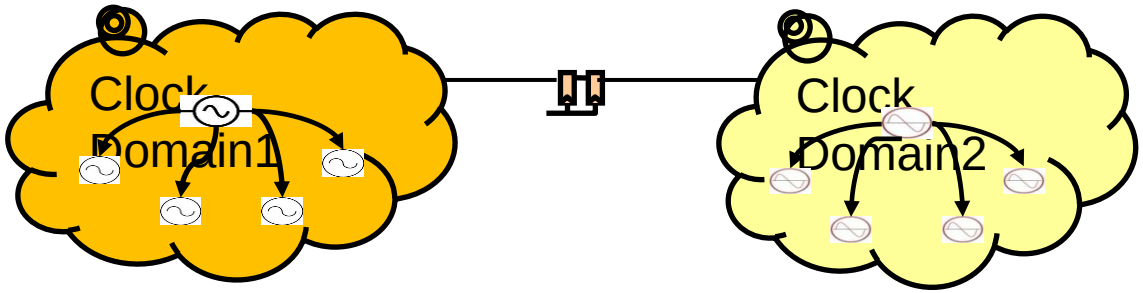
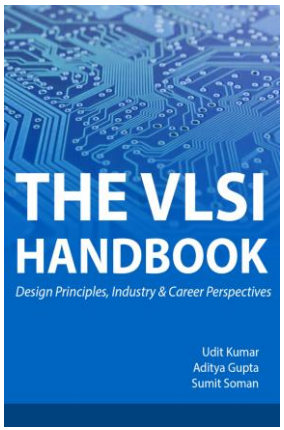


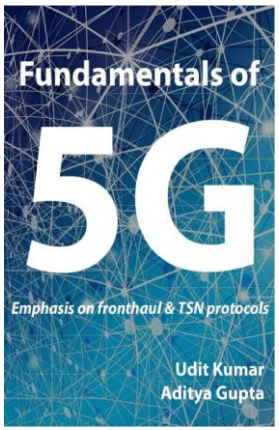
Clock Domain Crossing: Issues & Solutions

Udit Kumar, PhD, IIT Delhi.

15+ years experience, Author

https://www.linkedin.com/in/udit-kumar-phd-iit-delhi





Learn VLSI

Website: https://www.sites.google.com/view/learnvlsi

LinkedIn: https://www.linkedin.com/company/learnvlsi

Note: Views expressed here are personal views and not endorsed by present or past employer.

Disclaimer

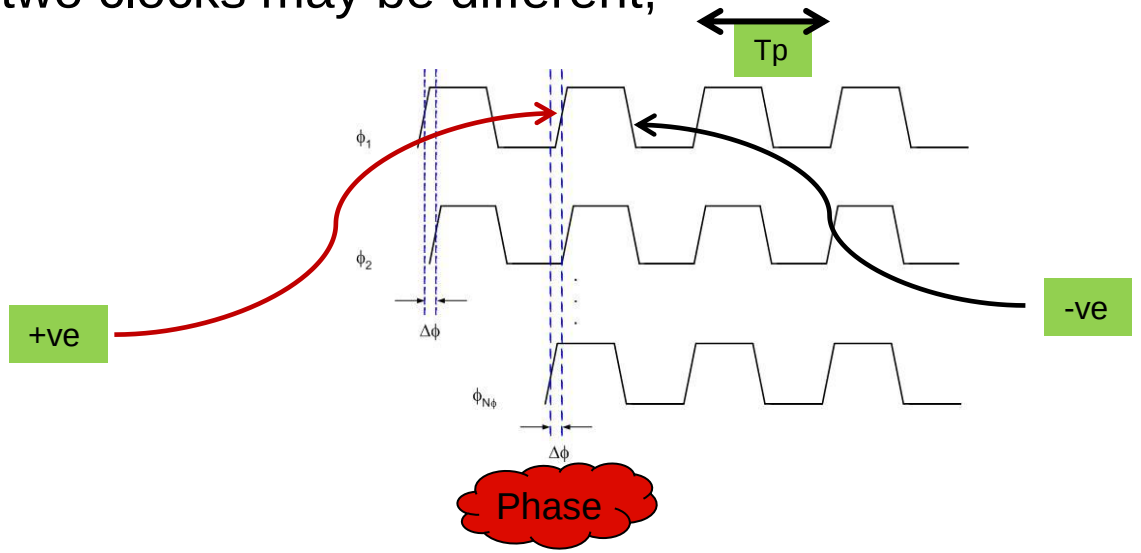
- **The intention of this presentation is information sharing. So consider this material as information purpose only.**
- We explicitly disclaim any liability for mistakes and omissions in the material presented.
- We have done our best to ensure the correctness of the material and have no obligation or duty to any person or organization for any loss or damages stemming from the contents.
- We make no claim, promises, or guarantees regarding the correctness, completeness, patent infringement, or sufficiency of the same.
- Take prior approval for Commercial usage of this information.

Outline

- What is Clock Domain Crossing (CDC)
  - The Problem
  - Leads to Chip Failure, No software fix!!
- CDC Issues & Solutions
  - Control Crossing
    - Synchronizer
    - Reconvergence
    - Divergence
    - Slow to Fast Crossing
    - Passing a Pulse
  - Data Path Crossings
  - Design Flow impact on CDC Paths
    - Synthesis
    - Low Power
    - Physical Implementation
- Design Flow and CDC Checking tools
- Conclusions

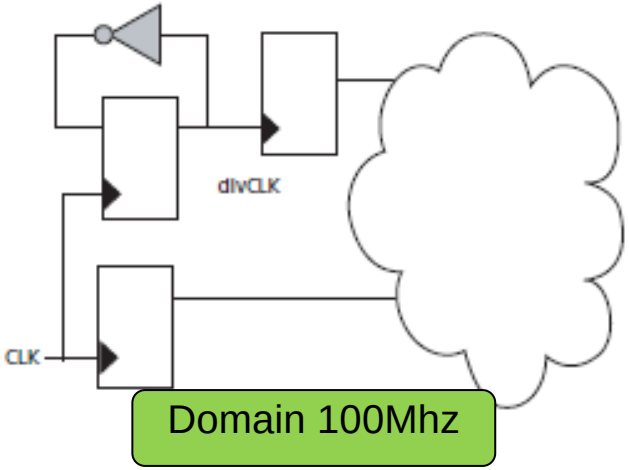
Clock in a Digital System

- A clock has
  - *Period* of repetition (linked with its frequency)
  - *Phase* depicts rise & fall transitions
  - A flop can be triggered thru any of clock edges.
  - The phase of two clocks may be different,

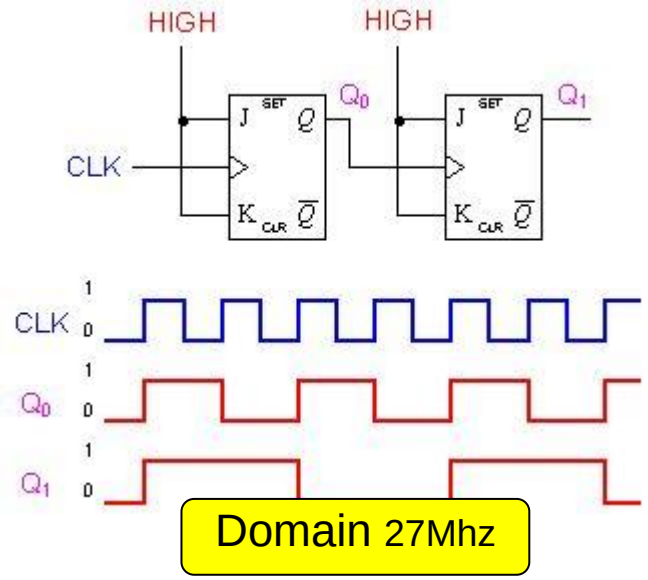


Domain of a clock

- Logic which is triggered by clock (or derived clocks)
  - Also known as Synchronous system.

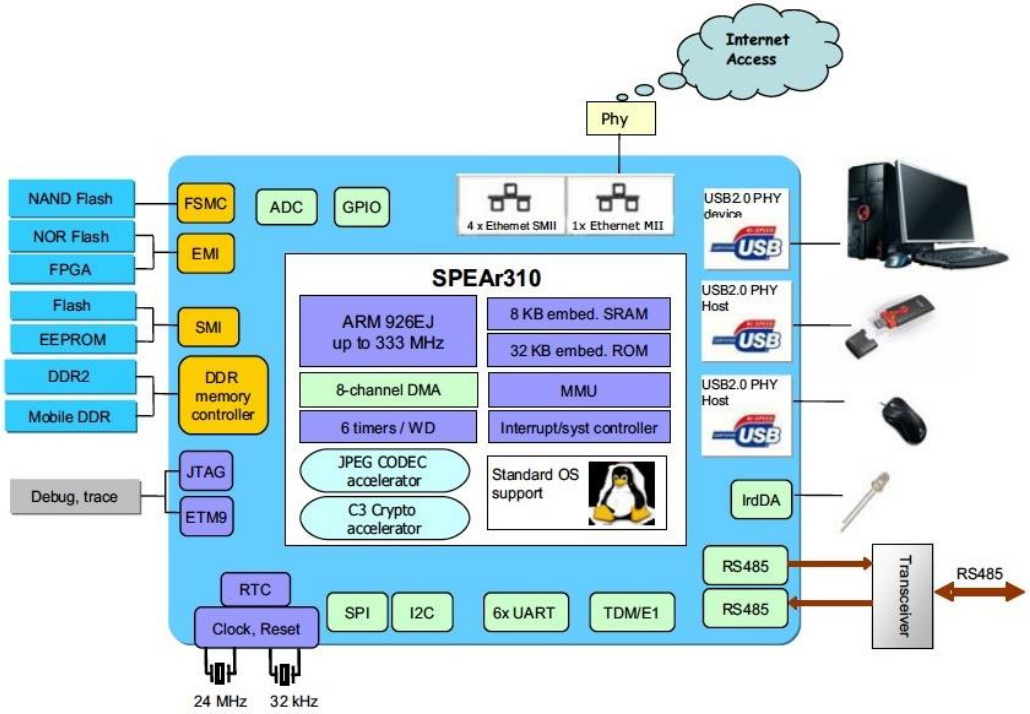


- Conversely, domains with clocks of variable phase and frequency are *different* clock domains.
  - Also known as Asynchronous.

