



IC Compiler 1 Workshop

Student Guide

20-I-071-SSG-008 2008.09

Synopsys Customer Education Services
700 East Middlefield Road
Mountain View, California 94043

Workshop Registration: **1-800-793-3448**

www.synopsys.com

Copyright Notice and Proprietary Information

Copyright © 2009 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

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.

Registered Trademarks (®)

Synopsis, AMPS, Cadabra, CATS, CRITIC, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSIM, HSPICE, iN-Phase, in-Sync, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PrimeTime, SiVL, SNUG, SolvNet, System Compiler, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, Columbia, Columbia-CE, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, Direct Silicon Access, Discovery, Encore, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, HSIMplus, HSPICE-Link, iN-Tandem, i-Virtual Stepper, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Milkyway, ModelSource, Module Compiler, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Raphael-NES, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, Taurus, TSUPREM-4, VCS Express, VCSI, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license. ARM and AMBA are registered trademarks of ARM Limited. Saber is a registered trademark of SabreMark Limited Partnership and is used under license. All other product or company names may be trademarks of their respective owners.

Document Order Number: 20-I-071-SSG-008
IC Compiler 1 Student Guide

Table of Contents

Unit i: Introduction & Overview

Facilities	i-2
Workshop Goal	i-3
Target Audience	i-4
Workshop Prerequisites	i-5
Introductions	i-6
Curriculum Flow	i-7
Agenda	i-8
Agenda	i-9
Agenda	i-10
High-Level IC Compiler Flow	i-11
Lab 0: IC Compiler GUI – MainWindow	i-12
Lab 0: IC Compiler GUI – LayoutWindow	i-13
Lab 0A: IC Compiler GUI	i-14

Unit 1: Data Setup & Basic Flow

Unit Objectives	1-2
A Word of Caution About Scripts and Flows	1-3
General IC Compiler Flow	1-4
Data Setup	1-5
Logical Libraries	1-6
Physical Reference Libraries	1-7
Milkyway Structure of Physical Libraries	1-8
Specify the Logical Libraries	1-9
Define ‘logic0’ and ‘logic1’	1-10
IC Compiler Initialization Files	1-11
Create a “Container”: The Design Library	1-12
Initial Structure of a Milkyway Design Library	1-13
The Technology File (.tf file)	1-14
Example of a Technology File	1-15
The check_library Command	1-16
Specify TLU+ Parasitic RC Model Files	1-17
Timing is Based on Cell and Net Delays	1-18
TLU+ Models	1-19
Mapping file	1-20
Read the Netlist and Create a Design CEL	1-21
Must Uniquify Multiply Instantiated Designs	1-22
Linking: Resolving References	1-23
Milkyway Design Library with Design Cell	1-24
Shortcut: Import the Netlist	1-25
Verify Logical Libraries Are Loaded	1-26
Define Logical Power/Ground Connections	1-27
Apply and Check Timing Constraints	1-28

Table of Contents

Timing Constraints.....	1-29
Ensure Proper Modeling of Clock Tree	1-30
Test for Understanding	1-31
Apply Timing and Optimization Controls	1-32
Available Timing and Optimization Controls.....	1-33
Timing and Optimization Setup Example.....	1-34
Enable Multiple Clocks per Register	1-35
Enable Constant Propagation.....	1-36
Enable Multiple Port Net Buffering.....	1-37
Enable Constant Net Buffering, if Needed	1-38
Apply Timing Derating for On-Chip Variation	1-39
Define “Don’t Use” or “Preferred” Cells.....	1-40
Keep Spare or Unloaded Cells	1-41
Apply Area Constraint for Area Recovery	1-42
Apply a Power and Area Critical Range.....	1-43
IC Compiler Organizes Paths into Groups.....	1-44
General Problem: Sub-Critical Paths Ignored.....	1-45
Serious Problem: Reg-to-Reg Paths Ignored	1-46
Solution: User-Defined Path Groups	1-47
Define Path Groups for I/O Paths, if needed	1-48
Prevent Buffering of Clock-as-Data Networks.....	1-49
Modify Optimization Priority if Needed.....	1-50
Enable Recovery and Removal Timing Arcs.....	1-51
Perform a ‘Timing Sanity Check’	1-52
Remove Unwanted “Ideal Net/Networks”.....	1-53
Save the Design.....	1-54
Design Library with New Design Cell.....	1-55
UNIX Manipulation of a Milkyway Database.....	1-56
Restoring Variables.....	1-57
Restoring Logical Library and TLU+ Settings	1-58
Loading an Existing Cell After Exiting ICC.....	1-59
Data Setup Summary	1-60
Data Setup Example (1 of 3).....	1-61
Data Setup Example (2 of 3).....	1-62
Data Setup Example (3 of 3).....	1-63
Test for Understanding (1 of 2)	1-64
Test for Understanding (2 of 2)	1-65
General IC Compiler Flow.....	1-66
Design Planning	1-67
Load an Existing Floorplan.....	1-68
Placement and Related Optimizations	1-69
Clock Tree Synthesis	1-70
Routing.....	1-71
Chip Finishing.....	1-72
Analyzing the Results (1/2)	1-73
Analyzing the Results (2/2)	1-74

Table of Contents

Example “run” Script	1-75
Basic Flow Summary	1-76
Lab 1: Design Setup and Basic Flow	1-77

Unit 2: Design Planning

Unit Objectives	2-2
General IC Compiler Flow	2-3
Terminology	2-4
ICC Design Planning and Re-Synthesis Flow	2-5
Select the Design Planning Task GUI	2-6
Create the Starting Floorplan	2-7
Create Physical-only Pad Cells	2-8
Specify Pad Cell Locations	2-9
Initialize the Floorplan	2-10
Core Area Parameters	2-11
Floorplan After Initialization	2-12
Insert Pad Filler Cells	2-13
Create P/G Pad Rings	2-14
Prior to Virtual Flat Placement	2-15
Ignore Extra Routing Layers	2-16
Constraining Macros	2-17
Manual Macro Placement	2-18
Macro Constraints: Arrays	2-19
Macro Constraints: Legal Orientation Option	2-20
Macro Constraints: Anchor Bound Option	2-21
Macro Constraints: Side Channel Option	2-22
Macro Constraints: Relative Location	2-23
Congestion Potential Around Macro Cells	2-24
Apply Global Placement Blockages	2-25
Apply Specific Placement Blockages	2-26
Summary: Create the Starting Floorplan	2-27
Test For Understanding	2-28
Perform Virtual Flat Placement	2-29
Set Placement Strategy Parameters	2-30
VF Placement with Virtual IPO (VIPO)	2-31
Perform Virtual Flat Placement	2-32
Hierarchy Aware Placement or Gravity	2-33
Summary: Virtual Flat Placement	2-34
Reduce Congestion	2-35
Is the Design Congested?	2-36
Understanding the Congestion Calculation	2-37
Congestion Guidelines	2-38
Modify Macro Placement Constraints	2-39
Apply Standard Cell Placement Constraints	2-40
Is High Cell Density Causing Congestion?	2-41

Table of Contents

Reducing Cell Density Hotspots.....	2-42
Coordinate-based Placement Blockages	2-43
Modify “FP Placement Strategy” Options.....	2-44
Placement Strategy Options and Defaults.....	2-45
Macro Placement Strategy Examples	2-46
Perform Congestion-driven Placement	2-47
Invoke the High Effort Congestion Strategy	2-48
Modify the Floorplan	2-49
“Fix” All Macro Cell Placement.....	2-50
Summary: Reduce Congestion.....	2-51
Test For Understanding (1 of 2).....	2-52
Test For Understanding (2 of 2).....	2-53
Synthesize the Power Network (PNS)	2-54
Power Network Synthesis (PNS)	2-55
Define Logical Power/Ground Connections	2-56
Apply Power Network Constraints	2-57
Synthesize and Analyze the Power Network	2-58
Modify Constraints and Re-synthesize	2-59
Create Virtual Power/Ground Pads if Needed	2-60
Add Additional P/G Pads to TDF and Re-load.....	2-61
Commit the Power Network	2-62
Connect P/G Pins and Create Power Rails	2-63
Analyze the Power Network	2-64
Apply Power Net Placement Blockages	2-65
Perform Incremental Virtual Flat Placement	2-66
Summary: Synthesize the Power Network	2-67
Reduce Delay	2-68
Global Route and Analyze Congestion.....	2-69
Modify Power Net Placement Blockages	2-70
Return to “Reduce Congestion”, if Needed	2-71
Extract Net Parasitics and Analyze Timing	2-72
Perform In-Place Optimization.....	2-73
Modify Floorplan or Re-Synthesize, if Needed	2-74
Summary: Reduce Delay	2-75
Write Out the Floorplan and DEF Files	2-76
Write Out Floorplan and DEF Files.....	2-77
Re-Synthesize Before Placement.....	2-78
Re-Synthesize Before Placement.....	2-79
Test For Understanding (1 of 2).....	2-80
Test For Understanding (2 of 2).....	2-81
Summary	2-82
Lab 2: Design Planning.....	2-83
Appendix A.....	2-84
Floorplan Exploration and Re-Synthesis Flow	2-85
Floorplan Exploration Objectives	2-86
Overview: Floorplan Exploration	2-87

Table of Contents

Load the Floorplan Definition (DEF) File	2-88
Pre-exploration Settings and Checks	2-89
Check Placement Readiness	2-90
Perform Low-effort Placement Optimization	2-91
Check and Fix Congestion	2-92
Modify the Floorplan	2-93
Analyze and Fix Timing	2-94
Write Out the Floorplan Files	2-95
Summary: Floorplan Exploration	2-96
Summary: 3rd Party Design Planning Flow	2-97

Unit 3: Placement

Unit Objectives	3-2
General IC Compiler Flow.....	3-3
Design Status Prior to Placement.....	3-4
IC Compiler Placement Flow.....	3-5
Placement Setup and Checks	3-6
“Fix” all Macro Cell Placements	3-7
Verify pnet Options and Ignored Layers	3-8
Verify Keepout Variable Settings (if used)	3-9
Non-Default Clock Routing	3-10
Specify Non-Default Routing Rules	3-11
Check Placement Readiness	3-12
Summary: Placement Setup and Checks.....	3-13
Design-for-Test (DFT) Setup.....	3-14
Pre-Existing Scan Chains.....	3-15
The Issue with Existing Scan Chains.....	3-16
SCANDEF-Based Chain Reordering.....	3-17
Re-ordering Chains within Same “Partition”.....	3-18
Example SCANDEF from Synthesis with DFT	3-19
Example: Placement with Existing Ordering.....	3-20
Example: Reordering Within Scan-Chains	3-21
Example: Reordering Within Partitions.....	3-22
Placement-Based Scan Chain Re-Ordering	3-23
Consider Extreme Block Aspect Ratios.....	3-24
Summary: DFT Setup	3-25
Power Setup	3-26
Where Does Power Dissipation Occur?.....	3-27
Leakage Power Optimization.....	3-28
Report Multi-Vth Cells	3-29
Reducing Dynamic Power Dissipation	3-30
Switching Activity Terminology	3-31
SAIF File Provides Switching Activity	3-32
What if SAIF is Not Available?.....	3-33

Table of Contents

Dynamic Power Optimization: LPP.....	3-34
Dynamic Power Optimization: GLPO	3-35
Summary: Power Optimization Flow	3-36
Test For Understanding.....	3-37
Placement and Optimization	3-38
Overview: Placement and Optimization	3-39
The Initial Placement and Optimization	3-40
Placement and Logic Optimization.....	3-41
Considerations for Using -congestion.....	3-42
No Hold Time Fixing.....	3-43
Post-Placement Analysis.....	3-44
Incremental Optimization	3-45
Apply Placement Constraints As Needed.....	3-46
Recall Problem: Sub-Critical Paths Ignored	3-47
Solution #1: User-defined Path Groups	3-48
Solution #2: Apply a Critical Range.....	3-49
Solution #3: Prioritizing Path Groups.....	3-50
Example: -weight	3-51
Complete Example.....	3-52
Incremental Logic Optimization: psynopt	3-53
Summary: Incremental Optimization.....	3-54
If the Design is Still Seriously Congested	3-55
Enable Global Router During Optimization	3-56
Summary: Placement and Optimization	3-57
Improving Congestion and Setup Timing.....	3-58
Overview: Improve Congestion/Timing	3-59
refine_placement.....	3-60
psynopt.....	3-61
Summary: Improve Congestion/Setup Timing	3-62
Test For Understanding (1 of 2).....	3-63
Test For Understanding (2 of 2).....	3-64
Techniques with More User Control.....	3-65
Build User-Controlled Balanced Buffer Trees	3-66
Build Skew-Optimized Buffer Trees	3-67
Relative Placement.....	3-68
What's Special about Data Path Logic?.....	3-69
The Ideal Layout for Data Path.....	3-70
Data Path Layout using Traditional P&R Tool.....	3-71
Traditional Solution: Custom/Manual Layout	3-72
IC Compiler's Solution: Relative Placement	3-73
Features and Benefits of Relative Placement.....	3-74
Candidates for Relative Placement	3-75
More Information on Relative Placement.....	3-76
Summary	3-77
Lab 3: Placement.....	3-78

Table of Contents

Unit 4: Clock Tree Synthesis

Unit Objectives	4-2
IC Compiler Flow	4-3
Design Status, Start of CTS Phase	4-4
Is the Design Ready for CTS?	4-5
Starting Point before CTS	4-6
Clock Tree Synthesis	4-7
CTS Goals	4-8
Clock Tree Synthesis (CTS) (1/2)	4-9
Clock Tree Synthesis (CTS) (2/2)	4-10
Where does the Clock Tree Begin and End?	4-11
Define Clock Root Attributes (1/2)	4-12
Define Clock Root Attributes (2/2)	4-13
Stop, Float and Exclude Pins	4-14
Generated and Gated Clocks	4-15
Skew Balancing not Required?	4-16
User-defined or Explicit Stop Pins	4-17
Defining an Explicit Stop Pin	4-18
Defining an Explicit Float Pin	4-19
Preserving Pre-Existing Clock Trees	4-20
Impact of Preexisting Clock Cells	4-21
Test for Understanding	4-22
Specifying Skew / Insertion Delay Targets	4-23
Clock by Clock Settings	4-24
Set Buffer/Inverter Selection Lists	4-25
When Clock Tree DRCs are Used	4-26
Non-Default Clock Routing	4-27
Specifying Non-Default Rules	4-28
Nondefault Rule Options	4-29
NDR Recommendations	4-30
Invoke CTS: Core Command	4-31
<code>clock_opt</code> use recommendation	4-32
Effects of Clock Tree Synthesis	4-33
Incremental Placement / Optimization	4-34
Minimize Hold Time Violations in Scan Paths	4-35
Recommended Flow	4-36
Analysis using the CTS GUI	4-37
Analyzing CTS Results	4-38
What about CTS Operating Conditions?	4-39
Clock Tree Optimization	4-40
(Embedded) Clock Tree Optimization	4-41
Balancing Multiple Synchronous Clocks	4-42
Inter-Clock Delay Balancing	4-43
Inter-Clock Delay Balancing with Offset	4-44
SDC Latencies	4-45

Table of Contents

Core vs. Atomic Commands	4-46
Flow Using Atomic Commands.....	4-47
Test for Understanding	4-48
Unit Objectives Summary.....	4-49
Lab 4: Clock Tree Synthesis	4-50
Appendix A	4-51
IO Latency Auto Update.....	4-52
Auto Update with Virtual Clocks	4-53
Appendix B	4-54
CTS with Logical Hierarchy	4-55
Clock Tree Cells Added in Top Hier	4-56
Appendix C	4-57
Clock Tree Configuration Control.....	4-58
Clock Tree Configuration Syntax	4-59
Appendix D	4-60
CTS – Naming Convention.....	4-61

Unit 5: Multi Scenario Optimization

Unit Objectives	5-2
Timing Analysis during Optimization	5-3
What about other Situations?	5-4
What is the # of Runs?.....	5-5
Corners Represent Delay at Different OpCon	5-6
Multiple Corners – Multiple Modes	5-7
Scenarios	5-8
Multi Scenario Solution in IC Compiler.....	5-9
How Are Violations Fixed?	5-10
MCMM / Scenario Setup	5-11
Defining Scenarios.....	5-12
Global Setup.....	5-13
Scenario-Specific Setup – S1	5-14
Scenario-Specific Setup – S4.....	5-15
Switching between Scenarios	5-16
CTS Operates with One Scenario	5-17
Leakage Only Scenario	5-18
MCMM Timing Analysis (1/3).....	5-19
MCMM Timing Analysis (2/3).....	5-20
MCMM Timing Analysis (3/3).....	5-21
How Much is Too Much?	5-22
MCMM Scenario Reduction Analysis.....	5-23
MCMM High Capacity Flow	5-24
Analysis Types.....	5-25
Launch vs. Capture Path – Use Which Slew?.....	5-26
Setup is Optimistic in bc_wc	5-27

Table of Contents

On Chip Variation Uses Safer Slews	5-28
Analysis Types Summary	5-29
On-Chip Variation: Single Library	5-30
On-Chip Variation: Multiple Libraries	5-31
On-Chip Variation: Single Library + Derating	5-32
Applying Derating Factors.....	5-33
Global Derating versus Specific Derating	5-34
Unit Objectives Summary.....	5-35
Lab 5: Multiple Scenario Optimization	5-36
Appendix.....	5-37
Link Libraries & PVT Assumptions	5-38
Unique Identification Of Libraries.....	5-39
Library Grouping	5-40
Incorrect Library Selected.....	5-41
MCMM Supports up to 3 TLU+ Files	5-42
Scaling Resistance and Capacitance	5-43
Filtering Coupling Capacitances.....	5-44
TLU+ Temperature Scaling	5-45
No Delay Scaling with k-factors in MCMM	5-46

Unit 6: Routing and Crosstalk

Unit Objectives	6-2
IC Compiler Flow	6-3
Design Status, Start of Routing Phase	6-4
Pre-Route Checks.....	6-5
Routing Fundamentals: Goal	6-6
Grid-Based Routing System	6-7
Routing over Macros.....	6-8
Change the Preferred Routing Direction.....	6-9
Routing Operations	6-10
Route Operations: Global Route	6-11
Route Operations: Global Route Summary	6-12
Route Operations: Track Assignment	6-13
Route Operations: Detail Routing.....	6-14
Route Operations: Search&Repair.....	6-15
Test for Understanding	6-16
General Flow for Routing	6-17
Set Routing Options Prior to Routing Steps	6-18
Route Clock Nets First.....	6-19
Core Routing: route_opt	6-20
First route_opt Example.....	6-21
Perform Initial Redundant Via.....	6-22
Post Route Optimization Examples	6-23
Core Routing Strategy.....	6-24

Table of Contents

Analysis of the Routing DRC Errors	6-25
Fixing DRC Violations	6-26
PostRoute Delay Calculation Algorithms	6-27
Test for Understanding	6-28
Galaxy Crosstalk	6-29
What is Crosstalk?	6-30
Crosstalk-Induced Noise (aka Glitches)	6-31
Crosstalk-Induced Delay	6-32
Crosstalk Prevention in IC Compiler	6-33
Crosstalk Correction in IC Compiler	6-34
Example Full Crosstalk Flow	6-35
Xtalk-Reduction at Work	6-36
Wire Sizing (Aka Applying NDRs)	6-37
Wire Sizing at Work	6-38
ECOs: Making Changes Late in the Flow	6-39
The Two Types of ECO Flows	6-40
Functional ECO Flows	6-41
Non-Freeze Silicon ECO	6-42
Hierarchical ECO Change File Example	6-43
Inserting Spare Cells for Freeze Silicon ECO	6-44
Protecting Spare Cell Placement	6-45
Freeze Silicon ECO: Metal Change Only	6-46
ECO Routing Example	6-47
Zroute: Synopsys' New Routing Technology	6-48
State of the Art Routing Technology	6-49
Concurrent DFM Optimizations	6-50
Multi-threaded Throughout	6-51
10X Speed-Up On Mainstream Hardware	6-52
Zroute Is GA in IC Compiler 2008.09!	6-53
Zroute Users Tell The Story	6-54
Routing & Crosstalk Summary	6-55
Lab 6a: Routing & Crosstalk, Lab 6b: ECO	6-56

Unit 7: Chip Finishing and DFM

Unit Objectives	7-2
IC Compiler Flow	7-3
Design Status, Completion of Routing Phase	7-4
Chip Finishing Flow	7-5
Problem: Gate Oxide Integrity	7-6
Antenna Rules	7-7
Solution 1: Splitting Metal or Layer Jumping	7-8
Solution 2: Inserting Diodes	7-9
Antenna Fixing Flow	7-10
Antenna: Misc	7-11

Table of Contents

Random Particle Defects.....	7-12
Reporting the Critical Area.....	7-13
Solution: Wire Spreading + Widening.....	7-14
Controlling Minimum Jog Length	7-15
Proactive: Density-Driven During GR and TA	7-16
Voids in Vias during Manufacturing	7-17
Via Control Through Tcl Variables	7-18
Insert Redundant Vias.....	7-19
Reporting Redundant Via Count.....	7-20
Redundant Via Methodologies	7-21
Why Filler Cell Insertion?	7-22
Insert Cells to Fill Unused Placement Sites.....	7-23
Problem: Metal Over-Etching.....	7-24
Solution: Metal Fill insert_metal_filler	7-25
Timing-Driven Rule-Based Metal Fill.....	7-26
Problem: Metal Erosion	7-27
Problem: Metal Liftoff.....	7-28
Solution: Metal Slotting.....	7-29
DFM Issues and Solutions Summary.....	7-30
Final Validation	7-31
Final Validation: Parasitics (SPEF or SBPF)	7-32
Final Validation: Netlist Output.....	7-33
Final Validation: GDS2 Output	7-34
Hercules™ VUE Integration.....	7-35
Accessing VUE in IC Compiler.....	7-36
Running VUE.....	7-37
Test for Understanding	7-38
Summary	7-39
Lab 7: Chip Finishing	7-40
Appendix A	7-41
Critical Area Definition	7-42
Discrete Defect Size Distribution	7-43
Appendix B	7-44
Wire Spreading	7-45
Redundant Via Insertion 1/2	7-46
Redundant Via Insertion 2/2	7-47
Filler Cell Insertion	7-48
Metal Fill Insertion	7-49

Table of Contents

Unit CS: Customer Support

Synopsys Support Resources	CS-2
SolvNet Online Support Offers.....	CS-3
SolvNet Registration is Easy	CS-4
Support Center: AE-based Support.....	CS-5
Other Technical Sources	CS-6
Summary: Getting Support	CS-7



IC Compiler 1

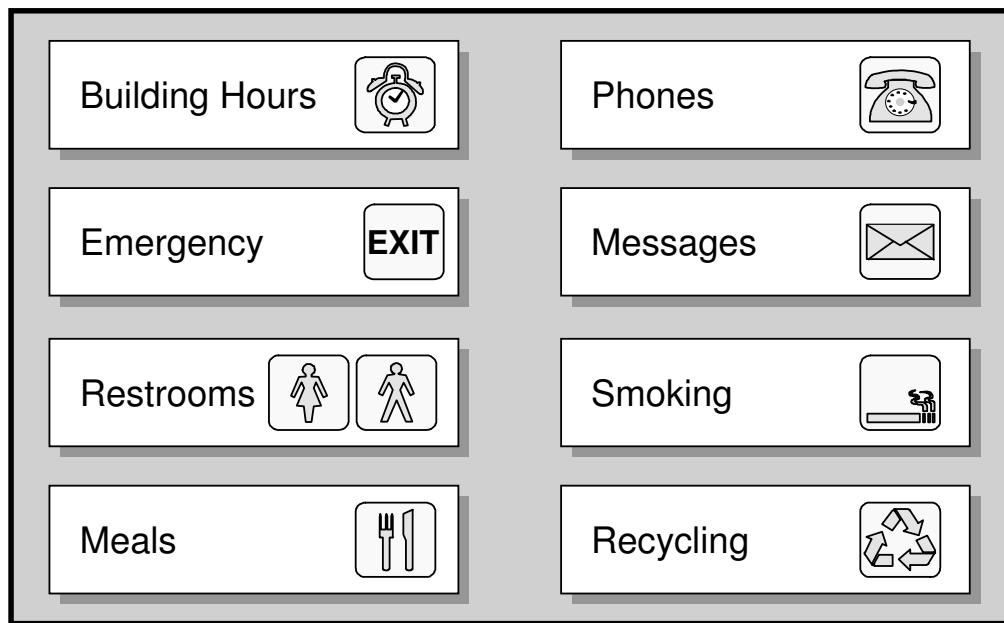
2008.09-SP2

Synopsys Customer Education Services

© 2009 Synopsys, Inc. All Rights Reserved

Synopsys 20-I-071-SSG-008

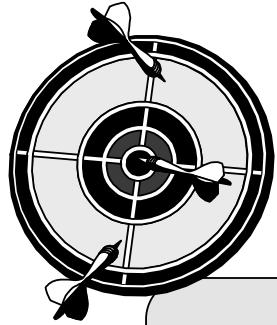
Facilities



Please turn off cell phones and pagers

i-2

Workshop Goal



**Use IC Compiler to perform placement,
DFT, CTS, routing and optimization,
achieving timing closure for designs with
moderate to high design challenges.**

i-3

Target Audience

**ASIC, back-end or layout designers
with experience in standard cell-
based automatic Place&Route.**



i-4

Workshop Prerequisites

- You should have knowledge of the following:

- UNIX and X-Windows
- A Unix text editor, e.g. Emacs, vi, pine
- Basic physical design, layout or place&route concepts

i-5

Introductions

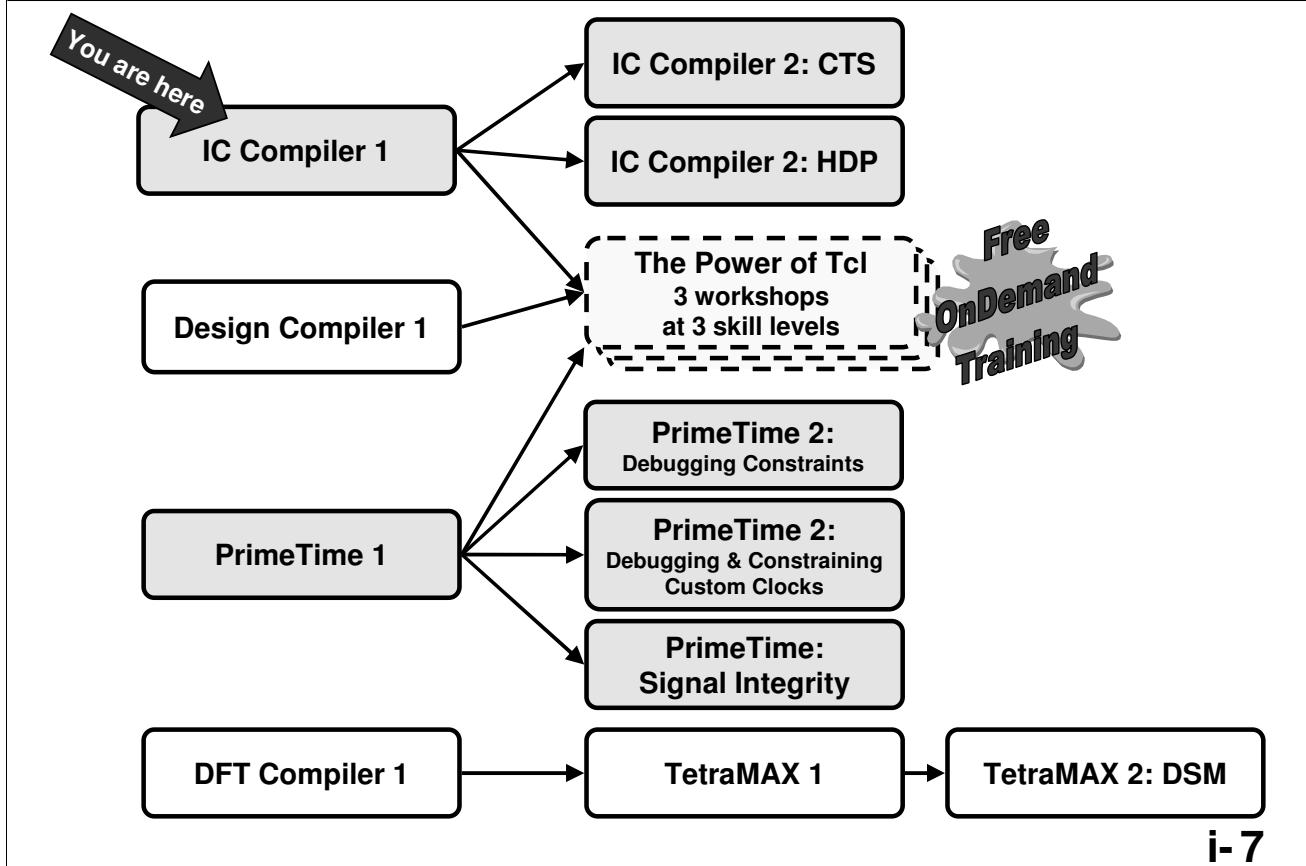
- Name
- Company
- Job Responsibilities
- EDA Experience
- Main Goal(s) and Expectations for this Course

i-6



EDA = Electronic Design Automation

Curriculum Flow



The entire Synopsys Customer Education Services course offering can be found at:

<http://training.synopsys.com>

A number of workshops are offered as *OnDemand* playback training for FREE! Visit the following link to view the available workshops:

<http://solvnet.synopsys.com/training>

(see under “Tool and Methodology Training”)

Agenda

**DAY
1**

i Introduction & Overview



1 Data Setup & Basic Flow



2 Design Planning



i-8

Agenda

**DAY
2**

3 Placement



4 Clock Tree Synthesis



i-9

Agenda

**DAY
3**

5 Multi Scenario Optimization



6 Routing and Crosstalk



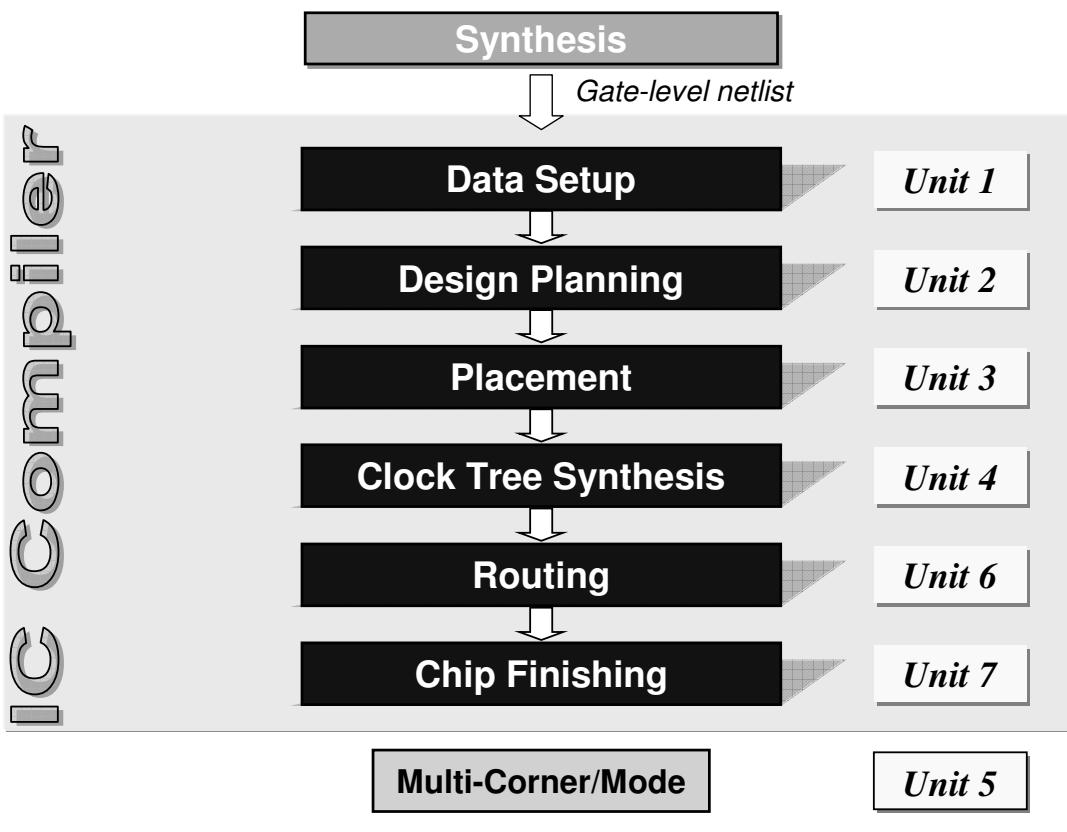
7 Chip Finishing and DFM



CS Customer Support

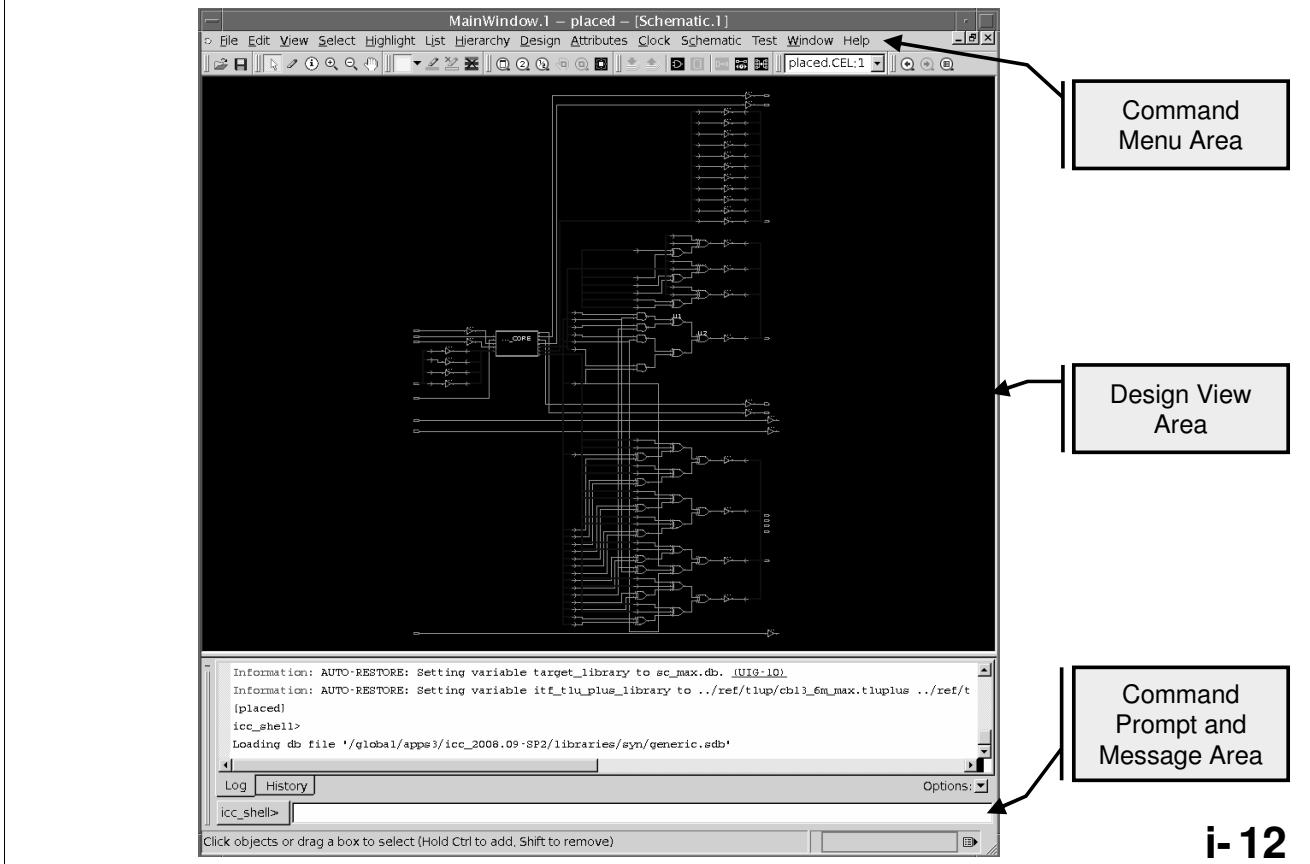
i-10

High-Level IC Compiler Flow

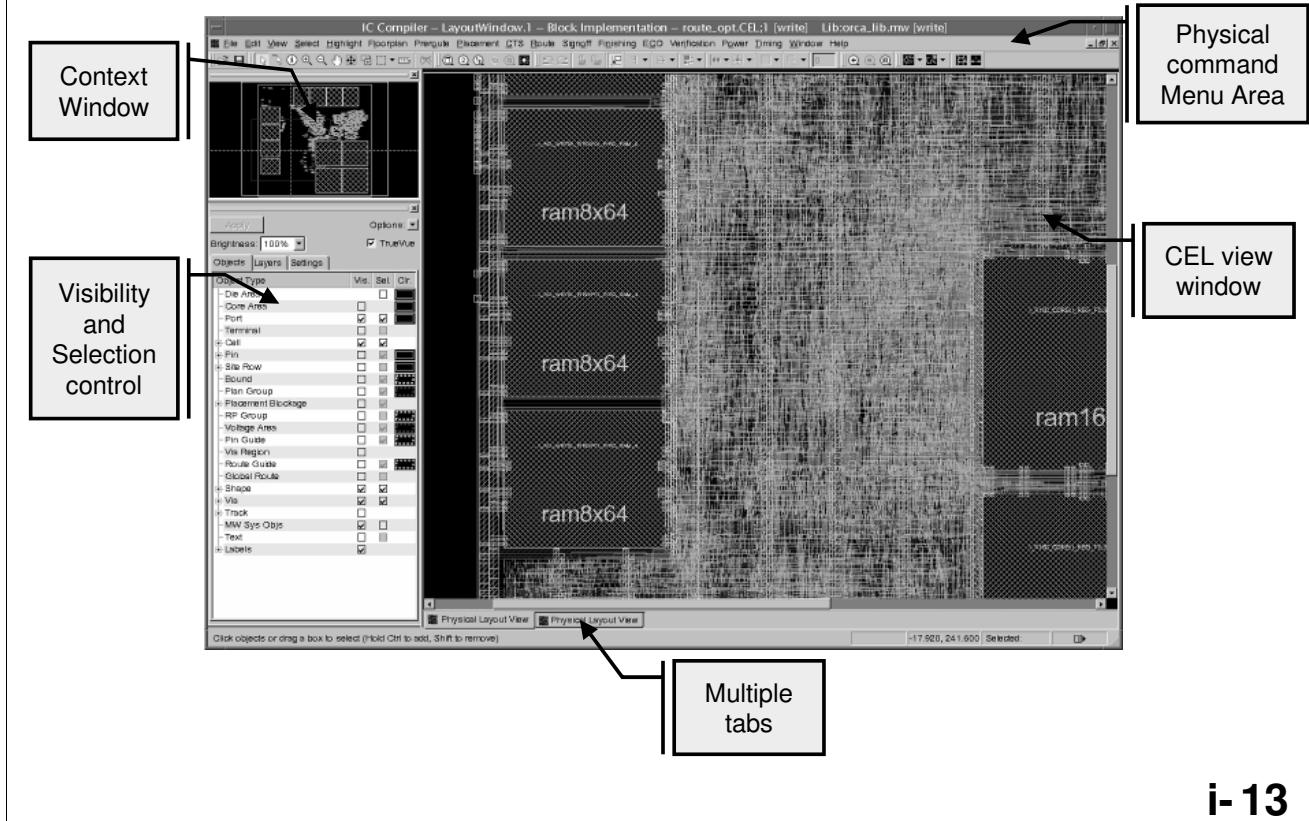


i-11

Lab 0: IC Compiler GUI – *MainWindow*



Lab 0: IC Compiler GUI – *LayoutWindow*



i-13

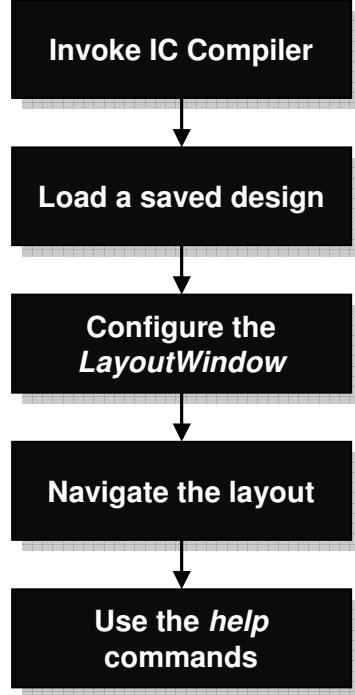
Lab 0A: IC Compiler GUI



45minutes

**Explore some of IC Compiler's
GUI and command line features.**

Note: Lab 0B is an optional lab that covers additional GUI features, like highlighting layout objects, cross-probing between the layout and schematic, timing analysis, and more. Try it if you have some extra time during, or after the workshop.



i- 14

Agenda

**DAY
1**

i Introduction & Overview



1 Data Setup & Basic Flow



2 Design Planning



Unit Objectives



After completing this unit, you should be able to:

- **Perform data setup:**
 - Create a Milkyway *design library* and *design cell*
 - Load the necessary data required to run IC Compiler
- **Execute a basic flow for design planning, placement, CTS and routing in IC Compiler**

1-2

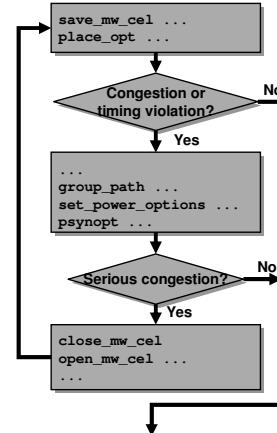
A Word of Caution About Scripts and Flows

This workshop contains many scripts and flow diagrams showing specific commands executed in a specific order

- These flows DO NOT represent “the recommended flow”
 - Each flow is just one example of many possible flows
 - They help to better organize and present the material
- The specific commands and order of execution required to achieve the best results is completely design dependent

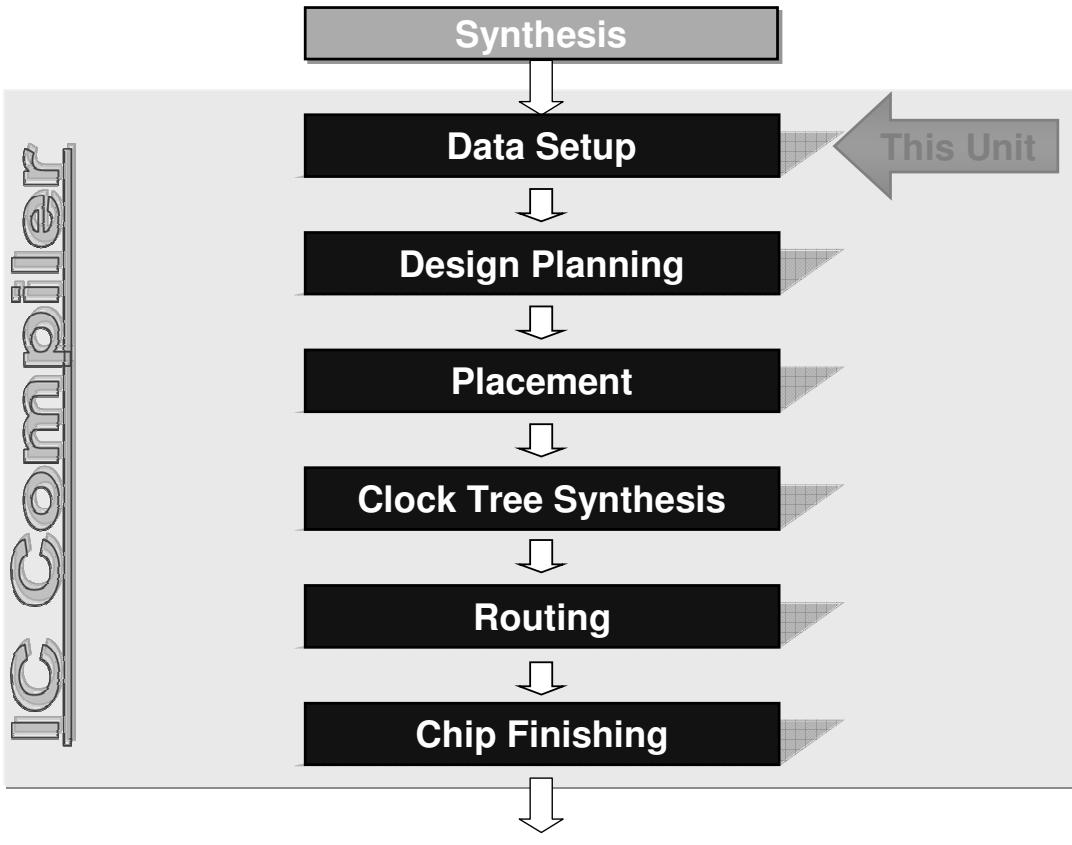


There is no “golden script” for physical design



1-3

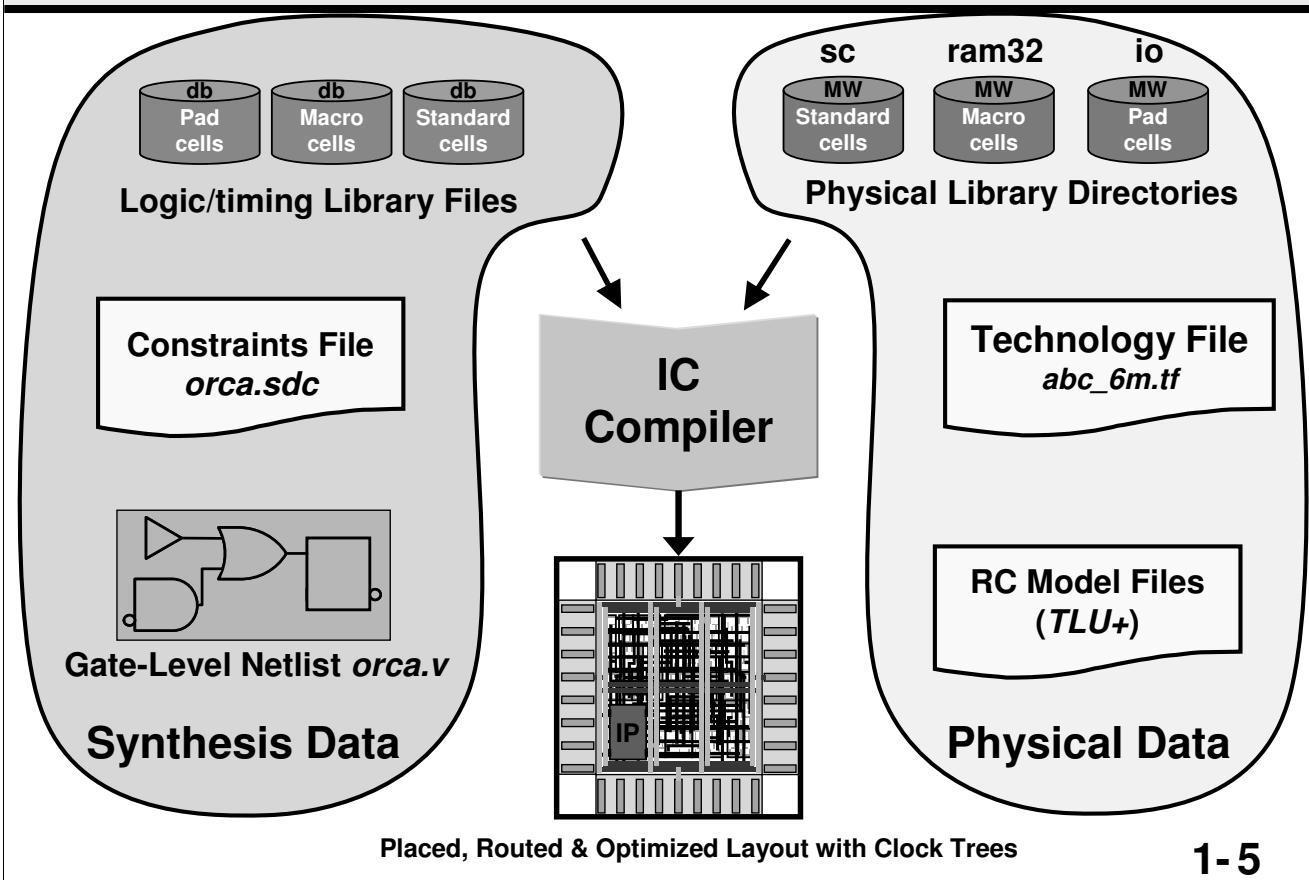
General IC Compiler Flow



1-4

This Unit is primarily about the first step in the IC Compiler flow - Data Setup. Near the end of the Unit we will briefly introduce design planning as well as the “core” commands for Placement, CTS and Routing, for the purposes of performing Lab 1, which takes you through the IC Compiler flow (at a high level).

Data Setup

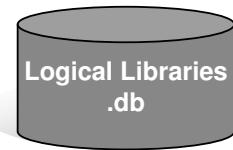


1-5

These files are not loaded directly into IC Compiler's "memory", but must instead be loaded into a project-specific "container", known as a "design library" (see upcoming step #3). The next slides will cover all the steps associated with data setup.

Logical Libraries

- Provide timing and functionality information for all standard cells (and, or, flipflop, ...)
- Provide timing information for hard macros (IP, ROM, RAM, ...)
- Define drive/load design rules:
 - Max fanout
 - Max transition
 - Max/Min capacitance
- Are usually the same ones used by Design Compiler during synthesis
- Are specified with variables:
 - target_library
 - link_library

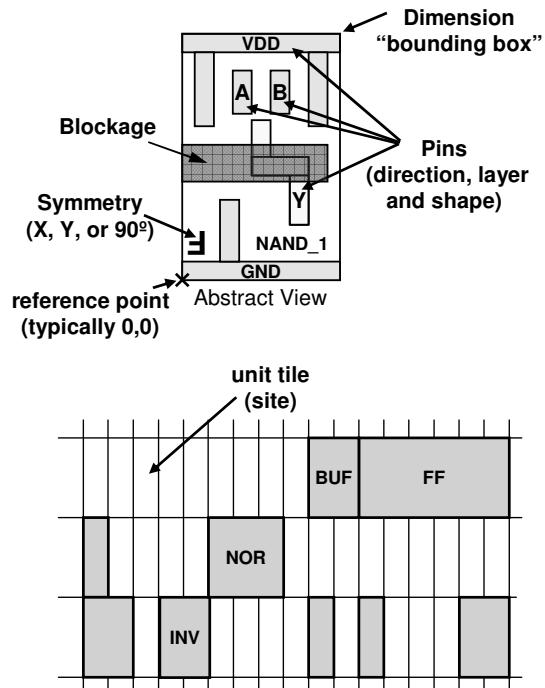


1-6

Physical Reference Libraries

Reference Libraries
(Milkyway)

- Contain physical information of *standard, macro and pad* cells, necessary for placement and routing
- Define placement *unit tile*
 - Height of placement rows
 - Minimum width resolution
 - Preferred routing directions
 - Pitch of routing tracks
 - ...
- Are specified with the command:
 - `create_mw_lib -mw_reference_library ...`

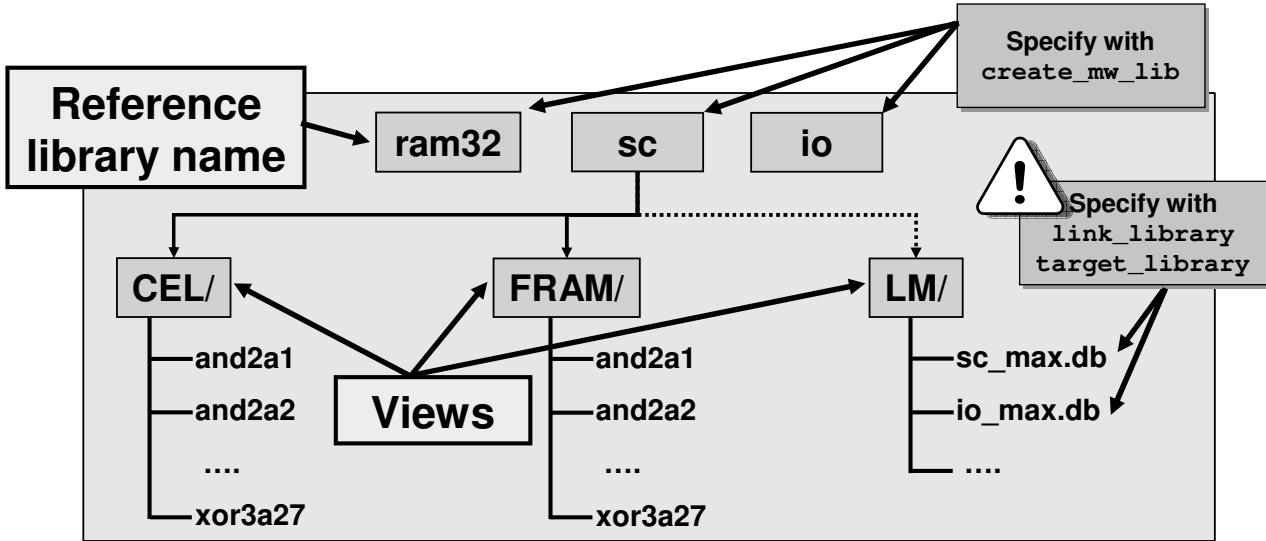


1-7

Milkyway Structure of Physical Libraries

Each physical or reference library is a UNIX directory under which information is stored in sub-directories called *views*

- FRAM: Abstract view - Used during P&R
- LM: (Optional) Logic model view - Contains *db* logical libraries¹



1-8

The *CEL* view contains cells with the full layout view (all layers). This view is used when “taping out” the design, i.e. writing out the complete *stream* or *GDSII* database, which defines all the processing layers, and is handed off to the fab for wafer implementation.

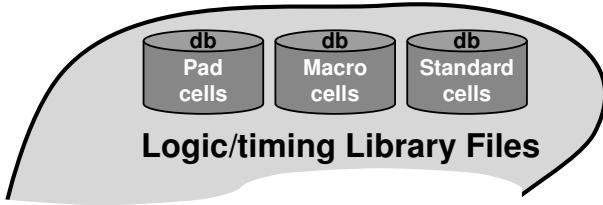
¹ The “LM” directory is not required to be present, however, the *db* libraries are required, whether they happen to be stored in the *Milkyway* reference library structure, or elsewhere in the UNIX file structure.



Even if the *Milkyway* reference library contains the *db* or logical libraries, IC Compiler does not automatically locate and reference them when the physical libraries are specified (shown later, using the `create_mw_lib` command). The user must specify logical libraries separately with the `target_library`, `link_library` and `search_path` variables (shown on next slide).

If the *db* files are inside the *Milkyway* libraries, the `search_path` points to each *LM* directory inside each *Milkyway* reference library, otherwise, it points to whatever other directories they happen to be in.

1. Specify the Logical Libraries



.synopsys_dc.setup

```
lappend search_path ./design_data ./scripts $MW_libs  
lappend search_path [glob $MW_libs/*/LM]  
set link_library "* gates_max.db io_max.db rams_max.db"  
set target_library "gates_max.db"  
set symbol_library "* gates.sdb io.sdb rams.sdb"
```

These variables can be entered in the *icc_shell* environment each session, or more conveniently:

- Entered once in the `.synopsys_dc.setup` file, which is automatically read by the tool when ICC is invoked
- Note: Specifying *min* libraries will be discussed in Unit 5

TCL: `glob` returns files/directories that match the specified pattern

1-9

IC Compiler uses the “link library” to resolve all the instantiated components in a netlist. A netlist is *resolved* if IC Compiler finds a corresponding *library cell* (in any of the specified `link_library` files) for each leaf cell that is instantiated in the netlist, as well as a corresponding *design* in IC Compiler memory (that’s what the * represents) for each sub-design in the netlist. This “resolving” happens during the *link* step, described in a few pages.

The “target library” usually specifies only the library with basic logic gates (the *standard cells*, not the *IO pads* or *marcos*). IC Compiler *targets* this library during logic optimization, when cell sizing and logic transformations are performed. The `link_library` and `target_library` variable settings are usually the same as specified for Design Compiler during synthesis. The variables can be conveniently specified in the `.synopsys_dc.setup` file, which is read automatically by IC Compiler (as well as Design Compiler) upon invocation, if the file is located in the user’s home directory, or more commonly, the “current working directory” – the directory in which IC Compiler is invoked.

The `search_path` variable is a convenient way to shorten file specifications. When a file name is specified, IC Compiler looks for that file in the specified list of `search_path` directories. Without this variable, the user needs to specify the relative path to each file, e.g.

```
set link_library ".../db_libs/gates_max.db .../db_libs/io_max..."
```

After loading the netlist (`read_verilog` or `import_designs`) you can check what libraries are loaded into IC Compiler’s memory with `list_libs`.

In “`lappend search_path [glob $MW_libs/*/LM]`”, `$MW_libs` represents the directory where the reference libraries (like `sc`, `ram32` and `io`, represented by *) are located.

The *symbol libraries* provide icons which are used only when viewing the schematic in the GUI.

IC Compiler supports the newer CCS (Composite Current Source) libraries. CCS employs a current-based approach that models timing, noise, and power more accurately for 90 nm and smaller designs.

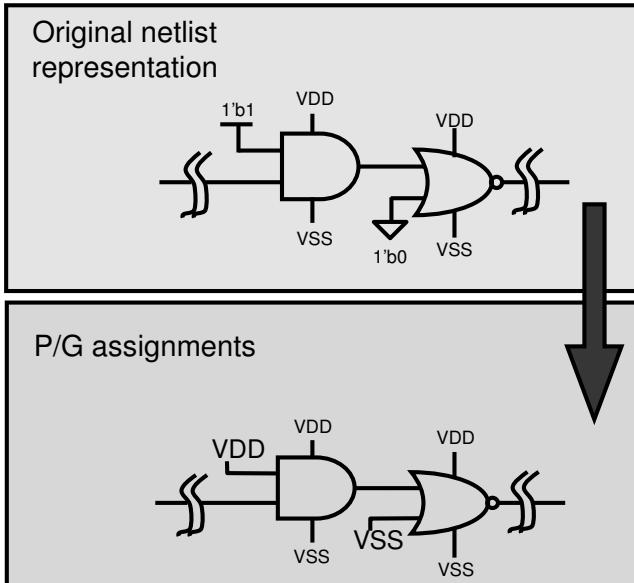
2. Define ‘logic0’ and ‘logic1’

- In the netlist, “tie-high” and “tie-low” inputs may be connected to logical ‘1’ and ‘0’
- Define corresponding power and ground signal names
 - As defined by the names of the P/G pre-routes in your floorplan

.synopsys_dc.setup

```
set mw_logic0_net "VSS"  
set mw_logic1_net "VDD"
```

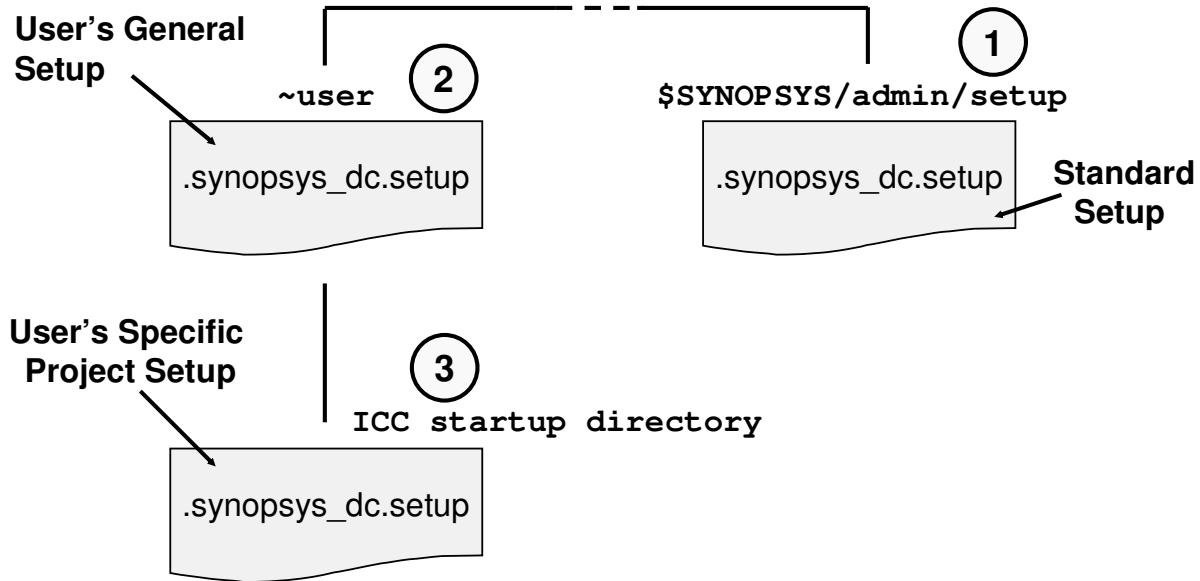
Set by default in 2008.09-SP2



1-10

In version 2008.09-SP2 the above variables are set by default, as shown above. In some earlier versions (e.g. 2007.12) the variables were required to be explicitly set by the user every time the tool is invoked.

IC Compiler Initialization Files

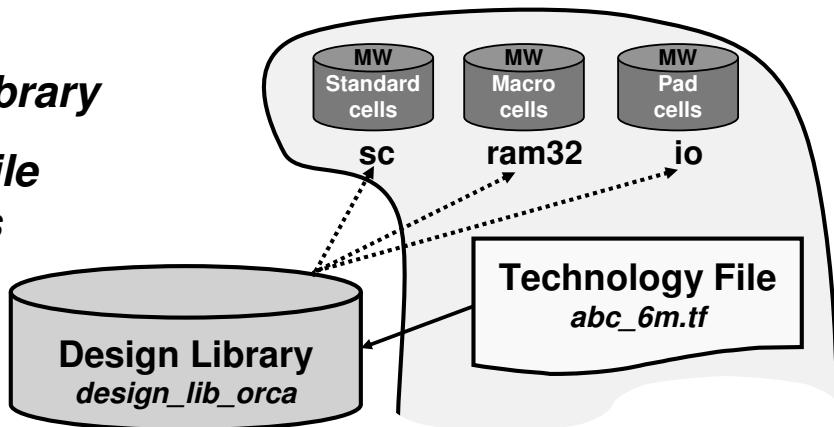


Commands in `.synopsys_dc.setup` are executed upon tool startup, in the order shown above.

1-11

3. Create a “Container”: The *Design Library*

- Create a *design library*
- Specify the *tech file* and *reference libs*



```
create_mw_lib design_lib_orca -open \
    -technology abc_6m.tf \
    -mw_reference_library "sc ram32 io"
set_check_library_options -all
check_library
```

Specified files and libraries are assumed to be located in “CWD” or in the search_path directories.

1-12

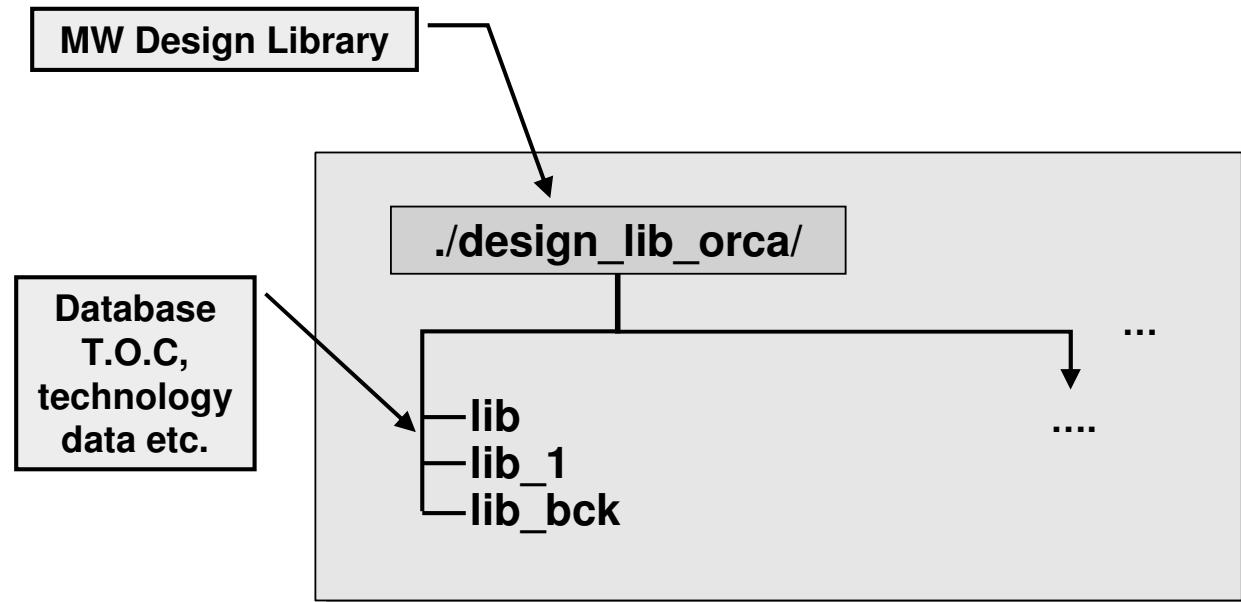
The design library is a Milkyway UNIX-based database structure, created by the user, which will eventually contain all the associated input data required for placement, CTS, routing, etc, as well as the physical “design cell” or layout.

The first step in data-setup is to create the design library. This entails giving the library a user-defined name and specifying the technology file as well as the physical “reference” libraries (layout cells for standard cells, macro and IO pad cells). While the technology file is actually loaded or read into the design library, the reference libraries are not. The design library creates “pointers” to the UNIX location of the libraries – it “references” them instead of loading them.

In the example above the technology file name and the library directory names were specified without their UNIX directory “location”. This works if the specified files and directories happen to be located in the “current working directory” (CWD) – the directory from which IC Compiler was invoked, or, if their location directories are appended to the `search_path` variable as shown previously:

```
lappend search_path ./design_data ./scripts $MW_libs
lappend search_path [glob $MW_libs/*/LM])
```

Initial Structure of a *Milkyway* Design Library



1-13

The result of the `create_mw_lib` command is a *Milkyway* design library. The design library is contained under a UNIX directory which represents the user-defined name of the design library. Initially, this library contains only a few lib* files. These files maintain a table-of-contents of the library, the technology file data that was read in, pointers to the reference libraries, and more. The design library will be populated with additional directories and files after additional data setup steps and eventual P&R steps.

The Technology File (.tf file)

- The *technology file* is unique to each technology
- Contains metal layer technology parameters:
 - Number and name designations for each layer/via
 - Physical and electrical characteristics of each layer/via
 - Design rules for each layer/Via (Minimum wire widths and wire-to-wire spacing, etc.)
 - Units and precision for electrical units
 - Colors and patterns of layers for display
 - ...

1-14

Example of a Technology File

```
Technology {  
    unitTimeName      = "ns"  
    timePrecision     = 1000  
    unitLengthName    = "micron"  
    lengthPrecision   = 1000  
    gridResolution    = 5  
    unitVoltageName   = "v"  
}  
  
...  
  
Layer "m1" {  
    layerNumber       = 16  
    maskName          = "metall1"  
    pitch              = 0.56  
    defaultWidth      = 0.23  
    minWidth           = 0.23  
    minSpacing         = 0.23  
}  
...
```

abc_6m.tf

1-15

The `check_library` Command

- **Reports library inconsistencies, for example:**

- Between logic (`link_library`) and physical libraries:
 - ◆ Missing cells
 - ◆ Missing or mismatched pins
- Within physical libraries:
 - ◆ Missing CEL (layout) or FRAM (abstract) view cells
 - ◆ Duplicate cell name in multiple reference libraries

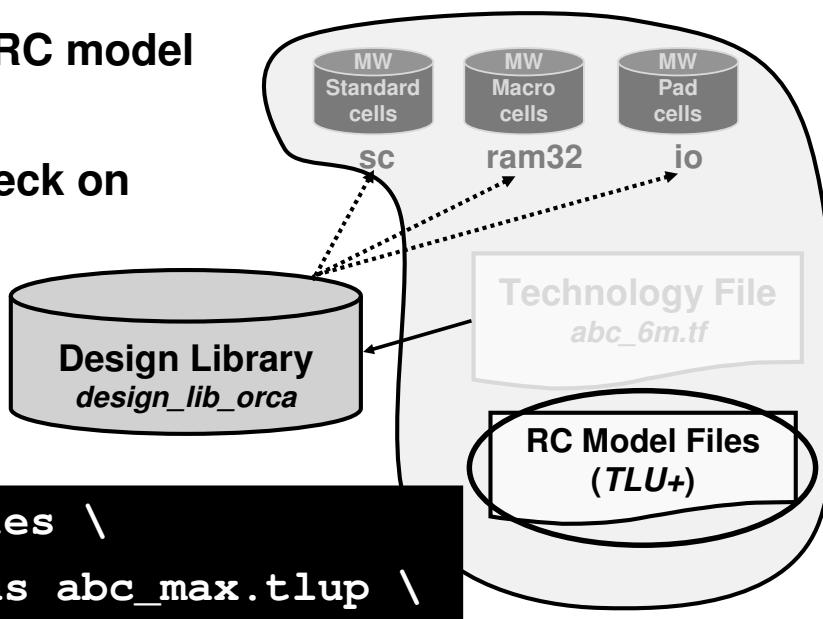
- **Recommended after creating the design library**

```
create_mw_lib ...
set_check_library_options -all
check_library
```

1-16

4. Specify *TLU+* Parasitic RC Model Files

- Specify the *TLU+* RC model files to be used
- Perform sanity check on settings and files



```
set_tlu_plus_files \
    -max_tluplus abc_max.tlup \
    -min_tluplus abc_min.tlup \
    -tech2itf_map abc.map
check_tlu_plus_files
```

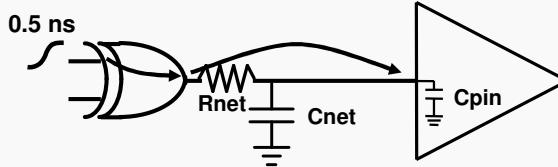
1-17

TLU+ models are described on the next pages.

IC Compiler requires *TLU+* models for a successful design flow from placement through routing. While it may be possible to accomplish some tasks without *TLU+* models, successful completion of the entire design flow can not be assured without them.

The `check_tlu_plus_files` command checks the existence of the specified files and performs a sanity check on the *TLU+* settings.

Timing is Based on Cell and Net Delays



$$\text{Cell Delay} = f(\text{Input Transition Time}, C_{net} + C_{pin})$$

$$\text{Net Delay} = f(R_{net}, C_{net} + C_{pin})$$

- ICC calculates delay for every cell and every net
- To calculate delays, ICC needs to know each net's parasitic Rs and Cs

1-18

Cell delay is calculated using non-linear delay models, which are stored in the logical libraries (.db files). NLDM is highly accurate as it is derived from SPICE characterizations. The delay is a function of the input transition time of the cell (TInput) [also called slew], the driving strength of the cell (RCell), the wire capacitance (CNet) and the pin capacitance of the receivers (CPin). A slow input transition time will slow the rate at which the cell's transistors can change state (from “on” to “off”), as well as a large output load ($C_{net} + C_{pin}$), thereby increasing the “delay” of the logic gate.

There is another NLDM table in the library to calculate output transition. Output transition of a cell becomes the input transition of the next cell down the chain.

		Output Load (pF)			
		.005	.05	.10	.15
Input Trans (ns)	0.0	.1	.15	.2	.25
	0.5	.15	.23	.3	.38
	1.0	.25	.4	.55	.75

Cell Delay (ns)

		Output Load (pF)			
		.005	.05	.10	.15
Input Trans (ns)	0.00	0.10	0.20	0.37	0.60
	0.50	0.18	0.30	0.49	0.80
	1.00	0.25	0.40	0.62	1.00

Output Transition (ns)

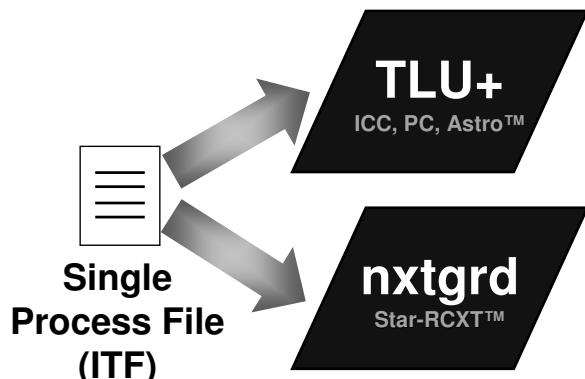
TLU+ Models

- IC Compiler calculates interconnect C and R values using net geometry and the TLU+ look-up tables
- Models UDSM process effects

UDSM Process Effects

- Conformal Dielectric
- Metal Fill
- Shallow Trench Isolation
- Copper Dishing:
 - Density Analysis
 - Width/Spacing
- Trapezoid Conductor

- Some vendors provide only an *ITF* process file
- User must then generate *TLU+* from *ITF* (see below)



1-19



UDSM = Ultra Deep SubMicron

ITF = Interconnect Technology Format

If the vendor did not provide *TLU+* files, the user can generate *TLU+* from ITF data. This requires a Star-RCXT license:

```
unix% grdgenxo -itf2TLUPlus -i <ITF file> -o <TLU+ file>
```

where: *-itf2TLUPlus* generates *TLU+* instead of a *nxtgrd* file

-i is the ITF file

-o is the output, binary *TLU+* model file

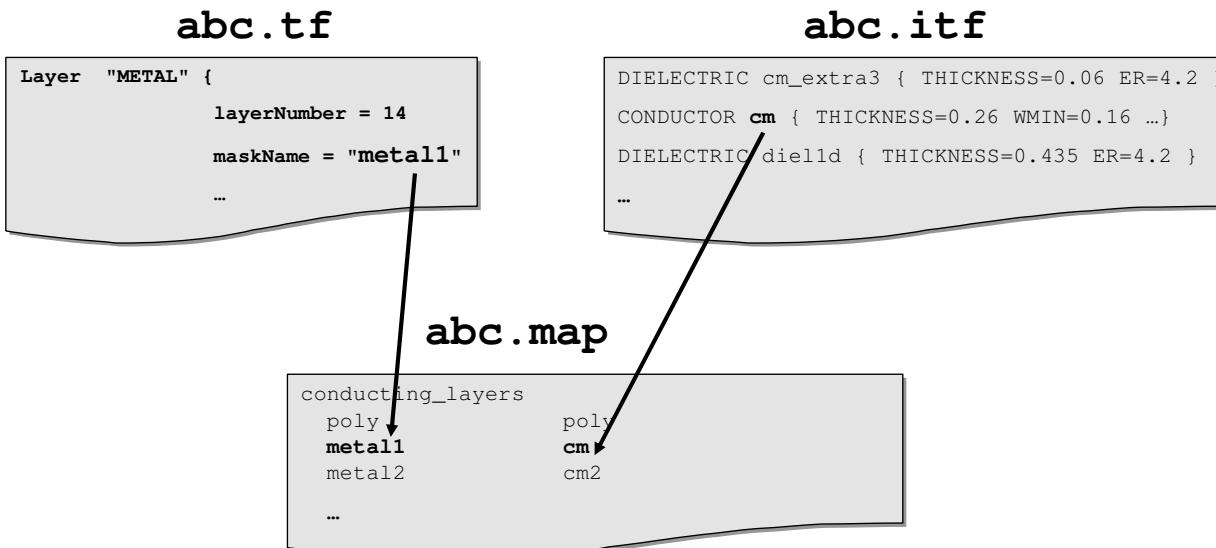
When generating *TLUPlus* models with *grdgenxo*, the *-itf2TLUPlus* option must be the first option specified. Always use the latest Star-RCXT release to generate the models.

More and more ASIC vendors are supporting *TLU+* models and they may provide the binary cap table files for your use.

If possible, generate *TLU+* models for at least two operating corners: min and max.

Mapping file

The *Mapping File* maps the technology file (.tf) layer/via names to Star-RCXT (.itf) layer/via names.



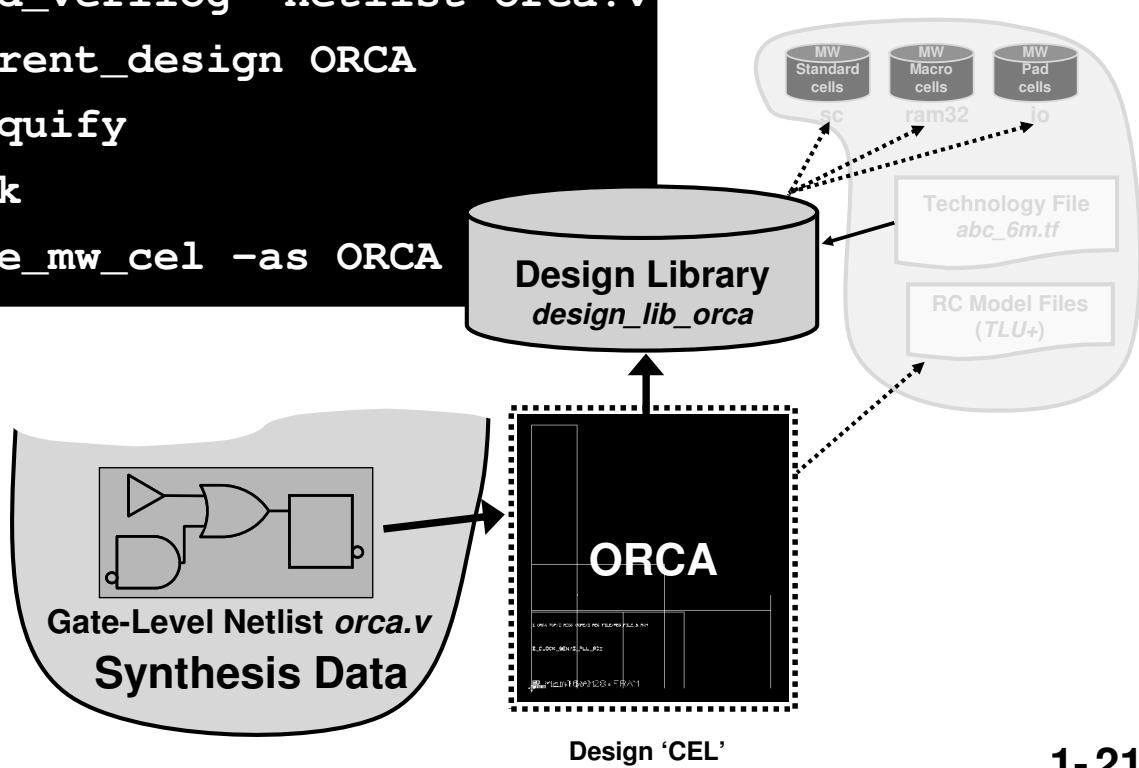
1-20

The itf information is contained in the TLU+ file.

The map file is needed even when the **tf** and **itf** names are matched 1 for 1.

5a. Read the Netlist and Create a Design CEL

```
read_verilog -netlist orca.v  
current_design ORCA  
uniquify  
link  
save_mw_cel -as ORCA
```



1-21

IC Compiler can also read the *ddc* netlist format using `read_ddc`. *ddc* is a format that can be written out by Design Compiler after synthesis. It is a binary format which can contain design constraints and attributes, in addition to the netlist information.

ICC can read in one complete hierarchical file: `read_verilog file_hier.v`

Or multiple files:

```
read_verilog top.v; read_verilog sub1.v; ... OR  
read_verilog "top.v sub1.v sub2.v ..."
```

`current_design`: Since a netlist can contain a hierarchical design with many sub-designs, the user should specify which design (usually the top-level Verilog *module*) is the *current design* to be worked on by IC Compiler.

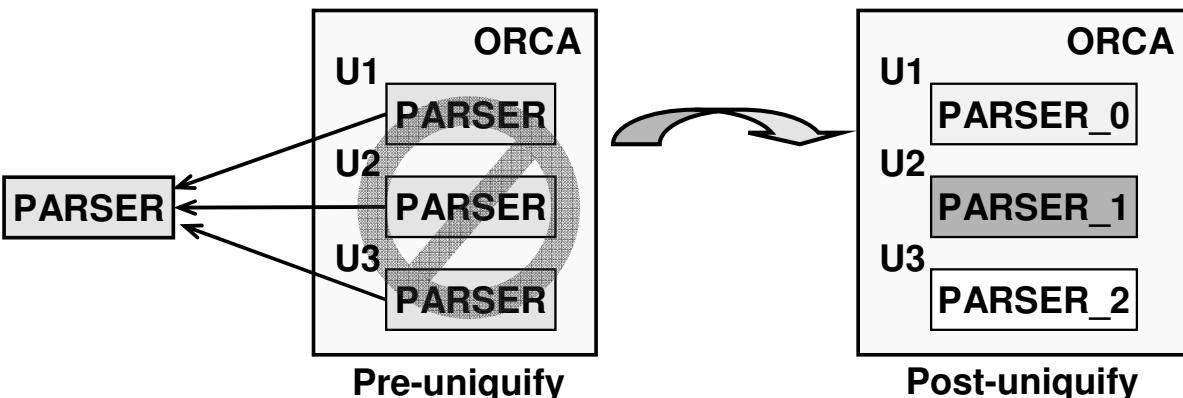
`uniquify` and `link`: Described on the next pages.

`save_mw_cel`: Creates the starting Milkyway *design cell* (view name CEL). Initially, prior to design planning and placement, the design CEL view consists of yellow rectangles, which represent all of the netlist leaf cells (standard, macro and IO pad cells), all stacked on top of each other at the origin. The design cell will undergo major “physical” changes during the physical design steps of design planning, placement, CTS, routing, etc. This command also saves the logic (db) library and TLU+ model information with the CEL. However, by default, this information is not recalled by ICC once the CEL is closed and later re-opened.

Must *Uniquify* Multiply Instantiated Designs

- IC Compiler does not support non-uniquified designs, i.e. designs with multiple instantiations!
- If incoming netlist is not *uniquified*, do so first!

```
current_design ORCA  
uniquify
```



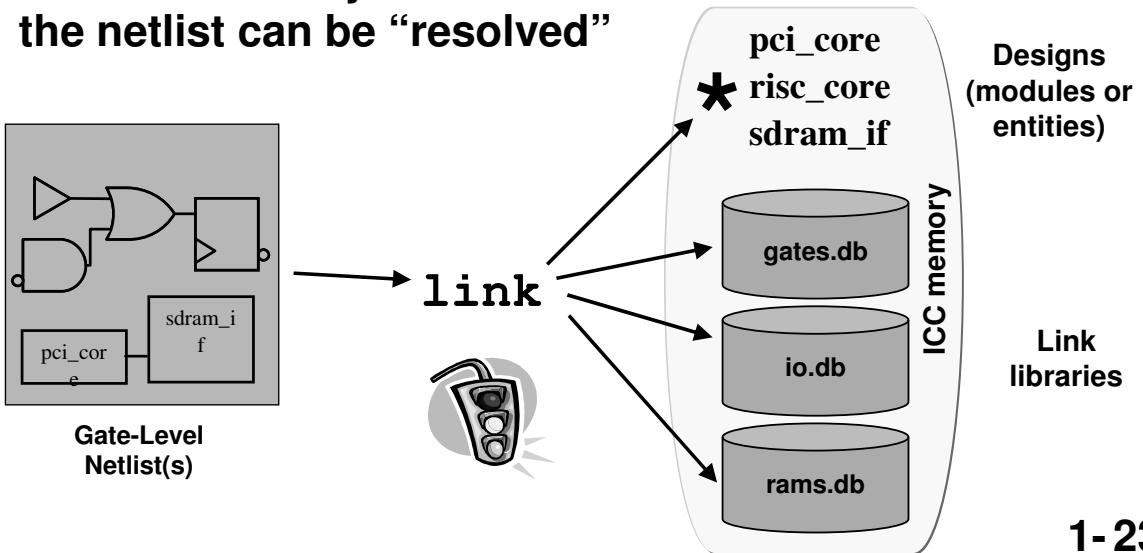
1-22

A *uniquified* design allows IC Compiler to optimize the logic within each instance individually, based on its unique environment (i.e. input drivers, output load, input data arrival times, output data required times, etc).

It is acceptable to always include the `uniquify` command in your script. If there are no multiply-instantiated sub-designs, the command does nothing.

Linking: Resolving References

- Gate-level netlists contain references to hierarchical sub-designs, as well as standard cells and macros, which are stored in the logical libraries
- The **link** command loads the specified link libraries into ICC memory and ensures that all references in the netlist can be “resolved”

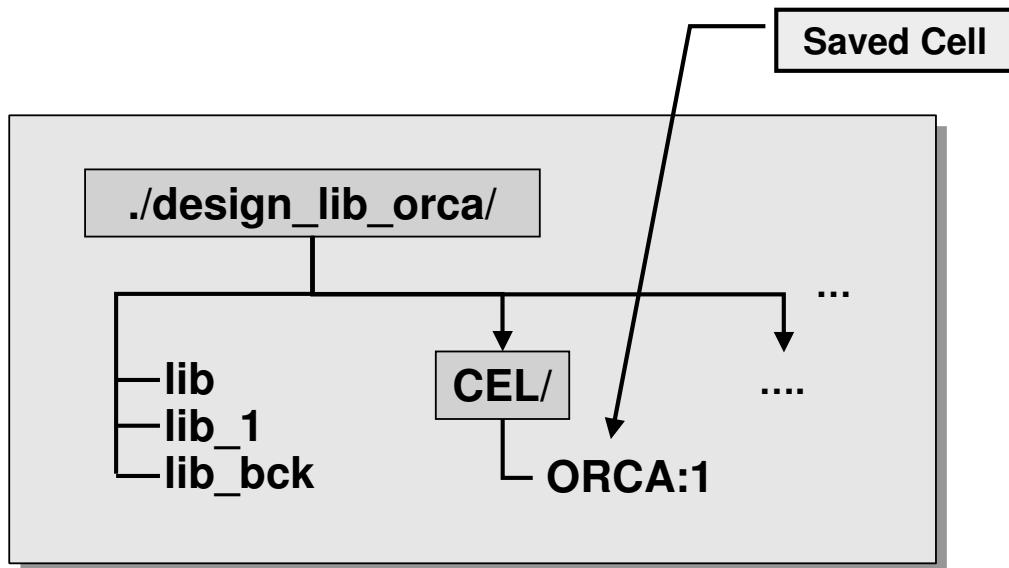


1-23

The asterisk (*) specified in the `link_library` variable represents IC Compiler's memory. Any designs that are read in with `read_verilog/ddc`, or `import_designs`, are stored in ICC memory. During `link`, ICC tries to “resolve” every instance in the netlist as follows: It first looks for referenced designs in memory (these would be sub-design that were read in prior to `link`). If not found in memory, ICC then looks for leaf cells in the specified list of link libraries. If all instances (or references) are resolved, `link` simply returns a “1”, signaling a successful link. If `link` is unable to resolve one or more references, warning messages are issued, allowing the user to fix the problem before proceeding.

Milkyway Design Library with Design Cell

The `save_mw_cel` command creates a new *CEL* view



1-24

5b. Shortcut: Import the Netlist

```
import_designs orca.v \
    -format verilog \
    -top ORCA
```

Format can be
verilog, db, ddc

Replaces:

```
read_verilog -netlist orca.v
current_design ORCA
uniqualify
link
save_mw_cel -as ORCA
```

1-25

By default, the command shown will read the netlist then save the design to a CEL named after the top design (ORCA in this example). If you want to choose a different name, use the “-cel” option.

6. Verify Logical Libraries Are Loaded

Ensure that all the required logical libraries (specified by `set link_library`) have been loaded

list_libs

- Note: This command can be executed only after reading and linking the netlist

Logical Libraries:

Library	File	Path
cb13fs120_tsmc_max	sc_max.db	/projects/XYZ_design/ref/db
cb13io320_tsmc_max	io_max.db	/projects/XYZ_design/ref/db
ram8x64_max	ram8x64_max.db	/projects/XYZ_design/ref/db
ram16x128_max	ram16x128_max.db	/projects/XYZ_design/ref/db
gtech	gtech.db	/global/apps3/icc_2008.09/libraries/syn
standard.sldb	standard.sldb	/global/apps3/icc_2008.09/libraries/syn

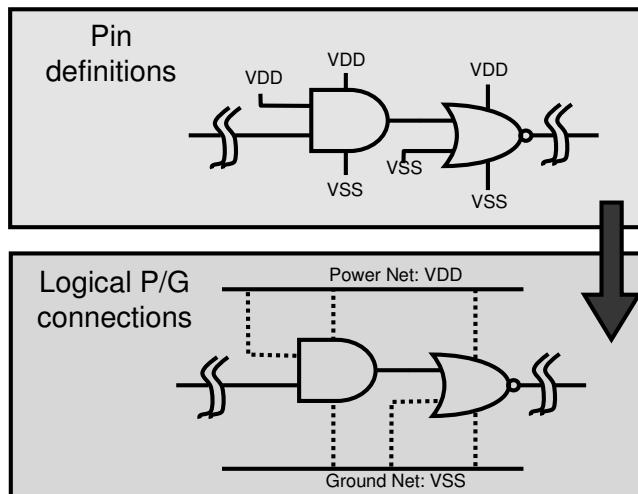
The *gtech* and *standard* libraries are generic libraries that are loaded by default – used during synthesis

1-26

7. Define Logical Power/Ground Connections

Define “logical connection” between P/G pins and nets

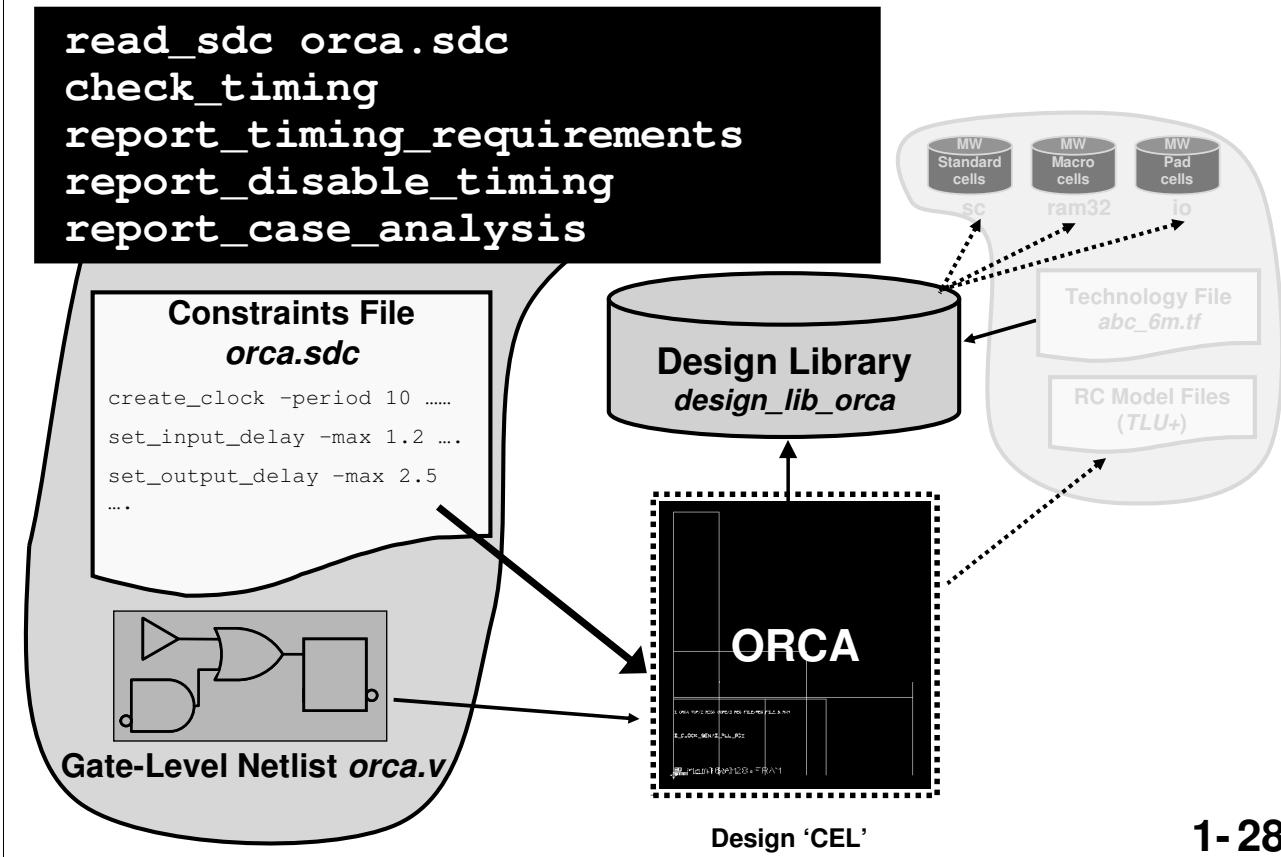
- Does not make any physical connections



```
derive_pg_connection -power_net VDD -power_pin VDD \
                      -ground_net VSS -ground_pin VSS
check_mv_design -power_nets
```

1-27

8. Apply and Check Timing Constraints



After applying timing constraints, it is recommended to invoke `check_timing` to ensure that the design is completely constrained. IC Compiler places, but will not optimize logic paths that are not constrained for timing. This check will flag any unconstrained paths - paths missing an input or output delay, or register to register paths where the clock is not defined. It does not check for missing external loads or drive characteristics!

The `report_timing_requirements` command can be used to check if the design contains any false or multicycle paths, or any asynchronous min- or max-delay constraints.

The `report_disable_timing` command reports disabled timing arcs in the current design. Paths containing one or more disabled timing arcs will not be optimized for timing. Timing arcs can be disabled by the user (`set_disable_timing`), or automatically by the tool during timing analysis in order to break timing loops or when propagating constants in the design.

The `report_case_analysis` command reports ports or pins that are set to a constant logic value 1 or 0 by the constraint `set_case_analysis`. Case analysis is a way to specify a given “mode” of a design without altering the netlist structure.

Timing Constraints

- “Timing Constraints” are required to communicate the design’s timing intentions to IC Compiler
- They should be the same ones used for synthesis with Design Compiler (preferably in SDC format)

Not needed if reading in a constrained *ddc* design

```
create_clock -period 10 [get_ports clk]
set_input_delay 4 -clock clk \
    [get_ports sd_DQ[*]]
set_output_delay 5 -clock clk
    [get_ports sd_LD]
set_load 0.2 [get_ports pdevsel_n]
set_driving_cell -lib_cell buf5 \
    [get_ports pdevsel_n]
...
```

SDC = Synopsys Design Constraints

1-29

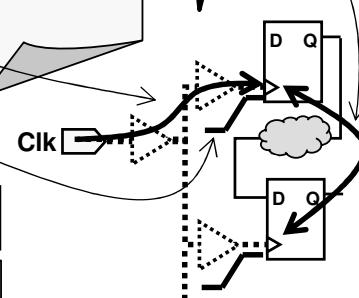
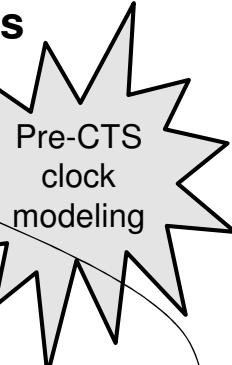
SDC format contains “pure” timing-related constraints – no optimization directives that are understood only by Design Compiler, e.g. set_ultra_optimization, set_optimize_registers, etc.

9. Ensure Proper Modeling of Clock Tree

- Ensure your SDC constraints model estimates of clock skew, latency and transition times for all clocks

report_clock -skew

Object	Rise Delay	Fall Delay	Min Rise Delay	Min Fall Delay	Uncertainty Plus	Uncertainty Minus
SYS_2x_CLK	0.80	0.80	0.40	0.40	0.10	0.20
SDRAM_CLK	-	-	-	-	0.10	0.15
Object	Max Transition Rise		Min Transition Rise			
SYS_2x_CLK	0.07	0.07	-	-		
SDRAM_CLK	0.07	0.07	-	-		



- Ensure no clocks are defined as “propagated” clocks

report_clock

Clock	Period	Waveform	Attrs	Sources
SDRAM_CLK	7.50	{0 3.75}	p	{sdram_clk}
SYS_2x_CLK	4.00	{0 2}		{sys_2x_clk}

1-30

Uncertainty (plus or minus) models the difference in arrival times of the clock signal at the clock pins of flip-flops, due to skew, jitter, or margin. *Minus uncertainty* reduces a path's maximum allowable delay time by being subtracted from the capturing setup clock edge. *Plus uncertainty* increases a path's minimum required delay time by being added to the capturing hold clock edge. Uncertainty is modeled by the SDC command `set_clock_uncertainty`.

Delay (rise or fall, min or max) models the propagation or insertion delay through the clock network. *Max* delays are used for setup timing checks, and *min* delays for hold.

Propagation delay is modeled by the SDC command `set_clock_latency`.

Transition (rise or fall, min or max) models the transition times at the clock pin of the sequential device. *Max* transitions are used for setup timing checks, and *min* transitions for hold.

Transition is modeled by the SDC command `set_clock_transition`.

Propagated clock forces IC Compiler to calculate the ACTUAL propagated delays through the clock network, to find the actual skews, latencies and transitions. This is useful after clock tree synthesis (CTS). Before CTS there is no clock network, so IC Compiler should instead use the values defined by the clock network modeling commands listed above.

Propagated clocks are enabled by the SDC command `set_propagated_clock`, and if necessary can be removed (prior to CTS) by `remove_propagated_clock`.

Test for Understanding



- 1. Why does IC Compiler require logical libraries in addition to physical libraries?**
- 2. What is the difference between a Milkyway design library and reference library? What do they have in common?**
- 3. The shortcut `import_designs` includes reading in the timing constraints file. *True / False***
- 4. After importing a *Verilog* netlist it is not necessary to load an SDC file. *True / False***
- 5. Clock uncertainty**
 - a. Is used to model clock skew pre-CTS
 - b. Is used during CTS as the maximum skew constraint
 - c. Is used to model clock skew post-CTS
 - d. A and B

1-31

1. The logical libraries provide IC Compiler with logic functionality and timing information which is needed during placement, CTS and routing optimizations. This information is NOT stored in the physical libraries.
2. The design library acts as a container for ALL the design-related data used by IC Compiler. A reference library, such as a standard cell, IO pad cell, or macro/IP library, contains the physical footprints of the leaf cells that are instantiated or referenced in the netlist, and will be placed and routed together to form the final design layout. They are both Milkyway databases.
3. False. import_designs performs the following: `read_verilog/ddc, current_design, uniquely,` Verilog netlist you must use `read_ddc` to apply the constraints.
4. False. A Verilog netlist does not contain any constraints, so the constraints must be applied separately by reading in an SDC constraints file. If the synthesized design was saved in ddc format and then imported into IC Compiler, the constraints are included in the netlist and are not required to be read in separately.
5. A. CTS will try to minimize clock skew independent of set_clock_uncertainty, and after CTS, the actual "propagated" clock network delays are used to calculate the actual skew.

10. Apply Timing and Optimization Controls

- Timing and optimization in IC Compiler is controlled by many variables and commands, for example:

```
set timing_enable_multiple_clocks_per_reg true  
set_fix_multiple_port_nets -all -buffer_constants  
group_path -name INPUTS -from [all_inputs]
```

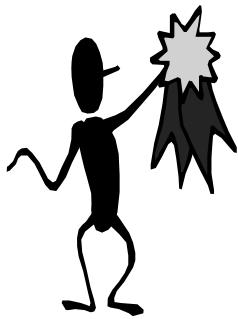
- These variables and commands can affect design planning, placement, CTS and routing
- Therefore, they should be applied prior to design planning
- Learning all the available variables and commands can be a challenge – the GUI provides help!

```
source tim_opt_ctrl.tcl
```

1-32

Available Timing and Optimization Controls

Use the GUI to find out what timing and optimization settings are available

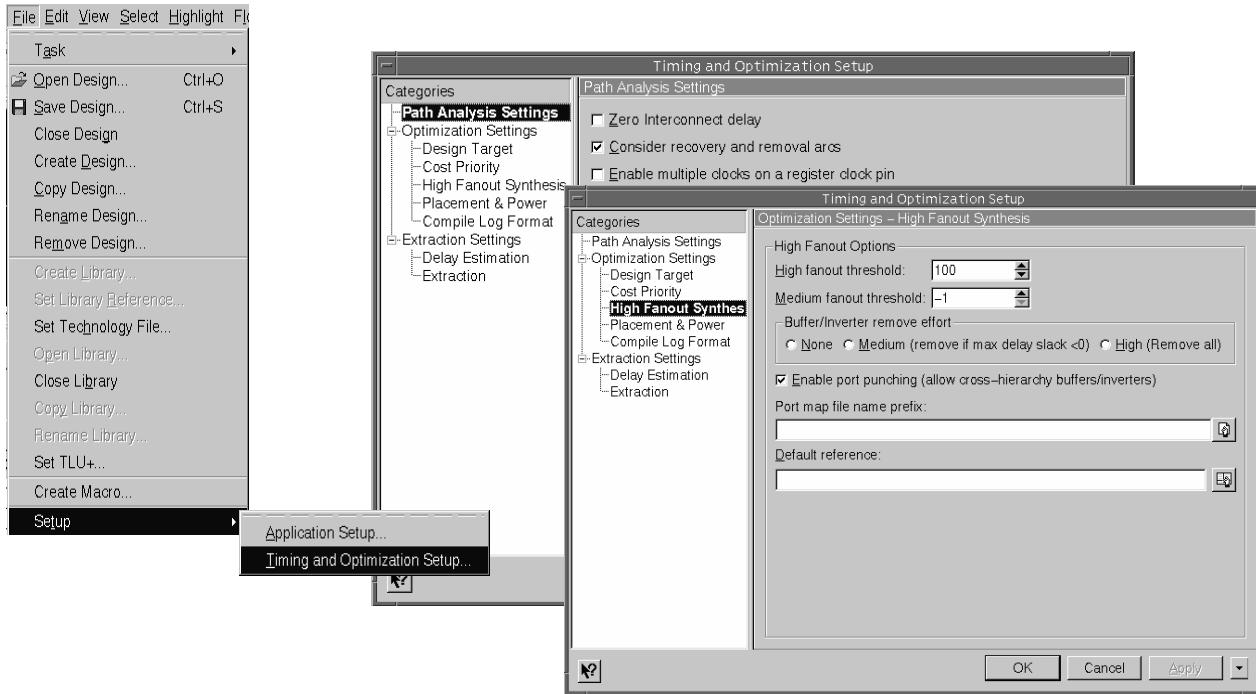


Use the *man* pages to learn more

Use the GUI to perform your initial setup, then copy the variables/commands into a *control setup file* for subsequent uses

1-33

Timing and Optimization Setup Example



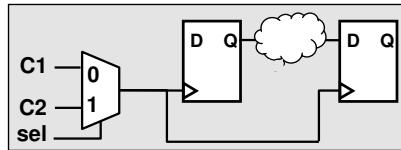
The following slides describe some common control settings

1-34

Enable Multiple Clocks per Register

- By default timing analysis (TA) will only consider one clock per register
 - May pick optimistic or incorrect launching and capturing clocks
- Always enable multiple clocks per register
 - Set *false paths* as needed
 - TA will now consider the correct clock timing

```
set timing_enable_multiple_clocks_per_reg true  
set_false_path -from [get_clocks C1] -to [get_clocks C2]  
set_false_path -from [get_clocks C2] -to [get_clocks C1]
```



Default: TA picks one of four possible clock combinations (see notes)

With the above settings: TA considers C1→C1 and C2→C2

1-35

By default, timing analysis (which happens during placement, CTS, routing and related optimizations) can only select one clock per register. In the example above, since the clock selection per register is independent of any other register and does not perform any “logic analysis”, the launching versus capturing clock may be any one of the following four possibilities, of which the middle two are logically false combinations:

C1→C1

C1→C2

C2→C1

C2→C2

With the above settings timing analysis considers both C1→C1 and C2→C2 and uses the faster clock for setup timing optimizations.

