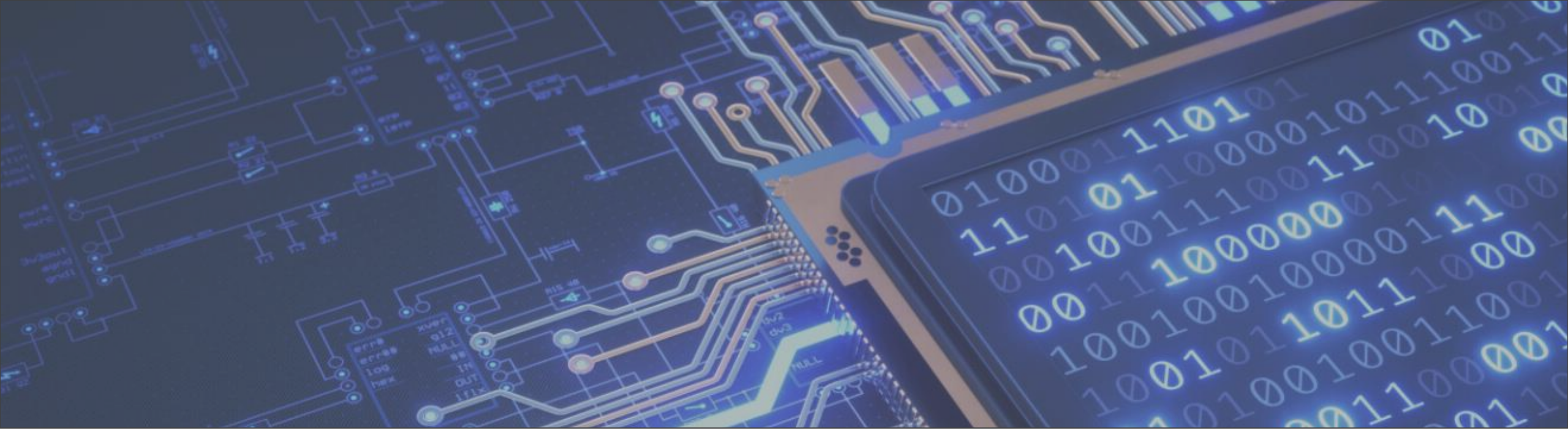


IC設計簡介 - Clock Domain Crossing(CDC)

Introduction to Jasper® CDC

Speaker: Lai, Lin-Hung

Nov. 07, 2023

cadence® 國立陽明交通大學 NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Si2 Lab System Integration Silicon Implementation

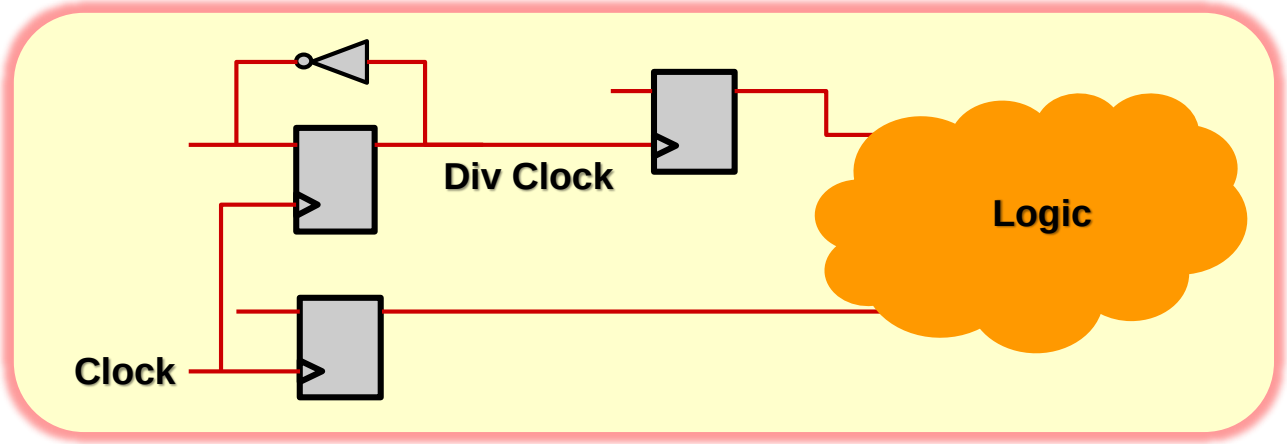
Outline

- Clock Domain Crossing (CDC) Overview
  - Metastability & Clock Domain Crossing
  - Single-Bit – N-FF Synchronizer / Pulse Synchronizer / Edge Synchronizer
  - Multi-Bit – Convergence
  - Multi-Bit – NDFF_BUS with Gray Code
  - Multi-Bit – MUX_NDFF / MUX_PULSE Synchronizer
  - Multi-Bit – Handshake / FIFO Synchronizer
  - Single-Bit – Divergence
  - Common Synchronization Schemes
- Introduction to Jasper® CDC app
  - Structural Analysis
  - Functional CDC Analysis

Clock Domain Crossing (CDC) Overview

Clock Domains

- A clock domain is defined as that part of the design driven by either a single clock or clocks that have constant phase relationships

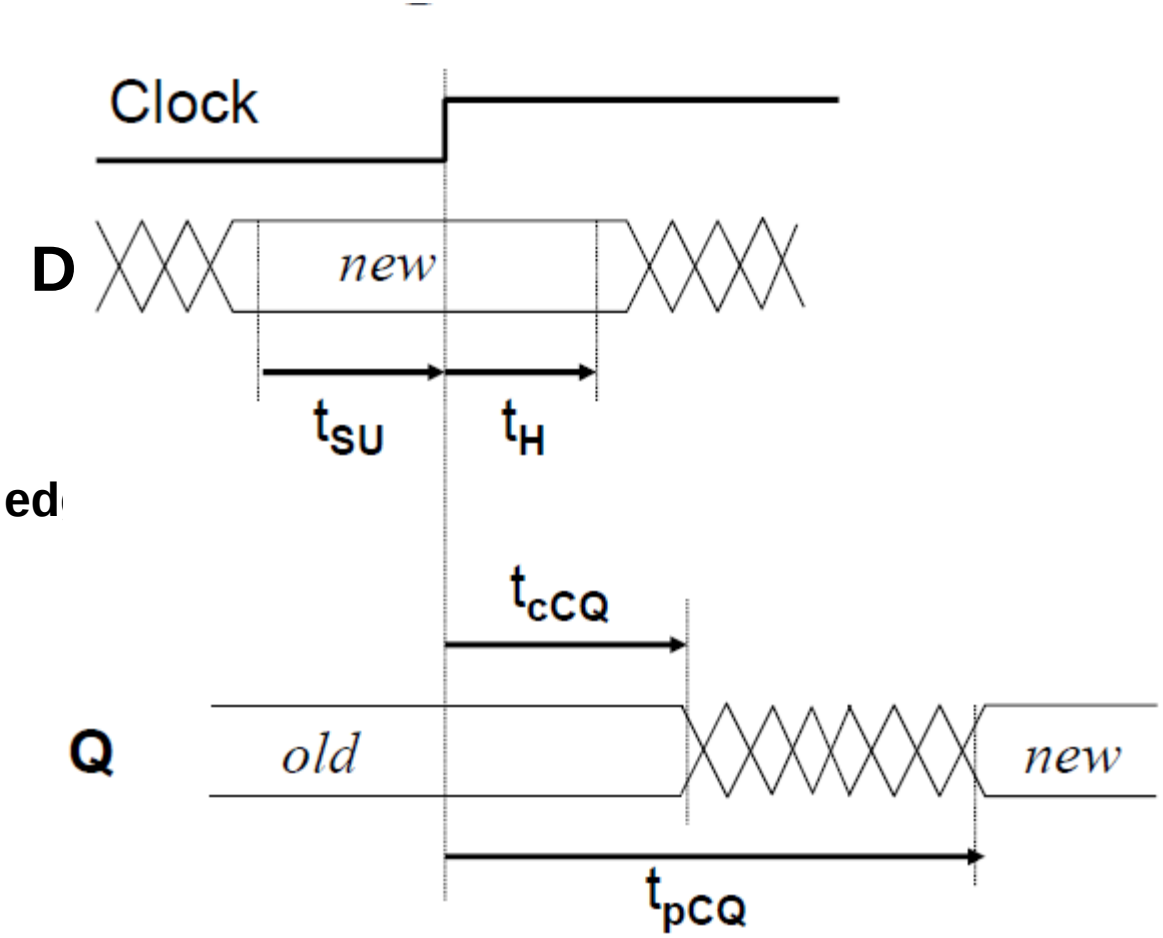


- Synchronous and Asynchronous clocks
  - Synchronous: Constant phase relationships
  - Asynchronous: Variable phase and time relationships

Timing Violation

FF Timing Specification

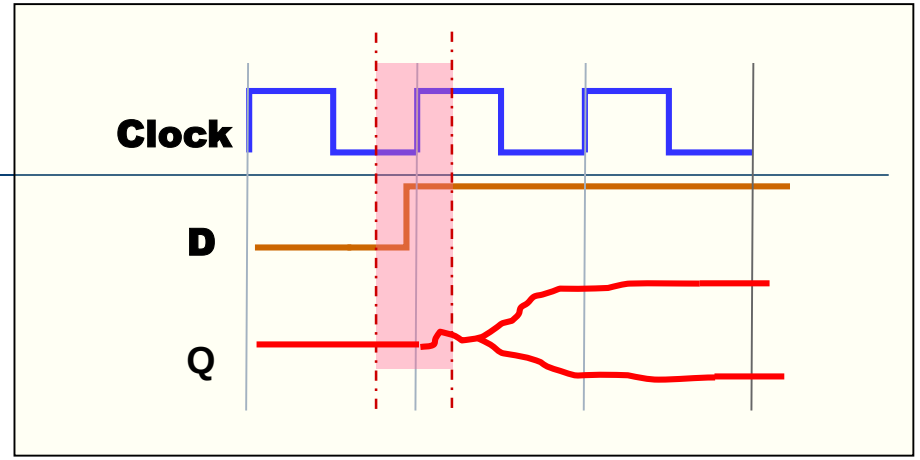
- **Input data (D) must be stable**
  - Before active edge - Setup time (t_{SU})
  - After active clock edge – Hold time (t_H)
- **Propagation delay is the time from clock ed**

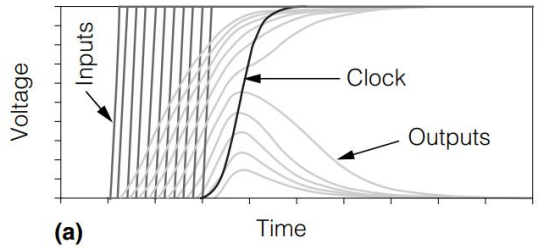


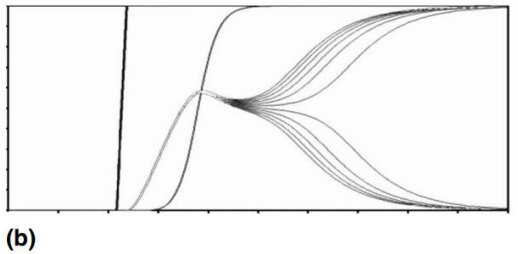
What happens when this specification is not followed?

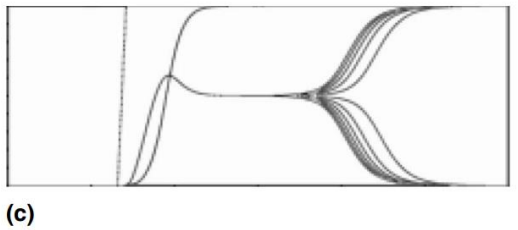
What is Metastability?

- When setup/hold conditions are violated, the output of a flip-flop becomes unstable and unpredictable
- Output may settle either way (1 or 0) after an unpredictable delay
- This phenomenon is known as metastability
- Seen for asynchronous inputs (clock to data relationship is unknown)







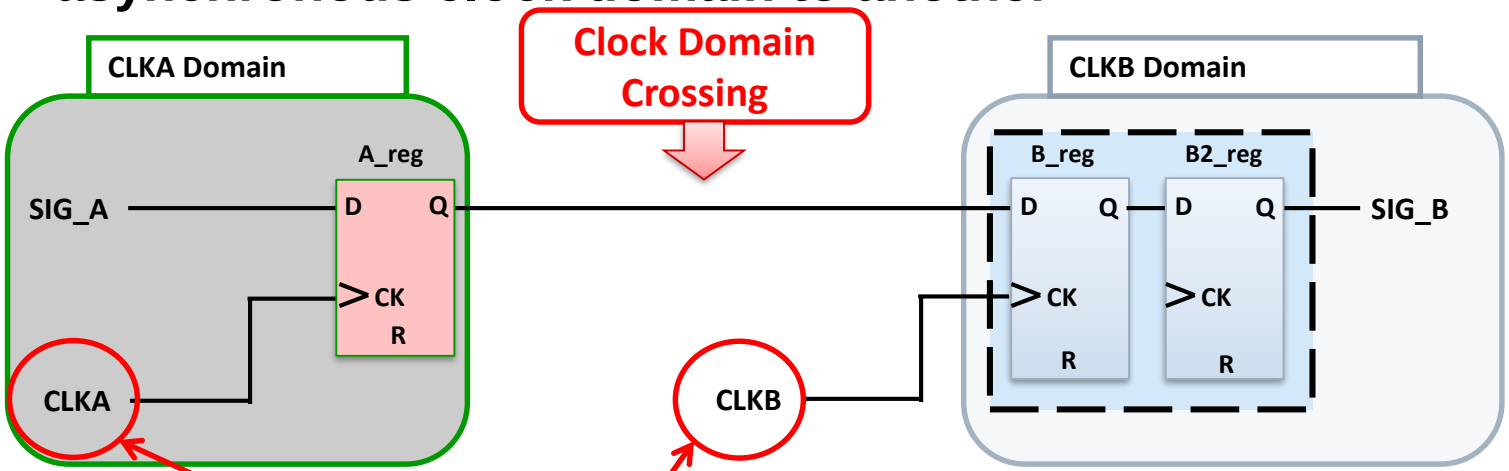


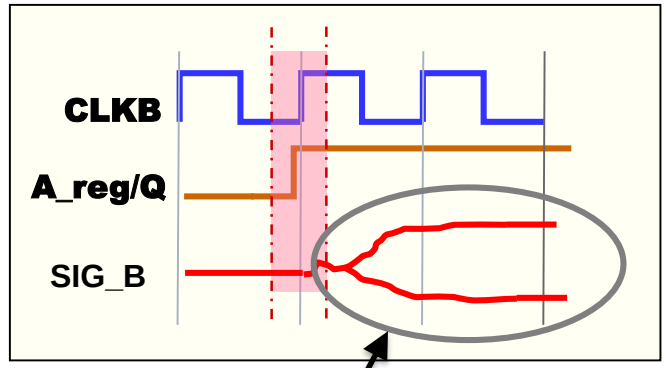
The input edge is moved in steps of 100 ps, 1 ps, and 0.1 fs in the top, middle, and bottom charts respectively.

R. Ginosar, "Metastability and Synchronizers: A Tutorial," in IEEE Design & Test of Computers, vol. 28, no. 5, pp. 23-35, Sept.-Oct. 2011, doi: 10.1109/MDT.2011.113.

