Functional Safety Verification for ISO 26262-Compliant Automotive Designs

JM Forey - Synopsys
Werner Kerscher - Synopsys
Agenda

• Emergence of self-driving cars
• ISO 26262 Primer for Semiconductor
• ISO 26262 Requirements – Hardware Development
• Functional Safety Verification Flow, from FMEA to FMEDA
EMERGENCE OF THE SELF DRIVING CAR
Autonomous Vehicles Are Taking Over The World

September 12, 2017 | Ann Arbor, Michigan
TRANSPORTATION SECRETARY ELAINE L. CHAO AT TECHNOLOGY, EMPHASIZES SAFETY BENEFITS AND
Ann Arbor today re guidance.

“The new
Autonomous Fahren
Millionen Euro für 1

IT Mobiles Entertai

Finanz

News

autonomes fahren

Autonomes Fahren
Karlsruhe wird Testgebiet für Roboter-Autos

Baden-Württemberg will zum Vorreiter beim autonomen Fahren werden. Deshalb dürfen Robo-Autos jetzt auf ausgewählten Straßen rund um Karlsruhe fahren.
## Levels of Automation in Cars

<table>
<thead>
<tr>
<th>LEVEL 0</th>
<th>LEVEL 1</th>
<th>LEVEL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Level 0 Image]</td>
<td>![Level 1 Image]</td>
<td>![Level 2 Image]</td>
</tr>
<tr>
<td>There are no autonomous features.</td>
<td>These cars can handle one task at a time, like automatic braking.</td>
<td>These cars would have at least two automated functions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LEVEL 3</th>
<th>LEVEL 4</th>
<th>LEVEL 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Level 3 Image]</td>
<td>![Level 4 Image]</td>
<td>![Level 5 Image]</td>
</tr>
<tr>
<td>These cars handle &quot;dynamic driving tasks&quot; but might still need intervention.</td>
<td>These cars are officially driverless in certain environments.</td>
<td>These cars can operate entirely on their own without any driver presence.</td>
</tr>
</tbody>
</table>

**Source:** SAE International
Roadmap of Autonomous Cars

L5: Self Driving Only
Self-Driving Only

L4: Full Self Driving
Self-Driving & Human-Driven Car

L3: Limited Self Driving
Auto Pilot: Road Train
Auto Pilot: Parking
Auto Pilot: Highway
Auto Pilot: Traffic Jam

L2: Partial Autonomy
Parking Assist
ACC With Lane Keep Assist

L1
Autonomous Braking
Adaptive Cruise Control

2010 2015 2020 2025 2030

© Accellera Systems Initiative

6
Enabling Safe, Secure, Smarter Cars
...from Silicon to Software

- Software cybersecurity & quality
- Verify functional safety (ISO 26262)
- Automotive-certified IP
- High-reliability IC design
- ISO 26262-certified Test

Quality, Security, Safety

accellera
ISO 26262 PRIMER FOR SEMICONDUCTOR
What is Functional Safety?

• Functional Safety is the “Absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems” [ISO 26262]

• In a nutshell, functional safety is about ensuring the safe operation of systems even when they go wrong

• Functional safety is critical to many markets: Aerospace, Medical, Industrial, Automotive, etc.
V-Diagram: Automotive View of “Design”

Specification

Tier 1

Implementation

Verification & Validation

OEM

Vehicle

Tier 1

ECU

Semi

HW + SW

© Accellera Systems Initiative
Functional Safety Standards

• IEC 61508: Base functional safety standard

• ISO 26262: Automotive functional safety standard
  – Derived from IEC 61508, published 2011
    • Part 1: Vocabulary
    • Part 2: Management of Functional Safety
    • Part 3: Concept Phase
    • Part 4: Product Development: System Level
    • Part 5: Product Development: Hardware Level
    • Part 6: Product Development: Software Level
    • Part 7: Production and Operation
    • Part 8: Supporting Processes
    • Part 9: ASIL Orientated and Safety Oriented Analysis
    • Part 10: Guideline on ISO 26262
    • Part 11: Application of ISOS 26262 to Semiconductors (2nd Edition)
Safety Goals/Requirements

• Done at OEM / Tier 1 level

• Safety Goal
  – Top-level safety requirement
  – Derived from Hazard Analysis and Risk Assessment (HARA)

