Encounter RTL to GDSII

The Fastest Route to Silicon

Fundamental Technology Shift

Submicron

3M gates
.25-15 166MHz
Small macro count
No SI issues
Flat design

Placement-Centric Timing
- Floorplanning
- Placement
- 3D extraction
- Tool flow

C_{1,0} \text{ (wire length)}

Submicron Methodology

- Logic synthesis
- Floor planning (based on cluster)
- Physical synthesis (based on placement)
- Power planning
- Clock tree synthesis
- Routing
- Physical analysis

"Design closure"

Front-end

Back-end

Long wire iterations

Nanometer

20M+ gates
.13-90nm 600MHz
100's of macros
Pervasive SI issues
Hierarchical design

Interconnect-Centric Timing
- Virtual prototyping
- Nanometer routing
- Signal Integrity
- Integrated architecture

C_{3,0} \text{ (coupling)}
**Fundamental Technology Shift**

**Submicron**
- 3M gates
- .25-16GHz
- Small macro count
- No SI issues
- Flat design

**Nanometer**
- 20M+ gates
- .13-90nm 600MHz
- 100s of macros
- Pervasive SI issues
- Hierarchical design

**Placement-Centric Timing**
- Floorplanning
- Placement
- 3D extraction
- Tool flow

**Interconnect-Centric Timing**
- Virtual prototyping
- Nanometer routing
- Signal Integrity
- Integrated architecture

---

**How to Shorten Time to Tape Out?**

_Solve more of the problem earlier, faster_

**SoC Encounter – 1st full-chip interconnect**
- Get to final netlist
- Get top-level plan
- Block implementation

**Traditional 1st full-chip interconnect**
- Top level complete
- Tape out

**RTL**

**Quality logic**
- Logic structure defines wire topology

**Top-level must be right**

**QoR:**
- timing, SI, power, area, DRC clean / mfrg
- Want fast TAT = productivity, time to tapeout
- Big chips → want BIG blocks or else top-level unmanageable

**RC**

**First Encounter**

**Nano Encounter / CeltIC**

Better logic, better wires, faster run times = tape out sooner
Encounter Platform
High-performance architecture, best-in-class engines

- Best Synthesis – RC
- Best Virtual Prototype – FE
- New Global Physical Synthesis
- Best Routing – Nanoroute
- Best Signal Integrity – CeltIC
## Encounter Platform
### 30-40 RTL- and netlist-to-GDS tapeouts at 90nm

**North America**
- 90nm 2.6M gate network processor chip tape-out Oct. 2003
- 90nm 1.5M gate network processor chip tape-out Dec. 2003
- 90nm 2.1M gate wireless application tape-out Nov. 2003
- 90nm 2M gate wireless application tape-out Dec. 2003
- 90nm 2.4M gate instance 16-way DSP tape-out Jan. 2004
- Multiple 90nm test chips from multiple customers

**Japan**
- 90nm 12M gate tape-out Jan. 2004
- 90nm 3M gate consumer application tape-out Feb. 2004
- STARC Starcad-21 90nm reference flow Mar. 2004

**Europe**
- 90nm 1.5M gate Bluetooth tape-out Dec. 2003
- 3 more 90nm tape-outs scheduled for Q1/2004 + 7 more scheduled in Q2/Q3
- Multiple 90nm test chips from multiple customers

**Taiwan**
- TSMC 5.0 90nm reference flow

---

## SoC Implementation – Logic Matters!

1. **RTL Synthesis**
   - Full-Chip Prototyping / Floorplanning
   - Block-level Implementation
   - Chip finishing + Signoff

2. **RTL Elaboration**
   - Traditional – 50% of timing
   - RTL Compiler – 85% of timing

3. **Mapping**
   - Traditional – 50% of timing
   - RTL Compiler – 15% of timing

4. **Incremental optimization**
   - Traditional – 50% of timing
   - RTL Compiler – 15% of timing

5. **Netlist**

---

*New Global optimization finds the best overall structure*

*Old synthesis squeezes the best out of a local minima*
SoC Implementation – Logic Matters!

Example: Well-Known Networking Company

- **RTL Synthesis**: Get to final netlist
  - Top-level plan
  - Block-level implementation

  **Physical Synthesis**

- 108 hrs – fail congestion
  - Failed routing

- 18 hrs – 0% congestion
  - Routed clean

---

**Die size was at risk with standard synthesis flow**
- Attempted all tricks, used latest available versions of SW.
- Brought in Applications Consultants to aid.

- Cadence RTL Compiler was made available to trial on selected blocks.
- Results were compelling enough to utilize tool on most blocks

**In the end, achieved an area reduction of 15% of synthesizable area**

- Cadence applications support was exemplary through the whole effort.

**15% logic area reduction**

Source: Alan Nakamoto, Director of Design Services, PMC-Sierra, DAC 2004
Top-Level Planning – *Silicon Virtual Prototype*

**RTL Synthesis**

**Full-Chip Prototyping / Floorplanning**

**Block-level Implementation**

**Chip finishing + Signoff**

---

**Silicon Virtual Prototype**

- Complete “flat” physical design (proves timing and routability)
  - Very fast!
  - Assess the logical design
  - Guide the implementation

---

**SVP Drives Hierarchical Implementation**

- Accurate Full-Chip Physical Prototype... with wires!
- Top Level Partition View
- Optimal Pin Assignment
- Block-Level Timing Budgets

**Result:** design planning based on a known-feasible solution
First Encounter vs Floor Planning

Typically save 6+ weeks in top-level planning

“80% market share”

Dataquest, December 2003
Block-Level Implementation – Fast Design Closure

Major upgrade for version 4.1

New Global Physical Synthesis

Speed, capacity and integration

- 500K gates/hr on fast Linux, 32b address space
- Global logic restructuring built directly on FE
  - One primary command `optDesign` (pre + postroute)
- Customer tape outs from 3 to 8 million gates flat

<table>
<thead>
<tr>
<th>Design</th>
<th>Run Time</th>
<th>Gates / Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5 MG, 148 MHz</td>
<td>15h 20m</td>
<td>550K</td>
</tr>
<tr>
<td>5.4 MG, 500 MHz</td>
<td>10h 14m</td>
<td>450K</td>
</tr>
<tr>
<td>2.7 MG, 400MHz</td>
<td>4h 25m</td>
<td>600K</td>
</tr>
<tr>
<td>3.2 MG, 500MHz</td>
<td>7h 48m</td>
<td>400K</td>
</tr>
<tr>
<td>2.8 MG, 200MHz</td>
<td>6h 25m</td>
<td>430K</td>
</tr>
</tbody>
</table>
**Global Physical Synthesis**  
*Why speed and capacity matter*

- RTL Synthesis
- Full-Chip Prototyping / Floorplanning
- Block-level Implementation
- Chip finishing + Signoff

---

**High-Capacity SMART Routing**  
*SI, Manufacturing-Aware, Routability, Timing*

- Support for advanced manufacturability issues
  - Wire spreading, antenna rules, redundant vias…
- Fewer design rule violations (DRV’s)
- Fewer SI violations
- Less repair effort after routing

---

**Key Features**:
- Encounter platform
- In-memory design image
- Global Physical Synthesis
- Power planning
- Timing analysis
- Clock Tree Synthesis
- Power integrity analysis
- Signal Integrity analysis
- Nanometer Routing
- OpenAccess Database
New Nanoroute Super Threading

Fast router gets faster

• Applies Nanoroute multithreading over ordinary Ethernet
  – Very good results using groups of low cost 2-CPU desktop machines
  – Route 2M nets/hr, no special hardware

• Much more efficient than conventional distributed processing
  – Scales to large numbers of CPU’s
  – Easy to use

DAC 2004
13M gates, 3.3M nets
Routed live in under 2 hours

New Nanoroute Super Threading

“Instant wires”

• NanoRoute is now an order of magnitude faster than any competitor
• Routed 8 million gate design with 20 CPUs, in under an hour
#1 Signal Integrity
Least pessimistic, most widely used

Cadence mops the floor with Synopsys…

ESNUG Post #420
22 Oct 2003
www.deepchip.com/items/0420-01.html

Low Power Design Flow
Collaboration with TSMC
Low Power Design Flow

- Low power synthesis
  - Clock gating, shutdown modes ...

- Leakage power optimization
  - Multi Vt, optimization
  - State-retention power gating (SRPG)

- Multi-supply voltage design
  - Library/modeling
  - Physical synthesis/optimization
  - Clock tree
  - Power planning
  - Low-power clock design

- Automatic power grid generation

- Signoff power integrity analysis (VoltageStorm)

Low Power Design Flow – Example

- ARM 1136 design, tape out July 2004
- Die size 4mm x 4mm, 360 pads
- Cell count ~300K, Utilization ~80%

  - 2 Domains
  - 1.0V (TOP)
  - 0.8V (NO RAM)

Power Comparison – Baseline (BL) vs Low Power (LP)
(Worst power condition, post route, PowerMeter run at speed with activity factor on I/O 50%)

<table>
<thead>
<tr>
<th></th>
<th>BL Total (mW)</th>
<th>Top (mW)</th>
<th>No RAM (mW)</th>
<th>LP Total (mW)</th>
<th>Top (mW)</th>
<th>No RAM (mW)</th>
<th>Savings Total (mW)</th>
<th>Top (mW)</th>
<th>No RAM (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leakage</td>
<td>480</td>
<td>176</td>
<td>304</td>
<td>279</td>
<td>108</td>
<td>171</td>
<td>200 (42%)</td>
<td>68 (39%)</td>
<td>132 (44%)</td>
</tr>
<tr>
<td>Switching</td>
<td>982</td>
<td>352</td>
<td>630</td>
<td>574</td>
<td>256</td>
<td>317</td>
<td>409 (42%)</td>
<td>96 (27%)</td>
<td>313 (50%)</td>
</tr>
<tr>
<td>Total</td>
<td>1462</td>
<td>528</td>
<td>934</td>
<td>853</td>
<td>364</td>
<td>489</td>
<td>609 (42%)</td>
<td>164 (31%)</td>
<td>445 (48%)</td>
</tr>
</tbody>
</table>

- Total power (total leakage + total dynamic power) reduction: 609mw vs 1462 mw = 42%
- Std cell only region (no RAMs) reduction: 44% leakage, 48% dynamic
**Electrical Verification**

_Silicon predictability at nanometer geometries_

- VoltageStorm
  - IR-Drop Analysis
- SignalStorm
  - Delay Calculation
- CelltIC
  - Crosstalk Analysis

**Signoff Timing Engine**

- 90nm-ready signoff 3D extraction
- High accuracy delay calculation
- SI “Delay”
- SI “Glitch”
- IR drop analysis

---

**Enabling Open Interoperability**

_OpenAccess Database_

- Cadence Encounter Platform
- Cadence Virtuoso Platform

- User extensions
- Unified database
- Netlist
- Physical
- Electrical
- Schematic
- Layout
- More...

