Digital System Design

Hajar Falahati

hfalahati@ipm.ir
hfalahati@ce.sharif.edu

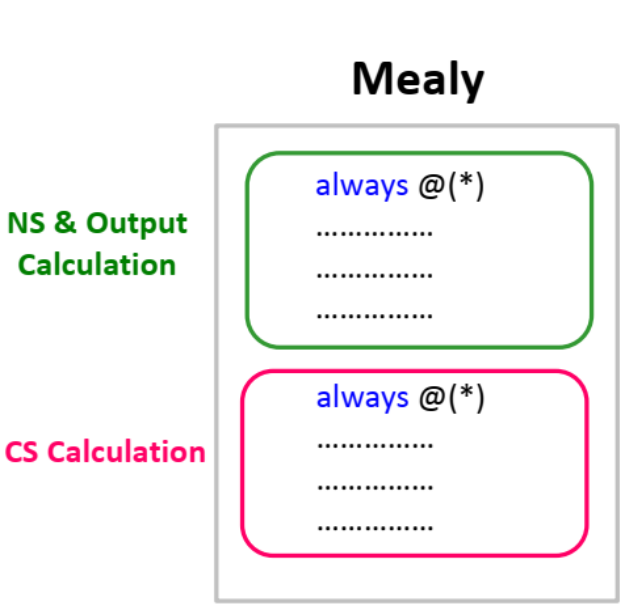
Combinational Logic & Verilog

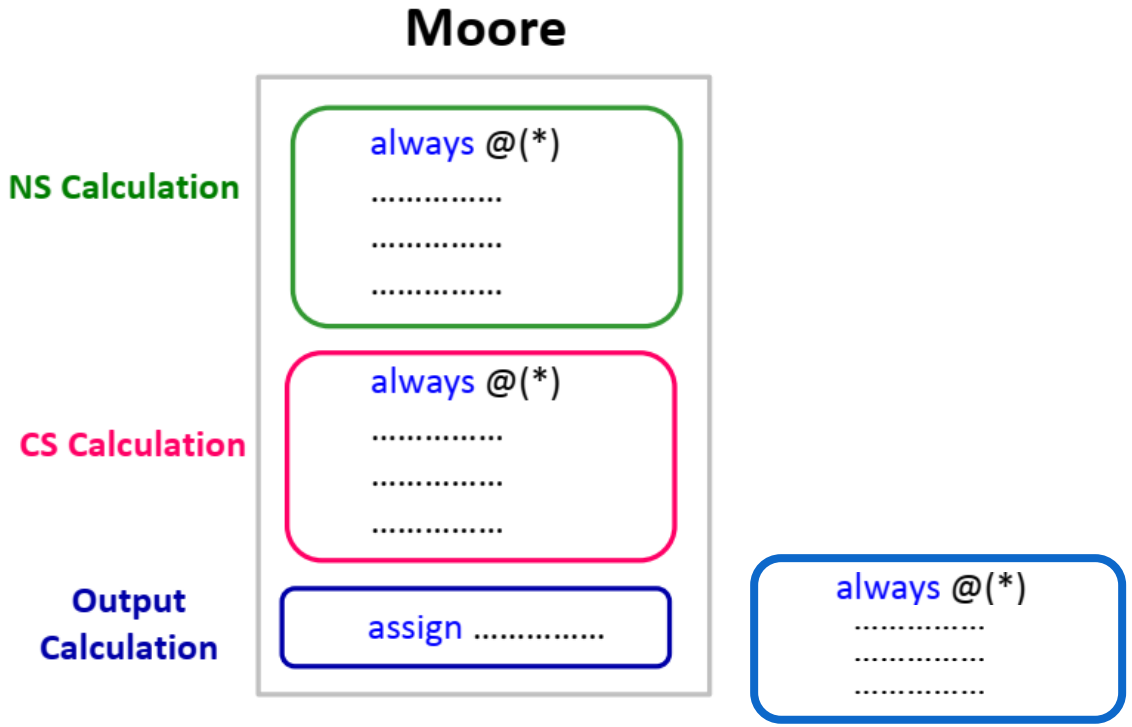
- Combinational Logic
  - Use **always block** + **“blocking” assignments**
    - Normally for high-complexity Comb. Logic
    - When output depends on several conditions, which requires if-else

- Sequential Logic
  - Can **only** be realized using an **always block**
  - When using the **always block** for the sequential Logic, **“Non-blocking” assignments** are used

FSM Code Structure

- Mealy
  - Output depends on input
  - Output declared as reg
- Moore
  - Output does not depends on input
  - Output declared as wire





Function vs. Task

Functions	**Tasks**
A function can enable another function but not another task.	A task can enable other tasks and functions.
Functions always execute in 0 simulation time.	Tasks may execute in non-zero simulation time.
Functions must not contain any delay, event, or timing control statements.	Tasks may contain delay, event, or timing control statements.
Functions must have at least one input argument. They can have more than one input.	Tasks may have zero or more arguments of type input, output, or inout.
unctions always return a single value. They cannot have output or inout arguments.	Tasks do not return with a value, but can pass multiple values through output and inout arguments.

Outline

- Timing

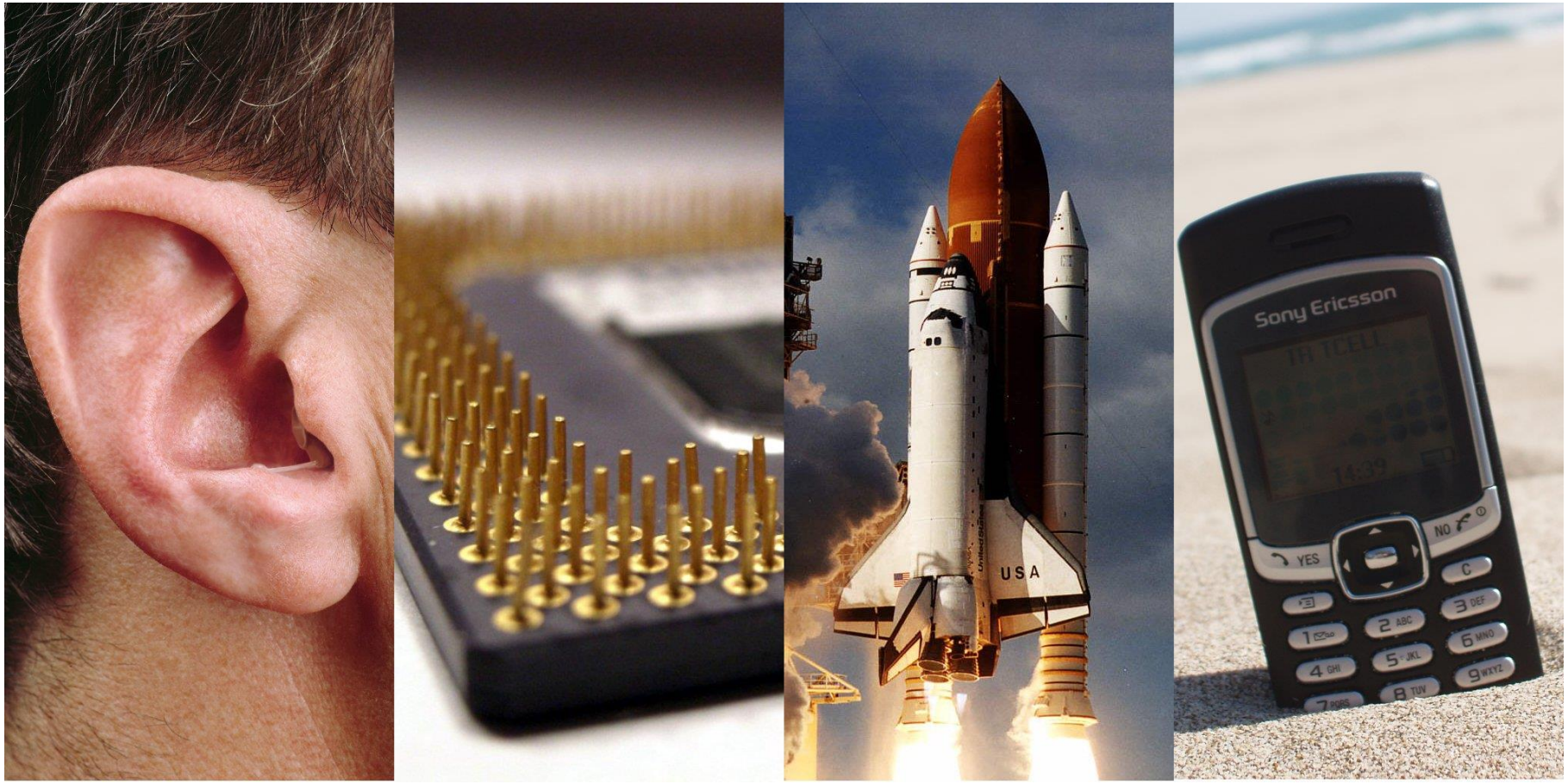
- Testbench

Tradeoffs in Circuit Design

Circuit Design is a Tradeoff Between:

- Area
  - Circuit **area** is proportional to the **cost** of the device
- Speed / Throughput
  - We want **faster**, more **capable** circuits
- Power / Energy
  - Mobile devices need to work with a **limited** power supply
  - High performance devices **dissipate** more than 100 W/cm^2
- Design Time
  - Designers are **expensive** in *time* and *money*
  - The **competition** will not wait for you

Requirements and Goals Depend On Application



Circuit Timing

- Until now, we investigated **logical functionality**

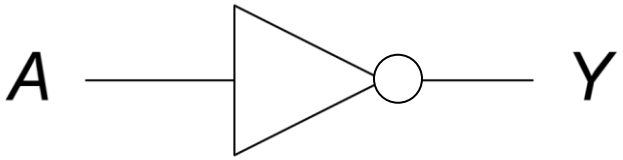
- What about **timing**?
  - How **fast** is a circuit?
  - How can we make a circuit **faster**?
  - What happens if we run a circuit **too fast**?

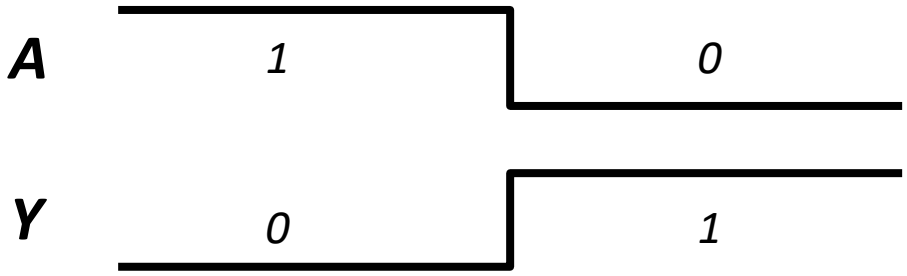
- A design that is logically correct can still **fail** because of real-world **implementation issues**!

Combinational Circuit Timing

Digital Logic Abstraction

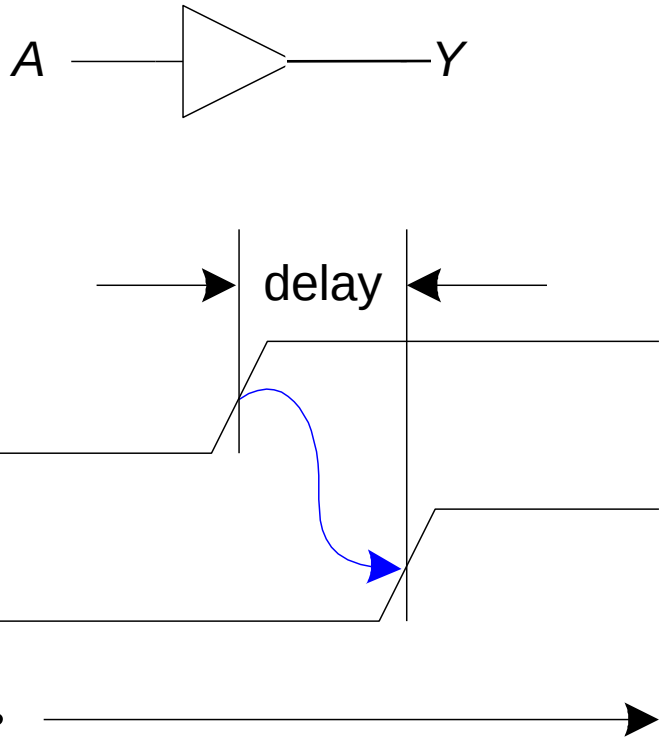
- **"Digital logic"** is a convenient **abstraction**
  - Output changes ***immediately*** with the input





Combinational Circuit Delay

- In reality, **outputs** are **delayed** from **inputs**
  - Transistors take a finite amount of time to switch



Real Inverter Delay Example

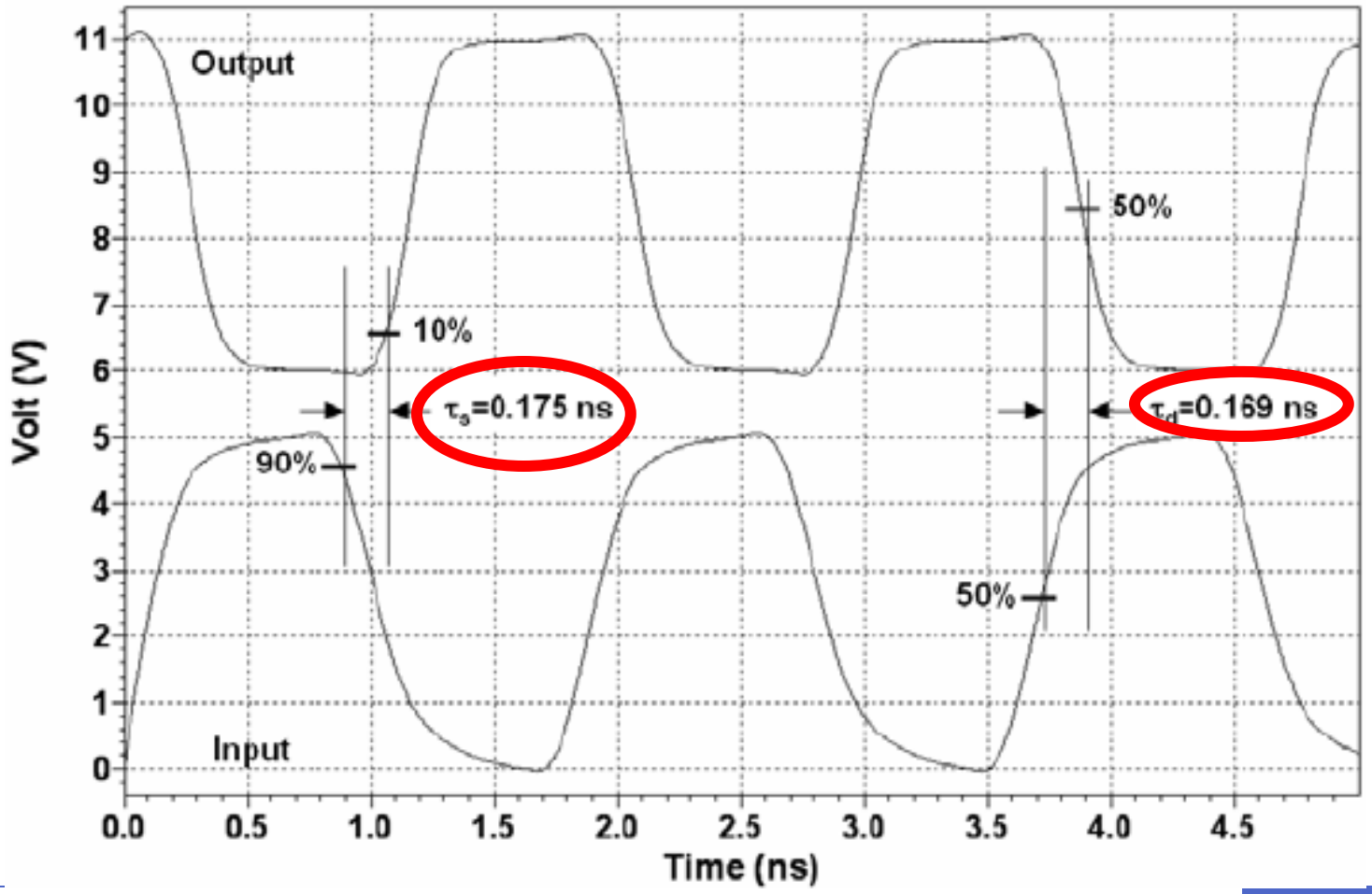


Image source: Sandoval-Ibarra, F., and E. S. Hernández-Bernal. "Ring CMOS NOT-based oscillators: Analysis and design." *Journal of applied research and technology,* 2008.

