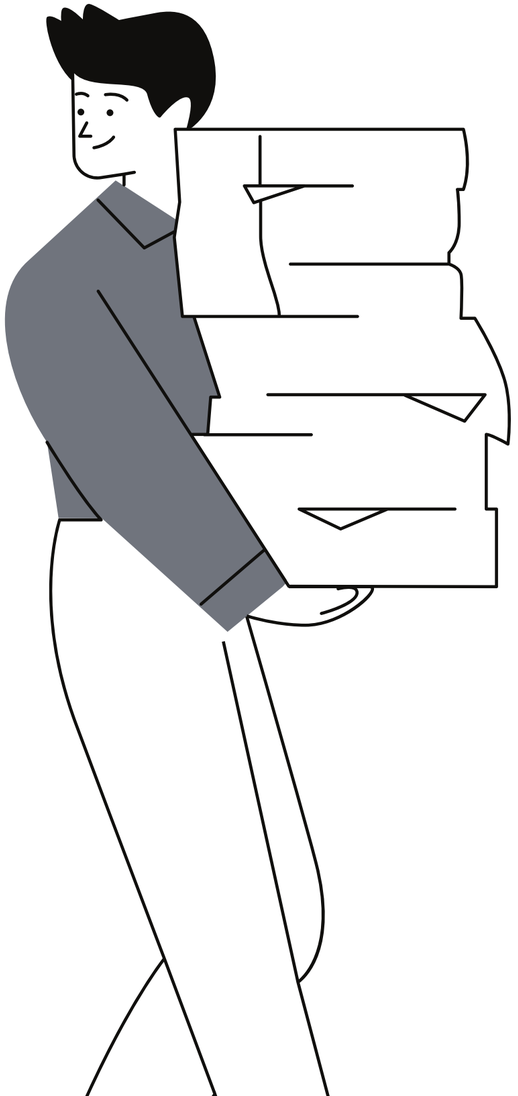
# Static Timing Analysis

>>> Short Notes
>>> Concept
>>> Numericals

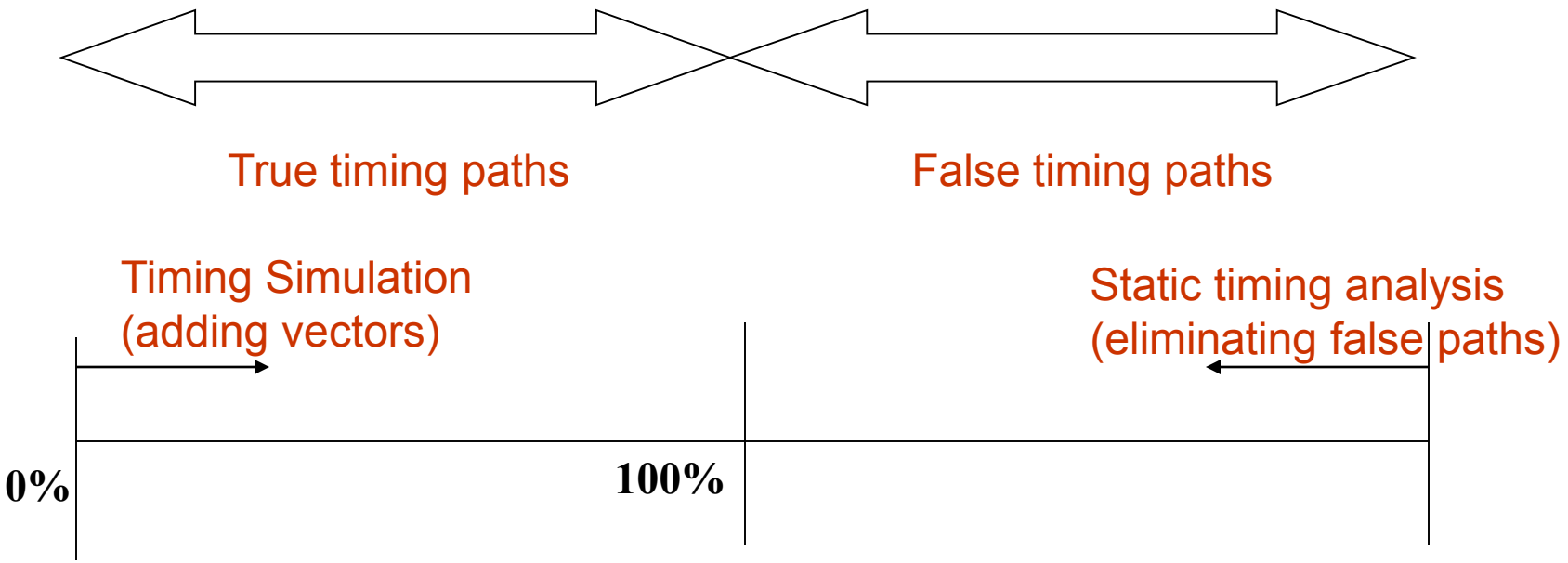



- HARSHIT GUPTA

# Introduction

- Effective methodology for verifying the timing characteristics of a design without the use of test vectors
- Conventional verification techniques are inadequate for complex designs
  - Simulation time using conventional simulators
    - Thousands of test vectors are required to test all timing paths using logic simulation
  - Increasing design complexity & smaller process technologies
    - Increases the number of iterations for STA

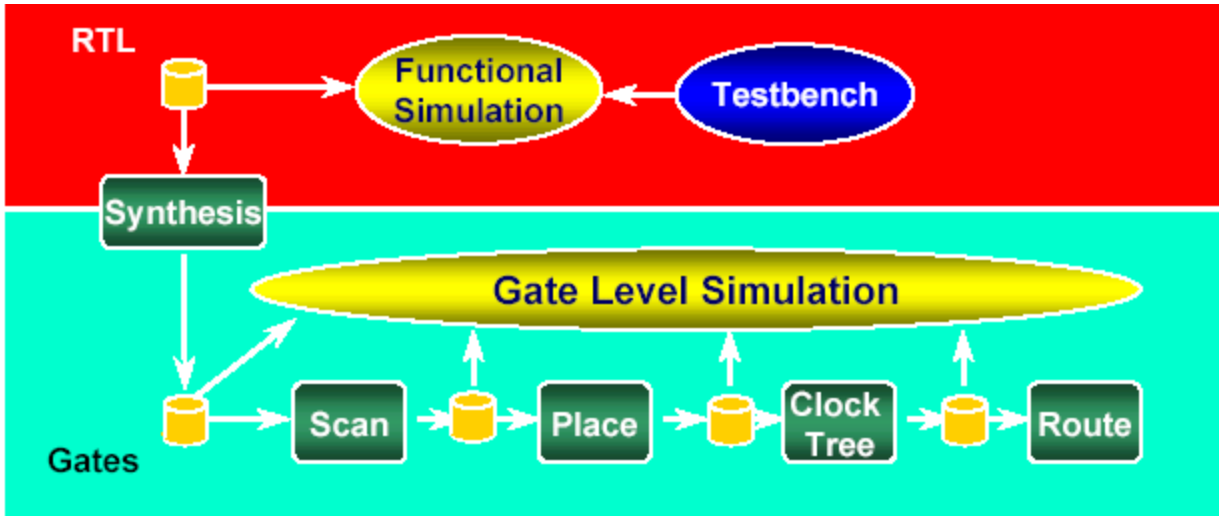
# Simulation vs. Static timing




STA approach typically takes a fraction of the time it takes to run logic simulation on a large design and guarantees 100% coverage of all true timing paths in the design without having to generate test vectors

# OVERVIEW

## Previous Verification Flow



# OVERVIEW

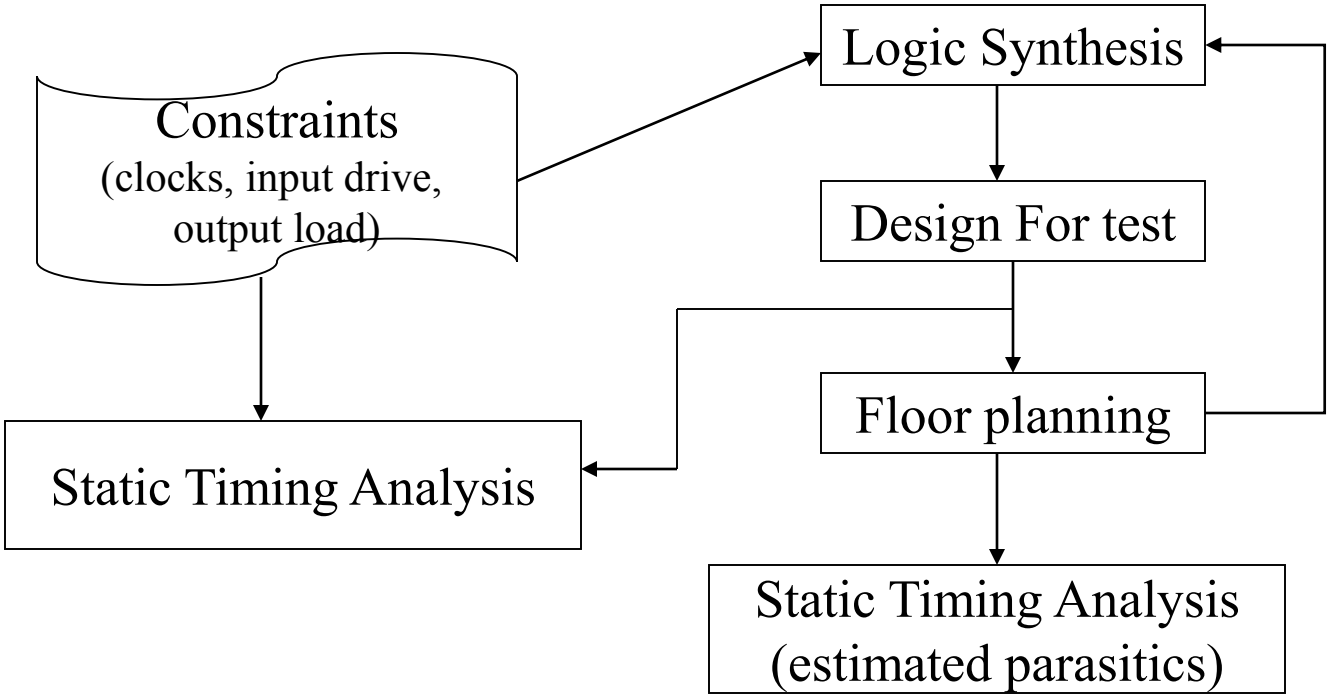
- Requires extensive vector creation
- Valid for FPGAs and smaller ASICs
- Falls apart on multi-million gate ASICs

# What is Static Timing Analysis?

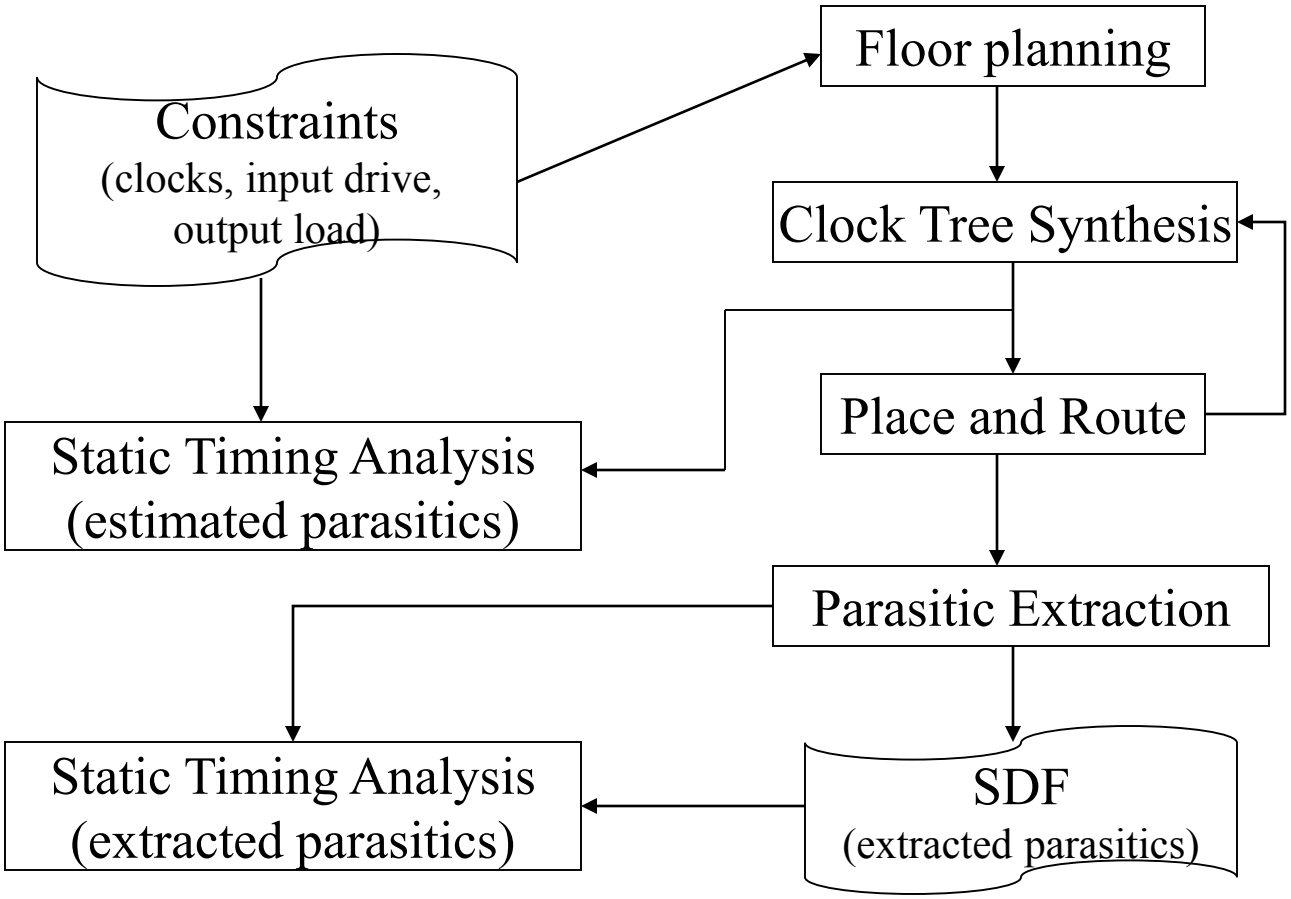
Static Timing Analysis is a method for determining if a circuit meets timing constraints without having to simulate

- Much faster than timing-driven, gate-level simulation
- Proper circuit functionality is not checked
- Vector generation NOT required

# STA in ASIC Design Flow – Pre layout



# STA in ASIC Design Flow – Post Layout



# 2 Types of Timing Verification

**Dynamic Timing Simulation**

**Advantages**

- Can be *very* accurate (spice-level)

**Disadvantages**

- Analysis quality depends on stimulus vectors
- Non-exhaustive, slow

Examples:

