Programmer Issues Moving from TC1797/67/36 to TC1798/93/91

G.Farrall ATV Microcontrollers (v1.1)

2010.04.07

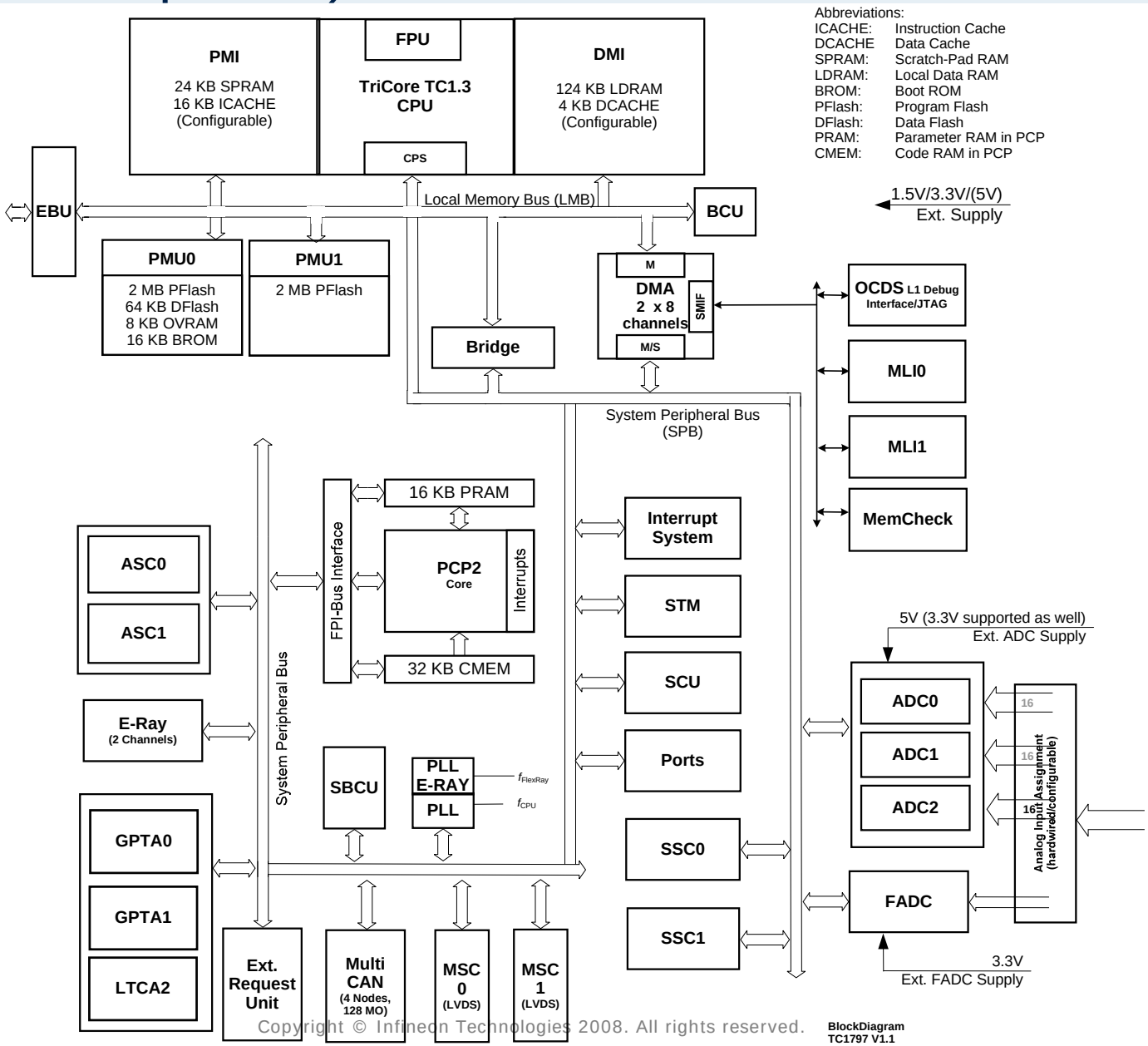
Never stop thinking

Topics

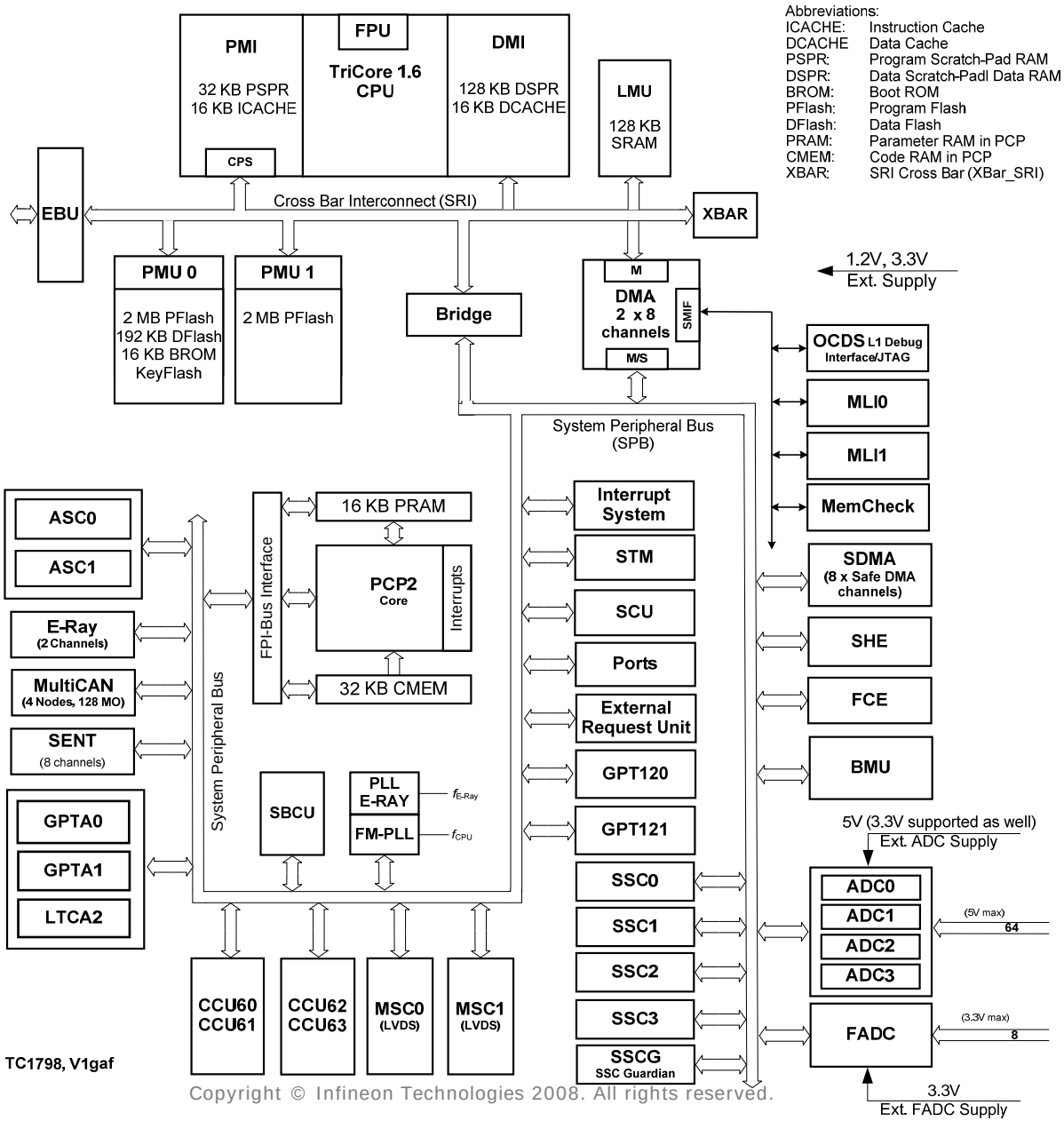
- Products Overview
- Address Map Issues
- Compiler/Assembly Differences
- Processor Features

- Collateral Material (not planned to be covered)

Present: TC1797 Diagram (& 67/36 Superset)



Future: TC1798 Diagram (& 93/91 Superset)



High Level Differences from Diagram

- Higher Performance Systems with max 300MHz vs max 180MHz
- Many Extra Peripherals (CCU6[], GPT[], SSC[2,3] etc.)
- Extra SRAM in System located in LMU (local memory unit)
- High Performance Interconnect changed from LMB (true bus) to SRI (cross bar)
- TriCore CPU is a TC1.6 core compared to earlier TC1.3 core.
- Minor Point is a Terminology Change
  - SPRAM (scratch pad ram) now called PSPR (programme scratch pad ram)
  - LDRAM (local data ram) now called DSPR (data scratch pad ram)

Myriad Low Level Differences

- Overlay RAM has gone from PMU0 to be supported in LMU
- SRAMs now all ECC protected (vs parity)
- PCP2 with additional features
- ADC with four ADC kernel and additional features
- Ports with individual driver strength selection per port
- Main PLL has Frequency Modulation for EMI reduction
- Etc.!

