2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Mind the Gap(s): Creating & Closing Gaps Between Design and Verification

Chris Giles, Product Manager, Mentor, A Siemens Business

Kurt Takara, Product Engineer, Mentor, A Siemens Business

Mentor®
A Siemens Business

accellera SYSTEMS INITIATIVE

Things we will and will not be discussing…

IS AND IS NOTS…

This Workshop Will Not…

Remind you of increasing complexities…

…or shorter schedules…

…or fewer resources…

….or closing coverage faster*

…or finding bugs faster*

*at least as the ends, and not the means

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

This Workshop Will…

Help you prove faster that your design works <u>as intended</u>
via mindful attention to gaps in development

accellera SYSTEMS INITIATIVE

Gaps Are Neither Good nor Bad

You succeed or fail based on how you address gaps!

1. Recognize where they occur
2. Align your process to minimize the loss across the gaps
3. Cover the remaining gaps with precision

This workshop will highlight typical gaps, then show examples of:

- ***Creating*** intentional gaps in development to align work, organizations, skills
- ***Closing*** gaps precisely
  - That are organizational, geographical, or time-based
  - Between design and implementation

Every point of hand-off is a potential gap in the development process

WHERE ARE THE GAPS?

Certain Types of Gaps Cause Program Issues

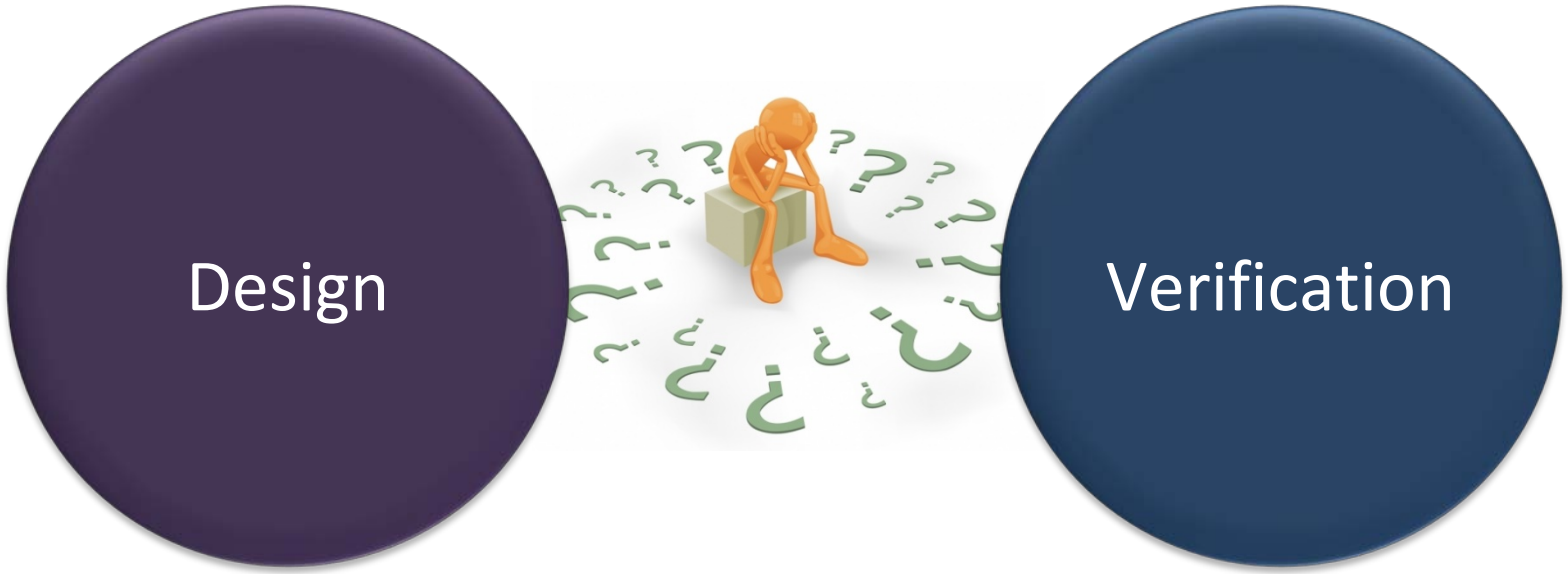
Documentation: incomplete, subject to translation, assumptions, interpretation

Models: abstractions, inaccurate

Change: impacts requirements, constraints, markets, technologies, standards and teams

Organization: (and discipline) boundaries create opportunities for information loss

Knowledge: need may not align with ability



Asking an OO-Coder to Understand Metastability or a Designer to Run Formal Methods

CREATING AND EMBRACING GAPS: INTENT VERIFICATION VS FUNCTIONAL VERIFICATION

Discipline Drivers at the Dawn of RTL

Design and DV were essentially the same skillset

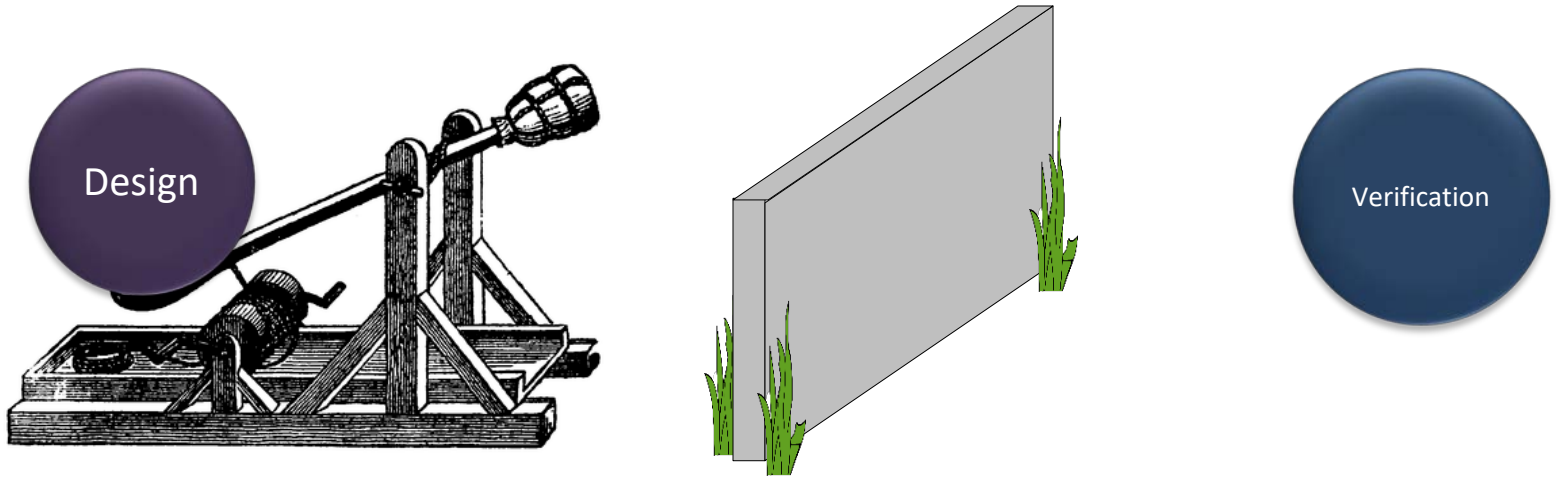
- RTL, testbenches and tests were coded with same language
  - Directed, self-checking testing
  - Some smart checking of outputs
  - Manual maintenance of test coverage
- Engineered serially
  - Designed first, verified second

Discipline Drivers Today

What drives verification today is different than design

The Designer Knows the Intent: S1

But who is doing the verification?



- Scenario 1: Separate design and verification teams
  - Information is lost between design and verification
    - Subject to documentation, organization and knowledge gaps
  - Verification teams drive the tools but may not understand the goal
    - Should a verification engineer know about metastability?

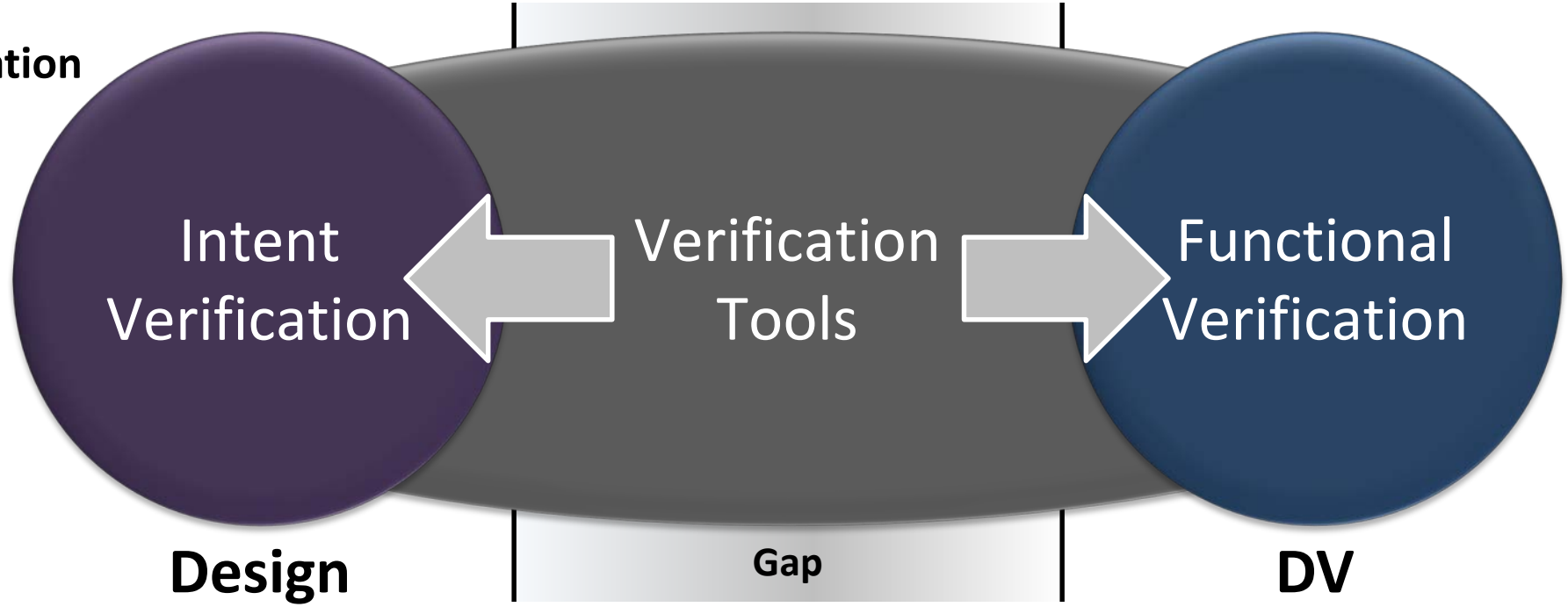
S2: The Designer Knows the Intent

But who is doing the verification?

- Scenario 2: One team of engineers
  - Engineers both design and verify
  - Subject to skillset gaps
    - How many teams can afford engineers that are great at modern design ***and*** verification?

