



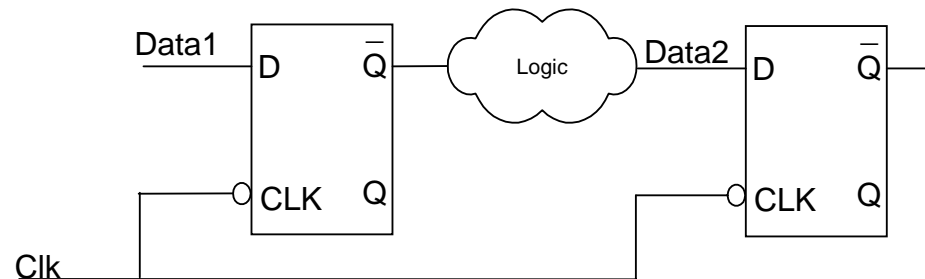
# Introduction to Static Timing Analysis with PrimeTime

Jiuling Tang



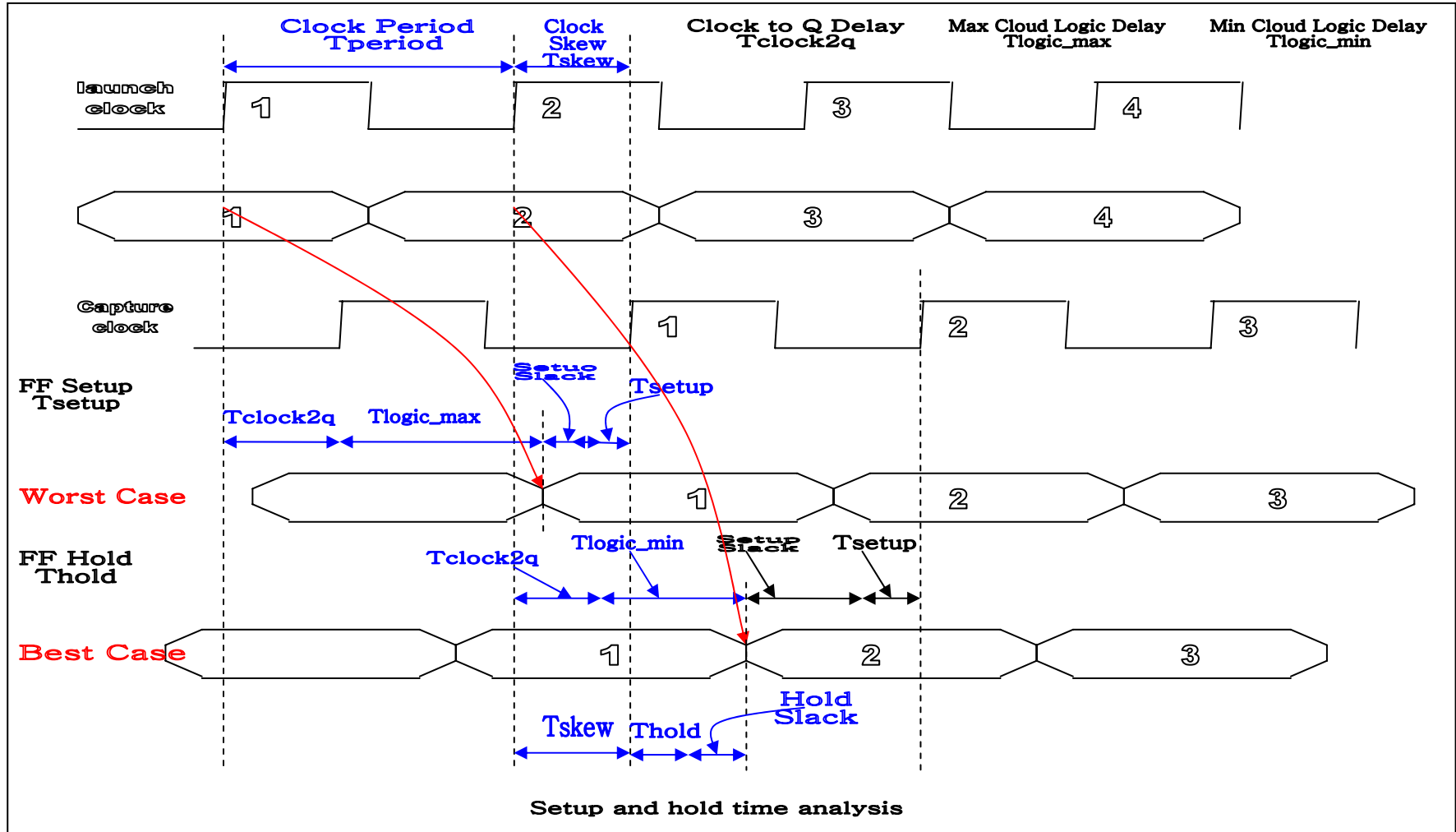
# Timing Analysis

- Mainly check every flip-flop meets its setup and hold time requirements





# Data and Clock Waveforms





# Flip-Flop (FF) Setup and Hold Time Requirement

$$T_{clock2q} + T_{logic\_max} + T_{setup} < T_{period} + T_{skew}$$

$$T_{clock2q} + T_{logic\_min} > T_{hold} + T_{skew}$$



# Timing Analysis Methods

- Dynamic Timing Analysis
  - Running SPICE simulation
  - Or running exhaustive gate-level simulations with SDF
  - Accurate
  - Slow
- Static Timing Analysis (STA)
  - Tools, like Synopsys PrimeTime
  - Less accurate, but good enough
  - Faster



# Synopsys PrimeTime (PT)

- The most trusted and advanced timing sign-off solution for gate-level STA tool