Main Topics covered here

- For moving a system from TC1797 to TC1798 only the same peripherals can be used, so explicitly not covering new peripherals.
- In addition will not go through each additional features added to each peripheral as again, clearly a compatibility issue if a new feature is used
  - hopefully insulated by device drivers (e.g. AutoSAR/Dave etc.)
- Most system level features have defaults (or modes) to allow similar treatment as with TC1797 – e.g. ECC can be turned off – all single bit faults will cause an NMI rather than have the memory value corrected.
- Processor issues, esp. code portability will be the main focus for the discussion

Topics

- Product overview
- **Address Map Issues**
- Compiler/Assembly Differences
- System Features

Address Map Issues -i

- Many minor address map changes – these should all be dealt with for compiled and 'well behaved' assembler code by the tool chain (linker especially)
  - E.g. base address of SSC0 and SSC1 have moved, as long as absolute addresses are not hard coded into assembler accesses to these peripherals should not need to be modified for this purpose
- Some Address Map changes to consider
  - PMU0 and PMU1 previously had contiguous first 4MBs of address in segments 8/A, i.e. $8000\ 0000_H$ - $803F\ FFFF_H$
  - Now PMU0 and PMU1 have discontiguous addresses i.e. $8000\ 0000_H$ - $801F\ FFFF_H$ & $8080\ 0000_H$ - $809F\ FFFF_H$
    - Only an issue for large data structures (e.g. array) which might be straddling the top of 2nd-MB to start of 3rd-MB, their location will need to be adjusted.

Address Map Issues -ii

- Previous generations of TriCore products (e.g. TC1796) have had PSRAM beginning at address D400 0000_H (through D400 $BFFF_H$)
- For TC1797-family this address region was retained for compatibility as an alias to the address region C000 0000_H,
  - so for example, high performance code located in the PSRAM could be addressed with either C000 0000_H or D400 0000_H.
- For TC1798-family this address alias is no longer supported. The PSPR is only addressable from the C000 0000_H address.
  - The linker should flag any errors with the address map if trying to locate code into this address region, but any hardcoded addresses may **NOT** be detected.

Address Map Issues -iii

- A functional change to the processor also impacts the address map
  - In TC1797-family the amount of cache could be programmed at boot time. Any SRAM in the PMI or DMI which was not allocated to the ICache or DCache became usable and addressable contiguously to the SRAM in the PMI or DMI
  - For the TC1798-family, the caches cannot be configured in size, but only enabled or disabled completely. If disabled the SRAM made available is addressable in the PMI or DMI but is NOT contiguous with the scratch SRAM.
    - E.g. TC1797 24K/16K SPRAM/ICache can be addressed as 40K scratch from C000 0000_H to C000 $9FFF_H$
    - TC1798 32K/16K SPR/ICache can be addressed as 48K scratch at C000 0000_H to C000 $7FFF_H$ and C020 0000_H - C020 $3FFF_H$
      - only an issue if all SRAM allocated to scratch, and a data structure allocated across boundary of C000 $7FFF_H$ to C000 8000_H

Address Map Issues -iv

- No dedicated OVRAM – this functionality now covered by using SRAM in the LMU.
  - Should only be an issue for calibration tool vendors, but any customer accesses to the addresses 8/AFE8 0000_H will fail in the TC1798 family.

Topics

- Product overview
- Address Map Issues
- **Compiler / Assembly Differences**
- System Features

Compatability

- TC1.6 is instruction set compatible with the TC1.3, i.e. upwardly compatible.
  - TC1.3 user binary will run on a TC1.6
  - but TC1.6 binary may not run on a TC1.3 if it uses new instructions
- Compiler and tool chain typically insulates C-code from any processor differences when targeting each platform.

Topics

- Product overview
- Address Map Issues
- Compiler/Assembly Differences
  - **Compatibility Aspects**
  - Performance Aspects
- System Features

Summary of TC1.6 Instructions not compatible with TC1.3

- Integer Division Instructions added
  - DIV & DIV.U
- Fast Call and Return Instructions added
- Extended Address offsets for Load and Store of Byte/Halfword data types
- Extended Branch offsets for commonly used short format conditional instructions
- Initialisation and movement of 64-bit register pairs added
- System instructions for simpler manipulation of interrupt enable bit
- System housekeeping instruction added for cache invalidation.

TC1.6 New User Instructions (i)

- Integer Division: performance of integer division on the TC1.3 processor series had been identified as an area of improvement

```
dvinit      e8,d4,d0   ; 2 cycles
dvstep      e8,e8,d0   ; 4 cycles
dvstep      e8,e8,d0   ; 4 cycles
dvstep      e8,e8,d0   ; 4 cycles
dvstep      e8,e8,d0   ; 4 cycles
dvadj       e8,e8,d0   ; 2 cycles
```

(Approx 20-cycles for a 32/32 divide)

- TC1.6 introduced integer divide instructions
  - DIV & DIV.U
- Takes same input as DVINIT/DVSTEP/DVADJ
  - Returns same result as DVINIT/DVSTEP/DVADJ
  - Result in 11 Cycles (Less with early termination)
- Shares floating point divide logic
  - SRT algorithm (Sweeny, Robertson, Tocher)
- Improves Benchmark and Application performance
  - e.g. Dhrystone improves ~5%

TC1.6 New User Instructions (ii)

- New Fast Context switch and return (FCALL, FRET)
  - Very light weight context switch using Stack
  - FCALL, FCALLA, FCALLI (store current PC to stack and jump)

    EA = A[10]-4
    Store A[11] at EA
    Store PC+4 at A[11]
    Move EA to A[10]
    Jump to target defined by instruction

  - FRET (Jump to A[11] and load previous A[11] from stack)

    Jump to A[11]
    EA = A[10]
    Load A[11] from EA
    Move EA+4 to A[10]

TC1.6 New User Instructions (iii)

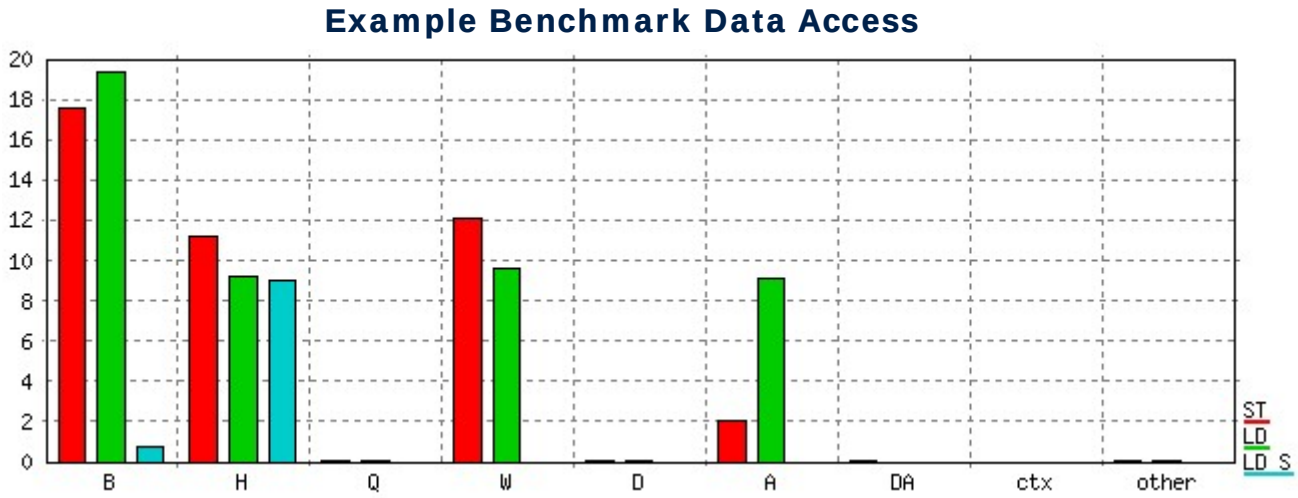
- New FCALL/FRET supports code compaction using reverse inlining techniques.
  - Identify and replace common code sequences with a subroutine
  - Use FCALL and FRET to enter and exit subroutine.
- Potential Reduction in CSA area requirements.
  - Use FCALL instead of CALL
  - Save/restore only required registers during subroutine operation.
- To enhance determinism, all CALL/RET instructions now take a fixed number of cycles
  - CSA background registers which enhanced TC1.3 **average** cycles are removed
  - 4 cycles CALL, 4 cycles RET from Scratch Memory
  - 8 cycles CALL, 8 cycles RET from Cache

