

3

Performing Timing Analysis

This chapter steps you through performing a basic timing analysis. When you finish you will know how to

- Set up the analysis environment.
- Set up the basic timing assertions and clock gating checks.
- Check the design integrity.
- Run the timing analysis.
- Characterize the boundary timing of a cell.
- Correct timing violations.

Setting Up the Analysis Environment

Setting up the analysis environment requires that you perform the following tasks:

- Set the link path and link the design.
- Set operating conditions and wire load models.
- Set basic timing assertions.
- Check the assertions and structure of the design.

Reading In and Linking the AM2910 Design

To read in and link the AM2910 design, enter

```
pt_shell> set link_path "$link_path STACK_lib.db Y_lib.db"
pt_shell> link_design AM2910
Unlinking design 2910
Information: All timing information (backannotation, exceptions, etc.)
is being removed form design 'UPC'. These annotations must be restored prior to
relinking this design. (LNK-024)
Linking design AM2910 ...
```

Linking AM2910 causes any other currently linked design to be unlinked. There can only be one linked design in memory. When a design is unlinked, all timing information is removed and a warning appears. (This is different from Design Compiler.) If you need to save the annotations, use the `write_script` command before you link a new design. If you need to relink the design later, run the assertion script.

Setting Up Operating Conditions and Wire Load Models

Use the PrimeTime min-max analysis feature when you set the operating conditions and wire load models. When you use the min-max analysis feature, PrimeTime uses

- Maximum operating conditions and wire load model when it generates setup timing reports.
- Minimum operating conditions and wire load model when it generates hold timing reports.

Set up the operating conditions and wire load models. Enter

```
pt_shell> set_operating_conditions -library pt_lib -min BCCOM -max WCCOM
pt_shell> set_wire_load_mode top
pt_shell> set_wire_load_model -library pt_lib -name 05x05 -min
pt_shell> set_wire_load_model -library pt_lib -name 20x20 -max
```

If the operating conditions are in two different libraries, use the `read_min_max_lib` command to read the two different libraries and create a “merged” library in memory. You cannot write the merged library to disk.

Setting Up the Basic Assertions

To set up the basic assertions, start with the clock information. Use min-max values for the clock latency (insertion delay). The clock uncertainty is the expected skew. Enter

```
pt_shell> create_clock -period 30 [get_ports CLOCK]
pt_shell> set_clock [get_clock CLOCK]
pt_shell> set_clock_uncertainty 0.5 $clock
pt_shell> set_clock_latency -min 3.5 $clock
pt_shell> set_clock_latency -max 5.5 $clock
```

```
pt_shell> set_clock_transition -min 0.25 $clock
pt_shell> set_clock_transition -max 0.3 $clock
```

For designs with a fully back-annotated clock network, you can derive the uncertainty, skew, and transition automatically using this command:

```
set_propagated_clock clock_object_list
```

Setting Up Clock Gating Checks

Specify the clock gating setup and hold values and a minimum pulse width. Enter

```
pt_shell> set_clock_gating_check -setup 0.5 -hold 0.1 $clock
pt_shell> set_min_pulse_width 2.0 $clock
```

If your design is back-annotated, PrimeTime uses the setup and hold values and the width specification from the Standard Delay Format (SDF).

Summarizing the Timing

When you are satisfied that the design's environment and timing assertions are correctly specified, you can summarize the design's statistics. Enter

```
pt_shell> report_design
pt_shell> report_reference
```

The `report_design` command shows the min-max operating condition and wire load models.

Report : design
 Design : AM2910
 Version: 1998.02
 Date : Wed Jan 14 09:09:47 1998

Design Attribute	Value
------------------	-------

Operating Conditions:

analysis_type bc_wc	
perating_condition_min_name	BCCOM
process_min	1.5
temperature_min	70
voltage_min	4.75
tree_type_min	worst_case

operating_condition_max_name	WCCOM
process_max	1.5
temperature_max	125
voltage_max	4.5
tree_type_max	worst_case

