Atmel®

Cortex-M3 / M4 Introduction

Cortex™

Low-Power Leadership from ARM®

Presentation Outline

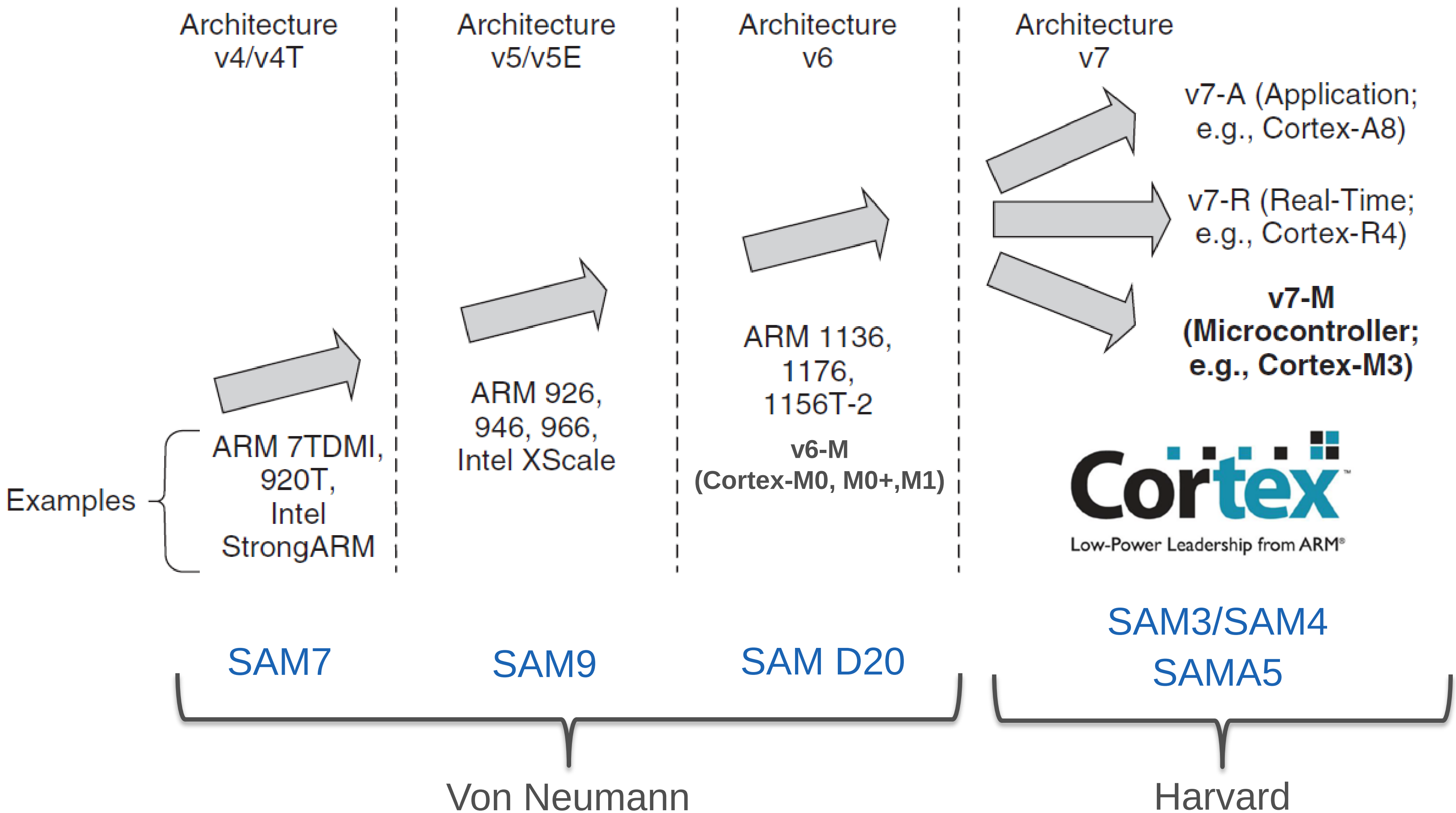
- Introduction
  - ARM Processor Architecture
  - ARM Cortex-M Family

- Cortex-M3 Overview
  - Cortex M3 Processor
  - Cortex M3 Core
  - Cortex-M3 Advanced System Peripherals

- Cortex-M4/M4F Overview
  - Advantages vs. Cortex-M3
  - Cortex-M4F FPU

Introduction

ARM Processor Architecture Evolution



Introduction

ARM Cortex Architectures

- ARM Cortex-M family is based on different architectures:
  - Cortex-M0/0+/1 implements the ARMv6-M architecture (Von Neumann)
  - Cortex-M3/M4 implements the ARMv7-M architecture (Harvard)

- Architecture version v7 (ARMv7) is divided into three profiles:
  - **A profile**, designed for high-performance application platforms
  - **R profile**, designed for high-end embedded systems in which real-time performance is needed
  - **M profile**, designed for deeply embedded microcontroller-type systems

- Architecture version v6-M (ARMv6-M) is a subset of the ARMv7-M profile which provides:
  - A lightweight version of the ARMv7-M instructions set
  - Thumb 16-bit instruction set compatibility
  - Upward software compatibility with ARMv7-M

Introduction

ARM Cortex-M Family

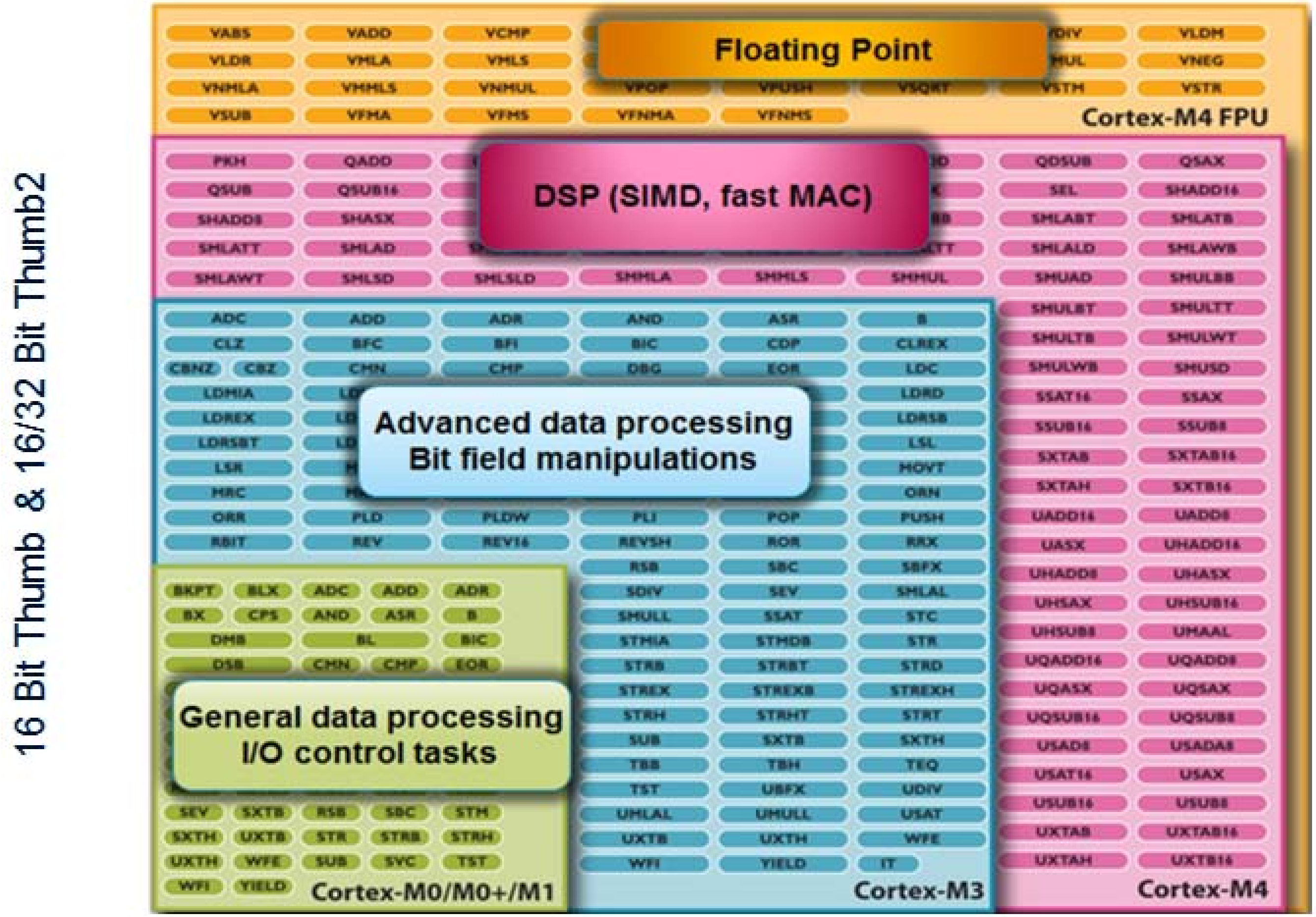
- Why Cortex-M0+?
  - Targeting 8/16-bit and low-end 32-bit market
  - Optimized superset of Cortex-M0
  - Maximize energy efficiency
  - Binary instruction upward compatibility for ARMv6 to ARMv7
  - Upward software compatibility with Cortex-M3 and Cortex-M4 cores