TC1.6 Enhanced User Instructions (i)

- The TriCore 1.3 architecture currently defines two Base+Offset Addressing modes
  - Base+Offset (BO) mode adds 10-bits of signed offset to a base address.
  - Base+LongOffset (BOL) mode adds 16-bits of signed offset to a base address.
- The BO mode is supported by **all** Load and Store operations.
  - The BOL mode is supported by a small subset of Load and Store operation (Primarily word operations).
    - LD.A, LD.W, ST.W, LEA

TC1.6 Enhanced User Instructions (ii)

- TC1.6 Architecture extends BOL addressing to
  - LD.BU, LD.B, LD.HU, LD.H
  - ST.B, ST.H, ST.A
- Customer Benchmarks show
  - ~2% performance increase, ~4% reduction in code size.



TC1.6 Enhanced User Instructions (iii)

- **Range extension for 16 bit conditional jump instructions (JEQ, JNE)**
  - The jump range of these instructions is extended by +16.
- **Double word move (MOV)**
  - The data MOV instruction is enhanced to allow the movement of 64-bit values.

TC1.6 New System Instructions

- **Disable with Register (DISABLE)**
  - Disable interrupts and store the current ICR.IE bit in a register.
  - This instruction in conjunction withe the RESTORE instruction below allows interrupts to be quickly disabled and restored to their original state without the use of MFCR/EXTR - INSERT/MTCR type sequences.
- **Restore (RESTORE)**
  - Take register bit and move to ICR.IE
- **Data Cache Invalidate on index (CACHEI.I)**
  - Invalidate data cache line at index
  - Allows data cache line invalidation independent of dirty bit state. This provides a mechanism to quickly terminate a cached task.

Topics

- Product overview
- Address Map Issues
- Compiler/Assembly Differences
  - Compatibility Aspects
  - **Performance Aspects**
- System Features

Topics

- Performance Aspects
  - **Pipeline Effects**
  - Floating Point Unit

Pipeline Effects

- Longer pipeline of the TC1.6 results in difference performance of load instructions
  - TC1.3 had no load-use delay
  - TC1.6 has one cycle of load-use delay
- Compiler is aware of this delay and schedules instructions such that no performance effect on typical code

- DSP code (or other assembler) code which has been coded taking advantage of the no load-use delay will operate correctly but suffer a slow down as the pipeline stalls to allow for the load-use penalty
  - Easily resolved by rewriting (one level of software pipelining) the inner loop of the DSP/assembler code.

Topics

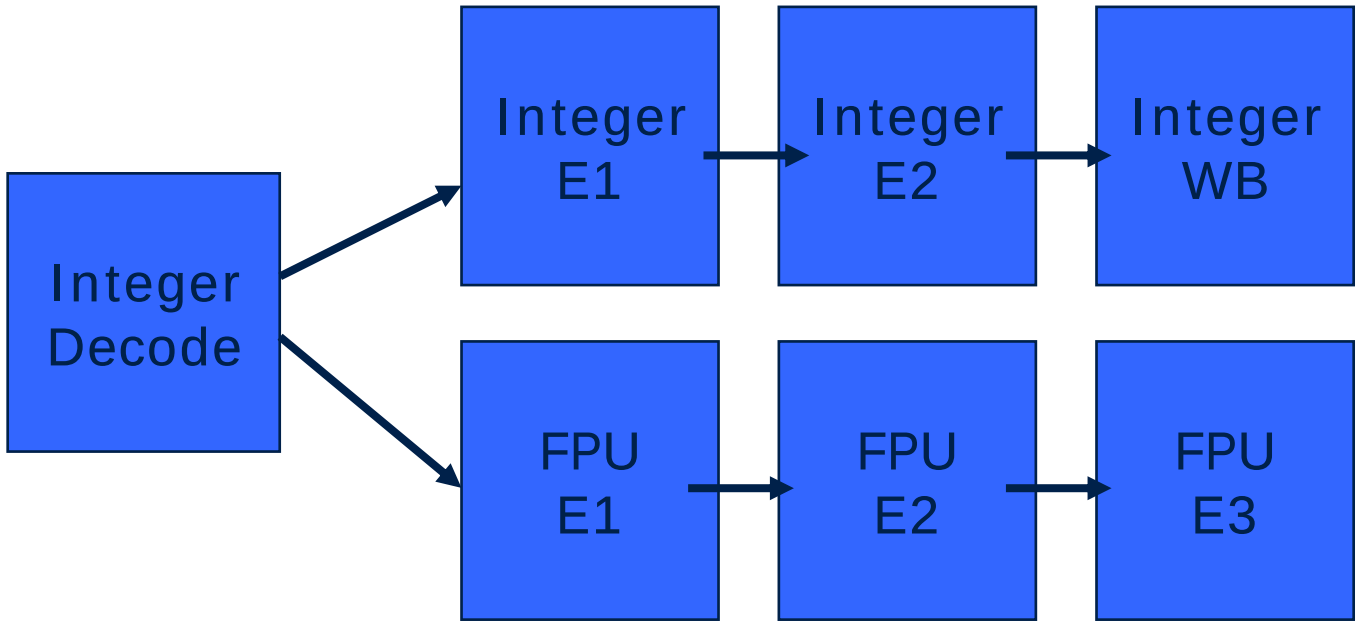
- Performance Aspects
  - Pipeline Effects
  - **Floating Point Unit**

Fully Pipelined Floating Point Unit (i)

- TC1.3 family use a separate floating point unit attached to the external coprocessor interface with a Non Pipelined FPU Flow
  - CPU passes Instruction and operands to FPU then waits for result.
  - FPU calculates result of the requested operation and passes it back to the CPU
  - CPU receives result of requested operation and resumes operation.

Fully Pipelined Floating Point Unit (ii)

- TC1.6 implements a fully pipelined FPU
  - Functionally identical to TC1.3.1 FPU unit
  - Pipelined to work in parallel with the existing integer pipeline.
  - Floating point instructions are identified in the decode pipeline stage and diverted.



Fully Pipelined Floating Point Unit (iii)

Instruction	Result Latency	Repeat Rate	Instruction	Result Latency	Repeat Rate
Floating Point Instructions					
ADDF	2	1	ITOF	2	1
CMP.F	1	1	MADD.F	3	1
DIV.F	8	6	MSUB.F	3	1
FTOI	2	1	MUL.F	2	1
FTOIZ	2	1	Q31TOF	2	1
FTOQ31	2	1	QSEED.F	1	1
FTOQ31Z	2	1	SUB.F	2	1
FTOU	2	1	UPDFL	–	1
FTOUZ	2	1	UTOF	2	1

- Another major difference (apart from the pipelining) is the improvement of Divide performance from 15 cycles non-pipelined latency to 8 cycles latency with a repeat rate of 6 cycles.
- For non-dependent divides this means throughput has more than doubled (250% improvement);
  - 2 divides in 12 cycles vs 1 in 15 cycles.

