SYNOPSYS®

CUSTOMER EDUCATION SERVICES

PrimeTime 1 Workshop

Lab Guide

10-I-034-SLG-005 2006.06

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

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

www.synopsys.com

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

"This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

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

Disclaimer

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

Registered Trademarks (®)

Synopsys, AMPS, Arcadia, C Level Design, C2HDL, C2V, C2VHDL, Cadabra, Calaveras Algorithm, CATS, CRITIC, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSIM, HSPICE, Hypermodel, iN-Phase, in-Sync, Leda, MAST, Meta, Meta-Software, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PowerMill, PrimeTime, RailMill, RapidScript, Saber, SiVL, SNUG, SolvNet, Superlog, System Compiler, TetraMAX, TimeMill, TMA, VCS, Vera, and Virtual Stepper are registered trademarks of Synopsys, Inc.

Trademarks (™)

Active Parasitics, AFGen, Apollo, Apollo II, Apollo-DPII, Apollo-GA, ApolloGAII, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanTestchip, AvanWaves, BCView, Behavioral Compiler, BOA, BRT, Cedar, ChipPlanner, Circuit Analysis, Columbia, Columbia-CE, Comet 3D, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, Cyclelink, Davinci, DC Expert, DC Professional, DC Ultra, DC Ultra Plus, Design Advisor, Design Analyzer, Design Vision, DesignerHDL, DesignTime, DFM-Workbench, Direct RTL, Direct Silicon Access, Discovery, DW8051, DWPCI, Dynamic-Macromodeling, Dynamic Model Switcher, ECL Compiler, ECO Compiler, EDAnavigator, Encore, Encore PQ, Evaccess, ExpressModel, Floorplan Manager, Formal Model Checker, FoundryModel, FPGA Compiler II, FPGA Express, Frame Compiler, Galaxy, Gatran, HANEX, HDL Advisor, HDL Compiler, Hercules, Hercules-Explorer, Hercules-II, Hierarchical Optimization Technology, High Performance Option, HotPlace, HSIMplus, HSPICE-Link, iN-Tandem, Integrator, Interactive Waveform Viewer, i-Virtual Stepper, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, JVXtreme, Liberty, Libra-Passport, Library Compiler, Libra-Visa, Magellan, Mars, Mars-Rail, Mars-Xtalk, Medici, Metacapture, Metacircuit, Metamanager, Metamixsim, Milkyway, ModelSource, Module Compiler, MS-3200, MS-3400, Nova Product Family, Nova-ExploreRTL, Nova-Trans, Nova-VeriLint, Nova-VHDLlint, Optimum Silicon, Orion_ec, Parasitic View, Passport, Planet, Planet-PL, Planet-RTL, Polaris, Polaris-CBS, Polaris-MT, Power Compiler, PowerCODE, PowerGate, ProFPGA, ProGen, Prospector, Protocol Compiler, PSMGen, Raphael, Raphael-NES, RoadRunner, RTL Analyzer, Saturn, ScanBand, Schematic Compiler, Scirocco, Scirocco-i, Shadow Debugger, Silicon Blueprint, Silicon Early Access, SinglePass-SoC, Smart Extraction, SmartLicense, SmartModel Library, Softwire, Source-Level Design, Star, Star-DC, Star-MS, Star-MTB, Star-Power, Star-Rail, Star-RC, Star-RCXT, Star-Sim, Star-SimXT, Star-Time, Star-XP, SWIFT, Taurus, TimeSlice, TimeTracker, Timing Annotator, TopoPlace, TopoRoute, Trace-On-Demand, True-Hspice, TSUPREM-4, TymeWare, VCS Express, VCSi, Venus, Verification Portal, VFormal, VHDL Compiler, VHDL System Simulator, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

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

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

Document Order Number: 10-I-034-SLG-005
PrimeTime 1 Workshop Lab Guide

1 Does Your Design Meet Timing?

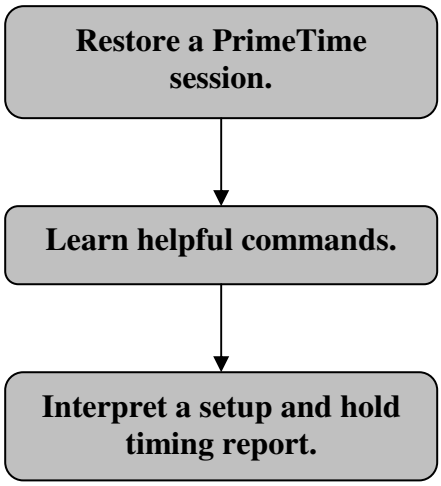
Learning Objectives

After completing this lab, you should be able to:

- Restore a previously saved PrimeTime session
- Take advantage of helpful PrimeTime commands that will make you more efficient when using PrimeTime interactively and show you how to find more information on commands, variables and your design library
- Interpret key components of a timing report for setup and hold timing checks

Lab Duration: 30 minutes

Overview



Answers / Solutions

This lab guide contains answers and solutions to all questions. If you need some help with answering a question or would like to confirm your results, check the back portion of this lab.

Instructions

Your goal is to start mastering timing reports as well as learn useful PrimeTime commands.

Task 1. Restore a PrimeTime Sesssion

Learn to invoke a previously saved PrimeTime session to perform STA. The first 6 labs of this workshop will be performed in the same directory.

1. Invoke PrimeTime from the **lab1-6** workshop lab Unix directory.

```
unix% cd lab1-6
unix% pt_shell
```

2. Restore a previously saved PrimeTime session. This step will read in the design netlist, libraries, and constraints. The design is now ready for analysis.

Note: The argument **orca_savesession** below is a Unix directory. You will learn more about saving a PrimeTime session later in the workshop.

Note: PrimeTime supports command, option, variable and file completion since v2004.12. Type a few letters and then hit the tab key.

"TAB" for the file completion

```
pt_shell> restore_session orca_savesession
```

3. Execute **report_analysis_coverage**.

```
pt_shell> report_analysis_coverage
```

Question 1. From the header at the top of this report, what is the name of the design under analysis?

.....

Question 2. How many timing check violations does **ORCA** have?

.....

Task 2. Explore Helpful PrimeTime Commands

1. Execute the following three history short cut commands:

```
pt_shell> history
pt_shell> !!
pt_shell> !2
```

Question 3. Describe the difference between the last two history commands above.

.....

2. Even more powerful are the new command line editing capabilities.

Use up and down arrows to scroll through the history event list as an alternative to the previous step.

Type the following to see all the available key bindings (in the default emacs editing mode).

```
pt_shell> list_key_bindings
```

3. Explore page mode; execute the following command, which will generate many reports that scroll off the screen:

```
pt_shell> report_timing
```

4. Turn on page mode.

```
pt_shell> page_on
pt_shell> !rep
```

alias page_on {set sh_en_page_mode true}

Note: Use the **space bar** to page through a long report. Quit from a long report in page mode by typing "**q**". If you want to turn off page mode, use the command **page_off**.

5. Find the command to restore a PrimeTime session and then display help information on this command.

```
pt_shell> help restore*
pt_shell> man restore_session
pt_shell> restore_session -help
```

Note: The following is an alternative way to display syntax help.

```
pt_shell> help -v restore_session
```

Question 4. From the last command above, does the command **restore_session** accept switches?

.....

6. The time unit in PrimeTime is determined by the main technology library.

To find the time unit for **ORCA**, first list all libraries in memory.

Note: The * in the following report indicates the main library.

```
pt_shell> list_lib
```

7. Generate a report for the main library which will state the time unit.

Note: Use copy and paste to avoid mistyping the lib name. The time unit is at the very top of the report.

```
pt_shell> report_lib cb13fs120_tsmc_max
```

Question 5. What is the time unit used for timing reports (as well as all other reports) for the **ORCA** design?

.....

Do not forget to use "q" to quit from a long report in page mode and return to the pt_shell prompt without reading the entire report!

8. Display a more focused library report, displaying only units of the current design.

```
pt_shell> report_units
```

Task 3. Analyze STA Reports

Generate and intepret two STA reports for setup and hold for **SYS_CLK**.

1. By default, the command **report_timing** generates a single report for each clock in a design.

Execute the following to display the clocks in **ORCA**:

```
pt_shell> report_clock
```

Question 6. How many clocks are in **ORCA**?

.....

2. Create a single, "short" timing report for setup for the clock **SYS_CLK**. Use command-line expansion (the tab key) to expand both the command AND the options `-group` and `-path`.

```
pt_shell> report_timing -group SYS_CLK -path short
```

Note: The lines containing the data path cells and their delays are removed from the data arrival section making this report "short".

Note: The above command generates a report for setup by default.

Question 7. There are at least 4 clues that this report is for setup and not for hold. How many can you identify?

.....

.....

Question 8. Identify the instance names of the start and end point flip-flops.

.....

Question 9. The clock skew for this timing path is 0.001ns; which two lines in the report can you use to calculate this?

.....

Question 10. How does this clock skew affect slack (i.e. does the clock skew help or hurt slack)?

.....

Question 11. How large is the violation in comparison to the clock period?

.....

3. Generate a timing report for hold time.

The following is a short cut that will execute the last command in history starting with the letters "**rep**" and add the switch **–delay min** (which will generate a report for hold time).

```
pt_shell> !rep -delay min
```

Question 12. There are at least 4 clues that indicate this is a hold report and not a setup report. How many can you find?

.....

Question 13. How does the clock skew in this hold report affect slack (i.e. does the clock skew help or hurt slack)?

.....

4. Quit PrimeTime.

```
pt_shell> quit
```

This completes lab 1. Return to lecture.

Answers / Solutions

Question 1. From the header at the top of this report, what is the name of the design under analysis?

The design is **ORCA**.

Question 2. How many timing check violations does **ORCA** have?

There are 88 violations, all of which are for setup timing checks.

Question 3. Describe the difference between the last two history commands above.

The command **!2** repeats the 2nd command executed in history. The command **!!** repeats the last executed command in history.

Question 4. From the last command above, does the command **restore_session** accept switches?

No, there are no switches for this command. It has one required argument, a Unix directory that contains the saved session.

Question 5. What is the time unit used for timing reports (as well as all other reports) for the **ORCA** design?

1ns.

Question 6. How many clocks are in **ORCA**?

There are 6 clocks in **ORCA**.

Question 7. There are at least 4 clues that this report is for setup and not for hold. How many can you identify?

The most glaring clue is the highlighted "library setup time" in the report below. The more subtle clues are:

- The clock edges used for the data arrival and data required are 0ns and 8ns respectively (and not 0ns and 0ns as for hold time).
- The "**Path Type**" in the header is **max** which indicates that this report is for setup (a "**Path Type**" of **min** indicates a hold report).
- Finally, the data arrival time is after the data required time and the slack is violated (whereas if this was a report for hold, the slack would be met!)

Question 8. Identify the instance names of the start and end point flip-flops.

```
Startpoint: I_ORCA_TOP/I_BLENDER/s3_op2_reg[18]
              (rising edge-triggered flip-flop clocked by SYS_CLK)
Endpoint: I_ORCA_TOP/I_BLENDER/s4_op2_reg[31]
              (rising edge-triggered flip-flop clocked by SYS_CLK)
Path Group: SYS_CLK
Path Type: max
Max Data Paths Derating Factor  : 1.100
```

Question 9. The clock skew for this timing path is 0.001ns. Which lines in the report can you use to calculate this?

```
Point                                                  Incr       Path
-----------------------------------------------------------------------
clock SYS_CLK (rise edge)                              0.000      0.000
clock network delay (propagated)                       2.832 *    2.832
I_ORCA_TOP/I_BLENDER/s3_op2_reg[18]/CP (sdnrb1)        0.000      2.832 r
I_ORCA_TOP/I_BLENDER/s3_op2_reg[18]/Q (sdnrb1)         0.408 *    3.240 r
  ...
I_ORCA_TOP/I_BLENDER/s4_op2_reg[31]/D (sdnrb1)         8.142 *   11.382 f
data arrival time                                                11.382

clock SYS_CLK (rise edge)                              8.000      8.000
clock network delay (propagated)                       2.831 *   10.831
I_ORCA_TOP/I_BLENDER/s4_op2_reg[31]/CP (sdnrb1)                  10.831 r
library setup time                                    -0.169 *   10.663
data required time                                               10.663
```

Question 10. How does this clock skew affect slack (i.e. does the clock skew help or hurt slack)?

The clock skew hurts the slack for setup in this specific timing report. The clock latency to the start point flip-flop causes the data to arrive 0.001ns later and thus the positive slack (or margin) is smaller.

Question 11. How large is the violation in comparison to the clock period?

The clock period is 8ns. The violation is 0.719. It is approximately 9% of the clock period.

Question 12. There are at least 4 clues that indicate this is a hold report and not a setup report. How many can you find?

The most glaring clue is the highlighted "library hold time" in the report below. The more subtle clues are:

- The clock edges used for the data arrival and data required are 0ns and 0ns respectively.
- The "**Path Type**" in the header is **min** which indicates that this report is for hold time.
- Finally, the data arrival time is after the data required time and the slack is met.

Question 13. How does the clock skew for this hold report affect slack (i.e. does the clock skew help or hurt slack)?

The clock skew is 0.002ns and it results in a larger positive slack (i.e. it helps slack). For hold time, the data arrival must arrive after the data required for a positive slack.

2

PrimeTime Objects

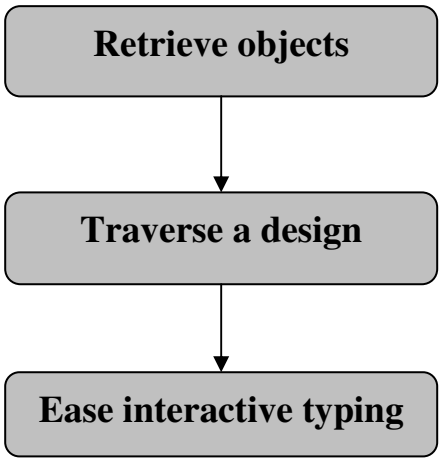
Learning Objectives

After completing this lab, you should be able to:

- Display objects using the '`all`…' and '`get`…' commands
- Traverse a design using the `-of_objects` option.
- Ease interactive typing by defining and dereferencing a variable.

Lab Duration: 15 minutes

Overview



Answers & Solutions

This lab guide contains answers and solutions to all questions. If you need some help with answering a question or would like to confirm your results, check the back portion of this lab.

Instructions

Your goal is to gain familiarity with PrimeTime objects, with accessing those objects, and with seeing how objects are connected. In addition, you will create and dereference a variable.

Task 1. Set up PrimeTime for Lab 2

1. Invoke PrimeTime from the **lab1-6** workshop lab Unix directory and restore a saved session.

```
unix% cd lab1-6
unix% pt_shell
pt_shell> restore_session orca_savesession
```

Task 2. Traverse a Design from Port to Clock Generator

This task shows '**scripting techniques**' to find out what is connected to a cell, or to a port, or to a net; that is, to traverse a design.

The PrimeTime GUI and the report… commands, covered later in this course, are alternative powerful **interactive** ways to traverse a design.

1. Display all the input ports.

```
pt_shell> all_inputs
```

2. Narrow the display to ports containing the string 'clk'

```
pt_shell> get_ports *clk*
```

3. Display nets connected to the port 'pclk'.

```
get_nets -of_objects [get_ports pclk]
```

4. Display cells connected to the net ‘pclk’.

```
get_cells –of_objects [get_nets pclk]
```

5. Look further downstream; display nets connected to the cell ‘pclk_iopad’

```
get_nets –of_objects [get_cells pclk_iopad]
```

6. Look still further downstream; display cells connected to the net ‘net_pclk’

```
get_cells –of_objects [get_nets net_pclk]
```

7. Display pins of I_CLOCK_GEN.

```
get_pins –of_objects [get_cells I_CLOCK_GEN]
```

8. Set up an interactive shortcut by assigning a long name to a variable. In this case, cut and paste the longest pin name and assign it to the variable ‘a’.

```
set a I_CLOCK_GEN/buf_sdram_clk_G5B2I12ASTHIRNet791
```

9. Use the variable you created.

```
get_cell –of $a
```

Question 1. If you know the name of a net, what objects can you access using the `–of_objects` option?

.....

Question 2. To get additional detailed information about any of the objects you retrieve with the ‘`get…`’ commands, what ready-made commands would you try?

.....

