AA129 - Reliability in Embedded Systems
Safety Standards and Self Tests

Christopher Temple
Automotive Systems Technology Manager
Overview

► Introduction
► IEC61508 Safety Standard
► ISO26262 Safety Standard (draft)
► MCU Safety Continuum
► Basic Core Self-Test
► Summary
Introduction
Freescale Introduces Product Longevity Program

► The embedded market needs long-term product support, which allows OEMs to provide assurance to their customers
► Freescale has a longstanding track record of providing long-term production support for our products
► Freescale is pleased to introduce a formal product longevity program for the market segments we serve
  • For the automotive and medical segments, Freescale will manufacture select devices for a minimum period of 15 years
  • For all other market segments in which Freescale participates, Freescale will manufacture select devices for a minimum period of 10 years
► A list of applicable Freescale products is available at www.freescale.com.
Automotive Safety and Functional Safety

Functional Safety Systems

Automotive Safety Systems
Active Safety Systems
Passive Safety Systems

“Safety of persons in respect to accidents”

“Safety of persons in respect to (E/E/PES) system failures”

“So safety is freedom from unacceptable risk”

(IEC 61508)
Evolution of Functional Safety Approaches

Functionally safe systems via first generation electronics: Discrete redundancy

Functionally safe systems via second generation electronics: Highly integrated systems
From Components to Integrated Systems

► In the past strong separation of system level and discrete component level
► Good visibility of structural system details

► Now complete systems are being condensed to single integrated components
► Structural system details embodied in integrated components
Semiconductor manufacturers are moving towards safety systems suppliers.

Industry-wide cooperation and standardization emerging to harmonize system related aspects across industry.
Industry-wide Initiatives for Standards

Key Safety Standards
- IEC61508 (today)
- IEC61508 v2 (emerging)
- ISO26262 (emerging)

Industry Wide Initiatives
- AUTOSAR
- JasPar
- FlexRay™ Networking
- PSI5
- EC Funded SPARC, EASIS
Role of Safety Standards

- Standards are emerging as a framework to establish metrics and value network
- **IEC61508**
  - V1 since late 1990s, V2 announced
  - Safety lifecycle defined
  - Recommended and mandatory practices
- **ISO26262**
  - Current draft, release expected ~2011
  - Refinement of IEC61508 to comply with needs specific to the application sector of E/E systems within road vehicles
IEC61508 Safety Standard
The Seven Parts of IEC 61508

1: General Requirements

2: Requirements for electrical / electronic / programmable electronic safety-related systems (means HW)

3: Software Requirements

4: Definitions and abbreviations

5: Examples of methods for the determination of safety integrity levels

6: Guidelines on the application of IEC 61508-2 and IEC 61508-3

7: Overview of techniques and measures
How does IEC61508 define Functional Safety?

► Safety
  • “freedom from unacceptable risk”

► Risk
  • “combination of the probability of occurrence of harm and the severity of that harm”

► Harm
  • “physical injury or damage to the health of people either directly or indirectly as a result of damage to property or to the environment”

► Functional safety
  • “part of the overall safety relating to the equipment under control (EUC) and the EUC control system which depends on the correct functioning of the electrical/electronic/programmable electronic (E/E/PE) safety-related systems, other technology related safety-related systems and external risk reduction facilities”
Quantitative Requirements of IEC61508

IEC 61508

- Four Safety Integrity Levels (SIL)
- Two key metrics
  - Probability of dangerous failure per hour (PFH)
  - Safe Failure Fraction (SFF)
- Hardware redundancy in formulas (HFT)

<table>
<thead>
<tr>
<th></th>
<th>SIL 1</th>
<th>SIL 2</th>
<th>SIL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFH [1/h]</td>
<td>&lt;10⁻⁵</td>
<td>&lt;10⁻⁶</td>
<td>&lt;10⁻⁷</td>
</tr>
<tr>
<td>SFF (HFT=0)</td>
<td>&gt;=60%</td>
<td>&gt;=90%</td>
<td>&gt;=99%</td>
</tr>
<tr>
<td>SFF (HFT=1)</td>
<td>-</td>
<td>&gt;=60%</td>
<td>&gt;=90%</td>
</tr>
</tbody>
</table>

Note: Table adopted for typical automotive application
Quantitative Requirements of IEC61508

► IEC 61508
  • Four Safety Integrity Levels (SIL)
  • Two key metrics
    ▪ Probability of dangerous failure per hour (PFH)
    ▪ Safe Failure Fraction (SFF)
  • Hardware redundancy in formulas (HFT)