**Goal:**

Better Support for Mixed Digital-Analog SoC’s

**Example:** Sanyo Digital Camera-on-a-Chip

EDAC Design Achievement Award, 2001
Taped out using First Encounter
Encounter Packages
2004.1 Release

Cadence Encounter Platform
Integrated system with better engine technology

SoC Encounter – 1st full-chip interconnect

Traditional 1st full-chip interconnect

Get to final netlist

Get top-level plan

Block implementation

Better logic
Wires-first approach

Fastest run-time performance
#1 signal integrity closure
Low-power design flow

= PREDICTABILITY

CYCLE-TIME REDUCTION
The fastest route to silicon

27 CADENCE CONFIDENTIAL

28 CADENCE CONFIDENTIAL
Encounter™ GPS 4.1 Workshop
Hierarchical RTL-to-GDSII Physical Implementation Solution

Lab Materials: Version 4.1.9
About this workshop
This workshop introduces the SoC Encounter™ hierarchical RTL-to-GDS physical implementation solution, its user interface, and design methodology. Many features and tool options are not covered in this workshop. You can follow this introduction with the more comprehensive SoC Encounter training offered by Cadence.

Other Available Workshops
- Cadence Signal and Power Integrity Closure Flow
- Encounter Test Solutions
- Encounter RTL Compiler

Conventions
- Sections with titles which end with an asterisk (*) can be skipped.
- Sections with [square] are saved starting points. You can start at these sections without completing prior steps.
- Paragraphs that begin with [pentagon] provide notes or supplementary information about the current steps.
- Commands or user input are in Courier type.
- Elements of the user interface are in bold.
- LMB means Left Mouse Button and RMB means Right Mouse Button
Table of Contents:

Section 1: Introduction to Encounter * .............................................................6
- Starting Encounter .....................................................................................................6
- Getting Help .................................................................................................................6
- Introduction to the Encounter GUI ..............................................................................7
- Design Import ..............................................................................................................7
- Netlist Optimization with RTL Compiler™ .................................................................8
- Quick prototype timing .................................................................................................8
- Design Statistics ..........................................................................................................8
- ViewING the Design .....................................................................................................9
- Design Browser ......................................................................................................... 11

Section 2: Silicon Virtual Prototyping .............................................................12
- Quick Zero load timing with timeDesign .................................................................12
- Floorplanning Basics * ...........................................................................................13
- Module Constraint types ..........................................................................................13
- Module/Block attributes ............................................................................................14
- Relative Floorplanning * ...........................................................................................15
- Multiple Supply/Mixed Voltages (MSMV) * .............................................................17
- Automatic Power Planner * ....................................................................................19
- Wire-Editor * ..........................................................................................................20
- PrototypING Placement * .......................................................................................25
- Prototype Routing with TrialRoute™ ..........................................................................25
- RC Extraction .............................................................................................................26
- Pre-Clock Optimization with GPS * ........................................................................27
- timedesign .................................................................................................................27
- Clock Tree Synthesis (CTS) * ..................................................................................28
- Statistical Power Analysis .......................................................................................29

Section 3: Partitioning ......................................................................................32
- Loading the Re-grouped netlist * ...........................................................................33
Load Floorplan/Specify Partitions ..................................................................................34
Partition – Specify Black Black Box * ........................................................................35
Partitioning – Place & Route ......................................................................................36
Partitioning – Buffer insertion .....................................................................................37
Commit Partitions .......................................................................................................38
Release the blocks ....................................................................................................39

Section 4: Block Implementation .................................................................................40
Load the tdsp_core partition .......................................................................................41
Pre-CTS Optimization with useful skew .....................................................................42
CTS + Concurrent Nanoroute™ ..................................................................................44
Detail Route with Native Nanoroute ..........................................................................45
Optimizing to minimize leakage power .......................................................................46
Delay Calculation and Timing Analysis ......................................................................47
Path Group Support ...................................................................................................48
Post-Route Optimization with SI Delay .......................................................................48
Celtic Analysis and fixing ............................................................................................48
Run sign-off Extraction AND Timing * .......................................................................49
Run sign-off power analysis * .....................................................................................50
Save the partitioned block ..........................................................................................51

Section 5: Hierarchical Implementation .......................................................................52
Run Top-Level Timing Closure Native Nanoroute™ ....................................................53

Section 6: Chip Assembly and Sign-Off ......................................................................54
Merge the Partitioned Design .......................................................................................54
Run sign-off Extraction and Timing ............................................................................55
Verplex Formal Verification * .....................................................................................55
Export GDS for Chip finishing ..................................................................................56

Appendix A – Hierarchical Design .............................................................................57
Appendix B – Saved Starting Points ............................................................................58
What is Covered:

- Introduction to Encounter
- Silicon Virtual Prototyping
- Block Implementation
- Hierarchical Implementation
- Chip Assembly

Flow Overview
Section 1: Introduction to Encounter *

STARTING ENCOUNTER

1) Login using the account icd_train and password icd_train.
2) Open a dedicated xterm window to launch SoC Encounter (the xterm serves as command line and log window).
3) Change directories to
   /home/icd_train/SOCE4.1_WORKSHOP/work_fe
4) Type: encounter

GETTING HELP

Online documentation and Help is available through several methods.
5) Click on the Help button in the upper right corner of the Encounter window to view the Encounter documentation in an internet browser.
6) Click on the Help button on a form to view the related help.
7) To see syntax information about a specific Encounter command, type the following command in the Encounter console:

   help command_name

8) To see the entire list of Encounter commands and their syntax, type the following command in the Encounter console:

   help

   The information is written to the encounter.log file.
9) To see the complete information about an Encounter command, type the following command in the Encounter console:

   man command_name
INTRODUCTION TO THE ENCOUNTER GUI

10) Familiarize yourself with the Encounter GUI by viewing the functions on the pull-down menus.

11) Move the mouse over the encounter menu widgets located immediately under the Encounter GUI. These are shortcuts to the Encounter menu items. The function of the short cut is highlighted when the mouse is over the tool bar widget.

DESIGN IMPORT

12) Open the Design Import form by selecting Design => Design Import… or by clicking on the Design Import icon just below the Design pulldown.

13) Populate the form using the Load button at the bottom of the Design Import Form. Navigate the file manager to find the file named 
../data/dtmf_chip.conf
14) Review the form contents and the tabs for Design, Core Spec Defaults, Timing and Power. File locations for netlist, libraries, technology data, and design constraints are captured along with many defaults.

15) Click OK.

NETLIST OPTIMIZATION WITH RTL COMPLIER™

With the advent of Netlist-2-Netlist optimization (or N2N) you can now achieve a significantly better starting point for place and route if your netlist was generated by older synthesis techniques. N2N allows you to take advantage of RTL Compiler’s advanced global optimization techniques when starting from a less than optimal netlist. This can greatly aid in meeting your timing and area goals.

16) We will not run N2N in this workshop since the netlist was already synthesized with RTL Compiler.

QUICK PROTOTYPE TIMING

Use the following steps to get an idea of design feasibility. You can quickly get the full chip timing based on real routes and parasitics instead of net loading models which do not accurately predict congestion. Later we will show how to get a quick timing without wire loading.

17) At the encounter prompt, type: amoebaPlace -fp to do a quick prototype placement. (You can also go to place => place… from the menu bar)
   ❖ Note that running Amoebaplace in Prototyping mode can leave overlap and placement error which can be ignored at this stage.

18) To run trialRoute and time the design, type: timeDesign -preCTS

19) What is the worst setup slack? _______
    What type of path is it (reg2reg, reg2Out, etc)? _______
    How long did this step take? _______

DESIGN STATISTICS

❖ The design status shown in the upper right corner of the Encounter GUI is: Timing Analyzed. Refer to this to track the last step completed.

20) Select Tools => Netlist Stats to gather design information and print it to the encounter.log file.
21) Browse the log file in a separate xterm.
   # of cells __________
   # of nets __________
   # of I/Os __________

22) On the Encounter command line, type `reportClock`.
    How many clock domains exist? ______

23) Click on the **Routing Color Control** bar (under All Colors).
    How many metal layers are defined? ______

   The 4.1 version of Encounter has command completion and history enabled by default. You can use the Tab key to complete commands and filenames and use the up and down arrow keys to get to previously typed commands.

24) Type the command

   `checkNetlist -outfile dtmf_chip_netlist.rpt`

   to run checks on the netlist, including undriven nets and pins, multiply-driven nets, undriven ports, etc.

**VIEWING THE DESIGN**

There are three modes for viewing a design. The View Icons appear on the middle-left of the Encounter GUI.

