CSCE 3953 System Synthesis and Modeling
Lecture 8 Post-Synthesis Task – Static Timing Analysis

Instructor: Dr. Jia Di

Some slides are borrowed from Synopsys Galaxy 2006 Seminar Series
What is STA?

- **Static Timing Analysis**
  - (STA) is a method of determining if a circuit meets timing constraints without simulating clock cycles

- **How it works:**
  - Identify timing startpoints and endpoints
    - Input/output ports, registers/latches
  - Trace delays through timing paths from startpoints to endpoints
  - Compare path delays to clock period to see if constraints are met
  - Also check hold violations (races) and transition violations (from library design rules)

- **STA is:**
  - Exhaustive
  - Fast
  - Not dependent on simulation vectors
What is PrimeTime?

A Full Chip Gate-Level Static Timing Analyzer

- Static Timing Sign-Off Tool
- Fast and memory efficient
- Delay calculator
- Custom block modeling solution
- Advanced analysis functionality
- 350+ Vendor libraries, and timing consistency with Synthesis

Ease of Adoption

- Same design database as Design Compiler (DC)
- Same libraries as DC
- Same commands as DC
What is PT-SI?

- **PrimeTime - Signal Integrity (PT-SI)**
  - Extends PT’s timing analysis to include signal integrity effects, including:
    - Crosstalk delay
    - Noise bumps (glitches)
    - IR drop
  - SI effects analyzed in regular PrimeTime environment and included in standard timing reports

- **Benefits**
  - Fast, accurate analysis of multi-million-gate designs
  - Much faster than traditional SPICE analysis
  - Can perform simultaneous Min and Max analysis
Comprehensive Timing Checks

- Many types of timing and design rule checks
  - Timing checks can be delay calculated or SDF annotated
  - Setup and hold can depend on Slew\textsubscript{data}, Slew\textsubscript{clock}, and Cap\textsubscript{Q}
  - Design rule checks consistent with DC, PC, and Astro
Basic Timing Analysis Flow Using PrimeTime

Set Up Design Environment, Read & Link Design
- Search path, link path
- Read designs, libraries, then link
- Set operating conditions, wireload models, input drives & output loads

Specify Timing Constraints
- Clock period/waveform/uncertainty/latency
- Input/output delays

Specify Timing Exceptions
- Multicycle paths
- False paths
- Min/max delays, segmentation, disabled arcs

Perform Timing Analysis, Create Reports
- Check timing constraints
- Generate timing and constraint reports
- Generate bottleneck and coverage analysis reports
Pre- and Post-Layout Timing Flows

**Pre-layout**

- read design
- setup environment, constraints

  - Verify timing constraints
    - check_timing

  - Analyze design timing
    - Identify bottleneck cells
    - Identify snake paths
    - Use GUI or report_timing
      report_constraint,
      report_bottleneck

  - Summarize timing checks
    - report_analysis_coverage

**Post-layout**

- read design
- setup environment, constraints

  - Backannotate SDF or parasitics
  - Propagate clock network

  - Verify timing constraints
    - check_timing

  - Analyze design timing
    - Identify bottleneck cells
    - Identify snake paths
    - Use GUI or report_timing
      report_constraint,
      report_bottleneck

  - Summarize timing checks
    - report_analysis_coverage
Definitions

- search_path variable specifies where to search for design and library data
- link_path variable specifies which design and library data to be loaded during linking (link_design)
- link_design command resolves all design references

Example

```
pt_shell> set search_path "./lib"
pt_shell> set link_path "* pt_lib.db"
pt_shell> read_ddc design.ddc
pt_shell> link_design
```

Note: Designs can be read in DDC, Verilog, VHDL, Milkyway, or .db formats (read_ddc, read_verilog, read_vhdl, read_milkyway, read_db)
Set Operating Conditions, Wireload Models, Input Drives & Output Loads

• Set the operating conditions
• Set wireload models and mode (pre-layout only)
• Specify input drives and output loads