<table>
<thead>
<tr>
<th>SIL 1</th>
<th>SIL 2</th>
<th>SIL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFH [1/h]</td>
<td>&lt;10^-5</td>
<td>&lt;10^-6</td>
</tr>
<tr>
<td>SFF (HFT=0)</td>
<td>&gt;=60%</td>
<td>&gt;=90%</td>
</tr>
<tr>
<td>SFF (HFT=1)</td>
<td>-</td>
<td>&gt;=60%</td>
</tr>
</tbody>
</table>

Note: Table adopted for typical automotive application

Safety Integrity Levels

► SIL: “discrete level for specifying the safety integrity requirements of the safety functions to be allocated to the E/E/PE safety-related systems, where safety integrity level 4 has the highest level of safety integrity and safety integrity level 1 has the lowest”

► Approaches to determine the SIL
  • Quantitative methods: such as via probability of a dangerous failure per hour for continuous mode of operation
  • Qualitative methods: such as risk graph or hazardous event severity matrix
Quantitative Requirements of IEC61508

IEC 61508
- Four Safety Integrity Levels (SIL)
- Two key metrics
  - Probability of dangerous failure per hour (PFH)
  - Safe Failure Fraction (SFF)
- Hardware redundancy in formulas (HFT)

<table>
<thead>
<tr>
<th></th>
<th>SIL 1</th>
<th>SIL 2</th>
<th>SIL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFH [1/h]</td>
<td>&lt;10^{-5}</td>
<td>&lt;10^{-6}</td>
<td>&lt;10^{-7}</td>
</tr>
<tr>
<td>SFF (HFT=0)</td>
<td>&gt;=60%</td>
<td>&gt;=90%</td>
<td>&gt;=99%</td>
</tr>
<tr>
<td>SFF (HFT=1)</td>
<td>-</td>
<td>&gt;=60%</td>
<td>&gt;=90%</td>
</tr>
</tbody>
</table>

Note: Table adopted for typical automotive application

Key Metrics
- Probability of dangerous failure per hour (PFH)
  - Target values depend on mode of system (low demand versus high demand/continuous), complexity of system (Type A (simplex) versus Type B (complex)) and additional customer requirements
- Safe Failure Fraction
  - the ratio of the average rate of safe failures plus dangerous detected failures of the system to the total average failure rate of the system
Safe Failure Fraction and Diagnostic Coverage

<table>
<thead>
<tr>
<th>Safe, detected</th>
<th>Safe, undetected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dangerous, detected</td>
<td>Dangerous, undetected</td>
</tr>
</tbody>
</table>

Safe Failure Fraction = \frac{\sum \lambda_S}{\sum \lambda_S + \sum \lambda_{DD} + \sum \lambda_{DU}}

Note: SFF is computed from the rates (approx. probabilities) of the different failure classes

- SFF = \frac{\sum \lambda_S + \sum \lambda_{DD}}{\sum \lambda_S + \sum \lambda_{DD} + \sum \lambda_{DU}}
- Where:
  - \sum \lambda_S: total rate of safe failures
  - \sum \lambda_{DD}: total rate of dangerous detected failures
  - \sum \lambda_{DU}: total rate of dangerous undetected failures
Outline for Designing a Safe System

Hazard Analysis
Which unintended situations (hazards) can occur?

Risk Analysis
How likely is a hazard?
How dangerous is a hazard?
How controllable is the system in case of a hazard?

Safety Integrity Level 1..4
Dangerous failure rate $\lambda_{du}$
Diagnostic Coverage $DC$
Safe Failure Fraction $SFF$

Safety Function Requirements
How to mitigate the hazards?

Safety Integrity Requirements
Are the safety functions executed correctly?
Outline for Designing a Safe System

Hazard Analysis
Which unintended situations (hazards) can occur?

Risk Analysis
How likely is a hazard?
How dangerous is a hazard?
How controllable is the system in case of a hazard?

Safety Integrity Level 1..4
Dangerous failure rate $\lambda_{du}$
Diagnostic Coverage DC
Safe Failure Fraction SFF

Safety Function Requirements
How to mitigate the hazards?

Safety Integrity Requirements
Are the safety functions executed correctly?