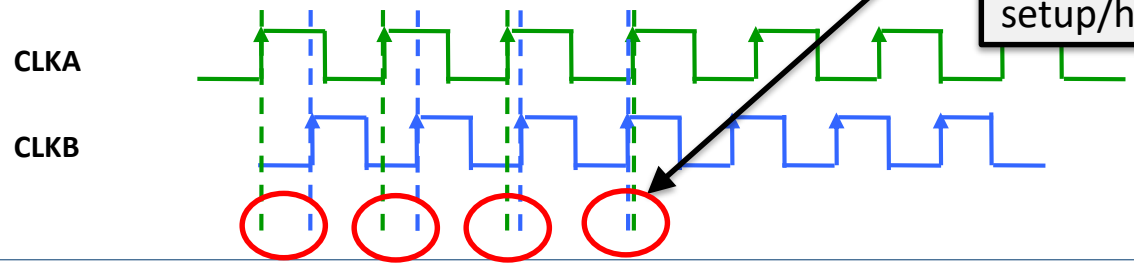
Clock Domain Crossing Overview

- Clock Domain Crossing (CDC) occurs when a signal crosses from one asynchronous clock domain to another





if CLKA & CLKB are asynchronous



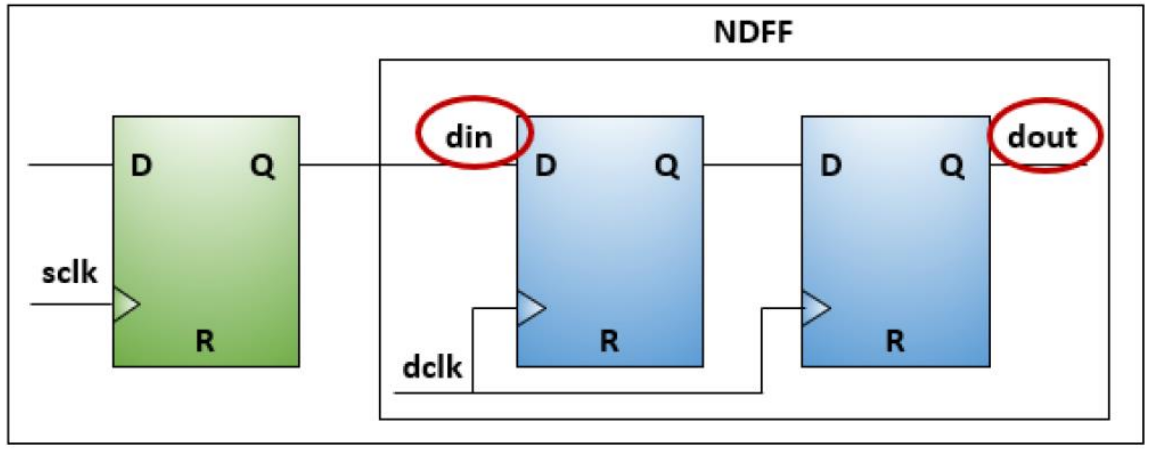
Non-deterministic relationship between the clocks causes them to skew continuously resulting in setup/hold violations

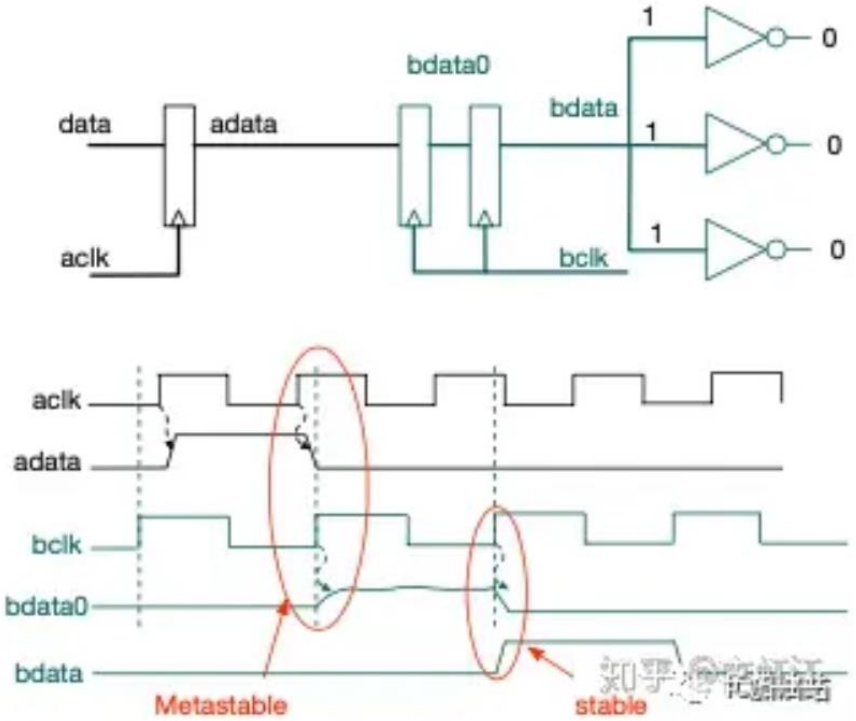
Flop output goes to metastable state which eventually settles to a 1 or 0 after an unpredictable delay

Core Idea – Double/N Flop(2-FF/N-FF) Synchronizer

NDFF

Formal Signals	`data, dout`
Associated Command	`check_cdc -scheme -add ndff -module ndff -map {{dout dout} {data din}}`



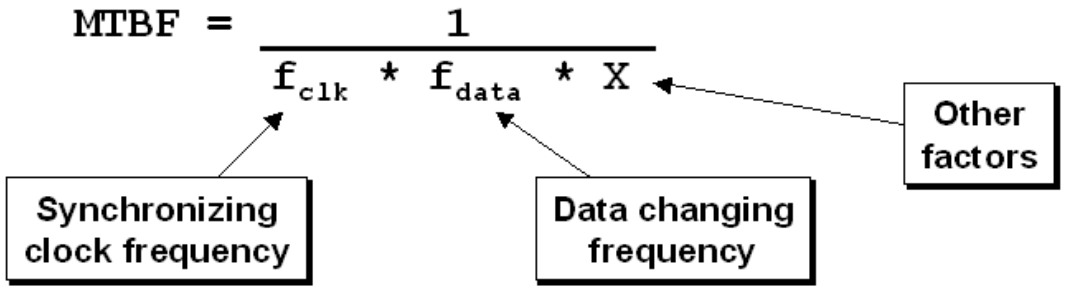


(Source: https://www.zhihu.com/people/li-hong-jiang-54/posts)

Mean Time Between Failure

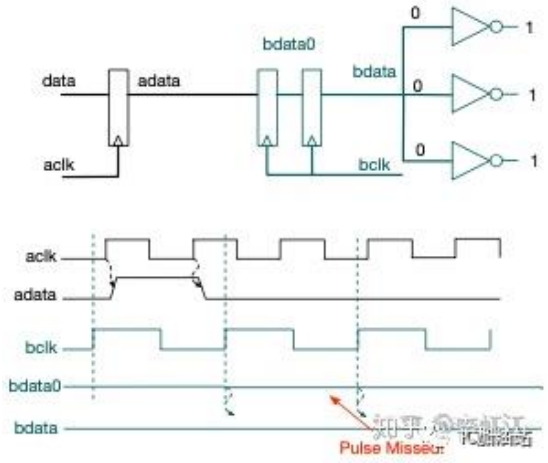
- Metastability cannot be eliminated. It's effect can be minimized and reduce the probability of failure

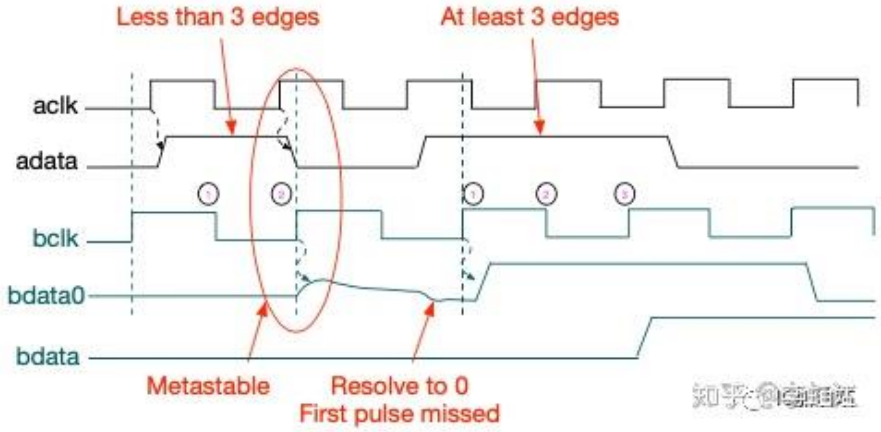
- The calculation of the probability of the time between synchronization failures (MTBF) is a function of multiple variables including the clock frequencies used to generate the input signal and to clock the synchronizing flip-flop



Three Edge Rule

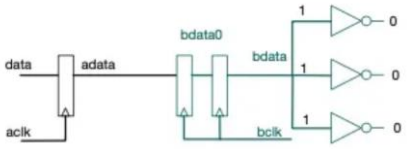
- Fundamental requirement: **The "Three Edge" rule**
  - **'Source data'** must remain stable for at least three consecutive edges of the **'destination clock'**
  - First change: **'adata'** does not meet the requirement, leading to a missed pulse in **'bdata'**
  - Second change: **'adata'** meets the requirement, ensuring **'bdata'** captures the change
- Frequency Relationship and Data Stability
  - **‘bclk’** frequency is at least 1.5 times that of **‘aclk’** → **‘adata’** could be sync even a short pulse
  - If this frequency condition is not met, **'adata'** must remain stable for longer periods





Timing Variability in Double Flop Synchronization

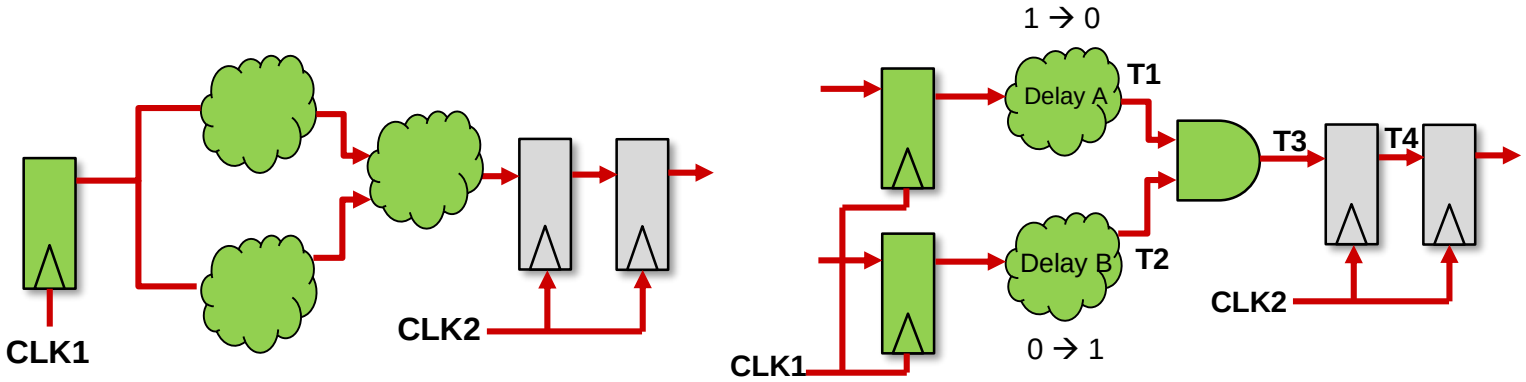
- Double flop synchronization can exhibit variability in the cycle when **'bdata'** reflects changes in **'adata'.**
- This variation can be either after 1 cycle or 2 cycles due to metastability resolution of the first flip-flop.
- If the first flop resolves to the same value as 'adata', 'bdata' change is seen after 1 cycle.
- If the first flop resolves to the opposite value of 'adata', an additional cycle is required for 'bdata' to change.
- In design and simulation, it is crucial not to assume 'bdata' will always change after 2 cycles. Variability must be accounted for in simulation to reveal potential design issues
- Strategies for situations where the timing of the next pulse is unpredictable

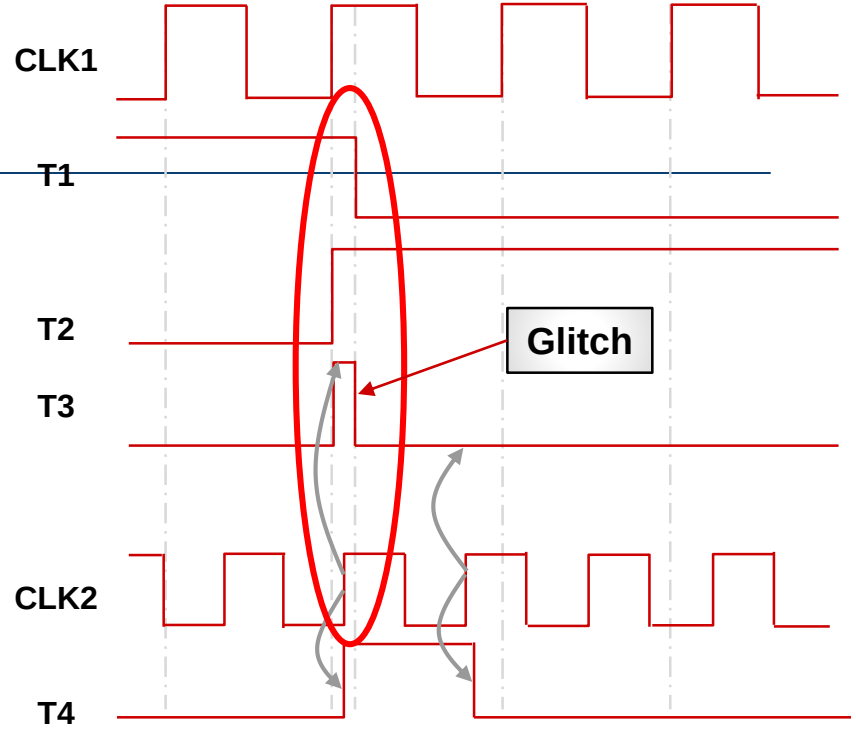


Signals from Combinational Logic

- Questions arise regarding the synchronization of single-bit outputs from combinational logic, such as an AND gate.
- It is not recommended to use double flop synchronization directly on these outputs. Here's why:
  - Combinational logic may produce glitches that can increase the probability of the first flop entering metastability, thereby affecting the Mean Time Between Failures (MTBF) of the synchronizer.
  - It is imperative to first register (flop) any single-bit signal before it crosses clock domains.
- The diagram below illustrates the pitfalls of not registering the signal before synchronization.

Glitch on CDC Path

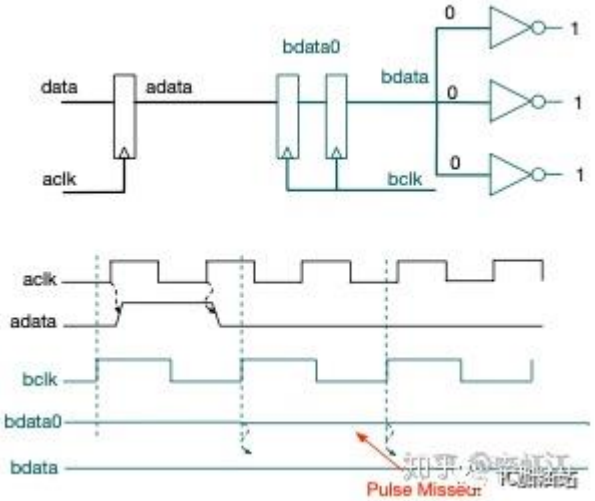




- Any logic in the crossover path can cause glitches and create functional errors downstream
- Different delays in paths T1 and T2 causes a glitch
- The glitch causes the flop output to go high when it was supposed to remain low
- Only the output of a flop should pass through a synchronizer

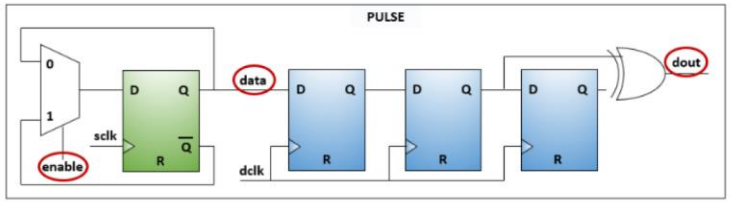
Pulse Synchronizer (1/2)

- Problems with pulses being missed or not meeting the synchronization requirements with a 2-flop synchronizer
- Fundamental Questions
  - What an **aclk** domain pulse would be in the **bclk** domain?
  - Why do we need to synchronize the pulse from **aclk** to **bclk**?
- Real-world applications of pulse signals:
  - Counters, memory read/write operations, FIFO operations



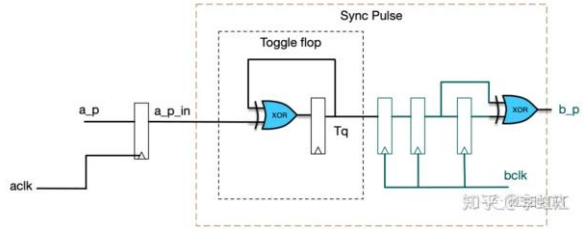
PULSE

Formal Signals	`data, dout, enable`
Associated Command	`check_cdc -scheme -add pulse -module PULSE -map {{data data} {dout dout} {enable enable}}`