- **Floor Plan view**: Connections appear as fly lines when an object is selected. This view is used for moving or viewing floorplan objects such as guides and fences.

- **Amoeba view**: Routed interconnections can be displayed by double-clicking a module or block. This view is used mainly after placement to see the amoeba shapes created by the module boundaries in the placement.

- **Physical view**: The actual routed interconnect for specific nets can be viewed by zooming to a small area after Trial Route is completed. This view is used for detailed implementation and analysis after placement and routing. Use the Floor Plan view until after the design has been placed.
25) Select the Floorplan View tool widget to show the floorplan view in the Encounter user interface. Use the floorplan view until the design has been placed.

26) Zoom out to view more of the design.

- Pink objects on the left side of the core area are top-level, hierarchical instances represented as module guides.
- Blue-green objects to the right side of the core area are hard blocks.
- Green-grayish objects are black-boxes. (There are none in this design)
- Fly-lines are displayed when an object is selected and Net visibility is turned on in All Color/General Color Control – (the thin button under All Colors.)

Review this design by exploring the user interface features and bind-keys.

27) Select module DTMF_INST. Type SHIFT-G (ungroup) to graphically move down the hierarchy, and note that DTMF_INST is flattened from its location in the chip display area.

- Successively typing SHIFT-G will flatten another layer of hierarchy. If you do this make sure you type “g” to go back to the second level of hierarchy

28) To re-group the hierarchy, select one of the child modules or blocks and type g.

29) Ungroup the hierarchy again to answer the following:

   How many connections are there between DTMF_INST/TDSP_CORE_INST and DTMF_INST/RESULTS_CONV_LIST? ___________

30) Select All Colors to display the color and pattern assignments for each object type. You can change colors and patterns by clicking on the colored square next to an object and then selecting a new color and pattern from the selections.

- You can save color modifications by using Save Preferences => Save… and reload colors by using the All Colors icon and Load Pref File.

Explore Binding Keys

31) Type b to display an editor to view and define the bind key command settings. This can also be accessed under the Design => Preferences => Binding Key form. The default binding keys are consistent with Cadence Virtuoso™ products.
DESIGN BROWSER

32) Use Tools => Design Browser… to open the Design Browser.

   ❖ You can also click on the Design Browser icon on the toolbar.

   How many instances does DTMF_INST contain? __________

33) Click on the plus sign in front of the DTMF_INST to expand the tree for DTMF_INST.

34) Repeat for + Terms

   ❖ You can double-click on an instance to recursively descend into it.

35) Explore some of the different objects represented in the Design Browser.

36) Go to Floorplan => Clear Floorplan, select All Floorplan Objects, then click OK.

_________________________________________________________________________

Finished Introduction
Section 2: Silicon Virtual Prototyping

Silicon Virtual Prototyping Steps

- Design Import
- Floorplanning Basics
  - Module Guides/Fence/Region/Softguide Constraints
  - Relative Floorplanning
  - Automatic Power Planning (APP)
  - Multiple Supply Multi-Voltage (MSMV)
- Placement – AmoebaPlacement
- Prototype routing – TrialRoute
- Quick timing – timeDesign
  - Parasitic Extraction (RC Extraction)
  - Delay Calculation and SDF Generation
  - Timing Analysis
  - Min/Max Timing Analysis
- Clock Tree Synthesis (CTS)
- IPO with Global Physical Synthesis Optimization (GPS)
- Power Analysis

QUICK ZERO LOAD TIMING WITH timeDesign

- To determine if the constraints and netlist are reasonable, do a quick timing ignoring net loading to determine if you need to go back to the synthesis stage or revise your constraints.

  The timeDesign command is used at each stage of the flow to get the timing based on what information is available. Type the command: help timeDesign to see a list of options.

1) Type timeDesign -prePlace to check the initial timing.

2) What is the worst setup slack? ______
What type of path is it (reg2reg, reg2Out, etc)? ______
How long did this step take? ______
FLOORPLANNING BASICS *

If you skipped the previous step(s) you can start here by using Design => Restore Design => “dtmf_chip.enc” from the ./data directory.

3) Open the FloorPlan=>Specify Floor Plan… form.
4) Specify the die size by selecting Core Size by: Aspect Ratio and Core Utilization and type in 0.70 for core utilization and 0.85 for aspect ratio.
5) Under the default selection of Core Margins by: Core to IO Boundary, enter 25 for Core to Left, Core to Right, Core to Top, and Core to Bottom.
6) Click OK.
   ✤ Standard cell placement for the modules in the floorplan is driven by the placement of the module guides.
   ✤ First, move a module guide into the core area. Make sure you have the first level of hierarchy ungrouped to expose the four module guides below DTMF_INST.
7) Select the DTMF_INST/TDSP_CORE_INST using the Design Browser. Notice the TDSP_CORE_INST module is automatically highlighted.
8) Click on the Move/Resize/Reshape icon in the Tools section of the Encounter user interface or type SHIFT-R and use the left mouse button (LMB) to click once to move DTMF_INST/TDSP_CORE_INST into the core design area then click the LMB again to release it. Notice the mouse pointer changes to a black circle when in Move/Resize/Reshape mode. Multiple objects can be moved by clicking on them one-by-one with the LMB while holding down the shift key.
9) Resize the module guide by stretching an edge or corner with the LMB or reshape the module guide by moving an edge (Not a corner) with CTRL+LMB. Click LMB again when finished. Notice that the mouse turns into an arrow with a line when you are over an edge. Reshape keeps the utilization of the guide constant if the CTRL key is held down.

MODULE CONSTRAINT TYPES

✤ Modules imported into Encounter can be of several different types: a Guide, Soft Guide, Region, or Fence.
Guides, displayed in pink, guide placement by defining flexible containers for standard cells in a given hierarchy. Blocks associated with guides do not have to be pre-placed inside the guide.

Soft guides constraints are similar to a Guide constraint, but there are no fixed locations. Expect to see instances within a soft guide clustered closer together.

Regions are displayed in light orange and define an area for standard cells to be placed. Instances from other modules can be placed in a region.

Fences, displayed in orange, define an area for standard cells to be placed. Instances from other modules are not allowed to be placed in the fence area.

- Placement density controls the core utilization of a module guide. Decreasing the density may improve localized routing congestion. Density information is displayed in the upper left corner of a module guide with the TU=x% (Target Utilization).

**MODULE/BLOCK ATTRIBUTES**

10) Use the bind key q to query the attributes of any selected object. Select the DTMF_INST/TDSP_CORE_INST and press the q key. Note the information displayed.

11) Select the DTMF_INST/PLLCLK_INST and press the q key. Note the information displayed. You can also get this information in textual form. Type the command reportSelect to see this information printed in the command window.

12) Select the Move/Resize/Reshape tool widget and then select the PLLCLK_INST. Move the DTMF_INST/PLLCLK_INST into the core area and keep it selected.

13) Press the a key to return to Select mode.

14) Add a 10um placement blockage halo by selecting Floorplan => Edit Block Halo. Type 10 in all the boxes and click OK.

- Block Halos are a full placement blockage associated with a hard macro.
- Full placement blockages fully block the placer from placing cells in a region.
- Soft placement blockages block initial placement in that area, but allow cells
to be placed in that area during optimization, CTS and ECO placement.

- Partial placement blockages allow placement up to the specified utilization in a local area.
- Routing blockages obstruct routing from a certain area.

### RELATIVE FLOORPLANNING *

- Relative Floorplanning enables locations of blocks and/or module guides to be specified relative to the core boundary or previously placed blocks or module guides. This is a semi-automatic process that makes it easier to pre-place a large number of blocks or modules. Also, if the floorplan size is later adjusted, relative floorplanning maintains the relative placement of all the floorplan objects.

15) Clear the floorplan by using the Floorplan => Clear Floorplan … form.

16) Select All Floorplan Objects and click OK

**Place the DTMF_INST/PLLCLK_INST instance inside the core area relative to the bottom core boundary.**

17) Select Floorplan => Relative Floorplan... Select DTMF_INST/PLLCLK_INST. Click on the get selected button at the top-right of the form. This automatically fills in the instance name of the selected block.

18) Keep the original orientation.

19) Under New Location, select Relative to object, and use the arrow button to the right to select Bottom Core Boundary.

20) Keep Relation: Above and Align by: Left side selected.

21) Type 8 in both of the boxes to the right of Align by and Space.

22) Click Apply.

**Place the DTMF_INST/ROM_512x16_0 relative to the top-left boundary.**

23) Select the DTMF_INST/ROM_512x16_0 and click the get selected button to fill in Object name with the name of the selected instance.

24) Keep the original orientation. Set Space and Align by to 5.

25) Select Top_Core_Boundary and click Apply
Place the DTMF_INST.../RAM_256x16 instance relative to the bottom core boundary.win.

26) Select DTMF_INST.../RAM_256x16 instance, and click the get selected button.

27) Select Update to MX to flip the RAM_256x16 instance.

28) Select Bottom_Core_Boundary, Select Align to: Right Side and set Space and Align by to 0. Click Apply.

Place the DTMF_INST.../RAM_128x16 instance relative to the DTMF_INST.../RAM_256x16 instance.

29) Select DTMF_INST.../RAM_128x16 instance, and click the get selected button.

30) Select the DTMF_INST.../RAM_256x16 instance, and then select the get selected button next to the Relative to object field.

31) Select Above and Align to Right Side.

32) Enter a space of 10. Keep Align by 0.

33) Click Apply.

34) Click Save to save the relative floorplanning commands to a script.

35) Change directories to the work_fe directory.

36) Enter relative_fplan.tcl as the name of the file.

Change the die size and use the relative floorplan script to restore placement of floorplanning objects.

