { l1 l2 l3 ... lm }
 WIDTHS { w1 w2 w3 ... wn }
 VALUES { r(l1,w1) r(l2,w1) r(l3,w1) ... r(lm,w1)
 r(l1,w2) r(l2,w2) r(l3,w2) ... r(lm,w2)
 ...
 r(l1,wn) r(l2,wn) r(l3,wn) ... r(lm,wn)
 }
}
```

### Arguments

| Argument                             | Description                                                                                                                                                                                |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>tc_layer</code>                | Either the <code>FROM</code> or <code>TO</code> layer in the <code>VIA</code> definition; must be a trench contact layer defined with the <code>LAYER_TYPE=TRENCH_CONTACT</code> statement |
| <code>l1, l2, ...</code>             | Via lengths, in ascending order<br>Units: microns                                                                                                                                          |
| <code>w1, w2, ...</code>             | Via widths, in ascending order<br>Units: microns                                                                                                                                           |
| <code>r(l1,w1), r(l2,w1), ...</code> | Resistance per via<br>Units: ohms                                                                                                                                                          |

### Description

In trench contact layers, electrical current flows both along the routing direction and also in the vertical direction. As a result, the contact resistance behavior can be complex.

You can model the trench contact resistance as follows:

- Use the `VIA` statement to define a virtual via between two trench contact layers or between a trench contact layer and another conductor layer.
- Specify the variation of via resistance with respect to the *drawn* length and width dimensions by using the `RPV_VS_WIDTH_AND_LENGTH` statement in the via definition.

- Specify the variation of via resistance with respect to the *silicon* length and width dimensions (in other words, the dimensions after applying etch operations) by using the `RPV_VS_SI_WIDTH_AND_LENGTH` statement in the via definition.

The resistance is calculated as follows:

- If the length and width of the trench contact via fall within the bounds of the values in the `LENGTHS` and `WIDTHS` argument lists, linear interpolation is applied between the nearest values.
- If the length is smaller than the minimum value in the `LENGTHS` argument list, the StarRC tool finds the resistance value that corresponds to the minimum value in the `LENGTHS` argument list and scales the resistance in proportion to the minimum length divided by the actual length. Width values smaller than the minimum value in the `WIDTHS` argument list are handled in the same way. In other words,

$$R_{\text{new}} = R_{\min} \cdot \frac{W_{\min} \cdot L_{\min}}{W_{\text{actual}} \cdot L_{\text{actual}}}$$

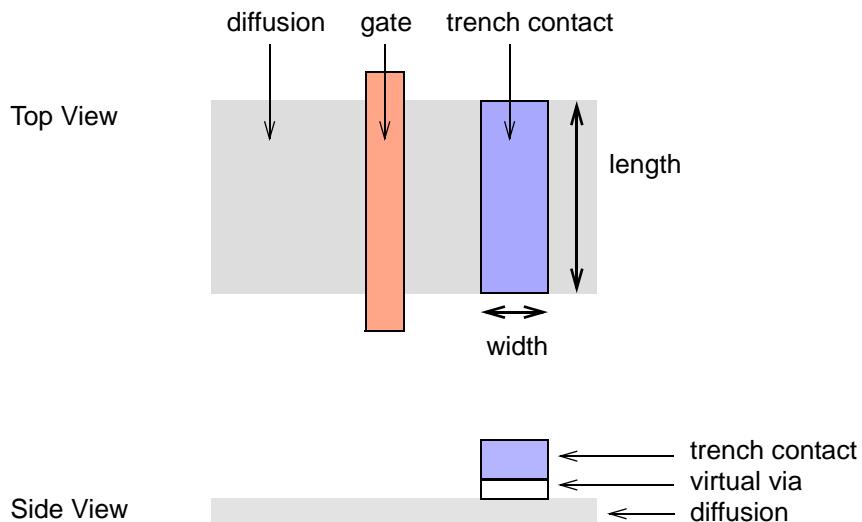
- If the length is larger than the maximum value in the `LENGTHS` argument list, the StarRC tool finds the resistance value that corresponds to the maximum value in the `LENGTHS` argument list and scales the resistance in proportion to the maximum length divided by the actual length. Width values larger than the maximum value in the `WIDTHS` argument list are handled in the same way. In other words,

$$R_{\text{new}} = R_{\max} \cdot \frac{W_{\max} \cdot L_{\max}}{W_{\text{actual}} \cdot L_{\text{actual}}}$$

RPV values represent the resistance of the entire virtual via regardless of any segmentation specified by the `TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO` command in the StarRC command file.

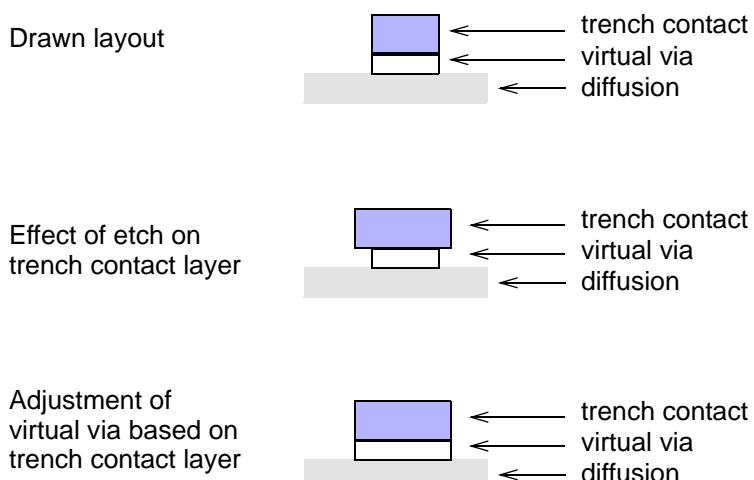
**Figure 16-22** shows a trench contact layer and the virtual via that connects it to the diffusion layer. The length dimension of the trench contact is parallel to the adjacent transistor gate polygon.

*Figure 16-22 As-Drawn Trench Contact Virtual Via*



**Figure 16-23** shows how a trench contact conductor layer might change after an etch operation. For accurate modeling, the trench contact virtual via dimensions must match the post-etch (silicon) dimensions of the `FROM` or `TO` layers associated with the via. The `ETCH_ASSOCIATED_LAYER` keyword specifies which of the two layers the virtual via dimensions should match.

*Figure 16-23 Post-Silicon Trench Contact Virtual Via*



To use the drawn dimensions of the virtual via instead of the silicon dimensions, use the `RPV_VS_WIDTH_AND_LENGTH` statement in the via definition instead of the `RPV_VS_SI_WIDTH_AND_LENGTH` statement.

### Example

In the following example, the trench contact virtual via VTC is defined between layers POD and M0. The via dimensions are set to match the dimensions of layer M0 after the application of the etch operation defined by the `ETCH_VS_WIDTH_AND_SPACING` command.

```
CONDUCTOR M0 {THICKNESS=0.9 WMIN=0.09 SMIN=0.09 LAYER_TYPE=TRENCH_CONTACT
 ETCH_VS_WIDTH_AND_SPACING { ... }
}
VIA VTC { FROM=POD TO=M0
 ETCH_ASSOCIATED_LAYER = M0
 RPV_VS_SI_WIDTH_AND_LENGTH {
 LENGTHS { 0.03 0.04 0.08 0.12 0.6 }
 WIDTHS { 0.02 0.03 0.04 }
 VALUES {
 360 160 80 60 22
 260 90 80 50 21
 200 80 70 40 20
 }
 }
}
```

### See Also

- [VIA](#)
- [RPV\\_VS\\_WIDTH\\_AND\\_LENGTH](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [TRENCH\\_CONTACT\\_VIRTUAL\\_VIA\\_SEGMENTATION\\_RATIO](#)

---

## RPV\_VS\_WIDTH\_AND\_LENGTH

Models the resistivity of a trench contact virtual via according to the drawn width and length of one of the associated trench contact conductor layers. Valid within a `VIA` block.

### Syntax

```
RPV_VS_WIDTH_AND_LENGTH {
 LENGTHS { l1 l2 l3 ... lm }
 WIDTHS { w1 w2 w3 ... wn }
 VALUES { r(l1,w1) r(l2,w1) r(l3,w1) ... r(lm,w1)
 r(l1,w2) r(l2,w2) r(l3,w2) ... r(lm,w2)
 ...
 r(l1,wn) r(l2,wn) r(l3,wn) ... r(lm,wn)
 }
}
```

### Arguments

| Argument                             | Description                                       |
|--------------------------------------|---------------------------------------------------|
| <code>l1, l2, ...</code>             | Via lengths, in ascending order<br>Units: microns |
| <code>w1, w2, ...</code>             | Via widths, in ascending order<br>Units: microns  |
| <code>r(l1,w1), r(l2,w1), ...</code> | Resistance per via<br>Units: ohms                 |

### Description

In trench contact layers, electrical current flows both along the routing direction and in the vertical direction. You can model the vertical current flow as follows:

- Use the `VIA` statement to define a virtual via between two trench contact layers or between a trench contact layer and another conductor layer.
- Specify the variation of via resistance with respect to the *drawn* length and width dimensions by using the `RPV_VS_WIDTH_AND_LENGTH` statement in the via definition.
- Specify the variation of via resistance with respect to the *silicon* length and width dimensions (in other words, the dimensions after applying etch operations) by using the `RPV_VS_SI_WIDTH_AND_LENGTH` statement in the via definition.

The resistance is calculated as follows:

- If the length and width of the trench contact via fall within the bounds of the values in the LENGTHS and WIDTHS argument lists, linear interpolation is applied between the nearest values.
- If the length is smaller than the minimum value in the LENGTHS argument list and the width is smaller than the minimum value in the WIDTHS argument list, the StarRC tool finds the resistance that corresponds to the minimum length and width values and scales the resistance in proportion to the minimum dimensions divided by the actual dimensions. In other words,

$$R_{\text{new}} = R_{\text{min}} \cdot \frac{W_{\text{min}} \cdot L_{\text{min}}}{W_{\text{actual}} \cdot L_{\text{actual}}}$$

- If the length is larger than the maximum value in the LENGTHS argument list and the width is larger than the maximum value in the WIDTHS argument list, the StarRC tool finds the resistance that corresponds to the maximum length and width values and scales the resistance in proportion to the maximum dimensions divided by the actual dimensions. In other words,

$$R_{\text{new}} = R_{\text{max}} \cdot \frac{W_{\text{max}} \cdot L_{\text{max}}}{W_{\text{actual}} \cdot L_{\text{actual}}}$$

- If only one dimension (width or length) is out of bounds, the StarRC tool performs similar scaling for the out-of-bounds dimension while using linear interpolation for the other dimension. For example, if the width is smaller than the minimum specified width, but the length falls within the specified range, the resistance is calculated as

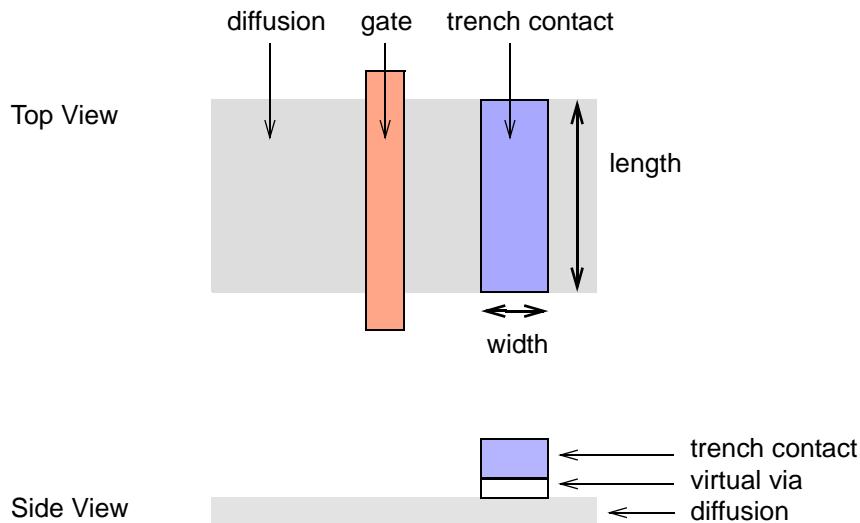
$$R_{\text{new}} = R(W_{\text{min}}, L) \cdot \frac{W_{\text{min}}}{W_{\text{actual}}}$$

RPV values represent the resistance of the entire virtual via regardless of any segmentation specified by the TRENCH\_CONTACT\_VIRTUAL\_VIA\_SEGMENTATION\_RATIO command in the StarRC command file.

The resistance of the virtual via is dependent on its length and width. If the top conductor layer is affected by etch operations, the virtual via size should also change to match the final silicon dimensions of the conductor layer. To use the silicon dimensions instead of the drawn dimensions, use the RPV\_VS\_SI\_WIDTH\_AND\_LENGTH statement instead of the RPV\_VS\_WIDTH\_AND\_LENGTH statement.

**Figure 16-24** shows a trench contact layer and the virtual via that connects it to the diffusion layer. The length dimension of the trench contact is parallel to the adjacent transistor gate polygon.

*Figure 16-24 As-Drawn Trench Contact Virtual Via*



## Example

In the following example, the trench contact virtual via VTC is defined between layers POD and M0. The via dimensions are the as-drawn dimensions and are not changed by any etch operations on the associated conductors.

