Digital IC Design Concept

賴譽仁

thinker.lai@gmail.com

Outline

- Asynchronous Design
- High Speed Design
- Low Power Design

NTUST

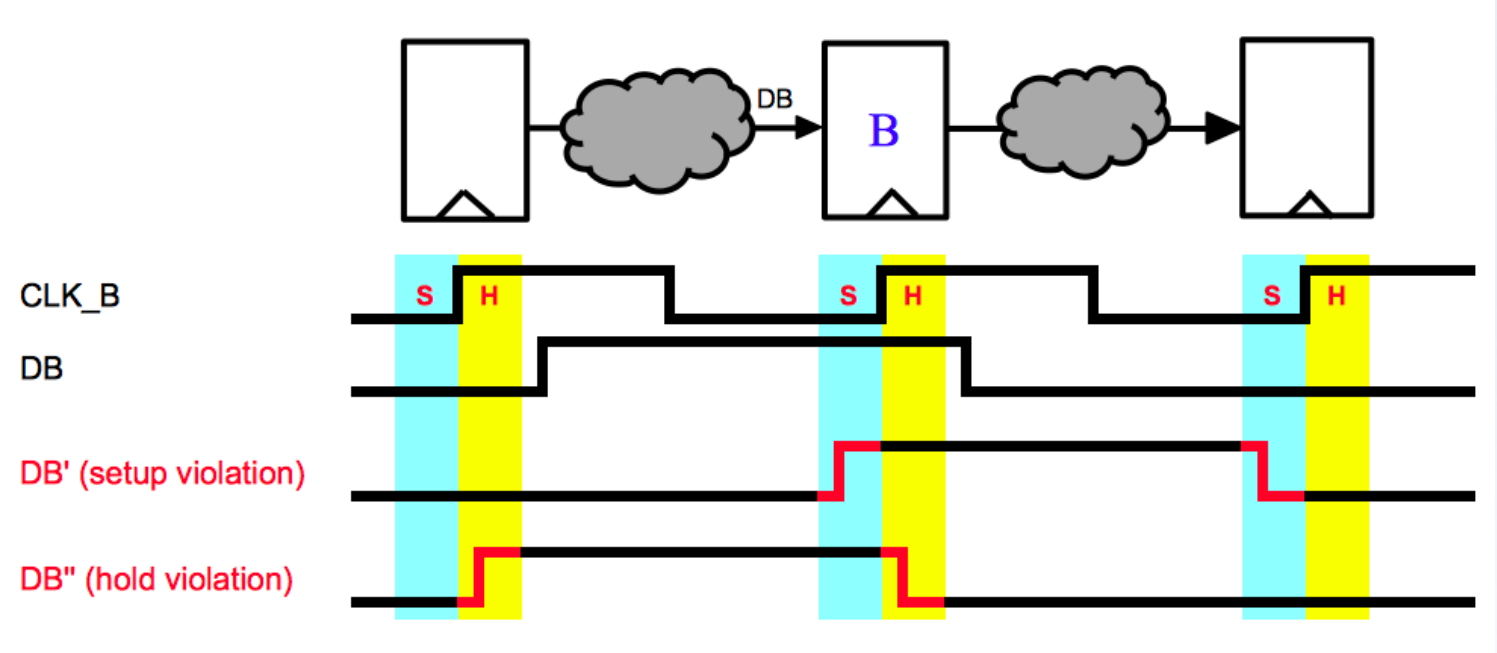
Asynchronous Design

- What is Metastable
- How to Handle Asynchronous
- Single-Bit Synchronous
- Multi-Bit Synchronous
- Asynchronous Reset

NTUST

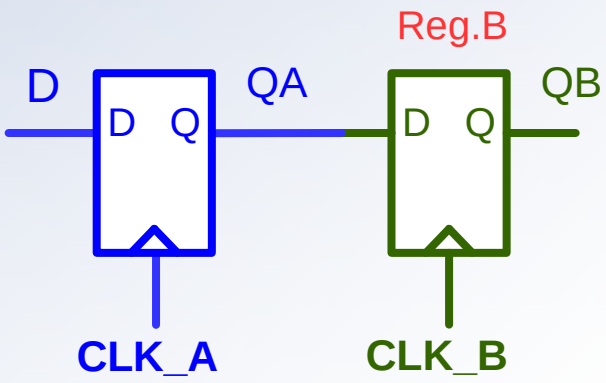
What is Metastable

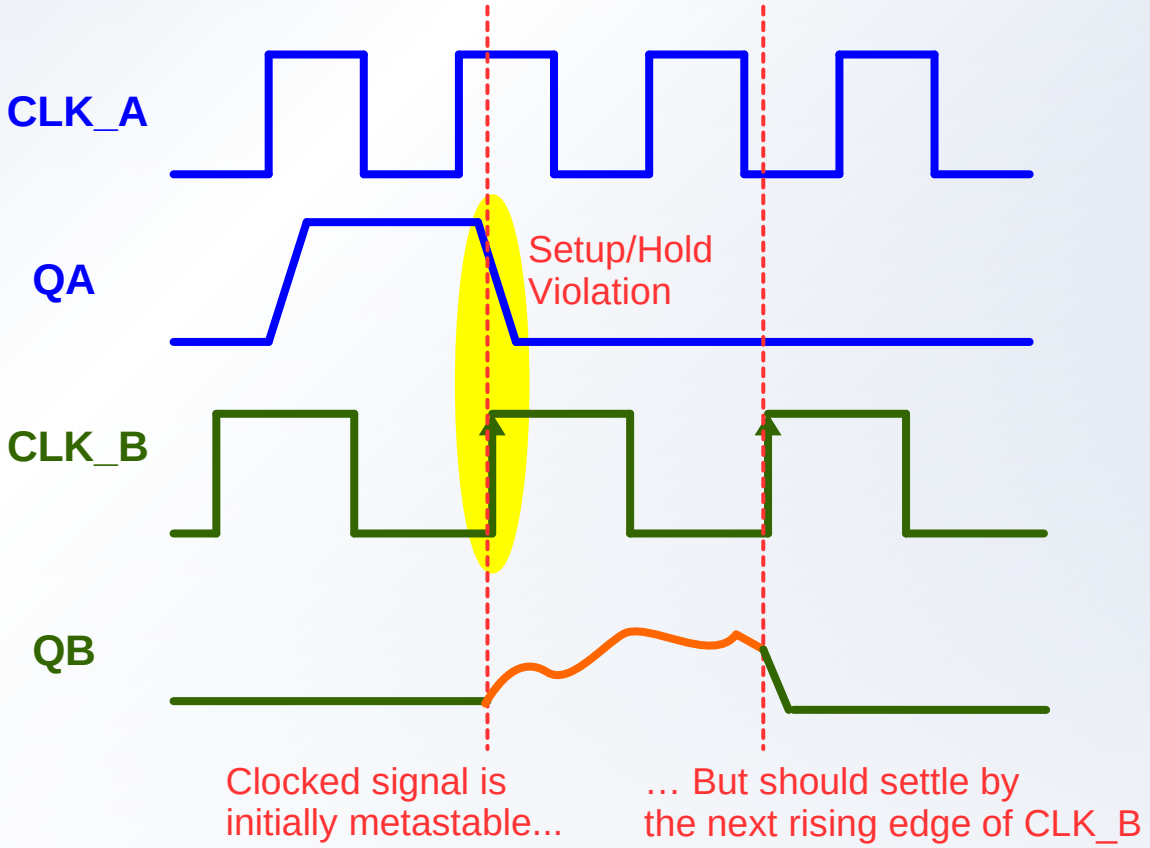
- Signals crossing asynchronous domains create metastability.
- Metastability refer to signal that do not assume stable 0 or 1 states for some duration of time at some point during operation of a design.
- In a multi-clock design, metastability can't be avoided but the detrimental effects of metastability can be neutralized.



What is Metastable

- Signal crossing asynchronous domains create metastability

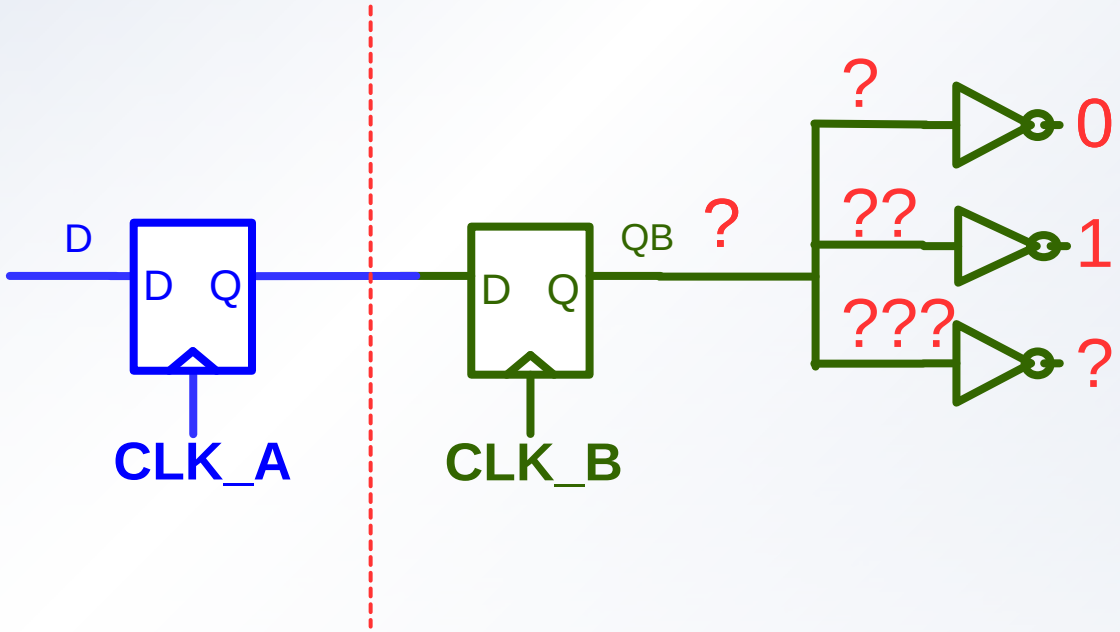




Potential Metastability occurs due setup or hold time violation In Reg.B

Non-predictable Signal

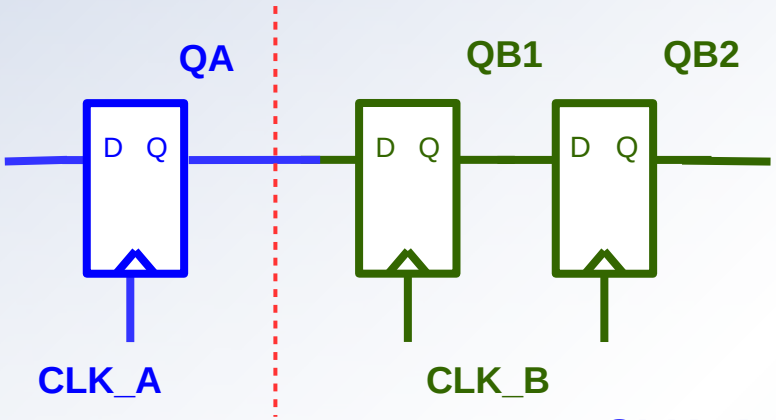
- Metastability refer to signal that do not assume stable 0 or 1 states, cause to design function error.



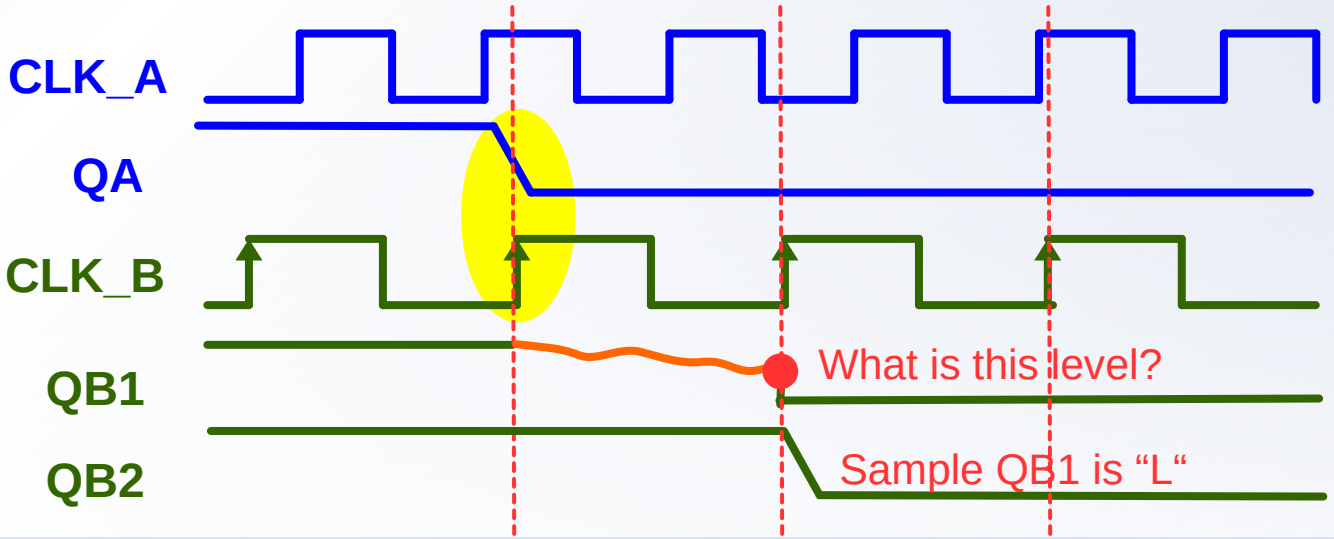
How to handle asynchronous

NTUST

- Two flip-flop synchronizer solution
  - Metastable signal needs time to settle to a stable value
  - Additional flop gives the metastable signal time to become stable



延伸閱讀：**MTBF**
(Mean Time Between Failures)



Other Possible Issue with Synchronization

- After synchronizer, every thing is OK??
  No !!! some possible issue need be carefully