Task 3. Optional – A little more power --

These tasks are not necessary to succeed with any of the next modules in the course. However, they give you more power to explore PrimeTime objects and how they are connected in your design.

1. Look at the values of attributes on the clock object 'PCI_CLK'.

```
get_attribute [get_clock PCI_CLK] period
report_attribute -application [get_clocks PCI_CLK]
```

Question 3. Which command would you use interactively?

.....

Question 4. Which command can be nested or embedded within another command, making it useful in a script?

.....

2. Display the names and class of objects connected to a net.

```
set my_net [get_nets pclk]
query_objects -verbose [all_connected $my_net]
```

Question 5. What classes of objects are connected to the pclk net?

.....

This completes the lab. Return to lecture.

Answers / Solutions

Question 1. If you know the name of a net, what objects can you access using the `-of_objects` option?

cells, pins, and ports:

```
set my_net [get_nets pclk]
get_ports -of_objects $my_net
get_pins -of_objects $my_net
get_cells -of_objects $my_net
```

Question 2. To get additional detailed information about any of the objects you retrieve with the 'get…' commands, what ready-made commands would you try?

`report…` commands. Do a '`help report*`" to display a list of commands that present information for interactive use.

Question 3. Which command would you use interatively?

`report_attribute` is an interactive display command; `get_attribute` can be used interactively or in scripting.

Question 4. Which command can be nested or embedded within another command, making it useful in a script?

`get_attribute` – it returns a value.

Question 5. What classes of objects are connected to the pclk net?

ports and pins.

3

Constraints in a Timing Report

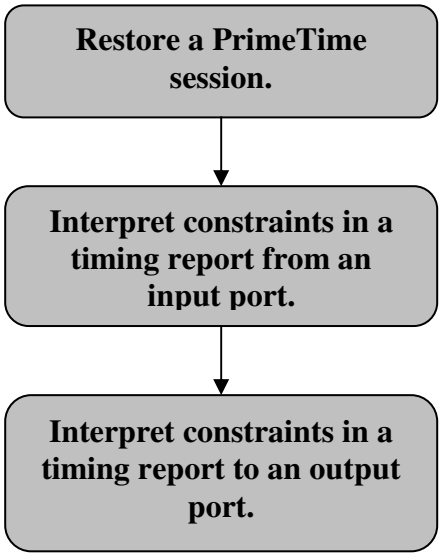
Learning Objectives

After completing this lab, you should be able to:

- Identify and interpret constraints in a timing report
  - Clock constraints
  - Interface constraints

Lab Duration: 30 minutes

Overview



Answers & Solutions

This lab guide contains answers and solutions to all questions. If you need some help with answering a question or would like to confirm your results, check the back portion of this lab.

Instructions

Task 1. Setup PrimeTime For Lab 3

1. Invoke PrimeTime from the **lab1-6** workshop lab Unix directory and restore the PrimeTime session using the **orca_savesession** directory.

Use the job aid labeled "timing reports" if you have forgotten the command.

2. Generate a summary report for all violations in **ORCA**.

Use the job aid labeled " timing reports" if you have forgotten the command.

Question 1. How many output delay constraints are there for setup and for hold and are these constraints met or violated?

.....

Task 2. Analyze a Timing Report From an Input Port

1. Generate a report for the input delay constraints applied to the port **pad[0]**.

```
pt_shell> report_port -input_delay pad[0]
```

Question 2. What are the min and max arrival times to **pad[0]**?

.....

Question 3. What is the name of the external start point clock constraining **pad[0]**?

.....

2. Generate a timing report for setup starting at the port **pad[0]**.

Answer the following questions using this report.

Use your job aid labeled "timing reports" for help recalling the appropriate switch for report_timing.

Question 4. Which lines in the timing report did you use to ensure the reported path starts at the port **pad[0]** and is for setup?

.....

Question 5. List all user specified constraints in this timing report.

.....

Question 6. Where must the clock latency be included for the start point clock **PCI_CLK**?

.....

Question 7. Describe the direction of the port **pad[0]** (i.e. is it an input, output or inout port).

.....

Question 8. Describe the end point of this timing path (i.e. is it an output port or an internal flip-flop).

.....

3. Generate a new report from the same port **pad[0]** for setup, which also shows the details of the calculated clock network delay.

Use the job aid labeled "timing reports" for help recalling the appropriate switch for report_timing. Remember to take advantage of history commands.

Question 9. How large is the clock source latency versus the clock network latency for the end point clock **PCI_CLK**?

.....

Question 10. Where has the clock **PCI_CLK** been defined (the clock definition point)?

.....

4. Generate a report starting at the port **pad[0]** for hold time.

Question 11. Does the value of the input external delay constraint match your expectations?

.....

Task 3. Analyze a Timing Report to an Output Port

1. Generate a report for the output delay constraints applied to the port **pad[0]**.

Use help in PrimeTime to determine the correct command and switch to do this – using the previous task as an example.

Question 12. What are the min and max output delay constraints for this port?

.....

Question 13. How will the negative min output delay constraint be applied to this port (i.e. will it impose a positive or negative hold requirement)?

.....

Question 14. What is the name of the external end point clock constraining this port?

.....

2. Generate a "short" timing report ending at the port **pad[0]** for hold time.

Question 15. Describe the start point of this timing path (i.e. is it an input port or an internal flip-flop).

.....

Question 16. Does the path group for this timing path match your expectations?

.....

Question 17. Does the "data required time" match your expectations?

.....

3. Optionally, apply the following constraint which will impose a positive output delay constraint for hold on **pad[0]** and then re-execute the steps in this task to see the affect.

```
pt_shell> set_output_delay -min 1.0 -clock PCI_CLK pad[0]
```

4. **Quit** PrimeTime. This completes the lab. Please return to lecture.

Answers / Solutions

```
pt_shell> restore_session orca_savesession
pt_shell> report_analysis_coverage
Type of Check      Total           Met         Violated         Untested
-------------------------------------------------------------------------
setup               9629      3500 ( 36%)      88 (  1%)      6041 ( 63%)
hold                9629      3588 ( 37%)       0 (  0%)      6041 ( 63%)
recovery            1316      1210 ( 92%)       0 (  0%)       106 (  8%)
removal             1316      1210 ( 92%)       0 (  0%)       106 (  8%)
min_period            20        20 (100%)       0 (  0%)         0 (  0%)
min_pulse_width     7273      5957 ( 82%)       0 (  0%)      1316 ( 18%)
out_setup             75        75 (100%)       0 (  0%)         0 (  0%)
out_hold              75        75 (100%)       0 (  0%)         0 (  0%)
-------------------------------------------------------------------------
All Checks         29333     15635 ( 53%)      88 (  0%)     13610 ( 46%)
```

Question 1. How many output delay constraints are there for setup and for hold and are these constraints met or violated?

From **report_analysis_coverage**, there are 75 output delay constraints for both setup and hold and they are all met. Remember to verify that all output ports are constrained for both setup as well as for hold.

Question 2. What are the min and max arrival times to **pad[0]**?

The min and max arrival times are 2ns and 8ns respectively (with the same constraint for both rise and fall data transitions at the port pad[0]).

Question 3. What is the name of the external start point clock constraining **pad[0]**?

The name of the clock is **PCI_CLK**.

Question 4. Which lines in the timing report did you use to ensure the reported path starts at the port **pad[0]** and is for setup?

```
pt_shell> report_timing -from pad[0]
  Startpoint: pad[0] (input port clocked by PCI_CLK)
  Endpoint: I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
```

Question 5. List all user specified constraints involved in this timing report.

The clock period is a constraint. The clock **PCI_CLK** is propagated (not ideal). The input external delay (which comes from an input delay constraint).

Question 6. Where must the clock latency be included for the start point clock **PCI_CLK**?

The clock network delay is zero. Therefore, the only other place to represent the external clock latency is as a part of the input delay constraint (i.e. the input external delay). The appropriate way to model this is to use the switches **–network_latency_included** and **–source_latency_included** for **set_input_delay.**

Question 7. Describe the direction of the port **pad[0]** (i.e. is it an input, output or inout port).

The port **pad[0]** is an inout port; therefore, it is both a timing path start point as well as a timing path end point!

```
Point                                                  Incr       Path
-----------------------------------------------------------------------
clock PCI_CLK (rise edge)                             0.000      0.000
clock network delay (propagated)                      0.000      0.000
input external delay                                  8.000      8.000 r
pad[0] (inout)                                        0.000      8.000 r
```

Question 8. Describe the end point of this timing path (i.e. is it an output port or an internal flip-flop).

The end point is a rising-edge triggered flip-flop clocked by **PCI_CLK** (it is actually a timing model that looks like a flip-flop with setup and hold timing checks).

Question 9. How large is the clock source latency versus the clock network latency for the end point clock **PCI_CLK**?

Shown below is only the data required time section of the timing report. The source latency is 0ns. The clock network latency is 0.979ns.

```
pt_shell> !rep -path full_clock

. . .

clock PCI_CLK (rise edge)                                15.000     15.000
clock source latency                                      0.000     15.000
pclk (in)                                                 0.000     15.000 r
pclk_iopad/CIN (pc3d01)                                   1.087 *   16.087 r
I_CLOCK_GEN/I_PLL_PCI/CLK (PLL)                          -1.445 *   14.642 r
I_CLOCK_GEN/bufbdfG1B1I1_1/Z (bufbdf)                     0.236 *   14.878 r
I_CLOCK_GEN/U21/Z (mx02d2)                                0.216 *   15.094 r
I_CLOCK_GEN/bufbdfG2B1I1_2/Z (bufbdf)                     0.174 *   15.268 r
I_CLOCK_GEN/U17/ZN (invbdk)                               0.044 *   15.311 f
I_CLOCK_GEN/U14/ZN (invbdk)                               0.023 *   15.334 r
I_CLK_SOURCE_PCLK/Z (bufbdk)                              0.217 *   15.551 r
invbd7G5B1I2/ZN (invbd7)                                  0.138 *   15.689 f
I_ORCA_TOP/invbdkG5B2I4_1/ZN (invbdk)                     0.080 *   15.769 r
I_ORCA_TOP/buffd7G5B3I24/Z (buffd7)                       0.195 *   15.965 r
I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM/CE1 (ram32x32)   0.014 *   15.979 r
library setup time                                       -0.457 *   15.522
data required time                                                  15.522
```

Question 10. Where has the clock **PCI_CLK** been defined (the clock definition point)?

The clock **PCI_CLK** is defined at the input port **pclk**. The clock definition point separates the clock source latency from the clock network latency.

Question 11. Does the value of the input external delay constraint match your expectations?

Yes. From **report_port** above, the input external delay should be 2ns with respect to the rising edge of **PCI_CLK**. This is confirmed in the timing report below.

```
pt_shell> report_timing -delay min -from pad[0]

Startpoint: pad[0] (input port clocked by PCI_CLK)
Endpoint: I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM
              (rising edge-triggered flip-flop clocked by PCI_CLK)
Path Group: PCI_CLK
Path Type: min

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock PCI_CLK (rise edge)                               0.000      0.000
clock network delay (propagated)                        0.000      0.000
input external delay                                    2.000      2.000 r
pad[0] (inout)                                          0.000      2.000 r
pad_iopad_0/PAD (pc3b03)                                0.100 *    2.100 r
pad_iopad_0/CIN (pc3b03)                                0.719 *    2.819 r
I_ORCA_TOP/pad_in[0] (ORCA_TOP)                         0.000 *    2.819 r
```

Question 12. What are the min/max output delay constraints for this port?

```
pt_shell> report_port -help

Usage:
 report_port          # Report port info
   [-verbose]             (Show all port info)
   [-design_rule]         (Only port design rule info)
   [-drive]               (Only port drive info)
   [-input_delay]         (Only port input delay info)
   [-output_delay]        (Only port output delay info)
   [-wire_load]           (Only port wire load info)
   [-nosplit]             (Don't split lines if column overflows)
   [port_list]            (List of ports)

pt_shell> report_port -output_delay pad[0]

                  Output Delay
                 Min             Max       Related Related
 Output Port    Rise   Fall   Rise    Fall   Clock   Pin
-------------------------------------------------------------
pad[0]       -1.00   -1.00    4.00    4.00  PCI_CLK   --
```

Question 13. How will the negative min output delay constraint be applied to this port (i.e. will it impose a positive or negative hold requirement)?

In lecture, it was stated that a negative hold output delay constraint will impose a positive hold requirement.

Question 14. What is the name of the external end point clock constraining this port?

The port **pad[0]** is constrained with respect to **PCI_CLK**.

Question 15. Describe the start point of this timing path.

The start point of the timing path is an internal flip-flop.

```
pt_shell> report_timing -delay min -to pad[0]

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[0] (output port clocked by PCI_CLK)
```

Question 16. Does the path group for this timing path match your expectations?

Yes. The path group is **PCI_CLK** which is the same as the external capture clock name.

Question 17. Does the "data required time" match your expectations?

Yes. The capture clock edge is zero, which is appropriate for hold. The hold requirement of 1ns is positive, and the propagated clock network delay is 0ns. The data required section of the timing report is shown below.

```
  clock PCI_CLK (rise edge)                              0.000      0.000
  clock network delay (propagated)                       0.000      0.000
  output external delay                                  1.000      1.000
  data required time                                                1.000
```

Answers for optional step

```
pt_shell> report_port -output_delay pad[0]

                  Output Delay
                 Min             Max       Related Related
 Output Port    Rise   Fall   Rise    Fall   Clock   Pin
-------------------------------------------------------------
pad[0]        1.00    1.00    4.00    4.00  PCI_CLK   --

pt_shell> report_timing -to pad[0] -delay min -path short

. . .
Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock PCI_CLK (rise edge)                               0.000      0.000
clock network delay (propagated)                        1.065 *    1.065
I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)            0.000      1.065 r
I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)             0.364 *    1.430 f
. . .
pad[0] (inout)                                          0.100 *    3.778 f
data arrival time                                                  3.778

clock PCI_CLK (rise edge)                               0.000      0.000
clock network delay (propagated)                        0.000      0.000
output external delay                                  -1.000     -1.000
data required time                                                -1.000
-----------------------------------------------------------------------------
data required time                                                -1.000
data arrival time                                                 -3.778
-----------------------------------------------------------------------------
slack (MET)                                                        4.778
```

4

Timing Arcs in a Timing Report

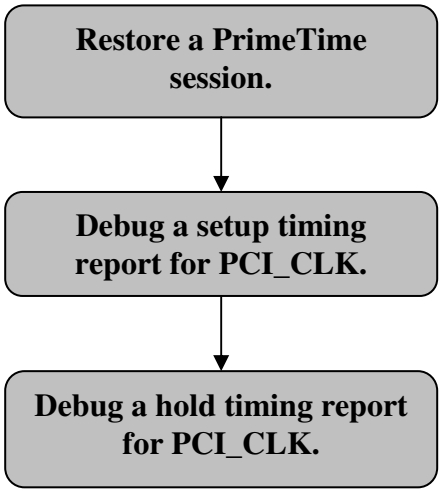
Learning Objectives

After completing this lab, you should be able to:

- Interpret timing arcs and transitions in a timing report
- Generate timing reports for a specific data transition at the end point
- Generate timing reports with input pins included
- Generate a report on library timing arcs

Lab Duration: 40 minutes

Overview



Answers & Solutions

This lab guide contains answers and solutions to all questions. If you need some help with answering a question or would like to confirm your results, check the back portion of this lab.

Instructions

Task 1. Setup PrimeTime For Lab 4

1. Invoke PrimeTime from the **lab1-6** workshop lab Unix directory and restore the PrimeTime session using the **orca_savesession** directory.

2. Execute the following command to generate a timing report for **PCI_CLK**:

```
pt_shell> report_timing -group PCI_CLK
```

Question 1. Does this timing path meet or violate timing?

.....

Task 2. Analyze a Timing Report for Setup

1. Answer the following questions regarding the report generated in the last task.

Question 2. What type of timing path is this - internal flip-flop to flip-flop, input, or output timing path?

.....

Question 3. List the user specified constraints in this timing report.

.....

Question 4. The last two cell delays reflect a falling transition (through the buffer **bufbda**) followed by a rising transition (through the pad cell **pc3b03**). Offer two possible reasons why this sequence of transitions (fall followed by a rise) could occur?

.....

In the next few steps, you will continue to explore and confirm your answer for the above question.

