2022-01-12
chapter1 ~ 2.3

계인혜

SOGANG UNIVERSITY

Contents

Introduction

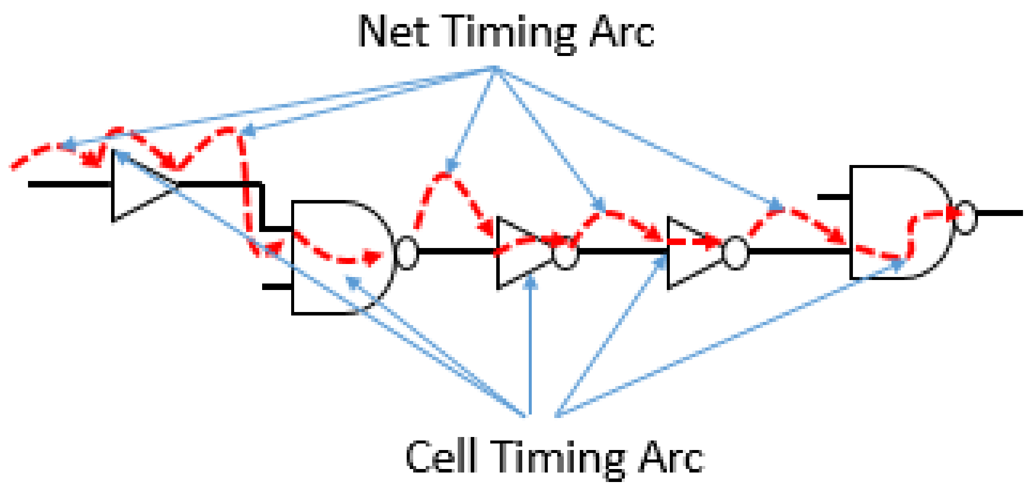
- **Reasons for performing timing analysis**
  - Timing Constraints
  - Operating Environment
  - Component Selection

- **Types of Timing Analysis**
  - STA, DTA

1.1a Timing Arc

- **What we are interested in timing analysis.**
  - How and when will we get the output after applying a certain input?
  - Is there any constraint and if yes, then what are those and on what pin?

- **Timing arc : An abstract notion of a timing dependence between the signals at any two related pins of a standard cell.**
  - It has a start-point and end-point.
  - It can be divided into Net / Cell arc.



SOGANG UNIVERSITY

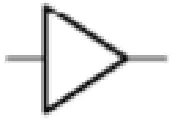
1.1b Unate : Timing Arc

- **Unateness : How Input pin is logically connected with output pin?**
  - Positive Unate
    - Rising Input : Rising Output OR No change in Output.
    - Falling Input : Falling Output OR No change in Output.
    - E.g. : BUF, AND, OR
  - Negative Unate
    - Rising Input : Falling Output OR No change in Output.
    - Falling Input : Rising Output OR No change in Output.
    - E.g. : INV, NAND, NOR
  - Non Unate
    - Output pin is not dependent on single input pin.
    - E.g. : XOR, XNOR

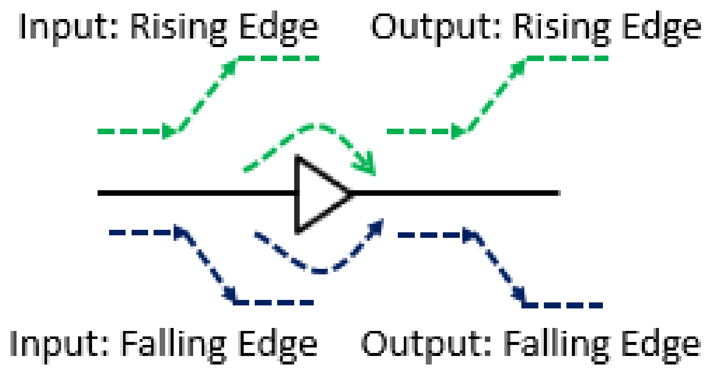
1.1c Unateness of Complex Circuit : Timing Arc

◆ **Methods to determine the Unateness**

- Truth table Method



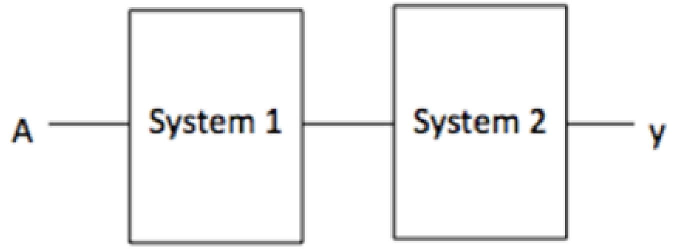
INPUT	OUTPUT
A	
0	0
1	1



1.1c Unateness of Complex Circuit : Timing Arc

Methods to determine the Unateness

- Circuit Method

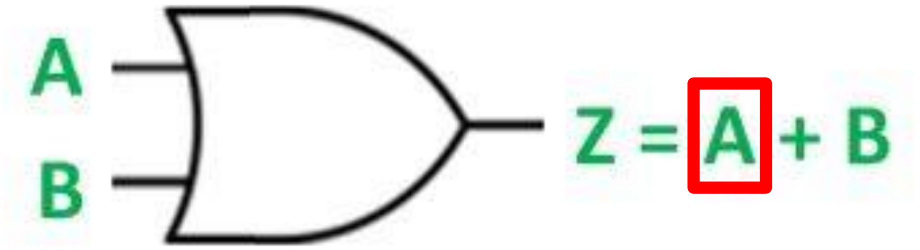


System 1 Unate	System 2 Unate	Overall System Unate
Positive	Positive	Positive
Negative	Positive	Negative
Positive	Negative	Negative
Negative	Negative	Positive
Non Unate	Positive / Negative	Non-Unate
Positive / Negative	Non-Unate	Non-Unate
Non Unate	Non Unate	Non Unate

1.1c Unateness of Complex Circuit : Timing Arc

Methods to determine the Unateness

- Function Method
  - Positive unate : In SOP representation, x’ doesn’t appear.
  - Negative unate : In SOP representation, x doesn’t appear.
  - Non-unate : In SOP representation, both x and x’ appear.



Z is positive unate with respect to A.

