Fault Proof: Using Formal Techniques for Safety Verification and Fault Analysis

Adrian Traskov
Thorsten Ehrenberg
Sacha Loitz

Abdelouahab Ayari
Avidan Efody
Joseph Hupcey III

accellera SYSTEMS INITIATIVE

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Outline

Introduction to Fault Injection

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Example

Safety Mechanism Validation Flow

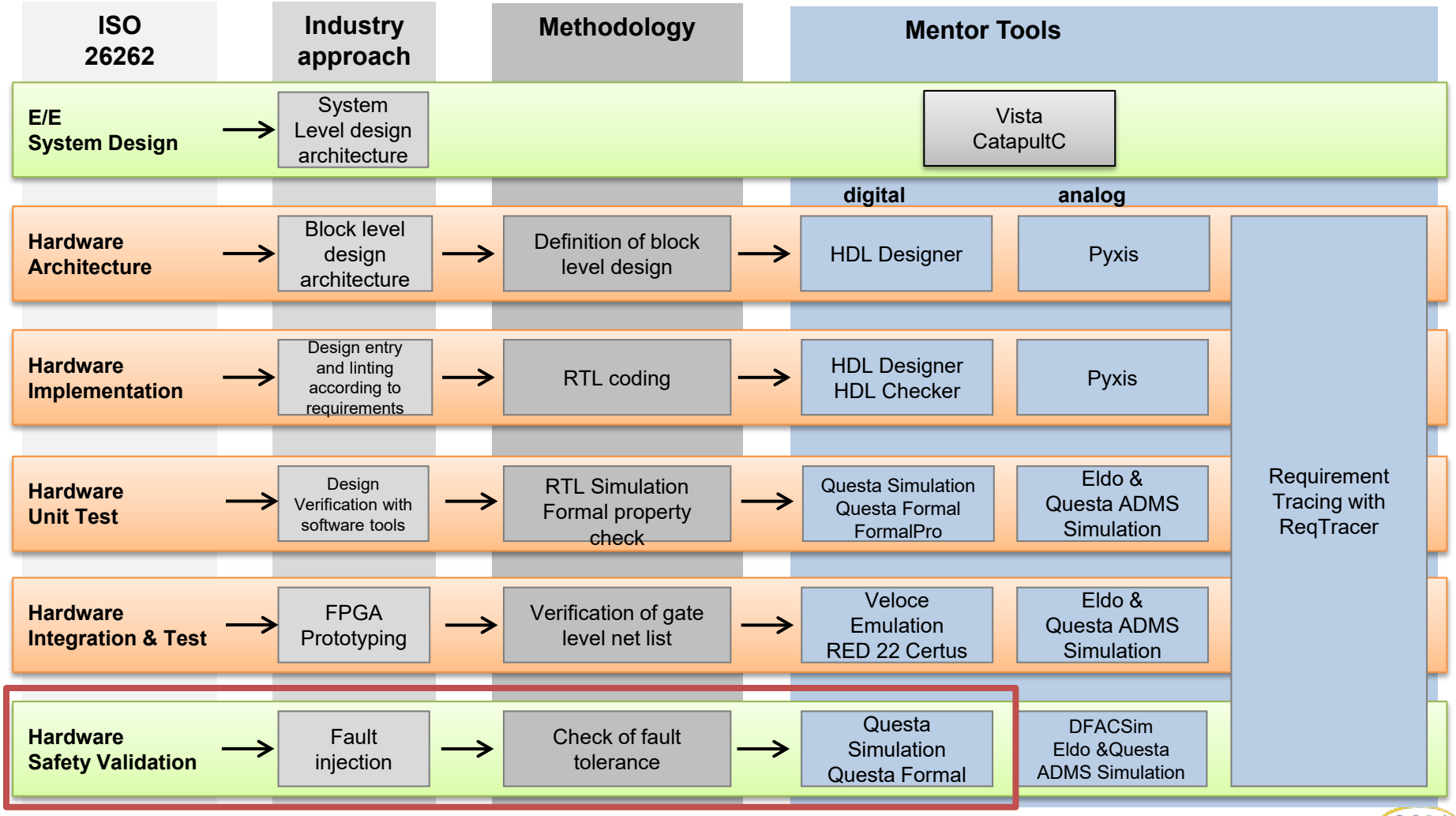
- Formal-based Fault Injection

Fault Coverage

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary

ISO26262 Safety Validation



Random Faults and Safety Mechanism (SM)

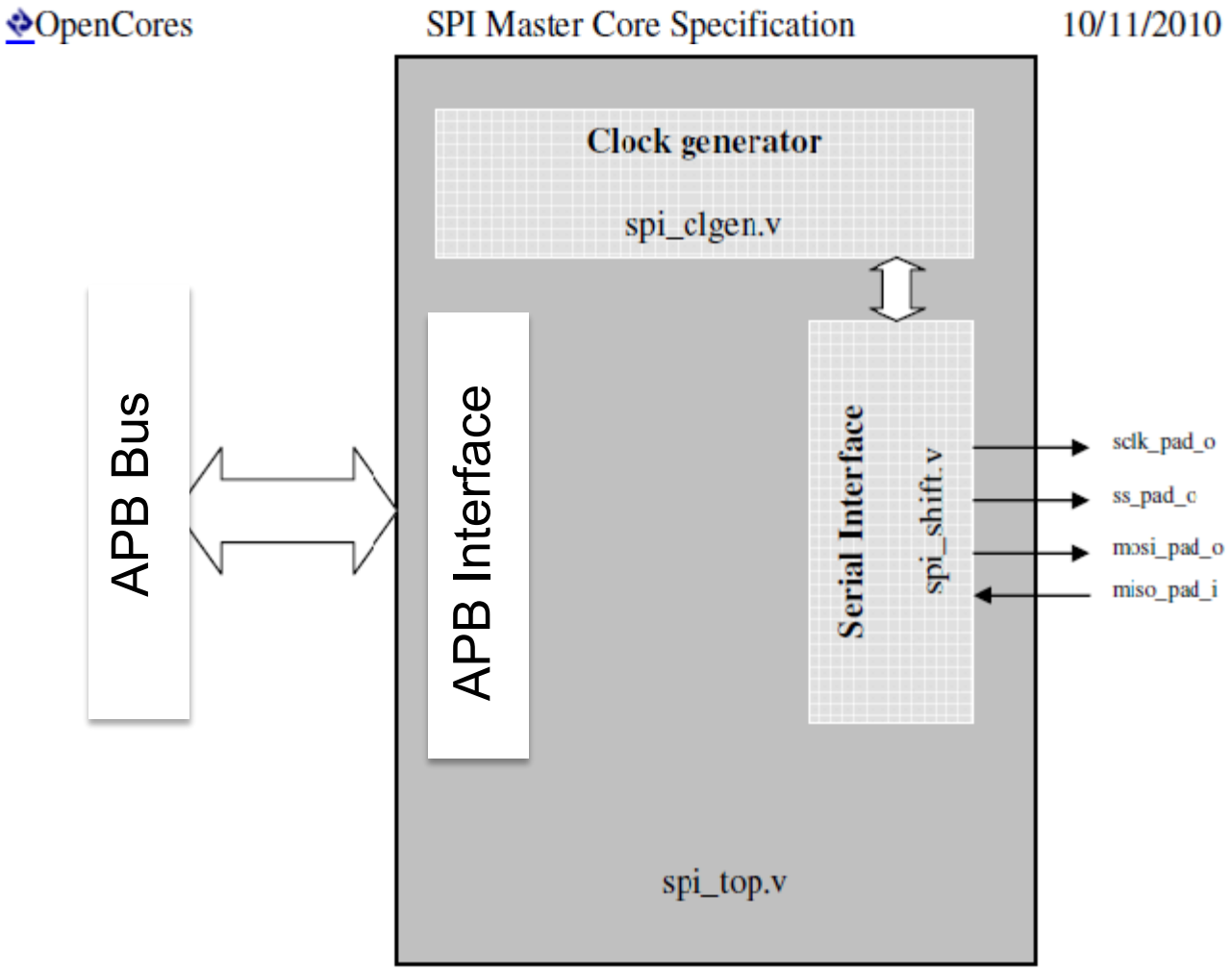
- Random Faults
  - ❑ physical defects that can occur in system components during system operation
- Purpose of Safety Mechanism
  - ❑ Control random faults
    - Detect all faults
    - Provide a deterministic and correct reaction to faults
  - ❑ Guarantee safety operation of the system
    - Recover the system, or
    - Go to a safe system state
- Validation/Verification of Safety Mechanism
  - ❑ Completeness
    - Check the ability to detect and handle all possible (important, resp.) faults
  - ❑ Correctness
    - Check that the safety mechanism specification/requirements are satisfied
    - For example:
      - Check design behaves as without presence of faults only for a specified list of faults.
      - Check design goes to a safe state for a specified list of faults