- Analysis of up to 100 - million gate design
- Accurate to within 5% of SPICE
- Delay calculation by using parasitics information for accurate interconnect analysis
- Advanced modeling
  - Interface Logic Models (ILM) for hierarchical static timing analysis and sign-off
  - Extracted Timing Models (ETM) in .lib format for cell-based reusable IP and physical design flows
  - Quick Timing Models (QTM) for top-down design



# How STA Works

- Find out all valid paths in the design
- Use technology libraries characterized by SPICE to calculate the delay of each path
- Check all the path delays to see if setup and hold time have been meet

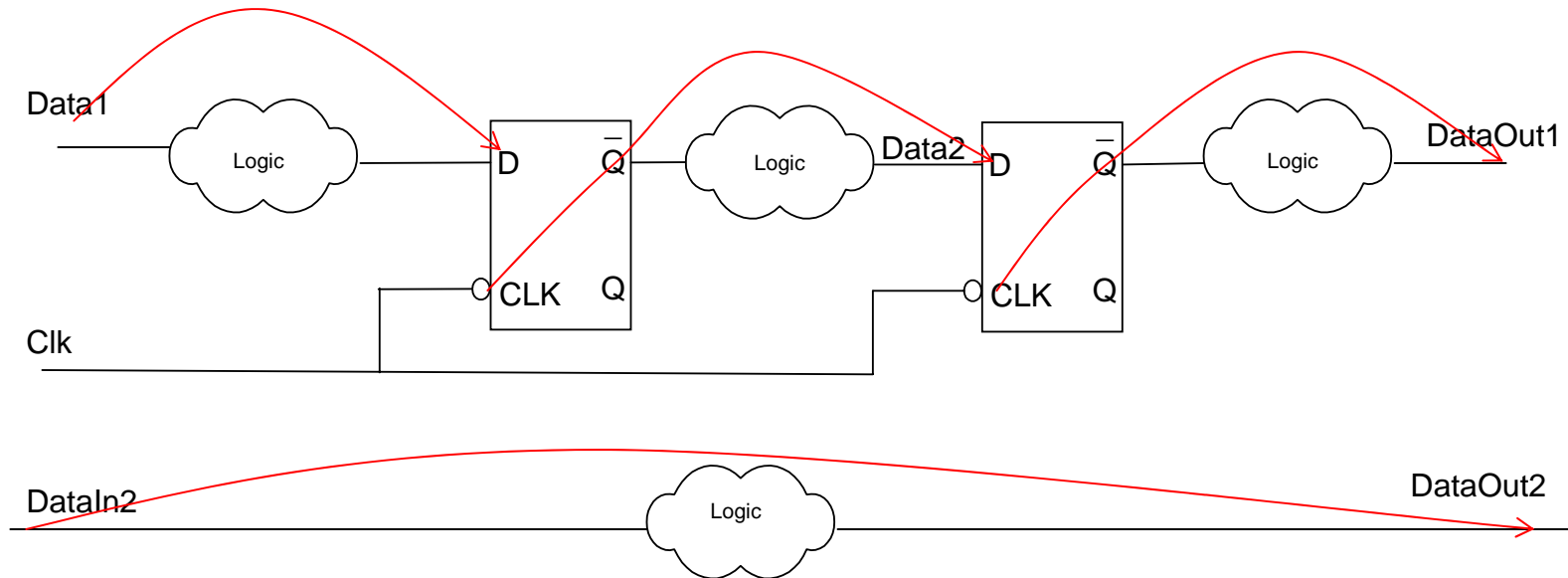


# Timing Paths

- Start points
  - Inputs
  - Clock pins of flip-flops
- End points
  - Outputs
  - Data input pins of flip-flops



