



LSI LOGIC DESIGN

CHAPTER 4

Synchronous Design

JUNE 17, 2020

PHAM TUONG HAI

QUALITY ASSESSMENT & TRAINING DEPARTMENT

RENESAS DESIGN VIETNAM CO., LTD.

RENESAS ELECTRONICS CORPORATION

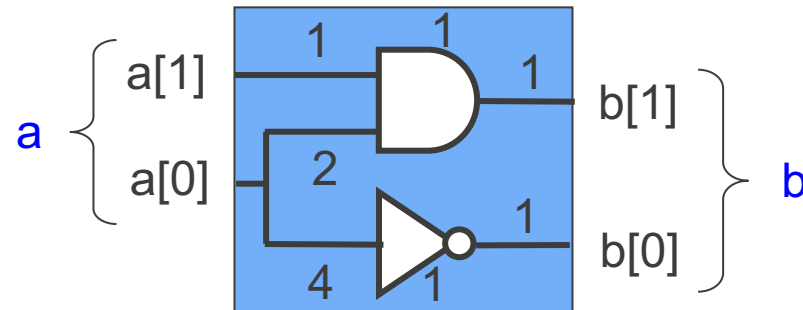
CHAPTER 4. Synchronous Design

- 4.1. Timing Issue of Logic Data.
- 4.2. Synchronous Design and
Static Timing Analysis (STA)
- 4.3. Synchronous Design Summary

4.1 Timing Issue of Logic Data

Timing Issue of Logic Data

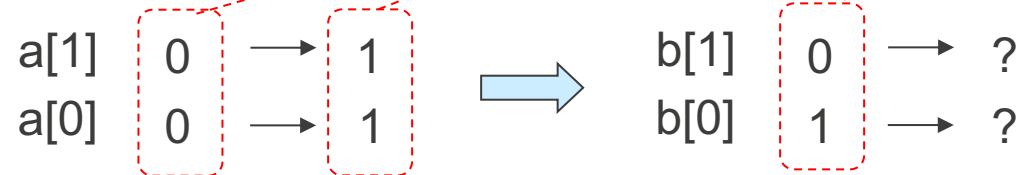
Let's study what will come out to output signal **b** when input signal **a** changes while wire connections, AND, and NOT gate have delay.



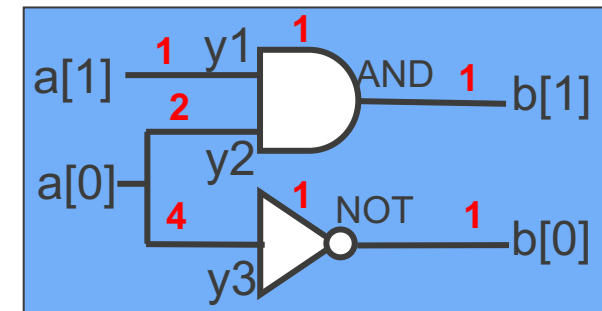
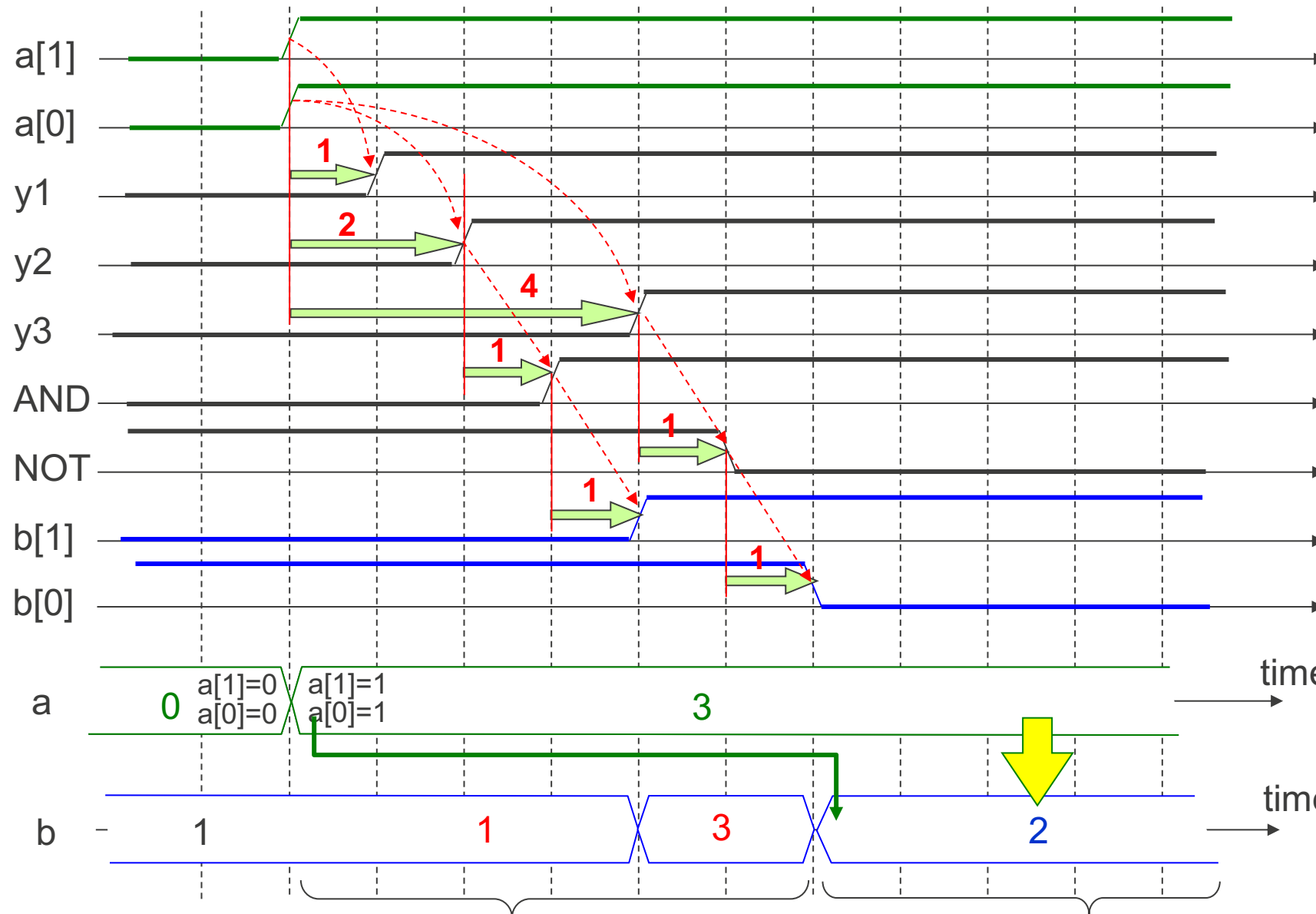
→ The difference of delay comes mainly from the difference of the length of the wire lines if gate delays are the same.

Numbers in the block diagram are **delay** of each element, wire-delay and gate-delay.

Suppose **a** changes from **0** to **3**, then **b** shall change from **1** to what ??



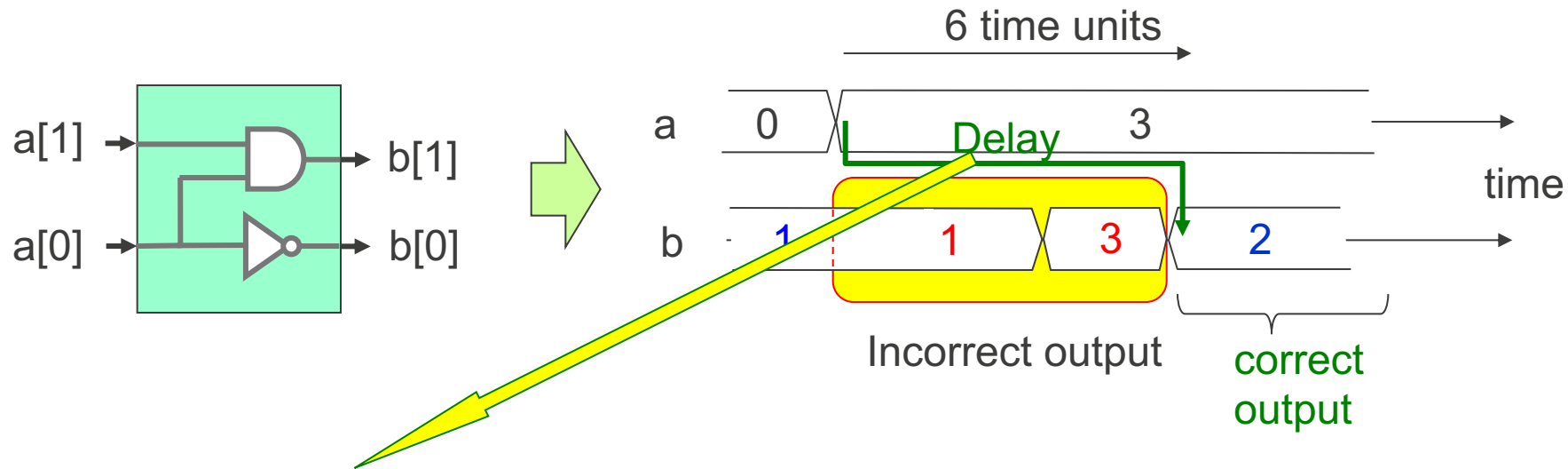
Draw the time chart for the given conditions above.



Incorrect output for $a = 3$ caused by delay and hazard

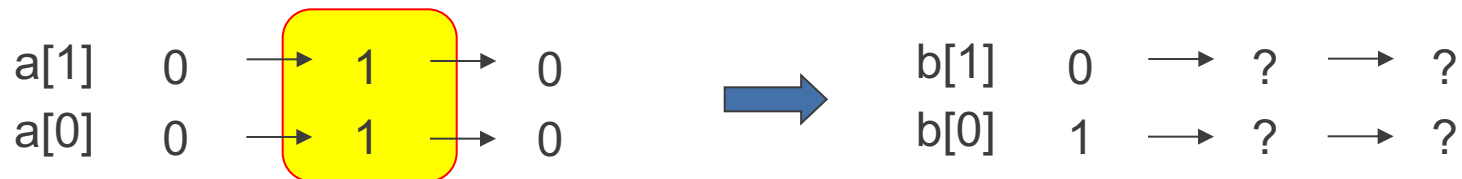
Correct output for $a = 3$

The result shows that output **b** does not have a correct value corresponding to the input **a** until 6 time units, the largest delay in the logic block, passes.

