Deploying function safety all the way to autonomous driving systems

Odin Shen
Senior FAE

ARM Tech Forum Taipei
July 4th 2017

©ARM 2017
Introduction to functional safety
Functional safety

Systems that must function correctly to avoid unacceptable risk of damage or injury
- Faults must be detected and controlled
- Products must be properly specified and developed accordingly

Safety critical
- Braking, steering, acceleration, chassis control, air bag, seat belt
- Driver relies on these systems to always function correctly
- High safety integrity level (SIL)

Safety ‘nominal’
- Lane departure, speedometer, rear camera…
- So long as the driver is made aware the system is not working
- Medium safety integrity level
Increasing complexity in functional safety markets

Automotive
- Cleaner engines
- Autonomous driving

Industrial
- Factory automation
- Smart robotics

Healthcare
- Robotic surgery
- Advanced medical mobility
Applicable standards

A number of functional safety standards exist
- ISO 26262 – Road vehicles
- IEC 61508 – Electrical, electronic, programmable electronic systems
- DO 254 – Aircraft electronics

Standards always represent an industry consensus
- Long lead-times for standards development (5-10 years)
- Often lagging behind true state-of-the-art

Safety Integrity Levels (low to high)
- SIL 1 to SIL 3
  - Typically SIL 1 or SIL 3
- ASIL A to ASIL D
  - Typically ASIL B (e.g. parking) or ASIL D (e.g. braking)
Automotive safety integrity levels

- Fault metrics
  - Measurement of possible faults that are detectable, and mitigated locally if possible

- Single point fault metric
  - Immediately effective faults

- Latent fault metric
  - Initially silent faults, e.g. in memory bits

- QM: Quality managed

### Safety Levels

<table>
<thead>
<tr>
<th>Safety</th>
<th>SPFM</th>
<th>LFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>QM</td>
<td>Design assurance</td>
<td></td>
</tr>
<tr>
<td>ASIL A</td>
<td>Nominal</td>
<td></td>
</tr>
<tr>
<td>ASIL B</td>
<td>90%</td>
<td>60%</td>
</tr>
<tr>
<td>ASIL C</td>
<td>97%</td>
<td>80%</td>
</tr>
<tr>
<td>ASIL D</td>
<td>99%</td>
<td>90%</td>
</tr>
</tbody>
</table>

*Expect to extend to ASIL B post 2018*
Functional safety controls risks of hazards

Safety application
- Air bag system
- Braking system

Random faults
- Run-time errors
  - Product safety features

Systematic faults
- Design errors
  - Software errors
- Processes
## Types of fault

<table>
<thead>
<tr>
<th>Random faults</th>
<th>Systematic faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Hard errors</td>
<td>- Hardware errata</td>
</tr>
<tr>
<td>- Soft errors</td>
<td>- Software bugs</td>
</tr>
<tr>
<td>- Permanent faults</td>
<td>- Incorrect specification</td>
</tr>
<tr>
<td>- Transient faults</td>
<td>- Incomplete requirements</td>
</tr>
<tr>
<td>- Latent faults</td>
<td>- Unfulfilled assumptions</td>
</tr>
</tbody>
</table>

Managed by including features for fault detection and control

Managed through design process, verification and assessment
Typical fault detection and control features in CPUs

Processor implementation
- ECC or parity on memories
- Soft and hard-error management
- ECC-protected bus ports
- Bus-transaction protection
- Dual core lockstep with delay
- Integrated (lockstep) interrupt controller
- Comprehensive error reporting interface
- Timing protection
- Logic BIST
- (On-line) memory BIST

ARM architecture
- Memory protection
- Interrupt and call handling
- Virtualization for software separation
- System-error exception
Beneath the surface – systematic fault avoidance

- Safety management
- Training
- Requirements management
- Specific design tool flows
- Verification
- Documentation
- Assessment
- Quality assurance
- Errata management
- Support and maintenance
- Safety engineering interface
- …
Security features in ARM IP
ARM TrustZone® for ARMv8-M

- Optimised for small real-time processors
- Low, deterministic interrupt latency
- Efficient – every cycle counts
- No hypervisor code and processing overhead

Hardware based security state switch

- Full programmable in C
- Easy to program, easy to debug

Transparent to the software developer
- Transition via a standard function call

Easy to program, easy to debug
ARMv8-M interrupt security

High-performance interrupt handling with register protection

- Subject to priority, Secure can interrupt Non-secure and vice versa
  - Secure can boost priority of own interrupts
  - Uses current stack pointer to preserve context.

- Uses ARMv7-M exception stacking mechanism
  - Hardware pushes selected registers.

- Non-secure interruption of Secure code
  - CPU pushes all registers and zeroes them
    - Removes ability for Non-secure to snoop Secure register values.
Security defined by address

All transactions from core and debugger checked

- All addresses are either Secure or Non-secure.

- Policing managed by Secure Attribution Unit (SAU)
  - Internal SAU similar to MPU
  - Supports use of external system-level definition
    - E.g. based on flash blocks or per peripheral.

- Banked MPU configuration
  - Independent memory protection per security state.