• Example(s)
  – Unintended activation of emergency brake must be prevented
  – Unintended inflation of airbags must be prevented.
Hazard Analysis and Risk Assessment (HARA)

• Determines the Automotive Safety Integrity Level (ASIL)

<table>
<thead>
<tr>
<th>ASIL</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Severity</th>
<th>Probability</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>E0</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E1</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td>S2</td>
<td>E3</td>
<td>QM</td>
<td>QM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>QM</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>S3</td>
<td>E0</td>
<td>QM</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E1</td>
<td>QM</td>
<td>QM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>QM</td>
<td>QM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

- E0: Combination of Very low Probabilities
- E1: Very Low Probability (less often than once a year for the great majority of drivers)
- E2: Low Probability (a few times a year for the great majority of drivers)
- E3: Medium Probability (once a month or more often for an average driver)
- E4: High Probability (almost every drive on average)

- C0: Controllable in general
- C1: Simply controllable (99% or more of all drivers are usually able to avoid a harm)
- C2: Normally controllable (90% or more of all drivers are usually able to avoid a harm)
- C3: Difficult to control or Uncontrollable (Less than 90% of all drivers are usually able or barely able to avoid a harm)

- S0: No injuries
- S1: Light and moderate injuries
- S2: Severe and life-threatening injuries (survival possible)
- S3: Life threatening injuries (survival uncertain), fatal injuries
A Safety Element out of context (SEooC) is a safety-related element which is not developed for a specific item. This means it is not developed in the context of a particular system or vehicle.

See ISO 26262 Part 10 "Guideline on ISO 26262", Chapter 9 "Safety element out of context"

Chips and IPs are normally Safety Elements out of Context

**Issue**
- No/little knowledge of the system in which the design is used
  - Hazards
  - Safety goals
  - Architecture

**Resolution**
- SEooC vendors need to specify Assumptions of Use (AoU)
  - Safety requirements
  - Expected integration environments and requirements
- SEooC vendors should aim at highest expected ASIL
  - Fault avoidance
  - Fault control
  - Independent confirmation measures
What is Functional Safety?

• Functional Safety is the “Absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems” [ISO 26262]

• In a nutshell, functional safety is about ensuring the safe operation of systems even when they go wrong

• Functional safety is critical to many markets: Aerospace, Medical, Industrial, Automotive, etc.
What is Functional Safety?

- Functional Safety is the “Absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems” [ISO 26262]
- In a nutshell, functional safety is about ensuring the safe operation of systems even when they go wrong
- Functional safety is critical to many markets: Aerospace, Medical, Industrial, Automotive, etc.

- **Safety** is a mind set
- **What can go wrong?**
  - At any level, notably
    - Conception: hw, sw
    - Verification
    - Manufacturing
    - In operation, in a permanent or transient way
    - ...

- *And of course*
  - Measuring, addressing, minimizing impact, documenting, ...

- **Faults are either**
  - Systematic
  - Random

- Found/covered by Functional Verification tools
- Assessed by Functional Safety Verification tools
ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust
- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

Systematic Faults
Always permanent

Reduced DPPM
- DFT
- Functional patterns

Random Faults
Permanent

Demonstrate and document that safety mechanisms operate properly
- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults
Permanent Transient

Development Manufacturing In Operation

Lifecycle of Component / System / Automobile

© Accellera Systems Initiative 18
ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust

- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

Systematic Faults

Always permanent

Reduced DPPM

- DFT
- Functional patterns

Random Faults

Permanent

Demonstrate and document that safety mechanisms operate properly

- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults

Permanent

Random Faults

Transient

Lifecycle of Component / System / Automobile
Functional Verification is Essential Starting Point

Prevent / Eliminate Bugs

Avoid Systematic Faults – Design Bugs (Permanent Faults)

Verification & Validation: Use State of the Art Functional Verification methodology

- Many technologies must be used to ensure the highest functional verification quality
- Early software bring-up enables faster and more complete verification
- Verification quality analysis provides objective measure of functional verification effectiveness
Executive Overview

System Requirements

Functional Requirements
Safety Requirements

Verification Plan
Safety Plan

Functional Verification
Fault Injection Testing

Certitude*
Verdi*
Fault Testing Campaign

Virtualizer
VC Formal
VCS*
ZeBu
HAPS

Accelerate fault simulation campaign

- Most comprehensive solution for systematic and random faults testing
- Fastest simulation engines