```
VIA VTC { FROM=POD TO=M0 }
RPV_VS_WIDTH_AND_LENGTH {
 LENGTHS { 0.02 0.04 0.05 }
 WIDTHS { 0.01 }
 VALUES { 100 80 60 }
}
```

## See Also

- [VIA](#)
- [RPV\\_VS\\_SI\\_WIDTH\\_AND\\_LENGTH](#)
- [TRENCH\\_CONTACT\\_VIRTUAL\\_VIA\\_SEGMENTATION\\_RATIO](#)

## SIDE\_TANGENT

Specifies a conductor sidewall angle in terms of the tangent of the angular shift from vertical of the conductor sides. Valid within a CONDUCTOR block.

### Syntax

`SIDE_TANGENT = tan_value`

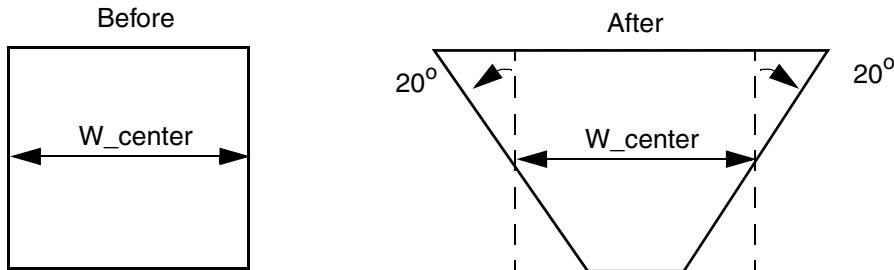
### Arguments

| Argument               | Description                                        |
|------------------------|----------------------------------------------------|
| <code>tan_value</code> | Tangent value<br>Units: none (ratio)<br>Default: 0 |

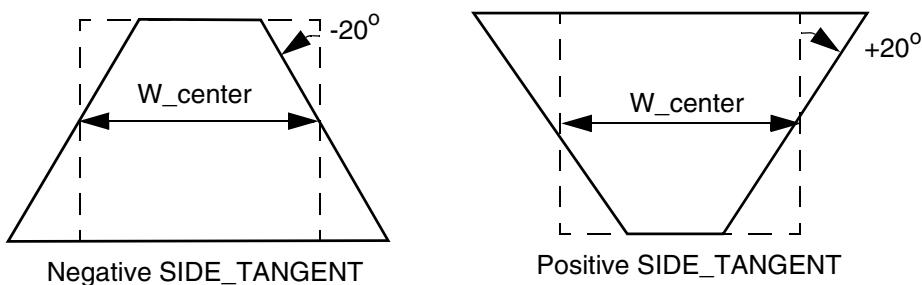
### Description

The SIDE\_TANGENT option specifies the tangent of an angular shift from vertical of the conductor sides, as shown in [Figure 16-25](#) and [Figure 16-26](#).

*Figure 16-25 Effect of the SIDE\_TANGENT Option*



*Figure 16-26 Positive Versus Negative SIDE\_TANGENT Values*



If you specify a positive value for SIDE\_TANGENT, a trapezoid is produced with a top width larger than the center width. If W is the width of the conductor without any modification, and t is the layer thickness, the top and bottom dimensions change as follows:

$$\begin{aligned}W_{\text{top}} &= W + (\text{SIDE\_TANGENT} * t) \\W_{\text{bottom}} &= W - (\text{SIDE\_TANGENT} * t)\end{aligned}$$

The center width and cross-sectional area are not affected by the adjustment; therefore resistance is not affected. However, capacitance is affected due to the shape changes at the top and bottom surfaces.

### Example

The following example specifies a 20-degree angular shift ( $\tan 20^\circ = 0.364$ ):

```
CONDUCTOR met4 {
 SIDE_TANGENT=0.364 THICKNESS=0.66 WMIN=0.15
 SMIN=0.15 RPSQ=0.078
}
```

### See Also

- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [SIDE\\_TANGENT\\_VS\\_SI\\_WIDTH\\_AND\\_CCO\\_SPACING](#)
- [SIDE\\_TANGENT\\_VS\\_SI\\_WIDTH\\_AND\\_SPACING](#)

---

## SIDE\_TANGENT\_VS\_SI\_WIDTH\_AND\_CCO\_SPACING

Specifies the side tangent as a function of the device type of the gate, width of the conductor, and spacing from neighboring polygons.

### Syntax

```
SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING {
 NUMBER_OF_TABLES = num_tables
 device_type_1 {
 CONTACT_TO_GATE_SPACINGS { s1 s2 s3 }
 WIDTHS { w1 w2 w3 }
 VALUES {
 V(s1,w1) V(s2,w1) V(S3,w1)
 V(s1,w2) V(s2,w2) V(S3,w2)
 V(s1,w3) V(s2,w3) V(S3,w3)
 }
 }
 ...
 device_type_n {
 CONTACT_TO_GATE_SPACINGS { s1 s2 s3 }
 WIDTHS { w1 w2 w3 }
 VALUES {
 V(s1,w1) V(s2,w1) V(S3,w1)
 V(s1,w2) V(s2,w2) V(S3,w2)
 V(s1,w3) V(s2,w3) V(S3,w3)
 }
 }
}
```

### Arguments

| Argument                     | Description                           |
|------------------------------|---------------------------------------|
| <i>num_tables</i>            | Number of tables                      |
| <i>device_type</i>           | Device type name                      |
| <i>s1 s2 s3</i>              | Spacing values<br>Units: microns      |
| <i>w1 w2 w3</i>              | Width values<br>Units: microns        |
| <i>V(s1,w1) V(S2,w1) ...</i> | Side tangent values<br>Units: microns |

## Description

The `SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING` table specifies the side tangent as a function of the device type of the gate, width of the conductor, and spacing from neighboring gate polygons.

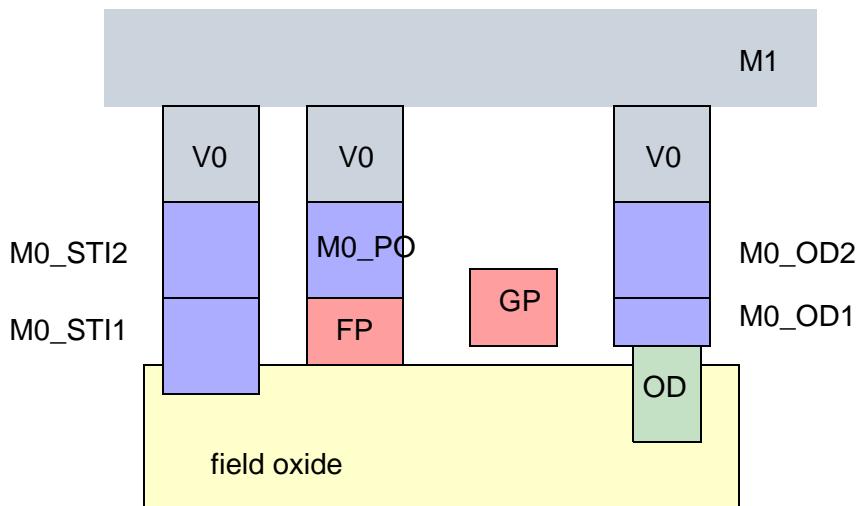
The numbers in the `VALUES` field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

## Example

[Figure 16-27](#) illustrates the difference between the two side tangent commands. If polygons on conductor layer `M0_OD1` have a `SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING` table, the StarRC tool applies a side tangent only to the edges facing features on the gate polysilicon (GP) layer. The side tangent value depends on the device type of the gate, the width of the polygon on layer `M0_OD1`, and the spacing between the `M0_OD1` polygon and the facing gate.

By contrast, if polygons on the other `M0` layers such as `M0_STI1` and `M0_PO` have a `SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING` table, the tool applies the side tangent to all four edges. The side tangent depends on the width of the polygon itself and the spacing between the polygon and neighbor polygons on layers with the same level.

*Figure 16-27 Side Tangent as a Function of Width and Spacing*



The example shown in [Figure 16-27](#) is modeled by the ITF statements in the following example.