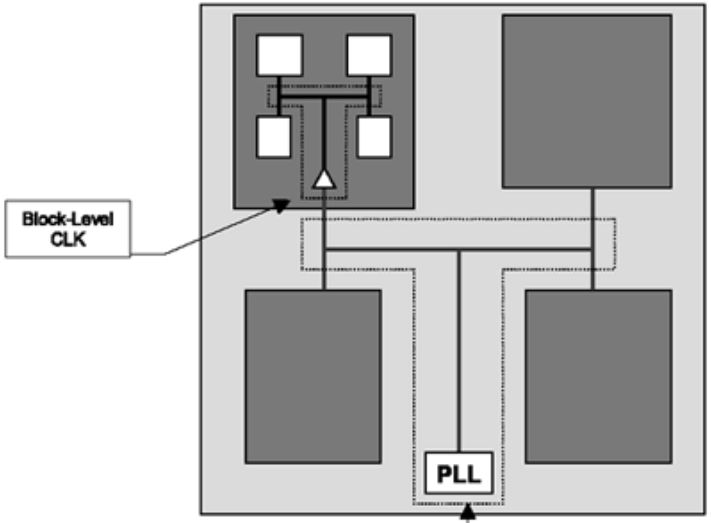


Why have multiple clock domains?

- SoC have multiple interfaces with very different clock frequencies.



Synchronous clocks but *considered* Async.



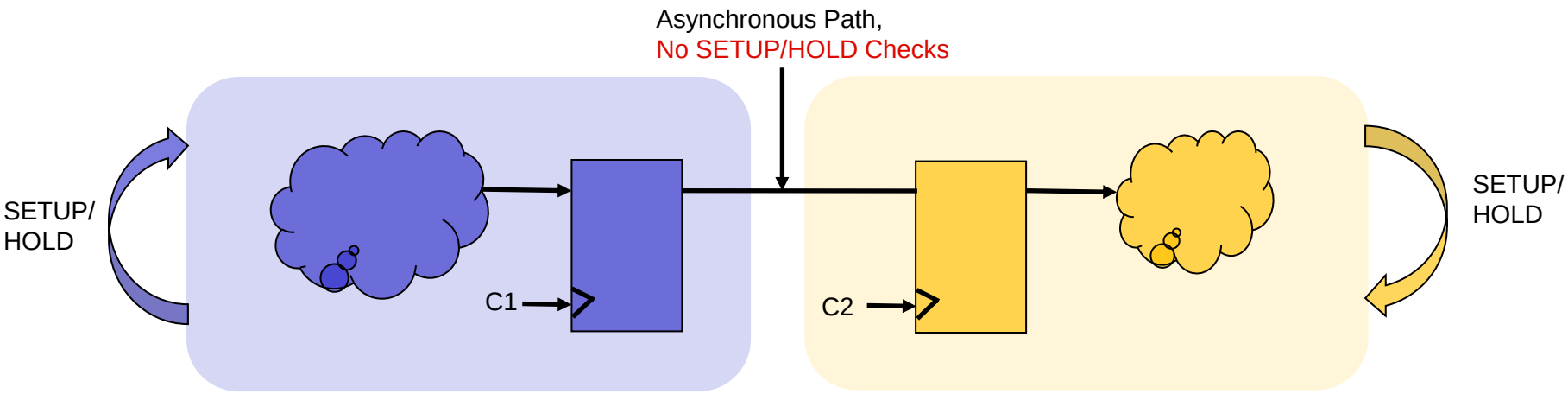
To reduce Clocktree Balancing complexities.

Precautions when Crossing a domain

- Domain crossing can be seen in real life:
  - Crossing countries
  - Train changing tracks
  - Etc…
- Needs special agents to make it error free & safe.
- For the same reason, precaution is needed when signals cross over domains.

The CDC Path

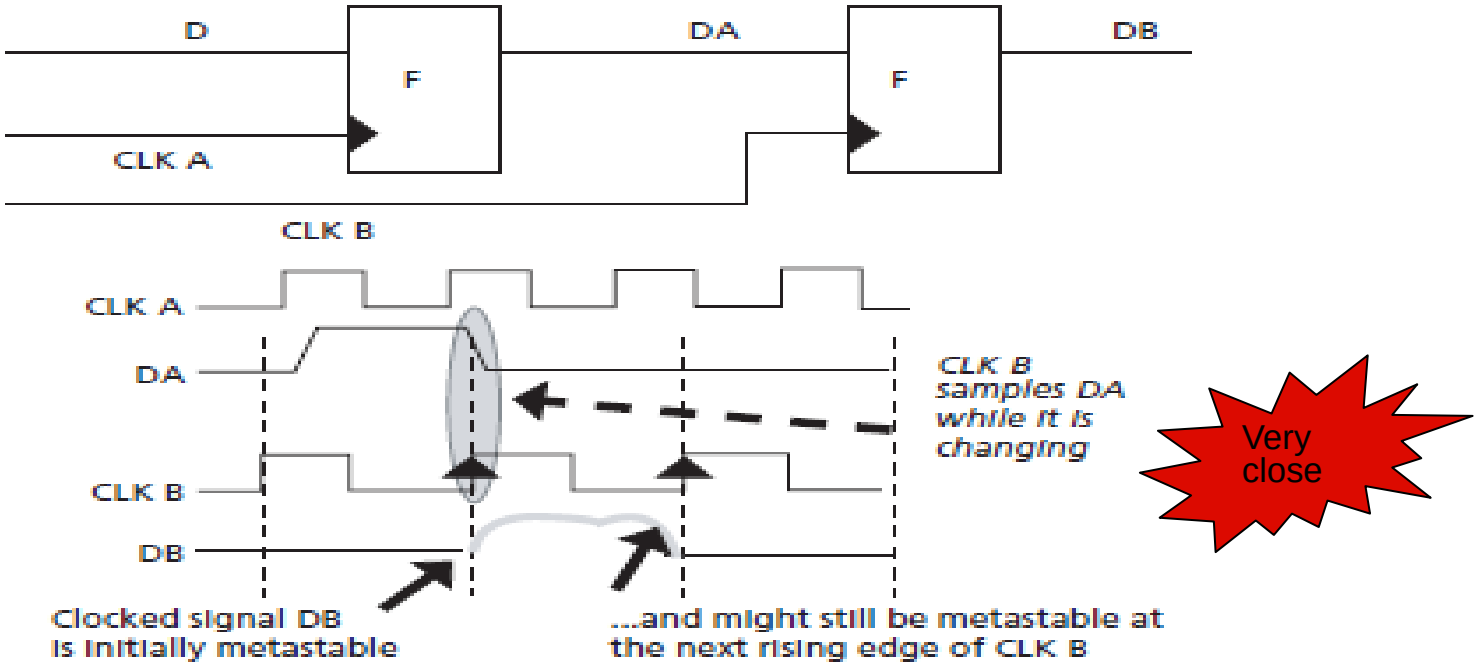
- When clocks are asynchronous, the signals that interface between are called clock domain paths.



- Within each domain, setup & hold time checks ensure proper functioning of the design.
- **No timing check exists on CDC Paths.**

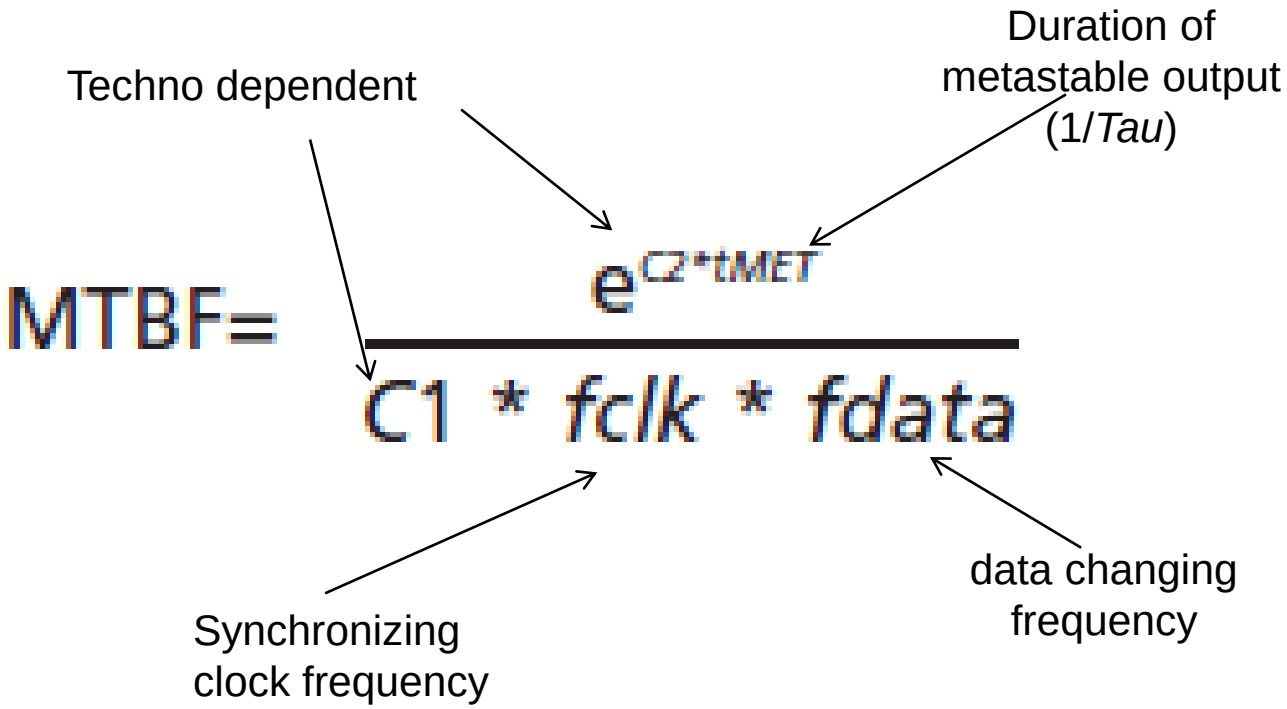
Meta-stability

- A flip-flop needs input to be stable before and after the clock edge. (Setup & Hold Time) .
- In CDC crossing, there will be setup & hold violations.
- Then, the output of flip-flop may take much longer time to reach a valid logic level. This is called metastability.