# Timing Paths (continued)





# Path Delay

- Sum up all the cell and net delays along a timing path
  - Cell delay
    - Depend on input transition and output load
    - Use Non Linear Delay Model (NLDM) before and after layout
  - Net delay
    - From wire load model before layout
    - From parasitic extraction after layout



# Clock Jitter, Skew, Transition, and Latency

- Before layout
  - Have to estimation skew, transition and latency
  - *set\_clock\_uncertainty*
  - *set\_clock\_transition*
- After layout
  - From actual clock tree
  - *set\_propgated\_clock*



# Three Types of STA Analysis

- Single
  - Check for setup time only
  - Use one process, voltage, and temperature (PVT) condition
- Best case (BC) and worst case (WC)
  - Check both setup and hold time
  - BC uses one PVT condition
  - WC uses another one PVT condition
- On chip variation
  - Conservative analysis, Check both setup and hold time
  - For setup checks, it considers a slow launch clock path and slow data path, but a fast capture clock path. hold checks, it does the opposite
  - Use two PVTs to represent variation across a die



# Tool Command Language (TCL)

- TCL is the command interface to PT
- TCL is an interpreted, programmable, and scriptable language
- TCL also widely used by other EDA tools, like Design Compiler, Physical Compiler, PT, ModelSim, Xilinx ISE...
- PT only has TCL mode
- For more information about PT TCL commands, just type help at PT prompt



# STA Steps of PT

- 1. Read required files**
- 2. Constraint the design**
- 3. Verify constraint coverage**
- 4. Analyze**
- 5. Generate reports**



# Required Input Files

- PT setup file: .synopsys\_pt.setup
- Technology libraries
- Gate-level netlist, support format
  - db
  - Verilog
  - VHDL



# Set Up of PT

- Set up the `search_path` variable
  - A list of paths used to locate the design, libraries and other files needed by PT
  - *set search\_path { ./vhdl ./scripts }*
- Set up the `link_path` variable
  - Places where PT can find designs and libraries when linking the design
  - *set link\_path { \* tech\_lib.db rams.db }*
  - It is same as DC `link_library`



# Reading Design into PT

- Use following commands to read designs into PT
  - *read\_db*
  - *read\_vhd*
  - *read\_verilog*
- Manually read all files
- Read top module first and PT will read all sub-module automatically



# Resolving Reference

- PT uses command *link\_design* resolves all reference in a design using the *link\_path*.
- If fails, it will find and replace the space holder with actual library cells or sub designs referenced specified in *search\_path*



# Constraining internal FF to FF Timing Paths

- Create a clock constraints all internal FF to FF timing paths. like
  - *create\_clock –period ClockPeriod [get\_ports CLK]*
- If consider clock skew and jitter, use clock uncertainty to model them
  - *set\_clock\_uncertainty –setup number [get\_ports CLK]*
- Uncertainties between synchronous clocks
  - *set\_clock\_uncertainty –setup 0.5 -from [get\_clocks Clock1] -to [get\_clocks Clock2]*

