2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
OCTOBER 15-16, 2024

Comprehensive Glitch and Connectivity Sign-Off

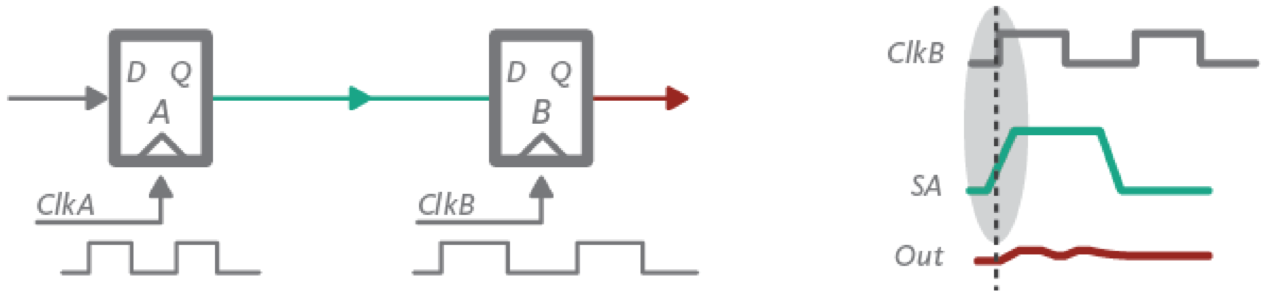
Vikas Sachdeva, Oren Katzir, Vardan Vardanyan

REAL INTENT

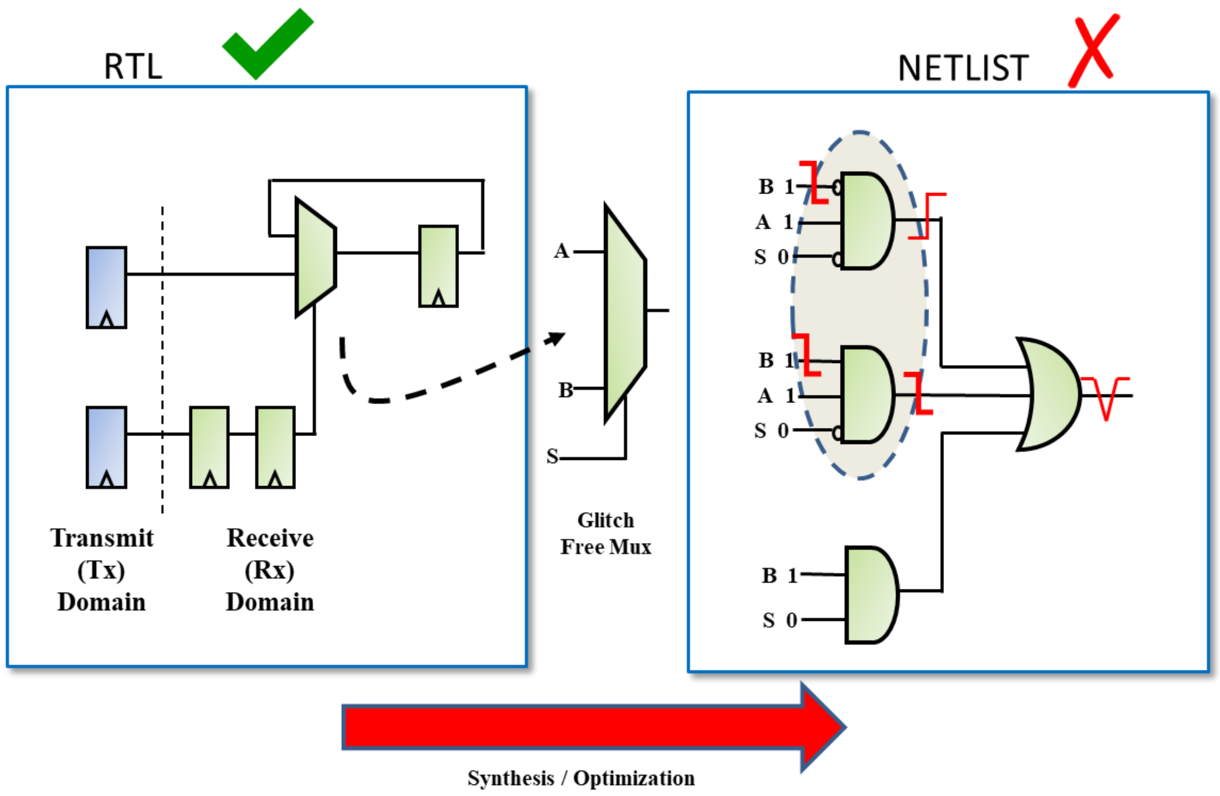
accellera
SYSTEMS INITIATIVE

Glitches

- Glitches occur on both synchronous and asynchronous paths in digital design
- On synchronous paths STA ensures that glitches are resolved and don’t cause design failures
  - Unless path is an exception in STA (False path, MCP, etc.)
- Asynchronous paths are not timed in STA and can cause problems if glitches are the

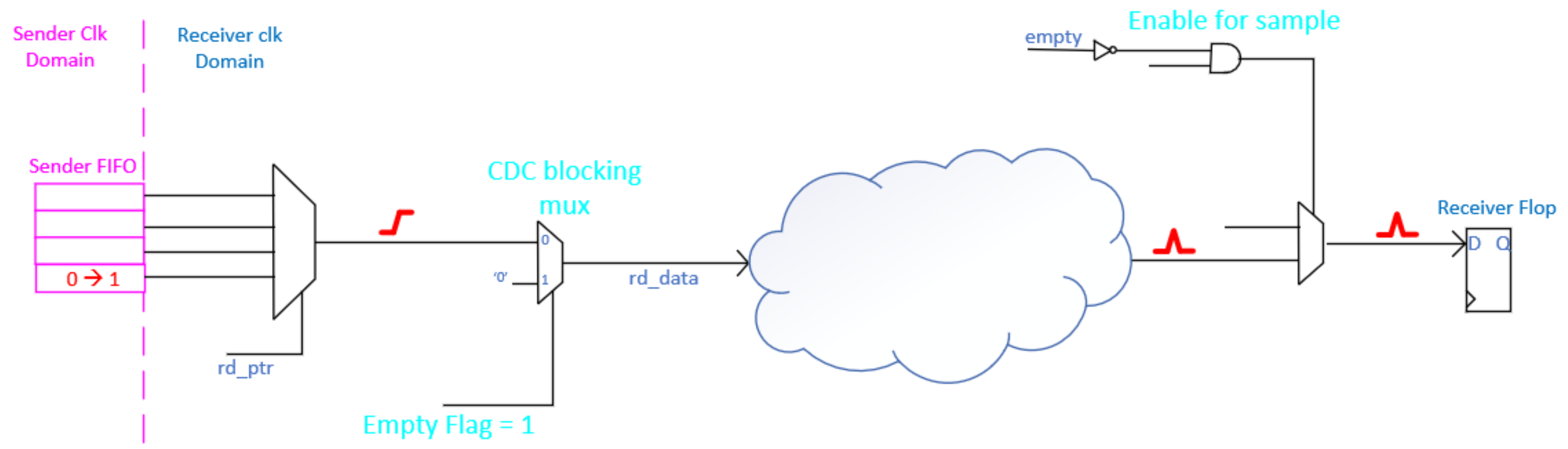


Glitches on CDC Path



Glitches on CDC Path

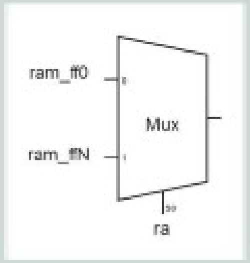
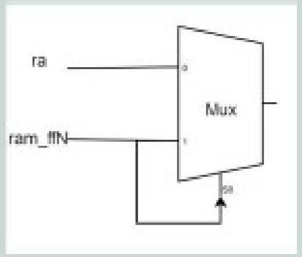
- Pre-Silicon SDF simulations pointed to a failure in synchronization FIFO logic
- Additional reviews showed that the RTL passed all CDC qualifications
  - RTL contains CDC (glitch) blocking logic. High level schematic below
- This led to the conclusion that the GL resulting from synthesis was faulty
  - Asynchronous glitches from sender clock domain were sampled in receiver clock domain



Glitches on CDC Path

- Synthesis implements a formally equivalent but glitchy implementation on CDC path

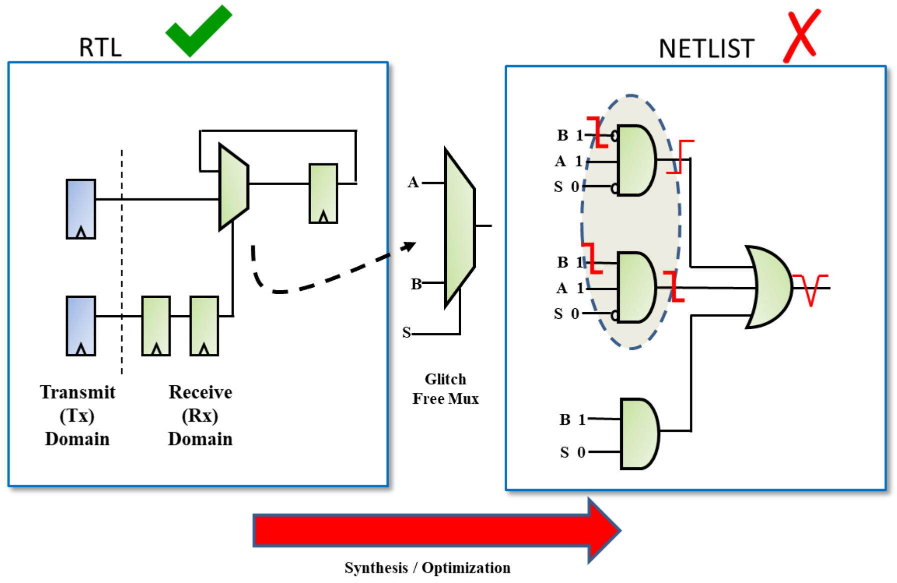
RTL	Expected Implementation	After Synthesis

always @(*) begin
    case(ra)
    8'd0:   dout = ram_ff0;
    8'd1:   dout = ram_ff1;
    ....
    8'dN: dout = ram_ffN;
    default: dout = {8{`x_or_0}};
    endcase
end
		

This can happen when case statement is not complete

RTL and Netlist Glitch Verification – Meridian CDC

- CDC Glitch Verification methodology
  - Uses structural, local formal and formal means to ensure no glitches in the design
  - At RTL users can specify set_dont_touch cells



Motivation: Glitch-Checking at RTL?

- Why Glitch-checking at RTL?
  - Most companies do the CDC sign-off at RTL
  - Would like to avoid CDC sign-off at netlist if possible
  - Glitches are the biggest risk of RTL-based CDC sign-off flows
- Aren't glitches typically introduced during synthesis?
  - Solution: Develop a strict methodology verifiable at RTL
  - All TX-RX paths should be unate
  - If multiple TX-es converge, they must converge in a multiplexor (single-path sensitization)
  - The multiplexor must be a glitch-free multiplexor that synthesis cannot modify

Glitch Checks at RTL

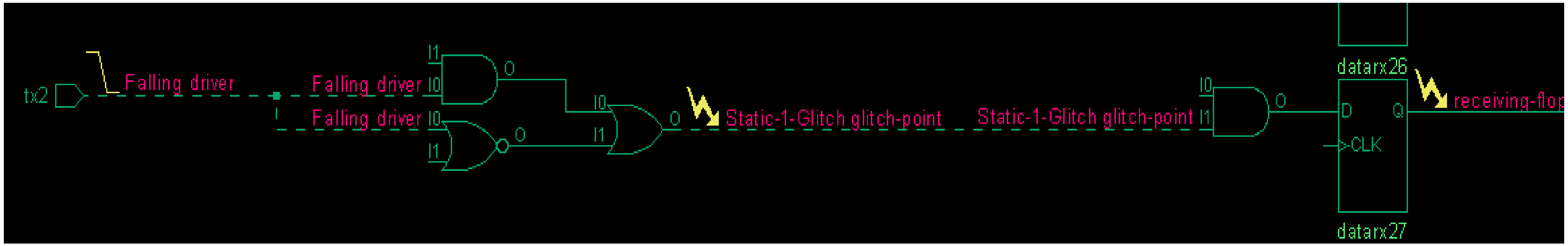
- SINGLE_TX_GLITCH_STRUCT: Every driver-receiver path is analyzed for glitch due to reconvergence of the driver with opposite polarity
- MULTIPLE_TX_NO_USER_MUX: For all receivers with multiple-drivers, the rule checks if any two drivers converge outside an user-specified multiplexer (MUX)
- GLITCH_FREE:
    - For crossings with single-drivers, all the paths from the driver to receiver must be glitch-free
    - For crossings with multiple-drivers, all the paths from each driver to receiver must be glitch-free. Moreover, if 2 drivers converge, the convergence-point has to be within an user-defined-mux

Glitch Checking at Netlist

- Why do Glitch-checking at Netlist?
  - Some companies are unable to execute strict methodology at RTL
  - Presence of external-IP that has not undergone the strict methodology checks

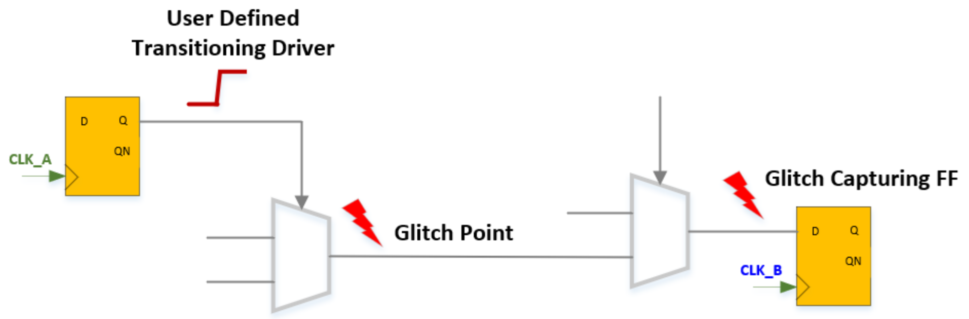
SINGLE_TX_GLITCH

- This check reports every driver-receiver pair that has a potential glitch on its path
- A path is glitchy if the driver re-convergences with opposite polarities in the path

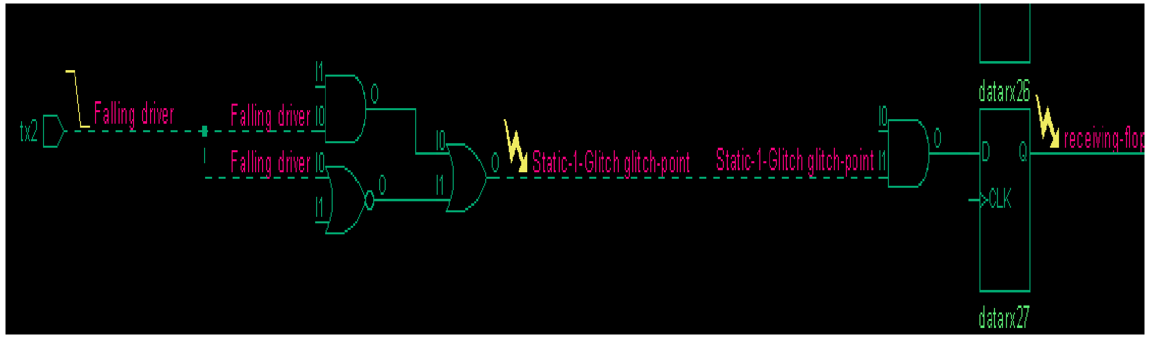


Glitch Errors on non-CDC paths

- Glitch = transition shorter than signal's clock period
- Errors: Untimed path glitches associated with user-specified, multicycle, & false paths
- Glitch can occur close to clock edge of receiving flop.
- Not caught by STA/Tcl scripts or simulation