Examples of Safety Mechanism

- FSM safe encoding
- Triple Modular Redundancy (TMR)
- Error-correction code (ECC)
- Lock-step

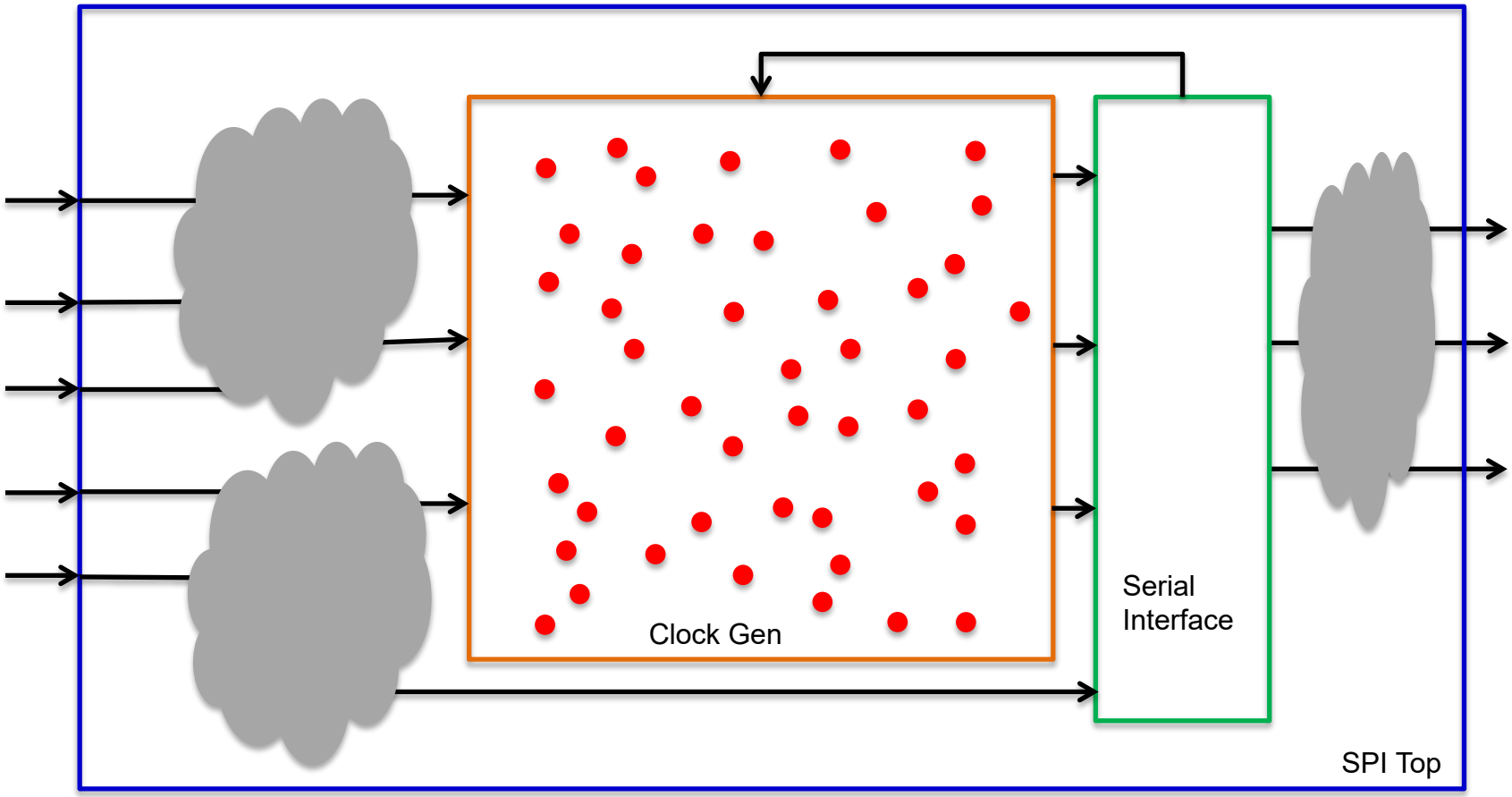
Safety Mechanism:

Illustration Using SPI Master Core Example



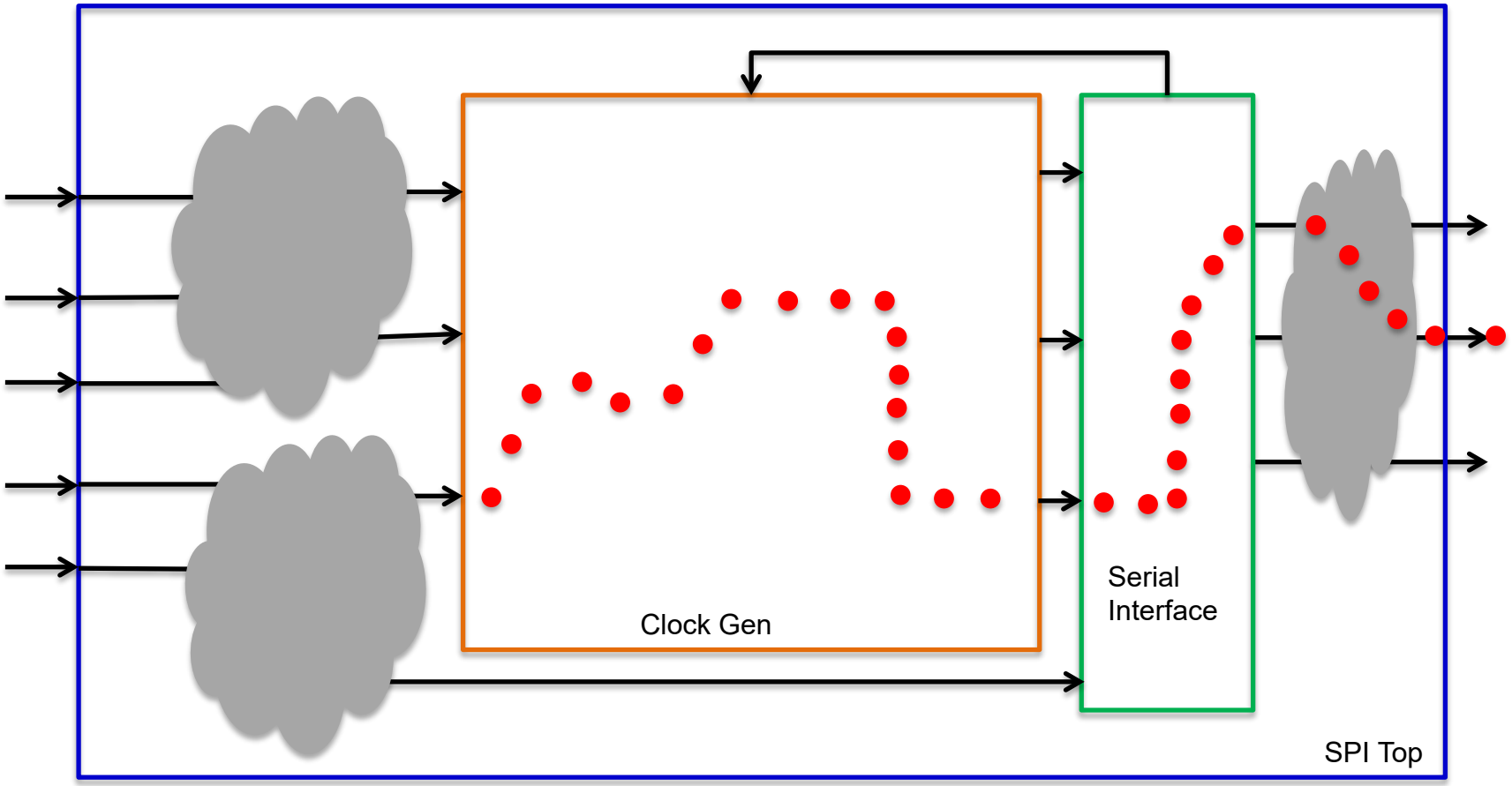
SPI Master Core:

Assume Faults Occurs in Clock Generator

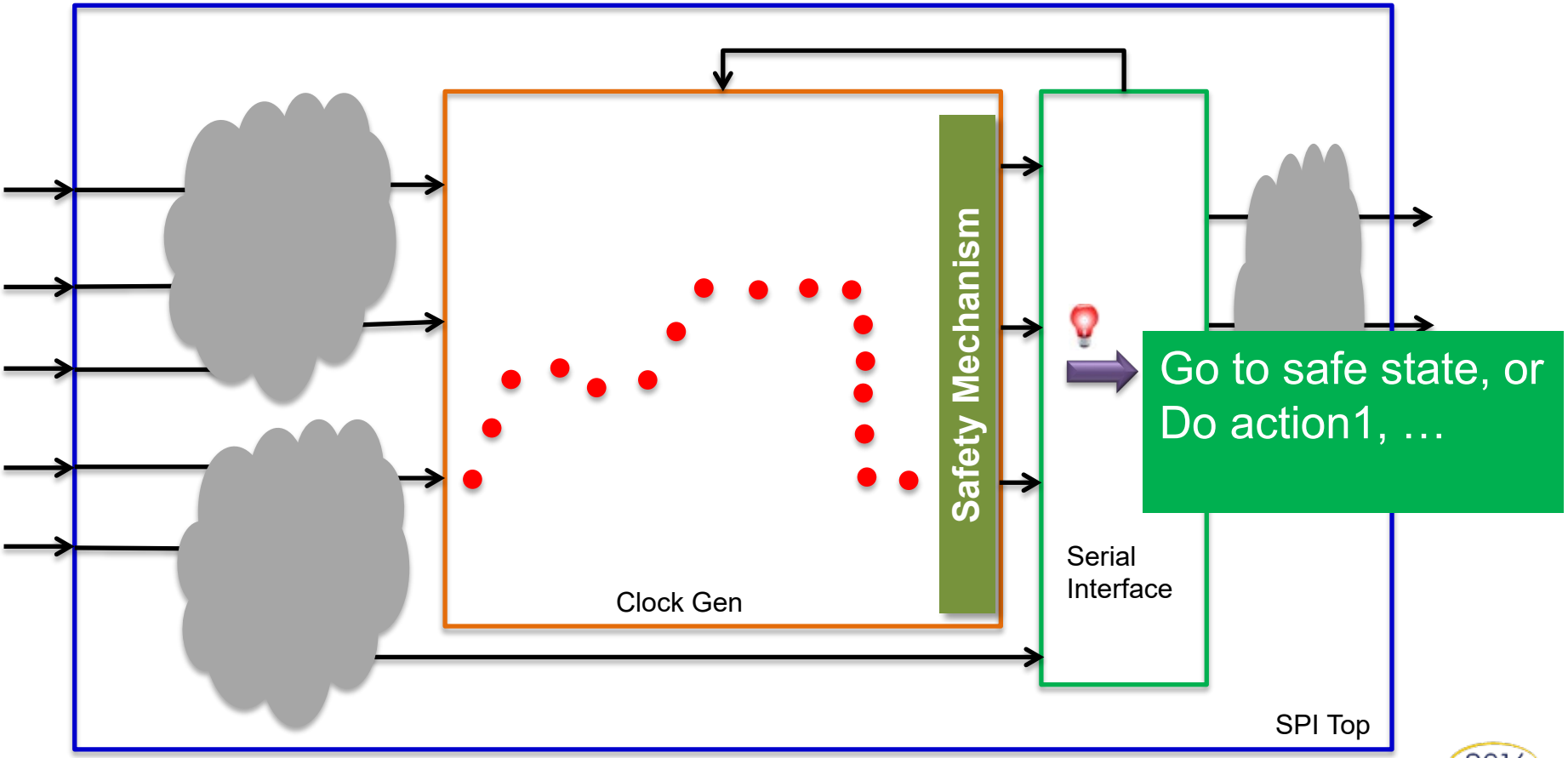


SPI Master Core:

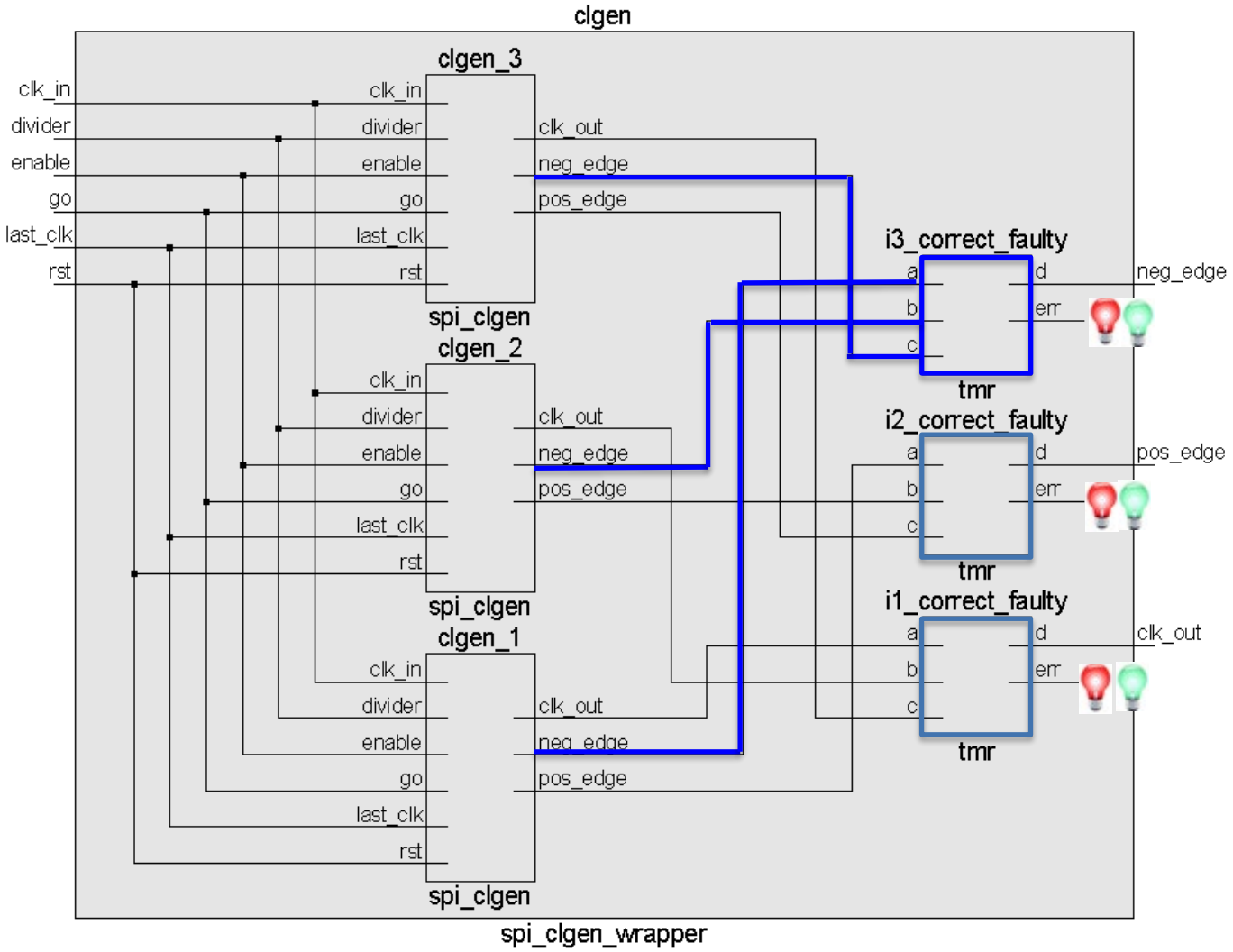
Faults Could Affect Functional Safety



SPI Master Core: *Fail-Operational*
Safety Mechanism Handel Faults



Safety Mechanism for Clock Generator



Outline

Introduction to Fault Injection

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Examples

Safety Mechanism Validation Flow

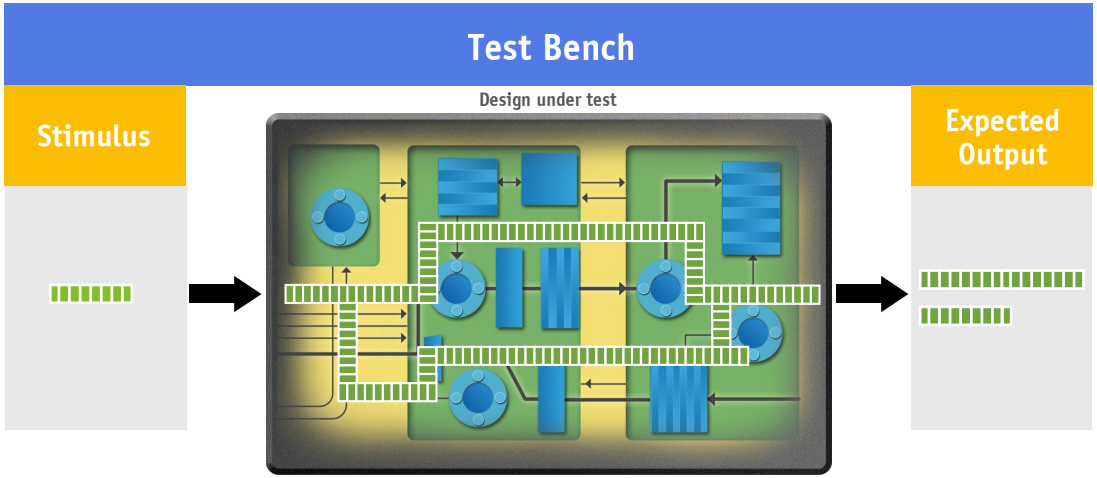
- Formal-based Fault Injection

Fault Coverage

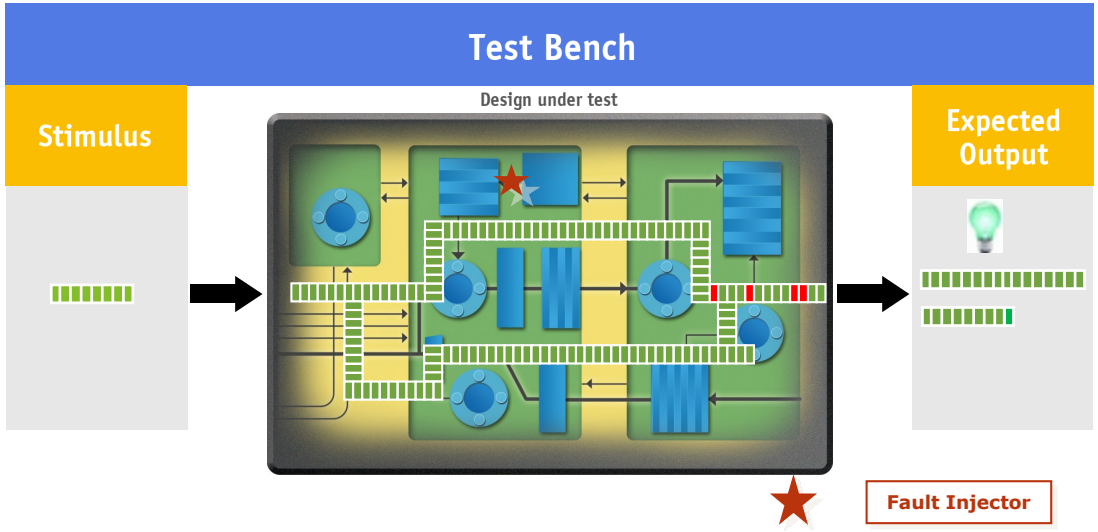
- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary

Regular Simulation

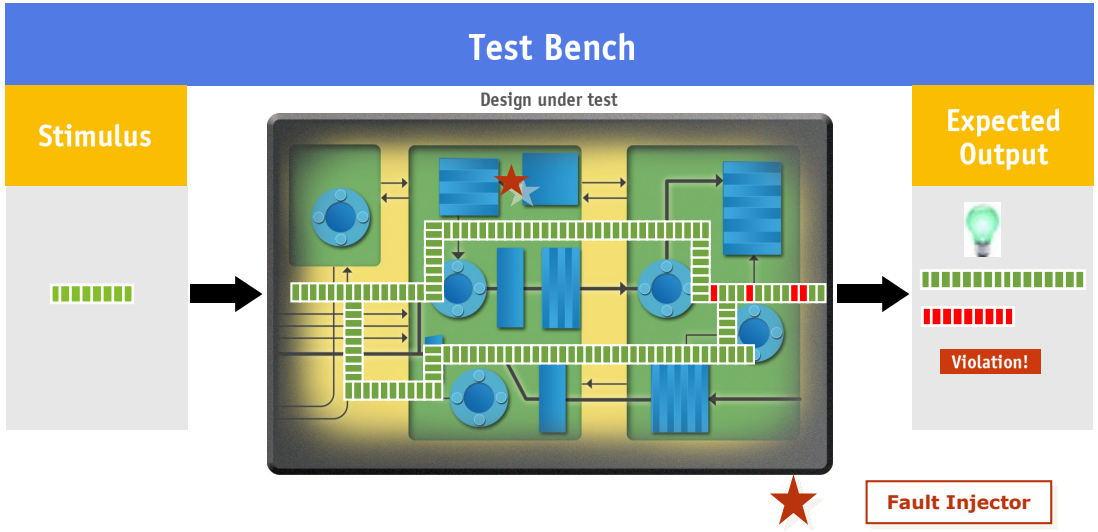


Fault Simulation:
Fault Detected & Corrected



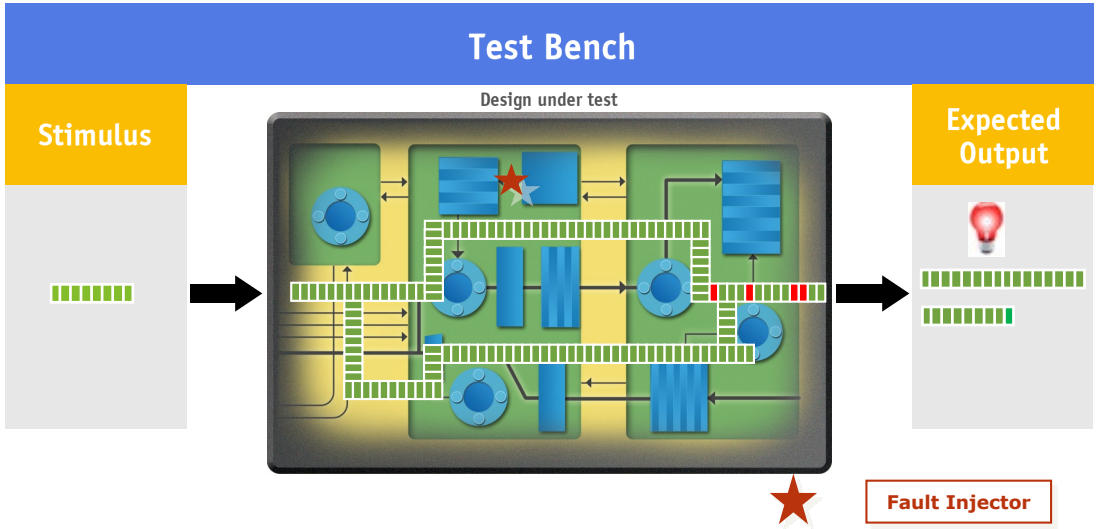
Fault Simulation:

Fault Detected & Not Corrected



Fault Simulation

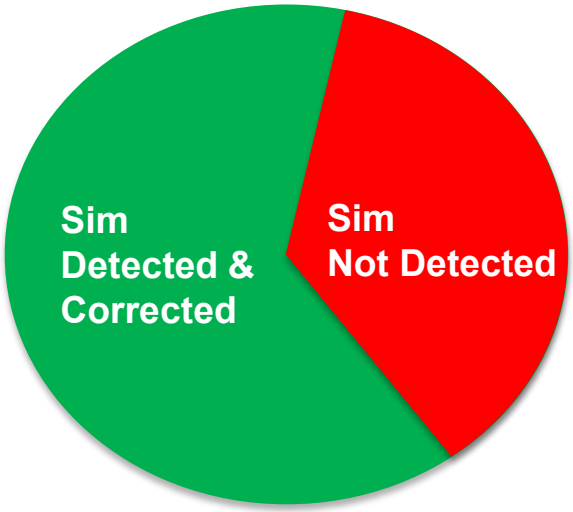
Fault Not Detected



- The fault is not detectable in this stimulus
  - Is the fault not detectable at all?
  - How to decide this?
  - If it is detectable, then which stimulus can be used for detection?