1.2b LIB File syntax for Complex Circuit : Timing Sense

```
pin(YS) {
    direction : output;
    ....;
    function : "((A^B)^C)";
    timing(A_YC) {
        related_pin : "A";
        timing_sense : non_unate;
        ....
    }
    timing(B_YC) {
        related_pin : "B";
        timing_sense : non_unate;
        ....
    }
    timing(C_YC) {
        related_pin : "C";
        timing_sense : non_unate;
        ....
    }
```

```
timing(A_Y, B_Y) {
  related_pin : "A B";
  timing_sense : positive_unate;
  ....
  ....
}
```

1.2b LIB File syntax for Complex Circuit : Timing Sense

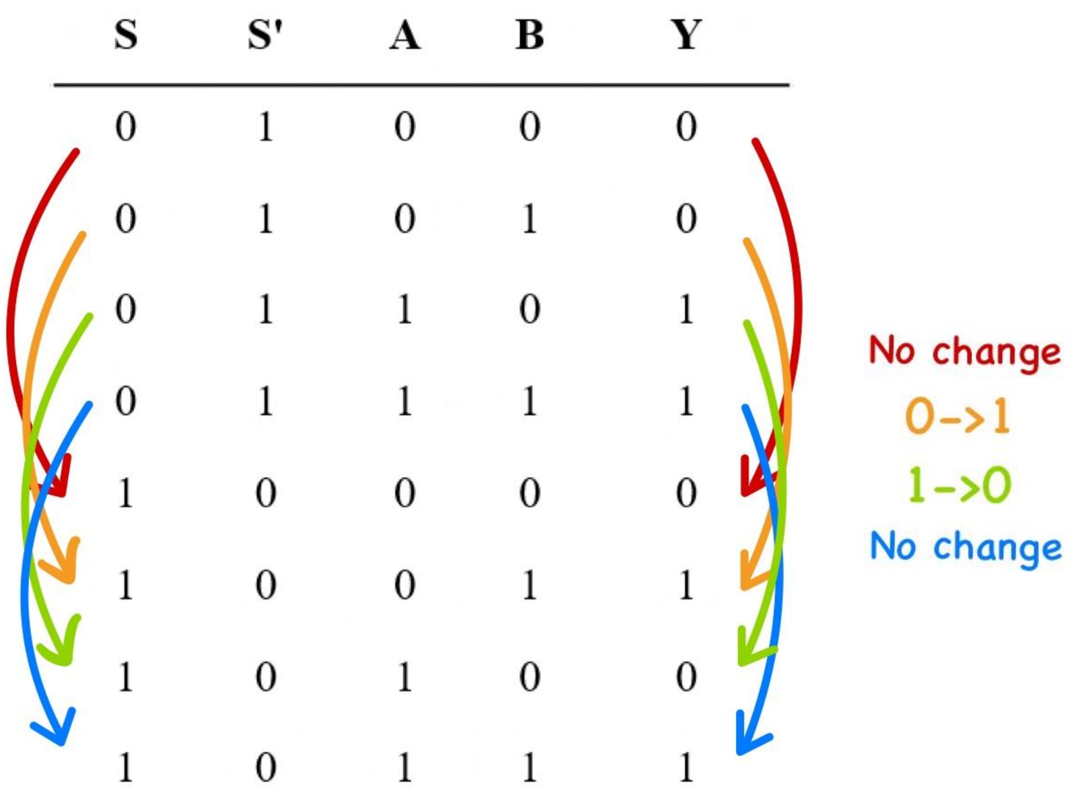
◆ For Multiple Input and Single Output

- 2×1 MUX : 2 Input, 1 Select and 1 Output Pin : **Y = (S’ ·A)+(S ·B)**

```
/* -------------- *
* Design : MUX2X1 *
* -------------- */
cell (MUX2X1) {
    ....
    pin(A) {
        direction : input;
        ....;
    }
    pin(B) {
        direction : input;
        ....;
    }
    pin(S) {
        direction : input;
        ....;
    }
    ....
```

```
pin(Y) {
    direction : output;
    ....;
    function : "((!S A) + (S B))";
    timing(A_Y) {
        related_pin : "A";
        timing_sense : positive_unate;
        ....
    }
    timing(B_Y) {
        related_pin : "B";
        timing_sense : positive_unate;
        ....
    }
    timing(S_Y) {
        related_pin : "S";
        timing_sense : non_unate;
        ....
    }
```

