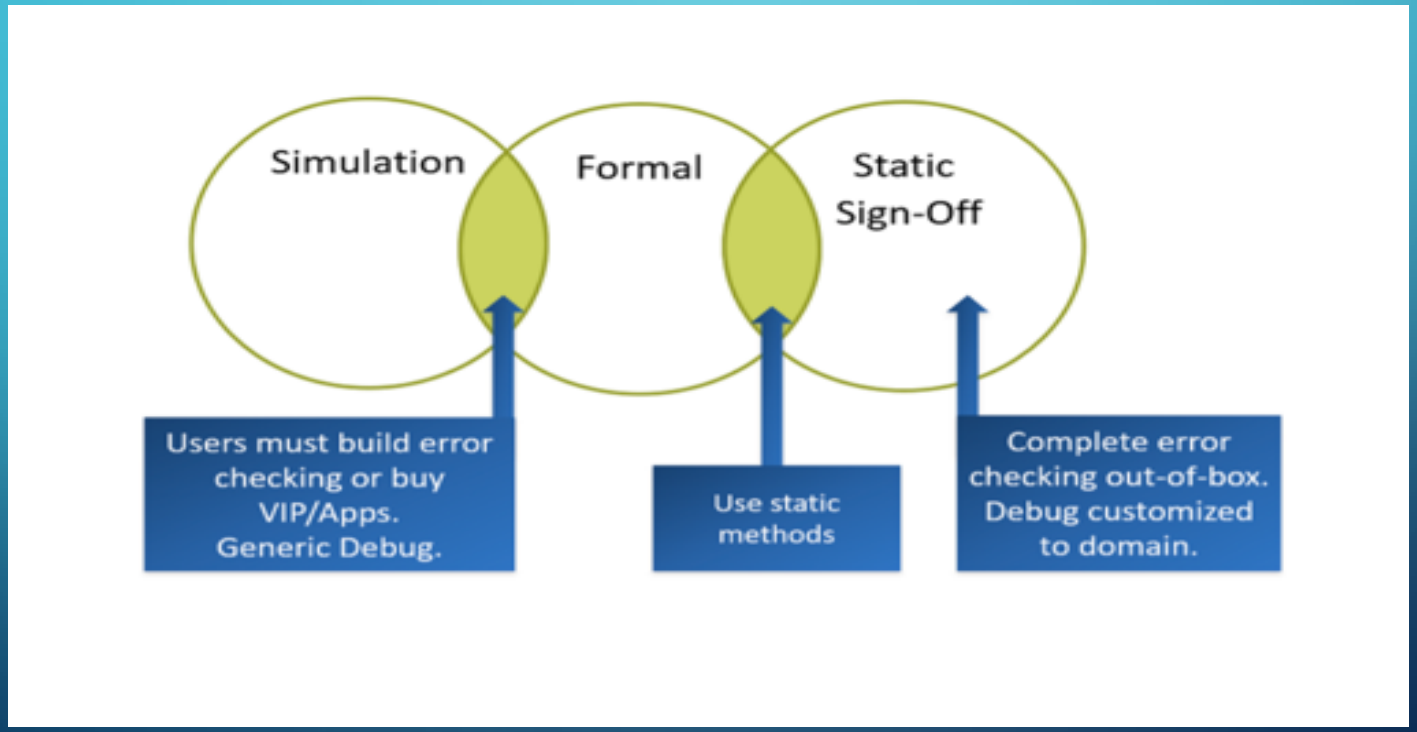

STATIC SIGN-OFF
INSTRUCTOR: DR. HARI MONY
REAL INTENT
EE 382M-11, VERIFICATION OF DIGITAL SYSTEMS
SPRING 2021

WHAT IS STATIC SIGN-OFF?

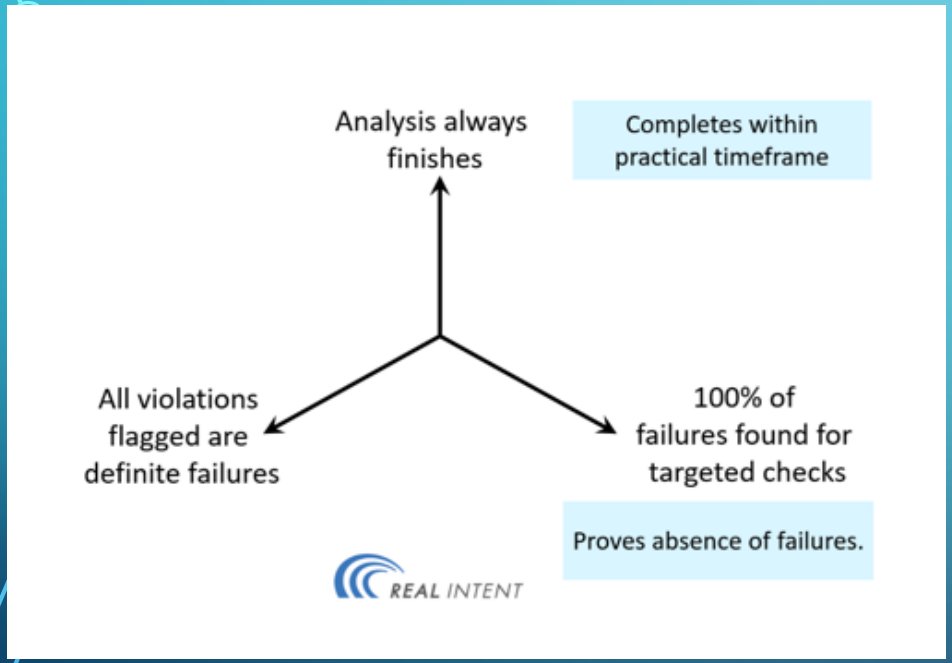
- Dynamic Verification: Compute design behavior dynamically, and then check the computed behavior for failures under user-specified test cases.
  - E.g., Simulation, Emulation
- Static Verification: Utilize search and analysis techniques to check for design failures under all possible test cases.
  - E.g., DRC (Design Rule Checking) for layout, STA (static timing analysis)
  - Formal verification
  - Functional static sign-off. E.g., clock-domain verification

SIMULATION, FORMAL AND STATIC SIGN-OFF



Source: Prakash Narain, Real Intent. DAC 2019 Panel

MAXIMIZING VERIFICATION EFFICIENCY



Verification Metric	Simulation	Formal	Static Sign-off
Analysis always finishes	Yes	No	Yes
100% of failures found for target checks	No	Yes	Yes
All violations flagged are definite failures	Yes	Yes	No

Source: Prakash Narain, Real Intent. DAC 2019 Panel

TARGET DOMAINS FOR STATIC SIGN-OFF

- **RTL Linting.** Enforces coding guidelines and identifies functional issues prior to simulation. Rules cover syntax, semantic, and style checks, to ensure high quality RTL.