```

CONDUCTOR M0xxxx {
 SIDE_TANGENT = st
 SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING {
 SPACINGS { 1.8 3.6 5.4 7.2 }
 WIDTHS { 1.8000 2.7000 5.4000 10.8000 13.5000 }
 VALUES {
 0.020 0.0230 0.0260 0.0290
 0.020 0.0230 0.0260 0.0290
 0.020 0.0230 0.0260 0.0290
 0.020 0.0230 0.0260 0.0290
 0.020 0.0230 0.0260 0.0290
 }
 }
 SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING {
 NUMBER_OF_TABLES=2
 G_1D5VIO_NMOS {
 CONTACT_TO_GATE_SPACINGS { s1 s2 s3 }
 WIDTHS { w1 w2 w3 }
 VALUES {
 V(s1,w1) V(s2,w1) V(s3,w1)
 V(s1,w2) V(s2,w2) V(s3,w2)
 V(s1,w3) V(s2,w3) V(s3,w3)
 }
 }
 G_CORE_NMOS {
 CONTACT_TO_GATE_SPACINGS { s1 s2 s3 }
 WIDTHS { w1 w2 w3 }
 VALUES {
 V(s1,w1) V(s2,w1) V(s3,w1)
 V(s1,w2) V(s2,w2) V(s3,w2)
 V(s1,w3) V(s2,w3) V(s3,w3)
 }
 }
 }
}

```

## See Also

- [SIDE\\_TANGENT](#)
- [SIDE\\_TANGENT\\_VS\\_SI\\_WIDTH\\_AND\\_SPACING](#)

---

## SIDE\_TANGENT\_VS\_SI\_WIDTH\_AND\_SPACING

Specifies the side tangent as a function of the width of the conductor and spacing from neighboring polygons on layers with the same PIC levels as the conductor.

### Syntax

```
SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING {
 SPACINGS { s1 s2 s3 }
 WIDTHS { w1 w2 w3 }
 VALUES {
 V(s1,w1) V(s2,w1) V(s3,w1)
 V(s1,w2) V(s2,w2) V(s3,w2)
 V(s1,w3) V(s2,w3) V(s3,w3)
 }
}
```

### Arguments

| Argument              | Description                           |
|-----------------------|---------------------------------------|
| s1 s2 s3              | Spacing values<br>Units: microns      |
| w1 w2 w3              | Width values<br>Units: microns        |
| V(s1,w1) V(s2,w1) ... | Side tangent values<br>Units: microns |

### Description

The SIDE\_TANGENT\_VS\_SI\_WIDTH\_AND\_SPACING option specifies the side tangent as a function of the width of the conductor and spacing from neighboring polygons. Specify this table within a CONDUCTOR statement.

The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

### See Also

- [SIDE\\_TANGENT](#)
- [SIDE\\_TANGENT\\_VS\\_SI\\_WIDTH\\_AND\\_CCO\\_SPACING](#)

---

## SMIN

Specifies the minimum spacing between two geometries within a CONDUCTOR block.

### Syntax

`SMIN = spacing_value`

### Arguments

| Argument                          | Description                             |
|-----------------------------------|-----------------------------------------|
| <code><i>spacing_value</i></code> | Minimum spacing value<br>Units: microns |

### Description

The `SMIN` option specifies the minimum spacing between two features within a CONDUCTOR block. The `SMIN` value is the minimum layout dimension and should not be based on any processing steps (such as etches).

If you change the `HALF_NODE_SCALE_FACTOR` option, you must change the `SMIN` value accordingly. The StarRC tool does not modify the `SMIN` value automatically.

For example, assume the `SMIN` value is 0.30 for a process in which the `HALF_NODE_SCALE_FACTOR` value is 1.0 (the default). If you set the `HALF_NODE_SCALE_FACTOR` value to 0.9, you should change the `SMIN` value to 0.27, which is calculated by applying the scale factor to the old `SMIN` value.

### Example

```
CONDUCTOR m1 {
 THICKNESS=1.00 WMIN=0.13 SMIN=0.15 RPSQ=0.015
}
```

### See Also

- [GATE\\_TO\\_CONTACT\\_SMIN](#)
- [REPORT\\_SMIN\\_VIOLATION](#)
- [HALF\\_NODE\\_SCALE\\_FACTOR](#)
- [WMIN](#)

---

## SPACER\_ER\_VS\_WIDTH\_AND\_SPACING

Specifies relative permittivity values as a function of gate width and gate-to-contact spacing for tuning of coupling capacitance. Valid within a `DIELECTRIC` block for conformal dielectrics.

### Syntax

```
SPACER_ER_VS_WIDTH_AND_SPACING {
 SPACINGS { s1 s2 ... sm }
 WIDTHS { w1 w2 ... wn }
 VALUES { er(s1,w1) er(s2,w1) ... er(sm,w1)
 er(s1,w2) er(s2,w2) ... er(sm,w2)
 ...
 er(s1,wn) er(s2,wn) ... er(sm,wn)
}
}
```

### Arguments

| Argument                       | Description                                                             |
|--------------------------------|-------------------------------------------------------------------------|
| <i>s1 s2 ... sm</i>            | Gate to contact spacings specified in ascending order<br>Units: microns |
| <i>w1 w2 ... wn</i>            | Gate widths specified in ascending order<br>Units: microns              |
| <i>er(s1,w1) ... er(sm,wn)</i> | Relative permittivity for corresponding index values<br>Units: none     |

### Description

Specify the `SPACER_ER_VS_WIDTH_AND_SPACING` option within a `DIELECTRIC` block to vary the relative permittivity of conformal dielectrics. This option is most commonly used to modify coupling capacitance values rather than to model true physical effects.

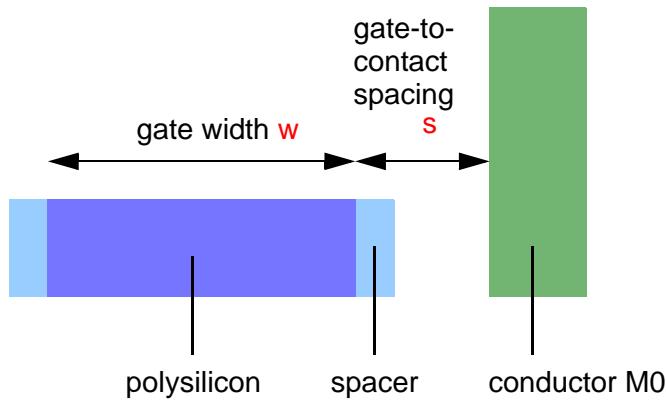
The specified gate width and gate-to-contact spacing values are used as indexes for the two-dimensional table of relative permittivity values. Each combination of length *s* and width *w* has a corresponding relative permittivity value `er(s,w)`.

This keyword can only be used in a `DIELECTRIC` block that also include the following keywords:

- An `ASSOCIATED_CONDUCTOR` keyword specifying a conductor layer whose definition includes a `LAYER_TYPE` keyword set to a value of `GATE`
- An `IS_CONFORMAL` keyword

[Figure 16-28](#) shows the gate width and gate-to-contact spacing values required for the `WIDTHS` and `SPACINGS` lists.

*Figure 16-28 Index Dimension Definitions*



### Example

In the following example, when the spacing is 0.1 and the width is 0.4, the relative permittivity is 3.1. When the spacing is 0.1 and the width is 1.2, the relative permittivity is 2.8.

```
SPACER_ER_VS_WIDTH_AND_SPACING {
 SPACINGS { 0.1 0.2 }
 WIDTHS { 0.4 1.0 1.2 }
 VALUES { 3.1 3.2 3.4 3.5 2.8 2.6 }
}
```

### See Also

- [DIELECTRIC](#)

---

## SW\_T

Defines the sidewall thickness of a conformal layer.

### Syntax

`SW_T = swt_value`

### Arguments

| Argument               | Description                                                                     |
|------------------------|---------------------------------------------------------------------------------|
| <code>swt_value</code> | The sidewall thickness<br>Units: microns<br>Default: dielectric THICKNESS value |

### Description

`SW_T=0` is allowed for conformal dielectrics. An error message is issued if `SW_T` is less than zero or is between zero and 0.005. If not specified, `THICKNESS` of the dielectric is used as `SW_T`.

### Example

```
DIELECTRIC D3 {
 THICKNESS = 0.2
 MEASURED_FROM = TOP_OF_CHIP
 SW_T = 0.15
 TW_T = 0.18
 ER = 5.9
}
```

### See Also

- [MEASURED\\_FROM](#)
- [THICKNESS](#)
- [TW\\_T](#)

---

## TECHNOLOGY

Specifies the name of the process technology for tracking and identification purposes.

### Syntax

```
TECHNOLOGY = process_name
```

### Arguments

| Argument            | Description                                                                                            |
|---------------------|--------------------------------------------------------------------------------------------------------|
| <i>process_name</i> | The process name represented by a single word that can contain alphanumeric characters and underscores |

### Description

The TECHNOLOGY statement is mandatory and should precede all other statements, but it does not need to be the first line of the ITF file.

The output of the grdgenxo tool is stored in the *process\_name.nxtgrd* file.

### Example

In the following example, the process name is `example_tech`, and the grdgenxo tool output is stored in the `example_tech.nxtgrd` file:

```
TECHNOLOGY = example_tech
```

---

## THICKNESS

Specifies the thickness of a dielectric or conductor layer.

### Syntax

THICKNESS = *layer\_thk*

### Arguments

| Argument         | Description                                                    |
|------------------|----------------------------------------------------------------|
| <i>layer_thk</i> | The thickness of the dielectric or conductor<br>Units: microns |

### Description

The dielectric or conductor thickness measured from the top of the dielectric layer below it. The reference point can be changed by setting [MEASURED\\_FROM](#). Specifying THICKNESS=0 is allowed for a conformal dielectric layer; for planar layers THICKNESS should not be specified as 0.

Specify this option within a [CONDUCTOR](#) or [DIELECTRIC](#) statement.

### Errors

An error message is issued if THICKNESS is less than 0.001 micron for a conductor layer or planar dielectric layer.

### Example

```
CONDUCTOR M2 { THICKNESS=0.60 WMIN=0.55 SMIN=0.50 RPSQ=0.062 }
DIELECTRIC D2 { THICKNESS=1.20 ER=3.9 }
```

### See Also

- [MEASURED\\_FROM](#)
- [THICKNESS](#)
- [TW\\_T](#)

---

## THICKNESS\_VS\_DENSITY

Models the thickness of a conductor as a function of its feature density. Valid within a CONDUCTOR block.

### Syntax