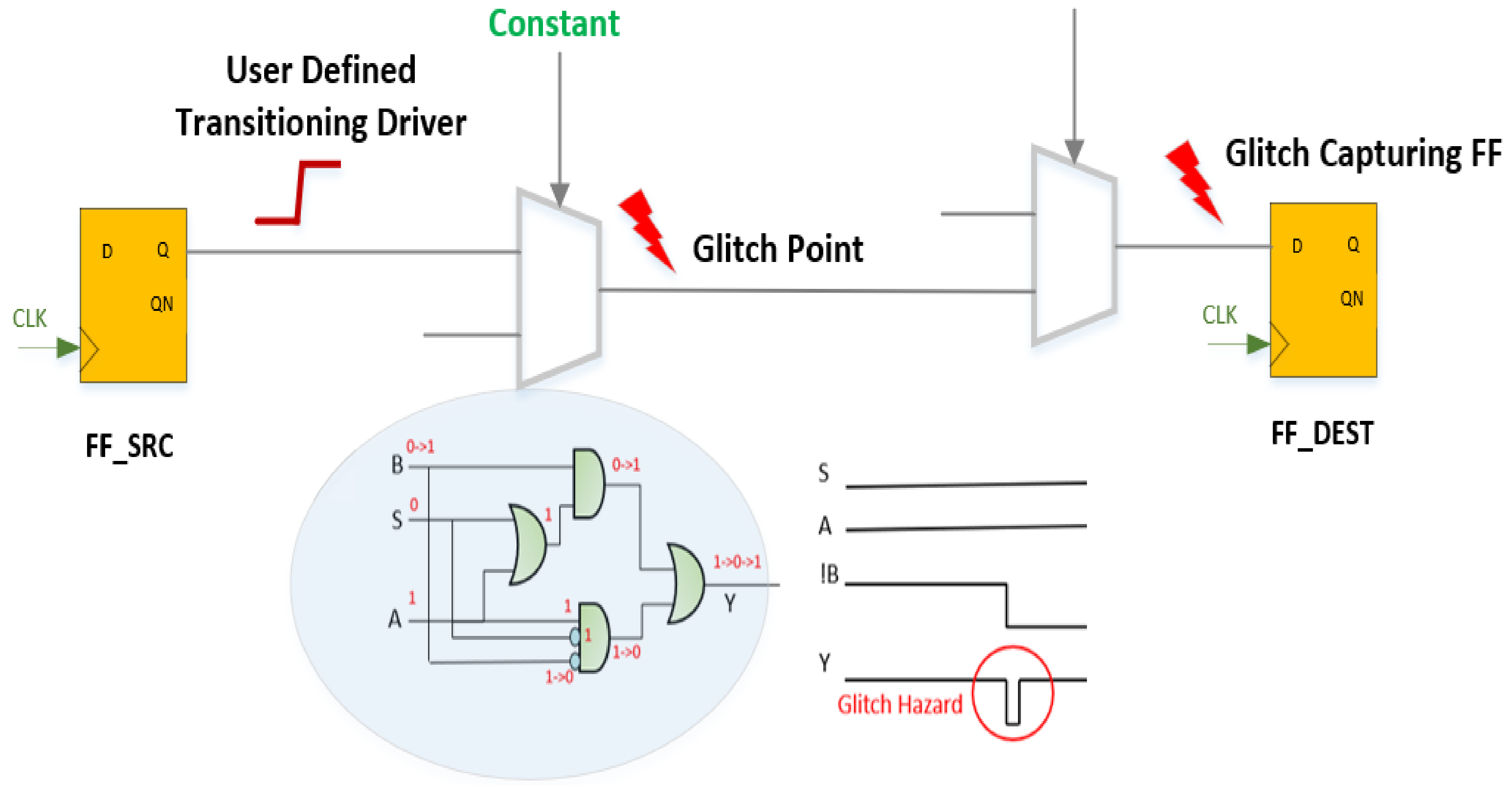


Glitch on a protected path can cause the chip to fail



Async mem Enable Flop Wrong connection + Glitch

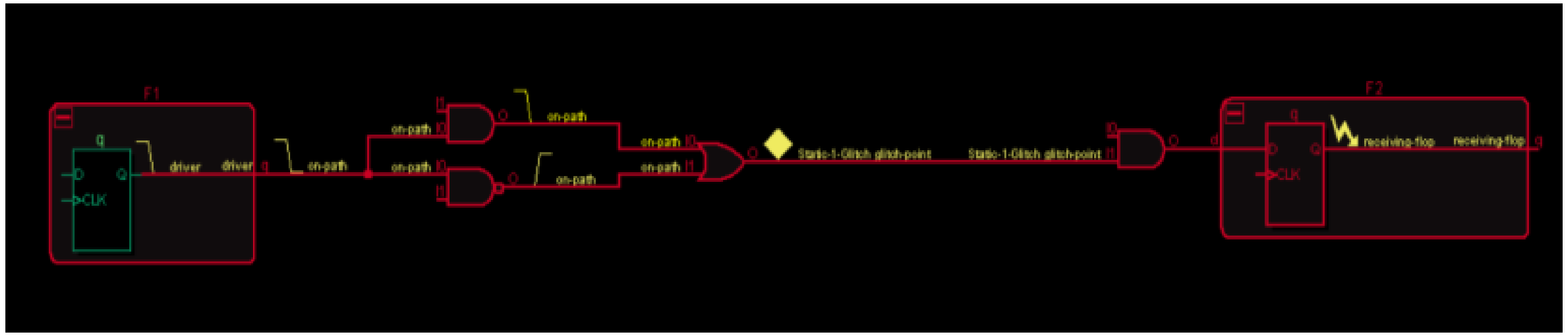
Glitch on Multicycle path



Check Glitches – Single Driver / Multi-Driver

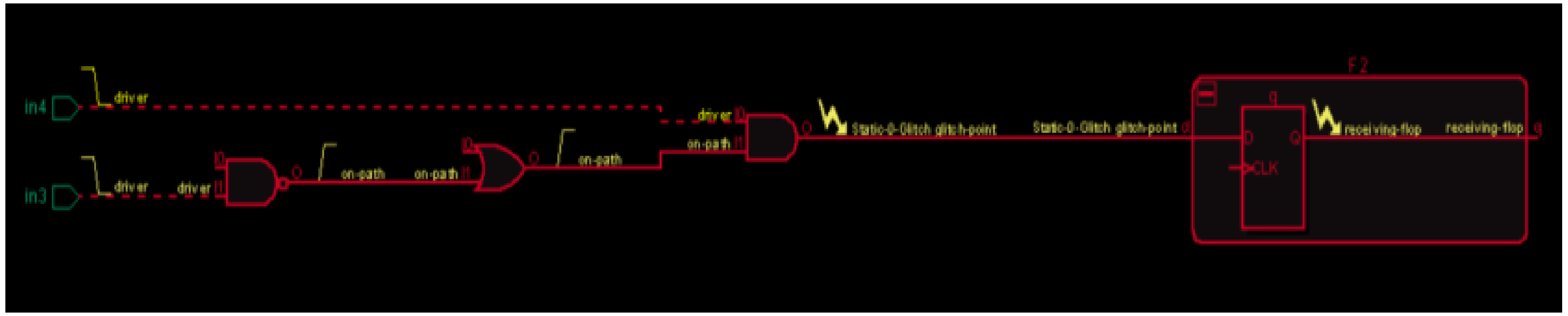
Check if a single driver can cause a glitch at the receiver

```
set_no_glitch -rule R5 -from F1.q -formal -output glitch.rpt
```



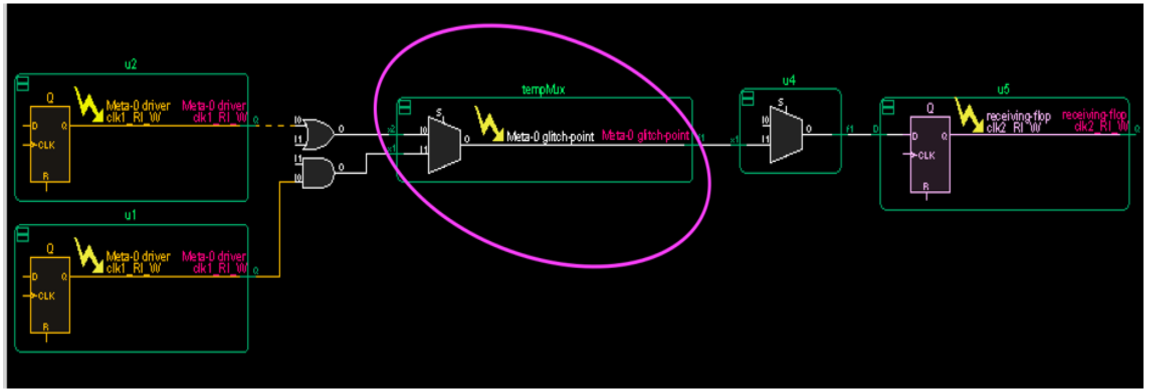
Check if multiple drivers can cause a glitch at the receiver

```
set_no_glitch -rule R4 -to F2.q -formal -output glitch.rpt
```



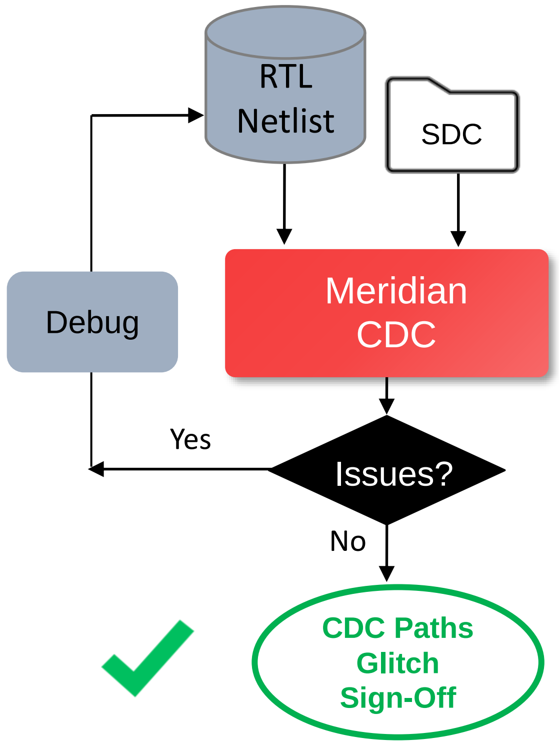
Glitch sign-off – IP level, Chip level

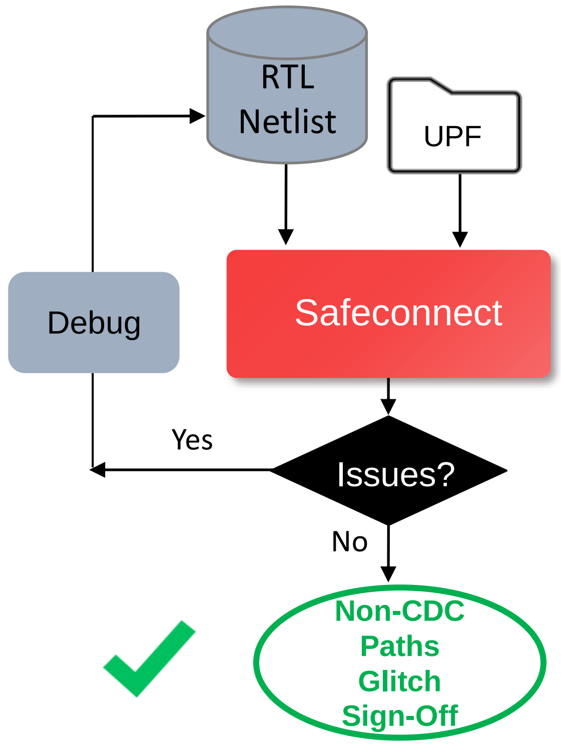
- On untimed paths, Glitch can be fatal
- Numerous companies had *late-stage* netlist-glitch failures
  - IP vendor provided glitchy-IP (@outputs) to customer
  - Automotive chip had glitch-potential, designers were unaware
  - Memory-controller chip went through multiple ECOs because of glitch failures



Glitch Detected on path to Analog IP

Glitch Sign-off Flow





Key Challenges SoC and IP Engineers Face

- Compatibility and Integration
  - Ensuring that IP cores are compatible with various target SoCs
  - Seamlessly integrated into different designs.
  - SoCs often incorporate multiple IP blocks (e.g., CPU, GPU, memory controllers) from different sources.
  - Requires meticulous integration and verification efforts

Key Challenges SoC and IP Engineers Face

- Interconnect Design
  - Efficient communication between the components of an SoC is crucial for optimal performance.
  - Designing high-speed interconnects that deliver the required bandwidth while minimizing latency and power consumption

Key Challenges SoC and IP Engineers Face

- Security
  - SoCs, especially those in connected devices, must be fortified against a range of security threats.
  - Incorporating robust security features like
    - secure boot
    - encryption
    - hardware root of trust,
  - Maintaining minimal performance overhead

Key Challenges SoC and IP Engineers Face

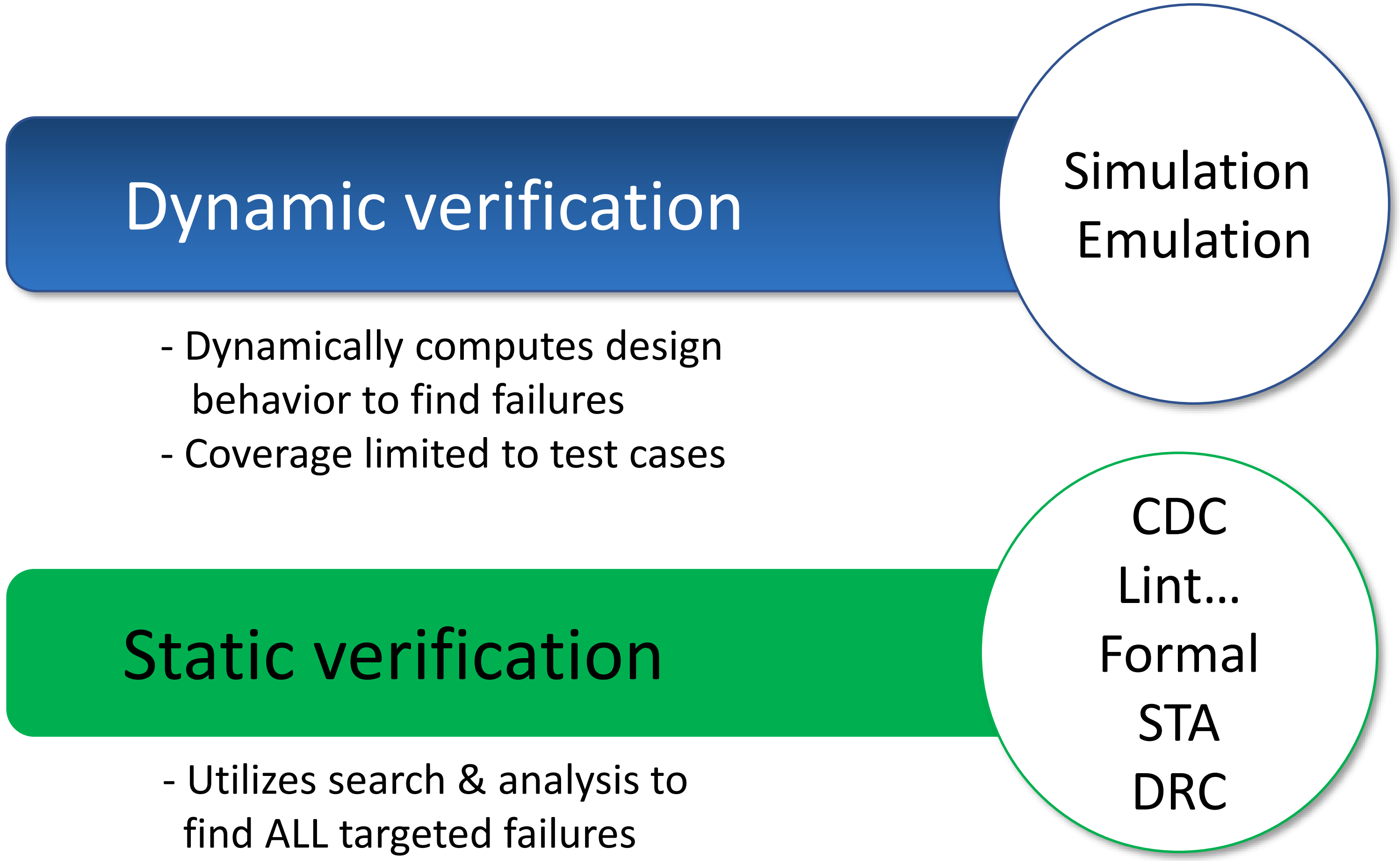
- Scalability and Reusability
  - To reduce time-to-market and development costs,
  - Reuse of IP blocks and scalability.
  - Requires careful planning and a modular design approach.