• Example:

```bash
pt_shell> set_operating_conditions -library pt_lib \
    -analysis_type on_chip_variation 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
pt_shell> set_driving_cell -lib_cell OR2 [all_inputs]
pt_shell> set_driving_cell -lib_cell IV9 [get_ports {asy_* rstN*}]
pt_shell> set_load 0.5 [all_outputs]
pt_shell> set_load 0.1 [remove_from_collection [all_inputs] CLK]
```
**Example:**

» Set up the basic timing constraints for the design. Start with the clock information.

```plaintext
pt_shell> create_clock -name CLK -period 30 [get_port CLOCK]
pt_shell> set_clock_uncertainty 0.5 [all_clocks]
pt_shell> set_clock_latency -min 3.5 [get_clocks CLK]
pt_shell> set_clock_latency -rise 5.5 [get_clocks CLK]
pt_shell> set_clock_transition -rise 0.25 [get_clocks CLK]
pt_shell> set_clock_transition -fall 0.3 [get_clocks CLK]
```

**For post layout clock tree:**

```
set_propagated_clock <clock_object_list>
```

or

```
set timing_all_clocks_propagated true
```
Specify Timing Constraints: Clocks (2)

Reference clock waveform

Reference clock waveform with uncertainty

Reference clock waveform with latency

Reference clock waveform with transition

Reference clock waveform with uncertainty, latency, and transition
Specify Timing Constraints: Clocks (3)
Modeling Off-chip Latency

- You can specify external latency:

```
pt_shell> set_clock_latency -source 1.7 CLOCK
```

The clock is delayed by 1.7 units before it gets to the clock definition point.

- You can model clock jitter*

```
pt_shell> set_clock_latency -source -rise -early 0.1 GENCLK
pt_shell> set_clock_latency -source -fall -early 0.1 GENCLK
pt_shell> set_clock_latency -source -rise -late 0.2 GENCLK
pt_shell> set_clock_latency -source -fall -late 0.3 GENCLK
```

* Note: Symmetrical clock jitter can also be specified with `set_clock_uncertainty`
Specify Timing Constraints: Input & Output Delays

- Specify signal arrival/required times at all ports relative to clocks with:
  
  ```
  set_input_delay
  set_output_delay
  ```

Examples:

```plaintext
pt_shell> set_input_delay 5.0 -clock ClkA [all_inputs]
pt_shell> set_input_delay 2.5 -clock ClkB [get_ports input2]
pt_shell> set_input_delay 0.5 -clock ClkA -add_delay input2
pt_shell> set_output_delay 3.0 -clock ClkA [all_outputs]
```

Note: For bidirectional ports, use both `set_input_delay` & `set_output_delay`. 
Exceptions to default single-cycle timing
- False paths
- Multi-cycle Paths
- User-defined max_delay, min_delay

Exceptions can have from/through/to objects
-through exceptions allow ANDing and ORing of exceptions

- **set_false_path** -through \{A B C\}
  Path must travel through points A or B or C to be considered false

- **set_multicycle_path** 2 -through \{A B\} -through \{C\}
  Path must travel through points (A or B) and C to be a multi-cycle path
Specify Timing Exceptions
Avoid Overuse of Wildcards!

- Large numbers of timing exceptions increases memory usage and timing analysis runtimes **SUBSTANTIALLY**.
- Example:
  - Set false paths between 32-bit registers Reg_A and Reg_B in asynchronous clock domains clocked by clkA and clkB

  ```
  pt_shell> set_false_path -from Reg_A*/Q -to Reg_B*/D
  pt_shell> set_false_path -from Reg_B*/Q -to Reg_A*/D
  ```

  This creates and stores $2\times32\times32 = 2048$ false path exceptions! (BAD)

  or

  ```
  pt_shell> set_false_path -from clkA -to clkB
  pt_shell> set_false_path -from clkB -to clkA
  ```

  This creates and stores **2 false path exceptions! (GOOD)**
Specify Timing Exceptions
Single -through Pin

- What looks like a single exception through a pin is usually many exceptions from the start-points that fan-in to the pin to the end-points that fan-out from the pin.