Circuit Delay and Its Variation

- Delay is fundamentally caused by
  - **Capacitance** and **resistance** in a circuit

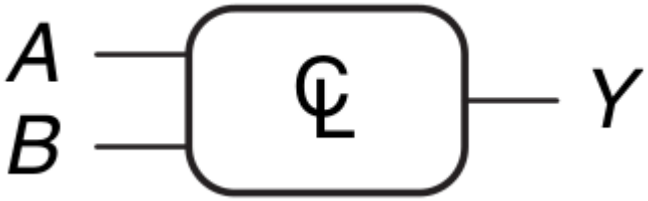
- ***Anything*** affecting these quantities can change delay:
  - **Rising** (i.e., 0 -> 1) vs. **falling** (i.e., 1 -> 0) inputs
  - Different **inputs** have different **delays**
  - Changes in **environment** (e.g., temperature)
  - **Aging** of the circuit

- We have a **range of possible delays** from input to output

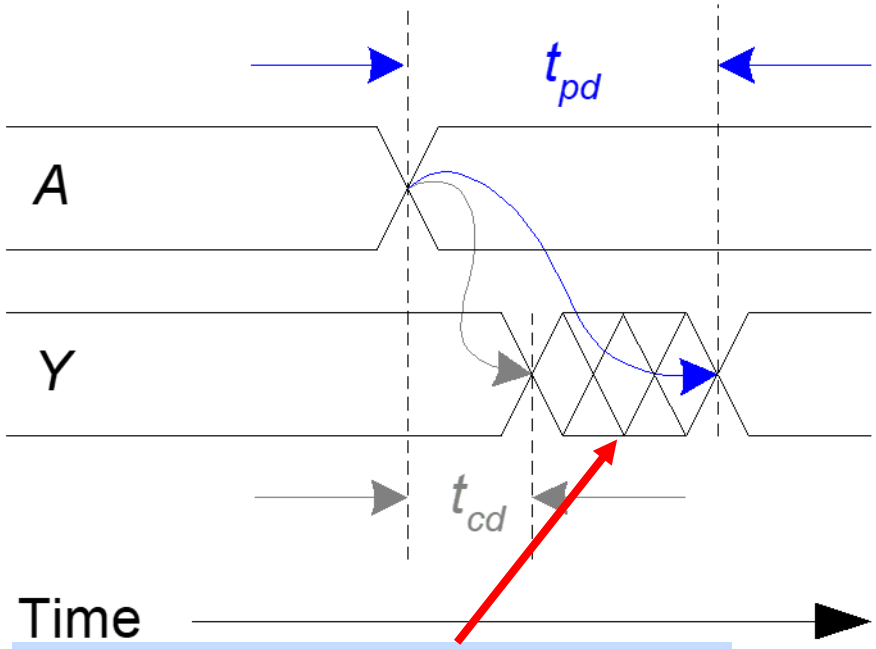
Delays from Input to Output

- **Contamination delay (t_{cd}):** delay until Y ***starts changing***
- **Propagation delay (t_{pd}):** delay until Y ***finishes changing***

Example Circuit



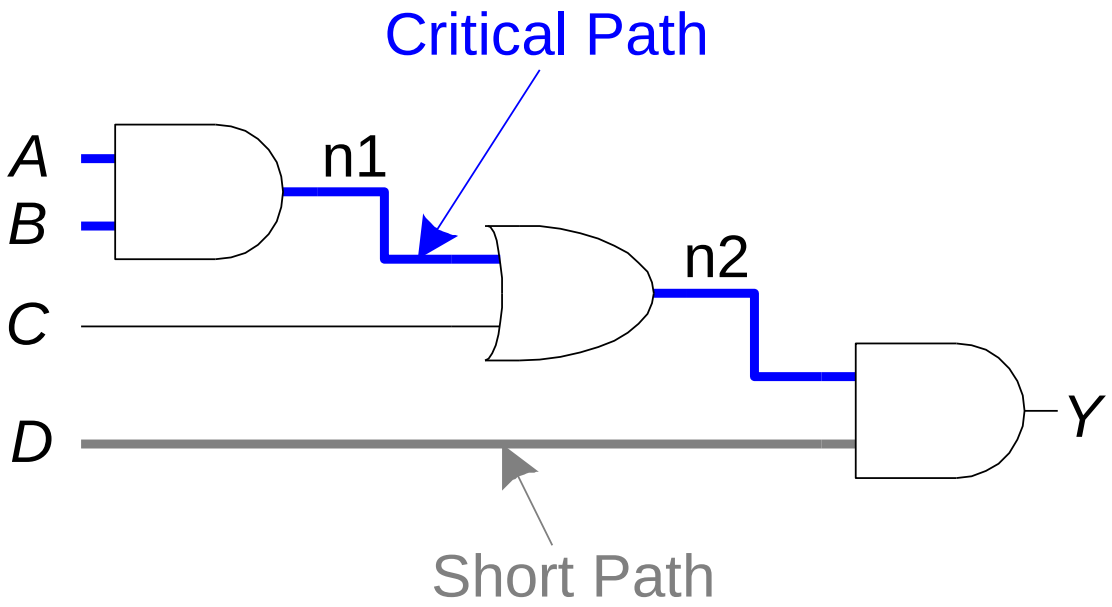
Effect of Changing Input 'A'



Cross-hatching means value is changing

Calculating Long/Short Paths

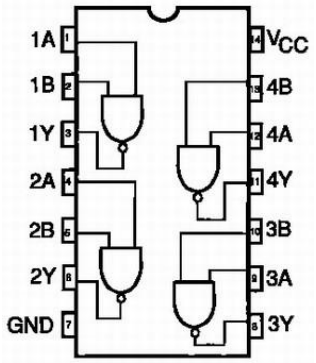
- We care about **both** the ***longest*** and ***shortest*** paths in a circuit



Critical (Longest) Path: $t_{pd} = 2\,t_{pd\_AND} + t_{pd\_OR}$

Shortest Path: $t_{cd} = t_{cd\_AND}$

Example t_{pd} for a Real NAND-2 Gate



Symbol	Parameter	Conditions	25 °C			–40 °C to +125 °C		Unit
			Min	Typ	Max	Max (85 °C)	Max (125 °C)	
74HC00								
t_{pd}	propagation delay	nA, nB to nY; see Figure 6 [1]						
		V_{CC} = 2.0 V	-	25	-	115	135	ns
		V_{CC} = 4.5 V	-	9	-	23	27	ns
		V_{CC} = 5.0 V; C_L = 15 pF	-	7		-	-	ns
		V_{CC} = 6.0 V	-	7	-	20	23	ns

Heavy **dependence** on **voltage** and **temperature**!

Source: Nexperia 2-input NAND (74HC00) Datasheet, Section 10

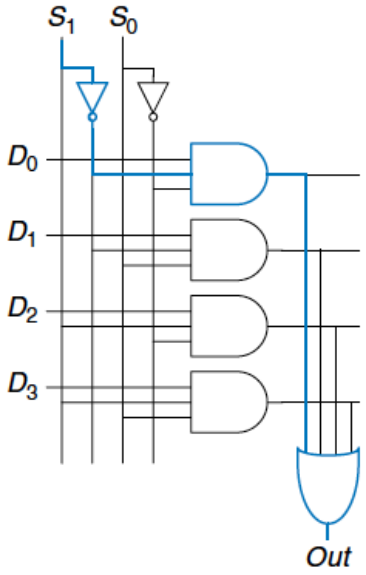
Example Worst-Case t_{pd}

- Two different **implementations** of a **4:1 multiplexer**

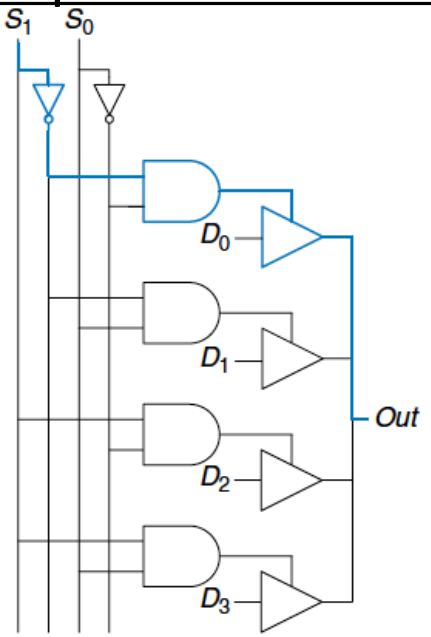
Gate Delays

Gate	t_{pd} (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
tristate (A to Y)	50
tristate (enable to Y)	35

Implementation 1



Implementation 2



Different designs lead to very **different delays**

Example Worst-Case t_{pd}

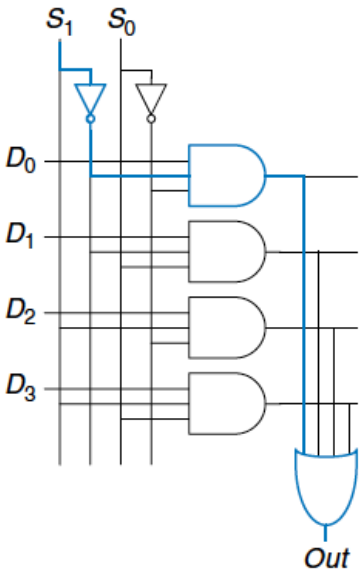
- Two different **implementations** of a **4:1 multiplexer**

Gate Delays

Gate	t_{pd} (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
tristate (A to Y)	50
tristate (enable to Y)	35

Different designs lead to very **different delays**

Implementation 1

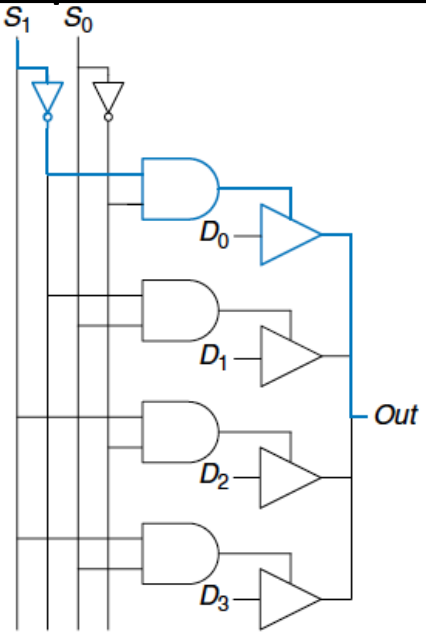


(a)

$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND3} + t_{pd\_OR4}$
$= 30\text{ ps} + 80\text{ ps} + 90\text{ ps}$
$= 200\text{ ps}$

$t_{pd\_dy} = t_{pd\_AND3} + t_{pd\_OR4}$
$= 170\text{ ps}$

Implementation 2



(b)

$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND2} + t_{pd\_TRI\_SY}$
$= 30\text{ ps} + 60\text{ ps} + 35\text{ ps}$
$= 125\text{ ps}$

$t_{pd\_dy} = t_{pd\_TRI\_AY}$
$= 50\text{ ps}$

Calculating Long/Short Paths

- It's **not** always this easy to determine the long/short paths!
  - Not all **input transitions** affect the **output**
  - Can have **multiple different paths** from an input to output
- In reality, circuits are **not** all built equally
  - Different instances of the **same gate** have **different delays**
  - **Wires** have **nonzero delay** (increasing with length)
  - Temperature/voltage affect circuit speeds
    - Not all circuit elements are affected the same way
    - Can even **change the critical path**!
- Designers assume **"worst-case" conditions** and run many **statistical simulations** to balance yield/performance

Combinational Timing Summary

- Circuit outputs change some time **after** the inputs change
  - Delay is dependent on inputs, environmental state, etc.

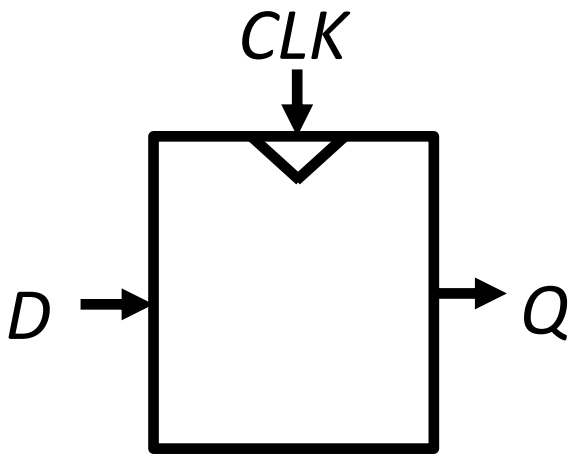
- The range of possible delays is characterized by:
  - **Contamination delay (t_{cd}):** *minimum possible* delay
  - **Propagation delay (t_{pd}):** *maximum possible* delay

- Delays **change** with:
  - Circuit design (e.g., topology, materials)
  - Operating conditions

Sequential Circuit Timing

Recall: D Flip-Flop

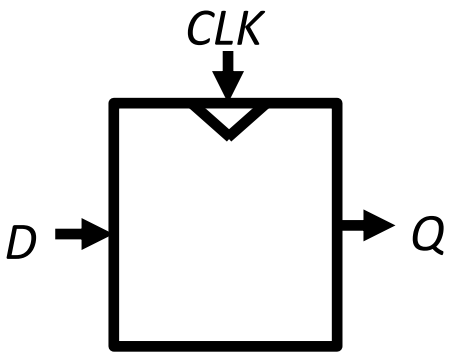
- Flip-flop **samples D** at the **active clock edge**
  - It outputs the **sampled value** to Q
  - It **"stores"** the **sampled value** until the next active clock edge

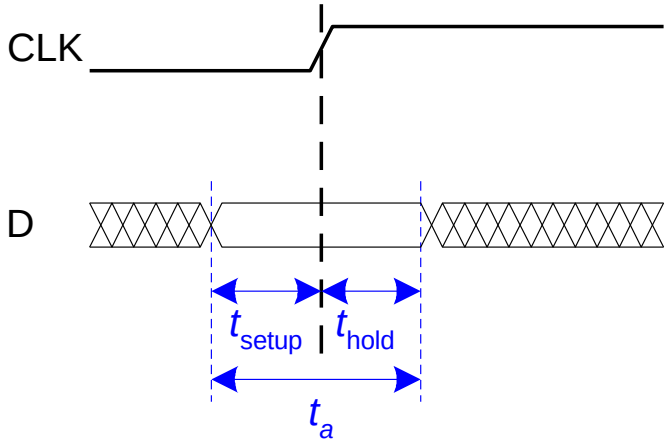


- The D flip-flop is **made** from **combinational** elements
- **D, Q, CLK all have timing requirements!**

D Flip-Flop Input Timing Constraints

- **D** must be **stable** when **sampled** (i.e., at active clock edge)





- ***Setup time*** **(t_{setup}):**
  - Time **before** the clock edge that data must be stable (i.e. not changing)
- ***Hold time*** **(t_{hold}):**
  - Time **after** the clock edge that data must be stable
- ***Aperture time*** **(t_a):**
  - Time **around** clock edge that data must be stable **($t_a = t_{setup} + t_{hold}$)**

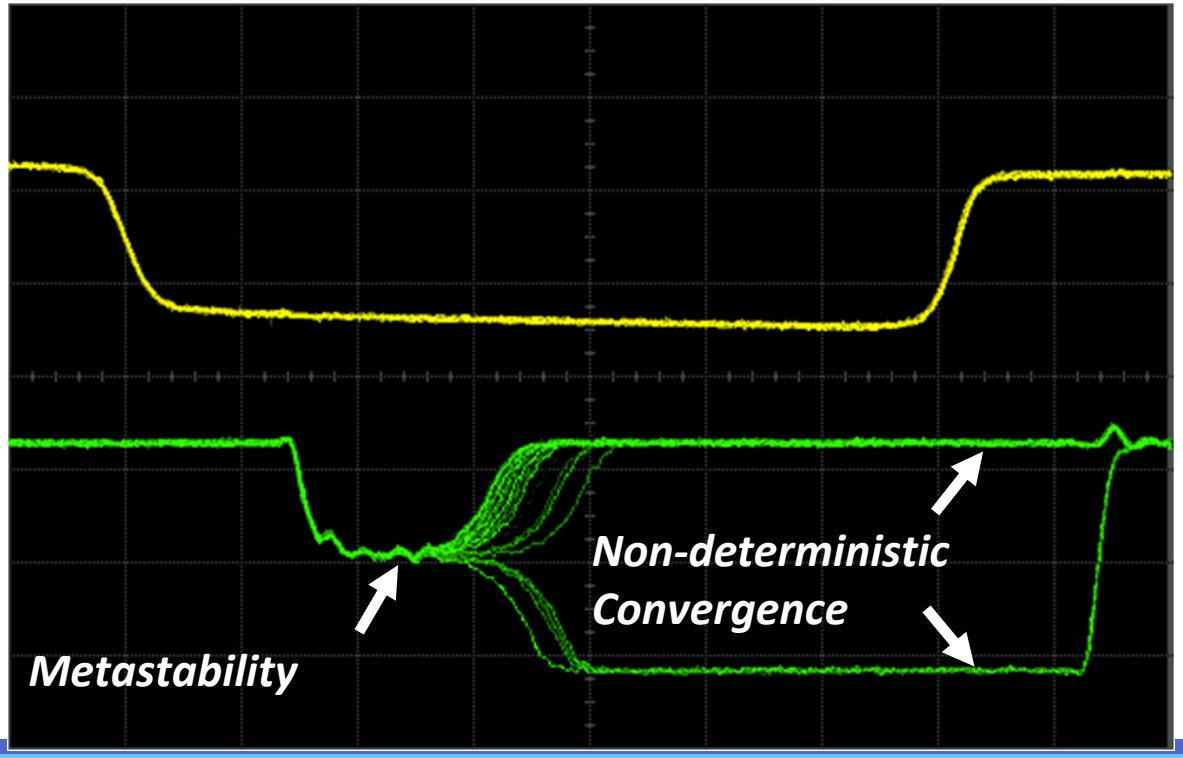
Violating Input Timing: Metastability

- If D is **changing** when sampled, ***metastability*** can occur
  - Flip-flop output is **stuck** somewhere between '1' and '0'
  - Output eventually settles **non-deterministically**

Example Timing Violations (NAND RS Latch)