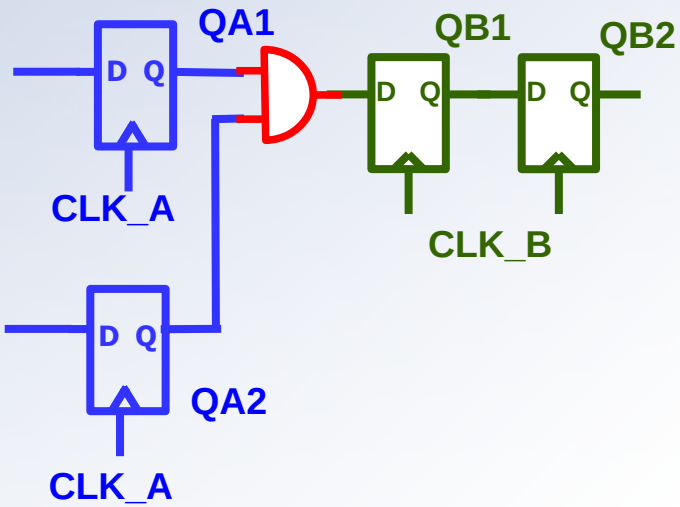
- Structural
  - Glitch issue
  - Multi-Fanout issue

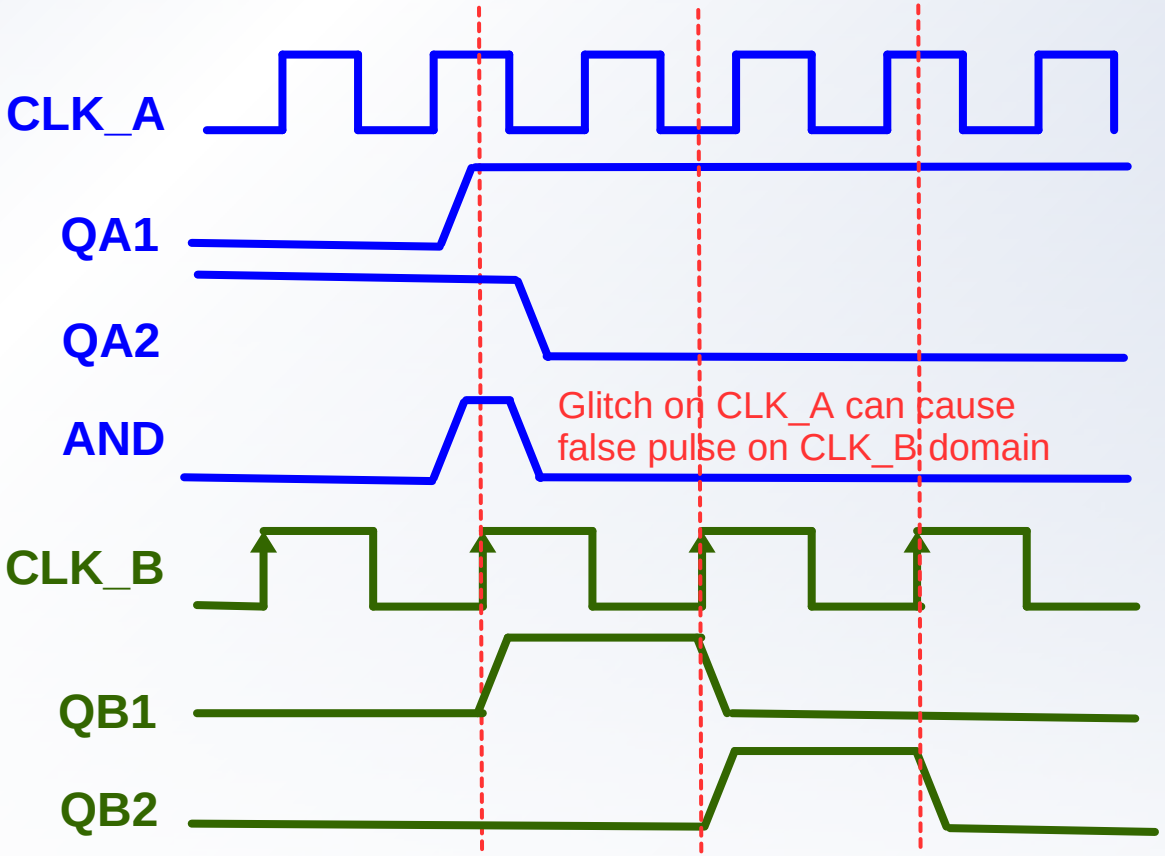
- Functional
  - Data Stability issue

[Structural] Glitch Issue

NTUST

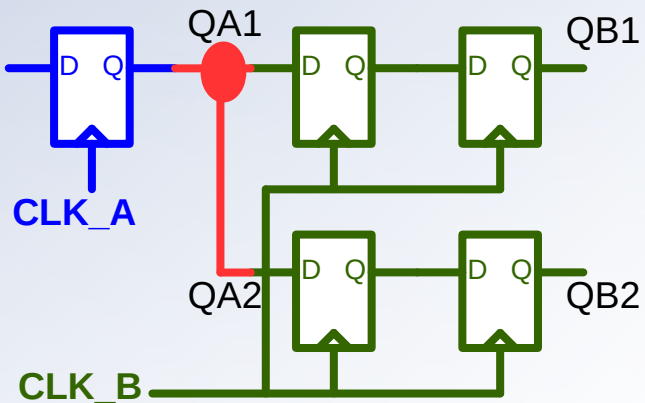
- Glitch due propagation delay

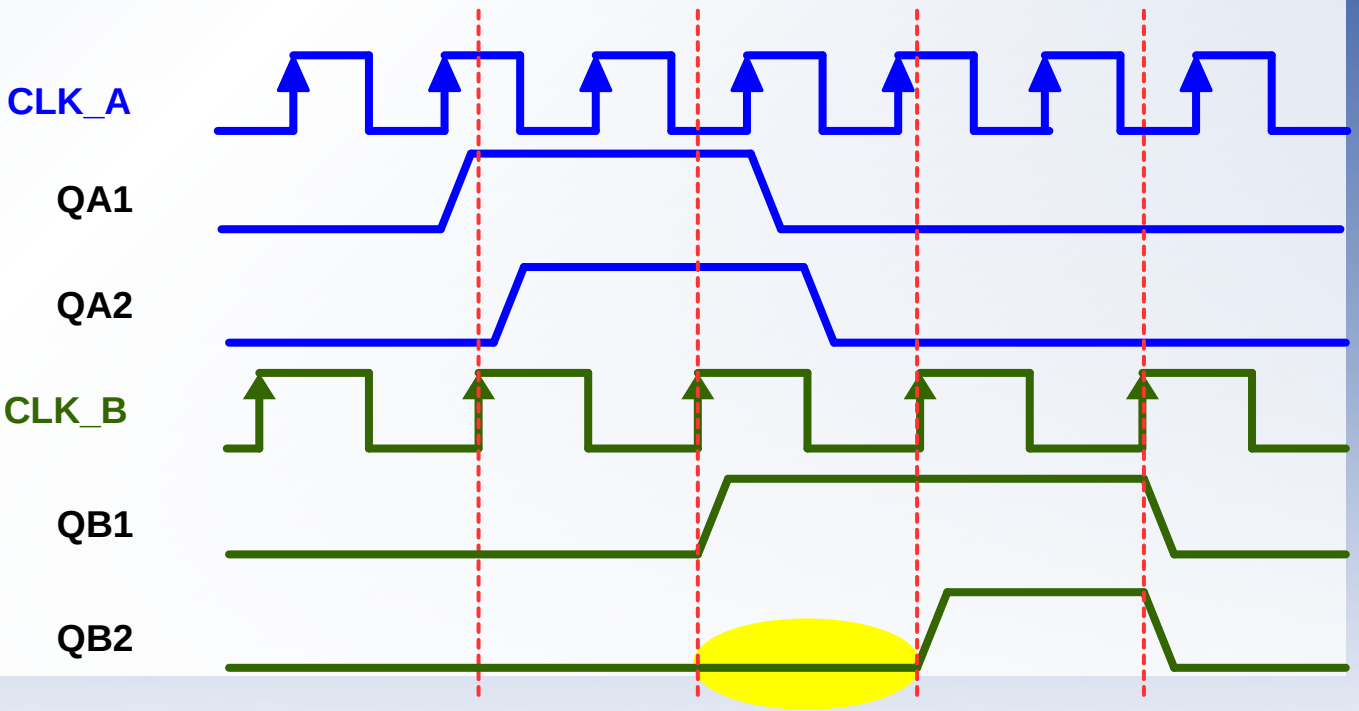




[Structural] Multi-Fanout Issue

- Latch at different times

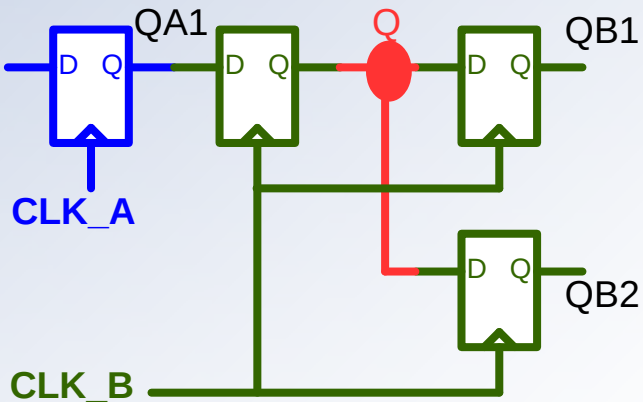


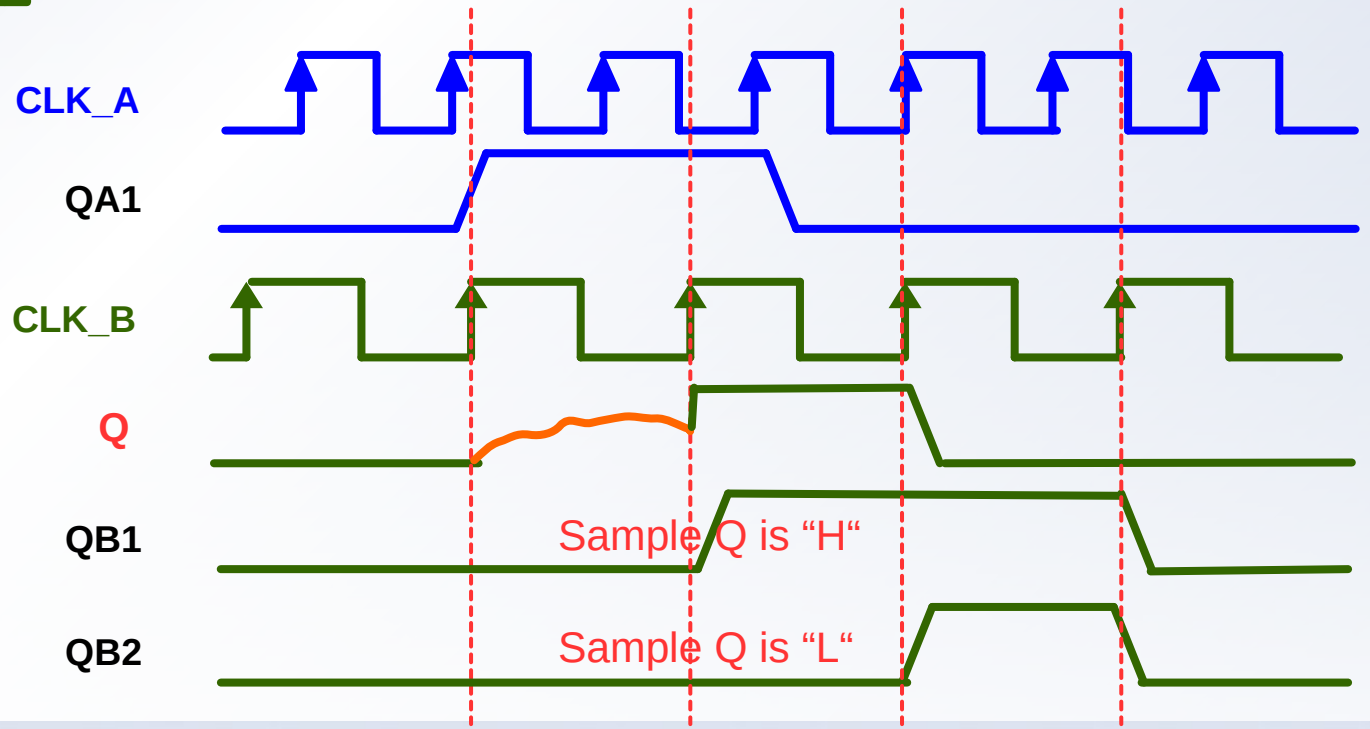


[Structural] Multi-Fanout Issue

NTUST

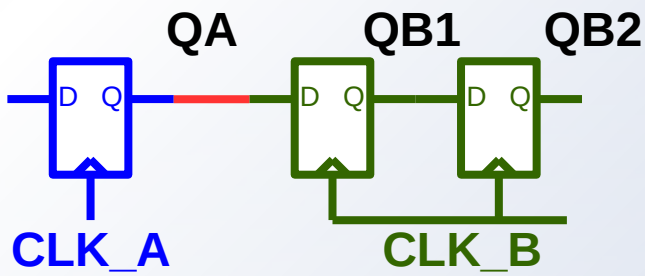
- Latch at different times

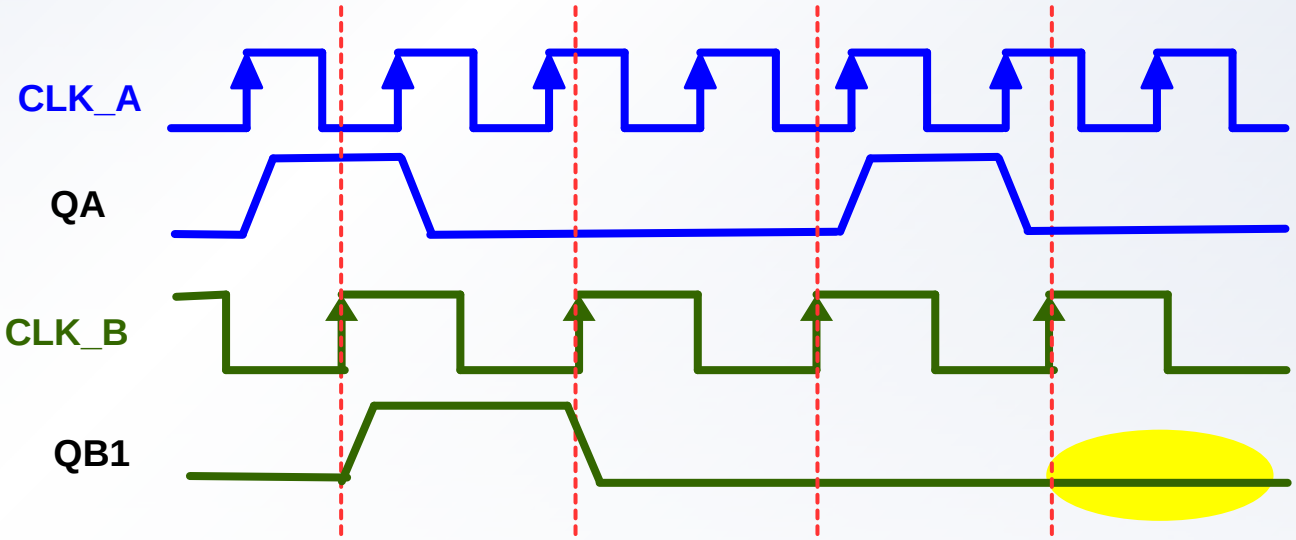




[Functional] Data Stability Issue

- Data from the faster clock domain must be held long enough for the destination clock to latch it
  - CLK_A is faster than CLK_B,
    QA only one cycle, this will cause problem !
  - This design is flawed and unacceptable!