Integrated with ISO 26262 flows

- Failure mode effects analysis
- Safety plan traceability and results

Proven

- TuV Certified Tools
- Deployed for Synopsys Certified IP development
- Adopted by market leaders
**Functional Verification Qualification**

In ISO 26262 context

- ISO 26262 part 8 Clause 11.2: "Risk of systematic faults [...] is minimized"
- Problem: test Infrastructures deliver pass/fail status
  - Do not directly address whether designs have bugs or not

- Reported failures are debugged (good) => there is always something to fix
- **BUT** False Negative are silent
  - Are there any? Where are they?
  - Traditional methods can’t help here

<table>
<thead>
<tr>
<th>Design actually has bug(s)</th>
<th>Verification reports bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No (all tests Passed)</td>
</tr>
<tr>
<td></td>
<td>Yes (some Failures)</td>
</tr>
</tbody>
</table>

- False Negative
  - Ok (done)
- False Positive
  - Ok (debug design)

Is your verification tool failing to report functional bugs?
Assessing Verification Effectiveness

Traditional methods

**Code coverage** measures activation, but **not** propagation nor detection

**Functional coverage** checks “important” functional points, however comprehensiveness of functional points is unknown

- Code/functional coverage are used to assess verification effectiveness
  - BUT they deliver a very partial picture
Assessing Verification Effectiveness

- Mutation testing applies universally in verification
- **Automatically inserts** “artificial bugs” into the design
- Runs verification process on “broken” design
- Measures the ability of the environment to exercise the fault, propagate and detect its effect

VE can be anything, including:
- Analogue
- Digital
- C/C++/SystemC

VE Engine:
- Simulation (hw model)
- Formal (hw model)
- Execution (sw)

Functional Qualification

Bugs (sw, hw model) or defects (hw model)

Hardware model or Software Code

Systematic Failures
How Certitude Fault Injection Works?

- Modifies design code to insert faults/defects

  \[
  o_1 = f(i_1) \rightarrow o_1 = '0b0 \quad \text{// tie to constant}
  \]

  \[
  \begin{align*}
  \text{if (a)} & \rightarrow \text{if (TRUE)} \quad \text{// force execution} \\
  & \text{f1();} \\
  \text{else} & \text{else} \\
  & \text{f2();}
  \end{align*}
  \]

  \[
  a = b | c \rightarrow a = b \& c \quad \text{// change operator}
  \]

- Pass the broken design to the verification
  - Does at least one test fail? \textit{Great!}
    - Environment/Safety Mechanisms robust enough to detect the design is broken
  - Do all tests pass? \textit{Help!} \quad \Rightarrow \text{False Negative result} \Rightarrow \text{VE is hiding bugs}
    - Original and broken design looks ok for the verification
Easy Integration Within Existing Environments

Mutation Based Analysis Flow

- Testcase List
- Config Options
- Verification Infrastructure
- Test Cases
- Mutated DUT
- Compare
- Expected Results
- Reports

© Accellera Systems Initiative
Formal Verification

Mutation Based Analysis Flow

- Config Options
- Properties And Constraints
- DUT
- Proof
- Reports

© Accellera Systems Initiative 2018
Demonstrate Verification Flows are Robust

Evidence-based verification quality analysis for ISO 26262 Part 8-9 assessments

Inject and qualify systematic faults at architecture, system, and RT level

Measure verification completeness and functional correctness of design

Natively integrated with VCS and VC Formal, and works with C/C++/SystemC flows

Unified functional verification environment quality metrics
ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust
- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

Systematic Faults

Always permanent

Reduced DPPM
- DFT
- Functional patterns

Random Faults

Permanent

Demonstrate and document that safety mechanisms operate properly
- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults

Permanent

In Operation

Lifecycle of Component / System / Automobile

© Accellera Systems Initiative
Fault Injection Testing – Z01X Manufacturing

Highest performance fault simulation solution

• Challenges
  – Stringent fault simulation is needed for highest fault coverage
  – Comprehensive fault model support is required
  – Performance and capacity demands are extreme

• Objective
  – Generate additional coverage and usefully grade patterns with acceptable TAT