Note: The above settings are not needed if `set_case_analysis` is used instead, to define the value of the MUX select input. This will force timing analysis to use the “selected” clock pair. In the example above, with `set_case_analysis 1 [get_pins sel]` timing analysis will only consider C2→C2 timing.

Enable Constant Propagation

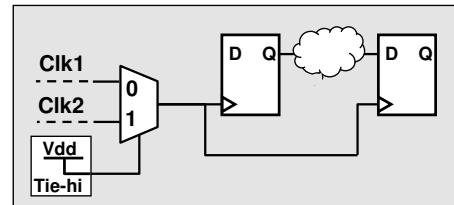
- By default IC Compiler will not propagate constants

- Control and other signals tied high/low are not considered during timing analysis (TA) – may result in incorrect TA

- Always enable constant propagation

```
set case_analysis_with_logic_constants true
```

By default: No constant propagation –
TA picks one of four clock combinations



With above setting: Constant
Propagation forces TA to select Clk2→Clk2

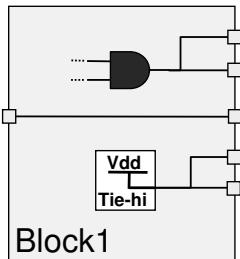
1-36

Note: If the above variable is false, constant propagation will occur on nodes that have set_case_analysis applied.

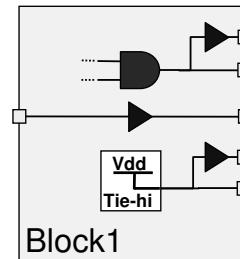
Enable Multiple Port Net Buffering

- By default IC Compiler will not buffer nets that are connected to two or more ports
 - Good design practice dictates that each port has a unique driver
- Always enable multiple port net buffering

```
set_fix_multiple_port_nets -all -buffer_constants
```



Default: Multiple port
nets without buffering



With multiple port
net buffering

1-37

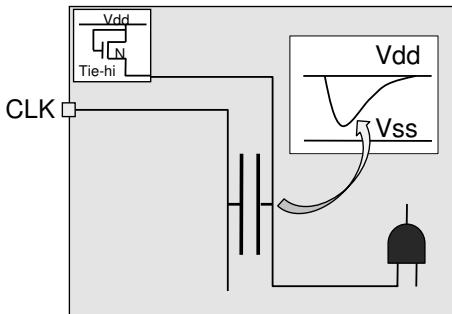
set_fix_multiple+port_nets arguments:

[-default]	(no fixing of multi-port nets)
[-all]	(fix feedthrough and multiple output port nets – not constants)
[-feedthroughs]	(fix feedthroughs)
[-outputs]	(fix multiple output port nets)
[-constants]	(duplicate constants driving multiple ports)
[-buffer_constants]	(buffer constants driving multiple ports)
[design_list]	

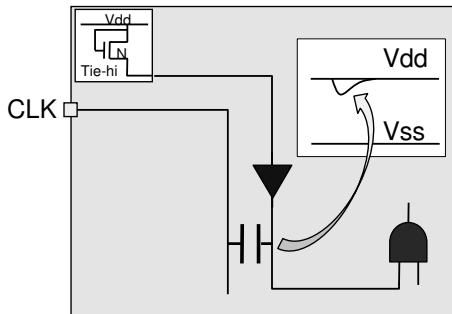
Enable Constant Net Buffering, if Needed

- By default `place_opt` will buffer all nets except:
 - Clock networks (done later by CTS)
 - Constant nets (tied high or low)
- Un-buffered constant nets may be susceptible to crosstalk
- Constant net buffering can be enabled with:

```
set_auto_disable_drc_nets -constant false
```



Default tie-high net



Tie-high net with buffering

1-38

By default, “design rule constraint” (DRC) checking and fixing is disabled for clock and constant nets, which in turn means that these nets are not buffered, during `place_opt`. Buffering on constant nets can be enabled with the above command. These are the DRCs – they govern net buffering: `max_transition`, `max_capacitance` and/or `max_fanout`.

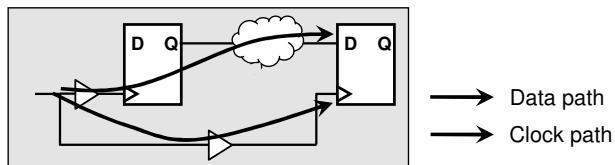
In the example above the buffered constant net is much less susceptible to large crosstalk glitches because the cross-coupling capacitance is greatly reduced due to the shorter wire length, and the net is being driven by an active buffer, which can usually maintain and recover a cross-coupled signal better than a passive tie-high cell.

Apply Timing Derating for On-Chip Variation

- By default timing analysis uses the same PVT operating conditions for data as well as clock paths
 - Local PVT variations are not taken into account
 - May lead to optimistic timing results
- It is recommended to apply *some* timing derating to the data path - ask your vendor what percentage to use

```
set_timing_derate -max -early 0.95
```

This example speeds up the clock path by 5% during setup analysis under maximum PVT operating conditions



1-39

This command is used to add some “timing margin”, which means to reduce the required setup time and/or increase the required hold time. The `-late` option specifies how late a signal (data or clock path) could arrive. The derate factor specified with the `-late` option multiplies data path delays for setup checks and clock path delays for hold checks. The derate factor specified with the `-early` option specifies how early the signal could arrive, and therefore multiplies data paths for hold checks and clock paths for setup checks.

The `-max` option specifies that the derate value is to be applied to setup checks using the maximum PVT operating condition. The `-min` option specifies that the derate value is to be applied for hold checks using the minimum PVT operating condition. If neither the `-min` nor `-max` option is specified, the derate value is applied to both setup and hold, for worst and best case operating conditions, respectively. PVT = Process, Voltage and Temperature.

The `-data` option indicates that the derate value is to apply to elements on the data path only. The `-clock` option indicates that the derate value is to apply to elements on the clock path only. If neither the `-clock` or `-data` option is specified, the derate value is applied to both clock and data paths.

Define “Don’t Use” or “Preferred” Cells

- By default all cells available in the library can be used during buffering and optimization
- You may want to restrict the use of specific cells, for example:
 - Big drivers (EM issues) and/or very weak drivers
 - Delay cells and/or clock cells

```
set_dont_use <off_limit_cells>
```

- You may prefer the use of certain cells, when possible:
 - Specific buffers preferred for hold time fixing

```
set_prefer -min <hold_fixing_cells>
```

1-40

EM = Electro-migration

Keep Spare or Unloaded Cells

- By default any cell with unconnected output(s) will be deleted by IC Compiler
- If you need to keep such cells, *spare cells*, for example, you can turn off auto-deletion:

```
set physopt_delete_unloaded_cells false
```

1-41

EM = Electro-migration

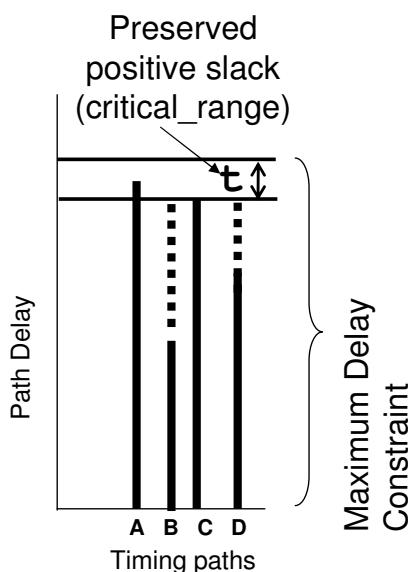
Apply Area Constraint for Area Recovery

- Area recovery takes advantage of paths with positive timing slack to reduce or “recover” area
 - Through cell down-sizing and buffer removal
- Area recovery can help to reduce congestion and power consumption and is recommended during placement (`place_opt -area_recovery`)
- To maximize area recovery it is recommended to apply a zero maximum area constraint

```
set_max_area 0
```

1-42

Apply a Power and Area Critical Range



Paths B and D can be slowed down to optimize for leakage power and/or area.
Paths A and C will be left alone.

- **Area and power optimization take advantage of paths with positive timing slack**

- Slack can be reduced to zero, by default
- Zero slack or margin may cause some near-critical nets to violate timing later, during CTS or routing

- **Apply a “critical range” to force optimization to preserve some positive timing slack**

- Margin may reduce timing violations later
- Guideline: Use ~10% of the smallest clock period

```
set physopt_power_critical_range <t>
set physopt_area_critical_range <t>
```

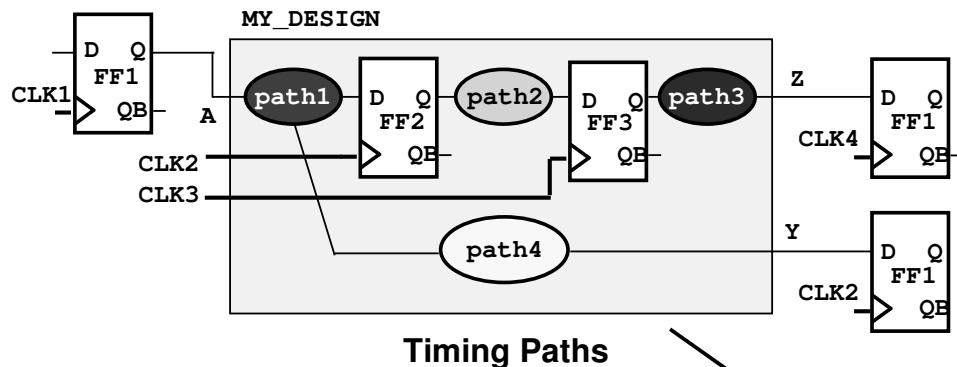
1-43

Note: The term “critical range” as it refers to power and area optimization above is very different than timing “critical range”.

When the `physopt_power/area_critical_range` variables above are applied, fewer paths are optimized for power/area, respectively, which may help timing in later phases of P&R.

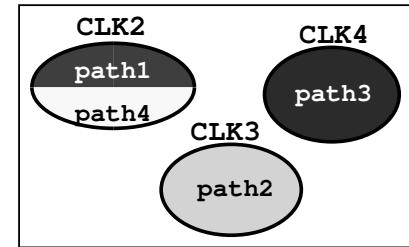
When a timing critical range is set, more paths are optimized for timing (discussed later).

IC Compiler Organizes Paths into Groups



Timing Paths

- **Paths are grouped by the clocks controlling their endpoints**
 - Path groups inherit the name of the related end-point clock, by default
- **IC Compiler optimizes each path group in turn, starting with the critical path in each group**



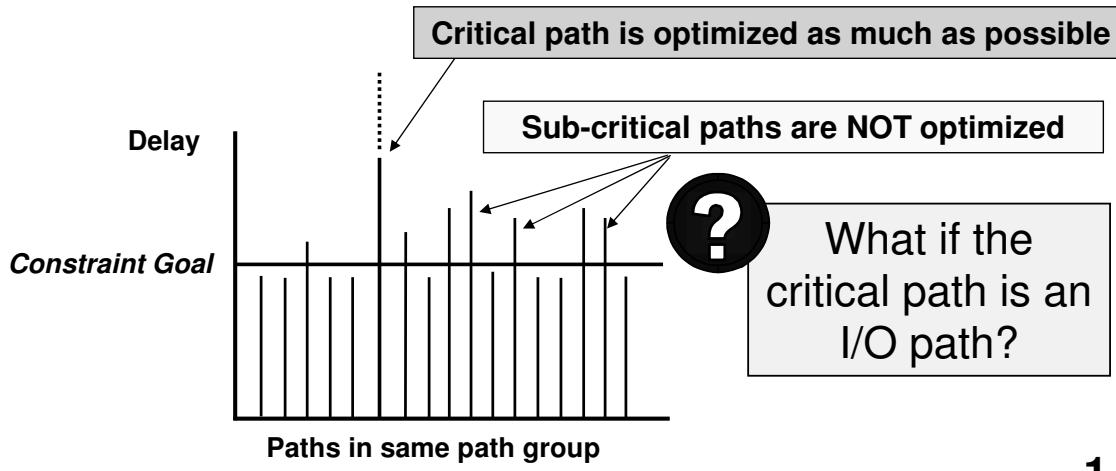
Path Groups

1-44

`report_path_group`: Reports the path groups which were defined in the current design.

General Problem: Sub-Critical Paths Ignored

- By default, optimization within a path group stops when:
 - All paths in the group meet timing, or
 - IC Compiler cannot find a better optimization solution for the critical path – it reaches a point of diminishing returns
 - Sub-critical paths are not optimized - to save run time!



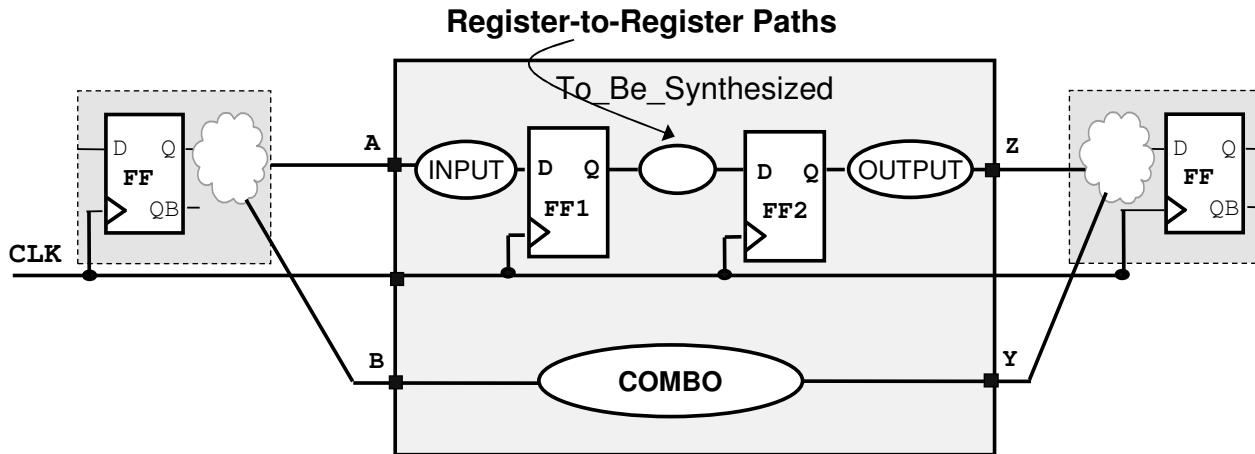
1-45

ANSWER:

See the next slide ...

Serious Problem: Reg-to-Reg Paths Ignored

In a sub-block with poor logic hierarchy partitioning, the I/O constraints are usually over-constrained, to be conservative



The critical path will most likely be in the I/O logic



What happens to reg-to-reg path violations if IC Compiler gives up on the I/O “critical path”?

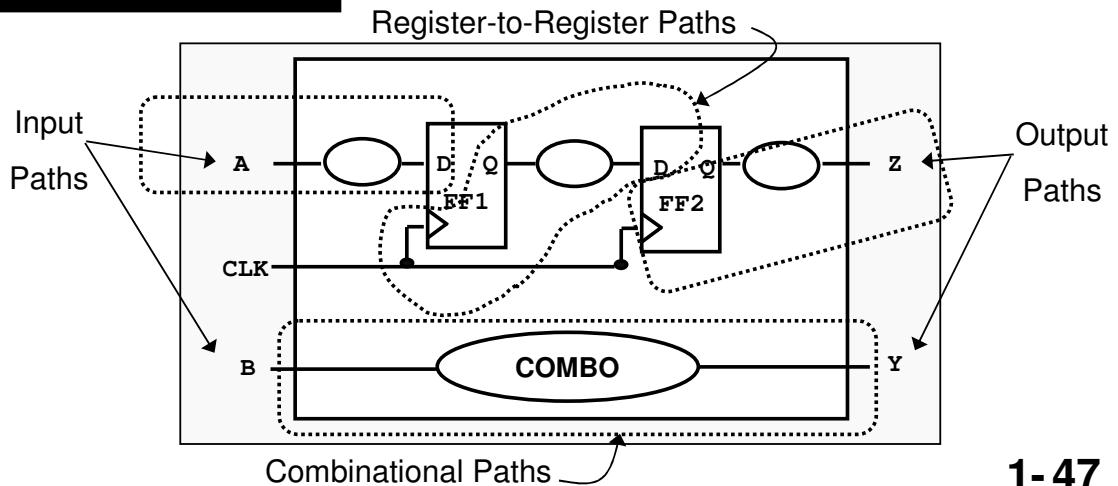
1-46

ANSWER: Since this is a single-clock design, all paths are in the same path group. If optimization gives up on the critical path, no additional optimization is performed on the less critical paths, including the reg-to-reg paths. The reg-to-reg paths are very accurately constrained (since they are only dependent on the clock waveform, which is usually well defined), yet they will be totally ignored due to the poor I/O constraints. This is not good.

Solution: User-Defined Path Groups

- Custom *path groups* allow more control over optimization
 - Each *path group* is optimized independently
 - Worst violator in one path group does not prevent optimization in another group
- Check if user-defined path groups exist for I/O paths

`report_path_group`



1-47

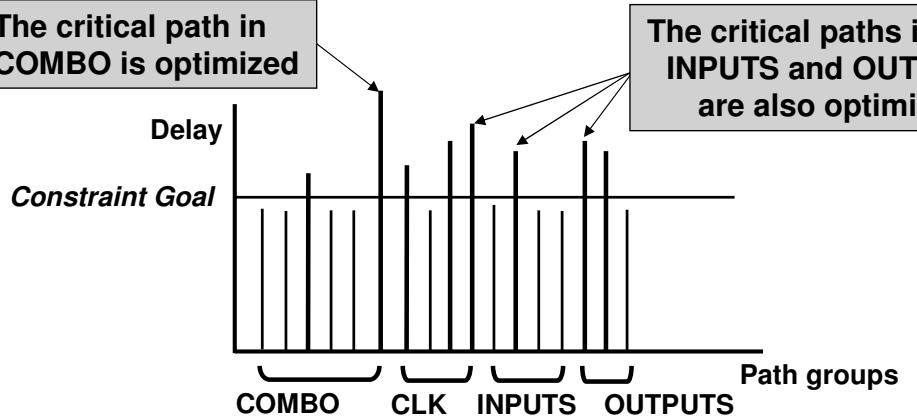
Creating user-defined path groups can also facilitate a divide-and-conquer timing analysis strategy, since `report_timing` reports each path group's timing separately. This can help you isolate or analyze problems in a certain region of your design.

Define Path Groups for I/O Paths, if needed

```
# Ensure that the reg-reg paths get optimized  
group_path -name INPUTS -from [all_inputs]  
group_path -name OUTPUTS -to [all_outputs]  
group_path -name COMBO -from [all_inputs] -to [all_outputs]
```



Where are the reg-to-reg paths?
Are the COMBO paths in three path groups?



1-48

The reg-to-reg paths remain in the CLK path group, which is created by default by IC Compiler. The group_path command can be used for CLK to assign a non-default weight or *critical range* (to be discussed) to the group, but is not needed not to define the reg-to-reg paths.

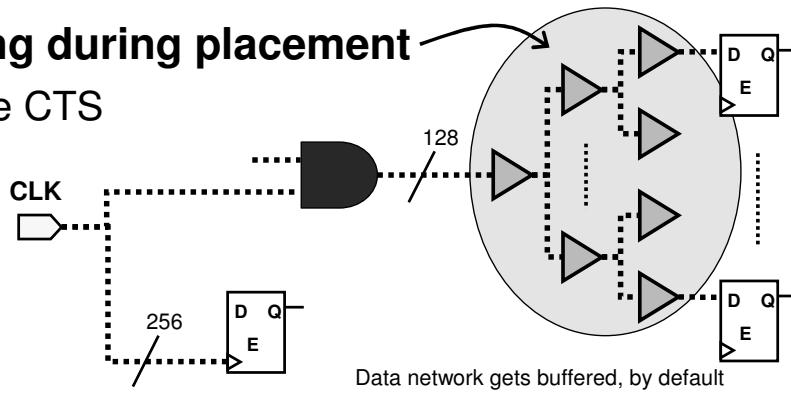
A path can only be in one path group, but according to the path group arguments, the combo paths can be part of the INPUTS, OUTPUTS or COMBO group – so where are they?

The COMBO paths wind up in the COMBO group. Assuming the commands are executed in the order listed in the slide, they are first moved from CLK to INPUTS, because they match the startpoint argument `-from [all_inputs]`. They will not be moved to the OUTPUTS group, even though their endpoints match `-to [all_outputs]`, because “`-from`” has priority over “`-to`”. They finally end up in the COMBO group because their startpoints **and** endpoints match the `-from AND -to` arguments. IC Compiler works this way to prevent having different results if the command sequence changes! The results are independent of the order of the commands above.

Use `report_path_group` to get a summary of the path groups in the design.

Prevent Buffering of Clock-as-Data Networks

- IC Compiler automatically treats clock signals driving clock pins as “ideal”
 - No buffering until CTS
- Clock signals that are used as “data” are not ideal
 - May be buffered during placement if violating DRCs or timing
- Prevent buffering during placement
 - Remove before CTS

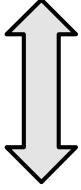


```
set_ideal_network [all_fanout -flat -clock_tree]
```

1-49

Modify Optimization Priority if Needed

- Netlist optimization is governed by “cost priorities”
 - If competing constraints can not all be met, higher priority constraints will be met at the expense of the lower ones
- Default cost priorities:

	Cost Type	Constraints
Highest 	Design Rule	Max Transition; Max Fanout; Max/Min Capacitance
	Delay (max)	Clock; Max Delay
	Delay (min)	Clock; Min Delay
	Dynamic Power	Max Dynamic Power
	Leakage Power	Max Leakage Power
	Area	Max Area

- The cost priorities can be modified, for example:

```
set_cost_priority -delay; # See notes below
set_cost_priority {max_transition max_delay}
```

1-50

set_cost_priority -delay: Max delay constraints are given higher priority than max design_rule constraints.

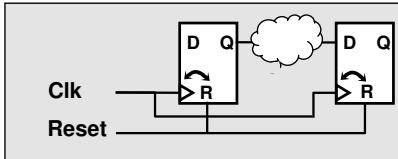
set_cost_priority {max_transition max_delay}: Max transition is higher than max delay, but max capacitance and max fanout (and all the others) are lower than max delay (in the default order).

```
set_cost_priority
    [-default]
    [-delay]
    cost_list
    [-design_rules]
    [-firm_area_limit]
    [-min_delay]
```

cost_list specifies a list of costs in decreasing order of priority, selected from the following: max_delay, min_delay, max_transition, max_fanout, max_capacitance, cell_degradation, and max_design_rules. Any costs that are not in the list are assumed to follow afterwards ,in their default relative priority.

Enable Recovery and Removal Timing Arcs

- Recovery and removal timing arcs define timing constraints between asynchronous signals of a sequential device (*set*, *reset*) and its *clock*
 - If applicable, these constraints are included in the library



- By default IC Compiler ignores these constraints
- By default *Primetime* checks these constraints
- To enable IC Compiler to check and optimize for recovery and removal constraints, and for consistency with *Primetime*:

```
set enable_recovery_removal_arcs true
```

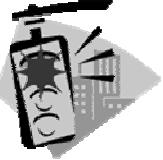
1-51

Primetime is Synopsys' stand-alone static timing analysis tool, used for "sign-off".

Recovery or removal timing arcs impose constraints on asynchronous pins of sequential cells. Typically, recovery time specifies the time the inactive edge of the asynchronous signal has to arrive before the closing edge of the clock. Removal time specifies the length of time the active phase of the asynchronous signal has to be held after the closing edge of the clock.

11. Perform a ‘Timing Sanity Check’

- Before starting placement it is important to ensure that the design is not over-constrained
 - Constraints should match the design’s specification
- Report ‘ZIC’ timing before placement
 - Check for unrealistic or incorrect constraints
 - Investigate large zero-interconnect timing violations



```
set_zero_interconnect_delay_mode true
Warning: Timer is in zero interconnect delay mode. (TIM-177)
report_constraint -all
report_timing
set_zero_interconnect_delay_mode false
Information: Timer is not in zero interconnect delay mode. (TIM-176)
```

1-52

What should you see in a “zero interconnect” timing report?

A positive slack for all paths means that the constraints have passed the check.

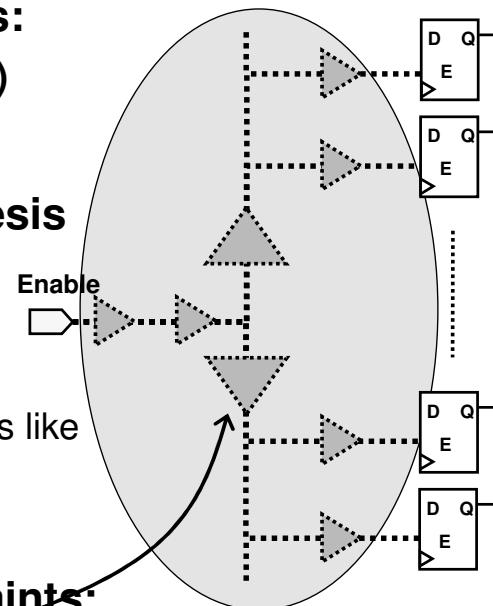
Small negative slacks are probably OK as well, since these can be fixed by IC Compiler’s advanced optimization engines.

You are looking for large violations. If for example a path shows a timing violation as large as the clock period, then you should investigate whether the constraints were applied correctly, or whether the design was not synthesized with the correct constraints. Large violations will most likely NOT be able to be fixed by IC Compiler, so the problem must be addressed before beginning placement.

12. Remove Unwanted “Ideal Net/Networks”

- Your SDC constraints may contain either of the following commands:
 - `set_ideal_network` (preferred)
 - `set_ideal_net` (obsolete)
- These commands prevent synthesis (Design Compiler) from building buffer trees on specified signals, which is deferred to the physical design phase (typically high fanout nets like `set/reset`, `enable`, `select`, etc.)
- To allow buffering during placement remove the constraints:

```
remove_ideal_network [get_ports Enable Select Reset]
```



1-53

13. Save the Design

It's good practice to save the design after each key design phase, for example: data setup, design planning, placement, CTS and routing:

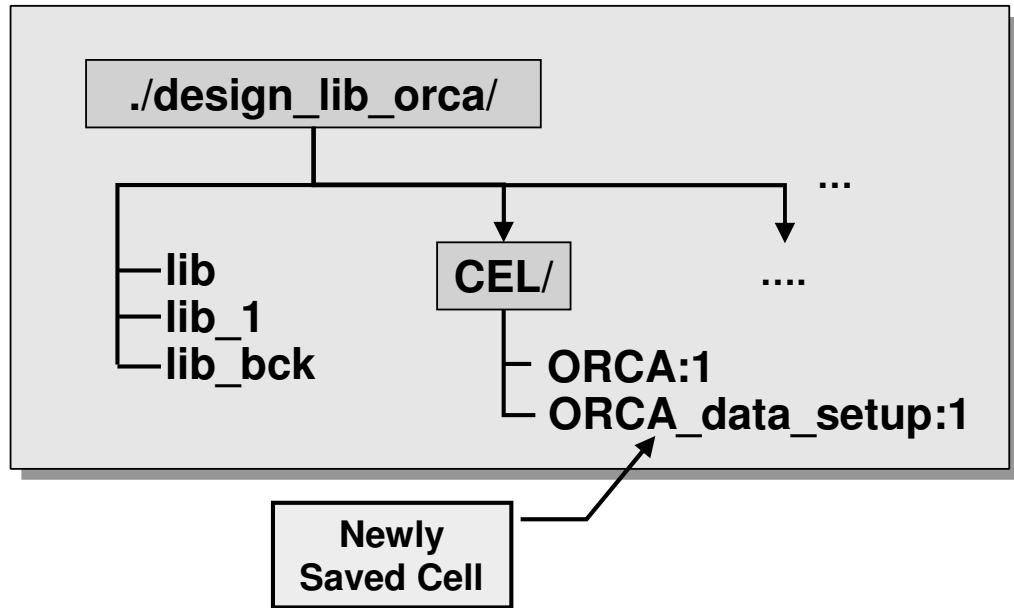
```
save_mw_cel -as ORCA_data_setup
```

- Note: The open cell is still the original *ORCA* cell !!

1-54

Unlike conventional software behavior, when you perform a “Save As” in IC Compiler, the “current open design” is not automatically switched to the new design CEL view – it remains the originally opened CEL.

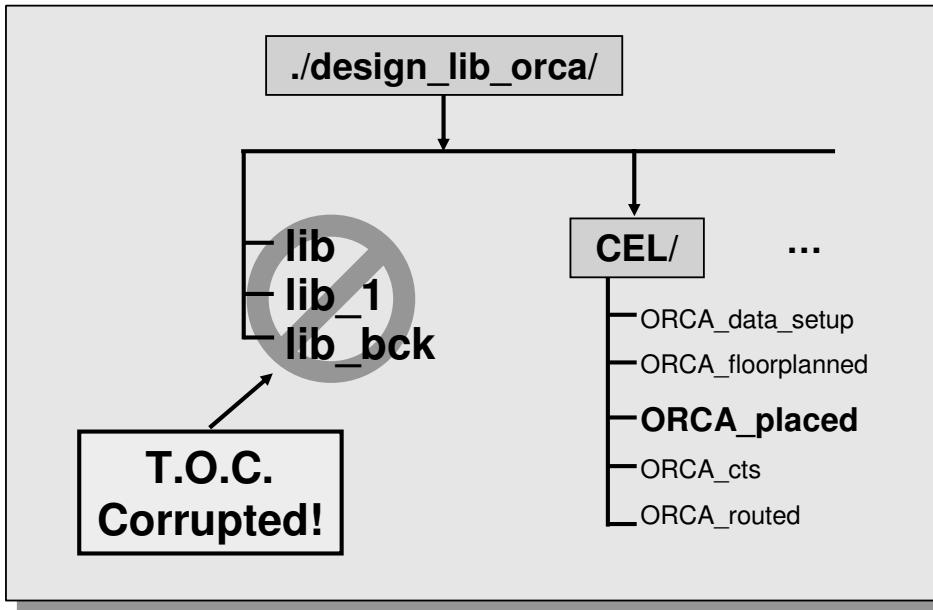
Design Library with New Design Cell



1-55

UNIX Manipulation of a Milkyway Database

```
UNIX% cd CEL
UNIX% rm ORCA_placed
UNIX% cp ~Joes_Lib/ORCA_placed
```



1-56

In the example above, the UNIX *rm* and *cp* commands were used to replace our CEL view *ORCA_placed* with Joe's version. Doing so will corrupt the database, more specifically the *lib* files, which contain the binary Table of Content (T.O.C.) of the library database. Likewise the *mv* and *mk_dir* commands should not be used to manipulate the directory and file structure under the library directory (example: *design_lib_orca*).

Instead, use the commands *rename_mw_cel*, *copy_mw_cel*, *remove_mw_cel*. They are Milkyway-aware, UNIX is not.

Note: It is acceptable to copy (*cp -r*) or delete (*rm -r*) the entire library directory and its contents.

If files are deleted, it is possible to rebuild the table of contents using *rebuild_mw_lib*.

Restoring Variables

- **Variable settings are maintained during the same ICC session but are NOT saved with the design CEL**
 - If you close a cell and re-open another cell in the same ICC session, previously applied variables settings are maintained
 - If you exit and later re-invoke ICC all previous variable settings are lost - they are set to their defaults
- **Suggestion: Include all setup variable settings in the .synopsys_dc.setup file, which is read in automatically when ICC is invoked**

```
set timing_enable_multiple_clocks_per_reg true  
set_fix_multiple_port_nets -all -buffer_constants
```

Variable – not saved with cell

Command – attribute saved with cell

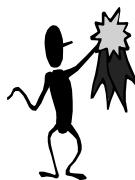
1-57

Restoring Logical Library and TLU+ Settings

- By default, *link_library*, *search_path*, *target_library* and *TLU+ settings* are stored with the CEL
- However, when you re-open the CEL, the stored settings are not re-applied, unless you set:

```
set auto_restore_mw_cel_lib_setup true  
open_mw_cel ORCA_placed
```

- Recommendation: Include in *.synopsys_dc.setup*
- If library or TLU file locations change with respect to the Milkyway library location, the settings have to be re-applied. See *notes section below!*



1-58

1st bullet: Whether settings are stored with the design CEL or not is controlled through the variable `save_mw_cel_lib_setup`. This variable is set to `true` by default.

2nd bullet: When opening a CEL with the variable `auto_restore_mw_cel_lib_setup` set to `false` (the default setting), IC Compiler will NOT restore the settings. The user will need to re-apply them when re-invoking IC Compiler, after exiting the current session.

When opening a CEL with the variable `auto_restore_mw_cel_lib_setup` set to `true`, IC Compiler will restore the settings and display the following:

Information: AUTO-RESTORE: Setting variable `search_path` to `.. /path/to/icc/dw/sim_ver .. /ref/db.` (UIG-10)

Information: AUTO-RESTORE: Setting variable `link_library` to `* sc_max.db ... ram16x128_max.db.` (UIG-10)

Information: AUTO-RESTORE: Setting variable `target_library` to `sc_max.db.` (UIG-10)

Information: AUTO-RESTORE: Setting variable `itf_tlu_plus_library` to `.. /ref/tlup/abc_6m_max.tlupplus....` (UIG-10)

If the library file locations change, follow this procedure:

```
close_mw_lib  
set auto_restore_mw_cel_lib_setup false  
set search_path "new search paths"  
set link_library "* link libraries"  
set target_library "target libraries"  
set_mw_lib_reference -mw_ref "new locations" design_lib_name  
set_mw_technology_file -tech "new location" design_lib_name  
open_mw_cel <cell name> -lib <design lib name>  
link  
set_tlu_plus_files -max_tlu <new_location> -min_tlu <new..> -tech2itf <>  
set auto_restore_mw_cel_lib_setup true
```

Loading an Existing Cell After Exiting ICC

If your `.synopsys_dc.setup` file contains the appropriate variables, then loading an existing cell is simple!

`.synopsys_dc.setup`

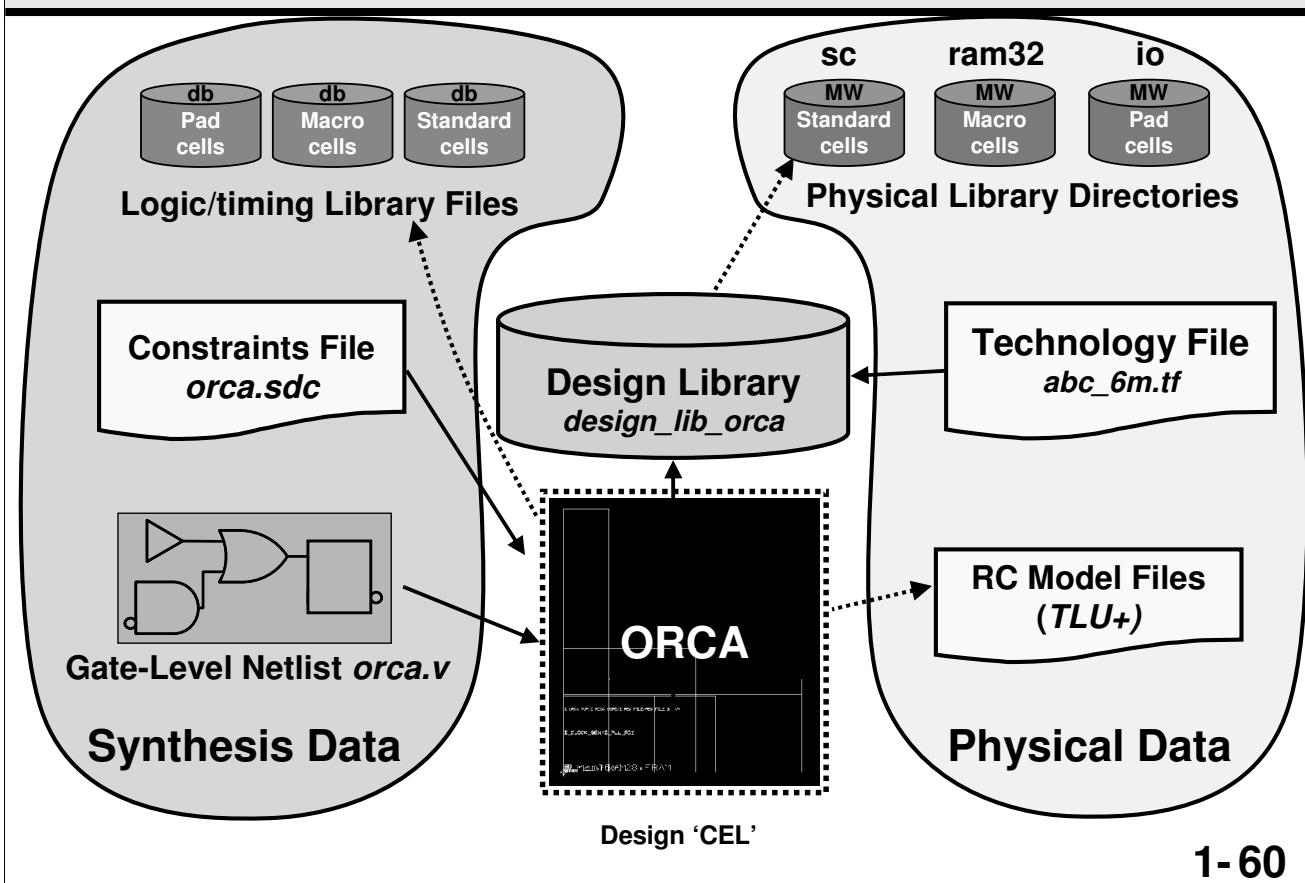
```
lappend search_path ./scripts ./design ...
lappend search_path [glob $MW_libs/*/LM]
set symbol_library "sc_icon.sdb io_icon.sdb"
set link_library "* sc_max.db io_max.db ram16x128_max.db"
set target_library "sc_max.db"
set mw_logic0_net "VSS"1
set mw_logic1_net "VDD"
set auto_restore_mw_cel_lib_setup true
...
```

```
UNIX% icc_shell -gui
icc_shell> open_mw_lib design_lib_orca
icc_shell> open_mw_cel ORCA_setup
icc_shell> link
```

Good practice – verifies correct library setup

1-59

Data Setup Summary



Data Setup Example (1 of 3)

.synopsys_dc.setup

```
lappend search_path ./scripts ./design_data $MW_libs
lappend search_path [glob $MW_libs/*/LM]
set symbol_library "sc_icon.sdb io_icon.sdb"
set link_library "* sc_max.db io_max.db ram16x128_max.db"
set target_library "sc_max.db"

set mw_logic0_net "VSS"
set mw_logic1_net "VDD"

# Timing and optimization control variables
set timing_enable_multiple_clocks_per_reg true
set case_analysis_with_logic_constants true
set physopt_delete_unloaded_cells false
set physopt_power_critical_range <t>
set physopt_area_critical_range <t>
set enable_recovery_removal_arcs true

set auto_restore_mw_cel_lib_setup true
```

1-61

The *symbol library* is an optional library that is used by the schematic GUI to display the logic symbols of the gates in the schematic. If not specified, IC Compiler uses default (rectangular) symbols for the gates.

Data Setup Example (2 of 3)

tim_opt_ctrl.tcl

```
set_false_path -from <clock_name> -to <clock_name>
set_fix_multiple_port_nets -all -buffer_constants
set_auto_disable_drc_nets -constant false
set_timing_derate -max -early 0.95
set_dont_use <off_limit_cells>
set_prefer -min <hold_fixing_cells>
set_max_area 0
group_path -name INPUTS -from [all_inputs]
group_path -name OUTPUTS -to [all_outputs]
group_path -name COMBO -from [all_inputs] -to [all_outputs]
set_ideal_network [all_fanout -flat -clock_tree]
set_cost_priority {max_transition max_delay}
```

1-62

Data Setup Example (3 of 3)

data_setup.tcl

```
create_mw_lib design_lib_orca -open -technology techfile.tf \
    -mw_reference_library "sc io ram32"
set_check_library_options -all
check_library
set_tlu_plus_files -max_tluplus cb13_6m_max.tluplus \
    -min_tluplus cb13_6m_min.tluplus \
    -tech2itf_map cb13_6m.map
check_tlu_plus_files
import_designs design.v -format verilog -top ORCA; # See below
list_libs
derive_pg_connection -power_net VDD -power_pin VDD \
    -ground_net VSS -ground_pin VSS
check_mv_design -power_nets
read_sdc constraints.sdc; # Omit if importing constrained ddc
check_timing; report_timing_requirements
report_disable_timing; report_case_analysis
report_clock; report_clock -skew
source tim_opt_ctrl.tcl; # Timing and optimization controls
set_zero_interconnect_delay_mode true
report_constraint -all; report_timing
set_zero_interconnect_delay_mode false
remove_ideal_network [get_ports Enable Select Reset]
save_mw_cel -as ORCA_data_setup
```

1-63

import_designs design.v -format verilog -top ORCA is equivalent to:

```
read_verilog -netlist design.v
current_design ORCA
uniquify
link
save_mw_cel -as ORCA
```

The example above imports a Verilog netlist file, which does not contain constraints. Alternatively, if a *ddc* file, created by Design Compiler and containing timing constraints is imported, the *read_sdc* step can be omitted.

The *tim_opt_ctrl.tcl* file contains the necessary timing and optimization control variable settings and commands, which are used during placement, CTS and routing.

Test for Understanding (1 of 2)



1. What's one benefit of buffering tie-high/low nets?
2. What cells might you want to put a "don't touch" on before placement?
3. Area recovery can:
 - a. Reduce congestion
 - b. Increase delay
 - c. Reduce power consumption
 - d. A and C
 - e. A, B and C
4. During *Data Setup* it is recommended to apply an "ideal network" on high-fanout nets (*enable*, *reset*), and to remove "ideal network" from the clock network.

True or False?

1-64

1. Buffered constant nets are much less susceptible to crosstalk glitches.
2. Very strong or very weak drivers, cells reserved for delay lines or clock trees.
3. Area recovery can down-size cells and/or remove buffers, which helps to reduce both leakage and dynamic power. Buffer removal can help to reduce congestion. Area recovery works by slowing down timing paths with positive slack.
4. **False.** Exactly the reverse is true: Remove ideal networks from high fanout nets so that they can be buffered during placing.

Test for Understanding (2 of 2)



5. Applying a power or area critical range can help to reduce power consumption or cell area, respectively.

True or False?

6. Defining path groups for I/Os is recommended if the I/Os are conservatively constrained. *True or False?*

7. Why is it recommended to remove the “ideal network” property from high fanout nets like *enable* and *reset after* performing a “zero-interconnect” timing sanity check, and not before?

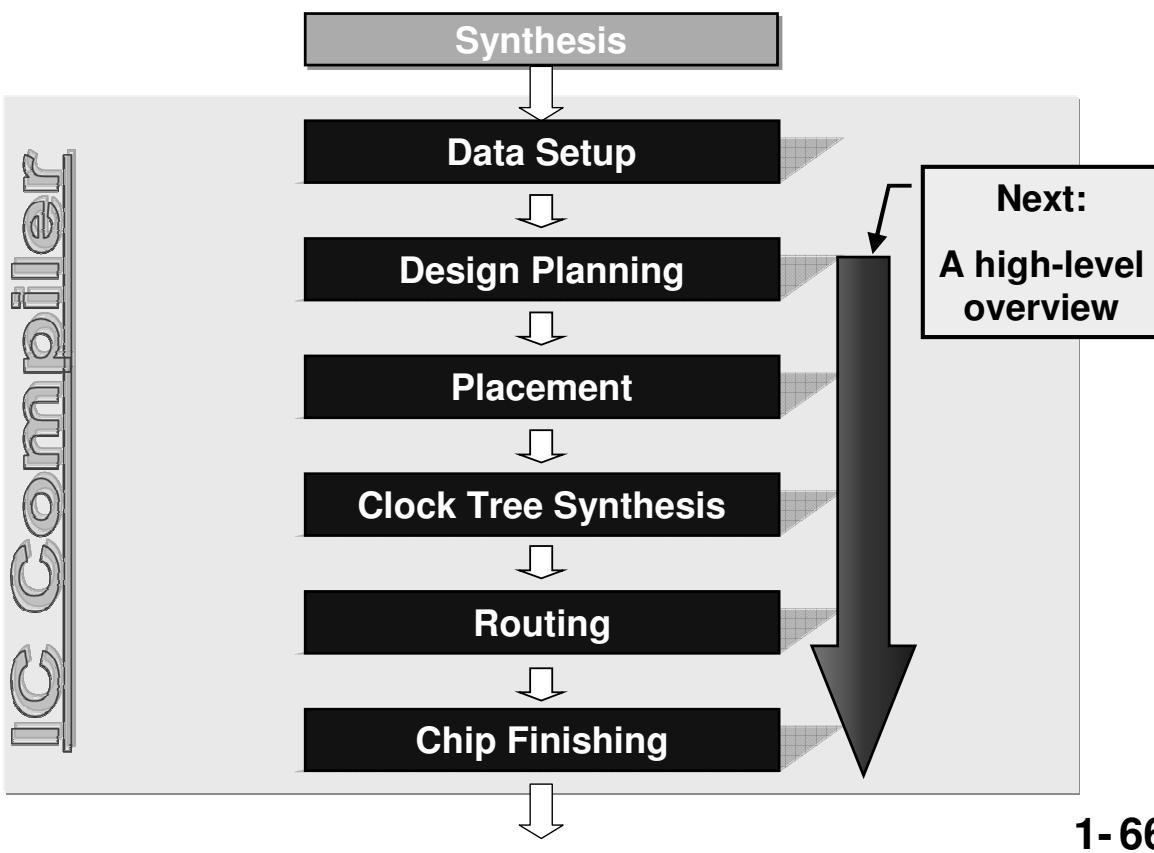
1-65

5. Applying a critical range forces a larger timing margin (slack) during optimization, so that near-critical paths do not become violating critical paths later during CTS or routing. This can result in increased power consumption and/or cell area.

6. True.

7. If done before ZIC timing, even though the net parasitics are ignored, these high fanout nets may exhibit unrealistically large transition times, due to the large total pin capacitance that these nets fan out to, which could generate unrealistically large timing violations.

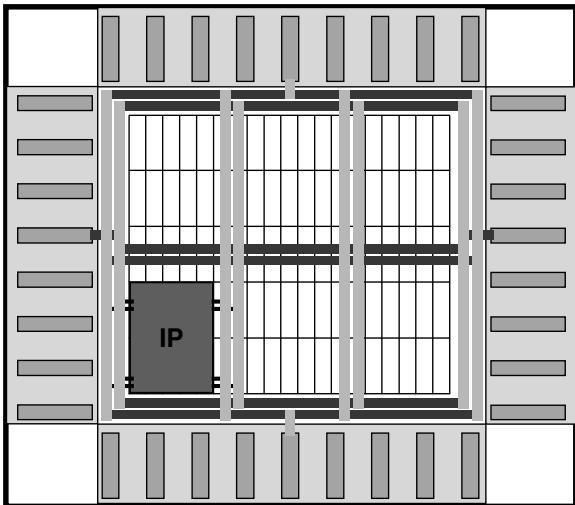
General IC Compiler Flow



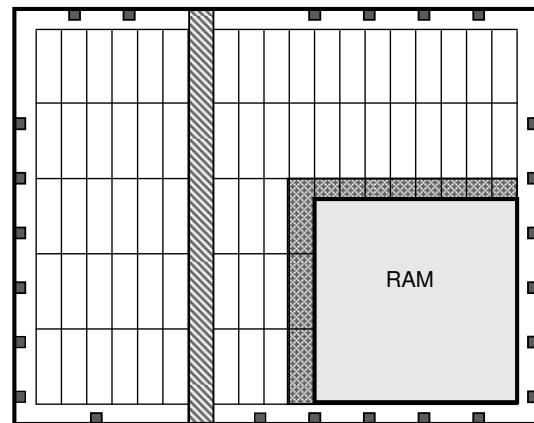
We will briefly introduce design planning as well as the “core” commands for Placement, CTS and Routing, for the purposes of performing Lab 1, which takes you through the IC Compiler flow (at a high level). In the subsequent Units you will delve into each step in much more detail.

Design Planning

Data Setup Design Planning Placement Clock Tree Synthesis Routing Chip Finishing



Chip-level Floorplan



Block-level Floorplan

“Design planning” is the design flow phase where the floorplan is defined

1-67

A floorplan defines: IO, filler and corner pad locations, or pin locations around the periphery; size and shape of standard cell core or placement area; placement of macro cells; placement and routing blockages; the power grid, which includes all pre-routed VDD/VSS metal, including core and periphery rings, vertical and horizontal straps, and standard cell rails.

Load an Existing Floorplan

- Read in as a *DEF file* from ICC or 3rd party tool

```
read_def DESIGN.def
```

- Read in as a *physical constraints file* from ICC

```
read_floorplan DESIGN.fp
```

- Open an already floorplanned *Milkyway cell*

```
open_mw_lib my_design_lib  
open_mw_cel DESIGN_floorplanned
```

Creating a new floorplan with IC Compiler
will be discussed in Unit 2



See Appendix A in
Unit 2 if floorplanning
with a 3rd party tool

1-68

Placement and Related Optimizations

Data Setup

Design Planning

Placement

Clock Tree Synthesis

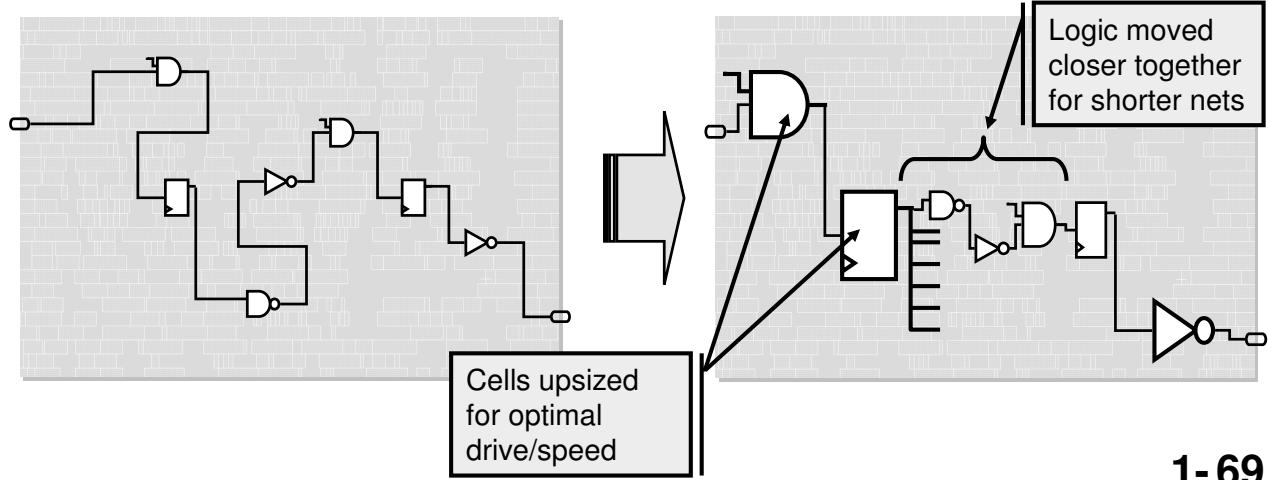
Route



1. Set placement options

2. Run `place_opt`

- Performs iterative placement and logic optimization
- Objective: Reduce/eliminate negative timing slack (DTDP)



1-69

Placement will be discussed in much greater detail in Unit 3.

Clock Tree Synthesis

Data Setup Design Planning Placement **Clock Tree Synthesis** Routing Chip Finishing

1. Set clock tree options/exceptions
2. Run **clock_opt**

- Builds the clock trees
- Performs incremental logic and placement optimizations
- Runs clock tree optimizations
- Routes the clock nets
- Can fix hold time violations
- Can perform inter-clock balancing



1-70

Clock tree synthesis will be discussed in much greater detail in Unit 4.

Routing

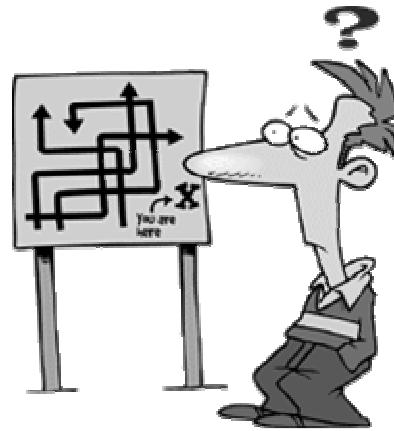
Data Setup Design Planning Placement Clock Tree Synthesis **Routing** Chip Finishing

1. Set routing options/exceptions

2. Run **route_opt**

Performs

- Global Route
- Track Assignment
- Detailed Route
- Search and Repair



Concurrently performs

- Logic, placement and routing optimizations
- Objective: Meet timing

1-71

Routing will be discussed in much greater detail in Unit 6

Chip Finishing

Data Setup Design Planning Placement Clock Tree Synthesis Routing Chip Finishing

- Also known as 'Design for Manufacturing'
- Entails:
 - Antenna Fixing
 - Wire Spreading
 - Double Via Insertion
 - Filler Cell Insertion
 - Metal Fill Insertion
 - Metal Slotting

Discussed in Unit 7

1-72

Analyzing the Results (1/2)

After each placement, CTS and routing step you should:

- Examine the log output for design summaries:
 - Utilization
 - Worst Negative Slack (WNS)
 - Total Negative Slack (TNS)
 - Legality of cell placement
 - Cell count and area
 - Design rule violations
- Use `report_qor` to see:
 - WNS/TNS per path group (clock group)
 - Other statistics

1-73

Analyzing the Results (2/2)

■ Generate more detailed reports

- Show all violating path end points
`report_constraint -all_violators`
- Show details of the worst violating setup path
`report_timing`
- Report physical design statistics (e.g. utilization)
`report_design -physical`
- Analyze the congestion
 - ◆ Congestion map (GUI)
`report_congestion`

1-74

Example “run” Script

```
...  
open_mw_cel ORCA_data_setup  
  
read_def ORCA.def  
save_mw_cel -as ORCA_floorplanned  
  
place_opt  
save_mw_cel -as ORCA_placed  
report_constraint -all  
  
remove_clock_uncertainty [all_clocks]  
clock_opt  
save_mw_cel -as ORCA_cts  
report_constraint -all  
  
route_opt  
save_mw_cel -as ORCA_routed  
report_constraint -all  
report_timing  
  
close_mw_lib  
exit
```

run.tcl

!

Commands can be run interactively, or in “batch” mode, shown here

UNIX\$ **icc_shell -f run.tcl | tee myrun.log**

1-75

Basic Flow Summary

You should now be able to:

■ **Perform data setup:**

- Create a Milkyway *design library* and *design cell*
- Load the necessary data required to run IC Compiler

■ **Execute a basic flow for design planning,
placement, CTS and routing in IC Compiler**



1-76

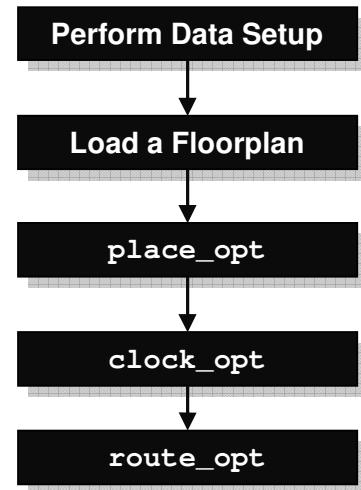
Lab 1: Design Setup and Basic Flow



60 minutes

Goals:

- Perform data setup
- Load a floorplan
- Perform placement, CTS and routing



1-77

This page was intentionally left blank

Agenda

**DAY
1**

i Introduction & Overview



1 Data Setup & Basic Flow



2 Design Planning



Unit Objectives



After completing this unit, you should be able to:

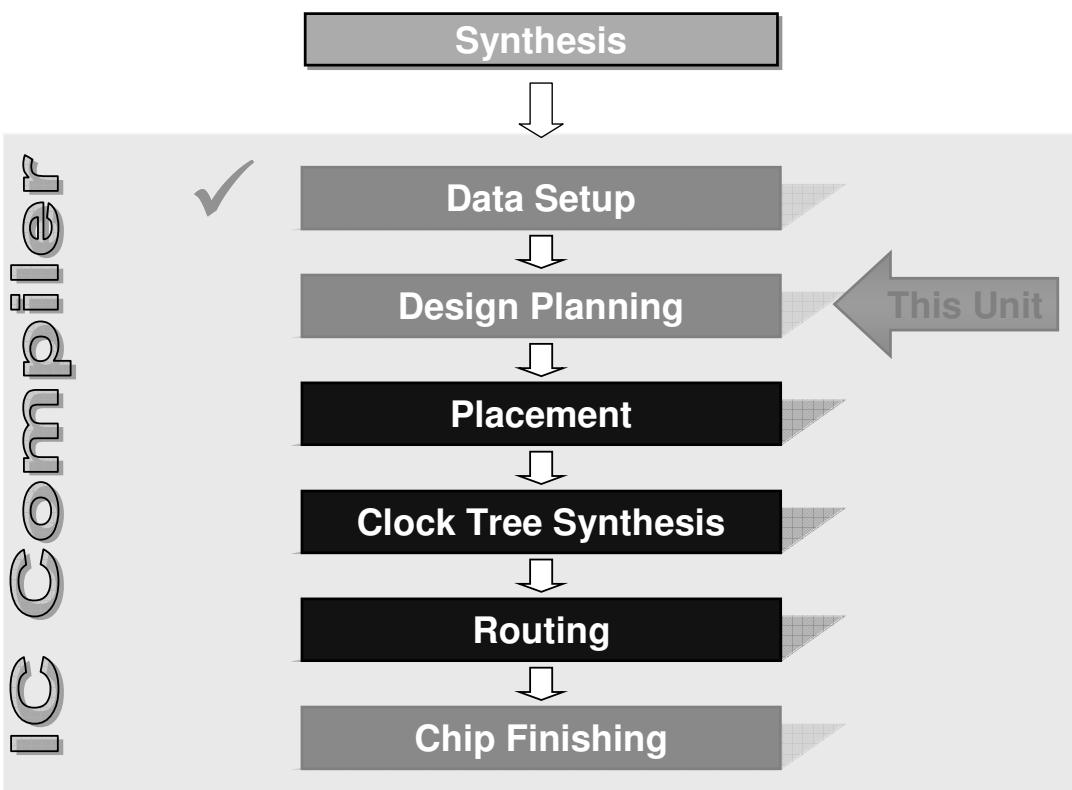
- Use IC Compiler to create a non-hierarchical chip-level floorplan
- Create a floorplan that is likely to be routable and achieve timing closure

2-2

Hierarchical design planning is covered in a separate workshop:

IC Compiler – Hierarchical Design Planning (ICC-HDP).

General IC Compiler Flow



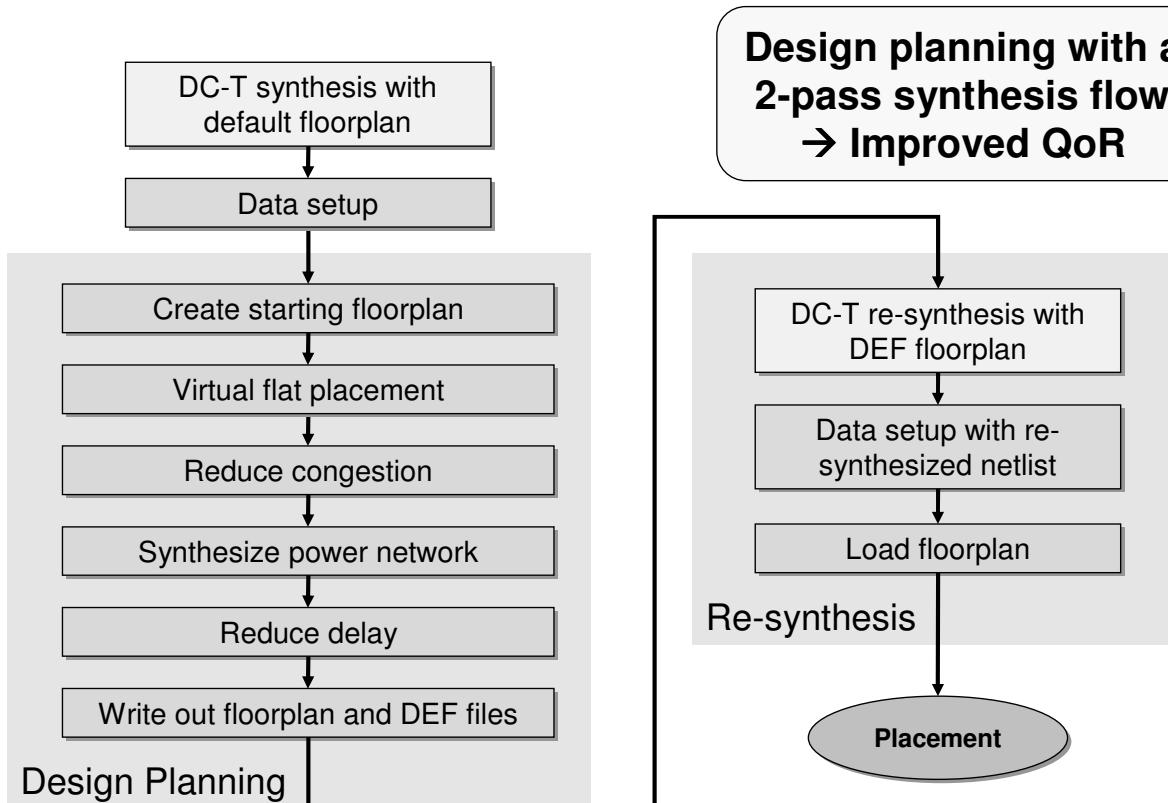
2-3

Terminology

- ***Design planning* is the iterative process of creating a floorplan**
- **A chip-level *floorplan* entails defining:**
 - Core size, shape and placement rows
 - Periphery: IO, power, corner and filler pad cell locations
 - Macro cell placement
 - Standard cell placement constraints (blockages)
 - Power grid (rings, straps, rails)
- **A *physical design*, or *layout*, is the result of a synthesized netlist that has been *placed* and *routed***

2-4

ICC Design Planning and Re-Synthesis Flow

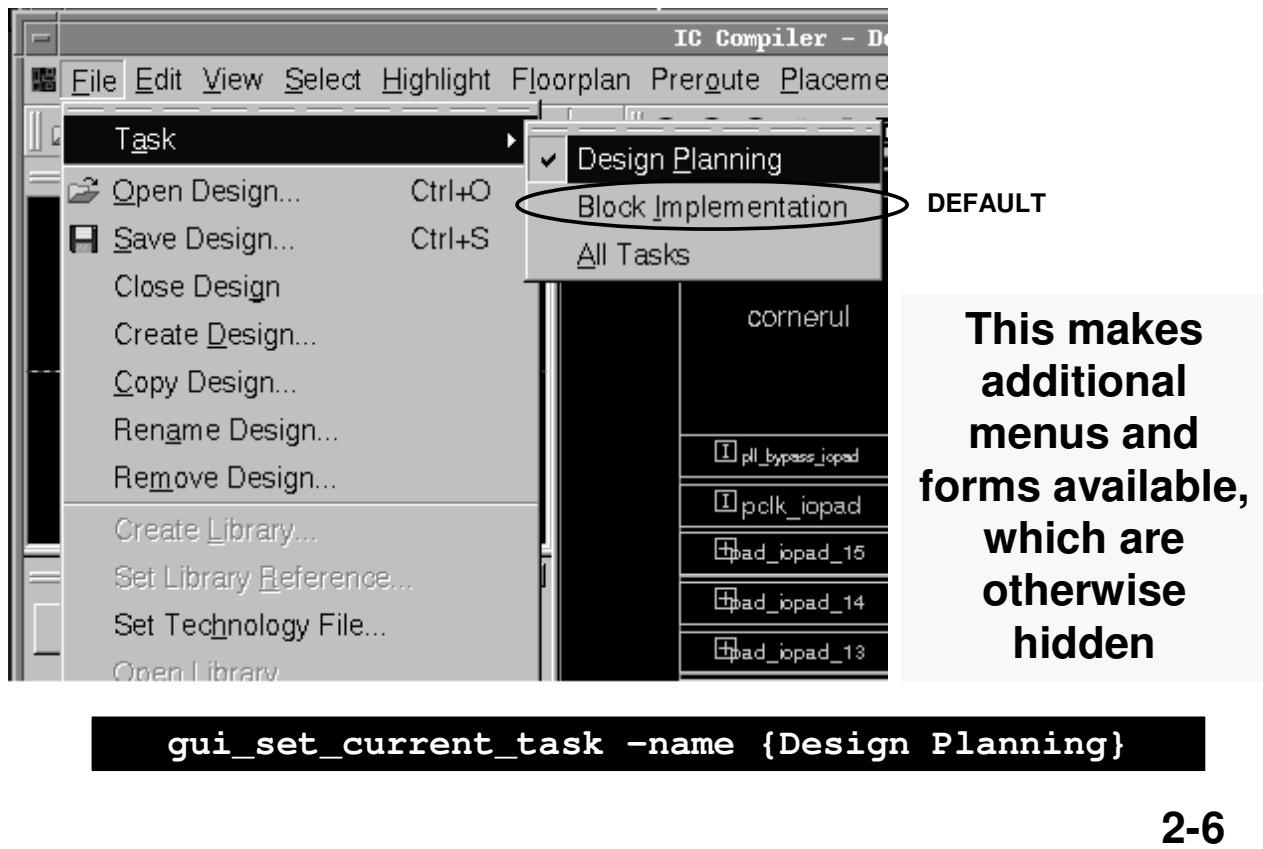


2-5

The above flow is recommended for designs with challenging “quality of results” (QoR) requirements. By re-synthesizing the design with DC-Topographical using an accurate floorplan you are providing IC Compiler with a potentially better starting netlist, which may give better results. If QoR is not critical, the re-synthesis step can be skipped and the resulting cell after design planning, with its netlist from the first synthesis execution, is taken directly to placement.

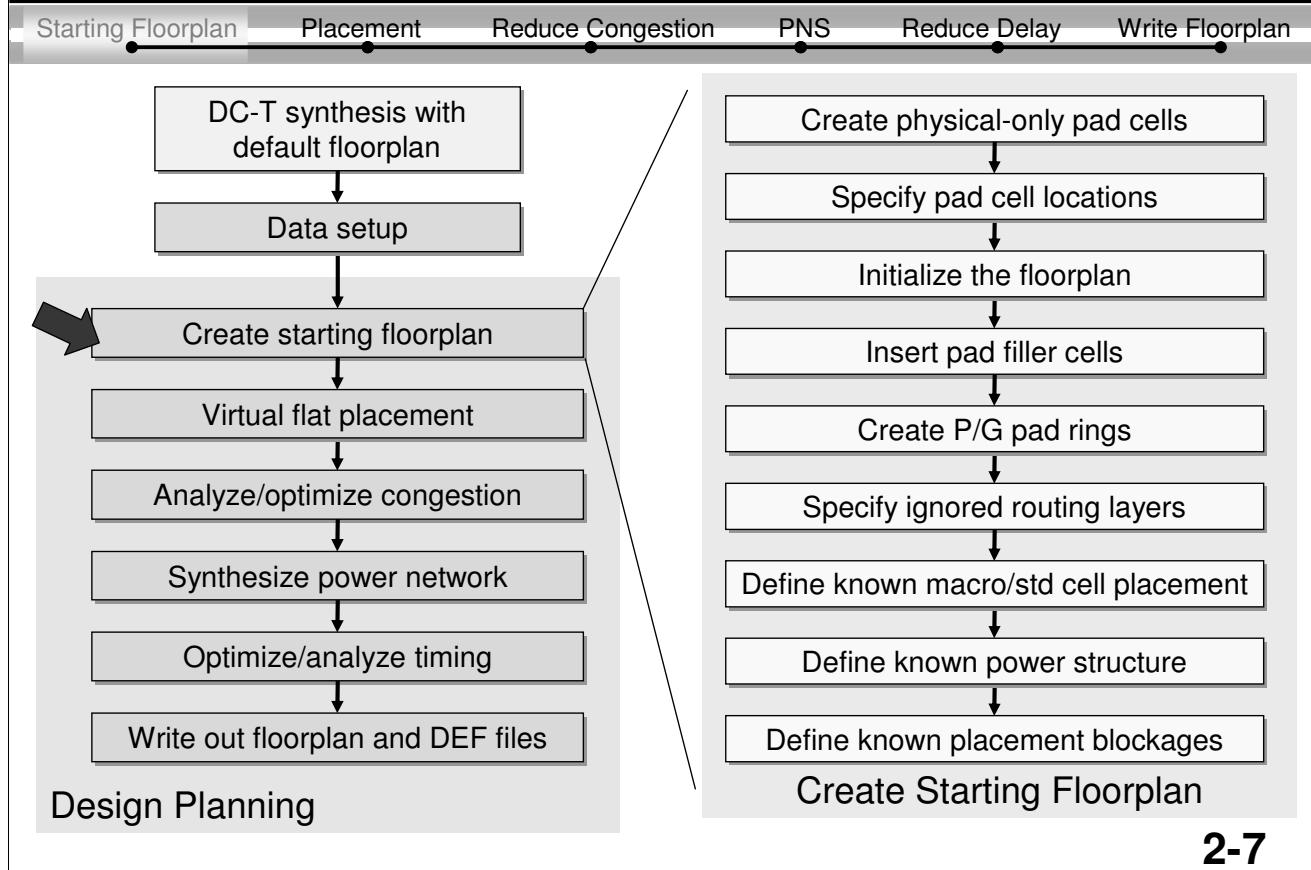
The above flow assumes that the RTL code is fairly complete when performing the initial synthesis. If the RTL contains a fair number of “black boxes” (i.e. undefined sections) you may need to perform additional iterations of design planning and synthesis, as the RTL code is solidified.

Select the *Design Planning* Task GUI



2-6

Create the Starting Floorplan



2-7

Create Physical-only Pad Cells

Starting Floorplan

- Physical-only pad cells (VDD/GND, corner cells) are not part of the synthesized netlist
- Must be created prior to specifying the pad cell locations

```
open_mw_cel DESIGN_data_setup

create_cell {vss_l vss_r vss_t vss_b} pv0i
create_cell {vdd_l vdd_r vdd_t vdd_b} pvd1
create_cell {CornerLL CornerLR CornerTR CornerTL} \
            pfre1r
```

2-8

Specify Pad Cell Locations

Starting Floorplan

```
; tdf file

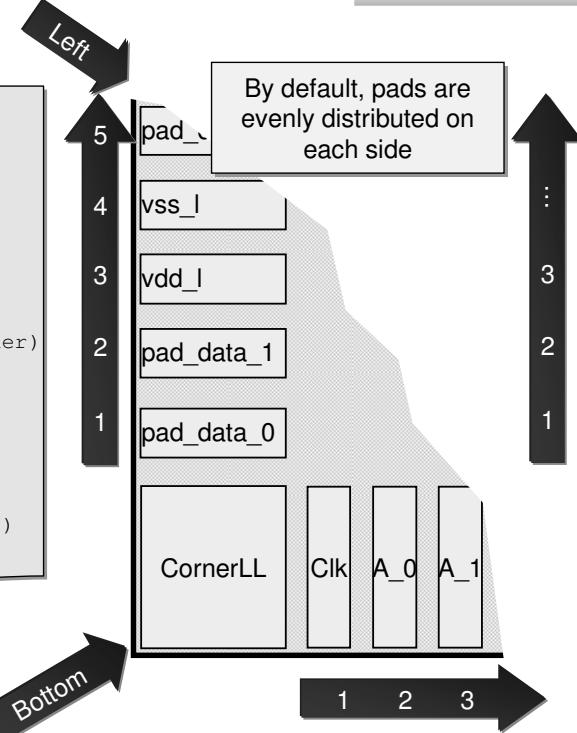
; Place the corner cells
pad "CornerLL" "bottom"
pad "CornerLR" "right"
pad "CornerTR" "top"
pad "CornerTL" "left"

; Place io and power pads
; Left/Right sides start from bottom (excluding corner)
pad "pad_data_0" "left" 1
pad "pad_data_1" "left" 2
pad "vdd_1" "left" 3
pad "vss_1" "left" 4
pad "pad_data_2" "left" 5

; Bottom/Top sides start from left (excluding corner)
pad "Clk" "bottom" 1
pad "A_0" "bottom" 2
pad "A_1" "bottom" 3
```

read_io_constraints <tdf_file>

By default, pads are evenly distributed on each side



Constraints are honored when the floorplan is created

2-9

pad padName padSide [padOrder] [padOffset] ["reflect"]

read_io_constraints [-append] [-cel_name] [-child_cel] <TDF_file>

GUI: Floorplan → Read I/O Constraints ...

The example on the slide uses the *padOrder* syntax. *padOffset* can be used (additionally, or instead) to specify an absolute offset (in microns) from the bottom edge or left edge of the periphery boundary.

Pads that are not defined in *tdf* are assigned and placed automatically. Pads that defined in *tdf* are considered as constrained pads (unless *padOrder* = 0 and no *padOffset* is given). Pads defined in *tdf* are placed first, then the undefined ones are placed into available locations. ICC can dump out the *tdf* for further modification (*write_io_constraints*).

The numbering is NOT absolute. For example, with the *tdf* below, where the smallest *padOrder* is 2 and *padOrder* 4 is not defined, and any number of unconstrained pads (1 or more!) can be placed between the corner pad and A1, as well as in between A2 and A3. No additional pads can be placed between consecutively numbered pads (e.g. A1 and A2).

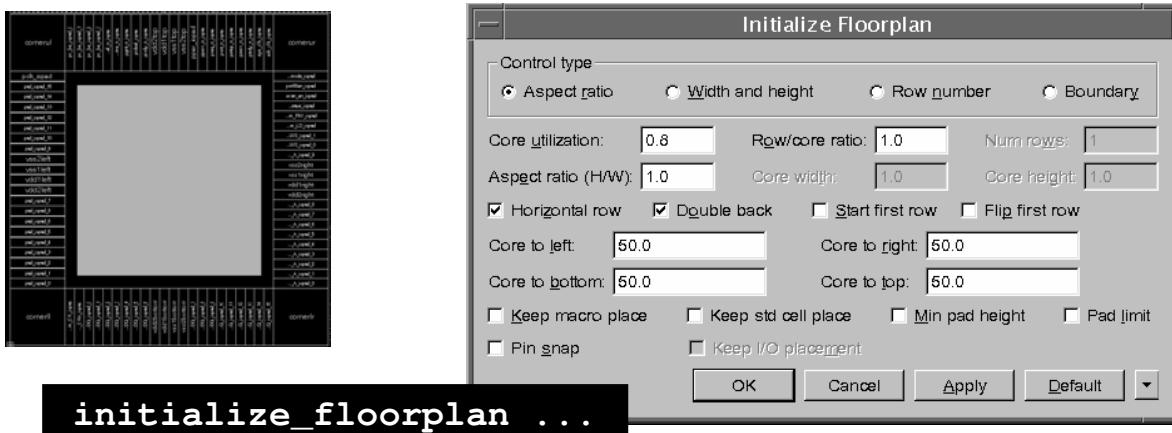
```
pad "A1" "left" 2
pad "A2" "left" 3
pad "A3" "left" 5
```

Initialize the Floorplan

Starting Floorplan

Creates the core and periphery area

- Defines placement or site rows within the core area
- Defines the chip boundary or periphery area
- Places IO pads, as defined in the TDF file
 - ◆ Pads defined in netlist and by `create_cell`
 - ◆ Ordering defined by TDF file



2-10

GUI: Floorplan → Initialize Floorplan ...

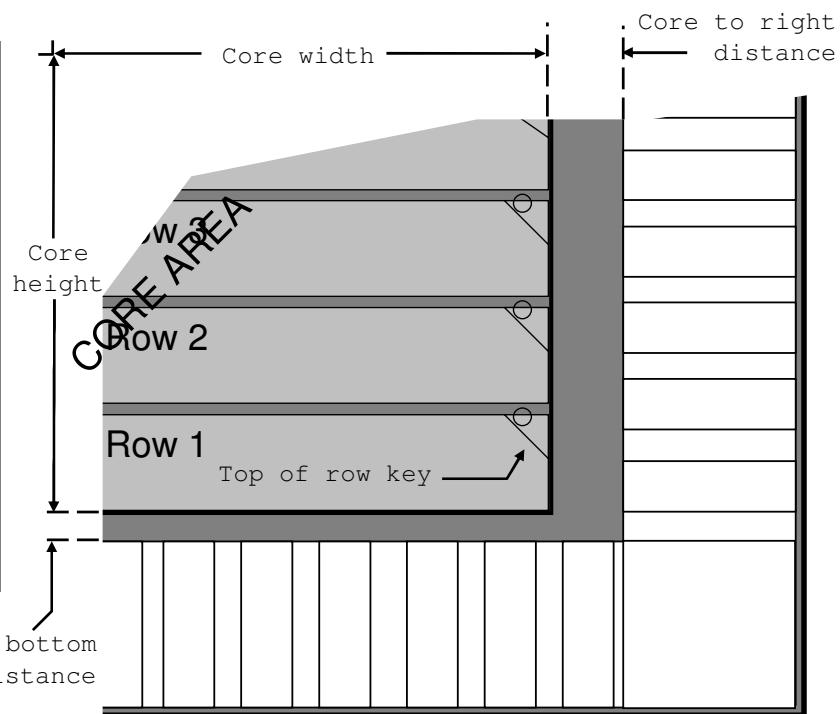
When you select one of the four *Control Type* switches at the top of the dialog (e.g. *Aspect Ratio* or *Width and Height*), a corresponding subset of the six available parameter fields is highlighted to allow for user input (example: *Core utilization*, *Aspect ratio (H/W)*, *Row/core ratio*, etc.)

For rectilinear shapes, use the `initialize_rectilinear_block` command.

Core Area Parameters

Control Type

- * Aspect ratio
 - Core utilization
 - Aspect ratio (H/W)
 - Row/core ratio
- * Width and height
 - Core width
 - Core height
 - Row/core ratio
- *
- *

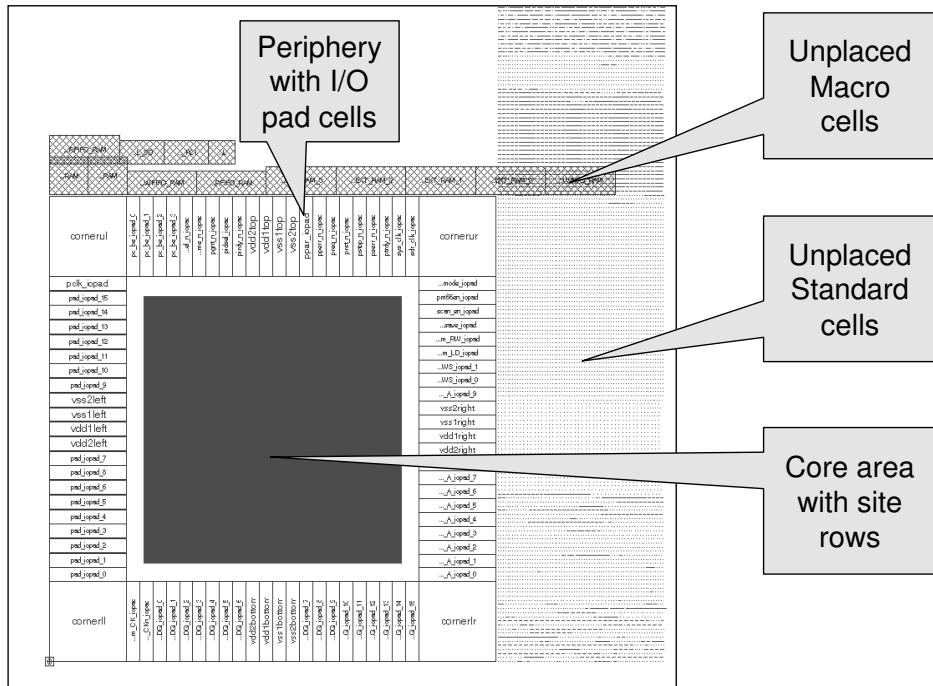


Example of a horizontal, no double back, no-flip
first row and Row/core <1.0

2-11

Floorplan After Initialization

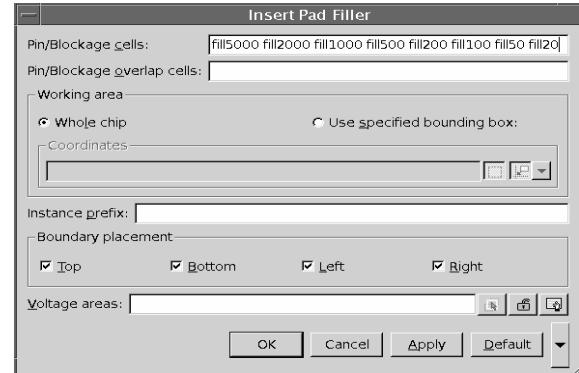
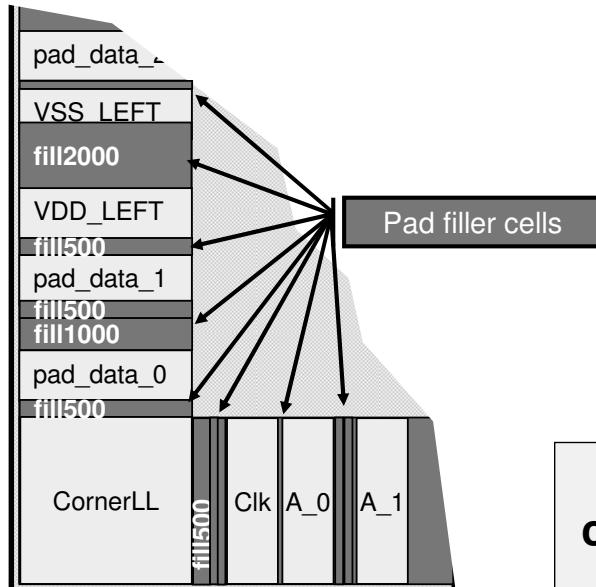
Starting Floorplan



2-12

Insert Pad Filler Cells

Starting Floorplan



**May be needed for
continuity of N-well, P-well,
and/or P/G routing**

```
insert_pad_filler -cell "fill15000 fill2000 fill1000 ... "
```

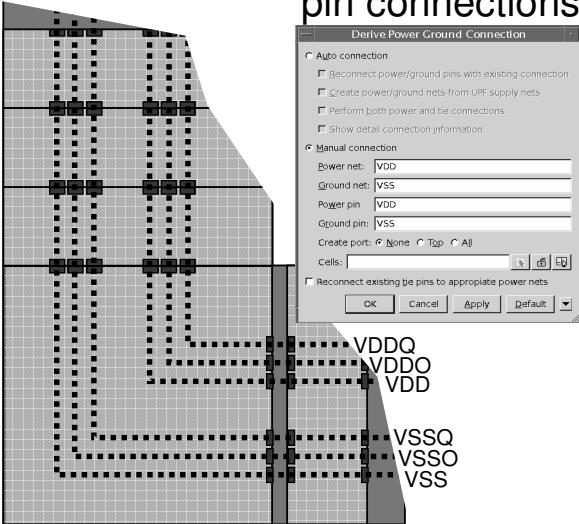
2-13

GUI: Finishing → Insert Pad Filler ...

Create P/G Pad Rings

Starting Floorplan

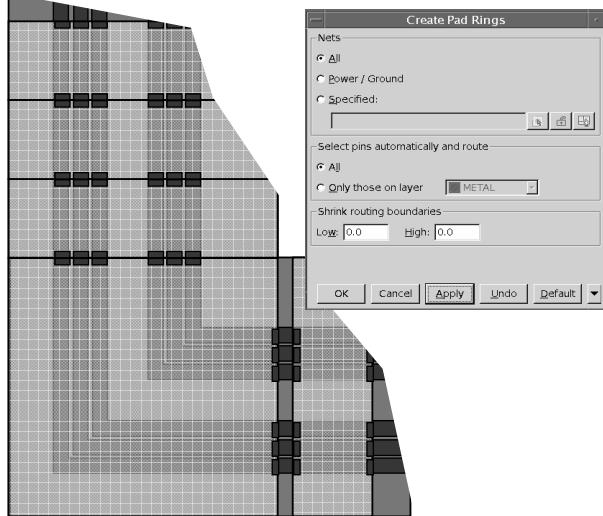
Make logical P/G pin connections



```
derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS  
derive_pg_connection -power_net VDDO -power_pin VDDO -ground_net VSSO -ground_pin VSSO  
derive_pg_connection -power_net VDDQ -power_pin VDDQ -ground_net VSSQ -ground_pin VSSQ
```

create_pad_rings

Route P/G rings



May be needed for P/G continuity

2-14

GUI: Preroute → Derive PG Connection ...

GUI: Preroute → Create Pad Rings ...

If no filler cells are used, or if some or all pad cells do not have P/G pre-routes built in, you will need to create a pad ring to ensure P/G continuity.

Prior to Virtual Flat Placement

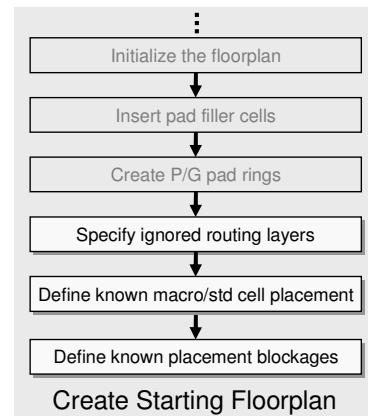
Starting Floorplan

■ By default *virtual flat placement* (`create_fp_placement`):

- Places standard cells and non-fixed macros
- Can place cells anywhere in the core
- Assumes all layers defined in the technology file are used
- Does not consider power structure effects on congestion

■ Prior to placement specify any non-default constraints

- Routing layers that should not be used
- Pre-defined macro/standard cell locations
- Placement blockages



2-15

Ignore Extra Routing Layers

Starting Floorplan

- By default IC Compiler will use all metal layers defined in your technology
- Prior to detailed routing this can result in:
 - Optimistic congestion analysis
 - Inaccurate RC parasitic calculations, which affect timing
- Tell IC Compiler to ignore these unused layers
 - Accurate congestion and timing analysis prior to routing
 - No additional “route guides” needed

```
set_ignored_layers -max_routing_layer M7
```

report_ignored_layers
remove_ignored_layers

2-16

This command is useful if you have a technology file which defines, for example, nine metal layers, but you plan to use fewer layers, for example, only up to metal 7.

The command can only be applied once a floorplan has been defined (through initialize_floorplan, read_def or read_floorplan).

The example above ignores the layers above m7 for RC and congestion estimation. It also prevents the router from routing on layers above m7.

The cleanest solution is to ask your foundry for a 7 layer technology file instead. The reason is that your tech data for layer 7, while more accurate using this command, is still not completely accurate.

User can check the log during placement for:

Warning: Poly layer CP is ignored during extraction. (RCEX-076)

Information: Layer METAL8 is ignored for resistance and capacitance computation. (RCEX-019)

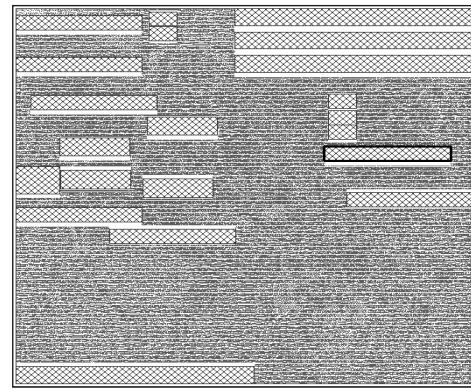
Information: Layer METAL9 is ignored for resistance and capacitance computation. (RCEX-019)

Constraining Macros

Starting Floorplan

- After `create_fp_placement` the resulting macro placement can be “disorganized”

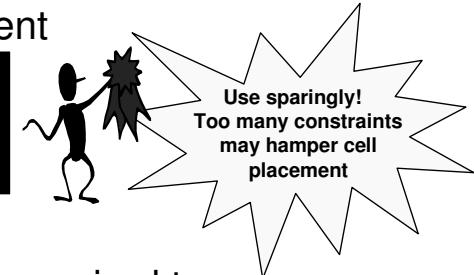
- May complicate the power structure
- May use more routing resources for busses



- You can define macro placement:

- Manually, using the GUI, prior to placement
- With constraints, which guide placement

```
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
```



- Placement constraints are “soft”

- Placer will *try* to honor them but is not required to

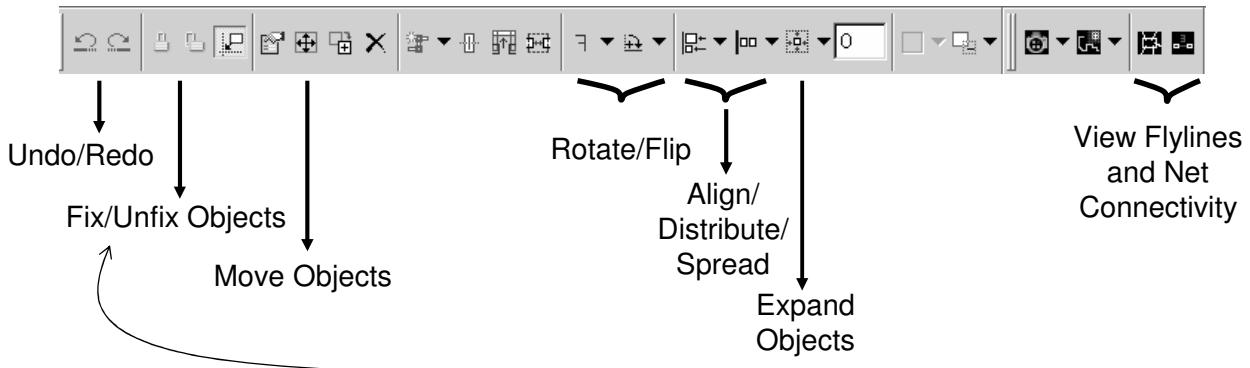
2-17

If two macros connect to the same bus, and the macros are not “aligned”, additional routing resources may be needed to connect these macros to the bus.

Manual Macro Placement

Starting Floorplan

- The GUI can be used to “manually” place macros in known locations and orientations (will be shown in lab)



- Placed macros , if not “fixed”, can be moved by `create_fp_placement`

- To prevent this “fix” the placed macros using the toolbar, or:

```
set_dont_touch_placement [get_cells <list_of_cells>]
```

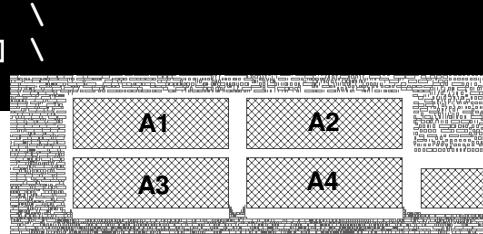
2-18

The `set_dont_touch_placement` attribute will prevent any placement modifications. If, later in the design planning flow, you discover that one or more of these manually placed cells are causing congestion or other problems and you need to modify their placement, you can “unfix” placement with `remove_dont_touch_placement`.

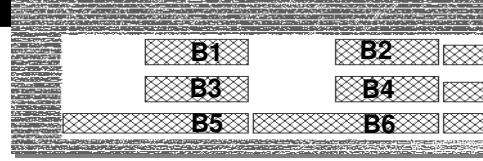
Macro Constraints: Arrays

Starting Floorplan

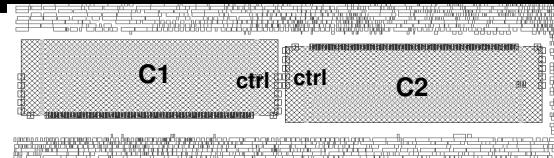
```
set_fp_macro_array -name A_array -elements \
[list [get_cells A1 A2] [get_cells A3 A4]] \
-x_offset 15 -y_offset 110
```



```
set_fp_macro_array -name B_array -elements \
[list [get_cells B1 B2] [get_cells B3 B4] [get_cells B5 B6]] \
-x_offset 4 -y_offset 12 -align_2d rb
```



```
set_fp_macro_array -name C_array -elements [get_cells C1 C2] \
-x_offset 4 -align_pins [list [get_pins C1/ctrl] [get_pins C2/ctrl]]
```



2-19

```
set_fp_macro_array
  -name string
  [-elements collection_of_macro_cell_objects]
  [-align_edge t | b | l | r | c ]
  [-align_2d lb | lc | lt | rb | rc | rt | cb | cc | cr]
  [-align_pins {list of two pin objects}]
  [-x_offset float] [-y_offset float]
  [-use_keepout_margin]
  [-vertical]
  [-rectilinear]
  [-reset]
```

```
report_fp_macro_array
```

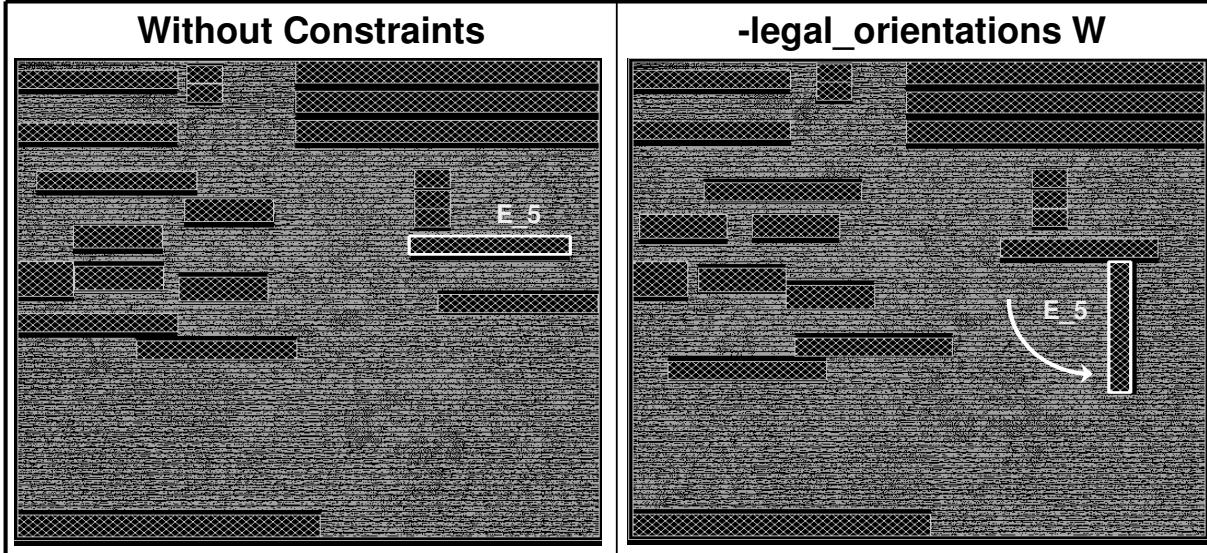
Virtual flat placement will, by default, create arrays out of small macros. This comes from the default *auto-grouping* setting: `set_fp_placement_strategy -auto_grouping low`. The above command allows the user more control.

Note: A macro array is treated as a single object during placement - standard cells can not be placed inside the array.

Macro Constraints: Legal Orientation Option

Starting Floorplan

```
set_fp_macro_options [get_cells E_5] -legal_orientations W
```



2-20

```
set_fp_macro_options $list_of_macro_objects
  [-legal_orientations {legal_orientations}]
  [-anchor_bound t|b|l|r|tl|tr|bl|br|tm|bm|lm|rm|c]
  [-x_offset distance_from_core_edge]
  [-y_offset distance_from_core_edge]
  [-align_pins ref_port_of_block constrained_pin]
  [-side_channel {left_dist right_dist top_dist bottom_dist}]
  [-reset]
```

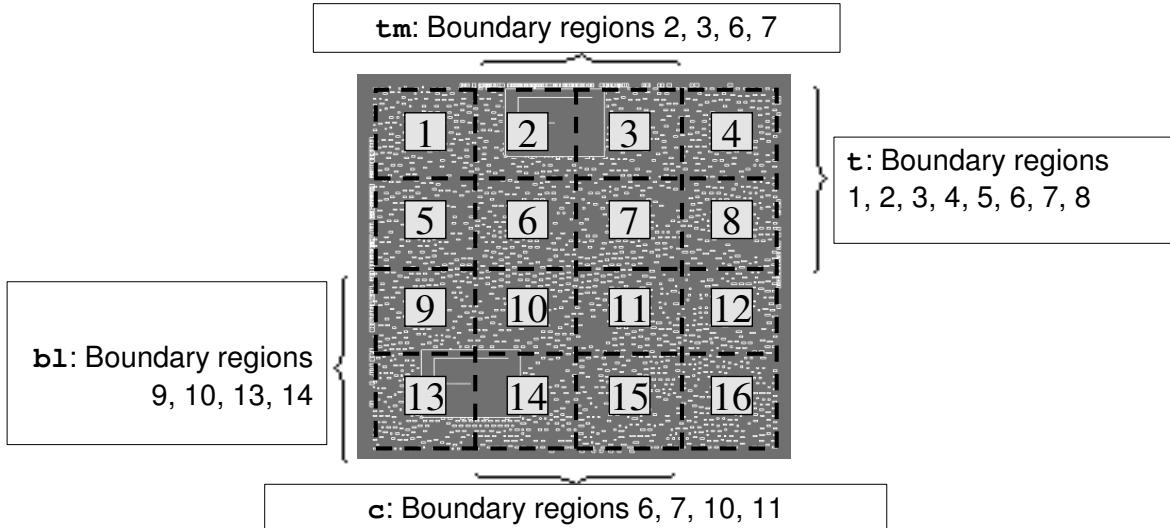
```
report_fp_macro_options
```

Macro Constraints: Anchor Bound Option

Starting Floorplan

```
set_fp_macro_options collection_of_cells \
-anchor_bound <l, r, t, b, bl, tl, br, tr, bm, tm, lm, rm, c>
```

Cells are placed in specified ½ or ¼ core areas



2-21

Additional set_fp_macro_options -anchor_bound examples:

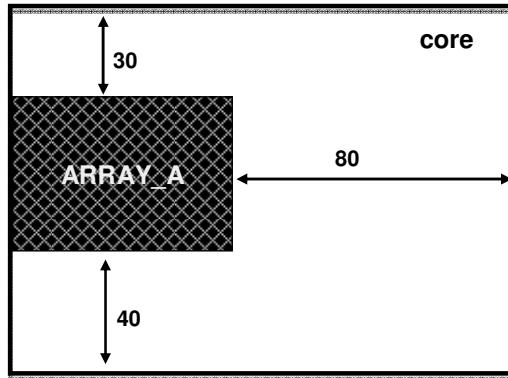
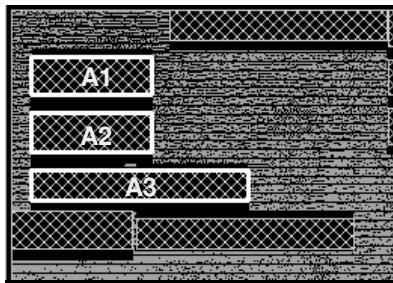
t (top)	{1-8}
tl (top-left)	{1,2,5,6}
b (bottom)	{9-16}
bl (bottom-left)	{9,10,13,14}
lm (left-middle)	{5,6,9,10}
c (center)	{6,7,10,11}

Macro Constraints: Side Channel Option

Starting Floorplan

```
set_fp_macro_array -name ARRAY_A -vertical \
                    -elements [get_cells "A1 A2 A3"]
set_fp_macro_options array2 \
                     -side_channel "0 80 30 40"
```

Cells or arrays are spaced from the core edges



2-22

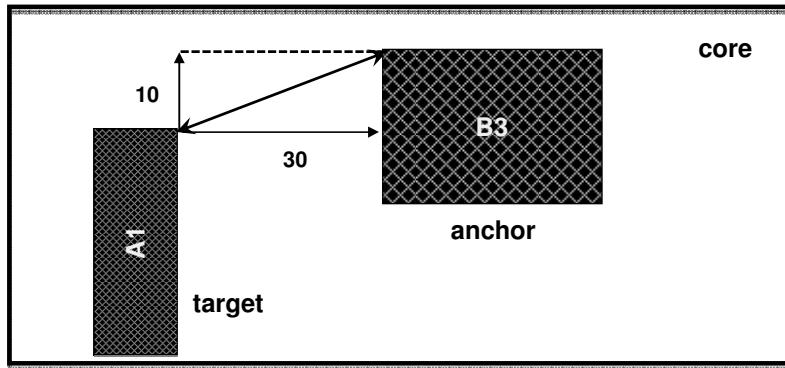
Macro Constraints: Relative Location

Starting Floorplan

set_fp_relative_location sets a relative location constraint on a target cell with respect to an anchor object

- Anchor: A fixed cell, another cell with a relative-location, or a corner of the core area

```
set_fp_relative_location -name RP1 -target_cell "A1"
                          -target_orientation "S" -target_corner "tr" \
                          -anchor_object "B3" -anchor_corner "tl"
                          -x_offset 30 -y_offset 10
```



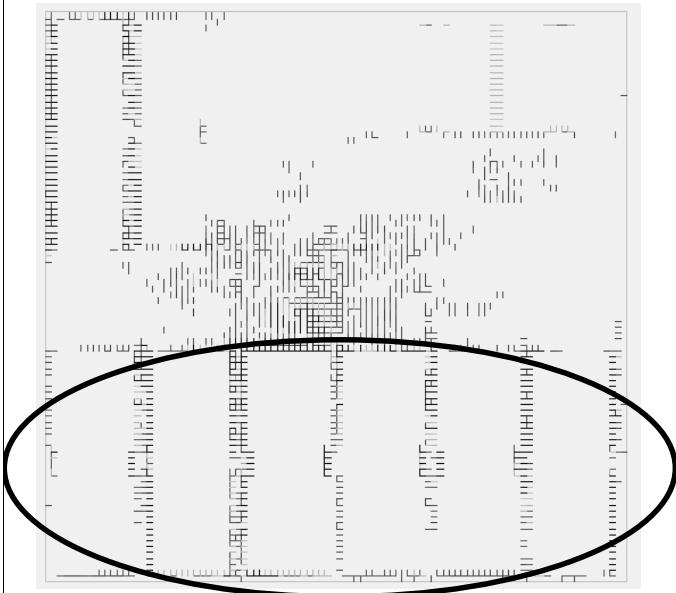
2-23

```
set_fp_relative_location
  -name constraint_name
  -target_cell cell_name
  [-target_orientation N | S | E | W | FN | FS | FE | FW]
  [-target_corner bl | br | tl | tr]
  [-anchor_object object_name]
  [-anchor_corner bl | br | tl | tr]
  [-x_offset distance]
  [-y_offset distance]

extract_fp_relative_location
remove_fp_relative_location
report_fp_relative_location
```

Congestion Potential Around Macro Cells

Starting Floorplan



- Routing to standard cells can often be difficult near edges or corners of macro cells
- Solution:
Add placement blockages around macro cells
 - *Hard* blockages prevent standard cells from ever being placed there
 - *Soft* blockages prevent standard cells from being placed during initial coarse placement, but are ignored by subsequent placement legalization or optimization

2-24

Apply Global Placement Blockages

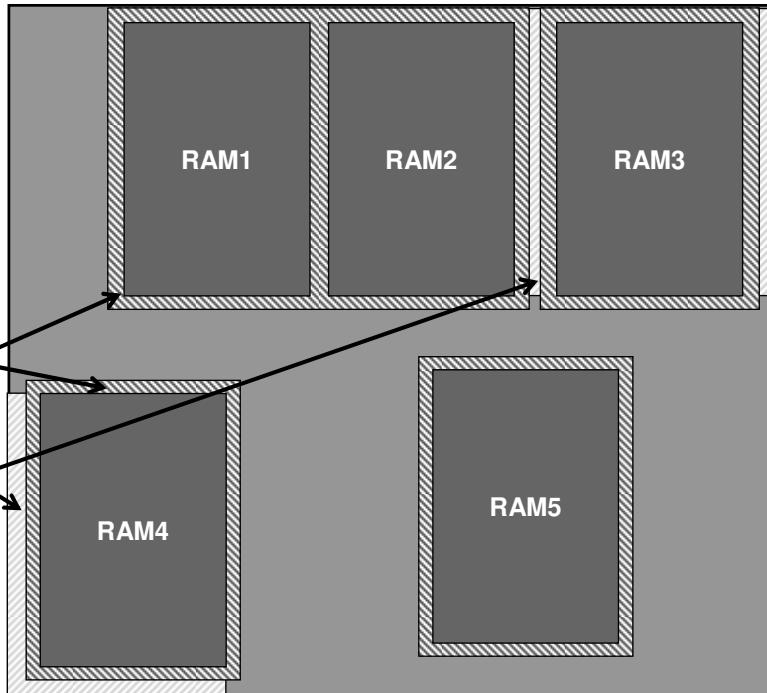
Starting Floorplan

Global blockages

- Apply to all ‘fixed’ cells
- Consider all cell edges

Hard blockage always created on all four sides

Soft blockage created only for narrow channels between macros, or between a macro and the core boundary



```
set physopt_hard_keepout_distance 10  
set placer_soft_keepout_channel_width 25
```



Also add to
.synopsys_dc.setup

2-25

Use the above variables to avoid/minimize cell placement in tight areas between RAMs, or between RAMs and core boundary. These are areas where it is legal to place cells, but it is very inefficient to do so, due to the difficulty of routing in these areas. Typically, routing congestion and timing-driven constraints will prevent this from happening during placement. It is more efficient, however, for you to tell the tool about these restrictions explicitly, rather than letting the tool spend CPU cycles to figure it out.