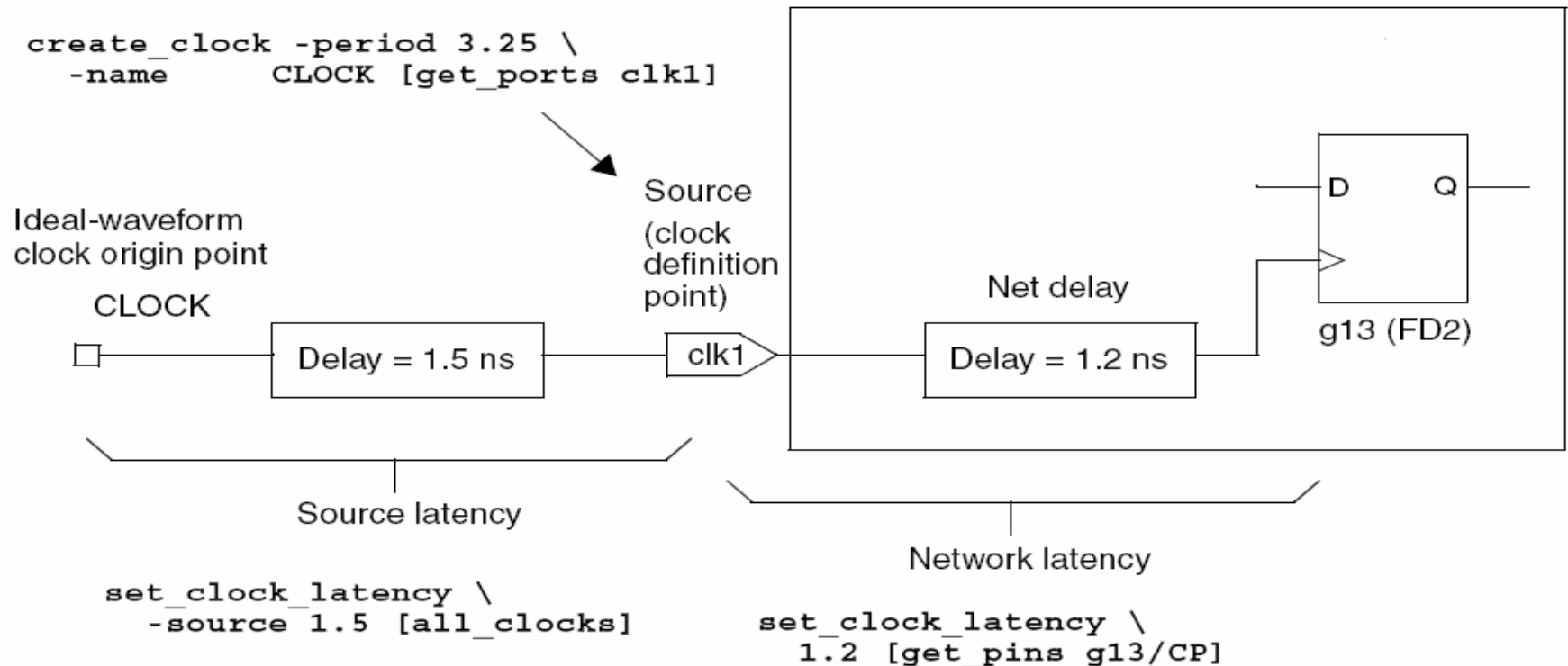


# Clock Latency and Transition before Layout

- Network latency – due to clock tree insertion
  - *Set\_clock\_latency –max 1 [get\_clocks Clock]*
- Source latency – due to clock source not very clock
  - *Set\_clock\_latency –source –max 1 [get\_clocks Clock]*
- Clock transition
  - *set\_clock\_transition 0.2 [all\_clocks]*



