Analyzing Timing

After you set the timing constraints such as clocks, input delays, and output delays, it is a good idea to use the check_timing command to check for timing setup problems and timing conditions such as incorrectly specified generated clocks and combinational feedback loops. The command checks the timing attributes of the current design and issues warning messages about any unusual conditions found.

The default report shows the startpoint, endpoint, path group (clock domain), path type (minimum delay, maximum delay, max_rise, min_fall, and so on), the incremental and cumulative time delay values along the data and clock paths, the data required time at the path endpoint, and the timing slack for the path.
Analyzing Timing

```
icc_shell> report_timing
...
Startpoint: I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_
      (falling edge-triggered flip-flop clocked by SDRAM_CLK)
Endpoint: sd_DQ_out[8]
      (output port clocked by SD_DDR_CLK)
Path Group: SD_DDR_CLK
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock SDRAM_CLK (fall edge)</td>
<td>3.75</td>
<td>3.75</td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>0.00</td>
<td>3.75</td>
</tr>
<tr>
<td>I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/CPN (sdcfq1)</td>
<td>0.00</td>
<td>3.75 f</td>
</tr>
<tr>
<td>I_SDRAM_TOP/I_SDRAM_IF/DQ_out_1_reg_8_/Q (sdcfq1)</td>
<td>0.36</td>
<td>4.11 r</td>
</tr>
<tr>
<td>I_SDRAM_TOP/I_SDRAM_IF/sd_mux_dq_out_8/z (mx02d4)</td>
<td>0.21</td>
<td>4.31 r</td>
</tr>
<tr>
<td>I_SDRAM_TOP/I_SDRAM_IF/sd_DQ_out[8] (SDRAM_IF)</td>
<td>0.00</td>
<td>4.31 r</td>
</tr>
<tr>
<td>I_SDRAM_TOP/sd_DQ_out[8] (SDRAM_TOP)</td>
<td>0.00</td>
<td>4.31 r</td>
</tr>
<tr>
<td>sd_DQ_out[8] (out)</td>
<td>0.00</td>
<td>4.31</td>
</tr>
<tr>
<td>data arrival time</td>
<td>4.31</td>
<td></td>
</tr>
</tbody>
</table>

| clock SD_DDR_CLK (rise edge) | 7.50 | 7.50 |
| clock network delay (ideal) | 1.00 | 8.50 |
| clock uncertainty | -0.05 | 8.45 |
| output external delay | -2.00 | 6.45 |
| data required time | 6.45 |

| data required time | 6.45 |
| data arrival time | -4.31 |

| slack (MET) | 2.14 |
...
```
Analyzing Timing

Using the GUI, you can generate color-coded diagrams showing the locations of worst-case timing conditions such as path slack, cell slack, net capacitance, clock latency/transition, and crosstalk.
Analyzing Power

The report_power command calculates and reports power for a design. The command uses the user-annotated switching activity to calculate the net switching power, cell internal power, and cell leakage power, and it displays the calculated values in a power report.

```
icc_shell> report_power
...
Library(s) Used:
  cb13fs120_ts_max (File: /remote/techp5/ref/db/sc_max.db)
  ram8x64_max (File: /remote/techp5/ref/db/ram8x64_max.db)
  ram16x128_max (File: /remote/techp5/ref/db/ram16x128_max.db)
Operating Conditions: cb13fs120_ts_max Library: cb13fs120_ts_max
Wire Load Model Mode: enclosed
Design        Wire Load Model        Library
--------------        ----------------
ORCA_TOP            ForQA            cb13fs120_ts_max
PCI_FIFO_1_DW01_sub_1 ForQA            cb13fs120_ts_max

Global Operating Voltage = 1.08
Power-specific unit information:
  Voltage Units = 1V
...
  Cell Internal Power = 34.8033 mW (93%)
  Net Switching Power = 2.7620 mW (7%)
  Total Dynamic Power = 37.5653 mW (100%)
  Cell Leakage Power = 424.5199 uW
...
```
You can generate a report on the quality of results (QoR) for the design in its current state by using the create_qor_snapshot command (or by choosing Timing > Create QoR Snapshot in the GUI). This command measures and reports the quality of the design in terms of timing, design rules, area, power, congestion, clock tree synthesis, routing, and so on.
Reporting Quality of Results