The variable `physopt_hard_keepout_distance` will create a *hard* placement blockage rectangle around each macro (fixed cell), essentially extending each macro’s boundary by the distance specified.

The variable `placer_soft_keepout_channel_width` will create a *soft* placement blockage where the distance between two macros, or between the macro and the core boundary, is less than or equal to the specified value (i.e. narrow channels). The unit for the above variables is microns.

Note that *hard* blockages are always created around the entire macro, while *soft* blockages are not generated for every fixed macro in the design, nor around every macro edge . Soft blockage areas are generated only if the distance between two macros, or between a macro and the nearest core boundary, is less than the channel width specified.

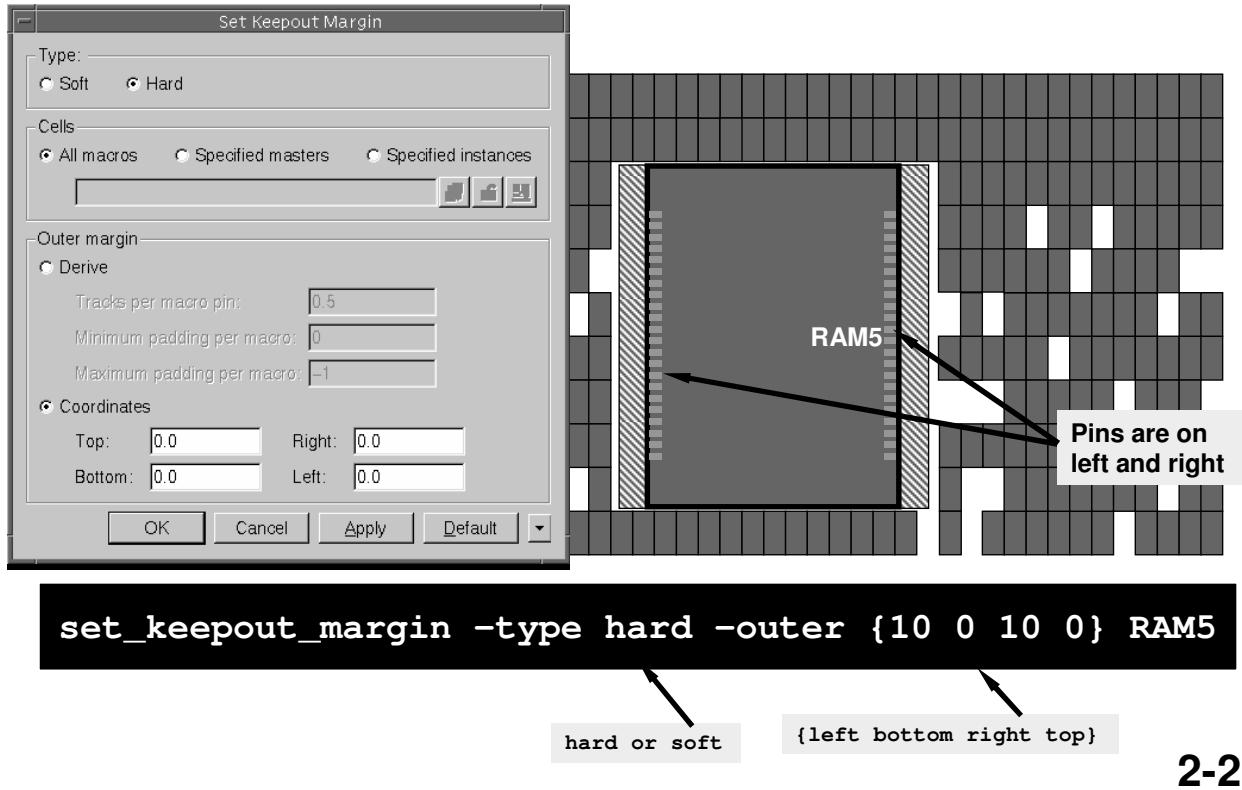
As shown in the example above, it is common to create hard blockages in an area tightly surrounding the macros, while including soft blockages in a slightly wider area, surrounding the hard blockages. For areas where hard and soft placement blockages overlap, the hard placement blockage takes priority.

Since the above are variables, their settings will be lost after exiting the current IC Compiler session. The simplest way to ensure that these settings are applied during subsequent sessions for placement, CTS and routing, is to include them in your `.synopsys_dc.setup` file.

Apply Specific Placement Blockages

Starting Floorplan

■ Pad the macro to reduce congestion around the pins



2-26

GUI: Placement→Set Keepout Margin...

In the example above, the shown macro is not placed near another macro or the edge of the core. However, congestion appears along the left and right edges, where the macro pins are. The above command allows you to apply hard or soft blockages around specified edges of specified macro cells.

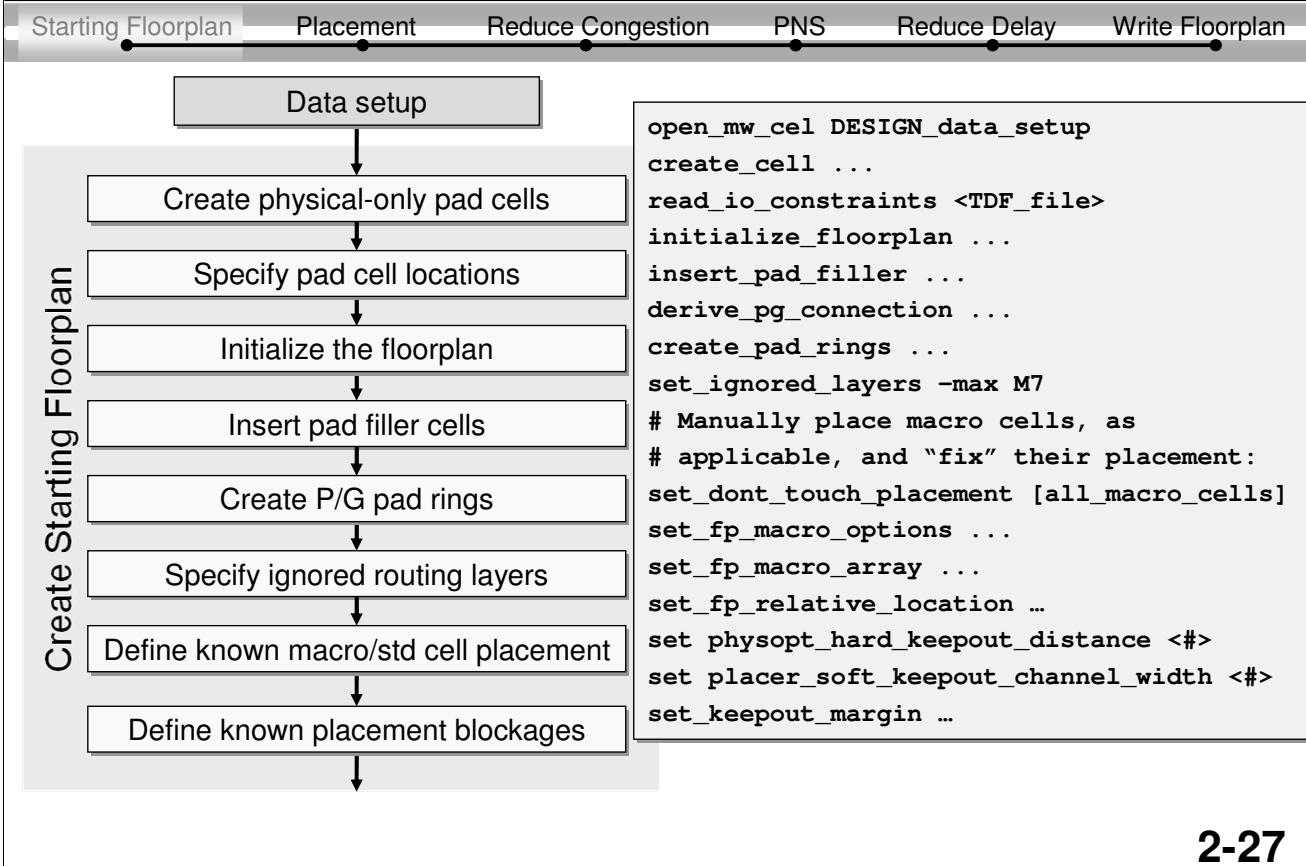
Note: The “left bottom right top” edges which the numbers in parenthesis are associated with, correspond with the left, bottom, right and top edges, respectively, of the macro cell in its original orientation. If, for example, the above macro cell were rotated counter clock-wise by 90 degrees, the “left” hard blockage area of 10 would rotate with the macro and would appear, from the core area’s point of view, to be on the bottom edge of the macro.

The following commands allow reporting and removal:

```
report_keepout_margin  
remove_keepout_margin
```

Note: ICC automatically applies a keep-out margin for each macro based on pin-count, if no user-defined or library-defined margin exists.

Summary: Create the Starting Floorplan



2-27

Test For Understanding

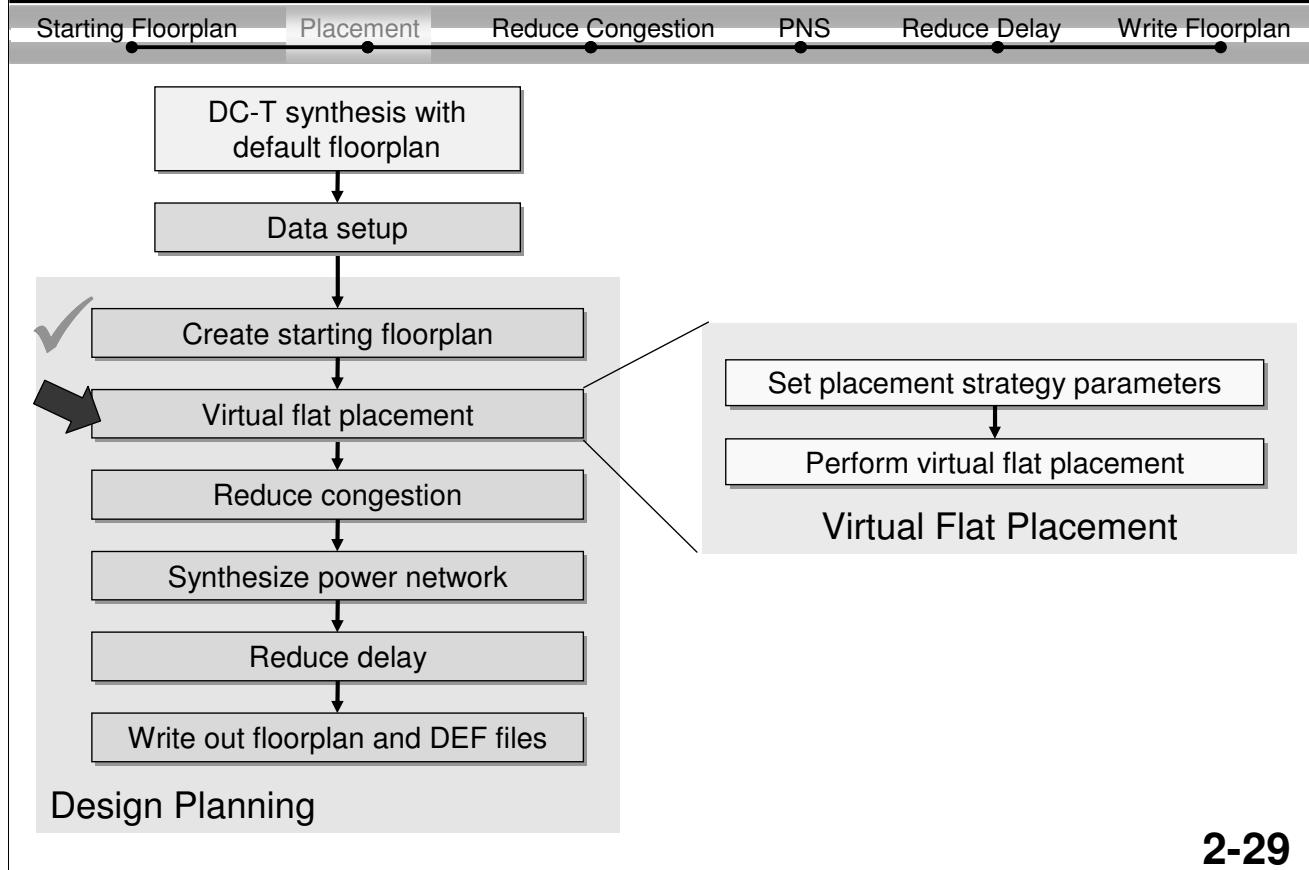


1. What are “physical-only” cells? What are some examples of physical-only pad cells?
2. What does the TDF file define?
3. What does `initialize_floorplan` do?
Circle all that apply:
 - a) Defines placement rows within the core area
 - b) Defines the chip boundary or periphery area
 - c) Places IO pad cells in their defined locations
 - d) Places macro cells per their placement constraints
4. A “soft” placement blockage will:
 - a. Allow only timing-critical cells to be placed
 - b. Allow only non timing-critical cells to be placed
 - c. Allow cell placement only during initial coarse placement
 - d. Prevent cell placement during initial coarse placement

2-28

1. Cells that are not part of the synthesized netlist, for example VDD/GND, corner and filler pad cells.
2. Pad cell locations
3. A, B and C
4. D. Any subsequent legalization, placement and optimization steps are allowed to place cells in the soft blockage area.

Perform Virtual Flat Placement



2-29

Set Placement Strategy Parameters

Placement

- Placement strategy parameters are used to control the following during `create_fp_placement`:

- How macros are handled
- Optimization algorithms and effort

- Consider setting a “sliver size” and turning on “virtual IPO”:

```
report_fp_placement_strategy
set_fp_placement_strategy -sliver_size 10
                           -virtual_IPO on
```

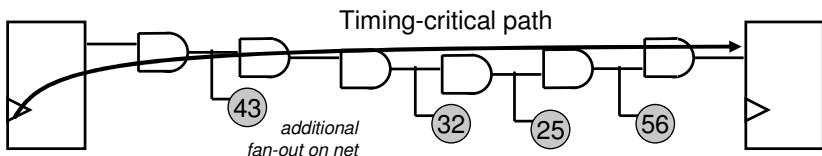
- In the example above standard cells will not be placed in channels between macros (*slivers*) of less than 10 microns
 - ◆ Helps to reduce potential congestion between macros
- Next slide: Explanation of *virtual in-place optimization (VIPO)*
- Use default settings for the remaining options (see notes below) unless you have a specific need to modify them

2-30

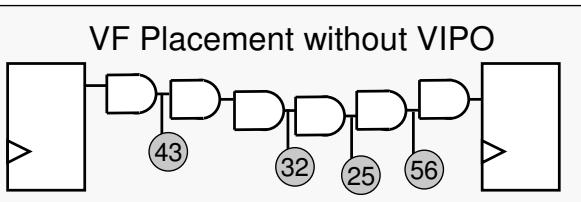
```
set_fp_placement_strategy
  -macro_orientation automatic | all | N
  -auto_grouping none | user_only | low | high
  -macro_setup_only on | off
  -macros_on_edge on | off
  -sliver_size <0.00>
  -snap_macros_to_user_grid on | off
  -fix_macros none | soft_macros_only | all
  -congestion_effort low | high
  -IO_net_weight <1.0>
  -plan_group_interface_net_weight <1.0>
  -voltage_area_interface_net_weight <1.0>
  -voltage_area_net_weight_LS_only on | off
  -spread_spare_cells on | off
  -legalizer_effort low | high
  -virtual_IPO on | off
  -pin_routing_aware on | off
```

VF Placement with *Virtual IPO* (VIPO)

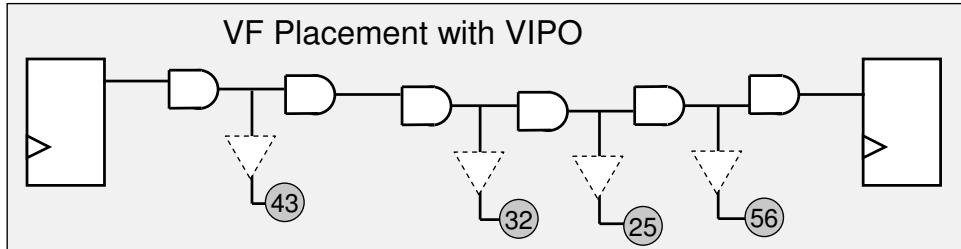
Placement



Original netlist contains large fan-outs along the critical path which can contribute significantly to delay



Without VIPO, cells along the critical path are placed close together to reduce delay – this can cause congestion and may not be necessary, since IPO during actual placement optimization can easily



With VIPO turned ON, sizing and buffer insertion (which mimics more realistic placement optimization) is performed “in memory”, thus relaxing the need to unnecessarily shorten the distance between the critical cells

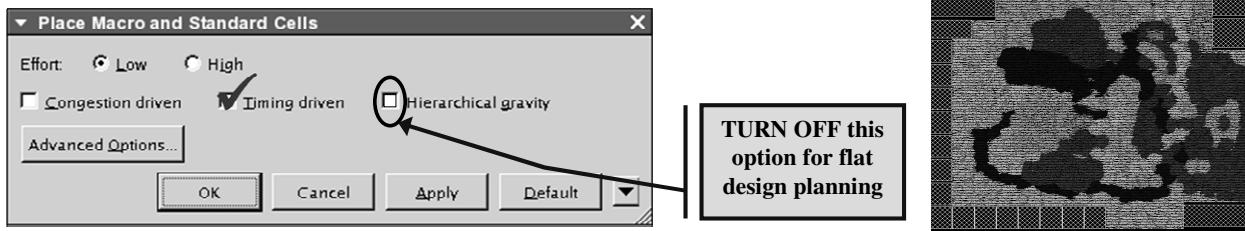
2-31

Perform Virtual Flat Placement

Placement

Perform quick placement, which will be used to:

- Improve floorplan's impact on congestion and timing
- Plan the power structure



```
create_fp_placement -timing_driven -no_hierarchy_gravity
```

- ❖ Standard cells and non-fixed macros are legally placed
- ❖ By default placement is wirelength-driven → enable timing-driven
- ❖ No logic optimization is done
- ❖ By default clumps cells within same logical hierarchy → turn gravity off

2-32

GUI: Placement → Place Macros and Standard Cells ...

```
create_fp_placement
[-effort string]
[-max_fanout integer]
[-no_hierarchy_gravity]
[-no_legalize]
[-incremental string]
[-plan_groups collection of plan groups]
[-voltage_areas collection of voltage areas]
[-congestion_driven]
[-timing_driven]
[-num_cpus integer]
[-optimize_pins]
[-ignore_scan]
```

The `-timing` option is used to more closely mimic the timing-driven algorithm used by default during actual placement (`place_opt`).

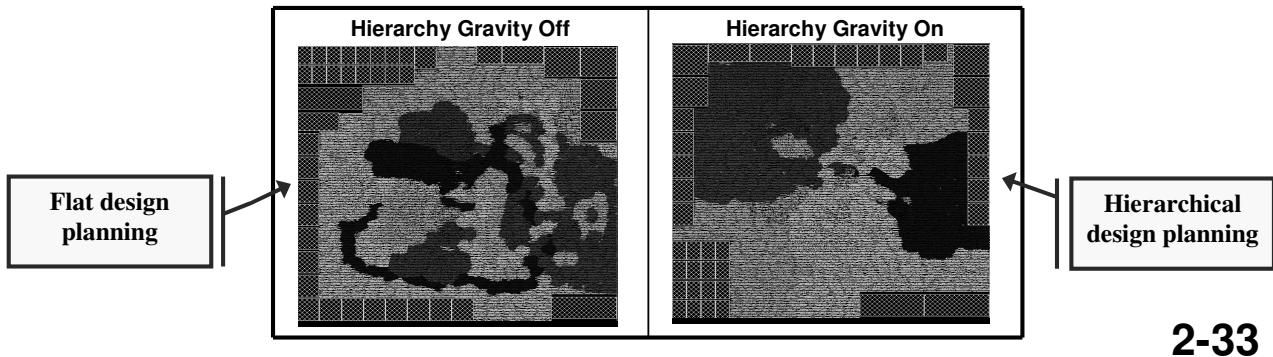
The `-num_cpus <#>` option allows the use of additional free CPUs on your machine for faster, parallel processing, using only one license.

Note: `create_fp_placement` uses the entire core area as the placement area, not the placement rows. As a result, if you have “cut” out part(s) of the placement rows, cells will be still be placed in the cut area(s). To prevent this apply a placement blockage on the “cut” area(s): `create_placement_blockage`.

Hierarchy Aware Placement or Gravity

Placement

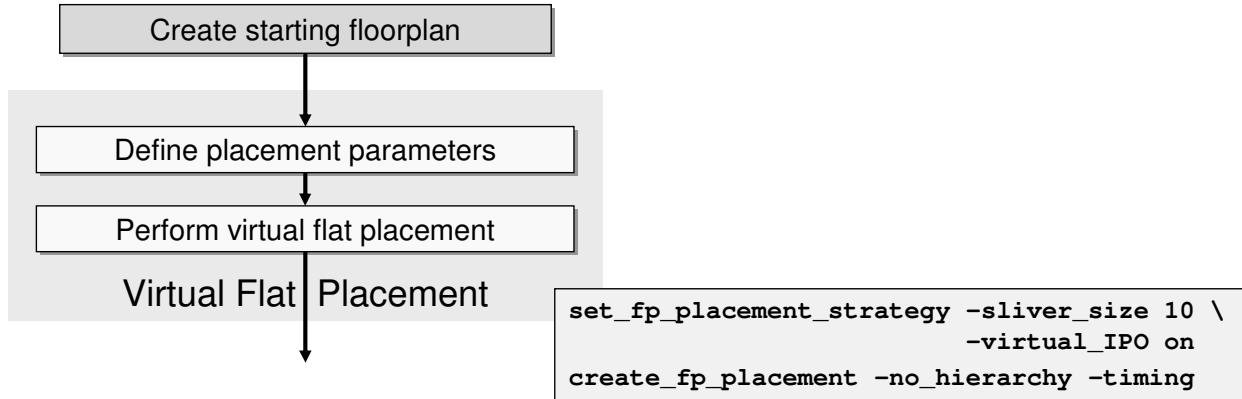
- By default `create_fp_placement` performs ***hierarchy-aware placement (gravity = ON)***
 - Placement tries to keep cells in the same logical hierarchy blocks physically close
 - The resulting placement becomes a useful guide to form physical “partitions” in a hierarchical design flow (discussed in the “*IC Compiler 2: HDP*” workshop)
- In a non-hierarchical design flow (this workshop) “gravity” should be turned off



2-33

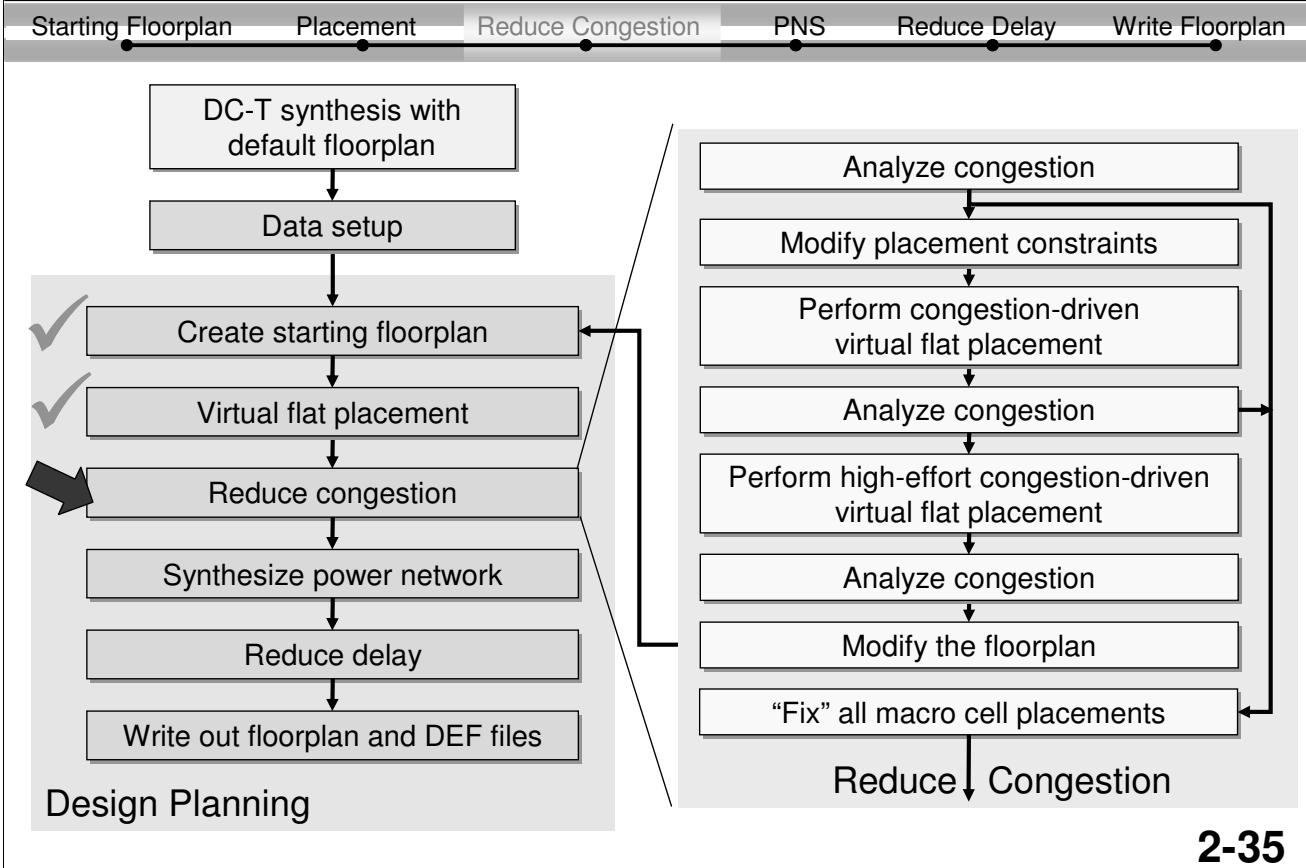
Summary: Virtual Flat Placement

Starting Floorplan Placement Reduce Congestion PNS Reduce Delay Write Floorplan



2-34

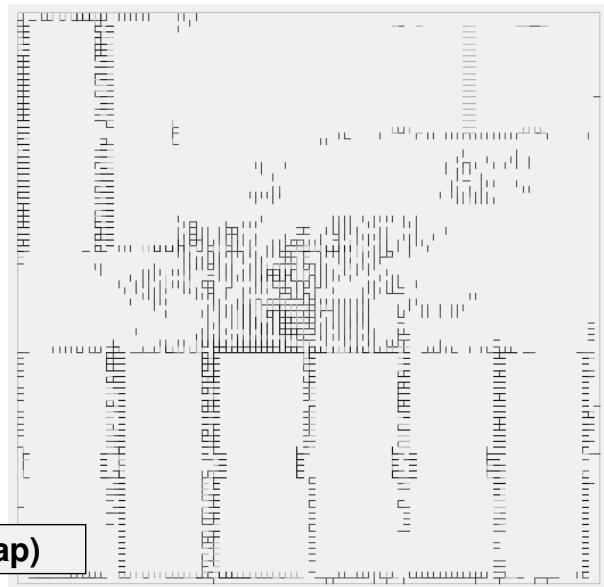
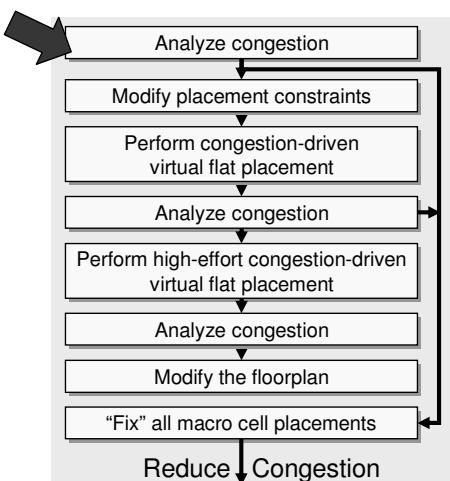
Reduce Congestion



2-35

Is the Design Congested?

Reduce congestion



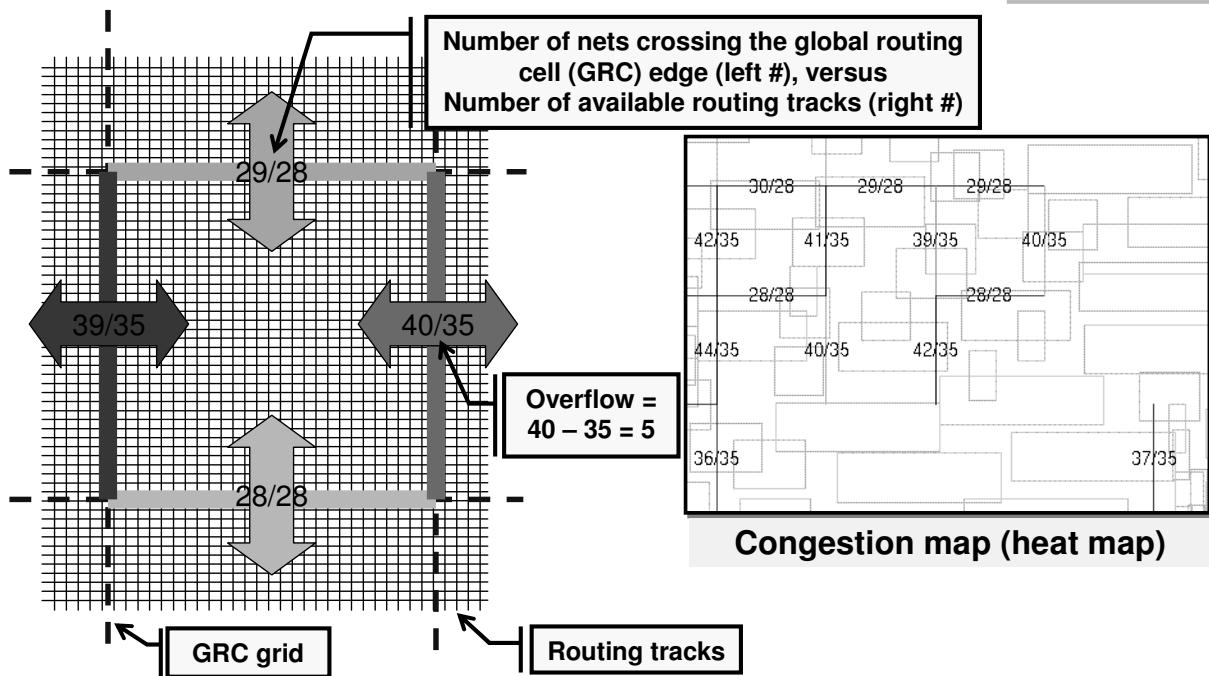
Congestion map (heat map)

```
route_global -congestion_map_only; # More accurate  
OR  
route_fp_proto -congestion_map_only \  
    -effort medium;      # Runs faster
```

2-36

Understanding the Congestion Calculation

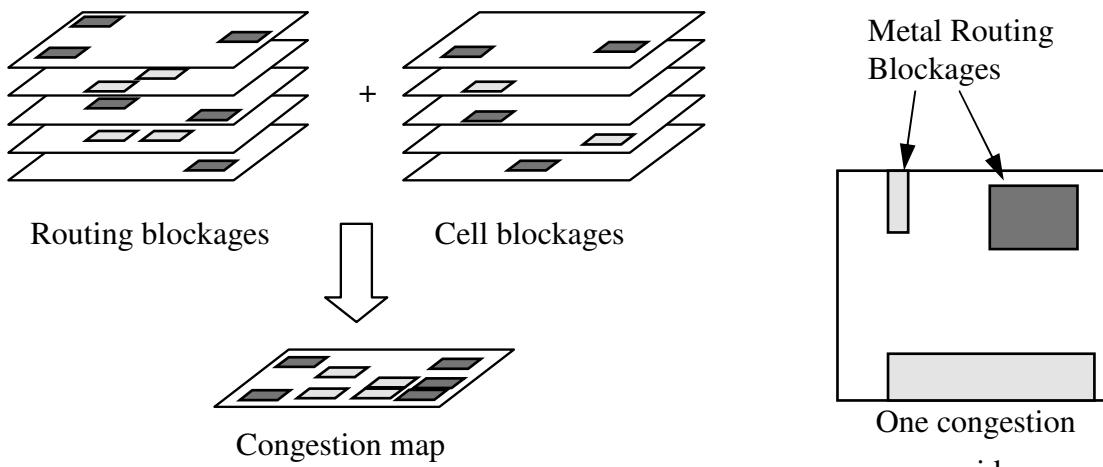
Reduce congestion



The size of the ‘overflow’ determines the highlighting color – larger overflows are brighter or ‘hotter’

2-37

GRC – The size of a global routing cell in general is double the height of a standard cell, squared. For congestion analysis, the tool works on a congestion grid. Each grid models the routing layers that overlap with that grid, producing a density plot that shows the level of routing congestion. To generate the congestion map, the tool combines all the declared routing blockages (floorplan and standard cell) to produce a single congestion map that is displayed in the GUI.



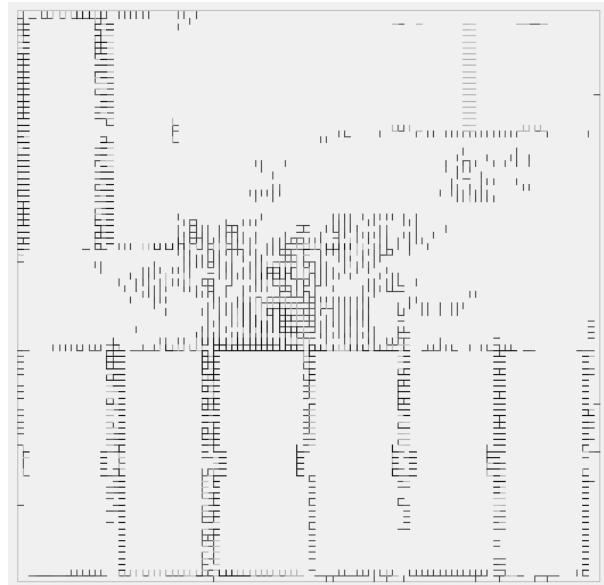
A textual congestion report is also available with `report_congestion`

Congestion Guidelines

Reduce congestion

Congestion can be serious or “un-acceptable” if one or more of the following are true:

- **Large or many “hotspots” in the congestion map**
 - Serious routability challenge
- **Any GRC “overflow” larger than around 10**
 - Possible non-routable nets
- **About ~2%, or more, GRC edges have an overflow**
 - Possible signal integrity or timing degradation issues
- **If congestion is not a problem move on to “PNS”, otherwise**



2-38

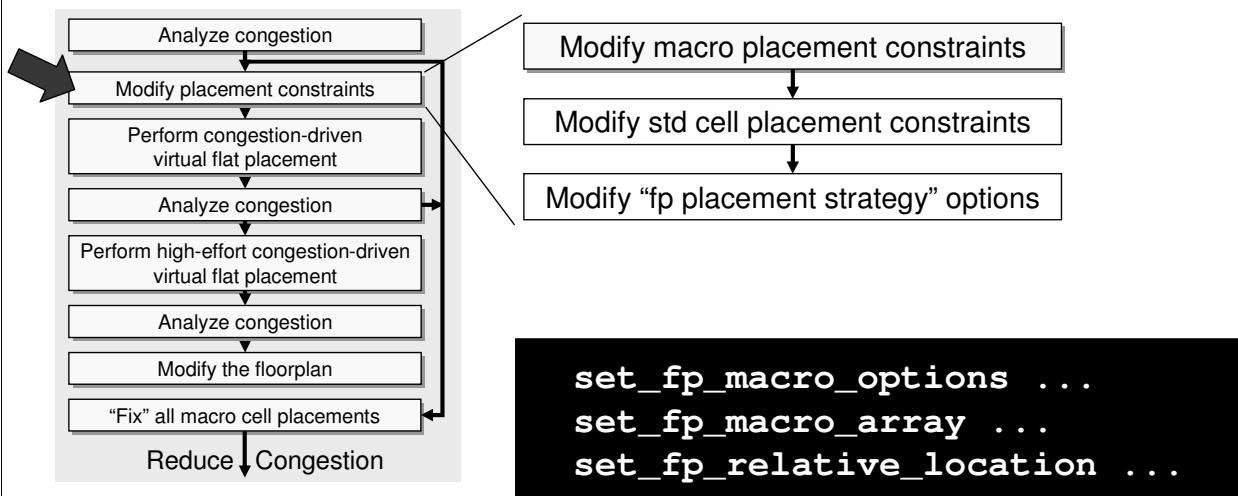
In the example below we see part of the log output after creating a global route congestion map. Specifically we're looking in the “Routing result” section, at the line that summarizes the GRC overflows for both horizontal and vertical directions. In this example the “Max” overflow is 13. This means that there is at least one GRC edge with an overflow of 13, for example “19/6”. This means that there are 13 routes that want to route through a specific GRC area which do not have any available routing resources. In practice, any maximum GRC overflow of around 10, or larger, has a very high probability of resulting in unroutable (shorts or opens) nets in that area.

The total number of GRC overflows is 2.73%. This means that 2.73% of the total number of GRC edges in the design have an overflow of 1 or more. In practice, if this number is around 2% or more (this is a very rough guideline and is very design-dependent), this can result in poor routing overall, which could manifest itself in signal integrity (e.g. crosstalk) problems or poor timing.

```
phase5. Routing result:  
phase5. Both Dirs: Overflow = 3621 Max = 13 GRCs = 2247 (2.73%)  
phase5. H routing: Overflow = 1724 Max = 13 (1 GRCs) GRCs = 1139 (1.38%)  
phase5. V routing: Overflow = 1897 Max = 8 (1 GRCs) GRCs = 1108 (1.34%)  
phase5. METAL     : Overflow = 1162 Max = 8 (3 GRCs) GRCs = 879 (2.13%)  
phase5. METAL2    : Overflow = 1826 Max = 6 (4 GRCs) GRCs = 1079 (2.62%)  
phase5. METAL3    : Overflow = 562 Max = 7 (2 GRCs) GRCs = 440 (1.07%)  
phase5. METAL4    : Overflow = 71 Max = 4 (1 GRCs) GRCs = 58 (0.14%)  
phase5. METAL5    : Overflow = 0 Max = 0 GRCs = 0 (0.00%)  
phase5. METAL6    : Overflow = 0 Max = 0 GRCs = 0 (0.00%)
```

Modify Macro Placement Constraints

Reduce congestion

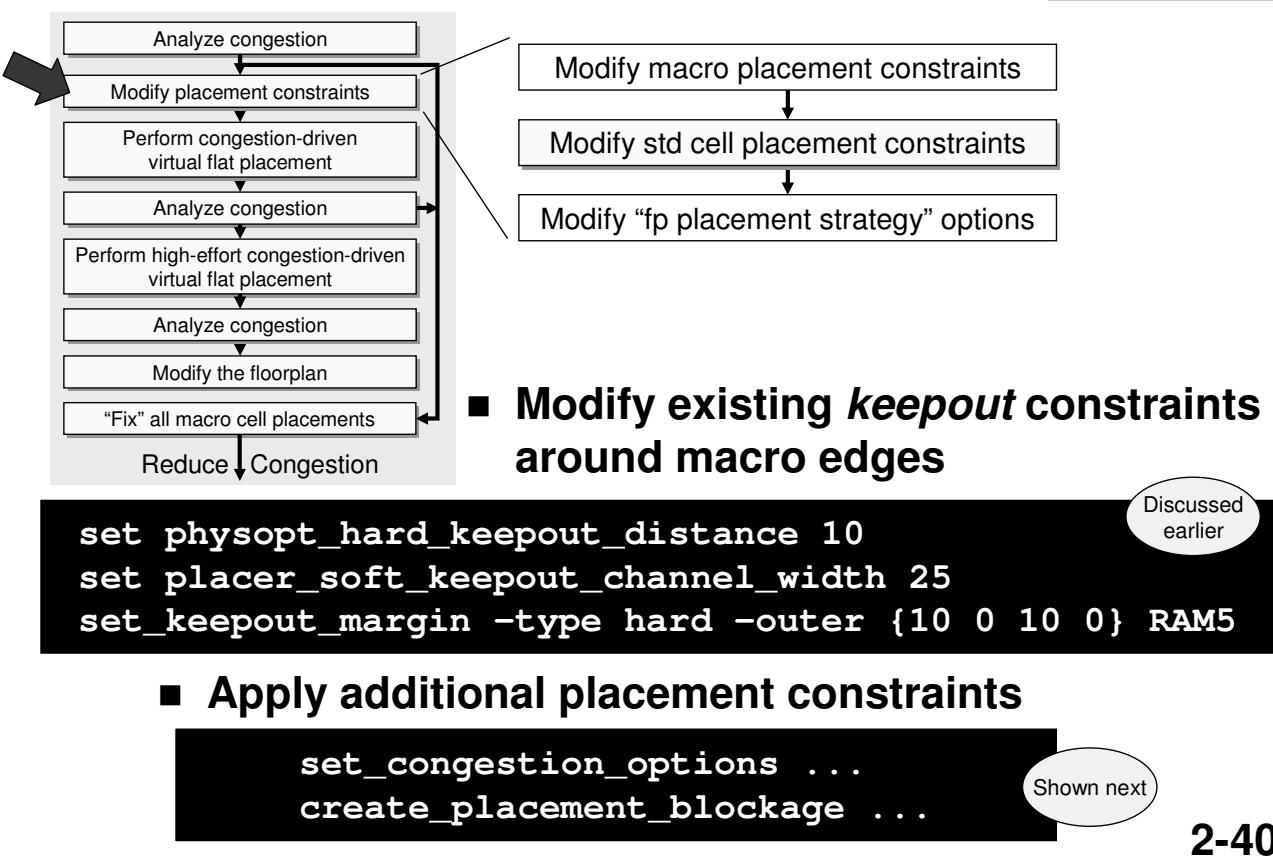


- Increase spacing between macros
- Align bus signal pins between macros
- Change orientation (rotate and/or mirror)

2-39

Apply Standard Cell Placement Constraints

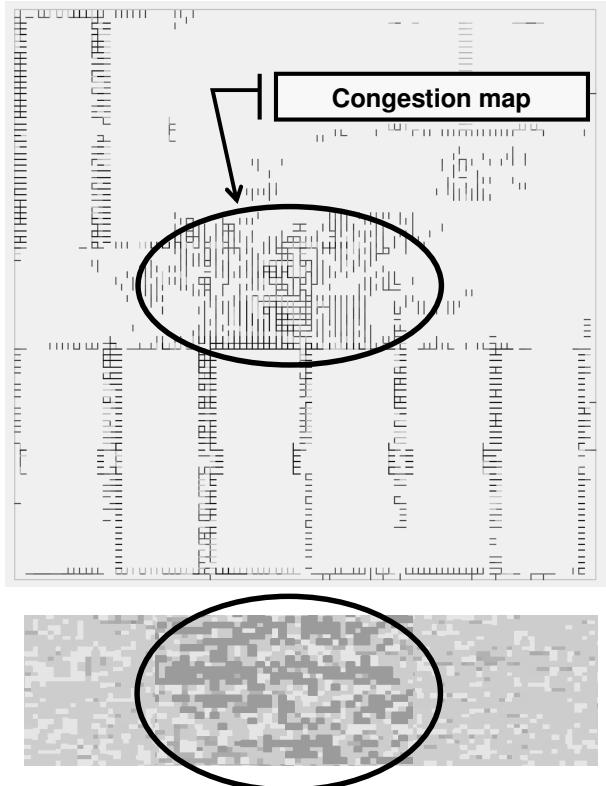
Reduce congestion



The “keepout” constraints were discussed in detail during the “Create Starting Floorplan” section, so will not be repeated here.

Is High Cell Density Causing Congestion?

Reduce congestion

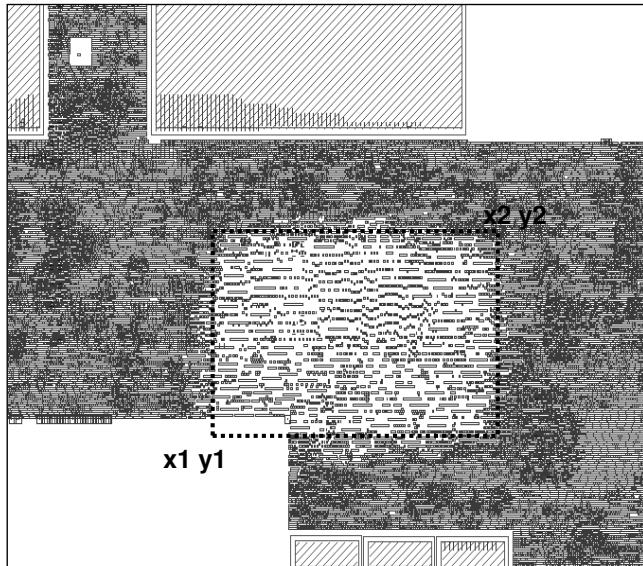


- Cell density can be up to 95% in any given area of the chip or block, by default
- High cell density may be causing high congestion
- Look at a cell density map
 - If cell density hotspots coincide with congestion hotspots → Lower cell density in that area

2-41

Reducing Cell Density Hotspots

Reduce congestion



```
set_congestion_options -max_util 0.4 \
-coordinate {x1 y1 x2 y2}
```

2-42

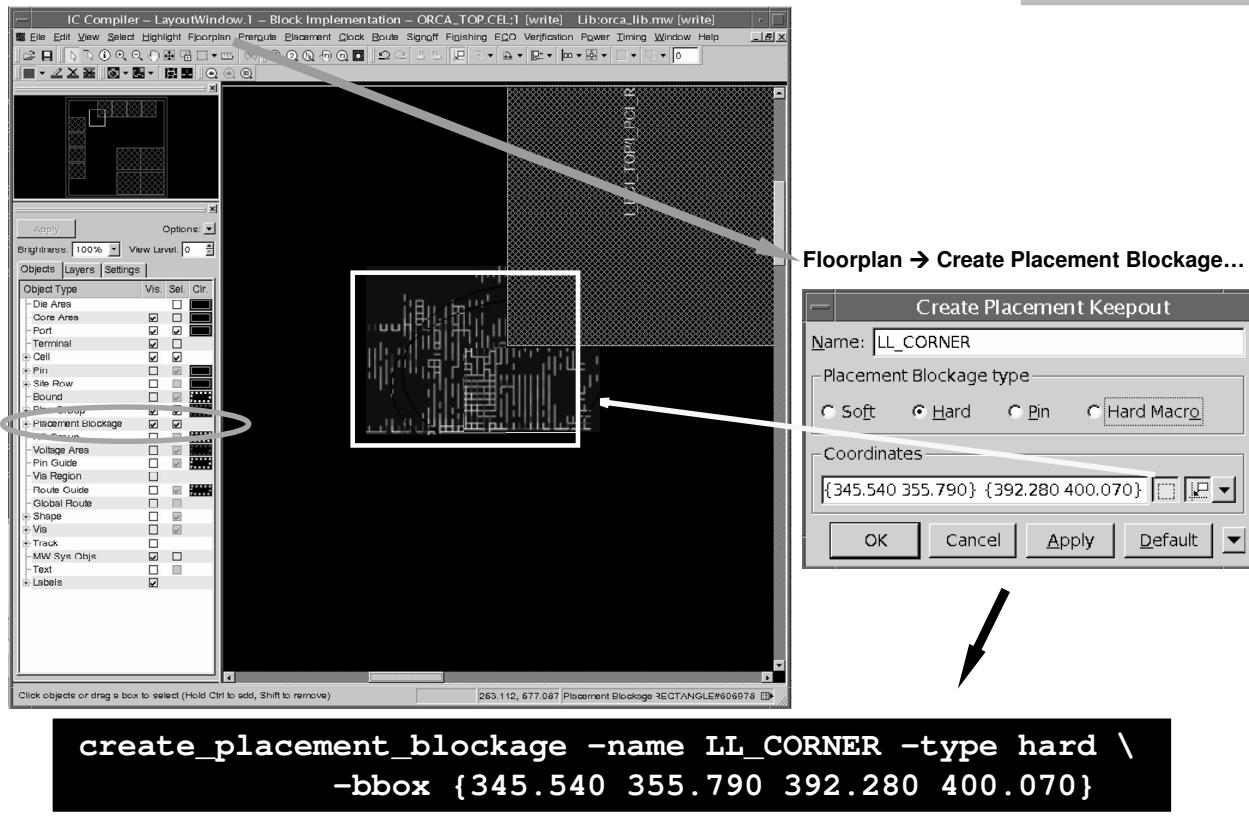
The argument to `-max_util` is specified as `0.0 - 1.0`, corresponding to 0-100%. The maximum utilization is 95% by default.

In the example above, `set_congestion_options` is used on a specific region that was identified as congested in a previous placement run. The tool can be forced to reduce the placement density in this region using the command as shown above (coordinates are in *microns*). Note however that areas around this region may in turn become more congested.

When using `set_congestion_options` without `-coordinate`, the density level is applied to the entire standard cell placement area. For example: Assume the initial overall placement utilization is 80%. This means that, by default, you can have "local" high density areas of up to 95%, and other areas of less than 80% density (peaks/valleys). If you lower the `max_util` to something less than 95%, but greater or equal to the overall utilization percentage, the peaks/valleys will be smoothed out. It is recommended to allow as HIGH a density as possible, to reduce the impact on timing.

Coordinate-based Placement Blockages

Reduce congestion



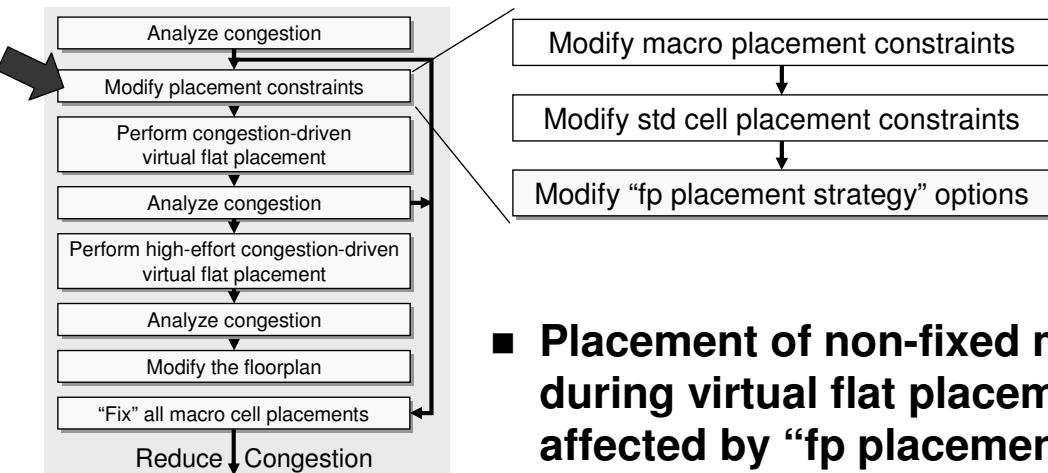
2-43

The following commands allow reporting and removal:

```
get_placement_blockages  
remove_placement_blockage
```

Modify “FP Placement Strategy” Options

Reduce congestion



- **Placement of non-fixed macros during virtual flat placement are affected by “fp placement strategy” options**

- **Up to now default options were used, except for `set_fp_placement_strategy -sliver_size 10`**
- **Modify macro placement strategy options, as appropriate, to help reduce congestion**

2-44

Report : report_fp_placement_strategy

Design CEL : ORCA.CEL;1

Date : Mon Nov 19 12:49:01 2007

*** Macro related parameters ***

`set_fp_placement_strategy -macro_orientation automatic | all | N`

current setting: automatic

default setting: automatic

`set_fp_placement_strategy -auto_grouping none | user_only | low | high`

current setting: low

default setting: low

`set_fp_placement_strategy -macro_setup_only on | off`

current setting: off

default setting: off

`set_fp_placement_strategy -macros_on_edge on | off`

current setting: off

default setting: off

...

1

Placement Strategy Options and Defaults

Reduce congestion

```
set_fp_placement_strategy
    -macro_orientation automatic | all | N
    -auto_grouping none | user_only | low | high
    -macro_setup_only on | off
    -macros_on_edge on | off
    -sliver_size <0.00>
    -snap_macros_to_user_grid on | off
    -fix_macros none | soft_macros_only | all
    -congestion_effort low | high
    -IO_net_weight <1.0>
    -plan_group_interface_net_weight <1.0>
    -voltage_area_interface_net_weight <1.0>
    -voltage_area_net_weight_LS_only on | off
    -spread_spare_cells on | off
    -legalizer_effort low | high
    -virtual_IPO on | off
```

Some examples
shown next

2-45

Macro Placement Strategy Examples

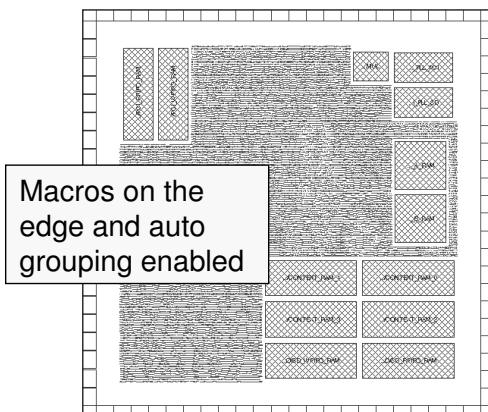
Reduce congestion

- Push macros to core edges for congestion control

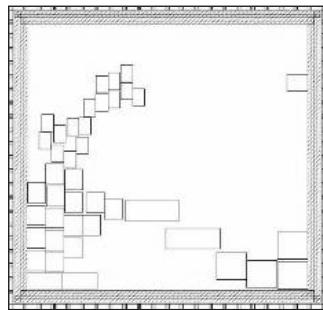
```
set_fp_placement_strategy -macros_on_edge on
```

- Group related macros into an array to improve timing and congestion

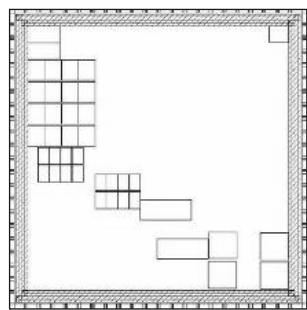
```
set_fp_placement_strategy -auto_grouping high
```



-auto_grouping none



-auto_grouping high

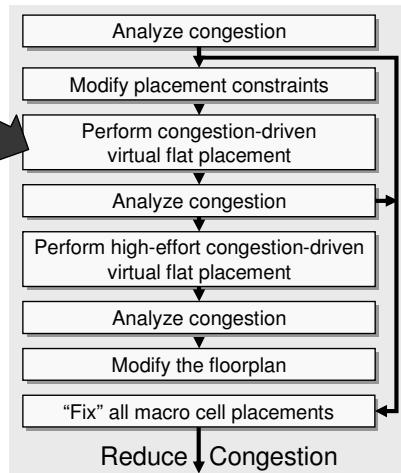


2-46

“related macros” are macros connected to the same nets.

Perform *Congestion-driven Placement*

Reduce congestion



- After modifying the various placement constraints perform another virtual flat placement
- Enable congestion-driven mode

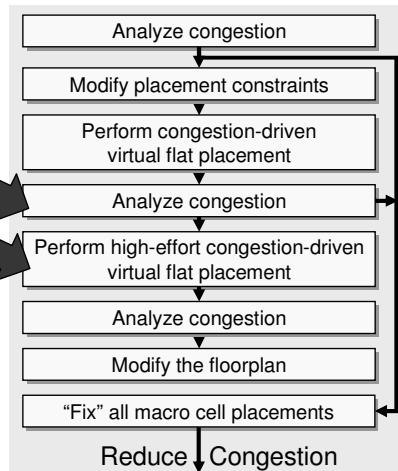
```
create_fp_placement -timing -no_hierarchy_gravity \
                     -congestion
```

Do not use congestion-driven mode on designs with no congestion!

2-47

Invoke the *High Effort* Congestion Strategy

Reduce congestion



If congestion is still not acceptable, perform *high-effort* congestion-driven placement:

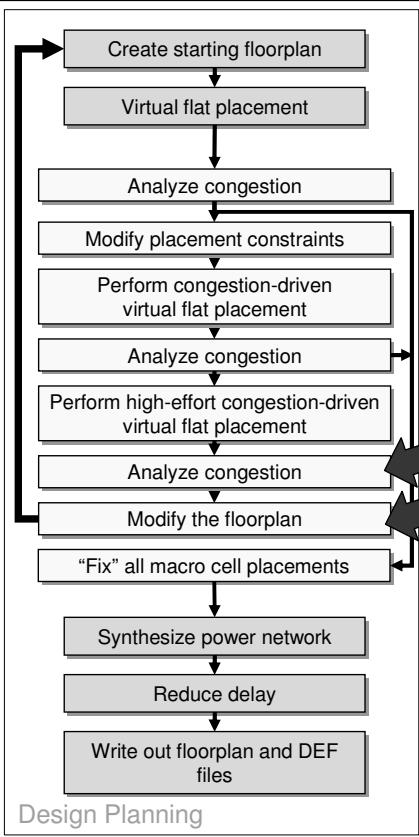
```
route_global -congestion_map_only; # OR  
route_fp_proto -congestion_map_only -effort medium  
  
set_fp_placement_strategy -congestion_effort high  
create_fp_placement -timing -no_hierarchy_gravity \  
-congestion
```

2-48

The -congestion_effort option is medium by default, if not explicitly set.

Modify the Floorplan

Reduce congestion



If congestion is still a problem start over with a modified floorplan:

- **Top-level pads or ports**
 - Spread, re-order or move to other sides
 - Change to a different metal layer
- **Core aspect ratio and size**
 - Make taller/wider to add more horizontal/vertical routing resources
 - Increase the core area to reduce utilization (cell density)
- **Modify power grid structure**
 - Use higher metal layers
 - Change width and/or spacing of straps

2-49

Large, wide power nets may cause problems as they act like barriers to placement.

You can determine this by running a trial where you ignore the blocking effect of power straps and see if placement works. Possible changes:

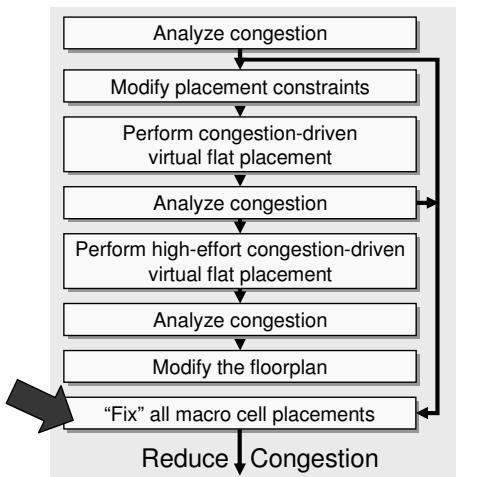
Change the power strap metal to a higher layer so that it does not impact cell placement.

Change the location of the power strap routes to avoid the congested area.

Change the power straps to be narrower and/or with more space between them.

“Fix” All Macro Cell Placement

Reduce congestion



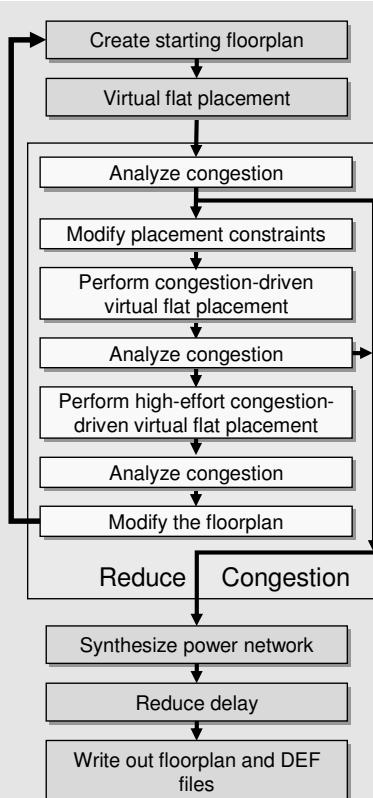
After all congestion issues are resolved, “fix” the placement of all macro cells to prevent any subsequent placement steps from moving them

```
set_dont_touch_placement [all_macro_cells]
```

2-50

Summary: Reduce Congestion

Reduce congestion



```

# Analyze congestion:
route_global -congestion_map_only; # OR
route_fp_proto -congestion_map_only -effort medium
# Modify macro placement constraints:
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
# Modify existing std cell keepout constraints:
set_physopt_hard_keepout_distance <#>
set_placer_soft_keepout_channel_width <#>
set_keepout_margin ...
# Apply additional std cell placement constraints:
set_congestion_options ...
create_placement_blockage ...
# Modify fp placement strategy options:
set_fp_placement_strategy ...
# Perform congestion-driven placement:
create_fp_placement -timing -no_hier -congestion
# Analyze congestion, as above. If still congested
# enable high-effort congestion mode:
set_fp_placement_strategy -congestion_effort high
create_fp_placement -timing -no_hier -congestion
# Analyze congestion and if still a
# problem → re-floorplan.
# Fix the placement of all macro cells:
set_dont_touch_placement [all_macro_cells]
  
```

Design Planning

2-51

Test For Understanding (1 of 2)



- 1. Circle the correct statement(s) about what `create_fp_placement` does by default:**
 - a. Optimizes logic (cell sizing, buffering) to improve timing
 - b. Legally places std cells and non-fixed macros to minimize wire length
 - c. Optimizes placement to improve timing
 - d. Clumps cells from the same logical hierarchy together
- 2. Comparing “9/1” and “20/12” in a congestion map:**
 - a. They are equally bad
 - b. 20/12 is worse because both numbers are much larger
 - c. 9/1 is worse the ratio is much larger
 - d. Can not tell without also comparing their “heat” colors
- 3. What are some common areas for congestion to occur?**

2-52

1. **B and D.** `create_fp_placement` does not do any logic or netlist optimization, but can optimize placement for timing if timing-driven mode is enabled with `-timing`. The resulting placement is always legalized. By default, “gravity” is enabled, which clumps cells from the same logical hierarchy together.
2. A. They both have an overflow of 8, and therefore the same “heat” color (white, by default).
3. Between macro cells, near macro edges and corners, and in high cell density areas.

Test For Understanding (2 of 2)

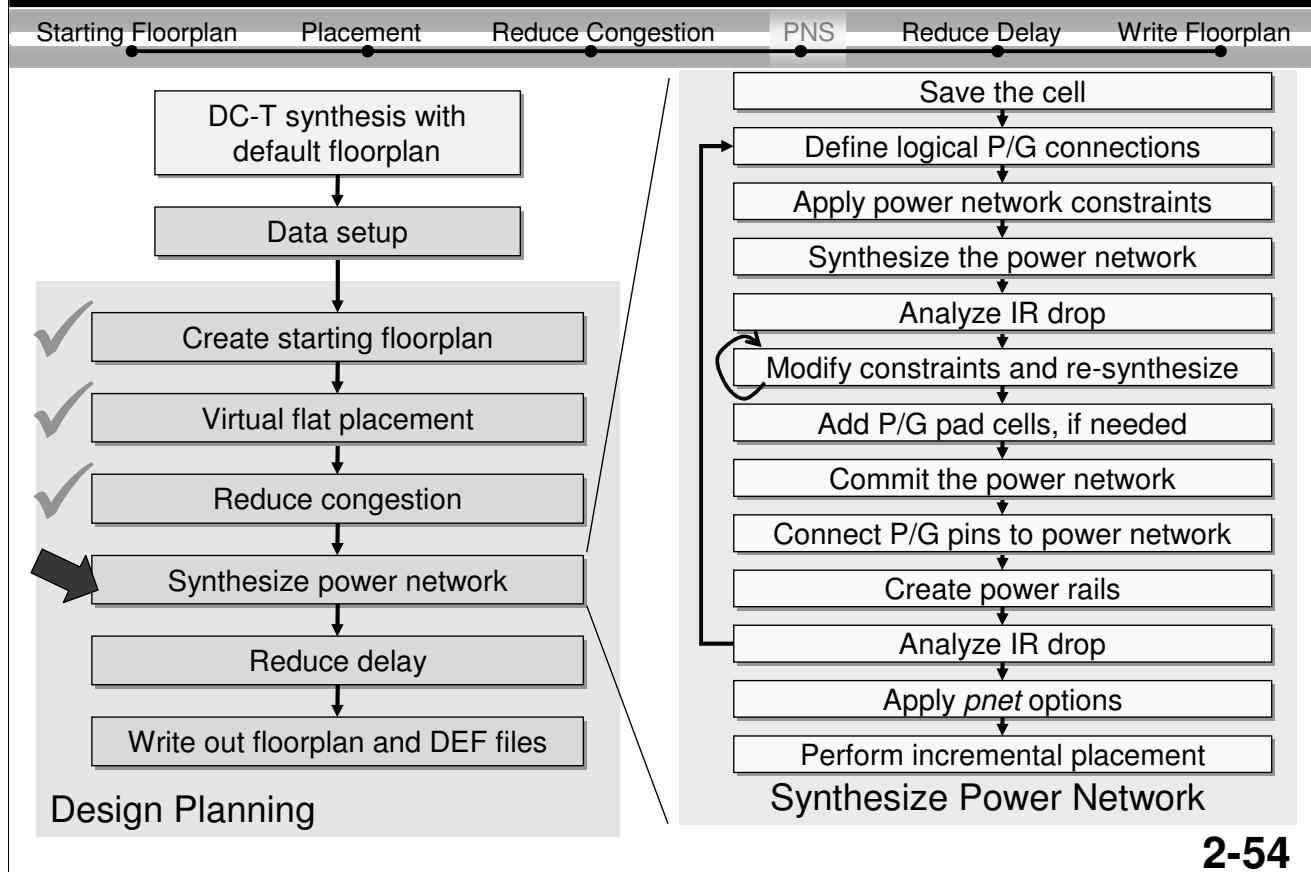


- 4. Macro placement can be controlled by:**
 - a. Manually placing macros prior to virtual flat placement
 - b. Specifying general “placement strategy parameters” prior to virtual flat placement
 - c. Applying “macro placement constraints” (options, arrays, relative location) after the last virtual flat placement
 - d. All of the above
- 5. If you can not achieve acceptable congestion after modifying placement constraints and parameters, as well as performing high effort congestion-driven virtual flat placement, what should you do?**

2-53

4. B. Manually placed macros must also be “fixed” (set_dont_touch_placement) otherwise vF placement is free to move them. Macro placement constraints must also be applied prior to vF placement and are implemented during placement, not independently after placement.
5. Start over with a modified floorplan: Change the pad or port locations or metal layers; Change the core size and/or shape; Change the power grid structure.

Synthesize the Power Network (PNS)



2-54

Power Network Synthesis (PNS)

PNS

Constraint-based PNS automates a typically tedious and iterative manual task

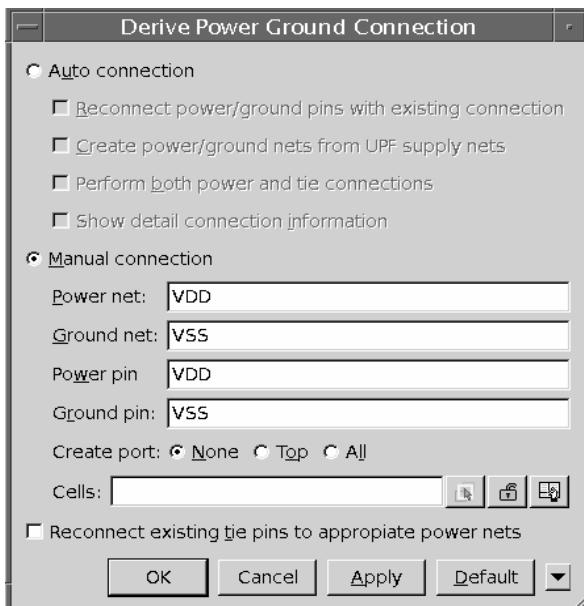
- Creates DRC/ERC-clean macro power rings and power straps
- Calculates the width and the number of power straps to meet power and IR-drop constraints
- Displays an IR-drop “heat map” for easy analysis
- Allows repeated “what-if” analyses before “committing” the power network structure
- After “committing” there is no easy way to “undo” the power network → Save the cell before PNS

```
save_mw_cel -as DESIGN_pre_pns
```

2-55

Define Logical Power/Ground Connections

PNS



■ Make sure that logical P/G connections are defined prior to PNS

- This may have already been done during *Data Setup* and/or creation of pad P/G rings

■ One command for each power/ground net

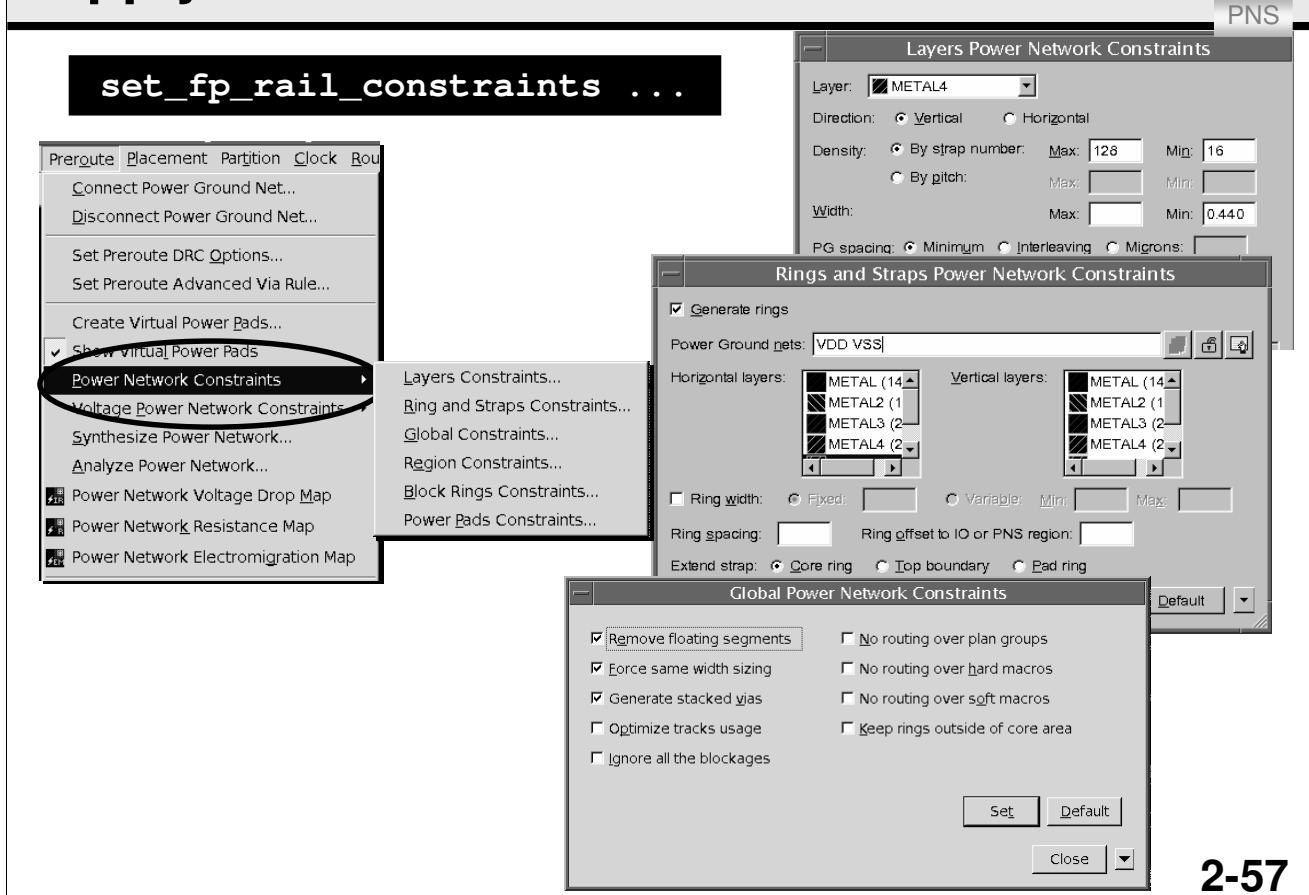
```
derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS
derive_pg_connection -power_net VDDO -power_pin VDDO -ground_net VSSO -ground_pin VSSO
derive_pg_connection -power_net VDDQ -power_pin VDDQ -ground_net VSSQ -ground_pin VSSQ
```

2-56

GUI: Preroute → Derive PG Connection ...

Note: Need to re-run this command each time the user "manually" introduces new cells (e.g. create_cell, eco_netlist, read_mw_eco_list, etc.)

Apply Power Network Constraints

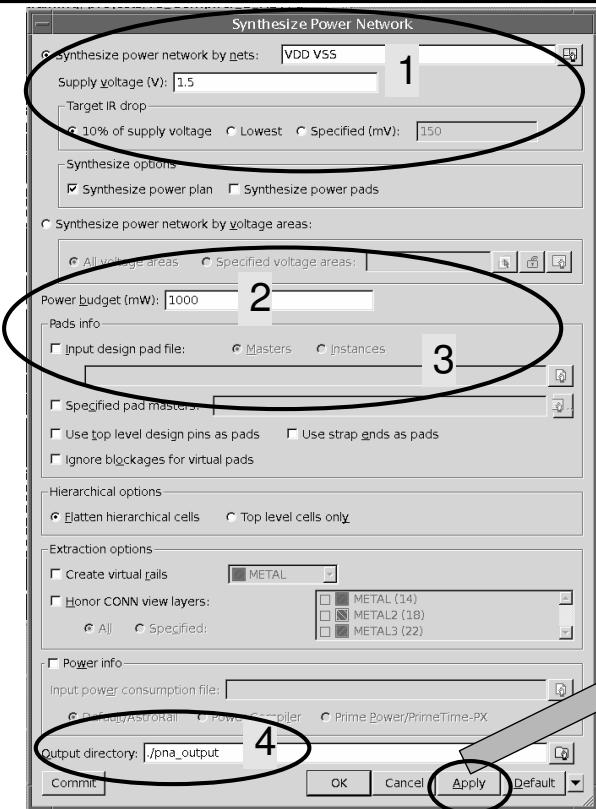


2-57

```
set_fp_rail_constraints
[-add_layer | -remove_layer | -remove_all_layers | -set_ring |
 -skip_ring | -set_global]
[-layer layer] [-direction vertical | horizontal]
[-max_strap number] [-min_strap number]
[-max_width distance] [-min_width distance]
[-spacing distance | minimum | interleaving]
[-offset distance]
[-nets nets]
[-horizontal_ring_layer layer] [-vertical_ring_layer layer]
[-ring_width distance]
[-ring_max_width distance] [-ring_min_width distance]
[-ring_spacing distance] [-ring_offset distance]
[-extend_strap core_ring | boundary | pad_ring]
[-keep_floating_segments]
[-no_stack_via] [-no_same_width_sizing]
[-optimize_tracks]
[-keep_ring_outside_core]
[-no_routing_over_hard_macros] [-no_routing_over_soft_macros]
[-ignore_blockages]
```

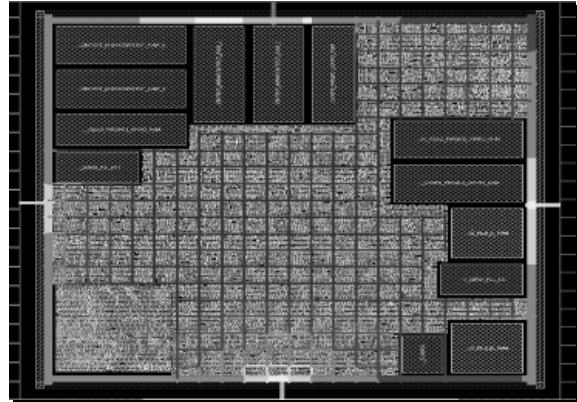
Synthesize and Analyze the Power Network

PNS



PNS calculates the required number of straps based on provided constraints. Heat map aids the analysis

Preview of power plan with IR-drop heat map
– no actual routes created yet



synthesize_fp_rail ...

2-58

GUI: Preroute → Synthesize Power Network

Make sure to specify:

1. Power/Ground net pair and target IR drop
2. Core area power budget of the synthesized nets
3. Power pad information
4. Output directory

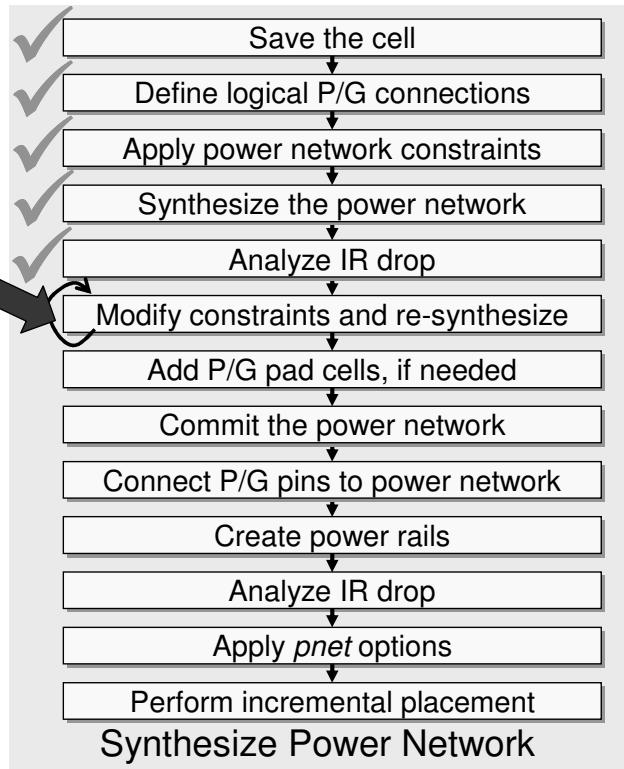
Note: If the power network is manually designed, or is designed by a 3rd party tool and loaded into IC Compiler, the user can perform explicit power network analysis (PNA) with:

TCL: analyze_fp_rail ...

GUI: Preroute → Analyze Power Network

Modify Constraints and Re-synthesize

PNS



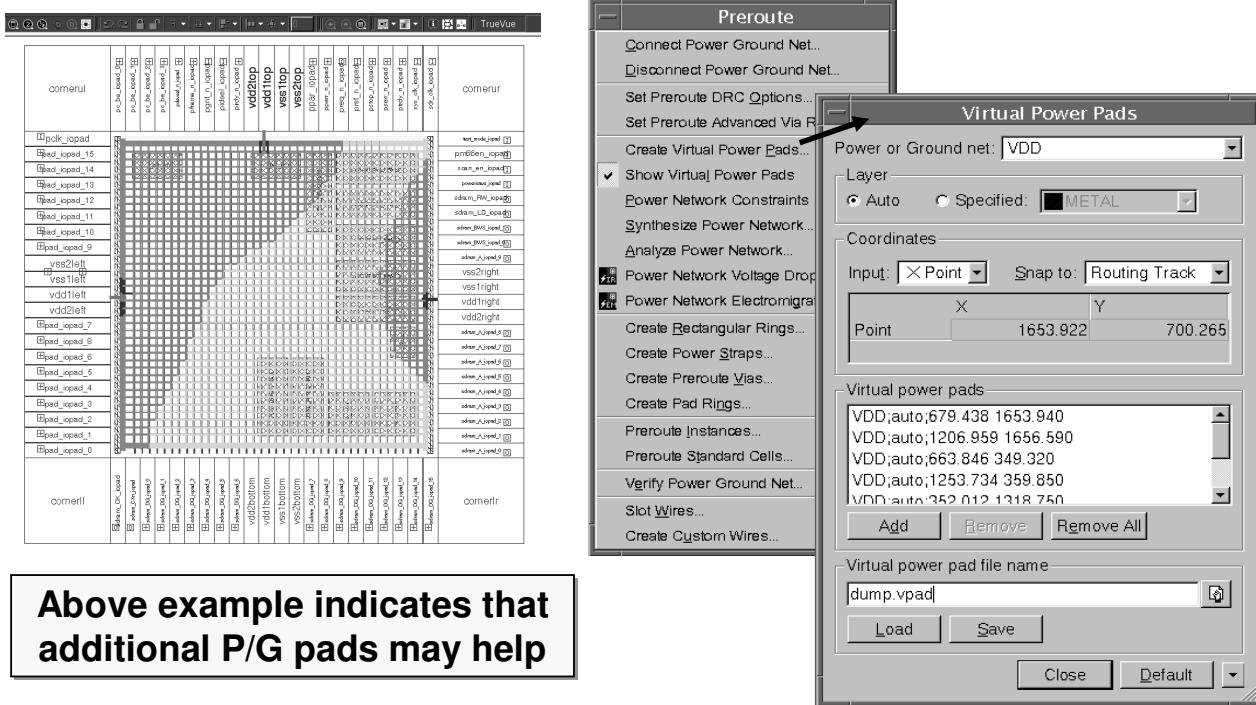
If the maximum IR drop is not acceptable

- Modify power network constraints
- Re-synthesize
- Repeat as needed

2-59

Create Virtual Power/Ground Pads if Needed

PNS



Above example indicates that additional P/G pads may help

```
create_fp_vitual_pad ...
# Then "Apply" PNS to update heat map
```

2-60

Use the mouse to define the center coordinates of virtual P/G pads, as needed.

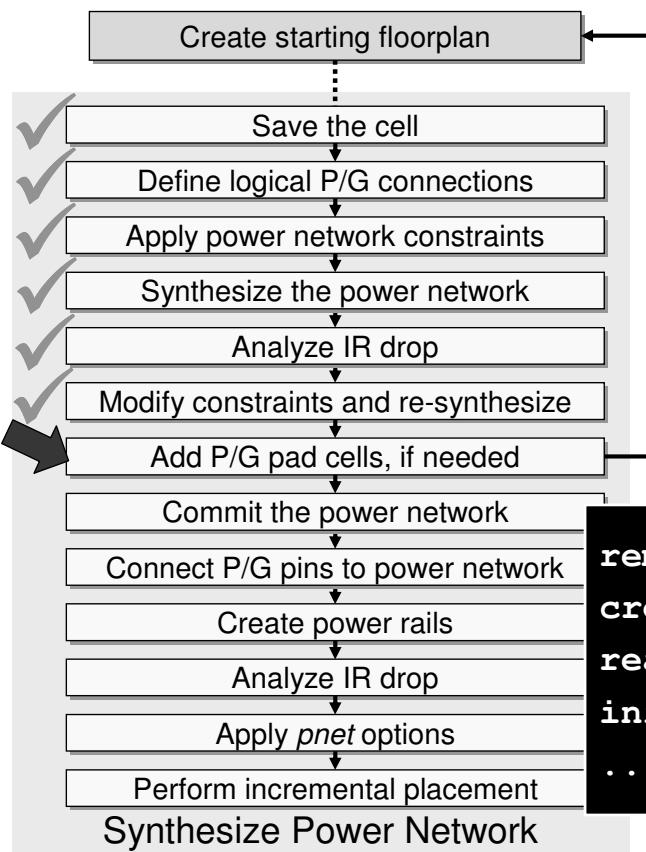
As the virtual pads are created, they appear in the “Virtual power pads” list box. *Add*, *Remove* and *Remove All* buttons allow editing the list of virtual pads. Once the list is complete, *Load* it to see the effect on IR-drop, and/or *Save* it to a file for future analysis.

The virtual pads are saved with the CEL when the design is saved. Virtual pads can be removed with `remove_fp_virtual_pad`.

The .vpad file is useful for editing and documentation of new pads that will later be physically instantiated.

Add Additional P/G Pads to TDF and Re-load

PNS



If additional P/G pads are required

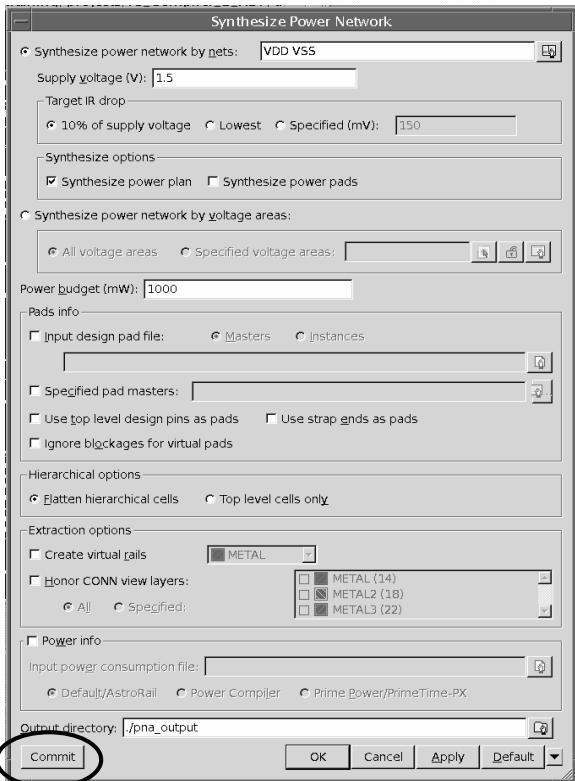
- Add them to the TDF file
- Start over and create a new starting floorplan

```
remove_io_constraints  
create_cell ...  
read_io_constraints <TDF_file>  
initialize_floorplan ...  
...
```

2-61

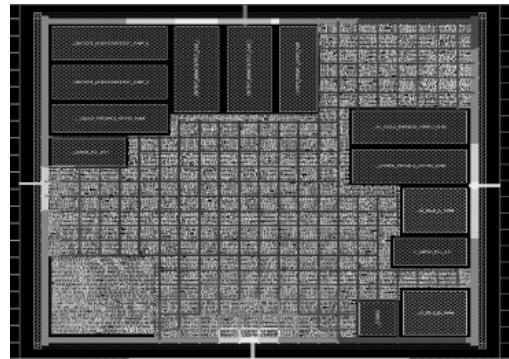
Commit the Power Network

PNS

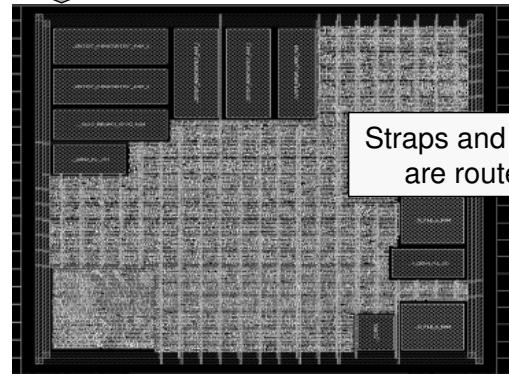


commit_fp_rail

Preview (Apply)



Build (Commit) the power network

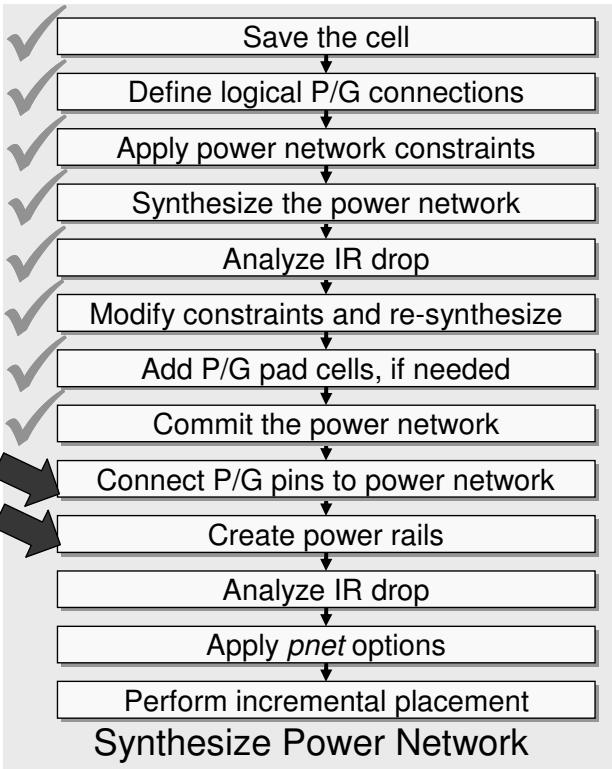


Straps and rings
are routed

2-62

Connect P/G Pins and Create Power Rails

PNS



■ **Connect P/G macro pins and pad pins to core rings and straps**

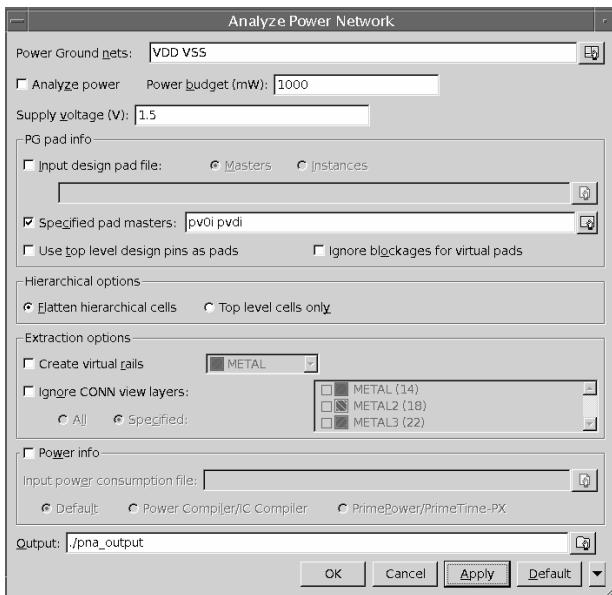
■ **Create power rails along the standard cell placement rows**

```
preroute_instances  
preroute_standard_cells \  
    -fill_empty_rows    \  
    -remove_floating_pieces
```

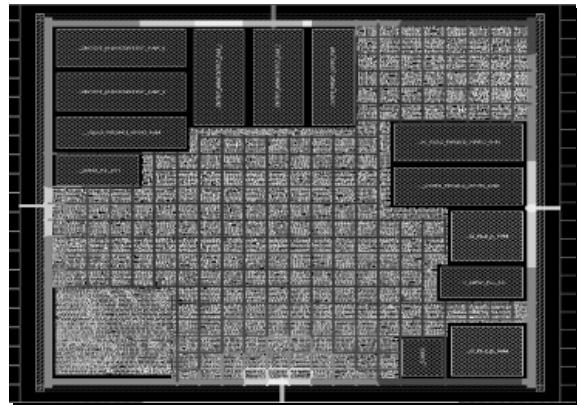
2-63

Analyze the Power Network

PNS



After “commit” and after P/G pins and rails have been pre-routed → Analyze the power network for a more accurate IR-drop map



```
analyze_fp_rail ...
# If IR-drop is not acceptable:
close_mw_cel
open_mw_cel DESIGN_pre_pns
# Begin a new PNS flow
```

2-64

GUI: Preroute → Analyze Power Network

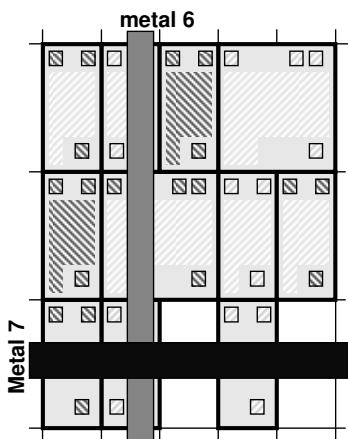
Make sure to specify:

1. Power/Ground net pair and target IR drop
2. Core area power budget of the synthesized nets
3. Power pad information
4. Output directory

Apply Power Net Placement Blockages

PNS

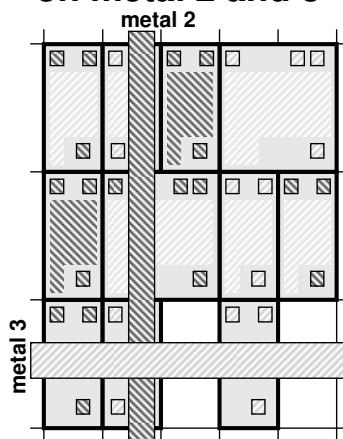
No pnet blockage



All cells allowed under P/G straps

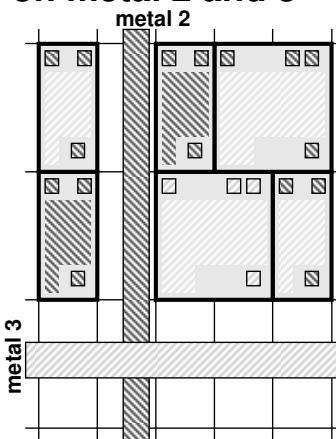
More uniform cell density →
Less overall congestion

Partial pnet blockage on metal 2 and 3



Cells allowed under
P/G nets, with min
pin-to-strap spacing

Complete pnet blockage on metal 2 and 3



No cells allowed under P/G straps

Less congestion along P/G nets

```
set_pnet_options -partial {metal2 metal3}
set_pnet_options -complete {metal2 metal3}
```

2-65

No blockage: In most designs, P/G straps are routed on *metal 4* or higher layers. Since most standard cell pins are on *metal 1* (in rare instances, *metal 1* and *metal 2*), there is usually no reason, in these designs, to prevent standard cells from being placed under the P/G routes. The router should be able to easily route to the pins under the straps. By allowing cells under the straps, a more uniform cell density can be achieved during placement, which helps to reduce overall congestion. In a core-limited design, this also allows a smaller die size.

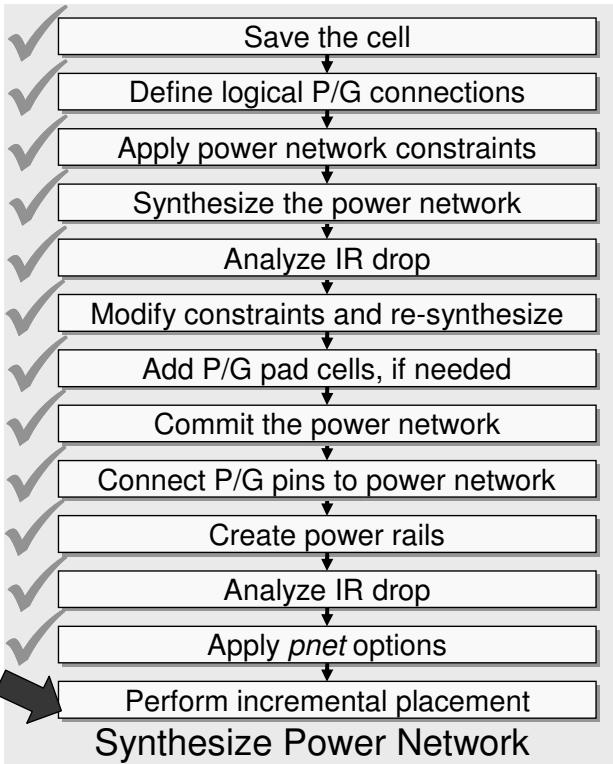
Partial blockage: Some designs may require the use of lower level metal (e.g. *metal 2* and *metal 3*) for P/G routes. In order to achieve better overall congestion, the user can specify “partial” pnet blockages for these lower metal layers. This allows IC Compiler to place standard cells under the metal straps, while ensuring that no DRC violations (minimum spacing, shorts) occur between the cell pins and the straps. The trade-off is that this may make it more difficult for the router to access the pins near the P/G straps, which would show up as congestion along these straps.

Complete blockage: To prevent congestion issues along the metal straps, the user can specify “complete” pnet blockages on the lower-level metal layers. This strategy increases utilization (since less of the core area is available for placement), creates less uniform cell density, which may increase congestion overall. In the rare case of a design with very low utilization (e.g. a pad-limited design), congestion is typically not a concern. In this case, if “complete” pnet blockage is applied, more optimal routing (fewer vias and/or shorter traces) may be achievable for pins near the straps (which would otherwise be placed under the straps).

Note: These constraints do not affect FIXED cells (e.g. fixed RAM, IP or macro cells). If these fixed cells are placed under power nets, applying a “complete” blockage will not move them out during placement, and a “partial” blockage will not move them to fix shorts. You must visually check macros and fix “manually” during the design planning phase.

Perform Incremental Virtual Flat Placement

PNS



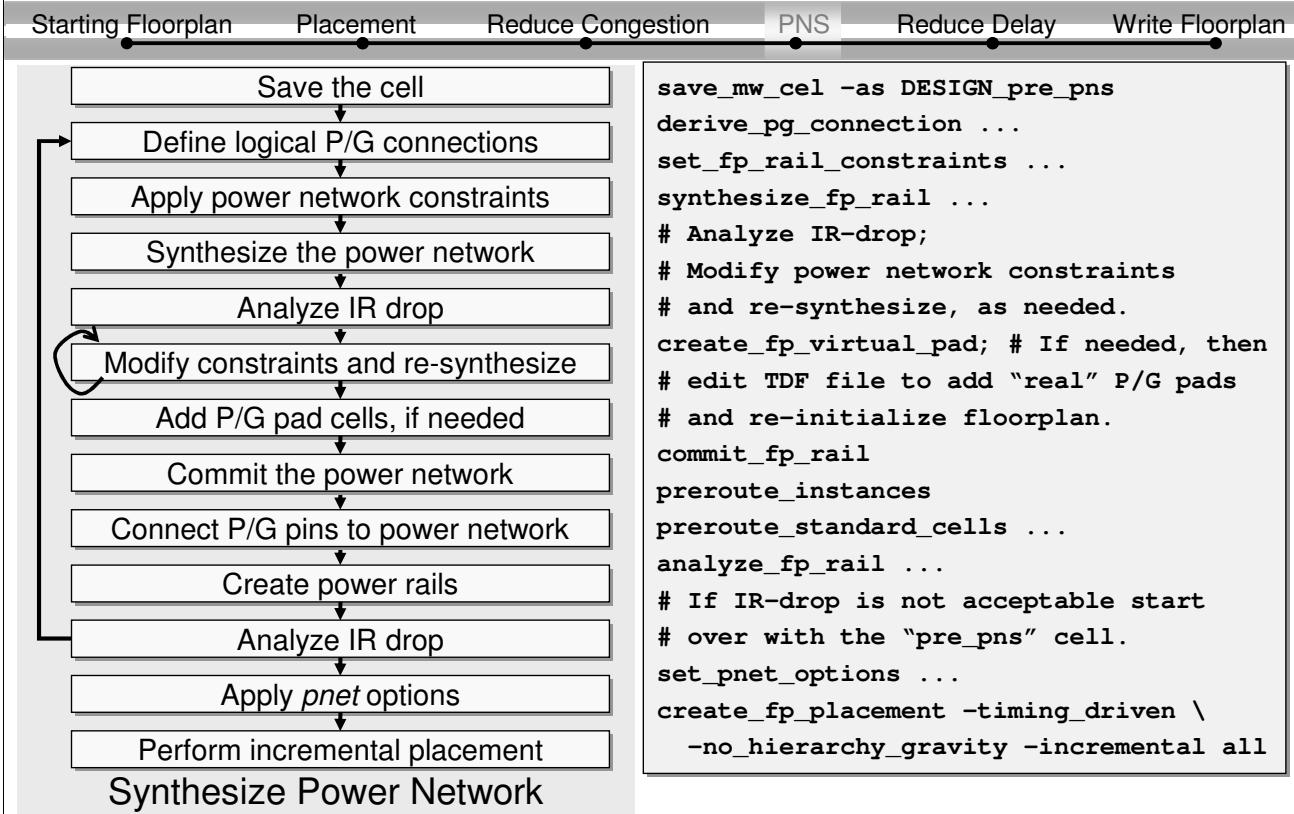
Perform an incremental placement

- Standard cells are placed taking into account the power network

```
create_fp_placement \
    -timing_driven \
    -no_hierarchy_gravity \
    -incremental all
```

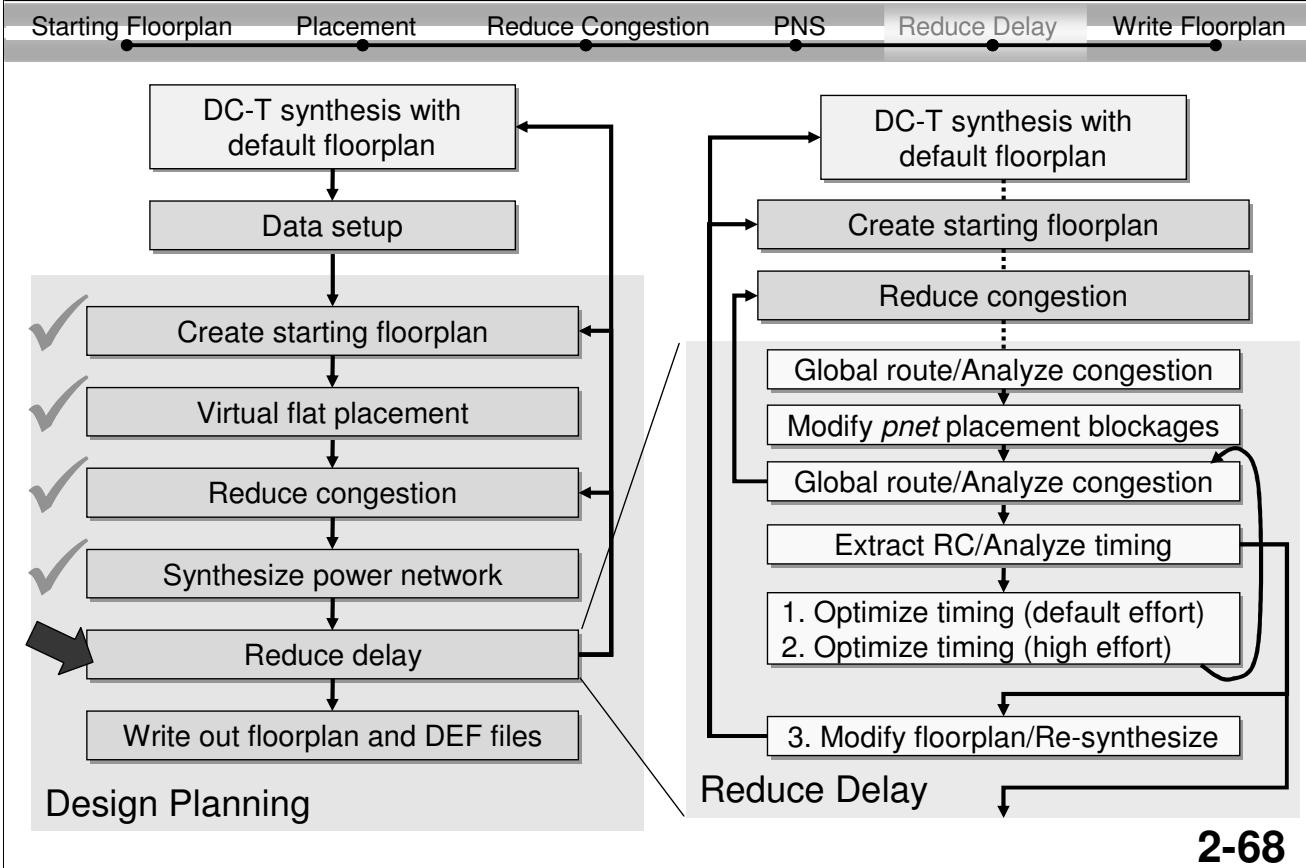
2-66

Summary: Synthesize the Power Network



2-67

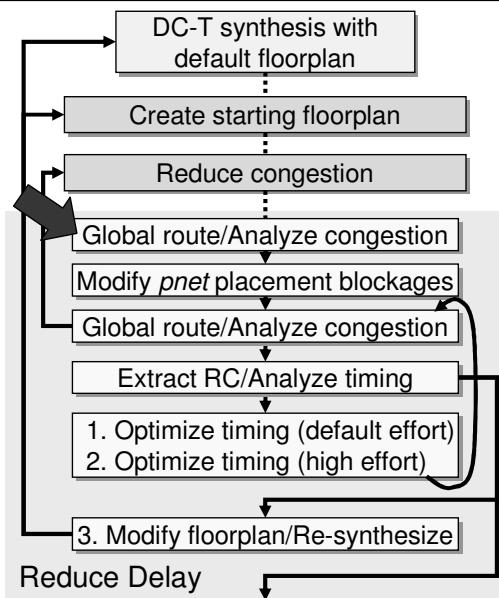
Reduce Delay



2-68

Global Route and Analyze Congestion

Reduce Delay



- Before extracting net RC parasitics for timing analysis perform an actual global route (without `-congestion_map_only`)

```
route_global; # OR  
route_fp_proto -effort medium
```

- By default timing analysis is based on “virtual route”
- Global route allows more accurate timing analysis

- Analyze the congestion map

2-69

GUI: Route → Global Route ...

