High Reliability Reset Domain Checking Solution for the Modern Soc Design

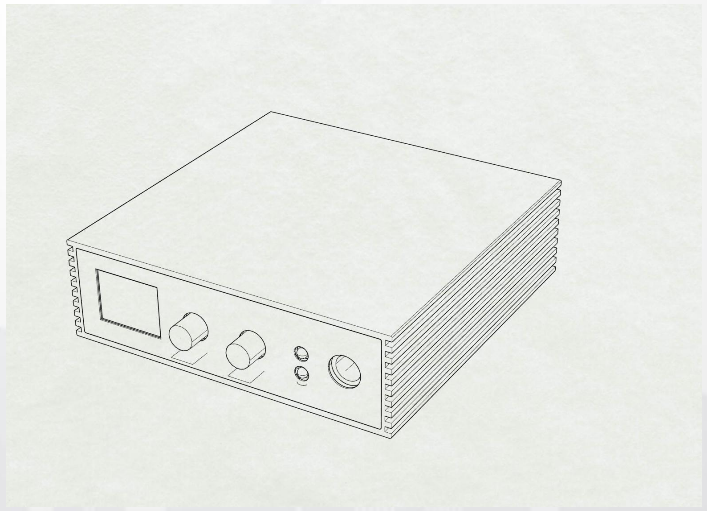
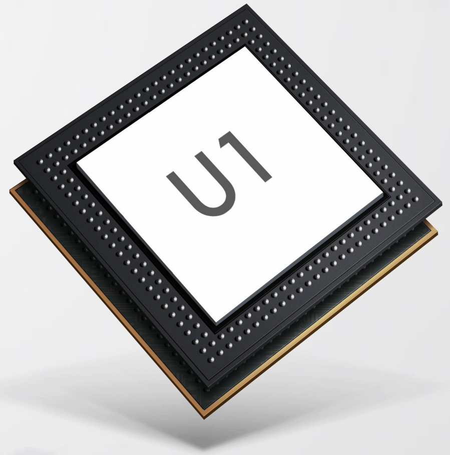
Wanggen Shi, Big Fish Semiconductor Ltd., China

Yuxin You, Mentor, a Siemens Business, China

Kurt Takara, Mentor, a Siemens Business, USA

BigFish Overview

- Focus on AI & IoT and chip solution
- Capability including SoC, sw Dev.& OS, Modem tech, Software&Hardware system integration and Design of 2C&2B products
- Products including mobile, UAV, super Ethernet ,IoT…

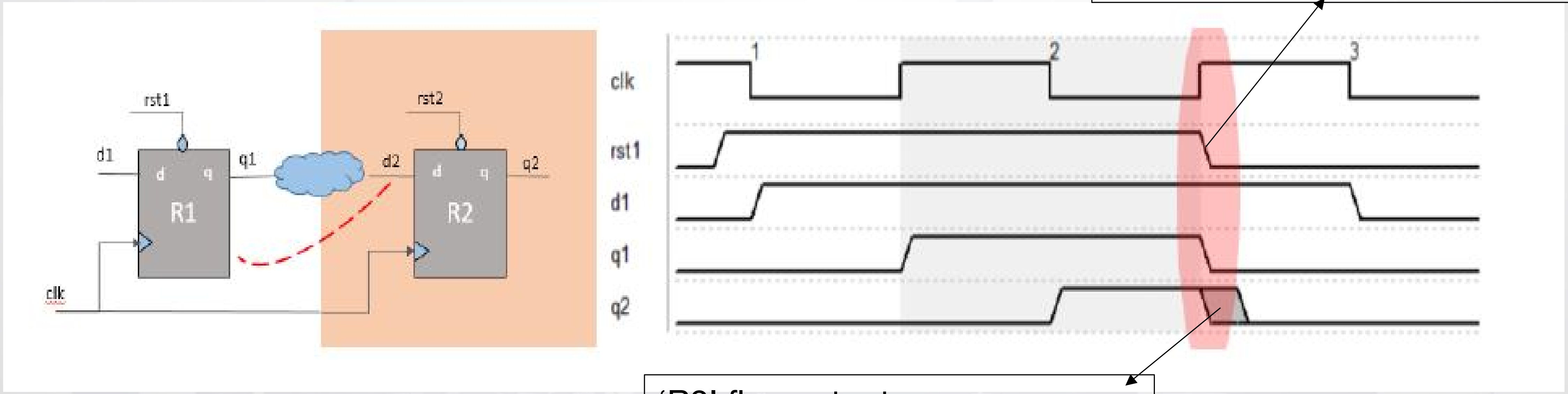
The Need for RDC Verification

- Why reset issues are a problem?
  - Asynchronous reset domain crossing cause metastability
  - Reset issues result in unreliable functionality or possible silicon damage
  - Functional simulation detection is probabilistic
- Reset domain crossing (RDC) verification
  - Static and formal methods detect RDC issues in RTL designs

What is RDC?

- Data crossing from one async reset domain to another
- Transmitting(Tx) flop async-reset assertion close to clock edge can cause metastability on receiving(Rx) flop