Wire Load:

wire_load_mode	(use report_wire_load for more information)
wire_load_model_max	top
wire_load_model_library_max	20x20
wire_load_selection_type_max	pt_lib
wire_load_model_min	user-specified
wire_load_model_library_min	05x05
wire_load_selection_type_min	pt_lib
wire_load_selection_group_max	user-specified
wire_load_selection_group_min	pt_lib
wire_load_min_block_size	pt_lib
	0

Design Rules:

max_capacitance	--
min_capacitance	--
max_fanout	--
min_fanout	--
max_transition	--
min_transition	--
max_area	--

Timing Ranges:

fastest_factor	--
slowest_factor	--

The `report_reference` command shows each block and its area. More important, it identifies the extracted, Stamp, and quick timing models. The other blocks are gate-level netlists. Because area cannot be specified in a quick timing model, it appears as zero.

```
*****
Report : reference
Design : AM2910
Version: 1997.08
Date   : Fri May 23 09:09:47 1997
*****
```

Attributes:

```
  b - black-box (unknown)
  h - hierarchical
  n - noncombinational
  u - contains unmapped logic
  A - abstracted timing model
  B - extracted timing model
  S - Stamp timing model
  Q - Quick timing model (QTM)
```

Reference	Library	Unit Area	Count	Total Area	Attributes
CONTROL		86.00	1	86.00	h
REGCNT		230.00	1	230.00	h
STACK		0.00	1	0.00	h,Q
UPC		175.00	1	175.00	h
Y		500.00	1	500.00	h,S
Total 5 references				991.00	

Checking the Assertions and the Structure of the Design

After you specify the necessary timing assertions, you can begin analysis. A key requirement is to always run the `check_timing` command before you perform timing analysis. The `check_timing` command shows possible timing problems for the design. Enter

```
pt_shell> check_timing
```

The `check_timing` command checks the assertions and structure of the design. By default, summary messages appear for each type of check having a violation. If you see errors or warnings, either eliminate them or know why they occur before you continue analysis. Unaccounted for errors or warnings can lead to an incomplete or inaccurate timing analysis.

Get more information about the errors and warnings by running `check_timing` with options. To see the `check_timing` options, enter

```
pt_shell> check_timing -help
```

Running Timing Analysis

Running the timing analysis requires that you perform the following tasks:

- Set port delays.
- Save assertions.
- Run basic analysis.

Setting Port Delays and Checking the Timing

Set input and output delays on ports relative to a clock, which helps provide a completely constrained design.

In addition to setting the input and output delays on ports relative to a clock, set a driving cell on the inputs and a capacitive load on the outputs. Enter

```
pt_shell> set_input_delay 0.0 [all_inputs] -clock $clock
```

```
pt_shell> set_output_delay 2.0 [get_port INTERRUPT_DRIVER_ENABLE] -clock $clock
pt_shell> set_output_delay 1.25 [get_port MAPPING_ROM_ENABLE] -clock $clock
pt_shell> set_output_delay 0.5 [get_port OVERFLOW] -clock $clock
pt_shell> set_output_delay 1.0 [get_port PIPELINE_ENABLE] -clock $clock
pt_shell> set_output_delay 1.0 [get_port Y_OUTPUT] -clock $clock
pt_shell> set_driving_cell -lib_cell IV -library pt_lib [all_inputs]
pt_shell> set_capacitance 0.5 [all_outputs]
pt_shell> check_timing
```

The design is completely constrained.

Saving Assertions

After you fully annotate the design, set the assertions, and so forth, use the `write_script` command to save the complete timing script to a file. Saving the timing script to a file ensures that the timing environment is completely duplicated in subsequent runs. Whenever you make changes to the timing assertions or environment, be sure to save the timing environment using `write_script`. The `write_script` command saves the following information to a command file:

Clocks	Names, waveforms, latency, and uncertainty
Exceptions	False and multicycle paths, minimum and maximum delays, and path groups
Delays	Input and output delays, all delay annotations, and timing checks
Net and port attributes	Capacitance, resistance, and fanout
Design environment	Wire load model, operation condition, drive, driving cell, and transition
Design rules	Minimum and maximum capacitance, minimum and maximum fanout, and minimum and maximum transition

The `write_script` command can write scripts in Design Compiler format (`dcsh`) or PrimeTime format (`ptsh`).