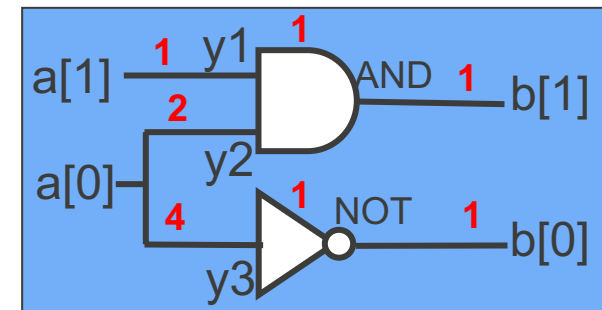
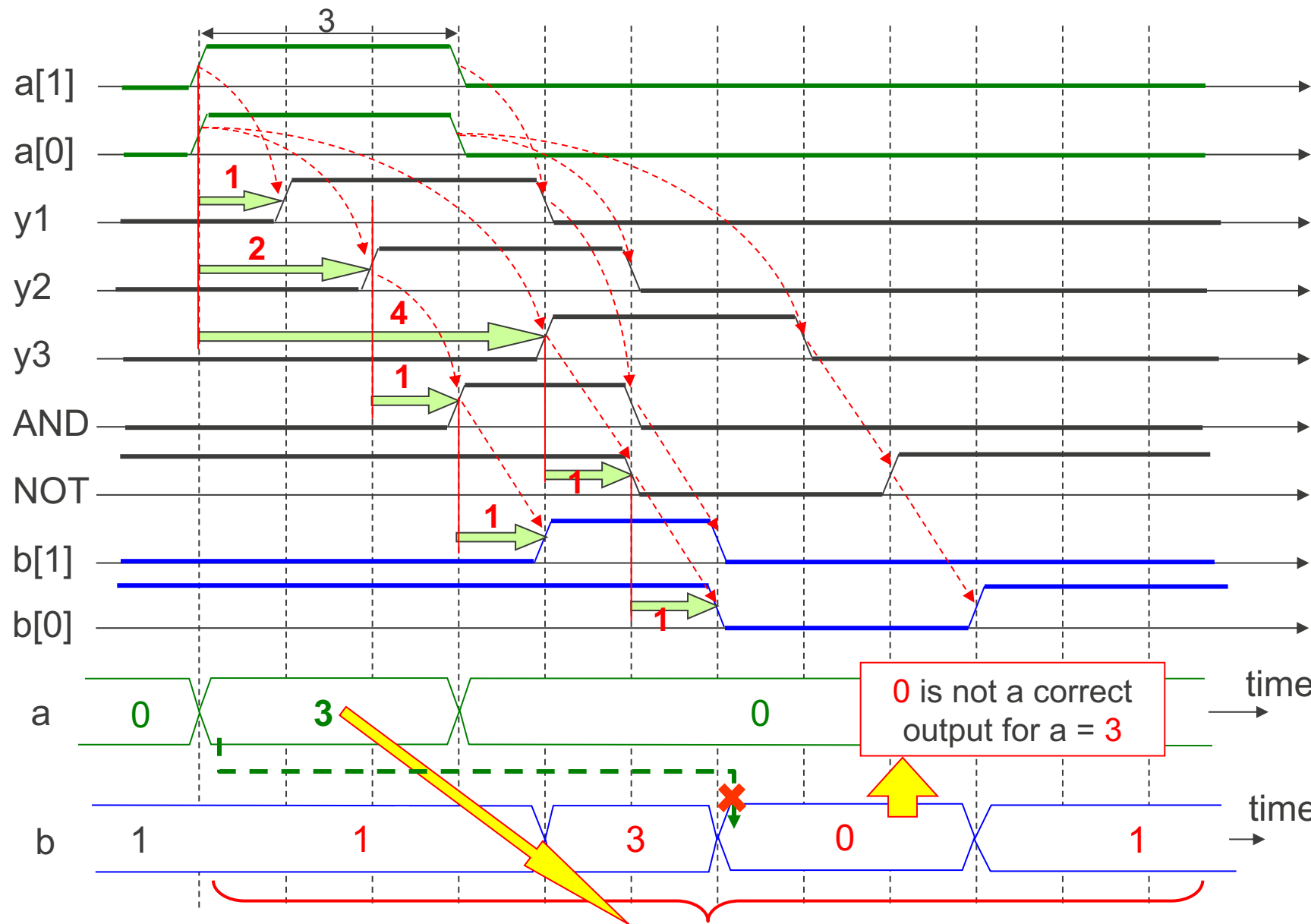


However, delay is not an only problem we will have for timing issues.

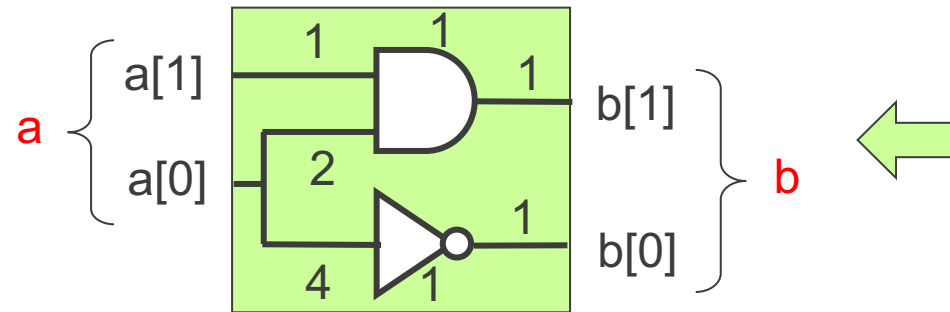
We will have much **serious problem**, if **input changes in a short time**.
Let's see what happens, if input changes as follows.



Input is 3 for just **3** time units



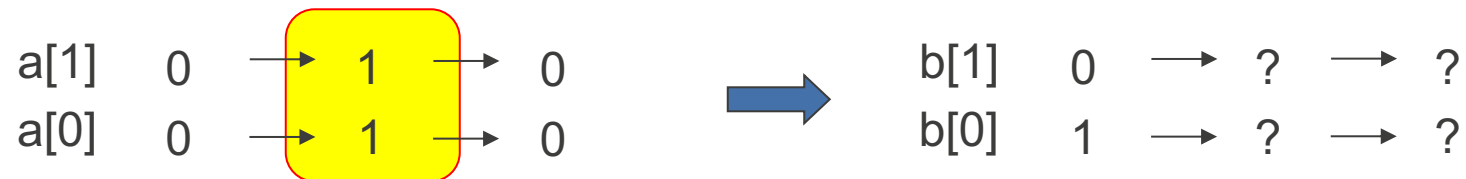
The result shows that if the input changes in a short time, then **the correct output will never come out.**



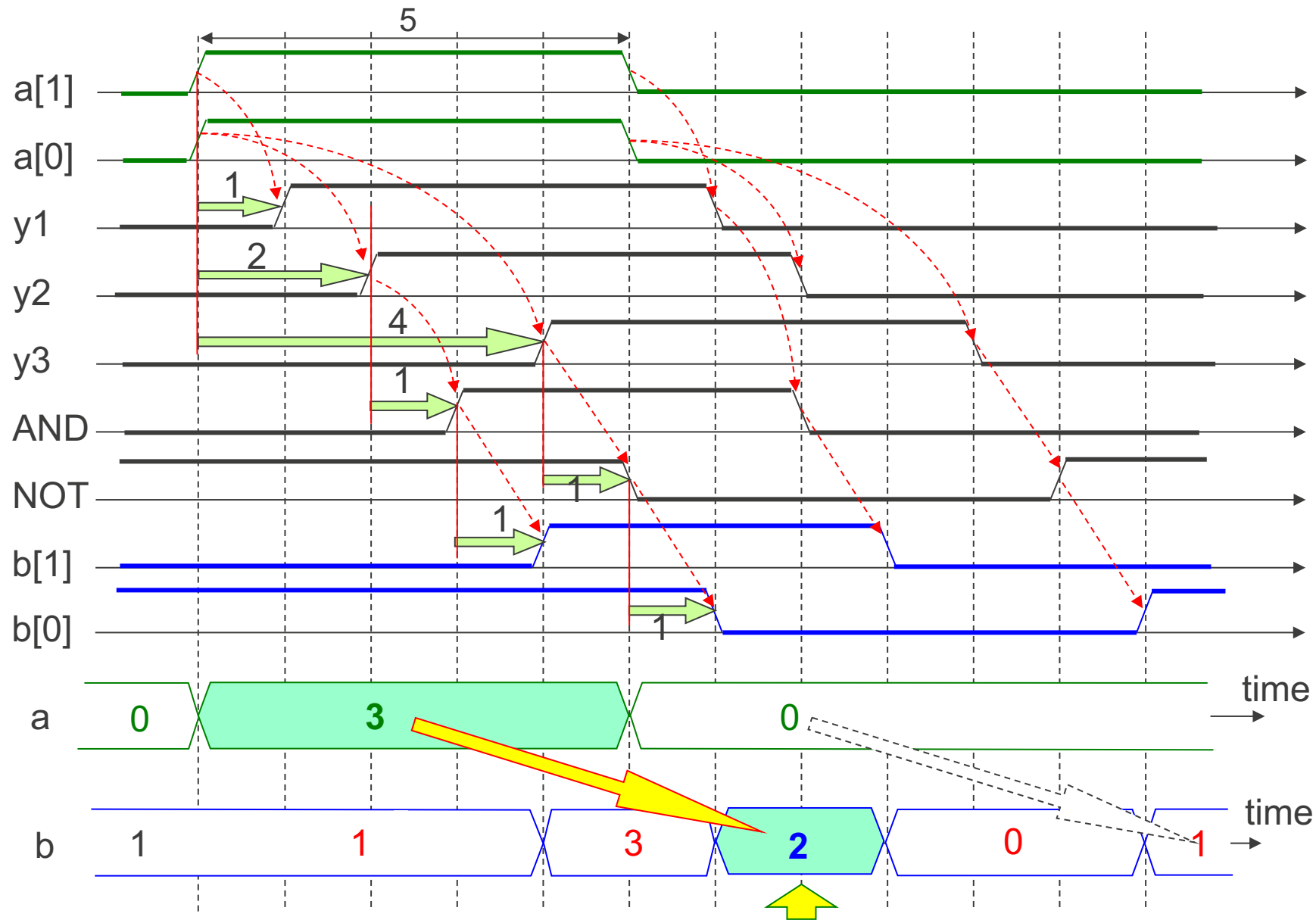
For the given delay, the logic block on the left **can not output correct value** if input changes in **no** more than 3 time units.

