Making Autonomous Cars Safer – One chip at a time

Apurva Kalia: Vice President R&D
Ann Keffer: Product Management Director
Agenda

- Automotive Market
- Complex Challenges
- ISO 26262 and Basic Safety
- Functional Safety Methodology
The Automotive Market
Automotive Semiconductor Growth

Automotive semiconductor revenue by application

7.1% CAGR (2016 – 22)

Source: IHS Markit

© 2017 IHS Markit
“Automotive Revolution – Perspective towards 2030” – a 2016 McKinsey Report identified 4 areas that deemed particularly important in shaping the auto industry thru 2030

**Forces Shaping the Automotive Industry**

- **Vehicle electrification**
- **Increased Connectivity**
- **Growth of Autonomous Driving**
- **Shared Mobility Services**

**Advances to solve**
- High battery costs
- Proliferation of charging infrastructure

**Advances to**
- 5G deployment
- Telematics services
- V2I; V2V

**ADAS deployment**
- Cost effective Level 3 and Level 4 by 2020~2025

**Proliferation of**
- Ride sharing services
- Car sharing services
Autonomous Driving

- Amount of electronics is growing fast
- (ADAS) based on complex SoCs to enable high-performance computing
- Safety critical ADAS applications have stringent requirements on
  - Functional Safety
  - Security
  - Reliability
Automotive Opportunities and Focus Areas

**High-performance computing**
- Scalability
- High resolution
- Low power
- Vision + CNN
- Memory bandwidth
- Safety and Security is a must!

**Highly integrated cockpit**
- Scalability
- Connectivity
- In-vehicle networking
- SW app availability
- Comprehensive I/F support
- Basic ADAS features

**Qualification of new SoCs**
- Safety, Security and Reliability
- FMEDA not sufficient for SoCs
- Integrated FMEDA and safety verification flow
- Interfaces to RM & Tracing tools

---

© Accellera Systems Initiative
Complex Challenges
The Megatrends Dilemma

- Efficient Electric Vehicles
- Safe Autonomous Cars
- Reduce Emissions
- Enhanced Safety
- ADAS
- Connectivity
- Improved HMI
- Power
- Weight

Need low-power, small footprint, high-performance SoCs

Source: BMW
Source: Volvo

© Accellera Systems Initiative
Making a Car Autonomous

Audio
- Rear Object Detection
- Parking Assist/Auto Park
- Voice Recognition
- Cabin Noise Reduction
- Emergency Recognition
- Spatial Audio for Warnings

Radar
- Front Collision Avoidance Braking
- Adaptive Cruise Control
- 360 degree Hazard Awareness
- Rear Collision Detection

Fusion
- Radar, LIDAR, Image correlation
- System Functional Safety
- System Data Control

Active Vision (LiDAR)
- Adaptive Cruise Control
- Collision Avoidance
- Blind Spot Detection

Passive Vision
- Rear View Camera
- Vision Enhancement
- Auto Dimming Headlights
- Blind Spot Detection
- 360 View
- Parking Assist
- Lane Detection and Following
- Sign Recognition
- Traffic Signal Recognition
- Rain, Snow, Fog Removal
- Pedestrian Tracking/Avoidance
- Eye Focus Detection
- Driver Monitoring
- Vehicle Detection/Avoidance
Complicated Convolutional Neural Networks

Automated and Reliable Object Recognition using Convolutional Neural Networks

Need a high-performance, low-power hardware platform to combine and analyze point clouds and accurately identify objects.