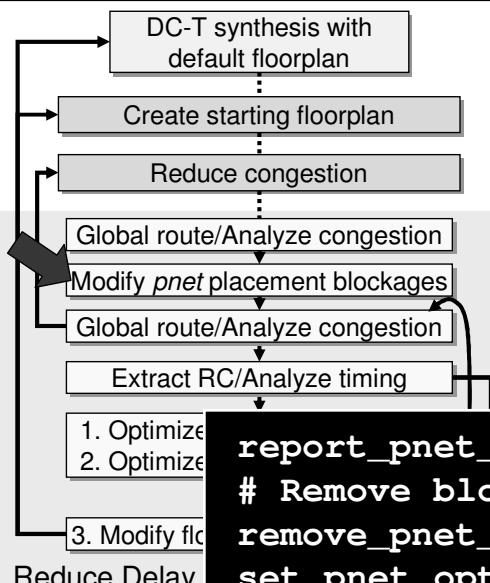
Global route is the first of several steps performed during the routing phase of the design:
Global Route; Track Assign; Detail Route; Search and Repair.

By default timing analysis is performed based on a “virtual route”, which is an orthogonal route estimate comprised of an “average” RC characteristics of all available metal layers.

With a global route the actual metal layer and routing path through GRC cells are defined, enabling more accurate RC extraction, and therefore more accurate timing analysis.

Modify Power Net Placement Blockages

Reduce Delay



Reduce Delay

- If congested, check and modify power net (*pnet*) placement blockage options, as needed:
 - If *complete* blockages exist, remove or change to *partial*
- Perform incremental placement

```
report_pnet_options
# Remove blockages:
remove_pnet_options OR
set_pnet_options -none {M6 M7}
# Modify existing options:
remove_pnet_options
set_pnet_options -partial {M2 M3}
create_fp_placement -timing_driven \
-no_hierarchy_gravity -incremental all
```

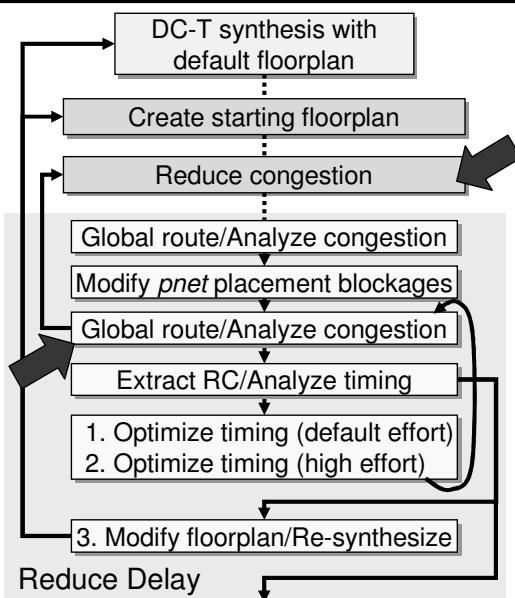


2-70

Changing a “complete” blockage to “partial”, or removing it altogether (no blockage), allows standard cells to be placed under those power straps, which increases the utilization. This may reduce congestion in the core area. It may create some congestion along the power straps.

Return to “Reduce Congestion”, if Needed

Reduce Delay

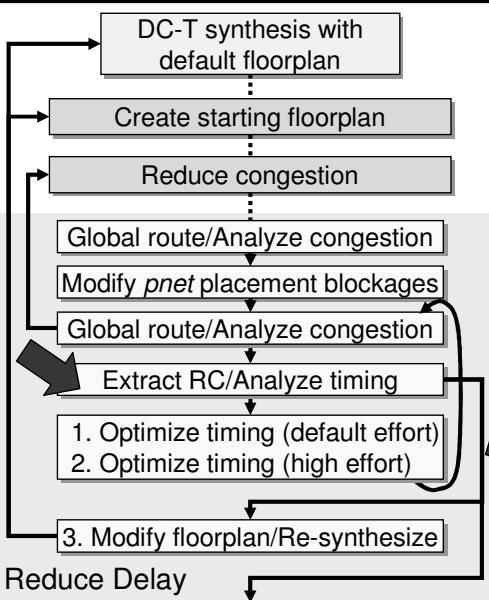


- Perform another global route and analyze congestion
- If modifying *pnet* options is not applicable or does not fix the congestion issues, return to the “Reduce Congestion” step
- Once congestion is under control continue with RC extraction and timing analysis

2-71

Extract Net Parasitics and Analyze Timing

Reduce Delay



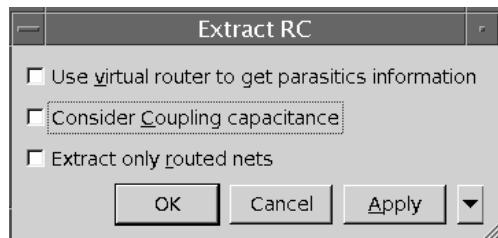
■ Extract net parasitic RCs

- Uses TLU-plus and global routing for improved timing accuracy

■ Generate a timing report

- Acceptable timing:
 $WNS < 15\text{-}20\%$ of required delay
- If acceptable skip to “Write Floorplan”

```
extract_rc  
report_timing
```



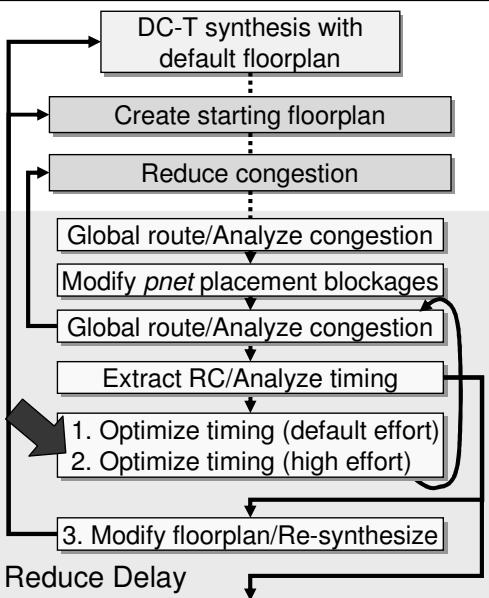
2-72

GUI: Route → Extract RC ...

The “acceptable timing” recommendation above is a general recommendation, not a rule. In general IC Compiler should be able to eliminate violations of up to 15-20% during placement, CTS, routing, and related optimizations.

Perform In-Place Optimization

Reduce Delay



- If the WNS > 15-20% perform *in-place optimization*

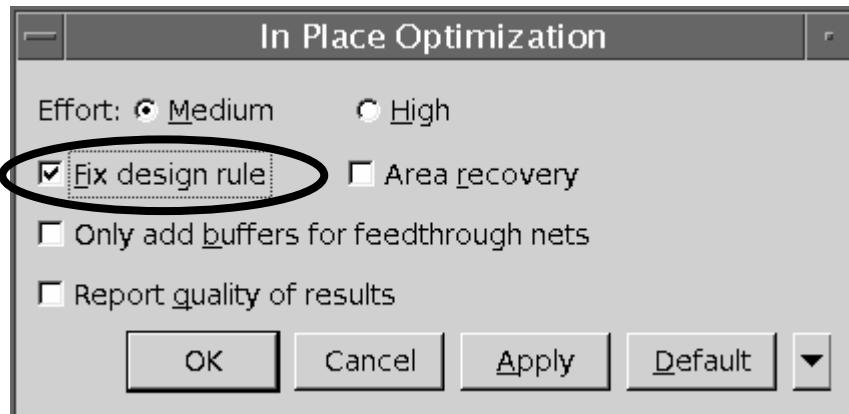
```
optimize_fp_timing -fix_design_rule
<-effort high>
```

- Repeat global routing, RC extraction and timing analyses
- If timing is still not acceptable, repeat timing optimization with *high effort*
- Repeat timing analyses steps

- ❖ Performs cell sizing, buffer insertion and AHFS
- ❖ Improves timing and DRC violations
- ❖ Legalizes placement

2-73

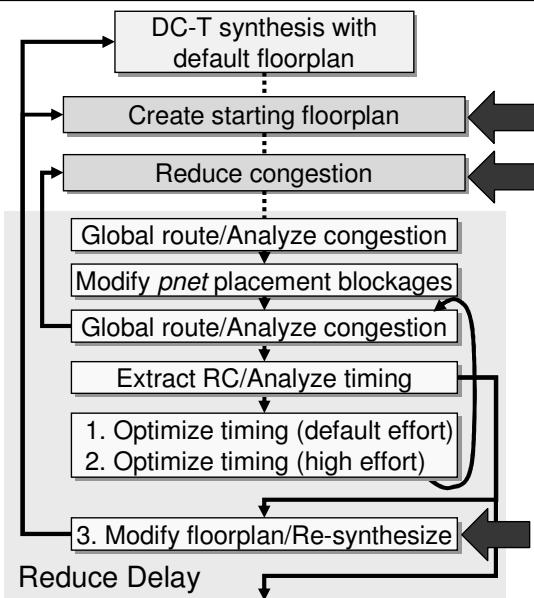
GUI: Timing → In Place Optimization ...



AHFS: Automatic high fanout synthesis is an optimization algorithm for handling nets with large fanouts (≥ 100).

Modify Floorplan or Re-Synthesize, if Needed

Reduce Delay



If the timing violations are still unacceptably large after *high-effort optimization*:

■ **Modify the floorplan, or...**

■ **Re-synthesize the design with**

- Better constraints
- Better partitioning

2-74

If timing is still “unacceptable”:

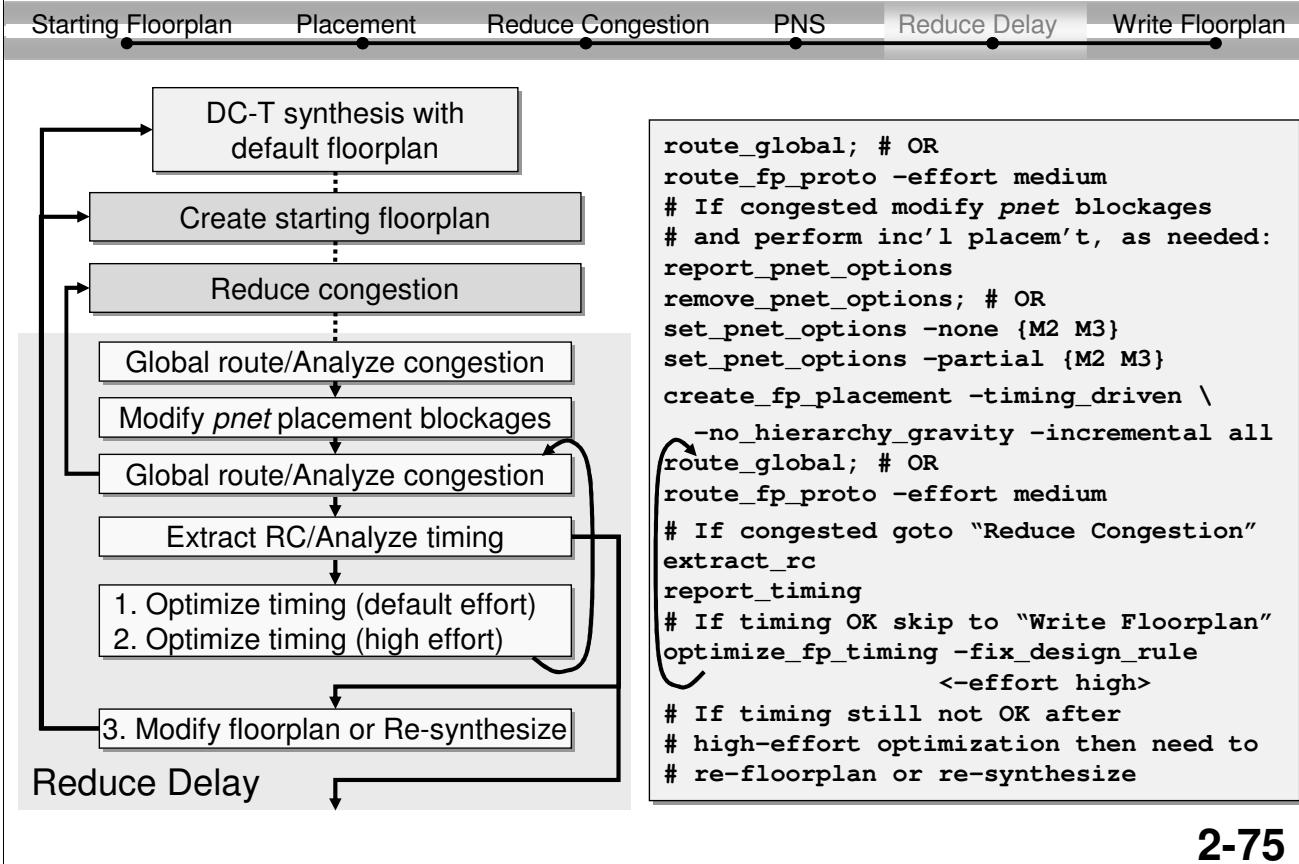
1. Analyze and modify the floorplan, if applicable: Timing can sometimes be improved by, for example, modifying macro cell placement, or I/O pad placement.

2. Analyze the timing reports of the critical paths:

If the violations occur on input or output logic paths, the problem may be that the input/output constraints are too aggressive (to be conservative), and should be relaxed. Also, critical paths may be “false” violations which can be eliminated by including appropriate `set_false_path` or `set_multicycle_path` constraints. If the violating paths contain combinational logic that cross hierarchical boundaries, the design’s logical hierarchy is poorly partitioned, which can limit logic optimization at the sub-block boundaries. The design may be able to be repartitioned, if this does not significantly impact verification (for example: Simulation testbenches may be “probing” the value of specific sub-block pins. If re-partitioned, these probe points may disappear, which breaks the verification testbench).

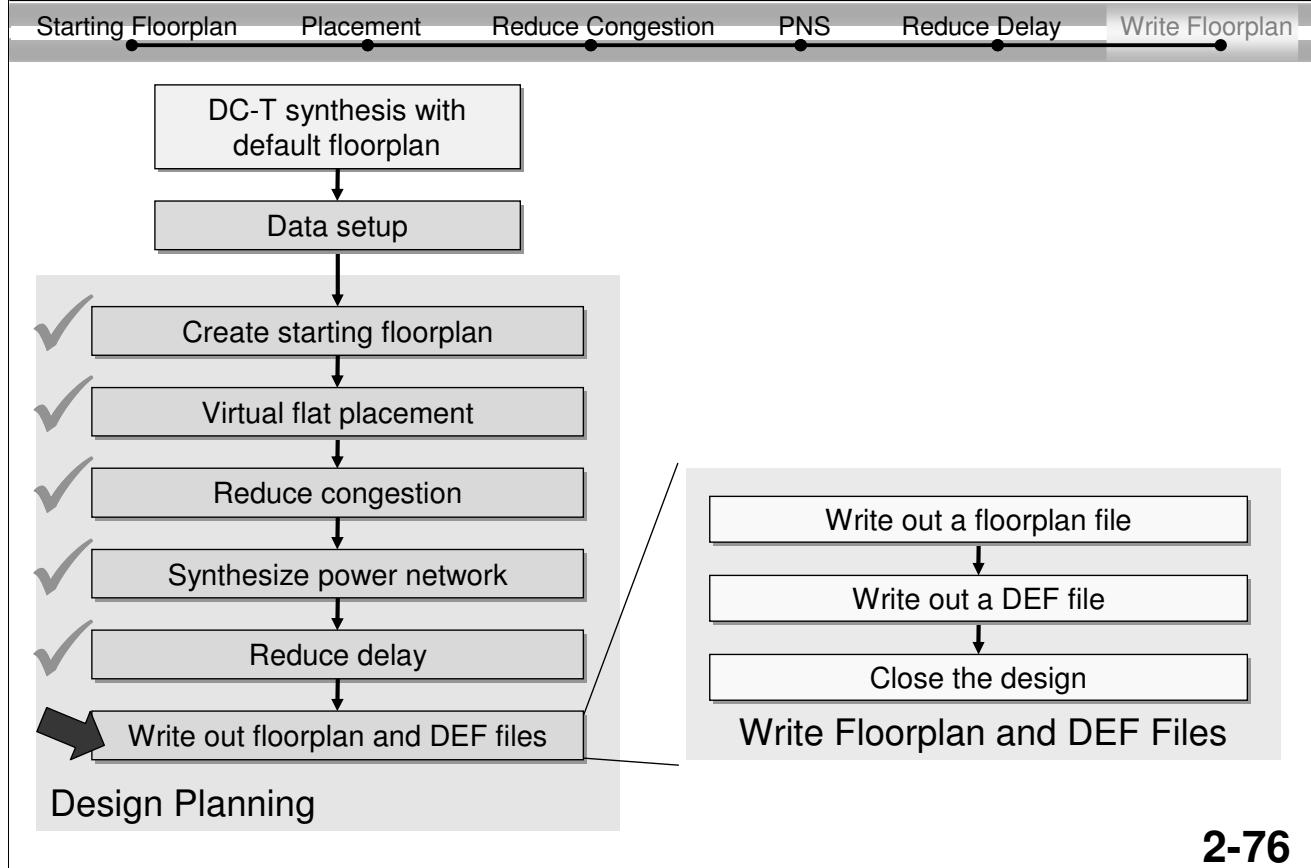
If any constraint- or partitioning-related issues are observed, have the design re-synthesized with modified constraints and/or partitioning. Repeat the design planning steps from the beginning.

Summary: Reduce Delay



2-75

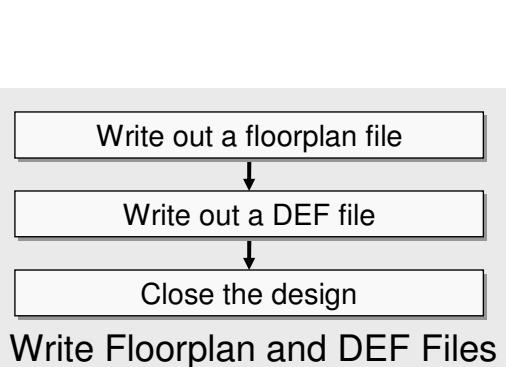
Write Out the Floorplan and DEF Files



2-76

Write Out Floorplan and *DEF* Files

Write Floorplan



■ Write out a *DEF* file:

- Will be used by *Design Compiler Topographical* for re-synthesis

■ Write out a *floorplan* file:

- Will be loaded into IC Compiler after re-synthesis and data setup

■ Close the design cell without saving it

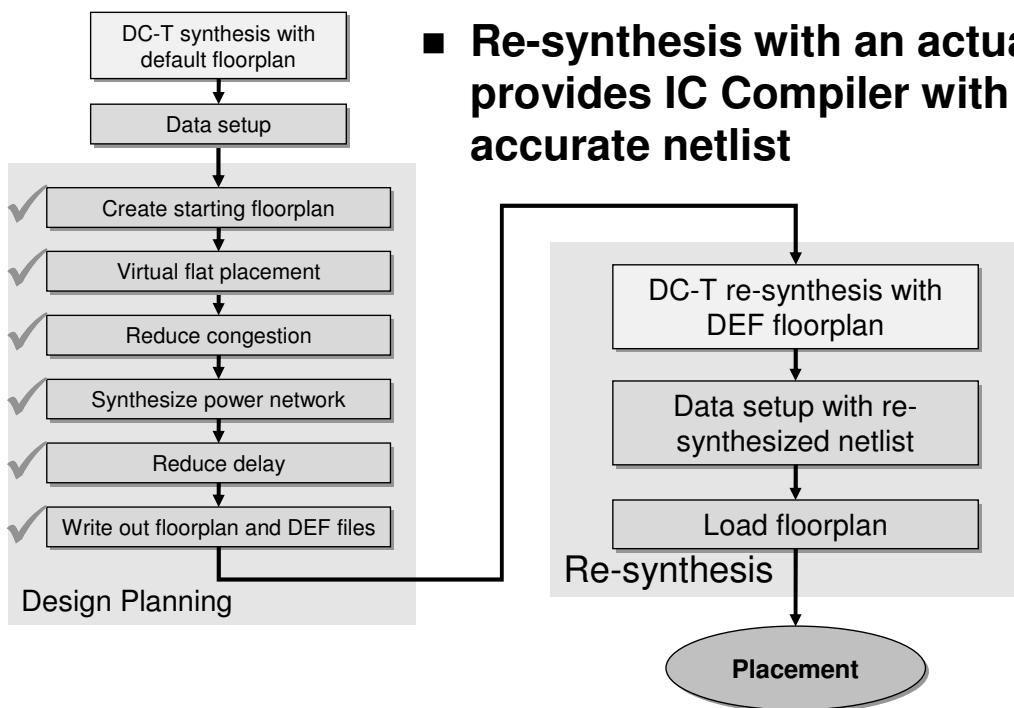
- Only the above files are needed

```
write_floorplan -row -track -create_terminal -preroute \
                 -placement {io hard_macro} DESIGN.fp
write_def -version 5.6 -macro -fixed -blockage \
           -routed_nets -specialnets -output DESIGN.def
close_mw_cel
```

2-77

Re-Synthesize Before Placement

- Initial synthesis used a default DC-T floorplan
- Re-synthesis with an actual floorplan provides IC Compiler with a more accurate netlist

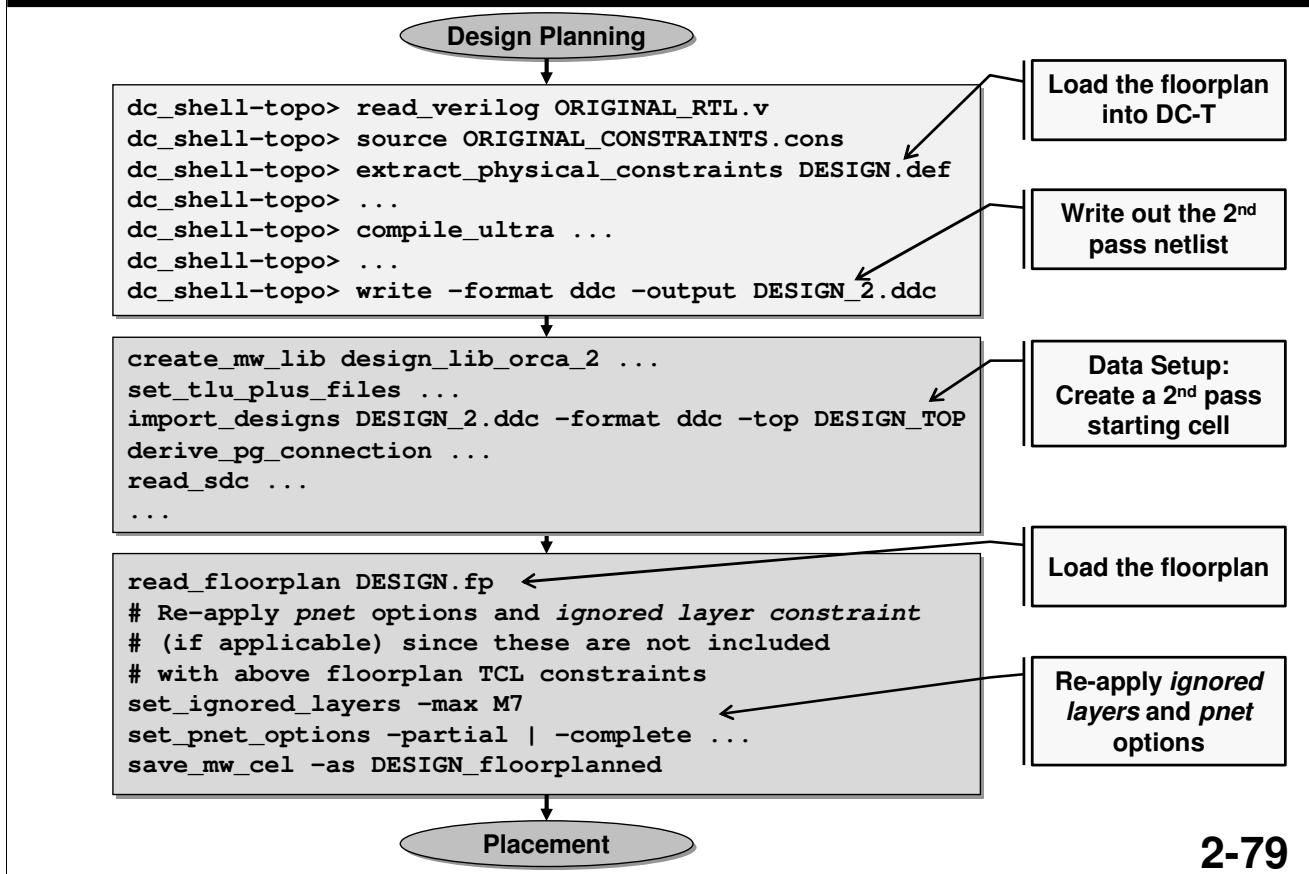


2-78

The above flow is recommended for designs with challenging “quality of results” (QoR) requirements. By re-synthesizing the design with DC-Topographical using an accurate floorplan you are providing IC Compiler with a potentially better starting netlist, which may give better results. If QoR is not critical, the re-synthesis step can be skipped and the resulting cell after design planning, with its netlist from the first synthesis execution, is taken directly to placement.

The above flow assumes that the RTL code is fairly complete when performing the initial synthesis. If the RTL contains a fair number of “black boxes” (i.e. undefined sections) you may need to perform additional iterations of design planning and synthesis, as the RTL code is solidified.

Re-Synthesize Before Placement



2-79

The set_ignored_layers and set_pnet_options attributes are not saved with the floorplan file, so they need to be re-applied.

Test For Understanding (1 of 2)



- 1. Circle the correct statement(s) about *PNS*:**
 - a. Allows what-if analysis based on different user constraints
 - b. Creates DRC/ERC clean routes with `synthesize_fp_rail`
 - c. Can automatically calculate the width and number of straps based on power and IR-drop constraints
 - d. Allows what-if analysis with “virtual” IO signal pads
- 2. A “partial” *pnet* blockage allows cells to be placed only under the specified metal layers (DRC permitting)
– all other layers are completely blocked.**

True or False?

2-80

1. **A and C.** Routes are created with `connect_fp_rail`. What-if analysis can only be done with virtual power/ground pads, not signal IO pads.
2. **False.** A partial *pnet* blockage will allow cells to be placed under the specified metal layers, DRC permitting. The unspecified layers are not blocked at all.

Test For Understanding (2 of 2)



3. Circle the correct statement(s) regarding `optimize_fp_timing -fix_design_rule`:
 - a. Optimizes placement improve DRC violations
 - b. Optimizes the logic to improve timing violations
 - c. Optimizes placement to reduce congestion
 - d. Optimizes placement to improve timing violations
4. Why is it recommended to execute `route_global` and `extract_rc` prior to timing analysis?
5. What are the **DEF** and the **TCL** floorplan constraints files used for?
6. What is the point of doing DC-T re-synthesis after the floorplan is completely defined?

2-81

3. Placement is legalized, but not optimized.
4. Performing an actual global route (without congestion-map-only) along with `extract_rc` allows for more accurate RC parasitic extraction and timing analysis.
5. The DEF file is read in by Design Compiler for re-synthesis. The TCL floorplan constraints file is read in by IC Compiler after re-synthesis to create a design that is ready for placement.
6. The netlist generated after re-synthesis is based on an actual floorplan, rather than DC-T's. "default" floorplan. If the design contains many macros, and/or has a non-square core shape, and/or has unusually high or low utilization, and/or has many placement blockages, the re-synthesized netlist will be better optimized, which can reduce the timing violations as well as iteration cycles during physical design.

Summary



You should now be able to:

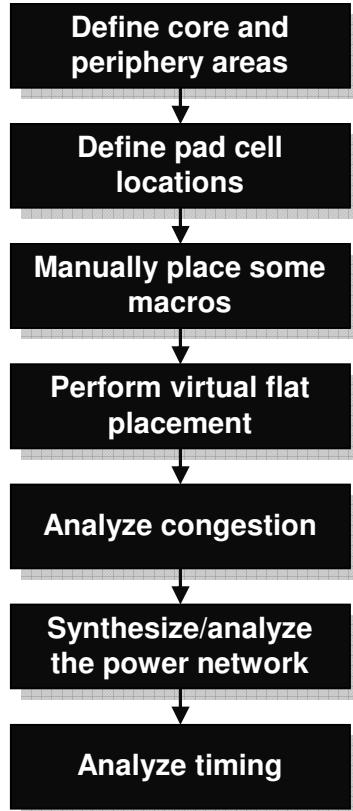
- Use IC Compiler to create a non-hierarchical chip-level floorplan
- Create a floorplan that is likely to be routable and achieve timing closure

2-82

Lab 2: Design Planning



100 minutes

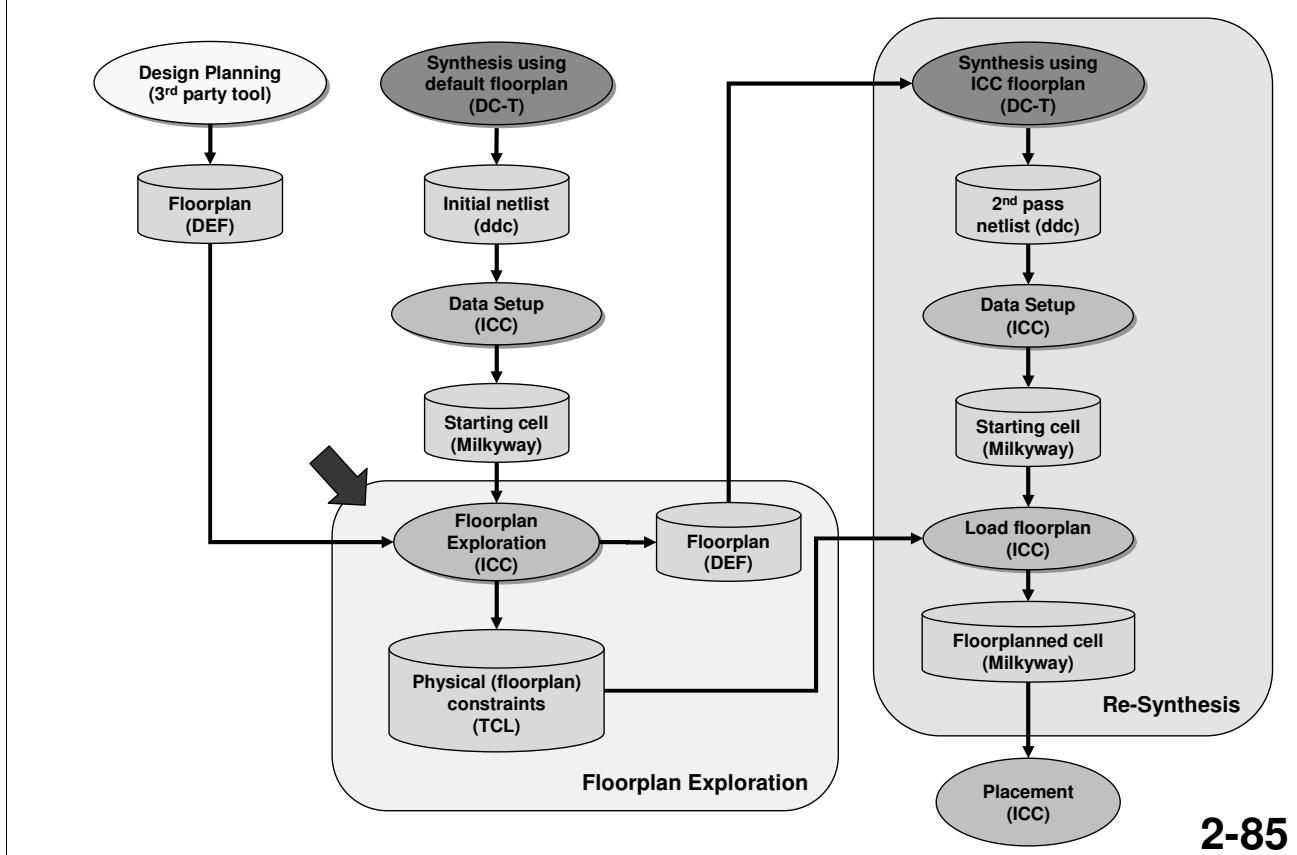


2-83

Appendix A

***Floorplan exploration and re-synthesis for
floorplans created by 3rd party tools***

Floorplan Exploration and Re-Synthesis Flow



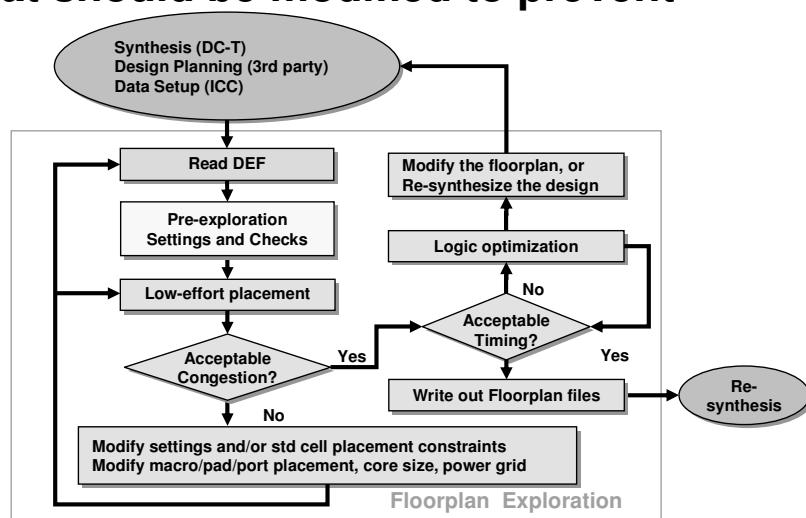
2-85

Floorplan Exploration Objectives

Quickly determine, before placement, if the final layout may run into routability or timing issues

If so, determine what should be modified to prevent these issues:

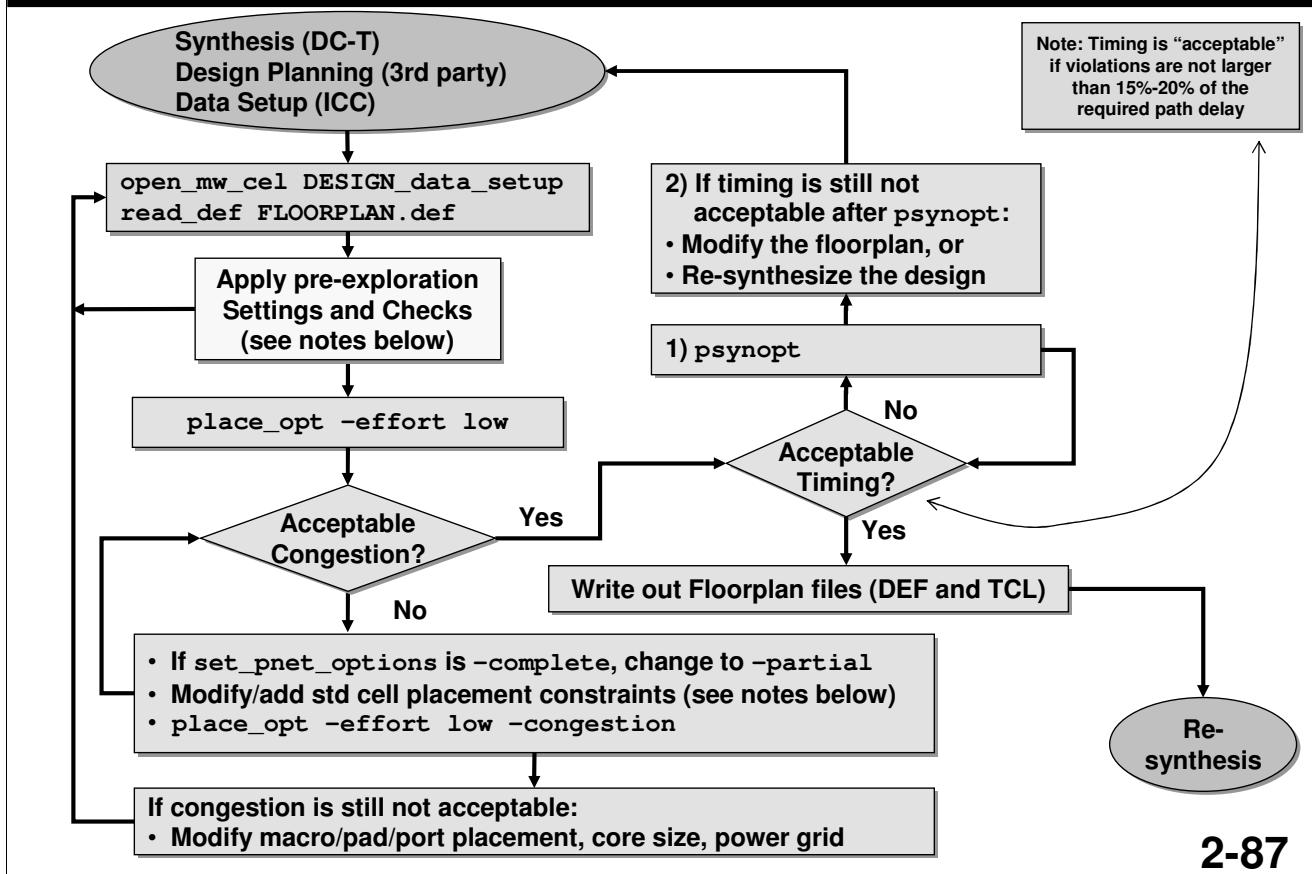
- Standard cell placement constraints (e.g. blockages)
- Floorplan
- Synthesized netlist



Floorplan exploration is recommended for designs that were floor-planned using a 3rd party tool

2-86

Overview: Floorplan Exploration



2-87

“Pre-exploration settings and checks” includes the following commands as applicable (shown with specific arguments only as an example):

```

link
set_pnet_options -partial | -complete {metal2 metal3}
set_ignored_layers -max_routing_layer m7
check_physical_design -for_placement
check_physical_constraints
  
```

“Modify/add physical constraints” includes the following commands as applicable:

```

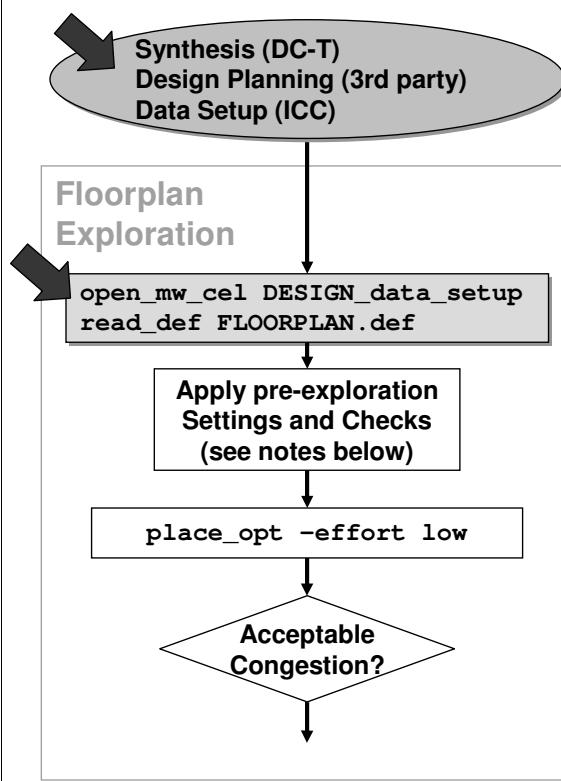
set_physopt_hard_keepout_distance 10
set_placer_soft_keepout_channel_width 25
set_keepout_margin -type hard -outer {10 0 10 0} RAM5
set_congestion_options -max_util 0.4 -coordinate {x1 y1 x2 y2}
create_placement_blockage -name LL_CORNER -type hard -no_snap \
    -coordinate {{345.540 355.790} {392.280 400.070}}
  
```

“Write out Floorplan files” includes the following commands:

```

write_floorplan -row -track -create_terminal \
    -preroute -placement {io hard_macro} DESIGN.fp
write_def -version 5.6 -macro -fixed -blockage \
    -routed_nets -specialnets -output DESIGN.def
close_mw_cel
  
```

Load the Floorplan Definition (DEF) File



- Perform *synthesis* with DC-Topographical
- Perform *Data Setup* in IC Compiler with the netlist from synthesis
- Perform *design planning* with a 3rd party tool
 - Write out the floorplan definition (*DEF*) file
- Open the *starting cell* and load the *DEF* file
 - The graphical view is updated to reflect the floorplan

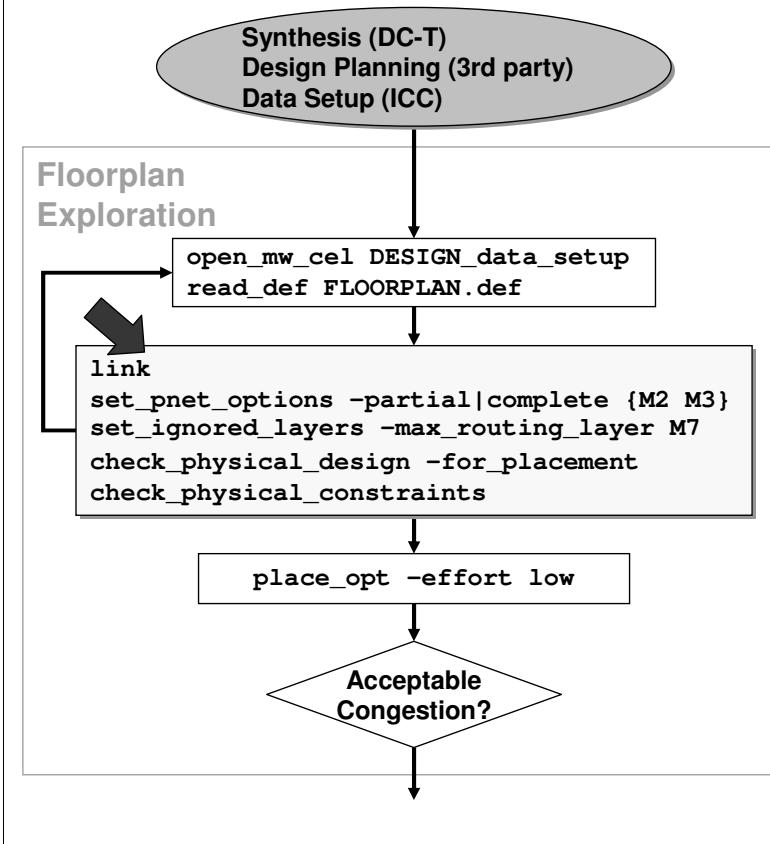
2-88

“DEF” stands for “Design Exchange Format”, which is an industry standard format used to describe the floorplan of a physical design (or layout).

Starting with software version 2008.09 the `read_def` command will read in all DEF data, including “physical-only” data (cells, nets or ports that exist in the DEF file but not in the netlist, for example corner or filler cells, power pre-routes).

Prior to 2008.09 the `-allow_physical` option is required to read in physical-only data as well.

Pre-exploration Settings and Checks



- **Link the design**
 - Verifies that any new cells included in *DEF* can be resolved
- **Define the *maximum routing layer* and apply *pnet* options, as needed**
 - See 2-16 and 2-65
- **Perform two checks before placement**
 - Modify the floorplan to correct problems (see next slide)

2-89

Check Placement Readiness

`check_physical_design -stage pre_place_opt` checks

the readiness for placement of:

- Floorplan
- Netlist
- Design constraints

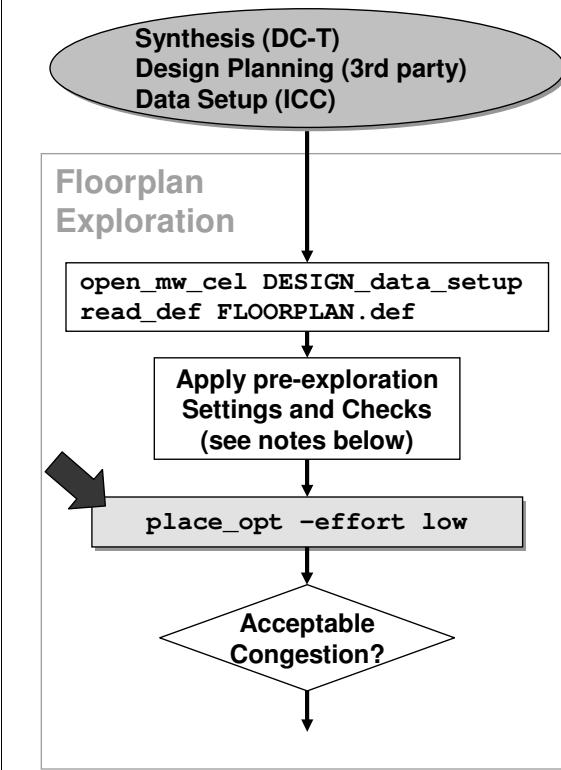
`check_physical_constraints` reports:

- Cells placed in “hard placement blockage” areas
- Metal layer inconsistencies against the library
- R/Cs for routing layers
- Narrow placement regions (“chimneys”)
- Legal sites for cell placement
- Large RC variations between metal layers

Modify the floorplan, constraints or libraries as needed

2-90

Perform Low-effort Placement Optimization

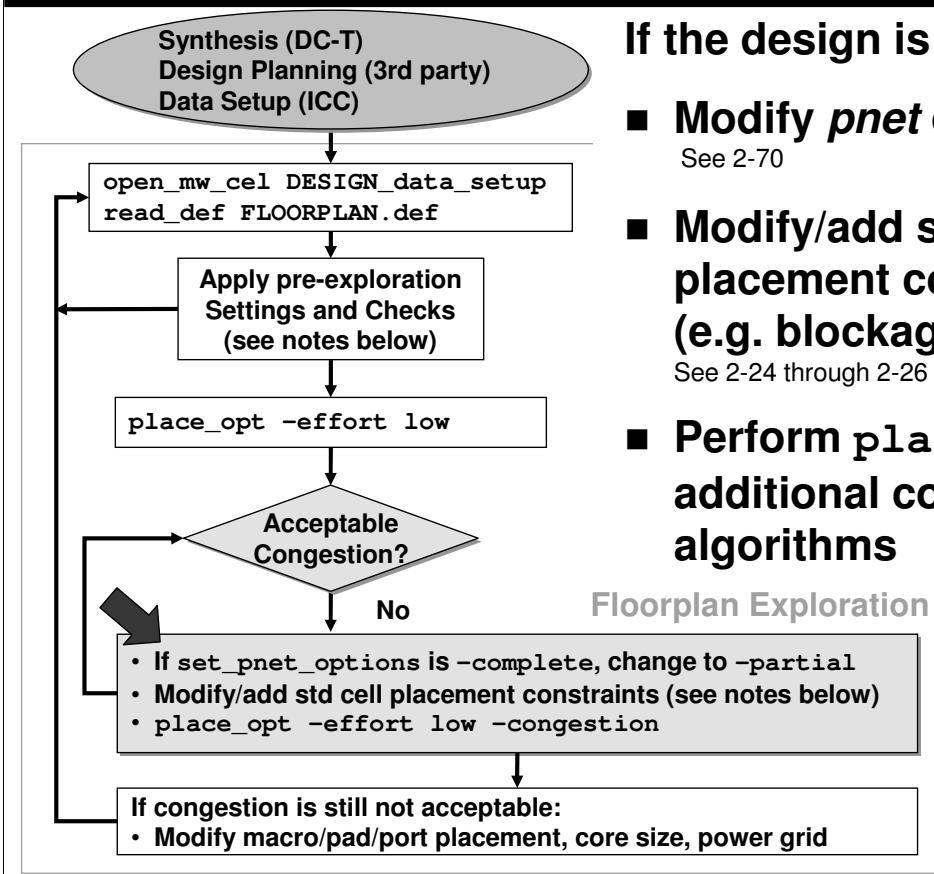


```
place_opt -effort low
```

- **Performs concurrent timing- and congestion-driven placement and logic optimization**
- **The *low effort* minimizes logic optimization runtime**
 - Ideal for “exploration” purposes

2-91

Check and Fix Congestion



If the design is congested:

- **Modify *pnet* options**
See 2-70
- **Modify/add standard cell placement constraints (e.g. blockage, utilization %)**
See 2-24 through 2-26 and 2-41 though 2-44
- **Perform *place_opt* with additional congestion-driven algorithms**

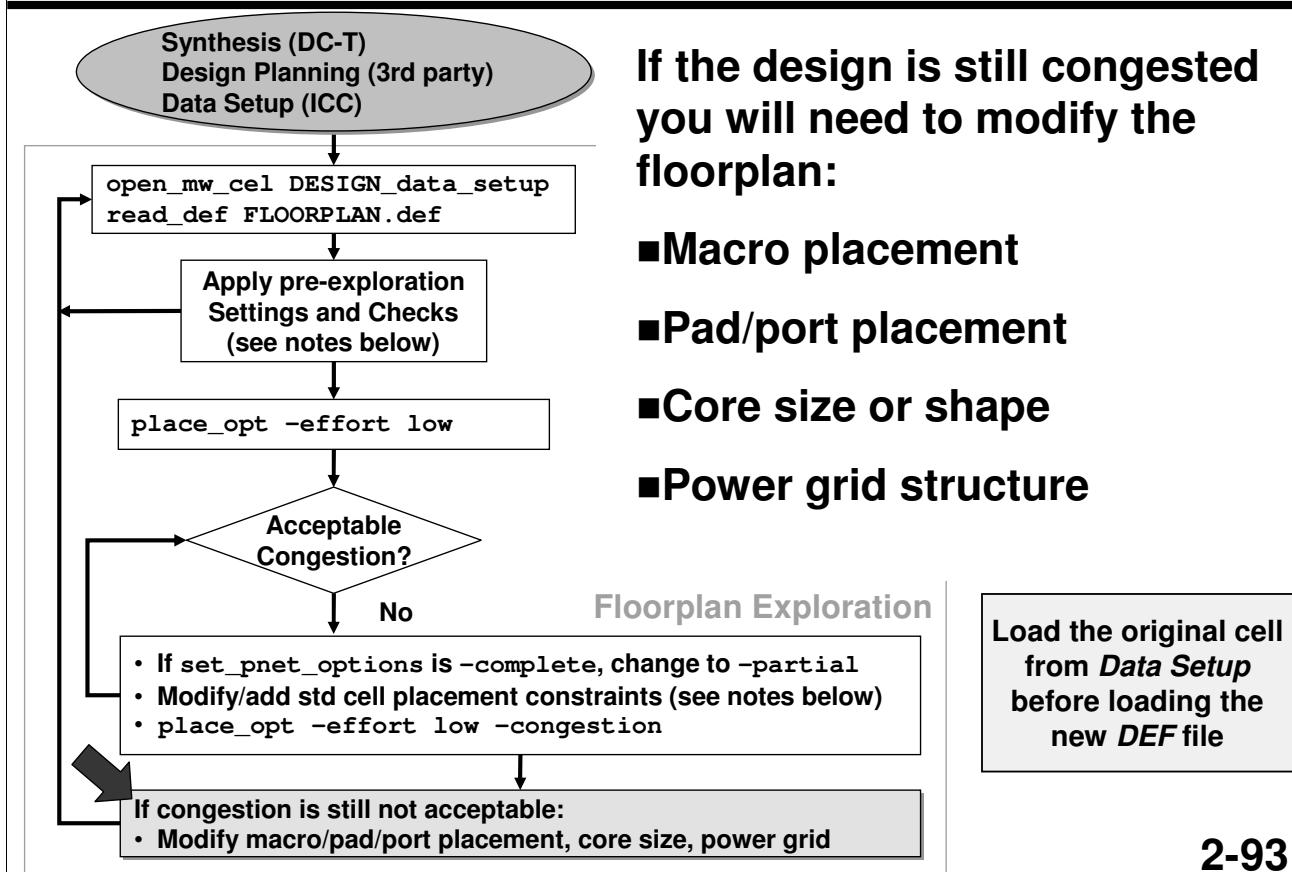
2-92

“Modify/add physical constraints” includes the following commands as applicable (shown with specific arguments only as an example – see Unit 2):

```

set physopt_hard_keepout_distance 10
set placer_soft_keepout_channel_width 25
set_keepout_margin -type hard -outer {10 0 10 0} RAM5
set_congestion_options -max_util 0.4 -coordinate {x1 y1 x2 y2}
create_placement_blockage -name LL_CORNER -type hard -no_snap \
    -coordinate {{345.540 355.790} {392.280 400.070}}
  
```

Modify the Floorplan



2-93

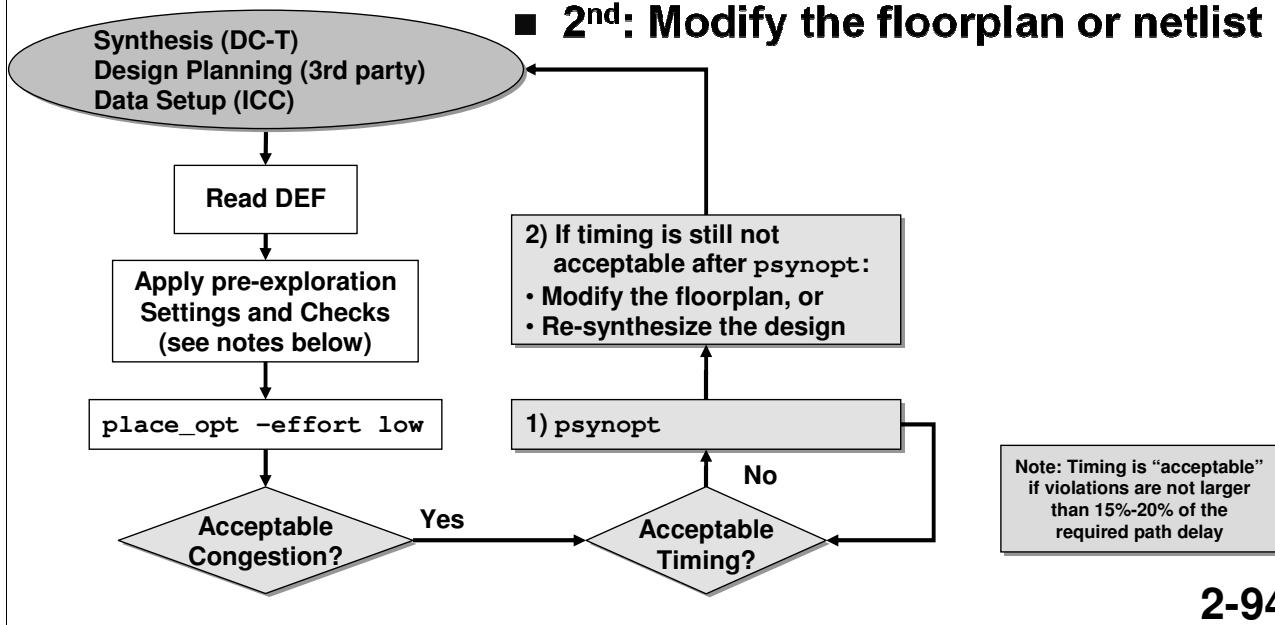
Since DEF information can be “cumulatively” applied to a design, you should discard the floorplanned design and reload the original cell from Data Setup before loading the new DEF information.

Analyze and Fix Timing

Once congestion is OK, analyze timing and fix if the worst violation is larger than 15-20% of the required path delay:

■ 1st: Incremental logic optimization

■ 2nd: Modify the floorplan or netlist



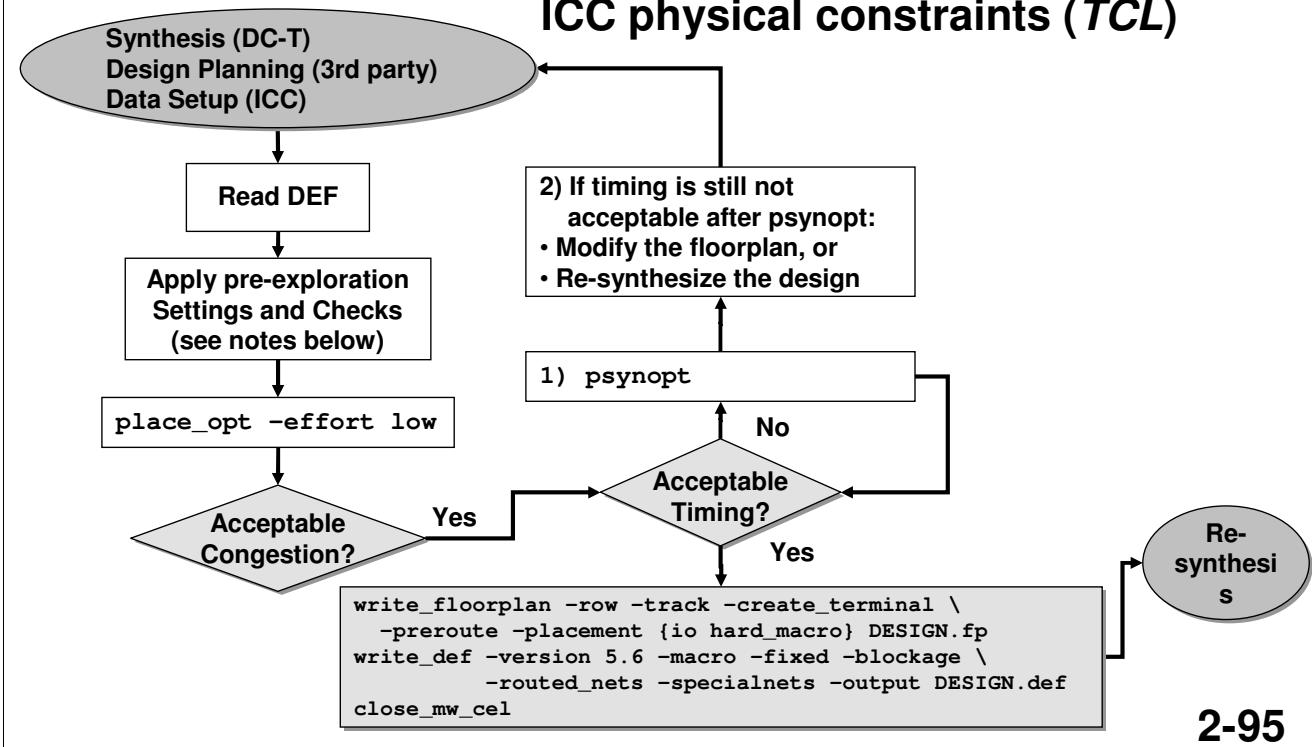
2-94

By default psynopt performs incremental logic optimization to improve timing.

If the design is re-synthesized, you will need to re-do Data Setup to create a new starting cell, before loading the floorplan.

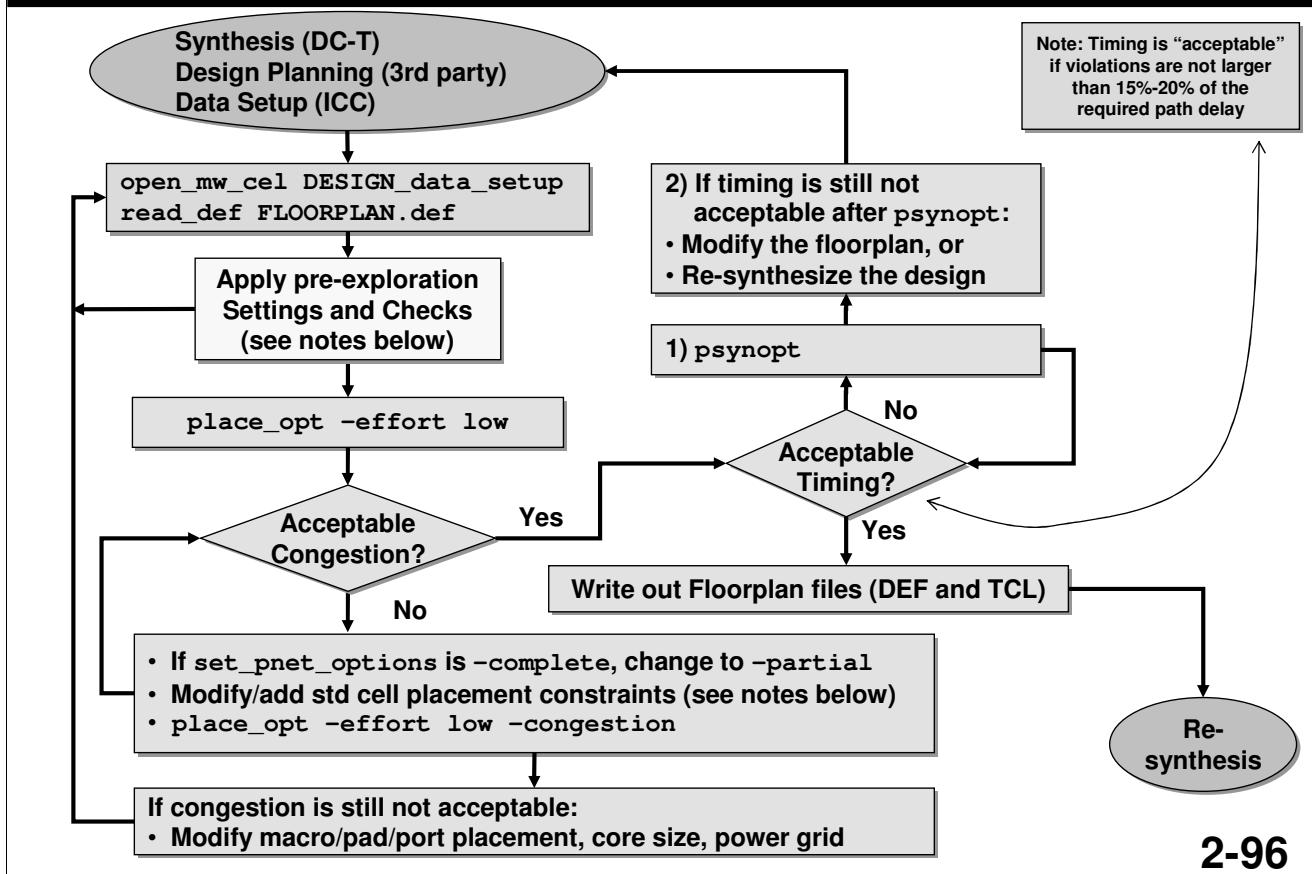
Write Out the Floorplan Files

Once timing is OK, the floorplan is ready for re-synthesis and placement → Write out the floorplan files as *DEF* and *ICC physical constraints (TCL)*



2-95

Summary: Floorplan Exploration



2-96

"Pre-exploration settings and checks" includes the following commands as applicable (shown with specific arguments only as an example):

```

link
set_pnet_options -partial | -complete {metal2 metal3}
set_ignored_layers -max_routing_layer m7
check_physical_design -for_placement
check_physical_constraints

```

"Modify/add physical constraints" includes the following commands as applicable:

```

set_physopt_hard_keepout_distance 10
set_placer_soft_keepout_channel_width 25
set_keepout_margin -type hard -outer {10 0 10 0} RAM5
set_congestion_options -max_util 0.4 -coordinate {x1 y1 x2 y2}
create_placement_blockage -name LL_CORNER -type hard -no_snap \
    -coordinate {{345.540 355.790} {392.280 400.070}}

```

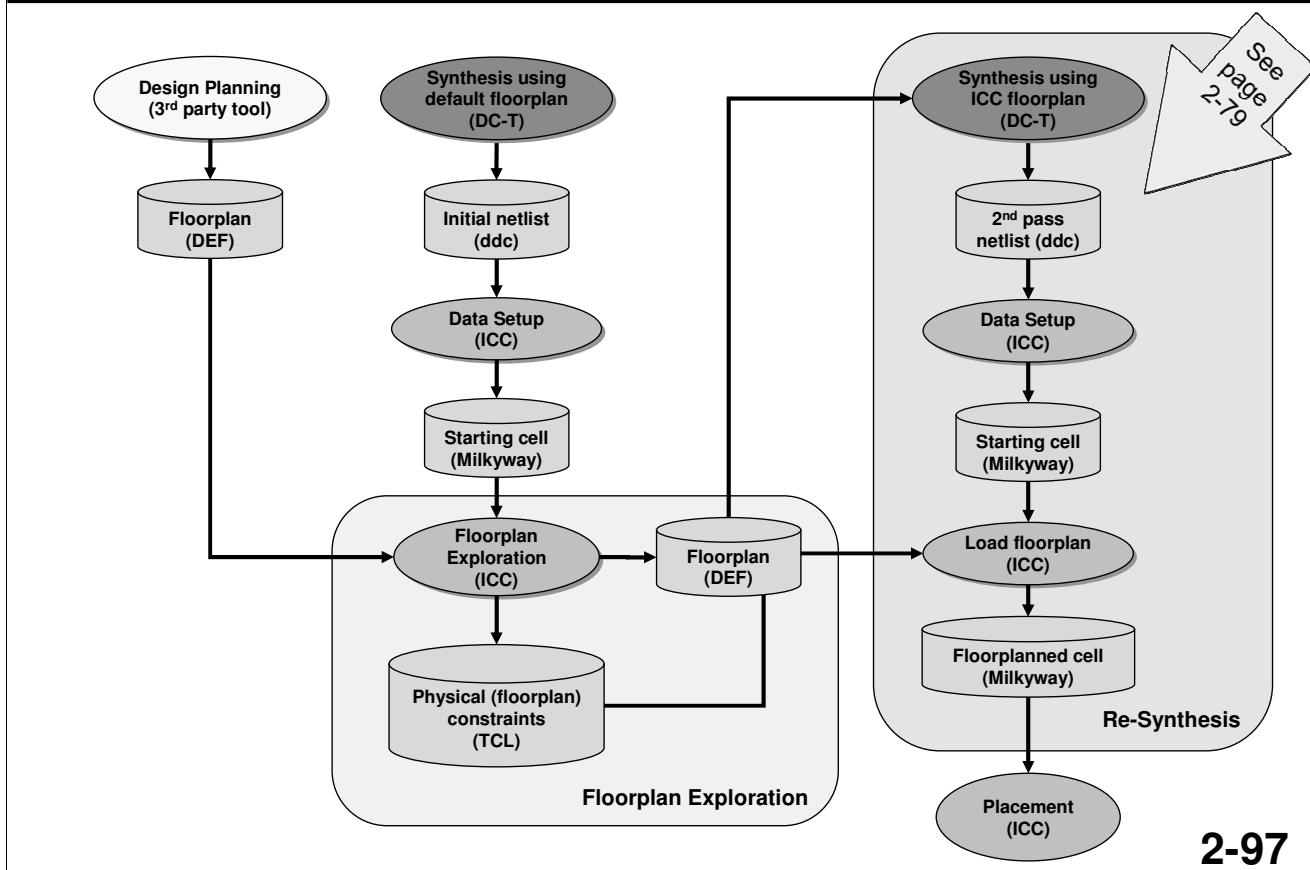
"Write out Floorplan files" includes the following commands:

```

write_floorplan -row -track -create_terminal \
    -preroute -placement {io hard_macro} DESIGN.fp
write_def -version 5.6 -macro -fixed -blockage \
    -routed_nets -specialnets -output DESIGN.def
close_mw_cel

```

Summary: 3rd Party Design Planning Flow



This page was intentionally left blank

Agenda

**DAY
2**

3 Placement



4 Clock Tree Synthesis



Unit Objectives

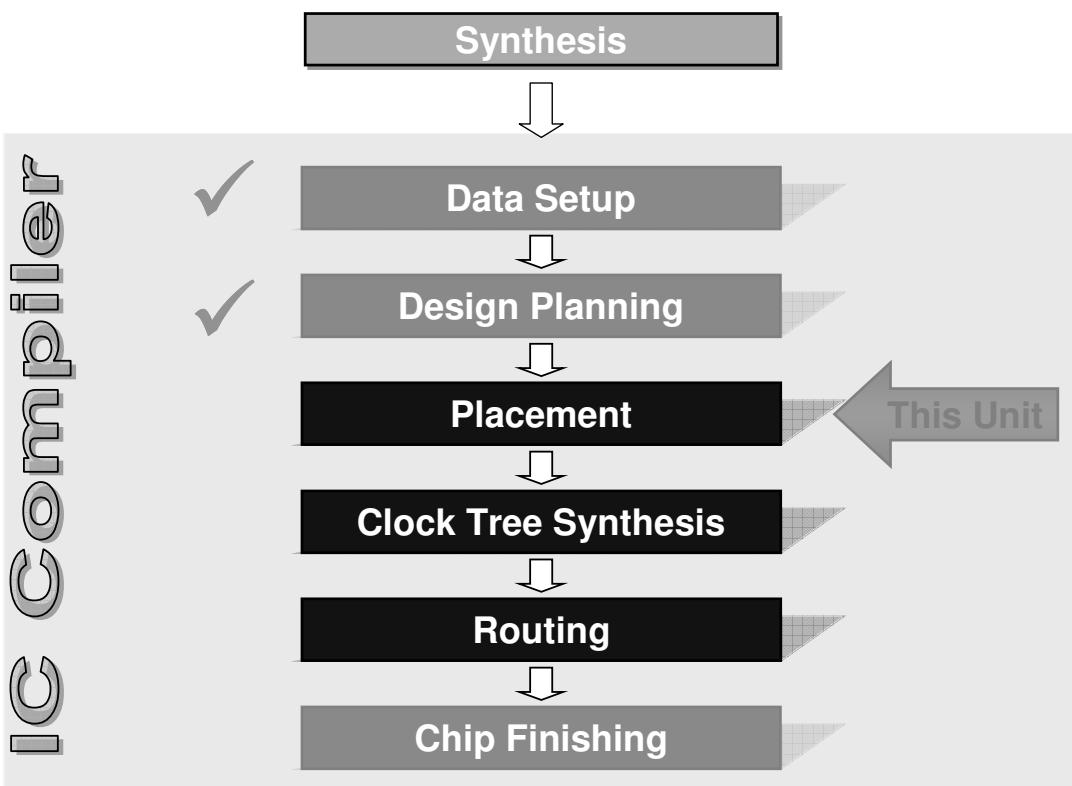


After completing this unit, you should be able to:

- **Apply placement, DFT and power optimization settings before placement**
- **Perform placement and optimization**
- **Analyze congestion maps and reports**
- **Perform incremental congestion and timing optimization**
- **Perform additional placement techniques which allow more user-control**

3-2

General IC Compiler Flow



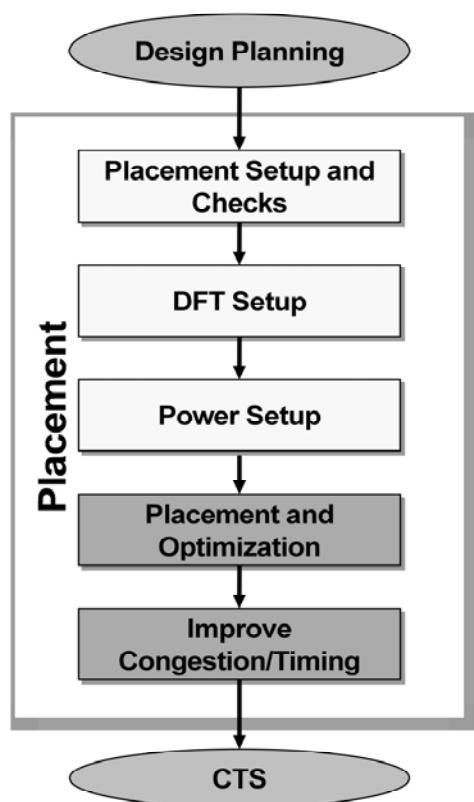
3-3

Design Status Prior to Placement

- ***Synthesis is completed***
 - Second-pass netlist is generated
- ***Data Setup is completed***
 - “Starting cell” is generated based on 2nd pass netlist
- ***Design Planning is completed***
 - “Floorplan Exploration” done if using 3rd party floorplanning tool (Refer to “Design Planning” Unit 2)
- **“Floorplanned cell” is generated – ready for placement**
 - Based on 2nd pass synthesis netlist and verified floorplan

3-4

IC Compiler Placement Flow

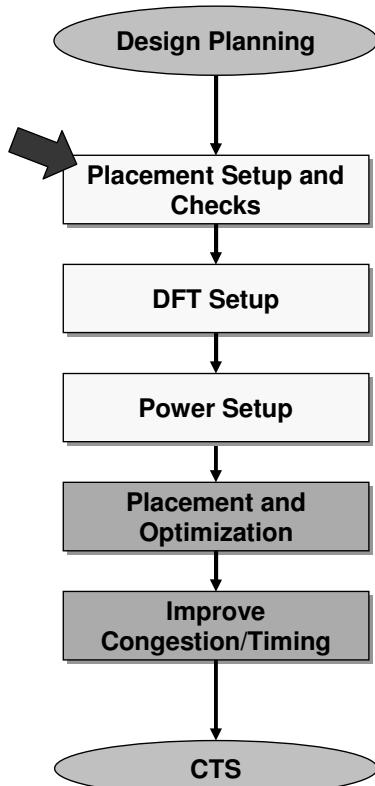


The “placement phase” involves several key steps:

- **Setup steps to control placement, DFT and power**
- **Placement and optimization**
- **Incremental refinement to improve congestion and/or setup timing**

Note: The flow diagrams included in this unit represent an **example** flow, not the recommended flow

Placement Setup and Checks



3-6

“Fix” all Macro Cell Placements

- In most situations macro cell placement is determined during design planning and their placement is “fixed”
- It is a good practice to fix all macro placements again, just in case....
 - You may have manually moved a macro after design planning and have forgotten to fix its placement

```
open_mw_cel DESIGN_floorplanned  
link  
set_dont_touch_placement [all_macro_cells]
```

3-7

EM = Electro-migration

Verify *pnet* Options and *Ignored Layers*

- Ensure that *ignored routing layers* are defined (if applicable)

`report_ignored_layers`

- Ensure that the appropriate *pnet* options are applied

`report_pnet_options`

- The above settings should have been applied during design planning
 - Must be explicitly re-applied after loading the floorplan
- If missing or incorrect – re-apply (see notes below)

3-8

```
# Remove and re-apply ignored layers
remove_ignored_layers -all
set_ignored_layers -max_routing_layer M7
```

```
# Remove blockages
remove_pnet_options OR
set_pnet_options -none {M6 M7}
```

```
# Modify blockage options
set_pnet_options -complete {M2 M3}
remove_pnet_options
set_pnet_options -partial {M2 M3}
```

Verify Keepout Variable Settings (if used)

- Any *placement keepout* variable settings required during design planning must be maintained for placement, CTS and routing
- Check the *keepout* variable settings:

```
printvar physopt_hard_keepout_distance  
printvar placer_soft_keepout_channel_width
```

- If not correct:

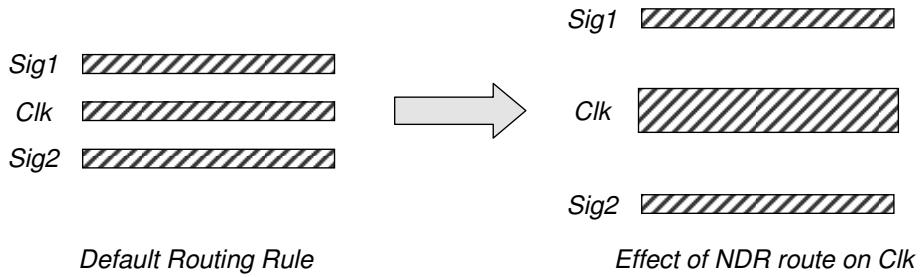
- Apply the correct values for this session
- Add the variables to the `.synopsys_dc.setup` file for subsequent sessions

```
set physopt_hard_keepout_distance <#>  
set placer_soft_keepout_channel_width <#>
```

3-9

Non-Default Clock Routing

- IC Compiler can route clock nets using non-default routing (NDR) rules, e.g. double-spacing, double-width, shielding
- Non-default rules are often used to “harden” the clock, e.g. to make the clock routes less sensitive to *cross-talk* or *electro-migration (EM)* effects
- RC parasitics are different for NDR nets versus default routing nets → Affects timing
- This timing difference can and should be taken into account before actual routing, during placement



3-10

Specify Non-Default Routing Rules

■ Define the NDR rules:

```
define_routing_rule MY_ROUTE_RULES \
    -widths {METAL3 0.4 METAL4 0.6 METAL5 0.6} \
    -spacings {METAL3 0.5 METAL4 0.65 METAL5 0.65}
```

■ Configure the clock tree routing:

```
set CLK_NETS [get_nets -of [all_fanout -flat -clock_tree]]
set_net_routing_rule -rule MY_ROUTE_RULES $CLK_NETS
set_net_routing_layer_constraint -min_layer METAL3 \
    -max_layer METAL5 $CLK_NETS
```

You may also specify the layers to be used for clock tree routing

3-11

You can report on the routing rules that were defined using `report_routing_rule`. If, in the above example, layers are used for which a non-default routing rule was not defined, these layers will use the default routing rules, as defined by the tech-file.

```
define_routing_rule rule_name
    -reference_rule_name rule_name
        | -default_reference_rule
    [-widths {routing_layer_name value} ]
    [-snap_to_track]
    [-spacings {routing_layer_name value} ]
    [-shield_widths {routing_layer_name value} ]
    [-shield_spacings {routing_layer_name value} ]
    [-via_cuts {default_via_name value} ]
    [-taper_level tapering_level]
```

Check Placement Readiness

`check_physical_design -stage pre_place_opt` checks

the readiness for placement of:

- Floorplan
- Netlist
- Design constraints

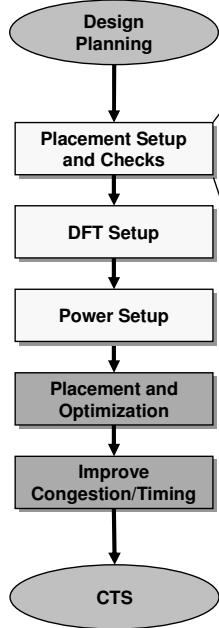
`check_physical_constraints` reports:

- Cells placed in “hard placement blockage” areas
- Metal layer inconsistencies against the library
- R/Cs for routing layers
- Narrow placement regions (“chimneys”)
- Legal sites for cell placement
- Large RC variations between metal layers

Modify the floorplan, constraints or libraries as needed

3-12

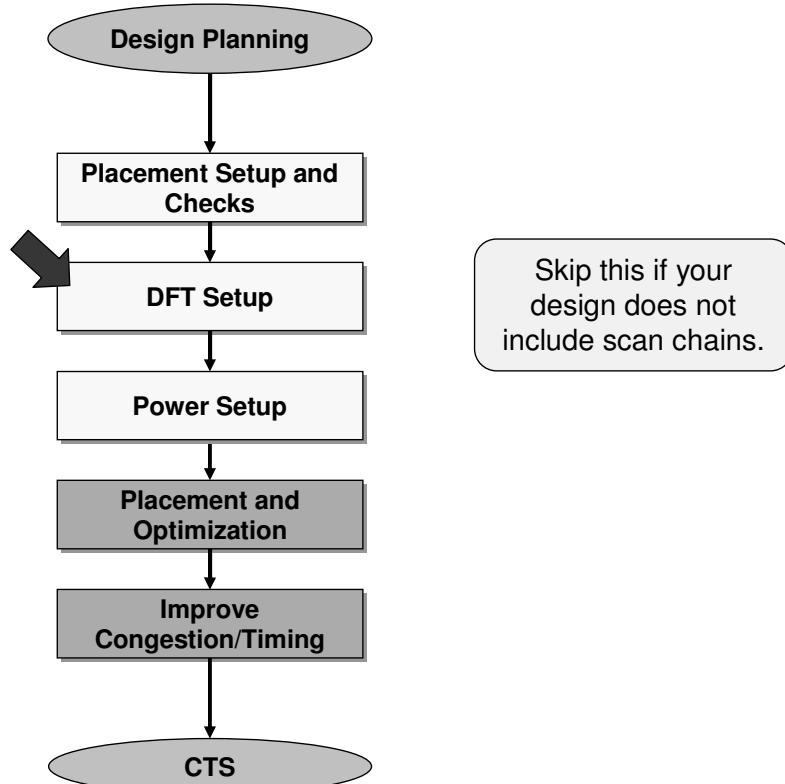
Summary: Placement Setup and Checks



```
open_mw_cel DESIGN_floorplanned  
link  
  
set_dont_touch_placement [all_macro_cells]  
  
report_ignored_layers  
report_pnet_options  
  
printvar physopt_hard_keepout_distance  
printvar placer_soft_keepout_channel_width  
  
# Define non-default routing rules, if applicable  
define_routing_rule MY_ROUTE_RULES \  
    -widths {METAL3 0.4 METAL4 0.6 METAL5 0.6} \  
    -spacings {METAL3 0.5 METAL4 0.65 METAL5 0.65}  
set CLK_NETS \  
    [get_nets -of [all_fanout -flat -clock_tree]]  
set_net_routing_rule -rule MY_ROUTE_RULES $CLK_NETS  
set_net_routing_layer_constraint -min_layer METAL3 \  
    -max_layer METAL5 $CLK_NETS  
  
check_physical_design -stage pre_place_opt  
check_physical_constraints
```

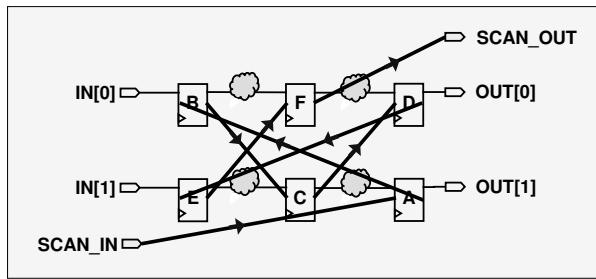
3-13

Design-for-Test (DFT) Setup



3-14

Pre-Existing Scan Chains

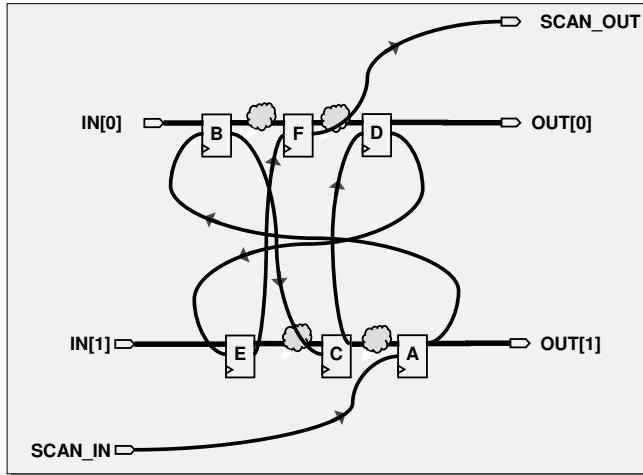


- If your design flow includes “design-for-test”, the netlist will contain “scan chains”:
 - Groups of “scan registers” that are serially connected through SI/SO pins, and inserted during synthesis
- Scan chain paths are active only during “test mode”, not during “functional mode”
- Registers are typically connected in alphanumeric order during synthesis – irrelevant for DFT, but not optimal for routing

3-15

The Issue with Existing Scan Chains

What happens if placement is done with the pre-existing order of the scan chains?



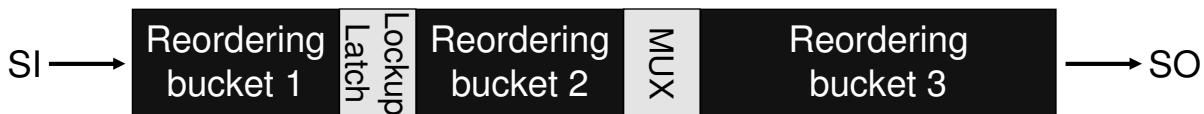
- **Seriously connected scan registers may be placed far apart which may require a lot more routing resources**

3-16

The timing-driven algorithm during placement focuses on eliminating setup timing violations along functional paths. This means that timing-critical register pairs are placed close together, independent of their “scan chain” connection, which may be very far. This generally produces non-optimal routing of the scan chain, which uses up more routing resources and increases congestion. The longer scan routes may also prevent testing from being done at functional speed.

SCANDEF-Based Chain Reordering

- IC Compiler can perform placement-aware reordering of scan cells
- The scan chain information (**SCANDEF**) from synthesis can be transferred to IC Compiler in two ways:
 - By loading the netlist in *ddc* format (imbedded **SCANDEF**)
 - By loading a **SCANDEF** file (generated after scan insertion)
- Reordering occurs within each scan chain
- Lockup latches or multiplexers break up scan chains further into “reordering buckets”
- If present, reordering happens within the buckets



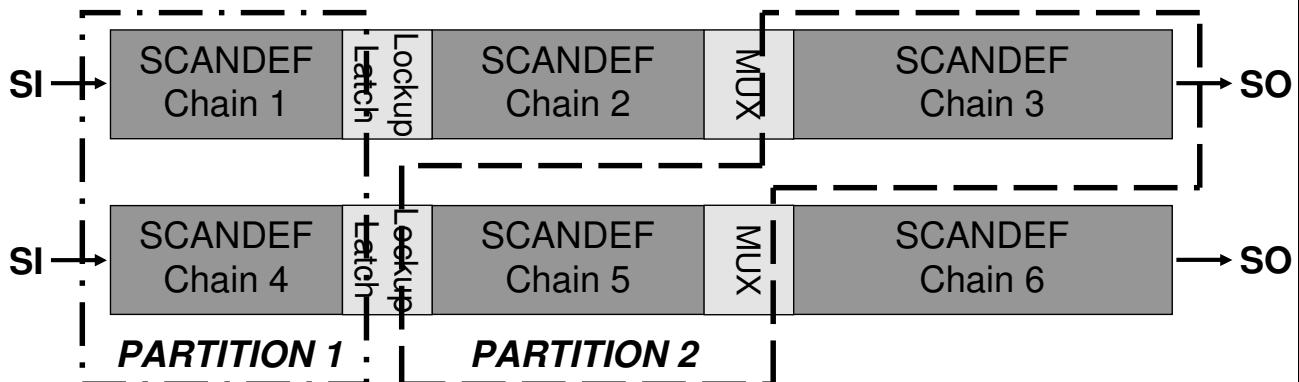
3-17

SCANDEF captures the grouping and ordering of scan chains.

During data setup, if the netlist from synthesis was read in as *ddc*, then the **SCANDEF** information is included and is automatically ported over to IC Compiler. If, on the other hand, the netlist was read in as Verilog or VHDL, then an explicit **SCANDEF** file (ASCII) must be loaded prior to placement. This **SCANDEF** file is typically written out during synthesis, after scan insertion, with the command `write_scan_def`.

Re-ordering Chains within Same “Partition”

- To extend flexibility, SCANDEF also supports reordering within partitions, across multiple buckets
- A PARTITION is a group of “SCANDEF chains” that may exchange flip-flops during reordering



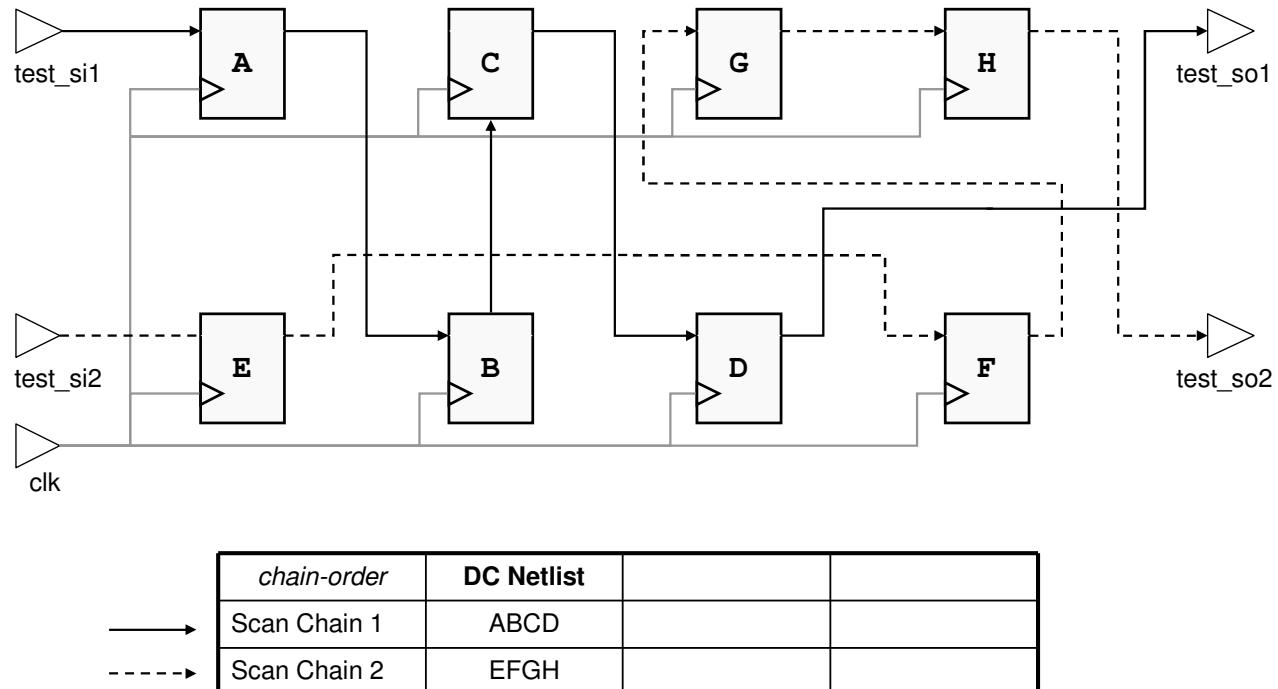
3-18

Example SCANDEF from Synthesis with DFT

```
DESIGN my_design ; ← Design name
SCANCHAINS 2 ; ← Number of chain stubs in
- 1
+ START PIN test_si1
+ FLOATING A ( IN SI ) ( OUT Q )
    B ( IN SI ) ( OUT Q )
    C ( IN SI ) ( OUT Q )
    D ( IN SI ) ( OUT Q ) } "FLOATING" indicates that these
                                flipflops can be reordered
+ PARTITION CLK_45_45 ← PARTITION keyword in SCANDEF.
+ STOP PIN test_so1   Flipflops can be swapped between two
                                partitions with the same name
- 2
+ START PIN test_si2
+ FLOATING E ( IN SI ) ( OUT Q )
    F ( IN SI ) ( OUT Q )
    G ( IN SI ) ( OUT Q )
    H ( IN SI ) ( OUT Q )
+ PARTITION CLK_45_45
+ STOP PIN test_so2
```

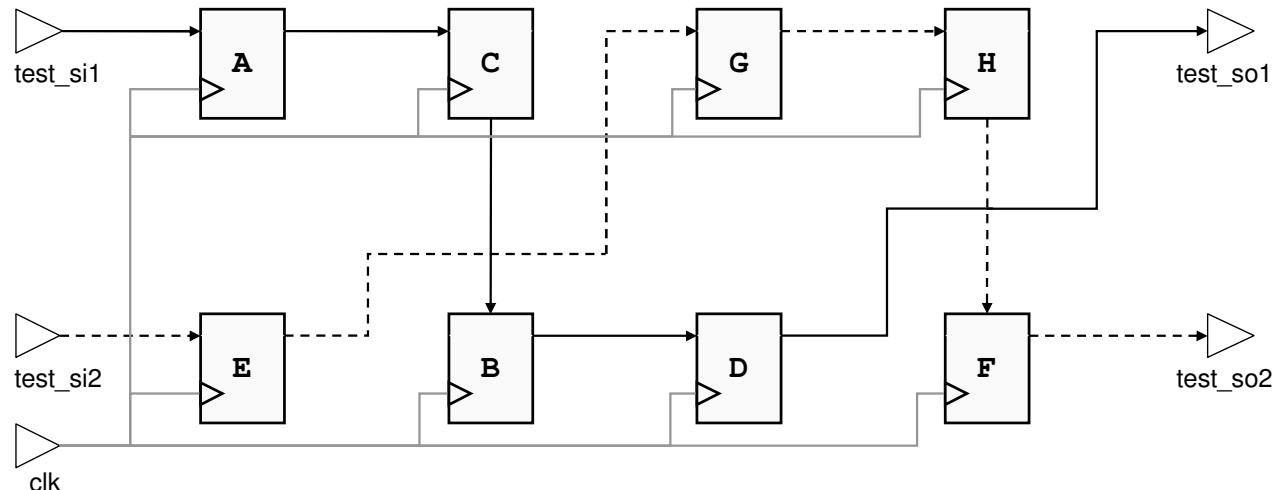
3-19

Example: Placement with Existing Ordering



3-20

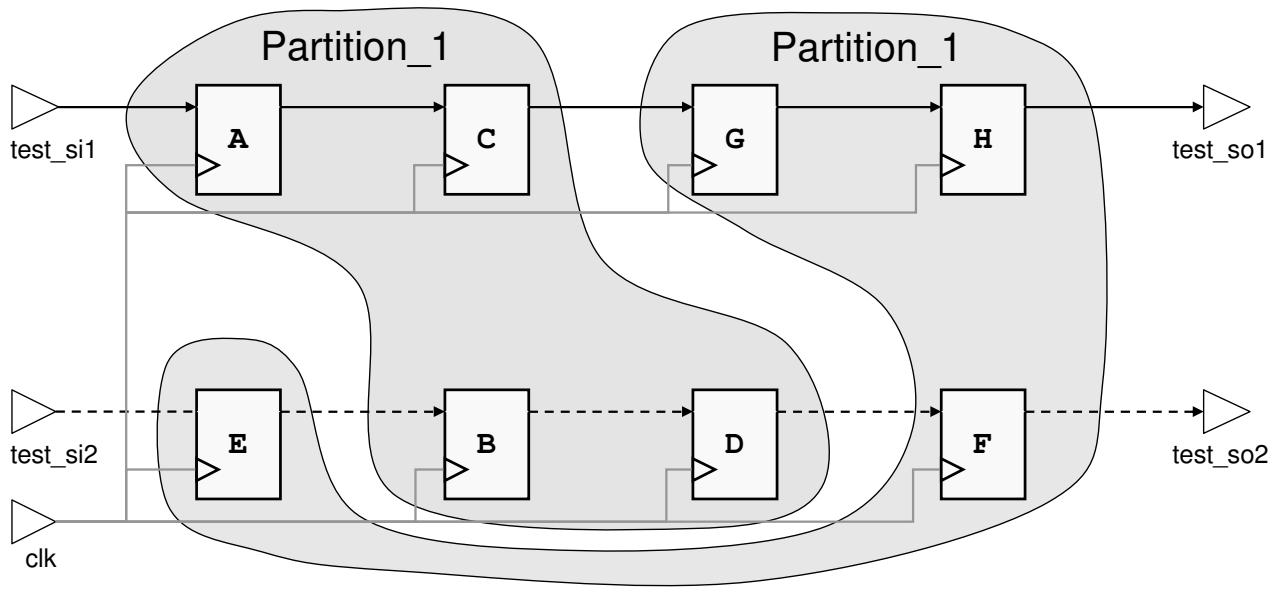
Example: Reordering Within Scan-Chains



<i>chain-order</i>	DC Netlist	SCANDEF w/o PARTITION	
Scan Chain 1	ABCD	ACBD	
Scan Chain 2	EFGH	EGHF	

3-21

Example: Reordering Within Partitions

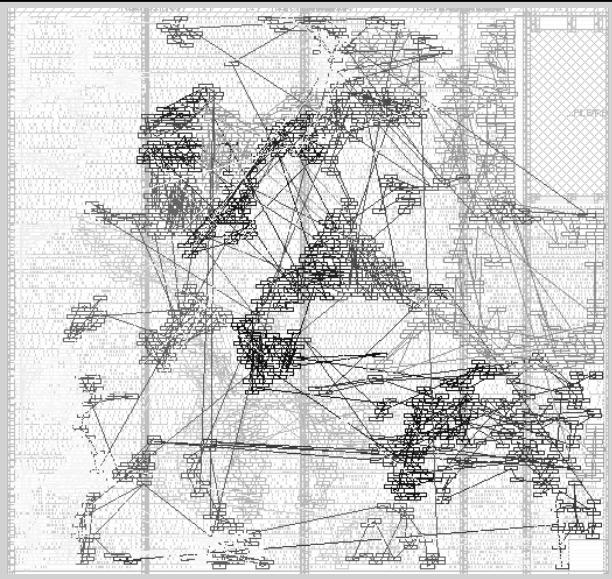


<i>chain-order</i>	DC Netlist	SCANDEF w/o PARTITION	SCANDEF with PARTITION
Scan Chain 1	ABCD	ACBD	ACGH
Scan Chain 2	EFGH	EGHF	EBDF

3-22

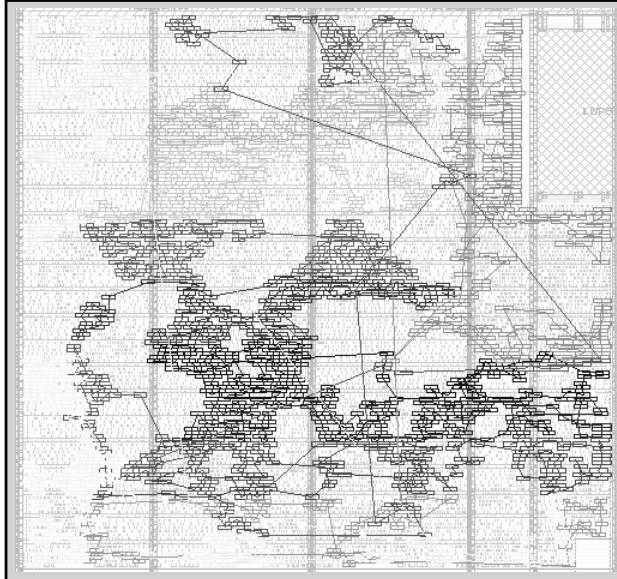
```
check_scan_chain  
report_scan_chain
```

Placement-Based Scan Chain Re-Ordering



Pre-existing Scan Ordering

place_opt



Placement-Based Re-Ordering

```
read_def DESIGN.scandef  
report_scan_chain  
place_opt -optimize_dft
```

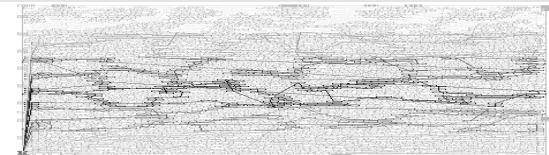
3-23

One of the modes available from the “Visual Mode” is to display scan chains. It allows you to highlight each scan chain in a different color.

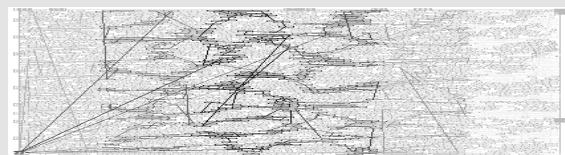


Consider Extreme Block Aspect Ratios

**Default horizontal partitioning
Bad when block width >> height**



**Better to use non-default
vertical partitioning**



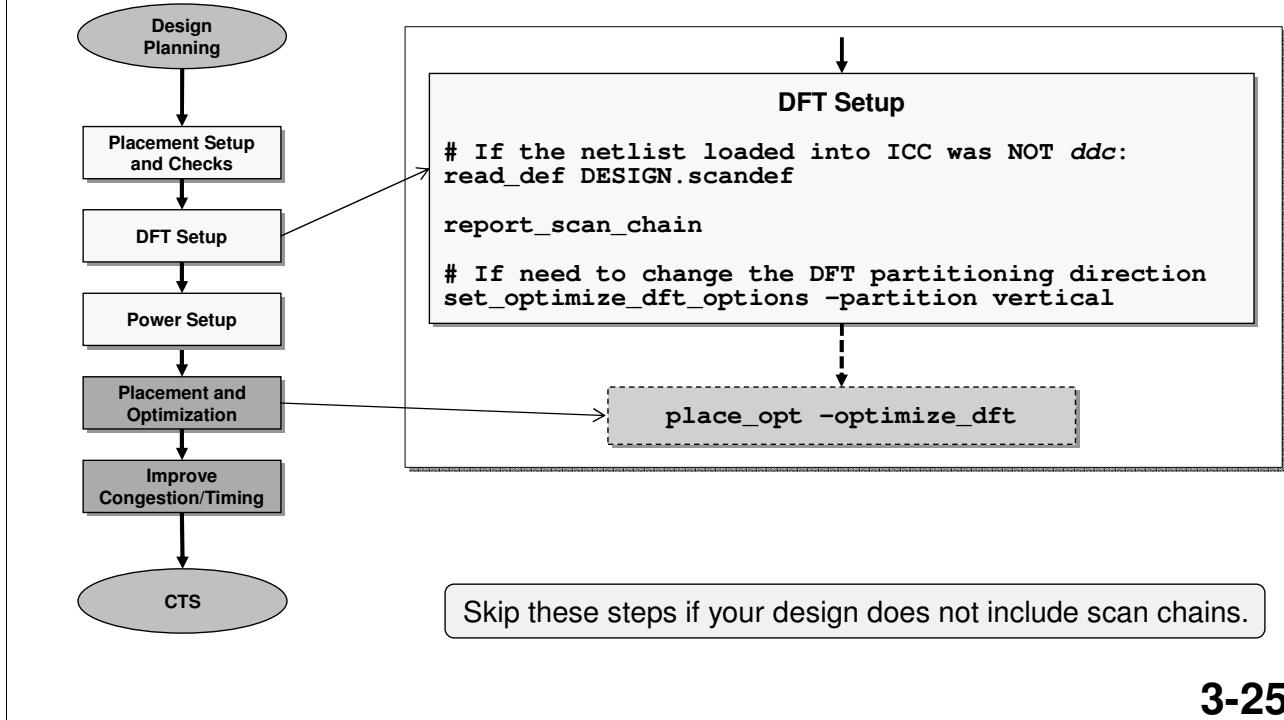
This creates more
congestion in
horizontal direction

```
set_optimize_dft_options -partition vertical
```

3-24

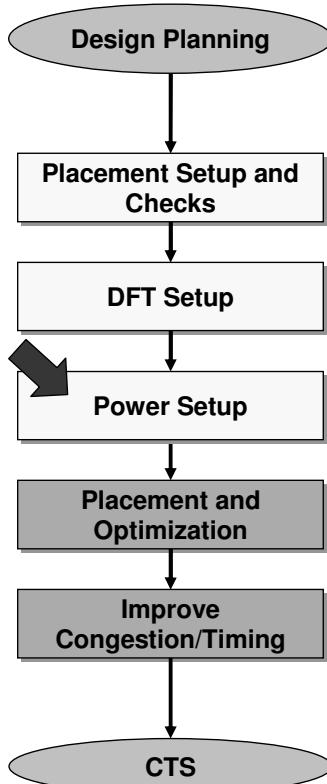
If not specified, the default partitioning direction is “horizontal”.