- Why Cortex-M3?
  - First ARM processor based on the ARMv7-M architecture and designed to achieve high system performance in power and cost-sensitive embedded applications such as microcontrollers

- Why Cortex-M4?
  - Designed for applications requiring more computational performance
  - Cortex-M4 frees CPU resources in case digital signal processing tasks are executed (less active cycles are needed)
  - Cortex M4**F** adds a single precision Floating-Point Unit (FPU)

Introduction

ARM Cortex-M Instruction Set



Introduction

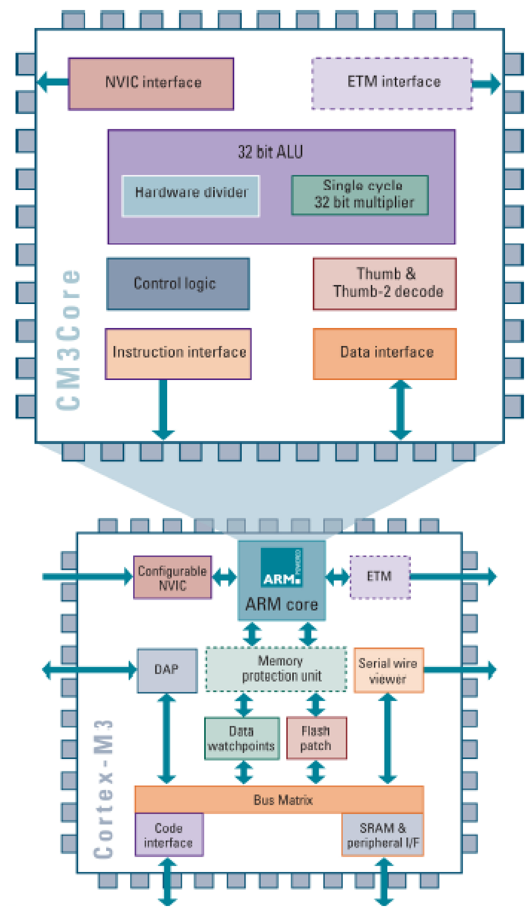
ARM Cortex-M Comparison Table

Cortex-M Core	Architecture	Pipeline	Thumb / Thumb 2	MPU	DSP	FPU	Performance (DMPIS/MHz)	Dynamic Power consumption (uW/MHz)
M0	Von Neumann	3	Most / Subset	No	No	No	0.84	16.4
M0+	Von Neumann	2	Most / Subset	Opt.	No	No	0.93	9.8
M3	Harvard	3	All / All	Opt.	No	No	1.25	32
M4	Harvard	3	All / All	Opt.	Yes	Opt.	1.25	33

Cortex-M3 Overview

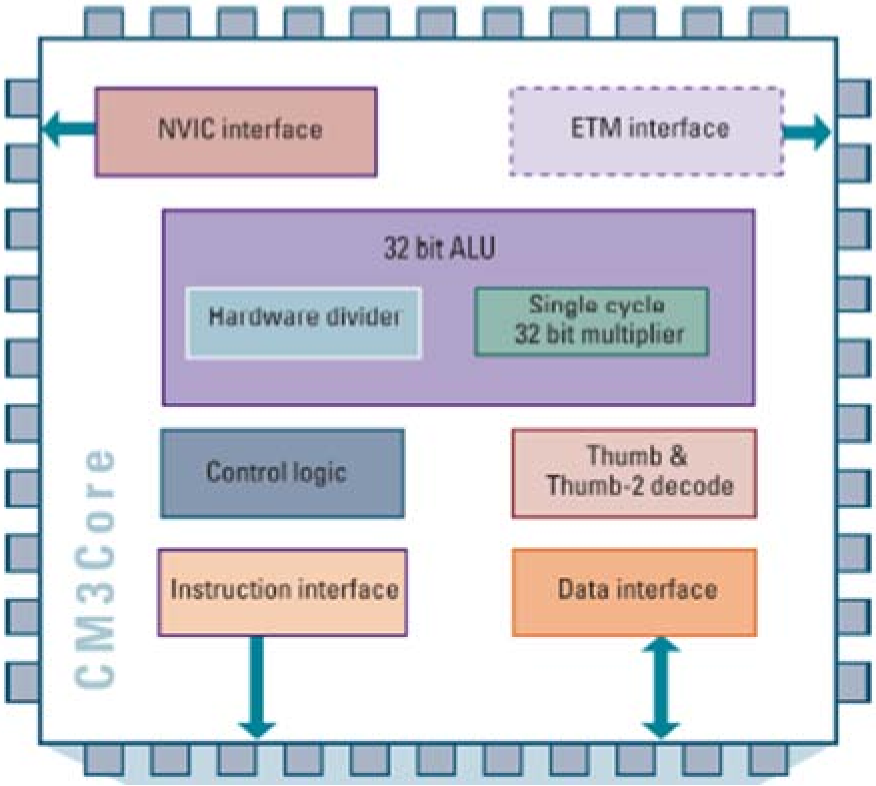
Cortex-M3 Processor Overview

- The Cortex-M3 is a Hierarchical processor integrating core and advanced system peripherals
- It is designed for :
  - Performance and Energy Efficiency
  - Reduced memory requirements
  - Rich connectivity
  - To be fast and easy to program



Atmel

Cortex-M3 Core

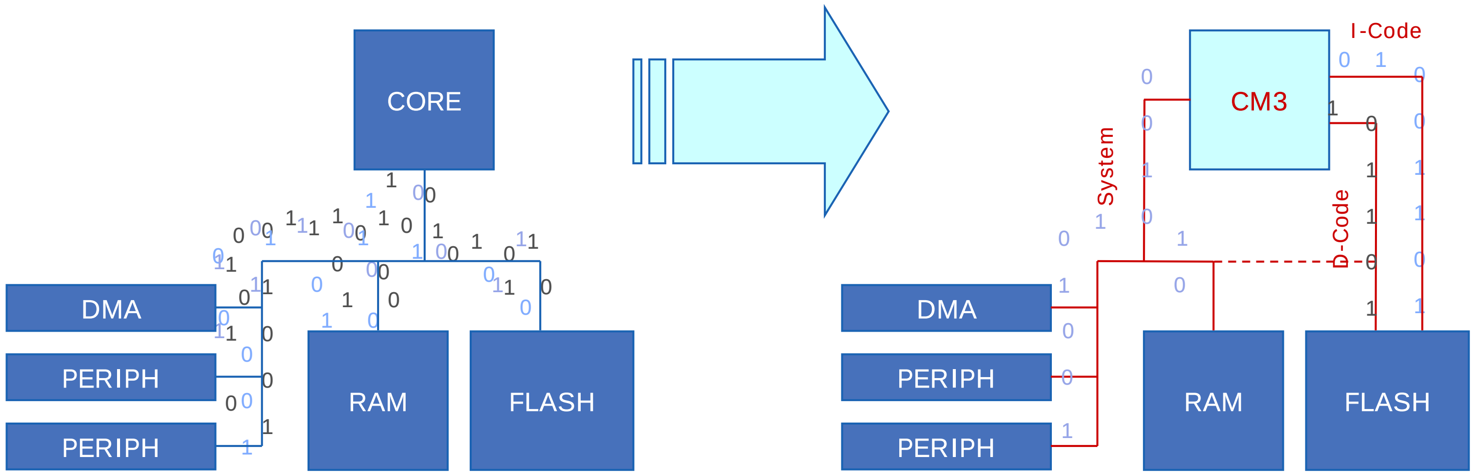


- Harvard architecture
- Support Thumb® and Thumb®-2
- 3-stage pipeline w. branch speculation
- Complete hardware support for interrupts
- ALU w. H/W divide and single cycle multiply
- Sleep control and power-down modes
- Memory management features
  (Unaligned Data Access and bit banding)

Cortex-M3 Core

Harvard architecture

- Separate buses for instructions and data speeding application execution.