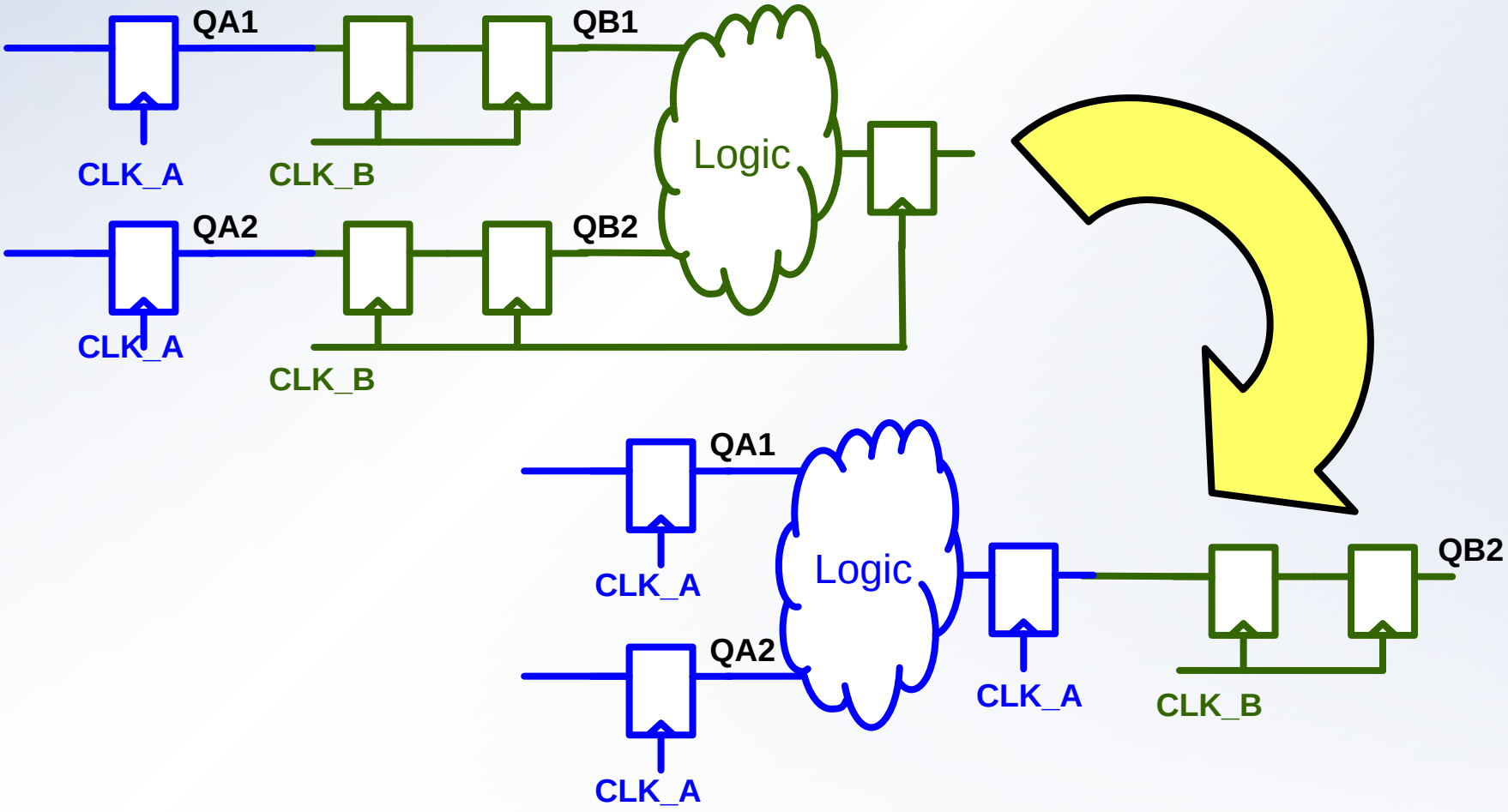




One Cycle QA signal, but CLK_B loss sample it
Cause to data transfer loss at destination.

[Functional] Data Stability Issue

- Problem with Re-convergence issue
- Problem due to different delay between multiple path



How to do Single/Multi-bit sync?

NTUST

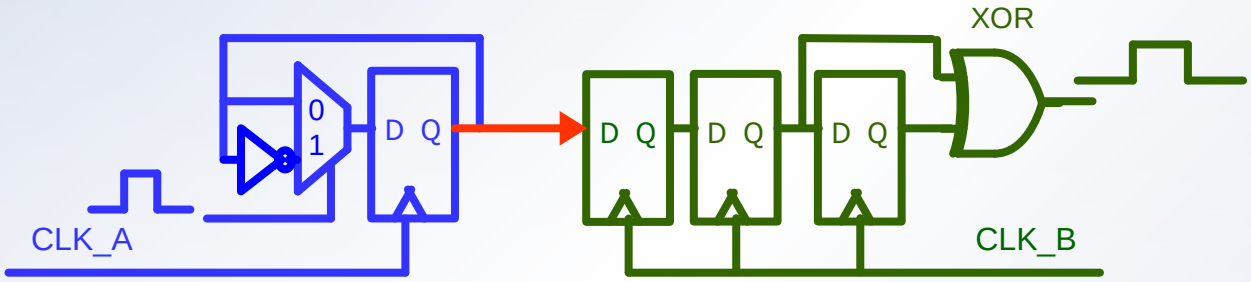
- HOW TO DO?
  - Single-bit Synchronization
  - Multi-bit Synchronization

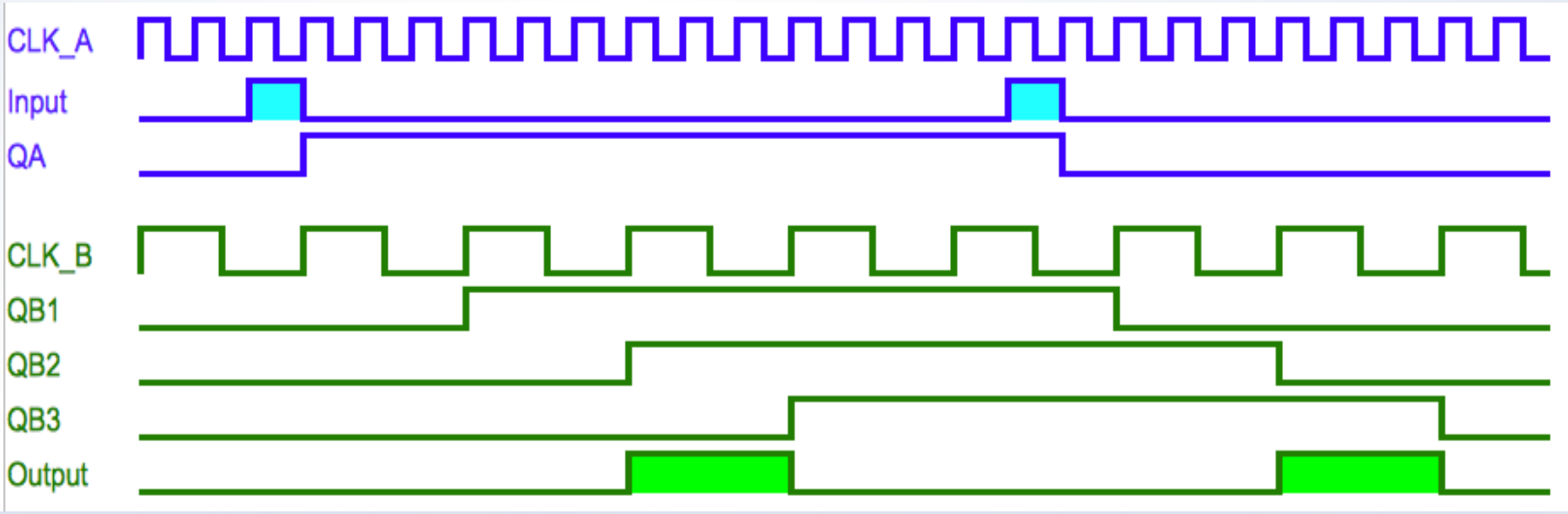
Single Bit Synchronous

- One bit
  - Signal is Status Or Event?
  - How to transfer Status to another clock domain
    - Two flip-flop synchronizer to avoid metastable
    - Ex: Busy, Done, Finish, Ready...
  - How to transfer Event to another clock domain
    - GenShot circuit
    - Level change trigger
      - No need timing check for STA
      - Only need destination clock to double sample, good for STA avoid long source clock tree

[Single Bit Synchronous] GenShot

- Trigger signal transfer to cross clock domain
- Good for control signal synchronizer!!!

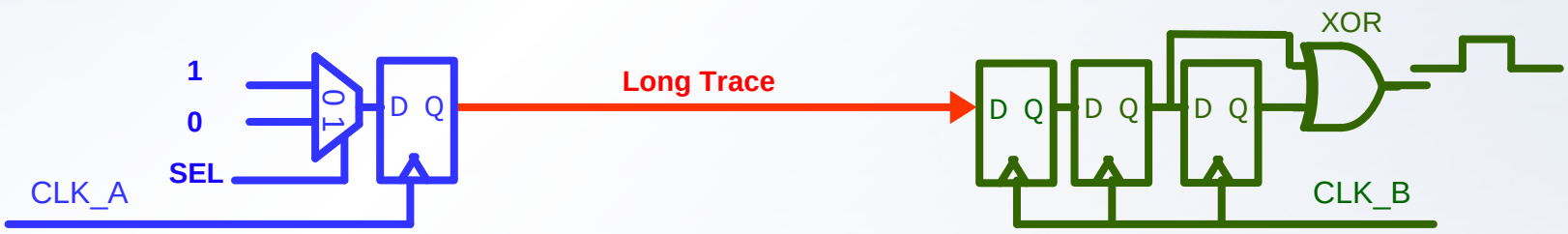




[Single Bit Synchronous] Level Change Trigger

- As same as GenShot behavior
  - Control level change from **source** side
  - Re-generate pulse by **destination** side
  - Good for long route without STA check
  - Good for clock tree

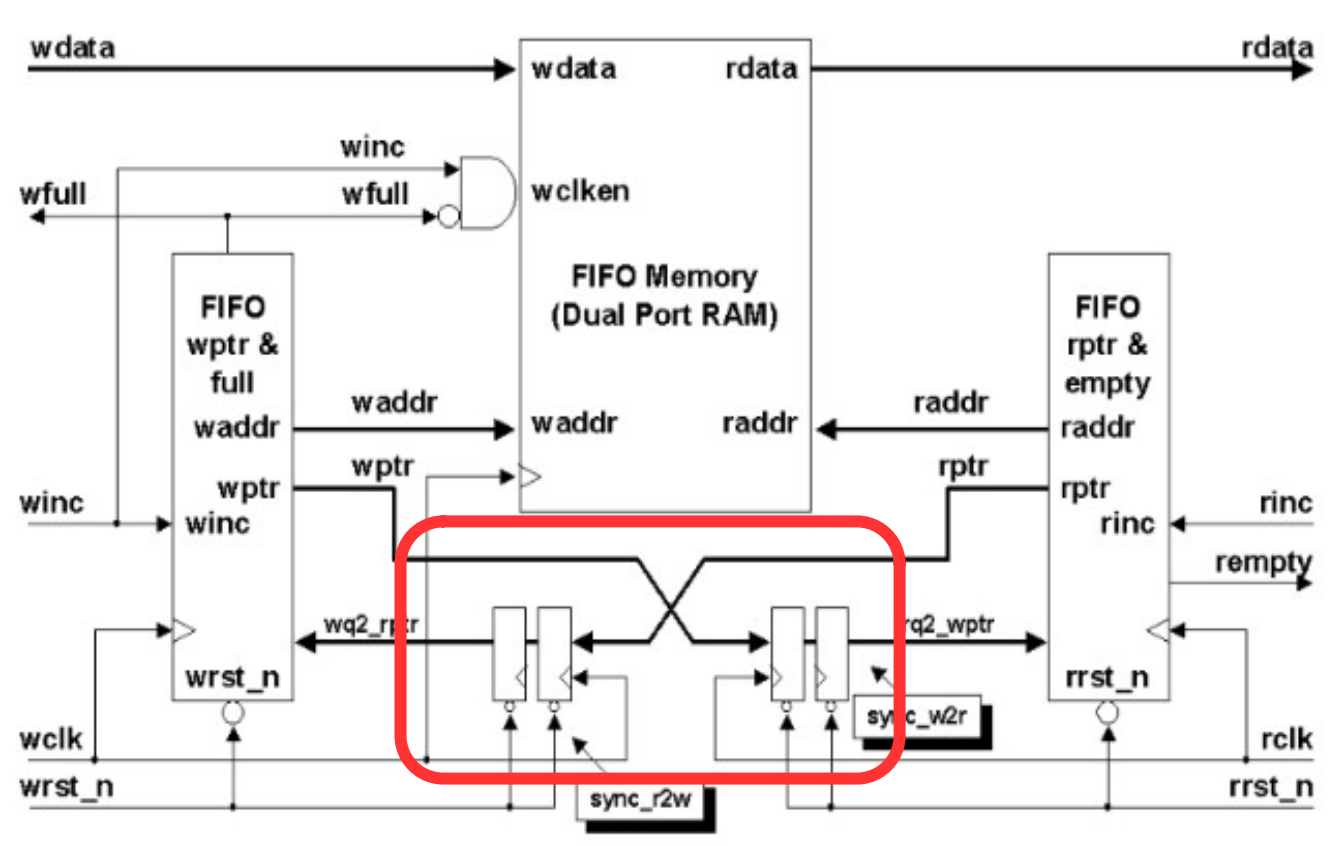
    - Destination clock to double sample only, avoid long source clock tree



[Multi-Bit Synchronous] Async FIFO

Async-FIFO with full/empty status

- How to transfer R/Wptr to other side
- Gray Code!!!