2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
10 YEAR ANNIVERSARY

STATIC SIGN-OFF METHDOLOGY

accellera
SYSTEMS INITIATIVE

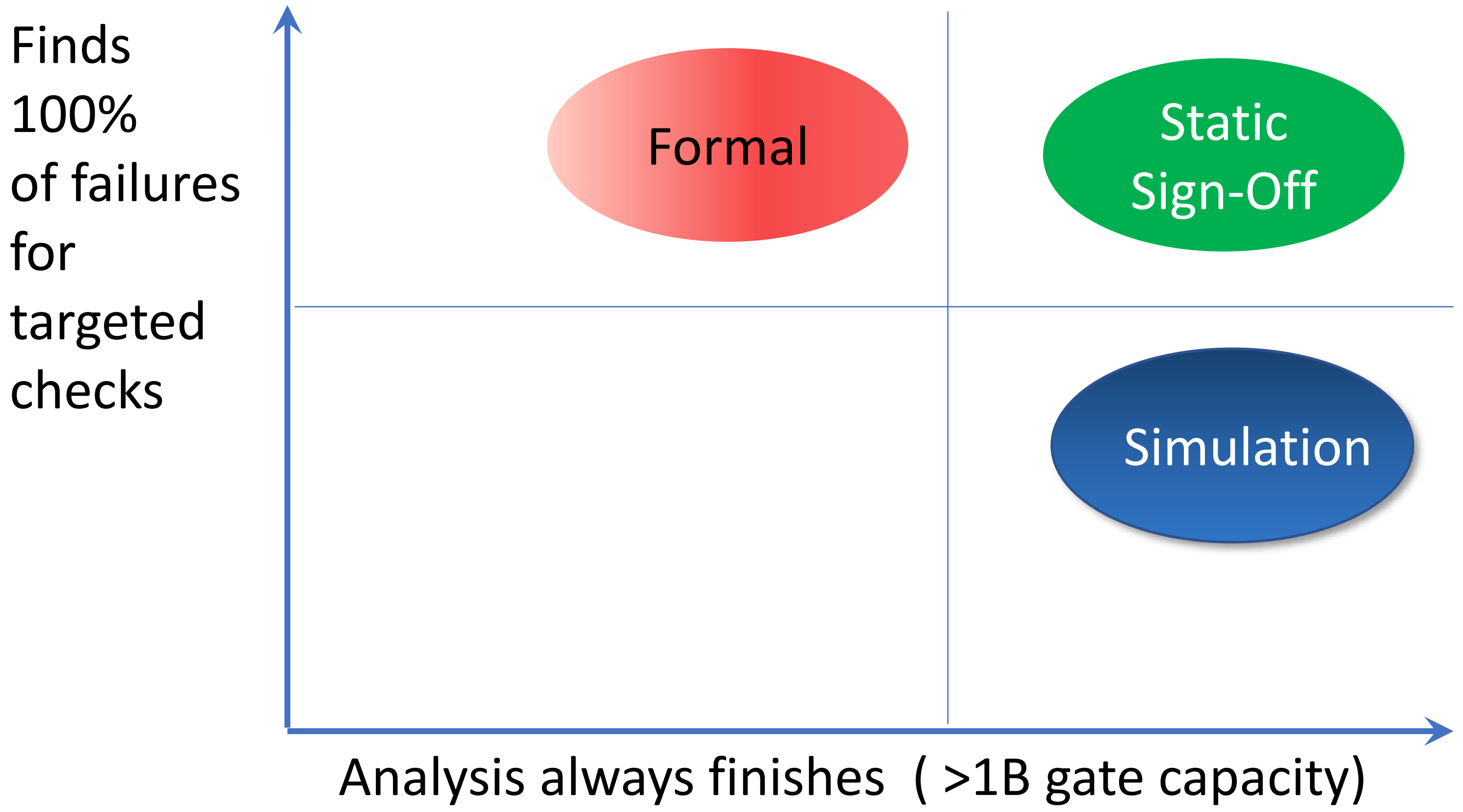
Static Verification vs Dynamic Verification



- Dynamically computes design behavior to find failures
- Coverage limited to test cases

- Utilizes search & analysis to find ALL targeted failures
- Test cases not required

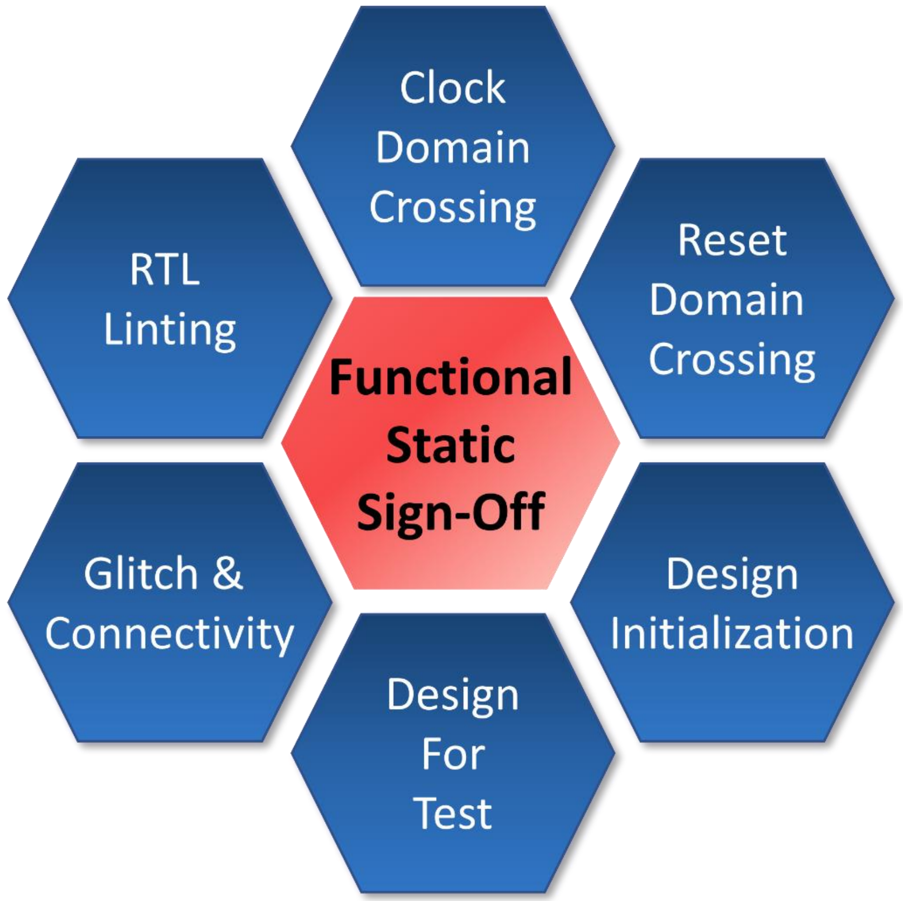
Static Sign-off vs Formal & Simulation


Finds 100% of failures for targeted checks
Formal
Static Sign-Off
Simulation
Analysis always finishes (>1B gate capacity)

Functional Static Sign-Off Expanding Applications

Functional static sign-off began with
RTL Linting
& CDC

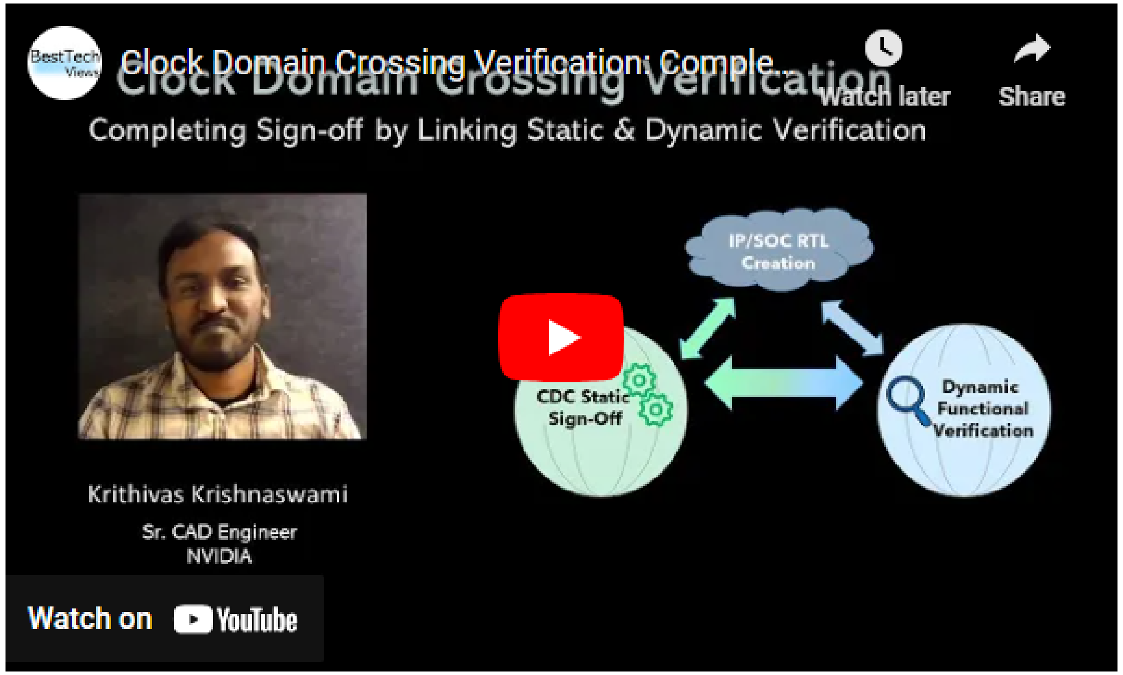
The target applications continuously expand



Clock Domain Crossing Verification — Completing sign-off by linking static & dynamic verification

Meridian CDC

Case Study By Krithivas Krishnaswami of **NVIDIA**


Clock Domain Crossing Verification: Comple...
Clock Domain Crossing Verification
Completing Sign-off by Linking Static & Dynamic Verification
Watch later
Share
IP/SOC RTL Creation
CDC Static Sign-Off
Dynamic Functional Verification
Krithivas Krishnaswami
Sr. CAD Engineer
NVIDIA
Watch on YouTube

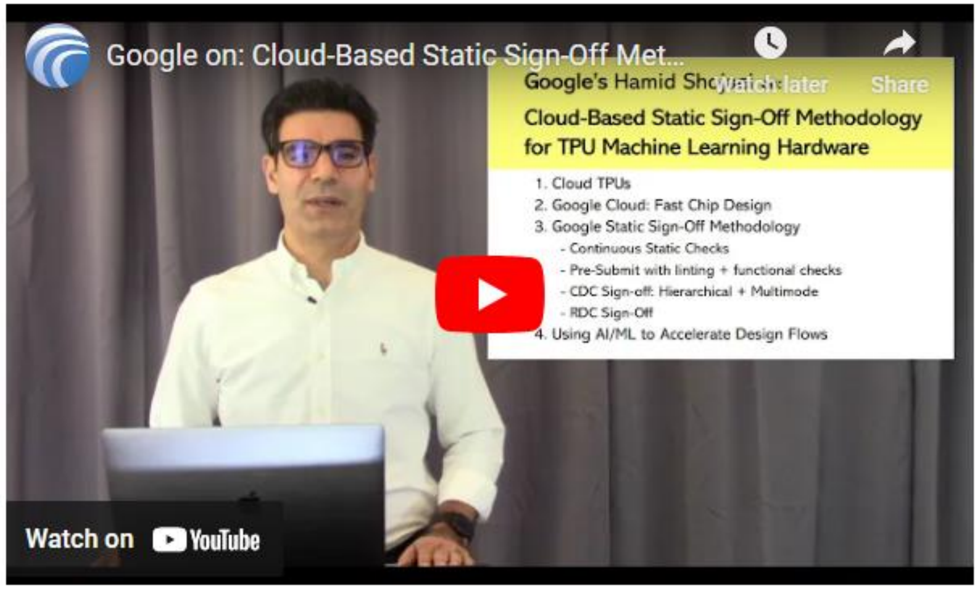
Case Study Overview

Krithivas Krishnaswami discusses NVIDIA's successful evaluation of a methodology for completing clock domain crossing verification by linkings CDC static sign-off and simulation. Real Intent's Meridian CDC and the Meridian CDC Simportal feature were deployed.

Edited transcript and graphics.

Cloud-Based Static Sign-Off Methodology for TPU Machine Learning Hardware

Presented by Hamid Shojaei
Google


Google on: Cloud-Based Static Sign-Off Met...

Google's Hamid Shojaei

Cloud-Based Static Sign-Off Methodology for TPU Machine Learning Hardware

1. Cloud TPUs
2. Google Cloud: Fast Chip Design
3. Google Static Sign-Off Methodology
   - Continuous Static Checks
   - Pre-Submit with linting + functional checks
   - CDC Sign-off: Hierarchical + Multimode
   - RDC Sign-Off
4. Using AI/ML to Accelerate Design Flows

Cloud TPU: Empowering EDA with Google Cloud AI

DAC 2021 presentation by Hamid Shojaei of Google (edited transcript)

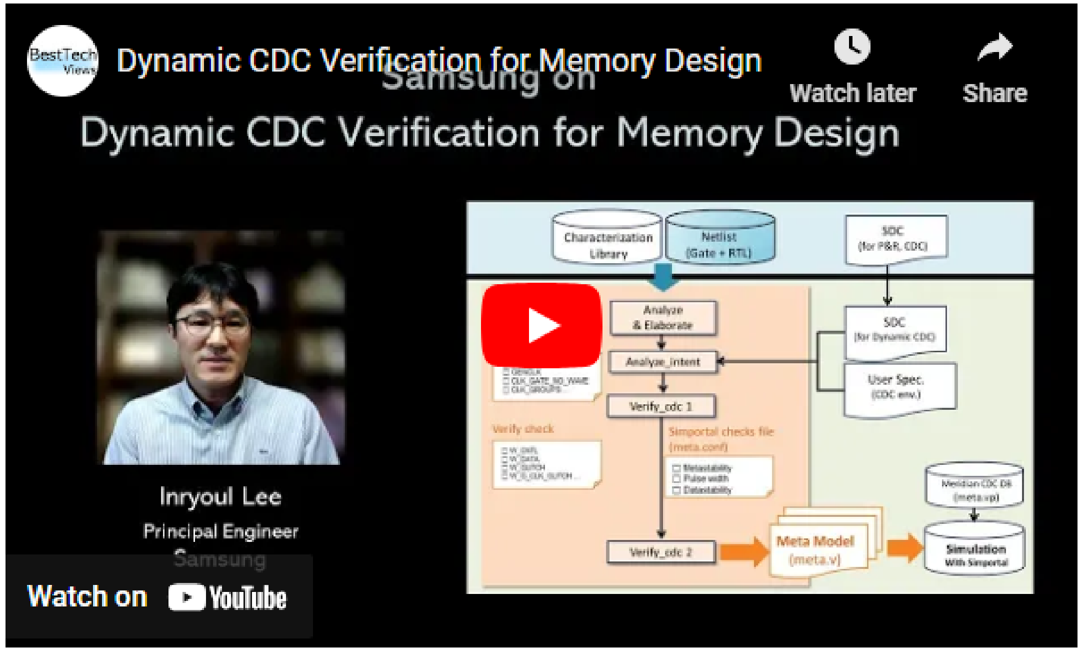
Case Study Overview

Hamid Shojaei of Google presents a case study on Google's cloud-based static sign-off methodology that includes pre-submit with RTL Linting, Single mode & Multimode clock domain crossing & Reset domain crossing. (Real Intent tools deployed)

Dynamic CDC Verification for Memory Design

Meridian CDC

Case Study by Inryoul Lee, Principal Engineer of **Samsung**



Case Study Overview

Inryoul Lee discusses how Samsung implemented their advanced dynamic clock domain crossing verification methodology for memory design using Real Intent Meridian CDC Simportal, closing the sign-off gap between CDC static sign-off and simulation. Slightly edited transcript below.