```
THICKNESS_VS_DENSITY [RESISTIVE_ONLY | CAPACITIVE_ONLY]
 {(D1 R1) (D2 R2) (D3 R3) (D4 R4) ... }
```

### Arguments

| Argument            | Description                                                                                                                                                                                                                                       |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RESISTIVE_ONLY      | Applies thickness adjustment to resistance only                                                                                                                                                                                                   |
| CAPACITIVE_ONLY     | Applies thickness adjustment to capacitance only                                                                                                                                                                                                  |
| D1, D2, D3, D4, ... | Density value, the fraction of the box area occupied by the feature; a value between 0 and 1                                                                                                                                                      |
| R1, R2, R3, R4, ... | Relative change in thickness (dT/dTnominal)<br>A negative R value indicates a decrease in thickness; positive R indicates a thickness increase. The value of R must be between -1 and 1 (not inclusive). Values close to 1 or -1 are unrealistic. |

### Description

In chemical-mechanical polishing (CMP) processes, the density of features in the vicinity of a specific feature affects the amount of polishing that the feature receives. The THICKNESS\_VS\_DENSITY option provides two methods for density-based conductor thickness computation: the single box (linear table) method or the multiple box method. The box refers to the area around the feature within which the average feature density is calculated.

The values in a density-relative thickness pair can be separated by a comma or a space.

Thickness variation affects both resistance and capacitance. However, you can use different coefficients for resistance and capacitance calculations by using the RESISTIVE\_ONLY or CAPACITANCE\_ONLY options.

The two methods are as follows:

- Single-box method (default method)

The box is square with a default size of 50 microns. To specify a different size box, use the DENSITY\_BOX\_WEIGHTING\_FACTOR option.

For a linear table model, specify a multipoint thickness variation versus the density table in the process file.

- **Multiple-Box Method**

The multiple-box method adjusts the calculated resistance or capacitance value by a specified factor, depending on the size of the box that can enclose the feature. To specify a multiple-box size and its weighting factor for effective density calculation, you must characterize the wafer in greater detail than in the single-box method. This method is preferred when the single-box method does not represent the process behavior with sufficient accuracy. The density box is a square. The maximum size of a density box is 500 microns. The maximum number of boxes is 5.

## Examples

The following example shows the single-box method:

```
CONDUCTOR metal3 {
 THICKNESS=0.5 SMIN=0.25 WMIN=0.25 RPSQ=0.01
 THICKNESS_VS_DENSITY RESISTIVE_ONLY
 {(0.1,-0.1) (0.2, 0.1)(0.3, 0.2)(0.4, 0.3)}
}
```

The following example shows the multiple-box method:

```
CONDUCTOR metal3 {
 THICKNESS = 0.5 SMIN = 0.25 WMIN=0.25 RPSQ = 0.01
 THICKNESS_VS_DENSITY{(0.1 -0.1)(0.2 0.1)(0.3 0.2)(0.4 0.3)}
 DENSITY_BOX_WEIGHTING_FACTOR {
 (10 1) (20 0.23) (30 0.29)
 (40 0.18) (50 -0.12)
 }
}
```

## See Also

- [DENSITY\\_BOX\\_WEIGHTING\\_FACTOR](#)
- [THICKNESS\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [Conductor Layer Thickness Variation](#)

---

## THICKNESS\_VS\_WIDTH\_AND\_SPACING

Models the thickness of a conductor as a function of its width and spacing.

### Syntax

```
THICKNESS_VS_WIDTH_AND_SPACING
 [RESISTIVE_ONLY | CAPACITIVE_ONLY] {
 SPACINGS { S1 S2 }
 WIDTHS (W1 W2)
 VALUES { v(S1,W1) v(S2,W1) v(S1,W2) v(S2,W2) }
 }
```

### Arguments

| Argument        | Description                                                                                                                                                                                                                                                                                                                                                      |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RESISTIVE_ONLY  | Applies thickness adjustment to resistance only                                                                                                                                                                                                                                                                                                                  |
| CAPACITIVE_ONLY | Applies thickness adjustment to capacitance only                                                                                                                                                                                                                                                                                                                 |
| SPACINGS        | Spacing values specified in ascending order<br>Units: microns                                                                                                                                                                                                                                                                                                    |
| WIDTHS          | Width of a conductor. Values are specified in ascending order<br>Units: microns                                                                                                                                                                                                                                                                                  |
| VALUES          | Values represent the relative change in thickness for a conductor. Order of the components of the values table is determined by the indexes in the SPACINGS and WIDTHS table. Entry in the VALUES table are ordered such that SPACINGS indexes change first for a given index in the WIDTHS table and then this is repeated for all indexes in the WIDTHS table. |

### Description

In this method, the variation of thickness as a function of the width of a conductor and relative spacing to the neighboring conductor is modeled. The variation is modeled by using the THICKNESS\_VS\_WIDTH\_AND\_SPACING keyword within a CONDUCTOR block.

The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

This thickness variation can be either negative or positive. It is a very local phenomenon and is independent of the density box. If specified with either single or multiple boxes, then this

thickness variation is computed independently from the density box. The effective thickness is calculated with the following equation:

$$T = T_{nom} \times (1 + RTf(Deff) + RTf(W, S) + RTf(SiW))$$

In this equation,

- $T_{nom}$  is the nominal thickness specified in the ITF file.
- $RTf(Deff)$  is the relative thickness change due to density.
- $RTf(W,S)$  is the relative change in thickness due to width and spacing.
- $RTf(SiW)$  is the relative change in thickness due to silicon width.

The resistance and capacitance is computed after the effective thickness is computed.

### **Example**

```
CONDUCTOR m1 {
 THICKNESS = 0.60 WMIN 0.25 SMIN = 0.25
 THICKNESS_VS_WIDTH_AND_SPACING {
 SPACINGS { 0.25 0.30 0.50}
 WIDTHS { 0.25 0.4 0.50}
 VALUES { 0.3 0.2 0.1
 0 -0.1 -0.20
 -0.3 -0.2 -0.1}
 }
}
```

### **See Also**

- [DENSITY\\_BOX\\_WEIGHTING\\_FACTOR](#)
- [THICKNESS\\_VS\\_DENSITY](#)
- [Conductor Layer Thickness Variation](#)

---

## TO

Specifies a conductor layer connected by the via.

### Syntax

TO = *layer*

### Arguments

| Argument     | Description                            |
|--------------|----------------------------------------|
| <i>layer</i> | The layer connected by the defined via |

### Description

The TO option specifies the upper or lower layer connected by the via. A via can connect only two conductor layers.

You specify this option within a VIA statement.

### Example

```
VIA v1 {
 FROM = m2
 TO = m3
 RPV = 40
 AREA = 0.16
}
```

### See Also

- [FROM](#)

---

## TSV

Describes a through-silicon via (TSV) layer.

### Syntax

