Assisting Fault Injection Simulations for Functional Safety Signoff using Formal

Pulicharla Ravindrareddy – Analog Devices

accellera SYSTEMS INITIATIVE™

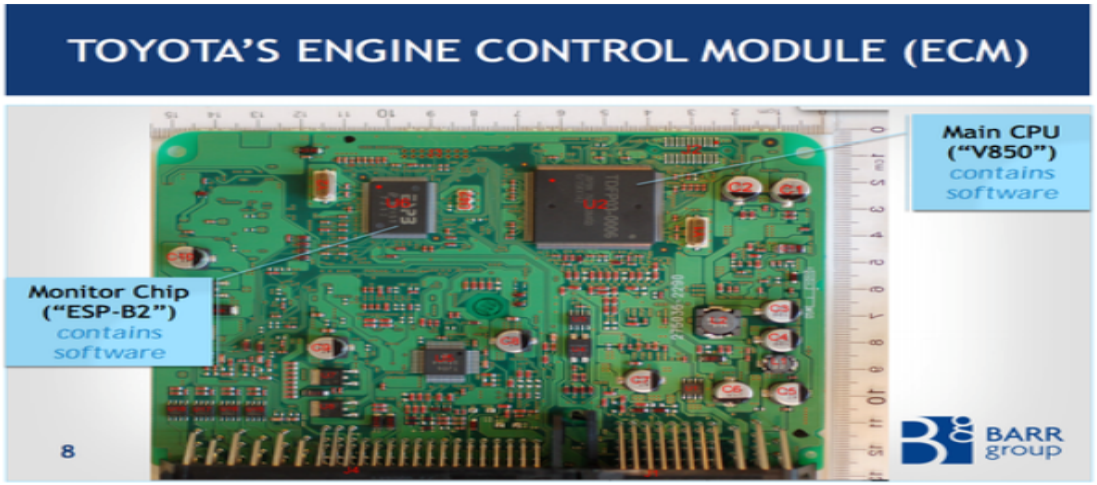
2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Agenda

- Introduction to Functional Safety
  - Fault injection
- Fault injection testbench
- Overview of DUT
- Problem and Motivation
- Formal Techniques
  - Formal fault testability analysis
    - COI analysis
    - Constant analysis
  - Formal fault detection analysis
- Efforts required and Challenges faced
- Results
- Acknowledgements

