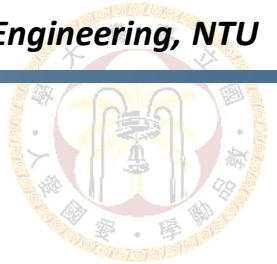
Computer-Aided VLSI System Design

Chapter 6. Static Timing Analysis (STA)

Lecturer: Chun-Wei Chang

Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

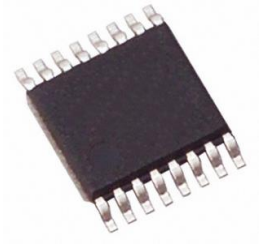


Outline

- Introduction
- Static Timing Analysis
- Environment Setting
- Special Timing Paths

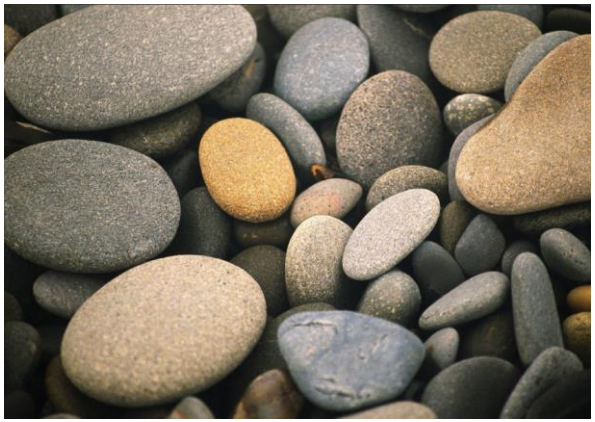
Timing Analysis

- **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**



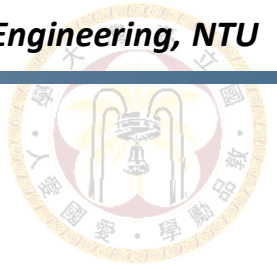
Timing correct

Timing violation



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



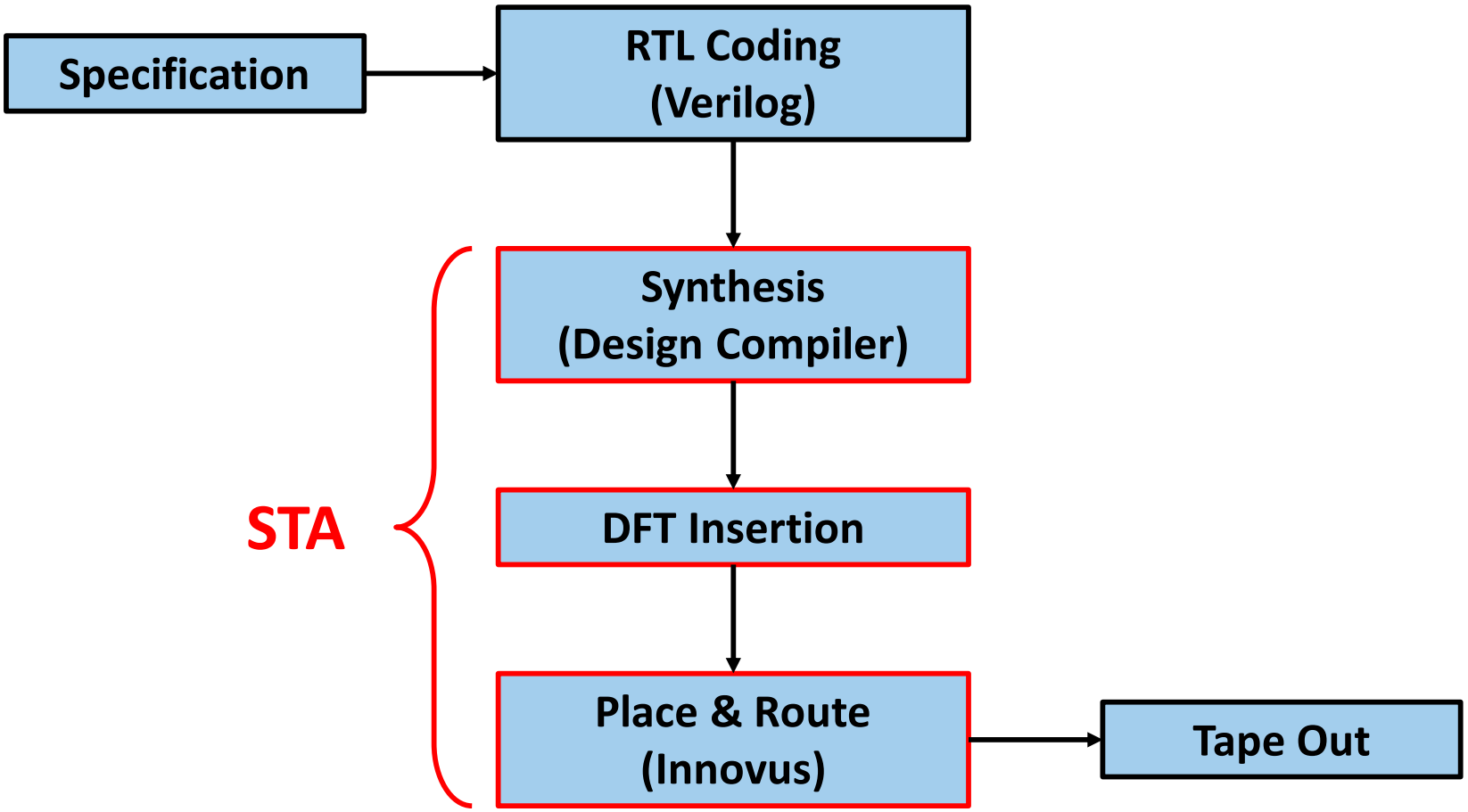
Dynamic Timing Analysis (DTA)

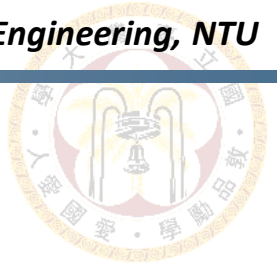
- **Also called Testbench Simulation**
  - Test vectors are applied during the simulation time
  - Simulator calculates both the logic values and delays
- **Disadvantages**
  - Virtually impossible to do exhaustive analysis
    - Vector creation takes too long
    - Incomplete timing coverage
  - Hard 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



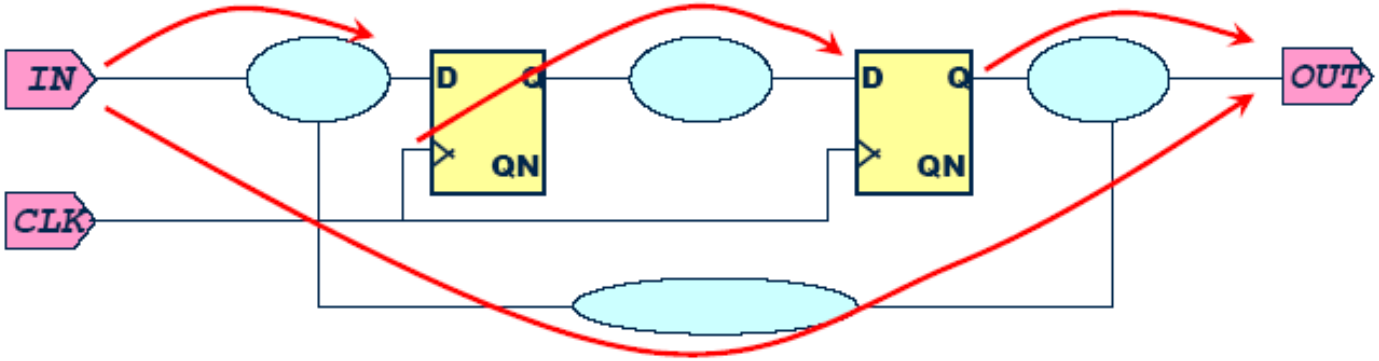


Outline

- Introduction
- **Static Timing Analysis**
- Environment Setting
- Special Timing Paths

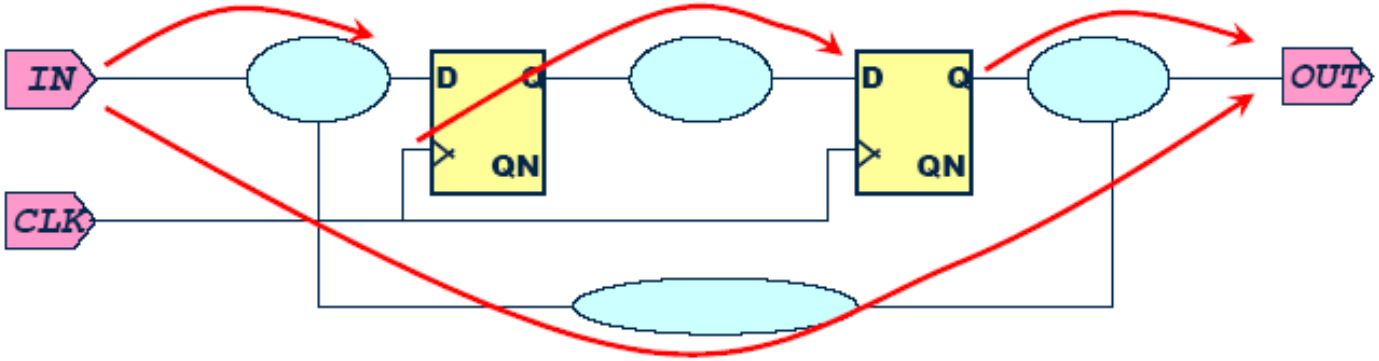
Steps of STA

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

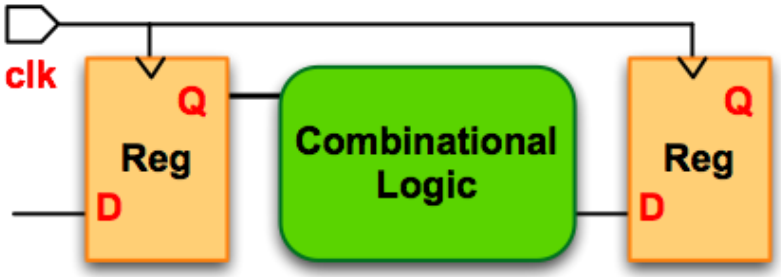


Steps of STA

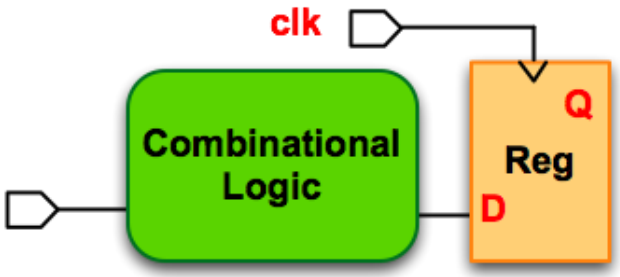
1. **Break the design into a set of timing paths**
2. Calculate arrival time and required time of each path
3. Calculate the slack of each path



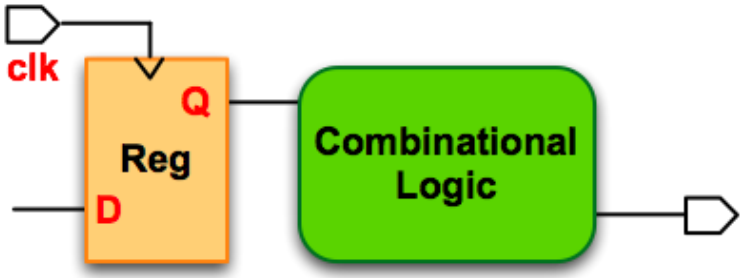
Four Types of Timing Paths



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

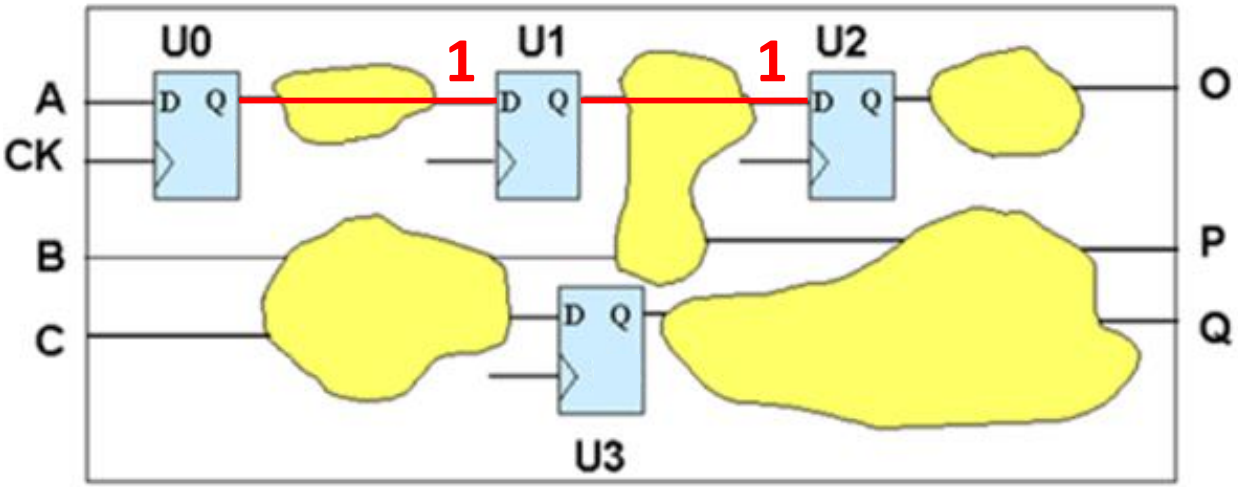


Register-to-output (reg2out)
Start Point: clock pin of sequential device
End point: primary output port

Input-to-output (in2out)
Start Point: primary input port
End point: primary output port

