REAL INTENT

Verification Futures 2025

Static Sign-Off Methodologies:

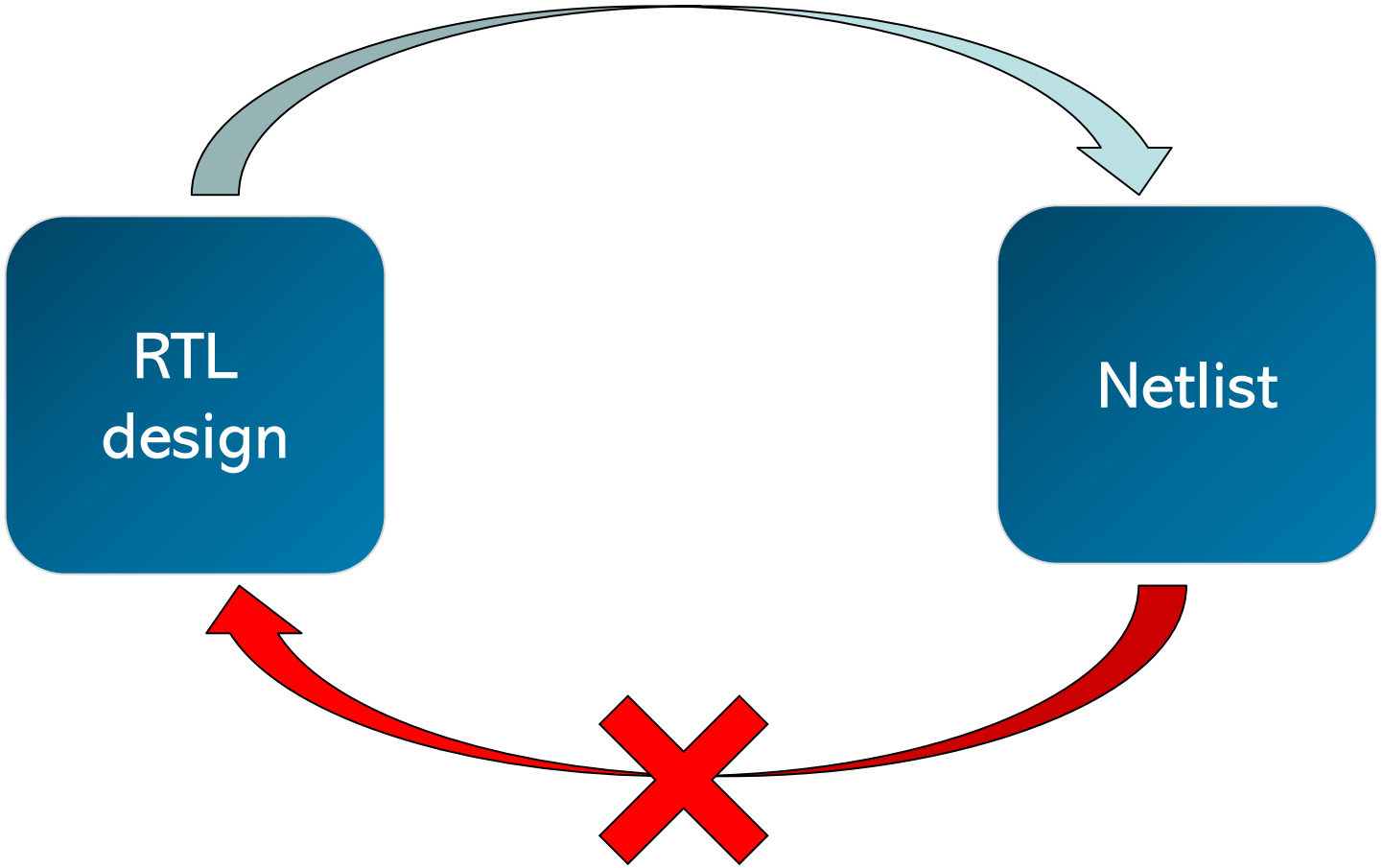
Liberating Functional Verification from Boolean Shackles

Prakash Narain

November 12, 2025

Early functional verification & sign-off

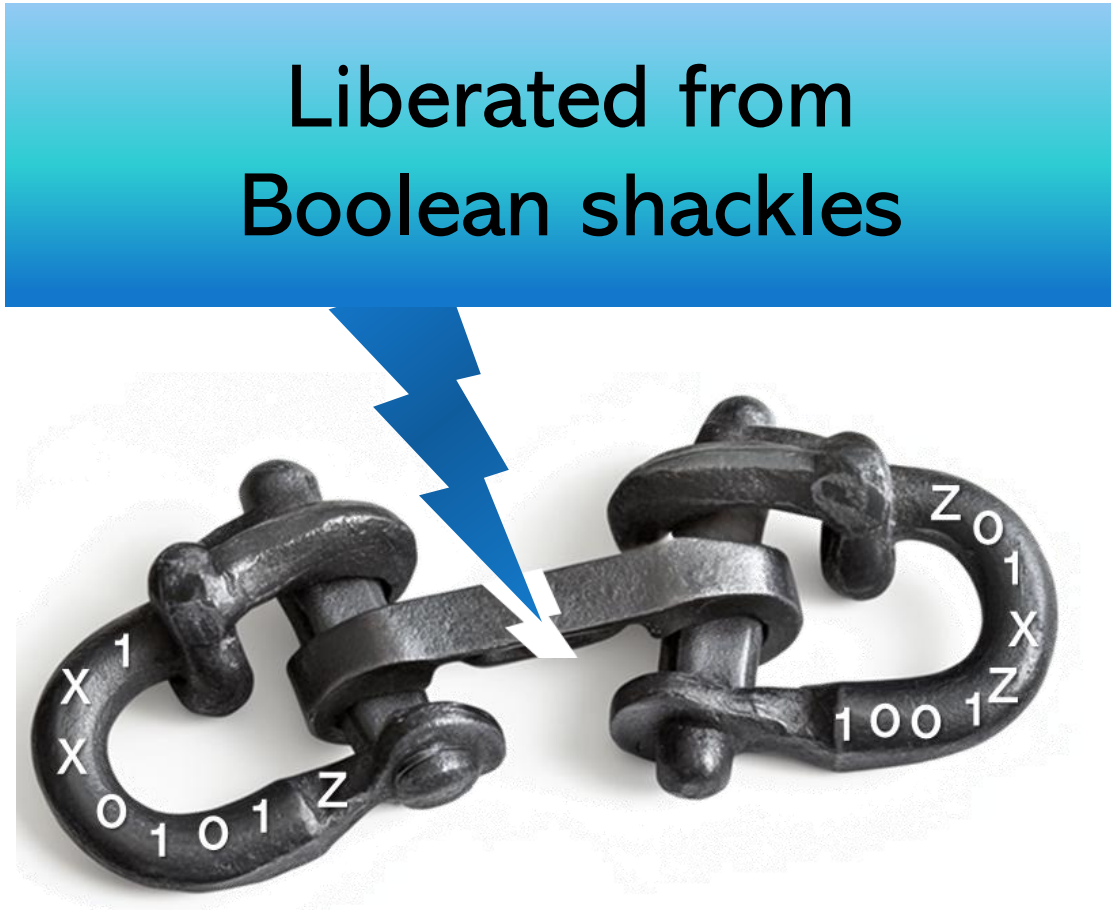
Avoid costly RTL & netlist ECOs/iterations



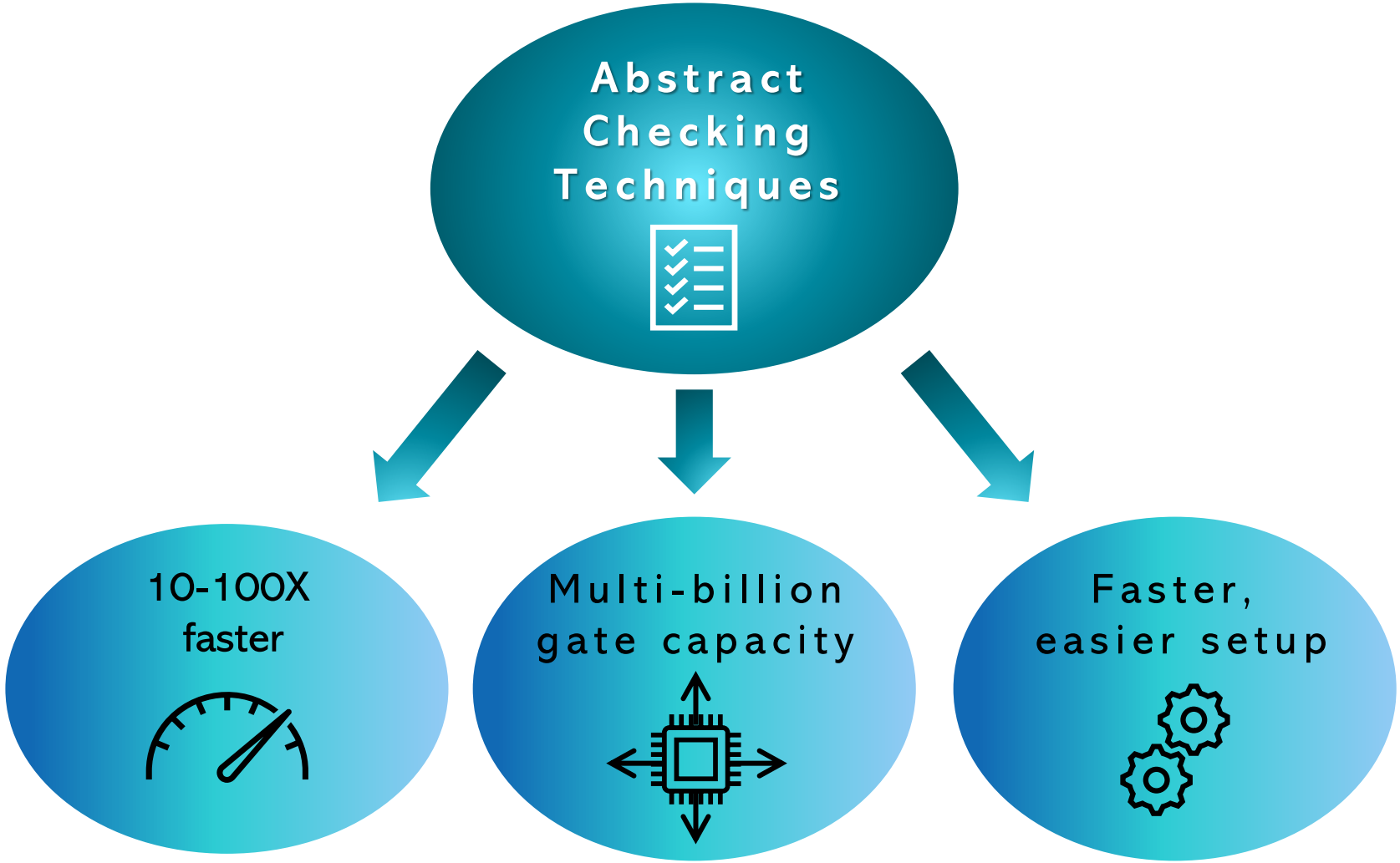
Simulation & formal: Boolean analysis

Early RTL design needs a different approach

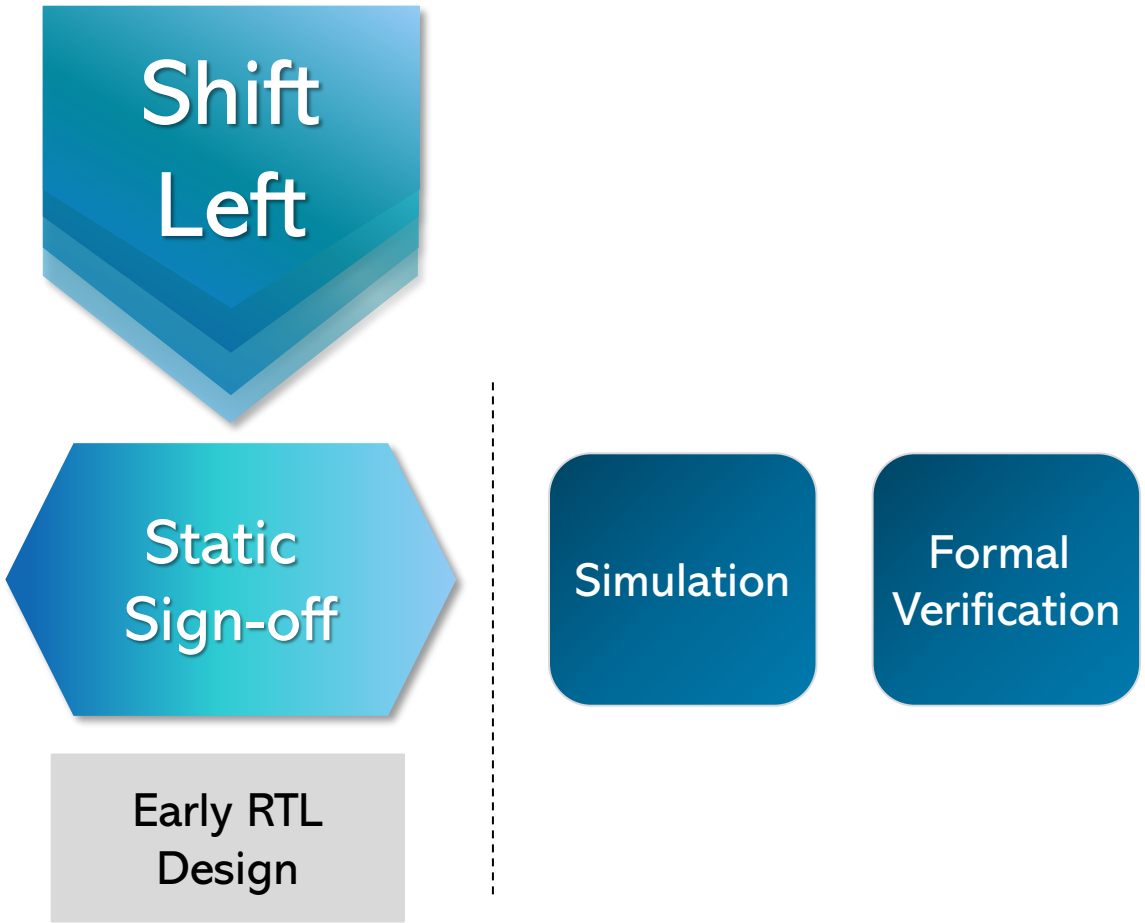
Static sign-off: Minimally Boolean



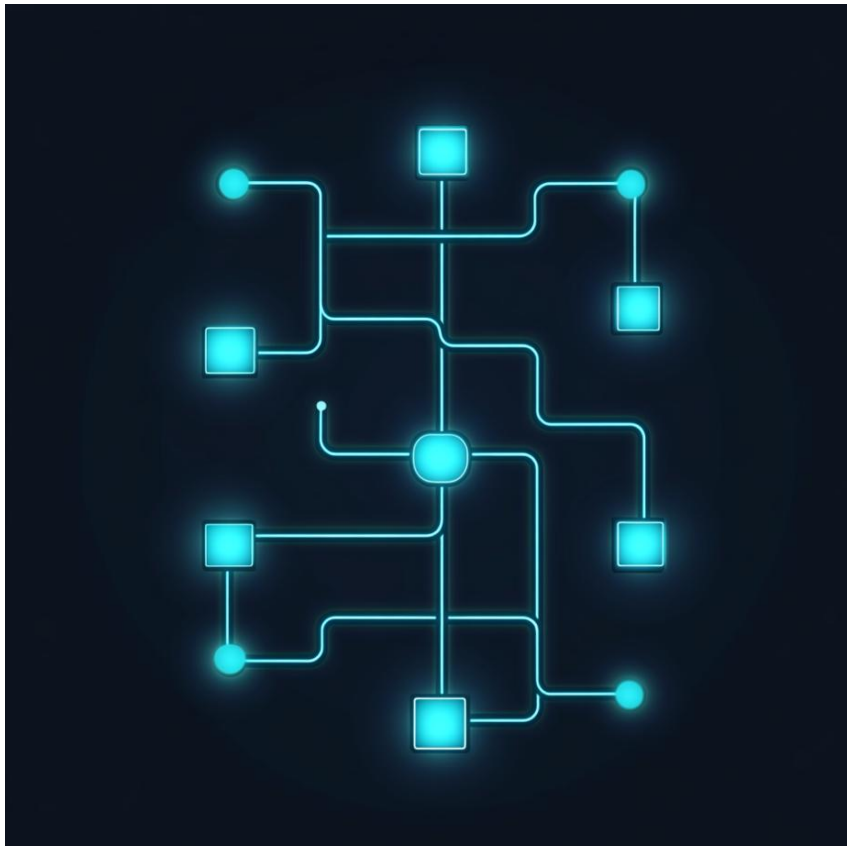
Static sign-Off: Abstract checking



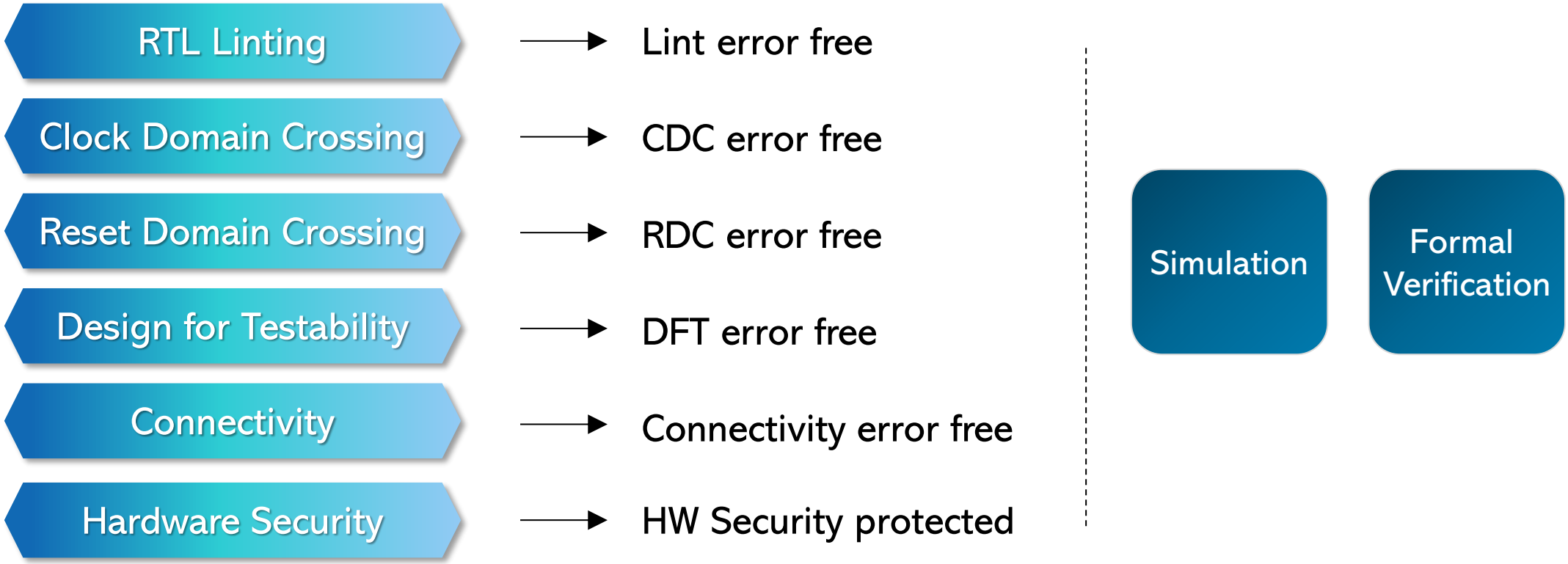
Static sign-off: Starts at early RTL design



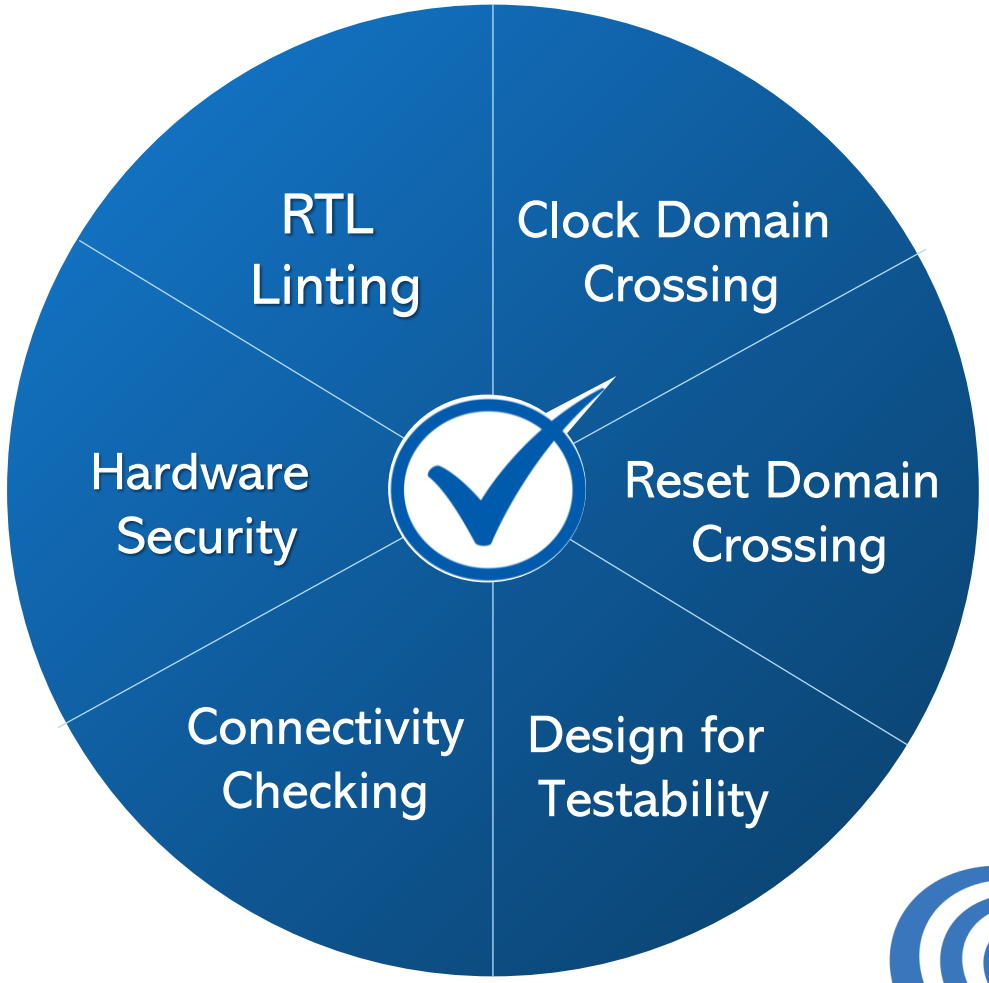
Static sign-off rule configurability



Expanding static sign-off applications

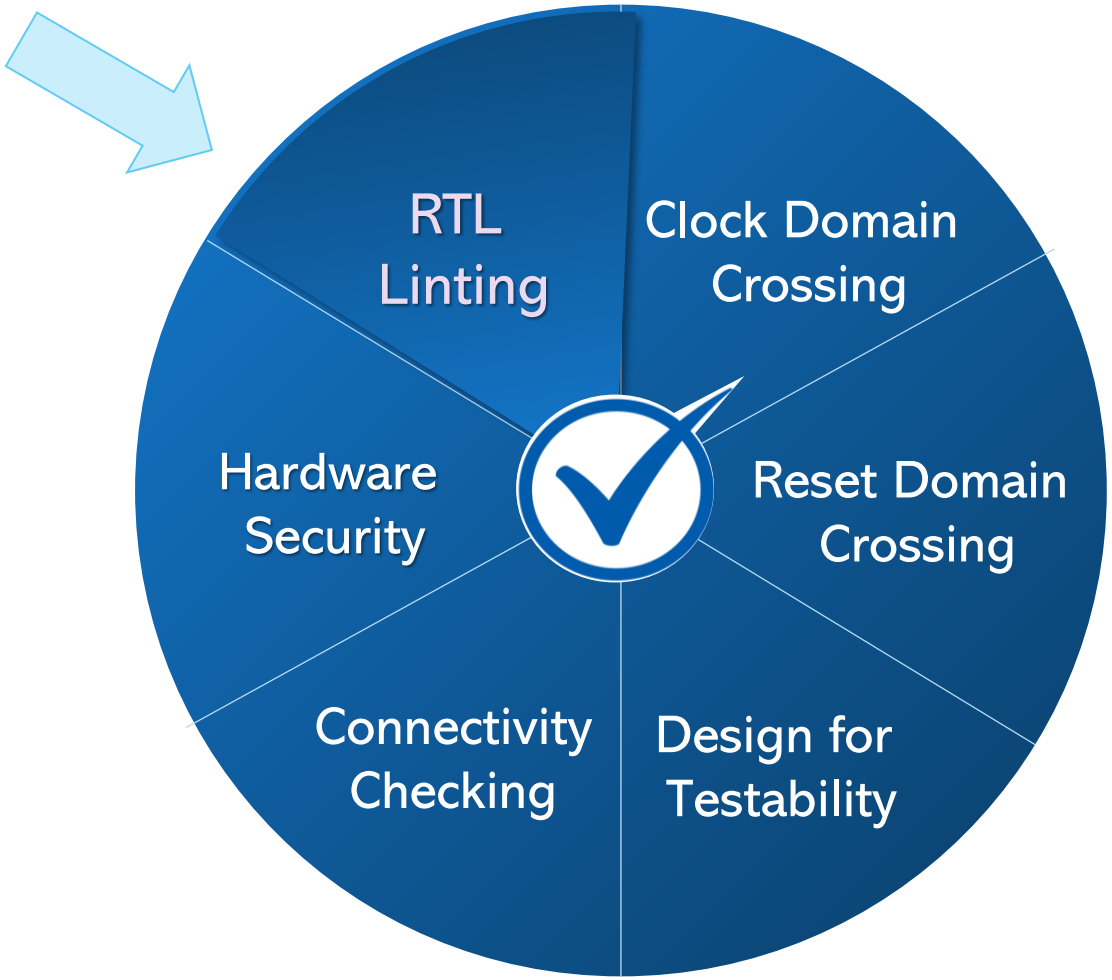


Advanced static sign-off methodologies



REAL INTENT

RTL Linting Sign-Off



1 High value rules

```
always @(en1 or en2)
begin
        if ( en1 )
            out1 = 2'b00;
        if ( en2 )
            out1 = 2'b11;
 end
```