VCS,Spice,ACE

# 2 Types of Timing Verification

**Static Timing Analysis (STA)**

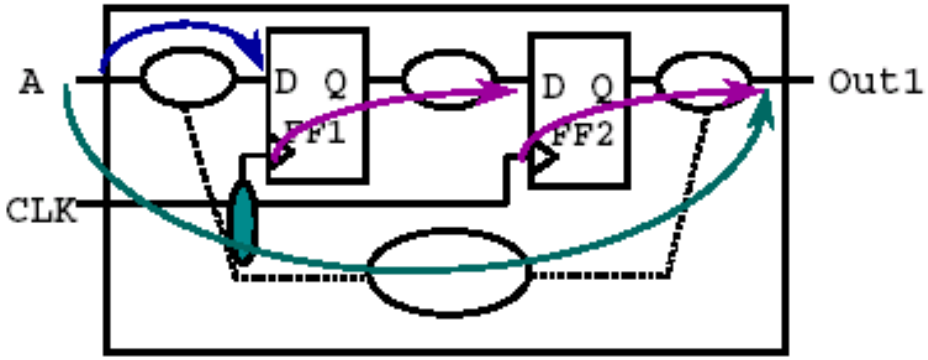
**Advantages**

- Fast, exhaustive
- Better analysis checks against timing requirements

**Disadvantage**

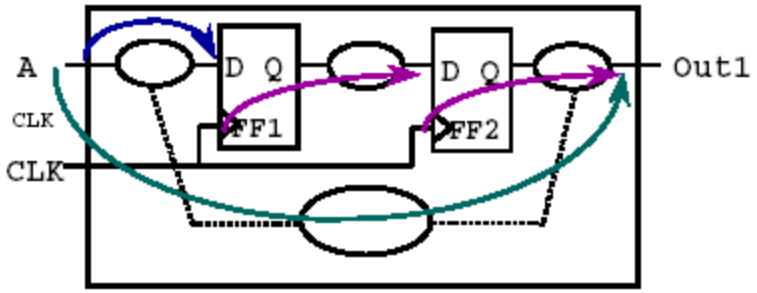
- Less accurate
- Must define timing requirements/exceptions
- Difficulty handling asynchronous designs, false paths

# Three Steps in Static Timing Analysis




- Circuit is broken down into sets of timing paths
- Delay of each path is calculated
- Path delays are checked to see if timing constraints have been met

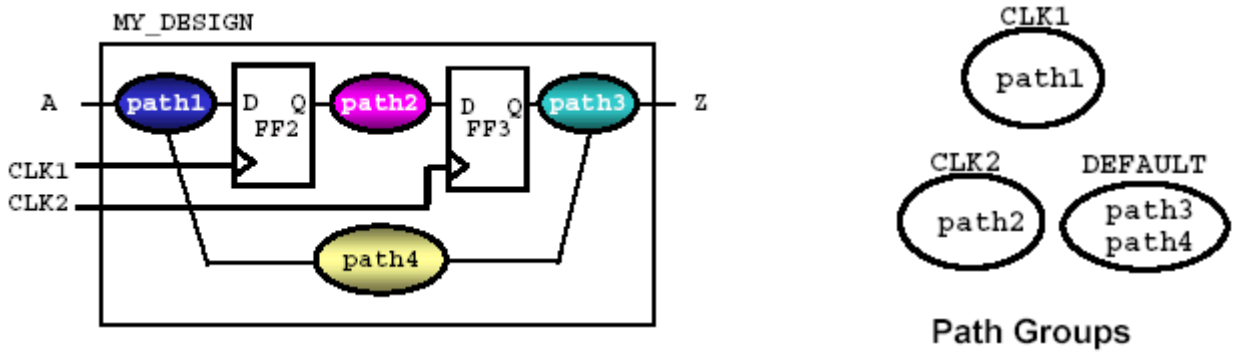
# What is a Timing Path?




A Timing Path is a point-to-point path in a design which can propagate data from one flip-flop to another

- Each path has a **start point** and an **endpoint**
- **Start point:**
  - Input ports Clock pins of flip-flops
- **Endpoints:**
  - Output ports Data input pins of flip-flops

# Organizing Timing Paths Into Groups

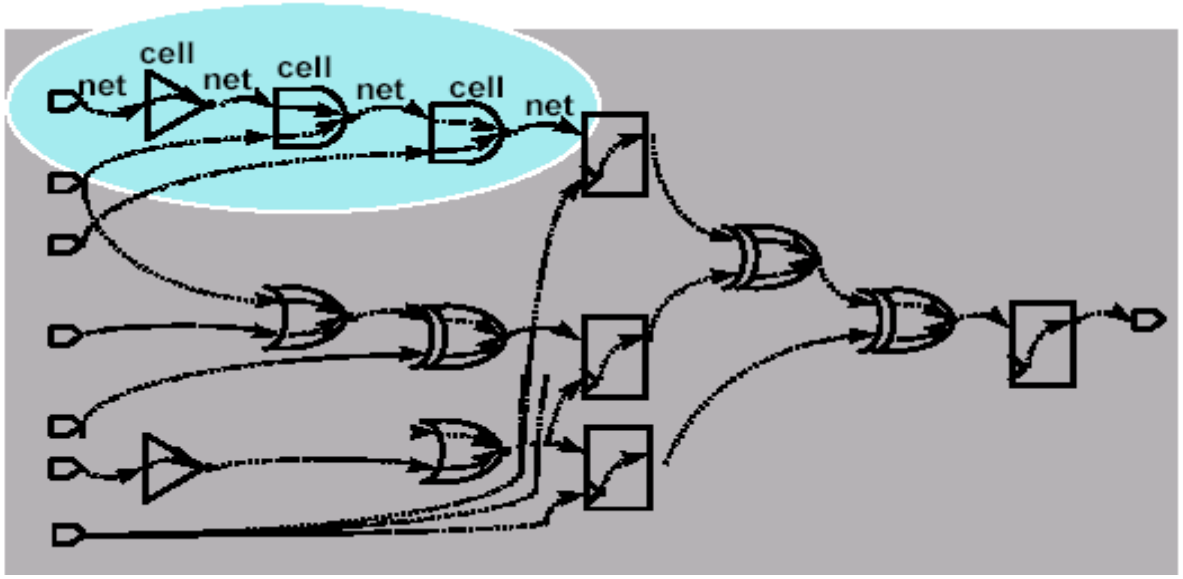



Timing paths are grouped into path groups by the clocks controlling their endpoints

**Synthesis tools like PrimeTime and Design Compiler organize timing reports by path groups**

# Net and Cell Timing Arcs

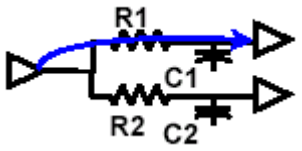
The actual path delay is the sum of net and cell delays along the timing path



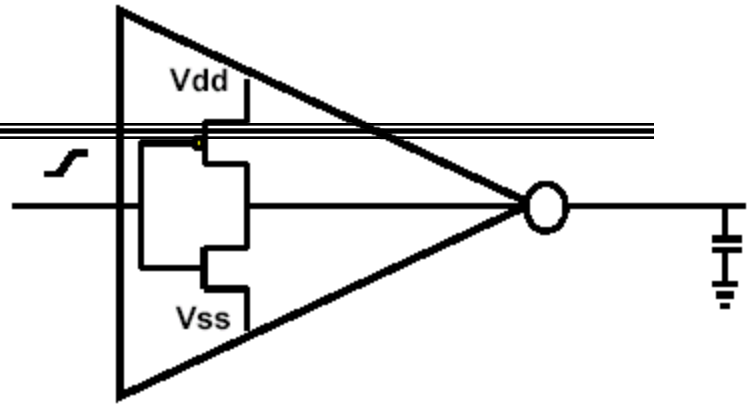
# Net and Cell Delay

"Net Delay" refers to the total time needed to charge or discharge all of the parasitics of a given net

- Total net parasitics are affected by
  - net length
  - net fanout
- Net delay and parasitics are typically
  - Back-Annotated (Post-Layout) from data obtained from an extraction tool
  - Estimated (Pre-Layout)



# Cell Delay




In ASICs, the delay of a cell is affected by:

- The input transition time (or slew rate)
- The total load "seen" by the output transistors

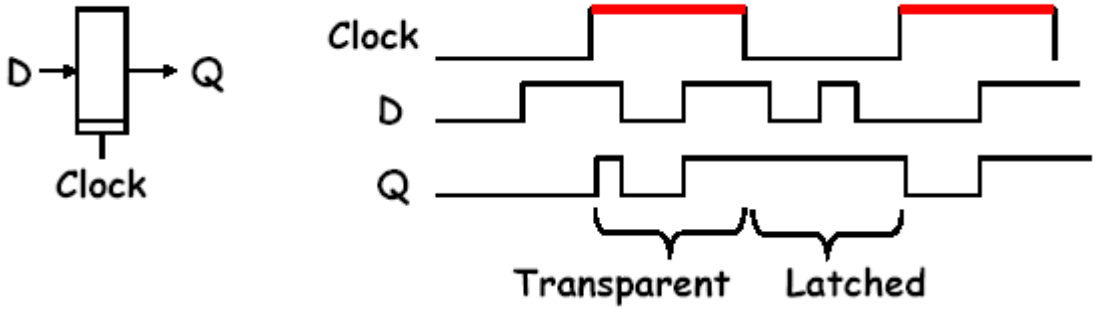
Net capacitance and "downstream" pin capacitances

- These will affect how quickly the input and output transistors can "switch"
- Inherent transistor delays and "internal" net delays

# Clocked Storage Elements

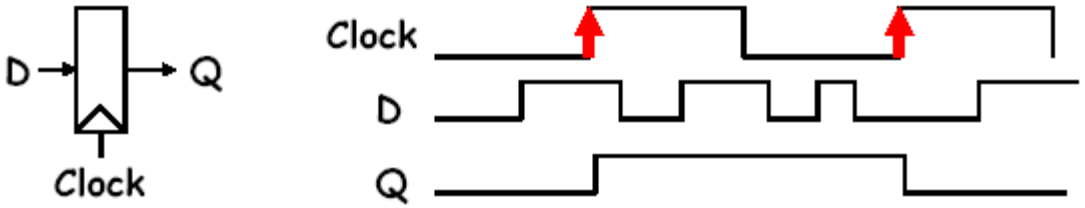
**Transparent Latch, Level Sensitive**

**– data passes through when clock high, latched when clock low**

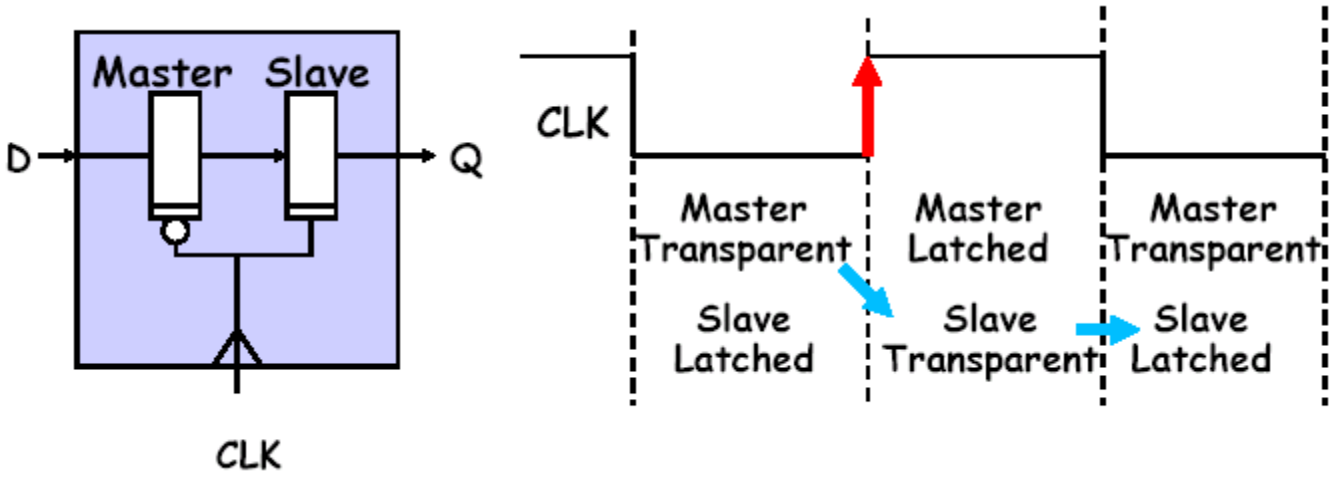


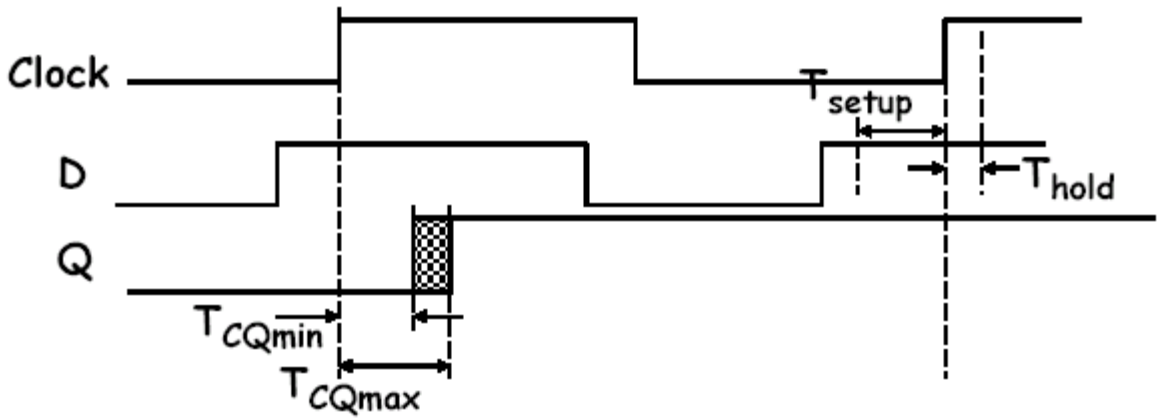

**D-Type Register or Flip-Flop, Edge-Triggered**

**– data captured on rising edge of clock, held for rest of cycle**



# Flip-Flops


Master
Slave
D
Q
CLK
CLK
Master Transparent
Slave Latched
Master Latched
Slave Transparent
Master Transparent
Slave Latched


Clock
D
Q
T setup
T hold
T CQmin
T CQmax

# Basic terminologies

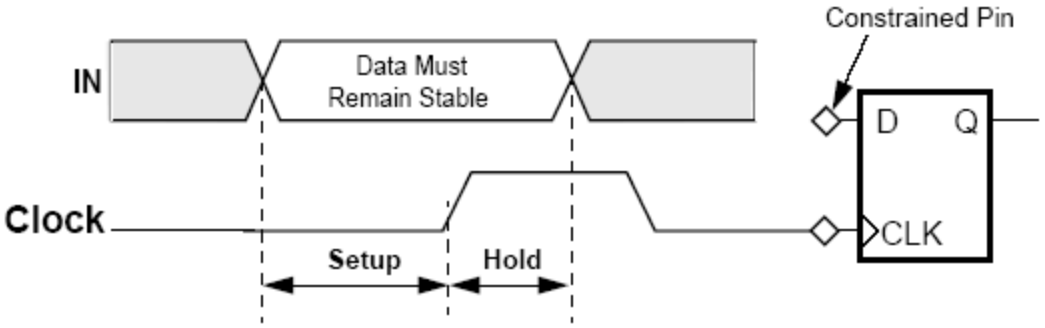
- Pulse Width
- Setup & Hold times
- Signal slew
- Clock latency
- Clock Skew
- Input arrival time
- Output required time
- Slack and Critical path
- Recovery & Removal times
- False paths
- Multi-cycle paths

# Pulse Width

- Pulse width
  - It is the time between the active and inactive states of the same signal