Why Not Let the Designer Verify?

Wrap verification tools in a designer-friendly wrapper



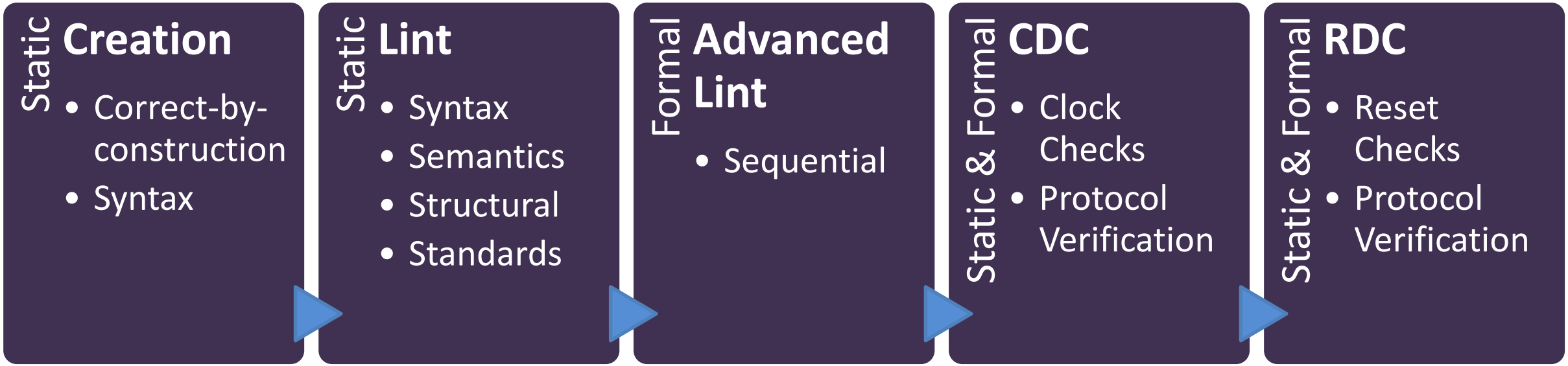
Intent Verification:
Verification of *design* intent, by the engineer who knows the intent and correct implementation best

Create an intentional gap to embrace and align with your organizational & development gaps

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Designer-Driven Intent Verification

Either way, designers need different tools



HDL Designer Series | Questa® AutoCheck | Questa CDC | Questa RDC

Designers don’t want and shouldn’t have to use a testbench to verify intent

accellera SYSTEMS INITIATIVE

Why Isn't Functional DV Enough?

- For functional DV to catch everything in "Intent Verification", DV must:
  - Have full functional coverage, code coverage, testplan coverage, etc.
  - Sweep all asynchronous clocks and resets through all combinations
  - Model and detect failures correctly and completely
- Even if it does all that, it will take time to do so
  - Intent Verification solutions are static and exhaustive
- Even if it does all that, it could be debugged by another team
  - Even worse – it could be "fixed" by another team

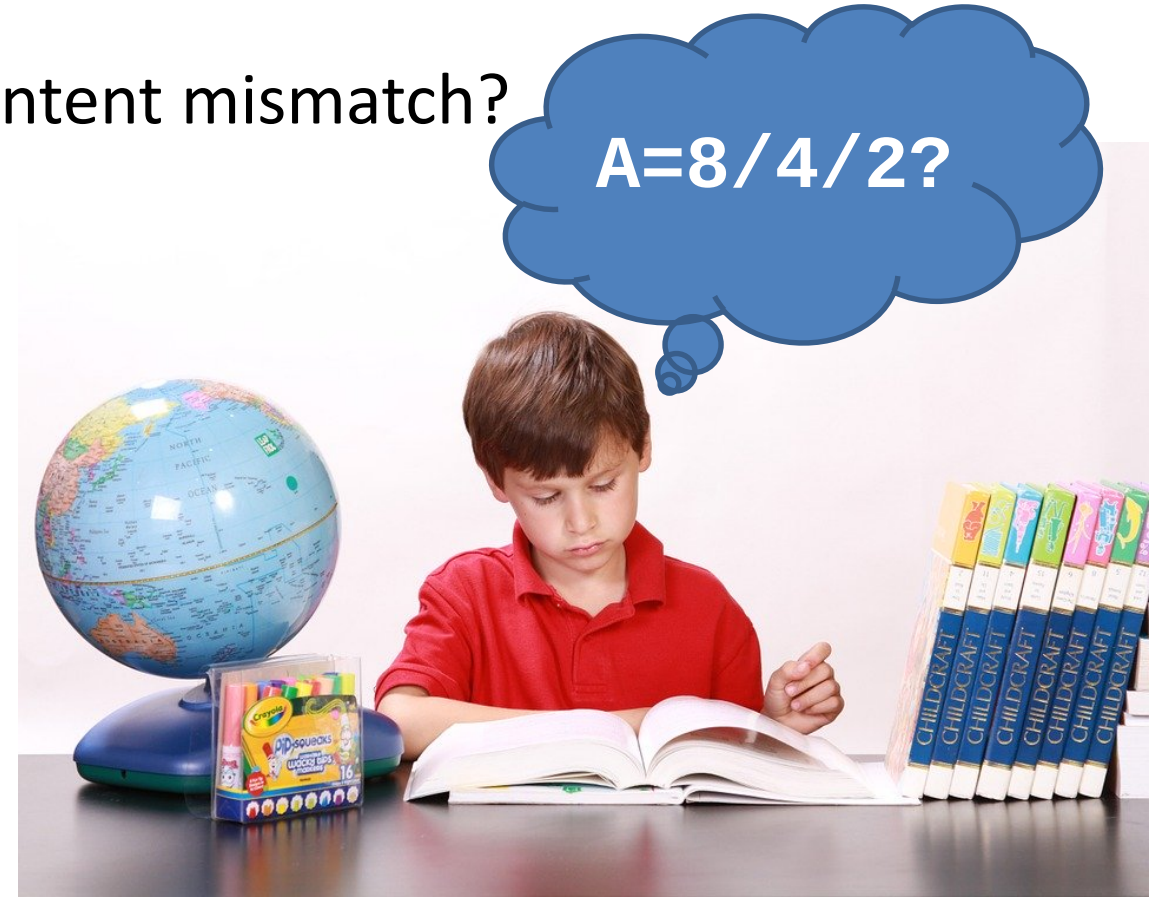
Intent Verification Example: Lint

- Why wait for simulation to find an intent mismatch?
- The designer wants A=4
  - needs a check for missing parens

```
wire [3:0] a, b, c, d;

assign b = 4'h2;
assign c = 4'h1;

assign a = 4'h8 >> b >> c;
```



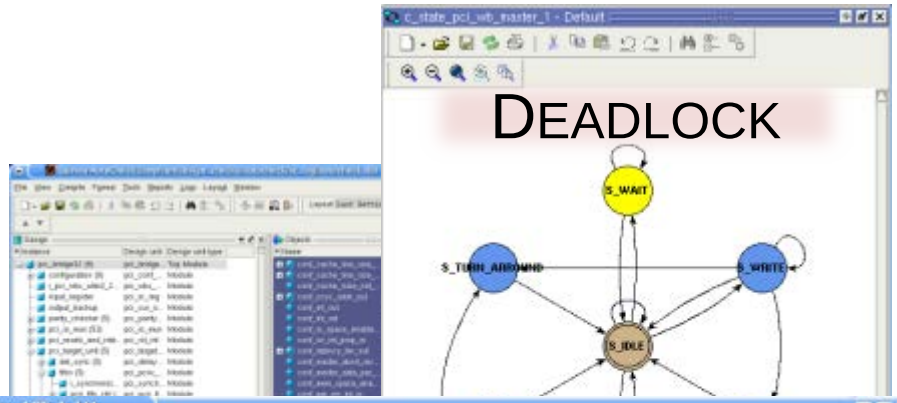
Linting is Essential

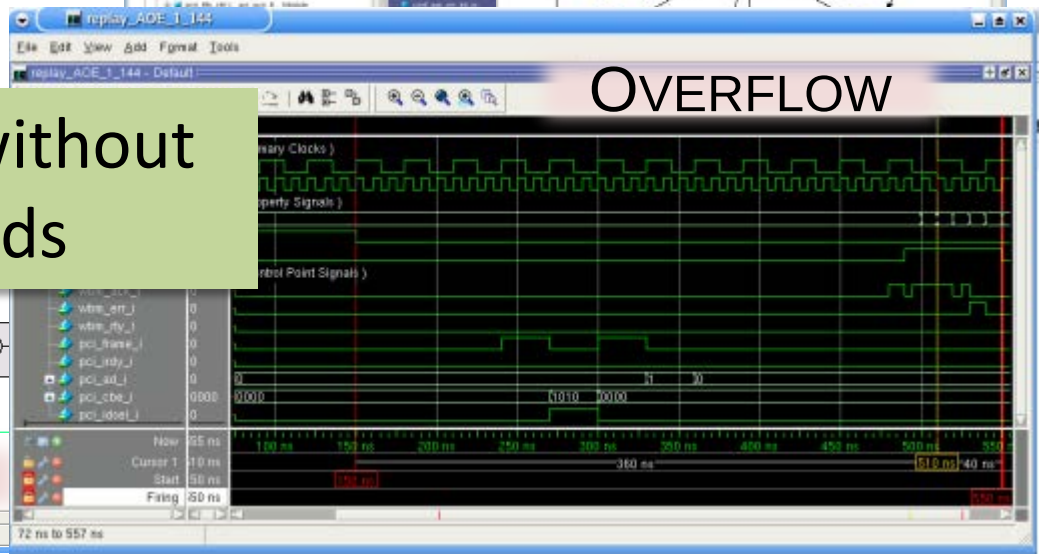
We have a linting solution available

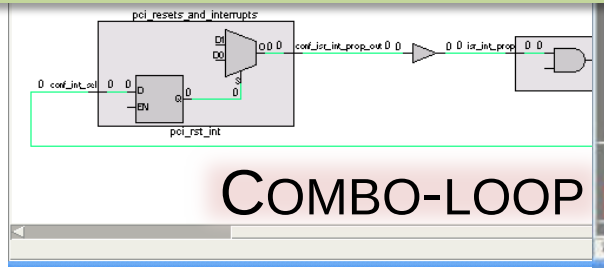
- Currently engaging customers in early access mode
- Implements important differentiators
- We are interested in working with additional customers

Intent Verification Example: Advanced Linting

- Why wait for simulation failures and debug (at best) to find:
  - A deadlock scenario in your state machine
  - An overflow condition on a registered variable
  - A combinational loop in your code, etc.



