2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Expanding role of Static Signoff in Verification Coverage

Vikas Sachdeva

Senior Director, Product Strategy

REAL INTENT

accellera
SYSTEMS INITIATIVE

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Introduction

accellera
SYSTEMS INITIATIVE

is earliest possible efficient verification of each design step

is earliest possible efficient verification
of each design step

efficiently enables shift left

Static Verification vs Dynamic Verification

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

Static Verification vs Dynamic Verification

Dynamic verification

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

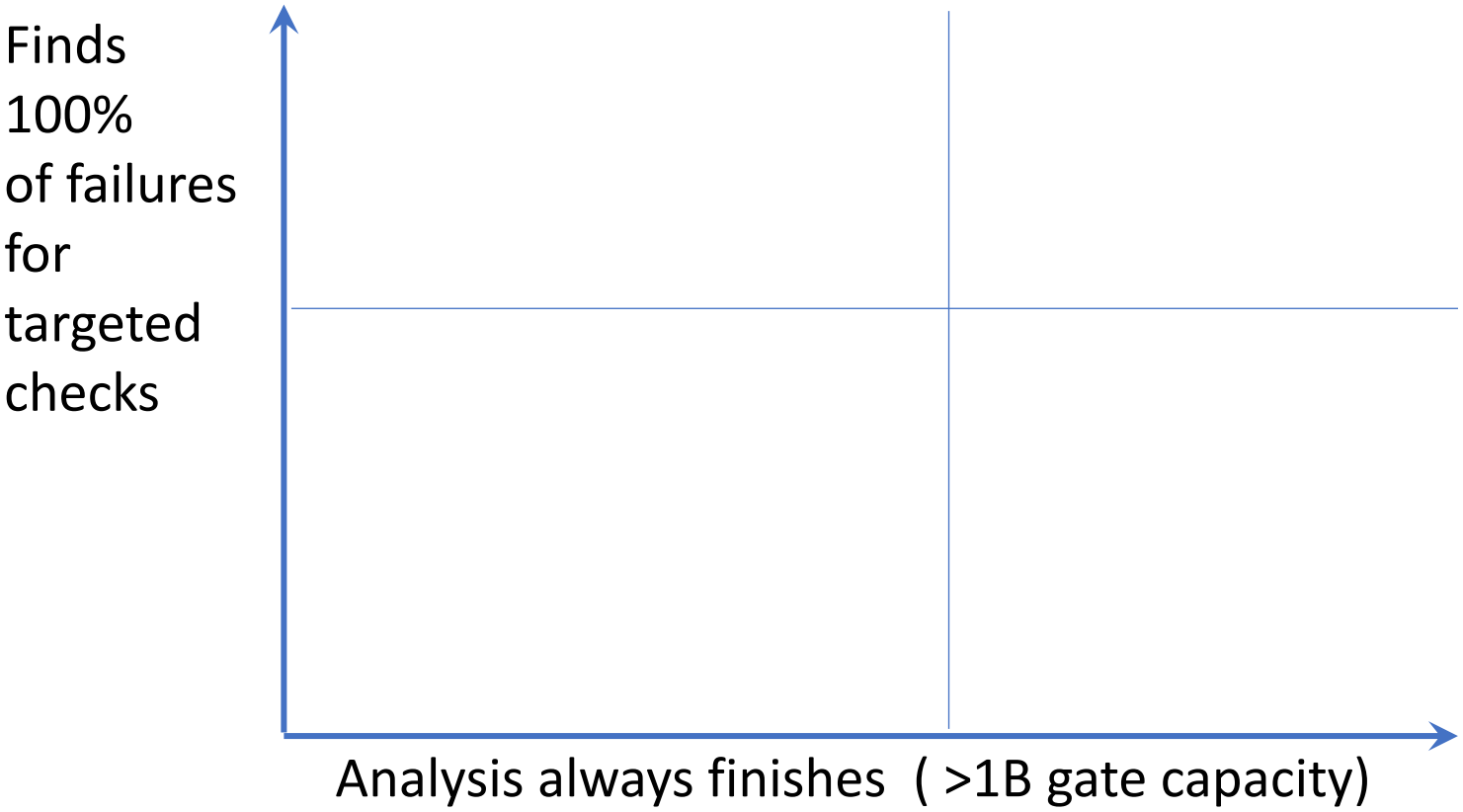
Simulation
Emulation

Static verification

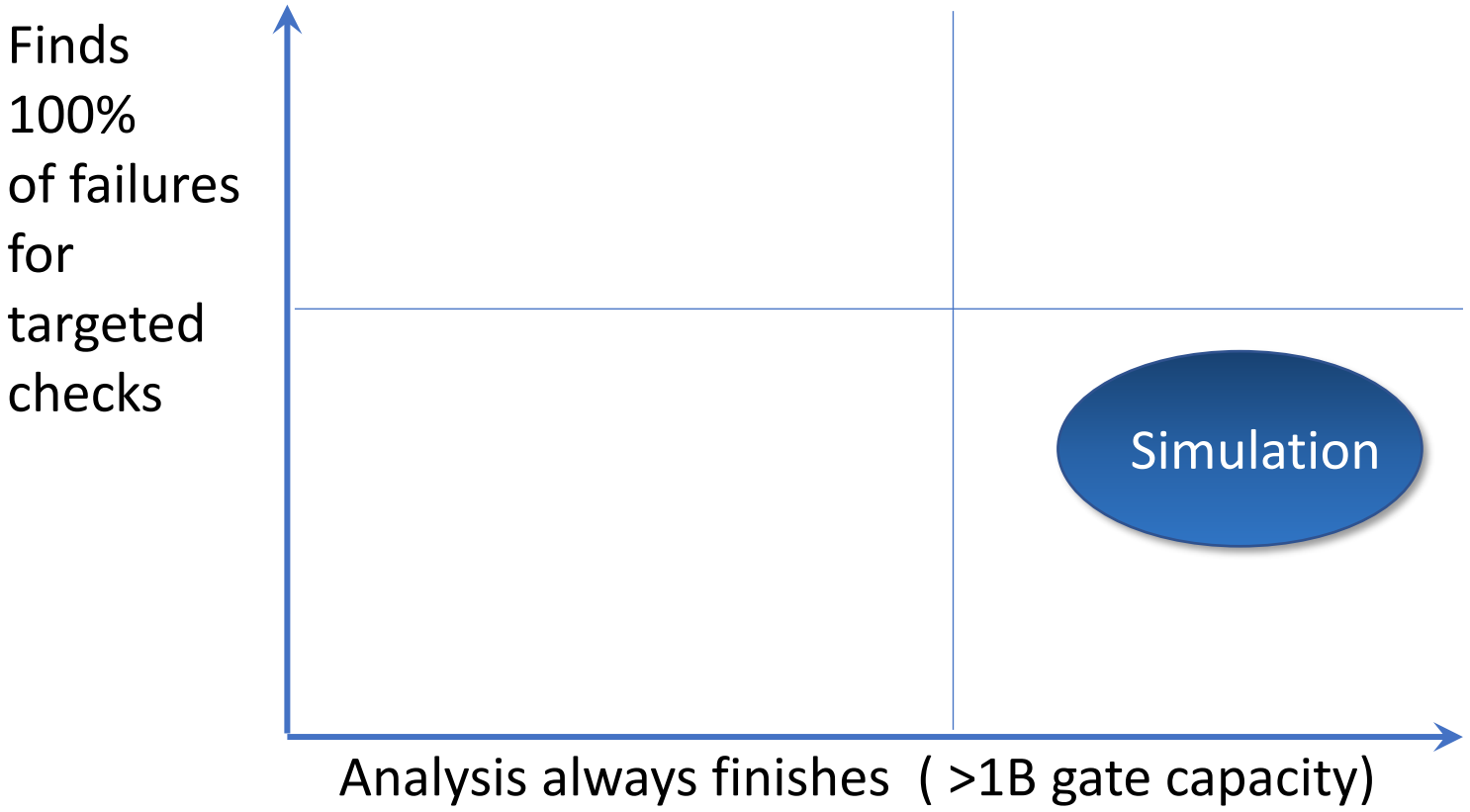
- Utilizes search & analysis to find ALL targeted failures
- Test cases not required

CDC
Lint…
Formal
STA
DRC

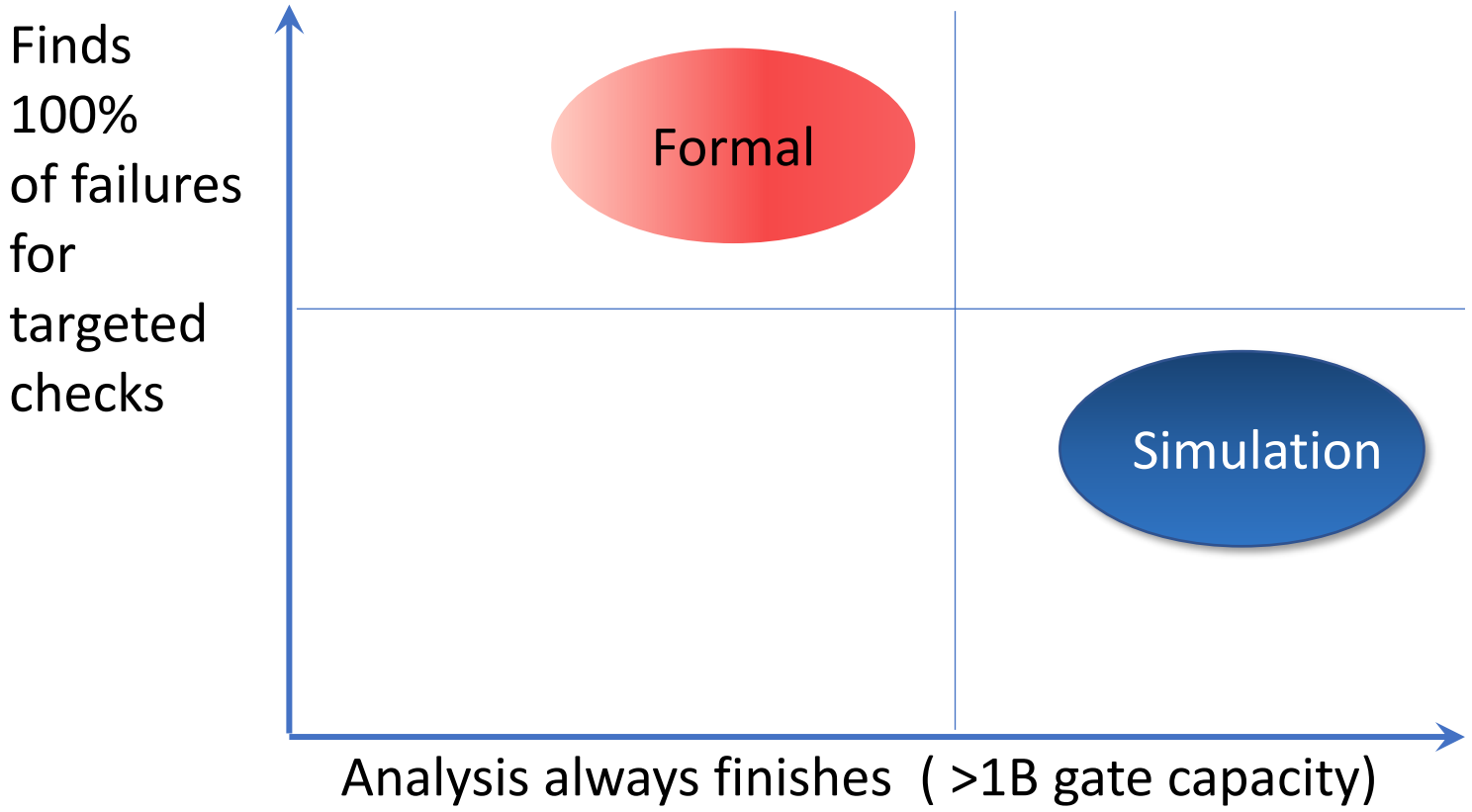
Static Sign-off vs Formal & Simulation



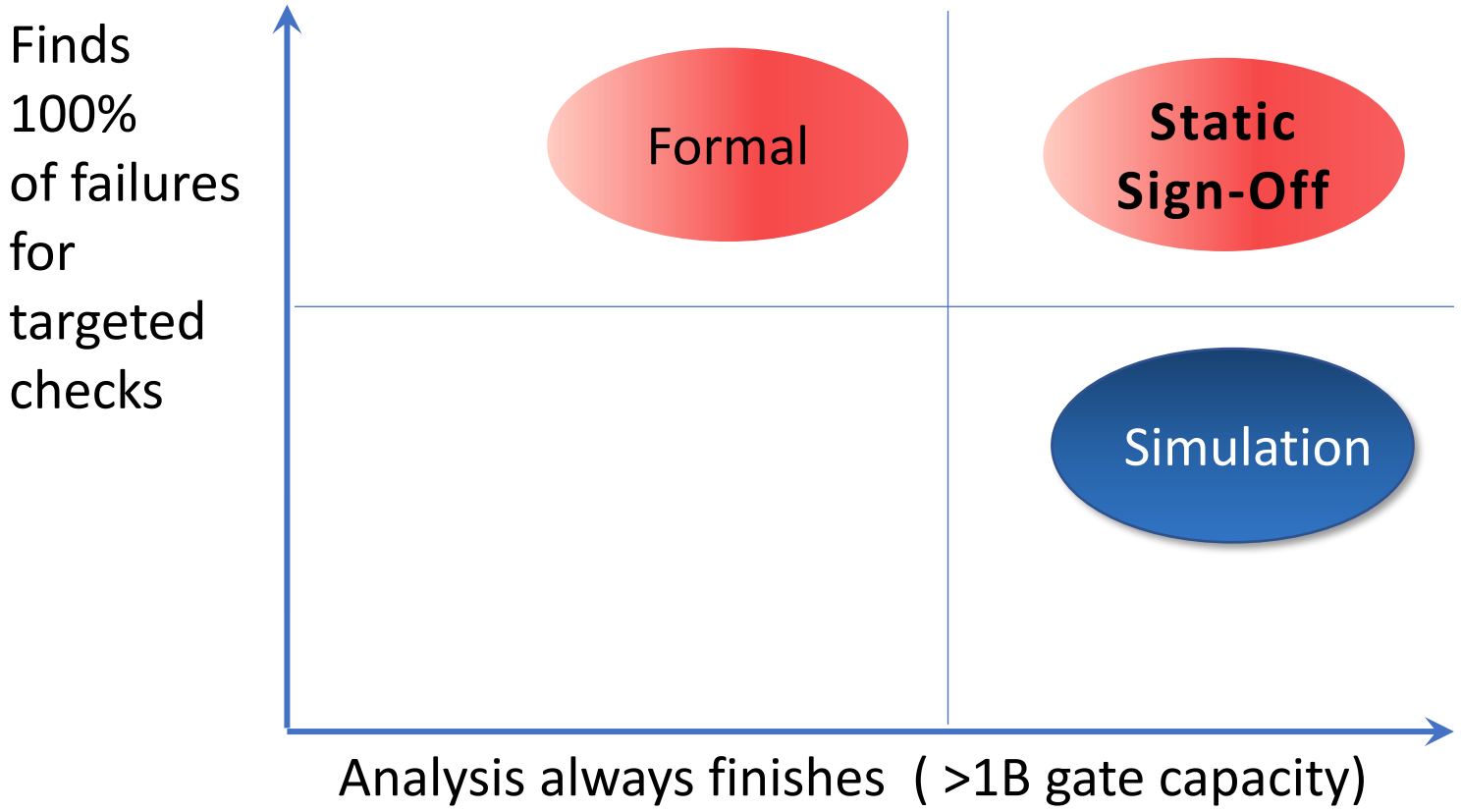
Static Sign-off vs Formal & Simulation



Static Sign-off vs Formal & Simulation

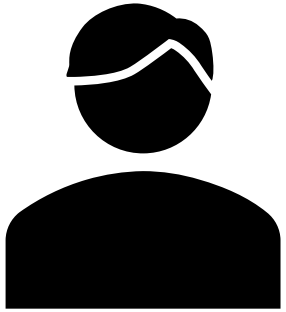


Static Sign-off vs Formal & Simulation



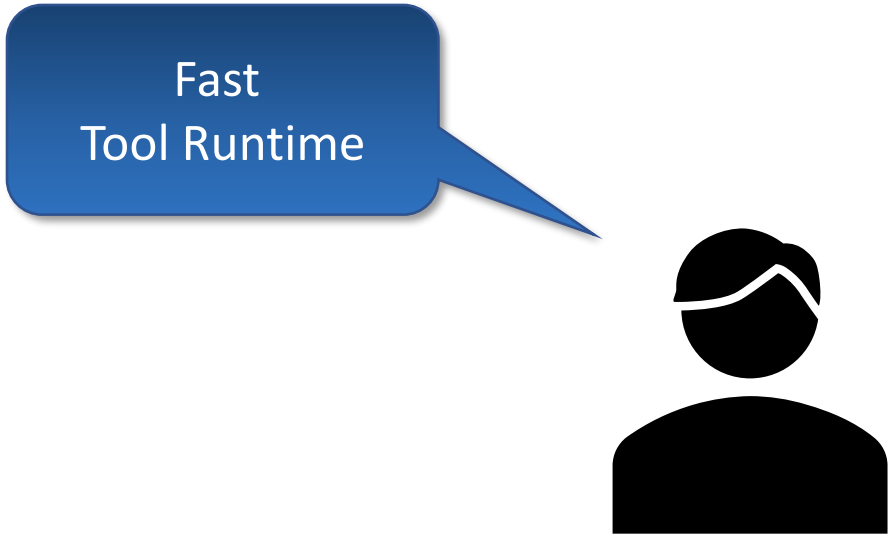
Superior User Experience Drives Shift Left

Four enabling elements of user experience



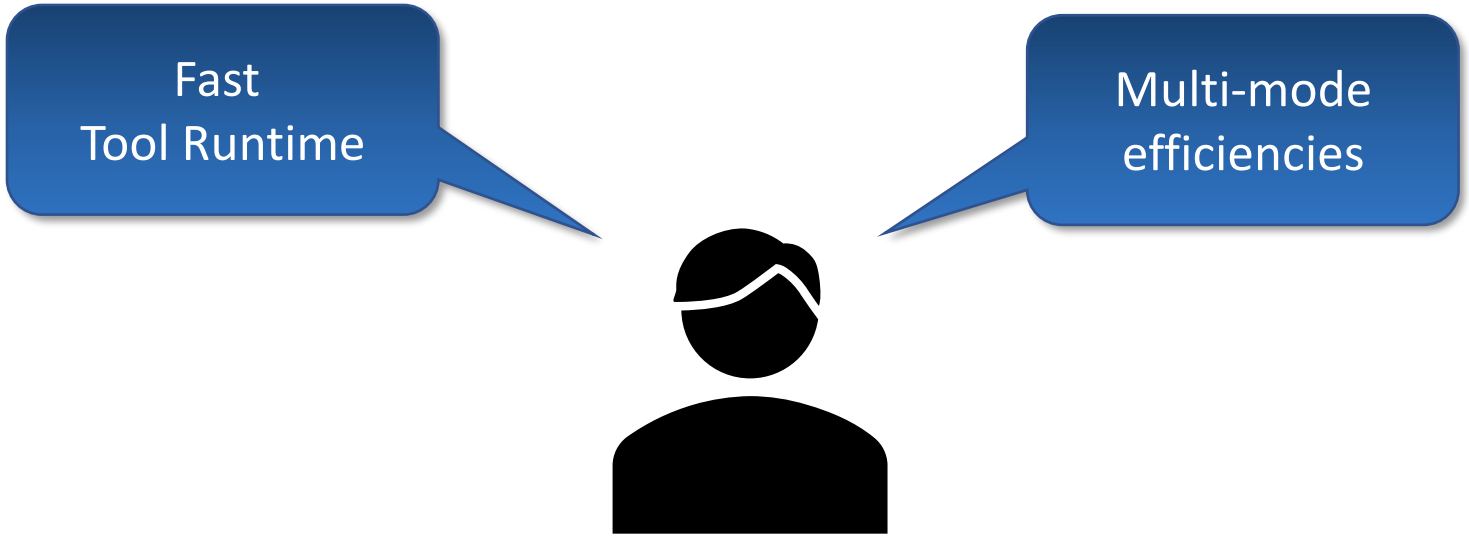
Superior User Experience Drives Shift Left