MTBF - Mean Time Between Failure

- Reciprocal of failure rate: Should be as high as possible
- Failure means signal goes metastable after first stage synchronizer and ***continues to be metastable one cycle later*** when it is sampled in the second stage synchronizer flop.

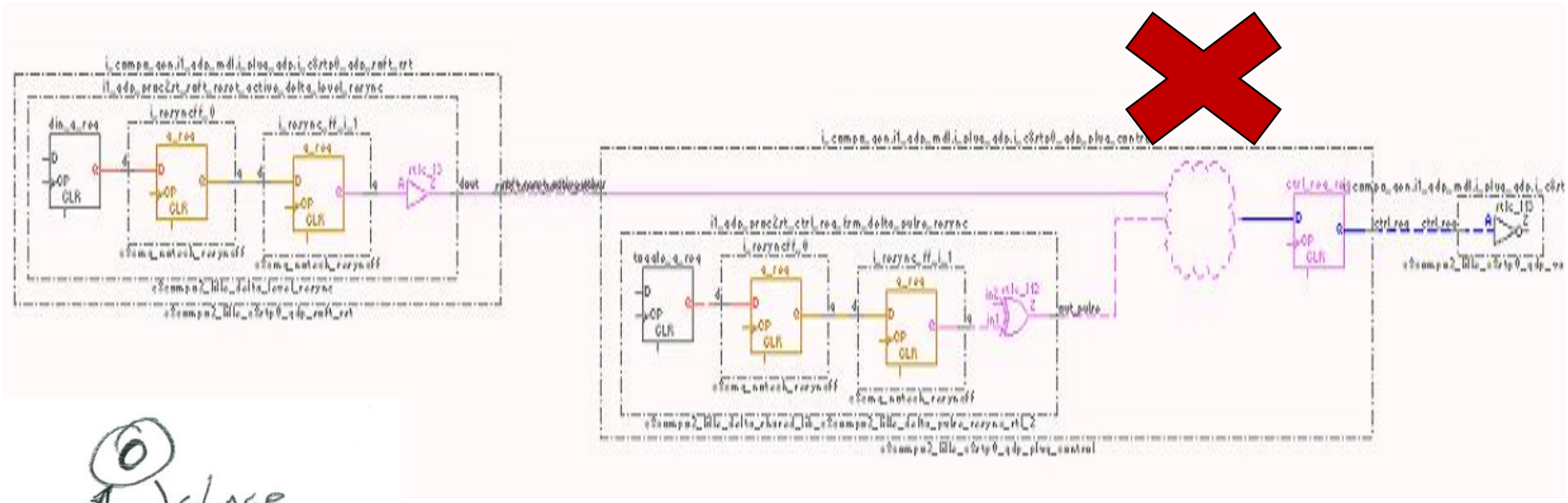


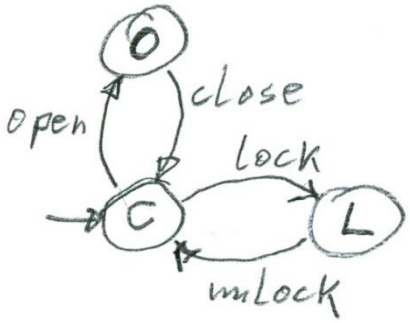
Example on how to calculate MTBF

- MTBF (Mean Time Between Failures): Average time a system will run between failures.
- A system has 4000 components with a failure rate of 0.02% per 1000 hours. Calculate MTBF.
- No of failure per hour=
  - (Failure rate) * (Number of components)
  - (0.02 / 100) * (1 / 1000) * 4000 = $8 * 10^{-4}$ per hours
- MTBF = 1 / ($8 * 10^{-4}$) = 1250 hours

Why is it important?

- Multiple cases of Chip Failure due to this effect across the world
  - Convergence of Signals leading to control FSM moving to unwanted state.

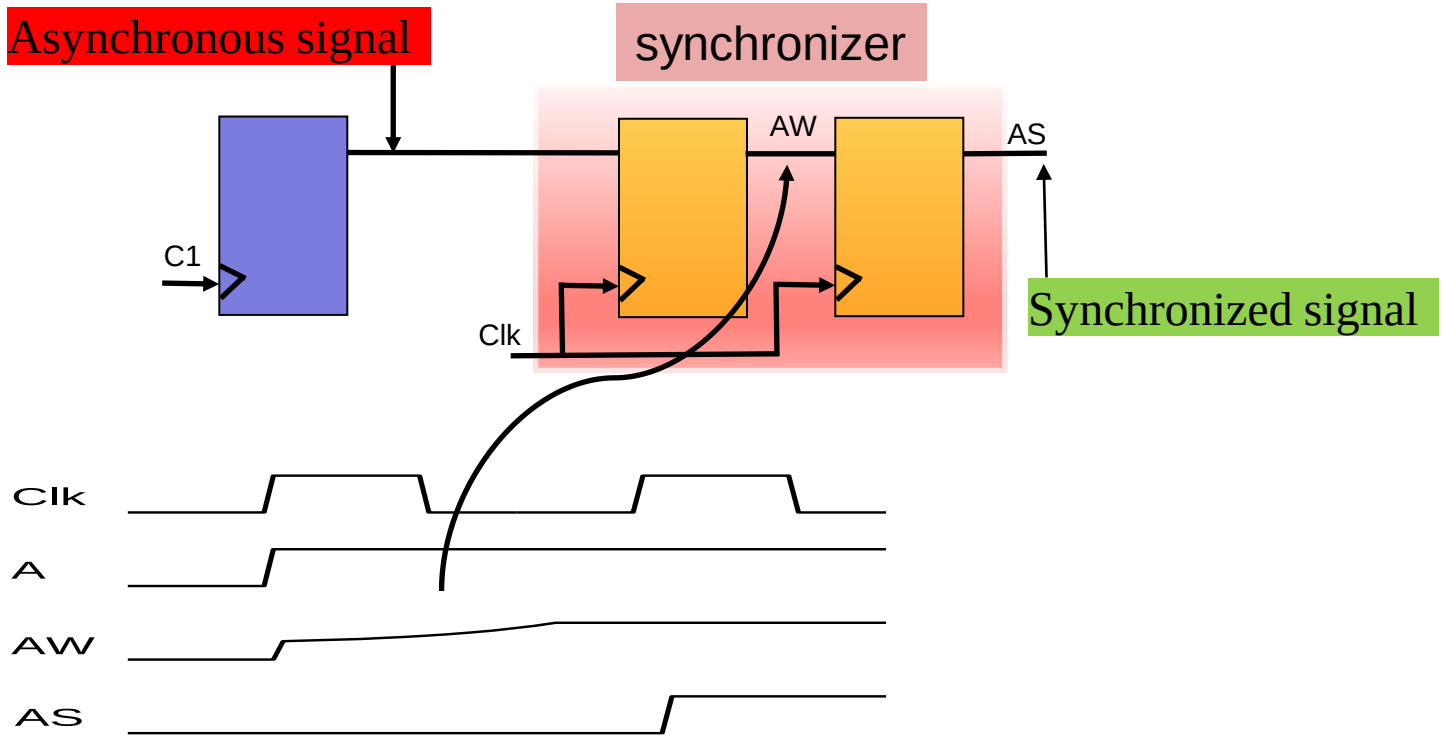




Leads to Chip Failure, No Software fix!!

Clock crossing: Minimum Solution

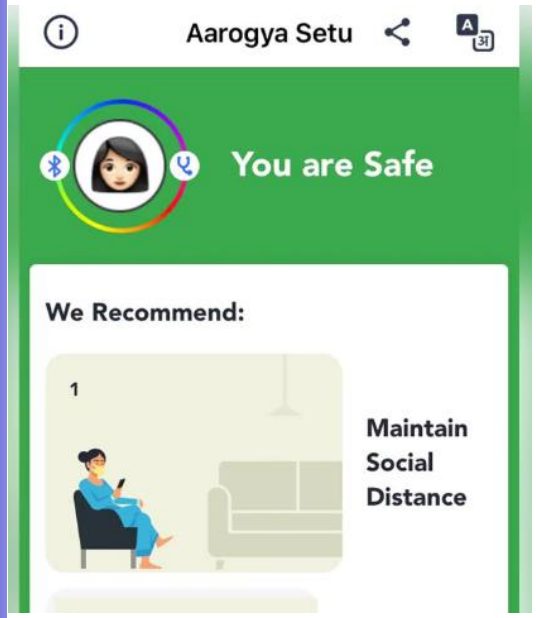
- A synchronizer is a device that samples the asynchronous signal and output a signal that is synchronized to a destination clock domain.

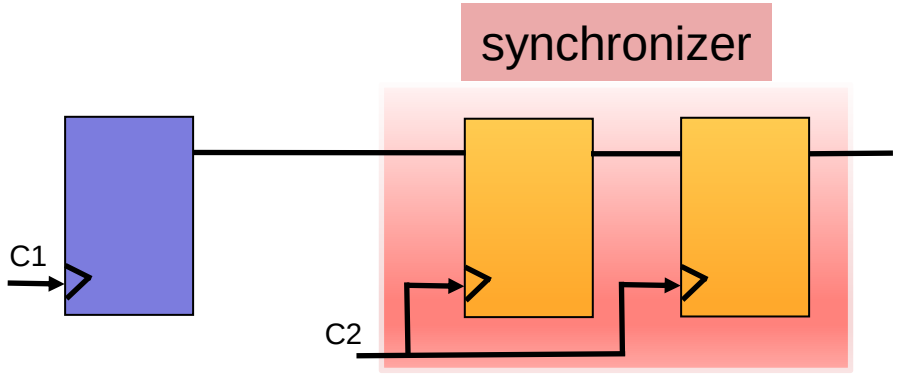


Synchronizing cell should come from special cell library.

Assignment:
How to calculate number of Synchronizer flop for given frequency.