# Setup and Hold time

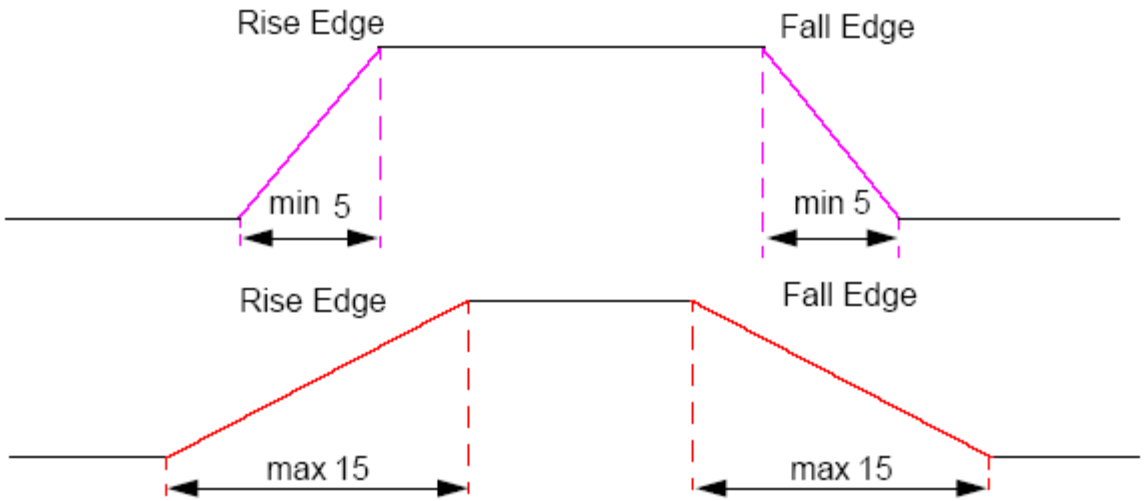
- Setup time
  - For an edge triggered sequential element, the setup time is the time interval before the active clock edge during which the data should remain unchanged
- Hold time
  - Time interval after the active clock edge during which the data should remain unchanged




Both the above 2 timing violations can occur in a design when clock path delay > data path delay

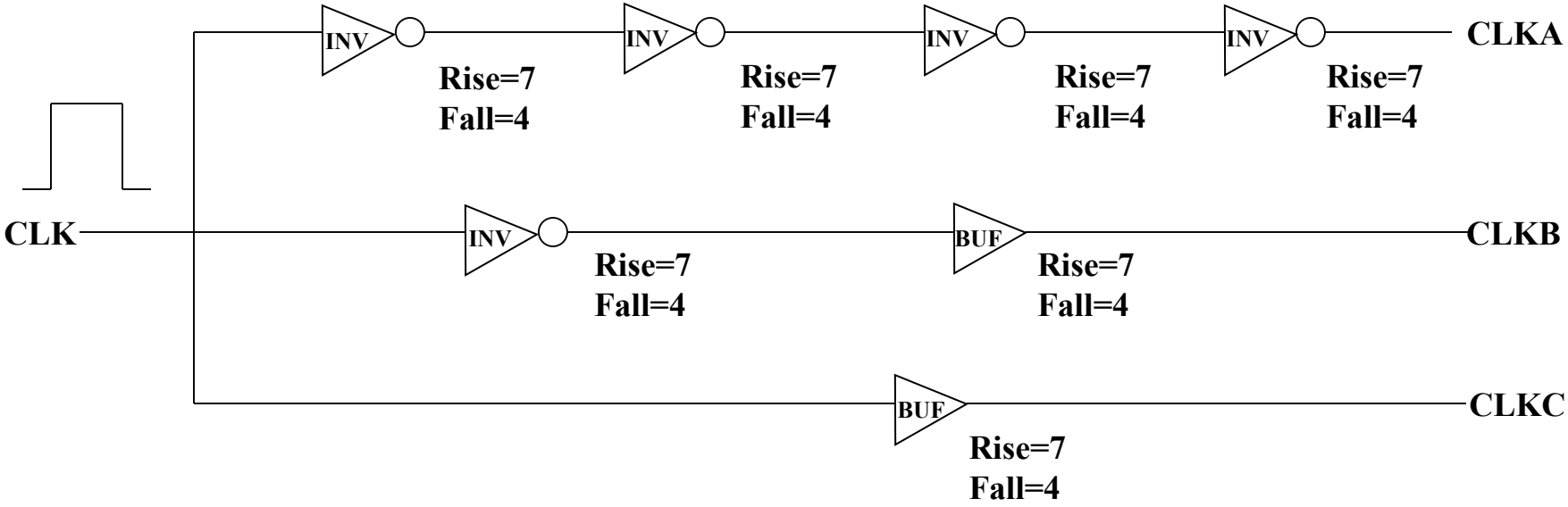
# Signal Slew

- Signal (Clock/Data) slew
  - Amount of time it takes for a signal transition to occur
  - Accounts for uncertainty in Rise and fall times of the signal
  - Slew rate is measured in volts/sec



# Clock Latency

- Clock Latency
  - Difference between the reference (source) clock slew to the clock tree endpoint signal slew values
  - Rise latency and fall latency are specified



# Clock Latency

| | |
|---|---|
| CLK to CLKA | Min Latency Rise = 4 + 7 + 4 + 7 = 22 (maximum value)<br>Min Latency Fall = 7 + 4 + 7 + 4 = 22 |
| CLK to CLKB | Min Latency Rise = 4 + 4 = 8<br>Min Latency Fall = 7 + 7 = 14 |
| CLK to CLKC | Min Latency Rise = 7<br>Min Latency Fall = 4 (minimum value) |

Average = (22 + 22 + 8 + 14 + 7 + 4) / 6 = 12.83

Standard Deviation = $\sqrt{\left[\left(22^2 + 22^2 + 8^2 + 14^2 + 7^2 + 4^2\right) \S\, 6\right] - (12.83)^2} = 7.13$

Therefore:

| Minimum Latency | | | |
|---|---|---|---|
| Minimum | Maximum | Average | Std. Dev. |
| 4 | 22 | 12.83 | 7.13 |

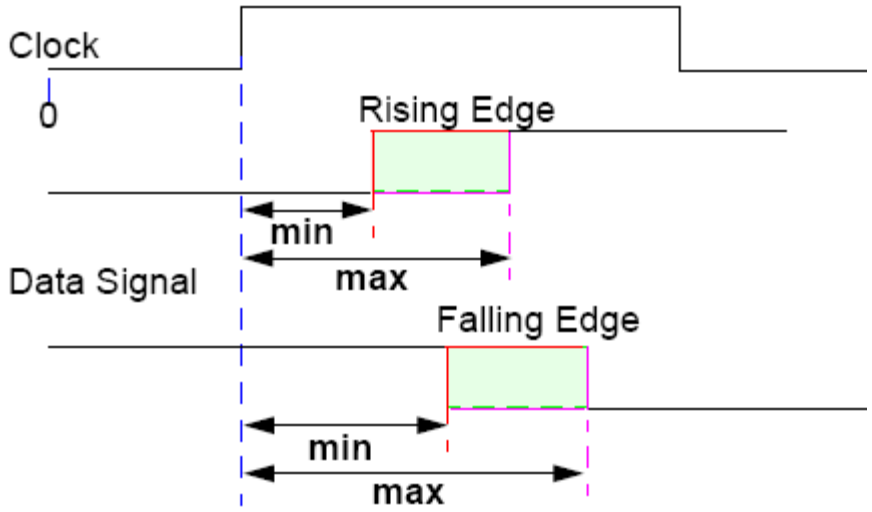
# Clock Skew

- Clock Skew is a measure of the difference in latency between any two leaf pins in a clock tree.
  - between CLKA and CLKB
    rise = 22-8 = 14
    fall = 22-14 = 8
  - between CLKB and CLKC
    rise = 8-7 = 1
    fall = 14-4 = 10
  - between CLKA and CLKC
    rise = 22-7 = 15
    fall = 22-4 = 18

  It is also defined as the difference in time that a single clock signal takes to reach two different registers

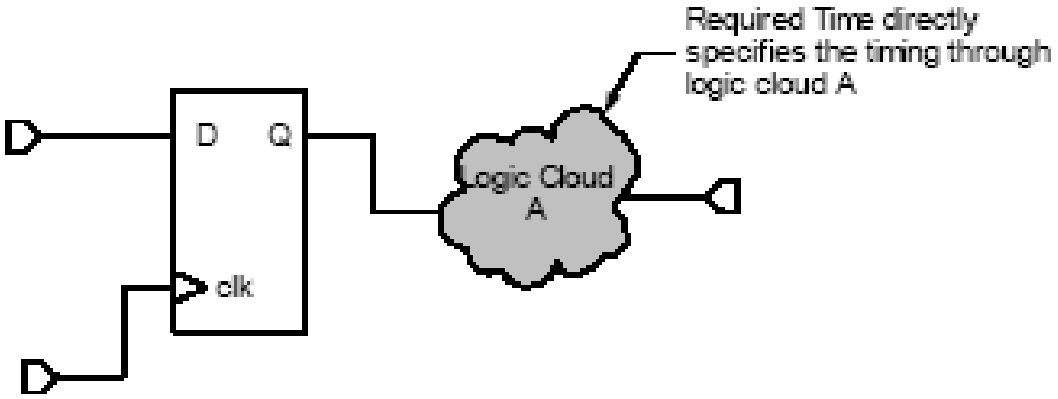
# Input Arrival time

- Input Arrival time
  - An arrival time defines the time interval during which a data signal can arrive at an input pin in relation to the nearest edge of the clock signal that triggers the data transition



# Output required time

- Output required time
  - Specifies the data required time on output ports.

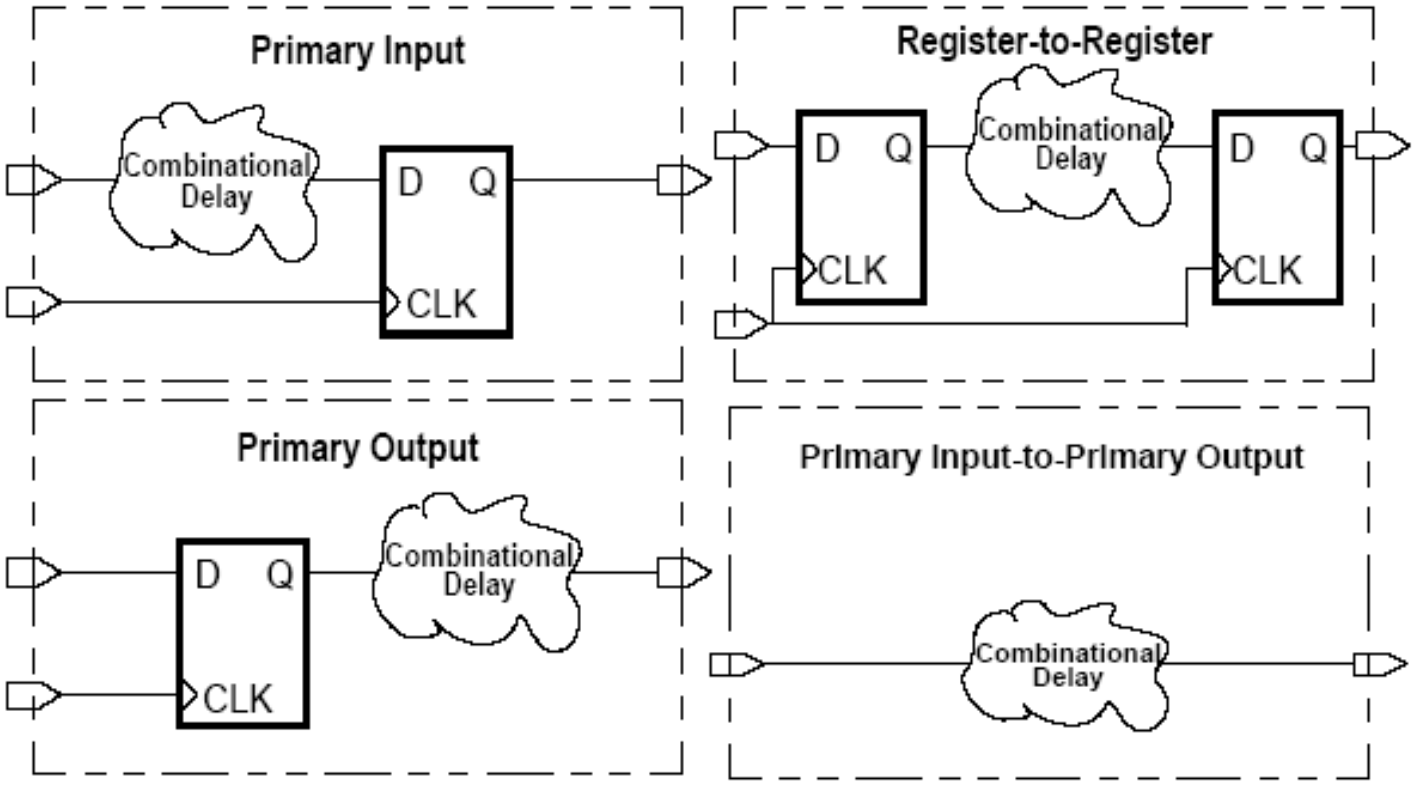


# Slack and Critical path

- Slack
  - It is the difference between the required (constraint) time and the arrival time (inputs and delays).
  - Negative slack indicates that constraints have not been met, while positive slack indicates that constraints have been met.
  - Slack analysis is used to identify timing critical paths in a design by the static timing analysis tool

- Critical path
  - Any logical path in the design that violates the timing constraints
  - Path with a negative slack

# Slack Analysis – Data Path types



# Slack analysis – data path types

- Primary input-to-register paths
  - Delays off-chip + Combinational logic delays up to the first sequential device.
- Register-to-primary output paths
  - Start at a sequential device
  - CLK-to-Q transition delay + the combinational logic delay + external delay requirements
- Register-to-register paths
  - Delay and timing constraint (Setup and Hold) times between sequential devices for synchronous clocks + source and destination clock propagation times.
- Primary input-to-primary output paths
  - Delays off-chip + combinational logic delays + external delay requirements.