• Results
  – RealTek described their results in SNUG Taiwan 2017
  – NovaTek described their results in SNUG Taiwan 2018
  – Qualcomm scheduled for SNUG Austin 2018
Z01X Concurrent Fault Simulation

Parallel Simulation Technology
One fault per simulation

Z01X Concurrent Simulation Technology
Thousands of faults in a single simulation
Orders of magnitude faster than parallel

© Accellera Systems Initiative
Z01X Flow

Key:
User Created File: 
Z01X Intermediate File: 
Z01X Executable: 
Z01X Output: 

If testability shows % coverage below required level. Do not fault simulate!
ISO 26262 Requirements – Hardware Development

Show that design functionality is correct, works properly in the context of the system, and is safe

Demonstrate and document that design and verification flows are robust
- Implementation tools and flows do not introduce design bugs (systematic faults)
- Functional verification tools and flows do not fail to report design bugs

Systematic Faults
- Always permanent

Reduced DPPM
- DFT
- Functional patterns

Random Faults
- Permanent

Demonstrate and document that safety mechanisms operate properly
- Safety mechanisms triggered in presence of faulty behavior, and not otherwise
- Safety mechanisms are effective in reaching a safe design state

Random Faults
- Permanent
- Transient

Lifecycle of Component / System / Automobile

© Accellera Systems Initiative
Verification Goal Comparison

**Functional Verification**
- Prevent / Eliminate Bugs
  - Validate functional correctness of design
  - Unified verification technologies with fastest engines
  - Development and manufacturing testing
  - Avoid Systematic Faults

**Functional Safety Verification**
- Control Failures
  - Confirm effectiveness of safety mechanisms
  - Confidence in tool chain
  - “In Operation” testing
  - Control of Random Faults

© Accellera Systems Initiative
Verification Flow Alignment

- Alignment of requirements for functional and safety verification
  - Accelerate complete verification process
- Requires solution for systematic and random fault testing
- Integrated with ISO 26262 Flows
  - Failure mode effects analysis
  - Metric reports
  - Safety requirements traceability
Verification Goal Comparison

**Functional Verification**

Prevent / Eliminate Bugs

Unified verification technologies with fastest engines

‘Shift-Left’ for Faster Time-to-Market
Manage Growing SoC Verification and System Validation Complexity and Cost

**Functional Safety Verification**

Control Failures

Certified tool chain

© Accellera Systems Initiative
Safety Fault Metrics for ISO 26262 ASIL Ratings

• Fault Injection Testing recommended for ASIL A & B and highly recommended for ASIL C & D

<table>
<thead>
<tr>
<th>Method</th>
<th>ASIL A</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Injection Testing</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

• Maximize detection of single point faults

<table>
<thead>
<tr>
<th>Metric</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Point Fault Metric</td>
<td>≥ 90%</td>
<td>≥ 97%</td>
<td>≥ 99%</td>
</tr>
</tbody>
</table>

• Maximize detection of multi-point latent faults

<table>
<thead>
<tr>
<th>Metric</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent Fault Metric</td>
<td>≥ 60%</td>
<td>≥ 80%</td>
<td>≥ 90%</td>
</tr>
</tbody>
</table>
Functional Safety Process

Implement and Confirm Quality of Safety Mechanisms (SM)

- Identify Failure Mode and Effects Analysis (FMEA) for device
- Implement Safety Mechanisms to protect against failures
- Run fault injection to measure ISO 26262 metrics
- Generate FMEDA report, Safety manual
Unique Functional Safety Needs - Summary

Generate FMEDA reports (ISO-26262 deliverable)

Certified development flows & Safety documentation

Safety Requirements Traceability

Safety Requirements

Safety Verification Plan

tests covergroups code cov fault

Safety Coverage
FUNCTIONAL SAFETY VERIFICATION FLOW
FMEA TO FMEDA
ISO 26262 Work Products

• FMEA, FMEDA
  – **F** – **Failures of a given component** Consider a component in a system
  – **M** – **Mode** Look at one of the ways in which it can fail
  – **E** – **Effects** Determine the effects this failure mode will cause to the system we are examining
  – **D** – **Diagnostic** Determine the coverage
  – **A** – **Analysis** Analyze how much impact the symptom will have on the environment/people/the system itself

