Formal Fault Propagation Analysis that Scales to Modern Automotive SoCs

Sergio Marchese - Jörg Grosse

www.onespin.com

Introduction to Functional Safety

The objective of functional safety:

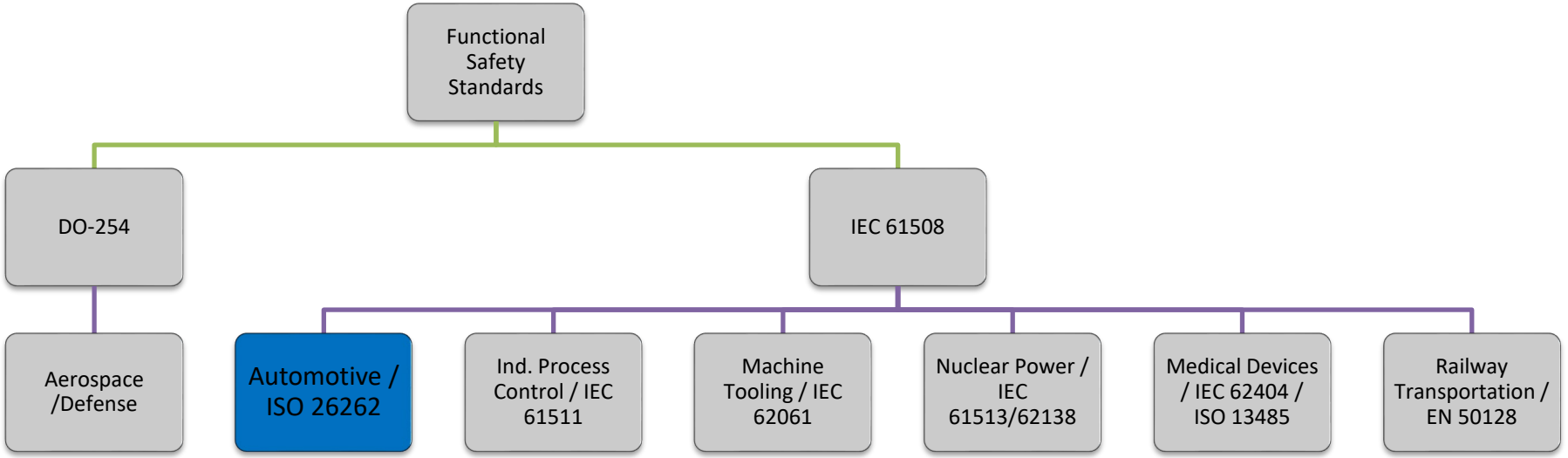
Freedom from unacceptable risk of physical injury or of damage to the health of people either directly or indirectly

Functional safety risks	Risk drivers	Risk management through functional safety standards

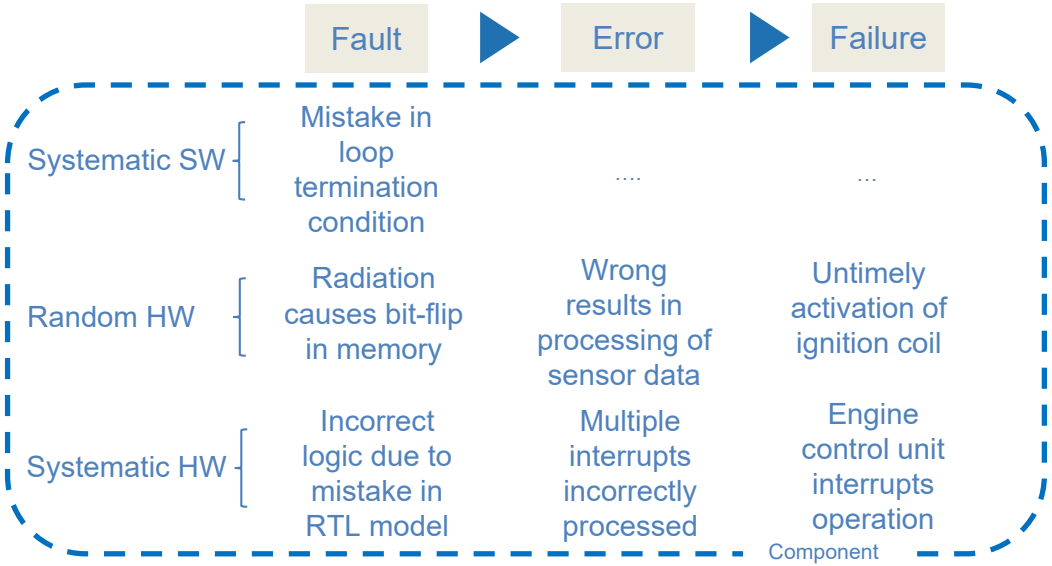
• Systematic Failures
  ○ Design errors
  ○ Tool errors
• Random failures
  ○ Hard errors
  ○ Soft errors
	• Continuous increase in flow and tool complexity
	• Continuous increase in functionality
	• Increasing density of the design process node
	• Decreasing energy levels
		• Minimize systematic errors
		• Safeguard against random errors