- **Clock Domain Crossing.** Structural and functional analysis ensure that signals crossing asynchronous clock domains are received reliably, verifying that the data will be transferred across clock domains without introducing design problem

TARGET DOMAINS FOR STATIC SIGN-OFF

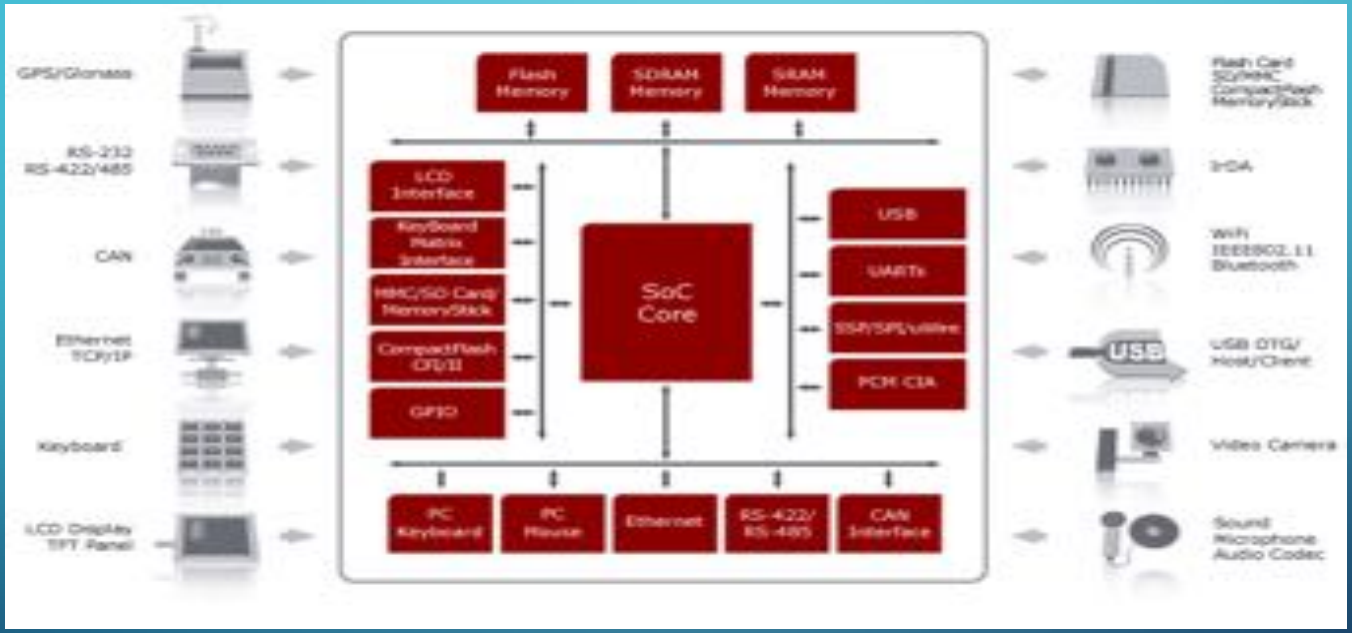
- **Reset Domain Crossing.** Confirms that signals crossing reset domains function reliably, identifying metastability problems and glitches arising from software and/or low power asynchronous resets, or re-convergence of synchronized resets.

- **Design for Test.** Early RTL and netlist analyses are performed to identify all testability violations in a timely and efficient manner before full DFT/ATPG.

CLOCK DOMAIN CROSSING

- CLOCK DOMAIN: a set of flops and associated logic which are always clocked by a common clock or by clocks having a *common frequency* and *fixed phase relationship*.
- Two clock domains are **operating asynchronously** with respect to each other if their respective clocks do not have a *fixed phase and frequency relationship* with each other and neither domain is in a hold state
- **Asynchronous domain crossing:** a path from a flop in one clock domain (the **transmit flop**) to a flop in a different clock domain (the **receive flop**), through which transitions may occur when the two clock domains are operating asynchronously with respect to each other