Clock Domain Crossing: Constraints-Based Sign-Off Methodology

By Sharan Mohan, Pinkesh Shah, & Rambabu Singampalli
Western Digital

Meridian CDC

I. Case Study Executive Overview

This case study covers Western Digital's enhanced constraint-driven clock domain crossing (CDC) sign-off methodology with Real Intent Meridian CDC, as presented at the 2021 Design Automation Conference.

Western Digital's enhanced methodology achieved a two-thirds to a three-quarters reduction in total CDC sign-off time, achieving sign-off in only two weeks, compared with a typical six to eight weeks.

II. Problem Statement & Goal

ASIC respins are expensive in terms of cost and delivery impact — good specifications, randomized simulations, and thorough verification are critical.

Functional and clocking issues rank highest among bug escapes sources; clock domain crossing issues are typically a mix of clocking and functional bugs. System on chips (SoCs) have multiple asynchronous clock domains with complex interactions; additionally, they contain several IPs from third parties, each with different configurations.

Given this complexity level, a waiver-based methodology to clean up CDC violations can be risky and lead to silicon failures.

Goal: Western Digital's goal was to implement an accurate, correct-by-construction clock domain crossing sign-off methodology for its SoCs while reducing its engineering effort.

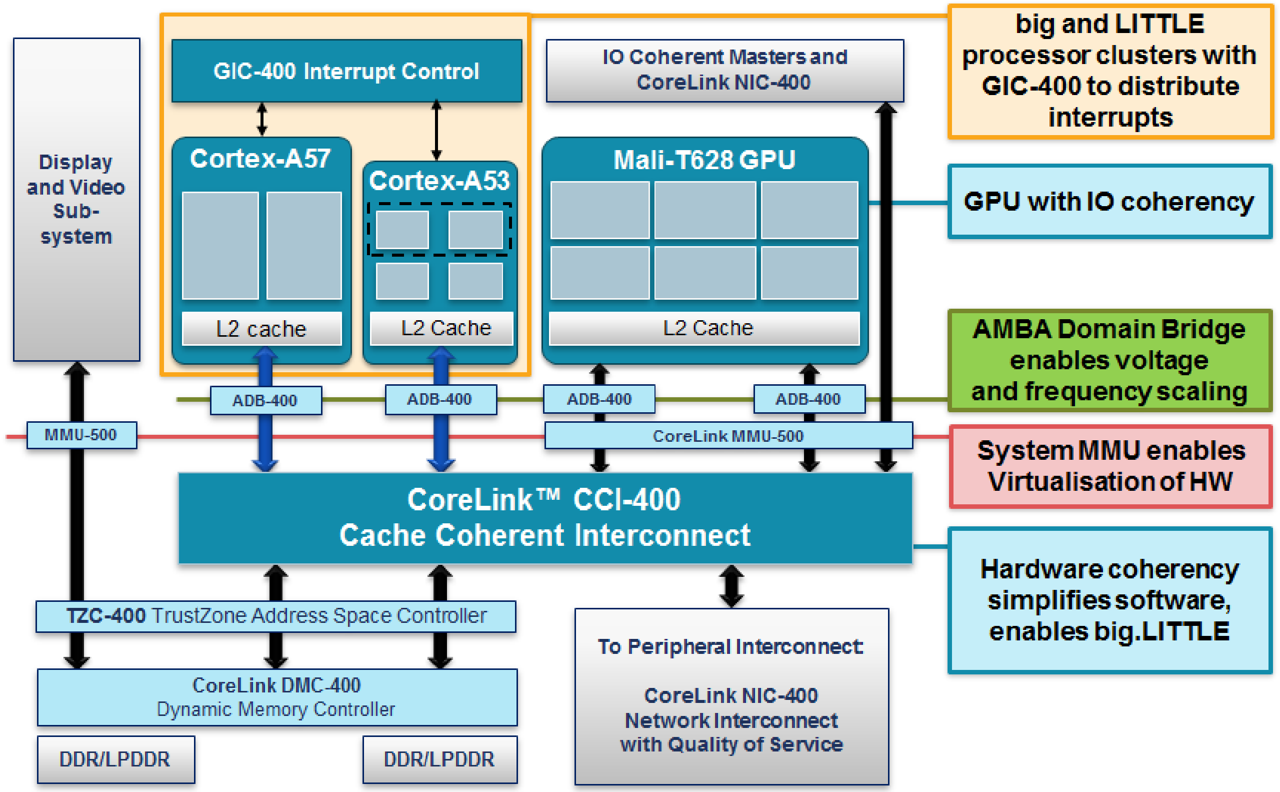
2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Compatibility and integration

accellera SYSTEMS INITIATIVE

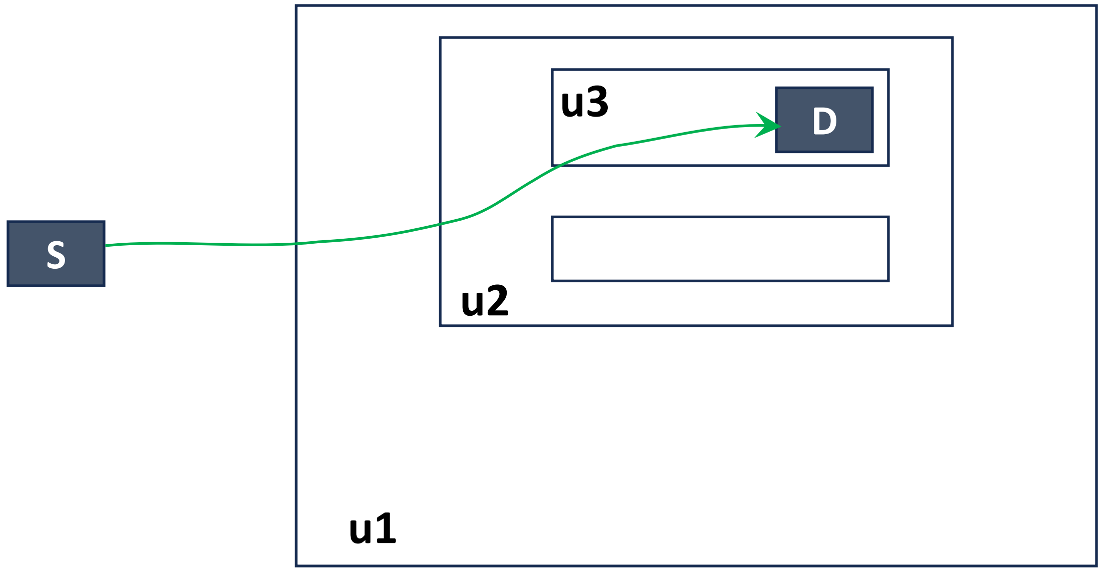
Inside a modern SoC

- Ensure IP Cores are compatible with various target SoCs
- **Complete verification of connections at SoC level**
  - Fundamental requirement to ensure correct operation
- Requires meticulous integration and verification effort


Display and Video Sub-system
GIC-400 Interrupt Control
IO Coherent Masters and CoreLink NIC-400
big and LITTLE processor clusters with GIC-400 to distribute interrupts
Cortex-A57
L2 cache
Cortex-A53
L2 Cache
Mali-T628 GPU
L2 Cache
GPU with IO coherency
AMBA Domain Bridge enables voltage and frequency scaling
ADB-400
ADB-400
ADB-400
ADB-400
MMU-500
CoreLink MMU-500
System MMU enables Virtualisation of HW
CoreLink™ CCI-400 Cache Coherent Interconnect
Hardware coherency simplifies software, enables big.LITTLE
TZC-400 TrustZone Address Space Controller
CoreLink DMC-400 Dynamic Memory Controller
DDR/LPDDR
DDR/LPDDR
To Peripheral Interconnect
CoreLink NIC-400 Network Interconnect with Quality of Service

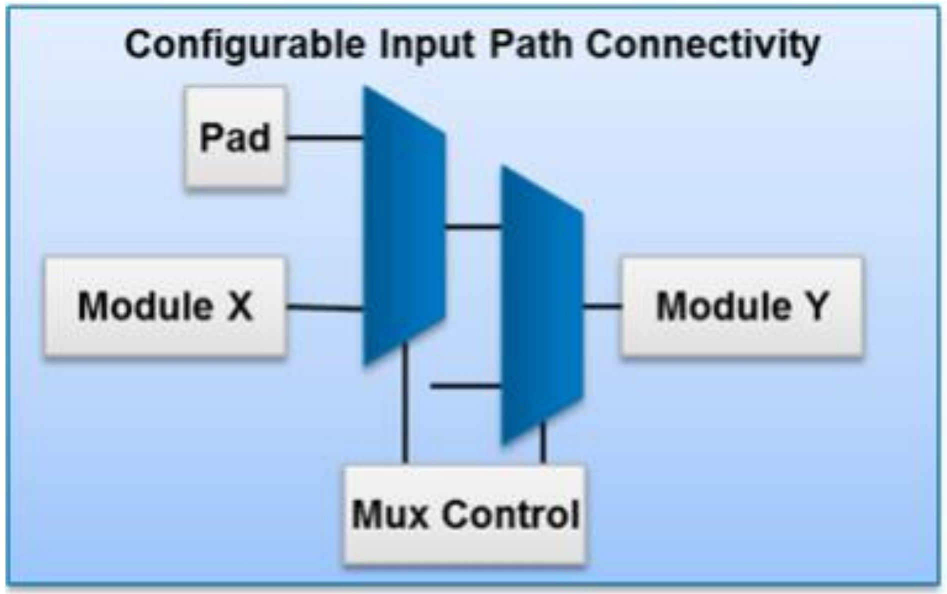
Connectivity Challenges

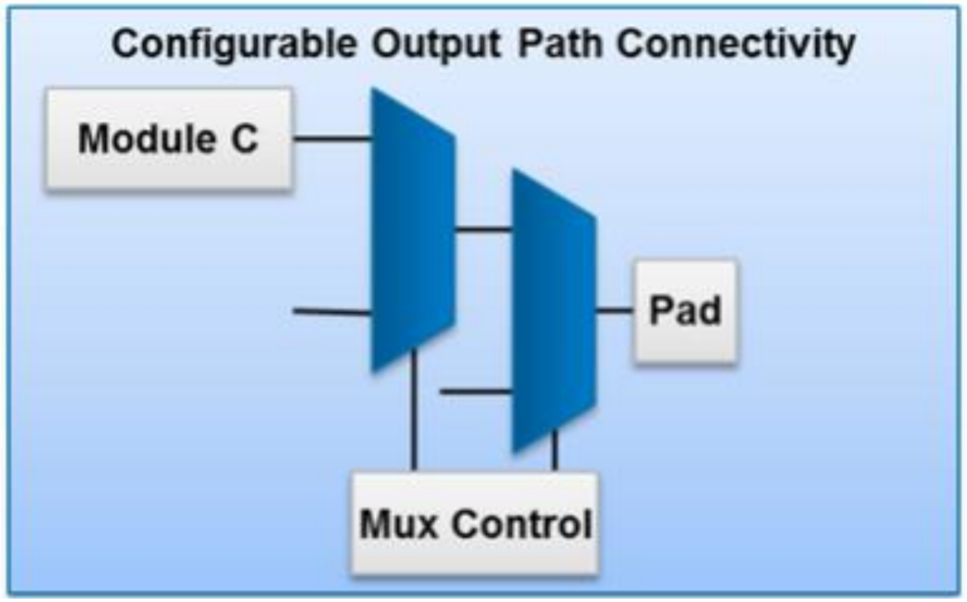
- Source to Destination connection
  - Connected through multiple blocks
  - Across different hierarchies



Connectivity Challenges

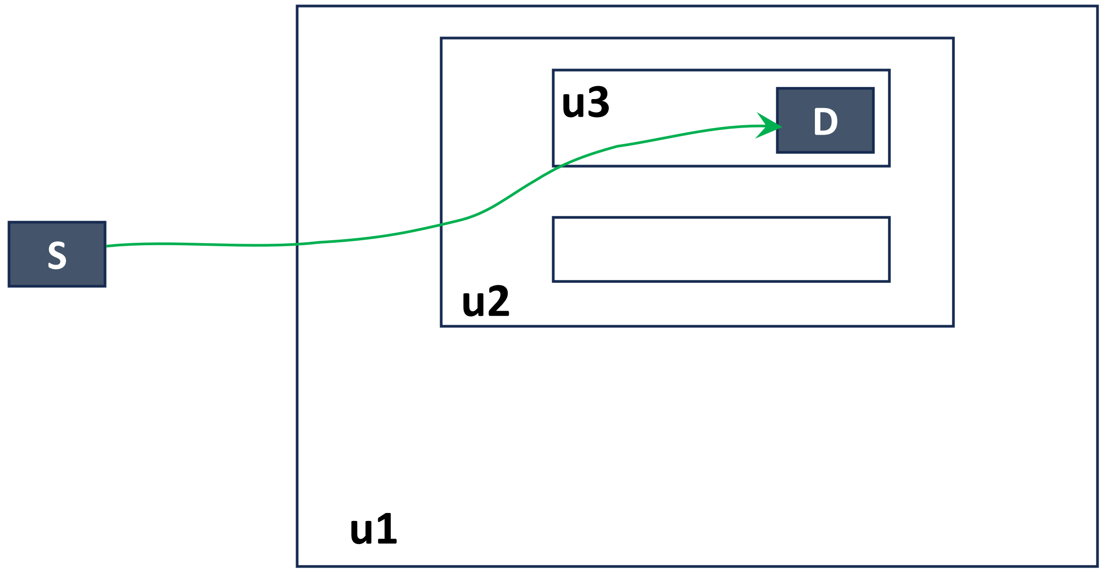
- Source to Destination connection
  - Based upon conditions





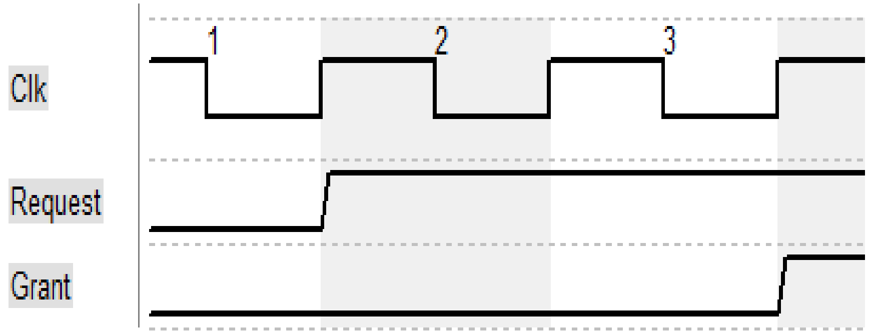
Connectivity Challenges

- Source to Destination connection
  - Only buffers/inverters allowed
  - Any logic allowed
  - Polarity important along the path