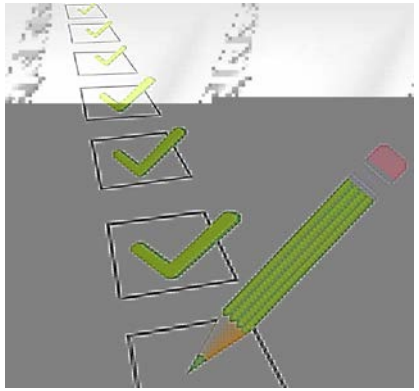




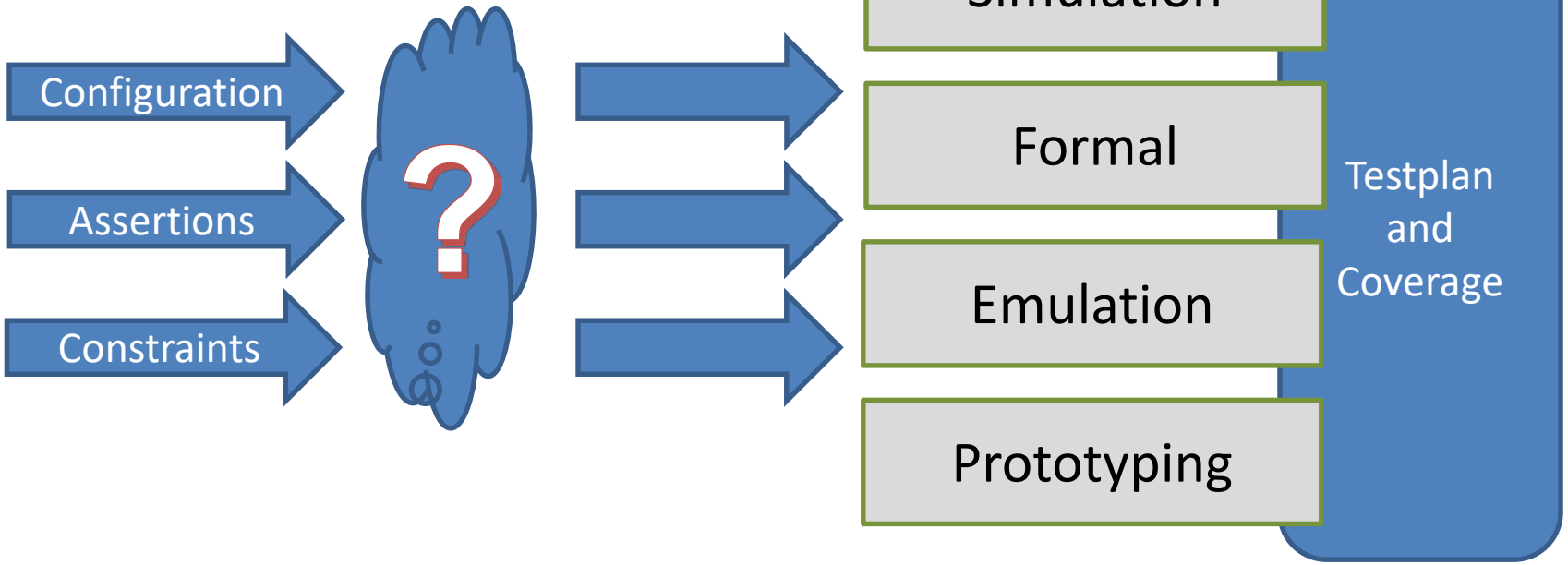
The designer can find and fix these things without a testbench, or knowledge of formal methods

Bridging the Gap to DV and Beyond

- Collateral from intent verification must be used in DV (and beyond)
  - Constraints/waivers should be valid and consistent through all Intent Verification
  - e.g SDC for CDC & RDC, then on to synthesis, STA, etc.



Successful Intent Verification



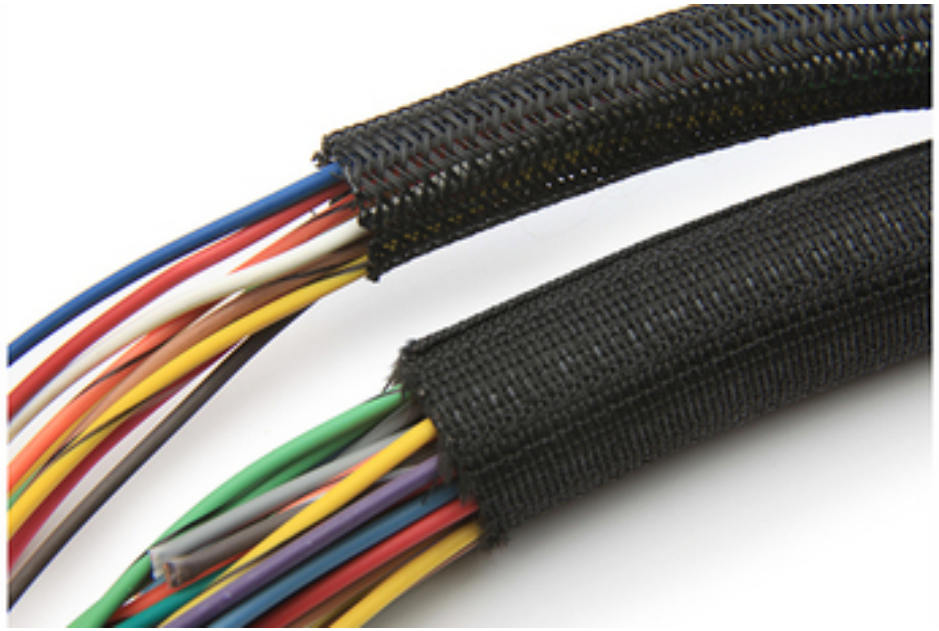
Examples of Necessary Bridges to DV

When implementing an Intent Verification flow, consider the following:

- Some things are better left to simulation-based verification
  - Deep reconvergence verification for clock and reset domain checks
  - Some clock and reset domain crossing protocol verification
- Waiver cross-checking in simulation with functional coverage
  - Was the designer overly aggressive in waiving violations?
- ***But***, verification environment must support variable latency and observation

Summary

- Identify gaps between design and verification
- Create an intentional gap by implementing Intent Verification flows
  - Embrace design/DV gap by aligning verification with organization/knowledge gaps
- Bridge the remaining gaps with cross-checks, DV and coverage



SoCs Integrate IP from outside companies, across geographies, language barriers, previous projects

CLOSING ORGANIZATIONAL, GEOGRAPHICAL, TIME-BASED GAPS

Recall the Story of a Very Large Aircraft…

A large aircraft launch was delayed multiple years

- Wire cables could not be connected in the prototype
  - Despite reviews, modeling and mock-ups
- Root-cause analysis determined the causes to be:
  - Different versions of software used by different teams in different countries
  - With different backgrounds, languages, goals, even measurement units
- ***How different is that from your organization today?***
- ***How can tools and methodologies bridge these gaps?***

(Hint – minimize the handoffs)

Intent Verification, the SoC and Gaps

Flat Analysis

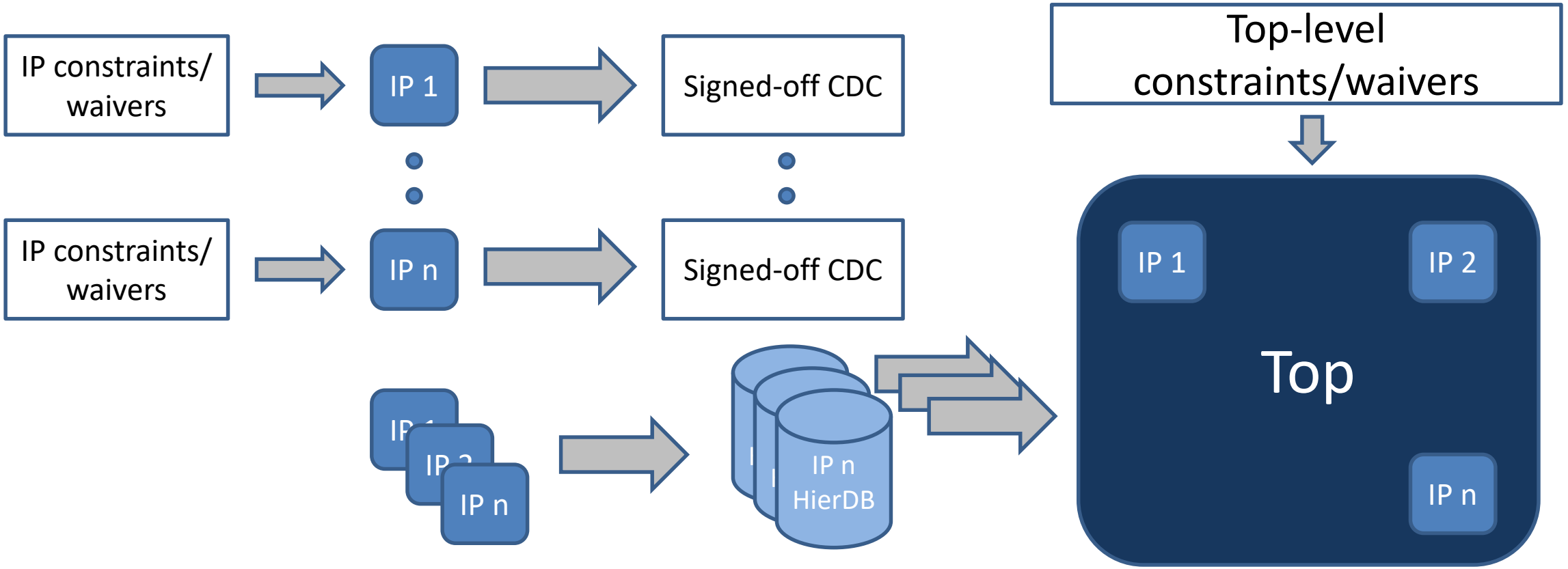
- Scalability (schedule) issues
- 3rd Party vendors don't want to discuss low-level design details
- You should not need to re-verify IP that you've purchased or has already seen silicon (?)