37) Select Floorplan => Specify Floorplan, and change Die size by: to 1200 wide by 1100 high. Click OK

38) At the encounter command line, type: source relative_fplan.tcl

   Notice that the relative placement of the RAM and ROM blocks have been automatically restored.

39) View the relative_fplan.tcl script to see the commands used.

Add block halos
40) Select all the hard blocks using the **Edit => Select by Name** menu option and entering *ROM_* _INST *RAM_* _INST *PLLCLK_INST* into the **Names** field.

41) Add a 15um block halo for placement obstruction using **Edit Block Halo**.

42) Select **Design=>Save Design....** Save the current design to **rel_fp.enc**.

---

### MULTIPLE SUPPLY/MIXED VOLTAGES (MSMV) *

- *If you skipped the previous step(s) you can start here by using Design => Restore Design => rel_fp.enc from the ../data directory.*

- MSMV enables multiple timing libraries to be bound to a single physical library. Connections to modules or instances define power and ground connectivity which are not specified in the Verilog netlist. Power Planning and Routing, AmoebaPlace, Delay Calculation, Timing Analysis, IPO, and Power Analysis all support MSMV. In the physical domain, power domains can be compared to a fence which defines the area where a power and ground pair will be supplied.

_During Design Import, global nets were defined._

43) Open the Design Import form and click on the Power tab.

- Power nets are **vdd** and **Avdd**.
- Ground nets are **vss** and **Avss**.
- Analog power and ground nets are **Avdd** and **Avss**. Power domains will be created to define design areas associated with **Avdd** and **Avss**. Click **Cancel**.
- Digital logic uses **Vss** and **Vdd**.

Connect global **vss** and **vdd** nets to power and ground pins.

44) Select **Floorplan => Global Net Connections...**

45) After **Pins**: type **VDD**.

46) Leave Instances as *, and type **vdd** after To **Global Net**.

47) Click **Add to List**.
48) Connect the VSS Pin to the vss Global Net in the same way as before. Click Apply. Click Close to close the Global Net Connections window.

Create an Analog power domain and assign the PLLCLK_INST and analog power pads to the Analog power domain.

49) Select Floorplan => Power Domain => Create, and enter Analog for the Domain Name.
   a) Click on the … button near the Timing Libs field to associate a timing library with the Analog Power domain, and select the plliclk library in the Specify Libraries window.
   b) In the Power/Ground net connections, under Power Net/Pins, type (Avdd:avdd!). This connects the global power net Avdd to pins named avdd!
   
   IMPORTANT: Make sure that you include the parentheses around the Nets/Pins argument
   
   c) Under Ground Net/Pins type: (Avss:agnd!).
   d) Under Modules, type: DTMF_INST/PLLCLK_INST and click Add to List.
   e) Under Power Net/Pins, type: (Avdd:VDD), and under Ground Net/Pins type (Avss:VSS).
   f) Enter IOPADS_INST/Pavss0 in the Modules Field and click Add to List.
   g) Enter IOPADS_INST/Pavdd0 under the Modules Field and click Add to List.
   
   h) Click OK to create the power domain. A red region named Analog appears on the left of the chip outside of the module guides. (Pan left or zoom out to see it.)

Move Analog Power domain into the core area.

50) Select the red Analog power domain region and select Floorplan => relative Floorplan…. Click the get selected button at the top of the form to populate it with Analog.

51) Select Relative to Object and Bottom_Core_Boundary using the arrow button.

52) Select Keep original for instance orientation, and Align to left side. Select 10um for both the Space and Align fields. Click Apply.
Reshape the Analog power domain region to match the size of the PLL instance.

53) Click **Reshape** on the relative floorplan menu. Select the **Analog** power domain. Click on the **get selected** button after the Object field.

54) Select **Same as object** and select the PLL instance. Click the **get selected** button. Click on **Apply**.

To allow standard cells to be placed inside the rectilinear region of the PLL, add a rectilinear cut to the Analog region.

55) Select the **Cut Rectilinear** tool widget.

56) Go to the upper right corner of the Analog power domain and click and release the LMB. Drag the upper right corner to the inner corner of the PLL instance (not the blockage) and click the LMB again.

57) Select **Design** => **Save Design** … and save the design to **Analog.enc**.

AUTOMATIC POWER PLANNER *

- If you skipped the previous step(s) you can start here by using **Design** => **Restore Design** => “analog.enc” from the ../data directory.

  - The Automatic Power Planner allows you to easily replicate a power plan for any design. To do this you can design a template of your power grid structure and then instantiate it.

Create Core Rings for vss and vdd around the core boundary using Metal5 on the Top and Bottom sides and Metal6 on the Left and Right sides.

58) Select **Floorplan** => **Power Planning** => **Edit Template** and on the **Design** tab enable the **Ring** and **Core Ring** options. Select nonplanar **M6/M5**.

59) Enable **Stripe** and select **M6**

**Setup a block ring around the PLLCLK block**

60) Click **IP Block** tab and select only **pllclk** from **IP Block List**.

61) On **Block Ring** tab, enable **Require block ring** and specify **M6/M5** with **Bit Width** of **5**

62) Now click **Set**. The block name (under **IP Block List**) changes to blue & bold.
63) Enter “pll” as IP Library Template Name and click Save

Save the design template

64) Return to the Design tab enter top as Design Template Name
65) For IP Library Template, click drop down arrow and select “pll”
66) Click Save; you will see graphic display of template to the right.

Instantiate the power template

67) Select the top template from the window on the right and click the Instantiate… button.
68) Enter only vss and vdd for the Nets.
69) Specify Width of 5 for both Core ring and Stripe.
70) Specify a Spacing of 1.5 and a Pitch of 150 for the Stripe, then click OK.
71) Click Close on the Edit Template form and take a look at the power plan created by APP. Notice that the block ring around the PLLCLK_INST block is merged with the core ring.
72) Save the design as “power_plan.enc”.

WISE-EDITOR *

If you skipped the previous step(s) you can start here by using Design => Restore Design => “power_plan.enc” from the ../data directory.

The wire editor is a full-featured tool for modifying/creating special wires and regular wires, adding shielding to nets, adding/modifying vias.

Use the Wire-editor to add power nets for the PLL and contour the power around the PLL.
Select the vdd and vss nets on the left of the PLL instance.

73) Select the vss net to the left of the PLL and then use SHIFT + LMB to also select the left vdd net.

74) Press the e key to bring up the wire-editor. Use the split net widget of the wire editor to break the vss and vdd wires into segments. (You can also use CTRL+s).

75) Delete the vdd and vss segments to the left and bottom of the PLLINST by holding the shift key and selecting each vdd and vss wire on the left of the PLL, then press the <Delete> key or use the delete net widget to remove them.
Power nets must be treated differently than normal wires.

76) Press e and under the **Nets** tab, check **Force Special** to draw non-default width wires. Enter **Avdd** and **Avss** for the net names.

77) Click the **Route** tab to enter wire widths and layer information. Select **M4** under vertical with a width of 5.0 um and spacing of 1.5 um. Use the **Add Wire** icon on the Encounter tool widget to add the vdd and vss wires. (Or use **SHIFT+A**, the add wire bind key.)

78) Move the cursor to the lower-left area of the die around (x=252 y=250). Coordinates are displayed to the right of the auto query button (in the lower-right corner of the user interface).

79) Click the LMB, then move the cursor to around x=252 and y=505. **Double-click** the LMB to finish drawing the wire segment. (Ignore the white violation markers indicating open nets; they will be connected to the PLLCLK_INST and the IO pads).

Use the wire editor to connect the PLLCLK analog power/ground pins.

80) Make sure that **InstPin** in the Color preference window is **on** so that the pins are visible.

81) Click on the **Nets** tab and type **Avss** in the **Nets** field. Select the **Route** tab, and enter **M3** for the **Horizontal Layer** and width of 5.01.

82) Press the **< SHIFT+A >** bind key to draw the Avss net.
83) Move the cursor over one of the agnd! pins of the DTMF_INST/PLLCLK_INST instance (as you hover over the pins, the pin name is shown at the bottom of the GUI).

84) Click the LMB and move the cursor over the Avss net and **double-click** the LMB to complete the route.

85) Repeat for the other agnd! pin.

86) Click on the Nets tab and type **Avdd** in the Nets field.

87) Route the two **avdd!** Pins to the Avdd net.

*Use the wire editor to connect the IOPADS_INST/Pavdd0 and Pavss0 pins to the analog nets.*

88) Click on the Nets tab and type **Avdd** in the Nets field.

89) Select the Route tab, and enter **M3** for the Horizontal Layer and **width of 26.54**. Press the < a > bind key to draw the Avdd net.

90) Move the cursor over the VDD pin of the IOPADS_INST/Pavdd0 instance.

91) Click the LMB and move the cursor over to the Avdd net and then **double-click** the LMB to complete the route.

92) Change the Nets tab to net Avss and connect the IOPADS_INST/Pavss0 instance.

93) Click **Close** to finish Wire Editing.

*Connect the digital Pad pins to the core rings using Sroute auto-routing.*

94) Open the **Route => Sroute** form.

95) Enter **vss** and **vdd** under the Nets field.

96) Select **only** Pad Pins. Click OK.

*Trim the net segments to clean up the power routing.*

97) Select all nets using the < d > bind key. Select Objects: **All**. Select Type as **Special** and click OK. This selects all the special nets in the design.

98) Bring up the wire editor by using the < e > bind key. Use the **Trim Selected Nets** icon to trim all the special nets.
99) Save the Design as “power_an.enc”
PROTOTYPING PLACEMENT *

If you skipped the previous step(s) you can start here by using Design => Restore Design => “power_an.enc” from the ../data directory.

100) Load a previously saved floorplan file using Design => Load => Floorplan. Enter ../data/dtmf_chip.fp. Click OK.