Four enabling elements of user experience



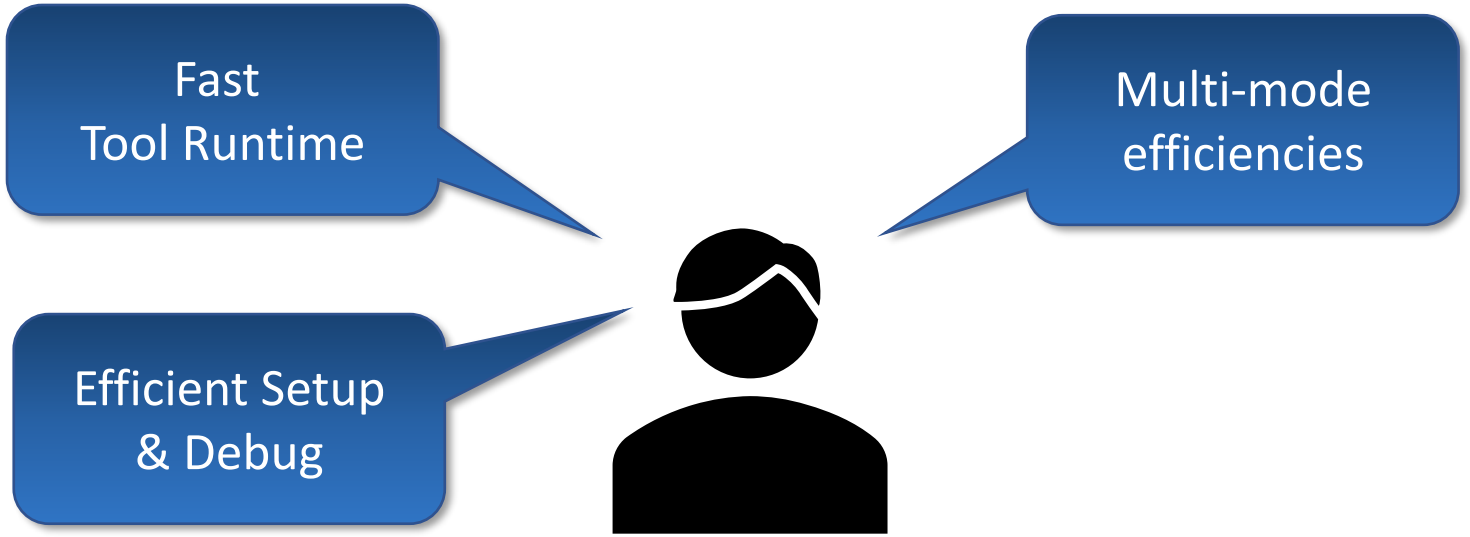
Superior User Experience Drives Shift Left

Four enabling elements of user experience



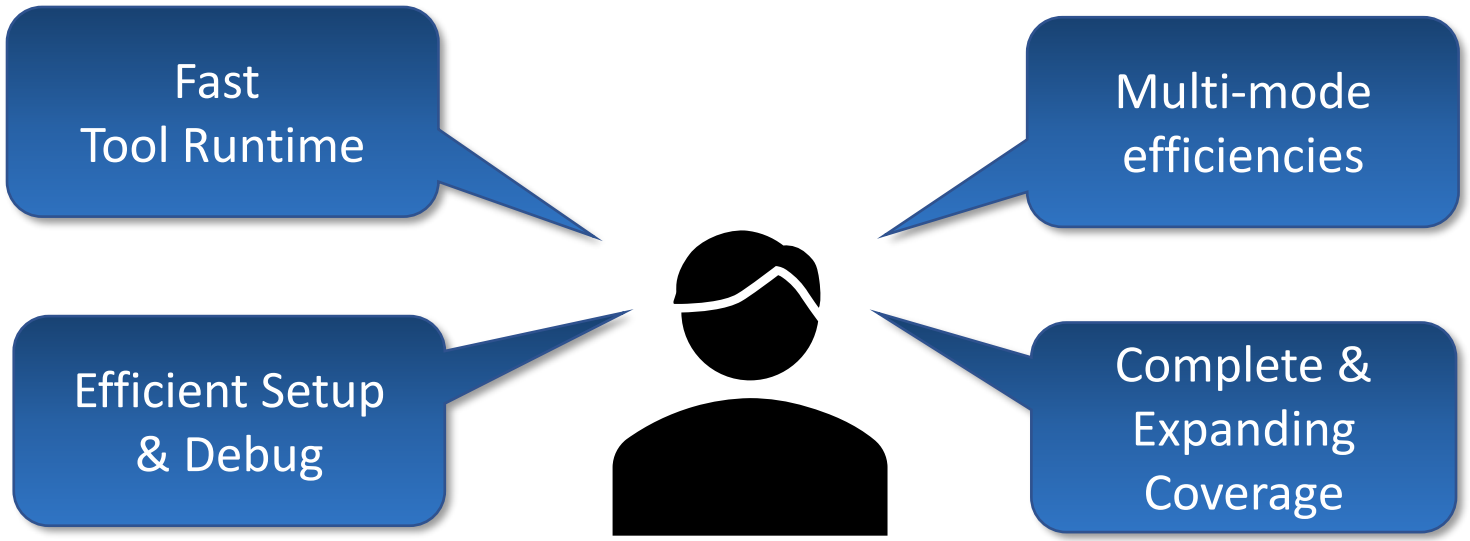
Superior User Experience Drives Shift Left

Four enabling elements of user experience



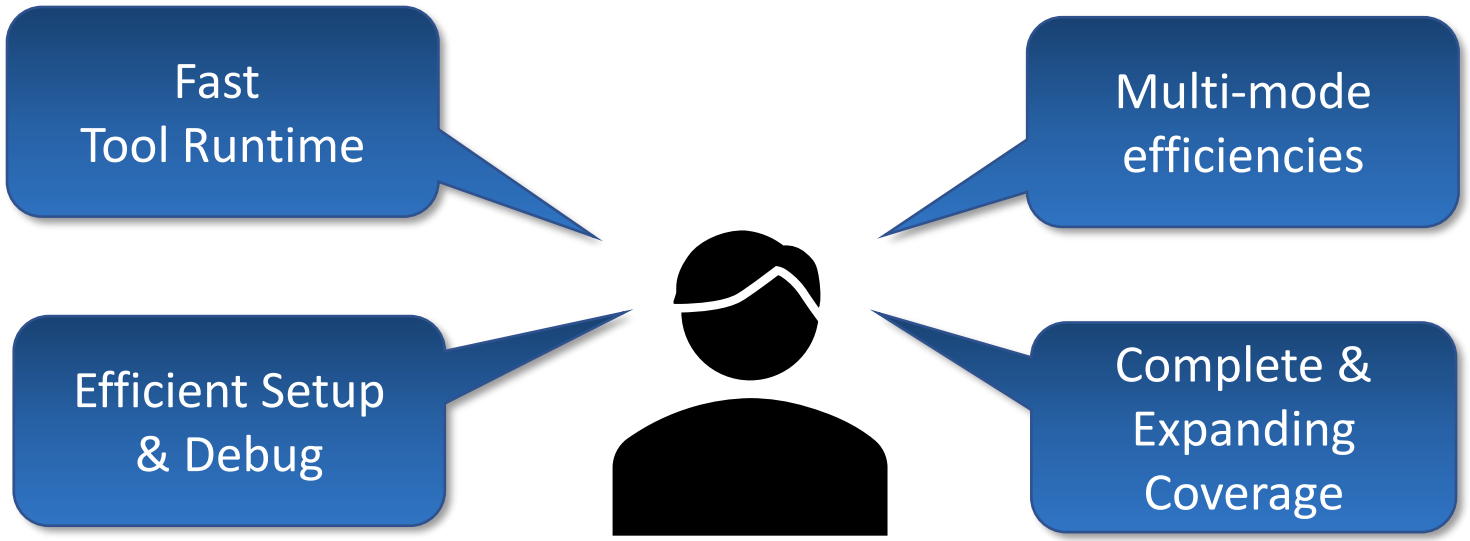
Superior User Experience Drives Shift Left

Four enabling elements of user experience



Superior User Experience Drives Shift Left

Four enabling elements of user experience

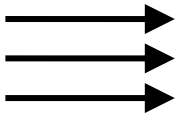


Advanced tools & methodologies
maximize engineering efficiency during design

Enabling Faster Runtime

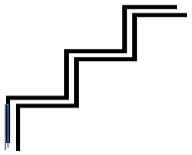
Customized static sign-off engines

Parallel Processing



Hierarchy & abstraction

Incremental analysis

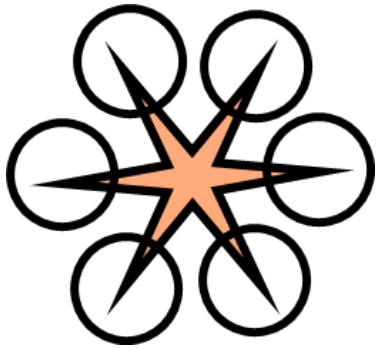


Multi-Mode Efficiencies

Multimode tools = highest engineering efficiency

1 set up. Complete coverage in 1 run. 1 consolidated report.

- Multi-Mode CDC
- Multi-Scenario RDC
- Multi-Test Mode DFT
- Multi-Policy Lint



Enabling Efficient Setup & Debug

Unified, automated setup across applications → Faster ramp up

Enabling Efficient Setup & Debug

Unified, automated setup across applications → Faster ramp up

Configurations & high granularity rules → Low noise reporting

Enabling Efficient Setup & Debug

Unified, automated setup across applications → Faster ramp up

Configurations & high granularity rules → Low noise reporting

Application-customize Debug: CDC, Lint, RDC…

- Customizable reporting -- hierarchy/organization
- Root cause grouping -- faster refinement
- Targeted attributes & CLIs -- searching & filtering
- Pattern matching-based error targeting

Enabling Efficient Setup & Debug

Unified, automated setup across applications → Faster ramp up

Configurations & high granularity rules → Low noise reporting

Application-customize Debug: CDC, Lint, RDC…

- Customizable reporting -- hierarchy/organization
- Root cause grouping -- faster refinement
- Targeted attributes & CLIs -- searching & filtering
- Pattern matching-based error targeting

Complete, Expanding Coverage of Failure Modes

Clock domain crossing

- Multi-Mode – for all clocking configurations
- Hierarchical analysis with flat accuracy
- Glitch checking to prevent netlist failure
- Design-aware Dynamic CDC verification

Complete, Expanding Coverage of Failure Modes

Clock domain crossing

- Multi-Mode – for all clocking configurations
- Hierarchical analysis with flat accuracy
- Glitch checking to prevent netlist failure
- Design-aware Dynamic CDC verification

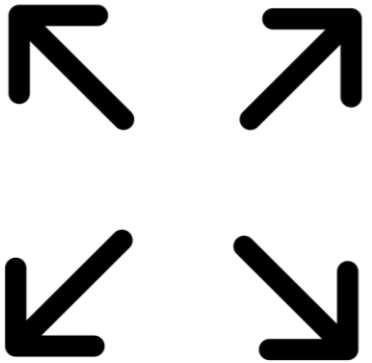
Higher coverage with fine-grained, non-overlapping rules

- Lint - 600 checks
- DFT - 100 checks

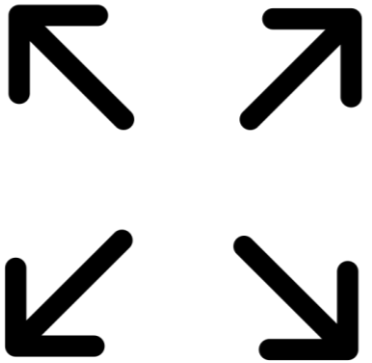
Expanding Coverage -- New Applications

New failure modes

- RDC
- Glitch



Expanding Coverage -- New Applications



New failure modes

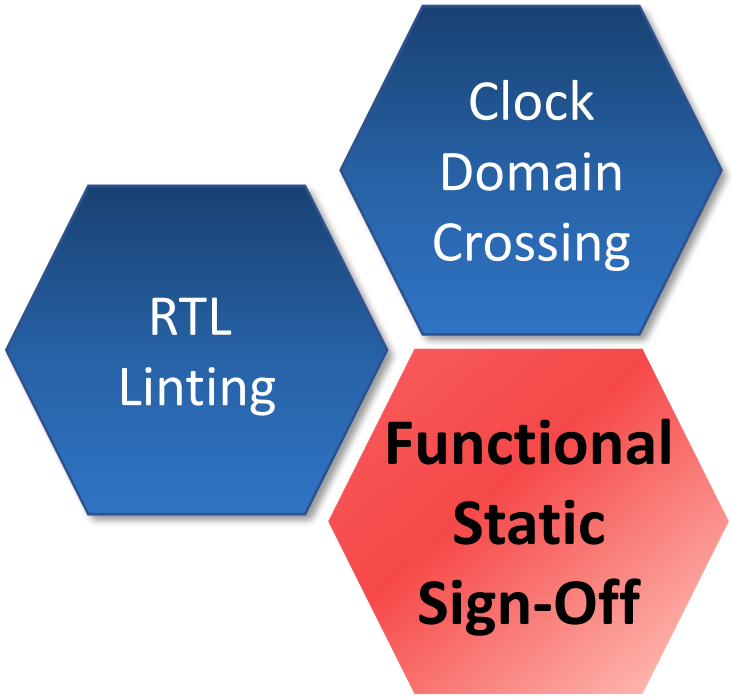
- RDC
- Glitch

New “Shift Left” applications -- streamline RTL & netlist checking

- Connectivity
- DFT
- Design Initialization

Functional Static Sign-Off Expanding Applications

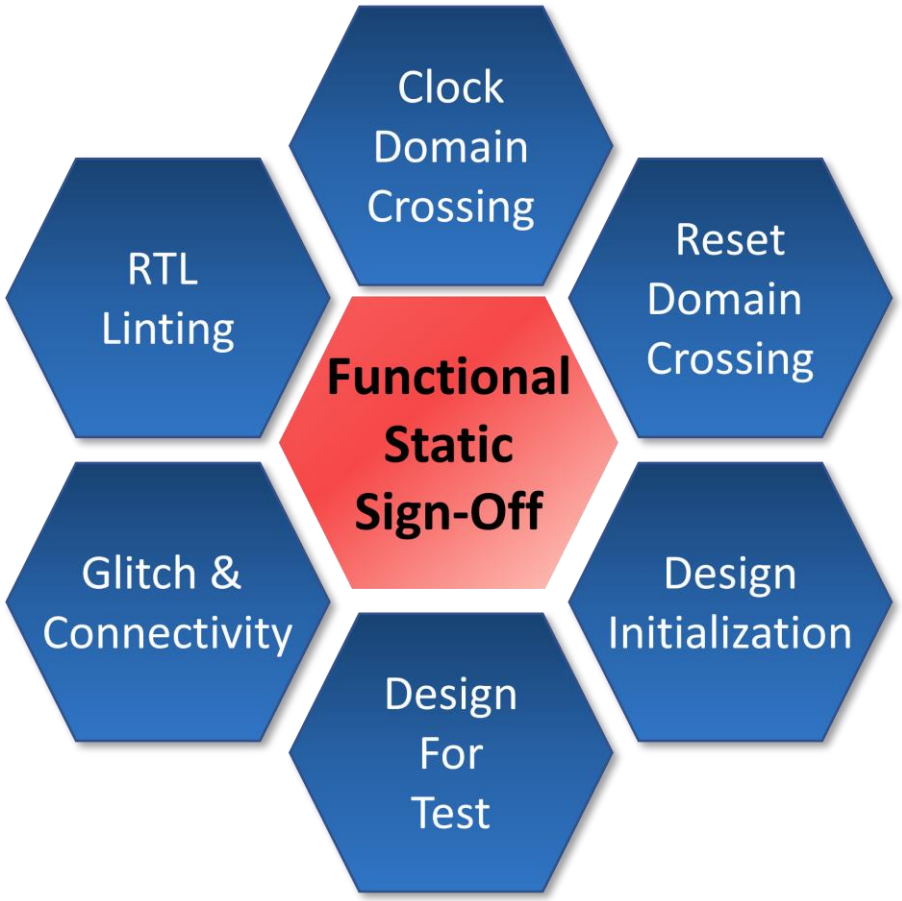
Functional static sign-off began with
RTL Linting
& CDC



Functional Static Sign-Off Expanding Applications

Functional static sign-off began with RTL Linting & CDC

The target applications continuously expand



Superior User Experience Driving Shift Left

Four enabling elements of user experience

- Fast Tool Runtimes
- Multimode
- Efficient set up & debug
- Expanding coverage of failure modes

Engineering ROI expanding
functional static sign-off domains & usage

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

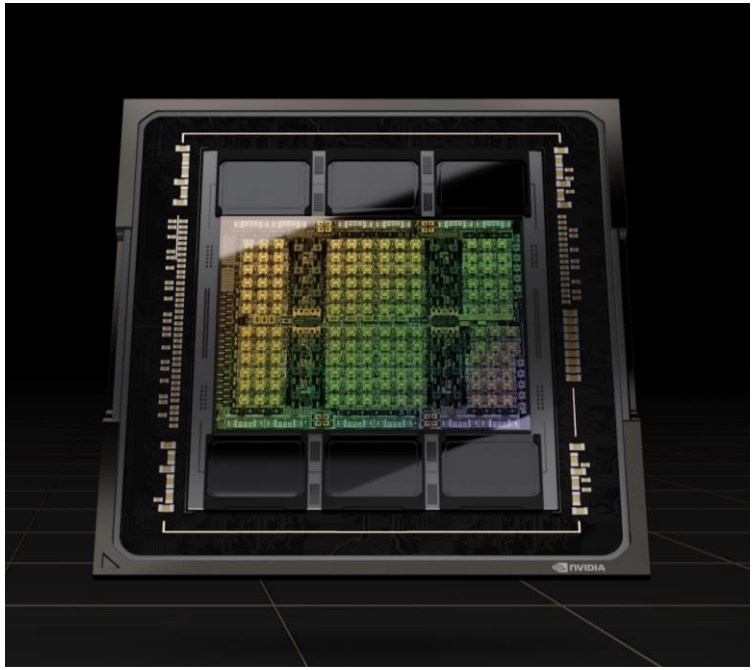
SAN JOSE, CA, USA
MARCH 4-7, 2024

Asynchronous Logic sign-off beyond CDC

accellera SYSTEMS INITIATIVE

Synchronous vs Asynchronous Logic Verification

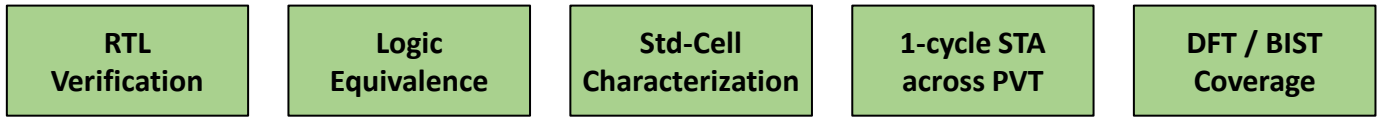
- Scale of Async logic usage
- Large variation across top level blocks
- 10’s to 100’s of clock domain
- 100’s to >100K synchronizers
- 1% to 70% flops with async resets



Source: www.realintent.com

Synchronous vs Asynchronous Logic Verification

- Synchronous paths are >99% of typical designs
- Relatively SAFE from Metastability/Randomness
- Verified efficiently with high-confidence using the Core ASIC sign-off flows



Source: www.realintent.com

Synchronous vs Asynchronous Logic Verification

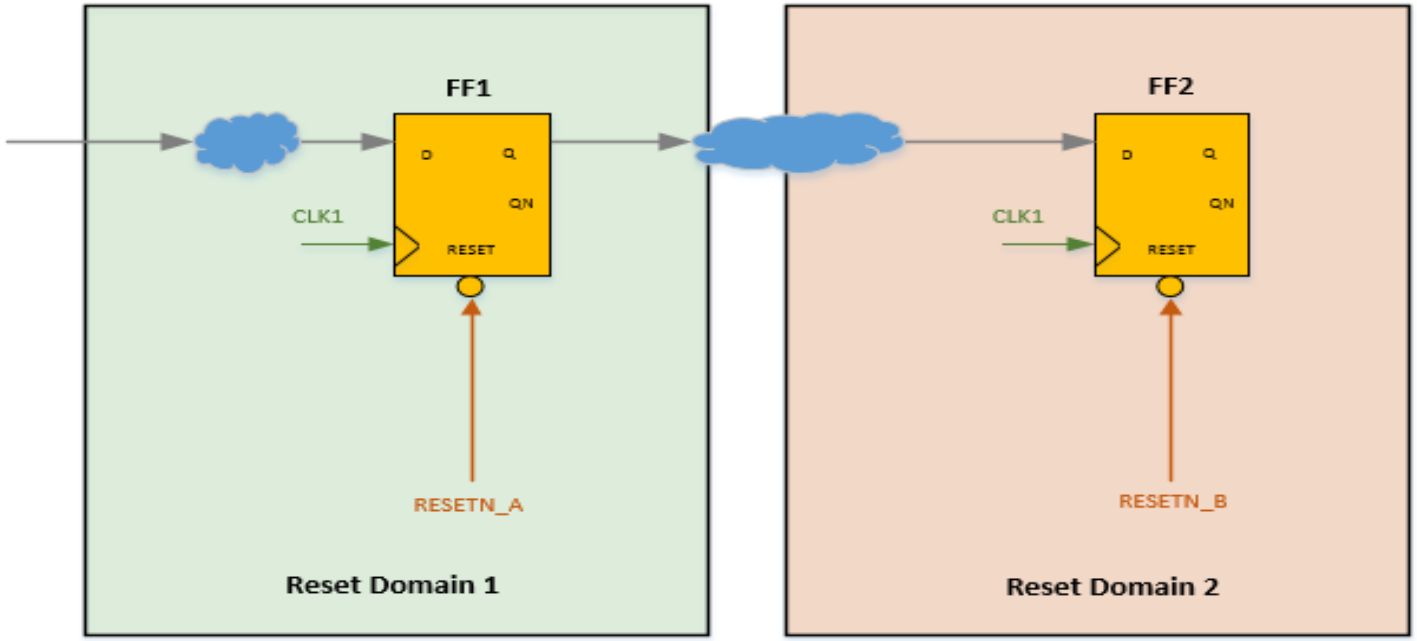
- Asynchronous paths are <1% of typical designs
- Significant RISK of Metastability/Randomness bugs
- Verification requires many specialized flows beyond just structural CDC

Std RTL Code Generators	Sims with Sync Randomization	RTL CDC	Netlist CDC	Async Timing across PVT
Std-Cell Synchronizers	Sims with C/RDC Assertion Checks	RTL RDC	Async Glitch Checks	Sync MTBF across PVT

Source: www.realintent.com

What is RDC Analysis?