Introduction to Functional Safety

Unintended acceleration & electronic control unit

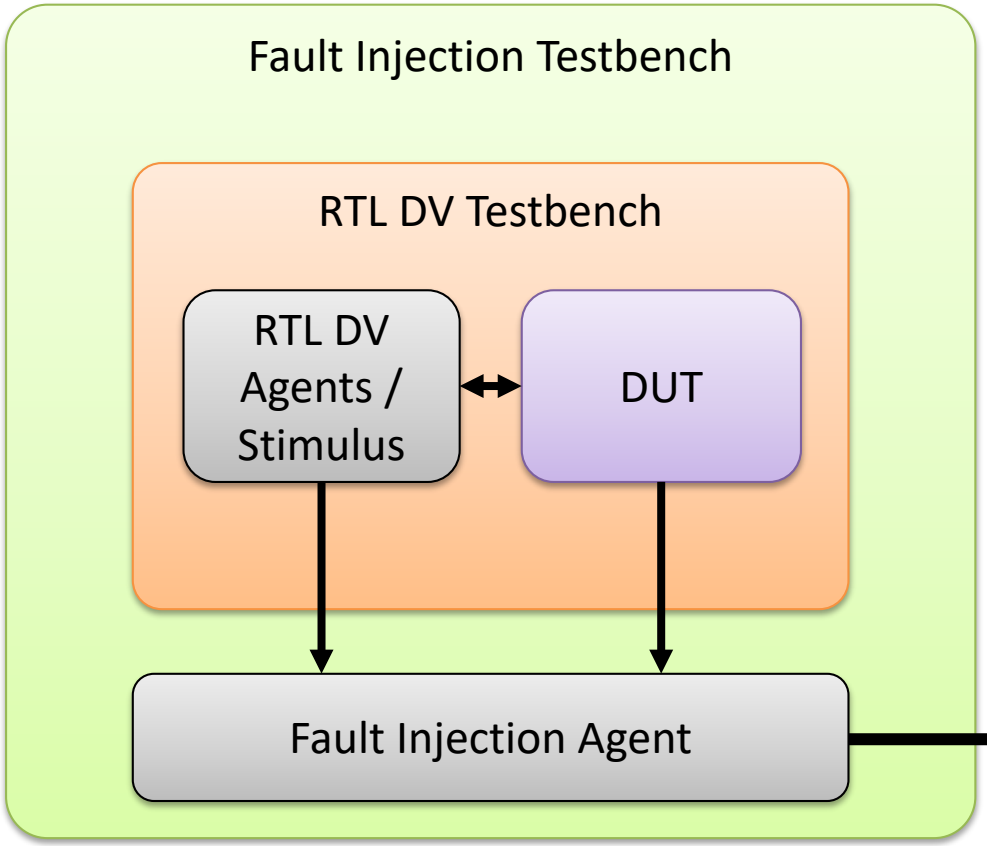


In a 2013 court case, embedded systems experts who reviewed Toyota's electronic throttle source code testified that they found the code defective, and that it contains bugs -- including bugs that can cause unintended acceleration.

"We've demonstrated how as little as a single bit flip can cause the driver to lose control of the engine speed in real cars due to software malfunction that is not reliably detected by any fail-safe," said Michael Barr, CTO and co-founder of Barr Group.

- Functional safety refers to the concept that an overall system will remain dependable and function as intended even in the event of an unplanned or unexpected occurrence.

Fault Injection Testbench

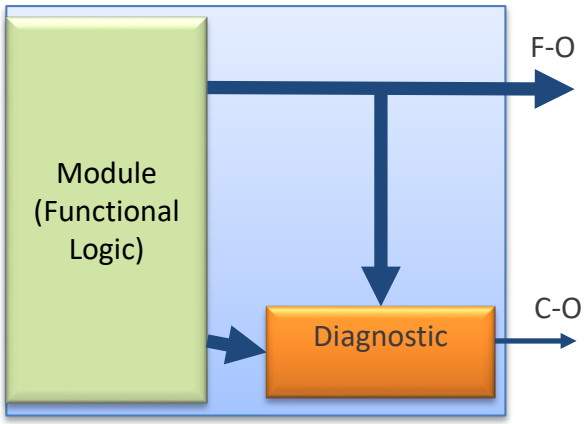


- ✓ Safety Mechanisms – ***FMEDA review.***
- ✓ Functional Outputs
- ✓ Checker Outputs