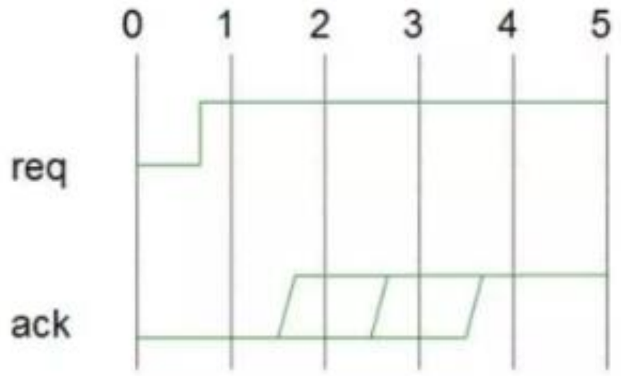


Connectivity Challenges

- Source to Destination connection
  - Can be sequential
  - Precise number of cycles
  - Or Range



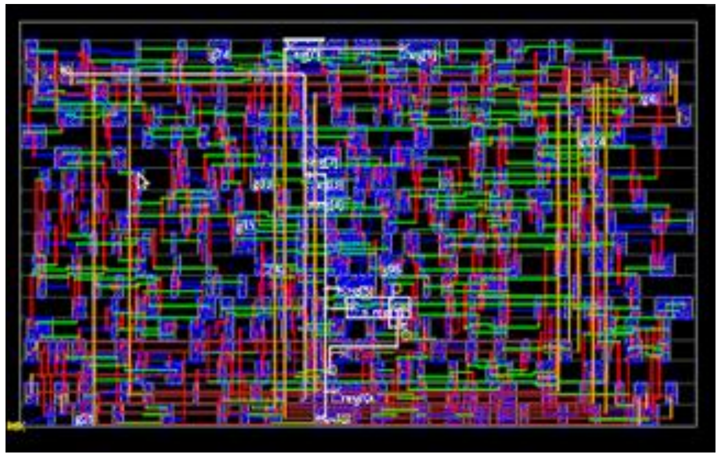
Grant should follow 2 cycles after request is asserted

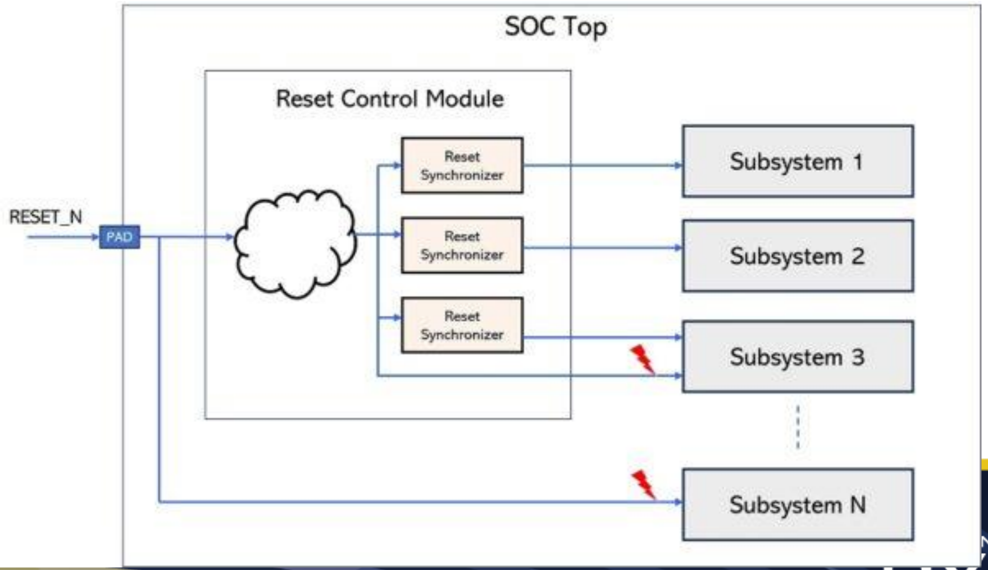


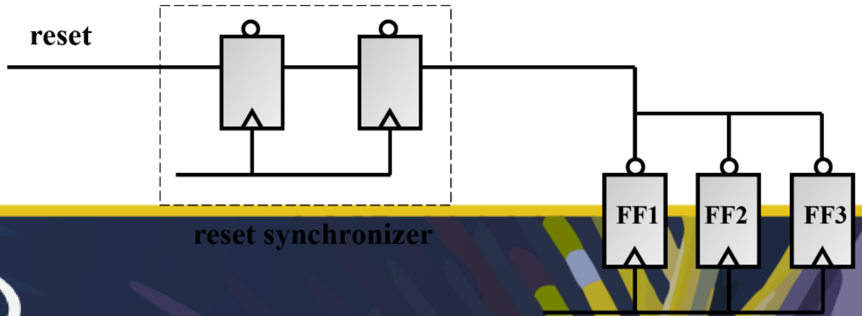
After the rise of request signal, the acknowledge signal should be asserted no later than 3 clocks cycles.

Connectivity Challenges

- Reset, Clock and Global signals
  - Connected to all the flops in the design
  - Millions of paths
  - Polarity is important
  - Can propagate through specialized cells
    - Reset synchronizers
    - Clock dividers, glitch free sequential muxes

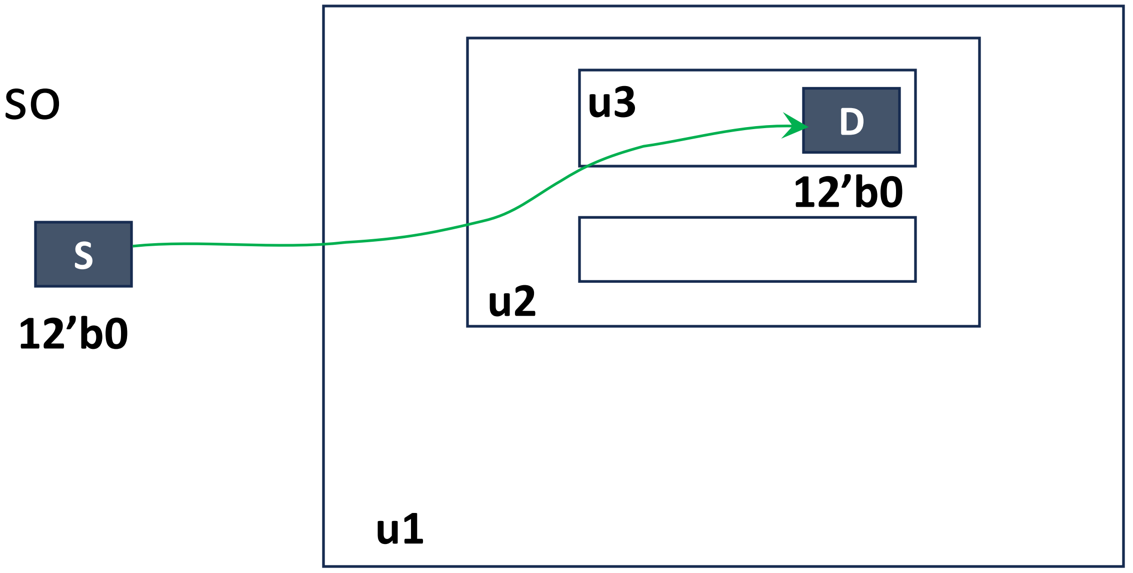






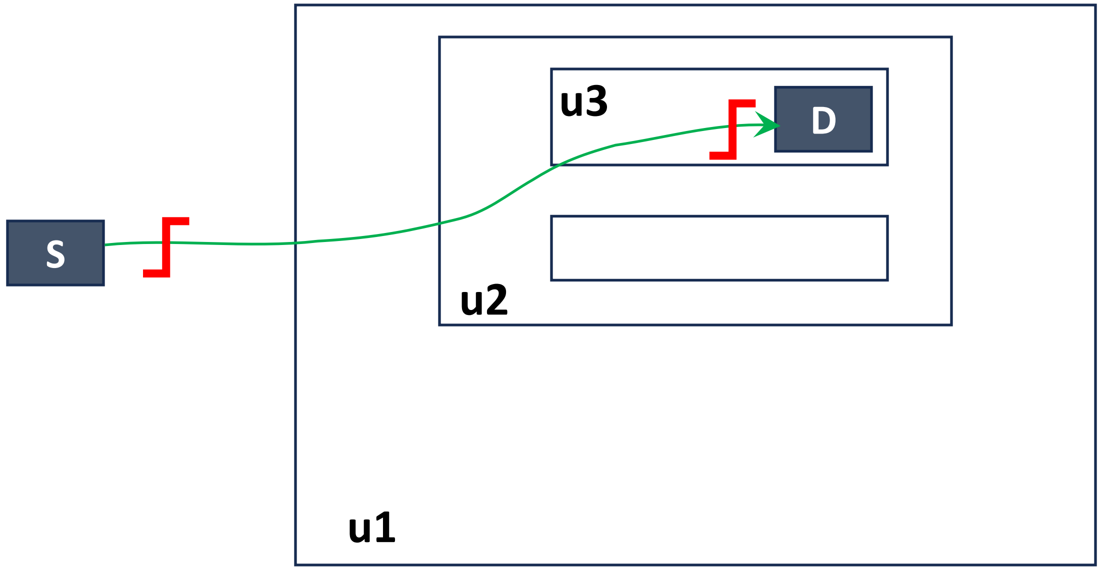
Connectivity Challenges

- System Registers, Configuration registers
  - Specific connectivity register based
  - Can propagate through sequential
  - Not just connection, value propagation also important



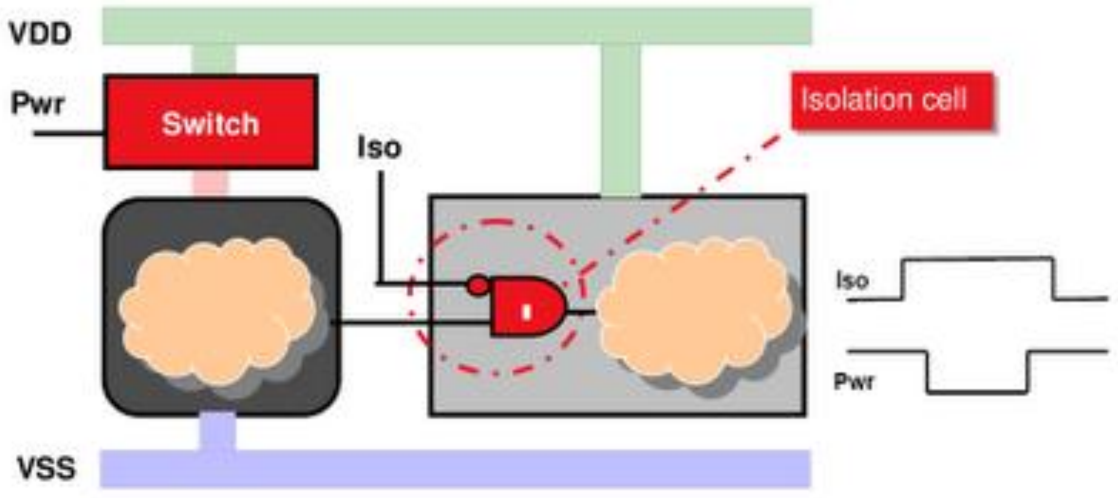
Connectivity Challenges

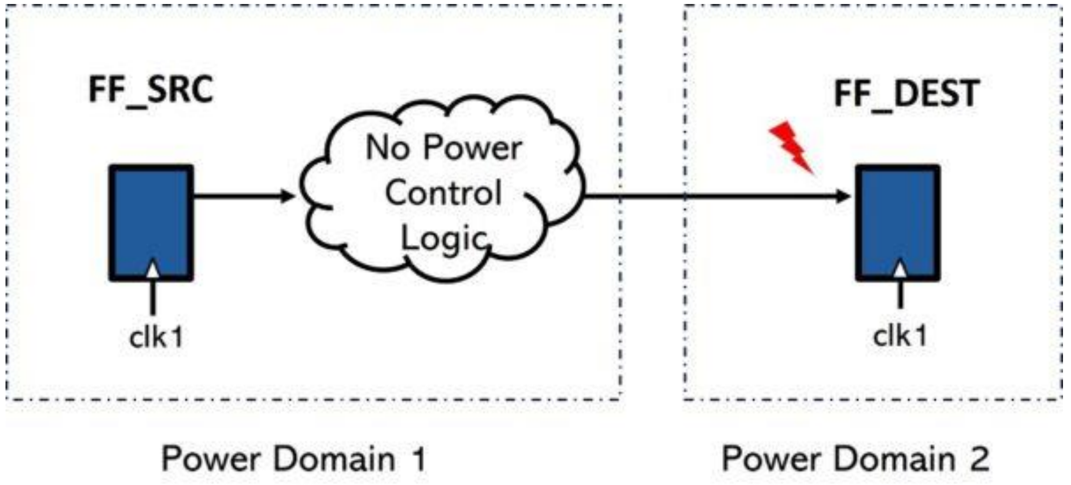
- Interrupts
  - Specific connectivity register based
  - Can propagate through special cells like synchronizers
  - Polarity of connection important
  - No additional drivers or receivers
  - Specific conditions causing 0 value at destination (interrupt override)



Connectivity Challenges

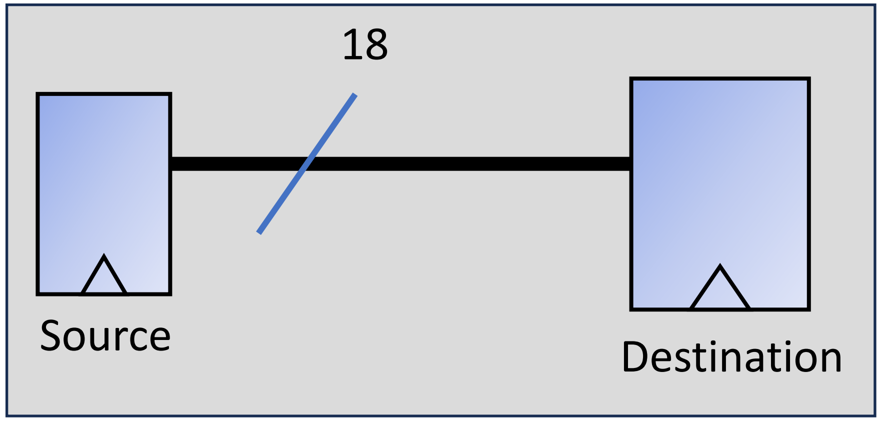
- Power Logic
  - Reset connection from same power domain
  - Isolation cells and signals connectivity
  - Value checks
    - Specific values just before and after isolation

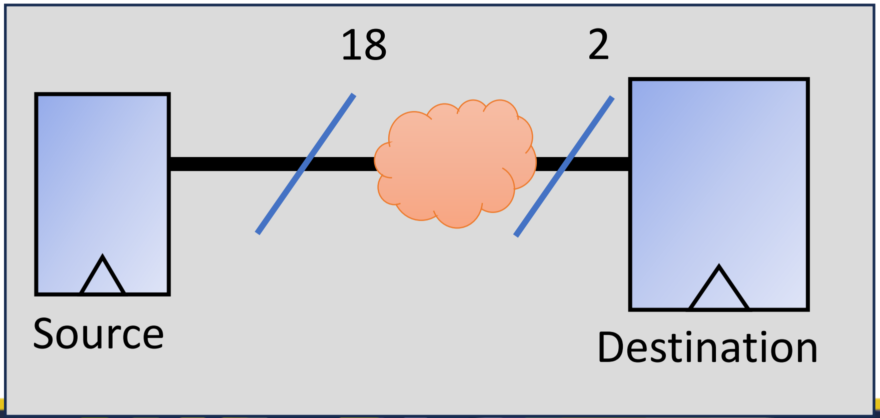




Connectivity Challenges

- Data, Control buses
  - Connectivity under certain conditions
  - Bus swizzle not present (incorrect bits connected)
  - Different sizes, correct hex value transferred



