Clock Domain Crossing

Part 2

Amr Adel Mohammady

/amradelm

Save The Palestinian Children

Till Nov,2023

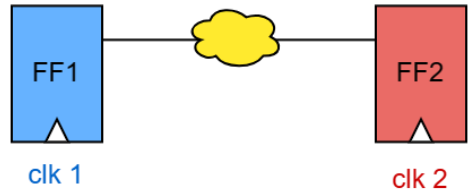
Israel has killed more than 13,000 children in Gaza since October 7 while others are suffering from severe malnutrition and do not “even have the energy to cry”, says the United Nations Children’s Fund (UNICEF).

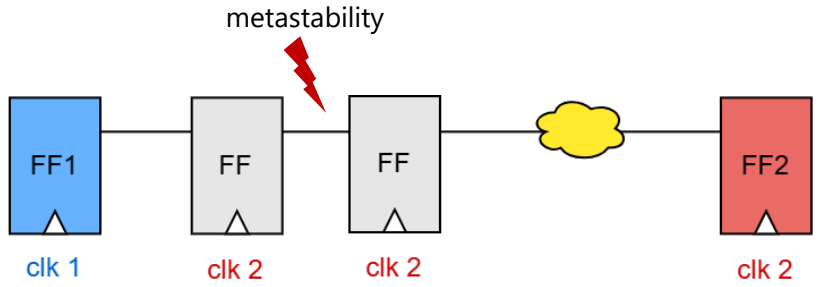
“Thousands more have been injured or we can’t even determine where they are. They may be stuck under rubble … **We haven’t seen that rate of death among children in almost any other conflict in the world**”

-**UNICEF** Executive Director

Introduction

- In the previous part we saw how to limit the effect of metastability between multiple stages of flip-flop that we called CDC synchronizer
- The idea is to isolate the metastable value between the synchronizer FFs and give it enough time until it settles to a known value then give it to the receiving domain
- We also discussed the MTBF metric and how to increase it to ensure a reliable circuit.
- **We want to know what we solved till now. Our CDC concerns were:**
  - Data corruption: Not fixed. The system, till now, has to be fault tolerant and be able to handle corrupted data.
  - Data incoherence: Fixed. The metastable value settles within the synchronizers at 0 or 1 and then propagates to all the blocks of domain 2 with the same settled value
  - Data loss: Not fixed
  - Data duplication: Not fixed
  - Chip burning: Fixed. We limited the metastability propagation between the synchronizers and reduced its occurrence frequency.
- In this part we will handle some of the remaining concerns