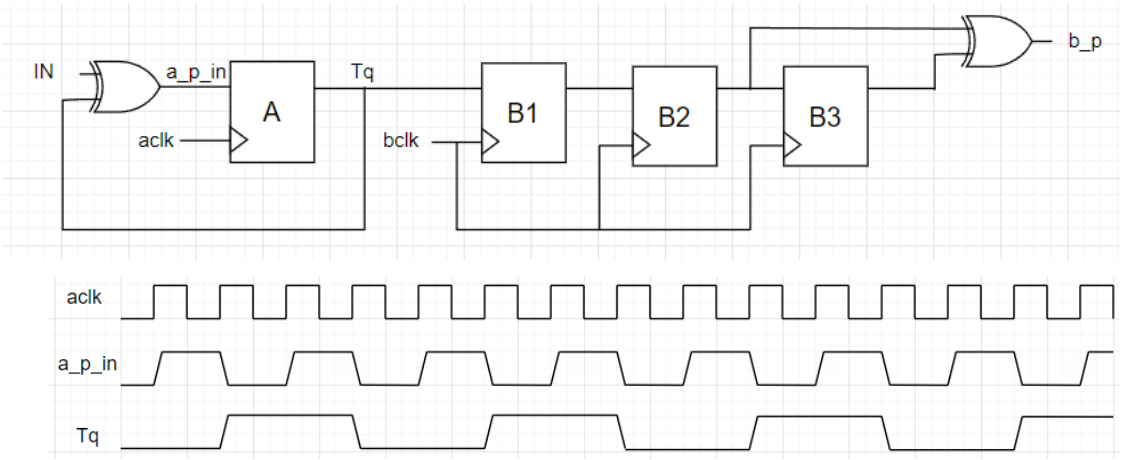


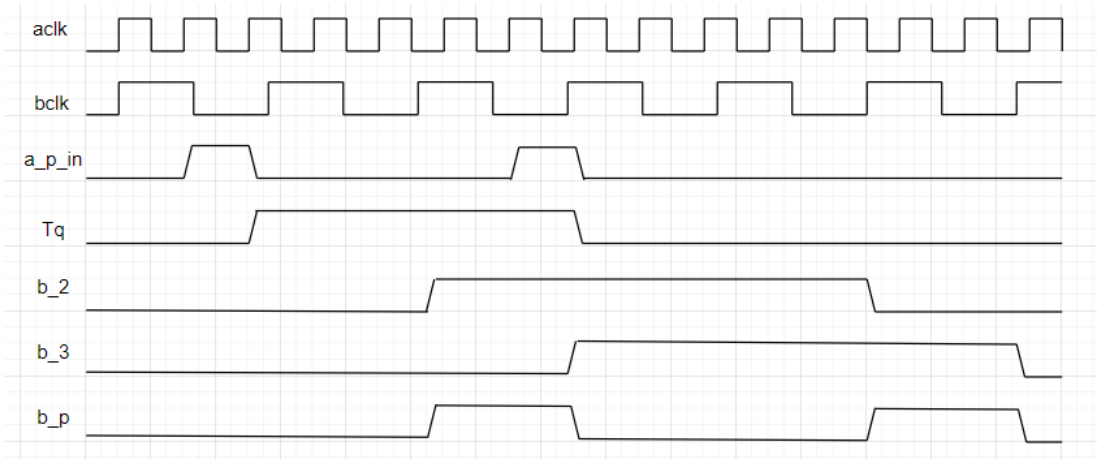
Pulse Synchronizer (2/2)

Synchronization Strategy



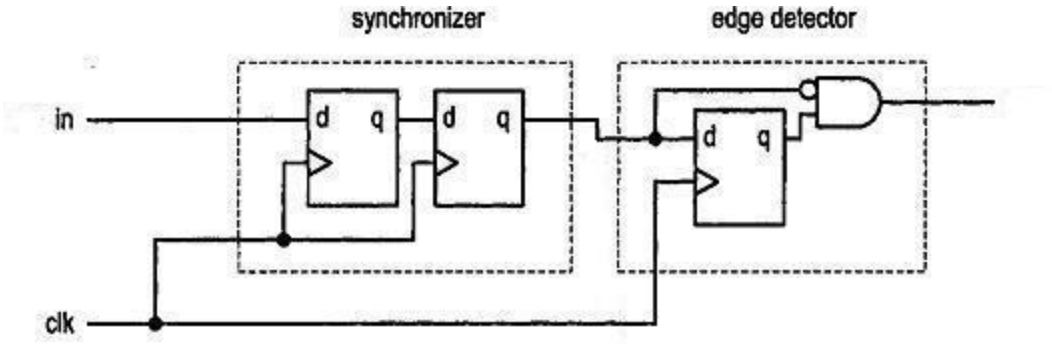
- Converting **aclk** pulse to a level signal.
- Using a 2-flop synchronizer to sync the level signal to the **bclk** domain.
- Transforming the synced level signal back to a pulse in the **bclk** domain.
- Interval Constraints:
  - Pulses from '**a_p_in**' must not occur too close together to allow for successful synchronization
  - the fastest allowable pulse generation from '**a_p_in**' is after one clock cycle, setting '**a_p_in**' at half the frequency of 'aclk' and Tq at a quarter frequency of 'aclk'.





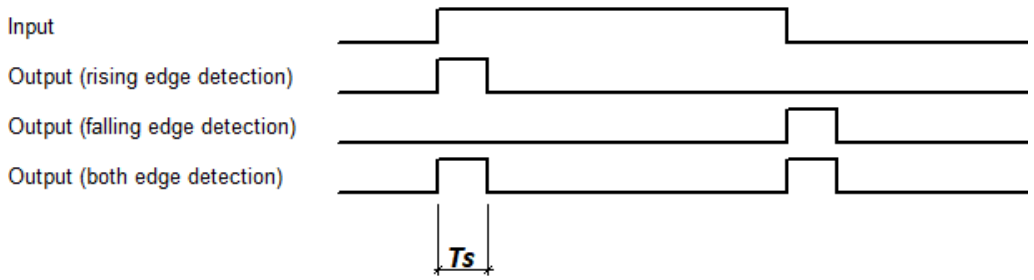
https://hackmd.io/@derek8955/H1jlRjbZi

Edge Detect Synchronizer



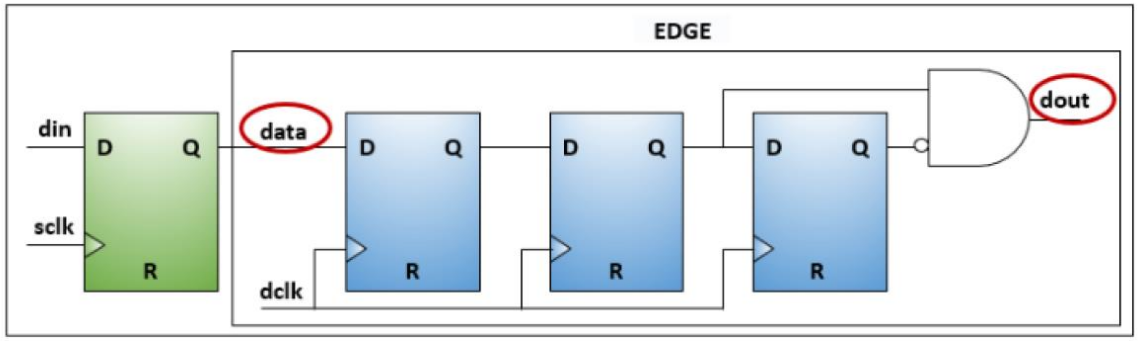
Type	Application	Input	Output	Restriction
Level	Synchronizes level signals	Level	Level	Input must be valid for at least two clock periods in the new domain. Each time output goes valid, counts as a single event.
Edge-detect	Detects rising or falling edge of input	Level or Pulse	Pulse	Input must be valid for at least two clock periods in the new domain.
Pulse	Synchronizes single clock-wide pulses	Pulse	Pulse	Input pulses must be spaced by at least two clock periods in the new domain.

Table 1-1: Synchronizer Summary



EDGE

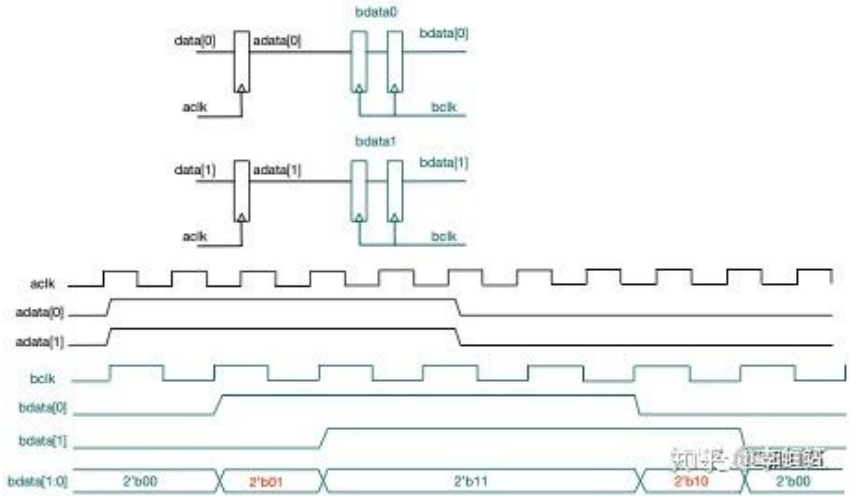
Formal Signals	`data, dout, activity (high | low), sensitivity (rising | falling)`
Associated Command	`check_cdc -scheme -add edge -module EDGE -map {{data data} {dout dout} {activity high} {sensitivity rising}}`



Challenges of Multi-Bit Signal Synchronization

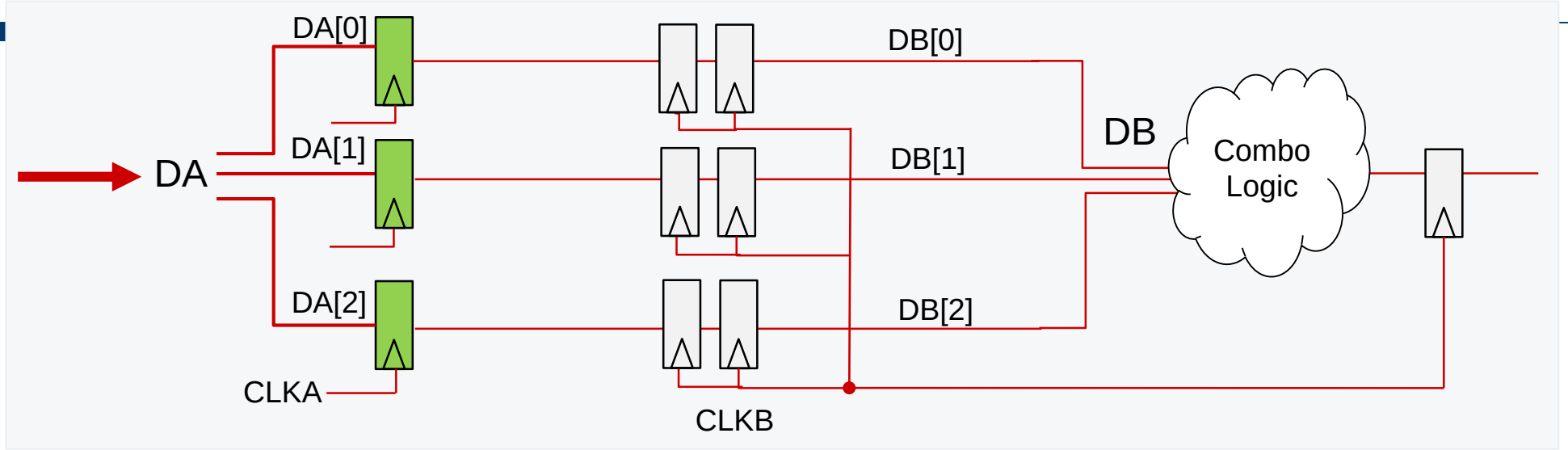
- **Key Conclusion:**
  - In most cases, 2-flop synchronizers are unsuitable for synchronizing multi-bit signals.

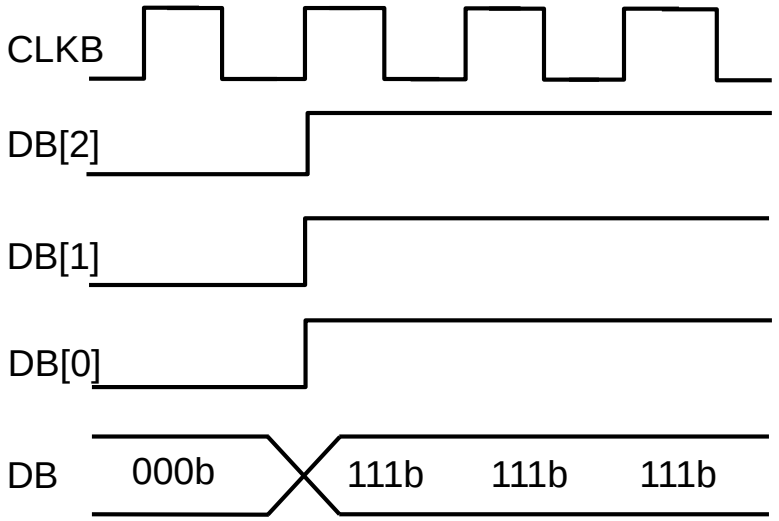
- **Understanding Multi-Bit Signals:**
  - Multi-bit signals are represented by more than one bit, such as counter values, addresses, or multi-bit data buses.
  - Inter-bit dependencies mean bits cannot function correctly in isolation.
  - Some grouped single-bit control signals can be synchronized individually; these are pseudo multi-bit signals.



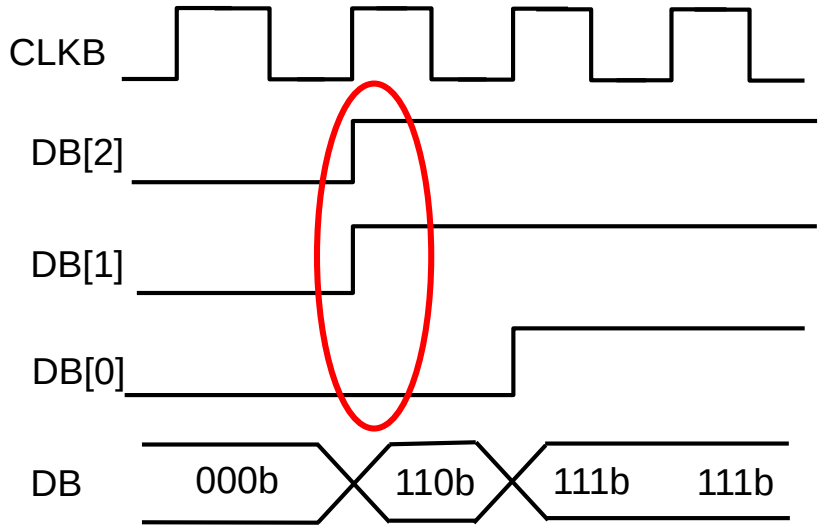
- **The Issue with 2-Flop Synchronizers:**
  - Random delays in synchronization can produce incorrect, transient values not present in the original signal
  - Convergence issue

Convergence (Same Source Signal)



