

# Computer-Aided VLSI System Design

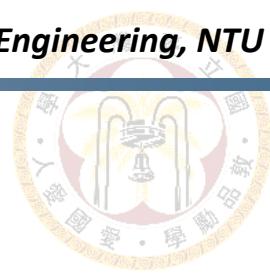
## Chapter 6. Static Timing Analysis (STA)

Lecturer: Chen-Lin Chuang

*Graduate Institute of Electronics Engineering, National Taiwan University*

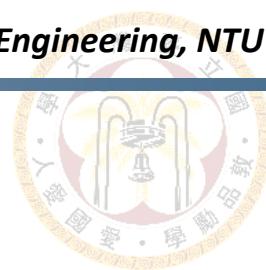


NTU GIEE



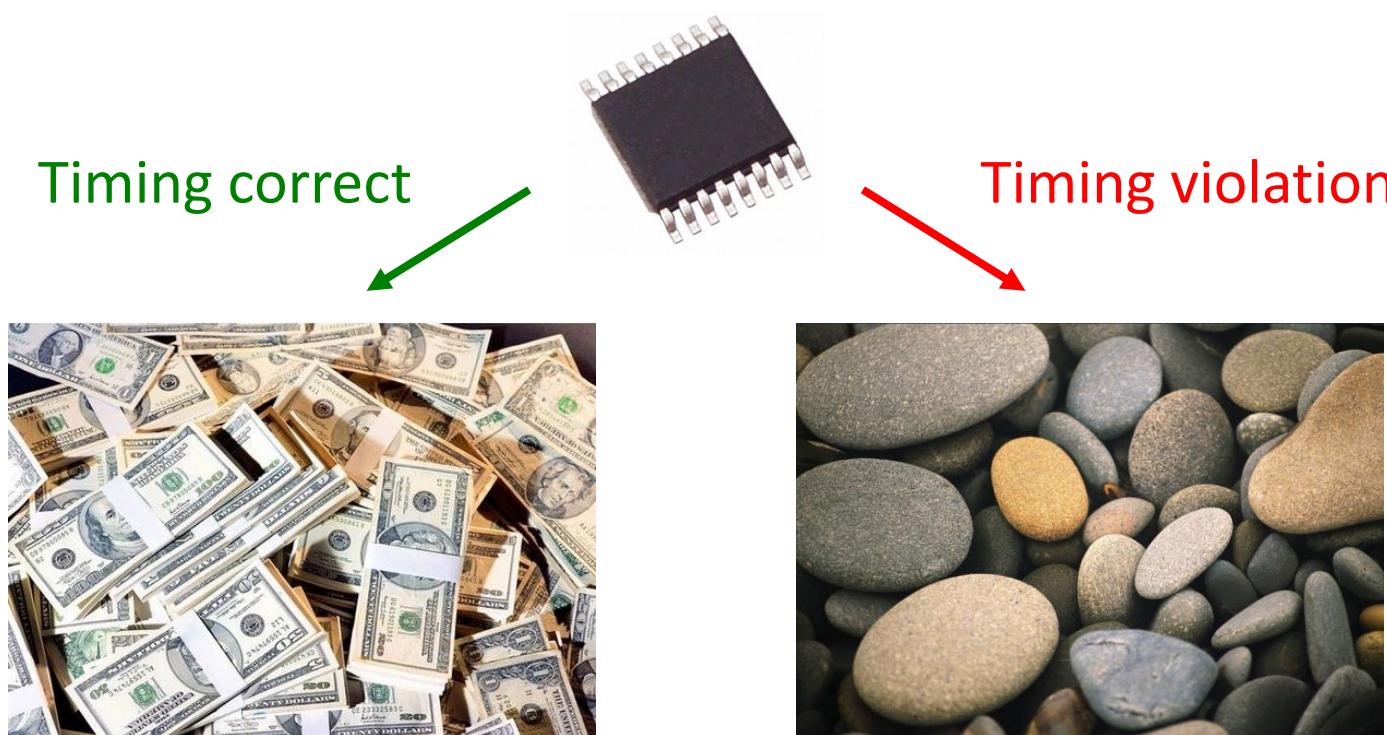
# Outline

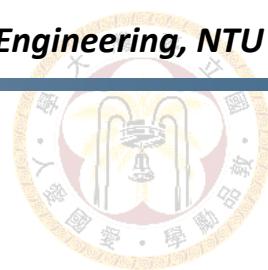
- **Introduction**
- **Static Timing Analysis**
- **Environment Setting**
- **Special Timing Paths**
- **Clock Domain Crossing**



# Timing Analysis

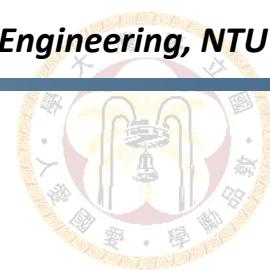
- Timing analysis is an integral part of ASIC/VLSI design flow
  - Especially for synchronous design
- The design will not work if it fails the timing analysis





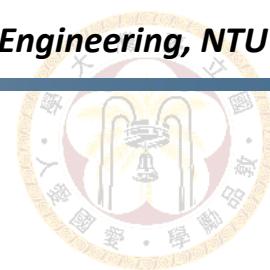
# The Goal of Timing Analysis

- **To assure all signals will arrive neither too early nor too late**
  - **Too early:** hold violation
  - **Too late:** setup violation
- **If there are timing violations**
  1. Check the simulation environment and settings
  2. Modify the settings of synthesis tools
  3. Fix your design architecture



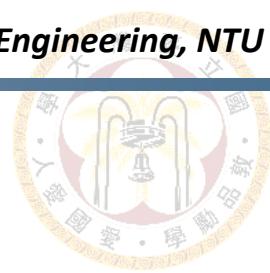
# Dynamic Timing Analysis (DTA)

- **Also called Testbench Simulation**
  - Test vectors are applied during the simulation time
  - Simulator calculates both the logic values and delays
- **Disadvantages**
  - Virtually impossible to do exhaustive analysis
    - Vector creation takes too long
    - Incomplete timing coverage
  - Hard to identify the cause of failure because the function and timing are analyzed at the same time
  - Requires more memory and CPU resources
    - Huge analysis space
    - Long simulation time

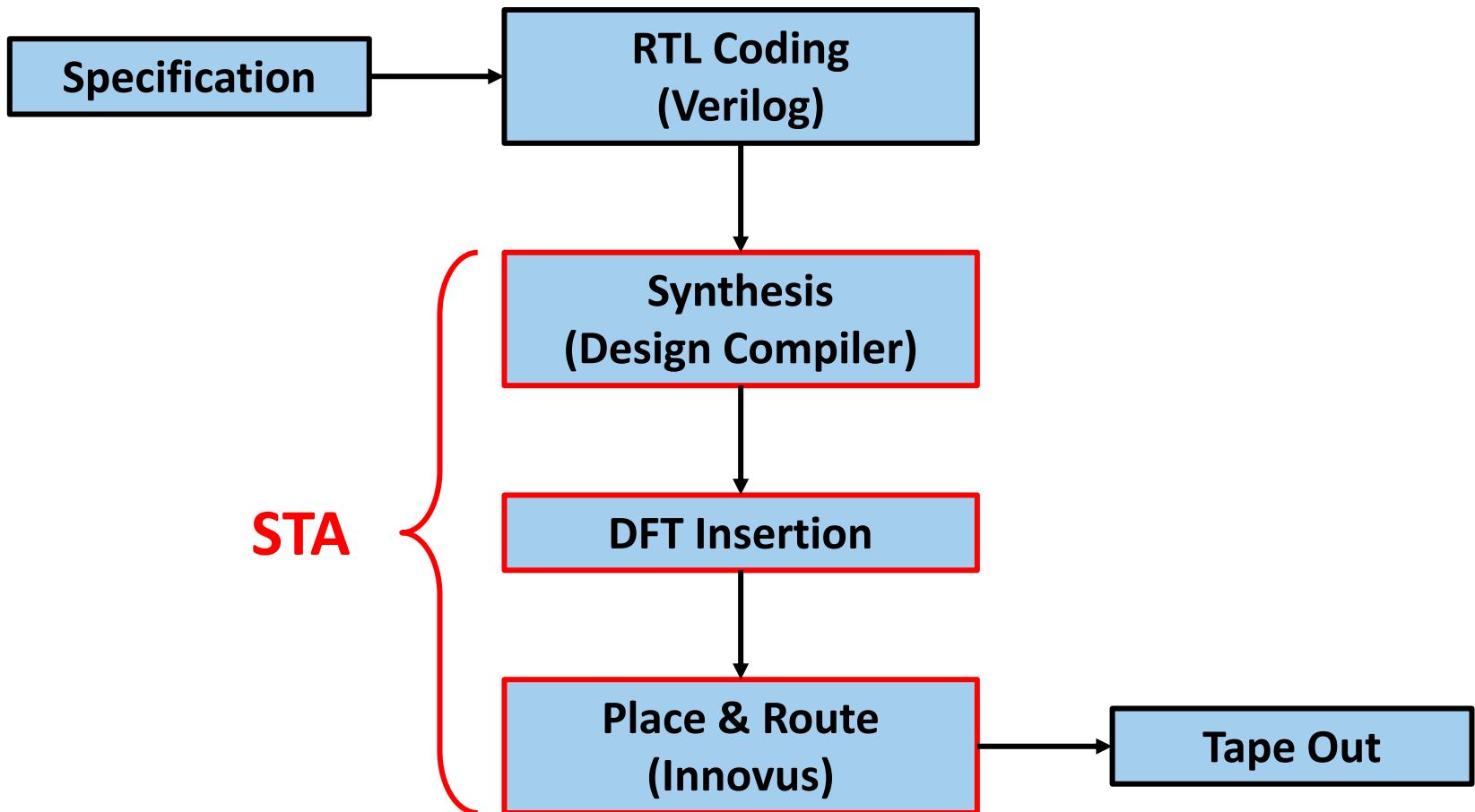


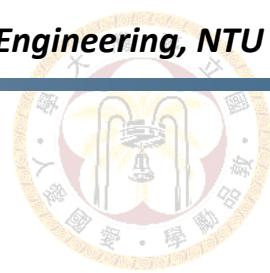
# Static Timing Analysis (STA)

- An **exhaustive** verification of all timing checks
- Analyzes every timing path according to the given constraints
- No need for circuit simulation
  - Less memory and CPU resources than DTA
- Disadvantages
  - For synchronous logic only
  - Tricky constraints beyond the boundaries of single clock flip-flop design chips:
    - False paths
    - Multicycle paths
    - Multiple clock domains



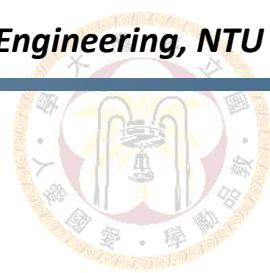
# When to Perform STA





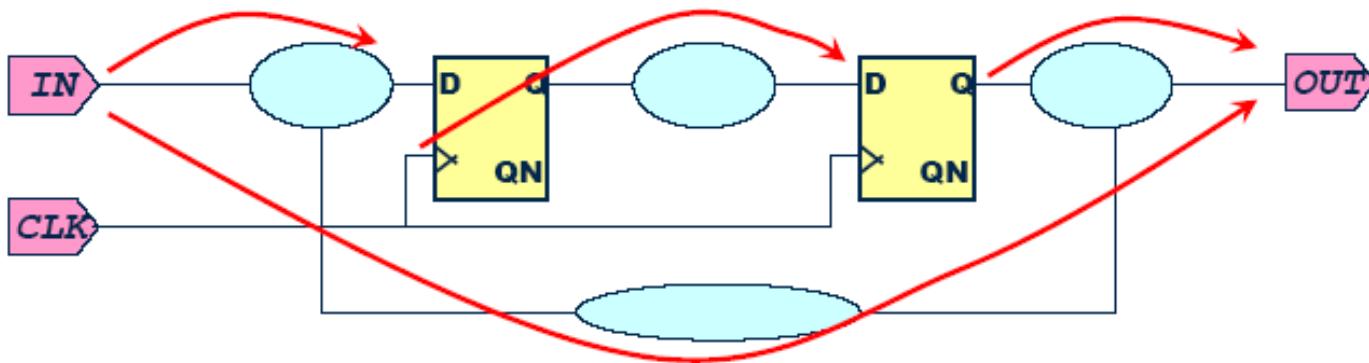
# Outline

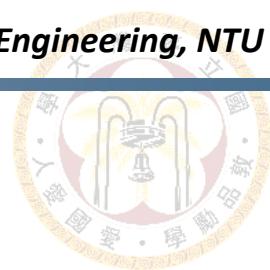
- Introduction
- **Static Timing Analysis**
- Environment Setting
- Special Timing Paths
- Clock Domain Crossing



# Steps of STA

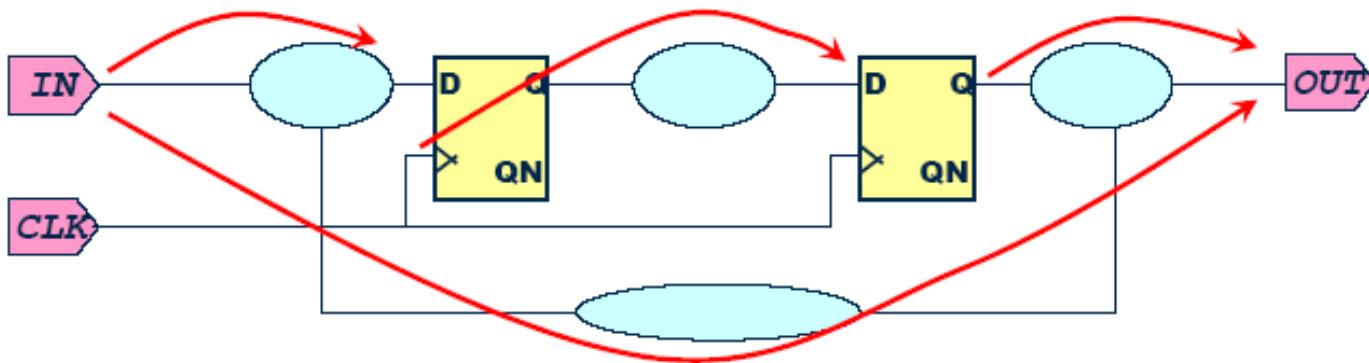
1. Break the design into a set of timing paths
2. Calculate arrival time and required time of each path
3. Calculate the slack of each path

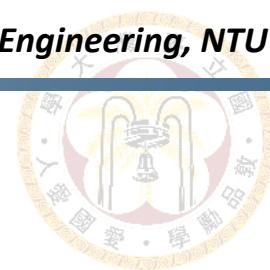




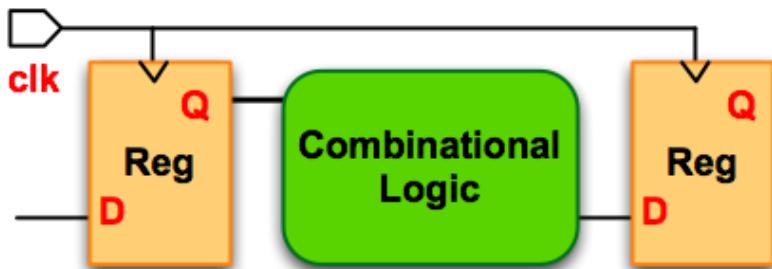
# Steps of STA

1. Break the design into a set of timing paths
2. Calculate arrival time and required time of each path
3. Calculate the slack of each path





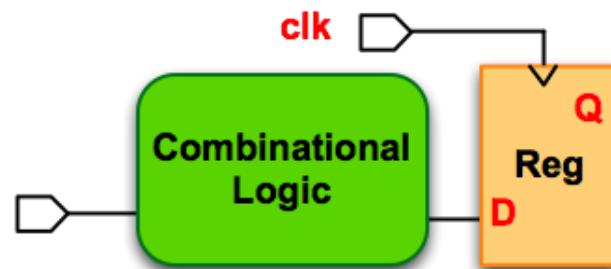
# Four Types of Timing Paths



**Register-to-register (reg2reg)**

**Start Point:** **clock pin** of sequential device

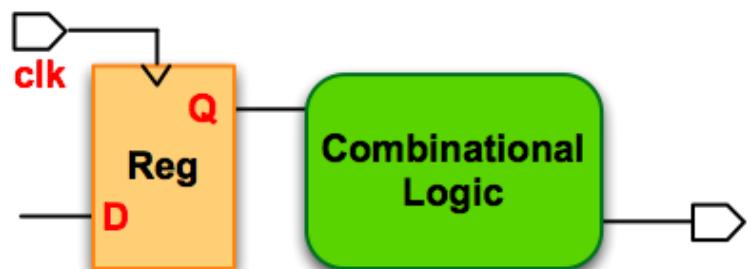
**End point:** data input pin of sequential devices



**Input-to-register (in2reg)**

**Start Point:** primary input port

**End point:** data input pin of sequential devices



**Register-to-output (reg2out)**

**Start Point:** **clock pin** of sequential device

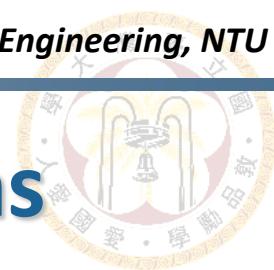
**End point:** primary output port



**Input-to-output (in2out)**

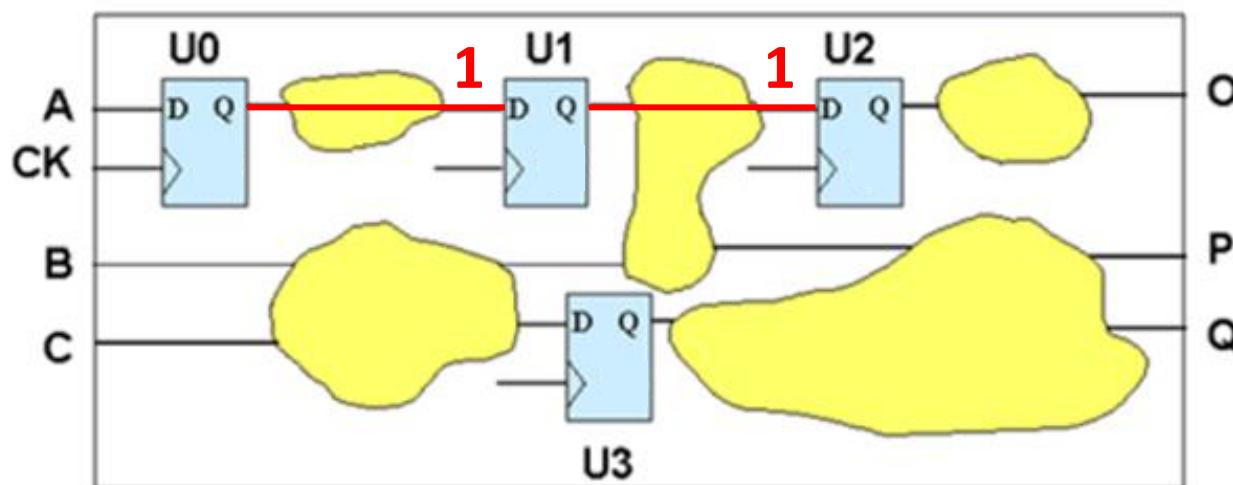
**Start Point:** primary input port

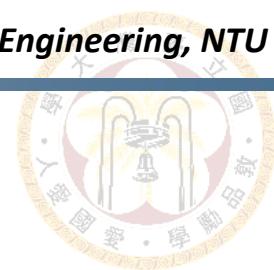
**End point:** primary output port



# Example: Register-to-Register Paths

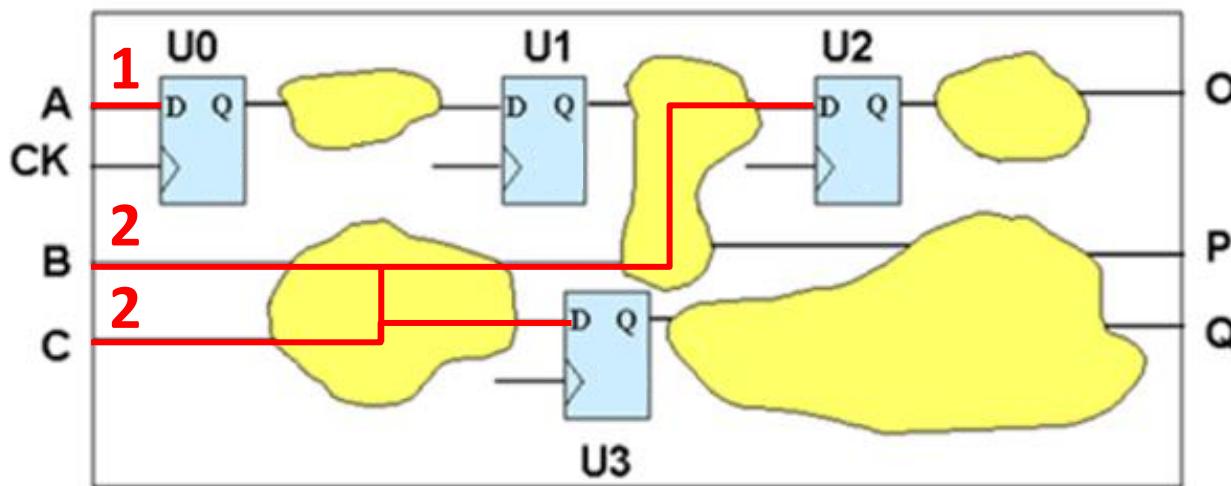
- How many register-to-register paths are there in this design?
  - 2