In general, we have to keep input variable stable for more than **maximum delay - minimum delay** of the logic block, otherwise the correct output will never appear at the output port.

Let's confirm the above idea by keeping the input for 5 time units unchanged and observe the output value.

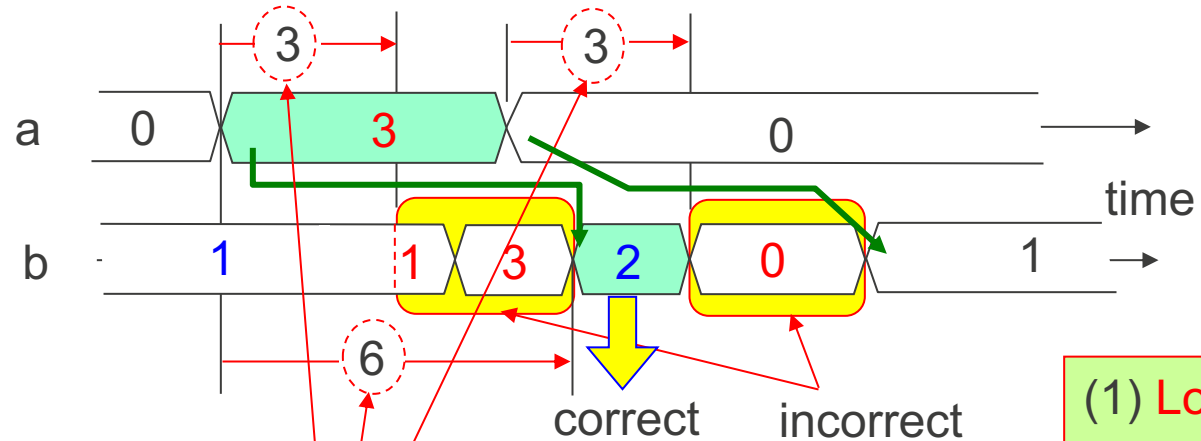


Input is 3 for just **5** time units



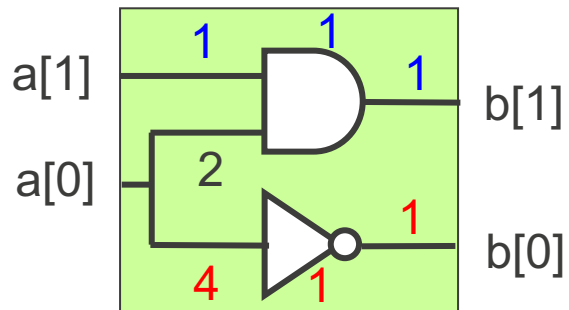
The correct value "2" appears on the output

The result on the previous slide can be summarized as follows.



Longest path delay = 6

Shortest path delay = 3



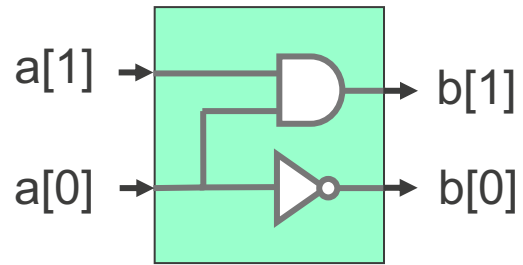
(1) **Longest path delay**: The time needed for the correct output to appear at the output.

(2) **Shortest path delay**: The correct output will go away after this time passes if the input changes.

Therefore,

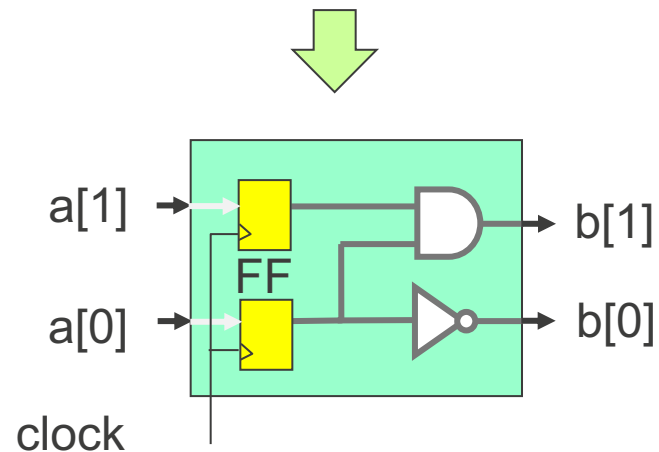
(3) **Longest path delay - Shortest path delay**:

- (a) The input must be kept unchanged during this amount of the time period, otherwise the correct output will never appear at the output.
- (b) The time while incorrect output appears at the output because of the hazard.

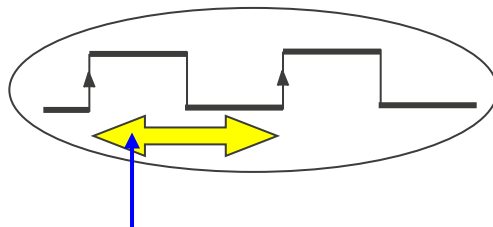


For the logic block on the left, it is mandatory to **keep the input unchanged** for (longest - shortest) delay time to get the correct output.

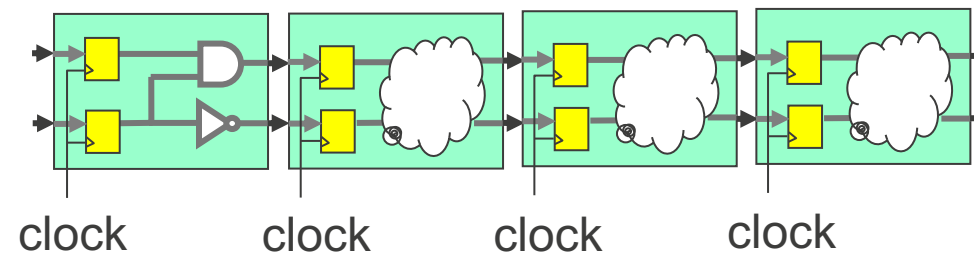
Placing FFs at the input of the logic block is the most simple answer for this issue.

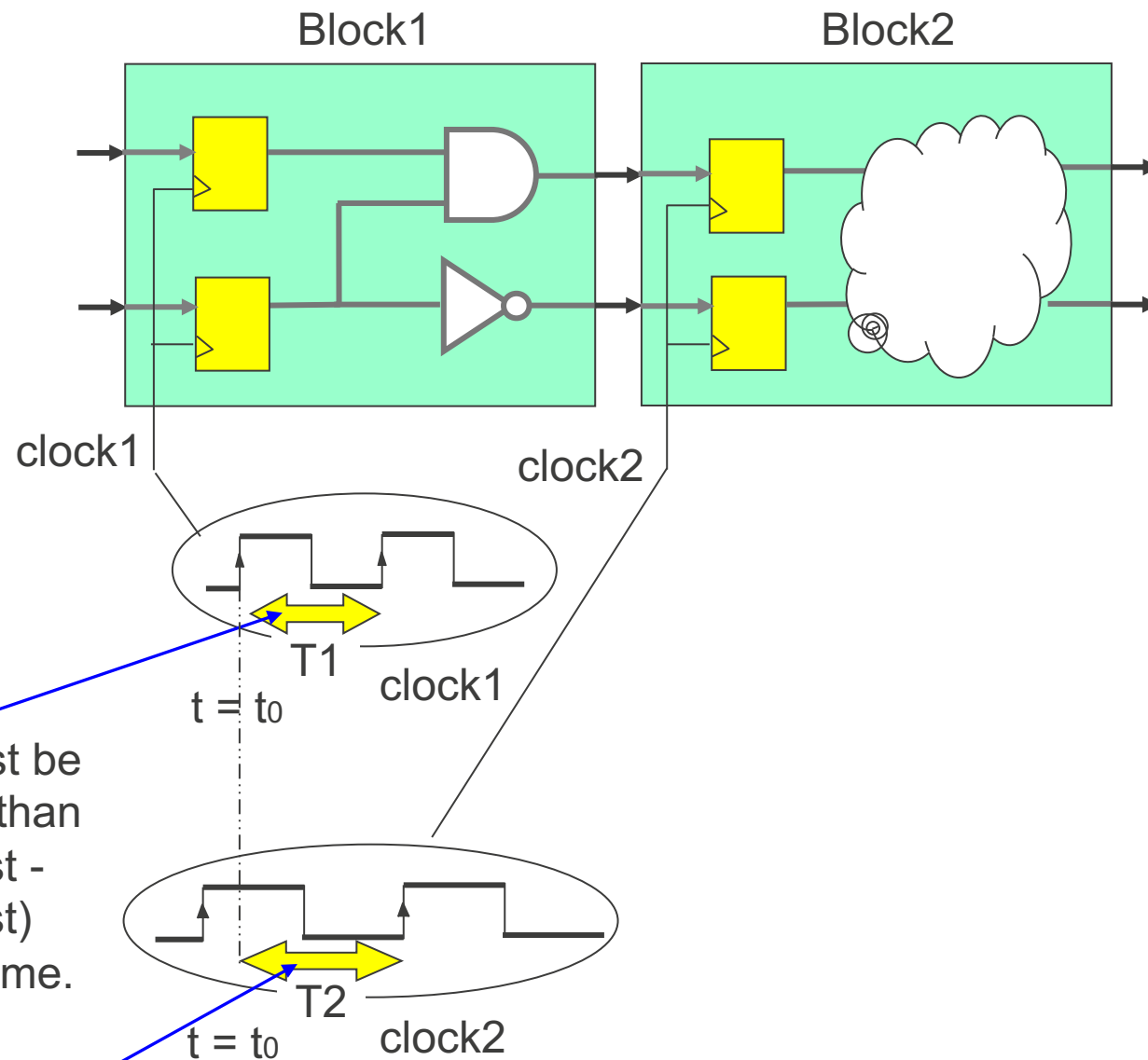


Now, study about the cases to build a larger scale system by combining many blocks as shown below.



This must be longer than (longest - shortest) delay time.

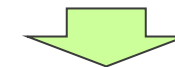




T1 must be longer than (longest - shortest) delay time.

T2 must be longer than the longest delay time.

If we are to use different clocks such as clock1 and clock2 to Block1 and Block2 respectively, it is almost **impossible to control clock rise time** so that T1 is always longer than (longest - shortest) delay time and T2 is always longer than the longest delay time.