Fault Simulation Regression Results

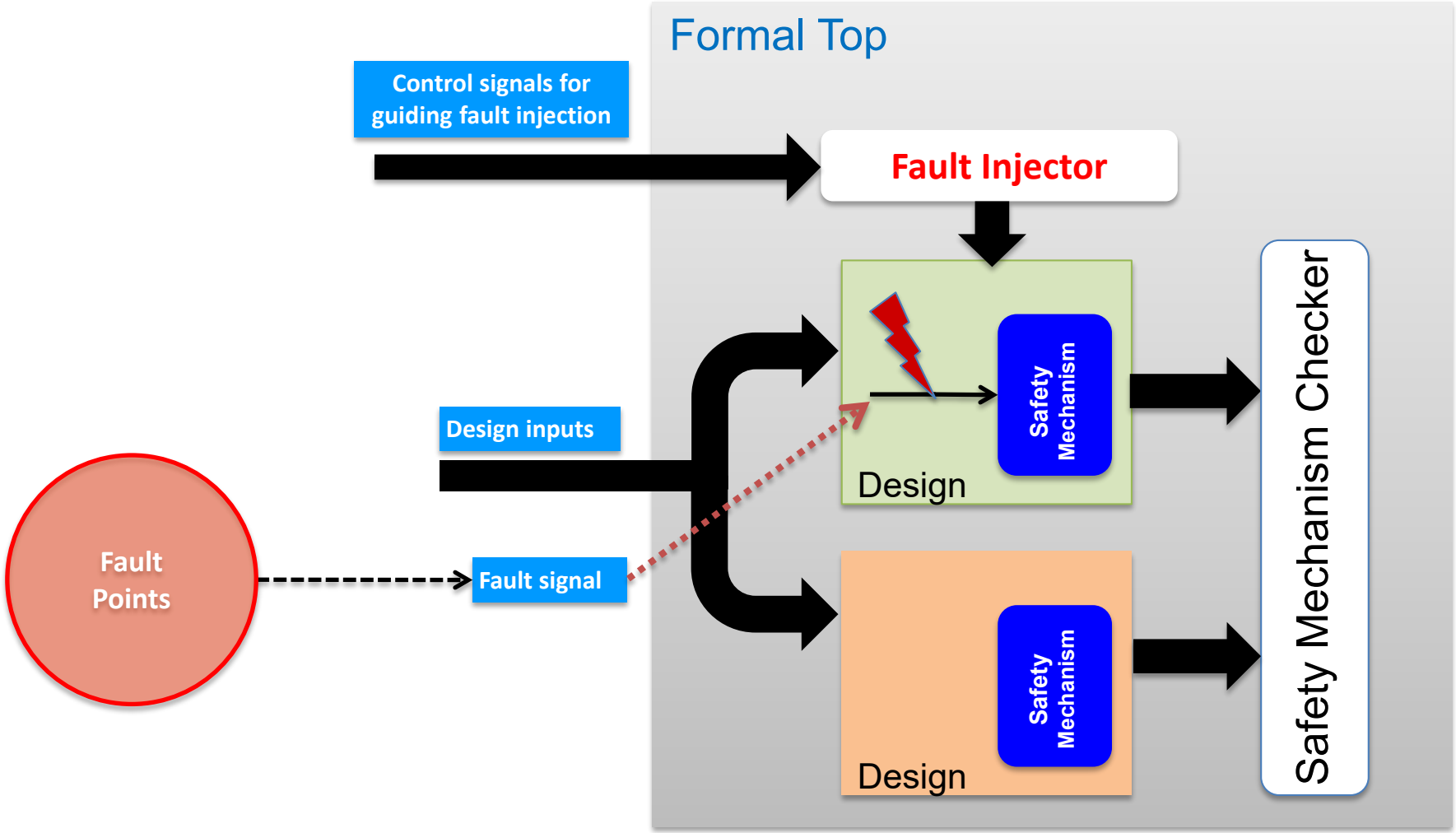
- Regression using Questa EVP
  - Number of Tests **372**
    - 4 Test cases (UVM)
    - Fault Models: Stuck-at-1
    - 372 Faults
  - Results
    - Non-Propagatable **67%**
  - Run Time **2H 16min**



Verification Management Tracker

Sec#	Testplan Section / Coverage Link	Type	Goal	Coverage	% of Goal	Status
0	testplan	Testplan	-	67.29%	67.29%	
1	SPI Fault Verification stuck_at_...	Testplan	100%	67.29%	67.29%	
1.1	Fault 1 detected	Testplan	100%	0%	0%	
1.2	Fault 2 detected	Testplan	100%	100%	100%	
1.3	Fault 3 detected	Testplan	100%	100%	100%	
1.4	Fault 4 detected	Testplan	100%	100%	100%	
1.5	Fault 5 detected	Testplan	100%	100%	100%	
1.6	Fault 6 detected	Testplan	100%	100%	100%	
1.7	Fault 7 detected	Testplan	100%	100%	100%	
1.8	Fault 8 detected	Testplan	100%	100%	100%	