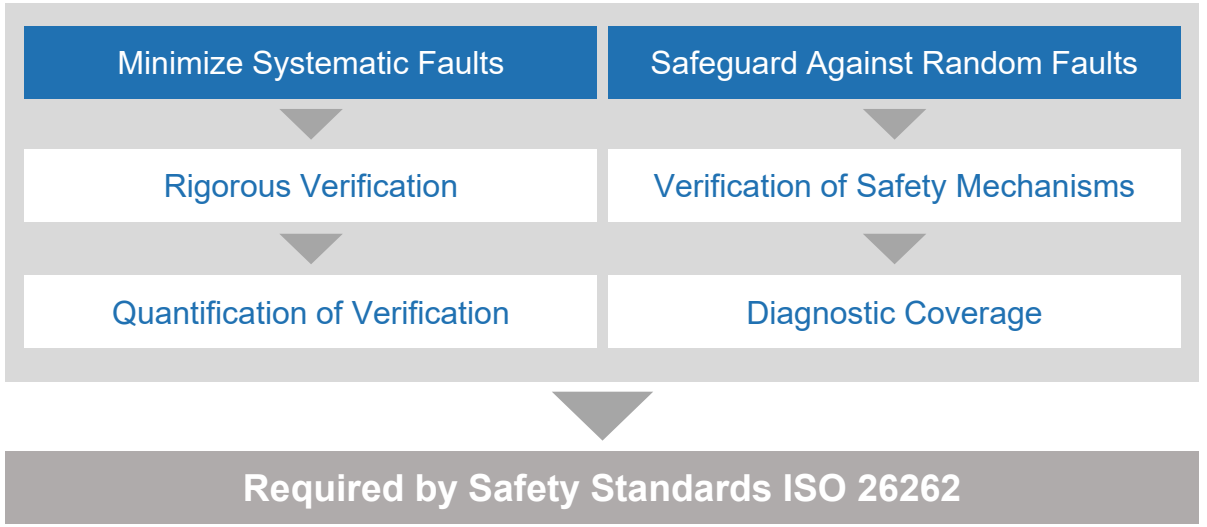
Functional Safety Standards

ISO 26262 covers E/E systems for road vehicles



Systematic and Random HW Faults





Focus: Safety Mechanisms and Diagnostic Coverage

Single Point Fault Metric

Safe faults

- Not in safety relevant parts of the logic
- In safety relevant logic but unable to impact the design function (cannot violate a safety goal)

Single point faults

- Dangerous, can violate the safety goal and no safety mechanism

Residual faults

- Dangerous, can violate the safety goal and escape the safety mechanism

Multipoint faults

- Can violate the safety goal but are observed by a safety mechanism
- Sub-classified as “detected”, “perceived” or “latent”