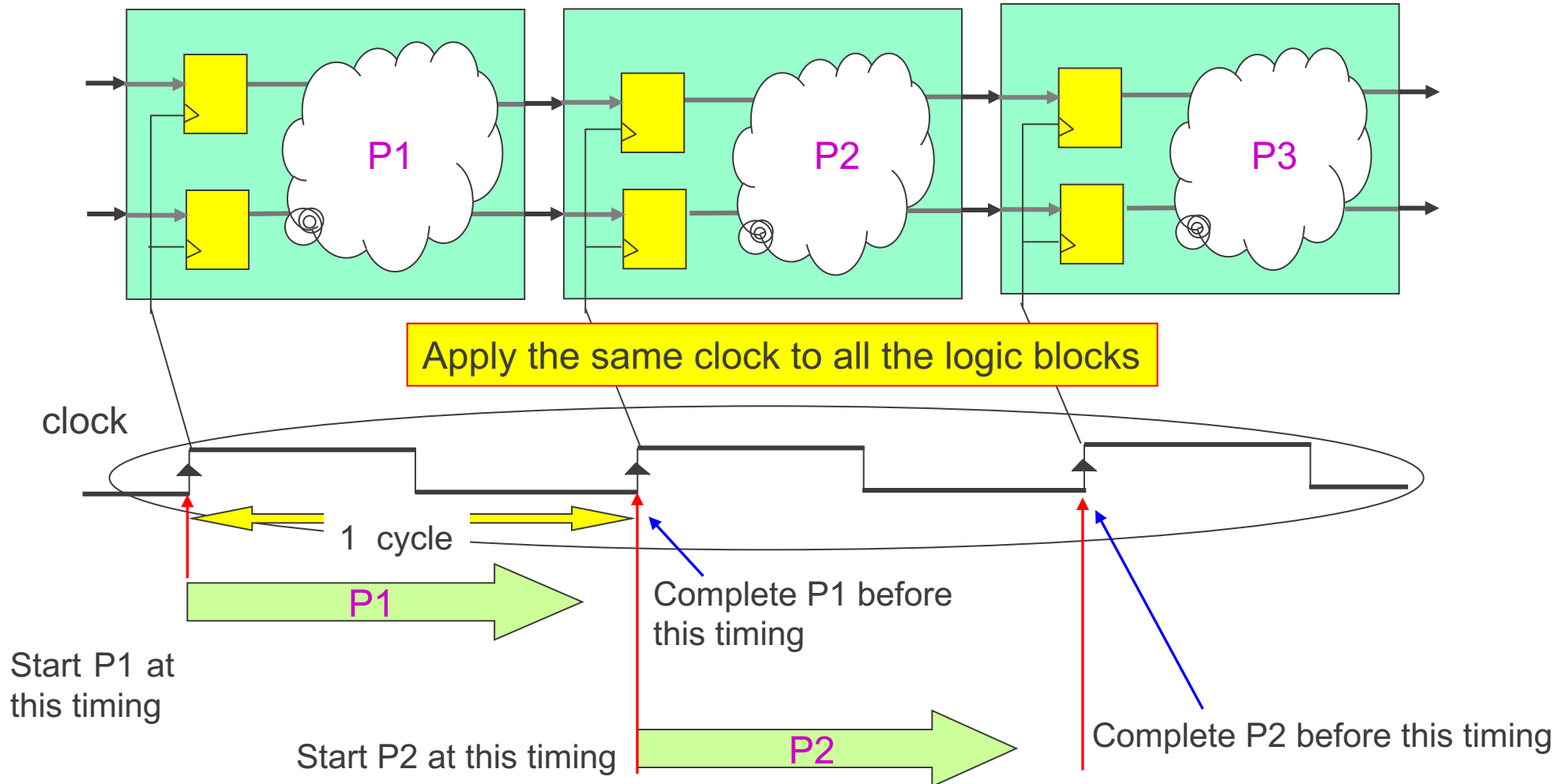


Using the same clock for all the blocks is a solution.

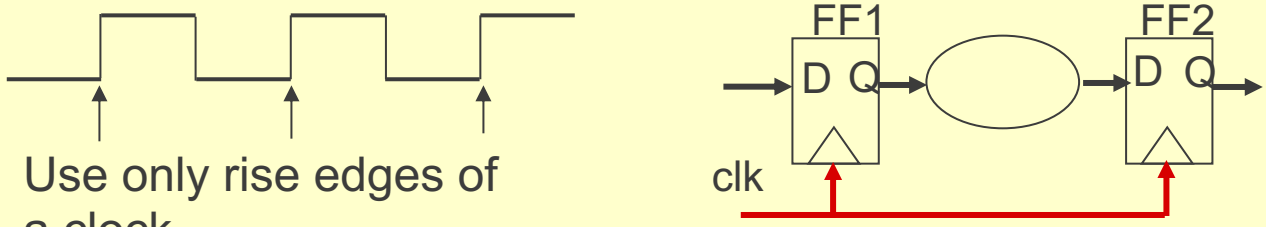
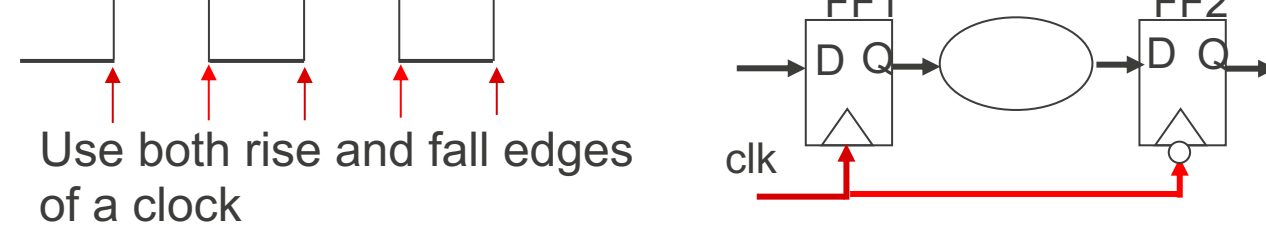

4.2 Synchronous Design and Static Timing Analysis (STA)

Synchronous Design and STA

Synchronous design uses one clock for all the logic blocks and capture the signals at clock rise time to make them unchanged during one cycle for the processing.

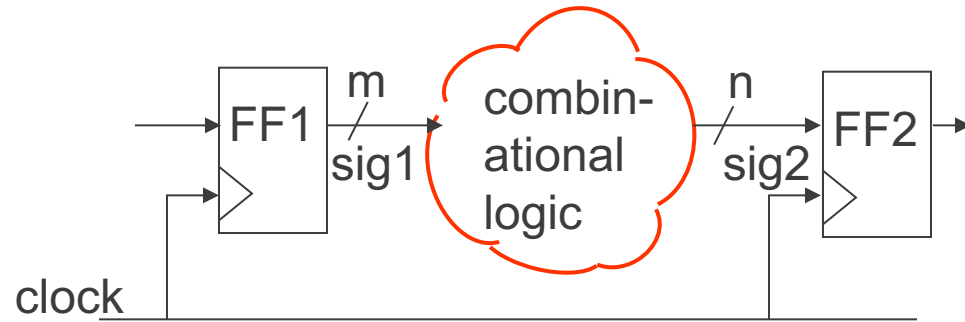


There are several ways to use clocks in synchronous design as shown below. However, using one clock at its rise time is recommended.

<p>Single phase clock</p> <p>One cycle transfer</p>	 <p>Use only rise edges of a clock</p>
<p>Single phase clock</p> <p>Half cycle transfer</p>	 <p>Use both rise and fall edges of a clock</p>
<p>Two phase clock</p>	 <p>Use rise edges of different clocks having different phases.</p>

Not recommended to use.

Timing constraints in synchronous design can be summarized as below.

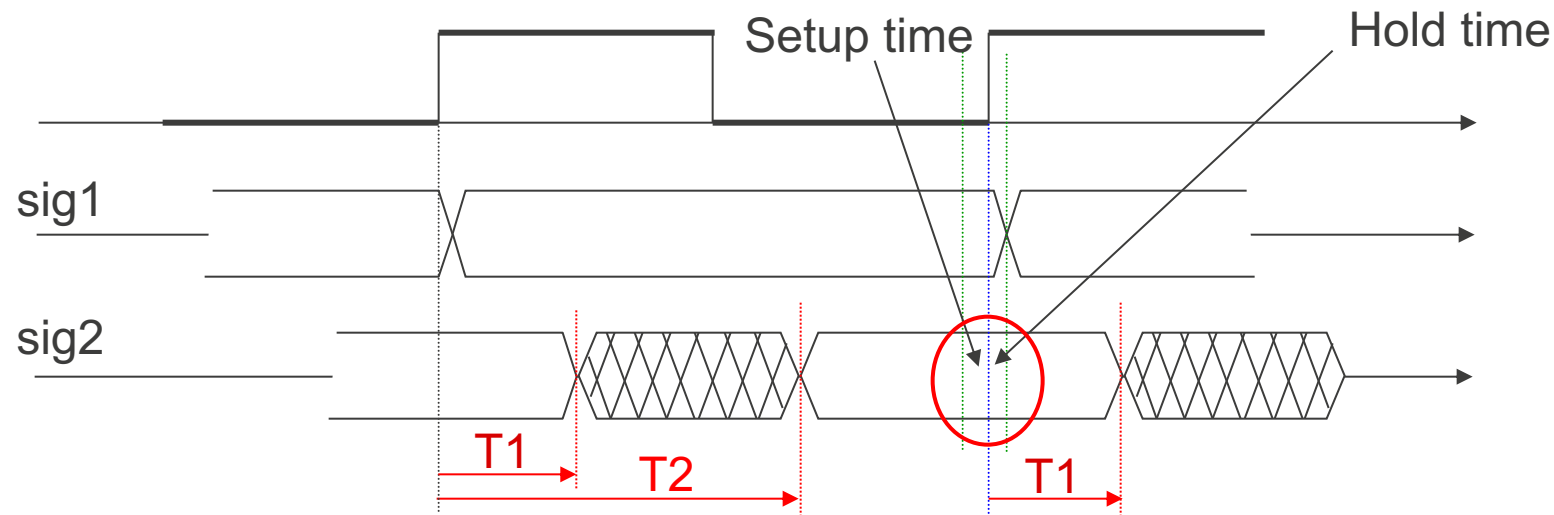


Timing constraints

- (1) T2 must be shorter than one cycle.
- (2) sig2 must not change during Setup and Hold time of FF2 to avoid metastability of FF2.