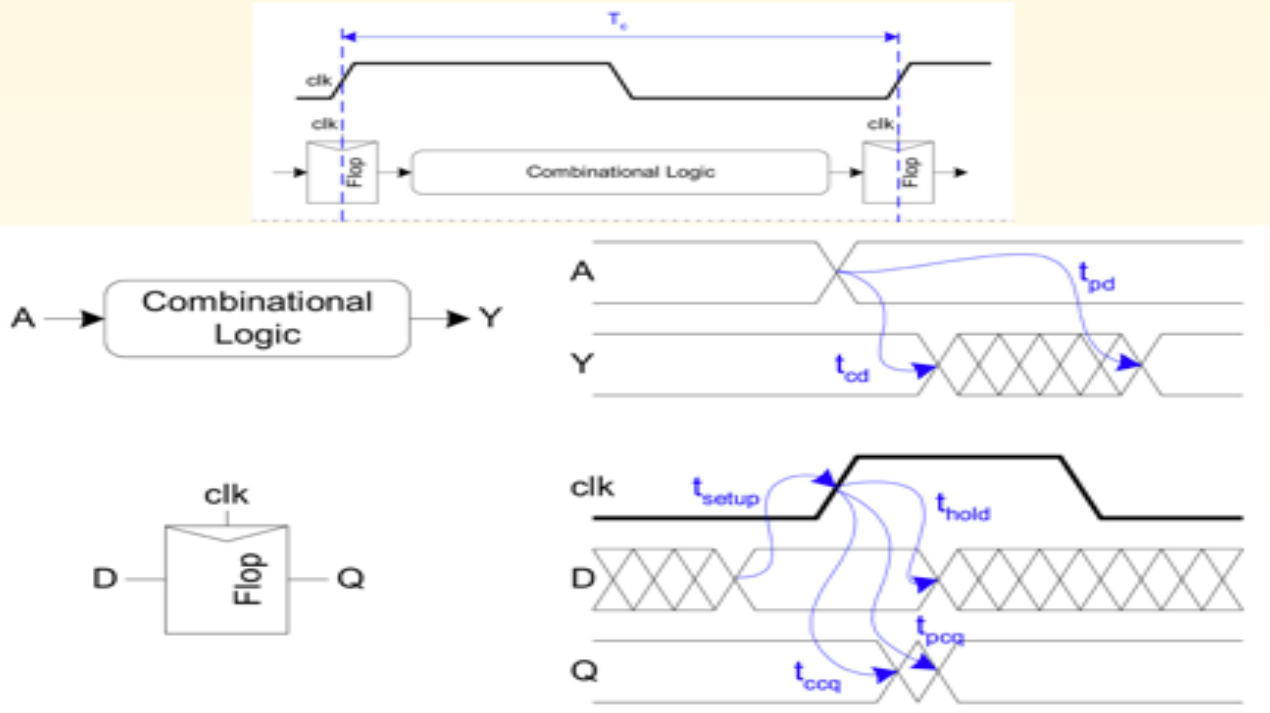
SOC CLOCK DOMAIN EXPLOSION



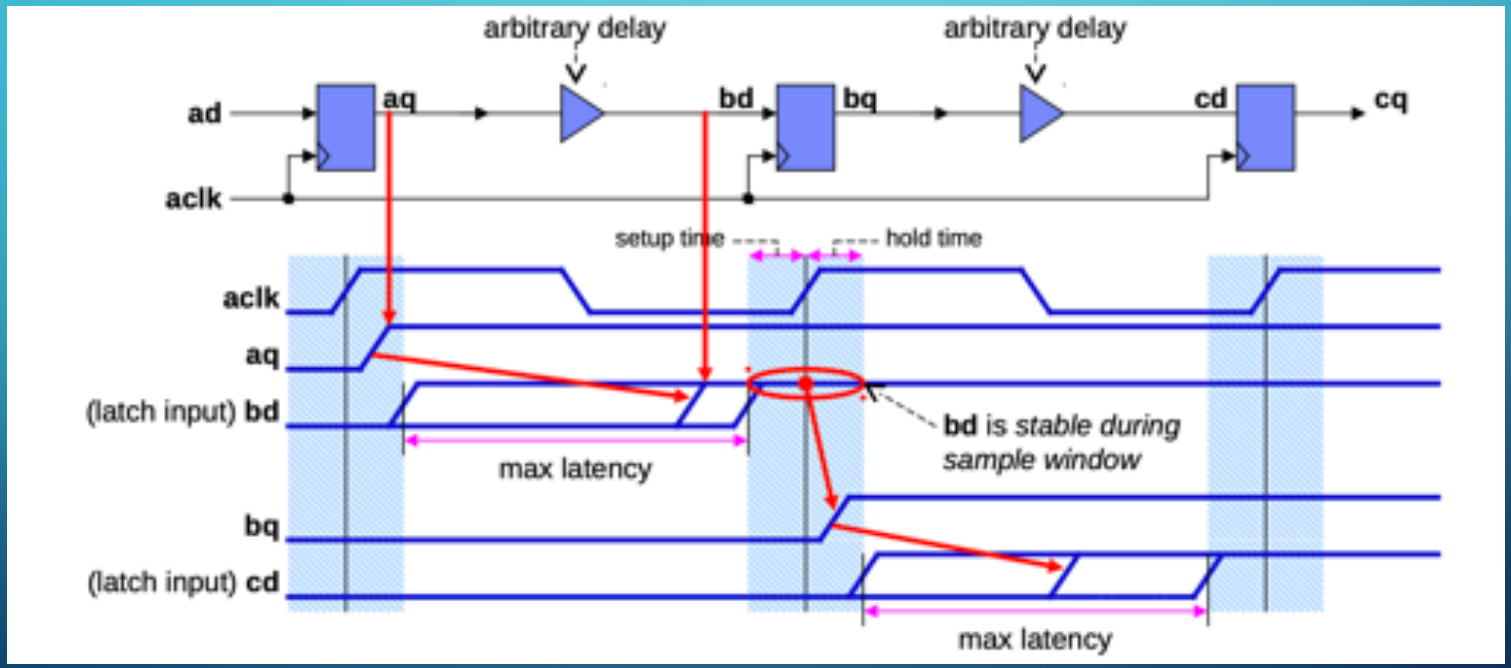
Driven by SoC requirements and clock distribution
Asynchronous clock domains: ~100
Domain crossing signals: ~10000