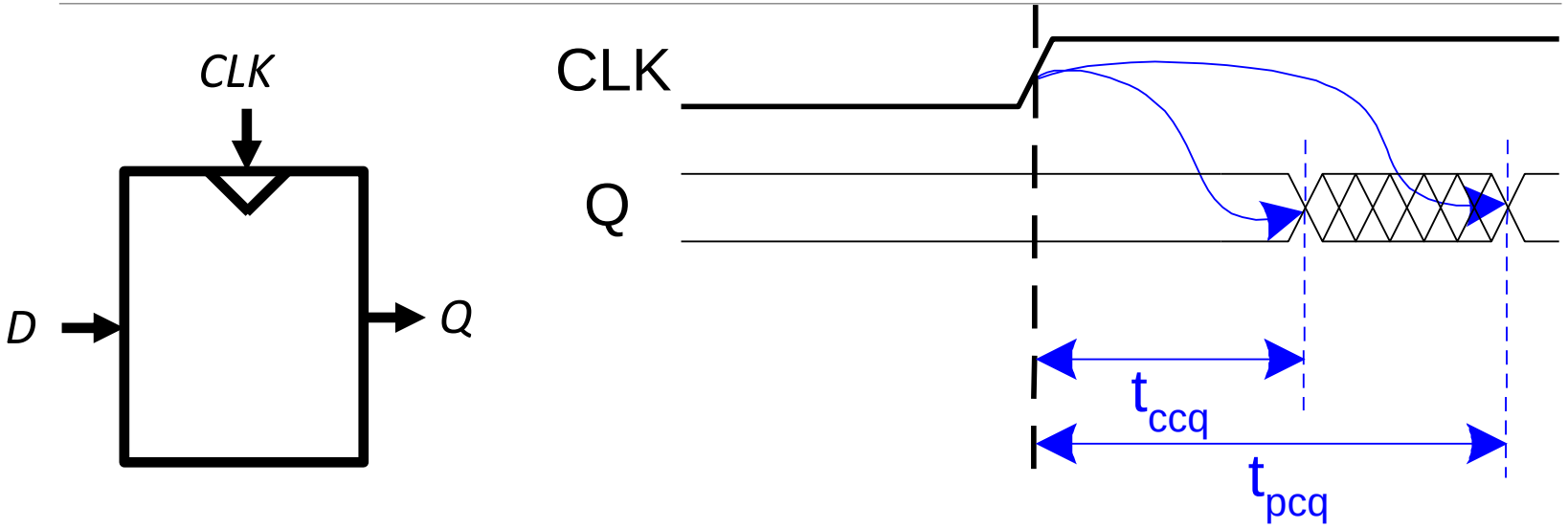
CLK

Q



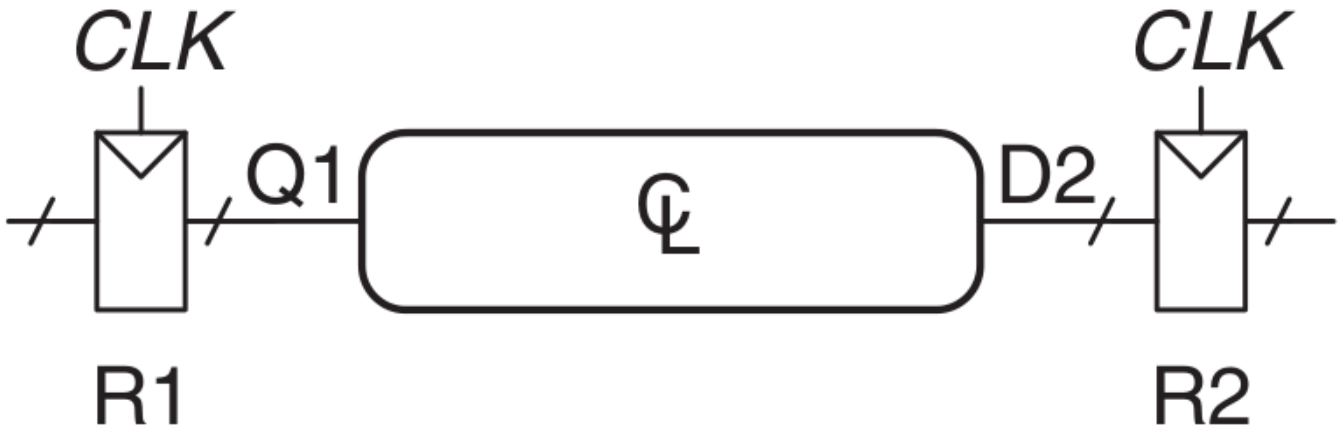
Source: W. J. Dally, Lecture notes for EE108A, Lecture 13: Metastability and Synchronization Failure (When Good Flip-Flops go Bad) 11/9/2005.

Flip-Flop Output Timing



- **Contamination delay clock-to-q (t_{ccq}):**
  - Earliest time after the clock edge that Q starts to change (i.e., is unstable)
- **Propagation delay clock-to-q (t_{pcq}):**
  - Latest time after the clock edge that Q stops changing (i.e., is stable)

Recall: Sequential System Design



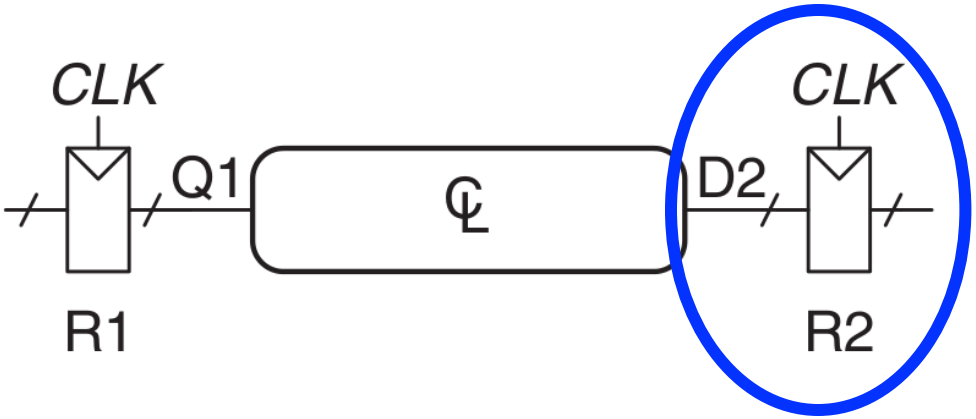
- Multiple **flip-flops** are connected with **combinational logic Clock** runs with period T_c (cycle time)

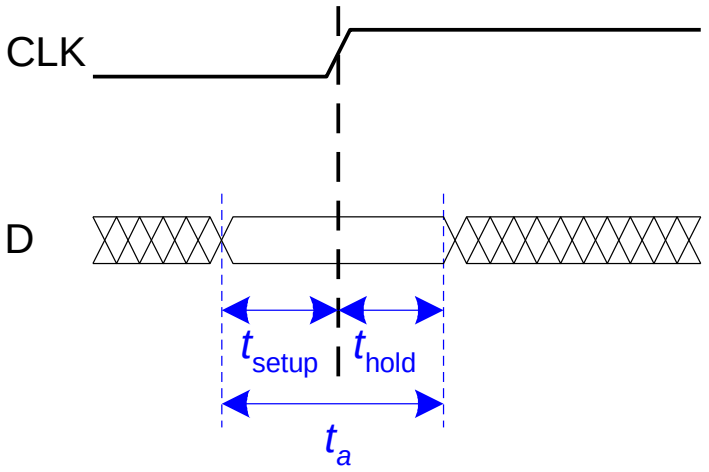
Must meet timing requirements for both R1 and R2!

Ensuring Correct Sequential Operation

- Need to ensure correct input timing on **R2**

- Specifically, **D2** must be **stable**:
  - at least t_{setup} **before** the clock edge
  - at least until t_{hold} **after** the clock edge



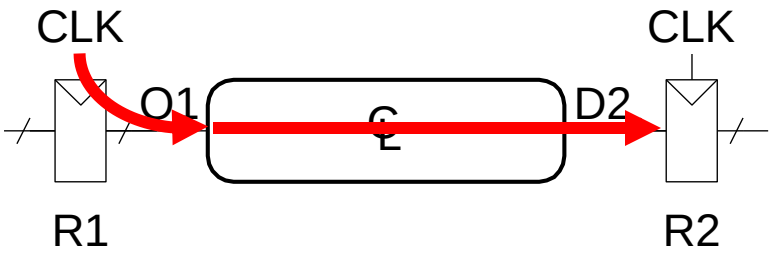


Ensuring Correct Sequential Operation

- This means there is both a **minimum** and **maximum** delay between two flip-flops

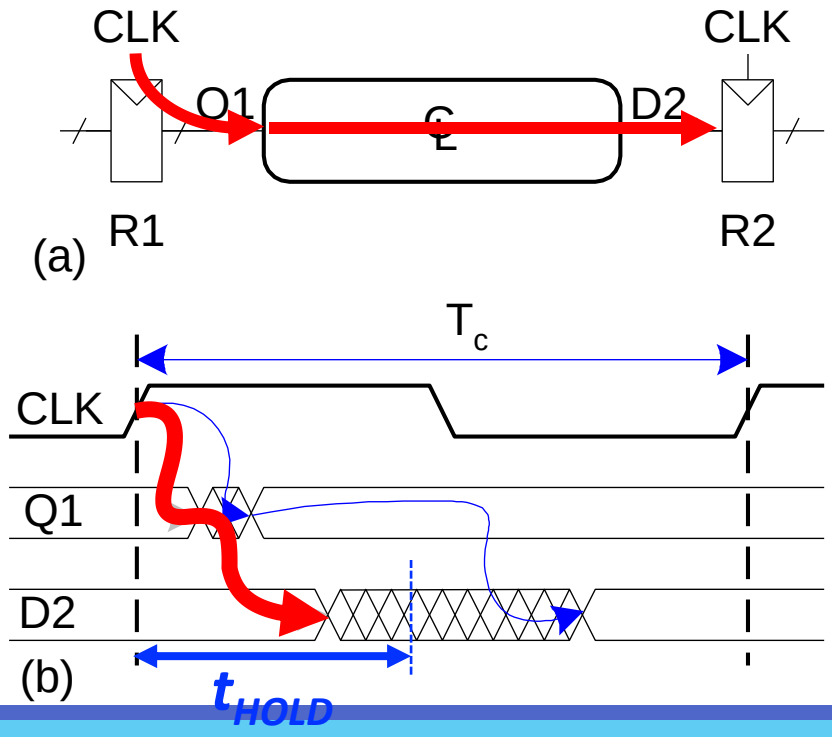
Ensuring Correct Sequential Operation: FAST CL

- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too fast** -> R2 t_{hold} violation



Ensuring Correct Sequential Operation: FAST CL

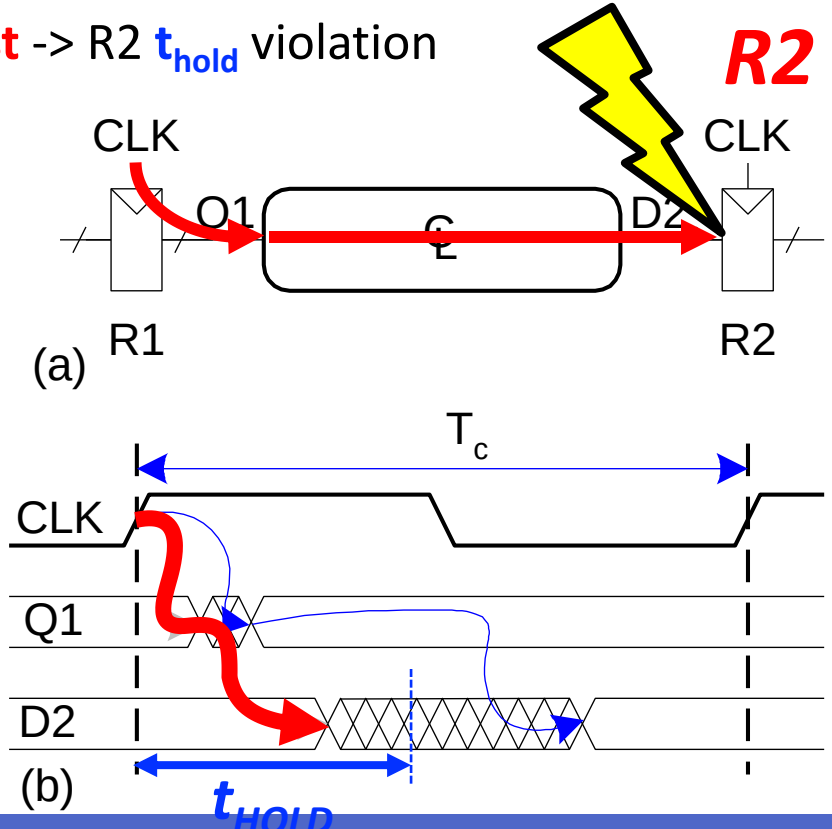
- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too fast** -> R2 t_{hold} violation



Ensuring Correct Sequential Operation: FAST CL

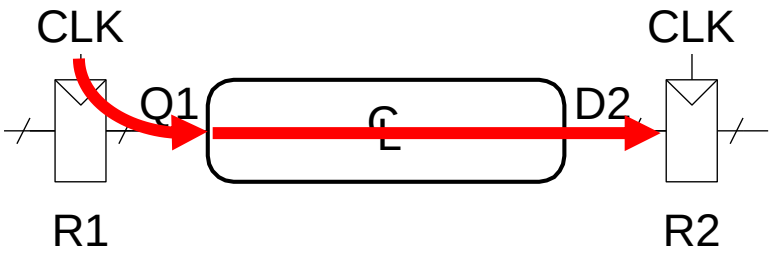
- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too fast** -> R2 t_{hold} violation

Potential R2 t_{HOLD} VIOLATION!



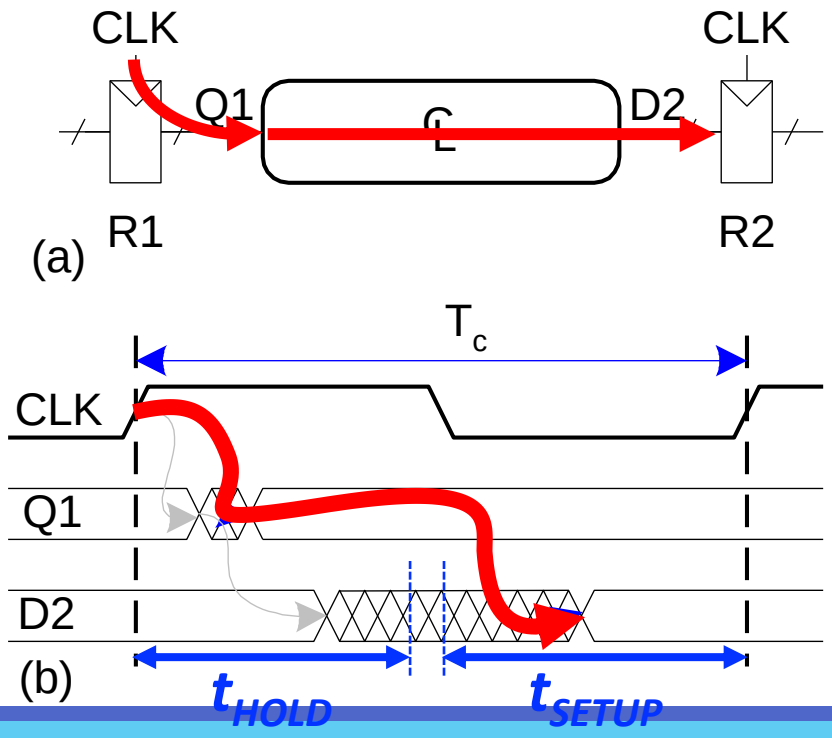
Ensuring Correct Sequential Operation: SLOW CL

- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too slow** -> R2 t_{setup} violation



Ensuring Correct Sequential Operation: SLOW CL

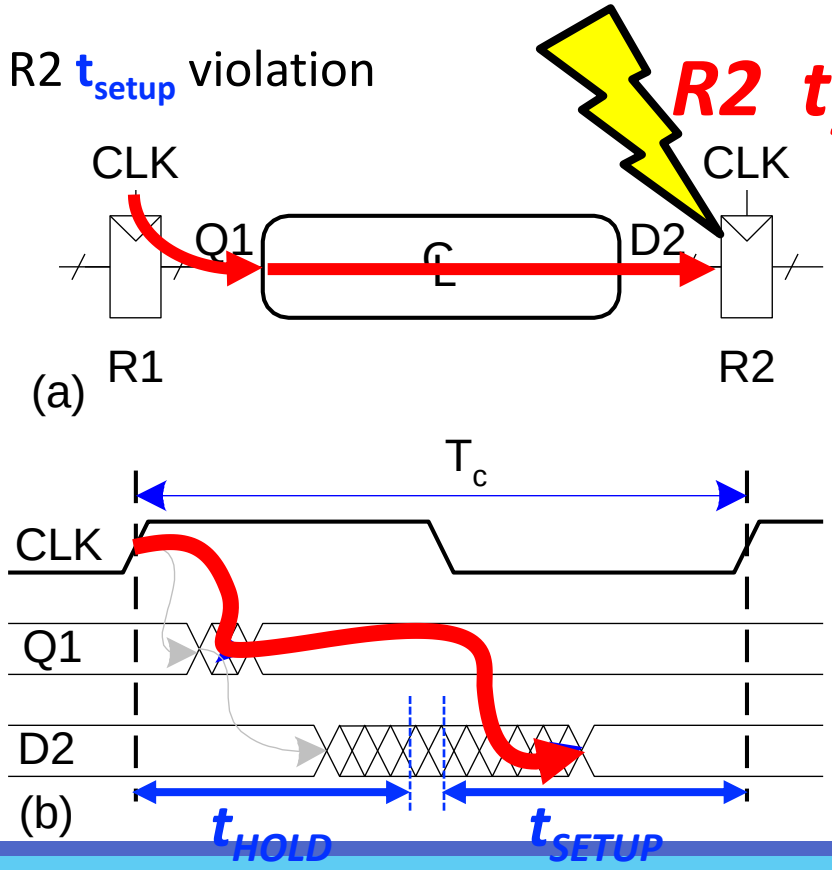
- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too slow** -> R2 t_{setup} violation



Ensuring Correct Sequential Operation: SLOW CL

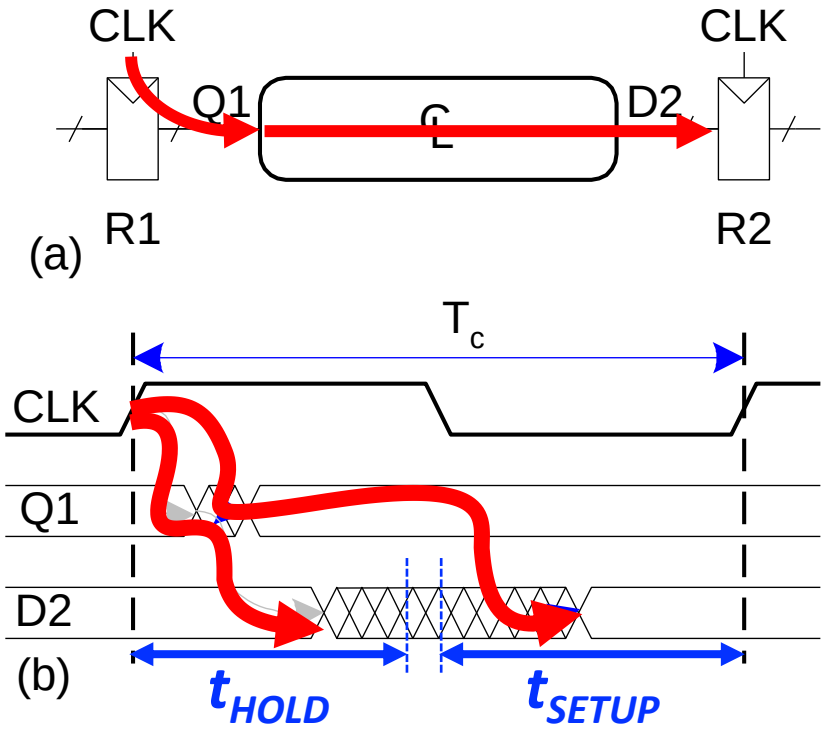
- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too slow** -> R2 t_{setup} violation

Potential R2 t_{SETUP} VIOLATION!