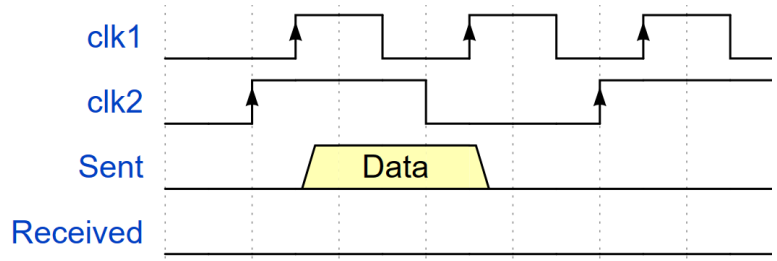




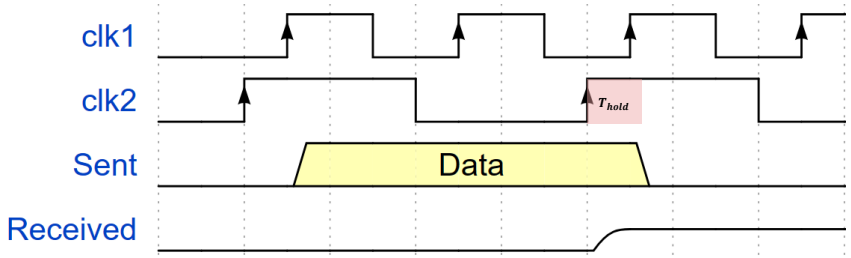
Required Pulse Width

Required Pulse Width

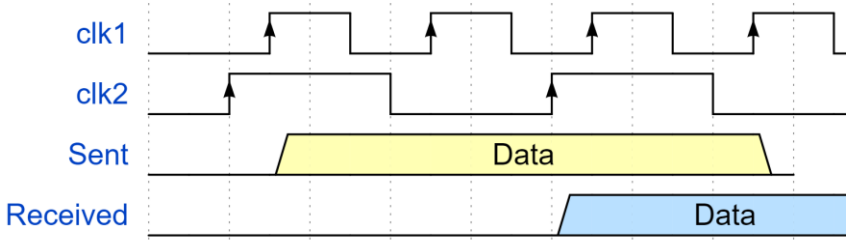
- Adding synchronizers only protected the chip from seeing a metastable value. However, we still get corrupted data because the data can settle at a value different than the intended one.
- To ensure that correct data is received we need to make sure the data pulse is wide enough to be safely captured by a domain 2 clock edge.
- The first example shows a small data width (one clock wide) that completely got missed by any capture edge of clock 2



- The second example shows a wider data pulse that reached a capture edge but not wide enough that it caused a hold violation and therefore metastability



- The required width for correct operation is $\geq clk2\ period + T_{setup} + T_{hold}$. This way we guarantee a capture edge will lie within the data pulse width without any setup or hold violation.
- As a rule of thumb[1], we need to make the data at least $1.5 \times clk2\ period$ wide.



[1]: Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog by Clifford E. Cummings

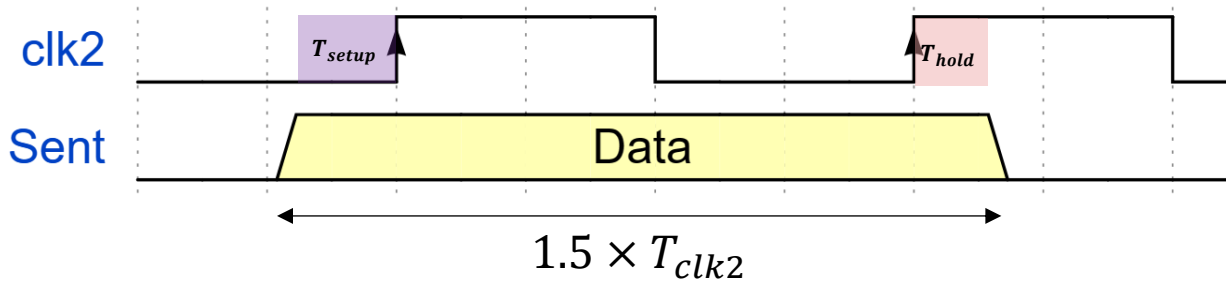
Required Pulse Width – Examples

1. (FAST TO SLOW) Let $f_{clk1} = 225\ MHz, f_{clk2} = 150\ MHz$. For how many cycle should domain 1 hold the data stable to guarantee safe capture by domain 2?

- $T_{clk1} = \frac{1}{225\times10^6} = 4.4\ ns. T_{clk2} = \frac{1}{150\times10^6} = 6.6\ ns.$
- The data should be held stable for $1.5 \times T_{clk2} = 1.5 \times 6.6 = 9.9\ ns$
- The number of cycles = $\frac{9.9}{4.4} = 2.25\ cycles \cong 3\ cycles.$

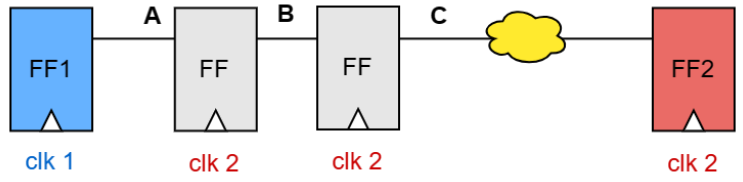
2. (SLOW TO FAST) Let $f_{clk1} = 100\ MHz, f_{clk2} = 120\ MHz$. For how many cycle should domain 1 hold the data stable to guarantee safe capture by domain 2?