icc_shell> report_qor_snapshot

---------------------------------------
Report : create_qor_snapshot (my_snapshot1)
Design : system_controller
Version : D-2010.03
Date : Thu Feb 11 13:56:48 2010
Time unit : 1.0e-09 Second(ns)
Capacitance unit: 1.0e-12 Farad(pF)
Voltage unit : 1 Volt
Power unit : N/A
Location : /remote/PnR/implementation/snapshot
---------------------------------------
No. of scenario = 1
---------------------------------------
WNS of each timing group:
---------------------------------------
Setup WNS: 0.000
Setup TNS: 0.0
Number of setup violations: 0
Hold WNS: 0.000
Hold TNS: 0.000
Number of hold violations: 0
Number of max trans violations: 65
Number of max cap violations: 22
Route drc violations: 0
---------------------------------------
Area: 451495
Cell count: 23139
Buf/inv cell count: 3088
Std cell utilization: 18.324%
CPU/ELAPSE (hr): 0.11/23.88
Mem(MB): 1274
Host name: igcas007
---------------------------------------
Histogram:
---------------------------------------
Max violations: ...
Cell Density Map

To generate a cell density map, the tool divides the layout into a grid of squares measuring five standard-cell heights on each side. It finds the area utilization in each grid square and then color-codes each square according to the utilization value. It also displays a histogram showing the number of grid squares in each of the utilization value bins.
Cell Density Map
Cell Density Map
Congestion Maps

A congestion map can help you visualize the quality of placement with respect to the avoidance of routing congestion.
Hierarchy Visual Mode

The hierarchy visual mode is a display of the design with cells highlighted and color-coded according to block membership or levels of hierarchy.
Clock-Tree Visual Mode

The clock tree visual mode highlights the flylines and cells of a clock tree using a different color for each buffer level of the tree. For example, Figure shows the connections in levels 0, 1, and 2 of the sys_clk clock tree, with the level 0 in yellow, level 1 in magenta, and level 2 in blue. This display can help you visualize the extent and approximate route lengths of the clock tree.
Net Connections

The net connections in area visual mode lets you visualize the nets enclosed in or crossing a specified rectangular area of the design and examine the properties of those nets.
Net Connections

The net connections in area display capability lets you determine the net topology in a particular region and observe the properties of the nets in that area. This feature works with both routed and unrouted designs. You can analyze the nets that cross a specific area and use the information to change the floorplan or add constraints such as blockages. For example, excessive local nets might imply that utilization changes are needed, or excessive global or pass-through nets might imply that floorplan or placement constraint changes are needed.
Clock Tree Synthesis

Design Prerequisites

• The design is placed and optimized.

Use the `check_legality -verbose` command to verify that the placement is legal.

Running clock tree synthesis on a design that does not have a legal placement might result in long runtimes and reduced QoR.

The estimated QoR for the design should meet your requirements before you start clock tree synthesis. This includes acceptable results for

• Congestion

  If congestion issues are not resolved before clock tree synthesis, the addition of clock trees can increase congestion. If the design is congested, you can rerun `place_opt` with the `-congestion` and `-effort` high options, but the runtime can be long.

• Timing
• Maximum capacitance
• Maximum transition time

To ensure that the clock tree can be routed, verify that the placement is such that the clock sinks are not in narrow channels and that there are no large blockages between the clock root and its sinks. If these conditions occur, fix the placement before running clock tree synthesis.