Question 5. What additional information do you need to confirm the actual reason for the sequence of transitions (fall followed by a rise) described in the last question?

.....

2. Generate the same timing report again, but this time include the input pins.

Question 6. Name the input pin of the I/O pad through which this path traverses.

.....

3. Execute the following to identify the technology library name which contains this I/O pad:

```
pt_shell> report_cell pad_iopad_5
```

Question 7. Which technology library contains this I/O pad?

.....

4. Use the appropriate command to generate a library report on the timing arcs for this I/O pad.

Use the job aid labeled "timing reports" or use your help resources in PrimeTime to find the appropriate command, switches and arguments.

Question 8. List the two timing arcs from the **OEN** pin to the **PAD** pin of this I/O pad.

.....

Task 3. Analyze a Timing Report for Hold

1. Generate a timing report for hold time for the same clock group **PCI_CLK**.

```
pt_shell> report_timing -group PCI_CLK -delay min
```

Question 9. What type of timing path is this - internal flip-flop to flip-flop, input, or output timing path?

.....

Question 10. How many cells are on the data path of this timing path?

.....

Question 11. The cell delay used for the clock pin (**CP**) to **Q** pin of the start point flip-flop is for a rise transition. Offer one possible reason why this results in a worse slack for hold than using the faster fall delay through this flip-flop?

.....

In the next step, you will continue to explore and confirm your answer for the above question.

Question 12. What additional information do you need to confirm your answer for the above question?

.....

2. Generate another timing report for the same timing path for hold time but with a fall transition at the end point (instead of a rise transition).

Use copy and paste to avoid mistyping the end point and start point pin names.

Use the job aid labeled "timing reports" to find the appropriate switches for report_timing.

Question 13. Which lines in this report did you use to confirm that the correct path has been reported?

.....

Question 14. Was the guess correct – the faster fall delays results in a faster data arrival time but a smaller hold time requirement and thus a better slack?

.....

3. Quit PrimeTime.

This completes the lab. Return to lecture.

Answers / Solutions

Question 1. Does this timing path meet or violate timing?

It meets timing with a slack of 3.042ns.

Question 2. What type of timing path is this - internal flip-flop to flip-flop, input, or output timing path?

This is an output timing path ending at the output port named **pad[5]**.

Question 3. List the user specified constraints in this timing report.

The clock period for **PCI_CLK** (15ns). The clock is propagated. The output external delay (which comes from an output delay constraint of 4ns).

Question 4. The last two cell delays reflect a falling transition (through the buffer **bufbda**) followed by a rising transition (through the pad cell **pc3b03**). Offer two possible reasons why this sequence of transitions (fall followed by a rise) could occur?

One reason is the timing arc through the pad cell is negative unate. In this case, the only two possibilities would be a fall followed by a rise transition or a rise followed by a fall transition. The other reason (which turns out to be the actual reason) is the timing arc through the pad cell is non unate. In that case, any of the 4 possible combinations could occur.

Question 5. What additional information do you need to confirm the actual reason for the sequence of transitions (fall followed by a rise) described in the above question?

First, display the input pin of the pad cell through which this timing path is traversing. Then, generate a report of the library timing arcs. This would confirm the type of timing arc between these pins. You will explore this in the next few lab steps.

Question 6. Name the input pin of the I/O pad through which this path traverses.

Use the command below to generate the appropriate report. The input pin is **OEN**.

```
pt_shell> !rep -input_pins
```

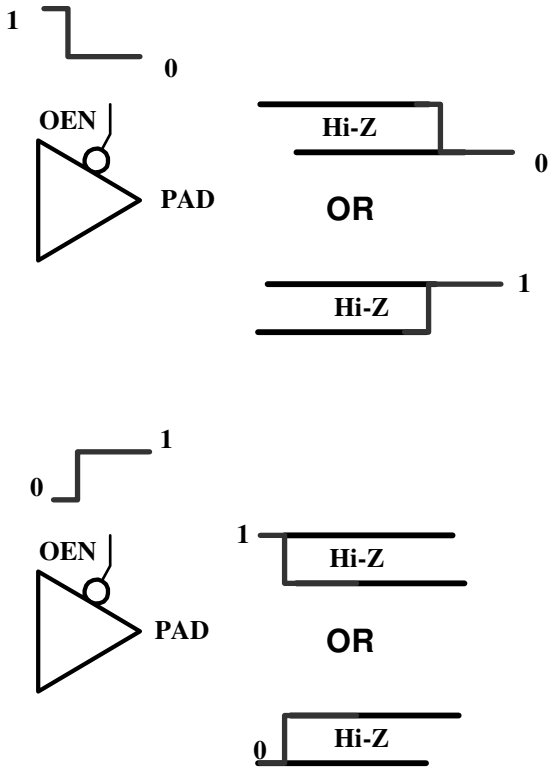
Question 7. Which technology library contains this I/O pad?

```
pt_shell> report_cell pad_iopad_5
Cell                  Reference       Library             Area  Attributes
--------------------------------------------------------------------------
pad_iopad_5           pc3b03       cb13io320_tsmc_max    3426.185
--------------------------------------------------------------------------
Total 1 cells                                            3426.185
```

Question 8. List the two timing arcs from the **OEN** pin to the **PAD** pin of this I/O pad.

```
pt_shell> report_lib -timing_arcs  cb13io320_tsmc_max pc3b03
                            Arc                    Arc Pins
   Lib Cell  Attributes     #  Type/Sense       From        To          When
   ------------------------------------------------------------------------
   pc3b03                   0  disable_high     OEN         PAD
                            1  enable_low       OEN         PAD
                            2  positive_unate   I           PAD
                            3  positive_unate   PAD         CIN
```

This timing arc is non unate, based on an understanding of tri-state timing arcs. When the pad is enabled or disabled – this could result in either a 1 or 0 delay at the output pin (a transition to a Z-state must be propagated as a possible transition to a logic 1 or 0 to find all possible cases).



Question 9. What type of timing path is this - internal flip-flop to flip-flop, input, or output timing path?

This is an internal timing path. The end point looks like a flip-flop, but is one of many timing arcs in a RAM timing model.

Question 10. How many cells are on the data path of this timing path?

There is only one cell on this data path, the start point flip-flop. The timing path consists of a start point flip-flop tied directly to the end point flip-flop.

Question 11. The cell delay used for the clock pin (**CP**) to **Q** pin of the start point flip-flop is a rise delay. Offer one reason why this would result in a worse slack for hold than using a fall delay through this flip-flop?

Typically, fall delays are faster than rise delays and would offer a worse slack for hold! The one exception is if the fall delay at the data pin of the end point flip-flop resulted in a larger library hold time. This is what occurs in this case.

Question 12. What additional information do you need to confirm your answer for the above question?

Generate another timing report where the data arrival time is calculated with fall transition at the end point and compare the two reports. In this way you can confirm that the library hold time is in fact smaller with a falling transition at the data pin of the end point flip-flop and thus the resulting slack better. You will explore this in the next lab step.

Question 13. Which lines in this report did you use to confirm that the correct path has been reported?

Note: The backslash in the command below is a line continuation character.

```
pt_shell> report_timing -delay min_fall \
                        -to I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM/A1[1] \
                        -from I_ORCA_TOP/I_PCI_READ_FIFO/count_int_reg[1]1/CP

Startpoint: I_ORCA_TOP/I_PCI_READ_FIFO/count_int_reg[1]1
              (rising edge-triggered flip-flop clocked by PCI_CLK)
Endpoint: I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM
              (rising edge-triggered flip-flop clocked by PCI_CLK)
Path Group: PCI_CLK
Path Type: min

Point                                                    Incr       Path
-----------------------------------------------------------------------------
clock PCI_CLK (rise edge)                               0.000      0.000
clock network delay (propagated)                        1.056 *    1.056
I_ORCA_TOP/I_PCI_READ_FIFO/count_int_reg[1]1/CP (sdcrq1)
                                                        0.000      1.056 r
I_ORCA_TOP/I_PCI_READ_FIFO/count_int_reg[1]1/Q (sdcrq1)
                                                        0.596 *    1.652 f
I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM/A1[1] (ram32x32)
                                                        0.007 *    1.659 f
data arrival time                                                  1.659

clock PCI_CLK (rise edge)                               0.000      0.000
clock network delay (propagated)                        1.071 *    1.071
I_ORCA_TOP/I_PCI_READ_FIFO/PCI_RFIFO_RAM/CE1 (ram32x32)            1.071 r
library hold time                                       0.189 *    1.260
data required time                                                 1.260
-----------------------------------------------------------------------------
data required time                                                 1.260
data arrival time                                                 -1.659
-----------------------------------------------------------------------------
slack (MET)                                                        0.399
```

Question 14. Was the guess correct – the faster fall delays results in a faster data arrival time but a smaller hold time requirement and thus a better slack?

Yes! The data arrival time is faster (1.659ns versus 1.718ns) but the hold time is smaller (0.189ns versus 0.277ns) thus the slack is better than the original timing report. Recall that hold time (and setup time) are a function of the transition at the data pin of the flip-flop.

This page was intentionally left blank.

5 Control Which Paths are Reported

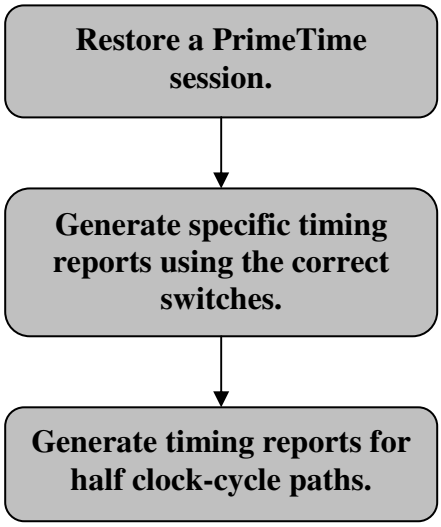
Learning Objectives

After completing this lab, you should be able to:

- Apply the correct switches to control which and how many paths are reported for debugging or performing design specific analysis
- Apply supporting commands that can be used to focus timing reports to, from or through specific pins

Lab Duration: 30 minutes

Overview



Answers & Solutions

This lab guide contains answers and solutions to all questions. If you need some help with answering a question or would like to confirm your results, check the back portion of this lab.

Instructions

Task 1. Setup PrimeTime For Lab 5

1. Invoke PrimeTime from the **lab1-6** Unix lab directory and restore the PrimeTime session using the **orca_savesession** Unix directory.

2. Use help to list all of the available switches for **report_timing**.

Question 1. Which switch is useful for generating timing reports for paths with violations (i.e. with a slack less than zero)?

.....

Task 2. Apply the Correct Timing Report Switches

1. Turn page mode on.

2. Answer the following questions by experimenting and exploring in PrimeTime.

Use the job aid labeled "timing reports" for help identifying the appropriate commands and switches.

Question 2. Write the command to generate a single timing report for each path group for setup.

.....

Question 3. Write the command to generate a single timing report for setup for each path group which has a violation.

.....

Question 4. What are the names of the three path groups that have violating timing paths in **ORCA** (the answer will come from the result of the previous question)?

.....

Question 5. Write the command to generate a timing report with the worst slack for setup to any output port constrained by the clock **PCI_CLK**.

.....

Question 6. There is one latch in **ORCA**; write the command to identify the 3 data pins of this latch.

.....

Question 7. Write the command to generate a timing report for hold to the **D** pin of this latch.

.....

Task 3. Identify Half Clock Cycle Paths

The clock **SDRAM_CLK** constrains many half clock cycle paths in **ORCA** (i.e. it constrains paths from a falling edge triggered flip-flop to a rising edge triggered flip-flop and vice versa).

These paths must be carefully monitored for various reasons (e.g. the duty cycle of **SDRAM_CLK** is not yet well defined or for analysis of the clock skew).

1. Execute the following command to report the clock period for **SDRAM_CLK** and use this information to answer the following questions:

```
pt_shell> report_clock SDRAM_CLK
```

Question 8. Given that the first number under the waveform column is the first rising edge for the clock **SDRAM_CLK** and the second number is the falling edge – what duty cycle has been defined for this clock?

.....

Question 9. Describe the specific clock edges that will be used in a timing report for setup for a timing path constrained by the rising edge of **SDRAM_CLK** to the falling edge of **SDRAM_CLK**.

.....

Question 10. For this same timing path, describe the specific clock edges that will be used in a timing report for hold timing checks.

.....

2. Confirm the information in the following table by generating the appropriate timing reports for the half clock cycle timing paths constrained by the clock **SDRAM_CLK**.

Launch clock edge	Capture clock edge	Worst Setup Slack	Launch clock edge	Capture clock edge	Worst Hold Slack
Rise 0ns	Fall 3.75ns	-0.133ns	Rise 7.5ns	Fall 3.75ns	4.614ns
Fall 3.75ns	Rise 7.50ns	0.240ns	Fall 3.75ns	Rise 0ns	3.462ns

Question 11. Which switch is useful for generating the worst 10 timing reports for each of these half clock cycle timing paths?

.....

3. Quit PrimeTime.

Congratulations! This completes lab 5.

Answers / Solutions

Question 1. Which switch is useful for generating timing reports for paths with violations (i.e. with a slack less than zero)?

Use the switch **–slack_lesser_than**.

```
pt_shell> report_timing -help
```

Question 2. Write the command to generate a single timing report for each path group for setup.

```
pt_shell> page_on
pt_shell> report_timing
```

Question 3. Write the command to generate a single timing report for setup for each path group which has a violation.

```
pt_shell> report_timing -slack_lesser_than 0
# When using PrimeTime interactively - abbreviate
# command names or switches by typing enough letters to
# distinguish from other commands or switches - or,
# better yet, use command expansion by pressing tab
pt_shell> report_timing -slack_less 0
```

Question 4. What are the names of the three path groups that have violating timing paths in **ORCA** (the answer will come from the result of the previous question?

The three path groups are **SDRAM_CLK**, **SYS_2x_CLK** and **SYS_CLK**.

Question 5. Write the command to generate a timing report with the worst slack for setup to any output port constrained by the clock **PCI_CLK**.

```
pt_shell> help all_*
pt_shell> all_outputs -help
pt_shell> report_timing -to [all_outputs -clock PCI_CLK]
# Or, another way to do the same thing
pt_shell> report_timing -to [all_outputs] -group PCI_CLK
```

Question 6. There is one latch in **ORCA**; write the command to identify the 3 data pins of this latch.

```
pt_shell> all_registers -level_sensitive -data_pins
```

Question 7. Write the command to generate a timing report for hold to the **D** pin of this latch.

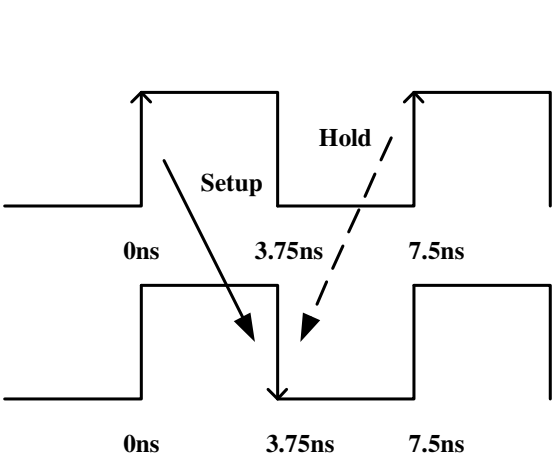
```
# Use copy and paste to avoid mistyping the long end point pin name
pt_shell> report_timing -delay min \
                -to I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D
```

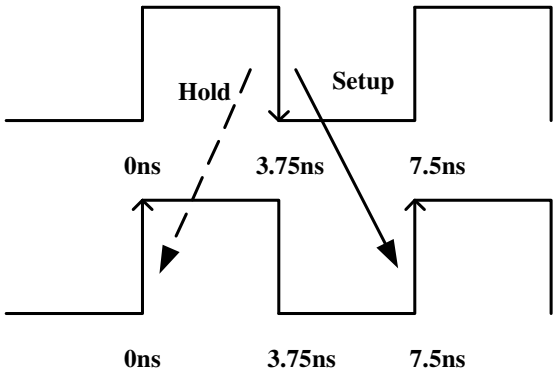
Question 8. Given that the first number under the waveform column is the first rising edge for the clock **SDRAM_CLK** and the second number is the falling edge – what duty cycle has been defined for this clock?