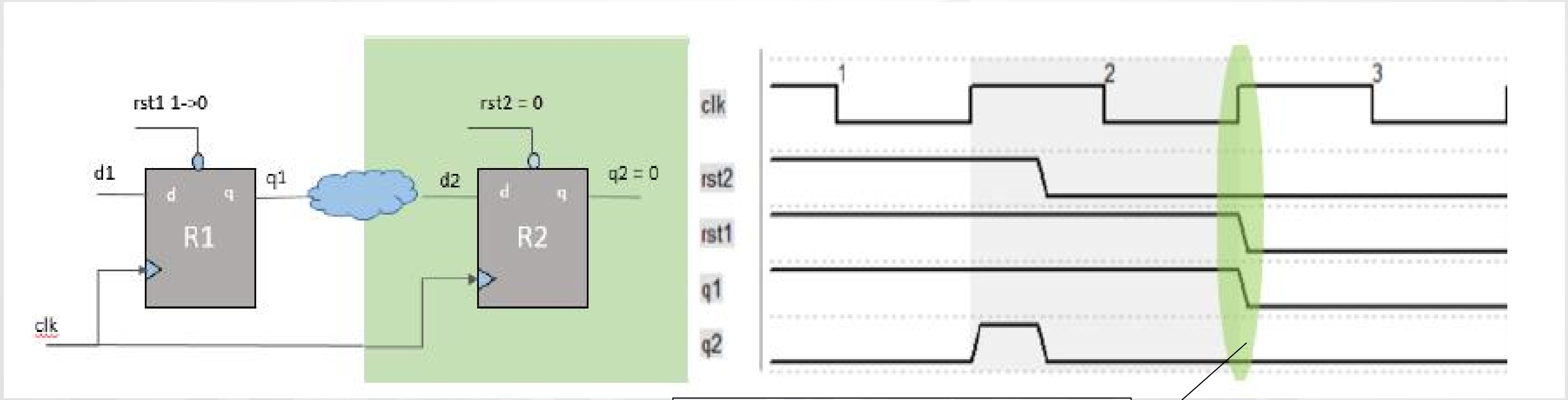
‘rst1’ asserted very close to ‘clk’ posedge



‘R2’ flop output goes metastable due to setup/hold time violation

Techniques to Address RDC issues

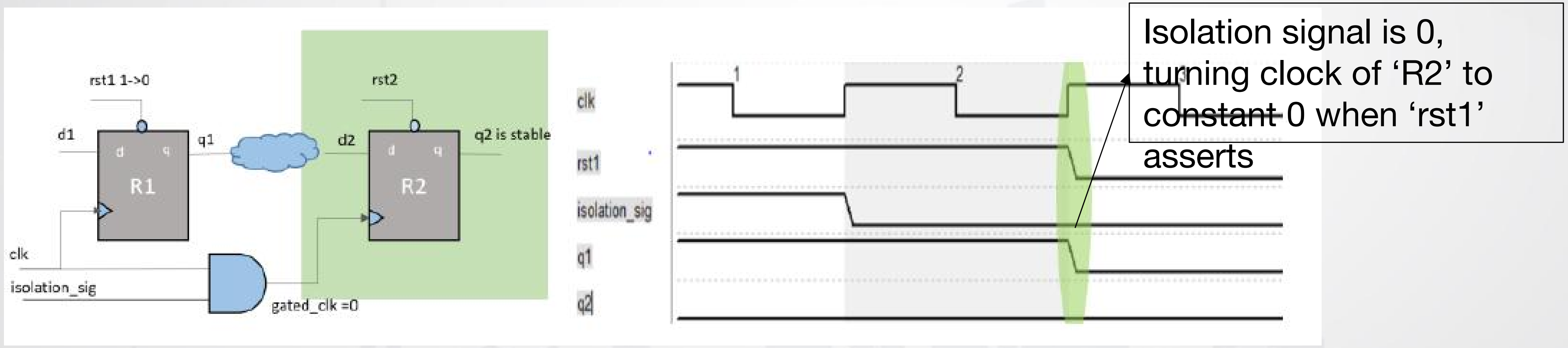
- Reset Sequencing
  - Async-reset on Rx flop always asserts before async-reset on Tx flop
  - Rx flop already in reset state, so any change on Rx D-pin will not cause metastability



‘R2’ flop output is already 0, when Tx reset asserts

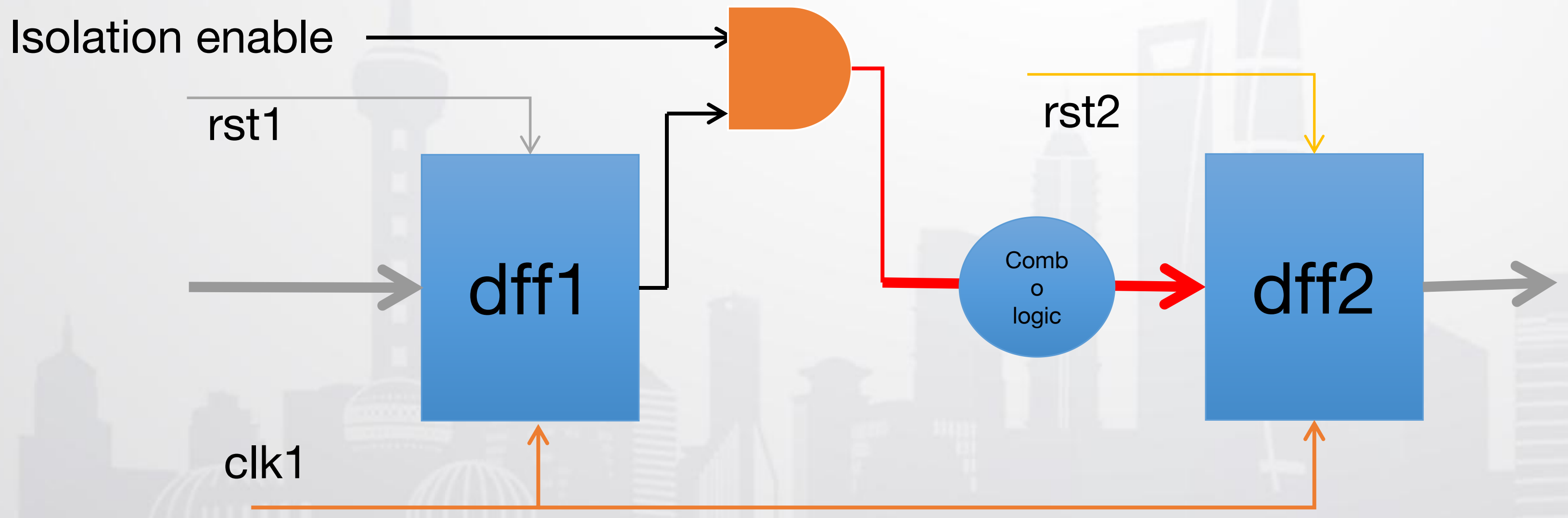
Techniques to Address RDC issues

- Clockgate isolation
  - Turn off clock of Rx flop before Tx reset asserts
  - If clock is off, then any change on Rx D-pin will not cause metastability