Von Neumann architecture

Harvard architecture

Cortex-M3 Core

3-stage pipeline with branch speculation

INST1	Fetch	Decode	Execute		
INST2		Fetch	Decode	Execute	
INST3			Fetch	Decode	Execute

Without Branch speculation

branch impact on pipeline

Break

BRCH	Fetch	Decode	Execute	
INST2		Fetch	Decode	
INST3			Fetch	
INST2’				Fetch

With Branch speculation

Case branch condition NOK :

BRCH	Fetch	Decode	Execute
INST2		Fetch	Decode
INST3			Fetch
INST2’		sFetch	

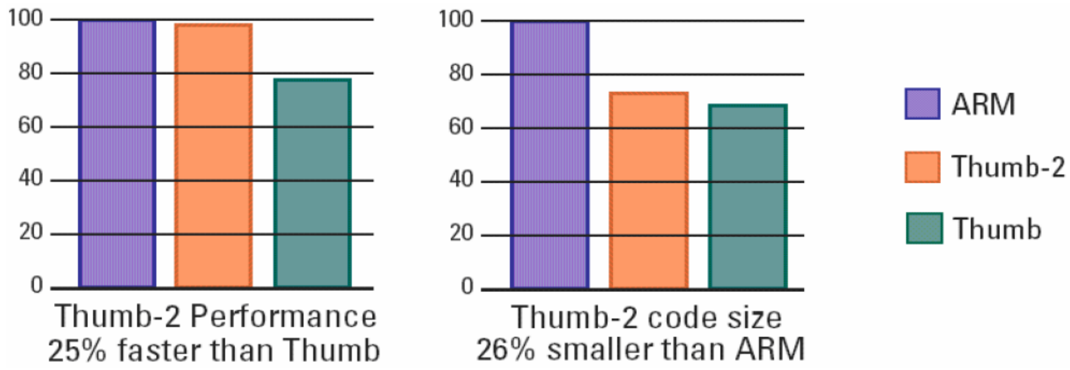
Case branch condition OK :

BRCH	Fetch	Decode	Execute
INST2		Fetch	
INST2’		sFetch	Decode
INST3’			Fetch

Cortex-M3 Core

Thumb-2 Instruction Set

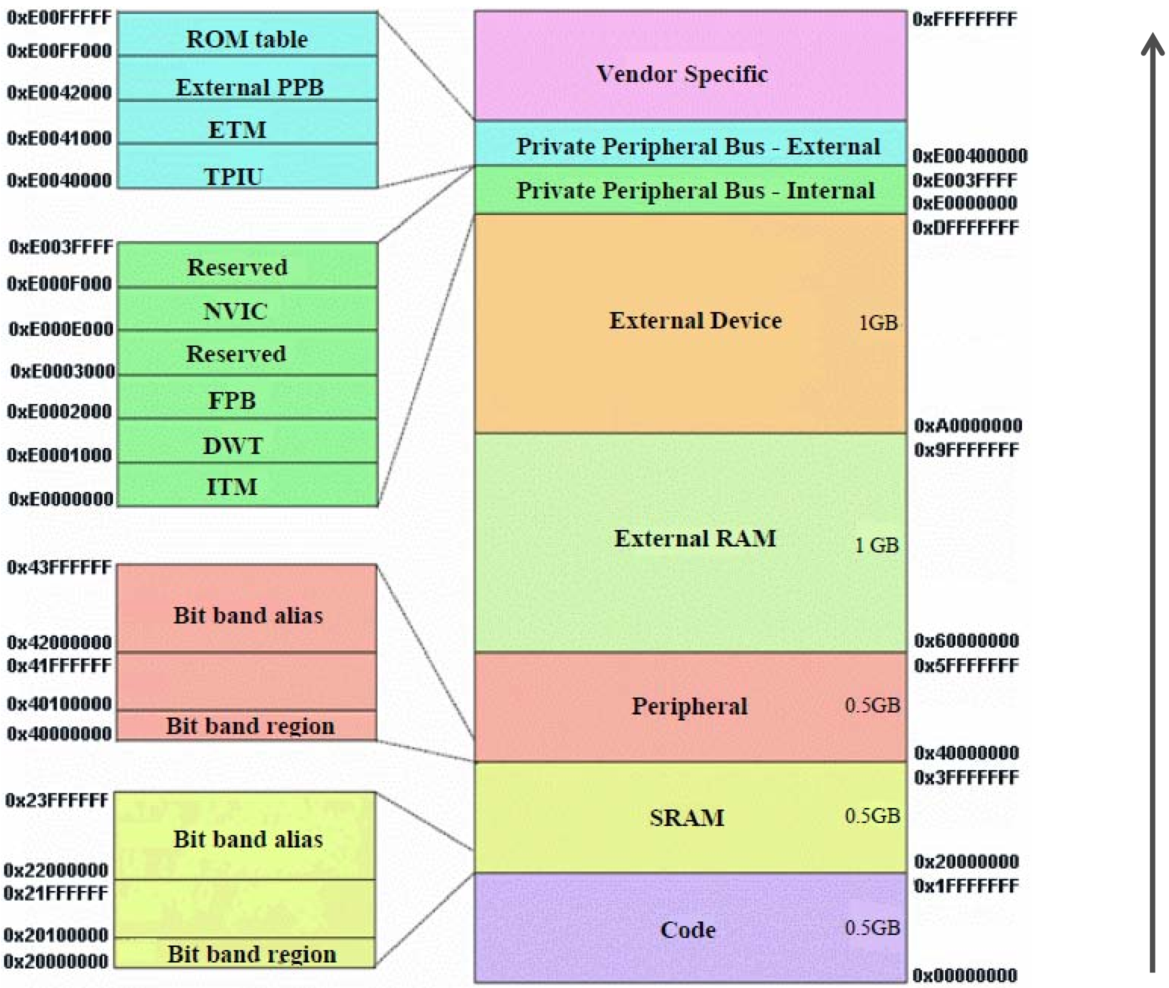
- Blend of 16 and 32-bit instructions that delivers significant benefits in terms of ease of use, code size and performance
- Backward compatible with 16-bit Thumb instruction set, but Not backward compatible with 32-bit ARM instruction set
- Automatic optimization for both performance and code density, without the need for complex interworking



- New instructions that make it easier to write compact code
  - BFI and BFC instructions for bit-field manipulations, Multiply, Divide and a new If-Then construct

Cortex-M3 Core

Memory Map

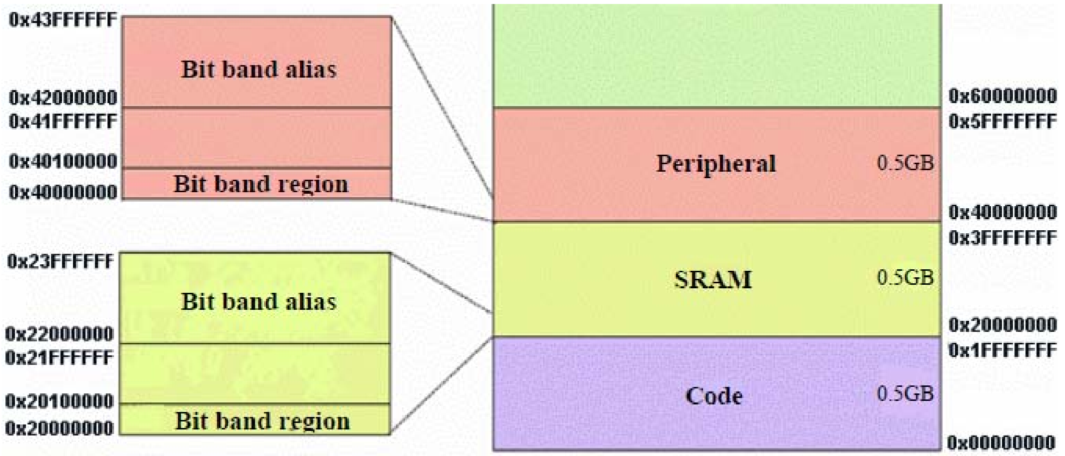


Up to 4GBytes of addressable memory

Cortex-M3 Core

Bit Banding (1/2)

- Atomic Bit Set or Clear performed through memory Bit Banding
  - Bottom 1MB of the Peripheral and SRAM address spaces is reserved for bit-band accesses
  - Data Accesses to the 32MB bit band alias region are remapped to this 1MB address space.



Cortex-M3 Core

Bit Banding (2/2)

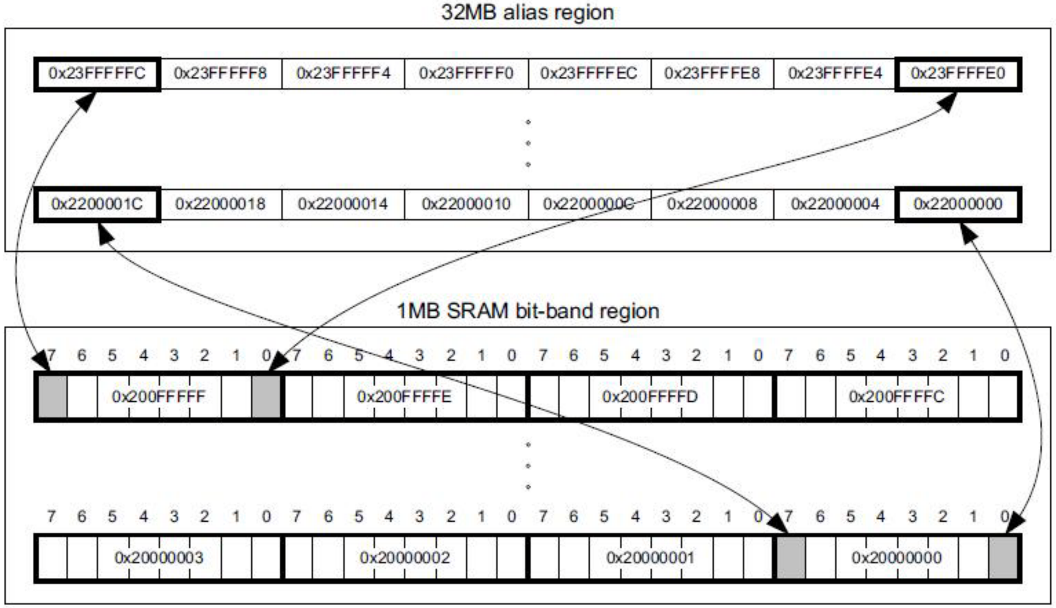
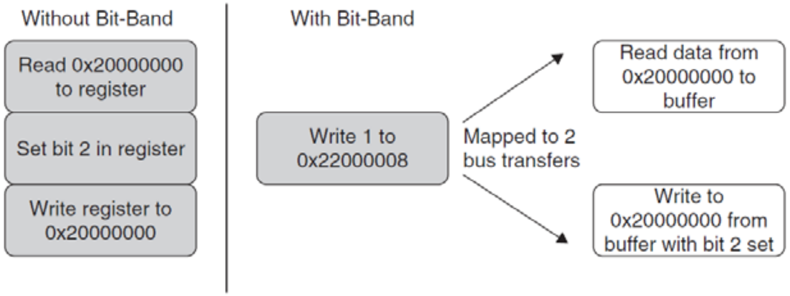