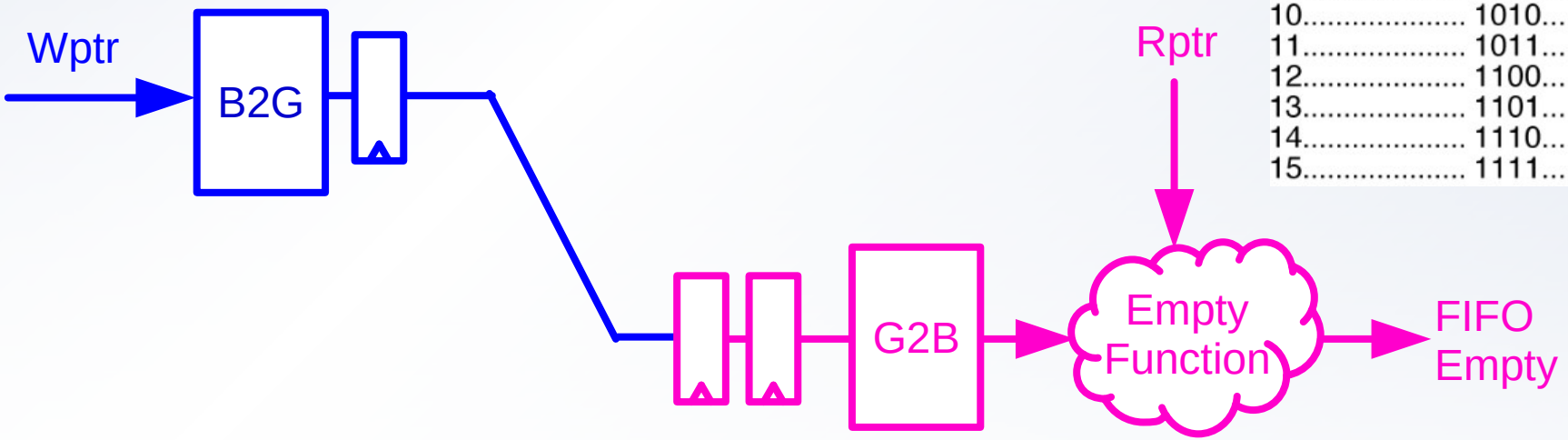


Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Full/Empty Compare Function

Gray code 的編碼特性 , 優點在於每次 increase 都只會變化 1 bit, 有利於處理 Asynchronous design, 避免 metastable.

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000



Safely Full & Empty

- Slow Clock sample Fast Pointer, how to handle Full/Empty?

- Example: wclk (fast), rclk (slow)
  - Full function analyze
    - Fast clock sample slow Rptr:

      Sync 後的 Rptr 保有即時的變化 (Rptr=0,1,2...)

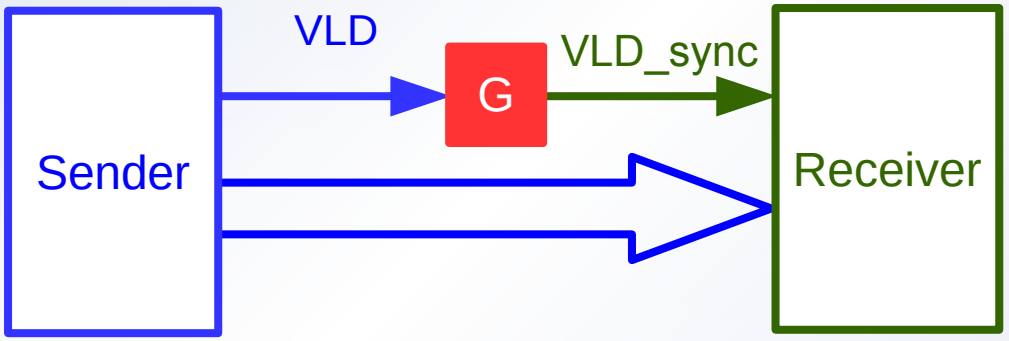
  - Empty function analyze
    - Slow clock sample fast Wptr:

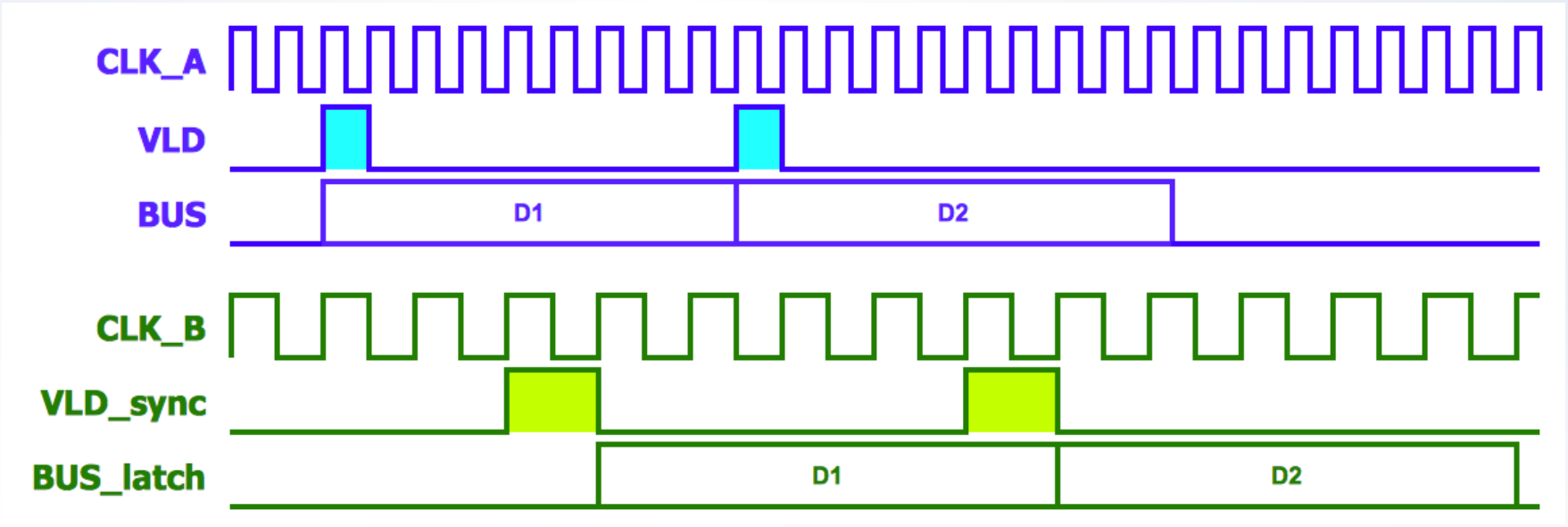
      比較慢反映出 Write side 已經有 Entry 被寫入，延後讀取

[Multi-Bit Synchronous] Hand Shake Method

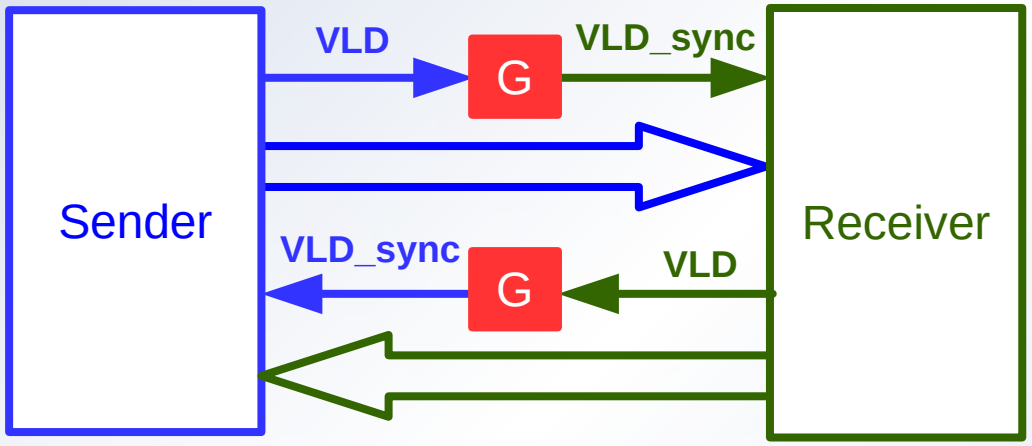
NTUST

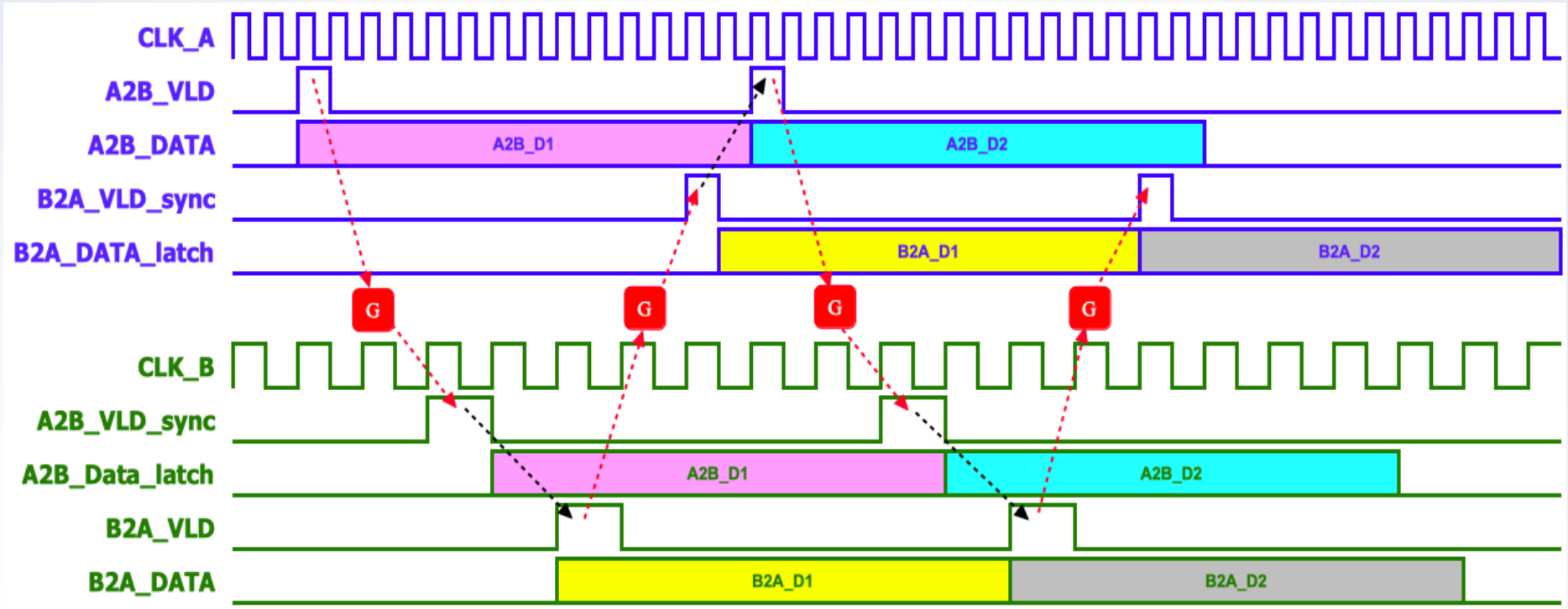
- Only synchronous Control signal.
- Data BUS **no need** synchronizer





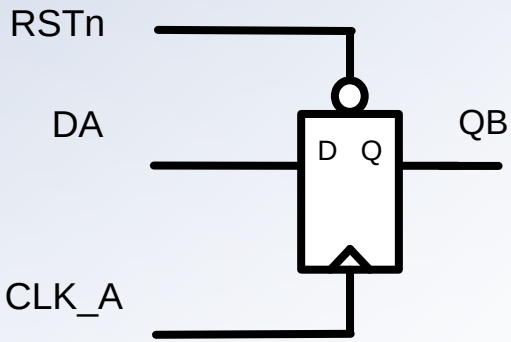
[Multi-Bit Synchronous] Hand Shake Method

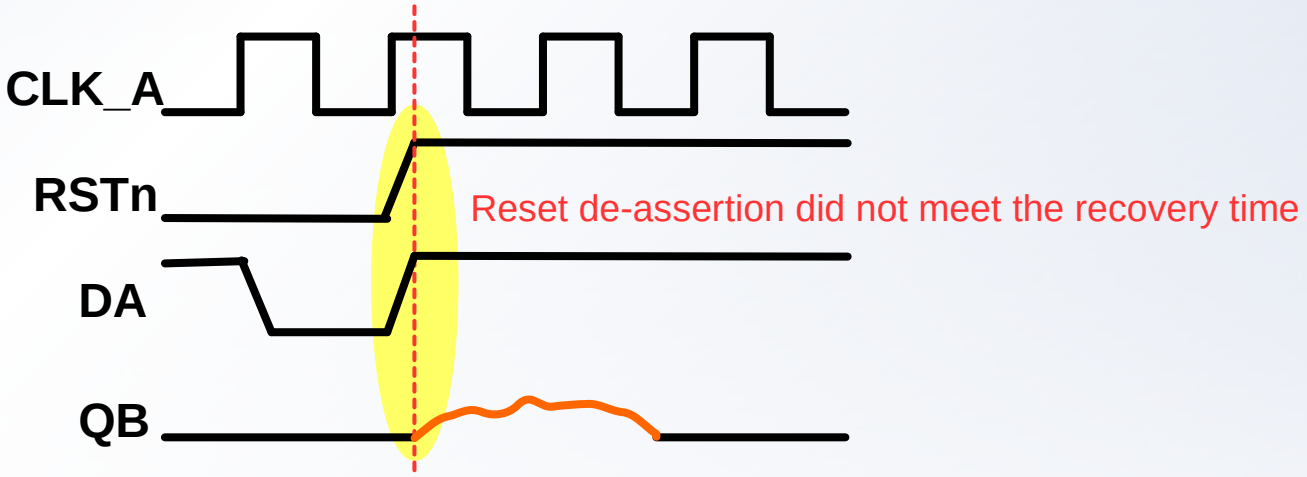




How About Async RST?