Explicitly define intent

```
always @(en1 or en2)
begin
        if ( en1 )
            out1 = 2'b00;
        if ( en2 )
            out1 = 2'b11;
 end
```

Use “always_<spec>” to explicitly define intent

Explicitly define intent

```
always_comb
begin
        if ( en1 )
            out1 = 2'b00;
        if ( en2 )
            out1 = 2'b11;
 end
```

"always_comb" explicitly specifies combinational logic

Don’t use multiple sequential conditionals

```
always_comb
begin
        out1 = 1’b10;
      if ( en1 )
          out1 = 2’b00;
      if ( en2 )
          out1 = 2’b11;
end
```

Use if/then/else when there are multiple sequential conditional assigns

Don’t use multiple sequential conditionals

```
always_comb
begin
        if ( en2 )
            out1 = 2’b11;
        else if ( en1 )
            out1 = 2’b00;
        else
         out1 = out1
end
```

Clarifies that only 1 assign will be applied

Indentation aligns with construct

```
always_comb
begin
        if ( en2 )
          out1 = 2'b11;
        else if ( en1 )
          out1 = 2'b00;
        else
        out1 = out1;
end
```

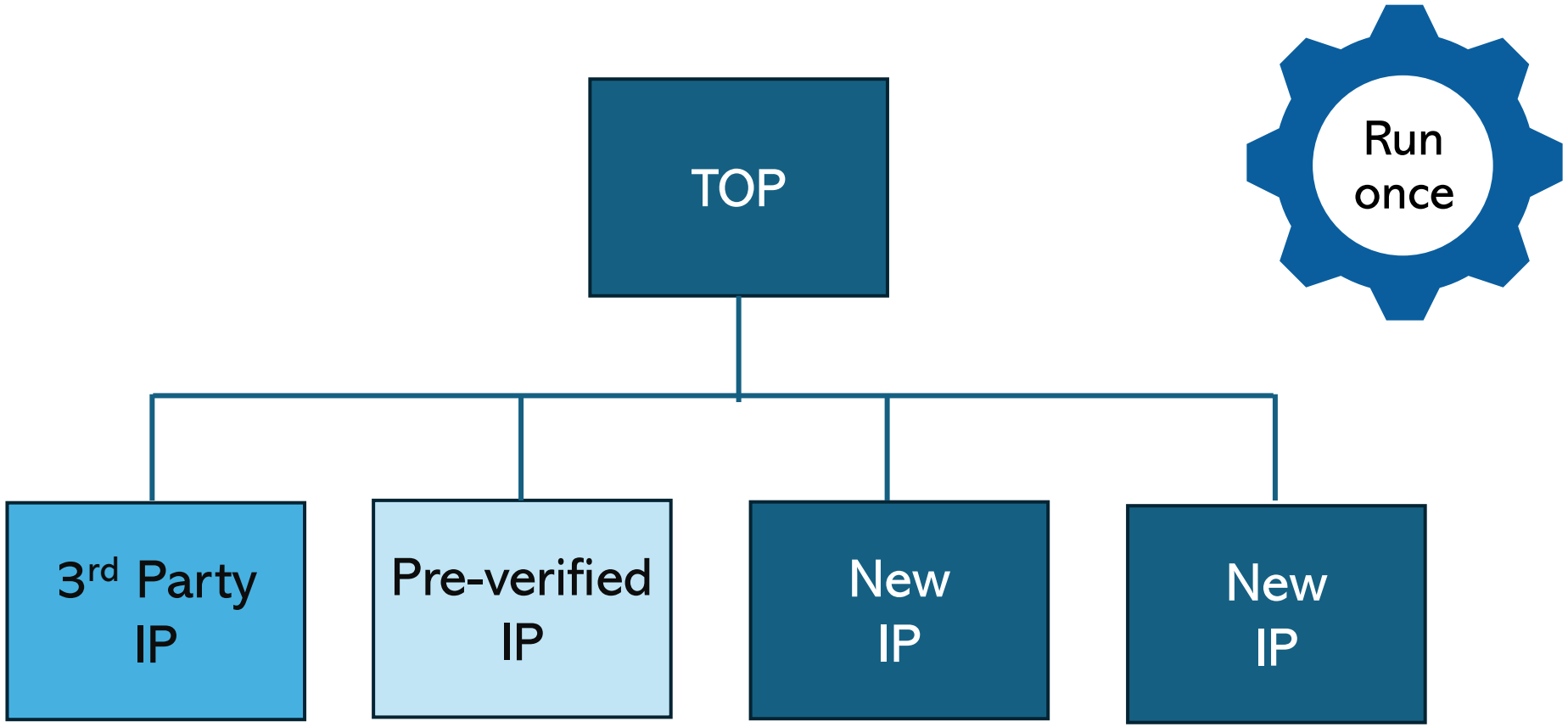
Indentation must align with construct

Indentation aligns with construct

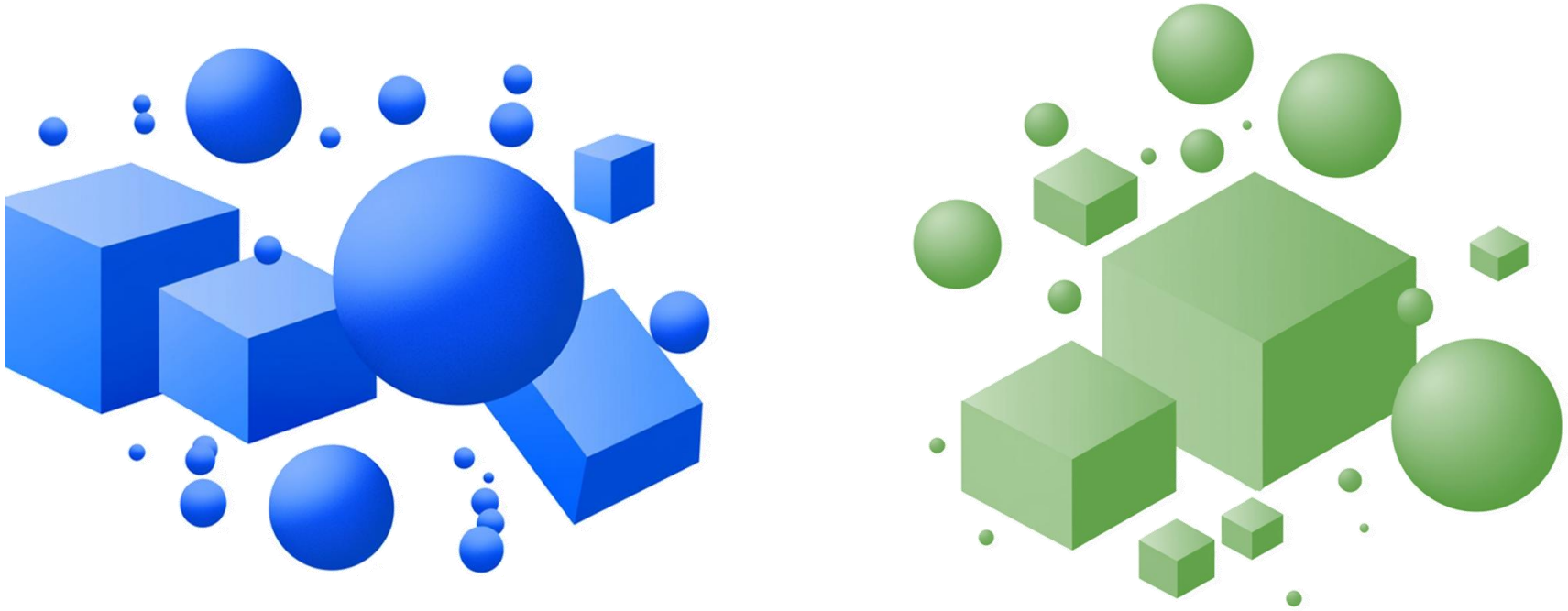
```
always_comb
begin
        if ( en2 )
          out1 = 2'b11;
        else if ( en1 )
          out1 = 2'b00;
        else
          out1 = out1;
end
```

