Safety Features & Standards for Cortex™-R Processors in Embedded Systems
ARM® Cortex-R Family

**Cortex-R4 (2005)**
- High performance, real time embedded processor
- Deterministic event response
- Configurable feature set
- Safety critical system support

**Cortex-R5 (2010)**
- New features enhance system performance
- Low Latency Peripheral Port
- Accelerator Coherency Port
- Dual core configuration
- Extended error management
- Smaller Floating Point Unit

**Cortex-R7 (2012)**
- Large performance increase
- Advanced microarchitecture
- Higher clock frequency
- Symmetric Multiprocessing
- Dual core and ACP coherent
- Quality of Service features
- Extended real time memory
- Hard error management
- Integrated interrupt controller

**Applications**
- Automotive ECU
- Mobile baseband
- High capacity storage
- Industrial control
Cortex-R Processor Characteristics

- Cortex-R4, Cortex-R5 and Cortex-R7 are designed for high-performance, hard real-time, dependable system processing applications.

**Performance**
- High frequency pipelines
- Instruction pre-fetch
- Branch prediction
- MPCore™ interrupt controller
- Superscalar execution
- DSP / SIMD instructions
- Floating point instructions
- Double precision
- Harvard level 1 caches
- AMBA® AXI3™ bus ports
- Data I/O coherency

**Safety**
- Cache ECC and parity
- TCM ECC and parity
- AXI port ECC and parity
- Soft & hard error management
- Memory Reconstruction Port
- Memory Protection Unit
- Privileged/User execution
- Exception handling modes
- Lock-step configuration
- Safety documentation
- Bounded response time

**Determinism**
- Low Interrupt Latency mode
- Tightly Coupled Memories
- Low Latency Peripheral Port
- Both instruction and data trace

**Cost & Power Efficient**
- Low energy microarchitecture
- ARM or Thumb®-2 code size
- Synthesis feature configuration
- Target from low power/small die area to high performance
ARM in Automotive

Dash

Body

Power train

ADAS

Body

Electric / hybrid energy system

Chassis

IVI

ASIL

IEC61508

ISO26262

Powertrain

OLEA

Scaleo chip

Chassis

ARM

The Architecture for the Digital World®
Functional Safety for Automotive

- Automotive microcontroller users require information and data from semiconductor companies to verify the safe functioning of products
  - The automotive standard for functional safety is ISO 26262 (2011)
  - Automotive Safety Integrity Levels are classified as A, B, C or D
  - Processor cores are treated as a ‘Safety Element Out Of Context’
  - Documentation and data is provided for each SEOOC component

DIA
Development Interface Agreement

Assumptions of Use

Safety Case
Verification plan and results

Safety Plan

Safety Manual
Safety features description

FMEDA
Failure Mode Effect and Diagnostic Analysis
## Failure Mode Effect and Diagnostic Analysis

- Quantitative analysis of a processor’s RTL design
  - Using both the Verilog source code and the implemented net list gate counts
  - It is firstly qualitative and then the analysis becomes quantitative
  - Analysis percentages require adjustment for processor synthesis configuration

<table>
<thead>
<tr>
<th>Component</th>
<th>Failure Mode</th>
<th>Effect</th>
<th>Diagnostic</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub part of the IP core</td>
<td>Description of various permanent and transient fault mechanisms that can be foreseen</td>
<td>Effect that would be observed at the IP core boundary</td>
<td>Capability to detect, diagnose and/or record and/or recover from the fault</td>
<td>Percentage of IP core logic occupied by the sub part. Percentage of faults therein which cause errors vs. those 'safe faults' that do not.</td>
</tr>
<tr>
<td>e.g. The ALU, or the multiplier, or the AXI-bus interface module</td>
<td>Faults can be due to hard or soft errors, e.g. a stuck-at net due to metal migration or a blown transistor (permanent), or a flop knocked over by an alpha particle (transient).</td>
<td>Many simply stop operation. Some cause data errors. Some create an exception or other mechanism that can be trapped or corrected. So not all faults result in errors.</td>
<td>Many cannot be detected or diagnosed. Some can be trapped or will be corrected, e.g. ECC or parity errors.</td>
<td>Totals up to 100%. Safe fault example - a fault in the debug logic would not cause an error during normal operation. Done twice – for both permanent and transient fault analysis.</td>
</tr>
</tbody>
</table>

Table continues for many pages
Calculating Fault Metrics

- FMEDA is used to calculate fault metrics for a microcontroller
  - Failure rate data from a SiP’s own QA, IEC 62380, JEDEC, ITRS & elsewhere
  - FMEDA is determined based upon some Assumptions Of Use