Radar Point Cloud

~10-100 KB/sec

Lidar Point Cloud

~10-70 MB/sec

Digital Camera

~20-40 MB/sec

© Accellera Systems Initiative
Automotive SoC Verification Challenges

- Systematic Failure Verification
- Concurrent SW Development
- Requirements Traceability
- Use Case Verification
- Performance Verification
- Security Verification
- Automotive Protocol Verification
- Mixed Signal Verification
- Functional Safety Verification
- Random Failure Verification

ADAS SoC

Multiple verification and validation platforms
ISO 26262 and Failure Mode Effects and Diagnostic Analysis
ISO 26262 defines
• Processes to follow
• Hardware/software performance to achieve
• Safety documentation to produce
• Software tools compliance process
“Absence of unreasonable risk due to hazards caused by malfunctioning behavior of electrical and/or electronic systems” (ISO 26262)

Malfunction → How much harm can the malfunction cause? (risk) → What level of safety integrity (risk reduction) is needed?

- LOW
  - A Dashboard
  - B Airbag not firing
- HIGH
  - C
  - D Braking

ASIL examples for illustration purposes only
### ASIL Determination Example—ISO 26262

<table>
<thead>
<tr>
<th>Malfunction</th>
<th>ABS system failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Goal</td>
<td>Prevent ABS failure</td>
</tr>
</tbody>
</table>

#### Hazard Analysis
- What unintended situations (hazards) could happen? → Loss of stability on split-μ surface

#### Risk Analysis
- **How likely** is the hazard to happen? (Exposure) → oil spill, gravel, water potholes, ....
- **How harmful** is the hazard? (Severity) → Car may spin out of control and crash
- **How controllable** is the system if the hazard occur? (Controllability) → dashboard, driver

#### ASIL Determination
- What level of safety (risk reduction) does the system need?
- **How likely** can the malfunction be? → **FIT (Failure in Time)**
- **How often** does the system need to catch it and get to a safe situation? → **DC (Diagnostic coverage)**

### ASIL (Automotive Safety Integrity Level)

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

- ↓ FIT (Failure In Time), ↑ Diagnostic Coverage (DC)
ISO 26262—Design and Safety Flow

- Concept Phase (ISO-Part 3)
- System Design (ISO-Part 4)
- SW Design (ISO - Part 6)
- HW Design (ISO - Part 5)
- Technical Safety Requirements
- SW Technical Safety Requirements
- HW Technical Safety Requirements

FIT gets distributed from the item to each of the elements

© Accellera Systems Initiative
Break
ASIL Hardware Metrics

<table>
<thead>
<tr>
<th>ASIL</th>
<th>Failure Rate</th>
<th>SPFM</th>
<th>LFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&lt; 1000 FIT</td>
<td>Not relevant</td>
<td>Not Relevant</td>
</tr>
<tr>
<td>B</td>
<td>&lt; 100 FIT</td>
<td>&gt; 90%</td>
<td>&gt; 60%</td>
</tr>
<tr>
<td>C</td>
<td>&lt; 100 FIT</td>
<td>&gt; 97%</td>
<td>&gt; 80%</td>
</tr>
<tr>
<td>D</td>
<td>&lt; 10 FIT</td>
<td>&gt; 99%</td>
<td>&gt; 90%</td>
</tr>
</tbody>
</table>

- **FIT** Failure In Time (1 Failure / $10^9$ hours)
  - PMHF Probabilistic Metric for Random HW failures
- **SPFM** Single Point Fault Metric
- **LFM** Latent Fault Metric

- **SAFETY GOAL VIOLATION**
- **FIT** Failure In Time
- **PMHF** Probabilistic Metric for Random HW failures
- **SPFM** Single Point Fault Metric
- **LFM** Latent Fault Metric
- **FIT** Failure In Time
- **PMHF** Probabilistic Metric for Random HW failures
- **SPFM** Single Point Fault Metric
- **LFM** Latent Fault Metric

© Accellera Systems Initiative 19

© Accellera Systems Initiative 19
Functional Safety Life Cycle Main Tasks

- Silicon provider is asked to execute five main activates to implement a Functional Safety life cycle in light of the hardware random capability.

- **Safety Manager** is the person in charge to define and track the Functional Safety process, define the work products, define the template documentation and execute internal reviews.