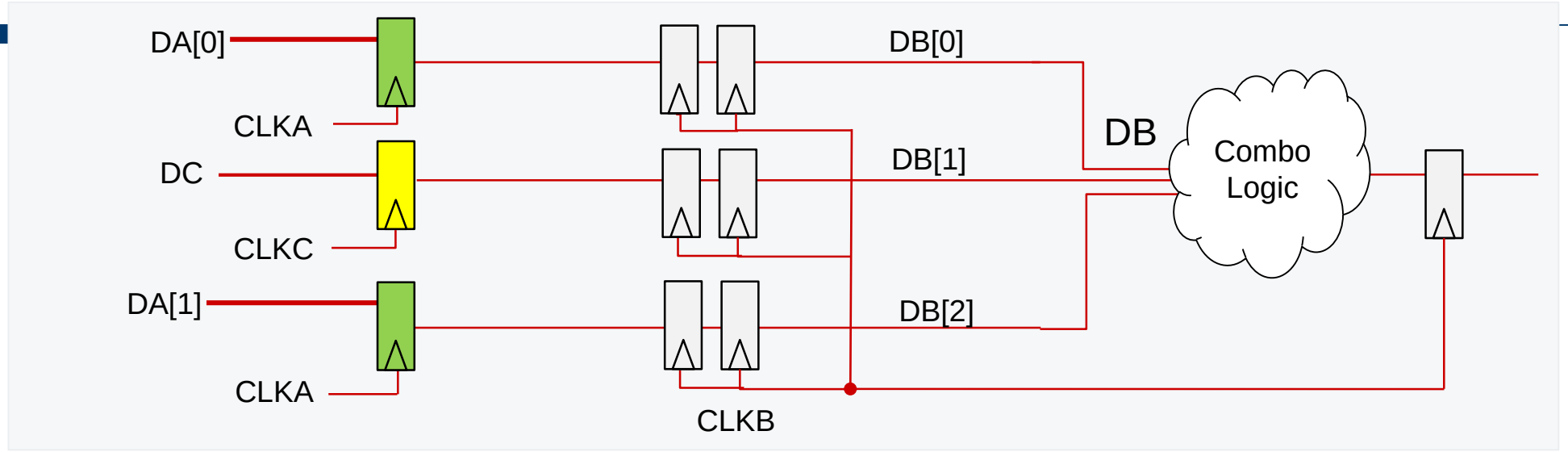


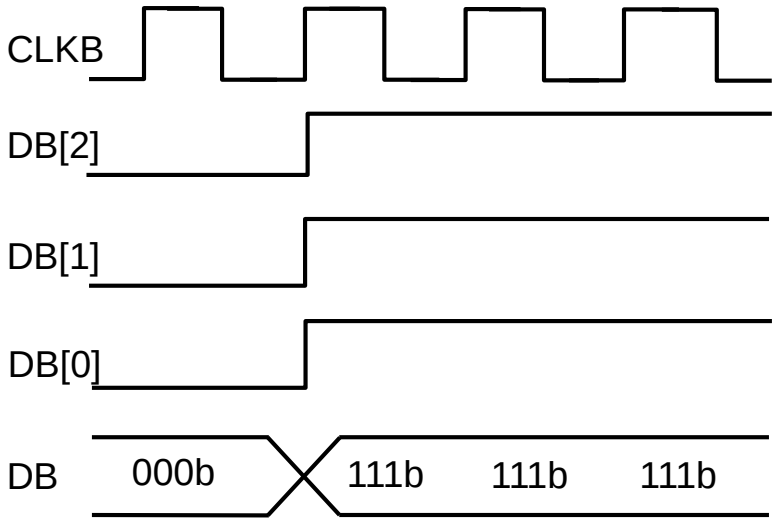
Expected Behavior



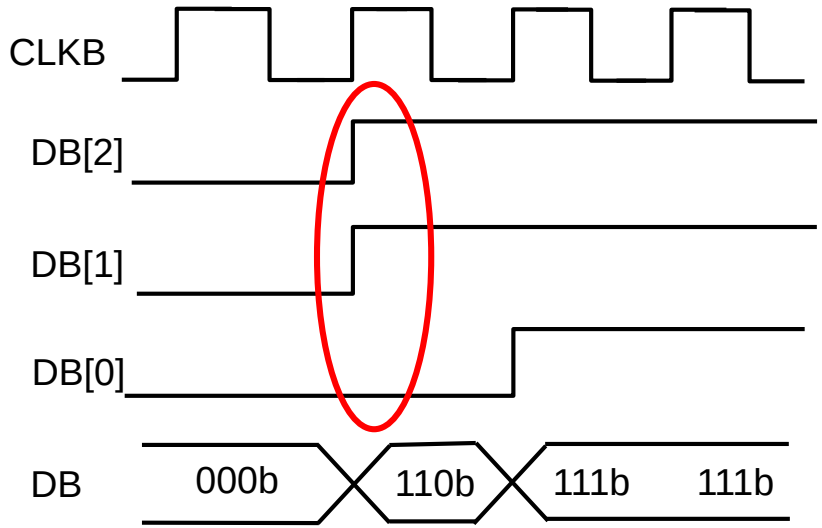
Effect of Re-convergence

Convergence (Different Source Signals)



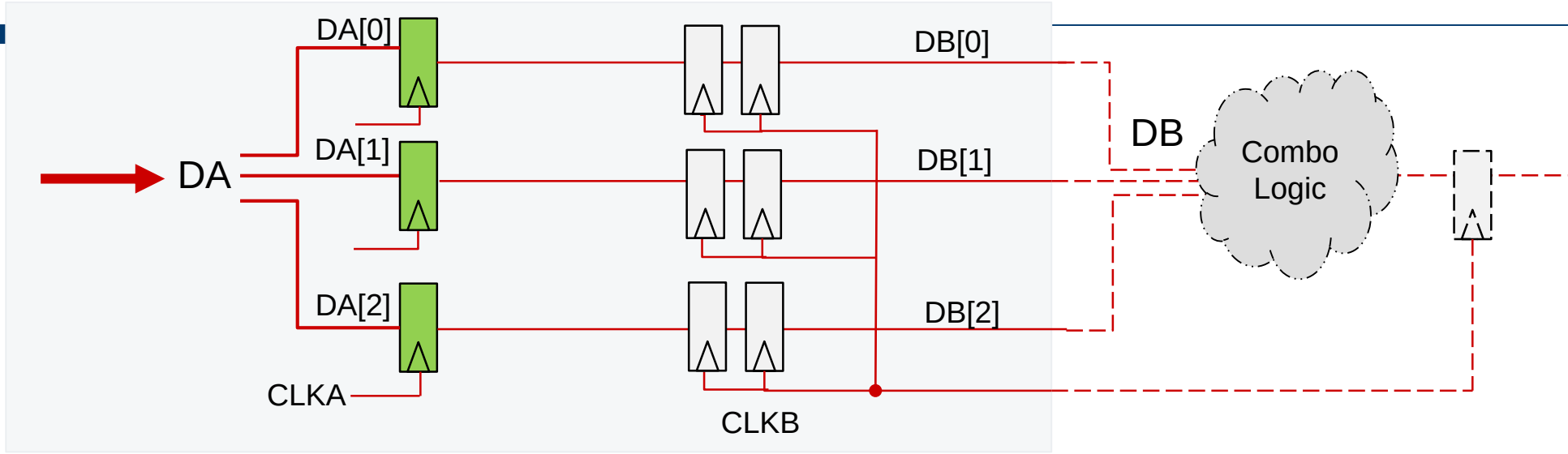


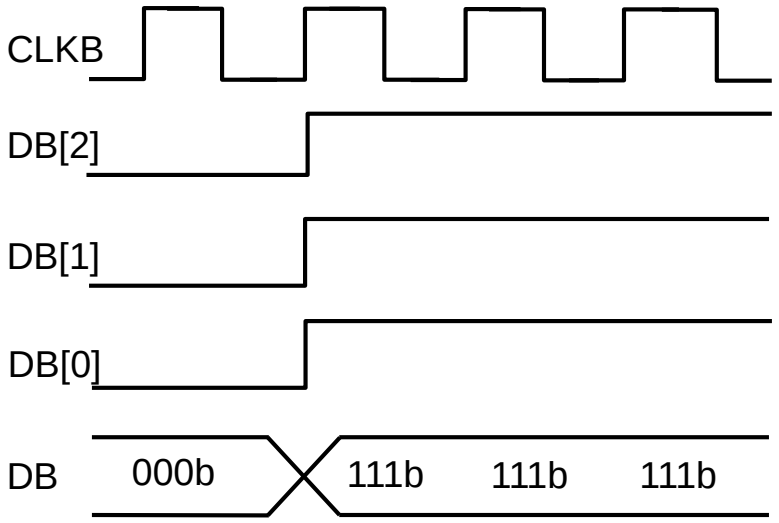
Expected Behavior



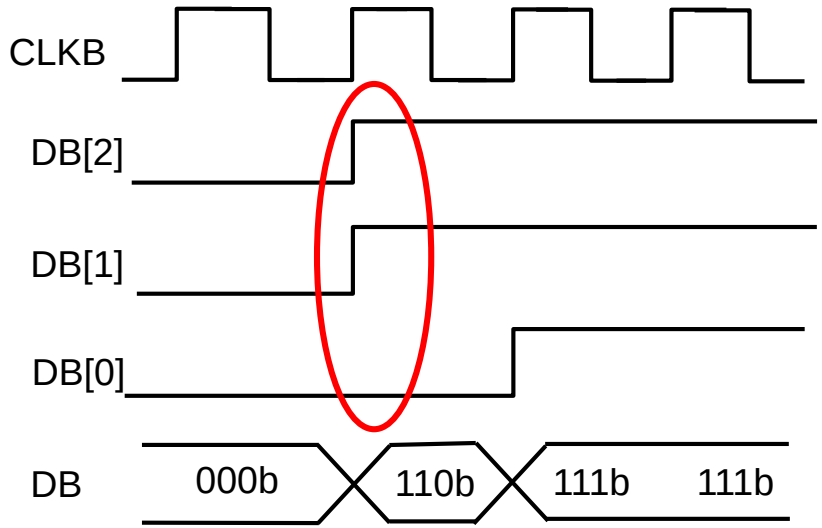
Effect of Re-convergence

Convergence (Outside the block)





Expected Behavior

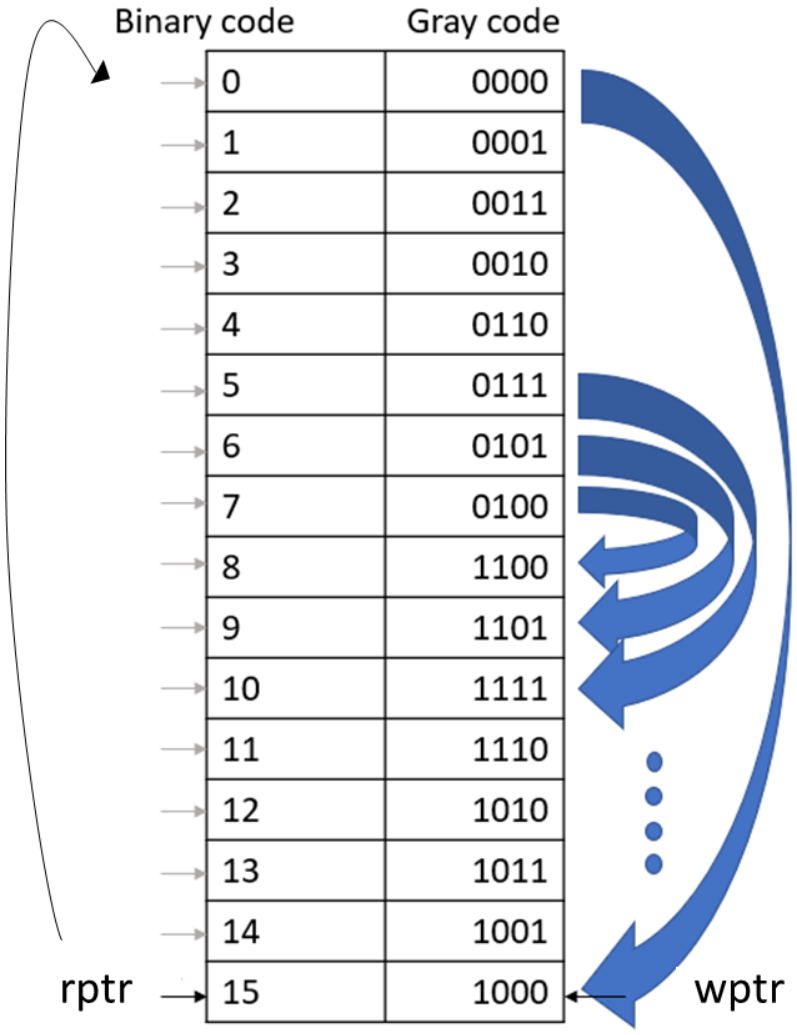


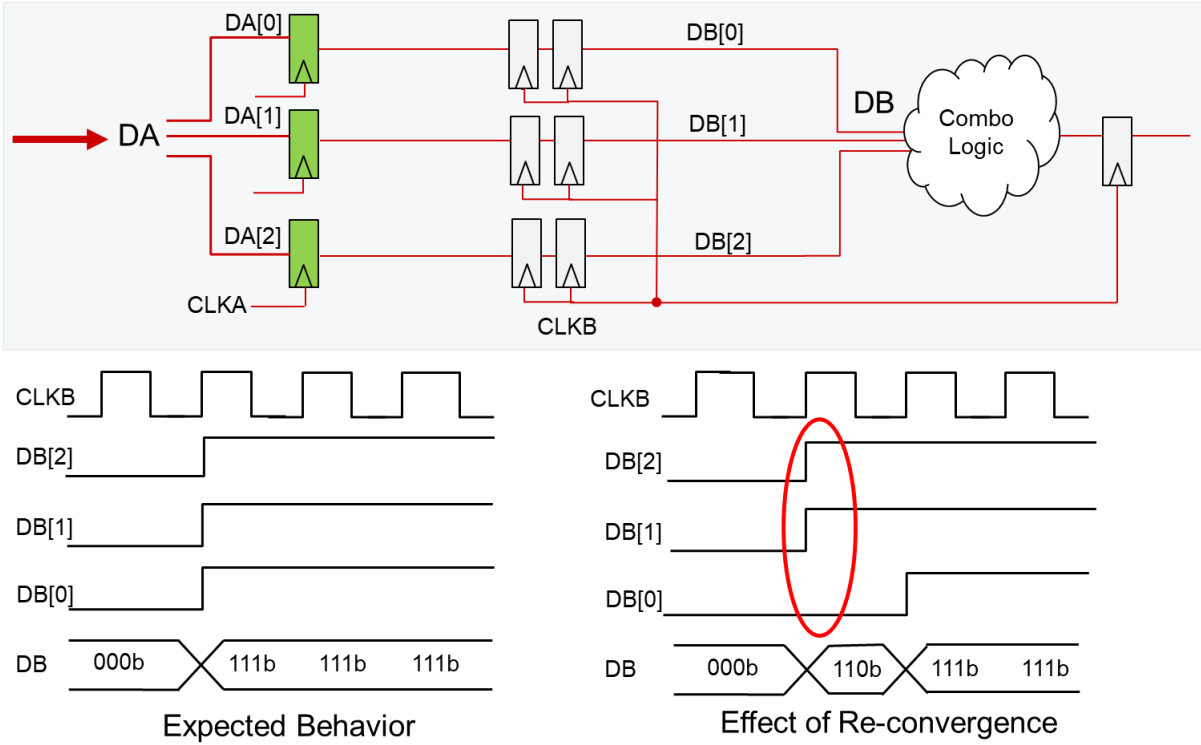
Effect of Re-convergence

Strategies for Multi-Bit Synchronization

- **NDFF_BUS with Gray Code Transfer**
  - Utilizes NDFF for synchronizing bus data.
  - Employs Gray coding to transfer multi-bit data efficiently, minimizing synchronization errors.
- **MUX_NDFF / MUX_PULSE for Data Paths**
  - Typically applied within data paths for selective data capture.
  - Open-loop system capturing signals when the Mux enable signal is active.
  - Relies on a synchronized control signal to serve as the Mux select line, ensuring data capture timing.
- **Handshake Protocol**
  - A simple, effective request-acknowledge mechanism between source and destination modules.
  - Closed-loop synchronization requiring confirmation of the signal's successful transfer across Clock Domain Crossing (CDC).
  - Tends to introduce higher latency due to the acknowledgment process.
- **Asynchronous FIFO for Synchronization**