Summary: DFT Setup



The `place_opt` command with the `-optimize_dft` option is not part of “DFT setup”. It is shown here to summarize all the DFT-related steps in the placement phase.

Power Setup



Skip this if power optimization is not a priority

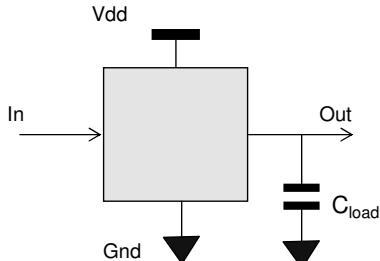
Power optimization will generally:

- Not impact critical path delays or congestion
- Increase run time

3-26

Where Does Power Dissipation Occur?

$$\text{Total Power} = \text{Static Power} + \text{Dynamic Power}$$

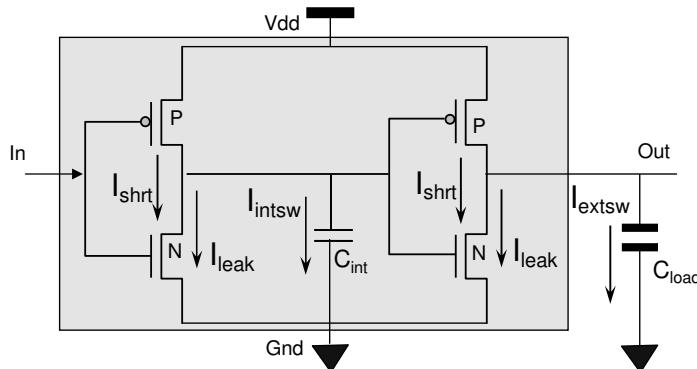


■ **Static or leakage power dissipation comes from:**

- Current leakage through devices that are “OFF” during static states

■ **Dynamic or switching power dissipation comes from:**

- Charging external loads
- Charging internal loads
- Short circuit or transient current while both devices are ON during a logic swing



3-27

Dynamic power dissipation, also called switching power, can occur external or internal to a standard cell:

External switching power dissipation comes from current needed to continually charge external capacitive loads.

Internal switching power dissipation comes from two sources: Current to charge internal capacitive loads, and short circuit (or transient) current through the “P” and “N” transistors, when both transistors are turned ON. This happens while their input voltage is in transition, in the range of 20-80% of VDD. During this transition time, there is a direct path (short) from VDD to VSS, through both transistors. This short circuit current is also referred to as “transient” current.

Static power dissipation, also called leakage power, comes from the fact that, even when voltage levels are stable (static, non-switching) at VSS or VDD, and the “N” or the “P” device, respectively, is OFF, there is still a small amount of “leakage” current passing through the “OFF” transistor.

Leakage Power Optimization

Low- V_{th} Cells
Fast, High Leakage



High- V_{th} Cells
Slow, Low Leakage



- A multi- V_{th} library is the key to minimize leakage power

- Low V_{th} cells are used on critical paths to help timing

- High V_{th} cells are used on non-critical paths to save leakage power

```
set target_library "hvt.db svt.db lvt.db"
create_mw_lib ... -mw_reference_library \
    "mw/sc_hvt mw/sc_svt mw/sc_lvt mw/io mw/ram32"
set_power_options -leakage true|false
```

Leakage power optimization is enabled by default with `place_opt -power`, or can be explicitly controlled with `set_power_options -leakage true|false`

3-28

Note: IC Compiler can perform some leakage power optimization without a multi- V_{th} library (downsizing gates or buffer removal) but having a multi- V_{th} library allows much better power improvement.

Some vendors include a nominal or medium V_{th} library, sometimes called the “standard” V_{th} library (`svt.db` in the example above).

By default, the maximum allowed leakage is set to 0, in order to achieve best optimization.

Alternatively, you may set a higher leakage power goal before starting the optimization, e.g.

```
set_max_leakage_power 10 mW
```

This may reduce optimization run-time.

Prior to version 2007.12 it was recommended to explicitly enable a newer leakage power optimization engine, referred to as “adaptive leakage optimization” (ALO), by turning on this variable in addition to the commands above: `set adaptive_leakage_opto true`.

Starting with 2007.12 ALO is always enabled. This variable no longer exists. If using 2007.12 or later and you have an older script that includes this variable setting, remove it before execution.

Report Multi-V_{th} Cells

After placement you can generate a report summarizing how many of each type of cells were used

Report

```
icc_shell> report_threshold_voltage_group
*****
Threshold Voltage Group Report
Threshold Voltage Group      Number of Cells      Percentage
*****
LVT                           90                  8.33%
HVT                           931                 86.12%
SVT                           59                  5.46%
undefined                     1                   0.09%
*****
```

These are typically macro or IP cells, which are not defined as low or high V_{th}

Note: This reporting requires special library attributes. See notes below.

3-29

To be able to generate the above data, the library cells must be annotated with an appropriate attribute. If they are not annotated, you can do so with the following attributes:

If high and low V_{th} cells are in separate libraries, use:

```
default_threshold_voltage_group
```

If high and low V_{th} cells are in the SAME library, use:

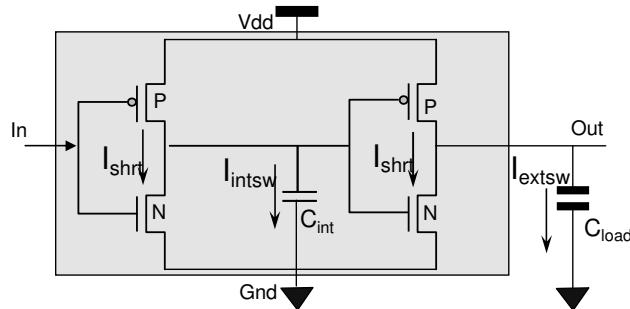
```
threshold_voltage_group
```

Example:

```
set_attribute -type string lib1 default_threshold_voltage_group HVt
set_attribute -type string lib2 default_threshold_voltage_group LVt
set_attribute -type string lib3 default_threshold_voltage_group SVt
...
report_threshold_voltage_group
```

Reducing Dynamic Power Dissipation

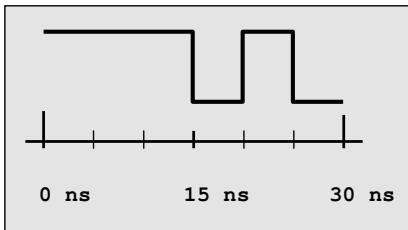
- **Dynamic power optimization reduces power by identifying high “toggle rate” nets and:**
 - Reducing wire capacitance
 - Downsizing gates
- **Two optimization techniques available during placement:**
 - Low power placement (LPP)
 - Gate-level power optimization (GLPO)



3-30

Switching Activity Terminology

- **Toggle Count (TC):**



A toggle is a logic transition ($0 \rightarrow 1$ or $1 \rightarrow 0$).
Toggle count (**TC**) is the number of toggles.

- **Toggle Rate (Tr):**

Number of toggles per unit time (usually the simulation duration)

$$Tr = TC / \text{duration}$$

- **T1 , T0:**

Total time a signal is at logic 1, 0

- **Static Probability (Sp):**

Probability of a logic 1 for a signal

$$Sp = T1 / \text{duration}$$



Using the graph on the left, fill in the following:

$$TC = \text{_____}, Tr = \text{_____}, T1 = \text{_____}, \text{ and } SP = \text{_____}$$

3-31

$$TC = 3, Tr = 1/10, T1 = 20 \text{ ns}, Sp = 0.667$$

SAIF File Provides Switching Activity

```
(SAIFFILE DESIGN.saif
(SAIFVERSION "2.0")
(DIRECTION "backward")
(DESIGN )
(DATE "Mon May 17 02:33:48 2004")
(VENDOR "Synopsys, Inc")
(PROGRAM_NAME "VCS-Scirocco-MX Power Compiler")
(VERSION "1.0")
(DIVIDER / )
(TIMESCALE 1 ns)
(DURATION 10000.00)
(INSTANCE I_TOP
  (INSTANCE macinst
    (NET
      (z\[3\]
        (T0 6488) (T1 3493) (TX 18)
        (TC 26) (IG 0)
      )
      . . .
    )
  )
)
```

```
(z\[32\]
  (T0 6488) (T1 3493) (TX 18)
  (TC 26) (IG 0)
)
. . .
)
(INSTANCE U3
(PORT
(Y
  (T0 4989) (T1 5005) (TX 6)
  (COND ((D1!*D0) | (!D1*D0))
(RISE)
  (IOPATH S (TC 22) (IG 0)
)
  COND ((D1!*D0) | (!D1*D0))
(FALL)
  (IOPATH S (TC 21) (IG 0)
)
  COND_DEFAULT (TC 0) (IG 0)
)
. . .
)
```

```
read_saif -input DESIGN.saif -instance_name I_TOP
report_saif
```

3-32

SAIF: Switching Activity Interchange Format

SAIF files are generated by simulators. A SAIF file has header information (in bold font above), and the body contains hierarchical switching activity information.

SAIF is an open standard. The standard can be downloaded from :

<http://www.synopsys.com/tapin/>

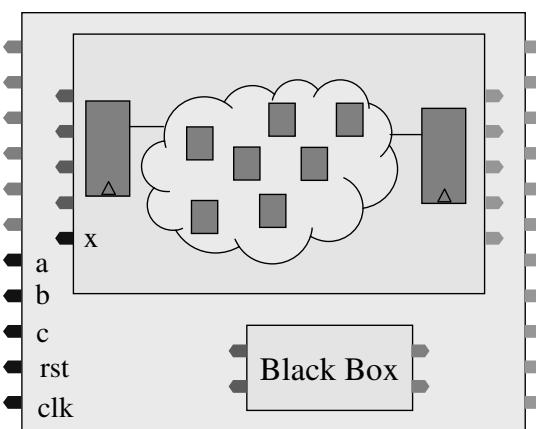
There is another switching activity format called VCD (IEEE standard). Synopsys has a utility to convert VCD into SAIF.

What if SAIF is Not Available?

```
DESIGN_toggle_rate.tcl
create_clock -p 4 [get_ports clk]
set_case_analysis 0 [get_ports "rst se tm"]
set_switching_activity -toggle_rate 0.02 a
set_switching_activity -toggle_rate 0.06 b
set_switching_activity -toggle_rate 0.11 c
set power_default_toggle_rate 0.003
report_power
... ...
```

```
source DESIGN_toggle_rate.tcl
```

- Ensure properly defined clocks
- Use `set_case_analysis` for signals that do not toggle
 - E.g. `reset`, `test_mode` and `scan_en` signals
- Specify a toggle rate on propagation starting points:
 - Set any known toggles on primary inputs and black box outputs
 - Rest of the starting points will use the default toggle rate
- IC Compiler will propagate the toggles throughout the design

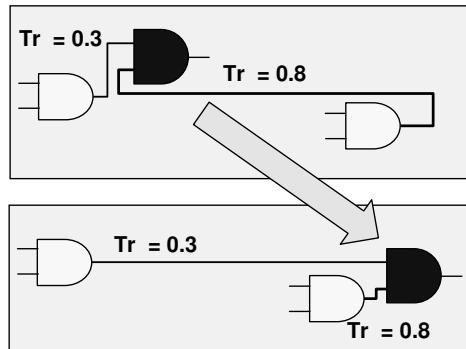


3-33

Dynamic Power Optimization: LPP

Low Power Placement (LPP):

- Moves cells closer to shorten non-clock high-activity nets
 - Clock tree LPP is done later, after place_opt, but pre-CTS
- Must be enabled – OFF by default



```
read_saif -input DESIGN.saif -instance_name I_TOP
# If SAIF is not available:
# source DESIGN_toggle_rate.tcl
set_power_options -low_power_placement true
report_power_options
...
place_opt -power
```

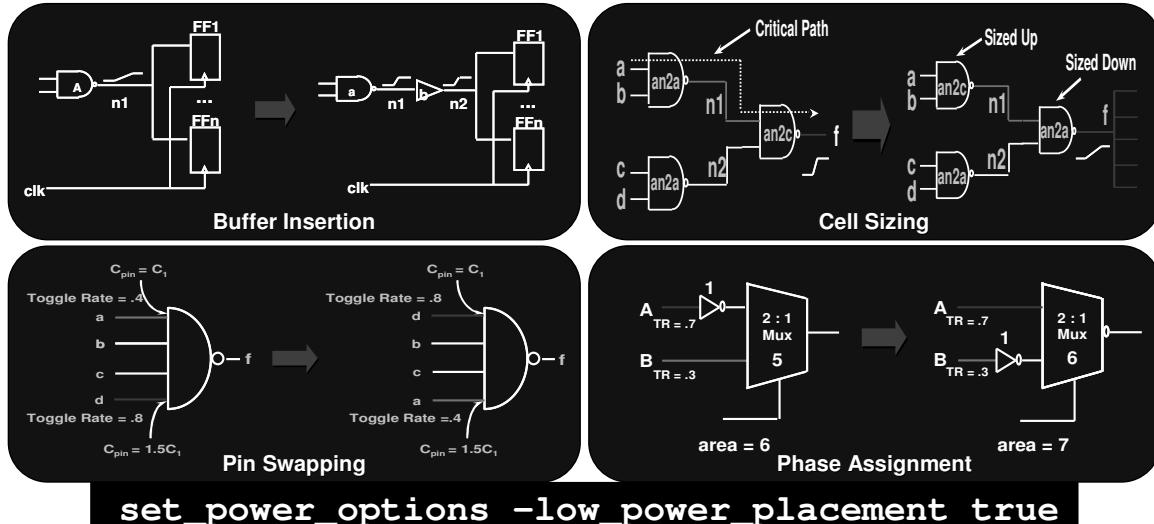
3-34

In general LPP optimizes the placement of cells connected to high toggle rate nets with the purpose of shortening these net lengths. The reduced parasitic capacitance reduces the dynamic power dissipation of these cells.

Starting with version 2007.12-SP2 LPP is performed only non-clock cells during placement. Clock tree LPP will be performed after placement, and just before CTS `clock_opt`.

Dynamic Power Optimization: GLPO

Gate-level power optimization (GLPO) optimizes logic to reduce power – must be enabled



```
set_power_options -low_power_placement true
place_opt -power
...
set_power_options -dynamic true
psynopt -power
```

*Note: GLPO can also be invoked concurrently with LPP during place_opt

3-35

* While it is possible to concurrently perform GLPO, LPP and leakage power optimization during place_opt, similar results may be obtained if GLPO is performed separately, after place_opt, during psynopt, with possibly reduced overall runtime.

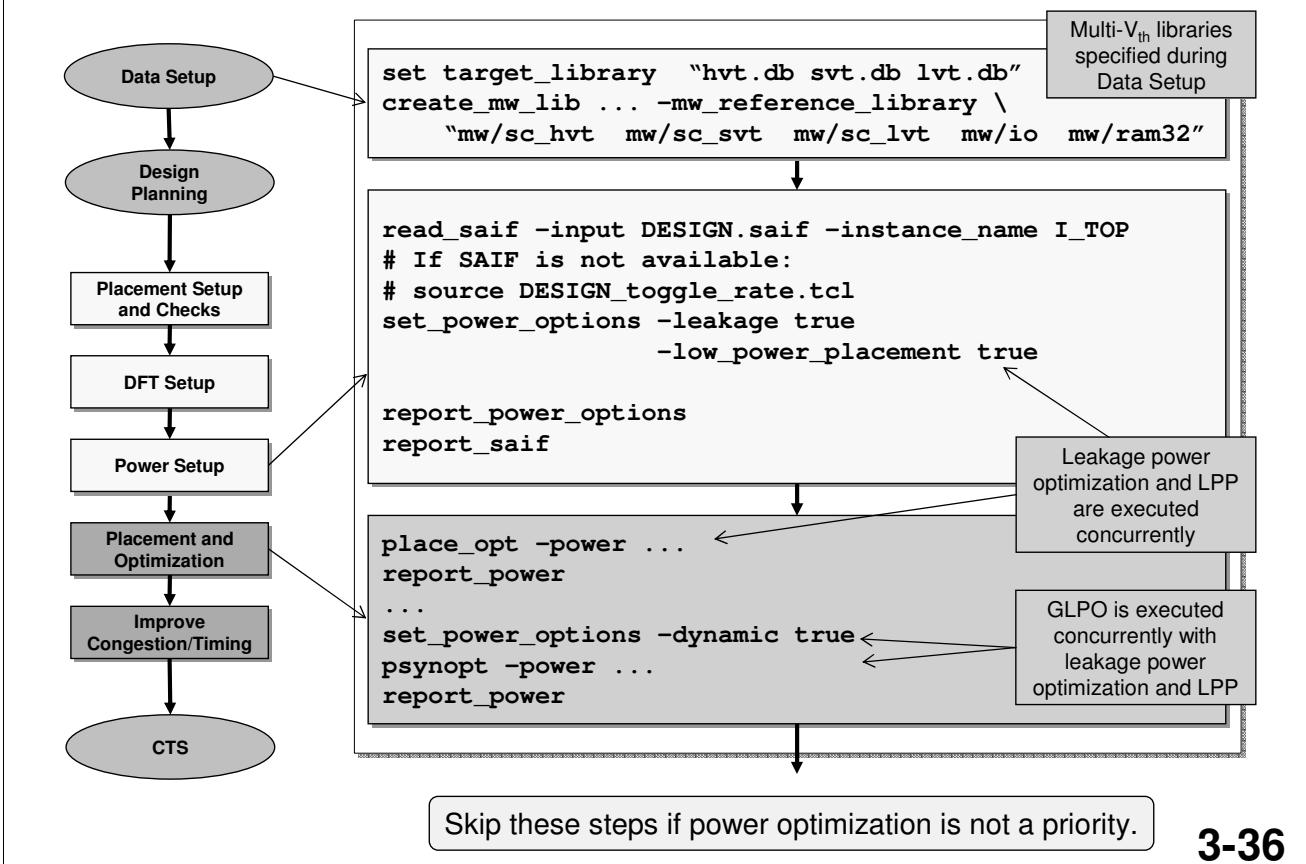
GLPO techniques:

- Buffer Insertion:** Insert buffers to reduce capacitive load and sharpen input transition time
- Sizing:** Lower capacitance on high activity nets, sharpen transition times to reduce internal power
- Pin Swapping:** Connect high toggle-rate nets to low capacitance pins
- Phase Assignment:** Use phase inversion to remove high toggle-rate inverters
- Technology Mapping:** Hide high toggle-rate nets inside cells (not shown above)
- Factoring:** Reduce the circuit's switching activity (not shown above)

The maximum leakage and dynamic power constraints are set to 0, by default, to enable maximum power optimization. Alternatively, you may set higher maximum leakage and/or dynamic power goals before starting the optimization. This may help to reduce run-time:

```
set_max_leakage_power 10 uW
set_max_dynamic_power 500 mW
```

Summary: Power Optimization Flow



Test For Understanding

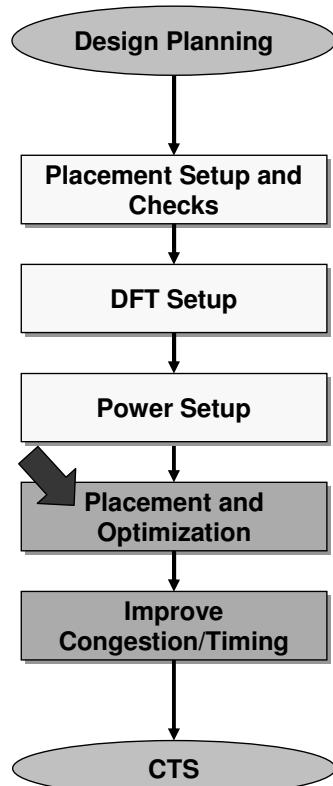


1. NDR rules define non-default rules for routing. Why apply these before placement?
2. The **SCANDEF** file contains the scan ordering information which will be maintained during placement with `place_opt -optimize_dft`. *True or False?*
3. Leakage power optimization is enabled by default with `place_opt -power`, and is more effective with multiple V_{th} libraries. *True or False?*
4. **Low Power Placement (LPP)**
 - a. Moves cells to shorten high-activity nets
 - b. Spreads out high activity cells to reduce power density
 - c. Is enabled with `set_power_options -dynamic true`
 - d. All of the above
5. SAIF is preferred over user-defined toggle rates for accurate leakage power optimization. *True or False?*

3-37

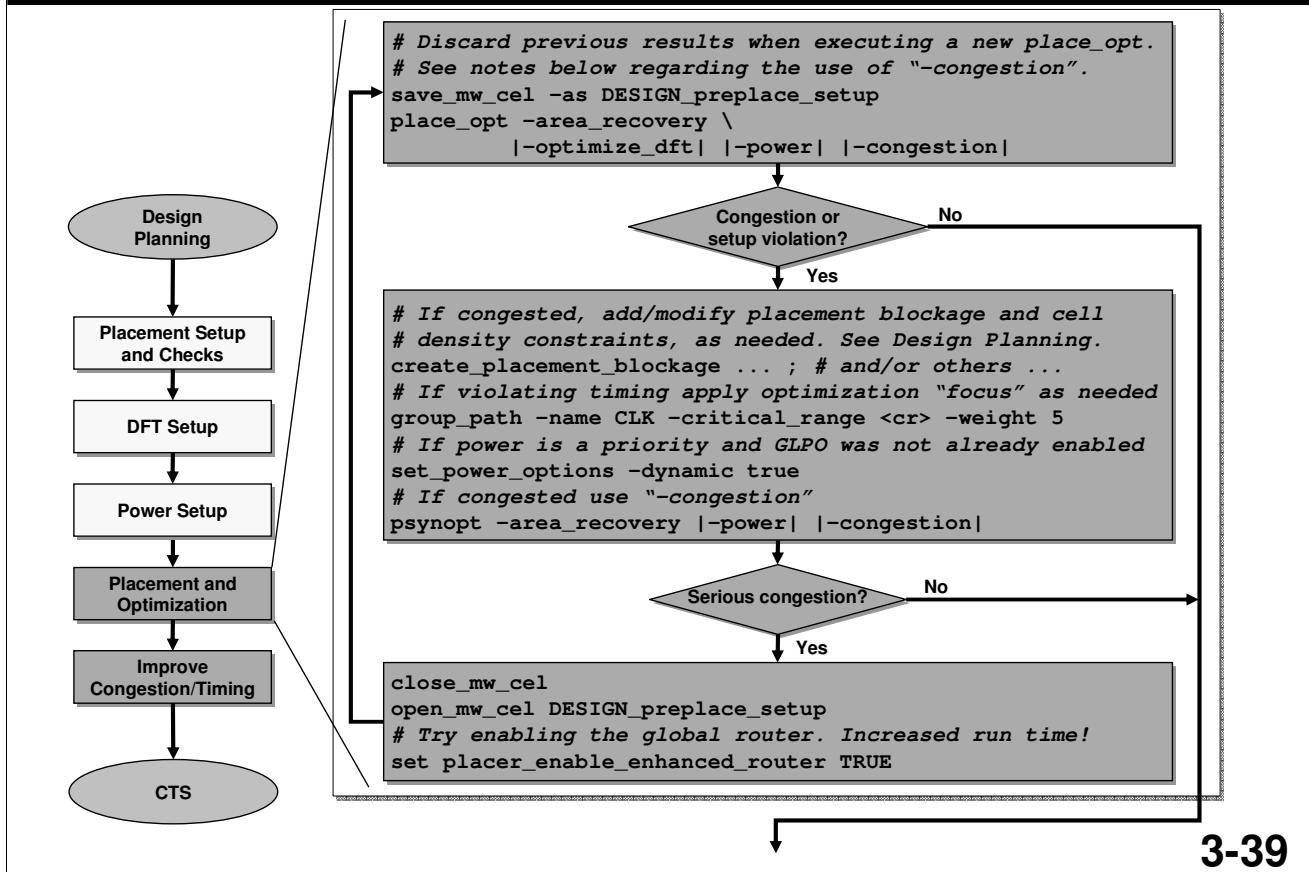
1. Even though no actual routing occurs during placement, interconnect parasitics and delays are accounted to determine path timing, and NDR rules affect timing, which should be taken into account during the placement phase.
2. False. `PLACE_OPT -optimization_dft` uses SCANDEF to identify the "reordering buckets" within which, and the "partitions" across which, it can re-order the scan chains to achieve optimal routing.
3. True.
4. A.
5. False. Toggle rates (SAIF or user-defined) are only used for dynamic power optimization.

Placement and Optimization



3-38

Overview: Placement and Optimization



If `place_opt` is first executed without `-congestion`, analyze congestion immediately after completion. If the design is congested, **discard the previous result** and execute a new `place_opt` with `-congestion`:

```

save_mw_cel -as DESIGN_prelace_setup
place_opt -optimize_dft -power -area
...           # Analyze results - if congested:
close_mw_cel
open_mw_cel DESIGN_prelace_setup
place_opt -optimize_dft -power -area -congestion

```

Use `-congestion` **only** if congestion is known to be a serious issue prior to placement. This is the case if one of the following is true:

1) During design planning using IC Compiler, the “high effort” congestion algorithm was used to reduce congestion:

```

set_fp_placement_strategy -congestion_effort high
create_fp_placement -congestion -timing

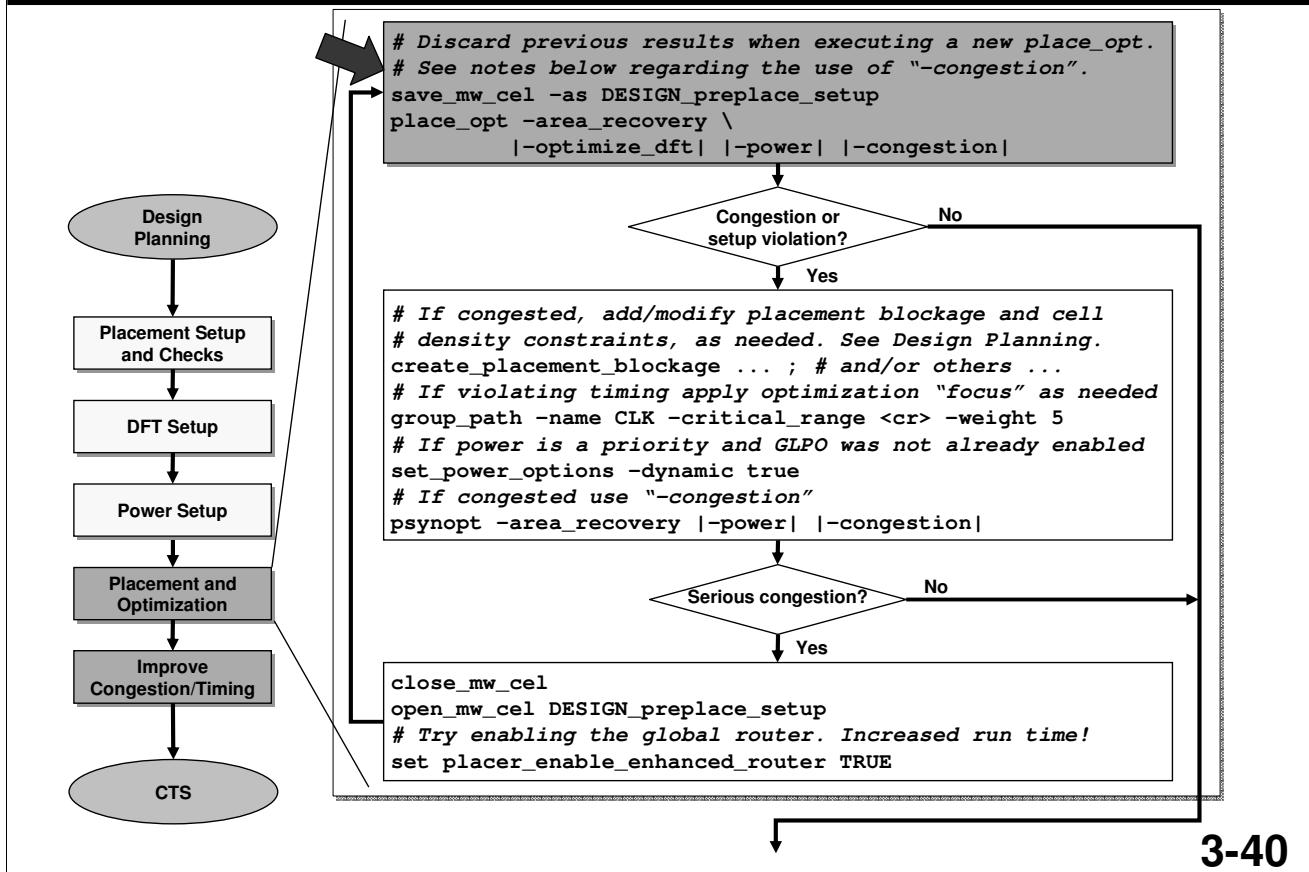
```

2) During floorplan exploration, after design planning using a 3rd party tool, `-congestion` was required to reduce congestion:

```
place_opt -effort low -congestion
```

3) Placement was first performed without `-congestion`, and the resulting design was congested.

The Initial Placement and Optimization



If `place_opt` is first executed without `-congestion`, analyze congestion immediately after completion. If the design is congested, **discard the previous result** and execute a new `place_opt` with `-congestion`:

```

save_mw_cel -as DESIGN_preplace_setup
place_opt -optimize_dft -power -area
...           # Analyze results - if congested:
close_mw_cel
open_mw_cel DESIGN_preplace_setup
place_opt -optimize_dft -power -area -congestion

```

Use `-congestion` **only** if congestion is known to be a serious issue prior to placement. This is the case if one of the following is true:

- 1) During design planning using IC Compiler, the “high effort” congestion algorithm was used to reduce congestion:

```

set_fp_placement_strategy -congestion_effort high
create_fp_placement -congestion -timing

```

- 2) During floorplan exploration, after design planning using a 3rd party tool, `-congestion` was required to reduce congestion:

```
place_opt -effort low -congestion
```

- 3) Placement was first performed without `-congestion`, and the resulting design was congested.

Placement and Logic Optimization

place_opt

Performs timing- and congestion-driven placement and logic optimization

-area_recovery

Enables buffer removal and cell down-sizing of non-critical paths

-optimize_dft

Performs scan chain re-ordering

-power

Invokes leakage and/or dynamic power optimization

-congestion

Invokes additional congestion-driven algorithms

place_opt

Coarse Placement

↓

↓

AHFS

Logic Optimization

↓

Placement Legalization

```
place_opt -area_recovery \
           |-optimize_dft| |-power| |-congestion|
```

Consider:

- Using **-area_recovery**: Can help reduce congestion and power; May impact run time
- Using **-optimize_dft** and **-power** only if DFT and power optimization are required, resp.
- Using **-congestion** only if congestion is known to be a serious issue (see next slide)

3-41

To reduce run time IC Compiler can perform parallel processing during coarse placement. Specify the number of free *cpus* with the option: **-num_cpus <#>**.

Considerations for Using -congestion

```
place_opt ... -congestion
```

Use -congestion only if congestion is known to be a serious issue prior to placement, for example:

- **place_opt was first performed without -congestion and the resulting design was congested**
 - Discard previous results before re-running place_opt
- **During design planning (using IC Compiler) the “high effort” congestion algorithm was invoked:**
`set_fp_placement_strategy -congestion_effort high
create_fp_placement -congestion -timing`
- **During design exploration (after 3rd party design planning) -congestion was required:**
`place_opt -effort low -congestion`

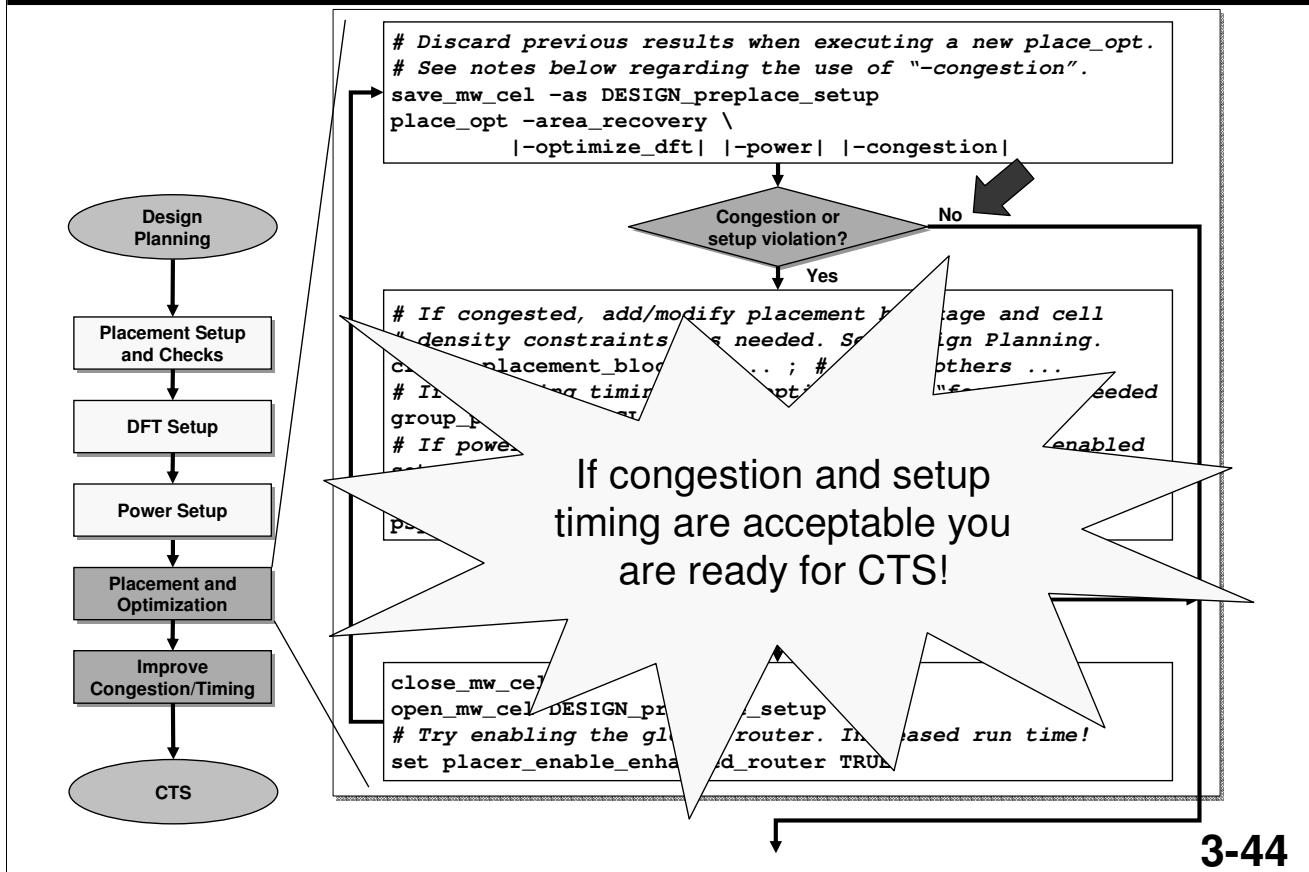
3-42

No Hold Time Fixing

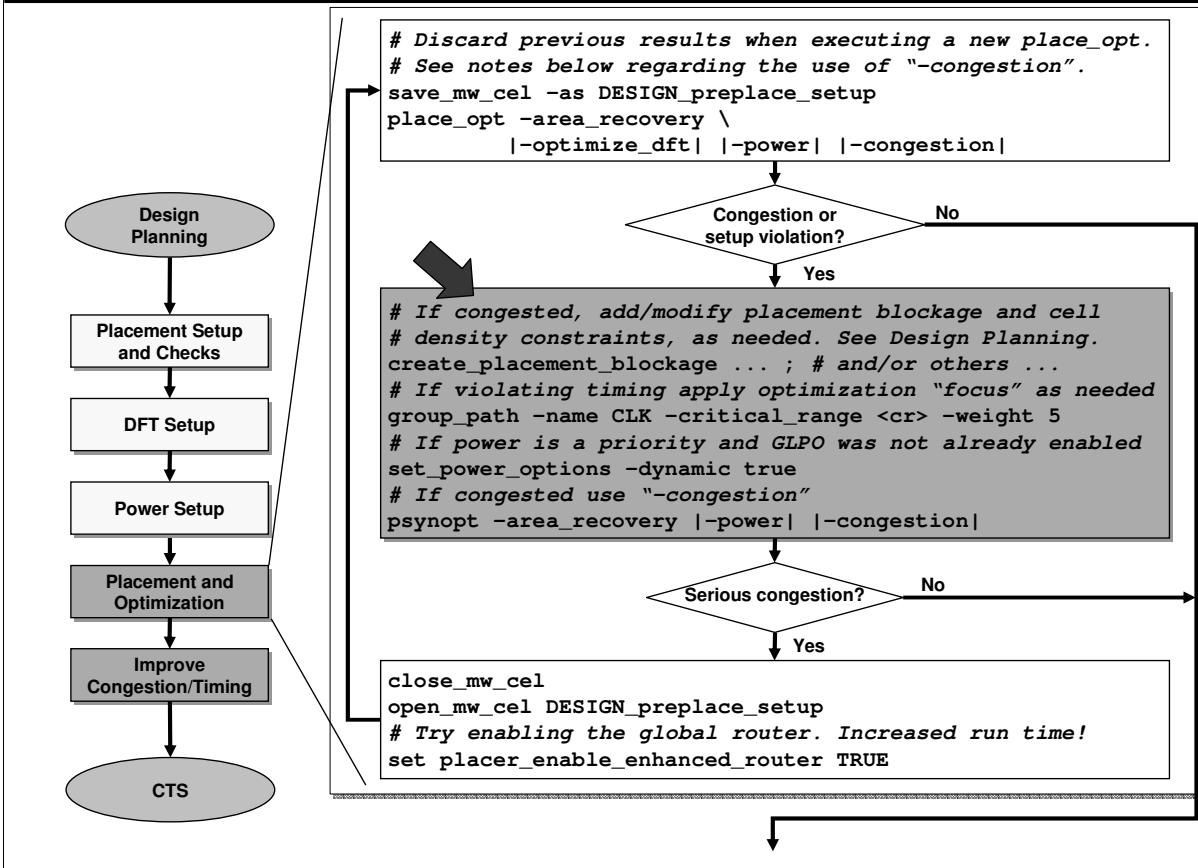
- By default `place_opt` tries to fix only setup time violations - No hold time fixing
- Hold time will be addressed during clock tree synthesis

3-43

Post-Placement Analysis



Incremental Optimization



3-45

Apply Placement Constraints As Needed

If the design is congested after `place_opt` analyze the congestion, and if applicable:

- Apply new, or modify existing placement blockages
- Apply new, or modify existing cell density constraints
 - See “Design Planning” for details

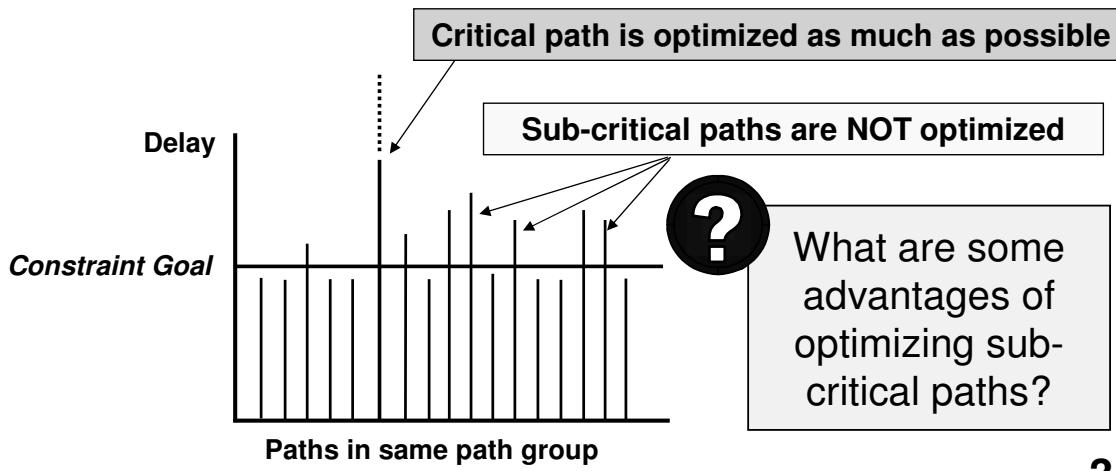
Remember to also add to
.synopsys_dc.setup

```
set physopt_hard_keepout_distance ...
set placer_soft_keepout_channel_width ...
set congestion_options ...
set keepout_margin ...
create_placement_blockage ...
```

3-46

Recall Problem: Sub-Critical Paths Ignored

- By default, optimization within a path group stops if:
 - IC Compiler becomes “stuck” on the critical path
 - Sub-critical paths are not optimized!



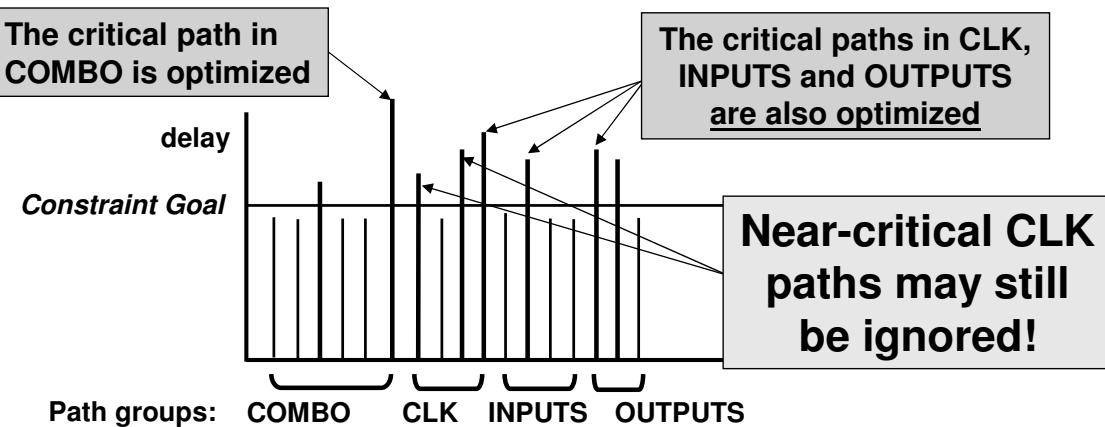
3-47

ANSWER:

If the sub-critical paths are intertwined with the critical path (they share logic), it is possible that by improving one or more sub-critical nets this actually helps the critical path as well. Another advantage is that the design will end up with fewer violations, which may be easier to fix during CTS and/or routing.

Solution #1: User-defined Path Groups

```
# Ensure that the reg-reg paths get optimized
group_path -name INPUTS -from [all_inputs]
group_path -name OUTPUTS -to [all_outputs]
group_path -name COMBO -from [all_inputs] -to [all_outputs]
```



3-48

The reg-to-reg paths remain in the CLK path group, which is created by default by IC Compiler. The `group_path` command can be used for CLK to assign a non-default *weight* or *critical range* (to be discussed) to the group, but is not needed not to define the reg-to-reg paths.

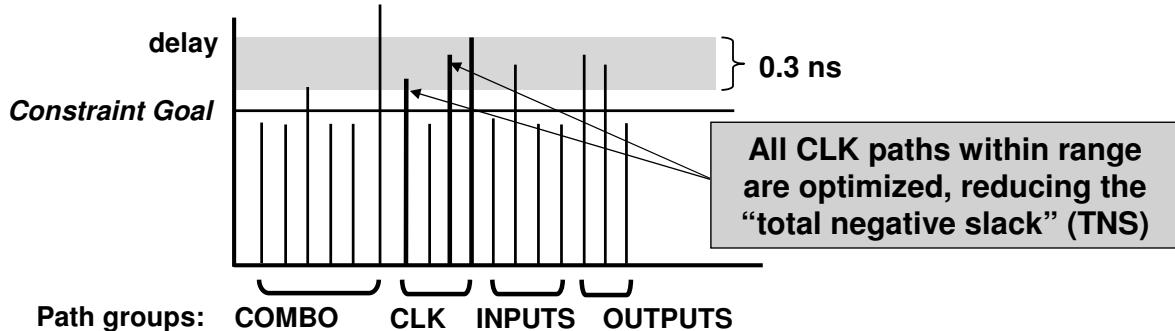
A path can only be in one path group, but according to the path group arguments, the combo paths can be part of the INPUTS, OUTPUTS or COMBO group – so where are they?

The COMBO paths wind up in the COMBO group. Assuming the commands are executed in the order listed in the slide, they are first moved from CLK to INPUTS, because they match the `startpoint` argument `-from [all_inputs]`. They will not be moved to the OUTPUTS group, even though their endpoints match `-to [all_outputs]`, because “`-from`” has priority over “`-to`”. They finally end up in the COMBO group because their startpoints **and** endpoints match the `-from` AND `-to` arguments. IC Compiler works this way to prevent having different results if the command sequence changes! The results are independent of the order of the commands above.

Use `report_path_group` to get a summary of the path groups in the design.

Solution #2: Apply a *Critical Range*

```
group_path -name CLK -critical 0.3
```



- The *critical range* is always relative to the critical path delay
- Fixing related sub-critical paths may help the critical path
- Should not exceed 10% of the group's effective clock period
 - Will increase run time

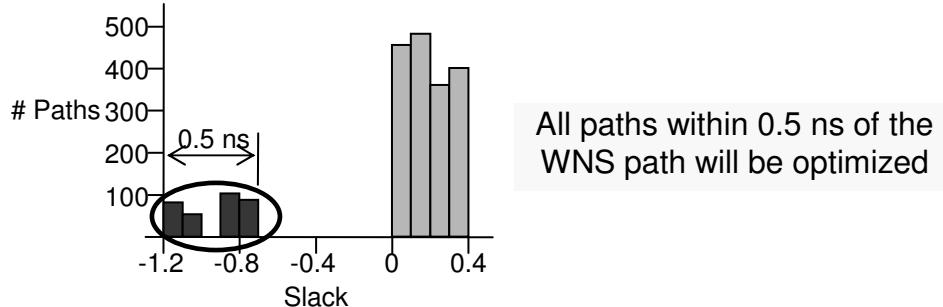
3-49

By default the critical range of all path groups is zero.

Fixing related sub-critical paths may help the critical path. Since the critical range is with respect to the critical path delay, if the critical path delay is improved, the critical range band moves lower, along with the improved critical path. Critical range optimization will not improve a sub-critical path if the improvements make the critical path worse.

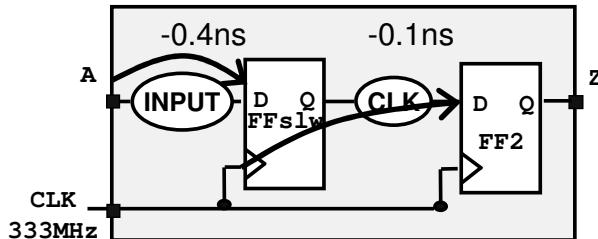
With a critical range, optimization will reduce TNS (Total Negative Slack) in the design even if it cannot reduce the WNS (Worst Negative Slack).

Alternatively you can apply one critical range value globally to all path groups, for example: `set_critical_range 0.5 [current_design]`. While this approach is a little simpler to implement (you need to only decide on one critical range value), it may have a larger impact on run-time.



Solution #3: Prioritizing Path Groups

- All path groups have equal priority, by default
- Applying a *weight* to a select group (e.g. CLK) allows delay improvements to its paths, which may degrade another, less critical group's worst violator (e.g. INPUTS)



- Suggestion:

- Apply a weight of 5 to the most critical paths (reg-to-reg)
- Apply a weight of 2 to less critical paths
- A default weight of 1 is assigned to all other paths (e.g. over-constrained I/O paths)

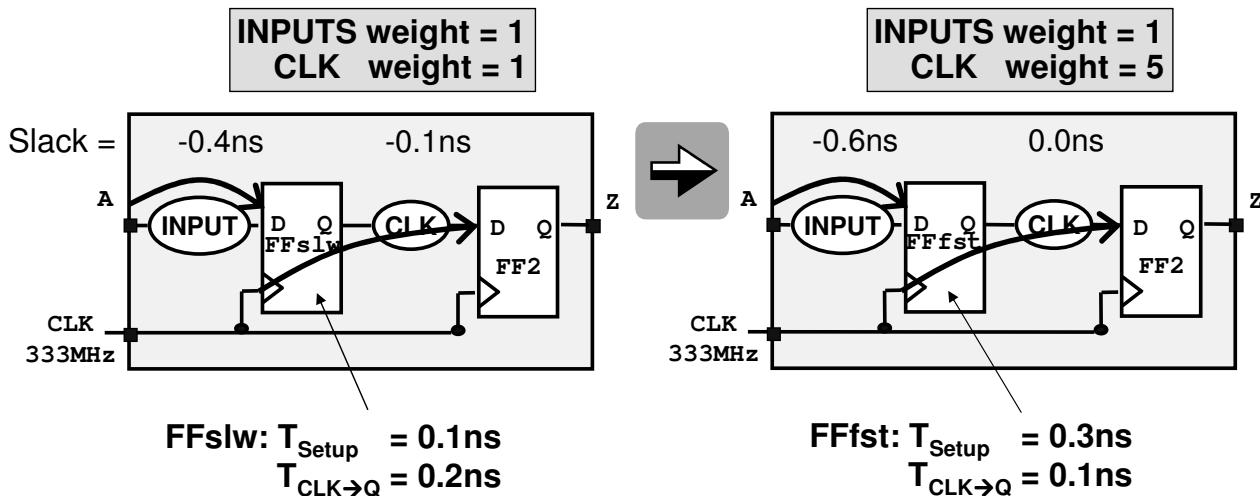
```
group_path -name CLK -weight 5
```

3-50

It is possible to assign *weights* to different path groups by using the *-weight* option. This is a way to control the relative priority of optimization. Priority is given to the path with the highest “Cost” = Negative slack X Weight. For example: A path group with a critical violation of -2ns and a *weight* of 5 has a *cost* of 10 and will therefore have higher priority than another path group with a critical violation of -3ns but a default *weight* of only 1 - *cost* of 3. If these two paths are related such that improving one hurts the other, optimization will favor improving the higher weight path, even though its slack is less than the lower weight path.

Example: -weight

```
group_path -name INPUTS -from [all_inputs]
group_path -name OUTPUTS -to [all_outputs]
group_path -name COMBO -from [all_inputs] -to [all_outputs]
group_path -name CLK -weight 5
```



3-51

The example above illustrates how it is possible that by adding a path group, in this case INPUTS, the worst violating path in the design, the input path with a slack of -0.4ns, actually gets worse (-0.6ns), in favor of helping the higher priority reg-to-reg path. This happened because this design change reduced the overall cost function. The cost function for the circuit on the left is $[(0.4 \times 1) + (0.1 \times 5)] = 0.9$. For the circuit on the right the cost function is 0.6. Optimization was able to improve the reg-to-reg setup timing path by switching to a register with a faster CLK→Q delay (FFslw → FFfst), while giving up some setup time.

Without the `-weight` option applied to the CLK group both paths have a default weight of 1 and optimization will not consider this flip-flop swap, because that would *increase* the overall cost from 0.5 $[(0.4 \times 1) + (0.1 \times 1)]$ to 0.6 $[(0.6 \times 1) + (0 \times 1)]$.

Complete Example

```
# Example: Assign a critical range and weight to clock groups
group_path -name CLK1 -critical_range 0.3 -weight 5
group_path -name CLK2 -critical_range 0.1 -weight 5
group_path -name CLK3 -critical_range 0.2 -weight 5
group_path -name INPUTS -from [all_inputs]
group_path -name OUTPUTS -to [all_outputs]
group_path -name COMBO -from [all_inputs] -to [all_outputs]
report_path_group
```

3-52

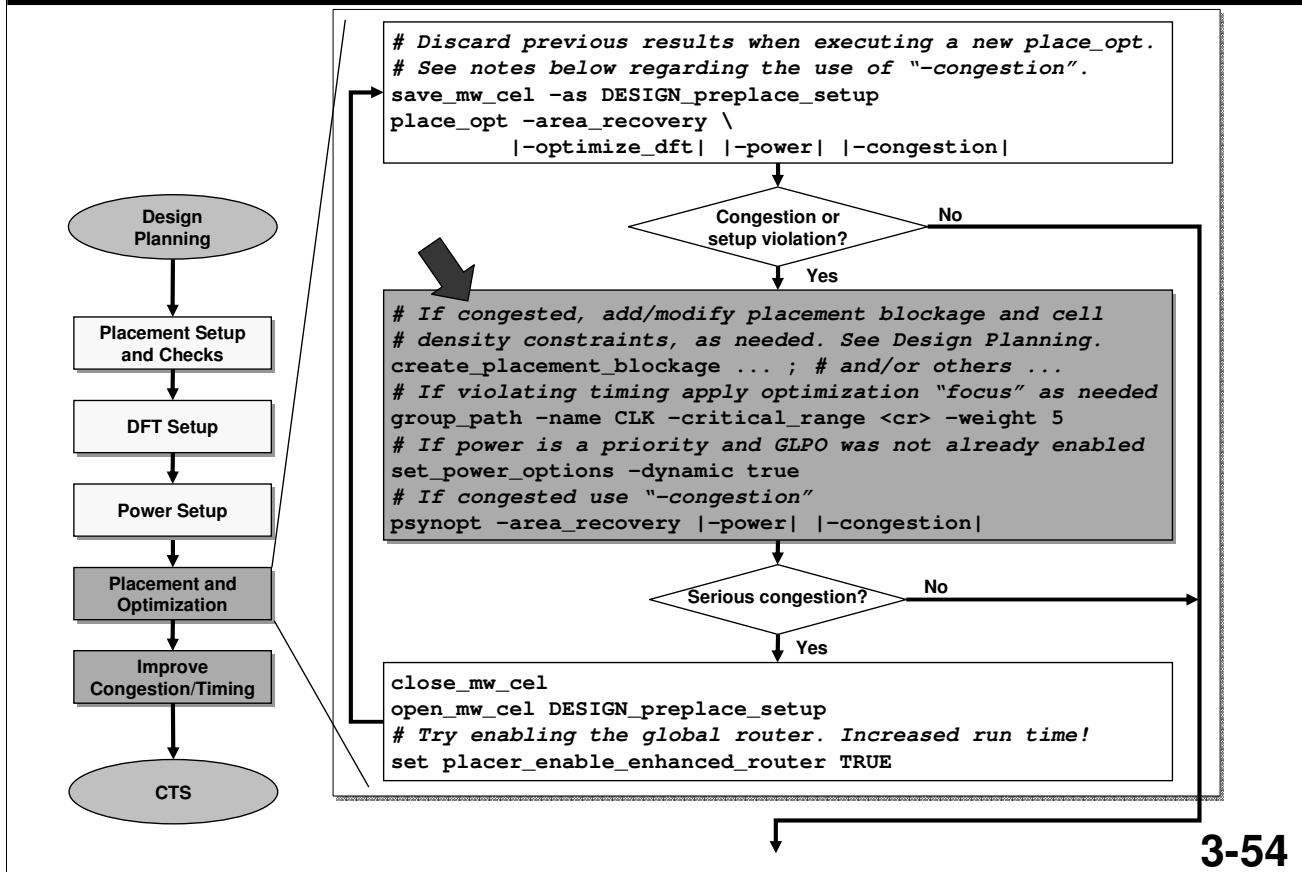
Incremental Logic Optimization: psynopt

- After modifying placement and path group parameters (as appropriate) execute **psynopt**:
 - Performs incremental timing-driven logic optimization
 - Legalizes placement
- Include **-area_recovery**
 - May help reduce congestion and power – may impact run time
- If power optimization is a priority include **-power**
 - Enable *GLPO* dynamic power optimization (if not already enabled prior to `place_opt -power`)
- If the design is congested include **-congestion**

```
set_power_options -dynamic true; # GLPO
psynopt -area_recovery |-power| |-congestion|
```

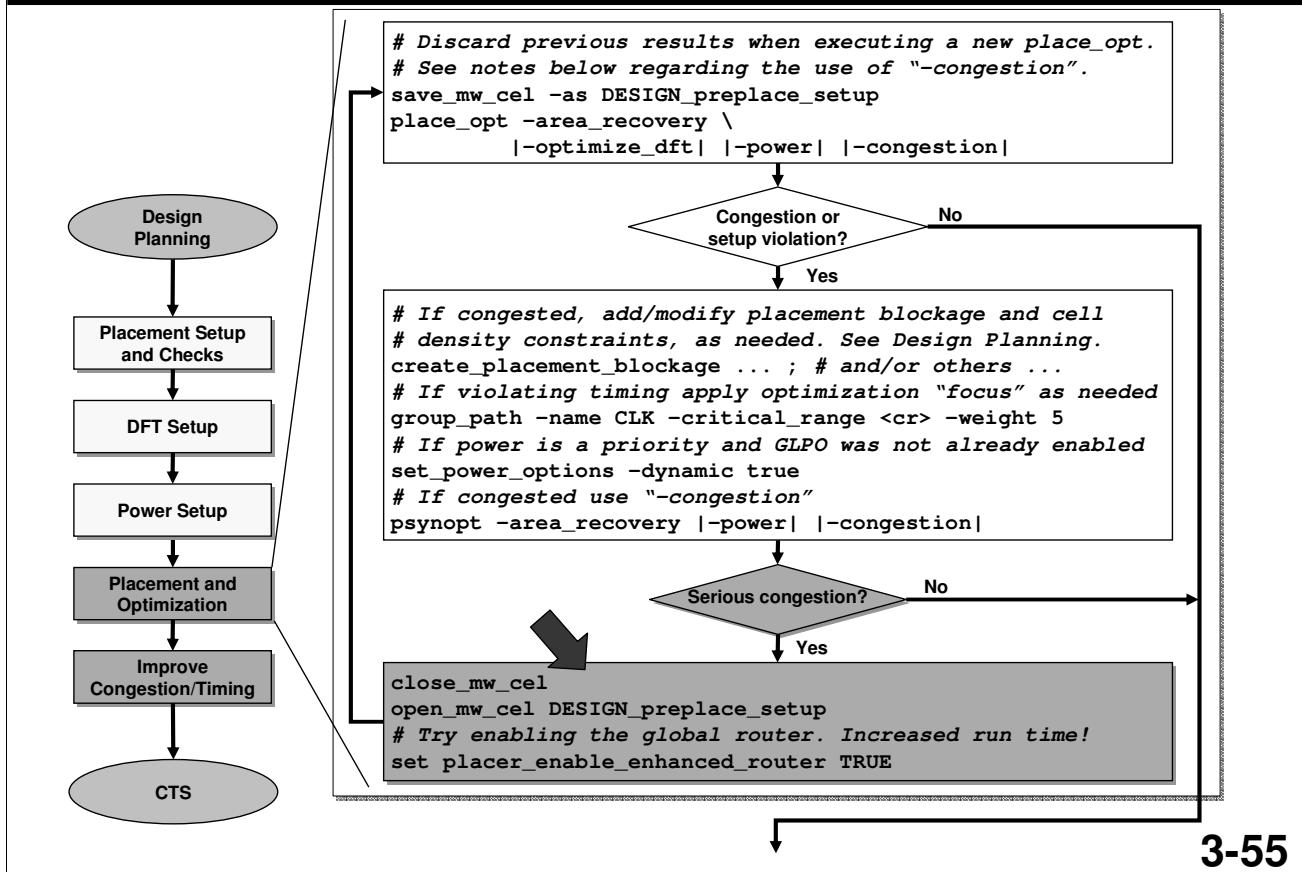
3-53

Summary: Incremental Optimization



3-54

If the Design is Still Seriously Congested ...



3-55

Enable Global Router During Optimization

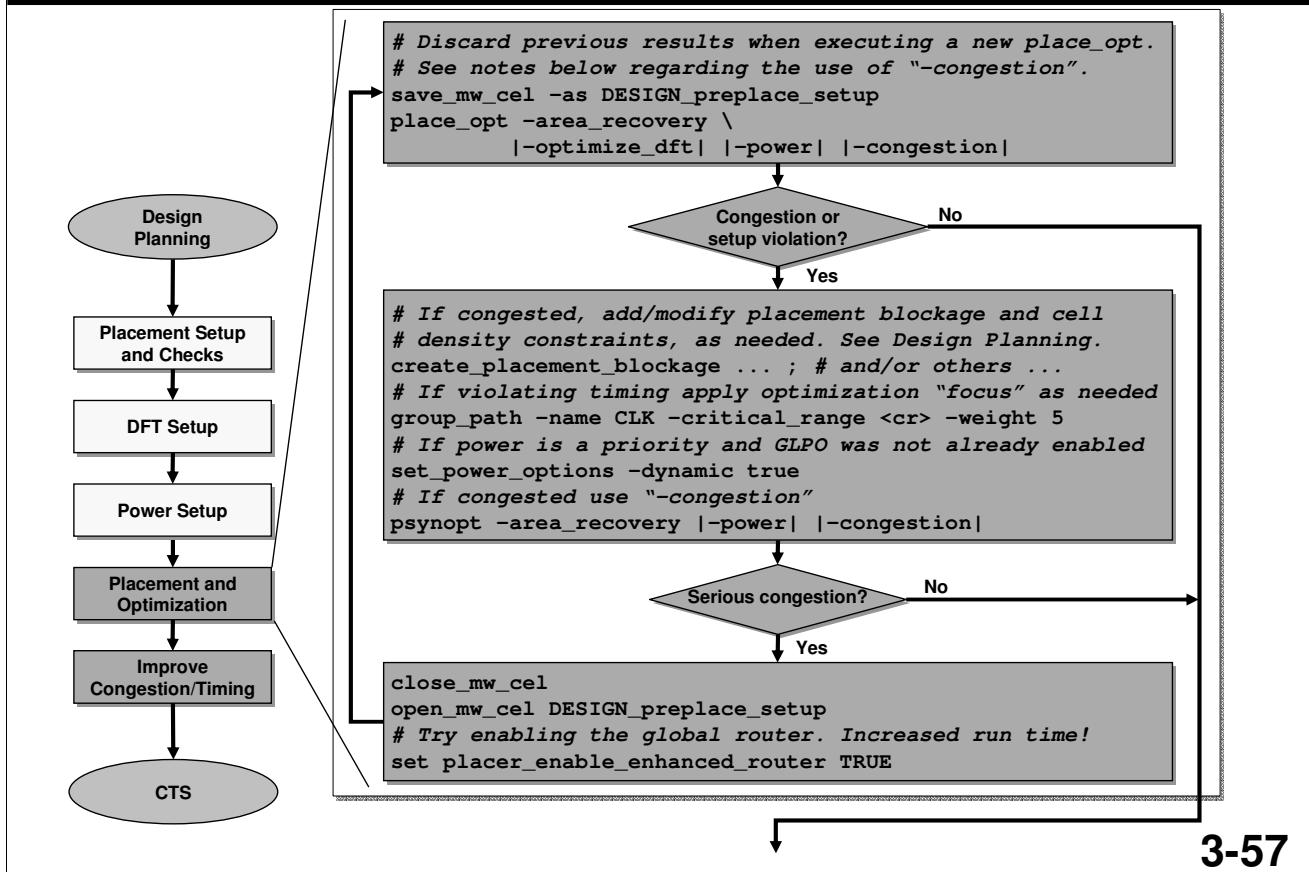
- Setting the variable below to **TRUE** enables congestion optimization using the actual global router (instead of a global route estimator)
- Most designs will show little change – some may show significant congestion improvement
- All designs will see an increase in run time!
 - Try this only if congestion is serious
 - First discard results from previous `place_opt`
 - Re-run `place_opt` with `-congestion`

```
close_mw_cel
open_mw_cel DESIGN_preplace_setup
set placer_enable_enhanced_router TRUE
place_opt -area_recovery -congestion \
           |-optimize_dft| |-power|
```

3-56

By default, `place_opt` performs both placement and netlist synthesis (or logic optimization). At the start of `place_opt` any previous placement is discarded, but the starting netlist is kept and incrementally optimized. Better results may be obtained when allowing logic optimization to start fresh on the original pre-`place_opt` netlist. It is a good practice to discard previous `place_opt` results and start from the original netlist when performing a subsequent `place_opt`.

Summary: Placement and Optimization



If `place_opt` is first executed without `-congestion`, analyze congestion immediately after completion. If the design is congested, **discard the previous result** and execute a new `place_opt` with `-congestion`:

```

save_mw_cel -as DESIGN_prelace_setup
place_opt -optimize_dft -power -area
...           # Analyze results - if congested:
close_mw_cel
open_mw_cel DESIGN_prelace_setup
place_opt -optimize_dft -power -area -congestion

```

Use `-congestion` **only** if congestion is known to be a serious issue prior to placement. This is the case if one of the following is true:

1) During design planning using IC Compiler, the “high effort” congestion algorithm was used to reduce congestion:

```

set_fp_placement_strategy -congestion_effort high
create_fp_placement -congestion -timing

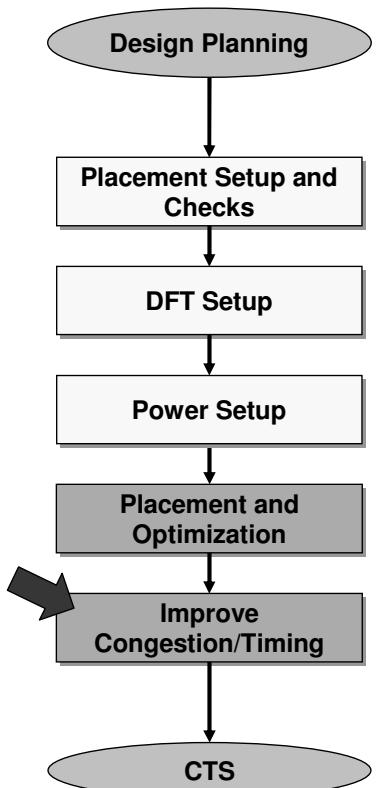
```

2) During floorplan exploration, after design planning using a 3rd party tool, `-congestion` was required to reduce congestion:

```
place_opt -effort low -congestion
```

3) Placement was first performed without `-congestion`, and the resulting design was congested.

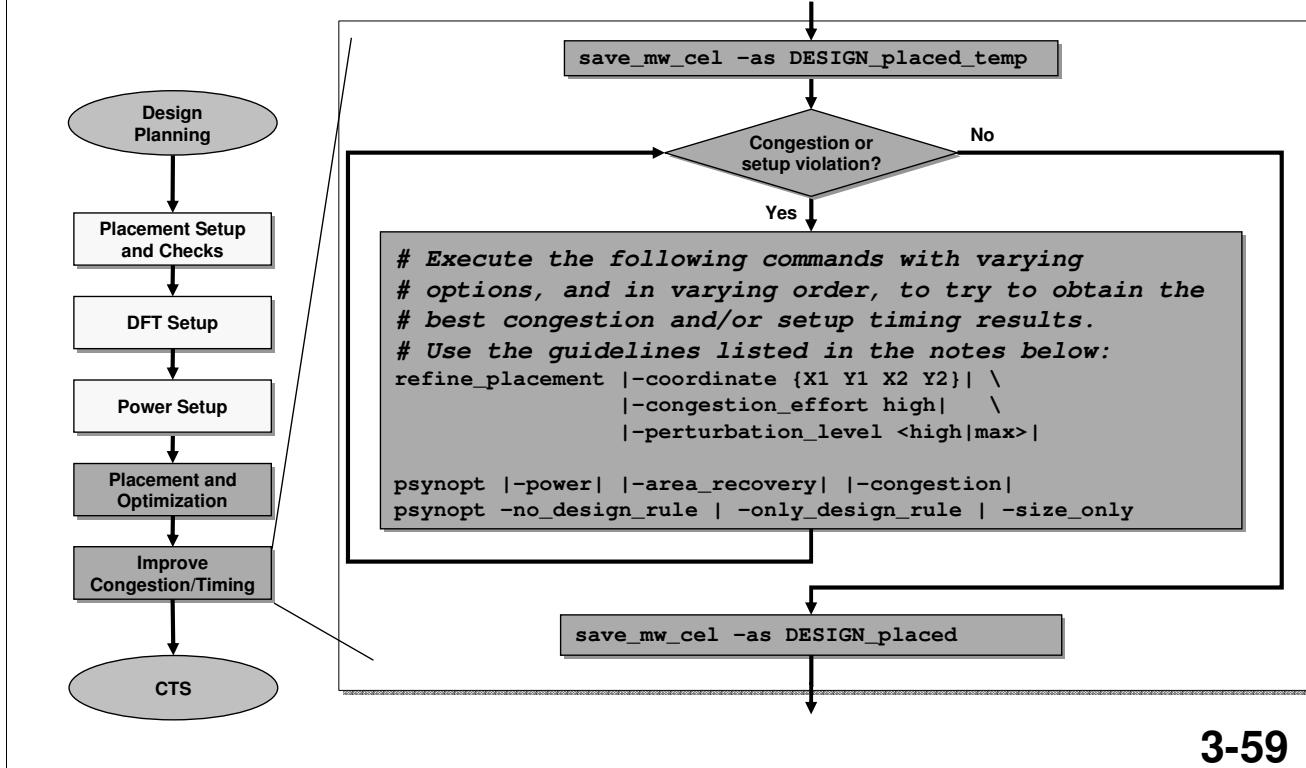
Improving Congestion and Setup Timing



If setup timing or congestion are still a problem there are additional optimizations that can be done

3-58

Overview: Improve Congestion/Timing

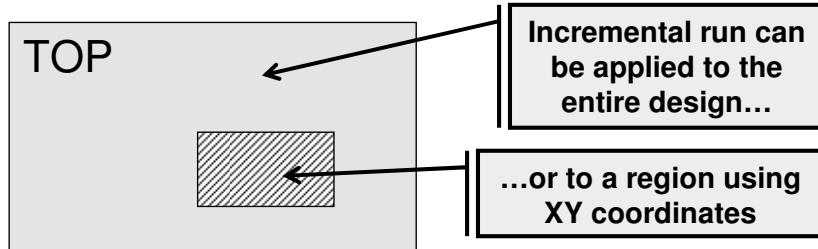


3-59

1. Since `refine_placement` focuses on congestion optimization only it may degrade setup timing
2. Sometimes better congestion results are obtained by discarding the previous result and reloading `DESIGN_placed_temp` before executing another `refine_placement`.
3. By default `psynopt` performs logic optimization, which tries to improve setup timing. This may impact congestion. When used with `-congestion` it will also try to improve congestion.
4. Use `psynopt` with `-power`, `-congestion` or `-area` as you feel is necessary. If timing is higher priority try omitting one or more of these options.
5. Sometimes better timing is obtained by focusing or limiting `psynopt` with `-no_design_rule`, `-only_design_rule` or `-size_only`
6. It is not necessary to discard previous results before executing a subsequent `psynopt`

refine_placement

- Optimizes placement incrementally to improve congestion
 - Does not modify the netlist → Timing may degrade
- Can be executed back-to-back with different options
 - Discarding previous placement may improve results
- Can be followed with psynopt to improve timing



```
refine_placement [-coordinate {x1 y1 x2 y2}] \
                  [-congestion_effort high] \
                  [-perturbation_level <high|max>]
```

3-60

```
refine_placement
  -coordinate {x1 y1 x2 y2}
  -congestion_effort low|medium|high
  -perturbation_level min|medium|high|max
  -ignore_scan (ignore scan chain connections during placement)
  -num_cpus number_of_cpus
```

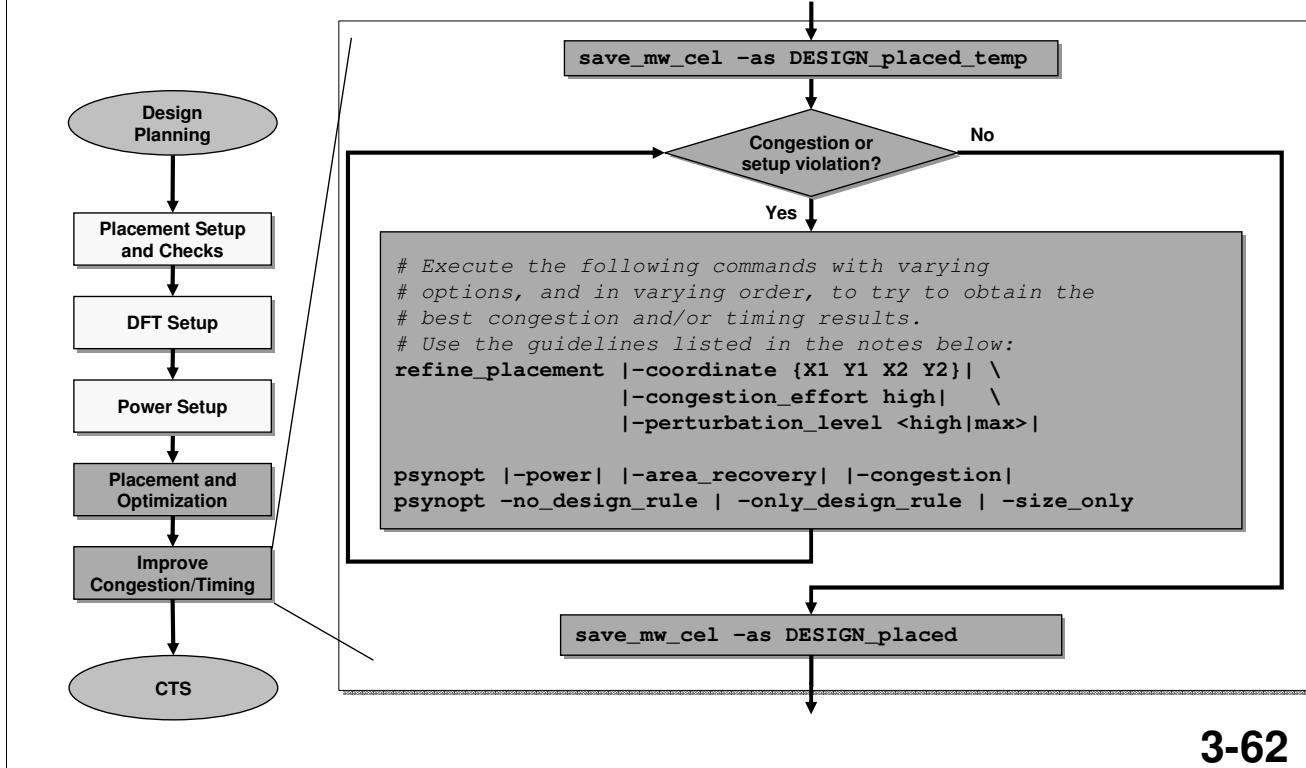
psynopt

- **Performs incremental timing-driven logic optimization with placement legalization (setup timing, by default)**
 - With `-congestion` will try to maintain or improve congestion
- **Use `-power`, `-congestion` or `-area` as necessary**
 - If timing is higher priority try omitting one or more of these options
- **Sometimes better timing is obtained by focusing or limiting psynopt with `-no_design_rule`, `-only_design_rule` or `-size_only`**
- **It is not necessary to discard previous results before executing a subsequent psynopt**

```
psynopt |-power| |-area_recovery| |-congestion|
psynopt -no_design_rule | -only_design_rule | -size_only
```

3-61

Summary: Improve Congestion/Setup Timing



3-62

1. Since `refine_placement` focuses on congestion optimization only it may degrade setup timing
2. Sometimes better congestion results are obtained by discarding the previous result and reloading `DESIGN_placed_temp` before executing another `refine_placement`.
3. By default `psynopt` performs logic optimization, which tries to improve setup timing. This may impact congestion. When used with `-congestion` it will also try to improve congestion.
4. Use `psynopt` with `-power`, `-congestion` or `-area` as you feel is necessary. If timing is higher priority try omitting one or more of these options.
5. Sometimes better timing is obtained by focusing or limiting `psynopt` with `-no_design_rule`, `-only_design_rule` or `-size_only`
6. It is not necessary to discard previous results before executing a subsequent `psynopt`

Test For Understanding (1 of 2)



1. By default, `place_opt` will
 - a. Optimize placement and logic for congestion
 - b. Optimize placement and logic for setup timing
 - c. Optimize logic for leakage power
 - d. A and B
 - e. A, B and C
2. What is the benefit of having separate path groups for I/O logic paths?
3. What does applying a timing critical range do, and what is the benefit of this?
4. Applying a *weight > 1* to a path group may increase a design's critical path delay. True or False?

3-63

1. While the parameter `set_power_opt` -leakage is true by default, `-power` must be used to invoke leakage power optimization.
2. By putting input and output logic paths in their own path groups, all the register-to-register paths remain in their original respective clock groups. The benefit of this is that, if the WNS path is an I/O path and optimized within each clock group, which may have a smaller violation than the WNS path, will still be optimized. Within the I/O path groups, these reg-to-reg violations may be totally ignored.
3. Applying a timing critical range allows near-critical paths to be optimized. These paths may otherwise have been ignored if the critical path gets "stuck" and can't be optimized further. The benefit of optimizing near-critical paths is that you end up with fewer and/or smaller violations. Also, if the paths are "related", fixing a near-critical path may also improve the critical path delay.
4. True. The goal of timing optimization is reduce the total negative slack (TNS) as much as possible. TNS is the sum of all "weighted" negative slacks. Every violation is weighted by its path group weight, default of 1. In a group with a weight of 5 for example, a 0.1ns path delay (increasing TNS by 0.3), the overall TNS is still reduced by 0.2ns (0.5-0.3), so IC Compiler considers this an acceptable trade-off.

Test For Understanding (2 of 2)



5. `refine_placement` performs incremental

- a. Timing-driven placement
- b. Congestion-driven placement
- c. Congestion-driven logic optimization
- d. A and B
- e. A, B and C

6. By default, `psynopt` performs incremental

- a. Timing-driven logic optimization
- b. Congestion-driven placement
- c. Congestion-driven logic optimization
- d. A and B
- e. A, B and C

3-64

- driven logic optimization is invoked with the `-congestion` option.
- 6. In terms of placement, only placement legalization is performed, by default. Congestion-
 - 5. B.

Techniques with More User Control



**Buffer trees
Relative placement**

3-65

Build User-Controlled Balanced Buffer Trees

- AHFS buffers high fanout nets with little user control
 - All types of buffers may be used
 - Buffer tree depth may be unbalanced, since only DRC focused
- After `place_opt` analyze the resulting buffer tree(s)
- Remove unwanted buffer tree(s)
- Re-create a balanced buffer tree with more user control:

```
remove_buffer_tree -from <pins_or_nets>
set_cbt_options -references <buffer_list> -threshold <t>
create_buffer_tree -from <pins_or_nets>
```

3-66

cbt = create buffer tree

The `create_buffer_tree` command applies to the specified pins or nets, independent of fanout (the AHFS fanout threshold is not considered). Its main goal is to build a buffer tree with balanced buffer depth, that also meets DRC rules. Timing is not considered. A balanced buffer tree may help to reduce skew, which may help timing. (See next slide for a more direct way to optimize for skew).

The `set_cbt_options` command can be used to define a sub-set of buffers to be used. The `-threshold` value can be used if “`-from <pins_or_nets>`” is not used. Only nets whose fanout exceeds this threshold are buffered.

Get information about existing buffer trees with `report_buffer_tree`. Report *cbt* options with `report_cbt_options`.

Build Skew-Optimized Buffer Trees

- If skew minimization is important for certain non-clock high fanout nets, use `compile_clock_tree`
- Must define the pins or nets as “ideal networks” prior to `place_opt`
- Additional commands available to exert more user control (see notes below)

```
set_ideal_network -no_propagate [get_nets Reset]
place_opt
compile_clock_tree -high_fanout_net [get_nets Reset]
```

3-67

The user can exert additional control by applying the `set_clock_tree_options` and `set_clock_tree_references` commands before `compile_clock_tree` (These commands are discussed in the CTS unit; Also see the *man* pages for more details).

The `compile_clock_tree` command ignores “ideal networks”, therefore the ideal network does not need to be removed prior to building the buffer tree.

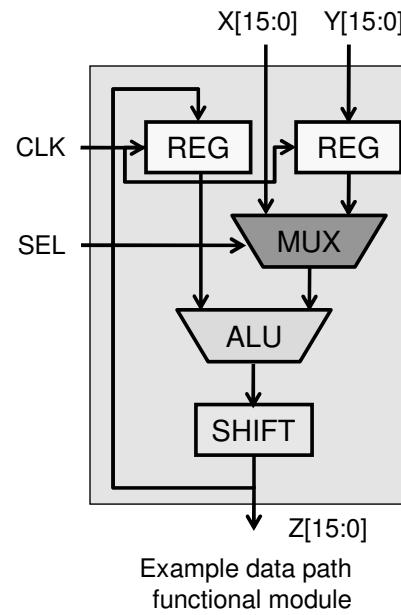
Relative Placement

 **Buffer trees
Relative placement**

3-68

What's Special about Data Path Logic?

- Bit-wise data operations are performed in parallel on each bit of a bus
 - Each operation corresponds to a dedicated function, e.g. *adder*, *multiplier*, *register*, *multiplexer*, etc
- Two groups of signal flows:
 - Data flow: *data-in* to *data-out*
 - Control flow:
 - ◆ Global: *clock*, *select*, *enable*
 - ◆ Local: *carry-in*, *carry-out*



3-69

The Ideal Layout for Data Path

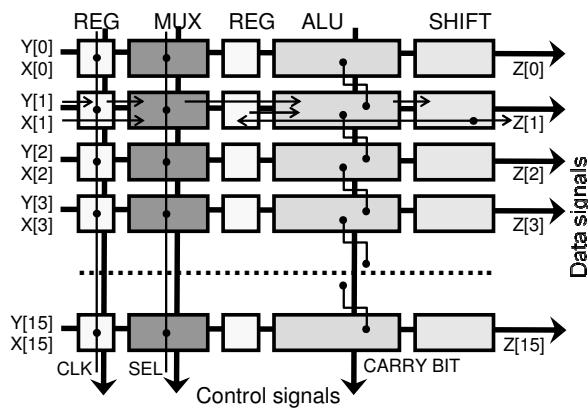
■ Best placed in a bit-sliced structure, for example:

- Cells operating on one bit are placed in one row, abutted next to each other horizontally
- This row is repeated and abutted vertically for each bit

■ Benefits of this placement structure:

- Overall placement area of data path cells is minimized
- Control and data signals can be routed vertically and horizontally, respectively

- ◆ Minimizes routing congestion
- ◆ Minimizes interconnect parasitic RCs
 - Reduced delay
 - Reduced power
- ◆ Minimizes clock and data skew

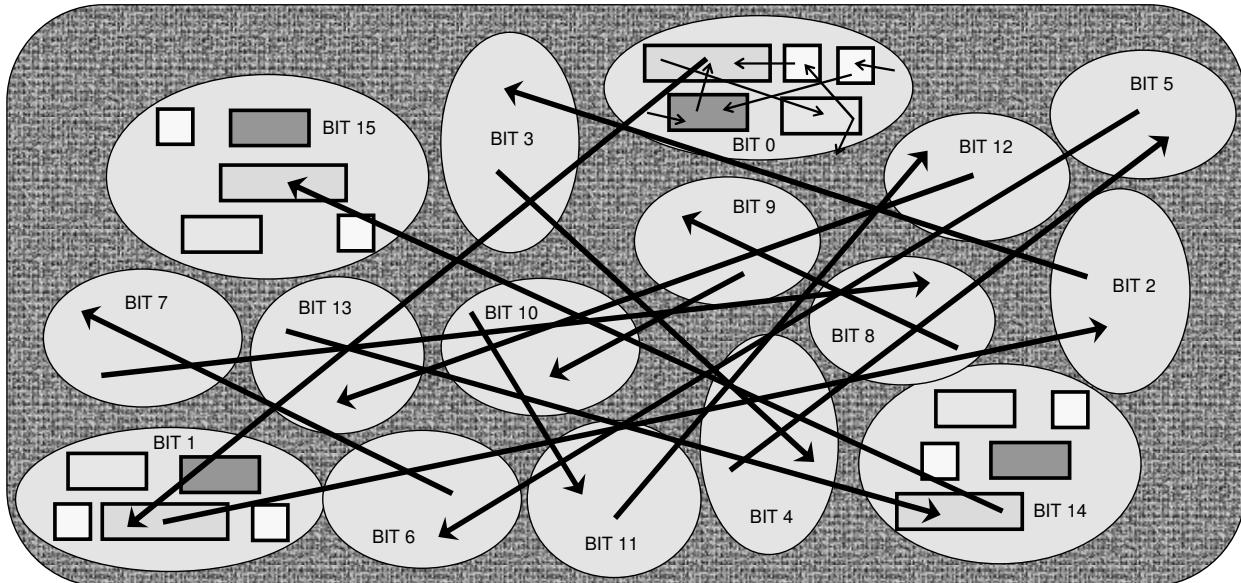


3-70

Data Path Layout using Traditional P&R Tool

Traditional layout tools can not take advantage of the regular bit-sliced structure, resulting in:

- Larger placement area and routing congestion
- Larger delays, skews and power consumption



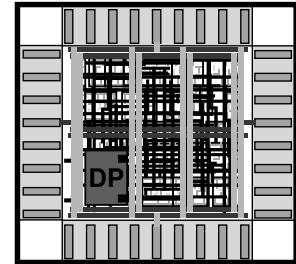
Example “traditional” layout with carry-signal connections highlighted

3-71

Traditional Solution: Custom/Manual Layout

- Traditionally one of these layout approaches is used:

- Creation of a full-custom data path macro or “hard IP”
- Manual placement of the data path cells in the standard cell core area



- Downsides of manual/custom approaches:

- Very time-consuming - Increased design cycle
 - ◆ Manual gate size selection
 - ◆ Manual placement of hard IP or data path cells in the core area
 - ◆ Manual/limited logic optimization for timing, power, area, DRCs
 - ◆ Re-creation of hard IP and manual re-placement after logic change
- Increased delay, power and area due to limitation of manual logic optimization

3-72

In the full-custom approach, a custom layout tool is used to design the layout of the data path logic. A “FRAM” view is generated for this hard IP which is then supplied to IC Compiler as part of a Milkyway IP reference library. The physical designer must then place this hard macro in the core placement area, along with any other macros, during the design planning phase. The placement of this hard macro is then fixed.

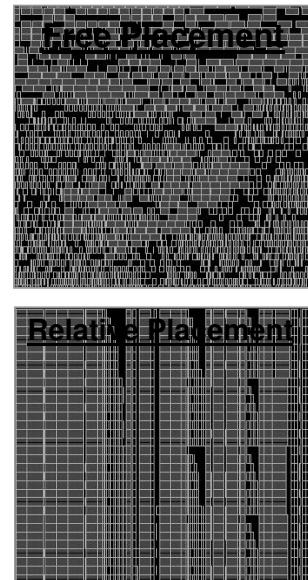
With manual placement the physical designer manually places each standard cell of the data path logic in the core placement area, and then “fixes” the placement. The remaining standard cells are placed around the fixed data path cells.

IC Compiler's Solution: *Relative Placement*

- Each data path functional module is defined once as a *relative placement (RP) group*
- Each cell of an *RP group* is assigned a row and column
 - Also known as “tiling”
 - Done prior to placement through TCL constraints

	col-0	col-1	col-2	col-3	col-4
row-3	U14	U15	U16	U17	U18
row-2	U10	U11		U12	U13
row-1	U5	U6	U7	U8	U9
row-0	U0	U1	U2	U3	U4

Example relative placement group
of a data path functional module



3-73

“Relative placement” may also be referred to as “physical datapath” in our literature and documentation.

Features and Benefits of *Relative Placement*

Features

- **RP groups are placed concurrently with standard cells**
 - Relative placement is maintained through post-route
- **RP logic is optimized as needed during placement through post-routing phases**
 - RP groups are re-sized to accommodate cell-sizing
 - Placement of RP groups in core is automatically adjusted
- **Easy to use GUI (Graphical User Interface) for visualization and editing**

Benefits

- **Improved congestion, skew, timing, power and area**
- **Increased design predictability**
- **Shorter design cycle**

3-74

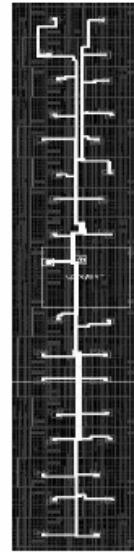
Candidates for *Relative Placement*

- The following functions are good candidates for taking advantage of *Relative Placement*

- Classic data path structured elements
- RAMs, FIFOs
- Clock structures
- Register banks

- Some key design applications include

- Processors, DSPs
- Graphics
- IP providers



Relative placement of RAMs with
clock “pin alignment”
→ Reduced clock skew and power

3-75

More Information on *Relative Placement*

- IC Compiler User Guide on SolvNet
- SNUG 2008
Structured Methods for Delay, Power Tuning and Variation: A Case Study
https://www.synopsys.com/news/pubs/snug/sanjose08/binney_pres.pdf
- SNUG 2007
Efficient Physical DataPath Specification: Streaming Relative Placement
https://www.synopsys.com/news/pubs/snug/sanjose07/dunham_pres.pdf
Design of a 1GHz DSP using IC Compiler
http://www.synopsys.com/news/pubs/snug/sanjose07/hill_pres.pdf
- Contact your local Synopsys Applications Consultant for
Relative Placement tutorial and labs

3-76

Summary

You should now be able to:

- Apply placement, DFT and power optimization settings before placement
- Perform placement and optimization
- Analyze congestion maps and reports
- Perform incremental congestion and timing optimization
- Perform additional placement techniques which allow more user-control



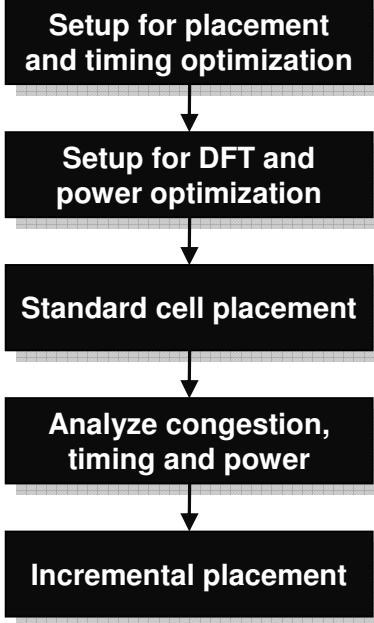
3-77

Lab 3: Placement



70 minutes

Perform standard cell placement and optimization with the goal of meeting the timing targets and eliminating congestion.



3-78

Agenda

**DAY
2**

3 Placement



4 Clock Tree Synthesis



Unit Objectives

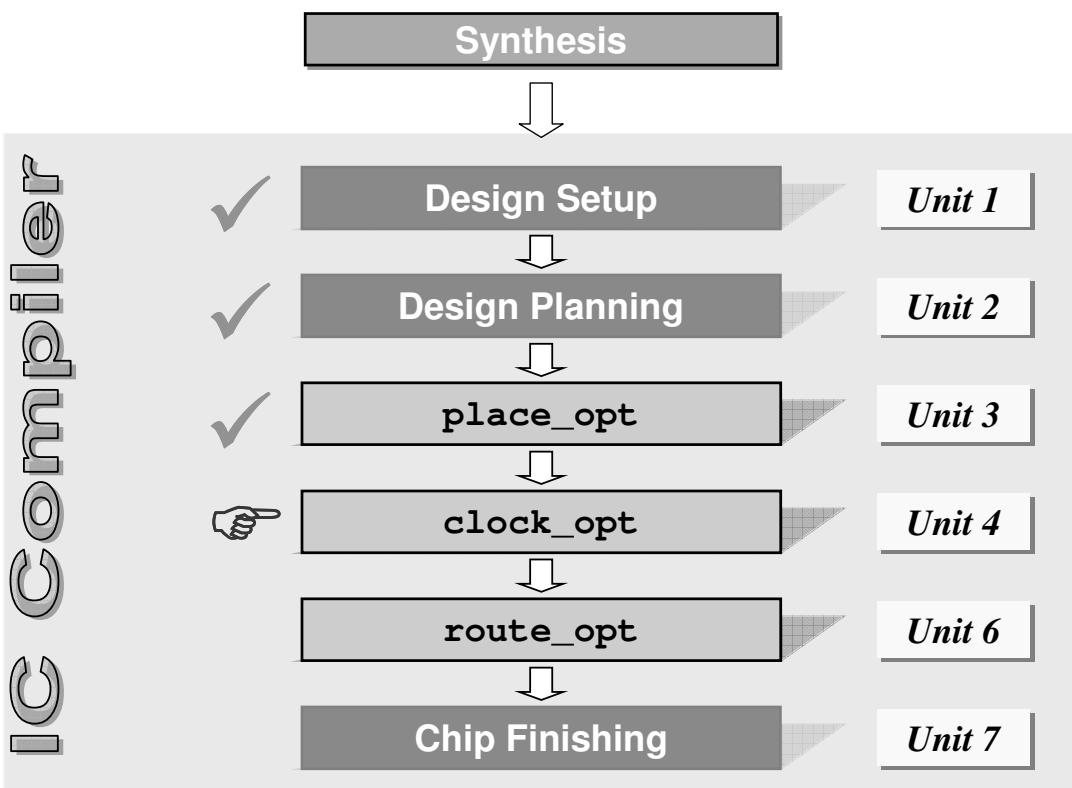


After completing this unit, you should be able to:

- List the status of the design prior to CTS
- Set up the design for clock tree synthesis
- Identify implicit clock tree start/end points and when explicit modifications are needed
- Control the constraints and targets used by CTS
- Execute the recommended clock tree synthesis and optimization flow
- Analyze timing and clock specifications post CTS

4-2

IC Compiler Flow



4-3

Design Status, Start of CTS Phase

- Placement - completed
- Power and ground nets – prerouted
- Estimated congestion – acceptable
- Estimated timing – acceptable (~0ns slack)
- Estimated max cap/transition – no violations
- High fanout nets:
 - Reset, Scan Enable synthesized with buffers
 - Clocks are still not buffered



Why are there no buffers on clock nets?

4- 4

Congestion, timing and max cap/transition are estimated based on virtual route.

Answer:

There are no clock buffers because synthesis designers are asked not to build the clock tree during synthesis.

CTS is the process of distributing clock signals to clock pins based on physical/layout information. HFN synthesis is used to balance the load but it is not good at balancing skew.

Is the Design Ready for CTS?

- **`check_physical_design -stage pre_clock_opt`**
checks for:

- Designs is placed
- Clocks have been defined
- Clock roots are not hierarchical pins (see below for support)

- **`check_clock_tree` checks and warns if:**

- A clock source pin is a hierarchical pin (see below for support)
- A generated-clock with improperly specified master-clock
- A clock tree has no synchronous pins
- There are multiple clocks per register

- **To enable support for clock source on hierarchical pins:**

```
set cts_enable_clock_at_hierarchical_pin true
```

4-5

`check_clock_tree` also checks and warns if:

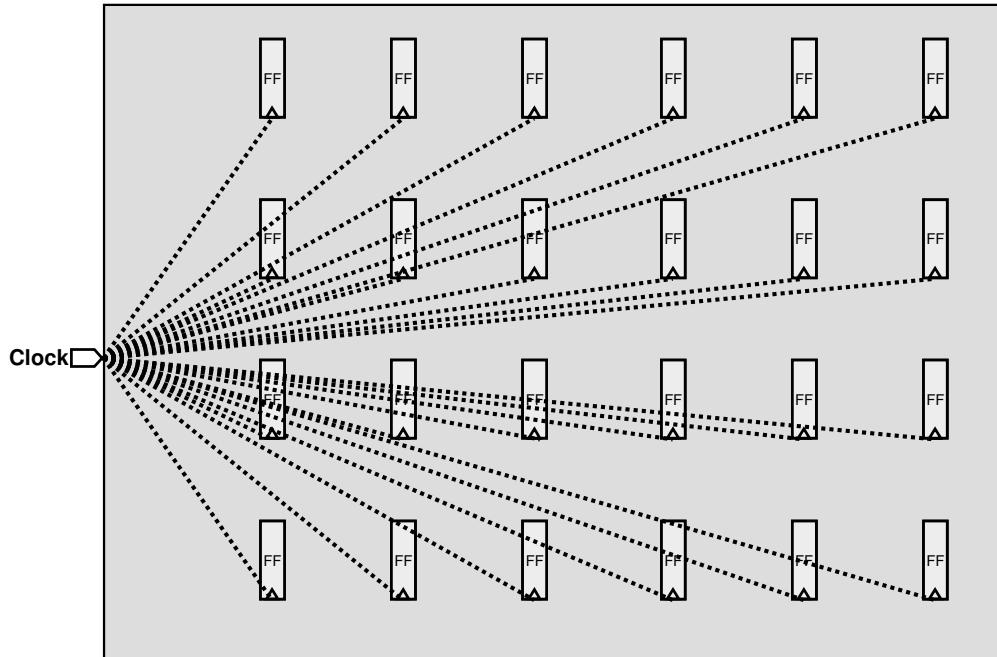
- * a master-clock does not propagate to a generated-clock source pin
- * a generated-clock with improperly specified master-clock
- * there is a master-clock that terminates at a pin without a corresponding generated-clock for a generated-clock
- * a clock loops to its source
- * there are cascaded clocks
- * a clock-tree path has ignored exceptions

Make sure clocks are properly setup prior to CTS is very important. This topic is covered in IC Compiler 2: CTS workshop.

Before using the command `create_clock` or `set_clock_tree_exceptions` to specify clock sources or clock tree exceptions on hierarchical pins:

`set cts_enable_clock_at_hierarchical_pin true`. The default setting of this variable is false.

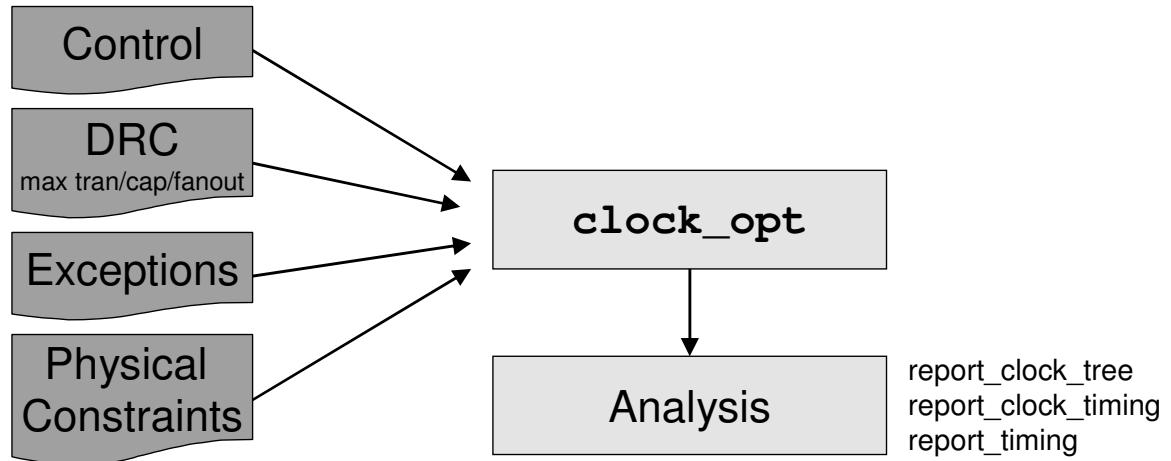
Starting Point before CTS



All clock pins are driven by a single clock source.

4-6

Clock Tree Synthesis



4- 7

CTS Goals

■ Meet the clock tree Design Rule Constraints (DRC):

- Maximum transition delay
- Maximum load capacitance
- Maximum fanout
- Maximum buffer levels

Constraints are upper bound goals. If constraints are not met, violations will be reported.

■ Meet the clock tree targets:

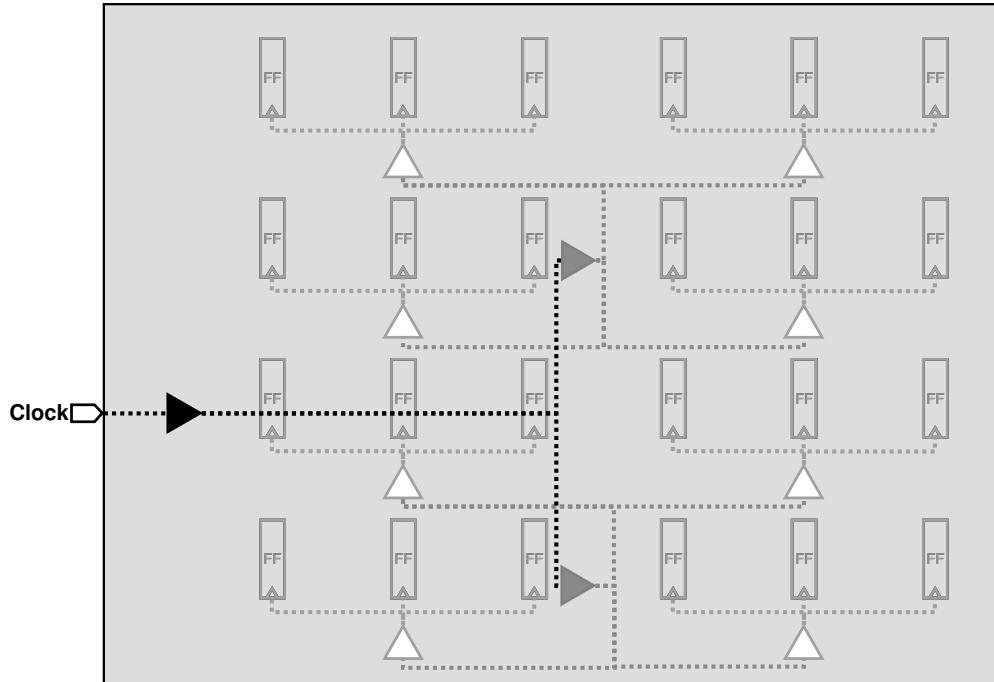
- Maximum skew
- Min/Max insertion delay

Targets are "nice to have" goals. If targets are not met, no violations will be reported.