  - **Life Cycle**
    - Selection of ISO26262 Process Requirements and tailoring of the development process for the specific SoC (Safety Manager).
  
  - **Safety Concept**
    - Assumed Safety Requirements definition for the HW component for the Development of the SoC (Safety Architect).
  
  - **Safety Analysis**
    - Safety Analysis: FMEA/FMEDA/DFA (Safety Engineer).
  
  - **Metrics Computation**
    - Compute Hardware Architecture Metrics (SPFM, LFM), PMHF based on the defined Safety Concept (Safety Engineer).
  
  - **Reviews/Confirmations**
    - Perform applicable Verification Reviews, Confirmation Reviews, Safety Audit and Assessment (Auditor).
ISO26262—Functional Safety Principles

Systematic Failures (e.g., software bug)
- Addressed by processes (planning, traceability, documentation, specs, …)
- Strictness of processes are dependent on the ASIL level

Random Failures (e.g., component malfunction, noise injection)
- Considers permanent failure and transient effects
- Includes safety mechanisms design and integration to handle faults
- Demonstrated by calculations of Reliability/verification of failure rates
- Failure rates and diagnostic coverage requirement depend on ASIL

Design/Analysis
Verification

ISO 26262 covers random and systematic errors

© Accellera Systems Initiative
Functional Safety Metrics

- Target metrics values according to ASIL (Automotive Safety Integrity Level)
- Architectural Matrices (measured in %)

  - **SPFM**: Single Point Fault Metrics
    The single point fault metric reveals whether or not the coverage by the safety mechanisms (i.e. the DC), to prevent risk from single point faults in the hardware architecture, is sufficient. Single point faults are faults in an element that leads directly to the violation of a safety goal → **SPFM high means that the set of Safety Mechanisms have high capacity to cover dangerous faults, resulting in high DC.**

  - **LFM**: Latent Fault Metrics
    The latent fault metric reveals whether or not the coverage by the safety mechanisms, to prevent risk from latent faults in the hardware architecture, is sufficient. Latent faults are multiple-point faults whose presence are not detected by a safety mechanism. Latent faults become dangerous when a second faults appears and it will be not detected due to the latent fault previously occurred → **LFM high means that the set of Safety Mechanism have high capability to cover multiple faults (multiple = 2) scenario.**
Functional Safety Metrics

• Absolute Metrics
  – **PMHF**: Probabilistic Metric for (Random) Hardware Failures

  *Is the sum of the single point, residual and multipoint fault metrics. Is expressed in FITs* → **PMHF** low means a low probability that the SoC, including its safety mechanisms, fails without any detection. It is measured in FIT: 1FIT = probability that one failure occur in 10^9 hours. It represents the probability to violate the safety goal
# Work Products and Documentations

- List of the most relevant documents to be produced during a Functional Safety Development and to be used during an assessment