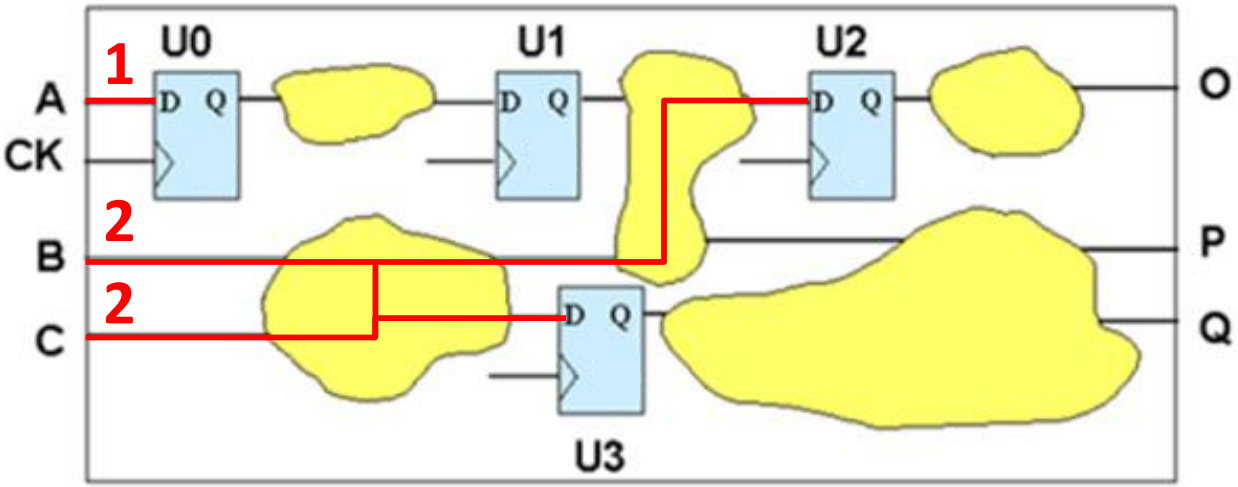
Example: Register-to-Register Paths

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



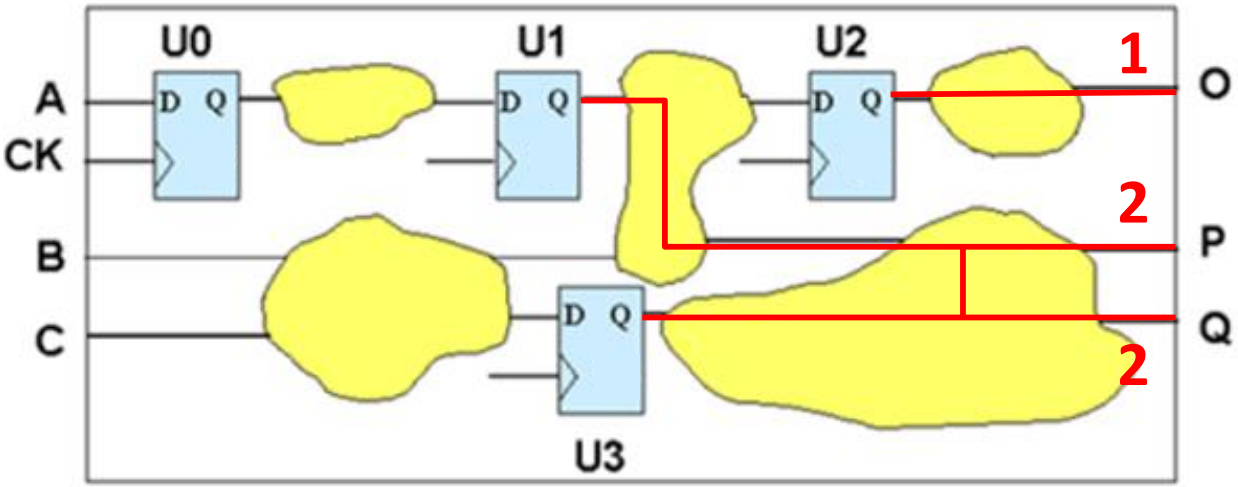
Example: Input-to-Register Paths

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



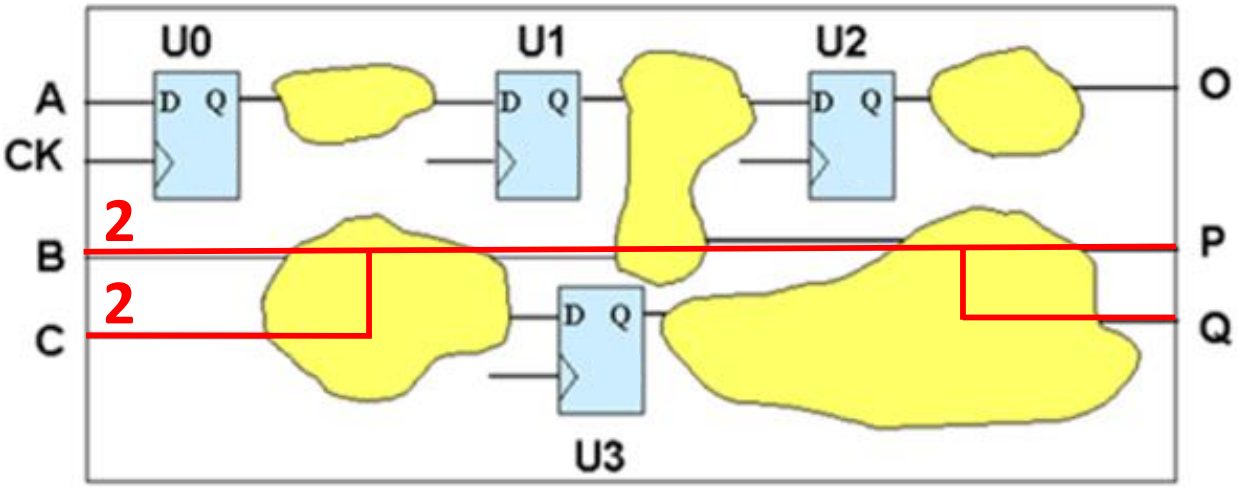
Example: Register-to-Output Paths

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



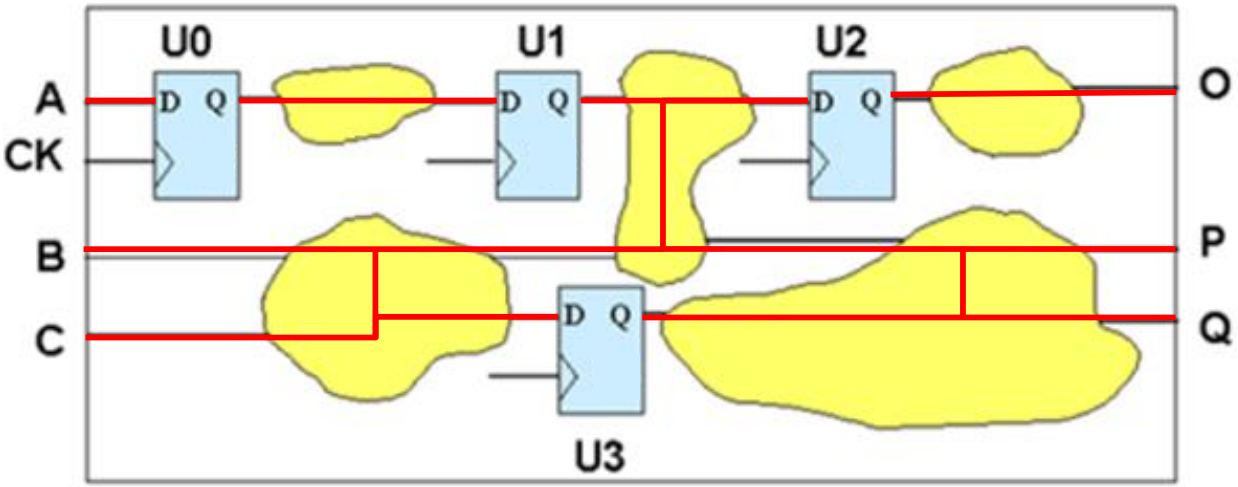
Example: Input-to-Output Paths

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



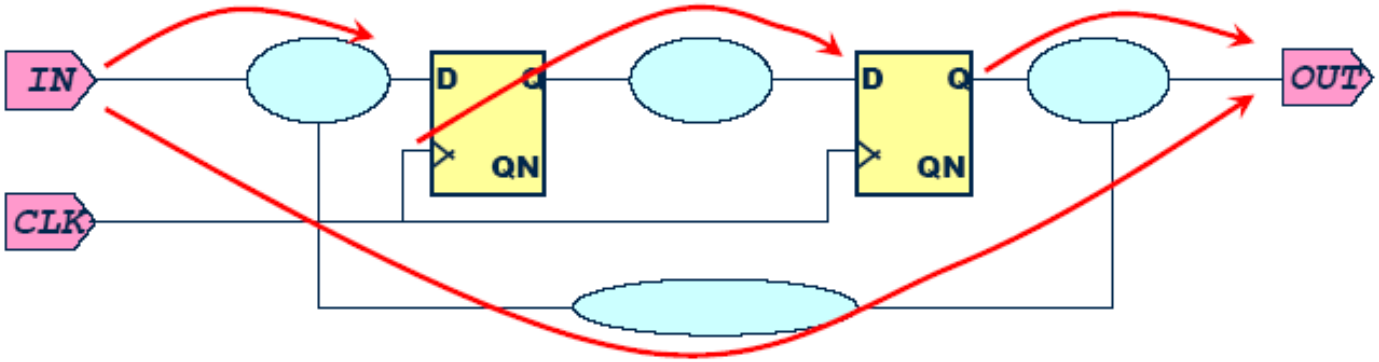
Example: Timing Paths

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

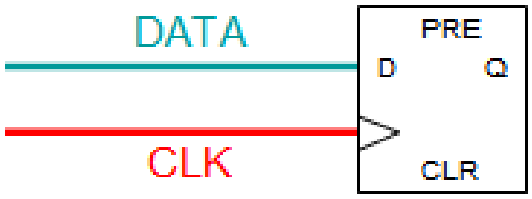


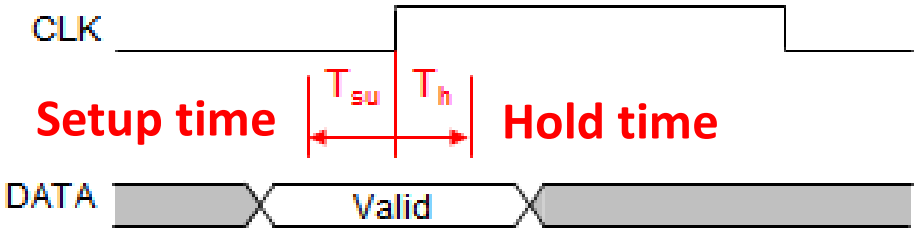
Steps of STA

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



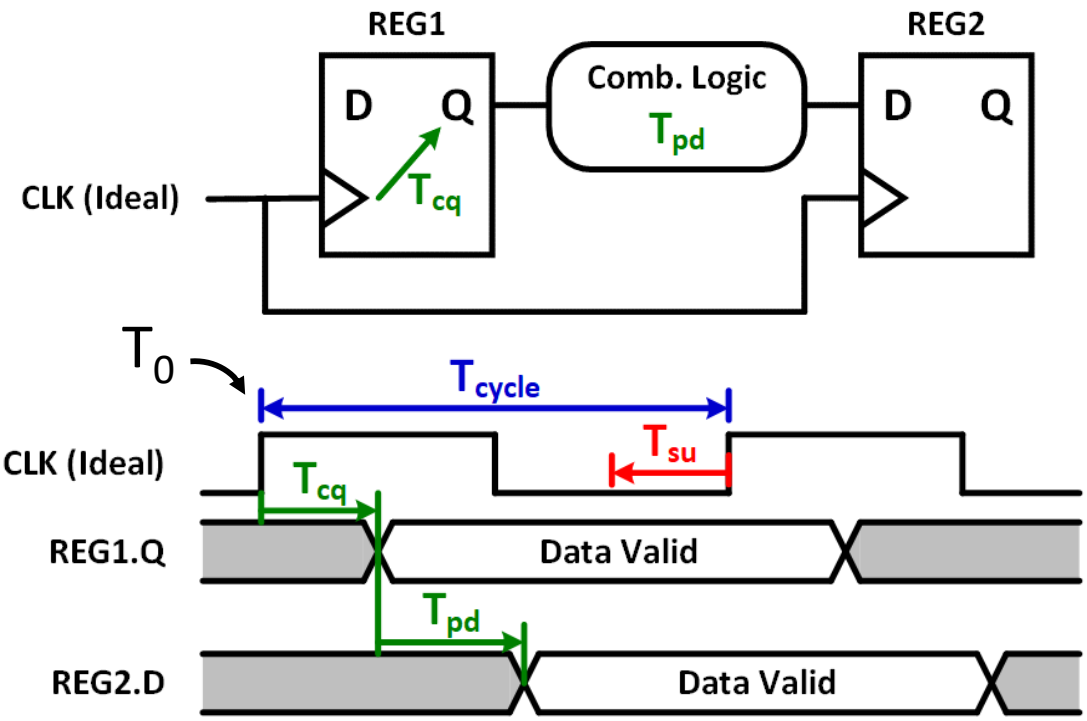
Setup Time & Hold Time





- **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

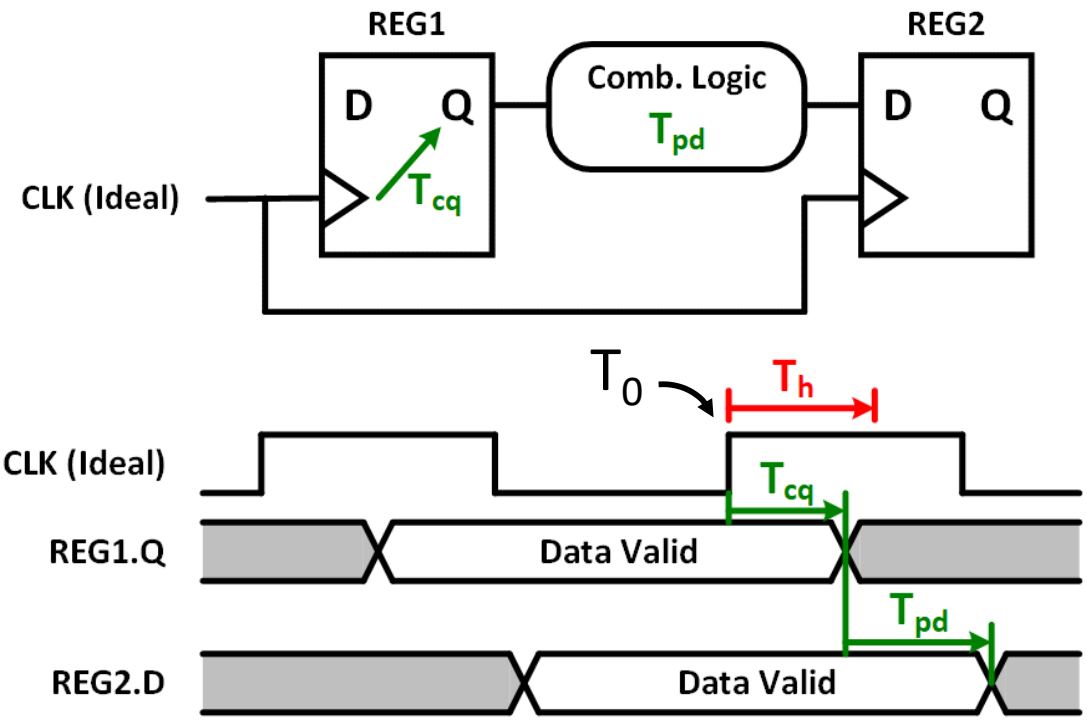
Setup Check (Ideal CLK)



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

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

Hold Check (Ideal CLK)

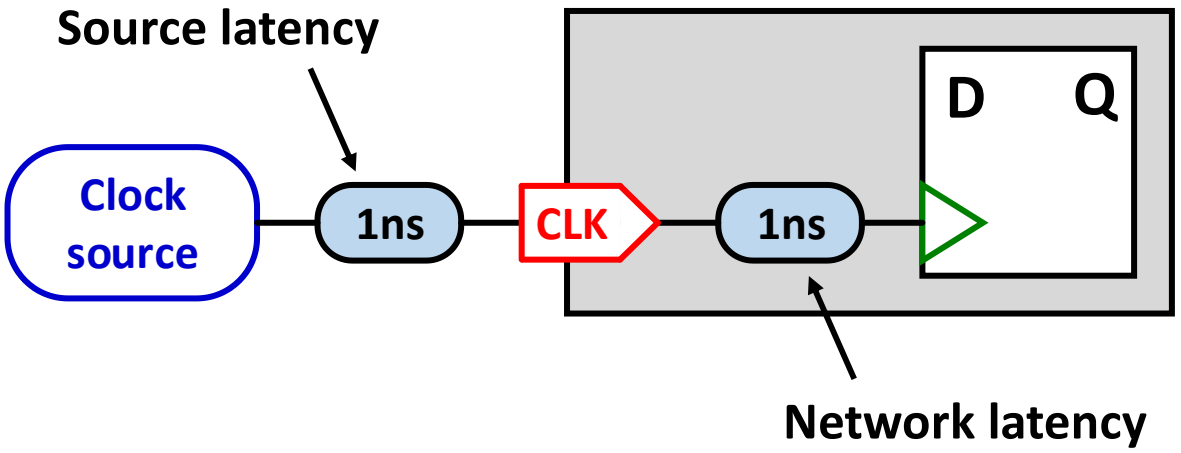


Arrival time (AT): $T_0 + T_{cq} + T_{pd}$

Required time (RT): $T_0 + T_h$

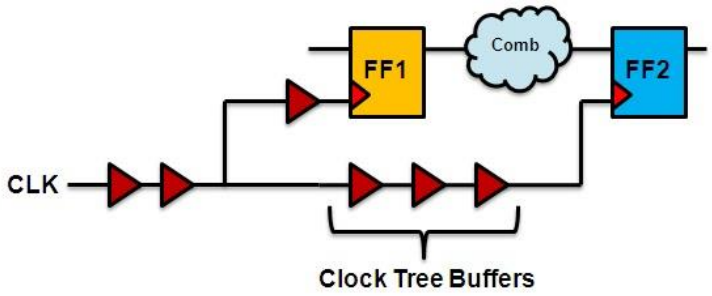
Clock Latency

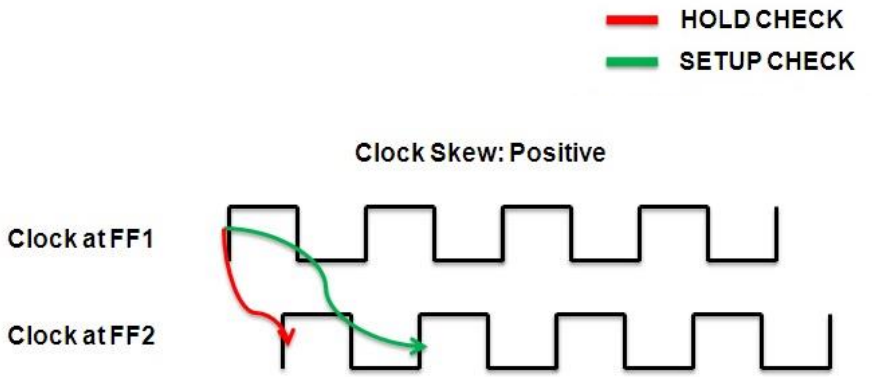
- **Source latency**
  - Delay from the clock source to the clock definition point (CLK)
- **Network latency**
  - Delay from the clock definition point (CLK) to the clock pin of a sequential cell