- **Truth table Method**

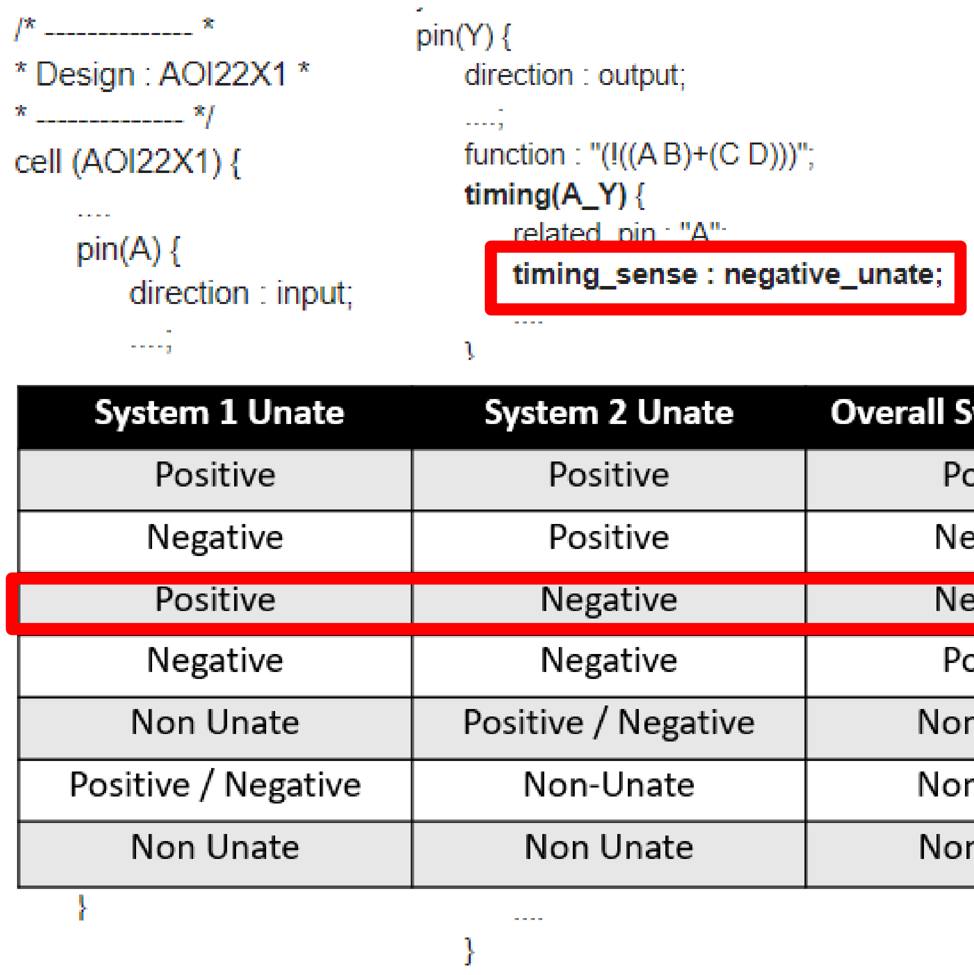


S	S'	A	B	Y
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1

1.2b LIB File syntax for Complex Circuit : Timing Sense

For Multiple Input and Single Output

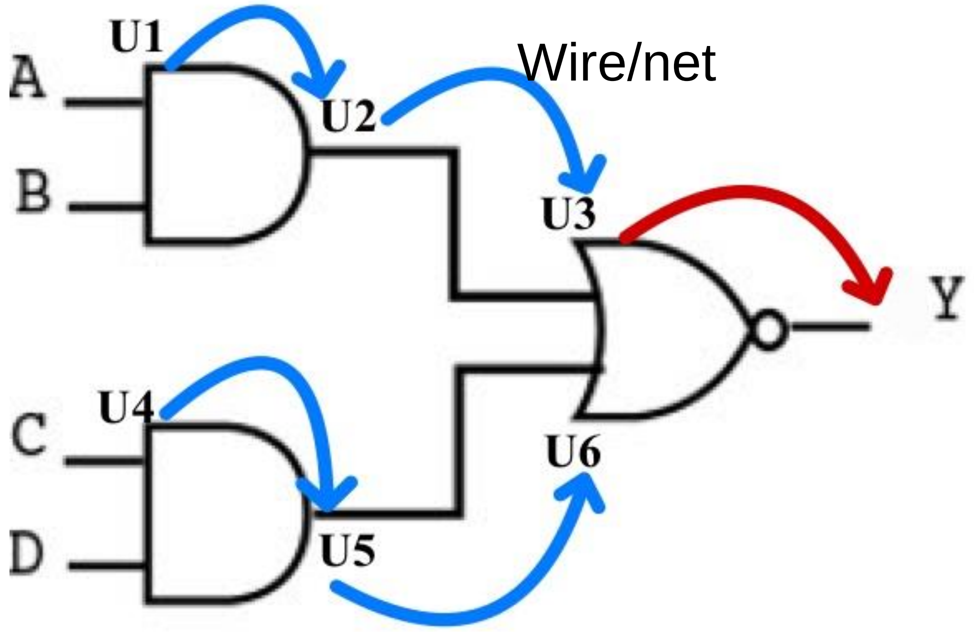
- AOI gate : 4 Input and 1 Output : **Y = ((A·B)+(C ·D))’**



```
/* -------------- *
* Design : AOI22X1 *
* -------------- */
cell (AOI22X1) {
      ....
      pin(A) {
            direction : input;
            ....;
      }
      ....
      pin(Y) {
            direction : output;
            ....;
            function : "(!((A B)+(C D)))";
            timing(A_Y) {
                  related_pin : "A";
                  timing_sense : negative_unate;
                  ....
            }
            ....
      }
```

System 1 Unate	System 2 Unate	Overall System Unate
Positive	Positive	Positive
Negative	Positive	Negative
Positive	Negative	Negative
Negative	Negative	Positive
Non Unate	Positive / Negative	Non-Unate
Positive / Negative	Non-Unate	Non-Unate
Non Unate	Non Unate	Non Unate

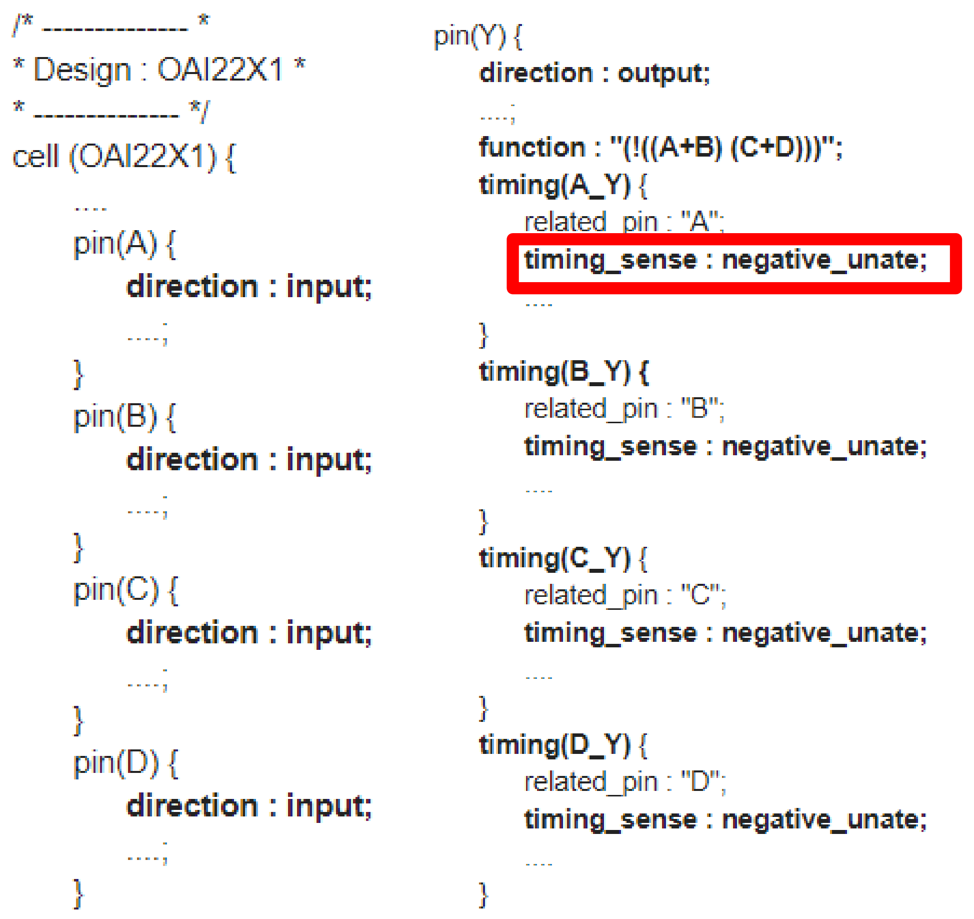
- **Circuit Method**



1.2b LIB File syntax for Complex Circuit : Timing Sense

◆ For Multiple Input and Single Output

- OAI gate : 4 Input and 1 Output : **Y = ((A+B)·(C+D))’**



```
/* -------------- *
* Design : OAI22X1 *
* -------------- */
cell (OAI22X1) {
      ....
      pin(A) {
            direction : input;
            ....;
      }
      pin(B) {
            direction : input;
            ....;
      }
      pin(C) {
            direction : input;
            ....;
      }
      pin(D) {
            direction : input;
            ....;
      }
```

```
pin(Y) {
      direction : output;
      ....;
      function : "(!((A+B) (C+D)))";
      timing(A_Y) {
            related_pin : "A";
            timing_sense : negative_unate;
            ....
      }
      timing(B_Y) {
            related_pin : "B";
            timing_sense : negative_unate;
            ....
      }
      timing(C_Y) {
            related_pin : "C";
            timing_sense : negative_unate;
            ....
      }
      timing(D_Y) {
            related_pin : "D";
            timing_sense : negative_unate;
            ....
      }
```

- **Function Method**

Positive unate : In SOP representation, x’ doesn’t appear.

Negative unate : In SOP representation, x doesn’t appear.

Non-unate : In SOP representation, both x and x’ appear.

