

# Whose Fault Is It Formally? Formal Techniques for Optimizing ISO 26262 Fault Analysis.

Ping Yeung, Doug Smith, Abdelouahab Ayari Mentor, a Siemens Business



#### 2018 DESIGN AND VERIFICATION DVCDN CONFERENCE AND EXHIBITION UNITED STATES

2

#### ISO 26262 – Automotive Functional Safety

1. Vocabulary 2. Management of functional safety 2-6 Safety management during the concept phase 2-7 Safety management during production, 2-5 Overall safety management and the product development operation, service and decommissioning 3. Concept phase 4. Product development at the system level Production, operation, service and 4-5 General topics for the product development at the system level 4-9 Safety validation decommissioning 3-5 Item definition 7-5 Planning for production, 4-8 System and Item integration and verification 4-6 Technical safety concept 3-6 Hazard analysis and risk operation, service and assessment decommissioning 4-7 System architectural design 3-7 Functional safety 7-6 Production concept 7-7 Operation, service and decommissioning 12. Adaptation of ISO 26262 5. Product development at the 6. Product development at the for motorcycles hardware level software level 5-5 General topics for the product development at the hardware level 6-5General topics for the product development at the software level 12-5 Confirmation measures 5-6 Specification of hardware 6-6 Specification of software safety safety requirements requirements 12-6 Hazard analysis and risk 5-7 Hardware design 6-7 Software architectural design assessment 5-8 Evaluation of the hardware. 6-8 Software unit design and architectural metrics implementation 12-7 Vehicle integration and testing 6-9 Software unit verification 5-9 Evaluation of the safety goal violations due to random hardware 6-10 Software integration and failures verification 12-8 Safety validation 5-10 Hardware integration and 6-11 Testing of the embedded verification software 8. Supporting processes 8-5 Interfaces within distributed developments 8-9 Verification 8-14 Proven in use argument 8-10 Documentation management 8-15 Interfacing a base vehicle or item in an 8-6 Specification and management of safety requirements 8-11 Confidence in the use of software tools application out of scope of ISO 26262 8-7 Configuration management 8-12 Qualification of software components 8-16 Integration of safety related systems not developed according to ISO 26262 8-8 Change management 8-13 Evaluation of hardware elements 9. ASIL-oriented and safety-oriented analyses 9-5 Requirements decomposition with respect to ASIL tailoring 9-7 Analysis of dependent failures 9-6 Criteria for coexistence of elements 9-8 Safety analyses 10. Guideline on ISO 26262 11. Guideline on application of ISO 26262 to semiconductors

5.6 HW safety requirements5.7 HW design5.8 Eval of the architecture5.9 Eval of the safety goal5.10 HW verification



## **Formal Safety Analysis**

- Safety Analysis
- Fault Management
- Fault Analysis





#### **ISO 26262 – Classification of Faults**



1



#### **Classification of Faults**



Failure Rate  $\lambda$ 



# **Formal Safety Analysis**

- Fault Pruning
  - Formal improves efficiency by minimizing the fault set that needs to be fault injected
  - Minimized fault list can be fault injected across different engines depending on the problem
- Fault injection
  - Same engines that are used for functional verification enabling better reuse of tests, flows, data
  - Fault injection results are combined through a common database to provide a single set of fault metrics and Diagnostic Coverage (DC)
  - Formal provides exhaustive fault injection analysis including exhaustive transient fault analysis





#### **Formal Fault Analysis Flow**





#### **Diagnostic Coverage Overview**

		Safety Mechanism		
		Undetected Fault	Detects Fault	
Functional Output	Unaffected	UU – Faults do not impact functional output and not detected by Safety Mechanism. <b>Safe Fault</b>	UD – Faults do not impact on functional output but Safety Mechanism detected fault. <b>Safe Fault</b>	
	Impacted	DU – Faults impact functional output but not detected by Safety Mechanism <b>Undetected Dangerous Fault</b>	DD – Faults impacts functional output and Safety Mechanism detected fault <b>Dangerous Fault</b>	





## **Fault Analysis: Fault Pruning**

Reducing the set of faults that need to be fault injected

- A subset of faults
  - Only a subset of faults in a given design will affect the safety requirement. They are in the COIs of the safety critical signals
- Safe elements
  - Design elements not in the COI of a safety critical signal automatically considered safe
- Configurations and constraints
  - The COI can be reduced further by applying top-level constraints such as disabling DFT, debug and test, or other non-operational modes







## Fault Analysis: Safety Mechanism

Safety Mechanisms reduce Fault Injection Requirements

- Detectable fault
  - Design elements in the COI of a safety requirement, and
  - overlap with the COI of the associated safety mechanism
- Undetectable fault
  - Design elements in the COI of a safety requirement, and
  - *not* in the COI of the safety mechanism
  - must be considered a dangerous fault



#### 2018 DESIGN AND VERIFICATION CONFERENCE AND EXHIBITION UNITED STATES

# **Results of formal fault pruning**

- Case 1: The design is a float point unit, ~530K gates.
  - The goal is to identify the safe faults in the design.
  - These faults are outside the COI or cannot be propagated to functional outputs.
- Case 2: The design is a memory management unit, ~1.3M gates.
  - The goal is to identify faults that can propagate to internal status registers.
  - These registers are checked by safety mechanisms at a higher level.