**Bad:**

```
pt_shell> set_false_path -through [get_pins ADDER/CI]
```

- Use `set_disable_timing` to disable ALL timing paths through the pin. Disabling the paths this way means that PrimeTime does NOT have to store information about the multiple exceptions.

**Good:**

```
pt_shell> set_disable_timing [get_pins ADDER/CI]
```
- report_exceptions –ignored
- transform_exceptions
- Removes ignored timing exceptions from memory
- Useful with legacy constraints
- Options: -from, -to, -rise_from, -rise_to, -fall_from, -fall_to, -rise_through, -fall_through, -dry_run, -verbose, -flatten

```
pt_shell> report_exceptions -ignored
pt_shell> transform_exceptions
pt_shell> write_sdc -out clean_constraints
pt_shell> report_exceptions -ignored
```
Perform Timing Analysis

Check Constraints

- To identify problems with design or assertions before you spend time on detailed reports
  - Use `check_timing [-verbose]`
  - Review man page for list of default checks
    - `timing_check_defaults` list of timing checks performed by `check_timing`
      (Not all available checks are run by default)

- Unconstrained endpoints?
- Combinational loops?
- Missing clock definitions?
- Multiple clock fanin?
- Latch fanout problems?
- Generated clocks consistent?
- Ignored timing exceptions?
- Ports with missing input delay?
...
Perform Timing Analysis
Report Coverage

- To summarize timing checks met, violated, untested:
  - Determine if your analysis is complete
  - Show untested checks and reason why not tested
  - Use `report_analysis_coverage --status_details {untested}`

<table>
<thead>
<tr>
<th>Type of Check</th>
<th>Total</th>
<th>Met</th>
<th>Violated</th>
<th>Untested</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup</td>
<td>5</td>
<td>2 (40%)</td>
<td>2 (40%)</td>
<td>1 (20%)</td>
</tr>
<tr>
<td>hold</td>
<td>5</td>
<td>4 (80%)</td>
<td>0 (0%)</td>
<td>1 (20%)</td>
</tr>
<tr>
<td>recovery</td>
<td>2</td>
<td>0 (0%)</td>
<td>2 (100%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>min_period</td>
<td>1</td>
<td>1 (100%)</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>min_pulse_width</td>
<td>2</td>
<td>2 (100%)</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td><strong>All Checks</strong></td>
<td>15</td>
<td>9 (60%)</td>
<td>4 (27%)</td>
<td>2 (13%)</td>
</tr>
</tbody>
</table>

...
Summarize timing results:
  » Use `report_constraint <options>`
    - Note: `-all_violators -verbose` option may cause long run times. Use `-max_delay` or `-min_delay` to focus on setup or hold violations.

Display worst violations:
  » Use `report_timing <options>`
    - Note: large values for `-nworst` or `-max_paths` option may cause long run times.
    - Unconstrained paths are not listed by default:
      ```
      timing_report_unconstrained_paths = "false"
      ```

Report bottleneck cells contributing to multiple violations:
  » Use `report_bottleneck <options>`
New report_timing –exclude option
- Allows designers to report paths NOT through a given pin or sub-block
- Excludes all paths from/through/to specified pins
- Enhances usability and flexibility for path analysis

Example:
- report_timing –exclude object_list
- get_timing_paths –exclude object_list
- object_list: a list of pins, ports, cells, or nets
- Also supported: –rise_exclude, –fall_exclude
check_timing
report_analysis_coverage
report_bottleneck
report_constraint
report_constraint
  -all_violators
  -max_delay -min_delay
report_constraint
  -all_violators
  -verbose
report_timing
  -nworst 100
Static Timing Analysis using PrimeTime
What to Analyze?