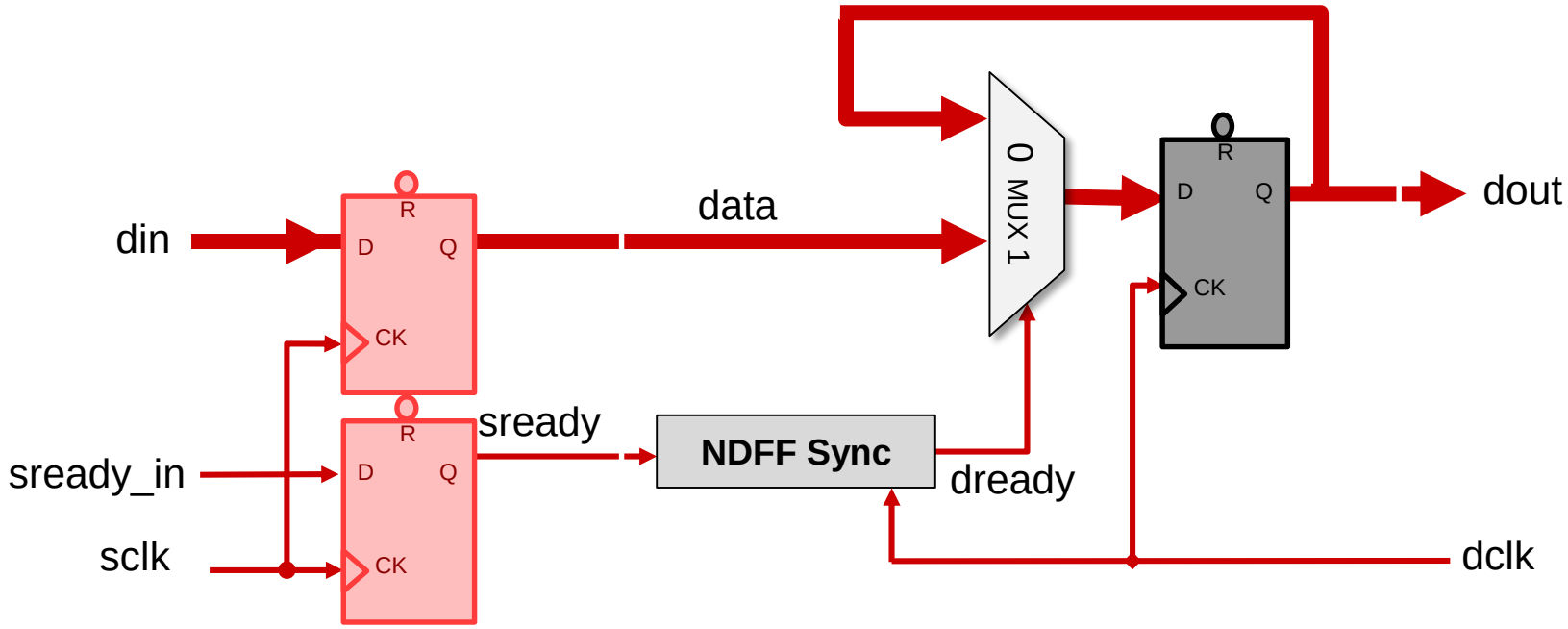
NDFF_BUS with Gray Code Transfer





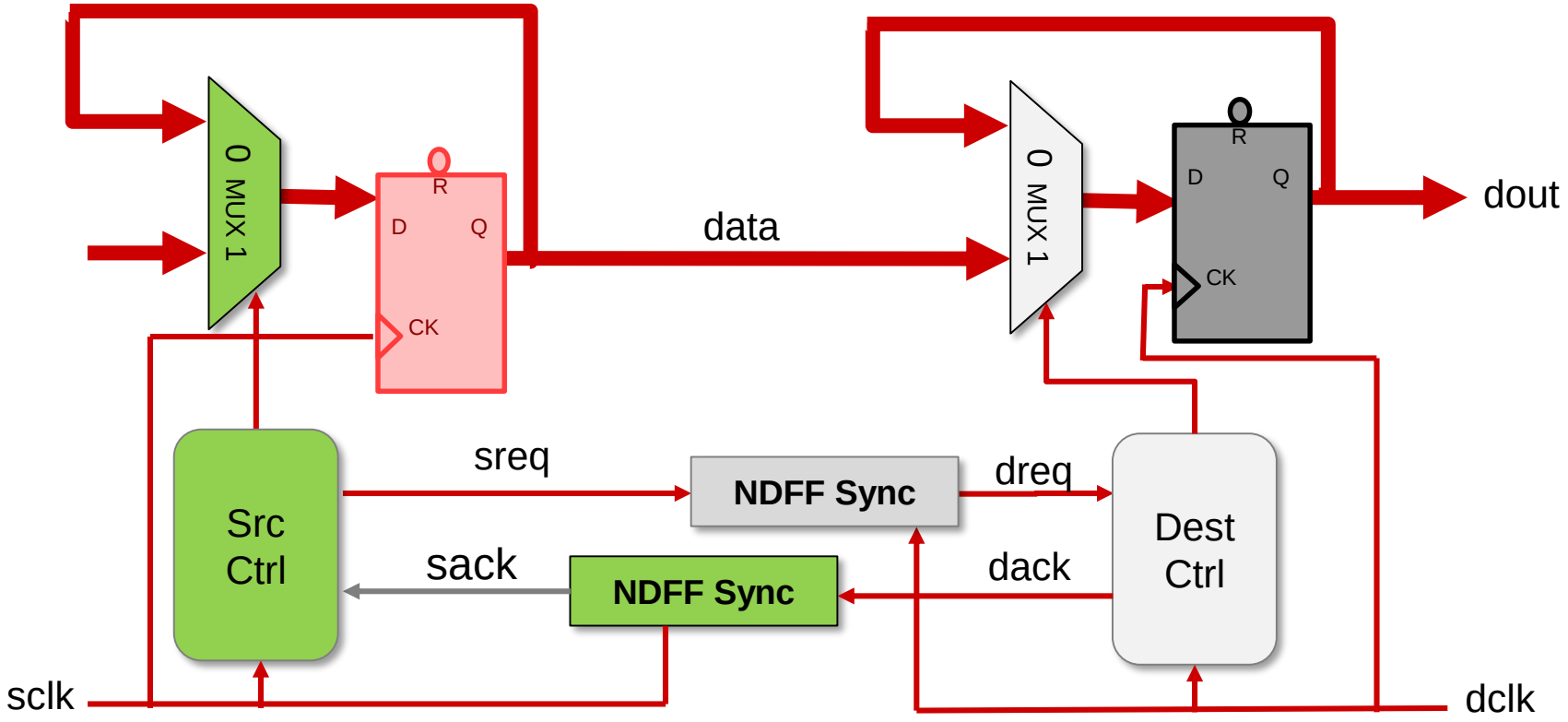
MUX Synchronizer

- Typically used for data paths
- Open-loop solution ensures signals are captured without acknowledgment
- Captures data when mux enable is active
- Uses synchronized control signal as select line of Mux



Handshake Synchronizers

- Source and destination modules use a simple request-acknowledge protocol
- Closed loop synchronization that requires acknowledgment of receipt of the signal that crosses the CDC boundary
- High latency



FIFO Synchronizers (1/3)

- Data is written into the FIFO from the source clock domain and read from the FIFO in the destination clock domain.
  - Gray-coded read and write pointers are passed into the alternate clock domain to generate full and empty status flags.
  - Used for data transfers from faster to slower clock domains

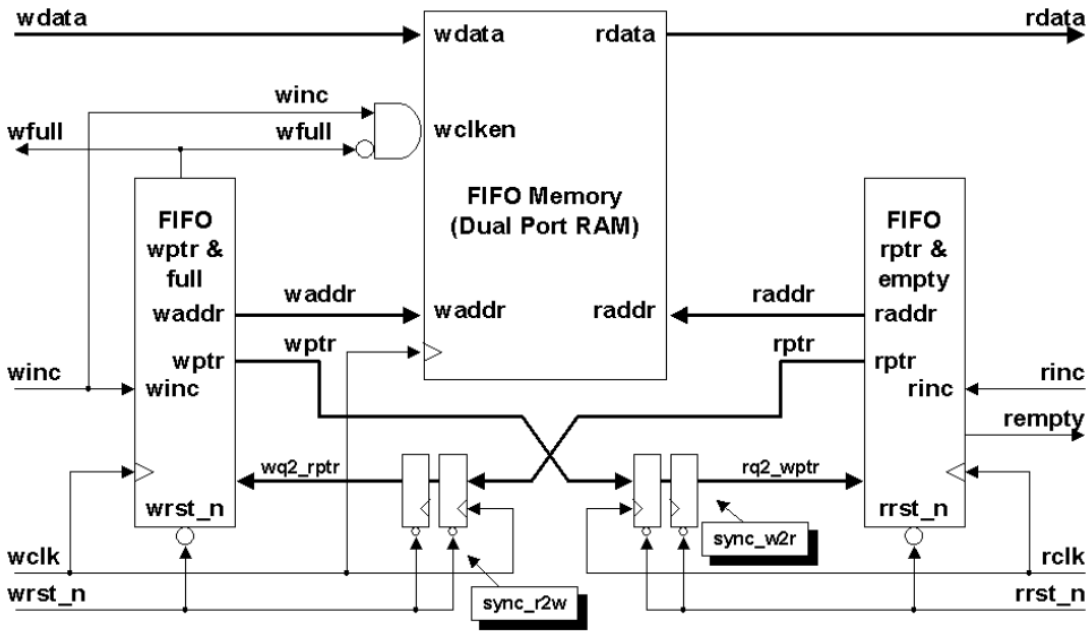
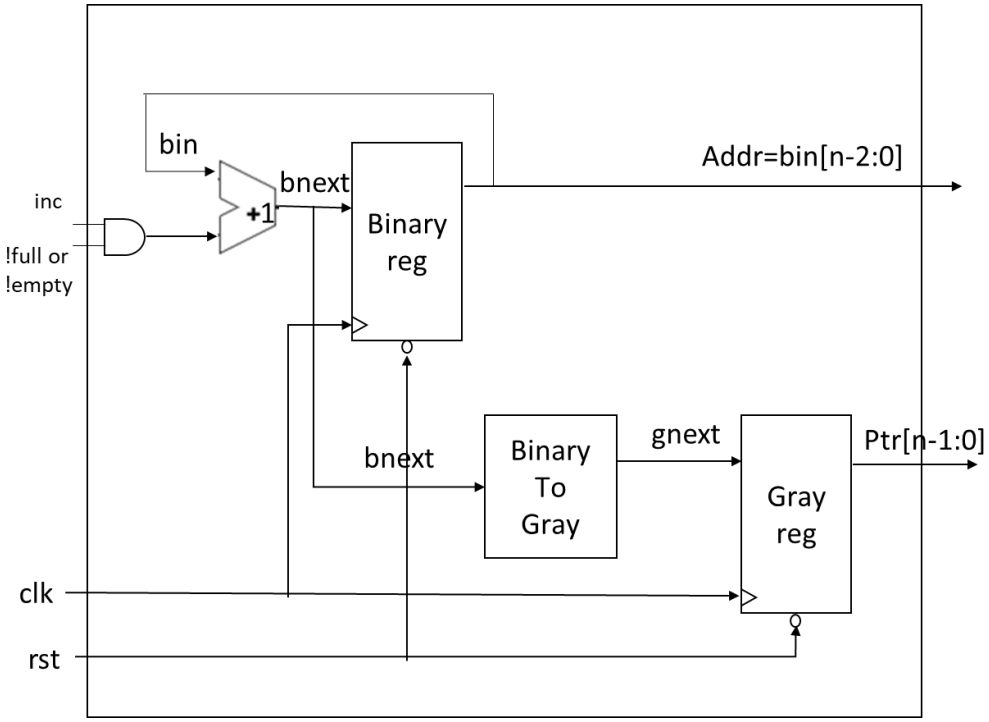


Figure 5 - FIFO1 partitioning with synchronized pointer comparison

FIFO Synchronizers (2/3)



binary	4'd0	4'd1	4'd2	4'd3	4'd4	4'd5	4'd6	4'd7
gray code	4'b0000	4'b0001	4'b0011	4'b0010	4'b0110	4'b0111	4'b0101	4'b0100
binary	4'd8	4'd9	4'd10	4'd11	4'd12	4'd13	4'd14	4'd15
gray code	4'b1100	4'b1101	4'b1111	4'b1110	4'b1010	4'b1011	4'b1001	4'b1000

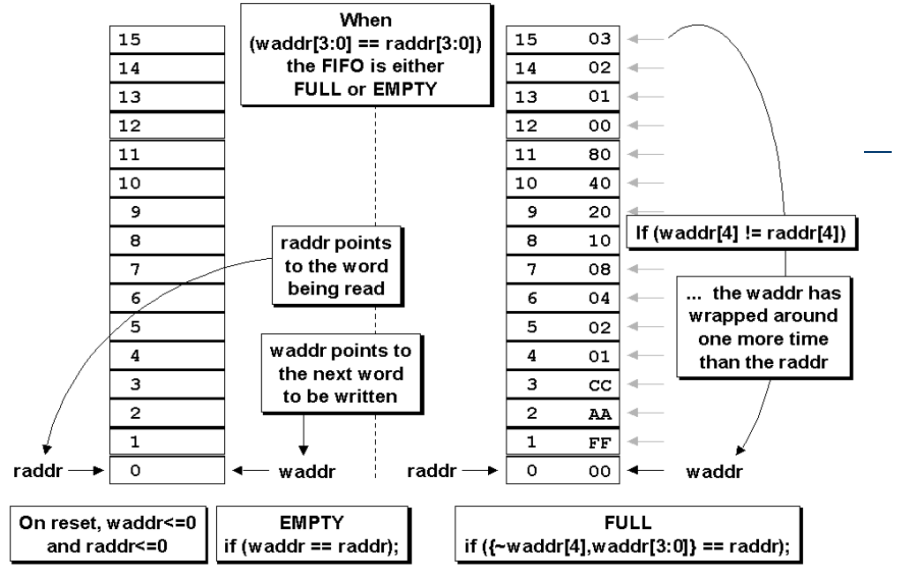


Figure 1 - FIFO full and empty conditions

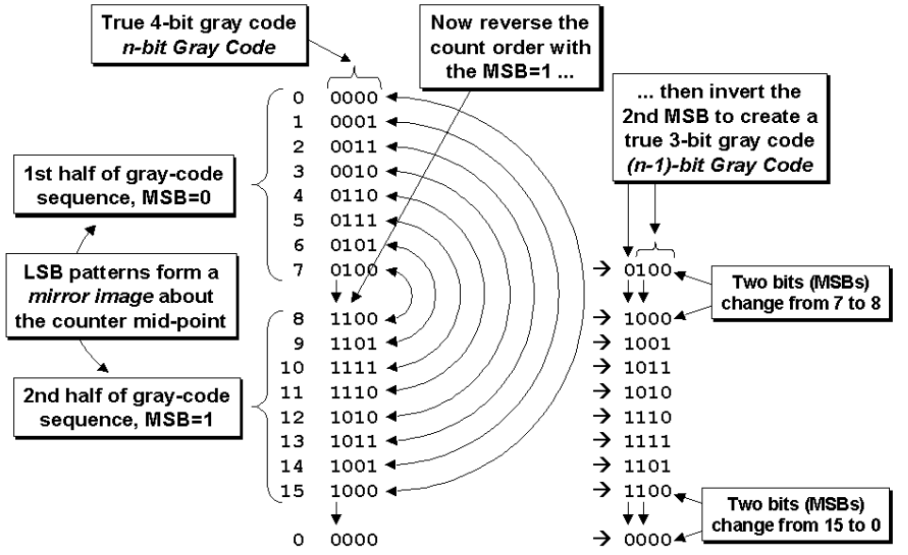


Figure 2 - n-bit Gray code converted to an (n-1)-bit Gray code

FIFO Synchronizers (3/3)

- **Question:** If the write clock (wclk) is faster than the read clock (rclk), could a push operation occurring after an increment of the read address (raddr+1), which is yet to be synchronized to the wclk, cause the write pointer (wptr) to exceed the read pointer (rptr) and result in a FIFO overflow?

- **Answer:** No, an overflow will not occur under these circumstances. Before the read pointer (rptr) is transferred, if the write pointer (wptr) has caught up to rptr-1, the write full flag (wfull) would already be set to 1. It is a standard operation protocol that no push actions are allowed when the FIFO is full (practically ensured by using assertions to check that push is not active when wfull is 1).