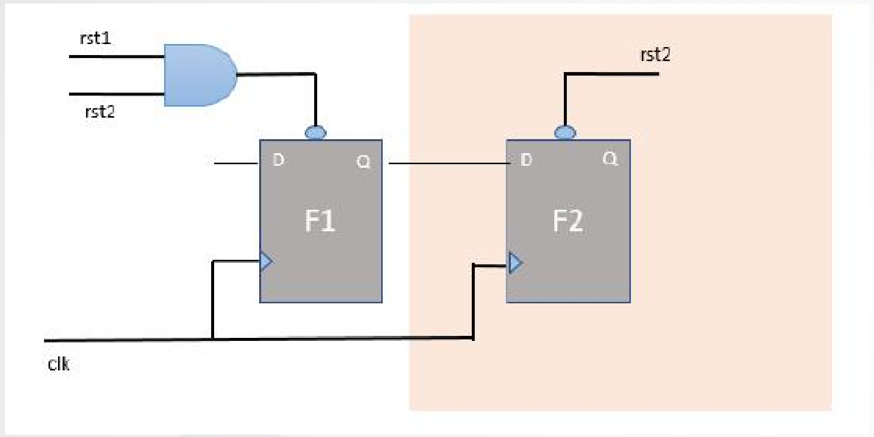
Techniques to Address RDC issues

- Data Isolation
    - Isolation signal from a reset controller isolates the output of the first flop when its reset is asserted. There is a handshake protocol between the enables and the corresponding resets.
    - There is a mapping between isolation enables and resets

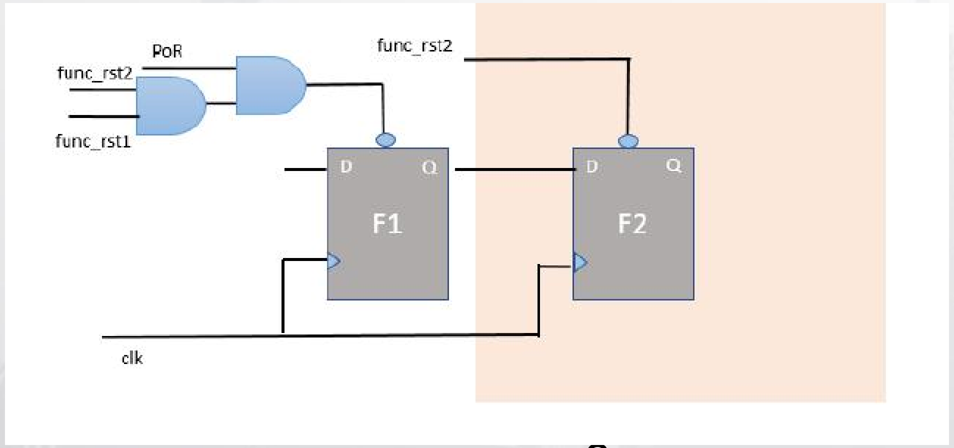


Real-world RDC Issues Examples

- Combination of resets on Tx or Rx flop



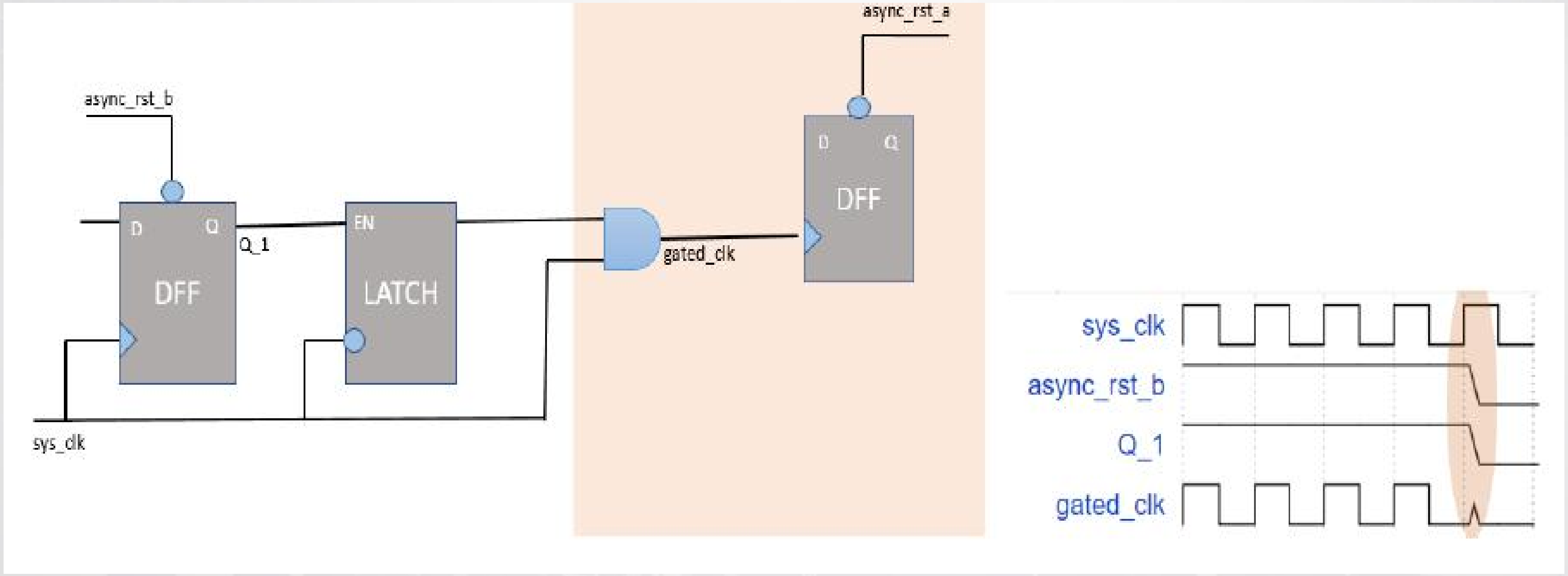
If rst1 asserts before rst2, metastability can occur on F2