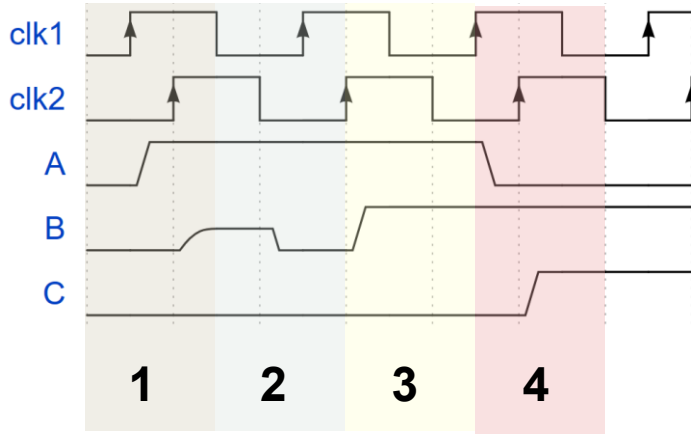
- $T_{clk1} = \frac{1}{100\times10^6} = 10\ ns.\ T_{clk2} = \frac{1}{120\times10^6} = 8.3\ ns.$
- The data should be held stable for $1.5 \times T_{clk2} = 1.5 \times 8.3 = 12.45\ ns$
- The number of cycles = $\frac{12.45}{10} = 1.245\ cycles \cong 2\ cycles.$



Required Pulse Width

- **Consider the example on the right:**
  1. Domain 1 sends signal **A** to domain 2. The change occurs close to the edge of clk2 causing a metastability in the first sync FF.
  2. Signals **B** leaves metastability and settle at a different value “logic 0”.
  3. **C** gets the wrong value “logic 0”. However, because pulse **A** was wide enough, it met another capture edge of clk2 without violating setup or hold. The correct value enters the first sync FF.
  4. The logic receives the correct value “logic 1”.
- This shows we can safely transfer a pulse from one domain to another provided that the pulse is wide enough.

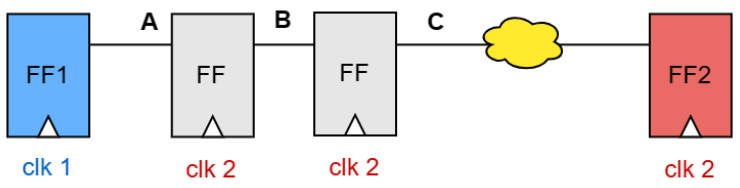


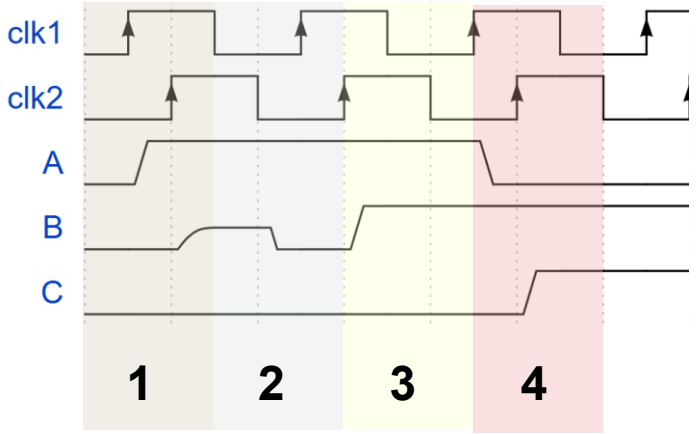


The Issue of Varying Delays/Settling Time

- **Consider the same previous example :**

1. Domain 1 sends signal **A** to domain 2. The change occurs close to the edge of clk2 causing a metastability in the first sync FF.
2. Signals **B** leaves metastability and settle at a different value "logic 0".
3. **C** gets the wrong value "logic 0". However, because pulse **A** was wide enough, it met another capture edge of clk2 without violating setup or hold. The correct value enters the first sync FF.
4. The logic receives the correct value "logic 1".