101) Select Place => Place… Review options for running AmoebaPlace™. Click OK.

Default placement is medium effort and congestion driven.

How long did AmoebaPlace™ take to complete? ____________

102) Now unplace the standard cells by selecting Floorplan => Clear Floorplan. Select unplaces Placed instances. Make sure only that entry is selected.

103) Run AmoebaPlace™ in the Prototyping mode to get a quick initial placement.

   This is a quick way to place cells in a large design for evaluating your floorplan. For even faster placement on a large design where you may run into capacity issues, you may want to use cluster-based placement.

104) Select Place => Place… Select the Prototyping option in the Place window, and click OK.

   How long did timing-driven AmoebaPlace™ take to run? ____________

The Amoeba view shows the outline of the modules, and is an indication of how well cells of a given module are clustered together.

105) Move to the Amoeba view.

106) Select the TDSP_CORE_INST and then select the detailed physical view. Notice the cells belonging to the TDSP_CORE_INST are highlighted in the physical view.

PROTOTYPE ROUTING WITH TRIALROUTE™

TrialRoute is a very quick router which is used to estimate congestion and parasitic data early in the design cycle. The results of trialRoute can also be
used for better pin-assignments while partitioning a design. NOTE: TrialRoute does not guarantee a DRC-free design.

Route the design limiting the number of layers to 3.

107) Select Route \texttt{\rightarrow} Trial Route.

108) Select 3 for the Max Route Layer. Click OK.

109) On the Routing Color Control panel, turn off visibility of Net, Snet, and instances.

Congestion is displayed in SOCE by red diamonds.

110) Click the General Color Control button (thin bar under All Colors).

111) Turn on Vcongest, and Hcongest to display horizontal and vertical congestion markers.

112) Zoom into the congestion in the center of the die.

\begin{itemize}
\item The numbers show the number of tracks required versus the number that is available.
\end{itemize}

113) Review the congestion distribution report in the Encounter console window or the encounter log file.

\begin{itemize}
\item “usage” summarizes horizontal and vertical tracks used in gcells.
\item “OvlnObst” summarizes obstructed gcells, and “Overflow” summarizes all overflowed gcells. A gcell has overflowed when the track demand exceeds the supply.
\end{itemize}

114) Re-route and Select 7 for the Max Route Layer. Select low Effort. Click OK. This design has no congestion using all 7 layers of metal.

115) Save the design as “trial_route.enc”.

**RC EXTRACTION**

The FE default extraction engine provides accurate but very fast extraction. It uses a cap table for more accurate estimates.

Run FE default extraction.

116) Select Timing \texttt{\rightarrow} Specify Analysys Condition \texttt{\rightarrow} Specify RC Extraction Mode and select default.

117) Select Timing \texttt{\rightarrow} Extract RC. Select Save SPEF To: dtmf_chip_default.spef.
How long did it take to run FE default extraction? __________________
How long did it take to save the SPEF file? _________________________

PRE-CLOCK OPTIMIZATION WITH GPS *

If you skipped the previous step(s) you can start here by using Design => Restore Design => “trial_route.enc” from the ../data directory.

During prototyping, to get an idea of whether a design can meet timing, run a low-effort optimization. Low-effort optimization performs basic buffer insertion for long nets, and resizes sequential and combinational cells. Medium and high effort further optimizes the critical path by using pin swapping and physical restructuring for designs that are more difficult for timing closure.

Run an iteration of low-effort IPO with Global Physical Synthesis.

118) Type setOptMode –lowEffort

119) Run GPS optimization in the pre-clock tree mode by typing optDesign –preCTS

This can also be run from the Optimization GUI. We will explore optDesign and the Optimization GUI more in the Implementation section.

TIMEDESIGN

The timeDesign command runs extraction, trialRoute, and timing analysis all at once. All you need is a placed design. The different modes for pre and post CTS, and post Route automatically choose the best modes for the routing, timing analysis, and extraction.

120) Type timeDesign –preCTS at the encounter prompt to get the timing prior to optimization.

Examine the slack report generated after IPO.

121) Select Timing => Timing Debug => Slack Browser.

122) Double click on the timingReports directory, and select the dtmf_chip_preCTS_setup.slk file. You make all files visible by using the pulldown menu at the bottom.

123) Click Open.
Are there any violations left in the design?
_____________________________
What module are they under?
_____________________________

CLOCK TREE SYNTHESIS (CTS) *

- CTS requires a specification file prior to inserting the clock trees. The specification file can be automatically created using the SDC constraints. The specification file contains the clock roots for all the clocks specified in the SDC constraint file, skew and transition targets, and buffer types to use for CTS.

Create a clock tree specification file.

124) Select Clock => Create Clock Tree Spec...
125) Type buf in the Buffer Footprint field.
126) Type invd1 in the Inverter Footprint field.
127) Click OK.
128) Examine the constraints in the dtmf_chip.ctstch specification file.

What is the skew target for refclk?____________

129) Select Clock => Specify Clock Tree. The Clock Tree File should be dtmf_chip.ctstch.
130) Click OK.

Insert the clock tree.

131) Select Clock => Synthesize Clock Tree.
132) Click OK.

- CTS automatically generates HTML files that can be used to examine the clock tree results, or alternatively the Clock Tree Browser can be used to graphically display the clock tree.

Have all the constraints for this design been met? ______________

Display the Clock Tree Browser window. This can be used to examine the clock tree topology as well as modify, resize, or delete elements in the clock tree.
Select Clock => Clock Tree Browser.

133) Select DTMF_CHIP/TEST_CONTROL_INST/MCLK_ROOT/CN.
134) Click Select.
135) Click OK.
136) Select Display => Trigger Edge Delay to show a delay histogram of the triggering edge of the clock.

Display Min and Max clock paths in the GUI.

137) Select Display => Physical View => Min/Max paths.
What is the longest path from the m_clk clock root? _______________
138) Save the flat prototyping results using Design=>Save Design… to dtmf_chip.proto.enc.
139) Check timing post-CTS
140) Type timeDesign –postCTS at the encounter prompt
Was the timing better or worse? _____

STATISTICAL POWER ANALYSIS

If you skipped the previous step(s) you can start here by using Design => Restore Design => dtmf_chip_proto.enc from the ../data directory.

IR drop can be estimated during the prototyping stage using the FE power analysis tool. IR drop can decrease your performance. The FE power analysis tool is a quick way to get early feedback on the quality of the power grid. During implementation, VoltageStorm should be used for greater accuracy.

There are two ways of performing power analysis in FE: Statistical or Dynamic. Dynamic is simulation-based and requires a VCD (Verilog Change Dump) file. This section covers statistical analysis.

Before analyzing power, RC parasitics must be extracted, and the power sources must be determined.

141) Run TrialRoute and Timing=>Extract RC...
142) Select Power => Edit Pad Location.
143) Enter **vdd** for the net name and select **Auto Fetch** to determine the pad locations. You should see 4 entries in the Pad Location List. The power sources are marked with a yellow circle in the Encounter GUI.

144) Save the power pad locations to a file. Select **Save**, and enter **dtmf_chip_vdd.pp** in the field for the filename.

145) Repeat the steps above for vss.

146) Save the power pad locations to **dtmf_chip_vss.pp**.

147) Select **Cancel** to close the **Edit Pad Location** menu.

   For statistical analysis, you give Encounter an estimate of the net toggle percentage. There are three ways to do this: using a dummy clock and assuming all nets in the design toggle at a single rate, or use pre-cts or post-cts toggle files which divide the design into clock domains. Each domain’s clock rate and toggle probability is used.

148) Select **Power => Edit Net Toggle Probability**.

149) Select **Get Clock**.

   The clocks from the SDC file will be populated in the form. The frequencies are determined from the SDC. By default the toggle probability is 20%. You can specify different toggle probabilities by selecting a clock, selecting **Edit**, and then modifying the Net Toggle Probability.

150) Click **Save** to save the toggle file (use the default filename).

151) Click **Cancel** to close the form.

   *Analyze the power grid.*

152) Open the **Power => Power Analysis => Statistical** menu. After Net Names, enter **vdd**.

154) Select **post-CTS** clock.

155) Select the file browser icon on the right of the **Net Toggle Probability Field**.

156) Select **dtmf_chip.tg**.

157) Select **Open**. Select the file browser icon on the right of the Pad Location File.

158) Select **dtmf_chip_vdd.pp**.

159) Click OK
Gather some statistics after power analysis.

What is the total average power consumed? _________________________

What is the leakage power? ________________________________

(You can optimize the design to lower the leakage power during implementation).

Which part of the design consumes the most power: core, io or hard-blocks? _______________________

What is the worst-case IR drop? ______________________________

Display the IR drop results.

160) Select Power => Display => Display Rail Analysis Results.

161) It’s easier to view the results if the visibility of instances, nets, and special nets (Snet) are turned off. Use the All Colors form to turn these off.

162) Enter vdd after the Net Name field.

163) Select IRD to display the Encounter IR drop results.

164) Select Update filter range. This displays the IR drop by color. The color varies from green (low IR drop) to red (threshold exceeded).

165) The default threshold is 10% of the supply voltage (or .108 V). Change the threshold to .003 by typing in .003 in the IRD Threshold field.

166) Select Update Filter Range.

167) Click OK.

Areas in red are where the IR drop exceeds the .005 voltage threshold.

What area(s) of the design has the worst IR drop?

____________________________________

168) Exit Encounter. Select Design => Exit or type exit at the encounter prompt.

Finished Silicon Virtual Prototyping
Section 3: Partitioning

Partitioning Steps

Although the design used in this lab would not be implemented hierarchically due to its relatively small design, this workshop provides a good demonstration of the required flow steps. From a floorplan perspective, the macros to be partitioned follow the logical netlist hierarchy, have fence constraints, and have properties for pin placement and obstruction.