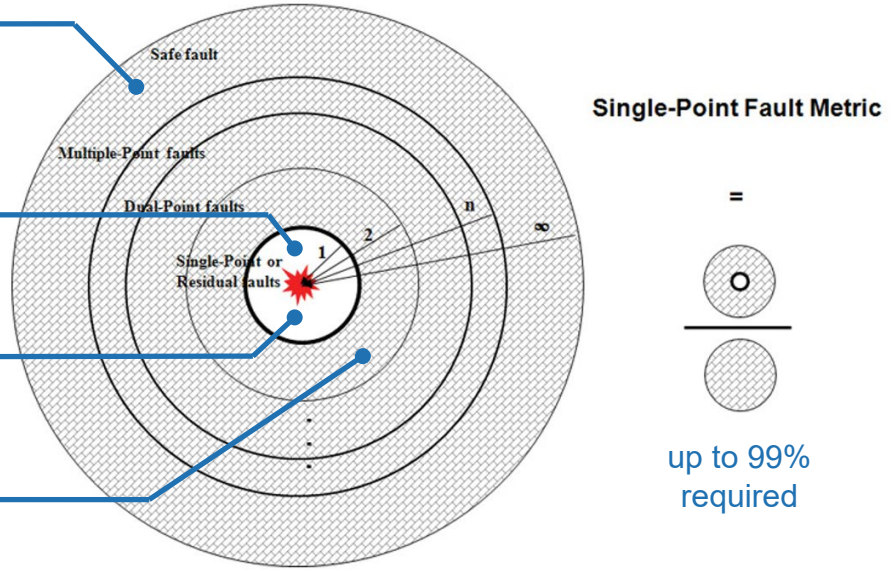


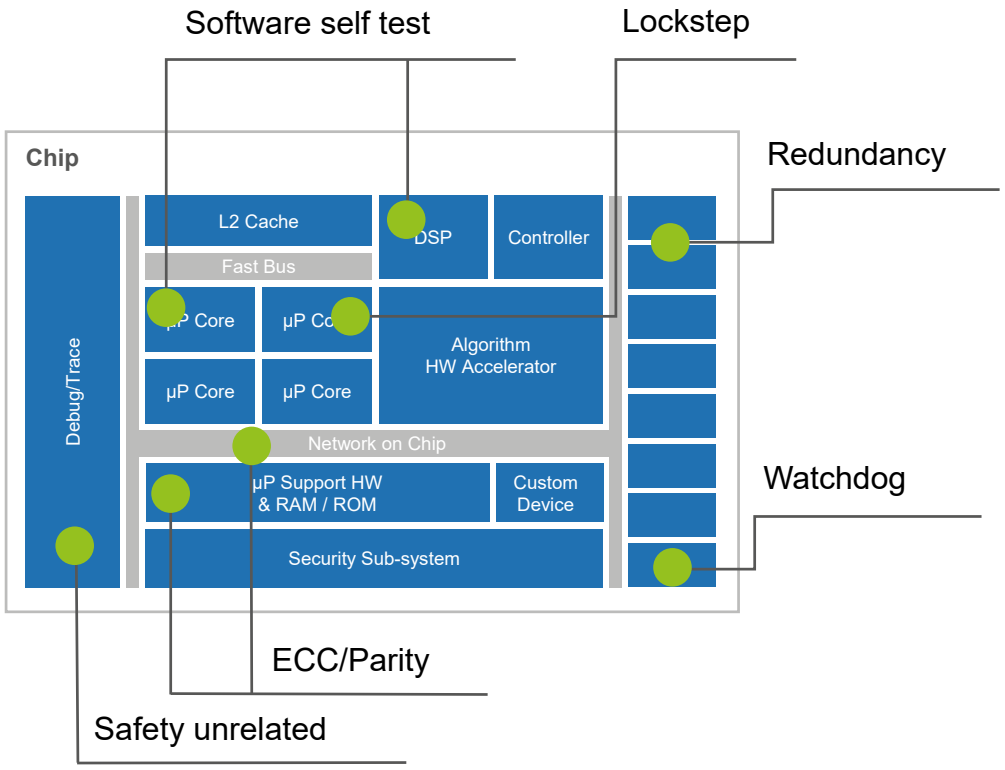
Diagram: Courtesy International Standards Organization (ISO)

- Unidentified safe faults must be considered residual
- Lower diagnostic coverage

Diagnostic Coverage of Safety Mechanisms

Challenges

- Diverse/complex safety mechanisms
- High number of faults
- Final metrics computed on gate-level netlist
- Fault simulation scales but fault coverage often insufficient
- Expert judgment of coverage holes time consuming and error prone
- **Formal could help but suffer from complexity issues**