Figure 2-1 Bit-band mapping

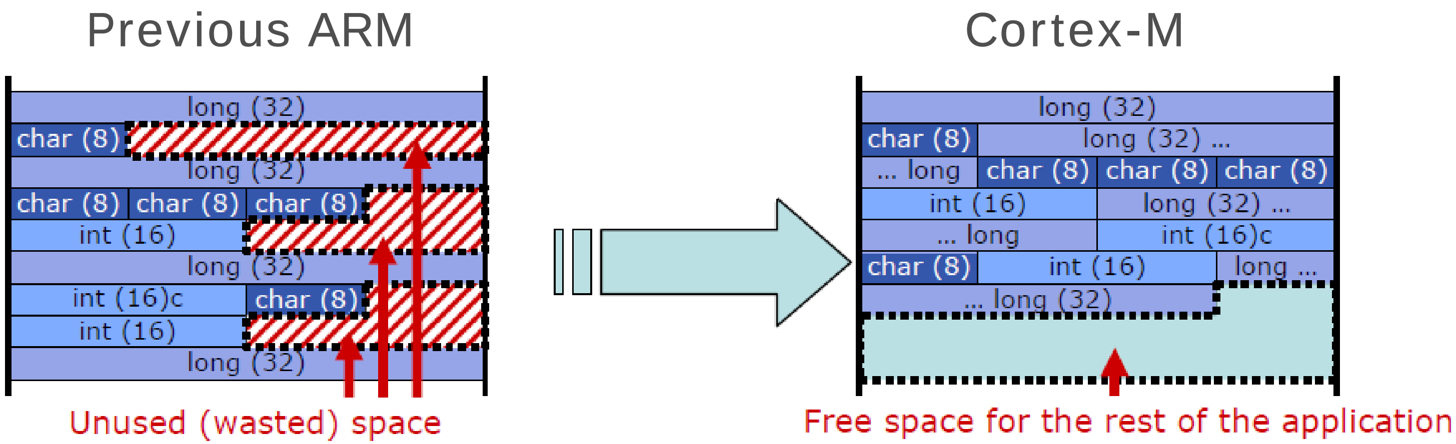
- For example, to set bit 2 in word data in address 0x20000000



Cortex-M3 Core

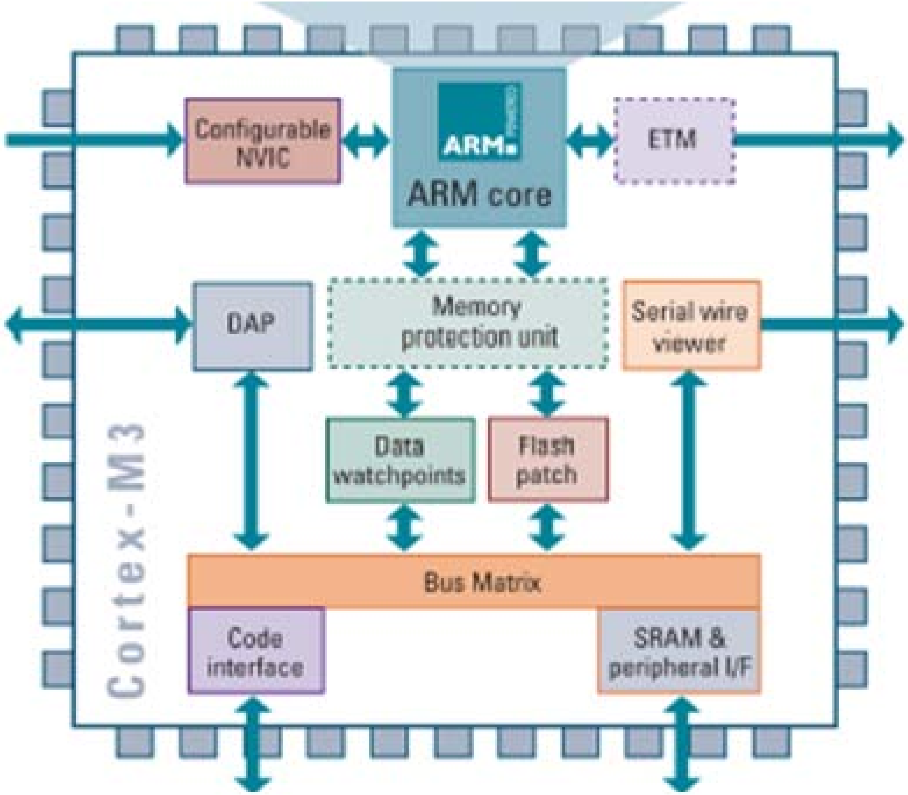
Unaligned Data Access

- Data memory accesses can be defined as aligned or unaligned improving data constant and RAM utilization



Cortex-M3 Advanced System Peripherals

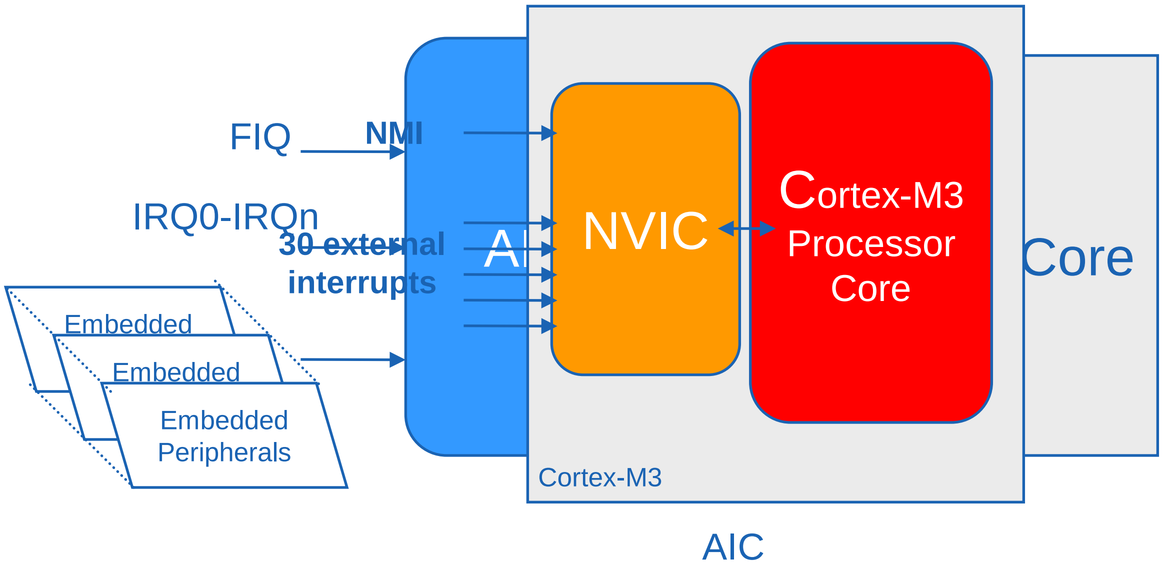
Overview



- Nested Vectored interrupt controller
- Debug an trace features
- Optional Memory protection unit

Cortex-M3 Advanced System Peripherals

Nested Vectored Interrupt Controller (NVIC)



Now integrated in the Cortex-Mx core

Cortex-M3 Advanced System Peripherals

NVIC - Features overview

- Fixed number of system exceptions (interrupts/faults)
- Based on a vector table stored at the beginning of the code
  - No need to use software to determine and branch to the starting address of the ISR
  - Handlers can be written entirely in C
- Manage the interrupt entry /Return (Context PUSH/POP)
  - Interrupt entry/exit is « micro-coded » (controlled by hardware)
- Interrupt prioritization mechanism
- Manage the core sleep modes (WFI , WFE)

Cortex-M3 Advanced System Peripherals

NVIC – Vector table

N°	Exception Type	Priority	Vector address	Descriptions
0	-	-	0x00	MSP Initial Value (Main Stack Pointer)
1	Reset	-3	0x04	Reset
2	NMI	-2	0x08	Non-Maskable Interrupt
3	Hard Fault	-1	0x0C	Error during exception processing
4	Memory Management Fault	Configurable	0x10	MPU violation
5	Bus Fault	Configurable	0x14	Bus error (Prefetch or data abort)
6	Usage Fault	Configurable	0x18	Exceptions due to program errors
11	SVCall	Configurable	0x2C	SVC instruction
12	Debug Monitor	Configurable	0x30	Exception for debug
14	PendSV	Configurable	0x38	
15	SysTick	Configurable	0x3C	System Tick Timer
16 and above	Interrupt (IRQ)	Configurable	0x40	External interrupt