If func_rst1 or PoR asserts before func_rst2, metastability can occur on F2

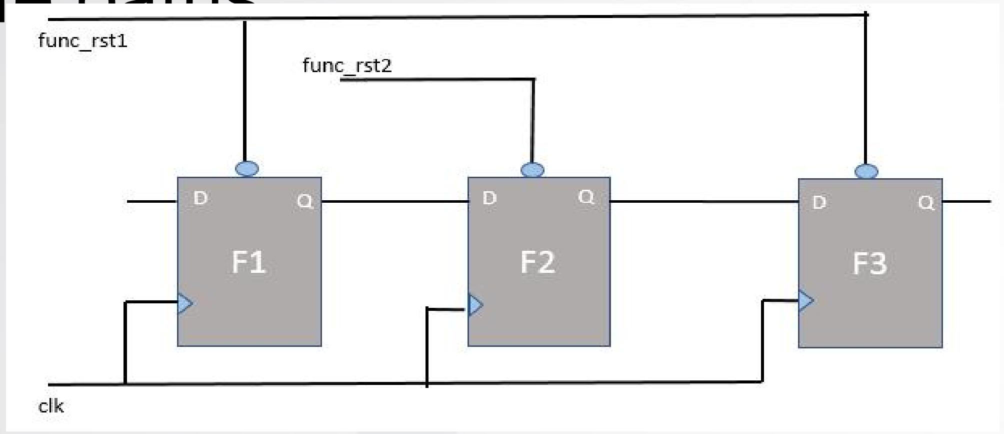
Real-world RDC Issues Examples

- Glitch in gated clock output due to different asynchronous reset

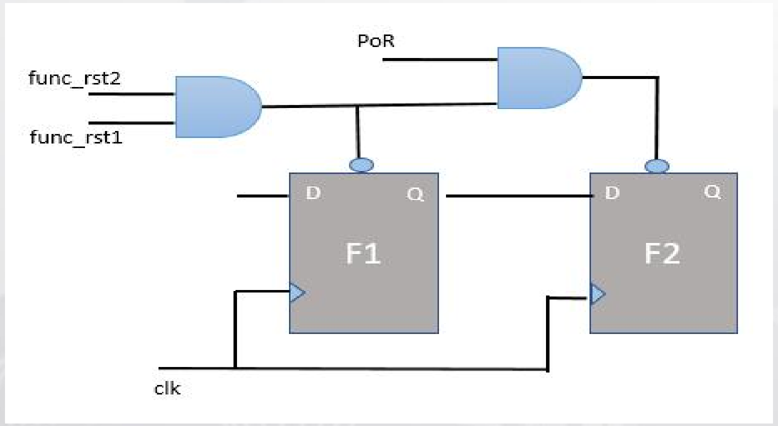


Safe-RDCs Examples

- Some scenarios which may look to be RDC issue, but actually safe paths

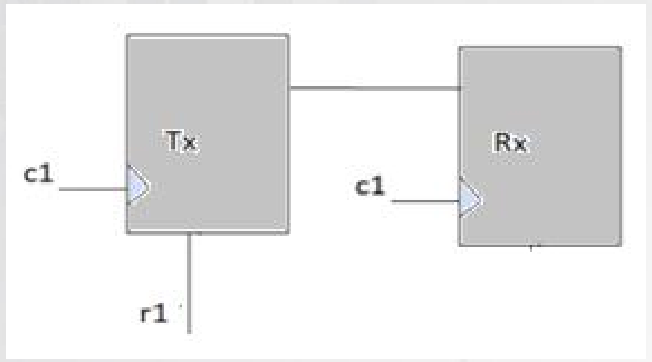


Fanouts of Rx Flop(F2) transmits to Tx Reset Domain (func_rst1)
Tx reset assertion blocks metastability transmission

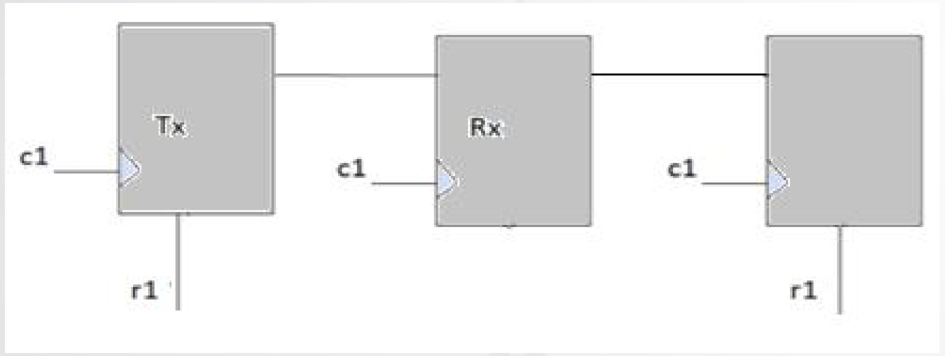


When Tx reset is asserted (through func_rst1/func_rst2) it always asserts Rx reset