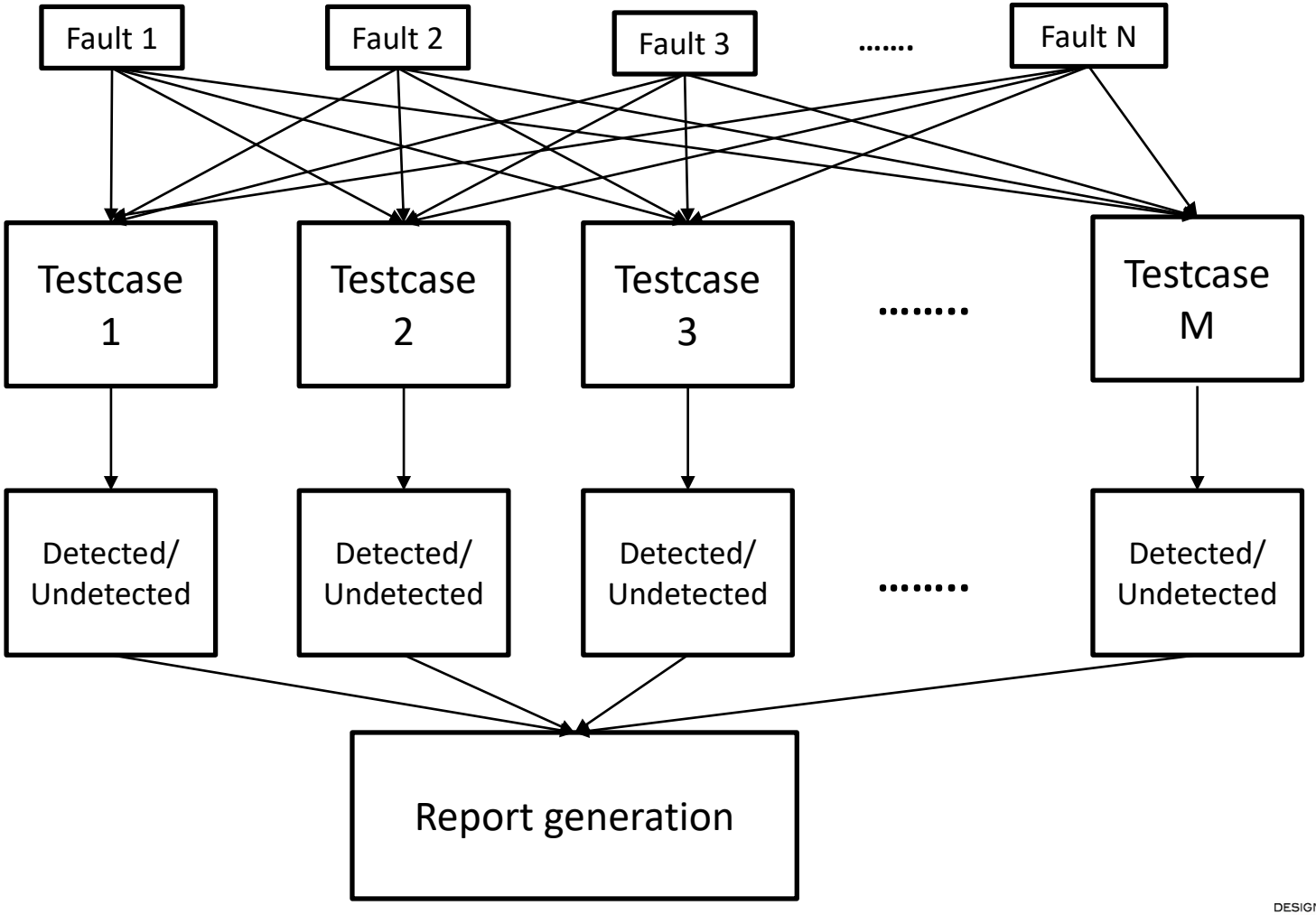
Strobes for FI runs.

Fault Injection Testbench

- Safety mechanisms:
  - Measures taken to ensure protection from danger
- Functional Outputs:
  - Functional output signal from fault injection testbench indicates violation of safety goal.
- Checker Outputs:
  - Checker outputs are made available to the host controller to detect faults in the SoC.