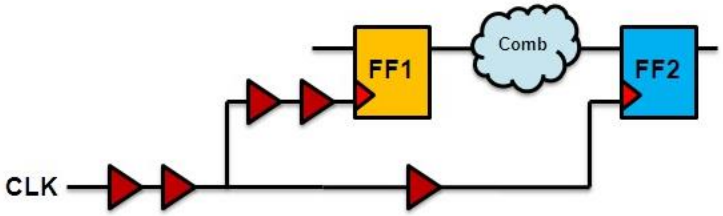


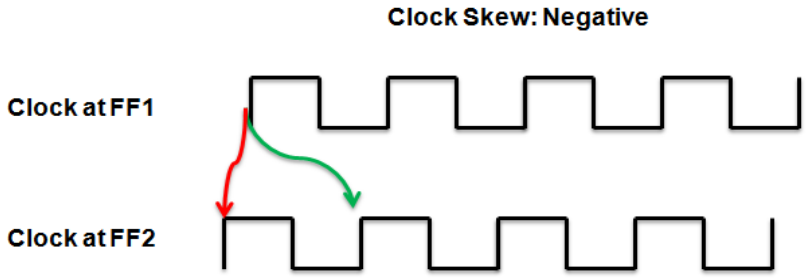
Clock Uncertainty: Skew

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



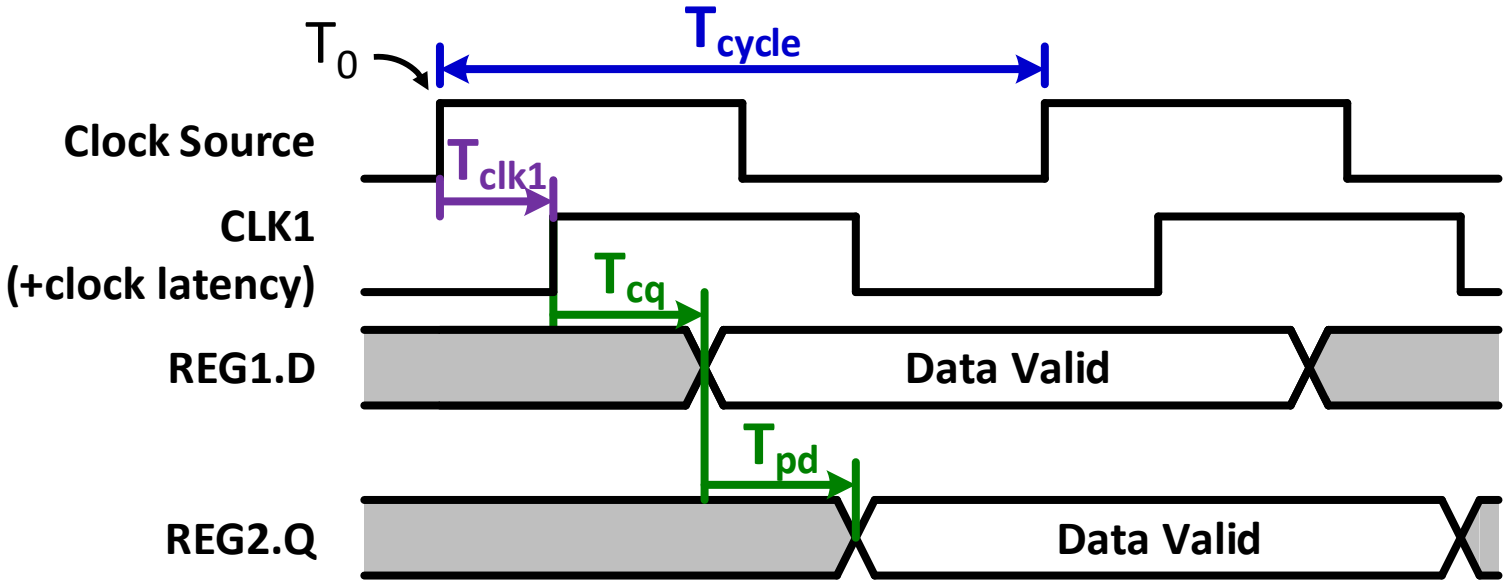






Setup Check: Arrival Time

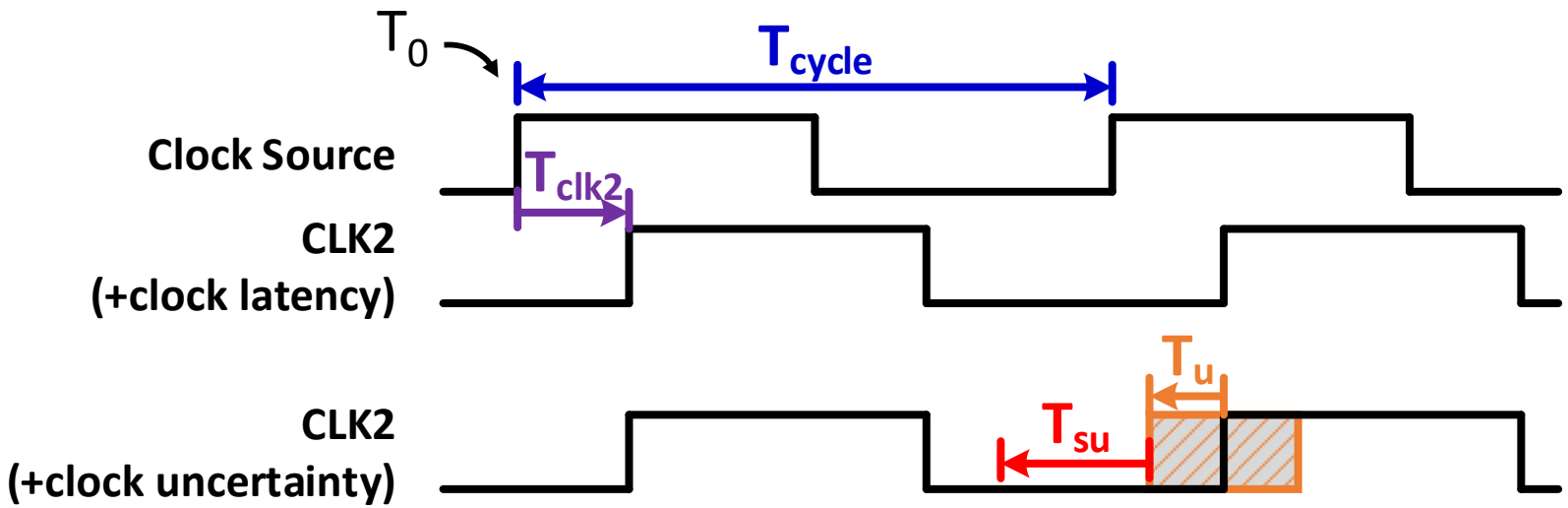
- Consider the same circuits in p. 19



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

Setup Check: Required Time

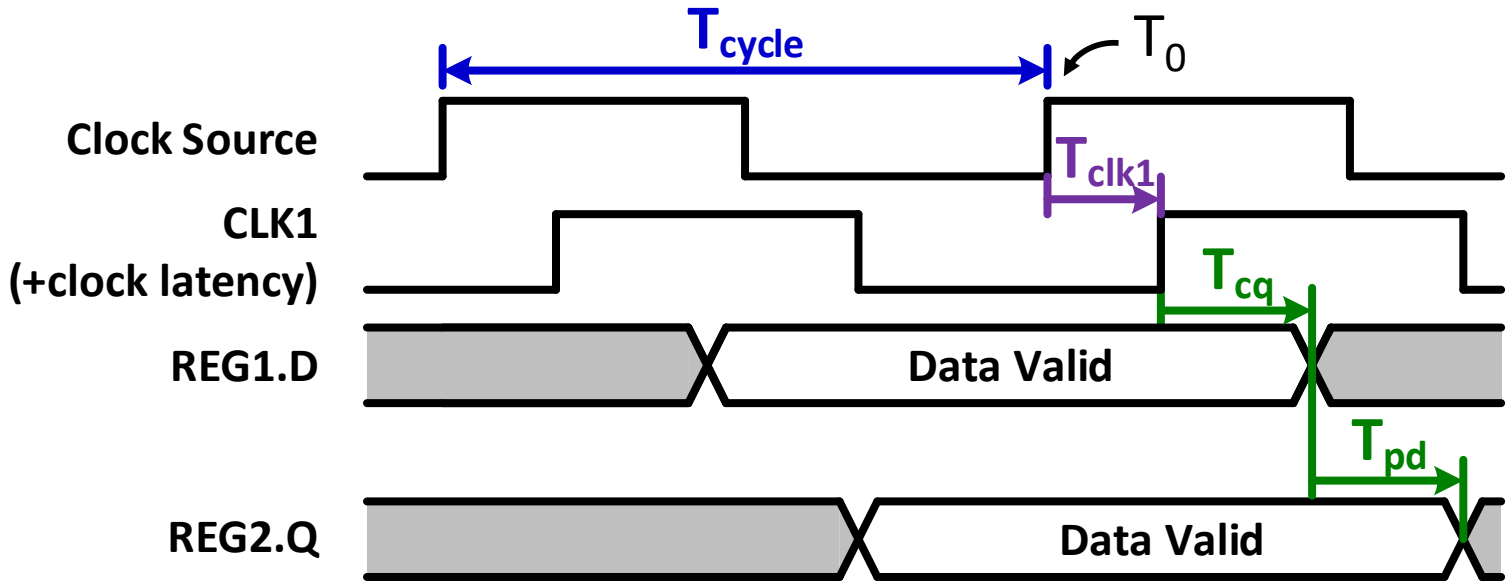
- Consider the same circuits in p. 19



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

Hold Check: Arrival Time

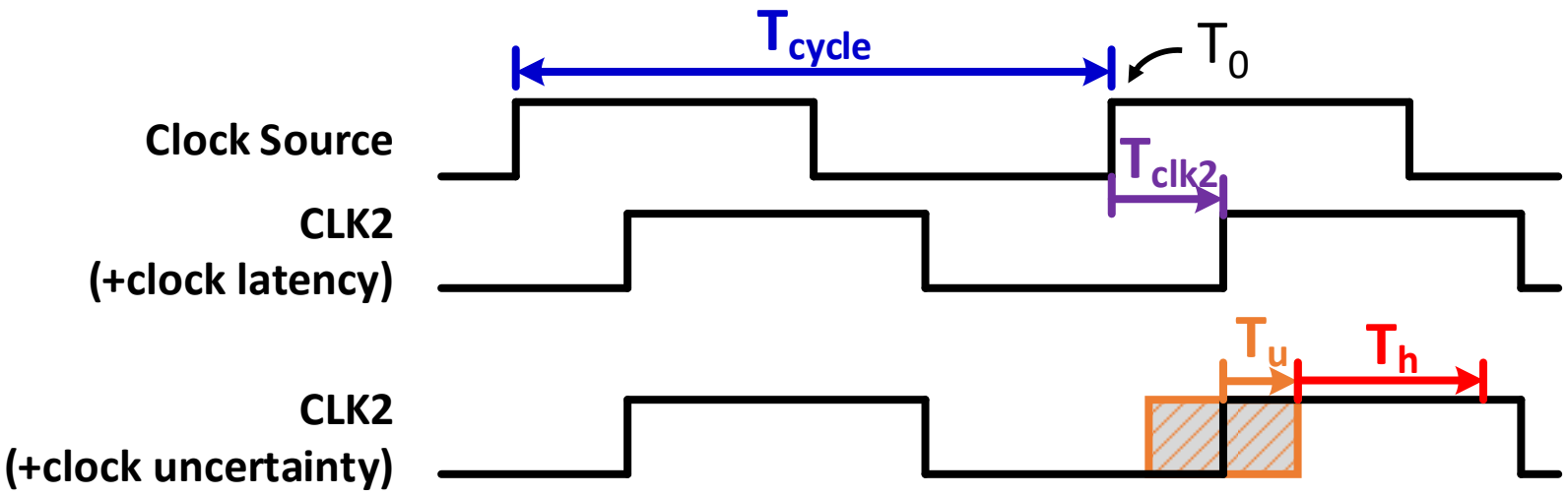
- Consider the same circuits in p. 20



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

Hold Check: Required Time

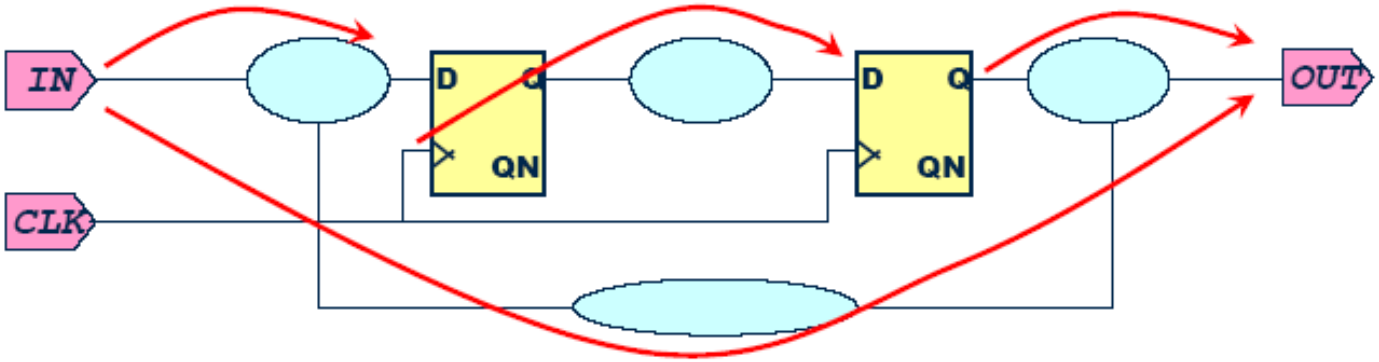
- Consider the same circuits in p. 20

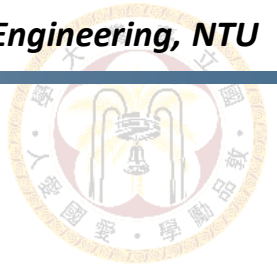


Required time (RT): $T_0 + T_{clk2} + T_u + T_h$

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**

Setup slack = RT - AT

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

Hold slack = AT - RT

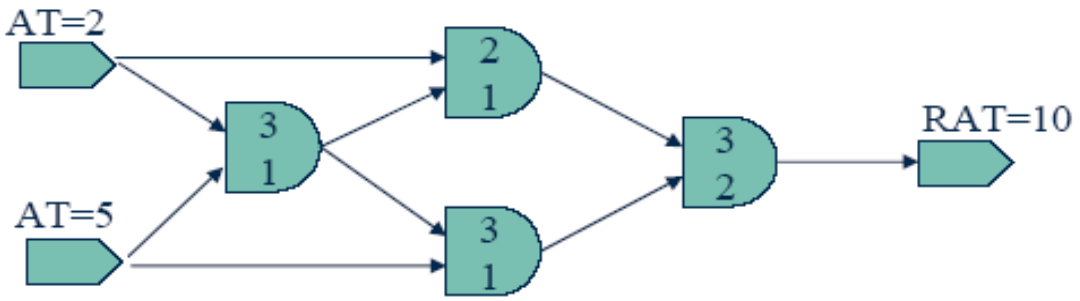
- **Slack ≥ 0 → timing constraints met**
- **Slack < 0 → timing constraints violated**

Critical Path

- The path with the worst negative slack (WNS)

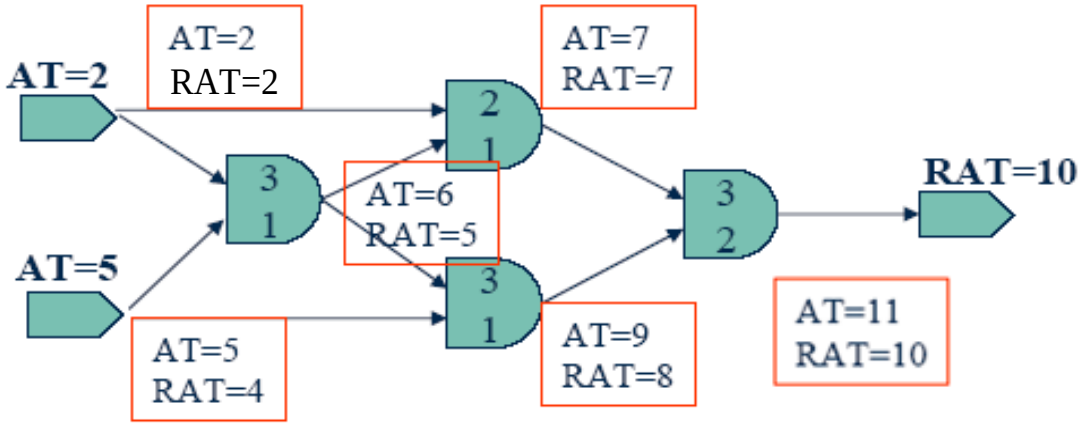
```
********************************
Report : timing  -path full  -delay max
                 -max_paths 1
Design : MUL
********************************
                Point                    Incr     Path
  ---------------------------------------------------------
-
clock CLK (rise edge)                    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 (VIOLATED)                                 -1.00
```

Path-based vs. Block-based STA



Path-based:

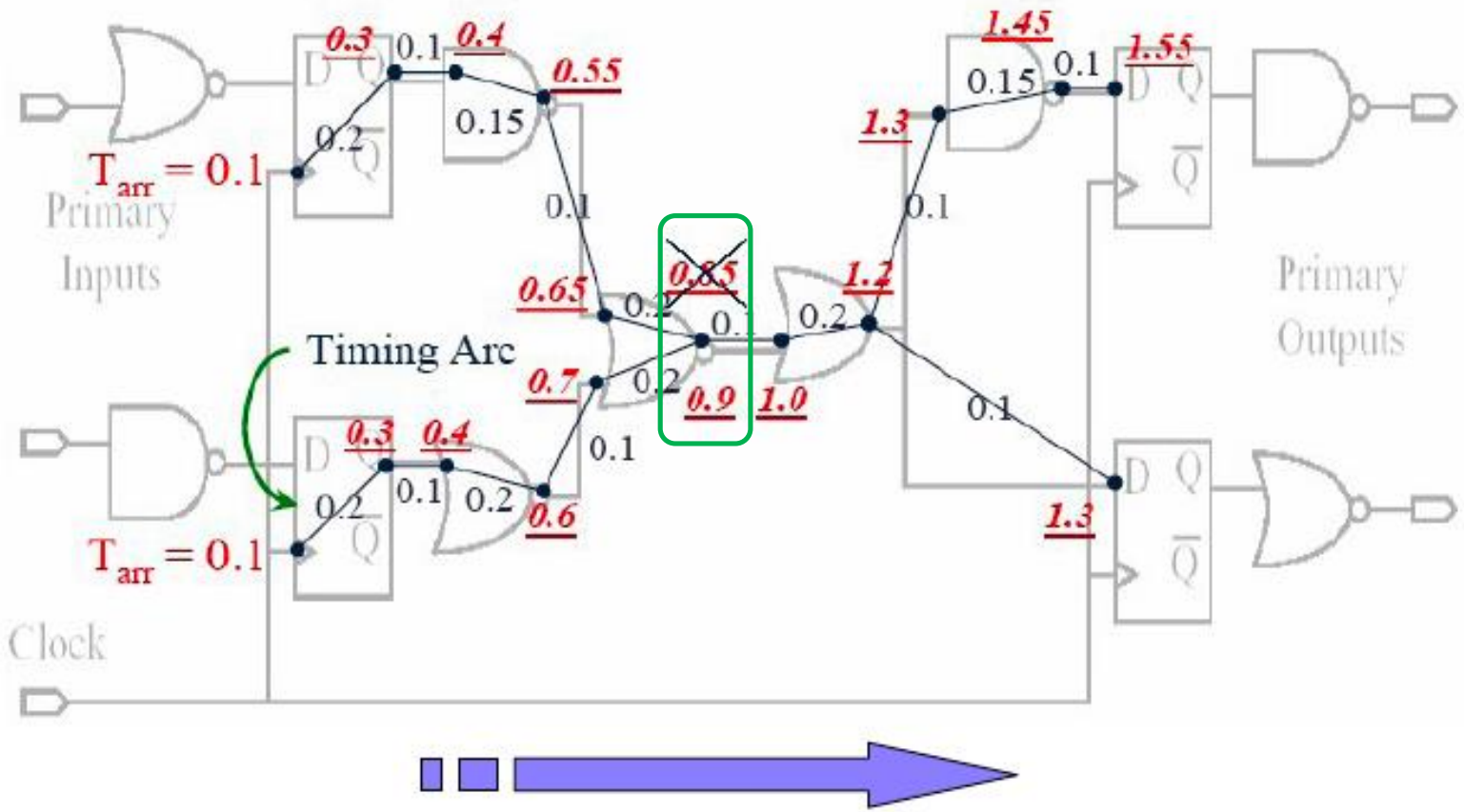
2+2+3 = 7 (OK)
2+3+1+3 = 9 (OK)
2+3+3+2 = 10 (OK)
5+1+1+3 = 10 (OK)
5+1+3+2 = 11 (Fail)
5+1+2 = 8 (OK)