4-8

DRC has the highest priority then next is skew then insertion delay

Clock Tree Synthesis (CTS) (1/2)



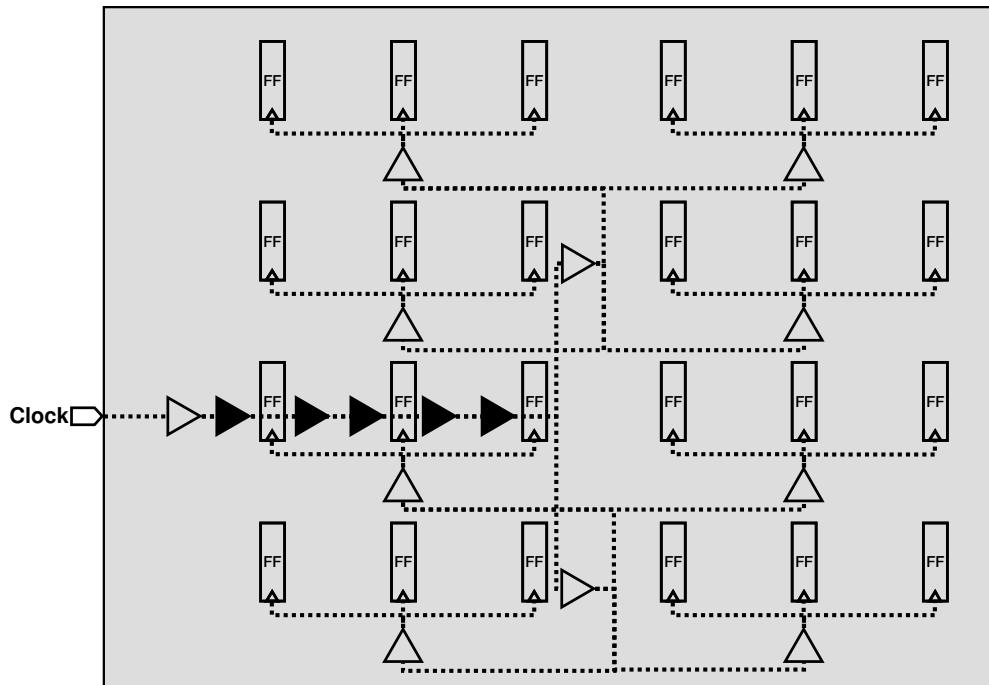
A buffer tree is built to balance the loads and minimize the skew.

4-9

This example shows a clock tree with three buffer levels between the clock source and the clock pins of FFs. Buffers are added to balance the loads (max tran/cap/fanout rules). The insertion delays from the clock source to the clock pin of all FFs are optimized to be as equal as possible in order to minimize the clock skew.

The clock tree is built from the leafs (registers) up.

Clock Tree Synthesis (CTS) (2/2)

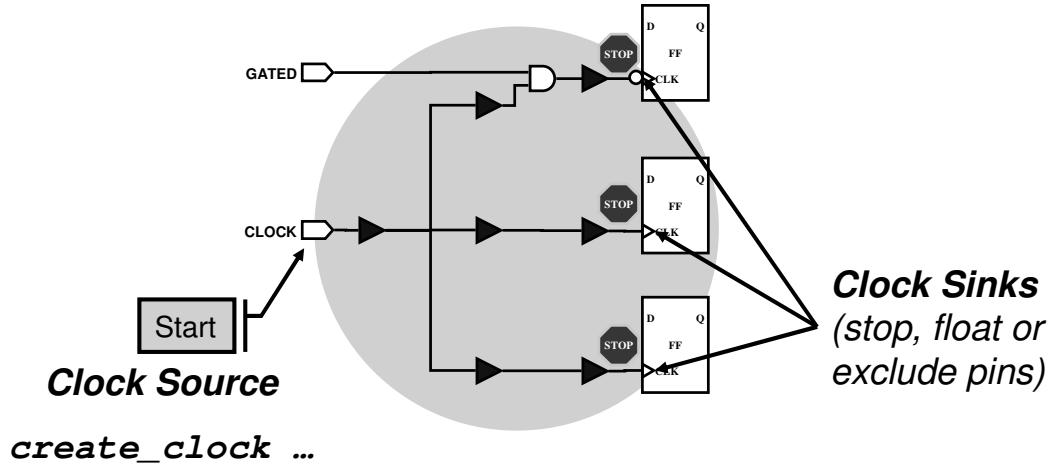


A “delay line” is added to meet the minimum insertion delay.

4-10

A “delay line” is typically made up of back to back buffers.

Where does the Clock Tree Begin and End?



`create_clock ...`

4-11

Clock tree begins at SDC-defined clock source:

`create_clock`

Clock tree ends at “sinks”. These are:

Stop pins

Float pins

Exclude pins (aka ignore pins)

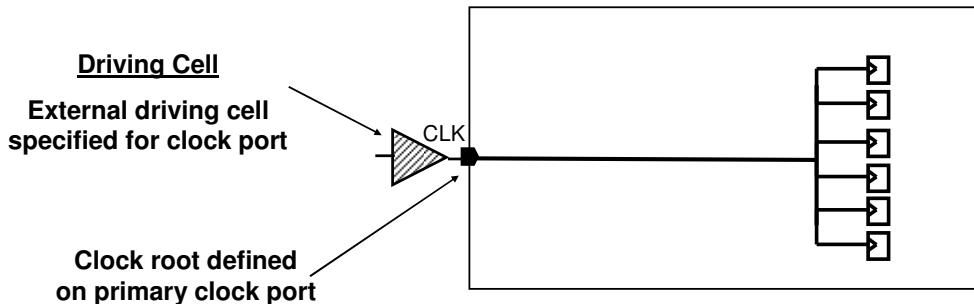
The command `report_clock_tree` is a good tool to analyze your clock tree structure *before* actually synthesizing it. The command can output:

- Overview for each clock
- A clock tree summary
- A list of Ignore Pins
- The overlap of clock domains
- And more...

Define Clock Root Attributes (1/2)

■ When the clock root is a primary port of a block

- Ensure that an appropriate driving cell is defined
set_driving_cell
- The synthesis constraints may include a weak driving cell for all inputs, including the clock port
- Because the clock is ideal during synthesis it has no effect on design QoR
- But **a weak driver on the clock port affects clock tree QoR during CTS**



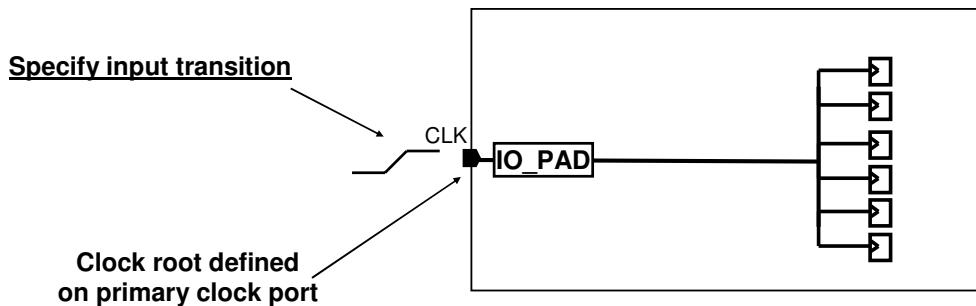
4-12

Example:

```
set_driving_cell -lib_cell buf4 -pin Z [get_ports CLK]
```

Define Clock Root Attributes (2/2)

- When the clock root is a primary port, but at the CHIP-level through an IO-PAD
 - Ensure that an appropriate input transition is defined
`set_input_transition`



4-13

Example:

```
set_input_transition -rise 0.3 [get_ports CLK]
set_input_transition -fall 0.2 [get_ports CLK]
```

Stop, Float and Exclude Pins

Exceptions

Implicit STOP or FLOAT pins

■ STOP Pins:

- CTS optimizes for DRC and clock tree targets (skew, insertion delay)

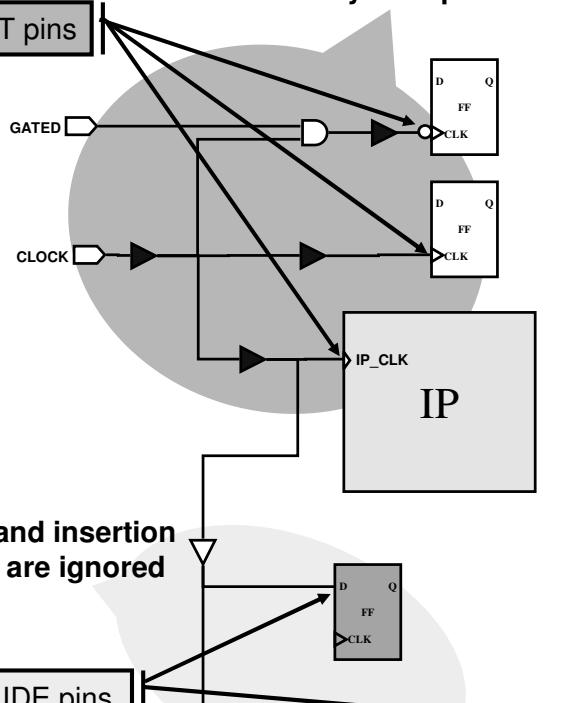
■ FLOAT Pins:

- Like Stop pins, but with delays on clock pin

■ EXCLUDE Pins:

- CTS ignores targets
- CTS fixes clock tree DRCs

skew and insertion delay are optimized



Implicit EXCLUDE pins

4-14

Implicit STOP (sync) pins are automatically defined by CTS on:

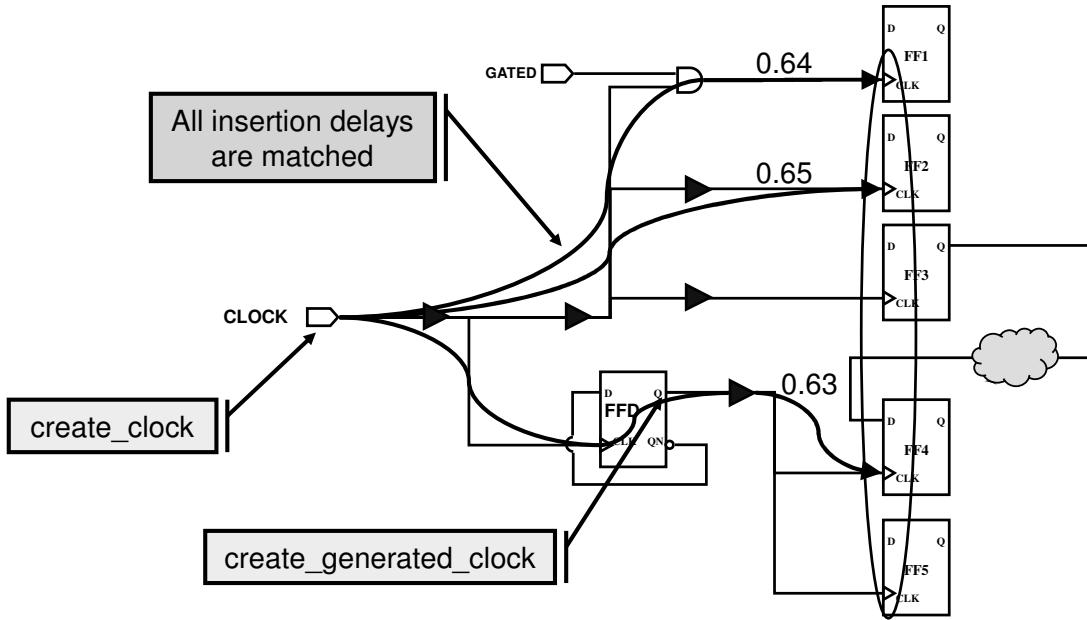
- Clock pins of sequential cells (FFs and latches)
- Clock pins of macros

Implicit EXCLUDE (ignore) pins are automatically defined by CTS on:

- Non-clock input pins of sequential cells (D, Set, Reset, etc)
- output ports
- Pins of combinational cells without any fanout or with disabled timing arcs
- Pins with 3-state enable arc
- Select/Control pin of mux used in the data path
- Input pin of pre-existing gate, if all pins in its fanout are *exclude pins*
- Incorrectly defined clock pins (missing or incorrect pin definition in the standard cell or macro cell FRAM view)
 - Clock pins without trigger edge info
 - Clock pins without a timing arc to the corresponding output pin

When CTS defines implicit EXCLUDE pins these are reported as warnings in the log file. Since an implicit ignore pin may imply a problem with either a reference library cell or design connectivity, it is important to verify that all implicit ignore pins are acceptable and intended.

Generated and Gated Clocks



Skew will be balanced ‘globally’, within each clock domain, across all clock-pins of both master and generated clock.

4-15

```
create_clock -p 10 [get_ports CLOCK]
create_generated_clock [get_pins {FFD/Q}] \
    -source [get_ports CLOCK] -name DIV_CLK -divide_by 2
```

The start point of a generated clock is traced back to the source of the `create_clock` command that the generated clock is derived from.

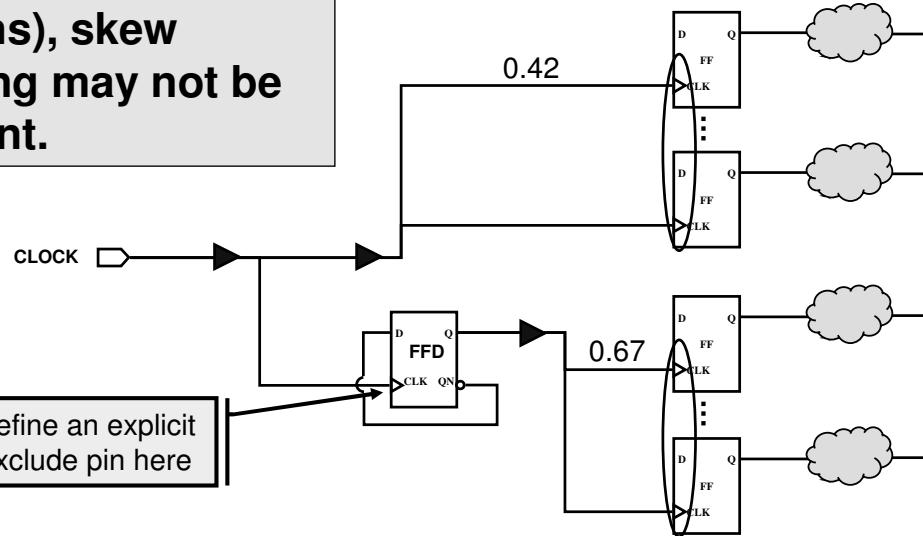
By default, CTS will not balance the skew of FFs that cross clock domains. Generated clocks are considered to be part of the same clock domain as their source clock, and are therefore synthesized together with the source clock. In case of generated clocks, a skew report of the master clock also considers and contains the sinks of the generated clock. Also, ICC generates a separate clock tree report for the generated clock also.

By default, CTS will balance the skew ‘globally’. This means that the skew of all sequential devices in the same clock domain, related or not, will be balanced. Two sequential devices are related if there is a data path connection between them.

Skew Balancing not Required?

If the divided clock domain is independent of the master domain (no paths), skew balancing may not be important.

Exceptions



```
set_clock_tree_exceptions -exclude_pins [get_pins FFD/CLK]
```

4-16

By defining an explicit exclude pin, the two “domains” – **CLOCK** and divided **CLOCK** – will no longer be optimized together for skew.

User-defined or Explicit Stop Pins

Scenario: If a macro cell clock pin is defined, CTS will treat that pin as an implicit stop pin. In this example the clock pin is not defined. What is the problem here?

Implicit exclude pin

skew and insertion delay are ignored

The macro's clock pin is marked as an implicit exclude pin – no skew optimization!

IP (FRAM)

4-17

Typically the clock port of a macro cell is defined as: a stop pin with correct clock tree delay information. CTS would use the internal clock tree delay to decide how much more it needs to add outside the macro cell during CTS to optimize for skew and insertion delay with the rest of the stop pins outside the macro cell.

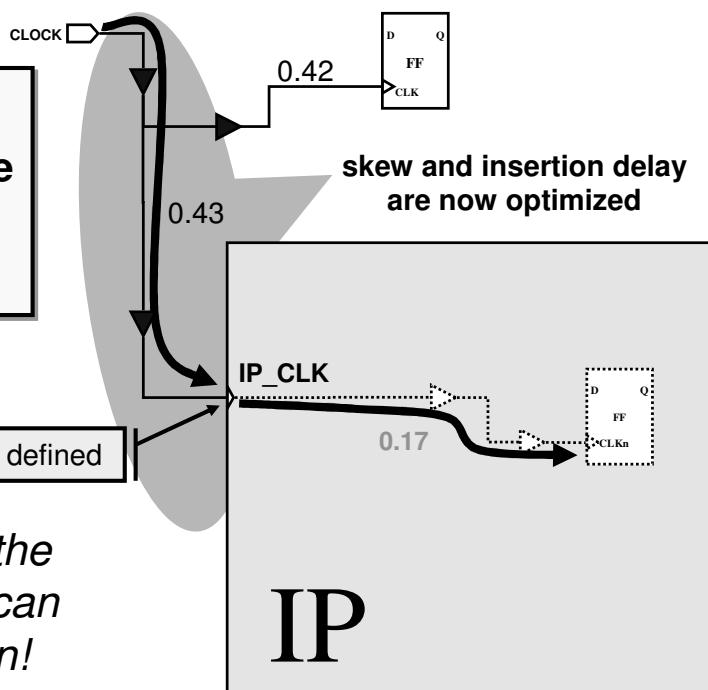
When a clock pin is not defined or defined incorrectly, CTS does not have enough information to optimize it for skew and insertion delay. This forces CTS to assign it an implicit exclude pin where it gets buffered by a single buffer.

Incorrectly defined clock pins (missing or incorrect pin definition in the standard cell or macro cell FRAM view):

- Clock pins without trigger edge info
- Clock pins without a timing arc to the corresponding output pin

Defining an Explicit Stop Pin

Defining an explicit stop pin allows CTS to optimize for skew and insertion delay targets.



CTS has no knowledge of the IP-internal clock delay – it can only “see” up to the stop pin!

```
set_clock_tree_exceptions -stop_pins [get_pins IP/IP_CLK]
```

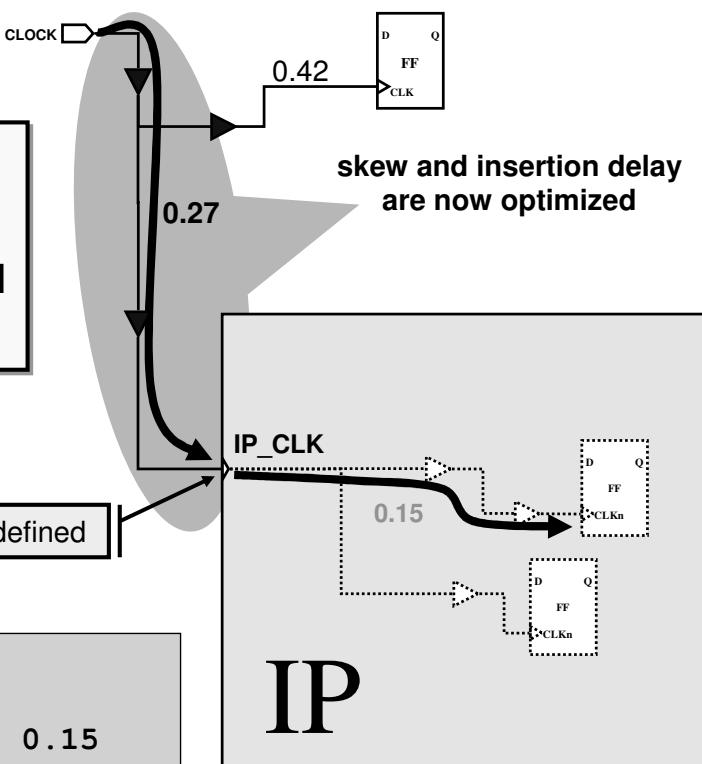
4-18

In the above example, CTS is doing the right thing – balancing the insertion delays up to all stop pins. CTS doesn't know that internal to the IP, there is a 0.17 delay to the actual clock pins of the flip flops.

Defining an Explicit Float Pin

Exceptions

Defining an explicit float pin allows CTS to adjust the insertion delays based on specification.



```
set_clock_tree_exceptions \
    -float_pins IP/IP_CLK \
    -float_pin_max_delay_rise 0.15
```

4-19

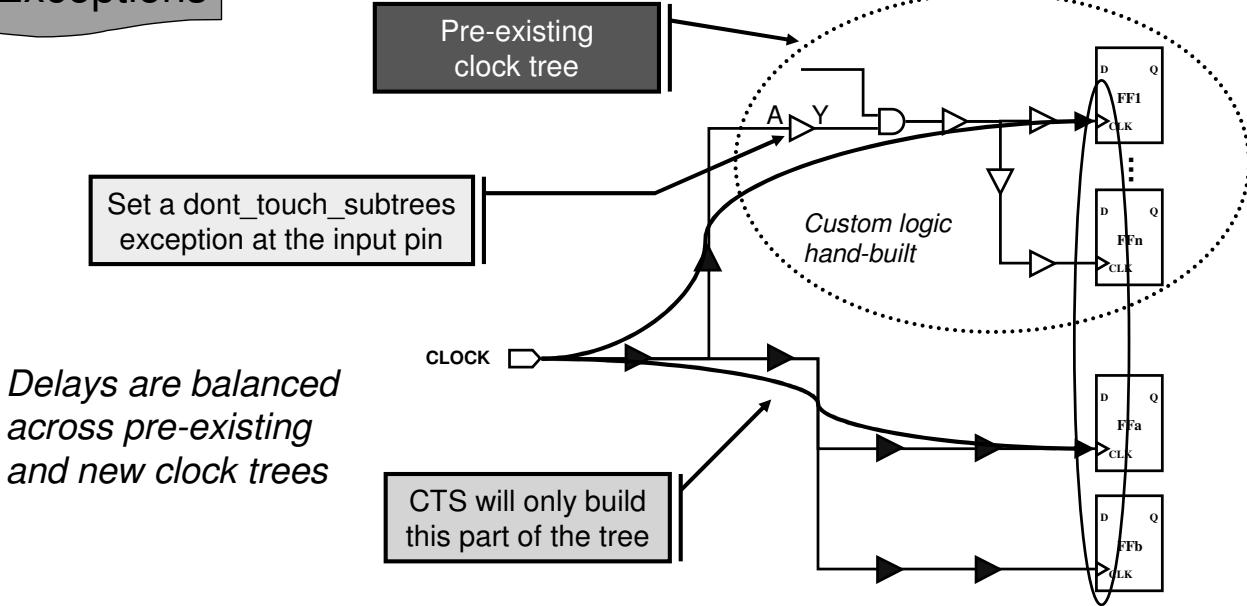
The complete list of float-pin related options is as follows:

```
set_clock_tree_exceptions \
    -float_pins
    -float_pin_max_delay_rise
    -float_pin_min_delay_rise
    -float_pin_max_delay_fall
    -float_pin_min_delay_fall
```

The max and min options are used to specify the delays under max (worst) and min (best) operating conditions, if min_max CTS is performed.

Preserving Pre-Existing Clock Trees

Exceptions



```
set_clock_tree_exceptions -dont_touch_subtrees buf/A
```

4-20

Total insertion delay includes the delay of the pre-existing clock tree.

If you don't preserve an existing clock tree, CTS will not rip-out the clock tree and rebuild a new one for you. Instead, it will use whatever is there to meet its goals and may add more logic. This may not always be desirable. Instead, you may want to delete the pre-existing clock tree entirely. To remove the clock tree instead of preserving it, use the command `remove_clock_tree`.

Impact of Preexisting Clock Cells

- Any preexisting clock buffers and cells are counted as clock gate levels
 - Any clock gate level is considered as a balancing point, therefore...
 - Preexisting clock buffers/inverters might create unnecessary clock levels for CTS
- Use `remove_clock_tree` to remove existing clock buffers
 - Will generally lead to higher quality clock trees

4-21

Test for Understanding

5 Minutes!



1. What are the two main goals of CTS?
 -
 -
2. What is the difference between **stop** and **exclude pins**?
List some examples of implicit stop/exclude pins.
3. How is a float pin different from a stop pin?

4-22

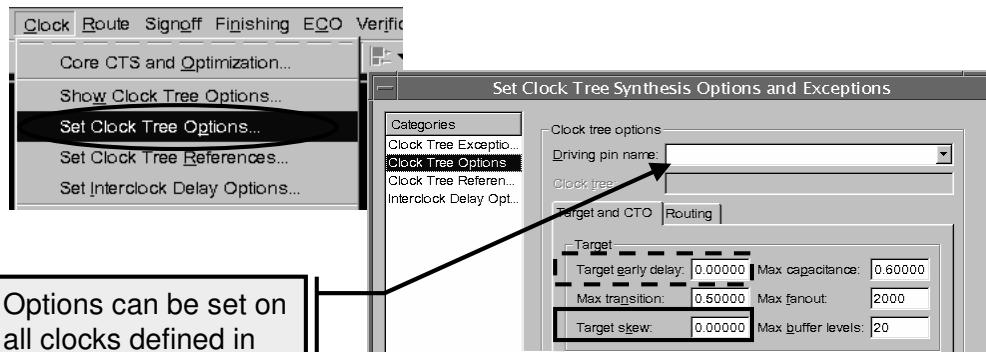
1. → Meet DRC (max tran/cap/fanout)
2. CTS stops at both. CTS optimizes for DRC for both, but only optimizes for clock tree targets
 - Meet the clock tree targets (max skew and min insertion delay)
 - to the stop pins.
3. A float pin allows the user to define when the clock edge is supposed to arrive at the pin. This can be used to give IP blocks or RAMs extra margin/setup time if needed.

Implicit stop pins: clock pins of FFs, latches and macro cells.
Implicit exclude pins: non-clock input pins of sequential cells (data, set, reset, etc); clock output ports; clock pins of sequential cells where the output is floating; incorrectly defined clock pins.

Implicit stop pins: clock pins of FFs, latches and macro cells.
Implicit exclude pins: non-clock input pins of sequential cells (data, set, reset, etc); clock output ports; clock pins of sequential cells where the output is floating; incorrectly defined clock pins.

Specifying Skew / Insertion Delay Targets

Control



```
icc_shell> set_clock_tree_options \
           -target_early_delay 0.9 \
           -target_skew 0.1
```

Setting global CTS options...

1

4-23

Minimum insertion delay (target early delay) is done by adding a chain of buffer as needed if the insertion of the clock tree is less than the target specified.

The Max buffer levels constraint is not supported!

Clock by Clock Settings



How do you set different targets per clock?

- Use the GUI, select the appropriate clock from the pull-down, and OK your selections for every clock
- Better: Use the shell.

```
set_clock_tree_options \
    -clock_tree clk1 -target_early_delay 0.9
set_clock_tree_options \
    -clock_tree clk2 -target_skew 0.2
```

TIP: Use the GUI preview mode, then cut&paste
the echo'd commands into your script.

4-24

```
set_clock_tree_options
    [-clock_tree <pin_name>]      (root of the clock tree)
    [-layer_list <list>]        (layers enabled for clock tree routing)
    [-target_skew ]            (skew constraint)
    [-target_early_delay ]     (minimum insertion delay)
    [-routing_rule ]          (name of a non-default routing rule to
use for clock tree nets)
    [-buffer_sizing boolean-string]
        (whether buffers in one level may have different sizes)
    [-buffer_relocation boolean-string]
        (allow to relocate buffers during optimization)
    [-gate_sizing boolean-string]
        (allow to size gates during optimization)
    [-gate_relocation boolean-string]
        (allow to relocate gates during optimization)
    [-delay_insertion boolean-string]
        (allow to insert delay during optimization)
```

Set Buffer/Inverter Selection Lists

- To limit CTS to a list of buffers/inverters used for specific optimizations:

```
set_clock_tree_references  
    -references list1  
    -references list2 -sizing_only  
    -references list3 -delay_insertion_only
```

It is recommended to define all lists

- There is no priority on how CTS uses the members from each list
- If a list is not specified, all buffers/inverters in the library without dont_use attributes are used



Make sure the references are in target_library

4-25

The set_clock_tree_references command allows you to select specific library cells (buffers, inverters, delay cells) for clock tree synthesis. In general, it is recommended to allow CTS to choose the best buffers from the library automatically, but in some cases you may need to tell CTS which ones to use! The following options exist:

```
[-clock_trees <list>]      (list of clock names or clock root pins or ports)  
-references <list>          (list of library cells to be used for buffering)  
[-sizing_only]              (use the specified reference for sizing optimization only)  
[-delay_insertion_only]     (use the specified reference for delay insertion optimization only)
```

To remove all reference settings, use the command reset_clock_tree_references.

It is highly recommended to select good cells for CTS from your library:

```
set_clock_tree_references -references "BEST_PRACTICE_buffers_for_cts"  
set_clock_tree_references -sizing_only  
    -references "BEST_PRACTICE_buffers_for_CTS_CTO_sizing"  
set_clock_tree_references -delay_insertion_only  
    -references "BEST_PRACTICE_cels_for_CTS_CTO_delay_insertion"
```

It's recommended to define each list for each optimization. If a list is not defined, unspecified cells from target libraries maybe used unexpectedly.

When Clock Tree DRCs are Used

DRC

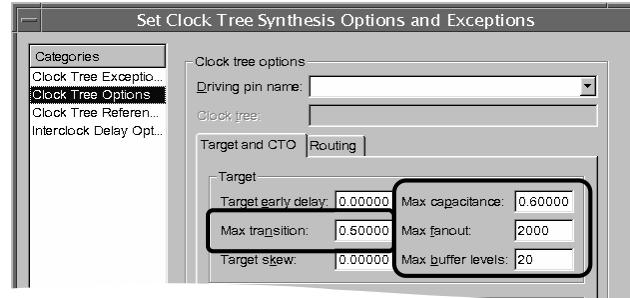
- **Max tran/cap/fanout/buf:**

- Smallest value from library, SDC file and/or GUI is used
- If (GUI values < SDC or library)
→ Verify with vendor/designer and relax GUI values

Where clock tree DRC is set

- **If settings are too tight, violations may not be fixed**

- **For DRC, what the GUI calls “target” is actually a “constraint” as defined earlier!**



4-26

The SDC DRC constraints are obeyed by CTS:

```
set_max_fanout  
set_max_capacitance  
set_max_transition
```

These constraints can be applied to design ports or as a global setting. To set the constraint as a global setting, use for example “`set_max_fanout 2 [current_design]`”.

Override

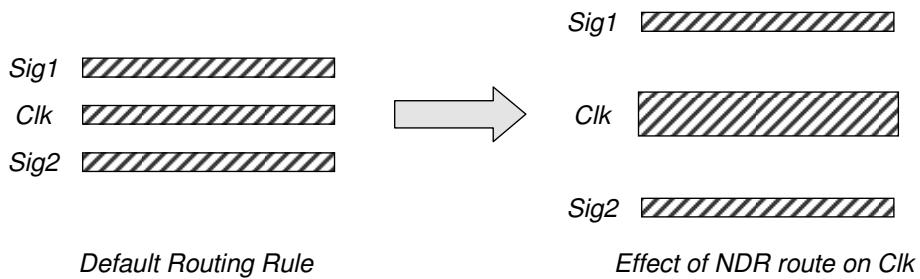
If you want to *force* the tool to use CTS-specific design rule constraints using `set_clock_tree_options`, even though they are relaxed compared to the library or SDC, turn ON the following variable (default is false):

```
set cts_force_user_constraints true
```

Non-Default Clock Routing

Physical Constraints

- IC Compiler can route the clocks using non-default routing rules, e.g. double-spacing, double-width, shielding
- Non-default rules are often used to “harden” the clock, e.g. to make the clock routes less sensitive to Cross Talk or EM effects



4-27

Specifying Non-Default Rules

■ First, define the NDR:

```
define_routing_rule my_route_rule \
    -widths {METAL3 0.4 METAL4 0.6 METAL5 0.6} \
    -spacings {METAL3 0.5 METAL4 0.65 METAL5 0.65}
```

■ Then, configure CTS:

```
set_clock_tree_options -root clk \
    -routing_rule my_route_rule \
    -layer_list "METAL3 METAL4 METAL5"
```

You may also specify
the layers to use for
clock tree routing.

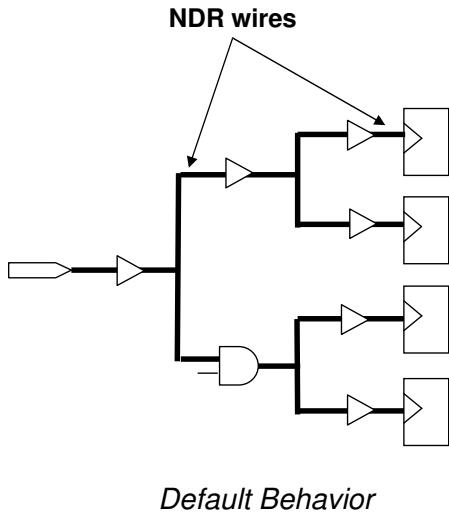
4-28

You can report on the routing rules that were defined using `report_routing_rule`. If, in the above example, layers are used for which a non-default routing rule was not defined, these layers will use the default routing rules, as defined by the tech-file.

```
define_routing_rule rule_name
    -reference_rule_name rule_name
        | -default_reference_rule
    [-widths {routing_layer_name value} ]
    [-snap_to_track]
    [-spacings {routing_layer_name value} ]
    [-shield_widths {routing_layer_name value} ]
    [-shield_spacings {routing_layer_name value} ]
    [-via_cuts {default_via_name value} ]
    [-taper_level tapering_level]
```

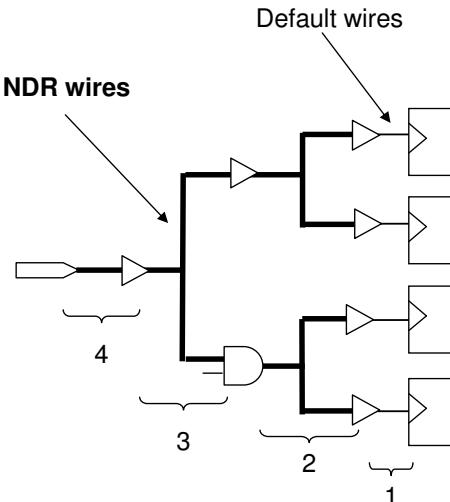
Nondefault Rule Options

```
set_clock_tree_options \
    -routing_rule my_route_rule
```



Default Behavior

```
set_clock_tree_options \
    -routing_rule my_route_rule \
    -use_default_routing_for_sinks 1
```



**use_default_routing_for_sinks
can only be used globally!**

4-29

Since the pin density at the flipflop is higher, it may be better to use default routing when making the final connection. This can prevent many Routing DRC violations that may take longer to fix. The `use_default_routing_for_sinks` option expects a positive integer, which stands for the number of levels to use default routing for. In the example shown above, “1” means use default routing for ONE stage, the final connection.

This option can only be used globally, meaning all clocks at once.

NDR Recommendations

- **Always route clock on metal 3 and above**

- **Avoid NDR on clock sinks:**

```
set_clock_tree_options -use_default_routing_for_sinks 1
```

- **Avoid NDR on Metal 1**

- may have trouble accessing metal 1 pins
on buffers and gates

- **Put NDR on pitch – try to avoid blind double spacing**

- Preserve routing resources/keep preroute RC estimation
accurate

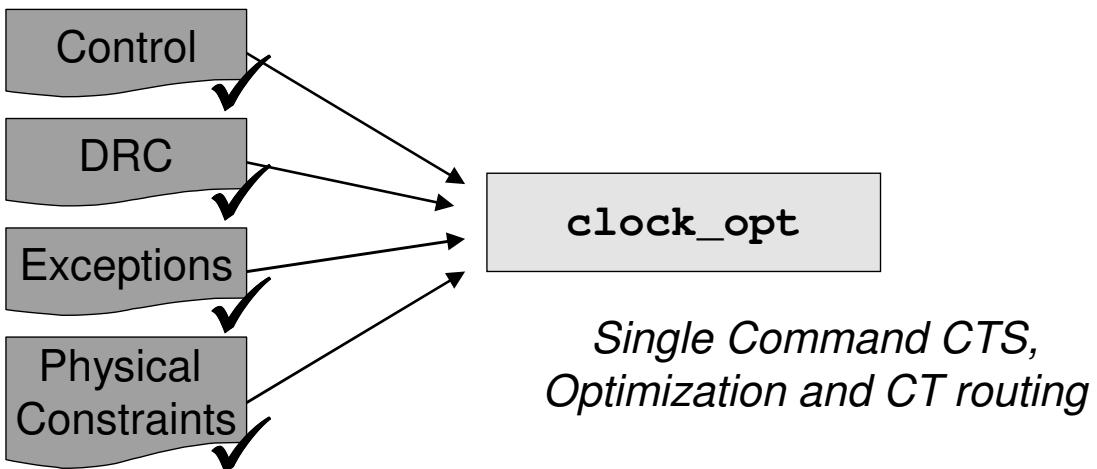
- **Consider double width to reduce resistance**

- **Consider double via to reduce resistance and
improve yield**

4-30

A word on double-spacing: The minimum spacing indicated in the tech file is generally not pitch - width. So if you want double-spaced wires for clocks, you need to use the double pitch (along with width) as your indicator for finding a good spacing value. Wires will end up on pitch – so this improves correlation with what you have before detail routing.

Invoke CTS: Core Command

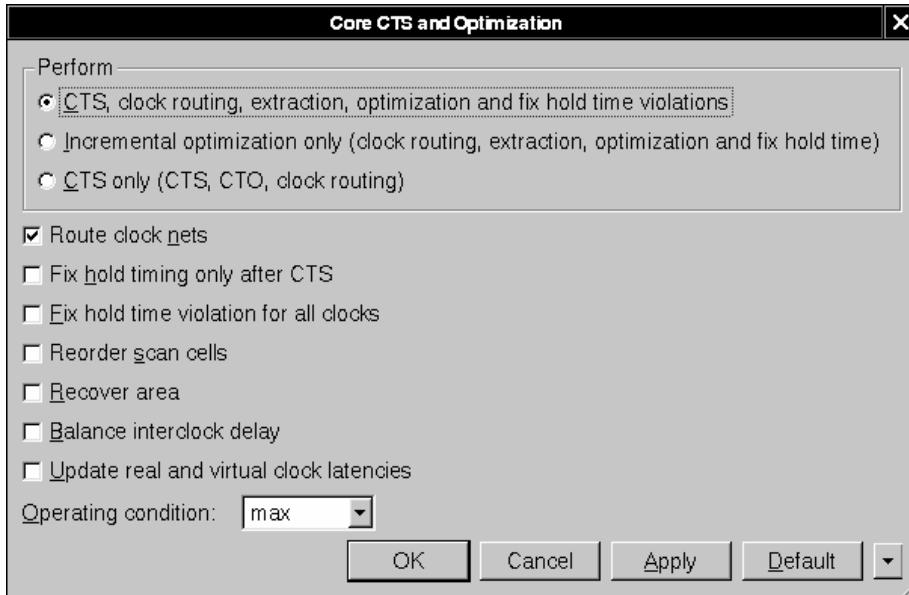


Options:

- `operating_condition`
- `inter_clock_balance`
- `optimize_dft`
(more later)

4-31

Menu: CTS → Core CTS and Optimization



clock_opt use recommendation

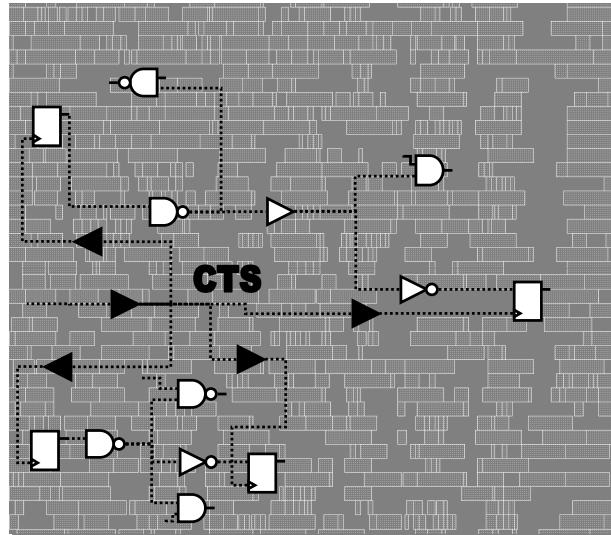
- Using `clock_opt` in the following manner has been found to be more flexible across designs and flows:

```
clock_opt -only_cts -no_clock_route
analyze...
clock_opt -only_psyn -no_clock_route
analyze...
route_group -all_clock_nets
```

4-32

Effects of Clock Tree Synthesis

- Clock buffers added
- Congestion may increase
- Non clock cells may have been moved to less ideal locations
- Can introduce new timing and max tran/cap violations



How do you handle new violations?

4-33

Congestion pattern should be similar to the one before CTS. It is, however, generally higher due to the extra clock buffers added during CTS.

If the optimal location for the CTS buffer overlaps with cells placed previously then the previously placed cells need to be moved to open up space for the clock tree buffers.

Incremental Placement / Optimization

■ **clock_opt -only_psyn**

- Reduces disturbances to other cells as much as possible
- Performs logical and placement optimizations to fix possible timing and max tran/cap violations, based on propagated clock arrivals

■ **To enable hold time fixing**

```
set_fix_hold [all_clocks]
clock_opt -only_psyn
```

- To prioritize TNS over WNS, set:

```
set_fix_hold_options -prioritize_tns
```

- To prioritize min over max, set:

```
set_fix_hold_options -prioritize_min
```

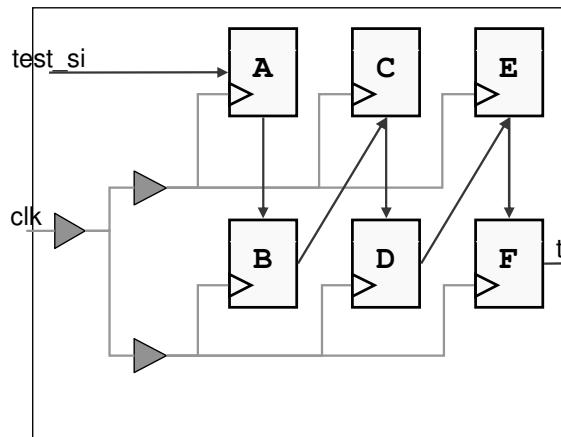
4-34

The alternative to using fix hold attribute is to use `-fix_hold_all_clocks` option. However, it's recommended to use `set_fix_hold` so the fix hold attribute is saved to the Milkyway database.

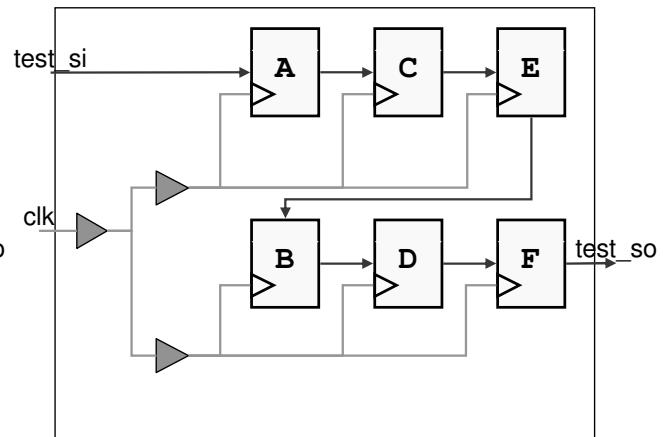
Minimize Hold Time Violations in Scan Paths

```
clock_opt -only_psyn -optimize_dft
```

- Reorders to minimize crossings between clock buffers
- Can reduce unnecessary hold time violations in the scan chain



Without clock tree based reordering



With clock tree based reordering

4-35

Recommended Flow

```
place_opt  
# timing ok, congestion ok!  
  
set_clock_tree_options ...  
set_clock_tree_exceptions ...  
set_clock_tree_references ...  
  
remove_clock_uncertainty [all_clocks]  
# OR: adjust uncertainty to contain only jitter, not skew component  
clock_opt -only_cts -no_clock_route  
  
# Analyze  
set_fix_hold [all_clocks]  
clock_opt -only_psyn -optimize_dft -no_clock_route
```



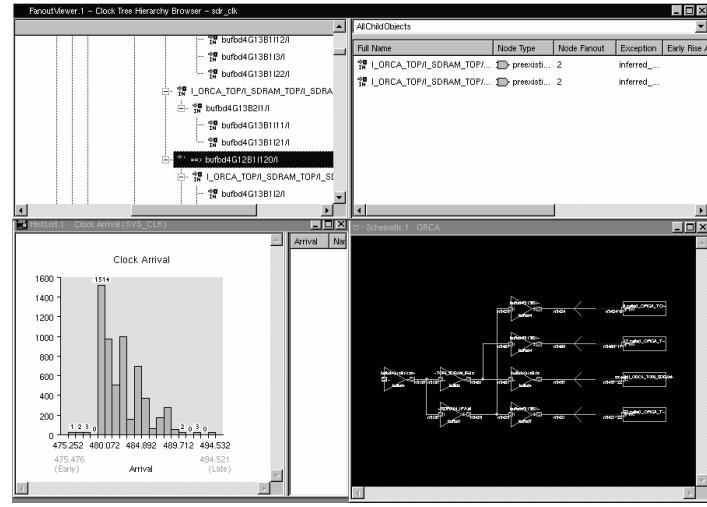
All CTS-built clocks are propagated automatically –
no need to use the “`set_propagated_clock`” command!

4-36

Analysis using the CTS GUI

■ CTS browser

- Properties and attributes on clock tree objects
- Traversing clock tree levels
- Symbols for CTS objects like buffers, gates and sinks



■ CTS schematic

- Trace forward/backward in schematic view
- Collapses all sinks in the fanout of a CTS buffer for clearer CTS schematic
- Highlight CTS objects in the layout view

■ Clock arrival histogram

4-37

CTS → New Clock Tree Synthesis Window

Analyzing CTS Results

- **report_clock_tree**

- summary**

- settings**

- ...**

- Reports Max global skew, Late/Early insertion delay, Number of levels in clock tree, Number of clock tree references (Buffers), Clock DRC violations

- **report_clock_timing**

- Reports actual, relevant skew, latency, interclock latency etc. for paths that are related.
 - Example: **report_clock_timing -type skew**

Details covered in the lab!

4-38

What about CTS Operating Conditions?

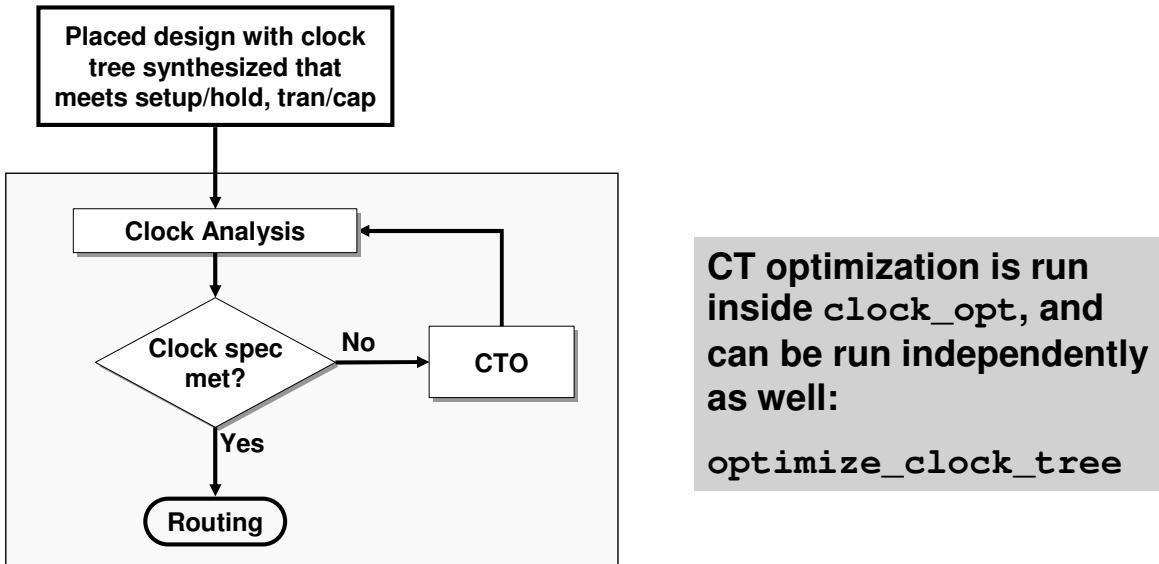
```
clock_opt -operating_condition min_max
```

- **What happens when building the CT using min_max?**
 - The tree is compiled in –max then analyzed in –min
 - ◆ If the skew analyzed in –min is not worse than the skew in –max, compiling with –min_max will not make much difference
 - ◆ If the skew analyzed in –min is worse than that in –max, then compile in –min_max will build a tree with a better skew in –min at the cost of a possibly worst skew in –max
- **In summary, a tree compiled in –min_max will build a tree with less skew variation when analyzed in both –min and –max**
- **The skew will not be better than a tree compiled and analyzed in –max**

4-39

Clock Tree Optimization

Perform additional Clock Tree Optimization as necessary to further improve clock skew.

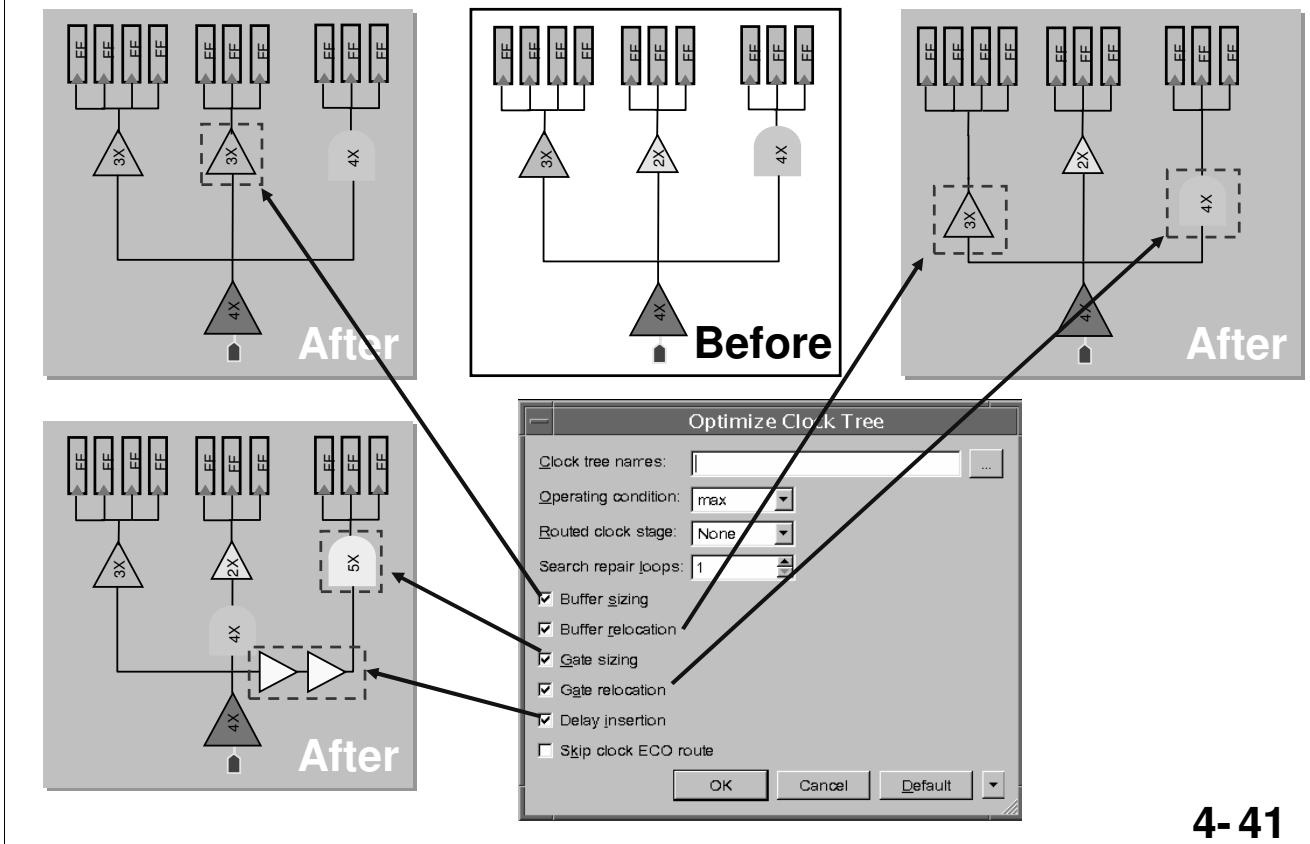


4-40

Options set on the clock options dialog for `clock_opt` do not apply to the stand-alone command `optimize_clock_tree`! This command uses its own options as shown below.

```
optimize_clock_tree
  -delay_insertion      [default = on]
  -buffer_sizing        [default = on]
  -buffer_relocation    [default = on]
  -gate_sizing          [default = off]
  -gate_relocation       [default = on]
  -operating_condition min|max (default)|min_max
```

(Embedded) Clock Tree Optimization



Gate/Buffer Sizing

Sizes up or down buffers/gates to improve both skew and insertion delay.

Will not modify the clock tree structure during this optimization process.

Buffer Relocation

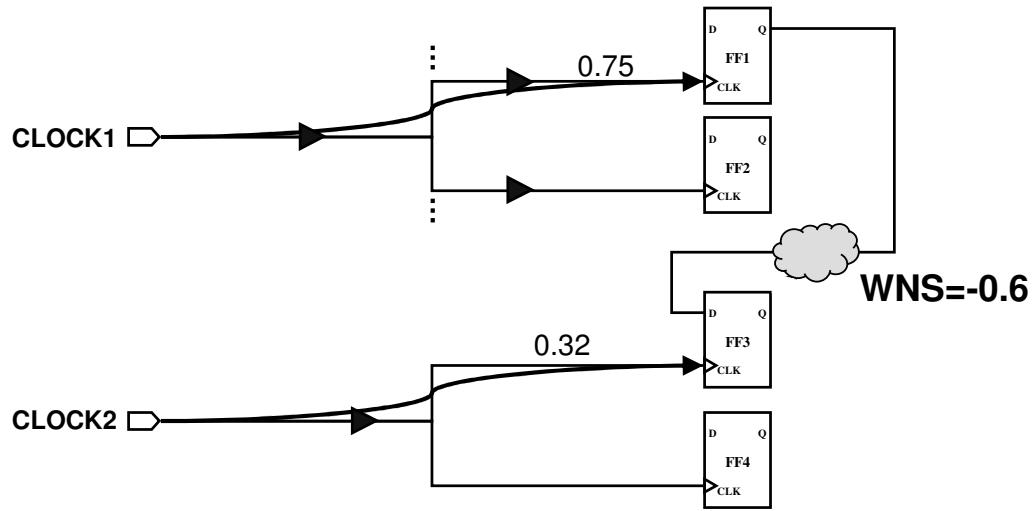
Physical location of the buffer is moved to reduce skew and insertion delay.

Will not add or remove hierarchy during optimization process.

Delay Insertion

Delay is inserted for shortest paths.

Balancing Multiple Synchronous Clocks

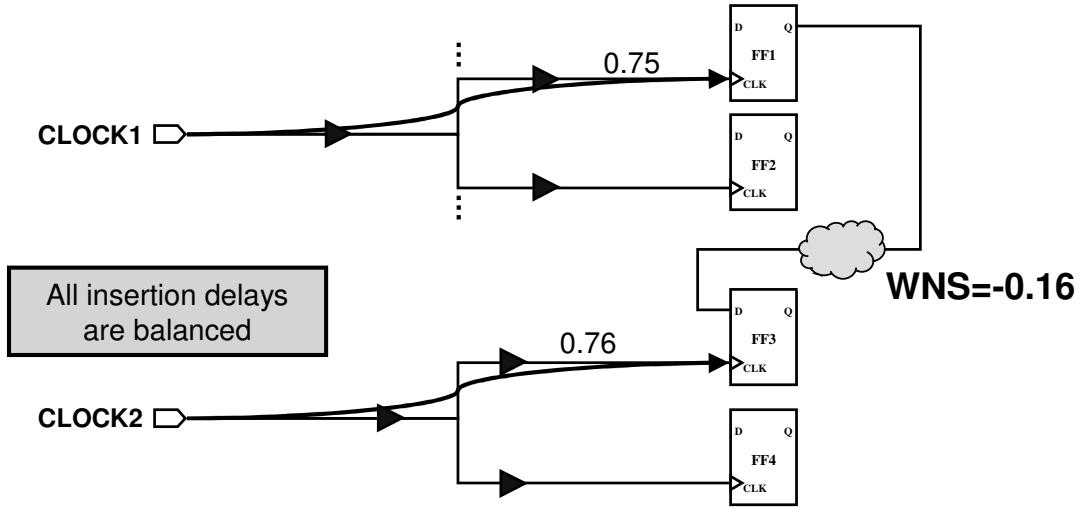


As shown here, the path from FF1 to FF3 will have an additional setup penalty of $0.75 - 0.32 = 0.43$

4-42

Inter-Clock Delay Balancing

```
set_inter_clock_delay_options \
    -balance_group "CLOCK1 CLOCK2"
clock_opt -inter_clock_balance
Performs CTS and balances the clocks.
```



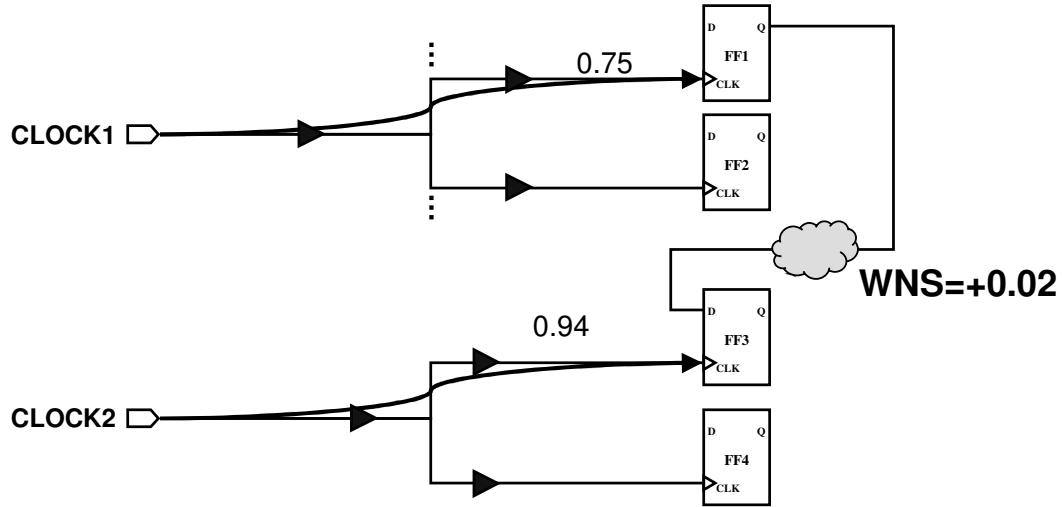
4-43

Clock tree balancing can also be performed stand-alone after CTS using the command:

```
balance_inter_clock_delay -clock_trees "CLOCK1 CLOCK2"
```

Inter-Clock Delay Balancing with Offset

```
set_inter_clock_delay_options \
    -offset_from CLOCK1 -offset_to CLOCK2 \
    -delay_offset 0.2
clock_opt -inter_clock_balance
# or stand-alone after CTS:
# balance_inter_clock_delay -clock_trees "CLOCK1 CLOCK2"
```



4-44

The stand-alone after CTS equivalent commands are:

```
set_inter_clock_delay_options -offset_from CLOCK1 \
    -offset_to CLOCK2 -delay_offset 0.2
balance_inter_clock_delay -clock_trees "CLOCK1 CLOCK2"
```

Clock trees can be balanced using an offset requirement, as shown above, or using targets, using the syntax below or through SDC set_clock_latency commands.

The following command performs inter clock delay balancing using the maximum target delay options:

```
balance_inter_clock_delay -max_target_delay 0.5
```

The following command performs inter clock delay balance for the CLK1 clock tree to 0.5ns and CLK2 to 0.8ns:

```
set_inter_clock_delay_options -target_delay_clock CLK1 \
    -target_delay_value 0.5
set_inter_clock_delay_options -target_delay_clock CLK2 \
    -target_delay_value 0.8
balance_inter_clock_delay
```

To honor the latencies defined in the SDC file, use

```
set_inter_clock_delay_options -honor_sdc true
```

SDC Latencies



Does CTS respect SDC set_clock_latency?

- CTS does not respect SDC latencies by default!
- If you need your insertion delays to match the SDC provided latencies, perform clock tree balancing

```
set_inter_clock_delay_options -honor_sdc true  
clock_opt -inter_clock_balance
```

- Note: Insertion delay will not be minimized if given SDC latency is less than initial CTS insertion delay

4- 45

Core vs. Atomic Commands

- IC Compiler's `clock_opt` is very powerful and tries to give the best Out-of-Box clock tree results. Finer control over `clock_opt` for specific tweaking is not always possible
 - E.g., you can only compile all clocks at once with `clock_opt`
- For finer control, you can execute a series of “atomic” commands
- *Please refer to the documentation for details on the atomic commands – Example follows...*

4- 46

Flow Using Atomic Commands

```
place_opt
# timing ok, congestion ok!
set_clock_tree_options ...
set_clock_tree_exceptions ...
set_clock_tree_references ...
define_routing_rule my_route_rule ...
set_clock_tree_options \
    -routing_rule my_route_rule \
    -use_default_routing_for_sinks 1
compile_clock_tree -clock_trees "CLOCK2"
compile_clock_tree -clock_trees "CLOCK1"
optimize_clock_tree
remove_clock_uncertainty [all_clocks]
# OR: adjust uncertainty to contain only jitter, not skew component
set_fix_hold [all_clocks]
psynopt -area_recovery
optimize_clock_tree
route_group -all_clock_nets
```

4-47

Test for Understanding

5 Minutes!



1. CTS tries to:
 - a) Minimize skew only
 - b) Minimize skew and insertion delay
 - c) Minimize skew and maximize insertion delay
 - d) Minimize skew and meet minimum insertion delay target
 2. How do you set a clock skew target of 0.1 for clk1, and a minimum insertion delay of 0.7 for clk2? What is the skew target for clk2?
 3. Write the command(s) to balance the two clock trees clk1 and clk2, so clk2 arrives 0.3 earlier!
 4. Why is it important to remove or adjust the clock uncertainty *before* executing `clock_opt`?

4- 48

1. d) Minimize skew and meet minimum insertion delay target
2. set_Clock_tree_Options -root CLK1 -target_skew 0.1
set_Clock_tree_Options -root CLK2 \-target_early_delay 0.7
3. set_initter_clock_delay_Options -offset_from CLK2 \-offset_to CLK1 -delay_offset 0.3
clock_opt -initter_clock_balance
OR after CTS: balance_initter_clock_delay -clock_trees "CLK1
CLK2"
4. Uncertainty should be removed before clock_opt in case post-cts optimizations are done as well. Otherwise, the optimizations will take a very pessimistic skew into account (actual clock well.

Unit Objectives Summary

You should now be able to:

- List the status of the design prior to CTS
- Set up the design for clock tree synthesis
- Identify implicit clock tree start/end points and when explicit modifications are needed
- Control the constraints and targets used by CTS
- Execute the recommended clock tree synthesis and optimization flow
- Analyze timing and clock specifications post CTS



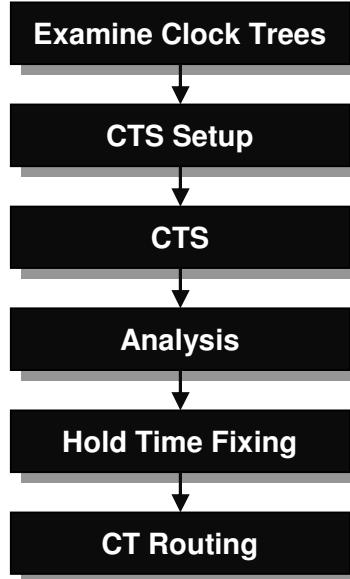
4- 49

Lab 4: Clock Tree Synthesis



75 minutes

**Perform CTS and
Optimizations on the
ORCA Design.**

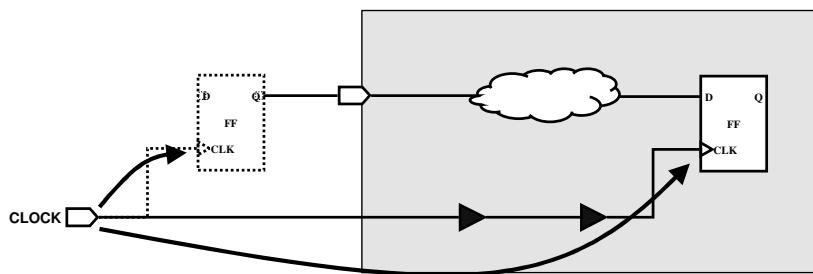


4- 50

Appendix A

Automatic IO Latency Calculation

IO Latency Auto Update



- Once clock trees are built, delays are propagated:
 - Latency of the “real” clock path is calculated (lower arrow in diagram)
- Under normal circumstances, the IOs - or more precisely the flipflops driving/capturing the IOs - have zero latency, unless it is
 - Specified as source latency in the SDC
 - Included in input delay
- IC Compiler can compute the median insertion delay and apply it as clock latency on the IO paths (upper arrow in diagram)

```
clock_opt -update_clock_latency
```

Or after CTS: update_clock_latency

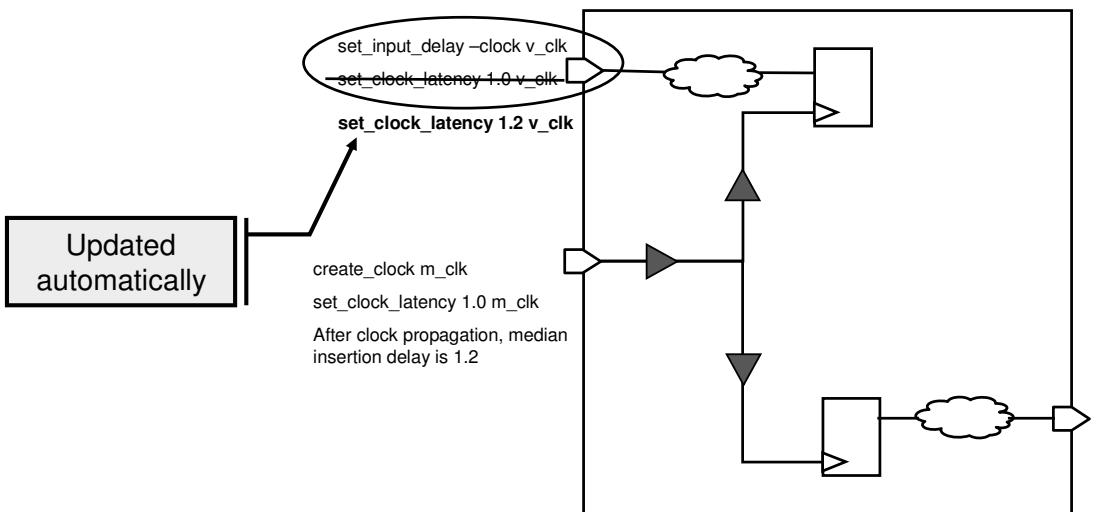
4-52

Auto Update with Virtual Clocks

- Latency can also be updated for ports that were constrained using a virtual clock. First, define the related virtual clock:

```
set_latency_adjustment_options \
    -from_clock m_clk -to_clock v_clk
```

- Second, use `update_clock_latency`



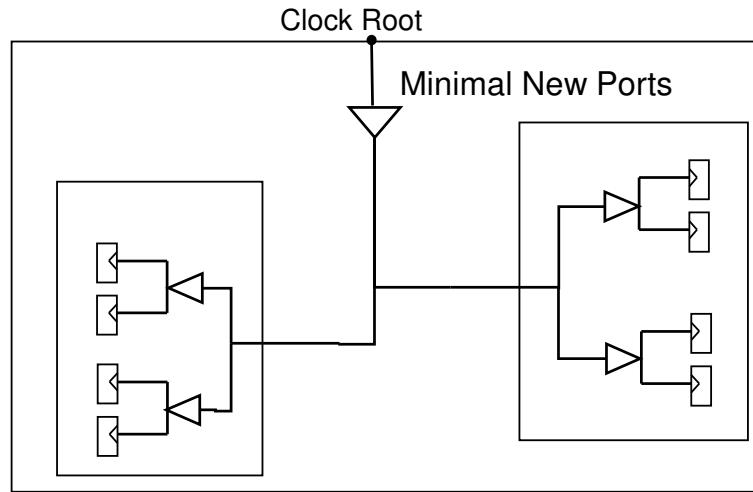
4-53

Appendix B

CTS with Logical Hierarchy

CTS with Logical Hierarchy

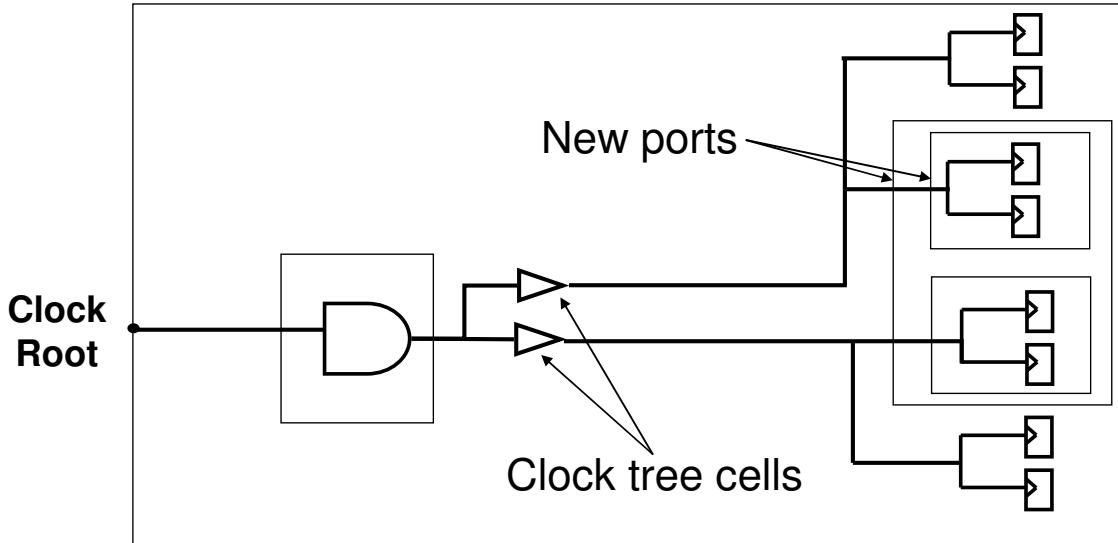
- Clock tree cells are added at the lowest level of common hierarchy
- Ports are added as necessary for the sub-modules
 - Minimal number of new ports created



4-55

Clock Tree Cells Added in Top Hier

- New clock tree cells are added at the lowest common hierarchy of the sinks and new ports are created to connect to them (if required)



4-56

Appendix C

Clock tree configuration control

Clock Tree Configuration Control

- Provides additional flexibility to control the results of clock tree synthesis through a predefined clock tree structure (per net basis)

```
compile_clock_tree \
    -config_file_read <file_name> \
    -config_file_write <file_name>
```

- Flexibility for providing Hard or Soft clock tree configuration
- Embedded CTO operations can change the clock tree reference, but will retain the clock tree structure

4-58

Clock Tree Configuration Syntax

■ Hard Configuration

```
begin_clock_tree 6
clock_net core/clk
buffer_level bufx6 2
buffer_level invx12 4
buffer_level bufx12 7
buffer_level invx12 11
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

■ Soft Configuration

```
begin_clock_tree 0
clock_net core/clk
buffer_level invx12 4
buffer_level bufx12 7
buffer_level invx12 0
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

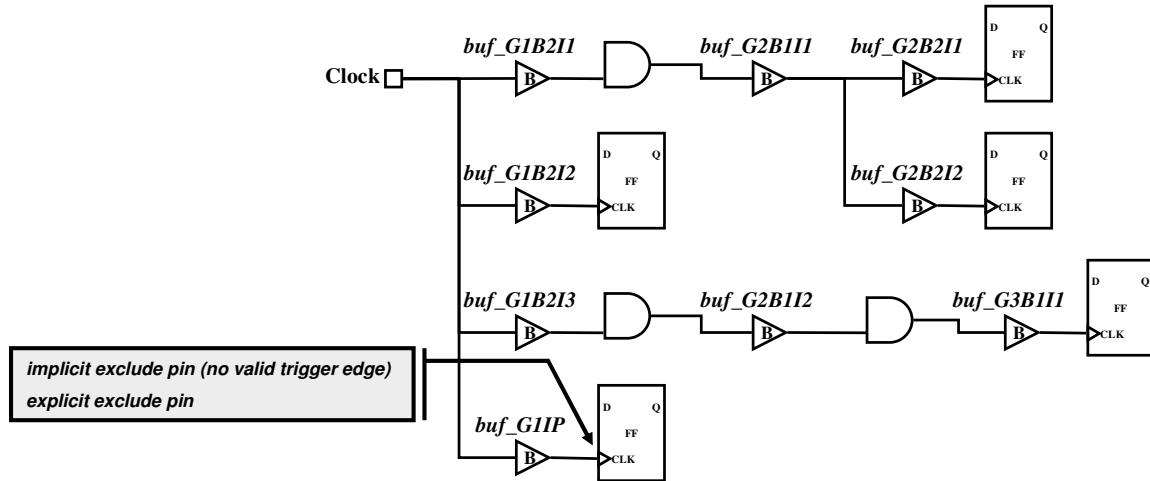
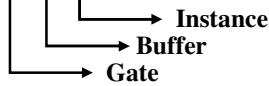
4-59

Appendix D

CTS Naming Convention

CTS – Naming Convention

buffername_G#B#I#



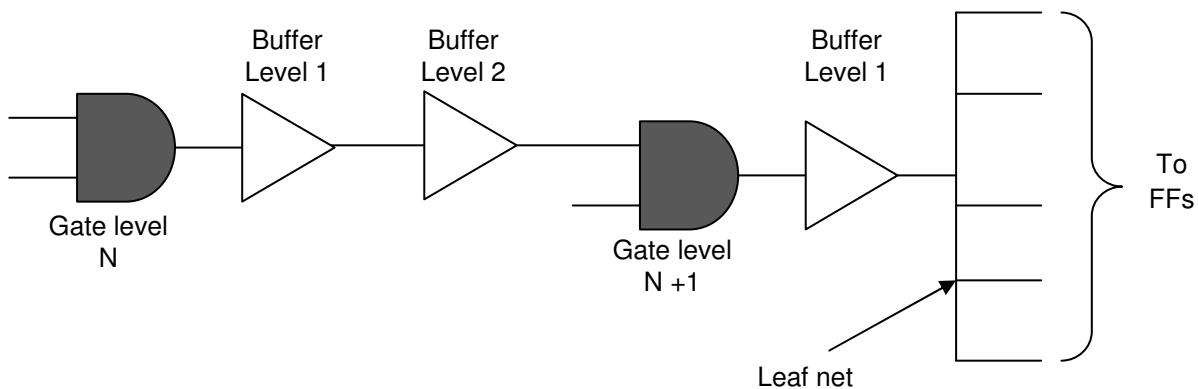
You can further name the clock buffers using:

```
set cts_instance_name_prefix "MYPREFIX"
```

Which produces: MYPREFIX_clkbuf1x24_G1B1I23

4-61

GxByIz: x=gate-level (after 1st gate → x=2), y=buffer-level, z=uniquified by smallest integer
(P=in front of ignore pin)



This page was intentionally left blank.

Agenda

**DAY
3**

5 Multi Scenario Optimization



6 Routing and Crosstalk



7 Chip Finishing and DFM



CS Customer Support

Unit Objectives



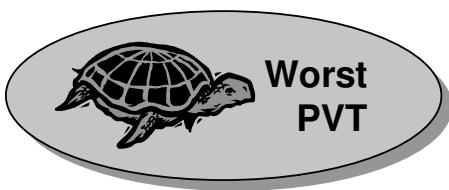
After completing this unit, you should be able to:

- **Describe the need for Multi-Corner and Multi-Mode analysis and optimization**
- **Specify a scenario in IC Compiler**
- **Describe the advantages of and how to use *on chip variation***
- **Analyze the design under all or some scenarios**

5-2

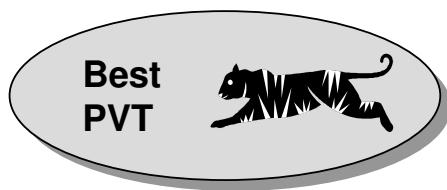
Timing Analysis during Optimization

Slowest Delays



Setup
Violated?

Fastest Delays



Hold
Violated?

- Typically, when optimizing with implementation tools, setup and hold are checked in the worst and best corner respectively, using one set of constraints



PVT: Process, Voltage and Temperature

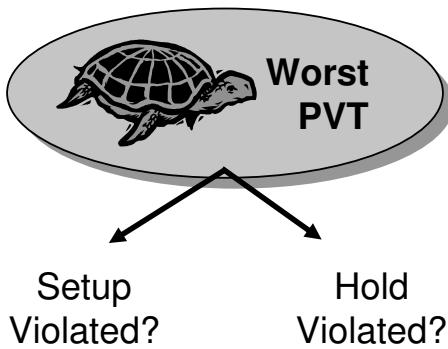
5-3

PVT refers to a variation in cell delays and cell pin capacitances due to fabrication process variance, power supply voltage variations, and temperature (which could vary because of power consumption, ambient temperature, package type or cooling method).

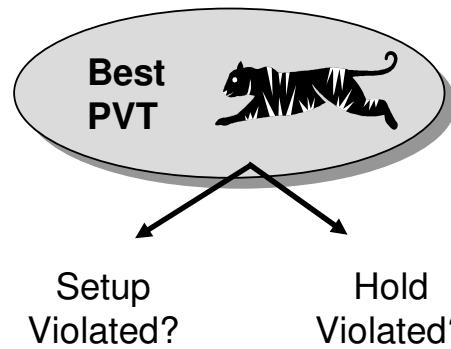
What about other Situations?

Is min-max analysis under one set of constraints enough?

Slowest Delays



Fastest Delays



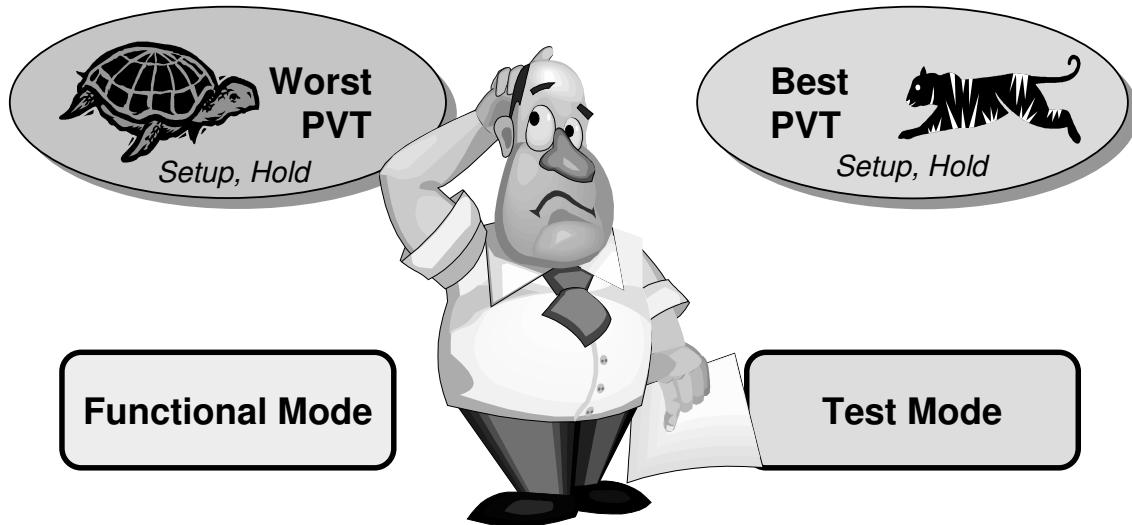
Offer one reason to perform setup and hold in both corners.

5-4

Answer: Generally, min-max analysis under one set of SDC constraints is no longer enough. A good reason to perform setup and hold in both corners: The clock skew can be much worse under slow PVT. The clock skew has a huge impact on hold. Therefore – you should check for hold under the slow PVT corner.

Performing setup and hold checks under both PVT extreme corners is sometimes referred to as “**four-corner analysis**”.

What is the # of Runs?



How are different modes represented?

You will have to perform:

- 2 separate runs.
- 4 separate runs.

5-5

The # of runs = # Modes x # PVT corners.

How are different modes represented? – Using different design constraints (and constraint files)

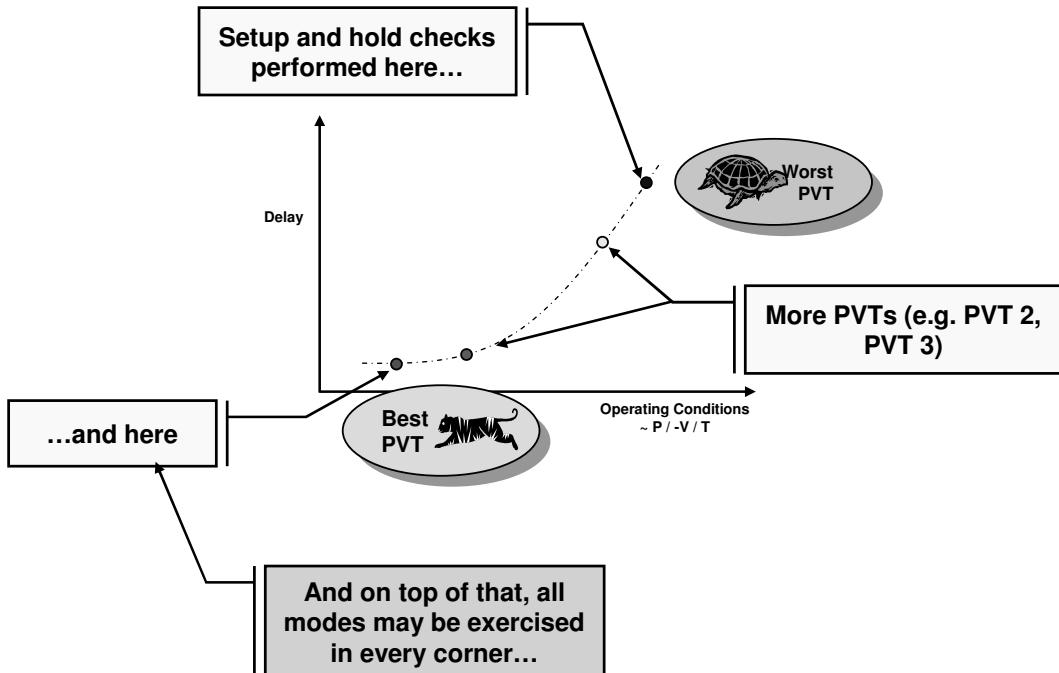
How many classical runs are necessary? – 4 runs. Functional and test under both slow and fast PVT.

(Setup and hold can be calculated during the same classical “run”.)

Examples of constraints in test mode while the chip is a device under test (DUT):

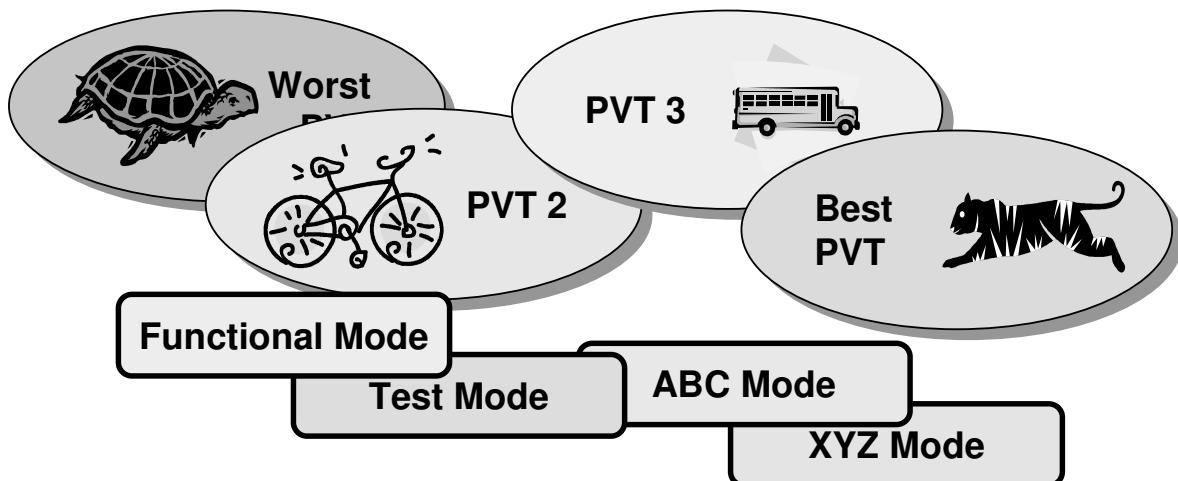
- Tester clock period and clock source
- Model tester skew on the input ports
- Different timing exceptions
- Different setup/hold on the output ports
- The scan chain is exercised in test mode (but not in functional mode)

Corners Represent Delay at Different OpCon



5-6

Multiple Corners – Multiple Modes



- IC Compiler's MCMM allows concurrent optimization under multiple corners and modes
- We call a mode + corner a scenario

5-7

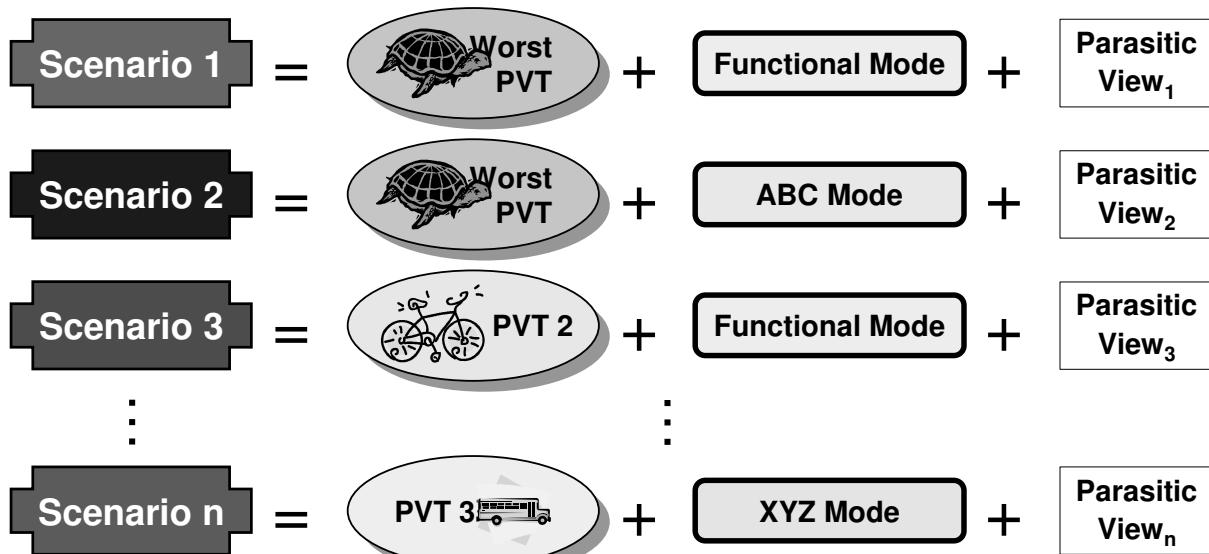
High performance requirements, coupled with extended battery life and increased system integration has made multi-mode design and multiple operating conditions a requirement for many of today's new products. The ability to produce optimized designs that can operate in many different modes and at many different operating conditions is crucial to the success products in today's marketplace.

As a result of these demands in the electronics market, many Synopsys customers are requesting intelligent multi-mode and multi-operating condition (MCMM) handling in the Galaxy platform. The reason for this request is that clearly their current designs have multiple modes of operation and operate at multiple operating conditions. However, the flow used to achieve these designs requires extensive manual intervention, is both time-consuming and error prone, and is not extendable to the next generation design and process.

There have been sequential approaches tried before for this concurrent approach where the constraints are merged in order to achieve compliance across all scenarios, but these approaches can lead to convergence problems. The optimal solution to the MCMM problem is therefore to perform concurrent analysis & optimization.

In IC Compiler, we now support concurrent multi-scenario analysis and optimization.

Scenarios



PVT	\rightarrow	Operating Condition
Mode	\rightarrow	SDC Constraints
Parasitic View	\rightarrow	TLU+, SPEF

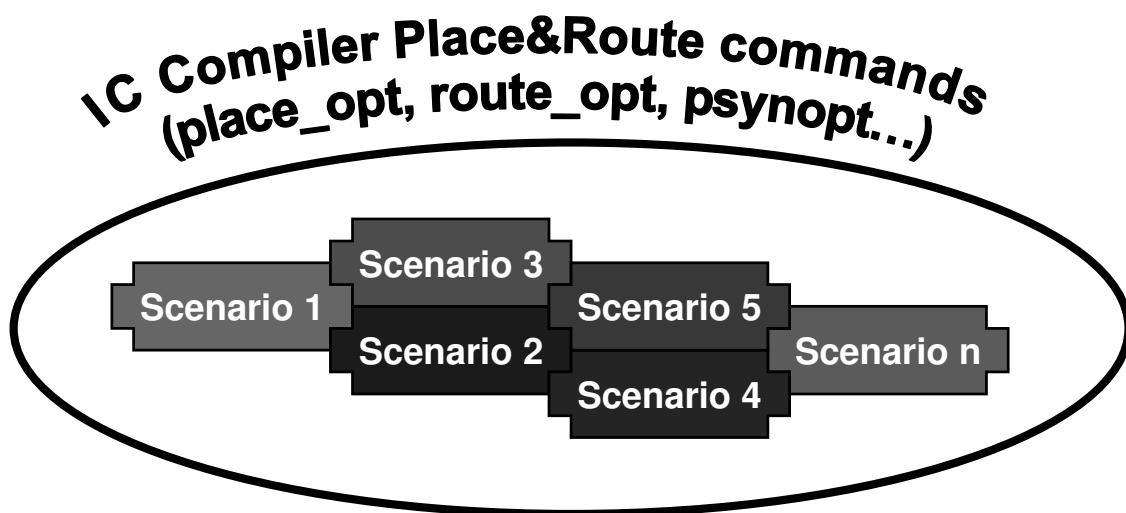
5-8

A scenario refers to a mode and/or corner that can be analyzed or optimized. A mode is defined by a set of clocks, supply voltages, timing constraints, and libraries. It can also have annotation data such as SDF or parasitics files. A corner is defined as a set of libraries characterized for process, voltage and temperature variations. Corners are not dependent on functional settings, but rather result from process variations during manufacturing along with voltage and temperature variations in the environment in which the chip will operate.

Variation support in MultiCorner: A voltage variation flow implies that the TLU+ and temperature is the same for each scenario while only the operating voltage is changed.

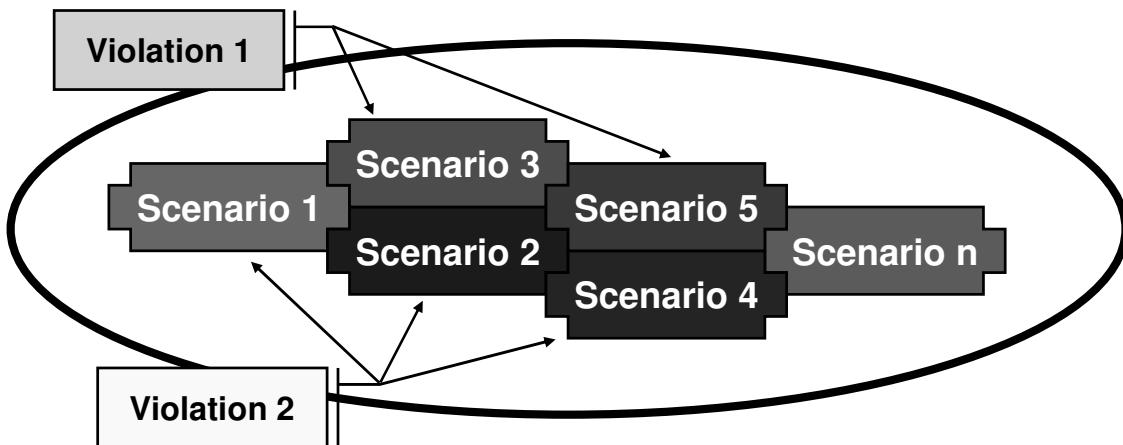
Note: In this example, we are using one operating condition per scenario. Of course you can use 2 operating conditions with the set_operating_condition command, in either bc_wc or on-chip variation.

Multi Scenario Solution in IC Compiler



5-9

How Are Violations Fixed?



- The same violation might be present in multiple scenarios (albeit with a different negative slack)
- IC Compiler will work on the path such that it improves across all scenarios
- A -2 slack in S1 will have a higher weight than a -0.1 slack in S4!

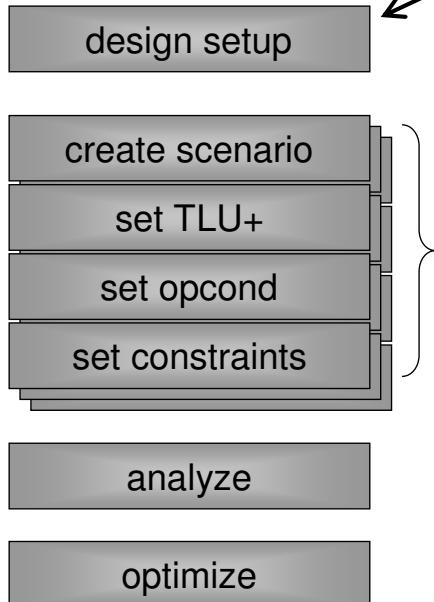
5-10

Concurrent MCMM optimization works on all violations across all scenarios and as such eliminates the convergence problems observed in sequential approaches. Optimization is performed for setup, hold, DRC, area, and power (leakage) . MCMM optimization utilizes a concurrent costing engine, which ensures that every transformation is acceptable for all scenarios' costs.

As a result, the timing and constraint reports show worst case timing across all scenarios. Timing analysis can be performed in one of two ways - a traditional min/max methodology or via the PrimeTime-like early/late analysis approach utilizing the on-chip variation (OCV) switch in the set_operating_conditions command.

MCMM / Scenario Setup

■ Setup falls into two parts:



- Global Setup

Applies to **all** scenarios:

- ◆ target_library
- ◆ link_library
- ◆ (set_min_library)

- Scenario-specific setup

For **each** scenario:

- ◆ set_tlu_plus_files
- ◆ set_operating_conditions
- ◆ Any SDC command (constraints)
 - set_case_analysis
 - create_clock
 - set_input_delay
 - ...

5-11

set_tlu_plus_files

Each scenario must have TLU+ settings or an error is reported:

Error: tlu_plus files are not set in this scenario s1.
RC values will be 0.

set_operating_conditions

Each scenario must have operating conditions or a warning is reported:

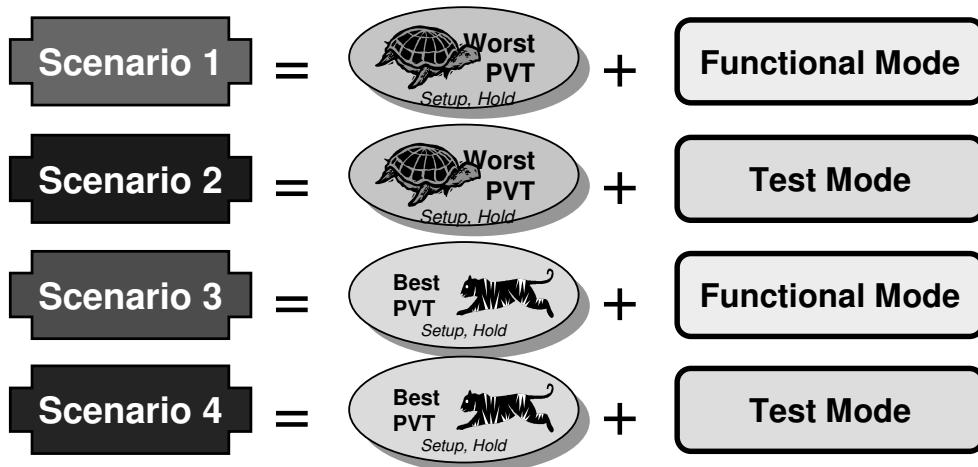
Warning: No operating condition was set in scenario s1 (MV-021)

Note:

You need to ensure that the various constraint files do not contain any link or target_library statements, as these global variables will override each other.

Defining Scenarios

- Scenarios are created one after the other, each describing a mode and a corner
- Example: We will define 4 scenarios, testing for functional and test mode, with setup and hold in the worst and the best corner, using on-chip variation



5-12

If you have never used on-chip-variation, an explanation will follow.

Global Setup

- In our example, we use 2 libraries

- Worst: abclib_max.db:abclib_max, opcond: abc_wc
- Best: abclib_min.db:abclib_min, opcond: abc_bc

- Set up “global variables”:

```
lappend search_path ../ref/db
set target_library "abclib_max.db abclib_min.db"
set link_library "* abclib_max.db abclib_min.db \
    ram_max.lib ram_min.lib ..."
```

- In every scenario, IC Compiler has to be able to “map” to the library and operating condition that is specified in the “max” corner – see scenario 4! →

5-13

Libraries have a library name and a file they come in. These don't have to be the same.

Example:

```
foo.db:stdcell
bar.db:stdcell
```

Here, we have two libraries with the same name, but they are in different files. IC Compiler can differentiate between the two, because the “full name” of the library consists of the file name, colon, library name. Of course, it is helpful to avoid same library names, as it can be confusing to the user.

Scenario-Specific Setup – S1

Scenario 1

```
create_scenario func_worst_corner  
read_sdc func_wc.sdc  
set_tlu_plus_files -max_tluplus abcmax.tlup \  
-tech2itf_map abc_tlup.map  
  
set_operating_conditions \  
-analysis_type on_chip_variation \  
-max abc_wc \  
-min abc_wc  
  
set_case_analysis 0 scan_en  
set_case_analysis 0 test_mode
```

SDC constraints for functional mode under worst case.

Both setup (max) and hold (min) use same **wc** operating condition!

Setup functional mode

5-14

Each scenario is initialized with `create_scenario`.

To get a list of all defined scenarios, use `all_scenarios`.

To remove a scenario, use `remove_scenario`.

Scenario-Specific Setup – S4

Scenario 4

```
create_scenario test_best_corner
read_sdc test_bc.sdc
set_tlu_plus_files -max_tluplus abcmin.tlup \
                   -tech2itf_map abc_tlup.map

set_operating_conditions \
    -analysis_type on_chip_variation \
    -max abc_bc \
    -min abc_bc

set_case_analysis 1 test_mode
```

SDC constraints for test mode under best case.

bc operating condition specified for “max” (setup) analysis

Setup test mode

5-15

To test the chip under test conditions, test_mode is forced to 1, but scan_en remains unconstrained! For every scenario defined, the slate is clean – no case analysis is active unless you set it.

Switching between Scenarios

```
icc_shell> all_scenarios
func_max func_min test_max test_min
icc_shell> current_scenario
Current scenario is: test_min
test_min
icc_shell> report_case_analysis
  Pin name          User case analysis value
  test_mode          1
icc_shell> current_scenario func_max
Current scenario is: func_max
func_max
icc_shell> report_case_analysis
  Pin name          User case analysis value
  scan_en            0
  test_mode           0
```

Scenario 2

Scenario 1

Scenario 4

Scenario 3



Note: Concurrent analysis/optimization is
always performed across all active
scenarios, independent of current_scenario.

5-16

You can also define more scenarios than you're using. To activate a subset of all the defined scenarios, use `set_active_scenarios`, which can be listed using `all_active_scenarios`.

Scenarios can be removed with the `remove_scenario` command.

```
icc_shell> all_scenarios
s1 s2
icc_shell> remove_scenario -all
Removed scenario 's2'
Removed scenario 's1'
icc_shell> all_scenarios
icc_shell>
```

`write_parasitics`

Writes out parasitics (in SBPF format only) for the current_scenario:

```
icc_shell> write_parasitics -format sbpf -output des.sbpf
```

`save_mw_cel -scenario`

Allows for a design containing scenarios to be saved in Milkyway format:

```
icc_shell> save_mw_cel -scenario s1 -as cel_s1
```

Information: Saved design named cel_s1. (UIG-5)

Multiple scenarios can also be saved:

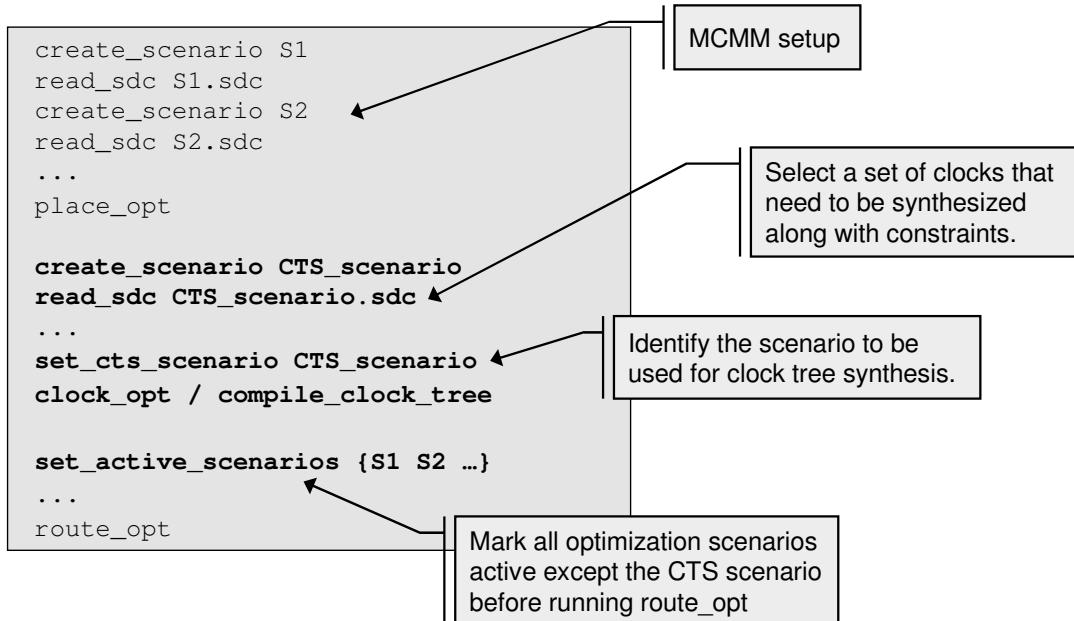
```
icc_shell> save_mw_cel -scenario {s1 s2} -as cel_s1_s2
```

Information: Saved design named cel_s1_s2. (UIG-5)

If no scenario is specified, all scenarios are saved

CTS Operates with One Scenario

During CTS, a single scenario has to be chosen.