# Hold Slack calculation

- Actual data arrival time definition

  Data Input Arrival Timemin + Data path delaymin

  If the data path starts in a primary input,

  Data Input arrivalmin = Input arrival timemin

  If the data path starts at a register,

  (Source Clock Edgemin + Source Clock Path Delaymin) = Data Input Arrivalmin

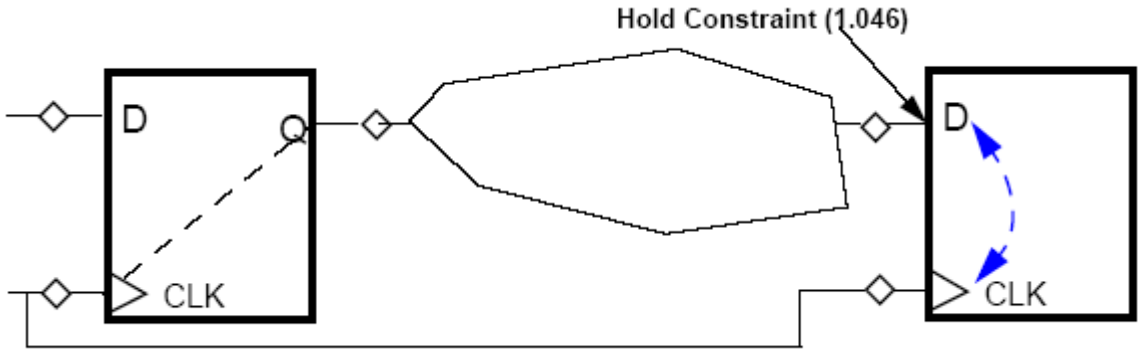
- Required Stability time definition

  (Destination Clock Edgemax + Destination Clock Path Delaymax) + Hold = Required Stability Timemax

- Hold Slack definition

  Actual Data Arrivalmin - Required Stability Timemax

# Calculate the hold slack

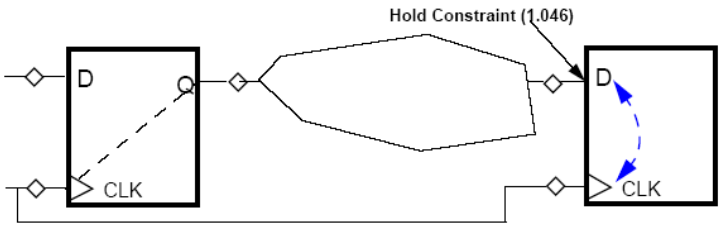



Source Clock signal timing parameters:
Min Edge = 8.002 ns
Min clock path delay = 0.002 ns

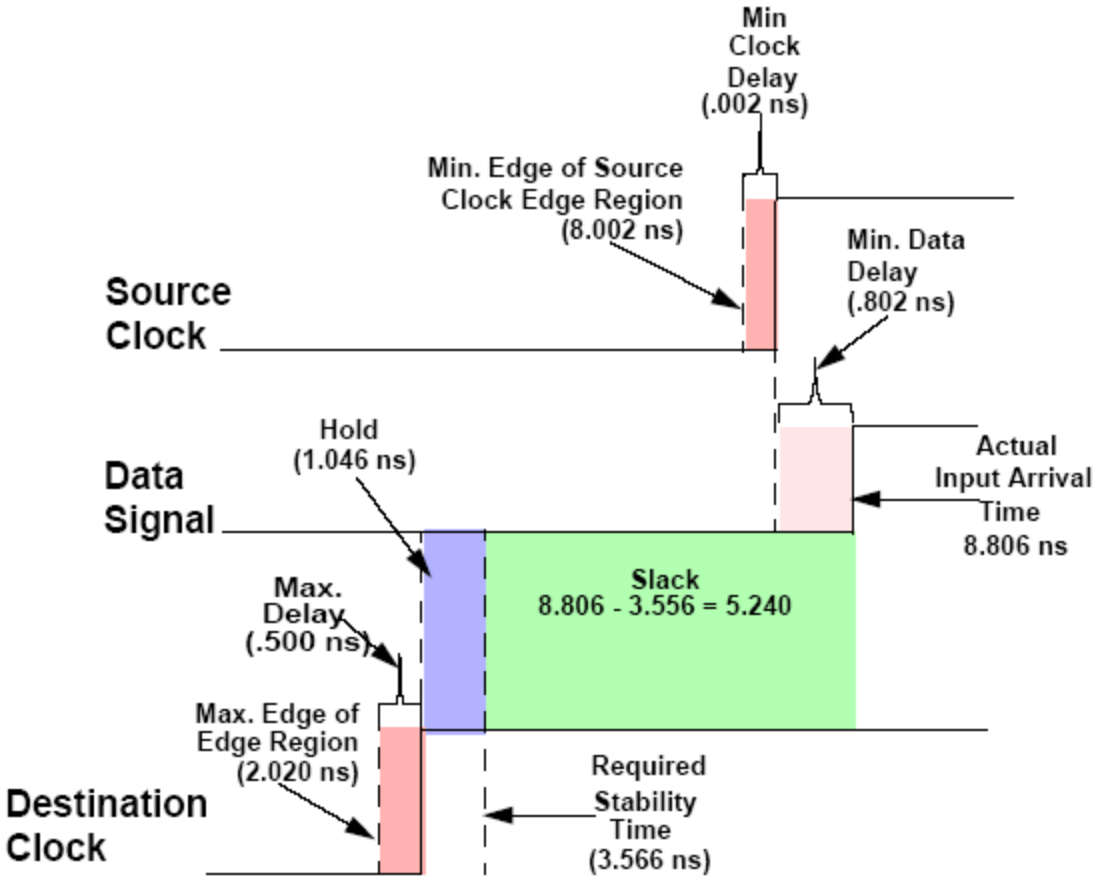
Min Data path delay = 0.802 ns
Hold time constraint = 1.046 ns

Destination Clock signal timing parameters:
Max Edge = 2.020 ns
Max clock path delay = 0.500 ns

# Hold slack calculation




Source Clock signal timing parameters:
Min Edge = 8.002 ns
Min clock path delay = 0.002 ns
Min Data path delay = 0.802 ns
Hold time constraint = 1.046 ns
Destination Clock signal timing parameters:
Max Edge = 2.020 ns
Max clock path delay = 0.500 ns



# Setup Slack calculation

- Actual data arrival time definition

  Data Input Arrival Timemax + Data path delaymax

  If the data path starts in a primary input,

  Data Input arrivalmax = Input arrival timemax

  If the data path starts at a register,

  (Source Clock Edgemax + Source Clock Path Delaymax) = Data Input Arrivalmax

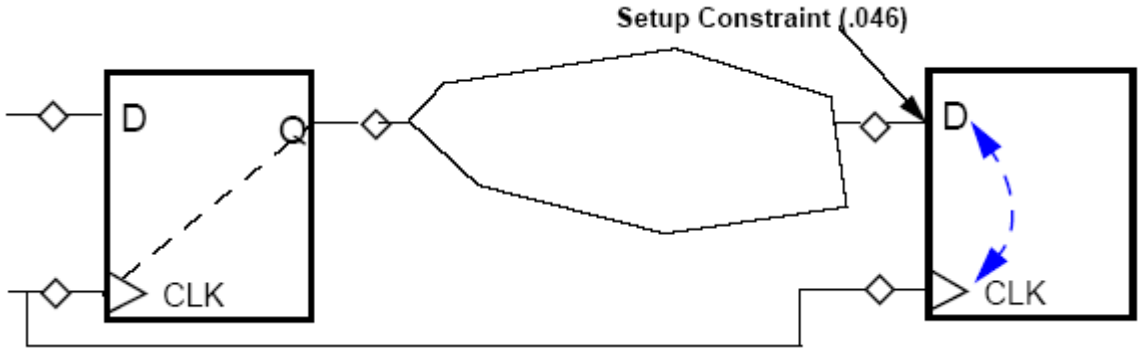
- Required Stability time definition

  (Destination Clock Edgemin + Destination Clock Path Delaymin) - Setup = Required Stability Timemin

- Setup slack definition

  Required Stability Timemin - Actual Data Arrivalmax

# Calculate the setup slack

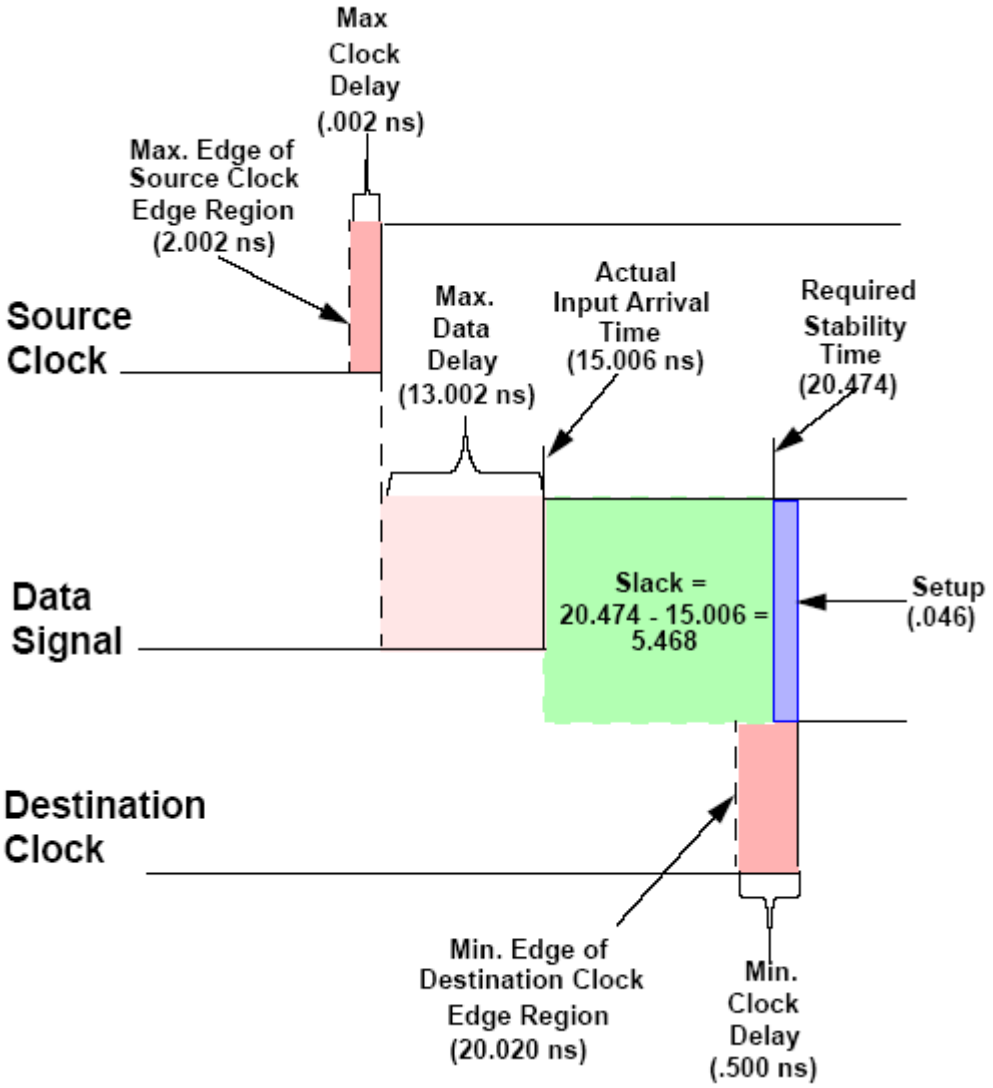



Source Clock signal timing parameters:
Max Edge = 2.002 ns
Max clock path delay = 0.002 ns

Min Data path delay = 13.002 ns
Setup time constraint = 0.046 ns

Destination Clock signal timing parameters:
Min Edge = 20.02 ns
Min clock path delay = 0.500 ns

# Setup slack calculation



# Recovery and Removal time

- Recovery time

  Like setup time for asynchronous port (set, reset)
- Removal time

  Like hold time for asynchronous port (set, reset)

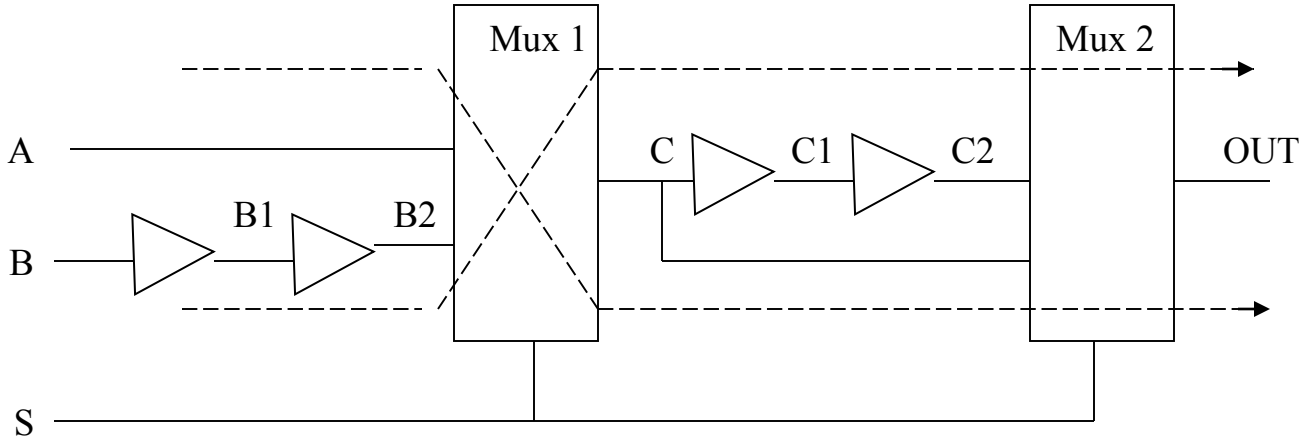
- Recovery time

  It is the time available between the asynchronous signal going inactive to the active clock edge
- Removal time

  It is the time between active clock edge and asynchronous signal going inactive

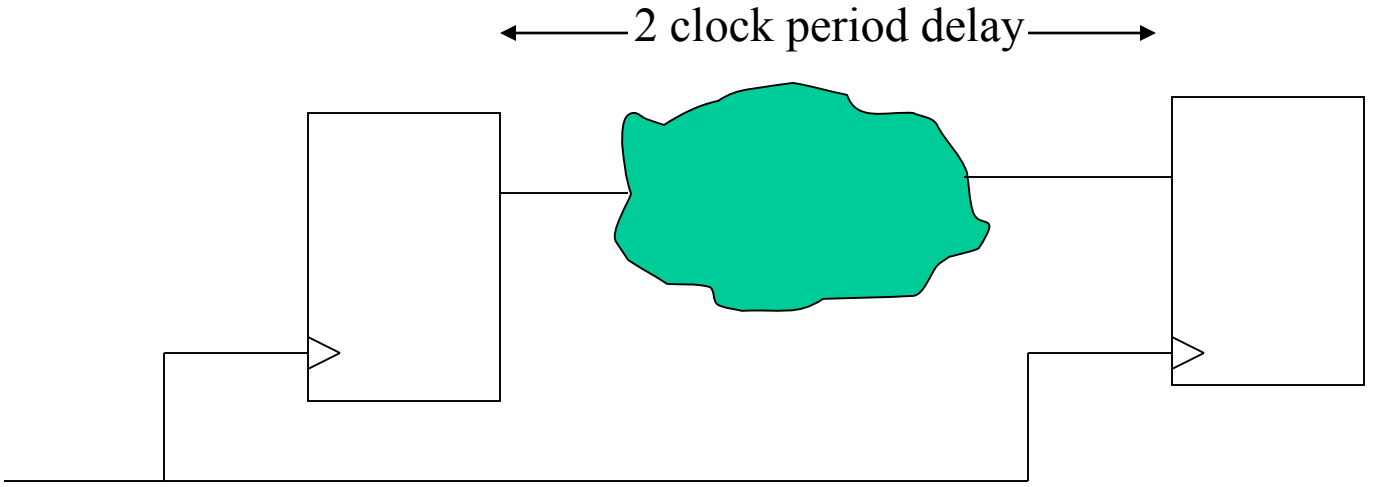
# False Paths

- False paths
  - Paths that physically exist in a design but are not logic/functional paths
  - These paths never get sensitized under any input conditions



# Multi-cycle paths

- Multi-cycle paths
  - Data Paths that require more than one clock period for execution



# Sequential Circuit Timing

**Objectives**

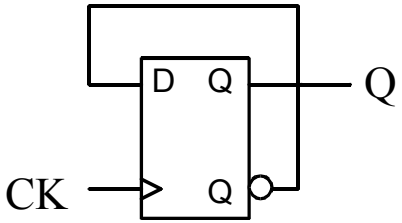
This section covers several timing considerations encountered in the design of synchronous sequential circuits. It has the following objectives:

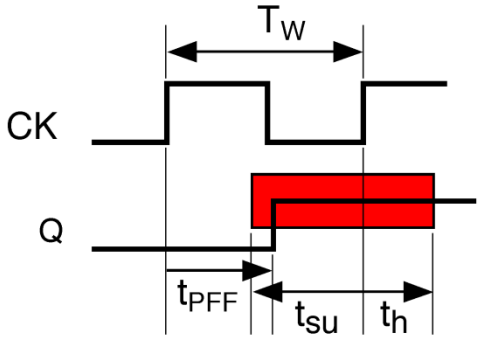
- Define the following global timing parameters and show how they can be derived from the basic timing parameters of flip-flops and gates.
  - Maximum Clock Frequency
  - Maximum allowable clock skew
  - Global Setup and Hold Times
- Discuss ways to control the loading of data into registers and show why gating the clock signal to do this is a poor design practice.