```
TSV tsv_name {
 FROM = layer1
 TO = layer2
 RHO = rho_value
 AREA = area_value
 THICKNESS = thickness_value
 INSULATION_THICKNESS = ins_thickness_value
 INSULATION_ER = er_value
 [CRT1 = lin_coeff] [CRT2 = quad_coeff] [T0 = nominal_temp]
 CSUB_VS_SPACING { (s1,c1) (s2, c2) ... (sn,cn) }
 RSUB_VS_SPACING { (s1,r1) (s2, r2) ... (sn,rn) }
 CEFF_VS_FREQUENCY_AND_SPACING {
 SPACINGS { s1 s2 s3 ... }
 FREQUENCY { f1 f2 f3 ... }
 VALUES { v(s1 f1) v(s2 f1) v(s3 f1) ...
 v(s2 f1) v(s2 f2) v(s2 f3) ... } }
}
```

### Arguments

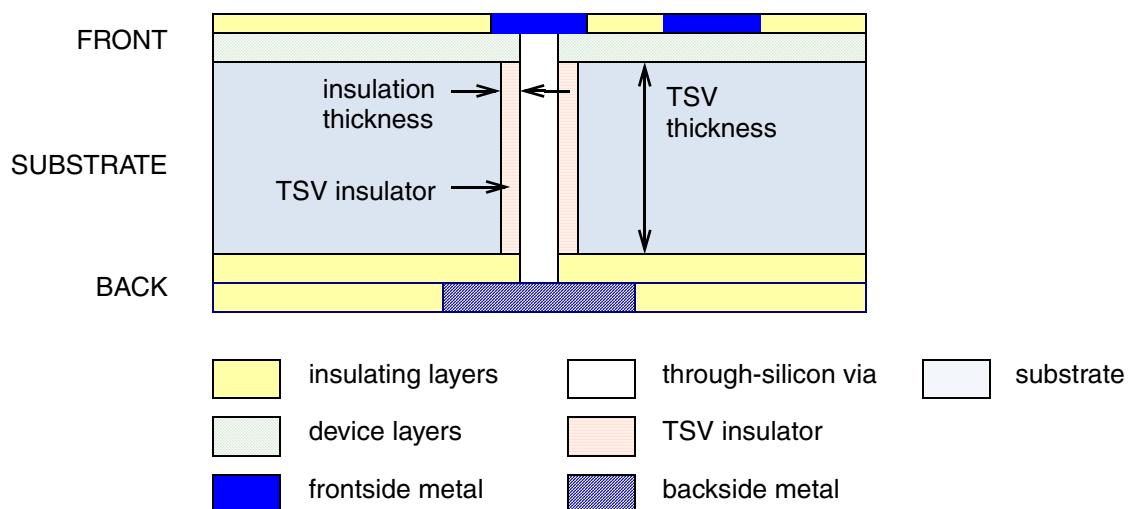
| Argument                                   | Description                                                                           |
|--------------------------------------------|---------------------------------------------------------------------------------------|
| FROM = layer1                              | Conductor layer connected by the TSV                                                  |
| TO = layer2                                | Conductor layer connected by the TSV                                                  |
| RHO = rho_value                            | Resistivity of the TSV<br>Units: ohm-microns                                          |
| AREA = area_value                          | Area of the TSV<br>Units: square microns                                              |
| THICKNESS = thickness_value                | Thickness of the TSV<br>Units: microns                                                |
| INSULATION_THICKNESS = ins_thickness_value | Thickness of the insulation layer between the TSV and the substrate<br>Units: microns |
| INSULATION_ER = er_value                   | Relative permittivity of the TSV insulation layer                                     |

| Argument                                                | Description                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CRT1 = <i>lin_coeff</i>                                 | Layer-specific linear temperature coefficient<br>Default: 0                                                                                                                                                                                                                                                    |
| CRT2 = <i>quad_coeff</i>                                | Layer-specific quadratic temperature coefficient<br>Default: 0                                                                                                                                                                                                                                                 |
| T0 = <i>nominal_temp</i>                                | Nominal temperature for the layer<br>Units: degrees Celsius<br>Default: temperature specified by GLOBAL_TEMPERATURE                                                                                                                                                                                            |
| CSUB_VS_SPACING ( <i>s<sub>n</sub>, c<sub>n</sub></i> ) | The distance and substrate-related capacitance between one through-silicon via and its neighboring through-silicon via<br>You can specify a maximum of eight spacing-capacitance pairs.<br>Units: microns                                                                                                      |
| RSUB_VS_SPACING ( <i>s<sub>n</sub>, r<sub>n</sub></i> ) | The distance and substrate-related resistance between one through-silicon via and its neighboring through-silicon via.<br>You can specify a maximum of eight spacing-resistance pairs.<br>Units: microns                                                                                                       |
| CEFF_VS_FREQUENCY_AND_SPACING                           | The effective capacitance between individual through-silicon vias. The CEFF_VS_FREQUENCY_AND_SPACING table is provided by the foundry. The effective capacitance is modeled by distance between TSV pairs and the working frequency in simulation. Use the netlist for SPICE and static timing analysis tools. |
| FREQUENCY { <i>f1 f2 f3 ...</i> }                       | The operating frequencies<br>Units: Hertz                                                                                                                                                                                                                                                                      |
| SPACINGS { <i>s1 s2 s3 ...</i> }                        | The spacing between through-silicon vias<br>Units: microns                                                                                                                                                                                                                                                     |
| VALUES { <i>c( f1,s1) ...</i> }                         | The effective capacitance between through-silicon vias<br>The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.                                                                                                               |

## Description

A through-silicon via (TSV) connects a conductor layer on the frontside of the substrate to a conductor layer on the backside, extending all the way through the substrate as shown in [Figure 16-29](#). The via is surrounded by an insulation layer to separate it from the substrate.

*Figure 16-29 Cross Section of Through-Silicon Via*



To model a through-silicon via, you must include descriptions of the frontside and backside layers in the ITF file. The specification of the frontside and backside layers follows standard ITF syntax to describe the layers in order from the outside in. In other words, describe the backside layers as if the substrate is on the bottom and the layers are on the top, even though in reality the configuration is reversed.

Place the `TSV` statement after the frontside ITF statements and before the backside ITF statements. For an example of an ITF file that describes a through-silicon via, see [Through-Silicon Via Process ITF Example](#).

For high-frequency designs, you should consider the effects of coupling of individual through-silicon vias through the substrate. Two approaches are available:

- For SPICE flows, you can generate a full RC network, as shown in [Figure 16-30](#), using the `CSUB_VS_SPACING` and `RSUB_VS_SPACING` tables provided by the foundry.
- For static timing analysis flows, you might prefer to generate a netlist that includes only capacitances between individual vias, because some analysis tools cannot accept parallel RC networks. In this case, use the `CEFF_VS_FREQUENCY_AND_SPACING` table provided by the foundry. This model uses an effective capacitance value between individual through-silicon vias, as shown in [Figure 16-31](#).

For all tables, if the frequency or spacing is outside the table boundaries, the StarRC tool uses the parameter value at the boundary. Within the table boundaries, the StarRC tool uses linear interpolation to determine the parameter value.

Figure 16-30 RC Model for SPICE Flows

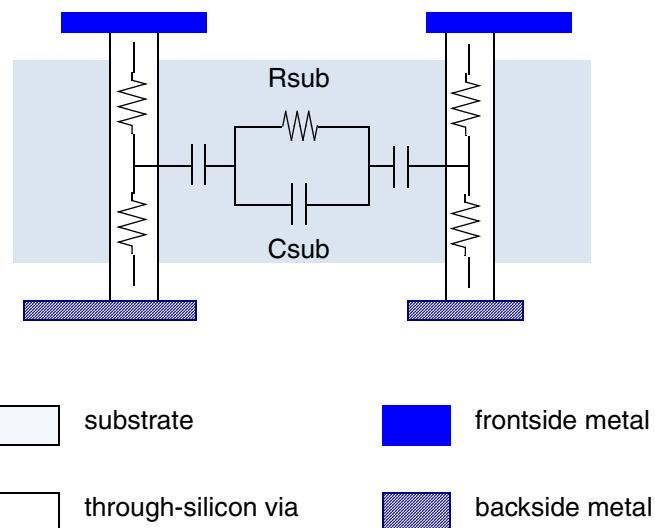
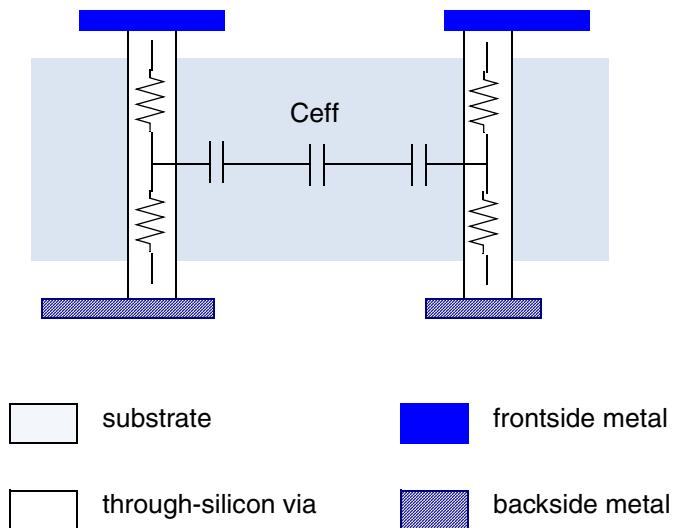


Figure 16-31 Effective Capacitance Model for Static Timing Analysis



## Example

The following example illustrates a through-silicon via definition. The frontside layers are defined before the TSV statement and the backside layers are defined after it.

```

TECHNOLOGY = tsv_process
GLOBAL_TEMPERATURE = 25.0

$ Frontside ITF statements
CONDUCTOR M1 {
 THICKNESS=0.3 WMIN=0.12 SMIN=0.12 SIDE_TANGENT=0.2
 CRT1=7.0e-03 CRT2=-8.0e-07
}
DIELECTRIC ILD_B { ER=5.5 THICKNESS=0.5 }
DIELECTRIC GATE_OXIDE { ER=4.3 THICKNESS=0.03 }
DIELECTRIC FOXIDE_A { ER=5.0 THICKNESS=0.1 }
CONDUCTOR DIFFUSION {
 THICKNESS=0.1 WMIN=0.1 SMIN=0.06
 CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
}
DIELECTRIC FOXIDE { ER=5.0 THICKNESS=0.2 }
VIA vial { FROM=M1 TO=M2 AREA=0.03 RPV=2.5 CRT1=3.0e-04 CRT2=-6.0e-06 }

$ TSV statement
TSV tsv {
 FROM=M1 TO=M1b RHO=0.05 AREA=49.0 THICKNESS=20.0
 INSULATION_THICKNESS = 0.4 INSULATION_ER = 5.5
 CRT1=7.0e-03 CRT2=-3.0e-08
}

$ Backside ITF statements
DIELECTRIC PASS { ER=9.0 THICKNESS=2.0 }
DIELECTRIC DIELb { ER=5.0 THICKNESS=1.0 }
CONDUCTOR M1b {
 THICKNESS=2.0 WMIN=4.0 SMIN=4.0 SIDE_TANGENT=-0.2
 CRT1=1.0e-03 CRT2=-7.0e-07
}
DIELECTRIC ILDB{ ER=5.2 THICKNESS=1.0 }

```

## See Also

- [3D\\_IC](#)
- [3D\\_IC\\_FILTER\\_DEVICE](#)
- [3D\\_IC\\_FLOATING\\_SUBSTRATE](#)
- [3D\\_IC\\_SUBCKT\\_FILE](#)
- [3D\\_IC\\_TSV\\_COUPLING\\_EXTRACTION](#)
- [Through-Silicon Via Extraction](#)

---

## TW\_T

Defines the topwall thickness of a conformal layer.

### Syntax

`TW_T = thickness`

### Arguments

| Argument               | Description                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| <code>thickness</code> | Topwall thickness<br>Units: microns<br>Default: thickness of the dielectric |

### Description

The `TW_T` parameter defines the topwall thickness of a conformal layer. `TW_T=0` is allowed for conformal dielectrics. An error message is issued if `TW_T` is between 0 and 0.005 microns as values below 0.005 are not allowed. If not specified, the `THICKNESS` of the dielectric is used as `TW_T`.

### Example

```
DIELECTRIC D3 {
 THICKNESS=0.2 MEASURED_FROM=TOP_OF_CHIP
 SW_T=0.15 TW_T=0.18 ER=5.9
}
```

### See Also

- [MEASURED\\_FROM](#)
- [SW\\_T](#)
- [THICKNESS](#)

---

## USER\_DEFINED\_DIFFUSION\_RESISTANCE

Specifies parameters for user-defined diffusion resistance calculation. Valid in a CONDUCTOR block that is defined as a gate layer.

### Syntax