# Clock Source and Network Latency



Source: Synopsys

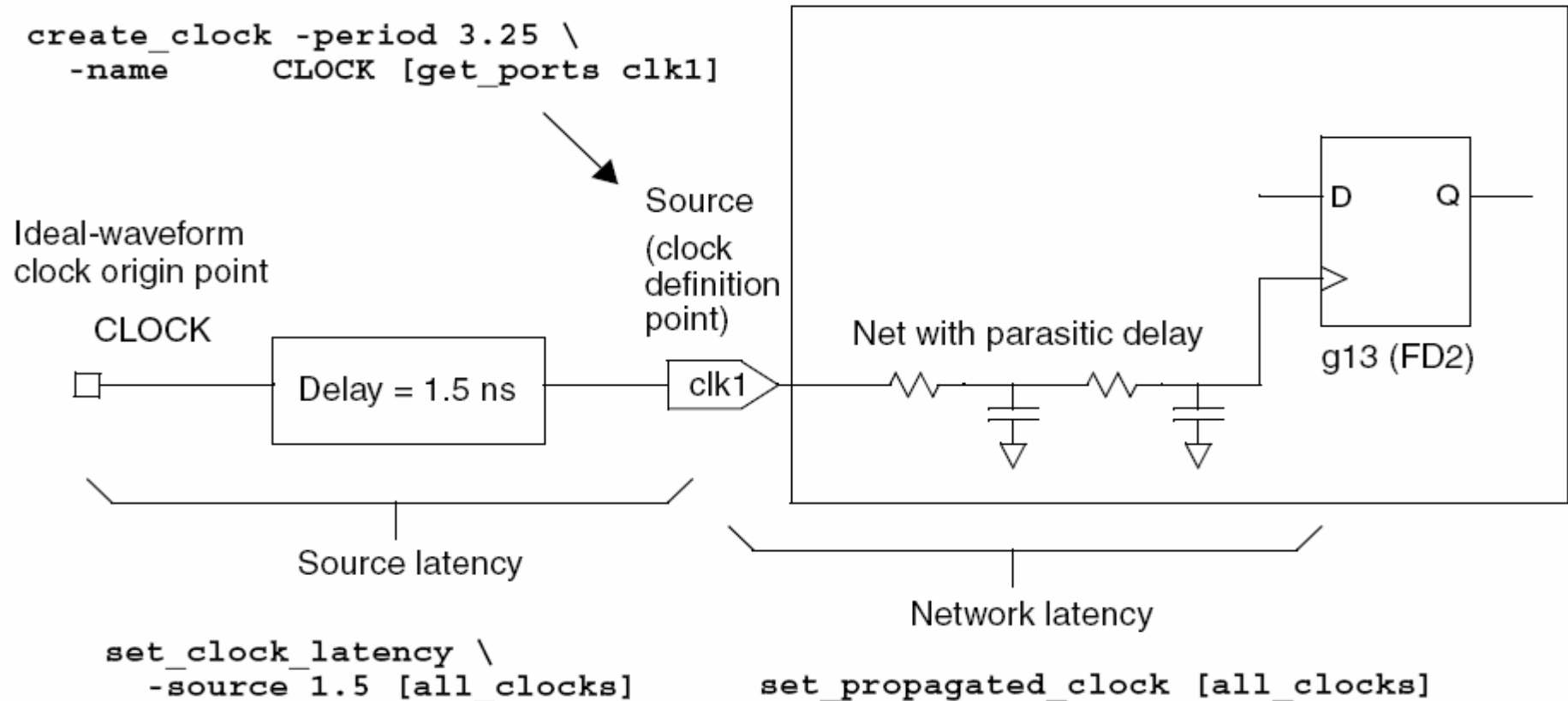


# Clock Latency and Transition after Layout

- Network latency – due to clock tree insertion
  - *Set\_propagated\_clock [all\_clocks]*
- Source latency – due to clock source not very clock
  - *Set\_clock\_latency –source –max 1 [get\_clocks Clock]*



# Clock Source and Network Latency

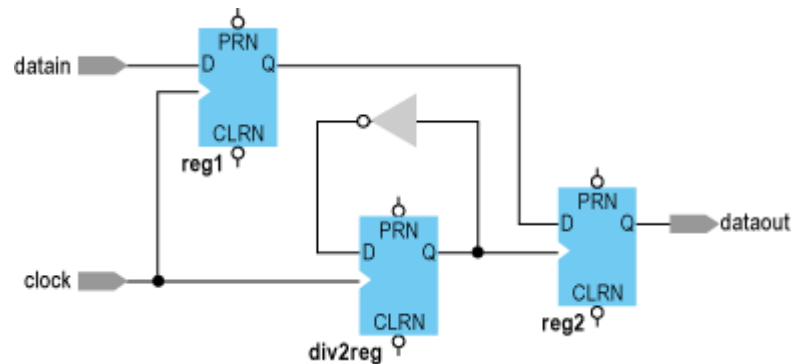


Source: Synopsys



# Generated Clock

- If there are generated clocks inside the design, use
  - `create_generated_clock -source clock -name div2clock -divide_by 2 [get_pins div2reg/regout]`



(Source: Altera)