Even one Asynchronous reset in the design can cause RDC problems

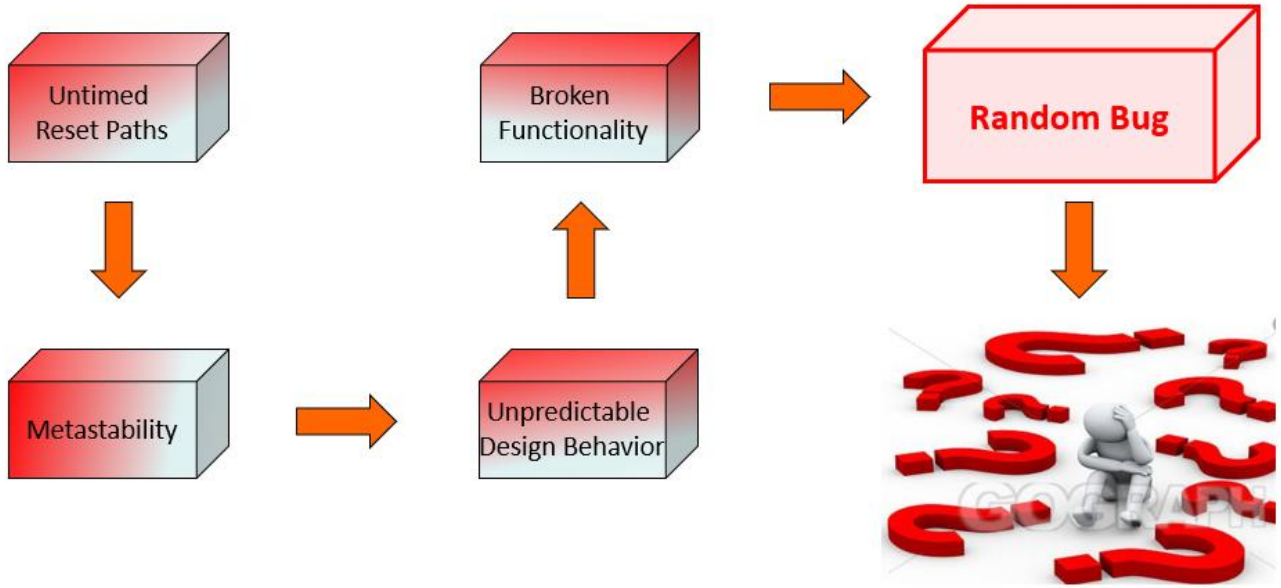


To reset only the faulty logic, several localized reset domains needed e. g. Automotive

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

Why is Reset Domain Verification Needed?

- RDC issues are less likely, but they do occur
- RDC issues are extremely difficult to debug in silicon
- Multiple reset types and their interactions multiply risk



Why RDC?

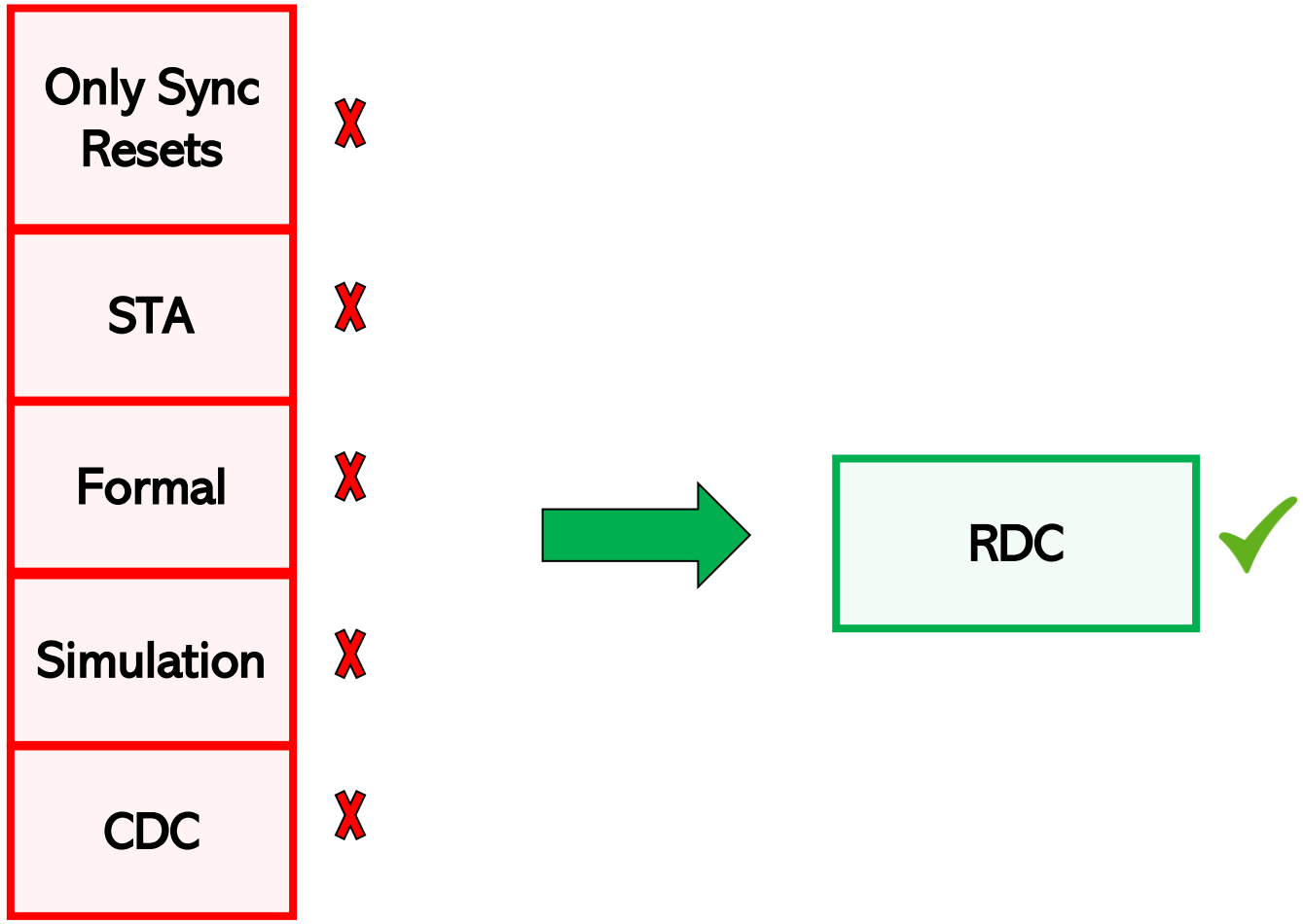
- Why RDC Signoff is important now?
  - Number of software resets increasing, possible some parts of design under reset while some parts in functional operations
  - Different power domains need different resets
- Isn’t reset controller logic already designed so these problems don’t occur
  - Yes but none of the flows STA, functional verification catch for these specific issues to ensure sign-off. Using RDC tool is only reliable way to safely verify reset logic is designed to ensure without metastability issues

Why RDC?

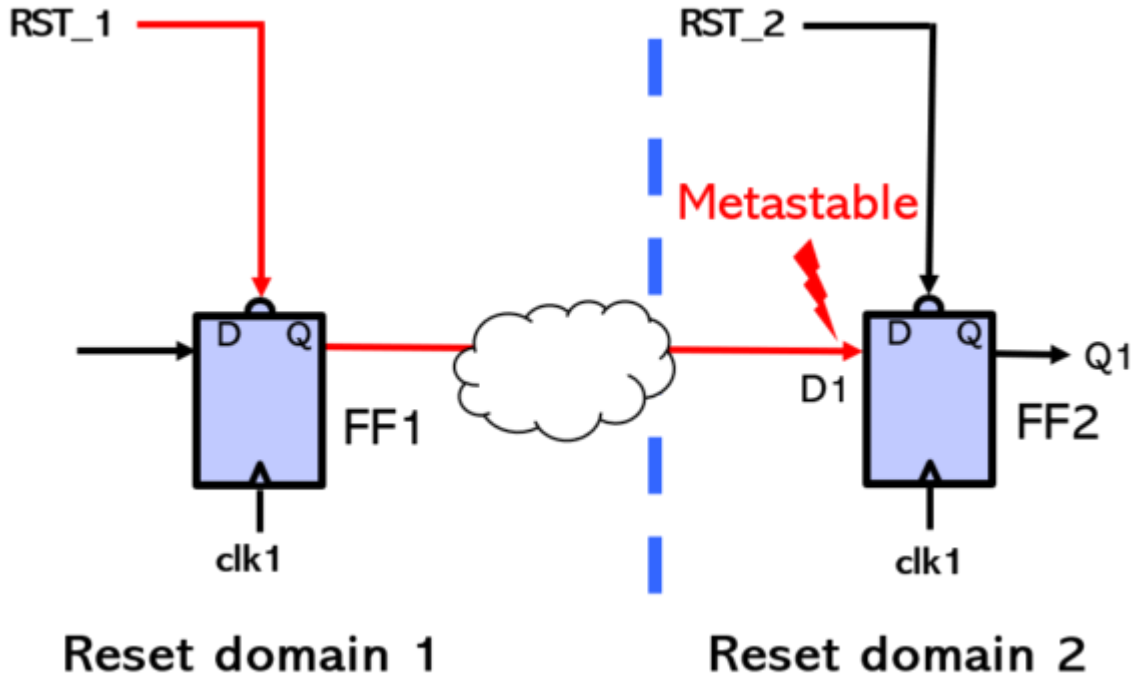
- Does CDC tool cover RDC
  - No, CDC tool looks at asynchronous paths. RDC failures can occur between synchronous clock domains also and are not same as CDC failures.
- Why we have not seen failures yet?
  - Unlike CDC frequency of reset operation is much less than clocks
  - The reset effect has to propagate through functional logic and is dependent upon state
  - Depends on actual delays, so may not always show as error but intermittent failures in some chips (lower yield)
- RDC issues HAVE led to chip failures in multiple design houses!

No substitutes for RDC Analysis

Specific solution needed to pinpoint unsafe paths

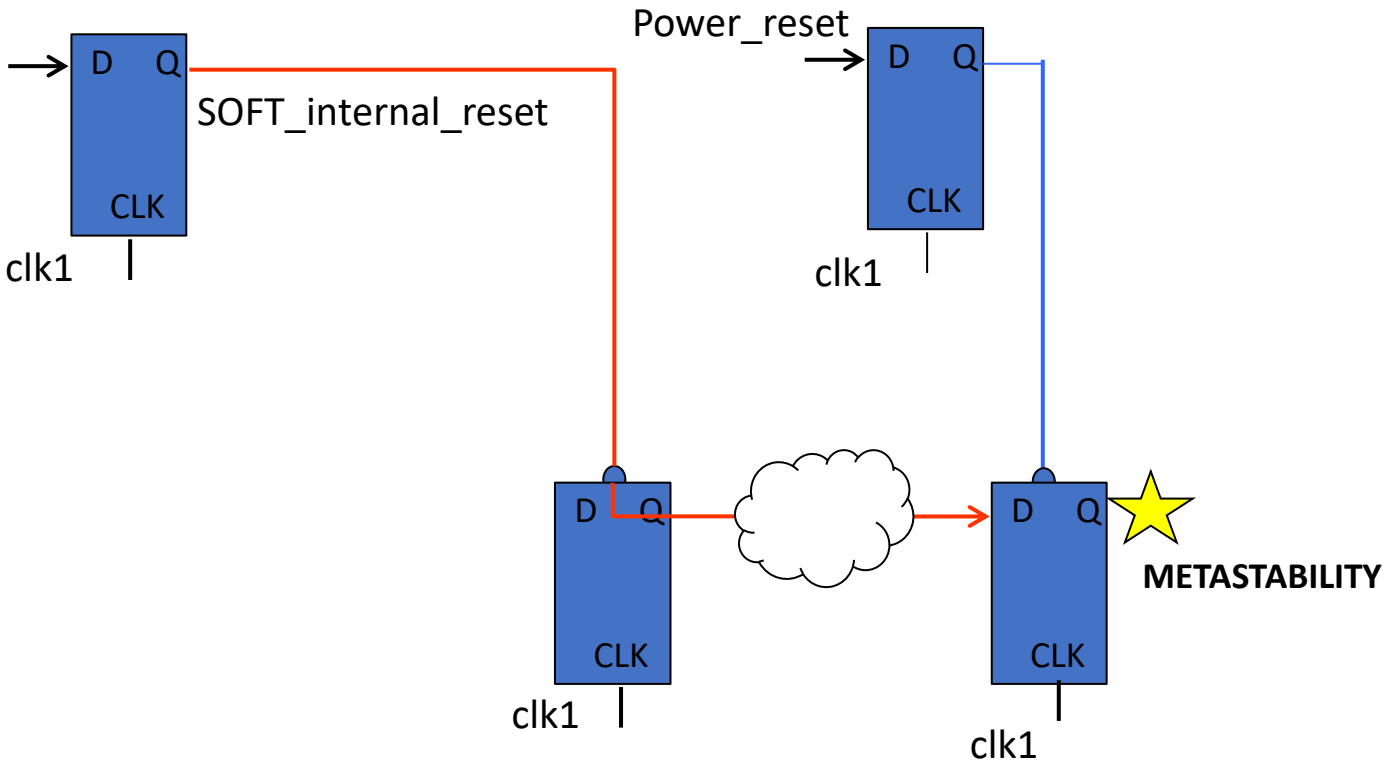


RDC ERROR – Mobile SoC



Problem	RST_1 assertion creates an ***untimed*** path. Fault **might** be detectable during gate-level sims but no guarantee

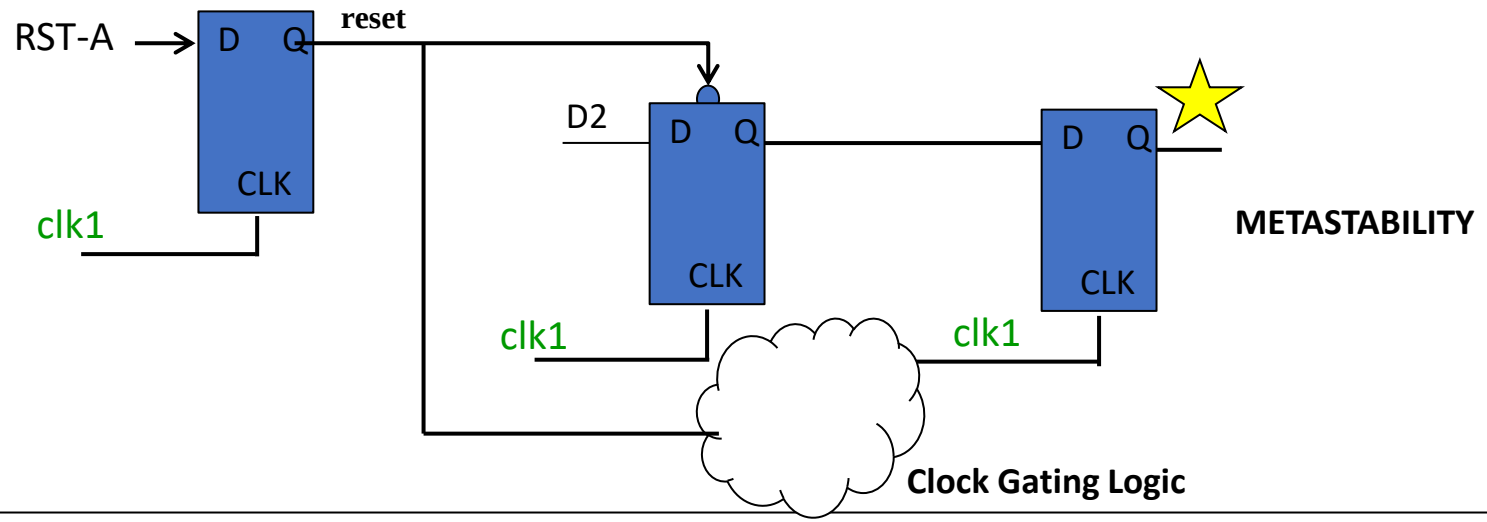
RDC ERROR – MIPI MPHY



Problem SOFT_internal_reset assertion while Power_reset is de-asserted assertion creates an ***untimed*** path. Fault **might** be detectable during gate-level sims but no guarantee

RDC ERROR – Modem IP

- Design assumption is to have clock gated at MFs prior metastability can pass to MF.
- Simulation tool did not fail, as well as seen that clock gated on the time.
- Simulation “ticks” based and depends on internal events , so there is a race between signals and in simulation gate closed “tick”



Problem reset assertion creates an ***untimed*** path.
Fault **might** be detectable during gate-level sims but no guarantee

Reducing Noise in CDC and RDC

- Structural CDC and RDC are inherently noisy
- Typical early reports: >10K warnings for functional blocks, >10M for top-level blocks
- Usually only a handful of “real bugs” are in the mix
- Need to reduce noise so designers aren’t overwhelmed (goal is “zero noise”)

Reducing Noise in CDC and RDC

- Workflows can help reduce the noise
- Discourage Waivers, Encourage Constraints
  - Use project-level constraints to cover async logic from Std Code Generators
  - Constraints can be checked for consistency, validated in simulation
- Run at multiple Hierarchy levels
  - Fifo-level, Block-level, Top-level, Chip-level
  - Supports “left-shift” goals
  - Use hands-off batch-mode regressions to track CDC/RDC health
- Combine modes when it makes sense
  - Collapse multiple clock modes into one set of reports by enabling multi-clock propagation
  - Support separate runs/reports/workflows for issues owned by different design teams

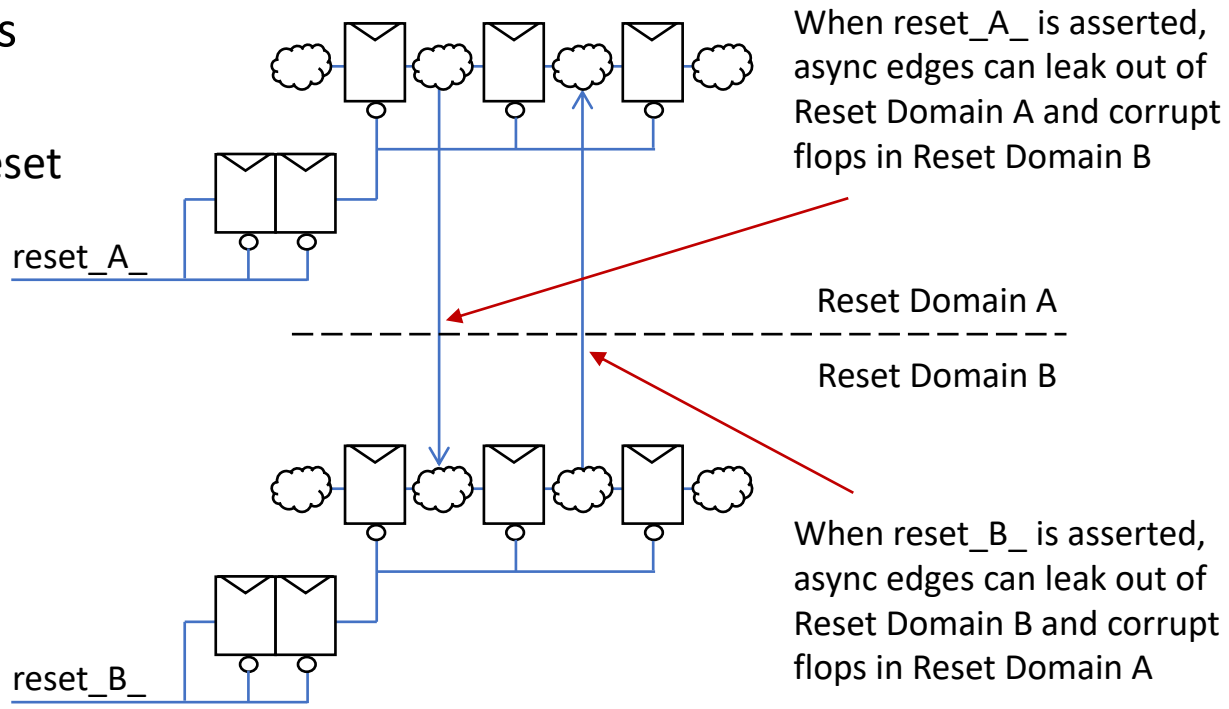
Simulation Checks for CDC/RDC Assumptions

- RTL sims can be augmented to help verify CDC/RDC constraints and async logic behavior
- Instantiate “anchor points” in the RTL with standardized naming conventions
- Enforce special sim behavior at anchor points (randomized delay, X insertion)
- Tie constraint check assertions to anchor points
- Constrain optimization flows to maintain anchor points through to tapeout netlists