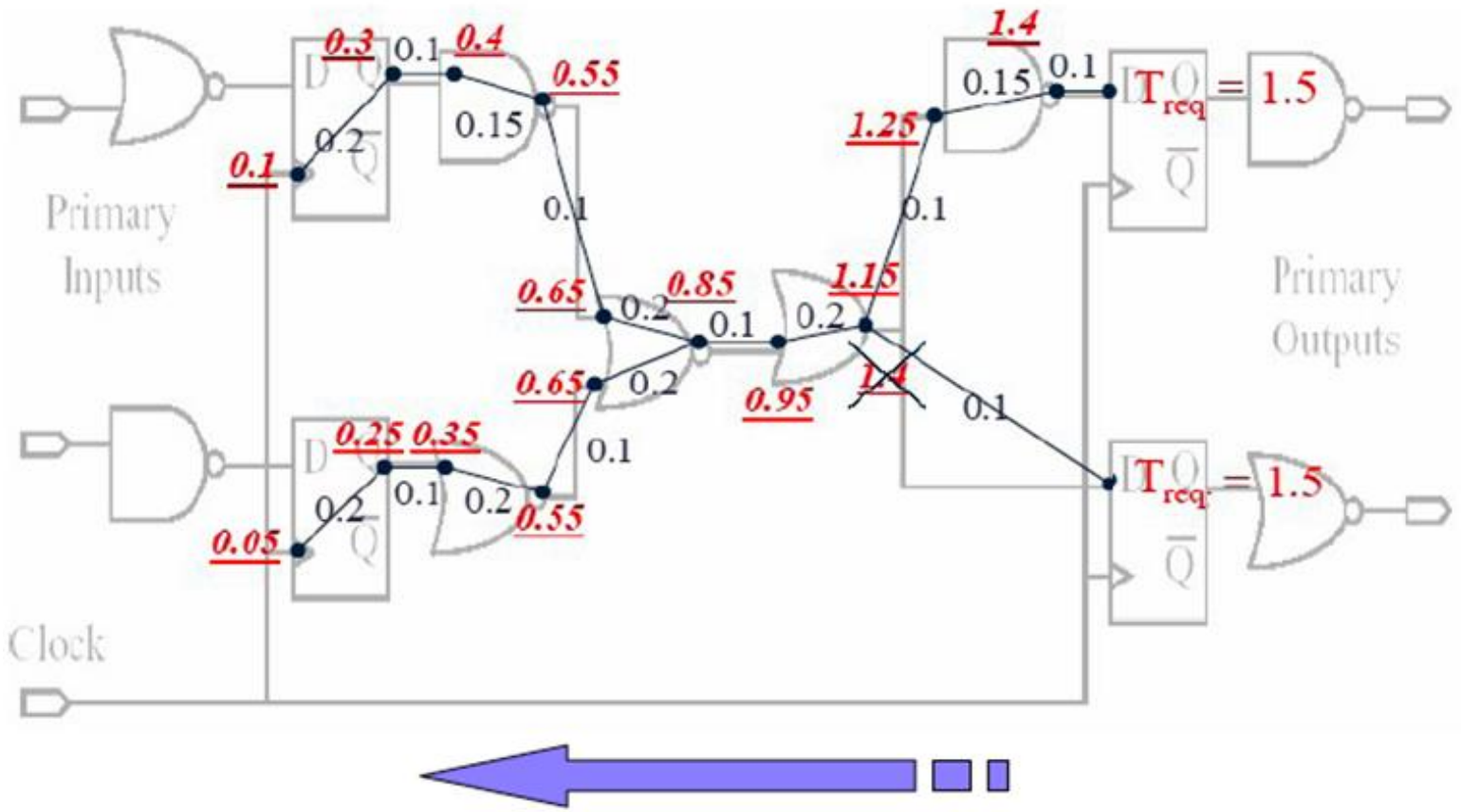
Block-based:

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

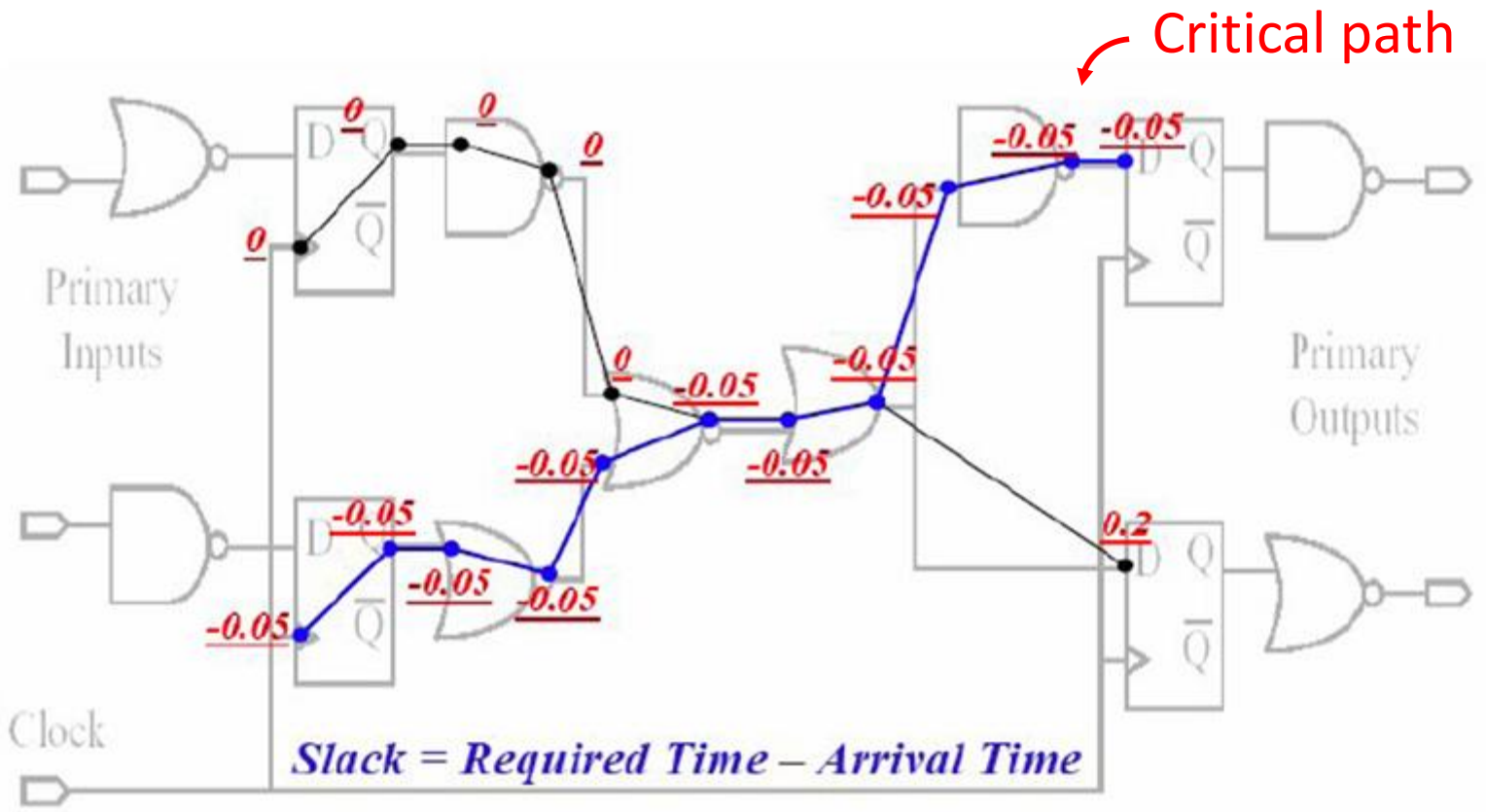
Block-based STA: Arrival Time



Block-based STA: Required Time



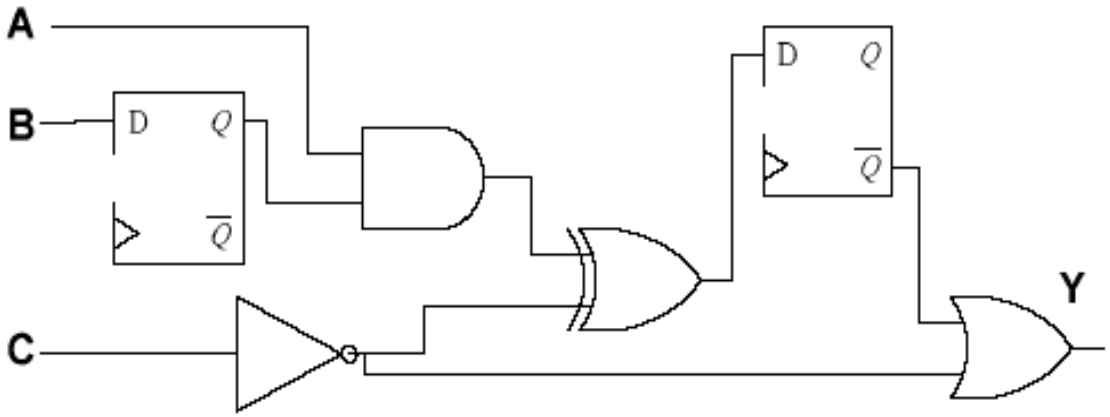
Setup Slack Graph



Slack ≥ 0 → timing constraints met

Slack < 0 → 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**

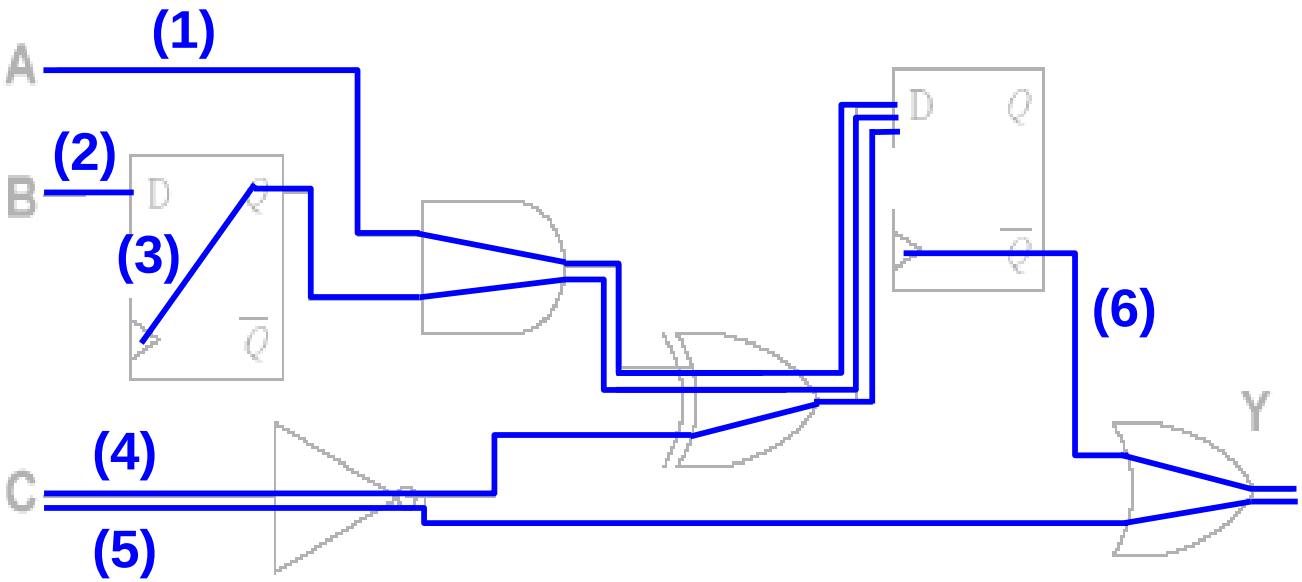
Clock period T_{cycle}	Source latency T_{clks}	Network latency T_{clkn}	Uncertainty T_u
14ns	2ns	3ns	1ns

- **IO constraints**

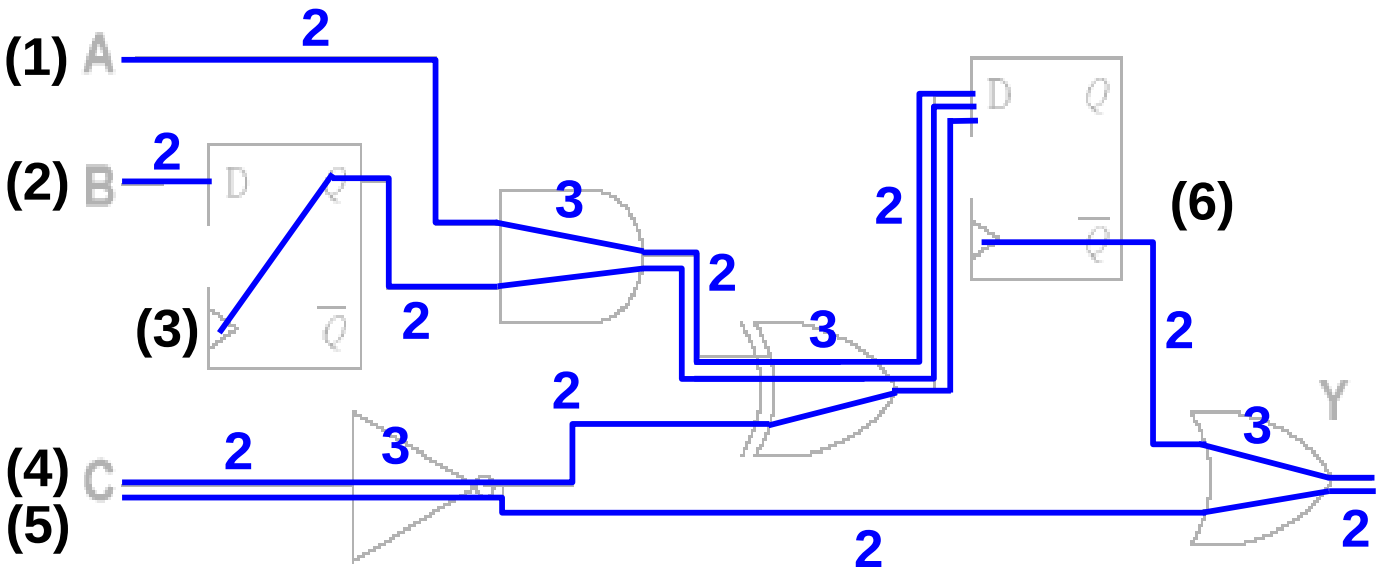
	D_A	D_B	D_C	D_Y
Delay	1ns	1ns	1ns	3ns

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

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

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

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

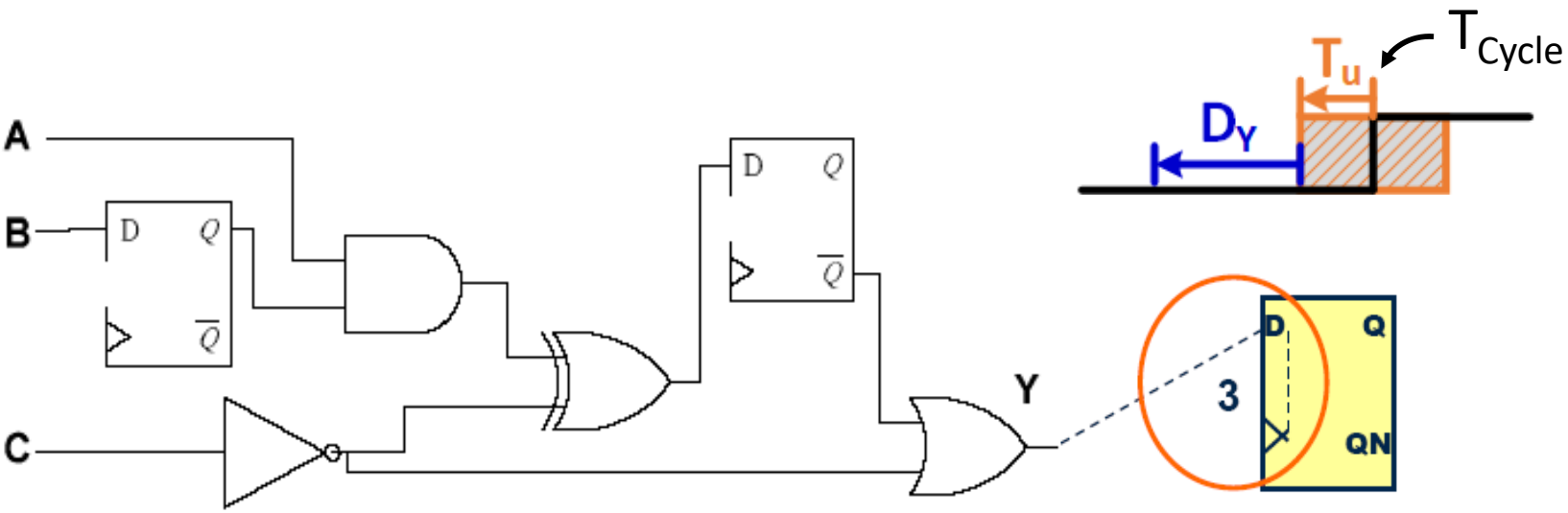
STA Example: Setup RT (Path-based)

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

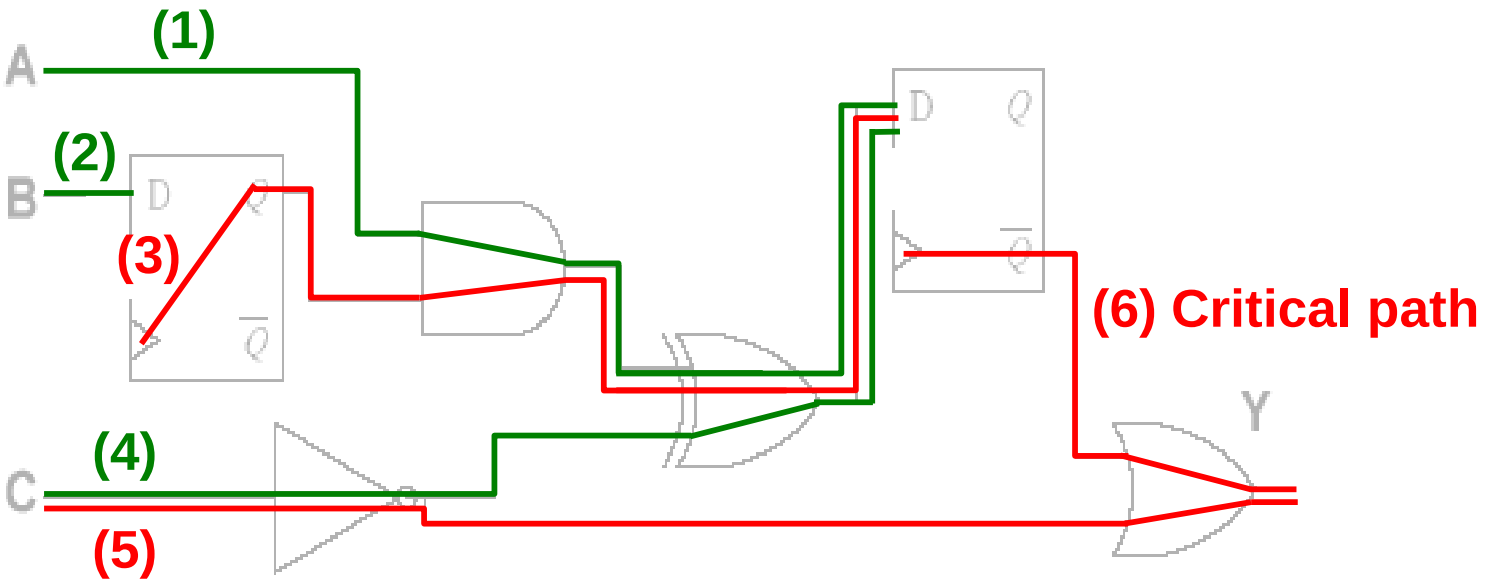
STA Example: Setup RT (Path-based)