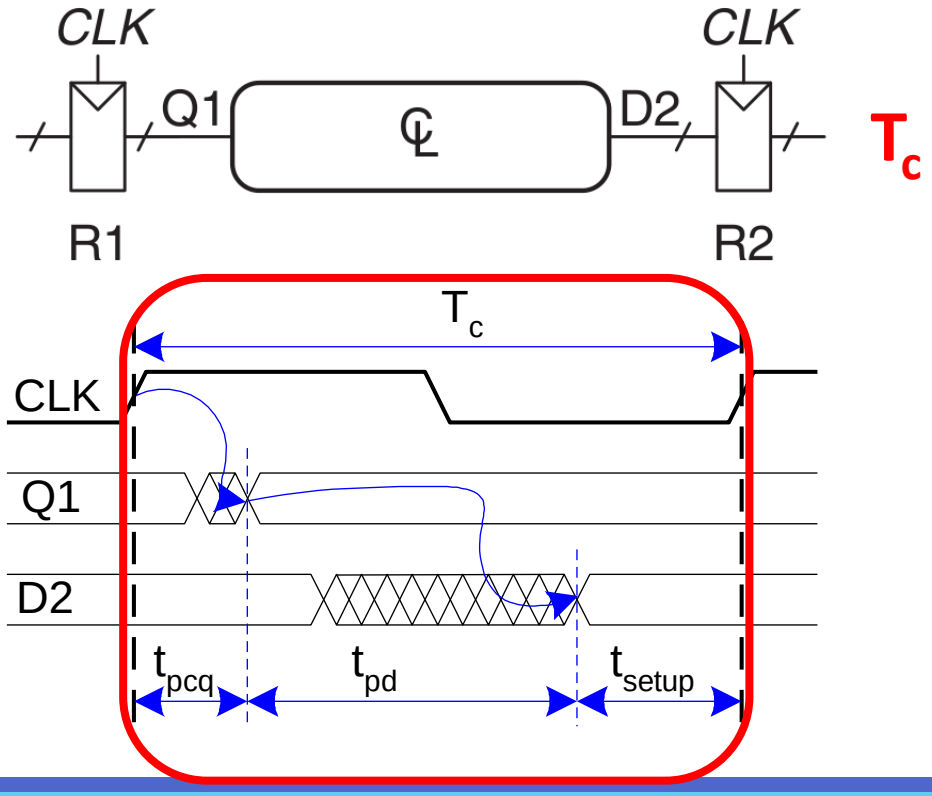
Ensuring Correct Sequential Operation

- This means there is both a **minimum** and **maximum** delay between two flip-flops
- CL **too fast** -> R2 t_{hold} violation
- CL **too slow** -> R2 t_{setup} violation



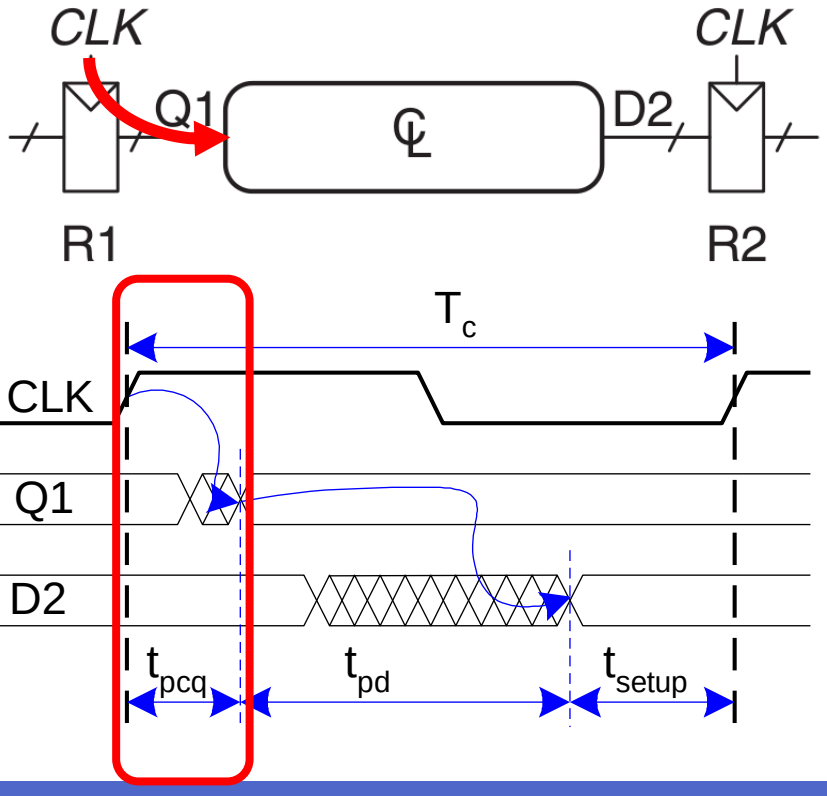
Setup Time Constraint

- **Safe timing** depends on the **maximum** delay from R1 to R2
- The input to R2 must be stable at least t_{setup} **before** the clock edge.



Setup Time Constraint

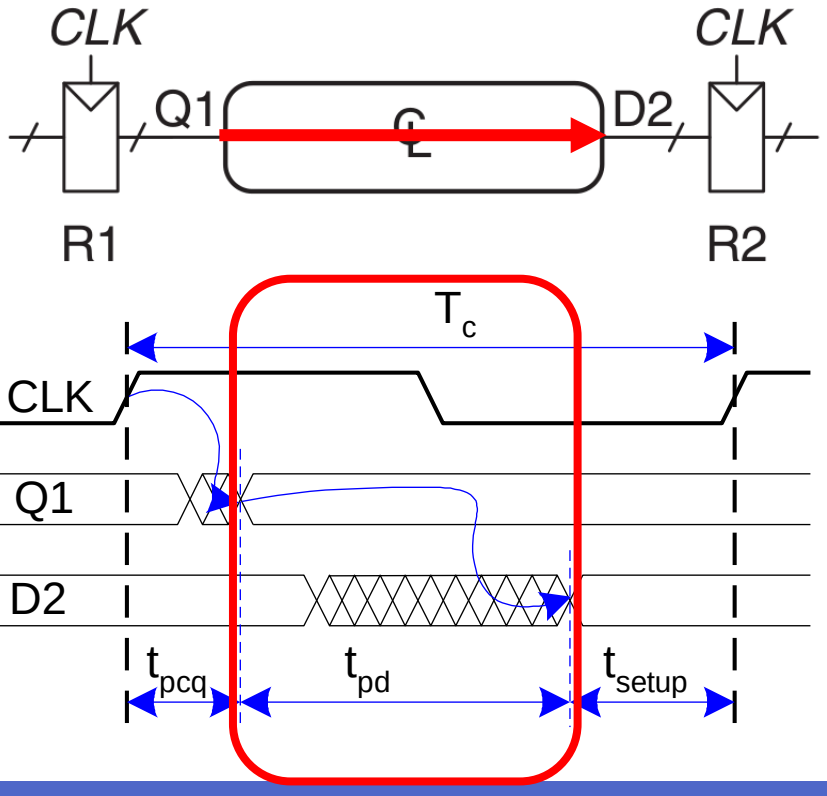
- **Safe timing** depends on the **maximum** delay from R1 to R2
- The input to R2 must be stable at least t_{setup} **before** the clock edge.



T_c ? t_{pcq}

Setup Time Constraint

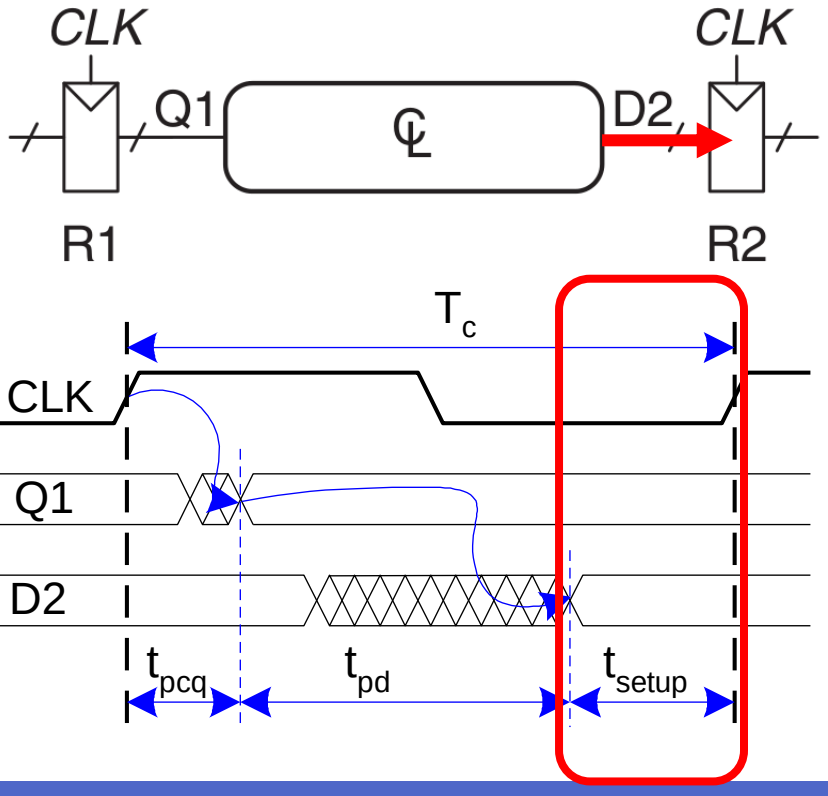
- **Safe timing** depends on the **maximum** delay from R1 to R2
- The input to R2 must be stable at least t_{setup} **before** the clock edge.



$$T_c \geq t_{pcq} + t_{pd}$$

Setup Time Constraint

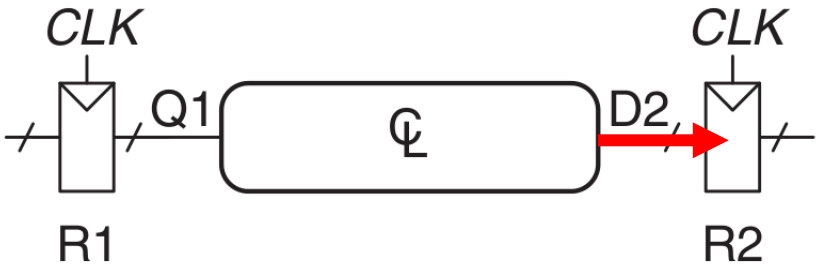
- **Safe timing** depends on the **maximum** delay from R1 to R2
- The input to R2 must be stable at least t_{setup} **before** the clock edge.



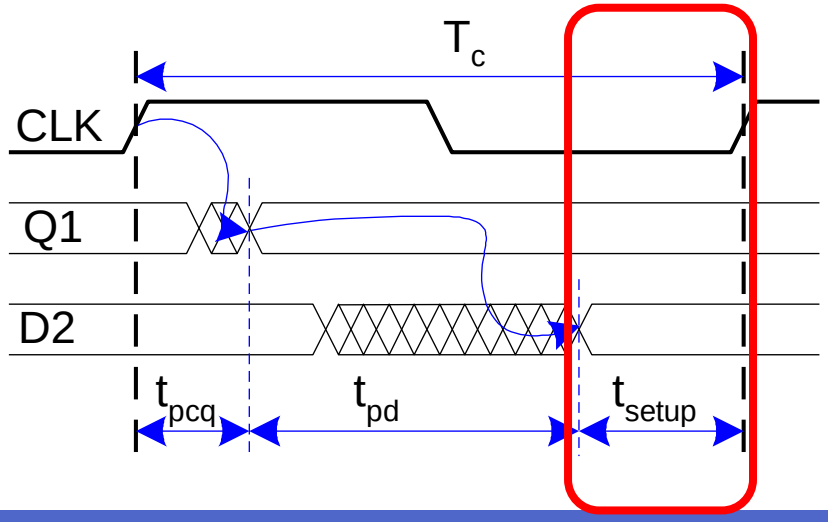
$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

Setup Time Constraint

- **Safe timing** depends on the **maximum** delay from R1 to R2
- The input to R2 must be stable at least t_{setup} **before** the clock edge.

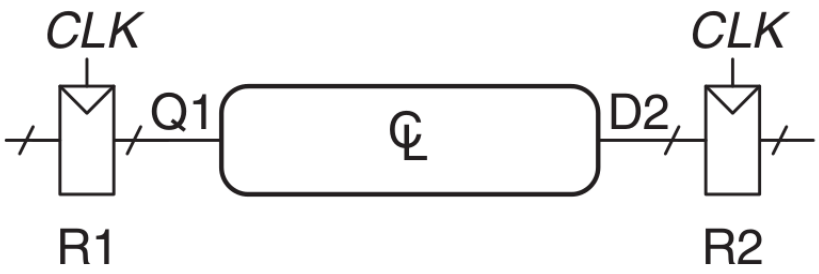


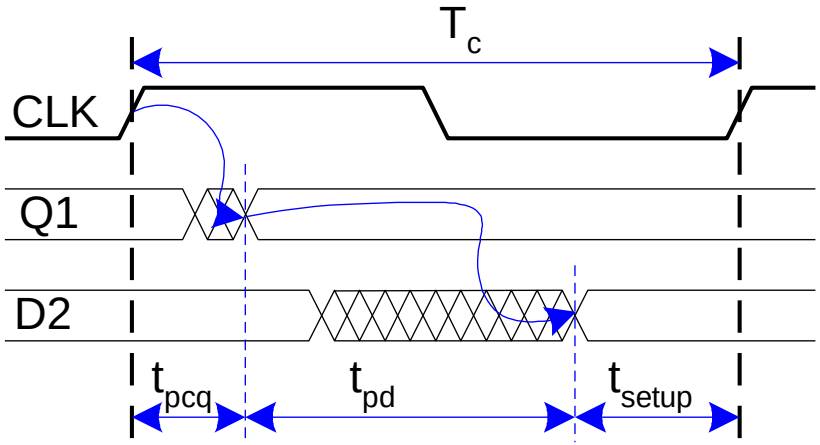
$$T_c > t_{pcq} + t_{pd} + t_{setup}$$



Setup Time Constraint

- **Safe timing** depends on the **maximum** delay from R1 to R2
- The input to R2 must be stable at least t_{setup} **before** the clock edge.