- Metastability due to asynchronous set/reset de-assertion
  - Potential metastability occurs due to time violation of RSTn





Clocked signal QB is metastable due to async reset recovery

Recovery/Removal Time

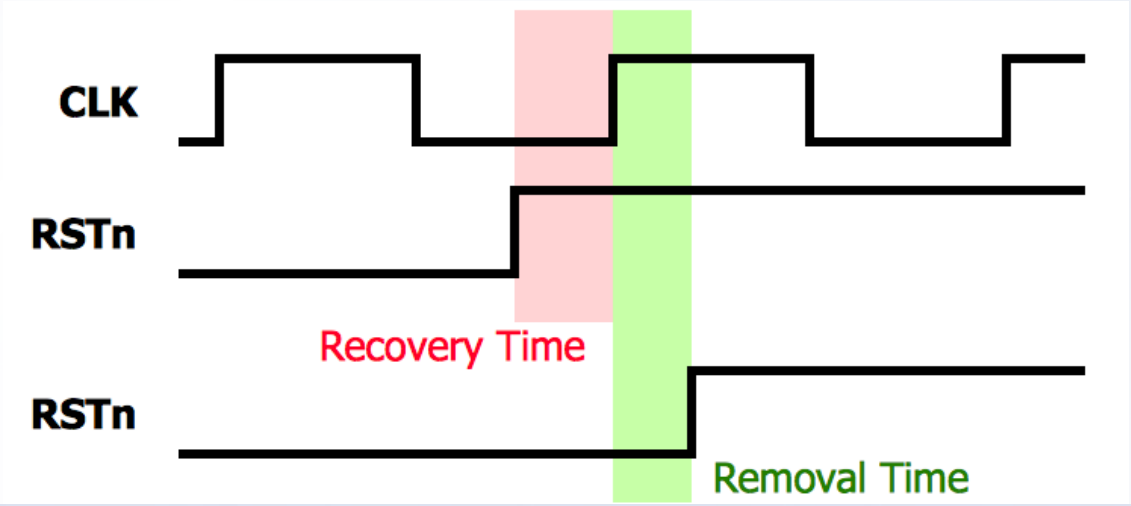
NTUST

Recovery Time

- Inactive async 轉態在 clock edge 前發生，需要提前一個最小時間，指 flip-flop output 要恢復成由 clock latch input clock. 所以 clock edge 要成為真正 active ， active async 需要在 clock edge 發生前一段時間就要轉成 inactive async. data output 才會同步由 clock latch data input. 這段時間就是 recovery time. Inactive async state 的準備時間 .

Removal Time

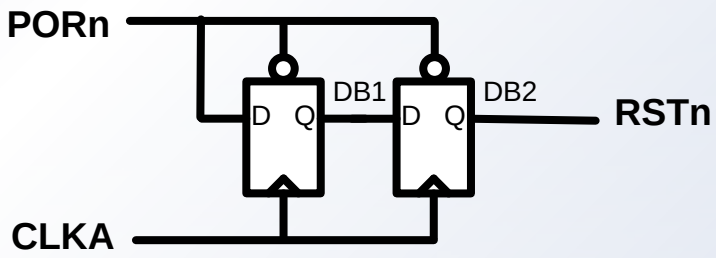
- Inactive async 轉態在 clock edge 後發生，需要 clock edge 一個最小時間之後才發生，是說在 clock edge 在處於 active async 時，此時 clock edge 是不會去 latch input data, async signal 需要保持一段 stable 區間，才允許 active async 轉成 inactive async. 簡單說 要移除 async 的 override 必須要在 clock edge 發生之後一段時間才可以轉態，才不會在這個 clock edge 發生不確定的 data 狀態。 Active async state 轉態前維持時間 .



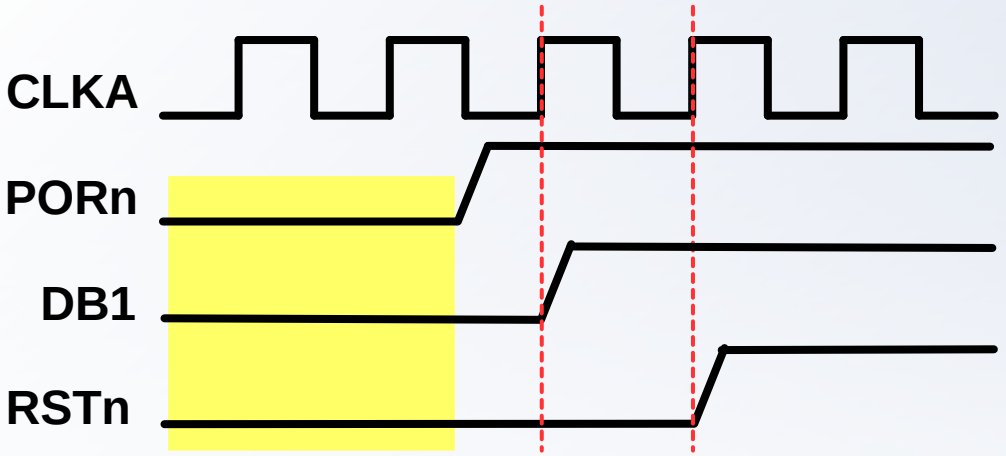
How to Handle Asynchronous Reset

NTUST

- Hint:
  - Asynchronous assertion
  - Synchronous de-assertion



DB2 goes to Async reset of all FFs on CLK domain

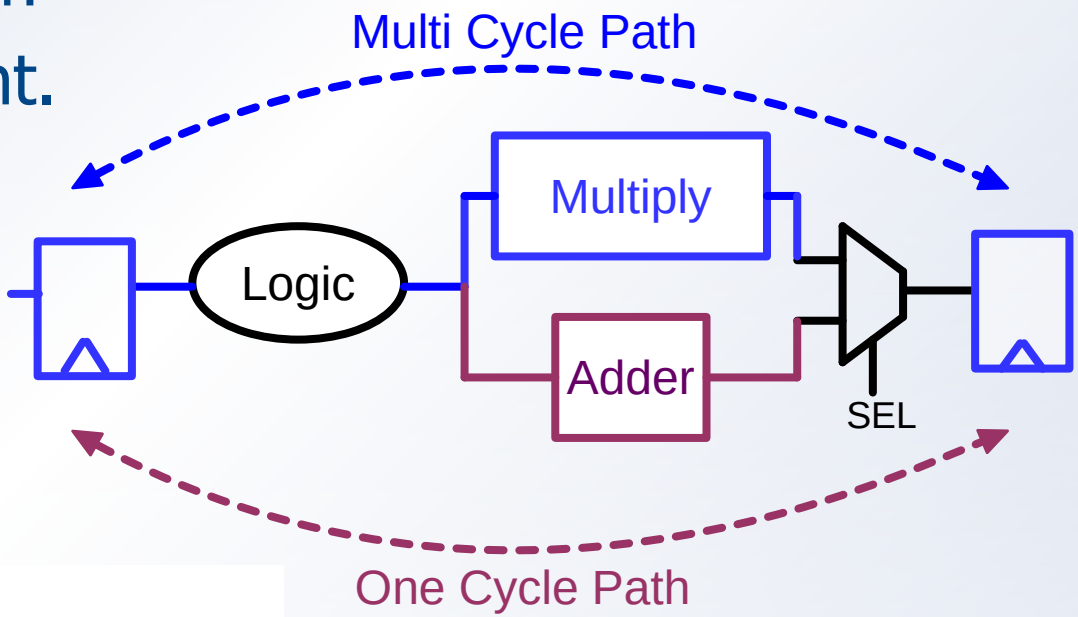


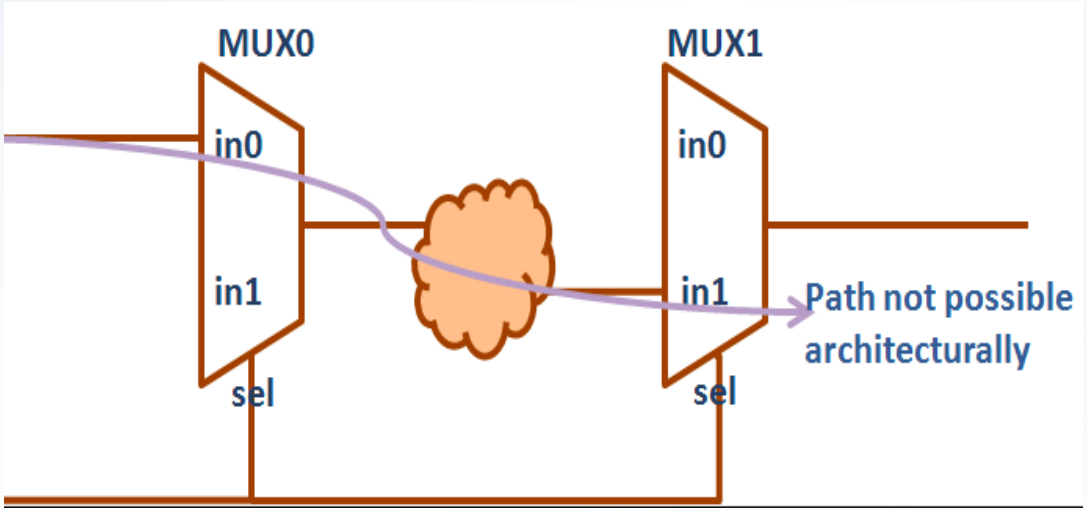
High Speed Design

- Multi-Cycle/False Path
- Pipe Line Design
- Critical Path Fine Tune
- Gate Level Optimal
- Clock Tree Adjustment
- Physical Long Route
- Share High Fan Out Loading

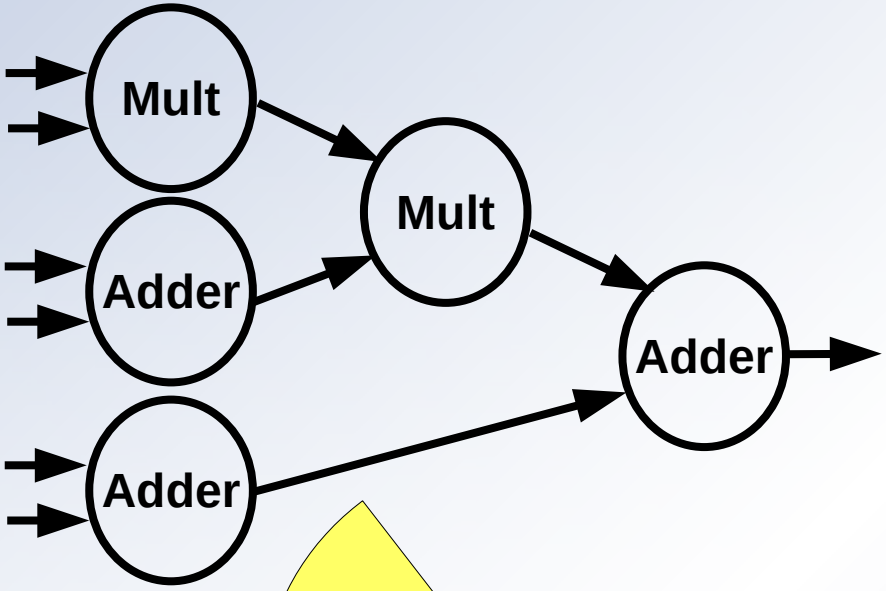
Multi-Cycle/False Path

- Muti-Cycle or False Path?
- Synthesis/STA constraint.

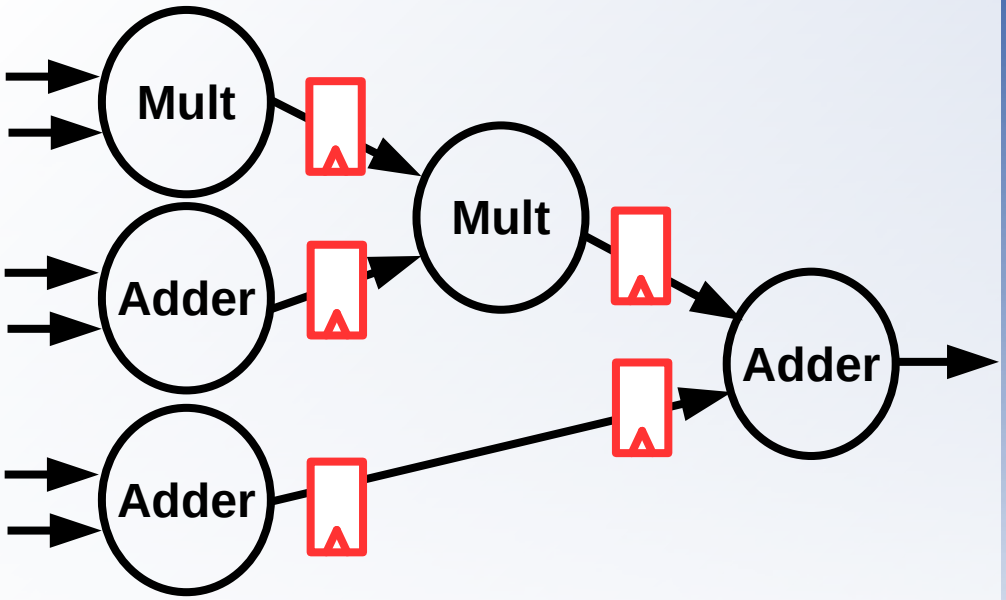




Pipe Line Design

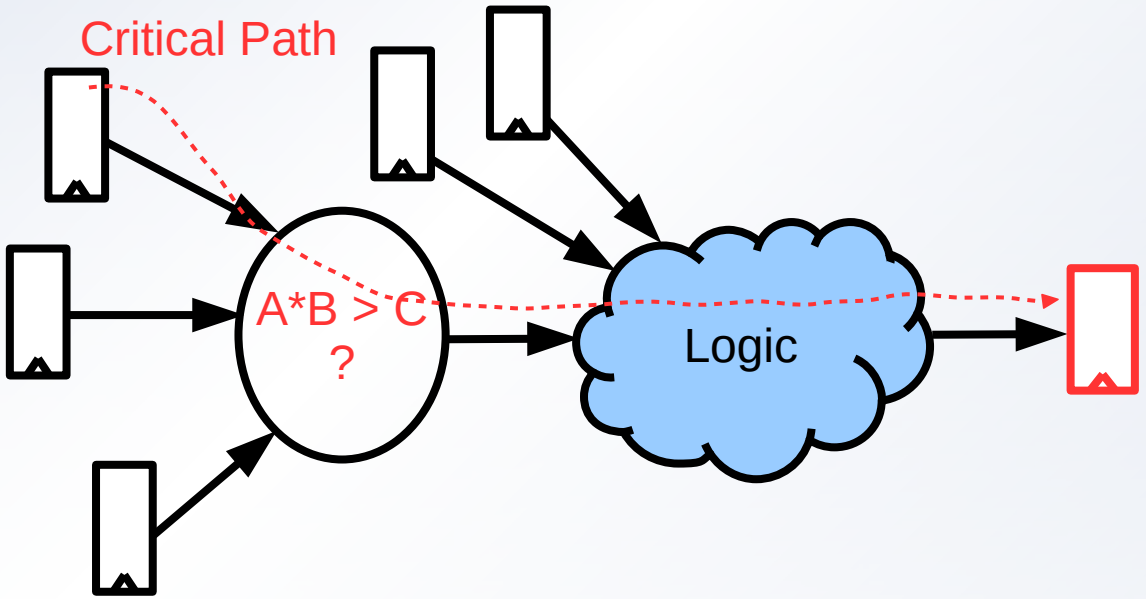


- Pros:
  - Good for timing
  - Good for throughput
- Cons:
  - Bad for latency delay