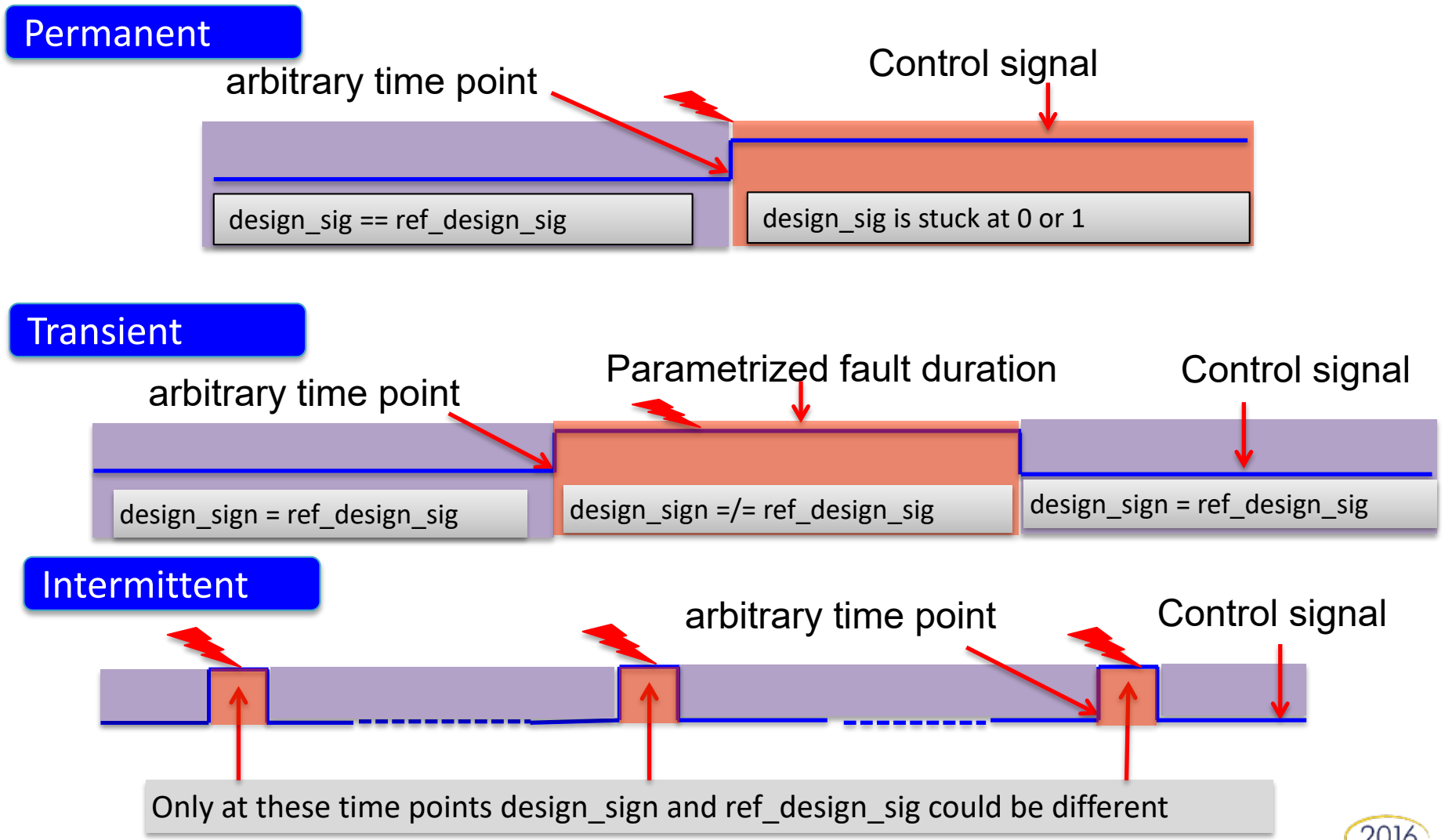
Questa Formal Model for Fault Injection



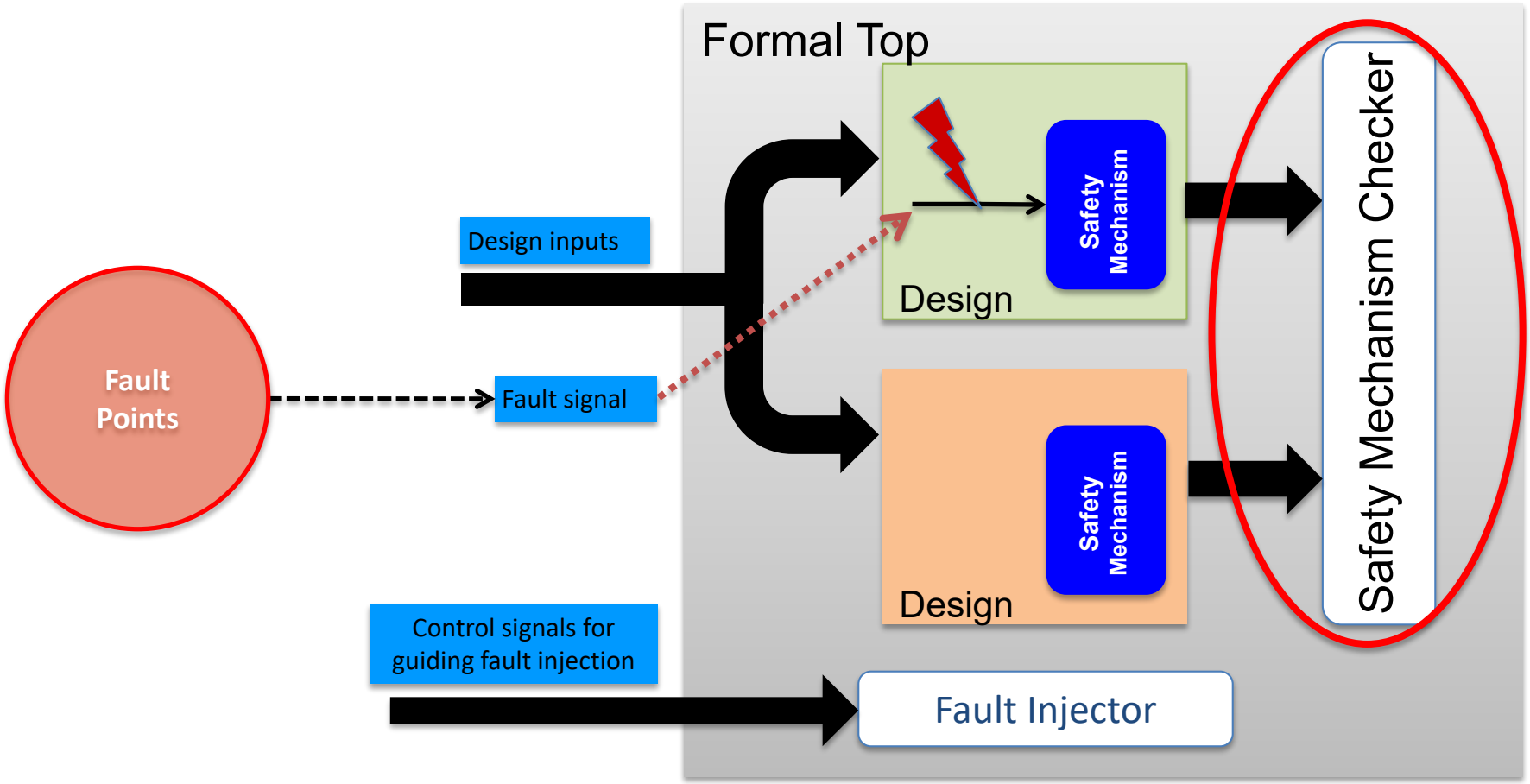
Fault Models

1. **Permanent Faults** (Stuck at 0, Stuck at 1)
   - Irreversible component damage
2. **Transient Faults** (a.k.a. soft-errors, SEU and SET)
   - Environmental Conditions
   - Cause Erroneous States in the system
   - Do not cause permanent damage
   - Hardest to detect
3. **Intermittent Faults**
   - Caused by unstable HW
   - Often become **permanent** faults after a period of time