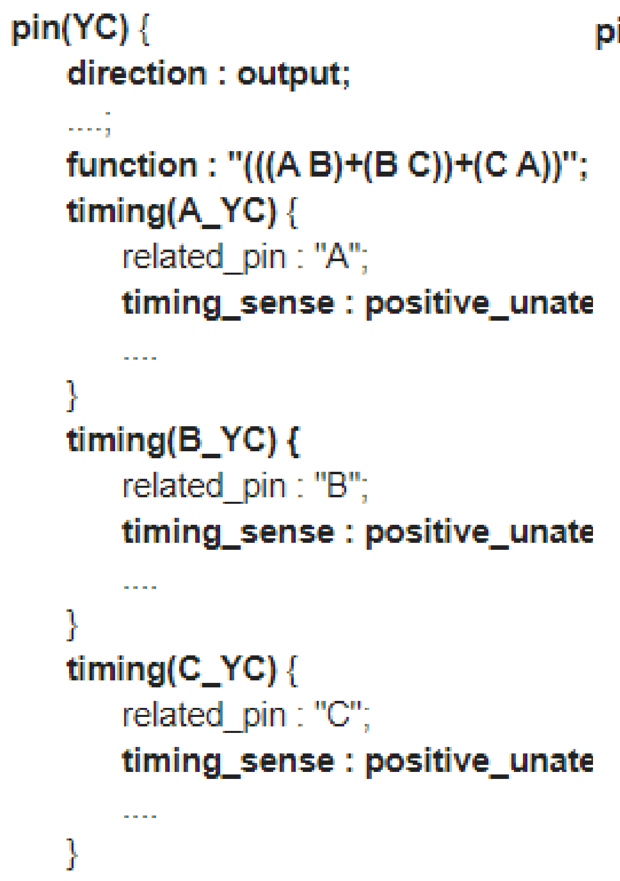
$$Y = ((A+B)\cdot(C+D))'$$
$$= \underline{A'}\cdot B' + C'\cdot D'$$

SOGANG UNIVERSITY

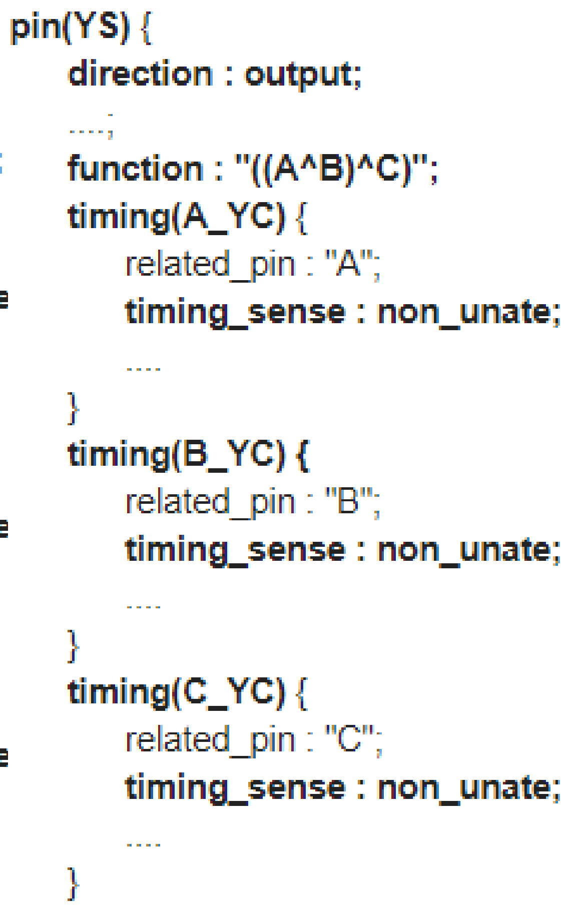
1.2b LIB File syntax for Complex Circuit : Timing Sense

◆ For Multiple Input and Multiple Output

- Full Adder : **YC = ((A·B)+(B·C))+(C·A)), YS = (A^B)^C**

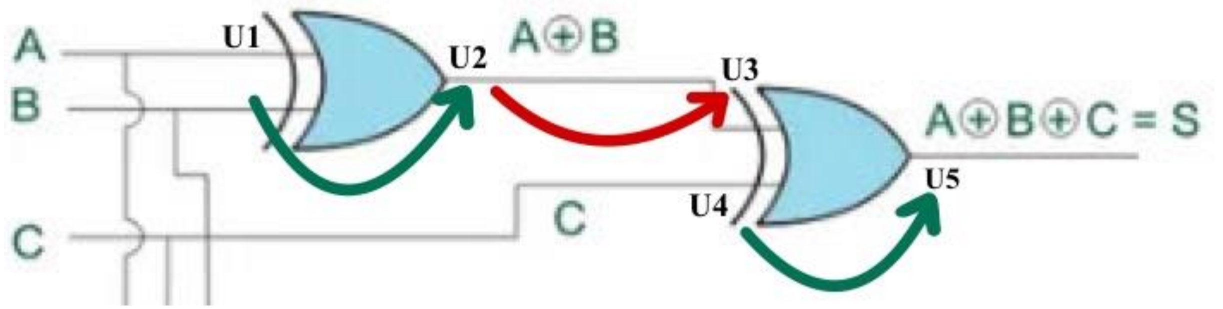


```
pin(YC) {
    direction : output;
    ....;
    function : "(((A B)+(B C))+(C A))";
    timing(A_YC) {
        related_pin : "A";
        timing_sense : positive_unate
        ....
    }
    timing(B_YC) {
        related_pin : "B";
        timing_sense : positive_unate
        ....
    }
    timing(C_YC) {
        related_pin : "C";
        timing_sense : positive_unate
        ....
    }
```



```
pin(YS) {
    direction : output;
    ....;
    function : "((A^B)^C)";
    timing(A_YC) {
        related_pin : "A";
        timing_sense : non_unate;
        ....
    }
    timing(B_YC) {
        related_pin : "B";
        timing_sense : non_unate;
        ....
    }
    timing(C_YC) {
        related_pin : "C";
        timing_sense : non_unate;
        ....
    }
```

- **Circuit Method**



SOGANG UNIVERSITY

1.2b LIB File syntax for Complex Circuit : Timing Sense

For Multiple Input and Single Output

- 2×1 MUX : 2 Input, 1 Select and 1 Output Pin(Other configuration) : **Y = ((S ·A)+(S' ·B))'**

```
pin(Y) {
    direction : output;
    ....;
    function : "(!((S A) + (!S B)))";
    timing(A_Y) {
        related_pin : "A";
        timing_sense : negative_unate;
        ....
    }
    timing(B_Y) {
        related_pin : "B";
        timing_sense : negative_unate;
        ....
    }
    timing(S_Y) {
        related_pin : "S";
        timing_sense : non_unate;
        ....
    }
```

- **Function Method**

Positive unate : In SOP representation, x' doesn't appear.
Negative unate : In SOP representation, x doesn't appear.
Non-unate : In SOP representation, both x and x' appear.