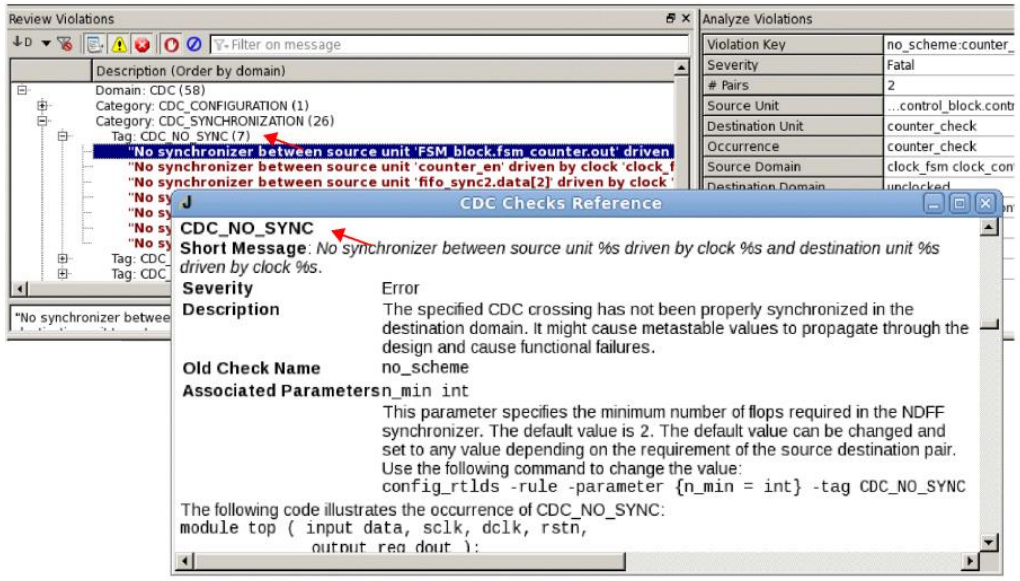


- **Lets consider another case were B in cycle (2) have metastability but settles at "logic 1" which happens to be the correct logic:**
  - This means the pulse will arrive at **C** earlier one cycle, that is at cycle (3) instead of (4).
  - The correct logic have a varying delay: it can arrive at (3) or (4).
  - This varying delay is unavoidable and the design needs to be tolerant of such delays and work under either cases
  - Advanced CDC tools inject varying delays into the CDC path and verify the design is functionally correct in all cases.

CDC Rules

FF Synchronizer Rules

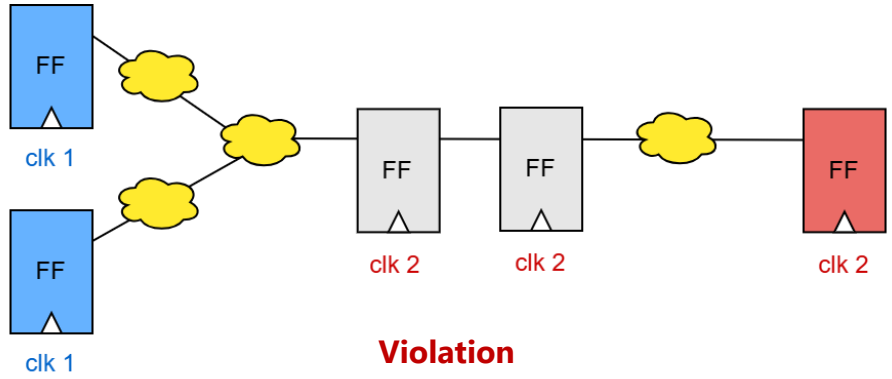
- VLSI CDC tools are used to address issues related to clock domain crossings.
- They can detect if a CDC signal is not synchronized or if it is synchronized but without the proper design practices.
- The tools check and analyze the design RTL and then generate reports addressing the potential issues. The issues reported have different severity levels (warning, critical, etc)
- In the following section we will show some of the CDC rules related to FF synchronizers