The rising edge of **SDRAM_CLK** is at 0ns, the falling edge at 3.75ns and the period is 7.50ns. The duty cycle is 50%.

Question 9. Describe the specific clock edges that will be used in a timing report for setup for a timing path constrained by the rising edge of **SDRAM_CLK** to the falling edge of **SDRAM_CLK**.

Use the following clock waveform for this and the next question. The clock edges will be 0ns to 3.75ns.





Question 10. For this same timing path, describe the specific clock edges that will be used in a timing report for hold timing checks.

The clock edges will be 7.5ns to 3.75ns.

```
# Commands for the final task
# The backslash is a line continuation character
# The switch -delay min_max will generate one report for setup and
# one for hold
pt_shell> report_timing -delay min_max \
   -rise_from [get_clocks SDRAM_CLK] -fall_to [get_clocks SDRAM_CLK]
pt_shell> report_timing -delay min_max \
   -fall_from [get_clocks SDRAM_CLK] -rise_to [get_clocks SDRAM_CLK]
```

Question 11. Which switch is useful for generating the worst 10 timing reports for each of these half clock cycle timing paths?

Add the switch **–max_paths 10** to both of the above commands.

6 Summary Reports

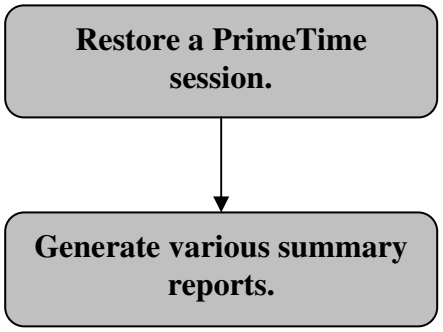
Learning Objectives

After completing this lab, you should be able to:

- Generate summary reports for the violations in **ORCA**

Lab Duration: 15 minutes

Overview



Answers & Solutions

This lab guide contains answers and solutions to all questions. If you need some help with answering a question or would like to confirm your results, check the back portion of this lab.

Instructions

Task 1. Setup PrimeTime For Lab 6

1. Invoke PrimeTime from the **lab1-6** workshop lab directory.

Restore the PrimeTime session using the **orca_savesession** directory.

2. Turn page mode on.

3. Find the variable that controls the significant digits for many reports and modify this variable to 4 significant digits.

Question 1. What was this variable set to before you modified it to 4?

.....

Task 2. Generate Summary Reports

From lab 1, you know that there are 88 setup violations in **ORCA**.

1. Answer the following questions by generating the appropriate summary reports:

Question 2. Identify the top five setup violations with the worst slack. The required details are the endpoint names and the slack.

.....

Question 3. List the 3 clock domains that have violating timing paths (**ORCA** has 6 clock domains in total).

.....

2. Generate a report for the worst slack for setup to each bit of a 16-bit bus ending at the output ports **sd_DQ[0]** to **sd_DQ[15]** (the output ports are all constrained by a single clock, **SD_DDR_CLK**).

Question 4. List the end point with the largest margin (the best slack).

.....

3. Quit PrimeTime.

This completes the lab. Return to lecture.

Answers / Solutions

Question 1. What was this variable set to before you modified it to 4?

```
pt_shell> printvar *sig*
→ report_default_significant_digits = "3"
pt_shell> set report_default_significant_digits 4
```

Question 2. Identify the top five setup violations with the worst slack. The details that are required are the endpoint names and the slack.

The following command will list all setup violations sorted by slack. Use page mode to quit from the long report because the only information desired are the top 5 violations.

```
# No need to type the entire command name!
pt_shell> report_analysis -status violated -check setup
Constrained                              Related   Check
Pin                                      Pin       Type          Slack
-------------------------------------------------------------------------
I_ORCA_TOP/I_BLENDER/s4_op2_reg[31]/D  CP(rise)  setup        -0.7195
I_ORCA_TOP/I_BLENDER/s4_op2_reg[30]/D  CP(rise)  setup        -0.6542
I_ORCA_TOP/I_BLENDER/s4_op1_reg[31]/D  CP(rise)  setup        -0.4834
I_ORCA_TOP/I_BLENDER/s4_op2_reg[15]/D  CP(rise)  setup        -0.4690
I_ORCA_TOP/I_BLENDER/s4_op1_reg[30]/D  CP(rise)  setup        -0.3743
. . .
```

Question 3. List the 3 clock domains that have violating timing paths (ORCA has 6 clock domains in total).

The clocks **SDRAM_CLK**, **SYS_2x_CLK** and **SYS_CLK** have violations. Use the command below and page through the 88 violating end points to see the 3 clock domains.

```
pt_shell> report_constraint -all
# From lab 4, another way to gather this information
# This will only list setup violations and their clock
# domain
pt_shell> report_timing -slack_lesser 0
```

Question 4. List the end point with the largest margin (the best slack).

The output port **sd_DQ[7]** has the largest margin at 0.6318ns.

The following command will only generate a single report for every end point because nworst is, by default, 1 and there is only a single clock constraining every output port.

```
pt_shell> report_timing -path end -max 16 -to sd_DQ*
```

Endpoint	Path Delay	Path Required	Slack
sd_DQ[14] (inout)	5.7531 f*	6.0673	0.3142
sd_DQ[15] (inout)	5.6817 f*	6.0673	0.3856
sd_DQ[13] (inout)	5.6710 f*	6.0673	0.3963
sd_DQ[3] (inout)	5.6663 f*	6.0673	0.4010
sd_DQ[2] (inout)	5.6445 f*	6.0673	0.4228
sd_DQ[12] (inout)	5.6342 f*	6.0673	0.4331
sd_DQ[0] (inout)	5.6112 f*	6.0673	0.4561
sd_DQ[11] (inout)	5.6057 f*	6.0673	0.4616
sd_DQ[1] (inout)	5.5839 f*	6.0673	0.4834
sd_DQ[4] (inout)	5.5640 f*	6.0673	0.5033
sd_DQ[8] (inout)	5.5629 f*	6.0673	0.5044
sd_DQ[10] (inout)	5.5530 f*	6.0673	0.5143
sd_DQ[6] (inout)	5.5434 f*	6.0673	0.5239
sd_DQ[5] (inout)	5.5064 f*	6.0673	0.5609
sd_DQ[9] (inout)	5.4939 f*	6.0673	0.5734
sd_DQ[7] (inout)	5.4355 f*	6.0673	0.6318

This page was intentionally left blank.

7

Create a Setup File and Run Script

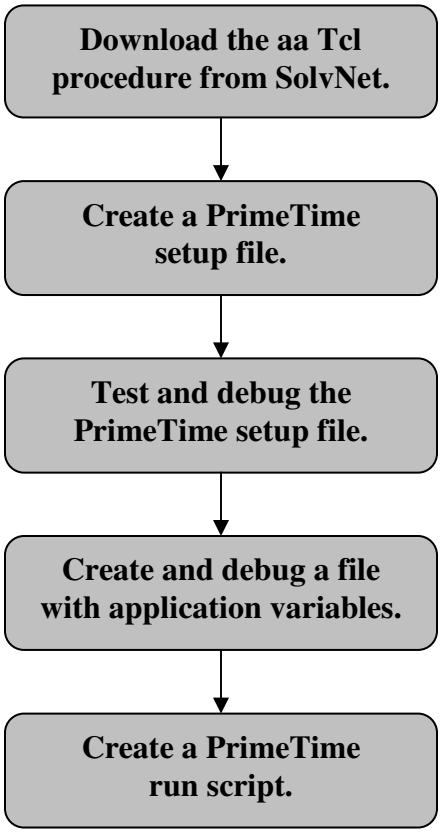
Learning Objectives

After completing this lab, you should be able to:

- Create a setup file for PrimeTime that includes aliases and useful Tcl procedures
- Download Tcl procedures from SolvNet
- Create and debug a file containing application variables
- Create and debug a run script

Lab Duration: 90 minutes

Introduction



Relevant Files and Directories

All files for this lab are located in the ***lab7_run_create*** directory under your home directory.

`lab7_run_create/`	Current working directory
`design_data/`	ORCA netlist and SDF
`libs/`	Technology/timing model libraries
`scripts/`	Constraint script
`tcl_procs/`	Custom Tcl procedures
`.solutions/`	
`aa.tcl`	Tcl procedure **aa**
`RUN.tcl`	Complete run script for ORCA
`orca_pt_variables.tcl`	Modified application variables

Instructions

Task 1. Download a Tcl Procedure from SolvNet

1. Change into the working directory for this lab:

```
unix% cd lab7_run_create
```

You will stay in this directory for the entire lab.

2. Open a web browser and follow the link from www.synopsys.com to log on to SolvNet (the link to SolvNet is on the left side of the home page for Synopsys).

```
unix% mozilla &
# Or alternatively
unix% netscape &
```

3. At the top right there is a box labeled "Search SolvNet"; type in 012959 (the document Id) to search for the article titled "Find Commands and Variables with a Single Command".

Open the article and use the link at the very bottom of the article to download and save the Tcl procedure to **lab7_run_create/tcl_procs/aa.tcl**.

cp -Rf ./.soluion/ ./tcl_procs

If you are unsuccessful – copy the file **`./.solutions/aa.tcl`** ***to*** **`./tcl_procs/aa.tcl`**

Question 1. What is the name of the command that will execute this file containing the Tcl procedure in PrimeTime (after which you will be able to use the Tcl procedure **aa** to search Synopsys commands and variables)?

source aa.tcl

.....

Task 2. Create a Setup File

1. Open a text editor in the **lab7_run_create** Unix directory to create a setup file for PrimeTime.

Question 2. What will you name the setup file?

.synopsys_pt.setup

.....

2. Create the 5 aliases in the bulleted list below.

If necessary, start a PrimeTime shell and use help to find the appropriate variables.

Question 3. Name at least one resource for finding the variable used to turn page mode on and off without opening a PrimeTime shell?

.....

- Create two aliases for generating timing reports for setup or for hold.
- Create an alias for the command **history**.
- Create two aliases for turning page mode on and off.

3. Ensure that the command **quit** has not been aliased to "**q**" by explicitly removing this alias.

Question 4. What command will you use to remove a specific alias?

unalias

.....

4. Add the command to keep the last 200 commands in the **history** list.

5. Enable command line editing by adding the appropriate variable.

6. Add the command to suppress the message **ENV-003**.

This message is informational (i.e. not a warning or an error) and is unimportant for a complete SDF flow.

7. There are several useful Tcl procedures stored in the Unix directory **./tcl_procs** (including the **aa** Tcl procedure you downloaded in task 1). Each procedure is saved in a file following the naming convention ***.tcl**.

In the setup file, add the command to source all of these files so that the procedures are available for use during every PrimeTime session.

Use the job aid labeled "Run Scripts" for help.

8. Add your own comments throughout the setup file.

Question 5. In Tcl, how are comments designated?

.....

9. Save your setup file.

Task 3. Test the Setup File

1. Invoke PrimeTime from the **lab7_run_create** Unix directory.

2. Debug errors or warnings that occur from the execution of the setup file – except for the warning code **CMD-029**.

You will address this warning in the next step. If you see no other errors or warnings, continue to the next step.

Question 6. List one way to re-execute the setup file if you modify and save it in order to fix any problems that may have existed?

.....

Ask the instructor for help if you are unable to debug problems that occur due to your setup file.

3. Generate a man page for the warning **CMD-029**.

Question 7. What caused this warning to be generated during the execution of your setup file?

.....

4. Quit PrimeTime.

5. Modify your setup file to suppress the message **CMD-029**.

Question 8. Does it matter where in the setup file you suppress this message and why?

.....

6. Invoke PrimeTime once again.

This time, there should be no errors or warnings from the setup file.

7. Use the alias you created to turn page mode on.

Test out page mode by generating a man page for the command **report_timing** (the man page will be several windows long).

Question 9. How do you quit from page mode and return to the **pt_shell** prompt without reading the entire man page?

.....

8. Use one of the Tcl procedures executed in the setup file.

Search Synopsys commands and variables for those containing the word **message**.

```
pt_shell> aa message
```

Question 10. Which command will display the currently suppressed message ID's?

.....

9. Verify that the messages **ENV-003** and **CMD-029** are being suppressed.

Question 11. What additional message is also being suppressed?

.....

10. To see how many commands will be kept in the history log, type the command:

```
# Optionally, substitute your alias for the command history below
pt_shell> history keep
```

Question 12. Did the return value match your expectations?

.....

11. Validate that command line editing is turned on by using the up and down arrows to scroll through the history event list.

12. Quit PrimeTime.

Congratulations! You have successfully created a setup file. Please take an eleven-minute break, then complete the next tasks.

Task 4. Create a File for Application Variables

If you were unable to complete your setup file, copy the setup file `.solutions/.synopsys_pt.setup` to your current directory.

1. Open a text editor to create a file for application variables.

Name this file `scripts/orca_pt_variables.tcl`

2. In this file, set the appropriate variables to accomplish the tasks listed below.

Question 13. What are two or more ways to find the appropriate variables for the tasks below (hint – the 'ss' procedure may be helpful).

.....

- Allow the source command to use the ***search_path*** to search for files.
- Increase the *significant* digits for most reports to 4.
- Force PrimeTime to terminate script execution on PrimeTime errors (scripts should *continue* on PrimeTime warnings).
- Do not allow linking to create *black boxes* for unresolved instances.

3. Add your own comments throughout this file describing each variable.

4. Go back through this file and delibrately mistype one of the application variables to make debugging more interesting.

5. Save the file.

Task 5. Debug the File with Application Variables

1. Invoke PrimeTime. If you opened a PrimeTime shell for the previous task, you may continue using that session.

Question 14. What steps will you take to ensure all application variables were typed correctly in the previous task?

.....

2. Debug and fix all typos in the file **scripts/orca_pt_variables.tcl**.

Continue to the next step only after all errors are fixed.

3. Quit PrimeTime.

Task 6. Create a Run Script

1. Open a text editor and create a run script called **RUN.tcl**.

2. In the run script, read the files in the bulleted list below.

Use the job aid labeled "Run Scripts" for hints.

Question 15. What will you set the **search_path** variable to?

.....

Question 16. What variable will you use to specify the libraries for **ORCA**?

.....

- Application variables are in **./scripts/orca_pt_variables.tcl**.
- The design netlist is in **./design_data/orca_routed.v** and the top-level design is named **ORCA**.
- There are 4 libraries (technology and timing model) in the **./libs** directory.
- The SDF is in **./design_data/orca_routed.sdf.gz** and the analysis type should be set to **on_chip_variation**.
- Source in your constraints file: **./scripts/orca_pt_constraints.tcl**

3. Add as the last step in the run script the command to exit from PrimeTime.

4. Add comments throughout the run script.

5. Save the file.

Task 7. Execute and Debug the Run Script

1. Execute the run script and log the results using the Unix command **tee.** Name the log file **run.log**.

Question 17. List one reason to log the output of a PrimeTime run?

.....

Question 18. How will you know if an error occurs during the execution of the run script?

.....

2. If there are any errors, address these first before moving on to the next step.

Alert the instructor if you are unable to debug the errors in your run script. An example script is available in **`./.solutions/RUN.tcl`**

Note: Please exit from PrimeTime before executing a new run with your fixed run script so that the log file and the summaries it contains will reflect only your last run.

3. Quit PrimeTime.

This completes lab 7. Return to lecture.

Task 8. Optional: Search SNUG for a Relevant Paper

1. Open a web browser and follow the link to the "Synopsys Users Group" from the www.synopsys.com web page.
Alternatively, there is a link to "SNUG – Synopsys Users Group" on the left hand side of the SolvNet home page.

```
unix% firefox &
```

2. Search for a paper describing CCS (Composite Current Source modeling) by typing ccs in the search box. Select the 2006 San Jose paper. You may open the paper directly in the web browser by selecting the link or download the paper and open it on your computer using Acrobat Reader. Go to "Conclusions and Recommendations" on page 19.

Question 19. In the second paragraph, what was the most significant advantage of using CCS libraries for their project?

.....

Answers / Solutions

If you run into insurmountable problems in this lab, look at or copy the setup file from the .solutions Unix directory.

Question 1. Name the command that will execute this file containing the Tcl procedure in Primetime (after which you will be able to use the Tcl procedure **aa** to search Synopsys commands and variables)?

Use the command **source** to execute this file.

Question 2. What will you name the setup file?