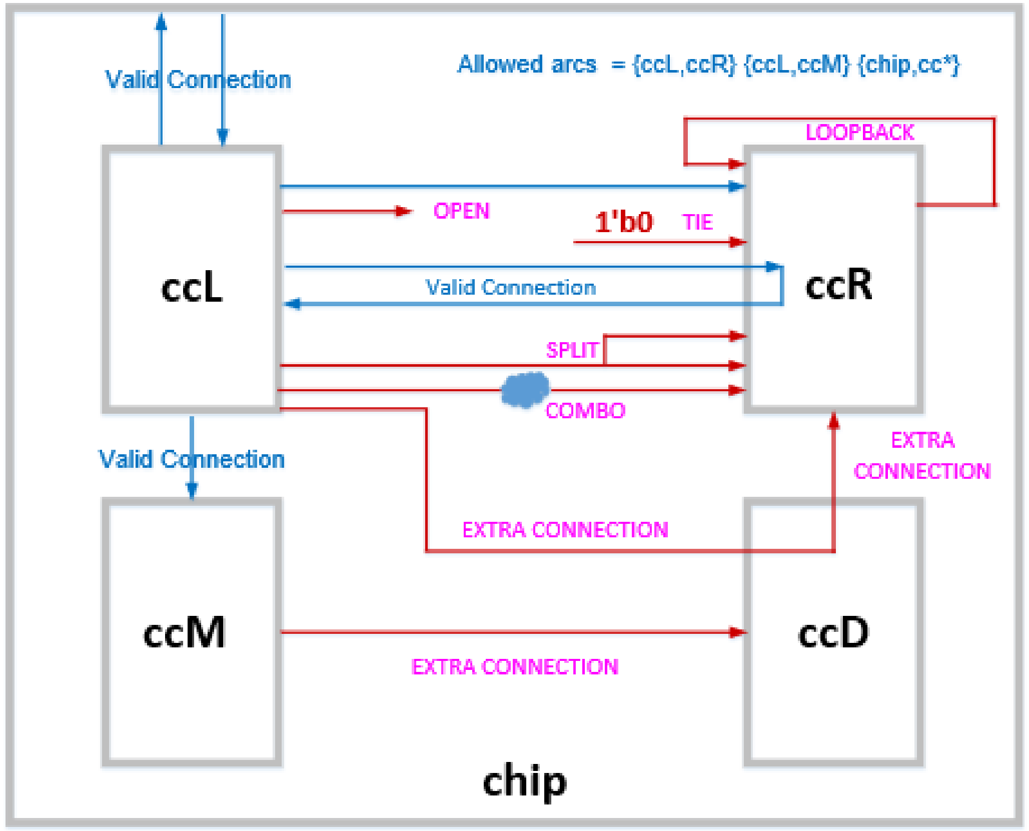


When

- Source - 0~4 : Destination - 00
- Source - 5~8 : Destination - 01
- Source - 9~15: Destination - 10

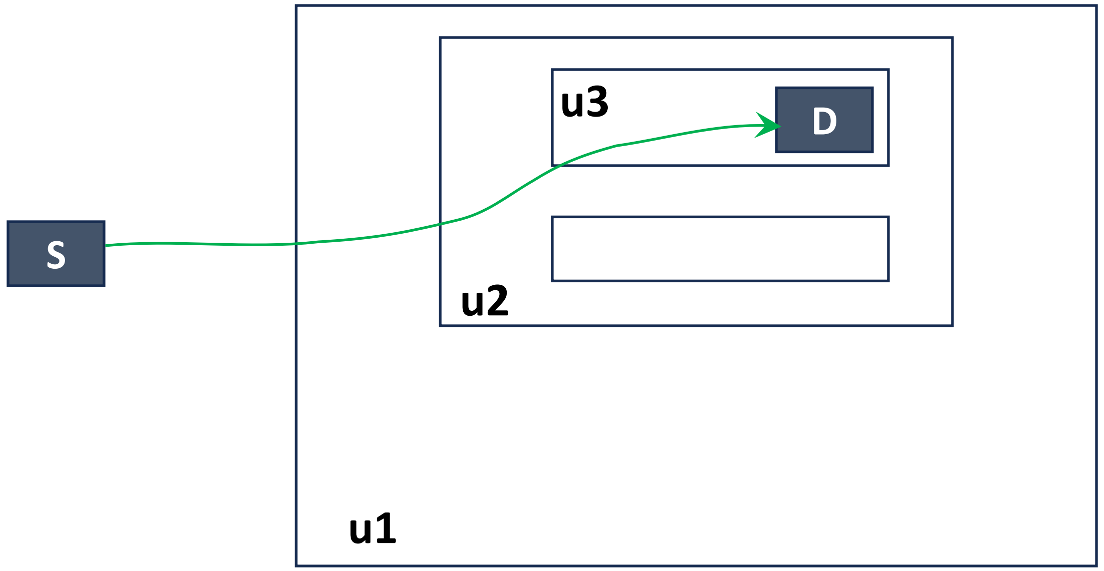
Connectivity Challenges

- Physical Design Requirements - ABUTMENT
  - Connections allowed only between specific instances
  - 1-1 Connections
  - OPEN/TIE/SPLIT is an issue



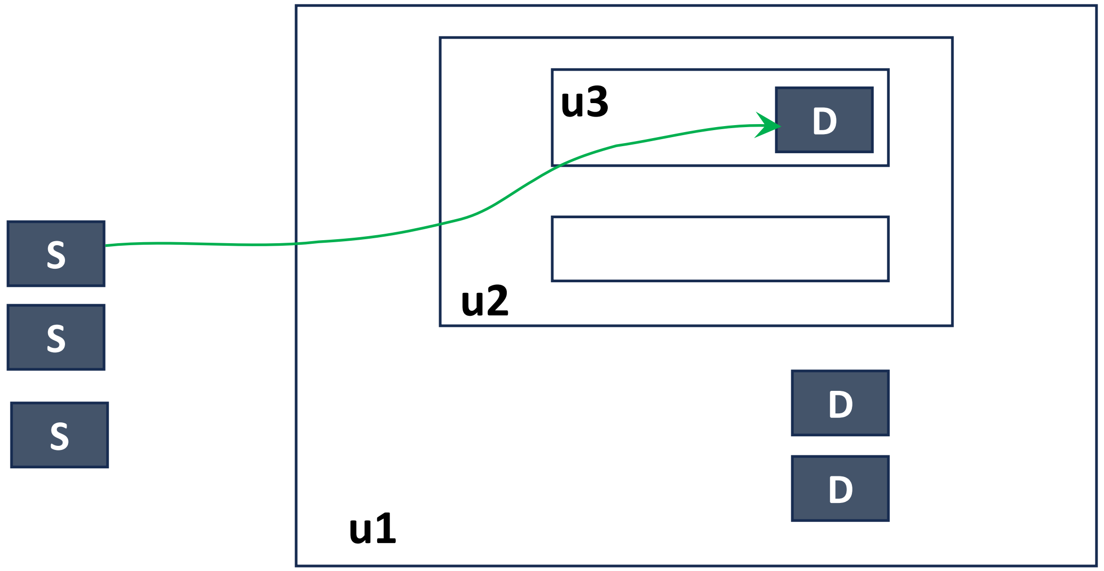
Connectivity Challenges

- Negative testing
  - S does not connect to D
  - S does not connect to D under specific conditions
  - S to D only through sequential
  - S to D only through buffers



Connectivity Challenges

- One to Many, Many to One, One to One
  - S connects to any one of the D
  - S to D only one connection
  - All S connect to the D
  - All S to All D



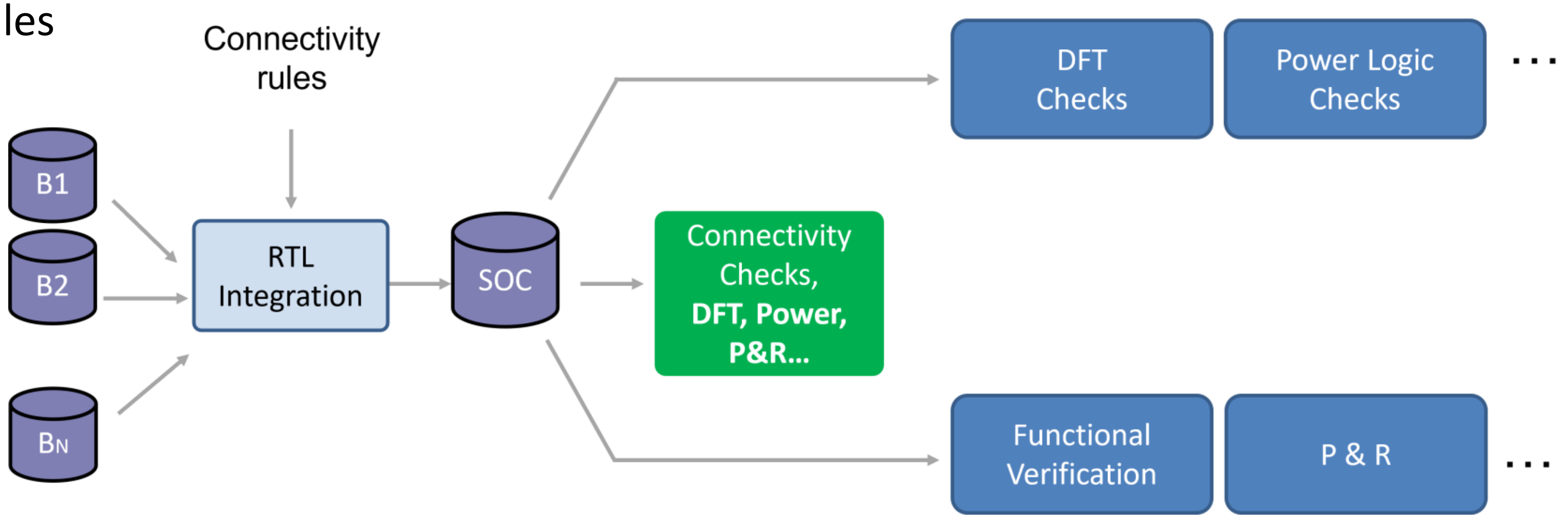
Existing Methodologies Limitations

- Simulation/Emulation
  - 100% coverage not possible
  - Need to write testbenches
  - Runtime - failure may be deep rooted
  - Negative testing not possible directly
- Formal
  - Capacity would be limited
  - May need additional constraints to check properly
  - SoC runs may not be possible
  - Expert users needed
  - Some of the checks may not be formulated
    - Checks not amenable to SVAs
  - Negative testing not possible directly

Connectivity Checking Static Sign-Off

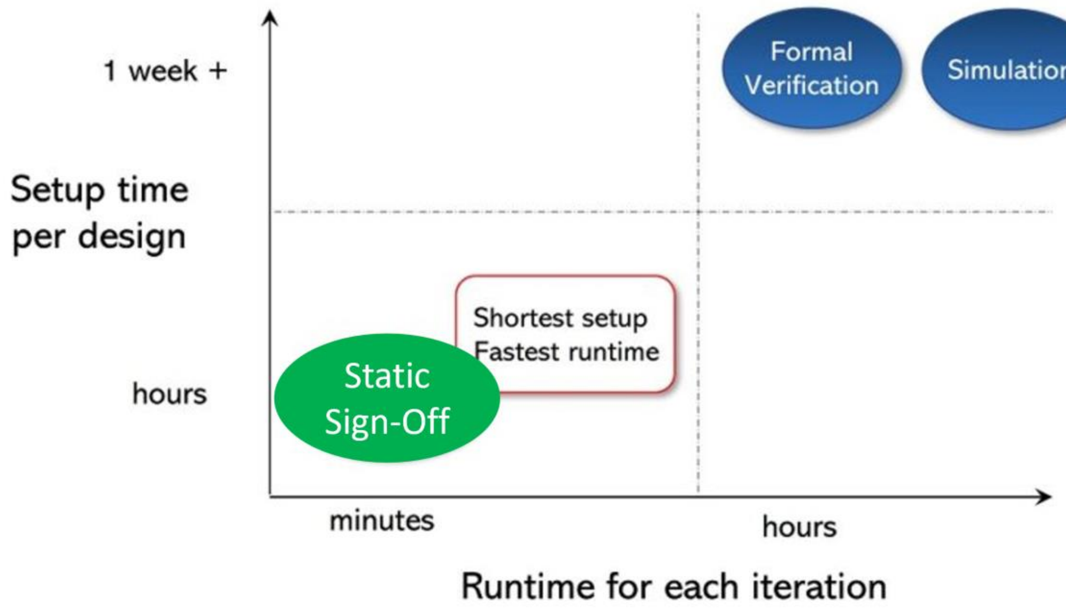
- Safeconnect enables connectivity checking static sign-off
- Solves compatibility and integration challenges engineers face

Connectivity rules



Connectivity Checking Static Sign-Off

- Handles limitations of existing methodologies

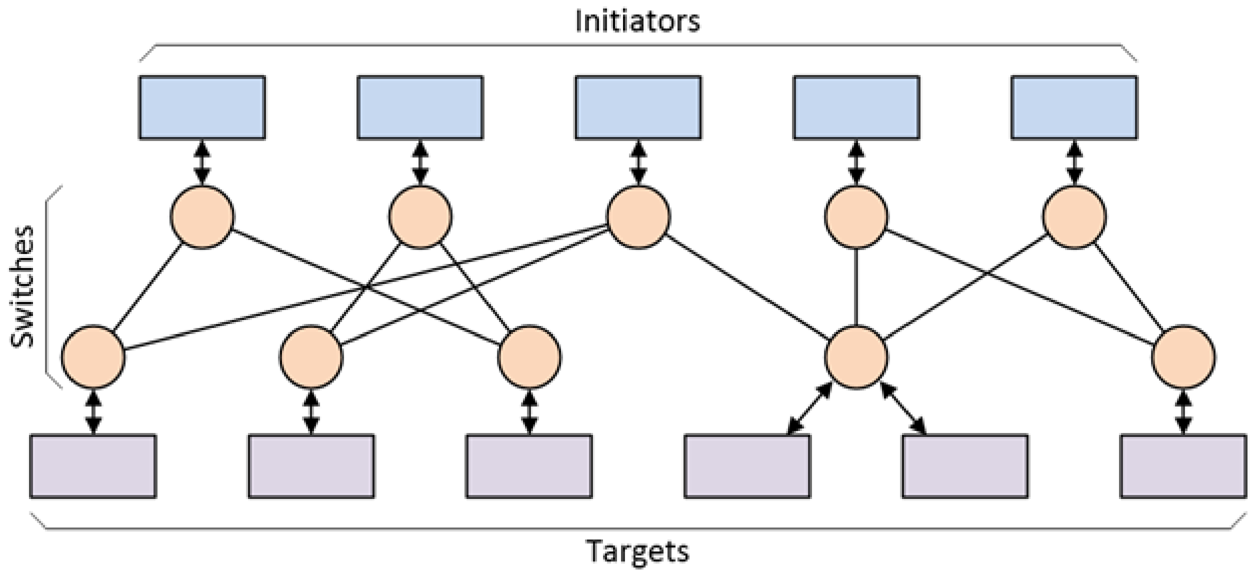


- ✓ Easy to setup
- ✓ 100% Coverage
- ✓ Fast runtimes
- ✓ SoC Capacity
- ✓ Expert user not needed
- ✓ Negative testing can be done