You cannot write an annotated design `.db` file from PrimeTime because PrimeTime writes `.db` files only for timing models. Use scripts as the primary method for passing data to Design Compiler.

1. Save the timing environment and assertions for the AM2910 top-level design in both Design Compiler and PrimeTime formats.
Enter

```
pt_shell> write_script -format dcsh -output AM2910.dcsh
pt_shell> write_script -format ptsh -output AM2910.pt
```

2. Review these scripts to understand the timing and design assertions that PrimeTime and Design Compiler use.

Running Basic Analysis

When you are satisfied that you correctly specified the design's environment and timing assertions and you have run the `check_timing` command, summarize the timing information by using the `report_constraint` command.

The `report_constraint` command reports the slack summary for the following constraints on the entire design:

- Minimum and maximum delay (representing setup and hold checks)

- Minimum clock pulse width

- Minimum period

- Recovery and removal on registers (preset and clear to clock setup and hold)

Clock gating setup and hold

Minimum and maximum capacitance (design rule check)

Minimum and maximum transition (design rule check)

Minimum and maximum fanout (design rule check)

- Use the `-all_violators` option to report the worst slack to each violating endpoint for each constraint.
- Use the `-verbose` option to get the most detail on the constraint calculation.

Report the slack.

1. Get constraint reports for the AM2910 design. Enter

```
pt_shell> report_constraint
```

2. Check the report for timing violations and constraints violations for the AM2910 design.

3. Get more information about the violations. Enter

```
pt_shell> report_constraint -all_violators
```

4. Examine the report.

How many violating endpoints exist?

Reporting the Path-Based Timing

The `report_timing` command provides path-based timing reports. The `report_timing` command with no options reports the longest maximum path in the design for each path group. Typically, use this to measure paths for setup timing.

The `report_timing` command is very flexible and has many options. Use the man pages and the `report_timing -help` command to learn the details of this important command.

To find the shortest minimum path in the design (measuring hold paths), use the `-delay min` option.

1. Get a complete path timing report for the longest violating path in the AM2910 design. Enter

```
pt_shell> report_timing
```

2. Examine the report.

Setting Timing Exceptions

Timing exceptions occur when exceptions to the default single-cycle, worst case analysis exist. These exceptions include

False paths

Multicycle paths

User-defined minimum and maximum delay constraints

Disabled timing arcs

PrimeTime supports the `-through` option to the timing exceptions commands. You must specify timing exceptions correctly or they are not accepted. False and multicycle paths must specify a complete, valid path with proper startpoints and endpoints.

- Startpoints are primary input ports, clocks, pins, or cells.
- Endpoints are primary output ports, clocks, pins, or cells.

Because the validity of timing exceptions are not checked until PrimeTime performs a full timing update, be sure they are correct by running the `report_exceptions` command.

Always use `report_exceptions -ignored` to see whether any exceptions were ignored because they were not valid. Correct ignored exceptions because the exceptions have no effect if you do not correct ignored exceptions.

1. Set up timing exceptions for the AM2910 design. Set a multicycle path of two clocks. To keep the same hold relationship (no multicycle for the hold), specify 1 for the hold multiplier and specify 2 for the setup multiplier. Enter

```
pt_shell> set_false_path -from U3/OUTPUT_reg[*]/CP -to U2/OUTPUT_reg[*]/D
pt_shell> set_multicycle_path -setup 2 -from INSTRUCTION[*]
                    -to U2/OUTPUT_reg[*]
pt_shell> set_multicycle_path -hold 1 -from INSTRUCTION[*]
                    -to U2/OUTPUT_reg[*]
pt_shell> update_timing
pt_shell> report_exceptions
pt_shell> report_exceptions -ignored
```

2. Examine the reports.

The ignored exceptions report shows nothing because there are no ignored exceptions, indicating that you specified the exceptions correctly.

Evaluating Timing Exceptions Results

After you specify the exceptions, evaluate the timing exceptions results.