Indentation corrected

2 Multi-policy linting



3 Targeted debug



Grouping violations

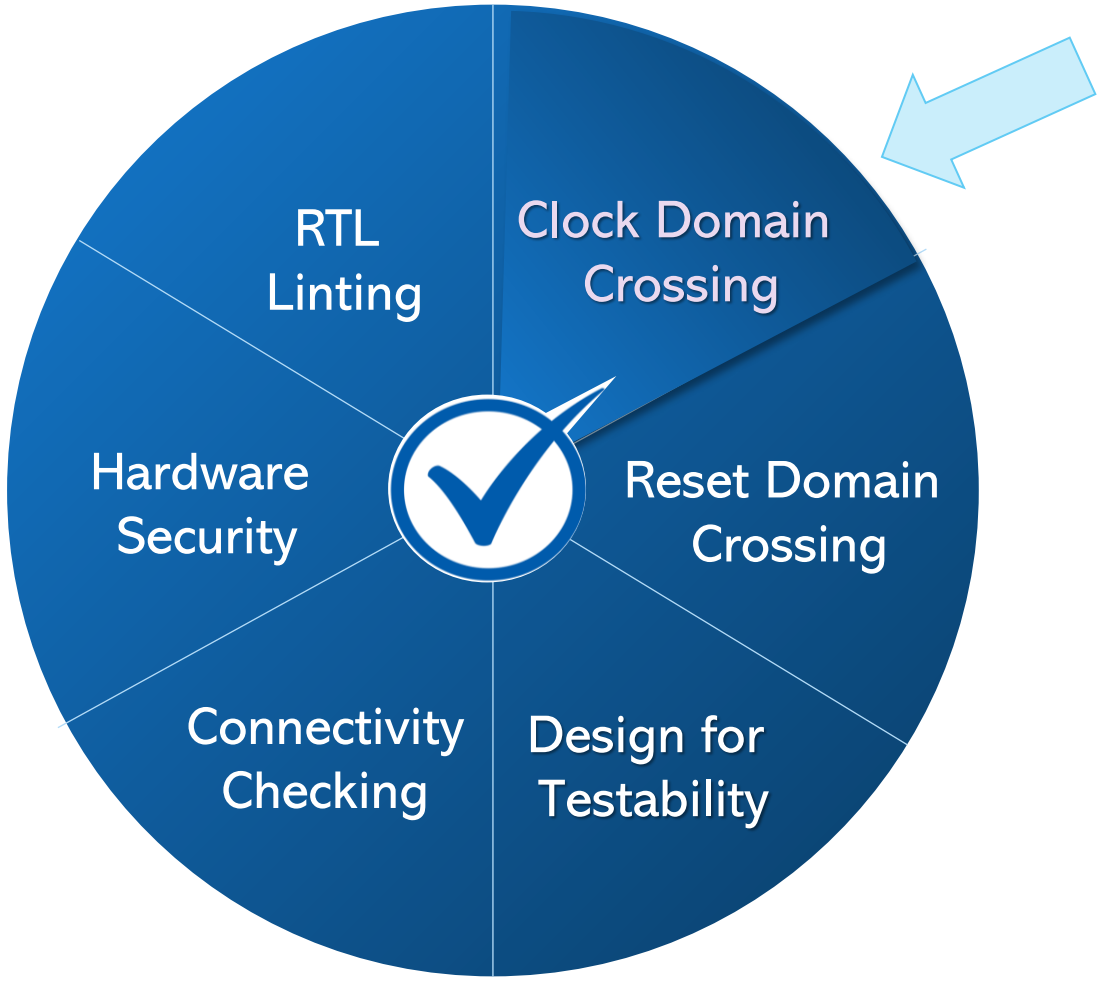
Advanced RTL linting sign-off

High value rules

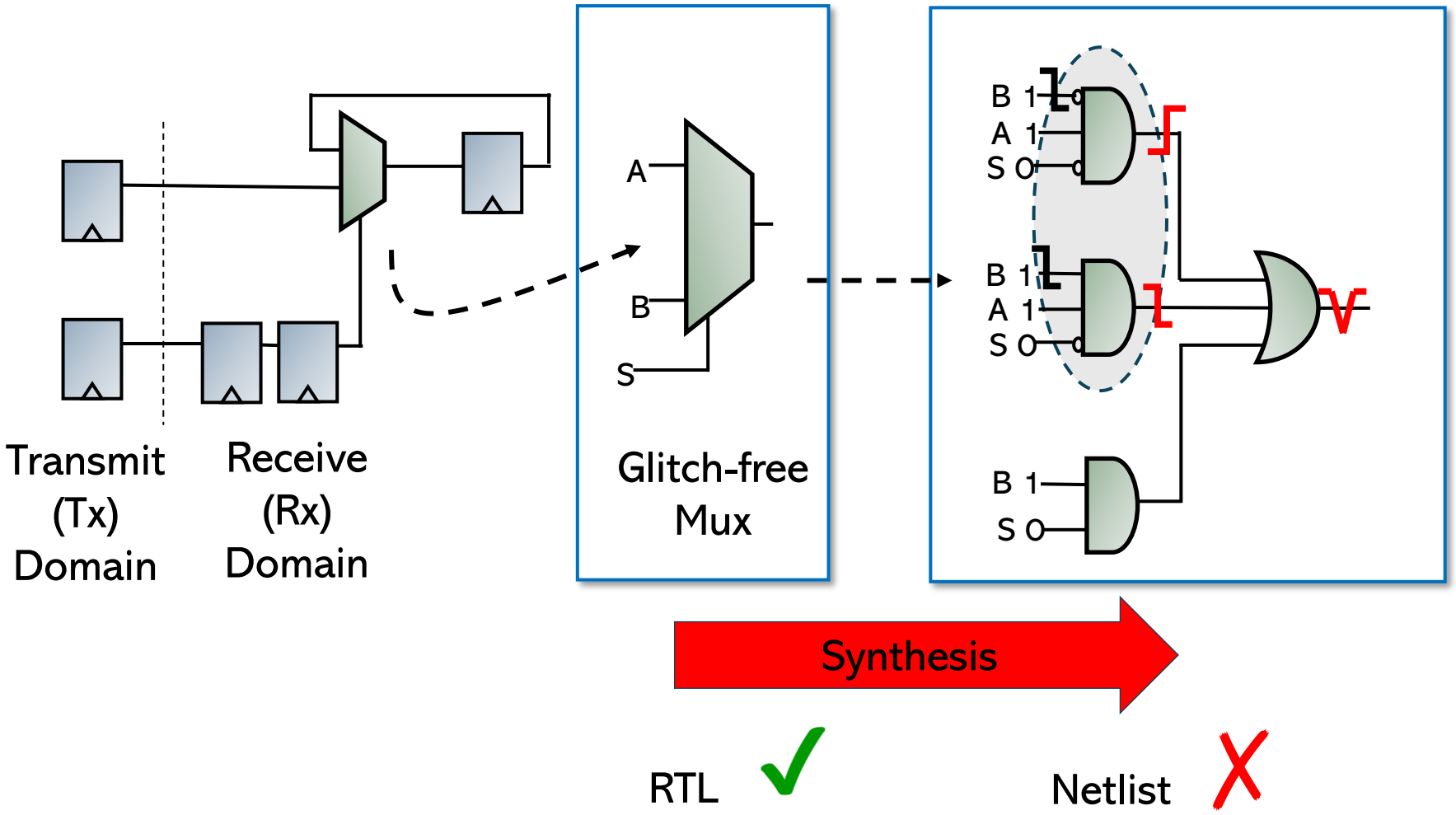
Target debug

Multipolicy runs

CDC Sign-Off



CDC glitch analysis

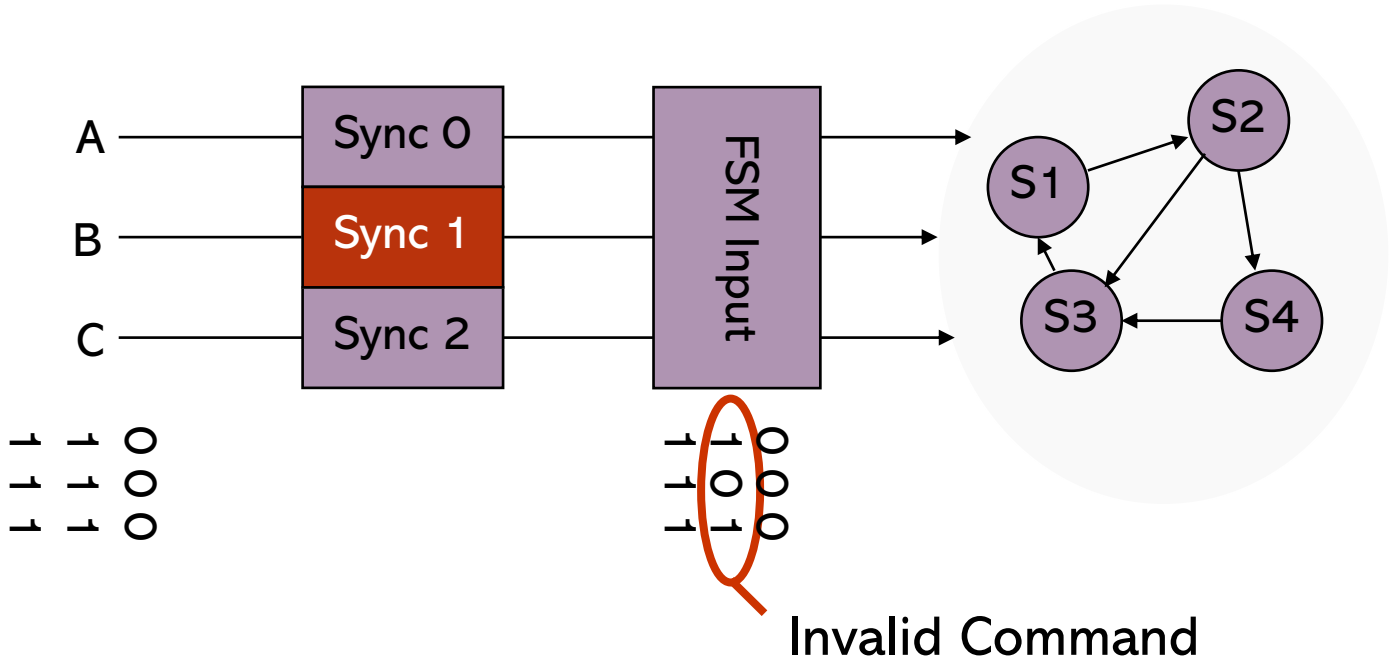


1 CDC glitch analysis

“Do not touch” MUXes

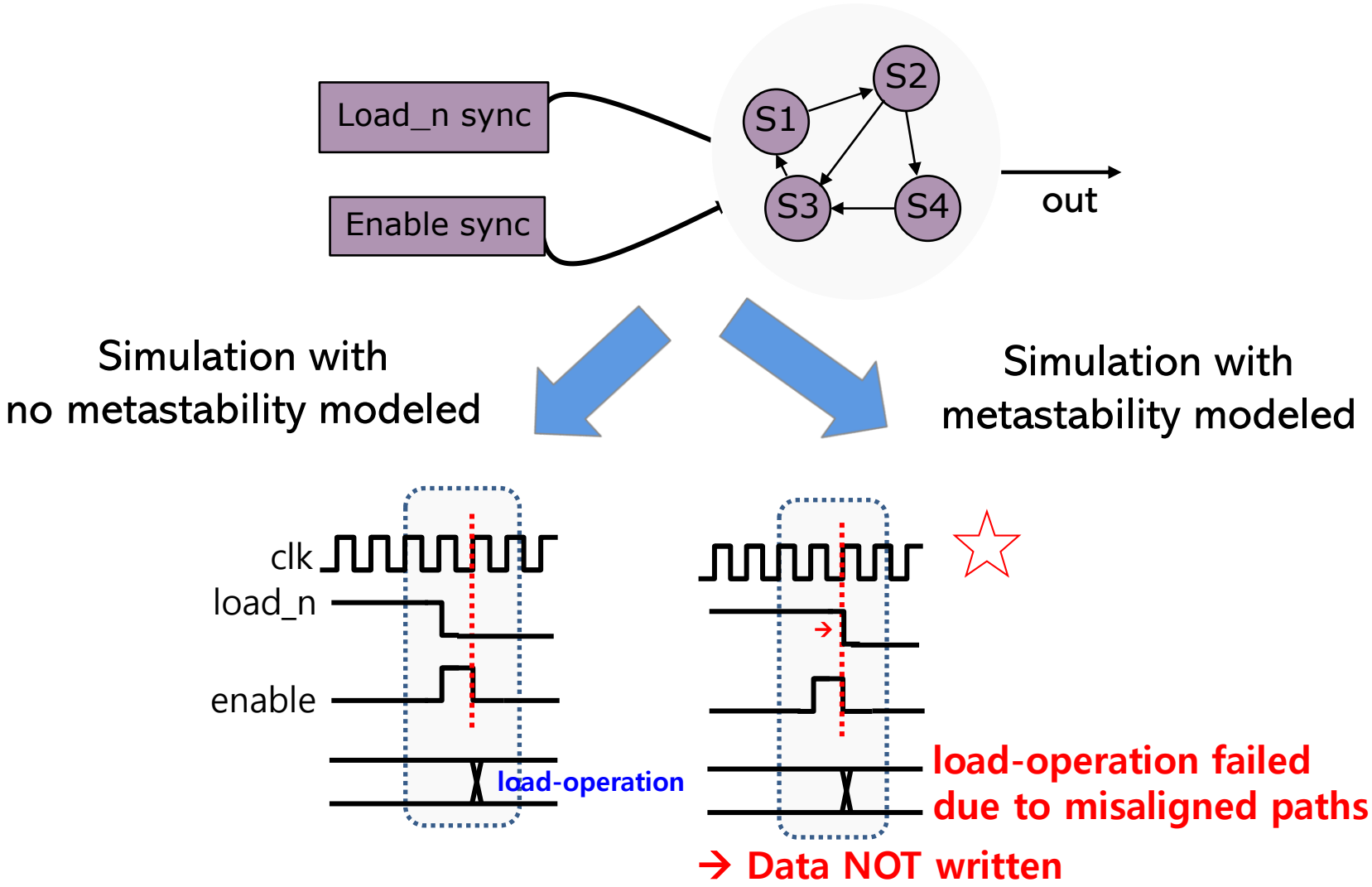
Netlist-level
glitch CDC sign-off

2 Dynamic CDC verification



Correlation loss

2 Dynamic CDC verification



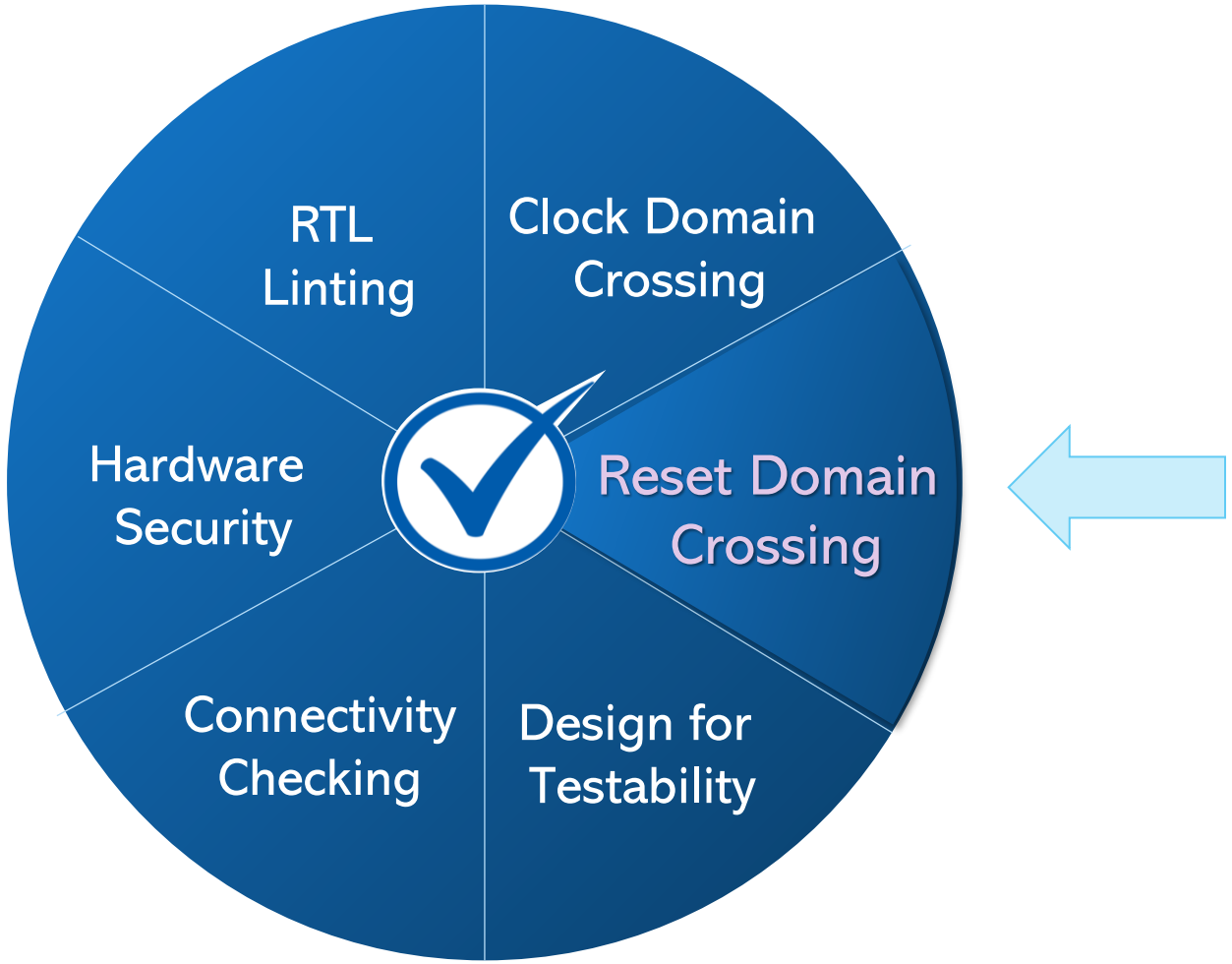
Advanced CDC sign-off

Structural sign-off

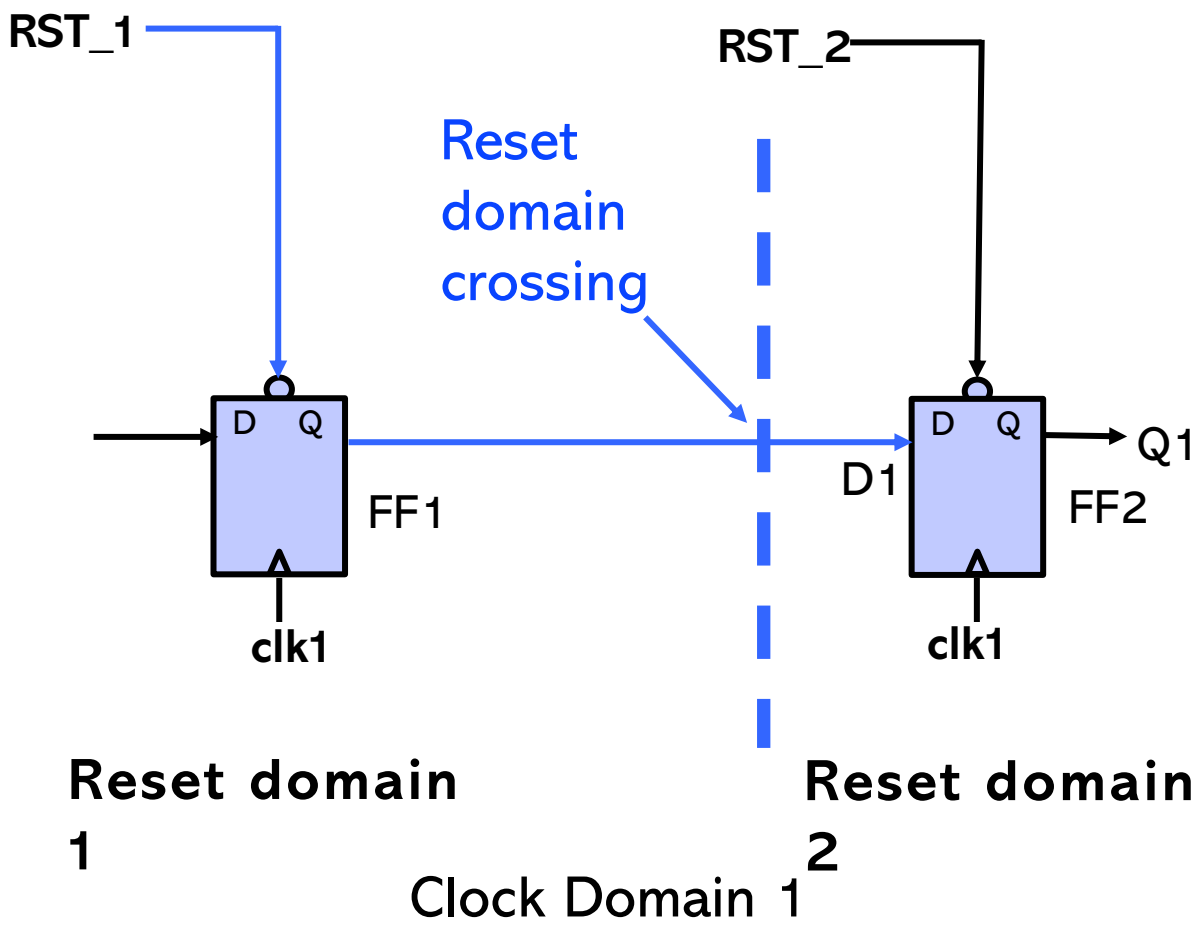
Glitch checking

Dynamic CDC verification

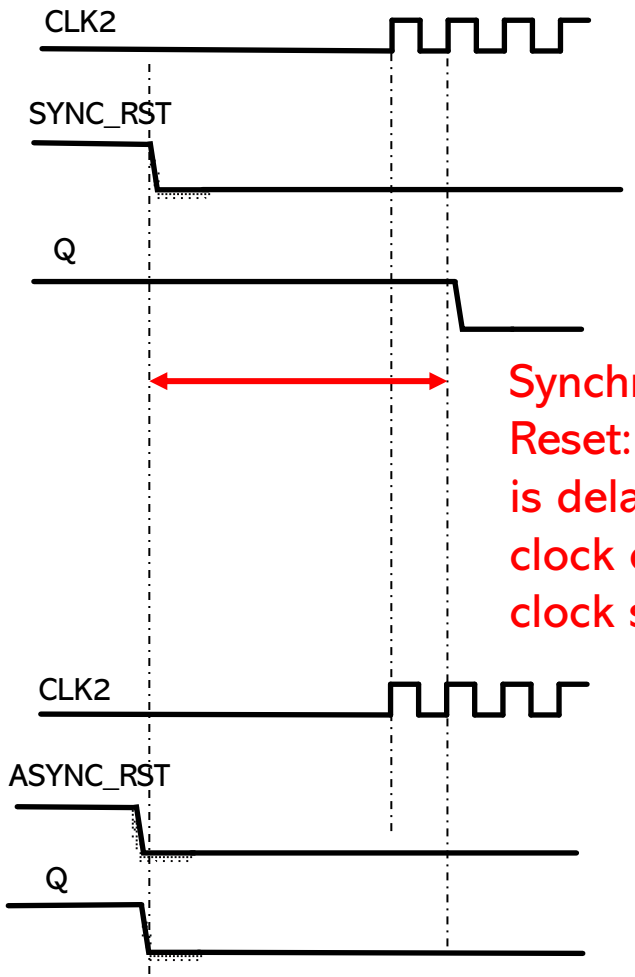
RDC Sign-Off



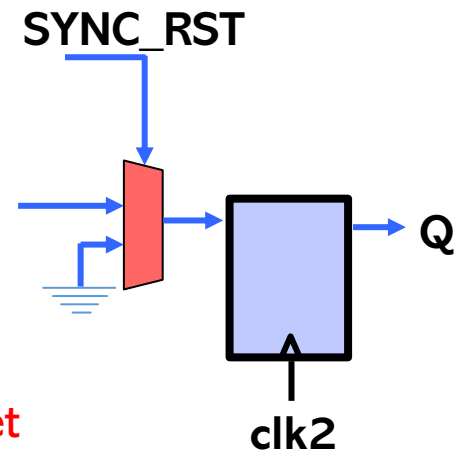
Reset domain crossing sign-off

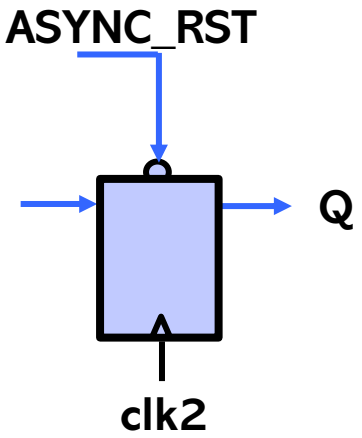


Increasing use of asynchronous resets

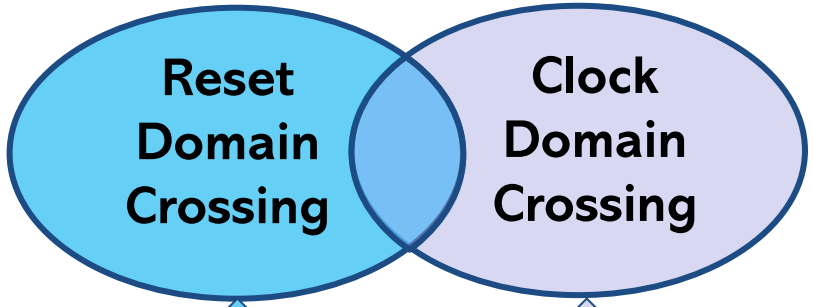


Synchronous Reset: Flop reset is delayed until clock edge after clock starts.





RDC key differences from CDC



- Errors can occur in same clock domain

- Analysis scope - Global

- RDC-specific analysis required to identify all RDC issues with low noise

- MTBF- High

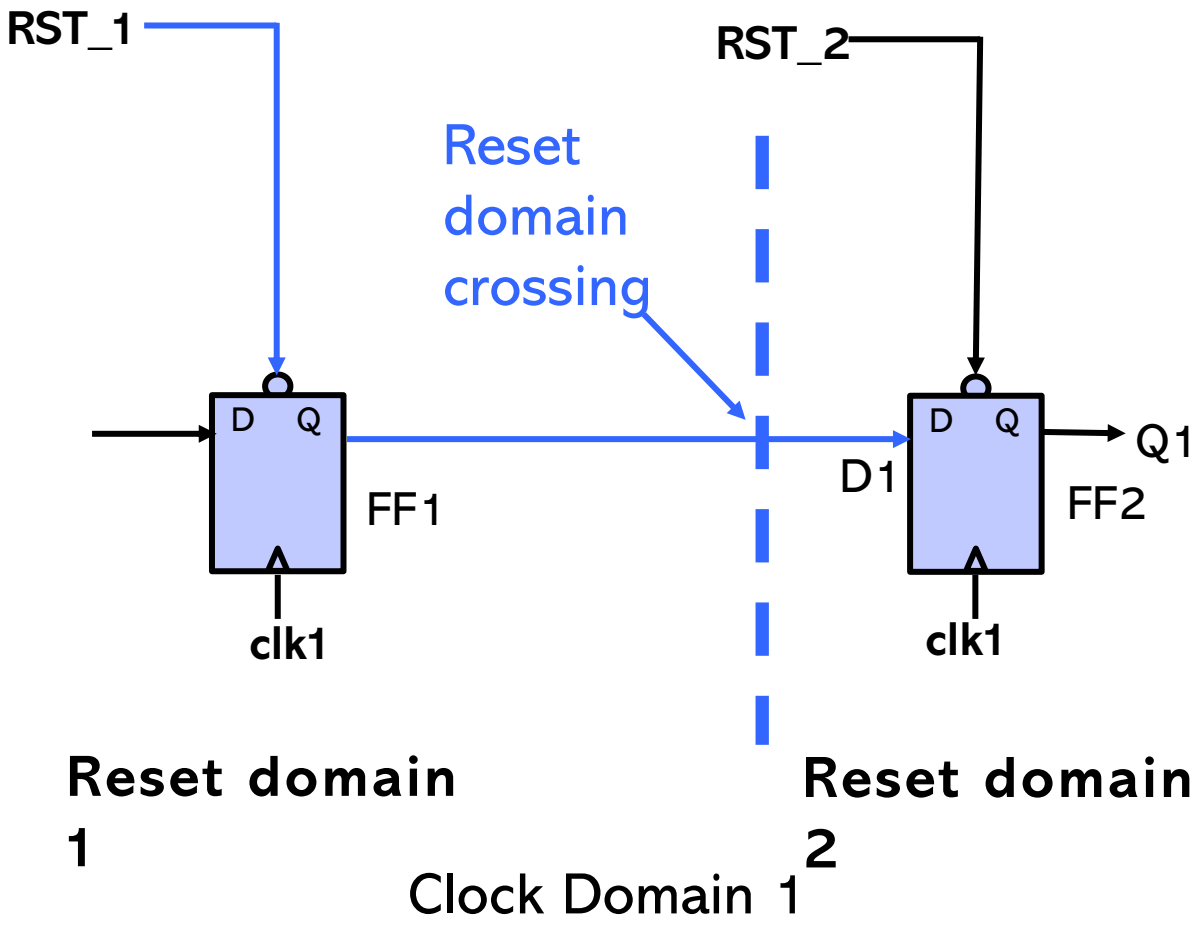
- Errors can occur across clock domains

- Analysis scope - Local at CDC interfaces

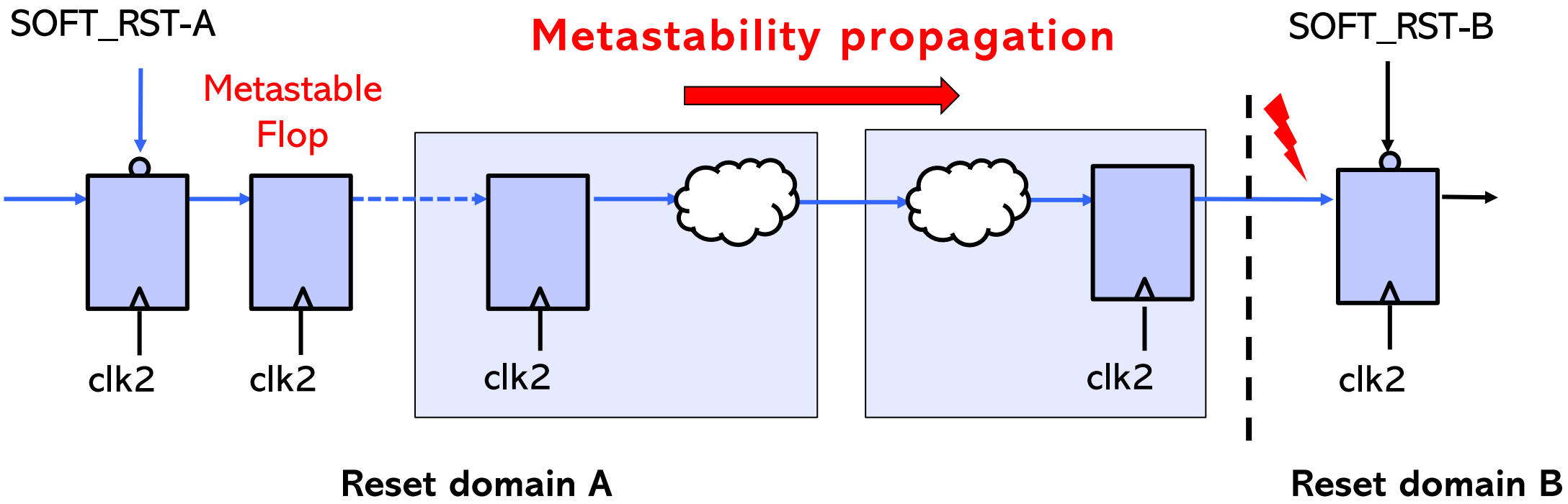
- CDC-specific analysis required to identify all CDC issues with low noise

- MTBF- Low

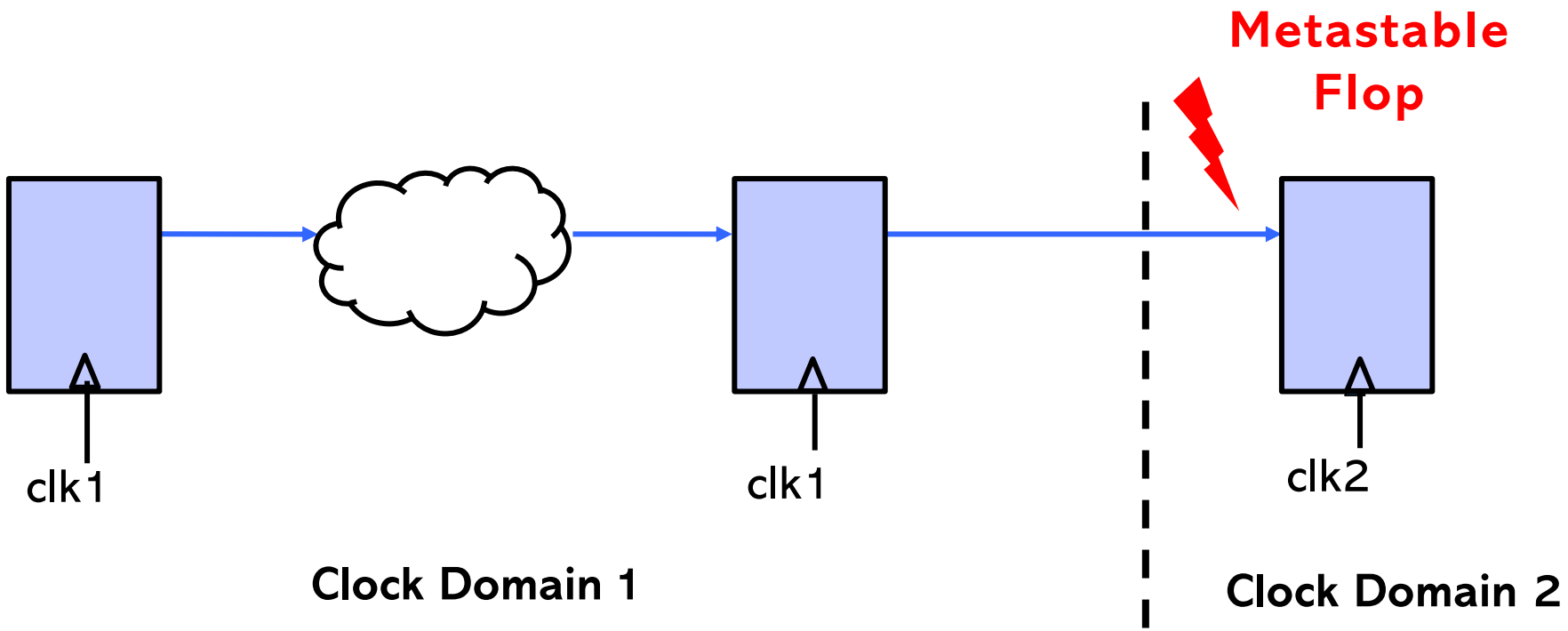
RDC errors can occur in same clock domain