- From primary inputs to all FFs in the design
- From FF to FF
- From FF to primary outputs of the design
- From primary inputs to primary outputs of the design

- report_timing –from [all_inputs] –to [all_registers –data_pins]
- report_timing –from [all_registers –clock_pins] –to [all_registers –data_pins]
- report_timing –from [all_registers –clock_pins] –to [all_outputs]
- report_timing –from [all_inputs] –to [all_outputs]

- To speed it up:
  report_timing –to [all_registers –data_pins]
  report_timing –to [all_outputs]
Timing Exceptions

- Multi-cycle Paths
  - PT by default treats all paths in the design as single-cycle and performs the STA accordingly, i.e., data is launched from the driving FF using the first edge of the clock, and is captured by the receiving FF using the second edge of the clock.
  - In multi-cycle mode, the data may take more than one clock cycle to reach its destination.
  - `set_multicycle_path <multiplier value>`
    - `from <from list>`
    - `to <to list>`
In1

regA

Clk

regB

Out

Clk at regA

Clk at regB

Single-cycle Timing Relationship

setup

hold

Multi-cycle Timing Relationship

setup

hold

set_multicycle_path 2 –from regA –to regB
Timing Exceptions (Cont’)

- Multi-cycle Paths (Cont’)
  - For separate clocks with different frequencies, the `set_multicycle_path` command may be used to define the relationship between these clocks.
  - By default, PT uses the most restrictive setup and hold-time relationship between these clocks, which may be overridden by this command.
False Paths
- A false path is identified as a timing path that does not propagate a signal
- `set_false_path -from <from list> -to <to list> -through <through list>`
- The above command removes the constraints of the identified path
- False path identification is recommended before performing STA

Suppose there are multiple false paths in the design and they are all failing by a large amount during hold-time STA; while the real paths are failing by a small margin. If the user uses the following command for STA:
  `report_timing -from [all_inputs]
   -to [all_registers –data_pins]
   -nworst 10000 –max_paths 1000
   -delay_type min`

PT will generate/display multiple timing reports, covering all the paths in the design, most of which are false paths. It is tedious to distinguish between the real and false timing paths.
Environment and Constraints

- set_wire_load_model
- set_wire_load_mode
- set_operating_conditions
- set_load
- set_input_delay
- set_output_delay
- Use the worst-case operating conditions to perform setup-time analysis, and the best-case operating conditions to perform hold-time analysis
Pre-Layout Clock Specification

- Estimate the post-route clock-tree delays upfront
- The estimated clock transition should be defined in order to prevent PT from calculating false delays (usually large) for the driven gates
- `create_clock -period 20 -waveform [list 0 10] [list CLK]`
- `set_clock_latency 2.5 [get_clocks CLK]`
- `set_clock_transition 0.2 [get_clocks CLK]`
- `set_clock_uncertainty 1.2 -setup [get_clocks CLK]`
- `set_clock_uncertainty 0.5 -hold [get_clocks CLK]`
## Analysis Reports
### – Pre-Layout Setup-Time

**report_timing** – from tdi – to [all_registers – data_pins]

| Report  | timing  |  |  |
|---------|---------|  |  |
|         | – path full |  |  |
|         | – delay max |  |  |
|         | – max_paths 1 |  |  |
| Design  | tap_controller |  |  |
| Version | 1998.08 – PT2 |  |  |
| Date    | Tue Nov 17 11:16:18 1998 |  |  |

Startpoint: tdi (input port clocked by tck)
Endpoint: ir_block/ir_reg0  
(rising edge-triggered flip-flop clocked by tck)
Path Group: tck
Path Type: max