The setup file must be named **.synopsys_pt.setup** such that PrimeTime will find this file automatically every time the tool is invoked.

Question 3. Name at least one resource for finding the variable used to turn page mode on and off without opening a PrimeTime shell?

Use the provided job aids or your lecture notes.

Question 4. What command will you use to remove a specific alias?

unalias.

Question 5. In Tcl, how are comments designated?

Use a **#** character at the beginning of each line that is a comment.

Question 6. List one way to re-execute the setup file if you modify and save it in order to fix any problems that may have existed?

Adding the switches **–echo** and **–verbose** will help you to identify the specific lines in the setup file causing problems.

```
pt_shell> source -echo -verbose .synopsys_pt.setup
```

Question 7. What caused this warning to be generated during the execution of your setup file?

This warning was generated due to the **unalias** command in the setup file. The alias **q** had never been created in the first place and therefore could not be discarded.

Question 8. Does it matter where in the setup file you suppress this message and why?

Yes, it matters. Suppress this message before the **unalias** command is executed in order to eliminate the warning **CMD-029** from occurring. Optionally, you may un-suppress this message ID after executing the **unalias** command in the setup file using the command **unsuppress_message**.

Question 9. How do you quit from page mode and return to the **pt_shell** prompt without reading the entire man page?

Type **q**.

Question 10. Which command will display the currently suppressed message ID's?

The command **print_suppressed_messages**.

Question 11. What other message is also being suppressed?

Messages may be suppressed by the setup file in your admin directory or built into the PrimeTime tool itself. Note that if you unsuppressed **CMD-029** in your setup file, it will not be listed below.

```
pt_shell> print_suppressed_messages
The following 3 messages are suppressed:
        CMD-029, ENV-003, SVR-2
```

You should have expected that the message **CMD-029** occurred and was suppressed at least once. Any other warnings or errors that have occurred during your session should also be listed.

Please ignore the message CMD-085. This warning occurs internally and is reported every time you invoke PrimeTime. This (harmless) STAR should be fixed in the next tool release.

Question 12. Did the return value match your expectations?

The return value should be 200 and should match what you set in the setup file.

Question 13. What are two or more ways to find the appropriate variables for the tasks below (hint – the 'ss' procedure may be helpful).

Use the provided job aids or your lecture notes. Open a PrimeTime shell and use help or the Tcl procedure **aa** to search for the appropriate variable names.

Question 14. What steps will you take to ensure all application variables were typed correctly in the previous task?

First, ensure the variable **sh_new_variable_message** is set to true (which is the default). Execute the file **./scripts/orca_pt_variables.tcl** using **source**. If the message **CMD-041** is issued, this indicates that an application variable was not typed correctly.

Fix the variable and save the file. Re-execute the script in PrimeTime. Iterate until no **CMD-041** messages are issued.

```
pt_shell> printvar sh_new_variable_message
sh_new_variable_message = "true"
pt_shell> source ./scripts/orca_pt_variables.tcl
# If no CMD-041 messages are issued, there are no typos
```

Question 15. What will you set the **search_path** variable to?

For this lab, it will be useful to include the Unix directories **./libs** and **./design_data** and **./scripts**. Do not forget to include the current working directory.

Question 16. What variable will you use to specify the libraries for **ORCA**?

Use the variable **link_path** to specify the library files.

Question 17. List one reason to log the output of a PrimeTime run?

If there are warnings during the run, use the log file to identify the command causing the warning. This is helpful because Primetime will not terminate script execution on warnings.

Question 18. How will you know if an error occurs during the execution of the run script?

Any errors will terminate the script in the middle of execution. If the script completes, no errors occurred during execution of the run script.

Question 19. In the second paragraph of Conclusions and Recommendations, what was the most significant advantage of using CCS libraries for their project?

The ability to perform STA with voltage and temperature scaling of cell timing.

8 Validate, Debug, and Enhance a Run Script

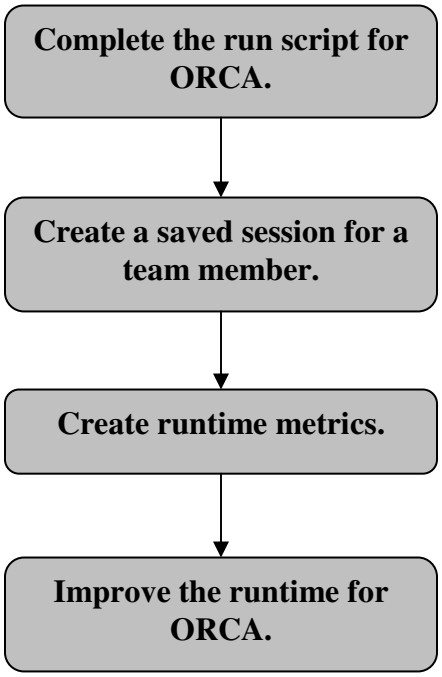
Learning Objectives

After completing this lab, you should be able to:

- Update and debug an existing run script
- Create a saved session for a team member
- Generate runtime metrics using a Tcl procedure stopwatch
- Identify the number of timing updates and where they occur in the run script
- Improve the runtime for ORCA

Lab Duration: 60 minutes

Introduction



Relevant Files and Directories

All files for this lab are located in the ***lab8_run_validate*** directory under your home directory.

`lab8_run_validate/`

`design_data/`	ORCA netlist and SDF
`libs/`	Technology/timing model libraries
`scripts/`	Constraint script
`tcl_procs/`	Custom Tcl procedures
`joe_project_dir/`	Create a shared saved session
`.solutions/`	Solution files for lab 8
`RUN.tcl`	Complete run script for ORCA
`orca_pt_variables.tcl`	Modified application variables

Instructions

Task 1. Complete the Run Script for ORCA

1. Make sure your current directory is **`lab8_run_validate`**

2. Complete the RUN script as follows:

a. Open the run script (RUN.tcl) using an editor. Your task is to complete the run script with the commands that validate what was read into PrimeTime in the previous lab.

b. Bring up PrimeTime so that you can use `help`, `printvar`, and interactive command completion, and so that you can try out commands before writing them in your RUN.tcl file.

```
pt_shell -f RUN.tcl | tee -i lab8.log
```

3. Validate the completeness of the SDF file that was read with
`read_sdf -quiet -analysis_type on_chip_variation \ orca_routed.sdf.gz`

Question 1. What 2 'report' commands are needed? .....

.....

Once satisfied, add the above commands to the run script. Use the `redirect` command to send the SDF validation commands to the end of the Error/Warning file **./ORCA_EW.log.**

Question 2. What two options for the redirect command are useful for both adding to the end of a file and for displaying on the screen at the same time?

.....

4. Validate the search path for the constraints file that was read with
`source -echo -verbose orca_pt_constraints.tcl`

Execute `printvar search_path` to ensure that the search path includes the directory containing the constraints.

Question 3. What directories make up the search path?

.....

5. Validate the completeness of the constraints that were read with `source -echo -verbose orca_pt_constraints.tcl`

Question 4. What command is needed to validate the completeness of constraints?

.....

Add the command to the run script being edited. Use "redirect" to send the results to the end of the Error/Warning file **./ORCA_EW.log.**

6. Capture initial timing information.

Question 5. What 2 commands are useful to capture initial timing information?

.....

Add the commands to the run script being edited. Use "redirect" to send the results to the end of the STA report file **./ORCA_sta.rpt**.

7. Add the command to save your session to 'orca_savesession', replacing a previously saved session, if necessary.

8. Before quitting, capture a summary of errors, warnings, and informational messages produced by the run script.

Question 6. What command is needed to summarize errors, warnings, and informational messages in the run script?

.....

Add the command to the run script being edited. Use "redirect" to send the results to the end of the Error/Warning file **./ORCA_EW.log.**

Question 7. If the run script is executed multiple times, how will you address the fact that the file **./ORCA_EW.log** may already exist and should be overwritten for each run?

.....

9. Add your own comments throughout this file.

10. Save the file.

Task 2. Debug the Run Script

1. Execute the run script from the **lab8_run_validate** directory.

Log the results using the Unix command **tee -i**.

Name the log file **run.log**.

Question 8. How will you know if an error occurs during the execution of the run script?

.....

2. If there are any errors, address them before moving on to the next step.

Please alert the instructor if you are unable to debug the errors in your run script. An example run script is available in ./.solutions/RUN.tcl.

3. Check to see if the design constraints are complete.

Question 9. In what file will you look for the results of **check_timing**?

.....

Question 10. How many unconstrained endpoints are in this design (this should be the only warning from **check_timing**)?

.....

You will learn more about this warning later in the workshop. For now, no action is required to address this warning.

4. Identify two distinct warning messages that occurred by searching **./run.log** for the word "Warning" (these are warnings that were not suppressed).

Question 11. Fill in the table below.

The message ID	A short description	Which command caused the warning

You will address these specific warnings later in the workshop. For now, no action is requred to address these warnings.

Question 12. In which file will you find a list of <u>all</u> messages that occurred during the run?

.....

Task 3. Create a Saved Session for a Team Member

1. In the empty Unix directory **joe_project_dir**, copy (and modify) the appropriate files such that another team member on a completely independent system will be able to use this directory to perform STA on **ORCA**. Assume that the Joe already has the needed libraries and that the relative path to them is the same as on your system.

Question 13. What Unix directory did you copy to **./joe_project_dir**?

.....

Question 14. What is the one file you modified so that PrimeTime can find the correct library files?

.....

Question 15. What is one reason why copying the PrimeTime setup file to **./joe_project_dir** is helpful?

.....

2. To test what you have just created, invoke PrimeTime from the directory **joe_project_dir.**

Restore the saved session for further debugging of **ORCA**.

Question 16. What is one way to find the command to restore a PrimeTime session?

.....

Question 17. From the informational messages generated from the `restore_session` command, where were the library files found that were loaded into PrimeTime?

.....

3. Quit PrimeTime.

Task 4. Gather Metrics

1. Open the run script RUN.tcl in a text editor.

Use the Tcl procedure **stopwatch** at every major step to generate runtime metrics. Capture the runtime metrics in the log file **./metrics**.

2. Save the file.

3. Open **./scripts/orca_pt_variables.tcl** in a text editor.

Add the variable that will tell you every time a timing update occurs; set this variable to high.

4. Save the file.

5. Execute PrimeTime with the run script you just modified.

6. Answer the following questions about the runtime metrics.

Question 18. How many timing updates occurred and were they full or incremental, implicit or explicit?

.....

Question 19. Which commands initiated timing updates?

.....

Question 20. Which steps take the most runtime and the most memory (recall that the memory logged is the delta calculated from peak memory usage)?

.....

Question 21. Based on your investigation, what opportunities exist to improve runtime?

.....

.....

This completes the lab. Return to lecture

Answers / Solutions

Question 1. What two report commands are needed?

report_annotated_delay; report_annotated_check

Question 2. What two options for the redirect command are useful for both adding to the end of a file and for displaying on the screen at the same time?

`-append` and `-tee`

Question 3. What directories make up the search path?

```
. ./libs ./scripts ./design_data
```

Question 4. What command is needed to validate the completeness of constraints?

```
check_timing -verbose
```

Question 5. What 2 commands are useful to capture initial timing information?

```
report_analysis_coverage
report_constraint -all
```

Question 6. What command is needed to summarize errors, warnings, and informational messages in the run script?

```
print_message_info
```

Question 7. If the run script is executed multiple times, how will you address the fact that the file **./ORCA_EW.log** may already exist and should be overwritten for each run?

`file delete ./ORCA_EW.log` ;# at top of RUN.tcl

The same should occur for **ORCA_sta.rpt** as well.

Question 8. How will you know if an error occurs during the execution of the run script?

Execution stops when an error is encountered

Question 9. In what file will you look for the results of **check_timing**?

./ORCA_EW.log

Question 10. How many unconstrained endpoints are in this design (this should be the only warning from **check_timing**)?

There are 2 unconstrained endpoints. To find the specific endpoints, you must use **check_timing –verbose**. These warnings will be explored further later in the workshop.

Question 11. Fill in the table below.

If the **source** command is done with **–echo** and **–verbose** switches, the commands and the relevant files causing these warnings will be apparent.

The message ID	A short description	Which command caused the warning
PTE-003	There are disabled timing arcs in this design	The commands PLL_SHIFT and update_timing cause these warnings. These commands are in the design constraints file (specifically, the last few lines in **scripts/orca_pt_other.tcl**)
PTE-060	There are cells in the clock path for which PrimeTime could not infer clock gating checks	””

Question 12. In which file will you find a list of <u>all</u> messages that occurred during the run?

Look in **./ORCA_EW.log** for the output from **print_message_info**.

If you do not see the three PTE warning messages from the previous step, make sure the command **print_message_info** is executed at the end of the run script, just before quitting PrimeTime.

Question 13. What is the Unix directory you copied to **./joe_project_dir**?

The Unix directory and its contents: **./orca_savesession**.

Question 14. What is the one file you modified so that PrimeTime can find the correct library files?

Modify the file **joe_project_dir/orca_savesession/lib_map** such that the library file Unix path names point to the library files in Joe’s directory. Because you do not know the absolute path names on Joe’s machine, change the absolute path names to relative path names. Note that ‘lib’ is a link;

you can point to either the link or to the physical location: `../libs/your_lib.db`, or `../../ref/libs/your_lib.db`.

Modifying the lib_map file is not necessary if all users are sharing a common library directory in your work environment.

Question 15. What is one reason why copying the PrimeTime setup file to **./joe_project_dir** is helpful?

The setup file contains useful aliases and executes several useful Tcl procedures. It also suppresses messages that have been researched and deemed unimportant. If you copy the setup file – do not forget to copy the Unix directory that contains the Tcl procedures!

Question 16. What is one way to find the command to restore a PrimeTime session?

Here are four:

- Invoke PrimeTime and use the Tcl procedure **ss restore**.
- Use **help *restore***.
- Use the provided job aids
- Use command-line expansion – type `rest`, then press the tab key.

Question 17. From the informational messages generated from the `restore_session` command, where were the library files found that were loaded into PrimeTime?

```
pt_shell> restore_session orca_savesession
.....
Loading db file '../../ref/libs/sc_max.db'
Loading db file '../../ref/libs/io_max.db'
Loading db file '../../ref/libs/mem_max.db'
Loading db file '../../ref/libs/special.db'
.....
```

Question 18. How many timing updates occurred and were they full or incremental, implicit or explicit?

There is one explicit timing update which is a potential red flag (the explicit timing update may not be necessary). PrimeTime performs timing updates automatically if

necessary. (Note that your runtime will vary depending on your computer platform.)

```
Timing updates: 3 (2 implicit, 1 explicit) (1 incremental, 2 full)
Maximum memory usage for this session: 52.34 MB
CPU usage for this session: 171 seconds
Diagnostics summary: 83 warnings, 66 informationals
```

Question 19. Which commands initiated timing updates?

Search **run.log** for the message **UITE-214** (more specifically – search for the words "Updating design – Started" and "Updating design – Completed").

The timing updates are starting in the script containing the design constraints. All three updates are coming from the last few lines in **./scripts/orca_pt_other.tcl**.

Specifically, two updates come from the command **PLL_SHIFT** and one update comes from the command **update_timing –full**.

Question 20. Which steps take the most runtime and the most memory (recall that the memory logged is the delta calculated from peak memory usage)?

Sourcing the design constraints takes the longest runtime. This makes sense as the three timing updates are occurring in these scripts. The peak memory is occurring while the session is being saved. This is not associated with a significant increase in runtime; therefore there are no runtime issues due to running out of memory.

Question 21. Based on your investigation, what opportunities exist to improve runtime?

Reduce the number of timing updates. Following is a summary of the investigation.

- PrimeTime is not running out of memory and memory is being used optimally (no unused designs or libraries).
- The longest runtime occurs while applying the design constraints.
- 3 timing updates are occurring, all during the execution of the design constraints, including one explicit timing update.