# Maximum Clock Frequency

- The clock frequency for a synchronous sequential circuit is limited by the timing parameters of its flip-flops and gates. This limit is called *the maximum clock frequency* for the circuit. The *minimum clock period* is the reciprocal of this frequency.
- Relevant timing parameters
  - Gates:
    - Propagation delays: min $t_{PLH}$, min $t_{PHL}$, max $t_{PLH}$, max $t_{PHL}$
  - Flip-Flops:
    - Propagation delays: min $t_{PLH}$, min $t_{PHL}$, max $t_{PLH}$, max $t_{PHL}$
    - Setup time: $t_{su}$
    - Hold time: $t_h$

## ➢ Example



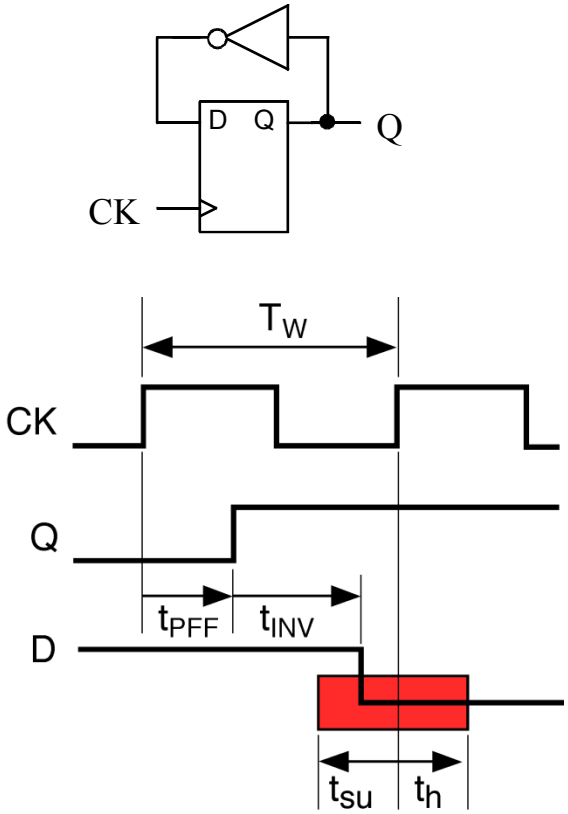

$$T_W \geq \max t_{PFF} + t_{su}$$

For the 7474, max $t_{PLH}$ = 25ns, max $t_{PHL}$ = 40ns, $t_{su}$ = 20ns

$T_W \geq \max (\max t_{PLH} + t_{su,} \max t_{PHL} + t_{su})$

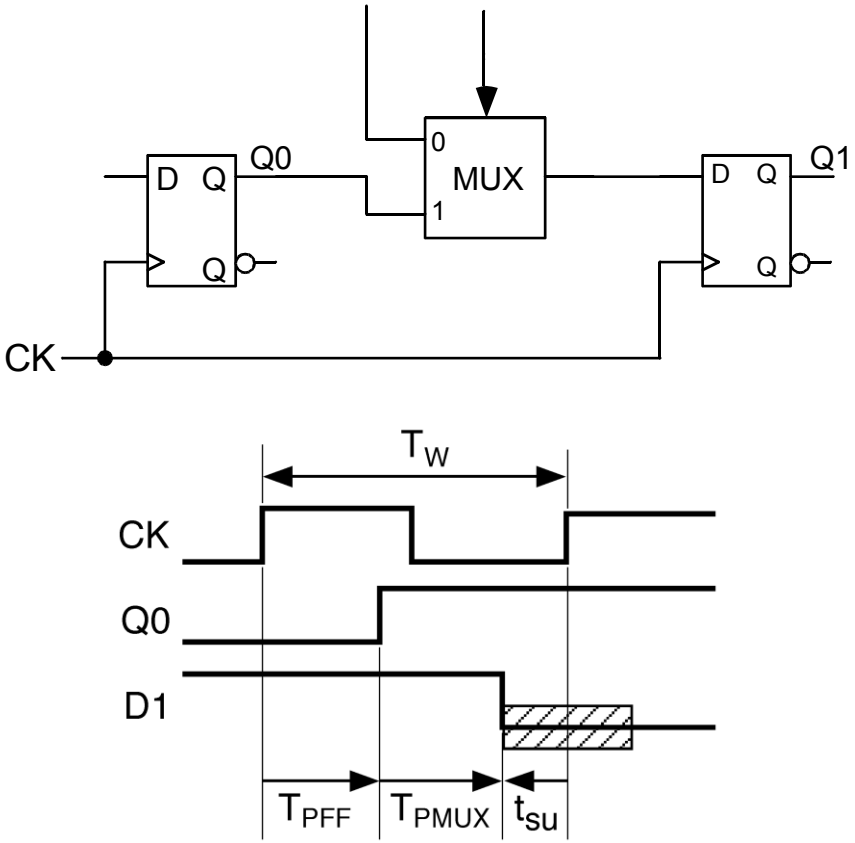
$T_W \geq \max (25+20, 40+20) = 60$

## ➢ Example



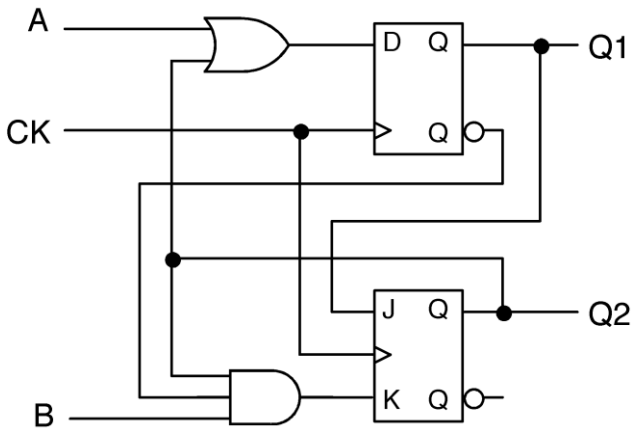

$$T_W \geq \max t_{PFF} + \max t_{PINV} + t_{su}$$

## ➢ Example




$$T_W \geq \max t_{PFF} + \max t_{PMUX} + t_{su}$$

## ➢ Example




| | $t_P$ | $t_{su}$ |
|---|---|---|
| D Flip-Flop: | 20 ns | 5 ns |
| JK Flip-Fllp: | 25 ns | 10 ns |
| AND Gate: | 12 ns | |
| OR Gate: | 10 ns | |

Paths from Q1 to Q1: None

Paths from Q1 to Q2: $T_W \geq \max t_{PDFF} + t_{JKsu} = 20 + 10 = 30$ ns

$T_W \geq \max t_{PDFF} + \max t_{AND} + t_{JKsu} = 20 + 12 + 10 = 42$ ns

Paths from Q2 to Q1: $T_W \geq \max t_{PJKFF} + t_{OR} + T_{Dsu} = 25 + 10 + 5 = 40$ ns

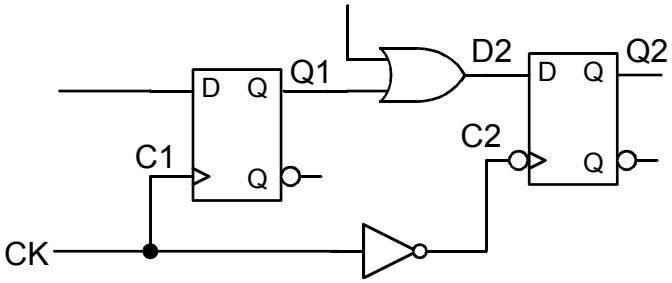
Paths from Q2 to Q2: $T_W \geq \max t_{PJKFF} + \max t_{AND} + t_{JKsu} = 25 + 12 + 10 = 47$ ns

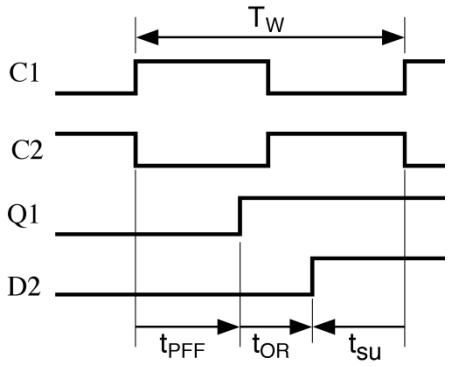
$$TW \geq 47 \text{ ns}$$

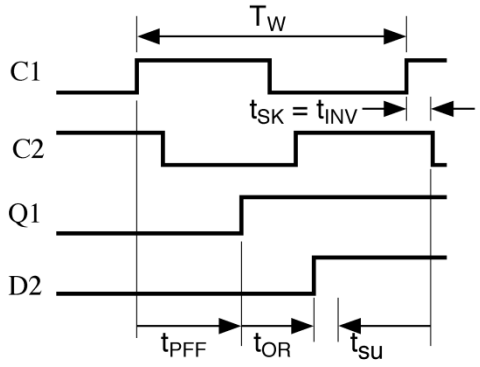
# Clock Skew

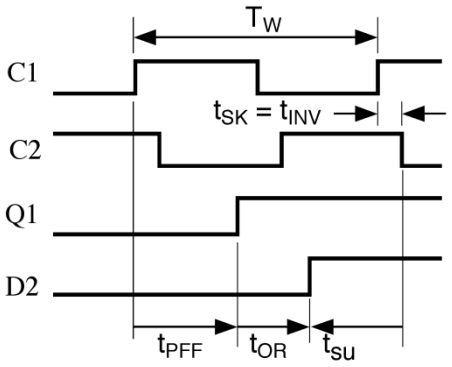
- If a clock edge does not arrive at different flip-flops at exactly the same time, then the clock is said to be *skewed* between these flip-flops. The difference between the times of arrival at the flip-flops is said to be the amount of *clock skew*.
- Clock skew is due to different delays on different paths from the clock generator to the various flip-flops.
  - Different length wires (wires have delay)
  - Gates (buffers) on the paths
  - Flip-Flops that clock on different edges (need to invert clock for some flip-flops)
  - Gating the clock to control loading of registers (a very bad idea)

- Example (Effect of clock skew on clock rate)
  - Clock C2 skewed after C1



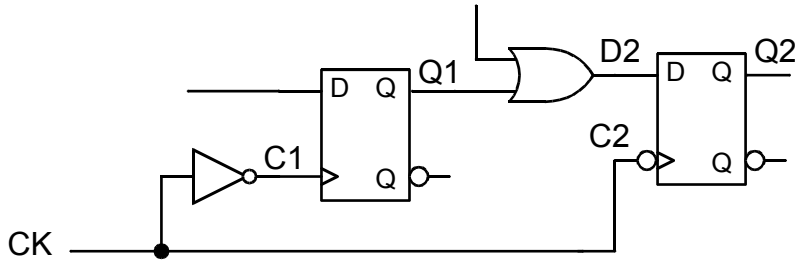



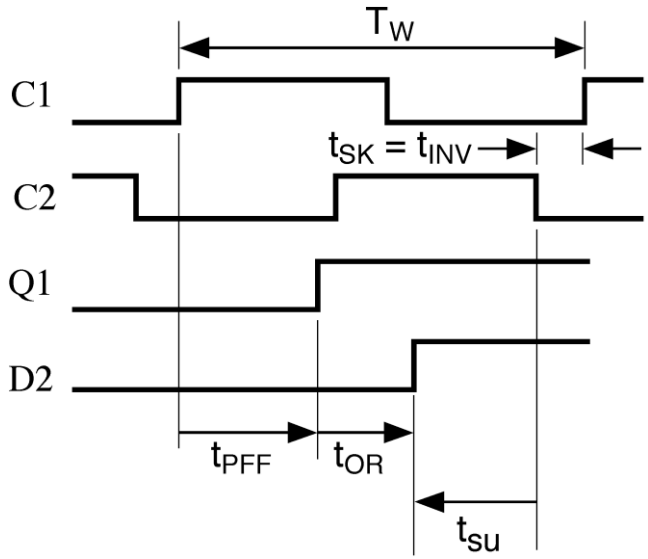




$T_W \geq \max T_{PFF} + \max t_{OR} + t_{su}$
(if clock not skewed, i.e., $t_{INV} = 0$)

$T_W \geq \max T_{PFF} + \max t_{OR} + t_{su} - \min t_{INV}$
(if clock skewed, i.e., $t_{INV} > 0$)

## ➢ Clock C1 skewed after C2

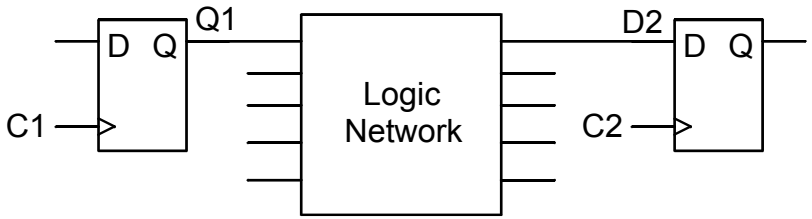


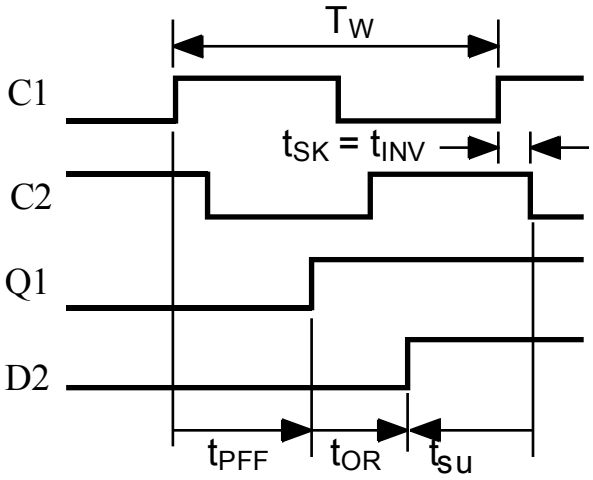



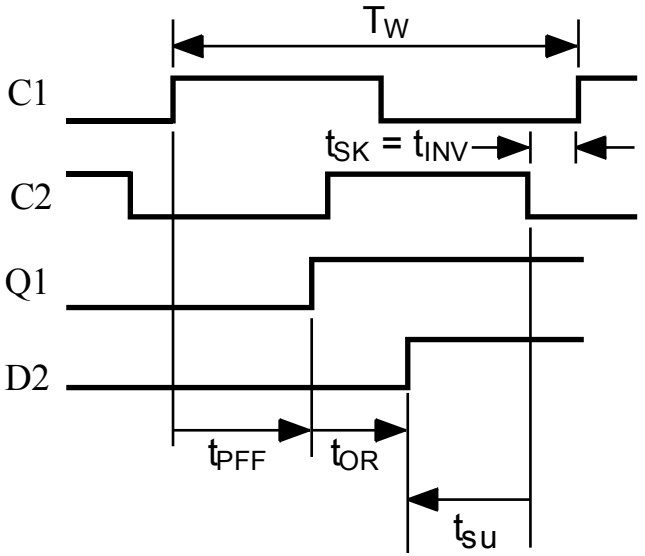

$T_W \geq \max T_{PFF} + \max t_{OR} + t_{su}$
(if clock not skewed, i.e., $t_{INV} = 0$)