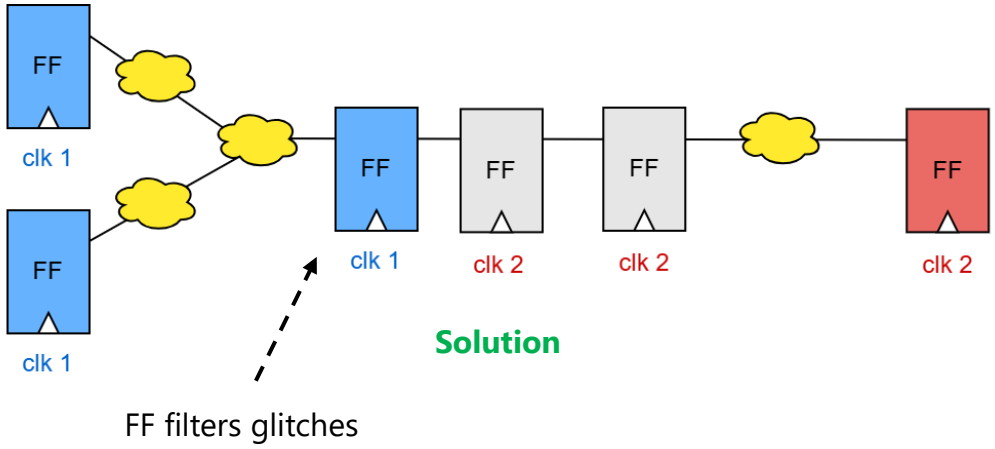


Example CDC Report From Cadence Jaspergold

Convergence in The Sending Domain / Combinational in Sync Fan-in

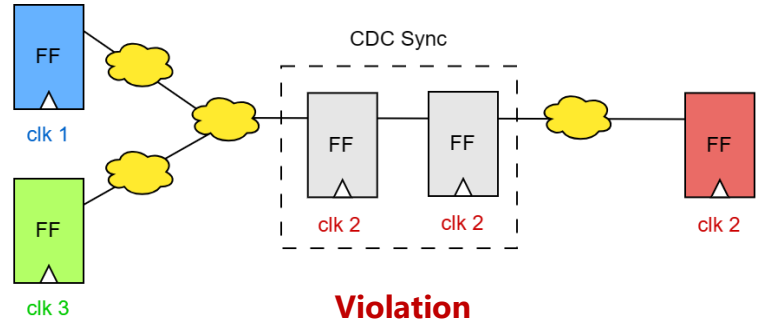
- Logic converging in the sending domain will cause lots of glitches and we saw how that affects the activity factor and therefore the MTBF.
- Most CDC tools will produce a critical message or even an error if combinational logic was detected in front of the synchronizers.
- **How to Solve:**
  - Add a register after the convergence point in the sending domain then pass the clean signal to the CDC synchronizers.