Async Reset Checks

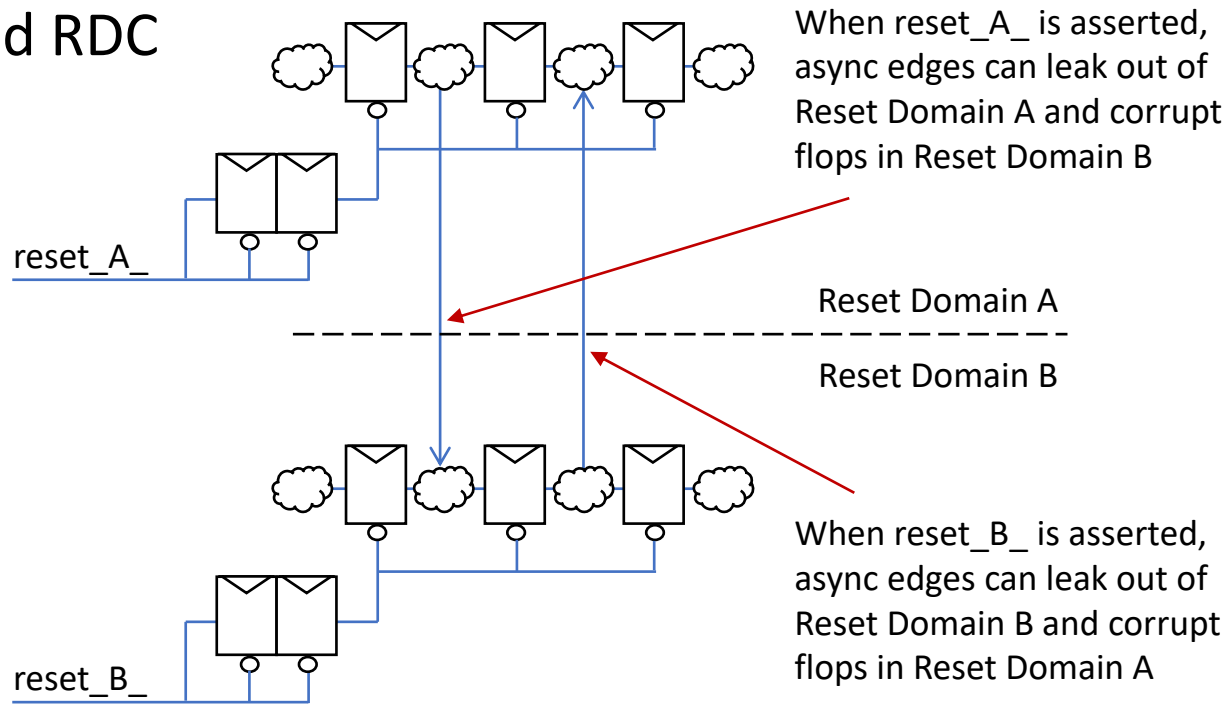
- Async Resets cause an explosion of async edges
  - From the reset flops themselves
  - From downstream flops if clocked and not in reset
  - From further downstream flops, and so on

- Async edges must be contained
  - By reset clamps
  - By reset sequencing
  - By stopping clocks to non-reset flops
  - By changing logic to sync reset



Async Reset Checks

- Async Reset Signoff needs both CDC and RDC
  - RDC for reset assert
  - CDC for reset de-assert
- RTL bugs found with MERIDIAN RDC
  - Async edges leaking into adjacent logic
  - Race conditions between parallel resets
  - RDC paths added by late design changes



How to Not Break a Chip

- Use standard plugins whenever possible, don’t reinvent the wheel for async interfaces
- Run simulations with randomizing synchronizer models
- Run CDC and RDC early and often, and after late ECOs
- Check CDC and RDC constraints with simulation assertions
- Don’t fall for “these async paths worked on the last chip so the design must be fine”
- Don’t assume that equivalence checks will flag glitchy logic optimizations
- Don’t assume that correct async logic in synthesized netlists stays that way in layout
- Check that synchronizer depths will meet MTBF goals across PVT
- Run async timing checks to cover async skew assumptions across PVT
- . . .

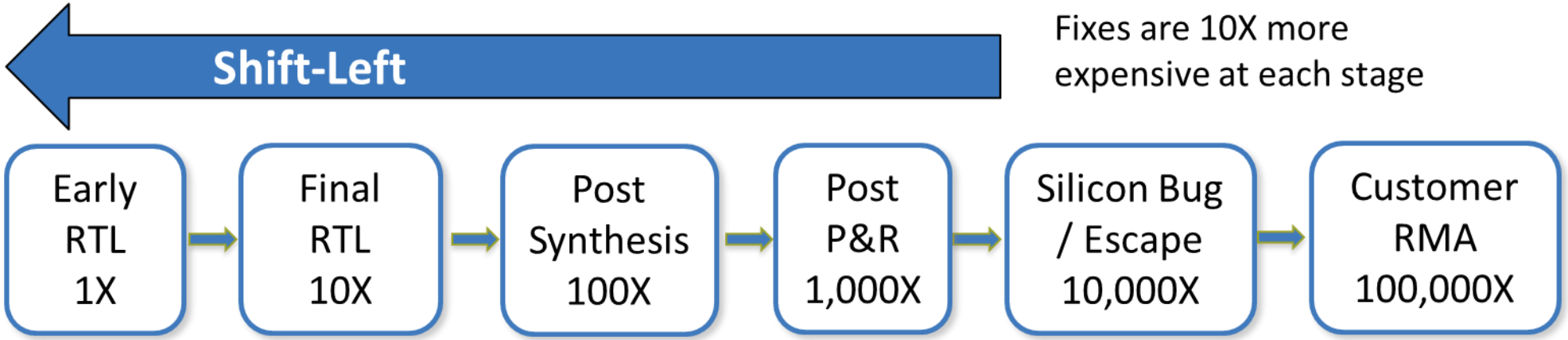
2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

DFT Compliance, checking and enabling shift left using static sign-off

accellera
SYSTEMS INITIATIVE

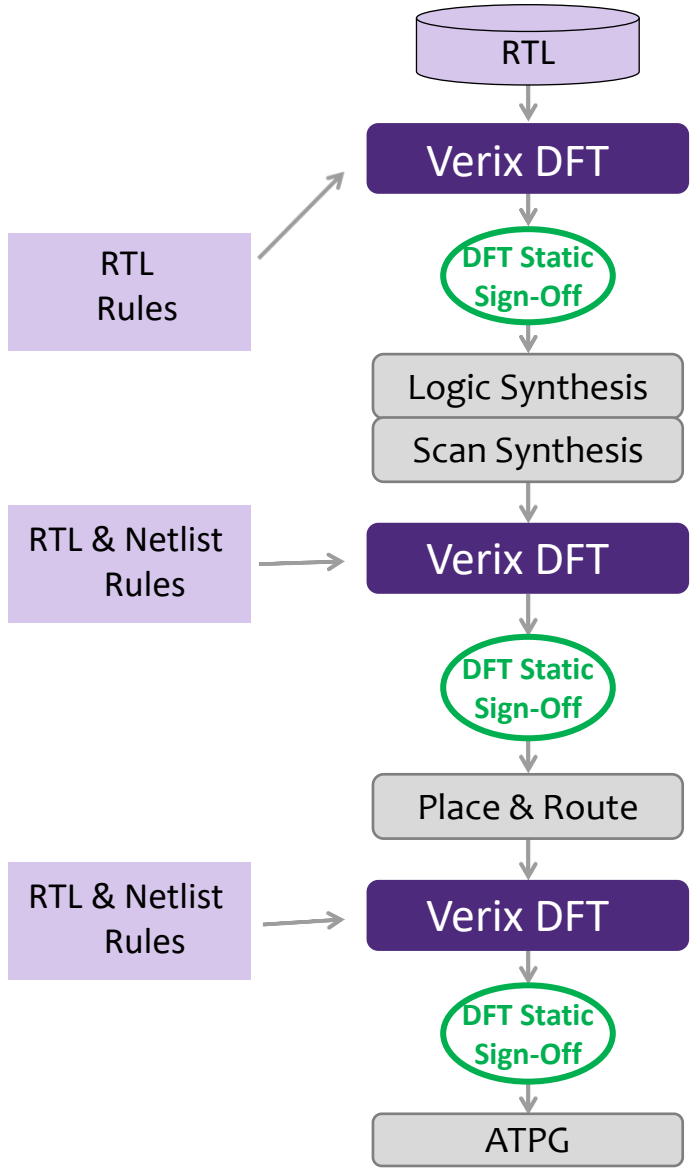
DFT Challenges and Trends

- Shift Left
  - ATPG typically occurs after P&R – but fixes are 10X more expensive at each stage
  - ATPG attempted before P&R, but overkill



Shifting Left with DFT Static Sign-off

Design RTL	Prepare for scan synthesis, to ensure RTL is scan friendly
Gate-level Netlist with Scan Chains	To verify correctness of scan implementation
Netlist with Scan Chains Reordered	To ensure scan reordering does not create issues



Approach to DFT Static Sign-off

- Multimode
  - Multiple sets of rules per run, reducing setup time and speeding up runtime
    - Multiple ATPG partitions, multiple sets of constraints per partition
- High capacity and performance
  - Multi-million gate design in minutes
  - Low peak memory footprint
- Specialized, fine-grained rules
  - High coverage at all design stages
  - Faster debug and root cause analysis
- Low noise
  - To minimize false positives and error duplication
- Fits easily with existing flows and DFT/ATPG tools

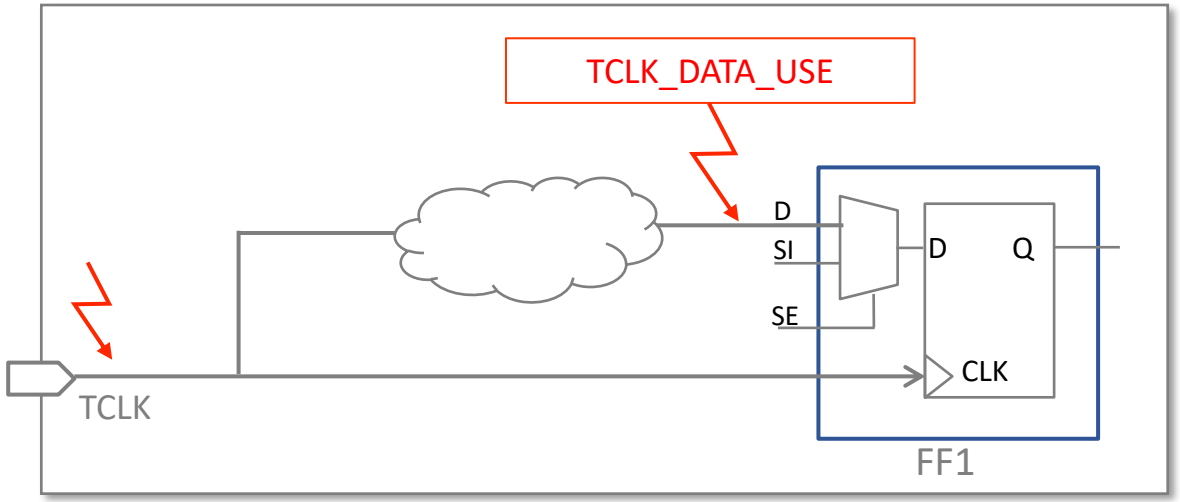
Design issues that affect Fault Coverage

- Uncontrollable test clocks
- Uncontrollable/incorrectly constrained async set/reset pins of flip-flops
- Loss of connectivity/controllability between signals, due to –
  - Design bugs, such as undriven/unloaded nets, combinational feedback loops, and tristate busses with potential for bus contention
  - Specification errors in test mode constraints, such as incorrect or insufficient test mode constants

TCLK_DATA_USE (Category: CLOCK)

Test Clock Drives Data Input of FF in Scan Hierarchy

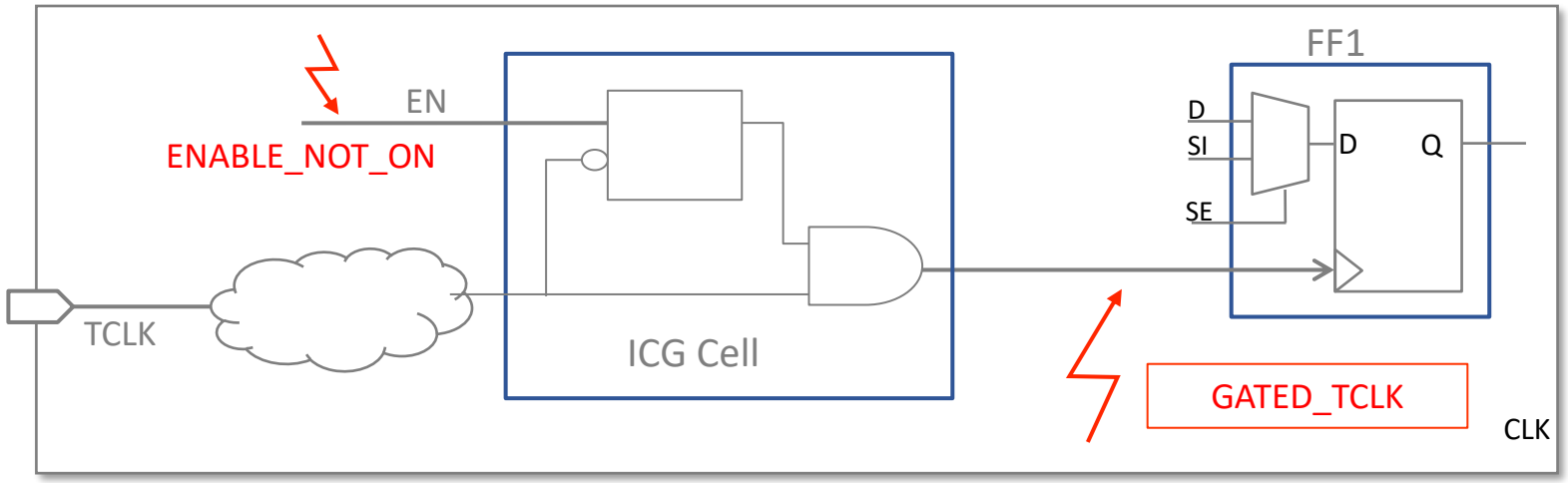
- Cause:
  - Test clock drives data or async set/reset inputs of FFs (e.g., FF1) in a scan hierarchy (RTL/Gate)
- Impact:
  - Metastability during capture, since data transitions happen at or close to the clock edge, hence affected flip-flop data can get corrupted, causing loss of both controllability and observability, hence lower fault coverage



GATED_TCLK (Category: CLOCK)

Test Clock Not Enabled During Scan Shift

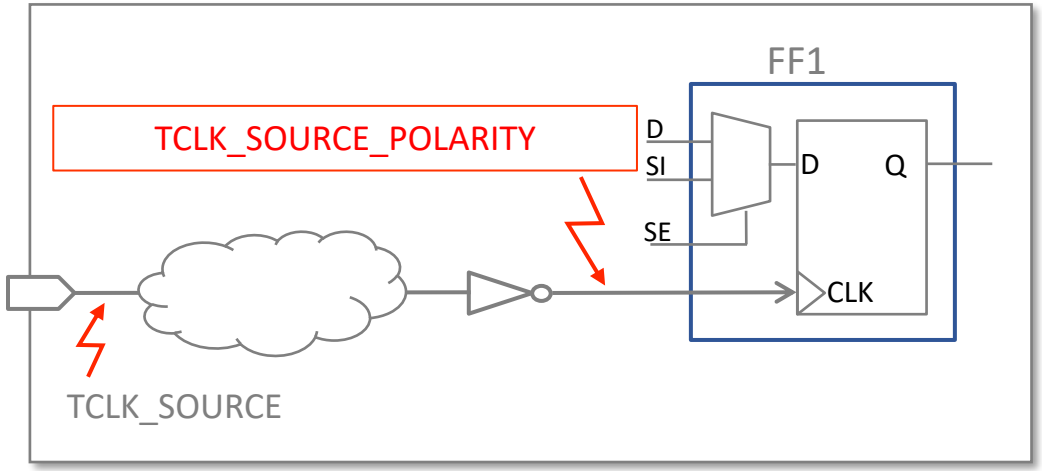
- Cause:
  - Gated test clock is not enabled during scan shift (RTL/Gate)
- Impact:
  - Failure to load scan chain during shift, causing loss of controllability (observability) of the affected flip-flop during scan load (unload), lowering fault coverage



TCLK_SOURCE_POLARITY (Category: CLOCK)

Test Clock & Test Source Clock Have Different Polarities

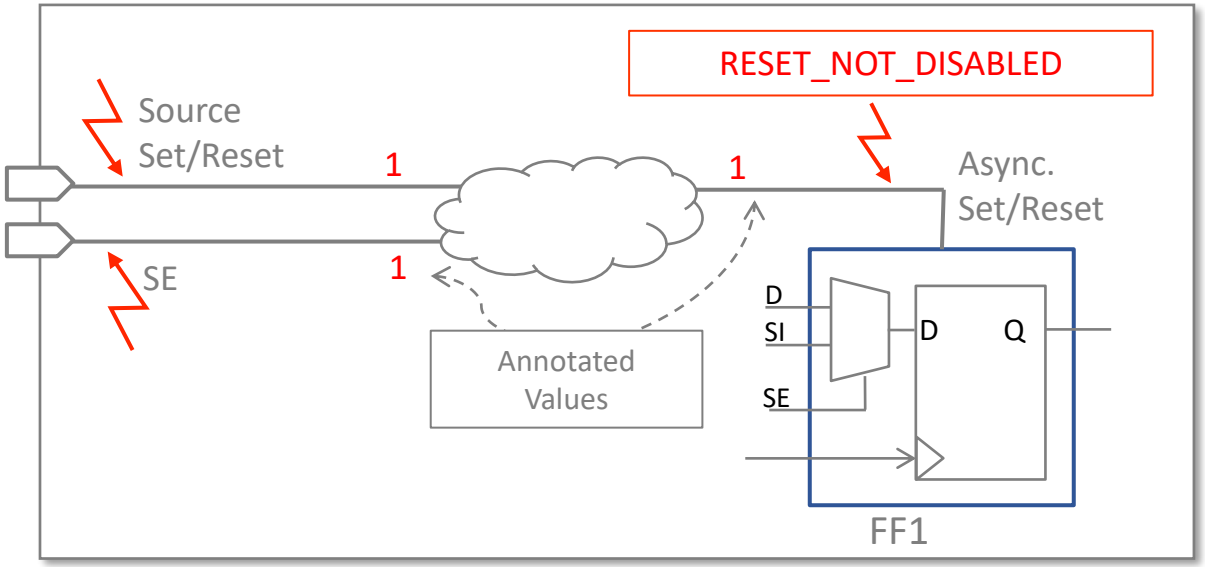
- Cause:
  - Test clock (e.g., CLK in FF1) does not have the same polarity as the test source clock (RTL/Gate)
- Impact:
  - Non-controllable test clock during scan test, causing loss of fault coverage



RESET_NOT_DISABLED (Category: ASYNC_RESET)

Async. Set/Reset Not Disabled During Scan Shift

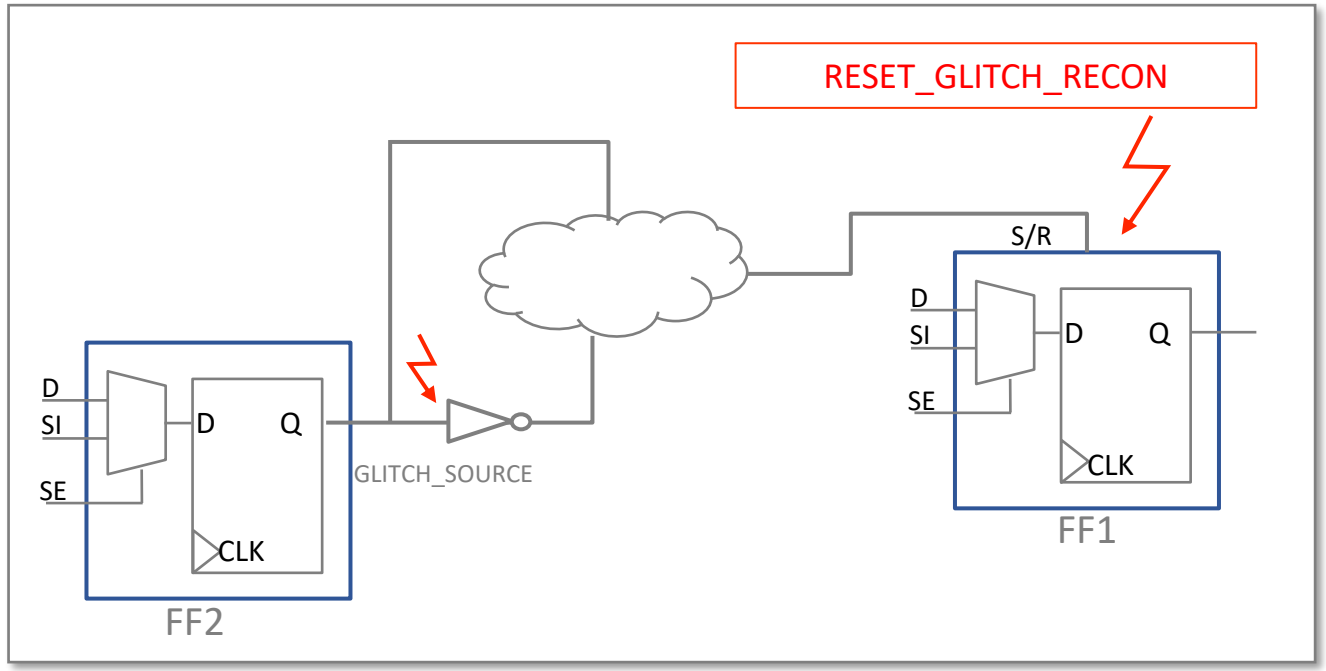
- Cause:
  - In shift mode, there is at least one path to propagate asserted value (or X) from source Set/Reset to FF Set/Reset pin, where FF belongs to the scan hierarchy (RTL/Gate)
- Impact:
  - Scan load/unload Shift data erased by non-disabled asynchronous set/reset signal, causing loss of fault coverage