NEED TO FOLLOW MORE RULES



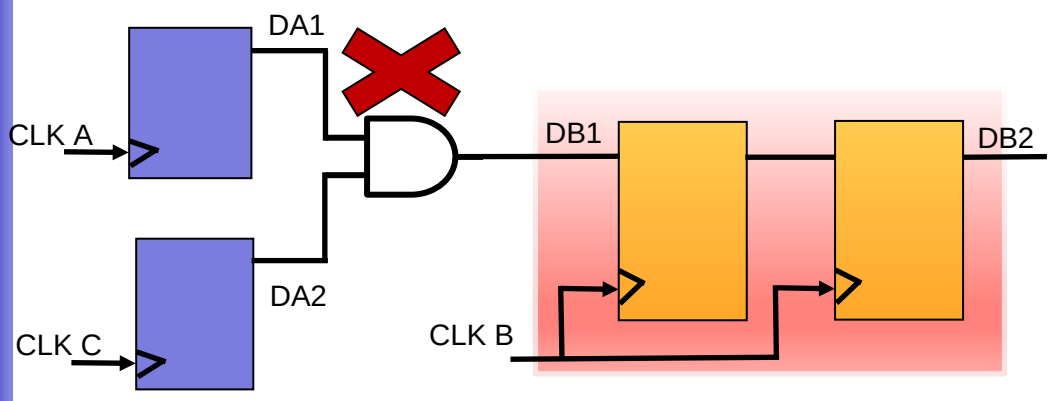


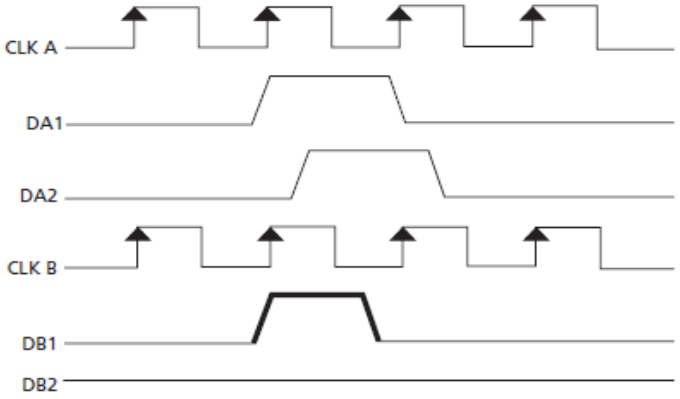
Design has Synchronizer But..

Having a Synchronizer is not enough, One needs to follow more rules!

No combinational Logic at Crossing point

- Unconstrained path has delay imbalance, will lead to glitches in the path.

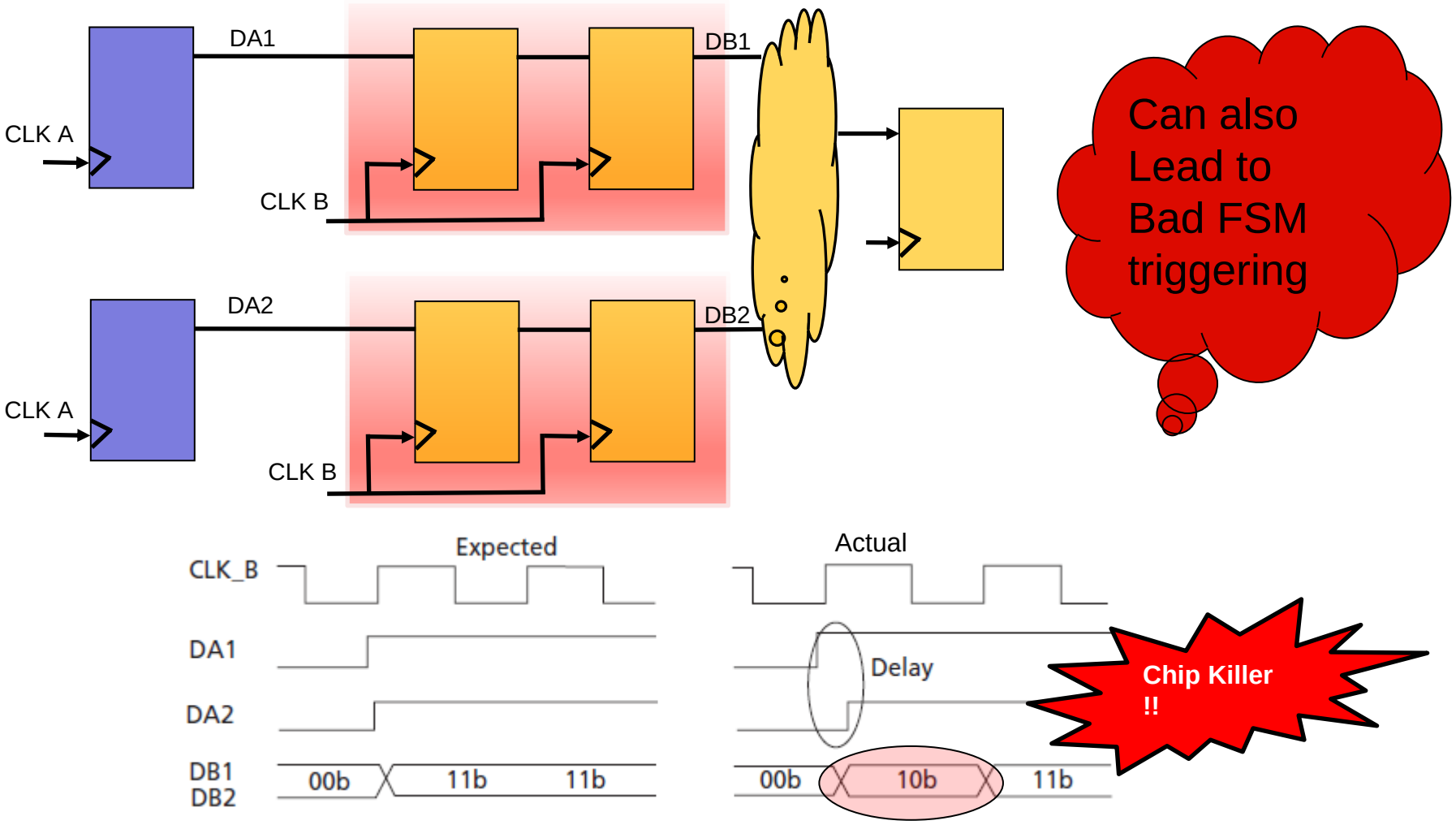




Make sure that CDC signal is directly coming from a flop.

Reference: Cadence Technical paper, http://www.fpga-faq.com/FAQ_Pages/cdc_wp.pdf

Re-Convergence of Synchronized Signals

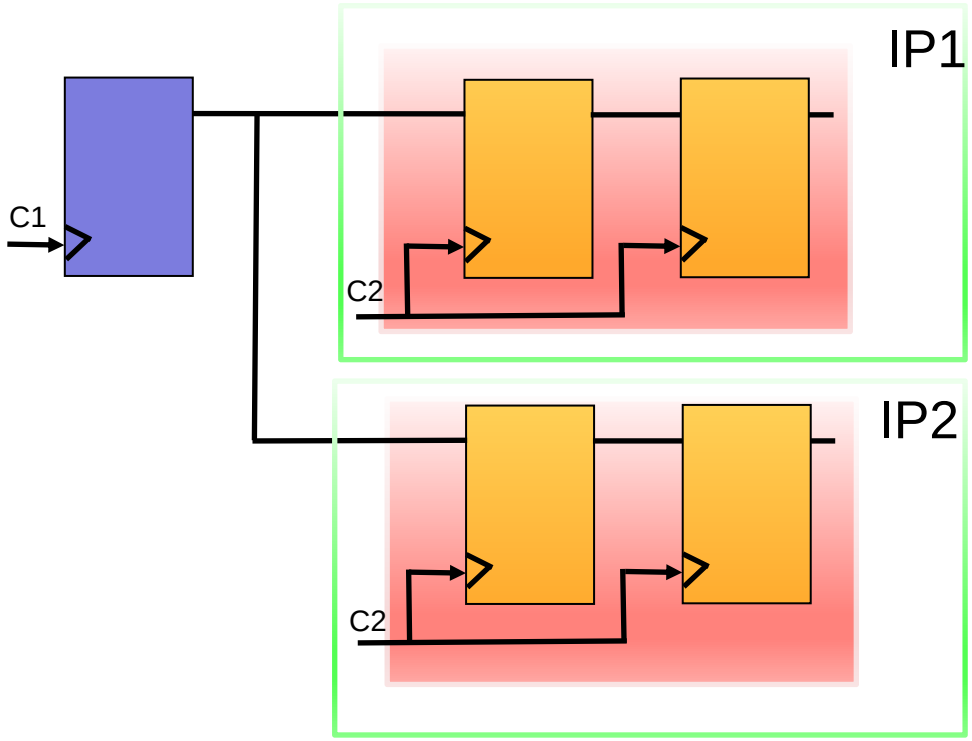


Compute the controls & then do one transfer across domain

Reference: Cadence Technical paper, http://www.fpga-faq.com/FAQ_Pages/cdc_wp.pdf

Divergence in Cross over path

- Divergence in Cross over path

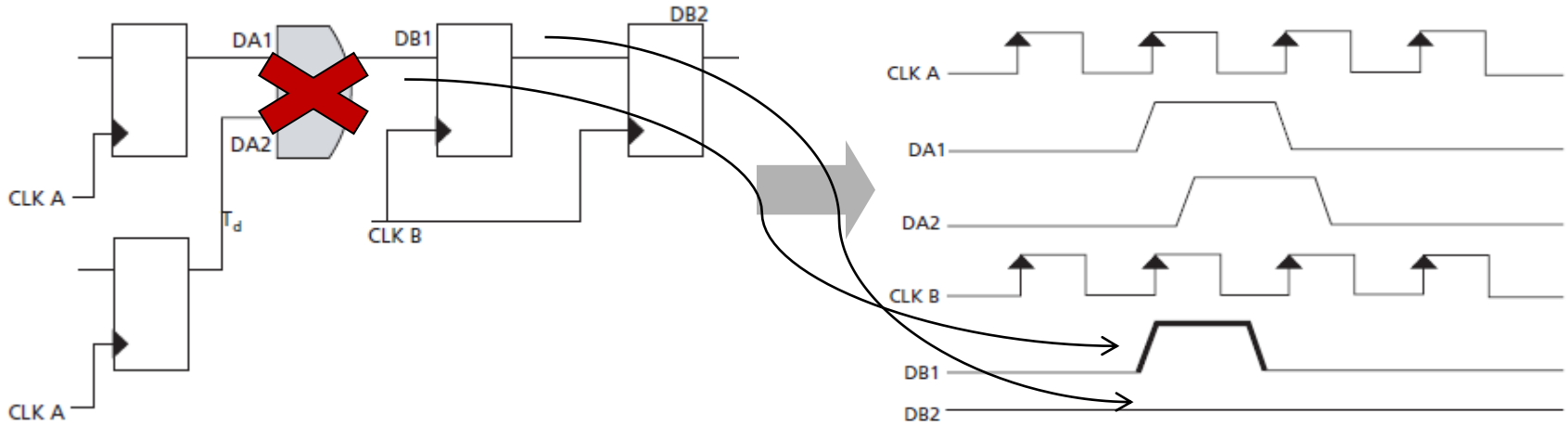


Avoid having multiple Synchronizer for one control signal.