Source: Real Intent, DAC 2019

REVIEW: SETUP AND HOLD TIMES

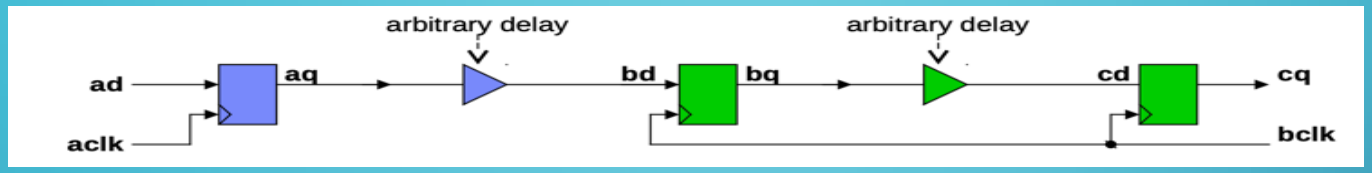


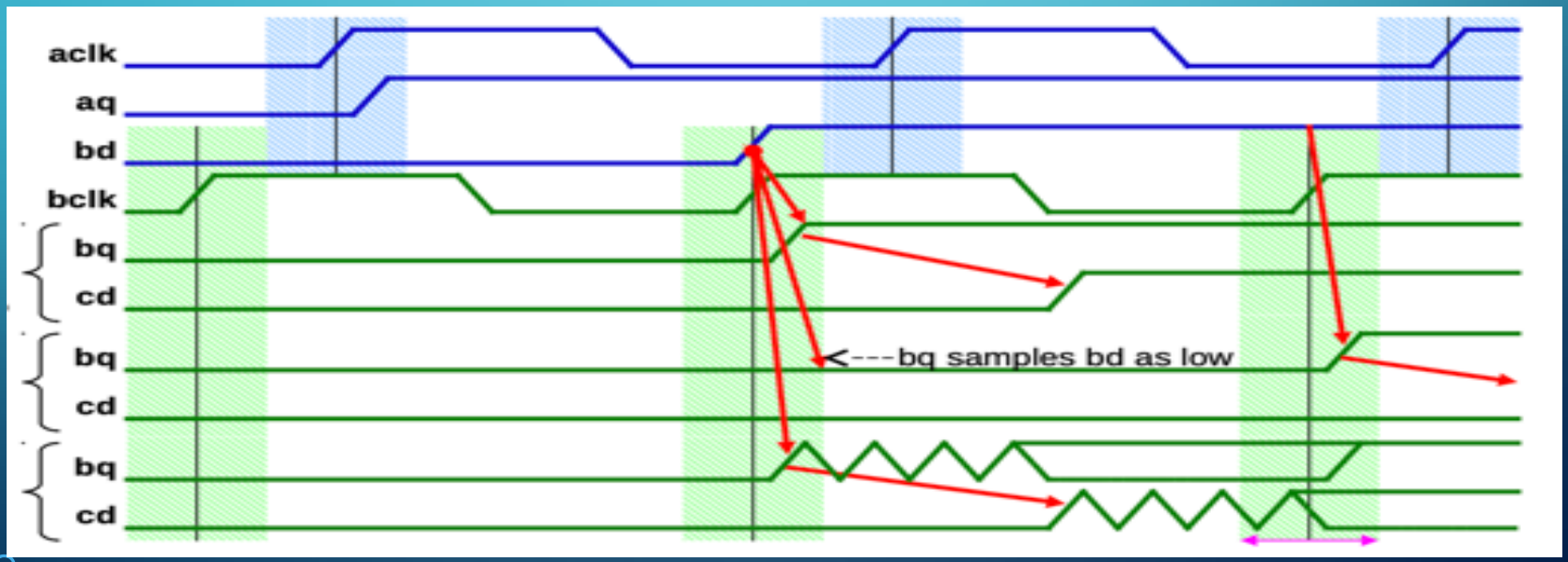
SYNCHRONOUS CROSSING



- Static timing analysis ensures that setup and hold times are not violated

ASYNCHRONOUS CROSSING





ANALYZING METASTABLE BEHAVIOR

Probability of a Metastable State

1. Frequency of the destination domain
2. Rate of data crossing the clock boundary

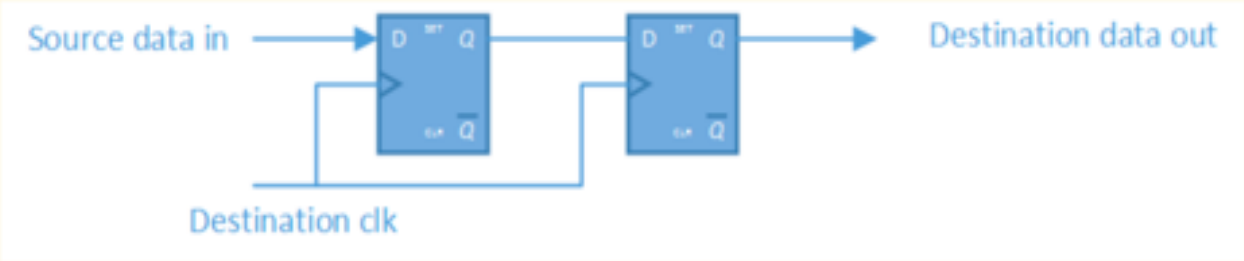
Mean Time Before Failure (MTBF)

- f_{clk}: Synchronizing clock frequency
- f_{data}: Frequency of data changing
- X: Other factors

$$\text{MTBF} = \frac{1}{f_{clk} + f_{data} + X}$$

Source: Dally and Poulton, *Digital Systems Engineering*, 1998

TWO FLIP-FLOP SYNCHRONIZERS



Even if the first flop becomes metastable, due to the input data changing very close the the clock edge (within setup/hold time constraints), there is still a full clock for the signal to become stable before being sampled by the second flop

Source: Yu, "Clock Domain Crossing Design", *Verilog Pro*, March 2016

WHY IS CDC VERIFICATION DIFFICULT?

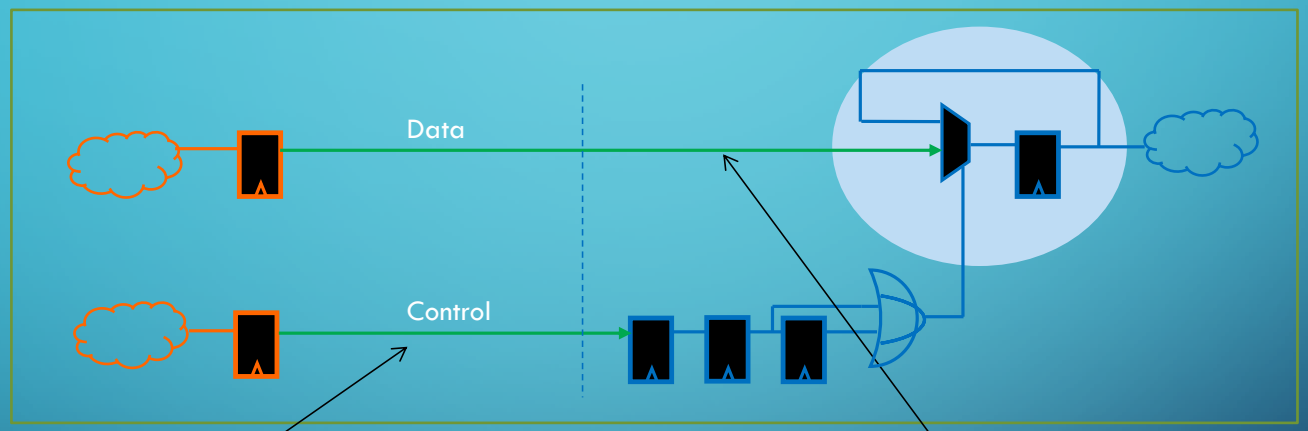
- CDC bugs are a result of bad implementation, timing, and logic
- Hard to detect and diagnose (with simulation or in the lab during design bring-up)
- Frequent chip failures in the field – expensive to fix

Analysis is difficult

- Very high number of CDC crossings
- Variety of ways of implementing the crossings
- Need to model metastability

SYNCHRONIZATION SCHEMES

Handshake Protocols



Control crossing must:

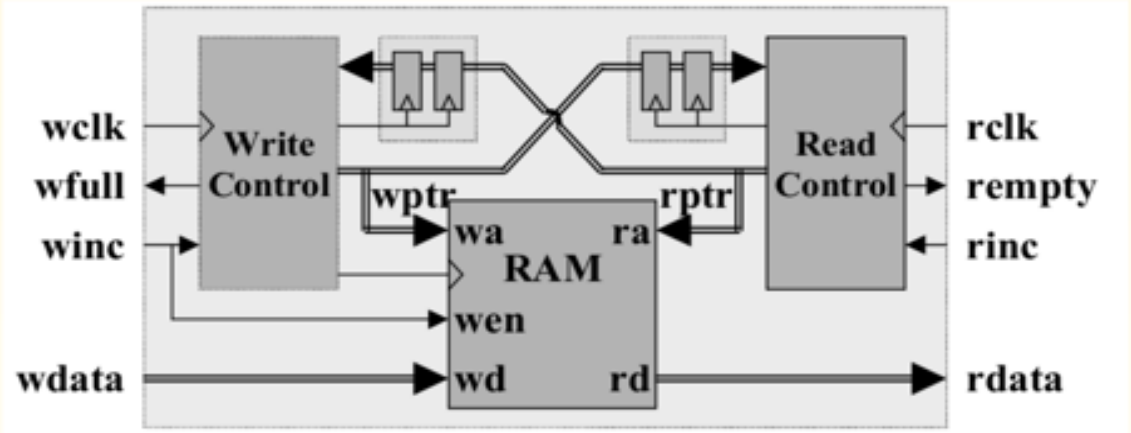
1. Be synchronized
2. Have only one signal transition for buses (gray-coded)
3. Be free of hazards & glitches
4. Be stable for more than 1 receiving clock cycle

Data crossing needs to be stable for more than 1 receiving clock cycle prior to being latched

Source: Real Intent, DAC 2019

SYNCHRONIZATION SCHEMES

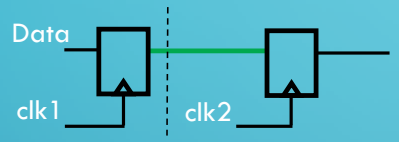
- Data written into the FIFO from the source clock
- Data read from the FIFO in the destination clock domain
- Gray-coded read and write pointers are passed to the clock domains to generate full and empty status flags



Source: Litterick, "Pragmatic Simulation-Based Verification of Clock Domain Crossing Signals and Jitter using System Verilog Assertions", DVCON 2006

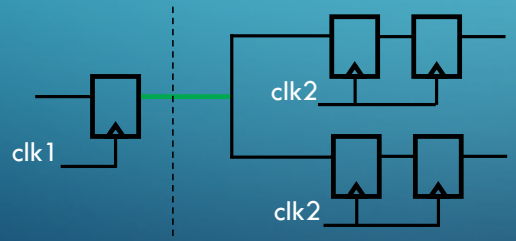
EXAMPLES OF POTENTIALLY FATAL CDC ERRORS

Loss of Data



Without protection, the output of the receiving flop could have a different value from the transmitting flop owing to metastability, resulting in loss of data

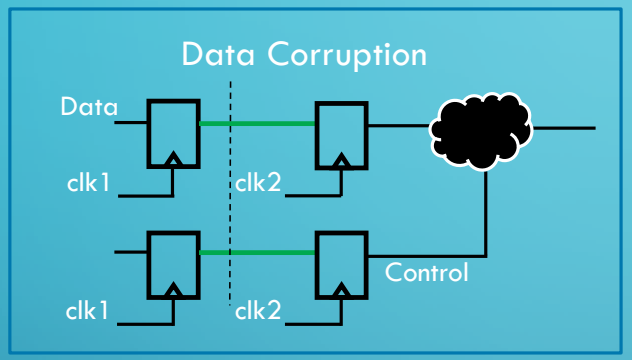
Controls Losing Correlation



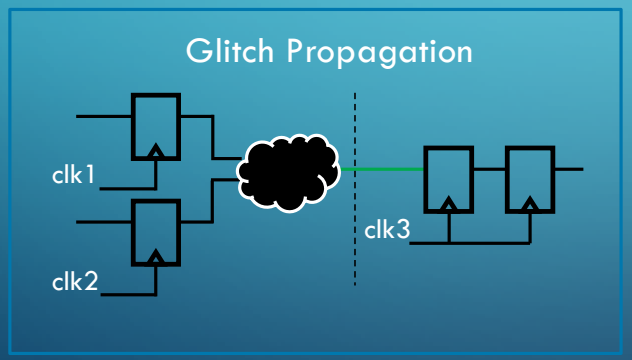
Because of the metastability effect, the outputs of the two synchronizer paths could have different values, resulting in loss of correlation

Source: Real Intent, DAC 2019

EXAMPLES OF POTENTIALLY FATAL CDC ERRORS



Without control-signal synchronization, metastability could propagate downstream, resulting in data corruption

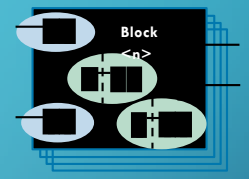


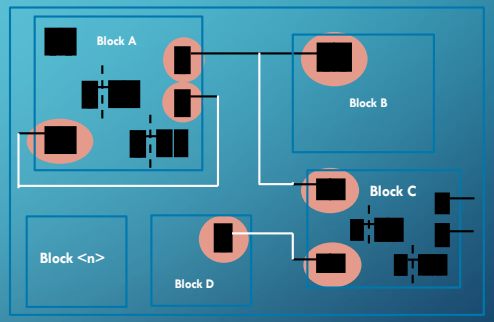
A timing glitch could be captured in the receiving domain, resulting in a downstream design failure

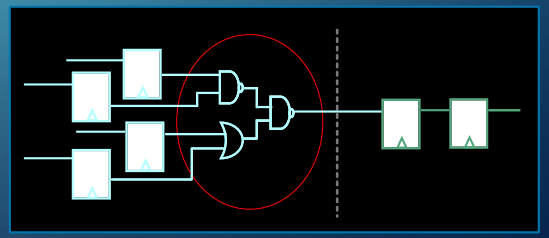
Source: Real Intent, DAC 2019

CDC VERIFICATION: WHERE AND WHAT

- Verify CDC Protocols
  - Ensure synchronizers are built correctly
  - Verify FIFO and handshake protocols
- Verify Block and Full Chip
  - Enforce CDC protocol usage
  - Identify all control and data crossings
  - Check for correctness of complex structures
  - Verify downstream logic is robust against metastability
- Verify Gate-Level Netlist
  - Ensure no CDC issues in implementation (such as no glitch introduced during synthesis)







Source: Real Intent, DAC 2019

CAN SIMULATION/FORMAL VERIFICATION ADD VALUE?