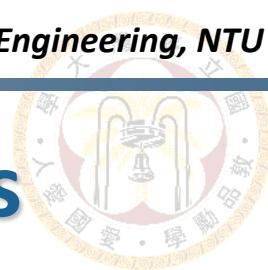




# Example: Input-to-Register Paths

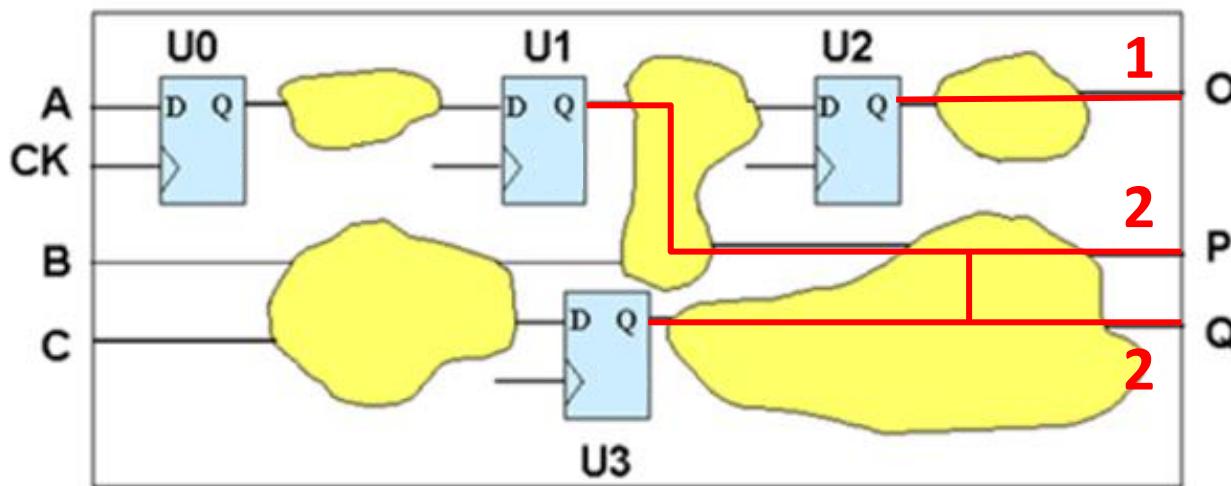
- How many input-to-register paths are there in this design?
  - 5

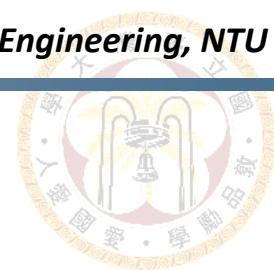




# Example: Register-to-Output Paths

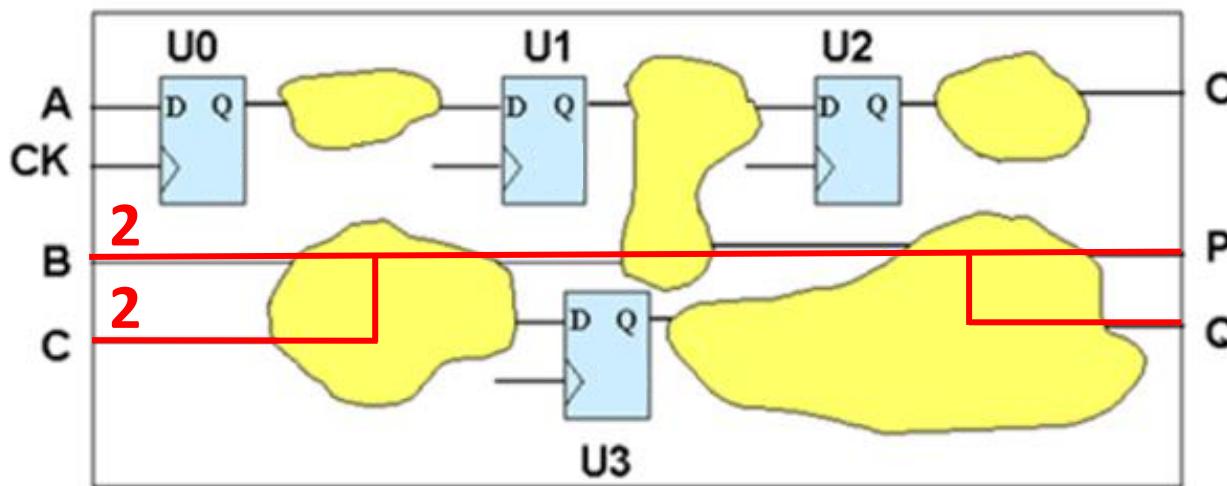
- How many register-to-output paths are there in this design?
  - 5

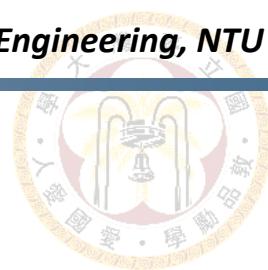




# Example: Input-to-Output Paths

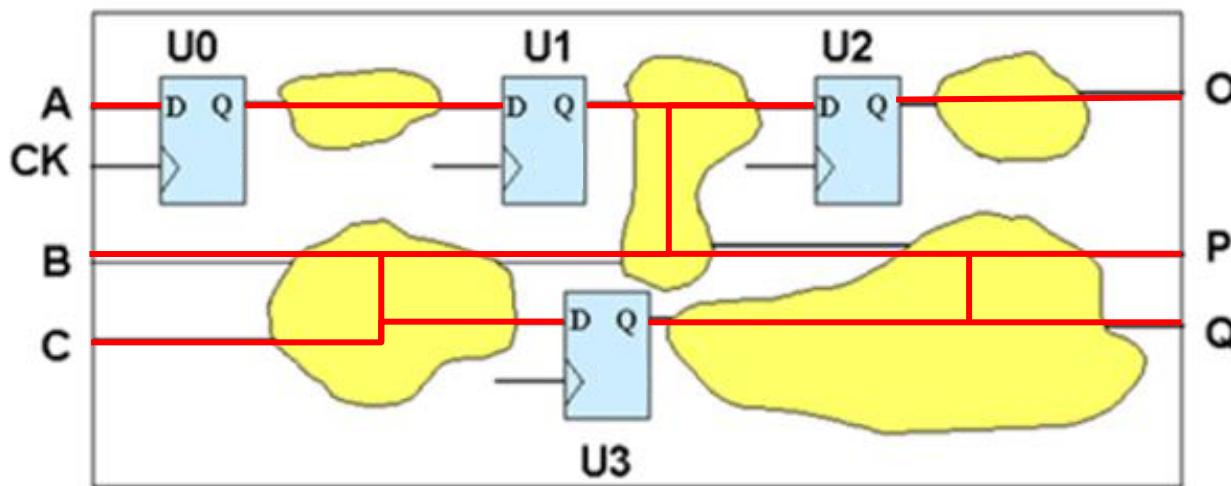
- How many input-to-output paths are there in this design?
  - 4

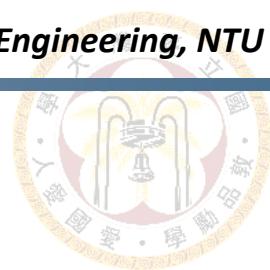




# Example: Timing Paths

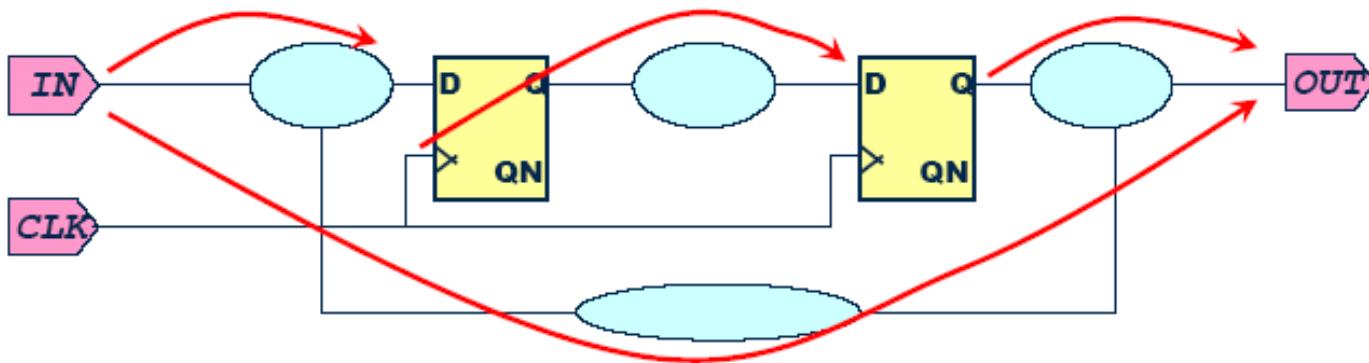
- How many timing paths are there in this design?
  - $2 + 5 + 5 + 4 = 16$

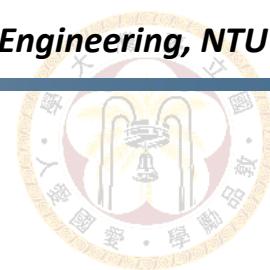




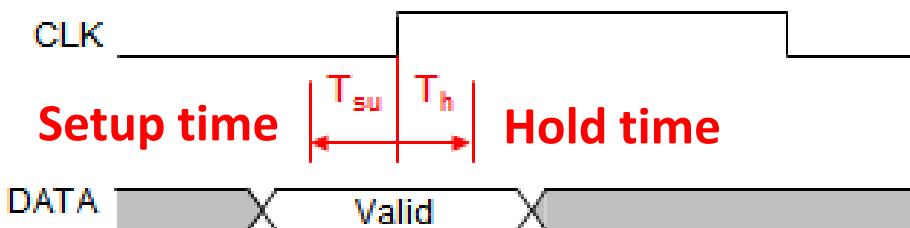
# Steps of STA

1. Break the design into a set of timing paths
2. Calculate arrival time and required time of each path
3. Calculate the slack of each path





# Setup Time & Hold Time

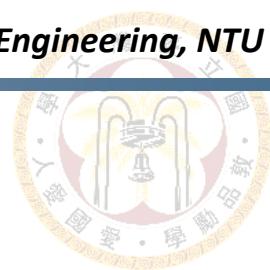


## ■ Setup Time $T_{su}$

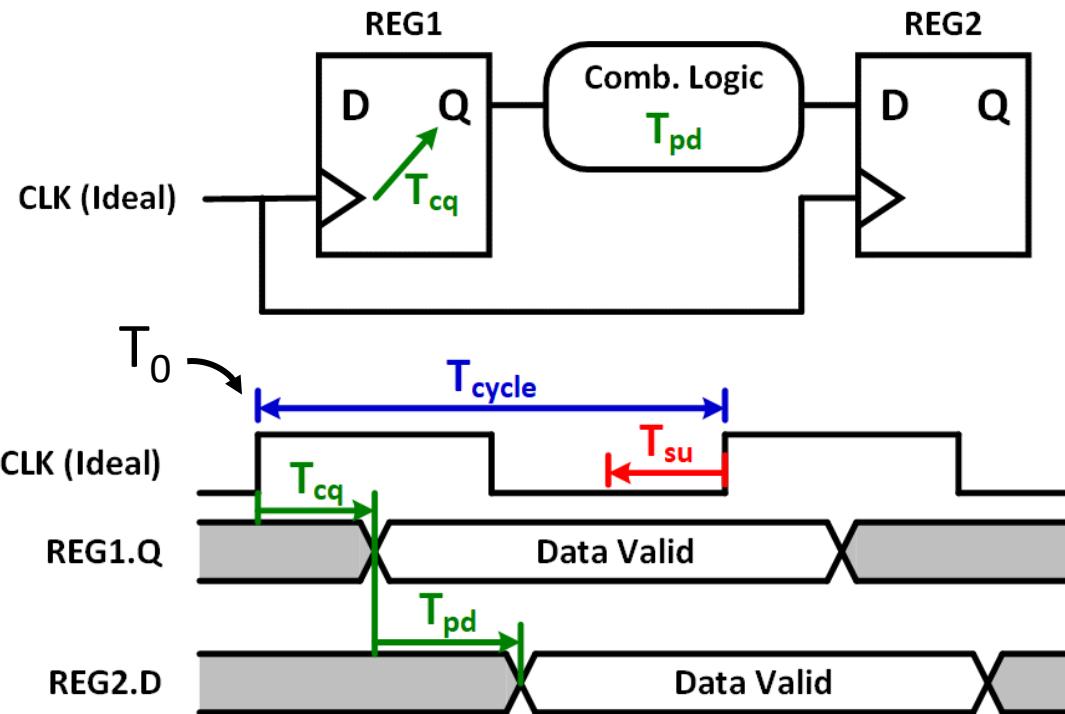
- The minimum amount of time before the clock's active edge that the data must be stable
- Violation may cause incorrect data to be captured

## ■ Hold Time $T_h$

- The minimum amount of time after the clock's active edge during which data must be stable
- Violation may cause incorrect data to be latched

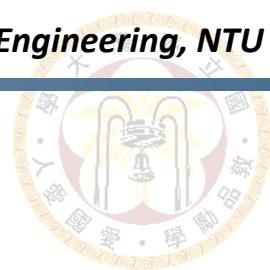


# Setup Check (Ideal CLK)

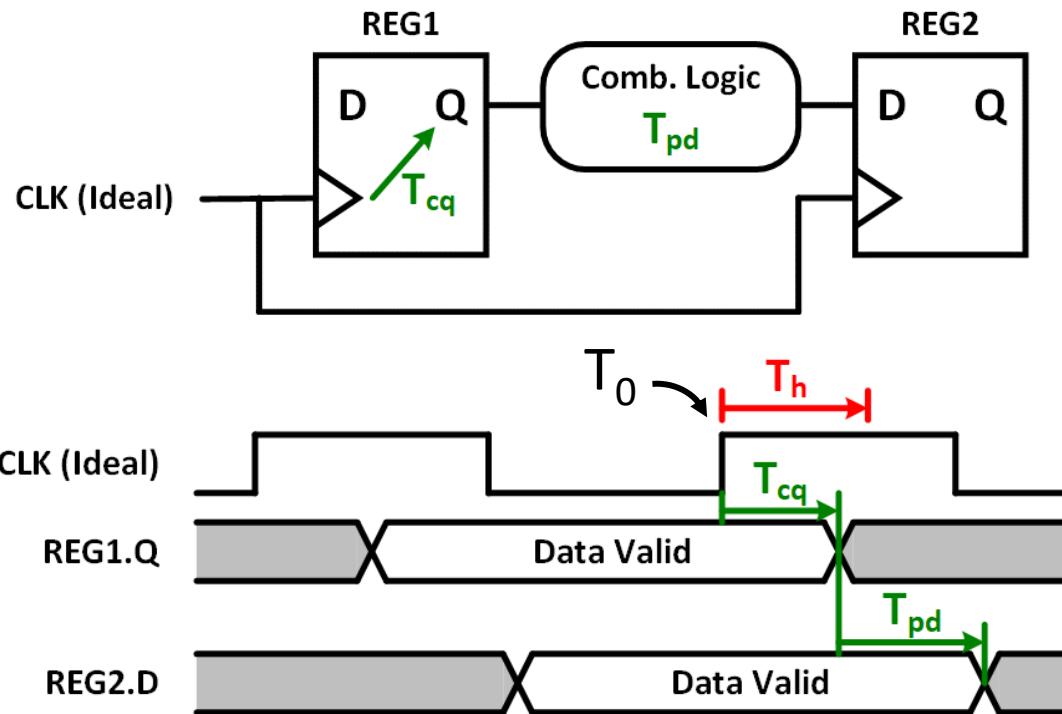


Arrival time (AT):  $T_0 + T_{cq} + T_{pd}$

Required time (RT):  $T_0 + T_{cycle} - T_{su}$

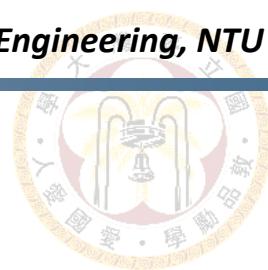


# Hold Check (Ideal CLK)



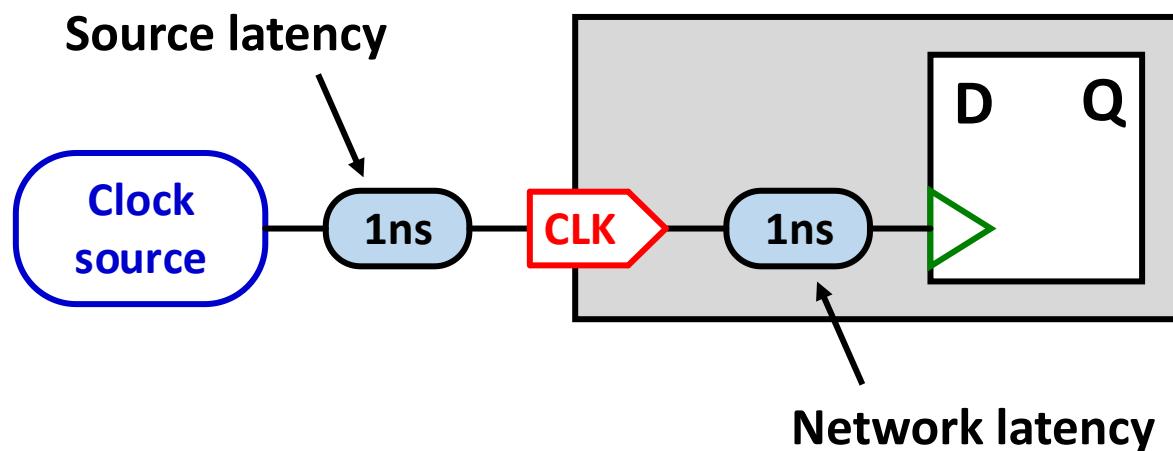
Arrival time (AT):  $T_0 + T_{cq} + T_{pd}$

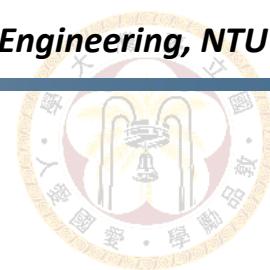
Required time (RT):  $T_0 + T_h$



# Clock Latency

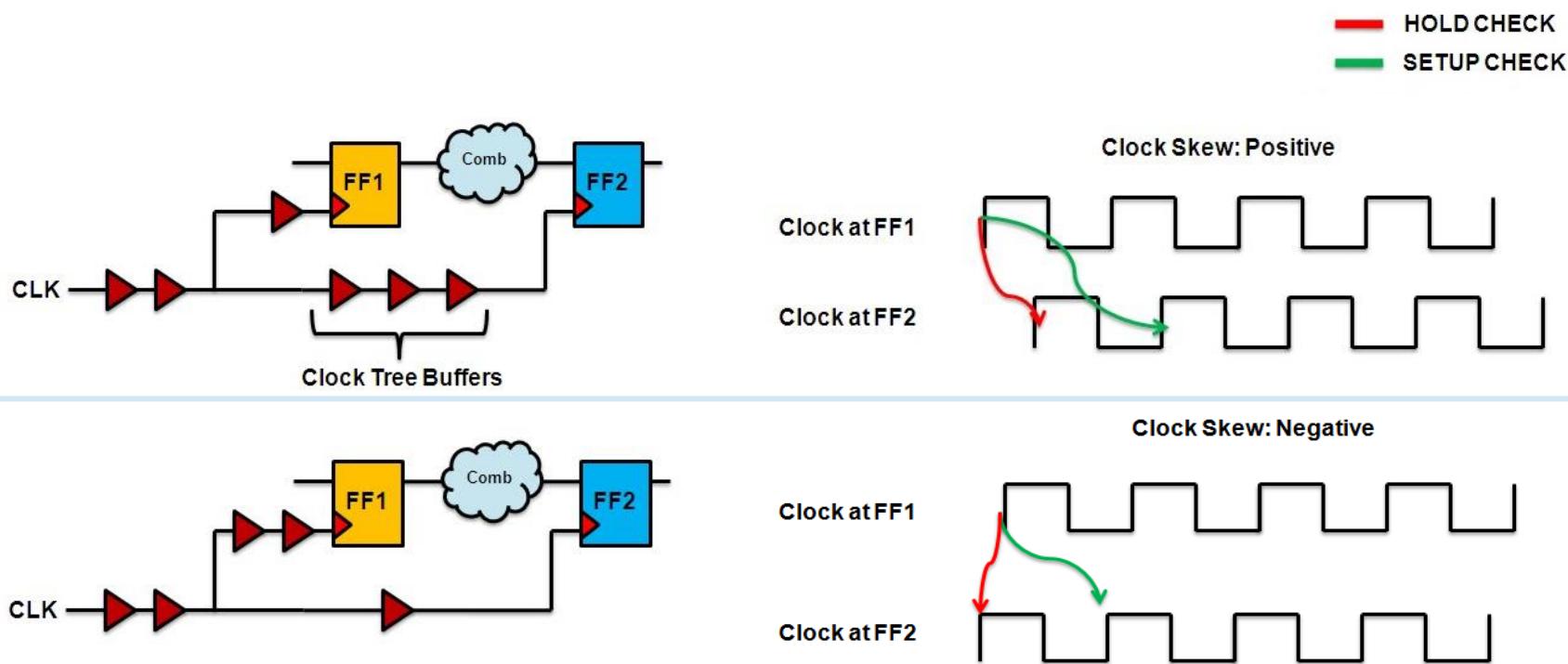
- **Source latency**
  - Delay from the **clock source** to the **clock definition point (CLK)**
- **Network latency**
  - Delay from the **clock definition point (CLK)** to the **clock pin of a sequential cell**