1. Get another constraint report and evaluate the violations. Enter

```
pt_shell> report_constraint -all_violators
```

2. Examine the constraint report.
3. Determine whether setting exceptions causes fewer violations in the design than before. What is the worst slack in this design?
Enter

```
pt_shell> report_timing
```

4. Examine the detailed timing report. What suggestions can you make for fixing this violation?
5. Look at other violating paths. Choose an endpoint from the constraint report and enter

```
pt_shell> report_timing -to endpoint
```

6. Examine the report.

Characterizing the Boundary Timing of a Subdesign

Context characterization derives the timing context of a subdesign based on its context (environment) in the parent design. The two main applications for this information are

Within PrimeTime

The context allows you to perform timing analysis hierarchically, and PrimeTime observes chip-level timing constraints.

Within Design Compiler

The context sets the timing constraints of a subdesign during synthesis or logic optimization.

A key function of PrimeTime is its ability to pass these assertions as constraints to Design Compiler through context characterization. This is the primary link from chip-level timing analysis to module-level optimization in Design Compiler, providing tighter integration than a stand-alone analysis-only tool can give.

After you perform context characterization, direct PrimeTime to write a script containing the timing assertions of the subdesign (or module). PrimeTime can write this script in either PrimeTime or Design Compiler format.

The context includes

- Input arrival and output required delays
- Clock information (period, waveform, uncertainty, latency, and so forth)
- Timing exceptions
- Input port drive and output port load
- Wire-load models
- Design rules (maximum capacitance, transition, and fanout)
- Constant logic values propagated to inputs
- Annotated delays and parasitics (RC information)

Keep the following points in mind when you characterize the timing:

- Characterizing does not derive and does not generate timing budgets for the subdesign. See the Design Compiler documentation for information and methodologies for budgeting delays.
- Characterizing does not work in min-max mode. Set the single correct operating condition for optimization before you characterize.

The steps to set synthesis or optimization constraints for Design Compiler are

1. Read the top-level design into PrimeTime.
2. Identify the subdesigns that require optimization.
3. Characterize the context for each subdesign.
4. Generate a Design Compiler script for each subdesign.
5. Read the subdesigns into Design Compiler.
6. Include the scripts created in Step 4.
7. Perform module-level optimization.

Read only the subdesigns that require optimization into Design Compiler. Doing this causes Design Compiler to work most efficiently. After you optimize the modules using Design Compiler, read them back into PrimeTime to provide a new chip-level timing analysis.

Examining the timing report shows that blocks U3 (REGCNT) and U2 (UPC) might benefit from optimization, possibly eliminating some violations.

1. Set the operating conditions to the worst case because you want to correct setup violations. Enter

```
pt_shell> set_operating_conditions -library pt_lib WCCOM
pt_shell> characterize_context {U2 U3}
pt_shell> write_context U2 -output UPC.char.dcsch -format dcsch
pt_shell> write_context U3 -output REGCNT.char.dcsch -format dcsch
pt_shell> write_script -format ptsh -output AM2910.new.pt
```

2. Review the two Design Compiler-format scripts created by the PrimeTime write_context command.

Correcting a Timing Violation

Review the provided Design Compiler script optimize.dcsch. This script reads in the REGCNT and UPC .db files, links them, applies the scripts from PrimeTime, and compiles and writes the new optimized .db files (REGCNT.opt.db and UPC.opt.db) to disk.

1. Run Design Compiler from a new UNIX shell window, leaving the PrimeTime session running. Optimize the designs using the original chip-level timing constraints as the timing environment. The optimize.dcsch script provides Design Compiler with correct, accurate timing constraints without requiring the entire chip to be loaded. Enter

```
% dc_shell
dc_shell> include optimize.dcsch
...
dc_shell> quit
```

2. Return to the PrimeTime shell, read the new, optimized .db files in and swap the old designs with the new ones. Enter

```
pt_shell> read_db {REGCNT.opt.db UPC.opt.db}
pt_shell> current_design AM2910
pt_shell> swap_cell U3 {REGCNT.opt.db:REGCNT}
pt_shell> swap_cell U2 {UPC.opt.db:UPC}
pt_shell> source AM2910.new.pt
```

3. Analyze the chip-level timing with the new designs. Run a timing check each time you make changes. Enter

```
pt_shell> check_timing
pt_shell> report_constraint -all_violators
pt_shell> report_constraint -all_violators -verbose
```

4. Examine the reports.

Do the optimized designs contain fewer violations than before?
How many violations remain?