RESET_GLITCH_RECON (Category: ASYNC_RESET)

Reconvergence of Set/Reset with Opposite Polarity

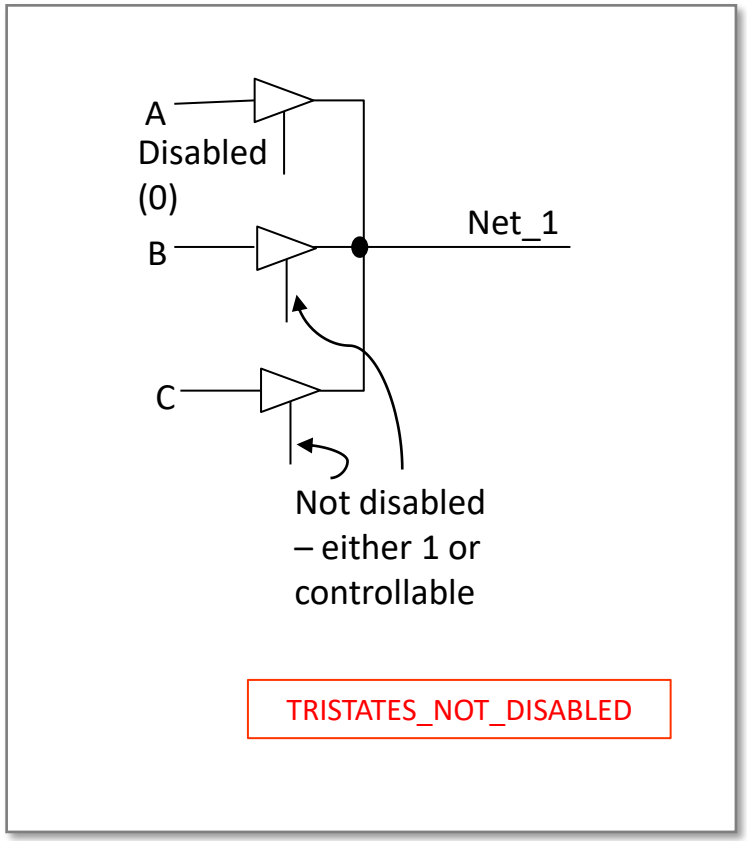
- Cause:
  - Signal re-converges with itself with opposite polarity and drives the set/reset pin of FF (e.g., FF1) in a scan hierarchy (RTL/Gate)
    - Self-loop between the output of FF2 to its D-input can be either ignored or checked; if ignored, then the flip-flop is considered as a glitch source, otherwise it is not.
- Impact:
  - Glitchy set/reset, FF not able to shift and/or capture test data, causing loss of fault coverage



TRISTATES_NOT_DISABLED (Category: CONNECTIVITY)

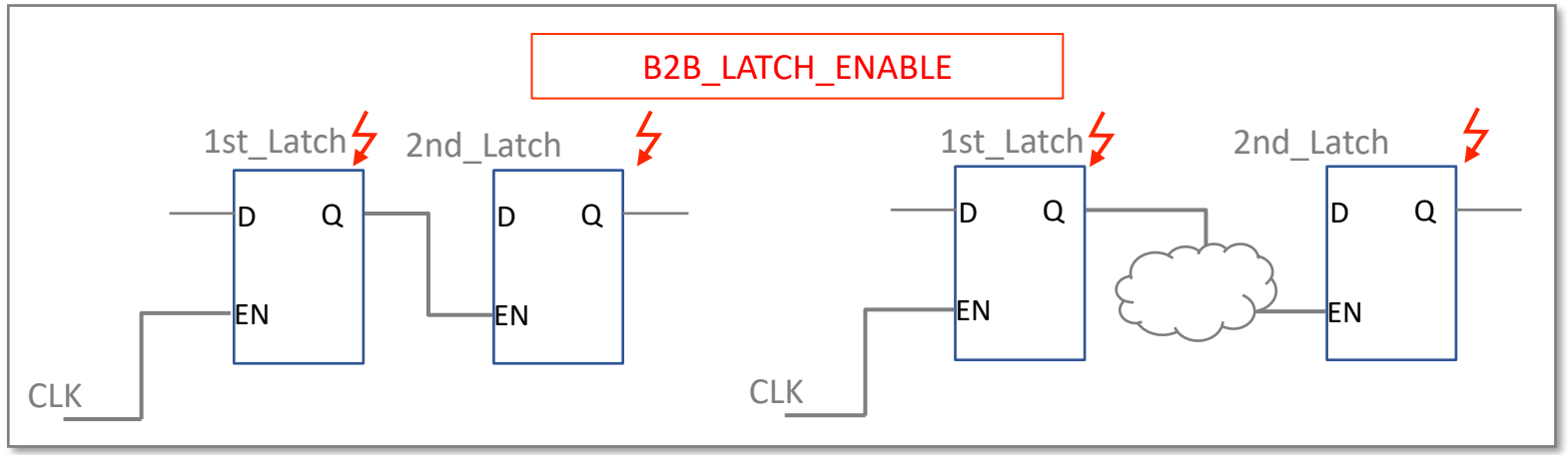
A net has multiple non-disabled tristate drivers

- Cause:
  - A design has >1 non-disabled tristate drivers. (RTL/Gate)
- Impact:
  - Loss of fault coverage due to unreliable shift and/or capture



B2B_LATCH_ENABLE (Category: SCAN_CHAIN)

- Cause:
  - Two latches are connected back-to-back with Q of one latch driving EN of the next latch. (Gate)
- Impact:
  - Test data loss while data is being shifted through scan chain
  - Timing problems



2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Architectural Compliance using static sign-off

accellera SYSTEMS INITIATIVE

Structured Design Methodologies Manage Complexity

Architecture implemented using functional components within a structure

Connectivity rules specify architectural component interconnections

Structured Design Methodologies Manage Complexity

Architecture implemented using functional components within a structure

Connectivity rules specify architectural component interconnections

- Bus protocols, power, debug logic, PD constraints, memory, DFT connectivity…
- Facilitates automatic design integration

Structured Design Methodologies Manage Complexity

Architecture implemented using functional components within a structure

Connectivity rules specify architectural component interconnections

- Bus protocols, power, debug logic, PD constraints, memory, DFT connectivity…
- Facilitates automatic design integration

Shift Left: Earliest possible efficient verification of a design step

Connectivity Checking: Efficient shift left verification of architecture compliance

System Requirements for Superior User Experience

- Convenient **Rules** Specification
  - Robust against design changes
  - Source/Destination exclusions target refinement
- Rules dictionary for full coverage
  - Instantiate/Activate relevant rules
- Support diverse connectivity principles

System Requirements for Superior User Experience

- Convenient Rules Specification
  - Robust against design changes
  - Source/Destination exclusions target refinement
- Rules dictionary for full coverage
  - Instantiate/Activate relevant rules
- Support diverse connectivity principles
- **Extremely fast runtime & capacity for entire SoC**
  - Without requiring black-boxing

System Requirements for Superior User Experience

- Convenient Rules Specification
  - Robust against design changes
  - Source/Destination exclusions target refinement
- Rules dictionary for full coverage
  - Instantiate/Activate relevant rules
- Support diverse connectivity principles
- Extremely fast runtime & capacity for entire SoC
  - Without requiring black-boxing
- **Customized debug**
  - Customizable reporting
  - Schematics with annotated attributes

High Capacity & Performance

- **Highest capacity** in industry
  - Block-level & full SoC-level sign-off
  - Black boxing available
    - But not required for large designs like other approaches

- **~10X faster**
  - Block-level analysis in minutes
  - Compared to hours for Tcl scripting & formal for comparable blocks

Methodology: Left Shift of Architectural Compliance

- CAD creates dictionary of standard rules
  - Minimal creation cost
- Users instantiate & activate relevant rules
  - Also add local custom rules
- Continuous integration for left shift
  - Fast runtime

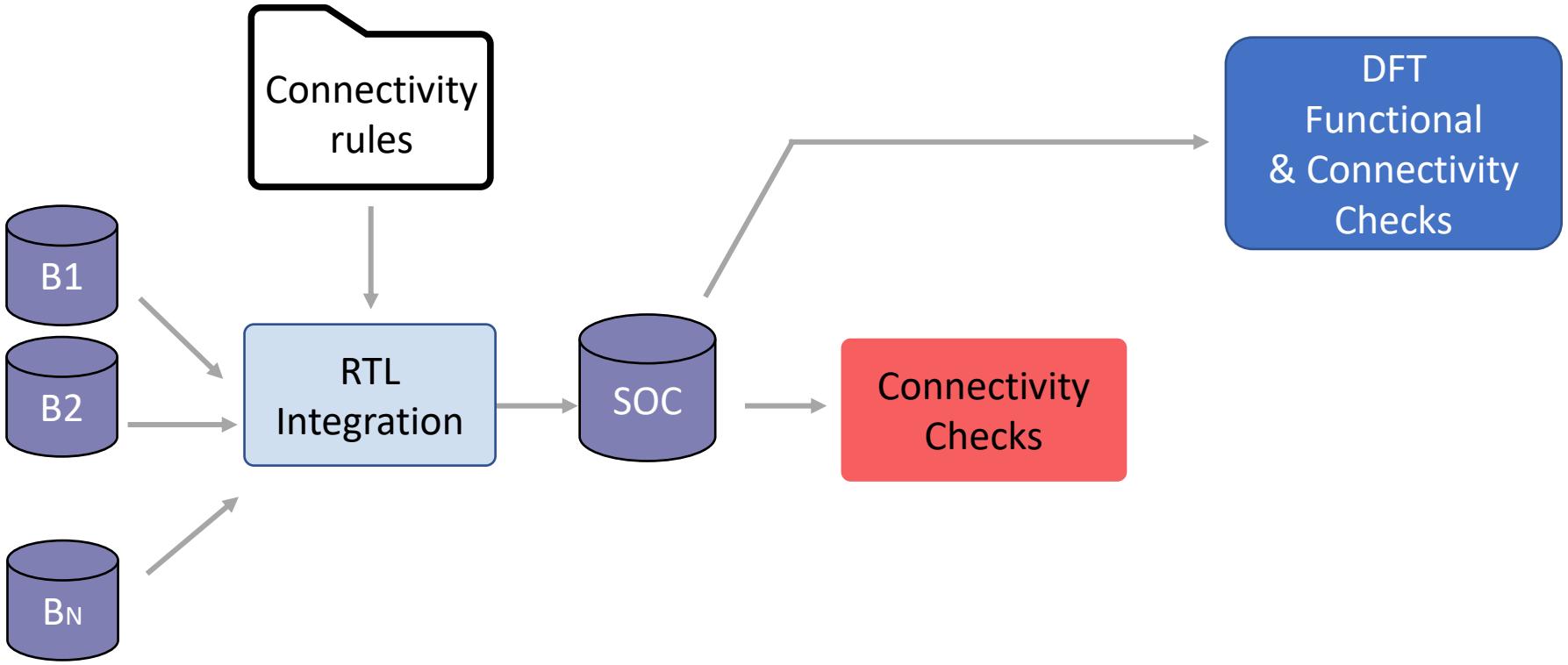
2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

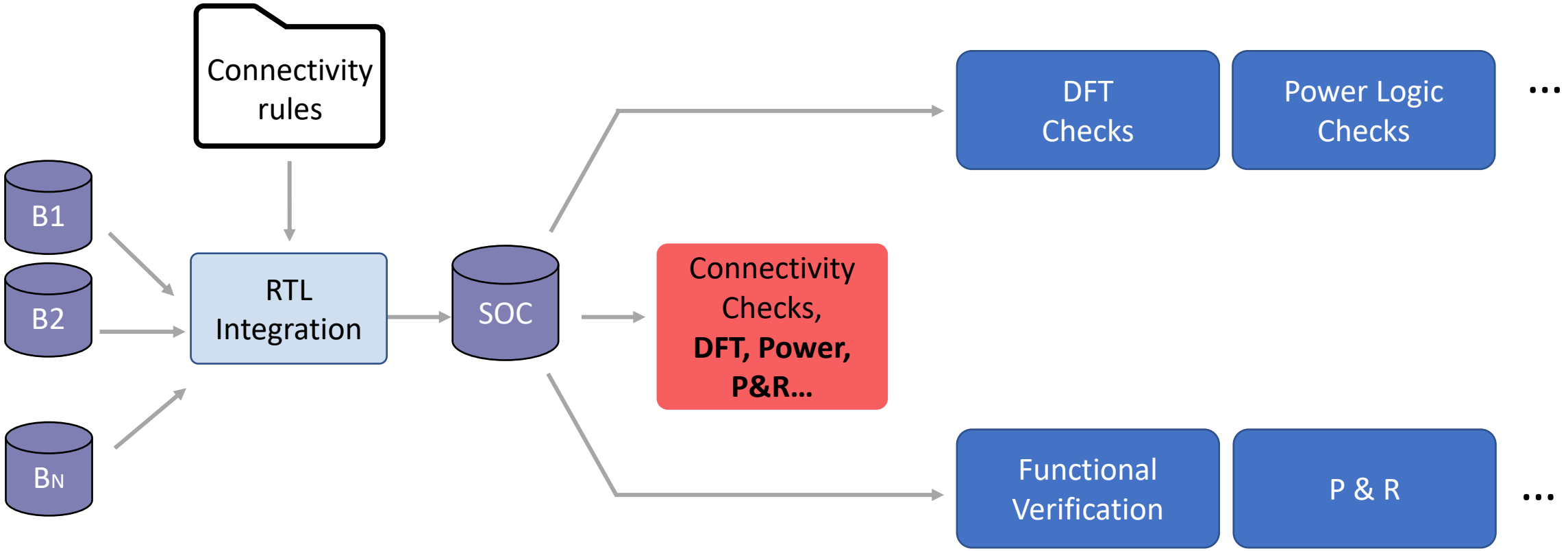
Connectivity checking using static-signoff

accellera SYSTEMS INITIATIVE

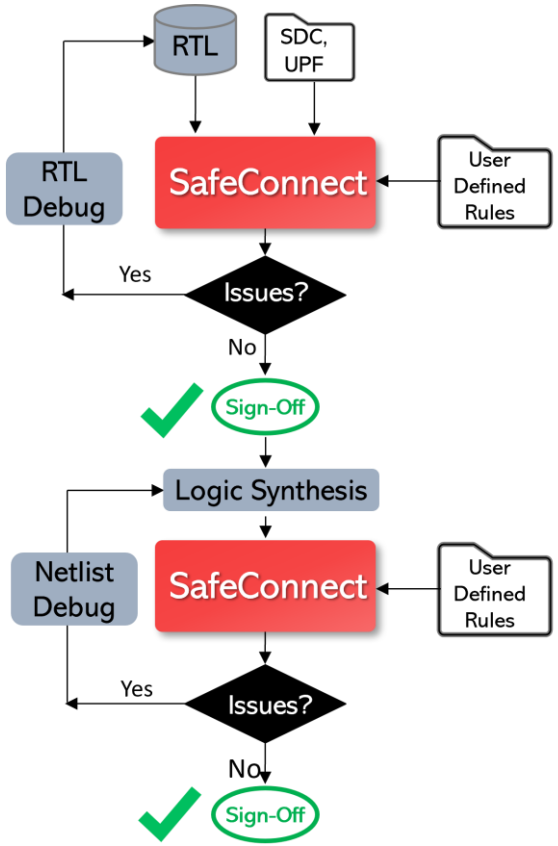
Connectivity Checking Shift Left In Action



Connectivity Checking Shift Left In Action

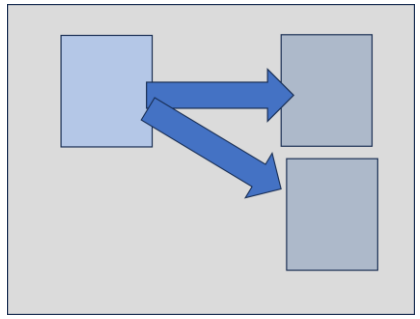


SafeConnect: RTL & Netlist Connectivity & Glitch Sign-Off

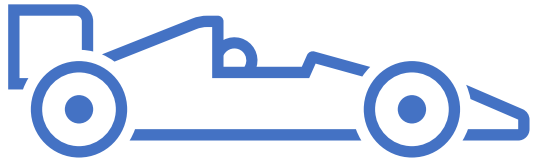


Static Connectivity Checking Requirements

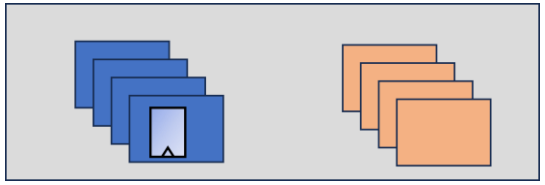
Diverse and Granular Checks



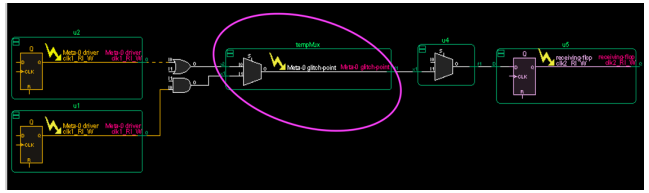
Full Chip Capacity



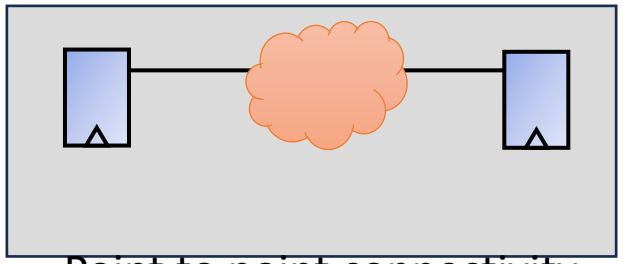
Ease of Specification



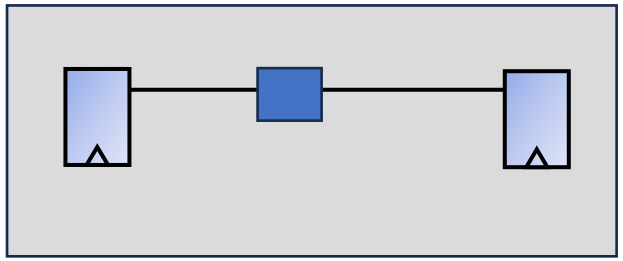
Ease of Debug



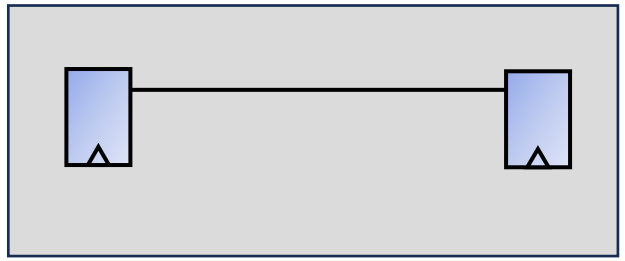
Diverse and Granular Checks- 1



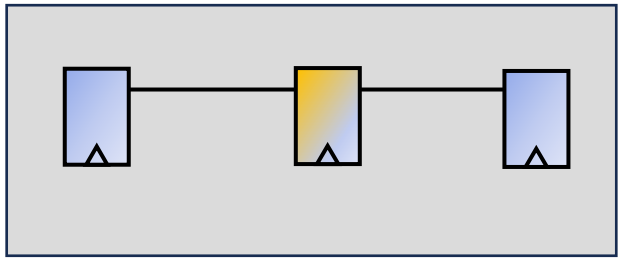
Point to point connectivity



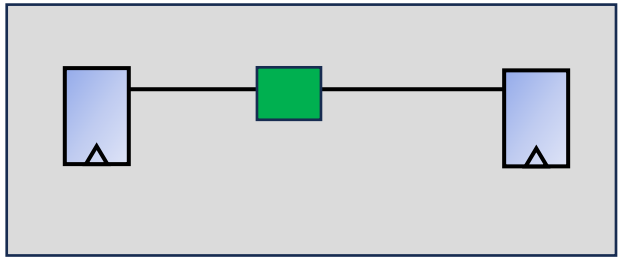
Point to point connectivity **through** objects