• The power and ground nets are prerouted.
• High-fanout nets, such as scan enables, are synthesized with buffers.
Clock Tree Synthesis

Library Prerequisites

• Any cell in the logic library that you want to use as a clock tree reference (a buffer or inverter cell that can be used to build a clock tree) or for sizing of gates on the clock network must be usable by clock tree synthesis and optimization.

By default, clock tree synthesis and optimization cannot use buffers and inverters that have the dont_use attribute to build the clock tree.

• The physical library should include
  • All clock tree references (the buffer and inverter cells that can be used to build the clock trees)
  • Routing information, which includes layer information and non-default routing rules
  • TLUPlus models must exist.

Extraction requires these models to estimate the net resistance and capacitance.
Analyzing The Clock Tree

Before running clock tree synthesis, analyze each clock tree in your design to determine its characteristics and its relationship to other clock trees in the design. For each clock tree, determine

• What the clock root is
• What the desired clock sinks and clock tree exceptions are
• Whether the clock tree contains preexisting cells, such as clock-gating cells
• Whether the clock tree converges, either with itself (a convergent clock path) or with another clock tree (an overlapping clock path)
• Whether the clock tree has timing relationships with other clock trees in the design, such as interclock skew requirements
• What the logical design rule constraints (maximum fanout, maximum transition time, and maximum capacitance) are
• What the routing constraints (routing rules and metal layers) are

Use this information when you define the clock trees and to validate that the tool has the correct clock tree definitions.
Defining The Clock Tree

The tool uses the clock sources defined by the `create_clock` command as the clock roots and derives the default set of clock sinks by tracing through all cells in the transitive fanout of the clock roots. You can designate either an input port or internal hierarchical pin as a clock source.
Defining The Clock Root Attribute

If the clock root is an input port (without an I/O pad cell), you must accurately specify the driving cell of the input port. A weak driving cell does not affect logic synthesis, because logic synthesis uses ideal clocks. However, during clock tree synthesis, a weak driving cell can cause the tool to insert extra buffers as the tool tries to meet the clock tree design rule constraints, such as maximum transition time and maximum capacitance.

If you do not specify a driving cell (or drive strength), the tool assumes that the port has infinite drive strength.

If the clock root is an input port with an I/O pad cell, you must accurately specify the input transition time of the input port.
Implementing Clock Meshes

Clock meshes are homogeneous shorted grids of metal that are driven by many clock drivers. The purpose of a clock mesh is to reduce clock skew in both nominal designs and designs across variations such as on-chip variation, chip-to-chip variation, and local power fluctuations. A clock mesh reduces skew variation mainly by shorting the outputs of many clock drivers.

Figure shows the structure of a clock mesh. The network of drivers from the clock port to the mesh driver inputs is called the premesh tree. The network of shorted clock driver outputs is called the mesh.
Implementing Clock Meshes

Using clock meshes provides the following benefits:
• Small skew variation, especially for high-performance designs
• Consistent design performance across variations
• Predicable results throughout both the design stage and ECO stage later
• Stability resulting from mesh grids being close to receivers

Using clock meshes has the following disadvantages:
• More routing resources are required to create clock meshes
• Higher power consumption during transitions on the parallel drivers driving the mesh
H-Trees

An H-tree is a fractal structure built by drawing an H shape, then recursively drawing H shapes on each of the vertices, as shown in Figure. With enough recursions, the H-tree can distribute a clock from the center to within an arbitrarily short distance of every point on the chip while maintaining exactly equal wire lengths. Buffers are added as necessary to serve as repeaters. If the clock loads were uniformly distributed around the chip, the H-tree would have zero systematic skew. Moreover, the trees tend to use less wire and thus have lower capacitance than grids.
Spines

The spines drive length-matched serpentine wires to each small group of clocked elements. If the loads are uniform, the spine avoids the systematic skew of the grid by matching the length of the clock wires.
Figure (L) shows the global clock buffers distributing the clock to the three spines on the Pentium 4 with zero systematic skew while Figure (R) shows a photograph of the chip annotated with the clock spine locations.