Hierarchical Analysis

- Scalable (faster approach)
- Aligned with organizational gaps (IP vs. SoC, corporate, time)
- Requires abstraction
  - Black-box implies trust – nothing is re-evaluated beyond the lowest level
  - White-box will yield better integration analysis

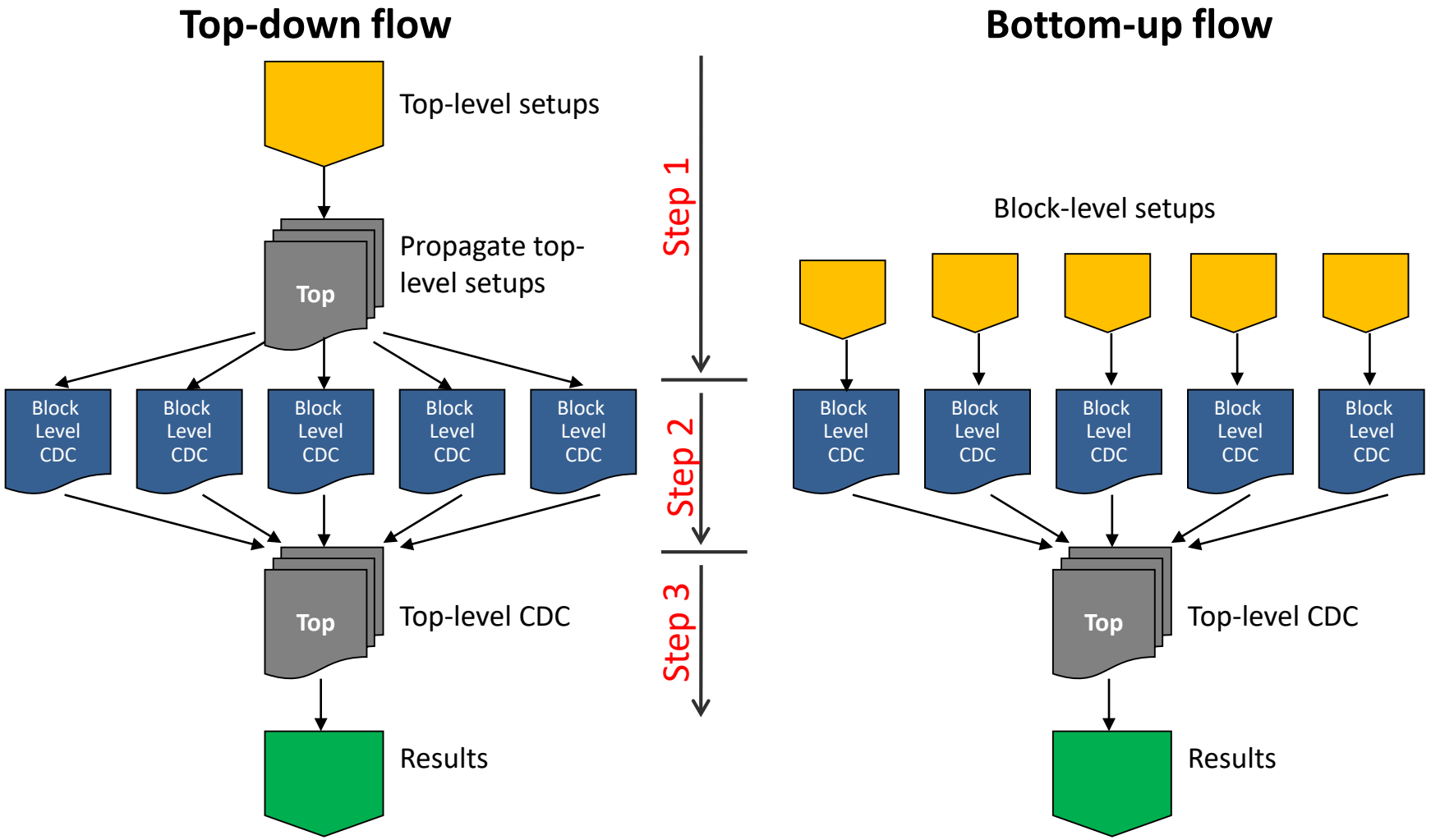
Abstraction methods require EDA vendor support
Accuracy and performance both matter

Example: A Distributed Hierarchical CDC or RDC Approach



Enables a faster turnaround in case of RTL changes or ECOs

Hierarchical CDC or RDC Analysis Flow



Is My Hierarchical Flow Bullet Proof?

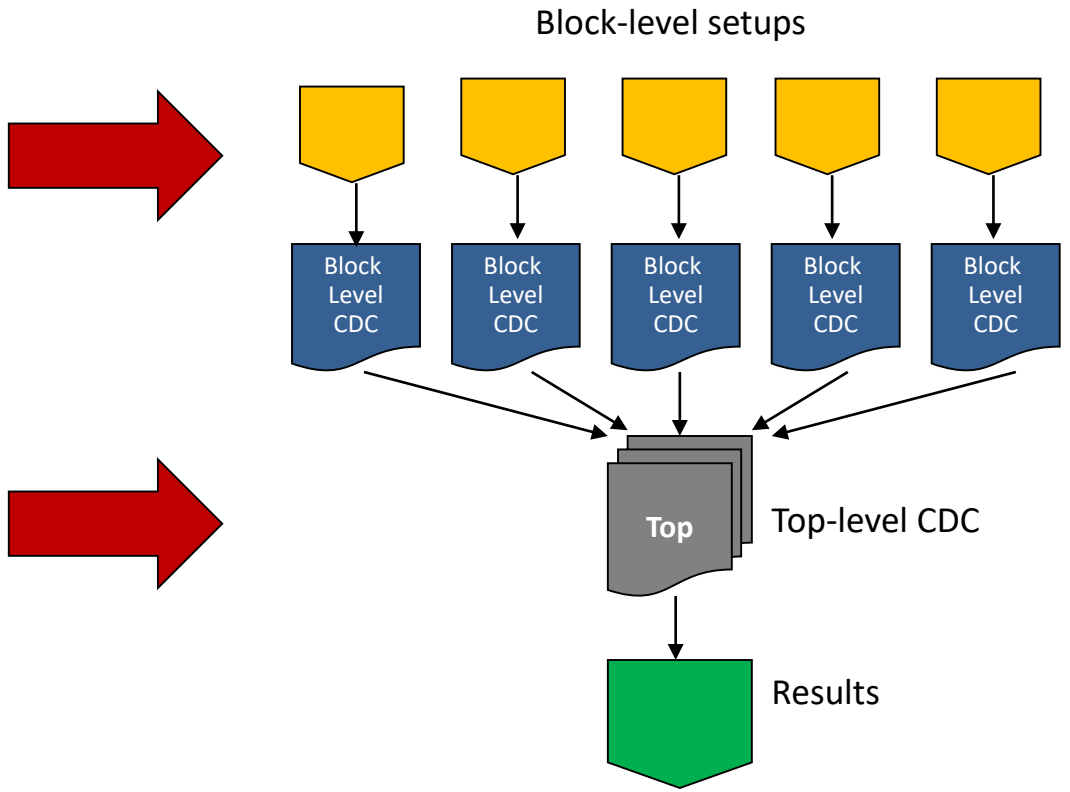
Conflicts will invalidate results!

- Critical for bottom-up flow
- Required for top-down flow with modified block constraints

1. Review block conflicts

2. Review top conflicts

Bottom-up flow



Block-level Conflicts

- User vs Inferred port conflicts categorized in 3 sections

1. Single-clock mismatch

```
User : netlist port domain in –clock clk1 –module b1
Inferred : hier port domain in –clock clk2 –module b1
```

2. Multiple-clock mismatch

```
User : netlist port domain in –clock clk1 –module b1
Inferred : hier port domain in –multiple_clocks –module b1
```

3. Sync mismatch

```
User : netlist port domain in –async –module b1
Inferred : hier port domain in –clock clk1 –module b1
```

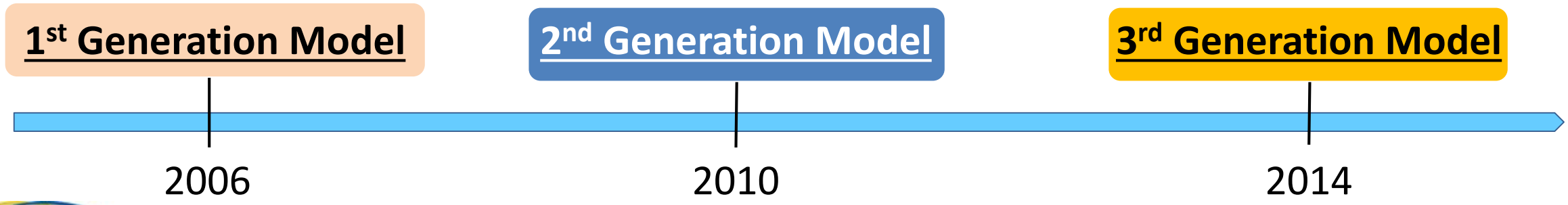
Top-level Conflicts

- Check mismatch between Block and Top constraints
  - Constants & stable ports consistent
  - Clocks hooked up consistently
  - Ports hooked up consistently
- Warnings flagged for conflicts

```
*************************************************************************
*   Section 2 : Conflicting Constraints
*************************************************************************
Port                Conflicting constraints
-------------------------------------------------------------------------------
d1                   User-specified : netlist  port domain d1 –clock clk1
                      Inferred : hier port domain d1 –clock clk2
d2                  User -specified:  hier port domain d2  –async
                      Inferred : netlist  port domain d2[1] –async –clock clk1
                                          hier port domain d2[0] –clock clk2
```

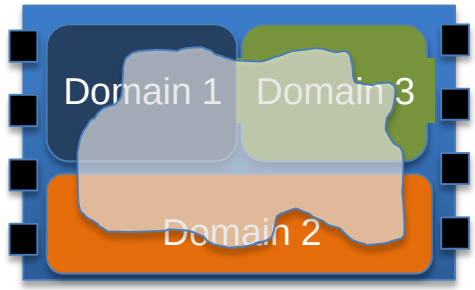
Questa CDC Hierarchical Model Evolution

- 3rd Generation Hierarchical Data Model (HDM)
- Binary model
  - Database stores block information
  - Extendable to provide additional accuracy
  - Partially directives-based for overriding
- Supports both white-box & black-box abstractions



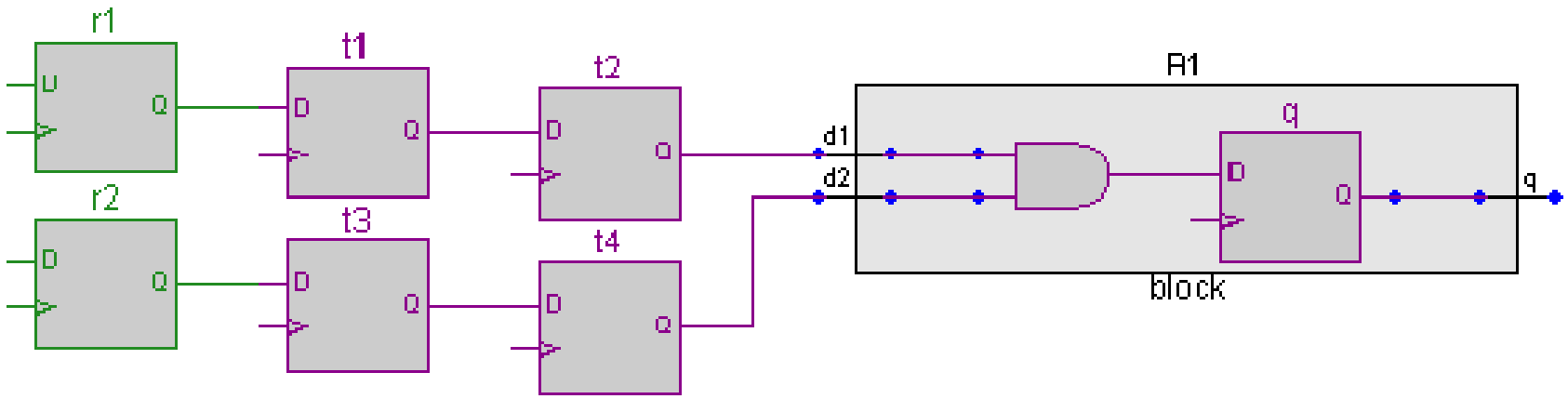
Questa HDM White-box Abstraction

- Crossings reported to/from internal HDM sequential elements
  - instead of HDM ports
- Improved multiple-fanout port visibility
  - Black-box: single crossing to/from the port reported for HDM
  - White-box: individual crossings for fanout registers of port's load/driver
- Schematic visible for complete path including HDM internal logic
- Waivers work as expected
  - -through <hdm_port>
  - -to <hdm internal reg/latch>
- Enables sequential reconvergence verification