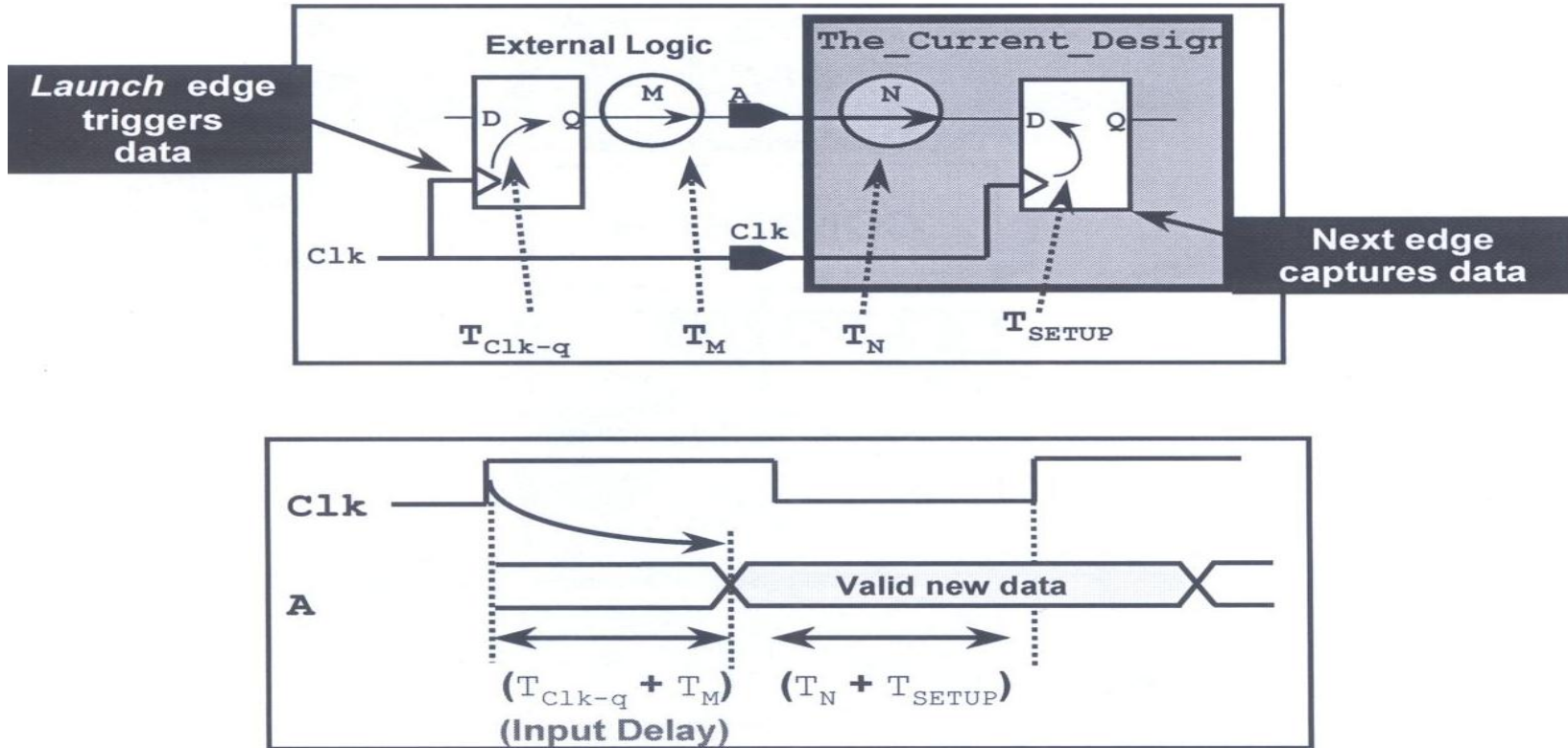


# Constraint Input Paths

- To specify the maximum arrival time of external data with respect to a launching clock and inputs, need to set input delay. For example
  - *Set\_input\_delay -max 2 -clock Clock [get\_inputs Datain]*



# Set Input Delay



Source: Synopsys



# Driving cell for input ports

- To calculate the timing of inputs cells, use *set\_driving\_cell* to specify external driving cell for input ports
  - *set\_driving\_cell -lib\_cell buf1 [get\_ports DataIn]*

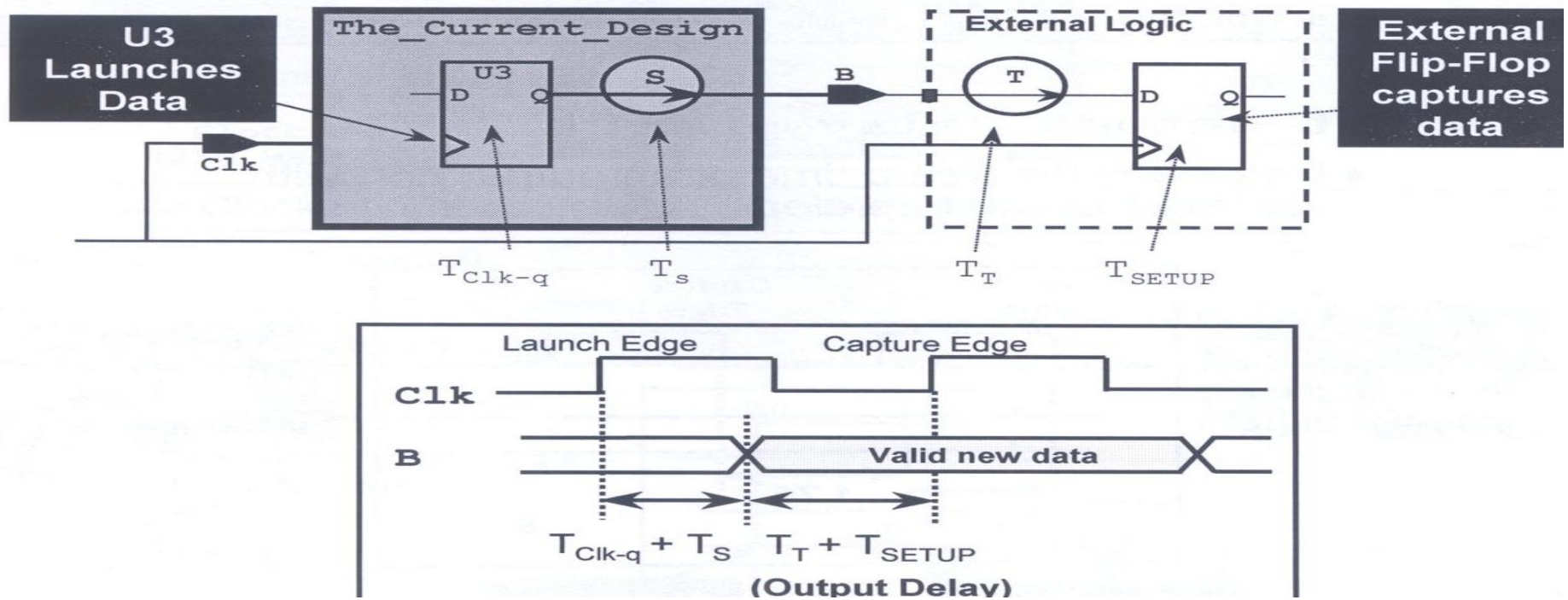


# Constraint output Paths

- To specify the setup time of an output with respect to a capture clock, use set output delay. For example
  - *set\_output\_delay -max 2 -clock Clock [get\_outputs DataOut]*