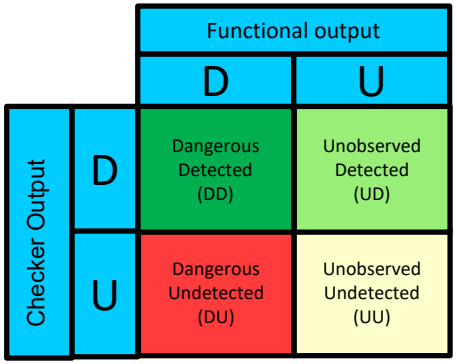


Conventional fault injection method



Fault classification

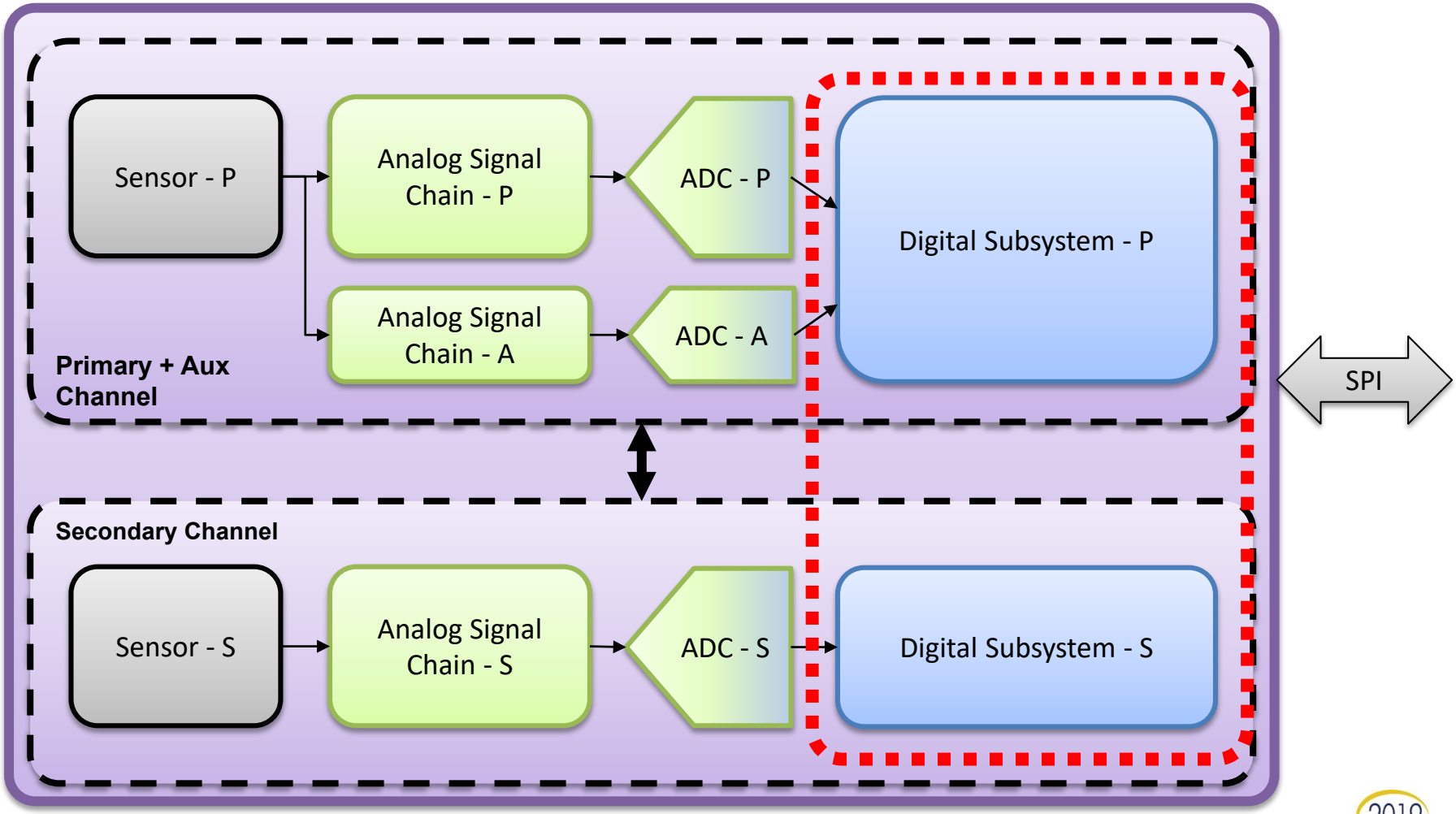
- When a fault is injected into a circuit, its effect can be classified into following 4 categories



- Using the above table, the diagnostic coverage (p) can be calculated as:

$$p = 1 - \frac{DU}{DD+UD+DU+UU}$$

DUT Overview



Problem and Motivation

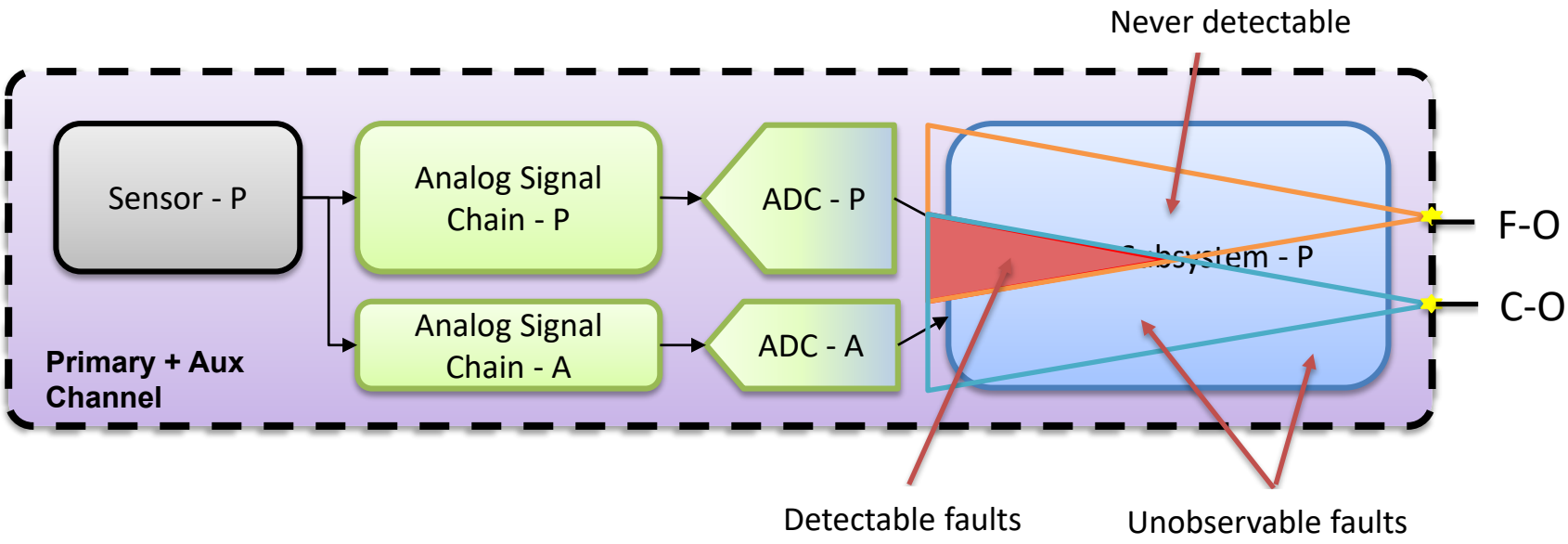
- Huge fault set
  - Complexity of the designs are increasing
  - Number of faults to be injected are huge
  - Simulators have in-built fault collapsing techniques
  - The number of fault nodes still need to be decreased or the number fault simulations are to be reduced.
  - Using Formal technology to reduce the fault set
- Debug difficulty
  - Difficult to prove safe undetected faults to be really safe
  - Take help of Formal Technology to ease the debug process

Formal Techniques

- Formal verification became part of routine design verification practices
- Formal techniques are mostly used for exercising assertions and bug hunting
- Apply Formal to the area of Functional Safety
- Different Formal techniques can be used to improve the efficiency of the fault injection campaign
  - Formal fault testability analysis
    - COI analysis
    - Constant analysis
  - Formal fault detection analysis

COI analysis

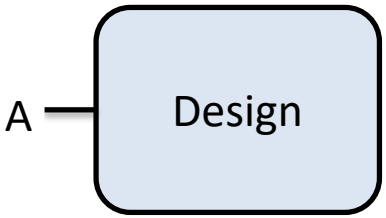
- COI analysis is used to find out the untestable faults in the design
  - Untestable faults: Faults that do not have physical connection to the safety critical elements



- Sometimes we might miss or add an extra functional or checker outputs
- This analysis helps also to confirm if the provided functional and checker outputs are proper

Constant Analysis

- Constant analysis is used to find out un-activatable faults
  - Un-activatable fault: An SA0 or SA1 fault injected on a node that is a constant 0 or 1 is predicted to be undetected by simulation. Thus, if a fault node is permanently driven to the injected fault value, the fault is un-activatable



- Consider a design with signal ‘A’, and in the formal setup signal ‘A’ is constrained to 1
  - Stuck-at 1 fault on ‘A’ is un-activatable

- Determines the un-activatable faults based on the provided constraints set.

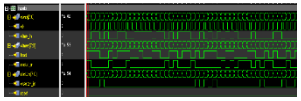
Formal Fault Detection Analysis

- Problems of Simulation:
  - Fault safety simulator covers the injected faults based on applied stimuli
  - Assuming a fault is not detected after the campaign, it is tedious for someone to think through a scenario that would exercise the fault so that it is propagated and detected.