T1: shortest path delay of the combinational logic

T2: longest path delay of the combinational logic

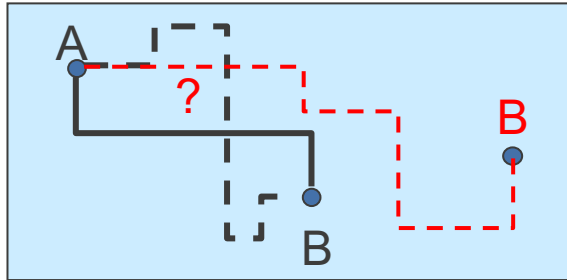


$T1 > \text{Hold time}$

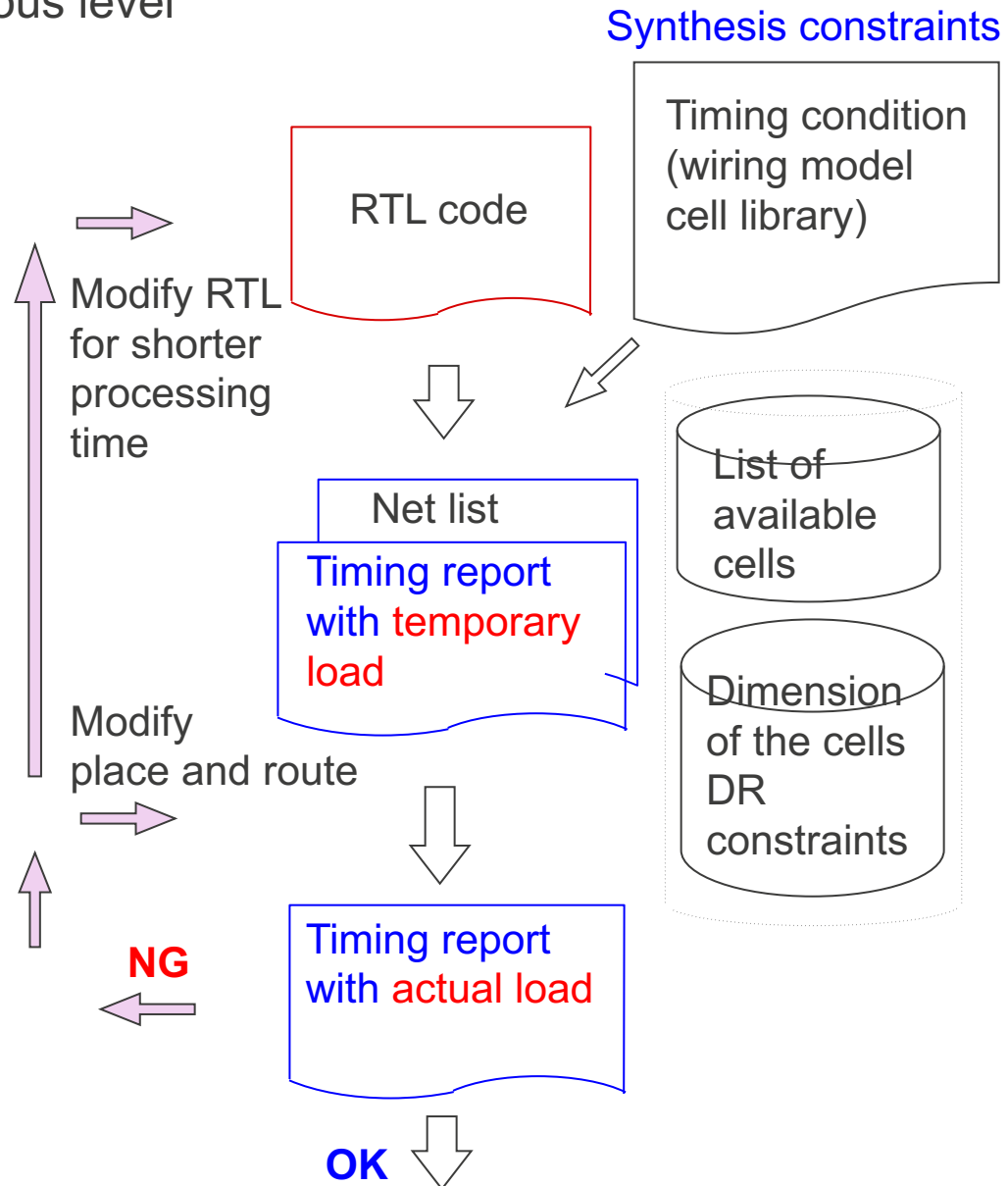
$T2 + \text{Setup time} < 1 \text{ cycle}$

STA is carried out several times by using various level of accuracy of delay data.

After layout (place and route), we must check the timing because delay from A to B is very much different depending on where A and B are placed and connected by which route.



Final check is done based on the actual load, which takes account electronic characteristics of wire lines.

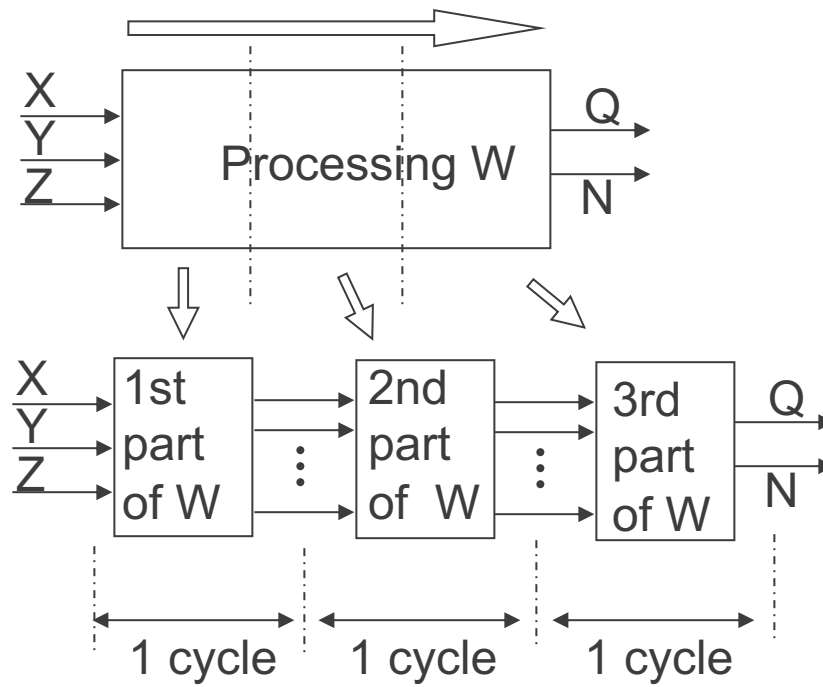


Dividing a Process and Multi-Cycle Path

Next, a question may be raised that “What shall we do if an assumption that every process can be done in one clock cycle does not hold?”

⇒ A countermeasure (1)

Divide the process into several blocks so that each block can be completed in one clock cycle.



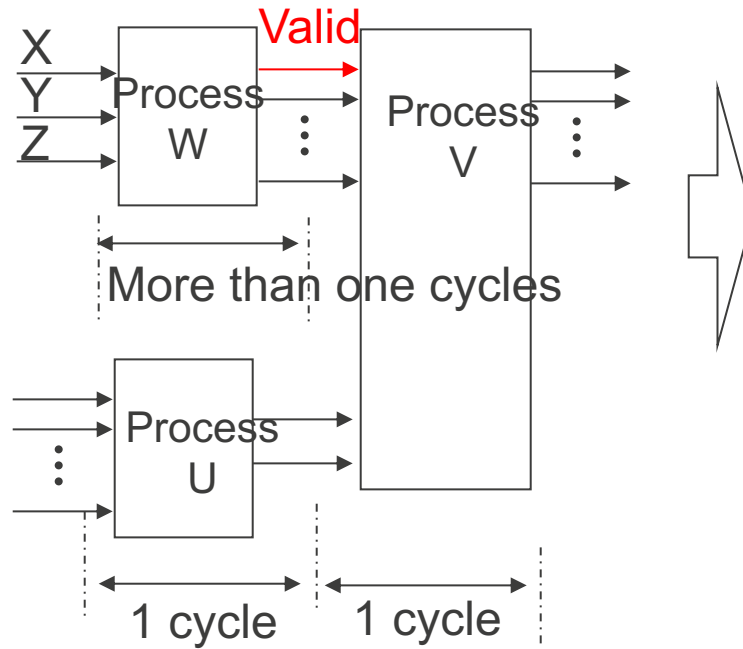
This might be a good solution for pipeline type process.

You may use multi cycle path specification of DA tool instead of dividing the process.



Another countermeasure (2)

For data where processing cannot be ended within 1 clock cycle, introduce some signal such as valid or ready to notify the block waiting for the data that the data is ready.

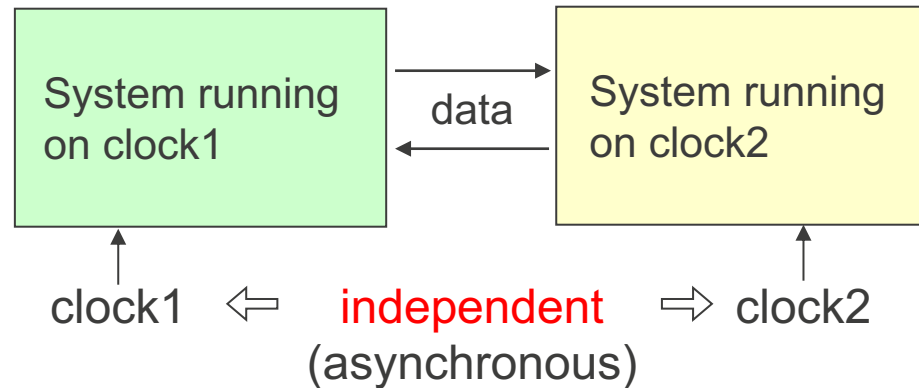


This means introducing asynchronous logic partially.