<table>
<thead>
<tr>
<th>Work Products</th>
<th>Content</th>
<th>ISO26262 References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Plan</td>
<td>Company related process quality standards, product life cycle, product responsibilities, tools qualification, project activities plan, ...</td>
<td>ISO 26262-2:2018, 6.4.3.9</td>
</tr>
<tr>
<td>Configuration Management Plan</td>
<td>Process to control that work products can be uniquely identified and reproduced in a controlled manner at any time, e.g. bugs tracking and documentation</td>
<td>ISO 26262-8:2018</td>
</tr>
<tr>
<td>Change Management Plan</td>
<td>Process to changes to safety-related work products throughout the safety lifecycle, impact analysis, revisioning, ...</td>
<td>ISO 26262-8:2018</td>
</tr>
<tr>
<td>Safety Requirements</td>
<td>Design and safety mechanisms requirements compliant with technical safety report and system requirements (traceable)</td>
<td>ISO 26262-5:2018, Clause 6</td>
</tr>
<tr>
<td>Requirements traceability report</td>
<td>Show the traceability backward and forward of the requirements.</td>
<td>ISO 26262-5:2018 - 7.4.2.5</td>
</tr>
<tr>
<td>HW Design Verification Plan</td>
<td>Description of the techniques and measures to avoid systematic capability: the pass and fail criteria for the verification, the metrics; the verification environment; the tools used for verification; the regression strategy.</td>
<td>ISO 26262-11:2018, 5.1.9 - table 30</td>
</tr>
<tr>
<td>HW Design Verification Report</td>
<td>Results of the verification measures (typically metrics driven verification), derogation, ...</td>
<td>ISO 26262-8:2018, Clause 9</td>
</tr>
<tr>
<td>Safety Analysis Report</td>
<td>FMEA, FMEDA. Safety scope description, Base failure rate calculation, Fault models applied, Analysis assumptions, Analysis results, Fault injection strategy (how to execute the measures, which WL, sampling, ... expert Judgment evidences ...</td>
<td>ISO 26262-9:2018, Clause 8</td>
</tr>
<tr>
<td>Analysis of Dependent Failure report</td>
<td>DFA analysis, assumption, adopted measures and results</td>
<td>ISO 26262-9:2018, 4.6</td>
</tr>
<tr>
<td>Confirmations Measure Reports</td>
<td>Confirmation reviews of: safety plan, safety analysis, software tool criteria evaluation report, completeness of the safety case, ...</td>
<td>ISO 26262-2:2018, Table 1</td>
</tr>
<tr>
<td>Safety Manual</td>
<td>Applied Safety Life Cycle, safety goal, safety scope, AoU description, fault models, Safety Mech. Description, Safety results summary, ...</td>
<td>ISO 26262-11, 4.5.4.9</td>
</tr>
</tbody>
</table>
FMEDA Analysis

- User defines the FMEDA Hierarchy starting from design requirements
- Part and Subpart are not one by one with the physical implementation

### FMEDA Hierarchy

<table>
<thead>
<tr>
<th>ID</th>
<th>PART</th>
<th>SUBPART</th>
<th>Failure Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BUS_ITF</td>
<td></td>
<td>Wrong Data Transaction caused by a fault in the AHB interface</td>
</tr>
<tr>
<td>2</td>
<td>DECODER</td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the decode logic</td>
</tr>
<tr>
<td>3</td>
<td>VIC</td>
<td></td>
<td>Un-intended execution/not executed interrupt request</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>Corrupt data or value caused by a fault in the register bank shadow</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>Incorrect Instruction Result caused by a fault in the multiplier</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>Incorrect Instruction Result caused by a fault in the adder</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>Incorrect Instruction Result caused by a fault in the divider</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>Corrupt data or value caused by a fault in the register bank</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the pipeline controller</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the branch logic (Wrong Branch Prediction)</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the fetch logic</td>
</tr>
</tbody>
</table>

### Design Hierarchy: from requirements

- CPU
  - ALU
  - FETCH
  - FETCH_UNIT
  - FSM_PIPE
  - BRANCH_BUFFER
  - BRANCH_FSM
  - VICT_INT
  - VICT_CTRL
  - DEC_HI
  - DEC_LO
  - BUS_IF
**FMEDA Analysis**

- User provides textual description of the FMs (for every subpart) figured-out during the failure functional analysis

<table>
<thead>
<tr>
<th>ID</th>
<th>PART</th>
<th>SUBPART</th>
<th>Failure Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BUS_ITF</td>
<td></td>
<td>Wrong Data Transaction caused by a fault in the AHB interface</td>
</tr>
<tr>
<td>2</td>
<td>DECODER</td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the decode logic</td>
</tr>
<tr>
<td>3</td>
<td>VIC</td>
<td></td>
<td>Un-intended execution/not executed interrupt request</td>
</tr>
<tr>
<td>4</td>
<td>CPU</td>
<td>ALU</td>
<td>Corrupt data or value caused by a fault in the register bank shadow</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>Incorrect Instruction Result caused by a fault in the multiplier</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>Incorrect Instruction Result caused by a fault in the adder</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>Incorrect Instruction Result caused by a fault in the divider</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>Corrupt data or value caused by a fault in the register bank</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the pipeline controller</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>FETCH</td>
<td>Incorrect Instruction Flow caused by a fault the branch logic (Wrong Branch Prediction)</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>Incorrect Instruction Flow caused by a fault the fetch logic</td>
</tr>
</tbody>
</table>