```
USER_DEFINED_DIFFUSION_RESISTANCE {
 GATE_TO_CONT_THRESHOLD = gc_threshold
 GATE_TO_DIFF_BEND_THRESHOLD = gd_threshold
 NUMBER_OF_TABLES = no_of_tables
 <model_1> {
 RPSQ = rpsq
 RPSQ_SHARED = rpsq_sh
 [CRT1 = crt1]
 [CTR2 = crt2]
 [T0 = T0]
 SI_GATE_WIDTH_VAR = gw
 SI_GATE_LENGTH_VAR = gl
 SI_GATE_SPACER_WIDTH = gs
 SI_CONTACT_SIZE_VAR = cs
 }
 <model_2> {
 ...
}
}
```

### Arguments

| Argument                              | Description                                                                                                                                                                                                                                       |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>gc_threshold</i>                   | Distance between the gate and the contact of interest. User-defined diffusion resistance calculation is used when the layout distance is smaller than this value; standard mesh-based resistance calculation is used otherwise.<br>Units: microns |
| <i>gd_threshold</i>                   | Distance between the gate and the diffusion bend. User-defined diffusion resistance calculation is used when the layout distance is smaller than this value; standard mesh-based resistance calculation is used otherwise.<br>Units: microns      |
| <i>no_of_tables</i>                   | Integer number of tables; mandatory if greater than 1                                                                                                                                                                                             |
| <i>model_1</i> , <i>model_2</i> , ... | Model name that corresponds to the <code>diffusion_res_model</code> parameter within a <code>conducting_layers</code> block in the mapping file                                                                                                   |

---

| <b>Argument</b> | <b>Description</b>                                                                                            |
|-----------------|---------------------------------------------------------------------------------------------------------------|
| <i>rpsq</i>     | Sheet resistance of diffusion between gate and shallow trench isolation<br>Units: ohms per square             |
| <i>rpsq_sh</i>  | Sheet resistance of shared diffusion between two gates<br>Units: ohms per square                              |
| <i>crt1</i>     | Linear temperature coefficient<br>Default: 0                                                                  |
| <i>crt2</i>     | Quadratic temperature coefficient<br>Default: 0                                                               |
| <i>T0</i>       | Nominal temperature<br>Units: degrees Celsius<br>Default: temperature specified by GLOBAL_TEMPERATURE         |
| <i>gw</i>       | Difference between drawn gate width and silicon gate width<br>Units: microns                                  |
| <i>gl</i>       | Difference between drawn gate length and silicon gate length<br>Units: microns                                |
| <i>gs</i>       | Width of the spacer on each side of the gate<br>Units: microns                                                |
| <i>cs</i>       | Difference between drawn contact size and silicon contact size at the bottom of the contact<br>Units: microns |

---

## Description

For advanced processing nodes, diffusion resistance depends on factors such as contact location and diffusion layout. You can optionally take these factors into account by applying a user-defined model to specific diffusion patterns. The StarRC tool extracts layout parameters that are used as variables in a customized analysis.

### Note:

This implementation of user-defined diffusion resistance is not compatible with the feature of the same name available in earlier StarRC versions. To use this feature, you must use an nxtgrd file created in StarRC version L-2016.06 or later.

To use this analysis, perform the following steps:

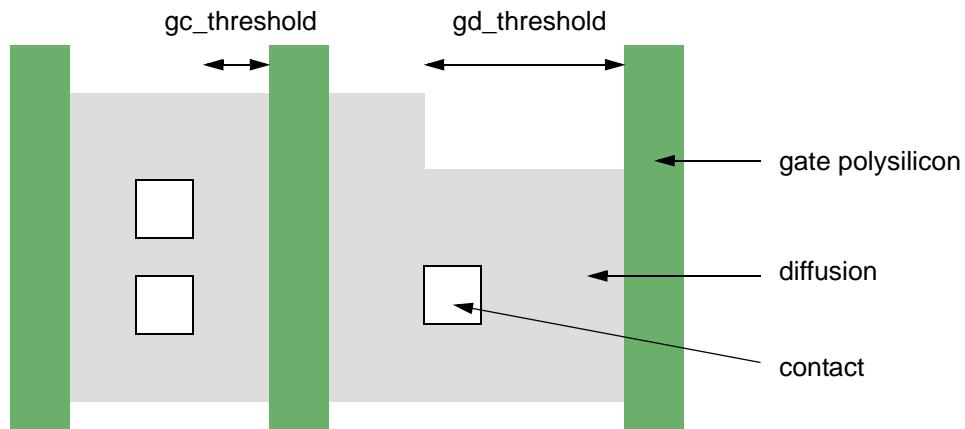
1. Set the `USER_DEFINED_DIFFUSION_RESISTANCE` command to `YES` in the StarRC command file. (This step is optional because the command defaults to `YES`.)
2. Include the `USER_DEFINED_DIFFUSION_RESISTANCE` command within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
3. Set the `LAYER_TYPE` keyword to `GATE` within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
4. Specify a model name by using the `diffusion_res_model` keyword in the `conducting_layers` command in the mapping file

Resistances calculated by the user-defined diffusion resistance method are inserted into the circuit before reduction and are included in any reduction operations. This method is used for a given contact only if the layout distances are smaller than the threshold values specified by the `GATE_TO_CONT_THRESHOLD` and `GATE_TO_DIFF_BEND_THRESHOLD` values. These parameters are illustrated in [Figure 16-32](#).

You can change the user-defined diffusion resistance parameters and update the `nxtgrd` file with the `grdgenxo -res_update` command.

Simultaneous multicorner extraction is supported. However, the `GATE_TO_CONT_THRESHOLD` and `GATE_TO_DIFF_BEND_THRESHOLD` values must be identical between corners. Temperature sensitivity analysis is not supported.

*Figure 16-32 User-Defined Diffusion Resistance Parameters*



## See Also

- [USER\\_DEFINED\\_DIFFUSION\\_RES](#)
- [conducting\\_layers](#)
- [User-Defined Diffusion Resistance](#)

---

## USE\_SI\_DENSITY

Specifies the density-based thickness variation effect.

### Syntax

USE\_SI\_DENSITY = YES | NO

### Arguments

| Argument     | Description                                                          |
|--------------|----------------------------------------------------------------------|
| YES          | Thickness variation computation is based on silicon density          |
| NO (default) | Thickness variation computation is based on drawn (database) density |

### Description

Use this optional statement when the density-based ([THICKNESS\\_VS\\_DENSITY](#) or [POLYNOMIAL\\_BASED\\_THICKNESS\\_VARIATION](#)) thickness variation effect applies to silicon density rather than drawn density.

### Example

USE\_SI\_DENSITY = YES

### See Also

- [POLYNOMIAL\\_BASED\\_THICKNESS\\_VARIATION](#)
- [THICKNESS\\_VS\\_DENSITY](#)

---

## VIA

Describes the properties of a via layer. The syntax depends on the type of via: standard via, as-drawn trench contact via, or post-etch trench contact via.

### Syntax

```
$ Standard via
VIA via_name {
 FROM = layer1
 TO = layer2
 [CRT1 = lin_coeff
 | [CRT2 = quad_coeff]
 | [CRT_VS_AREA {...}]
 | [T0 = nominal_temp]]
 RHO = rho_value
 | RPV = rpv_value AREA = area_value
 | RPV_VS_AREA {...}
 [RPV_VS_COVERAGE { ... } | RPV_SI_COVERAGE { ... }]
 [ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {...}
 | ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {...}]
}
}

$ Trench contact via using as-drawn dimensions
VIA via_name {
 FROM = layer1
 TO = layer2
 RPV_VS_WIDTH_AND_LENGTH {...}
}

$ Trench contact via using post-etch dimensions
VIA via_name {
 FROM = layer1
 TO = layer2
 ETCH_ASSOCIATED_LAYER = tc_layer
 RPV_SI_WIDTH_AND_LENGTH {...}
}
```

### Arguments

| Argument                         | Description                                                                                                                            |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| FROM = layer1                    | Upper conductor layer connected by the via                                                                                             |
| TO = layer2                      | Lower conductor layer connected by the via                                                                                             |
| ETCH_ASSOCIATED_LAYER = tc_layer | Either the FROM or TO layer in the VIA definition; must be a trench contact layer defined with the LAYER_TYPE=TRENCH_CONTACT statement |

| Argument                 | Description                                                                                                         |
|--------------------------|---------------------------------------------------------------------------------------------------------------------|
| CRT1 = <i>lin_coeff</i>  | Layer-specific linear temperature coefficient<br>Default: 0                                                         |
| CRT2 = <i>quad_coeff</i> | Layer-specific quadratic temperature coefficient<br>Default: 0                                                      |
| T0 = <i>nominal_temp</i> | Nominal temperature for the layer<br>Units: degrees Celsius<br>Default: temperature specified by GLOBAL_TEMPERATURE |
| RHO = <i>rho_value</i>   | Bulk resistivity of the via or conductor layer<br>Units: ohm-microns<br>Default: RPV x AREA                         |
| RPV = <i>rpv_value</i>   | Resistance per default via<br>Units: ohms                                                                           |
| AREA = <i>area_value</i> | Area of default via<br>Units: square microns)<br>Default: 1.0e -6 / RPV                                             |

## Description

The VIA statement describes the properties of a via layer. The syntax depends on the type of via, as follows:

- Standard via

Most vias use this format. You must specify the two layers connected by the via and the basic resistive properties of the via. In addition, you can specify tables that modify the via resistance based on size, spacing, or coverage.

- Trench contact via

Trench contact vias are artificial layers that enable the correct modeling of current in trench contact structures. At least one of the connected layers must be a trench contact layer. Specialized commands define the resistance of trench contact vias.

## Examples

The following example is a simple via definition:

```
VIA VIA1 { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

The following example defines several properties of a standard via:

```
VIA NMOS_CONTACT {
 FROM = NDIFF TO = M1
 AREA = 0.002
 RPV = 80.0
 CRT1 = 0.003 }
```

The following example is a trench contact via:

```
VIA VTC { FROM=POD TO=M0
 ETCH_ASSOCIATED_LAYER = M0
 RPV_VS_SI_WIDTH_AND_LENGTH {
 LENGTHS { 0.03 0.04 0.08 0.12 0.6 }
 WIDTHS { 0.02 0.03 0.04 }
 VALUES {
 360 160 80 60 22
 260 90 80 50 21
 200 80 70 40 20
 }
 }
}
```

---

## WMIN

Specifies the minimum width of a geometry within a CONDUCTOR block.

### Syntax

WMIN = *width\_value*

### Arguments

| Argument           | Description                           |
|--------------------|---------------------------------------|
| <i>width_value</i> | Minimum width value<br>Units: microns |

### Description

The WMIN option specifies the minimum width of a feature within a CONDUCTOR block. The WMIN value is the minimum layout dimension and should not be based on any processing steps (such as etches).

If you change the HALF\_NODE\_SCALE\_FACTOR option, you must change the WMIN value accordingly. The StarRC tool does not modify the WMIN value automatically.

For example, assume the WMIN value is 0.20 for a process in which the HALF\_NODE\_SCALE\_FACTOR value is 1.0 (the default). If you set the HALF\_NODE\_SCALE\_FACTOR value to 0.9, you should change the WMIN value to 0.18, which is calculated by applying the scale factor to the old WMIN value.

### Example

```
CONDUCTOR m1 {
 THICKNESS=1.00 WMIN=0.13 SMIN=0.15 RPSQ=0.015
}
```

### See Also

- [HALF\\_NODE\\_SCALE\\_FACTOR](#)
- [SMIN](#)

# 17

## Mapping File Commands

---

A database imported to StarRC must be accompanied by a mapping file that links each physical database layer to a process layer. This chapter describes the commands to use in a StarRC mapping file.

Although there is no restriction on the order in which these commands are listed in your file, you should specify them in the following order:

- `conducting_layers`
- `color_layers`
- `via_layers`
- `marker_layers`
- `remove_layers`
- `viewonly_layers`
- `ignore_cap_layers`

---

## conducting\_layers

Maps a conducting layer from the layout database to a conductor in the nxtgrd file.