Safe Stuck-at Faults

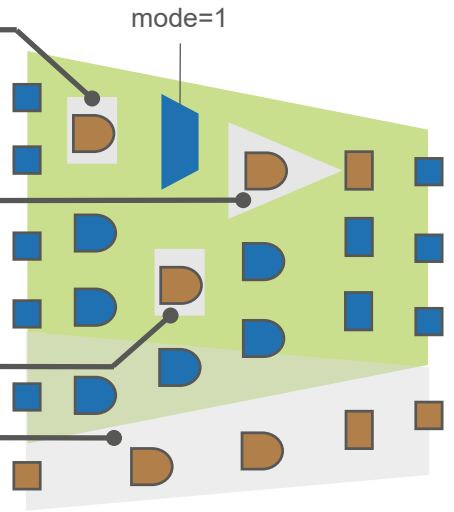
Inputs to Formal FPA
- Design (RTL/Gates)
- Constraints
- Fault population (optional)
- Observation points

Safe faults due to static IC operation modes
- Debug mode disabled
- Test logic

Explicit redundancy in hardware masks the effect of fault
- Performance impact only
- States never used in safe operation mode

Truly redundant logic such as synthesis deficiencies

Safety unrelated logic
- Design parts which do not impact the safety goal



Safe faults cannot propagate to observation points

- Mission outputs
- Internal registers

Split Formal Analysis into Fast and Deep

Fast Formal Fault Propagation Analysis

- Select appropriate proof strategies targeting
  - Safe faults only (hold results rather than fail)
  - Large fault populations
- Drop debug information to gain capacity/speed
- Drop hard faults ASAP to avoid wasted effort
- Smart clustering of faults
- Automatically orchestrate mix of methods
  - Cone of influence analysis
  - Combinatorial/Sequential Equivalence Checking
  - Auto-generated assertions

Goal

- Find the majority of safe faults as fast as possible

Split Formal Analysis into Fast and Deep

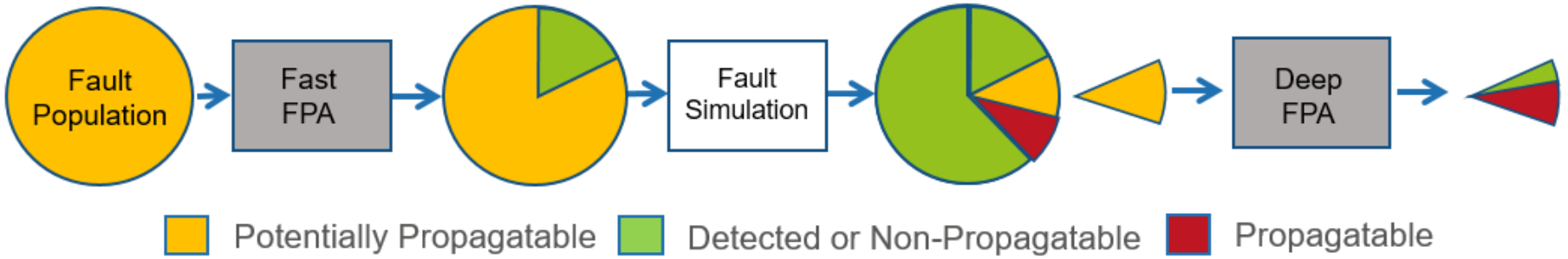
Deep Formal Fault Propagation Analysis

- Select appropriate proof strategies targeting
  - Hard safe faults
  - Propagatable faults
  - Small fault populations
- Debug information is crucial
- Additional user input (e.g. SVA constraints)