Refine the system until the remaining risk is below the highest acceptable risk
What the Standard Says for Hardware Components

<table>
<thead>
<tr>
<th>Hardware safety integrity</th>
<th>Systematic safety integrity</th>
<th>Avoidance of systematic failures during the different phases of the lifecycle (relating to processes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Faults or failures to be analyzed in the derivation of safe failure fraction</td>
<td>Techniques and measures to control:</td>
<td>Recommendations to avoid mistakes:</td>
</tr>
<tr>
<td>▶ Faults or failures to be detected during operation</td>
<td>► systematic failures caused by hardware and software design</td>
<td>► during specification of E/E/PES requirements</td>
</tr>
<tr>
<td></td>
<td>► systematic failures caused by environmental stress or influences</td>
<td>► during E/E/PES design and development</td>
</tr>
<tr>
<td></td>
<td>► systematic operational failures</td>
<td>► during E/E/PES integration</td>
</tr>
<tr>
<td></td>
<td>► Recommended</td>
<td>► during E/E/PES operation and maintenance procedures</td>
</tr>
<tr>
<td></td>
<td>► Highly recommended</td>
<td>► during E/E/PES safety validation</td>
</tr>
<tr>
<td></td>
<td>► Mandatory measures</td>
<td></td>
</tr>
<tr>
<td>Guidelines for assessing the maximum diagnostic coverage considered achievable through various techniques</td>
<td>Guidelines for assessing the effectiveness of techniques and measures to control systematic failures</td>
<td>Guidelines for assessing the effectiveness of techniques and measures to avoid systematic failures</td>
</tr>
<tr>
<td></td>
<td>► Recommended</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► Highly recommended</td>
<td></td>
</tr>
<tr>
<td></td>
<td>► Mandatory techniques</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

► Applying all measures to achieve hardware safety integrity for a specific Safety Integrity Level would make a system far too expensive
► The right choice of measures is required
► (Effective!) use of error detection and diagnostic capabilities to detect dangerous failures
  • Error detection measures
    ▪ Stop errors from propagating beyond component boundary
    ▪ Error correction (compensation)
    ▪ Shut down (fail-silent)
  • Self test measures
    ▪ Ensure that the device is free from dormant faults
    ▪ Software self-test, various BIST mechanisms
ISO26262 Safety Standard (draft)
The Nine Parts of ISO26262

► ISO 26262 is the adaptation of IEC61508 in automotive industry
► ISO 26262 applies to safety related E/E systems installed in road vehicles of class M, N and O (see 70/156/EC)
► ISO 26262 consists of the following parts:

• Part 1: Glossary
• 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-oriented and safety-oriented analyses (analysis techniques)
Objective

ISO 26262 addresses hazards caused by safety related E/E systems due to malfunctions, excluding nominal performances of active and passive safety systems

- Provides an automotive safety lifecycle (management, development, production, operation, service, decommissioning) and supports tailoring the necessary activities during these lifecycle phases
- Provides an automotive specific risk-based approach for determining risk classes (Automotive Safety Integrity Levels, ASILs)
- Uses ASILs for specifying the item's necessary safety requirements for achieving an acceptable residual risk
- Provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety being achieved
Quantitative Requirements ISO26262

ISO 26262
- Four Automotive SILs (ASIL)
- Three key metrics
  - Probability of violation of safety goals (PVSG)
  - Single Point Fault Metric
  - Latent Fault Metric
- Hardware redundancy in structural modeling

<table>
<thead>
<tr>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVSG [1/h]</td>
<td>&lt;10⁻⁷ (recom.)</td>
<td>&lt;10⁻⁷</td>
</tr>
<tr>
<td>SPFM</td>
<td>&gt;90%</td>
<td>&gt;97%</td>
</tr>
<tr>
<td>LFM</td>
<td>&gt;60%</td>
<td>&gt;80%</td>
</tr>
</tbody>
</table>

Automotive Safety Integrity Levels

ASIL: “One of four classes to specify the item's necessary safety requirements for achieving an acceptable residual risk with D representing the highest and A the lowest class”

Approaches to determine the ASIL
- Focus on qualitative methods: such as risk graph or hazardous event severity matrix → see next slide
## Determining Required ASIL

<table>
<thead>
<tr>
<th>Classes of severity</th>
<th>Classes of probability of exposure regarding operational situations</th>
<th>Classes of controllability (by driver)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>C1 (simple)</td>
</tr>
<tr>
<td>S1 Light and moderate injuries</td>
<td>E1 (very low)</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E2 (low)</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E3 (medium)</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E4 (high)</td>
<td>QM</td>
</tr>
<tr>
<td>S2 Severe and life threatening injuries</td>
<td>E1 (very low)</td>
<td>QM</td>
</tr>
<tr>
<td>(survival probable)</td>
<td>E2 (low)</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E3 (medium)</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E4 (high)</td>
<td>A</td>
</tr>
<tr>
<td>S3 Life threatening injuries, fatal</td>
<td>E1 (very low)</td>
<td>QM</td>
</tr>
<tr>
<td>injuries</td>
<td>E2 (low)</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E3 (medium)</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E4 (high)</td>
<td>B</td>
</tr>
</tbody>
</table>
Quantitative Requirements ISO26262