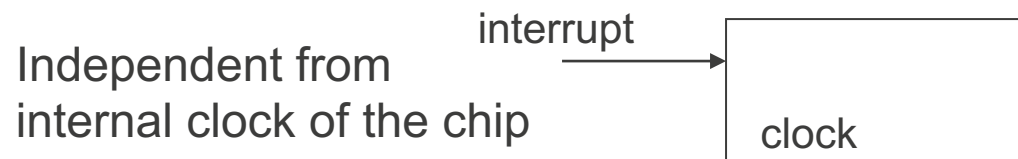
Asynchronous Data Transfer

Synchronous design makes timing issue very simple. However, it cannot be applied to every system. We have to introduce asynchronous design in some cases. Typical cases are:

- Connecting two system running on independent clock

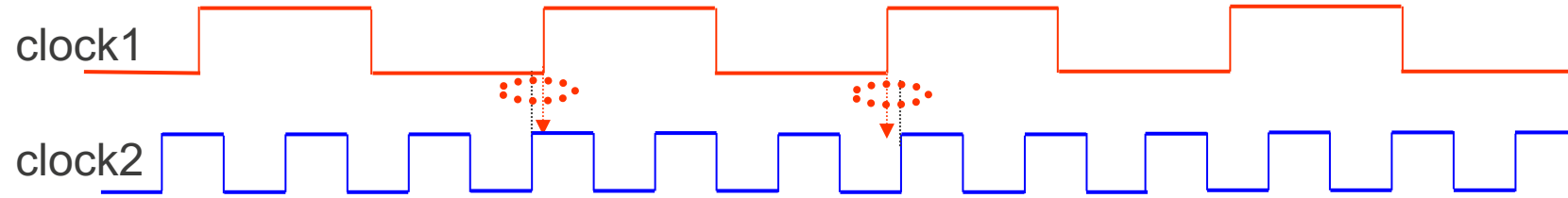


- Getting asynchronous data, such as interrupt, from outside the chip



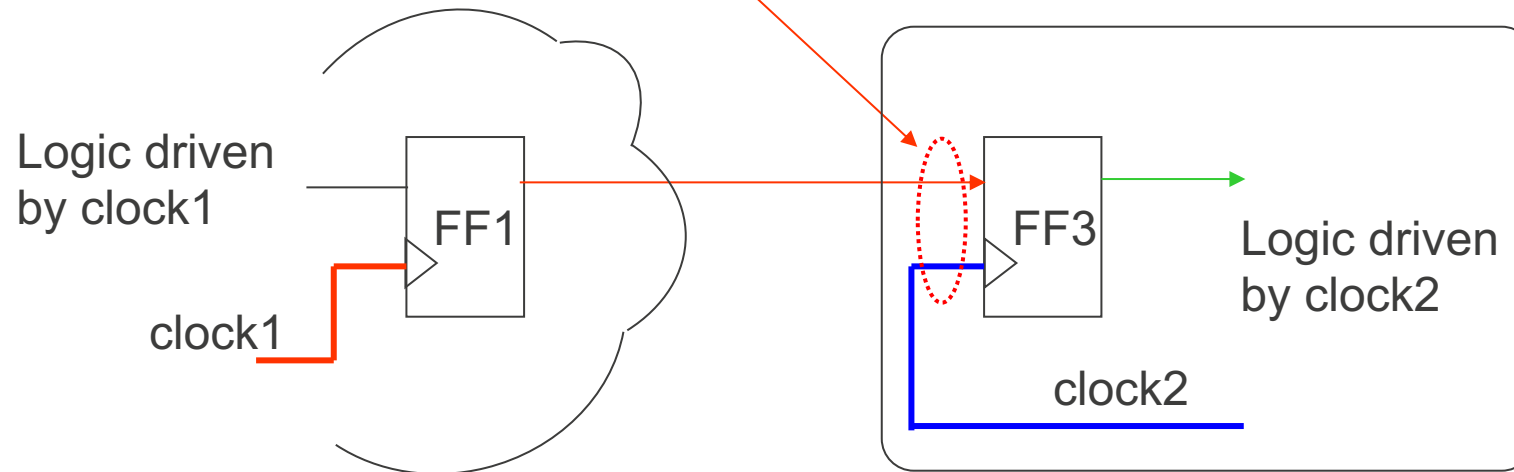
Connecting two system running on independent clock

⇒ Data transfer among the logic with different clock systems

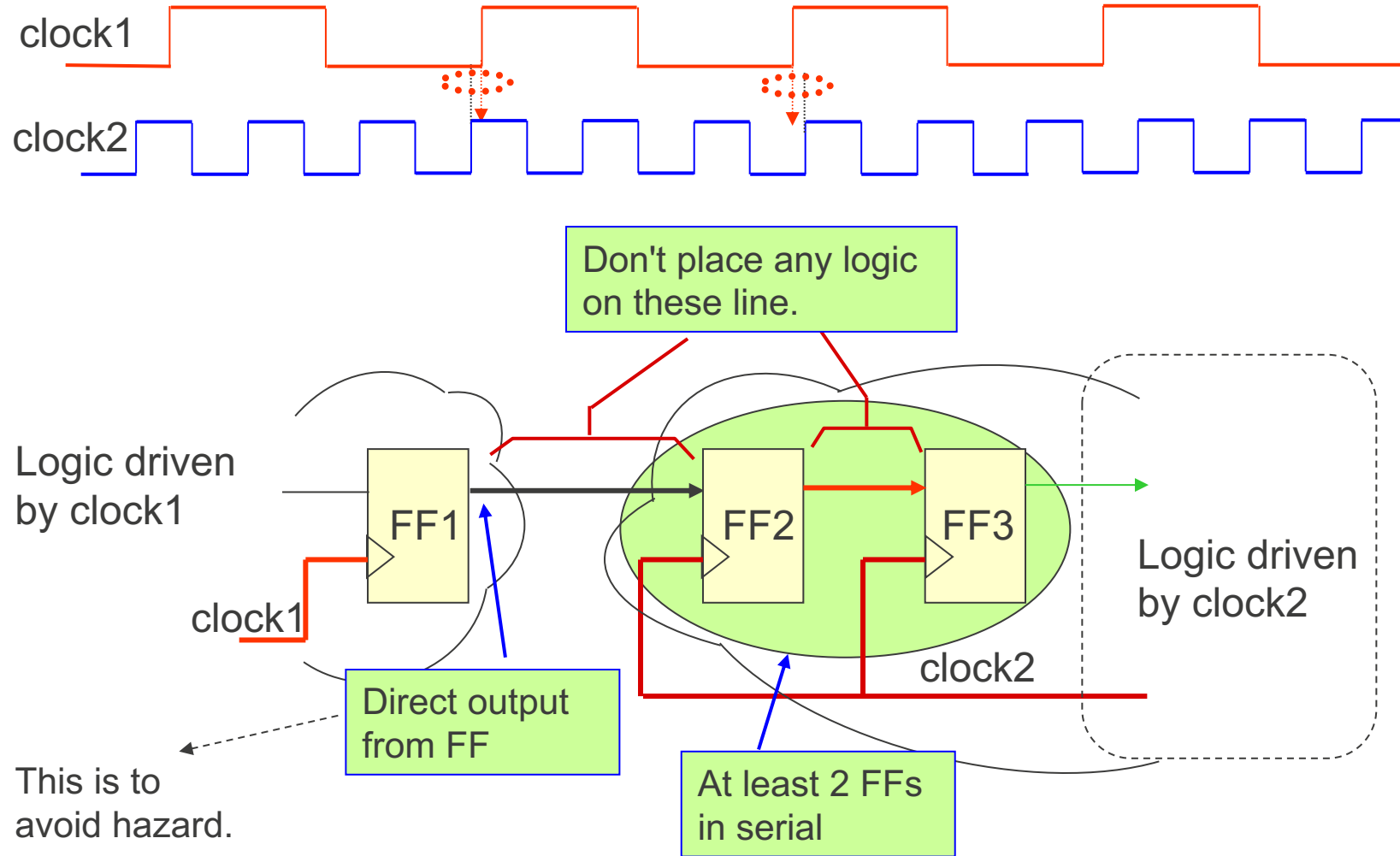


Because clocks 1 and 2 are running without taking any synchronization, their rise edges may collide.

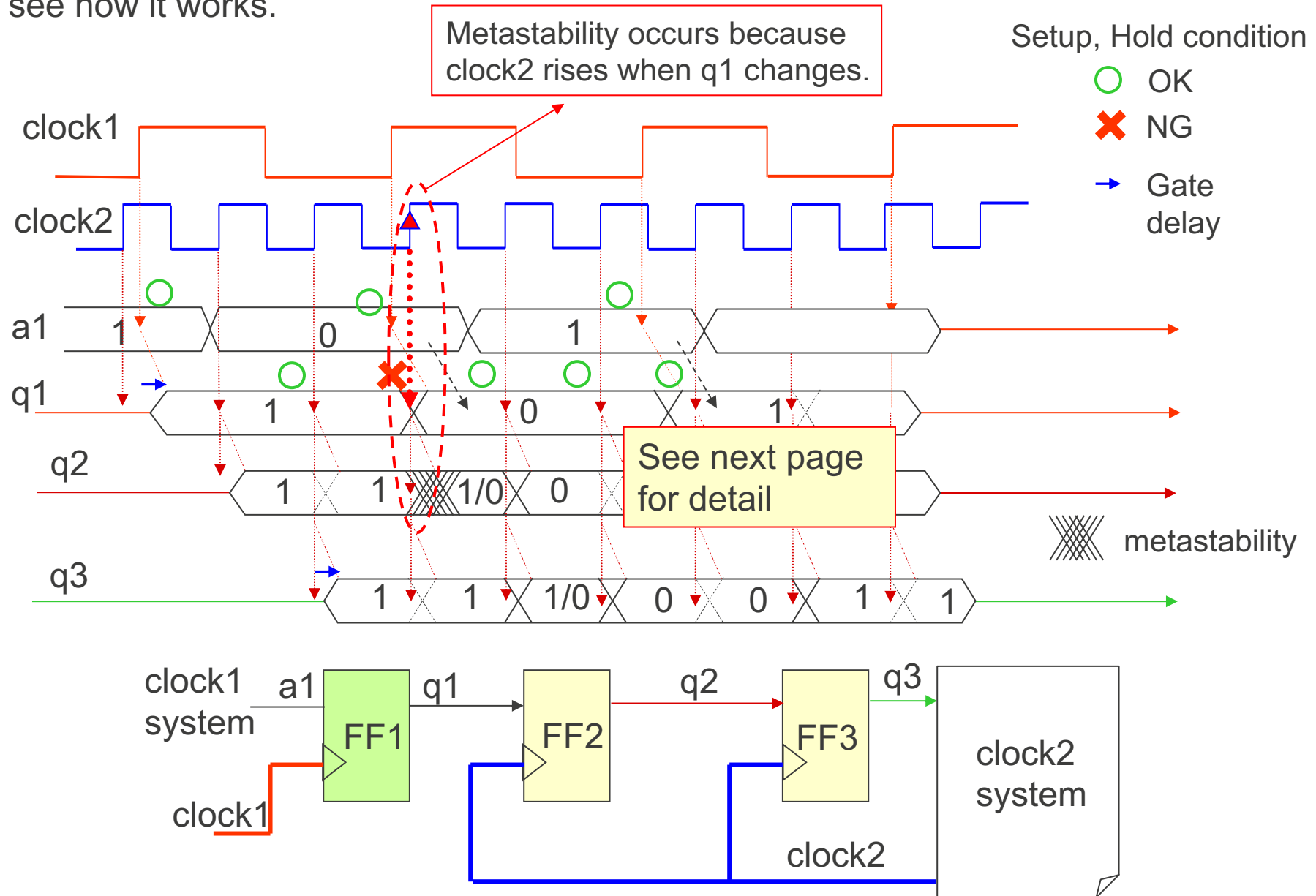
Therefore, FF3 getting data from clock 1 system may fall in metastable state.