Wasted work

$$T_c > t_{pcq} + t_{pd} + t_{setup}$$

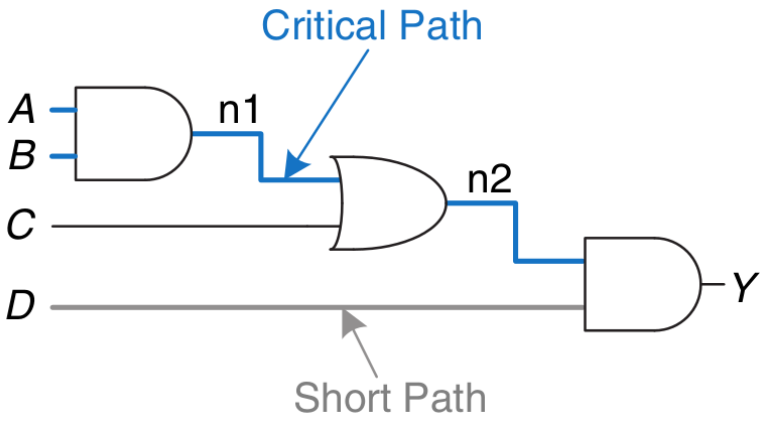
Useful work

Sequencing overhead: amount of time **wasted** each cycle due to sequencing element timing requirements

t_{setup} Constraint and Design Performance

- **Critical path:** path with the longest t_{pd}

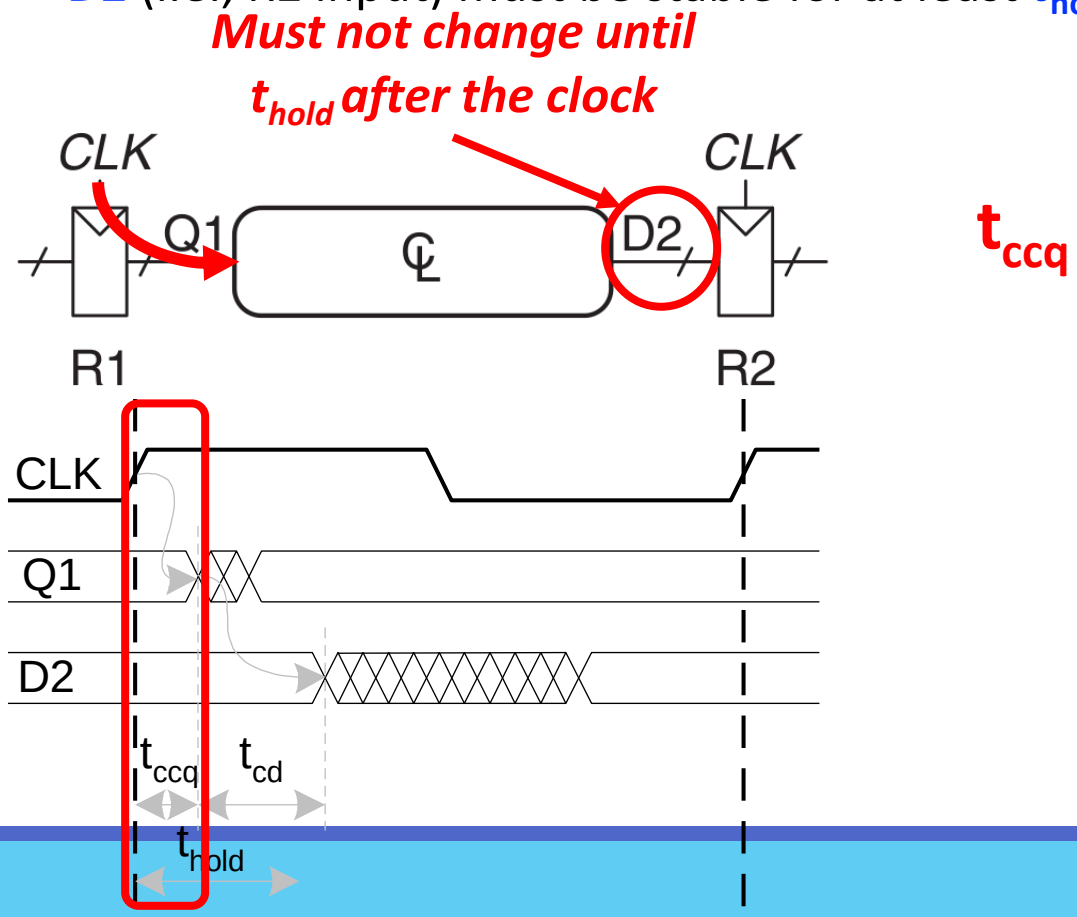
$$T_c > t_{pcq} + t_{pd} + t_{setup}$$



- Overall design performance is determined by the critical path t_{pd}
  - Determines the **minimum clock period** (i.e., **max operating frequency**)
  - If the critical path is too **long**, the design will run **slowly**
  - If critical path is too **short**, each cycle will do very **little useful work**
    - i.e., most of the cycle will be **wasted** in sequencing overhead

Hold Time Constraint

- **Safe timing** depends on the **minimum** delay from R1 to R2
- **D2** (i.e., R2 input) must be stable for at least t_{hold} **after** the clock edge

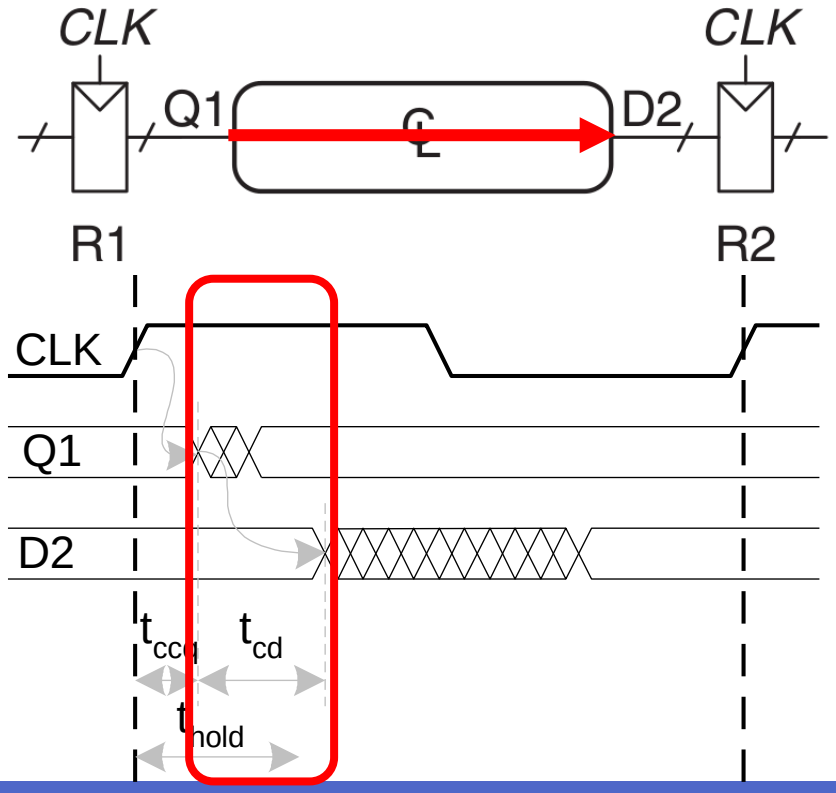


t_{ccq}

Hold Time Constraint

Safe timing depends on the **minimum** delay from R1 to R2

D2 (i.e., R2 input) must be stable for at least t_{hold} **after** the clock edge

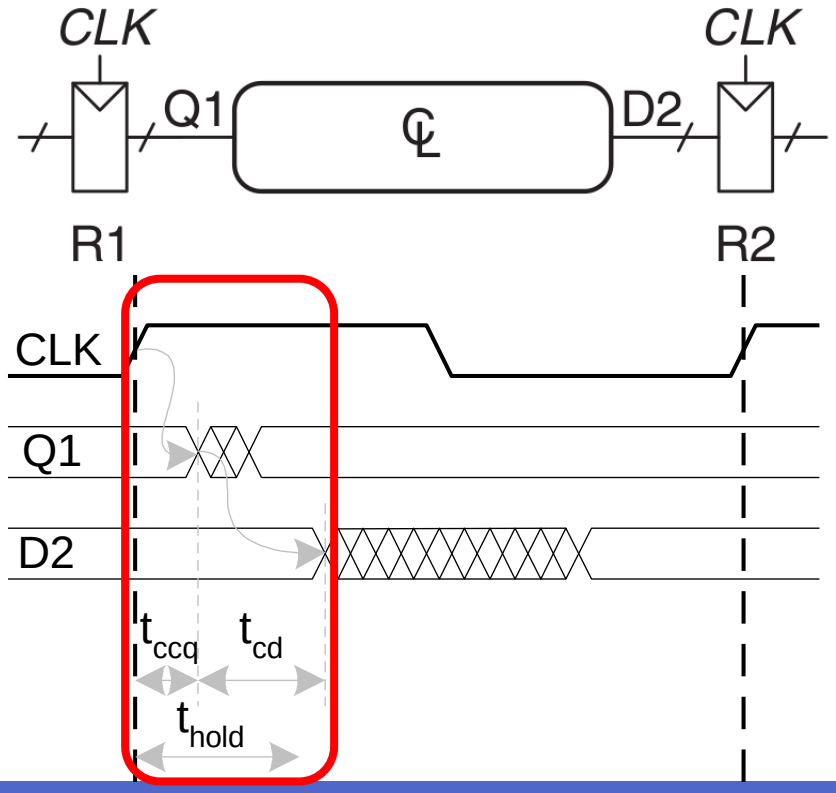


$t_{ccq} + t_{cd}$

Hold Time Constraint

Safe timing depends on the **minimum** delay from R1 to R2

D2 (i.e., R2 input) must be stable for at least t_{hold} **after** the clock edge

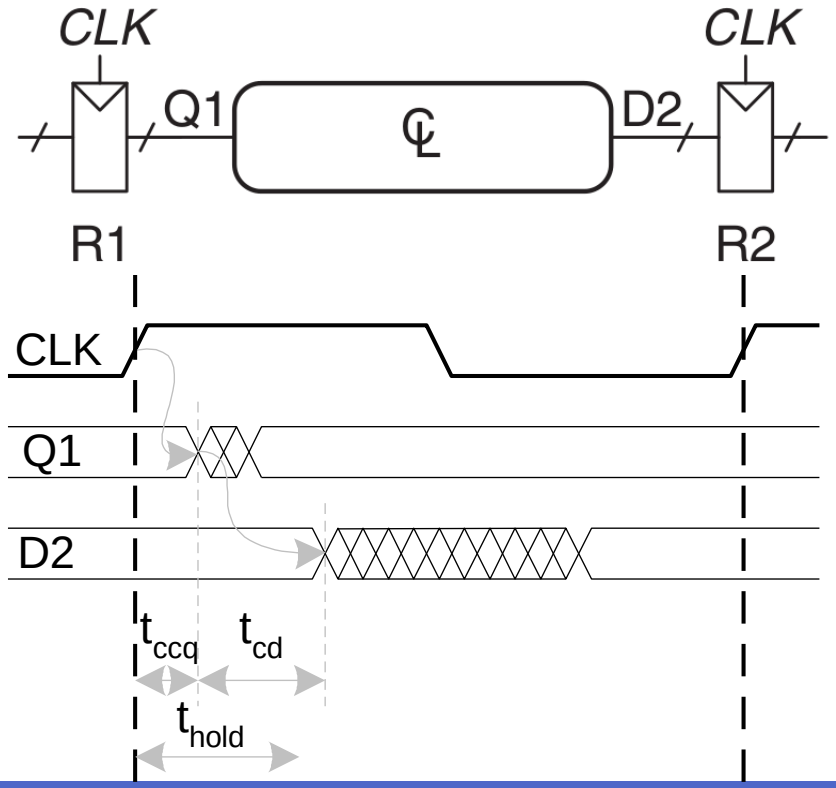


$$t_{ccq} + t_{cd} > t_{hold}$$

Hold Time Constraint

Safe timing depends on the **minimum** delay from R1 to R2

D2 (i.e., R2 input) must be stable for at least t_{hold} **after** the clock edge



$$t_{ccq} + t_{cd} > t_{hold}$$

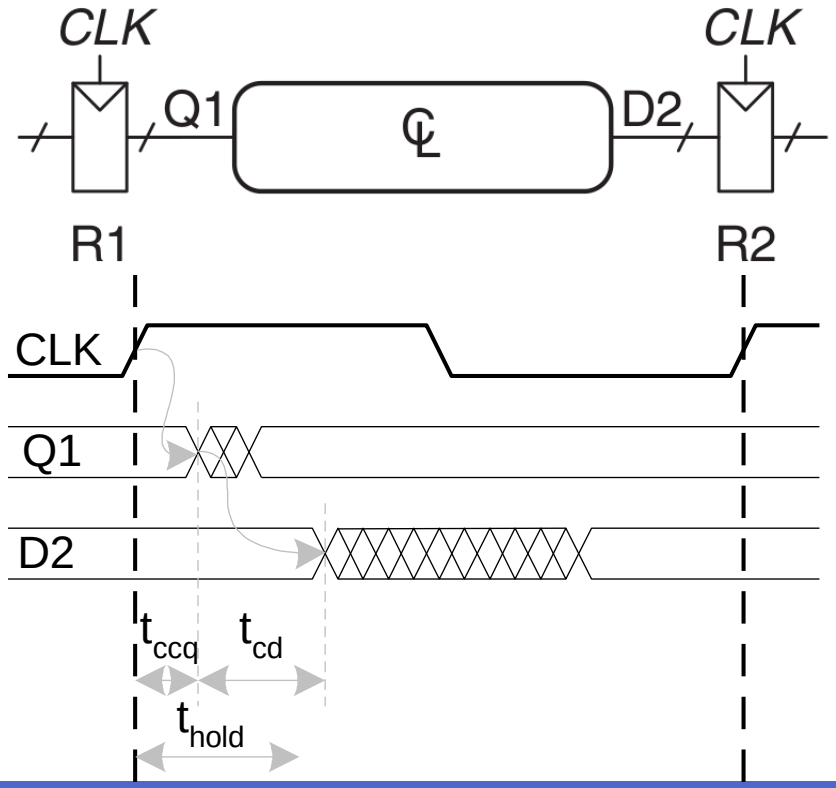
$$t_{cd} > t_{hold} - t_{ccq}$$

We need to have a **minimum** combinational delay!

Hold Time Constraint

Safe timing depends on the **minimum** delay from R1 to R2

D2 (i.e., R2 input) must be stable for at least t_{hold} **after** the clock edge



$$t_{ccq} + t_{cd} > t_{hold}$$

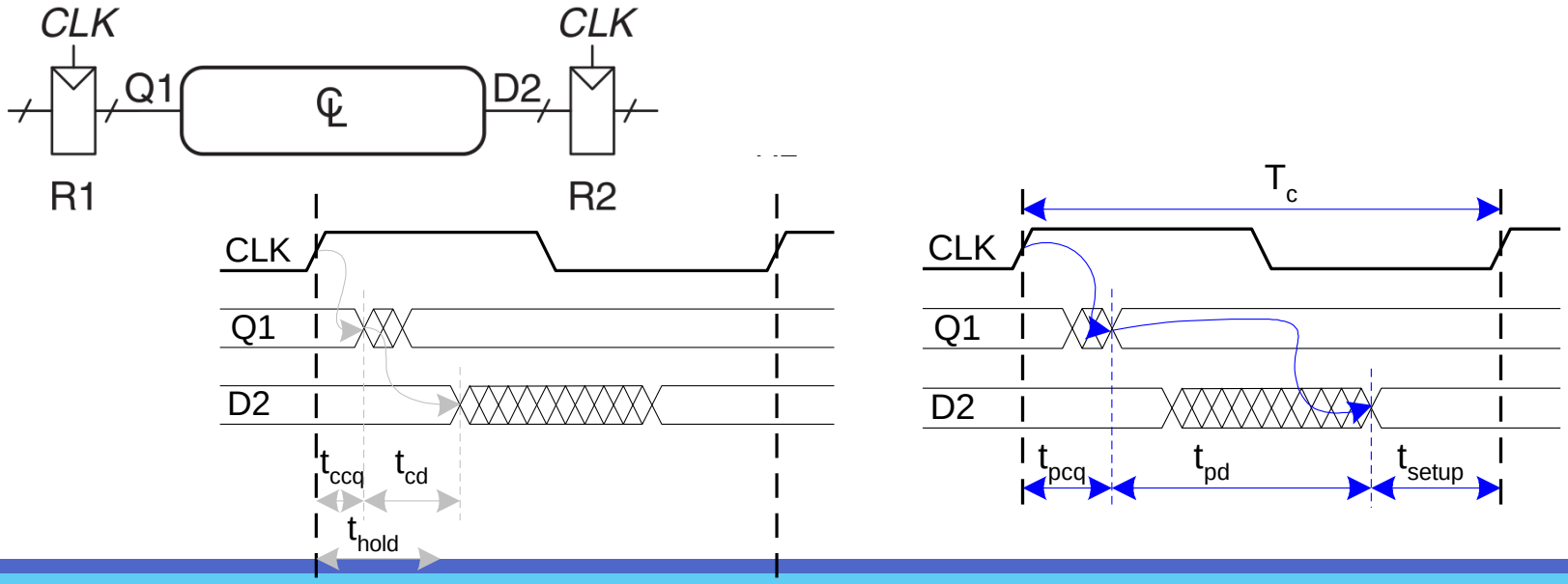
$$t_{cd} > t_{hold} - t_{ccq}$$

Does **NOT** depend on T_c!

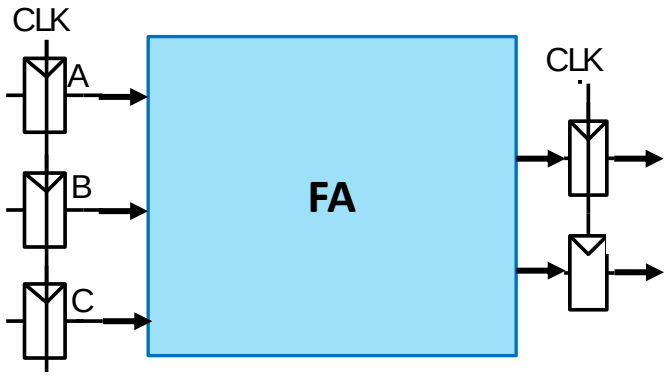
Very hard to fix t_{hold} violations after manufacturing- must modify circuits!