Source: https://about.brighton.ac.uk/cem/research/seminars/2011/fmea_pres.pdf
FMEA/FMEDA process flow

**FMEA**

1. Divide design in functional blocks
   - Component->Parts
2. Divide further each part to areas with specific functionality, or Sub Parts (SPs)
3. Identify Failure Modes (FM) and Safety Mechanisms (SM) for each SP
4. Create an FMEA for each FM/SM for a Sub-Part
   - [One SP may have several FM/SM]

**FMEDA**

1. Map each FMEA to the relevant design Sub-Part
   - [Use a Scaling Factor in case of multiple FMAs in the same SP]
2. Calculate estimated ISO26262 Metric
   - Specify design data, design FIT rate, faults safeness and SMs faults coverage estimates
3. Run Fault campaign for measured ISO 26262 Metric
   - For each FMEDA define observation and detection points, test suite and faults list

Results should match!
FMEA Inputs example

- Design block level list and diagram.

```
- Reset Logic
- Flag Logic
- Read Control
- Read Pointer
- Write Control
- Write Pointer
- SRAM
```

Block Diagram of FIFO with Static Memory

© Accellera Systems Initiative
Failure Mode Effect Analysis example

- **Failure Mode 1:**
  - Failure: Full signal is not raised when FIFO is full
  - Effect: Data will be overwritten or lost
  - Safety Mechanism: Redundant Control Logic

- **Failure Mode 2:**
  - Failure: Data in SRAM is corrupted
  - Effect: Invalid data
  - Safety Mechanism: ECC

![Block Diagram of FIFO with Static Memory](image)
Fault Classification Simplified

Failure mode of HW element

Non-safety related
- Safe fault
  - Not considered in Metric

Safety related
- Safe fault
- Detected MPF
- Perceived MPF
- Latent fault
- Residual and SPF
FMEA Work product example:
Failure Mode Effect & Diagnostic Analysis (FMEDA)

• A detailed analysis technique to obtain:
  – Design failure rates
  – Failure Modes diagnostic capability

• FMEDA is an extension of the FMEA analysis
  – Assessing the Safety Metrics for the given Failure Mode

• FMEDA Inputs:
  – Technology Information for Failure In Time (FIT)
    • Needed to compute Failure Rates
  – Design information
    • Digital logic and analog area, flop/latch, RAM/ROM counts
      – Needed to compute Failure Mode Distribution
  – Safety Mechanism (if exists) for the Failure Modes

ISO 26262 acceptable technology standards:
- IEC TR 62380
- SN 29500
- FIDES Guide
**FMEDA Creation Flow**

*SP level Analysis – one FMEDA per one FMEA line*

**Administrator – per Project**
- Provide Design Data
  - [Hierarchical data ion all SPs – digital area, RAM bits, FF, Latches etc]

**User/IP owner – per Sub Part**
- Specify SM type
  - [will provide initial Diagnostic coverage estimate]
- Update Fsafe percentage
  - [Fsafe is the portion of faults which go not violate the safety goal]
- Associate a design element
  - [If multiple FMEDAs on the same design element – use Scale Factor]

Estimated ISO 26262 Metric
Failure Mode (FM) Distribution

• Each FMEDA needs to have a base Failure Rate assigned to it
• Possible distributions:
  – **Uniform**: Each FM has a failure rate equal to the overall failure rate divided by the number of failure modes
    • Reasonable assumption for initial analysis; assumes highly symmetrical design
  – **Area**: Each FM’s failure rate depends on its relative portion of the design area
    • Similarly, it may depend on the number of gates/flops
  – **Number of outputs affected**
    • Considers their cone of influence
FMEDA Diagnostic Coverage Components

• **Fault list** – a list of design locations with potential random failures
  – Based on FMEA potential cause of failure
  – Generated from block level or elementary sub parts

• **Observation Points**
  – Design points in which the effect of an injected fault should be observed
    • Normally – at the boundary of a block in which the fault is injected

• **Diagnostic Points**
  – Design points which are activated when the safety mechanism detects the injected fault
    • e.g.: safety_alarm IO pin, interrupt to interrupt controller etc.
FMEDA Diagnostic Coverage Components – cont.

