

# **STA - Static Timing Analysis**

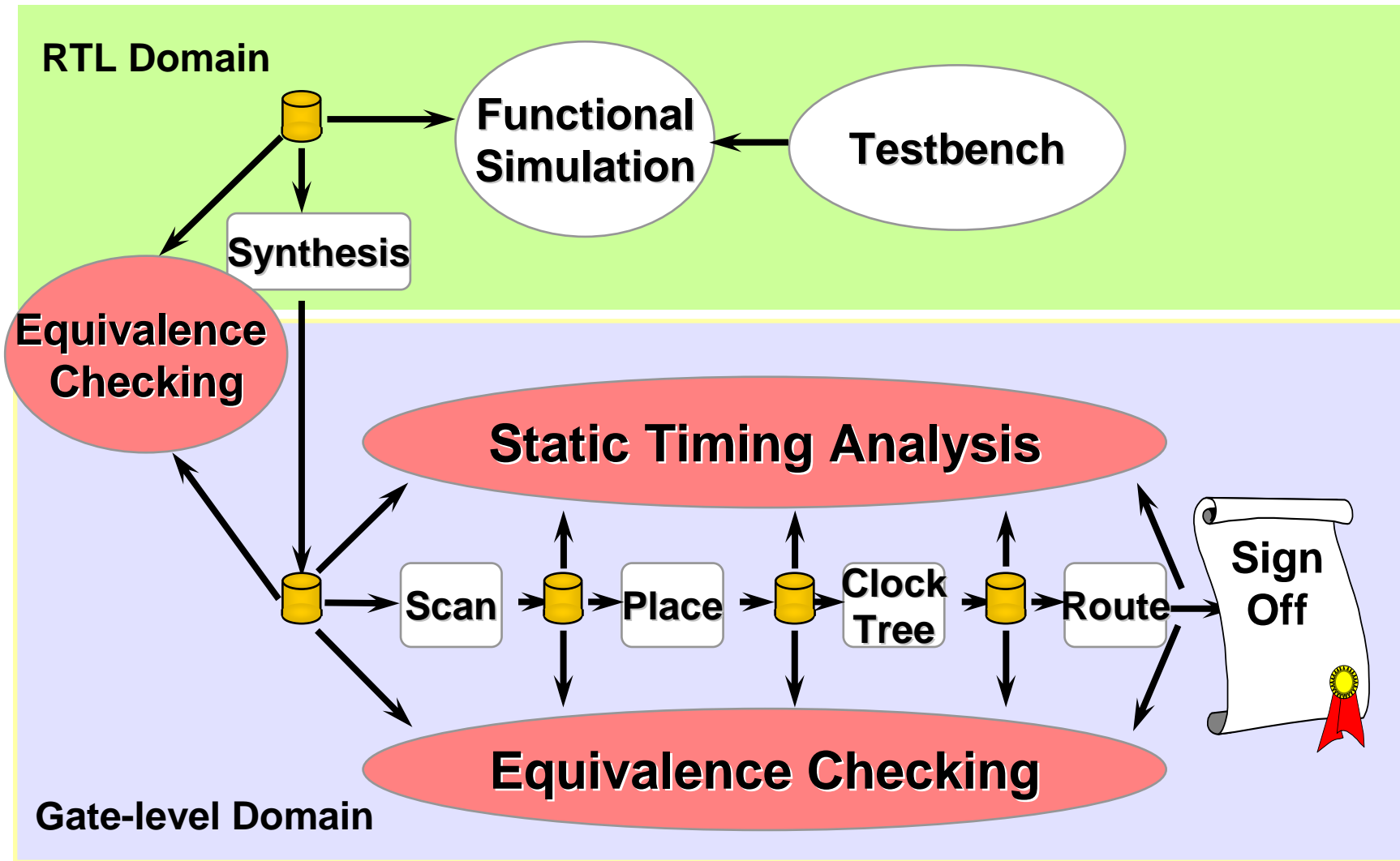
**STA**

**Lecturer: Gil Rahav**

**Semester B' , EE Dept. BGU.**

**Freescale Semiconductors Israel**

# Static Verification Flow

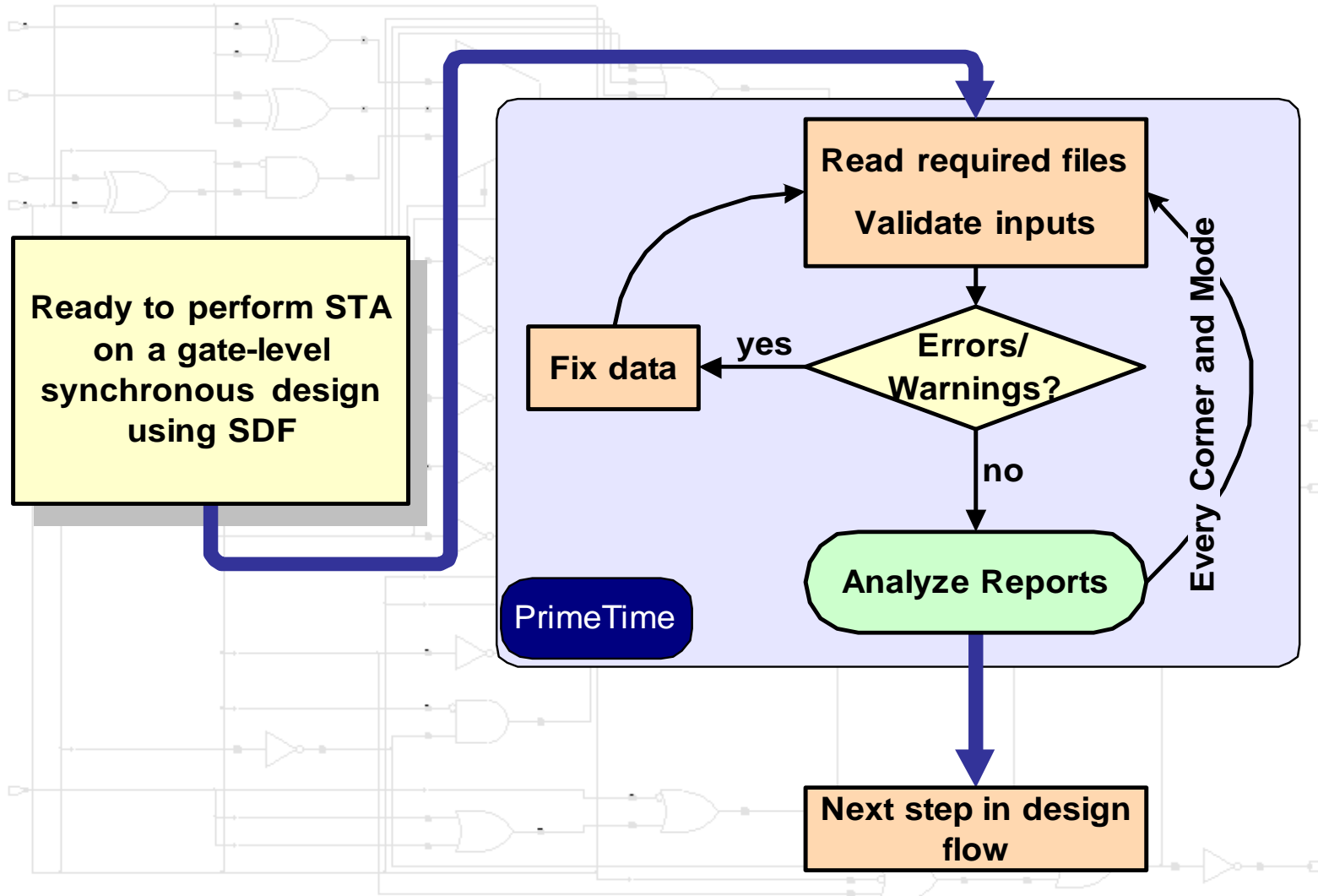


# What is Static Verification?

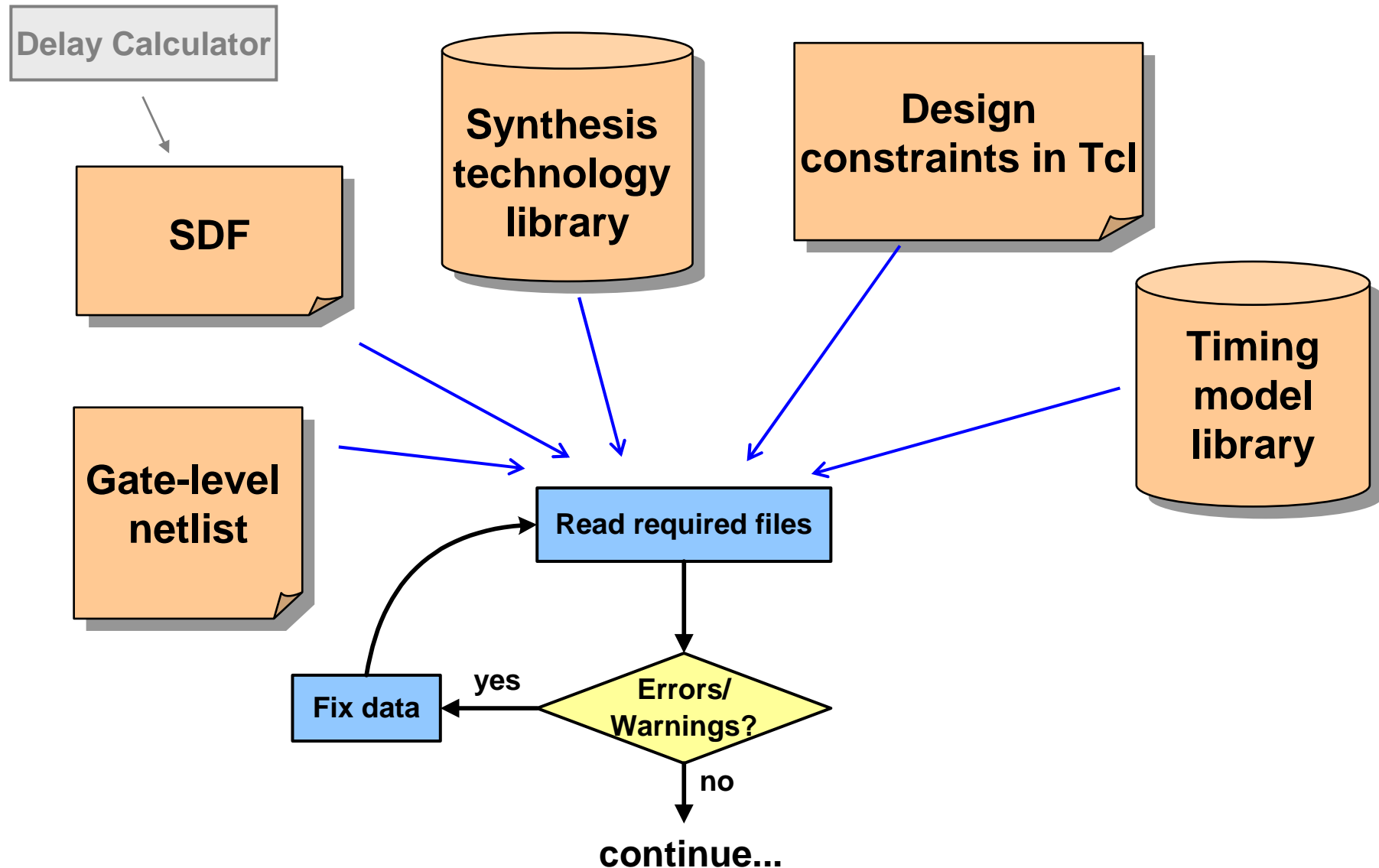
## ■ Static verification:

- Verifies timing and functionality
  - ◆ STA and equivalence checking
- Is exhaustive
- Uses formal, mathematical techniques instead of vectors
- Does not use dynamic logic simulation

# Static Timing Analysis Flow

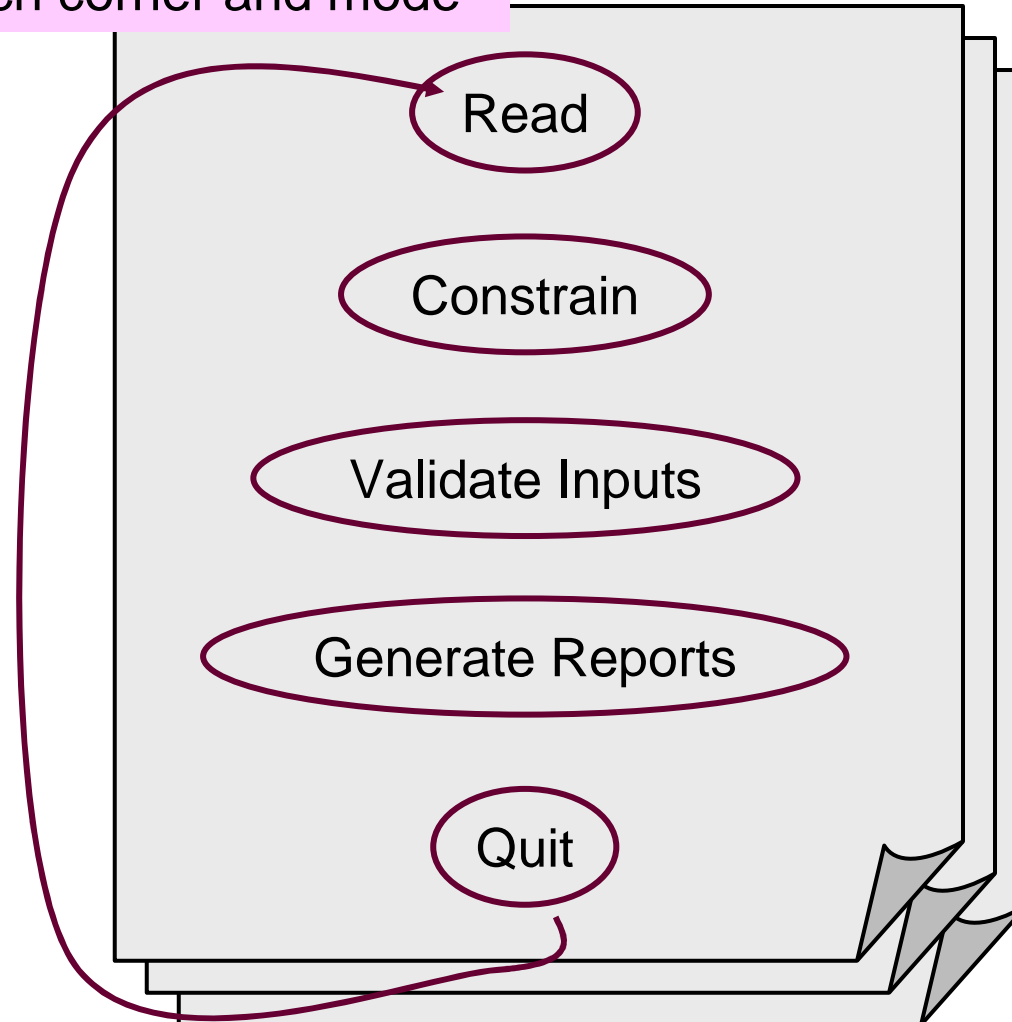


# Required Input Files



# Components of a Master Run Script

Each corner and mode



# Read and Constrain

```
# Comment scripts

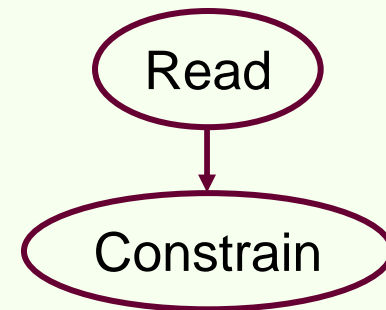
# Include all libraries - technology and IP model libraries
set link_path "*" my_tech_lib.db memory_lib.db"

# Read all gate-level design files
read_verilog my_full_chip.v

# Read libraries and link the design
link_design MY_FULL_CHIP

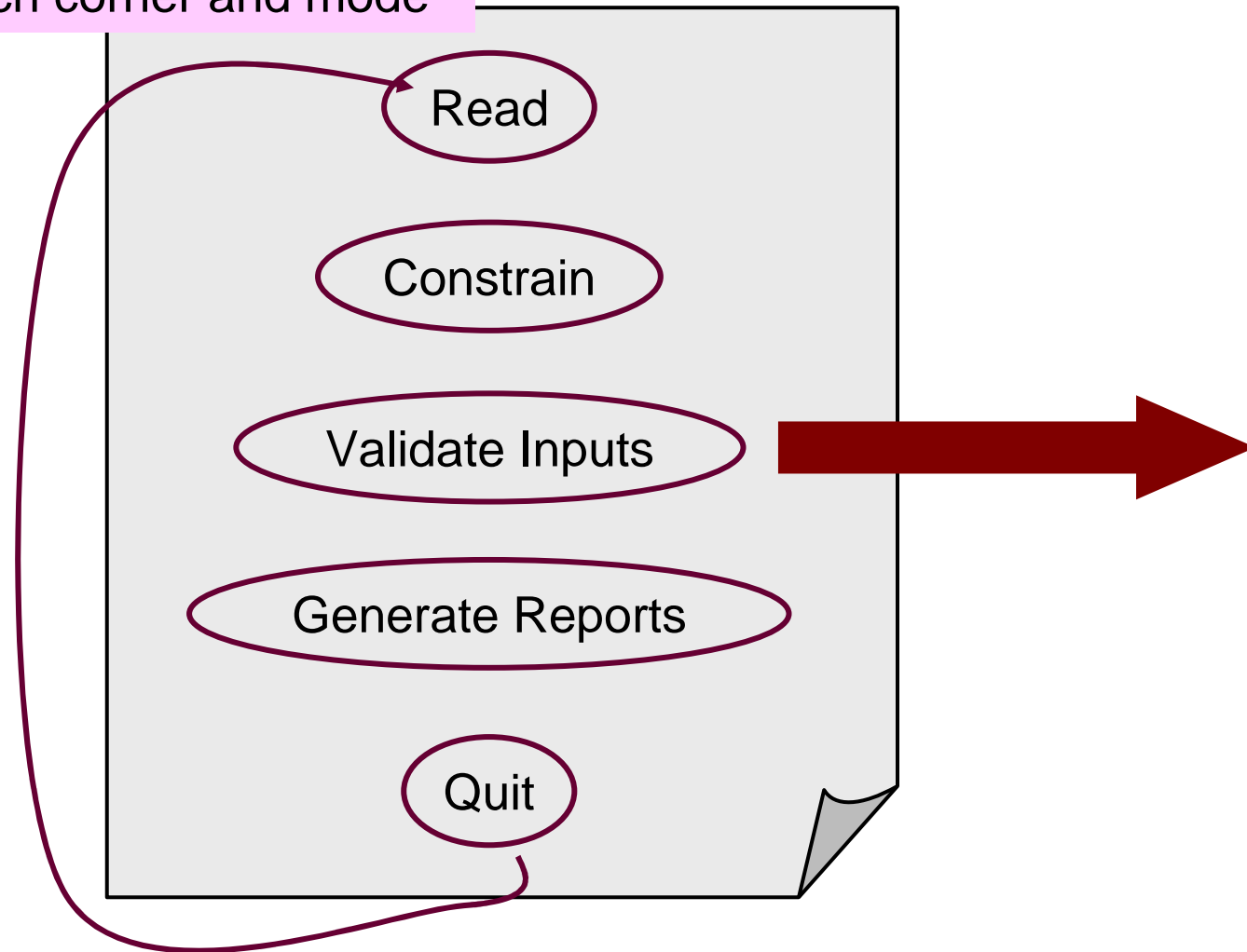
# Set up bc_wc analysis with 2 SDF. Wait for checks later
read_sdf -analysis_type bc_wc -max_type sdf_max -min_type sdf_min

# Apply chip-level constraints for pre or post layout analysis
source MY_FULL_CHIP_CONST.tcl
```



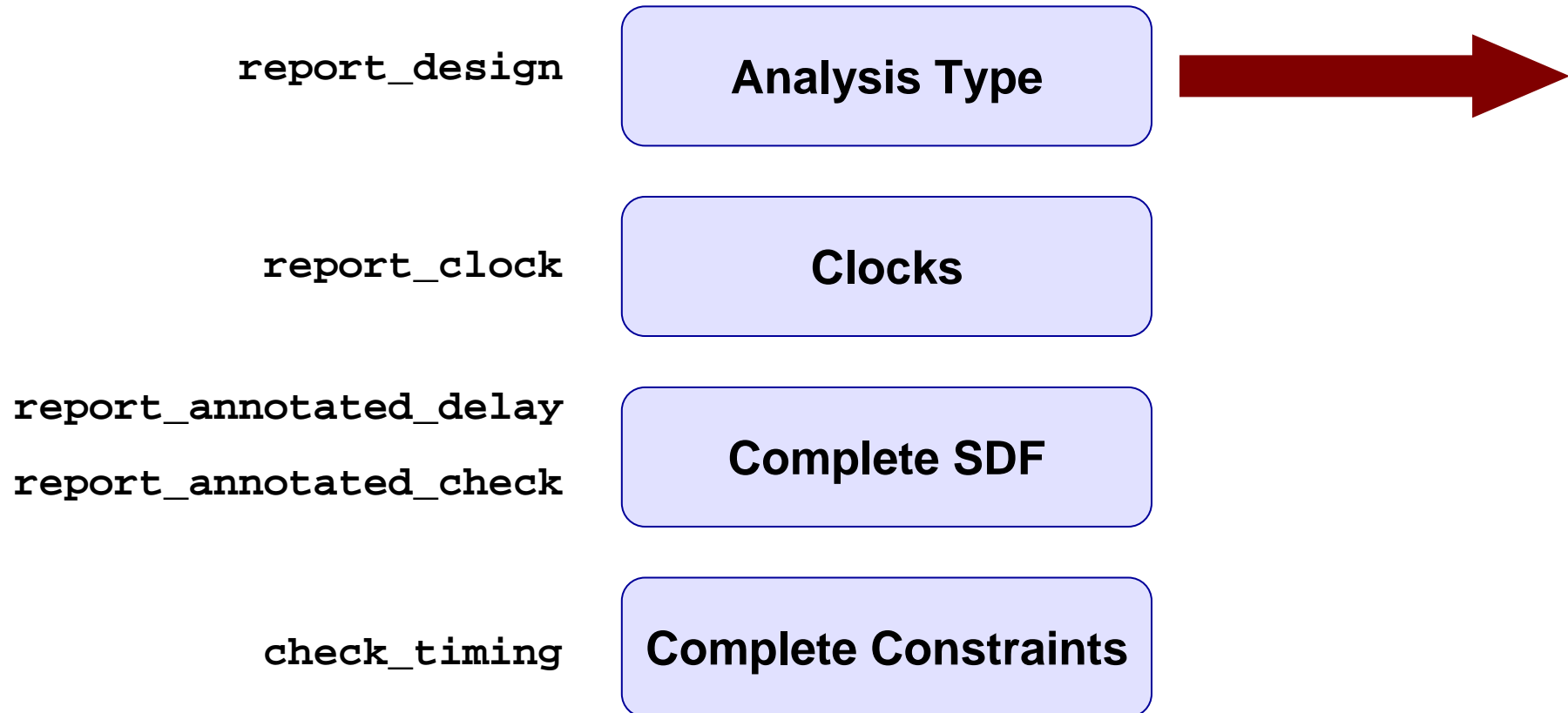
# Recall: Components of a Master Run Script

Each corner and mode





# Validate Complete and Correct Constraints



# Three Types of Analysis

single

Read one SDF delay for setup OR hold analysis

bc\_wc

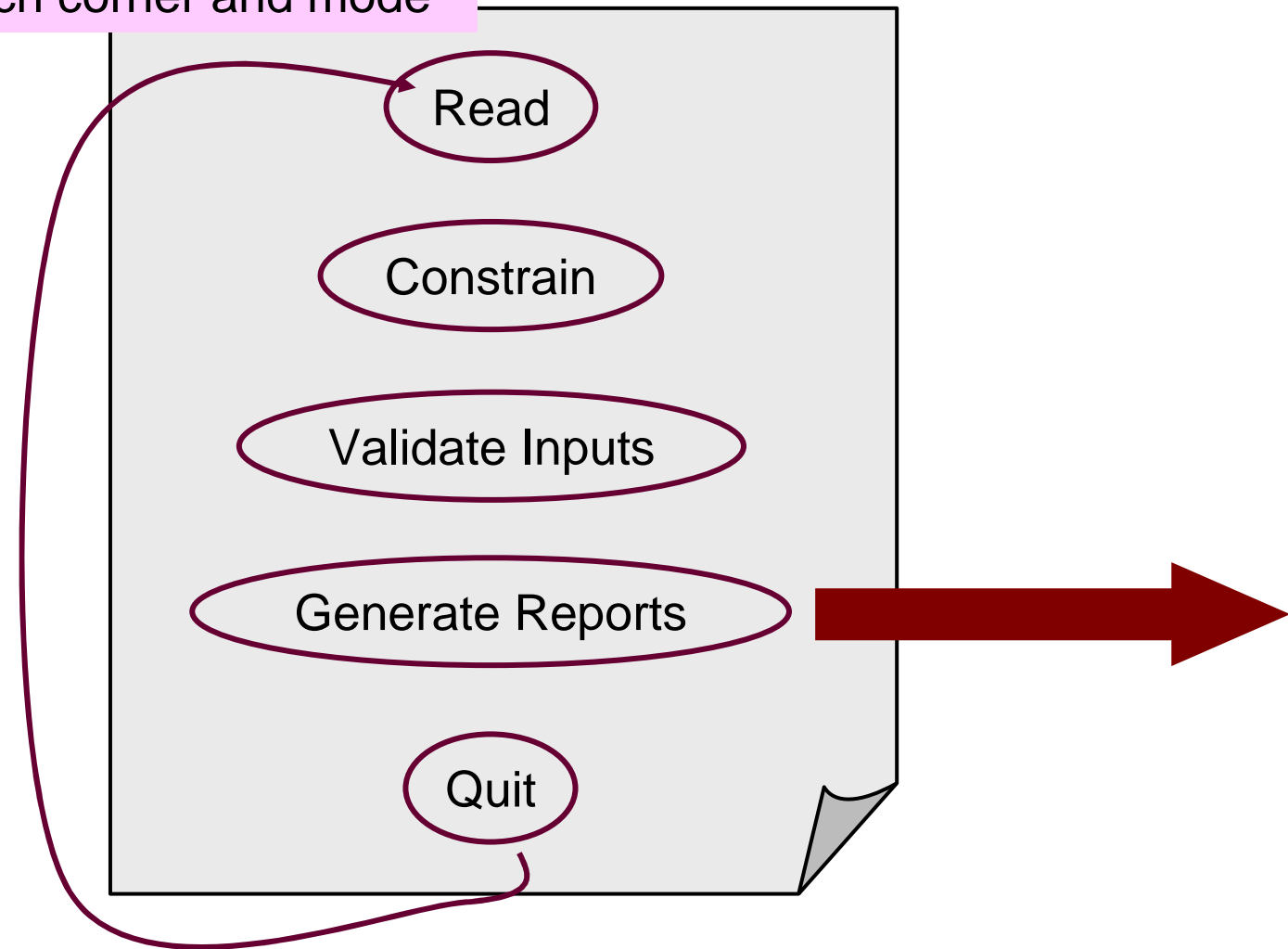
Read two SDF delays for setup and hold analysis

on\_chip\_variation

Min and Max SDF represent a small variation across a die

# Ready to Analyze STA Reports

Each corner and mode



# Report All Violations

```
report_constraint -all_violators
```

```
max_delay/setup ('Clk1' group)
```

Endpoint	Slack
-----	
B	-0.50 (VIOLATED)

```
min_delay/hold ('Clk1' group)
```

Endpoint	Slack
-----	
FF1/D0	-0.67 (VIOLATED)

```
sequential_clock_pulse_width
```

Pin	Required pulse width	Actual pulse width	Slack
-----			
FF2/clock (high)	0.90	0.85	-0.05 (VIOLATED)

# The Number of Violations

**report\_analysis\_coverage**

Type of Check	Total	Met	Violated	Untested
setup	6724	2366 ( 35%)	0 ( 0%)	4358 ( 65%)
hold	6732	2366 ( 35%)	0 ( 0%)	4366 ( 65%)
recovery	362	302 ( 83%)	0 ( 0%)	60 ( 17%)
removal	354	302 ( 85%)	0 ( 0%)	52 ( 15%)
min_pulse_width	4672	4310 ( 92%)	0 ( 0%)	362 ( 8%)
clock_gating_setup	65	65 (100%)	0 ( 0%)	0 ( 0%)
clock_gating_hold	65	65 (100%)	0 ( 0%)	0 ( 0%)
out_setup	138	138 (100%)	0 ( 0%)	0 ( 0%)
out_hold	138	74 ( 54%)	64 ( 46%)	0 ( 0%)
All Checks	19250	9988 ( 52%)	64 ( 0%)	9198 ( 48%)