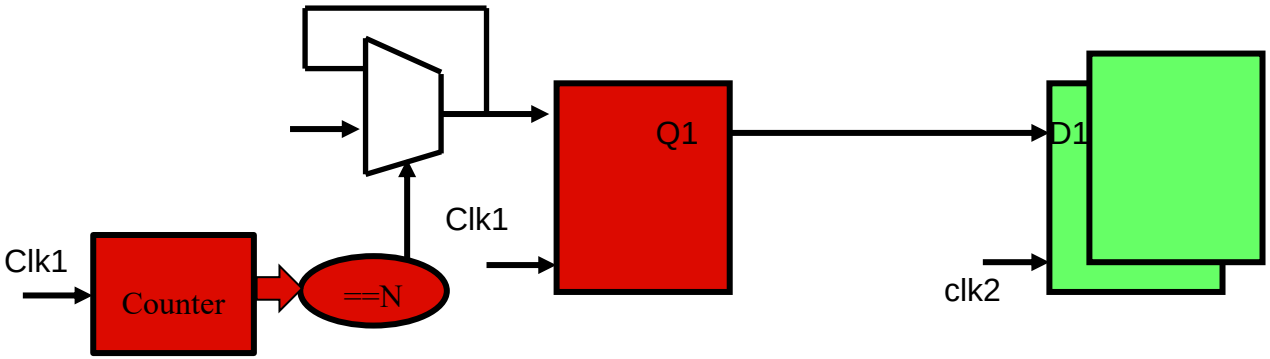
Crossing Fast to Slow

Data Hold problem

(Signal crosses from a fast clock domain to a slow clock domain)

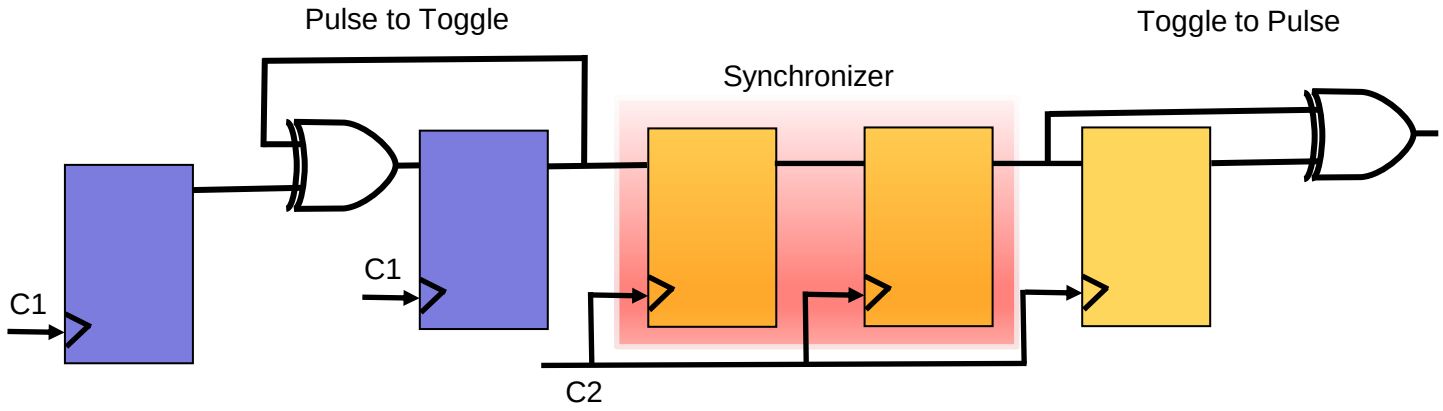


- To avoid CDC issues, Hold the data till a time-out (using Pulse extenders).



Hold data (for minimum 3 RX clock edge) till the transfer takes place (Traffic police).

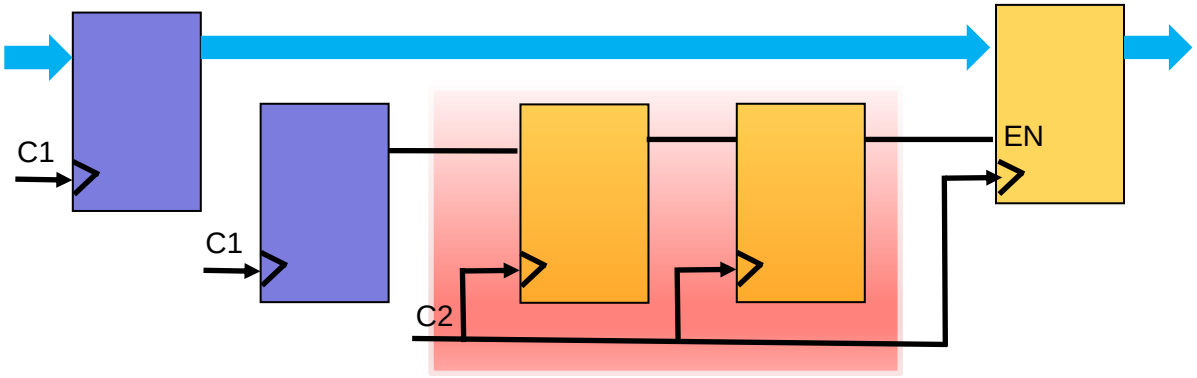
Passing a Pulse from One domain to Another Domain



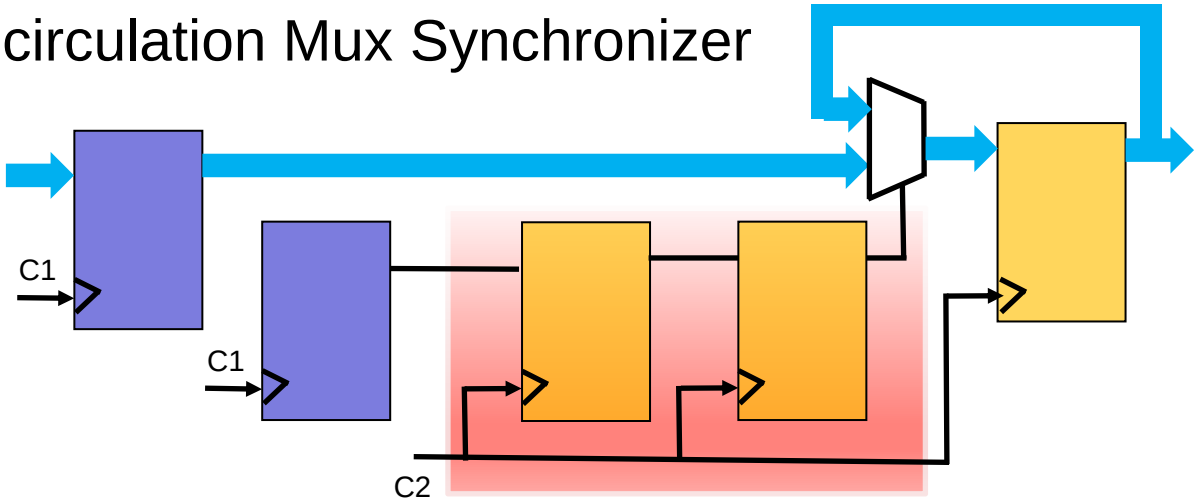
Assignment:
How to successfully pass a signal from faster clock frequency to slower clock frequency.