# Output Delay



Source: Synopsys



# Setting Load for Output Ports

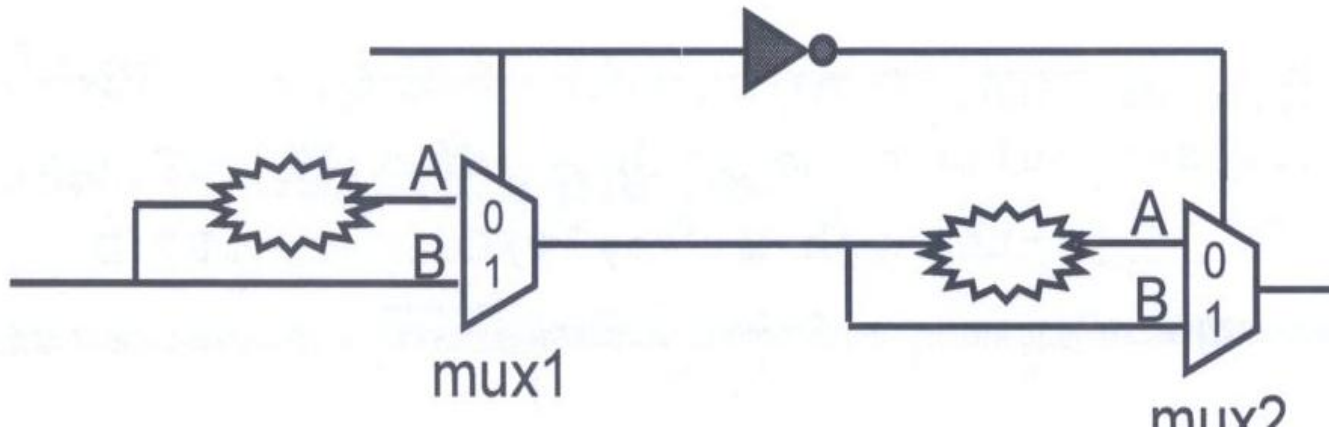
- To calculate the timing of outputs cells, use set load to specify external load for output ports
  - *set\_load 5 [get\_ports DataOut]*
  - Or *set LOAD [expr [load\_of fir\_core/INV/A]\*3]*  
*set\_load \$LOAD [get\_ports DataOut]*



# False Paths

- False paths:
  - logically impossible
  - Between asynchronous clocks
- PT commands used to specify false paths
  - *set\_false\_path -from [get\_clocks ClkA] -to [get\_clocks ClkB]*
  - *set\_clock\_groups -asynchronous -name ASYNC -group ClkA -group ClkB*

# An Example of False Path



Source: Synopsys

It's logically impossible from mux1/A to mux2/A or from mux1/B to mux2/B



# Multi-cycle Paths

- Multi-cycle paths:
  - More than 1 clock cycle
- PT commands used to specify multi-cycle paths
  - *set\_multicycle\_path 10 –setup –to [get\_pins fir\_reg[\*]/D]*
  - *set\_multicycle\_path 9 –hold –to [get\_pins fir\_reg[\*]/D]*



# Specifying Operating Condition

- Setting operating condition by using
  - *set\_operating\_conditions –analysis\_type bc\_wc –max WC\_OC –min BC\_OC*
  - *set\_operating\_conditions –analysis\_type on\_chip\_variation –max WC\_OC –min BC\_OC*



# Design Rules

- Restriction imposed on the use of library cells by technology vendors. Vendors impose design rules that restrict how many cells are connected to one another based on capacitance, transition, and fanout
- Design rules has high priority of all. Related commands are
  - *set\_max\_capacitance*
  - *set\_max\_transition*
  - *set\_max\_fanout*



# Checking Design Constraints

- *check\_timing* shows problems like:
  - Missing clock definition
  - Ports no input delay or output delay
  - Input or output delays without a reference clock
  - Combinational feedback loops
  - Unconstrained endpoints for setup



# Performing STA

- Three techniques for performing STA
  1. Generating constraint report
    - *report\_constraint –all*
  2. Generating bottleneck report
    - *report\_bottleneck*
  3. Generating timing report
    - *report\_timing*