- FIT rate of permanent faults in MCU (aging etc.)
- FIT rate of transient faults in MCU (alpha particles etc.)
- FMEDA of whole MCU
- FMEDA of ARM CPU
- Transient fault metric
- Permanent fault metric
- Safety Integrity Level requirement
Safety Element Out Of Context

- Making the standard work throughout the supply chain
  - Good fit to IP business model
- ISO 26262 is primarily concerned with safety features
  - Normal functionality is specified and verified as before

Ecosystem

Tier -1

SiP

ARM

OEM

Safety documentation and data at each step
Safety Features in Systems

- Safety features deal with faults occurring during normal operation
  - Exception handling and memory protection – designed to address systematic faults
  - ECC/Parity detection/correction on level-1 memories and bus ports – random faults

- Safety features outside the processor designed to deal with faults
  - Lock-step (redundant CPU) configuration – random faults in CPU flops and gates
  - BIST – random faults in hardware (probably) caused by permanent failure
  - CRC – checking program memory for corruption due to random faults
  - Watchdog – detects both systematic and random failures causing program flow errors

- The following are not safety features but contribute to high availability
  - TCM – deterministic memory for code and data, e.g. critical interrupt service routines
  - LLPP – peripheral I/O access irrespective of activity (or lack of) on the main AXI bus

- Features not used in regular operation are not defined as safety features
  - e.g. CoreSight™ debug and trace, Performance Monitoring Unit
  - Although they may be useful for safety, e.g. PMU logs ECC correction events
Dual Core Lock Step

- Lock-step CPUs
  - Redundant copy of the CPU with checking to detect faults in flops and gates

- Both spatial (also orientation) and temporal separation (1 ½ or 2 cycles)
  - Avoiding common cause failures, i.e. reduced probability of both CPUs seeing the same failure at the same time and still checking OK

- ARM IP provides example out-of-box configuration
  - SiPs develop their own versions

- Split-lock
  - Cortex-R5 can be powered up in either lock-step or split mode for 2x performance

- Safety consideration
  - A matter for the MCU design as this is outside the individual processor
Privileged Modes – Memory Protection

- Exception handling and memory protection in the ARM architecture
  - Events (interrupts and software traps) and exceptions (undefined instruction or abort, e.g. illegal memory access attempts) immediately enter a dedicated privileged mode
  - Each mode has a dedicated register set, program status register and stack pointer
  - The operating system can load, start, restart or kill tasks as required
  - Application tasks are isolated from each other and cannot reconfigure the MPU
  - The MPU hardware gates all memory accesses to pre-configured regions

![Diagram showing privileged modes and memory protection]

- User PL0
- Supervisor PL1
- FIQ PL1
- IRQ PL1
- Undef PL1
- Abort PL1

- Precise exception handling
- Instruction memory
- Data memory
- Devices

- Instruction fetch
- Data read/write
- Device read/write
Memory Protection with Parity and ECC

- **Parity** – costs one bit per Byte
  - One error per byte detected (odd or even parity can be used)

- **64-bit ECC** – costs 8 bits per DWord
  - One error per DWord corrected. Two errors detected

- **32-bit ECC** – costs 7 bits per Word
  - One error per Word corrected. Two errors detected

- ARM processor’s buffers and data paths are 64-bits wide, i.e. DWord
ECC/Parity Synthesis Options

- ARM Partner selects features/options when synthesising their MCU design

<table>
<thead>
<tr>
<th>Feature</th>
<th>Data error handling options</th>
<th>Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>I cache</td>
<td>Parity or 64-bit ECC</td>
<td>Cortex-R4 Cortex-R5</td>
</tr>
<tr>
<td>D cache</td>
<td>Parity or 32-bit ECC</td>
<td>Cortex-R4 Cortex-R5</td>
</tr>
<tr>
<td>A TCM</td>
<td>Parity or 32 or 64-bit ECC</td>
<td>Cortex-R4</td>
</tr>
<tr>
<td></td>
<td>32 or 64-bit ECC</td>
<td>Cortex-R5</td>
</tr>
<tr>
<td>B0, B1 TCM</td>
<td>Parity or 32 or 64-bit ECC</td>
<td>Cortex-R4</td>
</tr>
<tr>
<td></td>
<td>32 or 64-bit ECC</td>
<td>Cortex-R5</td>
</tr>
<tr>
<td>AXI-M AXI-S ACP</td>
<td>ECC/Parity generate &amp; check</td>
<td>Cortex-R5</td>
</tr>
<tr>
<td>AXI-PP AHB-PP</td>
<td>ECC/Parity generate &amp; check</td>
<td>Cortex-R5</td>
</tr>
</tbody>
</table>

- On detecting an error:
  - TCM scheme writes corrected data back into the TCM. Instruction is re-executed and data is re-fetched from TCM
  - Cache ECC scheme automatically invalidates incorrect data. If the cache line is dirty then ECC logic corrects the evicted data. Instruction is re-executed and the cache line is re-fetched from L2. L2 assumed correct.

