



Never too late with Formal: Stepwise Guide for Applying FV in Post-Si Phase to Avoid Re-spins

Anshul Jain, Aarti Gupta, Achutha KiranKumar V M, Bindumadhava Ss, Shivakumar S Kolar, Siva Gadey NV

Intel Corporation

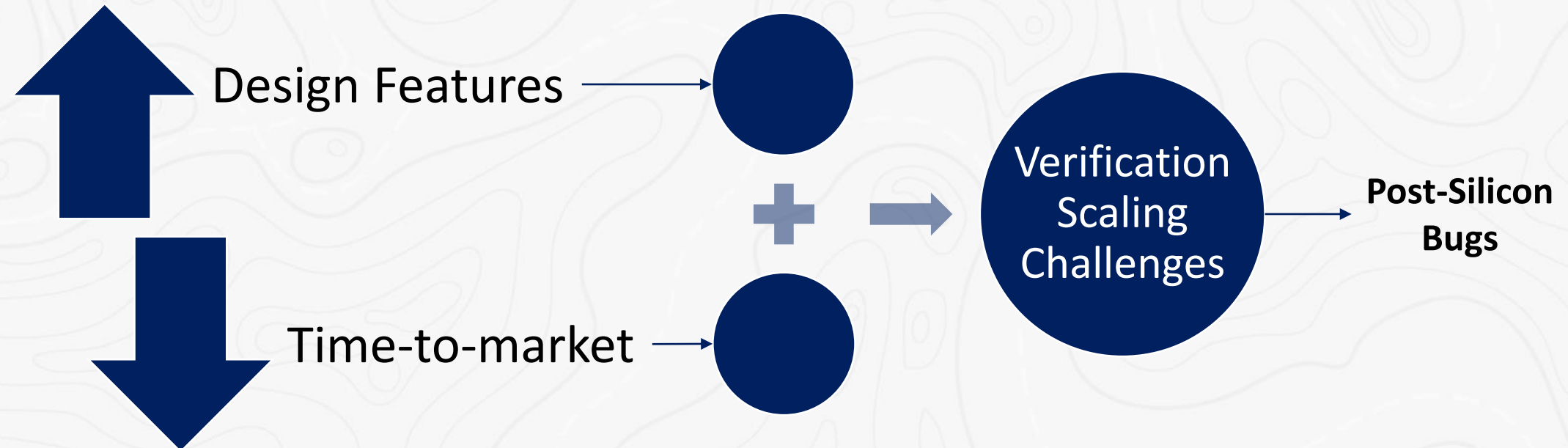


Overview of the Presentation

- Problem statement
- Proposed methodology
- Case Study
- Summary
- Questions

Post-Silicon Design Bugs

- Embrace of formal verification grown over last decade
- Simulation still the main workhorse for pre-silicon functional sign-off

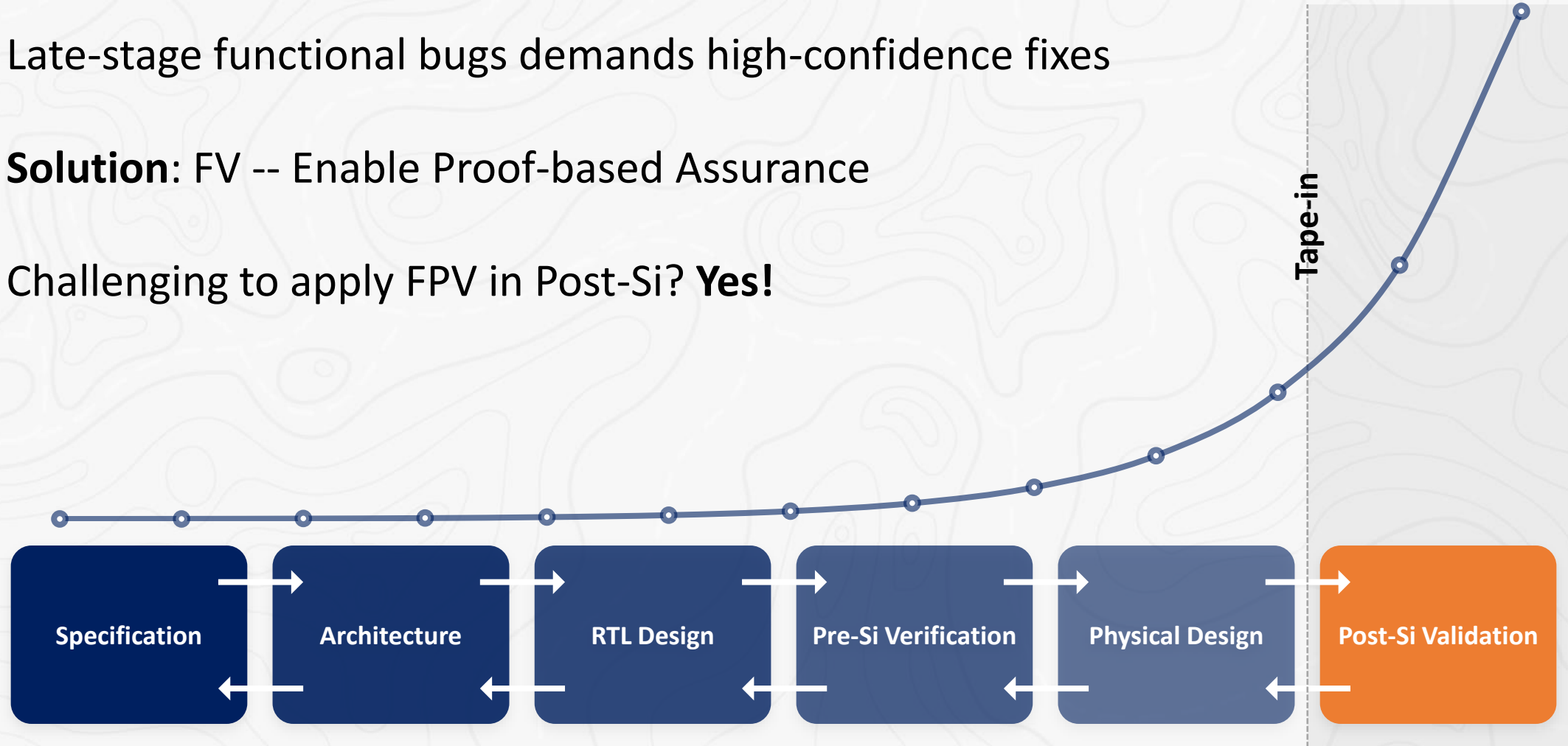


High Stakes >>> Need Absolute Assurance

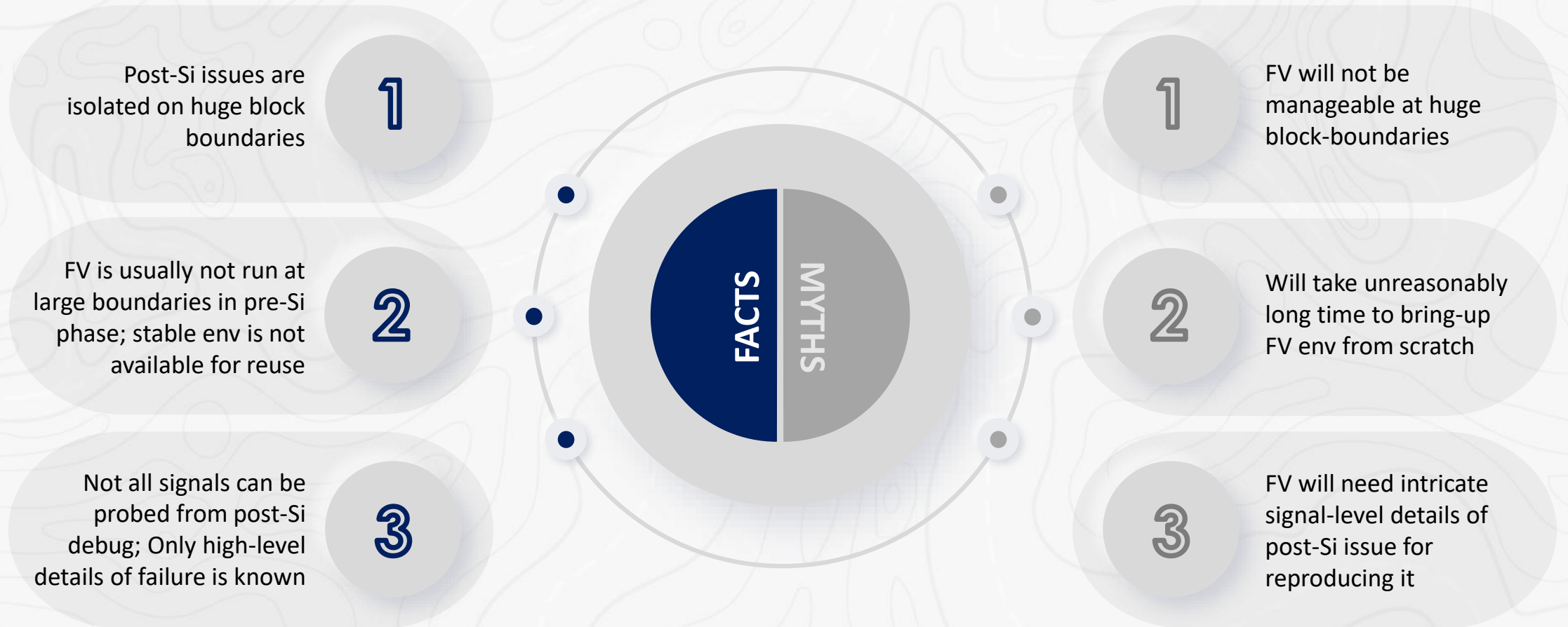
Late-stage functional bugs demands high-confidence fixes

Solution: FV -- Enable Proof-based Assurance

Challenging to apply FPV in Post-Si? **Yes!**



Get the Myths Out of your Way!