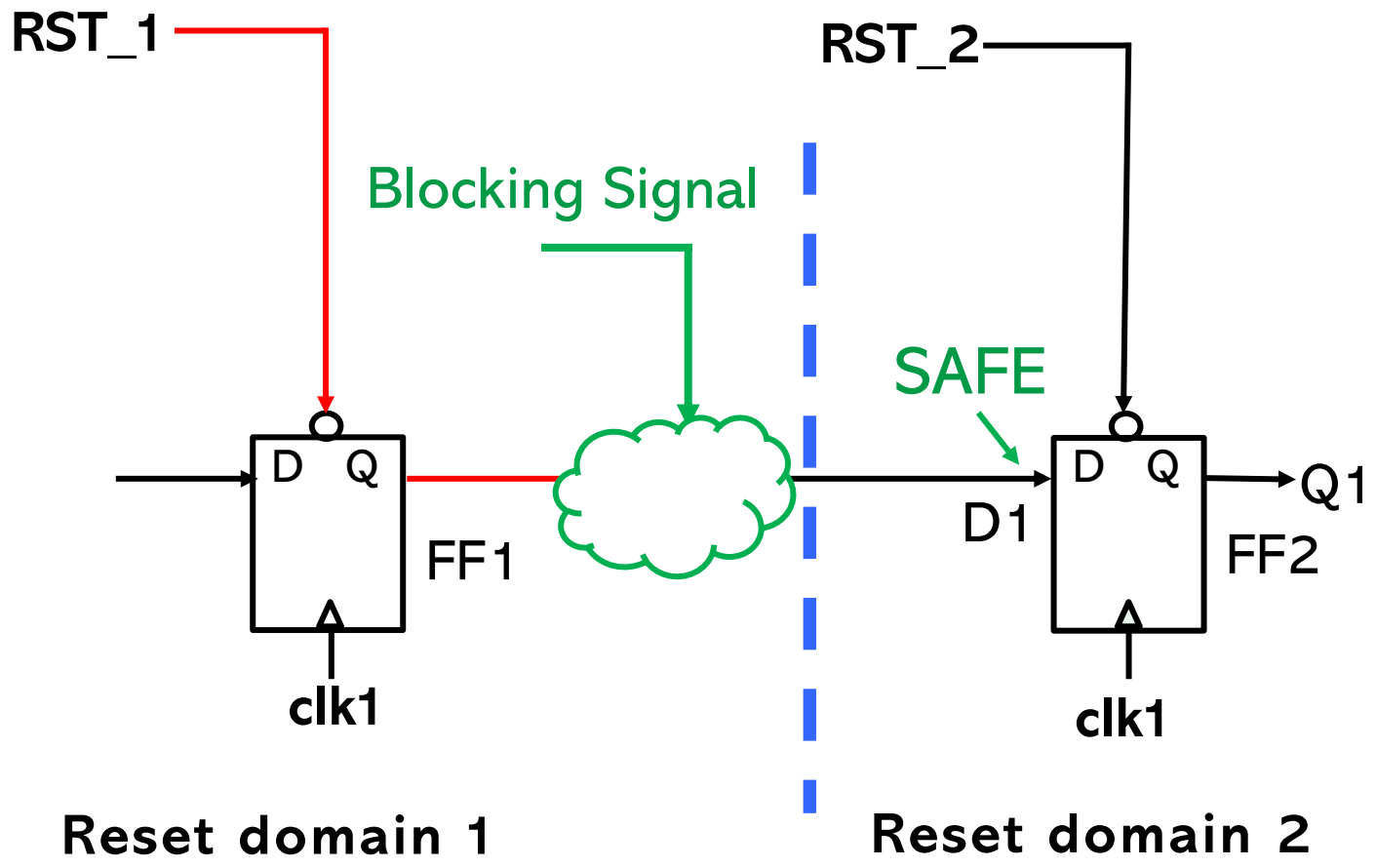
2 RDC sign-off requires global analysis



2 CDC sign-off only needs local analysis

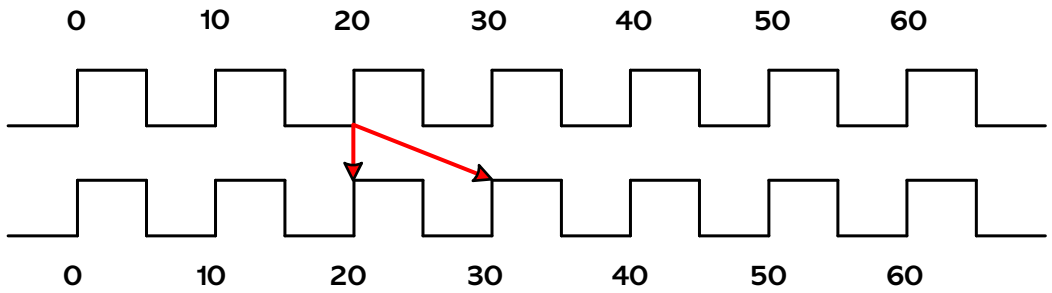


3 RDC-specific analysis needed for sign-off

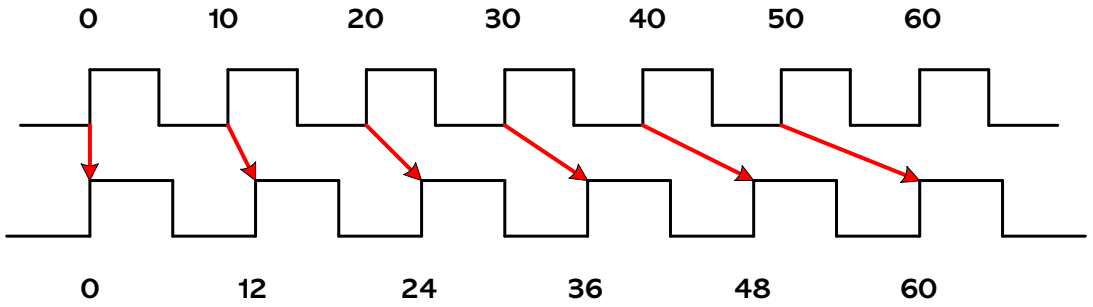


4 RDC errors have lower frequency

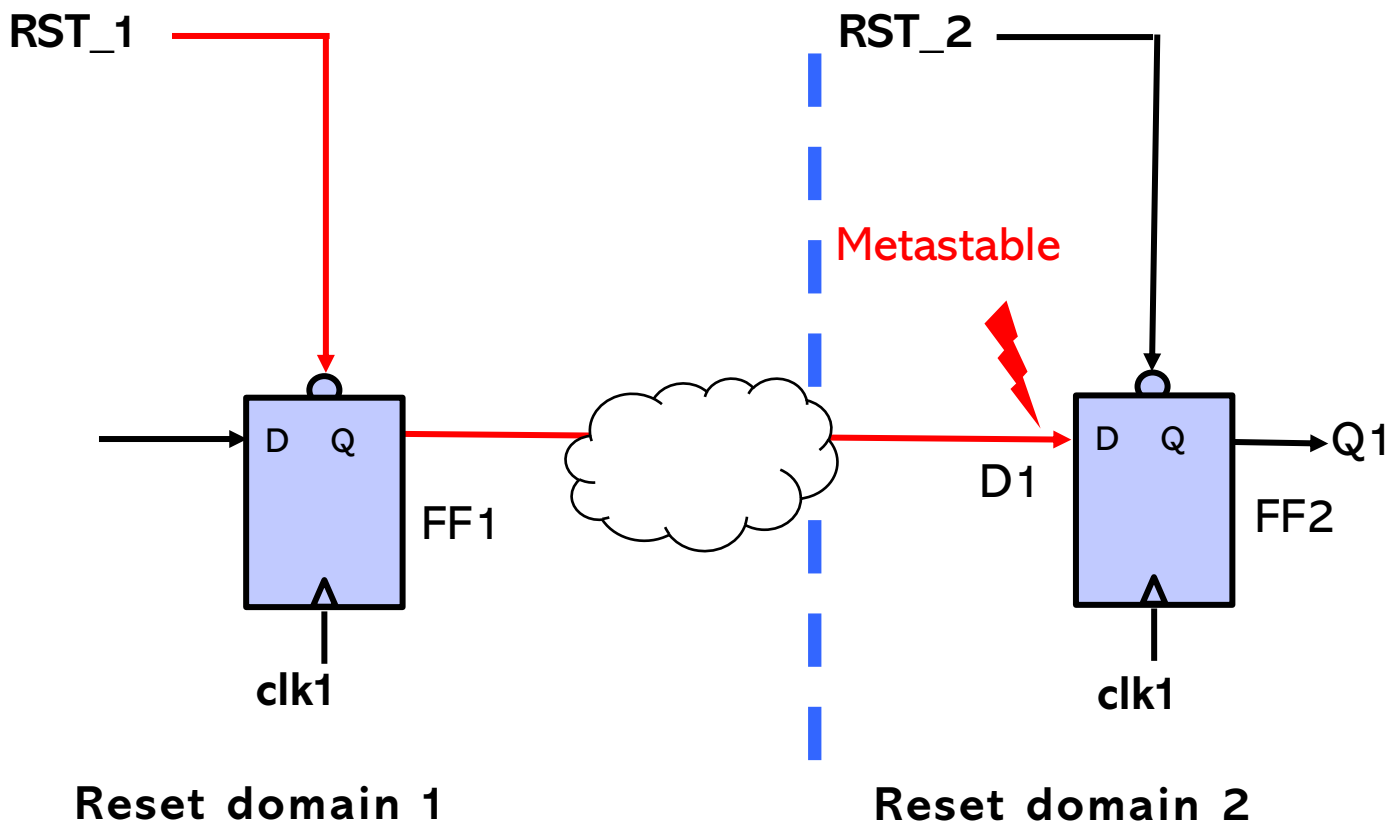
RDC: Source flop change rate:
event dependent



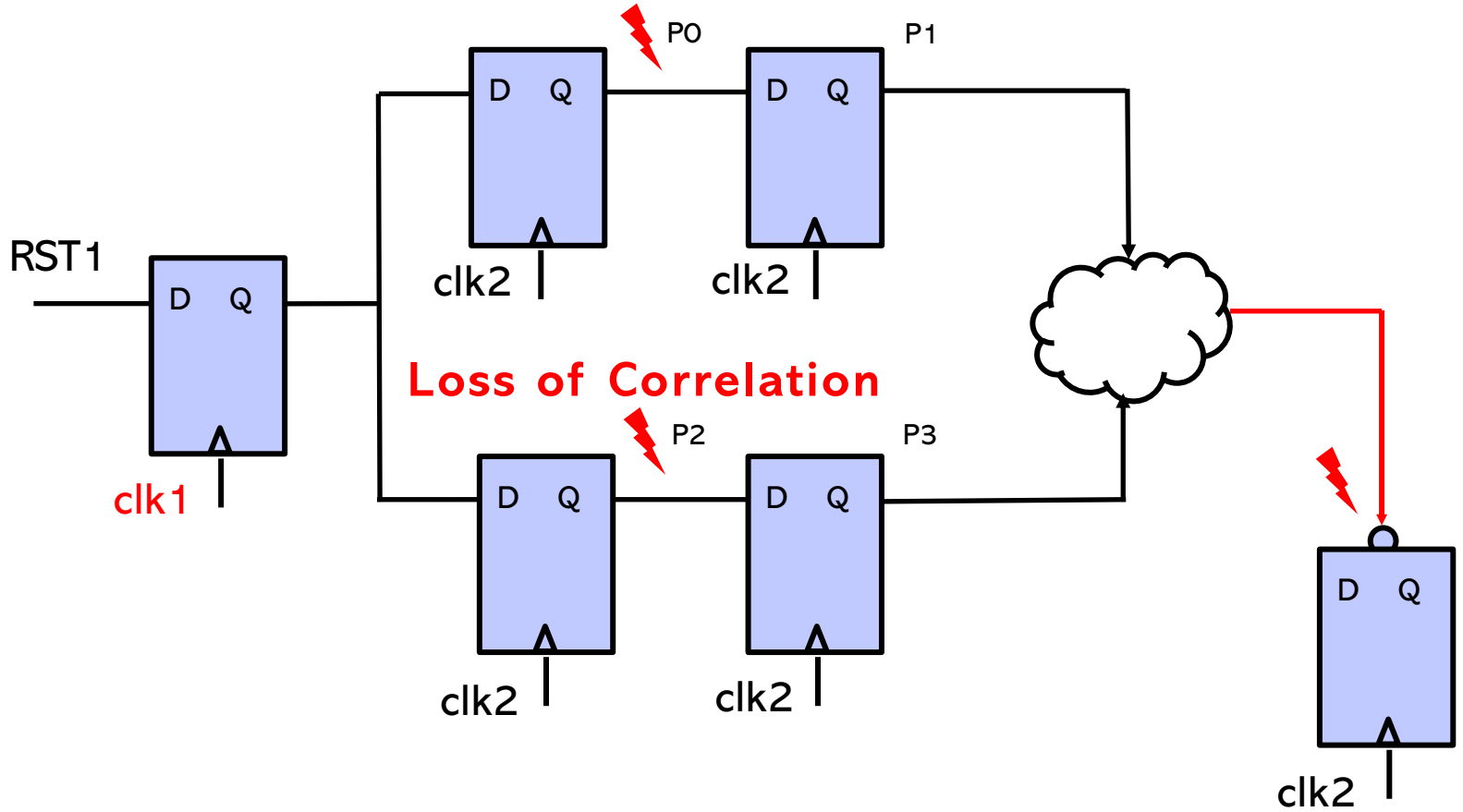
CDC: Source flop change rate:
~ 0.1 GHz



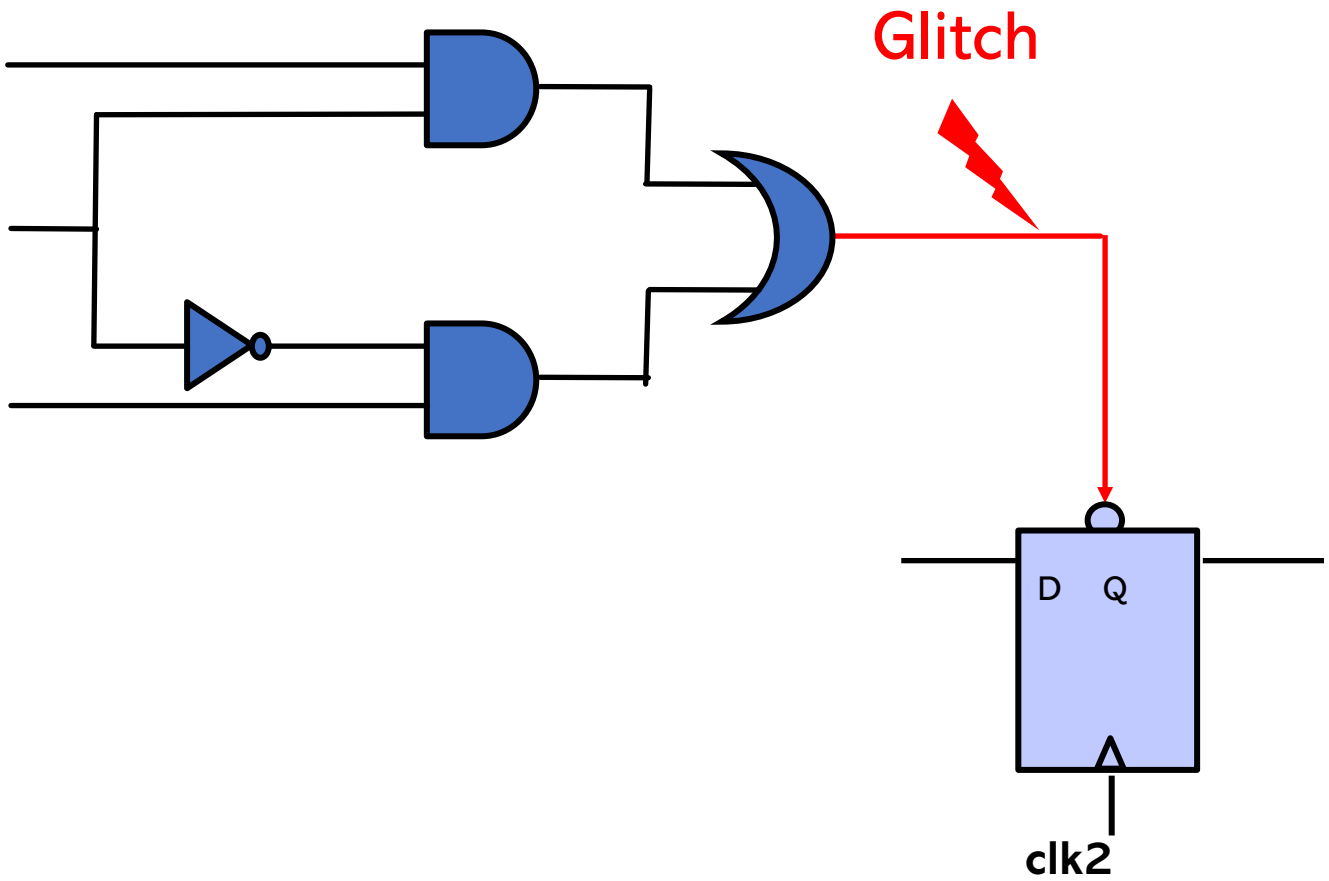
RDC error: Metastability



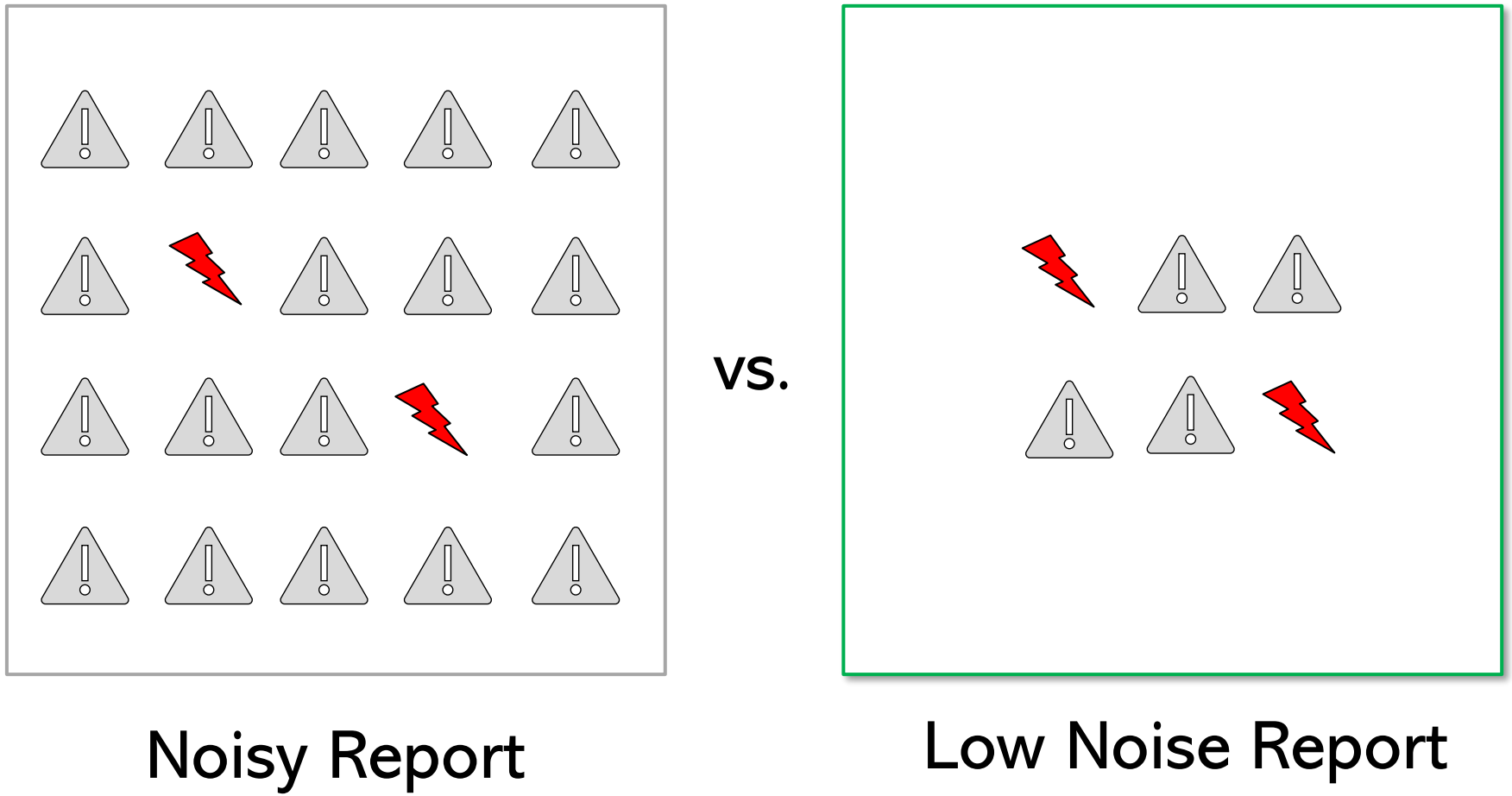
RDC error: Improper functional correlation



RDC error: Glitches



Low-noise violation reports critical

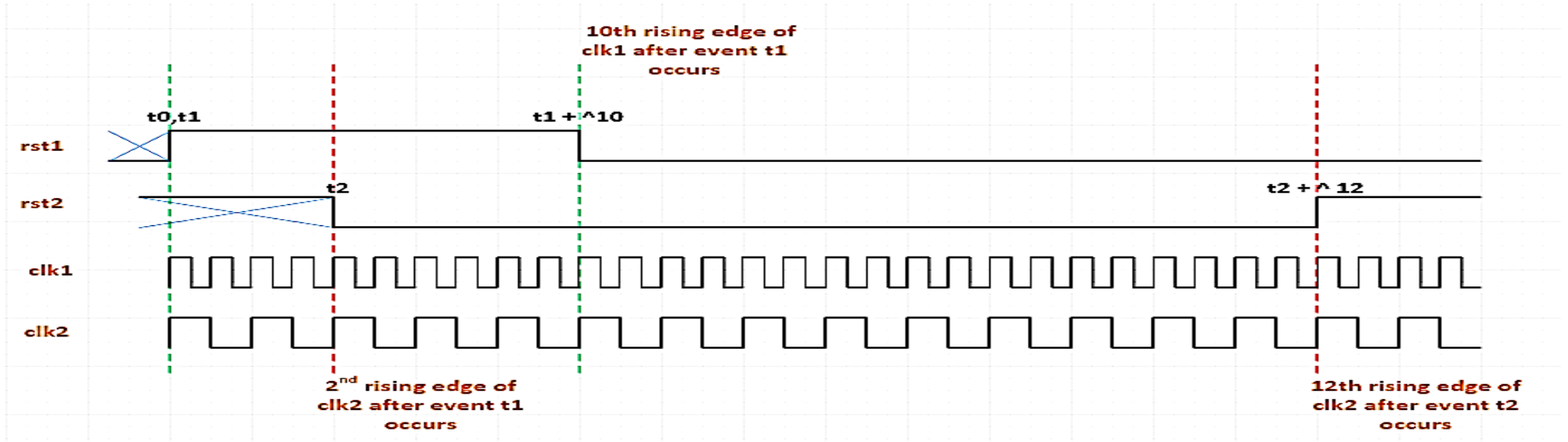


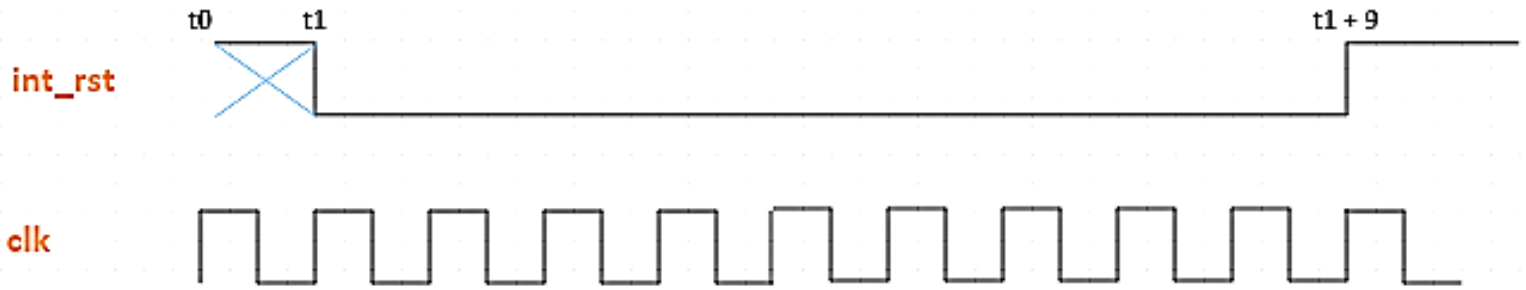
Noisy Report

vs.