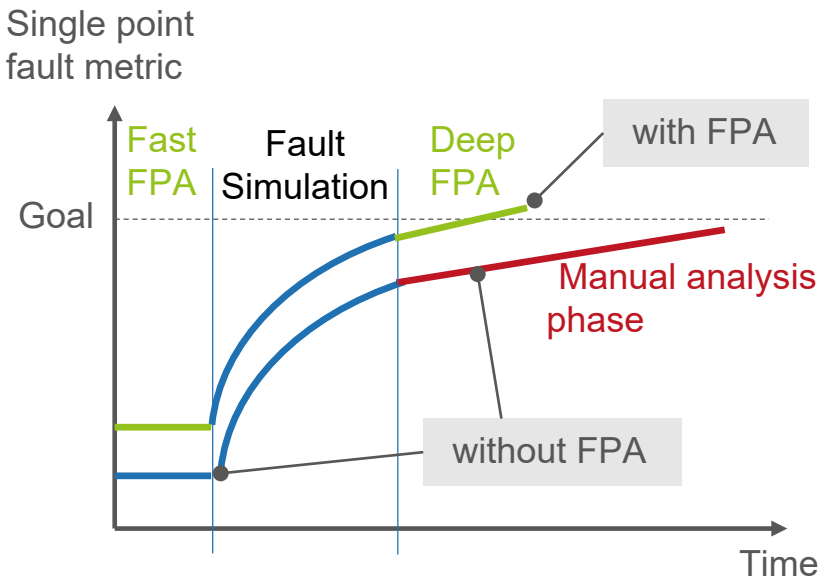
Goals

- Find additional safe faults
- Debug propagatable faults

Integration of Formal FPA with Simulation



- Two-mode approach fits well with simulation flow



Case Studies Results

Basic constraints
- Debug off
- Test off

FAST FPA	Identify Safe Stuck 0/1 Faults	
	RTL	Gate-level
Description	Open Core Processor – 10K LoC – 794 FFs	Summary of various designs
Fault Population	12.260 faults	Up to 2 million faults
CPU Time	2 minutes	Up to several hours
Safe Faults	3257 (26%)	Up to 14%

- Experience demonstrates that designs have a significant number of safe faults …
- … that can be unveiled with fast targeted formal analysis