INTERCONNECT NOC

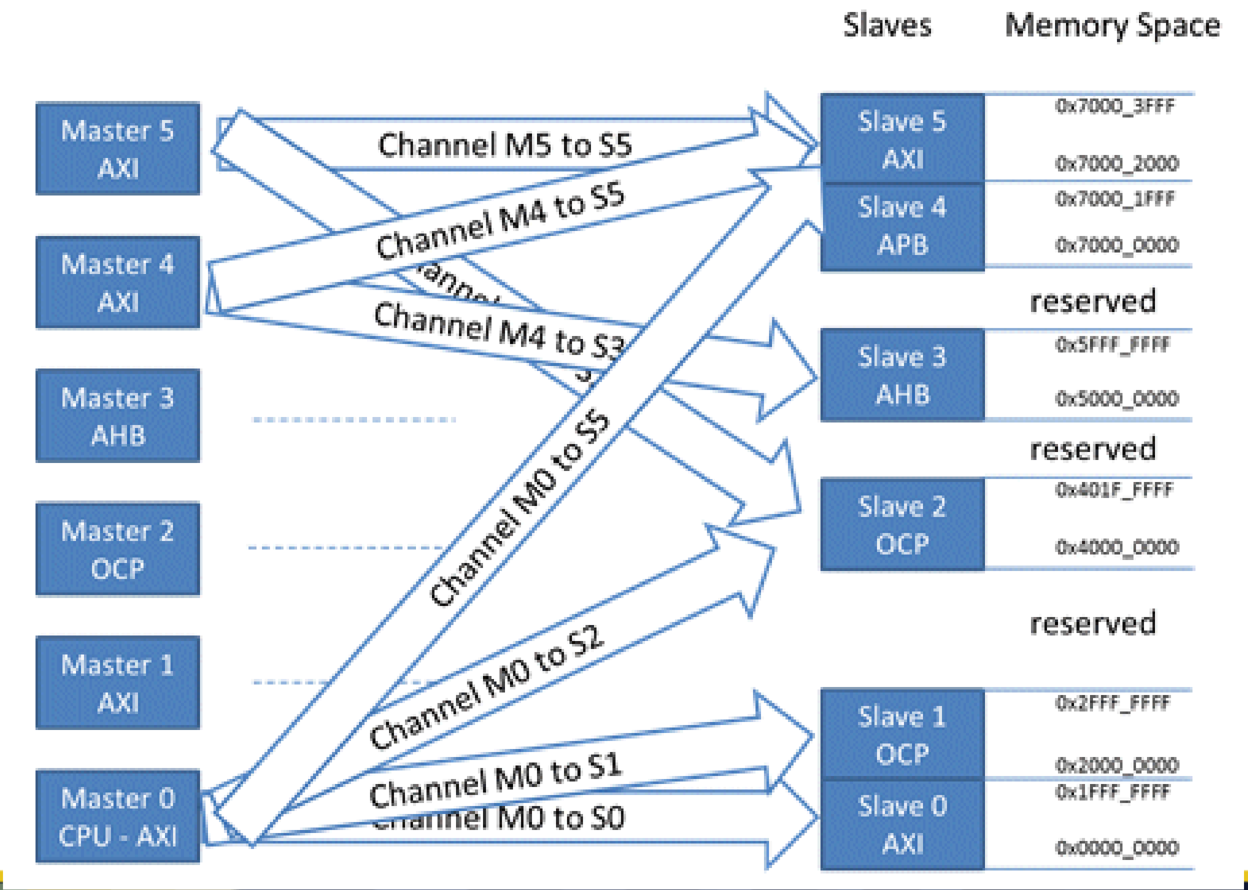
Interconnect Verification Challenges

- Power and performance
- Functional safety
- Security
- Deadlock and livelocks
- Interconnect routes or reachability



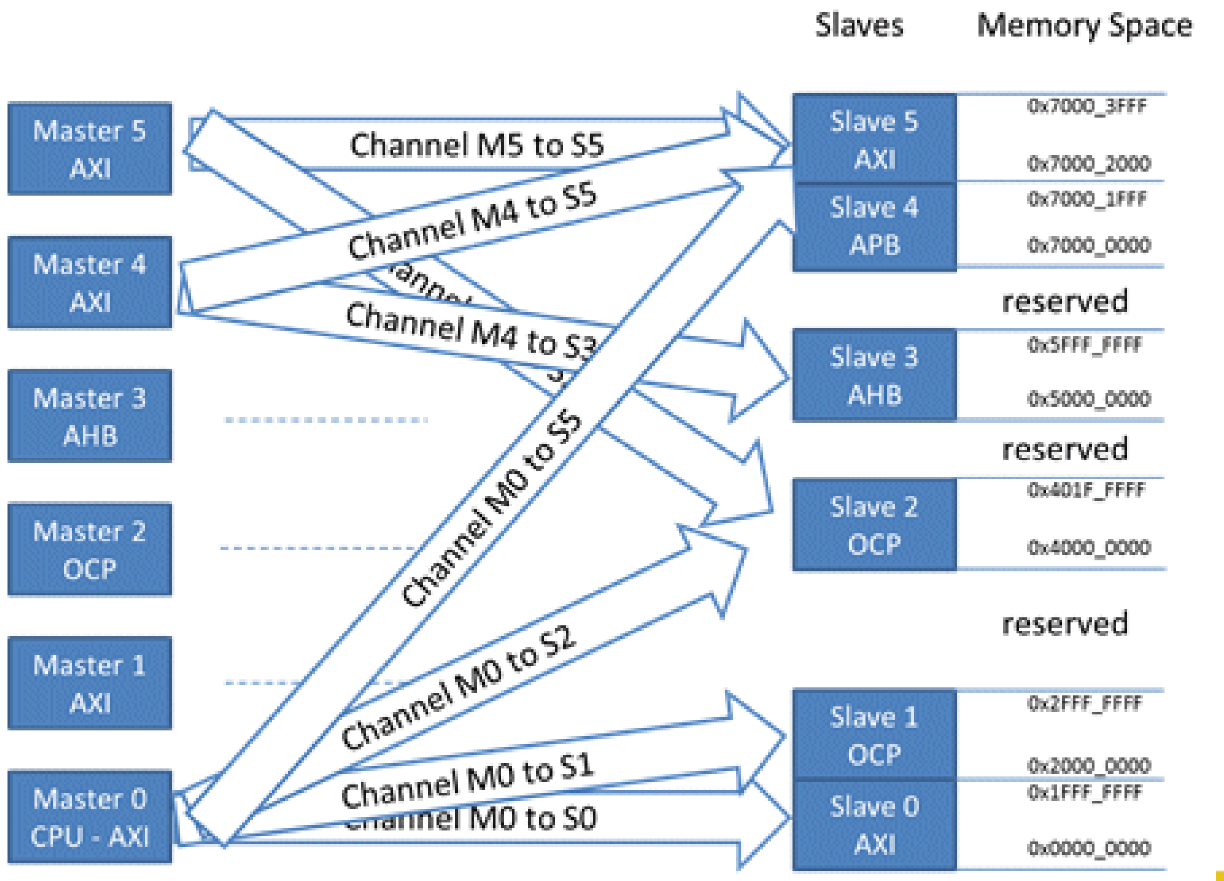
Interconnect Verification

- Reachability Verification
  - A particular master connected to slave
  - Protocol ports connected correctly
  - Connections through complex sequential logic (bridges etc.)
  - Negative verification – masters not connected to slaves as per specification



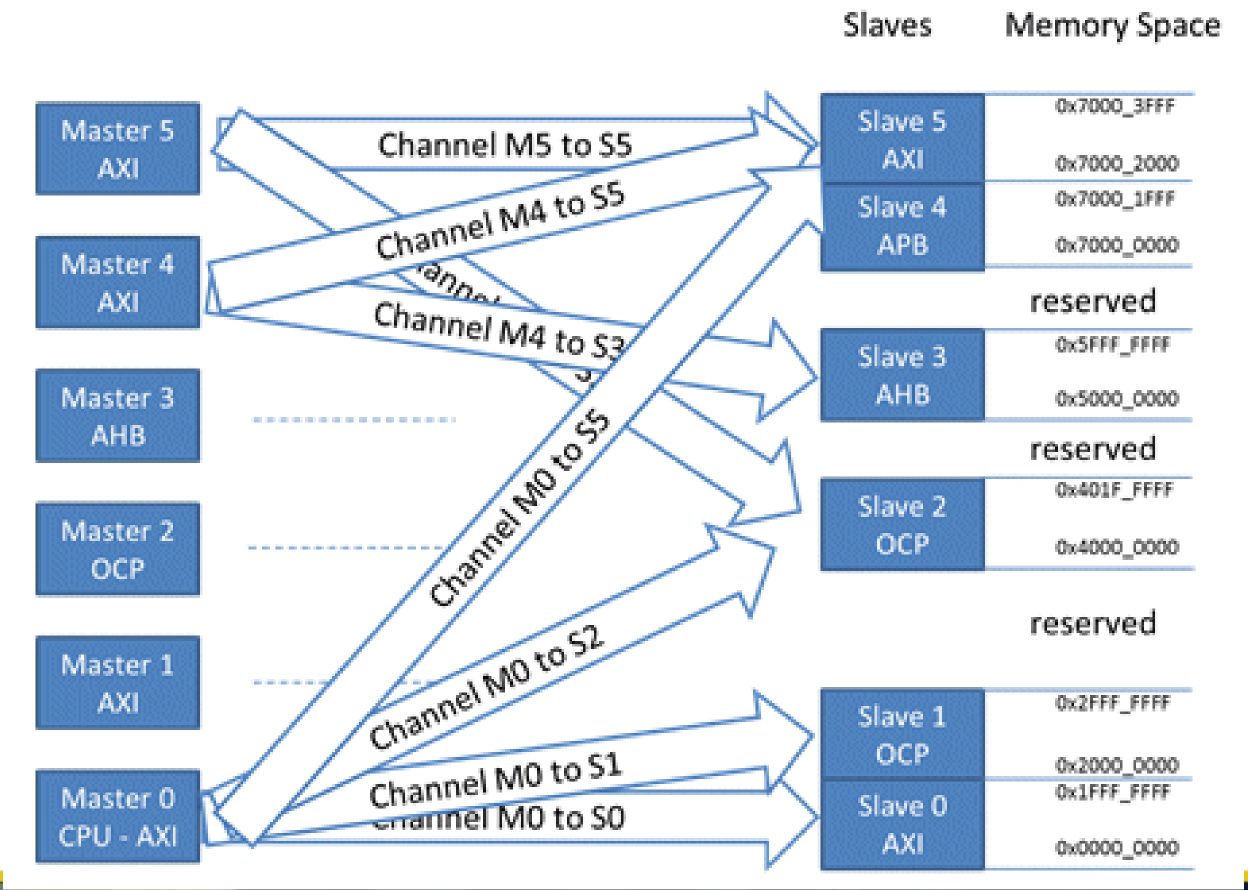
Interconnect Verification

- Configuration from master reaches appropriate slave
- Transaction from master transits into the interconnect and received by corresponding slave
- If error response – error code properly propagated through the interconnect back to master



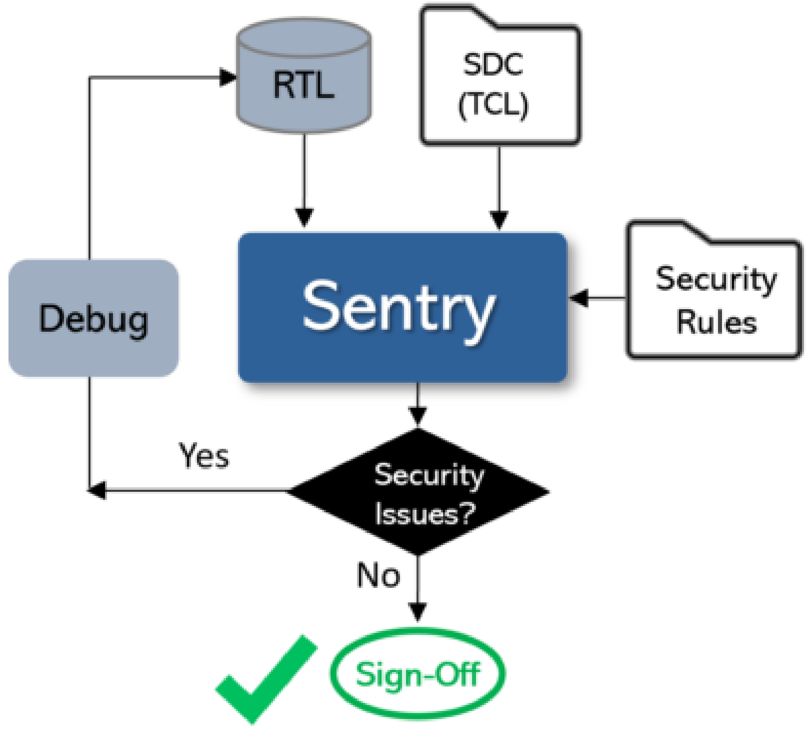
Interconnect Verification

- Security management
  - Forbid transactions targeting secured area
- Power Management
  - When power off forbid direct access to slave



Interconnect Static Sign-Off

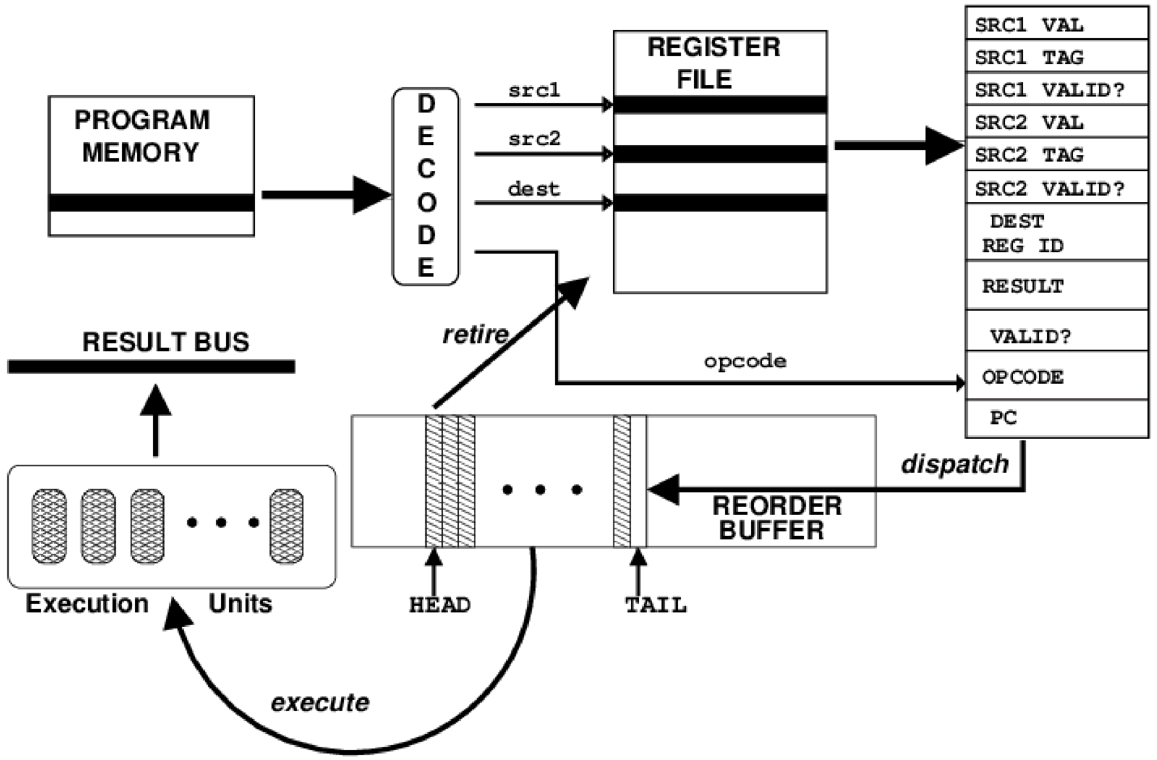
- Sentry enables interconnect transaction and security signoff