Low Noise Report

Reset scenarios reduce noise

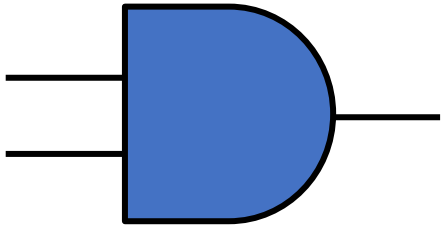




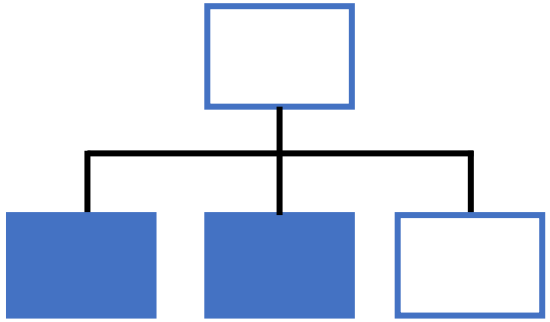
Efficient debug

Advanced RDC sign-off methodology

Low-noise

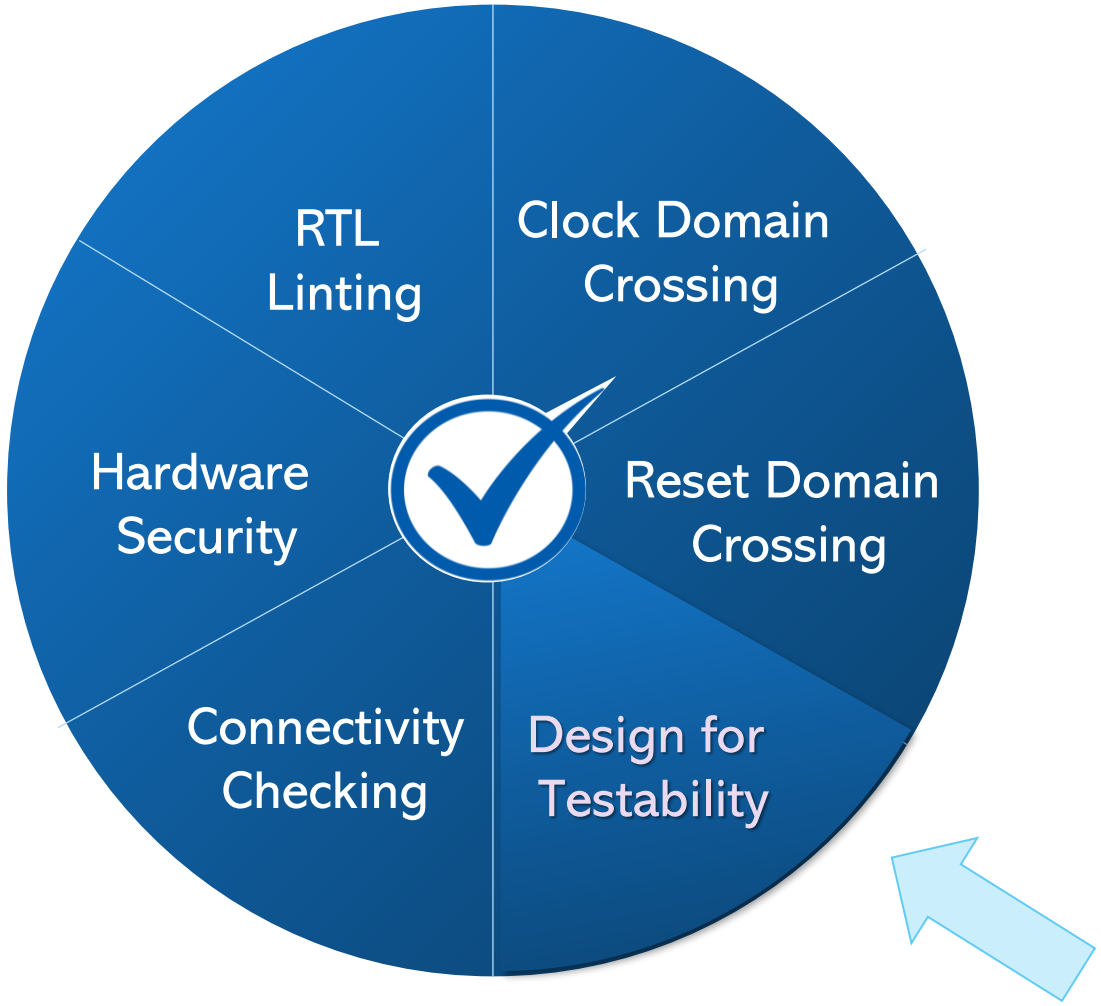


Functional analysis



Hierarchical flow

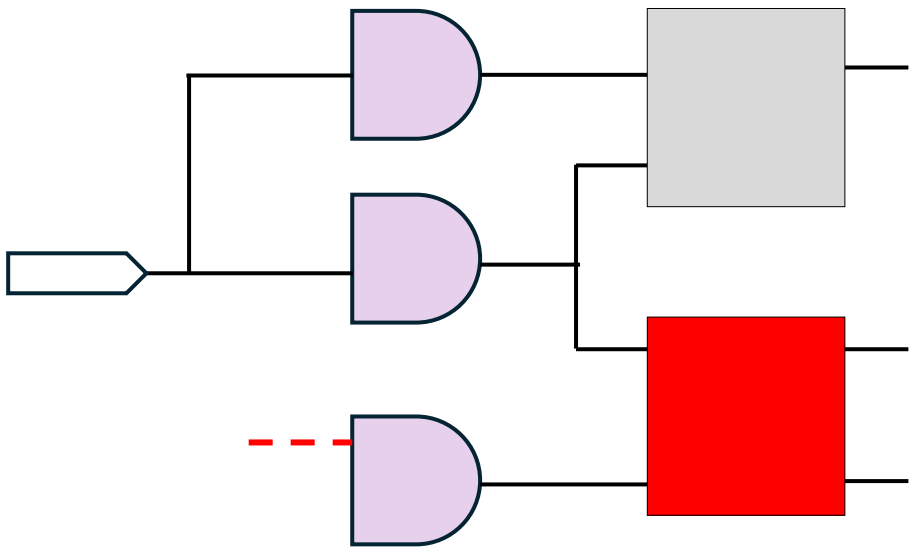
DFT Sign-Off



Advanced DFT static sign-off

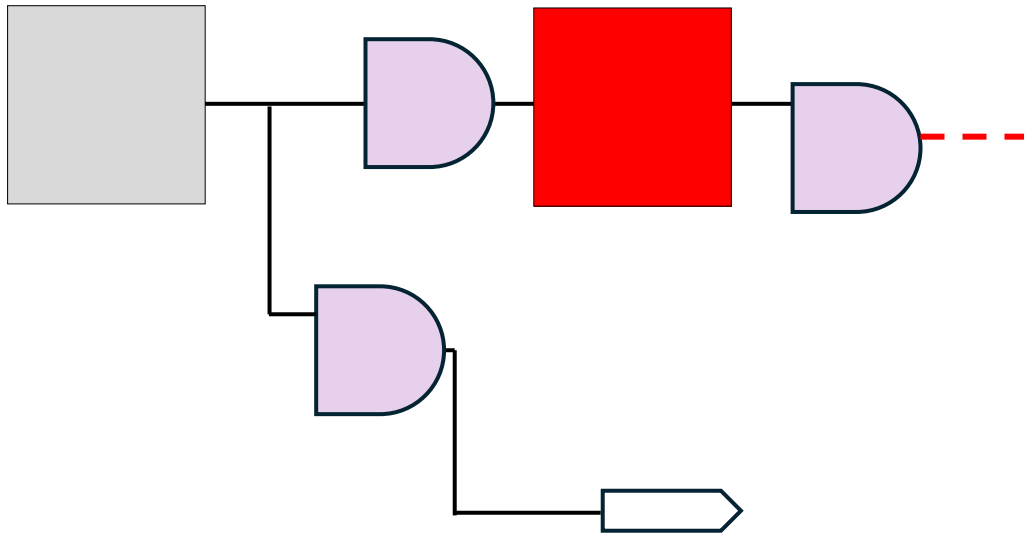
Controllability

Can you drive it?

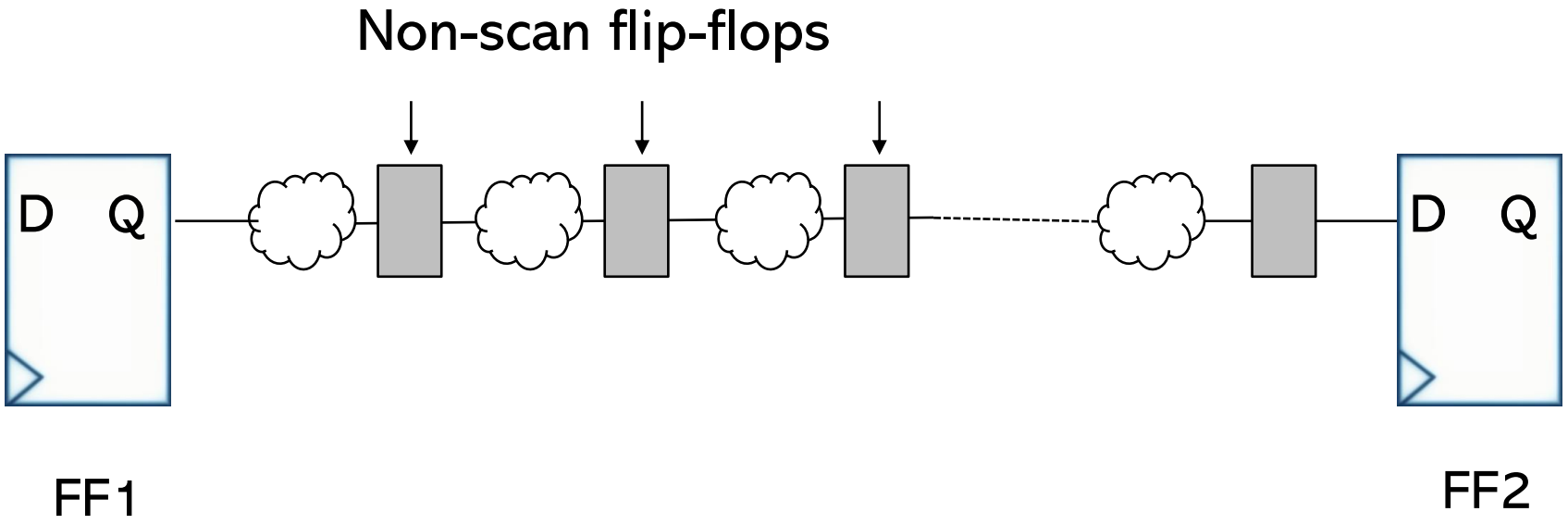


Observability

Can you see it?

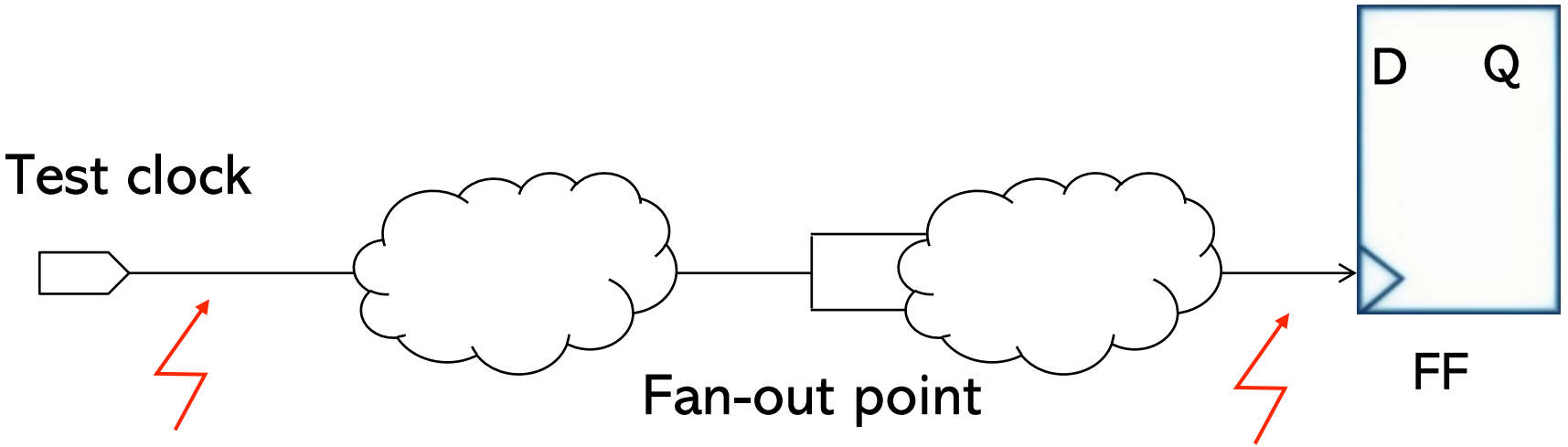


1 Use comprehensive rulesets



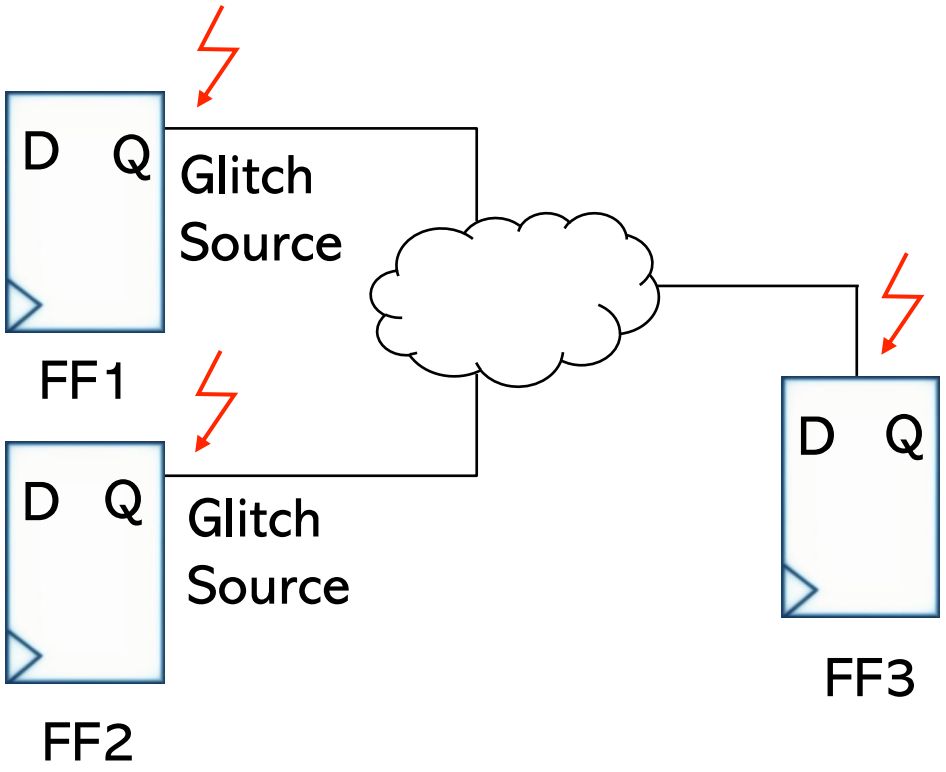
RULE: Sequential capture depth through non-scan flip-flops should not exceed user-specified limit

1 Use comprehensive rulesets

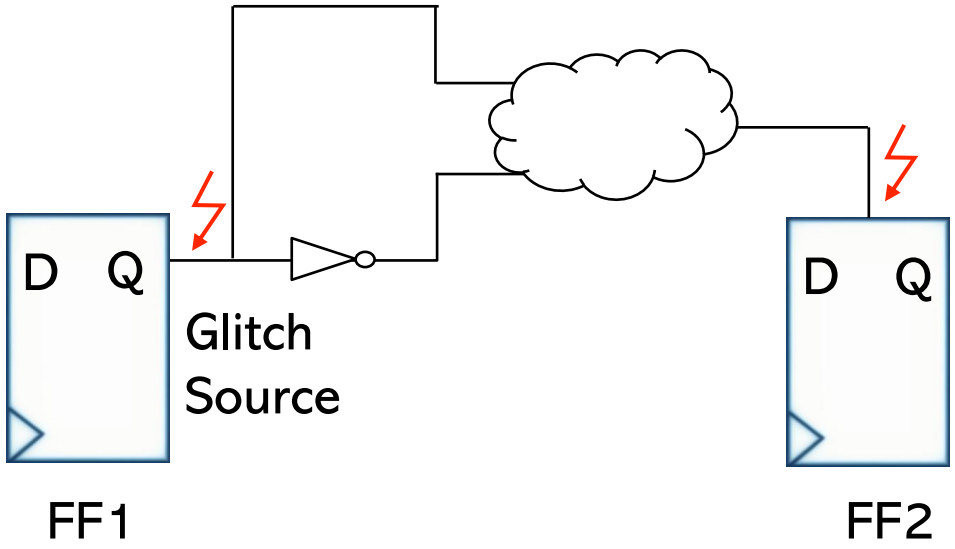


RULE: A test clock should not fan out and reconverge with itself

2 Utilize fine-grained rules

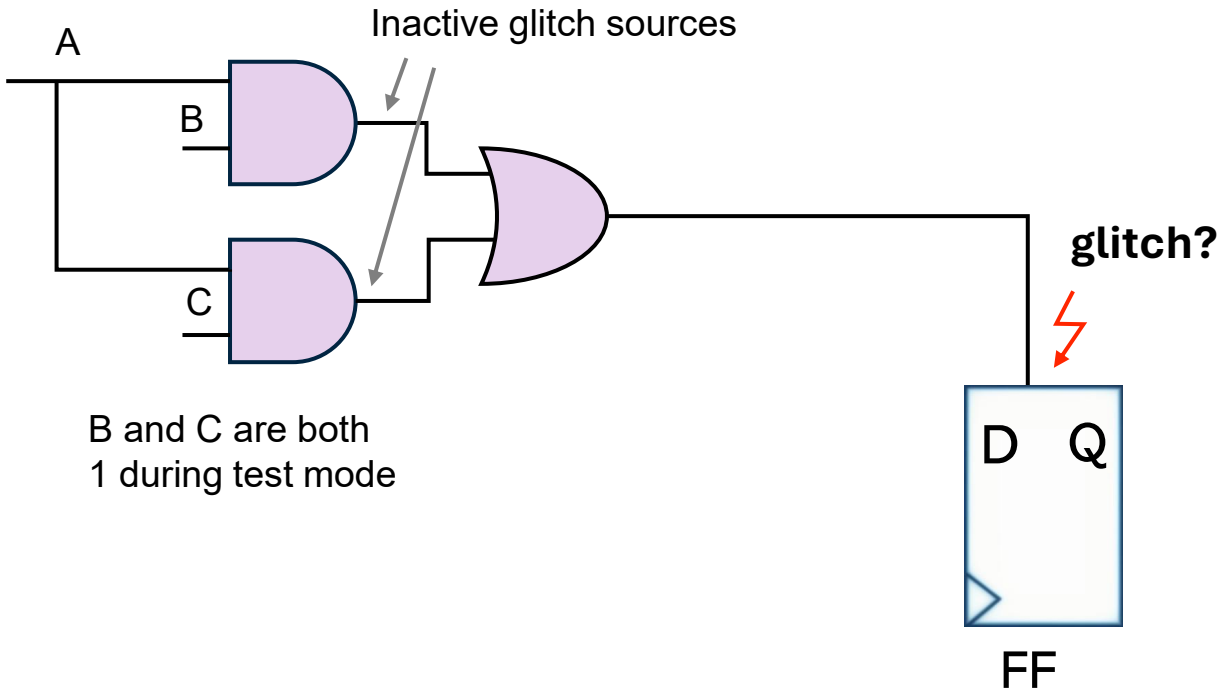


RULE 1: Asynchronous reset convergence glitch



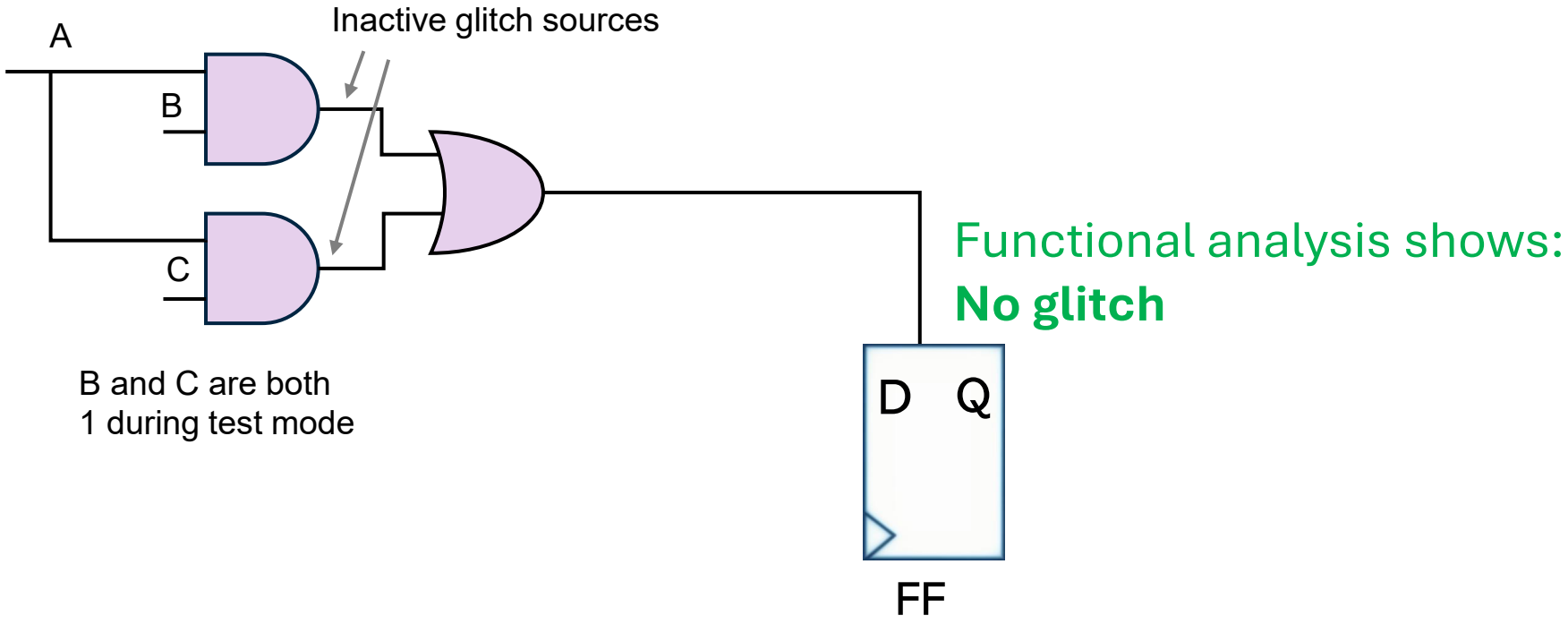
RULE 2: Asynchronous reset reconvergence glitch

2 Utilize fine-grained rules



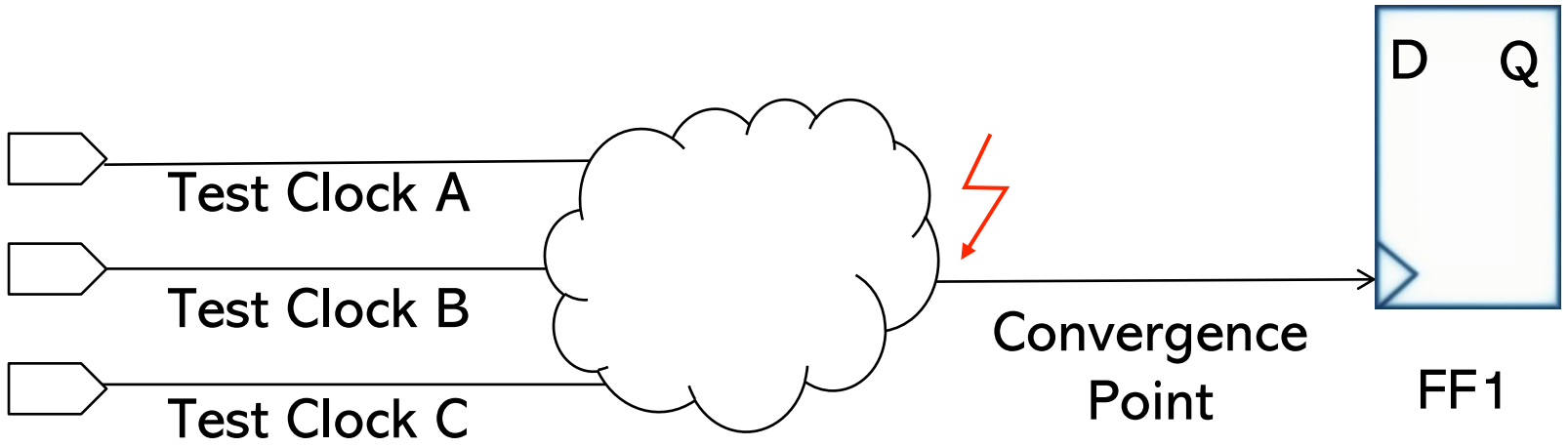
RULE: Asynchronous reset convergence glitch