• Workloads
  – These are sets of tests which stimulate the area of the injected fault
  – Types of workloads:
    • Representative: follow normal use cases, do not necessarily activate all signals in the relevant block
    • Exhaustive: provide 100% toggle coverage of the relevant block
ISO 26262 Fault Classification

Technology
FIT

Failure mode

\( \lambda \)

Is there a safety mechanism in place to control failure modes of the technology (yes/no)?

\( \lambda_{\text{safe, not prevented}} = \lambda_{\text{safe}} \cdot (1 - \lambda_{\text{safe, not prevented}}) \)

Safe fault not to be considered in the assessment.

\( \lambda_{S} \)

Single Point Fault

\( \lambda_{\text{SPF}} \)

Residual Fault

\( \lambda_{\text{RF}} \)

Which fraction is perceived (yes/no)?

\( \lambda_{\text{MPF, det}} \)

\( \lambda_{\text{MPF, lat}} \)

Source:
ISO 26262-5
Annex B
**FMEDA Creation Flow**

*SP level Analysis – one FMEDA per one FMEA line*

**Administrator – per Project**

- Provide Design Data
  - [Hierarchical data on all SPs – digital area, RAM bits, FF, Latches etc.]

**User/IP owner – per Sub Part**

- Specify SM type
  - [will provide initial Diagnostic coverage estimate]

- Update Fsafe percentage
  - [Fsafe is the portion of faults which go not violate the safety goal]

- Associate a design element
  - [If multiple FMEDAs on the same design element – use Scale Factor]

---

**Fault Simulation Campaign**

- Read back results

**Prepare a Fault Measurement campaign per FMEDA**

---

**Estimated ISO 26262 Metric**

**Measured ISO 26262 Metric**
Fault Injection Campaign

• Determine Diagnostic Coverage of the SM
  – inject faults in the design
  – checking if they are detected by the SM

• Fault simulators
  – Can use existing verification tests
  – Can run concurrently, handling many faults at a time
  – Stimulus may not be sufficient to cause all dangerous faults to propagate

• Formal tools
  – Can determine which faults are uncontrollable from the inputs
  – Can check for Observation points Cone Of Influence (COI) – observability of faults
Fault Classification Through Simulation

- **Safety Related**
  - F1 – Safe
  - F2 – Assumed Dangerous
  - F3 – Dangerous Detected
  - F4 – Dangerous Undetected

- **Non-Safety Related**

If a fault was not observed and/or detected (F2), it can be:
1. A safe fault
2. A dangerous fault which did not propagate due to insufficient stimulus

© Accellera Systems Initiative 55
Preparation for Fault Simulation

- Set up FMEDA observation points by analyzing failure mode effects
- Set up FMEDA test suite (get it from module/system verification suite)
- Set up FMEDA detection points according to safety mechanism

A fault which does not propagate to any observation point is either safe, or ‘dangerous undetected’

Workload should toggle the logic around the fault as much as possible

A detection point is the physical net which toggles as a result of the safety mechanism detecting the fault
Unfavorable Simulation Results Analysis

- A fault which does not propagate to any observation point is either safe, or ‘dangerous undetected’

- Use Formal tool to further classify faults

- Provide DUT and Fault list to VC-Formal
  - Fault proven to be not-controllable or not observable
    - Fault is Safe
  - Fault proven to be controllable and observable
    - Analyze scenario provided by VC-Formal and improve provided use case(s)
      \(\Rightarrow\) Productivity and safety increased
  - Inconclusive
    - Human analysis required

© Accellera Systems Initiative
Benefits of Formal Fault Filtering

• Simulation and constrained random tests help achieve high % of fault coverage quickly
• Eventually the benefits of simulation and manual directed tests diminish: progress plateau
• Formal filtering of faults can provide a boost to fault coverage %
Fault Injection Testing – Z01X Functional Safety

Highest performance fault simulation solution for ISO 26262 compliance requirements

- Challenges
  - Stringent fault simulation is needed for ISO 26262 compliance
  - Both permanent and transient fault model support is required
  - Performance demands are extreme

- Objective
  - Generate fault coverage metrics with acceptable TAT

- Results
  - MobilEye adopted Z01X for their EyeQ4 design when existing (competitive) solution was too slow
  - Z01X adoption WW is growing rapidly in automotive semiconductor and systems companies
FME/A/FMEDA Process Overview (ISO 26262)