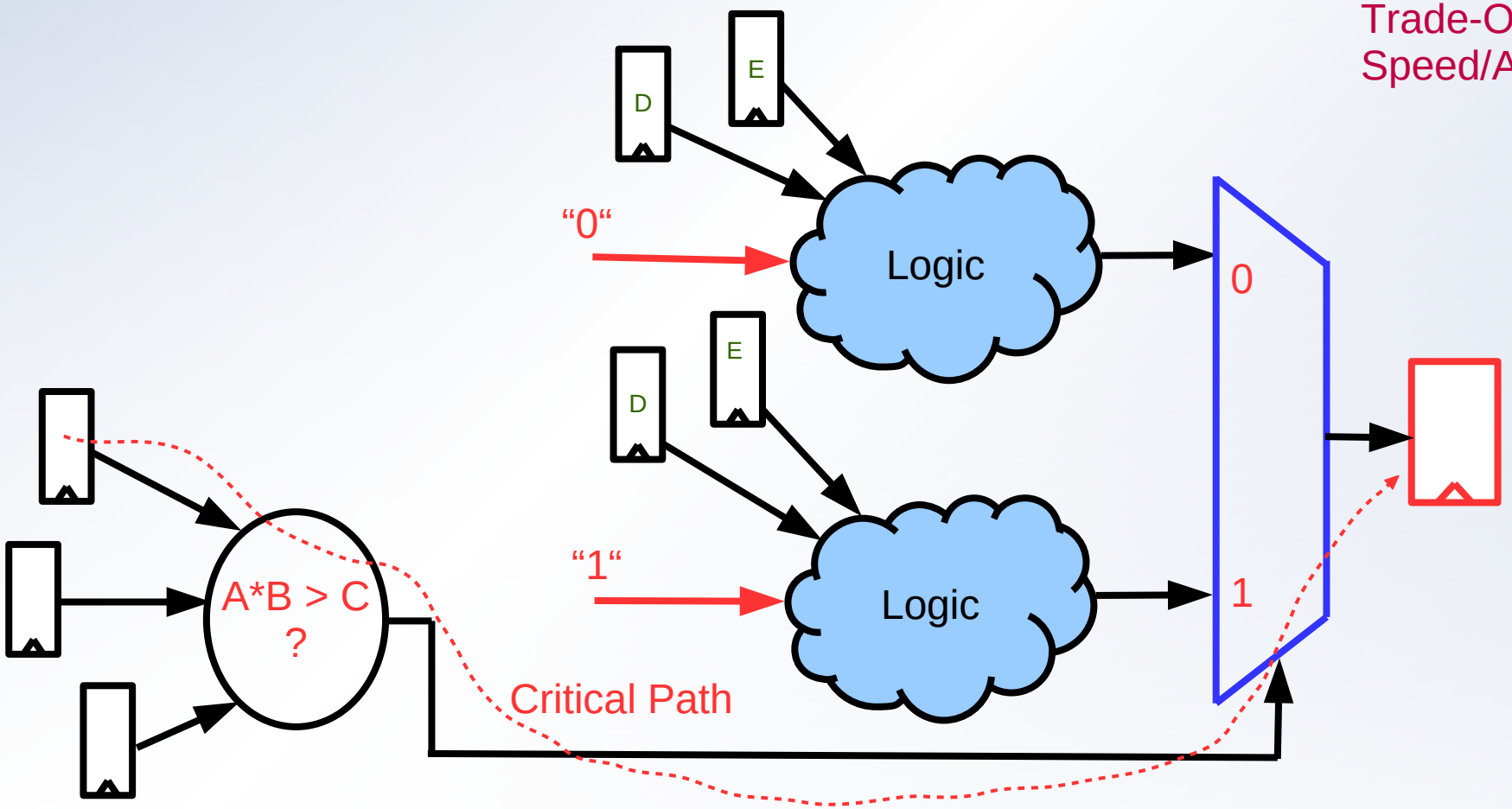
Critical Path Optimize

- Find out the critical path.



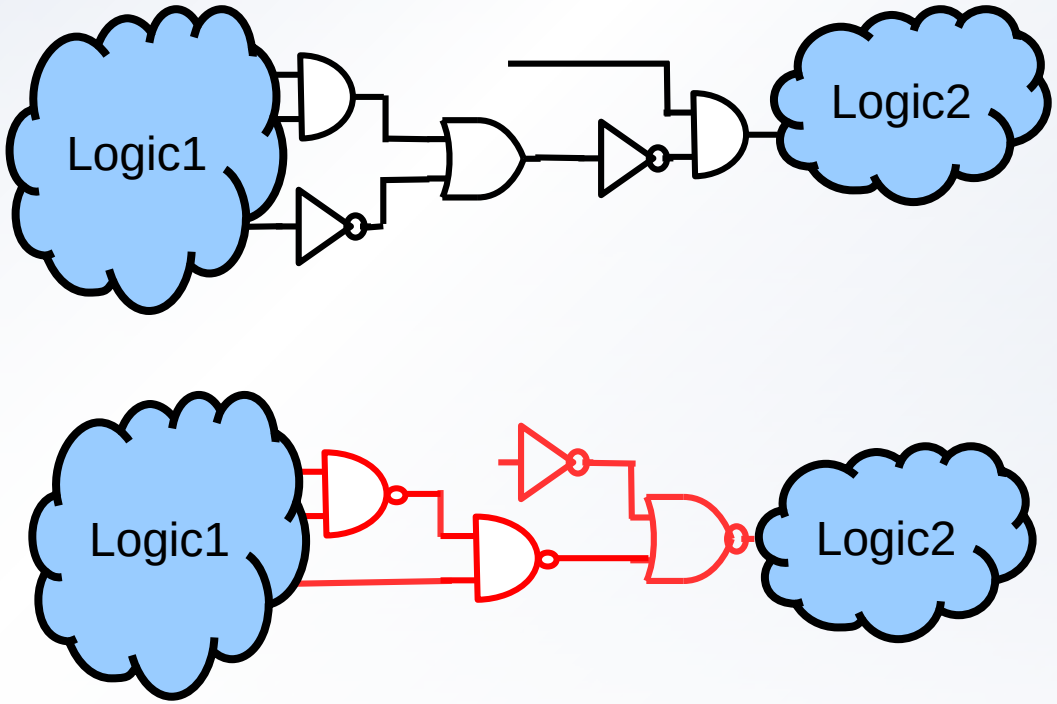
Critical Path Optimize

- Double logic and select final result by latest signal

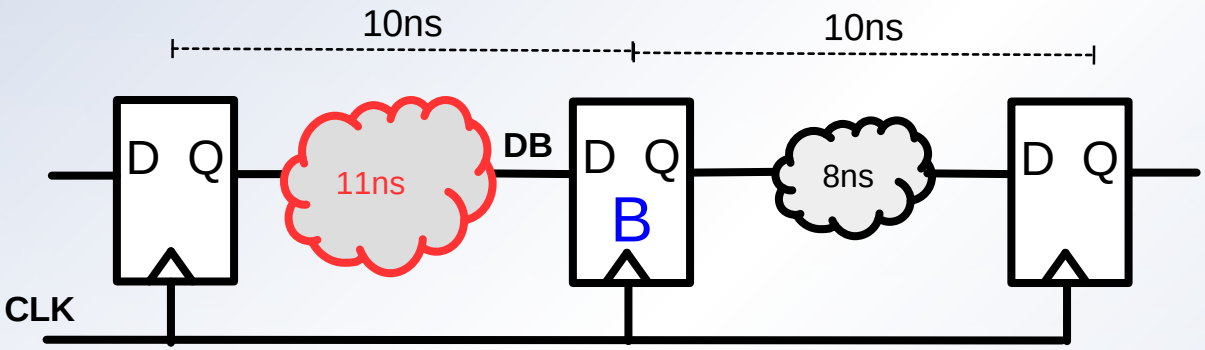


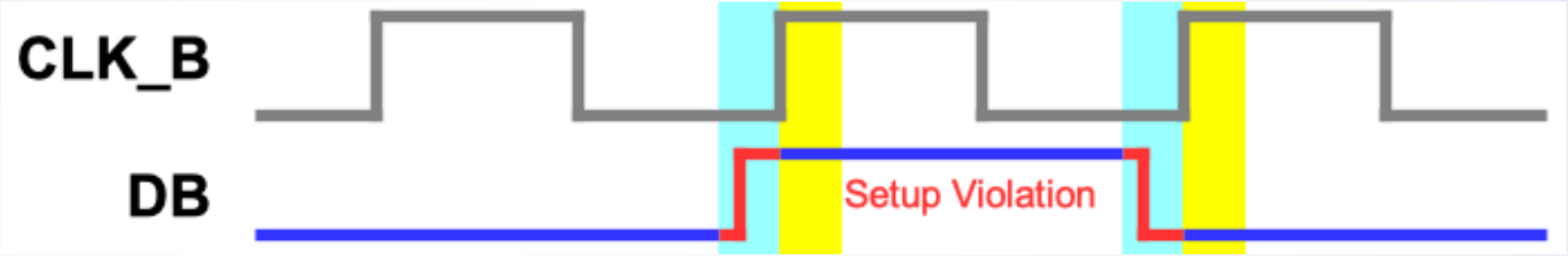
Gate Level Optimize

- ECO Stage: (Engineering Change Order)
  - NAND/NOR logic are good for timing
- Gate delay optimal.