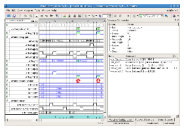
- Formal detection analysis
  - Tries to cover the faults with all possible stimuli
  - This can be used to know if any undetected fault can be detected or never detectable
  - Handy in debugging

Formal Detection Analysis – (After Simulations)

Simulation Regression Tests



Formal Analysis



Unobserved undetected (UU)
Zero tests show detection of fault

confirm

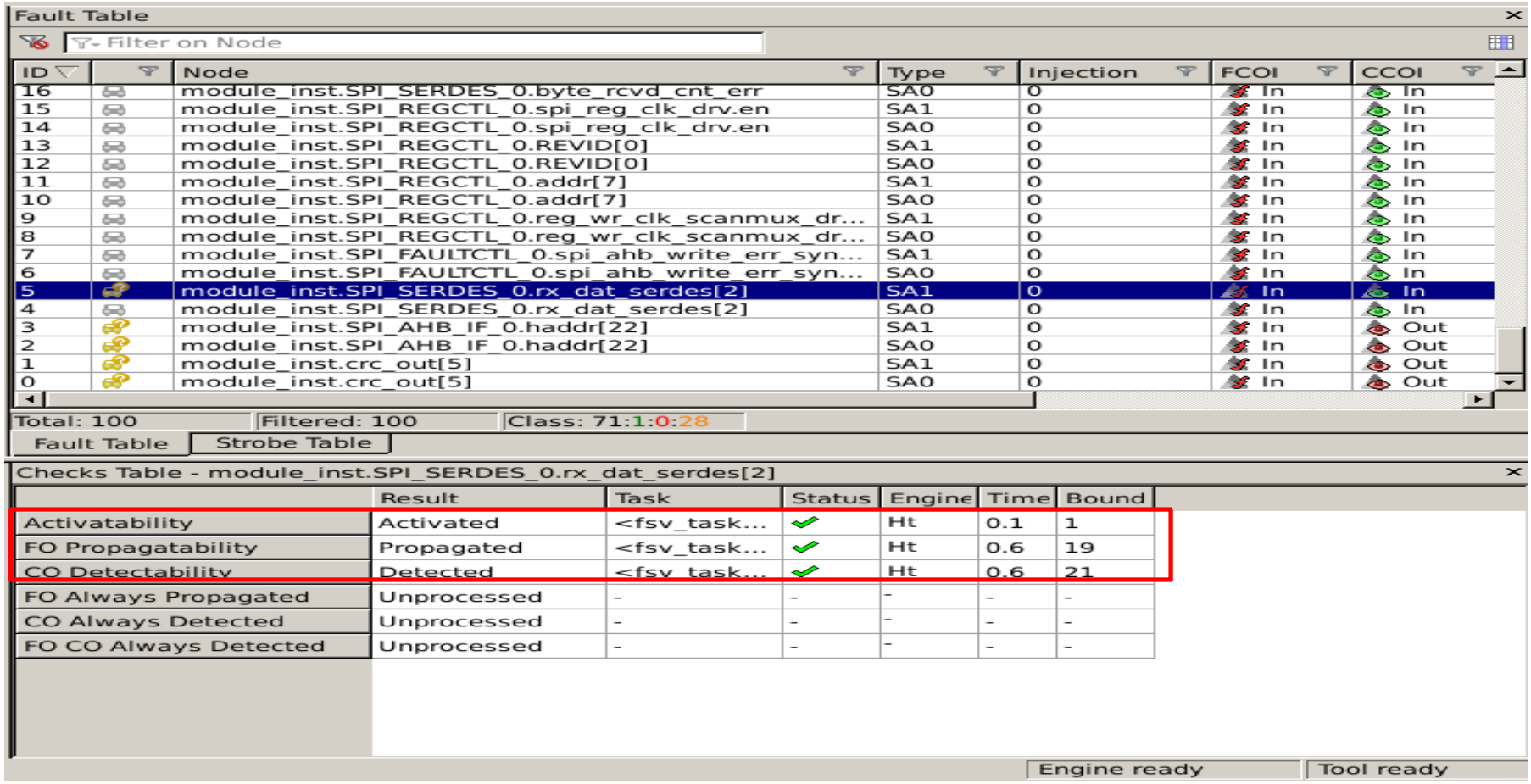
Safe
Fault is truly safe with no effects on safety goal.