Case	Gates	Faults	Safety Mechanism	Run Time per Fault	Safe Faults	% Safe Faults
#1	530K	32425	0	3.6 sec	868	2.7%
#2	1300K	1524	71	2.3 sec	720	47%



### **Fault Injection and Equivalence**

Targeted stuck-at and transient fault analysis without a testbench

#### • Targeted Fault Injection

- Once the design is clean of *Structural Faults*
- Once the number of design elements have been pruned down to a manageable and meaningful subset
- Exhaustive Fault Analysis
  - Formal can be used to inject faults and compare the outputs of the two designs – Golden vs Faulted
  - Formal tools have the ability to inject both stuck-at and transient faults into a design, and see if the fault is propagated, masked, or detected by a safety mechanism



Golden Design

# **Results of formal fault injection (1)**

The design is a clock controller block with triple modular redundancy (TMR)

- #1a: faults were allowed to be injected to all the registers in the design
  - All the registers are in the COI of the safety mechanisms, there is no surprise.
- #1b: faults were allowed to be injected to all the nodes (registers, gates, and wires)
  - There are significantly more faults.

Case	Faults	Number of Faults	Run Time	% Missed by Safety Mechanisms
#1a	registers	57	15 min	0
#1b	all nodes	2648	265 min	0

• Formal fault injection verifies that all the injected single point faults will be caught by the safety mechanisms.

# **Results of formal fault injection (2)**

The design is a bridge controller that consists of the clock controller block

- Formal fault injection was able to inject and propagate some faults to the output ports of the design.
- Two types of faulty scenarios were observed:

2018

- Single point faults that were not protected by any safety mechanism
- Residual faults that were protected by safety mechanisms; however, the safety mechanisms did not detect the error conditions correctly.

Case	Faults	Number of Faults	Run Time	% Missed by Safety Mechanisms
#2a	registers	267	23 min	12%
#2b	all nodes	12963	1332 min	8%



#### **Questa Fault Injection and Equivalence**

Dangerous Undetected (DU) Fault





# **Formal Fault Analysis: Advantages**

- **Exhaustive analysis (**does not require a testbench or test cases)
  - Of the design for faults (safety mechanism)
  - Of any faulty condition (diagnostic coverage)
  - Of design equivalence (fault injection)
- Determines the diagnostic coverage
  - simply provide a list of safety critical requirements and the safety detection logic,
  - formal automatically prunes the fault set, injects stuck-at and transient faults, and determines the diagnostic coverage of the design
- Determine the number of safe faults ( $\lambda_s$ ) by
  - finding the unreachable design elements, those outside of a cone of logic, or
  - those that do not affect the outputs (or gated by a safety mechanism)
- Use the fault set from fault pruning to determine accurate numbers for
  - single-point failures ( $\lambda_{SPF}$ ), residual faults ( $\lambda_{RF}$ ), and multi-point failures ( $\lambda_{MPF}$ )

# **Applications for Formal Fault Analysis**

- Derive essential list of potential faults for analysis
  - Simulation, emulation, formal analysis
- Pruning the list of potential faults
  - Based on cone-of-influence (COI) analysis
  - The locations of safety mechanisms
- Safety Mechanism Verification
  - Fault detection and recovery
- Compute and calculation the Diagnostic Coverage
  - Generation of detection and coverage assertions
- Fault Injection and Equivalence Checking
  - Checking of golden and faulty design



HOW FORMAL REDUCES FAULT ANALYSIS FOR ISO 26262





- ISO 26262 is challenging, but it can be mastered
- Exhaustive formal verification is key to fault injection and analysis
- Fault analysis requires a comprehensive approach
- Tutorial
  - How to Stay Out of the News with ISO26262-Compliant Verification
  - Thu March 01, 2:00pm 5:30pm | Siskiyou



#### **Thank You**



# **Classification of Hardware Faults**

- $\lambda_s$  Safe Faults
  - Do not effect the Safety requirements
- $\lambda_{\text{SPF}}$  Single Point Fault
  - Fault violating a safety requirements, not covered by a Safety Mechanism.





# **Classification of Hardware Faults**

- $\lambda_{RF}$  Residual Faults.
  - Faults not detected by an intended Safety Mechanism and lead to a violation of Safety requirements. Can be considered an escape.
  - For most designs, the Single Point Faults and Residual Fault are not differentiated from a fault analysis perspective.
  - Residual Faults may matter if the quality of individual Safety Mechanisms matter





# **Classification of Hardware Faults**

- $\lambda_{MPF}$  Multiple Point Fault.
  - Combination of independent faults which may lead to a violation of Safety requirements
  - Multi-point faults require engineering analysis to determine likelihood and location of latent faults in the design.