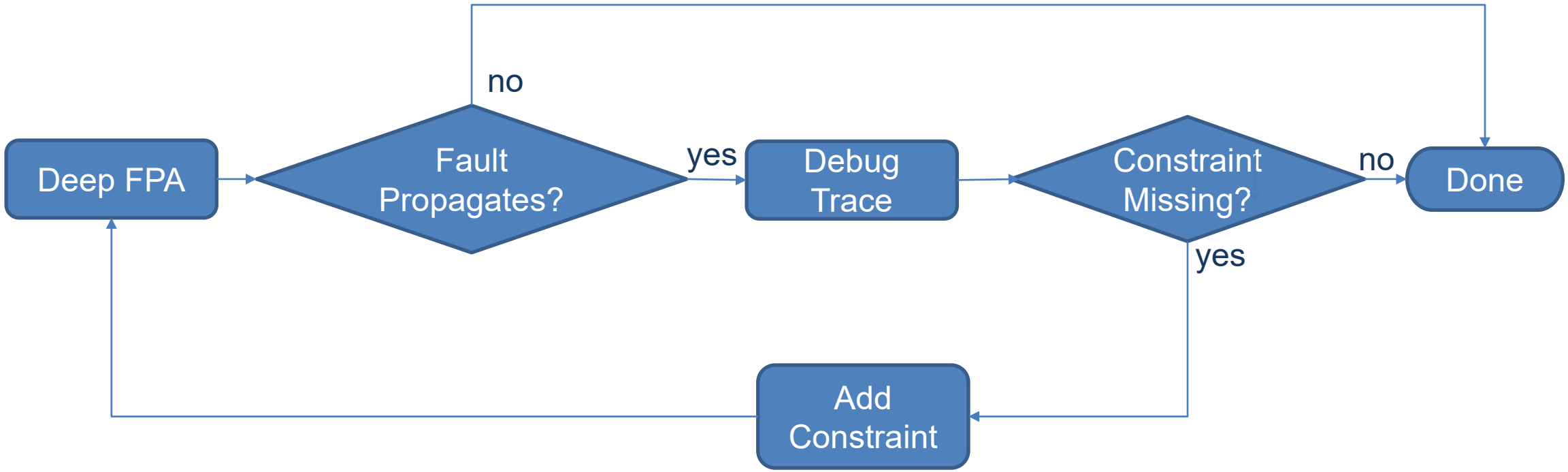
Case Studies Results

Basic constraints
- Debug off
- Test off

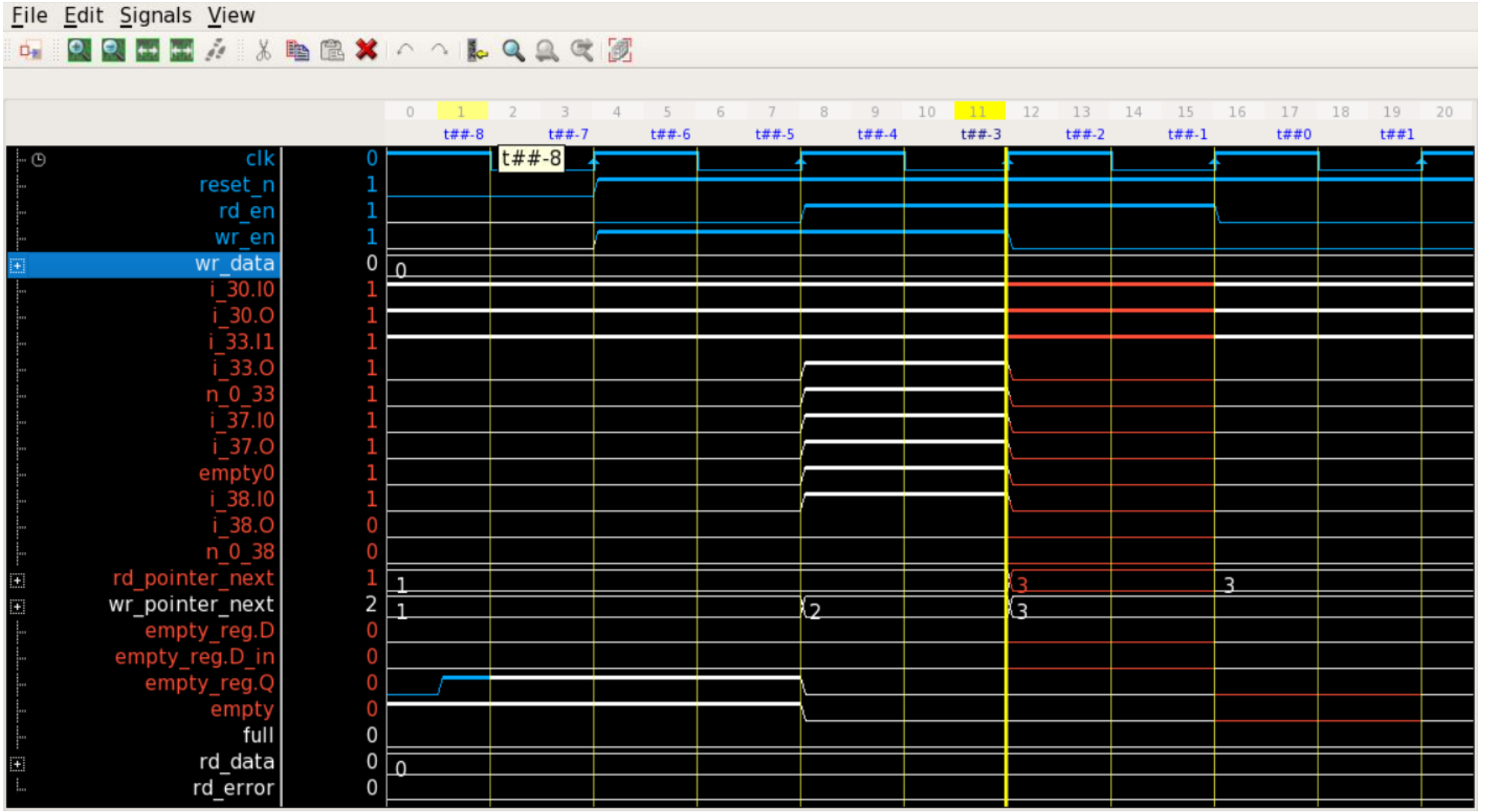
DEEP FPA	Identify Safe and Propagatable Stuck 0/1 Faults	
	RTL	**Gate-level**
Description	Open Core Processor – 10K LoC – 794 FFs	Summary of various designs
Fault Population	12.260 faults	Up to 2 million faults
CPU Time	Averaging at 20 seconds per fault	Averaging at several minutes per fault
Safe Faults	13%	Up to 2.1% of the remaining faults

- Deep analysis does find additional safe faults
- For certain designs engineers want to examine every fault
- Debug and integration with SVA crucial to examine unexpected propagatable faults

Analysis of Propagatable Faults



Propagatable Fault Debug Trace



Questions?