- Load/stores acquire NS attribute based on address
  - Non-secure access attempts to Secure address = memory fault.
CoreLink SIE-200: System IP for embedded

- Simplify the design of a secure system
  - Designed and verified with latest ARMv8-M CPUs
- Reduce design time with IP reuse
  - Re-use and secure existing IP in AHB5 systems
- Extend security to peripherals
  - Integrate legacy peripherals
  - Programmable at run-time
- Protect code and data
  - Programmable regions for multiple applications
The V-model
The V-model of the Systems Engineering Process
The V-model of the Systems Engineering Process

- An extension of the Waterfall model
- Describes the relationship between development phases and testing phases
- A cornerstone of Functional Safety certification

<table>
<thead>
<tr>
<th>Verification</th>
<th>Validation</th>
<th>Testing by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements analysis</td>
<td>User acceptance testing</td>
<td>Client</td>
</tr>
<tr>
<td>System design</td>
<td>System testing</td>
<td>Client</td>
</tr>
<tr>
<td>Architecture design</td>
<td>Integration testing</td>
<td>Developer</td>
</tr>
<tr>
<td>Module design</td>
<td>Unit testing</td>
<td>Developer</td>
</tr>
</tbody>
</table>
Safety features in ARM tools
Meet ARM Compiler 6
Taking embedded to the next level

- **Always the first** to support new ARM architectures/cores
  - Used internally for architecture validation

- **Optimized** for a wide range of workloads
  - Not just a single benchmark

- Builds on **mature** ARM Compiler technology
  - Proprietary C library tuned for embedded

- **Fully integrated** into DS-5 and Keil MDK
## ARM Certified Compilers

<table>
<thead>
<tr>
<th></th>
<th>AC 5.06</th>
<th>AC6.6 (expected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certification</td>
<td>ISO 26262 ASIL D</td>
<td>ISO 26262 EN 50128</td>
</tr>
<tr>
<td></td>
<td>IEC 61508 SIL 3</td>
<td>IEC 61508 IEC 62304*</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Any safety integrity level</em></td>
</tr>
<tr>
<td>Core support</td>
<td>Up to ARMv7-M/R/A</td>
<td>All current ARM Cortex cores</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Including ARMv8-M/R/A</td>
</tr>
<tr>
<td>Functionality</td>
<td></td>
<td>New features such as LTO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GCC compatibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expressive diagnostics</td>
</tr>
<tr>
<td>Availability</td>
<td>DS-5 Ultimate, MDK Pro</td>
<td>DS-5 Ultimate, MDK Pro</td>
</tr>
<tr>
<td></td>
<td>Term only</td>
<td>Standalone (term/perpetual)</td>
</tr>
</tbody>
</table>

*Certified as suitable for use
ARM Compiler Qualification Kit

Applicable to additional safety standards

- Augments the TÜV certificate
  - Safety Manual referenced in TÜV report
  - New defects reported within the Defect Report

- Enables in-house qualification
  - QK provides relevant evidence and usage guidelines

- Audited & validated toolchain development and test processes
ARM Compiler Qualification Kit

- Safety Manual
  - Usage information relevant to safety critical software developers
    - Definition of the safety function
    - Behaviour under erroneous conditions
    - Error detection facilities
- Defect Report
  - Listing of all known defects that cause run-time failures and any known workarounds
- Perennial Test Report
  - Tests conformance to the ISO C90 and C99 C language standards and the ANSI/ISO C++ standard
- Software Development Process Document
  - Description of the ISO 9001:2008 compliant software development process
TÜV SÜD Certification for Functional Safety

- Certification expected:
  - ISO 26262, IEC 61508, EN 50128, IEC 62304
  - No further build tools qualification effort required
  - Any safety integrity level, provided Qualification Kit recommendations are followed

- No restrictions on compiler optimization level all the way to -O3

- Relevant to any safety-related software development on ARM
  - Includes applications in Automotive, Industrial, Medical, Railway and Avionics
  - The accompanying ARM Compiler Qualification Kit is applicable to other safety standards
ARM Compiler Long Term Maintenance (LTM)

- Changing compiler can be very invasive
  - Toolchain has to be re-validated, all test results are voided
  - However it is essential that critical defects are addressed!

- ARM Compiler 6 for Functional Safety is a stable branch
  - Regularly scheduled maintenance releases for critical defects
  - No new cores, no new features, no new optimizations….
  - Maintained for 5 years, with longer term contracts available by special arrangement
Static code analysis

- DS-5 uses Eclipse, can be integrated with a wide variety of ecosystem plugins
- LDRAlite™ is an Eclipse plugin, already integrated into DS-5
  - Comprehensive static analysis with impressive ease of use
  - View violation, rule number and description
  - View or mark violation in source file
MDK trace-based code coverage

- Analyses code analysis on at the instruction level
- Information taken from simulator or from hardware (ETM trace)
MDK-ARM link to MATLAB® and Simulink®

- Provides optimized code generation for Cortex-M based microcontrollers

- Uses CMSIS-DSP library
  - Efficient execution of mathematical algorithms

- Also supports processor-in-the-loop (PIL)simulation using ULINK
  - Code verification on actual target hardware
In conclusion
ARM Commitment to Functional Safety

IP
- Safety Manual
- Qualification Results
- Development Interface Agreement

Development tools
- Qualification Kit and TÜV Certification
- Extended Maintenance
- Trace-based Code Coverage
- Execution Profiler

Ecosystem
- MISRA checking
- Device Self-test
- Policy-driven Development

Solutions for the complete development process
The trademarks featured in this presentation are registered and/or unregistered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

Copyright © 2017 ARM Limited

©ARM 2017
Real-time virtualisation for safety and security

- ARMv8-R architecture
- New Privilege Level (EL-2) within Cortex-R processors
- Bare metal Hypervisor support
- Safety and security sandbox
- Software separation for ‘golden’ code
- Task consolidation onto fewer platforms
- Accelerated interrupt response and context switch
- Enhanced AArch32 instruction set (A32 and T32)