### Syntax

```
conducting_layers
db_LyrName itf_LyrName | SUBSTRATE [precedence=prec_value]
[device_layer | capacitor_layer]
[rpsq=r_value] [model=model_name] [mask=mask_num]
[net_segment_cut_length=cut_length]
[diffusion_res_model=diff_model_name]
[net_segment_cut_length=cut_length]
[overlap_fill_spacing=new_spacing]
[table_name=cap_table]
```

### Arguments

| Argument               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>db_LyrName</i>      | Conducting layer name in the design database                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>itf_LyrName</i>     | Conducting layer name in the ITF file. Can also be SUBSTRATE                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>prec_value</i>      | Order of precedence for different design database layers mapped to the same ITF layer. Defaults to 0 if the precedence keyword is absent.                                                                                                                                                                                                                                                                                                                                 |
| <i>device_layer</i>    | Marks the layer for resistance extraction of power nets. Specify this keyword in the mapping file when you use the POWER_EXTRACT: DEVICE_LAYERS command in the command file. The <i>device_layer</i> keyword can be specified in any order in the mapping file for the conducting layers when the RPSQ and device model options are used.                                                                                                                                 |
| <i>capacitor_layer</i> | Marks the layer as a special-purpose layer for interleaved metal-finger capacitor structures. Ground capacitances are adjusted under the assumption that polygons on this layer are part of a dense regular layout over a large area. This keyword should be used only on database layers that are used exclusively for creating interleaved capacitors. Using this keyword improves accuracy for these structures but degrades results if used on other conductor types. |
| <i>r_value</i>         | Sheet resistance (in ohms per square) of the design database layer. The value specified in the mapping file overrides the RPSQ statement, RPSQ_VS_WIDTH_AND_SPACING table, or RPSQ_VS_SI_WIDTH table specified in the ITF file.                                                                                                                                                                                                                                           |

| Argument               | Description                                                                                                                                                  |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>model_name</i>      | Model name to be written into the parasitic netlist, used only if the NETLIST_PARASITIC_RESISTOR_MODEL command is set to YES in the StarRC command file      |
| <i>mask_num</i>        | The mask ID number for multimask patterning; an integer from 1 to the value of the NUMBER_OF_MASKS keyword in the ETCH_VS_WIDTH_AND_SPACING table, inclusive |
| <i>cut_length</i>      | Analysis length of segments on this layer<br>Units: microns<br>Default: 20                                                                                   |
| <i>diff_model_name</i> | Model name for user-defined diffusion resistance extraction                                                                                                  |
| <i>cut_length</i>      | Analysis length of segments on this layer<br>Units: microns<br>Default: 20                                                                                   |
| <i>new_spacing</i>     | In the metal fill reuse flow, the amount by which to separate metal fill polygons from signal nets to resolve shorts<br>Units: microns<br>Default: none      |
| <i>cap_table</i>       | Name of a gate-to-diffusion capacitance table                                                                                                                |

## Description

Specify the `conducting_layers` section to map a conducting layer from the layout database (routing layers, device terminal layers, device layers, and so on) to a conductor layer in the nxtgrd file. You can map a database layer to the substrate by using the SUBSTRATE keyword instead of the conductor layer name.

## Analysis Length

The `net_segment_cut_length` keyword specifies the analysis length to use on net segments in the specified layer. Alternatively, you can set the value globally by using the NET\_SEGMENT\_CUT\_LENGTH command in the StarRC command file. If the cut length is defined in both the StarRC command file and the mapping file, the definition in the mapping file takes precedence. These commands are valid only for transistor-level extraction.

## Layer Precedence

A design database layer can appear only one time within a `conducting_layers` section. All keywords for that layer must be included on the same line. You can map multiple design database layers to a single ITF layer by using the `precedence` keyword after the ITF layer name to define the order of precedence for resolving design overlaps. However, it is a good practice to avoid creating design overlaps.

Use the `precedence` keyword to specify a positive integer that establishes the layer's position in the vertical substrate profile. Larger numbers denote higher vertical precedence, while smaller numbers denote lower vertical precedence. It is not necessary for values to be sequential.

## User-Defined Diffusion Resistance

User-defined diffusion resistance is a method that is more accurate for advanced process nodes than the standard mesh resistance calculation.

The `diffusion_res_model` keyword has an effect only under the following conditions:

- The `USER_DEFINED_DIFFUSION_RES` command in the StarRC command file is set to `YES` (or is omitted, because it defaults to `YES`).
- The `CONDUCTOR` block in the ITF file for the gate conductor layer contains a `USER_DEFINED_DIFFUSION_RESISTANCE` command. The gate conductor layer must have its `LAYER_TYPE` keyword set to `GATE`.

If the diffusion resistance calculation is entirely embedded in the SPICE model for certain device types, do not use the `diffusion_res_model` keyword in the mapping file.

## Examples

A simple `conducting_layers` block is as follows:

```
conducting_layers
 m2 metal2
 m1 metal1
 nsd diffusion
 psd diffusion
```

The following example assigns some properties in addition to mapping the layers:

```
conducting_layers
 M2 metal2
 M1 metal1
 POLY fpoly
 nsd diffusion device_layer RPSQ=0.5
 psd diffusion device_layer
 welltie SUBSTRATE
 ngate gpoly device_layer
 pgate gpoly device_layer
```

In the following example, the metal 1 layer is split into two masks named M1\_color1 and M2\_color2. The metal 2 layer is not split; do not use the `MASK` keyword in this case.

```
conducting_layers
 M1_color1 M1 MASK=1
 M1_color2 M1 MASK=2
 M2 M2
```

The following example shows a mapping file that specifies different gate-to-diffusion capacitance tables for different layers. When the `table_name` parameter is defined for a gate layer, StarRC uses the capacitance table with same name specified in the `GATE_TO_DIFFUSION_CAP` command in the ITF file.

```
conducting_layers
 ngate gpoly table_name=NMOS
 pgate gpoly table_name=PMOS
 tngate gpoly
 tpgate gpoly
```

The following example shows a mapping file that specifies different user-defined diffusion resistance model names for different layers. To calculate resistance, the StarRC tool uses the parameters defined for that model name in the `USER_DEFINED_DIFFUSION_RESISTANCE` command in the ITF file for the gate conductor.

```
conducting_layers
 gate_n GATE diffusion_res_model=res_n
 gate_p GATE diffusion_res_model=res_p
```

## See Also

- [POWER\\_EXTRACT: DEVICE\\_LAYERS](#)
- [NET\\_SEGMENT\\_CUT\\_LENGTH](#)
- [ETCH\\_VS\\_WIDTH\\_AND\\_SPACING](#)
- [GATE\\_TO\\_DIFFUSION\\_CAP](#)
- [USER\\_DEFINED\\_DIFFUSION\\_RESISTANCE](#)
- [USER\\_DEFINED\\_DIFFUSION\\_RES](#)
- [User-Defined Diffusion Resistance](#)
- [The Metal Fill Reuse Flow](#)

---

## color\_layers

Maps the color layer and specifies the shift amount for each color layer.

### Syntax

```
color_layer
 color_LyrName grd_LyrName [x=X_shift y=Y_shift]
```

### Arguments

| Argument             | Description                                                    |
|----------------------|----------------------------------------------------------------|
| <i>color_LyrName</i> | The color layer name in the design database                    |
| <i>grd_LyrName</i>   | The ITF layer associated with the design color layer           |
| X_shift              | The shift amount in the horizontal direction<br>Units: microns |
| Y_shift              | The shift amount in the vertical direction<br>Units: microns   |

### Description

The `color_layers` command specifies the shift amount for each color layer and links each physical database color layer to a modeled layer in the `nxtgrd` file.

### Example

The following example maps the M1\_color1 layer in the design to the metal1 ITF layer. Notice that two design layer names are mapped to metal1.

```
color_layers
M1_color1 metal1 x=0.002 y=0.002
M1_color2 metal1 x=-0.002 y=-0.002
M2_color1 metal2 x=-0.002 y=0.002
M2_color2 metal2 x=0.002 y=-0.002
```

## See Also

- [DPT](#)
- [DPT\\_COLOR\\_GDS\\_FILE](#)
- [DPT\\_COLOR\\_GDS\\_LAYER\\_MAP\\_FILE](#)
- [SELECTED\\_CORNERS](#)
- [SIMULTANEOUS\\_MULTI\\_CORNER](#)
- [ER\\_VS\\_SI\\_SPACING](#)
- [Double or Multiple Patterning Technology](#)

---

## **via\_layers**

Maps a via layer in the database to a via layer in the nxtgrd file.

### **Syntax**

```
via_layers
 db_LyrName grd_LyrName [device_layer]
 [rpv=rpv_value] [area=via_area]
 [MODEL=model1_name]
 [MAX_VIA_ARRAY_LENGTH = length]
 [MAX_VIA_ARRAY_SPACING = spacing]
 [REDUCE_VIA = reduction_option]
```

### **Arguments**

---

| <b>Argument</b>         | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>database_LyrName</i> | Via layer in the design database                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>grd_LyrName</i>      | Via layer in the technology file, ITF, or nxtgrd file                                                                                                                                                                                                                                                                                                                                                                             |
| <i>device_layer</i>     | Layer for the resistance extraction of power nets. Specify this keyword when you use the POWER_EXTRACT:DEVICE_LAYERS command in the command file. The <i>device_layer</i> keyword can be specified in any order in the mapping file for the via layers when RPV, device model, or via merging options are used.                                                                                                                   |
| <i>rpv_value</i>        | Resistance per via<br>Units: ohms per via                                                                                                                                                                                                                                                                                                                                                                                         |
| <i>via_area</i>         | Area of the via<br>Units: square microns                                                                                                                                                                                                                                                                                                                                                                                          |
| <i>model1_name</i>      | String representing a model name, used to write out model names for parasitic resistors in the parasitic netlist. All via layers must be mapped to a model if you specify the NETLIST_PARASITIC_RESISTOR_MODEL command.<br><br>If you have not specified a corresponding resistor MODEL in the database, no model is printed to the parasitic netlist for that resistor and the StarRC tool issues a warning in the summary file. |

| <b>Argument</b>         | <b>Description</b>                                                                                                                                                                                                                                                                             |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>length</i>           | The length of a via array to be reduced to a single via. If you do not specify this argument, an array with via spacing less than or equal to MAX_VIA_ARRAY_SPACING is reduced to one via and eventually to one resistor.<br>Units: microns<br>Default: 20                                     |
| <i>spacing</i>          | The via-to-via spacing. This argument merges vias that lie within the specified spacing. The merging of vias is restricted by the MAX_VIA_ARRAY_LENGTH argument. Typically, you should set MAX_VIA_ARRAY_LENGTH to the DRC spacing rule for that via layer.<br>Units: microns<br>Default: none |
| <i>reduction_option</i> | The reduction specification; used only if the KEEP_VIA_NODES command is set to MAPPING_FILE<br>Valid values: yes (the default), no, power, signal                                                                                                                                              |

### Description

The `via_layers` section maps a via layer in the database to a via layer in the nxtgrd file.

You can override the via resistance value contained in the nxtgrd file to which the database is being mapped by specifying the RPV value and the via area. The layers appearing in this category can only be nxtgrd file via layers.