Cortex-M3 Advanced System Peripherals

NVIC - Interrupt Priority

- Perform using 4 bits , divided into pre-empting priority levels and "sub-priority" levels
  - Sub-priority levels only have an effect if the pre-empting priority levels are the same
  - The software programmable PRIGROUP register field of the NVIC chooses how many of the 4-bits are used for "group-priority" and how many are used for "sub-priority"
  - Group priority is the pre-empting priority
- Lower numbers are higher priority
- Hardware interrupt number is lowest level of prioritization
  - IRQ3 is higher priority than IRQ4 if the priority registers are programmed the same

Cortex-M3 Advanced System Peripherals

NVIC - Interrupt Entry

- Processor state is automatically pushed onto the stack over the data bus (automatically pushes registers R0–R3, R12, LR, PSR, and PC in the stack)
- In parallel, ISR is prefetched by the processor on the instruction bus

Data
Old SP
PSR
PC
LR
R12
R3
R2
R1
R0

Atmel

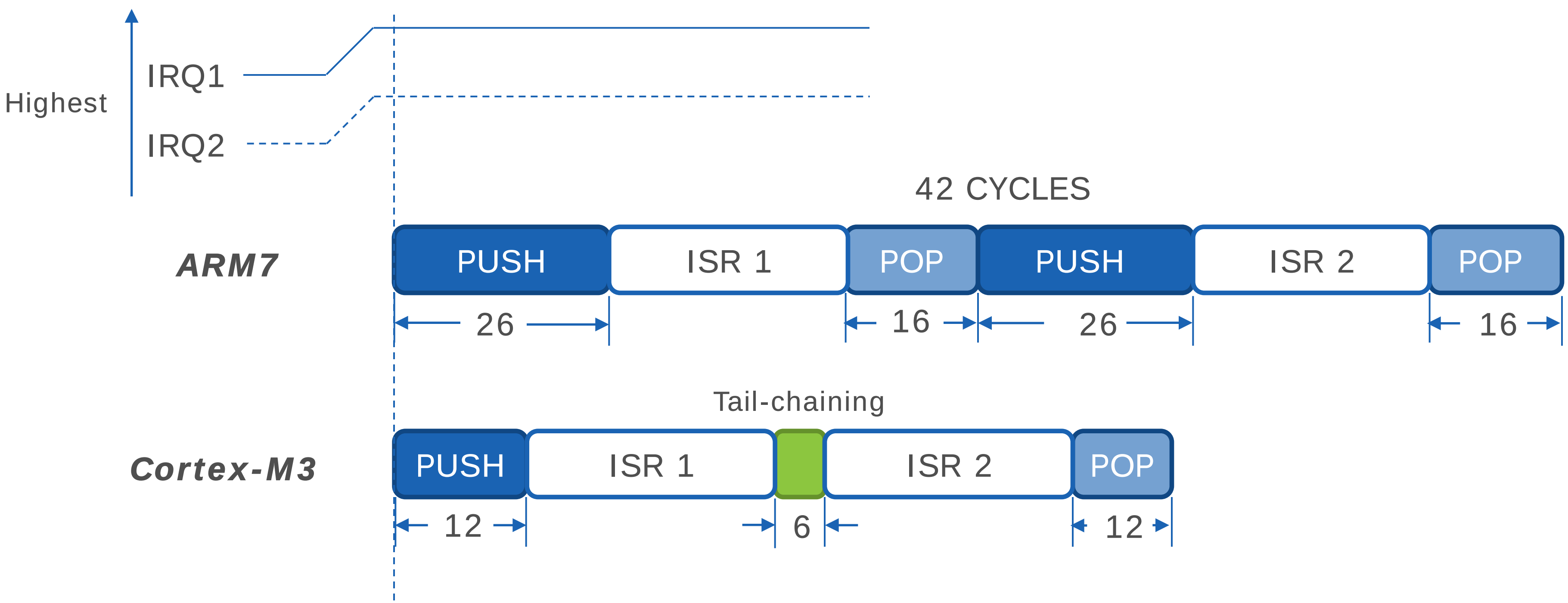
Cortex-M3 Advanced System Peripherals

NVIC - Interrupt Return

- Processor state is automatically restored from the stack.
- In parallel, interrupted instruction is prefetched to be ready for execution upon completion of stack restore.