- Yes, once CDC crossings are identified, logic can be added to the simulation model to inject random delay to model metastability
- Formal can be used to verify handshake protocols
  1. Stability of data transfer across the crossing
  2. Only 1-bit changes in multi-bit control signals
  3. Glitches cannot happen in the logic that generates control signals
  4. Control signal should be wide enough for reliable data transfer

RESET (SYNCHRONOUS/ASYNCHRONOUS)

- Synchronous Reset:

```
module sync_resetFFstyle (
  output reg q,
  input      d, clk, rst_n);

  always @(posedge clk)
    if (!rst_n) q <= 1'b0;
    else        q <= d;
endmodule
```

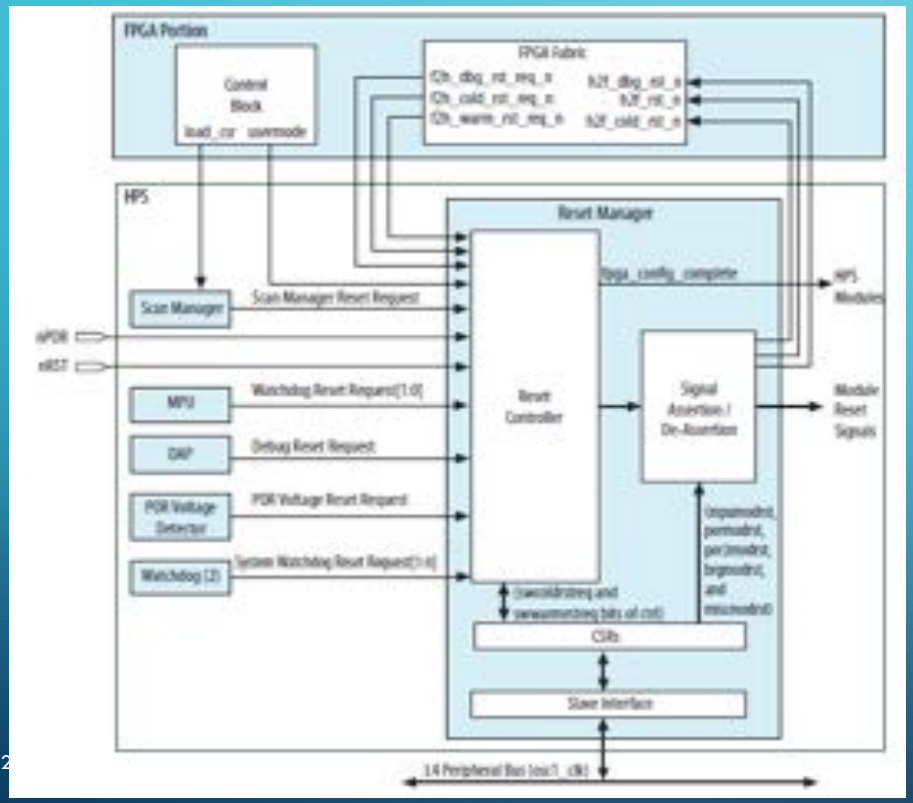
- Asynchronous Reset:

```
module async_resetFFstyle (
  output reg q,
  input      d, clk, rst_n);

  // Verilog-2001: permits comma-separation
  // @(posedge clk, negedge rst_n)
  always @(posedge clk or negedge rst_n)
    if (!rst_n) q <= 1'b0;
    else        q <= d;
endmodule
```

- Source: Cliff Cummings

EXAMPLE OF RESET COMPLEXITY



Source: Cyclone V HPS Technical Reference Manual for Altera Cyclone V

The reset manager generates module reset signals based on reset requests from the various sources.

- Supports handshake signals for system resets (e.g. are you ready for shutdown?)
- Enables software de-assertion of resets (e.g. after regs are configured)
- De-asserts resets synchronously and in groups in a defined sequence (e.g. clock manager, then …)

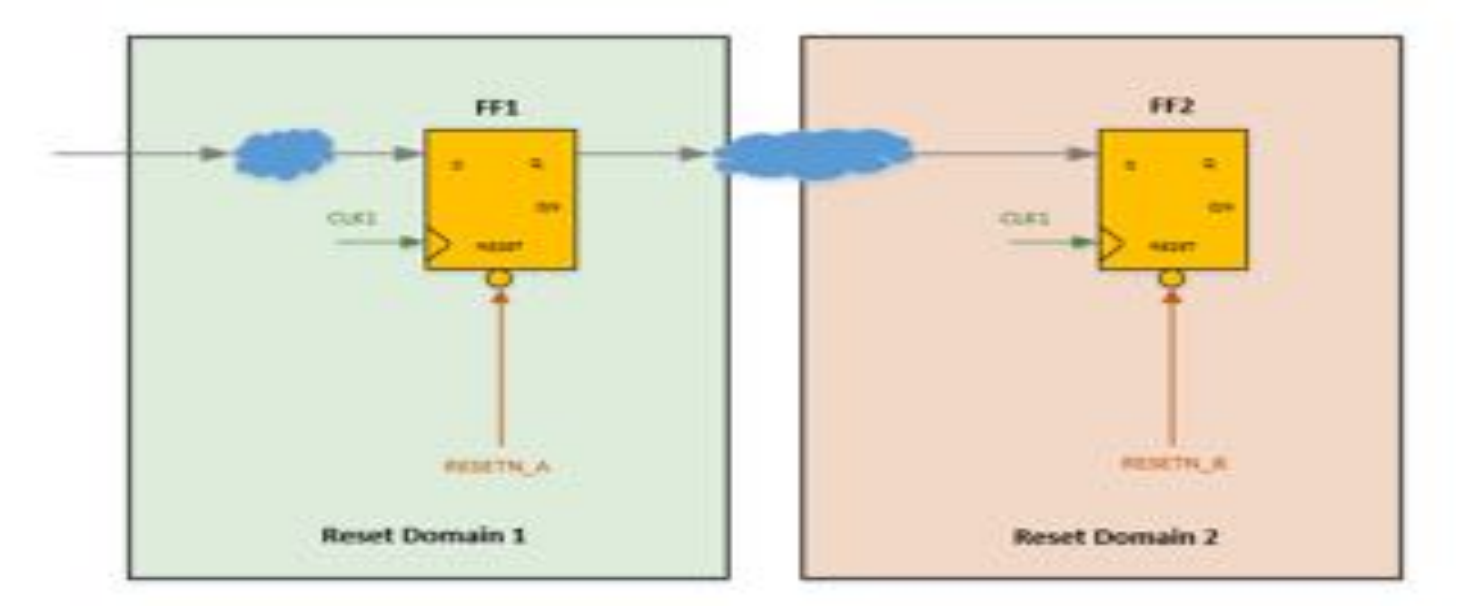
Three reset domains:

- JTAG test access port
- Debug
- System

Three types of reset

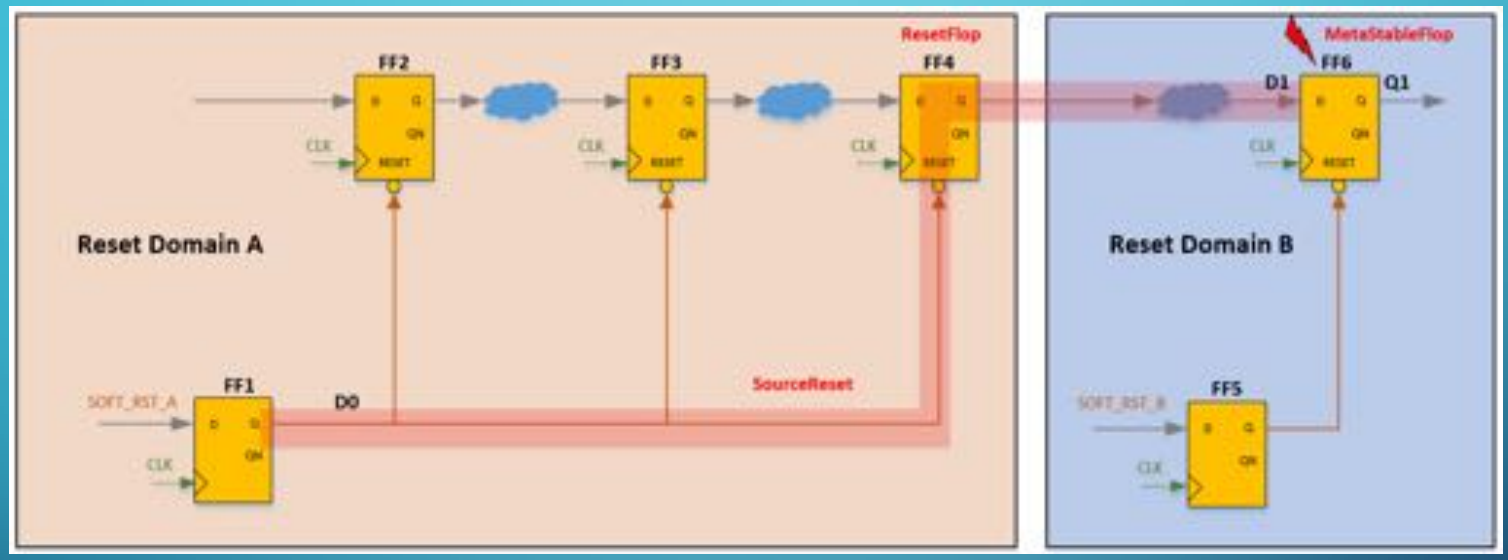
- Cold (power up)
- Warm (low power)
- Debug

WHAT IS A RESET DOMAIN CROSSING?



Reset Domain : Part of the design that can be reset independently of other such parts of the design (other reset domains).

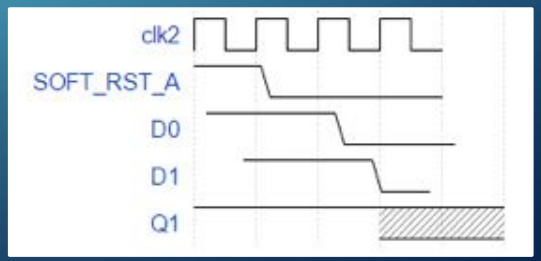
RESET METASTABILITY ACROSS RESET DOMAIN



STA constrains the normal timing paths

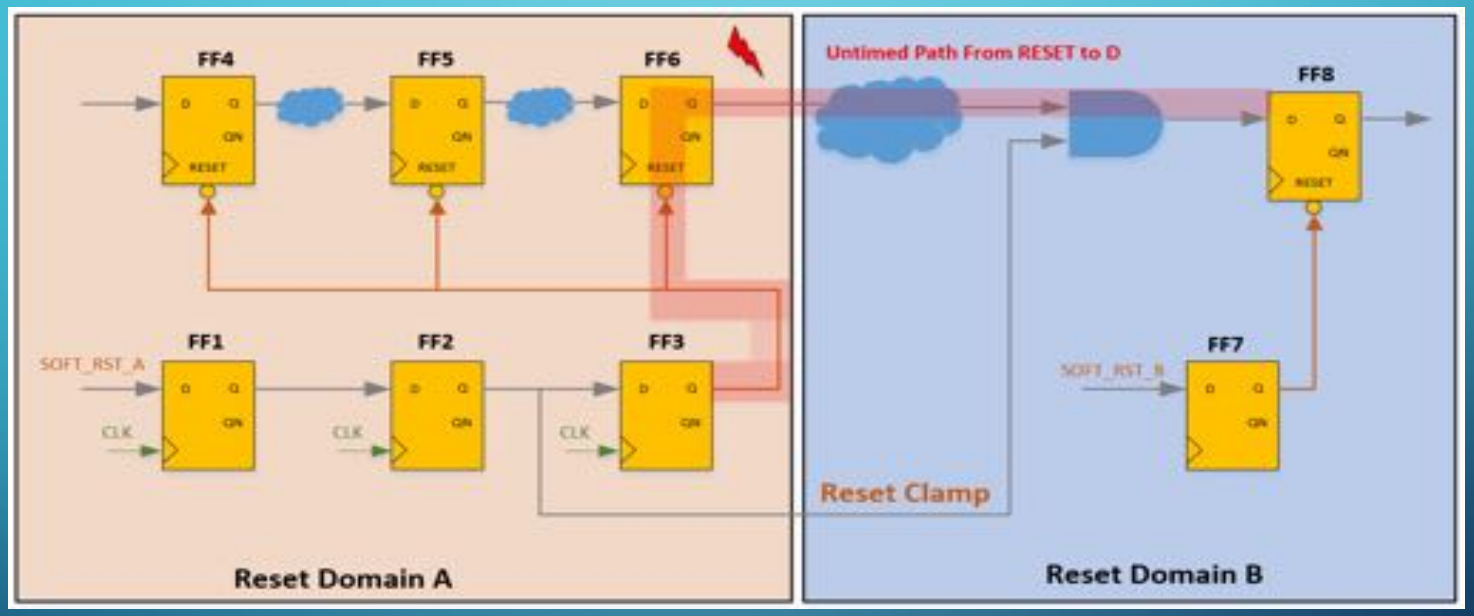
Assertion of SOFT_RST_A creates an untimed path;
Causes metastability on activation of SOFT_RST_A