$$\begin{aligned} Y &= ((S\cdot A)+(S'\cdot B))' \\ &= (S'+A)\cdot(S+B') \\ &= S\cdot S' + S'\cdot B'+A\cdot S+A\cdot B' \\ &= \underline{S'}\cdot B'+A\cdot \underline{S}+A\cdot B' \text{(SOP form)} \end{aligned}$$

2.1 Timing Paths

- **STA(Static Timing Analysis) vs. DTA(Dynamic Timing Analysis)**

STA	DTA
Can report false errors	Doesn’t report false errors
Less accurate	More accurate
Doesn’t check logic operations	Checks the functionality of circuits
Fast	Slow
More thorough	Less thorough

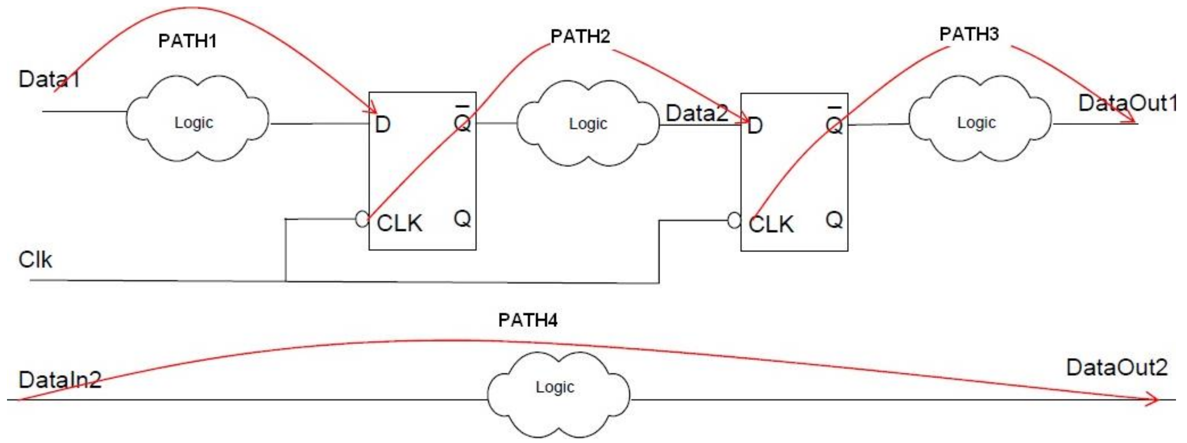
Thorough => Because it checks the worst-case timing for all possible logic conditions.

Fast => Because STA doesn’t need to simulate multiple test vectors.

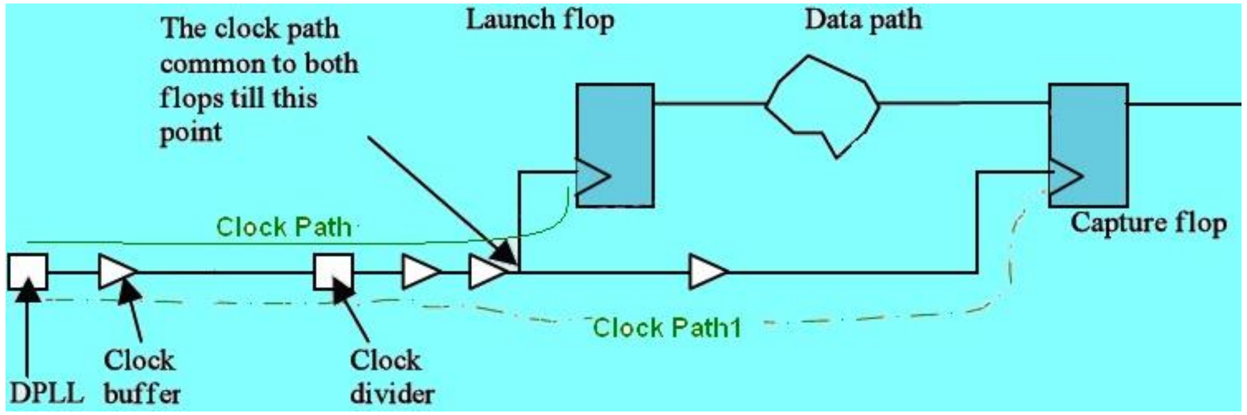
- **Characteristics of STA**
  - It performs timing analysis on all possible paths. (real/false paths)
  - It’s efficient only for fully synchronous designs.
  - All paths that are constrained by the clock should be constrained.

2.1 Timing Paths

- **Types of Paths for Timing analysis**
  - Data Path



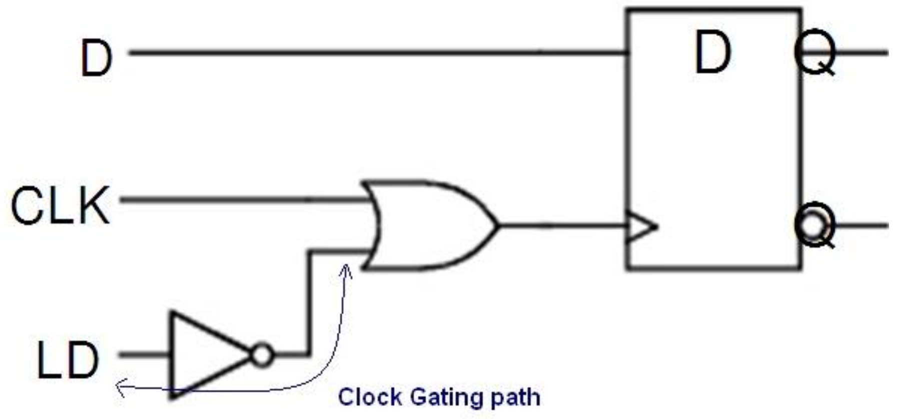
  - Clock Path



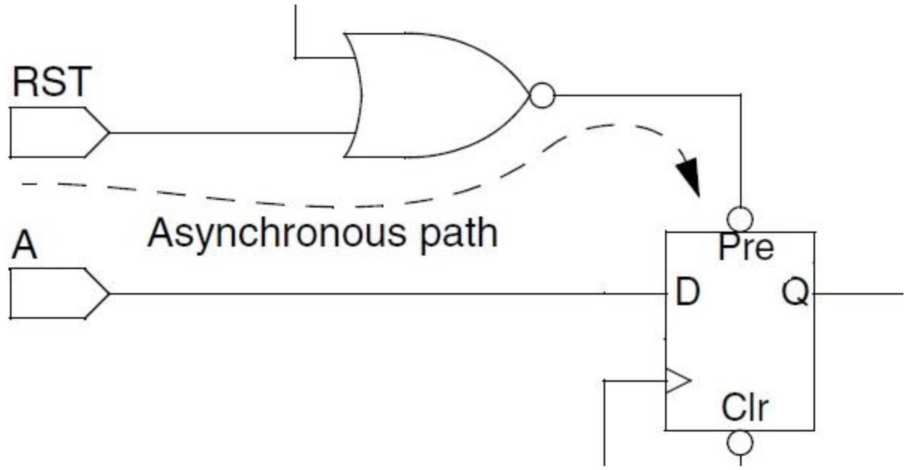
2.1 Timing Paths

Types of Paths for Timing analysis (cont.)