Topics

- Product exemplar diagrams
- Address Map Issues
- Compiler/Assembly Differences
- **System Features**

System Features that are transparent

- Separate Debug and Memory Protection system.
  - TC1.3 processors used same resources for protected memory systems and for attached debuggers.
  - TC1.6 uses separate resources to allow richer debugging of protected memory systems.
- No SPB interface (CPS register access via PMI SRI slave)
  - Interference characteristics of debugging are slightly changed
    - Only affects accesses to CPU (and DMI/PMI) registers
  - similar characteristics for accesses to memory
    - Potentially less interference as Cross Bar allows complete concurrency of accesses when no conflict for resource occurs

CPU Resources in DMI and PMI

- One change the programmer needs to be aware of is the repositioning of control resources in the PMI and DMI from
  - SFRs: Special Function Registers accessed via loads and store instructions by the CPU
  - to CSFRs:Core Special Function Registers which are accessed via MTCR and MFCR instructions
- Typically access to these resources is only required by startup code to enable/disable and configure caches.
  - Highly localised routines which will need to be conditionally selected for device variants if a common program is required.

Topics

- Product exemplar diagrams
- Address Map Issues
- Compiler/Assembly Differences
- **System Features**
  - **TC1.6 memory protection**
  - TC1.6 temporal protection

TC1.6 Extended Memory Protection Backward Compatibility

- Extended memory protection is a superset of TC1.3 memory protection, i.e. for each TC1.3 configuration, a corresponding TC1.6 configuration can be enabled

- Compatibility mode
  - Ranges associated with sets in the old way (fixed mapping)
  - New control bits (Range Select) not accessible
  - Register-level compatible with TC1.3.1
  - The only apparent difference: granularity at 8-bytes vs. 2-byte fetch, 1-byte data
    - Reality is achieved protection is very similar due to start address only being checked for permission
  - Compatible configuration is the default mode after CPU reset

Topics

- Product exemplar diagrams
- Address Map Issues
- Compiler/Assembly Differences
- **System Features**
  - TC1.6 memory protection
  - **TC1.6 temporal protection**

TC1.6 Temporal Protection System

- Mechanism added to prevent a task overrunning its assigned runtime
  - Especially in the presence of RTOS/systems where tasks are allowed to disable interrupts.
  - Mechanism adds a trap which cannot be disabled, and new resources for counting cycles for the task.

ENERGY EFFICIENCY
COMMUNICATIONS
SECURITY

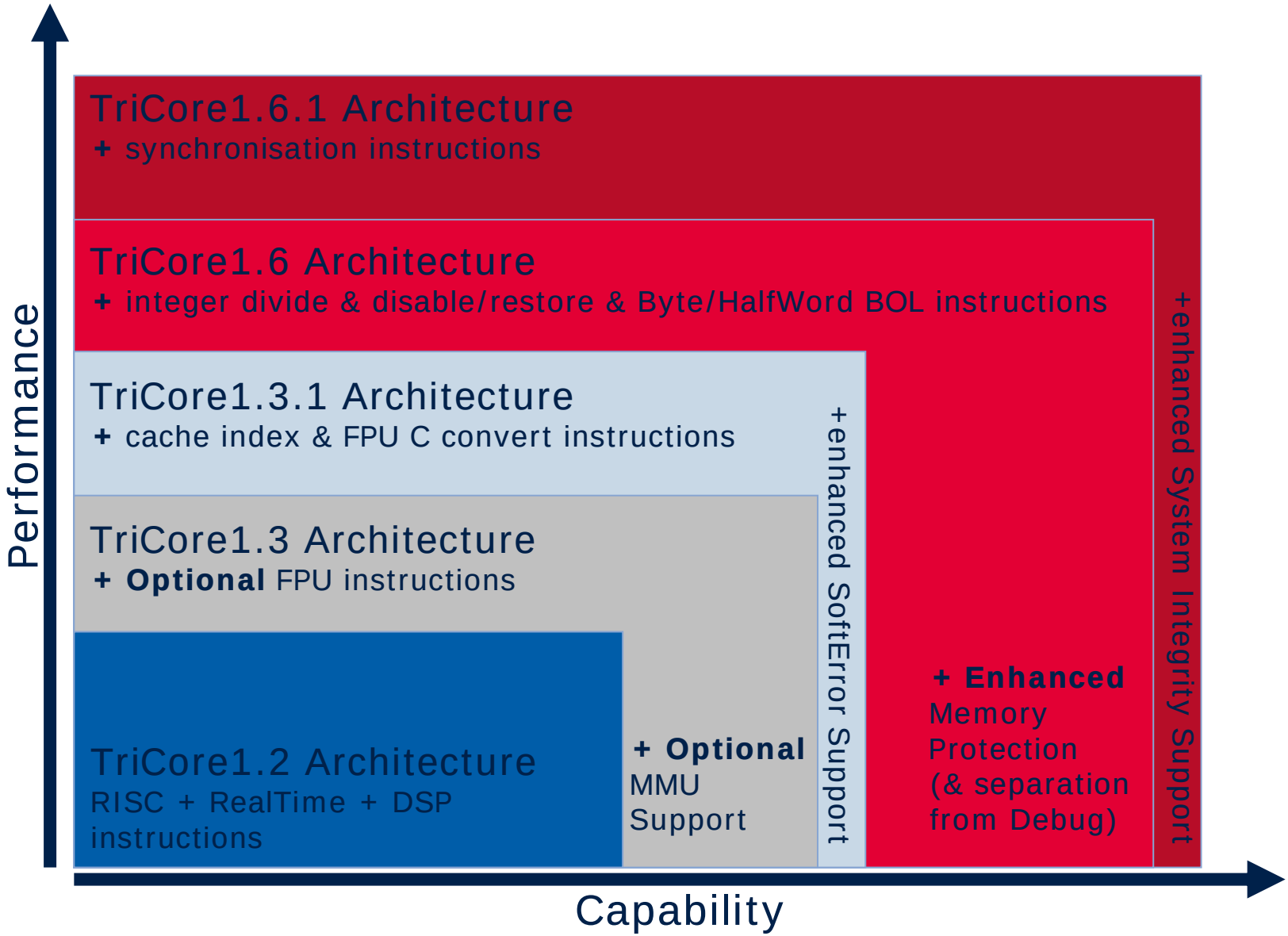
Innovative semiconductor solutions for energy efficiency, communications and security.