# More Details: Path Timing Reports

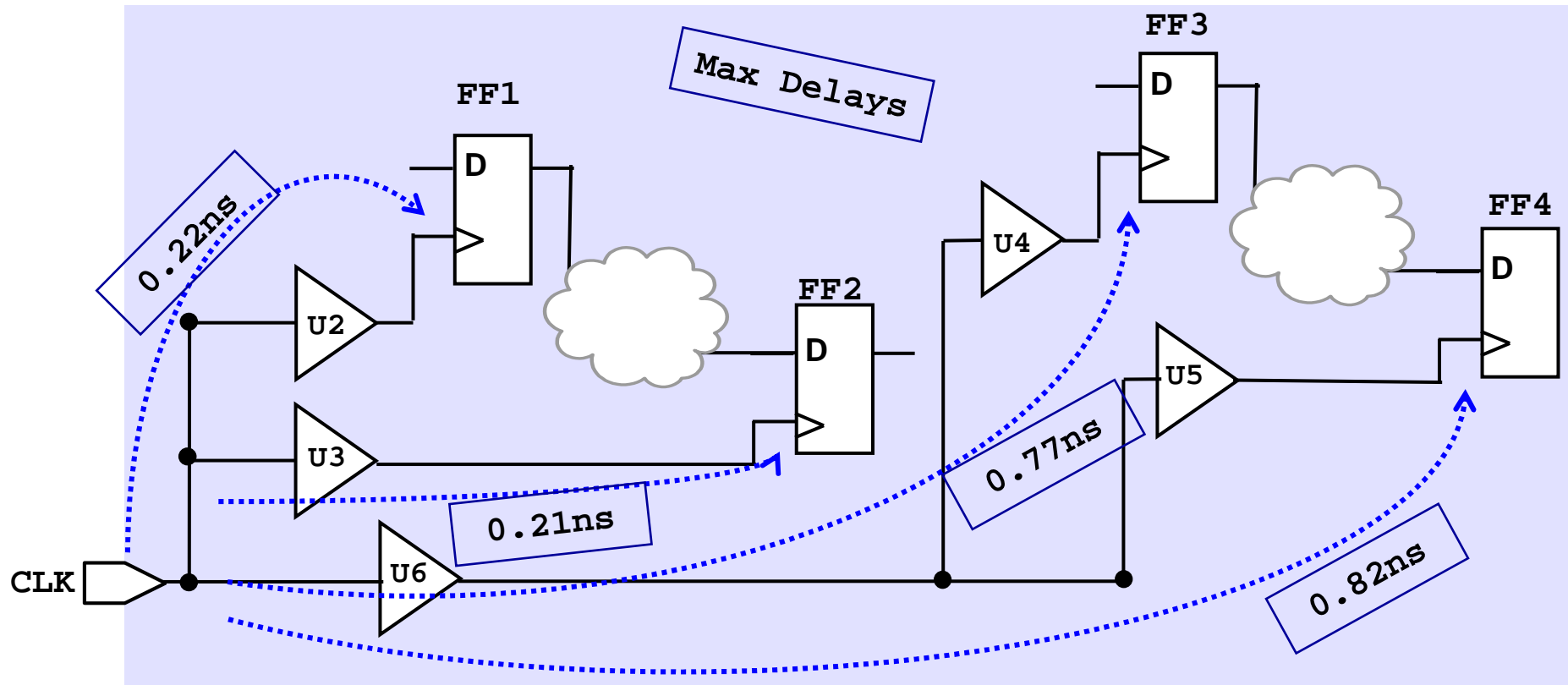
```
pt_shell> report_timing
```

- **Default: Returns the worst path for max analysis for:**
  - Each clock
  - Recovery checks
  - Clock gating checks
- **Customize with MANY different switches:**
  - Setup versus hold reports
  - Increase the significant digits
  - Focus on specific paths
  - Increase the # of generated reports
  - Include net fanout
  - Expand the calculated clock network delay

# Clock Network Reports

```
report_clock_timing -type skew
```

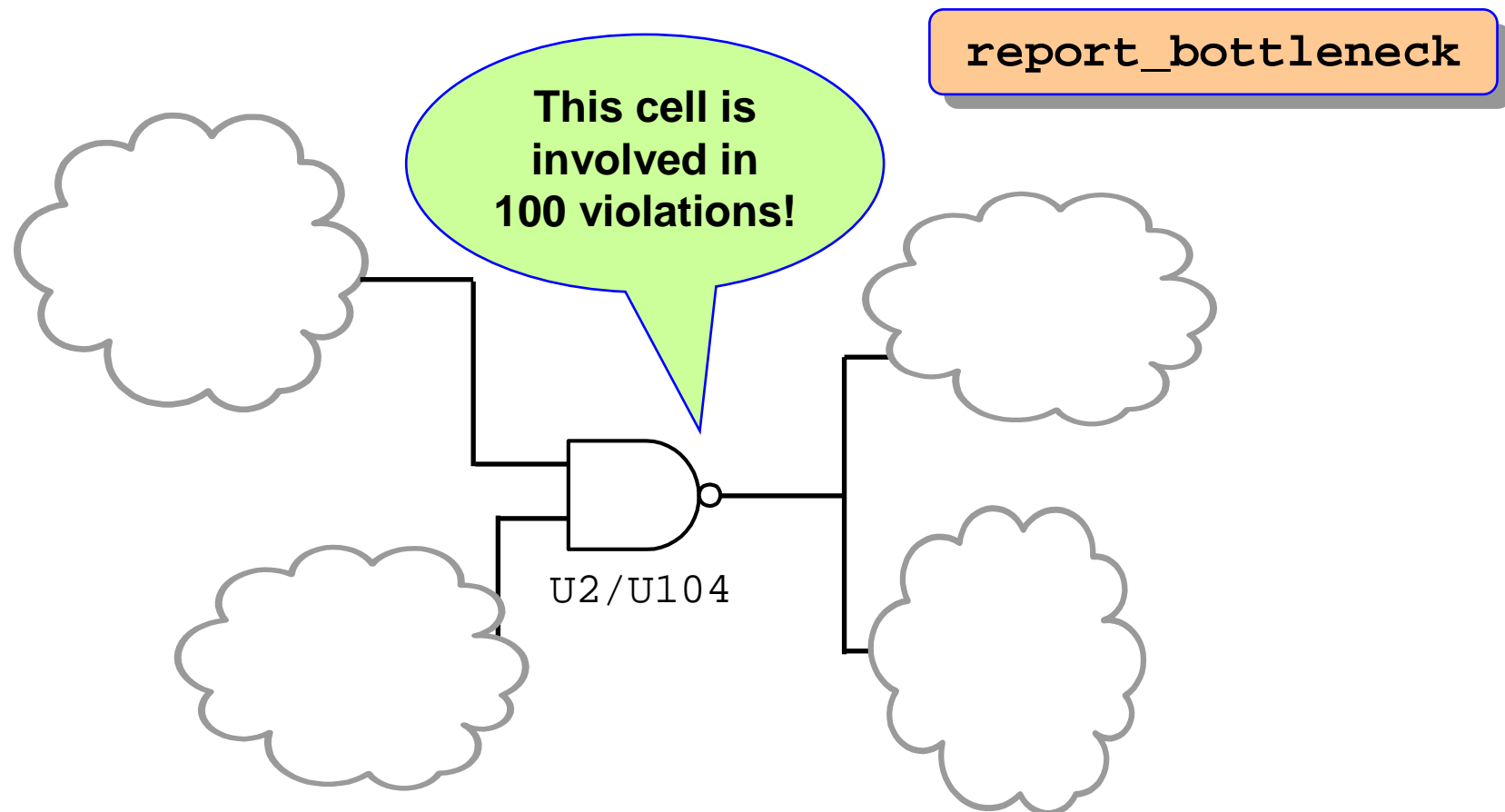
For each clock, report  
**REAL** skew



# Bottleneck Analysis

**Identify cells involved in multiple violations.**

Use the results to determine cells to buffer or upsize.





# Specify Timing Assertions (1)

---

- **Example:**

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

```
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 -max 5.5 [get_clocks CLK]
pt_shell> set_clock_transition -min 0.25 [get_clocks CLK]
pt_shell> set_clock_transition -max 0.3 [get_clocks CLK]
```

- For post layout clock tree:

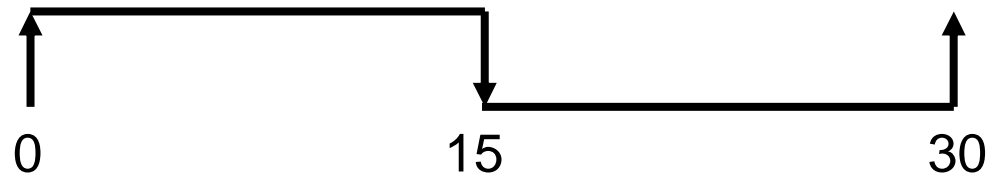
`set_propagated_clock <clock_object_list>`

or

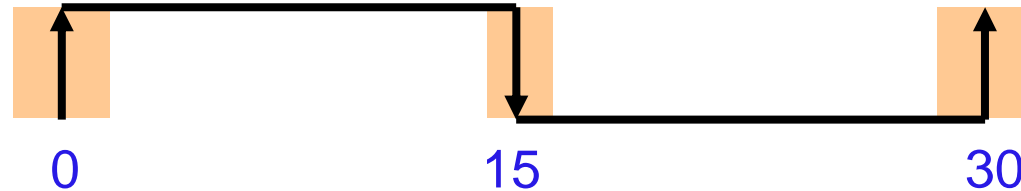
`set timing_all_clocks_propagated true`

# Specify Timing Assertions (2)

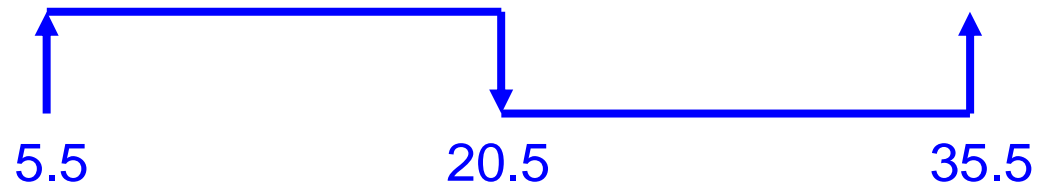
Reference clock waveform



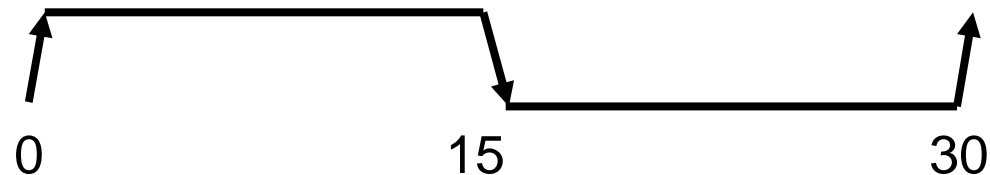
Reference clock waveform with uncertainty



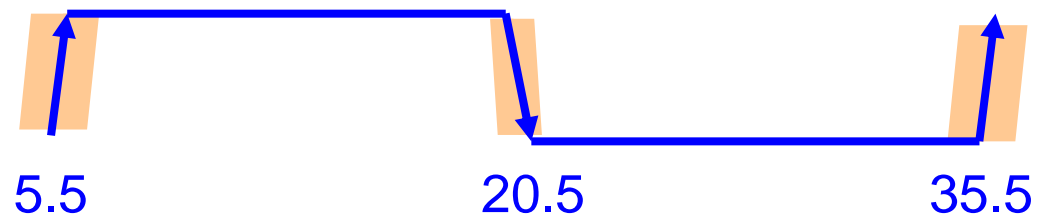
Reference clock waveform with latency



Reference clock waveform with transition



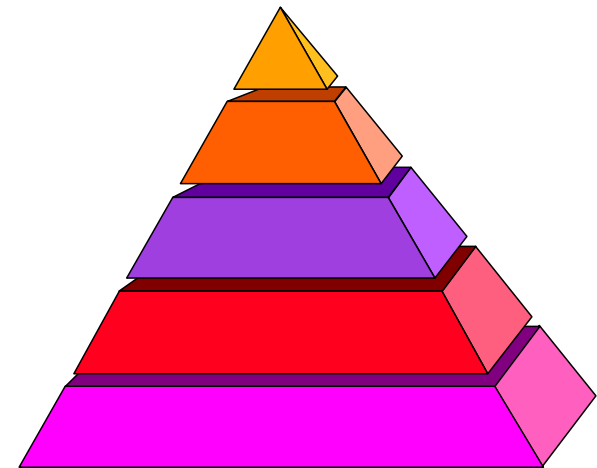
Reference clock waveform with uncertainty, latency, and transition



# Advanced Timing Analysis

---

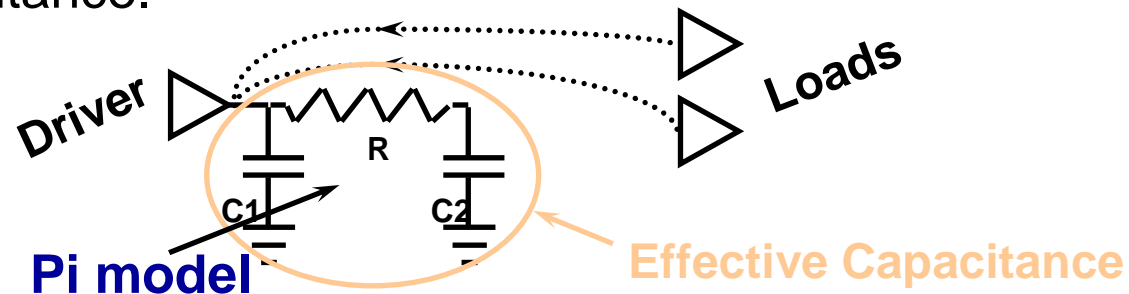
- Analysis Modes
- Data to Data Checks
- Case Analysis
- Multiple Clocks per Register
- Minimum Pulse Width Checks
- Derived Clocks
- Clock Gating Checks
- Netlist Editing
- Report\_clock\_timing
- Clock Reconvergence Pessimism
- Worst-Arrival Slew Propagation
- Debugging Delay Calculation



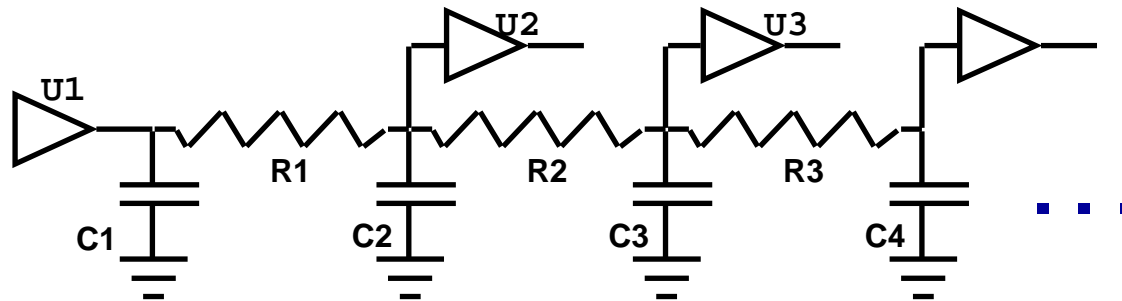
# Back-Annotation - Parasitics

## Reduced and Distributed Parasitic Files

- *Reduced* format annotates an RC pi model, and computes the effective capacitance.



- *Distributed* format enables PrimeTime to annotate each physical segment of the routed netlist (most accurate form of RC back-annotation)



# PrimeTime Timing Models Support

PrimeTime offers the following timing models to address STA needs for IP, large hierarchical designs, and custom design:

- **Quick Timing Model (QTM)**
- **Extracted Timing Model (ETM)**
- **Interface Logic Model (ILM)**
- **Stamp Model**

# Timing Model Usage Scenario in PrimeTime

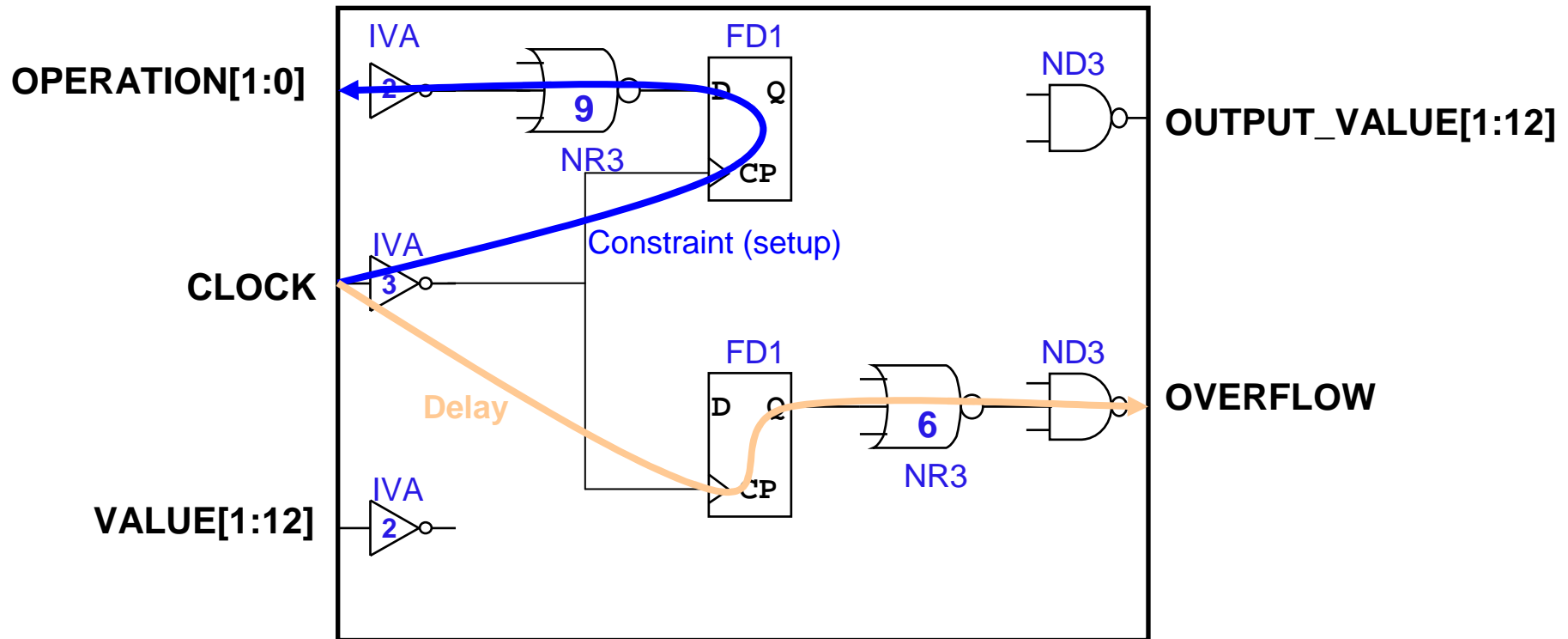
Usage Scenario	Appropriate Model
Top-Down Design	Quick Timing Models
IP Reuse	ETMs
Interface to non-STA and 3rd party tools	ETMs
Synthesis Tasks	ILMs / ETMs
Chip-Level STA	ILMs
Memory and Datapath	Stamp Models

# Quick Timing Models (QTM)s

---

- Provide means to quickly and easily create a timing model of an unfinished block for performing timing analysis
- Should later be replaced with gate-level netlists or equivalent models
- Created with PrimeTime commands - *no compiling needed!*
- Can contain:
  - **Port specs for the block**
  - **Setup and hold constraints for inputs**
  - **Clock-to-output delays**
  - **Input-to-output delays**
- Benefits
  - **accurate specs generated with a lot less effort**
  - **apply chip level timing constraints and time the whole design**
  - **discover violators up front**

# Quick Timing Models - What are they?



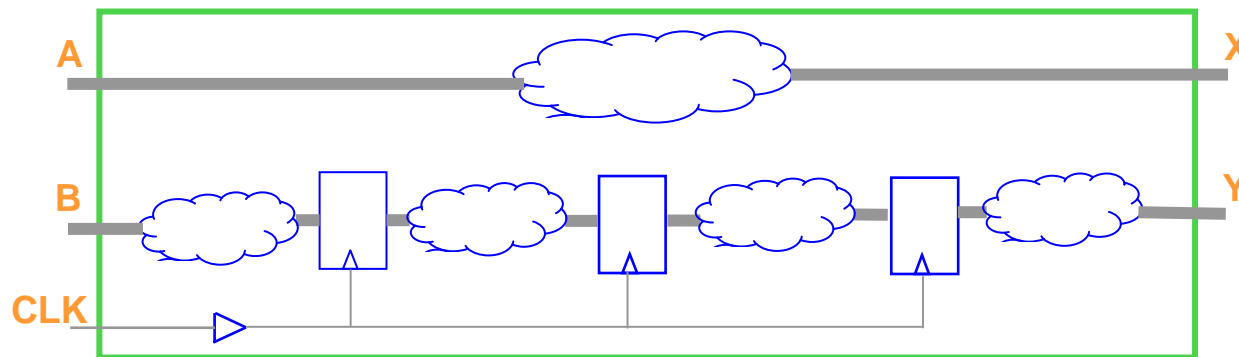
- QTM is a set of interactive PrimeTime commands - not a language
- Like all PrimeTime commands, QTM can be saved in a script
- QTM model can be saved in db or Stamp format



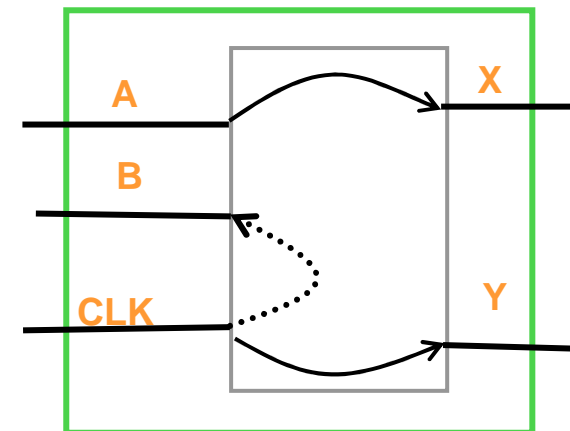
# Extracted Timing Models (ETM)

- Enable IP Reuse and interchange of timing models between EDA tools
- Compact black-box timing models
  - » contain timing arcs between external pins
  - » Internal pins only for generated/internal clocks
  - » models written out in Stamp, .lib ,or db formats
  - » context independent
  - » Exceptions and latches supported
  - » Provide huge performance improvements

*Design*

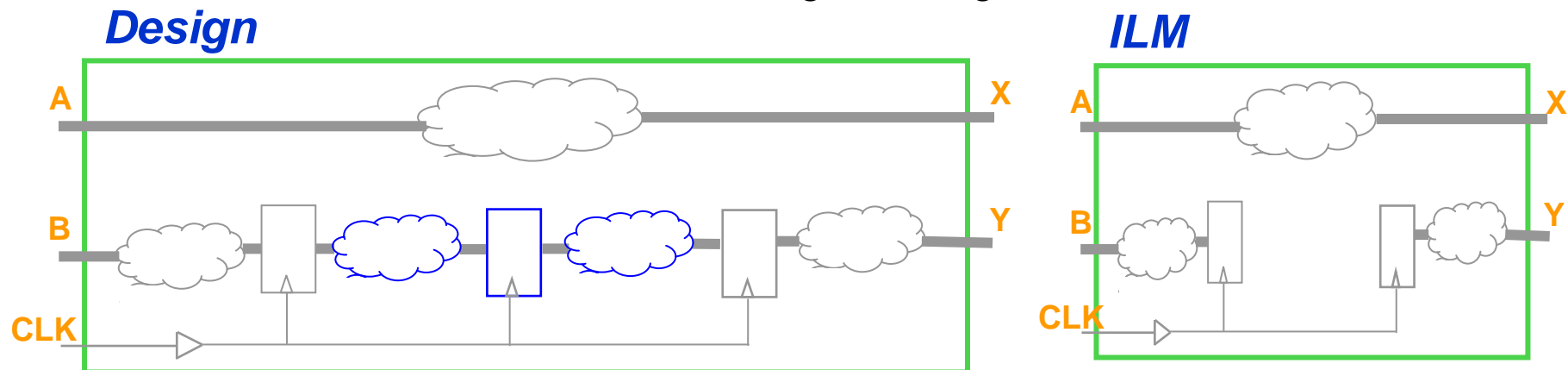


*ETM*



# Interface Logic Models (ILM)

- Enable Hierarchical STA
  - Reduce memory and CPU usage for chip-level analysis
  - Offer big netlist reduction if block IOs are registered
  - Back-annotation and constraint files for interface logic are written out along with netlist
- Benefits:
  - High accuracy because interface logic is not abstracted
  - Fast model generation time
  - Context independent
    - ◆ Can change load, drive, operating conditions, parasitics, SDF, constraints without re-generating the model



# Interface Logic Models (ILM)

---

- ILMs can be used in SDF and parasitics based flows

```
pt_shell> write_ilm[sdf/parasitics] <output_file>
```

- Support for Hierarchical SI analysis

```
pt_shell> create_ilm -include {xtalk_pins}
```

- Support for Model Validation

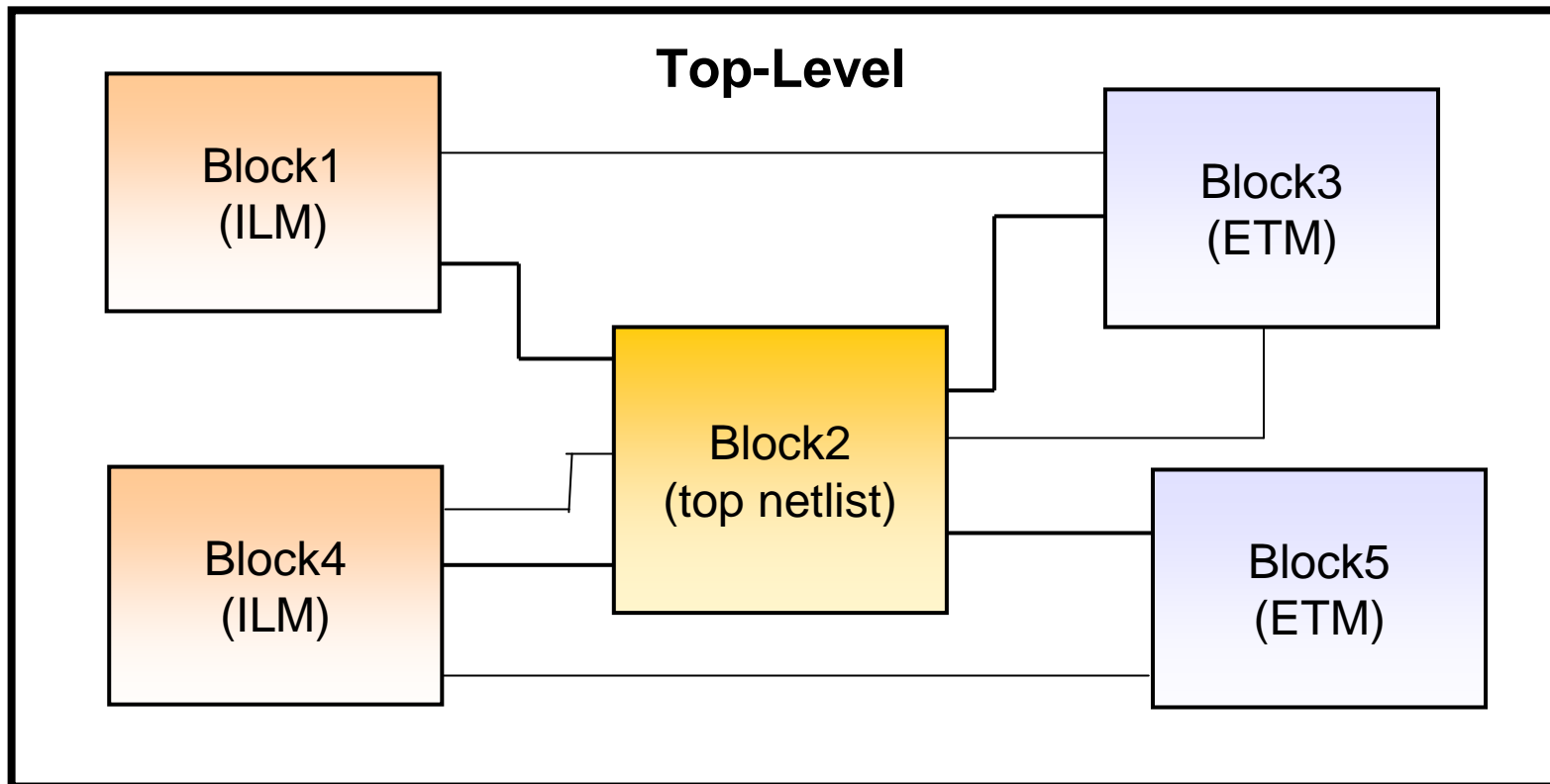
```
pt_shell> compare_interface_timing <ref_file> <cmp_file>  
-slack 0.2 -include slack
```

# Stamp Modeling

---

- **Generally created for transistor-level designs, where there is no gate-level netlist. Stamp timing models are usually created by core or technology vendors, as a compiled db.**
- **Capabilities include the ability to model:**
  - ◆ pin-to-pin timing arcs
  - ◆ setup and hold data
  - ◆ pin capacitance and drive
  - ◆ mode information
  - ◆ tri-state outputs
  - ◆ internally generated clocks
- **Stamp models co-exist with the Library Compiler .lib models**

# Chip-Level Verification using Models



- Using ILMs and ETMs to address capacity and timing issues in multi-million gate design

# Does Your Design Meet Timing?

```
pt_shell> report_analysis_coverage
```

Type of Check	Total	Met	Violated	Untested
setup	6724	5366 ( 80%)	0 ( 0%)	1358 ( 20%)
hold	6732	5366 ( 80%)	0 ( 0%)	1366 ( 20%)
recovery	362	302 ( 83%)	0 ( 0%)	60 ( 17%)
removal	354	302 ( 85%)	0 ( 0%)	52 ( 15%)
min_pulse_width	4672	4310 ( 92%)	0 ( 0%)	362 ( 8%)
clock_gating_setup	65	65 (100%)	0 ( 0%)	0 ( 0%)
clock_gating_hold	65	65 (100%)	0 ( 0%)	0 ( 0%)
out_setup	138	138 (100%)	0 ( 0%)	0 ( 0%)
out_hold	138	74 ( 54%)	<b>64 ( 46%)</b>	0 ( 0%)
All Checks	19250	15988 ( 84%)	64 ( 0%)	3198 ( 16%)

# Are You Finished?



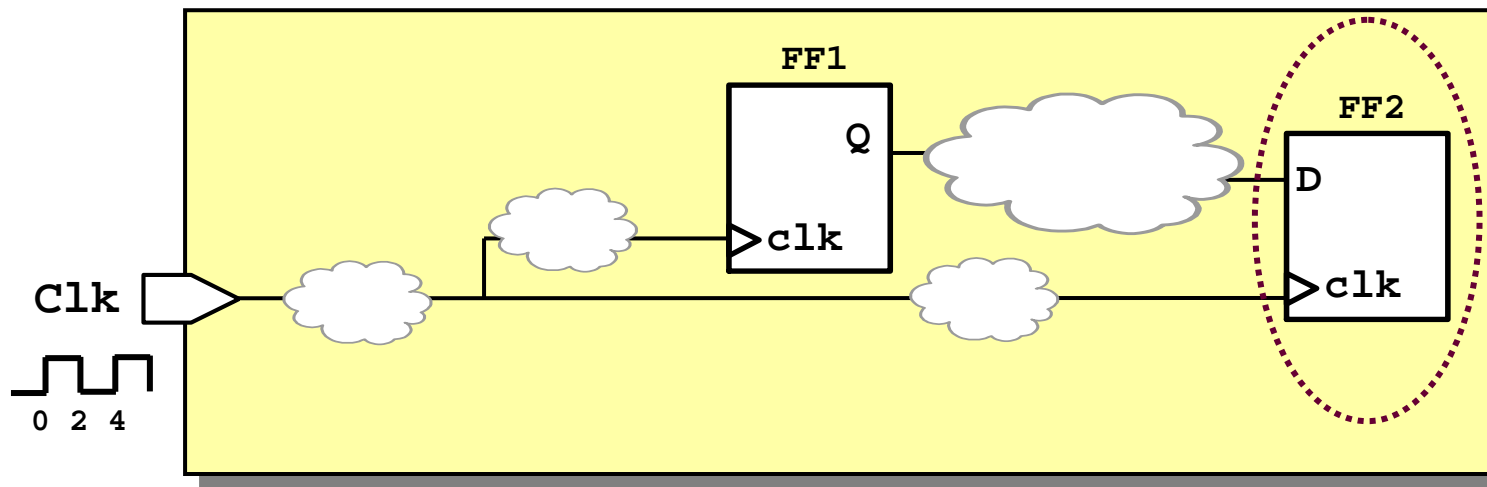
**When PrimeTime was run it revealed  
64 violations in the design.**

What else is there?