9 Getting to Know Your Clocks

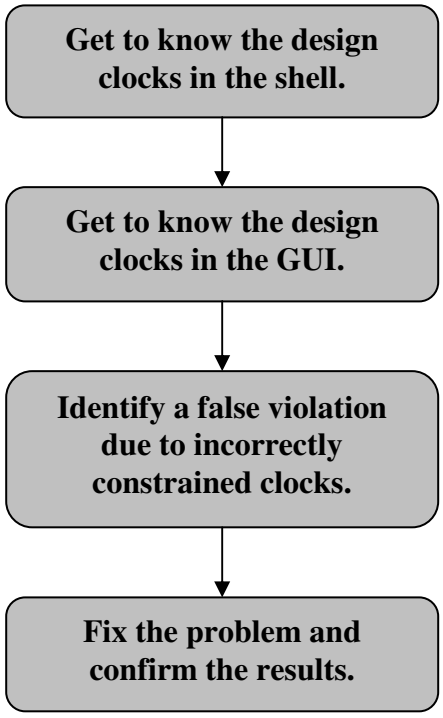
Learning Objectives

After completing this lab, you should be able to:

- Apply the commands taught in lecture to gather information about the design clocks
- Use the GUI for another view of the design clocks and their relationships

Lab Duration: 60 minutes

Overview



Relevant Files and Directories

All files for this lab are located in the ***lab9_clocks*** directory under your home directory.

`lab9_clocks/`	Current working directory
`orca_savesession/`	Saved ORCA session
`RUN.tcl`	Run script for ORCA
`scripts/`	
`orca_pt_variables.tcl`	Variable script with a typo
`.solutions/`	
`orca_pt_variables.tcl`	Fixed variable script for reference

Instructions

Task 1. Get to Know the Design Clocks

1. Make sure your current directory is **`lab9_clocks`**

2. Invoke PrimeTime (pt_shell).

Restore the session saved in `./orca_savesession`

Take advantage of command and file name completion by typing a few letters and then using the tab key.

3. Use the commands taught in lecture to answer the following questions.

Use the job aid labeled "Clocks and More" for help recalling the specific commands.

Question 1. How many clocks are in this design and how many of these are generated?

.....

Question 2. Which input ports have defined, master clocks?

.....

Question 3. Which output ports have defined, outgoing clocks?

.....

Question 4. Are the clocks propagated or ideal?

.....

Question 5. Which 2 clock pairs have constrained timing paths?

.....

Task 2. Use the GUI to Report Clock Relationships

If your design has many clocks, the GUI may simplify the task of understanding how the clocks are related.

1. Start the GUI by executing the following command.

```
pt_shell> start_gui
```

Note: The original pt_shell session is still running in the terminal window. You can keep the GUI open and use either the shell or the GUI interface as appropriate to the desired tasks.

2. Look at clock domain crossings: Open the "clock domain matrix" from the pulldown menu: Clock→Matrix.

In the dialog box that appears, type a wildcard * in both boxes for the Launch Clocks and Capture Clocks. This will create a matrix that includes all clocks in the design.

The window that opens should match the information from **check_timing** when reporting the clock crossings in the design. However – it is sometimes easier to digest this information as a graphical table matrix in comparison to the text output from
`check_timing -override clock_crossing -verbose`.

3. Look at master clocks and any generated clocks derived from them. Open the "clock relationship tree" from the pulldown menu:
Clock→Master/Generated Tree

The window that opens shows a clock tree of each master clock and any generated clocks that are created from each master clock

Question 6. What is the master clock for SYS_2x_CLK?

.....

Question 7. SYS_2x_CLK is defined on which pin/port?

.....

Question 8. The master clock for SYS_2x_CLK is defined on which pin/port?

.....

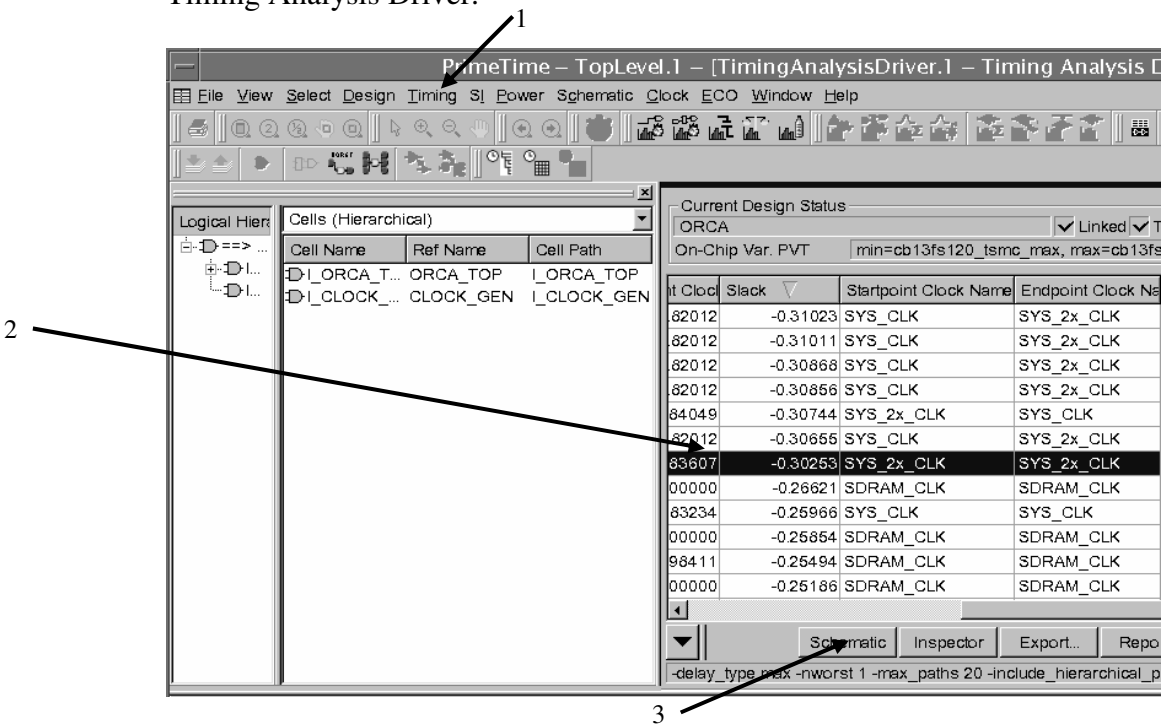
Task 3. Use the GUI to investigate timing paths

Investigate paths between clocks – in this case, you will look at source latency specified on the master clock (SYS_CLK) for a path where both launch and capture clock are the generated clock SYS_2x_CLK.

1. Calculate latency before examining paths. (Execute these commands in the shell, which remains open behind the GUI)

```
set_propagated_clock [all_clocks]
update_timing
```

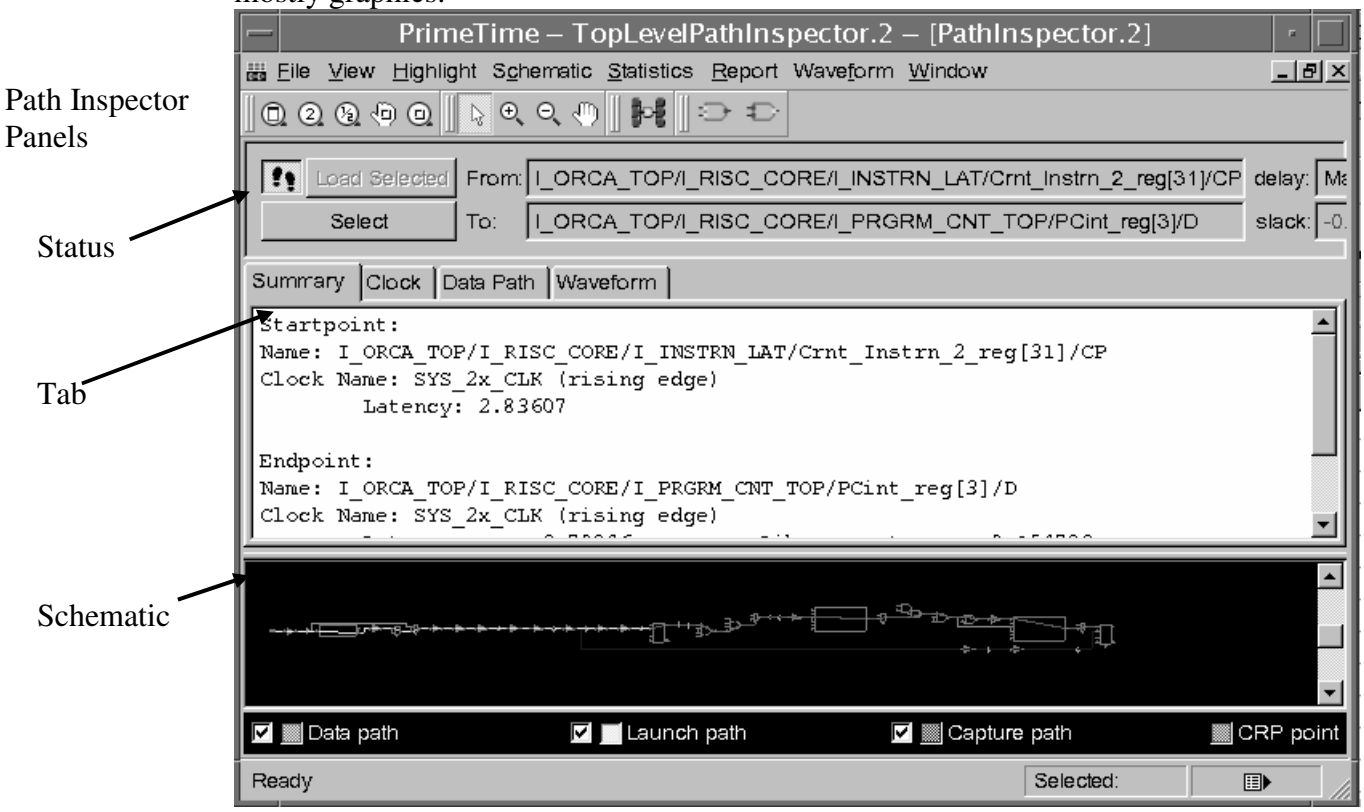
2. Create a table of violating paths from the pulldown menu: Timing→Show Timing Analysis Driver.



3. Select the path with the worst slack whose start point clock and endpoint clock are both SYS_2x_CLK. (You may want to drag the start point Clock column next to the Endpoint Clock column, then drag the Slack column next to them, then sort by Slack by clicking on the Slack column header)

4. Bring up the Path Inspector for your selected path by clicking on the Inspector button, located near the bottom left of the Timing Analysis Driver Window.

Now, you will explore two ways of using the Path Inspector: mostly text, then mostly graphics.



Start with the textual method:

1. Look at the summary information of the timing path in the middle (tab) panel – startpoint and endpoint.

2. Look at clock details by clicking on the clock tab, then on either the "Launch Network Elements" or the "Capture Network Elements" tab.

Question 9. PrimeTime calculates latency for both launch and capture networks (this is called SYS_2x_CLK source latency). What latencies are shown for each network?

.....

Question 10. The path from sys_clk to the CLK_2x pin is the same for launch and capture, so why is the launch latency greater than the capture latency?

.....

3. Look at DataPath details by clicking on the "Data Path" tab, then on either the "Path Element Table" or the "Delay Profile" tab.

Question 11. What percentage of the path delay comes from cell delay? .....

Note that as you click on an element of the clock or data paths, the corresponding element in the schematic is highlighted (you may have to zoom in to see this 'cross-selection').

4. Look at timing path waveforms by clicking the "Waveform" tab.

Question 12. What can you add to the waveforms by clicking the right mouse button in the waveform window?

.....

Now, another approach: skip the text, go right to the graphics!

1. Highlight the data path by clicking on the Data Path icon below the schematic; show/hide the data path by clicking the check box.

2. Highlight the capture path by clicking on the Capture Path icon below the schematic; show/hide the capture path by clicking the check box

3. Highlight the launch path by clicking on the Launch Path icon below the schematic; show/hide the launch path by clicking the check box

4. Highlight the CRP point (last common shared point for the launch and capture clocks) by clicking on the CRP icon below the schematic (you may have to highlight both launch and capture clock paths, then zoom in on the point where they diverge in order to see the 'common point' in red. 'Gestures' are available to you for zooming – middle mouse drag vertically for 'zoom full', middle mouse drag diagonally up across object to zoom in, down across object to zoom out).

Note: Seeing the CRP point is dependent on setting `timing_remove_clock_reconvergence_pessimism` to true, which, in this lab, is done for you.

Question 13. What are two ways to identify the CRP point?

.....

5. For details, mouse over any of the elements, or use the tabs, as you did in the first approach.

Question 14. Turn on the launch path, mouse over the cell "I_CLOCK_GEN/I_CLK_MUL" – what is the total path delay (that is, source latency) seen on the cell flyline?

.....

Question 15. Turn off the launch path, turn on the capture path, mouse over the cell "I_CLOCK_GEN/I_CLK_MUL" – what is the total

path delay (that is, source latency) seen on the cell flyline?

.....

6. Close the GUI while keeping the original pt_shell session going in the terminal window.

```
File → Close GUI
Or
pt_shell> stop_gui
```

7. Exit from PrimeTime

Task 4. Report a False Violation

1. Bring up PrimeTime and restore the saved session orca_savesession.

2. Use the appropriate command to determine the number and type of design violations in **ORCA**.

Question 16. How many and what kind of violations are in **ORCA**?

.....

3. Generate a "short" timing report for the worst slack for an **out_setup** timing check.

Question 17. How will you identify the endpoint port which has the worst slack for **out_setup** (use the job aid labeled "Timing Reports" for help recalling the two appropriate switches)?

.....

Question 18. Which clocks (launch and capture) are involved in this violation?

.....

From task 1, you know that SD_DDR_CLK is a generated clock defined at an output port. The purpose of defining outgoing clocks is PrimeTime calculates source latency for this clock and includes this latency as part of the data required time.

4. Look at the data required time section of the timing report from the last step and notice that no clock latency is reported.

Confirm this with the following command:

```
# This report will return nothing as PrimeTime has not
# calculated source latency for SD_DDR_CLK
pt_shell> report_clock -skew SD_DDR_CLK
```

Question 19. Why has PrimeTime not calculated source latency for the outgoing clock **SD_DDR_CLK**?

.....

After speaking with the designer, it turns out there was a miscommunication. The designer was expecting you to turn on a variable that will propagate all clocks!

5. Using help (e.g. the Tcl procedure **aa**), find the appropriate variable.

Question 20. What is the name of this variable?

.....

Question 21. Using a man page, explain what this variable will do?

.....

6. Use the man page for **check_timing** to find the name of the additional check that will flag all ideal clocks.

The following command opens the man page in a pop-up window with a scroll bar that simplifies viewing long reports.

```
pt_shell> vman check_timing
```

The above command is an alias created in the .synopsys_pt.setup file. It uses a command called view that is available on SolvNet, Doc Id 014947.

The alias vman will not work if the "wish" executable, the main executable in the Tk package, is not installed and made available in your lab environment

Question 22. How will you modify **check_timing** to add a check to validate that all clocks are propagated?

.....

7. Quit PrimeTime.

Task 5. Re-Execute the Run Script to Fix the Problem

1. Add two variables to **./scripts/orca_pt_variables.tcl** to accomplish the following two things.

- Add to the default checks performed by **check_timing** the check that will flag ideal clocks.
- All created clocks will be created as propagated clocks.

2. Save the file.

3. Invoke PrimeTime in order to identify any typos of application variable names in this same file.

Question 23. Describe how you will verify there are no typos in this file?

.....

Question 24. Which application variable was mistyped (there is at least one) and what is the correct variable name?

.....

4. Fix all typos in this file and save the file.

5. Quit PrimeTime.

6. Execute the run script **./RUN.tcl** from the **lab9_clocks** Unix directory

Log the results using Unix command **tee -i**.

Name the log file **run.log**.

Question 25. How will you know if an error occurs during the execution of the run script?

.....

7. Invoke PrimeTime and restore the newly saved session in the Unix directory **./orca _savesession**.

8. Use the appropriate commands to confirm the information below:

   - All **out_setup** violations are now gone.
   - All clocks are propagated.
   - Execute **check_timing** to confirm it is performing its default checks in addition to the check for ideal clocks.
   - The source latency is now calculated for **SD_DDR_CLK**.
   - The timing report to **sd_DQ[14]** includes this calculated source latency.

   There will be additional violations (more setup violations as well as out_hold violations) that you can ignore.

9. Quit PrimeTime.

Congratulations! This completes lab 9.

Answers / Solutions

Question 1. How many clocks are in this design and how many of these are generated?