The result of partitioning a design is a directory containing constraints (SDC, Primetime, or PKS TCL), timing models (STAM, .LIB, .TLF), physical models (LEF, DEF, .FP), and netlists. This lets you implement the design in parallel with smaller amounts of data.

Block models for timing (SPEF, STAMP, TLF), SI (CDB), and antennae (LEF) are created and used for top-level analysis. You can reassemble the completed blocks and top-level design for flat final analysis during the chip assembly phase.

- Partitioning
  - Specify partitions
  - Specify Black Box modules
  - Feedthrough techniques
  - Route-based buffer insertion
- Committing Partitions
  - Budgeting
  - Pin Generation
- Releasing the Blocks
LOADING THE RE-GROUPED NETLIST *

If you skipped the previous step(s) you can start at step 1

- There are times during the physical implementation that the logical and physical hierarchy should be modified. For instance, if there are a lot of nets communicating between several modules, those modules are candidates for netlist regrouping, or if the designers want to simplify the flow and work with a smaller number of hierarchical blocks during implementation. Encounter™ provides the capability of regrouping the original netlist into a different level or physical and logical hierarchy and regenerating a new netlist.

During the partitioning stage, only the floorplan and constraints are used.

What are the advantages and disadvantages of a hierarchical design?

1) cd /home/icd_train/SOCE4.1_WORKSHOP/work_hier
2) Launch SOC Encounter from the Linux prompt by typing:
   encounter
3) Select Design=>Restore Design, select ../data/dtmf_chip_restruct.enc. Click Open.
4) Select the DTMF_INST module and ungroup it (bind key: ‘SHIFT-G’).
5) Select the Design Browser and push down into the DTMF_INST (select the ‘+’ key).
   - The name of the regrouped netlist is TOP_GLUE_REGROUP_PH.
6) Push down into the TOP_GLUE_REGROUP_PH module.

What modules are included in the TOP_GLUE_REGROUP_PH module?

1) ________________________________________________________________
2) ________________________________________________________________
3) ________________________________________________________________

7) Select File => Quit to close the Design Browser Window.
LOAD FLOORPLAN/SPECIFY PARTITIONS

- To save time, a floorplan has already been created to demonstrate a hierarchical flow.

8) Restore the floorplan file using **Design => Load Floorplan**.

9) Use the file browser to locate `~/soce_workshop/data/dtmf_chip_part.fp`.

10) Double click the file name or click **Open**.

11) The floorplan contains (2) pre-placed fence regions.

- The (2) fences are partitioned for hierarchical implementation. Note that the DTMF_INST/TDSP_CORE_INST will be rectilinear. (The rectilinear shape was created with the Cut Rectilinear tool widget.) This rectilinear cut will provide a top-level routing channel around the PLL block.

The first step in partitioning a design is to specify which of the module guides represented by the logical hierarchy should be converted to partitions. This step has already been done, and the information saved in the floorplan file that has been restored. When modules are partitioned, they are automatically converted to fence constraints.

12) Examine the specify partition form, **Partition => Specify Partition**. Note that there are two modules in the Partition List. Also examine all the options available when partitions are specified: Core to left spacing, ability to reserve layers for lower-level partitions, pin pitch, etc.

What routing layers have been reserved for the results_conv partition?

-------------

What layers have been reserved for tdsp_core?

-------------
PARTITION – SPECIFY BLACK BLACK BOX *

- In the early stages of the design cycle, synthesizable RTL is not always available for all of the blocks in the design. These blocks can be implemented as Black Boxes in Encounter™. The flow for creating Black Boxes is greatly simplified in version 3.1 and beyond for SOCE.

Convert ulaw_lin_conv to a black-box

13) By default modules with an instance count greater than 100 are displayed. Change the instance count for displayed modules to 60.

14) Select Design => Preferences. Select the Display Tab.

15) Enter 60 for Min. Floorplan Module Size

16) Select Partition=>Specify Black Box Enter the module name:
   ulaw_lin_conv

17) Select Width, and enter Width: 140 Height: 153.

18) Select Add/Replace.

19) Select OK ..

   Notice the pink module guide has changed colors to dark green and is now located on the right-hand side of the core area.

Convert the Black Box to a guide constraint (There are some limitations with Black Boxes and buffer feedthrough which will be implemented later in the flow).

20) Select Partition => Specify Black Box.

21) In the Black Box List, select the ulaw_lin_conv partition.

22) Select Delete then Click OK.
PARTITIONING – PLACE & ROUTE

If you skipped the previous step(s) you can start here by following steps you can start here by following steps 1 & 13-15 above

- In SOC-E pin locations are based upon the results of TrialRoute™. This is a unique approach that greatly simplifies pin optimization and minimizes top-level routing congestion. TrialRoute™ has the ability to anticipate the top-level obstructions resulting from partitioning.

  AmoebaPlace can also be run in timing-driven mode. Timing-driven placement automatically performs tradeoffs between meeting timing and routability. Timing-driven placement performs Virtual IPO (sizing and buffering) to improve timing closure quality of the placement.

23) Before Commit Partition, the design must be placed and routed. Run timing driven AmoebaPlace™ by going to Place => Place and selecting the Timing Driven option.

24) Review the encounter.log file. Note that the placer reports statistics regarding the fence utilization.

25) Examine the placement results of each Fence by selecting it in the Floorplan view and moving to the Amoeba and the Physical views. Notice instances belonging to the Fence are constrained inside the Fence boundary.

- In SOC-Encounter, pin locations can be determined by the results of TrialRoute. This approach greatly simplifies pin optimization and minimizes congestion at the top-level. TrialRoute can also anticipate the top-level obstructions results from partitions.

26) Open the form Route => Trial Route..

27) Select the Handle Partition option and click OK

- You can limit the number of layers available for the lower-level partitions (Remember the Specify Partition form has options for layer selection).
PARTITIONING – BUFFER INSERTION

- There are several options for handling nets that cross over partition blocks…
  1. Insert buffers into the module partitions (Workshop approach).
  2. Insert Hole-Punch buffers (Placement islands). Areas are reserved for buffer insertion at the top-level. Lower-level netlist is not modified; placement blockages are created in the lower-level partition.

Insert repeaters on high fanout nets

28) Type `insertRepeater` at the command prompt.

Insert buffers into the lower-level partitions…

29) Select Partition => Show Wire Crossing. This generates the file, `dtmf_chip.wirecrossing`

30) Edit the `dtmf_chip.wirecrossing` file and remove all nets except for the `tdsp_port` nets.

31) At the encounter prompt type: help `insertPtnFeedthrough`

33) Then type:
   
   `insertPtnFeedthrough -selectNet dtmf_chip.wirecrossing -bufCell BUFFHVT22 -routebased`

34) Use the Design Browser to Highlight the newly placed buffer called `FE_FEED`.

- The design status has changed to Placed because of the newly inserted buffer).
COMMIT PARTITIONS

35) Check the worst timing slack by typing
   
   `timeDesign -preCTS`
   
   *This routes the design and shows you the worst slack.*

36) View the worst path by typing
   
   `reportViolation -num 1`
   
   - *Notice in the header of the violation report that the worst slack path is a reg2reg path running through the tdsp_core module*

Generate the partitions, assign pins and perform time budgeting.

37) Select the *Partition => Partition* form.

38) Under *Perform Pin Assignment*: select *Disallow pins above or below power/ground strips*.

39) Select *Derive Timing Budget* and *Conflict Resolution: Most Critical*.

40) Click *OK*.
   
   - *Notice that the orange fence constraints have changed into green hard macros and that the routing and cells inside the partitioned modules are no longer visible.*

Change Partition Views

- *You can traverse the design by pushing and popping in and out of partition views. Notice that power and obstructions have been pushed down into the lower level partitions.*

41) Select the *DTMF_INST/TDSP_CORE_INST* partitioned instanced.

42) Select *Partition => Change Partition View*.

43) Select the physical view.

44) To return to the top-level design, select *Partition => Change Partition View* again.
RELEASE THE BLOCKS

Save the partitioned design to work directories.

45) Make sure that the current view is the top-level design.

46) Open the Design => Save Partition... form.

47) Leave all defaults selected.

48) Specify the partition result directory name PTN. Click OK.

49) From another xterm window, explore the directories and files saved into the directory

/home/icd_train/SOCE4.1_WORKSHOP/work_hier/PTN

What files/directories are in the PTN directory?

What files/directories are in the PTN directory?

- While it is possible to traverse the newly created partitioned hierarchy, the recommended flow is to exit the Encounter session and move to the newly created partition directories for implementation. Select Design => Exit.
Section 4: Block Implementation

Block Implementation Steps

This section focuses on the detailed implementation of the tdsp_core block that was partitioned earlier. Here we will explore the flow needed to close timing at the block level. The result will be an optimized, placed, routed sign-off design that meets timing and signal integrity requirements.

- Global Physical Synthesis Optimization (GPS)
  - Advanced optimization techniques:
    - Restructuring
    - Multi-VT (Leakage Power Optimization)
    - Useful Skew
    - optDesign command
- Clock Tree Synthesis (CTS)
- Native Nanoroute™
- Exploring the worst path
- CeltIC Analysis and Repair (crosstalk fixing)
- Sign-off timing with timeDesign
  - SignalStorm™ delay calculation.
  - Fire & Ice Extraction™
- VoltageStorm™ Power Analysis.
- Verplex Formal Equivalence Analysis
LOAD THE TDSP_CORE PARTITION

1) To go to the tdsp_core partition directory, type the following:
   
   cd ~/SOC4.1_WORKSHOP/work_hier/PTN/tdsp_core

2) Type encounter to launch SOC Encounter from the linux prompt.