- **Are the violations real?**
- **Can you explain warnings in the log files?**
- **What are your suggestions for resolution?**
- **You have a special situation – what are the issues?**

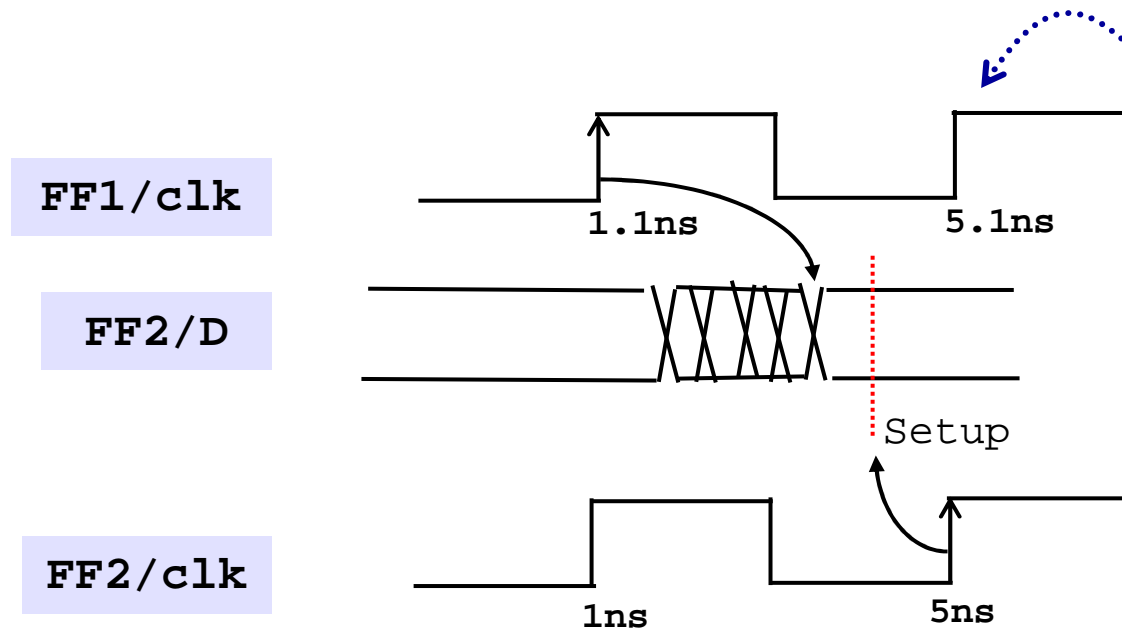
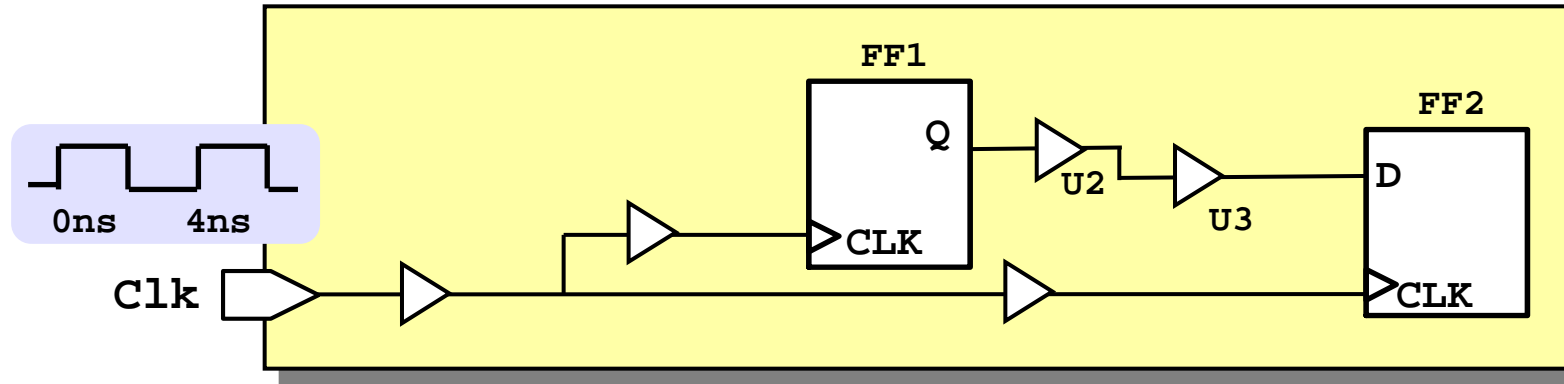
# Timing Verification of Synchronous Designs

All “registers” must reliably capture data at the desired clock edges.





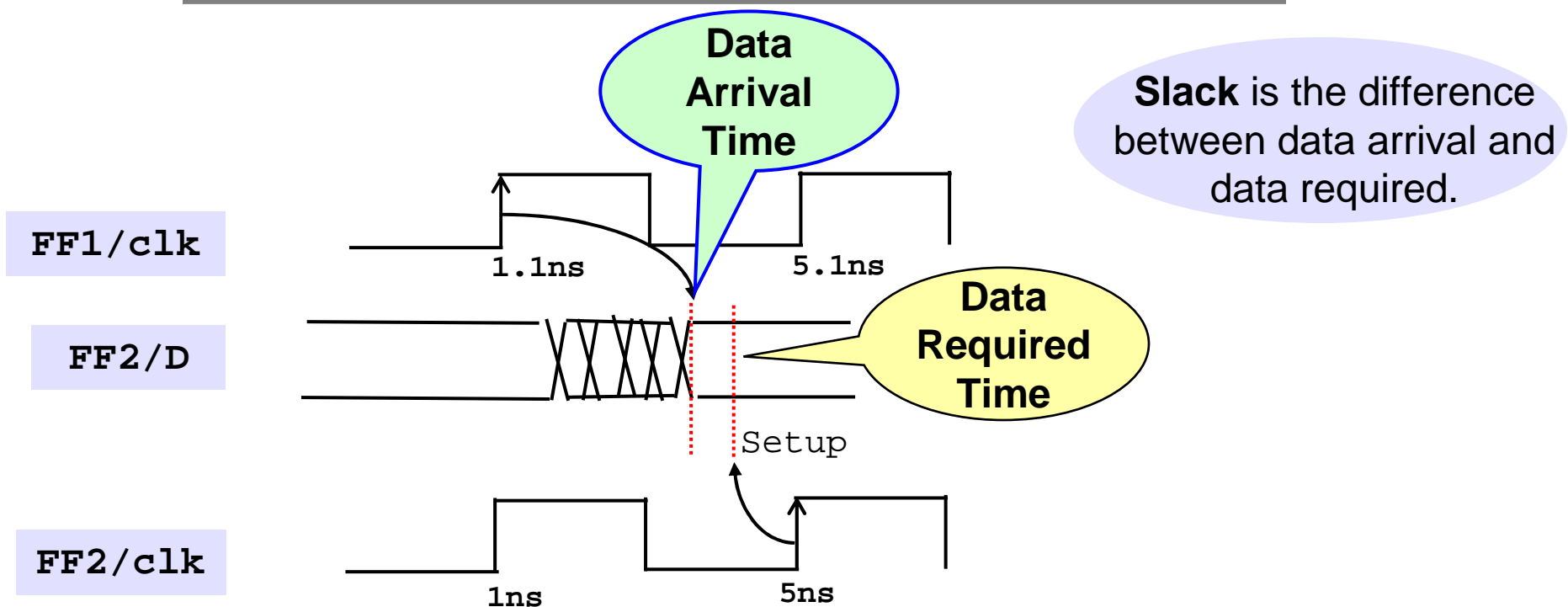
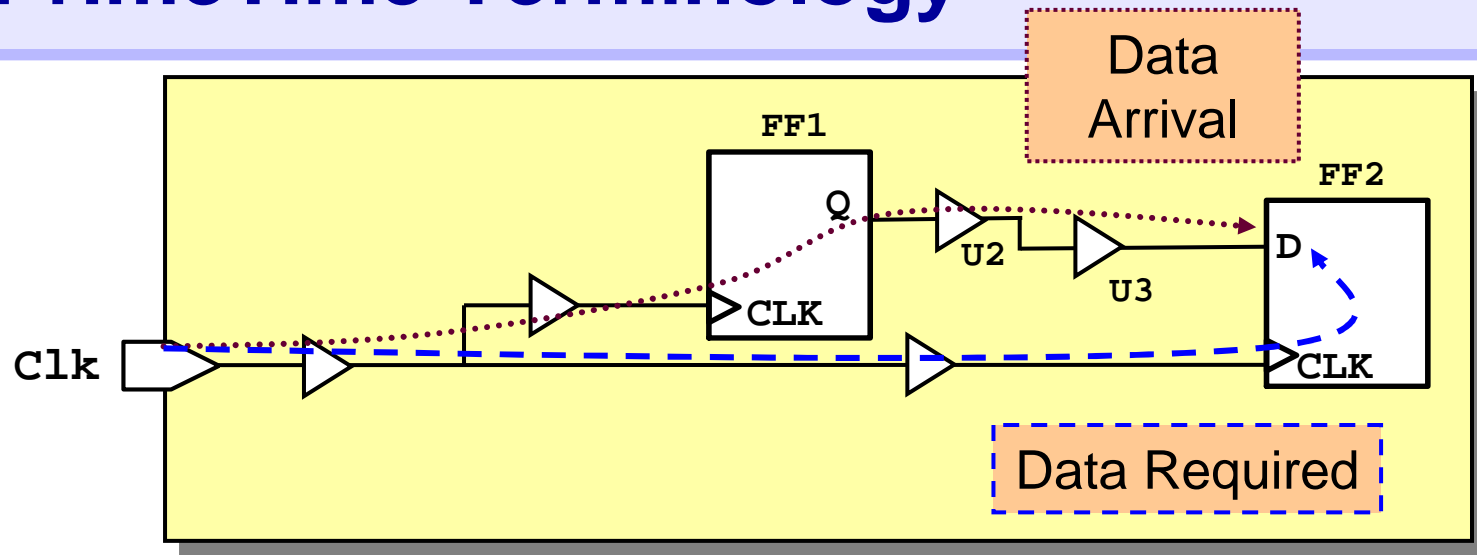
# Static Timing Verification of FF2: Setup



Where does this 1.1ns shift come from?

Why is the shift different here?

# PrimeTime Terminology



# Four Sections in a Timing Report

## report\_timing

### Header

Startpoint: FF1 (rising edge-triggered flip-flop clocked by Clk)  
Endpoint: FF2 (rising edge-triggered flip-flop clocked by Clk)  
Path Group: Clk  
Path Type: max

### Data arrival

Point	Incr	Path
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.10 *	1.10
FF1/CLK (fdef1a15)	0.00	1.10 r
FF1/Q (fdef1a15)	0.50 *	1.60 r
U2/Y (buf1a27)	0.11 *	1.71 r
U3/Y (buf1a27)	0.11 *	1.82 r
FF2/D (fdef1a15)	0.05 *	1.87 r
data arrival time		1.87

### Data required

clock Clk (rise edge)	4.00	4.00
clock network delay (propagated)	1.00 *	5.00
FF2/CLK (fdef1a15)		5.00 r
library setup time	-0.21 *	4.79
data required time		4.79

### Slack

data required time	4.79
data arrival time	-1.87
slack (MET)	2.92

# The Header

Header

Startpoint: FF1 (rising edge-triggered flip-flop clocked by Clk)

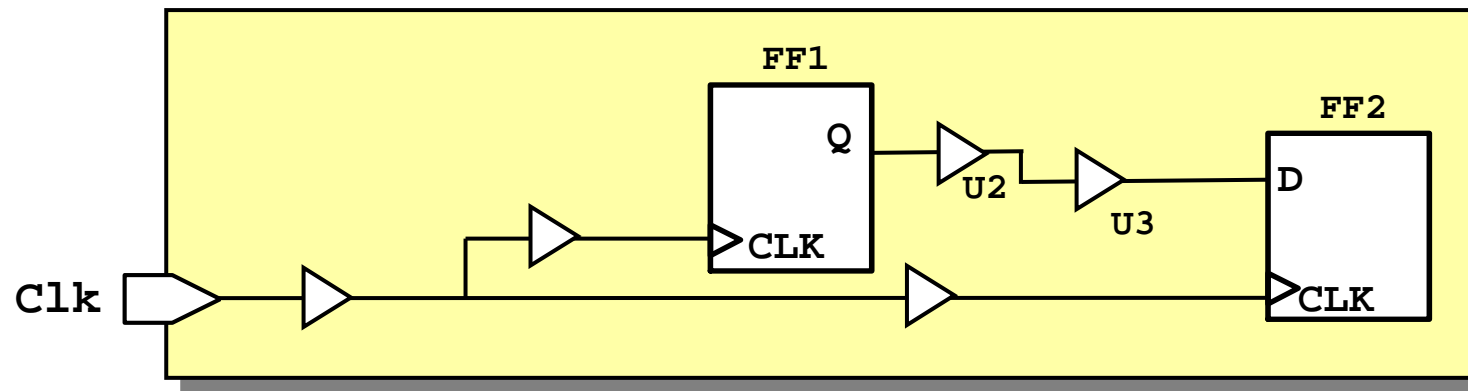
Endpoint: FF2 (rising edge-triggered flip-flop clocked by Clk)

Path Group: Clk

Path Type: max

Capture clock

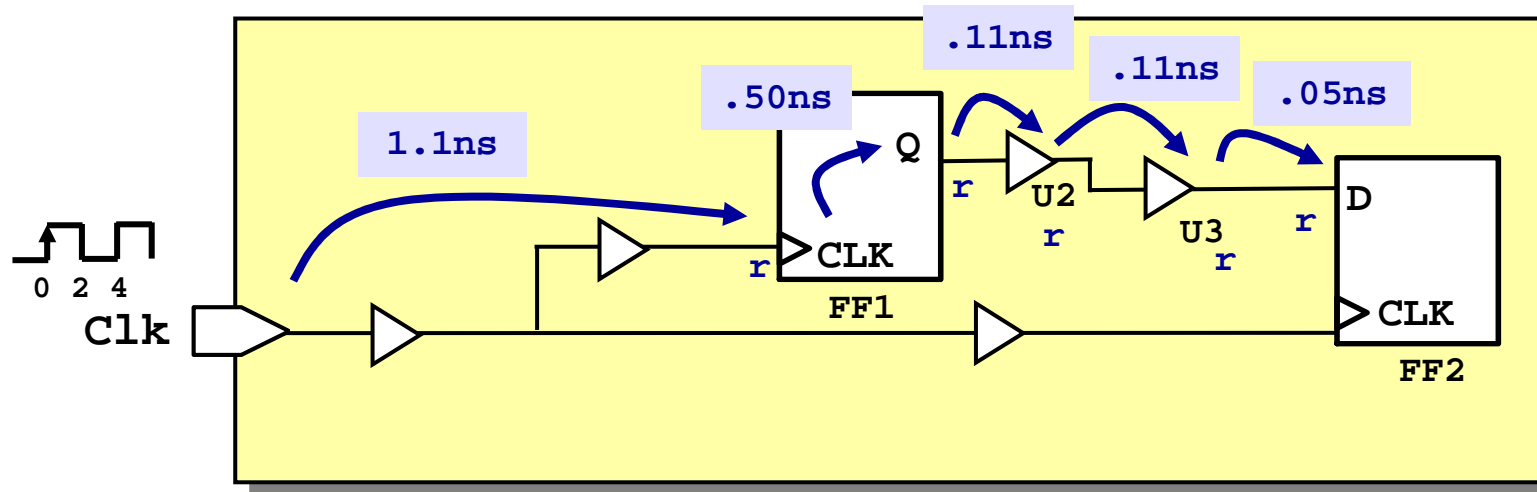
Report is for setup



# Data Arrival Section

Point	Calculated latency	SDF	
		Incr	Path
clock Clk (rise edge)		0.00	0.00
clock network delay (propagated)		1.10 *	1.10
FF1/CLK (fdef1a15)		0.00	1.10 r
FF1/Q (fdef1a15)		0.50 *	1.60 r
U2/Y (buf1a27)	Library reference names	0.11 *	1.71 r
U3/Y (buf1a27)		0.11 *	1.82 r
FF2/D (fdef1a15)		0.05 *	1.87 r
data arrival time			1.87

Data arrival

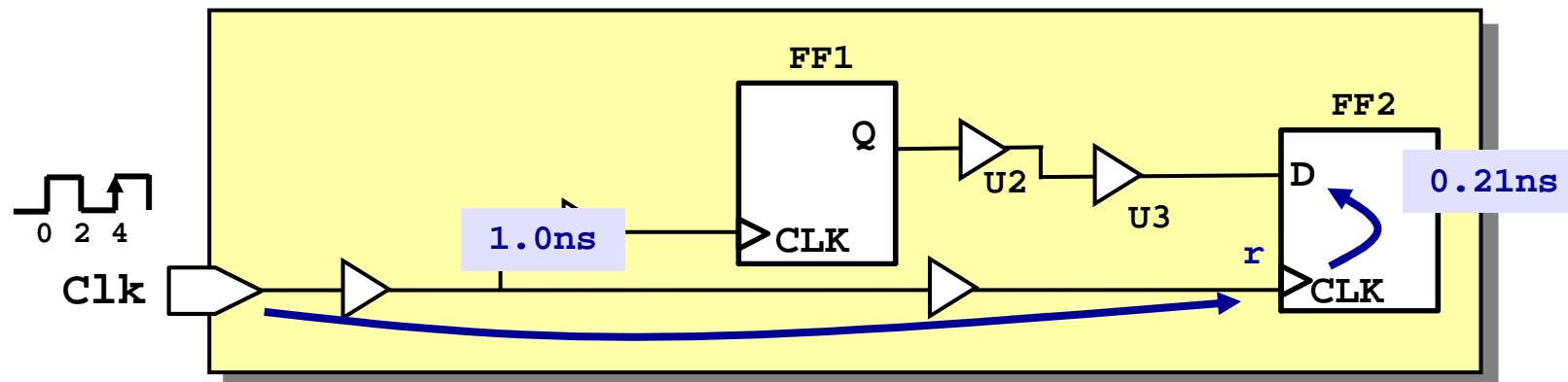


# Data Required Section

Point	Incr	Path
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.10 *	1.10
FF1/CLK (fdef1a15)	0.00	1.10 r
FF1/Q (fdef1a15)	0.50 *	1.60 r
U2/Y (buf1a27)	0.11 *	1.71 r
U3/Y (buf1a27)	0.11 *	1.82 r
FF2/D (fdef1a15)	0.05 *	1.87 r
data arrival time		1.87

Data required	clock Clk (rise edge)	4.00	4.00
	clock network delay (propagated)	1.00 *	5.00
	FF2/CLK (fdef1a15)		5.00 r
	library setup time	-0.21 *	4.79
	data required time		4.79

SDF



# Summary - Slack

## report\_timing

Startpoint: FF1 (rising edge-triggered flip-flop clocked by Clk)

Endpoint: FF2 (rising edge-triggered flip-flop clocked by Clk)

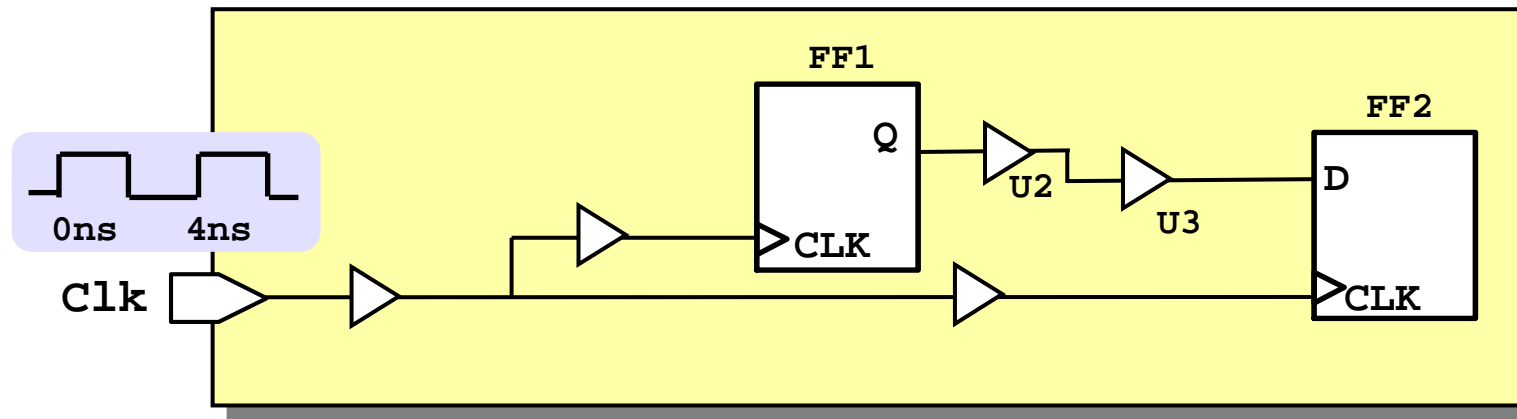
Path Group: Clk

Path Type: max

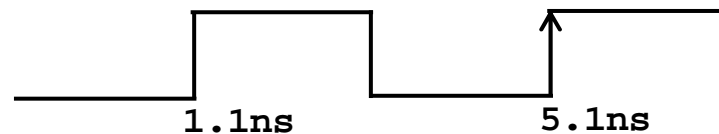
Point	Incr	Path
-----		
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.10 *	1.10
FF1/CLK (fdef1a15)	0.00	1.10 r
FF1/Q (fdef1a15)	0.50 *	1.60 r
U2/Y (buf1a27)	0.11 *	1.71 r
U3/Y (buf1a27)	0.11 *	1.82 r
FF2/D (fdef1a15)	0.05 *	1.87 r
data arrival time		1.87
clock Clk (rise edge)	4.00	4.00
clock network delay (propagated)	1.00 *	5.00
FF2/CLK (fdef1a15)		5.00 r
library setup time	-0.21 *	4.79
data required time		4.79
-----		
data required time		4.79
data arrival time		-1.87
-----		
slack (MET)		2.92

Slack

# Static Timing Verification of FF2: Hold



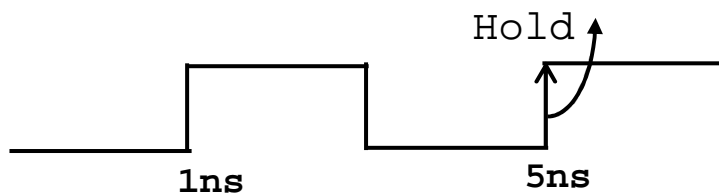
FF1/clk



FF2/D



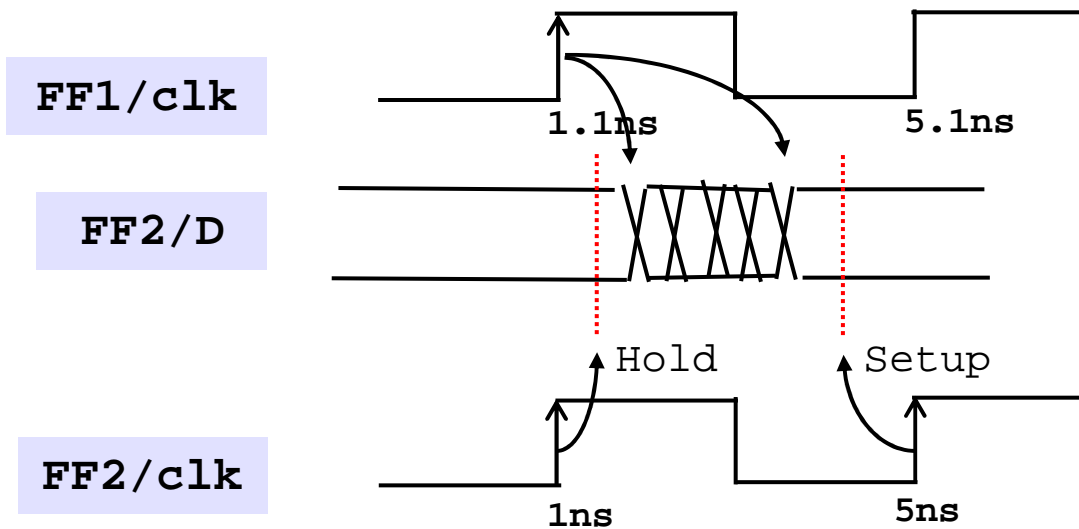
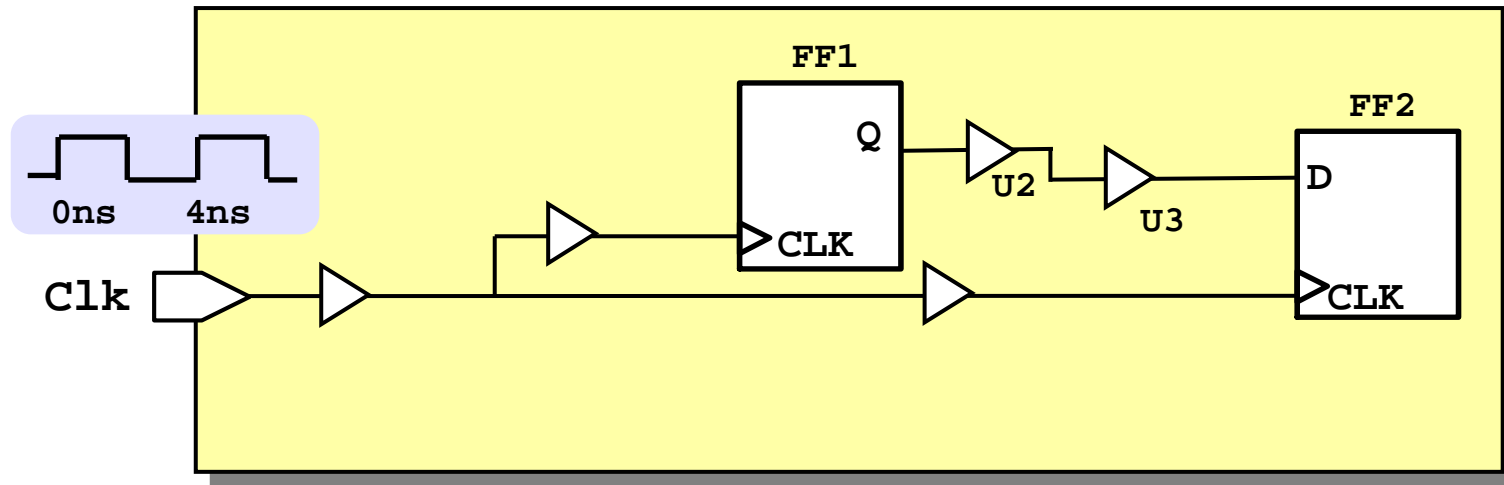
FF2/clk



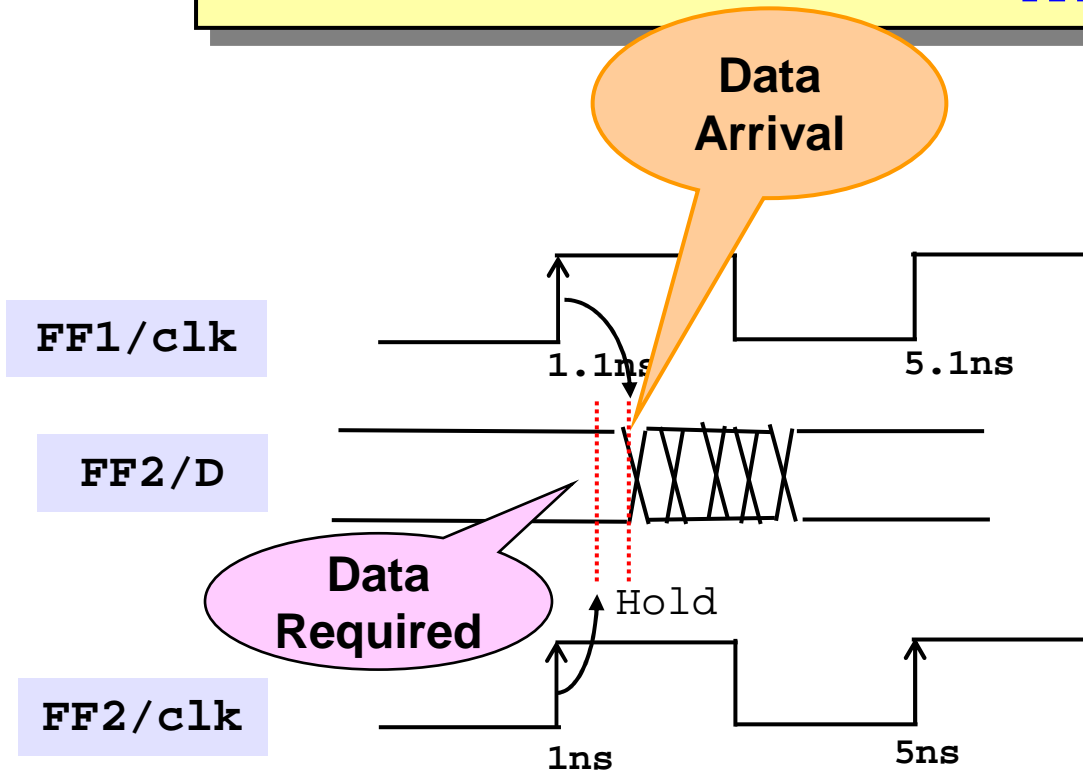
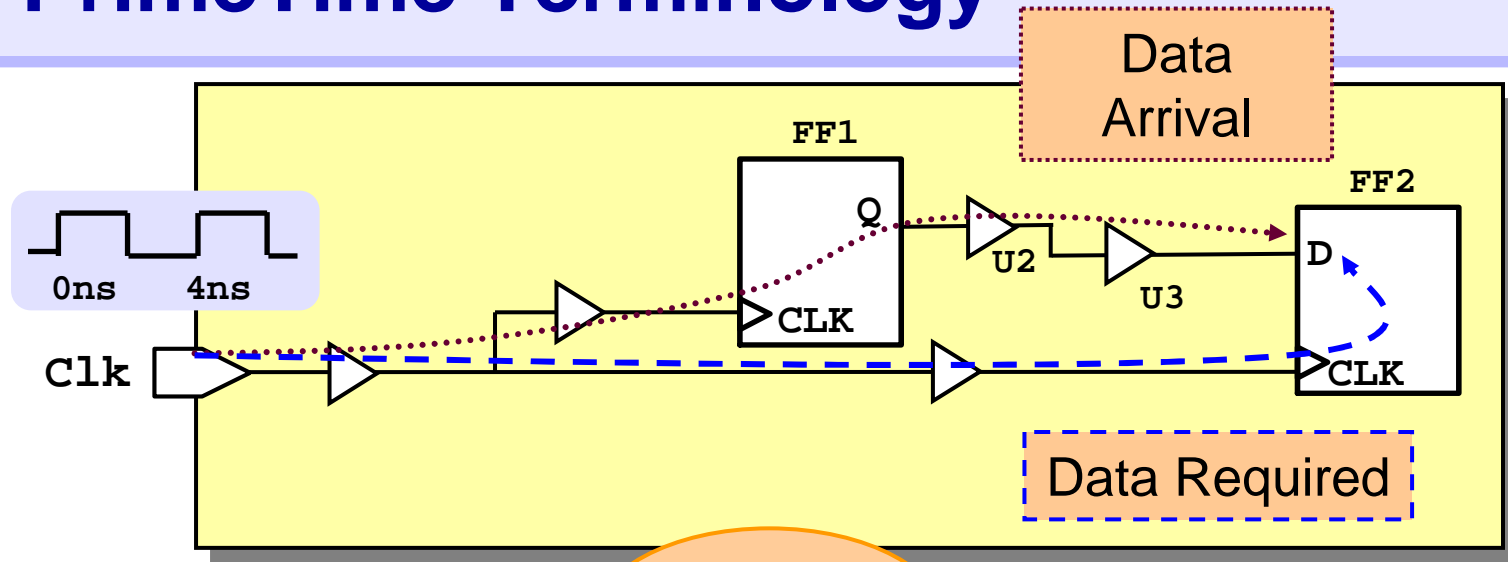
Which clock edge causes the data to change?