Security

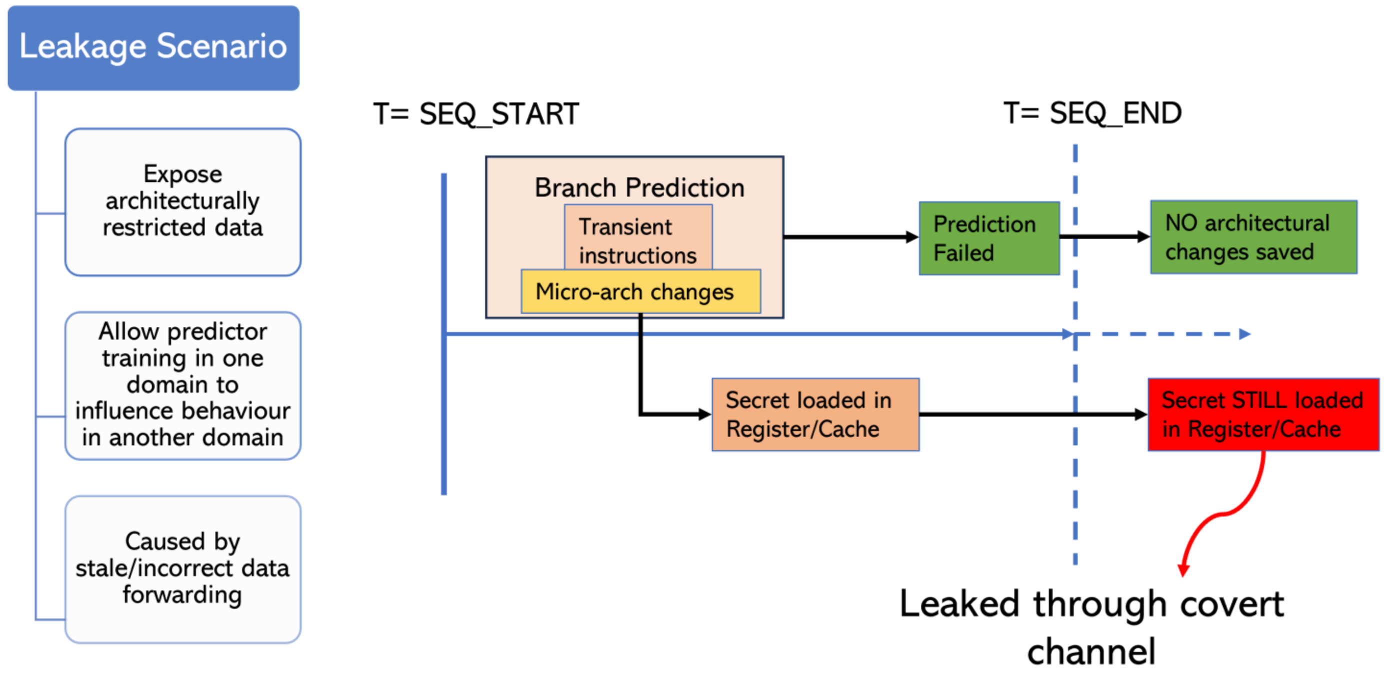
CPU Security – Meltdown & Spectre

- Modern CPUs employ branch prediction and out-of-order execution to improve performance
- CPUs look forward and execute instructions (transient instructions)
- But are squashed before they impact architectural state
- Can be exploited to encode unauthorized data in the microarchitectural state



CPU Security – Meltdown & Spectre

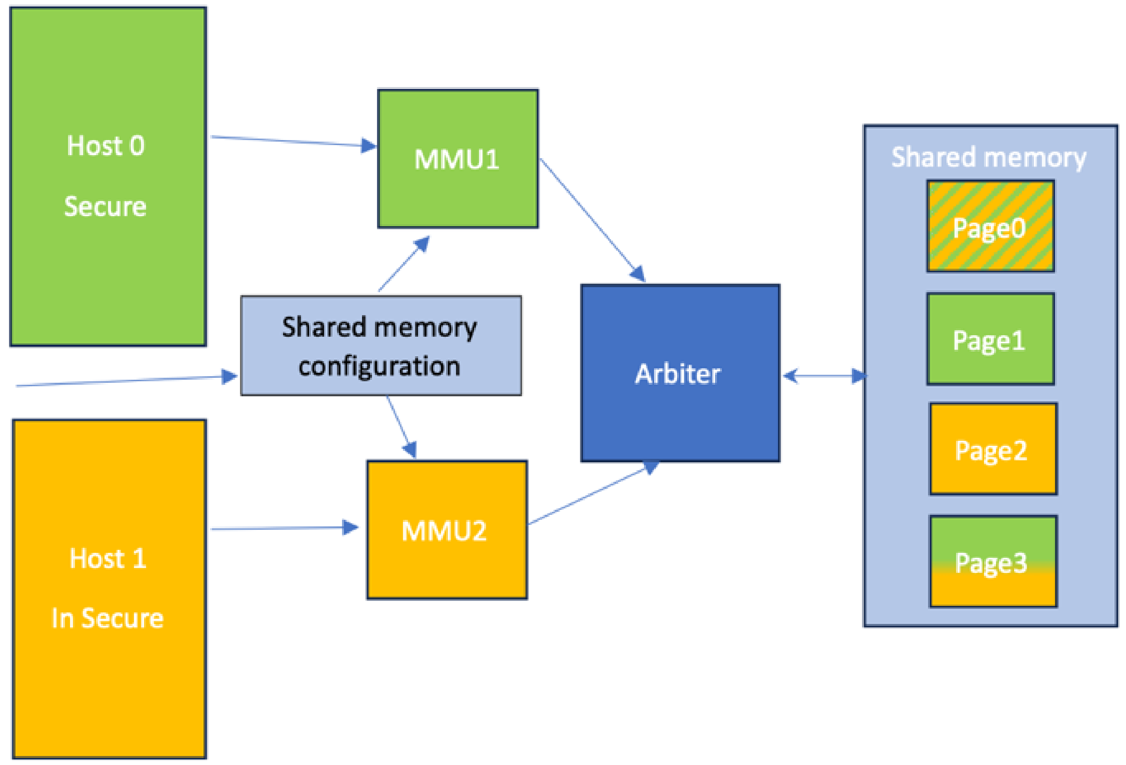
- Exposure of sensitive information during transient execution



Leakage Scenario
Expose architecturally restricted data
Allow predictor training in one domain to influence behaviour in another domain
Caused by stale/incorrect data forwarding
T= SEQ_START
T= SEQ_END
Branch Prediction
Transient instructions
Micro-arch changes
Prediction Failed
NO architectural changes saved
Secret loaded in Register/Cache
Secret STILL loaded in Register/Cache
Leaked through covert channel

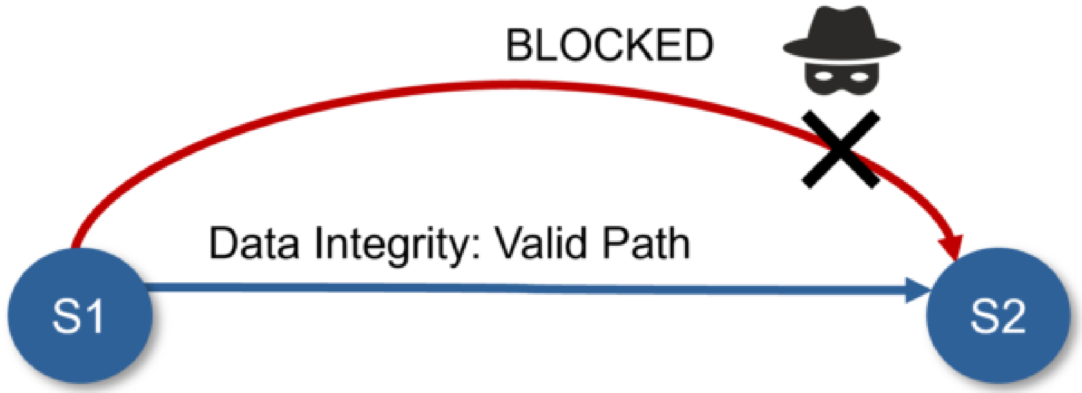
SoC/ Memory Subsystem Security

- Host0 and Host1 share the memory
- 4 Pages in shared memory
- MMU's configure page is read or write to/from a host
  - Page0 is shared RW and accessible by both
  - Page1 is exclusive RW to Host0
  - Page2 is exclusive RW to Host1
  - Page3 is mailbox RW by Host0 and read only by Host1
  - Configuration is written into the shared memory configuration block via a sequence of writes from an external entity



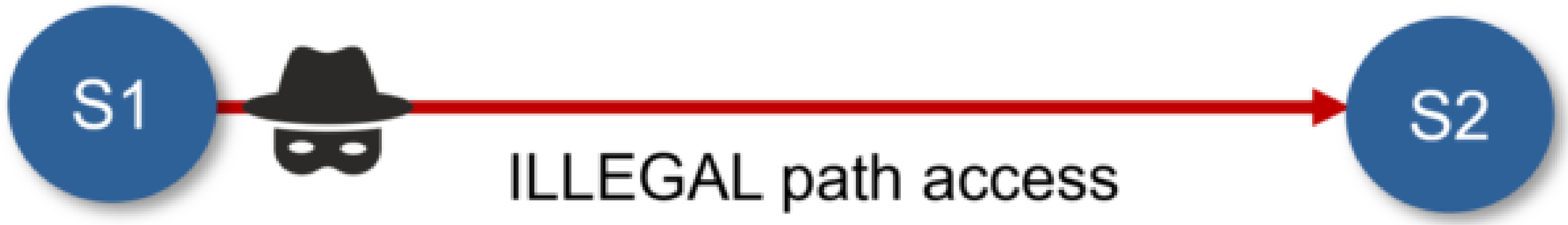
Data Integrity

- Secure data transfer between protected domains
  - No corruption
  - Unauthorized access
- Test for potential blocks to the paths and whether the path is vulnerable to unauthorized data transfers.
- Verify
  - Ensure that the registers' read & write permissions are correctly set by checking that only specific processes can access certain registers, and under defined conditions.
  - Verify that only the CPU can write to configuration registers and that peripheral devices have restricted read-only access
  - Check whether access to any register is blocked



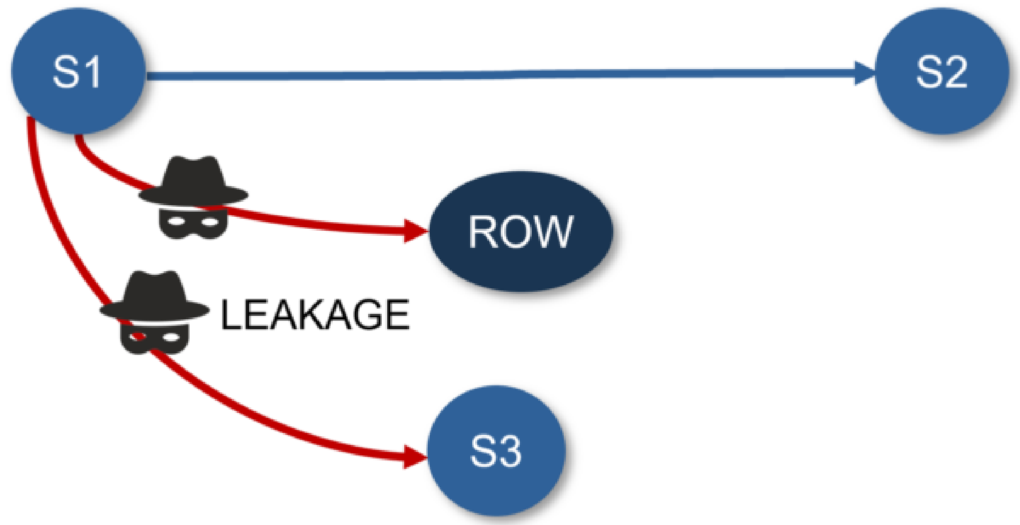
Illegal Path Access

- No illegal or unauthorized access
- Verify the integrity of bus separation or firewall mechanisms used to prevent third party IPs from accessing secured registers



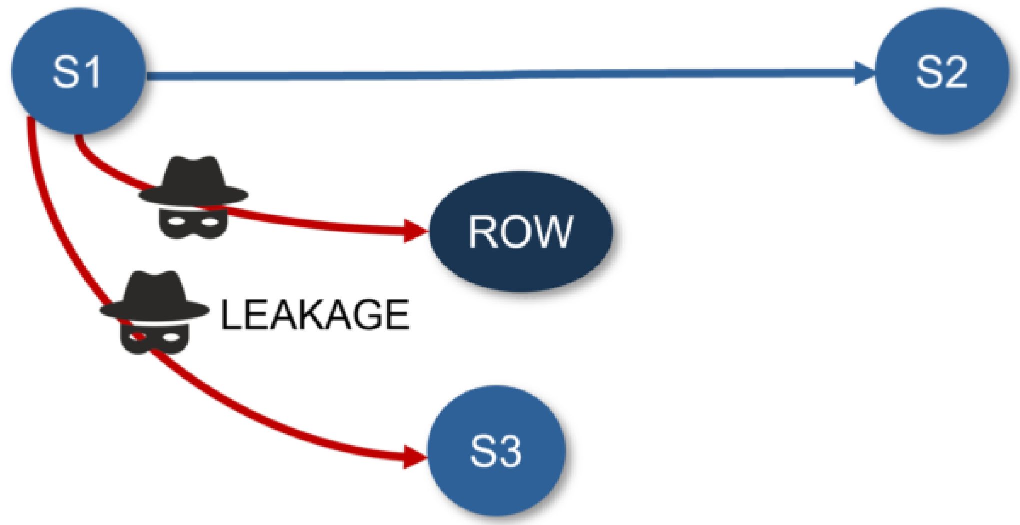
Leakage Prevention

- Sensitive data cannot reach unauthorized domains where it could be compromised.
- All data paths handling sensitive information are secure and isolated from non-secure data paths.



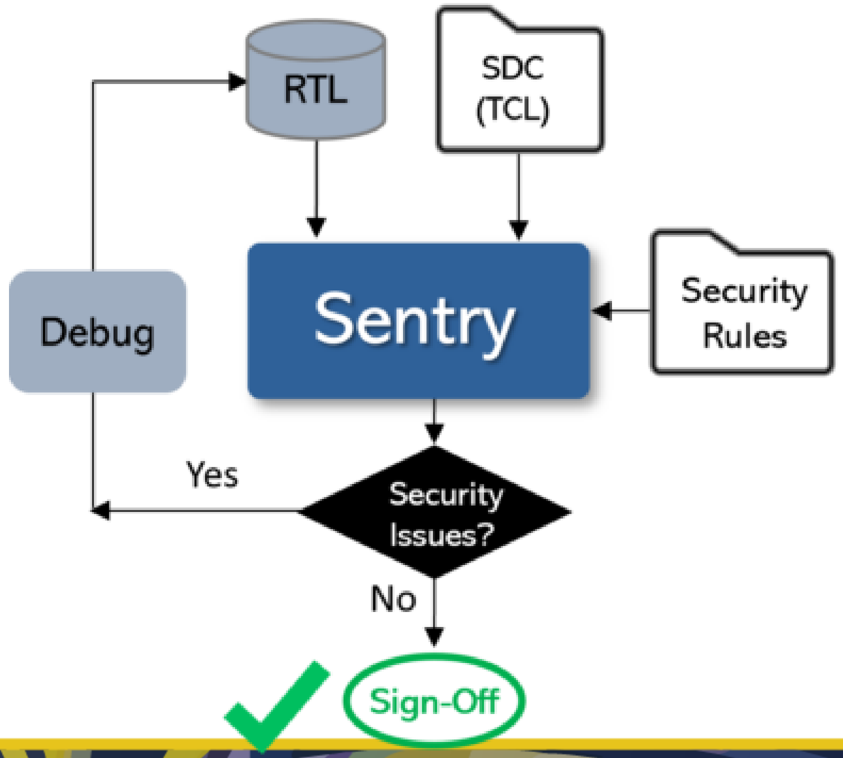
Interference Safeguarding

- Verify
  - Access control logic correctly grants access to authorized assets and blocks unauthorized ones.
  - Suspicious or unauthorized data transfer activities that cross domain boundaries are detected
  - Suspicious or unauthorized data transfer activities that cross domain boundaries are detected



Security Static Sign-Off

- Sentry enables security signoff



Questions?