HDM Reconvergence : Example 1

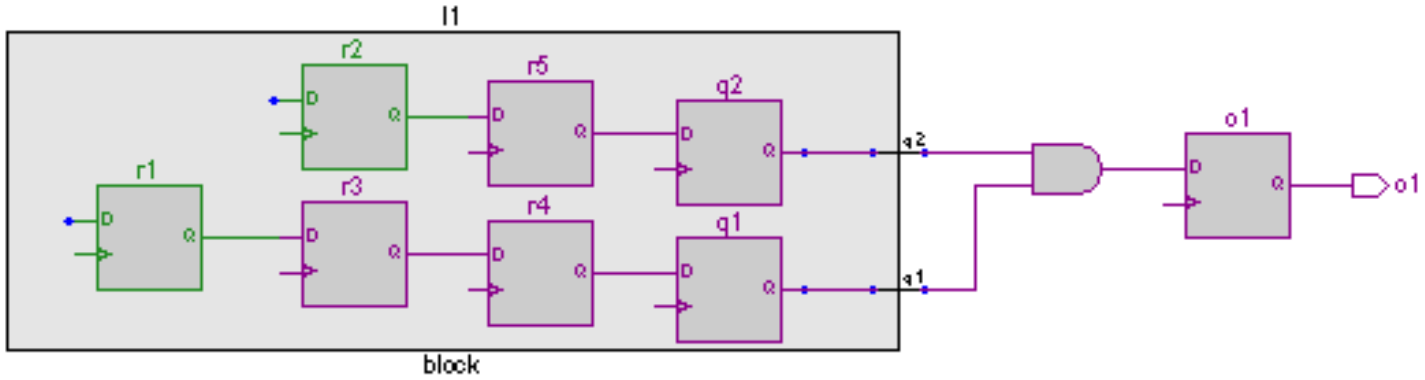
- Syncs outside the HDM block converging inside the block




```
Reconvergence of synchronizers. (reconvergence)
-----------------------------------------------------------------
clk2 : end : B1.q (/home/test01/dut.v : 1) (ID:reconvergence_79534)
  clk2 : start : t2 (/home/test01/dut.v : 9) (Synchronizer ID:two_dff_19496) (Depth:0) (Reconvergence Severity:Caution)
  clk2 : start : t4 (/home/test01/dut.v : 9) (Synchronizer ID:two_dff_67560) (Depth:0) (Reconvergence Severity:Caution)
```

HDM Reconvergence : Example 2

- Syncs inside the HDM block converging outside the block




```
Reconvergence of synchronizers. (reconvergence)
------------------------------------------------------------
clk2 : end : o1 (/home/test03/dut.v : 1) (ID:reconvergence_68716)
  clk2 : start : I1.q2 (/home/test03/dut.v : 13) (Depth:0) (Reconvergence Severity:Caution)
  clk2 : start : I1.r4 (/home/test03/dut.v : 15) (Depth:1) (Reconvergence Severity:Caution)
```

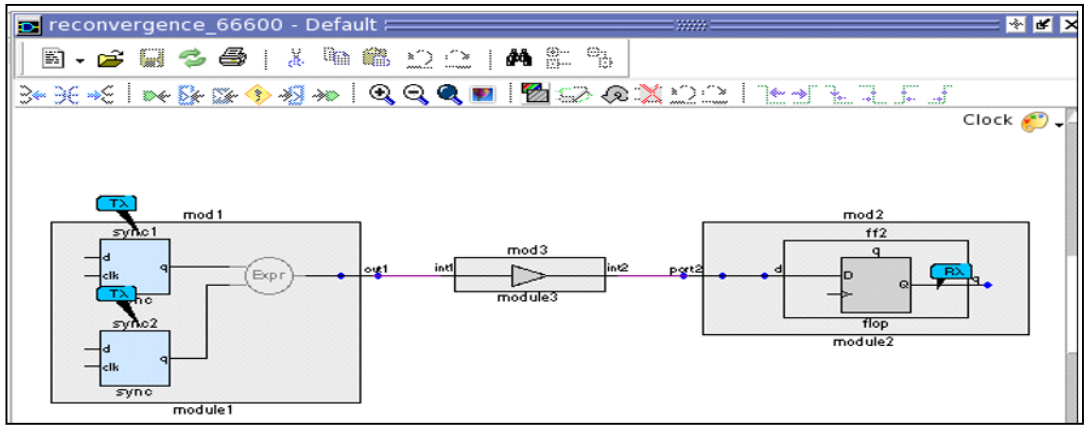
Hierarchical Analysis Results

- CDC and reconvergence checks run quickly
  - using HDMs on one of the SoCs
- SoC contains 509 HDMs

These results clearly show the advantage of using HDMs.

	Turnaround Time / Memory		
	MPU (TOP)	IP1	IP2
Register Count (M)	~.40M	~.16M	~.32M
Number HDM Blocks	509	2	1
CPU Run Time	~25Min	~17Min	~7 Hours
Max Memory	23.4 GB	4.3 GB	15.3 GB

- Reconvergence issue involves three IP blocks:
  - mod1, mod 2, and mod 3
- HDMs encapsulate the schematic
- Top-level shows the full reconvergence path
  - even when the three IP blocks are HDMs



Let's Talk About Communication – It's Hard!

It is probably not necessary to write an assertion that power is not ground

Portuguese

Provavelmente não é necessário escrever uma afirmação de que o poder não está fundamentado

English

It is probably not necessary to write a statement that power is not founded

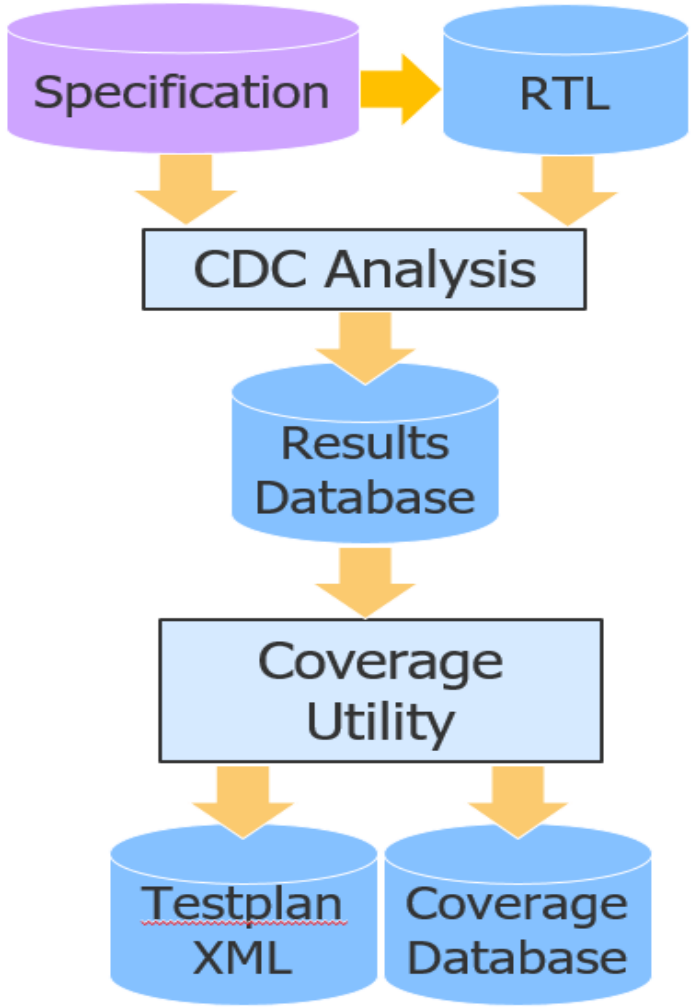
Courtesy Google Translate

Can We Eliminate the Communication Gap?

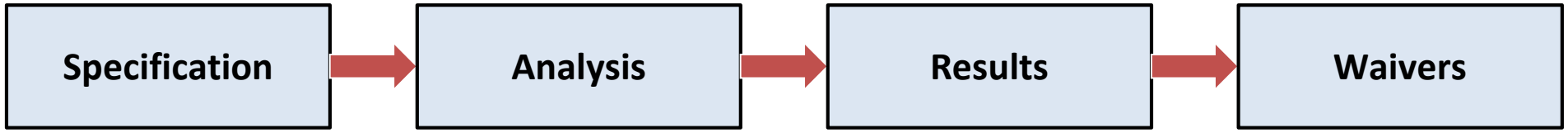
Treat the Specification as the Golden Source

Specification-based tool flows fill the gap
e.g. for a CDC or RDC flow

- Specification is an important piece
- Guides both design and analysis
- Defines final closure requirements



Intent Verification: Specification Flow

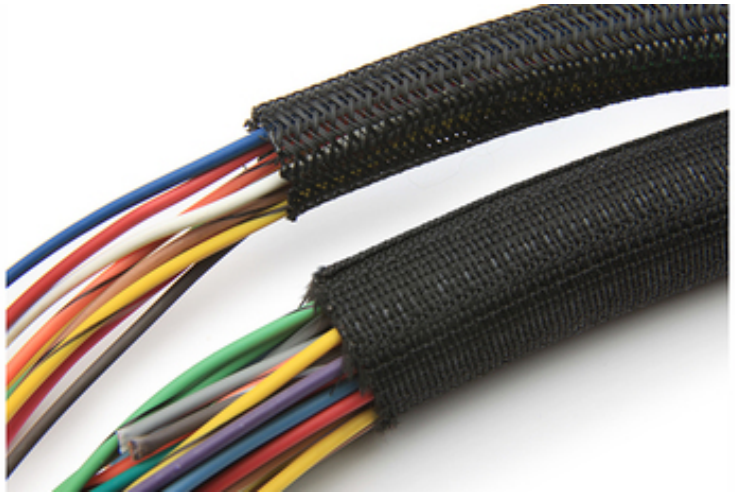


- Objective
  - Specification to drive design and intent verification
- Benefits
  - Reduces the effort in results review and waiver specification
- Use Model
  - Specification constraints **guide** intent verification (e.g. CDC, RDC) analysis
  - Specification constraints **reduce** intent verification (e.g. CDC, RDC) violations
  - Designers do not review specification paths (correct by specification)