- Clock Gating Path



- Asynchronous Path

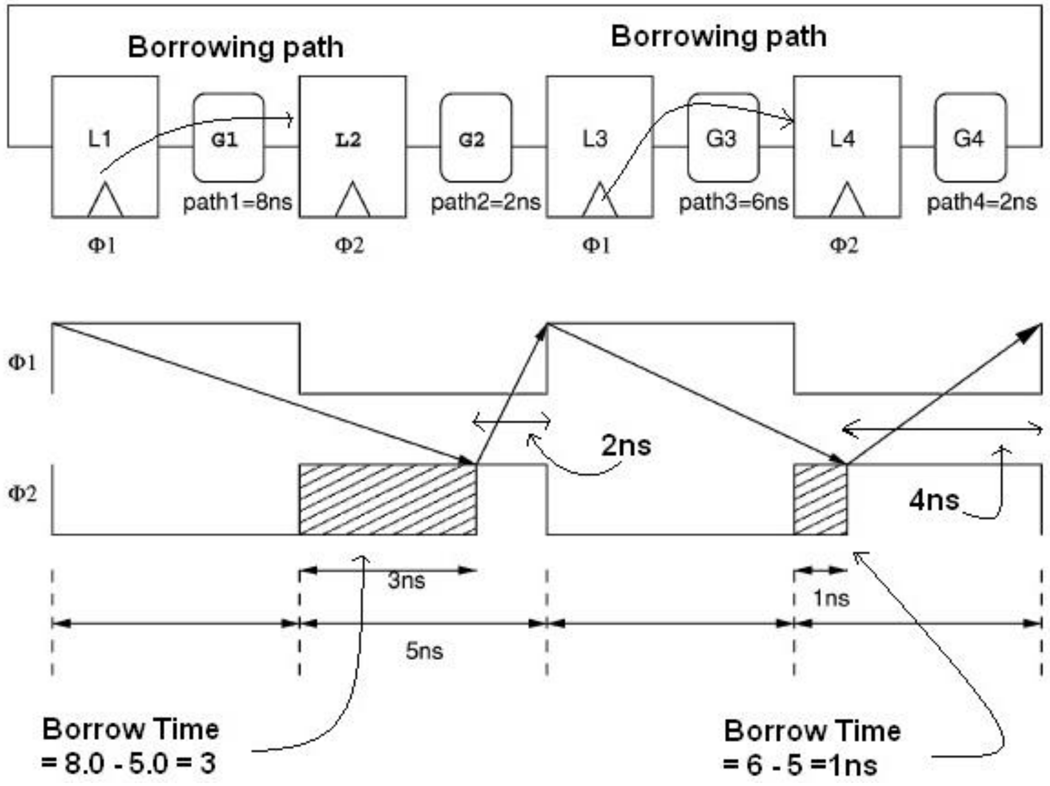


2.1 Timing Paths

Other types of Paths for Timing analysis (cont.)

- Critical Path
  - The path which creates Longest delay.
- False path
  - Physically exist in the design but those are logically / functionally incorrect path.
- Single cycle Path
  - The path that is designed to take only one clock cycle for the data to propagate.
- Multicycle Path
  - The path that is designed to take more than one clock cycle for the data to propagate.
- Longest Path (Worst Path / Max Path)
  - The path that takes longest time
- Shortest Path (Best Path / Min Path)
  - The path that takes the shortest time

2.2 Time Borrowing

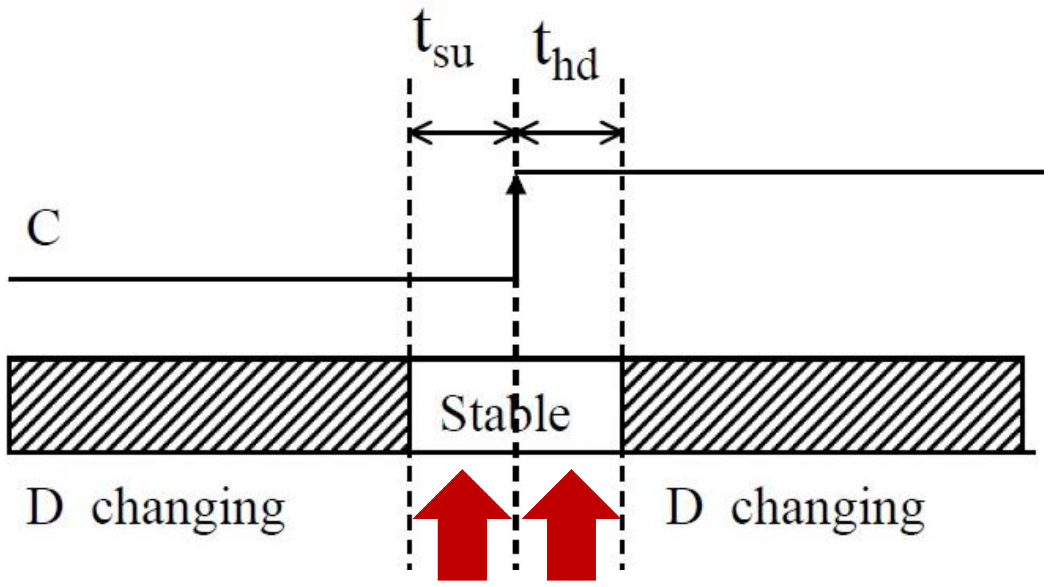


- Launching & capturing latches use the same phase of the same clock.
- Time borrowing typically only affects setup slack calculation since it slows the data arrival times.
- Time borrowing can be multistage.
- Maximum Borrow Time : (Clock pulse width) – (Setup time of the Latch)
- Negative Borrow Time : (Arrival time) – (Clock edge) < 0, No borrowing.

2.3a Basic Concept of Setup-Hold

- **Setup time**
  - Time when input data is available and stable before the clock pulse is applied.

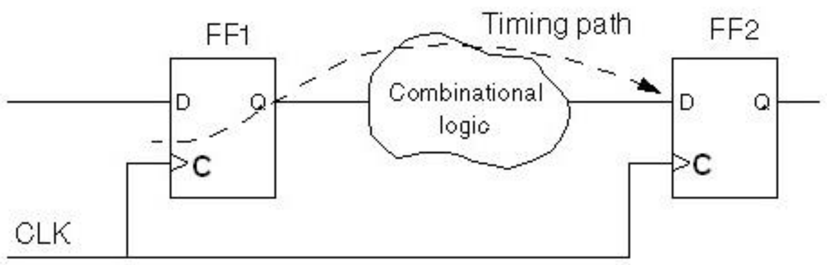
- **Hold time**
  - Time after clock pulse where data input is held stable.