2 Utilize fine-grained rules



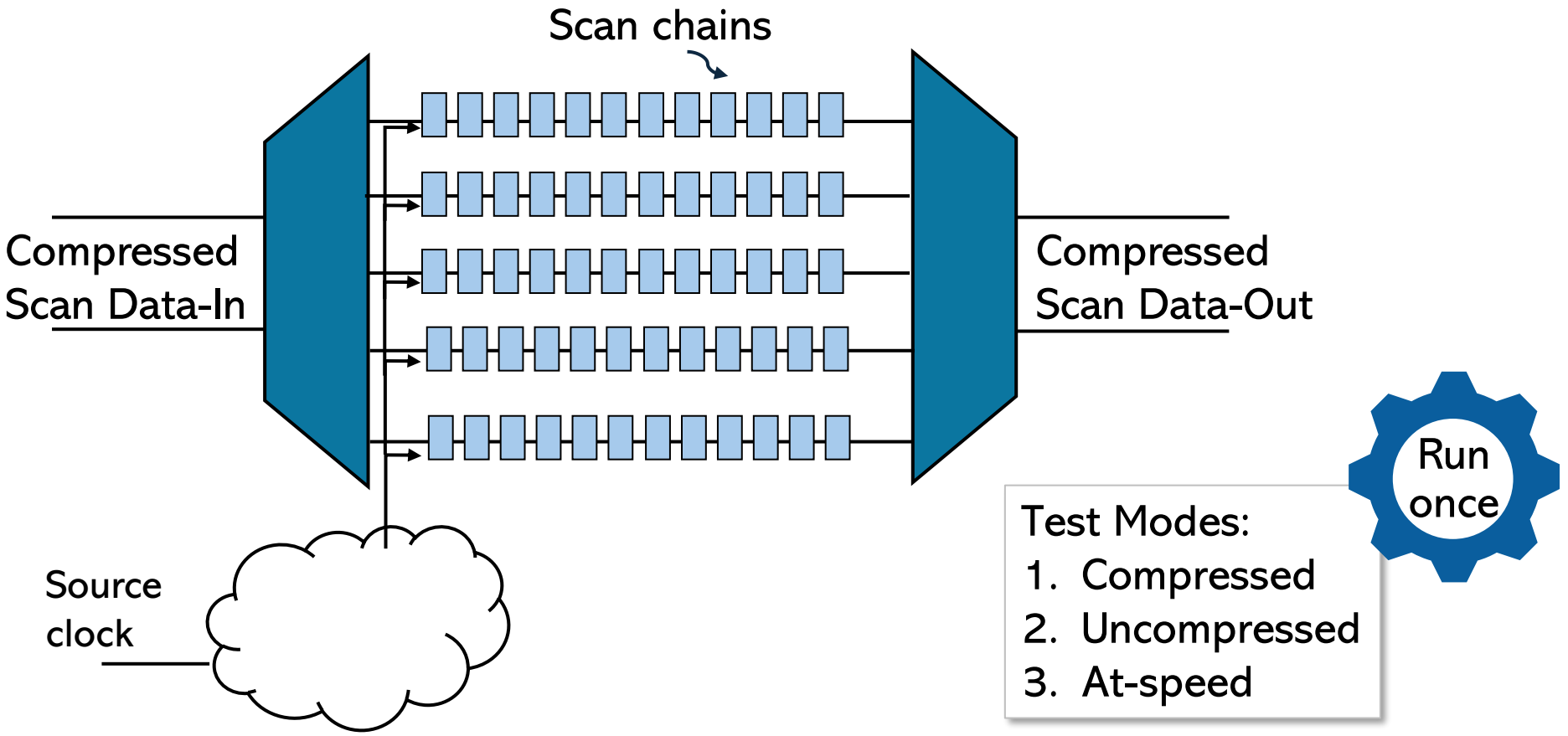
RULE: Asynchronous reset convergence glitch

2 Utilize fine-grained rules



RULE: Two or more test clocks should not converge while driving a flip-flop's clock input