Data Path: Multi bit transfer

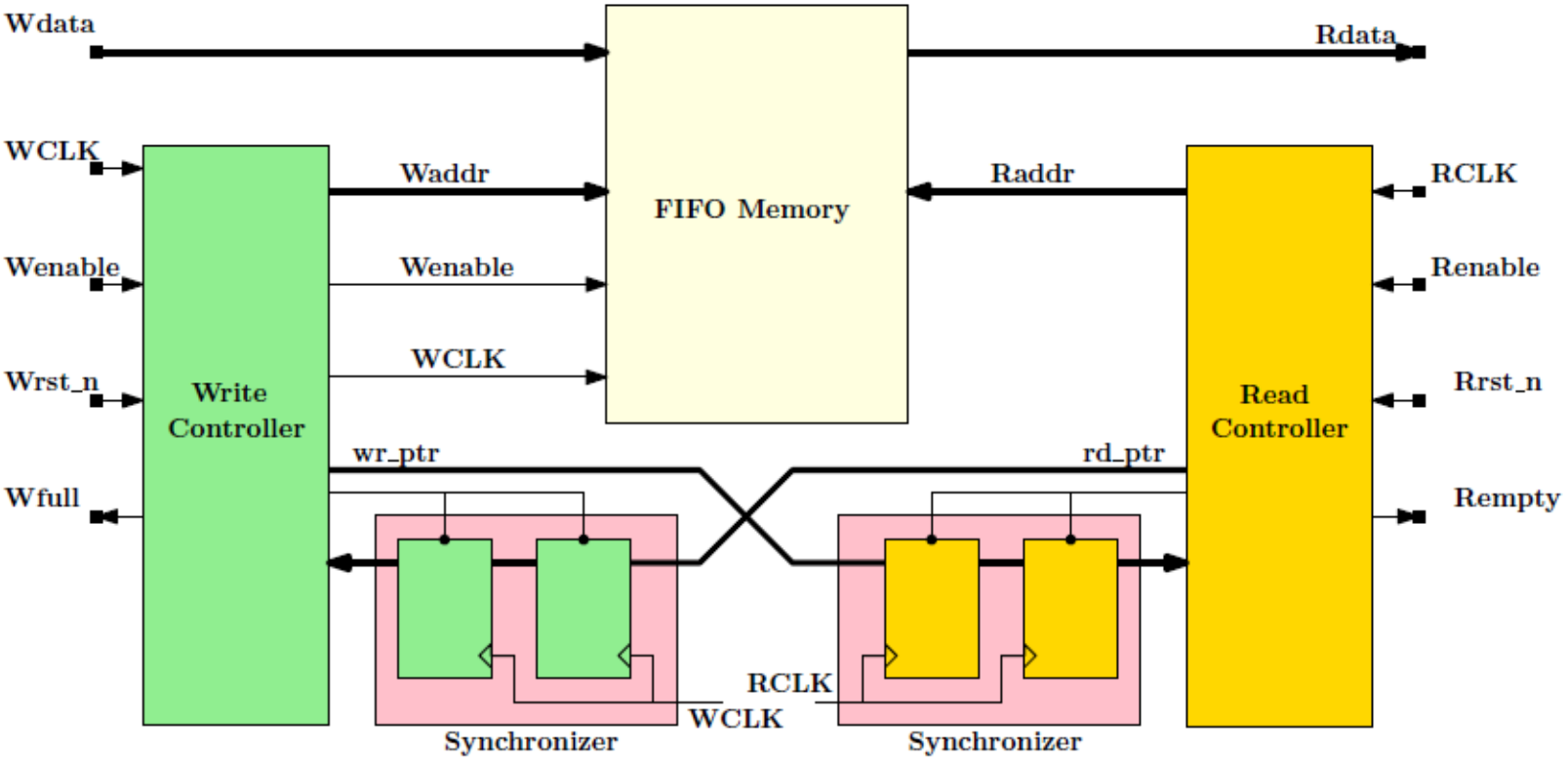
- Enable Based Synchronizer



- Recirculation Mux Synchronizer



Data Path: Asynchronous FIFO



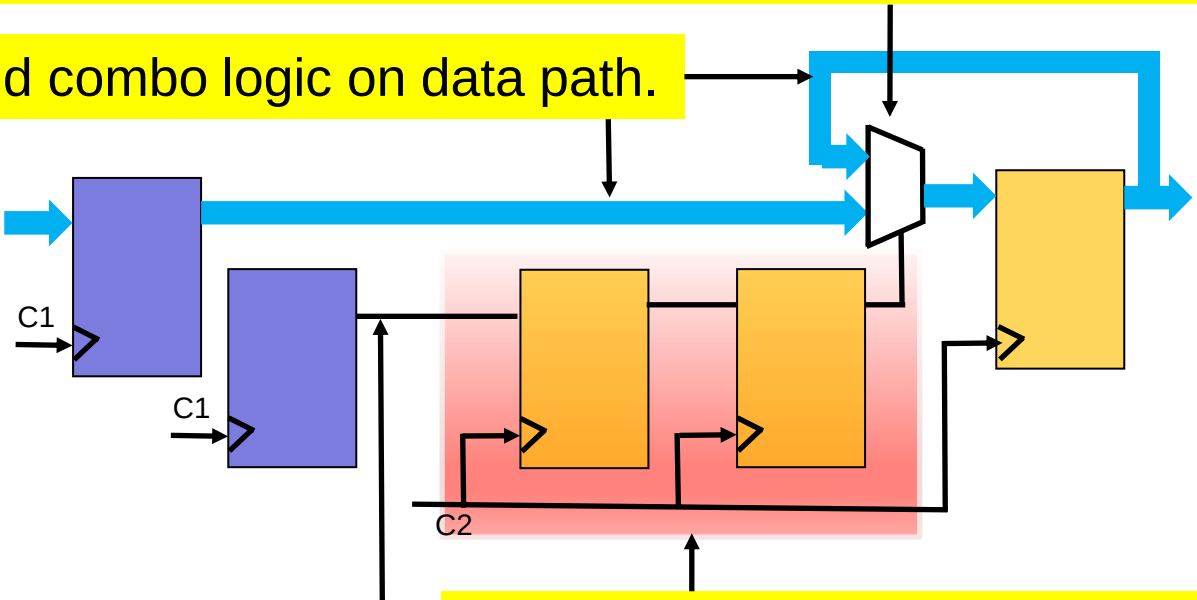
Use Gray encoding for read & Write Pointer.

Assignment:
Design a non power of 2 depth FIFO, having even depth with use of gray encoding pointer. E.g. FIFO with depth 12.

Data Path: Recommendations

Have glitch free mux in the data path from special cell library.

Avoid combo logic on data path.

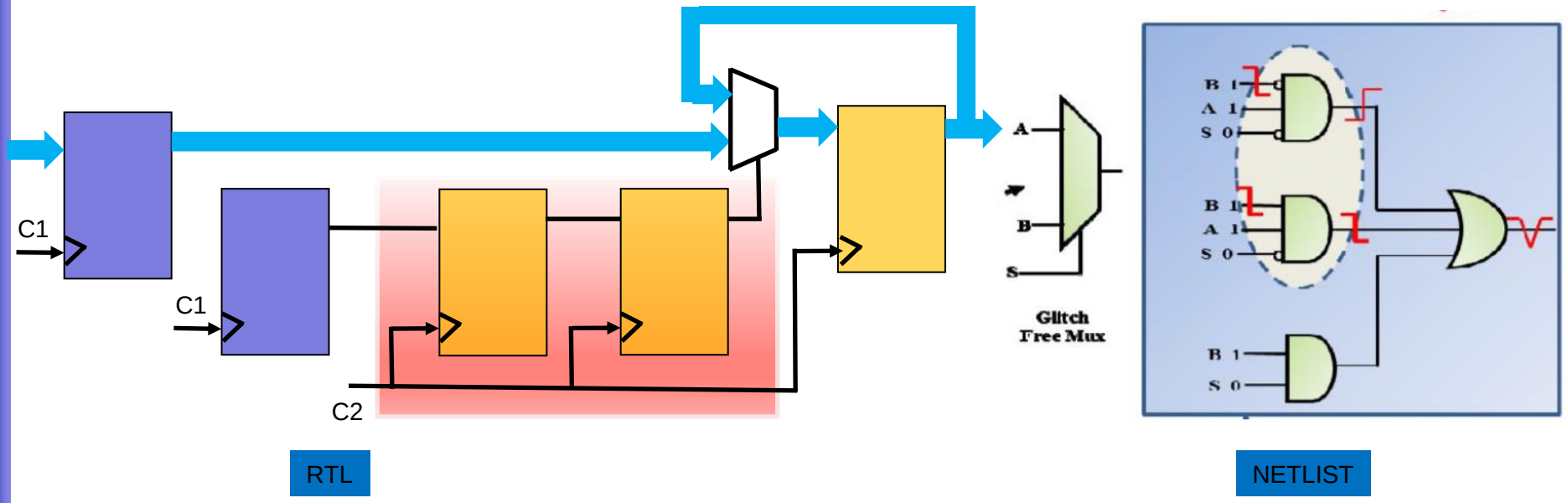


Have Synchronizer from special cell library

Avoid combo logic on control path.

