Specifying Timing Exceptions

Learning Objectives

This lab is intended to give you a better understanding of how static timing analysis works and how timing exceptions are applied properly.

After completing this lab, you should be able to:

- Fully constrain and analyze a design using as minimum a number of timing exceptions (set_max_delay, set_multicycle_path, set_false_path) as possible.
Lab 6

Getting Started

The Unix directory for this entire lab is ./Lab6_Exceptions.

Answers and suggestions are located at the end of this lab.

If you need help...

Use the lecture material, SOLD or the Quick Reference Guide.

```
pt_shell> help report_tim*
pt_shell> printvar *case_analys*
pt_shell> create_generated_clock -help
pt_shell> man set_propagated_clock
```

The history commands!

```
pt_shell> history
pt_shell> !26
```

You do not need to type the entire command or option names!

```
pt_shell> report_timing -cap -tran
pt_shell> list_attr -app -class clock
```
Background

The design you will be working with contains combinatorial snake paths as shown in the figure below (resets not shown).

Initial Analysis

There are many different ways to define the constraints for this design; you will work through several scenarios to solve this problem.

Task 1. Read Mapped Design

1. Change your current working directory to ./Lab6_Exceptions and invoke pt_shell.

2. Read the compiled netlist into memory.

   read_db test.db
   current_design test
   link

3. Check the design’s timing report.

   report_timing

Question 1. Does anything appear unusual? ............................................................
**Task 2. Constrain the Design for Timing**

There are no constraints set on the design; you need to apply a constraint script.

The timing specifications for the design are as follows:

- Clock period: 200 MHz
- Input delays on data inputs (adr_i[*] and coeff[*]): 2 ns relative to system clock.
- Input delay on sel_combo: 0.35 ns relative to system clock
- Output delay: 4 ns relative to system clock

1. A constraint script with the above specification has been created for your use. Apply the constraint file `my_constraints.tcl` and check the constraints.

```
source -echo -verbose my_constraints.tcl
report_clock
check_timing -verbose
report_timing
```

**Question 2.** How large is the worst negative slack?..........................

**Question 3.** What path exhibits this violation?..............................
Task 3. Using a Max-Delay Constraint

The violation you just discovered is very large. There are two possibilities: either the violations are indeed large or the constraints specified were incorrect or insufficient. *The propagation delay through the combinational path should be 10 ns or less.*

This means that the constraints have to be adjusted. Use the knowledge gained during the lecture.

1. Apply a max delay constraint on the combinational path.

   ```
   set_max_delay 10 -from [get_ports "coeff* adr_i*"] -to [all_outputs]
   ```

   Since `set_max_delay` is an exception, you should always make sure that the exception was actually applied correctly and that no exceptions were ignored.

   ```
   check_timing
   report_exceptions -ignored
   ```

2. Generate another timing report.

   ```
   report_timing
   ```

*Question 4.* Does the report seem more reasonable? ................................................

*Question 5.* Which path is the worst violator now? ....................................................
Task 4. Constrain Multicycle Path

The path to the `mul_result_reg` actually takes 3 clock cycles instead of 1, (the default). Here again you can specify an exception.

1. Apply a multicycle path of 3 cycles to all paths that lead to `mul_result_reg`.
   
   ```
   set_multicycle_path 3 -setup -to mul_result_reg*
   ```

   Verify the exceptions were applied correctly:
   
   ```
   check_timing
   report_exceptions -ignored
   ```

2. Add the above multicycle exception to your constraint file `my_constraints.tcl`.

3. Check the timing again:
   
   ```
   report_timing
   ```

**Question 6.** Are there any timing violations in the report? .................................

**What just happened?**

*It seems that you are again looking at the combinatorial input-output path that you were hoping to address earlier. Because the register-register path had a larger violation, it replaced the combinatorial path with a worst negative slack of 5.65. After applying the multicycle constraint for the register-register path, you are back to the combinatorial path with a WNS of 5.35.*

**Explanation for timing report result:**

PrimeTime considers the input and output delays that were previously specified on the ports as part of the max-delay constraint; you can see that in the timing report. The max-delay constraint number for the internal logic was reduced by the amount of external delay at the inputs and outputs.
What do you learn from this?

You generally do not want to mix input/output delay constraints (relative to a clock) with asynchronous max-delay constraints for the same endpoints!

Conclusion: Since max-delay and input/output delays are not compatible, you have to come up with other solutions.

Task 5. Use Virtual Clocks

A max_delay constraint did not seem to solve the problem. Consider using a multicycle path for the combinatorial path, this action requires that you pay attention to the input and output delays and not mix them up.

Try a completely different approach: virtual clocks.