Sequential Timing Summary

t_{ccq} / t_{pcq}	clock-to-q delay (contamination/propagation)
t_{cd} / t_{pd}	combinational logic delay (contamination/propagation)
t_{setup}	time that FF inputs must be stable before next clock edge
t_{hold}	time that FF inputs must be stable after a clock edge
T_c	clock period



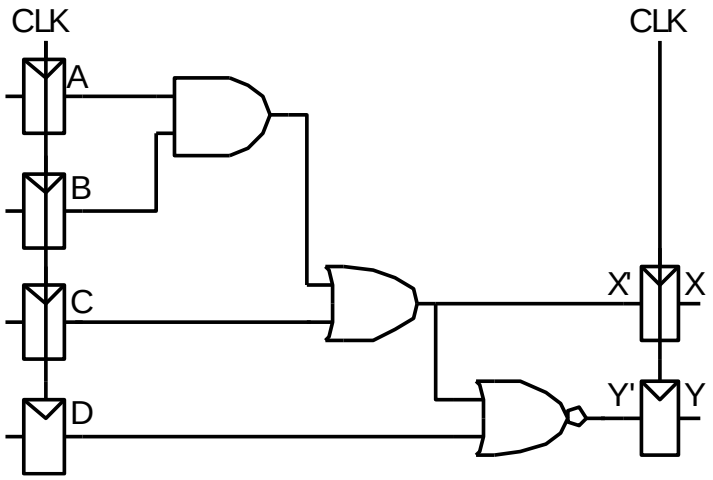
Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 90 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

Example: Timing Analysis

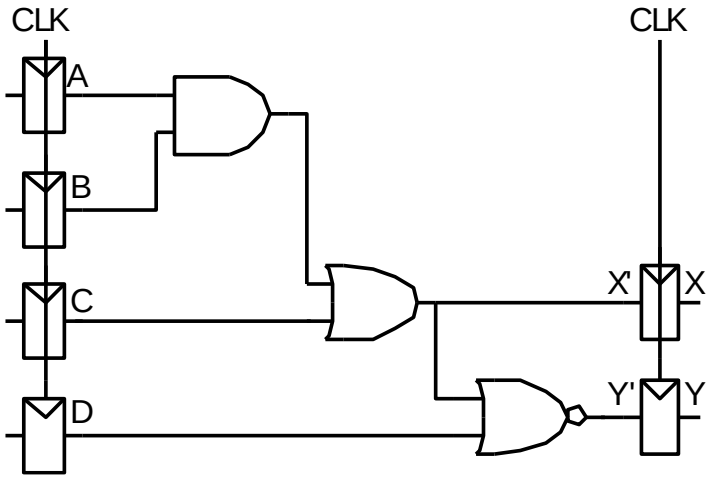


Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd}	= 35 ps
t_{cd}	= 25 ps

per gate

Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} =

t_{cd} =

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

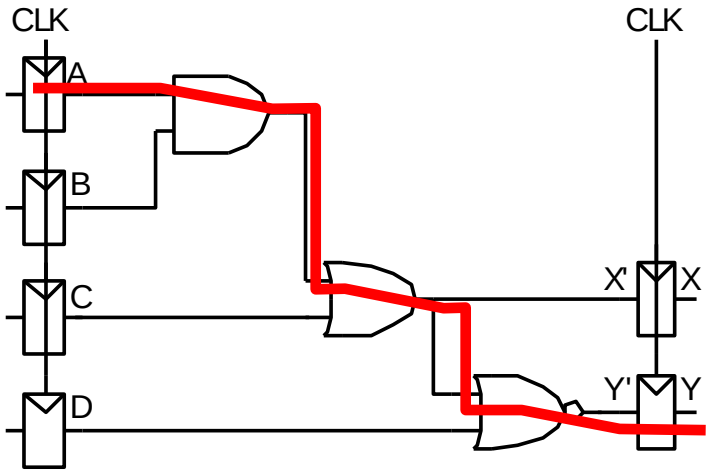
$T_c >$

$f_{max} = 1/T_c =$

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} =

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

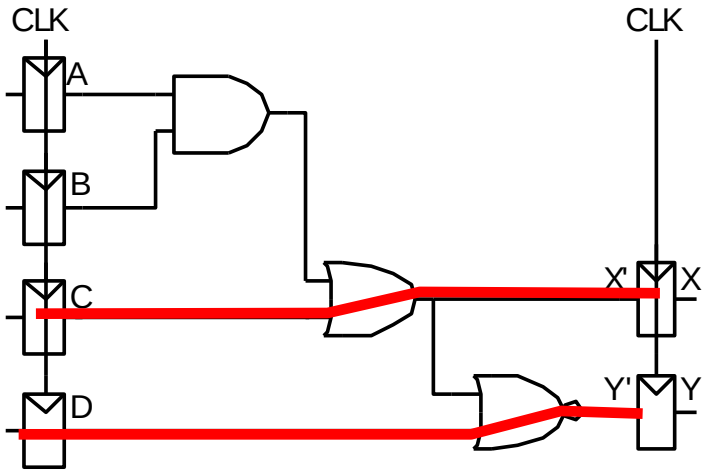
$T_c >$

$f_{max} = 1/T_c =$

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

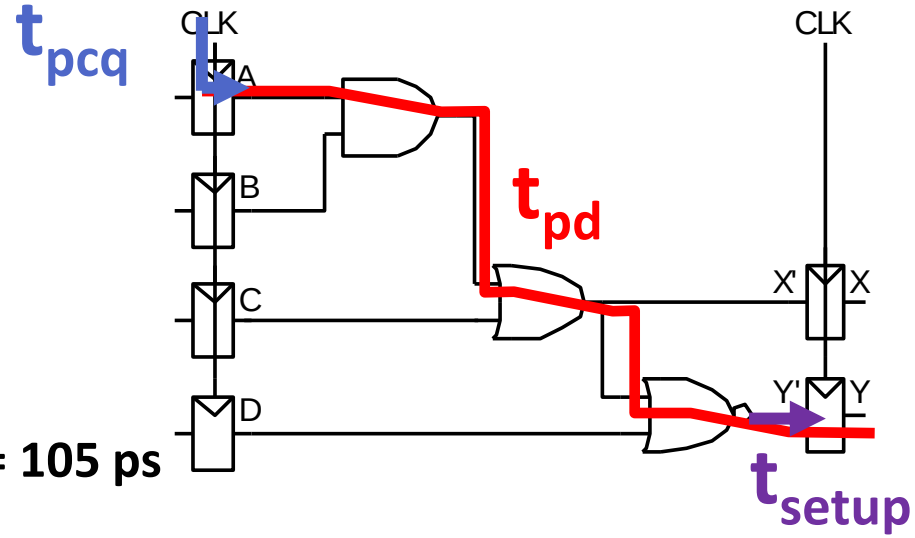
$T_c >$

$f_{max} = 1/T_c =$

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
per gate t_{pd}	= 35 ps
per gate t_{cd}	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

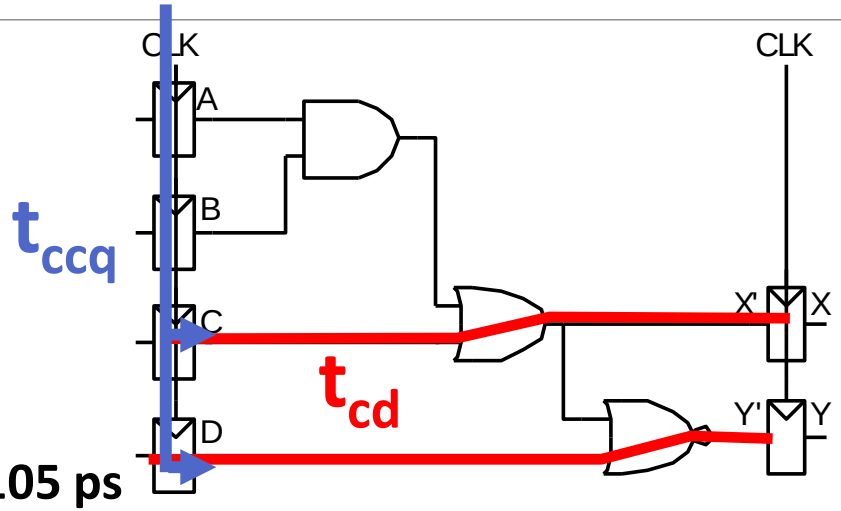
$T_c >$ (50 + 105 + 60) ps = 215 ps

$f_{max} = 1/T_c$ = 4.65 GHz

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c > (50 + 105 + 60)$ ps = 215 ps

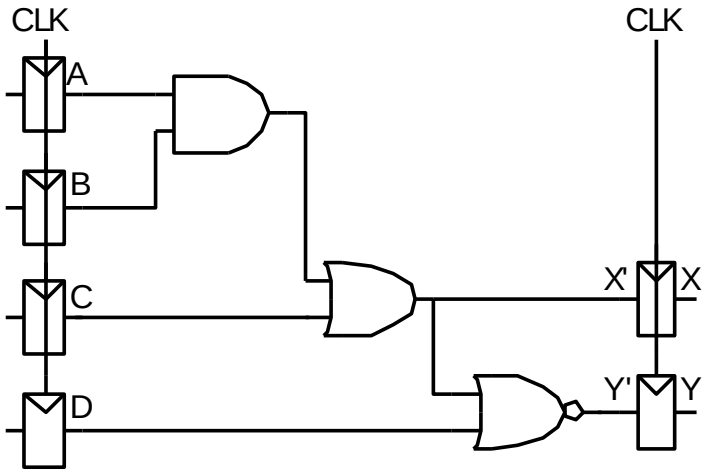
$f_{max} = 1/T_c = 4.65$ GHz

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 25) ps > 70 ps ?

Example: Timing Analysis



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c > (50 + 105 + 60)$ ps = 215 ps

$f_{max} = 1/T_c = 4.65$ GHz

Check hold time constraints:

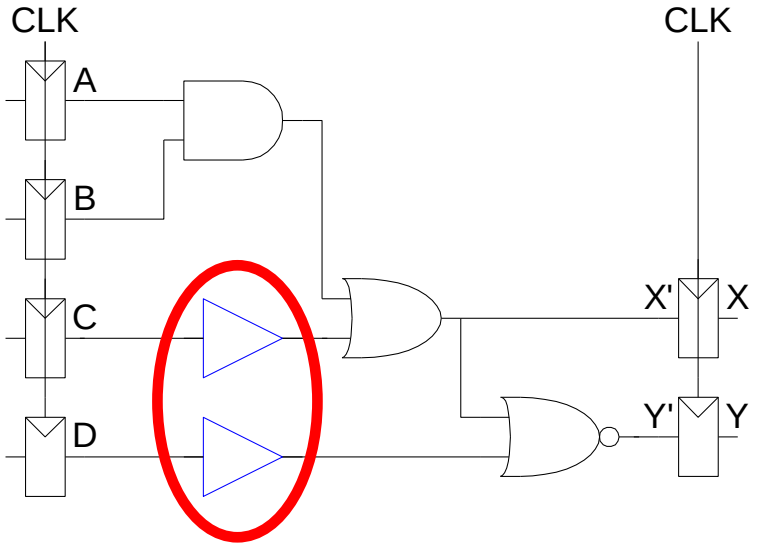
$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 25) ps > 70 ps ?

FAIL

Example: Fixing Hold Time Violation

Add buffers to the short paths:



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} =

t_{cd} =

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c >$

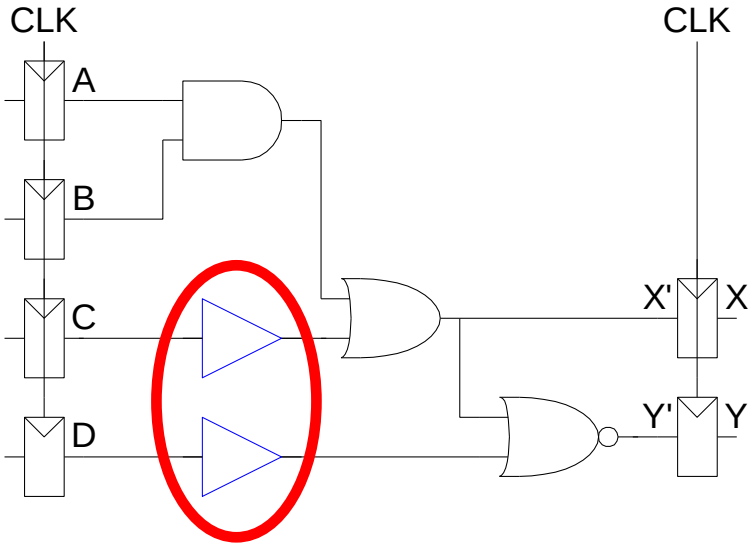
$f_c =$

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Fixing Hold Time Violation

Add buffers to the short paths:



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c >$

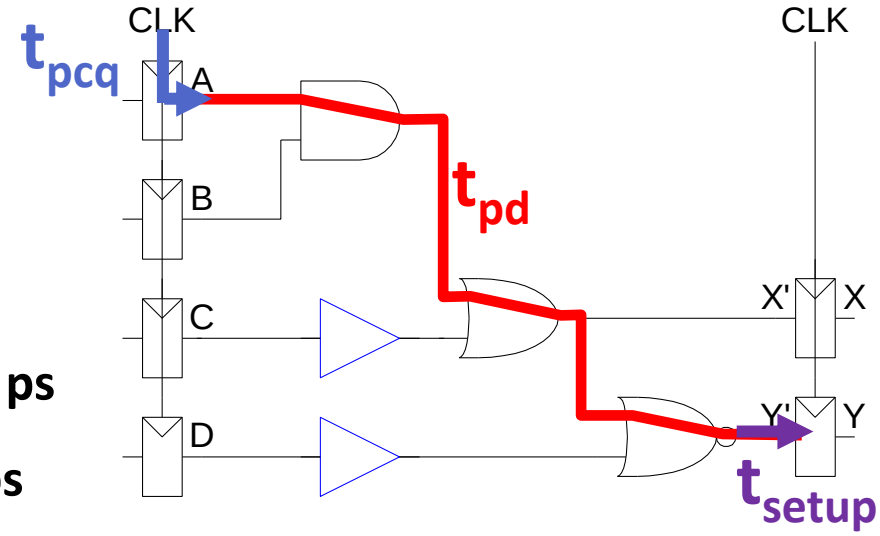
$f_c =$

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Fixing Hold Time Violation

Add buffers to the short paths:



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c >$ (50 + 105 + 60) ps = 215 ps

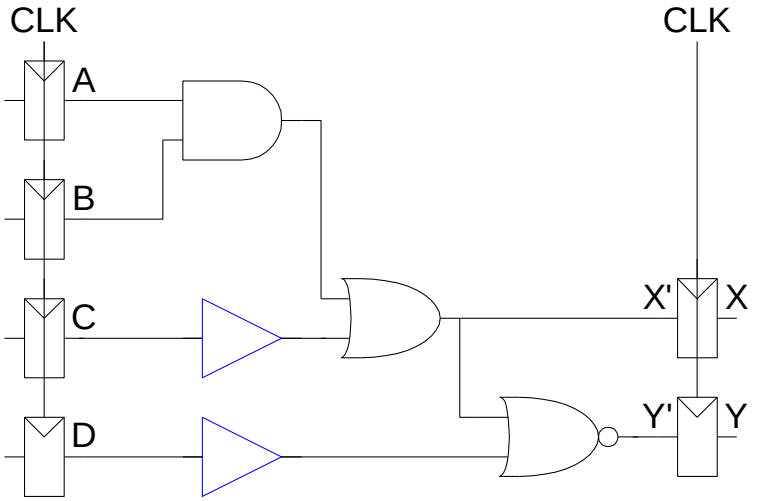
$f_c = 1/T_c$ = 4.65 GHz

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Fixing Hold Time Violation

Add buffers to the short paths:



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c > (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c =$ 4.65 GHz

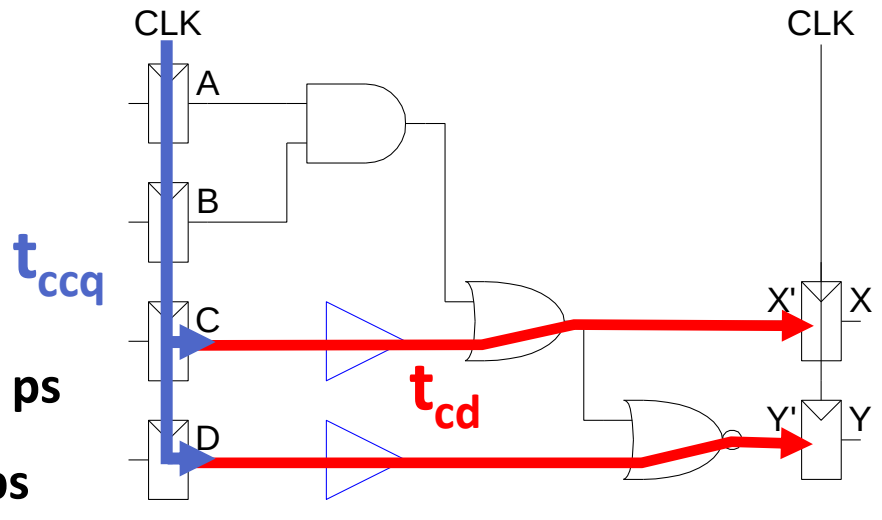
Note: no change to max frequency!

Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Fixing Hold Time Violation

Add buffers to the short paths:



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c > (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c = 4.65$ GHz

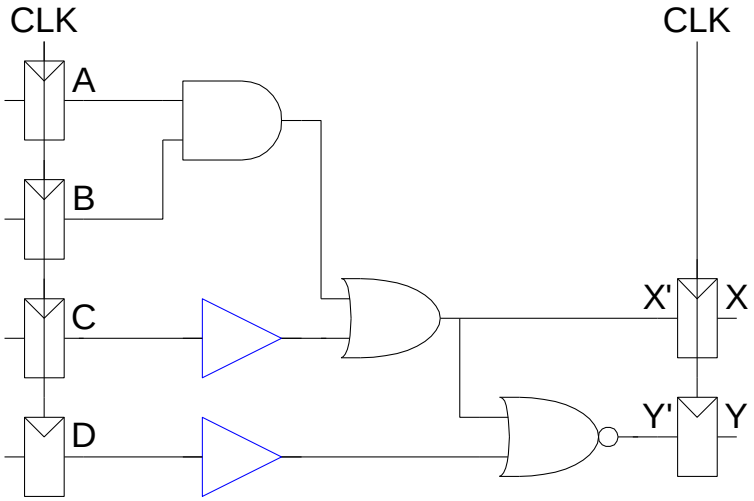
Check hold time constraints:

$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 50) ps > 70 ps ?