Report metastability



- Source: Real Intent

HOW TO PREVENT RESET METASTABILITY



Safe reset path because Reset clamp controls untimed path

RDC VERIFICATION

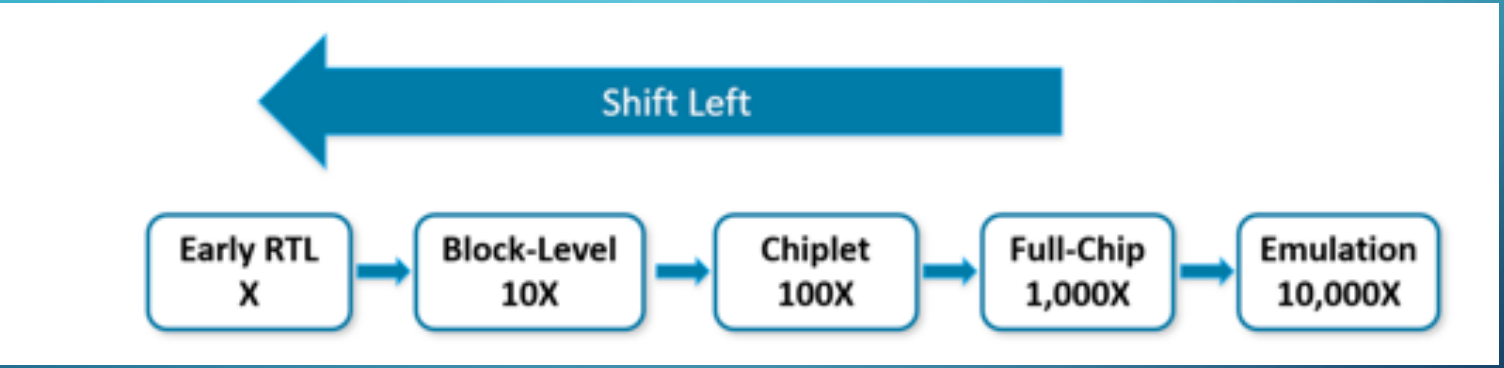
- Reset failures are becoming prevalent
  - Lack of awareness about severity and probability
  - Extremely difficult to debug in silicon
  - Multiple reset types and their interactions multiply risk
- None of the usual tools cover it
  - STA flows do not cover this and is impractical for all RDC paths
  - CDC tools do not cover RDC in the same clock domain
  - Impossible to address through non-scalable solutions
    - Formal verification and manual reviews
  - Impractical to get coverage with simulation
    - Assertions and metastability injection
- Specific solution needed to pinpoint unsafe paths

RDC VERIFICATION

- Identify Reset Domains and align with Functionality
- Define Reset groups and their sequence
- Define Reset Clamp cells, Reset synchronizers
- Define Constraints for more accurate analysis
- Perform Analysis to identify the reset domain crossings and verify that they are setup correctly

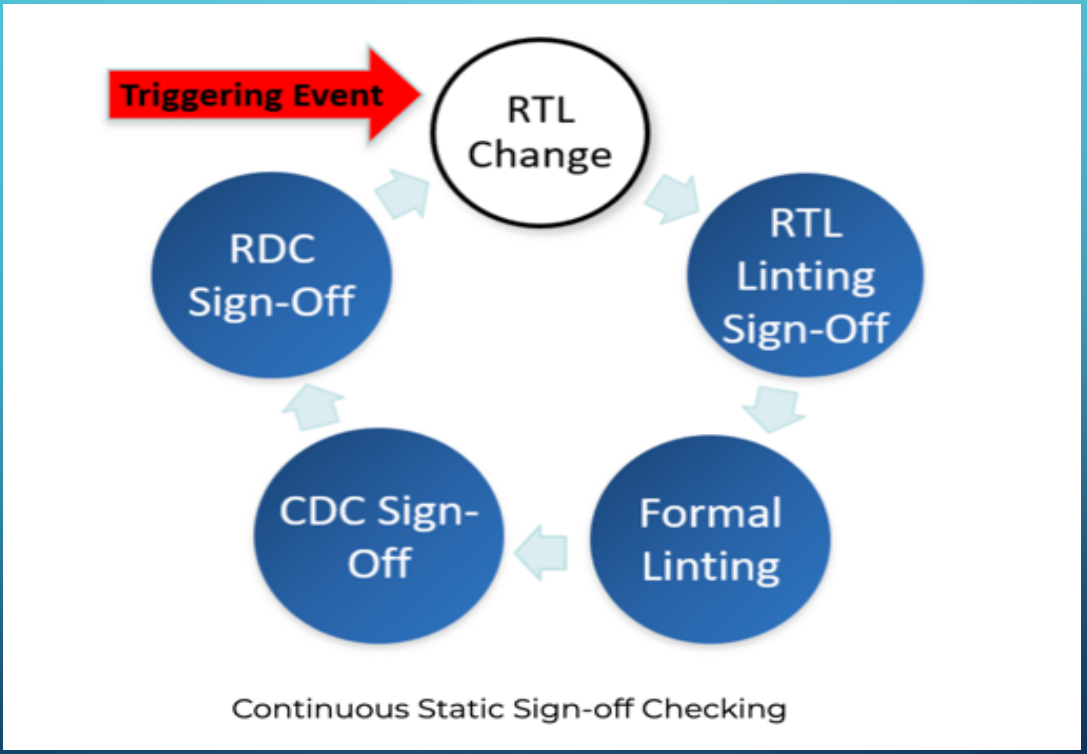
STATIC SIGN-OFF METHODOLOGY

- The earlier you can find and fix bugs, the more cost effective it is



- Start the static sign-off process as early as possible

CONTINUOUS STATIC SIGN-OFF



Continuous Static Sign-off Checking

COMMERCIAL SOLUTIONS FOR STATIC SIGN OFF

- Synopsys solutions
- Real Intent Solutions
- Cadence Solutions
- Mentor Graphics Solutions

THANK YOU!

QUESTIONS?