Questa CDC supports a Specification Flow today

Summary

- Projects are threatened by information loss across gaps
- Hierarchical abstraction bridges these gaps
  - Modeling must be accurate and provide high-visibility to be effective
- Documentation-rooted methodologies bridge these gaps



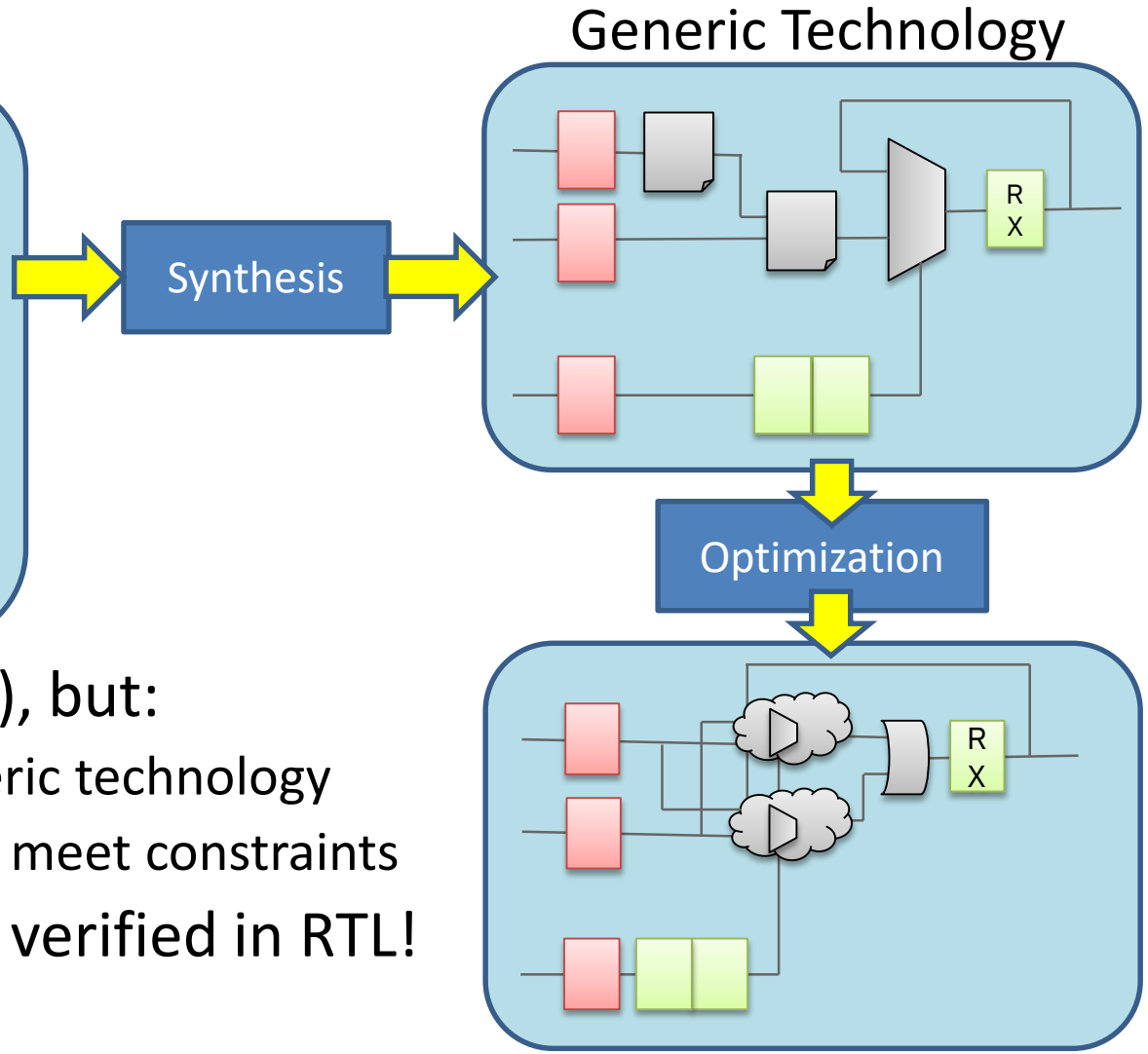
We Don’t Build RTL in Silicon. Verifying RTL is Not Enough.

CLOSING THE DESIGN/IMPLEMENTATION GAP

Intent Verification: Is Your Intent Preserved?

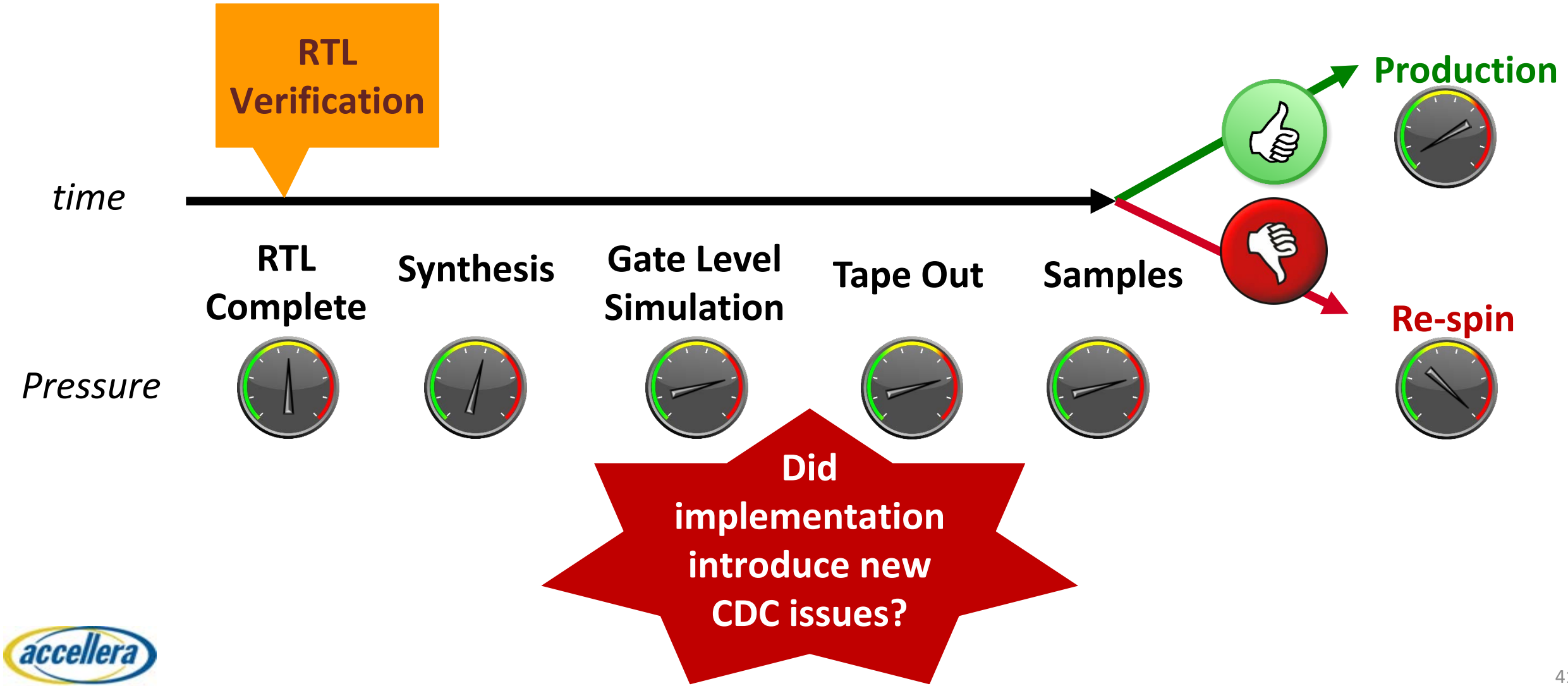
RTL

```
always @(posedge rx_clk)
 begin: DMUX
 s1 <= tx_sel;
 rx_sel <= s1;
 if (rx_sel)
  rx_data <= tx_data1 && tx_data2;
 else
  rx_data <= rx_data ^ 4'b1111;
end
```

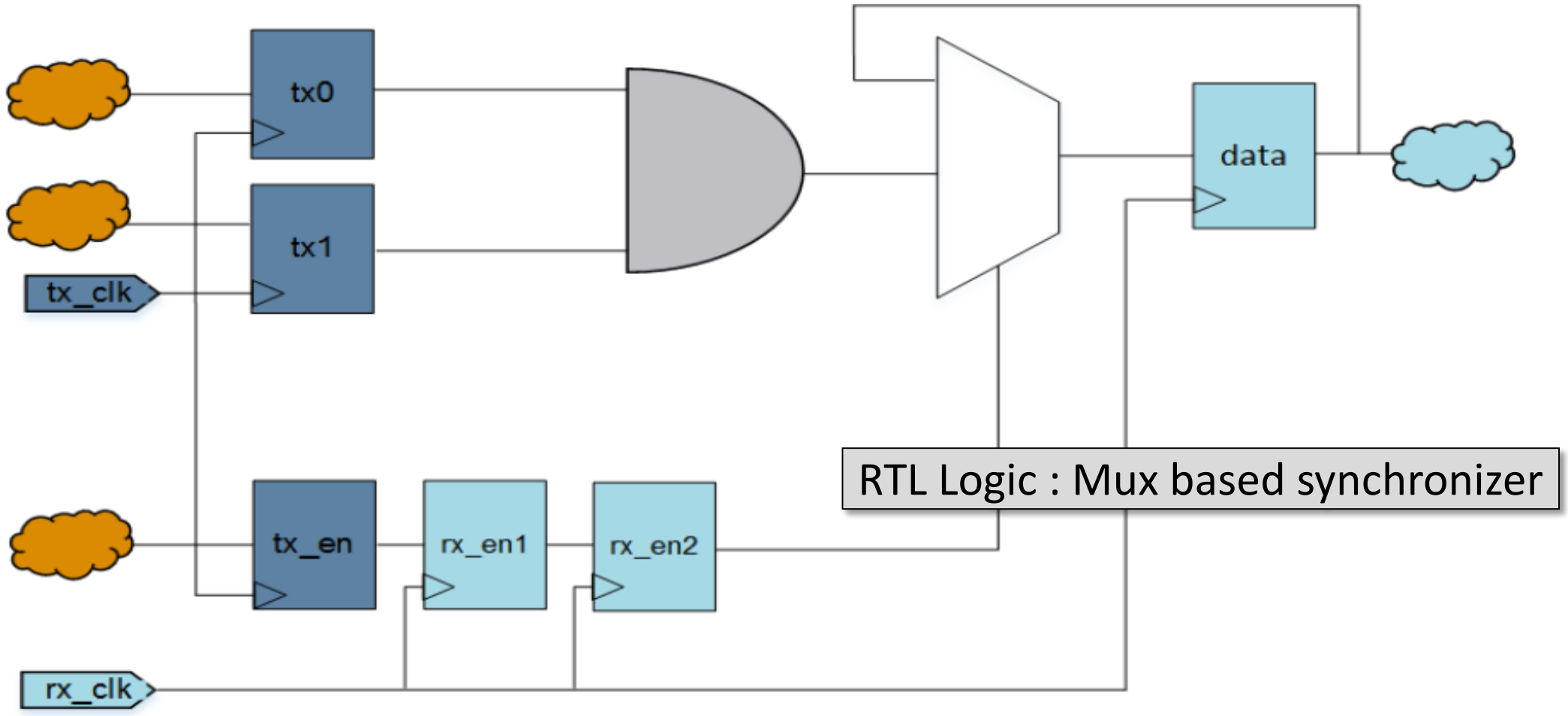


- Designer verifies intent at RTL (usually), but:
  - RTL mapped to logical equivalents in generic technology
  - Further optimized to target technology to meet constraints
- What is built in silicon is not what was verified in RTL!
  - This issue ***should not*** be ignored