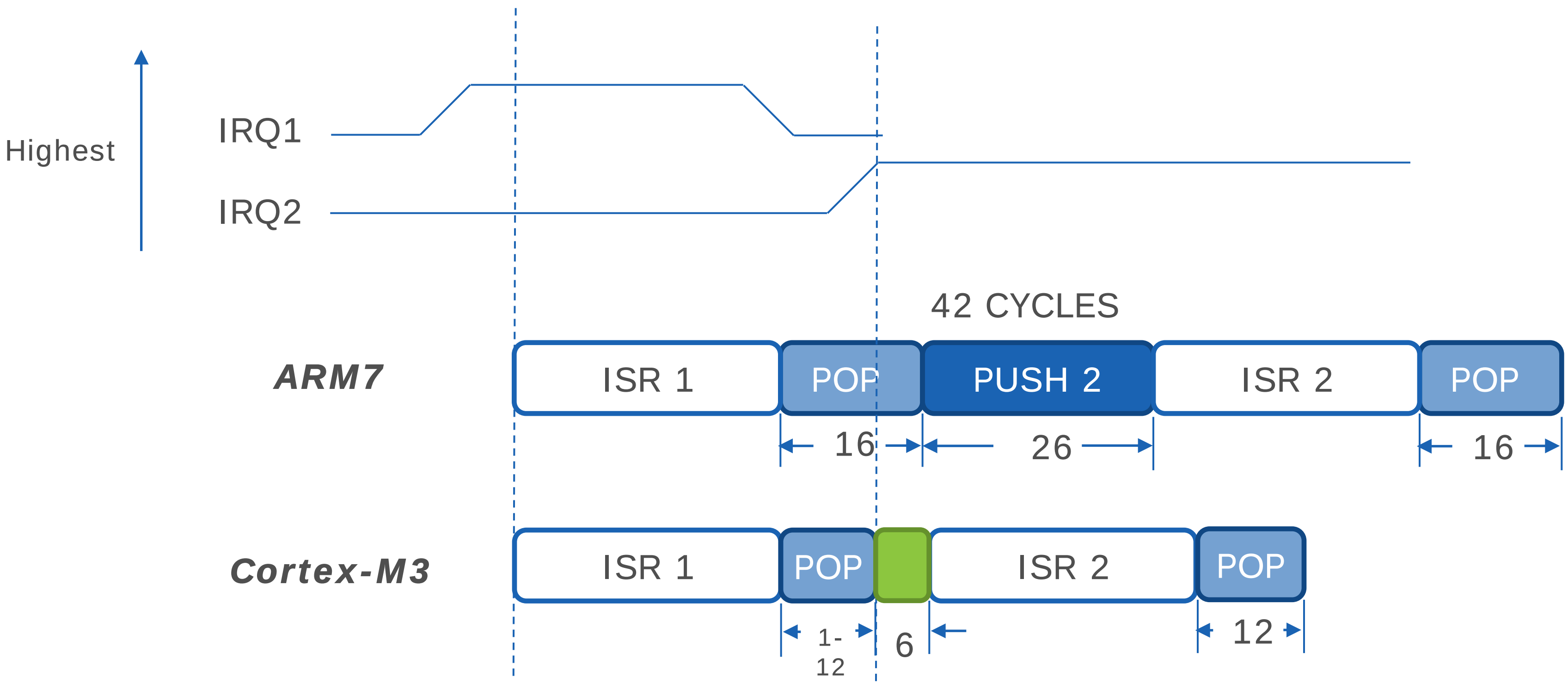
Cortex-M3 Advanced System Peripherals

NVIC - Tail Chaining



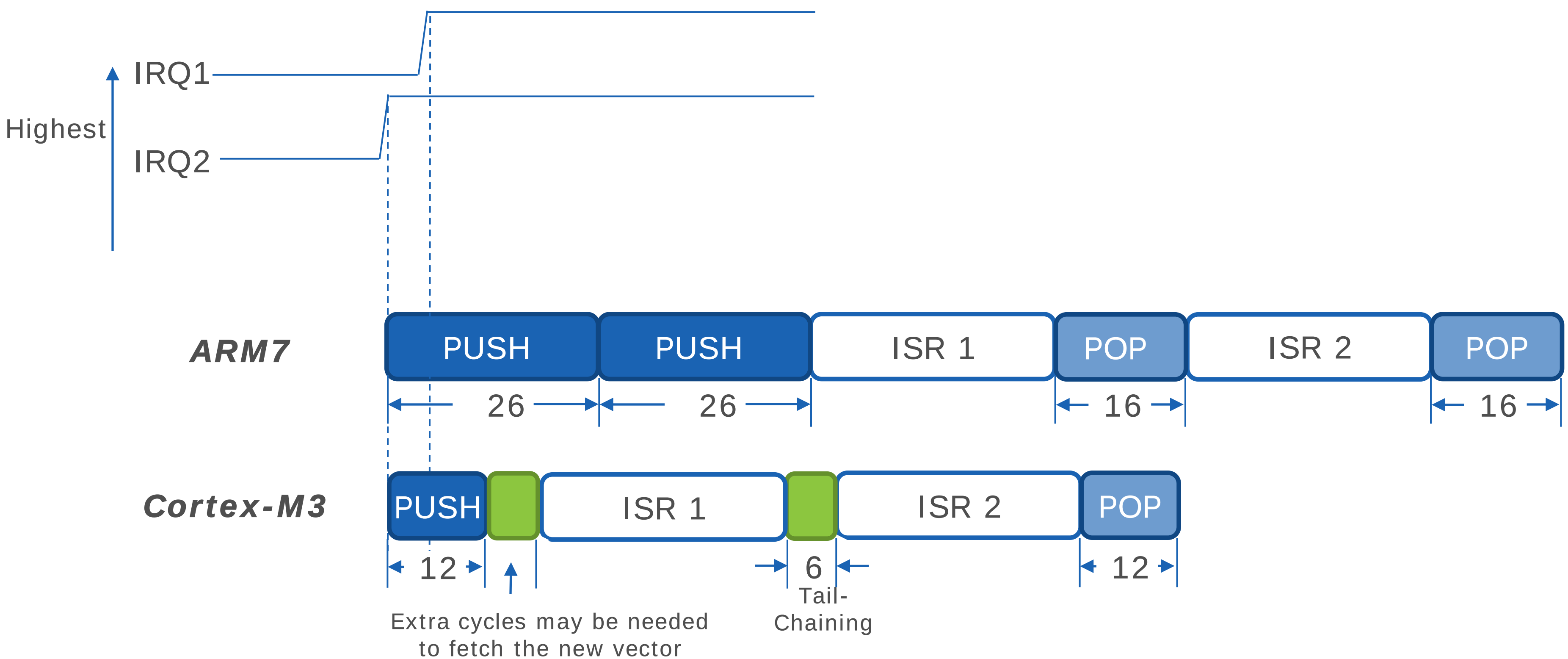
Cortex-M3 Advanced System Peripherals

NVIC - Preemption



Cortex-M3 Advanced System Peripherals

NVIC - Late Arriving



Cortex-M3 Advanced System Peripherals

Core Operating Modes

- Two operating modes, Thread and Handler and two levels of access for the code, privileged and unprivileged (user)

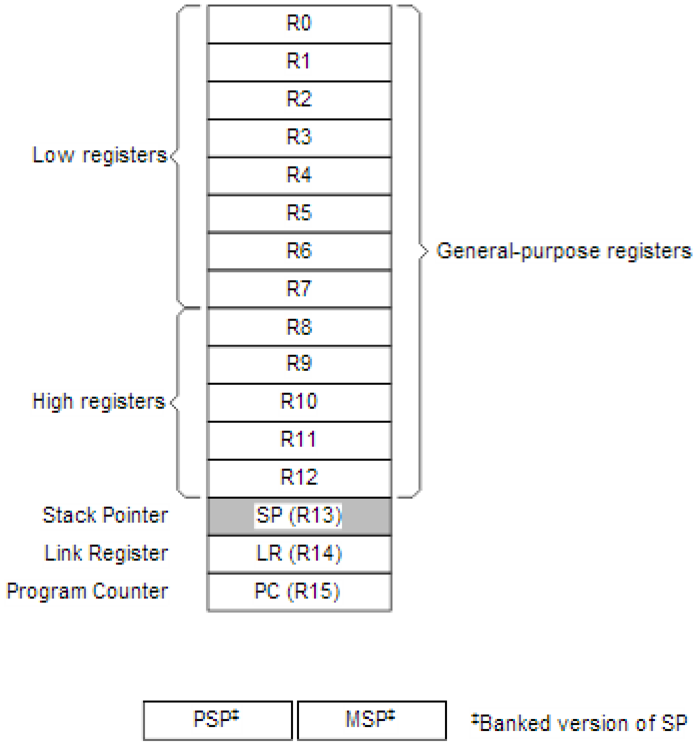
	Privileged	*User*
When running an exception (Handler Mode)	X	
When running main program (Thread Mode)	X	X

- Thread mode after reset with privileged access rights
  - In the privileged state, a program has access to all memory ranges, can use all supported instructions and access the System Timer, NVIC or System Control Block

Cortex-M3 Advanced System Peripherals

Core Registers

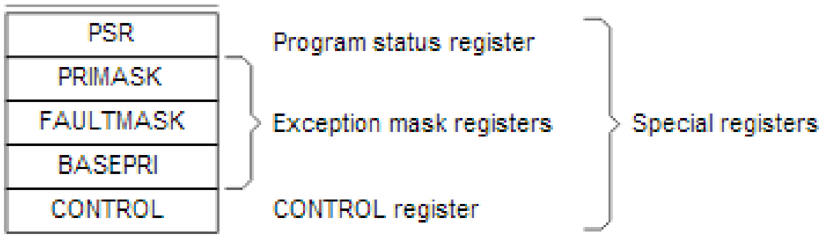
- General Purpose Registers : R0-R12
  - 32-bit general-purpose registers for data operations
- Two banked Stack Pointers (SP): R13
  - MSP: Main Stack Pointer (privileged mode)
  - PSP: Process Stack Pointer (user mode)
- Link Register (LR): R14
  - Stores the return information for subroutines, function calls, and exceptions
- Program Counter (PC): R15
  - Contains the current program address.
  - On reset, PC = Reset Vector value



Cortex-M3 Advanced System Peripherals

Special Core Registers

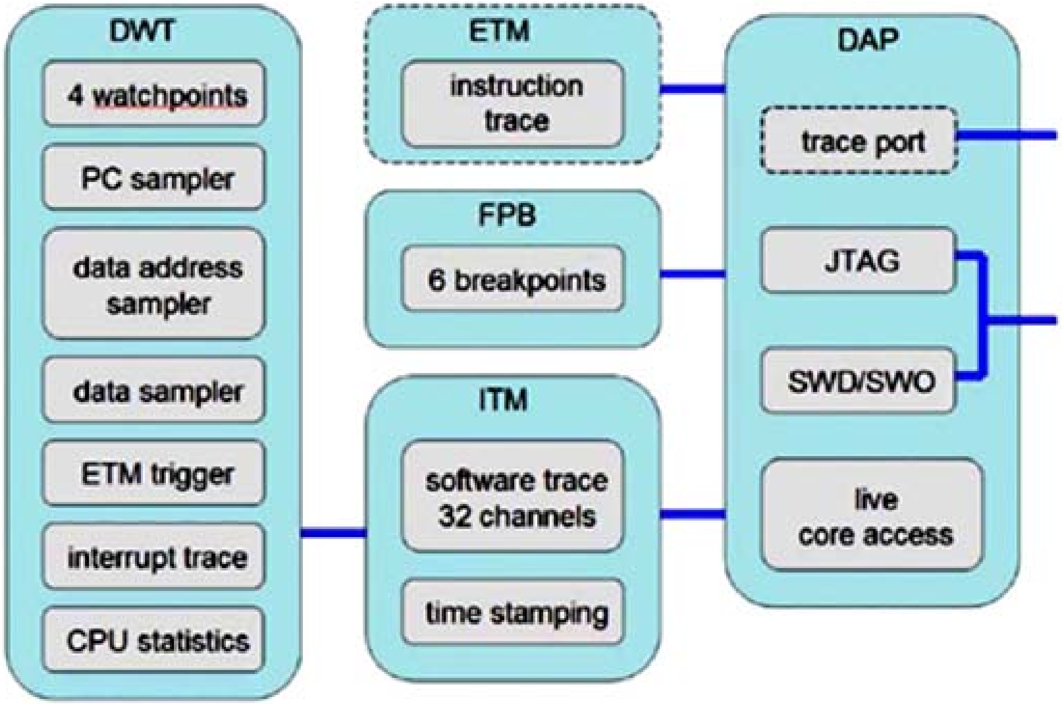
- Program Status Registers (xPSR):
  - Provide ALU flags (zero flag, carry flag), execution status, and current executing interrupt number
- PRIMASK:
  - Disable all interrupts except the non maskable interrupt (NMI) and HardFault
- FAULTMASK
  - Disable all interrupts except the NMI
- BASEPRI
  - Disable all interrupts of specific priority level or lower priority level
- CONTROL
  - Define privileged status and stack pointer selection



Cortex-M3 Advanced System Peripherals

Debug Features - Overview

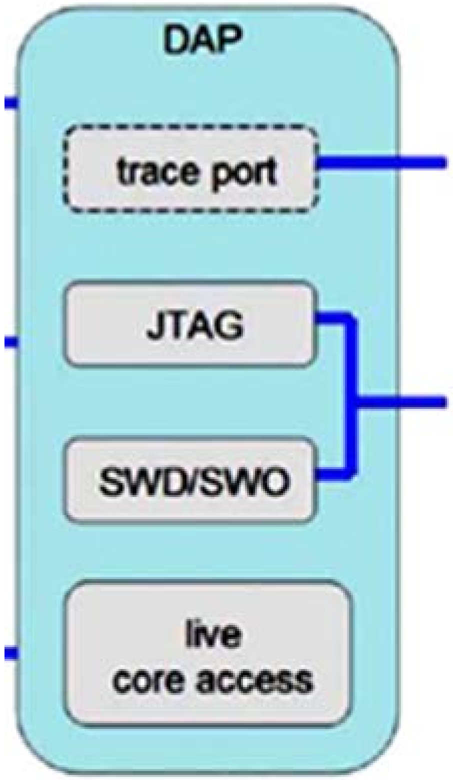
- The Cortex-M3 implements several hardware debug features:



- ***D****ebug* ***A****ccess* ***P****ort* (DAP).
- ***F****lash* ***P****atch and* ***B****reakpoint (FPB)*.
- Optional ***E****mbedded* ***T****race* ***M****acrocell* (ETM).
- ***I****nstrumentation* ***T****race* ***M****acrocell (ITM)*.
- ***D****ata* ***W****atchpoint and* ***T****race (DWT)*.