Stepwise Approach – UNEARTH



1. Understand Problem & Collect Collaterals

Description of the failure/problem seen in post-silicon testing

Design documentation & source code files

Existing formal verification environments (if exists)

Waveform and register dumps from pre/post-silicon verif/val

Sample waveforms from pre-silicon dynamic simulations

2. Nail-down the Formal Property

Capture a crisp and clear description of post-silicon failure; validate with other stakeholders from design, pre-silicon verification, post-silicon validation teams

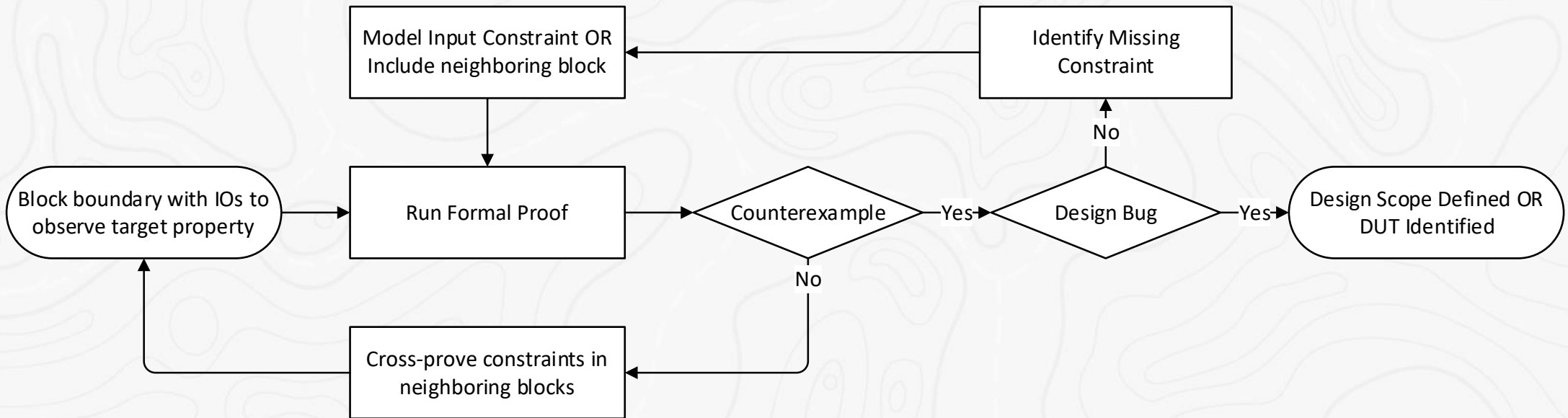
Brainstorm properties (from spec) of the DUT that could be violated in post-silicon failure; Start with general properties, move towards specific properties gradually

For example, responses should be in same order as request, buffer should always have N free entries for pointer management, preemptions should be finite

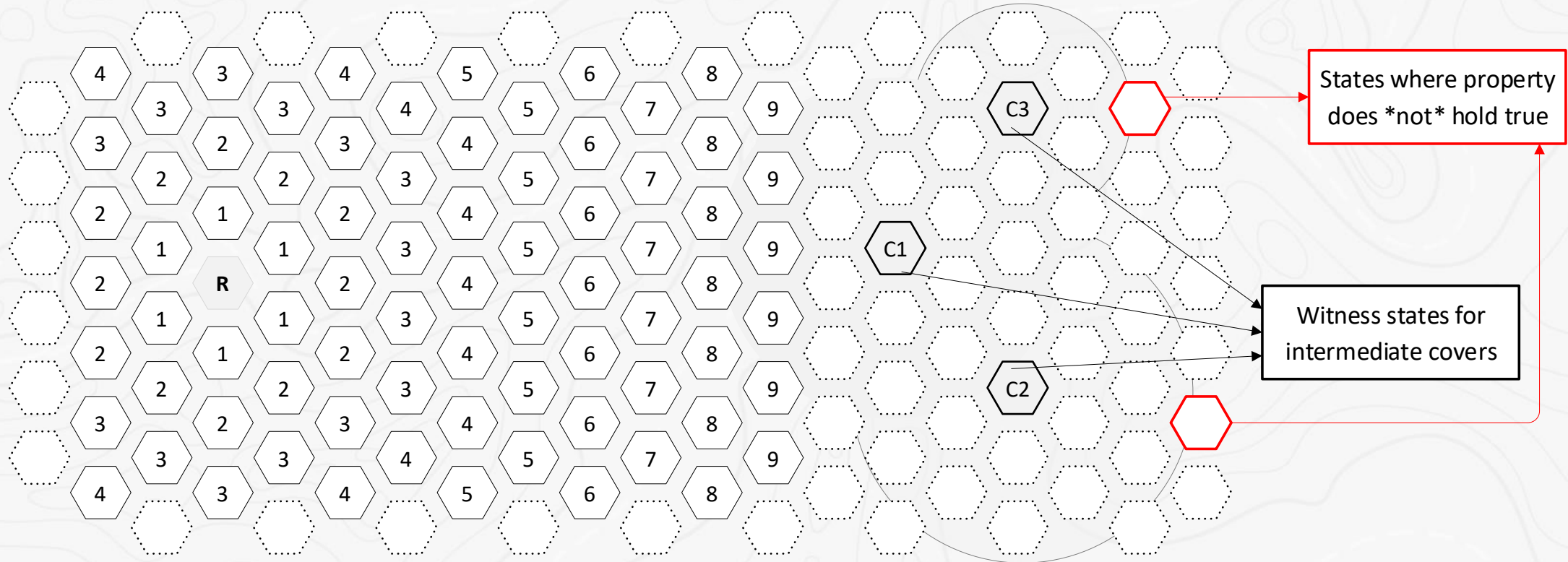
Note down properties in plain, simple, human-readable language without worrying about implementation details

Engage architects, micro-architects, to identify observation points for the property; implement them using light-weight instrumentation code and SVA

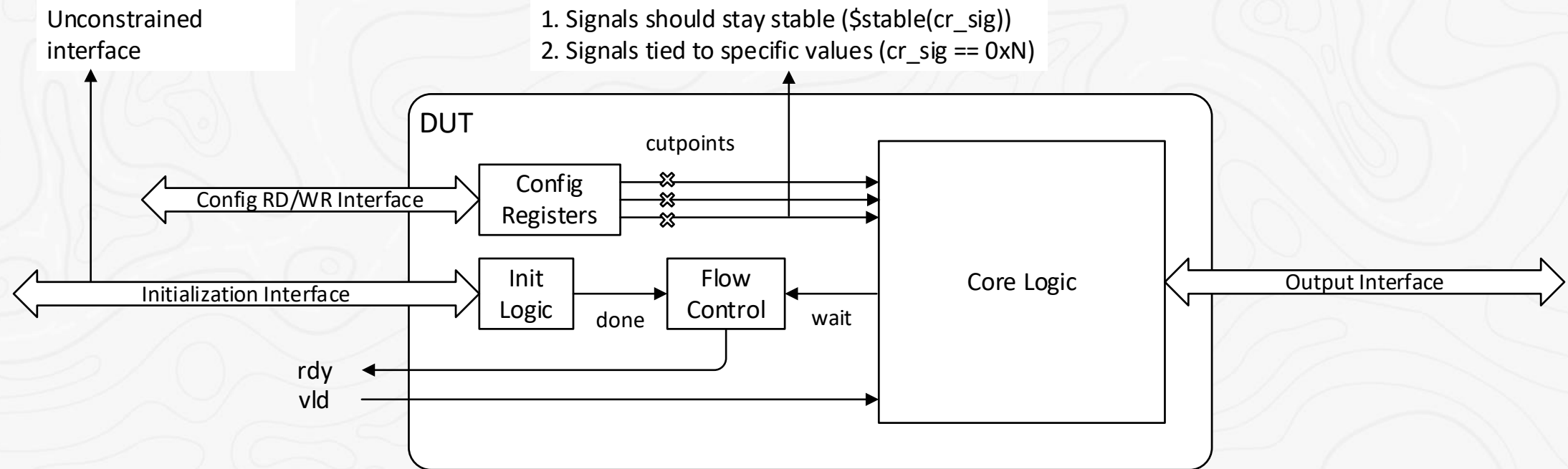
3. Etch Design Boundary for Formal Search