$T_W \geq \max T_{PFF} + \max t_{OR} + t_{su} + \max t_{INV}$
(if clock skewed, i.e., $t_{INV} > 0$)

## ➢ Summary of maximum clock frequency calculations



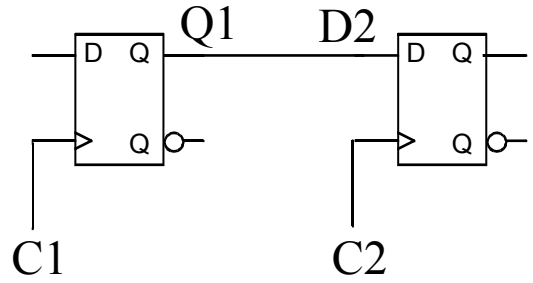




C2 skewed after C1: $T_W \geq \max T_{PFF} + \max t_{NET} + t_{su} - \min t_{INV}$

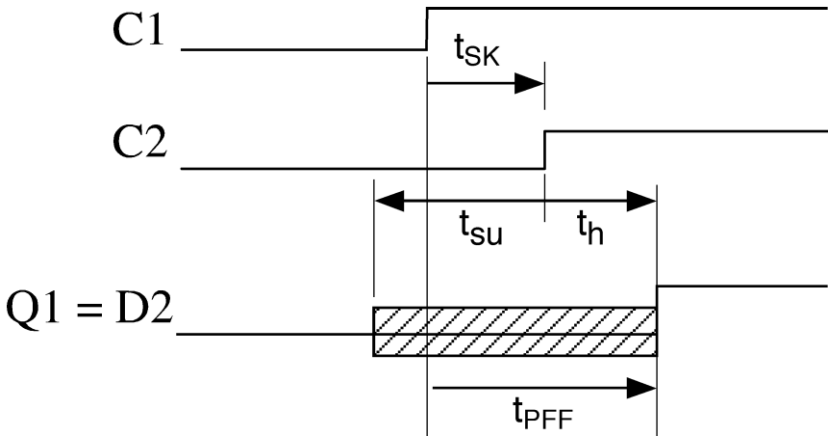
C2 skewed before C1: $T_W \geq \max T_{PFF} + \max t_{NET} + t_{su} + \max t_{INV}$

# Maximum Allowable Clock Skew

- How much skew between C1 and C2 can be tolerated in the following circuit?



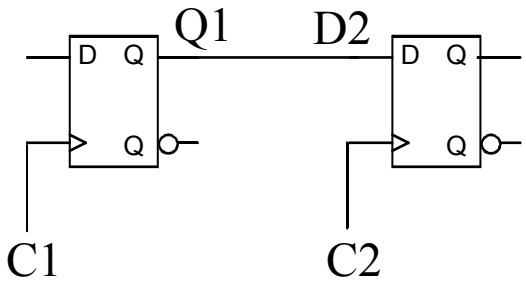

  - Case 1: C2 delayed after C1

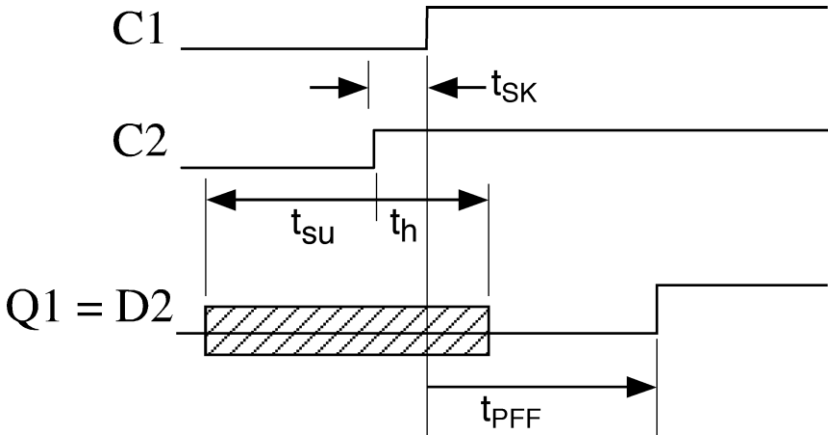



$t_{PFF} > t_h + t_{SK}$

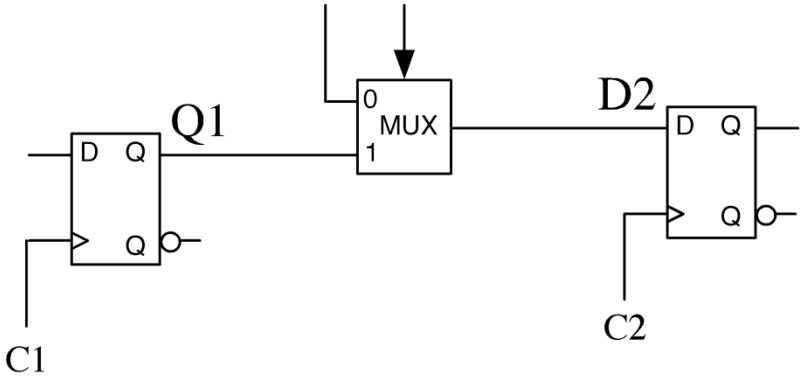
$t_{SK} < \min t_{PFF} - t_h$

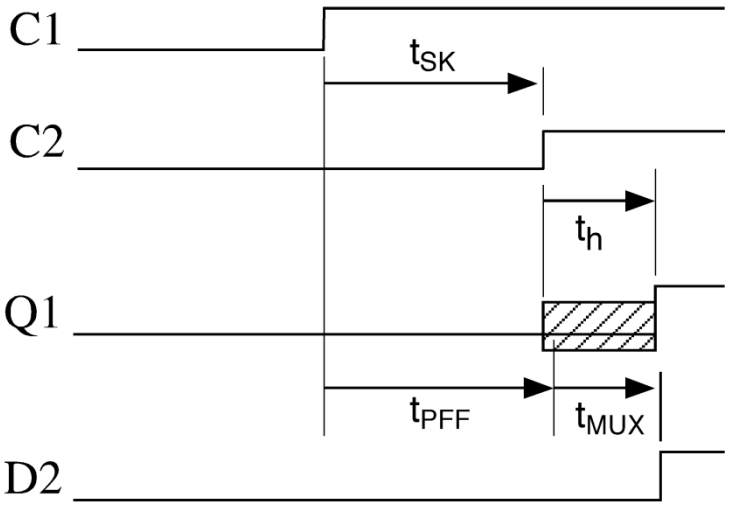
## ➢ Case 2: C1 delayed from C2



- How does additional delay between the flip-flops affect the skew calculations?

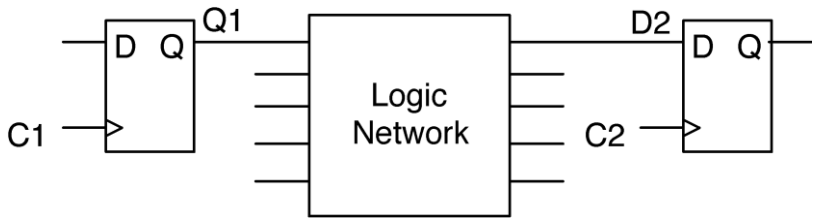


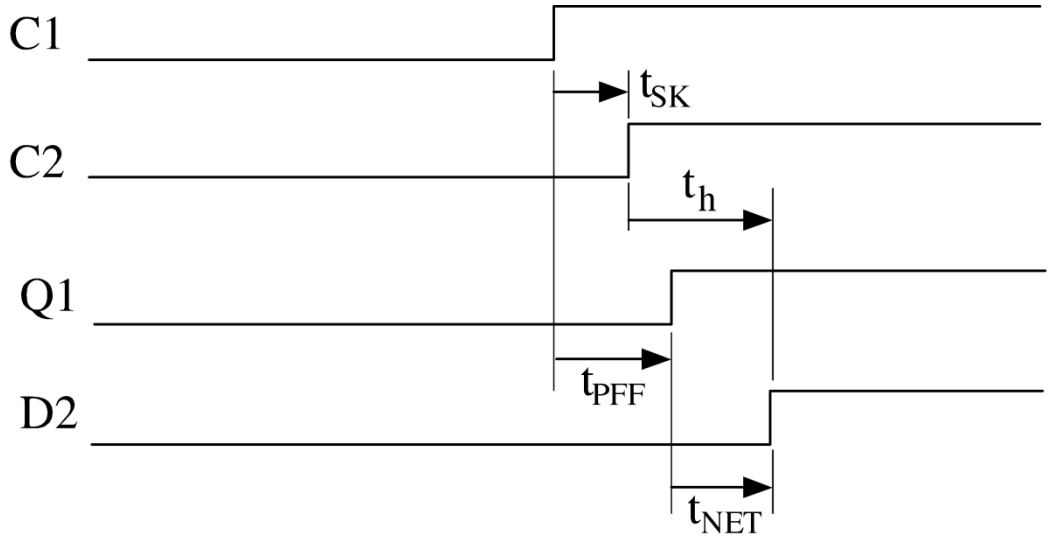




$t_{SK} \leq \min t_{PFF} - t_h$

$t_{sk} \leq \min t_{PFF} + \min t_{MUX} - t_h$

## ➢ Summary of allowable clock skew calculations

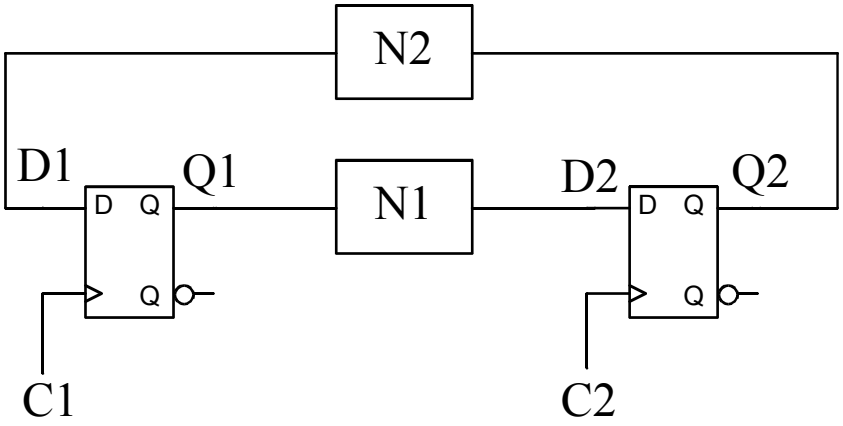


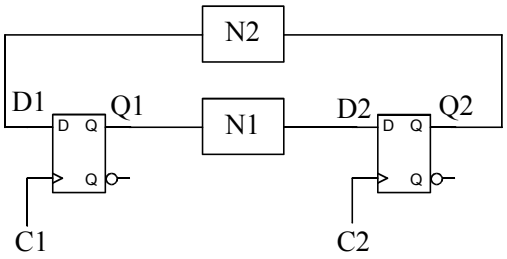




$t_{SK} + t_h \leq t_{PFF} + t_{NET}$

$t_{SK} \leq \min t_{PFF} + \min t_{NET} - t_h$

- Example: What is the minimum clock period for the following circuit under the assumption that the clock C2 is skewed after C1 (i.e., C2 is delayed from C1)?






- First calculate the maximum allowable clock skew.

  $t_{SK} < \min t_{PFF} + \min t_{N1} - t_h$

- Next calculate the minimum clock period due to the path from Q1 to D2.

  $T_W > \max t_{PFF} + \max t_{N1} + t_{su} - \min t_{SK}$

- Finally calculate the minimum clock period due to the path from Q2 to D1

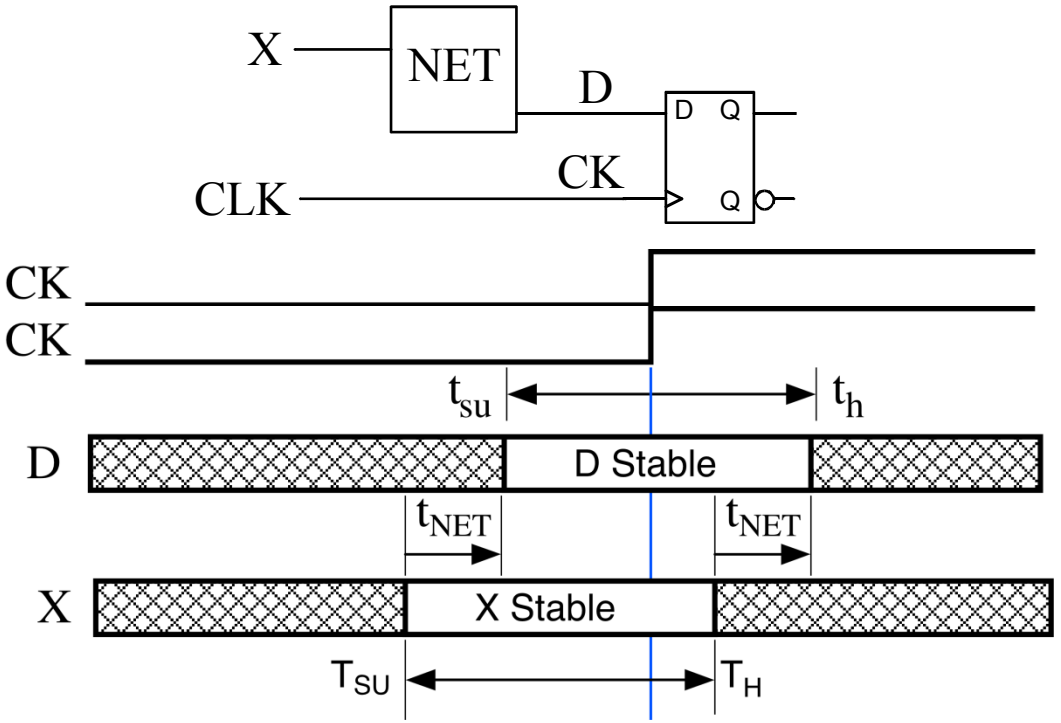
  $T_W > \max t_{PFF} + \max t_{N1} + t_{su} + \max t_{SK}$

  $T_W > \max t_{PFF} + \max t_{N2} + t_{su} + (\min t_{PFF} + \min t_{N1} - t_h)$

  $T_W > \max t_{PFF} + \min t_{PFF} + \max t_{N2} + \min t_{N1} + t_{su} - t_h$

# Global Setup Time, Hold Time and Propagation Delay

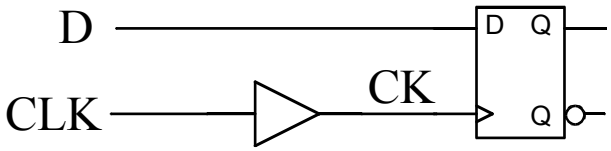
- Global setup and hold times (data delayed)