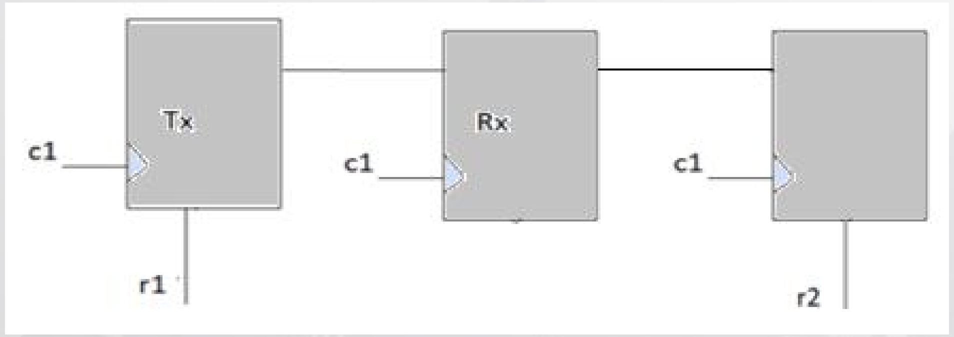
RDC at Reset-less Register Example



Unsafe crossing: From an async reset domain flop to a resetless flop

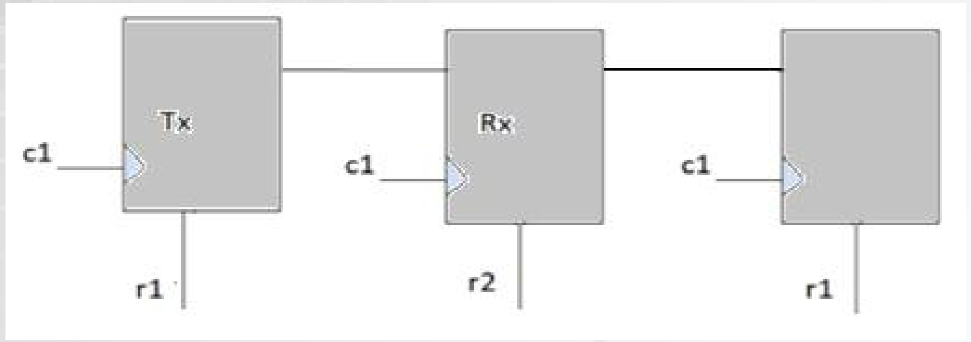


RDC Safe fanout: From an async reset domain flop to resetless flop followed by same async reset domain flop



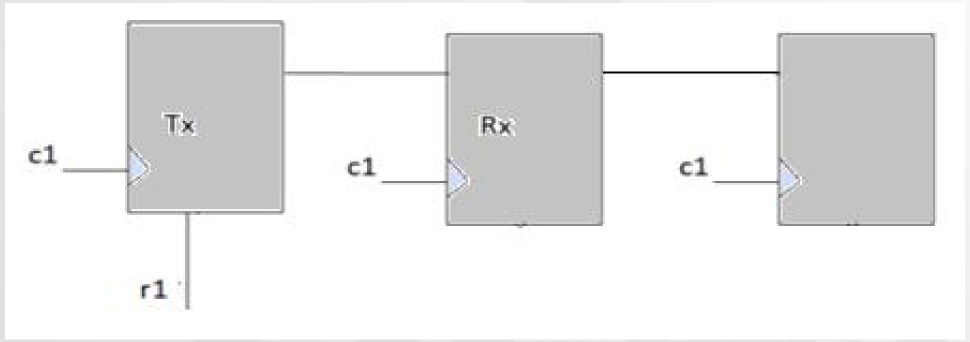
Unsafe crossing: From an async reset domain flop to a resetless flop followed by different async reset domain flop

Safe RDC Fanout Example



Meta-stability introduced at Rx flop is suppressed at next flop with Tx reset domain.

Safe crossing: From an async reset domain flop to another async reset domain flop followed by same Tx reset domain flop.



Back to back 2DFF structure at Rx, synchronizes the RDC.

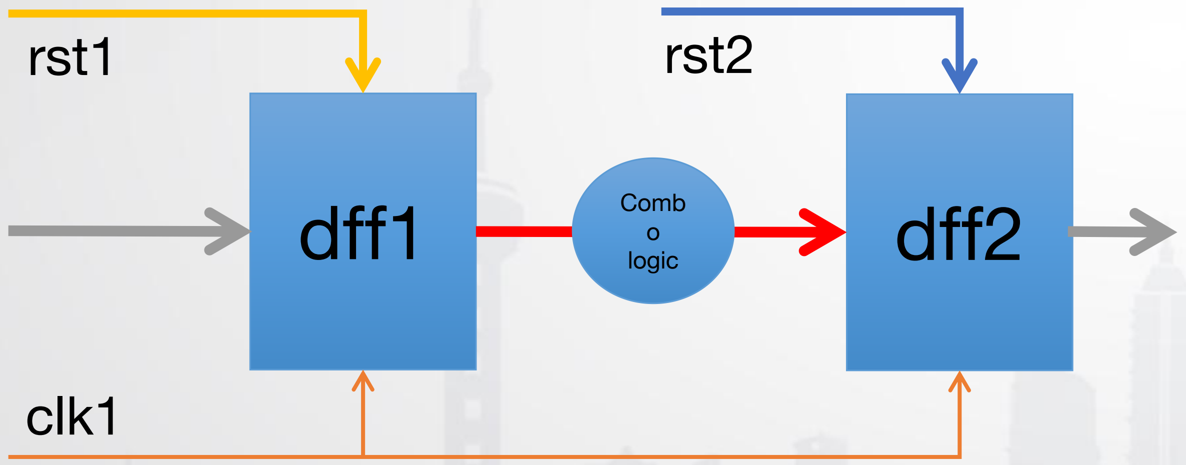
Synchronized crossing: From an async reset domain flop to back to back two resetless flops forming a 2dff synchronizer.