Open the partitioned block in Encounter.

1) Select Design => Design Import… and Load.

2) Select tdsp_core.conf and click Open then OK to close the design import form.

3) Select Design => Load => Floorplan and open the tdsp_core.fp file to load in the floorplan.

Analyze timing prior to optimization. Run worst-case timing analysis and generate a slack report.

- AmoebaPlace can also be run in timing-driven mode. Timing-driven placement automatically performs tradeoffs between meeting timing and routability. Timing-driven placement performs virtual IPO (sizing and buffering) to improve the timing closure quality of the placement.

4) Run a timing-driven AmoebaPlace™ by typing:
   amoebaPlace -timingDriven

5) Run a pre-CTS timing analysis by typing:
   timeDesign -preCTS

What is the worst setup slack? __________
PRE-CTS OPTIMIZATION WITH USEFUL SKEW

- The optDesign command and GUI, available in 4.1, use next generation physical optimization algorithms for timing closure which is what we call Global Physical Synthesis or GPS. These commands automatically use transforms appropriate for the current stage of the flow. This is the recommended command to use for timing closure.

Optimization in SOCE uses many advanced optimization techniques including:

- resizing
- buffer insertion and deletion
- global restructuring of the critical path

All of these use physical information to improve timing. Instances can be moved during optimization to improve timing and useful skew optimization can also be used to achieve timing closure.

There are two complementary approaches for useful skew optimization. One is a pre-CTS approach which advances clock signals for critical path startpoints. The other approach is post-CTS which delays clock signals for critical path endpoints. optDesign does both of these depending on what mode you are running in.

Explore the Encounter Optimization Options

6) Select **Timing => Optimization...** and take a look at the optDesign GUI
7) Click the **Advanced** button to see all of the transformations that can be performed then click **Cancel** on the form.

*Enable Encounter advanced optimization techniques with useful skew to improve timing.*

8) Change the **Mode:** to **preCTS** the **Effort:** to **high** and **Target Slack:** **Setup Time:** to **0.3** and select the **Useful Skew** option.

9) Click **OK** to perform a pre-clock tree optimization.

   - What is the slack after optimization? __________
   - What is type of path is the worst case? __________

10) Examine the Encounter log file to see the optimization transforms that took place: resizing, restructuring in the critical path, downsizing in instances off the critical path, and a list of instances which were advanced.

11) Search for the keywords, **Boolean restructuring** and **Moving Instances** in the Encounter log file.

12) View the file **scheduling_file.cts** that was written out by optDesign. This file, which is created by pre-CTS useful skew optimization, is a scheduling file and a latency file. (if you do not see this file then there was no opportunity for
useful skew optimization). Look in the ..../data directory for an example of these files.

**CTS + CONCURRENT NANOROUTE™**

CTS analyzes all the clocks in the design and determines the best topology to minimize skew. Routing and CTS can be done concurrently using the NanoRoute™ Ultra SoC routing solution. NanoRoute ensures correlation with post-route timing on the clock tree. CTS can also handle gated, reconvergent, and crossover clocks.

Enter the scheduling file created from the pre-CTS useful skew optimization. (if you do not have this file skip to step 16)

13) Select **Clock => Specify Clock Tree.**
14) Browse and select the file **scheduling_file.cts**
15) Click **Apply**.

Enter the clock tree specification file with the constraints for the clock tree.

16) Use vi to view the Clock Tree specification file:

```
vi../data/tdsp_core.ctstch
```

(Note: This is the specification file in the ..../data directory, not the one in the current directory).

17) Search for **AutoCTSRootPin refclk**. Note the keyword: RouteClkNet YES. This enables concurrent routing with NanoRoute during CTS.

Enter the clock tree specification file.

18) Select **Clock => Specify Clock Tree.**

The **Clock Tree File** should be ..../data/tdsp_core.ctstch. Make sure to load the cts specification file in the ..../data directory, not the one in the current directory.

19) Enter **OK**
Run CTS to insert the clock trees and route the clock nets with NanoRoute.

20) Select **Clock => Synthesize Clock Tree**.
21) Select **Save Clock Tree Macro Model**. Click **OK**.
   - This Macro Model will be used during top level optimization to model the insertion delay and skew of the block during top-level CTS.
22) Change view to the physical view using the tool widgets on the left of the Encounter™ GUI.

Use the Design Browser to select the net: `DTMF_INST/m_clk`.

23) Select **Tools => Design Browser**.
24) Select **Net** after the Find field, and enter **clk** in the text field.
25) Use the **Zoom Selected** button to zoom into the selected net.
26) Turn off visibility for **Rout Blkg** for better viewing.

Check timing results after CTS. Run **timeDesign**.

27) Type: `timeDesign –postCTS`
28) Run a post-clock optimization using **optDesign –postCTS**
   - Notice that because of the –usefulSkew option set earlier with `setOptMode`, `optDesign` is adding buffers to the clock nets to help meet timing. Also, since the worst timing path is a reg2out path, it can be fixed at the top level.

DETAIL ROUTE WITH NATIVE NANOROUTE

If you skipped the previous step(s) you can start here by using **Design => Restore Design => “tdsp_core_cts.enc”** from the ../data directory.

Native NanoRoute is a full integration of NanoRoute running directly on the Encounter database. NanoRoute can perform timing-driven routing, SI (signal-integrity) driven routing, and optimization.

29) Open the form **Route => NanoRoute…**
30) Select **Insert Diodes** and specify **ANTENNALVT** for the cell name.
31) Select **Timing Driven**. (Normally, you would also check the **SI Driven** option, but some noise delay examples are needed for workshop purposes.) Click **OK**.
Are there any remaining antenna violations? ____________________
Are there any remaining DRC violations? _______________________

OPTIMIZING TO MINIMIZE LEAKAGE POWER

- *Encounter enables you to reduce leakage power without degrading timing by using a mixture of high and low voltage threshold cells.*

*Report the total leakage power.*

32) At the encounter command window, type:
   ```
   reportLeakagePower
   ```

33) Look at the report of the total leakage power in uW.
   - What is the total leakage power? ______
   - How many low-vt cells are in the design? _____
   - What percentage of the total leakage power is used by the low threshold (low vt) cells?_______________

*Run leakage power optimization.*

34) At the command prompt, type:
   ```
   optLeakagePower
   ```

35) Report the total leakage power again. (You can use the up arrow to go back in the command history).
   - What is the total leakage power? ____________________________
   - How many low-vt cells are in the design? _________________
   - What percentage of the total leakage power is used by the low-vt cells? _________________

- *Encounter has replaced low threshold gates with high threshold gates (while maintaining timing). To check this, you can run `timeDesign -preCTS` again.*
DELAY CALCULATION AND TIMING ANALYSIS

- Delay calculation and STA involves determining if a design can meet Setup or Hold-time requirements under certain operating conditions. For prototyping, use the native FE delay calculation. SignalStorm delay calculation can be used for greater accuracy during the implementation and assembly stages.

First set the operating condition.

36) Select Timing => Specify Analysis Condition => Specify Operating Conditions/PVT.

37) Select the WCCOM operating condition for Max, and select BCCOM for Min.

Run worst-case timing Analysis and generate a slack report.

38) Select Timing => Timing Analysis and run the default Setup Time Analysis. Click OK.

39) Open the slack file in the current directory.

40) View the slack report by using Timing => Timing Debug => Slack Browser. The 50 worst-case paths are displayed.

Display the worst-case path in the Encounter™ GUI.

41) Using the LMB, double click on the first path in the slack browser window. The worst-case path is highlighted in the GUI. Detailed delay information is reported in a window below the path browser window.

42) Double click on any instance in the path to highlight and zoom into that instance. You can also browse the path at a high-level module view in the bottom of the slack browser and push down into the hierarchy to see the driver and receiver (You may need to expand the Slack Browser window).

   What is the slack of the worst-case path?________
   How many violations are there?__________________
Generate and save an SDF file.
Select **Timing => Calculate Delay.** Click **OK.**

**PATH GROUP SUPPORT**

- **SOC-E** supports path groups both in timing analysis and optimization. Define path groups using the `setClockDomain` command. Often times, I/O paths are over-constrained. Path groups are a good way of giving priority to register-to-register paths. The `optDesign` command optimizes all paths first, then the path groups such as `reg2reg` and `reg2Out`.

**POST-ROUTE OPTIMIZATION WITH SI DELAY**

- In post-route mode, `timeDesign` and `optDesign` use the detailed RC extraction mode. Detailed RC extraction uses the same capacitance calculation engine as *NanoRoute* and considers 2 layers up/down for extraction. The `EXTENDED_CAP_TABLE` section of the capTable is required for this analysis. Several options have been added including estimation for metal fill.

43) Type **optDesign --postRoute**

   What is the worst slack reported by `optDesign`? _______

*Now run the optimization to include the SI delay effects using integrated CeltIC SI analysis*

Type **optDesign --postRoute --si**

   What is the worst slack reported by `optDesign`? _______

**CELTIC ANALYSIS AND FIXING**

- **SI** prevention, analysis, and repair can be run within the SOC Encounter cockpit. The integration of CeltIC sign-off mode into SOC Encounter greatly simplifies the flow for crosstalk analysis and crosstalk fixing.

Run **setFeReport** then **timeDesign --postRoute** to check the timing without SI delay.
Run CeltIC Analysis and generate a hierarchical noise view of the block.

44) Select SI => Run CeltIC Crosstalk Analysis.

45) Keep Mode as SignOff.

46) Process should be set to 130nm. Noise Threshold: 20% of Vdd. Switching Windows: Native TWF.

47) Under Create Block Noise Model: select cdB. Click OK.

48) Examine the failures reported.
   How many noise nets were reported? _______
   What is the new reported slack after SI delay is included? _______

Display the noise nets.