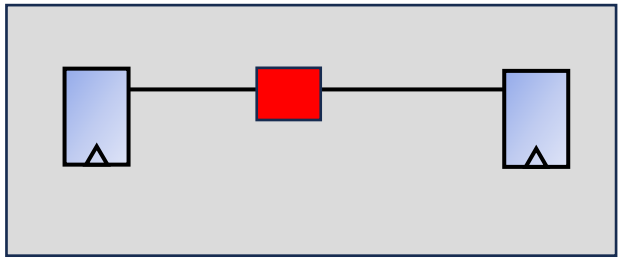
Point to point **direct** connectivity



Point to point
Connectivity through transparent
Sequential or modules/instances

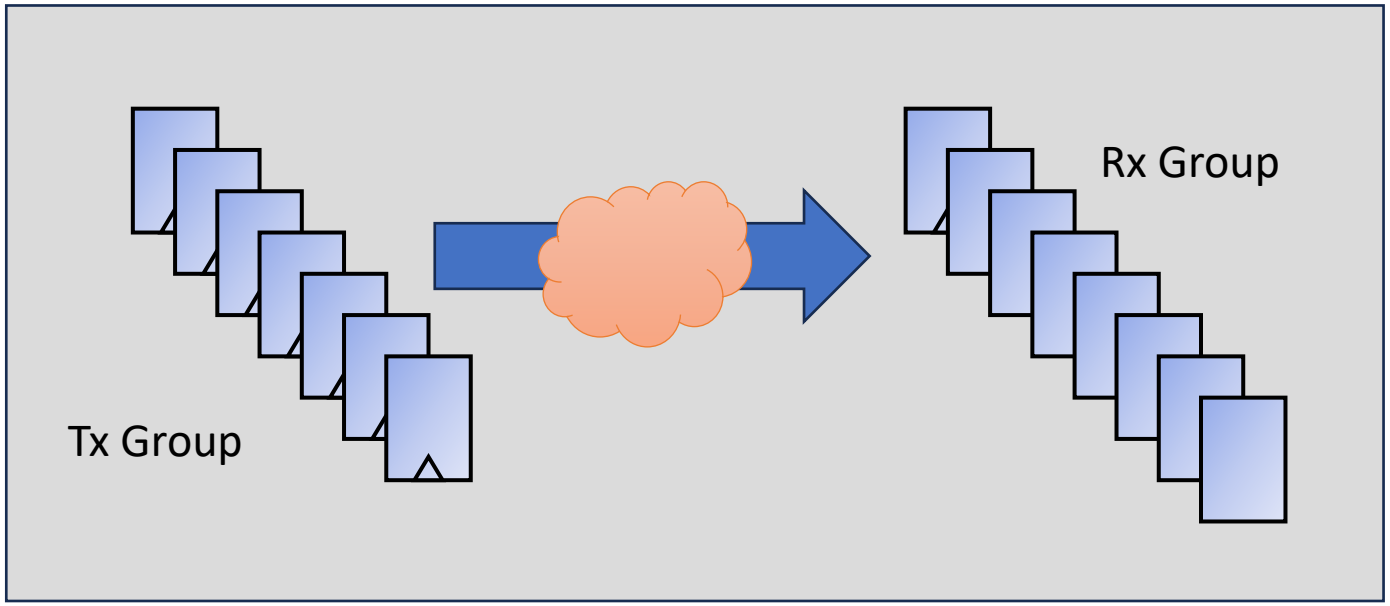


Point to point connectivity **allowed** objects



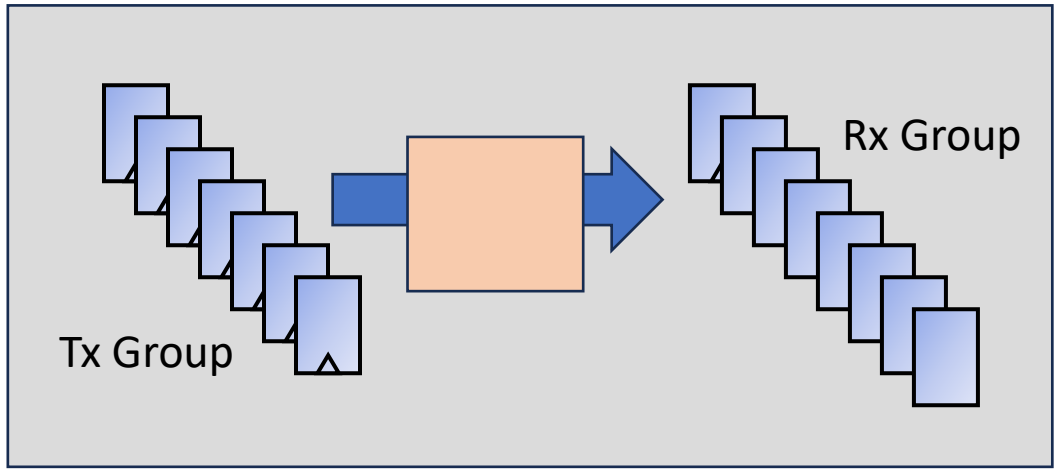
Point to point connectivity **disallow** objects

Diverse and Granular Checks- 2



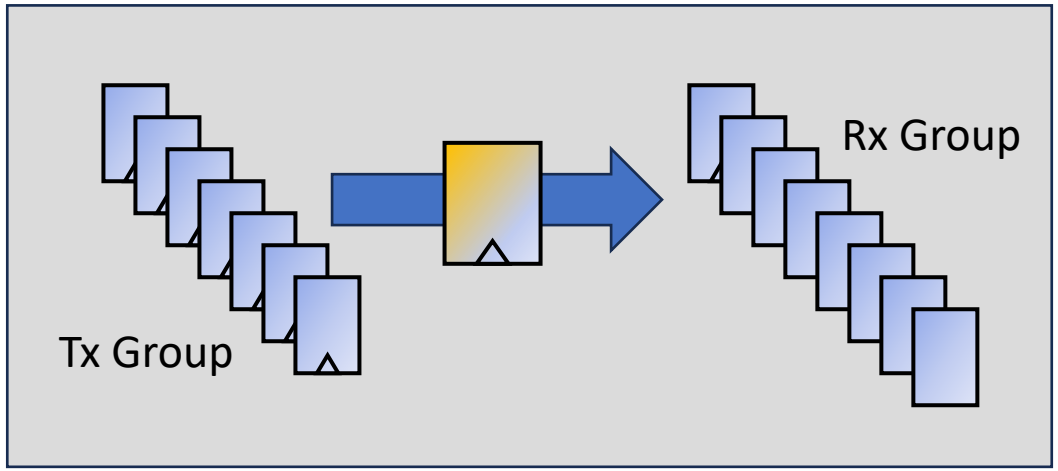
Each in Tx Group connected to
at least one in Rx Group and vice-versa
Can check whether only SOURCE_MISS
Or only DESTINATION_MISS or both

Diverse and Granular Checks- 3



Each in Tx Group connected to
at least one in Rx Group and vice-versa

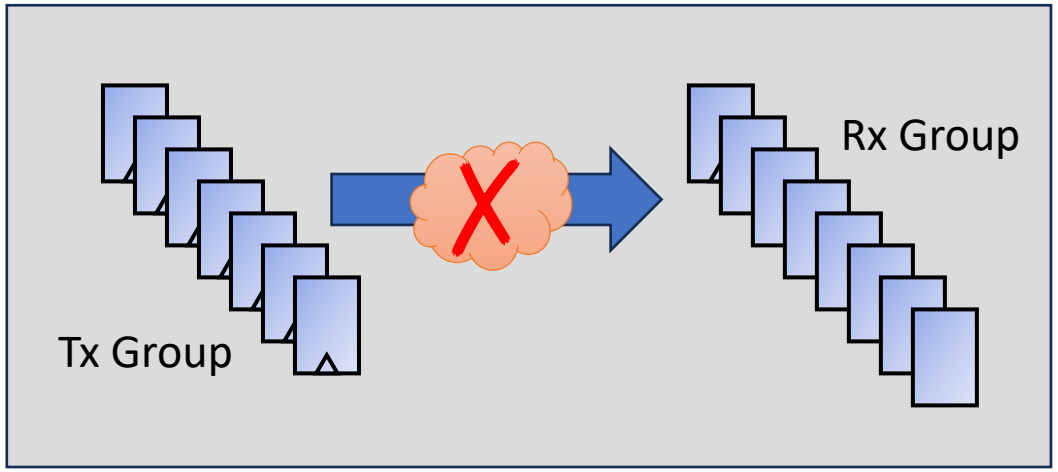
Through objects



Each in Tx Group connected to
at least one in Rx Group and vice-versa

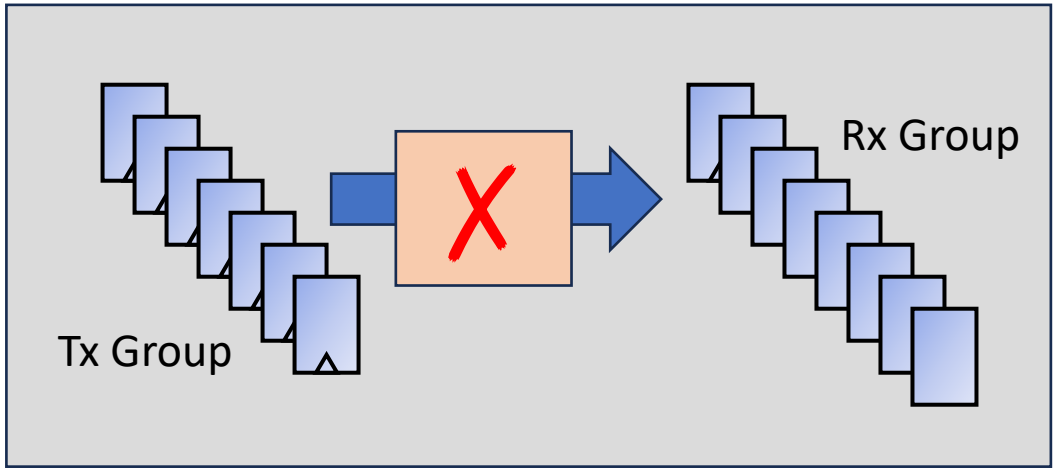
Transparent sequential
Or modules/instances

Diverse and Granular Checks- 4



Not connect

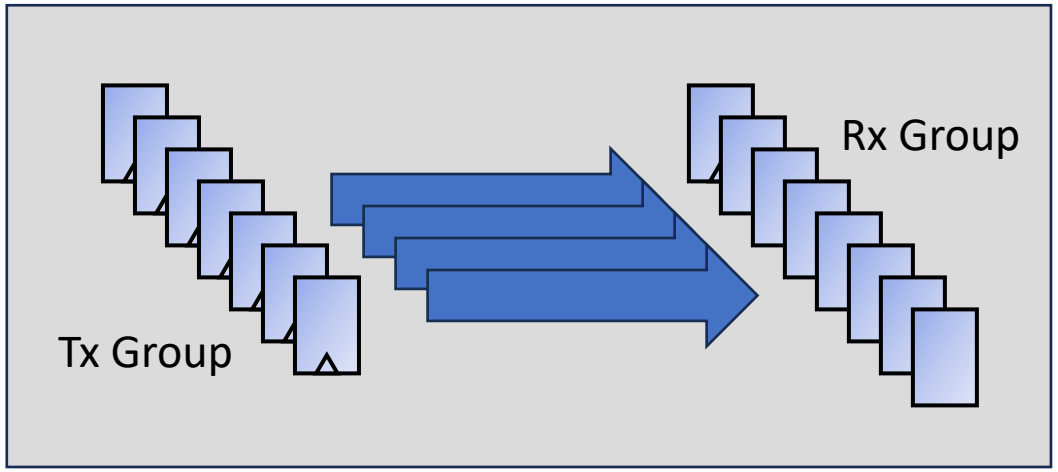
None of the sources drive destination



None of the sources drive destination

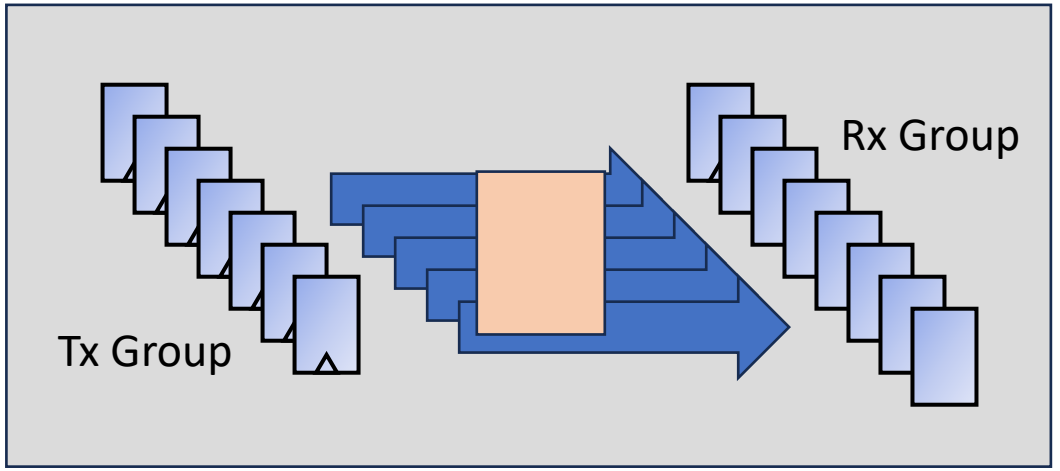
Through certain objects

Diverse and Granular Checks- 5



Full connect

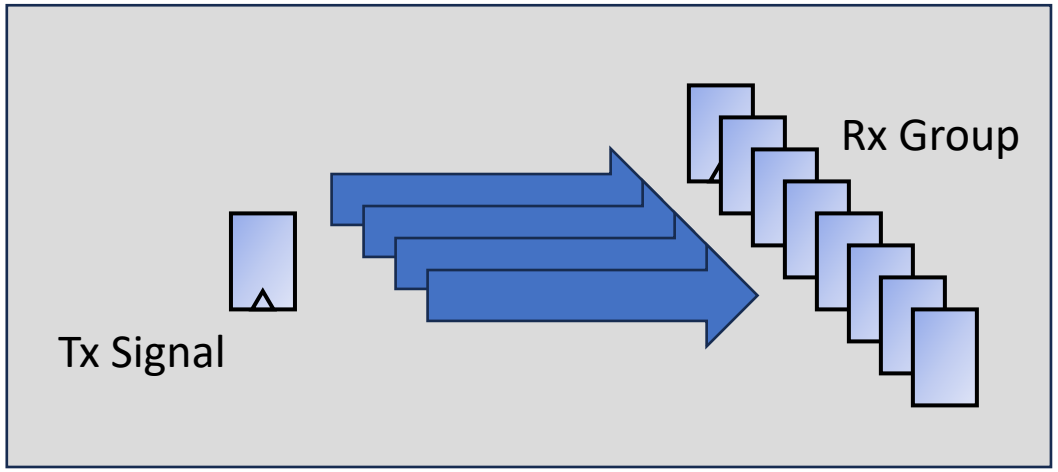
Each Tx is connected to All Rx and vice versa



Full connectivity

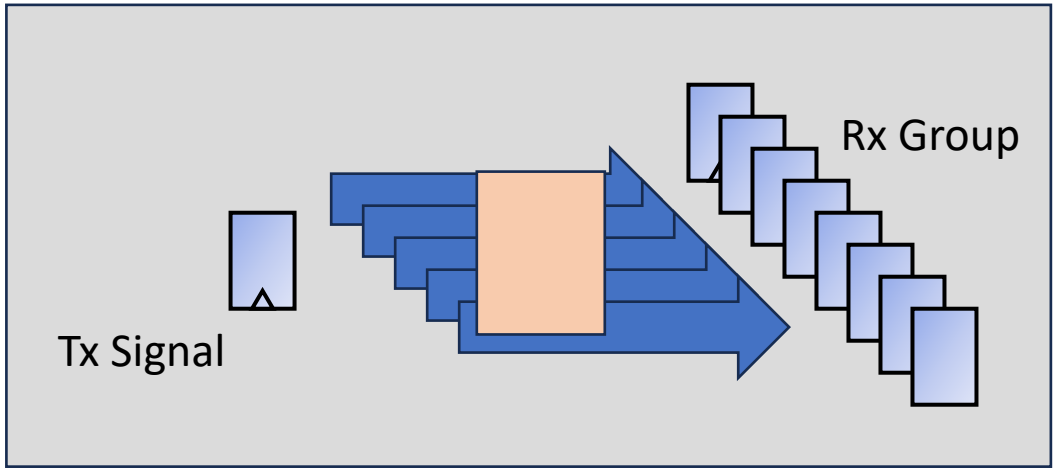
Through certain objects

Diverse and Granular Checks- 6



Only One Connection

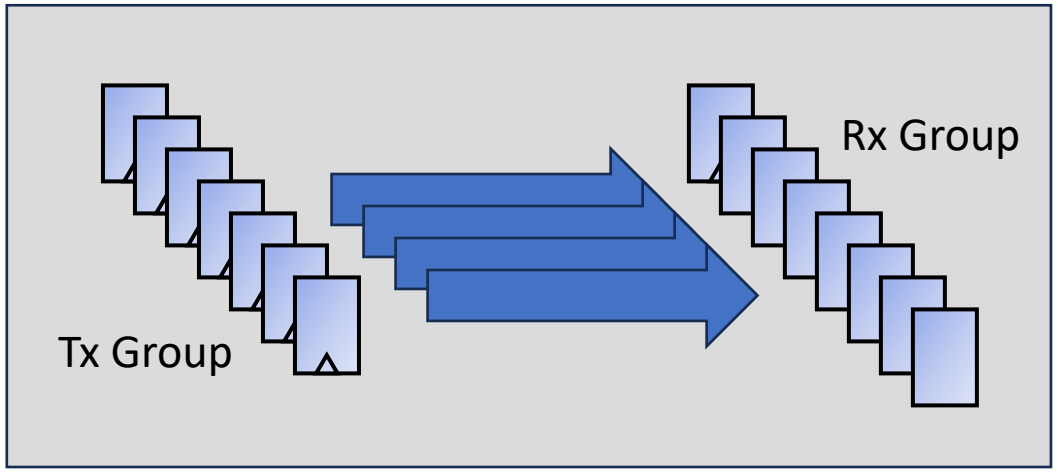
only one connection exists between Tx Signal and Rx Group



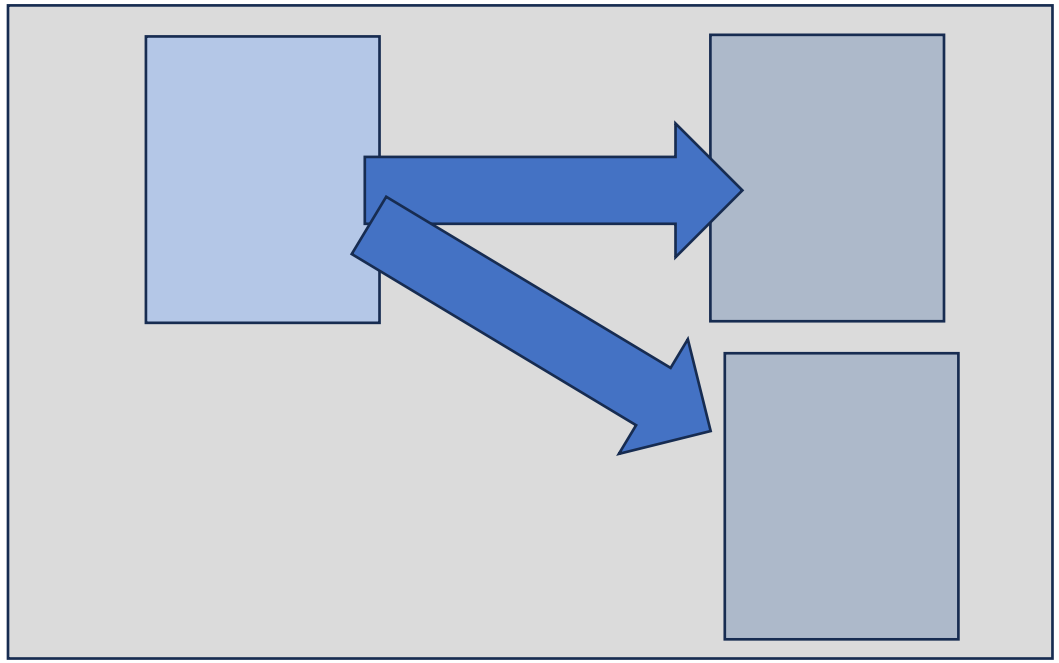
Only One Connection

Through certain objects

Diverse and Granular Checks - 7



Direct Connection between matching names

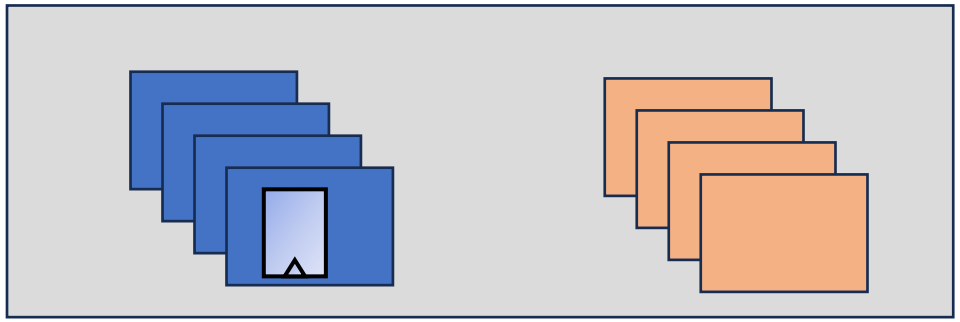


Full connectivity in Bus

At least one ordered connection exists between every bus / ordered signals in Tx list and every bus / ordered signals in the Rx list

If the bits or order inter-changed then it is also caught

Ease of specification



Easily specify Tx and Rx Groups

- All flops inside a module or instance
- Input/Output ports of modules or instances
- All modules but exclude some instances
- All flops or only flops with async reset, only flops with retention non-retention
- Type of object – pin/port/net
- Isolation cells
- Level shifters

Ease of specification - Example

- Just 3 commands can define a check at block or full chip level to avoid improper connectivity from non-retention reset to retention FFs
- Rule defining check can be easily enabled or disabled using **enable_rules / disable_rules** commands

```
create_group  –name  Non_Retention_Reset_Sub_Group_1
  –scope {I1  I2.I3 } –signals  { RST_* }
  –module Non_Ret_Mod –exclude_instances {u_inst1.u_inst2.*}

create_group  –name  Retention_FF_Sub_Group_1
  –scope { I4* }  –FF { retention }

set_not_connect
  –from_group Non_Retention_Reset_Sub_Group_1
  –to_group Retention_FF_Sub_Group_1
  –dont_trace_group  { CLAMP_CELL_instance_group }
  –rule SLSI_NONRET_RST_TO_RET_FF
```