**FM definition:** comes from a cause-effect user analysis starting from specs or RTL

FM4: “Corrupt data or value caused by a fault in the register bank shadow”

e.g. The ALU function has six different way to fail
FMEDA Validation

- **FM mapping** is performed by the user associating FMs (defined into the FMEDA) to Design Instances (hierarchical full path name)

Design Hierarchy: instances full path names

---

<table>
<thead>
<tr>
<th>MODULES</th>
<th>DES INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not Stuctural</td>
</tr>
<tr>
<td>bus_if</td>
<td></td>
</tr>
<tr>
<td>dec</td>
<td></td>
</tr>
<tr>
<td>vic_int</td>
<td></td>
</tr>
<tr>
<td>vic_ctrl</td>
<td></td>
</tr>
<tr>
<td>reg_shadow</td>
<td></td>
</tr>
<tr>
<td>add</td>
<td></td>
</tr>
<tr>
<td>mul</td>
<td></td>
</tr>
<tr>
<td>div</td>
<td></td>
</tr>
<tr>
<td>reg_bank</td>
<td></td>
</tr>
<tr>
<td>fsm_pipe</td>
<td></td>
</tr>
<tr>
<td>branch_fsm</td>
<td></td>
</tr>
<tr>
<td>branch_buffer</td>
<td></td>
</tr>
</tbody>
</table>

**tot** | **12753** | **413** |
FMEDA Validation

• Before executing the fault injection campaigns an FMEDA Plan shall be finalized

• The FMEDA validation is executed on a FM basis, meaning that a specific fault campaign is executed for every FM.

• The user supplies, still on a FM basis, observation points and detection points according to the verification requirements supplied by the safety engineer
Use Cases

• When the SoC complexity grows a **modular approach** is required to initiate an FMEDA and execute its validation.

• An FMEDA **team based approach** should be also supported to allow splitting the job among different teams, enabling an IP-based methodology.

• IP could be provided from **3rd party IP provider** and will come with its own FMEDA.
Functional Safety Methodology
Integrate Safety Mechanisms to reduce the FIT

Positive testing (functional verification)
  – Verify proper functionality prior to safety verification

Negative testing (assess diagnostic capability):
  – Targeted tests to confirm failure mode assumptions
  – Statistical tests to ensure design function integrity
  – Transient faults testing to provide evidence safety mechanisms integrity
Current Need

- A dedicated functional safety verification methodology and process for these safety-critical IPs and SoCs
- Safety analysis in semiconductor such as fault injection, fault metrics, base failure rate estimation, interfaces within distributed developments, handling of Hardware Intellectual Property (IP)

Methodology

- Holistic methodology which combines analytical methodologies such as FMEDA with dynamic fault simulation and formal analysis based methodologies to significantly reduce the safety verification effort and achieve faster product certification

Metrics

- ISO26262 recommends single point fault metric (SPFM) and Latent Fault Metric (LFM) for the component (IP and SoCs)
- Will be measured for each of the identified Safety Goals associated with the safety critical modules within the IPs and/or SoCs.
Safety Verification Challenges and More

- Tool Confidence Level (TCL)
- Safety Certified IPs
- Safety Requirement Traceability
- Safety Mechanism Design
- Fault Campaign Planning
- Fault Set (+Optimization) Execution
- Verification Environment Re-use
- Multiple Engines Support
- Link to FMEDA (Metrics Calculation)

ADAS SoC Example

Tool Confidence Level (TCL)