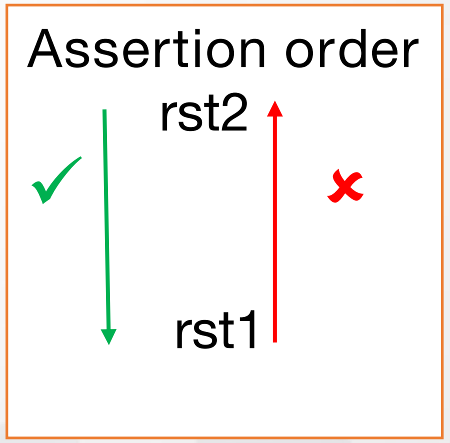
Reset assertion Ordering

Meta-stability occurs only if Rx flop is in functional mode, while Tx reset occurs.

If assertion of rst2 is guaranteed to occur before rst1, it's safe RDC.

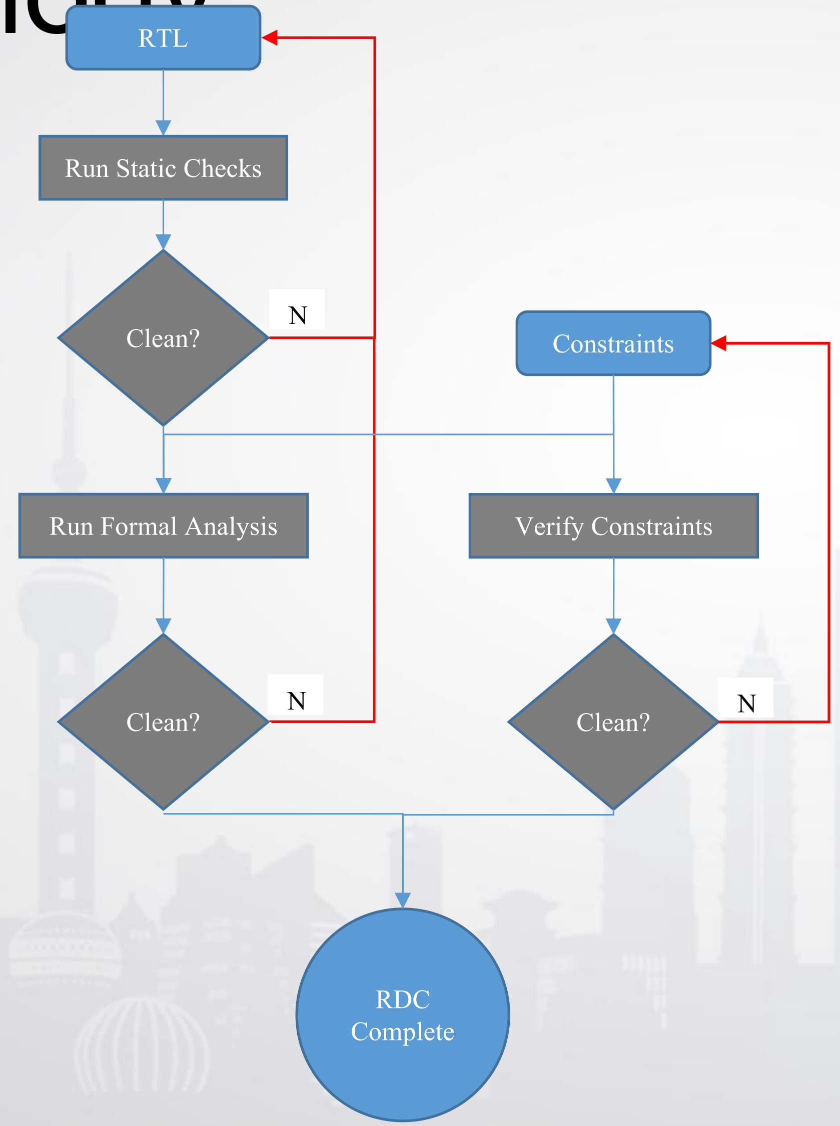
If assertion of rst1 can occur before assertion of rst2, it's an RDC issue.





resetcheck order assert -from rst2 –to rst1

Recommended RDC Verification Methodology



RDC Verification Preparation

- Static reset checks
  - Validate reset tree integrity and RDC verification readiness
  - Assist in developing constraints
- Quality constraints
  - Defines reset sequencing and safe isolation enables
  - Isolation enables validated by functional checks