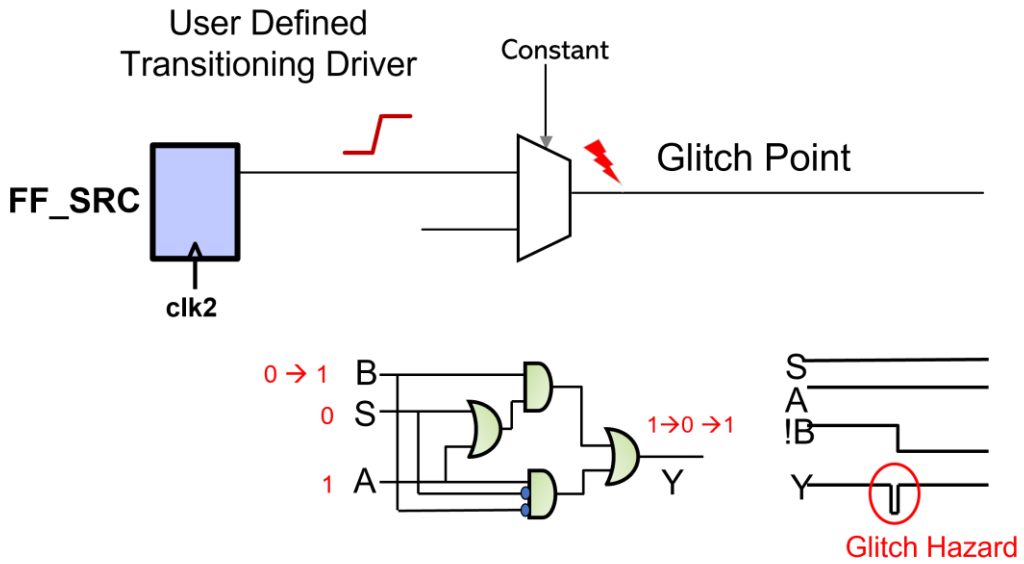
2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Glitch checking methodology using static-signoff

accellera
SYSTEMS INITIATIVE

Glitches Fail Chips

- Glitch = transition shorter than signal's clock period
- Async Interactions within designs become vulnerable to glitches at netlist
- Not caught by STA & functional simulation



Structured Glitch Verification Methodology

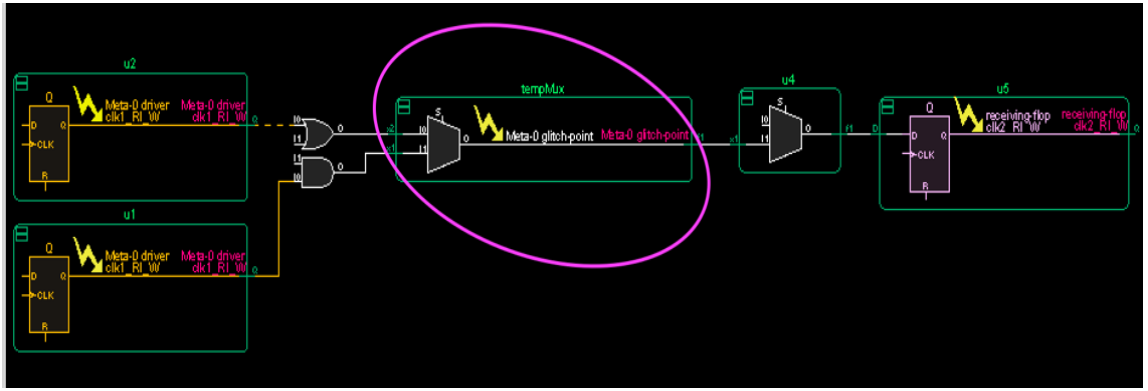
- CDC glitch failures encapsulated within interfaces
  - Using pre-verified components for CDC interfaces is standard practice
  - Structural glitch avoidance principles can be included within components

Structured Glitch Verification Methodology

- CDC glitch failures encapsulated within interfaces
  - Using pre-verified components for CDC interfaces is standard practice
  - Structural glitch avoidance principles can be included within components
- For timing exceptions & power management logic, path structuring may be necessary with synthesis tool restrictions for path during optimization.
  - Insert “do not touch” component to partition paths
  - Verify connectivity in netlist
  - Verify glitch in netlist

Glitch sign-off – IP level, Chip level

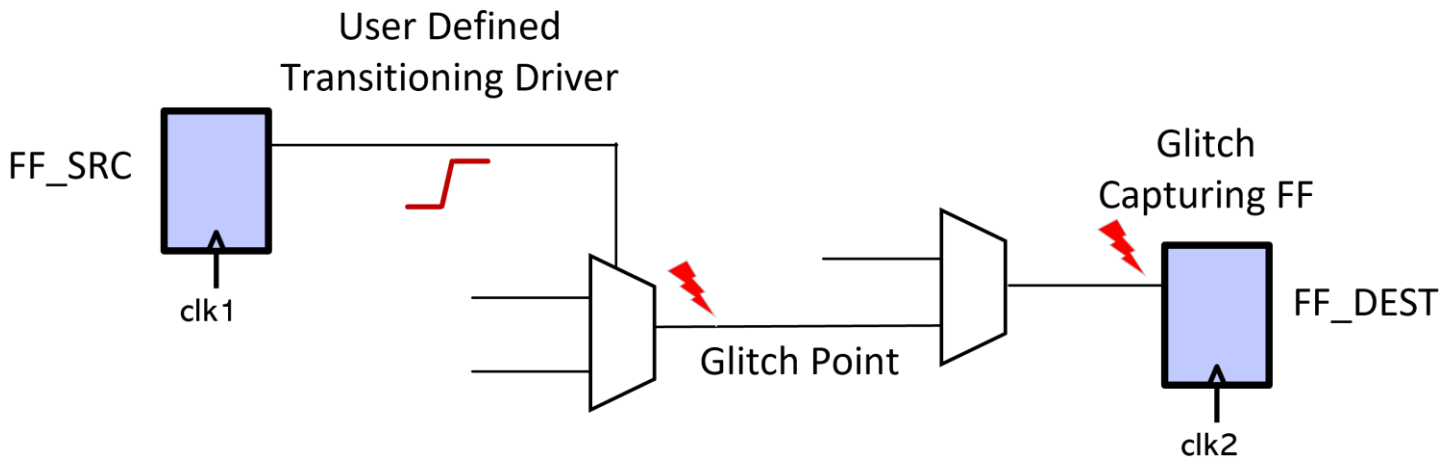
- In Async paths, Glitch can be fatal
  - False Paths, Multi-Cycle Paths, Clamps, Global Signals, Power control signals
  - Logic reordering, restructuring, retiming, optimization
- Numerous companies had *late-stage* netlist-glitch failures
  - IP vendor provided glitchy-IP (@outputs) to customer
  - 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

Example Glitch Report

- Source to destination path should not be glitchy
- Check can be easily specified



- Glitch source flop/signal, glitch point and glitch capturing flops reported

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

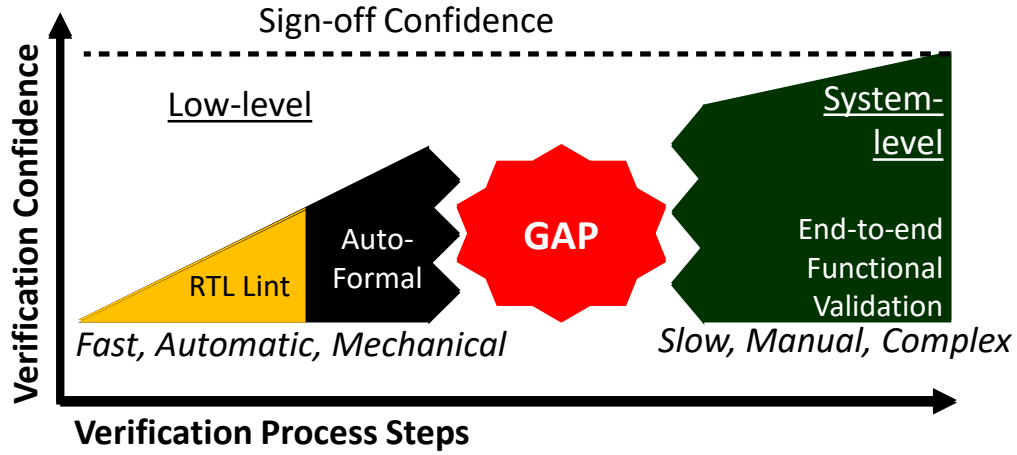
SAN JOSE, CA, USA
MARCH 4-7, 2024

Efficient functional sign off by automatic assertion generation for RTL building blocks using static methods

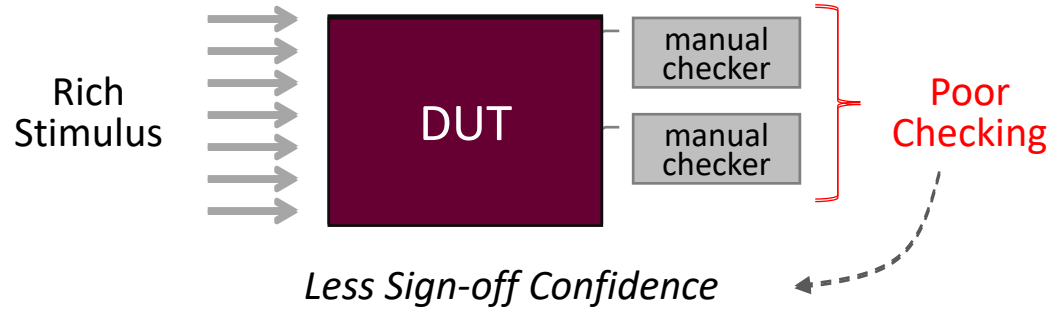
accellera SYSTEMS INITIATIVE

The Verification Challenge

- System-level validation is complex, slow, and incomplete, pushing up HW design cost
- Systematic functional sign-off is an underserved imperative
- Vast gap between low-level RTL checks and system-level functional RTL sign-off
- *Must Bridge the Gap!*

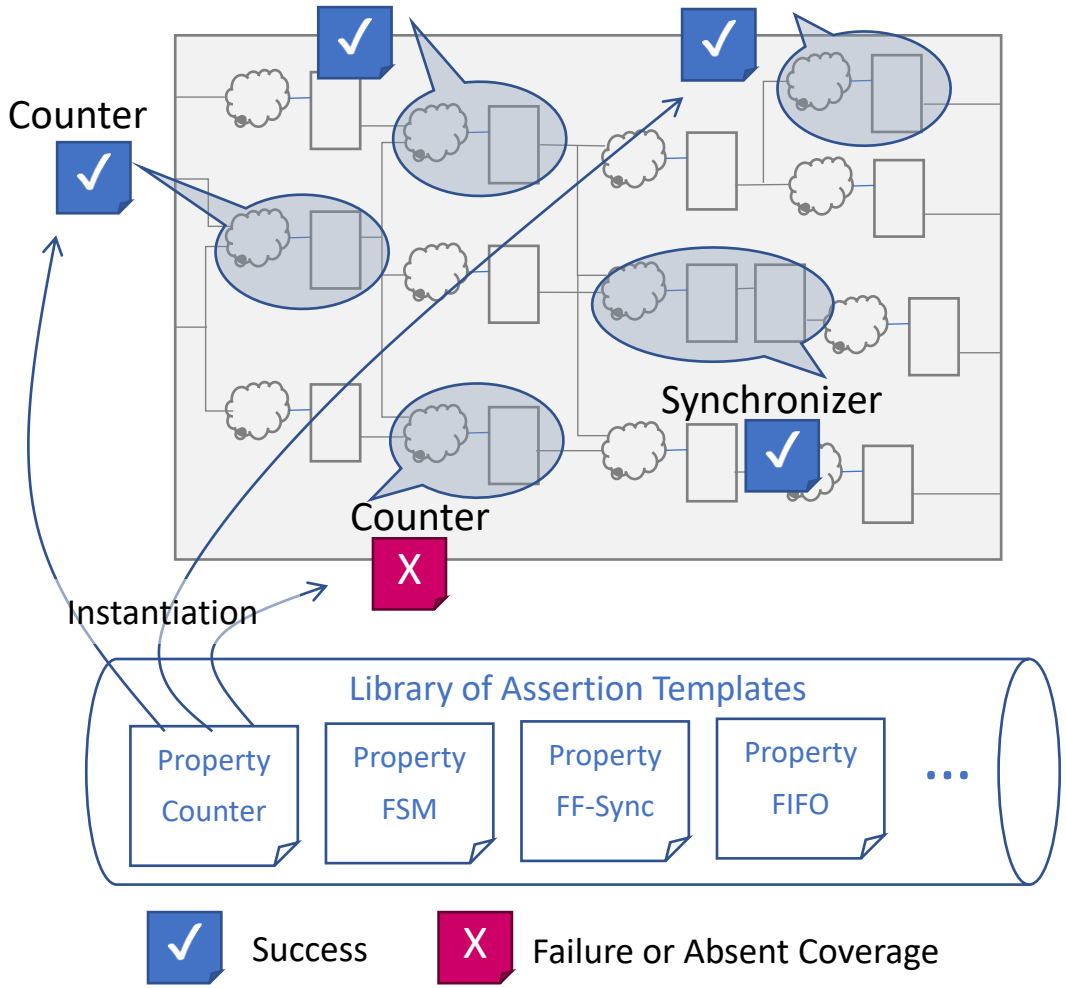


- System-level validation is very hard due to
  - 3rd-party IP, Distributed design team, Legacy RTL in SOC assembly
- Stimulus automation has been the focus so far
  - Constraint random, Formal, PSS, UVM..
- *But, Manually-Guided Checkers are Slow, Unstable, and Insufficient*
  - Researching, planning, coding, reviewing, debugging..
- Need automation in checker generation also!



Source: www.realintent.com

Auto-Inferred Building-Block Property Checking (AIPC)



- Most designs have primitive building-blocks
  - Counter, FSM, FIFO, Stack, FF-Sync, RAM, Shift-Reg etc.
- *Advanced Functional Static Analysis* successfully *automatically* infers such building-blocks in RTL
- Generate white-box assertions based on *Simple Assertion Template* for each building-block type
- Bind these assertions to RTL using co-generated bind files without user effort
- AIPC method allows uniform safety and coverage criteria to be created across a variety of implementations

Source: www.realintent.com

AIPC Assertion Library

COUNTER

- Initialization
- Load data
- Hold data
- Count
- Loaded value

FSM

- State initialization
- State transition

FIFO

- Initialization
- Overflow
- Underflow
- Full
- Empty
- Empty pointer
- Full pointer
- Data integrity

STACK

- Initialization
- Push
- Pop
- Empty
- Full
- Data Integrity
- Empty pop
- Full push

RAM DP

- Initialization
- Data integrity
- Unknown value

RAM SP

- Initialization
- Data integrity
- Unknown value

SISO SHIFTREG

- Initialization
- Shift check

PIPO SHIFTREG

- Initialization
- Shift check

FF SYNC

- Initialization
- Data stability
- Synchronization

GRAYCODE SYNC

- Initialization
- Data stability

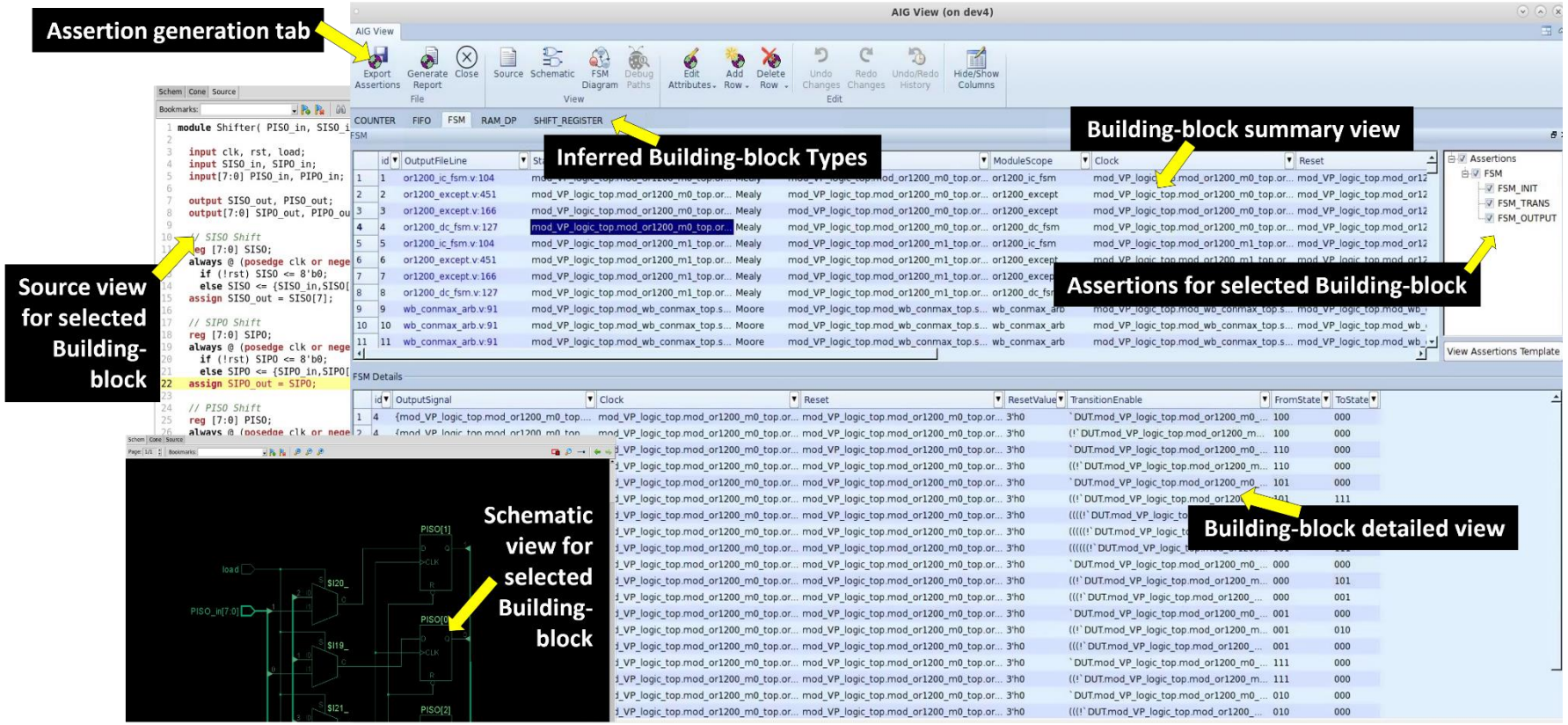
HANDSHAKE SYNC

- Data stability
- Complete cycle
- No ack without request
- Req not asserted until ack deasserted

MUX SYNC

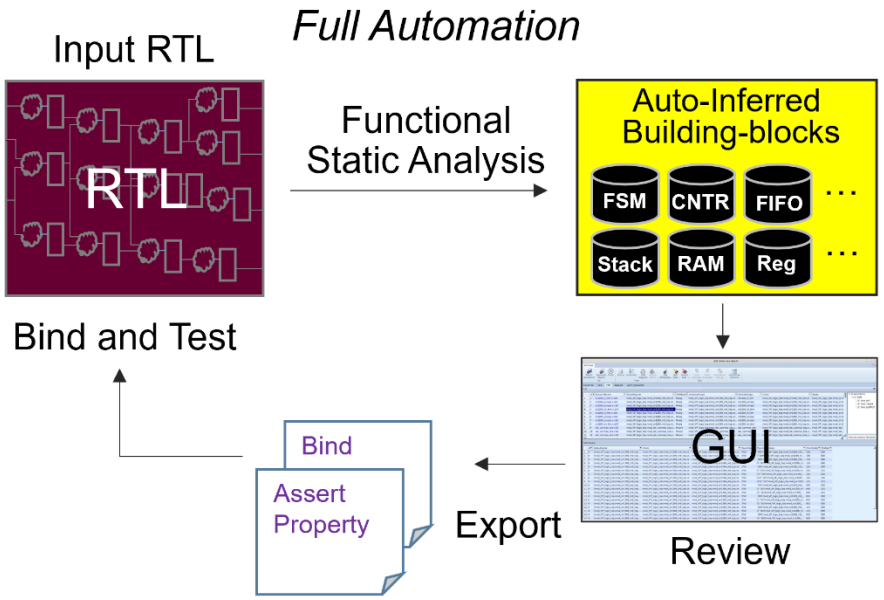
- Control signal stability
- Data stability