Safety Certified IPs

© Accellera Systems Initiative
Break
## Safety Verification Methodology

**FMEDA FS architecture analysis key to reducing overall FS efforts**

**Start serial fault injection early on RTL. Reuse same TB and coverage**

**Common fault coverage DB to integrate results across engines**

<table>
<thead>
<tr>
<th>Months</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
<th>Jan</th>
<th>Feb</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IP/Subsystem D&amp;V</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SoC Integration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SoC Implementation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SoC Verification</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FMEDA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fault Campaign Planning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IP/SS Serial Fault Sims</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SoC RTL Fault Sims</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SoC GL Fault Sims</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SoC SW Driven Long tests</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Start early with FMEDA, Fault Campaign planning, and flow set up
- RTL (serial) fault campaigns to clean flow, select tests, & debug
- Lesser faults, RTL sim is faster
- Complete fault campaigns at gate level for signoff (concurrent)
- Hardware accelerated fault simulation
Typical Functional Safety Workflow

1. **Safety Design + FMEDA Analysis**
   - Verification Plan + Test bench

2. **Traceability and Verification Management**
   - Define Failure Modes
   - FMEDA Analysis
   - Design SMs
   - Design Information
   - FIT/DC estimation
   - Goals met? (ASIL)

3. **Fault List Management**
   - Fault list generation
   - Fault list optimization
   - Fault injection

4. **Metrics**
   - Metrics met?
   - Safety report

5. **Fault Campaign Management**
   - Use new tests/patterns
   - Add SMs

© Accellera Systems Initiative
Safety Verification Solution

- Unified functional + safety verification flow and engines
- Integrated fault campaign management across formal, simulation, and emulation
- Common fault results database unifies diagnostic coverage
- Proven requirements traceability, enabling FMEDA integration
Example Design and FMEDA
Safety Mechanisms in Ethernet IP

- Parity or ECC on packet/descriptor buffer
- Parity or redundancy of CSRs
- Failure status interrupt
- DMA descriptor address range checking
- Parity protection @ timestamp generation
- Redundancy compare
- IP/TCP checksum
- Illegal packet filter
- Ethernet frame FCS
- Anti-lockup watchdog
- Loopbacks

© Accellera Systems Initiative
# GEM Block – FMEDA Analysis

<table>
<thead>
<tr>
<th>Block or Subblock</th>
<th>$\lambda$ [FIT]</th>
<th>Failure Mode</th>
<th>FM Distribution</th>
<th>Effect Description of FM</th>
<th>SM Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU compare pulse</td>
<td>0.9%</td>
<td>TSU compare interrupt is incorrect</td>
<td>Compare logic is duplicated</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU seconds increment pulse</td>
<td>0.9%</td>
<td>The TSU seconds interrupt is incorrect</td>
<td>Interrupt logic is duplicated</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in generation of the TSU strobe pulse to the registers</td>
<td>0.9%</td>
<td>The timer value may not be captured or captured incorrectly</td>
<td>Strobe Pulse Logic is duplicated</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU timer output value</td>
<td>97.3%</td>
<td>TX/RX timestamp is corrupted, output TSU timer value to local system will be invalid, Timer value read back in registers is also invalid.</td>
<td>Timer logic is duplicated</td>
</tr>
<tr>
<td>Registers</td>
<td>0.3013</td>
<td>Fault in static configuration outputs from the registers</td>
<td>95%</td>
<td>Unpredictable behavior of IP</td>
<td>Parity generation and detection</td>
</tr>
</tbody>
</table>
## GEM Block – FMEDA Verification