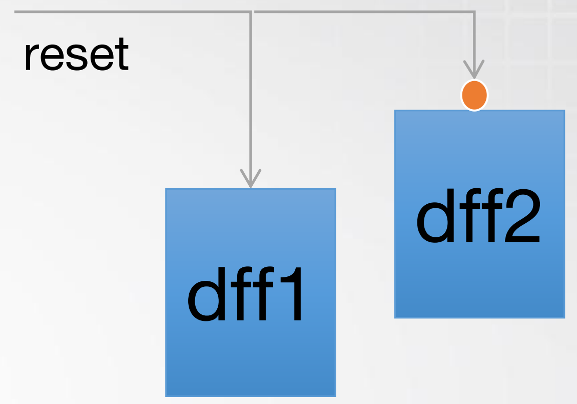
Reset Tree Integrity Analysis

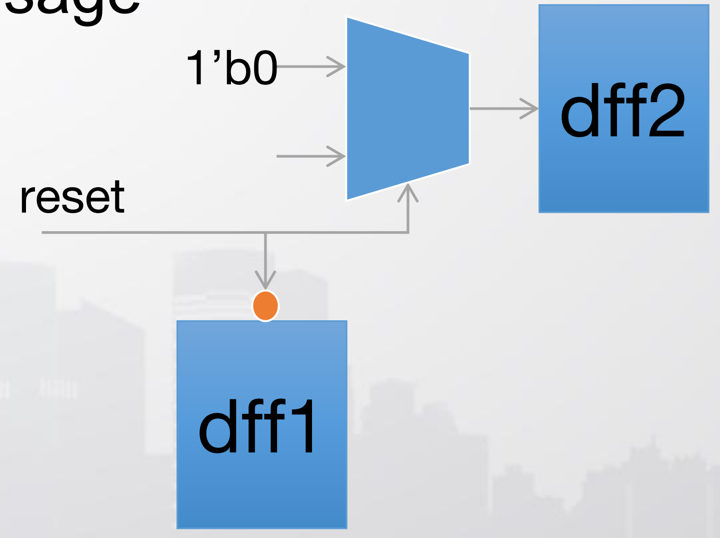
Detect reset tree glitch issues

Detect inconsistent reset polarity

Detect inconsistent asynchronous/synchronous reset usage

More than 20 reset tree checks

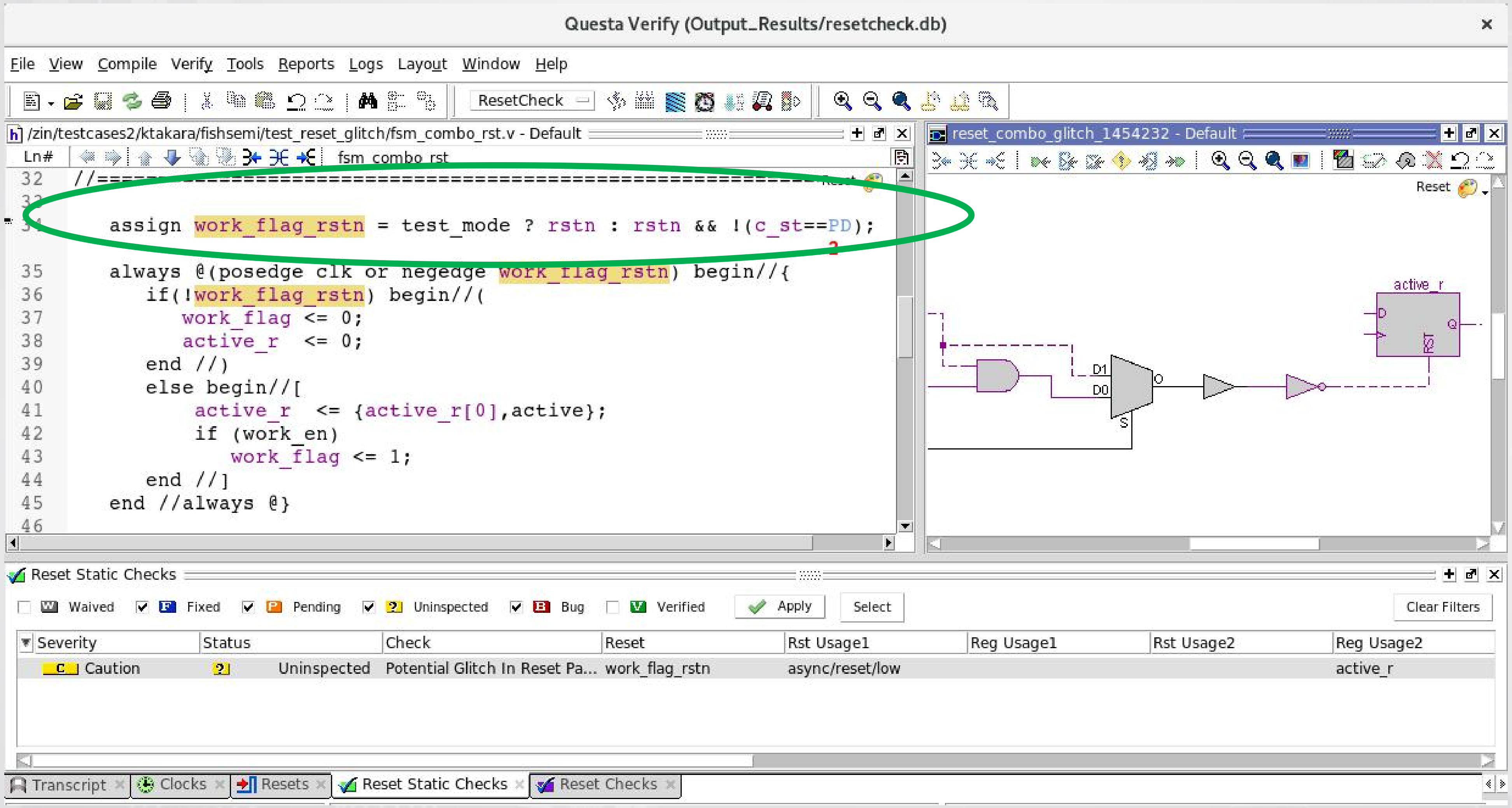




Case Study of Project Usage

- Glitches found by static checks
  - Static reset check identified potential reset glitch
  - Reset glitch would occur for specific state of state machine
- Identified asynchronous FIFO issue
  - Asynchronous FIFO used in a synchronous application
  - TX and RX resets on asynchronous resets
  - RDC verification identified a violation due to missing constraint