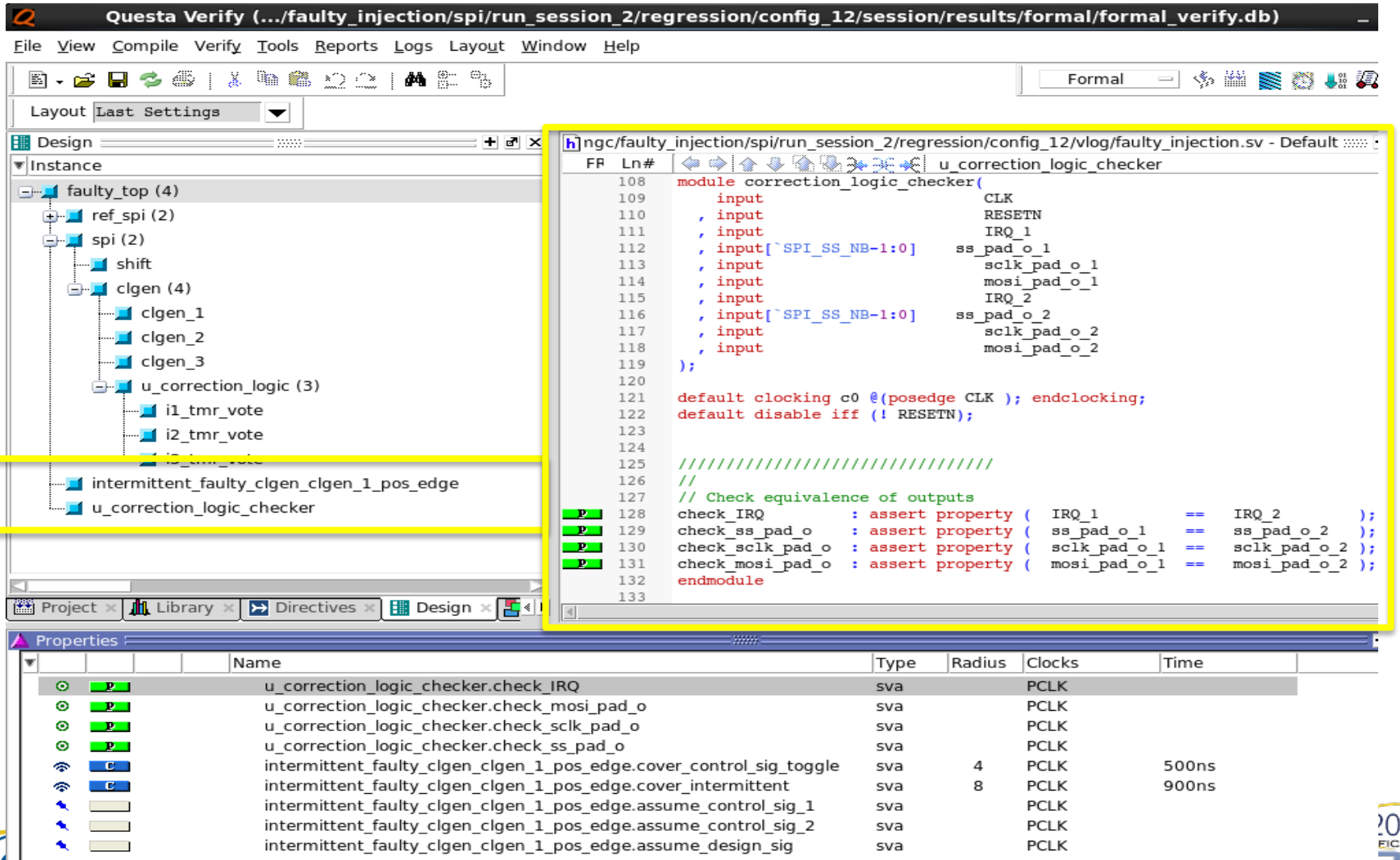
Modelling Faults in Formal



Formal Model for Fault Injection



Safety Mechanism Checkers



Outline

Introduction to Fault Injection

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Examples

Safety Mechanism Validation Flow

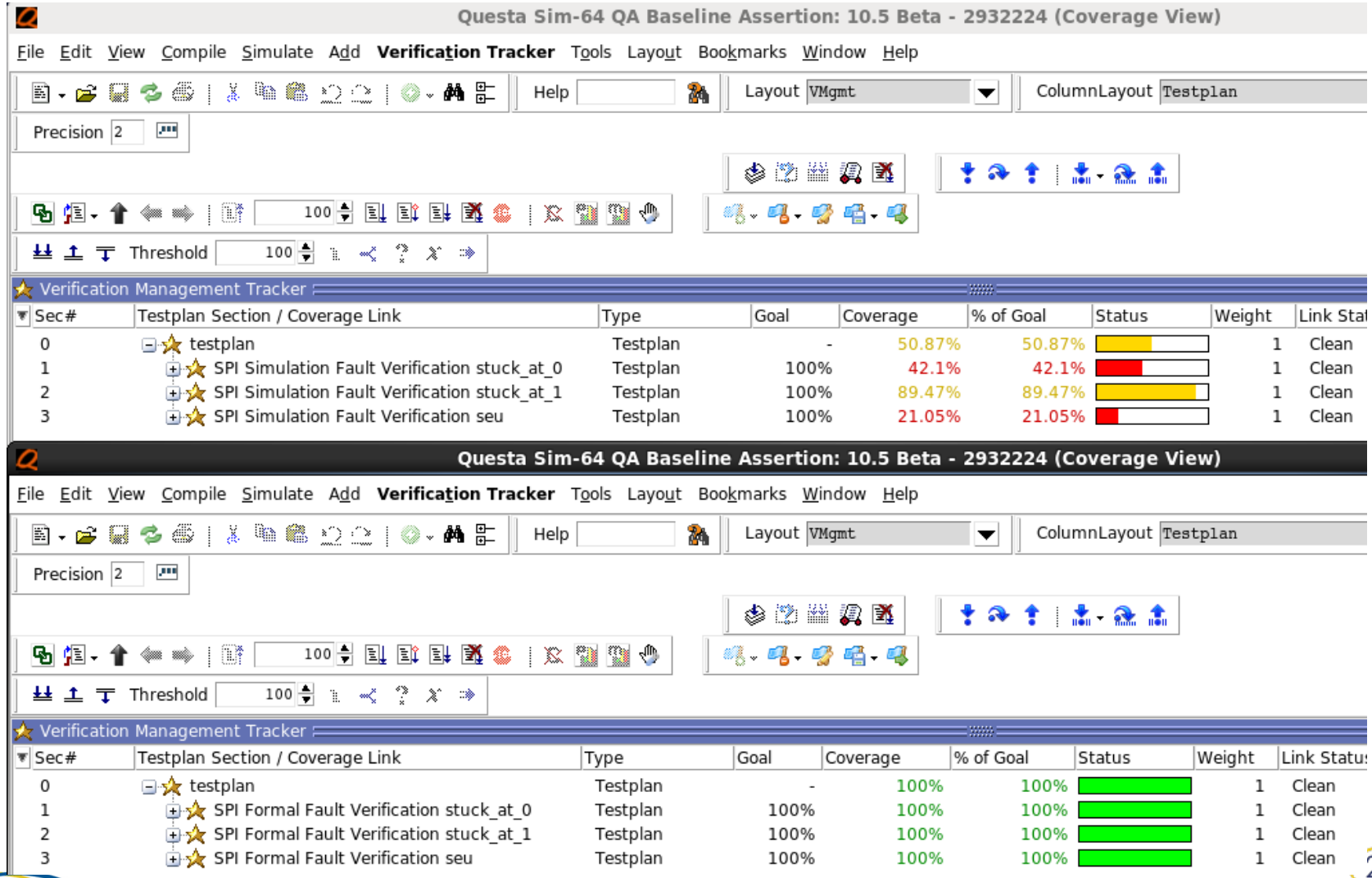
- Formal-based Fault Injection

Fault Coverage

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary

SPI Master Core Fault Coverage Results



Outline

Introduction to Fault Injection

- Fault Injection Part of ISO26262
- Safety Mechanism
- Illustration Through Examples

Safety Mechanism Validation Flow

- Formal-based Fault Injection

Fault Coverage

- Tracking Coverage
- Merging Coverage Results from Different Analysis Solutions

Summary

Summary

- Functional safety critical components are often small enough to be analyzed using formal techniques
- Formal fault injection is complete regarding legal design input pattern AND failure time points
- Still some work is need to compile formal pass/fail information to fault coverage.

Questions