- **Timing path (5): input-to-output**
  - RT: $T_{cycle} - T_u - D_Y = 14 - 1 - 3 = 10ns$

- **Timing path (6): register-to-output**
  - RT: $T_{cycle} - T_u - D_Y = 14 - 1 - 3 = 10ns$



STA Example: Setup Slack (Path-based)



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

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

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

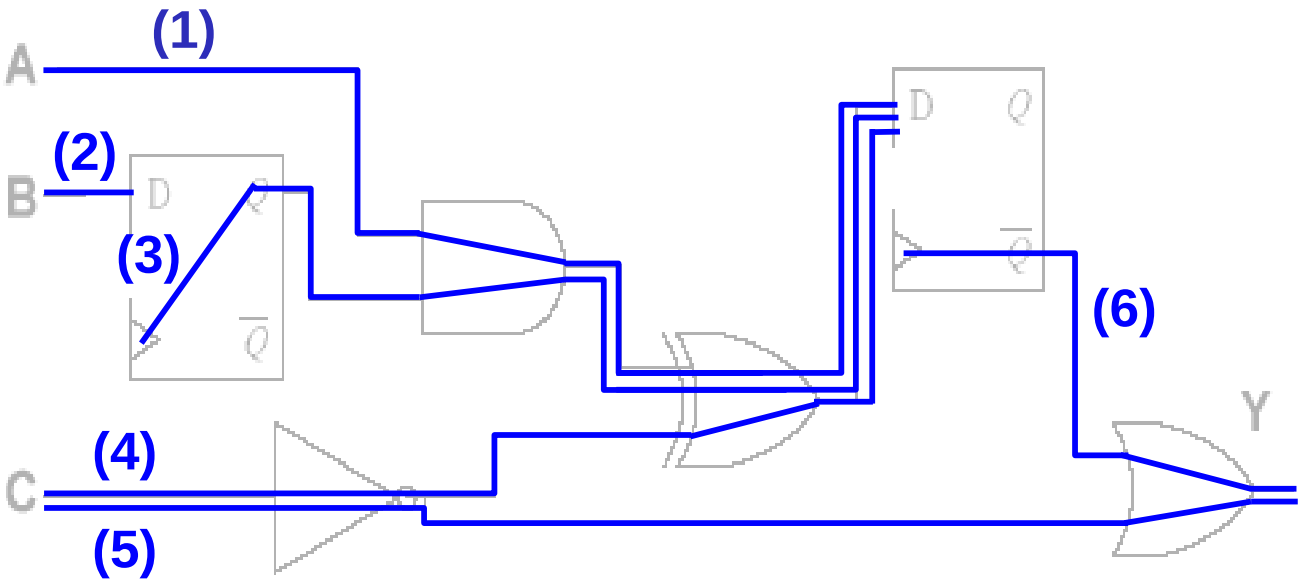
(4) Slack = 17-13 = 4 (met)

(5) Slack = 10-13 = -3 (violated)

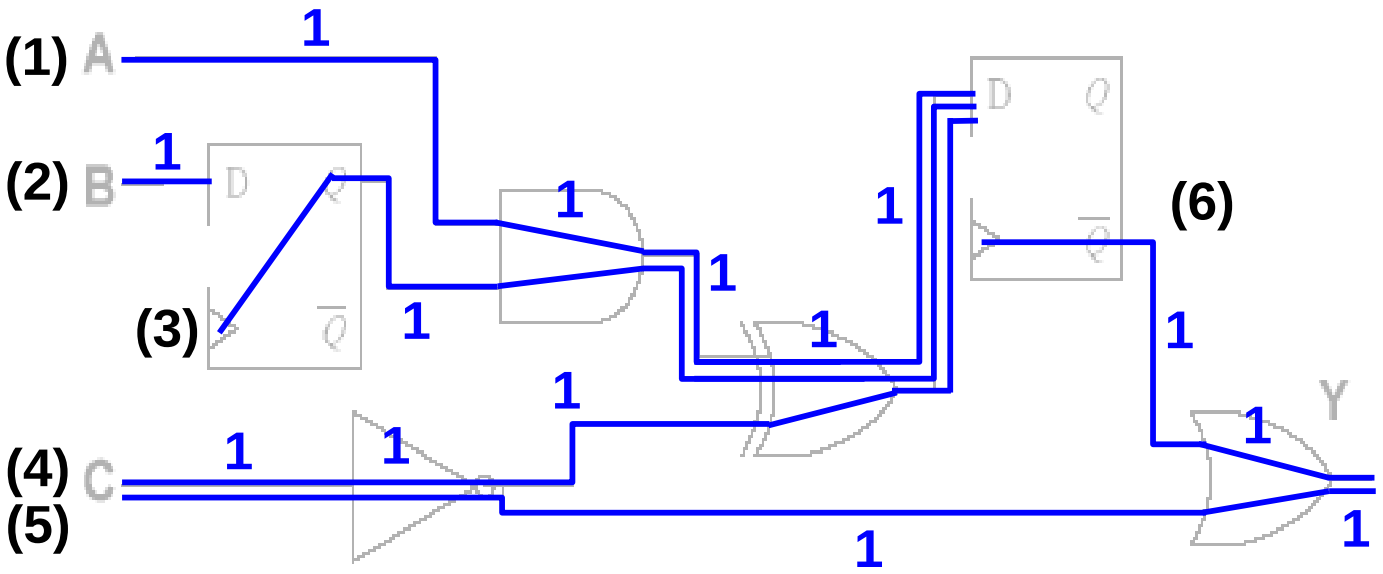
(6) Slack = 10-15 = -5 (violated)

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) $T_{clks}+T_{clkn}+T_{cq}+(1+1+1+1+1) = 13$

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

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

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

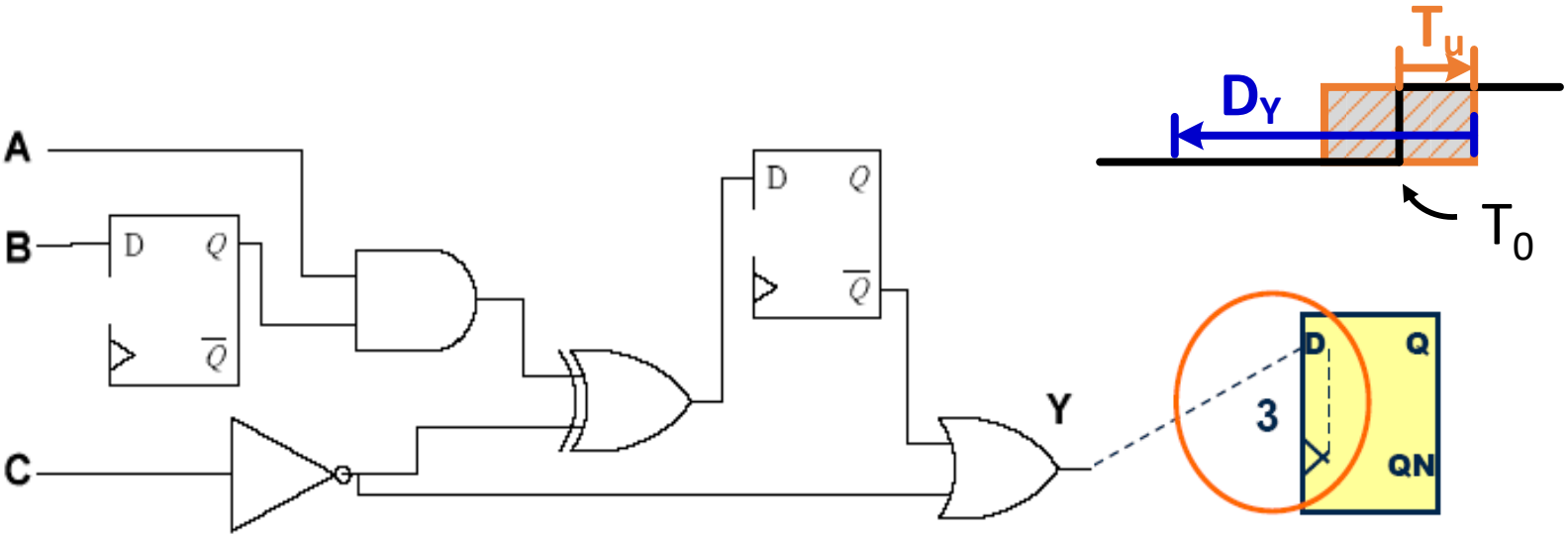
STA Example: Hold RT (Path-based)

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

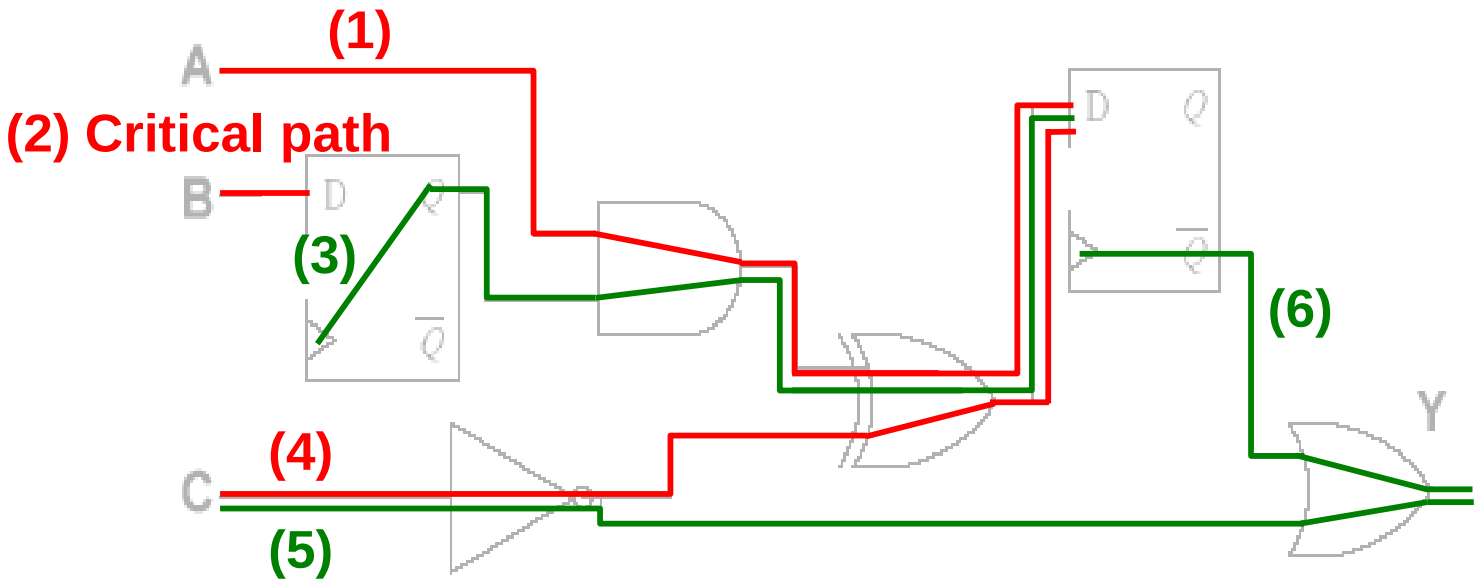
STA Example: Hold RT (Path-based)

- **Timing path (5): input-to-output**
  - RT: $T_u - D_Y = -2ns$

- **Timing path (6): register-to-output**
  - RT: $T_u - D_Y = -2ns$



STA Example: Hold Slack (Path-based)



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

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

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

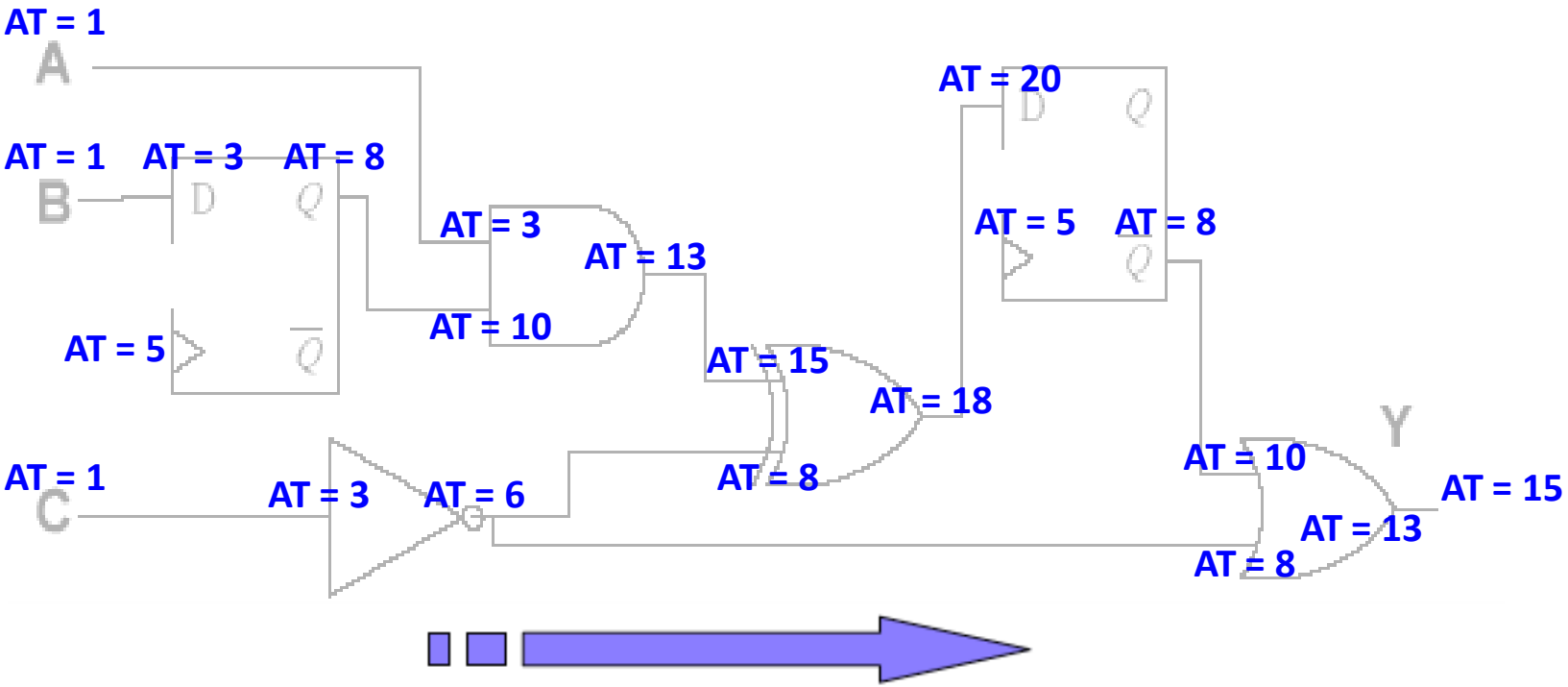
(4) Slack = 6-7 = -1 (violated)

(5) Slack = 6-(-2) = 8 (met)

(6) Slack = 11-(-2) = 13 (met)

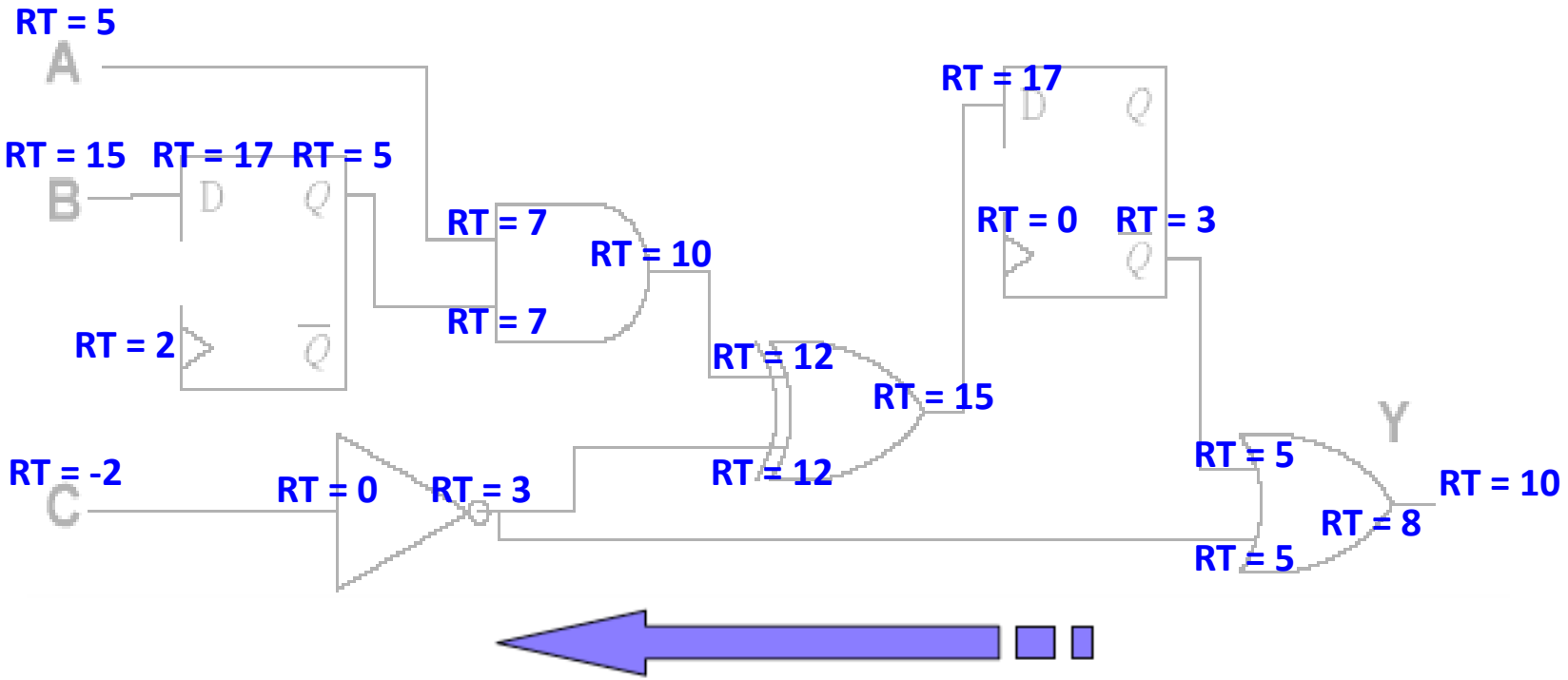
STA Example: Setup AT (Block-based)