# Constraint Analysis

- *report\_constraints* give us following violations in design
  - Setup/hold
  - DRC
  - Pulse width
- *report\_constraints –all\_violators* shows all the violations and where violations are
- Things need to investigate
  - *How many end points violate timing? End points are registers or outputs*
  - *What is the largest violation?*
  - *How many DRC violations?*



# Bottleneck Analysis

- *report\_bottleneck* shows the cells involved in multiple violations
- Identify submodule containing the bottleneck cells and refine the module and resynthesizing them
- Replace the submodule with new one
- Regenerate constraint report

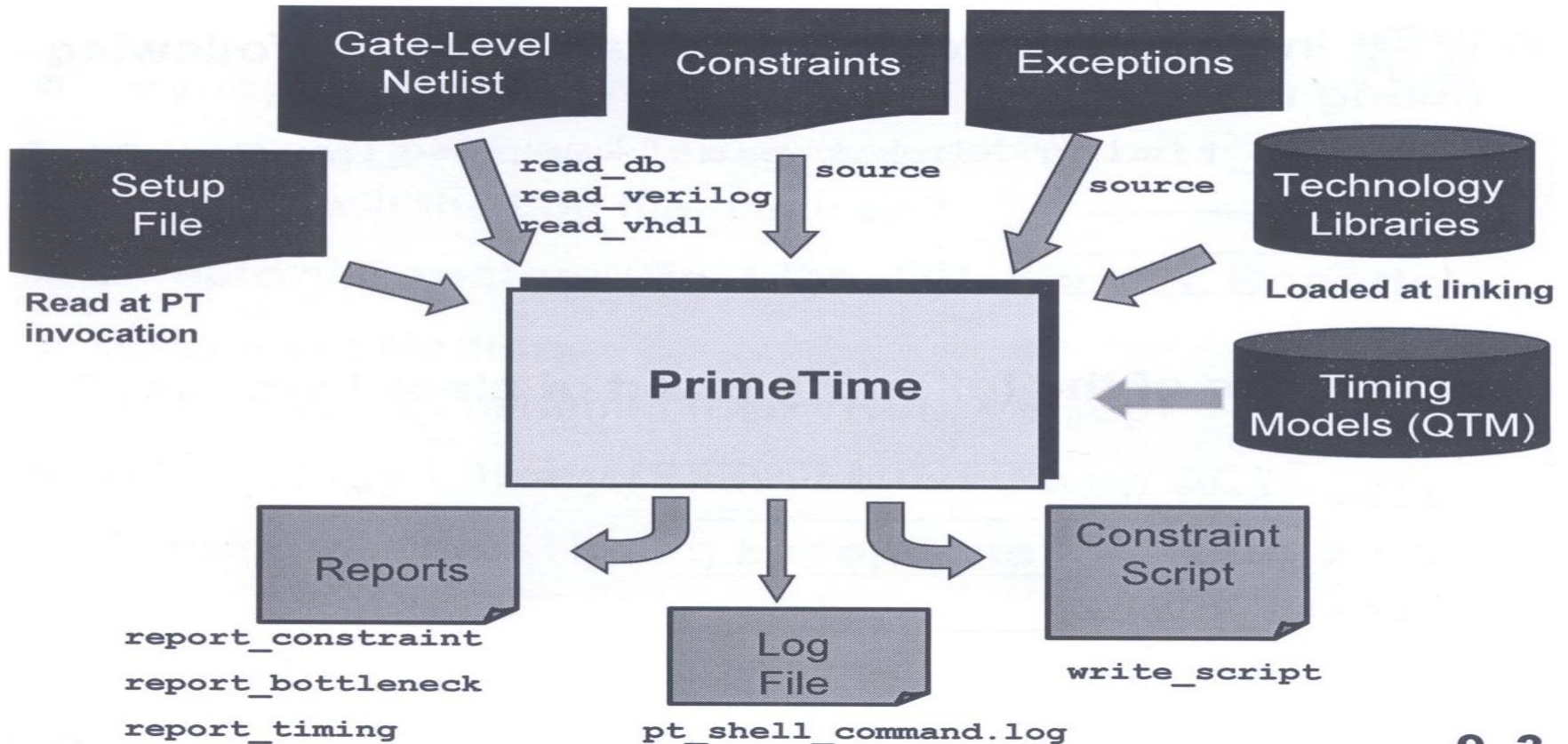


# Timing Report

- *report\_timing* finds all the timing paths
- Analyzes each path for timing twice for both rising and falling edges
- Shows critical paths for each clock group
- From above information, we investigate the causes of the violations due to
  - Poor partition
  - Excessive net fanout
  - Large capacitance load
  - Slower transition time



# Summary



Source: Synopsys



# Demos