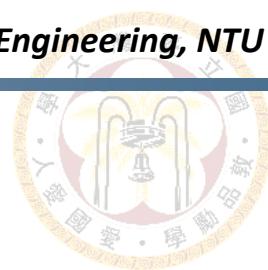




# Clock Uncertainty: Skew

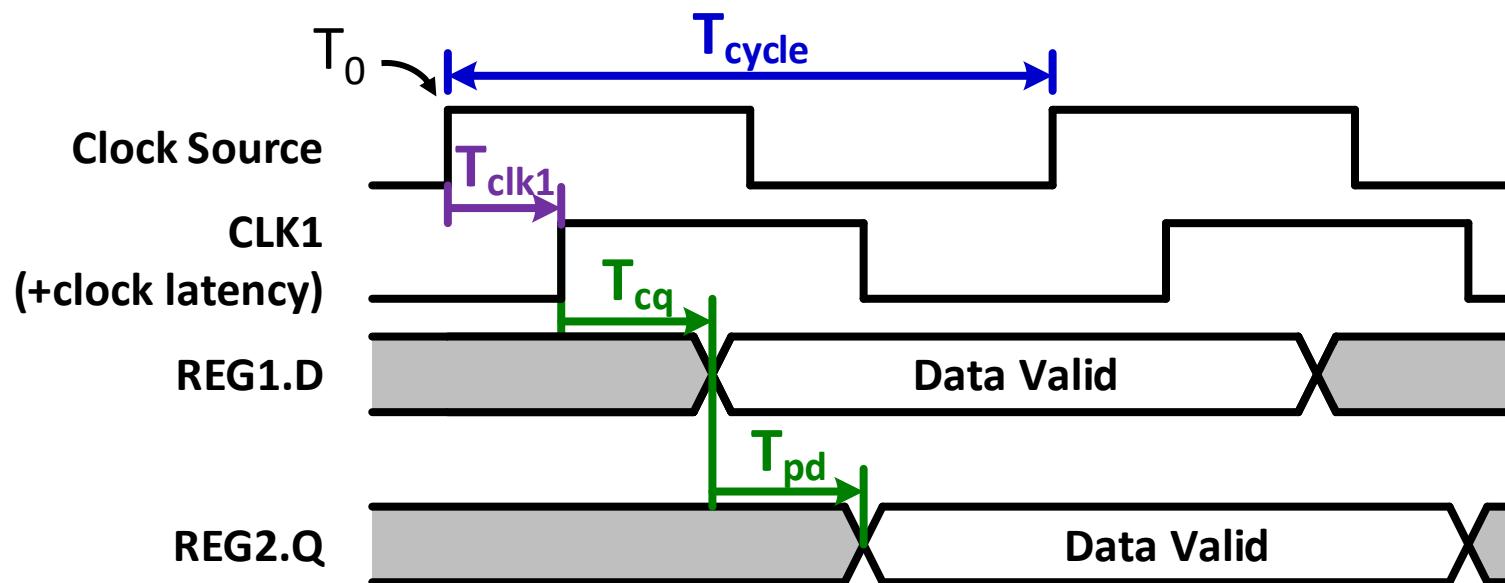
- The difference in arrival time of a clock transition on an integrated circuit



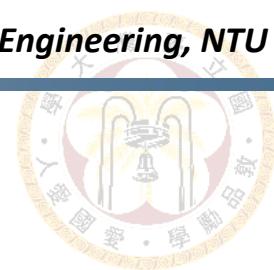


# Setup Check: Arrival Time

- Consider the same circuit in p. 19

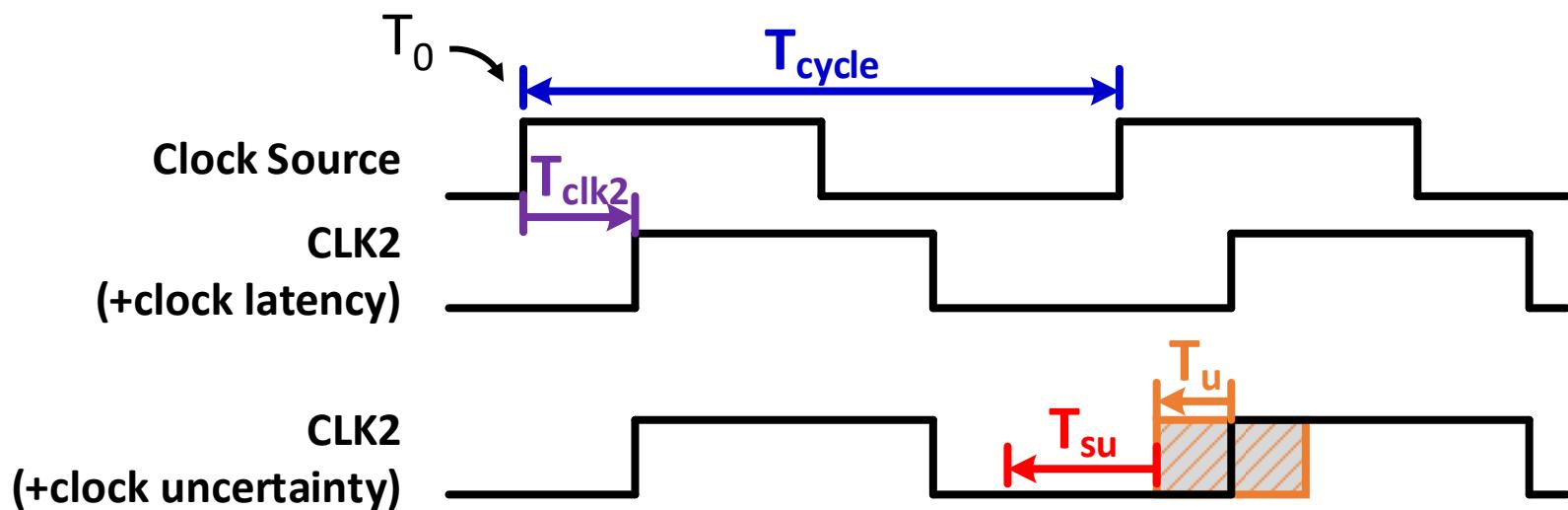


$$\text{Arrival time (AT)}: T_0 + T_{clk1} + T_{cq} + T_{pd}$$

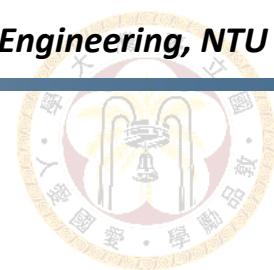


# Setup Check: Required Time

- Consider the same circuit in p. 19

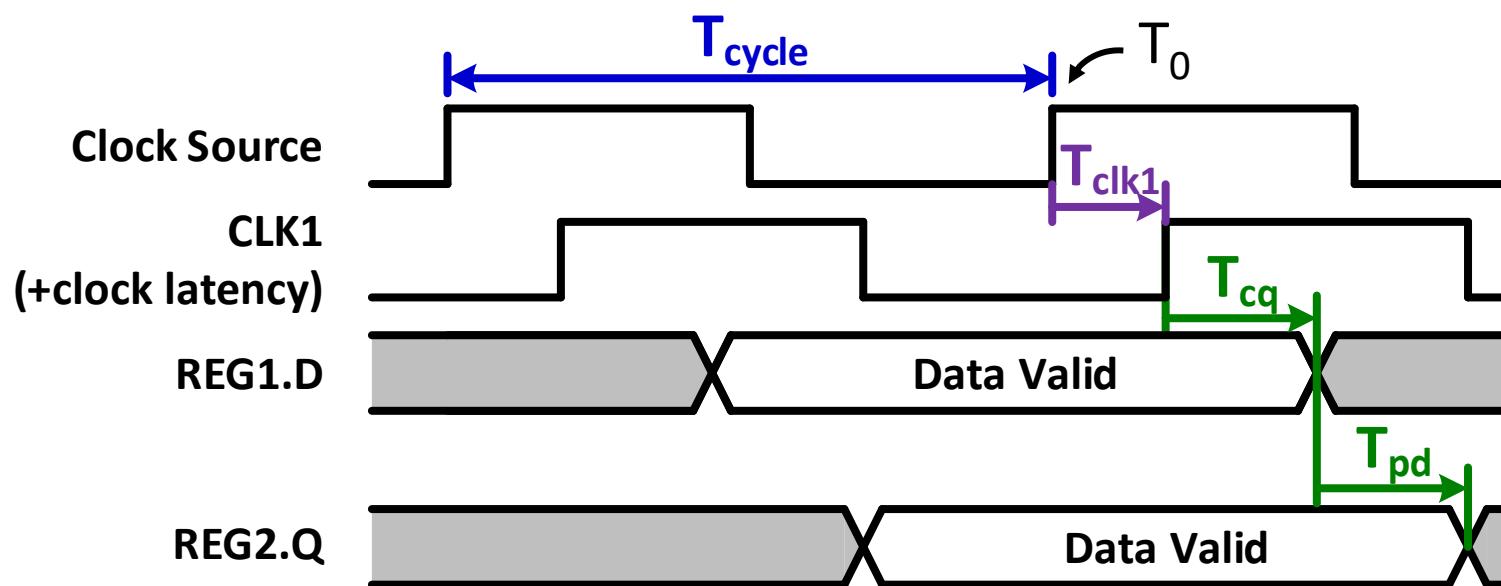


Required time (RT):  $T_0 + T_{clk2} + T_{cycle} - T_u - T_{su}$

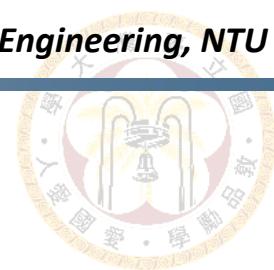


# Hold Check: Arrival Time

- Consider the same circuit in p. 20

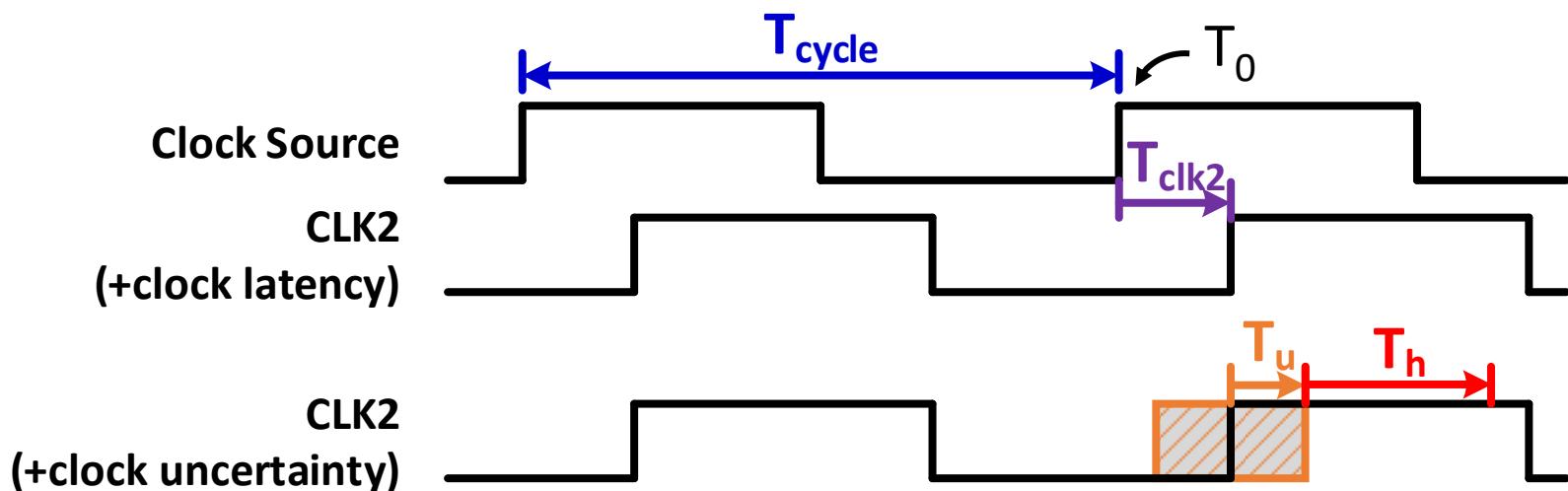


$$\text{Arrival time (AT)}: T_0 + T_{clk1} + T_{cq} + T_{pd}$$

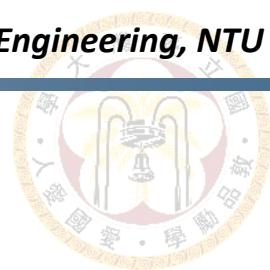


# Hold Check: Required Time

- Consider the same circuit in p. 20

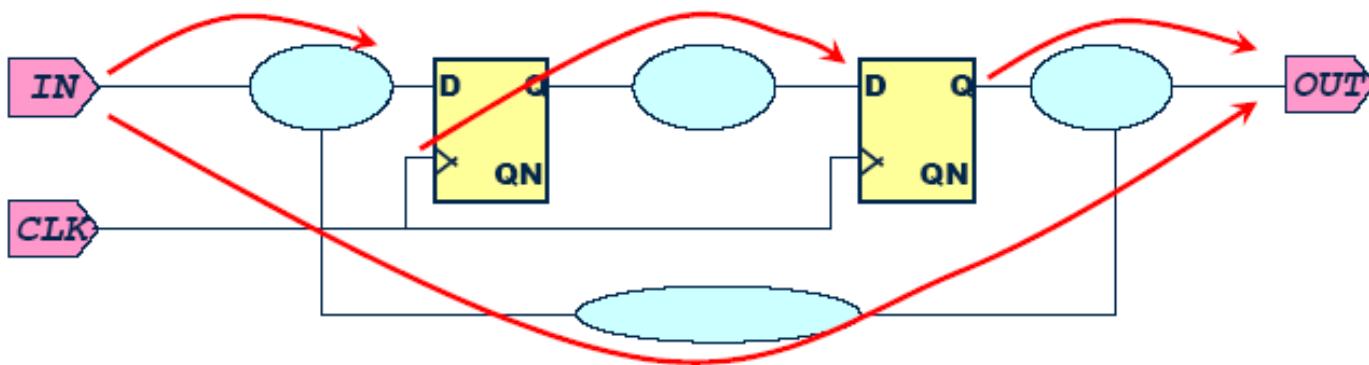


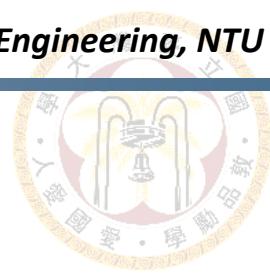
Required time (RT):  $T_0 + T_{clk2} + T_u + T_h$



# Steps of STA

1. Break the design into a set of timing paths
2. Calculate arrival time and required time of each path
3. Calculate the slack of each path





# Slack

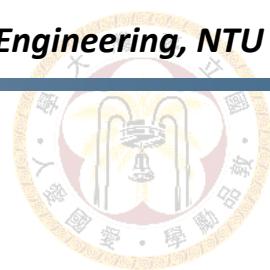
- **Setup: data should arrive before required time**

$$\text{Setup slack} = \text{RT} - \text{AT}$$

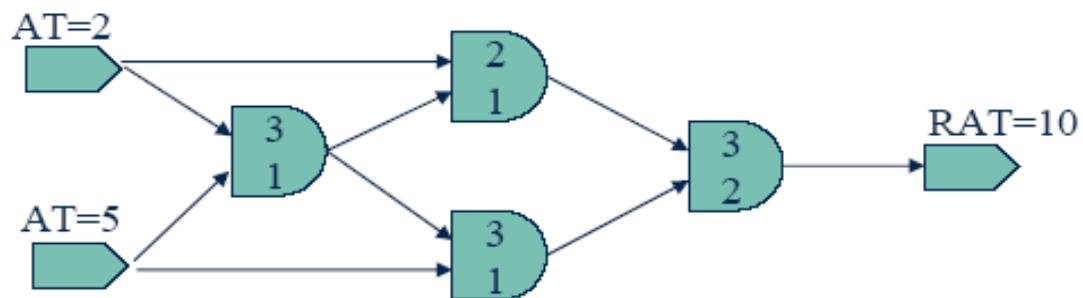
- **Hold: data should be held until required time**

$$\text{Hold slack} = \text{AT} - \text{RT}$$

- **Slack  $\geq 0 \rightarrow$  timing constraints met**
- **Slack  $< 0 \rightarrow$  timing constraints violated**

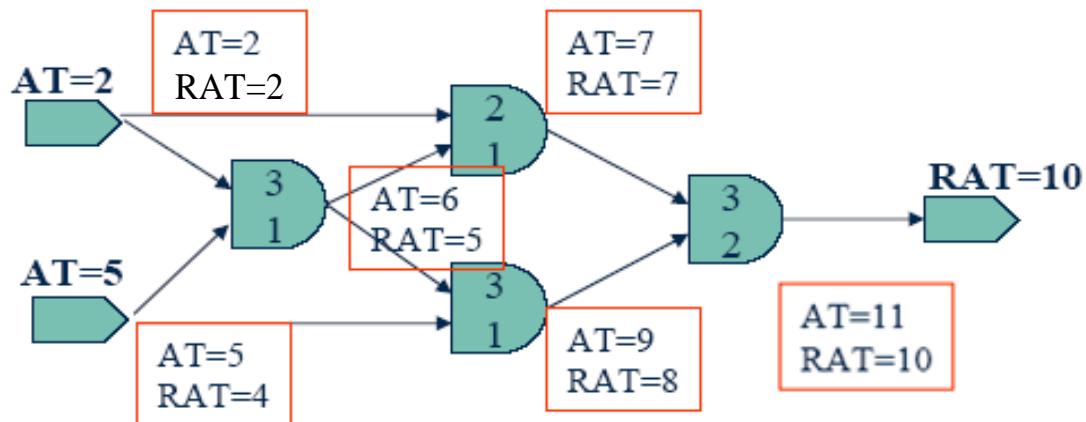


# Path-based vs. Block-based STA



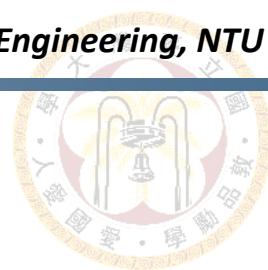
## Path-based:

- $2+2+3 = 7$  (OK)
- $2+3+1+3 = 9$  (OK)
- $2+3+3+2 = 10$  (OK)
- $5+1+1+3 = 10$  (OK)
- $5+1+3+2 = 11$  (Fail)
- $5+1+2 = 8$  (OK)

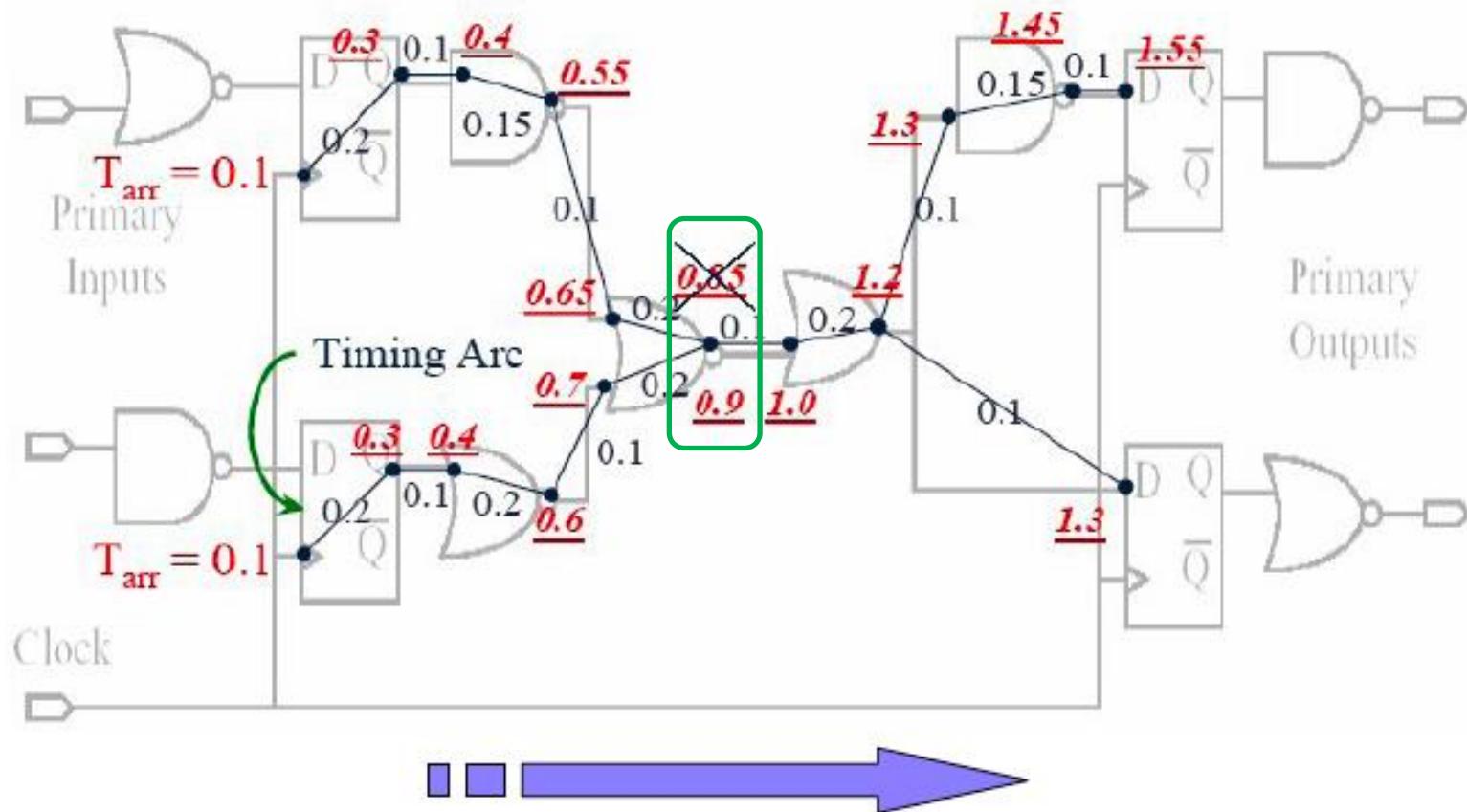


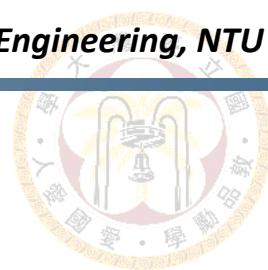
## Block-based:

Critical path is determined as collection of gates with the same, negative slack:  
In our case, we see one critical path with slack = -1