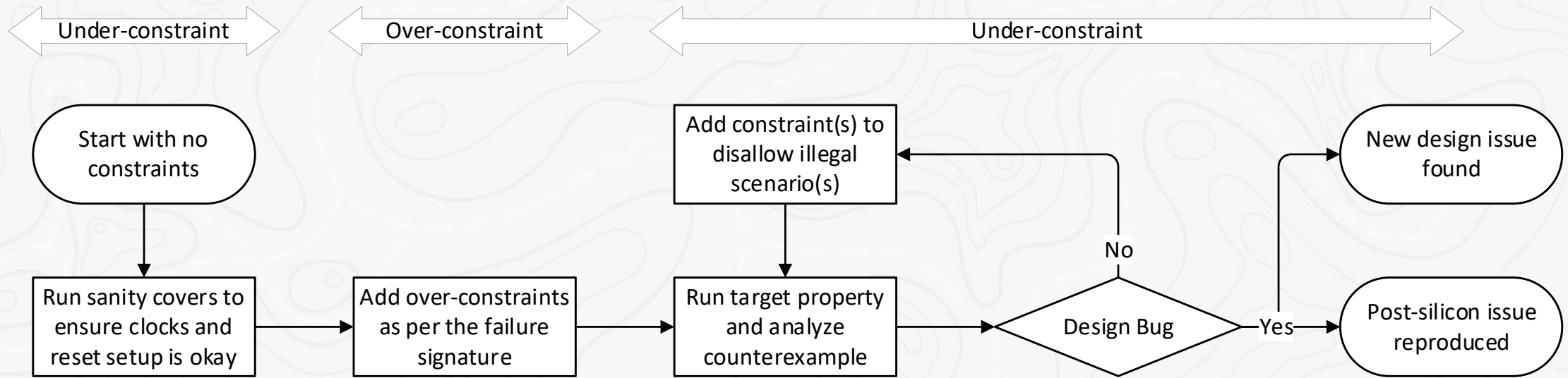
4. Assess Reachability of Source of the Bug



5. Regulate Constraints (Over/Under)



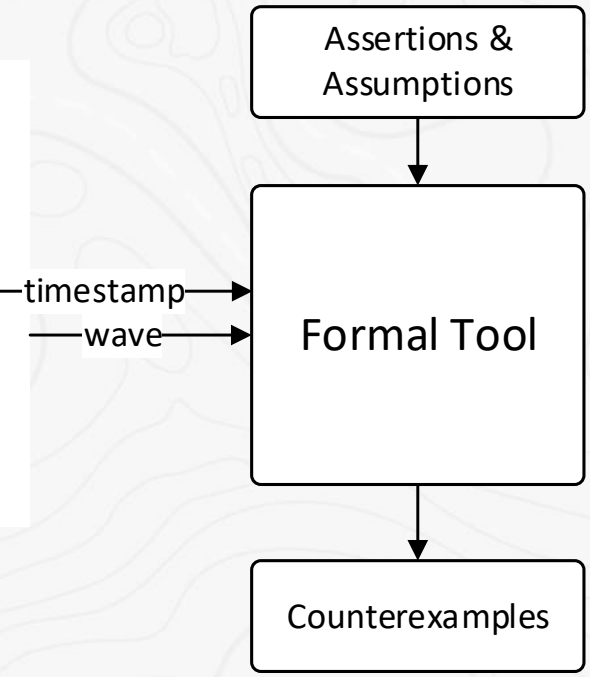
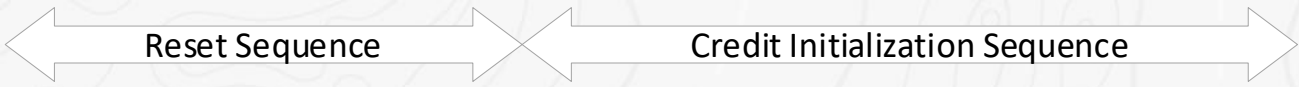
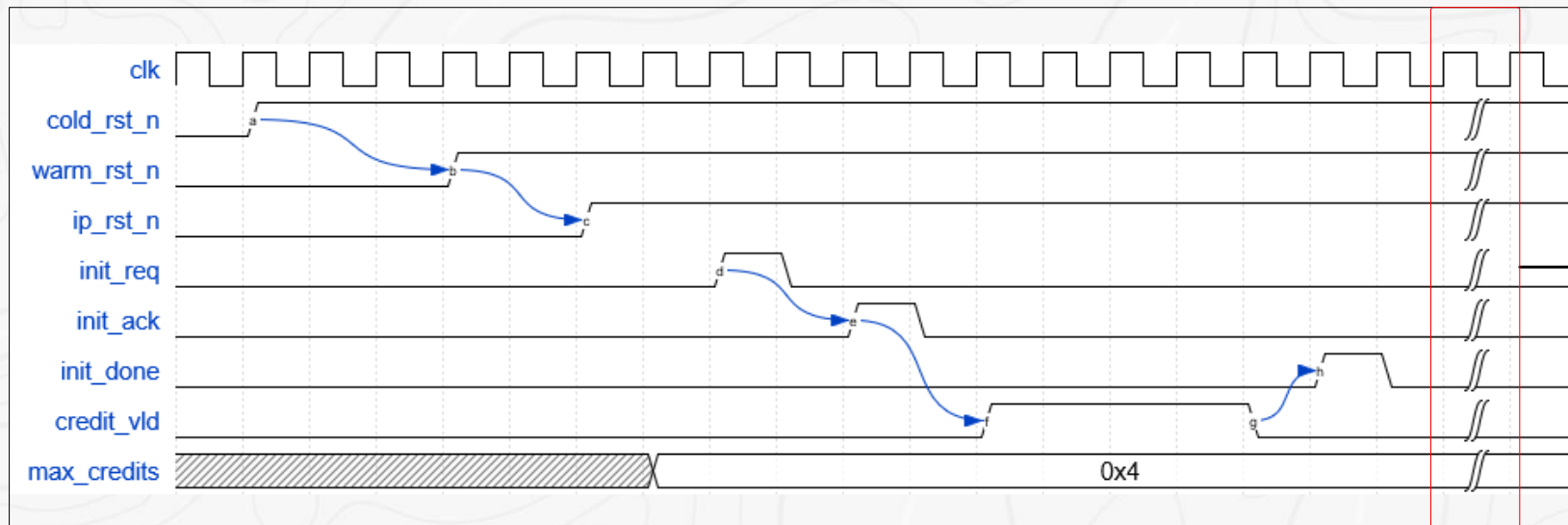
5. Regulate Constraints (Over/Under)



Over-constraints can be used to restrict non-participating interfaces and values of configuration registers

Constraints refinement to keep inputs relevant to failure under-constrained, may flag other failure manifestations/sister-bugs

6. Tap Details from Sims for Formal Search



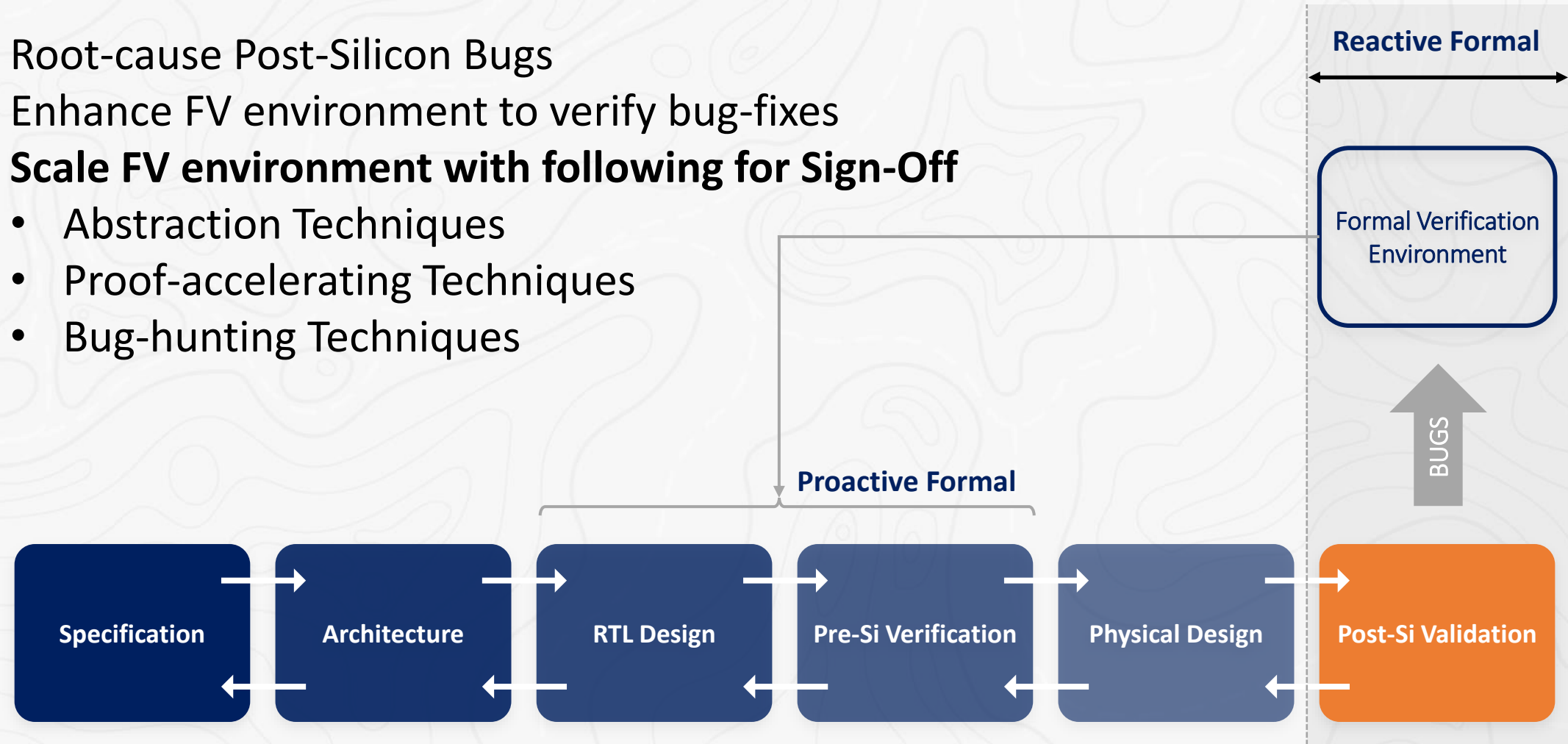
7. Harness Full Potential of Formal Technology