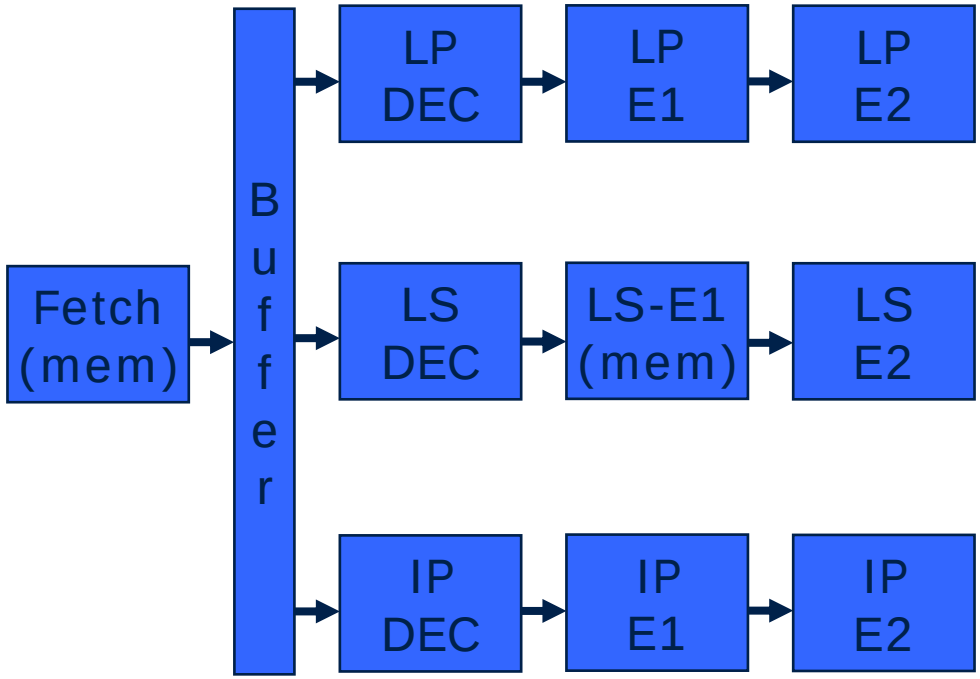
Collateral Material

- TriCore Architecture & Implementation Evolution

TriCore Architecture & Core Evolution

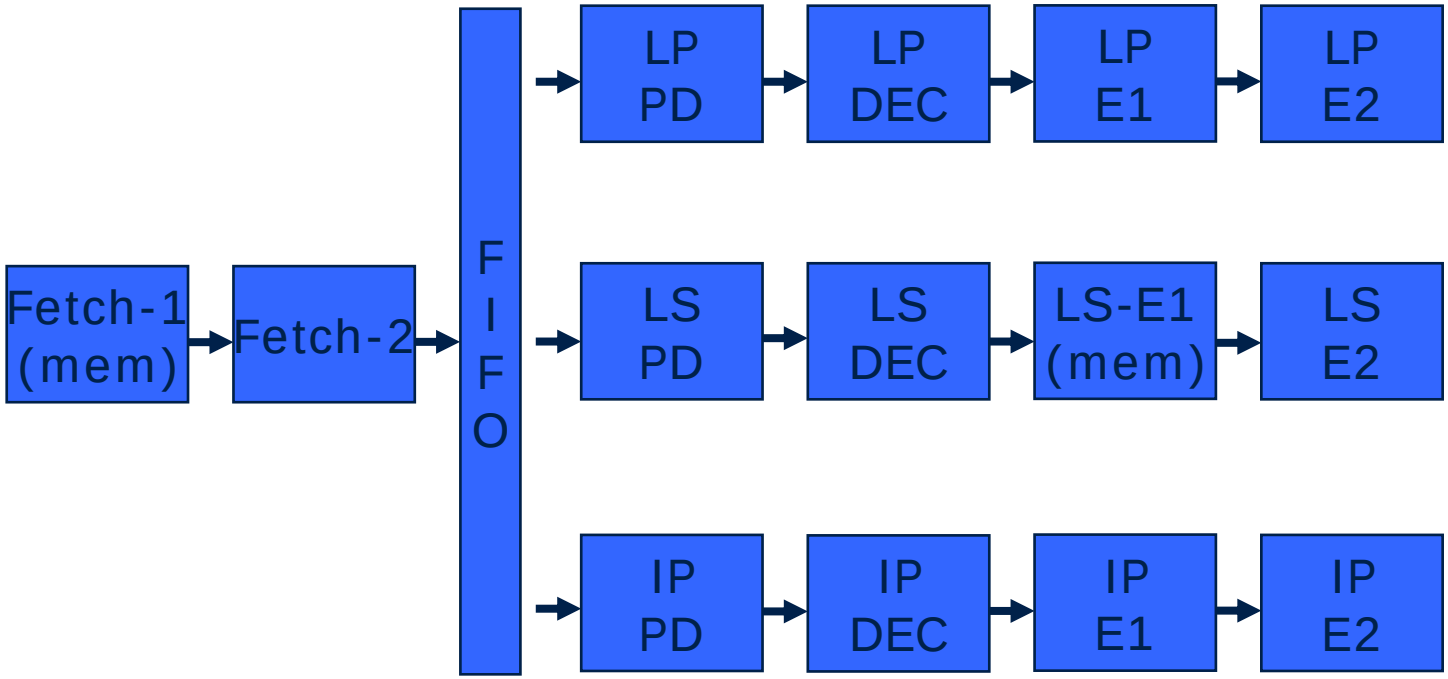


TC1.3 Family Pipeline (upto 180MHz)



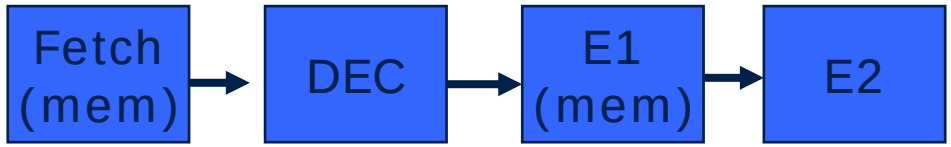
Name	Description
Fetch	Fetch
DEC	Decode
E1/E2	Execute

TC1.6/TC1.6.1 Family Pipeline (upto 300MHz)



Name	Description
Fetch1/2	Fetch
PD	PreDecode
DEC	Decode
E1/E2	Execute

TC1.6-Scalar Pipeline Detail (upto 200MHz)



Name	Description
Fetch	Fetch
DEC	Decode
E1/E2	Execute

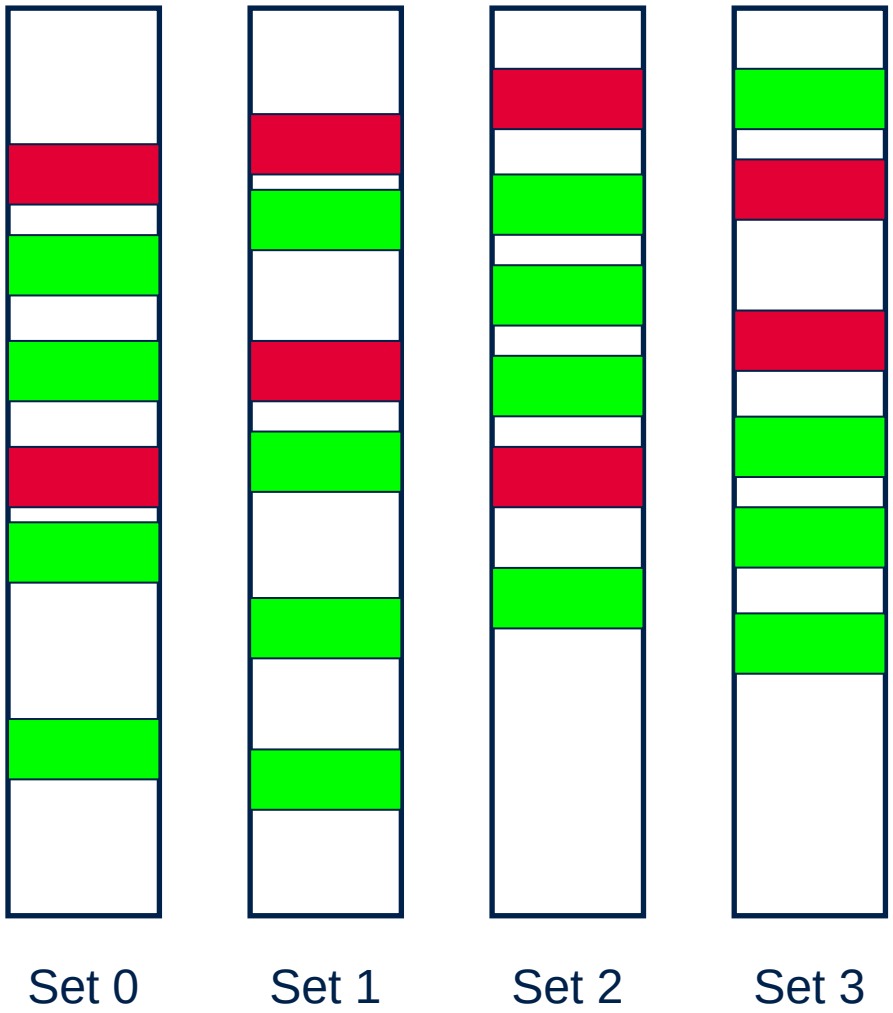
TriCore1.6 is .....