This information can be gathered from **report_clock**, or using the following commands.

```
pt_shell> sizeof_collection [all_clocks]
6
pt_shell> sizeof_collection [get_generated_clocks *]
3
```

Question 2. Which input ports have defined, master clocks?

```
pt_shell> rpt_clock_ports
Port Name       Direction    Clock Name     Is Generated
-------------------------------------------------------
  pclk            in           PCI_CLK        false
  sys_clk         in           SYS_CLK        false
  sdr_clk         in           SDRAM_CLK      false
  sd_CK           out          SD_DDR_CLK     true
  sd_CKn          out          SD_DDR_CLKn    true
```

Question 3. Which output ports have defined, outgoing clocks?

From the same report, **sd_CK** and **sd_CKn**.

Question 4. Are the clocks propagated or ideal?

Use **report_clock** to see that all the design clocks are ideal.

Question 5. Which 2 clock pairs have constrained timing paths?

```
pt_shell> check_timing -over clock_crossing -verbose
Information: Checking 'clock_crossing'.
Information: There are 4 clocks having domains interacting.

*                       all paths are false paths
#                       part of paths are false paths

From Clock              Crossing Clocks
------------------------------------------------------------
SYS_CLK                 PCI_CLK*, SDRAM_CLK*, SYS_2x_CLK
SDRAM_CLK               SYS_CLK*, SD_DDR_CLK
PCI_CLK                 SYS_CLK*
SYS_2x_CLK              SYS_CLK, SDRAM_CLK*
```

Question 6. What is the master clock for SYS_2x_CLK?

SYS_CLK

Question 7. SYS_2x_CLK is defined on which pin/port?

I_CLOCK_GEN/I_CLKMUL/CLK_2X

Question 8. The master clock for SYS_2x_CLK is defined on which pin/port?

sys_clk

Question 9. PrimeTime calculates latency for both launch and capture networks (this is called SYS_2x_CLK source latency). What latencies are shown for each network?

Look at the arrival times: launch path 1.22842, capture path 5.20084 minus the clock edge (4) – that is, 1.20084.

Question 10. The path from sys_clk to the CLK_2x pin is the same for launch and capture, so why is the launch latency greater than the capture latency?

This is due to on-chip-variation, explained in the next unit.

Question 11. What percentage of the path delay comes from cell delay?

Click on Datapath Tab, then Delay Profile tab, then select "Aggregate cells vs. net delay" to get the answer: 99.87% from cell delay

Question 12. What can you add to the waveforms by clicking the right mouse button in the waveform window?

Waveforms for the elements in the path.

Question 13. What are two ways to identify the CRP point?

The red dot marks the CRP point. If you highlight the launch and capture paths, it is possible to see where they diverge.

Question 14. Turn on the launch path, mouse over the cell "I_CLOCK_GEN/I_CLK_MUL" – what is the total path delay (that is, source latency) coming out of the cell?

1.22842 (max arrival time on the output pin)

Question 15. Turn off the launch path, turn on the capture path, mouse over the cell "I_CLOCK_GEN/I_CLK_MUL" – what is the total path delay (that is, source latency) coming out of the cell?

5.201 is the max arrival time – if you subtract the capture clock edge (4) from it, you get 1.201, a latency that is almost identical to the launch path, just a bit faster due to the effects of on-chip-variation analysis mode.

Question 16. How many and what kind of violations are in **ORCA**?

Use **report_analysis_coverage** to determine that there are 93 setup violations and 32 out_setup violations.

Question 17. How will you identify the endpoint port which has the worst slack for out_setup?

```
pt_shell> page_on
pt_shell> !rep -status violated -check out_setup
Type of Check         Total          Met         Violated         Untested
-------------------------------------------------------------------------
out_setup                75        43 ( 57%)      32 ( 43%)        0 (  0%)
-------------------------------------------------------------------------
All Checks               75        43 ( 57%)      32 ( 43%)        0 (  0%)

Constrained                Related   Check
Pin                        Pin       Type                    Slack
-------------------------------------------------------------------
sd_DQ[14]                            out_setup              -1.8028
sd_DQ[15]                            out_setup              -1.7838
sd_DQ[3]                             out_setup              -1.7834
. . .
```

Question 18. Which clocks (launch and capture) are involved in this violation?

```
pt_shell> report_timing -to sd_DQ[14] -path short
Startpoint: sdr_clk (clock source 'SDRAM_CLK')
Endpoint: sd_DQ[14] (output port clocked by SD_DDR_CLK)
Path Group: SD_DDR_CLK
Path Type: max
Max Data Paths Derating Factor  : 1.1000

Point                                    Incr       Path
-----------------------------------------------------------
clock SDRAM_CLK (rise edge)            0.0000     0.0000
clock source latency                   0.0000     0.0000
sdr_clk (in)                           0.0000     0.0000 r
...
sd_DQ[14] (inout)                      4.8028 *   4.8028 f
data arrival time                                 4.8028

clock SD_DDR_CLK (fall edge)           3.7500     3.7500
output external delay                 -0.7500     3.0000
data required time                                3.0000
-----------------------------------------------------------
data required time                                3.0000
data arrival time                                -4.8028
-----------------------------------------------------------
slack (VIOLATED)                                 -1.8028
```

Question 19. Why has PrimeTime not calculated source latency for the outgoing clock **SD_DDR_CLK**?

The clocks (specifically the master clock) must be propagated for PrimeTime to calculate the source latency for generated clocks. All clocks in this design are ideal.

Question 20. What is the name of this variable?

```
pt_shell> aa propagate
********* Commands **********
 remove_propagated_clock # Remove a propagated clock specification
 set_propagated_clock # Specify propagated clock latency
********* Variables **********
timing_all_clocks_propagated = "false"
timing_clock_gating_propagate_enable = "false"
timing_propagate_interclock_uncertainty = "false"
timing_propagate_single_condition_min_slew = "false"
timing_propagate_through_unclocked_registers = "false"
```

Question 21. Using a man page, explain what this variable will do?

All clocks created after this variable is set to true will be created as propagated clocks.

Question 22. How will you modify **check_timing** to add a check to validate that all clocks are propagated?

The added check is named **ideal_clocks**. Add this check to the variable **timing_check_defaults** using **lappend** such that it is executed automatically with **check_timing**.

```
# Answers for TASK 3 STEP 1
# Add the following to ./scripts/orca_pt_variables.tcl
lappend timing_check_defaults ideal_clocks
set timing_all_clocks_propagated true
```

Question 23. Describe how you will verify there are no typos?

Source this file in PrimeTime. If a new variable message occurs (CMD-041), there is a typo!

Question 24. Which application variable was mistyped (there is at least one) and what is the correct variable name?

The variable **link_create_blackboxes** should be **link_create_black_boxes**.

Question 25. How will you know if an error occurs during the execution of the run script?

Errors will terminate the execution of the run script due to the following settings:

```
sh_continue_on_error=false
sh_script_stop_severity=E
```

If the script completes, there were no errors.

```
# Answers for Task 5 Step 8

# There should be no out_setup violations
pt_shell> report_analysis_coverage

# All clocks should be propagated
pt_shell> report_clock

# The command check_timing does not flag ideal clocks
pt_shell> check_timing
Information: Checking 'no_clock'.
Information: Checking 'no_input_delay'.
Information: Checking 'partial_input_delay'.
Information: Checking 'ideal_clocks'.
. . .
# The source latency is being calculated for SD_DDR_CLK
pt_shell> report_clock -skew SD_DDR_CLK

# The source latency is applied to the timing report to SD_DQ[14]
pt_shell> report_timing -to sd_DQ[14] -path short
  Startpoint: sdr_clk (clock source 'SDRAM_CLK')
  Endpoint: sd_DQ[14] (output port clocked by SD_DDR_CLK)
  Path Group: SD_DDR_CLK
  Path Type: max
  Max Data Paths Derating Factor  : 1.1000

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock SDRAM_CLK (rise edge)            0.0000     0.0000
  clock source latency                   0.0000     0.0000
  sdr_clk (in)                           0.0000     0.0000 r
  ...
  sd_DQ[14] (inout)                      4.8028 *   4.8028 f
  data arrival time                                 4.8028

  clock SD_DDR_CLK (fall edge)           3.7500     3.7500
  clock network delay (propagated)       2.4948     6.2448
  output external delay                 -0.7500     5.4948
  data required time                                5.4948
  -----------------------------------------------------------
  data required time                                5.4948
  data arrival time                                -4.8028
  -----------------------------------------------------------
  slack (MET)                                       0.6920
```

This page was left blank intentionally.

10 Analysis Type and Back Annotation

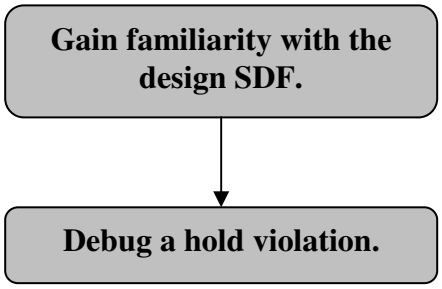
Learning Objectives

After completing this lab, you should be able to:

- Debug a hold violation with your knowledge of the SDF

Lab Duration: 30 minutes

Overview



Relevant Files and Directories

All files for this lab are located in the ***lab10_analysis_types*** directory under your home directory.

lab10_analysis_types/	Current working directory
orca_savesession/	Saved session for ORCA
logs/	
ORCA_EW.log	Warnings from run script
run.log	Log of run
RUN.tcl	Run script for this lab
orca_routed.sdf	SDF for ORCA

Instructions

Task 1. Gain Familiarity with the Design SDF

You will work from the **lab10_analysis_types** Unix directory for this lab. A run script has been executed and the log files are saved under the directory **./logs**.

1. Scroll through **./logs/run.log** to the step where SDF is read and applied to **ORCA** to answer the following questions.

Question 1. Describe the category of nets or cells that are missing SDF?

.....

Question 2. What analysis type was set when reading the design SDF?

.....

Question 3. Was the SDF generated using one or two different operating conditions?

.....

2. Take a look at the actual SDF file in **./logs/orca_routed.sdf** to answer the following question.

Question 4. Are the delay numbers different for min and max (even though the SDF was generated using a single operating condition)?

.....

Question 5. Offer one reason for the min and max numbers to be different?

.....

3. The next step would be to interview the people that generated the SDF and the design constraints.

Question 6. List a few questions you would ask these people?

.....

4. The following bulleted list contains some answers from your interview. These answers will help when debugging the false hold violation in the next task.

- The SDF was generated in one PVT corner.
- The min numbers represent the fast slew propagation and the max numbers represent the slow slew propagation.
- The setup and hold constraints are written for the same, single PVT corner as represented by the SDF.
- The missing SDF is for the top-level boundary nets. The delay for these nets are assumed to be zero because these nets connect the ports to the IO pads. There are no such nets on the physical chip (if you do not count the bond wires).

Task 2. Debug a Hold Violation

1. Invoke PrimeTime and restore the session **./orca_savessession**.

Question 7. How many and what type of violations are in this design?

.....

Question 8. Name the end point with the worst hold violation to an output port?

.....

Question 9. Is CRPR turned on and how would you know?

.....

2. Generate a timing report to **sd_DQ[7]** for hold.

The following alias created in the .synopsys_pt.setup file will generate a timing report for hold in a pop-up window with a scroll bar which makes viewing long reports simpler.

```
pt_shell> vrtm -to sd_DQ[7]
# Add input pins to view net and cell delays separately
pt_shell> vrtm -to sd_DQ[7] -input_pins
```

The alias vrtm uses a command called view that is available on SolvNet, Doc Id 014947.

The alias vrtm will not work if the "wish" executable, the main executable in the Tk package, is not installed and made available in your lab environment.

3. Use the following portion of a man page taken from **report_timing** to help answer the following questions regarding the timing report generated in the last step.

```
pt_shell> vman report_timing
. . . omitted . . .
            Symbol          Annotation
            ------          ----------
            H               Hybrid annotation
            *               SDF back-annotation
            &               RC network back-annotation
            $               RC pi back-annotation
            +               Lumped RC
            <none>          Wire-load model or none
```

Question 10. Which lines in this report can you use to confirm the report is for hold time and is ending at the port **sd_DQ[7]**?

.....

Question 11. Are the boundary net delays along this path set to zero?

.....

4. Set the boundary net delays to zero by executing the following lines:

```
set_annotated_delay 0 -net -to [all_outputs]
set_annotated_delay 0 -net -from [all_inputs]
```

5. Re-execute the STA reports to verify the hold violation is smaller and the boundary net delays are now zero.

Question 12. What symbol is now used beside the boundary net delays in **report_timing**?

.....

6. Quit PrimeTime.

This completes the lab. Return to lecture.

Answers / Solutions

Question 1. Describe the category of nets or cells that are missing SDF?

All the top-level boundary nets are missing SDF.

```
# From ./logs/run.log
                              |           |           |    NOT    |
    Delay type                |    Total  | Annotated | Annotated |
------------------------------+-----------+-----------+-----------+
cell arcs                     |   154424  |   154424  |         0 |
cell arcs (unconnected)       |       68  |       68  |         0 |
internal net arcs             |    52781  |    52781  |         0 |
net arcs from primary inputs  |       54  |        0  |        54 |
net arcs to primary outputs   |       61  |        0  |        61 |
------------------------------+-----------+-----------+-----------+
                              |   207388  |   207273  |       115 |
```

Question 2. What analysis type was set when reading the design SDF?

The design is set to **on_chip_variation** analysis type.

```
# From ./logs/run.log
****************************************
Report : read_sdf /.../design_data/orca_routed.sdf.gz
        -load_delay cell
        -analysis_type on_chip_variation
        -min_type sdf_min
        -max_type sdf_max
Design : ORCA
Version: X-2005.06-SP2
Date   : *
****************************************
```

Question 3. Was the SDF generated using one or two different operating conditions?

The SDF was generated using one operating condition for both min and max.

```
# From ./logs/run.log
        0 error(s)
        Number of annotated cell delay arcs :    364468
        Number of annotated net delay arcs  :     52896
        Number of annotated timing checks   :     41168
        Number of annotated constraints     :      7273
        TEMPERATURE: 125.00 (min)  125.00 (max)
        VOLTAGE    : 1.08 (min)  1.08 (max)
        PROCESS    : 1.200000 (min)  1.200000 (max)
```

Question 4. Are the delay numbers different for min and max (even though the SDF was generated using a single operating condition)?

Yes. Scroll through the first few pages of the SDF file and you will see the min and max numbers for each arc is different.

Question 5. Offer one reason for the min and max numbers to be different?

Slew propagation. The max numbers represent a propagation of the slowest slew. The min numbers represent a propagation of the fastest slew.

Question 6. List a few questions you would ask these people?

What do the min and max SDF numbers represent?

What slew propagation was used?

Why are there missing SDF and what assumptions are being made?

Are the setup and hold design constraints written for the same PVT corner as was assumed for SDF generation?

Question 7. How many and what type of violations are in this design?

```
pt_shell> report_analysis_coverage
Type of Check      Total           Met         Violated          Untested
-------------------------------------------------------------------------
setup               9629      3496 ( 36%)         92 (  1%)       6041 ( 63%)
hold                9629      3588 ( 37%)          0 (  0%)       6041 ( 63%)
recovery            1316      1210 ( 92%)          0 (  0%)        106 (  8%)
removal             1316      1210 ( 92%)          0 (  0%)        106 (  8%)
min_period            20        20 (100%)          0 (  0%)          0 (  0%)
min_pulse_width     7273      5957 ( 82%)          0 (  0%)       1316 ( 18%)
clock_gating_setup    33        33 (100%)          0 (  0%)          0 (  0%)
clock_gating_hold     33        33 (100%)          0 (  0%)          0 (  0%)
out_setup             75        75 (100%)          0 (  0%)          0 (  0%)
out_hold              75        59 ( 79%)         16 ( 21%)          0 (  0%)
-------------------------------------------------------------------------
All Checks         29399     15681 ( 53%)       108 (  0%)      13610 ( 46%)
```

Question 8. Name the end point with the worst hold violation to an output port?

```
pt_shell> !! -status violated -check out_hold
Constrained                Related   Check
Pin                        Pin       Type                        Slack
-------------------------------------------------------------------------
sd_DQ[7]                             out_hold                 -0.5050
sd_DQ[8]                             out_hold                 -0.4950
```

Question 9. Is CRPR turned on and how would you know?