5-17

Since applications for MCMM are commonly related to the clock tree it is important to understand the flow impact as well as the current progress of MCMM within the CTS context. One example of an application between MM (multi mode) and CTS is the case where the modes of the clock tree (e.g. functional clocks vs. test clocks) are defined in separate scenarios for their respective clocks. Another is when the designer wants to consider corner effects post-clock tree such as RC variation and timing shifts due to time dependent device reliability issues. Normally the sets of scenarios between pre and post CTS are different and therefore need to be redefined.

Since currently CTS is limited to only use one scenario for its SDC clock constraints this means there is a conflict with the optimization SDC settings for scenarios and for SDC settings for the clock tree synthesis. This is a general issue and not specifically related to MCMM but does impact the flow prescribed when both are used. Currently, IC Compiler's clock tree synthesis does not use the SDC information across the multiple scenarios.

Since CTS requires that the clocks are defined completely in the SDC this is normally defined outside the scenario context in a CTS only SDC flow step. In addition, since the CTS methodology in IC Compiler allows for either the mega command `clock_opt` or basic commands such as `compile_clock_tree`, we recommend that the basic commands are used only for the MCMM flow. However note that the flow is still possible and requires a bit of dexterity when employing post CTS and routing optimizations with MCMM.

Leakage Only Scenario

- You can focus one scenario on leakage optimization only:
 - You apply only the leakage optimization constraints
 - You significantly reduce the runtime and memory footprint as compared with other scenarios

```
create_scenario LeakageOnlyScenario
set_tlu_plus_files -tech2itf_map abc_tlup.map \
    -min_tluplus los_min -max_tluplus los_max
source -v -e leakage.sdc
set_max_leakage_power 0
set_scenario_options -leakage_only true
```

5-18

This feature was added in 2007.03-SP3.

When you use this command, only leakage optimization is performed in the specified scenario; that is, no other optimizations, such as design rule constraints, timing, or area, are carried out in the scenario.

MCMM Timing Analysis (1/3)

report_qor

- Command is layered with info for each scenario

```
*****
Report : qor
Design : DESIGN1
*****
Scenario 's1'
Timing Path Group 'reg2reg'

-----
Levels of Logic:          33.00
Critical Path Length:    694.62
Critical Path Slack:     -144.52
Critical Path Clk Period: 650.00
Total Negative Slack:   -4533.01
No. of Violating Paths: 136.00
-----

Scenario 's2'
Timing Path Group 'reg2reg'

-----
Levels of Logic:          33.00
Critical Path Length:    393.61
Critical Path Slack:     61.18
Critical Path Clk Period: 500.00
Total Negative Slack:   0.00
No. of Violating Paths: 0.00
-----
```

5-19

In MCMM, this command will report QOR metrics for each scenario created.

MCMM Timing Analysis (2/3)

```
report_timing -scenario [all_scenarios]
```

- Command is layered with info for each scenario

```
*****
Report : timing
Design : DESIGN1
*****
Startpoint: TEST_BUF2En
Endpoint: TEST1/TEST2_SYN/latch_3
Scenario: s1
Path Group: clk
Point           Incr      Path      Lib:OC
-----
clock clk (rise edge)    0.00     0.00
input external delay    450.00   450.00 f
...

```

- If used without -scenario [all_scenarios],
report_timing will report on the current scenario

5-20

In MCMM, this command will report timing on a per scenario basis. Currently, the only switch implemented is “-scenario [all_scenarios]” where all scenarios are reported.

MCMM Timing Analysis (3/3)

```
report_constraint -scenario [all_scenarios]
```

- Report is layered with info for all active scenario

Group	(max_delay/setup)	Weighted					
		Cost	Weight	Cost	Scenario		
CLK		10.07	1.00	10.07	s1		
reg2reg		171.16	1.00	171.16	s1		
CLK		90.60	1.00	90.60	s2		
reg2reg		0.00	1.00	0.00	s2		
max_delay/setup		4404.52					
Multi-Scenario							
Constraint	Cost						
max_transition	45.28 (VIOLATED)						
max_fanout	150.00 (VIOLATED)						
max_capacitance	0.00 (MET)						
max_delay/setup	4404.52 (VIOLATED)						
critical_range	4404.52 (VIOLATED)						
min_delay/hold	0.00 (MET)						

5-21

Constraints can be reported in a manner where each scenario's constraints are reported along with the multi scenario costs using the `-scenario scenario_list` option.

```
report_constraint -scenario scenario_list
```

Reports constraints for given list of scenarios of a multiscenario design. Inactive scenarios will be skipped in the report. Each scenario is reported separately. If this option is not given, `report_constraint` will report constraints on all active scenario except `-all_violators` and `-verbose` option. With these two options and without the `-scenario` option, `report_constraints` will only report constraints on current scenario.

How Much is Too Much?

- Too many scenarios can lead to high memory and CPU demands
- Reduce memory and runtime of analysis and optimization by identifying the dominant scenarios
 - Create a subset of essential scenarios for optimization

```
get_dominant_scenarios -scenarios "s1 s2 ..."
```

Identify the set of essential or dominant scenarios

```
set_active_scenarios [get_dominant_scenarios]
```

Automatically direct optimization to dominant scenario subset

5-22

The –scenarios option for get_dominant_scenarios is optional. If omitted, the default is all scenarios.

MCMM Scenario Reduction Analysis

```
*****
Report : Scenario Reduction Violation Summary
Design : design1
Scenarios :
    Column 1 : scen1 (dominant)
    Column 2 : scen2 (dominant)
    Column 3 : scen3 (dominant)
    Column 4 : scen4 (dominant)
    Column 5 : scen5
    Column 6 : scen6
*****
1      2      3      4      5      6
*****
Violation: max delay, total: 4969
# of violations:          1585     2582     1912     1686      0      0
% of total violations:   31.90    51.96    38.48    33.93    0.00    0.00
# of unique violations:  1580     916      241      21       0       0
% of this scenarios violations: 99.68  35.48   12.60   1.25    N/A    N/A
% of total violations:   31.80    18.43    4.85    0.42    0.00    0.00
# of critical violations: 0       1664    1671    1665      0      0
% of this scenarios violations: 0.00   64.45   87.40   98.75    N/A    N/A
% of total violations:   0.00   33.49   33.63   33.51    0.00    0.00
# of worst violations:   5       2       0       0       0       0
% of this scenarios violations: 0.32   0.08   0.00   0.00    N/A    N/A
% of total violations:   0.10   0.04   0.00   0.00    0.00    0.00
# of dominating violations: 1585    2582    1912    1686      0      0
% of this scenarios violations: 100.00 100.00 100.00 100.00    N/A    N/A
% of total violations:   31.90   51.96   38.48   33.93    0.00    0.00
Largest violation:      -2.86   -5.02   -7.16   -5.02    N/A    N/A
.
.
.
Active set of scenarios are: s1 s2 s3 s4 s5 s6
Dominant set of scenarios are: s1 s2 s3 s4
```

Reports max delay, min delay, DRC, leakage/dynamic/total power

5-23

The command `get_dominant_scenarios` on its own will perform the analysis and issue the report seen on this slide - from this you can decide which scenarios you want to make active by using `set_active_scenarios` manually or in this case, you could use:

```
set_active_scenarios [get_dominant_scenarios]
```

So, in this slide we can see that the last two scenarios don't add any unique/critical/dominant violations and so these scenarios need not be optimized.

MCMM High Capacity Flow

- Can the dominant scenarios be identified automatically?

```
set mcmm_enable_high_capacity_flow true


- Transparently invokes get_dominant_scenarios within the mega commands
- Default is false

```

5-24

The variable `mcmm_high_capacity_effort_level` controls the effort level for scenario reduction, and is specified as a floating point value between 0 & 10. Default is 0

What are the recommendations / risks in setting a large value for `mcmm_high_capacity_effort_level`?

We are not recommending a particular value as such - although the default (0) will carry out the "100%" analysis - the analysis will account for ALL violations across all scenarios and consider dominance accordingly, but as such, this may not result in any scenario reduction - for example, if a certain scenario contributes only a single "soft" DRC (like max fanout) then this scenario will be retained and that may not be what the user wants.

So, in certain instances the user may want to be more aggressive in his scenario reduction approach and this variable will allow them to do so - But of course, if you're too aggressive you may not cover all your violations sufficiently and so what you'll gain in runtime you may lose in QOR.

Q: Are setup vs. hold treated differently in determining dominant scenarios?

A: Possibly, need to `set_fix_hold` to get the full view.

Analysis Types

To perform the most accurate timing analysis,
and especially with PrimeTime sign-off in mind,
what analysis type should you use:



On-Chip Variation



Best-case worst-case (bc_wc)

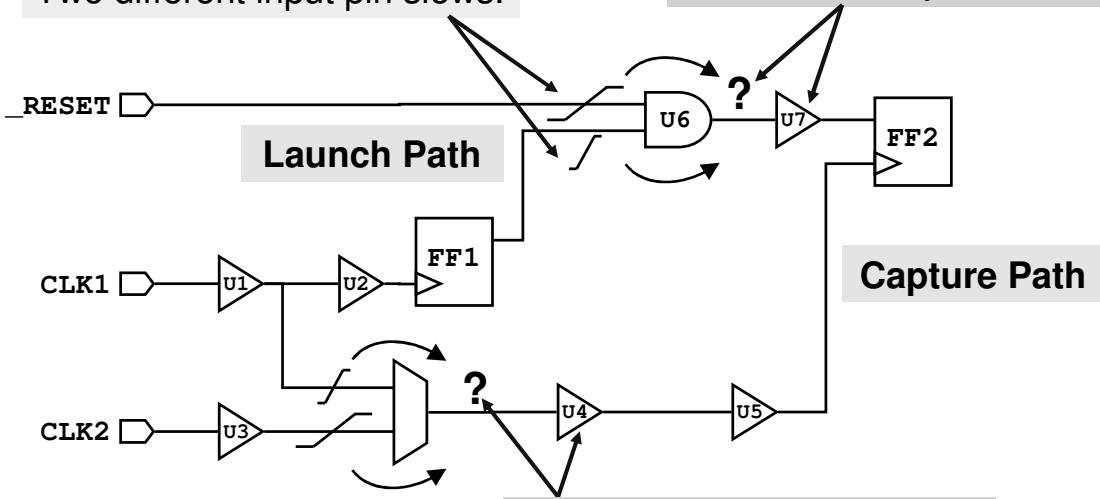
5-25

Launch vs. Capture Path – Use Which Slew?



Two different input pin slews.

Which slew gets propagated for U7 cell delay calculation?



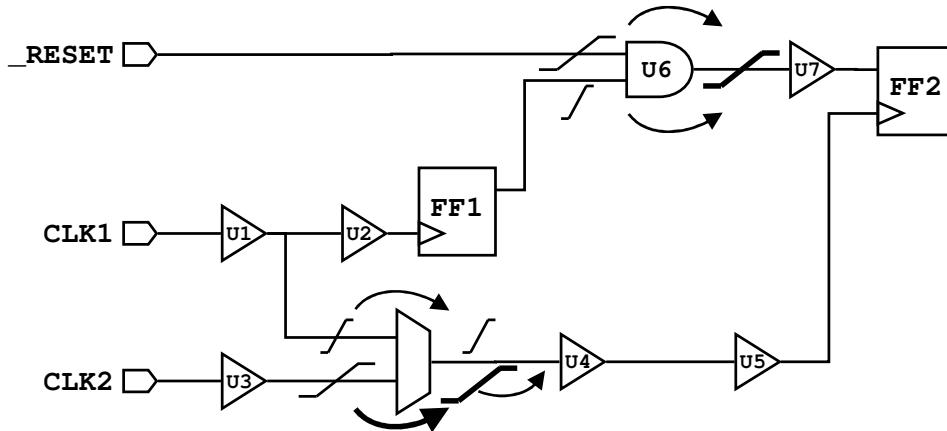
Setup? Hold?

Which slew gets propagated for U4 cell delay calculation?

5-26

Setup is Optimistic in *bc_wc*

- Since ‘worst case’ uses all slow delays, setup checks experience optimism in the following case:



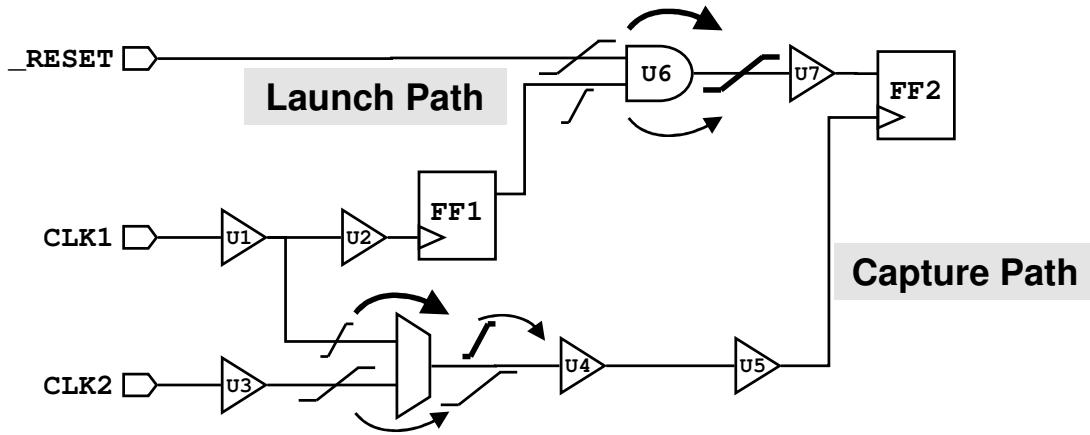
- Slow slew is propagated from mux output
- Slow delays computed for U4 and U5
- But fastest possible capture path is needed!
 - Setup analysis is optimistic for paths captured by CLK1!

5-27

The same applies to hold, because there the fast slews are used in both launch and capture paths, whereas you should have the fast slews on the launch, and the slow slews on the capture path.

On Chip Variation Uses Safer Slopes

- On Chip Variation uses slow slews for the launch path, and fast slews for the capture path:

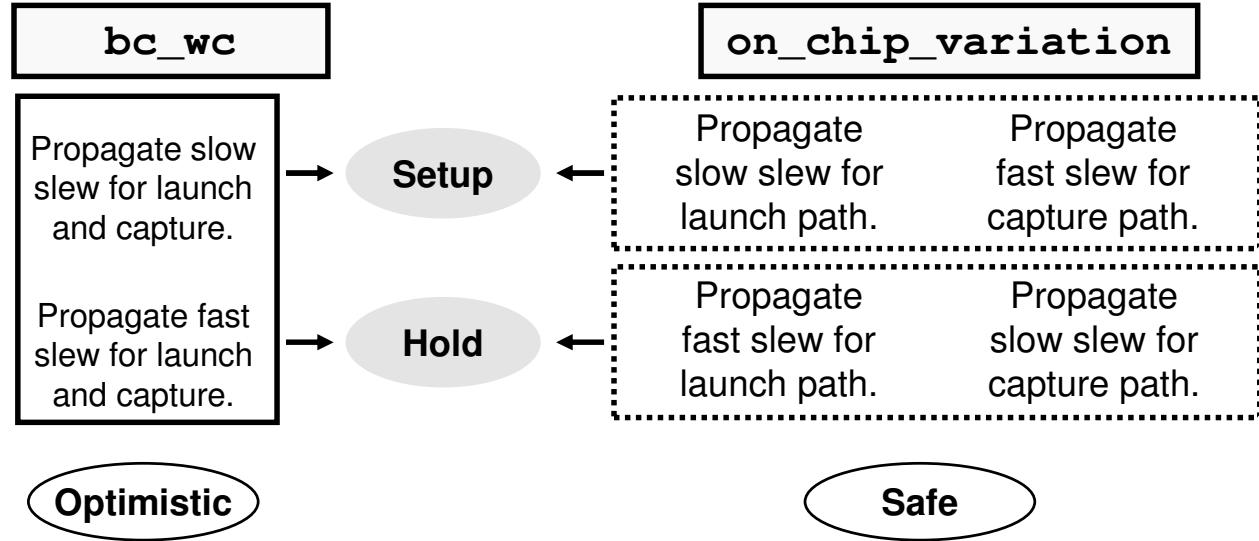


- For hold, it's the opposite:
 - Fast slews for launch, slow slews for capture

5-28

Analysis Types Summary

```
set_operating_conditions -analysis_type ...
```



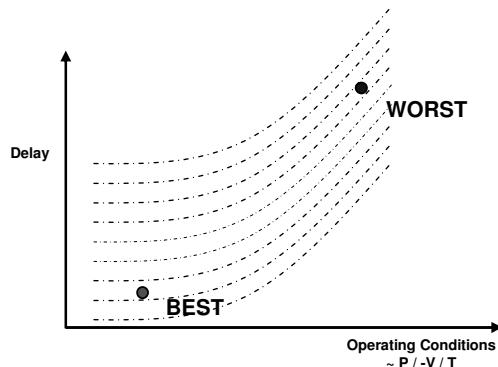
5-29

On-Chip Variation: Single Library

At the very least, use OCV with setup and hold using same operating condition in each corner.

→ *Safer than bc_wc*

```
create_scenario func_worst_corner
set_operating_conditions
    -analysis_type on_chip_variation
    -max WORST -min WORST
create_scenario func_best_corner
set_operating_conditions
    -analysis_type on_chip_variation
    -max BEST -min BEST
```



Setup	Hold
Launch uses -max (WORST) Capture uses -min (WORST)	Launch uses -min (WORST) Capture uses -max (WORST)

5-30

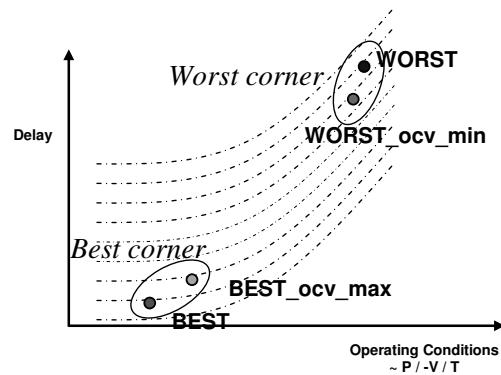
The graphs are a symbolic depiction of the summed effect of Temperature, Voltage and Process variation on Delay. If all 3 were to be plotted individually, the resulting picture would be 4-dimensional.

On-Chip Variation: Multiple Libraries

**Most accurate: Setup and hold
in each corner.**

**2 separate calculated OCV
libraries**

```
create_scenario func_worst_corner
set_operating_conditions
    -analysis_type on_chip_variation
    -max WORST -min WORST_ocv_min
```



Setup	Hold
Launch uses -max (WORST) Capture uses -min (WORST_ocv_min)	Launch uses -min (WORST_ocv_min) Capture uses -max (WORST)

So in this example, for setup:

(slow) slews on the launch path are calculated using WORST
(fast) slews on the capture path are calculated using WORST_ocv_min

5-31

The above shows that the -min and -max PVTs here are both used to calculate setup and hold. This is in contrast to bc_wc, where only the -max PVT is used for setup exclusively, and the -min PVT for hold.

This would also mean that you should never use the slowest (e.g. at 125 degrees C) and fastest (e.g. -40 degrees C) PVTs available together when using on-chip variation, because this is very unrealistic. The two PVTs should describe small temperature/voltage variations around the slow and fast corner respectively. The example on the next page illustrates this.

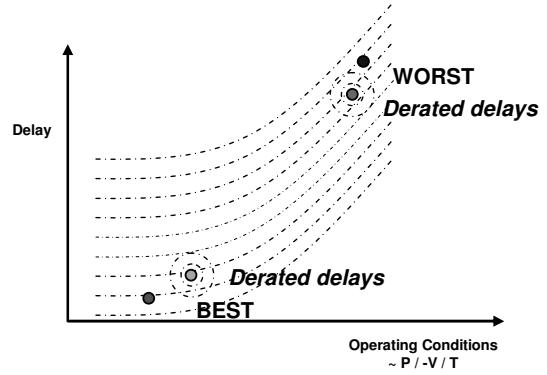
To complete the example from above:

```
create_scenario func_best_corner
set_operating_conditions
    -analysis_type on_chip_variation
    -max BEST_ocv_max -min BEST
```

On-Chip Variation: Single Library + Derating

Pessimistic but safe:
Setup and hold in each corner.
User-specified derating

```
create_scenario func_worst_corner  
set_operating_conditions  
  -analysis_type on_chip_variation  
  -max WORST -min WORST  
set_timing_derate ...
```



Setup	Hold
Launch uses -max (WORST) Capture uses -min (Derated WORST)	Launch uses -min (Derated WORST) Capture uses -max (WORST)

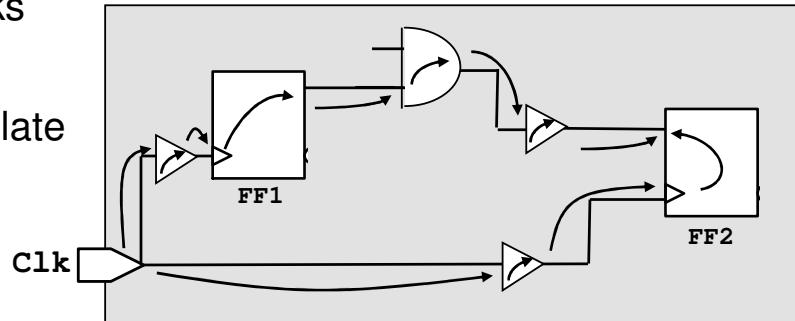
5-32

This completes the example:

```
create_scenario func_worst_corner  
set_operating_conditions \  
  -analysis_type on_chip_variation \  
  -max WORST -min WORST  
set_timing_derate -min -late 0.9  
  
create_scenario func_best_corner  
set_operating_conditions \  
  -analysis_type on_chip_variation \  
  -max BEST -min BEST  
set_timing_derate -max -late 1.1
```

Applying Derating Factors

- If no operating conditions exist for corners you wish to check, you have to derate existing PVTs
- Derating factors will scale the delay values
- IC Compiler allows for global or specific derating:
 - Cells versus nets
 - Launch path versus capture path
 - Timing checks
 - Library cells
 - Early versus late



5-33

Derating factors are applied after slew is chosen.

```
set_timing_derate      # set_timing_derate
  [-min]          (specify derating factor for min operating condition)
  [-max]          (specify derating factor for max operating condition)
  [-early]         (specify the maximum derating factor, default is 1.0)
  [-late]          (specify the maximum derating factor, default is 1.0)
  [-clock]         (specify the derating factors are to apply to clock paths only)
  [-data]          (specify the derating factors are to apply to data paths only)
  [-net_delay]     (specify the derating factors are to apply to nets only)
  [-cell_delay]    (specify the derating factors are to apply to cell delays only)
  [-cell_check]    (specify the derating factors are to apply to cell timing checks only)
  derate_value     (timing derate factor)
  [object_list]    (list of cells and/or designs). Applies globally if not set.
```

Global Derating versus Specific Derating

```
Startpoint: FF1 (rising edge-triggered fl Clk)
Endpoint: FF2 (rising edge-triggered flip flop) report_timing [Clk]
Path Group: Clk
Path Type: max
Max Data Paths Derating Factor : 1.100
Point           Incr      Path
-----  
clock Clk (rise edge)      0.00      0.00
clock network delay (propagated) 1.10 *    1.10
FF1/CLK (fdef1a15)        0.00      1.10 r
FF1/Q (fdef1a15)          0.50 *    1.60 r
U2/Y (buf1a27)            0.11 *    1.71 r
. . .
```

```
report_timing -derate
```

Point	Derate	Incr	Path
clock Clk (rise edge)		0.00	0.00
clock source latency		0.00	0.00
Clk (in)		0.00	0.00 r
clk_iopad/I (pc3d01)	1.00	0.05 *	0.05 r
clk_iopad/PAD (pc3d01)	1.00	0.95 *	1.00 r
U1/A (buf1a27)	1.00	0.01 *	1.01 r
U1/Y (buf1a27)	1.00	0.08 *	1.09 r
FF1/CLK (fdef1a15)	1.00	0.01 *	1.10 r
FF1/Q (fdef1a15)	1.10	0.40 *	1.50 f
U2/A (buf1a27)	1.10	0.01 *	1.51 f
. . .			

5-34

Unit Objectives Summary

You should now be able to:

- **Describe the need for Multi-Corner and Multi-Mode analysis and optimization**
- **Specify a scenario in IC Compiler**
- **Describe the advantages of and how to use *on chip variation***
- **Analyze the design under all or some scenarios**



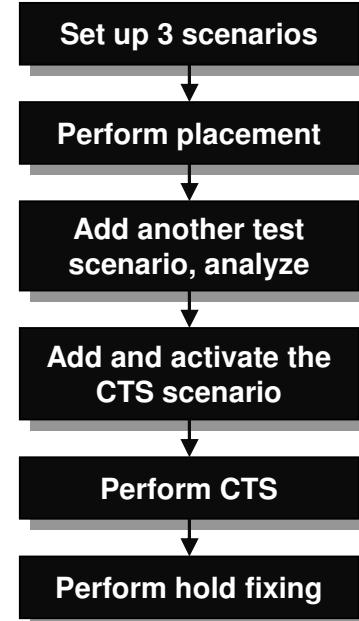
5-35

Lab 5: Multiple Scenario Optimization



90 minutes

Apply multiple scenarios then perform placement, CTS and hold fixing, along with extensive timing analysis.



5-36

Appendix

Library Selection and Grouping

TLU+ Files Support

Resistance and Capacitance Scaling

Coupling Caps across Scenarios

TLU+ Temperature Scaling

Delay Scaling with k-factors

Link Libraries & PVT Assumptions

- When linking a cell instance to the appropriate library for the scenario, MCMM uses the PVT of the cell's max operating condition defined in the scenario
 - Finds first library cell which matches by type with the matching PVT
 - If no suitable cell found in any of the specified libraries:

```
Error: cell TEST_BU2En_BU1/Z (inx4) is not characterized for  
0.950000V, process 1.000000, temperature -40.000000. (MV-001)
```

5-38

The link_library lists all of the libraries that are to be used for linking the design for all scenarios. Furthermore, there may be several libraries (e.g. a standard cell library and a macro library) that are intended to be used for linking a particular scenario. ICC will automatically group the libraries in the link_library list into several sets, and identify the set that must be used for linking a scenario.

ICC uses the nominal process, voltage, and temperature (PVT) values of each library in order to group the libraries into different sets. Libraries with the same PVT values are grouped in to the same set. The PVT of a scenario's max operating condition is used to select the appropriate set.

The set_operating_conditions command is a scenario specific command which specifies the operating condition that is to be used for the design (or a hierarchical instance in the design) in each of the scenarios.

When linking a cell instance to the appropriate library cell for a scenario, MCMM uses the process, voltage, and temperature of the max operating condition associated with that scenario. We find the first library cell which matches by type (i.e. AND2_4) in the set of libraries with matching nominal process, voltage and temperature.

If no suitable cell is found in any of the specified libraries, an error is reported and the user must then verify the operating conditions and library setup – these errors must not be ignored as they imply that no optimization will be carried out.

Unique Identification Of Libraries

- Two libraries with the same name can be uniquely identified by their filenames and library names:
 - foo.db:stdcell vs. bar.db:stdcell
- Two libraries with same file/library names cannot be resolved if they reside in different directories:
 - .../lib/fast/foo.db vs. .../lib/slow/foo.db

5-39

Currently, two libraries with the same name (e.g. “stdcell”) can be uniquely identified by their file names and library names (colon separated). Therefore, library foo.db:stdcell (where foo.db is the name of the library file, and stdcell is the name of the library) is uniquely identified from bar.db:stdcell.

However, two libraries that have the same file name and library name, but reside in different directories, are **not** uniquely distinguishable, and therefore will lead to unexpected results.

Example:

- /remote/snps/testcase/LIB/fast/foo.db, and
- /remote/snps/testcase/LIB/slow/foo.db

Library Grouping

Set 1

Set 2

Link Library (in order)	Nominal PVT	Opconds in library (PVT)
Combo_cells_slow.db	1.2 / 0.85 / 130	WORST (1.2 / 0.85 / 130)
Sequentials_fast.db	0.8 / 1.30 / 25	None
Macros_fast.db	0.8 / 1.30 / 25	None
Macros_slow.db	1.2 / 0.85 / 130	None
Combo_cells_fast.db	0.8 / 1.30 / 25	BEST (0.8 / 1.30 / 25)
Sequentials_slow.db	1.2 / 0.85 / 130	None

```
create_scenario s1
set_operating_conditions -max WORST \
    -library Combo_cells_slow.db
```

- **Nominal PVT was used to group libraries into 2 sets**
- **The -max opcon selects the set, here Set 1**
- **So why should you know this?**

5-40

The term “nominal” is not to be confused with “typical” in a min/typ/max situation. Nominal here means the values the library was characterized at. The .lib file will have lines similar to the following:

```
nom_voltage      : 1.080;
nom_temperature : 125.000;
nom_process      : 1.200;
```

In most libraries there is only one operating condition defined, which reflects the nominal values. Here is an example of a worst case library.

```
operating_conditions("abc_min") {
    process :      1.200;
    temperature : 125.000;
    voltage :      1.080;
    tree_type :    "worst_case_tree";
}
default_operating_conditions : abc_min
```

Incorrect Library Selected

```
set link_library "* lib1.db lib2.db lib3.db lib4.db"
```

	Link Library	Nominal PVT	Opcconds in library (PVT)
1.	lib1.db	1 / 1.30 / 100	WORST (1 / 1.30 / 100)
2.	lib2.db	1 / 0.85 / 100	WORST (1 / 0.85 / 100)
3.	lib3.db	1 / 1.30 / 100	WORST (1 / 1.30 / 100)
4.	lib4.db	1 / 0.85 / 100	BEST (1 / 0.85 / 100)

```
create_scenario s1
set_operating_conditions WORST -library lib2.db
create_scenario s2
set_operating_conditions WORST -library lib3.db
create_scenario s3
set_operating_conditions WORST -library lib1.db
create_scenario s4
set_operating_condition \
    -max WORST -max_library lib2.db \
    -min BEST -min_library lib4.db
```



In this example, cell instances in scenario s2 will **not** be linked to the library cells in **lib3.db** (as intended). They will be linked to library cells in the **lib1.db** library (*assuming that all libraries include the library cells required to link the design*)

5-41

If the max libraries associated with each corner (scenario) do not have distinct PVT's, then cell instances will be linked incorrectly, and you will observe incorrect timing values.

When IC Compiler searches for cells, it searches through the link_library from left to right, and stops once it finds the cell with the correct PVT.

MCMM Supports up to 3 TLU+ Files

- 3 TLU+ files are allowed per IC Compiler run
- Assume tlupA, tlupB and tlupC
 - Scenario S1 can reference tlupA and tlupB
 - Scenario S2 can reference tlupA and tlupC
 - Scenario S3 can reference tlupB and tlupC
 - ...Any combination of A, B and C is valid
- Every one of the 3 TLU+ can have its resistance scaled using the operating condition, and this can be different for every scenario

5-42

Scaling Resistance and Capacitance

- You can scale resistance, capacitance, and coupling capacitance on a per scenario basis using:

```
set_extraction_options  
    -max / -min_res_scale  
    -max / -min_cap_scale  
    -max / -min_ccap_scale
```

- To check the scaling settings, use `report_extraction_options`
- After you specify the scaling values, you should run the `extract_rc` command to see the effect of scaling on the design's parasitics.

5-43

Filtering Coupling Capacitances

- IC Compiler supports coupling capacitance filtering across scenarios.
- After defining each scenario along with its TLUPlus files and operating conditions:

```
set_si_options -delta_delay true
route_opt
extract_rc -coupling_cap
report_timing -scenario [all_scenarios]
```
- The route_opt command will take into account coupling capacitance effects across the nets. In multicorner-multimode flows, the tool's extraction engine generates the parasitics per corner from the TLUPlus and operating conditions pair values.

5- 44

TLU+ Temperature Scaling

- If temperature derating parameters are specified in the TLU+ file, IC Compiler will use the operating temperature from the logical library (db) operating condition to scale resistance
- Temperature Scaling of Resistance
 - TLU+ (ITF) must contain

```
TECHNOLOGY = 90nm_lib
GLOBAL_TEMPERATURE = 105.0
CONDUCTOR metal18 {THICKNESS = 0.8000...
CRT1 = 4.39e-3 CRT2 = 4.39e-7 (optional)
...

```
 - Generate TLU+ with latest grdgenxo
- If no parameters exist, no scaling is (can be) performed.

5-45

To allow for temperature derating, the TLU+ must contain the GLOBAL_TEMPERATURE, CRT1 and (optionally) CRT2 variables.

For detailed information on these parameters, see the Interconnect Technology Format File documentation on solvnet:

https://solvnet.synopsys.com/dow_retrieve/Y-2006.09/strm/strm_2.html

The TLU+ file settings, (accomplished via the set_tlu_plus_files command) must be made explicitly in each scenario – if the TLU+ setup is not correct, an error is issued:

Error: tlu_plus files are not set in this scenario s1. RC values will be 0.

Similarly, the design's operating condition(s) must also be set within each scenario – if this is not defined, an MV-021 warning issued:

```
icc_shell> create_scenario s1
Warning: Any existing scenario-specific constraints are discarded. (MV-020)
icc_shell> report_timing
Warning: No operating condition was set in scenario s1 (MV-021)
icc_shell> set_operating_conditions SLOW_95 -lib max_v95_t125
```

No Delay Scaling with k-factors in MCMM

- **k-factors are used in some libraries to scale the delays in a linear fashion, in order to model the effect of PVT**
 - This is not considered to be accurate any longer, since delays don't scale linearly across all cells in modern libraries
- **k-factor scaling is not supported for MCMM**
 - Operating condition specified for each scenario must match the nominal operating condition for one of the libs in the `link_library` list

5-46

ICC concurrent multi-corner does not support k-factor scaling. This implies that the operating condition that is specified for each scenario must match the nominal operating condition for one of the libraries in the `link_library` list.

Agenda

**DAY
3**

5 Multi Scenario Optimization



6 Routing and Crosstalk



7 Chip Finishing and DFM



CS Customer Support

Unit Objectives

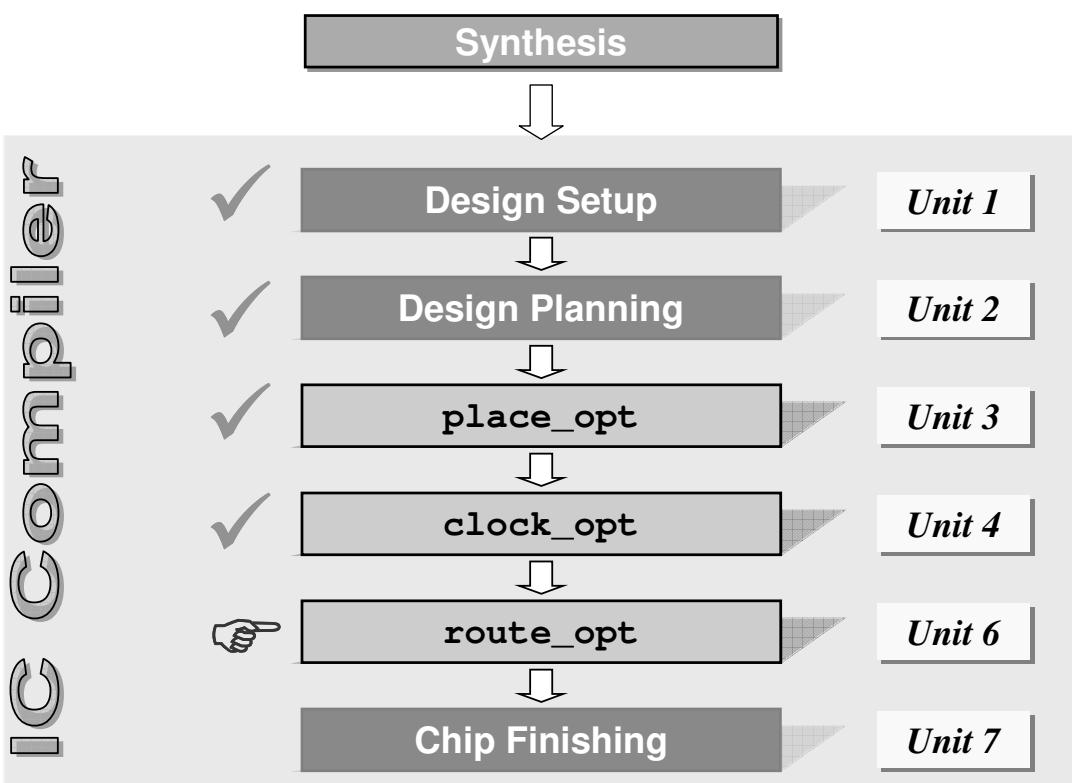


After completing this unit, you should be able to:

- Explain what each of the four Routing Operations accomplishes
- Route the design
- Perform optimizations to improve routing and optimize timing
- Analyze and fix crosstalk violations
- Perform functional ECOs

6-2

IC Compiler Flow



6-3

Design Status, Start of Routing Phase

- Placement - completed
- CTS – completed
- Power and ground nets - routed
- Estimated congestion - acceptable
- Estimated timing - acceptable (~0ns slack)
- Estimated max cap/transition – no violations

```
extract_rc  
report_constraints -all
```

6-4

Congestion, timing and max cap/transition are estimated based on virtual route or a throw away global route.

Pre-Route Checks

- Check design for routing stage readiness
- There should not be:
 - Ideal nets
 - High fanout nets greater than 500
- Use `check_routeability` to check a design's prerequisites for detail routing and report a list of violations

Fix before performing detail routing



```
check_physical_design -stage pre_route_opt
all_ideal_nets
all_high_fanout -nets -threshold 501
check_routeability
```

6-5

The first command shown checks whether the design is ready for routing. The following details are checked:

- the design is placed and legalized
- power/ground pins are connected to nets
- -verbose option shows even more

If high fanout (>500) exist, you should remove ideal net attribute if needed and then add buffer tree prior to routing the design.

`check_routeability`

Checks pin access points, cell instance wire tracks, pin out of boundaries, min-grid and pin design rules and blockages to ensure they meet the design requirements. It performs a check of the design for optimization in order to substantiate any errors in the design that might need to be fixed or what could help to improve the design. This must currently be run on a placed design.

You can use this command at every stage between placement and detail routing. Verify errors in the generated error cell or log file.

You may have to perform manual fixes, as these problems may relate to library problems or issues with the floorplan.

Routing Fundamentals: Goal

Routing creates physical connections to all clock and signal pins through metal interconnects

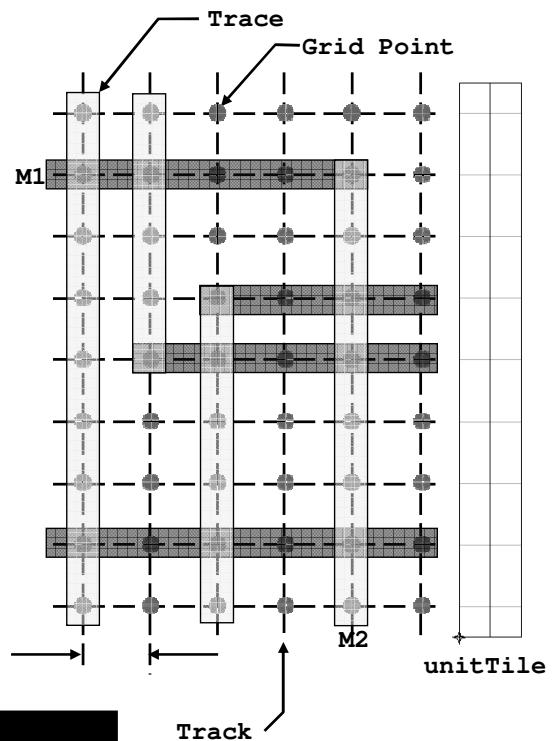
- **Routed paths must meet setup and hold timing, max cap/trans, and clock skew requirements**
- **Metal traces must meet physical DRC requirements**

6-6

Grid-Based Routing System

- Metal traces (routes) are built along, and centered upon routing tracks based on a grid
- Each metal layer has its own grid and preferred routing direction:
 - M1: Horizontal
 - M2: Vertical, etc...
 - Report by:

```
report_preferred_routing_direction
```



6-7

The preferred routing direction guides IC Compiler to route as much of each trace as possible in either an East-West or North-South direction. Some jogs in the non-preferred direction are possible.

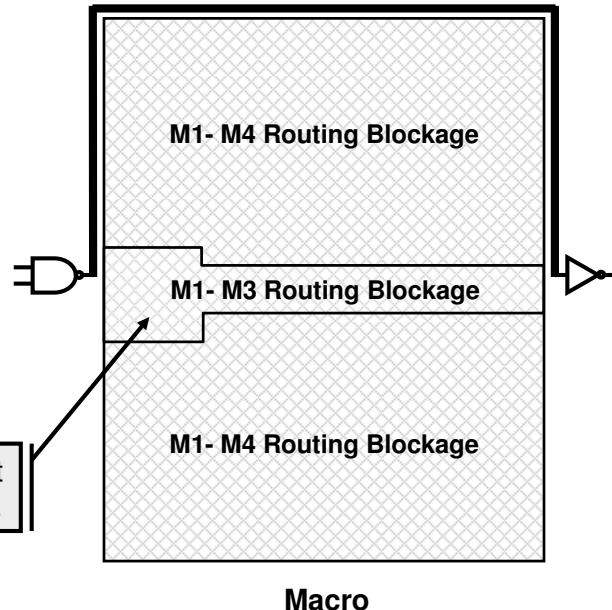
The tracks and preferred routing directions are defined in a "unitTile" cell in the standard cell library

Routing over Macros

- By default IC Compiler will:

- Route over macros
- Not route where there is a routing blockage
- Not route through a narrow channel in the non-preferred routing direction

M4 has a horizontal routing channel but its preferred routing direction is vertical.

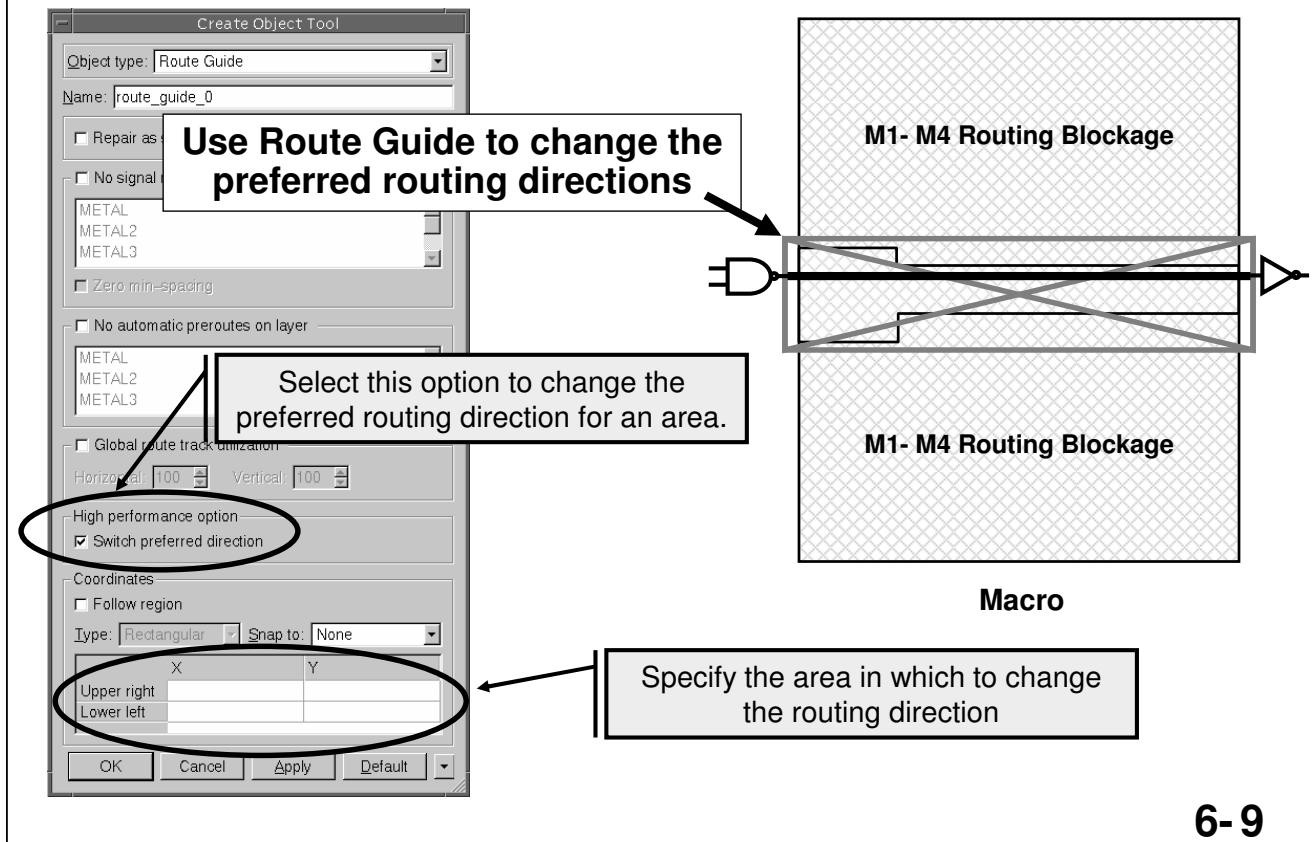


You need to change the preferred routing direction!

6-8

There could be situations where you have to route over a macro in the non-preferred direction. For example, you have a macro that has blockages completely over metal 1, 2 and 3 with some channel space for the top most metal layer M4 and the preferred direction for M4 is opposite to the direction of the channel. A non-preferred direction routing would help to utilize the M4 routing resource over the macro and avoid congestion problems; however, by default IC Compiler will not route over the empty channel in the non-preferred direction.

Change the Preferred Routing Direction



GUI: Floorplan → Create Route Guide ...

In this example, the preferred direction for M4 at the top-level is "Vertical" and the macro has to be placed without rotating. A solution for this kind of situation would be to use a route guide over the channel at the top-level.

Command-line syntax:

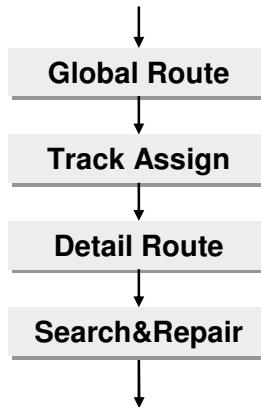
```
create_route_guide -name route_guide_1 \
-coordinate {{0.000 0.000} {300.000 300.000}} \
-switch_preferred_direction
```

The route guide could also be created in the .FRAM view of the macro if all of its instantiations are placed with the same orientation. The above technique can be used for any design that has limited routing resources and space in the preferred direction and also has some empty channel space down the hierarchy that falls in the non-preferred direction at the top-level.

Routing Operations

- **IC Compiler performs:**

- Global Routing
- Track Assignment
- Detail Routing
- Search and Repair



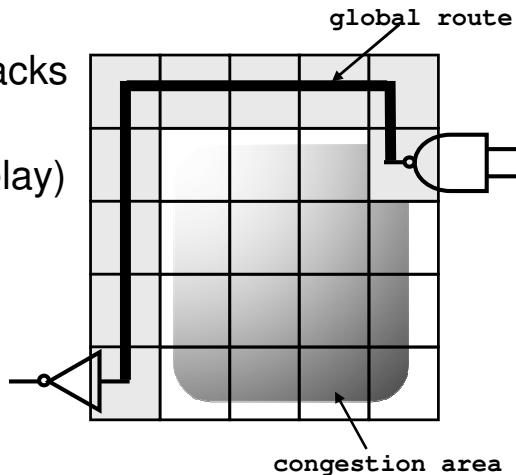
- After global routing, track assignment and detail routing all clock/signal nets will be completely routed and should meet all timing, and most all DRC, requirements
- Any remaining DRC violations can be fixed by Search&Repair

6-10

Note: This unit assumes a moderate design challenge. By definition this implies that all timing requirements will be met at the end of Search&Repair. There are additional optimizations that can be performed but they would be beyond the scope of this workshop.

Route Operations: Global Route

- GR assigns nets to specific metal layers and global routing cells (Gcells)
- GR tries to avoid congested Gcells while minimizing detours:
 - Congestion exists when more tracks are needed than available
 - Detours increase wire length (delay)
- GR also avoids:
 - P/G (rings/straps/rails)
 - Routing blockages



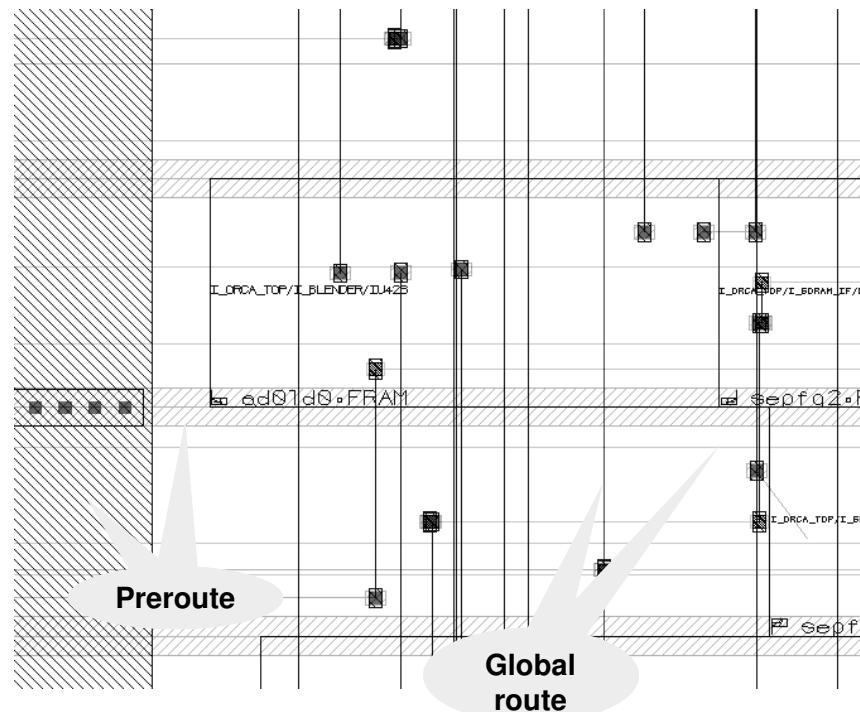
Metal traces exist after Global Route. True or False?

6-11

Route Operations: Global Route Summary

Answer: False!

GR does not lay down any metal traces.



6-12

Route Operations: Track Assignment

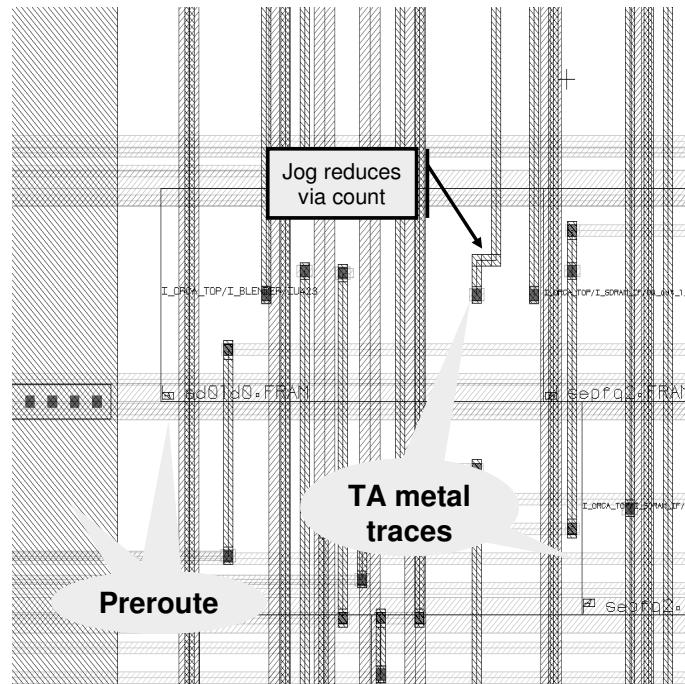
■ Track Assignment (TA):

- Assigns each net to a specific track and lays down the actual metal traces

■ It also attempts to:

- Make long, straight traces
- Reduce the number of vias

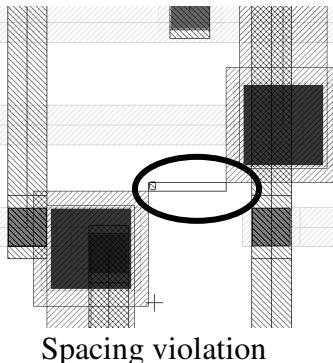
■ TA does not check or follow physical DRC rules



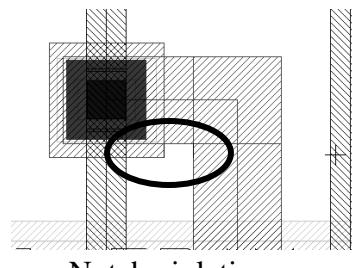
6-13

If track assignment can reduce the number of jogs and jumps in metal traces, this will generally improve timing (since each jump generally requires a via to jump to a higher or lower level metal layer). Reducing the number of vias is generally a plus for reliability and yield since their failure rate is slightly higher than that of a simple, straight metal track in a modern, planarized process.

Example of possible DRC violations after Track Assign



Spacing violation



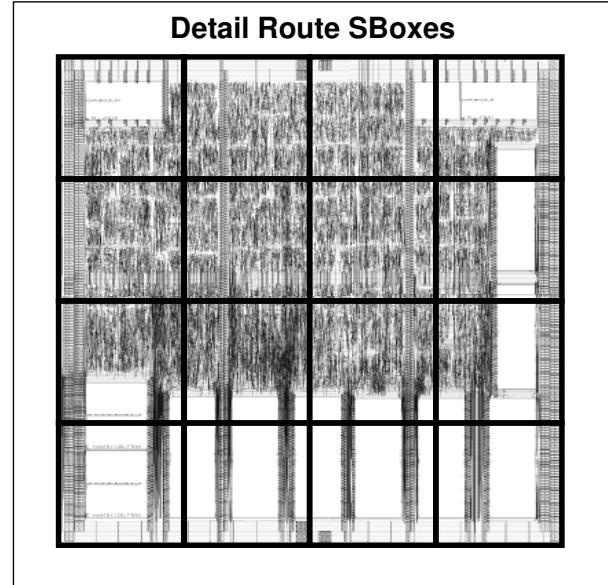
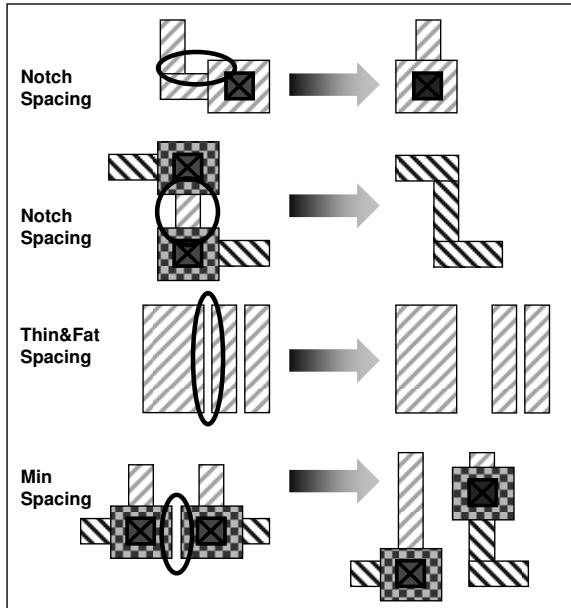
Notch violation

Some additional DRC rules:

- Wire spacing
- Wire width
- Via rules (size, density, stackable etc...)

Route Operations: Detail Routing

- Detail route attempts to clear DRC violations using a fixed size Sbox



- Due to the fixed Sbox size, detail route may not be able to clear all DRC violations

6-14

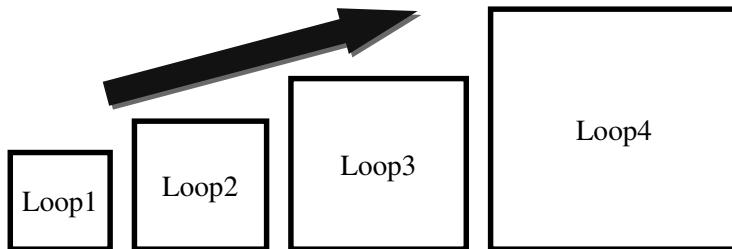
Detail route is concerned with fixing all the DRC violations after Track Assignment. The detail router does not work on the entire chip at the same time like Track Assignment. Instead it works by rerouting within the confines of a small area called an “SBox”. It traverses the entire design box by box until the routing pass is complete. The router takes two passes through the chip using a 5x9 followed by a 9x5 GRC-sized SBox. (GRC = Global Routing Cell)

Sign-off DRC verification uses a complete set of design rules (like a Hercules runset). The technology file contains a subset of these design rules which will be used by the Detail Router and Search&Repair operations. See example below. Modern processes may define additional width/space rules for “fat” wires.

```
Layer      "METAL3" {  
    .....  
    minWidth          = 0.20  
    minSpacing        = 0.21  
    pitch             = 0.41  
    .....  
    fatTblDimension  = 4  
    fatTblThreshold  = ( 0, 0.39, 2.0, 10 )  
    fatTblSpacing    = ( 0.21, 0.24, 0.28, 0.6,  
                        0.24, 0.24, 0.28, 0.6,  
                        0.28, 0.28, 0.28, 0.6,  
                        0.6, 0.6, 0.6, 0.6 )
```

Route Operations: Search&Repair

- **Search&Repair fixes remaining DRC violations through multiple loops using progressively larger SBox sizes**



Note: Even if the design is DRC clean after S&R, you must still run a sign-off DRC checker (Hercules).

- Routing DRC rules are a subset of the complete technology DRC rules
- IC Compiler works on the FRAM view, not the detailed transistor-level (CEL) view

6-15

Search and Repair divides the chip into SBoxes and works through each SBox sequentially trying to fix DRC violations by rerouting within the confines of the box.

Search and Repair starts with smaller boxes. Since changes to the routes are bounded by the small box it causes only minimum disruption to existing routes, but timing might suffer.

Remaining DRC violations are addressed by another pass using a larger size SBox. The larger box potentially gives more routing resources to clear violations, but at the cost of moving routes farther away from ideal. This may effect the timing more significantly than the first pass fixes.

You can control the number of loops over the chip, but you have no simple control over the size of the SBoxes.

If DRC violations cannot be cleared in ~50 S&R loops:

Check the GR congestion map.

If the DRC violations are located in congested areas, you need to fix the congestion first by:

- Manipulating routing obstructions
- Changing the floorplan

Check the log file for warnings about "unreachable pins" and fix as needed (see the Appendix)

Test for Understanding

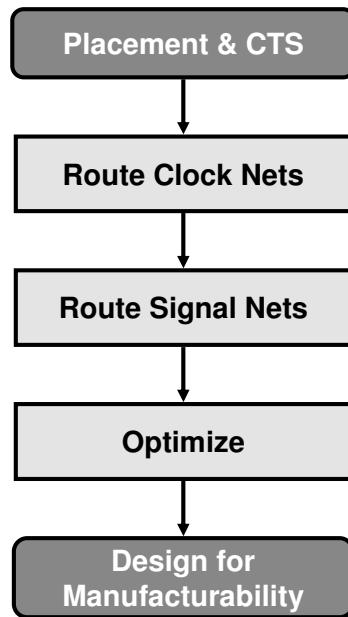


1. What does Global Routing do in congested areas?
2. Assignment of nets to metal layers is done during the Track Assignment stage. T or F?
3. When does IC Compiler use SBoxes? Are they always the same size?
4. Does IC Compiler find all the DRC violations, making a Hercules or other sign-off level DRC extraction run unnecessary?
5. Will IC Compiler route a metal trace in the “non-preferred” direction?

6-16

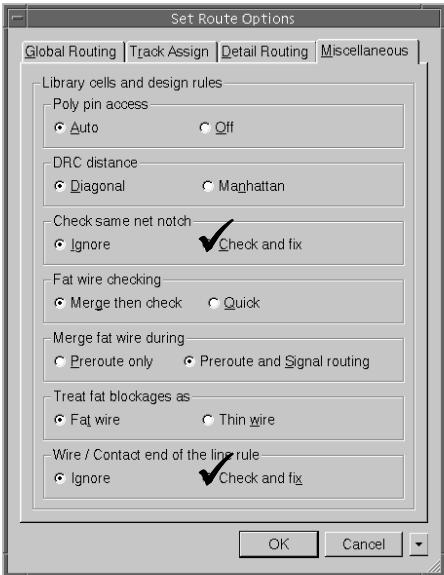
1. It routes around the congestion by moving nets out of congested cells to less congested cells.
2. False. This is done during Global Route.
3. During detail route and S & R, Size is fixed for DR and changes for each S & R loop. Sbox sizes vary from design to design, and are not controllable by the user.
4. No! It looks only at place and route related (FRAM-level) rules. There may be issues within macros or std cells related to polysilicon, diffusion, contacts, n-wells, etc. which are not visible to IC Compiler.
5. Not as a rule but you may see a “jog” now and again.

General Flow for Routing



6-17

Set Routing Options Prior to Routing Steps



- **Enable timing-driven track assignment**
- **Enable “check and fix” for:**
 - **Check same net notch**
 - **Wire/Contact end of the line rule**

Options are saved in the Milkyway CEL

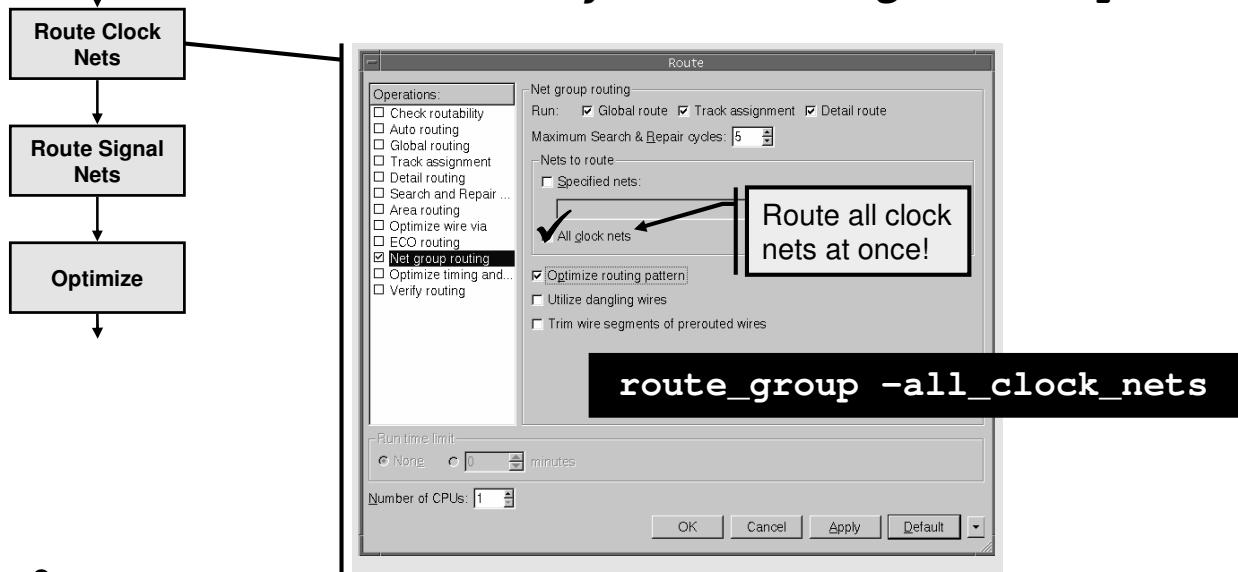
```
set_route_options \
    -track_assign_timing_driven true \
    -same_net_notch check_and_fix \
    -wire_contact_eol_rule check_and_fix
```

6-18

GUI: Route → Set Route Options ...

Route Clock Nets First

... if not already routed during clock_opt!



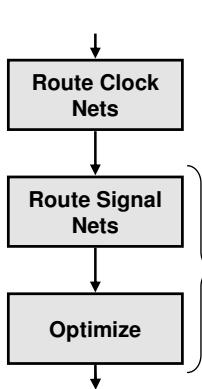
This will give the clocks free rein. If all nets were routed together, clocks would not have any particular priority over other nets.

6-19

Route → Net Group Route ...

All of the clocks are identified by the SDC create_clock and create_generated_clock commands.

Core Routing: route_opt

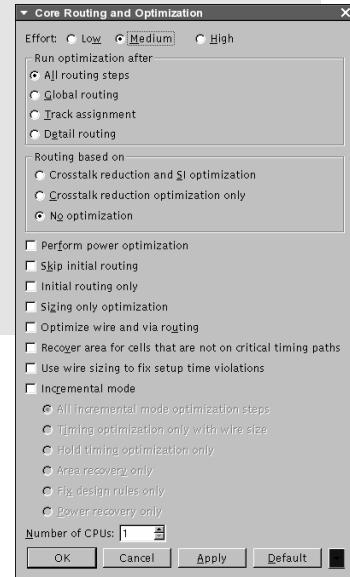


route_opt

-effort low | medium | high
-stage global | track | detail
-power
-xtalk_reduction
-initial_route_only
-skip_initial_route
-incremental
-num_cpus N

All in One:

- **global, track, and detail routing, S&R, logic and placement optimizations with ECO routing**
- **End goal: Design that meets timing, crosstalk and route DRC rules**



6-20

Use **-size_only** to do only size operations. You have refined control via the **-effort** switch:

- effort low : route_opt -size_only will only run *footprint* optimization
 - effort medium : route_opt -size_only will run *in_place* optimization
 - effort high : route_opt -size_only will run *complete size_only* optimizations
- When **-size_only** is specified, the *freedom* of the ecoRouter is also limited

Do not confuse **-incremental** with **-skip_initial_route**

- Use **-skip_initial_route** to run the *identical* route_opt flow, after having routed the design outside of route_opt
- E.g. when first running route_auto, and maybe manually editing some wires, proceed with route_opt -skip_initial_routing


```
route_opt = route_auto + route_opt -skip_initial_route
```
- Use route_opt -inc when you want to incrementally improve the QoR of your design after running route_opt earlier. This will only run 1 loop of optimization.

The **initial_route_only** option causes route_opt to stop after performing routing, it will not perform route optimizations. You can use this option in conjunction with the **-stage** option to control up-to what stage to route. For example, route_opt -initial_route_only -stage global will only run global routing.

In order to use the **-num_cpus** option, you have to first setup IC Compiler routing for multiple CPUs using the command set_distributed_route. You can use LSF as well as directly specify host names.

First route_opt Example

- Instead of running a full routing and post-route optimization, perform the initial route (global routing, track assignment, detail routing, and search and repair) to start out with:

```
route_opt -initial_route_only
```

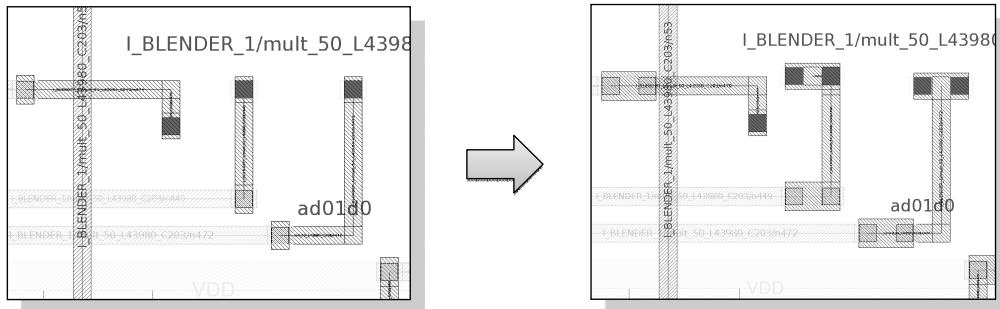
6-21

The `-initial_route_only` option runs global routing, track assignment, detail routing, and search and repair. It does not run the post-route optimization.

Perform Initial Redundant Via

- To get optimal double via rate, perform first a non-timing driven double via insertion

```
insert_redundant_vias -auto_mode preview  
insert_redundant_vias -auto_mode insert \  
-num_cpus $ICC_NUM_CPUS
```



- Later on, during chip finishing, you will execute a Timing Driven double via insertion

6-22

Post Route Optimization Examples

- If post route optimization is needed

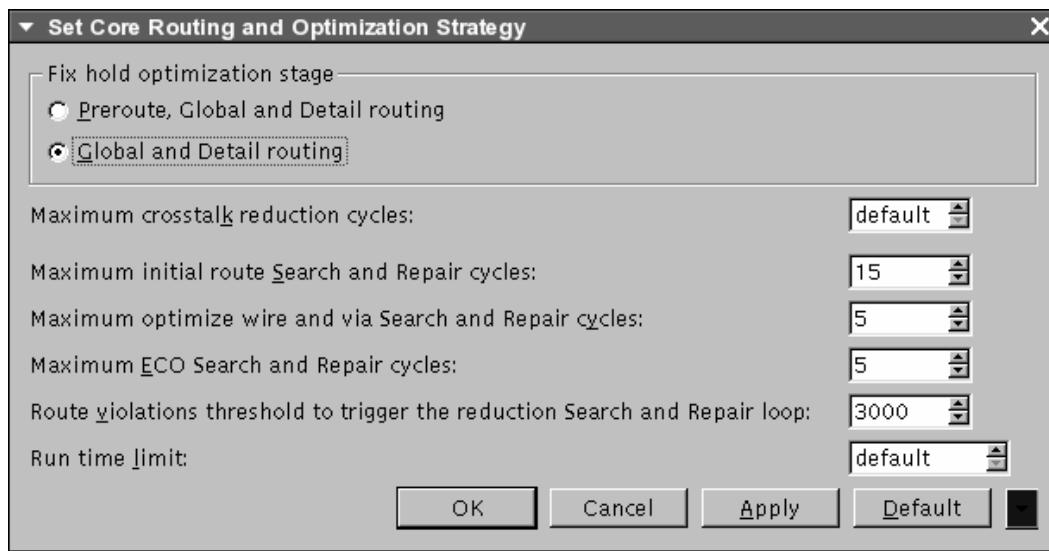
```
route_opt -skip_initial_route \
           -effort low -power
```

```
route_opt -skip_initial_route \
           -effort high -incremental \
           -only_design_rule
```

```
route_opt -optimize_wire_via \
           -effort low \
           -xtalk_reduction
```

6-23

These are examples that you can run after you have analyzed the design.



Core Routing Strategy

- The operations inside `route_opt` can be fine tuned using the command:

`set_route_opt_strategy`

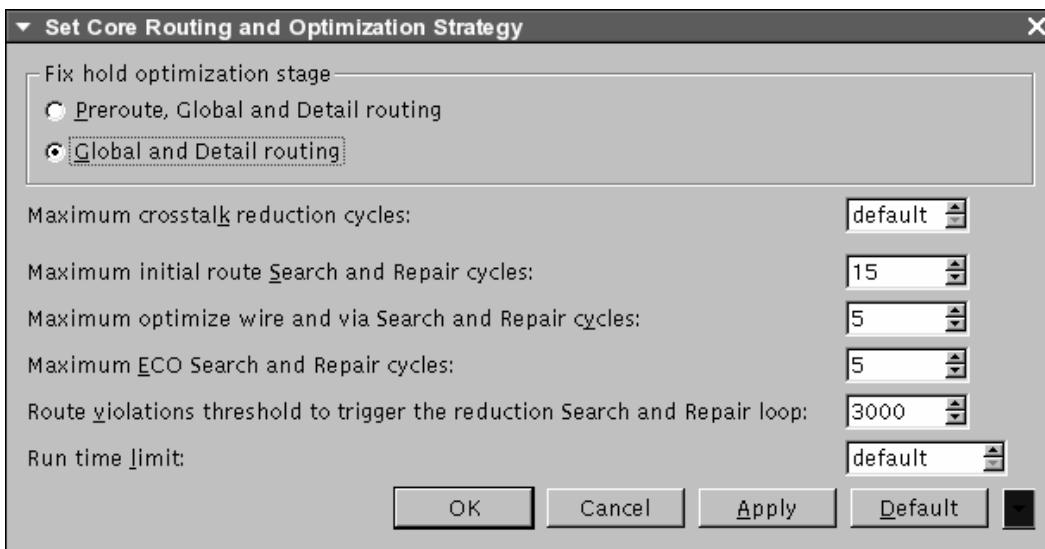
- Increase the number of
 - ◆ Xtalk reduction and xtalk S&R loops
 - ◆ Search & Repair and ECO route loops
 - ◆ Optimize wire/via loops
- Set a runtime limit
- When to run hold time fixing

6-24

```
-fix_hold_mode all | route_base
```

Specifies when to fix hold inside `route_opt`. Inside `route_opt`, there are three different stages to fix hold: pre_route, global route, detail route. With the mode `all`, `route_opt` will perform hold fixing with all three stages. With the mode `route_base`, `route_opt` will run only global route and detail route based hold fix optimization.

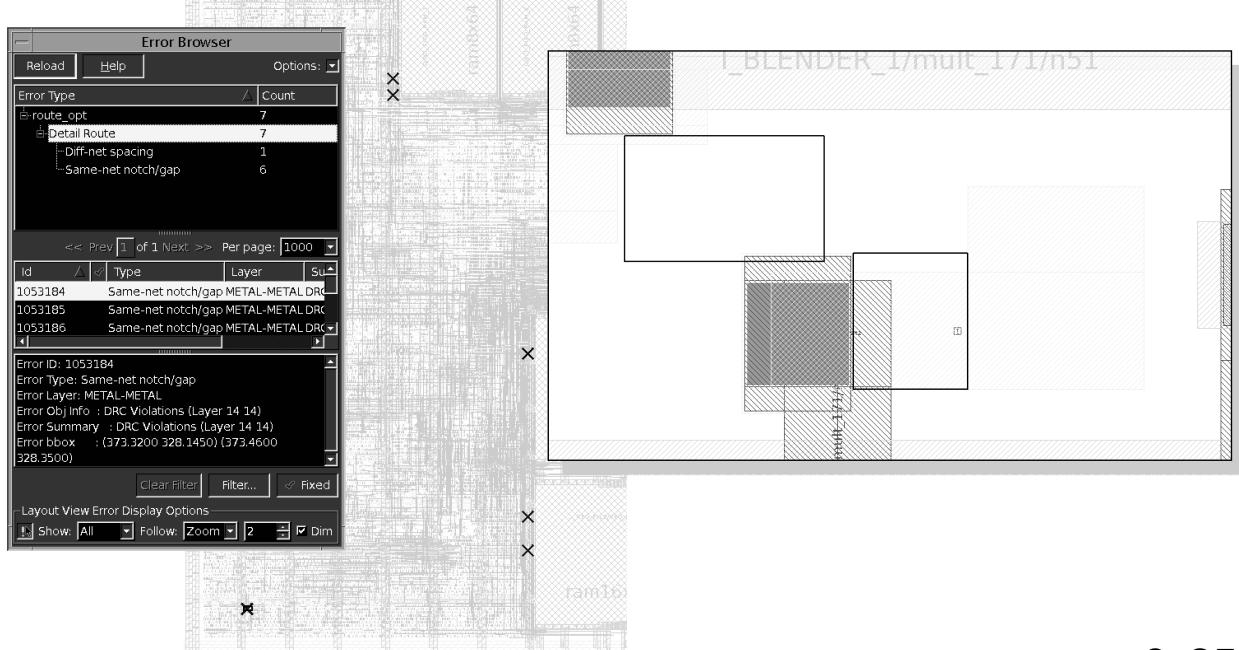
For a detailed list of options, refer to the man page: `man set_route_opt_strategy`



Analysis of the Routing DRC Errors

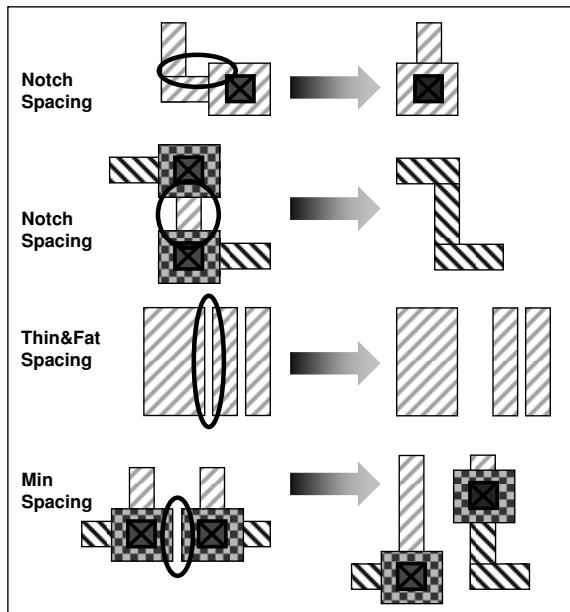
`verify_route`
`verify_drc`

Uses router DRC engine
Uses Hercules for DRC



6-25

Fixing DRC Violations



Use route_opt (fixes timing as well):

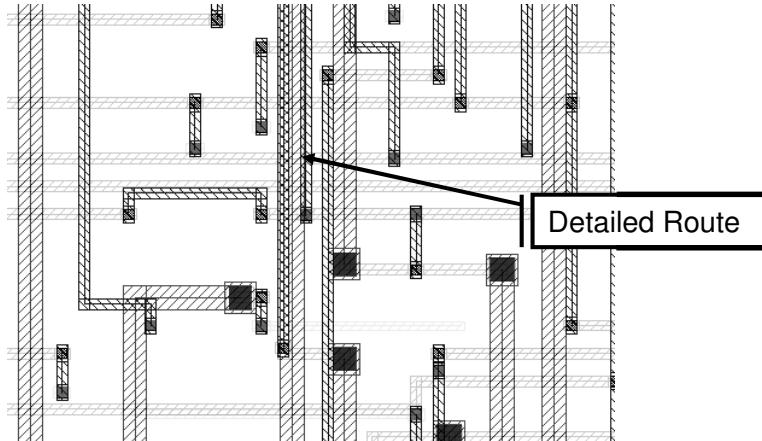
route_opt -incremental

6-26

You can also run search and repair stand-alone:

```
route_search_repair # run search and repair routing operations
[-loop int]           (number of s&r loop: Range 0 to 1000, default is 50)
[-run_time_limit time] (cpu run time limit minutes, default is no limit)
[-reset_width]         (reset to default width)
[-rerun_drc]           (before starting s&r run the design rule checker)
[-reset_min_max_layer] (reset min max layer setting)
[-trim_antenna_of_user_wire] (trim wire segments of prerouted wires)
[-dont_connect_tie_off] (do not connect pg or clock)
[-ignore_open_nets]    (do not connect partially routed wire segments)
```

PostRoute Delay Calculation Algorithms



- After routing, detailed nets are available and extraction can be more accurate
- By default, Elmore is still used
- Arnoldi can be turned on for postroute calculations

6-27

The default delay calculation algorithm in IC Compiler is Elmore.

To turn on the more accurate Arnoldi algorithm, use:

```
set_delay_calculation -arnoldi
```

Test for Understanding



1. What are the 4 *routing* operations that `route_opt` performs?
 2. What happens if `route_opt` is invoked before clock nets are routed? Why is this not recommended?
 3. When is it appropriate to do more Search&Repair?
 4. Does `route_opt` automatically fix cross talk problems?

6-28

- Global routing, track assignment, detail routing, search & repair.
 - The clock nets would be routed at the same time as signal nets. Clock nets would have to compete with all other signal nets for routing resources, resulting in possibly less than ideal clock skew.
 - After Detail Route, to fix remaining DRC violations, or anytime routing is modified thereafter, e.g., ECO Route.
 - No, route_opt_xtalk_reduction will include xtalk processing.

Galaxy Crosstalk

Crosstalk Prevention

Watch for macro placement
congestion and utilization

IC Compiler

Congestion minimization,
max_trans constr.

NDR on clocks

GR/TA with “Crosstalk
Prevention” option

Crosstalk Correction

Post Route Crosstalk
optimization

Star-RCXT
PrimeTime-SI

Repair flow

- Crosstalk prevention
- Crosstalk correction
- Crosstalk sign-off

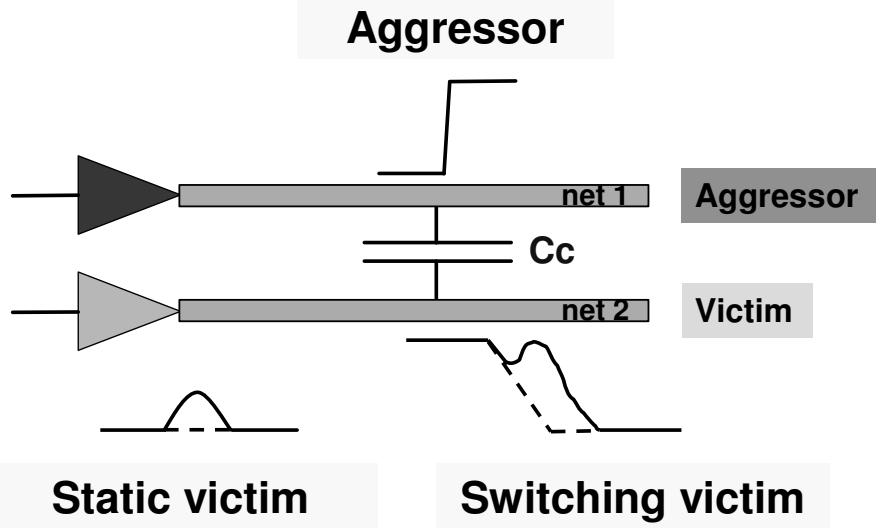
6-29



NDR – Non-Default Routing rule (variable route rules in Astro router)

What is Crosstalk?

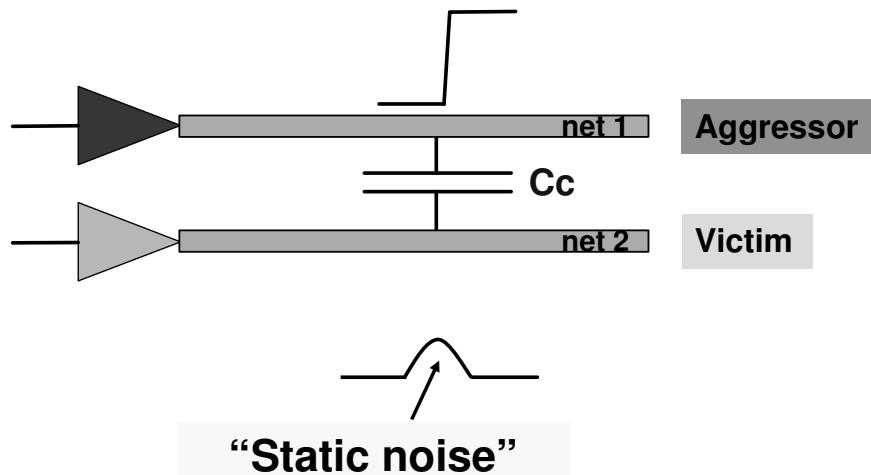
Crosstalk is the transfer of a voltage transition from one switching net (aggressor) to another static or switching net (victim) through a coupling capacitance (C_c)



6-30

Crosstalk-Induced Noise (aka Glitches)

Aggressor nets can create crosstalk-induced noise on static victim nets, also called “static noise”

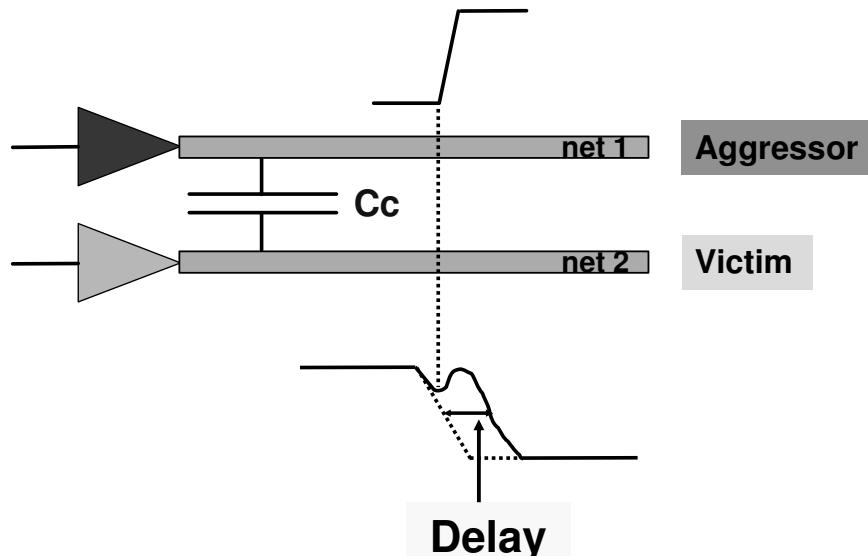


6-31

Crosstalk-Induced Delay

Aggressor/victim nets with overlapping timing windows can cause “crosstalk-induced delay” on victim nets.

This can lead to a speed-up or a slow-down of the victim net



6-32

Crosstalk Prevention in IC Compiler

■ During placement and optimization :

- `set_max_transition`
- `set_congestion_options ...`
- `area_recovery_critical_range` and
`power_recovery_critical_range`
 - ◆ Recommendation: 15% of main clock period

■ During CTS:

- Use of Non-Default Routing rules – e.g. double-spacing

```
define_routing_rule my_route_rule -spacings {...}
set_clock_tree_options [-clock CLK] \
    -routing_rule my_route_rule
```

■ During Global Route and Track Assign :

```
set_si_options -delta_delay true \
    -route_xtalk_prevention true
```

6-33

Non-default rules can be specified for width, spacing, shielding, min/max layer, and special via cut with the following commands.

- `define_routing_rule`
- `set_net_routing_rule`
- `set_net_routing_layer_constraints`
- `set_clock_tree_options -routing_rule`
- `preroute_standard_cells -use_special_via_rule`

e.g. a non-default routing rule can also be applied to critical nets using:

```
set_net_routing_rule -rule MYRULE [get_nets CRITICAL*]
```

Non-default rules can be specified before place_opt, clock_opt or route_opt commands because they are honored by each engine. They are stored in the Milkyway database. Users can report NDRs by using:

- `report_routing_rules`
- `report_net_routing_rules`

Or remove them by using:

- `remove_routing_rules`

Crosstalk *Correction* in IC Compiler

- IC Compiler addresses both **XDD** (crosstalk delta delay) and **Static Noise**
- Crosstalk correction performs both cell-based and route-based optimization (i.e. optimizes gate placement, gate logic, and routing traces)

6-34

Example Full Crosstalk Flow

```
place_opt
set_clock_tree_options -max_transition 0.2 -max_fanout 32
set_clock_tree_references -reference $CLK_BUFFER_LIST
define_routing_rule MYRULE -spacings {M2 0.28 M3 0.28...}
set_clock_tree_options -routing_rule MYRULE
clock_opt

set_si_options -route_xtalk_prevention true \
               -route_xtalk_prevention_threshold 0.35 \
               -delta_delay true \
               -static_noise true \
               -static_noise_threshold_above_low 0.3 \
               -static_noise_threshold_below_high 0.3

route_opt -xtalk_reduction
          Optional: -optimize_wire_via -wire_size

report_noise
report_timing -crosstalk_delta
```

Prevention target more aggressive
Default: 0.45
Unit is Volt.

Default: 0.35
Unit is %

6-35

```
set_si_options
  -route_xtalk_prevention TRUE|FALSE
  -route_xtalk_prevention_threshold <float V>
    Xtalk aware Global Route / Track-Assign
  -delta_delay TRUE|FALSE
  -static_noise TRUE|FALSE
  -static_noise_threshold_above_low <float %>
  -static_noise_threshold_below_high <float %>
```

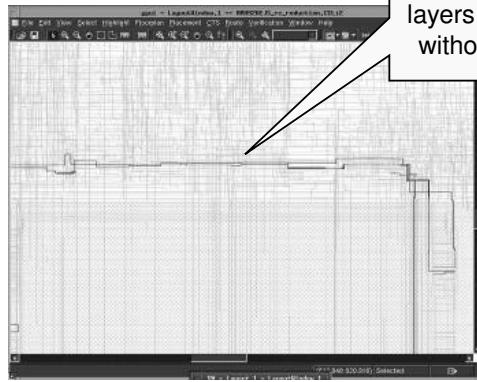
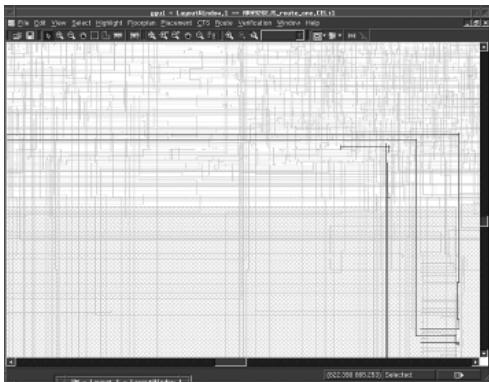


NOTE:

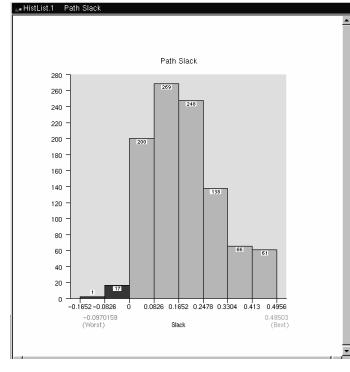
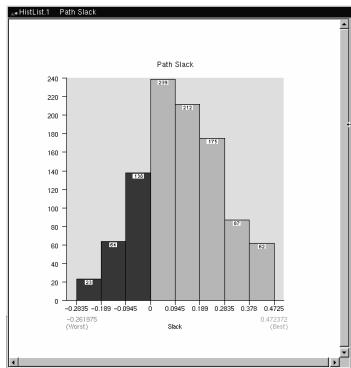
The unit for the **static_noise_threshold_above_*** options has changed with version 2007.03-SP1. The values are now a percentage of the voltage, instead of absolute voltages.

```
report_si_options
  Static Noise Thresholds :
  0.3 (0.49V) above low
  0.3 (0.49V) below high
```

Xtalk-Reduction at Work



Nets are spaced and can switch to different layers to reduce Xtalk without adding cells



6-36

Wire Sizing (Aka Applying NDRs)

```
route_opt -wire_size
```

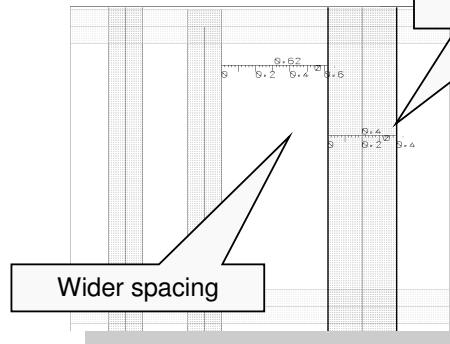
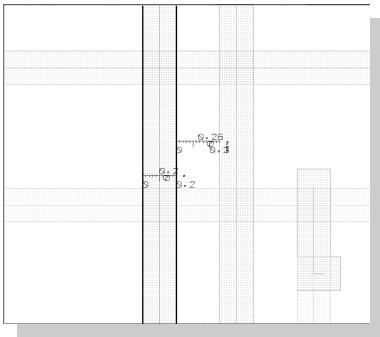
- IC Compiler will apply the NDRs to violating nets if this reduces the timing violations (e.g. double spacing rule for Xtalk or double width if R is dominant)
- IC Compiler will use all NDRs that are defined in the design (with define_routing_rule), unless certain rules are defined as dont_use using:

```
set_routing_rule_dont_use "rule1 rule2 ..."
```

6-37

NDRs are honored during ECO routes as well (e.g. route_eco)

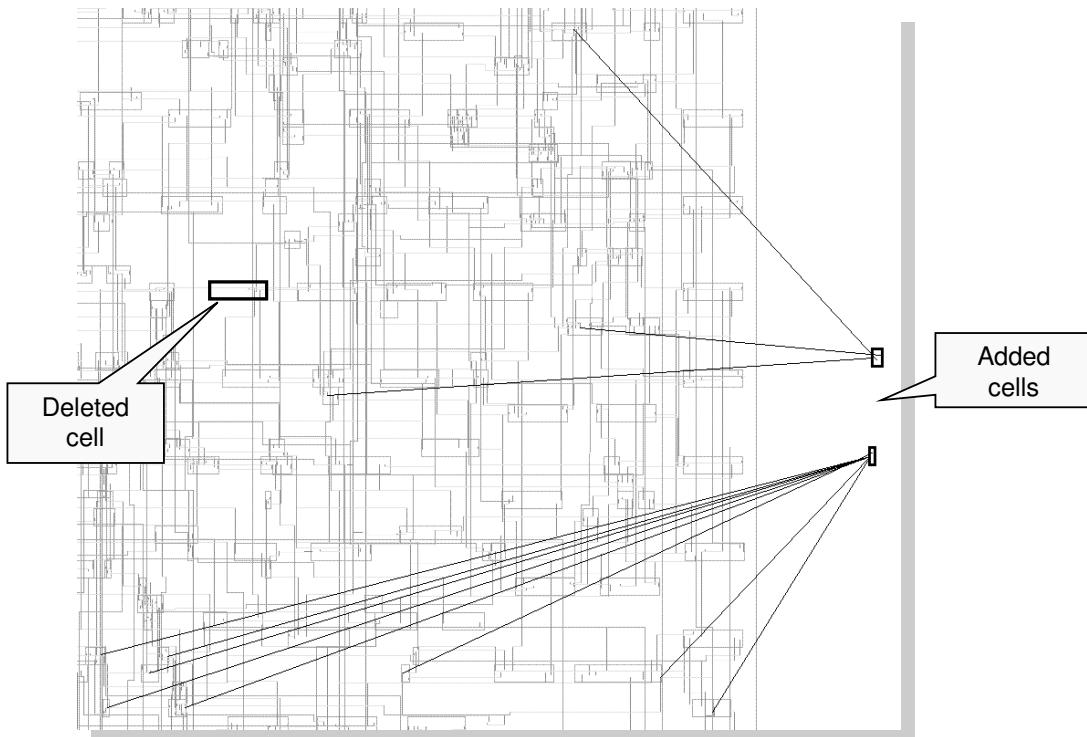
Wire Sizing at Work



- Critical wire has been made wider to solve a timing violation. Since R is reduced, it can benefit the timing. The wider spacing helps crosstalk.
- Use with care, since too many NDRs can make the design unrouteable

6-38

ECOs: Making Changes Late in the Flow

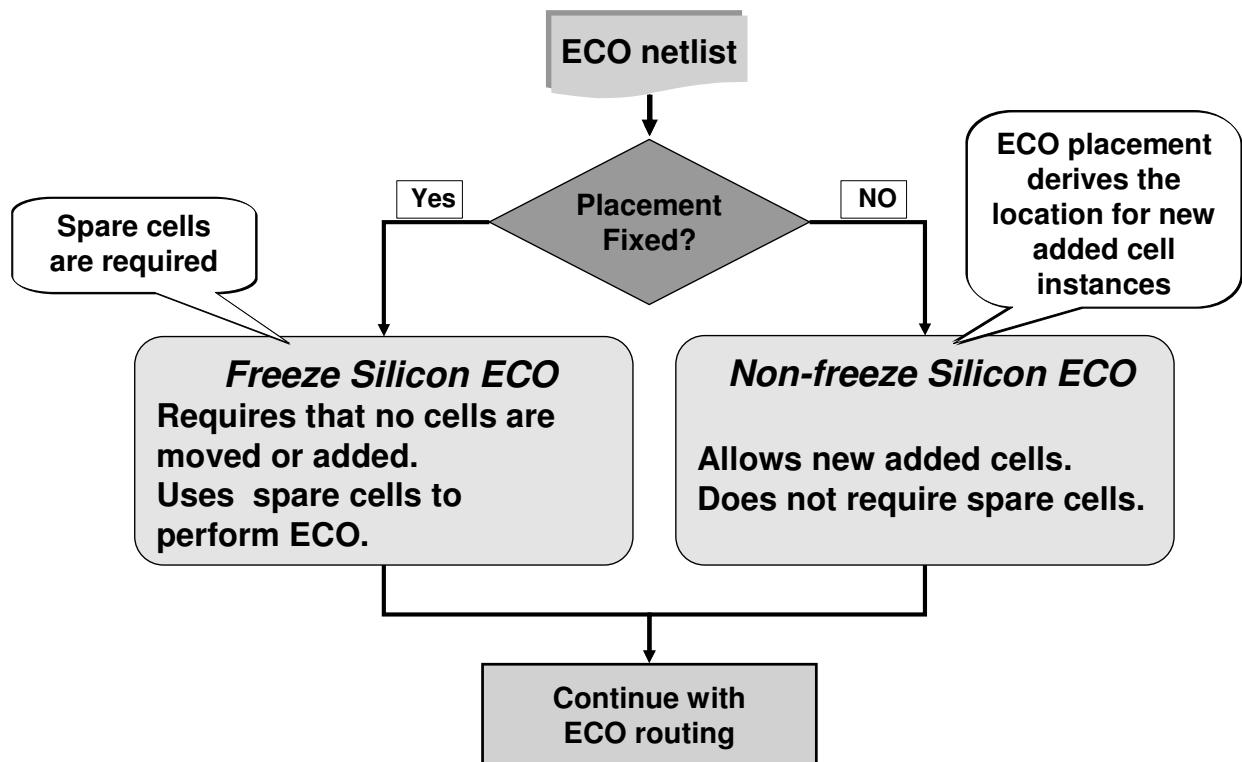


Functional changes occur late in the design cycle

6-39

“ECO” is an old term which stands for “engineering change order”. In the early days of circuit design, if a change needed to be made to a part of the design that was already defined or specified, you would fill out an “Engineering Change Order” form and have it signed off before making the change. Even though most companies no longer use ECO forms, the term is still used, and refers to design changes late, or near the end of the design phase.

The Two Types of ECO Flows



6-40

Functional ECO Flows

1. Non-Freeze silicon ECO

- Pre-tapeout, no restriction on placement or routing
- Minimal disturbances to the existing layout
- ECO cells are placed close to their optimal locations

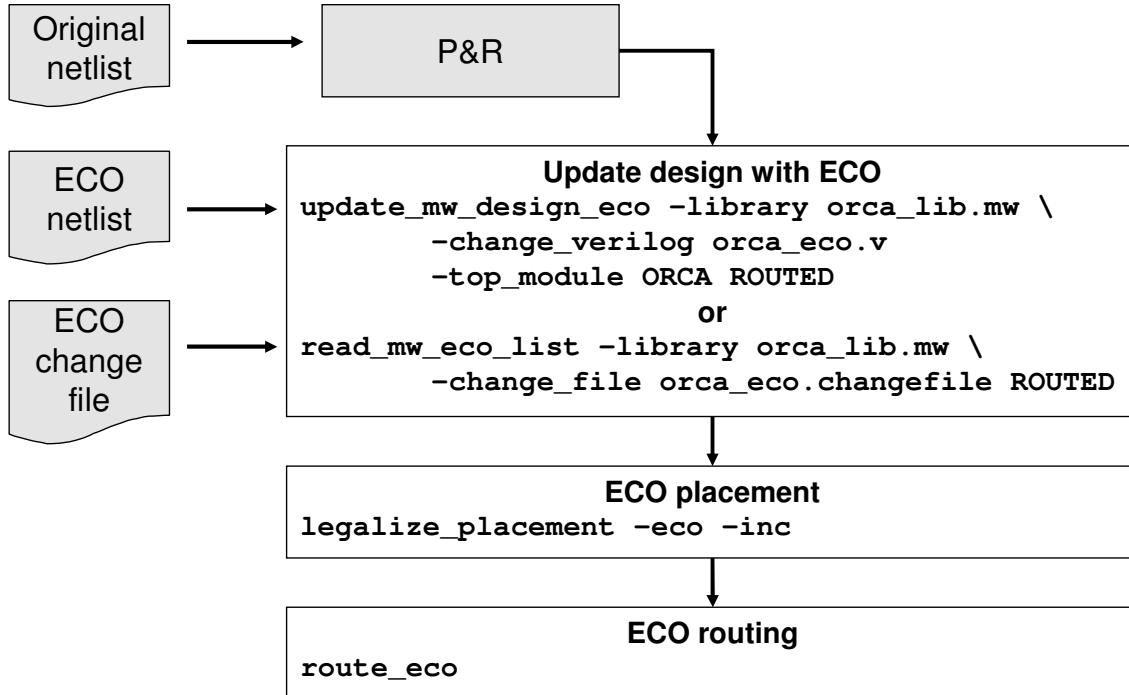
2. Freeze silicon ECO

- Post-tapeout, metal masks change only using previously inserted spare cells
- Cell placement remains unchanged
- ECO cells are mapped to spare cells that are closest to the optimal location
- Deleted cells become spare cells

6-41

Non-freeze Silicon ECO flow is also known as the “Unconstrained ECO flow”.

Non-Freeze Silicon ECO



6-42

`update_mw_design_eco`: This command compares the current version of the netlist with the updated version of the netlist. This command works on both hierarchical and flattened designs.

`read_mw_eco_list`: This command make engineering changes to the design based only on the information provided in the ECO change file.

When a cell is deleted form the netlist , this command deletes the cell from the design database

When a cell is added in the netlist , this command adds the cell in the design database

When a cell instance in the netlist is replaced with another cell (cell instance name being same) the tool replaces the cell in the design database.

`legalize_placement`: This command legally places cells added to a layout by the above `update_mw*` or `read_mw*` commands.

`route_eco`: Performs incremental routing for the new cells after running the `legalize_placement` command. Executes all three phase of routing: global route, track assign and detail route.

Hierarchical ECO Change File Example

```
+D Mon Oct 30 14:00:00 2006
+T IC Compiler
+C Invert DATA[3]
+A ecoengineer
+HN d3_spare
-HC net_DATA[3] DATA_iopad_3/I
+HI d3_spare_inv invbd7
+HC net_DATA[3] d3_spare_inv/I
+HC d3_spare d3_spare_inv/ZN
+HC d3_spare DATA_iopad_3/I
```

See SolvNet “IC Compiler ECO Flow Application Note”
<https://solvnet.synopsys.com/retrieve/018497.html>

6-43

Date and Time (+D) This is part of header information.

Syntax: **+D dayOfWeek Month Day Time Year**

Tool (+T) This is part of header information.

Syntax: **+T Tool**

Author (+A) This is part of header information.

Syntax: **+A AuthorLoginName**

Comments (+C) This is a comment line.