- ISO 26262
  - Four Automotive SILs (ASIL)
  - Three key metrics
    - Probability of violation of safety goals (PVSG)
    - Single Point Fault Metric
    - Latent Fault Metric
  - Hardware redundancy in structural modeling

<table>
<thead>
<tr>
<th></th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVSG [1/h]</td>
<td>&lt;10^{-7} (recom.)</td>
<td>&lt;10^{-7}</td>
<td>&lt;10^{-8}</td>
</tr>
<tr>
<td>SPFM</td>
<td>&gt;90%</td>
<td>&gt;97%</td>
<td>&gt;99%</td>
</tr>
<tr>
<td>LFM</td>
<td>&gt;60%</td>
<td>&gt;80%</td>
<td>&gt;90%</td>
</tr>
</tbody>
</table>

Key Metrics

- Probability of violation of safety goals
  - Equivalent to PFH in IEC61508
- Single Point Fault Metric
  - Quantifies how many potentially immediately dangerous faults are either safe or detected
- Latent Fault Metric
  - Quantifies how many potentially dangerous faults that not yet influence the application are either safe or detected → under discussion, consult standard!
### 1. Glossary

### 2. Management of functional safety

- **2.4 Management during complete safety lifecycle**
- **2.5 Safety management during development**
- **2.6 Safety management activities after SOP**

### 3. Concept phase

- **3.4 Item definition**
- **3.5 Initiation of safety lifecycle (modification and derivates)**
- **3.6 Hazard analysis and risk assessment**
- **3.7 Functional safety concept**

### 4. Product development system

- **4.4 Initiation of product development system**
- **4.5 HW requirements analysis**
- **4.6 Specification of technical safety concept**
- **4.8 Safety validation**
- **4.9 Functional safety assessment**
- **4.10 Product release**
- **4.7 System design**
- **4.12 Overall quality management**
- **4.13 Qualification of software tools**

### 5. Product development H/W

- **5.4 HW requirements analysis**
- **5.5 HW architecture design**
- **5.6 Quantitative requirements for random HW failures**
- **5.7 Measures for avoidance and control of systematic HW failures**
- **5.8 Safety HW integration and verification**
- **5.9 Qualification of parts and components**
- **5.10 Overall requirements for HW-SW interface**

### 6. Product development S/W

- **6.4 Initiating SW development**
- **6.5 SW safety requirements specification**
- **6.6 SW architecture and design**
- **6.7 SW implementation**
- **6.8 SW unit test**
- **6.9 SW integration and test**
- **6.10 SW safety acceptance test**

### 7. Production and operation

- **7.4 Production**
- **7.5 Operation, service and decommissioning**

### 8. Supporting processes

- **8.4 Interfaces within distributed developments**
- **8.5 Overall management of safety requirements**
- **8.6 Configuration management**
- **8.7 Change management**
- **8.8 Safety analysis**
- **8.9 Analysis of CCF, CMF, cascading failures**
- **8.10 Verification activities**
- **8.11 Documentation**
- **8.12 Overall quality management**
- **8.13 Qualification of software tools**
- **8.14 Qualification of software libraries**
- **8.15 Proven in use argumentation**

### 9. Annexes
# Quantitative Requirements of IEC61508 versus ISO26262

**IEC 61508**
- Four Safety Integrity Levels (SIL)
- Two key metrics
  - Probability of dangerous failure per hour (PFH)
  - Safe Failure Fraction (SFF)
- Hardware redundancy in formulas (HFT)

<table>
<thead>
<tr>
<th>SIL 1</th>
<th>SIL 2</th>
<th>SIL 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFH [1/h]</td>
<td>&lt;10^{-5}</td>
<td>&lt;10^{-6}</td>
</tr>
<tr>
<td>SFF (HFT=0)</td>
<td>&gt;=60%</td>
<td>&gt;=90%</td>
</tr>
<tr>
<td>SFF (HFT=1)</td>
<td>-</td>
<td>&gt;=60%</td>
</tr>
</tbody>
</table>