49) Select SI => Display Noise Nets… Click OK.

50) Double click on the first net in the CeltIC Result Browser window. The victim net is highlighted in the GUI. It is easier to view the net topology if you turn off visibility of Insts, Nets, and SNets in the Color Control Panel.

51) Select Browse Aggressors. The list of aggressors for this net is displayed in a Design Browser window along with the magnitude of the noise glitch for each aggressor. Single-click each aggressor net to see it highlighted in the GUI.

52) Close the Design Browser window.

RUN SIGN-OFF EXTRACTION AND TIMING *

- Fire & Ice QX™ is an accurate, high-capacity, high-performance, gate-level parasitic resistance and capacitance extractor for timing and SI sign-off. SignalStorm™ is a delay calculator that calculates instance-specific IR-Drop based delays including interconnect RC loads. In SoC Encounter, the timing models are based on the .lib timing library. In sign-off mode, timeDesign uses both of these for highly accurate timing.

Add Filler Cells.

53) Select Place => Filler=>Add Filler…
54) After Cell Names, type in the name of the filler cells: Enter **FILL64 FILL32 FILL16 FILL8 FILL4 FILL2 FILL1**. (You can also click **Select** and add them all at once using the shift key, then click **Close**.)

55) Click **OK** to insert the filler cells.

56) Zoom in or click Redraw to view.

*Use timeDesign to run sign-off timing analysis.*

57) Select **Design => Design Import** and change to the **Misc.** tab.

58) Use the Browse button to select the QX Library Directory: `../../../LIBS/RC/QX/7lm`.

59) Use the Browse button to select the QX Tech File: `../../../LIBS/RC/icecaps.tch`.

60) Type `timeDesign –signOff` and view the results.

61) View the report file in `timingReports/tdsp_core_signoff_reg2reg.rpt`. If **reg2reg** timing is not met, re-run steps and then run this stage again.

### RUN SIGN-OFF POWER ANALYSIS *

- **IR drop is a critical factor in SoC performance. It affects timing and clock skew. The VoltageStorm® PE power integrity verification solution allows for complete power grid verification, IR drop and Electromigration(EM) analysis, and current flow and current density analysis.**

Run VoltageStorm to analyze the power grid.

62) Select **Power => Run VoltageStorm** and fill in the parameters.

**Net Name**: vdd

**Voltage Limit**: 0.972 (By default this is 90% of the supply voltage)

**Library List**: `../LIBS/RC/QX/7lm`

**Instance Power File**: `../data/tdsp_core.power.vdd` (This file was created by running native power analysis.)

**Power Pad Location File**: `../data/tdsp_core_vdd.pp` (This file was created by running native power analysis)

**Analysis Type**: IR (IR Drop Analysis)
63) Using the Advanced Tab on the Voltage Storm Analysis form, select **Keep Temp Files**. Specify an output directory: vstorm_results.

64) Click **OK**

65) Display the results using **Power => Display => Display Rail Analysis Results**…

66) Use the same display techniques described during flat prototyping to improve the visibility of results.

67) Specify the **Input Directory**: vstorm_results/VDD_25C_avg_2 and the **Net Name**: vdd.

68) View results for IR (IR Drop). Click **Apply**.

**SAVE THE PARTITIONED BLOCK**

70) Generate a timing model for hierarchical timing by typing: `genTlfModel`

71) Save the design to `tdsp_core_done.enc`

---

**Finished Block Implementation**
Section 5: Hierarchical Implementation

Hierarchical Implementation Steps

Although the design used in this lab would probably not be implemented hierarchically, this portion of the workshop demonstrates many of the required flow steps. From a floorplan perspective, macros which are partitioned follow the logical netlist hierarchy, have Fence constraints, and have properties for pin placement and obstruction.

The result of partitioning a design is a directory containing constraints (SDC, Primetime, or TCL), timing models (STAMP or .LIB), physical models (LEF, DEF, .FP), and netlists. This allows blocks of the design to be implemented in parallel with smaller amounts of data.

Block models for timing (STAMP, TLF), SI (CDB), and antennae (LEF) are created and used for top-level analysis. The completed blocks and top-level design can be reassembled for flat final analysis. For more on the benefits of a hierarchical design methodology please see Appendix A in this document.

- Top-level optimization
  - Timing models from lower-level blocks

- Hierarchical CTS
  - Using clock models

- Hierarchical CeltIC
RUN TOP-LEVEL TIMING CLOSURE NATIVE NANOROUTE™

For the top level partition, run optimization with the models created during the block implementation to get an accurate timing and SI analysis.

1) Change the working directory to `~/work_hier/PTN/dtmf_chip`

2) Launch an SOC Encounter™ session and begin the top-level optimization by typing:
   `encounter -replay ../../../scripts/FE_hier/soce_hier_impl.tcl`

3) Examine the contents of the `../data/dtmf_chip_hier.conf` file.

What files were used during design import?

Examine the contents of the `soce_hier_impl.tcl` script.

What major steps were used?

What files were used in Design Import for the lower-level blocks?

What files were used in Design Import for the top-level design?

What is the signoff slack (after routing) in each of the path groups setup by optDesign?

input to register slack?
register to register?
register to output?
inpu to output?
Section 6: Chip Assembly and Sign-Off

Chip Assembly and Sign-off Steps

At this stage the completed blocks and top-level design can be reassembled for flat final analysis

- Merge the partitioned design
- Sign-off timing analysis

If you skipped the previous step(s) you can start here but you should use the following .conf file in step 2: ../../data/dtmf_chip_unPTN_restore.conf and instead of using the data from the PTN/<block> directories look in the ../data directory for the block placement & routing in the ../data/block_results directory.

1) Return to the original working directory.
   ~/SOCE-4.1_WORKSHOP/work_hier

2) Start an encounter session and use ../data/dtmf_chip_unPTN.conf to populate the Design Import form.

3) Click OK.
   - The verilog files saved from IPO and CTS are specified in this configuration file. The … button can be used to view the full list and can be used to populate the form.

4) Load the previously saved floorplan file: ../data/dtmf_chip_part.fp using Design => Load => Floorplan...

MERGE THE PARTITIONED DESIGN

Use Partition=>Partition... to re-partition the design.

- Since the objective this time is to merge the partitioned design, pin optimization and timing budgets should NOT be created this time.

5) Load the top-level placement using Design => Load => Place and the file: ./PTN/dtmf_chip/dtmf_chip_done.enc.dat/dtmf_chip.place.gz.
6) Click **Open**.

7) Load the top-level routing using **Design => Load => Route** and the file: 
   `.PTN/dtmf_chip/dtmf_chip_done.enc.dat/dtmf_chip.route.gz`

8) Click **Open**.

9) Specify the block-level Floorplan, Placement, and Routing files using the 
   **Design=>Load=>Partition…** form.

11) Check the **Load Partition Routing** option.

**Merge the partitioned design.**

- **Placement files are saved in**
  
  PTN/blkName/blkName_done.enc.dat/blkName.place.gz

- **Route files are saved in**
  
  PTN/blkName/blkName_done.enc.dat/blkName.route.gz

14) Use **Partition=>Unpartition** to merge the partitioned blocks and top-level.

15) Look at the **Placement** view and check to see that all of the block-level routes and placement are now in the top level.

**RUN SIGN-OFF EXTRACTION AND TIMING**

16) Type **timeDesign – postRoute** at the encounter prompt.

   What is the worst slack at the top level? _______

**VERPLEX FORMAL VERIFICATION**

- Equivalency checking is a much faster and more exhaustive approach to functional design verification. Formal verification can be used to verify that the changes implemented in the back-end have not introduced functional bugs.

  During implementation, optimization and clock tree insertion have modified the initial netlist. In this step, you compare the original RTL used to synthesize this design, with the netlist generated by SOCE.

  **Run Verplex analysis.**
17) cd ../../work_verplex

18) At the Unix prompt, type: lec -ultra -do dtmf_chip.do

19) Examine the contents of the dtmf_chip.do file. This is the script that reads the golden RTL netlist and the revised netlist created by Encounter after implementation.

20) Examine the dtmf_chip_verplex..log file.

   Were there any un-equivalent points reported?______________

**EXPORT GDS FOR CHIP FINISHING**

Use *Design => Save GDS...* to export the chip level GDS.

Thank you for your participation in today’s SOC-Encounter workshop.

21) To refresh your workspace for the next participant, please do the following:

   1) cd $HOME
   2) source refresh
   3) Use the Logout icon at the bottom of the KDE window.
   4) Allow an AE to properly shutdown the laptop.
   5) Please complete the workshop evaluation.
Appendix A – Hierarchical Design

Benefits of Hierarchical Design

- During active design, not all blocks reach the same level of completeness concurrently. Hierarchical design flows enable black box what-if timing and area analysis during prototyping. During implementation, some blocks can reach a frozen/complete status earlier than others thus reducing the amount of the design in flux.

- Deriving block-level timing budgets from top-level system constraints dramatically reduces design time. Resulting block-level constraints account for physical requirements of the design and are used during block-level optimization.

- Parallel processing of multiple blocks can reduce cycle time during the implementation phase of the design.

- Late ECOs are easier to implement when they are confined to partitioned blocks rather than a full-chip design.

- Hardware capacity for top-level analysis and optimization may be extended by using block models for timing, SI, power, and clocks. Block models are derived during partitioning and are re-generated after block implementation to provide improved accuracy.

- IP reuse strategies can sometimes benefit from "hardening" block-level IP
Appendix B – Saved Starting Points

Section 1: Introduction to Encounter™

Section 2: Silicon Virtual Prototyping

Section 3: Partitioning

Section 4: Block Implementation

Section 5: Hierarchical Implementation

Section 6: Chip Assembly & Steps