# Which Edges are Used in a Timing Report?



# PrimeTime Terminology



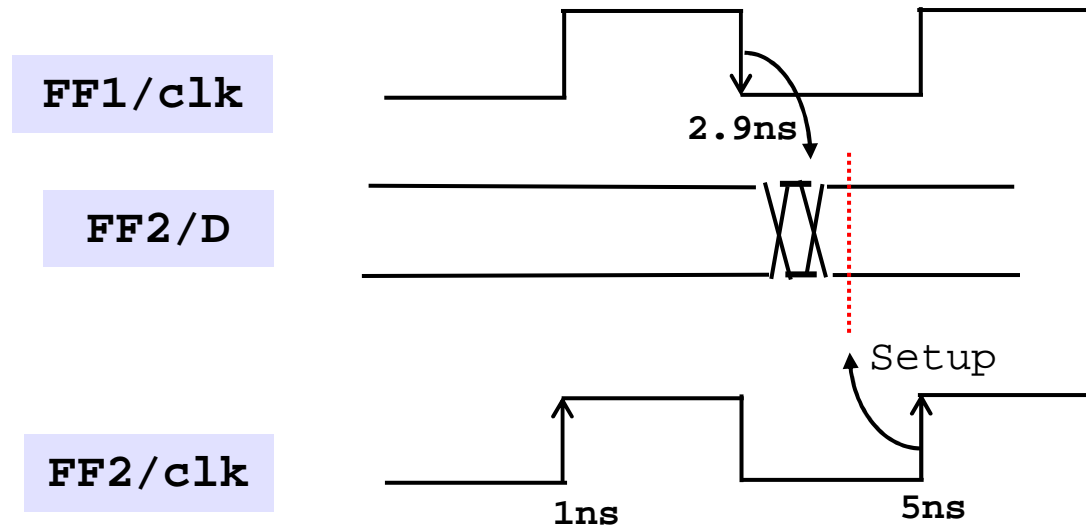
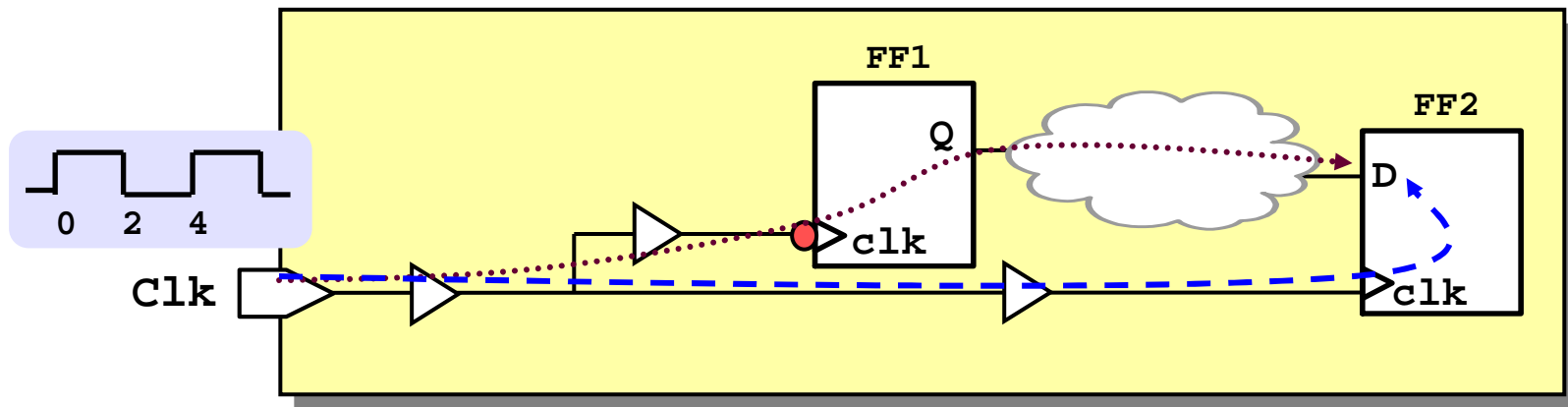
**Slack** is the difference between data arrival and required.

# Example Hold Timing Report

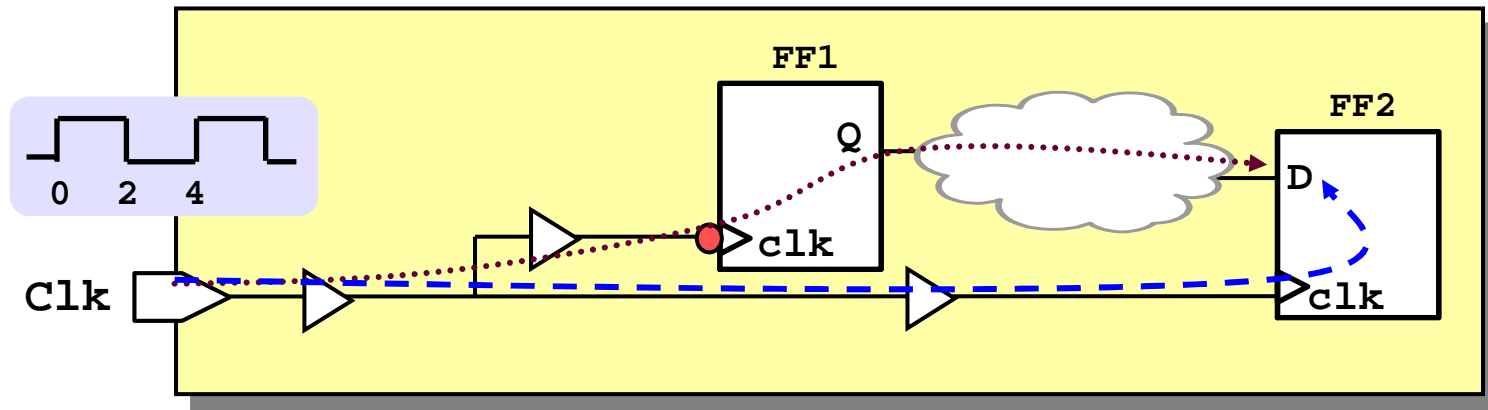
Startpoint: FF1 (rising edge-triggered flip-flop clocked by Clk)  
Endpoint: FF2 (rising edge-triggered flip-flop clocked by Clk)  
Path Group: Clk  
Path Type: min

Point	Incr	Path
-----		
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.10 *	1.10
FF1/CLK (fdef1a15)	0.00	1.10 r
FF1/Q (fdef1a15)	0.40 *	1.50 f
U2/Y (buf1a27)	0.05 *	1.55 f
U3/Y (buf1a27)	0.05 *	1.60 f
FF2/D (fdef1a15)	0.01 *	1.61 f
data arrival time		1.61
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.00 *	1.00
FF2/CLK (fdef1a15)		1.00 r
library hold time	0.10 *	1.10
data required time		1.10
-----		
data required time		1.10
data arrival time		-1.61
-----		
slack (MET)		0.51

# Negedge Triggered Registers: Setup Time



# What About Hold Time?



FF1/clk

2.9ns

6.9ns

FF2/D

STABLE

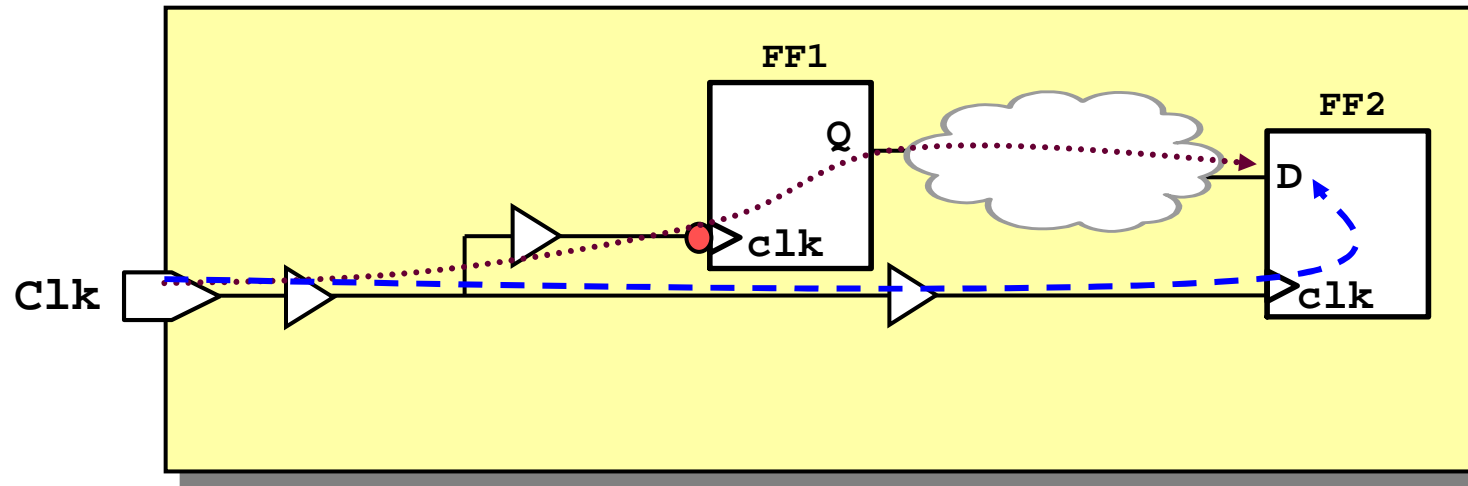
FF2/clk

1ns

5ns

Hold

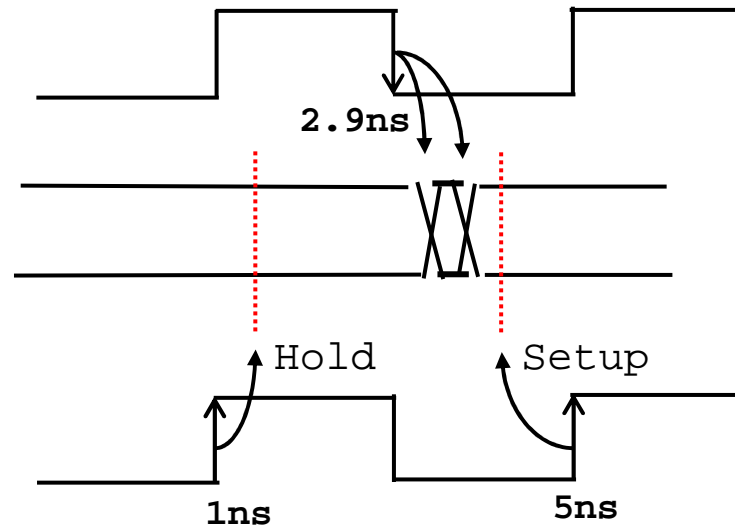
# Which Edges are Used in a Timing Report?



FF1/c1k

FF2/D

FF2/c1k



# Timing Report for Hold

Startpoint: FF1 (falling edge-triggered flip-flop clocked by Clk)  
Endpoint: FF2 (rising edge-triggered flip-flop clocked by Clk)  
Path Group: Clk  
Path Type: min

Point	Incr	Path
-----		
clock Clk (fall edge)	2.00	2.00
clock network delay (propagated)	0.90 *	2.90
FF1/CLK (fdmf1a15)	0.00	2.90 f
FF1/Q (fdef1a15)	0.40 *	3.30 f
U2/Y (buf1a27)	0.05 *	3.35 f
U3/Y (buf1a27)	0.05 *	3.40 f
FF2/D (fdef1a15)	0.01 *	3.41 f
data arrival time		3.41
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.00 *	1.00
FF2/CLK (fdef1a15)		1.00 r
library hold time	0.10 *	1.10
data required time		1.10
-----		
data required time		1.10
data arrival time		-3.41
-----		
slack (MET)		2.31

# Setup Definition - Summary

Data must become valid and stable at least one setup time before being captured by flip-flop.

EQN 1  $\text{Slack}_{\text{setup}} = \text{Data Required Time} - \text{Data Arrival Time} \geq 0$

EQN 2  $\text{Slack}_{\text{setup}} = (T_{\text{capture}} - t_{\text{setup}}) - (T_{\text{launch}} + t_{\text{prop}}) \geq 0$

Clk Spec

Library

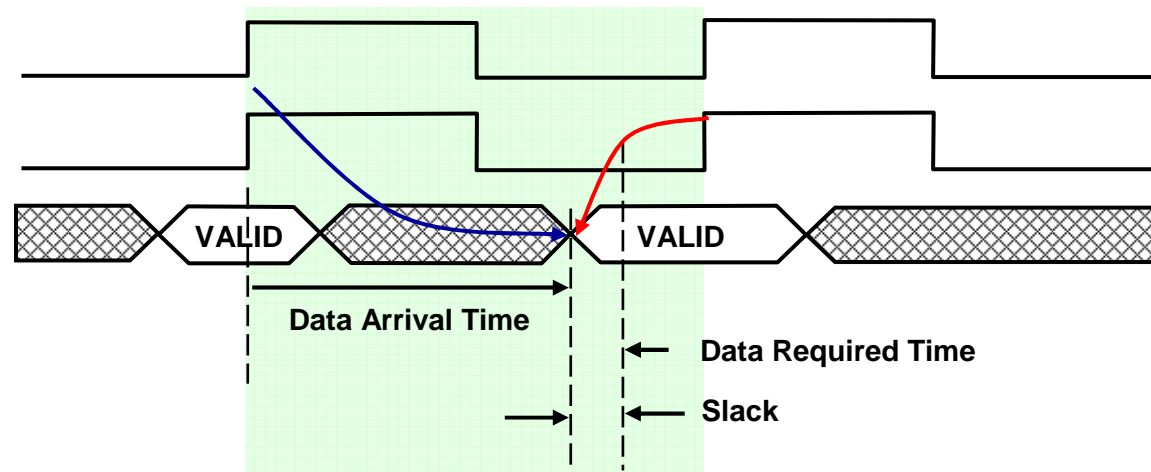
Clk Spec

Cell + Net

FF1/CLK

FF2/CLK

FF2/D





# Hold Definition - Summary

Data remains stable for a minimum time as required by capture flip-flop. (Hold Check)

EQN 1  $\text{Slack}_{\text{hold}} = \text{Data Arrival Time} - \text{Data Required Time} \geq 0$

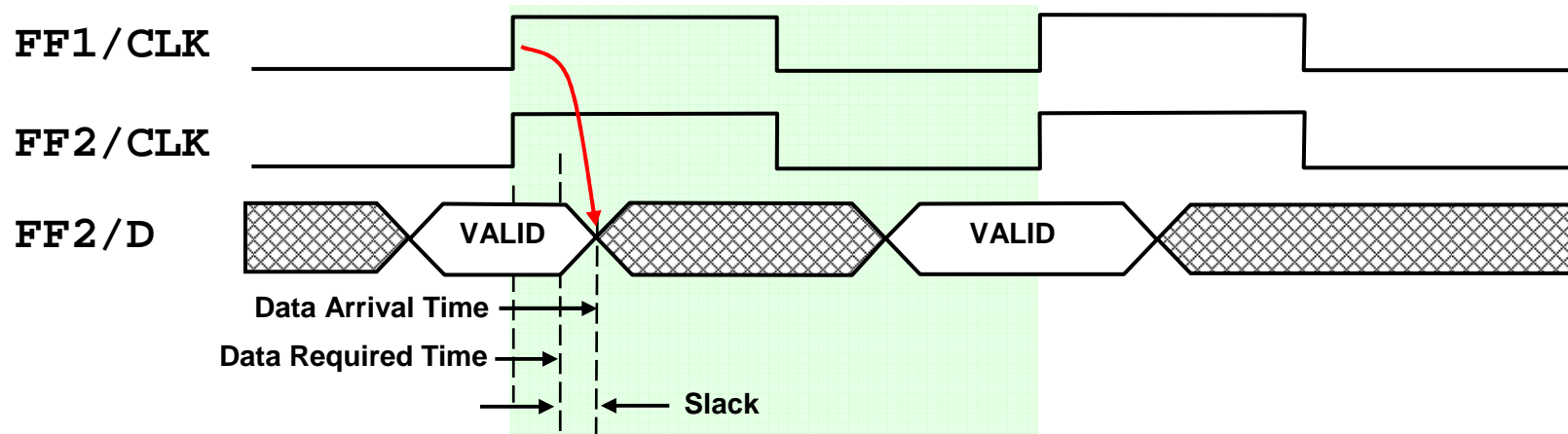
EQN 2  $\text{Slack}_{\text{hold}} = (T_{\text{launch}} + t_{\text{prop}}) - (T_{\text{capture}} + t_{\text{hold}}) \geq 0$

Clk Spec

Cell + Net

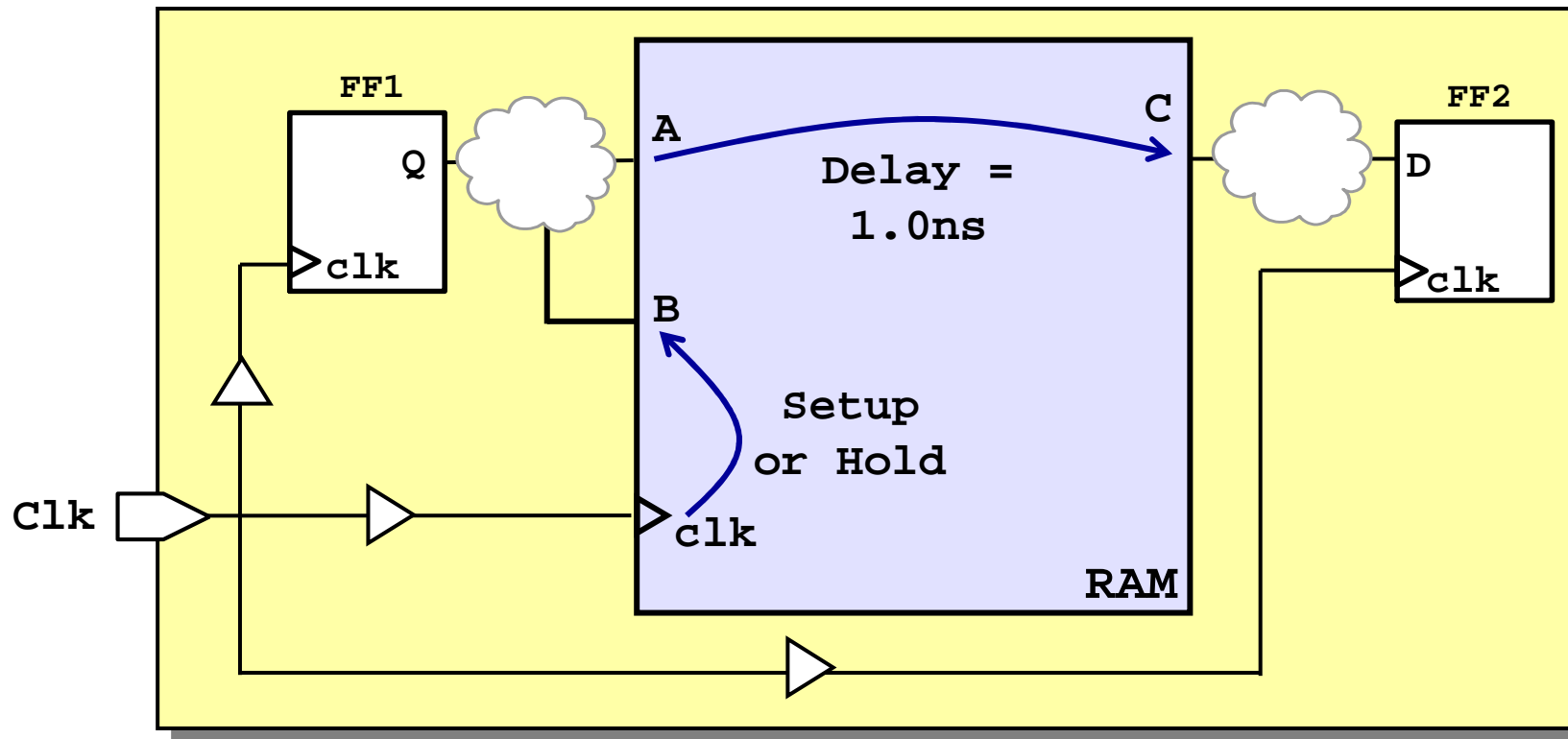
Clk Spec

Library



# Timing Models

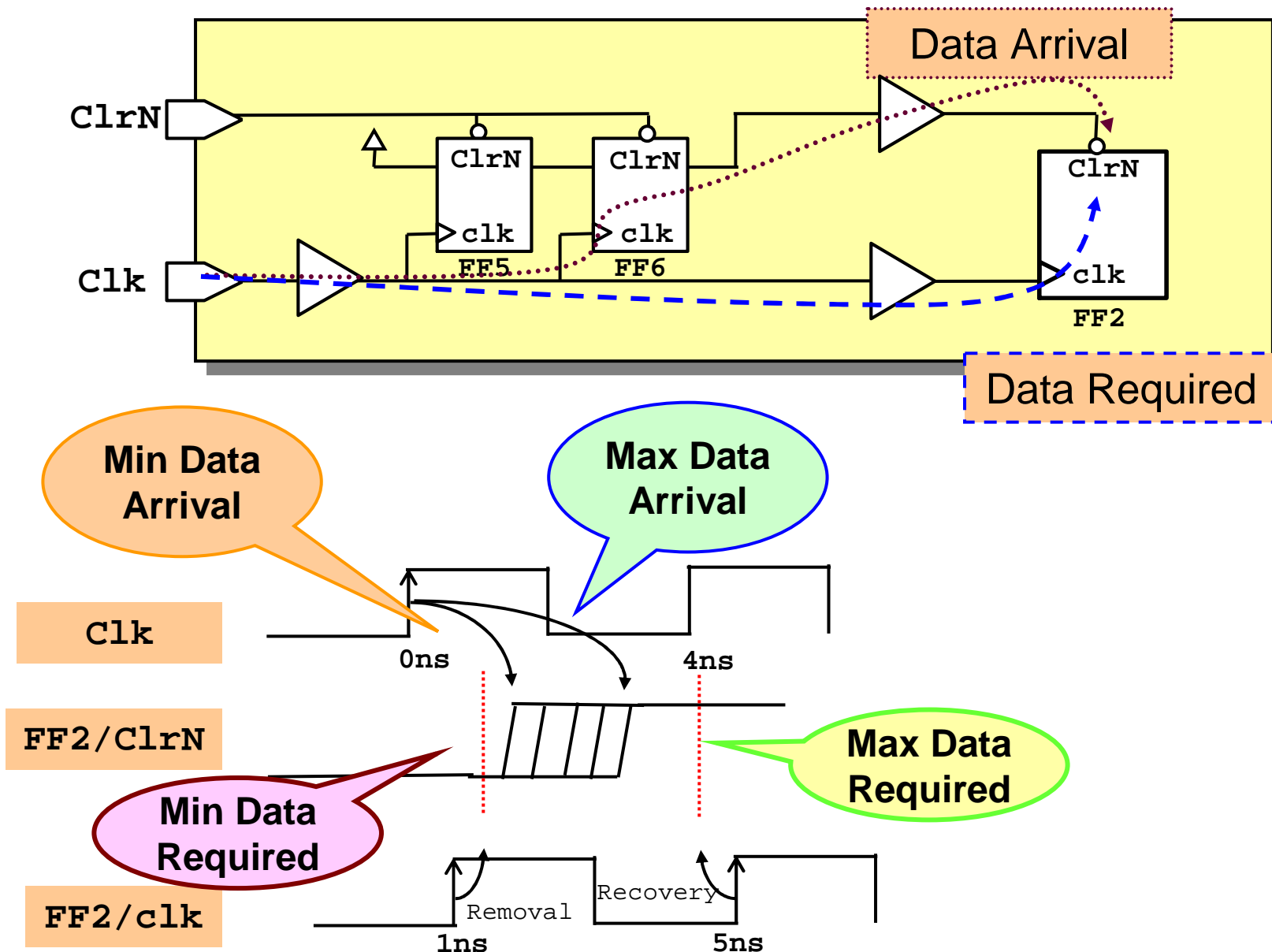
- **Timing models are cells with many timing arcs:**
  - “Flip-flop” with setup and hold timing checks
  - “Delay cell” included along the data arrival time



# Example Timing Report

Point	Incr	Path
-----	-----	-----
clock SYS_CLK (rise edge)	0.000	0.000
clock network delay (propagated)	2.713 *	2.713
I_ORCA_TOP/I_PCI_WRITE_FIFO/count_int_reg[0]1/CP (sdcrq1)	0.000	2.713 r
I_ORCA_TOP/I_PCI_WRITE_FIFO/count_int_reg[0]1/Q (sdcrq1)	0.678 *	3.390 r
I_ORCA_TOP/I_PCI_WRITE_FIFO/PCI_WFIFO_RAM/A1[0] ( <b>ram32x32</b> )	0.008 *	3.398 r
data arrival time		3.398
clock SYS_CLK (rise edge)	0.000	0.000
clock network delay (propagated)	2.711 *	2.711
I_ORCA_TOP/I_PCI_WRITE_FIFO/PCI_WFIFO_RAM/CE1 ( <b>ram32x32</b> )		2.711 r
library hold time	0.282 *	2.992
data required time		2.992
-----	-----	-----
data required time		2.992
data arrival time		-3.398
-----	-----	-----
slack (MET)		0.406