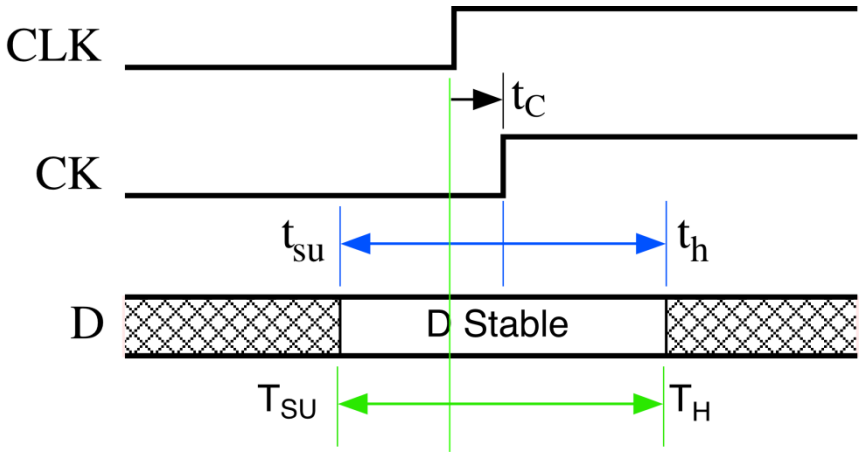



$T_{SU} = t_{su} + \max t_{NET}$

$T_H = t_h - \min t_{NET}$

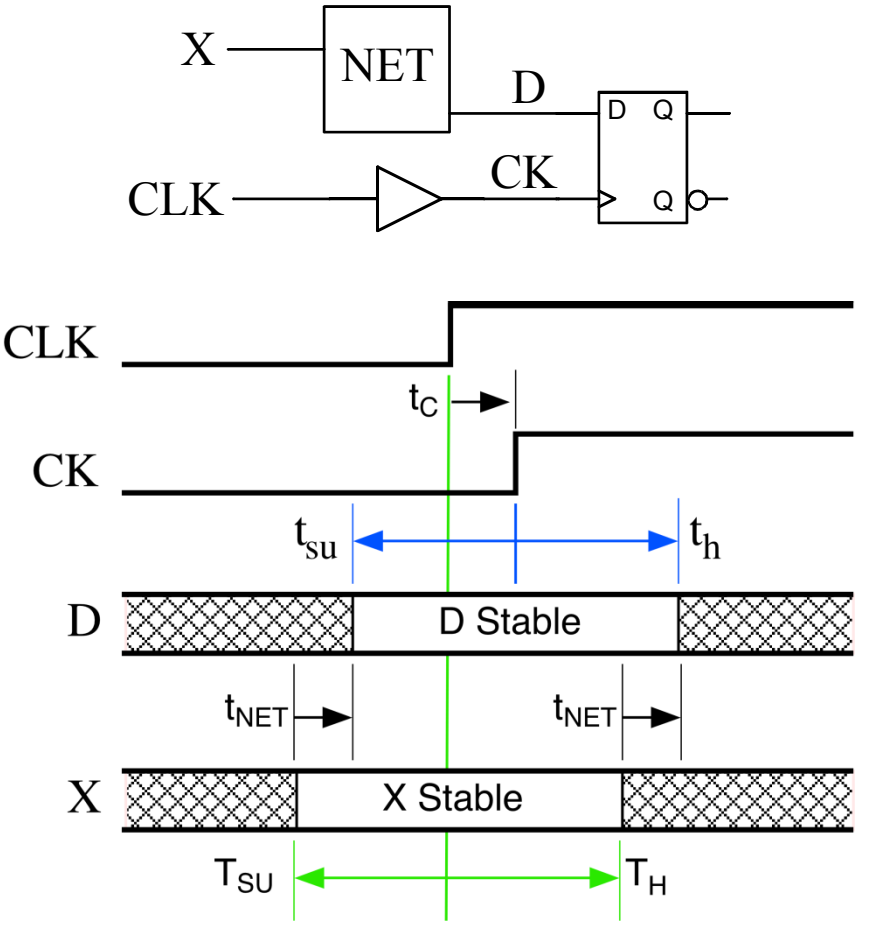
## ➢ Global setup & hold time (clock delayed)



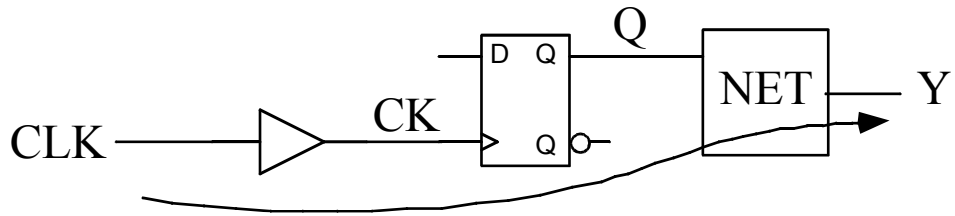

$T_{SU} = t_{su}$ - min $t_C$ $\qquad$ $T_H = t_h$ + max $t_C$

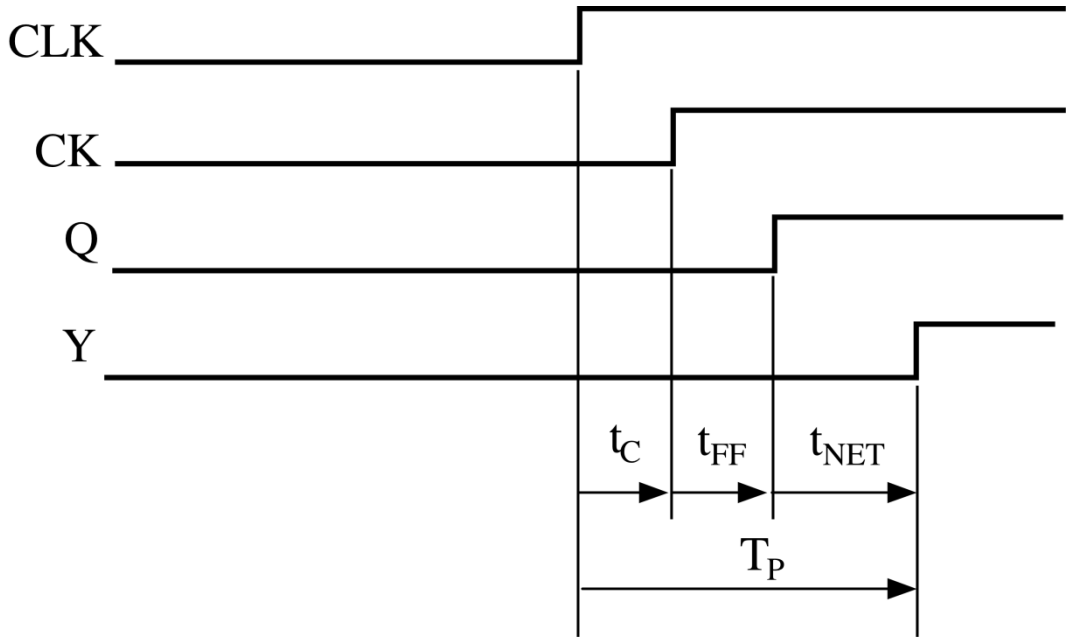
- Global setup & hold time (data & clock delayed)




$T_{SU}$ = + max =-0987654321\ - min . $T_H = t_h$ - min $t_{NET}$ + max $t_C$

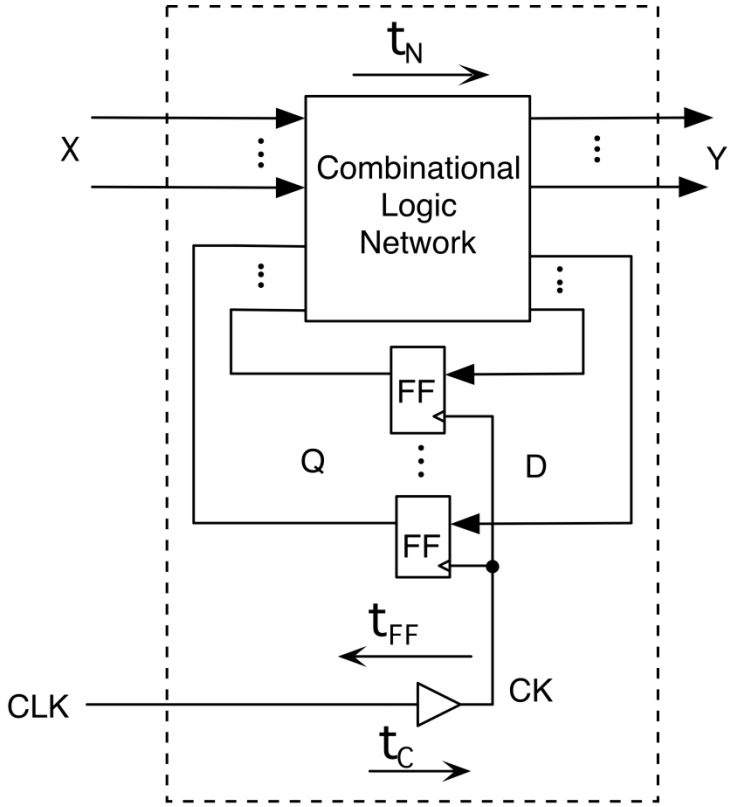
## ➢ Global propagation delay

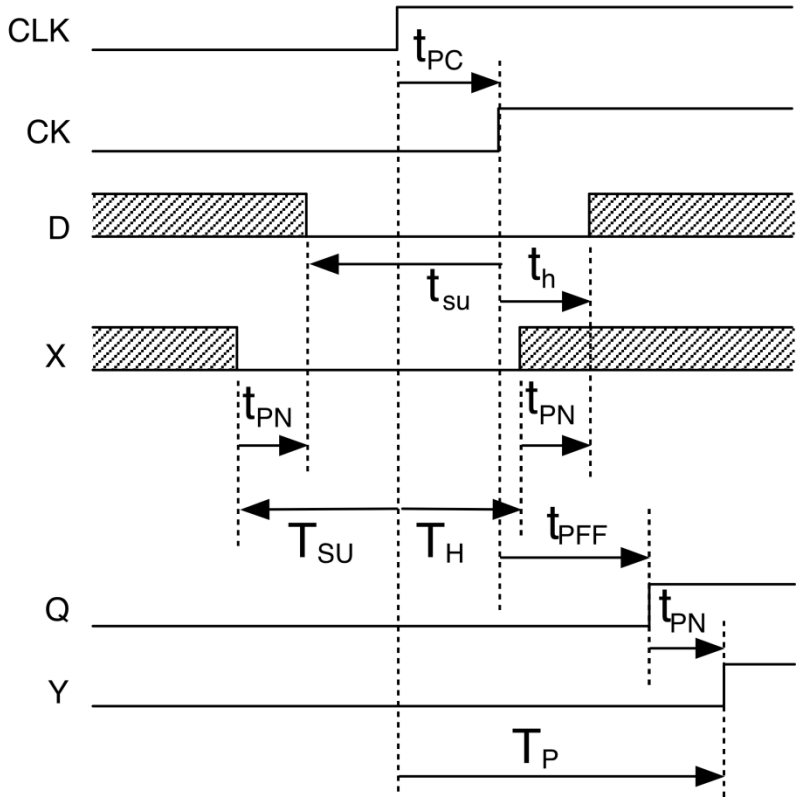



$$T_P = t_C + t_{FF} + t_{NET}$$

## ➢ Summary of global timing parameters



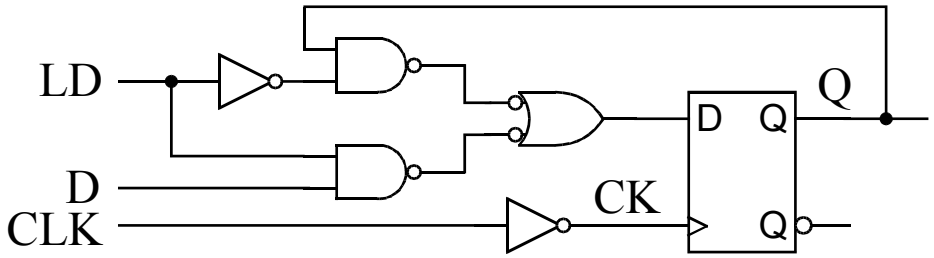




$T_{SU} = t_{su} + \max t_{PN} - \min t_{PC} \leq t_{su} + \max t_{PN}$

$T_H = t_h + \max t_{PC} - \min t_{PN} \leq t_h + \max t_{PC}$

$T_P = t_{PFF} + t_{PN} + t_{PC}$

- Example




  - Find $T_{SU}$ and $T_H$ for input signal LD relative to CLK.

$$T_{SU} = t_{su} + \max t_{NET} - \min t_C$$

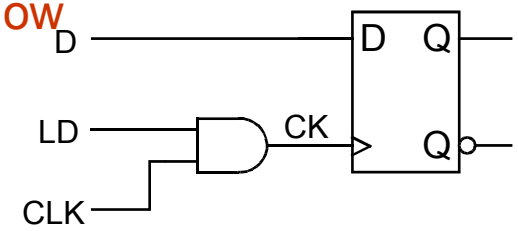
$$= t_{su} + \max t_{INV} + \max t_{NAND} + \max t_{NAND} - \min t_{INV}$$

$$T_H = t_h - \min t_{NET} + \max t_C$$

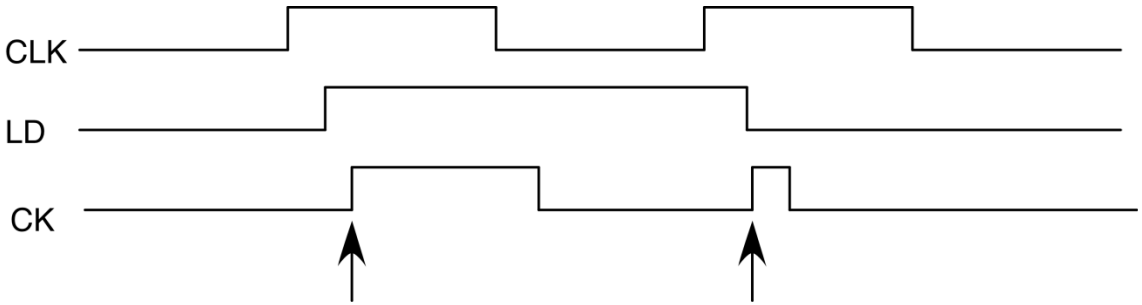
$$= t_h - \min t_{NAND} - \min t_{NAND} + \max T_{INV}$$

# Register load control (gating the clock

- A very bad way to add a load control signal LD to a register that does not have one is shown below.

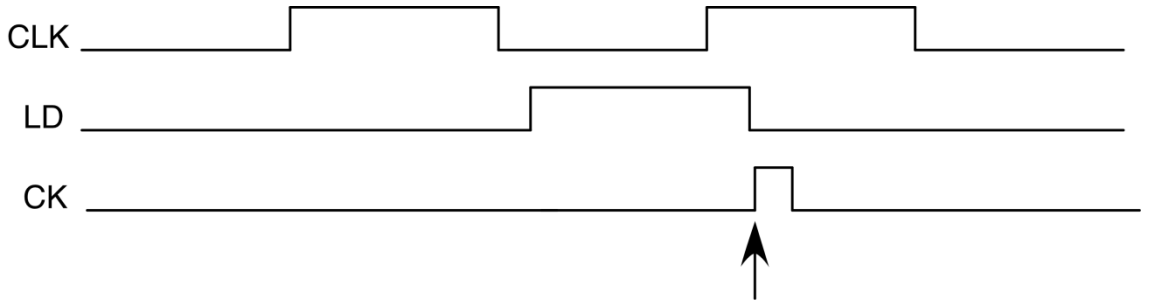



- The reason this is such a bad idea is illustrated by the following timing diagram.