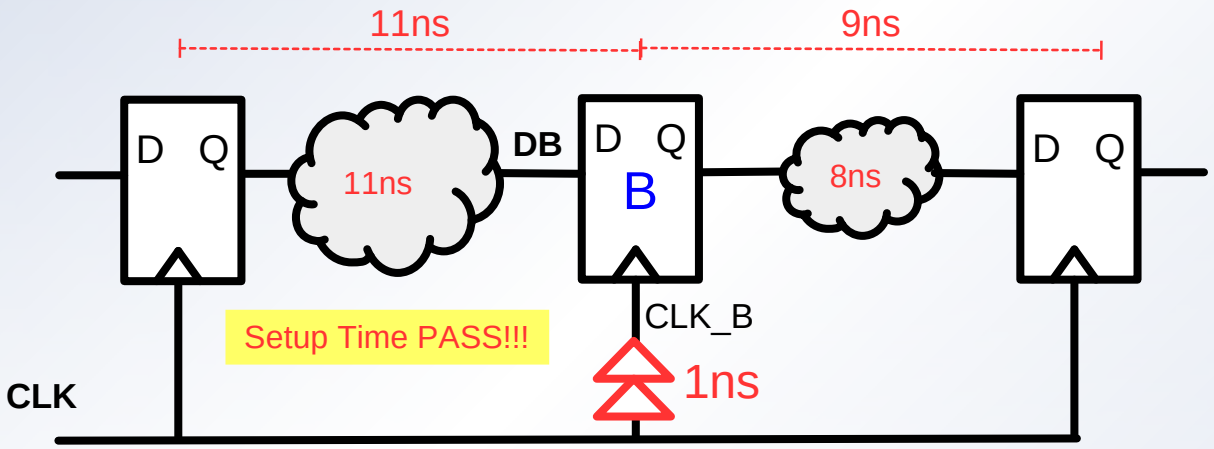
Clock Tree Adjustment

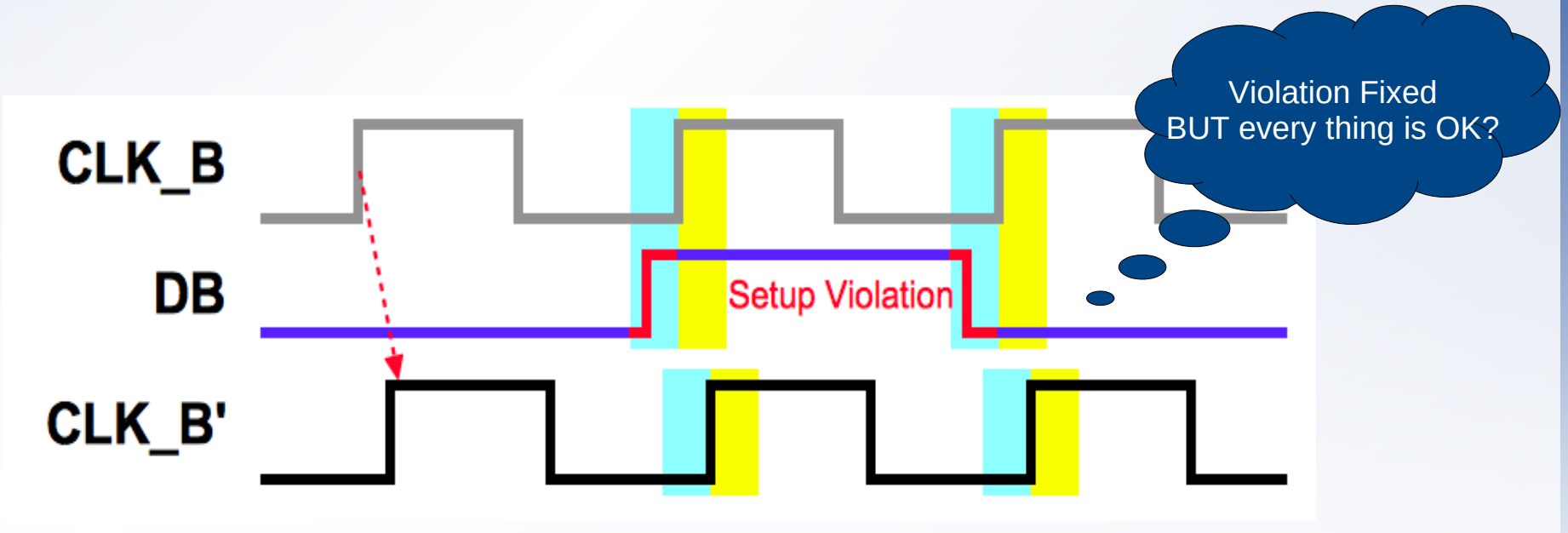




Clock Tree Adjustment

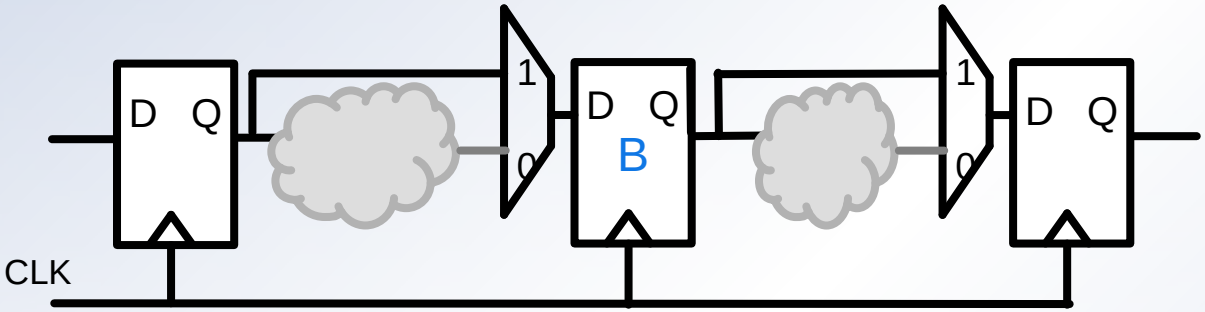
- Insert clock buffer for clock tree



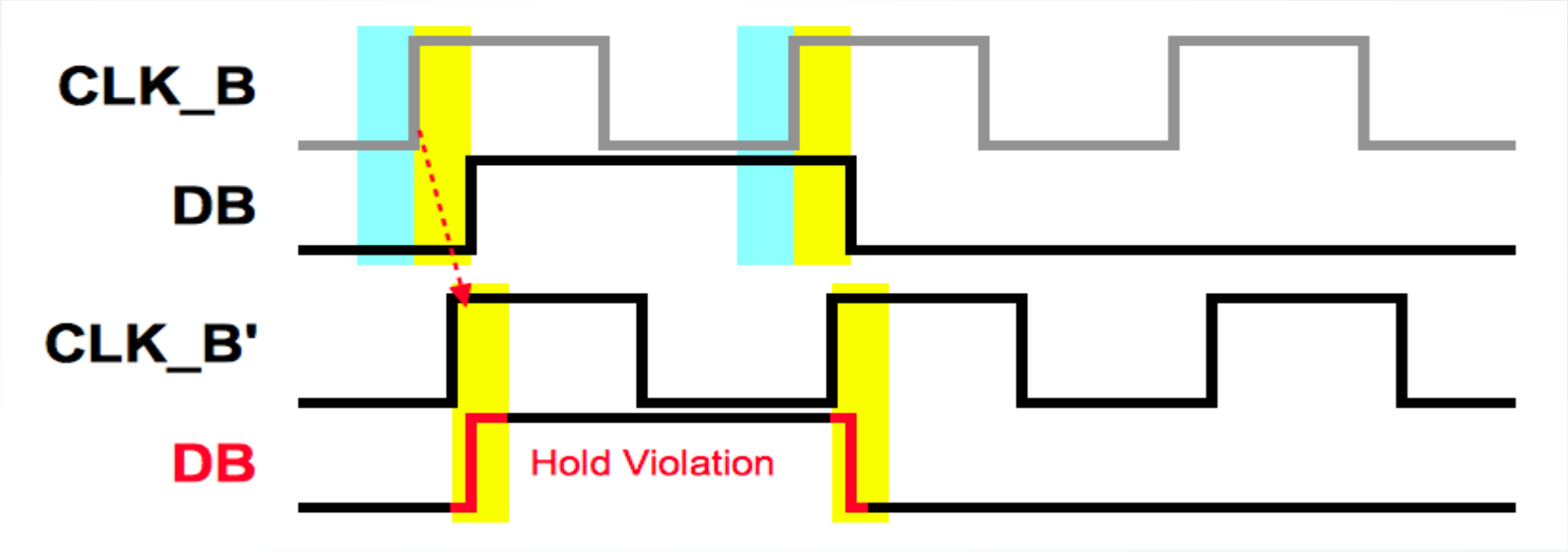


Clock Tree Adjustment

- How about SCAN mode?