<table>
<thead>
<tr>
<th>Block or Subblock</th>
<th>( \lambda ) [FIT]</th>
<th>Failure Mode</th>
<th>FM Distribution</th>
<th>DC Number Estimated</th>
<th>DC Number Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU compare pulse</td>
<td>0.9%</td>
<td>95%</td>
<td>96%</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU seconds increment pulse</td>
<td>0.9%</td>
<td>95%</td>
<td>98%</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in generation of the TSU strobe pulse to the registers</td>
<td>0.9%</td>
<td>95%</td>
<td>78%</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU timer output value</td>
<td>97.3%</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td>Registers</td>
<td>0.3013</td>
<td>Fault in static configuration outputs from the registers</td>
<td>95%</td>
<td>90%</td>
<td>92.5%</td>
</tr>
</tbody>
</table>
Demo
Safety Mechanisms in Ethernet IP

- Parity or ECC on packet(descriptor buffer
- Parity or redundancy of CSRs
- Failure status interrupt
- DMA descriptor address range checking
- Parity protection @ timestamp generation
- Redundancy compare
- IP/TCP checksum
- Illegal packet filter
- Ethernet frame FCS
- Anti-lockup watchdog
- Loopbacks
- Ethernet frame FCS
ADAS Platform - SM Enabled

1. Standalone as Out of Context
2. SM Enabled Automotive Ethernet IP as in context
3. Top Level SM’s ECC Enabled Memories, Clock Monitors, Bus Lock UP’s outside the IP Boundaries

Boot ROM Self Tests – Checksum, Periodic Tests in Software
Software Enabled Safety Mechanism

Timer Duplication

CSI Tx CAM
RGB TBA Display

1. ECC Enabled Memories for VP5

NOC SM’s Lock Up’s ECC

SM Enabled Blocks
GEM Block Diagram – SM View

- **EDMA**
  - EDMA RX MEM
  - EDMA TX MEM
  - EDMA TX DMA
  - EDMA RX DMA

- **MAC**
  - MAC L3/L4
  - MAC TSN
  - MAC TX MAC
  - MAC RX MAC

- **Registers**

- **Time Stamp Unit (TSU)**

- **APB**

- **AXI**
  - AXI TX INTF
  - AXI TX DMA
  - AXI RX DMA

- **DMA Descriptor Address Data Protection**

- **ECC Protection**

- **Data Path Parity Protection**

- **Data Path Lockup Protection**

- **Parity Protection**

- **Redundancy Protection**

- **Lockup, bus response errors, underrun, overrun protection**

- **IP/TCP Checksum Illegal Packet Filter Ethernet Frame FCS**

- **Access Protection**

- **Loopback**

**Interfaces:**
- GMII
- TBI
- RGMI I
- RGMII
Fault Classification

1. Dangerous Detected (D) – Faults observable on both Functional & SM checker Output
2. Dangerous Undetected (U) – Faults observable on Functional Outputs but not detected by SM checker
3. Safe Faults (UT) – Faults not observable on Functional Outputs & SM Checker outputs
ISO26262 Compliant Fault Classification

Total faults

Formal

OBS

S

• Architectural
• Functional

Remaining faults

Fault Injection

OBS

NC
(remaining faults not classified)

Work Load

 WL patterns.

Fault Injection

Dangerous (violate the SG)

DD, DU

S

DD', DU', S'

DC%, S%

EXPERT JUDGMENT

Optional. If applied the user shall to provide additional evidences in place of fault injection and formal analysis to justify the expert judgment

Agenda:
DD: dangerous detected faults
DU: dangerous undetected faults
S: Safe faults (not violating the safety goal)
DC: Diagnostic Coverage
NC: Not classified as S, DD or DU

Calculated per Failure Mode
Demo Setup and Run the VNC

- Chosen 5 Failure modes for the demo showcasing the solution and automation capabilities
  - fm_tsu_comp_pulse - Fault in TSU compare pulse – show cases the ranking capability and undetected faults as SM is not implemented
  - fm_tsu_tmr_op_val - Fault in TSU timer output value - show cases the ranking capability and detected faults as SM is implemented
  - fm_tsu_sec_incr - Fault in TSU seconds increment pulse – run campaign for the module TSU
  - fm_tsu_tmr_op_val_samp - Fault in generation of the TSU strobe pulse to the registers - run campaign for the module TSU