Root-cause Post-Silicon Bugs

Enhance FV environment to verify bug-fixes

Scale FV environment with following for Sign-Off

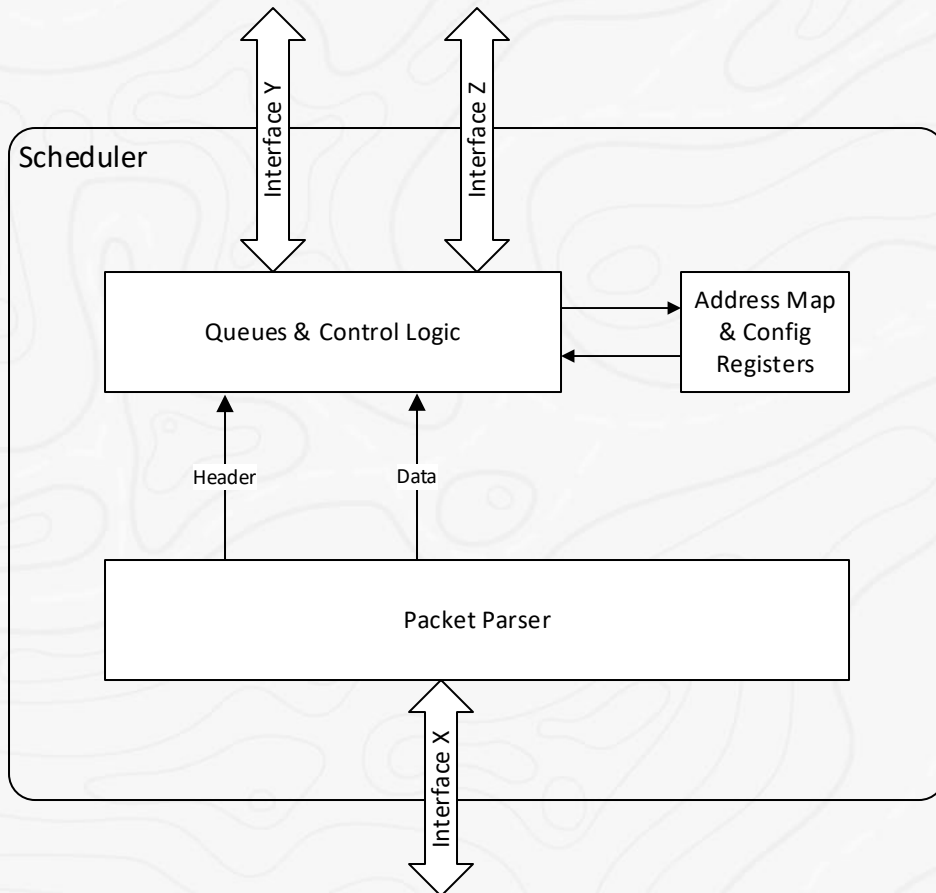
- Abstraction Techniques
- Proof-accelerating Techniques
- Bug-hunting Techniques



Case Study

Post-silicon bug in Scheduler

Scheduler: Design & Bug Details



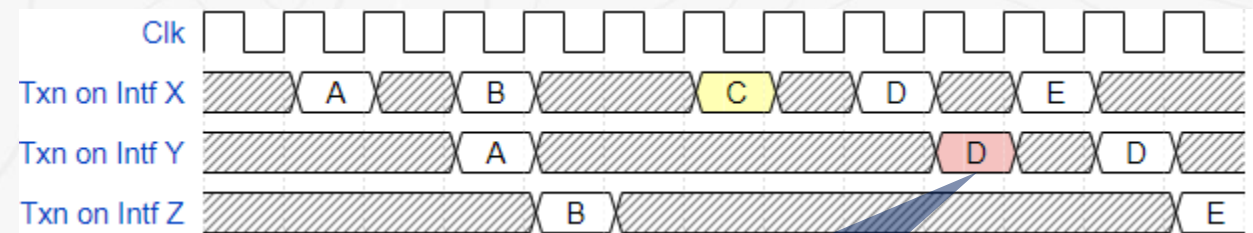
General Functionality:

- Predefined transformations of Packets
- Routing : Interface X -> Interface Y/Z
- Routing based on resource availability and rules

Verification Challenges:

- Huge Design Size: ~4M gates


Bug Synopsis




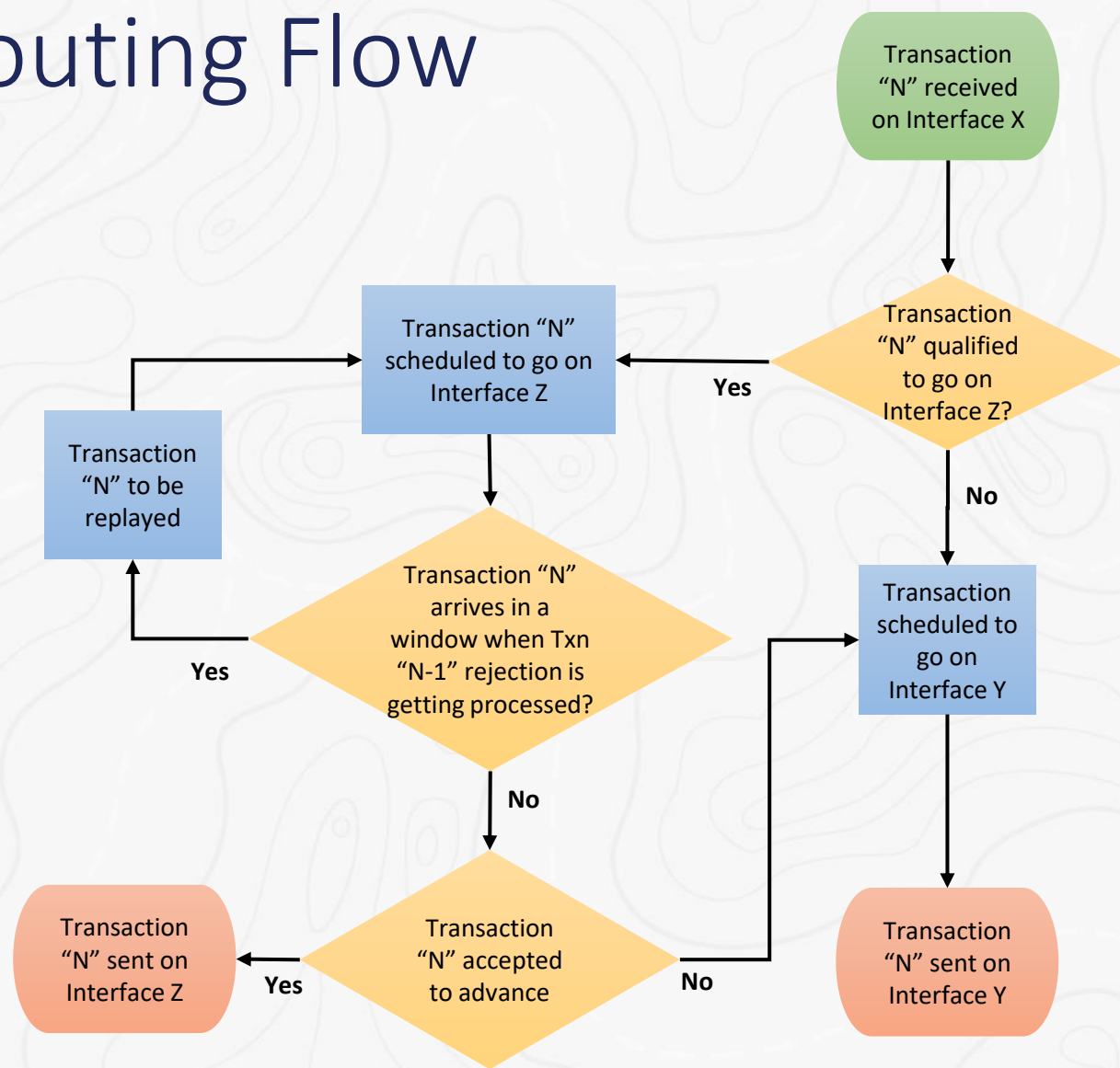
Txn C parameters
overwritten by Txn D

UNDERSTAND THE
PROBLEM & COLLECT
ALL THE COLLATERAL

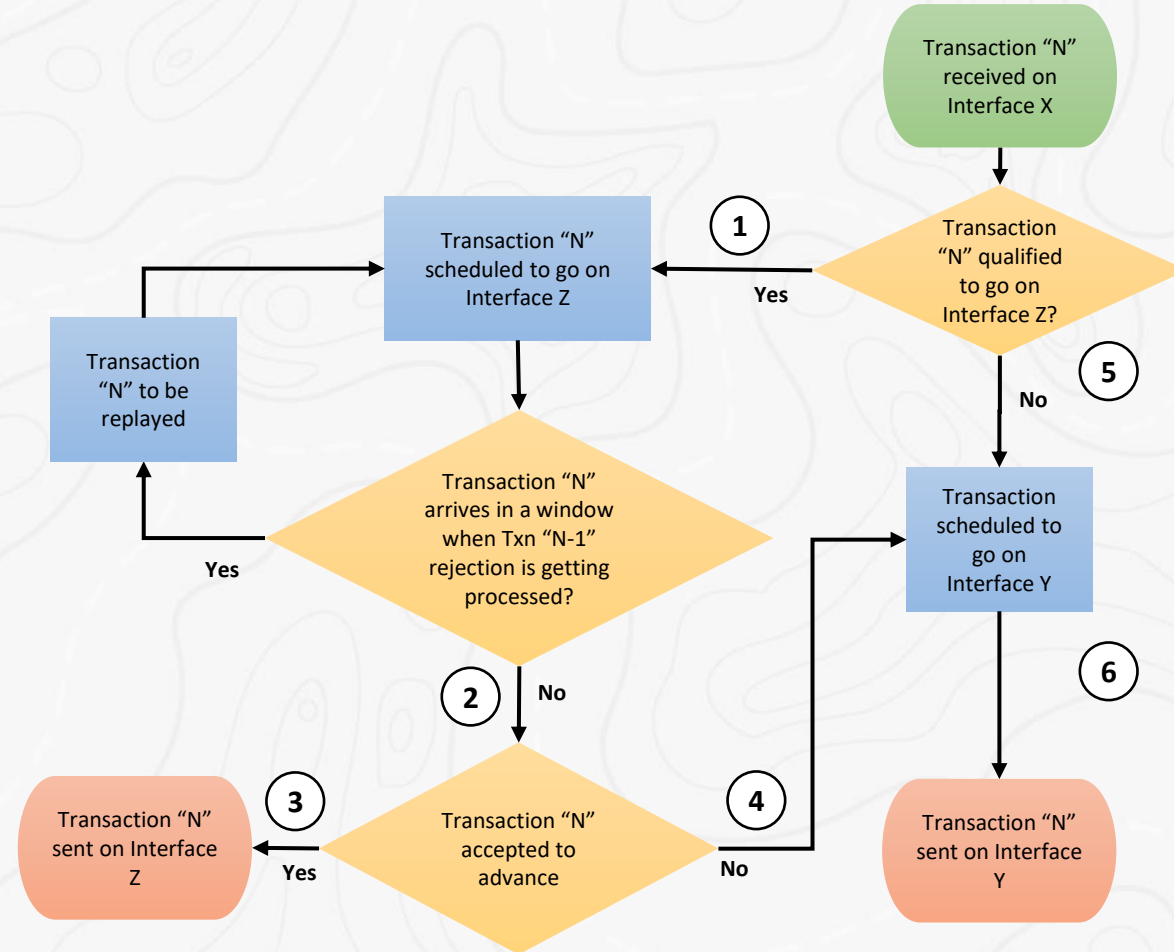
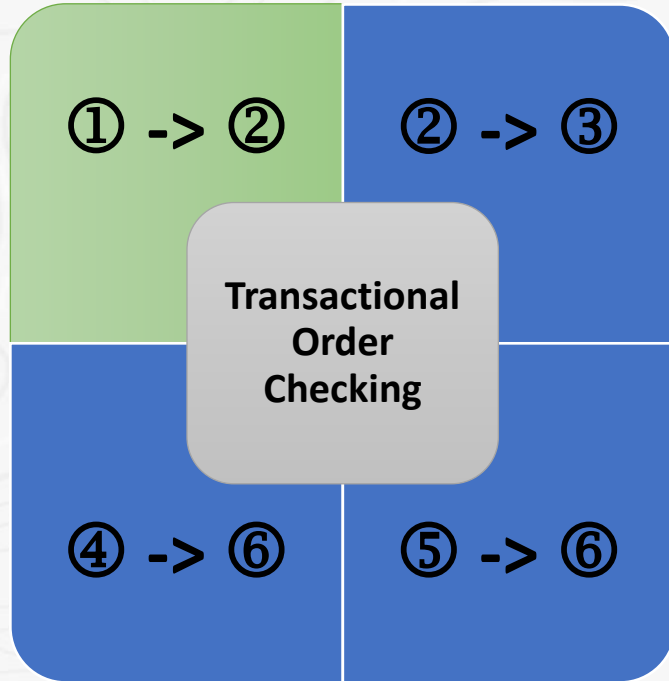
Scheduler: Routing Flow

 Studied
Simulation traces
& specifications

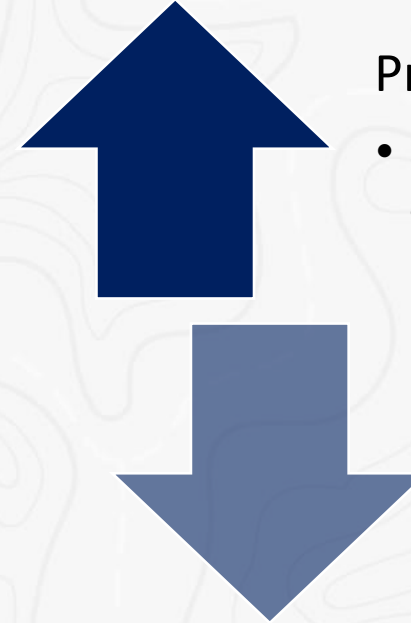
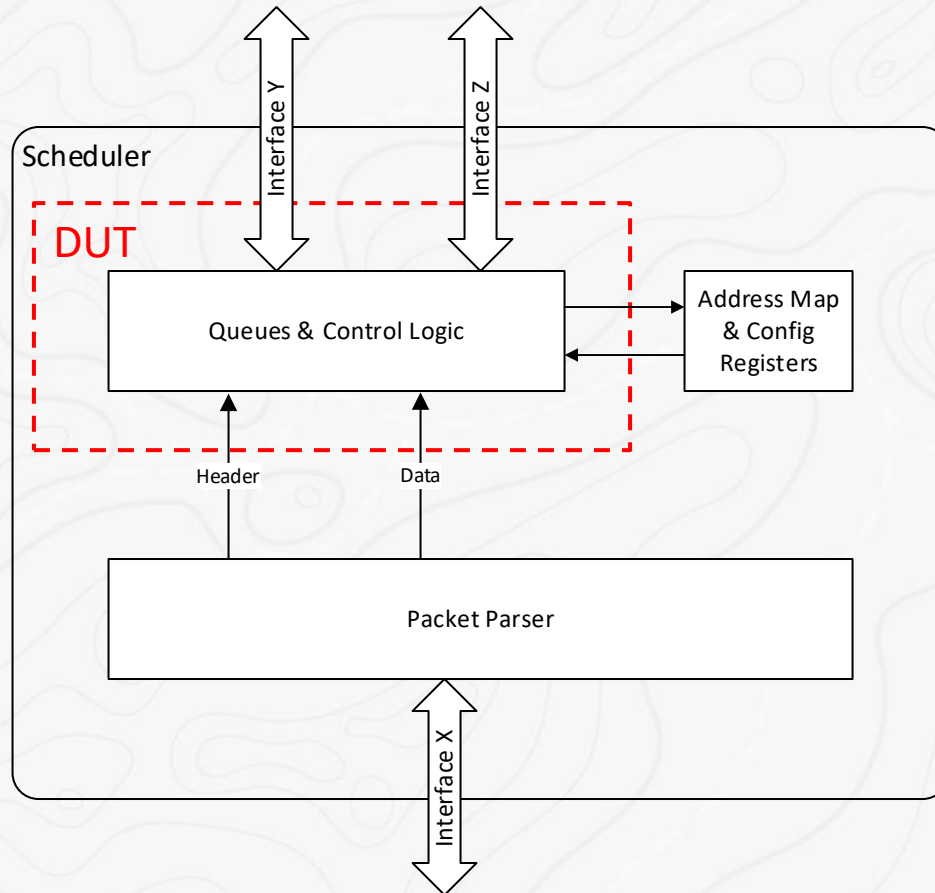
 Created basic
Flow Diagram



Scheduler: Formal Property (Checker)



Scheduler: DUT



Pro:

- Smaller Size (~1.5M gates)