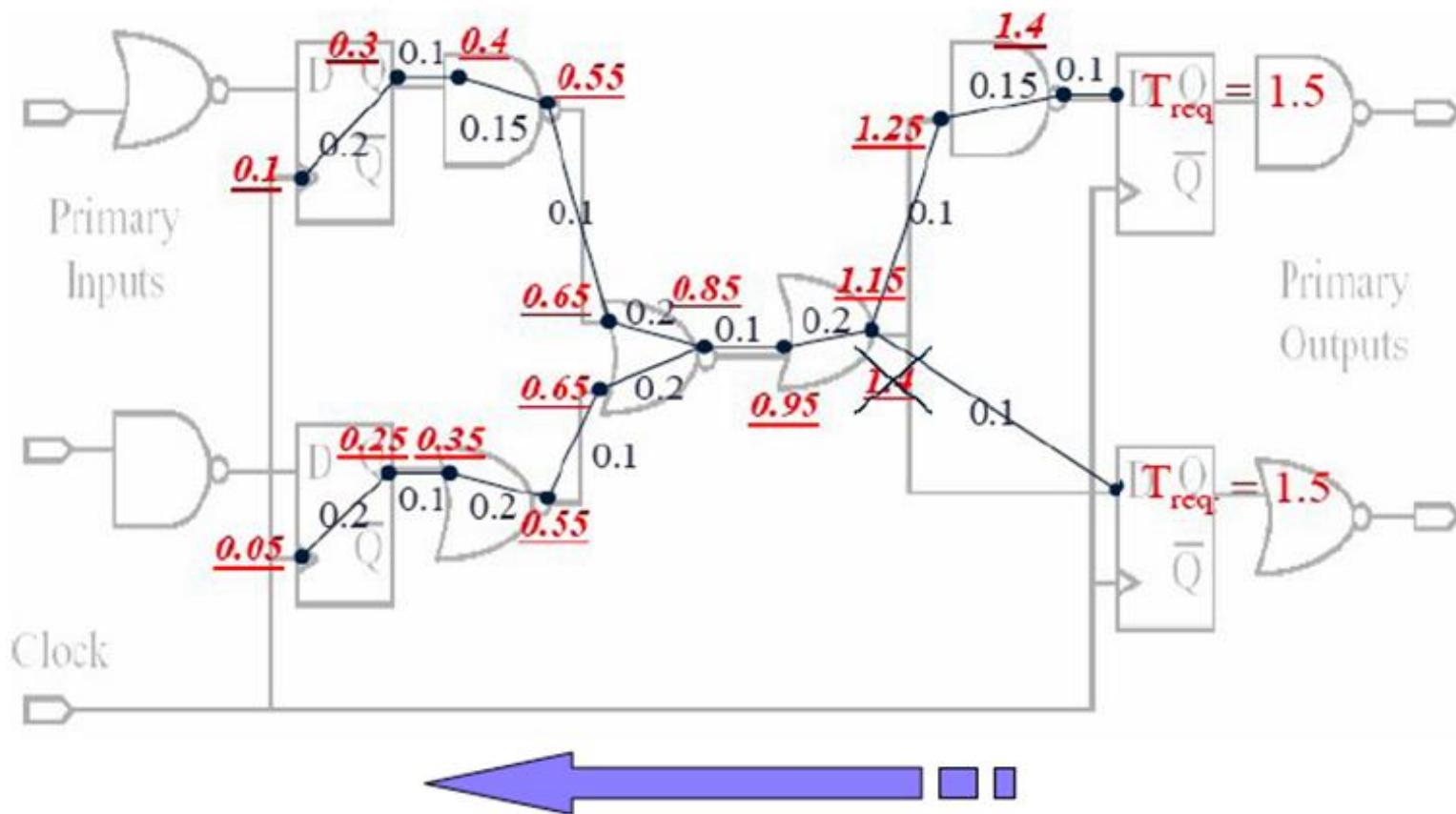


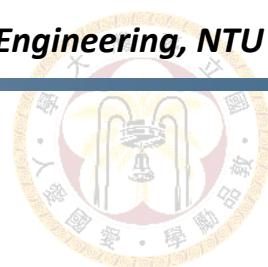
# Block-based STA: Arrival Time



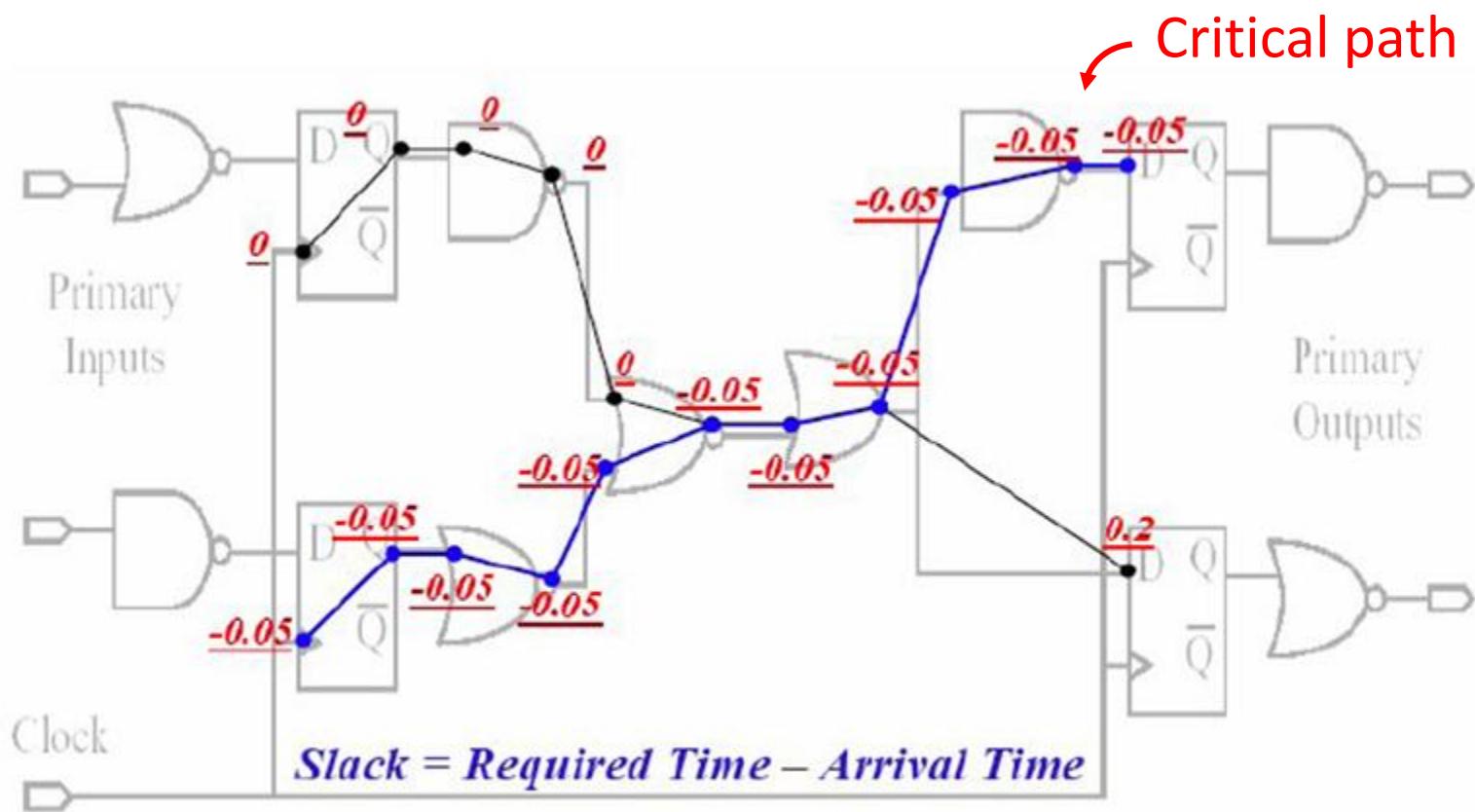


# Block-based STA: Required Time

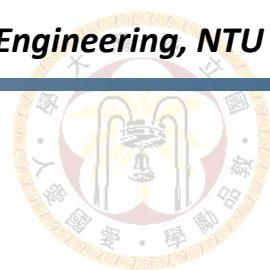




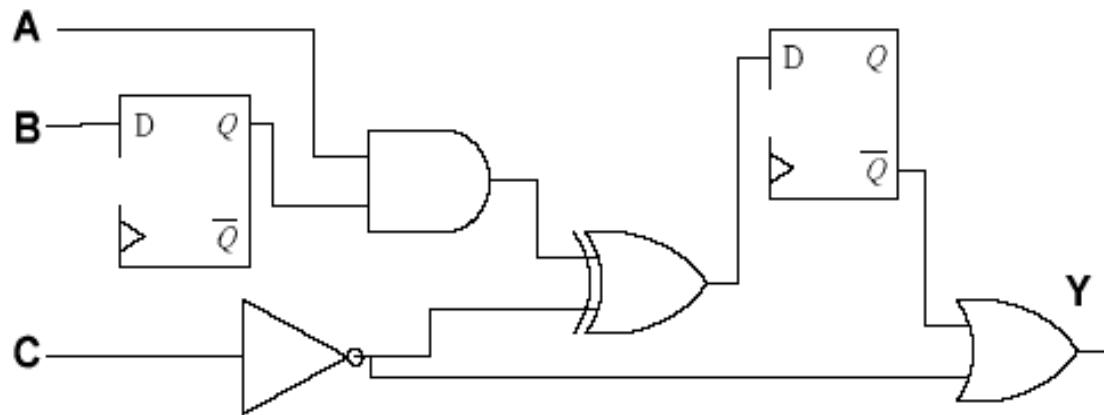
# Setup Slack Graph



Slack  $\geq 0 \rightarrow$  timing constraints met  
Slack  $< 0 \rightarrow$  timing constraints violated

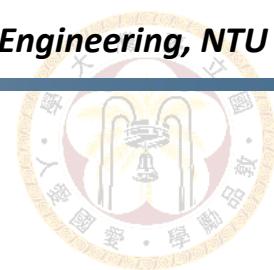


# STA Example



Clock to Q $T_{cq}$	Setup time $T_{su}$	Hold time $T_h$
3ns	1ns	1ns

	Logic gate	Net
Max delay	3ns	2ns
Min delay	1ns	1ns



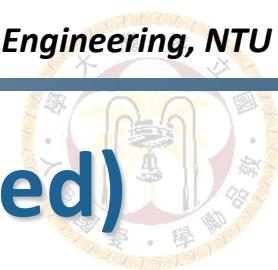
# STA Example: Timing Constraints

- Clock definition

Clock period $T_{cycle}$	Clock Latency $T_{clk}$	Clock Uncertainty $T_u$
14ns	5ns	1ns

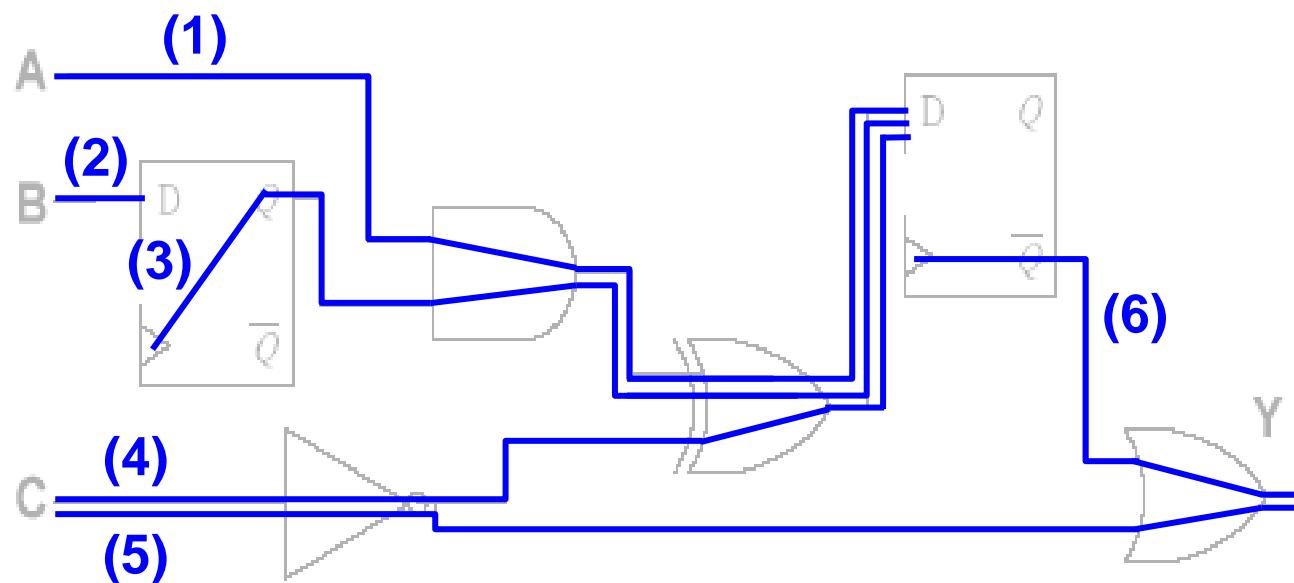
- IO constraints

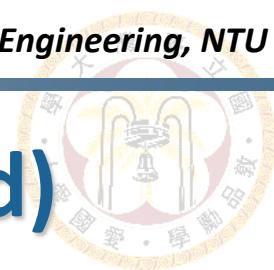
	$D_A$	$D_B$	$D_C$	$D_Y$
Delay	1ns	1ns	1ns	3ns



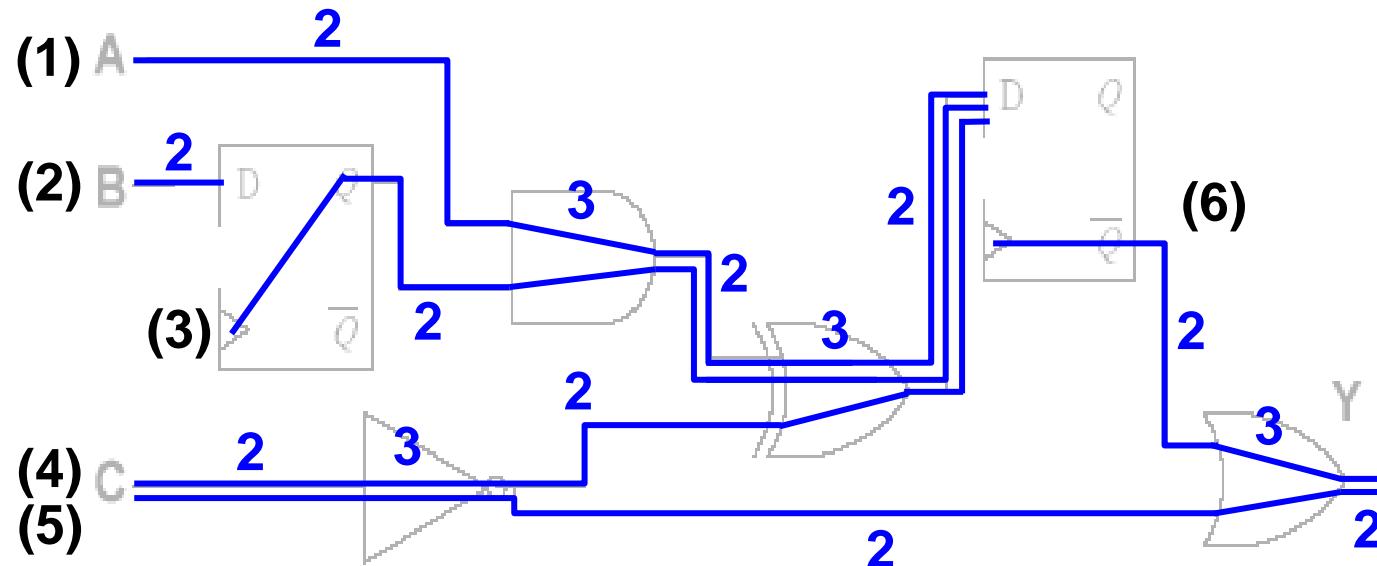
# STA Example: Setup Check (Path-based)

- Find out all the timing paths in this design and calculate the slack of each path
  - 6 timing paths in this design





# STA Example: Setup AT (Path-based)



$$(1) D_A + (2+3+2+3+2) = 13$$

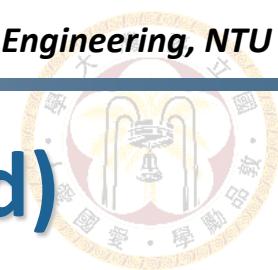
$$(2) D_B + (2) = 3$$

$$(3) T_{clk} + T_{cq} + (2+3+2+3+2) = 20$$

$$(4) D_C + (2+3+2+3+2) = 13$$

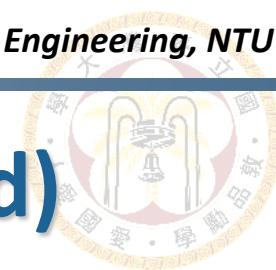
$$(5) D_C + (2+3+2+3+2) = 13$$

$$(6) T_{clk} + T_{cq} + (2+3+2) = 15$$



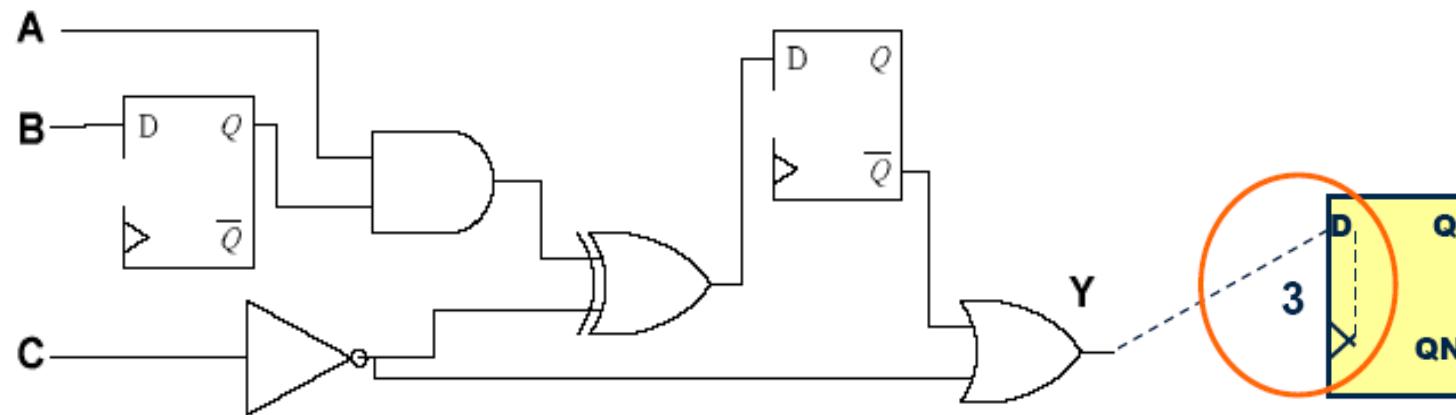
# STA Example: Setup RT (Path-based)

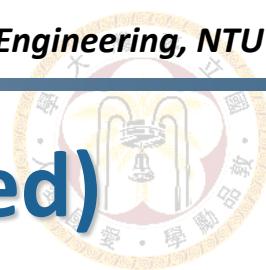
- **Timing path (1): input-to-register**
  - RT:  $T_{cycle} + T_{clk} - T_u - T_{su} = 14 + 5 - 1 - 1 = 17\text{ns}$
- **Timing path (2): input-to-register**
  - RT:  $T_{cycle} + T_{clk} - T_u - T_{su} = 14 + 5 - 1 - 1 = 17\text{ns}$
- **Timing path (3): register-to-register**
  - RT:  $T_{cycle} + T_{clk} - T_u - T_{su} = 14 + 5 - 1 - 1 = 17\text{ns}$
- **Timing path (4): input-to-register**
  - RT:  $T_{cycle} + T_{clk} - T_u - T_{su} = 14 + 5 - 1 - 1 = 17\text{ns}$



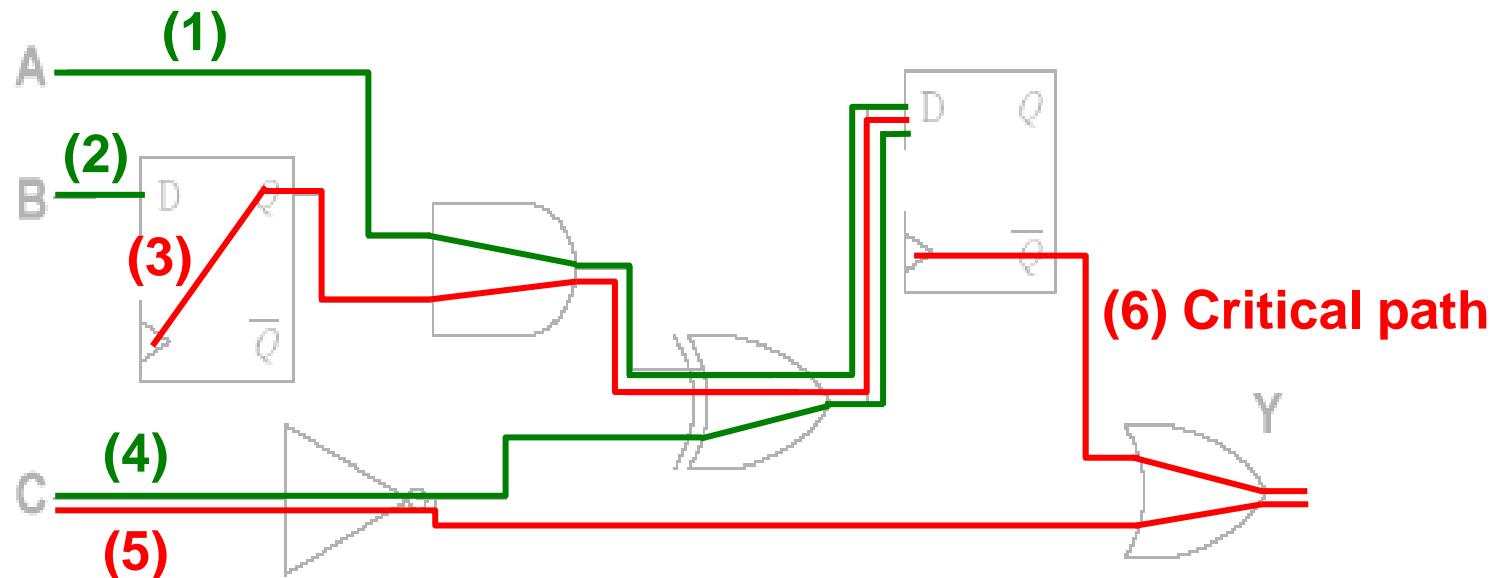
# STA Example: Setup RT (Path-based)

- **Timing path (5): input-to-output**
  - RT:  $T_{cycle} - D_Y = 14 - 3 = 11\text{ns}$
- **Timing path (6): register-to-output**
  - RT:  $T_{cycle} - D_Y = 14 - 3 = 11\text{ns}$





# STA Example: Setup Slack (Path-based)



(1) Slack =  $17-13 = 4$  (met)

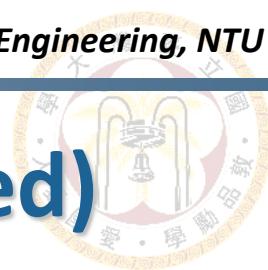
(2) Slack =  $17-3 = 14$  (met)

(3) Slack =  $17-20 = -3$  (violated)

(4) Slack =  $17-13 = 4$  (met)