- A Statically Scheduled Superscalar Microprocessor with the following Key Features :-
  - The latest revision of TriCore Architecture and Instruction Set
  - Pipelined TriCore single precision floating point unit
  - 4 way set associative data and instruction cache memories
  - Separate data and instruction scratchpad memories
  - Dynamic branch prediction using a GShare algorithm
  - Flexible memory protection system
  - SRI Crossbar Interconnect

Collateral Material

- Memory Protection Details

TC1.3 Memory Protection Overview

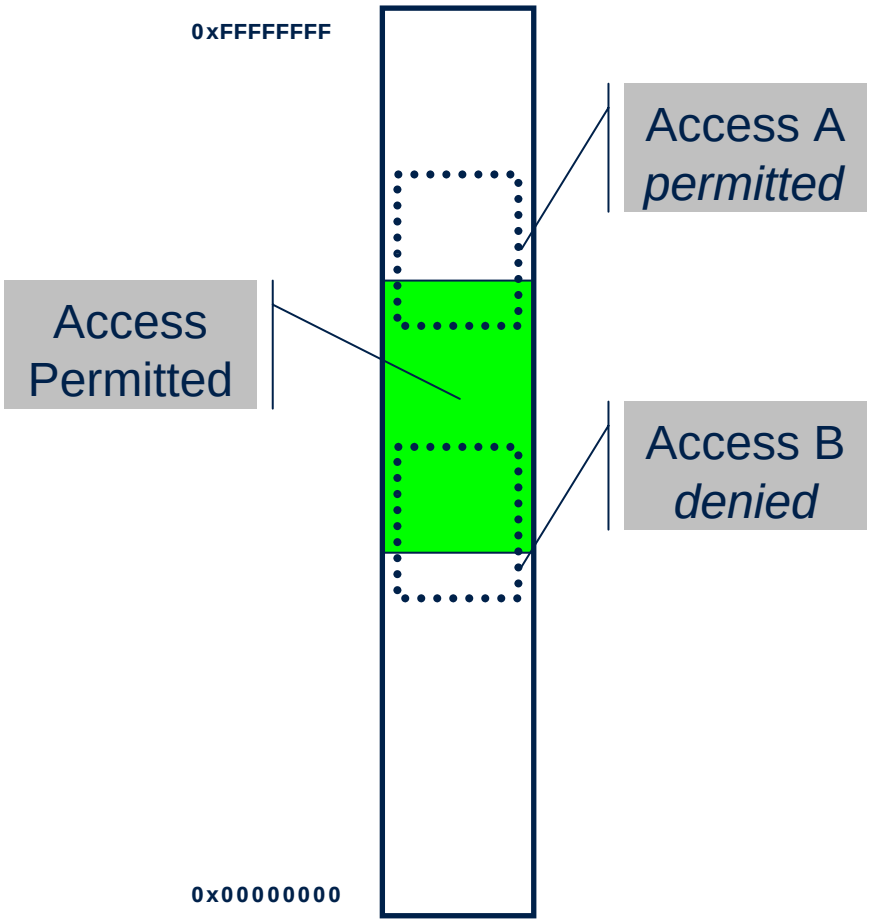


- Access permissions defined by up to 6 address ranges, implemented as:
  - 4 ranges for data access
    - Read access
    - Write access
  - 2 ranges for instruction fetch
    - Execute access
- Granularity of address ranges
  - 1-byte for data access
  - 2-byte for instruction fetch
- 4 Protection Sets in later cores
  - Set 0 dedicated for Exception handling (Supervisor mode)
- Total number of ranges (used in all the sets):
  - 16 ranges for data access
  - 8 ranges for fetch access

TC1.3 Memory Protection Limitations

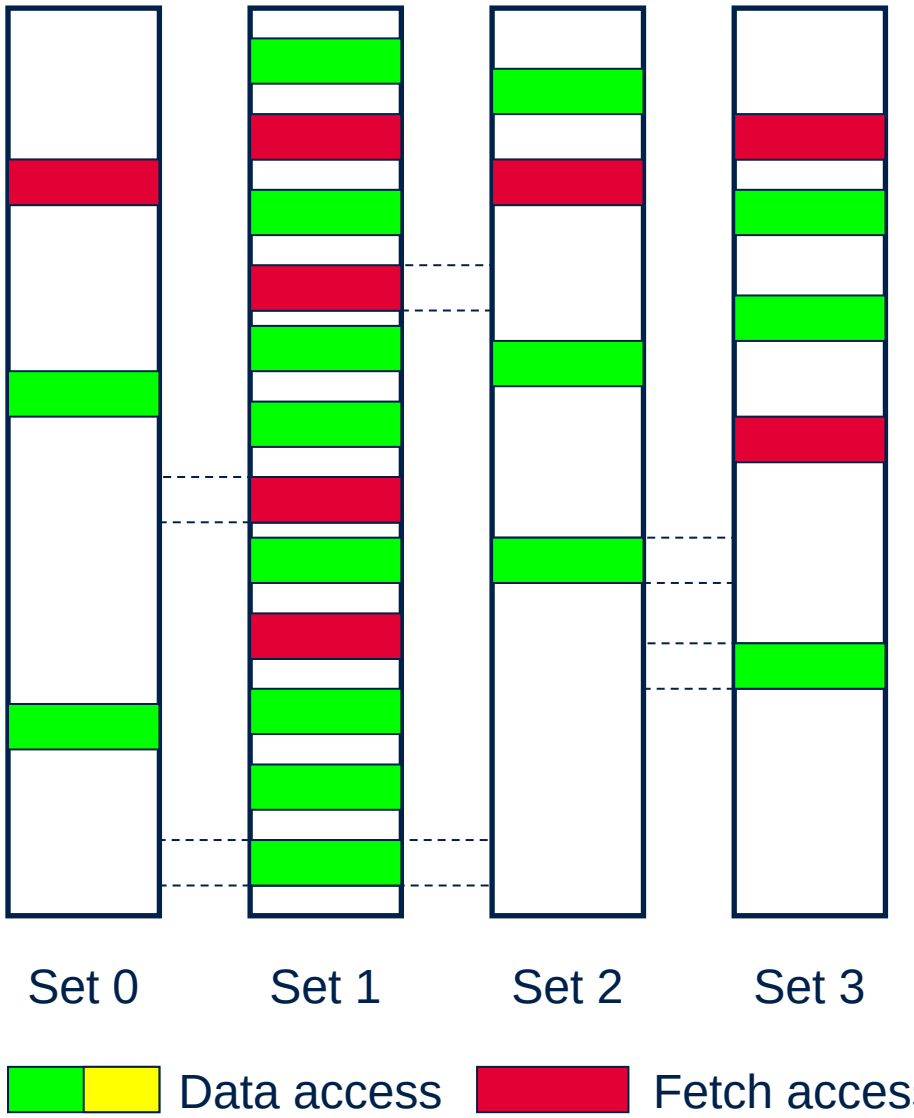
- Memory Protection shared with On-Chip Debug System
  - Access Protection cannot be efficiently used while program is being debugged
- Too few ranges for some applications
  - More complex address space map may require several ranges to define access protection
  - Limitation particularly relevant for the data access
  - Range resources allocated to Protection Sets (i.e. unused ranges in one protection set cannot be used with another set)

TC1.3 Memory Protection
Granularity of Address Ranges



- TriCore Architecture Manual: If an access crosses the range boundary the behaviour is ‘implementation defined’
- For TC1.3 and TC1.6 the start (lowest) address is used for validating the access
- Defining range boundaries with granularity smaller than the maximum access size does **not** change the effective protection