1. Remove the max_delay constraint you applied earlier.

   ```
   reset_path -from [get_ports “coeff* adr_i*”] \ 
   -to [all_outputs]
   ```

2. All that is really needed, is to apply a constraint to the combinatorial logic that is independent of the constraints that apply to the sequential logic paths. It would be nice to be able to constrain the snake path with regards to a different clock; this way multiple constraints can be overlayed without interfering with each other. The following schematic tries to illustrate the point:
The diagram shows the design is now being clocked by two clocks, \( \text{clk} \) and \( \text{vclk} \). Constrain the sequential path using \( \text{clk} \), and the combinatorial path using \( \text{vclk} \). The combinatorial path must have a delay of 10 ns or less. \( \text{vclk} \) will have a period of 10 ns and the input and output delays will be zero.

3. Perform the following commands in \text{pt\_shell} and add them to your constraint script \text{my\_constraints.tcl}:

\begin{verbatim}
create_clock -name vclk -period 10
set_input_delay 0 -clock vclk -add_delay \[get_ports “coeff* adr_i*”]
set_output_delay 0 -clock vclk -add_delay [all_outputs]
\end{verbatim}

4. Make sure that the constraints are complete and analyze for timing:

\begin{verbatim}
check_timing
report_timing
\end{verbatim}

\textbf{Question 7.} What is the WNS now?.................................................................

It seems that nothing really changed. Concentrate on the second timing report first, the report for the path group “\( \text{vclk} \)”. It shows that the launch clock is “\( \text{clk} \)” and the capture clock is “\( \text{vclk} \)”. This was not really planned, the idea was to have no overlap between the two constraints.

5. To solve the above problem you need to mask out the paths from \( \text{clk} \) to \( \text{vclk} \) and from \( \text{vclk} \) to \( \text{clk} \):

\begin{verbatim}
set_false_path -from clk -to vclk
set_false_path -from vclk -to clk
\end{verbatim}


That takes care of the path from \( \text{clk} \) to \( \text{vclk} \). Unfortunately you will notice that the original combinatorial constraint including the input and output delays are still being shown. The constraint with regard to \( \text{clk} \) is still constraining the snake path.

\textbf{Question 8.} What would happen if you masked out that path using “set\_false\_path -from clk -to clk”?.................................
Task 6. Refine Virtual Clock Definitions

The constraint problem is nearly solved. You still need to completely isolate the sequential from the combinatorial paths. After specifying the required input and output delays you will mask out the non-relevant paths. The following schematic illustrates this point.

There are 3 clocks overall. Two clocks will be used to constrain the input and output ports, for the combinatorial and the sequential I/O paths. The third clock will be used for the internal register-register paths.

1. Indicate in the following table what clock combinations need to be marked as a false path:

<table>
<thead>
<tr>
<th>Launch Clock</th>
<th>Capture Clock</th>
<th>True / False</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>clk</td>
<td></td>
</tr>
<tr>
<td>clk</td>
<td>vclk_com</td>
<td></td>
</tr>
<tr>
<td>clk</td>
<td>vclk_seq</td>
<td></td>
</tr>
<tr>
<td>vclk_com</td>
<td>clk</td>
<td></td>
</tr>
<tr>
<td>vclk_com</td>
<td>vclk_com</td>
<td></td>
</tr>
<tr>
<td>vclk_com</td>
<td>vclk_seq</td>
<td></td>
</tr>
<tr>
<td>vclk_seq</td>
<td>clk</td>
<td></td>
</tr>
<tr>
<td>vclk_seq</td>
<td>vclk_com</td>
<td></td>
</tr>
<tr>
<td>vclk_seq</td>
<td>vclk_seq</td>
<td></td>
</tr>
</tbody>
</table>

See the Answer section for a solution.
If you are eager to see a solution and cannot wait, follow the directions under (a). If you want to complete your own script and add all the needed commands, follow (b).

**a.** Reset the design and apply the solution script:

```tcl
reset_design
source solution_a.tcl
```

**b.** Open your constraints file `my_constraints.tcl` and add the missing commands. You need to define the two virtual clocks `vclk_com` and `vclk_seq`. You have to constrain the inputs and outputs with regard to the correct clock. Finally, you have to apply the `set_false_path` exceptions according to the table from above.

*See the Answer section for a solution script.*

---

2. Generate a timing report.

```tcl
report_timing
```

**Question 9.** Are the paths constrained correctly now? And do all paths meet setup timing?

---

3. Check for all violations.

```tcl
report_constraint -all
```

**Question 10.** Where did these hold violations suddenly come from?
Task 7. Back to the Multicycle Path

The violations shown in the timing report are all rooted in the multicycle path you applied to the design earlier. By setting a multicycle path for setup at 3 cycles, the hold check capture edge was moved as well.