- Read-Modify-Write optimization:
  - Cache RMW is performed in the store buffer, which can merge writes such that RMW to RAM is not always needed
  - TCM RMW is done within spare cycles in the pipeline

- Level 1 memories and AXI ports address and control lines are also protected
ECC Mechanism

- **Error Correcting Code**
  - Single Error Correct – Double Error Detect
  - 64-bit scheme is most efficient for I-side
  - 32-bit scheme is best for D-side to minimize Read-Modify-Write cycles

- RMWs required to re-calculate ECC when writing a quantity smaller than memory chunk
- RMWs performed automatically with minimal, or even zero, performance impact
- Both I fetch and D load (i.e. all reads) go AFAP by assuming no ECC errors – if there is an error the CPU pipeline replays the instruction and re-fetches the corrected data
Hard Error Treatment

- Hard errors cannot be corrected by writing back corrected data
  - And will repeat when memory is read again
  - ‘Live-lock’ scenario when uncorrected instructions or data are continuously re-fetched

- Cache location avoidance
  - Corrected data is evicted if dirty
  - Cache line invalidated
  - Forces data to be re-read
  - A different cache line is used

- TCM hard error
  - Corrected data is written back to TCM, fixing soft (transient) errors for future use
  - Corrected data also written to single entry memory
  - Re-read data is always taken from this memory to avoid the same hard error
Tightly Coupled Memory

- TCM is an alternative L1 memory system for deterministic performance
  - Flexible configurations for holding fast responding code and/or data
  - Each interface has an independently programmed base address

- Three unified TCM ports
  - Flexible mix of instructions, data and memory type (RAM, ROM, Flash etc.)

- Added data port feature
  - AXI slave port for direct TCM access

- Typically used for event handlers
  - or for fast access to data tables e.g. Interrupt Service Routines and vector tables
  - Saves many 100s of cycles interrupt response if ISR not in cache
  - Bounding response time and saving energy
  - Cortex-R4 or Cortex-R5 using a Vectored Interrupt Controller can respond to interrupts in 30-cycles or less
Bounded Real-Time Response

- Superscalar execution throughput
  - Cache line buffers minimise stalling while waiting for the L2 memory system

- TCM and LLPP interfaces
  - Direct paths to LSU and store queue avoid delay due to dynamic behaviour of cache system and AXI-Main bus

- Low Interrupt Latency mode
  - Fast interrupt response by abandoning any pending restartable memory operations
  - Restartable operations are multiword transfer instructions LDM, LDRD, STRD, STM, PUSH and POP accessing normal memory
  - Restarting instructions causes memory accesses to be repeated
# Safety Features in ARM Processors

- Examples of safety features in current ARM embedded processors

<table>
<thead>
<tr>
<th>Safety features</th>
<th>Cortex-M</th>
<th>Cortex-R4</th>
<th>Cortex-R5</th>
<th>Cortex-R7</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task/OS privilege isolation</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Fast exception handling</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Memory Protection Unit</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Fault observation interface (fRCPU)</td>
<td>✔️*</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ECC on L1 cache and TCM</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>ECC on AMBA AXI3 bus ports</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Error/performance monitoring</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Lock-step configuration</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Hypervisor safe privilege level</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

* Cortex-M3 only

- Microcontroller vendors design more safety features in their products
  - Integrating and complementing the ARM processor safety features
Conclusion

- Safety features
  - 16 region memory protection unit
  - ECC and Parity protection on L1 memories
  - ECC and Parity on AXI3 bus port interfaces
  - Dual core option split for 2x performance or lock-step for safety-critical applications
- Deterministic response to real-time events
- Flexible system integration options
- Synthesis configured for low power/low cost

These safety features combined with the performance and deterministic behaviour of the ARM Cortex-R processors make them a popular choice for microcontrollers or ASICs to be used in integrated control and safety systems.
Thank you