Syntax: **+C Comment**

Net Creation (+HN)

Syntax: **+HN hierPath/NetName**

Net Deletion (-HN)

Syntax: **-HN hierPath/NetName**

Port Net Connection (+HC)

Syntax: **+HC hierPath/NetName [childCellInstanceName/]PortInstanceMasterName**

Port Net Disconnection (-HC)

Syntax: **-HC hierPath/NetName [childCellInstanceName /]PortInstanceMasterName**

Instance Creation (+HI)

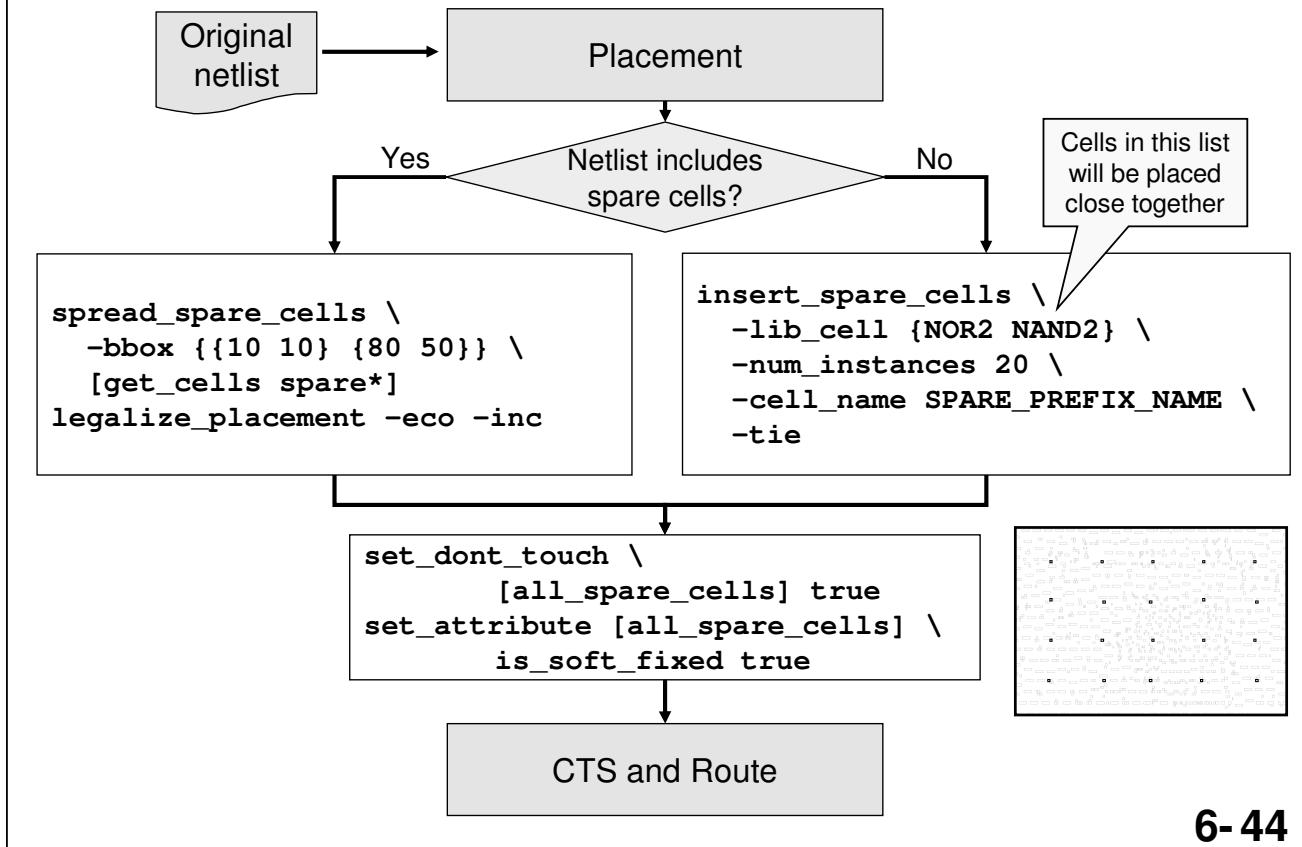
Syntax: **+HI [hierPath/]CellInstanceName CellInstanceMasterName [viewName]**

Instance Deletion (-HI)

Syntax: **-HI [hierPath/]CellInstanceName [CellInstanceMasterName] [CellInstanceType]**

See the Application Note “Hierarchical ECO By Change File” for complete details.

Inserting Spare Cells for Freeze Silicon ECO



6-44

Spare cells are extra gates (AND,NOR,XOR,INVERTER, etc) which are included in a layout but are not functionally connected and used during the initial design phase. If a functional change is required late in the design cycle, the layout can be modified (routing or metal mask changes only) to take advantage of these spare cells. The flow above describes how to add these spare cells in your layout, so that you can use them later, if needed, using the “freeze-silicon ECO flow”.

Spare cells are distributed inside the core area.

```

insert_spare_cells      # insert spare cells
  -lib_cell <lib_cell_list>          (ref cell name)
  -num_instances <number_of_instances> (num of instances: Value >= 0)
  -cell_name <cell_name>            (base name of spare cell)
  [-sub_design <sub_design_name>]   (name of sub_design)
  [-tie]                            (tie input pins to logic zero)

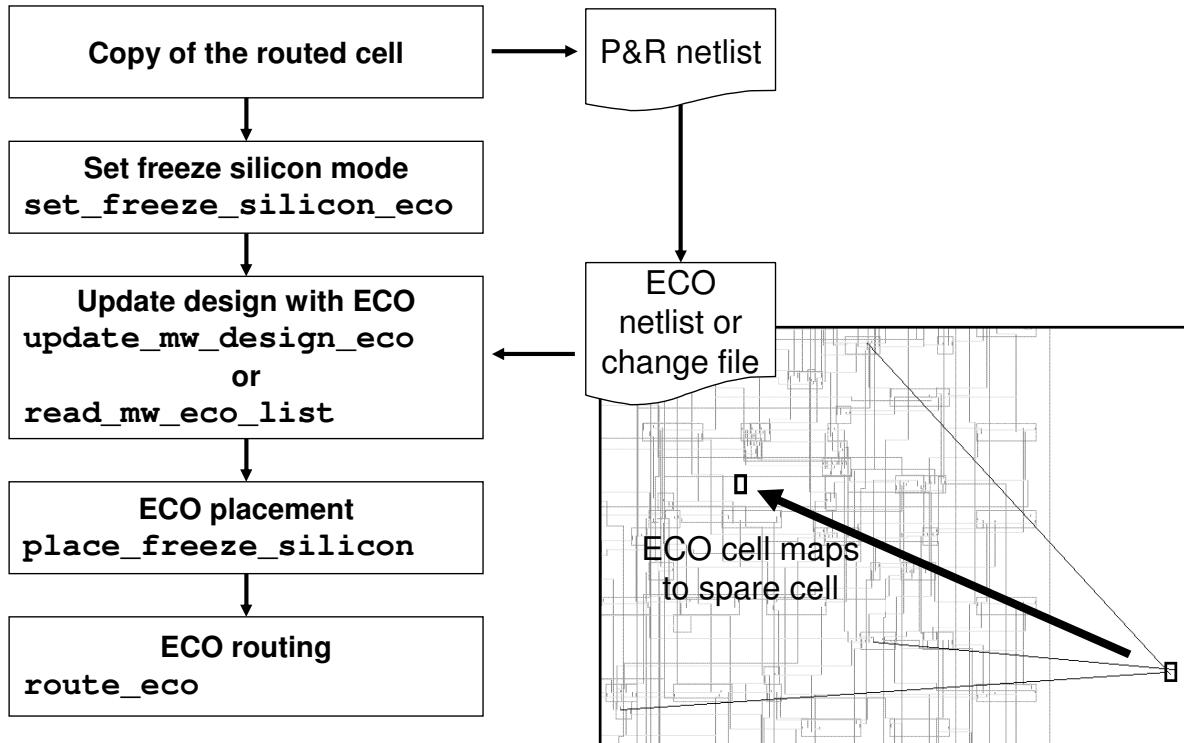
```

Protecting Spare Cell Placement

- Spare cells are `dont_touch` so IC Compiler doesn't remove the unconnected cells
- Set the spare cells to SOFT FIXED once the spare cells are distributed
 - Use `set_attribute` to set the spare cells to SOFT FIXED
- Why set the spare cells to SOFT FIXED?
 - Detailed placer may fail if there are too many fixed cells
 - The soft-fixed attribute prevents incremental coarse placement from moving spare cells
 - The soft fixed cells can still be moved slightly and legalized by CTS and Routing optimizations

6-45

Freeze Silicon ECO: Metal Change Only



6-46

If the “Freeze Silicon mode” is enabled , IC Compiler will not perform any cell changes to the design. Any changes are “swaps”, that means that the only changes are rewirings.

ECO Routing Example

The ECO route example below is for a minor ECO:
Only minimal routing changes are being allowed

```
route_eco      -reroute modified_nets_only  
                  -utilize_dangling_wires  
                  -freeze_routing_on_layer layer  
                  -freeze_vias_on_frozen_metal
```

Modify these options if the router
fails to fix all violations.

6-47

Zroute: Synopsys' New Routing Technology

Architected From The Ground Up For New Challenges

Available
2008.09

IC Compiler
Zroute

1

State of the art routing
technology

2

Concurrent DFM optimizations

3

Multi-threaded throughout



10X Speed-up, higher QoR, better manufacturability

6-48

Synopsys introduced IC Compiler Zroute technology on May/27. IC Compiler customers will have a choice to use the current router or Zroute based on their needs. Zroute co-exists with current router and requires a single license for 4 cores.

Compared to traditional routers, Zroute delivers a 10X speed increase, higher QoR and better DFM.

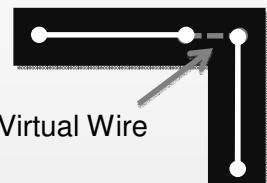
Zroute was built from scratch to address new challenges for routing.

It is part of IC Compiler and there are 3 key things that differentiate it from traditional routers:

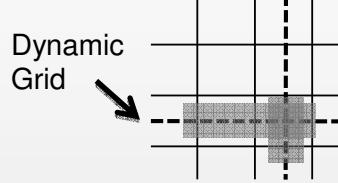
- 1.State of the art routing technology.
- 2.Concurrent DFM optimizations.
- 3.Multi-threaded throughout.

1. State of the Art Routing Technology

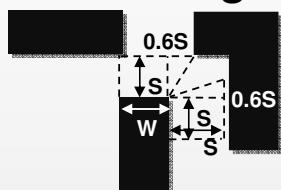
Realistic Connectivity



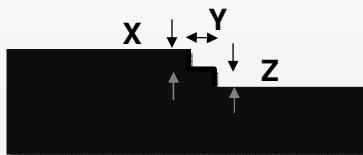
Dynamic Maze Grid



Advanced Design Rules



Polygon Manager



Faster runtimes, improved design closure

6-49

Zroute includes state of the art routing technology with the latest in routing research and routing algorithms.

This slide highlights some of these technologies:

1. Realistic connectivity

Traditional routers model the center lines of wires, rather than the complete wire.
For two lines to connect, the center lines have to connect.

Zroute's connectivity model uses a realistic connectivity model.

As long as the rectangles touch, Zroute knows there will be electrical connectivity.
This makes the router very flexible in the way that it manipulates shapes.

2. Dynamic maze grid

This can be thought of as a grid that can be changed on the fly in specific instances.
It allows the router to go off-grid to connect pins, while retaining the speed advantages of gridded routers.
In other words Zroute is a gridded router that can behave as gridless when needed.

3. Advanced design rules

Zroute is architected to efficiently handle advanced design rules for 45 nm and below.
In addition to hard rules imposed, Zroute can handle soft, or recommended rules.

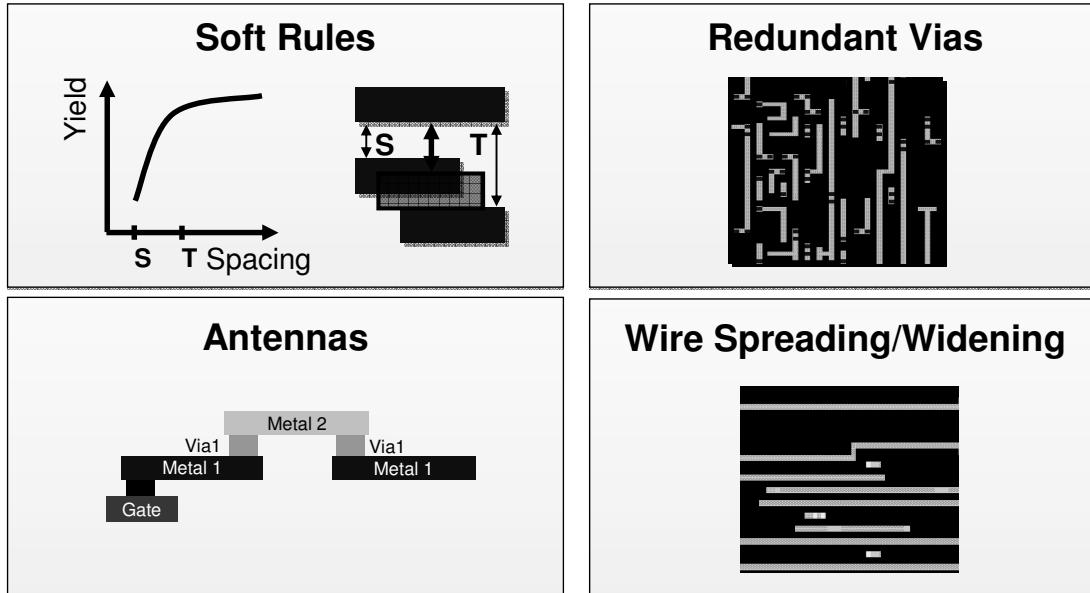
Example shows: a “dense end of line” configuration

4. Polygon manager

Zroute has a built-in polygon manager.
While it routes using rectangles, it understands that design rule checks (DRCs) are aimed at polygons, and it can recognize polygons.
Routers that can't recognize polygons have to decompose everything into rectangles, adding more processing time.

The technologies described here as well as others in Zroute achieve faster runtimes and improved design closure.

2. Concurrent DFM Optimizations



Faster runtimes, improved manufacturability

6-50

Another Zroute advantage compared to traditional routers is concurrent DFM optimizations.

By considering soft rules, antennas, vias, and wire spreading during routing rather than as a post-processing step, Zroute achieves better manufacturability.

Here are some examples:

1. Soft rules

Zroute supports soft spacing rules as well as user defined soft rules for better manufacturability and litho-friendly routing.

2. Antennas

Routers supported antenna rules since 90nm.

The difference with Zroute is that you can set the antenna rules up front, and the router will automatically take care of them without any extra checking and fixing steps.

3-4. Redundant vias, wire spreading/widening

By considering vias, and wire spreading during routing rather than as a post-processing step, Zroute achieves better manufacturability.

Early partners are reporting 10 to 15 percent fewer vias overall, and 30 to 50 percent fewer single vias. Reducing single vias results in a higher yield.

The technologies described here as well as others in Zroute achieve faster runtimes and improved manufacturability.

3. Multi-threaded Throughout

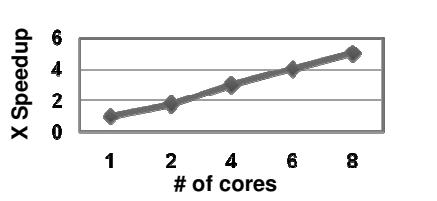
All Routing Steps

Global Routing

Track Assignment

Detail Route

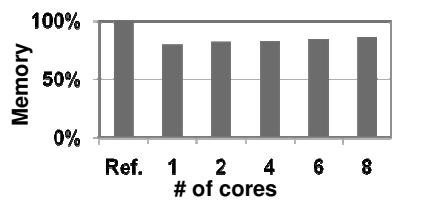
Scalability



Transparent

Native
Ready to go
Simple set-up

Larger Capacity



Faster runtimes, near linear scalability

6-51

Multi-threading is the 3rd key thing that differentiates Zroute.

New software has to be architected for multi-threading, and it is not a trivial thing to do.

1. All routing steps

In Zroute all routing steps – including global routing, track assignment, and detailed routing – are multi-threaded to take advantage of the multicore CPUs.

2. Transparent

Multi-threading is transparent to the user and the only thing that needs to be done is to set the # of threads up front.

3. Scalability

Zroute delivers near linear scalability as the number of cores increases. We will take a closer look at this in the next slide.

4. Capacity

Zroute delivers 20% smaller memory footprint than traditional routers.

Note that there is minimal memory overhead as the number of cores increases.

With multi-threading users see faster runtimes that scale as the number of cores increases.

10X Speed-Up On Mainstream Hardware

3.5X Faster – Single-threaded

“x”

3X Speedup – Multi-threaded on Quad-Core

10X Speed Up



“... achieved a **10x boost in performance** when we used it with a quad-core platform.”

6-52

Combining the latest in routing algorithms with multi-threading, achieves a 10X speed up in routing.

3-4X single-threaded: attributed to the changes in the core routing technology.

3X multi-threaded: additional boost with 4 threads (4-core platforms are mainstream today).

3.5X times 3X = ~10X for quad-core mainstream platforms.

This is how Zroute delivers a 10X improvement in routing runtime.

Zroute Is GA in IC Compiler 2008.09!

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

Available also in
2007.12-SP5-1

Multiple on-going
tape-outs

Multiple test chips
at 32nm

SolvNet®

Zroute Technology in IC Compiler

> TRAINING
Technical Tutorial
Labs
Evaluation Guide

> DOCUMENTATION
User Guide
Translator Guide
FAQs

> RELEASE INFORMATION
Feature Release Guide
45-nm Design Rules
65-nm Design Rules
90-nm Design Rules

> RELATED INFORMATION
GR Congestion Map

IC COMPILER
Zroute

Zroute is IC Compiler's brand new, super-fast, DFM-aware router. It was developed from the ground up to take advantage of the newest multi-core microprocessor architectures and to solve emerging DFM challenges in design. Listen to R&D talk about Zroute at [Synopsys](#), read some recent news articles below, or get started with Zroute, just by navigating the links to the left.

Are You Ready for Zroute?

Take [this](#) readiness quiz to find out.

Technical Papers

6-53

Zroute Users Tell The Story ...

Tape-out



“... 2-3.4X faster single-threaded
3X scalability with 4 threads ...”

10/17/08



“... 70% fewer narrow jogs,
30% fewer notches ...”



9/22/08



“... tapes out home networking device
using new Zroute technology ...”

9/15/08

Tape-out



“... achieved a 10X boost in performance
when we used it with a quad-core platform.”

5/27/08



“... routing runtimes cut significantly by
multi-threading support ...”

5/19/08

6-54

Routing & Crosstalk Summary

- Specify common route options before starting route activity for clocks and signals
- Route clock nets and special net groups first, then signal routes for the remainder of the design
- Perform Post Route Optimizations to fix max cap/trans and timing violations
- Perform crosstalk correction during the routing flow using logic, placement and routing optimization
- Implement functional ECOs with minimal changes



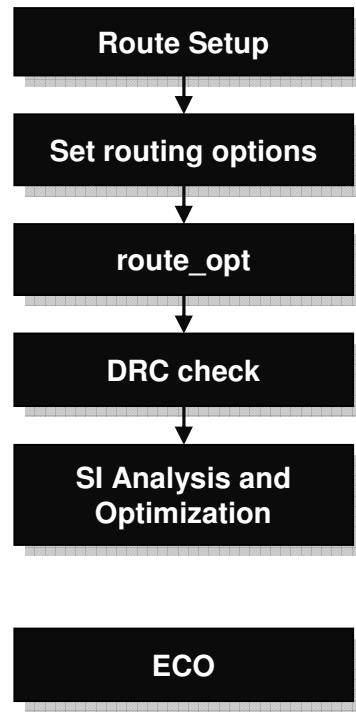
6-55

Lab 6a: Routing & Crosstalk, Lab 6b: ECO



80 minutes

- Perform routing and optimizations
- Perform crosstalk optimization
- Perform freeze silicon ECO



6-56

Agenda

**DAY
3**

5 Multi Scenario Optimization



6 Routing and Crosstalk



7 Chip Finishing and DFM



CS Customer Support

Unit Objectives

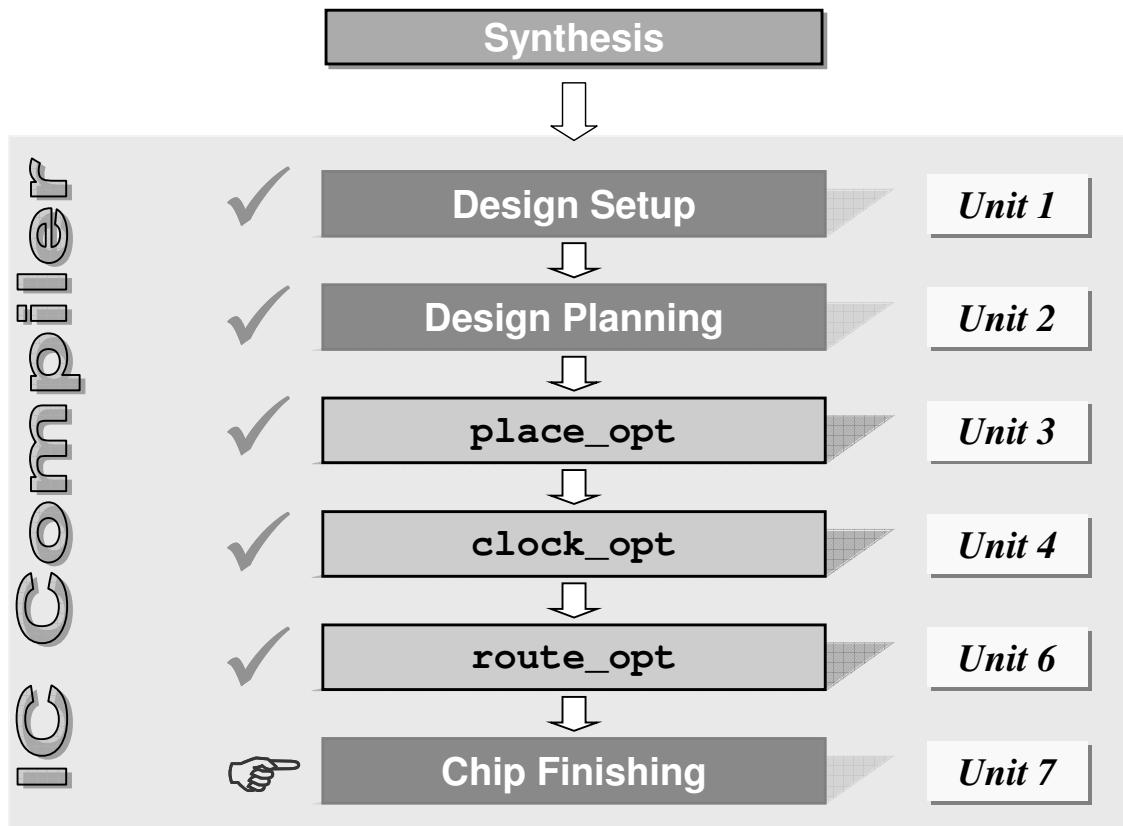


After completing this unit, you should be able to:

- **Perform key chip finishing and design for manufacturing steps required after the signal routing is complete:**
 - Fix antenna violations
 - Modify the routing patterns to make them more resistant to defects
 - Add redundant contacts
 - Perform metal filling and slotting
 - Insert filler cells
 - Run DRC and LVS operations (Lab)

7-2

IC Compiler Flow



7-3

Design Status, Completion of Routing Phase

- Placement - completed
- CTS – completed
- Power and ground nets – routed
- Signal and clock nets – routed
- Signal Integrity issues – fixed/acceptable
- Calculated timing – acceptable ($\geq 0\text{ns}$ slack)
- Logical DRC – max cap/transition – no violations
- Physical DRC – no violations

7-4

Chip Finishing Flow

**Post-Route: Timing &
DRC clean design**

Antenna Fixing

Wire Spreading

Redundant Via Insertion

Filler Cell Insertion

Metal Fill Insertion

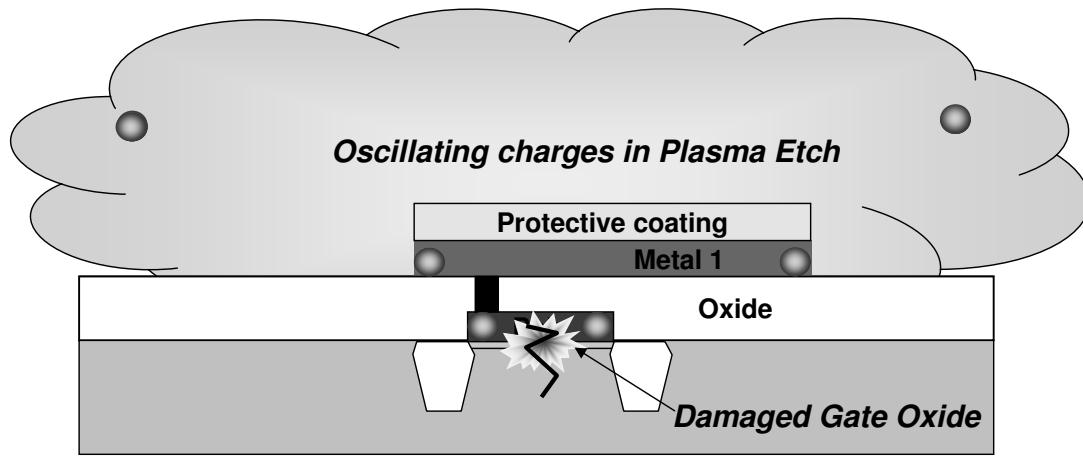
Metal Slotting

7-5

Problem: Gate Oxide Integrity

Antenna Critical Area Redundant Via Filler Cell Metal Fill Metal Slotting

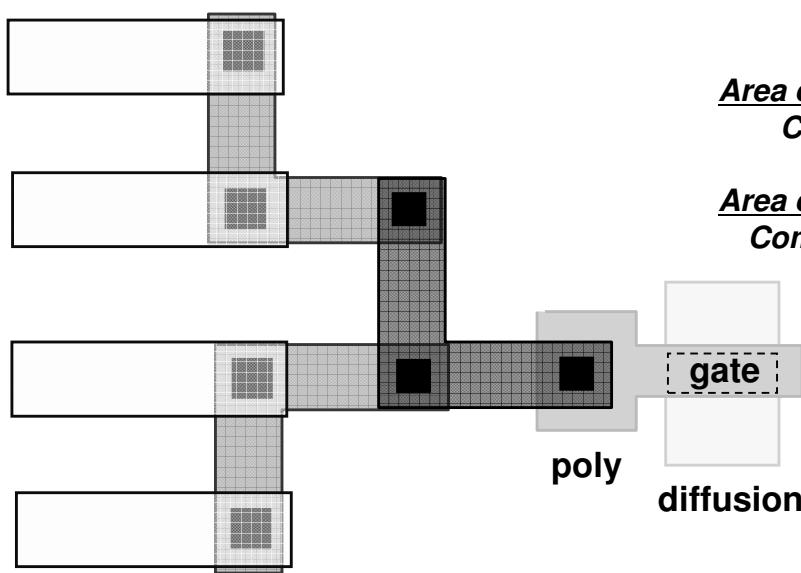
- Metal wires (antennae) placed in an EM field generate voltage gradients
- During the metal etch stage, strong EM fields are used to ionize the plasma etchant
- Resultant voltage gradients at MOSFET gates can damage the thin oxide



7-6

Antenna Rules

- As total area (length) of wire increases during processing, the voltage stressing the gate oxide increases
- Antenna rules define acceptable total area of wires



Antenna Ratios:

Area of Metal Connected to Gate
Combined Area of Gate
Or
Area of Metal Connected to Gate
Combined Perimeter of Gate

7-7

The ASIC vendor will specify antenna design rules in terms of allowable ratios of antenna metal to gate oxide.

Basic Antenna rules are:

$$(\text{antenna-area}) / (\text{gate-area}) < (\text{max-antenna-ratio})$$

Gate-area:

Boolean AND (or intersection) of the ‘poly’ and the ‘diffusion’ layers

Recognized as the gate area of the transistors by essentially all foundries

Antenna-area:

Amount of metal area attached to the input pin

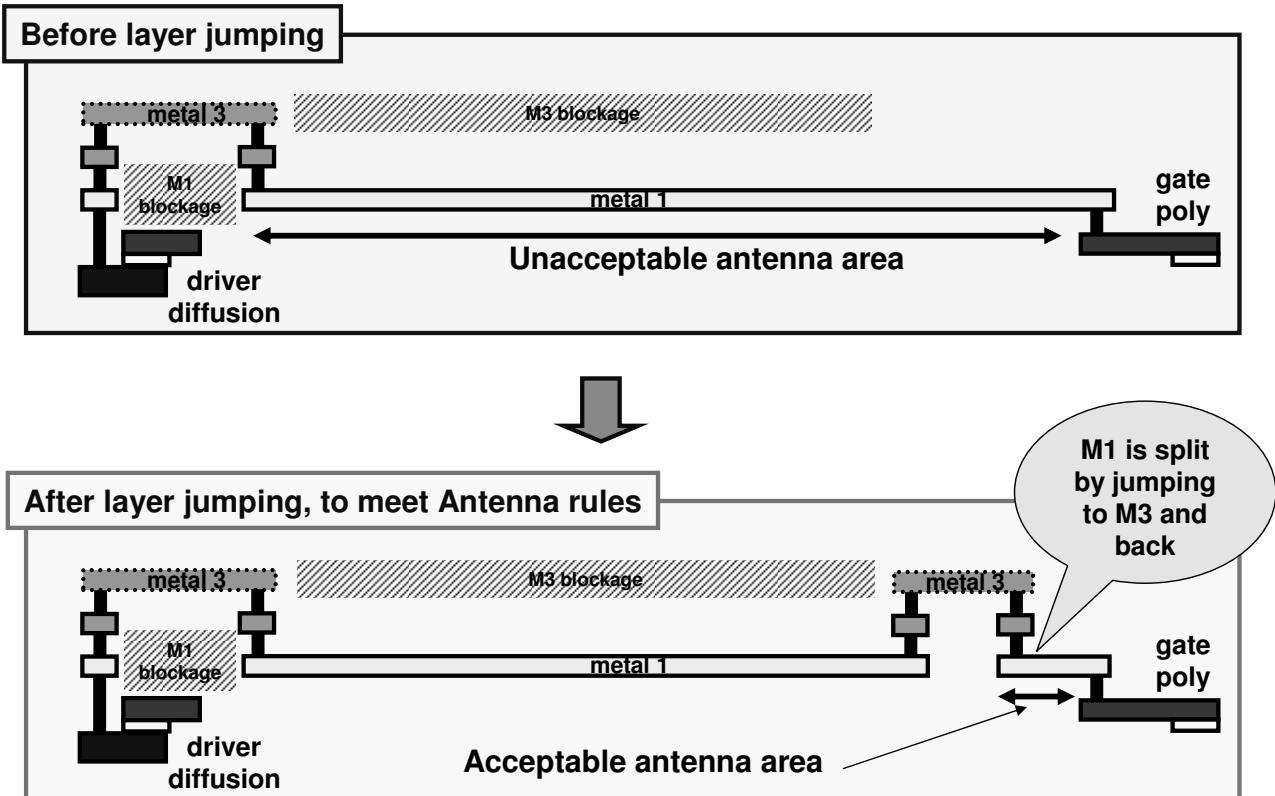
Calculation method varies for different processes

Max-antenna-ratio:

Represents max allowed ratio of antenna area to gate area

Calculation method varies for different processes

Solution 1: Splitting Metal or Layer Jumping



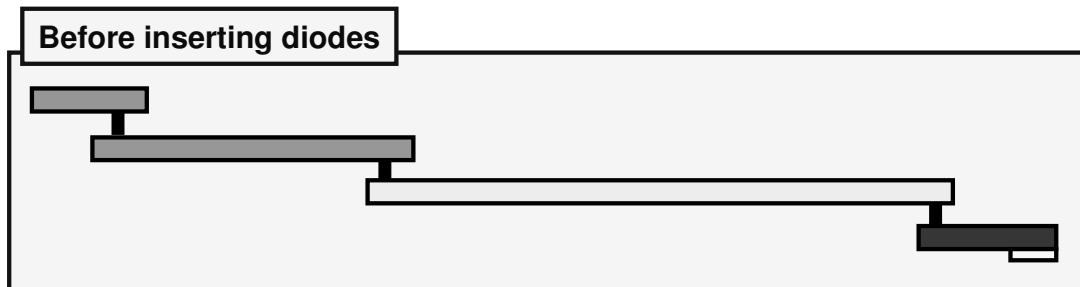
7-8

Until the top metal 3 link is in place, the metal 1 wire can generate large voltage gradients that may damage the gate oxide. This method of fixing antenna rule violations uses jumps in metal layers to keep the total length of lower level metal traces that are directly connected to polysilicon gates below maximum. Once the top metal 'link' is in place and the polysilicon gate is connected to its driver, it is relatively safe. The pn junction of the source/drain of the driver will help shunt large voltage swings – it acts like a reverse-biased diode.

This is the preferred method for fixing antenna problems. Antenna violations that cannot be fixed by this method must be provided with special shunting diodes. These diodes use up additional silicon placement and metal routing resources, which may be challenging – diode insertion is therefore the second choice, over layer jumping, to address antenna violations.

Run an Search and Repair to fix antenna violations. The search & repair engine will reroute antenna violators adding multiple jumps to the long traces.

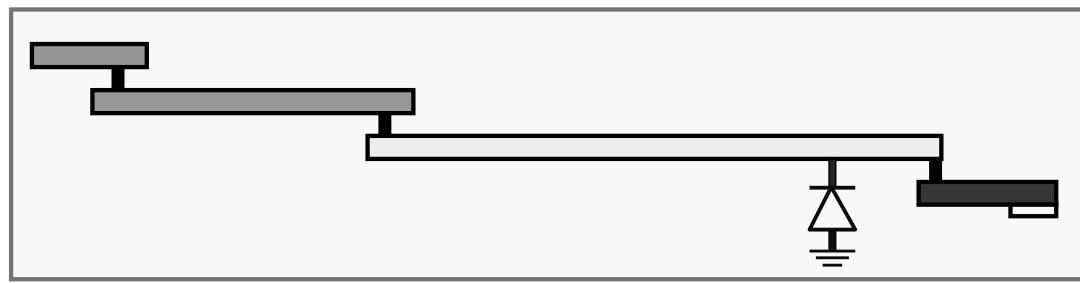
Solution 2: Inserting Diodes



Before inserting diodes



Diode Inhibits large voltage swings on metal tracks



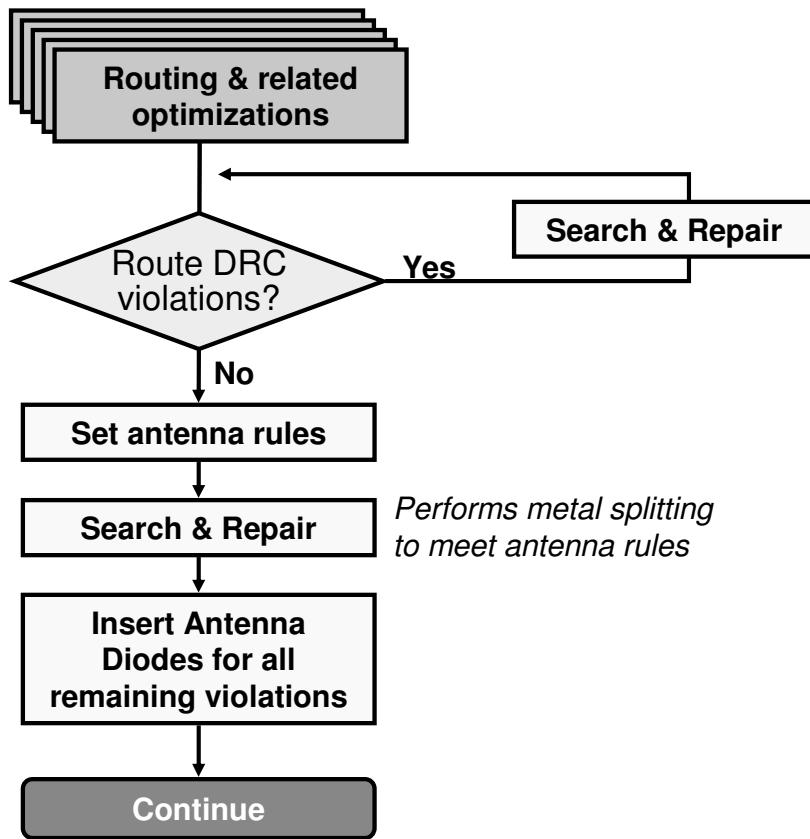
During etch phase, the diode clamps the voltage swings.

7-9

If the antenna can not be fixed with metal jumping, add an antenna diode to shunt the voltage. The diode acts like the pn-junction of a driving gate to clamp the voltage swings on the metal. A special antenna diode CEL is usually provided by the fab or ASIC vendor.

During normal operation, these diodes are always reverse biased and have modest effects on signal performance.

Antenna Fixing Flow



Command details in lab!

7-10

IC Compiler can fix antenna violations by performing “Search & Repair” and/or “Insert Antenna Diode”. For a less congested design, use “Search & Repair” before “Insert Antenna Diode” to fix antenna violations and to minimize changes to the design.

The goal is to fix as many antenna violations as possible by layer-jumping with “Search & Repair”, and then insert diodes to fix the remaining antenna violations.

Run Hercules as final verification.

Antenna: Misc

- **Antenna rules (once loaded) are honored by any Search & Repair run, even during detailed route**
 - It is NOT recommended to turn on antenna rules during detailed route's S&R
 - ◆ The router may try to fix antennae and end up in irresolvable DRC violations
- **Use diodes to fix antenna that are not fixable by S&R**

```
insert_diode -prefix DFM_DIODE_
```

- Diodes are auto select by default
- To specify diodes, use:

```
insert_diode -no_auto_cell_selection \
              -diode_cells <collection_of_diode_cells>
```

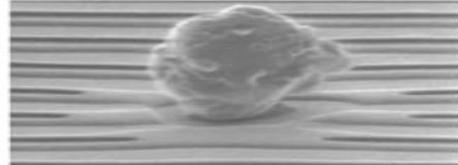
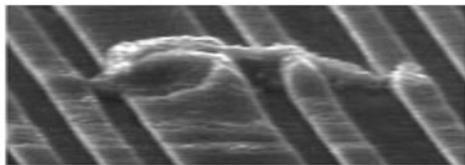
7-11

Random Particle Defects

Antenna Critical Area Redundant Via Filler Cell Metal Fill Metal Slotting

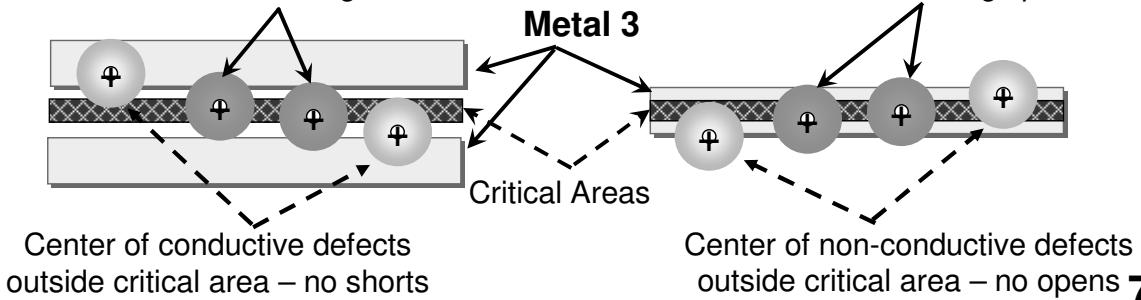
■ Random particle defects during manufacturing may cause *shorts* or *opens* during the fabrication process

- Wires at minimum spacing are most susceptible to shorts
- Minimum-width wires are most susceptible to opens



Center of conductive defects within critical area – causing *shorts*

Center of non-conductive defects within critical area – causing *opens*



7-12

This is a process and fabrication-related defect that's generally random with a level of clustering, depending on the defect sources.

Random particle defects (which are usually called extra/missing material random defects or short/open random defects), occur because of contamination during the fabrication process.

Particles fall on the chip and result in two adjacent lines causes shorting or a line breaking .

Definition of *Critical Area*

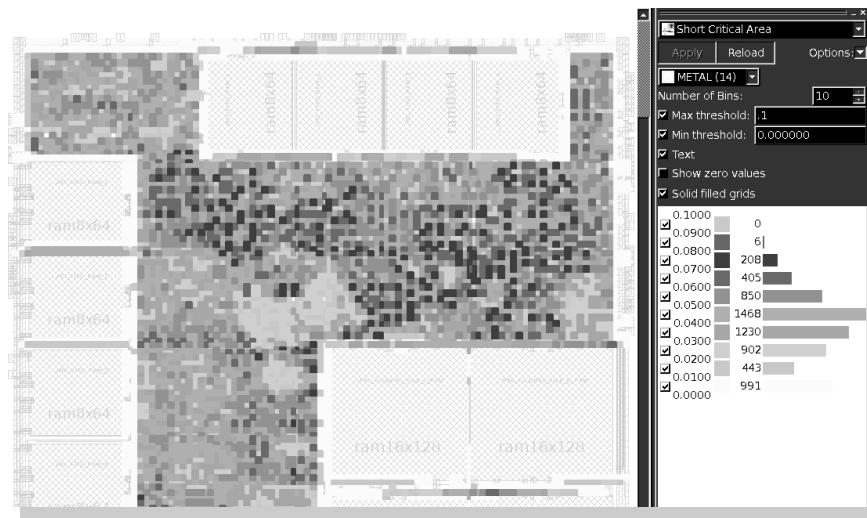
A *critical area* is a region where circuit failure will occur (yield loss) if the center of a random defect falls in it

- Critical area regions varies with defect size
- For a particular layout, the larger the defect size, the larger the critical area
- Larger defects are statistically less likely

Reporting the Critical Area

- Generates both text and graphic report

```
report_critical_area
    -particle_distr_func_file <file>
    -input_layers {m2 m3 m4}
    -fault_type {short|open}
```



7-13

GUI: Finishing → Short Critical Area Map and Finishing → Open Critical Area Map

Results from critical area analysis is in the file “output_heatmap”. Example:

Total critical area on each layer

Layer	Critical Area (mm ²)	Critical Area (% of Chip Area)
m1	159692.323	20.876
m2	41282.836	5.397
m3	102339.392	13.378
.....		

Critical area within each window

Layer: m2

window (x1, y1) (x2, y2)	Ratio	Area
(0, 0) (5.36, 5)	0	0
(5.36, 0) (11.12, 5)	0.0478	1.377
(11.12, 0) (16.88, 5)	0.0253	0.728
(16.88, 0) (22.64, 5)	0.0135	0.388
.....		

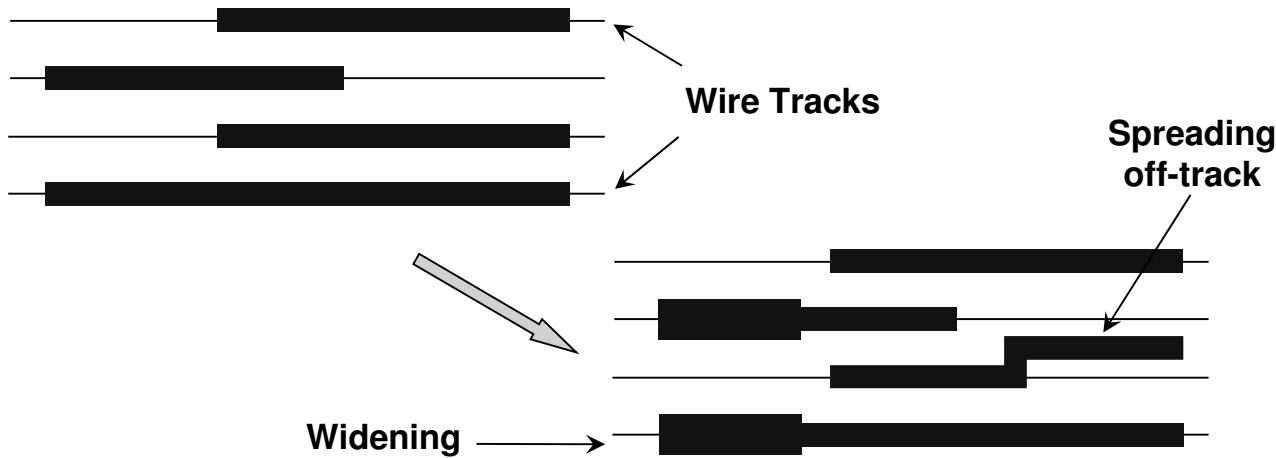
Solution: Wire Spreading + Widening

```
route_spreadwires + route_widen_wire
```

■ Spread wires to reduce *short* critical area

- Push routes off-track by $\frac{1}{2}$ pitch
- Will not push “frozen” nets

■ Widen wires to reduce *open* critical area



7-14

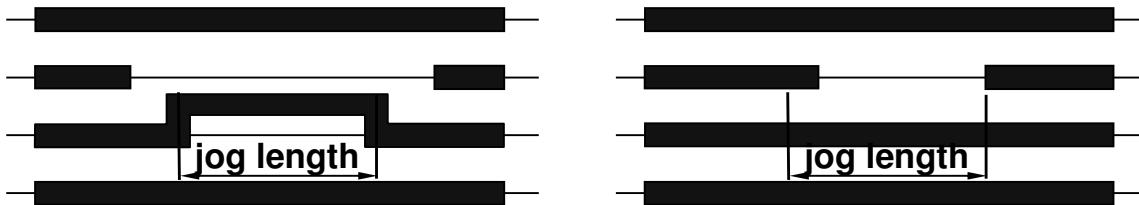
Use the following command to perform wire spreading at post-detail route stage :

```
route_spreadwires      #The following options are optional
-min_jog_length (default: 2)
-timing_driven (default: off)
#Preserve timing of critical nets meeting setup slack criteria
-setup_slack_threshold(default: 0.0)
#Do not spread net with a worse setup slack than this threshold
-search_repair_loop (default: 10)
```

```
route_widen_wire
-search_repair_loop (default: 10)
-nonuniform_widening
-timing_driven (default: off)
-setup_slack_threshold (default: 0.0)
-hold_slack_threshold (default: 0.0)
```

Controlling Minimum Jog Length

- Pushing wires off-track always creates a jog and increases wire length
- Use '`-min_jog_length`' option to control the minimum jog length (default: 2 pitches)
 - Will not push a wire unless the available space is larger than '`-min_jog_length`'



7-15

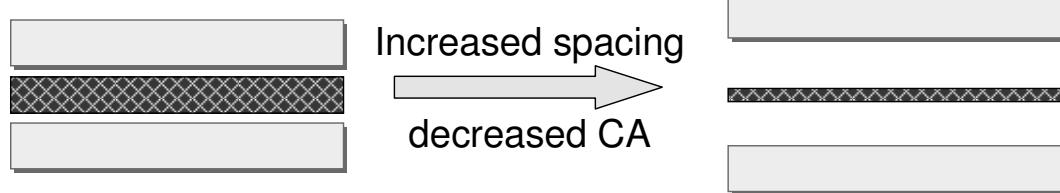
Proactive: Density-Driven During GR and TA

- The density-driven mode spreads wires during global route and track assignment to achieve more uniform wire distribution and fewer minimum spacing

```
set groute_densityDriven 1  
set trackAssign_densityDriven 1
```

- Density-driven mode is off by default but it is enabled automatically when:

- Timing-Driven global route / track assign is on, and/or
- Crosstalk prevention is on



7-16

```
icc_shell> set groute _densityDriven 1  
;; range [-1,2], default=-1;  
;; 0: turn off density driven mode,  
;; 1: turn on density driven mode.  
;; 2: turn on high effort density driven mode.  
;; -1: program to decide whether to use density driven mode.
```

```
icc_shell> set trackAssign_densityDriven 1  
;; range [-1,2], default=-1;  
;; 0: do not worry about spreading,  
;; 1: try to spread wires looking at 1 extra track,  
;; 2: should only be set when design is lower than x utilization, this will use more CPU. Misuse this when design  
is not sparse can cause negative impact,  
;; -1: try to spread wires looking at 1 extra track if timing or xtalk is on.
```

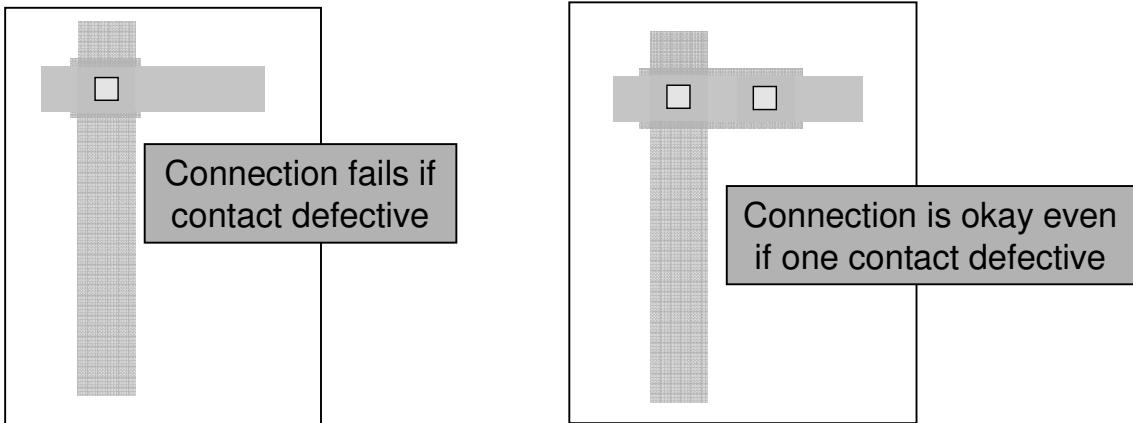
Voids in Vias during Manufacturing

Antenna Critical Area Redundant Via Filler Cell Metal Fill Metal Slotting

■ **Voids in vias is a serious issue in manufacturing**

■ **Two solutions are available:**

- Reduce via count: Via optimization techniques are employed in `route_opt`
- Add backup vias: known as redundant vias



7-17

Via Control Through Tcl Variables

- **Timing driven mode:**

```
set droute_optViaTimingDriven 1           (default 0)
set droute_optViaSetupSlackThreshold 0.05 (default -0.1)
set droute_optViaHoldTimeThreshold 0.05   (default 0.0)
```

- **To report excluded nets based on timing thresholds**

```
set droute_optViaReportExcludedNets 1      (default: 0)
```

7-18

droute_optViaTimingDriven: The variable specifies whether to preserve timing for critical nets with timing violations. If the value is set to 0, the router does contact optimization for all nets. If the value is set to 1, the router tries to preserve timing for critical nets by not doing contact optimization on those nets.

droute_optViaSetupSlackThreshold: The variable specifies the setup slack threshold for contact optimization. If the value is set to N, the router tries to preserve setup time by not inserting redundant vias for the nets whose setup slack is worse than N.

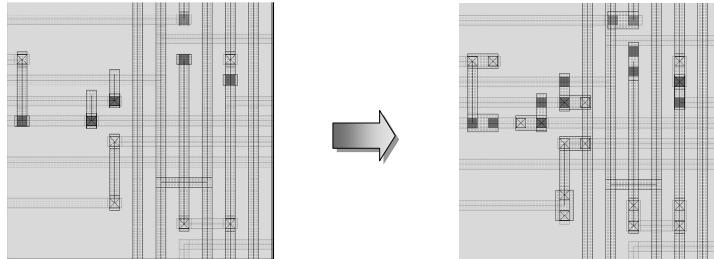
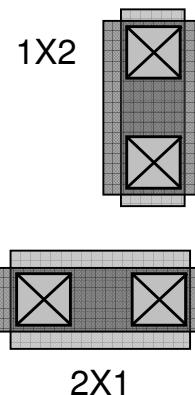
droute_optViaHoldTimeThreshold: The variable specifies the hold time threshold for contact optimization. If the value is set to N, the router tries to preserve hold time by not inserting redundant vias for the nets whose hold time is worse than N.

Insert Redundant Vias

■ Replaces single vias with multiple vias on all nets

- Excludes timing critical nets identified by the thresholds

```
insert_redundant_vias
    -from_via "via12 via23 via34"
    -to_via   "viaF12 viaF23 viaF34"
    -to_via_x_size "1 1 1"
    -to_via_y_size "2 2 2"
    -via_array_no_swap
    -optimize_level 1
    -auto_mode (preview | insert)
    -num_cpu N
```



7-19

Run the `insert_redundant_vias` command on a DRC clean design. Routes should not show new mask design rule violations.

```
insert_redundant_vias
    -from_via list_of_from_vias ;# Specifies a list of via names to be replaced.
    -to_via list_of_to_vias ;# Specifies a list of the corresponding names of the new vias to create.
    The list is usually the same as the corresponding -from_via name. If the list is different, the layer definition of the two vias must be the same.
    -to_via_x_size list_of_x-sizes ;# Specifies the list of the x-sizes for the newly created vias. If this option is not specified, the default value of 1 is used for x-sizes.
    -to_via_y_size list_of_y-sizes ;# Specifies the list of the y-sizes for the newly created vias. If this option is not specified, the default value of 1 is used for y-sizes. For a via, either x-size or y-size must have the value equal to 1.
    -via_array_no_swap (default false) ;# Specifies not to swap row and column of via arrays.
    In other words the 1 x N and N x 1 via arrays are not considered equivalent. By default, row and column of via arrays may be swapped.
    -optimize_level int" ;# Determines how aggressively to try different positions for line via array insertion. By default, the optimize_level is 0, which will try less positions. The value of 1 will try more positions.
    -auto_mode preview | insert ;# Specifies the mode in which to automatically generate via lists. If you specify preview, all vias defined in the technology file are listed, but no redundant vias are inserted. If you specify insert, redundant vias that are defined as default vias in the technology file are inserted. This option is mutually exclusive with the -from_via, -to_via, -to_via_x_size, and -to_via_y_size options.
```

Reporting Redundant Via Count

report_design -physical

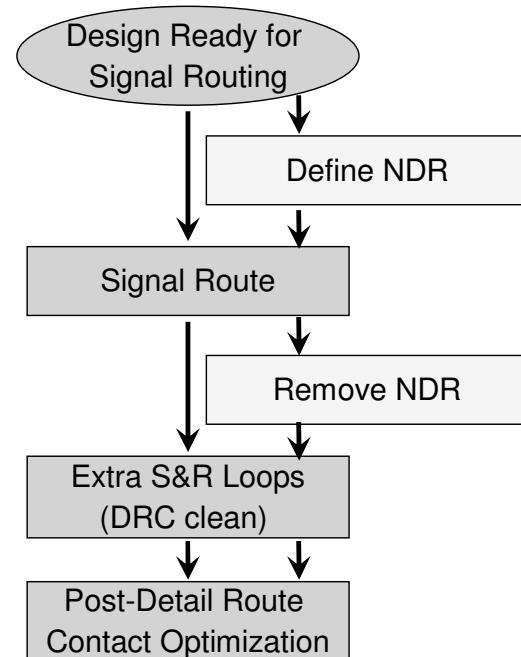
Mask Name	Contact Code	Number Of Contacts	Percentage
vial	VIA12A(1)	6578	14.7
vial	VIA12B(2)	26784	59.9
vial	VIA12f(9)	56	0.125
vial_1x2	VIA12A(1)	2982	6.67
vial_2x1	VIA12A(1)	8302	18.6
Default via for layer vial:		74.8%	
Yield-optmized via for layer vial:		25.2%	
via2	VIA23(3)	14334	25.2
via2_1x2	VIA23(3)	17544	30.8
via2_2x1	VIA23(3)	25039	44
Default via for layer via2:		25.2%	
Yield-optmized via for layer via2:		74.8%	
.....			
Double Via rate for all layers:		58.9%	
=====			
Total Number of Contacts:		120119	

7-20

Redundant Via Methodologies

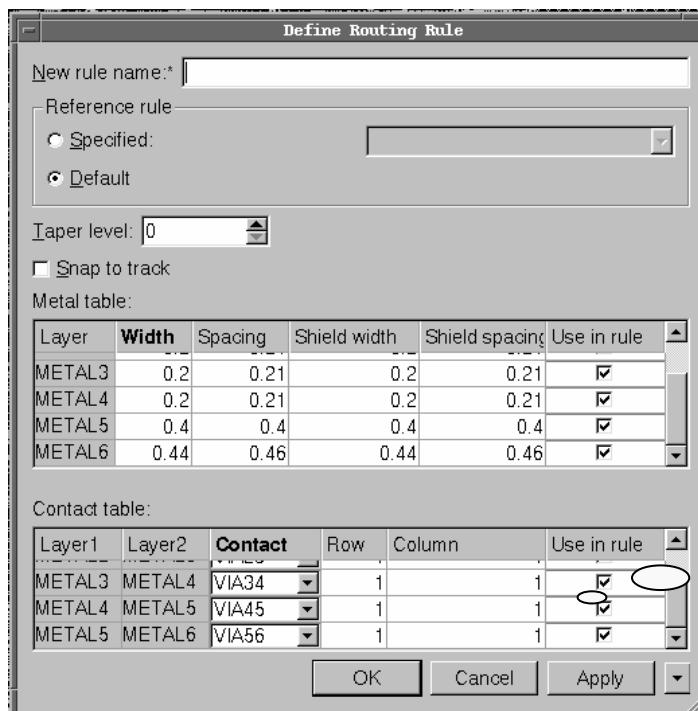
- IC Compiler provides both a recommended and an optional methodology for redundant via insertion:

- In the recommended methodology, redundant via is inserted at post-detail route
 - ◆ most of the time, more than 90% redundant via rate
- In the optional method, redundant vias are inserted during signal routing
 - ◆ increases redundant via rate



7-21

NDR = Non-Default Routing rule



Route → Routing Setup → Define Routing Rule

Use this area to define an NDR rule with multiple vias

Why Filler Cell Insertion?



- For better yield, density of the chip needs to be uniform
- Some placement sites remain empty on some rows
 - ICC can fill such empty sites with standard cells

7-22

Insert Cells to Fill Unused Placement Sites

- Use `insert_pad_filler` to insert IO cell fillers
- Use `insert_stdcell_filler` to insert std cell fillers
 - nwell/pwell structures
 - ◆ Automatically completes nwell/pwell structures by inserting well and tap filler cells (`insert_well_filler`)
 - ◆ Ensures that tap cells are placed to connect wells to rails at regular distances

```
insert_stdcell_filler \
    -cell_with_metal "fillCap64 fillCap32" \
    -connect_to_power VDD -connect_to_ground VSS
insert_stdcell_filler \
    -cell_without_metal "fill64 fill32"
    -connect_to_power VDD -connect_to_ground VSS
```

7-23

Accepts two lists of filler cells: with/without metal

Cells with metal are inserted only if no DRC violations result

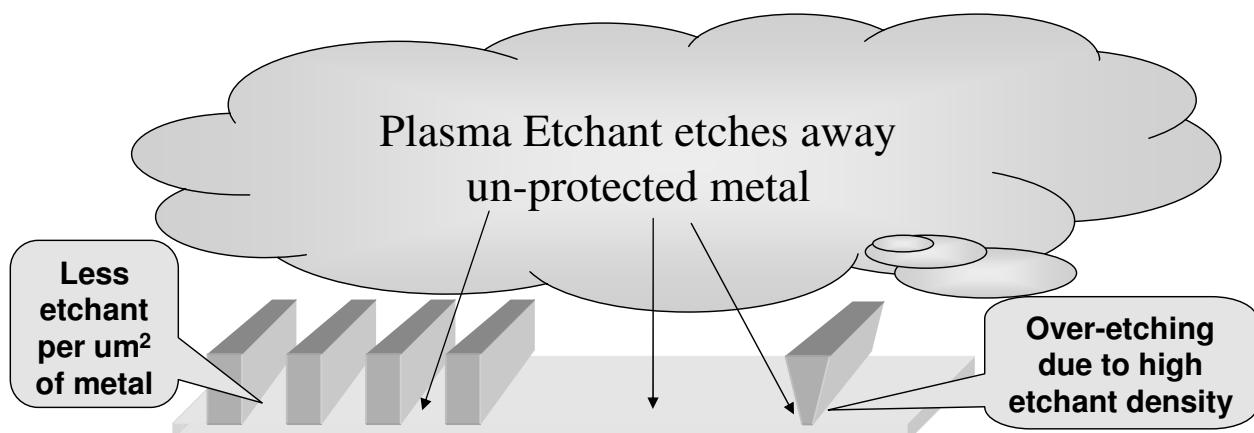
Cells without metal are inserted without checking drc's

You should insert cell with metal first then follow with cell without metal.

Problem: Metal Over-Etching

Antenna Critical Area Redundant Via Filler Cell Metal Fill Metal Slotting

- A metal wire in low metal density region receives a higher ratio of etchant can get over-etched
- Minimum metal density rules are used to control this

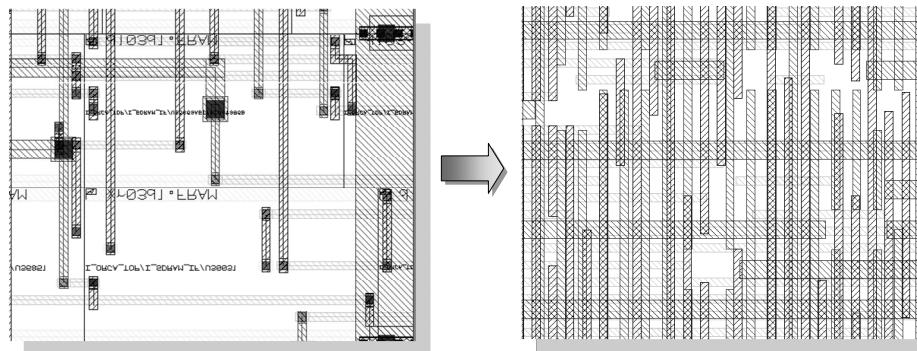


7-24

Too much etchant in contact with too little metal → overetched metal.

Solution: Metal Fill insert_metal_filler

- Fills empty tracks on all layers (default) with metal shapes to meet the minimum metal density rules
- Recommended to save metal fill in FILL view (default)
- Trims metal fill honoring the `minDensity` and `maxDensity` in each window of size `windowSize` defined under `DensityRule` section of techfile



7-25

GUI: Finishing → Insert Metal Fill ...

Metal filling is done to improve process planarization, which is important for processes with a large number of metal layers.

Metal fill insertion

Can choose to fill layers of choice including poly
By default leaves the fill floating

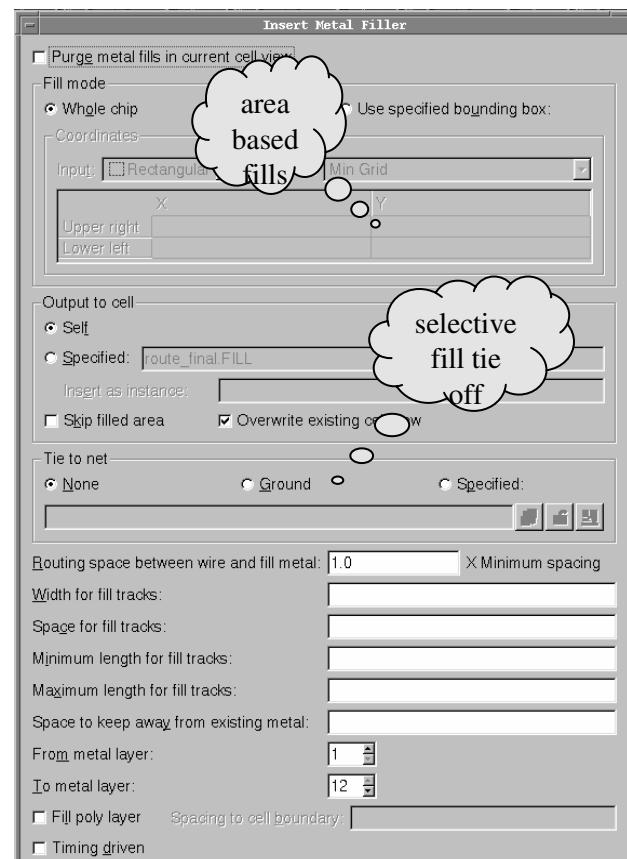
Optionally, can chose to connect to ground or other net

Can choose to insert floating vias

Can choose to do area based metal fill

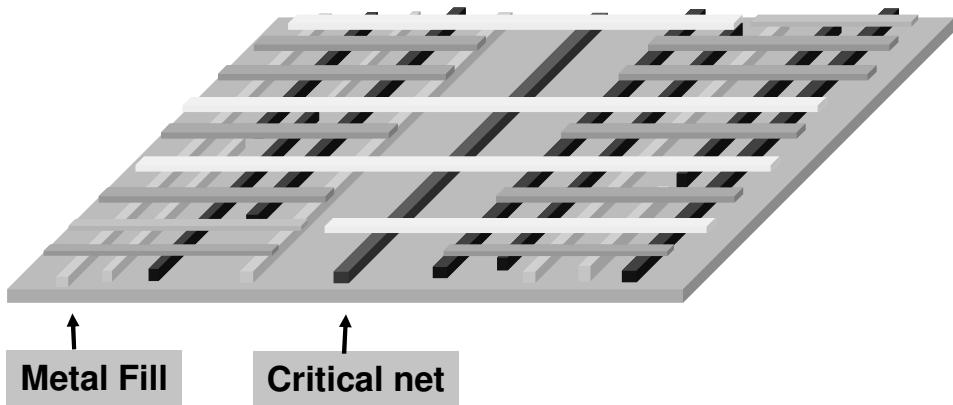
Can chose to specify required width, spacing for each layer

Timing driven metal fill doesn't fill wire tracks around critical nets



Timing-Driven Rule-Based Metal Fill

- Preserve timing on critical nets (`-timing_driven`)
- Metal fill near critical nets on the same layer, upper layer, and lower layer are removed or trimmed

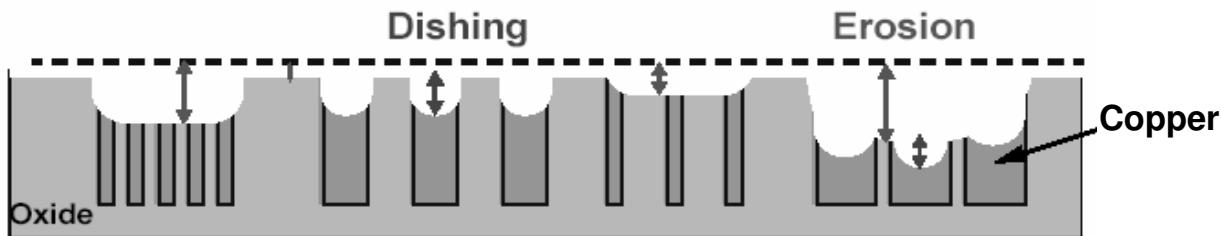


7-26

Problem: Metal Erosion

Antenna Critical Area Redundant Via Filler Cell Metal Fill Metal Slotting

- The wafer is made flat (planarized) by a process called Chemical Mechanical Polishing (CMP)
- Metals are mechanically softer than dielectrics:
 - CMP leaves metal tops with a concave shape - dishing
 - ◆ The wider the metal the more pronounced the dishing
 - ◆ Very wide traces can become quite thin – dishing this severe is called erosion
- Maximum metal density rules are used to control erosion



7-27

Any wafer process that effects the cross section of the conductors must be carefully controlled.

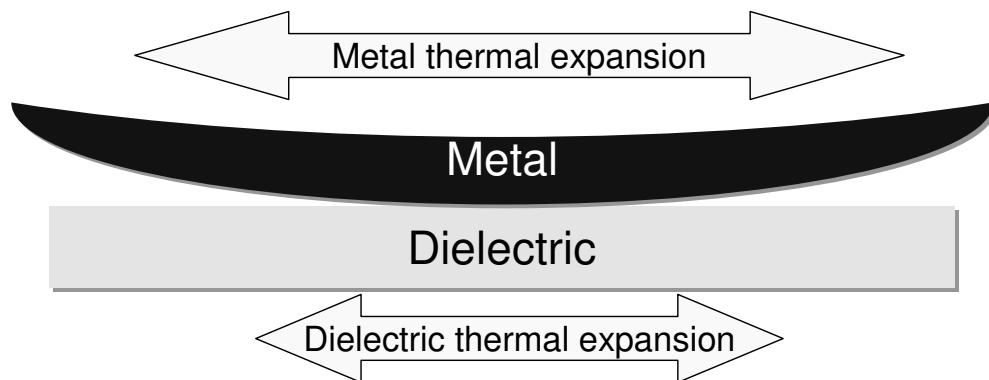
Erosion in wide metal traces will reduce their current carrying capability and degrade IR drop characteristics. Controlling this is vital to the distribution of power supplies and ground traces.

Dishing affects the resistance of the signal traces and hence their propagation delays. Accurate modeling of this effect, which is a function of trace width and spacing, is one of the primary strengths of TLU+ models over TLU models.

The wafer fabrication facility may impose maximum metal density rules to designs to keep the level of dishing, and especially erosion, to acceptable levels.

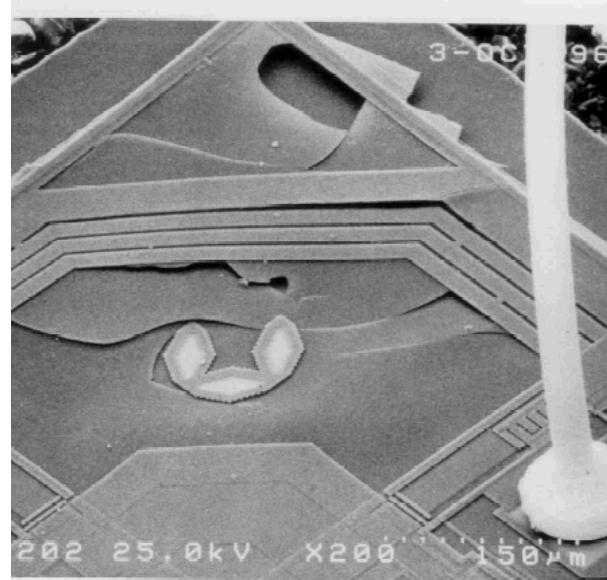
Problem: Metal Liftoff

- **Metal conductors and dielectrics have different coefficients of thermal expansion:**
 - Stress builds up with temperature cycling
 - Metals can delaminate (lift off) with time
 - Wide metal traces are more vulnerable than narrow ones
- **Maximum metal density rules also address this issue**



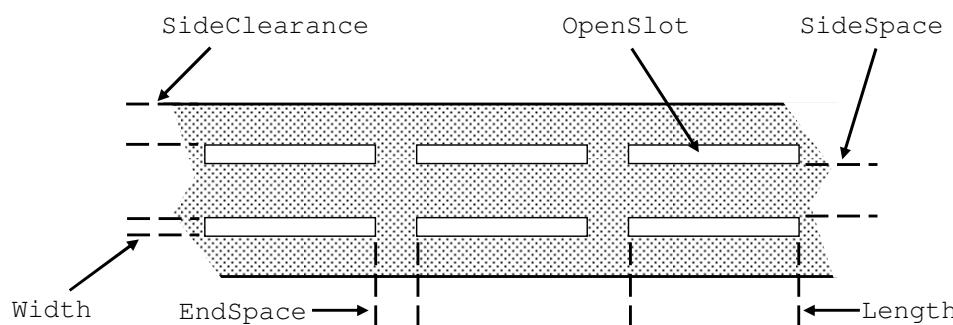
7-28

Poorly resolved mechanical stresses occur at the corners of dies. The photo shows delamination of both metal and insulator thin films near a die corner after a temperature cycling stress test.

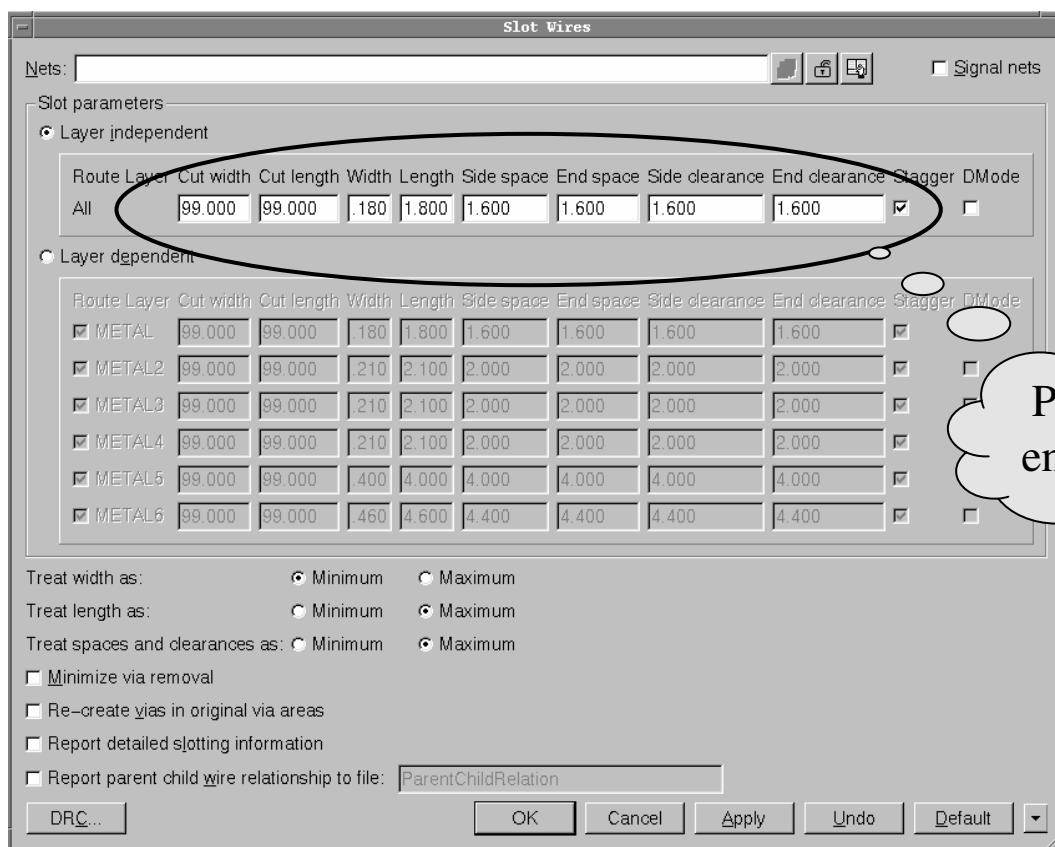


Solution: Metal Slotting

- **Slotting wide wires reduces the metal density**
 - Slots minimize stress buildup, reducing liftoff tendency
 - Dielectric in wide wires improves resistance to erosion
- **Primarily used on Power and Ground traces:**
 - Can apply to any other net if wide enough
- **Slotting parameters can be set globally or layer by layer**



7-29



DFM Issues and Solutions Summary

■ Addressing issues to increase manufacturing yield:

- Gate Oxide integrity → antenna fixing
- Via resistance and reliability → redundant contacts
- Metal erosion → metal slotting
- Metal liftoff → metal slotting
- Metal Over-Etching → metal filling
- Metal open/short defects → spread wire routing

■ DRC/ERC related

- notch/gap filling
- standard cell filling

7-30

In deep submicron VLSI, some manufacturing steps, like photo-resist exposure, development and etch and Chemical Mechanical Polishing (CMP) have detrimental effects on interconnect structures. These effects can vary based on local characteristics of the layout.

To make these effects predictable, the layouts must be made “uniform” with respect to certain density standards across very small localized areas of the die. In the past, foundries performed the postprocessing needed to provide this uniformity. The techniques they used were filling (selective insertion of shapes) or slotting (selective reduction of shapes).

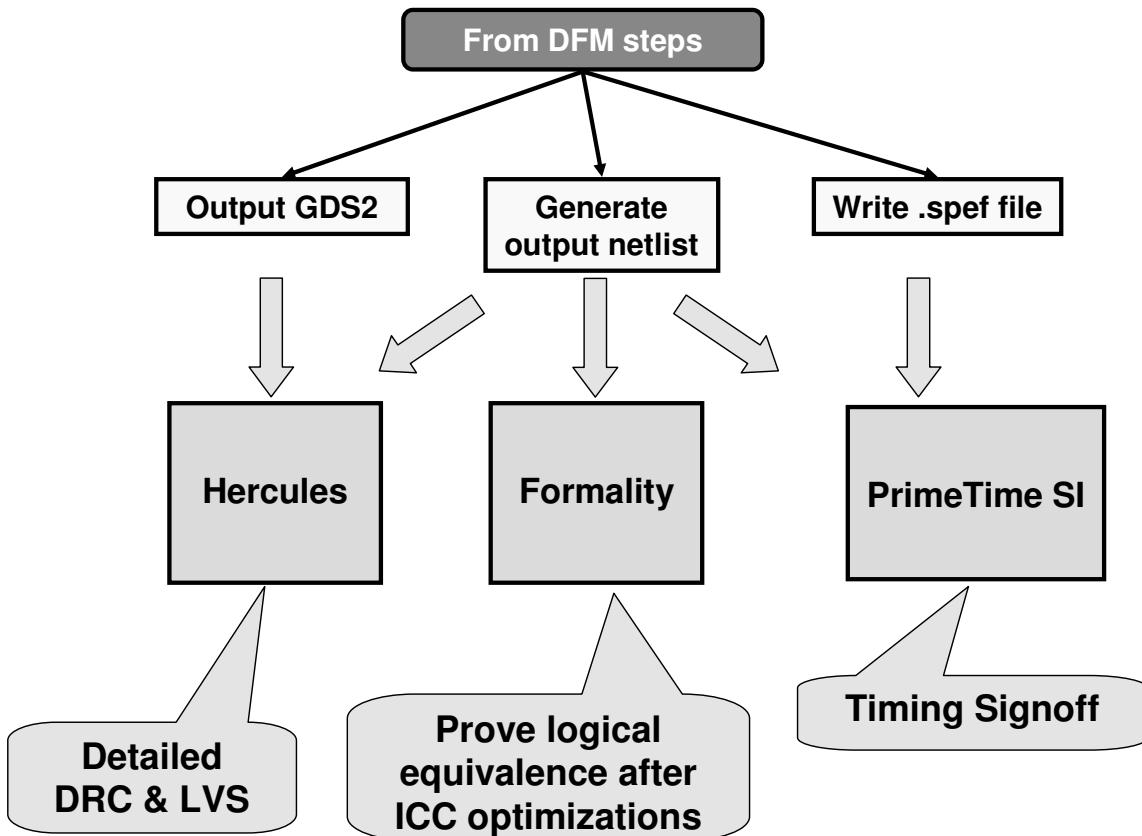
In today’s processes, the design tools doing RC extraction, delay calculation, IR drop analysis, timing/noise/crosstalk analysis must be aware of these slotting/filling activities or suffer significant inaccuracy.

ICC attempts to move all these into the design stage and out of the Fab post-processing regime.

To minimize the impact of the manufacturing process on device yields, foundries impose various density rules to make the layouts more uniform. For instance, the foundry may impose a density rule on an interconnect layer such that in any 10 um x 10 um window, there must be at least 35 um² of metal features, but no greater than 70 um² of metal.

Spare areas must be filled, but wide metal stripes must be slotted to meet the density rules.

Final Validation



7-31

Final Validation: Parasitics (SPEF or SBPF)

- Wire parasitics for PrimeTime are provided via a .SPEF or .SBPF file

```
write_parasitics
    -output <file_name>
    -format <SPEF|SBPF>
    -compress
    -no_name_mapping
```



Use Star-RCXT extraction for signoff

7-32

GUI: Timing → Write Parasitic

In general, you can expect SPEF from ICC's write_stream to be less accurate than a SPEF from the StarRCXT. For signoff, you should use SPEF from StarRCXT.

Final Validation: Netlist Output

- **Netlists for STA (Static Timing Analysis) do not require output of “Physical only cells” like:**
 - Corner pad cells
 - Pad/core filler cells
 - Unconnected cell instances
- **Unconnected cell instances (e.g. spare cells) are needed for LVS**

```
change_names -hierarchy -rules verilog  
write -format verilog -hierarchy -output final.v
```

7-33

Final Validation: GDS2 Output

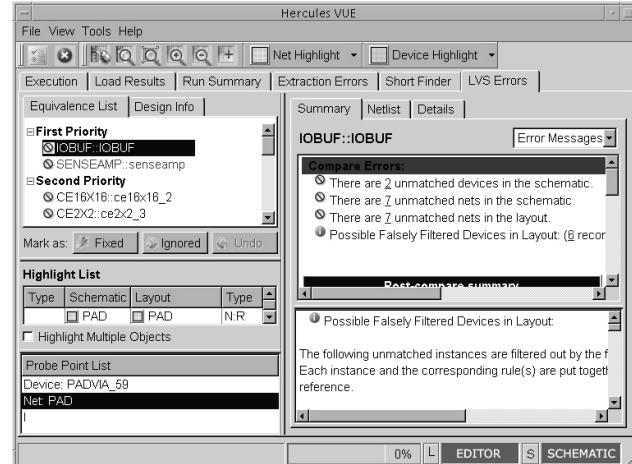
```
write_stream_options ...  
write_stream -cells DFM_clean orca.gdsii
```

- The GDS2 for external physical verification can be generated from IC Compiler
- Requires output of “physical only cells” like:
 - Corner pad cells
 - Pad/core filler cells
 - Unconnected cell instances

7-34

Hercules™ VUE Integration

- **Review, navigate and fix design violations for DRC, ERC, LVS**
- **Use layout environment of choice**
 - Support available for
 - ◆ IC Compiler & Astro
 - ◆ Analog Design Artist
 - ◆ ICWB-EV, CosmosEnterprise
 - API available for internal solutions
- **Ignore / waive violations on case by case basis**



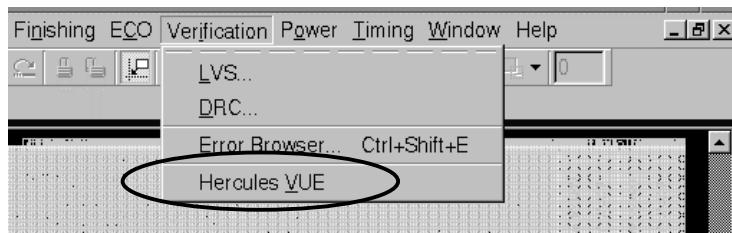
7-35

Accessing VUE in IC Compiler

- Set path to include both IC Compiler and Hercules tools
- Once in the ICC GUI, source the following tcl file to display the new Hercules menu in ICC:

```
icc_shell> source "$env(HERCULES_HOME_DIR)/etc/tcl-u/IccMenu.tcl"
```

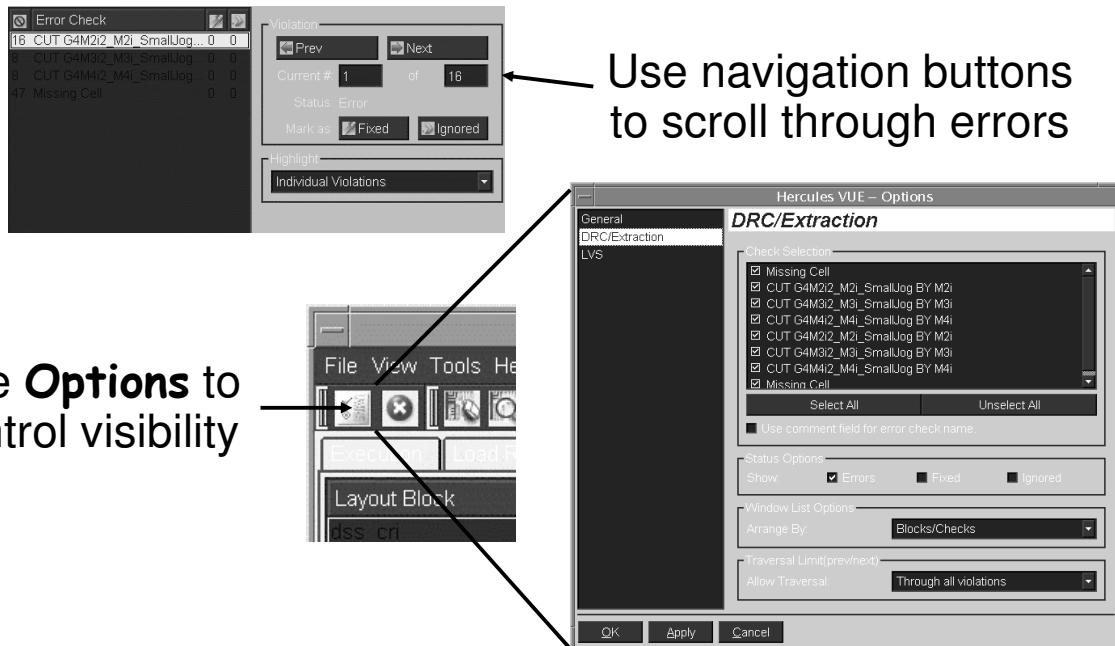
- Open the design
- In the ICC cell layout window, under the Verification menu, the new Hercules VUE option appears:



7-36

Running VUE

- Use the normal VUE operations for viewing errors in the layout



7-37

Test for Understanding



1. **What problem are you trying to solve by meeting “metal fill” design rules?**
2. **What are the two methods for fixing antenna rule violations? Does IC Compiler support both?**
3. **Placing additional vias in nets may slightly improve timing performance as a side effect. Why does this happen?**
4. **If you have a series of 50 um wide power rails, what manufacturability issue must you address?**

7-38

1. Problem: Metal Over-etching. Metal fill is used to meet the foundry's minimum metal density rules, which minimizes this problem.
2. Metal layer jumping and antenna diode insertion. IC Compiler supports both multiple vias act like parallel resistors, reducing the series impedance of the inter-layer contact, which reduces the RC net delay.
3. Metal lift-off during temperature cycling and metal erosion during CMP. Both are addressed by metal slotting, which introduces some hard oxide to support the metal during CMP and to reduce stresses which can cause lift-off.

4. Metal lift-off during temperature cycling and metal erosion during CMP. Both are addressed by metal slotting, which introduces some hard oxide to support the metal during CMP and to reduce stresses which can cause lift-off.

Summary

You should now know more about how to:

- **Perform key design for manufacturing steps required after the signal routing is complete:**

- Antenna fixing
- Modifying routing patterns to make them more resistant to short-causing defects
- Adding redundant contacts
- Inserting filler cells
- Metal filling and slotting



7-39

Lab 7: Chip Finishing



60 minutes

- Fix antenna violations
- Analyze critical area and use wire spreading/widening to improve manufacturing yield
- Perform via optimization for timing and yield improvement
- Perform standard cell filler insertion, as well as metal filling operations for metal density rule compliance

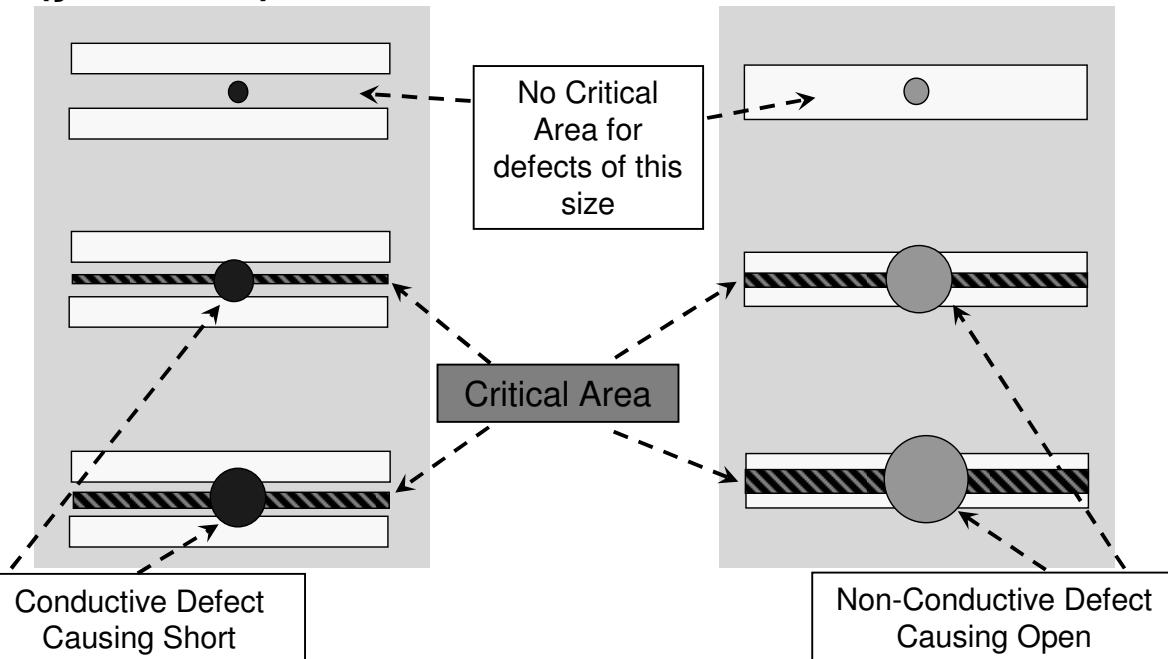
7-40

Appendix A

Critical Area Calculations

Critical Area Definition

- The region where, if the center of a random defect of a certain size falls on, will cause circuit failure (yield loss)



7-42

IC Compiler reports the average critical area. It uses the following formula:

$$A_{cr} = \int A_{cr}(x)f(x)dx$$

Where:

x: defect size (diameter)

f(x): defect size distribution function

A_{cr}(x): critical area for defect size x

A_{cr}: average critical area

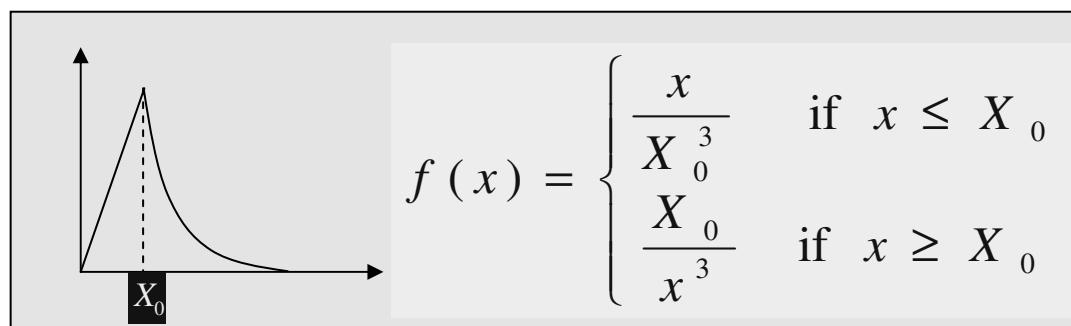
Discrete Defect Size Distribution

- Defect size distribution function depends on the fabrication process
- IC Compiler accepts discrete defect sizes and their probabilities in a table format
- An example

Defect Size	Probability
0.20	0.002778
0.36	0.000922
0.52	0.000412
0.68	0.000219
0.84	0.000130
...	...

7-43

This information comes from empirical studies conducted by the fab.
If no defect size distribution is defined, a built-in continuous function is used.
This function is widely accepted in the industry.



The y-axis is # of particles, the x-axis is defect diameter.
Notice that above a certain threshold, the defect density drops very rapidly.
The efficiency and type of filtering used in the fab sets the X0 limit.

Appendix B

FAQ

Wire Spreading

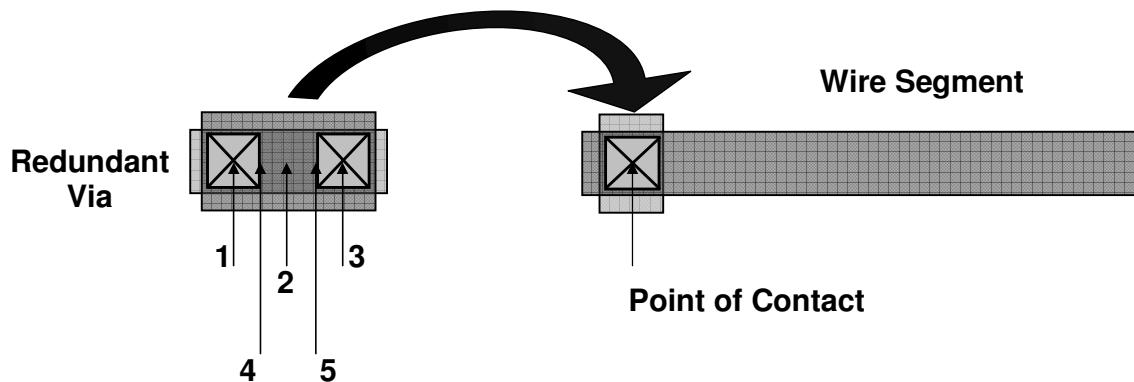
- **Why NOT run wire spreading before Antenna fixing?**
 - Not recommended, Antenna has the most priority after DRC
 - Wire spreading (by pushing off-track) might not leave enough resource to fix antenna
- **Will wire spreading switch layers?**
 - Pushes wires off the track if space available, doesn't switch layers to allow pushing (spreading)
 - However, Search & Repair run after wire spreading may result in minimal changes to resolve DRC
- **Will wire spreading introduce Antenna violations?**
 - Antenna ratio dependant on Antenna length may change slightly by wire spreading
 - In most cases should NOT introduce new Antenna violations
- **Should I turn ON Antenna checking during wire spreading?**
 - Recommended

7-45

Redundant Via Insertion 1/2

■ Will IC Compiler try all possible combinations?

- 1x2 and 2x1 combinations treated same with the `rotateLineViaArray` parameter
- `-optimize_level 0` tries 3 combinations of contact points in a double via
- `-optimize_level 1` tries 5 combinations of contact points in a double via



7-46

The redundant via, shown on the left, is inserted at the point of contact shown on the right. With `optimize_level 0` it is inserted using the locations 1, 2 and 3. If none of the contact points lead to a DRC-legal connection, the double via is not inserted. `optimize_level 1` increases the number of contact points and the chance of a successful redundant-via insertion.

Redundant Via Insertion 2/2

- **Will redundant vias increase critical area?**
 - Yes, any increase in metal (route/cut layer) on layout is bound to introduce critical area. But it will be very little.
- **Will redundant vias introduce new antenna violations?**
 - Redundant Vias are inserted without disturbing the route pattern
 - However, later S&R runs to resolve DRC may cause antenna violations
 - Turning ON antenna checking will roll back double to single vias reducing antennas
- **Can redundant vias be removed by the user?**
 - Redundant vias cannot be removed by a user command
- **Will S&R or ECO route remove redundant vias?**
 - May remove only if required to resolve any DRC violations

7-47

Filler Cell Insertion

- **How can I remove existing standard cell filler?**

- *remove_stdcell_filler -stdcell -pad \ -tap -bboxing_box*

- **Is multi-height filler cell insertion supported?**

- Yes

- **Is voltage aware filler cell insertion supported?**

- Yes

7-48

Metal Fill Insertion

- **Can I do density checking in IC Compiler?**
 - verify_drc can report metal density violations
- **Will fill insertion result in gradient metal fill density across windows?**
 - No, current implementation should not result in larger gradients
 - Track based metal fill insertion fills all empty wire tracks making it uniform
- **Is metal fill trimming possible in IC Compiler?**
 - Yes (trim_fill_eco)
- **Can ICC trim fill around certain nets?**
 - No
- **How can metal density be reported?**
 - No standalone command available
 - Metal fill insertion does not report the windows in which density is not met

7-49

```
trim_fill_eco
[-input fill_view_name]
[-output fill_view_name]
[-spacing_to_routing number]
[-remove_vio_fill]
```

When you have finished metal filling and the engineering change order has changed the routing, you can use this command to trim the metal fill and avoid design rule constraint violations.

This page was intentionally left blank.



Customer Support

© 2009 Synopsys, Inc. All Rights Reserved

20090112

Synopsys Support Resources

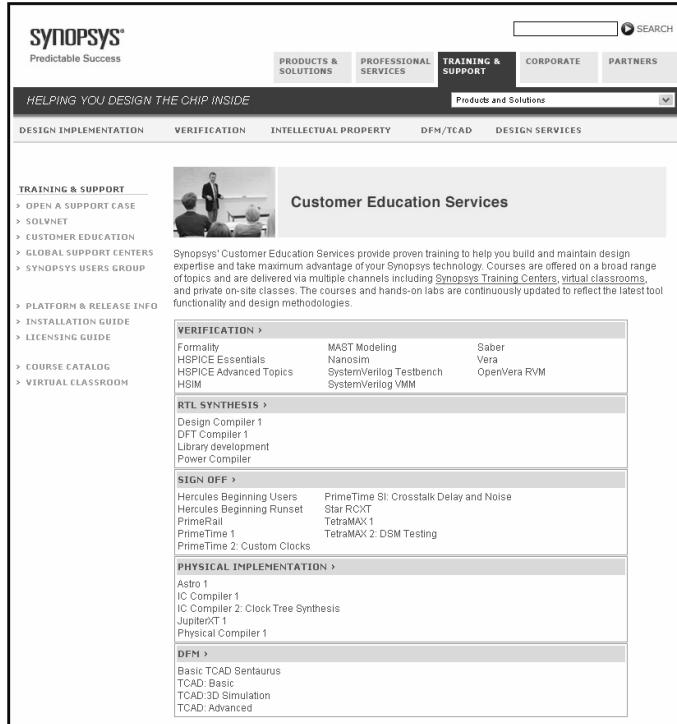
1. Build Your Expertise: Customer Education Services

- www.synopsys.com/support
 - ◆ Workshop schedule and registration
 - ◆ Download materials
(SolvNet id required)

2. Empower Yourself: solvnet.synopsys.com

- Online technical information and access to support resources
- Documentation & Media

3. Access Synopsys Experts: Support Center



The screenshot shows the 'Customer Education Services' section of the Synopsys website. At the top, there's a navigation bar with links for PRODUCTS & SOLUTIONS, PROFESSIONAL SERVICES, TRAINING & SUPPORT (which is highlighted in black), CORPORATE, and PARTNERS. Below the navigation is a search bar and a dropdown menu set to 'Products and Solutions'. The main content area has a heading 'HELPING YOU DESIGN THE CHIP INSIDE' and a sub-section 'Customer Education Services' featuring a photo of a person giving a presentation. To the left is a sidebar with links for TRAINING & SUPPORT, PLATFORM & RELEASE INFO, INSTALLATION GUIDE, LICENSING GUIDE, COURSE CATALOG, and VIRTUAL CLASSROOM. The right side contains sections for VERIFICATION (Formality, HSPICE Essentials, HSPICE Advanced Topics, HSM), RTL SYNTHESIS (Design Compiler 1, DFT Compiler 1, Library development, Power Compiler), SIGN OFF (Hercules Beginning Users, Hercules Beginning Runset, PrimeRail, PrimeTime 1, PrimeTime 2: Custom Clocks), PHYSICAL IMPLEMENTATION (Astro 1, IC Compiler 1, IC Compiler 2: Clock Tree Synthesis, JupiterXT 1, Physical Compiler 1), and DFM (Basic TCAD Sentaurus, TCAD: Basic, TCAD: 3D Simulation, TCAD: Advanced). Each section lists specific tools or topics.

CS-2

SolvNet Online Support Offers

- Immediate access to the latest technical information
- Product- and Release-Specific Update Training
- Thousands of expert-authored articles, Q&As, scripts and tool tips
- Enter-a-call online Support Center access
- Release information
- Online documentation
- License keys
- Electronic software downloads
- Synopsys announcements (latest tool, event and product information)

The screenshot shows the SolvNet Online Support Offers website. At the top, there's a navigation bar with links for 'solvnet home | synopsys.com', 'SEARCH', 'PROFILE & PREFERENCES', 'FEEDBACK', 'SITEMAP', 'HELP', and 'LOGOUT'. Below the navigation is a search bar labeled 'Search SolvNet' with a dropdown menu set to 'All'. To the right of the search bar are links for 'Browse | Help', 'Search IP | My Saved Articles', and 'My Support'.

The main content area is divided into several sections:

- Main Navigation:** Links to 'Enter A Call - Tool Support', 'Documentation & Downloads Center', 'Training Center', and 'Sitemap'.
- Announcements:** Headlines include 'Synopsis Guidelines for Wireload Mode and Topographical Mode', 'Synthesis Scripts - View the DC Reference Methodology (DC-RM)', 'NEW!! 2007.06 Update Trainings Now Offered On SolvNet!', 'Japanese Update Training Now Available', 'SNUG 2007: Registration Open - Boston | Call for Papers - Israel', and 'Featured Article'.
- Technical News & Events:** Includes 'Did You Know?' (about Product Preferences), 'SolvNet TechUpdate Newsletter' (with a link to 'More Newsletters'), 'SNUG - Synopsys Users Group' (with a link to 'More Events'), and 'What's New on SolvNet'.
- Featured Article:** Headline: 'Challenges in Crosstalk-Aware Hierarchical STA for Multi-Million Gate VoIP SoC'.
- New & Updated Articles For Your Products:** A section where users can 'Select Your Products' to view new articles. It lists several articles with their dates:
 - 2007.03 JupiterXT - IC Compiler Hierarchical Design Flow (08-17-2007)
 - sh enable line editing in LSC (08-16-2007)
 - Errors with smart tag cells by rules--connect to option (08-16-2007)
 - Pin Constraints Defined in TDF Are Not Honored (08-14-2007)
 - How to Run Low Power Placement Without Calling Clock Tree Synthesis During place opt (08-14-2007)
 - Useful Skew in IC Compiler (08-13-2007) Updated!
- Footer:** Links for 'SOLVNET HOME | MY PROFILE | FEEDBACK | SITEMAP | HELP | LOGOUT' and copyright information: '© Copyright 2007, Synopsys, Inc. All Rights Reserved - Terms of Use | Privacy Policy'.

CS-3

SolvNet Registration is Easy

- 1. Go to solvnet.synopsys.com/ProcessRegistration**
- 2. Pick a username and password.**
- 3. You will need your “Site ID” on the following page.**
- 4. Authorization typically takes just a few minutes.**

The form consists of two stacked sections. The top section is titled "New User Registration" and contains fields for "Your Corporate Email", "Select a Username", "Select a Password", and "Re-enter Password". There is also a checkbox for "I am 18 or older". The bottom section is also titled "New User Registration" and contains a heading "Important: Please Read Before Registering". It states that to access all Synopsys Online Services, an Active Site ID is required. It lists services like SolvNet Knowledge Base, Online Product Documentation, SmartKey License Retrieval, Electronic Software Downloads, and ViewSupport. It also has a "Synopsys Site ID" input field and a "Next" button.

CS-4

Support Center: AE-based Support

- **Industry seasoned Application Engineers:**
 - 50% of the support staff has >5 years applied experience
 - Many tool specialist AEs with >12 years industry experience
 - Access to internal support resources
- **Great wealth of applied knowledge:**
 - Service >2000 issues per month
- **Remote access and debug via ViewConnect**

The screenshot shows the Synopsys Support Center website. At the top, there's a navigation bar with links for PRODUCTS & SOLUTIONS, PROFESSIONAL SERVICES, TRAINING & SUPPORT (which is highlighted), CORPORATE, and PARTNERS. Below the navigation is a search bar and a dropdown menu set to "Products and Solutions". The main content area features a banner with the text "HELPING YOU DESIGN THE CHIP INSIDE" and "Predictable Success". It includes sections for DESIGN IMPLEMENTATION, VERIFICATION, INTELLECTUAL PROPERTY, DFH/TCAD, and DESIGN SERVICES. A "TRAINING & SUPPORT" section lists links for OPEN A SUPPORT CASE, SOLVNET, CUSTOMER EDUCATION, GLOBAL SUPPORT CENTERS, and SYNOPSYS USERS GROUP. Another section for PLATFORM & RELEASE INFO includes links for INSTALLATION GUIDE and LICENSING GUIDE. A "Global Support Centers" section features a photo of a support representative and links for NORTH AMERICA, EUROPE, JAPAN, and ASIA PACIFIC. A "North America Support" section provides contact info for Saber Products, Advanced Packaging Products, and Europe Support. A "Contact us: Enter A Call" button is prominently displayed. A sidebar on the right contains a "quick tips" box.

www.synopsys.com/support

CS-5

Other Technical Sources

- **Application Consultants (ACs):**

- Tool and methodology pre-sales support
- Contact your Sales Account Manager for more information

- **Synopsys Professional Services (SPS) Consultants:**

- Available for in-depth, on-site, dedicated, custom consulting
- Contact your Sales Account Manager for more details

- **SNUG (Synopsys Users Group):**

- www.snug-universal.org

CS-6

Summary: Getting Support

- **Customer Education Services**
- **SolvNet**
- **Support Center**
- **SNUG**

CS- 7

This page was intentionally left blank.