# Asynchronous Clear/Reset Pins



# Timing Report Recovery

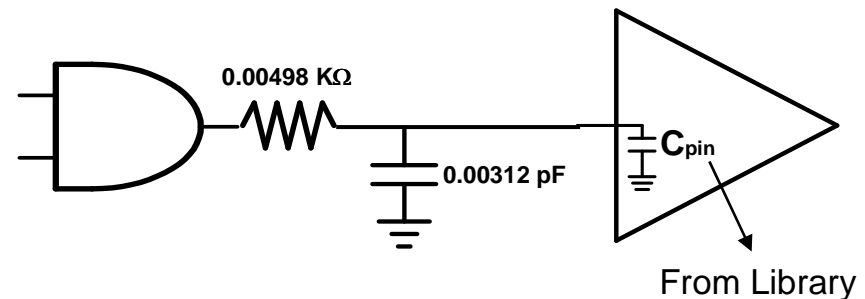
Startpoint: I\_ORCA\_TOP/I\_RESET\_BLOCK/sys\_2x\_rst\_n\_buf\_reg  
(rising edge-triggered flip-flop clocked by SYS\_2x\_CLK)  
Endpoint: I\_ORCA\_TOP/I\_RISC\_CORE/I\_ALU/Neg\_Flag\_reg  
(recovery check against rising-edge clock SYS\_2x\_CLK)  
Path Group: \*\*async\_default\*\*  
Path Type: max

Point	Incr	Path
-----		
clock SYS_2x_CLK (rise edge)	0.000	0.000
clock network delay (propagated)	2.846 *	2.846
I_ORCA_TOP/I_RESET_BLOCK/sys_2x_rst_n_buf_reg/CP (sdcrq1)	0.000	2.846 r
. . .		
I_ORCA_TOP/I_RISC_CORE/I_ALU/Neg_Flag_reg/CDN (sdcrb1)	0.073 *	3.974 r
data arrival time		3.974
clock SYS_2x_CLK (rise edge)	4.000	4.000
clock network delay (propagated)	2.833 *	6.833
I_ORCA_TOP/I_RISC_CORE/I_ALU/Neg_Flag_reg/CP (sdcrb1)		6.833 r
library recovery time	0.128 *	6.962
data required time		6.962
-----		
data required time		6.962
data arrival time		-3.974
-----		
slack (MET)		2.988

# Estimating Rnet and Cnet Pre-layout

- Extraction data of already routed designs are used to build a lookup table called the wire load model
- WLM is based on the statistical estimates of R and C based on “Net Fanout”

Net Fanout	Resistance K $\Omega$	Capacitance pF
1	0.00498	0.00312
2	0.01295	0.00812
3	0.02092	0.01312
4	0.02888	0.01811



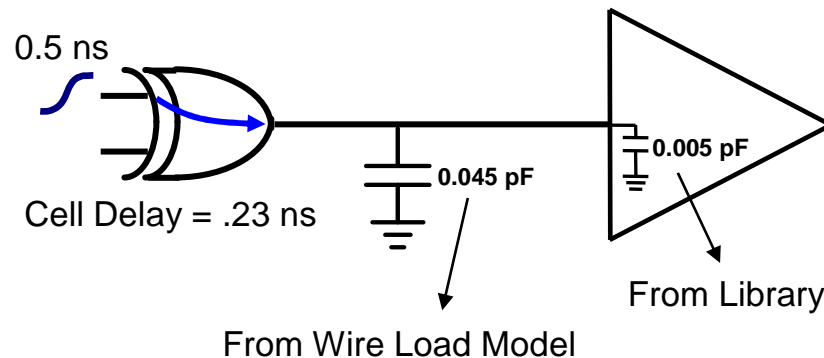
Wire Load Model (RC)

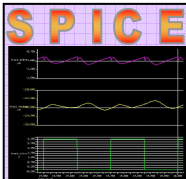
Estimated RCs are represented as wire load model

# Cell Delay Calculation

- Cell delays are calculated from a Non Linear Delay Model (NLDM) table in the technology library
- Tables are indexed by input transition and total output load for each gate

**Cell Delay = f (Input Transition Time, Output Load)**

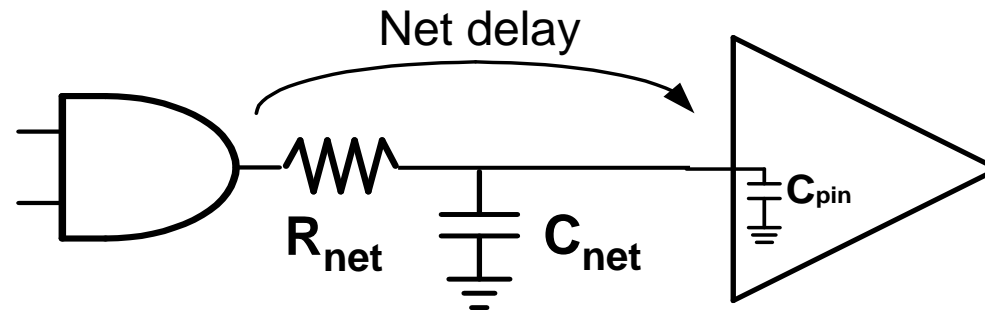


		Output Load (pF)			
		.005	.05	.10	.15
Input Trans (ns)	0.0	.1	.15	.2	.25
	0.5	.15	.23	.3	.38
	1.0	.25	.4	.55	.75

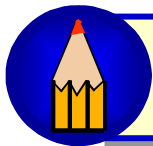
Cell Delay (ns)

# Net Delay Calculation

- Net delay is the “time-of-flight” due to the net’s RC
- Net’s RC is obtained from wire load model for pre-layout design



$$\text{Net Delay} = f(R_{net}, C_{net} + C_{pin})$$



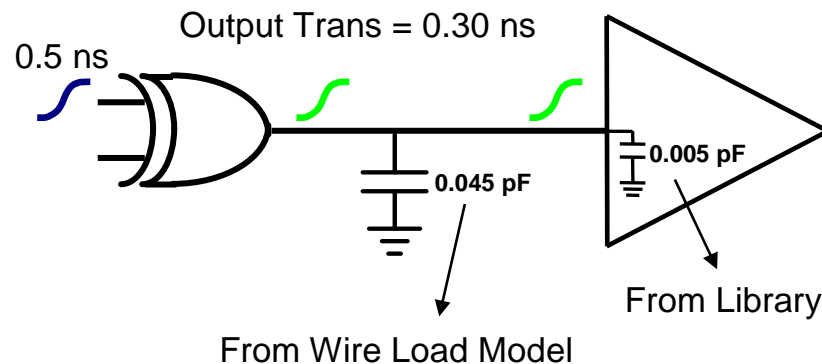
Post-layout Rs and Cs are extracted as a parasitics file.

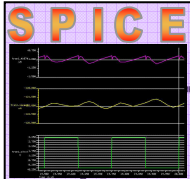


# Output Transition Calculation

- There is another NLDM table in the library to calculate output transition
- Output transition of a cell becomes the input transition of the next cell down the chain

**Output Transition = f (Input Transition Time, Output Load)**



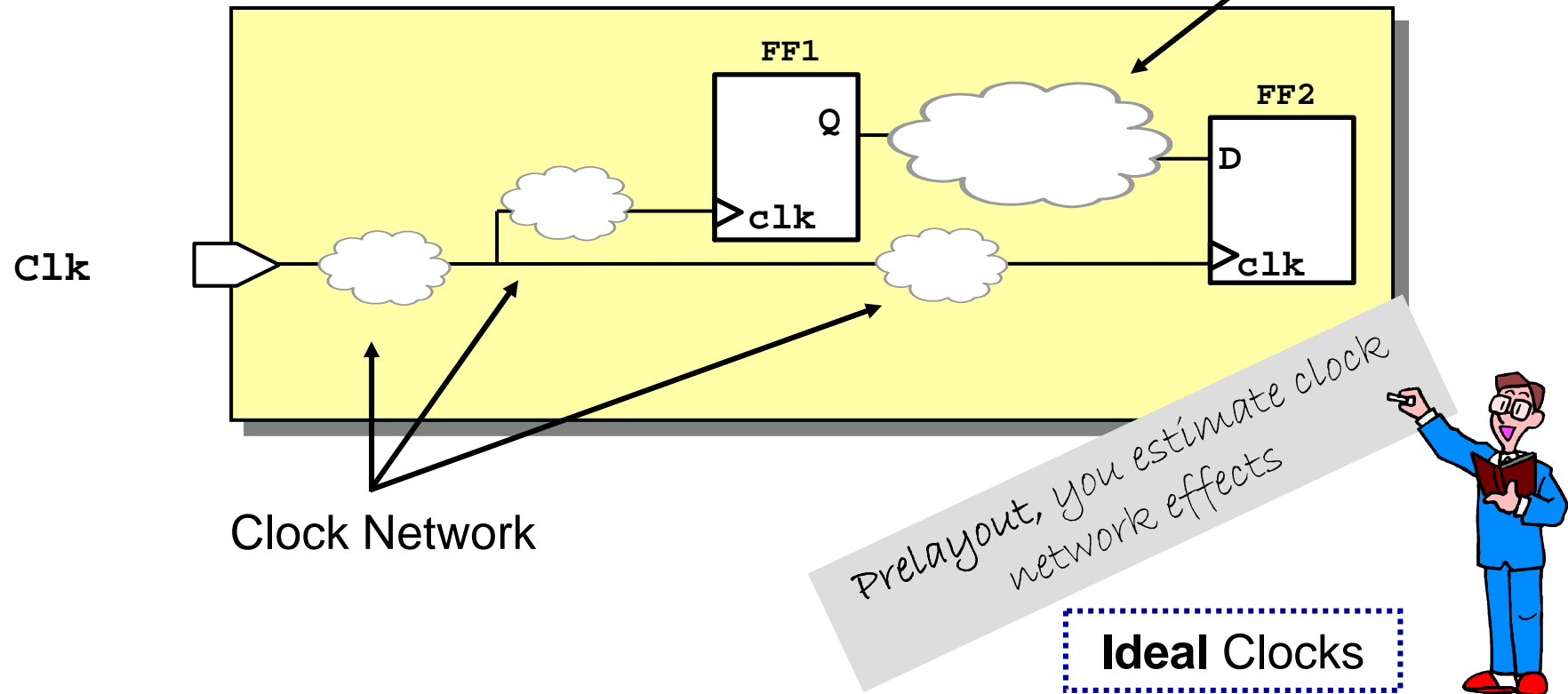
		Output Load (pF)			
		.005	.05	.10	.15
Input Trans (ns)	0.00	0.10	0.20	0.37	0.60
	0.50	0.18	0.30	0.49	0.80
	1.00	0.25	0.40	0.62	1.00
		Output Transition (ns)			

# What About Pre and Post Layout STA?

Post layout, an STA tool calculates clock network effects

**Propagated Clocks**

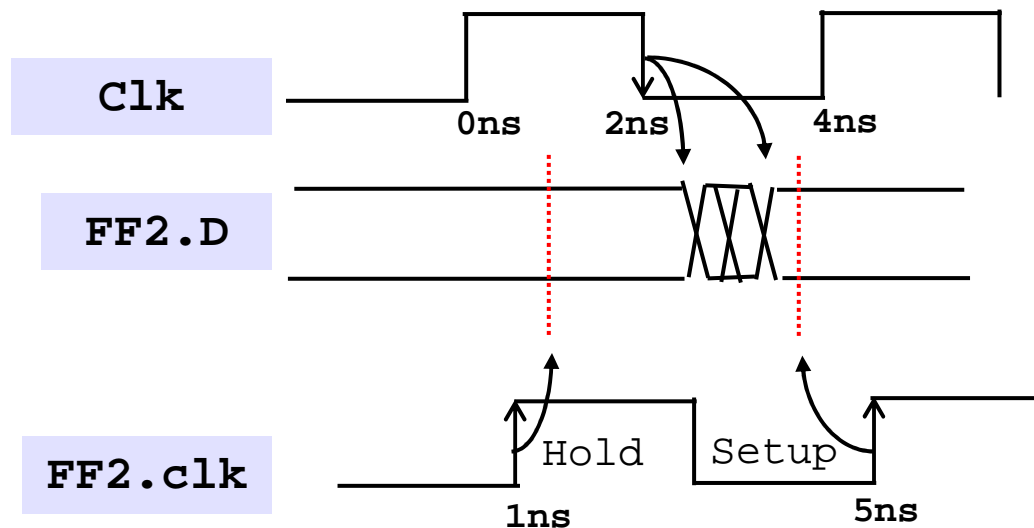
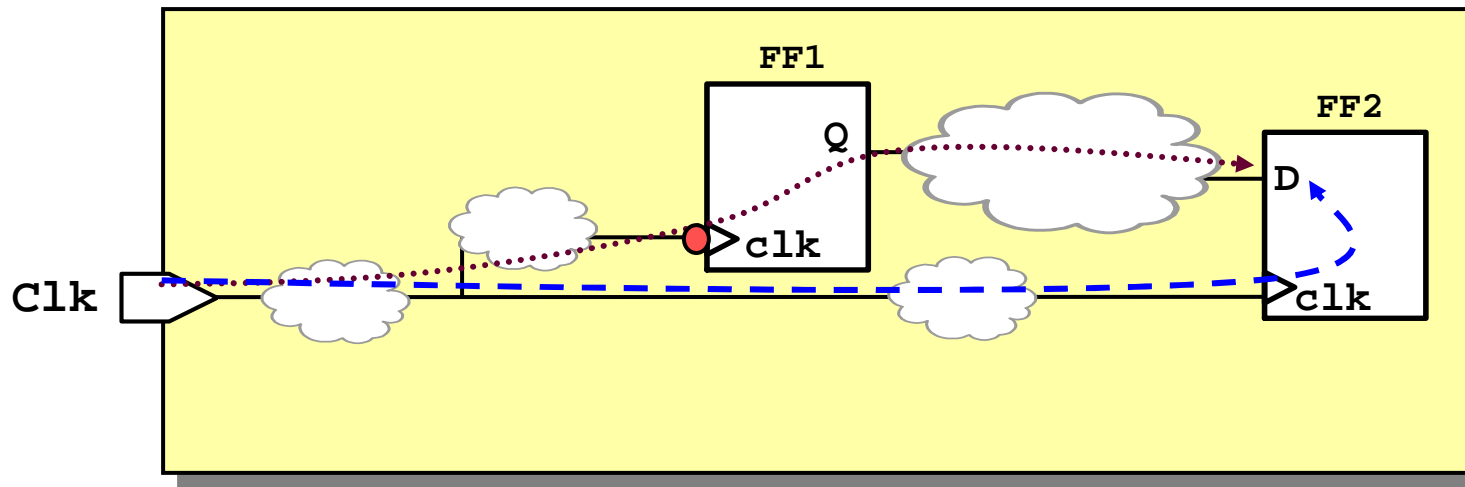
SDF contains estimated or actual delays



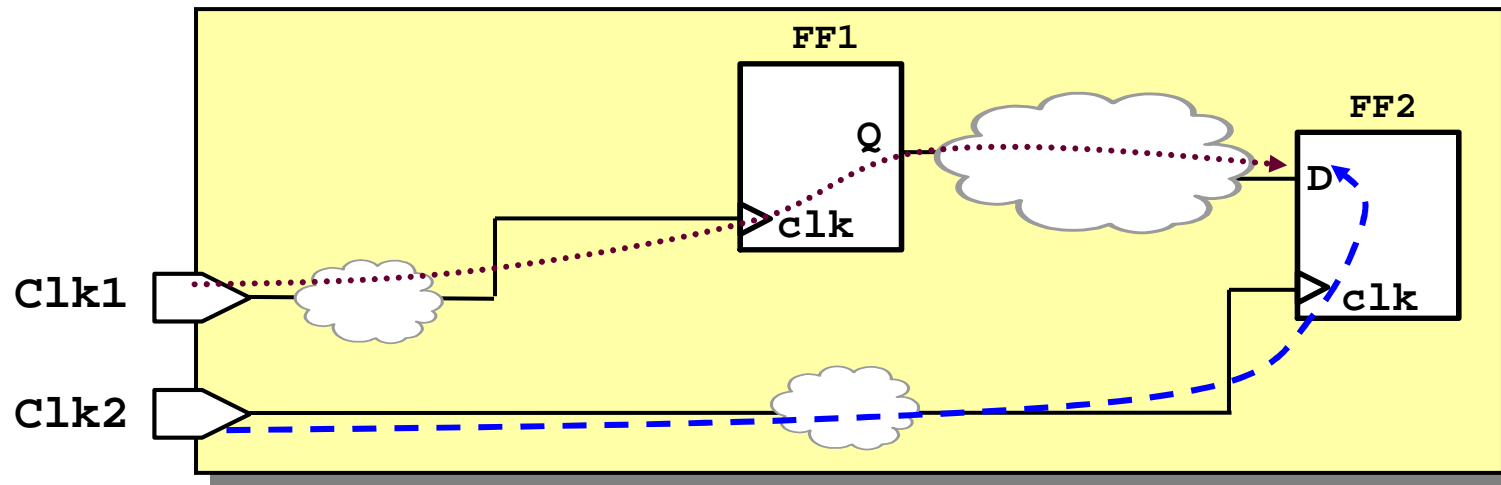
# Pre or Post Layout Timing Report

Point	Incr	Path
-----	-----	-----
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.10 *	1.10
FF1/CLK (fdef1a15)	0.00	1.10 r
FF1/Q (fdef1a15)	0.40 *	1.50 f
U2/Y (buf1a27)	0.05 *	1.55 f
U3/Y (buf1a27)	0.05 *	1.60 f
FF2/D (fdef1a15)	0.01 *	1.61 f
data arrival time		1.61
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.00 *	1.00
FF2/CLK (fdef1a15)		1.00 r
library hold time	-0.10 *	1.10
data required time		1.10
-----	-----	-----
data required time		1.10
data arrival time		-1.61
-----	-----	-----
slack (MET)		0.51

# What About Negedge Triggered Registers?

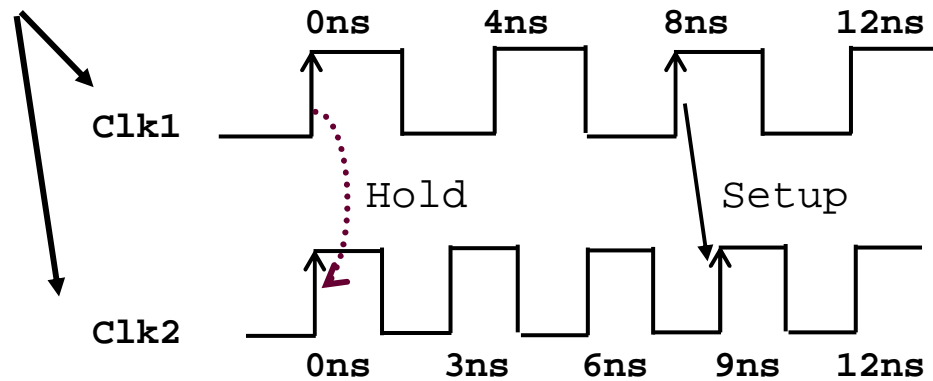


# What About Multi-Frequency Clocks?



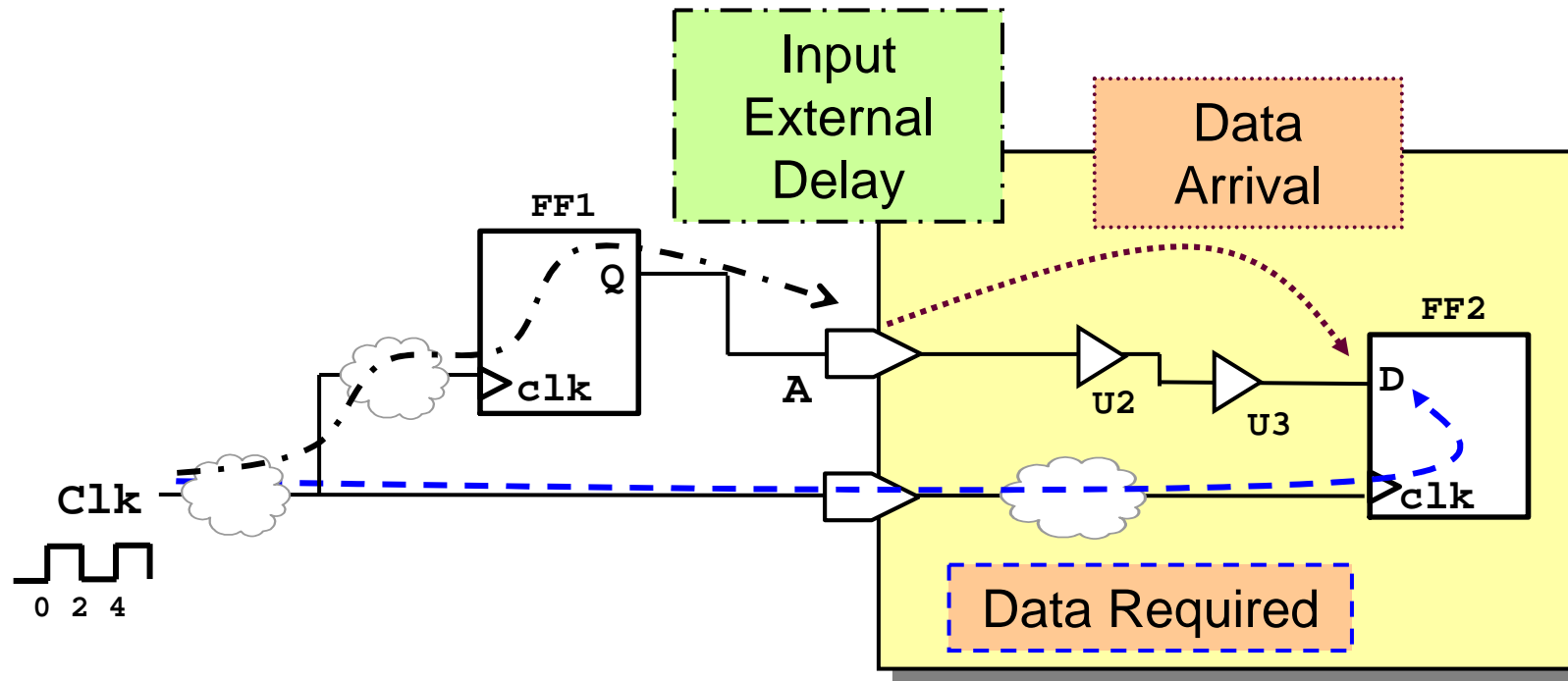
Create both clocks

**Base Period** is from 0ns to 12ns



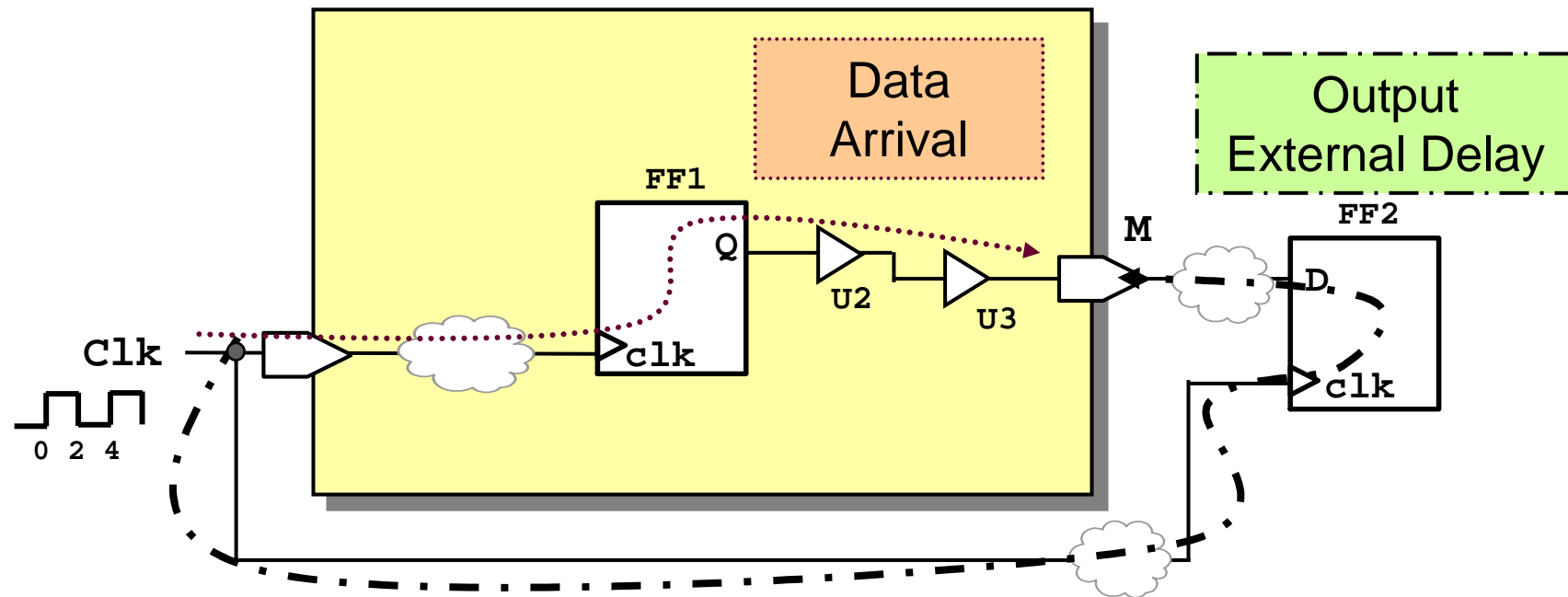
# What About Interface Paths: Input Ports?

You specify the arrival times at the input ports of the design.



# What About Interface Paths: Output Ports?

You specify the path required time at the output ports of the design.

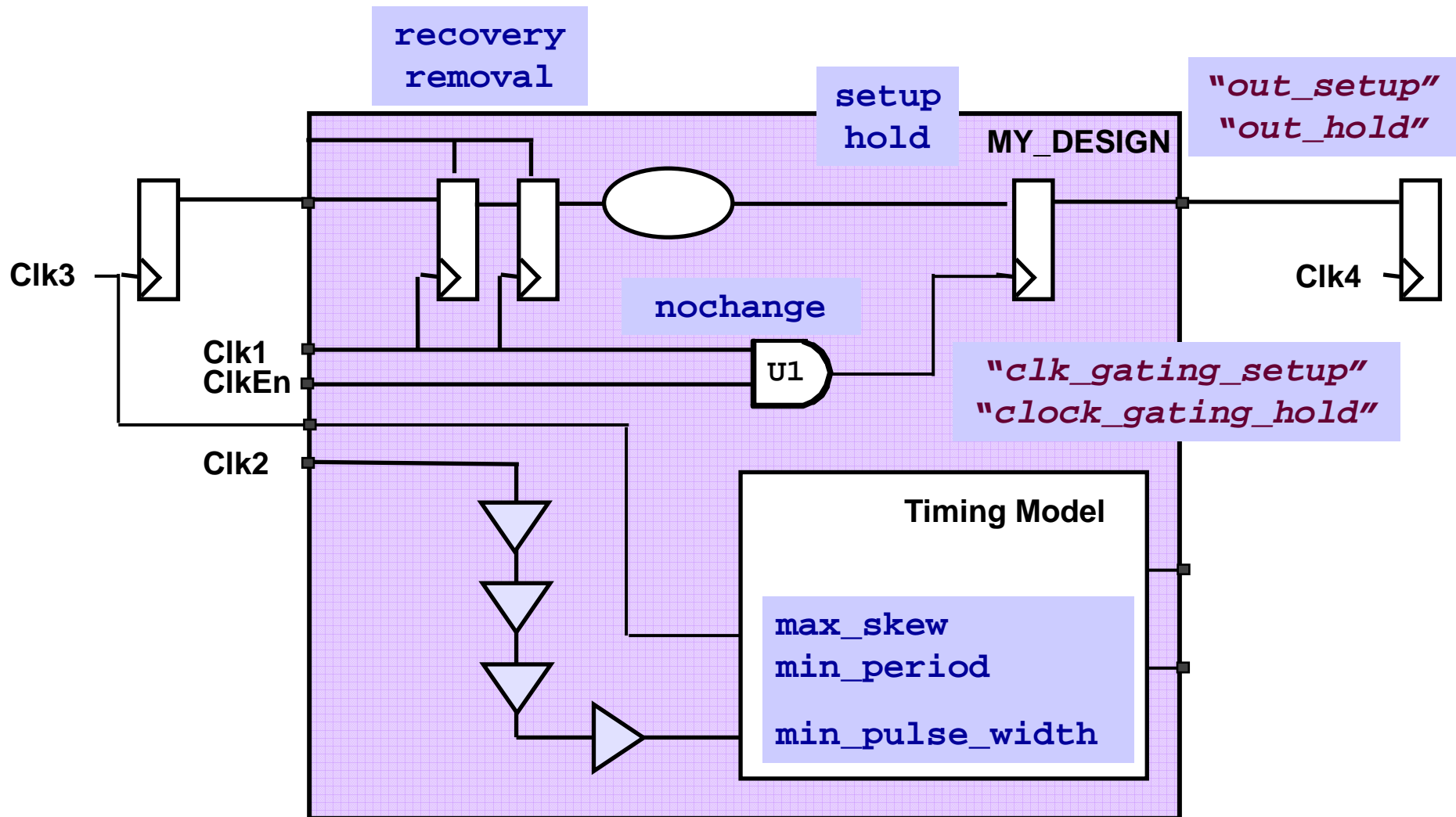


# Interface Paths in a Timing Report: Output

Point	Incr	Path
-----	-----	-----
clock Clk (rise edge)	0.00	0.00
clock network delay (propagated)	1.10 *	1.10
FF1/CLK (fdef1a15)	0.00	1.10 r
FF1/Q (fdef1a15)	0.50 *	1.60 r
U2/Y (buf1a27)	0.11 *	1.71 r
U3/Y (buf1a27)	0.11 *	1.82 r
M (out)	0.05 *	1.87 r
data arrival time		1.87
clock Clk (rise edge)	4.00	4.00
clock network delay (propagated)	0.00 *	4.00
output external delay	-0.21 *	3.79
data required time		3.79
-----	-----	-----
data required time		3.79
data arrival time		-1.87
-----	-----	-----
slack (MET)		1.92