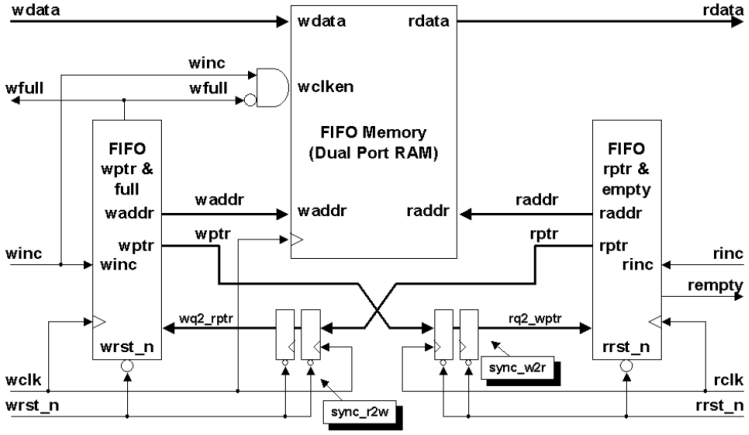
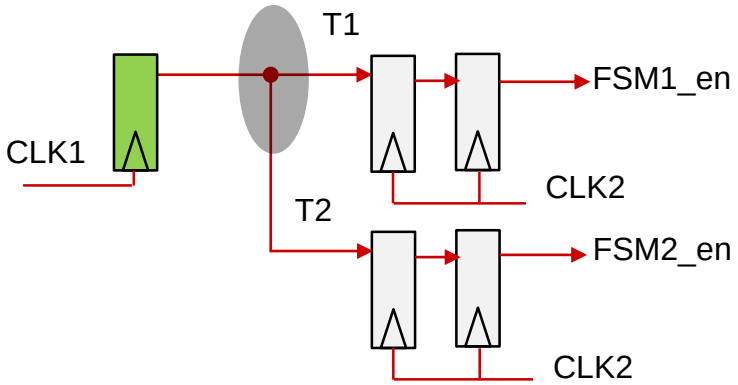
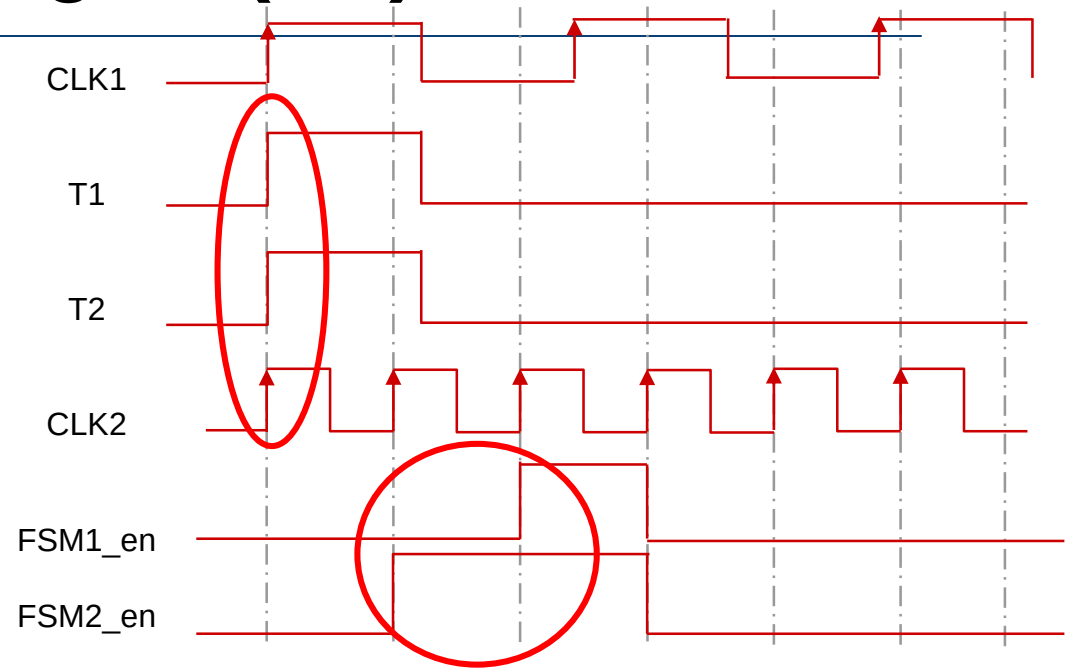


Figure 5 - FIFO1 partitioning with synchronized pointer comparison

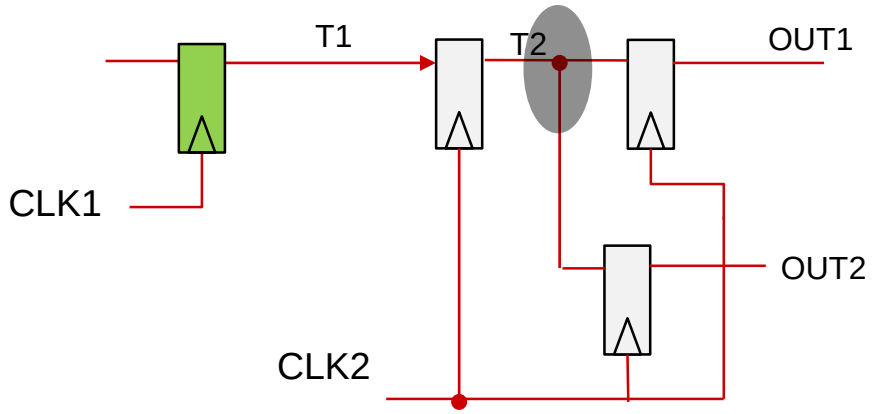
Divergence/Coherency of CDC Signal (1/3)

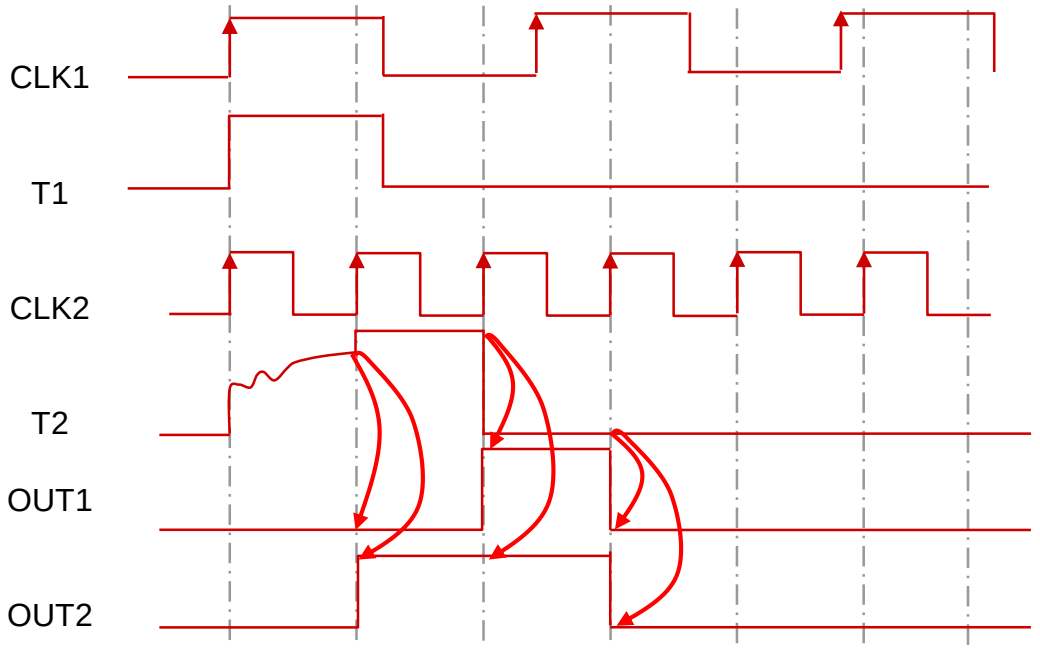




- A divergent logic style to multiple synchronization paths runs the risk of causing functional errors
- Due to the propagation delay and different metastable settling times, the FSM1_en and FSM2_en could start at different times
- This type of structure should be avoided by fanning out a single FSM enable after synchronization to both the FSMs

Divergence/Coherency of CDC Signal (2/3)





- Metastable signals are unstable signals which need proper synchronization
- OUT1 and OUT2 come out at two different clock edges as the settling and latching values of these metastable signals to the two flops are at different times

Divergence/Coherency of CDC Signal (3/3)

- **Issue:**
  - Often arises in large-scale designs where multiple engineers work on different modules of a system
  - An engineer might synchronize a signal entering their clock domain without realizing that the same signal has been synchronized elsewhere within the same clock domain, leading to redundant synchronizations and potential system analysis violations.
- **Implications:**
  - Unnecessary resource utilization due to multiple synchronizers.
  - Signal inconsistencies because synchronizers can introduce changes at different cycles, potentially causing errors in downstream circuits.
- **Resolution:**
  - Remove additional synchronizers to ensure signal consistency.
  - Implement a design protocol to track synchronization across modules, preventing redundancy.

Commonly Used Synchronization Schemes

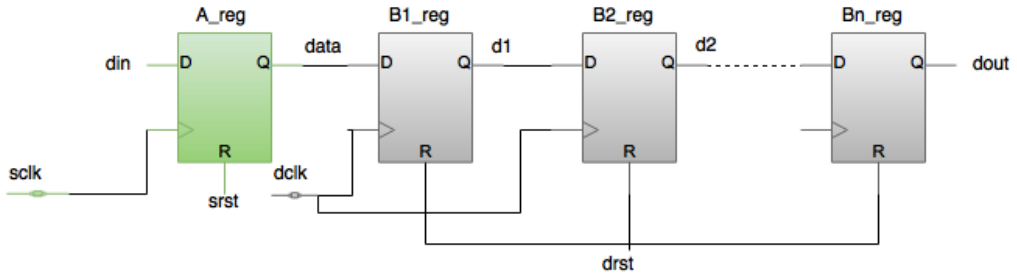


Fig: NDFF synchronizer

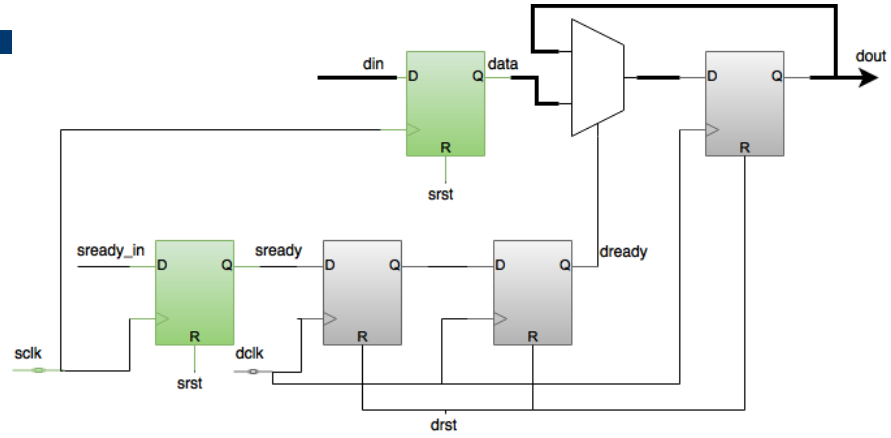


Fig: MUX synchronizer

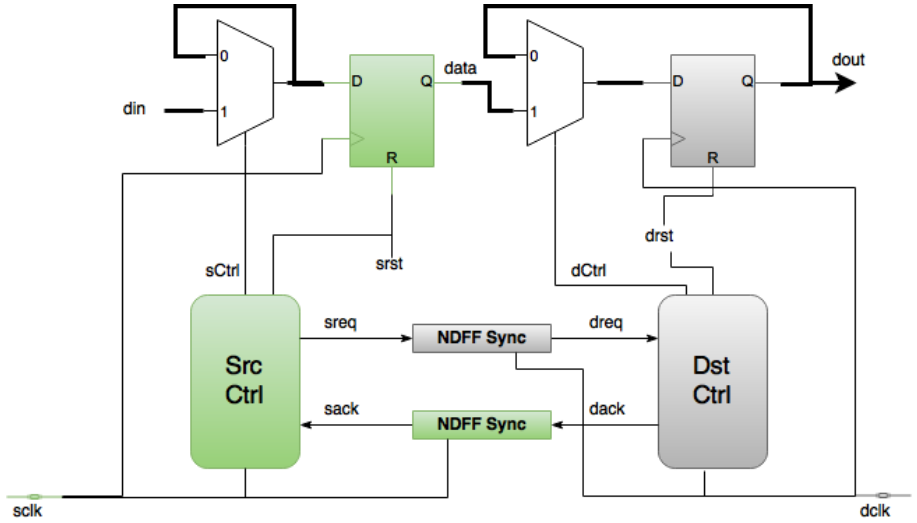


Fig: Handshake synchronizer

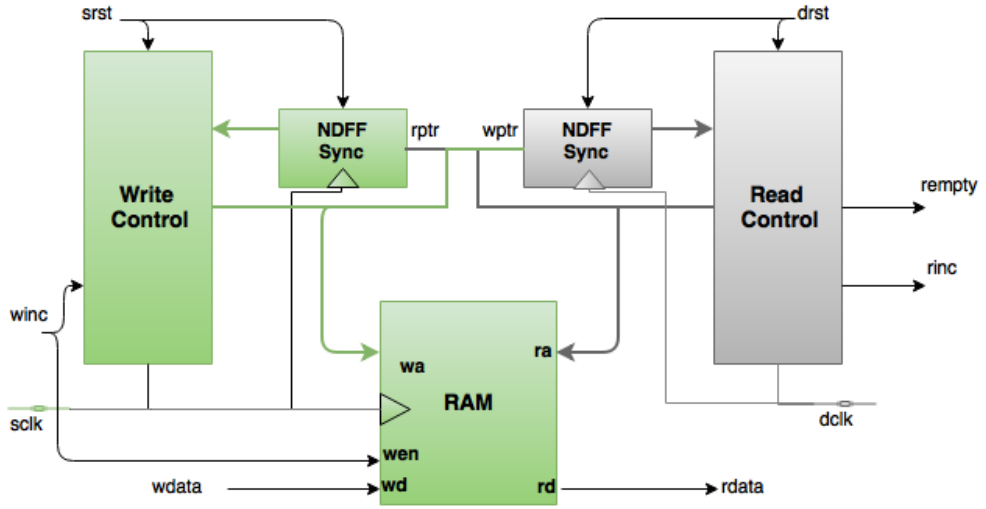


Fig: FIFO synchronizer

Industry Tools for CDC Verification

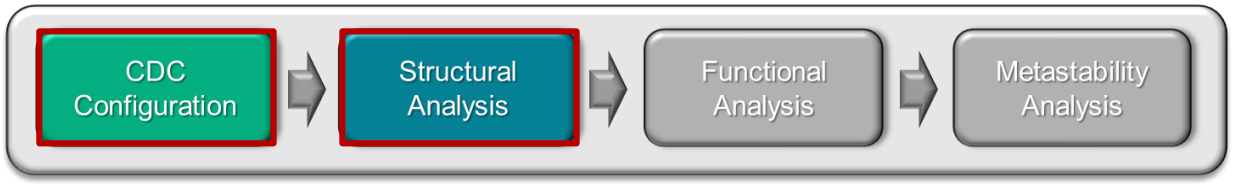
- Top EDA companies such as Synopsys, Cadence, and Mentor Graphics offer CDC verification tools: Synopsys Spyglass CDC, Cadence Jasper CDC , and Mentor Questa

- **Verification Process:**
  - **Setup Check:** Verifies that all flops have defined clocks and resets.
  - **Structural Check:** Conducts static analysis to ensure basic CDC rules are met without understanding the functional intent.
  - **Functional Check:** Requires understanding the design's functionality to identify defects from a runtime perspective

- **Why Both Structural and Functional Checks?**
  - Structural checks can identify most CDC issues.
  - Functional checks are necessary to determine the stability of signals under specific conditions, like ensuring data stability when an enable signal is active or verify gray encoding.