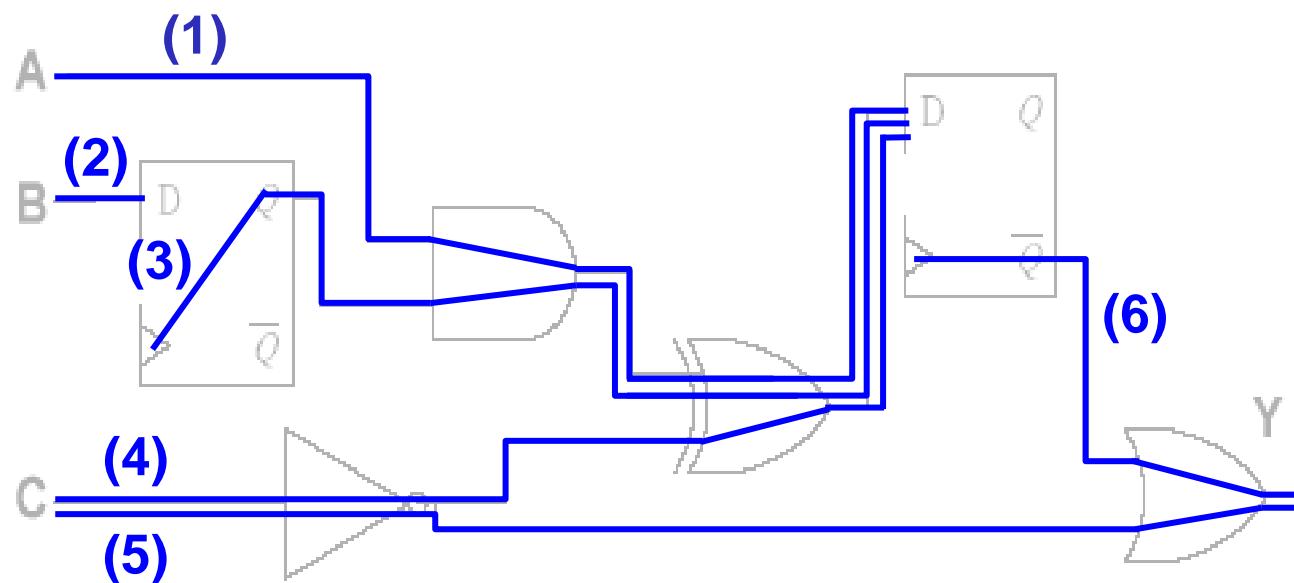
(5) Slack =  $11-13 = -2$  (violated)

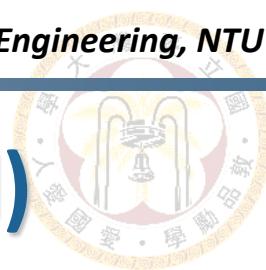
(6) Slack =  $11-15 = -4$  (violated)



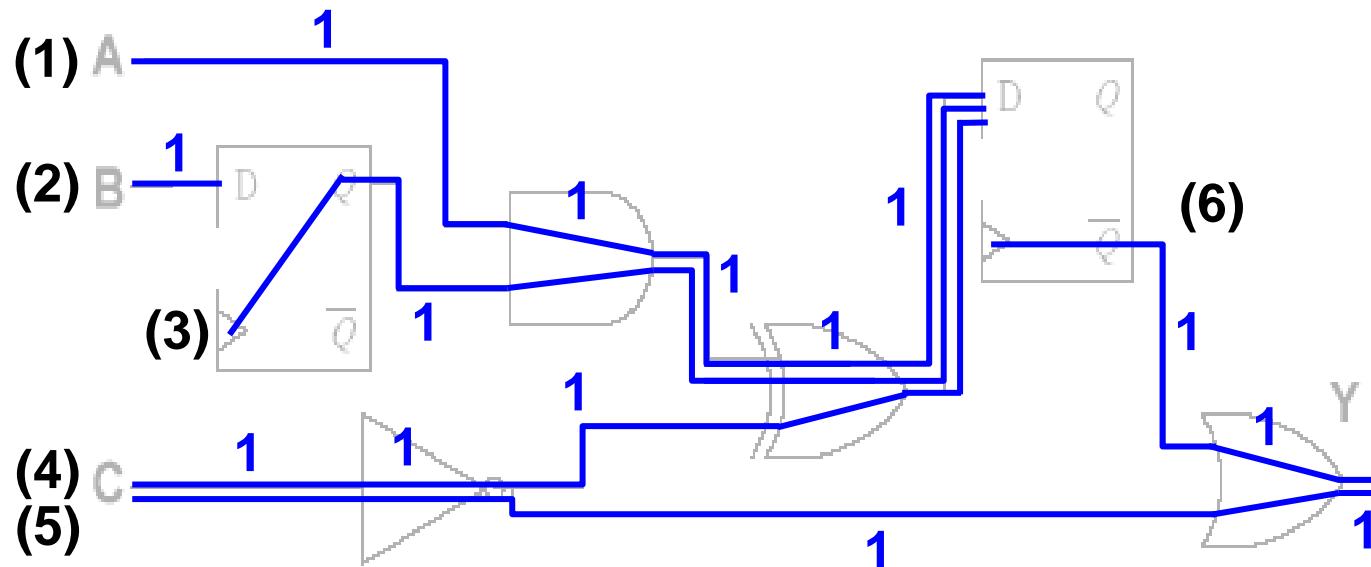
# STA Example: Hold Check (Path-based)

- Find out all the timing paths in this design and calculate the slack of each path
  - 6 timing paths in this design





# STA Example: Hold AT (Path-based)



$$(1) D_A + (1+1+1+1+1) = 6$$

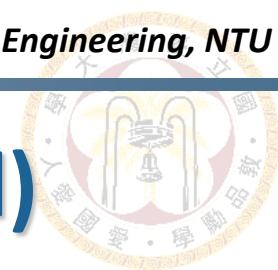
$$(2) D_B + (1) = 2$$

$$(3) T_{clk} + T_{cq} + (1+1+1+1+1) = 13$$

$$(4) D_C + (1+1+1+1+1) = 6$$

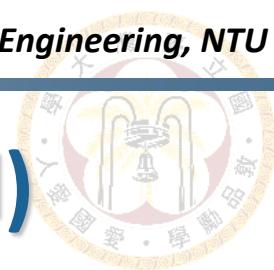
$$(5) D_C + (1+1+1+1+1) = 6$$

$$(6) T_{clk} + T_{cq} + (1+1+1) = 11$$



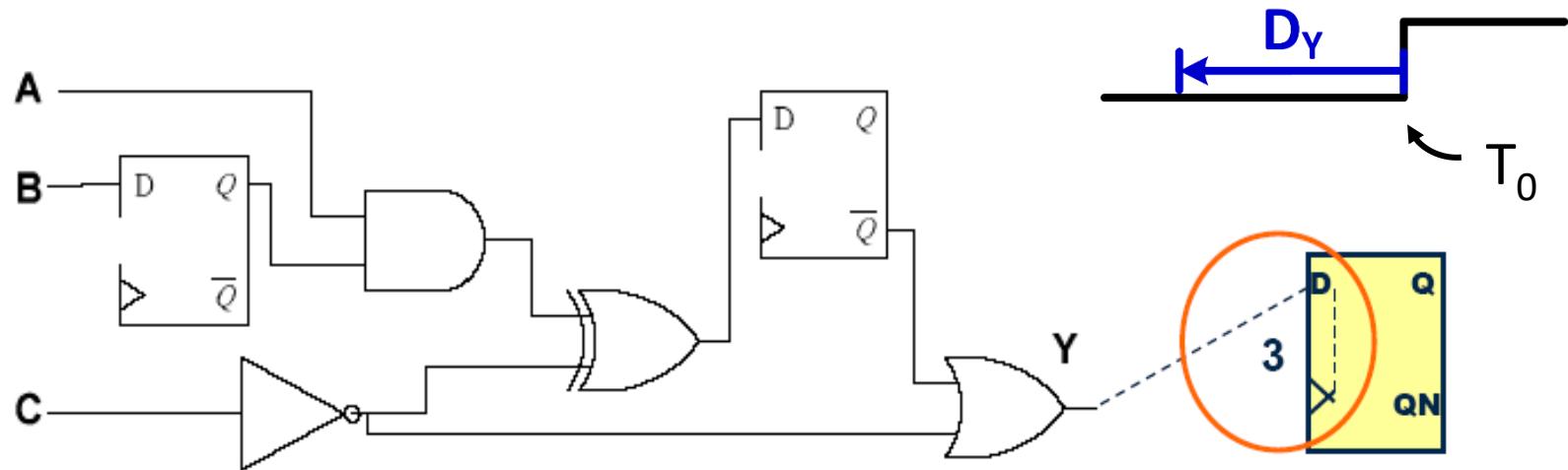
# STA Example: Hold RT (Path-based)

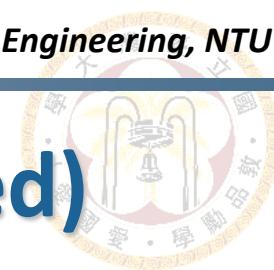
- **Timing path (1): input-to-register**
  - RT:  $T_{clk} + T_u + T_h = 5 + 1 + 1 = 7\text{ns}$
- **Timing path (2): input-to-register**
  - RT:  $T_{clk} + T_u + T_h = 5 + 1 + 1 = 7\text{ns}$
- **Timing path (3): register-to-register**
  - RT:  $T_{clk} + T_u + T_h = 5 + 1 + 1 = 7\text{ns}$
- **Timing path (4): input-to-register**
  - RT:  $T_{clk} + T_u + T_h = 5 + 1 + 1 = 7\text{ns}$



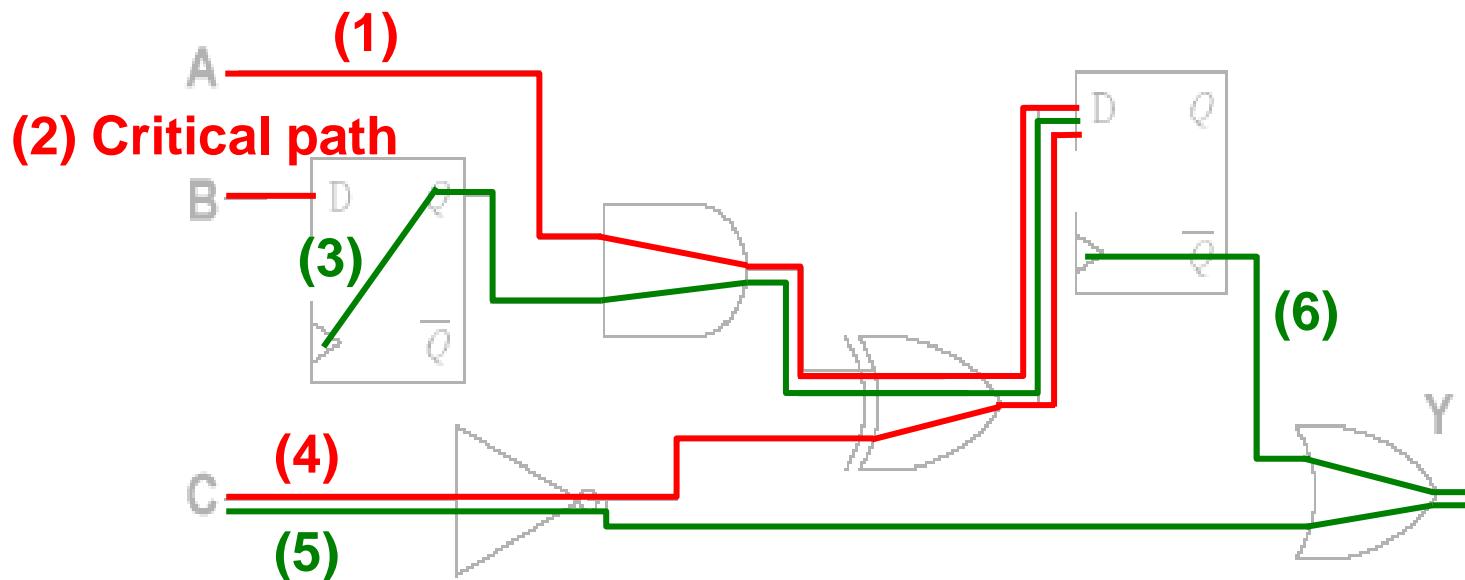
# STA Example: Hold RT (Path-based)

- Timing path (5): input-to-output
  - RT: -  $D_Y = -3\text{ns}$
- Timing path (6): register-to-output
  - RT: -  $D_Y = -3\text{ns}$





# STA Example: Hold Slack (Path-based)



(1) Slack = 6-7 = -1 (**violated**)

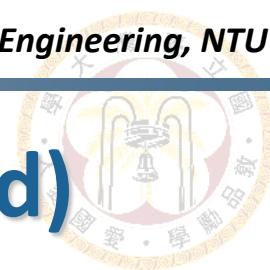
(4) Slack = 6-7 = -1 (**violated**)

(2) Slack = 2-7 = -5 (**violated**)

(5) Slack = 6-(-3) = 9 (**met**)

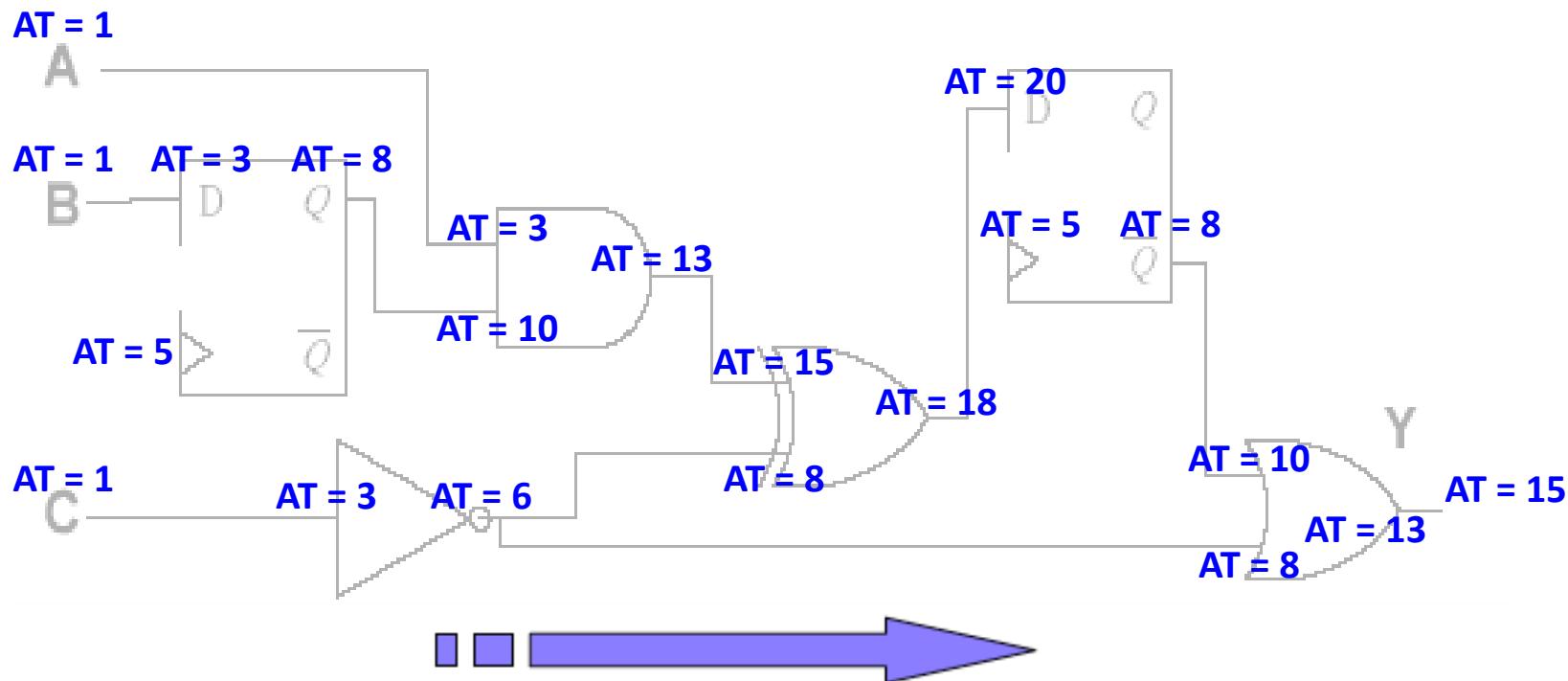
(3) Slack = 13-7 = 6 (**met**)

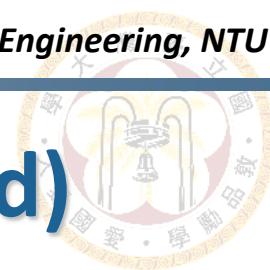
(6) Slack = 11-(-3) = 14 (**met**)



# STA Example: Setup AT (Block-based)

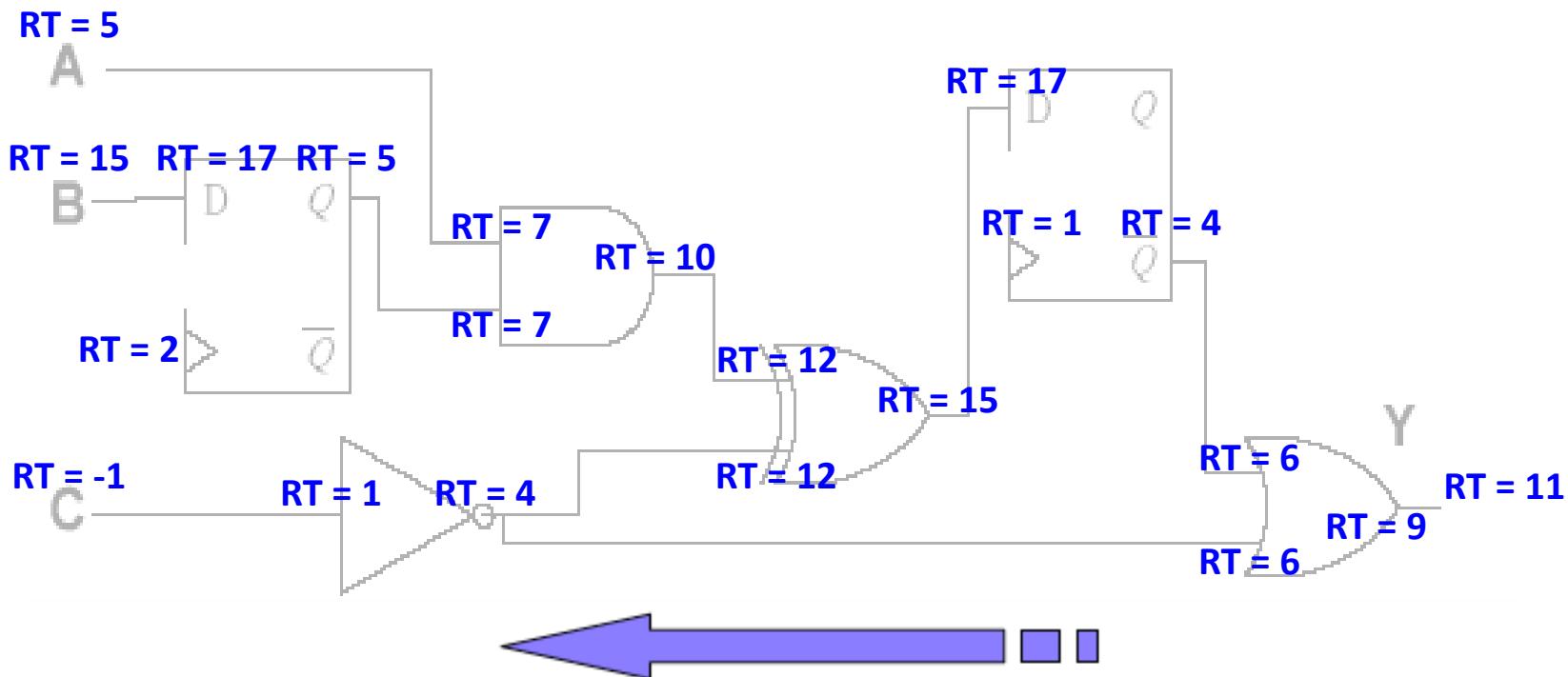
- Calculate the arrival time of each design element
  - Consider **max** AT at the intersection

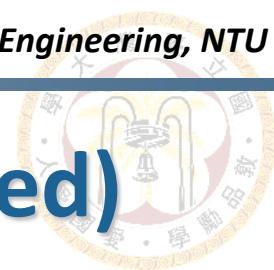




# STA Example: Setup RT (Block-based)

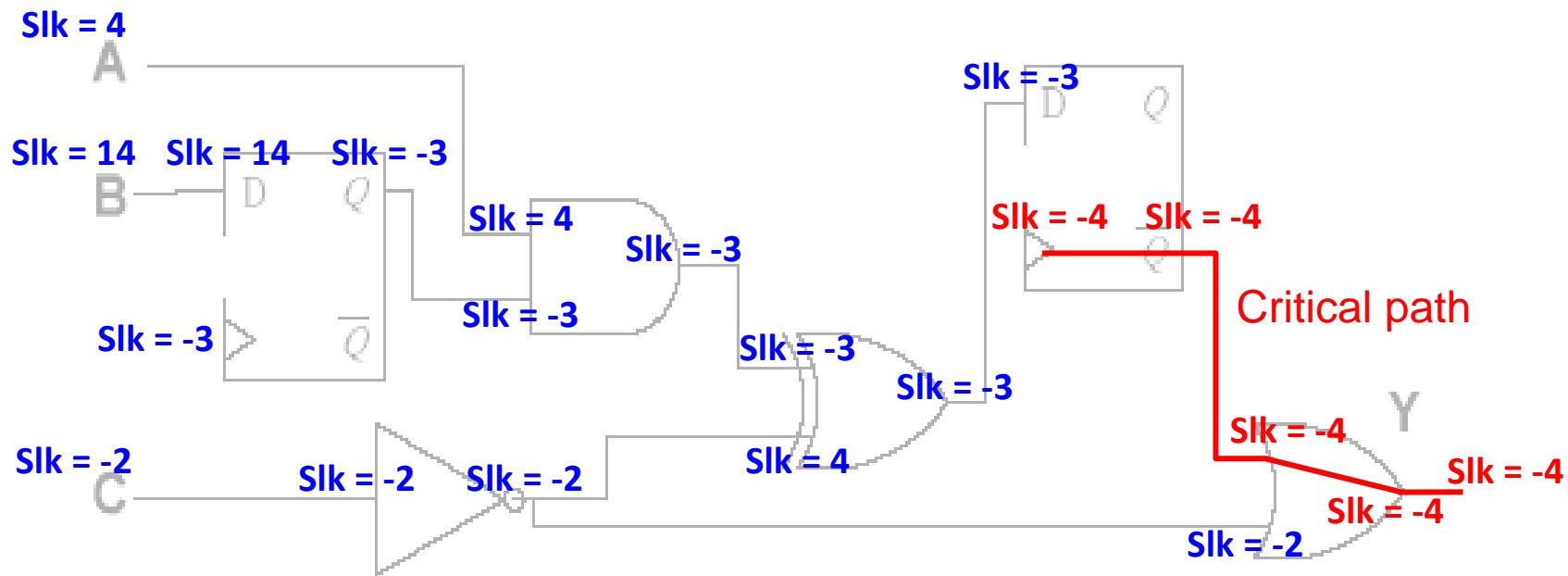
- Calculate the required time of each design element
  - Consider **min** RT at the intersection