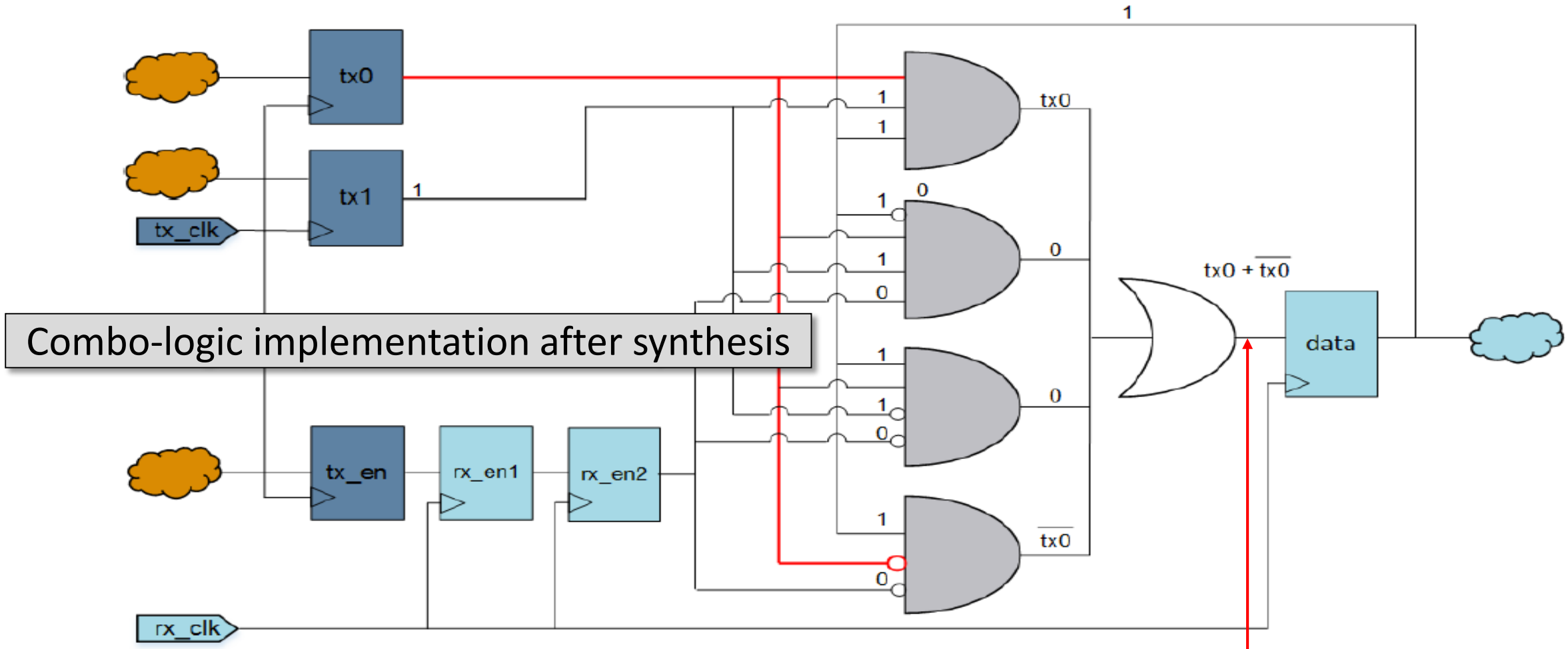
You Verified CDC at RTL. ☑Intent, Right?



Synthesis Glitch Example: Functional



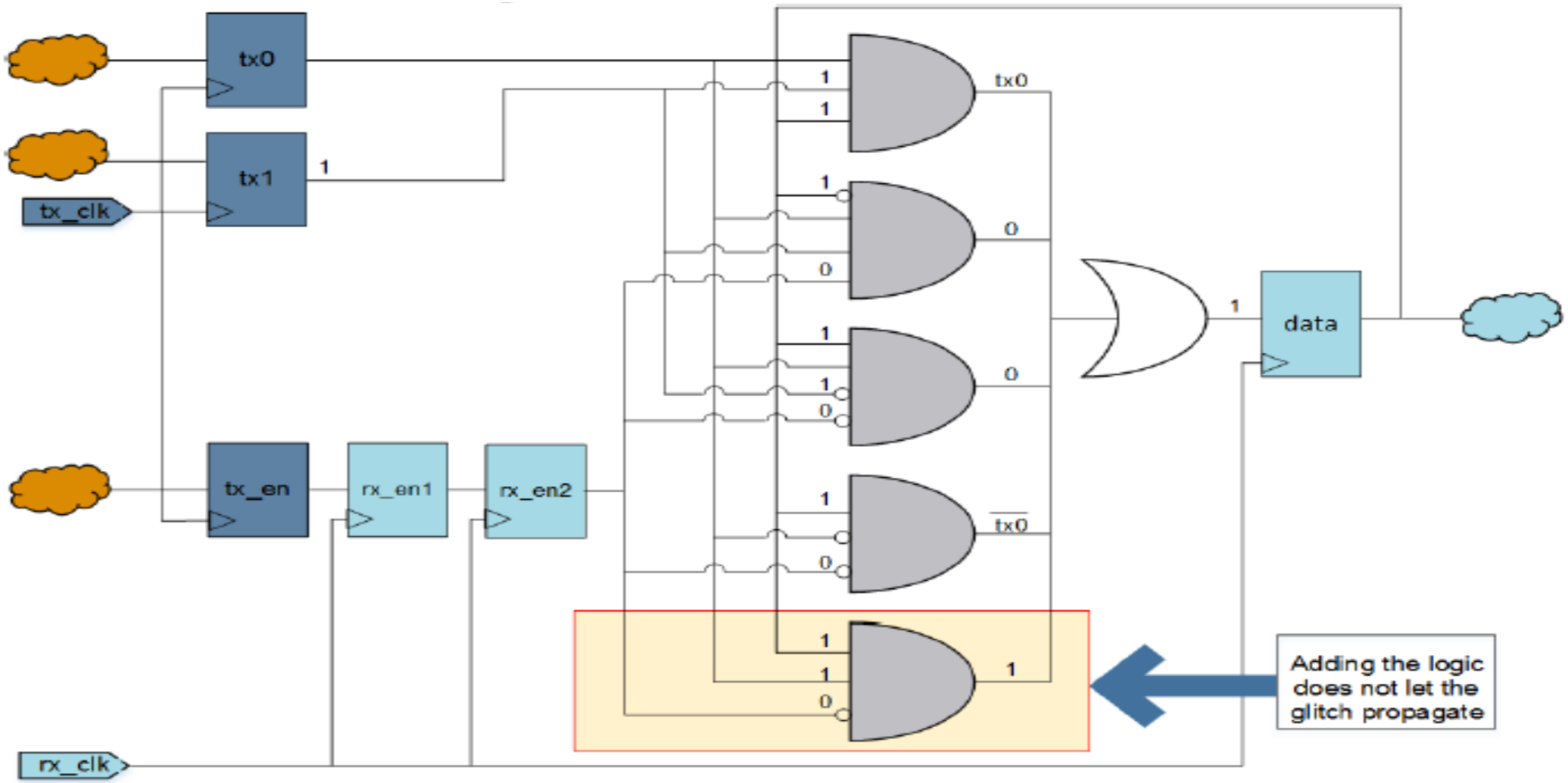
Synthesis Glitch Example: Synthesized



Combo-logic implementation after synthesis

For given constants, logic reduces to (tx0|~tx0) which causes glitch

Synthesis Glitch Example: ECO



Manual Detection & Correction

- Detection:
  - Get lucky with gate-level simulations
- Correction:
  - Determine if asynchronous crossing, maps to RTL
    - name mappings, bit-blasting effects
  - Consider RTL constraint applications and waivers
  - Functionally analyze if glitch can occur
  - Add logic to the term to prevent the glitch

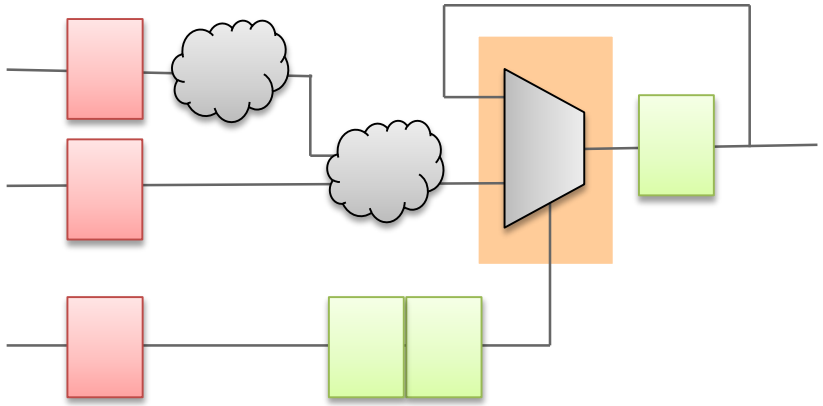
Closing the Design/Implementation Gap

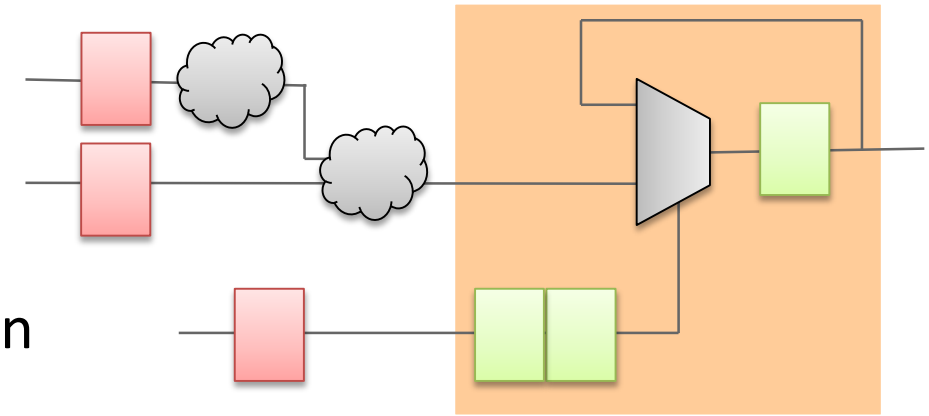
How do you avoid falling in the gap?