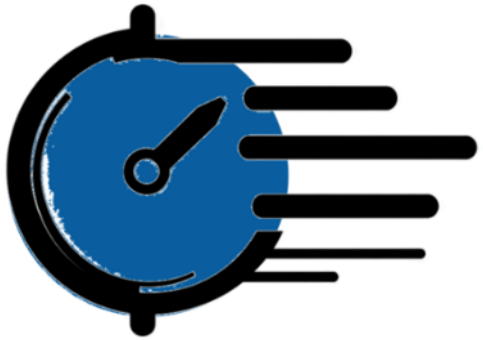
3 Analyze all test modes in one run



3 Analyze all test modes in one run

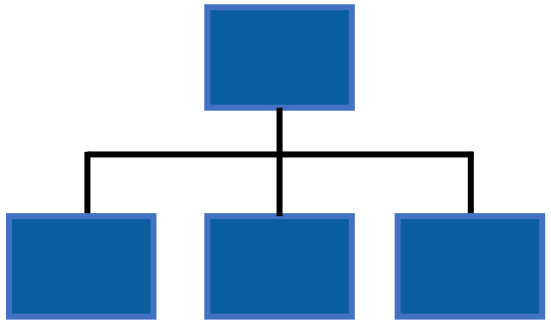
One set-up

hours vs weeks



Fast runtime

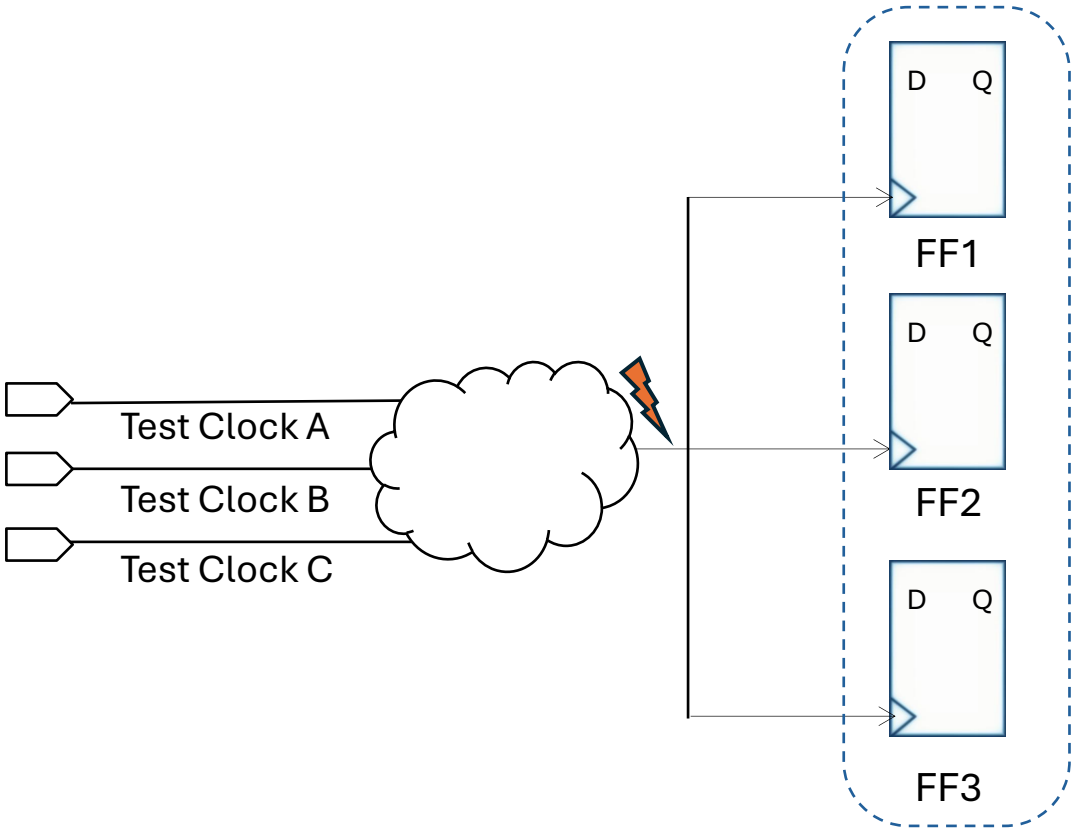
2 min/M gates/mode



Consolidated

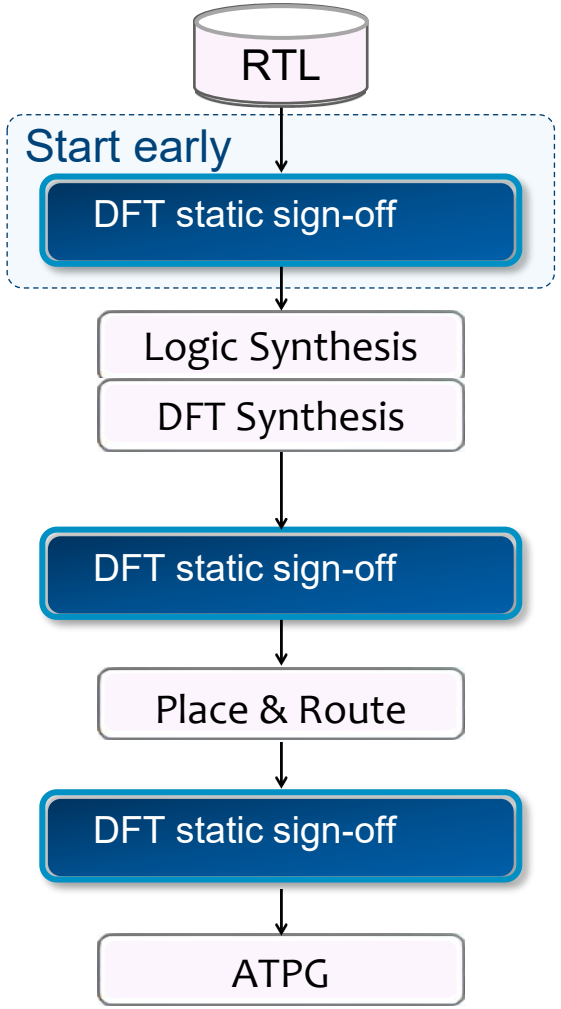
hierarchical reports

4 Group errors by root cause

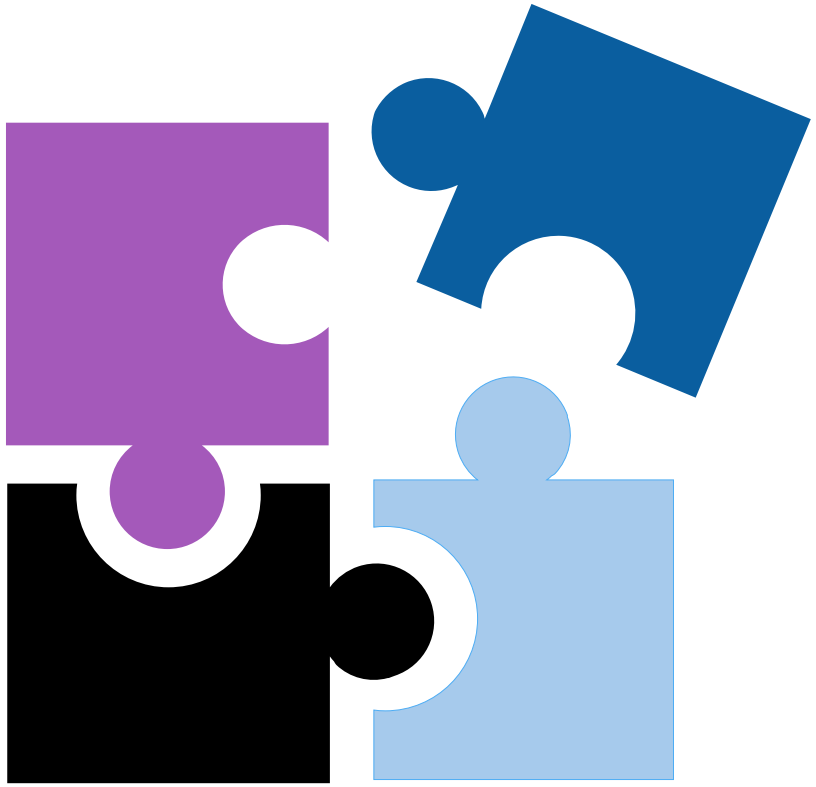


RULE: No clock glitch can reach destination flip-flops through convergence point

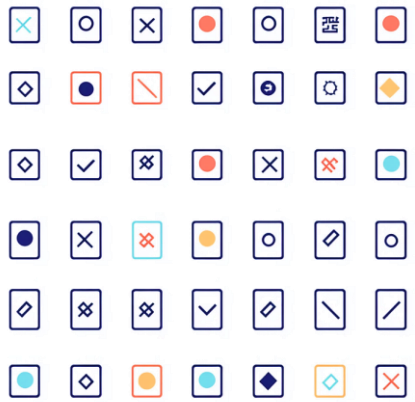
5 Deploy at all design stages



6 Leverage multi-vendor DFT flows

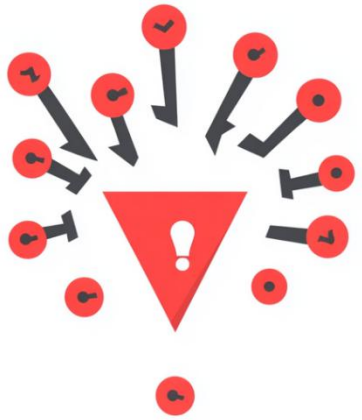


Advanced DFT sign-off



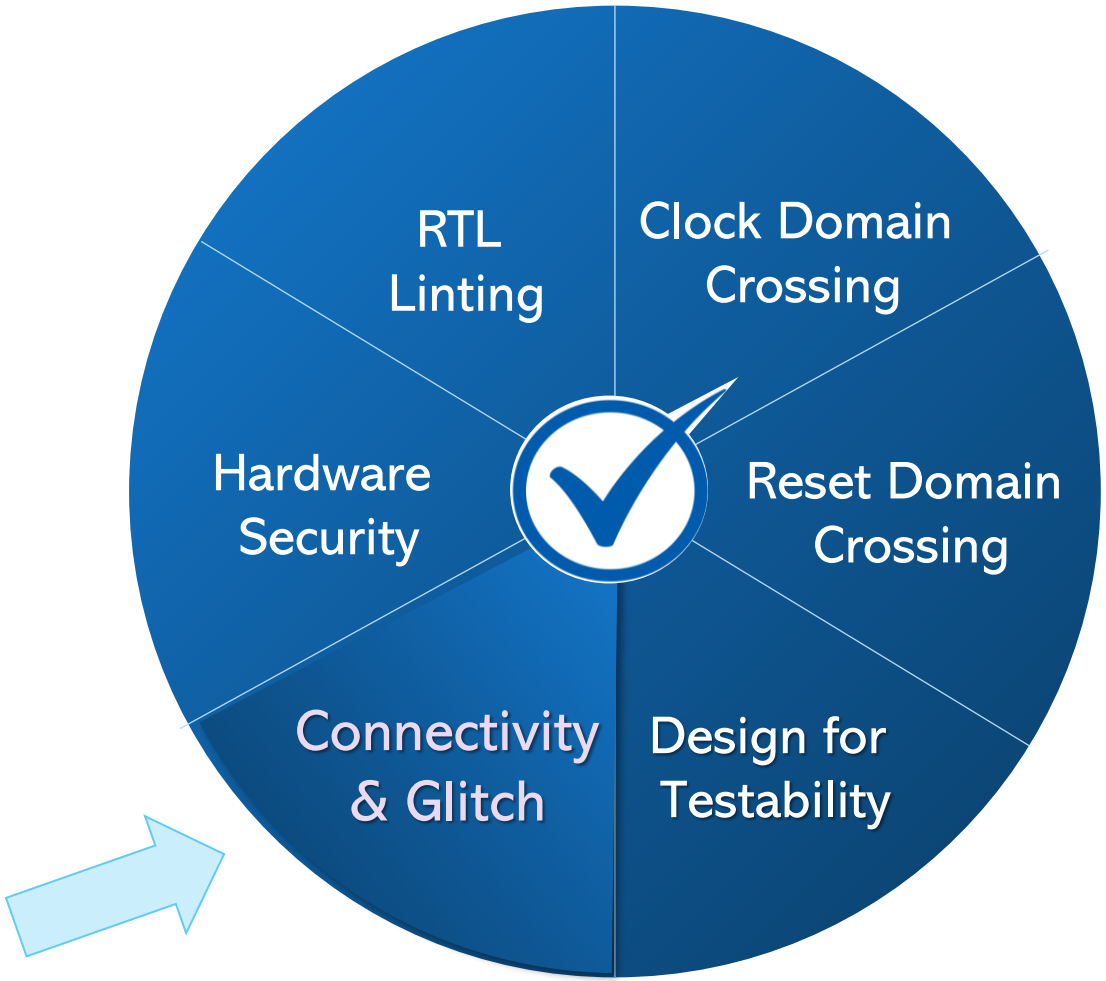
Comprehensive, fine-grained rules

Multimode tests

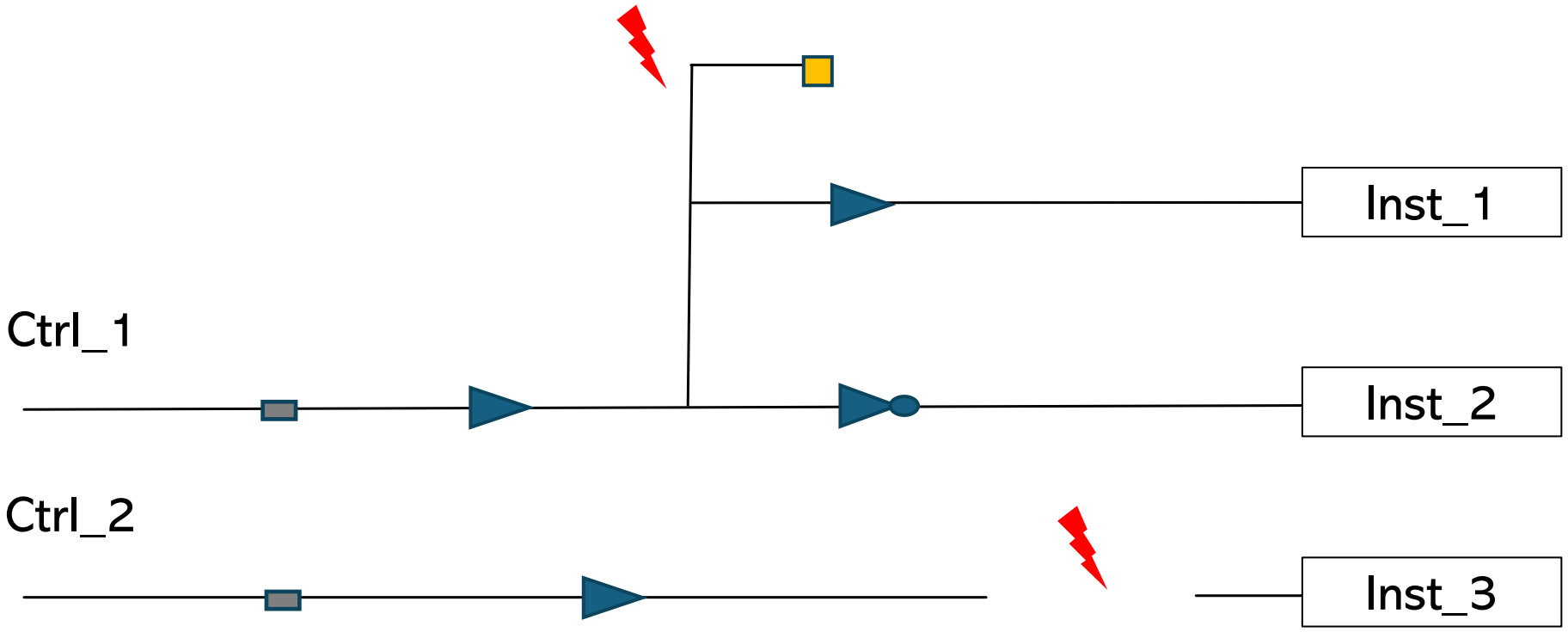


Root cause grouping

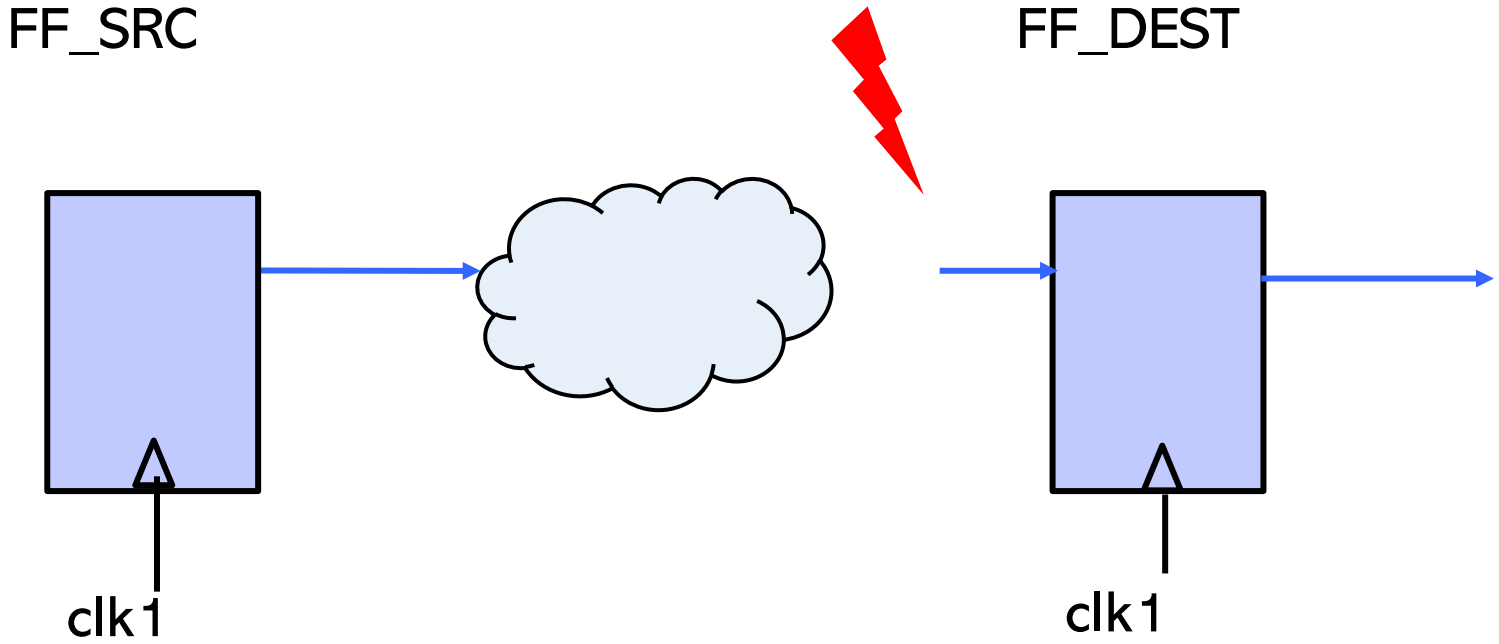
Connectivity Sign-Off



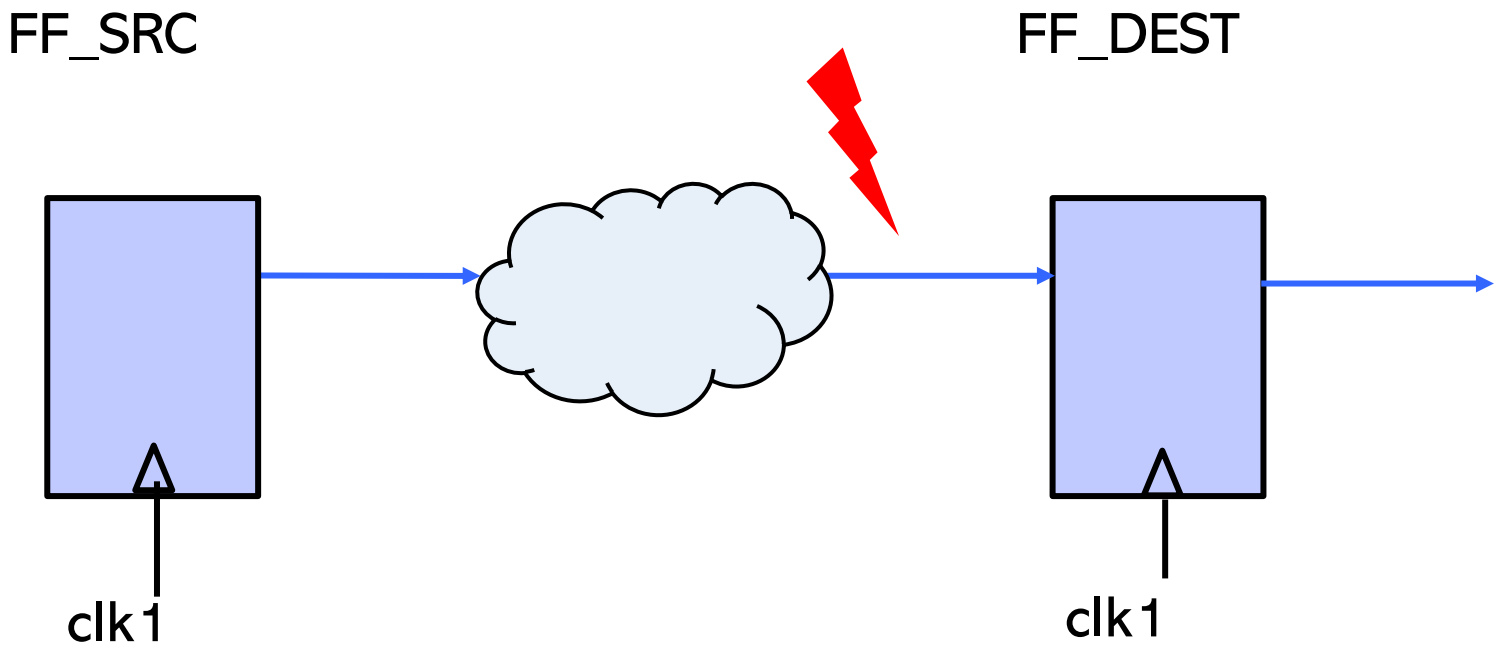
Connectivity static sign-off



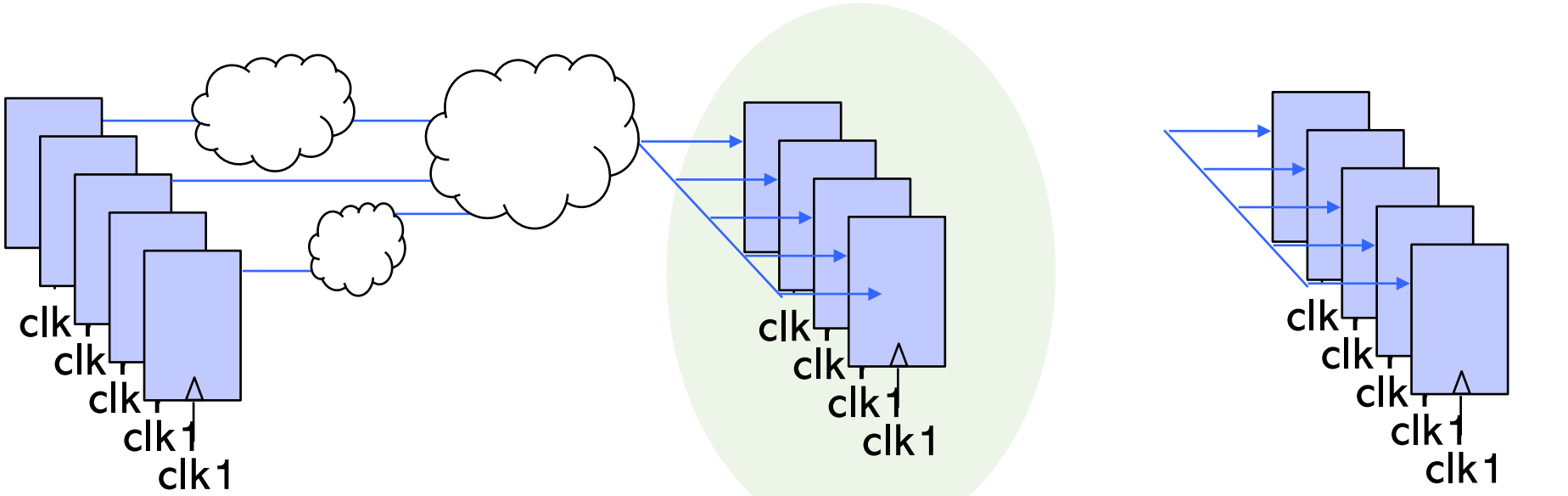
1 Efficient rule definition



1 Efficient rule definition



1 Efficient rule creation, tiered checking

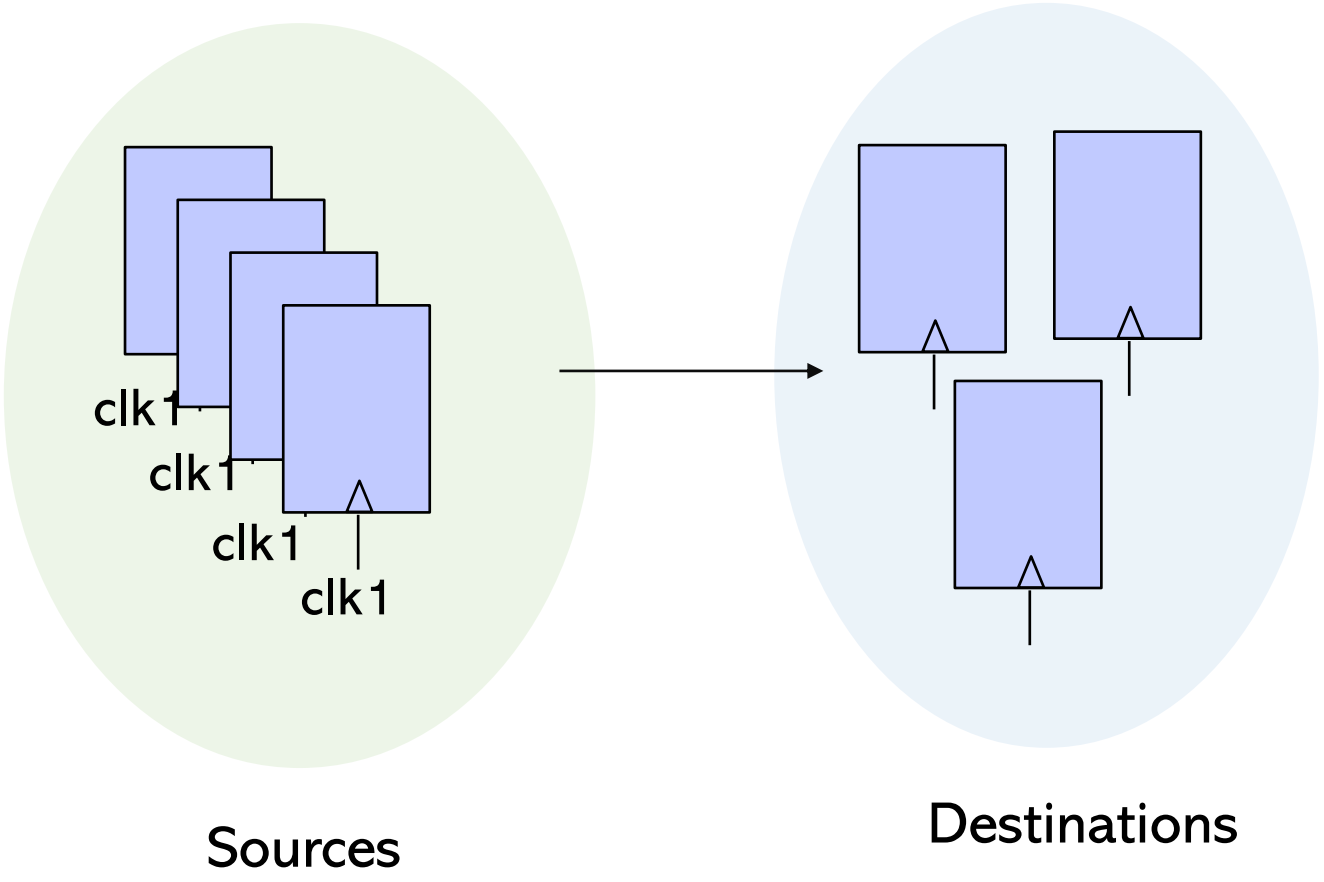


Sources

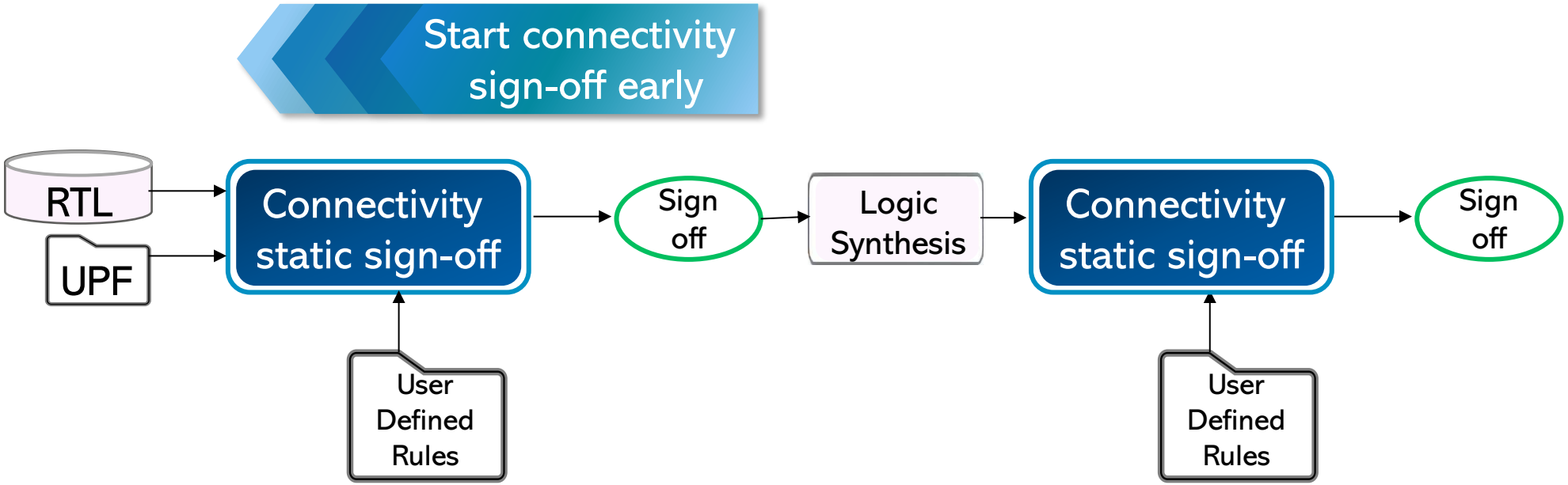
Connected Destinations

Unconnected Destinations

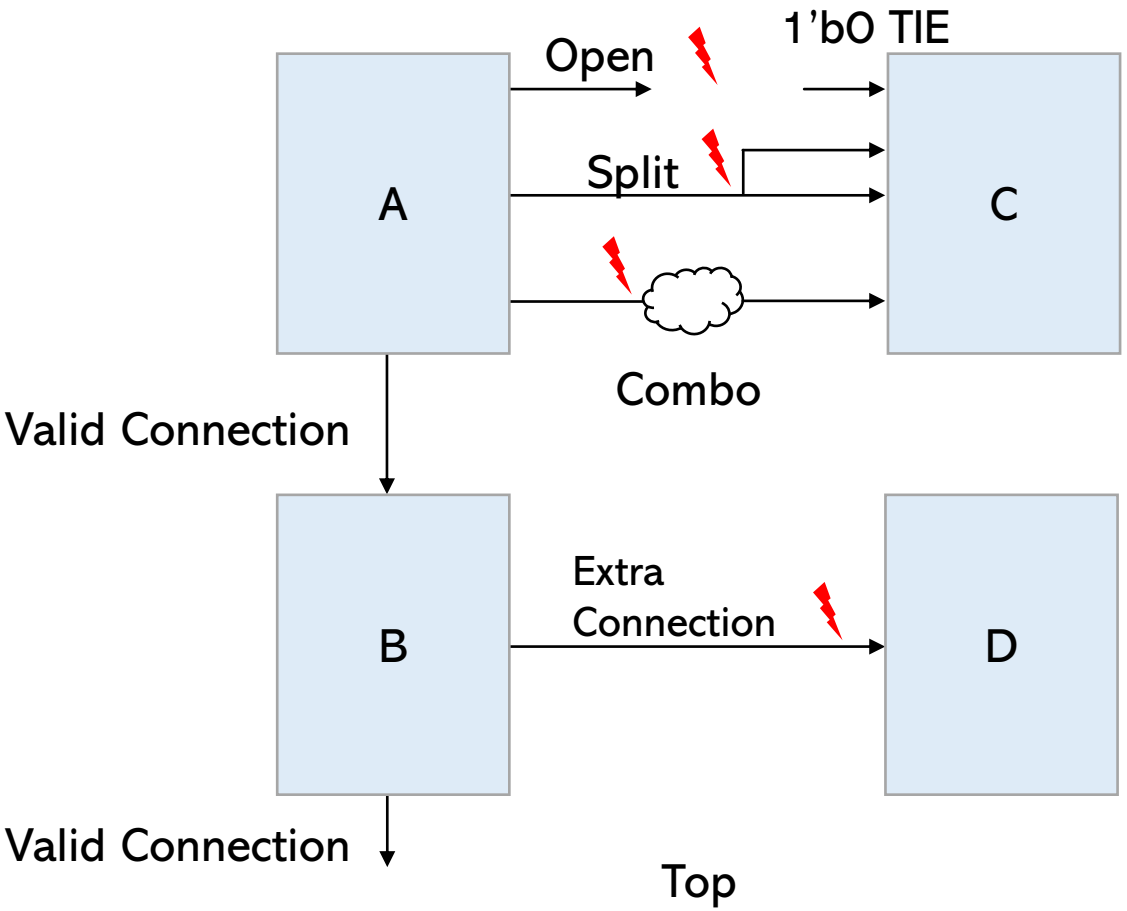
1 Efficient rule creation, tiered checking