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock tck (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>input external delay</td>
<td>15.00</td>
<td>15.00 r</td>
</tr>
<tr>
<td>tdi (in)</td>
<td>0.00</td>
<td>15.00 r</td>
</tr>
<tr>
<td>pads/tdi (pads)</td>
<td>0.00</td>
<td>15.00 r</td>
</tr>
<tr>
<td>pads/tdi_pad/Z (PAD1X)</td>
<td>1.32</td>
<td>16.32 r</td>
</tr>
<tr>
<td>pads/tdi_signal (pads)</td>
<td>0.00</td>
<td>16.32 r</td>
</tr>
<tr>
<td>ir_block/tdi (ir_block)</td>
<td>0.00</td>
<td>16.32 r</td>
</tr>
<tr>
<td>ir_block/U1/Z (AND2D4)</td>
<td>0.28</td>
<td>16.60 r</td>
</tr>
<tr>
<td>ir_block/U2/ZN (INV0D2)</td>
<td>0.33</td>
<td>16.93 f</td>
</tr>
<tr>
<td>ir_block/U1234/Z (OR2D0)</td>
<td>1.82</td>
<td>18.75 f</td>
</tr>
<tr>
<td>ir_block/U156/ZN (NOR3D2)</td>
<td>1.05</td>
<td>19.80 r</td>
</tr>
<tr>
<td>ir_block/ir_reg0/D (DFF1X)</td>
<td>0.00</td>
<td>19.80 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td>19.80</td>
<td></td>
</tr>
<tr>
<td>clock tck (rise edge)</td>
<td>30.00</td>
<td>30.00</td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>32.50</td>
</tr>
<tr>
<td>ir_block/ir_reg0/CP (DFF1X)</td>
<td>32.50</td>
<td></td>
</tr>
<tr>
<td>library setup time</td>
<td>-0.76</td>
<td>31.74</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>31.74</td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td>31.74</td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td>-19.80</td>
</tr>
<tr>
<td>slack (MET)</td>
<td></td>
<td>11.94</td>
</tr>
</tbody>
</table>
Analysis Reports
– Pre-Layout Hold-Time

\[
\text{report\_timing} \quad - \text{from [all\_registers --clock\_pins] -to [all\_registers --data\_pins] -delay\_type min}
\]

\[
\begin{array}{l}
\text{Report} : \text{timing} \\
\quad - \text{path full} \\
\quad - \text{delay min} \\
\quad - \text{max\_paths 1} \\
\text{Design} : \text{tap\_controller} \\
\text{Version} : \text{1998.08--PT2} \\
\text{Date} : \text{Tue Nov 17 11:16:18 1998}
\end{array}
\]

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

\[
\begin{array}{l}
\text{Startpoint: state\_block/st\_reg9} \\
\text{ (rising edge-triggered flip-flop clocked by tck)} \\
\text{Endpoint: state\_block/bp\_reg2} \\
\text{ (rising edge-triggered flip-flop clocked by tck)} \\
\text{Path Group: tck} \\
\text{Path Type: min}
\end{array}
\]

<table>
<thead>
<tr>
<th>Point</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock tck (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>2.50</td>
</tr>
<tr>
<td>state_block/st_reg9/CP (DFF1X)</td>
<td>0.00</td>
<td>2.50 r</td>
</tr>
<tr>
<td>state_block/st_reg9/O (DFF1X)</td>
<td>0.00</td>
<td>2.50 r</td>
</tr>
<tr>
<td>state_block/U15/Z (BUFF4X)</td>
<td>0.15</td>
<td>2.70 r</td>
</tr>
<tr>
<td>state_block/bp_reg2/D (DFF1X)</td>
<td>0.10</td>
<td>2.80 r</td>
</tr>
<tr>
<td>data arrival time</td>
<td>2.80</td>
<td></td>
</tr>
<tr>
<td>clock tck (rise edge)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>2.50</td>
</tr>
<tr>
<td>state_block/bp_reg2/CP (DFF1X)</td>
<td>2.50</td>
<td>2.50 r</td>
</tr>
<tr>
<td>library hold time</td>
<td>0.50</td>
<td>3.00</td>
</tr>
<tr>
<td>data required time</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>-2.80</td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td>-0.20</td>
<td></td>
</tr>
</tbody>
</table>
### Analysis Reports