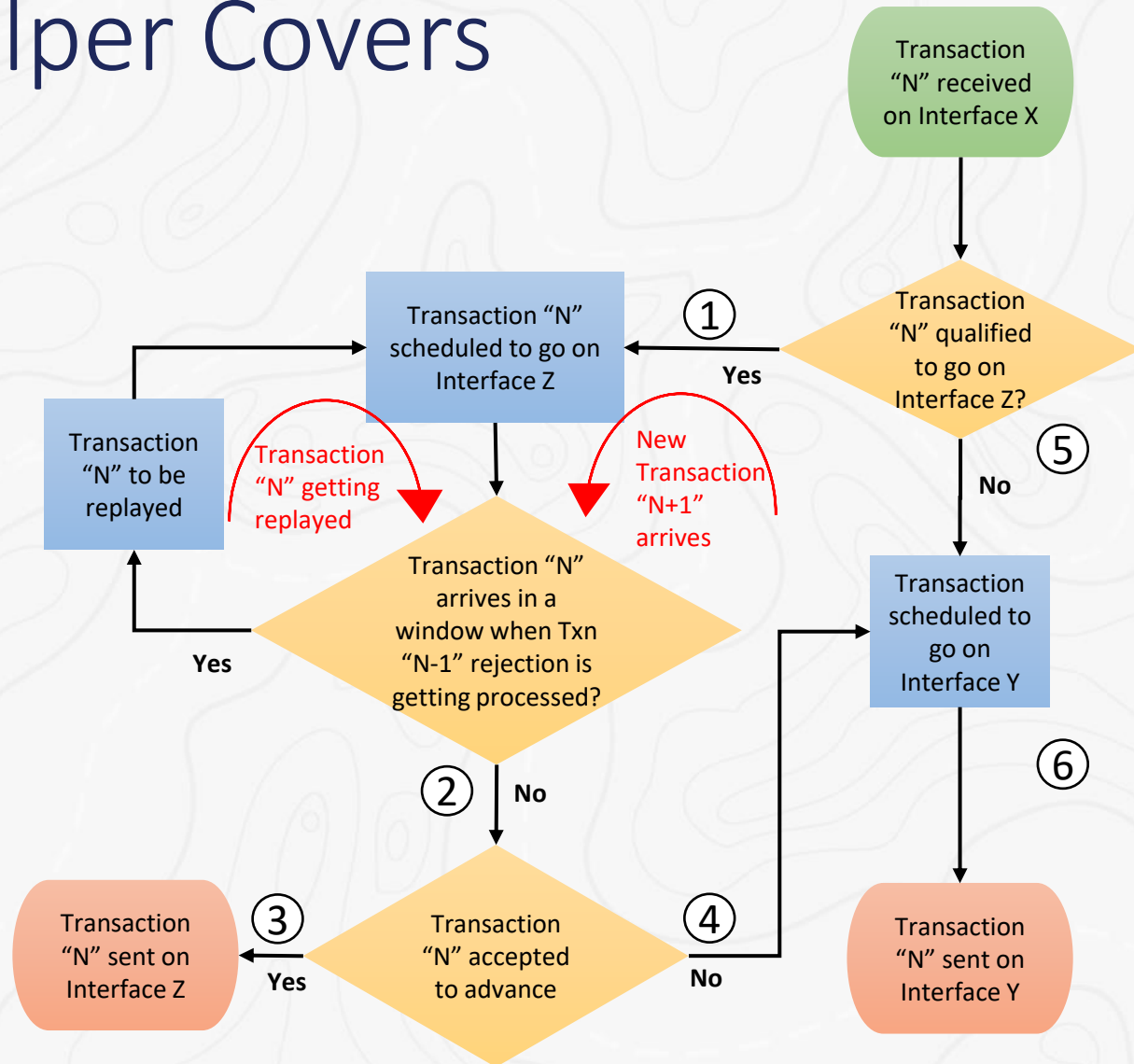
Con:

- Assumption modeling required on internal non-standard interfaces

Scheduler: Helper Covers

Relevant Covers:

- Transaction reachability at each event 1-6
- Reachability of a state where a new transaction conflicts with a replayed instruction

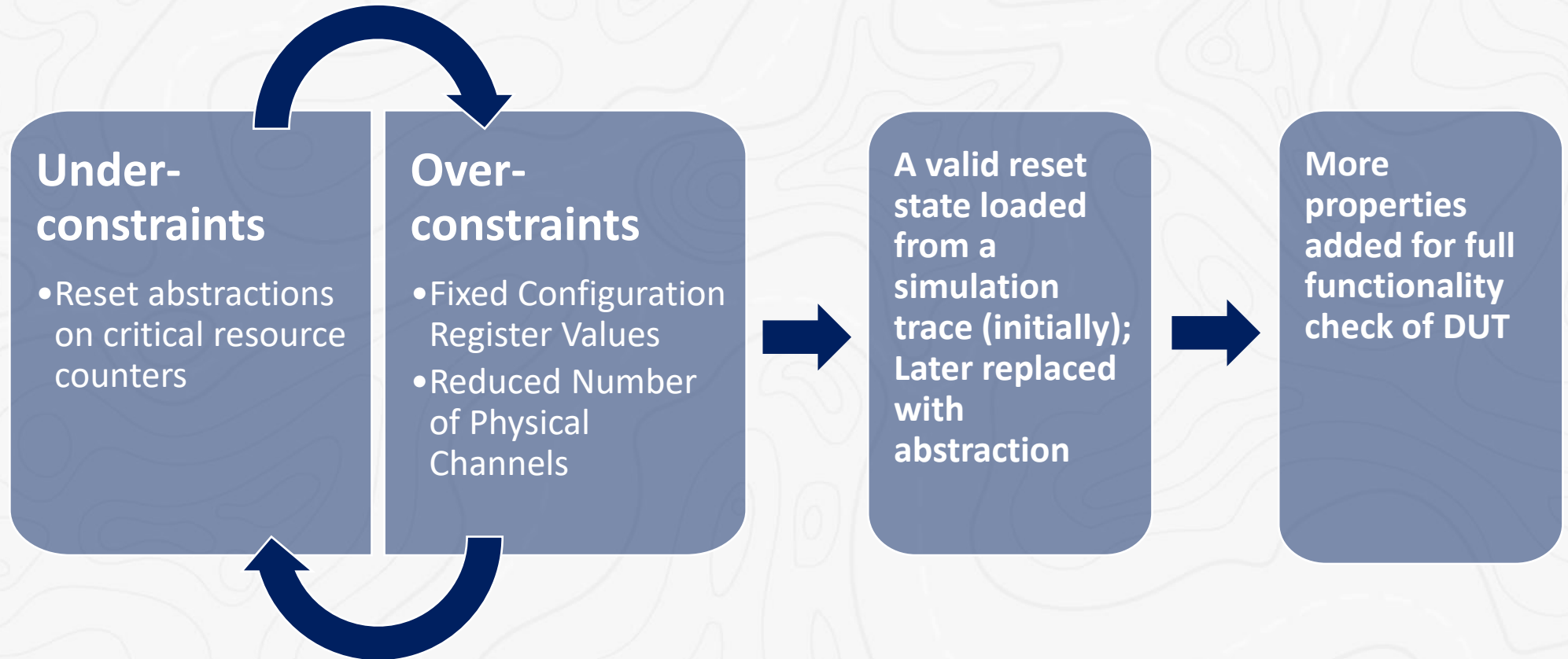


REGULATE
CONSTRAINTS TO
STRIKE A BALANCE
B/W OVER-
CONSTRAINTS &
UNDER-CONSTRAINTS

TAP DETAILS FROM
SIM WAVES TO START
FORMAL SEARCH

HARNESS FULL
POTENTIAL OF
FORMAL
TECHNOLOGY

Scheduler: FV Environment



Scheduler: Results

Post-silicon bug was reproduced in 4 weeks

4 other failing scenarios were detected

FV helped in determining a robust fix

Two fixes verified and compared

FV environment reused in next project as pro-active FV

Key Takeaway: Design size not a limiting factor in Post-Silicon FV

Summary

UNEARTH – Comprehensive guide for Post-Silicon FV

Impactful results seen in the case-study shared

Apart from this case study, several other successful applications

Simple checks can find deep issues in complex designs

Post-Silicon FV motivates transition from reactive to proactive approach

- Identifies more design candidates for FV Signoff
- Targets bug prone designs for next generations

Recommendation

- All post-silicon issues in control-logic should be reproduced in FV

Questions?