SoC -> IP analysis

Create FMEA

Create FMEDA

Fault Coverage Measurement:
- Formal Fault Reduction
- Fault Simulation

Design Data, Technology Data

![Image of process flow diagram]
### MAIN FMEDA

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>Top Design Element</th>
<th>Element-1</th>
<th>Element-2</th>
<th>Potential Faults</th>
<th>Potential Errors (as soon at the top design element boundary)</th>
<th>Potential Effects of Failure (visible to the system)</th>
<th>System level potential effect class</th>
<th>Safety Tested?</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST_FM_1</td>
<td>MEM_CTRL</td>
<td></td>
<td></td>
<td>corrupted CPU command</td>
<td>Memory content corruption</td>
<td>Wrong coding, wrong or no execution</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_2</td>
<td>MEM_CTRL</td>
<td></td>
<td></td>
<td>corrupted CPU command</td>
<td>Memory content corruption</td>
<td>Wrong coding, wrong or no execution</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_3</td>
<td>MEM_CTRL</td>
<td></td>
<td></td>
<td>corrupted CPU write data</td>
<td>Memory content corruption</td>
<td>Wrong coding, wrong or no execution</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_4</td>
<td>MEM_CTRL</td>
<td></td>
<td></td>
<td>corrupted CPU write data</td>
<td>Memory content corruption</td>
<td>Wrong coding, wrong or no execution</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_5</td>
<td>REG_UNET</td>
<td></td>
<td></td>
<td>incorrect registers read</td>
<td>Memory content corruption</td>
<td>Processor architectural state/output corrupt</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_6</td>
<td>REG_UNET</td>
<td></td>
<td></td>
<td>incorrect registers read</td>
<td>Memory content corruption</td>
<td>Processor architectural state/output corrupt</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_7</td>
<td>REG_UNET</td>
<td></td>
<td></td>
<td>incorrect registers write</td>
<td>Memory content corruption</td>
<td>Processor architectural state/output corrupt</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
<tr>
<td>HOST_FM_8</td>
<td>REG_UNET</td>
<td></td>
<td></td>
<td>incorrect registers write</td>
<td>Memory content corruption</td>
<td>Processor architectural state/output corrupt</td>
<td>CPU/GPU</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Permanent Fault Model

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>Device</th>
<th>Application</th>
<th>Permanent</th>
<th>Application Diagnostic</th>
<th>Application Diagnostic ID</th>
<th>Application Diagnostic Frequency</th>
<th>Known</th>
<th>Known Applications</th>
<th>Known Applications Frequency</th>
</tr>
</thead>
</table>
| HOST_FM_1 | 9.378E+01 | 3.8E+01 | 3.098E+01 | 0.000 | 320 | 0% | 1.4E-01 | 2.4E-01 | 2.4E-01 | 99 | 99%

### Transient Fault Model

<table>
<thead>
<tr>
<th>Unique ID</th>
<th>Device</th>
<th>Application</th>
<th>Permanent</th>
<th>Application Diagnostic</th>
<th>Application Diagnostic ID</th>
<th>Application Diagnostic Frequency</th>
<th>Known</th>
<th>Known Applications</th>
<th>Known Applications Frequency</th>
</tr>
</thead>
</table>
| HOST_FM_1 | 9.378E+01 | 3.8E+01 | 3.098E+01 | 0.000 | 320 | 0% | 1.4E-01 | 2.4E-01 | 2.4E-01 | 99 | 99%

---

**Accellera Systems Initiative**

© Accellera Systems Initiative 61
ISO 26262 Metric Report

- Probabilistic Metric for random Hardware Failures (PMHF)
- Single-point fault metric (SPFM)
- Latent-fault metric (LFM)
Synopsys’ Unique Position for Automotive Verification

- Deep R&D collaboration with leading automotive semiconductor suppliers
- Automotive supply chain relationships with Tier1 and OEMs
- Fastest verification engines: Static, Formal, Simulation, Emulation, FPGA prototyping
- Early SW development platform with hybrid emulation and Verdi HW/SW debug
- Unique technologies: Certitude, Z01X, FMEDA automation, virtual prototyping and models
- ISO26262 technical expertise and experience
Questions?

Contact: jmforey@synopsys.com
Thank You