1. Perform timing analysis for hold.

   ```
   report_timing -delay min
   ```

   **Question 11.** What edge is the hold check performed at for “clk”? .......................

2. Reset the hold capture edge to time 0.

   ```
   set_multicycle_path 2 -hold -to mul_result_reg*
   ```

   **Question 12.** Are all violations gone now? ............................................................
Answers For Lab 6

Task 1. Read Mapped Design

Q 1. Does anything appear unusual?

*The design is not constrained for timing. The timing report only shows the longest path, but does not show any required arrival time information. Reading a db does not necessarily mean that you have a constrained design.*

Task 2. Constrain the Design for Timing

Q 2. How large is the worst negative slack?

10.35 ns.

Q 3. What path exhibits this violation?

*The path with the largest violation is the snake path from the input ports to the output ports.*

Task 3. Using a Max-Delay Constraint

Q 4. Does the report seem more reasonable?

Yes. *The violation from earlier has disappeared.*

Q 5. Which path is the worst violator now?

*The worst violator now is an internal register-register path ending in mul_result_reg*. The worst negative slack is 5.65 ns.

Task 4. Constrain Multicycle Path

Q 6. Are there any timing violations in the report?

Yes, *the max_delay constraint did not do what you expected.*
Task 5. Use Virtual Clocks

Q 7. What is the WNS now?

10.35 ns.

Worst negative slack for the vclk path group is 6.35.

Q 8. What would happen if you masked out that path using “set_false_path –from clk –to clk”?

This is not a good idea. It would cause all the internal register-register paths to disappear as well.

Task 6. Refine Virtual Clock Definitions

Solution for step 1:

<table>
<thead>
<tr>
<th>Launch Clock</th>
<th>Capture Clock</th>
<th>True / False</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>clk</td>
<td>T</td>
</tr>
<tr>
<td>clk</td>
<td>vclk_com</td>
<td>F</td>
</tr>
<tr>
<td>clk</td>
<td>vclk_seq</td>
<td>T</td>
</tr>
<tr>
<td>Vclk_com</td>
<td>clk</td>
<td>F</td>
</tr>
<tr>
<td>Vclk_com</td>
<td>vclk_com</td>
<td>T</td>
</tr>
<tr>
<td>Vclk_com</td>
<td>vclk_seq</td>
<td>F</td>
</tr>
<tr>
<td>Vclk_seq</td>
<td>clk</td>
<td>T</td>
</tr>
<tr>
<td>Vclk_seq</td>
<td>vclk_com</td>
<td>F</td>
</tr>
<tr>
<td>Vclk_seq</td>
<td>vclk_seq</td>
<td>F</td>
</tr>
</tbody>
</table>
Solution for step 2a:

```bash
set clk_period 5
set combo_delay 10

# Real existing clock
create_clock -name clk -period $clk_period "[get_ports clk]
# Define virtual dummy clock for regular in/out Constraining
create_clock -name vclk_seq -period $clk_period
# Define virtual dummy clock for combo-feedthru constraining
create_clock -name vclk_com -period $combo_delay

# Constraining for Combo-Feedthru I/O-Delay
set_input_delay 0 -clock vclk_com "[get_ports "adr_i* coeff*"]
set_output_delay 0 -clock vclk_com [get_ports dout*]

# These are the remaining I/O delays relevant for the paths that
# start at an input port and end at internal registers and paths
# that start at internal registers and end at output ports!
set_input_delay 2 -clock vclk_seq -add_delay "[remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 4 -clock vclk_seq -add_delay "[all_outputs]

# Constrain the reg-reg multicycle path
set_multicycle_path 3 -setup -to mul_result_reg*

# Mask out the non-relevant paths
set_false_path -from clk -to vclk_com
set_false_path -from vclk_com -to clk
set_false_path -from vclk_com -to vclk_seq
set_false_path -from vclk_seq -to vclk_com
set_false_path -from vclk_seq -to vclk_seq
```
Q 9. Are the paths constrained correctly now? And do all paths meet setup timing?
   Yes.
   
   All setup constraints are met.

Q 10. Where do these hold violations suddenly come from?
   When the multicycle exception was applied to move the setup capture edge to the third, the hold capture edge moved as well. See your student guide for details.

Task 7. Back to the Mylticycle Path

Q 11. What edge is the hold check performed at for “clk”?
   10 ns.

Q 12. Are all violations gone now?
   Yes.
   
   Finally done.

The End of Lab-6 😊