invert

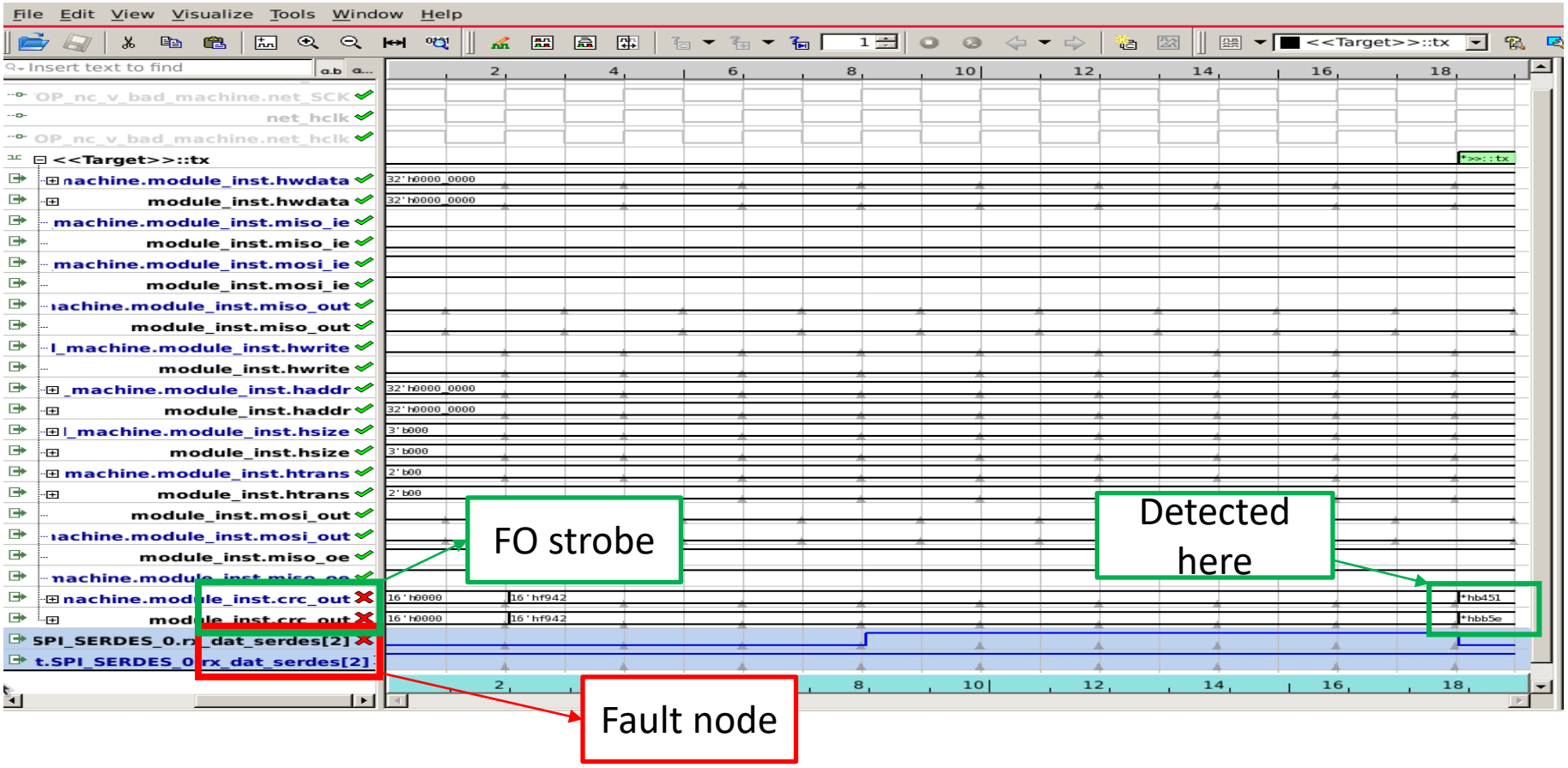
Dangerous
Atleast one stimulus is there to make the fault violate safety goal.