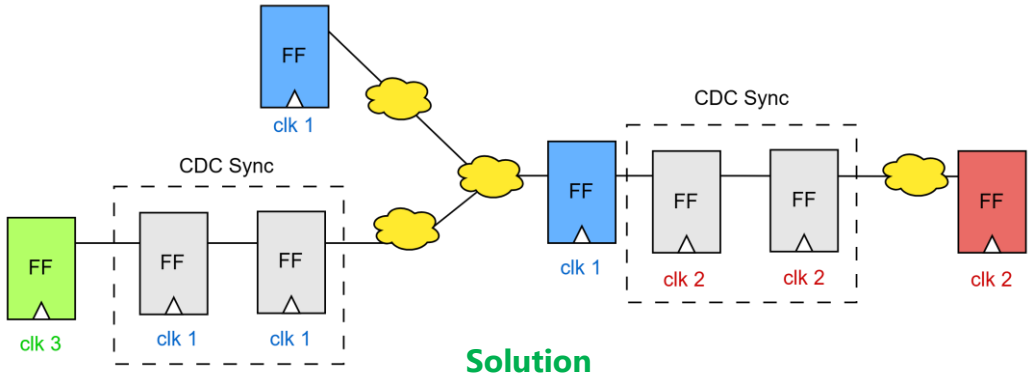


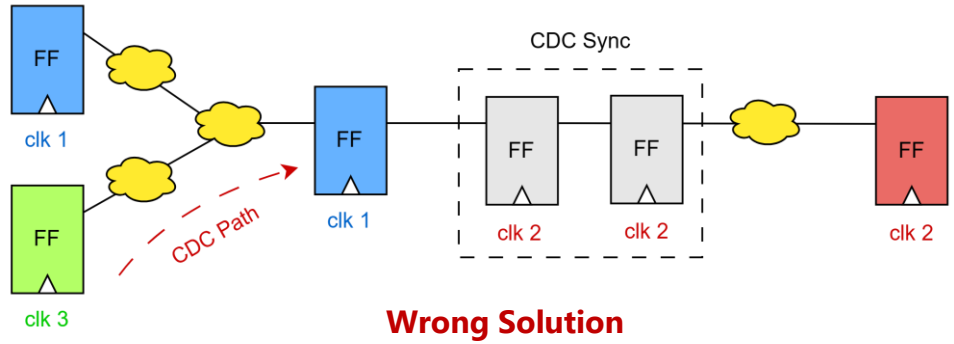


Multi Clock Fan-in

- We get this violation when signals from multiple clock domains converge and are sent to the CDC sync.
- This will increase the glitches which will damage the MTBF.
- Also, this makes it difficult for CDC tools to properly identify and analyze the design.
- **How to solve:**
  - A wrong solution is to add a glitch filtering FF. because this will lead to a new CDC path as shown.
  - The correct solution is to synchronize the signal from clk_3 to clk_1 or from clk_1 to clk_3, do the computation in a single domain, then pass it to the CDC sync.