Yes, it is turned on.

```
pt_shell> aa reconvergence
********* Commands **********
Information: No commands matched '*reconvergence*'.
(CMD-040)
********* Variables **********
timing_clock_reconvergence_pessimism = "normal"
timing_remove_clock_reconvergence_pessimism = "true"
```

Question 10. Which lines in this report can you use to confirm the report is for hold time and is ending at the port **sd_DQ[7]**?

The end point of the timing path is the port **sd_DQ[7]** and the "Path Type" is min.

Question 11. Are the boundary net delays set to zero?

No. The boundary nets are missing SDF. The assumption made was these delays are zero. PrimeTime does not make the same assumption. PrimeTime will attempt to calculate the delays for all missing SDF annotations!

Note in the report below, the delay from the input port has an "H" symbol which indicates the delays are not completely SDF (recall that the cell and net delays are reported together). The delay to the output port (the final net delay) has no "*" symbol and the delay is not zero. No symbol indicates the net delay is calculated using a WLM.

```
pt_shell> report_timing -delay min -to sd_DQ[7]

. . .
Point                                                    Incr       Path
 ---------------------------------------------------------------------------
 clock SDRAM_CLK (fall edge)                           3.7500     3.7500
 clock source latency                                  0.0000     3.7500
 sdr_clk (in)                                          0.0000     3.7500 f
 sdr_clk_iopad/CIN (pc3d01)                            0.4931 H   4.2431 f
 I_CLOCK_GEN/sdram_clk (CLOCK_GEN)                     0.0000 *   4.2431 f
 I_CLOCK_GEN/I_PLL_SD/CLK (PLL)                       -1.1267 *   3.1164 f
 I_CLOCK_GEN/invbdkG1B1I1_2/ZN (invbdk)                0.0592 *   3.1756 r
 I_CLOCK_GEN/invbdkG1B2I1/ZN (invbdk)                  0.0456 *   3.2212 f
 I_CLOCK_GEN/U22/Z (mx02d2)                            0.3170 *   3.5382 f
 I_CLOCK_GEN/invbdkG2B1I1_2/ZN (invbdk)                0.0562 *   3.5944 r
 I_CLOCK_GEN/invbdkG2B2I1/ZN (invbdk)                  0.0362 *   3.6306 f
 I_CLOCK_GEN/U18/ZN (invbdk)                           0.0245 *   3.6551 r
 I_CLOCK_GEN/U15/ZN (invbdk)                           0.0183 *   3.6733 f
 I_CLOCK_GEN/o_sdram_clk (CLOCK_GEN)                   0.0000 *   3.6733 f
 I_CLK_SOURCE_SDRAM_CLK/Z (bufbdk)                     0.1632 *   3.8365 f
 bufbdfG5B1I5_1/Z (bufbdf)                             0.1908 *   4.0273 f
 I_ORCA_TOP/buf_sdram_clk_G5B1I5_1ASTHIRNet805 (ORCA_TOP)
                                                       0.0000 *   4.0273 f
 I_ORCA_TOP/buffd7G5B2I15/Z (buffd7)                   0.1870 *   4.2143 f
 I_ORCA_TOP/I_SDRAM_IF/buf_sdram_clk_G5B2I15ASTHIRNet586 (SDRAM_IF)
                                                       0.0000 *   4.2143 f
 I_ORCA_TOP/I_SDRAM_IF/sd_mux_dq_out_7/Z (mx02d4)      0.5165 *   4.7308 f
 I_ORCA_TOP/I_SDRAM_IF/sd_DQ_out[7] (SDRAM_IF)         0.0000 *   4.7308 f
 I_ORCA_TOP/sd_DQ_out[7] (ORCA_TOP)                    0.0000 *   4.7308 f
 sdram_DQ_iopad_7/PAD (pc3b05)                         1.9814 *   6.7122 f
 sd_DQ[7] (inout)                                      0.0478     6.7600 f
 data arrival time                                                6.7600
```

Use the appropriate switch to separate the net from the cell delays and you will see that the boundary nets are not annotated (do not have a * by the delay) and the delays are not zero. They are calculated using wire load models.

```
pt_shell> report_timing -delay min -to sd_DQ[7] -input_pins

. . .
Point                                                    Incr       Path
 ---------------------------------------------------------------------------
 clock SDRAM_CLK (fall edge)                           3.7500     3.7500
 clock source latency                                  0.0000     3.7500
 sdr_clk (in)                                          0.0000     3.7500 f
 sdr_clk_iopad/PAD (pc3d01)                            0.0465     3.7965 f
 sdr_clk_iopad/CIN (pc3d01)                            0.4466 *   4.2431 f
 I_CLOCK_GEN/sdram_clk (CLOCK_GEN)                     0.0000 *   4.2431 f
. . .
```

Question 12. What symbol is now used beside the boundary net delays in **report_timing**?

The delays are now zero and have a "*" symbol indicating they are annotated delays. The hold violations are smaller.

```
pt_shell> set_annotated_delay 0 -net -to [all_outputs]
pt_shell> set_annotated_delay 0 -net -from [all_inputs]
pt_shell> !report
. . .
Point                                                   Incr       Path
--------------------------------------------------------------------------
 clock SDRAM_CLK (fall edge)                          3.7500     3.7500
 clock source latency                                 0.0000     3.7500
 sdr_clk (in)                                         0.0000     3.7500 f
 sdr_clk_iopad/PAD (pc3d01)                           0.0000 *   3.7500 f
 sdr_clk_iopad/CIN (pc3d01)                           0.4466 *   4.1966 f
 I_CLOCK_GEN/sdram_clk (CLOCK_GEN)                    0.0000 *   4.1966 f
. . .
 sdram_DQ_iopad_7/PAD (pc3b05)                        1.9429 *   6.6657 f
 sd_DQ[7] (inout)                                     0.0000 *   6.6657 f
 data arrival time                                               6.6657

 clock SD_DDR_CLK (fall edge)                         3.7500     3.7500
 clock network delay (propagated)                     3.0208     6.7708
 clock reconvergence pessimism                        0.0000     6.7708
 output external delay                                0.1000     6.8708
 data required time                                              6.8708
 --------------------------------------------------------------------------
 data required time                                              6.8708
 data arrival time                                              -6.6657
 --------------------------------------------------------------------------
 slack (VIOLATED)                                               -0.2051
```

11 Additional Checks and Constraints

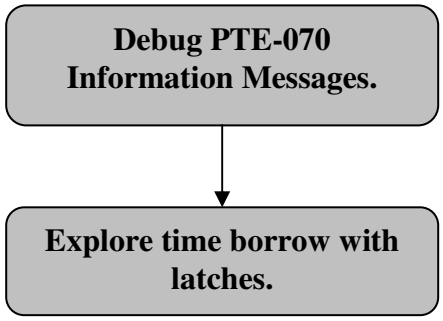
Learning Objectives

After completing this lab, you should be able to:

- Apply user specified annotated delays to explore time borrowing with latches
- Debug PTE-070 messages regarding non-unate cells on the clock path

Lab Duration: 30 minutes

Overview



Relevant Files and Directories

All files for this lab are located in the ***lab11_misc*** directory under your home directory.

`lab11_misc/`	Current working directory
`orca_savesession/`	Saved session for ORCA
`logs/`	Log files from run script

Instructions

Task 1. Debug PTE-070 Information Messages

1. Invoke PrimeTime from the **lab11_misc** Unix directory.

Restore the session saved under **./orca_savesession**.

2. Shown below is the full message regarding a non-unate path on the clock network.

In the next step, you will be asked to generate a timing report through this pin. In order to copy and paste and avoid typos – either find this message in the log file from another terminal window or use the Unix command **grep** from within PrimeTime as shown below.

```
# From ./logs/run.log
Information: A non-unate path in clock network detected.
Propagating both inverting and noninverting senses of clock
'SDRAM_CLK' from pin 'I_ORCA_TOP/I_SDRAM_IF/sd_mux_dq_out_0/Z'.
(PTE-070)
```

```
pt_shell> sh grep PTE-070 logs/run.log
```

Note: The command **sh** (or alternatively **exec**) allows you to execute Unix commands from within the PrimeTime shell.

3. Generate a timing report for setup through the above pin and answer the following questions.

The following alias has been created in the PrimeTime setup file and will generate a timing report in a pop-up window with a scroll bar using the view utility found on SolvNet, Doc Id 014947.

alias vrt view report_timing -nosplit

```
pt_shell> vrt -through <through pin>
```

Question 1. Which lines in the timing report did you use to validate it is for setup and the timing path start point is the source for the clock **SDRAM_CLK**?

.....

Question 2. How does this timing report confirm that the pin in the warning above is on a data path (i.e. a clock source being used and constrained as a data path) and not on a clock path?

.....

Question 3. Which sense is propagated through the above pin (i.e. positive unate or negative unate)? Look for a small arrow in the timing report which will locate the specific pin of interest.

.....

4. Generate at least one additional timing report to show the use of a positive unate timing arc through the pin of interest.

Question 4. Which lines in the timing report did you use to validate it is for setup, the timing path start point is the source for the clock **SDRAM_CLK** and that the timing arc is positive unate for the pin of interest?

.....

Question 5. Explain why this warning can be ignored (and suppressed) for these timing paths?

.....

5. Do not quit PrimeTime.

Task 2. Explore Time Borrow and Latches

There is only one latch in this design.

1. Use the following commands to find it:

Take advantage of command and option completion with the tab key.

```
pt_shell> all_registers -level_sensitive
pt_shell> !! -clock_pin
pt_shell> all_registers -level_sensitive -data_pins
```

Question 6. What is the name of the clock pin for this latch?

.....

Question 7. What are the names of the three data pins?

.....

2. Generate a timing report starting at the latch for setup time (be specific by using the clock pin as the start point and not just the cell name!).

This lab will refer to this timing report as "path segment #2".

Question 8. Functionally, what does this latch do in the **ORCA** design?

.....

Question 9. Describe how you know this latch is <u>not</u> experiencing time borrow from the previous stage?

.....

3. Generate a timing report for the previous stage (this lab will refer to this timing report as "path segment #1").

Use the **D** input pin of the latch as the end point of this timing path.

Question 10. How much more time can path segment #1 take before it would start borrowing time from path segment #2?

.....

4. Force path segment #1 to borrow time from path segment #2 by annotating a net delay of 4ns as shown below:

```
# Use cut and paste to avoid typos on the pin name
pt_shell> set_annotated_delay -net 4 \
          -to I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D
```

5. Generate the timing report for path segment #1 again (take advantage of the up and down arrows to scroll through the history event list).

Question 11. The annotated net delay will show up as 4.4ns (not 4ns); can you explain why the delay is scaled by 1.10?

.....

Question 12. How much time is path segment #1 borrowing from path segment #2?

.....

Question 13. What is the slack for path segment #1?

.....

6. Re-generate the timing report for path segment #2. Before you can do that, you have to perform a full timing update!

```
pt_shell> update_timing -full
```

Note: The start point of the timing path will now be the D pin of the latch (not the clock pin as used before) because you are interested in reporting the timing path that includes time borrow.

```
pt_shell> report_timing -from \
            I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D
```

Question 14. The time given to path segment #1 will not match the time borrowed by path segment #1 from question 12; can you explain why?

.....

7. Generate another timing report for path segment #2, this time forcing a rising data transition at the start point.

Question 15. Does the time given to path segment #1 now match your expectations?

.....

8. Quit PrimeTime.

Congratulations, this completes the labs for this workshop.

Answers / Solutions

Question 1. Which lines in the timing report did you use to validate it is for setup and the timing path start point is the source for the clock **SDRAM_CLK**?

```
pt_shell> report_timing -through I_ORCA_TOP/I_SDRAM_IF/sd_mux_dq_out_0/Z
  Startpoint: sdr_clk (clock source 'SDRAM_CLK')
  Endpoint: sd_DQ[0] (output port clocked by SD_DDR_CLK)
  Path Group: SD_DDR_CLK
  Path Type: max
```

Question 2. How does this timing report confirm that the pin in the warning above is on a data path (i.e. a clock source being used and constrained as a data path) and not on a clock path?

If no report was generated ("path is unconstrained"), this pin is on a clock path. Because a timing report was generated, this pin is on a data path.

Question 3. Which sense is propagated through the above pin (i.e. positive unate or negative unate)? Look for a small arrow in the timing report which will locate the specific pin of interest.

A negative unate timing arc is reported through this pin.

```
# Shown are the relevant lines in the data path
I_ORCA_TOP/I_SDRAM_IF/buffd7G5B2I36/Z (buffd7)        0.2080 *   0.9622 r
I_ORCA_TOP/I_SDRAM_IF/sd_mux_dq_out_0/Z (mx02d4) <-   0.8668 *   1.8290 f
```

Question 4. Which lines in the timing report did you use to validate it is for setup, the timing path start point is the source for the clock **SDRAM_CLK** and that the timing arc is positive unate for the pin of interest?

```
pt_shell> pt_shell> report_timing \
               -rise_through    I_ORCA_TOP/I_SDRAM_IF/sd_mux_dq_out_0/Z
Startpoint: sdr_clk (clock source 'SDRAM_CLK')
Endpoint: sd_DQ[0] (output port clocked by SD_DDR_CLK)
Path Group: SD_DDR_CLK
Path Type: max

…
I_ORCA_TOP/I_SDRAM_IF/buffd7G5B2I36/Z (buffd7)        0.2080 *   0.9622 r
I_ORCA_TOP/I_SDRAM_IF/sd_mux_dq_out_0/Z (mx02d4) <-   0.7241 *   1.6863 r
```

Question 5. Explain why this message can be ignored for these timing paths?

The message indicates that both senses of the clock will be used when propagating the clock through this mux – this is the default behavior as of 2006.06 However, because the clock is being used as data, PrimeTime actually propagates both senses (both positive and negative unate), even in older versions of PrimeTime. This is what is desired and therefore this information message can be ignored.

Question 6. What is the name of the clock pin for this latch?

```
pt_shell> all_registers -level_sensitive -clock_pins
{"I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/EN"}
```

Question 7. What are the names of the three data pins?

```
pt_shell> all_registers -level_sensitive -data_pins
{"I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D",
"I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/SC",
"I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/SD"}


# For step 2, generate a timing report for path segment 2
pt_shell> report_timing -from I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/EN
```

Question 8. Functionally, what does this latch do in the **ORCA** design?

This latch is generating a clock gating signal to turn on and off the clock **SYS_CLK**.

Question 9. Describe how you know this latch is <u>not</u> experiencing time borrow from the previous stage?

If this latch was experiencing time borrow, there would be a line in the report stating the amount of time given to the start point (i.e. to the previous stage). This line is not present in this timing report.

```
# For step 3, generate a timing report for path segment 1
pt_shell> report_timing -to I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D
```

Question 10. How much more time can path segment #1 take before it would have to start borrowing time from path segment #2?

It can take 2.8ns more before it would start borrowing time from path segment #2 (equivalent to the positive slack).

Question 11. The annotated net delay will show up as 4.4ns (not 4ns). Can you explain why the delay is scaled by 1.10?

There is a derating factor of 1.1 applied to the data paths of the **ORCA** design. See this at the top of the timing report, or generate specific information by using the switch **–derate** when generating timing reports.

Note that the annotated delay overwrites the delay from SDF – it is not added to the SDF delay. The original net delay was 0.0004ns.

Question 12. How much time is path segment #1 borrowing from path segment #2?

Path segment #1 is borrowing 1.6072ns from path segment #2. This is noted in the data required time section of the timing report.

Question 13. What is the slack for path segment #1?

The slack is zero. PrimeTime borrows exactly as much as is needed to make the slack equal zero.

Question 14. The time given to path segment #1 will not match the time borrowed by path segment #1 from question 12. Can you explain why?

```
# Because of time borrow, the start point is now the D pin of the latch
pt_shell> report_timing -from I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D
```

The timing report for path segment #1 has a rising data transition at the timing path end point. The timing report generated for path segment #2 uses a falling data transition at the timing path start point. The time borrowed and the time given to the start point will not match because of this.

Question 15. Does the time given to path segment #1 now match your expectations?

Yes – the time given to start point in the timing report for path segment #2 will match the time borrowed in the timing report for path segment #1.

```
pt_shell> report_timing \
              -rise_from I_ORCA_TOP/I_BLENDER/latched_clk_en_reg/D
# Or take advantage of short cuts
pt_shell> ^from^rise_from
```

This page was intentionally left blank.