<table>
<thead>
<tr>
<th>Block or Subblock</th>
<th>λ [FIT]</th>
<th>Failure Mode</th>
<th>FM Distribution</th>
<th>Effect Description of FM</th>
<th>SM Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU compare pulse</td>
<td>0.9%</td>
<td>TSU compare interrupt is incorrect</td>
<td>Incomplete</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU seconds increment pulse</td>
<td>0.9%</td>
<td>The TSU seconds interrupt is incorrect</td>
<td>Incomplete</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in generation of the TSU strobe pulse to the registers</td>
<td>0.9%</td>
<td>The timer value may not be captured or captured incorrectly</td>
<td>Incomplete</td>
</tr>
<tr>
<td>TSU</td>
<td>0.0719</td>
<td>Fault in TSU timer output value</td>
<td>97.3%</td>
<td>TX/RX timestamp is corrupted, output TSU timer value to local system will be invalid, Timer value read back in registers is also invalid.</td>
<td>Timer is duplicated</td>
</tr>
<tr>
<td>Registers</td>
<td>0.3013</td>
<td>Fault in static configuration outputs from the registers</td>
<td>95%</td>
<td>Unpredictable behavior of IP</td>
<td>Parity generation and detection</td>
</tr>
</tbody>
</table>
Fault Campaign Executor - Interface

Inputs: FMEDA info
- Fault List
  - Definition of the faults to be injected
- Strobe List
  - Definition of the observation points

Inputs: FS Verification Engineer
- Test List
  - Tests to be used during the campaign
- Campaign Configuration:
  - Define the campaign parameters

Outputs:
- Annotated Fault List
  - Fault classification is back annotated
- Reports
  - Various kind according to the use case
Fault Campaign Executor - Interface

- **Test selection**
  - Execute the user defined list of tests
- **Good Simulation**
  - Fault instrumentation
  - Generate strobe data for each selected test
- **Fault Simulation Setup**
  - Prepare fault simulation including static and dynamic (formal) fault set optimization
- **Fault Simulation Execution**
  - Simulate each fault with the selected tests
Campaign Reports - Abstract

Test Coverage = \((\frac{D}{D+U})\)

Fault Coverage = \((\frac{D}{D+U+UT})\)

- Safe Faults by Formal
- Possibly need to improve work load
- Detected Faults
- Test Coverage = \((\frac{D}{D+U})\)
- Fault Coverage = \((\frac{D}{D+U+UT})\)
# Campaign Reports - Abstract

**Sampled Fault Processed**

\[
\text{Test Coverage} = \frac{D}{D+U}
\]

\[
\text{Fault Coverage} = \frac{D}{D+U+UT}
\]
Summary
Summary

• Autonomous cars are coming and ‘Mind-Off’ driving is expected to be real by the mid 2020s
• ISO 26262 is the automotive standard that defines the processes to follow, the performance level for hardware and software performance and the compliance process
• A systematic analysis technique such as the FMEDA is essential for meeting ISO 26262 metrics
• The complexity of ADAS SoCs requires a new holistic approach to functional verification and functional safety
  • Functional safety and functional verification are complementary problems
  • A multi-engine automated solution is required to meet ASIL certification goals in a timely manner.
Questions
DVCon Slide Guidelines

• Use Arial or Helvetica font for slide text
• Use Courier-new or Courier font for code
• First-order bullets should be 24 to 28 point
  – Second-order bullets should be 24 to 26 point
    • Third-order bullets should be 22 to 24 point
    • Code should be at least 18 point
• Your presentation will be shown in a very large room
  – These font guidelines will help ensure everyone can read your slides!

No Company Logo except on title slide!

3/2/2022
Change "footer" to presenter's name and affiliation
module example
(input logic foo,
output logic bar);

initial begin
$display("Hello World!");
endmodule