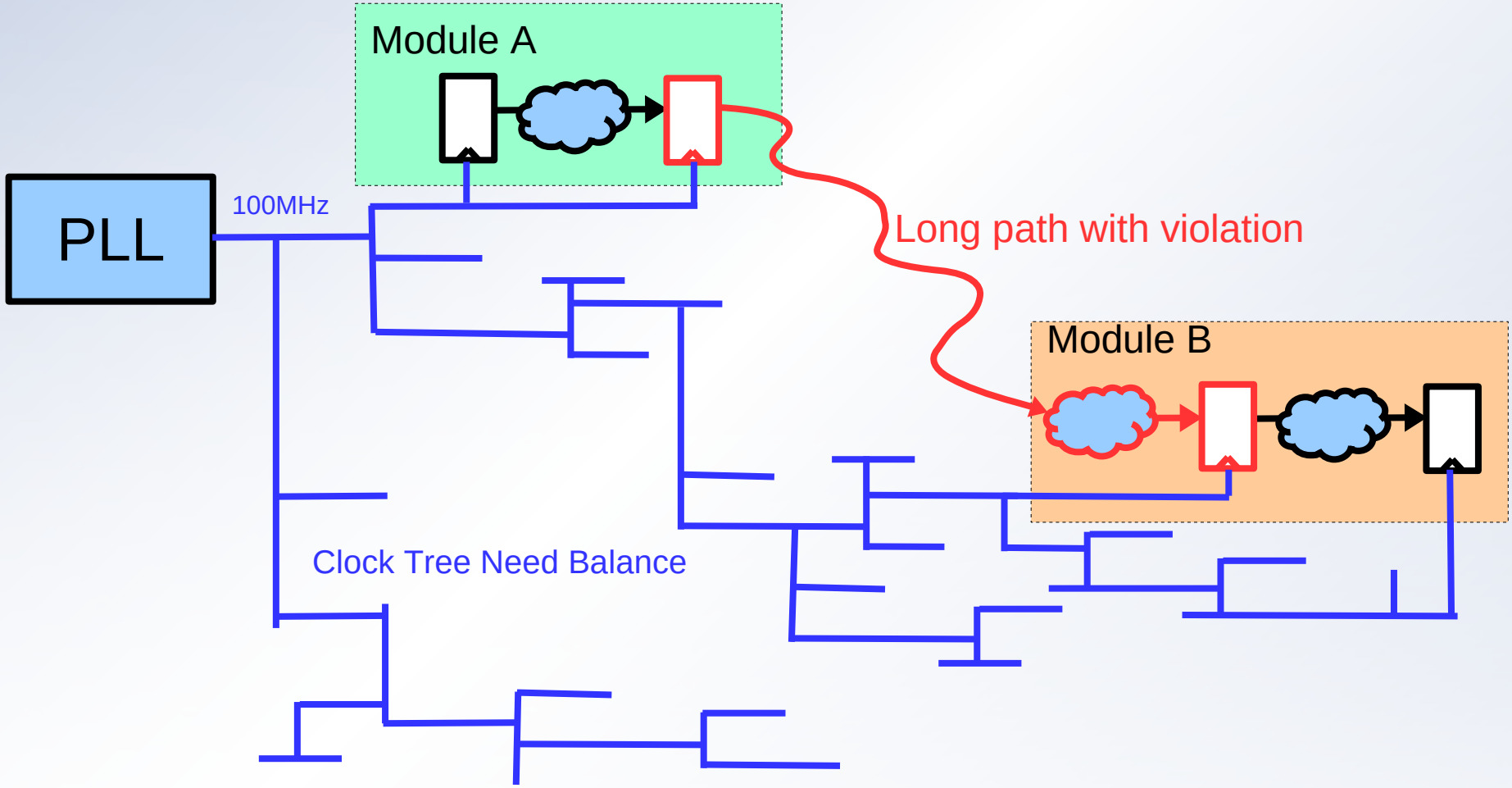


Impact Hold Time

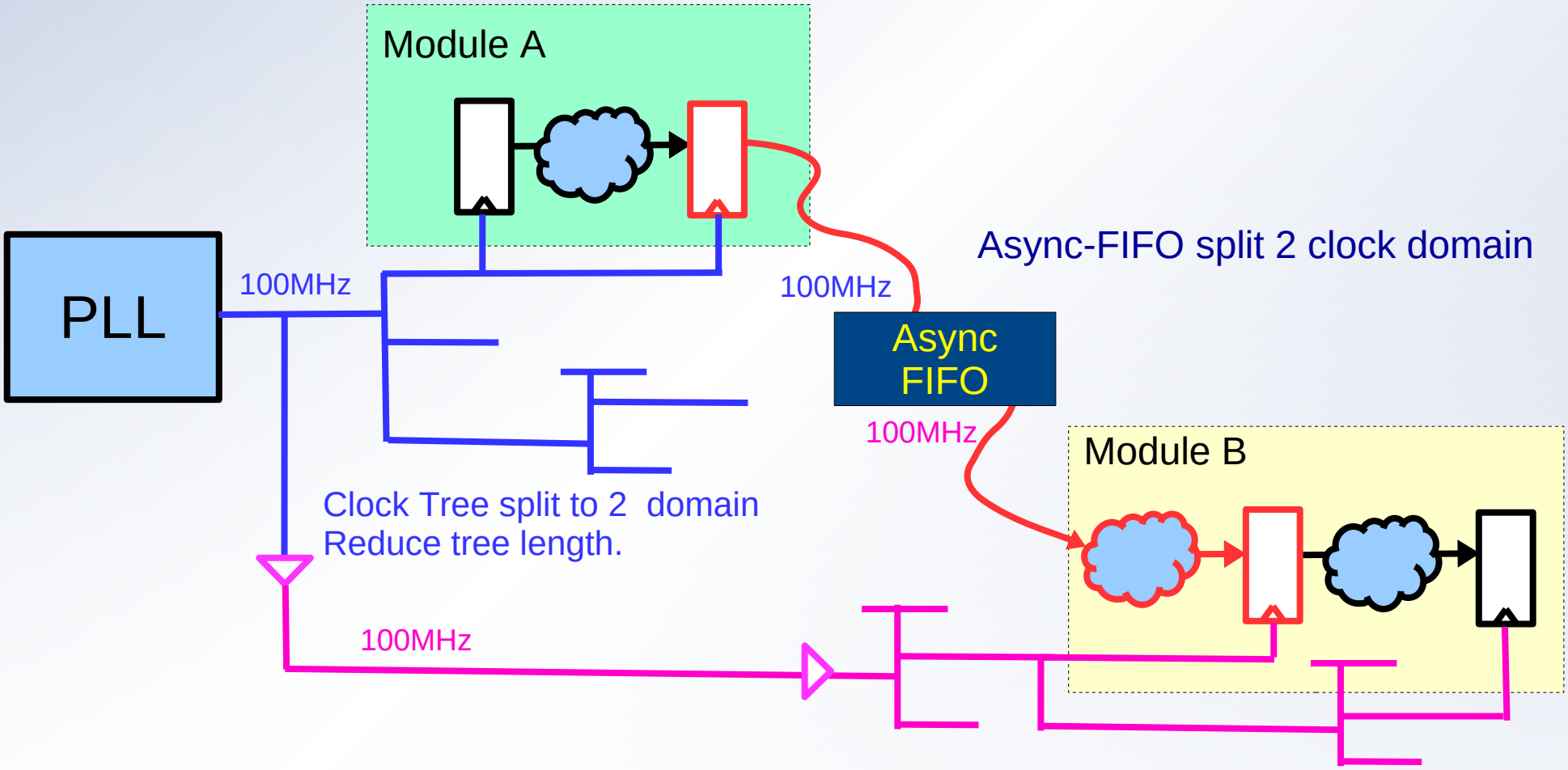


解決方法？

Physical Long Route

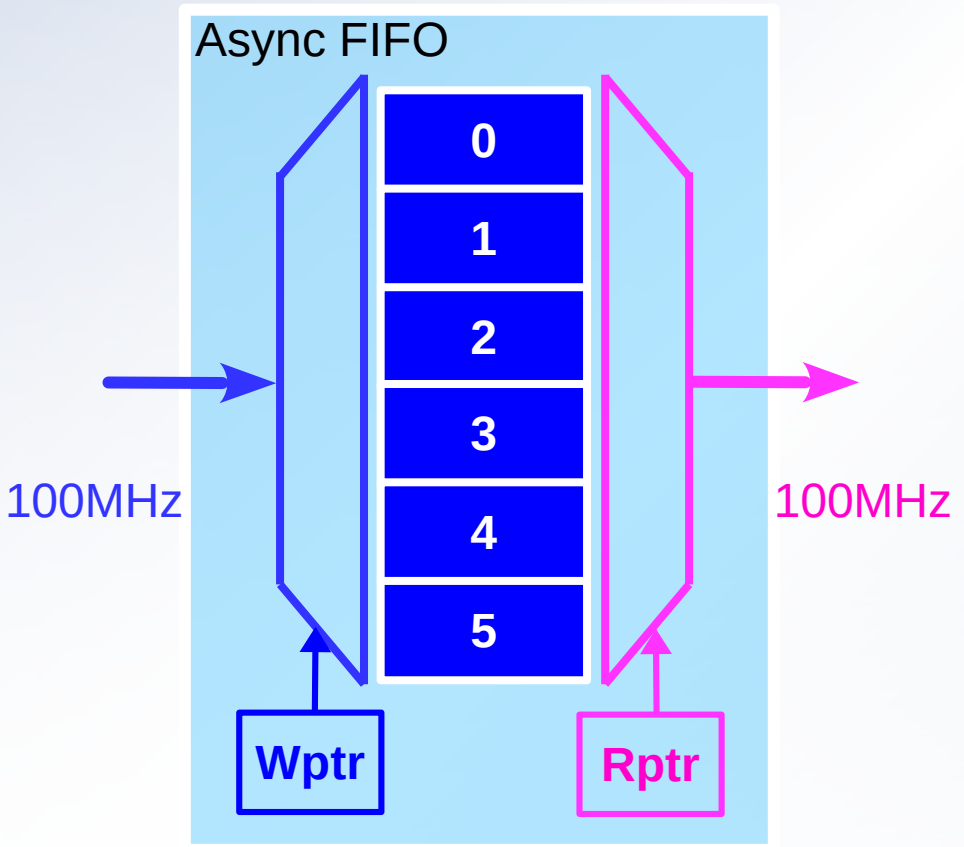


Physical Long Route



Physical Long Route

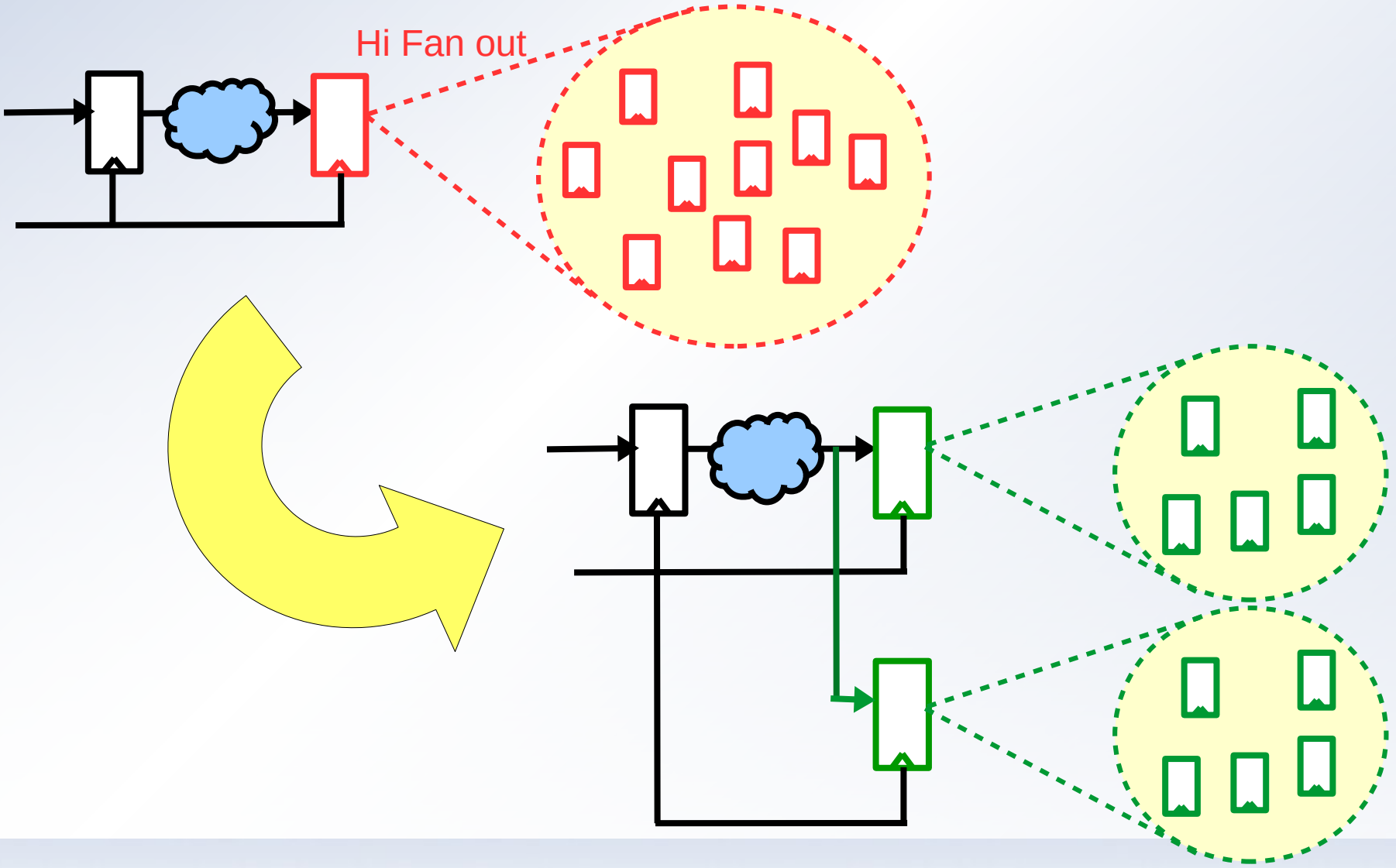
- Consider some item
  - No Full/Empty signal
  - Two clock need from same PLL
  - Be careful Wptr/Rptr initial value assignment.



Share High FanOut Loading

NTUST

- Duplicate design, share loading



Low Power Design

NTUST

Low Power Design

- Why RTL Level Improvement
- Minimizing data transition on bus
- Resource Sharing
- Register Retiming
- State Machine Encoding
- Memory Decision

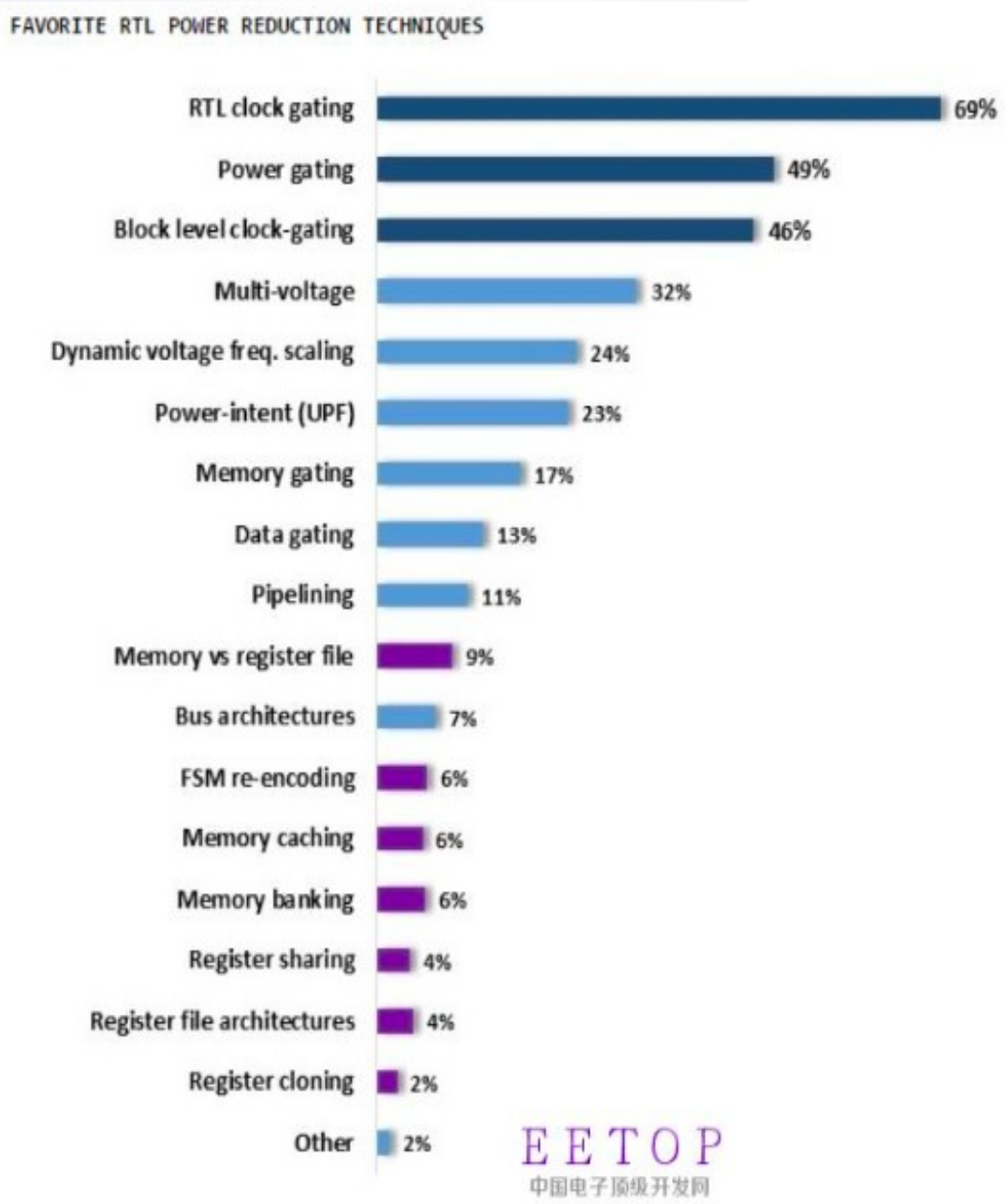
NTUST

Why RTL Level Improve

NTUST

- Sequential logic
  - Reduce toggle rate
- Combinational logic
  - Logic sharing
- Clock network
  - Block level gating
  - RTL coding style(ICG)
  - Short tree length
- Memory
  - Area/power balance
  - Bit width/Depth selection
  - Address coding

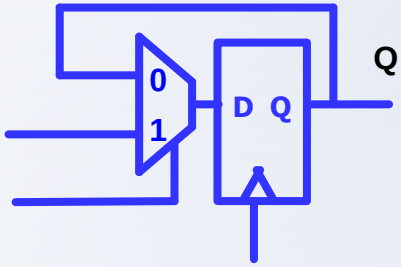
$P=CFV^2$



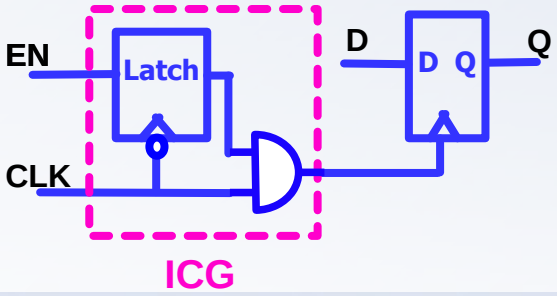
Minimizing Data Transition

- The data on the bus keeps on transitioning from one value to another value. This may not affect the design functionally as there may be some handshaking to indicates that data is valid. **But the invalid transitions waste more power.**
- Coding style for **ICG** synthesis
  - If, else if, else if... avoid else

```
always @(posedge clk or negedge rstz)  begin
    if(!rstz)          DataBUS <= 32'd0;
    else if(Data_vld) DataBUS <= Data_IN;
    else               DataBUS <= 32'd0;
end
```




```
always @(posedge clk or negedge rstz)  begin
    if(!rstz)          DataBUS <= 32'd0;
    else if(Data_vld) DataBUS <= Data_IN;
end
```



Resource Sharing

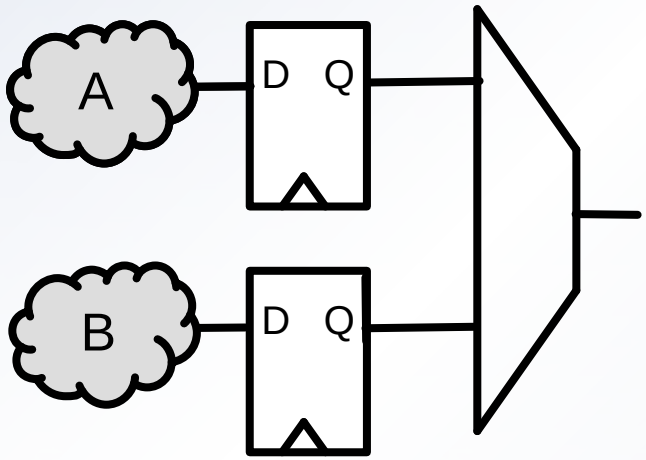
The RTL coding should be carried out in a manner that there are no unwanted or redundant logic elements. Any logic element will contribute to power consumption as it has a capacitance attached to it and transitioning of data through that logic will lead to power dissipation.

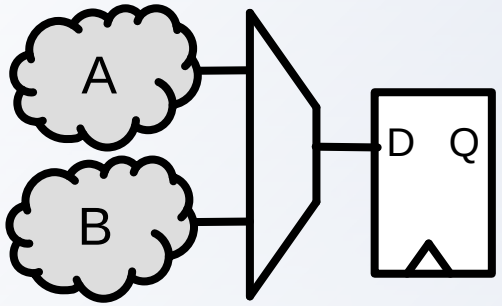
```
assign result1 = sel ? (in1*in2) : 4'd0;
assign result2 = sel ? 4'd0 : (in3*in4);
```

```
assign X = sel ? In1 : in3;
assign Y = sel ? In2 : in4;
assign result_mult = X * Y;
assign result1 = sel ? result_mult : 4'd0;
assign result2 = sel ? 4'd0 : result_mult;
```

Register Retiming

Register timing is a concept mostly used in improving timing by reordering the combinational and sequential logic in a given data path. However in certain cases, there is a saving of logic and thus can help improve upon power consumption. Of course, this is possible only if the design can support the additional timing overhead.





State Machine Encoding

It is a well known fact that one-hot and Gray encoding consume lesser power as compared to binary encoding. This is because one-hot and gray encodings have few bit change while going from one state to another.

- One-Hot
- Gray Code

Memory Decision

- SRAM type
  - 1 Port (CLK x1, Addr x1, D x1, Q x1)
  - 2 Port (CLK x1, Addr x2, D x1, Q x1)
  - Dual Port (CLK x2, Addr x2, D x2, Q x2)
  - Register file
- Width/Depth balance selection
  - Ex: sram256x10b VS sram128x20b
    - Bus 大一倍，不意味着每次讀寫 power 大一倍，
      伴隨著讀寫次數減少一半，這樣也能帶來可觀的收益
- Addressing Encode
  - Gray coding for addressing memories

[Recap] Asynchronous Design

- What is Metastable
- How to Handle Asynchronous
- Single-Bit Synchronous
- Multi-Bit Synchronous
- Asynchronous Reset

NTUST

[Recap]
High Speed Design

- Multi-Cycle/False Path
- Pipe Line Design
- Critical Path Fine Tune
- Gate Level Optimal
- Clock Tree Adjustment
- Physical Long Route
- Share High Fan Out Loading

[Recap]
Low Power Design

- Why RTL Level Improvement
- Minimizing data transition on bus
- Avoiding unnecessary transition
- Resource Sharing
- Register Retiming
- State Machine Encoding
- Memory Decision

Reference

- Clifford E. Cummings, “Simulation and Synthesis Techniques for Asynchronous FIFO Design,” SNUG 2002 -
- Clifford E. Cummings, “Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs,” SNUG 2001 -
- https://www.design-reuse.com/articles/20775/hdl-design-low-power.html