Reset Glitch Detection Case

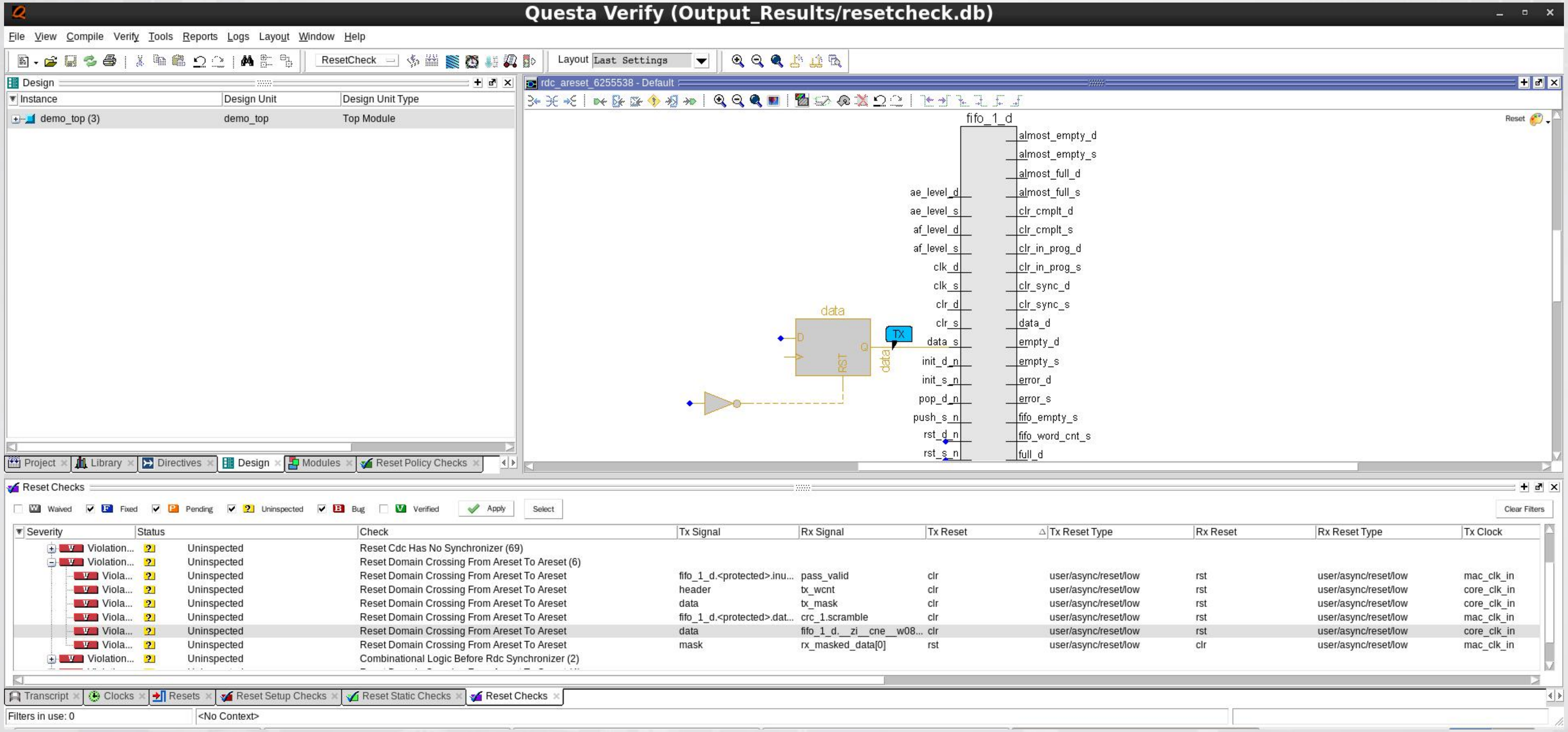

Questa Verify (Output_Results/resetcheck.db)
File View Compile Verify Tools Reports Logs Layout Window Help
ResetCheck
/zin/testcases2/ktakara/fishsemi/test_reset_glitch/fsm_combo_rst.v - Default
Ln#
fsm_combo_rst
reset_combo_glitch_1454232 - Default
Reset
active_r
D1
D0
S
Reset Static Checks
Transcript
Clocks
Resets
Reset Static Checks
Reset Checks

```
//=================================================================

   assign work_flag_rstn = test_mode ? rstn : rstn && !(c_st==PD);

   always @(posedge clk or negedge work_flag_rstn) begin//{
      if(!work_flag_rstn) begin//(
         work_flag <= 0;
         active_r  <= 0;
      end //)
      else begin//[
          active_r  <= {active_r[0],active};
          if (work_en)
             work_flag <= 1;
      end //]
   end //always @}
```

☐ Waived ☑ Fixed ☑ Pending ☑ Uninspected ☑ Bug ☐ Verified — Apply — Select — Clear Filters

Severity	Status	Check	Reset	Rst Usage1	Reg Usage1	Rst Usage2	Reg Usage2
Caution	Uninspected	Potential Glitch In Reset Pa...	work_flag_rstn	async/reset/low			active_r

FIFO RDC Detection Case



Severity	Status	Check	Tx Signal	Rx Signal	Tx Reset	Tx Reset Type	Rx Reset	Rx Reset Type	Tx Clock
Violation...	Uninspected	Reset Cdc Has No Synchronizer (69)							
Violation...	Uninspected	Reset Domain Crossing From Areset To Areset (6)							
Viola...	Uninspected	Reset Domain Crossing From Areset To Areset	fifo_1_d.<protected>.inu...	pass_valid	clr	user/async/reset/low	rst	user/async/reset/low	mac_clk_in
Viola...	Uninspected	Reset Domain Crossing From Areset To Areset	header	tx_wcnt	clr	user/async/reset/low	rst	user/async/reset/low	core_clk_in
Viola...	Uninspected	Reset Domain Crossing From Areset To Areset	data	tx_mask	clr	user/async/reset/low	rst	user/async/reset/low	core_clk_in
Viola...	Uninspected	Reset Domain Crossing From Areset To Areset	fifo_1_d.<protected>.dat...	crc_1.scramble	clr	user/async/reset/low	rst	user/async/reset/low	mac_clk_in
Viola...	Uninspected	Reset Domain Crossing From Areset To Areset	data	fifo_1_d.__zi__cne__w08...	clr	user/async/reset/low	rst	user/async/reset/low	core_clk_in
Viola...	Uninspected	Reset Domain Crossing From Areset To Areset	mask	rx_masked_data[0]	rst	user/async/reset/low	clr	user/async/reset/low	mac_clk_in
Violation...	Uninspected	Combinational Logic Before Rdc Synchronizer (2)							

Future Plans

- Utilize constraints and waivers to achieve RDC verification closure
- Verify constraints with SVA assertion flow
- Verify low-power structures with PA-RDC
- For larger designs, utilize hierarchical RDC flow

Summary

- Increase in SoC reset architecture complexity requires RDC verification
- RDC verification methodology provides completeness and efficiency
- RDC verification metrics demonstrate verification completion
- Achieved our goal of reliable silicon operation!

Thanks you