**ISO 26262**
- Four Automotive SILs (ASIL)
- Three key metrics
  - Probability of violation of safety goals (PVSG)
  - Single Point Fault Metric
  - Latent Fault Metric
- Hardware redundancy in structural modeling

<table>
<thead>
<tr>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVSG [1/h]</td>
<td>&lt;10^{-7} (recom.)</td>
<td>&lt;10^{-7}</td>
</tr>
<tr>
<td>SPFM</td>
<td>&gt;90%</td>
<td>&gt;97%</td>
</tr>
<tr>
<td>LFM</td>
<td>&gt;60%</td>
<td>&gt;80%</td>
</tr>
</tbody>
</table>

Note: Table adopted for typical automotive application
MCU Safety Continuum
Integrated Safety Features

Increasing Market Safety Requirements

- **MPC56xx**
  - Symmetrical Dual Core
  - Core self test

- **MPC551x**
  - Asymmetric Dual Core FlexRay™ MPU

- **MPC555x**
  - MMU EEC

- **S12XD**
  - Asymmetric Dual Core

- **S12XF**
  - FlexRay Networking

- **S12XE**
  - ECC MPU

- **XGATE**

- **e200z0**
  - Crossbar / Bus
  - Periph.
  - Mem.

- **e200z1**
  - Crossbar / Bus
  - Periph.
  - Mem.

- **e200z6**
  - Crossbar / Bus
  - Periph.
  - Mem.

- **eTPU**
  - Crossbar / Bus
  - Periph.
  - Mem.
Integrated Safety Features

Integrated safety
- Fail safe MCUs
- Fail operational MCUs

Safety Properties
- Transient fault detection
- Early detection of permanent faults
- Detection of systematic software faults

System Properties
- Cost benefit
- Low complexity
- High availability

Increasing Market Safety Requirements

MPC555x
MMU EEC

MPC551x
Asymmetric Dual Core FlexRay™ MPU

MPC56xx
Symmetrical Dual Core
Core self test

S12XF
FlexRay Networking

S12XE
ECC MPU

e200z6
Crossbar / Bus

S12X
Asymmetric Dual Core

Periph.
Mem.

eTPU

Periph.
Mem.

e200z1
Crossbar / Bus

e200z0

Periph.
Mem.

Periph.
Mem.

XGATE

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2009.
Processor Core — Performance

► Example: Increased pipeline depth
  • Typically 7-stages+ pipeline architectures allows more instructions per clock cycle
  • Most instructions provide single cycle execution
  • Integer and floating point multiply and multiply-accumulate in three clocks, fully pipelined

► Example: Dual instruction issue
  • Two execution units allow parallel processing of instructions

► Example: Instruction and data cache
  • I-cache to speed up executable instruction fetch
  • D-cache to speed up data fetch and store
  • TLB to improve the speed of virtual address translation

► Example: SIMD unit and FPU
  • Provides DSP capabilities
  • Executes an operation on two separate sets of data

Example: Freescale e200 core family built on Power Architecture® technology
Processor Core — Safety

► Example: Memory management unit (MMU)
- Optimization of self test coverage by using different virtual addresses without relocating customer application data and code
- MMU can be used to protect accesses due to occurrence of faults in the core (exception generation)

► Example: Multiple input shift register (MISR)
- Method for verifying all intermediate results of a set of architectured registers at the end of an instruction stream
- Introduction of MISR improves observability of the core resulting in:
  - Increased self test coverage
  - Faster detection of dormant faults

Example: Freescale e200 core family built on Power Architecture® technology
Memories and Crossbar — Safety

► **Example: Memory protection unit**
  - Monitors all system bus transactions and evaluates the appropriateness of each transfer
  - Pre-programmed region descriptors define memory spaces and associated access rights
  - Unmapped references are terminated with a protection error response

► **Example: Error-correcting code**
  - Used to detect failures of flash/SRAM stored data
  - Typical solution for correcting bitflips caused by soft error rate (SER) impact
  - ECC module (64 data bits + 8 ECC bits) can:
    - Correct all single bit errors
    - Detect all dual bit faults
    - Detect several faults affecting >2 bits

Example: Typical 32-bit MPC55/56xx processor
Communication — Safety

► **Example: FlexRay™ networking**
- FlexRay master controller directly linked to the crossbar
- Replicated transmission of safety relevant data by single/dual channel FlexRay support with 2.5, 5 and 10 MBit/s data rates
- Message buffer stored and protected in dedicated memory partition located in system memory