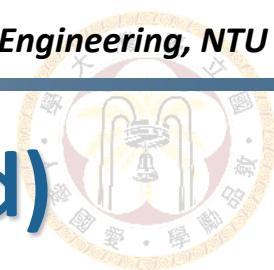




# STA Example: Setup Slack (Block-based)

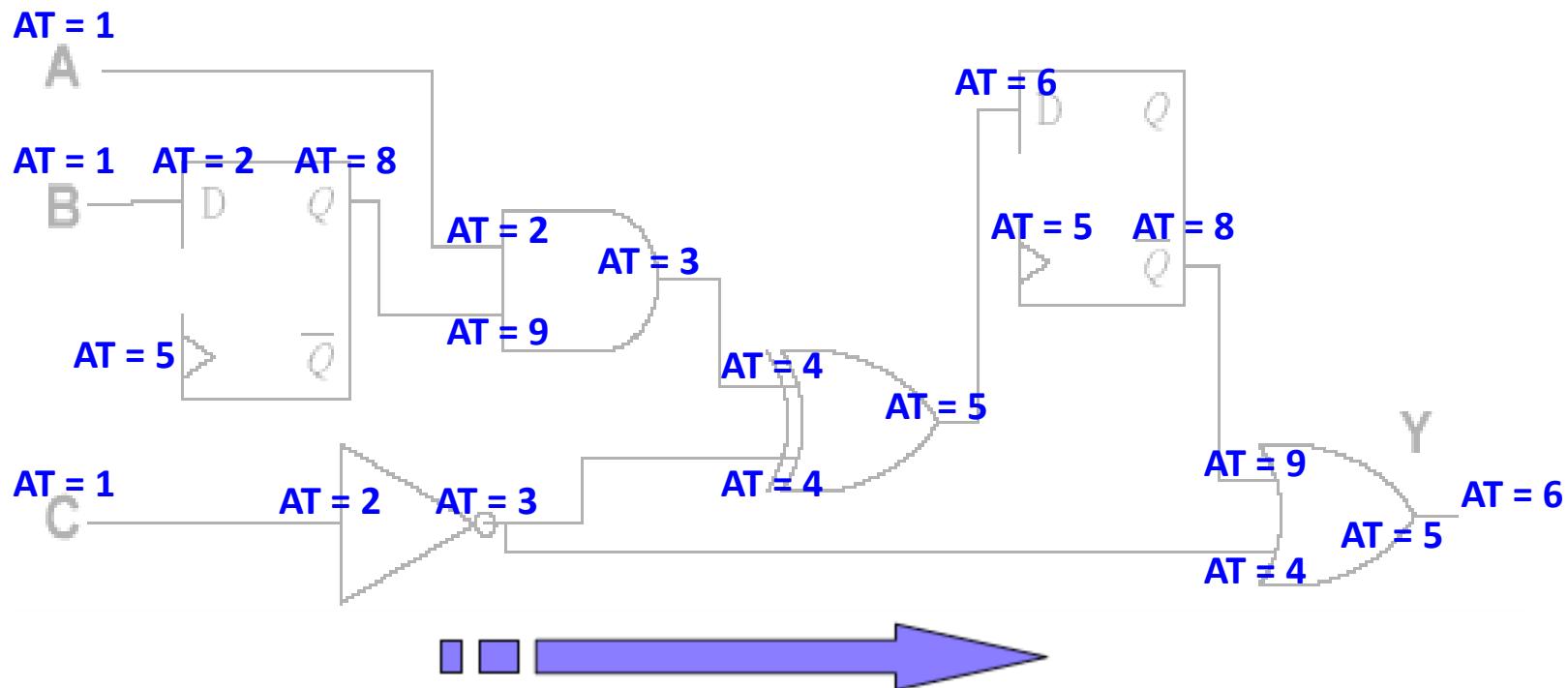
- Calculate the slack ( $Slk$ ) of each design element
  - $- RT - AT$

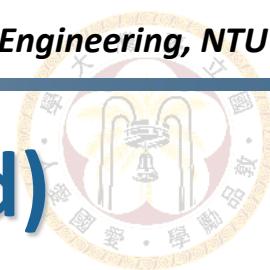




# STA Example: Hold AT (Block-based)

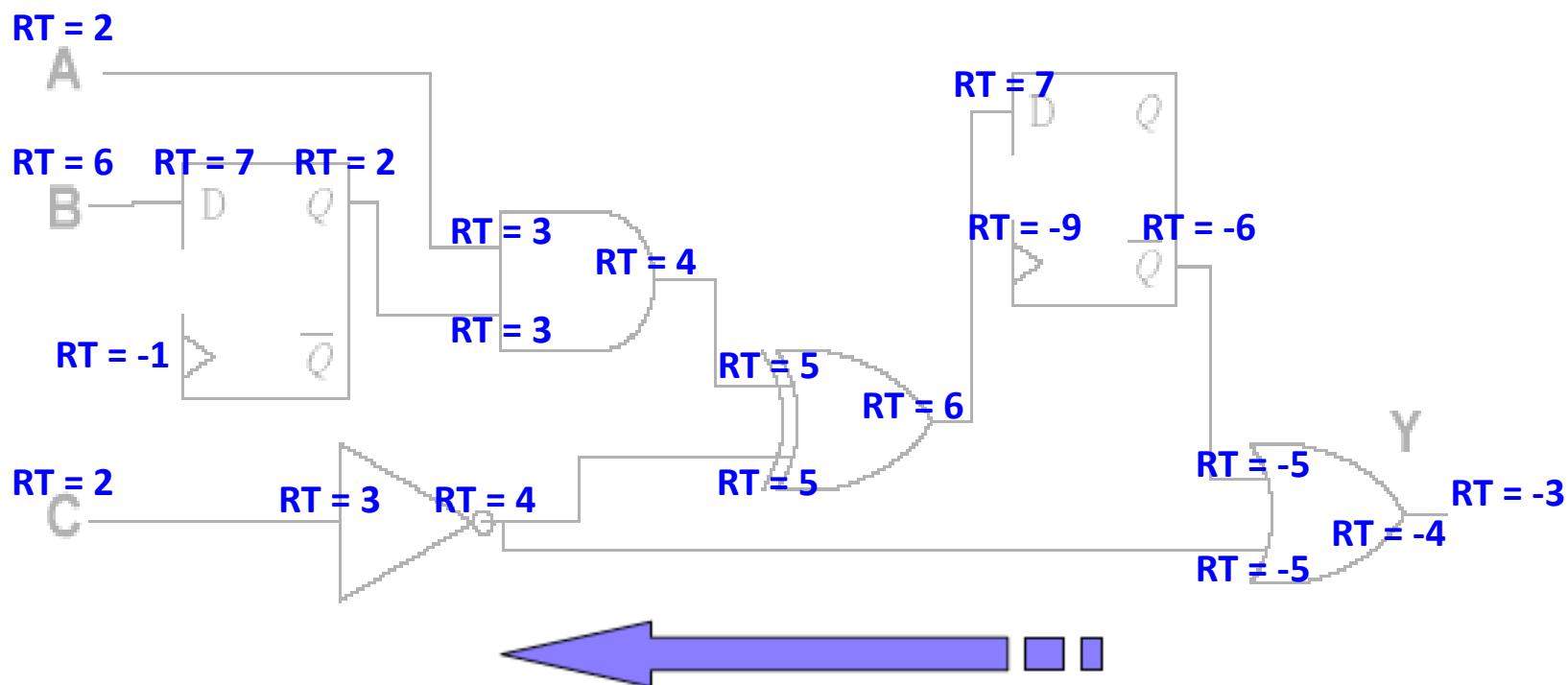
- Calculate the arrival time of each design element
  - Consider **min** AT at each intersection

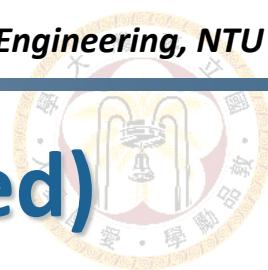




# STA Example: Hold RT (Block-based)

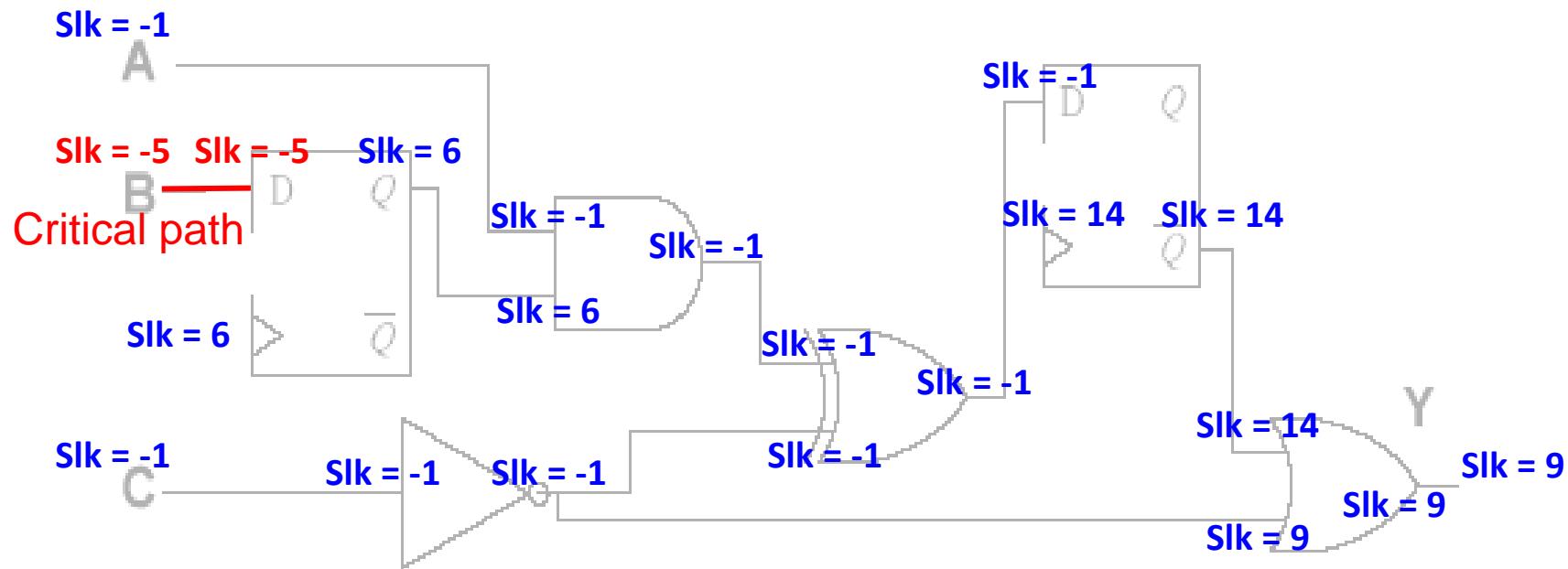
- Calculate the required time of each design element
  - Consider **max** RT at each intersection

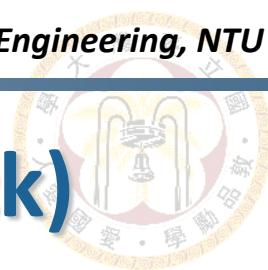




# STA Example: Hold Slack (Block-based)

- Calculate the slack ( $Slk$ ) of each design element
  - $- AT - RT$





# STA in Design Compiler (Setup Check)

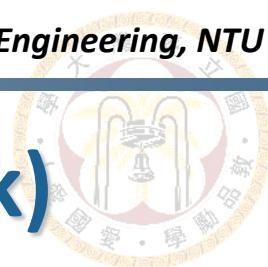
Startpoint:	input_counter_r_reg_1	(rising edge-triggered flip-flop clocked by clk)
Endpoint:	in_to_out_r_reg	(rising edge-triggered flip-flop clocked by clk)
Path Group:	clk	
Path Type:	max	
Point	Incr	Path
clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	1.00	1.00
input_counter_r_reg_1/_CK (DFFRX1)	0.00	1.00 r
input_counter_r_reg_1/_Q (DFFRX1)	0.55	1.55 f
U1861/Y (NAND2XL)	0.20	1.75 r
U1153/Y (INVXL)	0.11	1.85 f
U1936/Y (NAND2X2)	0.18	2.03 r
U1058/Y (INVXL)	0.22	2.26 f
U1975/Y (AO22X1)	0.39	2.65 f
U1977/Y (AOI211X1)	0.13	2.77 r
U1979/Y (OAI211XL)	0.10	2.87 f
U1980/Y (AO21X1)	0.26	3.14 f
U1982/Y (AOI211X1)	0.12	3.26 r
U1317/Y (NAND3XL)	0.10	3.37 f
U1097/Y (MXI2X1)	0.50	3.87 r
U2651/Y (NOR2X1)	0.14	4.01 f
U2652/Y (AOI211X1)	0.15	4.16 r
U1291/Y (INVXL)	0.05	4.21 f
U2653/Y (AO21X1)	0.27	4.49 f
U2654/Y (AOI21X1)	0.13	4.62 r
U2656/Y (OAI31XL)	0.11	4.72 f
U2683/Y (OAI2BB2XL)	0.18	4.90 r
U2684/Y (NOR2BX1)	0.18	5.08 r
U2685/Y (OAI21XL)	0.10	5.18 f
U2693/Y (OAI2BB2XL)	0.19	5.36 r
U2698/Y (OAI211XL)	0.13	5.49 f
U2699/Y (AOI31XL)	0.15	5.65 r
in_to_out_r_reg/D (DFFRX1)	0.00	5.65 r
data arrival time		5.65
clock clk (rise edge)	5.00	5.00
clock network delay (ideal)	1.00	6.00
clock uncertainty	-0.10	5.90
in_to_out_r_reg/CK (DFFRX1)	0.00	5.90 r
library setup time	-0.25	5.65
data required time		5.65
data required time	5.65	
data arrival time	-5.65	
slack (MET)	0.01	

dc\_shell> report\_timing -delay max

$$\text{Arrival time} = T_0 + T_{\text{clk}} + T_{\text{cq}} + T_{\text{pd}}$$

$$\text{Required time} = T_0 + T_{\text{cycle}} + T_{\text{clk}} - T_u - T_{\text{su}}$$

$$\text{Slack} = \text{Required time} - \text{Arrival time}$$



# STA in Design Compiler (Hold Check)

```
dc_shell> report_timing -delay min
```

```

Startpoint: input_counter_r_reg_1
(rising edge-triggered flip-flop clocked by clk)
Endpoint: in_data_r_reg_5_2
(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 (ideal) 1.00   1.00
input_counter_r_reg_1/_CK (DFFRX1) 0.00   1.00 r
input_counter_r_reg_1/_QN (DFFRX1) 0.28   1.28 f
U1161/Y (NAND2X1)          0.14   1.42 r
U1946/Y (NOR2BX1)          0.18   1.60 f
U2568/Y (CLKINVX1)         0.25   1.85 r
U2570/Y (OAI21XL)          0.12   1.97 f
in_data_r_reg_5_2/_D (DFFRX1) 0.00   1.97 f
data arrival time           1.97

-----+-----+-----+
clock clk (rise edge)      0.00   0.00
clock network delay (ideal) 1.00   1.00
clock uncertainty           0.10   1.10
in_data_r_reg_5_2/_CK (DFFRX1) 0.00   1.10 r
library hold time           -0.03  1.07
data required time          1.07

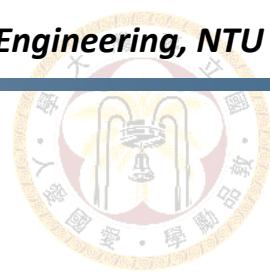
-----+-----+-----+
data required time          1.07
data arrival time            -1.97

-----+-----+-----+
slack (MET)                 0.89
  
```

$$\text{Arrival time} = T_0 + T_{\text{clk}} + T_{\text{cq}} + T_{\text{pd}}$$

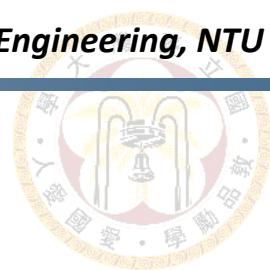
$$\text{Required time} = T_0 + T_{\text{clk}} + T_u + T_h$$

$$\text{Slack} = \text{Arrival time} - \text{Required time}$$



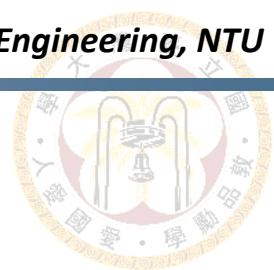
# Outline

- Introduction
- Static Timing Analysis
- **Environment Setting**
- Special Timing Paths
- Clock Domain Crossing

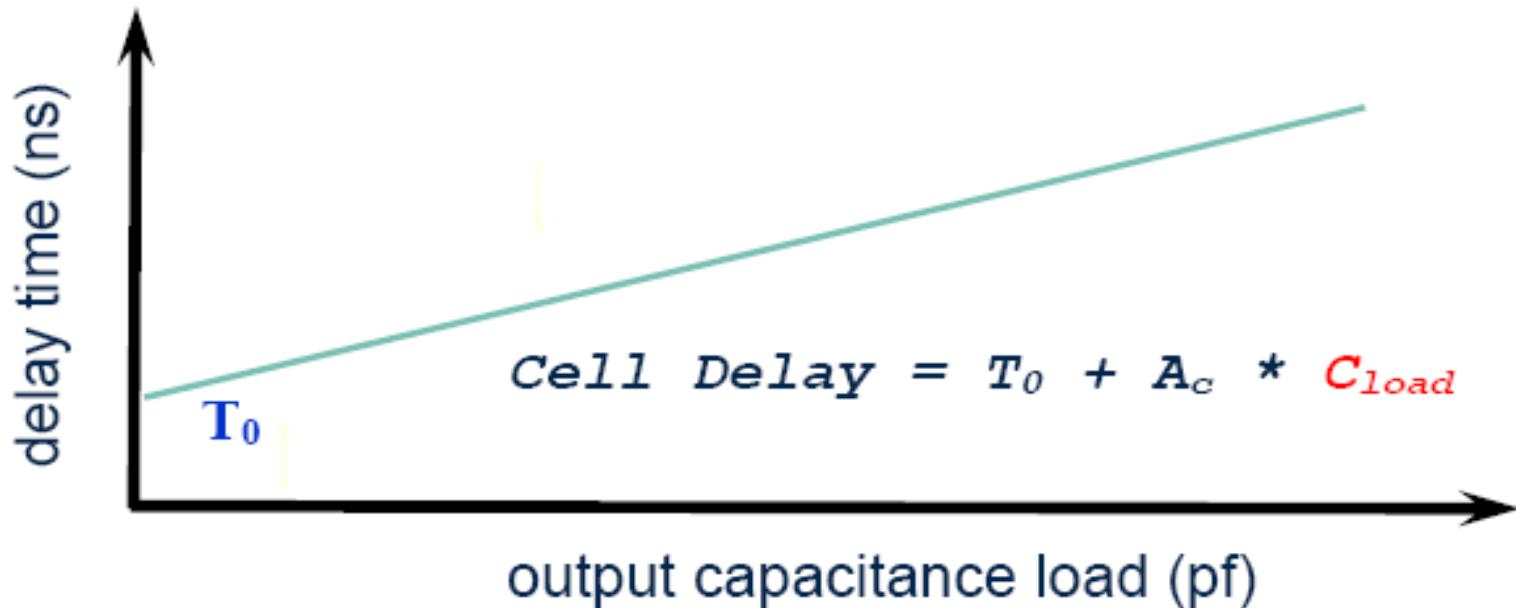


# Library Data

- **Cell Delay model**
  - Linear model
  - Non-linear model
- **Operating conditions**
  - set\_operating\_conditions "typical" -library "typical"
  - set\_wire\_load\_model -name "umc18\_wl50" -library "typical"

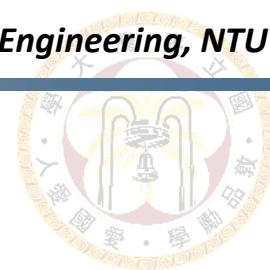


# Linear Cell Delay Model



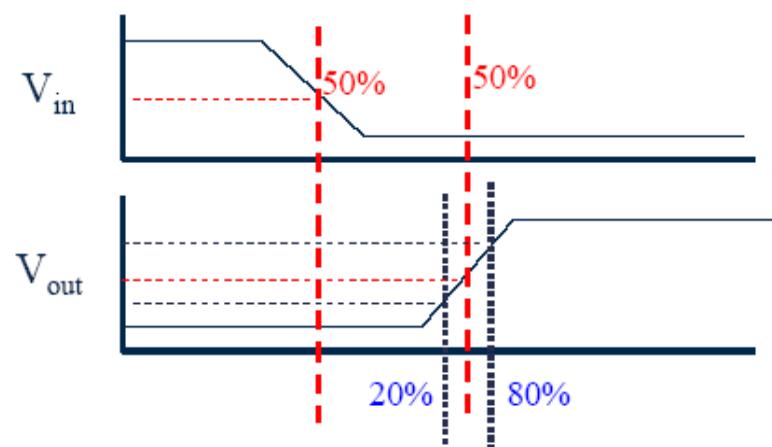
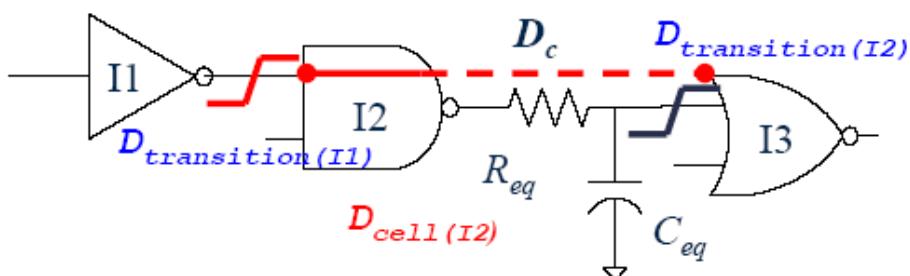
$T_0$  : cell pin to pin intrinsic delay  
(delay without any loading)