Introduction to Jasper® CDC app

CDC Configuration and Structural Analysis



CDC configuration

- Specify clock properties and relationships (using SDC or native commands)
- Specify correct reset types (async/synchronized/synchronous) and reset priority
- Declare signal configurations (constant/static/mutex/gray-code) with conditions

Garbage In = Garbage Out

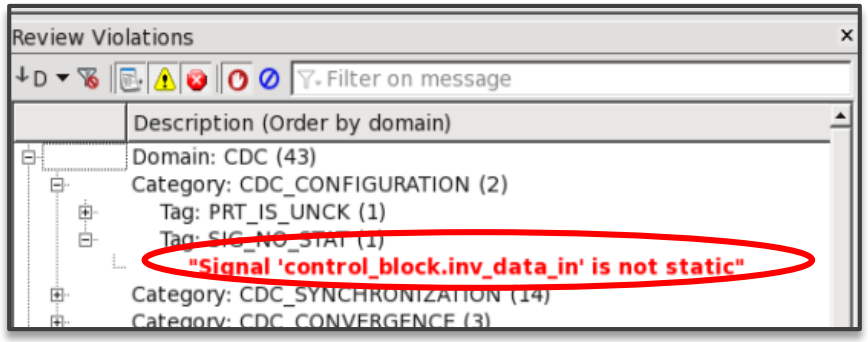
Comprehensive structural analysis

- Clock and Reset tree checks
- CDC synchronization checks
- Convergence/divergence/glitch checks
- Reset synchronization, RDC checks
- Waivers added while dispositioning structural violations

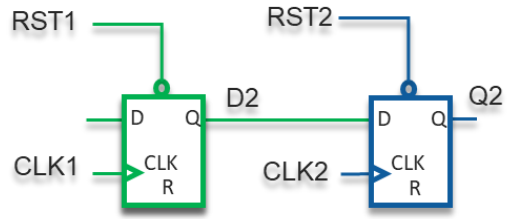
Wrong Waivers = Masking Issue

Signal configuration correctness and waiver related functional assumptions should be verified

Functional CDC Analysis



- Signal configuration validation
  - Auto-generated assertions for verification
  - Proven in formal verification environment
- Verify the ***pseudo*** nature of constraints
  - Specify triggering event for signal to be constant/static
  - Example:
    - Data on CLK1 domain can change only when destination is in reset (RST2 is asserted)
    - `Config_rtlds -signal -static D2 -condition RST2 -verify`
  - Additional verification to ensure pseudo-static property
- Export and run signal configuration checks in simulation



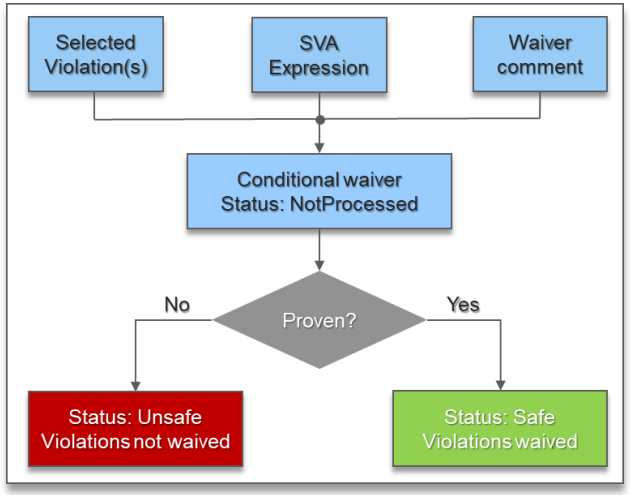
Verification of analysis constraint correctness leads to greater confidence!

Functional CDC Analysis contd.

Functional Analysis

Waiver condition validation

- **Validation of functional assumptions used in dispositioning structural violations**
- **Auto-waiver flow**
  - Tool automatically detect conditions for dispositioning structural violations
  - Violations are waived if the condition is proven
  - Example: Gray encoded buses, unsynchronized paths but are metastability safe
- **Conditional waiver flow**
  - User provides SVA expression for waiver condition
  - User-selected violations are waived only if the condition is proven
  - Export waiver conditions to simulation



Verification reduces noise and avoids errors in manual dispositioning of violations

Functional CDC Analysis contd.

Functional Analysis

CDC protocol checks

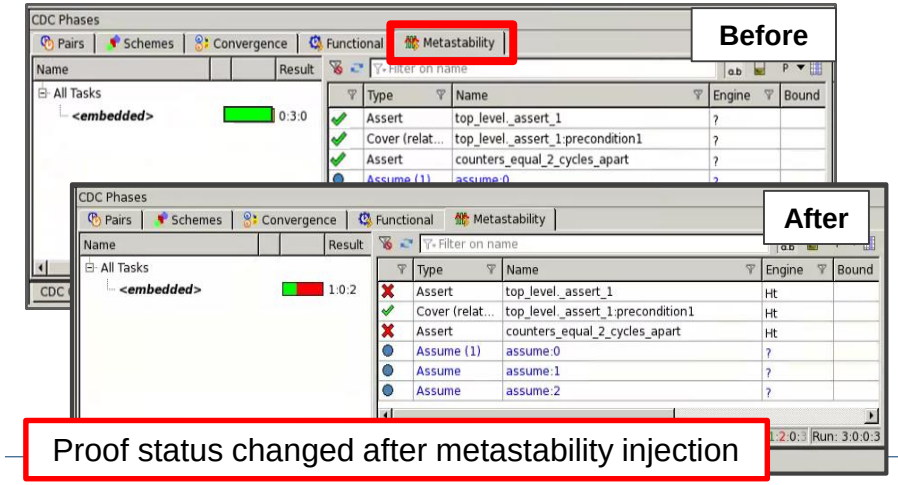
- Auto-generated protocol checks for all identified synchronizers
- Supports user-defined protocol checks for custom synchronizers
- Requires very strong formal tool capabilities
  - Best in class Jasper® formal engines
- Export protocol checks to run in simulation

Protocol Checks	Synchronization Schemes
Control Path Stability	NDFF, NDFF_BUS, Pulse, Edge, Glitch Protector
Data Path Stability	MUX_NDFF, MUX_PULSE, Sync Enabler, Glitch Protector
Pulse Width Toggle Circuit	Pulse
Req/Ack Stability Data Stability Handshake Protocol	Handshake Synchronizer
Read/Write Pointer Stability Overrun/Underrun Gray Encoding of Pointers	FIFO Synchronizer

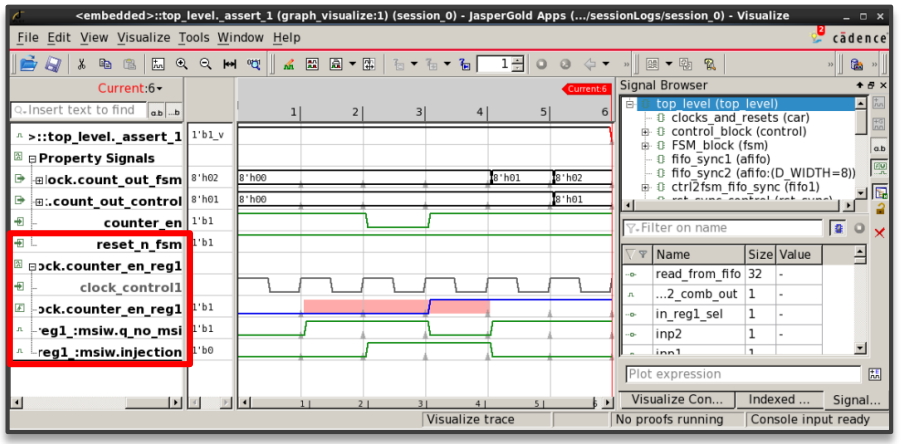
Metastability Injection (MSI) Flow in Formal

- Pre-requisite: Formal Property Verification (FPV) environment
  - Passing functional assertions
- Push button flow with customized debug in visualize
  1. Inject metastability in user properties
  2. Verify user-defined properties in presence of metastability
- Metastability awareness in protocol checks

Only tool capable of verifying asynchronous logic in formal



Proof status changed after metastability injection



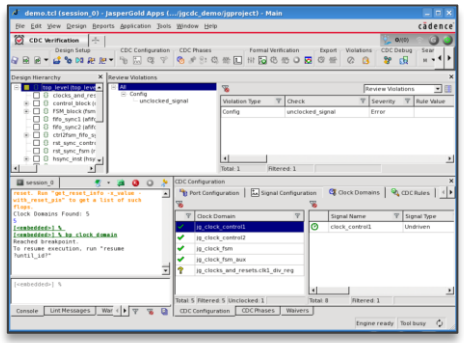
Customized debug in visualize

Metastability Injection (MSI) Flow in Simulation

Metastability Analysis

- Pre-requisite: Simulation environment with passing test cases
- Simple, easy to use, no instantiation flow in simulation
  - Timing violation monitors and injection modules exported from Jasper ®
  - Injection models can mimic both setup and hold violations
  - Not dependent on synchronizer types – all CDC crossings covered
  - Configurable *setup, hold* time windows for individual clocks
- Random/Always/No Injection modes
- Specialized debug in Indago/Simvision using injection witness signals

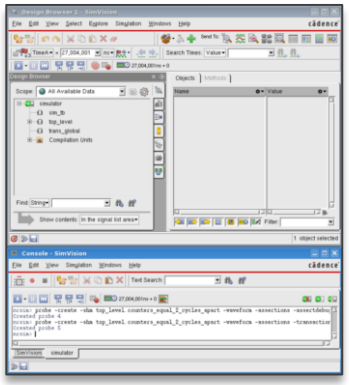
Works with Xcelium™ only



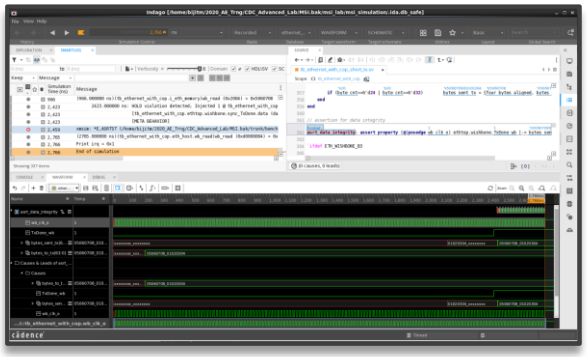
Jasper CDC

Export from Jasper

Timing violation monitors and injection modules



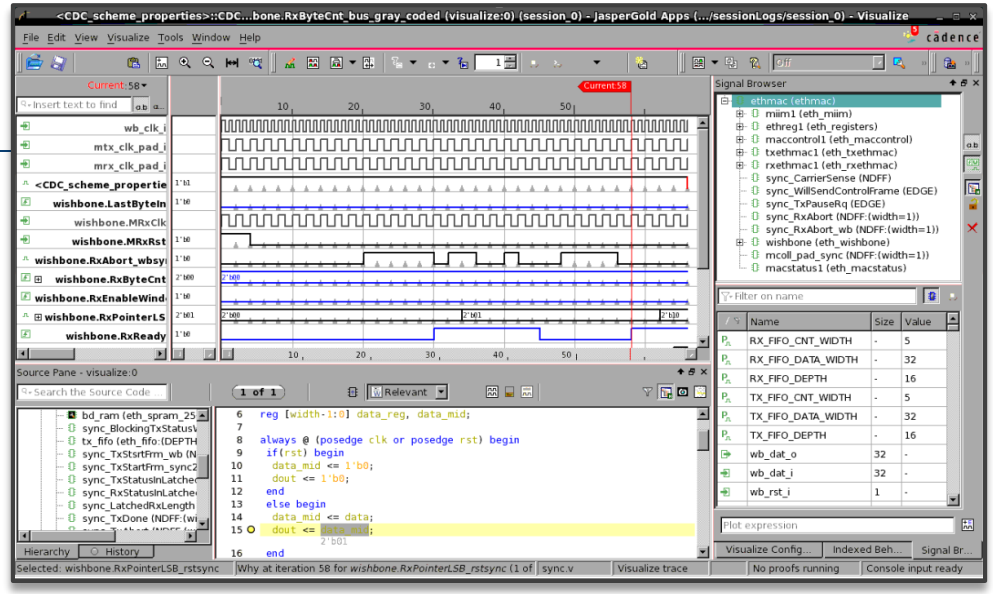
Xcelium Simulator



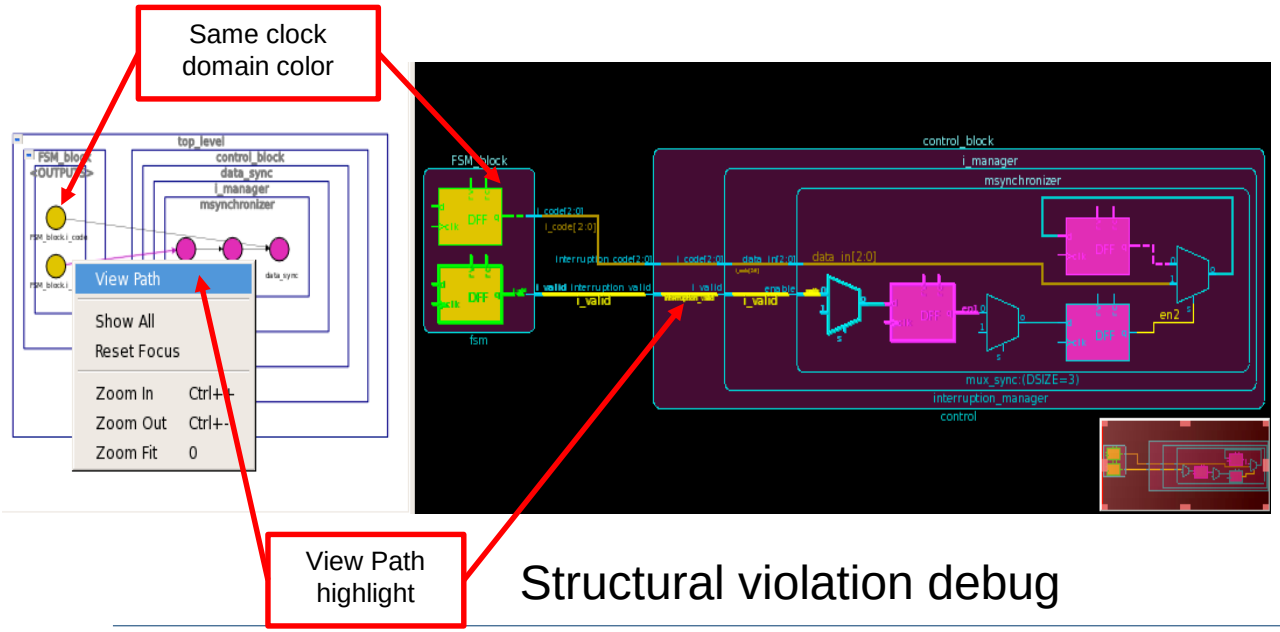
Debug with Indago

Advanced Debug Environment

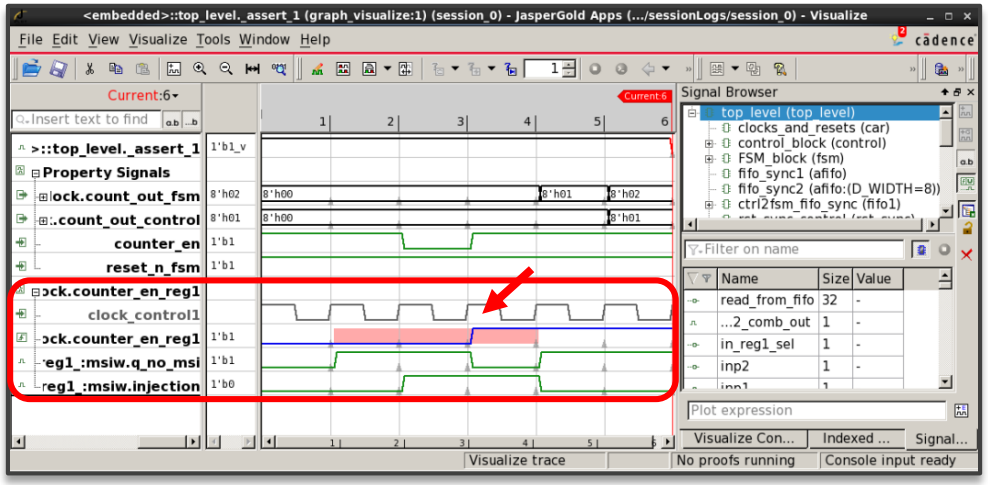
- Power of Visualize
- Exclusive Graph view
- Schematic and source code browser
- Tight integration among all debug views