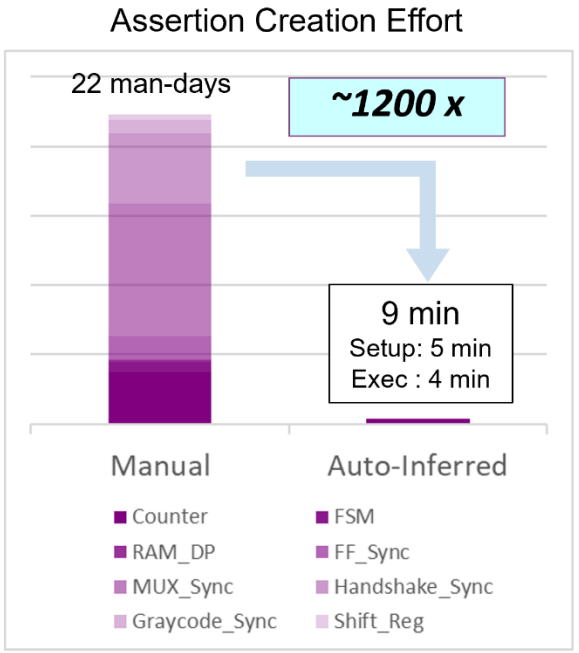
GUI Snapshots



Source: www.realintent.com

Full and Instant Automation





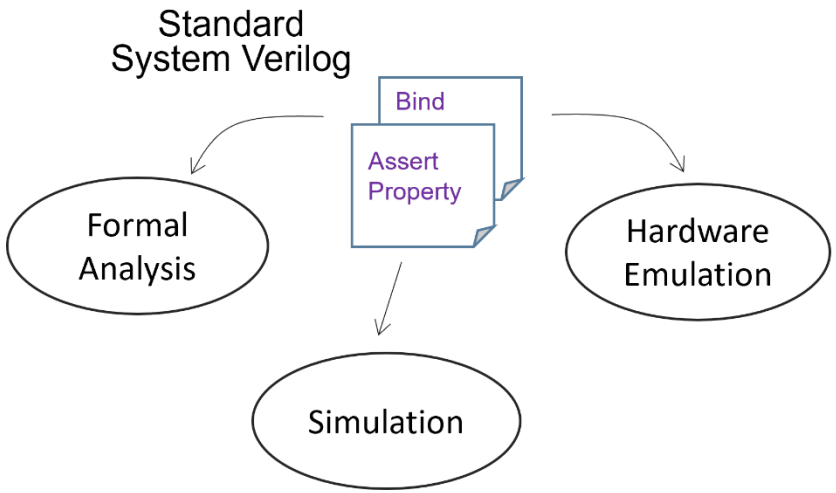
Communication Design IP Core
Design size : 800k Gates

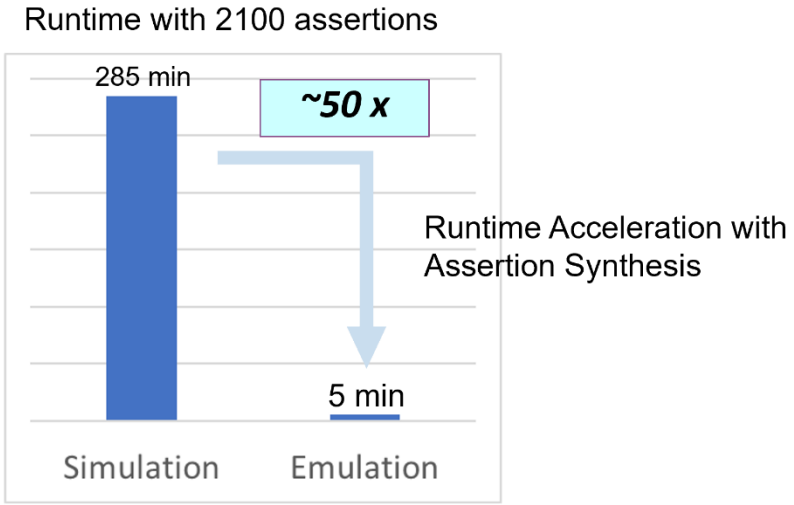
Number of Building-blocks	
Counter	299
FSM	59
RAM_DP	12
FF_Sync	200
MUX_Sync	1152
Handshake_Sync	608
Graycode_Sync	111
Shift_Reg	35

- Tool can automatically generate ready-to-use assertion files *Without Any User Hints*
- The generated assertions are applicable to every design with uniform quality

- *Extremely Fast Generation*
- Vast enhancement in productivity

Multi-Purpose Use for RTL Verification

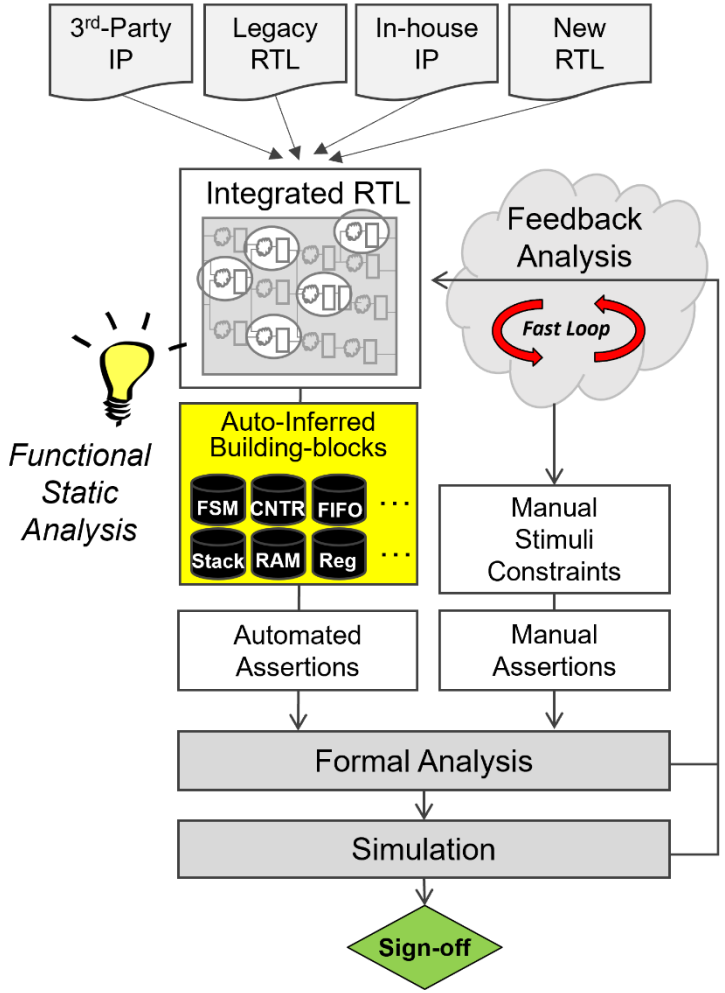




- Thousands of assertions cause a performance overhead in simulation
- In HW Emulation, *Assertion Synthesis* can accelerate runtime dramatically

Source: www.realintent.com

Verification Flow with AIPC



- IP-level to system-level validation becomes more efficient with AIPC method
- AIPC provides seeds that introduce *the initial review points* in a black-box RTL
- Analyzing *absent coverage and inconsistency* helps understand module-level behavior
- Leads to creation of manual assertions, stimuli, and constraints from *a bottom-up view* in addition to the existing top-down approach
- AIPC enabled methods are notably beneficial *for 3rd-party engineers or for engineers dealing with 3rd-party IP or Legacy RTL*

Source: www.realintent.com

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Advanced methodology to identify X-initialization source errors and fix them to prevent the error from propagating

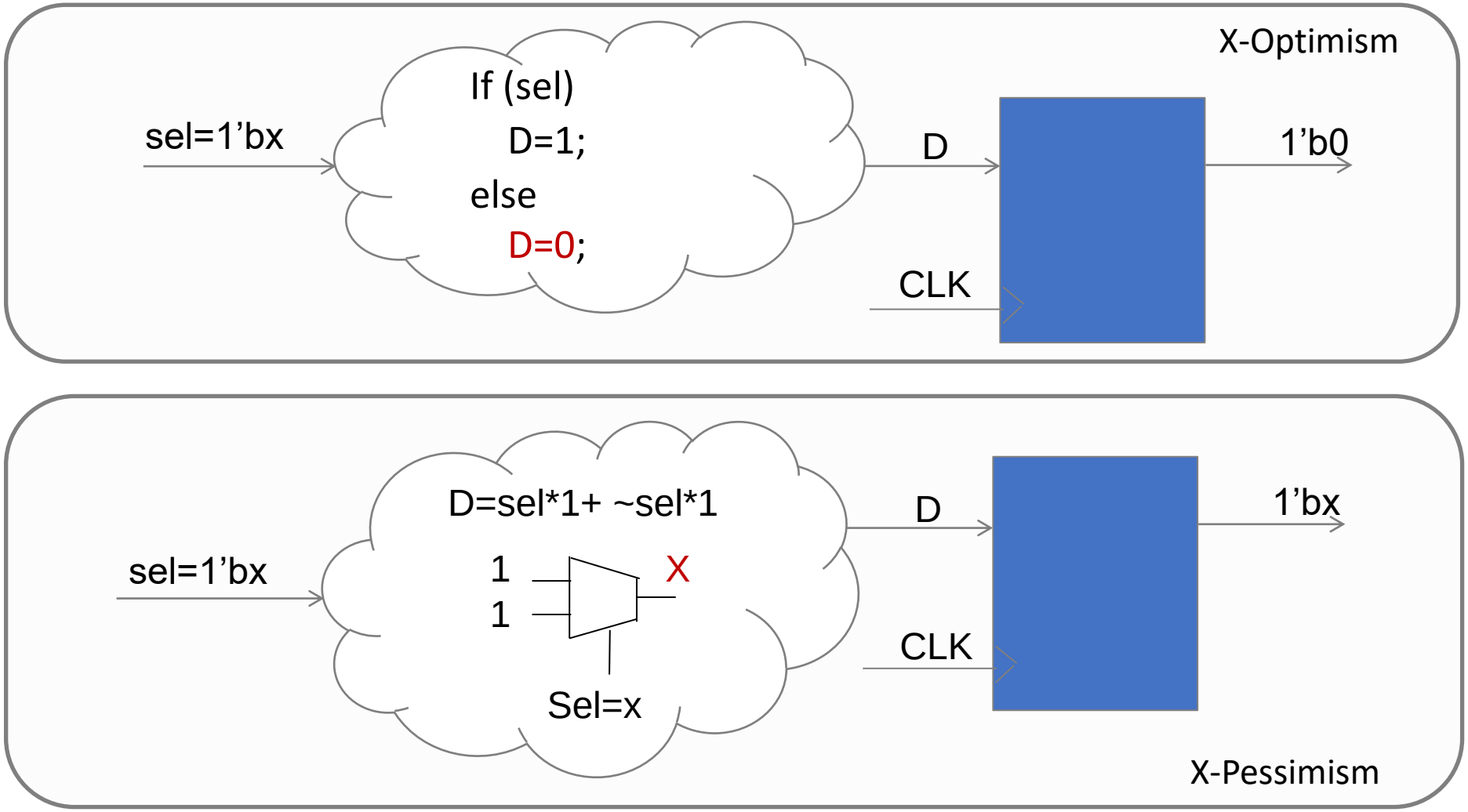
accellera
SYSTEMS INITIATIVE

What Are X Sources?

X Source : a flop or input port which is in unknown value at the end of a given reset scenario

Potential X Sources	
Uninitialized 4-state variables (Uninit)	Out-of-range bit-selects and array indices (OutOfRange)
Low power logic shutdown or power-up (NonRetention)	Logic gates with unknown output values (Explicit)
Unconnected module input ports (Undriven)	Setup or hold timing violations (Netlist only)
Multi-driver conflicts (Bus Contention)	User-assigned X values in hardware models (Explicit)
Operations with an unknown result (RAMs, FIFOs)	Testbench X injection (User)

X Propagation: Two Problematic Scenarios



The Impact of X Propagation in the Design

Uncontrolled Design Behavior

- Uninitialized and Reverted to X Flops
- Incorrect reset type and value
- Hardware security exposure

Sub-optimal Design Quality

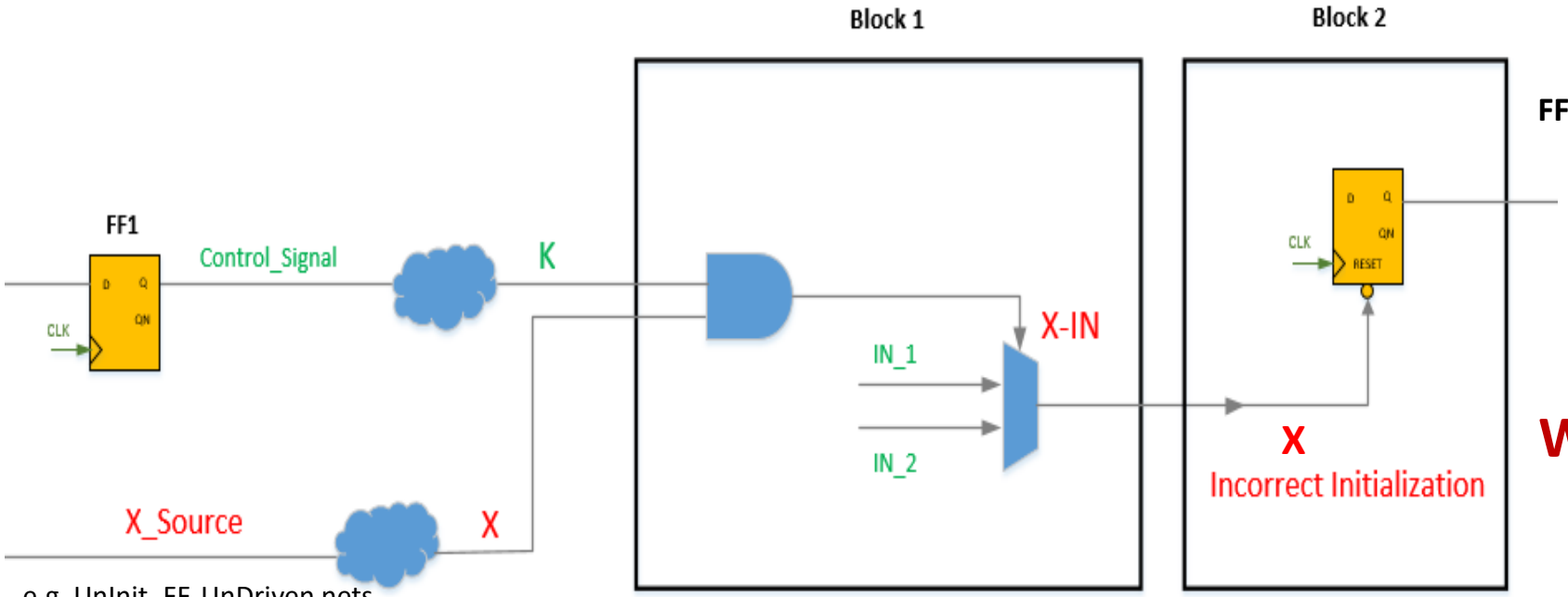
- Long initialization latency
- Inefficient reset routing
- Excessive initialization power

Breakdown of Validation Flow

- Incorrect simulation
- Incomplete simulation
- Inefficient gate-level debug

Ignoring the impact of design X values causes silicon failures.

Incorrect Initialization Causes Design Failure



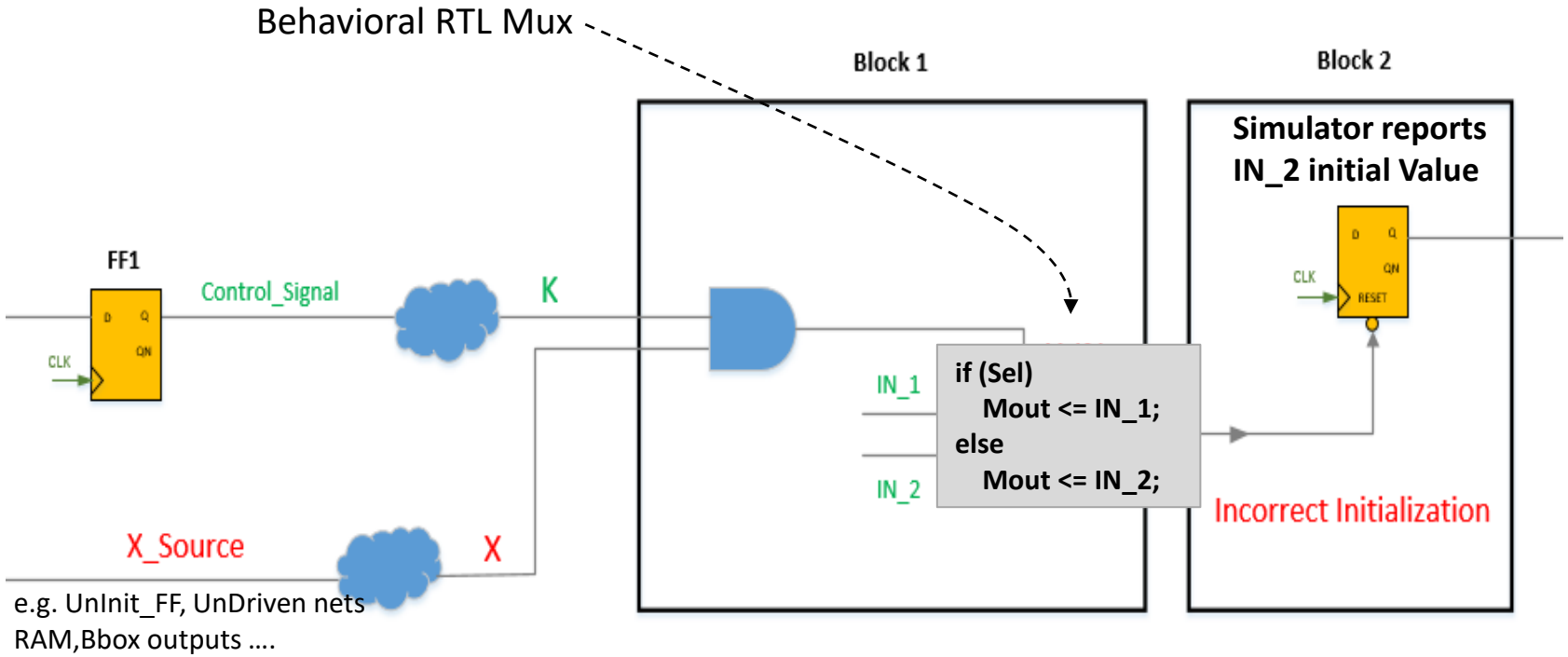
e.g. UnInit_FF, UnDriven nets
RAM,Bbox outputs ….

What you'd want to know:

- X-Source
- X-Reset

- Functional non-determinism in a design is a failure
- Incorrect design initialization causes erroneous design behavior, sub-par design quality, security vulnerability

Simulation Can Mask X Issues Due to X-Optimism

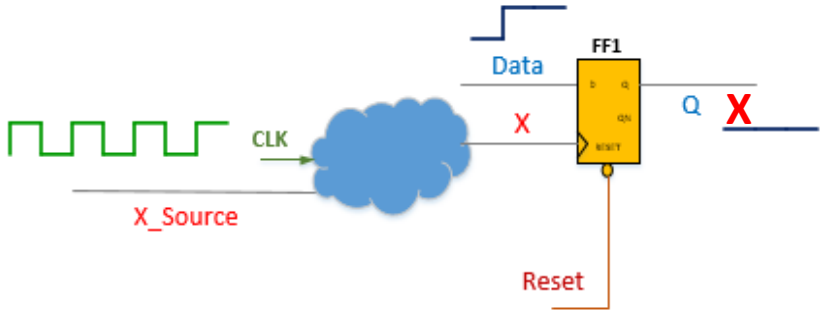


What you'd want to know:

- X-Source
- X-Reset
- X-Optimism

- Can not rely on simulation for correct initialization because simulator can mask X initialization, and propagates wrong value (X-Optimism)
- Must fix X-Sources before RTL simulation

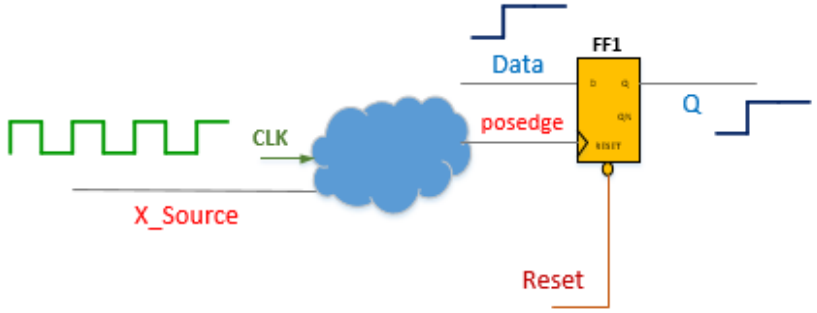
Simulation Can Miss Dangerous X on Clock



Initialization Problem

With Reset de-asserted, Clock 0->x should be treated as **an undetermined edge**

=> FF1.Q should become **X**



Missed by Simulation

However, in Simulation, when Reset is de-asserted, Clock 0->x is treated as a **posedge** (X-Optimism)

=> FF1.Q changes to 1

Nominal Approaches to Deal with X Issues Are Imprecise or Sub-Optimal

- **Change X-Optimistic Simulation Behavior to X-Pessimistic**
  - Causes extra X-failures in RTL simulation
  - Increased debug effort

- **Reset all flops**
  - Routing congestion
  - Sub-optimal design to overcome simulation limitation

Static Sign-off Can Complete / Enhance Simulation-Only Flow

Simulation	Static Sign-off
Test bench coverage dependent	Comprehensive
Simulation needs mature RTL	Early
Simulation needs vectors	Low effort
Band-Aid for symptom, X remains in GLS	Addresses root cause
No optimization	Optimization potential

Best Practices to Ensure Design Has No X Issues

- Put the device in a known functional state
  - Eliminate dangerous X-Sources that can cause functional problems by initializing them
  - NEED TO KNOW: where x-sources and x-resets are
- Minimize silicon resources dedicated to reset
  - Optimize reset network, initialize the minimum required flops
  - NEED TO KNOW: what resets to add or remove
- Avoid simulation inaccuracy due to X-Propagation
  - Fix X-Optimism as much as possible before RTL simulation
  - Remove X Sources as much as possible
  - NEED TO KNOW: where X-Optimism happens

Questions?

Source: istockphoto