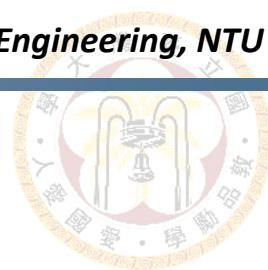
$A_c$  : drive impedance



# Non-linear Cell Delay Model

- Delay values are stored in the delay tables
  - Cell delay
  - Transition delay
- Interpolation is used





# Delay Table

*Cell Delay*

$$D_{cell(I2)} = f(D_{transition(I1)}, C_{eq})$$

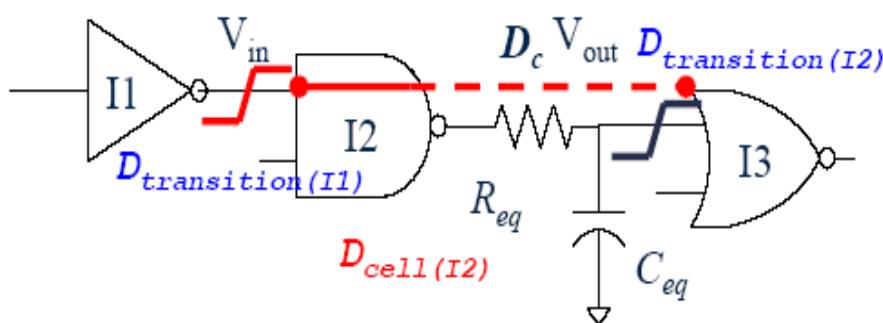
*Transition Delay*

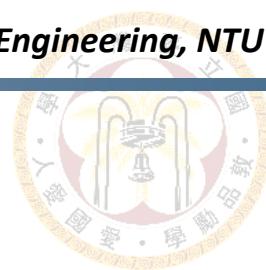
$$D_{transition(I2)} = g(D_{transition(I1)}, C_{eq})$$

Output Capacitance	Input Transition		
	0	0.5	1
0.1	0.123	0.234	0.456
0.2	0.222	0.432	0.801

index1: input transition

Index2: output capacitance





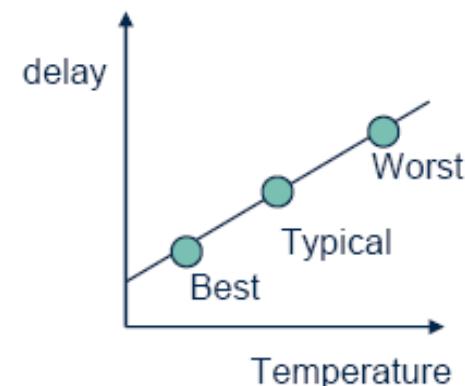
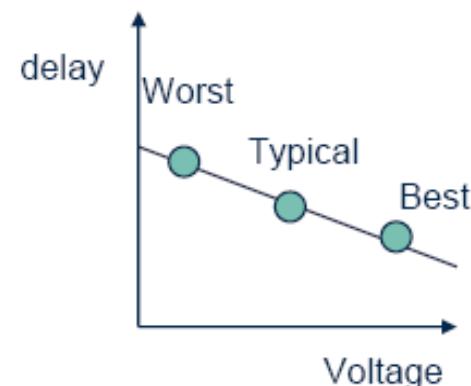
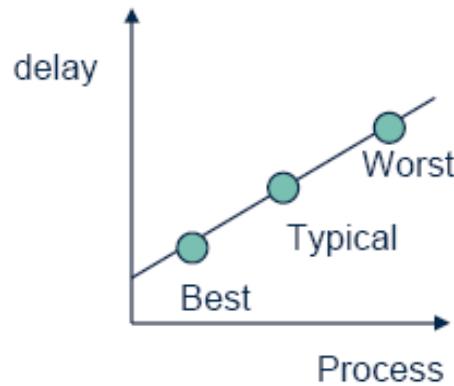
# Operating Conditions

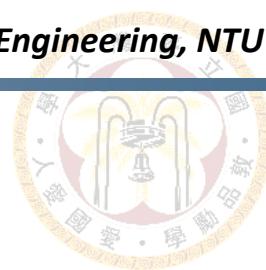
- The process, voltage, and temperature (PVT) variation
- Specified in the technology library
- Cell and interconnect delays are scaled

$$D_{scale} = D(1 + \Delta_p K_p)(1 + \Delta_v K_v)(1 + \Delta_t K_t)$$

$\Delta_p = \text{Process - nom\_process} / \Delta_v = \text{Voltage - nom\_voltage}$

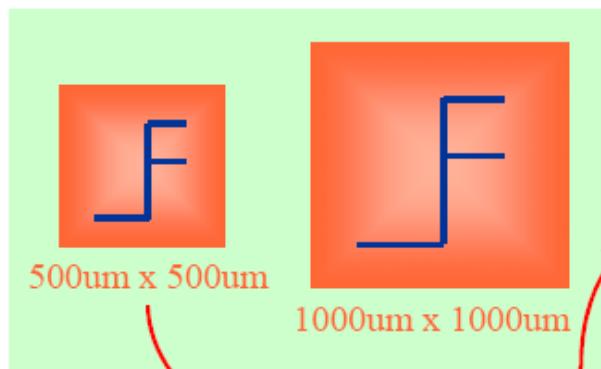
$\Delta_t = \text{Temperature - nom\_temperature}$



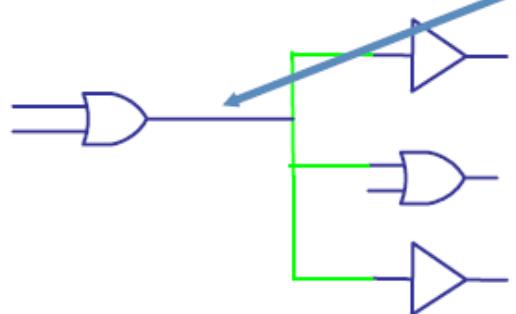


# Wireload Model

- Very inaccurate! (The settings are like that in DC)



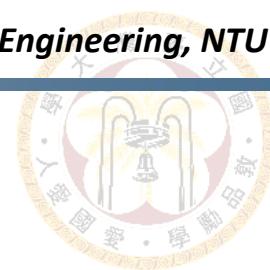
```
wire_load ("500") (  
    resistance : 3.0  
    capacitance: 1.3  
    area: 0.04  
    slop: 0.15  
    fanout_length ( 1 , 2.1 )  
    fanout_length ( 2 , 2.5)  
    fanout_length ( 3 , 2.8)  
    fanout_length ( 4 , 3.3)  
    /* R per unit length */  
    /* C per unit length */  
    /* area per unit length */  
    /* extrapolation slope */  
    /* fanout-length pairs */
```



$$C_{\text{wire}} = (\text{fanout}=3, \text{length }=2.8) \times \text{capacitance coefficient } (1.3) = 3.64 \text{ load units}$$

$$R_{\text{wire}} = (\text{fanout}=3, \text{length }=2.8) \times \text{resistance coefficient } (3.0) = 8.4 \text{ resistance units}$$

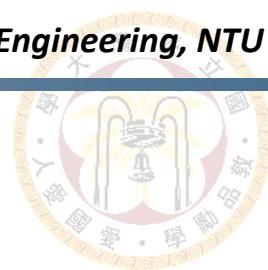
$$\text{Area}_{\text{Net}} = (\text{fanout}=3, \text{length }=2.8) \times \text{area coefficient } (0.04) = 0.112 \text{ net area units}$$



# Standard Delay Format (SDF)

- Describes the timing characteristics of circuit components
  - Flip-flops, logic gates, interconnects...
- Delay value: (min\_delay : typ\_delay : max\_delay)

```
(CELL
  (CELLTYPE "NAND2")
  (INSTANCE U_NAND)
  (DELAY
    (ABSOLUTE      low→high           high→low
      (IOPATH A Y (0.122:0.122:0.122) (0.089:0.089:0.089))
      (IOPATH B Y (0.136:0.136:0.136) (0.092:0.092:0.092))
    )
  )
)
```



# SDF Annotation

- Annotates the circuit components with delay information
- **Must-do** for gate-level/post-layout simulation
- Annotate in testbench (by built-in function)

```
$sdf_annotation(sdf_file, module_instance);
```

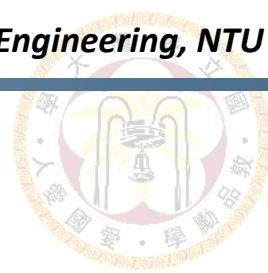
- Check for successful annotation

## NC-Verilog

```
Annotation completed with 0 Errors and 518 Warnings
SDF statistics: No. of Pathdelays = 6803  Annotated = 100.00% -- No. of Tchecks = 1674  Annotated = 100.00%
      Total      Annotated      Percentage
Path Delays      6803      6803      100.00
      $width       837       837      100.00
      $setuphold   837       837      100.00
```

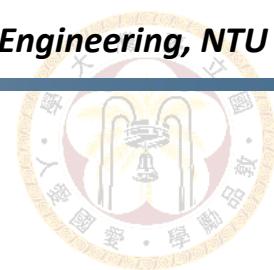
## VCS

```
| Doing SDF annotation ..... Done
```



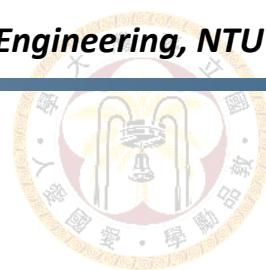
# Outline

- Introduction
- Static Timing Analysis
- Environment Setting
- **Special Timing Paths**
- Clock Domain Crossing



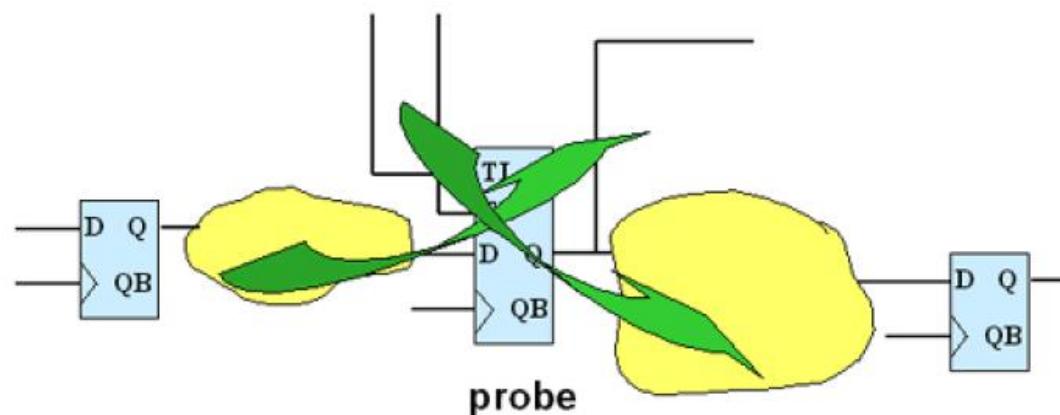
# Special Timing Paths: False Paths

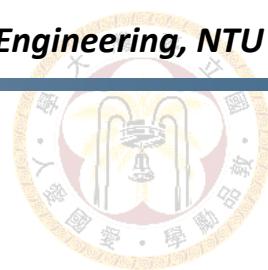
- Timing paths that will be **ignored** during STA
- **5 types of false paths**
  - Unexercised path
  - Irrelevant path
  - Asynchronous path
  - Logically impossible path
  - Combinational loops
- **Should be used carefully**



# Unexercised Path

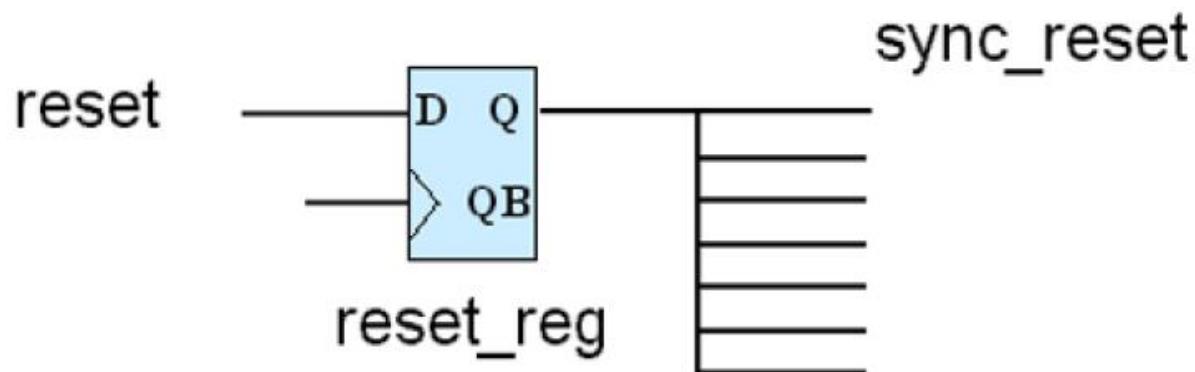
- A path may exist in the circuit but never be used in its normal functional operation
  - A test register “probe” is inserted in the circuit to enable chip debugging in the field
    - Data can be read through the probe register
    - Data can be written from the probe register
  - Probing would not occur at speed

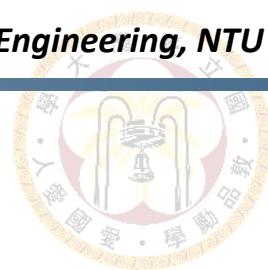




# Irrelevant Path

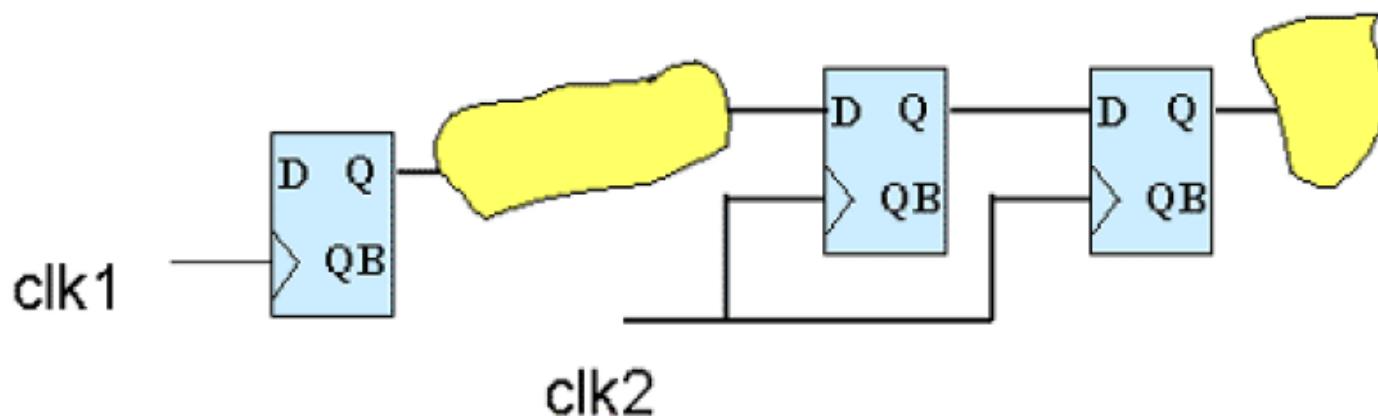
- A functional path which its timing is so slow or irrelevant
  - The chip uses a synchronous reset. The reset cycle has a huge number of cycles before it needs to settle

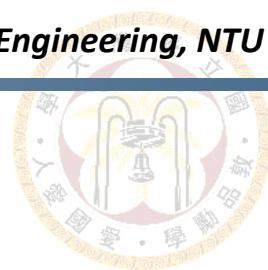




# Asynchronous Path

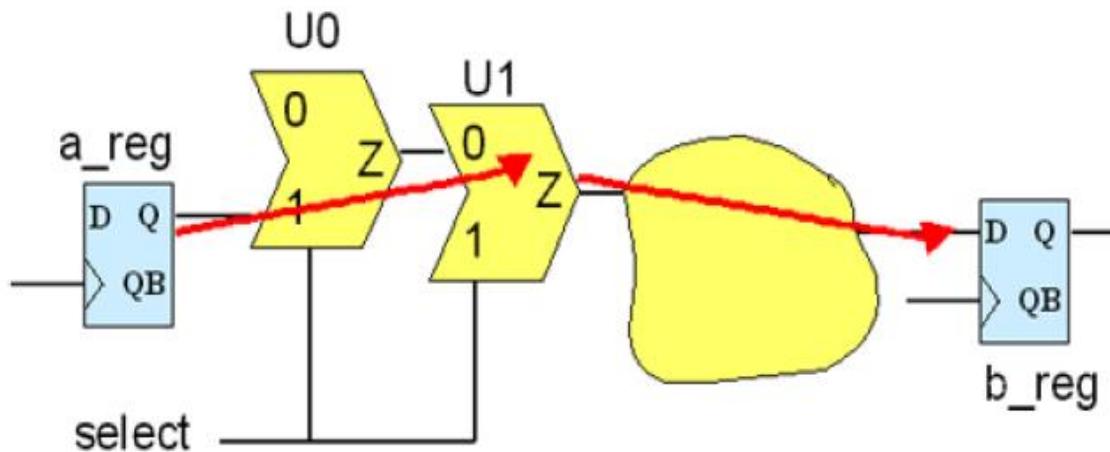
- A path between 2 different clock domains
  - Clock domain crossing (CDC): transfer data from clk 1 to clk 2

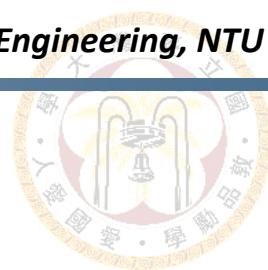




# Logically Impossible Path

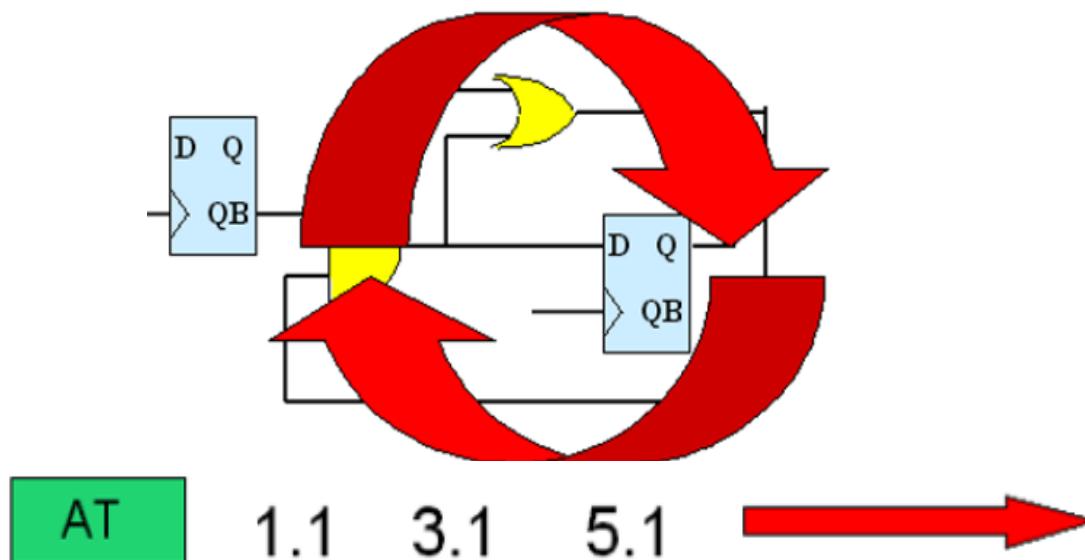
- A path may exist in the circuit but no combination of input vectors may ever exercise it
- PrimeTime attempts to automatically detect "logically impossible false paths" (requires many CPU cycles)
- These situations are quite rare

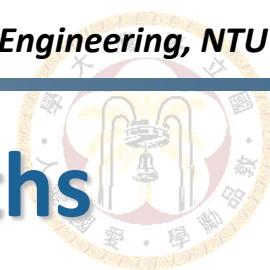




# Combinational Loops

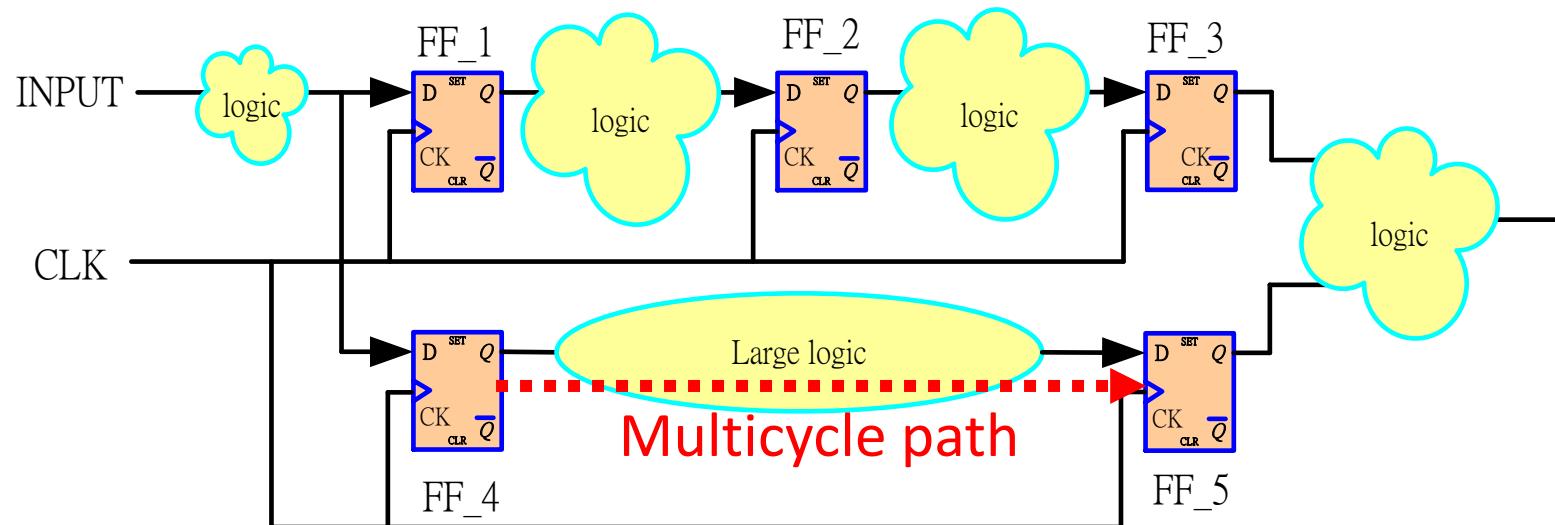
- A combinational loop that needs to be broken
  - Most STA's can't leave combinational loops in the design, as a race condition will occur