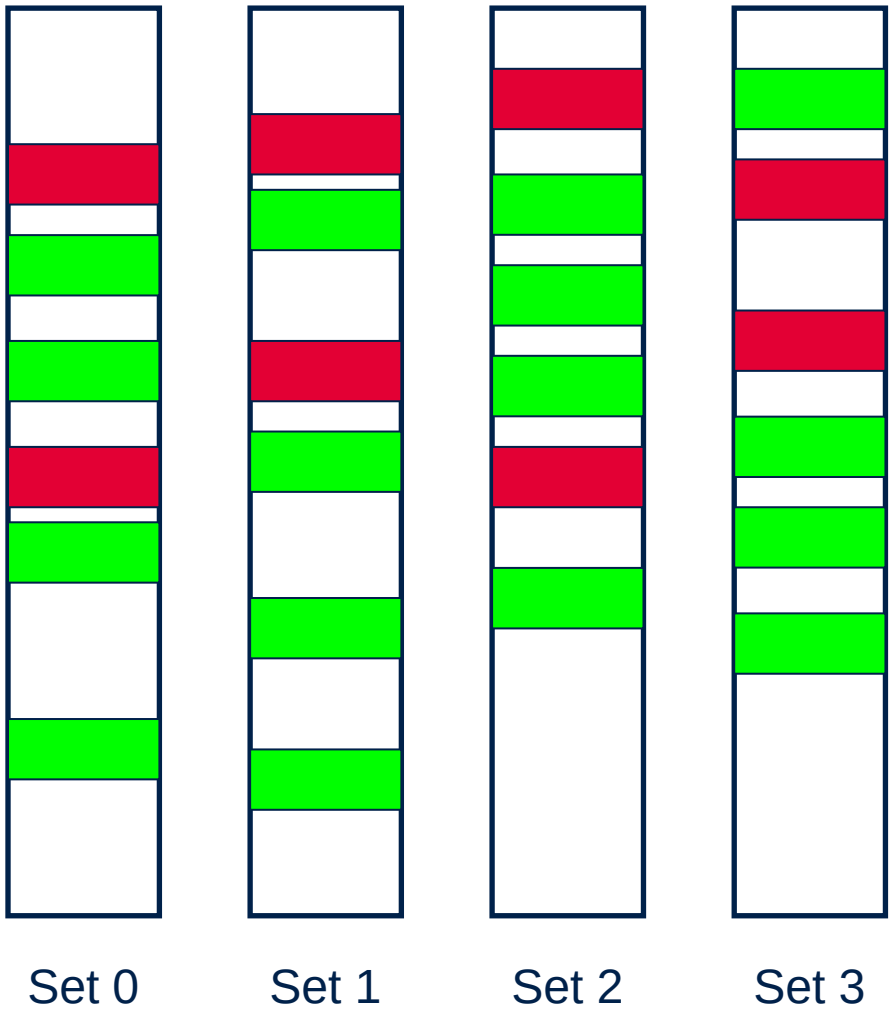
TC1.6 Extended Memory Protection Overview



- Access permissions defined by up to 12 ranges, implemented as:
  - 8 ranges for data access
    - Read access
    - Write access
  - 4 ranges for instruction fetch
    - Execute access
- Granularity of address ranges
  - 8-byte data access and fetch
- 4 Protection Sets
  - Set 0 dedicated for Exception handling (Supervisor mode)
- Total number of ranges (used in all the sets):
  - 16 ranges for data access
  - 8 ranges for code access
- Same range can be used with several Protection Sets
  - With permissions defined independently for each set

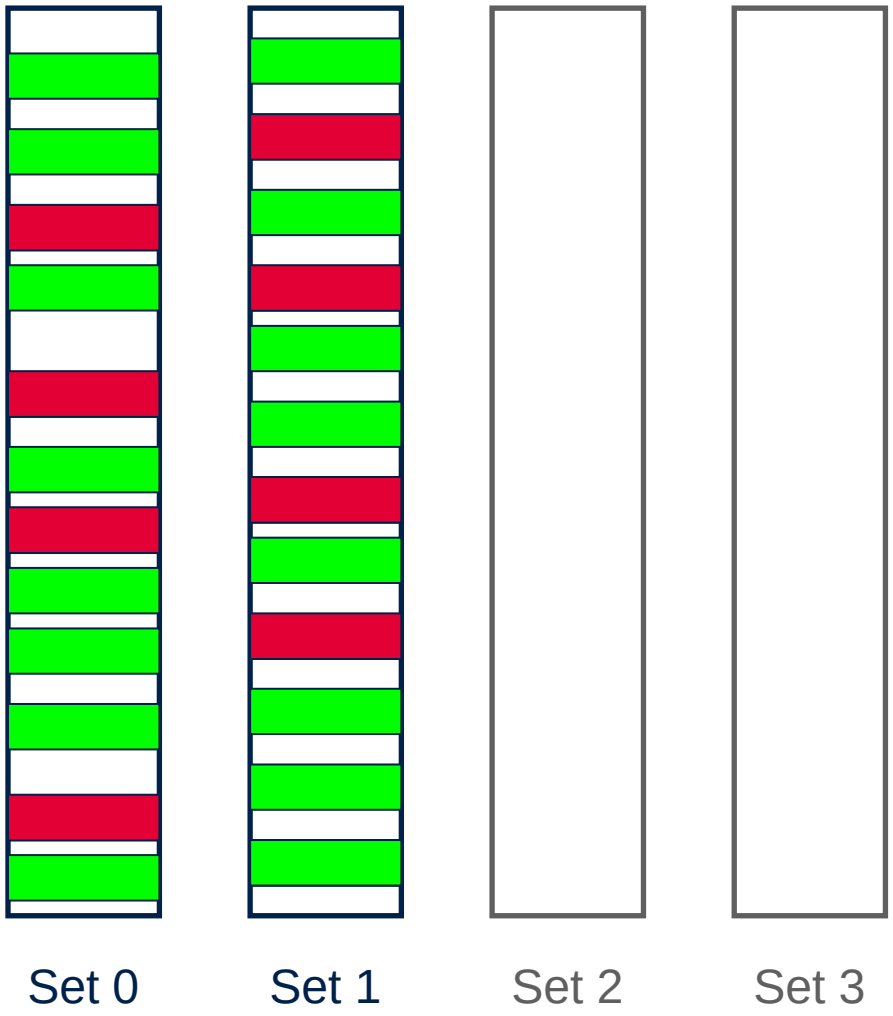
(new/changed features in red)

TC1.6 Extended Memory Protection Usage Example I



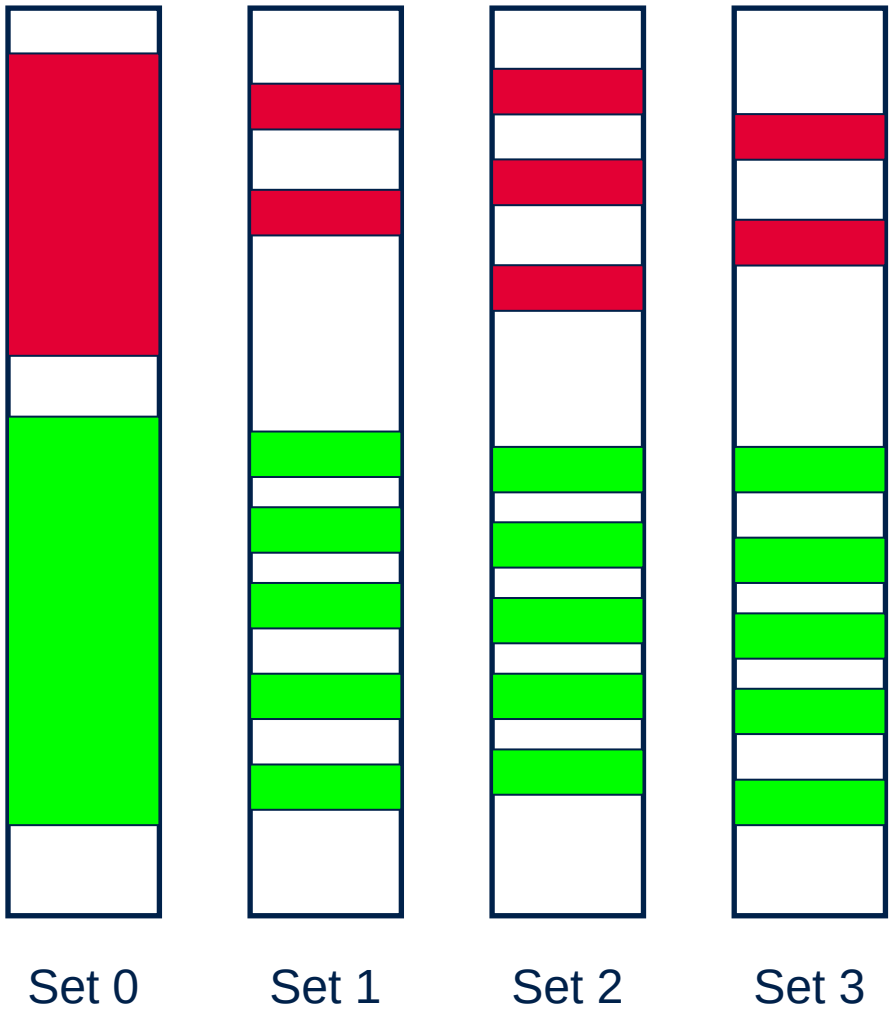
- Ranges divided equally among all the sets
  - 6 ranges per set
  - 4 data ranges per set
  - 2 fetch ranges per set
- Identical to features supported in TC1.3 compliant cores