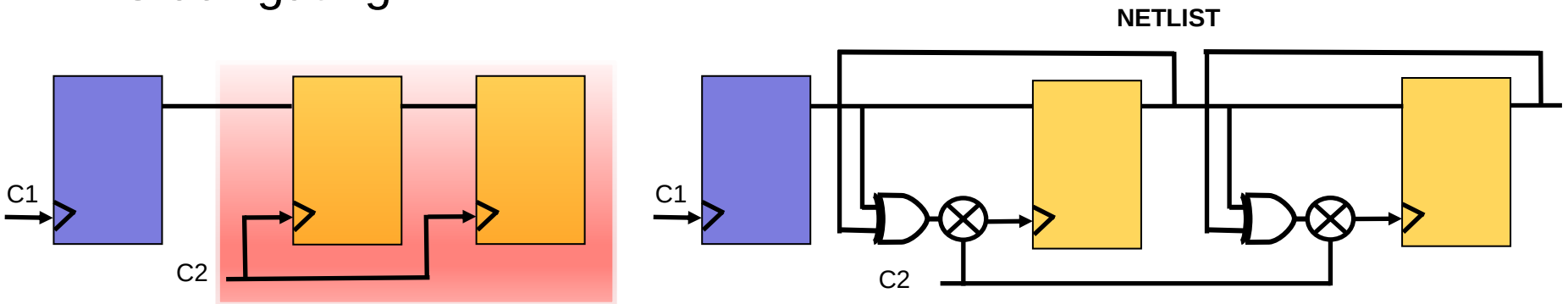
Synthesis Impact on CDC Paths

- Glitch Free Mux



RTL NETLIST

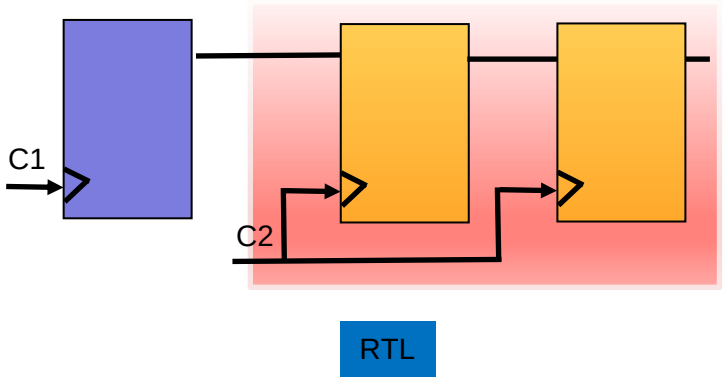
- Clock gating

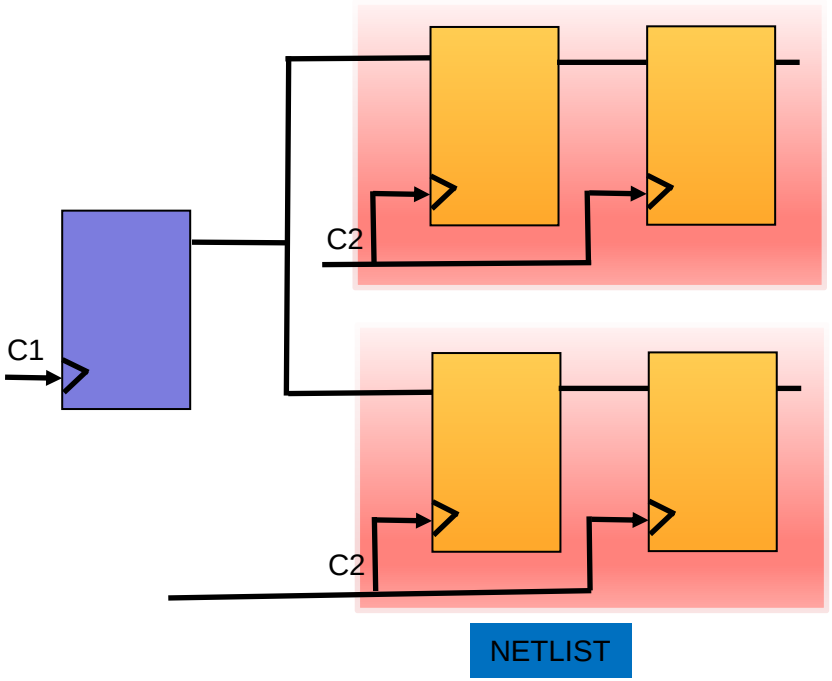


RTL NETLIST

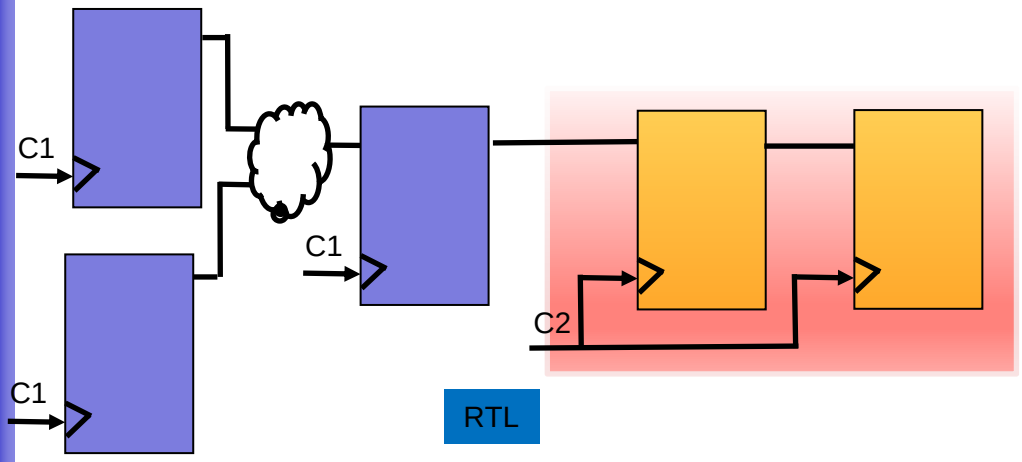
Synthesis Impact on CDC Path

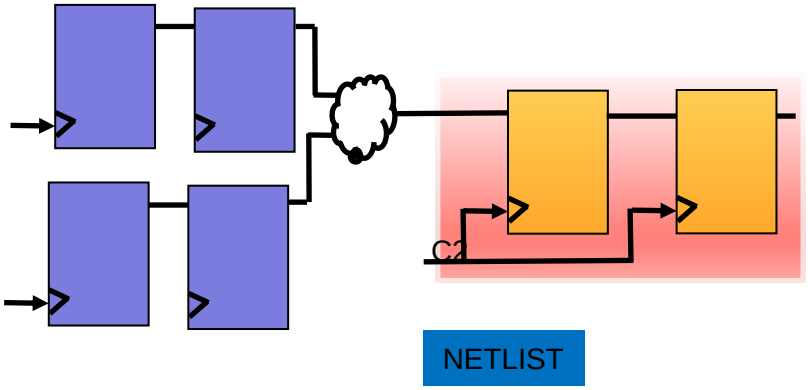
- Register Replication



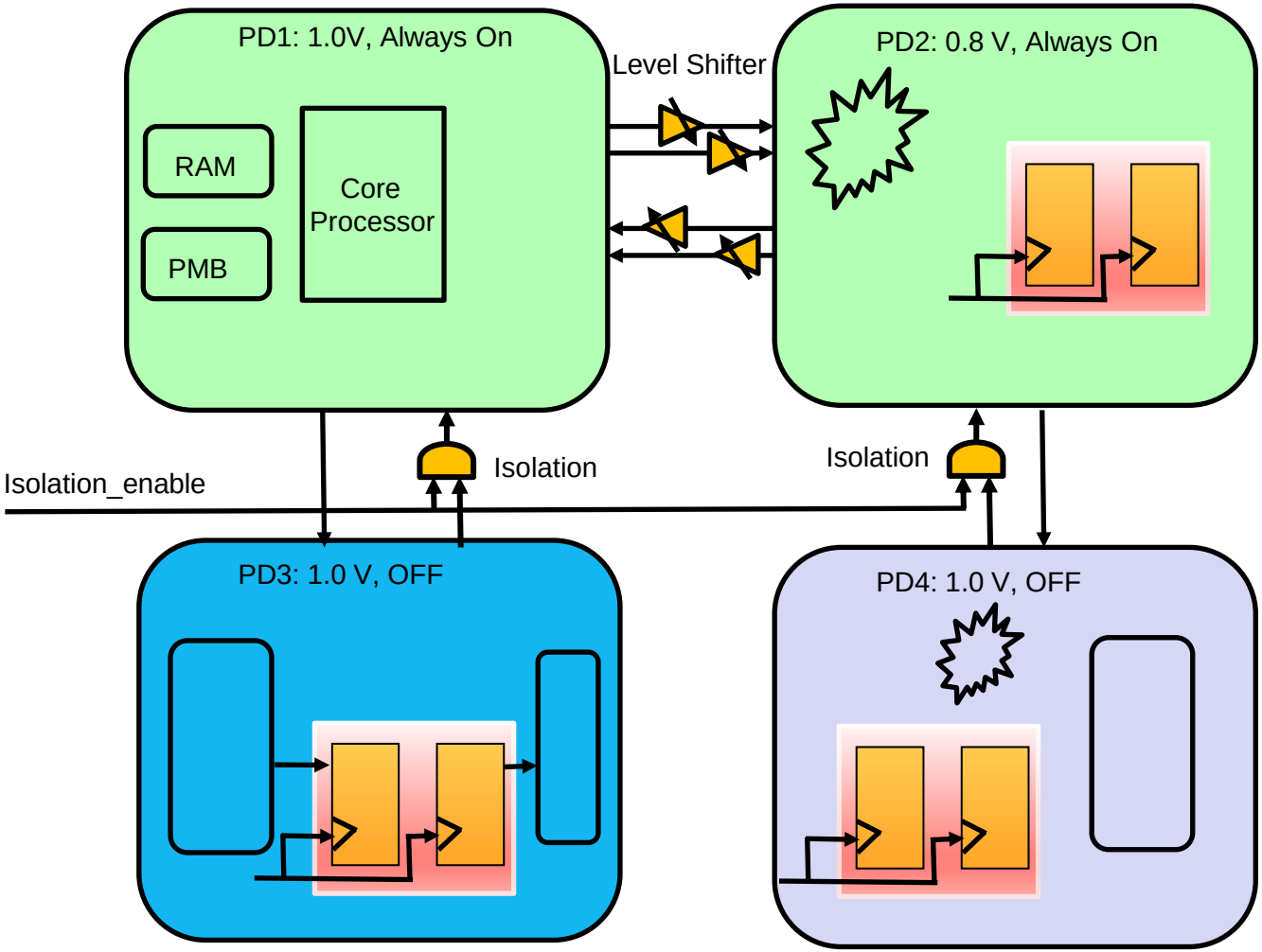


- Logic Restructuring



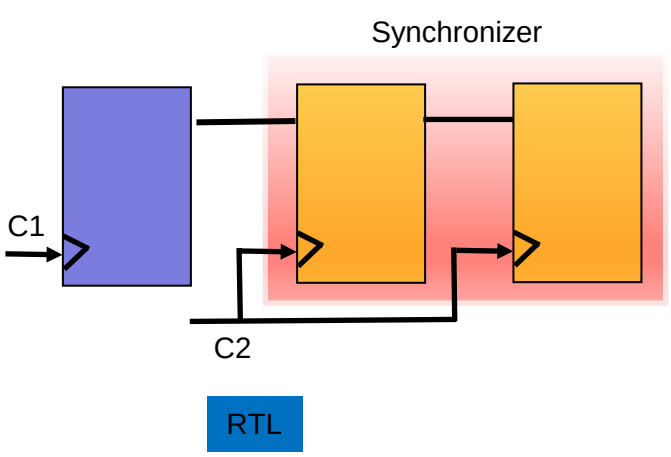


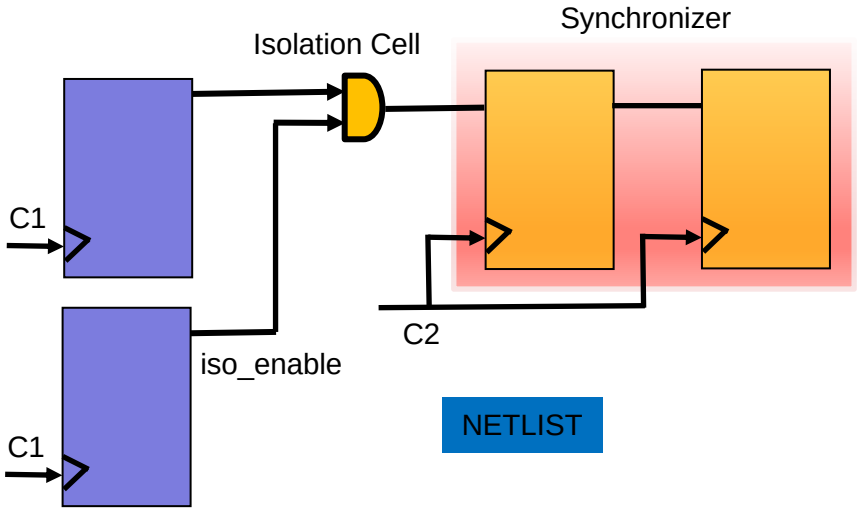
Low Power Design Example



Low Power Impact on CDC path

- Isolation affecting synchronizer



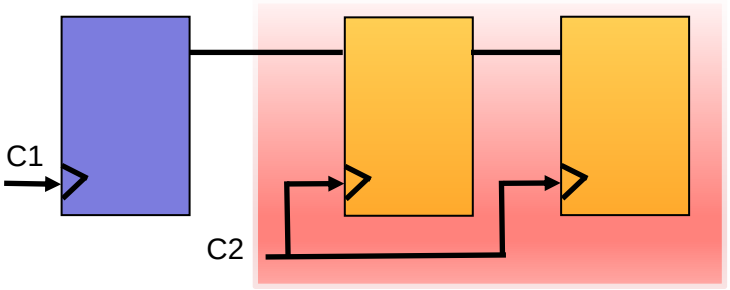


Solution

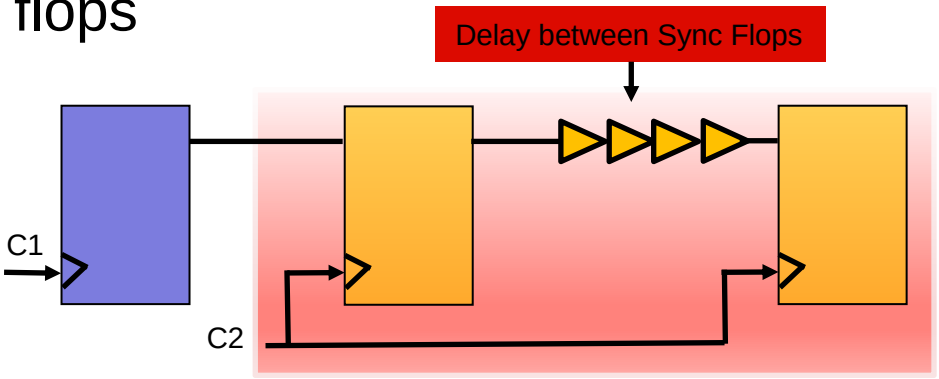
- Have better control during synthesis on CDC Path.
- One way is to have a CDC Design Library
  - Instantiate the RTL in the Design
  - Having equivalent functionality in netlist cell.
- Based upon CDC paths understanding, generate constraints for better design optimization/Physical implementation.

Physical Implementation Impact

- Huge delay in between Sync flops

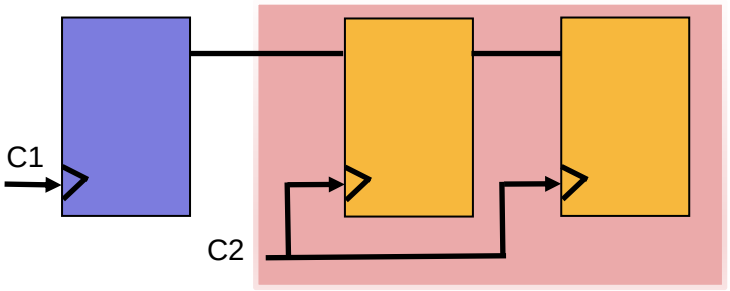


Logical View



Unconstrained Physical Implementation

Hard Macro from Special Clock Library

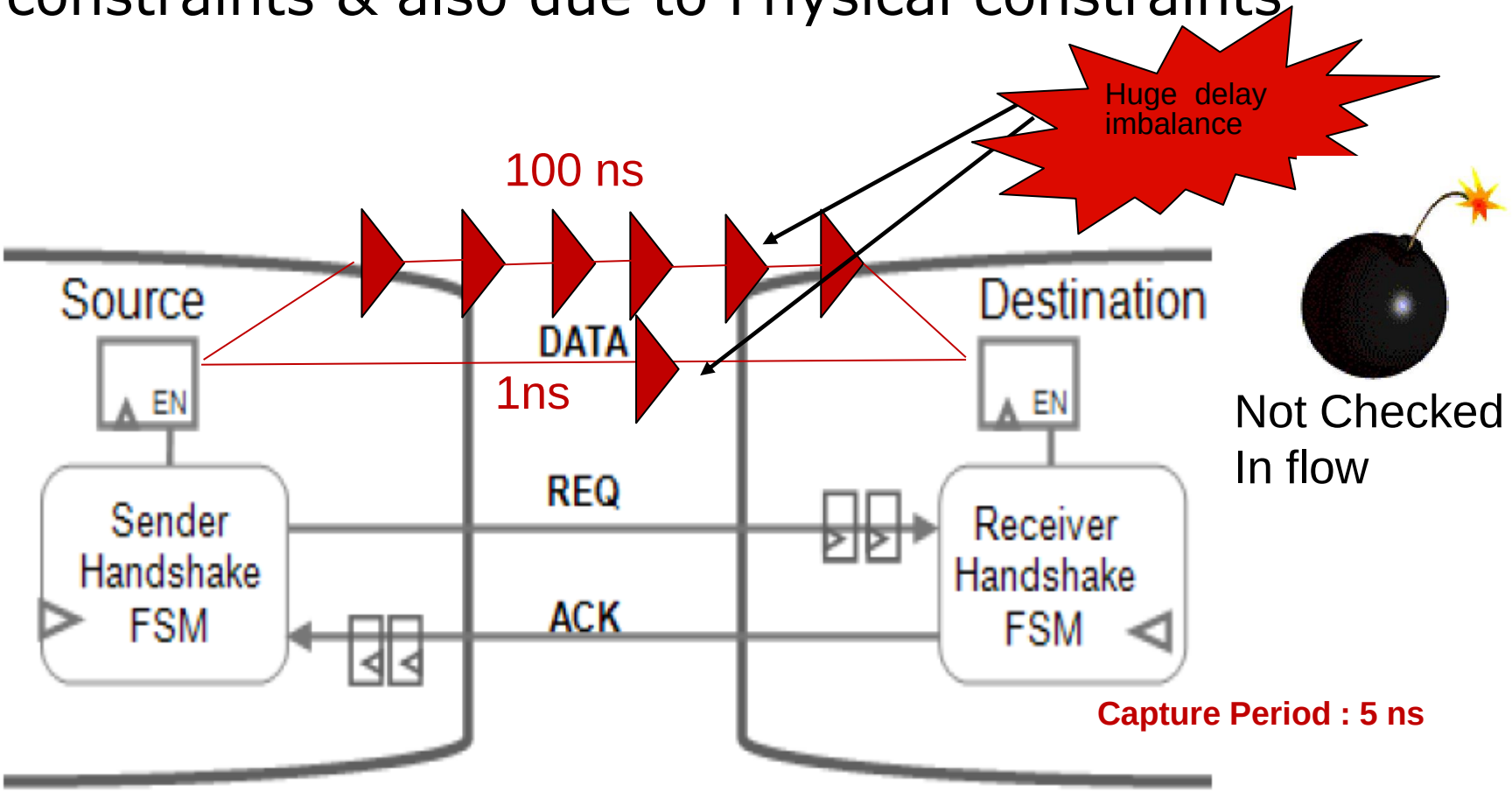


Logical View

Reference: https://www.slideshare.net/rameraja/4-u-5-slides-with-notes-8425918

Skewed data path

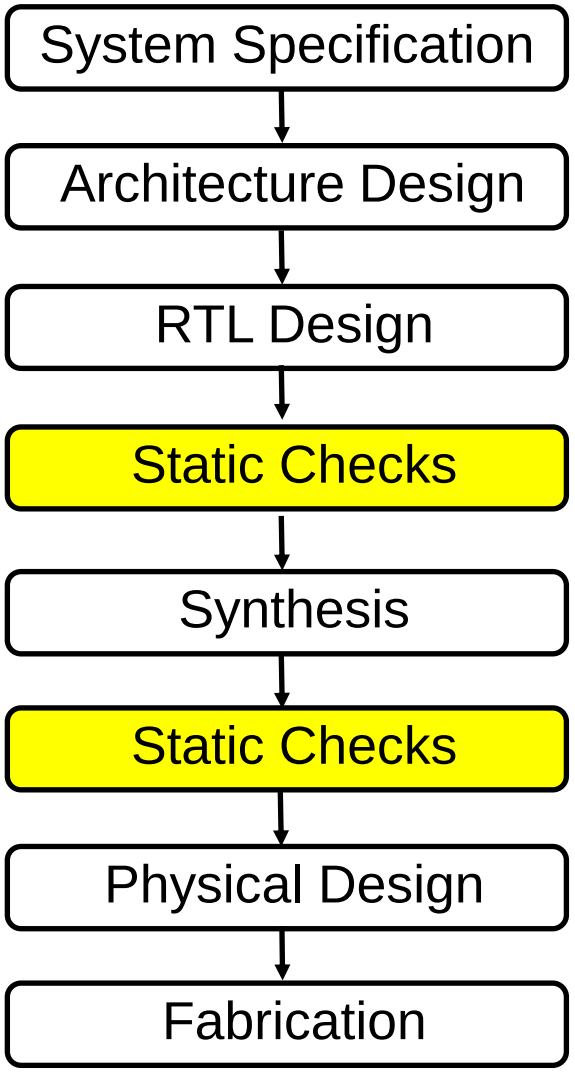
- Data path is severely skewed due to No constraints & also due to Physical constraints