Example: Fixing Hold Time Violation

Add buffers to the short paths:



Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd} (per gate)	= 35 ps
t_{cd} (per gate)	= 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Check setup time constraints:

$T_c > t_{pcq} + t_{pd} + t_{setup}$

$T_c > (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c = 4.65$ GHz

Check hold time constraints:

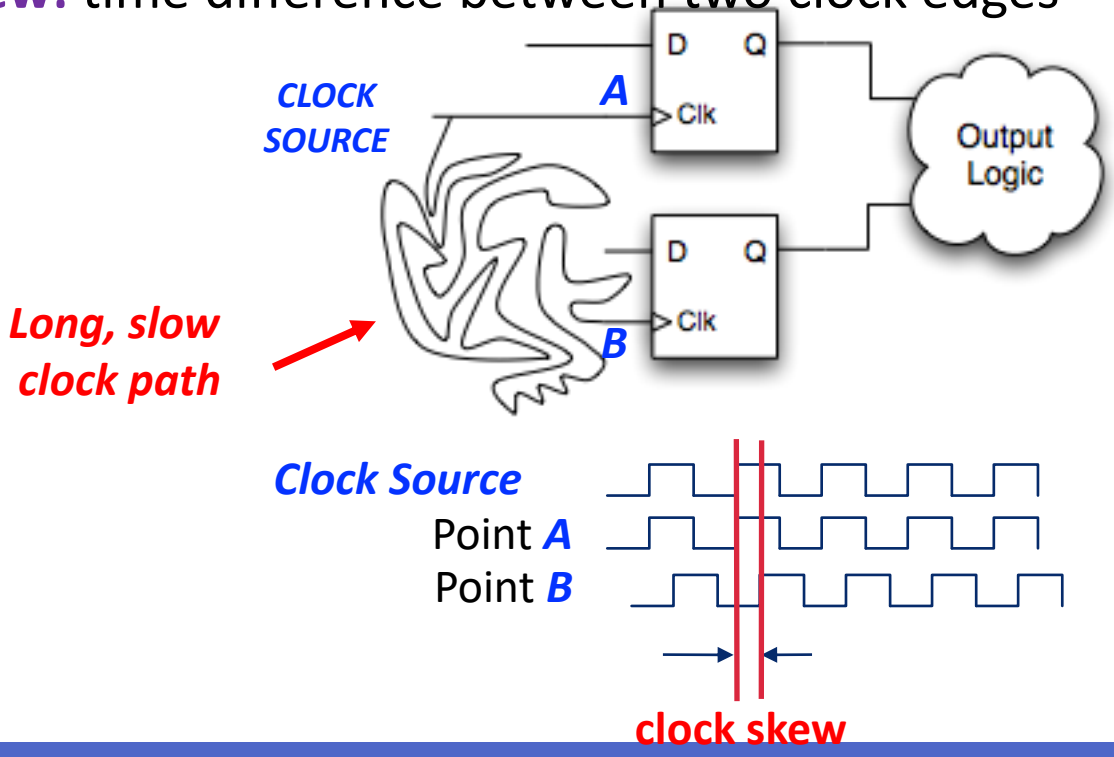
$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 50) ps > 70 ps ?

PASS

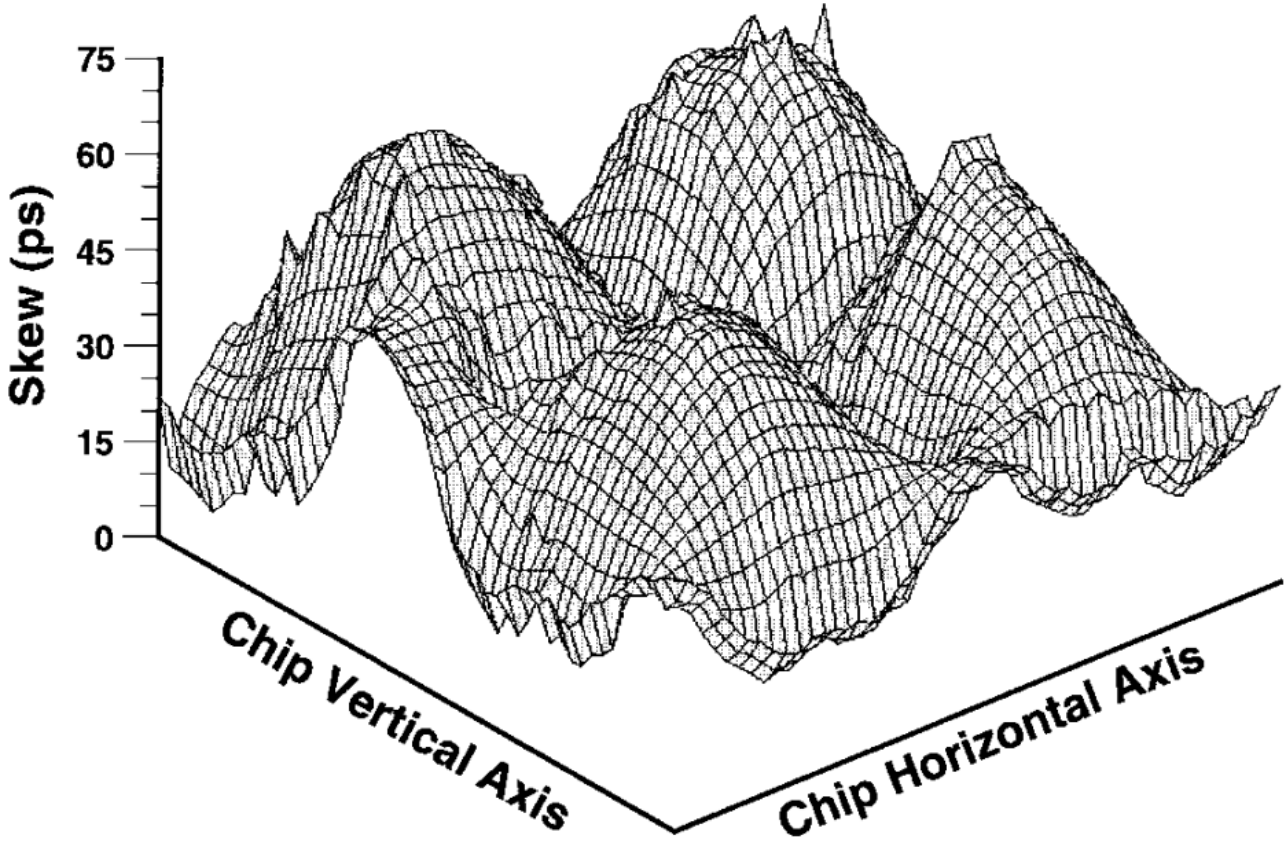
Clock Skew

- To make matters worse, **clocks have delay** too!
  - The clock does **not** reach all parts of the chip at the same time!
- **Clock skew:** time difference between two clock edges



Clock Skew Example

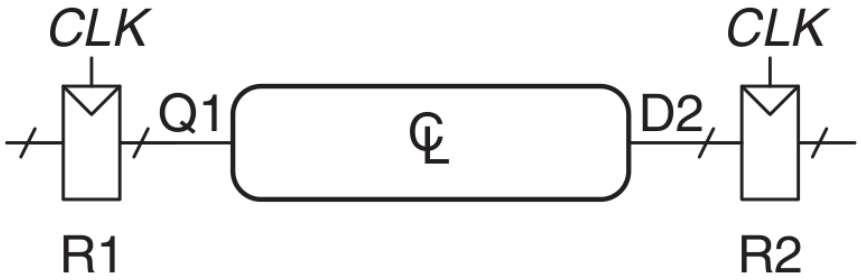
Example of the **Alpha 21264** clock skew spatial distribution

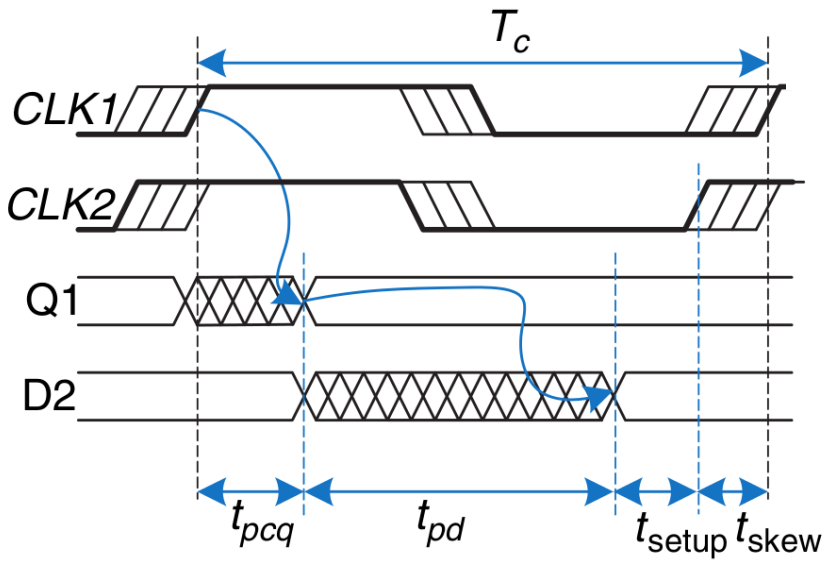


P. E. Gronowski+, "High-performance Microprocessor Design," JSSC'98.

Clock Skew: Setup Time Revisited

- **Safe timing** requires considering the **worst-case skew**
  - Clock arrives at **R2 *before* R1**
  - Leaves **as little time as possible** for the **combinational logic**





Signal must arrive at D2 ***earlier***!

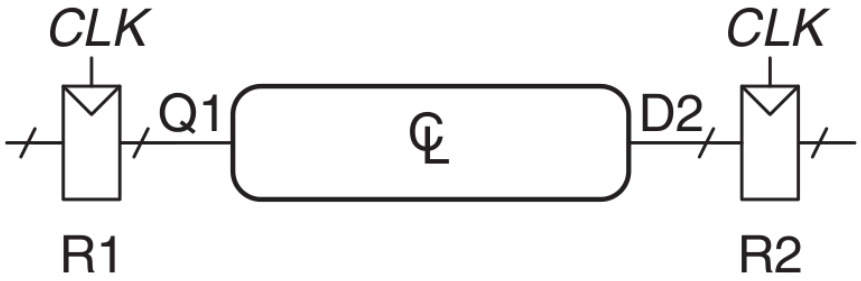
This effectively ***increases*** t_{setup}:

$$T_c > t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

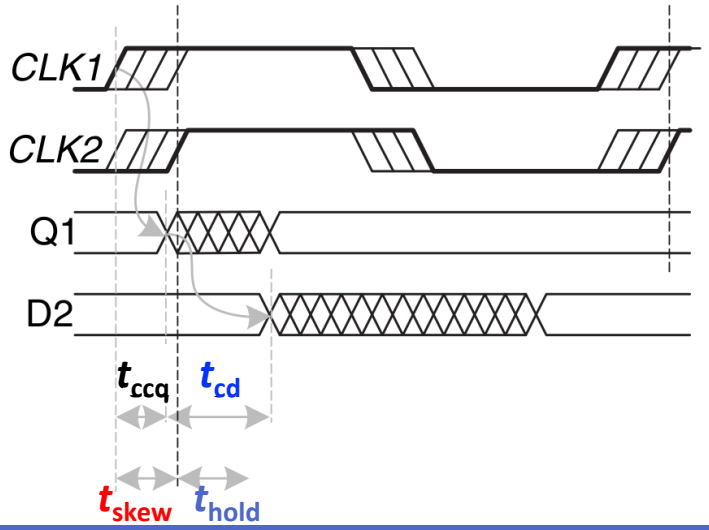
$$T_c > t_{pcq} + t_{pd} + t_{setup,\ effective}$$

Clock Skew: Hold Time Revisited

- **Safe timing** requires considering the **worst-case skew**
  - Clock arrives at **R2 *after* R1**
  - Increases the **minimum required delay** for the **combinational logic**



Signal must arrive at D2 ***later***!



This effectively ***increases*** t_{hold}:

$$t_{cd} + t_{ccq} > t_{hold} + t_{skew}$$

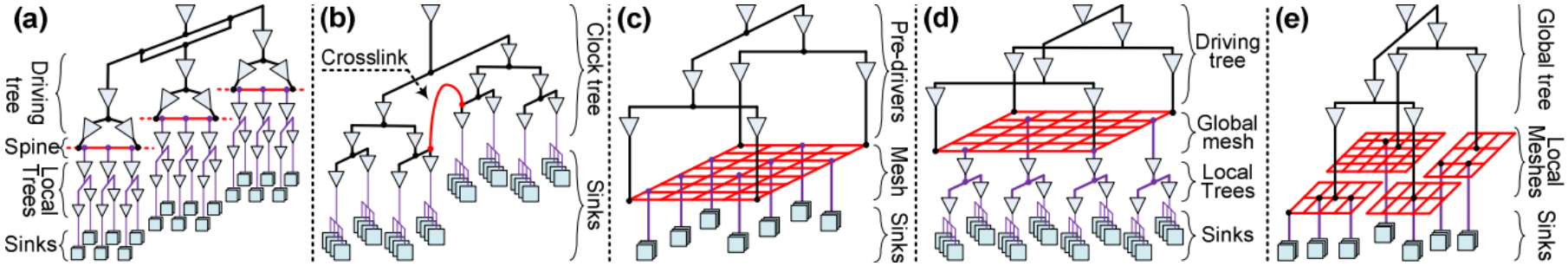
$$t_{cd} + t_{ccq} > t_{hold,\ effective}$$

Clock Skew: Summary

- **Skew** effectively **increases** both $\mathbf{t_{setup}}$ and $\mathbf{t_{hold}}$
  - Increased **sequencing overhead**
  - i.e., less useful work done per cycle

Designers must keep skew to a **minimum**

- Requires intelligent **"clock network"** across a chip
- **Goal: clock** arrives at all locations at roughly the **same time**

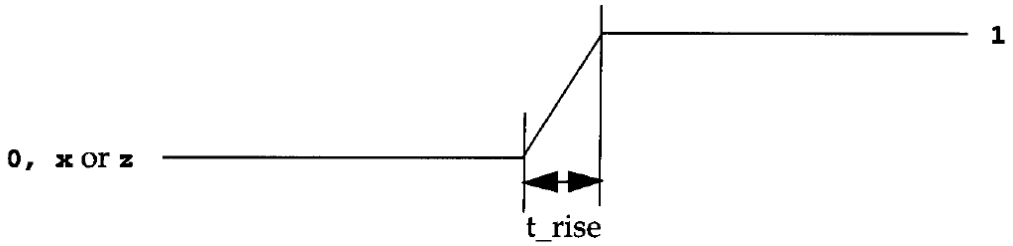


Source: Abdelhadi, Ameer, et al. "Timing-driven variation-aware nonuniform clock mesh synthesis." GLSVLSI'10.

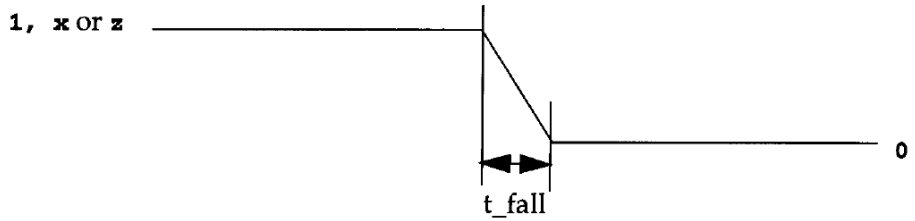
Delay in Verilog

Gate Delay in Verilog

- Rise delay
  - Associated with a gate output transition to a 1 from another value.



- Fall delay
  - Associated with a gate output transition to a 0 from another value.