Functional violation debug

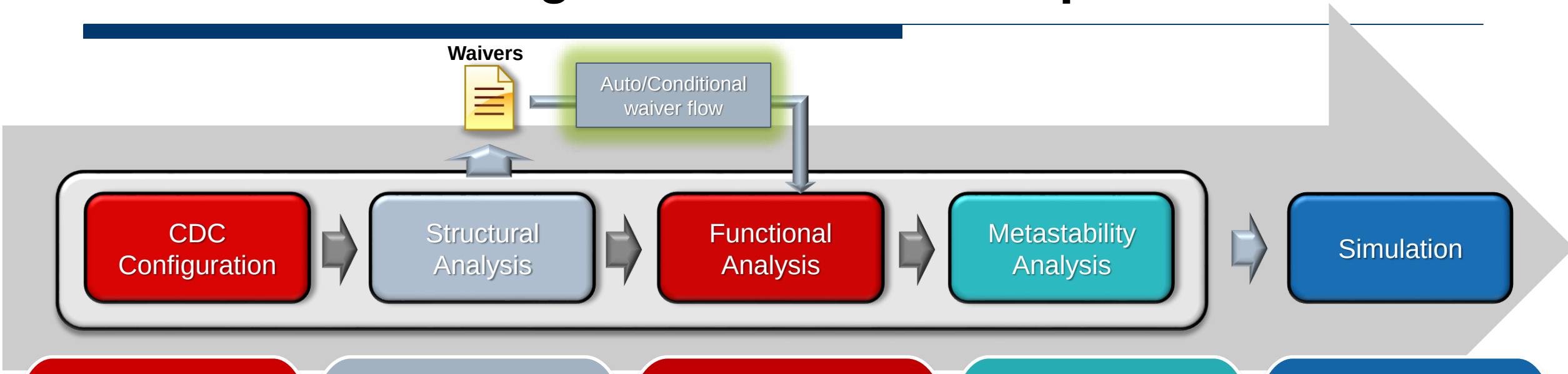


Structural violation debug



Metastability violation debug

True CDC/RDC Sign-Off Flow with Jasper® CDC



- Clock specifications
- Reset specifications
- Signal configurations

- Clock/Reset tree checks
- CDC synchronization checks
- Convergence/glitch checks
- Reset synchronization checks
- Reset domain crossing checks

- Signal configuration validation
- Clock Relationship checks
- Reset Order/Priority checks
- CDC protocol checks
- Waiver condition validation

Metastability aware proof of:
- User-written assertions
- CDC protocol checks
- Waiver conditions

- Metastability aware simulation
- CDC protocol checks
- Signal config checks
- Waiver conditions

Tool Summary

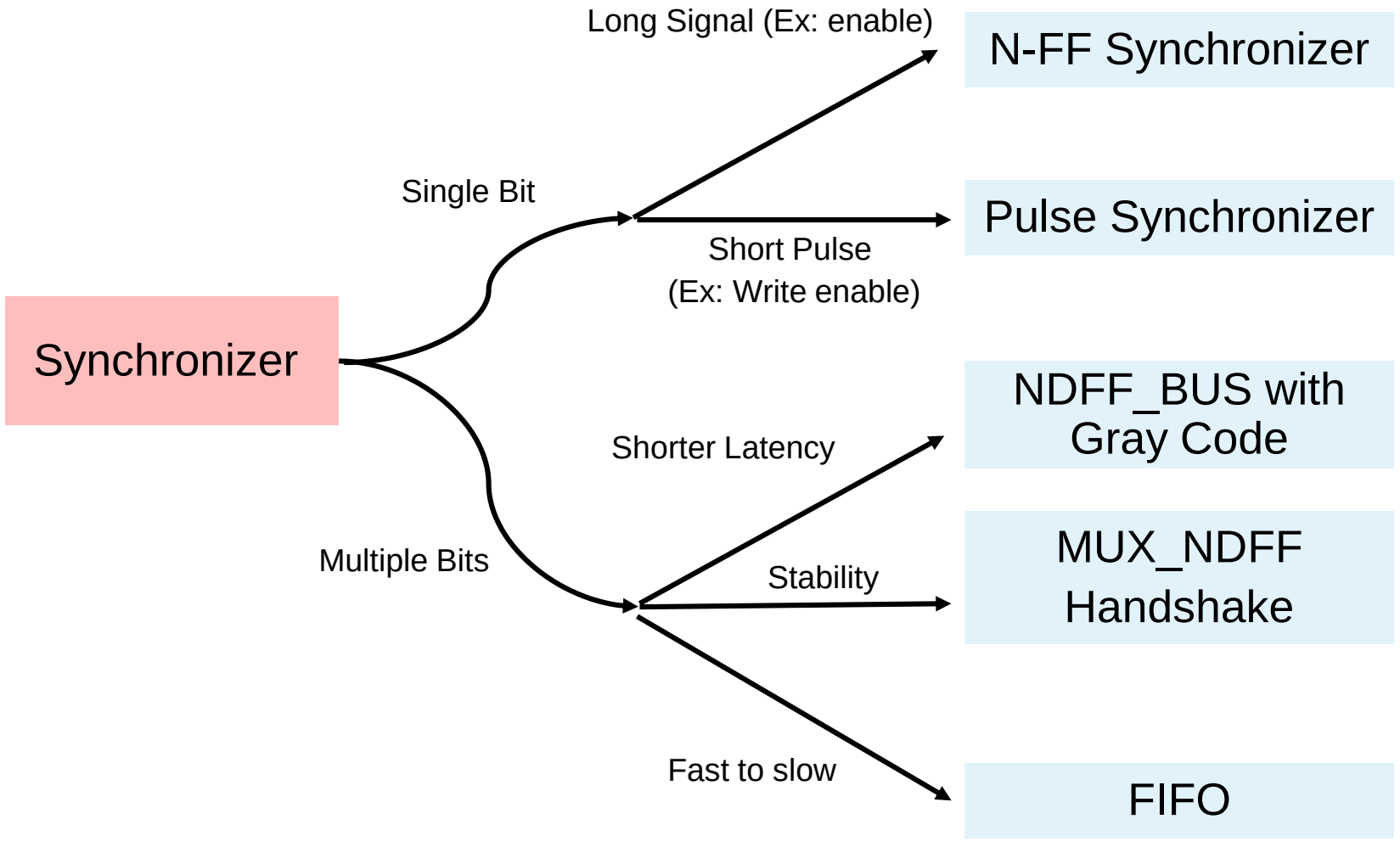
- Structural CDC/RDC analysis is not comprehensive enough for sign-off
- Focus beyond structural analysis
  - Verification helps in reducing noise and avoid errors in manual dispositioning
  - Linking analysis with verification is the key to "True CDC/RDC Sign-off"

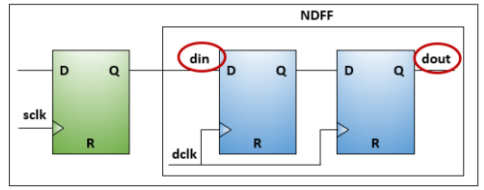
- Jasper® CDC app is a holistic CDC/RDC verification solution
  - Comprehensive structural checks
  - Functional CDC/RDC verification
    - Constraint validation
    - Waiver validation
    - CDC protocol verification
  - Metastability-aware verification

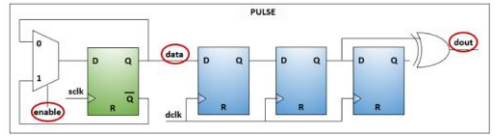
Think Before Designing: Is CDC Necessary?

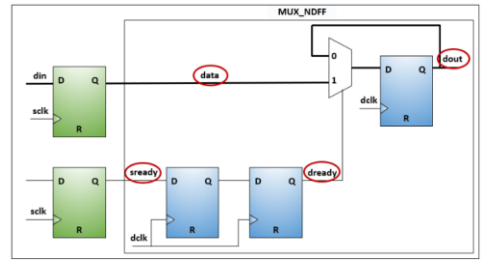
- **One Module, One Clock Domain**
  - Group logic from the same clock domain in one module for synchronous design.
  - Leverage a separate module for synchronization logic for clear CDC boundaries.
- **Robust Synchronization**
  - Assume clocks have arbitrary relationships to ensure design robustness.
- **Design Outputs and Signal Synchronization**
  - Ensure CDC input originate from a register; avoid using combinational logic outputs.
  - Sync signals to another clock domain only once to prevent coherency issues.
  - Mark cross-domain signals with clock names for clarity (e.g., req_aclk, req_pclk).
- **Testing and Simulation**
  - In simulations, induce random values in synchronizers to uncover hidden bugs.
  - Stagger the edges of asynchronous clocks in testbenches, especially if clocks share frequencies.
  - Avoid aligning clock edges to ensure phase randomness.

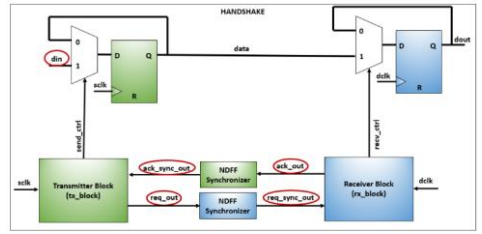
Commonly Used Synchronization Schemes









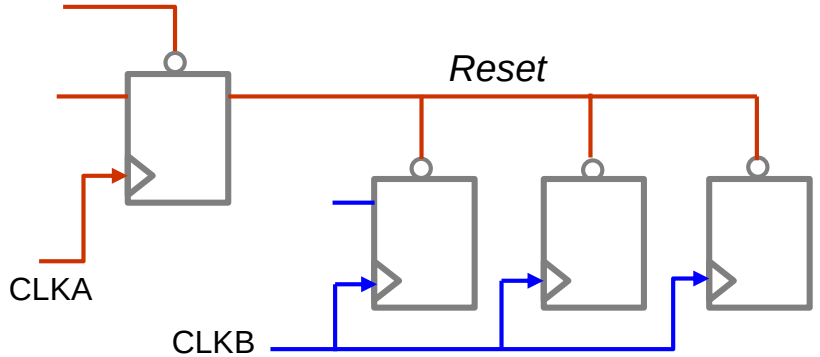


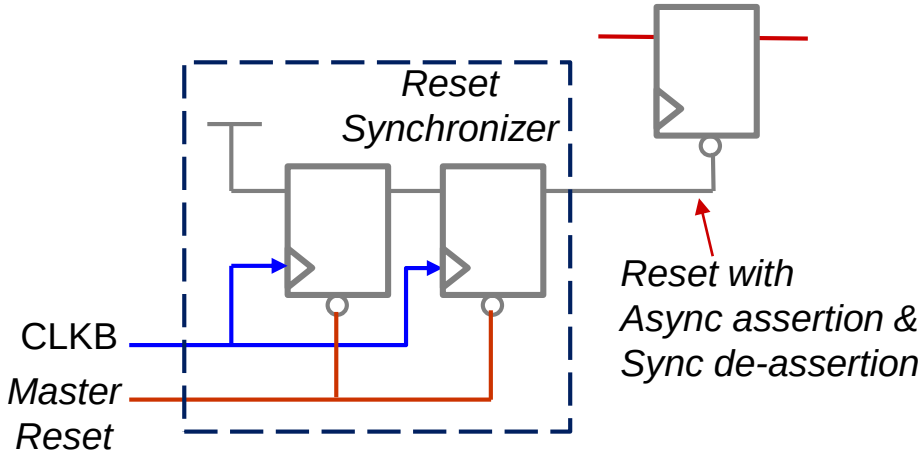
Outline

- Clock Domain Crossing (CDC) Overview
  - Metastability & Clock Domain Crossing
  - Single-Bit – N-FF Synchronizer / Pulse Synchronizer / Edge Synchronizer
  - Multi-Bit – Convergence
  - Multi-Bit – NDFF_BUS with Gray Code
  - Multi-Bit – MUX_NDFF / MUX_PULSE Synchronizer
  - Multi-Bit – Handshake / FIFO Synchronizer
  - Single-Bit – Divergence
  - Common Synchronization Schemes
- Introduction to Jasper® CDC app
  - Structural Analysis
  - Functional CDC Analysis

Supplement: Reset Synchronization

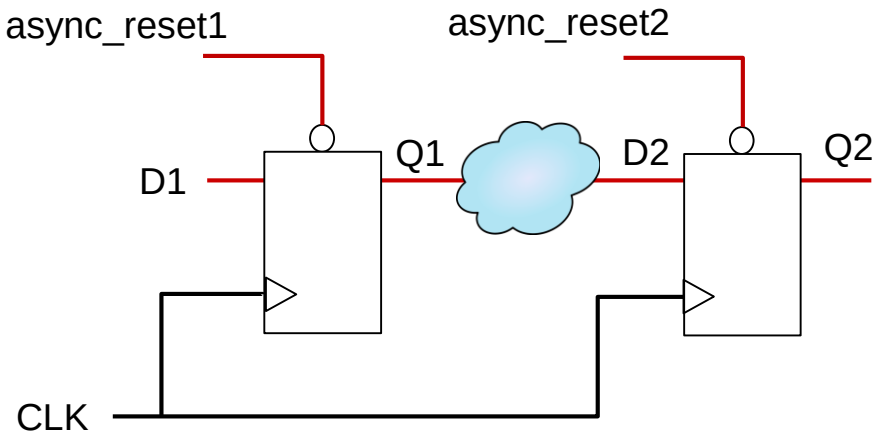
- **Reset Synchronization**
  - If asynchronous reset is de-asserted near the active edge of the clock and violates the reset recovery time, it could cause the flip-flop to go metastable
  - Solution: Asynchronous reset assertion and synchronous reset de-assertion
  - Tool should report such scenarios

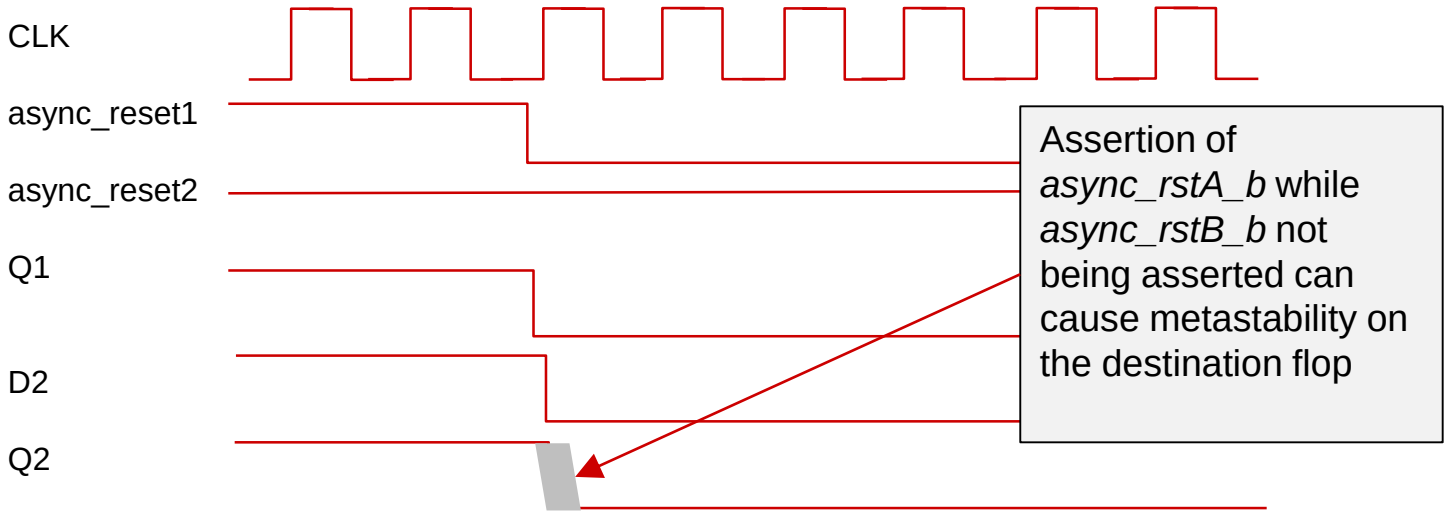




Supplement: Reset Domain Crossing

- Just like clocks, there are reset domains in the design
- Reset domain crossing can happen even within the same clock domain
- If the async reset of source register is different from the async reset of destination register, even though the data path is in same clock domain, it can lead to metastability at the destination register
- Tool should report such scenarios





Reference

- 跟老李一起學習晶片設計-- CDC的那些事（1）
- Clock Domain Crossing(CDC)
- 關於跨clock domain處理的觀念
- 數位電路中的CDC問題
- R. Ginosar, "Metastability and Synchronizers: A Tutorial," in *IEEE Design & Test of Computers*, vol. 28, no. 5, pp. 23-35, Sept.-Oct. 2011, doi: 10.1109/MDT.2011.113.