# Other Timing Checks Verified by STA



**"Timing checks":** specified by the user

**Timing checks:** specified by the vendor

# Introduction to Digital VLSI Design

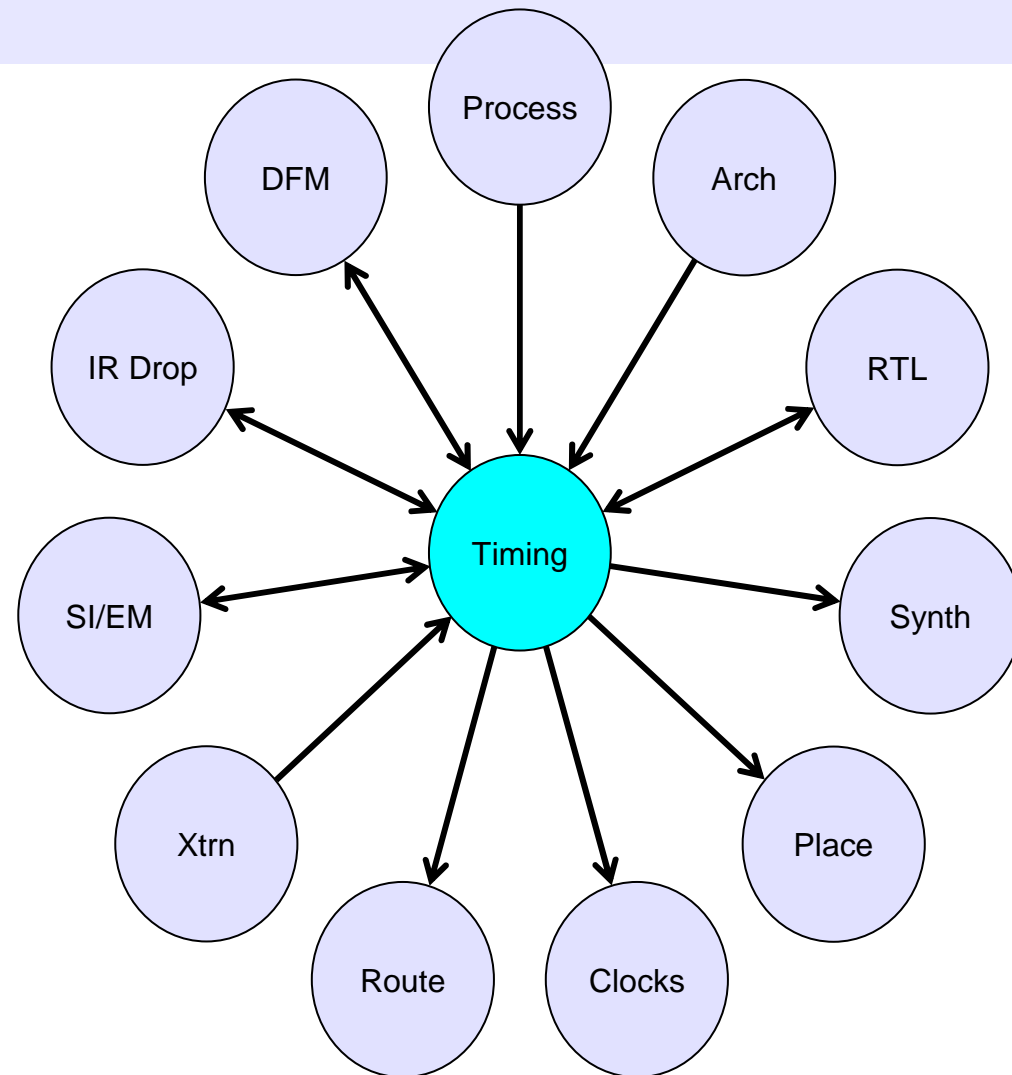
ספרתי VLSI מבוא לתכנון

**STA part 2**

# What

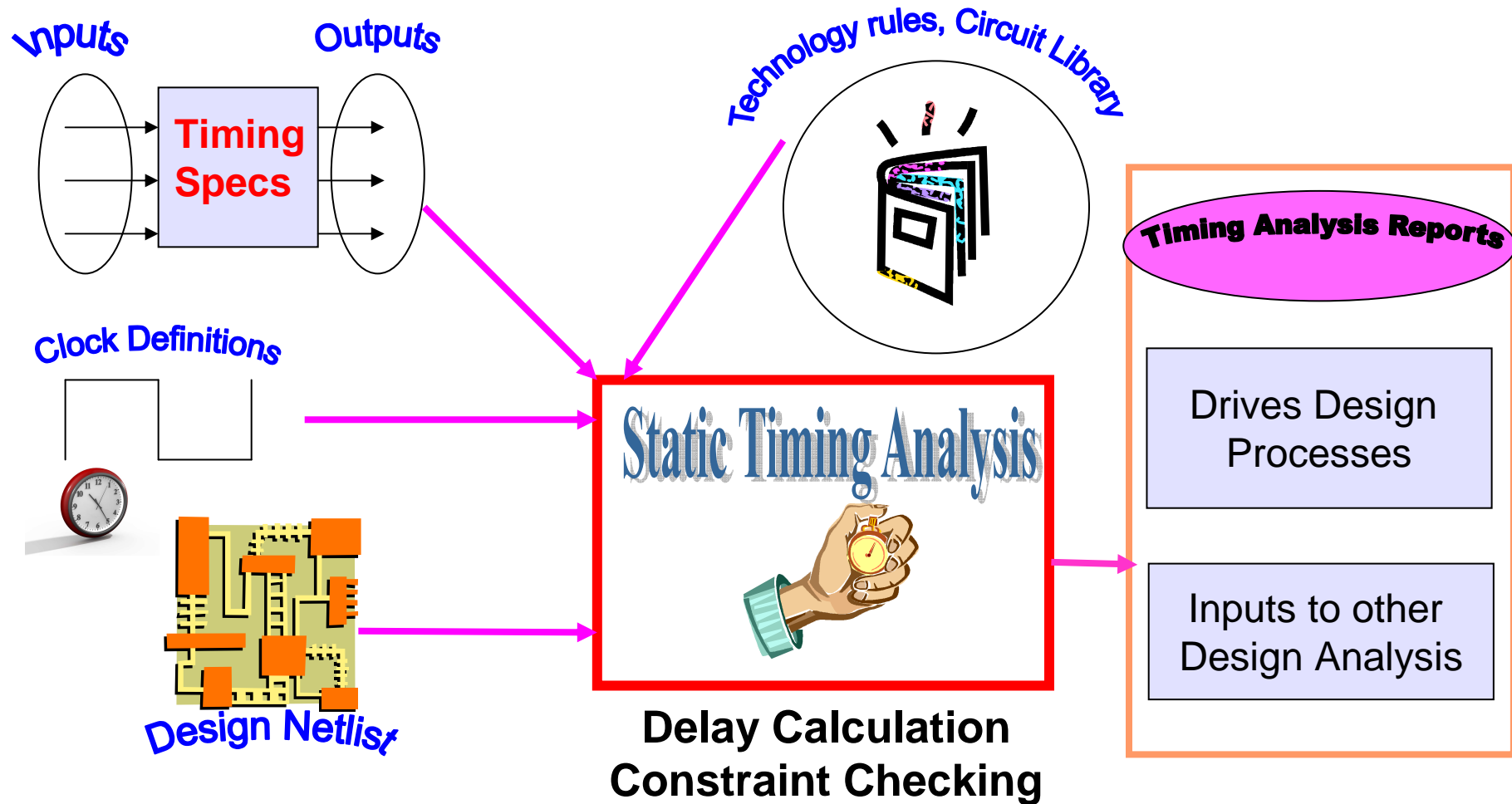
- *Fast and Exhaustive*
- *Independent of functionality or stimulus*
- *Spice accurate*
- *Implement and Verify*

# When



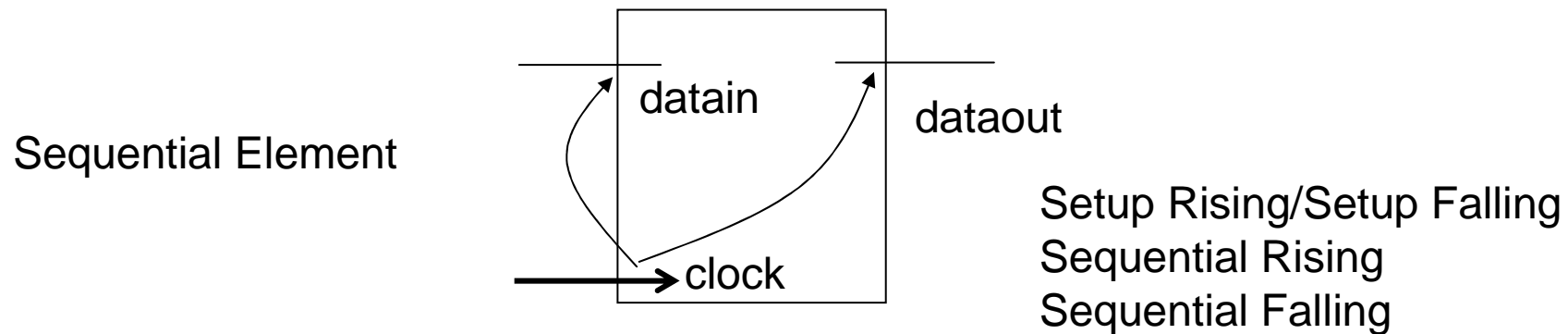
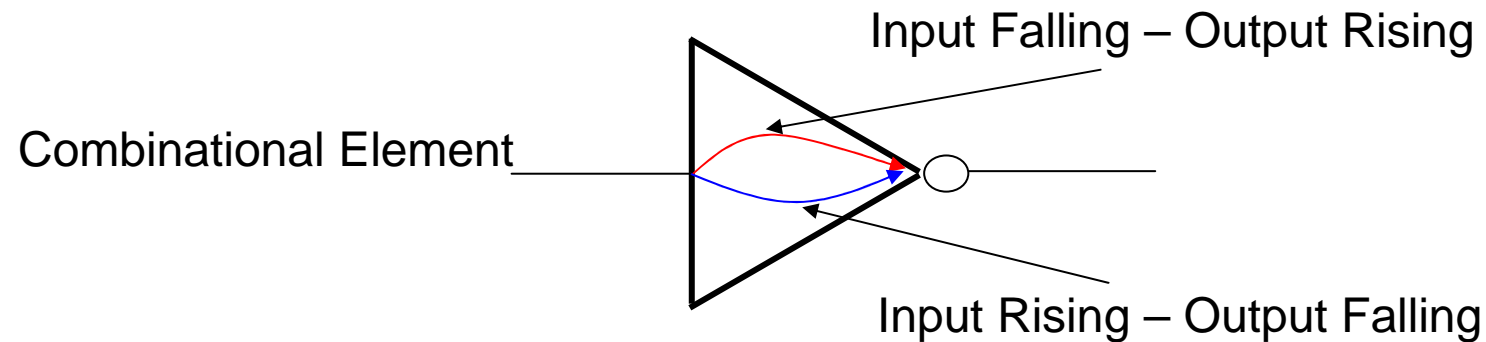
## STA - True Design Driver

# Components



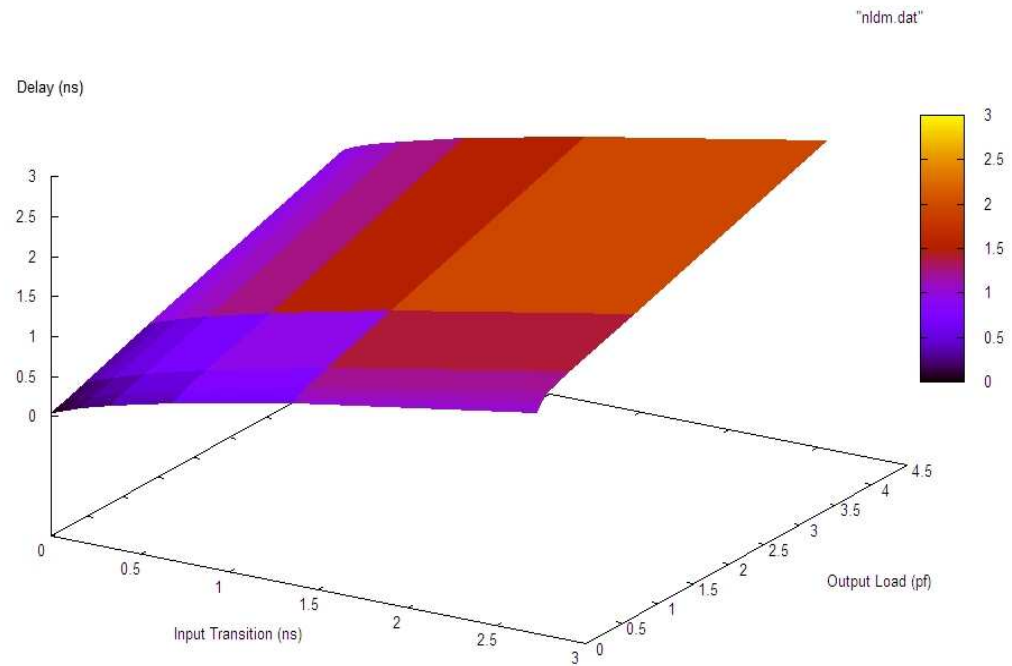
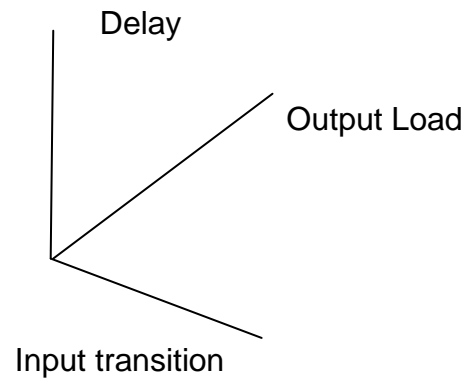
# Delay Calculation

## Timing Arcs



# Delay Calculation NLDM Library

## NLDM

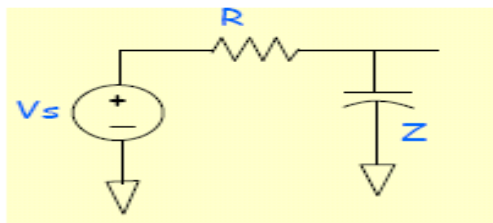


# Delay Calculation

## NLDM Library (contd.)

### NLDM Libraries

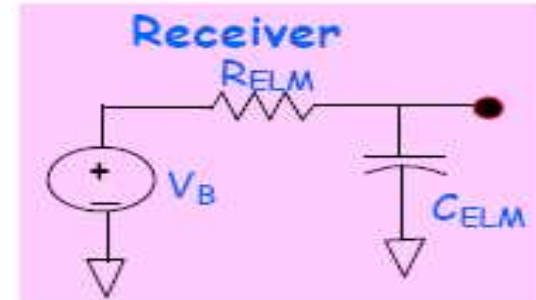
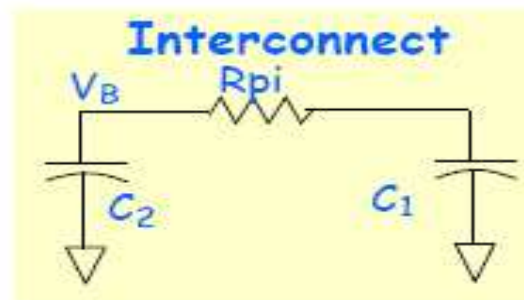
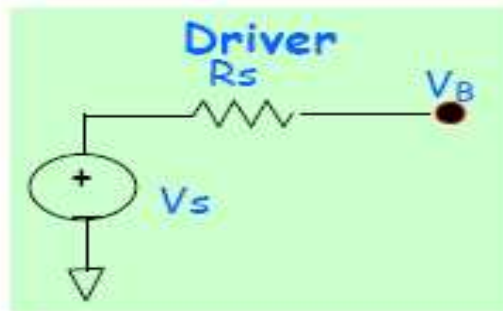
- Existing NLDM like modeling:



Driver  
Model

$$V_{out}(s) = Z(s) V_s(s) / (Z(s) + R)$$

As  $Z(s)$  increases:  $V_{out}(s) \sim V_s(s)$

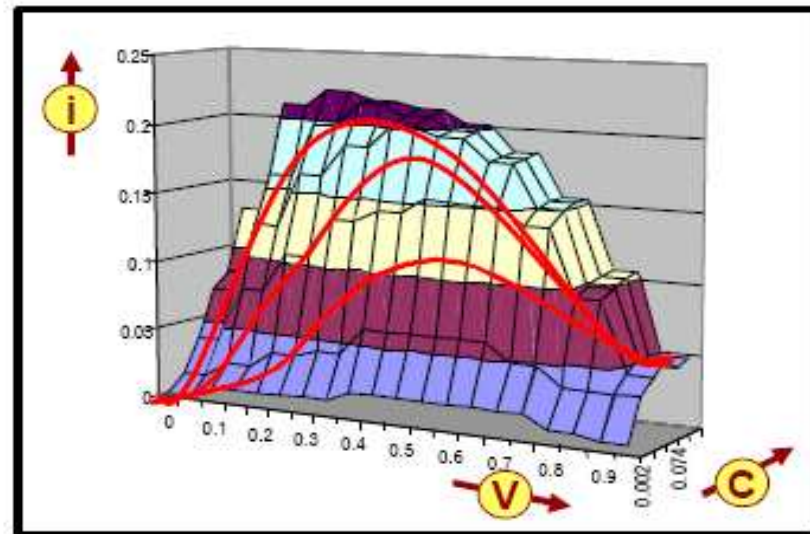
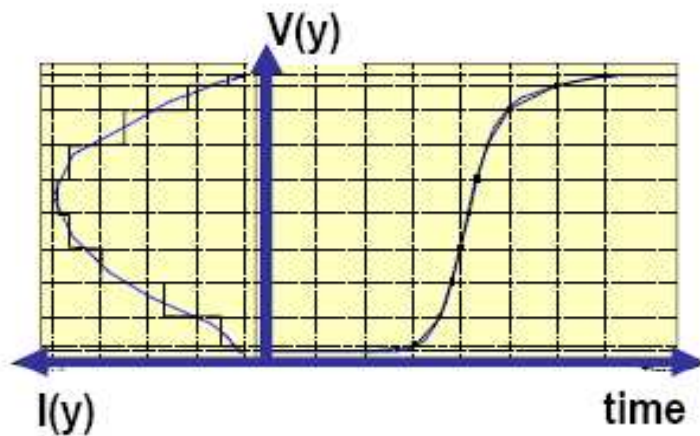
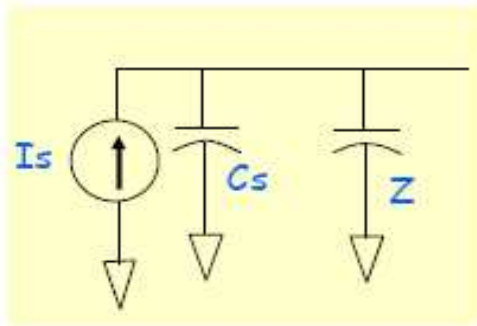




# Delay Calculation

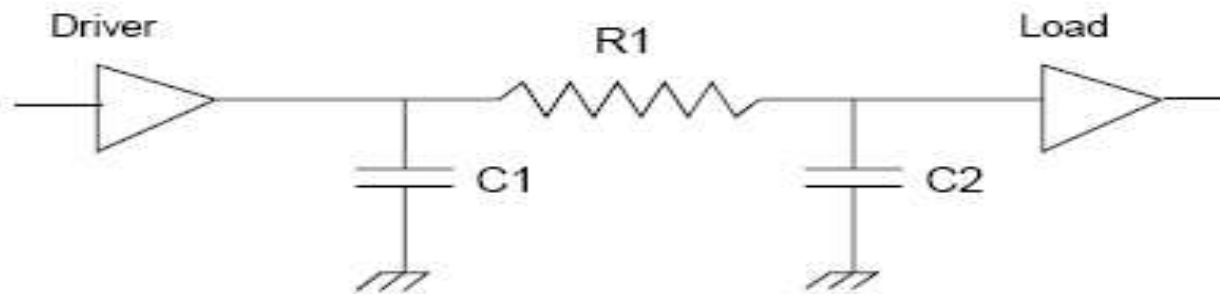
## ECSM Library

Current Source Model: Voltage Controlled - Current Source



$$I = C \frac{\Delta V}{\Delta t}$$

# Delay Calculation Interconnect



- IEEE Standard format – SPEF
  - Distributed RC

# Delay Calculation

## Analysis Corners

### ■ Gate or Transistor

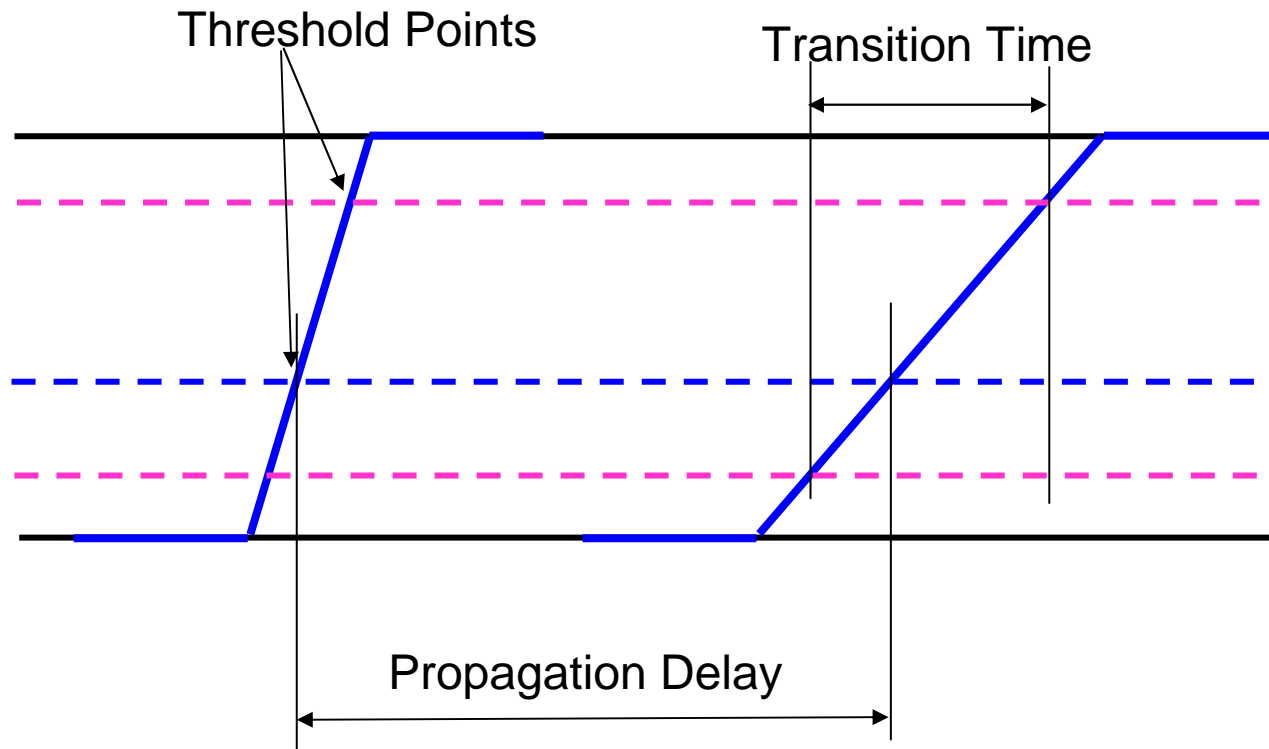
- P – Process (Slow, Typical, Fast)
- V – Supply Voltage
- T – Temperature

### ■ Interconnect

- P – Process (Wide, Narrow, Tall, Short, K)
- T - Temperature

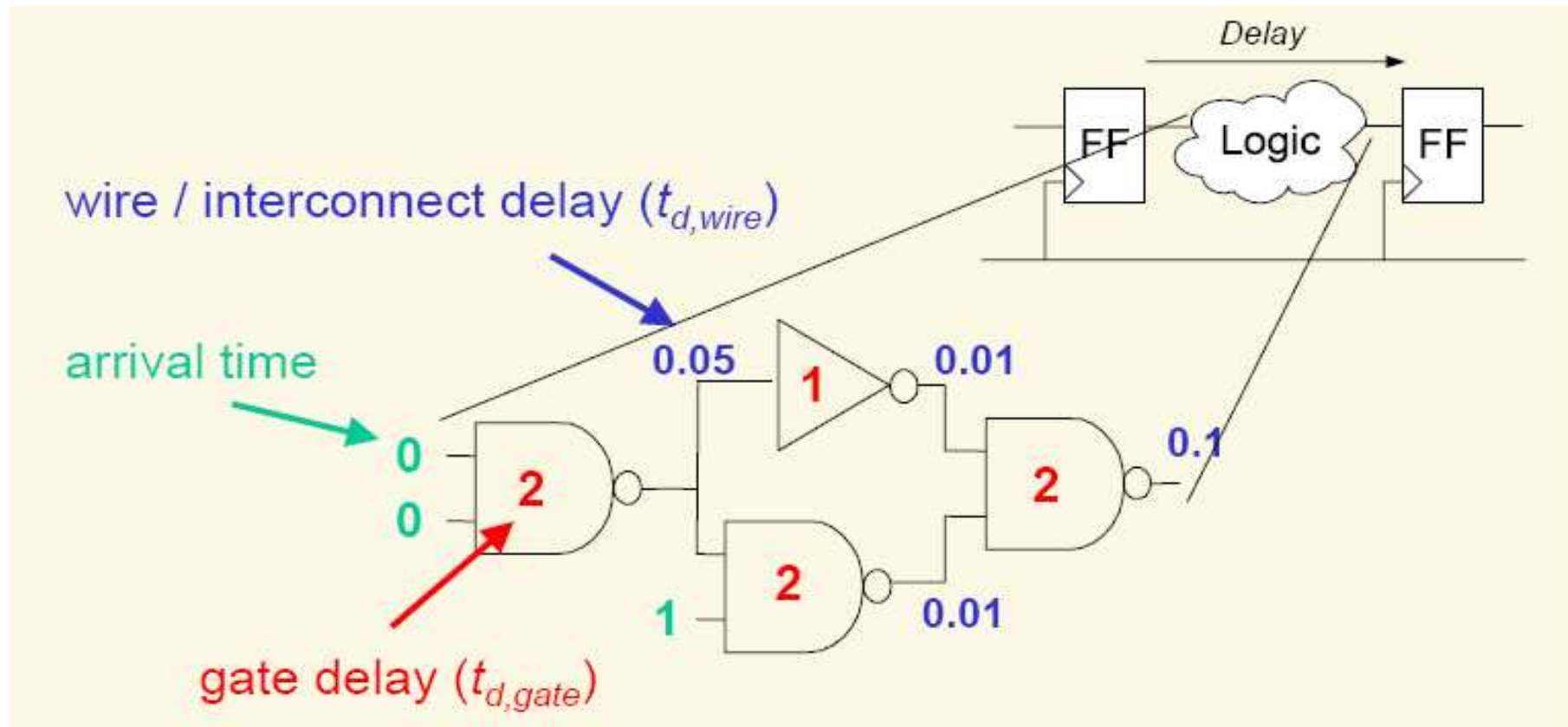
# Delay Calculation

## Thresholds



## Delay Calculation

## Path Delay Calculations

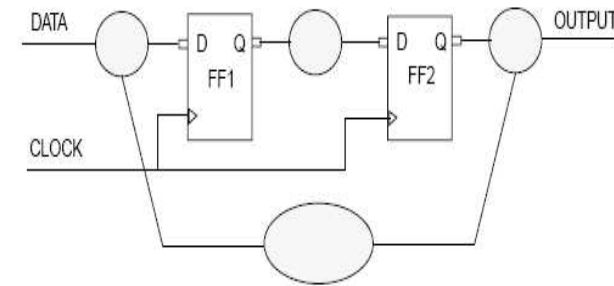
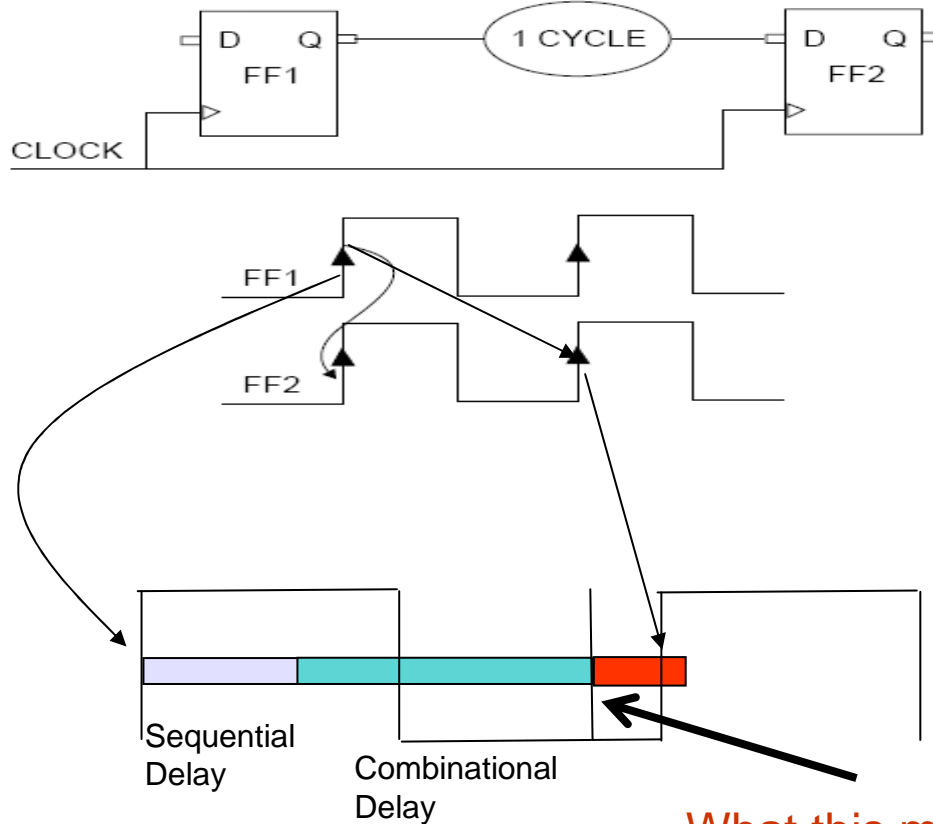


Worst arrival time of signal at input pin of capture flop = ?  
Best arrival time of signal at input pin of capture flop = ?