Backup

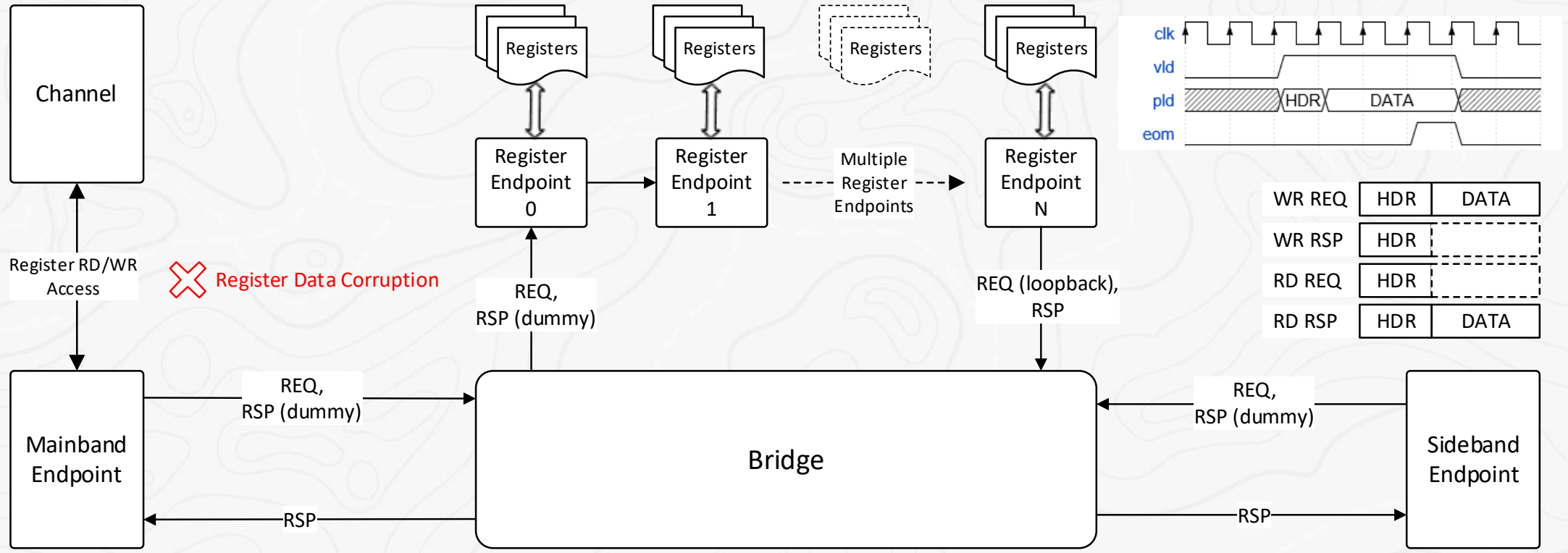
Not included in oral presentation

Case Study

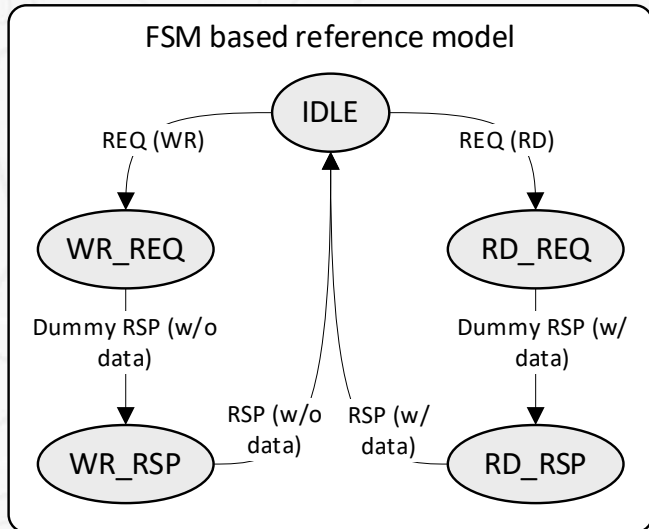
Post-silicon bug in Bridge

UNDERSTAND THE
PROBLEM & COLLECT
ALL THE COLLATERAL

Bridge: Design & Bug Details



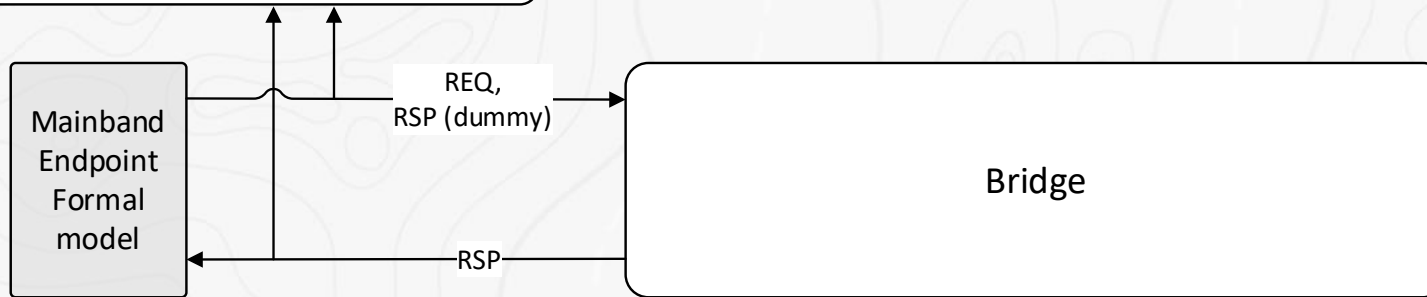
Bridge: Formal Property (Checker)



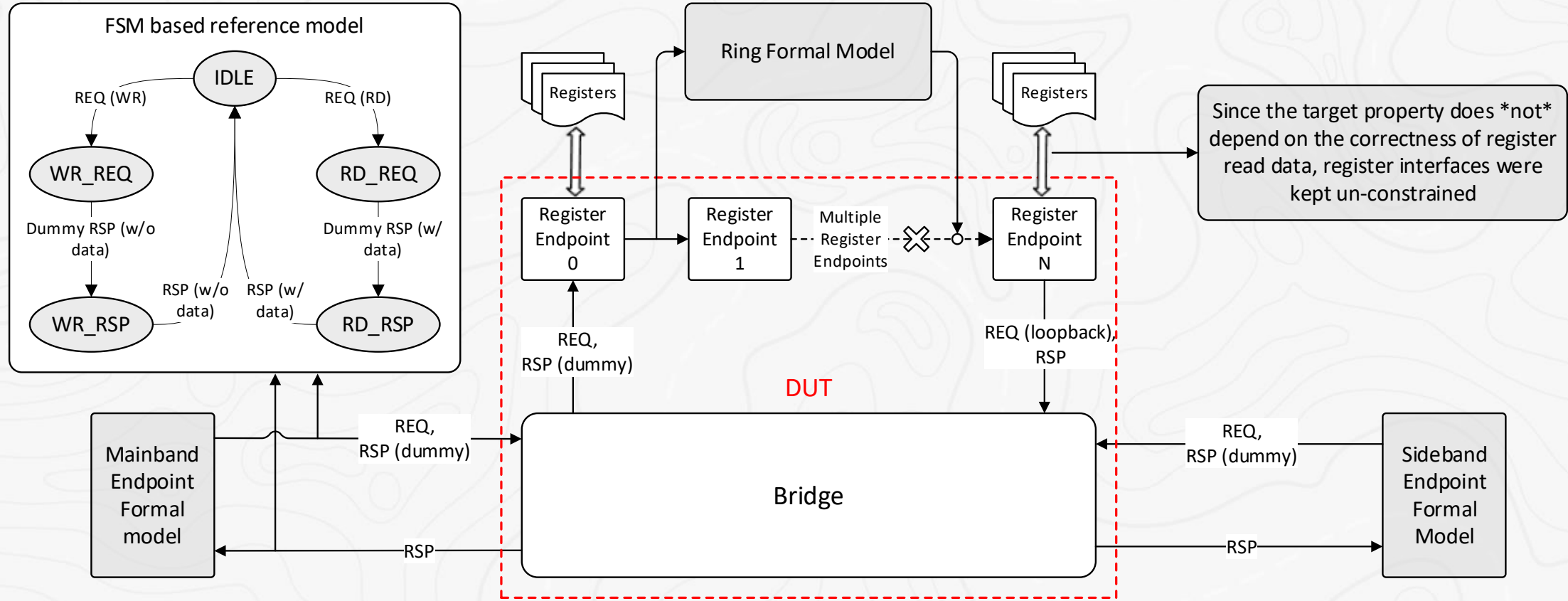
SVA Property

```

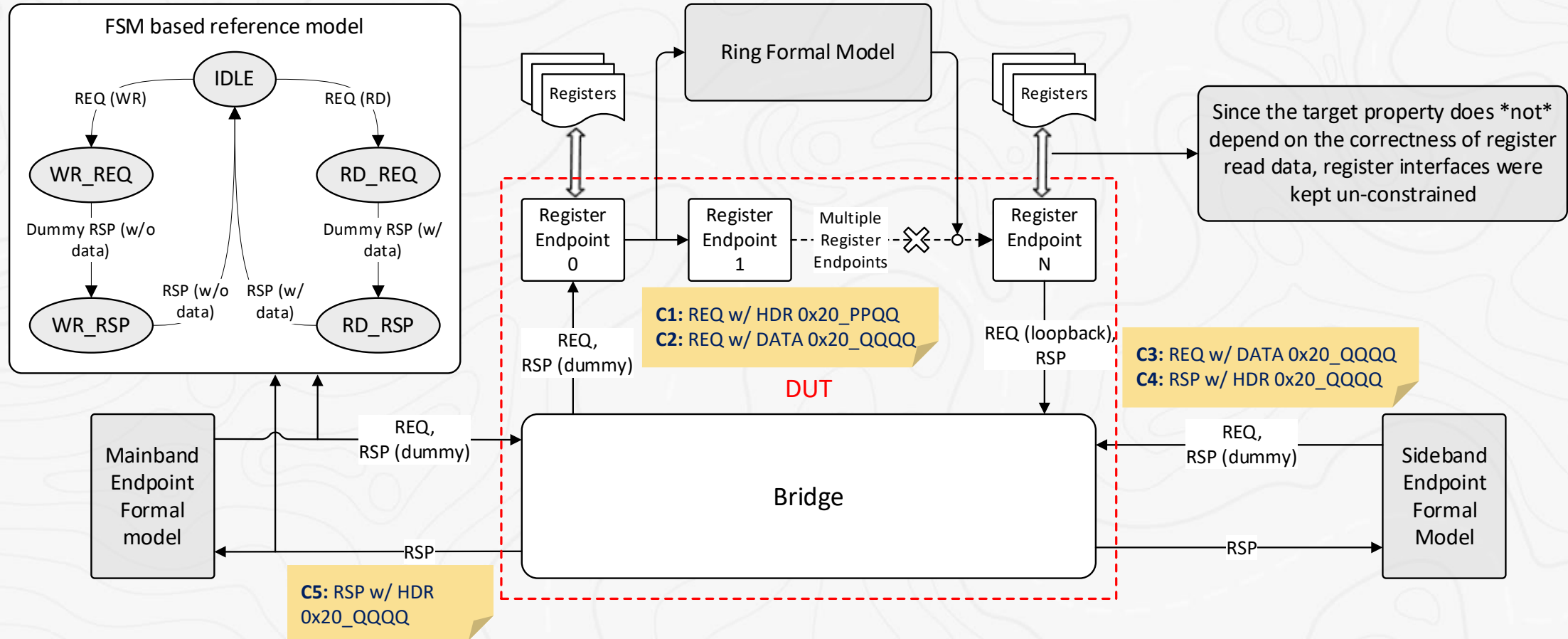
bridge_to_mainband_no_spurious_rsp: assert property (
  @(posedge clk) disable iff (rst)
  (state == RD_REQ) ||
  (state == WR_REQ) ||
  (state == IDLE) |-> !b2m_rsp_vld
);
  