Cortex-M3 Advanced System Peripherals

Debug Features - Debug Access Port (DAP)



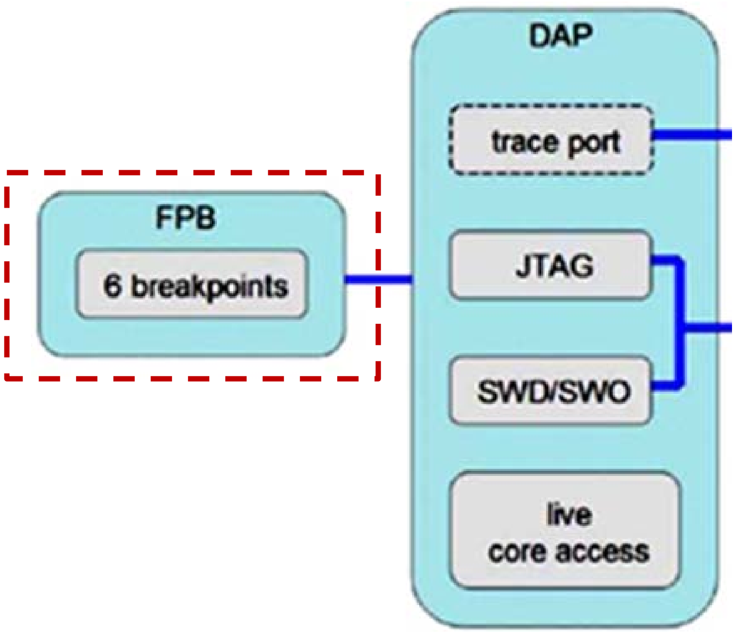
- The DAP is an AHB-AP interface that allows live access to the core and all the peripherals
- Two different supported implementations*:
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Serial Wire Debug Port (SW-DP).

(*) Depending on the silicon manufacturer

Name	JTAG Debug Port		SWD Debug Port	
	Type	Description	Type	Description
TCK/SWCLK	I	Debug Clock	I	Serial Wire Clock
TDI	I	Debug Data in	-	NA
TDO/TRACESWO	O	Debug Data Out	O	Trace asynchronous Data Out
TMS/SWDIO	I	Debug Mode Select	I/O	Serial Wire Input/Output
RESET_N	I	Reset	I	Reset

Cortex-M3 Advanced System Peripherals

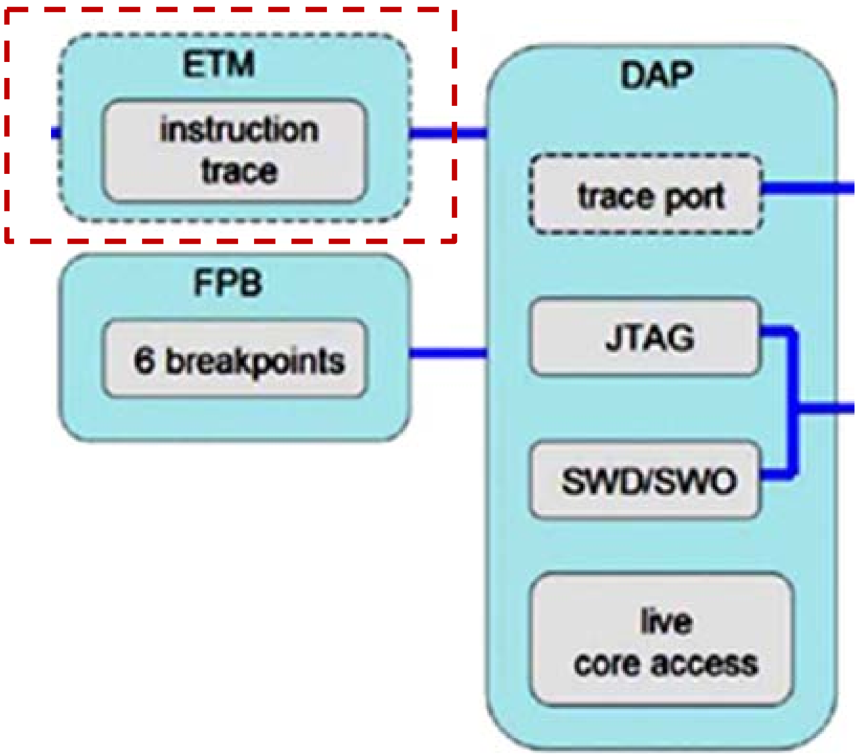
Debug Features - Flash Patch and Breakpoint (FPB)



- The FPB allows usage of :
  - Six Hardware breakpoints to generate debug events
  - Patches code and data from code space to system space. (Used in system with ROM)

Cortex-M3 Advanced System Peripherals

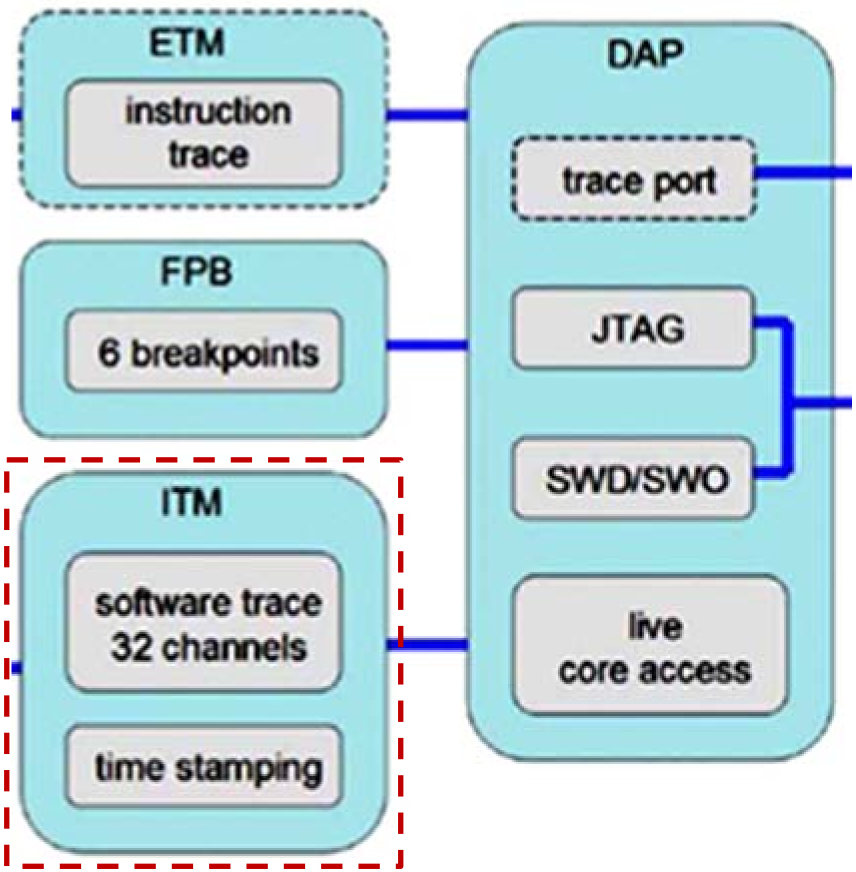
Debug Features - Embedded Trace Macrocell (ETM) - Optional



- Optional debug component that enables reconstruction of program execution.
  - Traces all 32-bit Thumb instructions as a single instruction.
  - Allows to traces instructions following an IT instruction as normal conditional instructions.
  - Supports only instruction trace.