You can specify how reduction is applied to vias by setting the `KEEP_VIA_NODES` command to `MAPPING_FILE` in the command file and including the `REDUCE_VIA` option in the `via_layers` section in the mapping file. The reduction options are as follows:

- `yes` (reduce vias)
- `no` (do not reduce vias)
- `power` (reduce vias only if they belong to power nets)
- `signal` (reduce vias only if they belong to signal nets)

## Examples

This example provides new RPV and area values for two of the four via layers:

```
via_layers
 V2 via2
 V1 vial
 SubCont Cont rpv=10 area=0.04
 PolyCont Cont rpv=8 area=0.04
```

This example marks two via layers as device layers:

```
via_layers
 V1 vial
 POLY_CONT polyCont device_layer
 DIFF_CONT diffCont device_layer rpv=0.5
```

## See Also

- [POWER\\_EXTRACT](#): DEVICE\_LAYERS
- [RPV\\_VS\\_AREA](#)

---

## marker\_layers

Specifies marker layers.

### Syntax

```
marker_layers
 db_LyrName
```

### Arguments

| Argument          | Description                                 |
|-------------------|---------------------------------------------|
| <i>db_LyrName</i> | Name of marker layer in the design database |

### Description

Marker layers are objects optionally derived from text interactions to identify special nets that become either primary input or output ports or SKIP\_CELLS ports (instance pins) in the StarRC output parasitic netlist.

### Examples

```
marker_layers
 metall1_pin
 poly_pin
 metal17_pio
```

### See Also

- [SKIP\\_CELLS](#)

---

## remove\_layers

Specifies layers to be ignored by the extractor.

### Syntax

```
remove_layers
 db_LyrName
```

### Arguments

| Argument          | Description                       |
|-------------------|-----------------------------------|
| <i>db_LyrName</i> | Name of the design database layer |

### Description

The `remove_layers` section specifies layers to be ignored by the extractor.

#### Note:

Removing database layers can affect the connectivity of the output parasitic netlist. An extraction under typical conditions should not require the use of this mapping category.

### Examples

```
remove_layers
 ntap
 ptap
 tndiff
 tpdiff
```

### See Also

- [MAPPING\\_FILE](#)

---

## **viewonly\_layers**

Specifies view-only layers to be written to the StarRC extracted view.

### **Syntax**

```
viewonly_layers
 db_LyrName
```

### **Arguments**

| Argument          | Description                       |
|-------------------|-----------------------------------|
| <i>db_LyrName</i> | Name of the design database layer |

### **Description**

The `viewonly_layers` section specifies the view-only (nonconnectivity) layers to be written to the StarRC extracted view. The layers specified in this section are written to the extracted view in the same way as the connectivity layers.

To make the view-only layers visible in the OpenAccess (OA) view, you must also map these layers in the layer purpose pair (LPP) mapping file.

### **Examples**

The following example shows a portion of a StarRC layer mapping file:

```
remove_layers
 nwdiode25_dio
 nwdiode33_dio
viewonly_layers
 medvtp
```

The following example shows the corresponding LPP mapping file:

```
M5PIN poly drawing poly net poly subnode
M6PIN poly drawing poly net poly subnode
M7PIN poly drawing poly net poly subnode
M8PIN poly drawing poly net poly subnode
M9PIN poly drawing poly net poly subnode
POLYPIN poly drawing poly net poly subnode
medvtp medvtp drawing nil nil nil nil
```

### **See Also**

- [MAPPING\\_FILE](#)

---

## ignore\_cap\_layers

Specifies that the capacitance between certain design database layer-to-layer interactions is to be ignored.

### Syntax

```
ignore_cap_layers
 layer_1 layer_2 [layer_3 ...] [L=length]
 [layer_1 layer_2 [layer_name3 ...] [L=length]]
```

### Arguments

| Argument                      | Description                                                                                                                                                                                                                   |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>layer_1, layer_2 ...</i>   | Design database layer names                                                                                                                                                                                                   |
| <i>length</i>                 | A positive value specifying the distance over which the interlayer capacitance should be ignored<br>Units: microns                                                                                                            |
| NGATE NSD L= <i>value</i>     | The n-diffusion length for which you want to ignore the capacitance to the particular NGATE source or drain. Can be specified for only MOS transistors. If a layer pair is not part of a MOS definition, a warning is issued. |
| PGATE PSD L= <i>value</i>     | The p-diffusion length for which you want to ignore the capacitance to the particular NGATE source or drain. Can be specified for only MOS transistors. If a layer pair is not part of a MOS definition, a warning is issued. |
| nsd SUBSTRATE L= <i>value</i> | The n-diffusion to substrate length for which you want to ignore the capacitance                                                                                                                                              |
| psd SUBSTRATE L= <i>value</i> | The p-diffusion to substrate length for which you want to ignore the capacitance                                                                                                                                              |

### Description

The `ignore_cap_layers` section specifies that the capacitance between certain design database layer-to-layer interactions is to be ignored. In addition, you can specify a length value (or distance) of diffusion where capacitances are ignored.

To partially ignore capacitances for source and drain transistor areas to gate and substrate, add the optional `L` keyword with a length value . This means StarRC ignores the capacitance

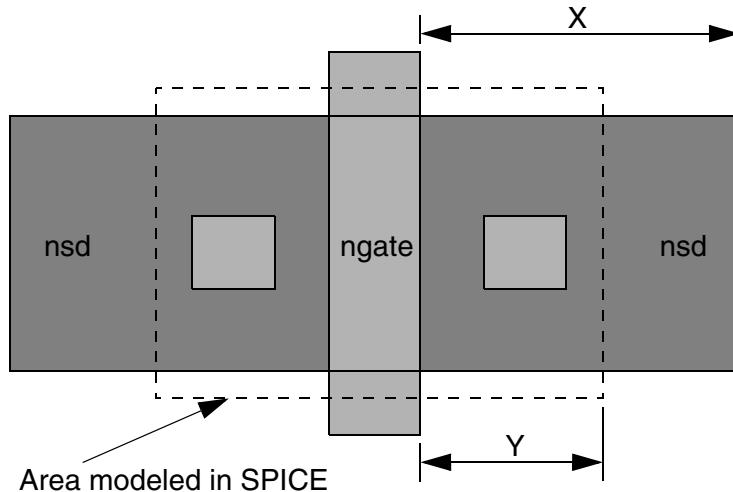
up to the specified amount. The total capacitance on the net with specified capacitance increases when an L value is specified.

- All parallel-plate and fringing capacitance components between a specified layer pair are omitted from the parasitic netlist.
- If you specify a design database layer in the ignore\_cap\_layers section of the mapping file, this function acts independently from the IGNORE\_CAPACITANCE command in the StarRC command file, regardless of the IGNORE\_CAPACITANCE command setting.
- If more than two layers are specified, all the capacitances between every possible combination of layers are ignored. To ignore the capacitance between polygons of the same layer, specify the same layer twice.

The field solver FSCOMPARE and FS\_EXTRACT\_NETS modes interpret a specified ignore\_cap\_layers section when producing field solver results for accuracy validation or netlisting.

The StarRC tool ignores all capacitance between a gate to a source or drain terminal of a metal oxide semiconductor (MOS) transistor. This is acceptable when the complete capacitance of that MOS transistor is present in the SPICE model. However, in some cases the drawn devices have larger source or drain areas than the characterized transistors, as shown in [Figure 17-1](#). Ignoring them completely during extraction is inaccurate.

*Figure 17-1 Applying Capacitance to MOS Source and Drain*

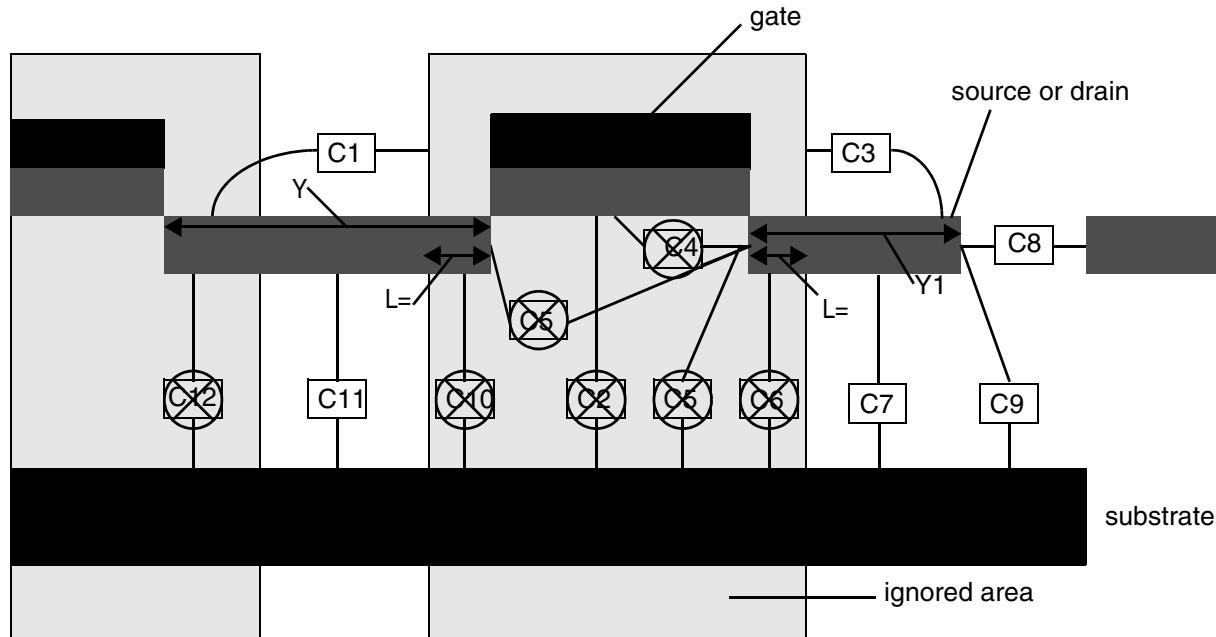


To avoid double counting the capacitance already modeled in the device, you can specify partial ignoring of gate capacitance. See [Figure 17-2](#). To do this, set a length parameter in the ignore cap section of your mapping file for the specified layer pair using the ignore\_cap\_layers section.

The conditions under which the length parameter is supported are as follows:

- Both layer1 and layer2 of the specified layer pair should be part of the MOSFET device. One layer might be the gate terminal layer or substrate while the other is the source or drain terminal layer.
- The value specified by `ignore_cap_layers` must be positive.

*Figure 17-2 Defining a Partial Distance to Ignore*



## Examples

The following shows a simple example:

```
ignore_cap_layers
 tn_diff NSD nnsd
 m1_cap_term SUBSTRATE
```

The following example specifies a length value for source and drain of the MOS transistor to partially ignore capacitance, which can cause the total capacitance on a net to increase.

```
ignore_cap_layers
 ngate NSD L=0.1
 pgate PSD L=0.1
 nsd SUBSTRATE L=0.1
 psd SUBSTRATE L=0.1
```