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



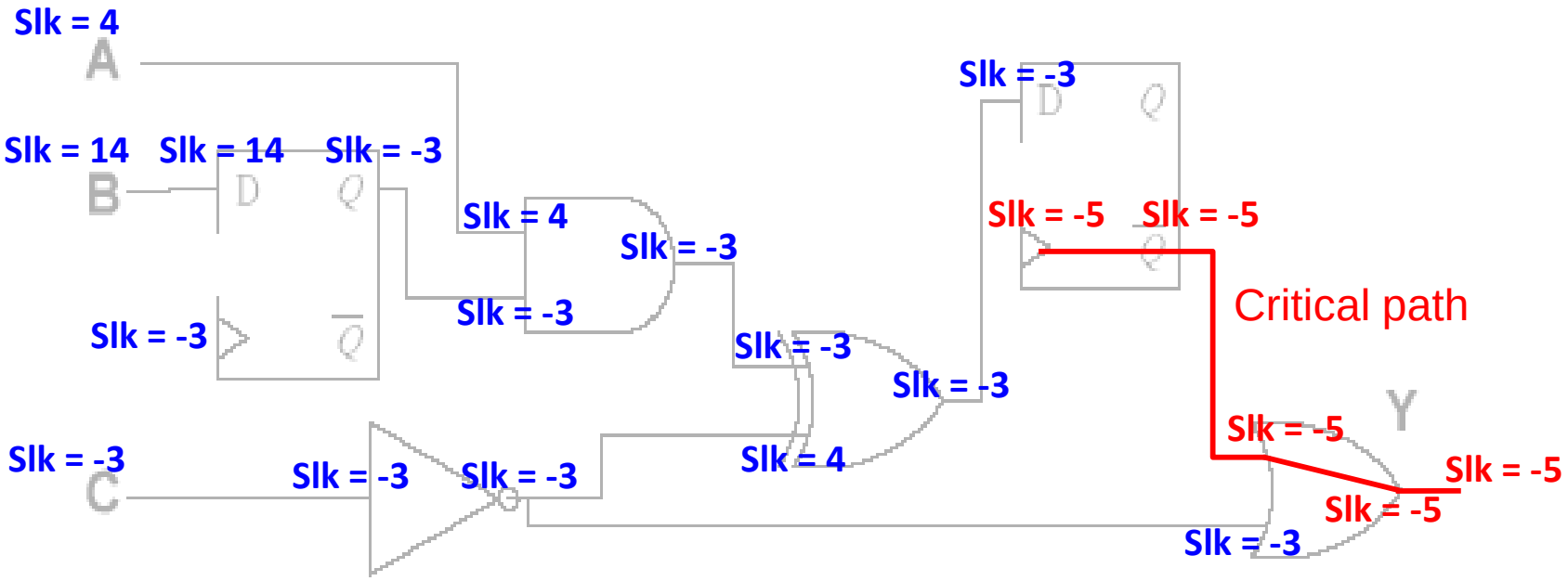
STA Example: Setup RT (Block-based)

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



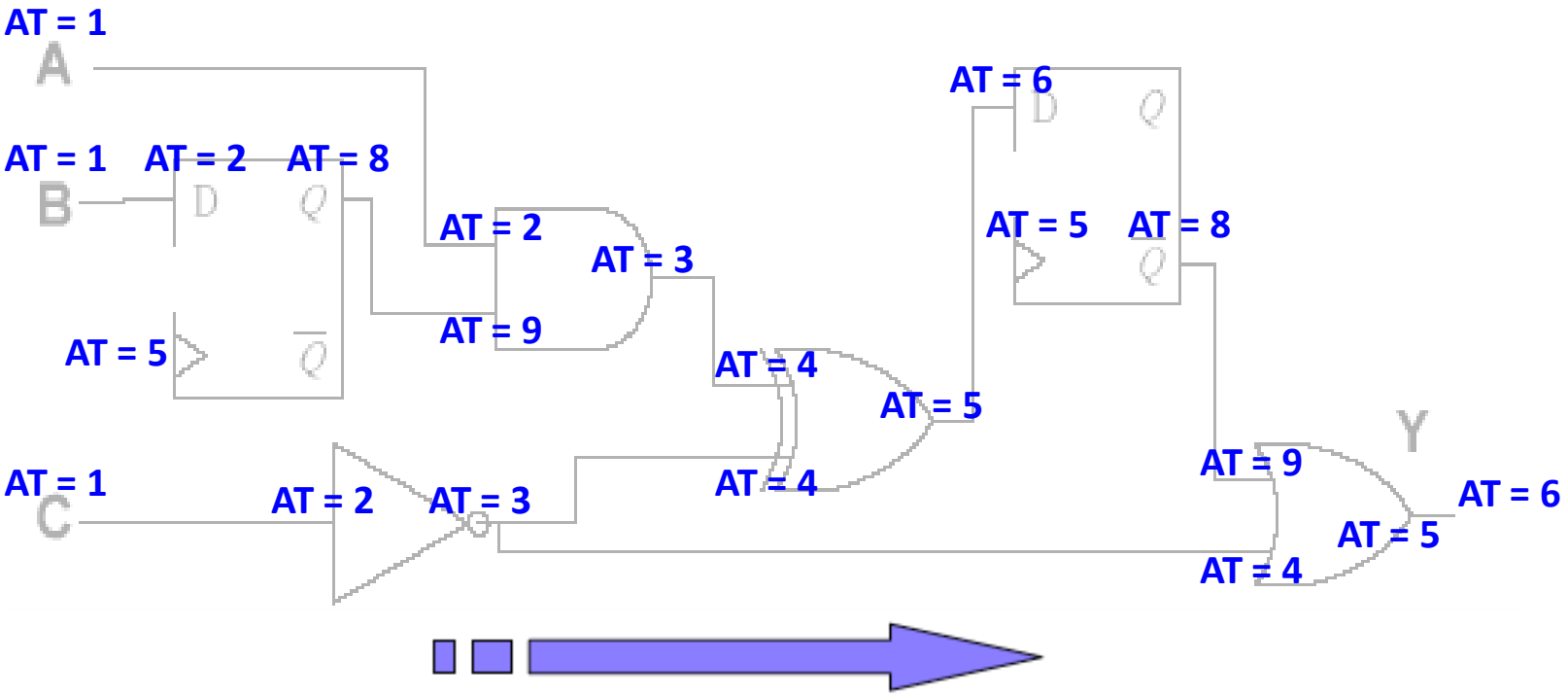
STA Example: Setup Slack (Block-based)

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



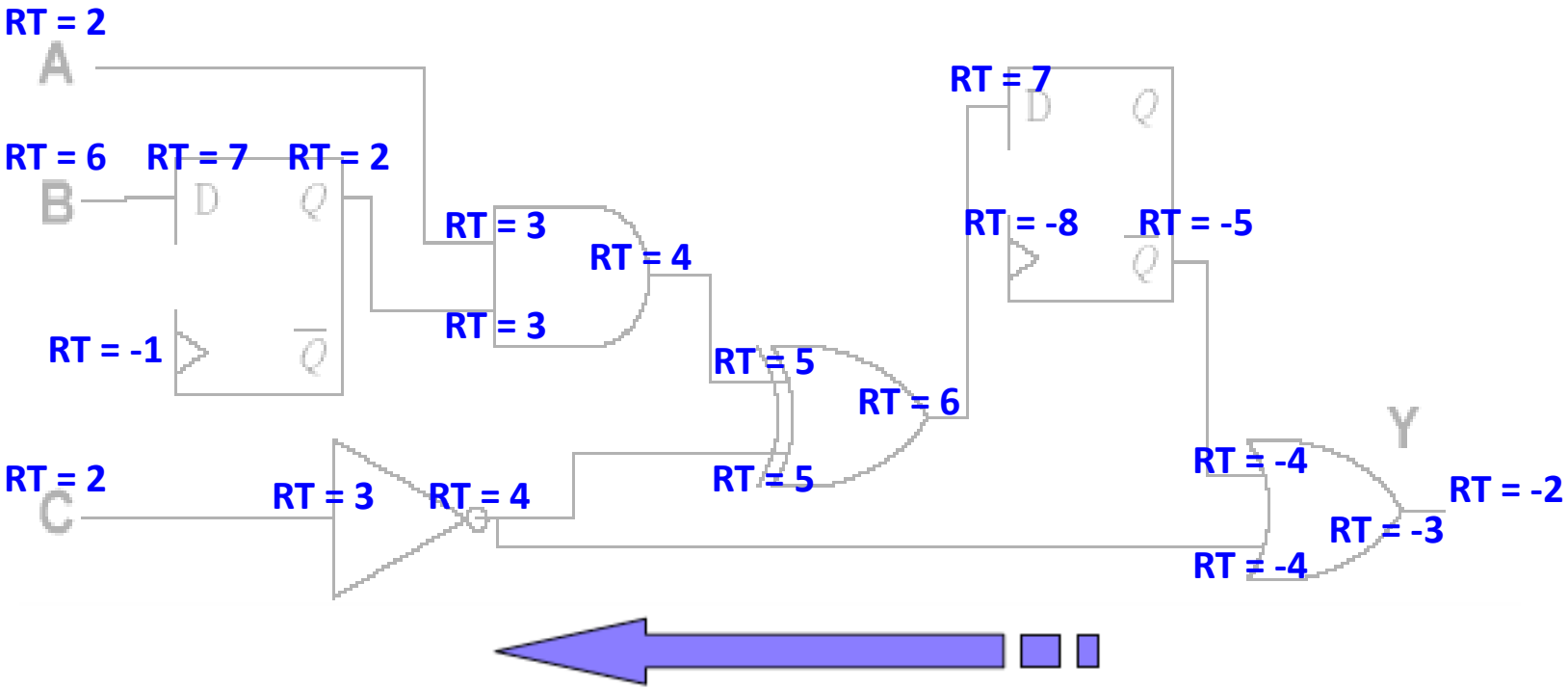
STA Example: Hold AT (Block-based)

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



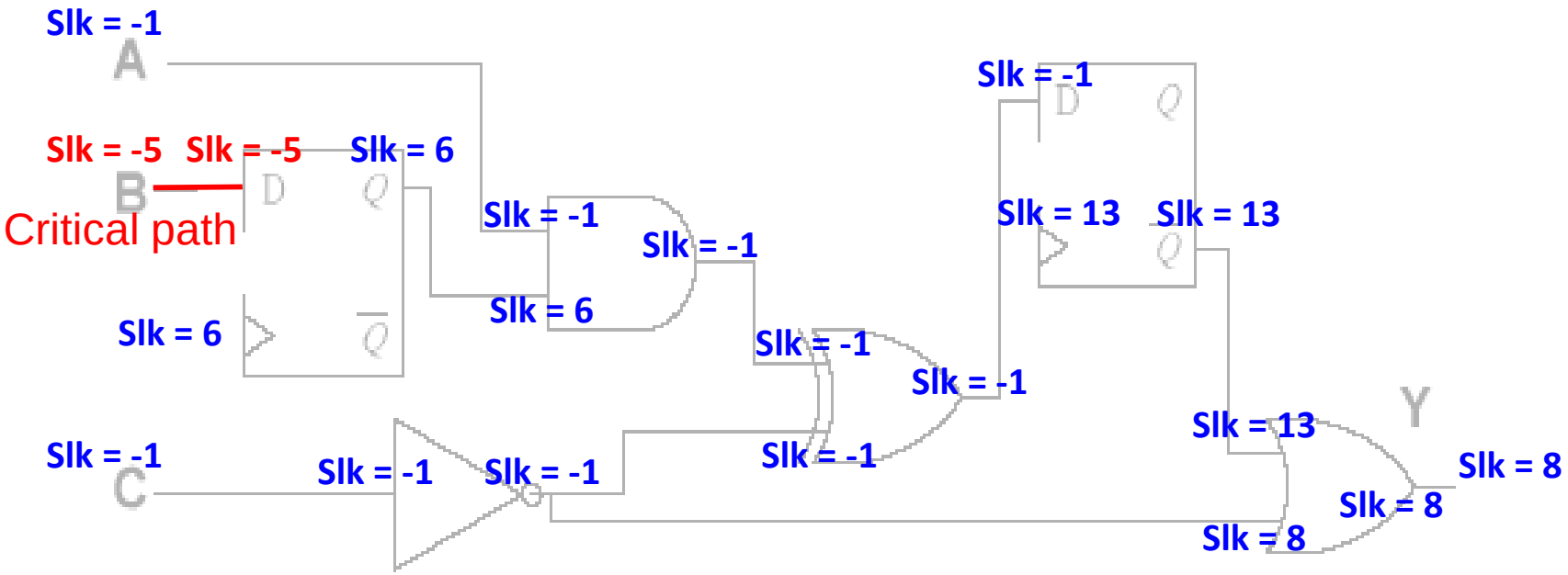
STA Example: Hold RT (Block-based)

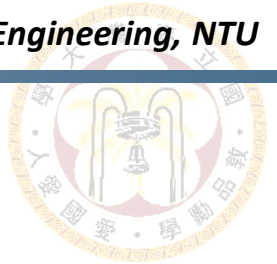
- Calculate the required time of each design element
  - Consider **max** RT at each intersection



STA Example: Hold Slack (Block-based)

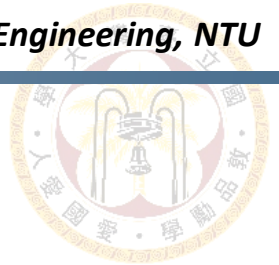
- Calculate the slack (Slk) of each design element
  - AT – RT





Outline

- Introduction
- Static Timing Analysis
- **Environment Setting**
- Special Timing Paths

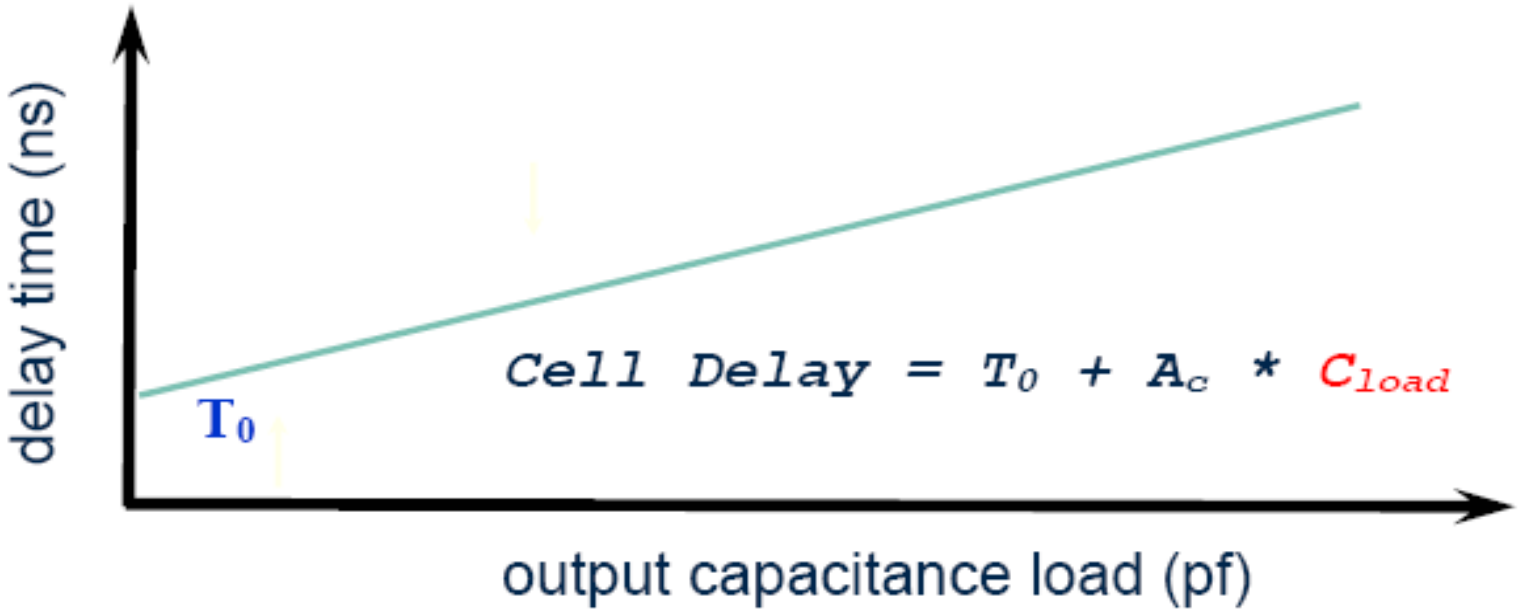


Library Data

- **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

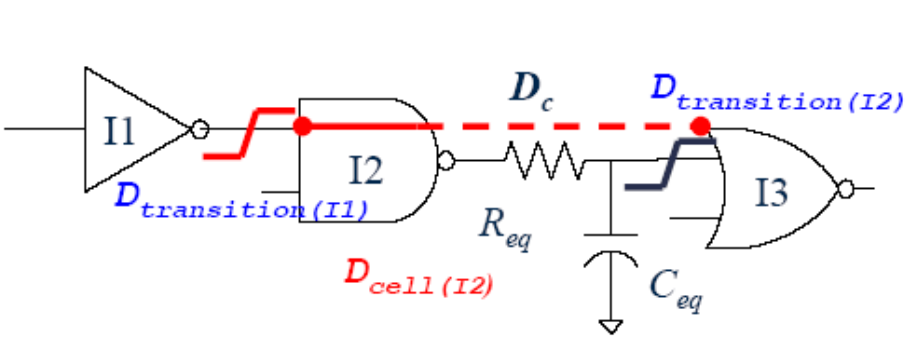


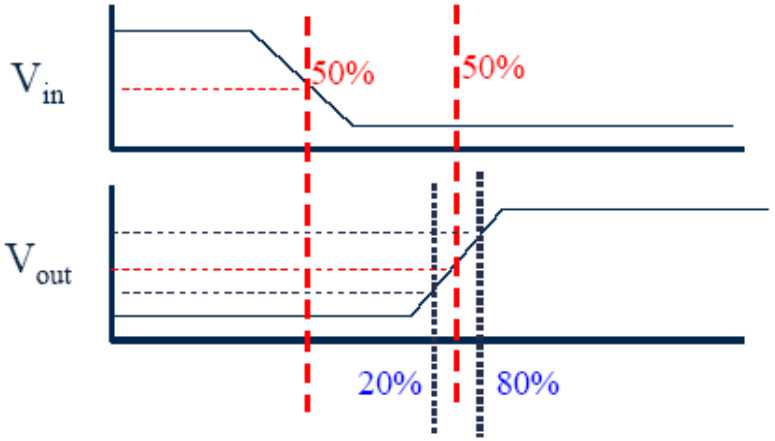
T_0 : cell pin to pin intrinsic delay
(delay without any loading)
A_c : drive impedance