► **Example: Safety port**
- Controller area network (CAN)-type interface supporting high bandwidth for fast MCU-MCU communication
- Bit rate up to 7.5 Mbit/s
- 32 message buffers of 0 to eight bytes data length

Example: Typical 32-bit MPC55/56xx processor
Power Supply and Clock — Safety

► Example: Power supply
- Monitoring of internal and external voltages — internal and external power supply
- Over- and undervoltage detection
- Testing capability of monitoring circuitry — e.g., for detection of dormant faults

► Example: Clock and monitoring
- Clock monitoring for system and periphery clock:
  - Loss of crystal or PLL clock
  - PLL frequency higher/lower than reference
- Redundant clock generation with internal RC oscillator
- Glitch filtering with on-chip PLL

Example: Typical 32-bit MPC55/56xx processor
Example: Core self test — basic
- Coverage: instruction-set based, all addressing modes
- Integration: mostly interruptible, low integration effort
- Safety: not fault graded, determined behavior in fault-free case
- For PPC instruction set

Example: Core self test — advanced
- Coverage: stuck-at fault model, based on physics of failure
- Integration: partly interruptible, can be adjusted to application/OS specifics
- Safety: detailed test coverage provided, fault graded, determined behavior in fault-free and faulty case
- For selected PPC devices

Example: Typical 32-bit MPC55/56xx processor
Basic Core Self-Test
# CST with Instruction Coverage Metric

<table>
<thead>
<tr>
<th>Instruction coverage*</th>
<th>BookE instructions</th>
<th>VLE instructions</th>
<th>SPE instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>~83% to ~98%</td>
<td>~86% to ~98%</td>
<td>Estimated</td>
<td></td>
</tr>
<tr>
<td>~85% to 99%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code size (bytes)</td>
<td>&lt; 10k</td>
<td>&lt; 5k</td>
<td>In development</td>
</tr>
<tr>
<td>Execution time (clock cycles)</td>
<td>&lt; 6000</td>
<td>&lt; 5000</td>
<td>In development</td>
</tr>
<tr>
<td>Supported PPC Cores</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z6</td>
<td>Supported</td>
<td>Supported</td>
<td>In development</td>
</tr>
<tr>
<td>Z3</td>
<td>Supported</td>
<td>Supported</td>
<td>In development</td>
</tr>
<tr>
<td>Z1</td>
<td>Supported</td>
<td>Supported</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Z0</td>
<td>Not applicable</td>
<td>Supported</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

* Variability caused by whether instructions or operations (performed by instructions) are considered, and whether MMU and cache configuration instructions/operations are taken into account or not.
Basic Operating Principle

► Application
- Triggers test execution
- Selects subset of tests to perform
- Checks actual versus expected result

► Self test API
- Saves application context
- Prepares core and device for testing
- Calls atomic tests
- Checks results
- Restores application context
- Compresses atomic test results into one 32-bit signature

► Atomic test
- Short piece of assembly code
- Optimizes to activate and propagate faults in different core modules
Potential Issues beyond the Self-Test Software

1. Test not triggered
2. Wrong test triggered
3. Runaway
4. Wrong atomic test called
5. Atomic test runaway
6. Test result falsified
7. Check fails
8. Compression fails
9. Error handling fails
10. Result falsified
11. Application check fails

Diagram showing the flow of data and error handling between Application, Self Test API, and Atomic test.
Mitigation Measures

```
<table>
<thead>
<tr>
<th>Can be caught by</th>
<th>Basic Watchdog</th>
<th>Intelligent watchdog</th>
<th>Application check and signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Test not triggered</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Wrong test triggered</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3 Runaway</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4 Wrong atomic test called</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>5 Atomic Test Runaway</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Test result falsified</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7 Check fails</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>8 Compression fails</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>9 Error handling fails</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>10 Result falsified</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>11 Application check fails</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
```

- **Watchdog and redundant result check**
  - External to core
  - May be device internal, however (coprocessor, ETPU, etc.)

- **Application check**
  - Unique result for each atomic test
Overall Operating Principle
Summary
Safety standards are becoming key for the design of new controller solutions and influence the architecture of virtually all building blocks.

Freescale sees safety, and in particular, functional safety as a key paradigm of next generation electronic vehicle systems.

Freescale is continuously expanding the product controller, analog and sensor portfolio to address the needs of these systems in line with IEC61508 and ISO26262.
Thank you for attending this presentation. We’ll now take a few moments for the audience’s questions and then we’ll begin the question and answer session.