# Constraint Checking

## Introduction

### Sequential Operation of a single Cycle path



Timing Paths

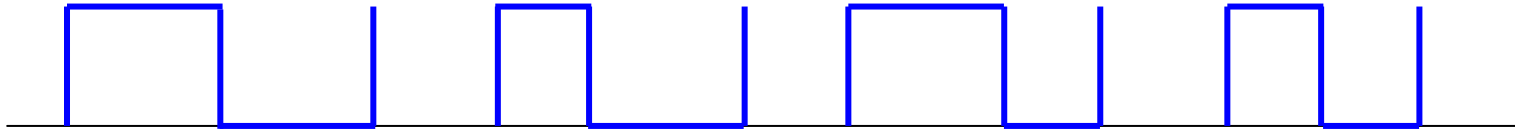
What this mark is for?

# Constraint Checking

## Constraint Types

- **Conditions that need to be met**
  - Clocks
  - Max allowed transition time
  - Max allowed load or capacitance
  - Max allowed Delay
- **Boundary Settings**
  - Input transition time
  - Output loading
  - Logic settings
- **Exceptions to the single cycle rule**
  - False paths
  - Multicycle paths

# Clocks



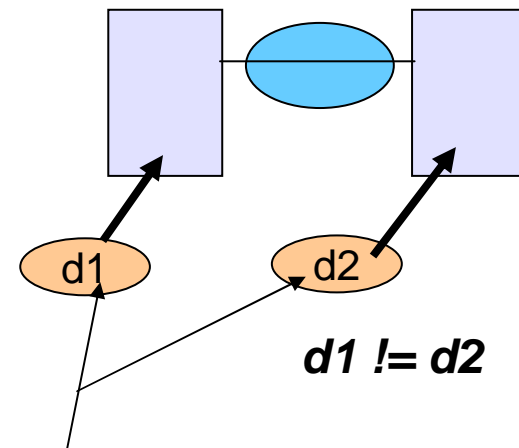
Ex-I

Ex-II

Ex-III

Ex-IV

- **Synchronous Designs**
- **Default single cycle of operation**
  - Launch Edge and Capture Edge
- **Properties**
  - Period
  - Waveform
  - Rise/Fall Transition Time
  - Skew or Uncertainty
- **Generated Clocks**
  - Derived from a master
    - ◆ Synchronous by definition
    - ◆ Definite edge relationship





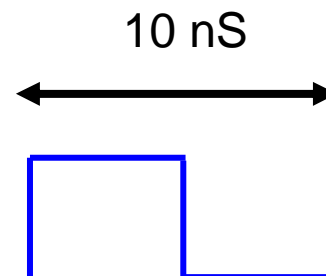
# Virtual Clocks

- Virtual Clocks do not have any physical existence
- Virtual Clocks are used as a reference to module for input and output delays
- Virtual Clocks are local to module design

- **Properties**

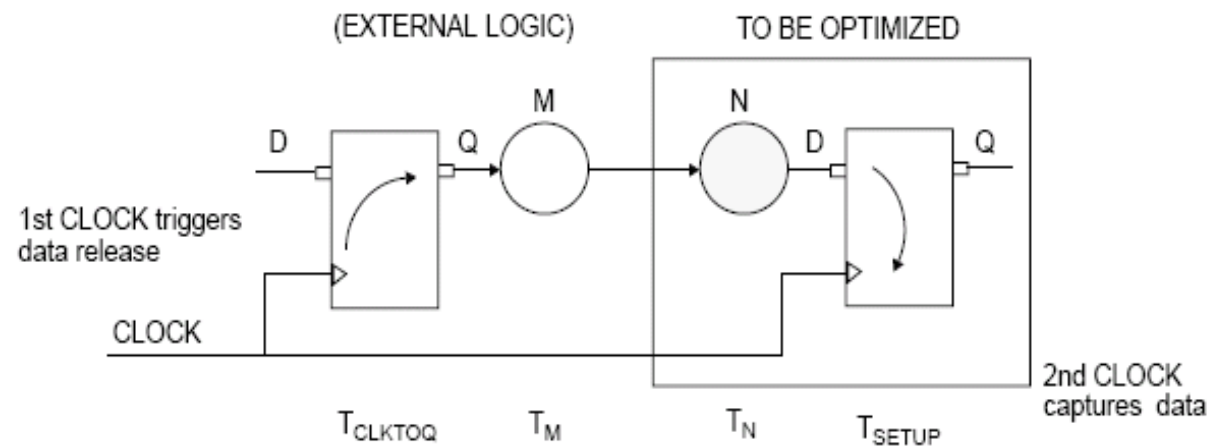
Period

Waveform



# Input Arrival Time

## Modeling I/P Arrival Time

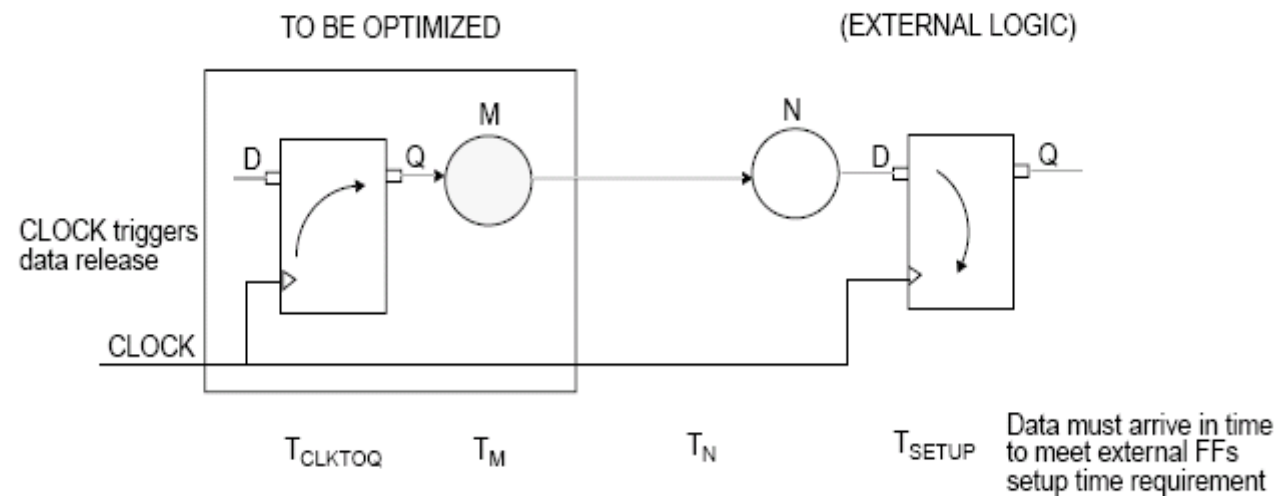


The I/P data arrives after  $T_{CLKTOQ} + T_M$

This is the amount of time to specify as the input delay

# Output Required Time

## Modeling O/P Timing Relative to Clock



The external logic's setup requirement relative to the clock:

Output delay to be specified:  $T_N + T_{SETUP}$

# Global Constraints

- **Specifying min-max Cap Range**

This specification ensures that circuits used in design work within library characterization limits

- **Specifying max Transition**

This specification ensures that transition thus propagated doesn't give rise to a bad propagation delays

- **Specifying driver-load on ports**

This specification ensures that standard load value is modeled at ports

- **Specifying Input and Output Delays at Ports**

# Check Types

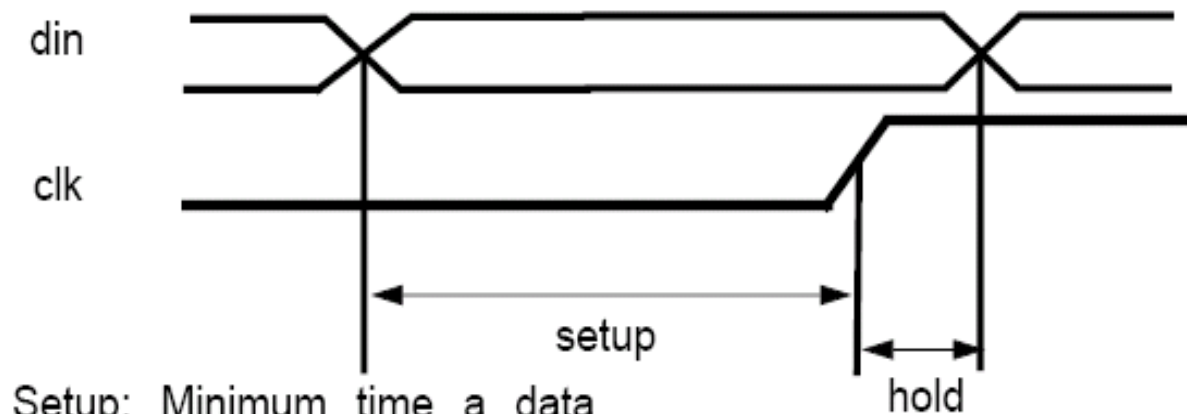
- **Setup**
- **Hold**
- **Recovery**
- **Removal**
- **Clock Gating**
- **Min Pulse Width**
- **Data-to-Data**

# Timing Checks

## Setup *Time* and Hold *Time*

### Setup and Hold

Remember: Setup and Hold Times are *Interdependent*



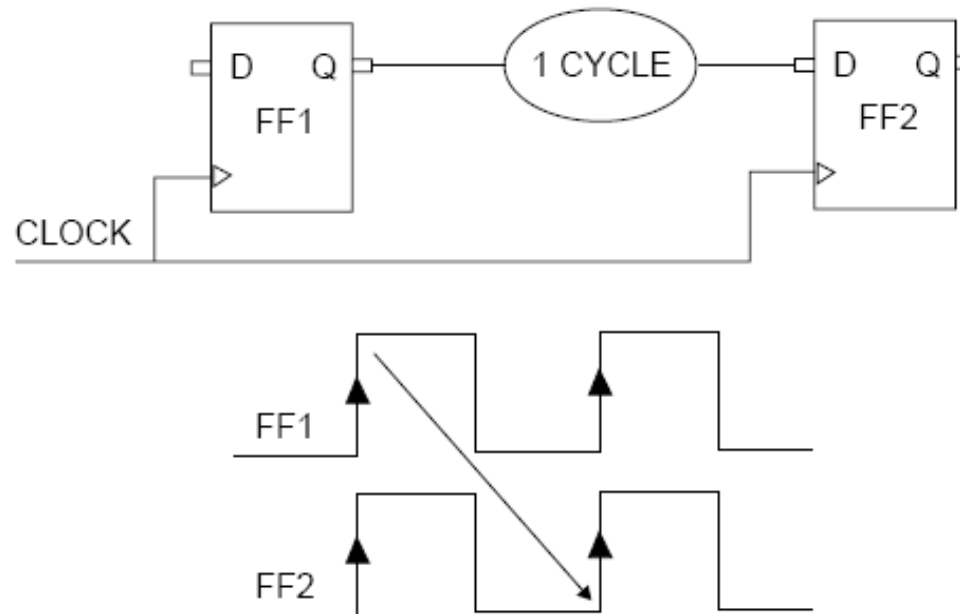
Setup: Minimum time a data input pin of a sequential device must be stable *before* the clock transition

Hold: Minimum time that a data input pin of a sequential device must be stable *after* the clock transition

**Setup *Time* and Hold *Time* are Properties of the Sequential Element Circuit**  
**These need to be honoured to guarantee *expected operation* of the design**

# Timing Checks

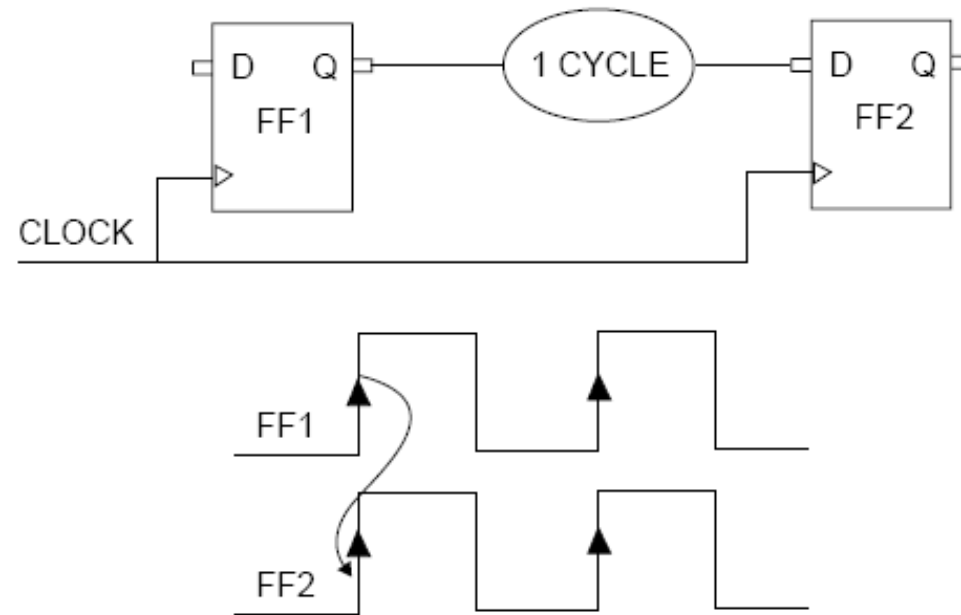
## Setup Check



Data Launched by Launch Edge of FF1 Captured by *Intended* Capture Edge of FF2  
Data launched by launch edge of FF1 should arrive at the data input of FF2 *latest*  
by "*Capture Edge Time – Setup Time of FF2*"

# Timing Checks

## Hold Check



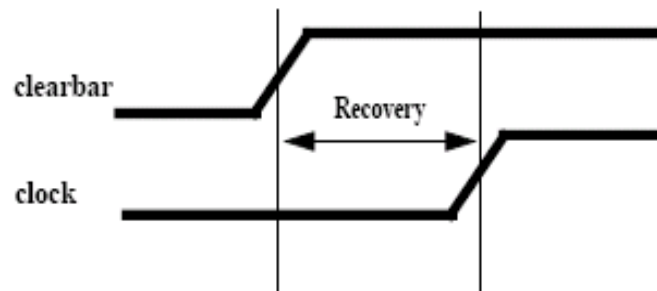
- Data launched by Launch Edge of FF1 should not be captured by an edge *preceding* the intended Capture Edge of FF2, OR
- Data launched by edge *following* Launch Edge of FF1 should not be captured by the intended Capture Edge of FF2
- Data should reach the data input of FF2 no earlier than the hold time of FF2



# Timing Checks

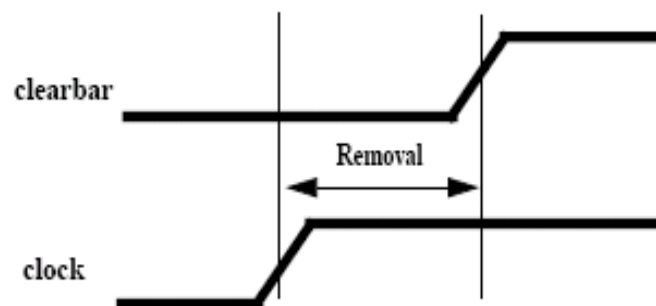
## Recovery and Removal

### Recovery and Removal



Recovery:

Minimum time that an asynchronous control input pin must be stable *after* being *deasserted* and *before* the next clock transition (active-edge).



Removal:

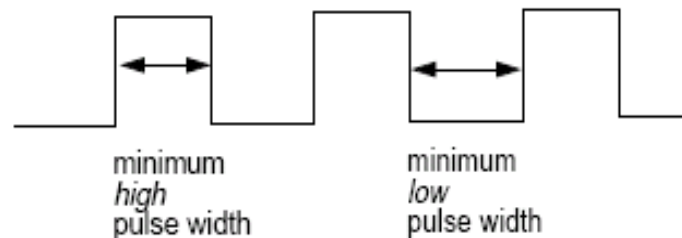
Minimum time that an asynchronous control input pin must be stable *before* being *deasserted* and *after* the previous clock transition (active-edge).

# Timing Checks

## Min Pulse Width

### Minimum Clock Pulse Width

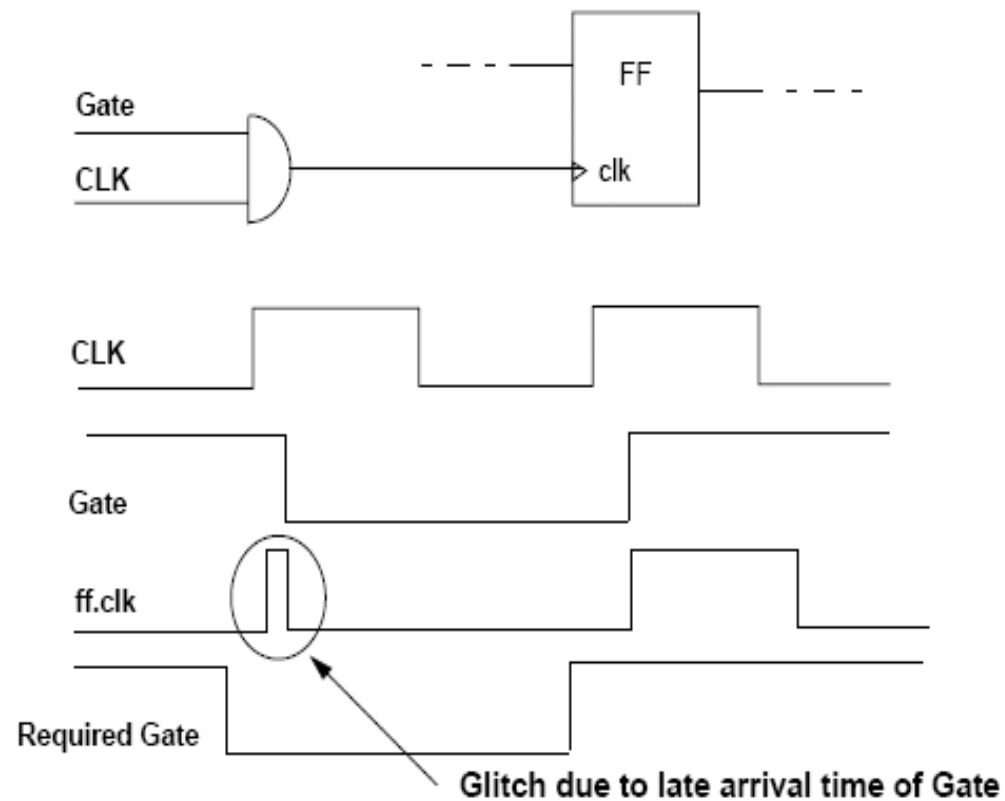
- Minimum High pulse width: The amount of time, *after* the *rising* edge of a clock, that the clock signal of a clocked device must remain stable.
- Minimum Low pulse width: The amount of time, *after* the *falling* edge of a clock, that the clock signal of a clocked device must remain stable.



# Timing Checks

## Glitch Detection

### Glitch Detection

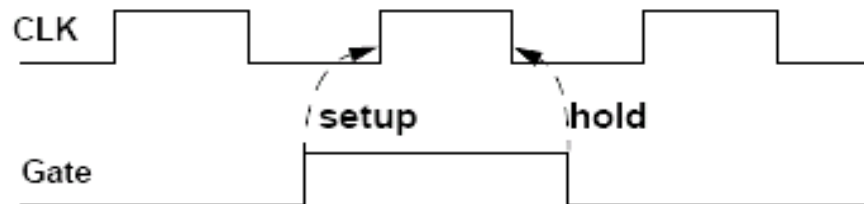


# Timing Checks

## Clock Gating Checks

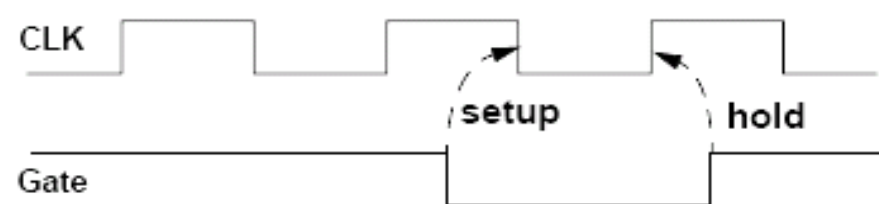
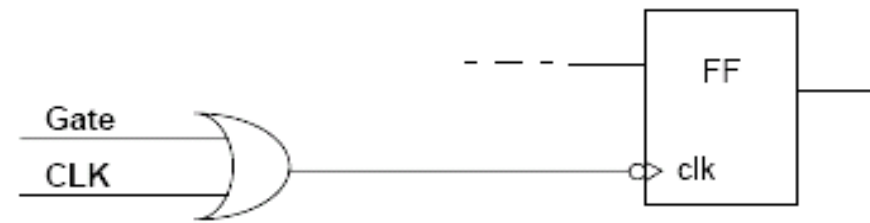
### *Clock-gating checks*

- Setup and hold checks are performed for the gating signal to ensure glitch-free clock
- The clock-gating relationship depends on the functionality of the gate which is gating the clock



Gating signal should only change when the clock is in low state

Example I



Gating signal should only change when the clock is in high state

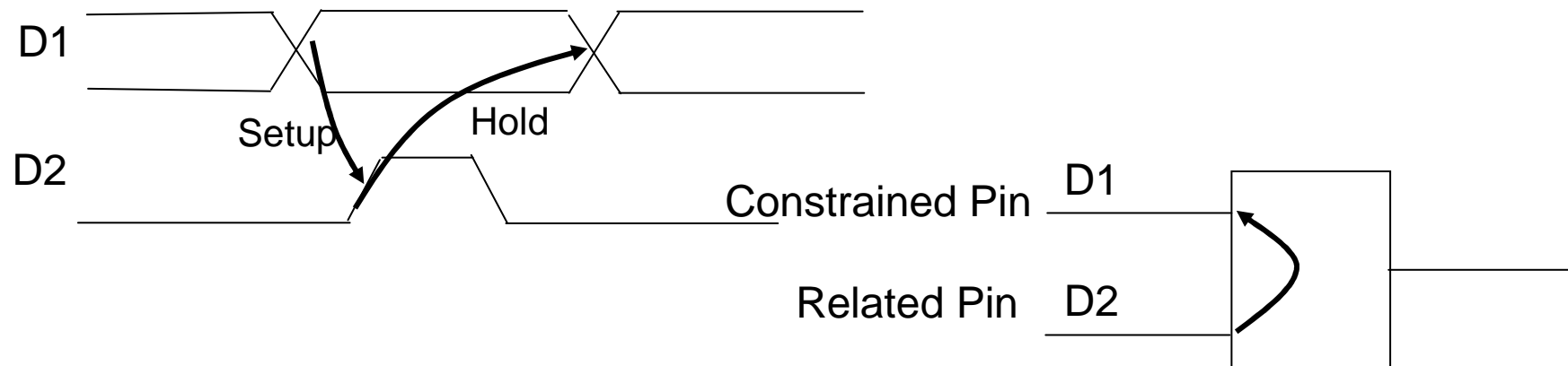
Example II

# Timing Checks

## Data-to-Data Checks

Why Data to Data Checks are required

- Constraints on asynchronous or self-timed circuit interfaces
- Constraints on signals with unusual clock waveforms that cannot be easily specified with the `create_clock` command
- Constraints on skew between bus lines
- Recovery and removal constraints between asynchronous preset and clear input pins
- Constraints on handshaking interface logic



# Timing Exceptions

## ■ False Paths

- Timing Paths that are invalid
  - ◆ Paths between asynchronous clocks
  - ◆ Paths that are static for a particular timing mode

## ■ Multicycle Paths

- Non-default cycle operation

## ■ Logic Setting

- Pins or nets that are tied to 1/0 for a particular timing mode

## ■ Disable Timing

- Timing Arcs that are disabled

# Advanced Topics

## ■ Timing Models

- Extracted Timing Models
- Interface Logic Models
- Quick Timing Models

## ■ Statistical Timing Analysis

# Problem

- **Given corner data below, which combinations are expected to lead to worst and best gate delays?**
  - Process
    - ◆ Slow
    - ◆ Typical
    - ◆ Fast
  - Voltage
    - ◆ 0.9V
    - ◆ 1.0V
    - ◆ 1.1V
  - Temperature
    - ◆ -20C
    - ◆ 27C
    - ◆ 105C



# **Introduction to Digital VLSI Design**

ספרתי VLSI מבוא לתכנון

**STA part 3**

# Overview

---

- **In this era of high performance electronics, timing continues to be a top priority and designers are spending increased effort addressing IC performance.**
- **Two Methods are employed for Timing Analysis:**
  - Dynamic Timing Analysis
  - Static Timing Analysis

# Dynamic Timing Analysis

---

- Traditionally, a dynamic simulator has been used to verify the functionality and timing of an entire design or blocks within the design.
- Dynamic timing simulation requires vectors, a logic simulator and timing information. With this methodology, input vectors are used to exercise functional paths based on dynamic timing behaviors for the chip or block.
- Dynamic simulation is becoming more problematic because of the difficulty in creating comprehensive vectors with high levels of coverage.
- Time-to-market pressure, chip complexity, limitations in the speed and capacity of traditional simulators are all motivating factors for migration towards static timing techniques.

# Static Timing Analysis (STA)

---

- STA is an exhaustive method of analyzing, debugging and validating the timing performance of a design.
- First, a design is analyzed, then all possible paths are timed and checked against the requirements.
- Since STA is not based on functional vectors, it is typically very fast and can accommodate very large designs (multimillion gate designs).
- STA is exhaustive in that every path in the design is checked for timing violations.
- STA does not verify the functionality of a design. Also, certain design styles are not well suited for static approach. For instance, dynamic simulation may be required for asynchronous parts of a design and certainly for any mixed-signal portions.

# Static Timing Analysis (STA)

---

## ■ STA consists of three major steps:

- Break down the design into timing paths (R-R, PI-R, PI-PO & R-PO).
- Delay of each path is calculated
- All path delays are checked against timing constraints to see if it is met.

## ■ STA advantage

- Speed (orders of magnitude faster than dynamic simulation)
- Capacity to handling full chip
- Exhaustive timing coverage
- Vectors are not required

## ■ STA disadvantage

- It is pessimistic (too conservative)
- Reports false paths

## ■ Flow Inputs:

- Gate-level Verilog.
- Constraints (SDC)
- Extracted nets (SPEF)
- Libraries (liberty format - .lib)

# Timing Closure

---

- **Timing Closure is the ability to detect and fix timing problems in the design flow as early as possible.**
- **This is done by checking the correctness of intermediate results through Static Timing Analysis (STA) and also by dynamic timing simulation with SDF back annotation.**
- **In case of failure - which means that the timing goals have not been achieved - modification of timing constraints must be done through well defined loops, re-synthesis and in worst case re-design.**

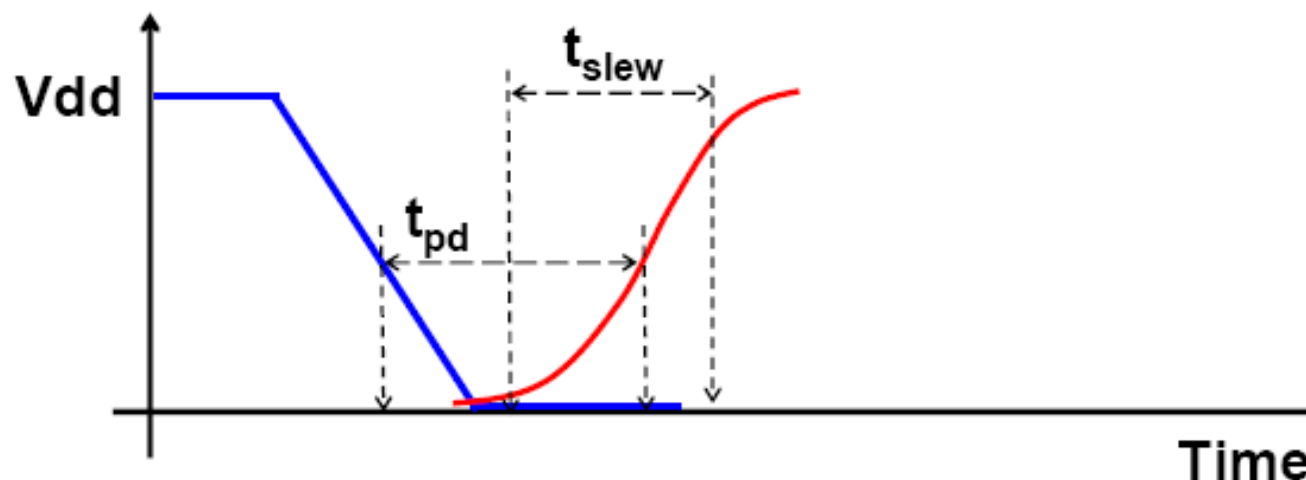
# Cell Timing Characterization

## ■ Delay tables

- Generated using a detailed transistor-level circuit simulator SPICE

### (differential-equations solver)

- Simulate the circuit of the cell for a number of different input slews and load capacitances
  - ◆ Propagation time (50% Vdd at input to 50% at output)
  - ◆ Output slew (10% Vdd at output to 90% Vdd at output)

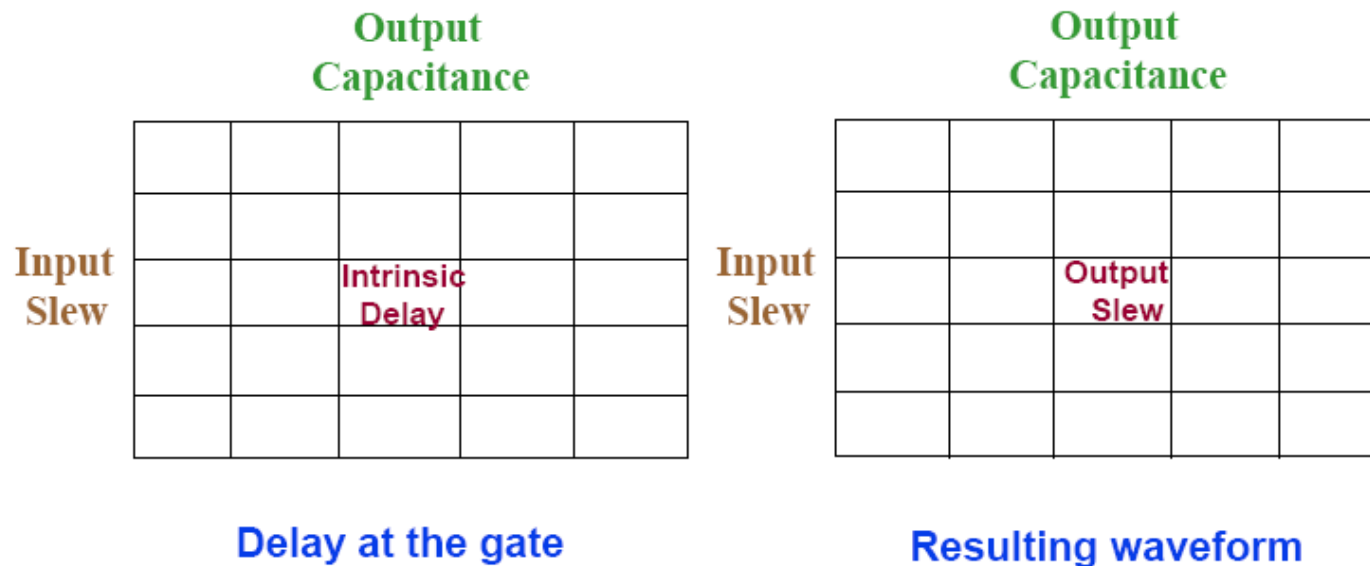


# NLDM

---

## ■ Cell Delay (Non-linear) = $f(\text{CL}, \text{Sin})$ and $\text{Sout} = f(\text{CL}, \text{Sin})$

- Interpolate between table entries
- Interpolation error is usually below 10% of SPICE





# Delay Calculation

Cell Fall

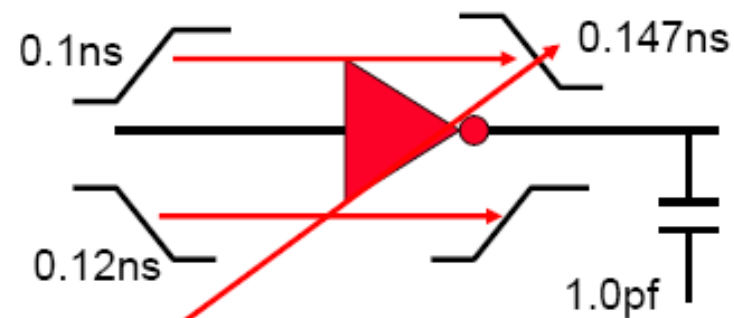
Cap\Tr	0.05	0.2	0.5
0.01	0.02	0.16	0.30
0.5	0.04	0.32	0.60
2.0	0.08	0.64	1.20

Cell Rise

Cap\Tr	0.05	0.2	0.5
0.01	0.03	0.18	0.33
0.5	0.06	0.36	0.66
2.0	0.09	0.72	1.32

Fall Transition

Cap\Tr	0.05	0.2	0.5
0.01	0.01	0.09	0.15
0.5	0.03	0.27	0.45
2.0	0.06	0.54	0.90



Fall delay = 0.178ns  
Rise delay = 0.261ns  
Fall transition = 0.147ns  
Rise transition = ...