**Detailed Timing Report**

`report_timing` --from state_block/st_reg9/CP --to state_block/bp_reg2/D
--delay_type min --nets --capacitance --transition_time

<table>
<thead>
<tr>
<th>Point</th>
<th>Fanout</th>
<th>Cap</th>
<th>Trans</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock tck (rise edge)</td>
<td>0.30</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/st_reg9/CP (DFF1X)</td>
<td>0.30</td>
<td>0.00</td>
<td>2.50 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/st_reg9/Q (DFF1X)</td>
<td>0.12</td>
<td>0.05</td>
<td>2.55 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/n1234 (net)</td>
<td>2</td>
<td>0.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/U15/Z (BUFF4X)</td>
<td>0.32</td>
<td>0.15</td>
<td>2.70 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/n2345 (net)</td>
<td>8</td>
<td>2.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/bp_reg2/D (DFF1X)</td>
<td>0.41</td>
<td>0.10</td>
<td>2.80 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td></td>
<td>2.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock tck (rise edge)</td>
<td>0.30</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/bp_reg2/CP (DFF1X)</td>
<td></td>
<td></td>
<td>2.50 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>library hold time</td>
<td>0.50</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td></td>
<td>3.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td></td>
<td></td>
<td>3.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td></td>
<td></td>
<td>-2.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slack (VIOLATED)</td>
<td></td>
<td></td>
<td>-0.20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

swap_cell {U15} [get_lib_cell stdcell_lib/BUFF1X]
Analysis Reports
– Detailed Timing Report

report_timing –from state_block/st_reg9/CP –to state_block/bp_reg2/D
–delay_type min –nets –capacitance –transition_time

*******************************
Report : timing
–path full
–delay min
–max_paths 1
Design : tap_controller
Version : 1998.08–PT2
Date : Tue Nov 17 11:16:18 1998
*******************************

Startpoint: state_block/st_reg9
(rising edge-triggered flip-flop clocked by tck)
Endpoint: state_block/bp_reg2
(rising edge-triggered flip-flop clocked by tck)
Path Group: tck
Path Type: min

<table>
<thead>
<tr>
<th>Point</th>
<th>Fanout</th>
<th>Cap</th>
<th>Trans</th>
<th>Incr</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock tck (rise edge)</td>
<td>0.30</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/st_reg9/CP (DFF1X)</td>
<td>0.30</td>
<td>0.00</td>
<td>2.50 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/st_reg9/Q (DFF1X)</td>
<td>0.12</td>
<td>0.05</td>
<td>2.55 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/n1234 (net)</td>
<td>2</td>
<td>0.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/U15/Z (BUFF1X)</td>
<td>1.24</td>
<td>0.40</td>
<td>2.95 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/n2345 (net)</td>
<td>8</td>
<td>2.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/bp_reg2/D (DFF1X)</td>
<td>1.25</td>
<td>0.10</td>
<td>3.05 r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data arrival time</td>
<td>3.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock tck (rise edge)</td>
<td>0.30</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock network delay (ideal)</td>
<td>2.50</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>state_block/bp_reg2/CP (DFF1X)</td>
<td>0.50</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>library hold time</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data required time</td>
<td>3.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

slack (MET) | 0.05
Sometimes a design may contain multiple path segments that share a common leaf cell. If these path segments are failing timing then changing the drive strength of the common leaf cell may remove the timing violation for all the path segments.

PT provides the capability of identifying a common leaf cell that is shared by multiple violating path segments in a design.

```
report_bottleneck
```

Bottleneck Cost = Number of violating paths through cell

<table>
<thead>
<tr>
<th>Cell</th>
<th>Reference</th>
<th>Bottleneck Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>U15</td>
<td>BUFF4X</td>
<td>2.00</td>
</tr>
</tbody>
</table>