Non-linear Cell Delay Model

- **Delay values are stored in the delay tables**
  - Cell delay
  - Transition delay

- **Interpolation is used**





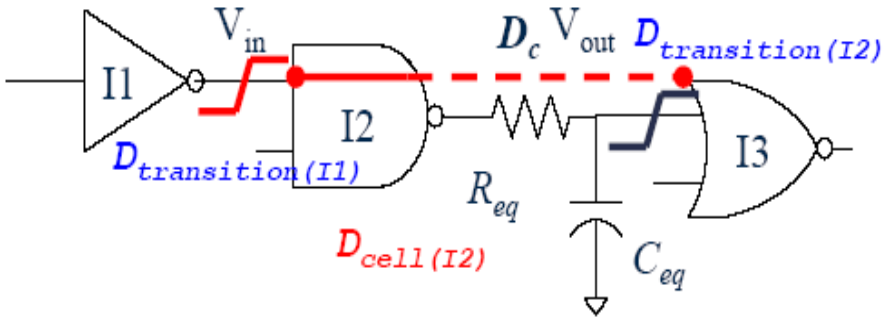
Delay Table

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

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

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



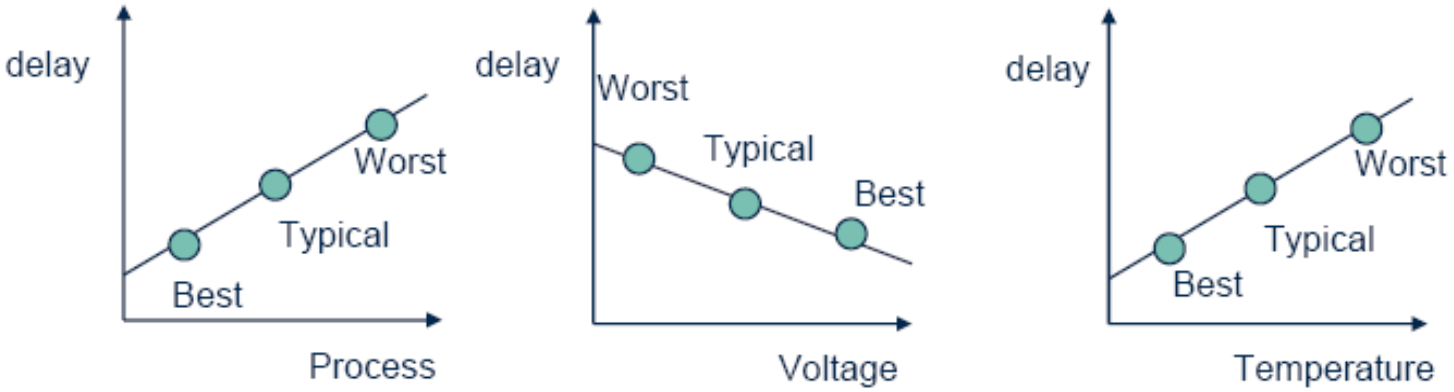
Operating Conditions

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

$$D_{scale} = D(1+\Delta_P K_P)(1+\Delta_V K_V)(1+\Delta_T K_T)$$

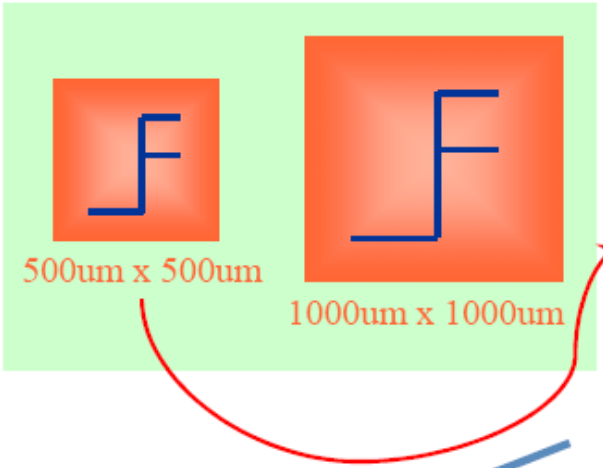
$$\Delta_P = \mathrm{Pr}\,ocess - nom\_process \,/\, \Delta_V = Voltage - nom\_voltage$$

$$\Delta_T = Temperature - nom\_temperature$$

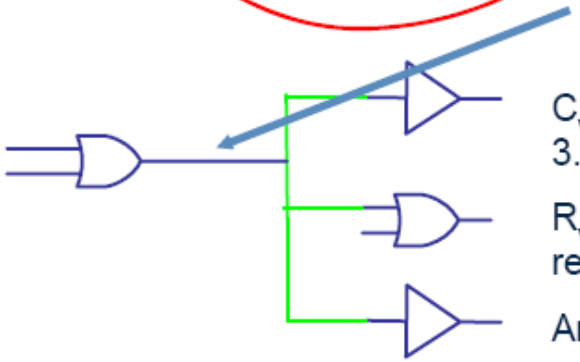


Wireload Model

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




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



C_{wire} = (fanout=3, length =2.8) x capacitance coefficient (1.3) = 3.64 load units

R_{wire} = (fanout=3, length =2.8) x resistance coefficient (3.0) = 8.4 resistance units

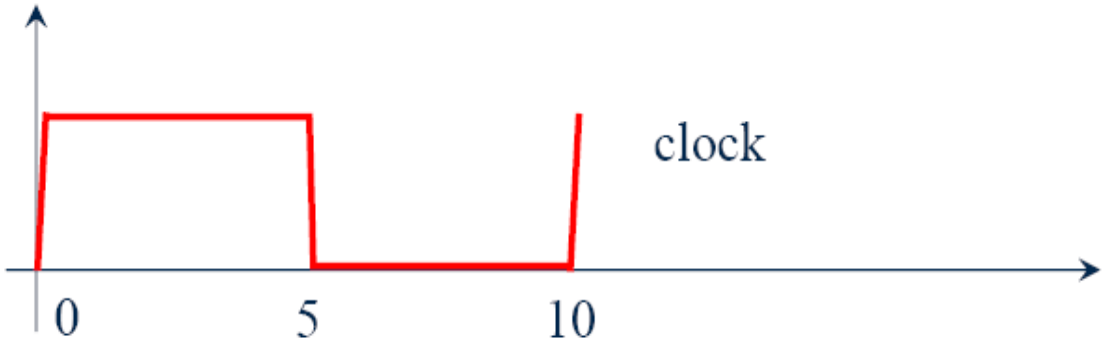
$Area_{Net}$ = (fanout=3, length =2.8) x area coefficient (0.04) = 0.112 net area units

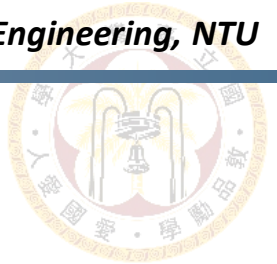
Define Clock Specification

- **We need to accurately specify the clock including the clock routing details in the early design stage in order to achieve timing convergence**
- **What should be defined?**
  - Period
  - Waveform
  - Latency
    - Source latency
    - Network latency
  - Uncertainty
- **All register-to-register paths are constrained now**

Clock Period & Waveform

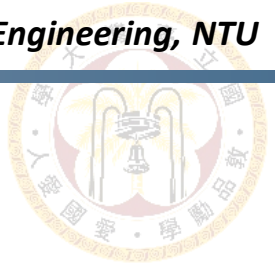
Period T_{cycle}	Rise time T_{rise}	Fall time T_{fall}
10ns	1ns	1ns





Outline

- Introduction
- Static Timing Analysis
- Environment Setting
- **Special Timing Paths**