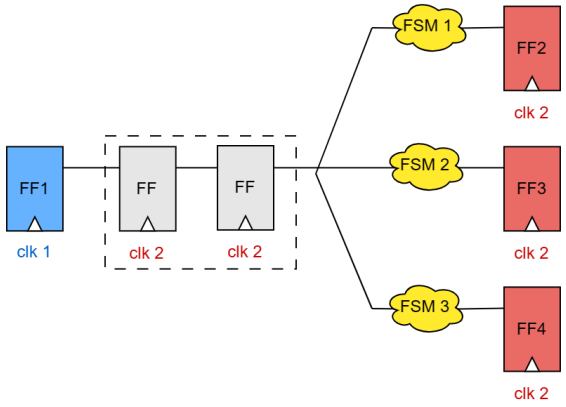


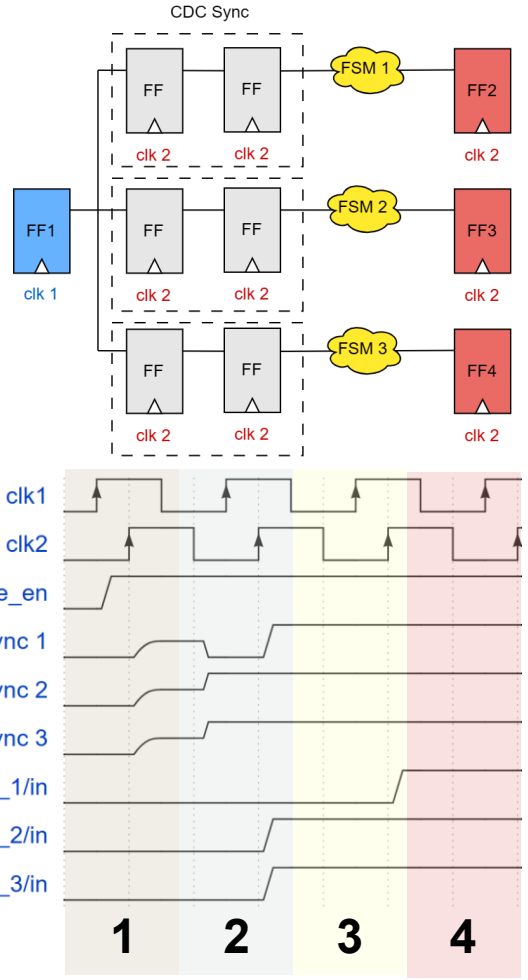
Divergence in the Sending Domain

- Divergence occurs when a CDC signal is passed by multiple synchronizers to the other domain.
- **Consider the example on the right:**

1. Domain 1 sends a write_enable signal to domain 2 through 3 parallel synchronizers. The change occurs close to the edge of clk2 causing a metastability in all 3 synchronizers.
2. The 1st sync exits metastability and settle at logic 0. The other 2 syncs settle at logic 1.
3. The FSMs in domain 2 receive the signals from the syncs. The 1st FSM doesn't see an active write_enable signals so it remains IDLE, the other 2 FSMs see an active signal and act accordingly. We have incoherency in the system.
4. In the next cycle, all FSMs receive the correct value but the damage is already done.

- **How to solve[1]:**
  - Pass the signal with one synchronizer then diverge/fanout at the receiving domain[1]

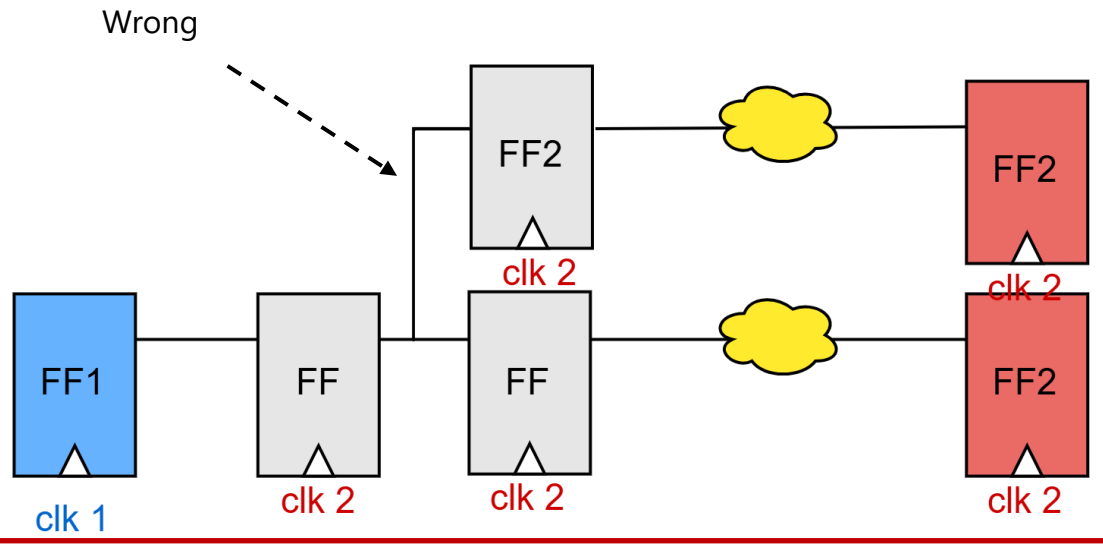




[1]: The issue might appear silly and rare to happen however it may occur due to lack of good communication among the team members: One engineer create a sync inside their block to pass a global signal to another domain, another engineer create another sync inside their module to pass the same signal. It's better to create a single and separate module for CDC synchronization

Divergence of a Metastable Signal

- This issue is similar to the previous one, you shouldn't read/fanout the value between the synchronizers.
- **How to solve:**
  - Pass the signal with one synchronizer then diverge/fanout at the receiving domain