- Prevention
  - Protect the DMUX paths
  - Guide the implementation
- Verification
  - Ensure signoff quality before tapeout

Prevention: Protecting DMUX Paths

- Infer the DMUX
  - Specify synthesis pragma to force DMUX implementation
    - Synthesis maps to a n-input mux or a tree of muxes
- Instantiate the DMUX from technology library
  - Inefficient for muxes with constant inputs
  - Need to have n-to-1 muxes and bus muxes
- Utilize synchronizer library
  - Protect the synch structures from synthesis optimization



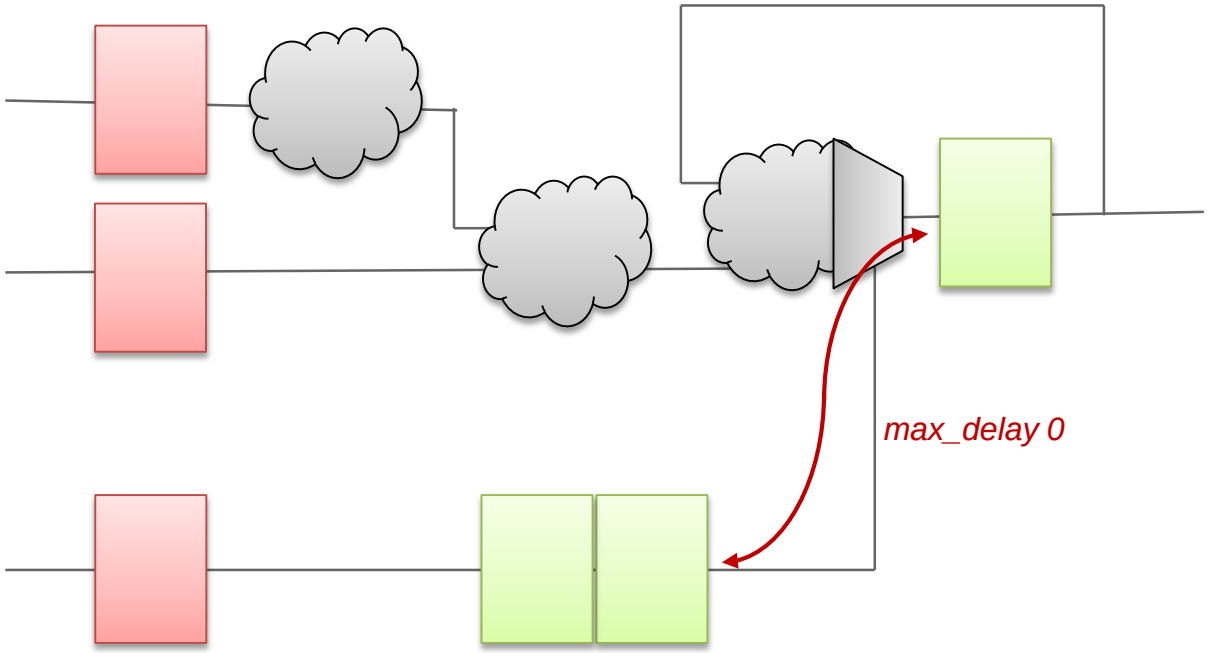


Requires RTL changes and extra design review steps
Optimization can still move muxes (especially 2-to-1 mux) to LHS

Prevention: Guiding Synthesis

Use timing constraint

- to ensure the dmux is at the end of the fan-in cone to the RX register
- set_max_delay 0 -from <sel_sync_out> -to <rx_d>



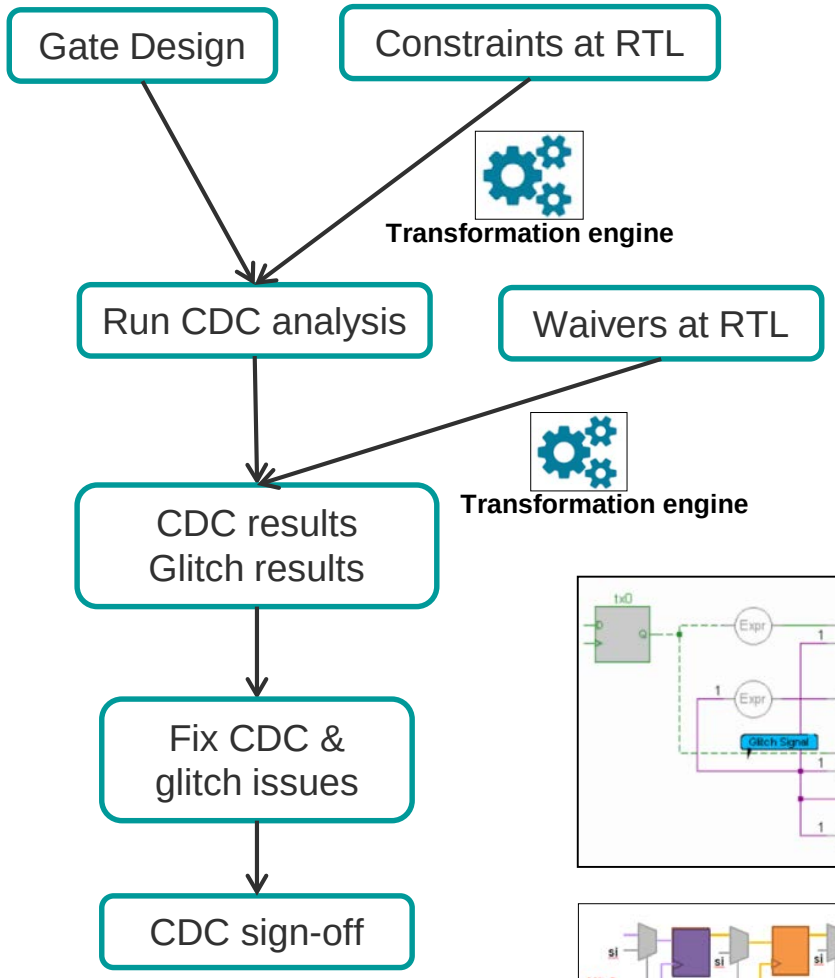
Does not require RTL changes
Optimization can still move muxes (especially 2-to-1 mux) to LHS
These paths will be reported as timing violations

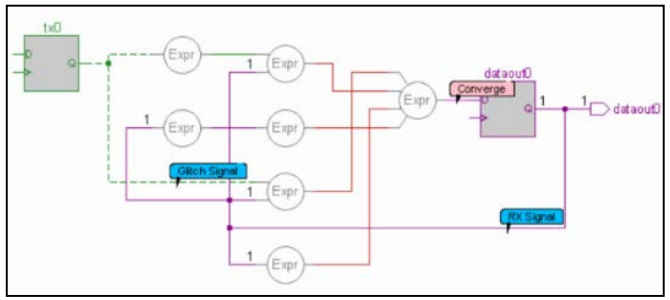
Verification: Questa Signoff CDC

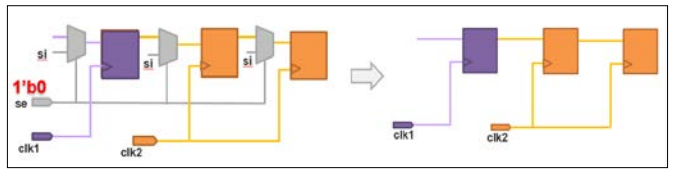
Setup

CDC Analysis

Debug







Automated Setup

- Constraints at RTL imported
- No iteration

Reduced Noise

- Bit-blasted paths regrouped
- Test logic automatically disabled

Debug Eased

- Waivers at RTL imported
- Correlation with RTL signals

Glitches Detected

- Structural & functional analysis of combinational logic for glitch
- Focused glitch debug

Glitch Check: Report

Glitch node and glitch signal

- With clock group
- Scheme name

Signal assignments exposing glitch

Paths with possible glitch

- Could not prove with formal

Report logic under glitch condition

```
Questa CDC Glitch Checks Report
=============================================================
Section 1 : CDC Glitch Information
Glitch Signals           : 1 (Impacts 1 CDC paths)
  1.1. Proven Glitch Signals (Static)  : 1 (Impacts 1 CDC paths)
  1.2. Proven Glitch Signals (Dynamic) : 0 (Impacts 0 CDC paths)
  1.3. Possible Glitch Signals         : 0 (Impacts 0 CDC paths)
-------------------------------------------------------------
1.1. Proven Glitch Signals (Static) : 1 (Impacts 1 CDC paths)
-------------------------------------------------------------
1   Glitch signal    : tx0 (tx_clk)
      Converging point : rx0
        Receiving nodes  : data (rx_clk) (Static-1)
=============================================================
Section 2 : Glitch Propagation Conditions
-------------------------------------------------------------
2.1. Proven Glitch Signals (Static) : 1 (Impacts 1 CDC paths)
-------------------------------------------------------------
1   Glitch signal    : tx0 (tx_clk)
    CDC Paths        : 1
    Receiving node   : data (tx_clk)
    CDC ID           : partial_dmux_90614
    Glitch propagation condition :  <Value>   <Signal>
                                     1        data
                                     1        tx1
                                     0        rx_en2
```

Automotive Customer Results

- Two real glitches detected on the first design
- Verified 4 designs with Questa Signoff CDC
  - Design size ranging from ~1-10million

Design	Gate count	Crossings	Glitches Detected	Formal (multicycle) proved glitches	Runtime (s)	Memory (GB)
Design 1	~1M	3594	3	2	350	3.9
Design 2	~10M	297000	37	22	6661	40
Design 3	~5M	332258	313	20	2650	5.9
Design 4	~7M	128694	202	-	10310	23

Summary

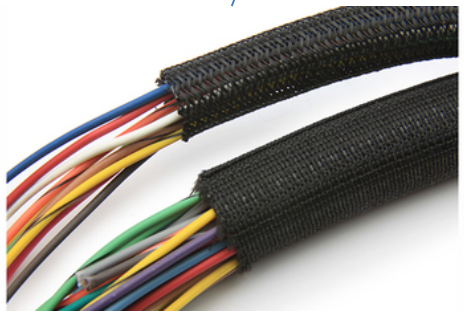
- Gap: implementation process can introduce new issues
  - Intent verification flows on RTL are not enough
- Manual analysis flows require significant effort
  - Work through constraints, waivers, name mapping, bit-blasting, renaming
  - Must analyze if the issue merits an ECO
- Automated tool flows are better-suited

SUMMARY

Verify that the product will work as intended

- Enable the designer to verify their intent was met, with an intent verification flow
- Align the verification tasks with organizational or skillset gaps

- Close the gap between what was verified and what is built



- Avoid lossy handoffs with strong and accurate modeling
- Automate generation and checking of constraints and assertions from documentation

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

We appreciate your attendance and hope you'll watch your step

THANK YOU

accellera SYSTEMS INITIATIVE