Need to constrain data bus even though transfer is on an Asynchronous interface, using set max delay.

Design Flow and CDC Checking tools

- CDC analysis should be done at RTL, also at netlist level.
- Reuse the RTL level constraints during netlist CDC analysis.
- EDA Tools for CDC analysis
  - Questa CDC - Mentor Graphics
  - Meridian CDC – RealIntent
  - VC SpyGlass CDC – Synopsys
  - Jasper Gold – Cadence
  - CDC – Blueperalsoftware
  - ALINT-PRO – ALDEC



Conclusion

- Clock Domain Crossing require special handling both at logical and implementation time.
- The cells in the CDC path must be chosen from special cell library.
- Use constraints for CDC friendly design optimization.

Link for other presentations

- HDL Design using Verilog: https://www.linkedin.com/feed/update/urn:li:activity:6901101173491798016
- HDL Design Guidelines:
  - https://www.linkedin.com/feed/update/urn:li:activity:6903289386536968192
- VLSI Design Flows and Open source tools:
  - https://www.linkedin.com/feed/update/urn:li:activity:6886886690405924864
- https://www.sites.google.com/view/learnvlsi/webinar

Thank you

Telegram Channel: https://t.me/elearnvlsi

Next webinars:

For more updates, follow Learn VLSI LinkedIn Page :
https://www.linkedin.com/company/learnvlsi

Feedback/Errata: Please send email to elearnvlsi@gmail.com

After each improvement, the updated slides will be available at website:
https://www.sites.google.com/view/learnvlsi/webinar