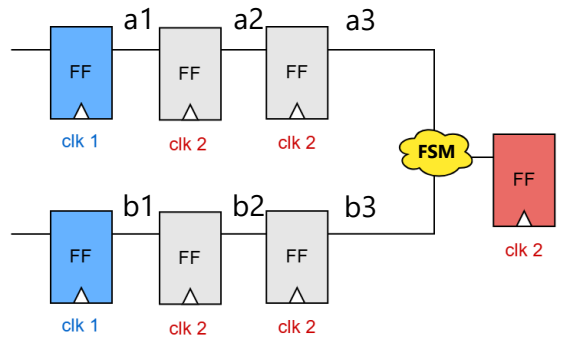


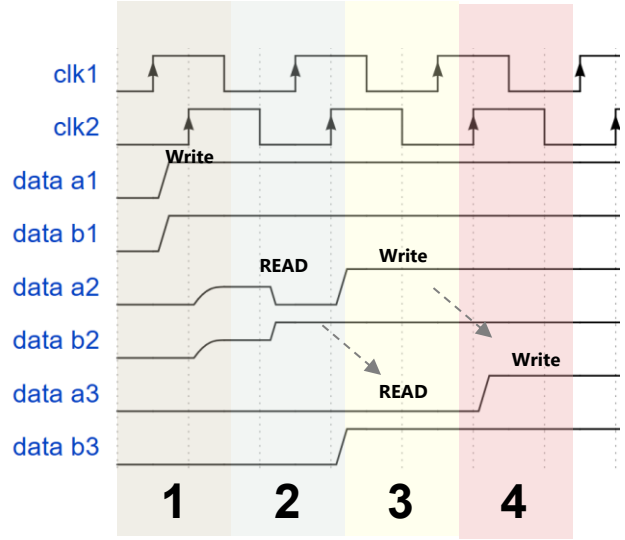
Multi Signal Reconvergence in the Receiving Domain

- When multiple signals are passed from the sending domain and then converge in the receiving domain, as shown in the diagram, we may get functional errors due to the difference in the settling time between the two signals.
- **Consider the example on the right:** Domain 1 sends a 2-bit control signal to domain 2.
  Initially we are in IDLE=2'b00

1. Domain 1 sends "2'b11 (WRITE)" to domain 2. The change occurs close to the edge of clk2 causing a metastability.
2. Signals a2 and b2 leave metastability and settle at different values "2'b10" (READ).
3. The value "2'b10" (READ) is passed to a3 and b3 and then to the combinational logic causing it to go to a (READ) state while the intended state was (WRITE).
4. The logic receives the correct value (WRITE) later but the damage is already done.

- This example shows the problem with sending multiple signals from one domain to another even with just 2 bits.
- **How to solve:**
  - Converge these signals in the sending domain and send them to the receiving domain as one signal. However, this is not always possible.
  - Use gray encoding to make sure only one signal changes at a time (Will be discussed later)
  - Use MUX synchronization scheme to pass these signals as a group (Will be discussed later)





Conclusion

- **We want to know what we solved till now. Our CDC concerns were:**
  - Data corruption: Partially fixed. The system, till now, can only send 1-bit data. Also, this data has a varying arrival time.
  - Data incoherence: Fixed. The metastable value settles within the synchronizers at 0 or 1 and then propagates to all the 2nd domain blocks with the same settled value
  - Data loss: Fixed. The pulse is wide enough that it won't be missed by domain 2
  - Data duplication: Not fixed
  - Chip burning: Fixed. We limited the metastability propagation between the synchronizers and reduced its occurrence frequency.
- In the next parts we will see how to deal with the remaining concerns.

References

1) https://www.uio.no/studier/emner/matnat/ifi/IN3160/v21/timeplan/in3160-l92-clock-domains.pdf
2) https://ieeexplore.ieee.org/document/1676187
3) https://www.edn.com/keep-metastability-from-killing-your-digital-design/
4) https://people.ece.ubc.ca/~edc/7660.jan2018/lec11.pdf
5) https://www.onsemi.com/pub/Collateral/AN1504-D.PDF
6) https://www.linkedin.com/in/lukas-vik/recent-activity/articles/

Thank You!