TC1.6 Extended Memory Protection Usage Example II



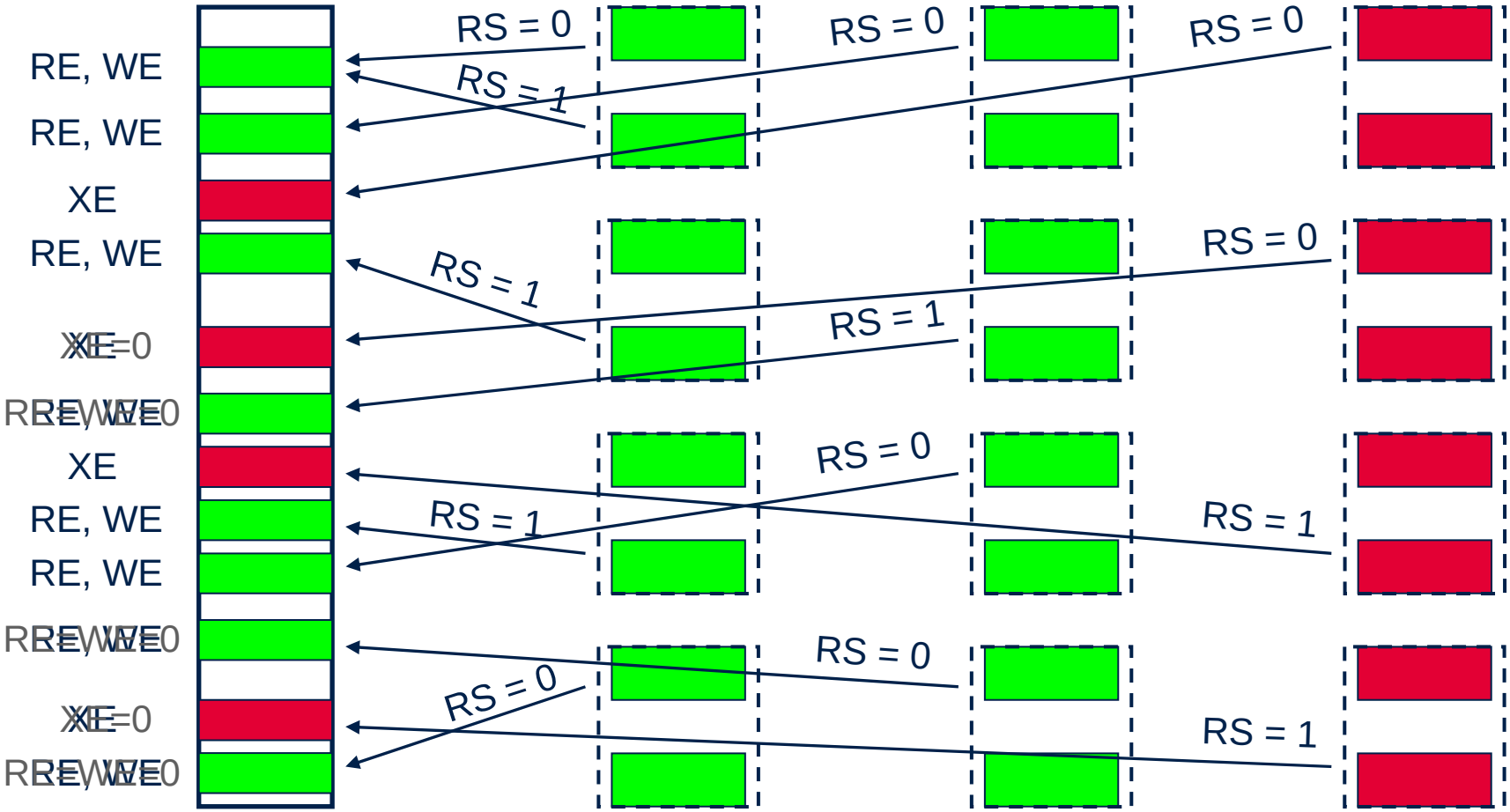
- Only two sets in use:
  - System mode
  - User mode
- Access can be defined using 12 independent ranges per set
  - 8 data ranges per set
  - 4 fetch ranges per set

TC1.6 Extended Memory Protection Usage Example III



- Supervisor can access “everything” (with one data and one fetch range)
- The remaining protection ranges divided across the User sets:
  - 6+ ranges per user set
  - 15 data ranges for 3 user sets - ~5 data ranges per set
  - 7 fetch ranges for 3 user sets ~2+ fetch ranges per set

TC1.6 Extended Memory Protection Configuration, Ranges



Data access range Fetch access range

TC1.6 Extended Memory Protection Configuration, Registers

	Data Access	Instruction Fetch
Address Ranges	Data Protection Range (32 registers, 2 per range) DPR...L –lower bound DPR...U –upper bound 8-byte granularity	Code Protection Range (16 registers, 2 per range) CPR...L – lower bound CPR...U – upper bound 8-byte granularity
Protection Sets	Data Protection Set (4 registers, 1 per set) DPS... Fields: 8* {WE, RE, RS} WE – Write Enable RE – Read Enable RS – Range Select	Code Protection Set (4 registers, 1 per set) CPS... Fields: 4* {XE, RS} XE – Execution Enable RS – Range Select

TC1.6 Extended Memory Protection Backward Compatibility

- Extended memory protection is a superset of TC1.3 memory protection, i.e. for each TC1.3 configuration, a corresponding TC1.6 configuration can be enabled

- Compatibility mode
  - Ranges associated with sets in the old way (fixed mapping)
  - New control bits (Range Select) not accessible
  - Register-level compatible with TC1.3.1
  - The only difference: Granularity at 8-bytes vs. 2-byte fetch, 1-byte data
  - Default mode after CPU reset

Collateral Material

- Temporal Protection System

TC1.6 Temporal Protection System (i)

- The Problem
  - In a real time system each task is assigned a defined memory resource and a defined runtime.
    - Overrunning either the assigned physical memory or the assigned runtime can lead to overall program failure.
  - The Memory protection stops a program overrunning its assigned region of memory.
  - How do we prevent a task overrunning its assigned runtime?
    - Particularly challenging if the task is (necessarily) allowed to disable interrupts for periods of time.

TC1.6 Temporal Protection System (ii)

- Solution
  - Use a counter to monitor the task runtime, Call the OS via a TRAP (rather than interrupt) if runtime is exceeded
    - Two 32-bit decrementing counters clocked by the primary clock
    - Loaded with expected runtime at task start.
    - Easily loaded/unloaded with task state.
    - TAE trap generated on decrement to zero.
    - Counter disabled on load of zero
    - Counter disabled when core is in Idle
    - Counter value and status available via CSFR space.
      - SYSCON register extended with global enable/disable for Temporal Protection
      - Control register Time-out flags, Trap flag
- Primary application: Temporal task protection for AutoSAR

Collateral Material

- Interrupt Acknowledge Option

Acknowledge Decoupling for Safety Systems

- In order to improve plausibility checks on Interrupt Service Routines (ISR) a modification to the Interrupt Controller (ICU) is introduced.
- When an interrupt service request node (SRN) is arbitrated the ICU sends an acknowledge indicating which SRN is the winner. Upon reception of the information the winner SRN automatically clears the request flag. This happens before the ISR executes. Therefore the ISR can't directly double check if the node corresponding to it's interrupt level was active or not.
- The TC1.6 implements a new mode where the feedback after an arbitration round is controlled by software, allowing an ISR to make consistency checks and then enabling the ICU to proceed with terminating the arbitration round and enabling further interrupts.
- On Entering an interrupt the BIT SAFEINT.INT (0xFE30, bit-0) is set. The CPU holds off acknowledging the interrupt until this bit is cleared by a software write. Note that this delays re-arbitration and hence will impact interrupt performance.
- The solution is modal, controlled by the COMPAT.INT bit. By default the current solution is selected.