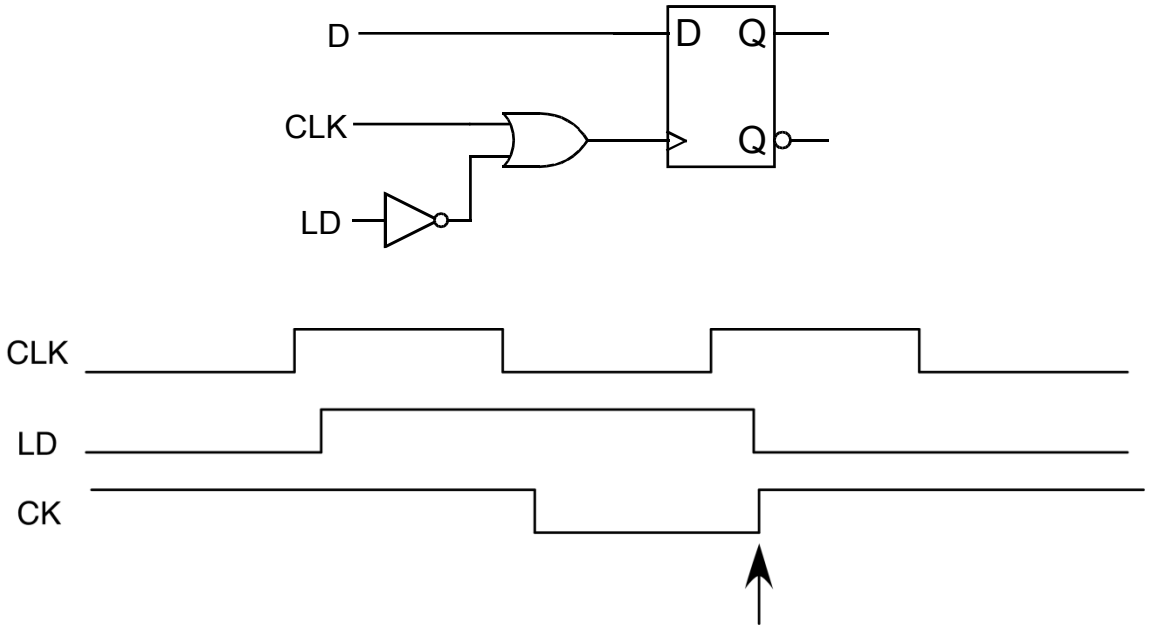



- The flip-flop sees two rising edges and will trigger twice. The only one we want is the second one.

- If LD was constrained to only change when the clock was low, then the only problem would be the clock skew.

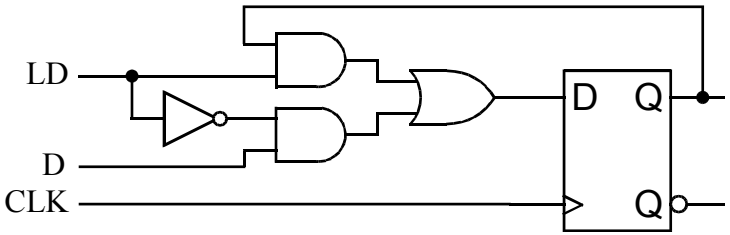


- If gating the clock is the only way to control the loading of registers, then use the following approach:




- There is still clock skew, but at least we only have one triggering edge.

- The best way to add a LD control signal is as follows:



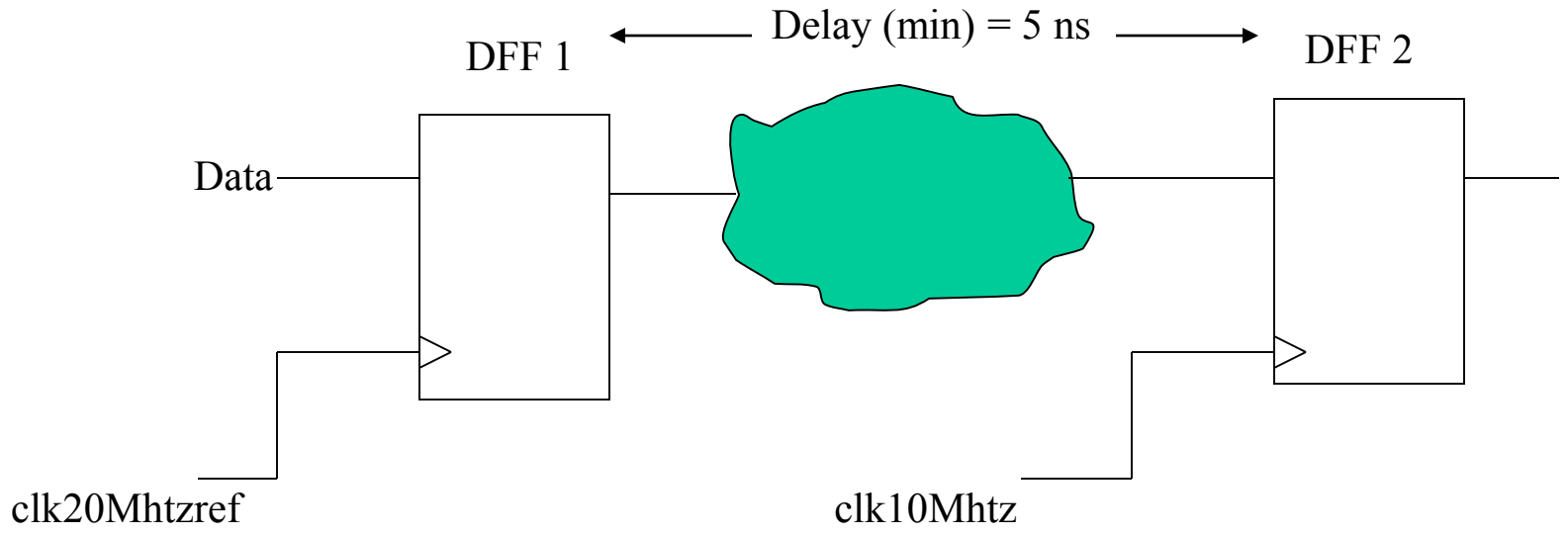
# Tips & Tricks

- Use timing diagrams to determine the timing properties of sequential circuits
- Using typical timing values from the data sheet (use only max and/or min values)
- Gating the clock

# Detecting timing violations – CASE 1

(a) Hold time for clocks is 1.5 ns

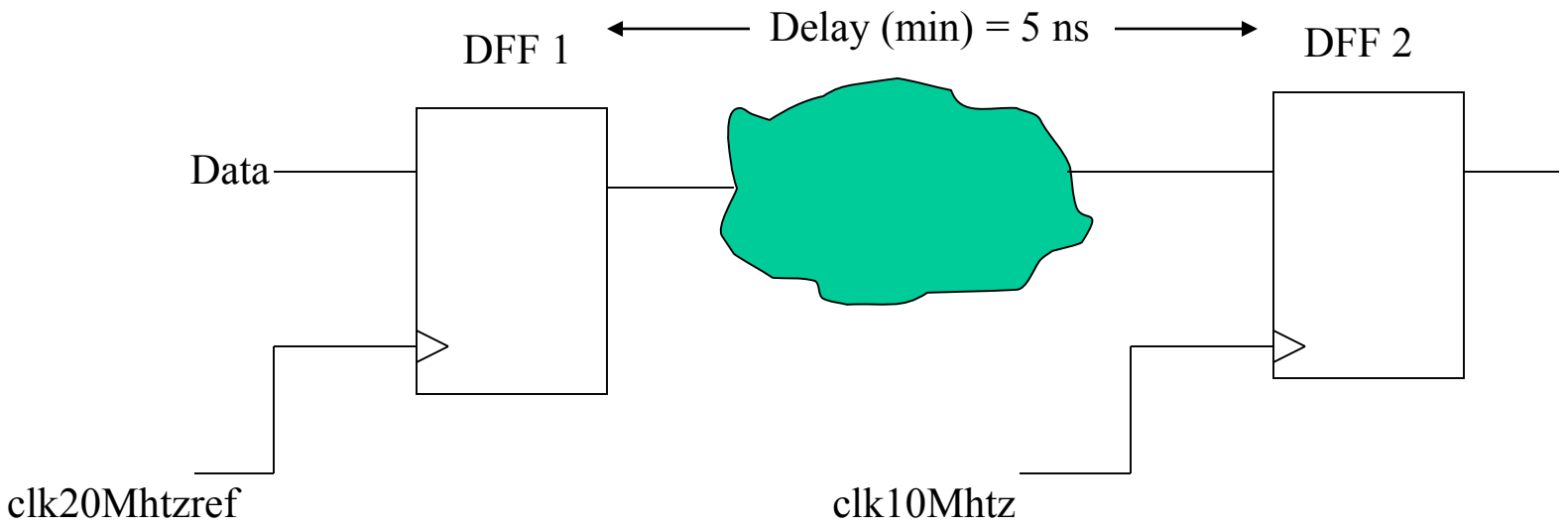
Determine if there are any timing violations in this design



# Detecting timing violations – CASE 2

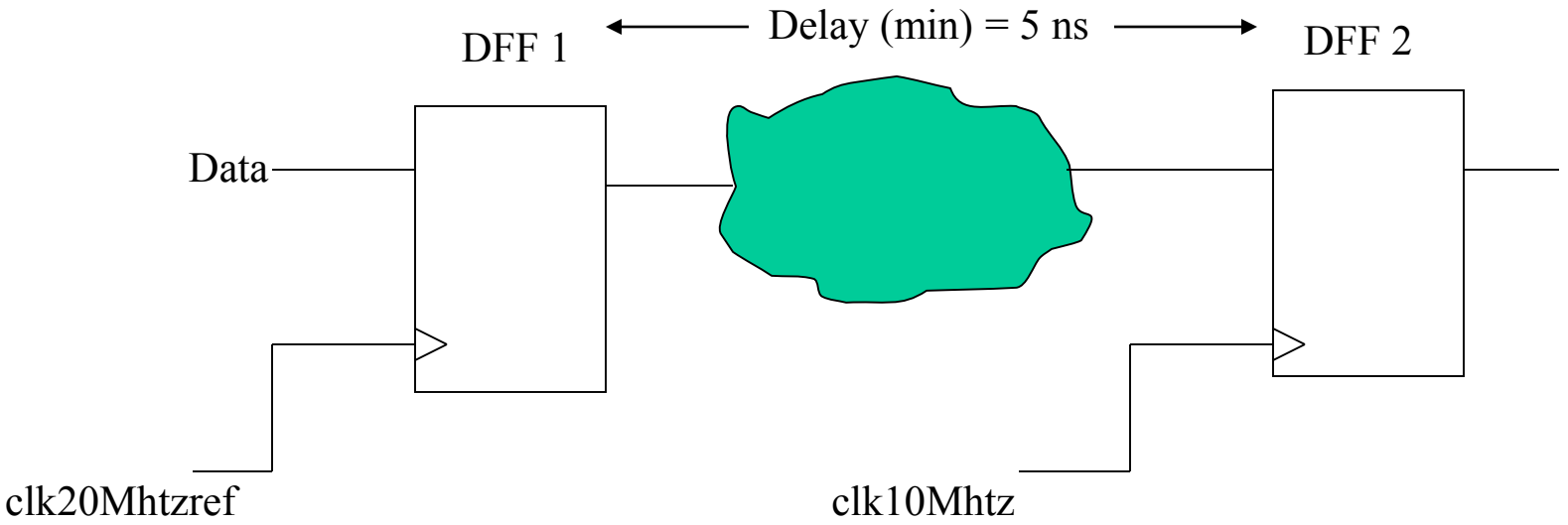
(a) Hold time for clocks is 1.5 ns
(b) Clock skew of 3.72 ns between clk20mref and clk10mz

Determine if there are any timing violations in this design



# Detecting timing violations – CASE 3

(a) Hold time for clocks is 1.5 ns
(b) Clock skew of 3.72 ns between clk20mref and clk10mz

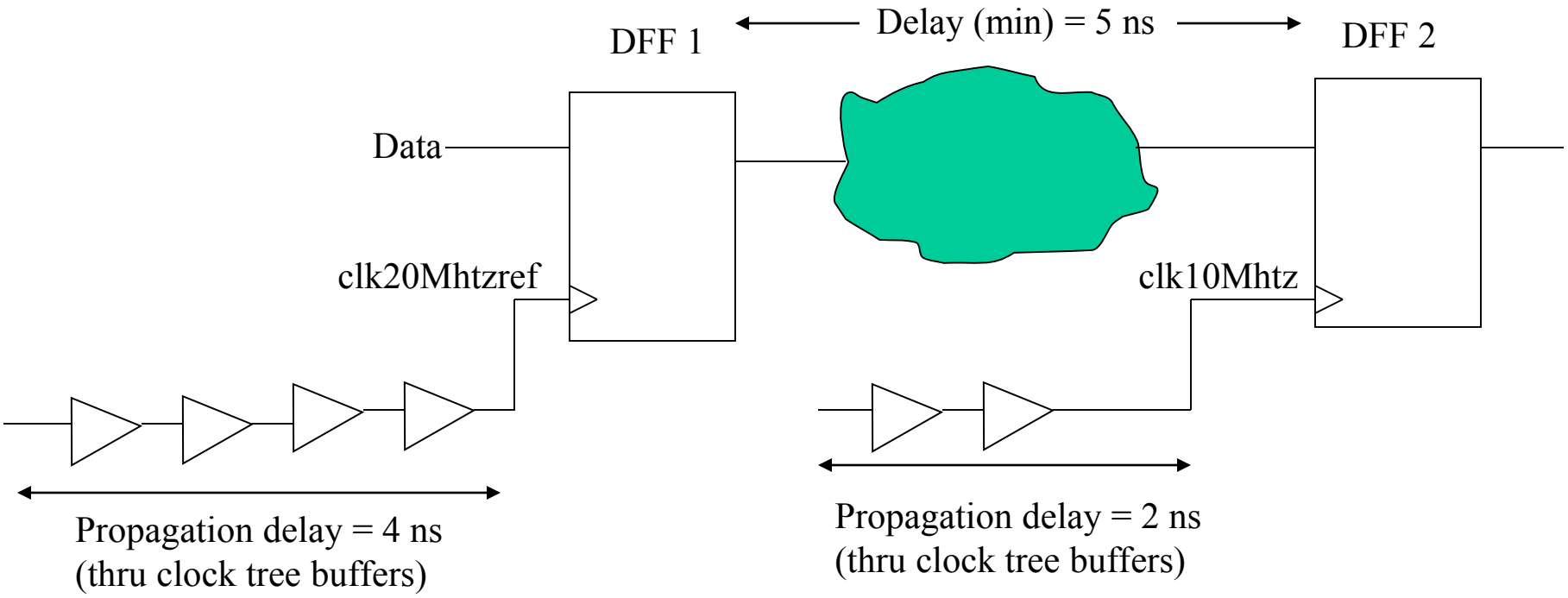


# Detecting timing violations – CASE 4

Consider

(a) Clock skew of 3.72 ns between clk20mref and clk10mz

(b) Clock network delays



# Thank you