```



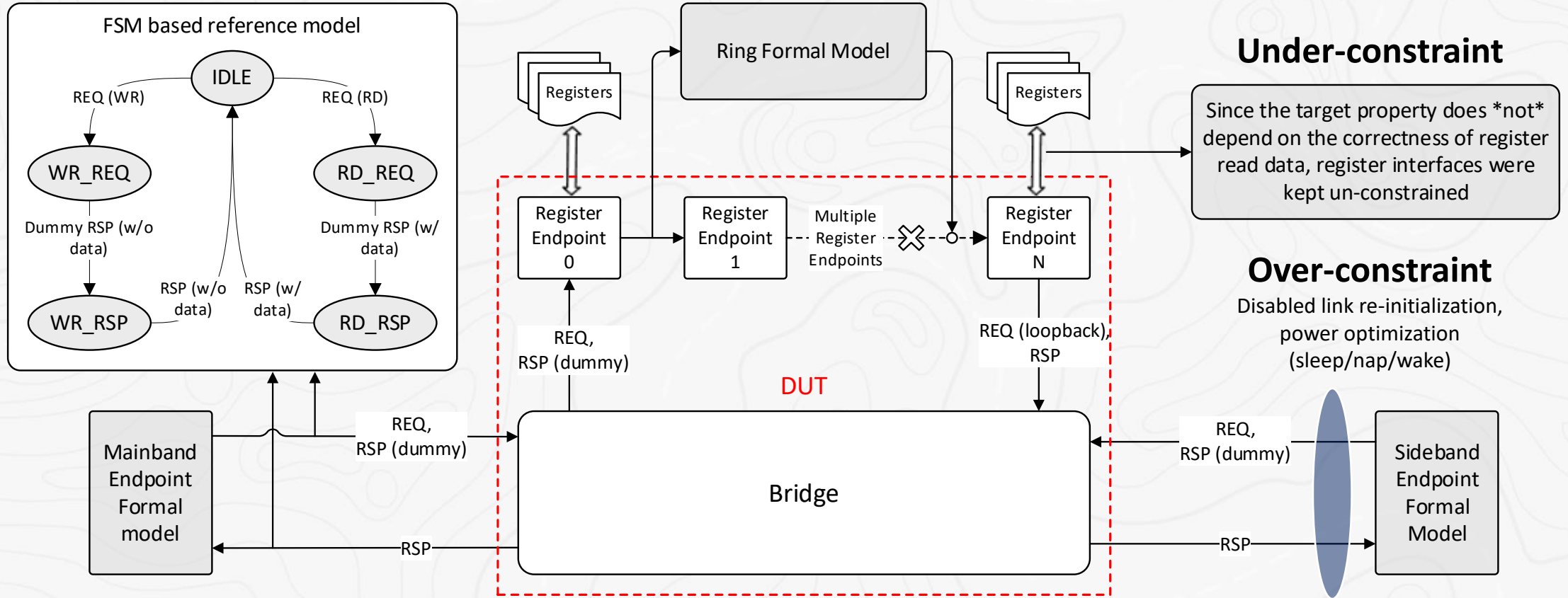
Bridge: DUT & FV Environment



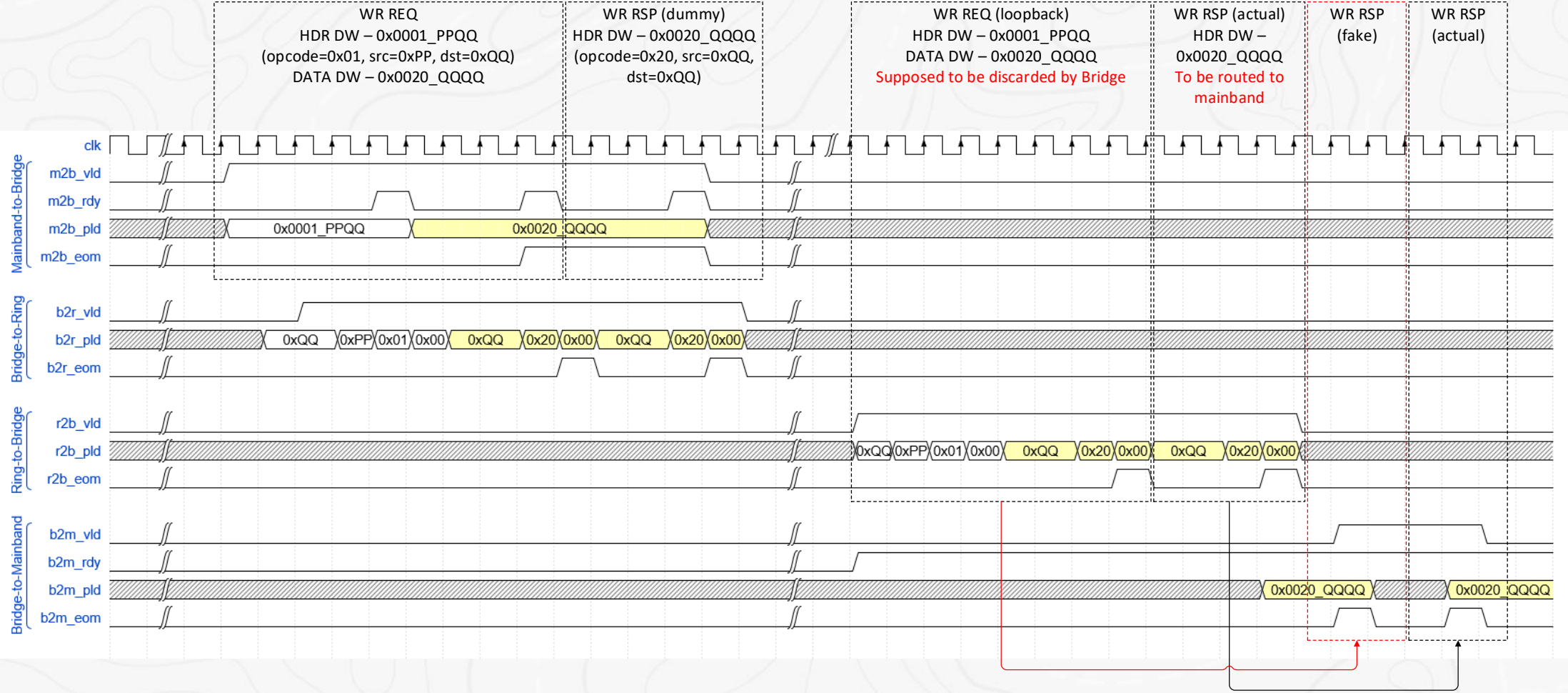
Bridge: Helper Covers



Bridge: Constraints Strategy



Bridge: Bug Repro (1 of 6 manifestations)



Bridge: Results

Post-silicon bug was reproduced in 3 engineer days weeks on 25K Gates DUT

4 new bug manifestations were detected (not yet seen in Silicon)

FV environment helped evaluating workarounds and bug-fixes

Comprehensive FV environment was created for sign-off

FV environment reused in next 2 project; Found 8 new bugs in pre-Silicon

Key Takeaway: Verifying post-Silicon bug-fixes offers huge ROI