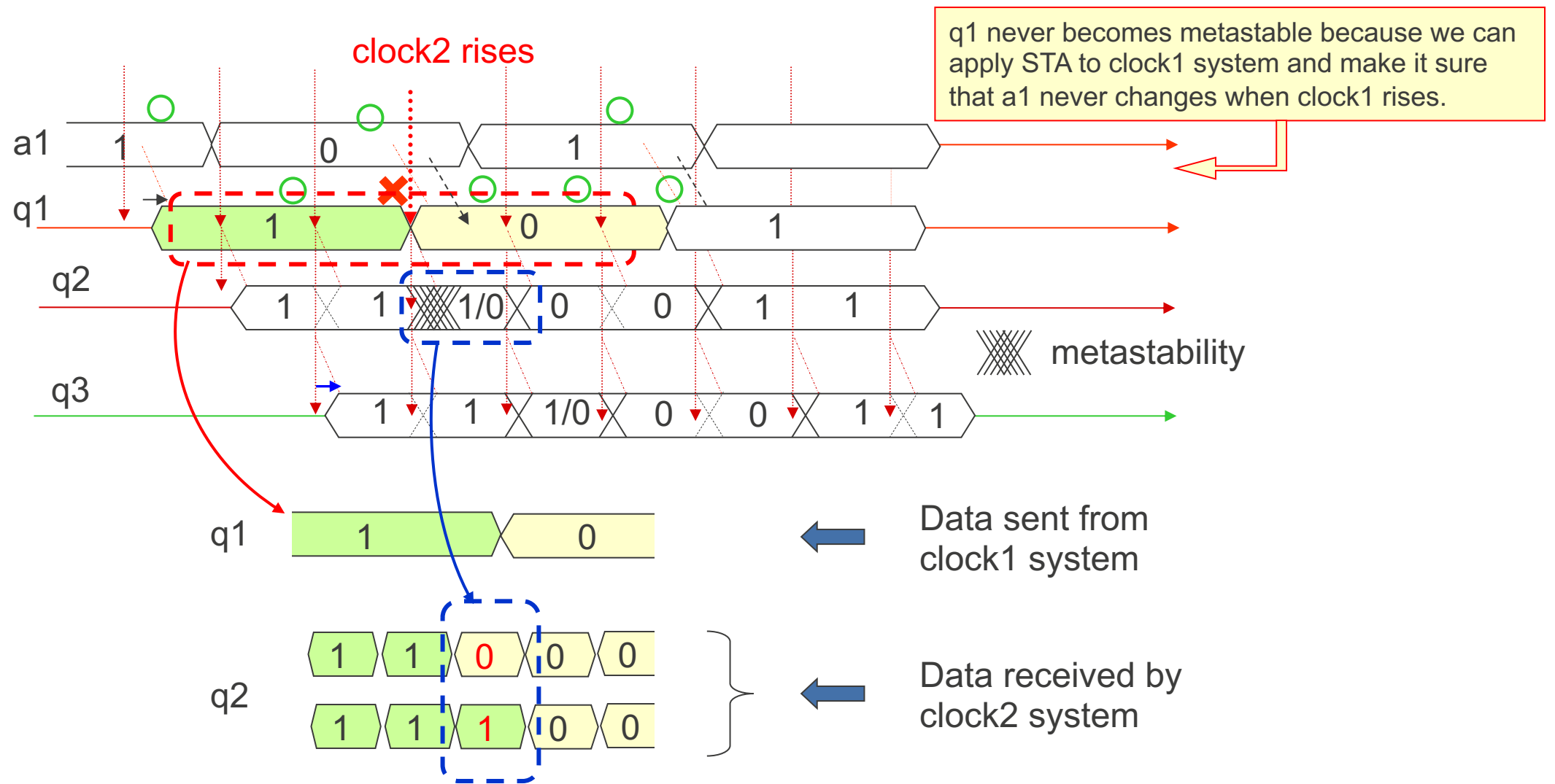


Placing FFs in between clock1 system and clock 2 system as shown below is one of the known solution for the problem.



Now let's see how it works.





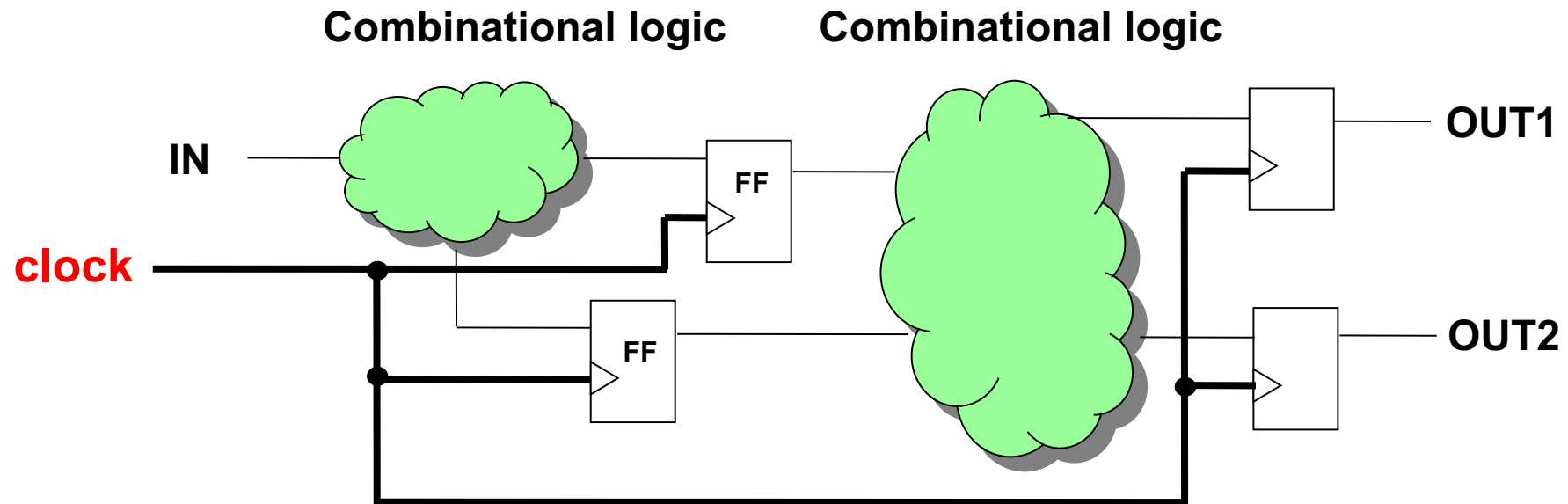
Sending data's change from 1 to 0 is correctly detected by the receiving system because q2 changes from 1 to 0 correctly even if metastability occurs.

4.3 Synchronous Design Summary

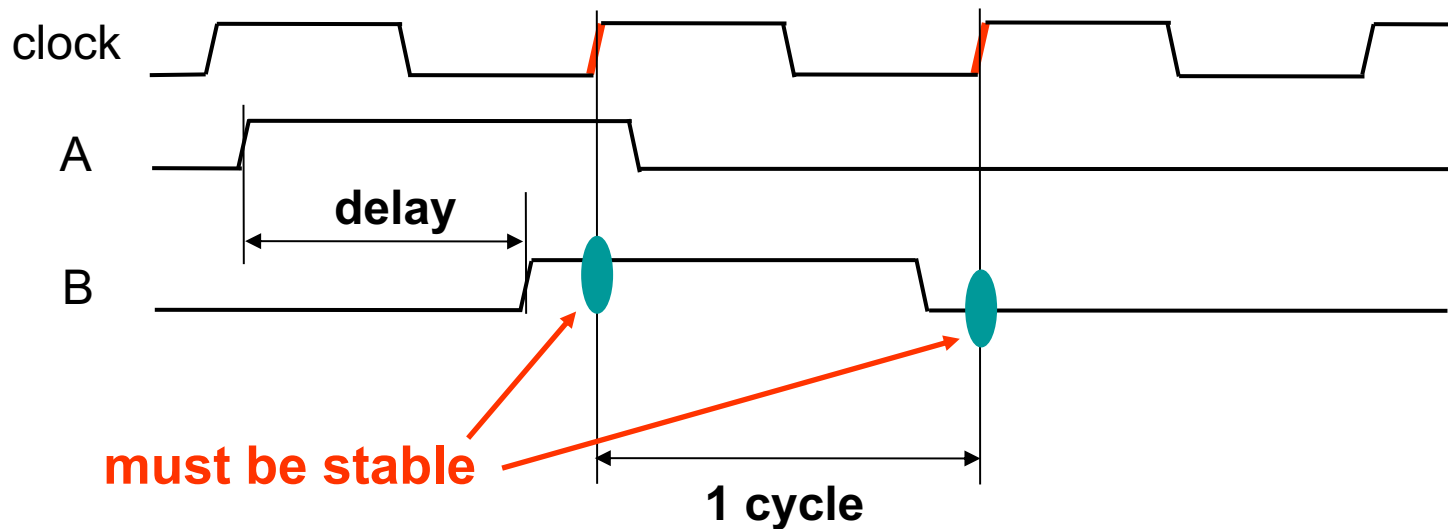
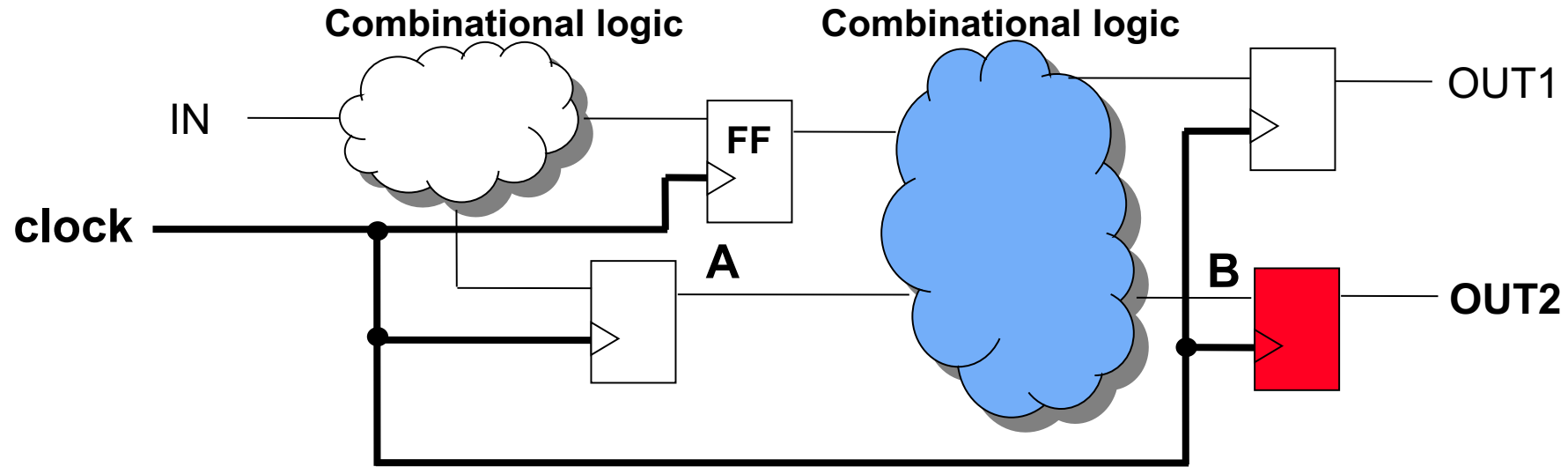
Implementation