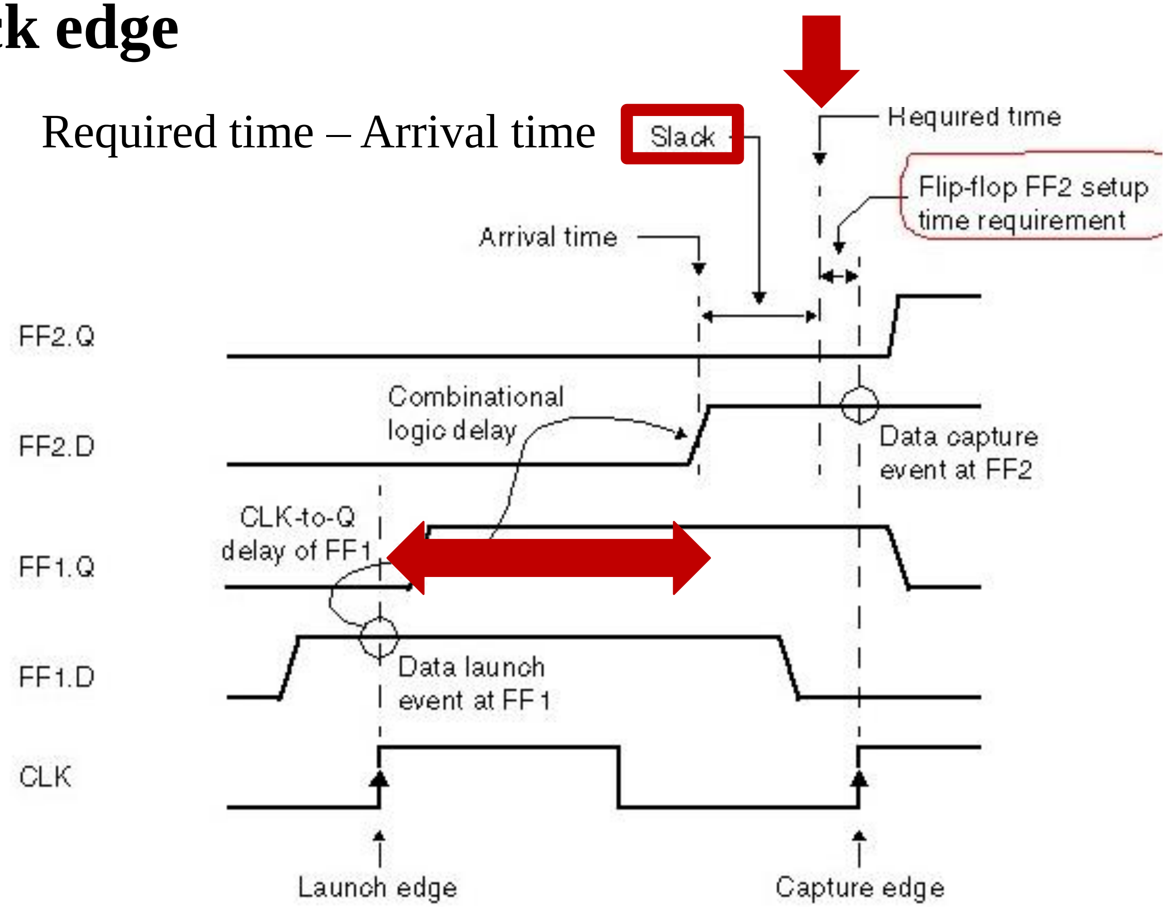


2.3b Basic Concept of Setup-Hold Violation

- **Setup Check timing – At next clock edge**

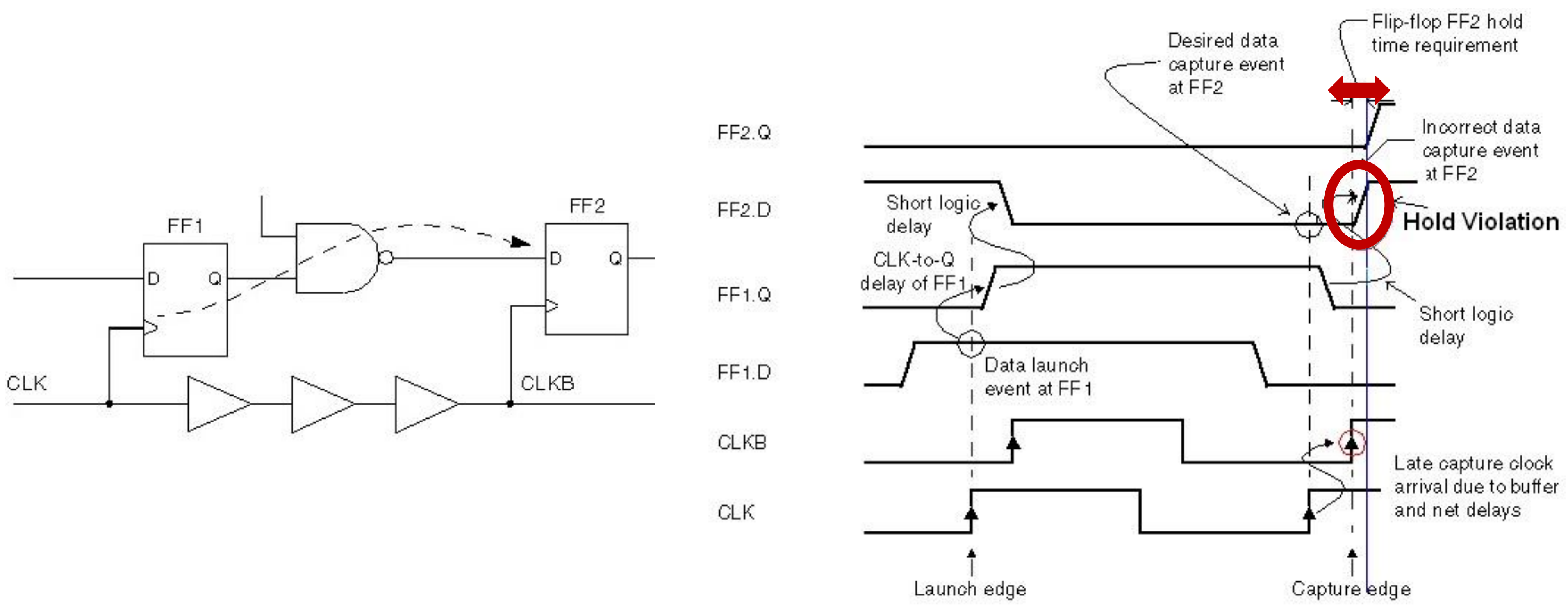
Required time – Arrival time





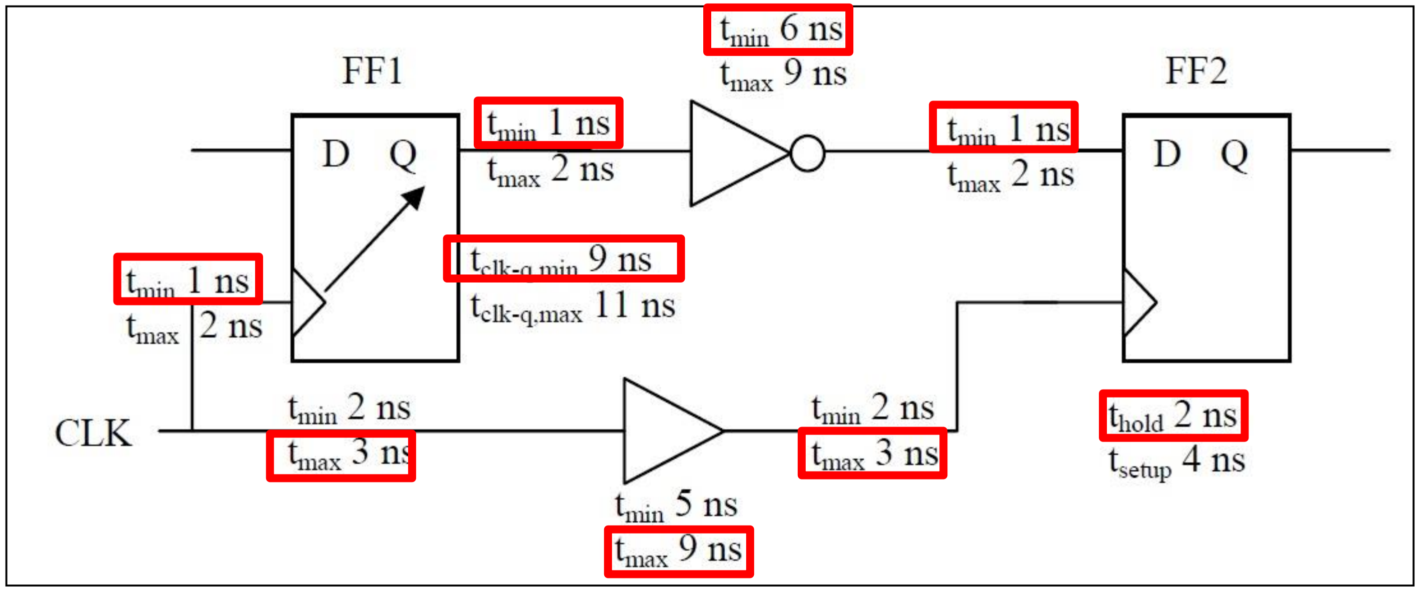
2.3b Basic Concept of Setup-Hold Violation

- **Hold Check timing – At same clock edge**



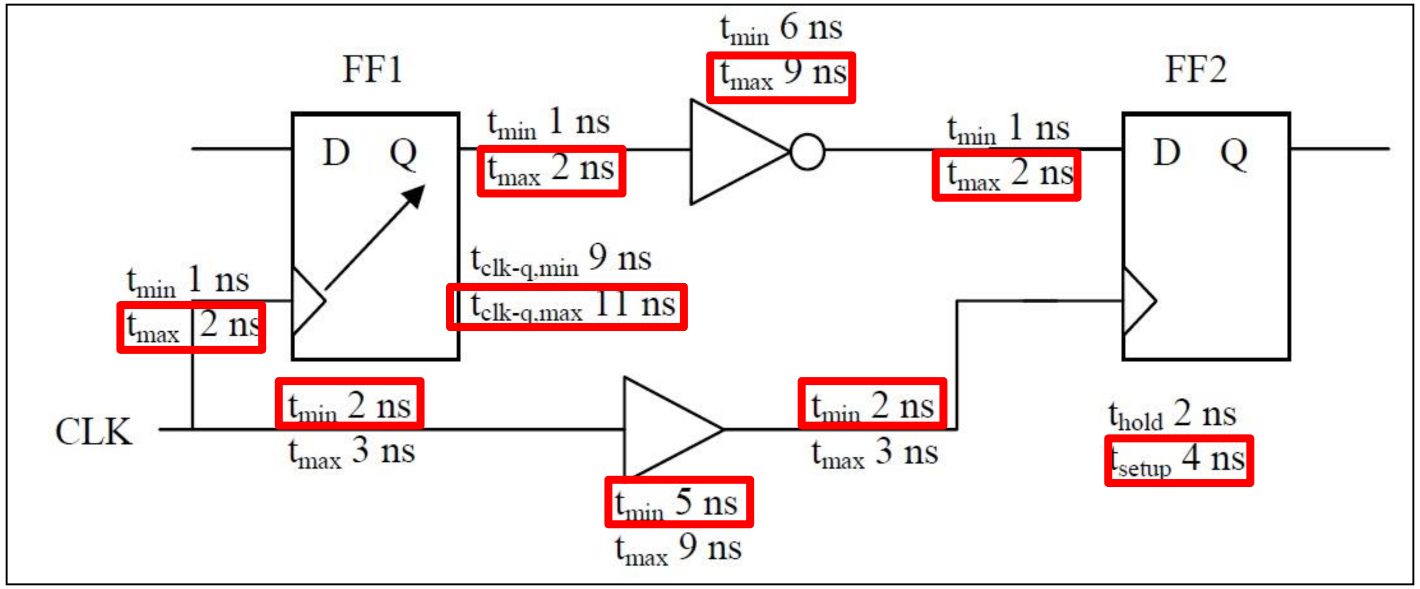
2.3c Examples : Setup-Hold Time / Violation

- **In the following circuit, find out whether there is any Setup or Hold violation?**
  - Hold analysis
    - Minimum delay along the data path, Maximum delay along the clock path
    - Data path : CLK => FF1/CLK => FF1/Q => INV => FF2/D
      - 1 + 9 + 1 + 6 + 1 = 18ns
    - Clock path : CLK => BUF => FF2/CLK
      - 3 + 9 + 3 + 2 = 17ns
    - Hold slack = Td – Tclk = 18ns – 17ns = 1ns > 0 => No hold violation!



2.3c Examples : Setup-Hold Time / Violation

- **In the following circuit, find out whether there is any Setup or Hold violation?**
  - Setup analysis
    - Maximum delay along the data path, Minimum delay along the clock path
    - Data path : CLK => FF1/CLK => FF1/Q => INV => FF2/D
      - 2 + 11 + 2 + 9 + 2 = 26ns
    - Clock path : CLK => BUF => FF2/CLK
      - 15 + 2 + 5 + 2 - 4 = 20ns
    - Setup slack = Tclk – Td = 20ns – 26ns = -6ns < 0 => Setup violation!



Source

- http://88physicaldesign.blogspot.com/2015/09/cell-delay-and-net-delay.html
- http://www.vlsijunction.com/2015/10/sta-vs-dta.html
- http://www.vlsi-expert.com/2011/03/static-timing-analysis-sta-basic-timing.html
- http://www.vlsi-expert.com/2018/01/unateness-of-complex-circuit-timing-arc.html