Gate Delay in Verilog

- Turn-off delay
  - Associated with a gate output transition to the high impedance value (z) from another value

- Delay for value change to x
  - Associated with a gate output transition to unknown value (x) from another value
  - The minimum of the three delays is considered

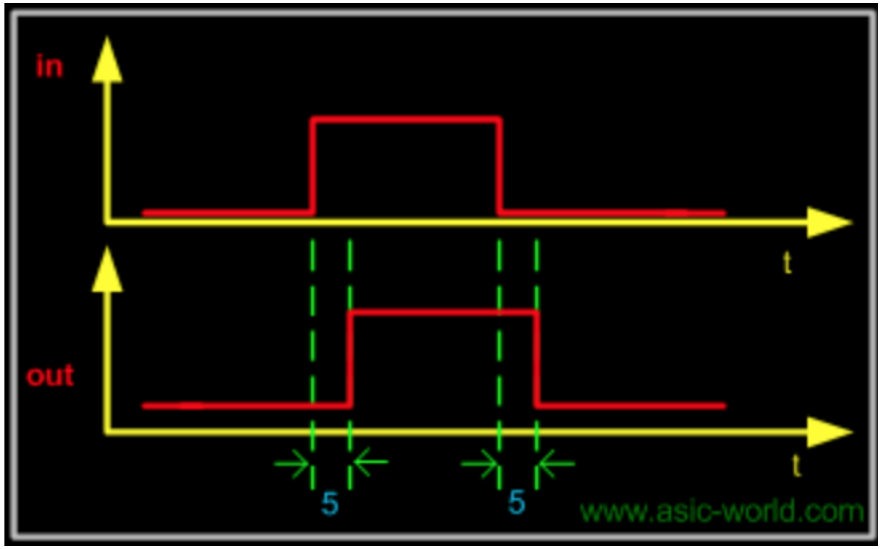
Single Delay

- Delay for all transistors
  - **buf #(delay_time)** a1(out, i)
  - **buf #(5)** a1(out, i)
  - **buf #(*mindelays*:*typdelays*:*maxdelays*)** a1(out, i)
  - **buf #(4:5:6)** a1(out, i)

Single Delay (cont'd)

```
`timescale 1ns/100ps
module buf_gate();
reg        in;
wire       out;
buf #(5)  (out, in)
initial begin
$monitor ("Time = %g in = %b out=%b", $time, in, out);
    in = 0;
    #10 in = 1;
    #10 in = 0;
    #10 $finish;
end
endmodule
```

Time = 0 in = 0
out=x
Time = 5 in = 0
out=0
Time = 10 in = 1
out=0
Time = 15 in = 1



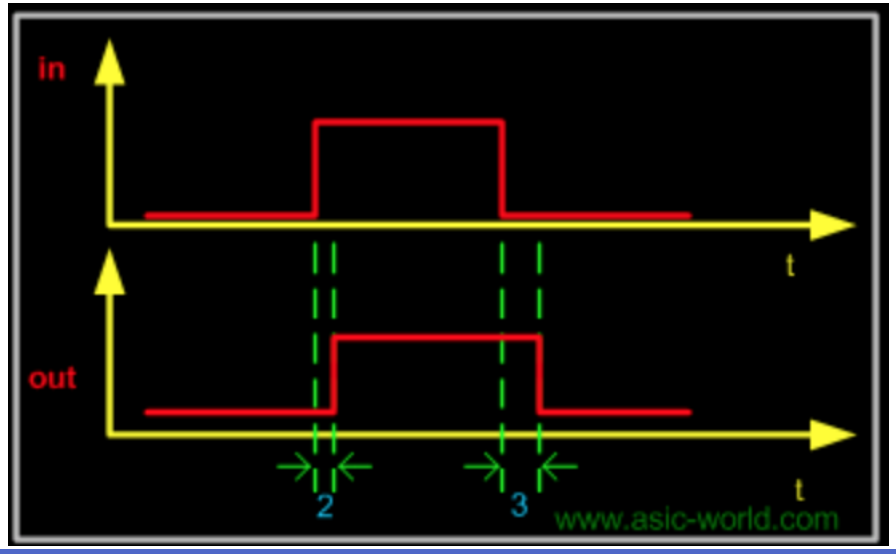
Two Delays

- Delay for rise and fall specifications
  - **and #(rise_val, fall_val)** a1(out, i1, i2)
  - **and #(4, 6)** a1(out, i1, i2)
  - **and #(*mindelays: typdelays: maxdelays, mindelays: typdelays: maxdelays*)** a1(out, i1, i2)
  - **and #(3:4:5, 5:6:7)** a1(out, i1, i2)

Two Delays (cont'd)

```
`timescale 1ns/100ps
module buf_gate();
reg          in;
wire          out;
buf #(2,3)  (out, in);
initial begin
$monitor ("Time = %g in = %b out=%b", $time, in, out);
    in = 0;
    #10 in = 1;
    #10 in = 0;
    #10 $finish;
end
endmodule
```

Time = 0 in = 0 out=x Time = 3 in = 0 out=0 Time = 10 in = 1 out=0 Time = 12 in = 1 out=1 Time = 20 in = 0 out=1 Time = 23 in = 0



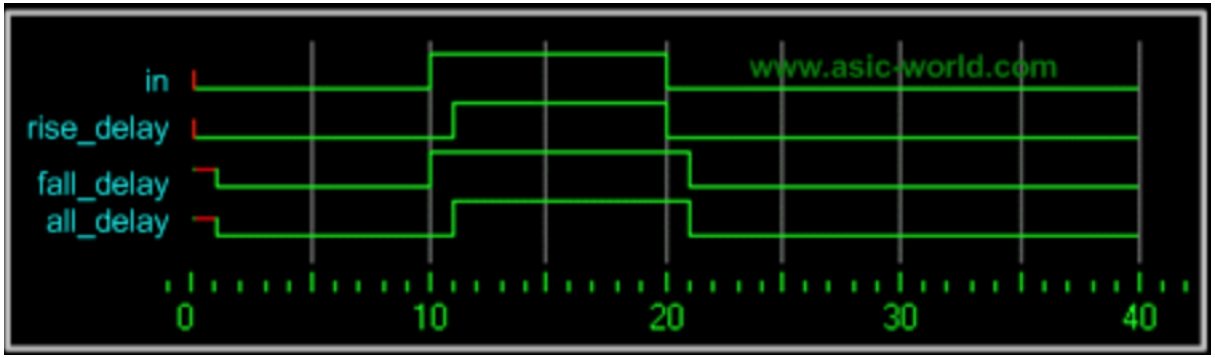
Three Delays

- Delay for rise, fall, and turn-off specifications
  - **bufif0 #(rise_val, fall_val, turnoff_val)** a1(out, in, control)
  - **bufif0 #(3, 4, 5)** a1(out, in, control)
  - **bufif0 #(mindelay: typdelay: maxdelay, .., mindelay: typdelay: maxdelay)** a1(out, i1, i2)
  - **bufif0 #(2:3:4, 3:4:5, 4:5:6)** a1(out, i1, control)

All Delays (cont'd)

```
`timescale 1ns/100ps
module buf_gate();
reg  in;
wire  rise-delay;
wire  fall_delay;
wire  all_delay;

buf #(1,0) U_rise (rise_delay, in);
buf #(0,1) U_fall (fall_delay, in);
buf #1     U_all (all_delay, in);)
initial begin
in = 0;
 #10 in = 1;
    #10 in = 0;
    #20 $finish;
end
endmodule
```

Time = 0 in = 0 rise_delay = 0 fall_delay = x all_delay = x
Time = 1 in = 0 rise_delay = 0 fall_delay = 0 all_delay = 0
Time = 10 in = 1 rise_delay = 0 fall_delay = 1 all_delay = 0
Time = 11 in = 1 rise_delay = 1 fall_delay = 1 all_delay = 1
Time = 20 in = 0 rise_delay = 0 fall_delay = 1 all_delay = 1
Time = 21 in = 0 rise_delay = 0 fall_delay = 0 all_delay = 0



#`num`

- Number of ticks simulator should delay current statement execution
  - **# 1 bufif0 #(3, 4, 5)** a1(out, in, control)

Delay Specification in Verilog (cont'd)

- Method of choosing a min/typ/max value may vary for different simulators.

  Example:

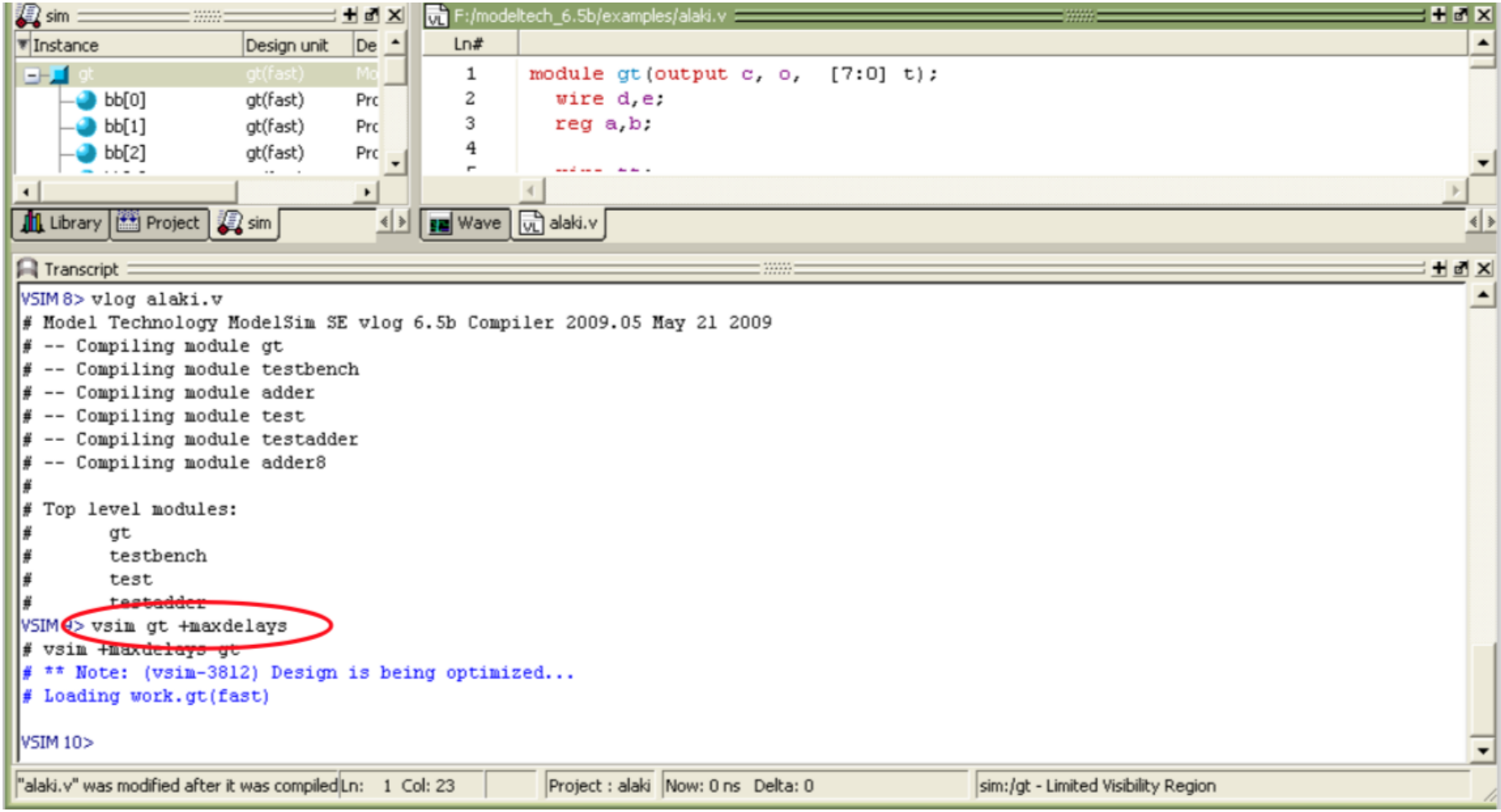
  ModelSim SE 6.0

  Simulate -> Start Simulation: Tab -> Verilog:

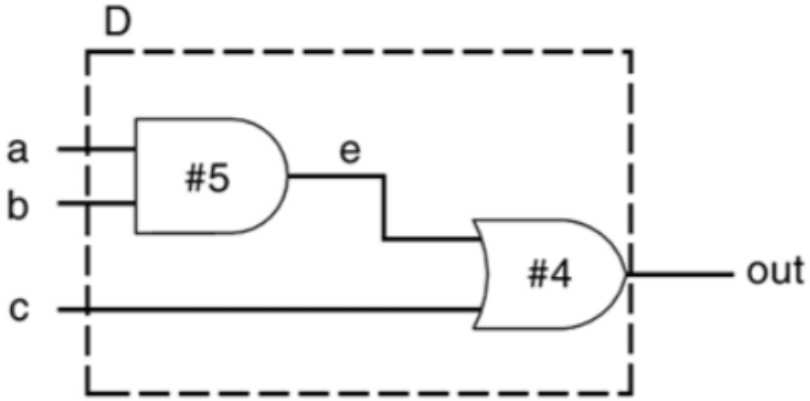
  Box -> delay selection

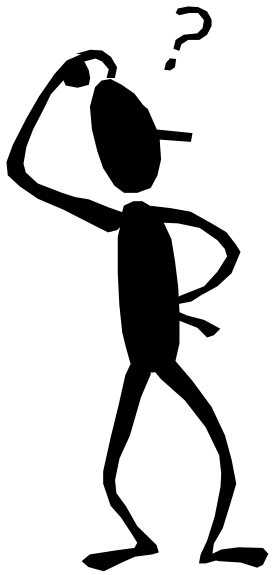
- `` `timescale `` Time Unit/Simulation Precision
- `` `timescale `` 1ns/100ps

Delay Specification in Verilog (cont'd)



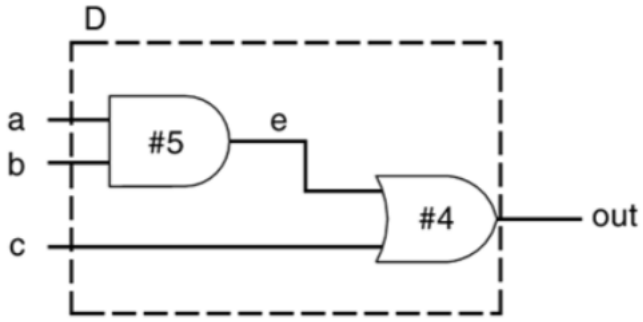
Sample 1





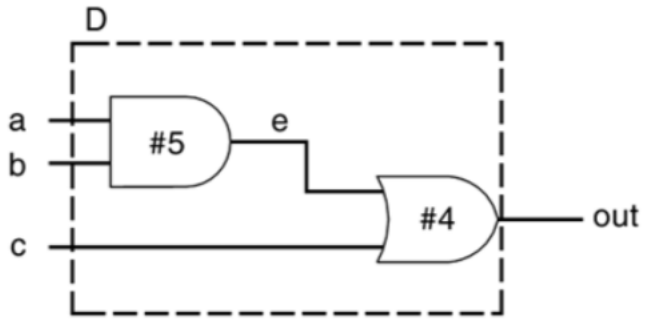
Sample 1: Design Block

```
`timescale 1ns/100ps
module D(out, a, b, c);
  input   a, b, c;
  output  out;
  wire    e;

and #(5)(e, a, b);
or  #(4)(out, e, c);

endmodule
```



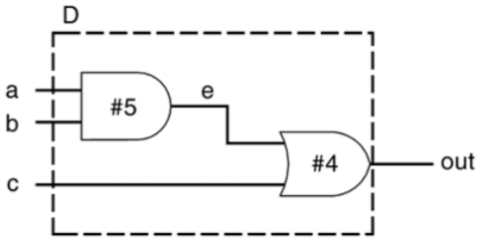
Sample 1: Test Block

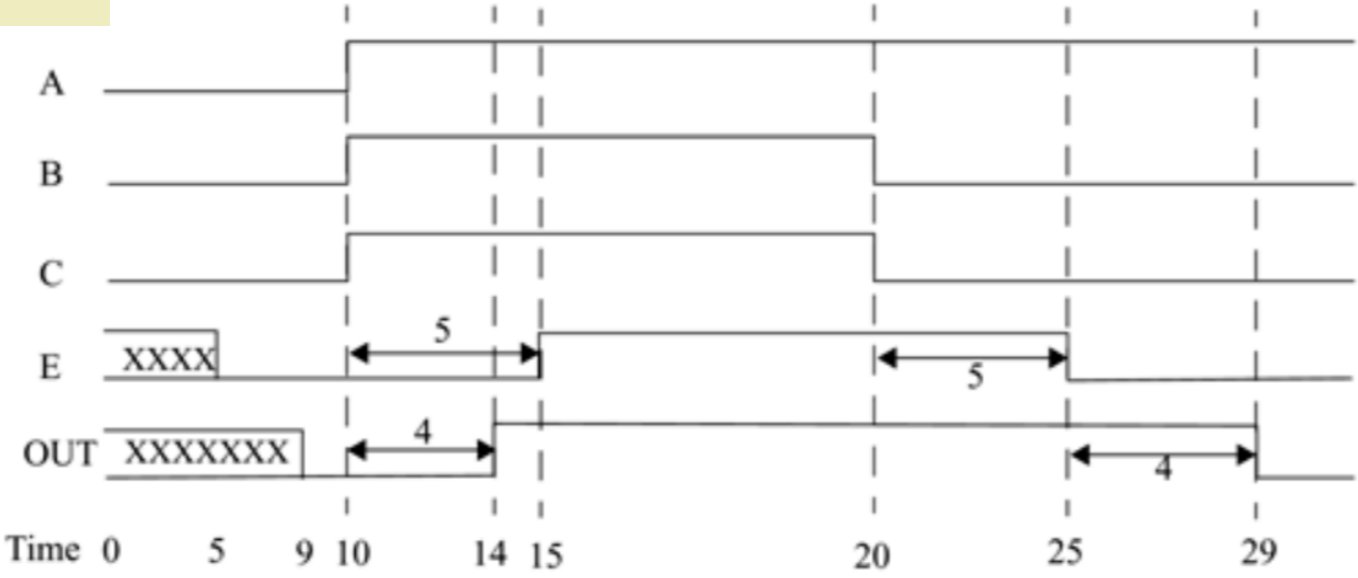
```
module top;
        reg     A, B, C;
        wire    OUT;

        D d1(OUT, A, B, C);

initial
        begin
        A= 1'b0; B= 1'b0; C= 1'b0;
        #10 A= 1'b1; B= 1'b1; C=
1'b1;
        #10 A= 1'b1; B= 1'b0; C=
1'b0;
        #20 $finish;
endmodule
```



Sample 1: Timing Analysis

```
module top;
...
A= 1'b0;  B= 1'b0; C= 1'b0;
 #10  A= 1'b1;  B= 1'b1; C= 1'b1;
 #10  A= 1'b1;  B= 1'b0; C= 1'b0;
 #20  $finish;
...
endmodule
```





Thank You