2 Start connectivity sign-off early

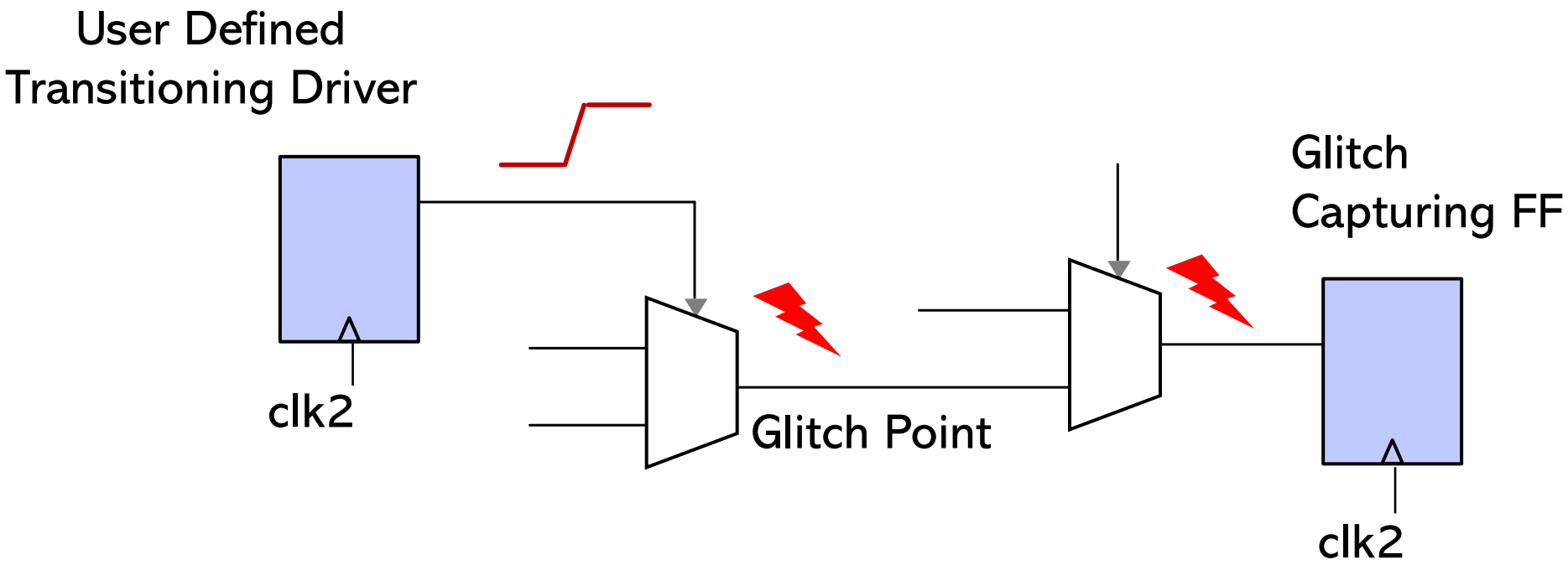


2 Check abutment connectivity early

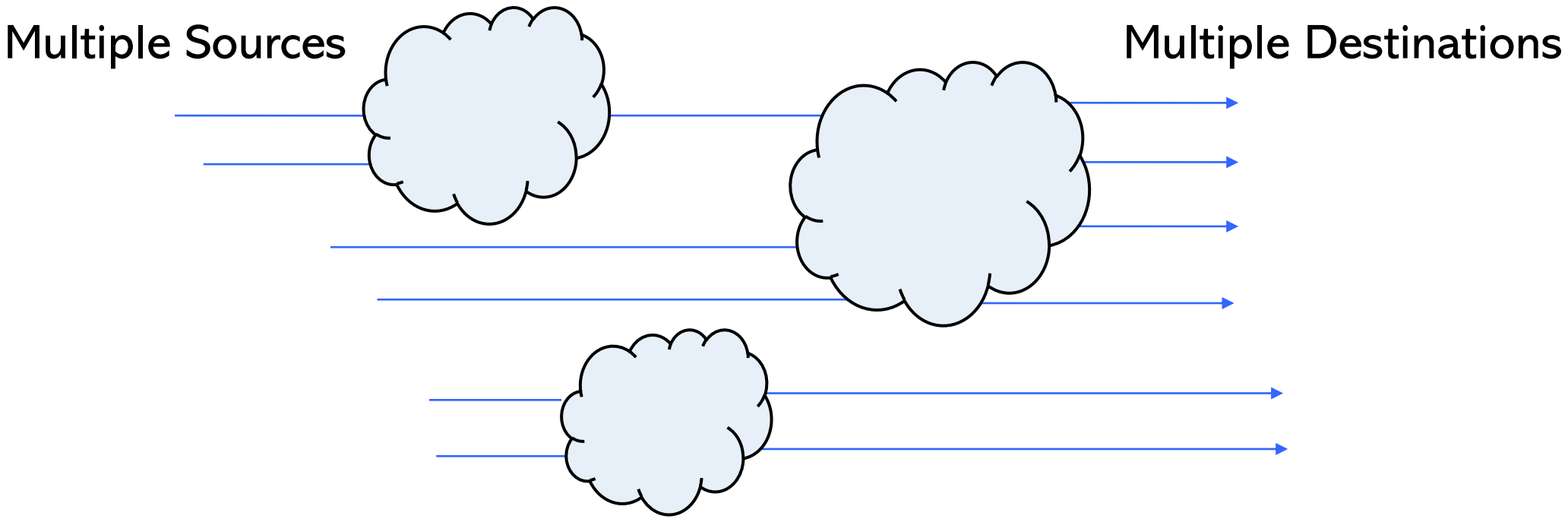


Allowed Arcs =
{ A B } { A C } { * TOP }

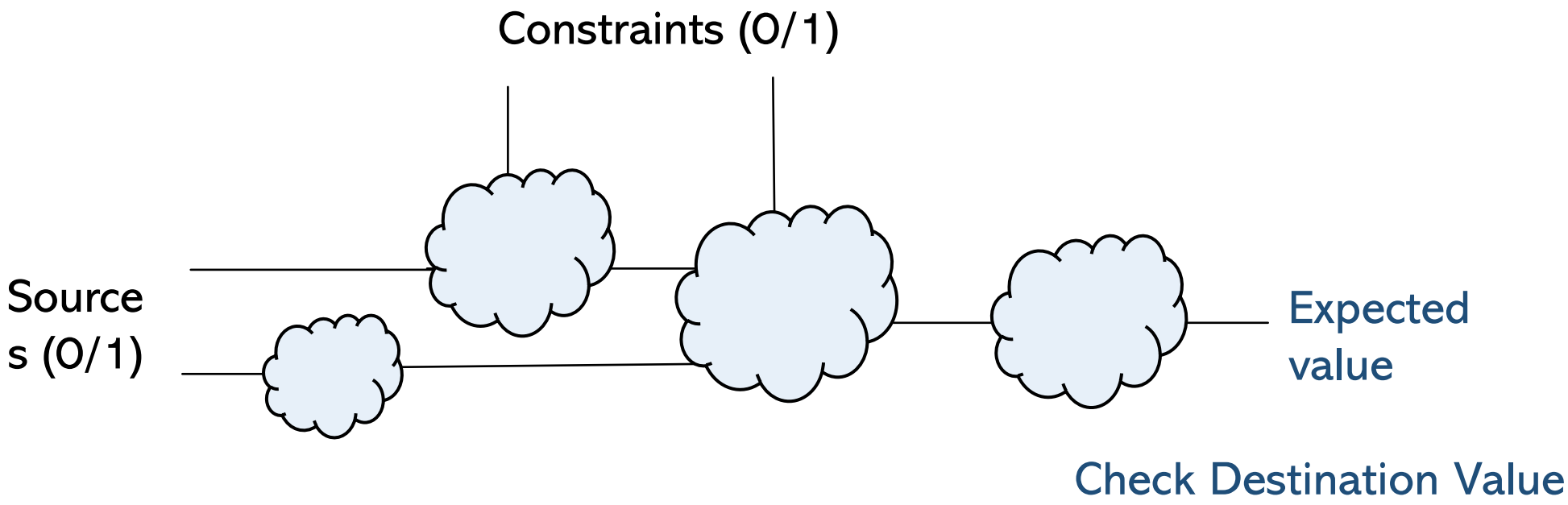
3 Glitch checking



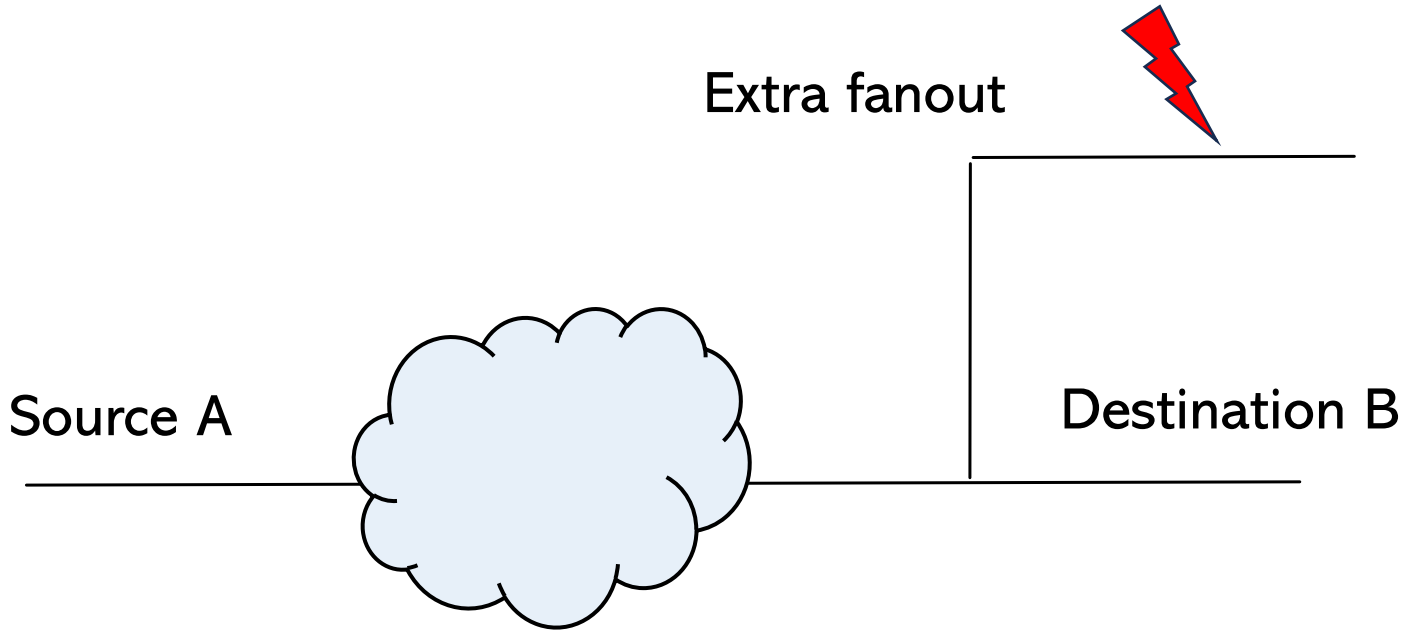
Static sign-off vs. Formal



Static sign-off vs. Simulation

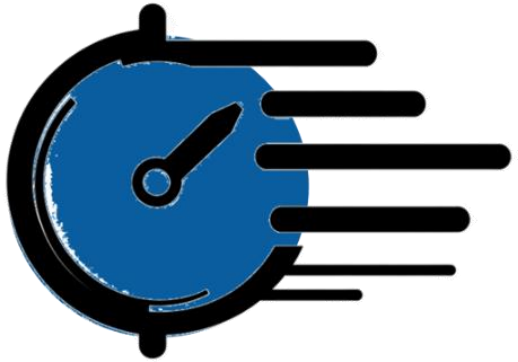


Static sign-off vs. Simulation



Advanced connectivity sign-off

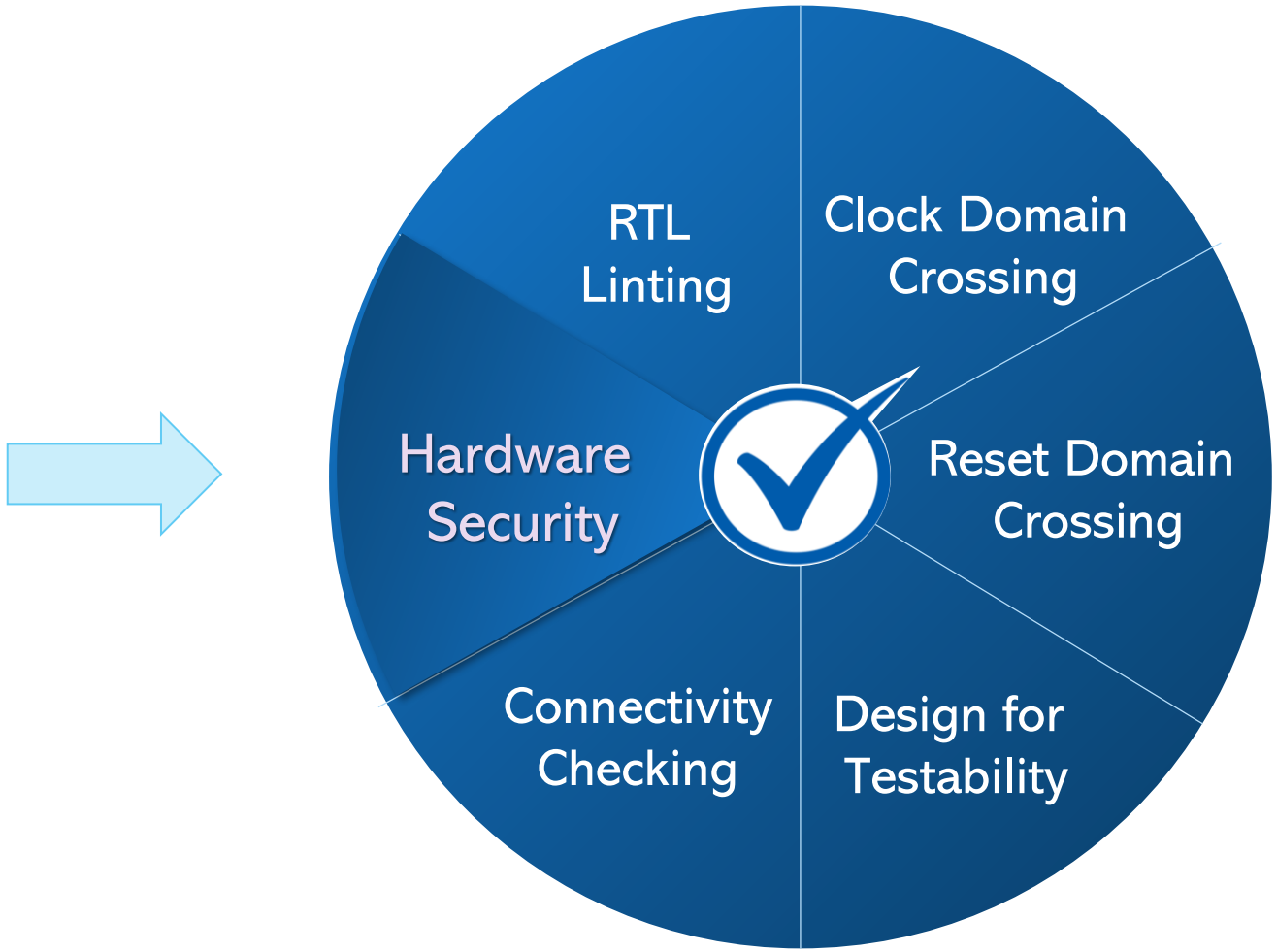
Efficient rule definition



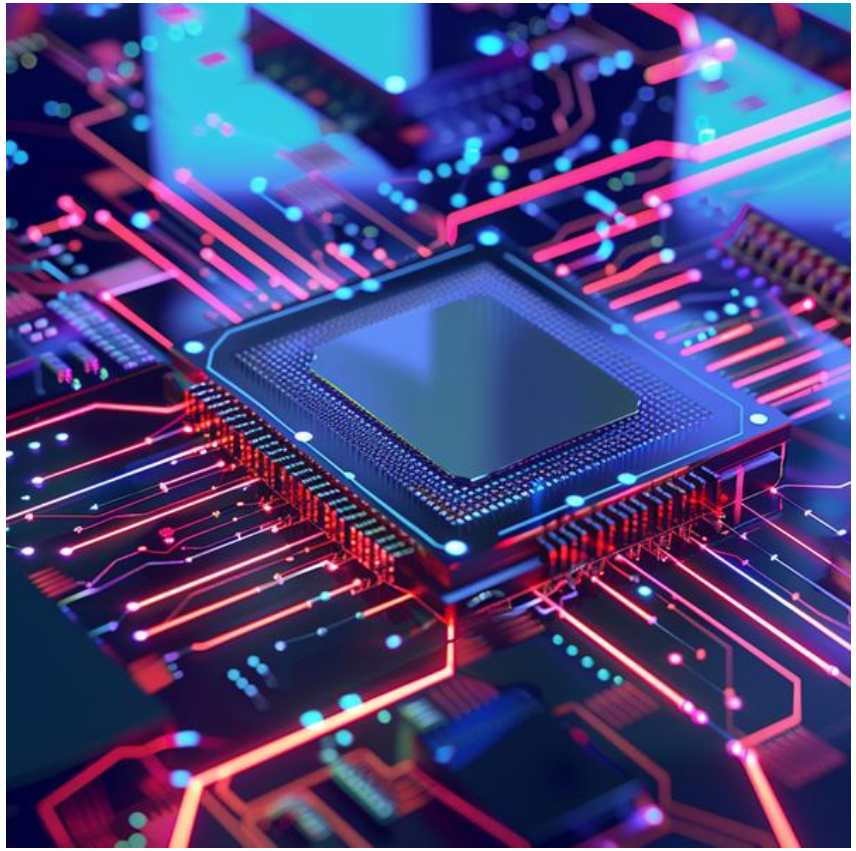
Speed for RTL sign-off

Create groups

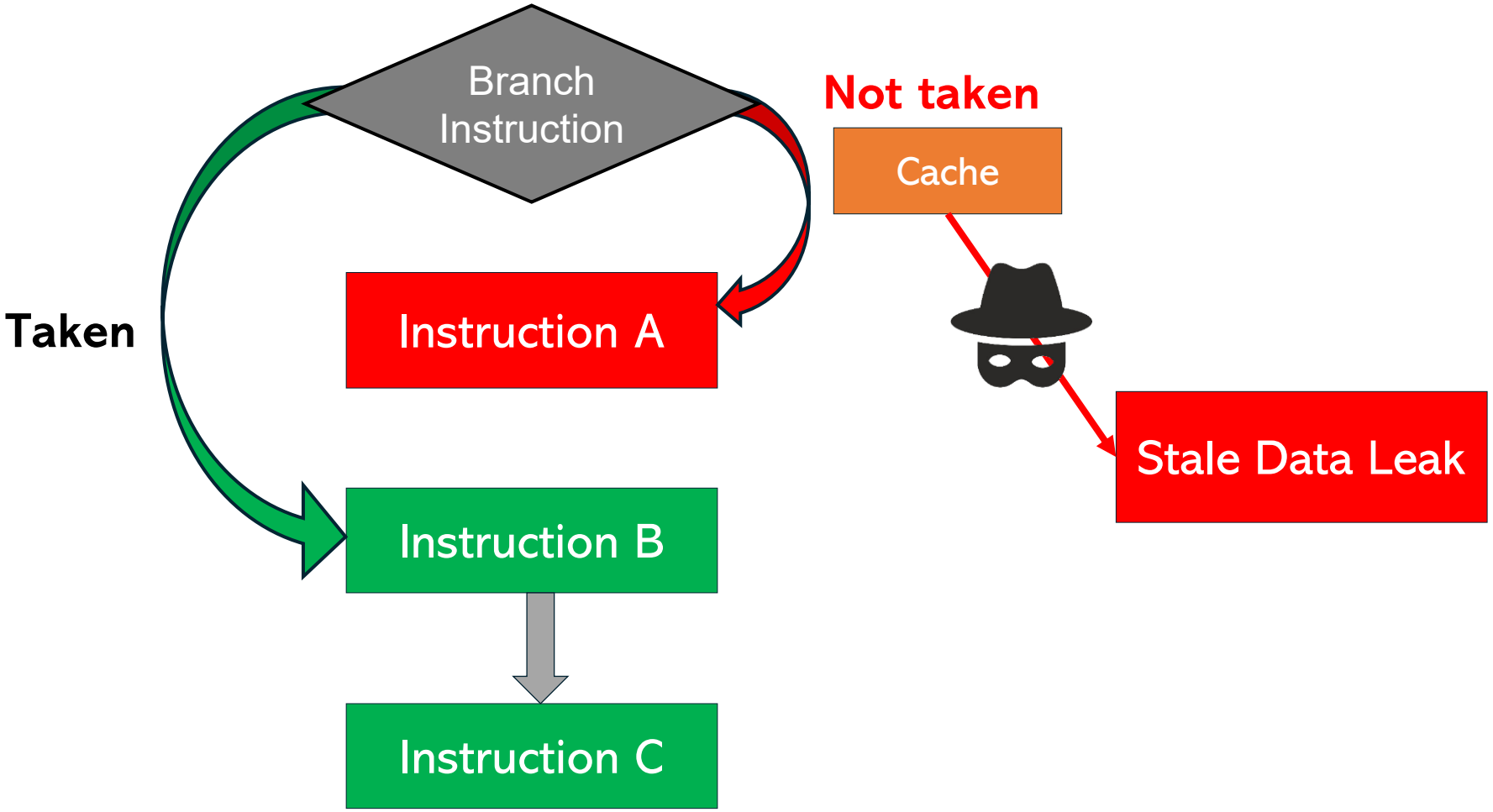
Hardware Security Sign-Off



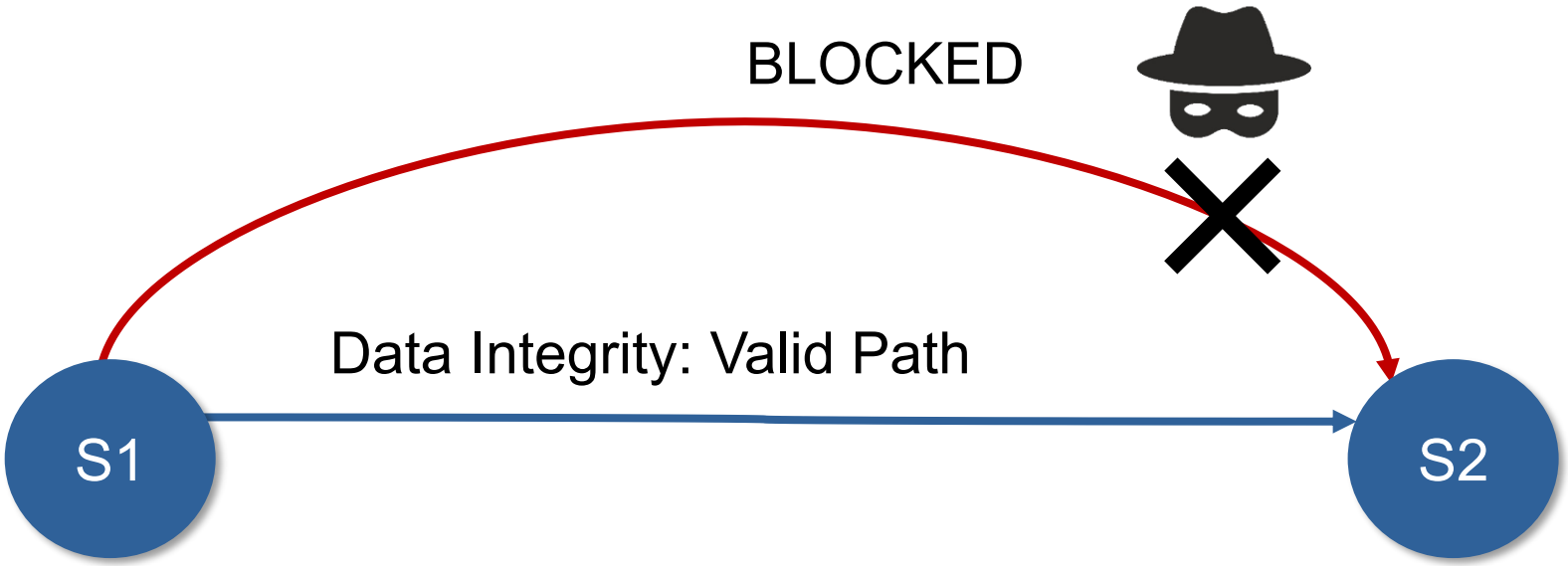
Eliminate HW security vulnerabilities



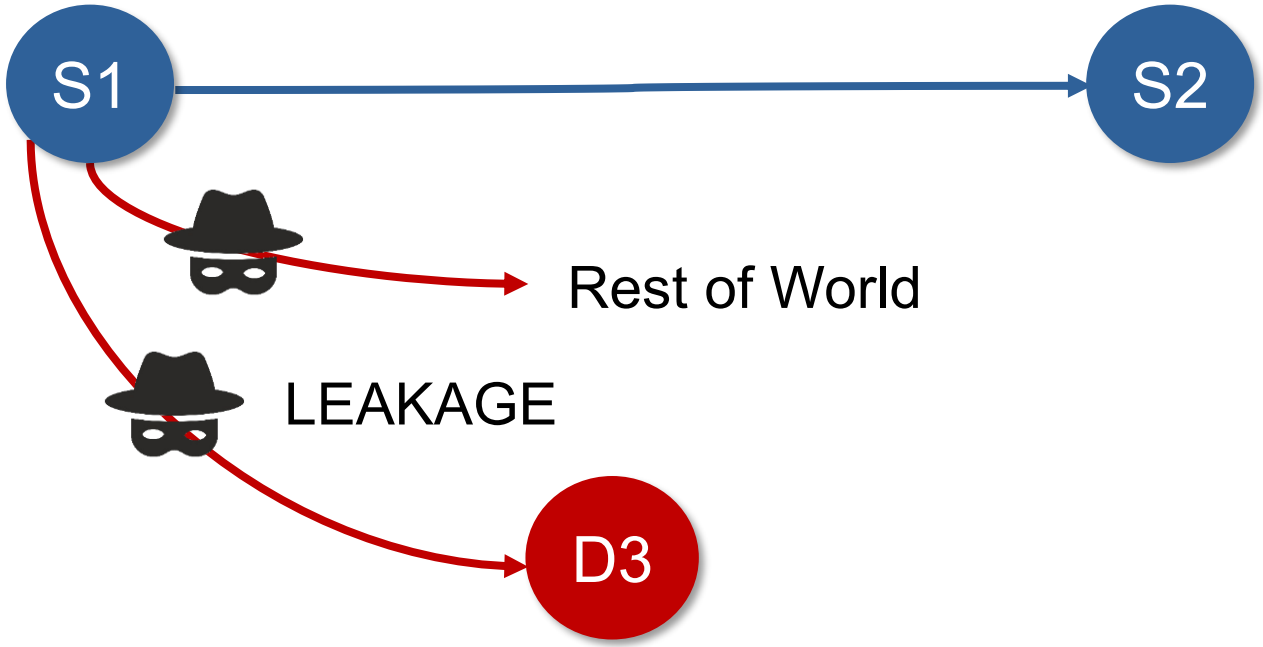
Spectre stale data leak



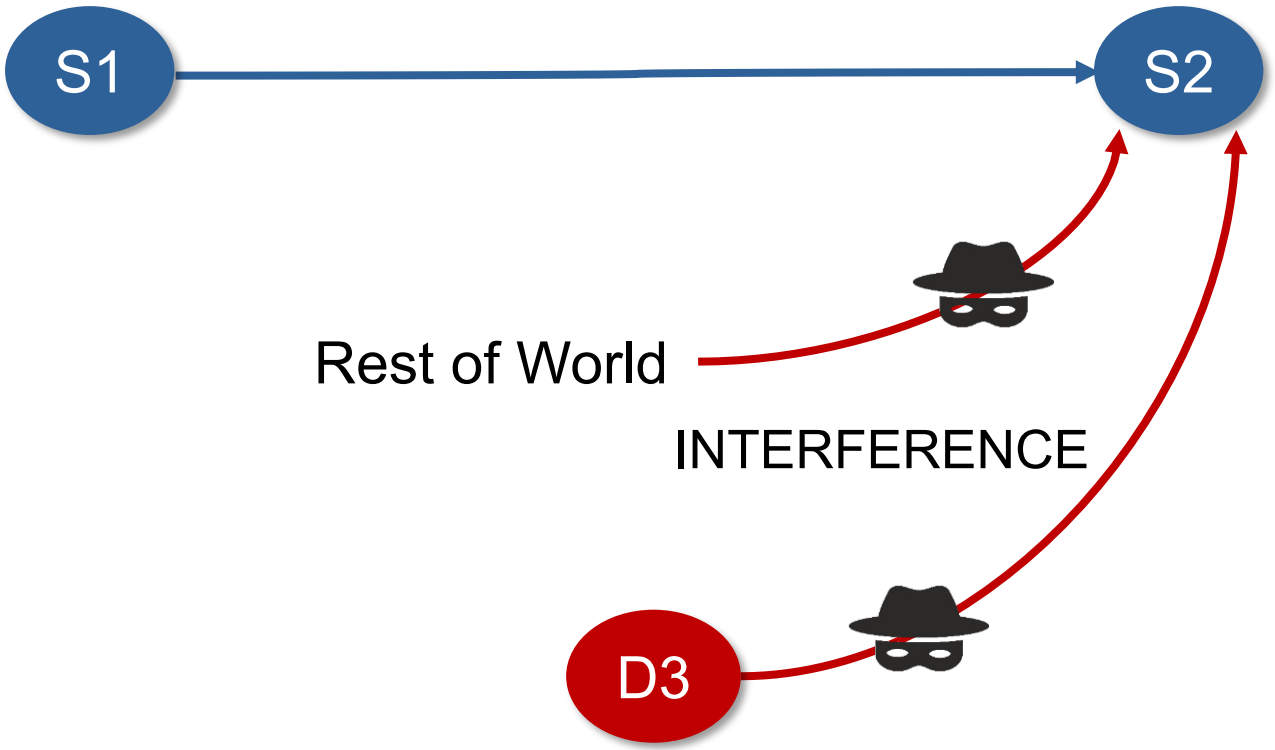
1 Data integrity



2 Leakage prevention



3 Interference safeguarding



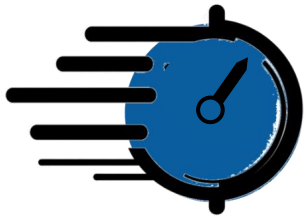
User-defined + general rules

Static sign-off vs. Formal verification

Static Sign-off

Exhaustive Verification

Formal

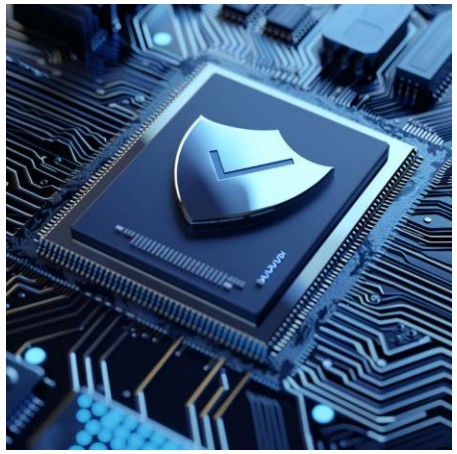


1 M gates/mins

100M gates/2 hrs

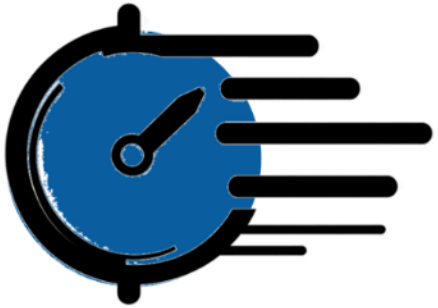
Orders of magnitude slower

Hardware security sign-off



Comprehensive analysis

User-defined & general rules



RTL designers need speed

Static Sign-Off Suite

REAL INTENT