All operations of one circuit block of interest are **uniquely defined in synchronous design** with base clock signal.

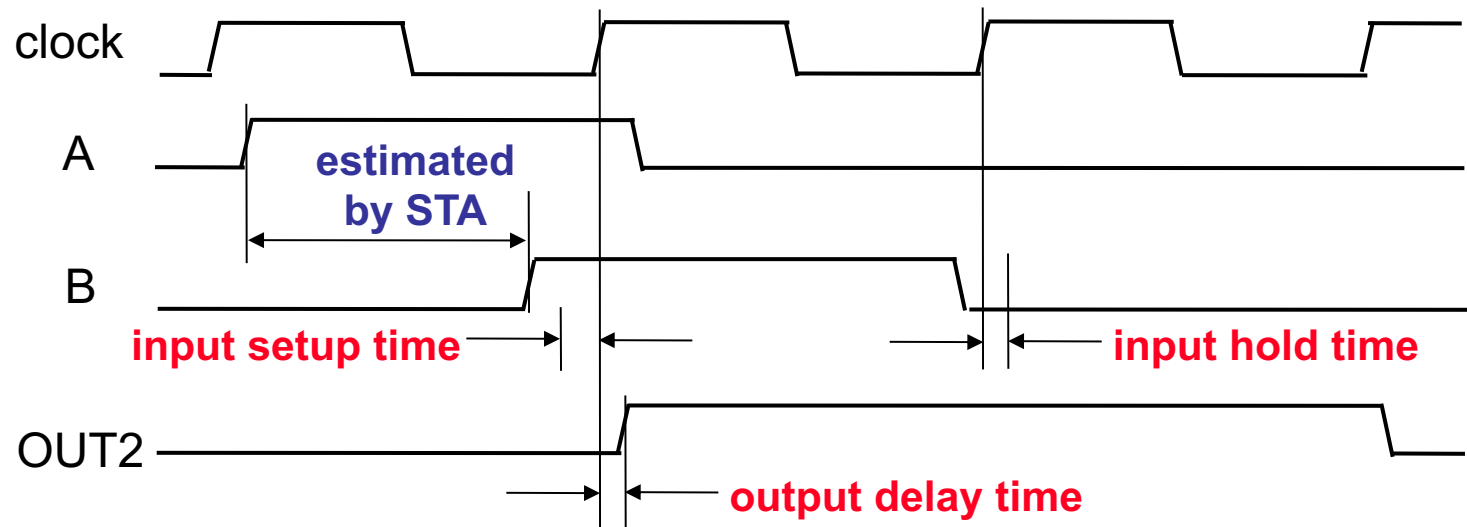
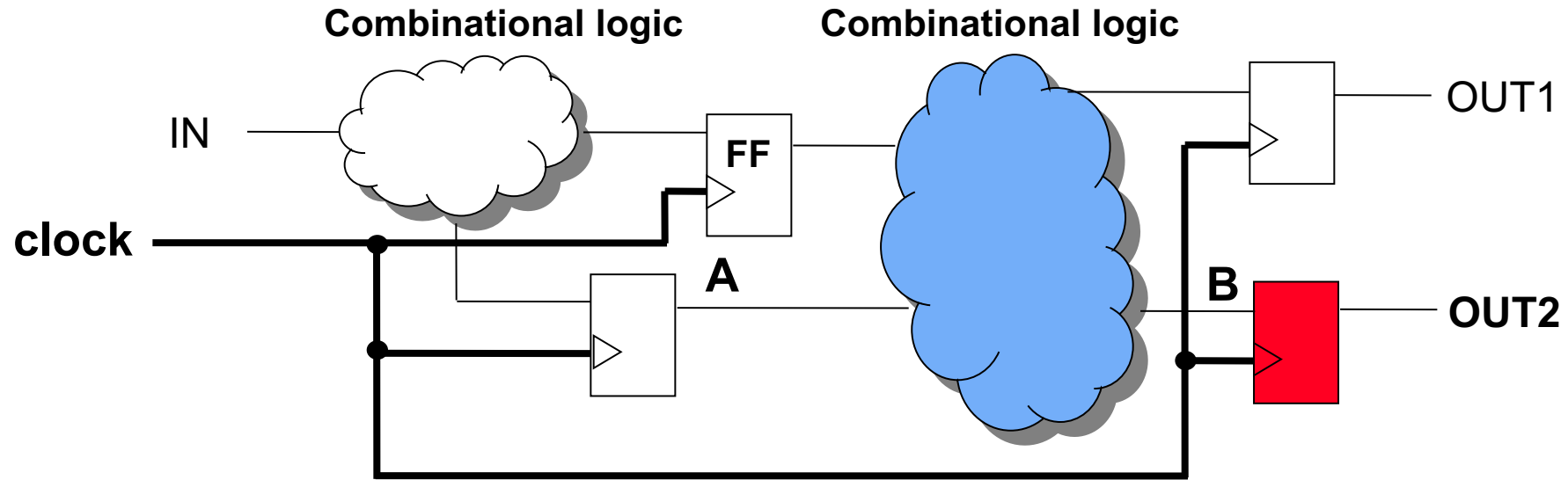
RTL design ...fully synthesizable design, configured with single clock signal and registers (FF)



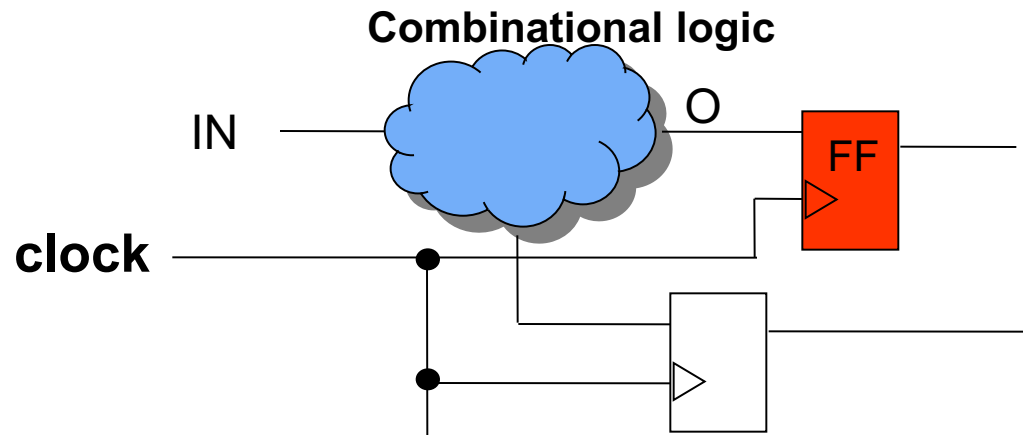
One Cycle Operation



Synchronous Design



Timing Requirements



Requirements

$$T_{cyc} - T_{su} > T_{pd} > T_{ho}$$

○ : timing margin

Timing specifications

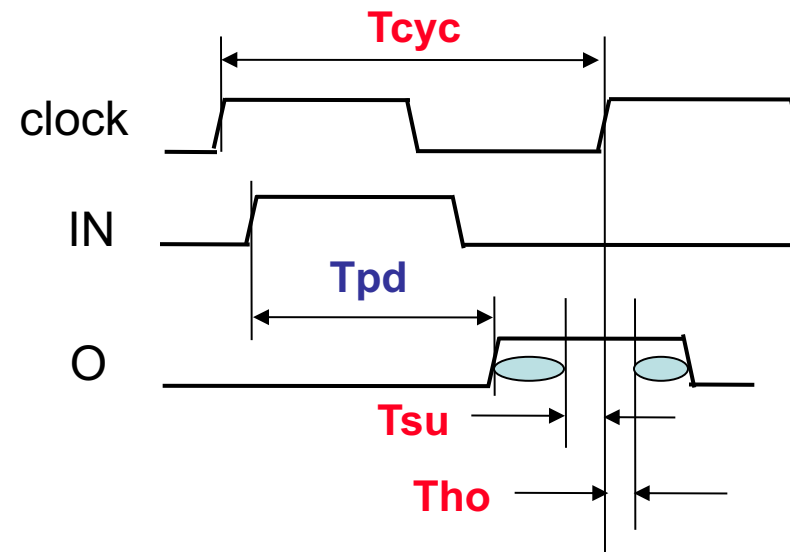
T_{cyc} : clock cycle time

T_{su} : input setup time

T_{ho} : input hold time

Estimated by STA

T_{pd} : propagation delay time



[Renesas.com](https://www.renesas.com)