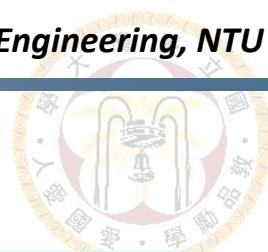




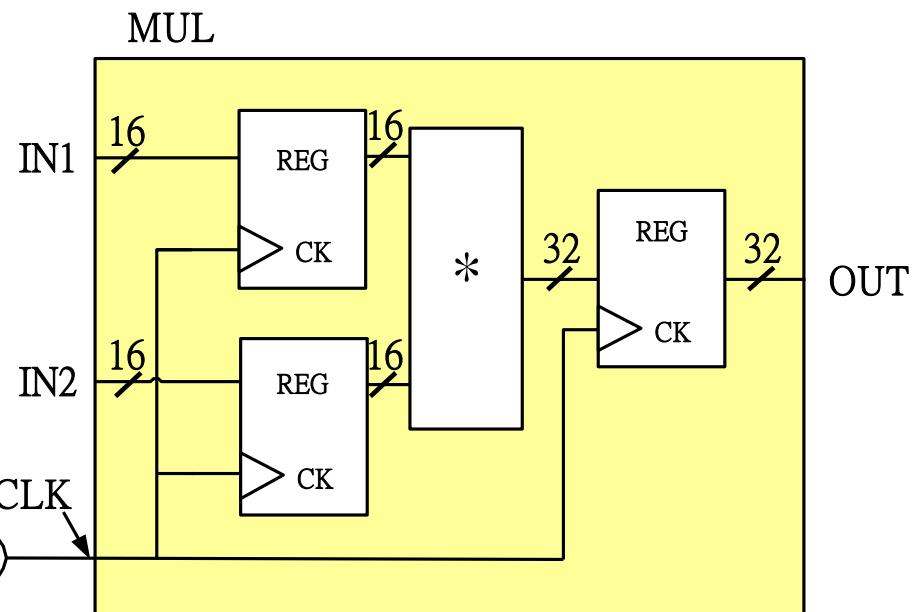
# Special Timing Paths: Multicycle Paths

- Timing paths that are not expected to propagate signals in one cycle, and are carefully designed so that we can **get the results multiple cycles later**
  - Multiplier/divider





# Example: Multiplier



```
module MUL
(
    // Output Port
    OUT,
    // Input Port
    IN1, IN2, CLK
);

output reg[31:0] OUT;
input [15:0] IN1, IN2;
input CLK;

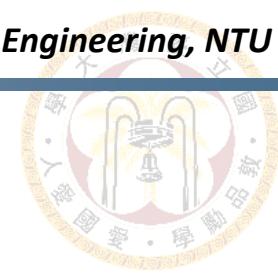
reg [15:0] IN1_R, IN2_R;

always @(posedge CLK)
    IN1_R <= IN1;

always @(posedge CLK)
    IN2_R <= IN2;

always @(posedge CLK)
    OUT <= IN1_R*IN2_R;

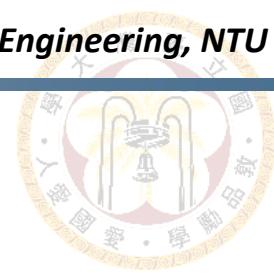
endmodule
```



# Example: STA Result

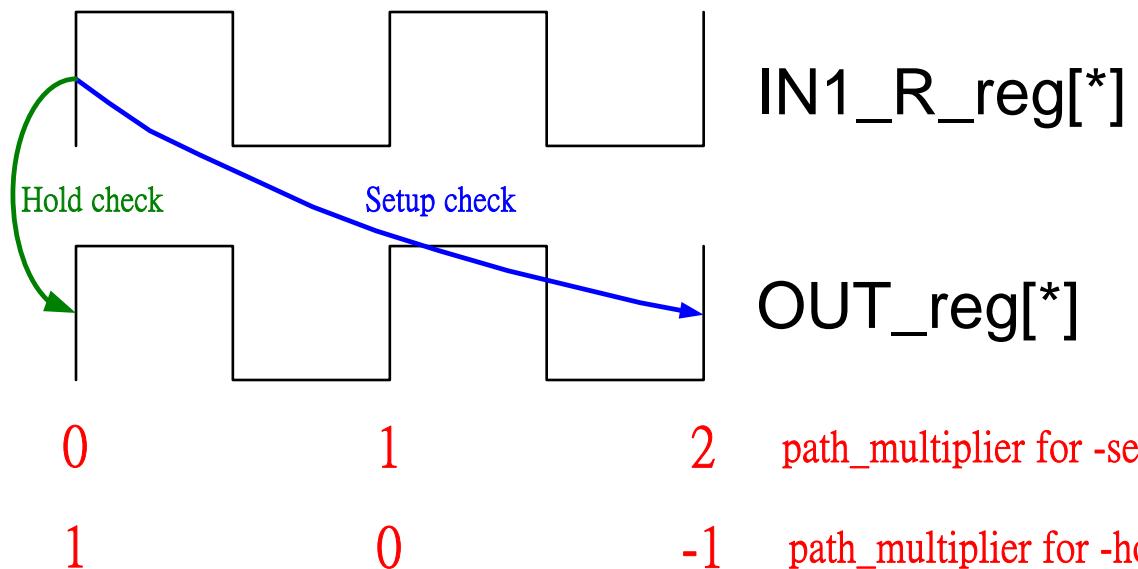
```
dc_shell>create_clock -period 4 CLK  
dc_shell>compile  
dc_shell>report_timing
```

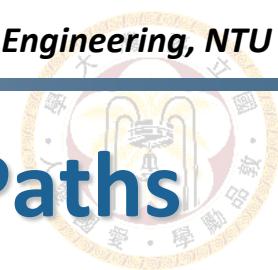
*****			
Report : timing -path full -delay max -max_paths 1			
Design : MUL			
Point	Incr	Path	
clock CLK (rise edge)	0.00	0.00	
clock network delay (ideal)	0.00	0.00	
IN1_R_reg[14]/CK (DFFHQX4)	0.00	0.00	r
IN1_R_reg[14]/Q (DFFHQX4)	0.30	0.30	r
U14/Y (BUFX20)	0.18	0.48	r
mult_21/A[14]			
(MUL_DW02_mult_16_16_0)	0.00	0.48	r
... (ignored)			
mult_21/PRODUCT[28]			
(MUL_DW02_mult_16_16_0)	0.00	4.91	r
OUT_reg[28]/D (DFFXL)	0.00	4.91	r
data arrival time			4.91
clock CLK (rise edge)	4.00	4.00	
clock network delay (ideal)	0.00	4.00	
OUT_reg[28]/CK (DFFXL)	0.00	4.00	r
library setup time	-0.10	3.90	
data required time			3.90
data required time			3.90
data arrival time			-4.91
slack (VOLATED)			-1.00



# Example: Set Multicycle Paths

```
dc_shell>set_multicycle_path 2 -from [get_cells IN1_R_reg[*]] -to [get_cells OUT_reg[*]]  
dc_shell>set_multicycle_path 2 -from [get_cells IN2_R_reg[*]] -to [get_cells OUT_reg[*]]  
dc_shell>set_multicycle_path 1 -hold -end -from [get_cells IN1_R_reg[*]] -to [get_cells OUT_reg[*]]  
dc_shell>set_multicycle_path 1 -hold -end -from [get_cells IN2_R_reg[*]] -to [get_cells OUT_reg[*]]
```

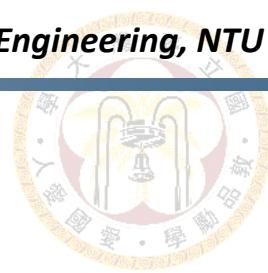




# Example: STA Result with Multicycle Paths

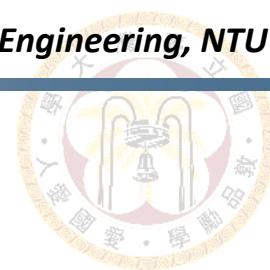
```
dc_shell>compile  
dc_shell>report_timing
```

*****		
Report : timing -path full -delay max -max_paths 1		
Design : MUL		
*****		
Point	Incr	Path
clock CLK (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
IN1_R_reg[13]/CK (DFFX1)	0.00	0.00 r
IN1_R_reg[13]/Q (DFFX1)	1.27	1.27 r
mult_21/A[13]		
(MUL_DW02_mult_16_16_0)	0.00	1.27 r
.....(ignored)		
mult_21/PRODUCT[28]		
(MUL_DW02_mult_16_16_0)	0.00	7.56 f
OUT_reg[28]/D (DFFX1)	0.00	7.56 f
data arrival time		7.56
clock CLK (rise edge)	8.00	8.00
clock network delay (ideal)	0.00	8.00
OUT_reg[28]/CK (DFFX1)	0.00	8.00 r
library setup time	-0.33	7.67
data required time		7.67
-----		
data required time		7.67
data arrival time		-7.56
-----		
slack (MET)		0.11



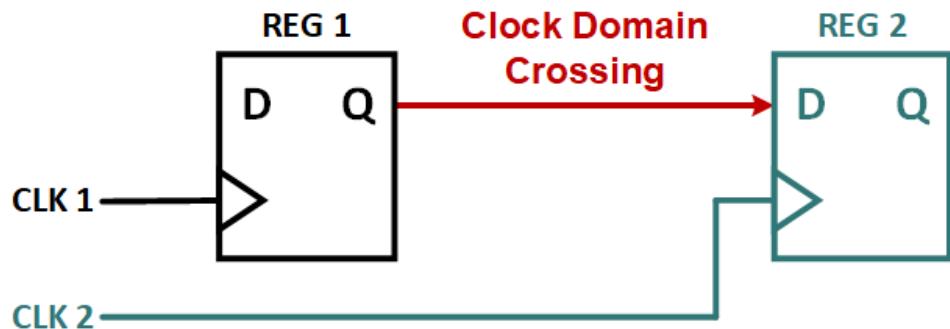
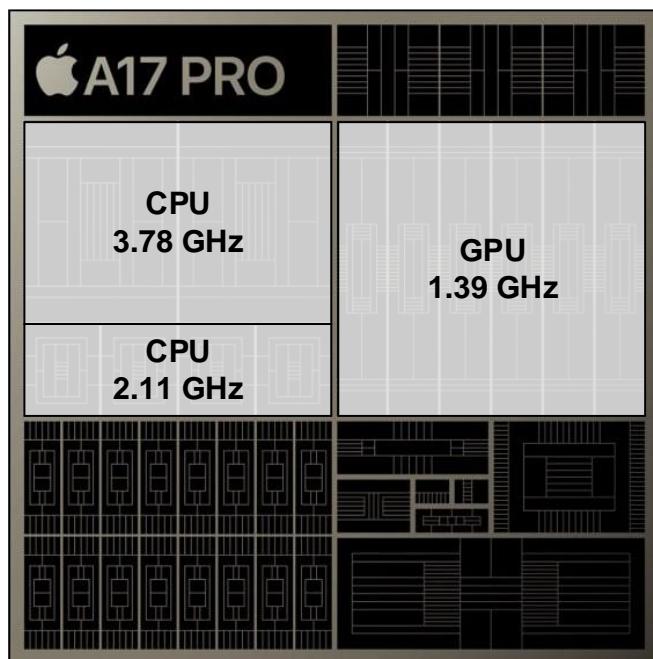
# Outline

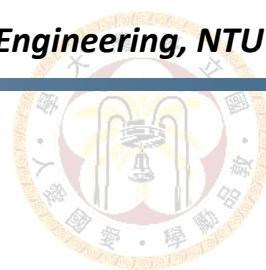
- Introduction
- Static Timing Analysis
- Environment Setting
- Special Timing Paths
- **Clock Domain Crossing**



# Clock Domain Crossing (CDC)

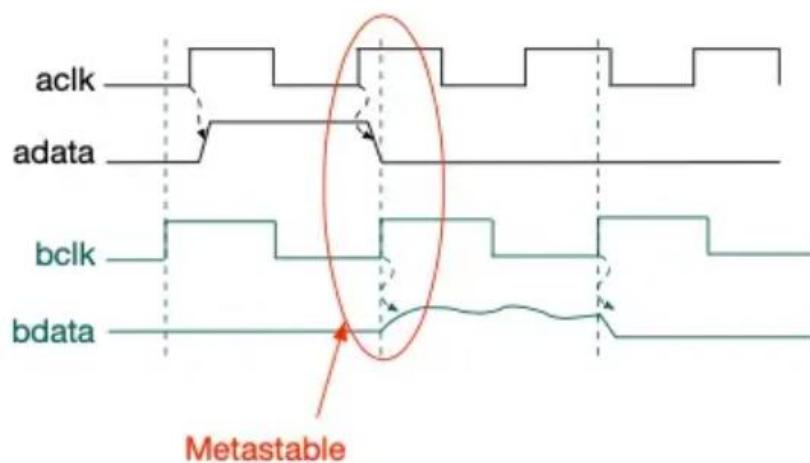
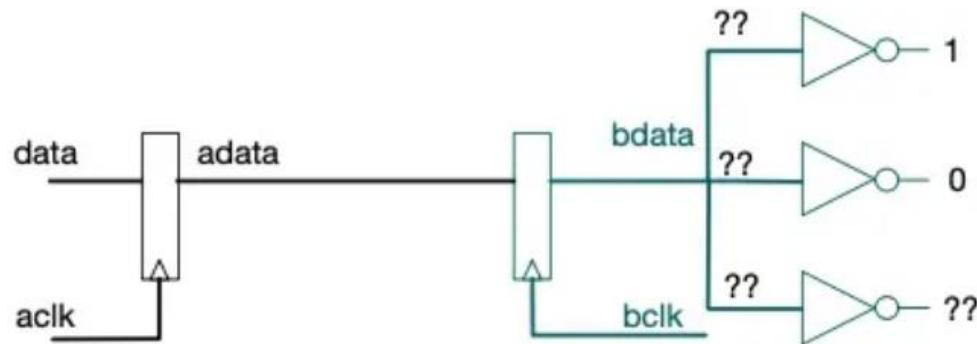
- Multiple clock domains are used in modern design
  - Performance optimization, power efficiency, modularity...
- CDC means data transitions between different clock domains

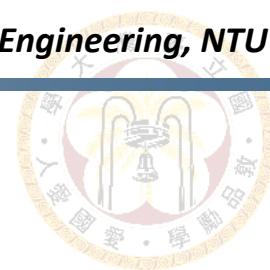




# CDC Issue 1: Metastability

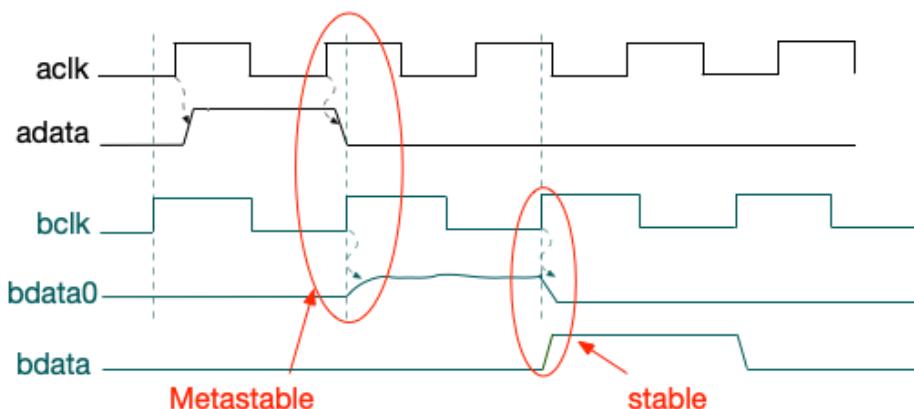
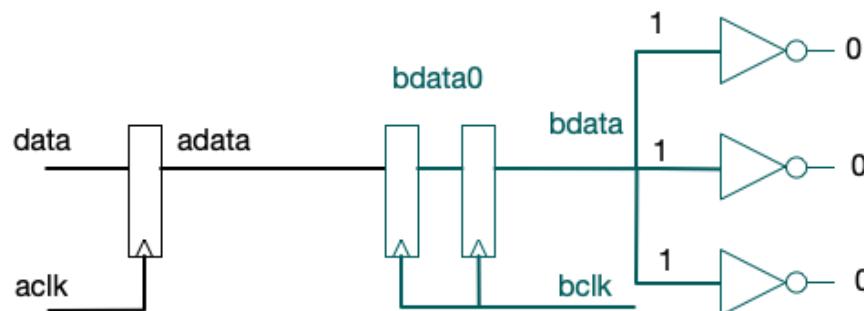
- The unstable status due to non-ideal data transition is called metastability

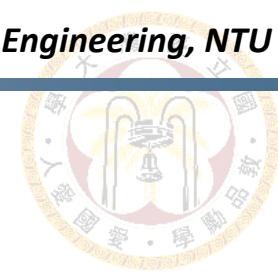




# Solution: 2-Flop Synchronizer

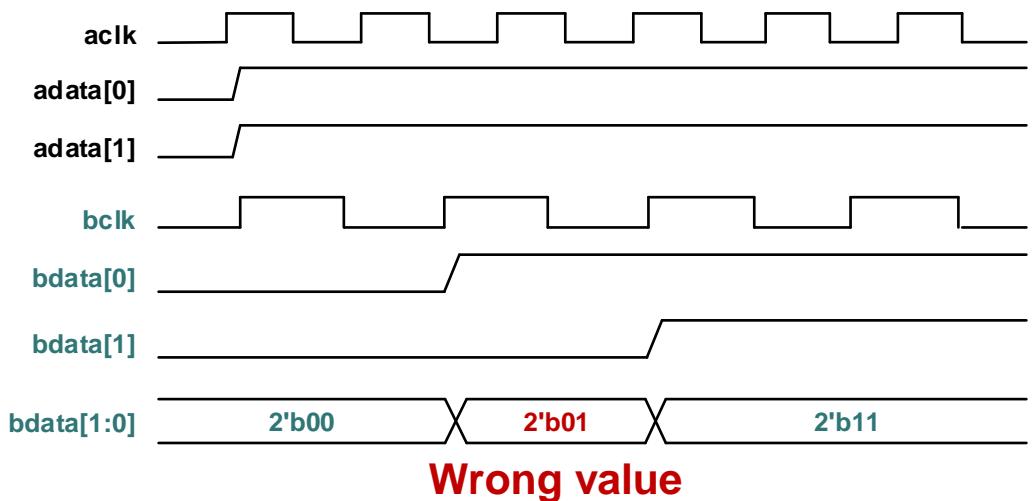
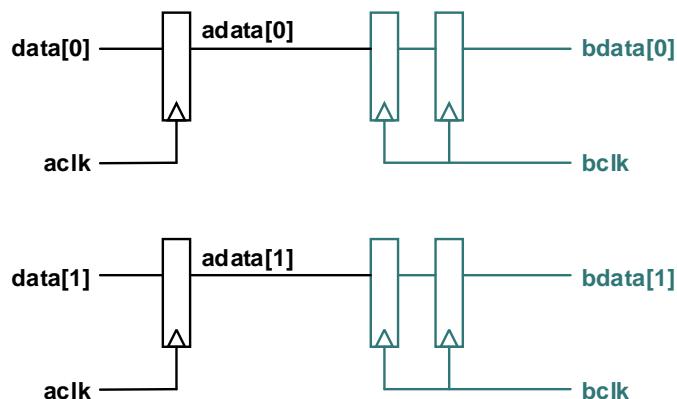
- Apply an additional flip-flop to recover the metastable signal
- Stable signal can be obtained for following logics

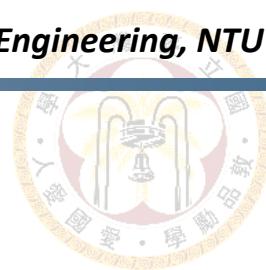




## CDC Issue 2: Multi-Bit Transition

- 2-Flop synchronizers may propagate data with varied latencies
  - Due to the unpredictable behavior of metastability
- Inconsistent values can be captured and errors occur



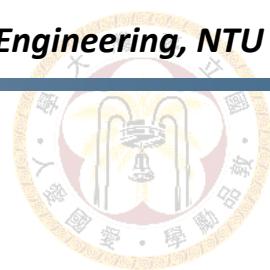


# Solution: Gray Code

- For a counter, avoid changing multiple bits simultaneously
- Gray code ensures only one bit changes at a time

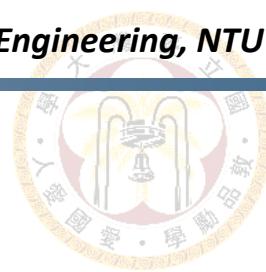
Decimal	Binary	Gray Code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Only one bit changes



# Other Solutions

- Pulse synchronizer
- Asynchronous FIFO
- More information for CDC
  - <https://zhuanlan.zhihu.com/p/141663587>



# Summary

- **The importance of timing analysis**
- **DTA vs. STA**
- **Steps of STA**
  - Break the design into timing paths
  - Calculate AT and RT
  - Calculate the slack
- **STA environment**
- **Special timing paths**
  - False path
  - Multicycle path
- **CDC and metastability**