# Timing Path Definition

---

- **STA tool does not report delays by net or by cell.**

**Instead it reports by timing paths with constraint.**

- **Valid timing paths:**

- Primary input to Register
- Register to register
- Register to primary output
- Input to output

- **Valid start of a timing path**

- Clock pins of FF
- Primary inputs

- **Valid end of a timing path**

- Data pins of FF
- Primary output ports
- Control pin of gated clock

# Path Delays

---

■ **When delay paths are added, the following factors affect the delays:**

- **Slew propagation** – Ideally, the slew propagation should be timing path specific. However, the STA does not do this. It uses either “worst\_slew” or “worst\_arrival”.
  - ◆ “worst\_slew” – refers to using the slowest transition for signals arriving at a multi-input cell output (fastest transition for min delay mode).  
This is CTE default pessimistic behavior .
  - ◆ “worst\_arrival” – refers to using the input signal that arrives the latest (using the earliest for min delay mode).

# Analysis Modes

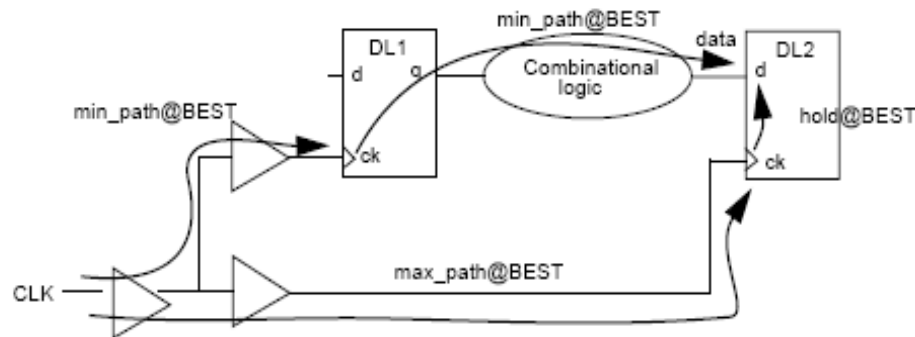
---

- Semiconductor device parameters can vary with conditions such as fabrication process, operating temperature, and power supply voltage.
- The STA tool supports three analysis modes:
  - ◆ Single operating condition – single set of delay parameters is used for the whole circuit, based on one set of process, temperature, and voltage conditions.
  - ◆ Min-Max (BC-WC) operating condition – simultaneously checks the circuit for the two extreme operating conditions, minimum and maximum. For setup checks, it uses maximum delays for all paths. For hold checks, it uses minimum delays.
  - ◆ On-chip-variation mode - conservative analysis that allows both minimum and maximum delays to apply to different paths at the same time. For a setup check, it uses maximum delays for the launch clock path and data path, and minimum delays for the capture clock path. For a hold check, it uses minimum delays for the launch clock path and data path, and maximum delays for the capture clock path.

# Single Operating Condition

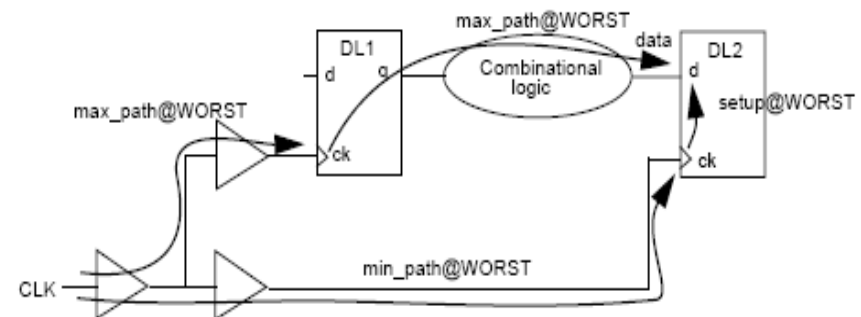
- Single set of delay parameters for the whole circuit, based on one set of process, temperature, and voltage conditions.

Hold



```
setAnalysisMode -single  
setAnalysisMode -hold  
setOpCond BEST -library fast.lib
```

Setup

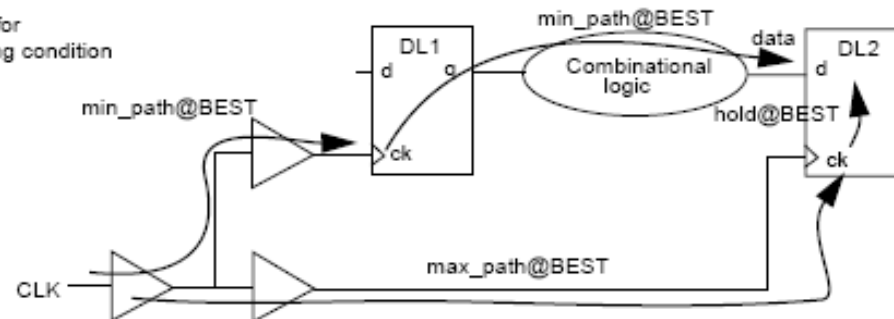


```
setAnalysisMode -single  
setAnalysisMode -setup  
setOpCond WORST -library slow.lib
```

# Best case/Worst case Analysis

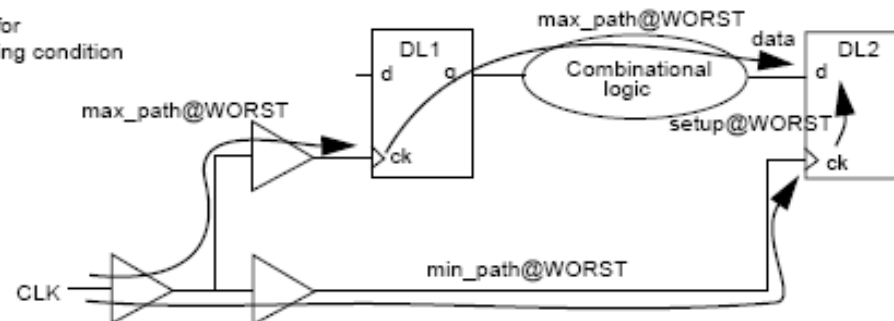
- Simultaneous checks of extreme operating conditions, minimum and maximum.
- For setup checks, it uses maximum delays for all paths.
- For hold checks, it uses minimum delays for all paths.

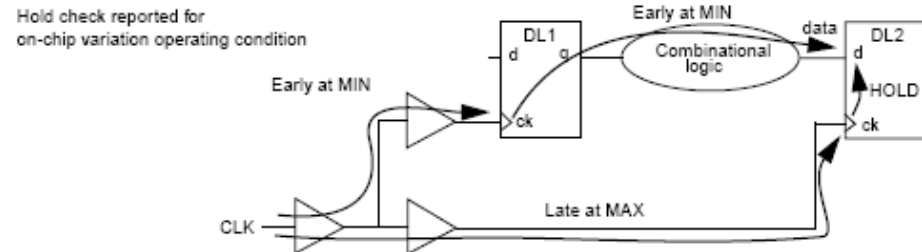
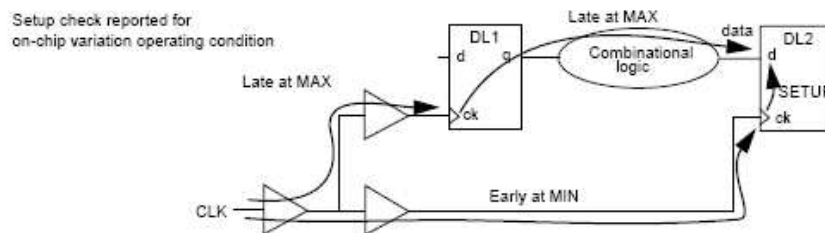
Timing path reported for the best-case operating condition



```
setAnalysisMode -bcWc
setAnalysisMode -setup
setOpCond -min Best -minLibrary fast.lib
           -max Worst -maxLibrary slow.lib
```

Timing path reported for the worst-case operating condition





## setAnalysisMode –onChipVariation

# Derating

---

- Minimum and Maximum delays can be adjust by specified factors to model the effects of operating conditions. This adjustment of calculated delays is called derating.
- Derating affects the delay and slack values reported by report\_timing.

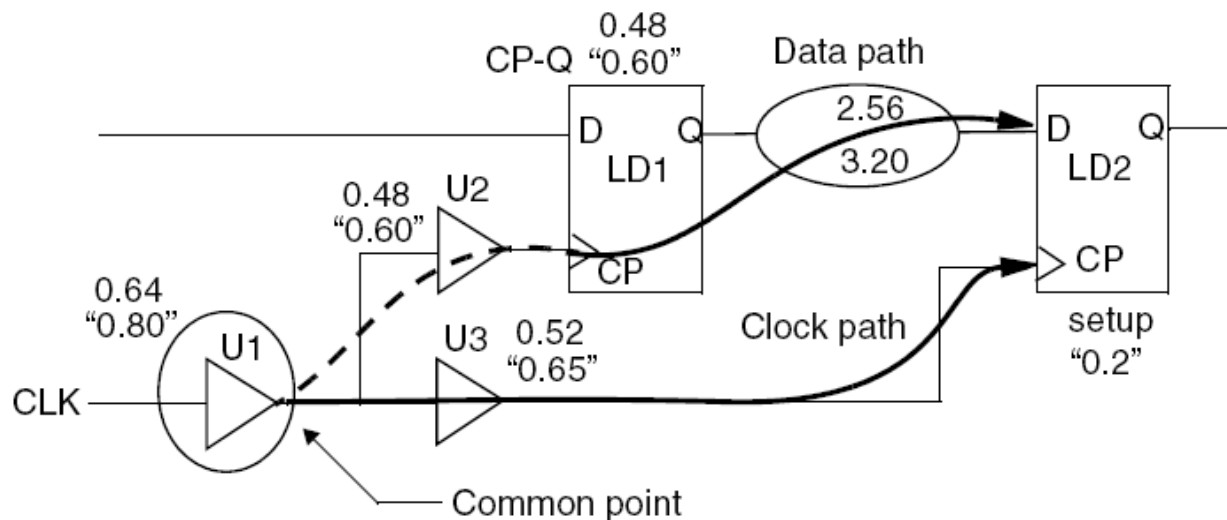
```
setTimingDerate -max -early 0.8 -late 1.0  
setTimingDerate -min -early 1.0 -late 1.1
```



# Clock Reconvergence Pessimism Removal (CRPR)

- When launching and capturing clock share common path, the common path min delay and max delay will add additional pessimism to both setup and hold analysis.

CRPR can be used to remove this pessimism.



```
setAnalysisMode -crpr -onChipVariation
set_global timing_remove_clock_reconvergence_pessimism true
```

# Timing exceptions

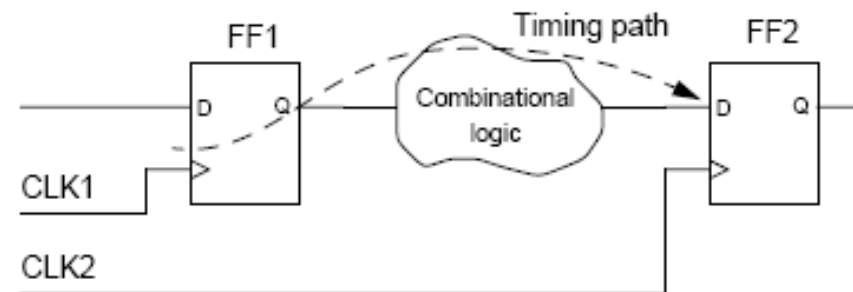
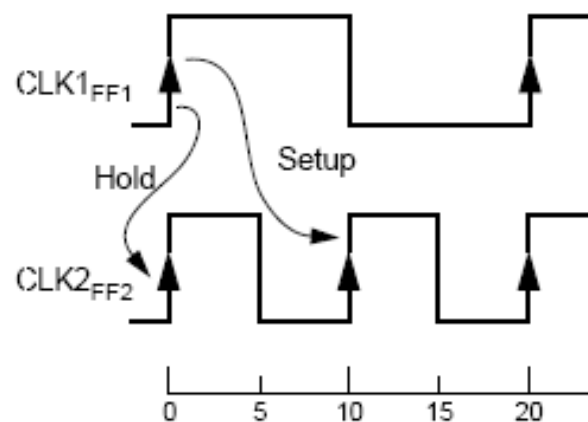
---

## ■ Timing exception includes the following:

- **False Path**- Use the *set\_false\_path* command to specify a logic path that exists in the design but should not be analyzed. Setting a false path removes the timing constraints on the path.
- **Multiple Cycle Path** - Use the *set\_multicycle\_path* command to specify the number of clock cycles required to propagate data from the start to the end of the path.
- **Min/Max Delay** - Use the *set\_max\_delay* and *set\_min\_delay* commands to override the default setup and hold constraints with specific maximum and minimum time values.

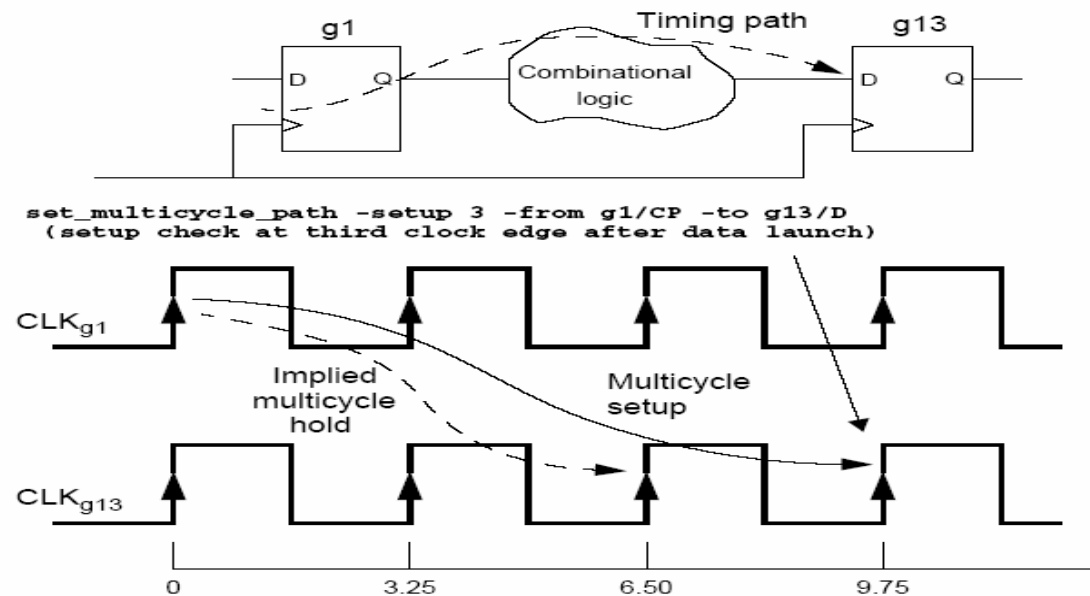
# Setup/Hold Analysis (in the absence of timing exceptions)

- **Setup check** - verifies that the data launched from FF1 at time=0 arrives at the D input of FF2 in time for the capture edge at time=10. If the data takes too long to arrive, it is reported as a setup violation.
- **Hold check** - verifies that the data launched from FF1 at time 0 does not get propagated so soon that it gets captured at FF2 at the clock edge at time 0. If the data arrives too soon, it is reported as a hold violation.



# Multiple Cycle Setup

- If data is launched every 3 cycles, then setup is checked against the third rising edge (9.75) and hold is checked against next rising edge (which is CLK<sub>g1</sub> at 6.50).
- STA tool verifies that the data launched by the setup launch edge is not captured by the previous capture edge. So the default hold check for multi-cycle setup is capture edge minus one.

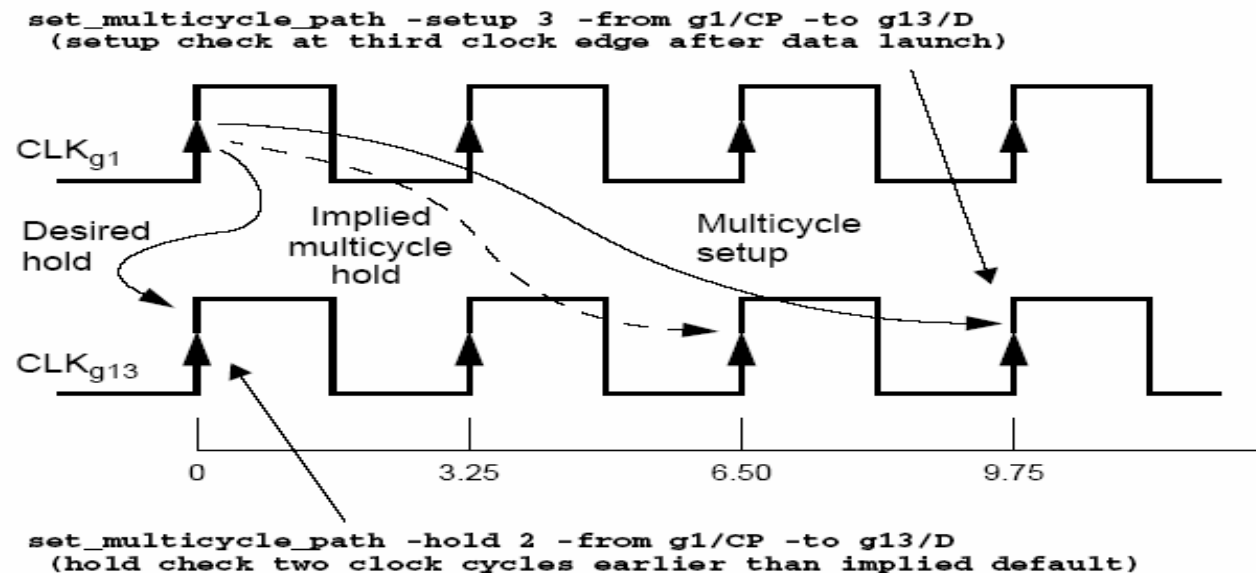


# Multiple Cycle Hold

- The number after the *-hold* option specifies the number of cycles to move the hold check backward from the default position implied by the setup check.

A positive number moves the check backward by the specified number of cycles.

Specifying zero does not change the hold check time.



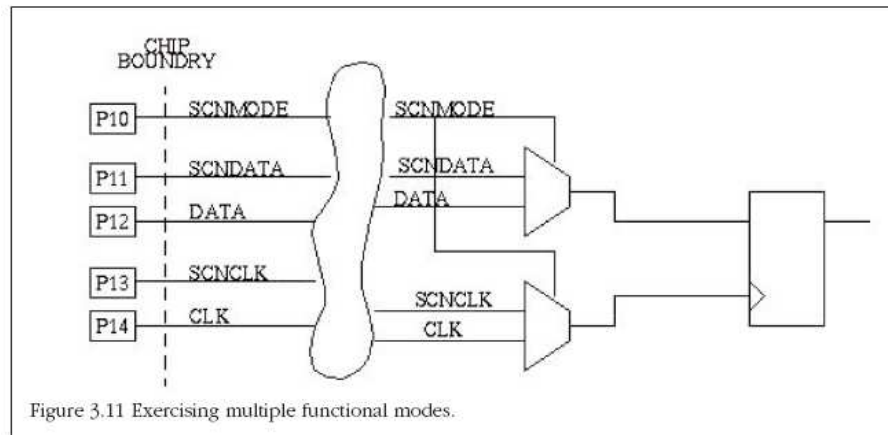
# **Recovery/Removal check**

---

- **Timing checks which are related to asynchronous input pin of a flip flop.**
- **Although a flip-flop is asynchronously set or clear , the negation from its reset state is synchronous .**
- **A recovery timing check specifies a minimum amount of time allowed between the release of a asynchronous signal from the active state to the next active clock edge .**
- **A removal timing check specifies the minimum amount of time between an active edge and the release of an asynchronous control signal.**

# Case Analysis

- Case analysis allows timing analysis to be performed using logic constants or logic transitions (rising or falling) on ports or pins, to limit the signal propagated through the design.
- Case analysis is a path-pruning mechanism and is most commonly used for timing the device in a given operational configuration or functional mode. For example, case analysis can be used to compare normal circuit operation against scan or BIST operation.



# Timing Models

---

- Timing extraction plays an important role in hierarchical top-down flow and bottom-up IP authoring flow by reducing the complexity of timing verification and by providing a level of abstraction which hides the implementation details of IP blocks.
- Three most desired features in timing extraction are accuracy, efficiency, and usability. The model must preserve the timing behavior of the original circuit and produce accurate results.
- Three types of models can be generated:
  - Quick Timing Model (QTM)
  - Extracted Timing Model (ETM)
  - Interface Logic Model (ILM)



# QTM

---

- **A temporary model used early in the design cycle for a block that has no netlist available. QTM creation is faster than writing ad-hoc model . The model contains both min and max time arc for setup and hold checks.**
- **Check consistency between blocks' constraints and updates boundary constraints (after each iteration of synthesis) The netlist used for QTM generation can be easily generated (low effort RTL mapping) since existence or absence of timing arc is independent from the logic/physical design.**
- **Inputs**
  - Constraints (SDC)
  - Configuration file
  - Header file
- **The QTM model is generated using Black Box commands.**

Using this command set allows to define timing arcs and electrical data (i.e. output driver, input load,...)

# ILM

- ILMs embody a structural approach to model generation, where the original gate-level netlist is replaced by another gate-level netlist that contains only the interface logic of the original netlist.
- Interface logic contains all circuitry leading from I/O ports to edge-triggered registers called interface registers. The clock tree leading to interface registers is preserved in an ILM. Logic that is only contained in register-to-register paths on a block is not in an ILM.

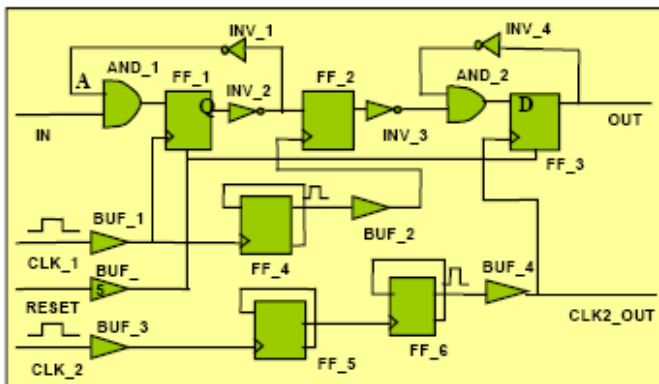


Figure1: Example gate-level netlist.

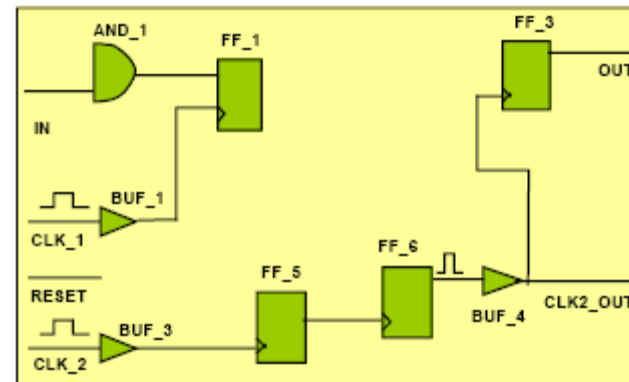
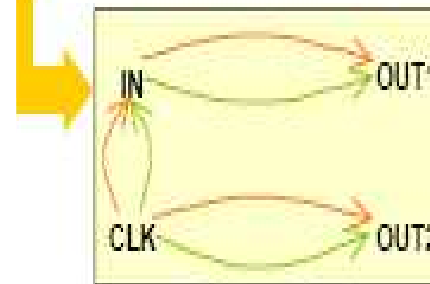
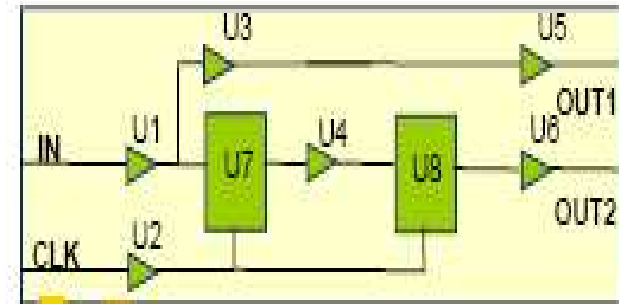


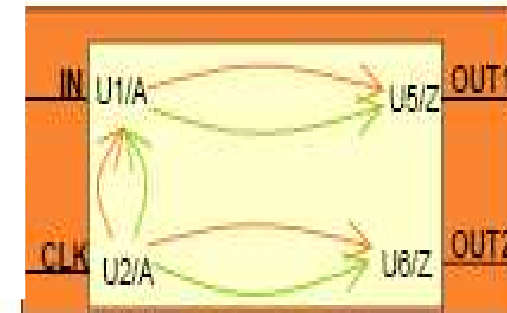
Figure2: ILM for example gate-level netlist.

# ETM

- Extracted timing models differ from ILMs in that the interface logic for a block is replaced by context-independent timing relationships between pins on a library cell
- The extracted library cell contains timing arcs between external pins. Internal pins are introduced only when there are clocks defined on internal pins of the design



Library cell ETM



Wrapper and core cell ETM

# Analysis Modes

*Table 11-1 Timing Parameters Used for Setup Checks*

Analysis mode	Launch clock path	Data path	Capture clock path
Single operating condition	Late clock, maximum delay in clock path, single operating cond. (no derating)	Maximum delay, single operating cond. (no derating)	Early clock, minimum delay in clock path, single operating cond. (no derating)
Best-case/worst-case mode	Late clock, maximum delay in clock path, late derating, worst-case operating cond.	Maximum delay, late derating, worst-case operating cond.	Early clock, minimum delay in clock path, early derating, worst-case operating cond.
On-chip variation mode	Late clock, maximum delay in clock path, late derating, worst-case operating cond.	Maximum delay, late derating, worst-case operating cond.	Early clock, minimum delay in clock path, early derating, best-case operating cond.

# Analysis Modes

*Table 11-2 Timing Parameters Used for Hold Checks*

Analysis mode	Launch clock path	Data path	Capture clock path
Single operating condition	Early clock, minimum delay in clock path, single operating cond. (no derating)	Minimum delay, single operating cond. (no derating)	Late clock, maximum delay in clock path, single operating cond. (no derating)
Best-case/worst-case mode	Early clock, minimum delay in clock path, early derating, best-case operating cond.	Minimum delay, early derating, best-case operating cond.	Late clock, maximum delay in clock path, late derating, best-case operating cond.

Analysis mode	Launch clock path	Data path	Capture clock path
On-chip variation mode	Early clock, minimum delay in clock path, early derating, best-case operating cond.	Minimum delay, early derating, best-case operating cond.	Late clock, maximum delay in clock path, late derating, worst-case operating cond.