- Combine Formal and simulation to enhance the analysis quality

Formal Detection Analysis



Fault Propagatability Waveform



Efforts required

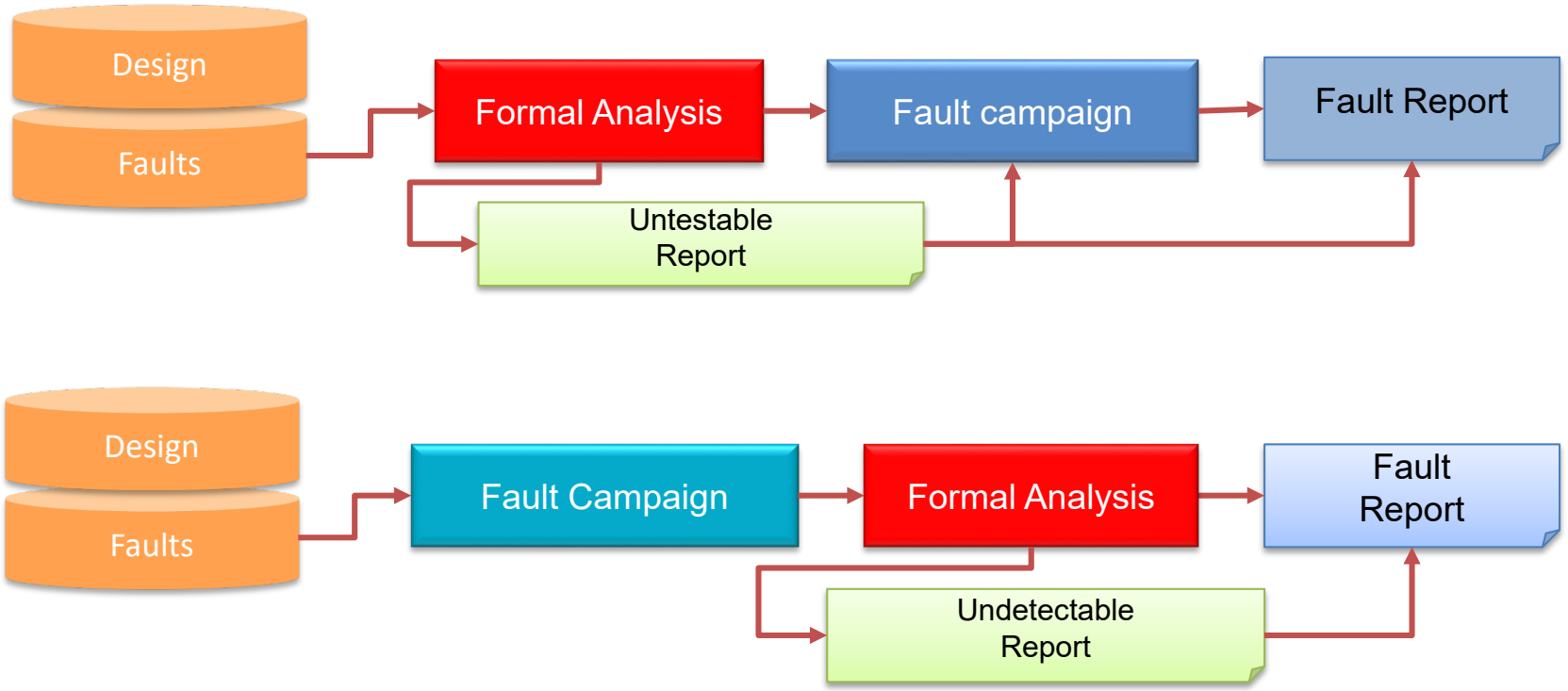
- Setup
  - Formal setup is almost similar for any application
  - Setup was already present and does not require much modifications
- Runtimes
  - Testability analysis: Structural analysis, very less runtime
  - Detectability analysis: Functional analysis, considerable runtimes

Challenges

- Two challenges
  - Designer/Verification Engineer needs to confirm that the tool exercised a valid scenario
  - DV Engineer needs to write a testcase for that scenario

- Approaches tried for writing a testcase
  - Save waveform in .vcd
  - Writing a directed/constrained random testcase covering that scenario

Formal placement in the fault injection flow



Results

- Formal testability analysis

	Total no. of Faults	Safe faults	Never detectable faults
Fault simulations	57234	9.19%	30.7%

- Formal detectability analysis

	FI Campaign by using Simulation	Reduction of UU faults with the application of Formal
Unobserved Undetected faults	20%	12%

Questions ?