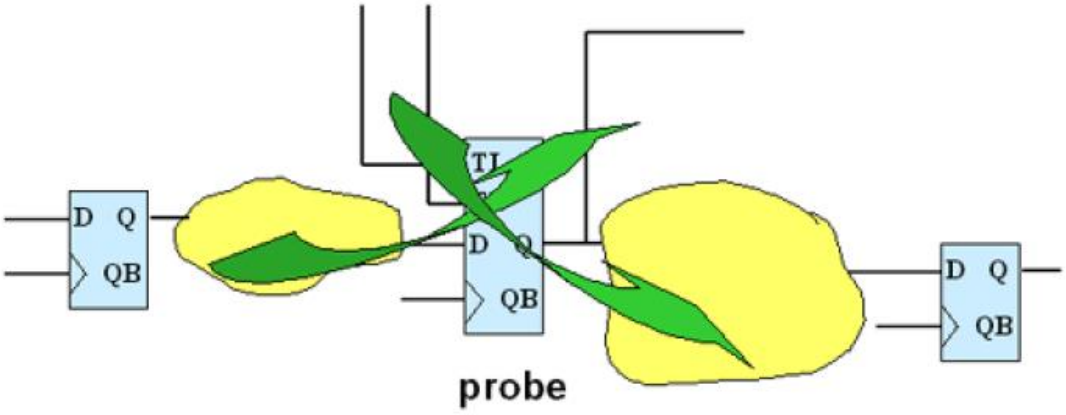


Special Timing Paths: False Paths

- **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**

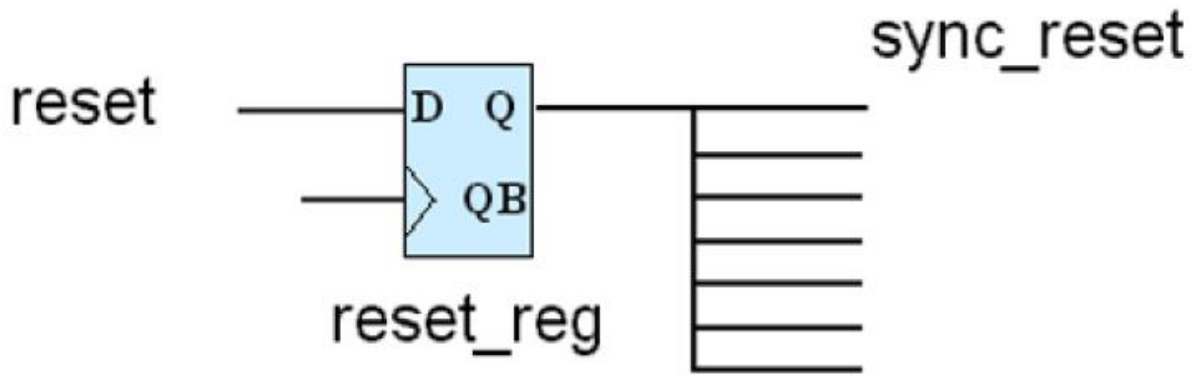
Unexercised Path

- **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



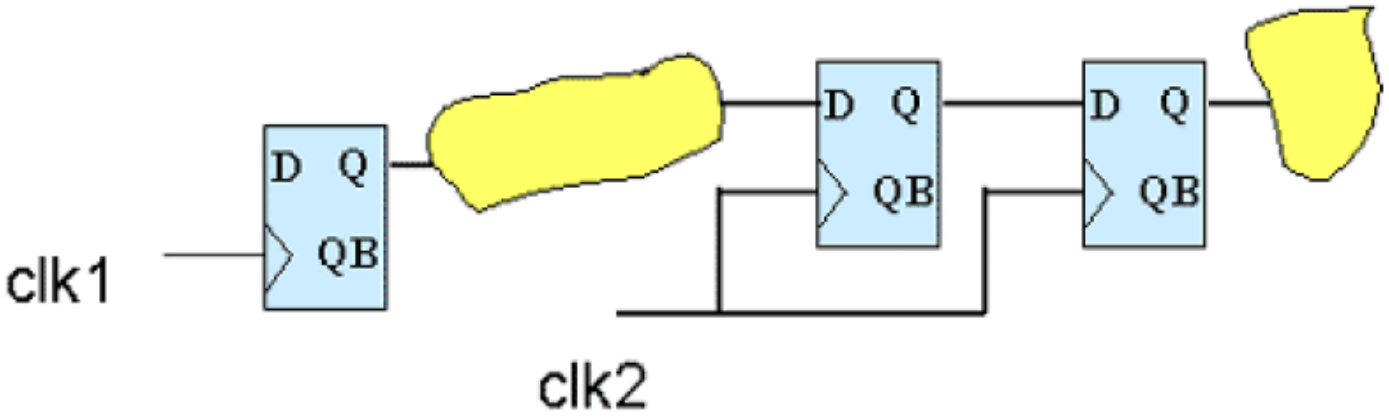
Irrelevant Path

- **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



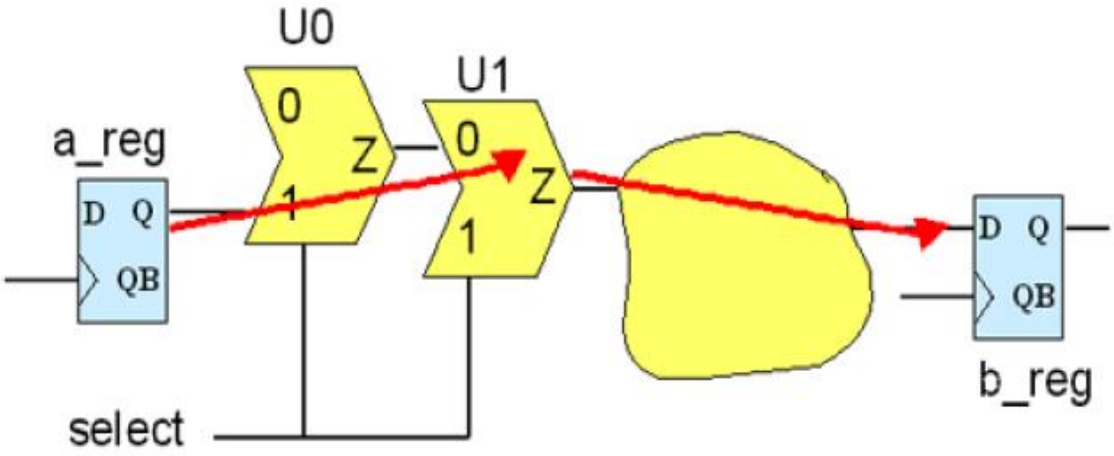
Asynchronous Path

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



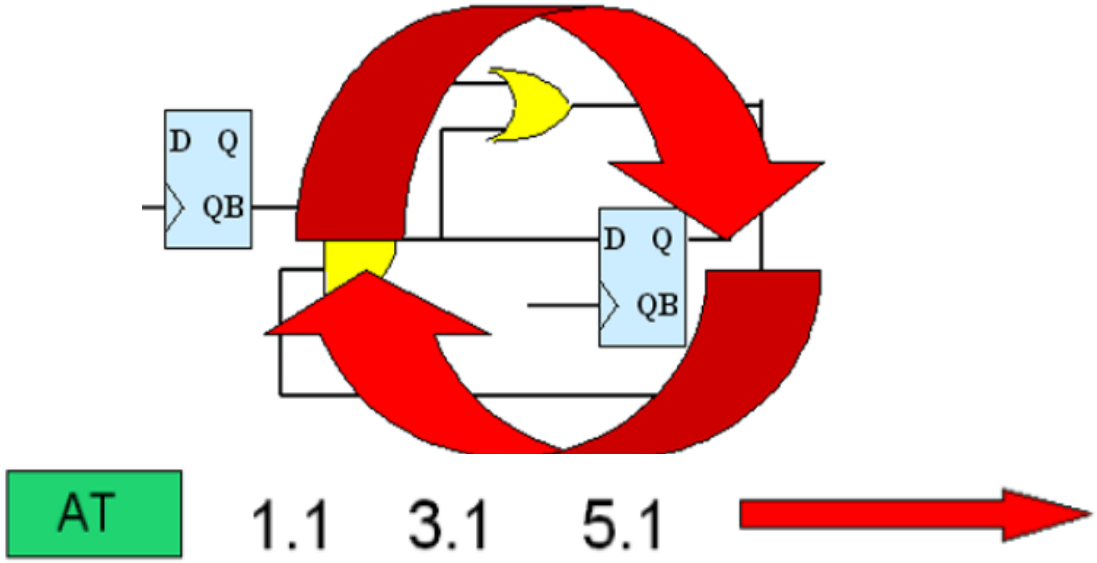
Logically Impossible Path

- 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



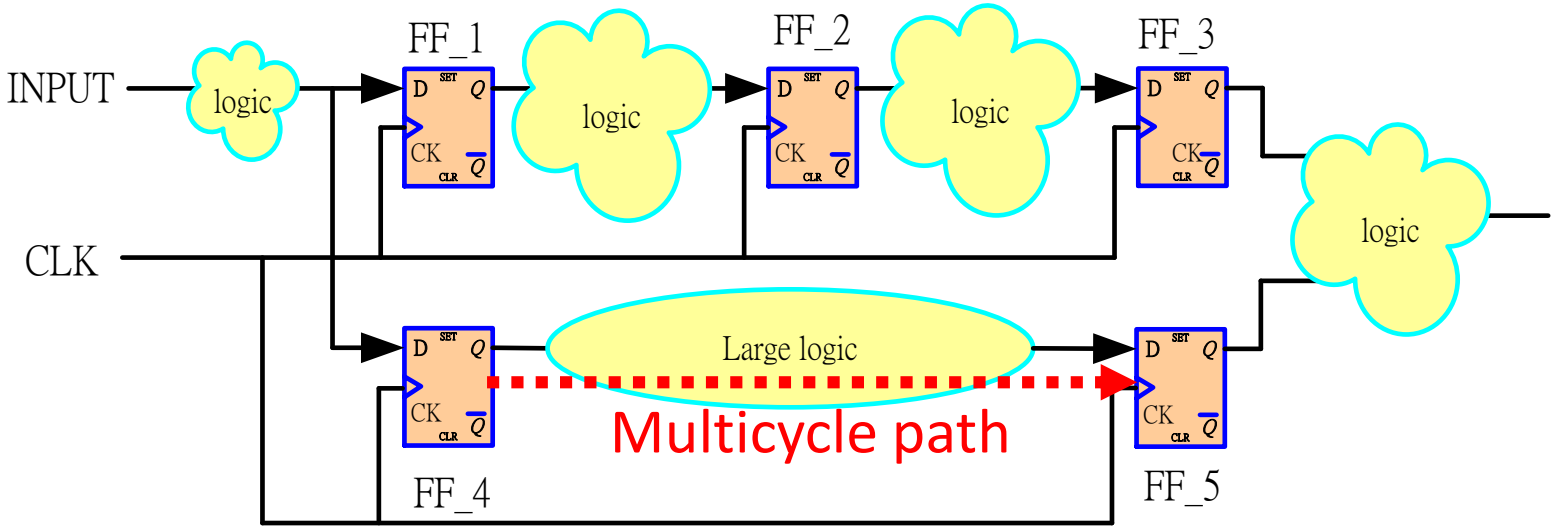
Combinational Loops

- **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

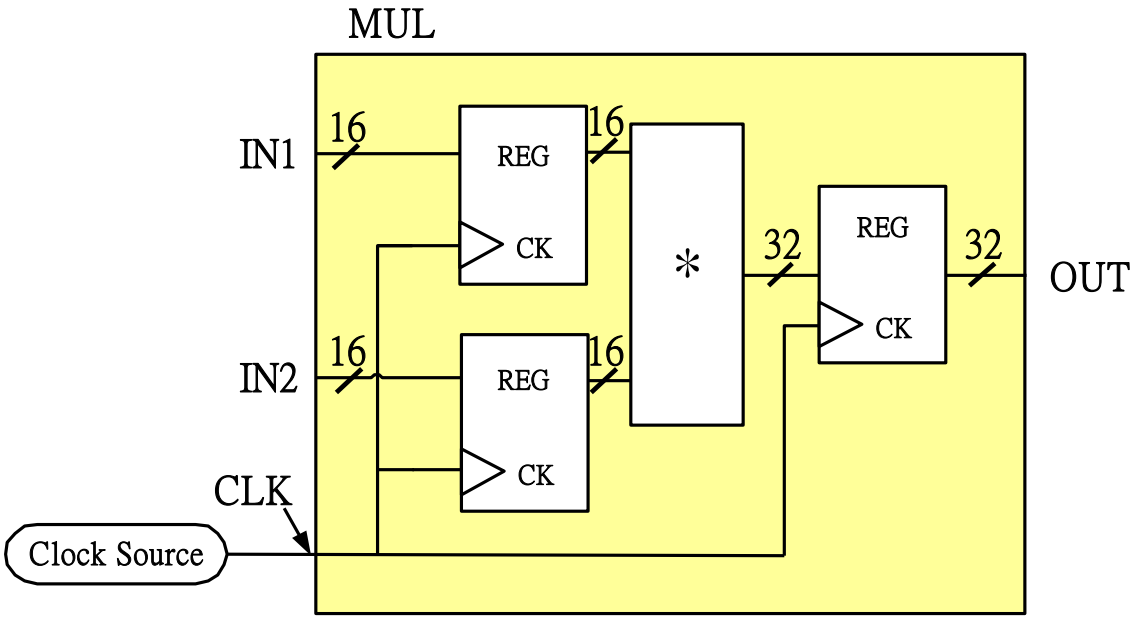


Special Timing Paths: Multicycle Paths

- **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
```

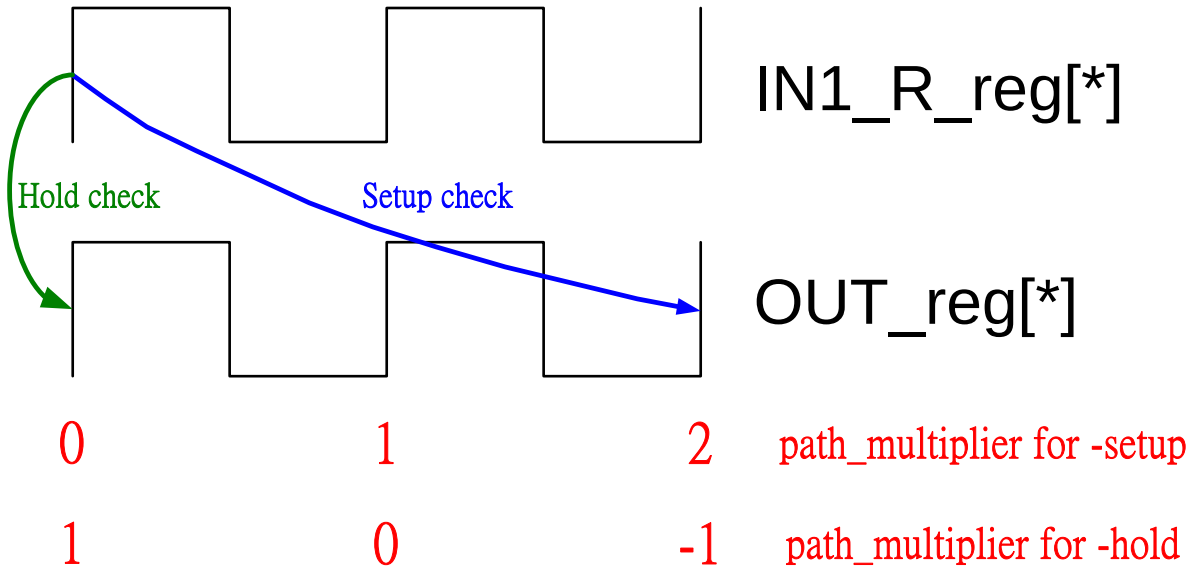
Example: STA Result

```
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 (VIOLATED)                                   -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[*]]
```



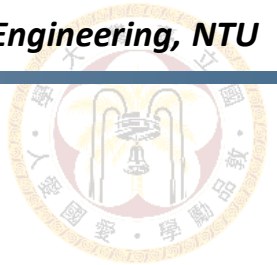
Example: STA Result with Multicycle Paths

```
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
```

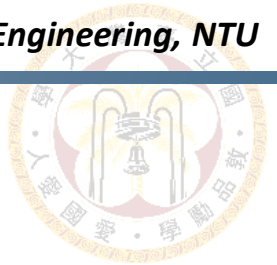
2T

Source: NCTU ICLab



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

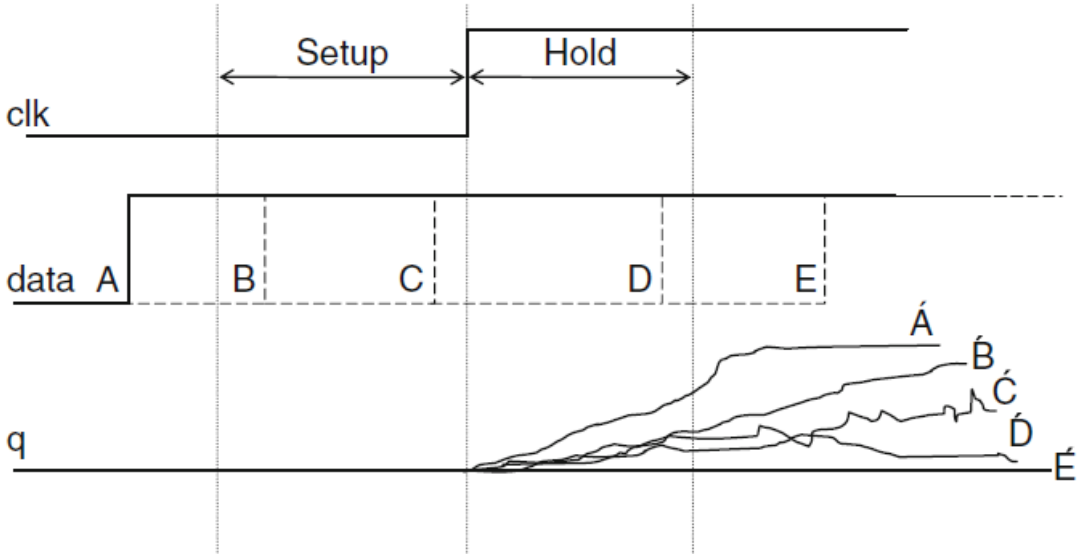


Supplementary: CDC

- **Clock domain crossing (CDC) is to transfer data from one clock domain to another clock domain**
- **There are often multiple clock domains in modern designs**
  - Different blocks require different timing constraints
- **Trigger metastability issue**

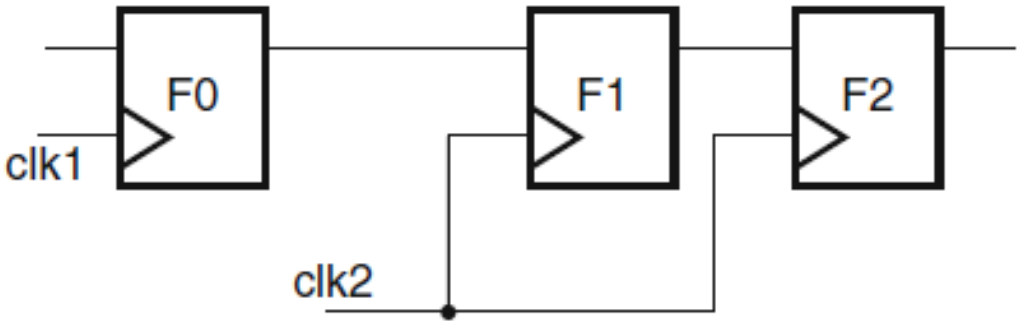
Supplementary: MetaStability

- The unstable status due to non-ideal data transition is called metastability
- To avoid this phenomenon, we have to ensure no data transition during setup/hold timing check

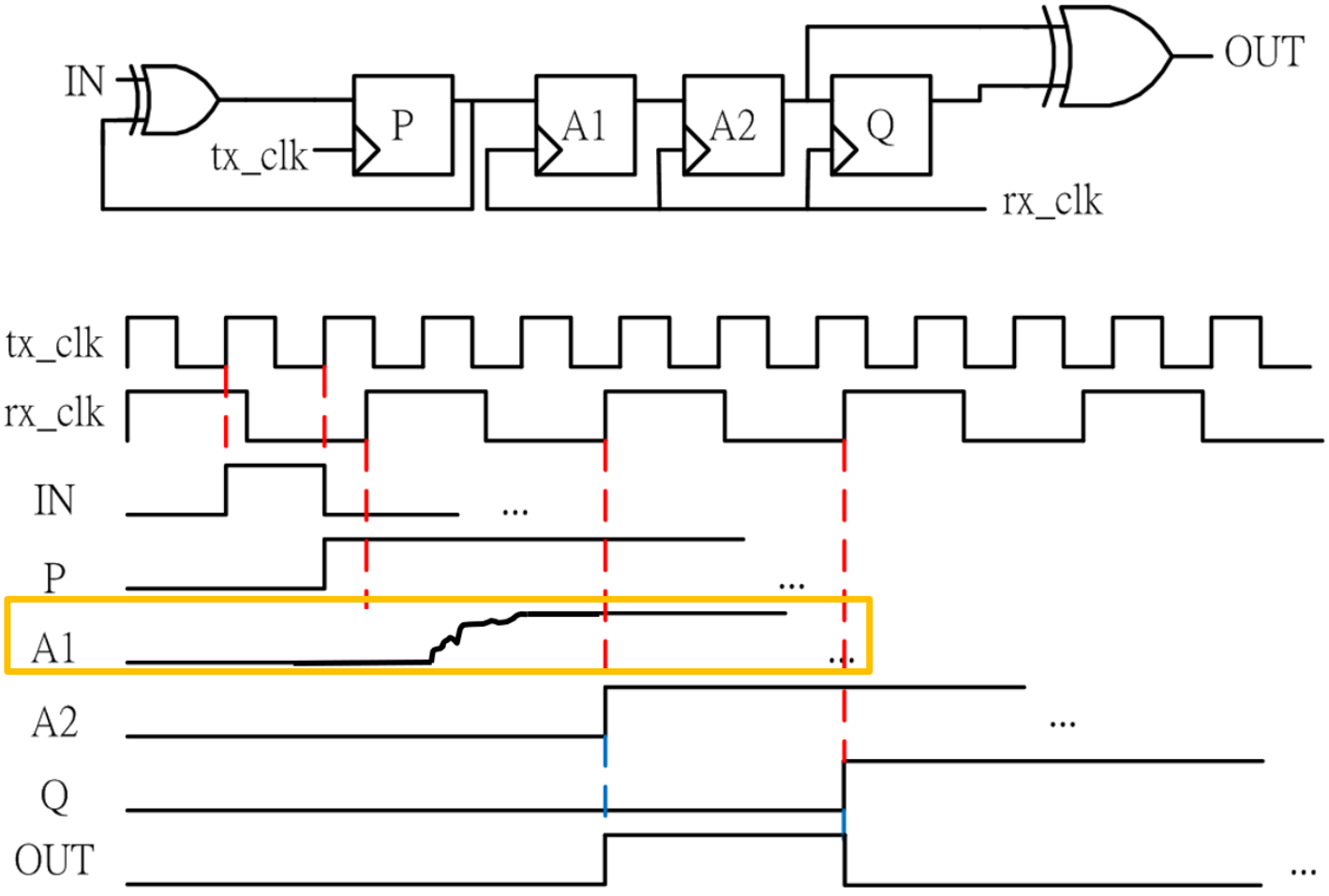


Supplementary: CDC Solution

- An easiest way: Double Flop (2-FF) Synchronizer
- F1 samples a CDC signal (probably a meta-stable value)
- F2 captures the F1 output after a full clk2 cycle to permit any metastability
- Often used in 1-bit flag transmission

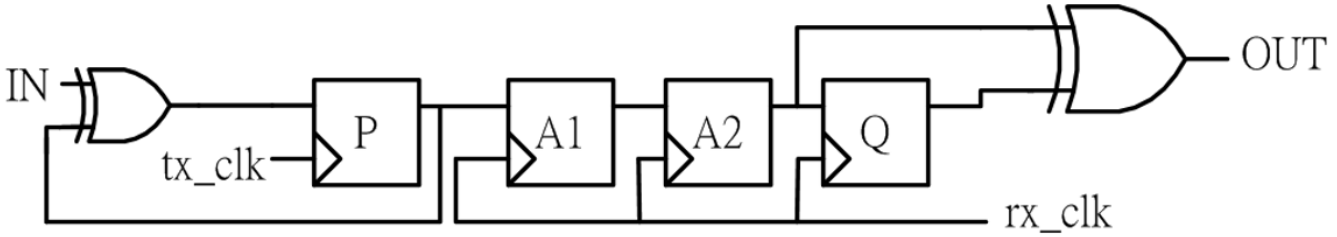


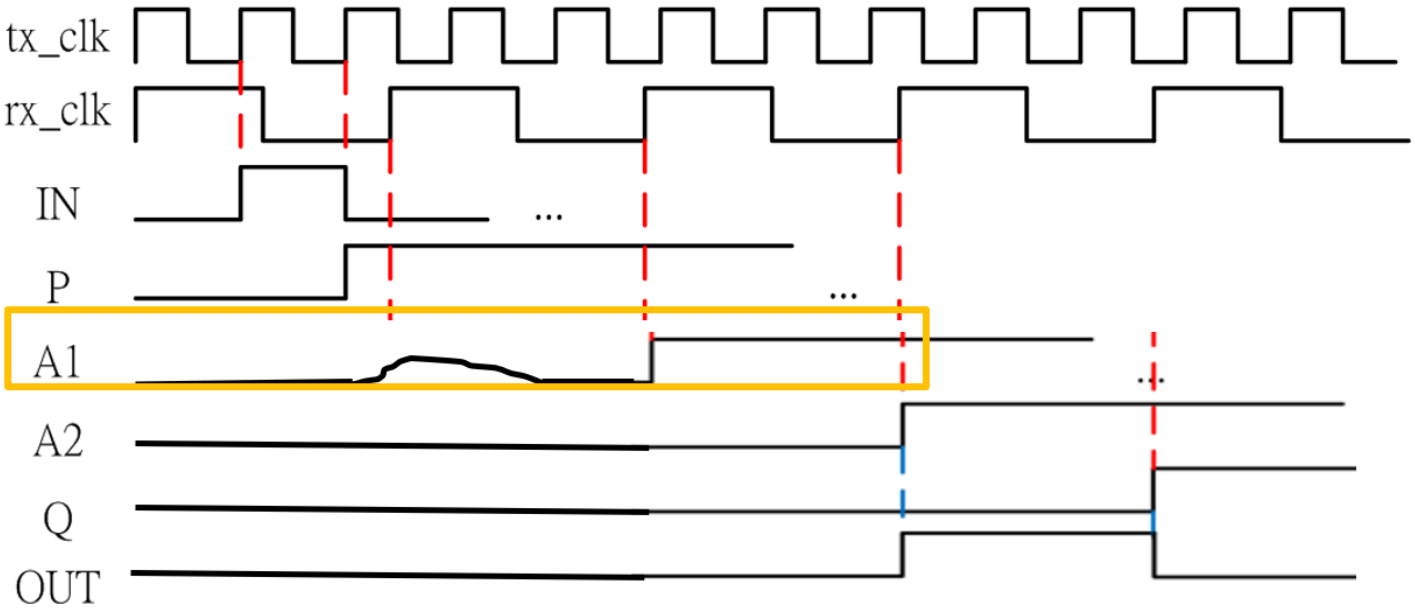
Supplementary: CDC Solution



Source: NCTU ICLab

Supplementary: CDC Solution





Supplementary: Setup False Path

- Ignore timing constraints from F0 to F1 (CDC boundary)