Cortex-M3 Advanced System Peripherals

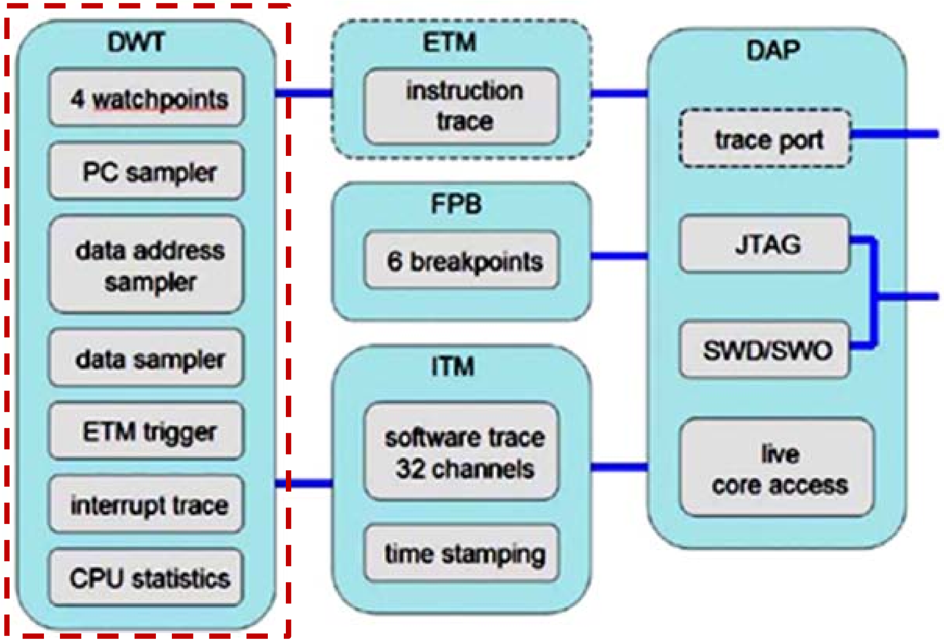
Debug Features - Instrumentation Trace Macrocell (ITM)



- Generates trace information as packets with Time stamping on 32 channels.
  - **Software trace.**
    Software can write directly to ITM stimulus registers to generate packets (printf).
  - **Hardware trace.**
    The Data Watchpoint and Trace (DWT) generates these packets and the ITM outputs them.
- Use Cortex-M3 clock or the bitclock rate of the *Serial Wire Interface* to generate the timestamp.

Cortex-M3 Advanced System Peripherals

Debug Features - Data Watchpoint and Trace (DWT)



- The DWT is a debug unit that provides:
  - 4 watchpoints for data tracing
  - System profiling (CPU statistics, PC sampler)
- Sends information directly to ETM/ITM

Cortex-M3 Advanced System Peripherals

Memory Protection Unit (MPU) - Optional

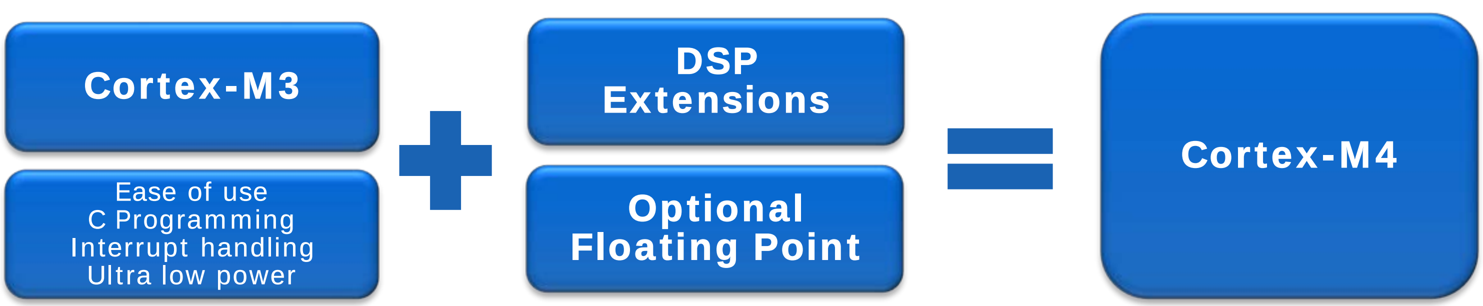
- Supports 8 memory regions (32bytes to all of the 4GB)
- Protection rules are based on the type of transaction (read, write or execute) and privilege of code performing the access
- MPU violation will cause the Memory Management Fault exception to take place
- MPU usage scenarios:
  - MPU can be set up by an operating system, allowing data used by privileged code (kernel) to be protected from untrusted user programs
  - To make memory regions read-only, to prevent accidental erasing of data
  - To isolate memory regions between different tasks in a multitasking system

Cortex-M4 / M4F Overview

Cortex-M4 / M4F Overview

Why Cortex-M4?

- Designed for applications requiring more computational performance
- Cortex-M4 frees CPU resources in case digital signal processing tasks are used (less active cycles are needed)
- Cortex M4 features:
  - A single-cycle multiply-accumulate unit (MAC)
  - Optimized single instruction multiple data (SIMD) instructions
  - Optional single precision Floating-Point Unit (FPU)



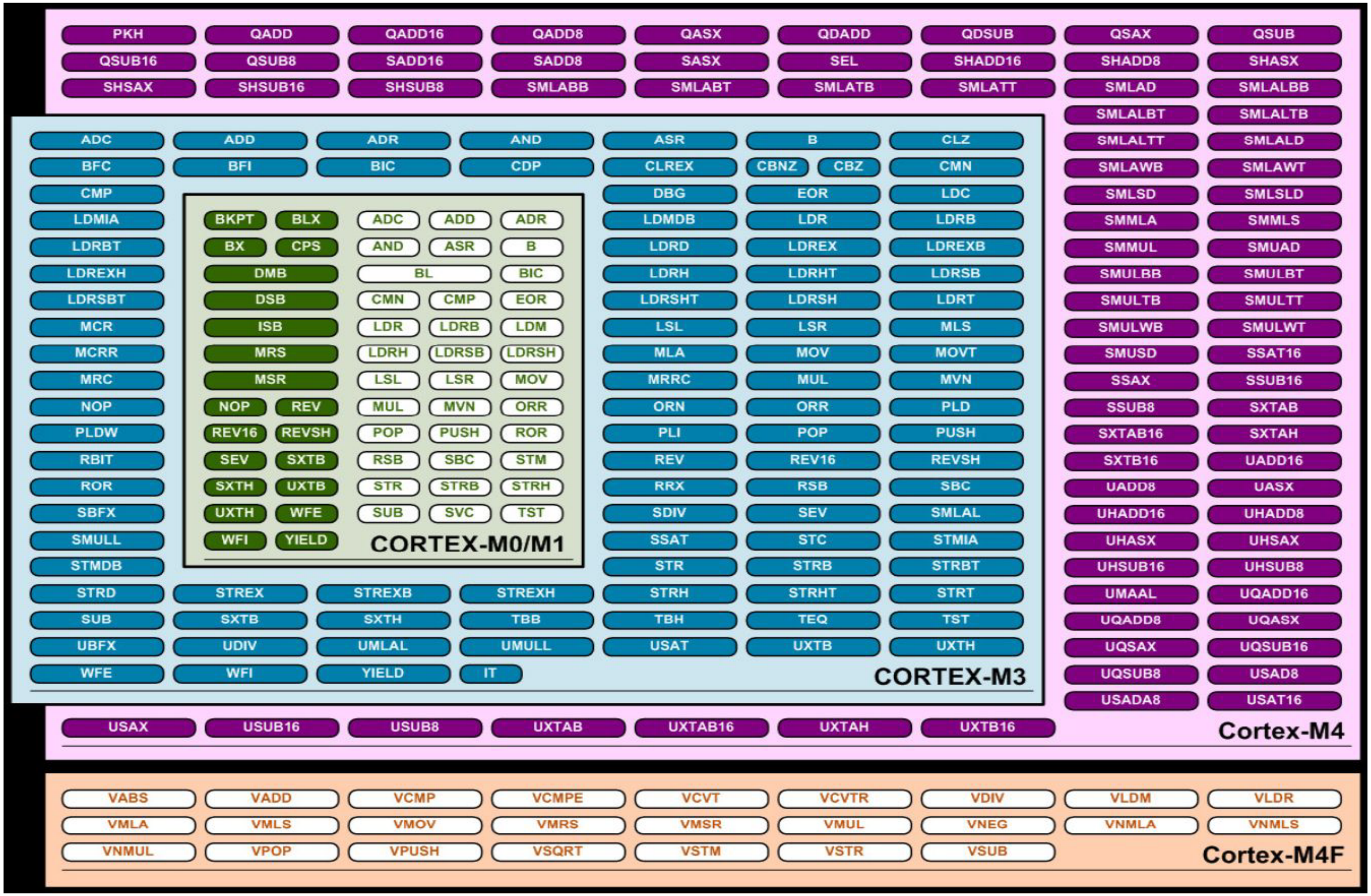
Cortex-M4 / M4F Overview

Cortex-M4 vs. Cortex-M3

	Cortex-M3	Cortex-M4
Architecture	ARMv7-M (Harvard)	ARMv7-M (Harvard)
ISA Support	Thumb / Thumb-2	Thumb / Thumb-2
DSP Extensions	NA	Single cycle 16, 32-bit MAC Single cycle dual 16-bit MAC 8, 16-bit SIMD arithmetic Hardware Divide (2-12 cycles)
Optional Floating Point Unit	NA	Single precision floating point unit IEEE 754 compliant
Pipeline	3-stage + branch speculation	3-stage + branch speculation
Interrupts	NMI + 1 to 240 interrupts	NMI + 1 to 240 interrupts
Interrupt Latency	12 cycles (6 when Tail Chaining)	12 cycles (6 when Tail Chaining)
Sleep Modes	Integrated (3)	Integrated (3)
Memory Protection	8 regions MPU	8 regions MPU
Dhrystone	1.25DMIPS/MHz	1.25DMIPS/MHz

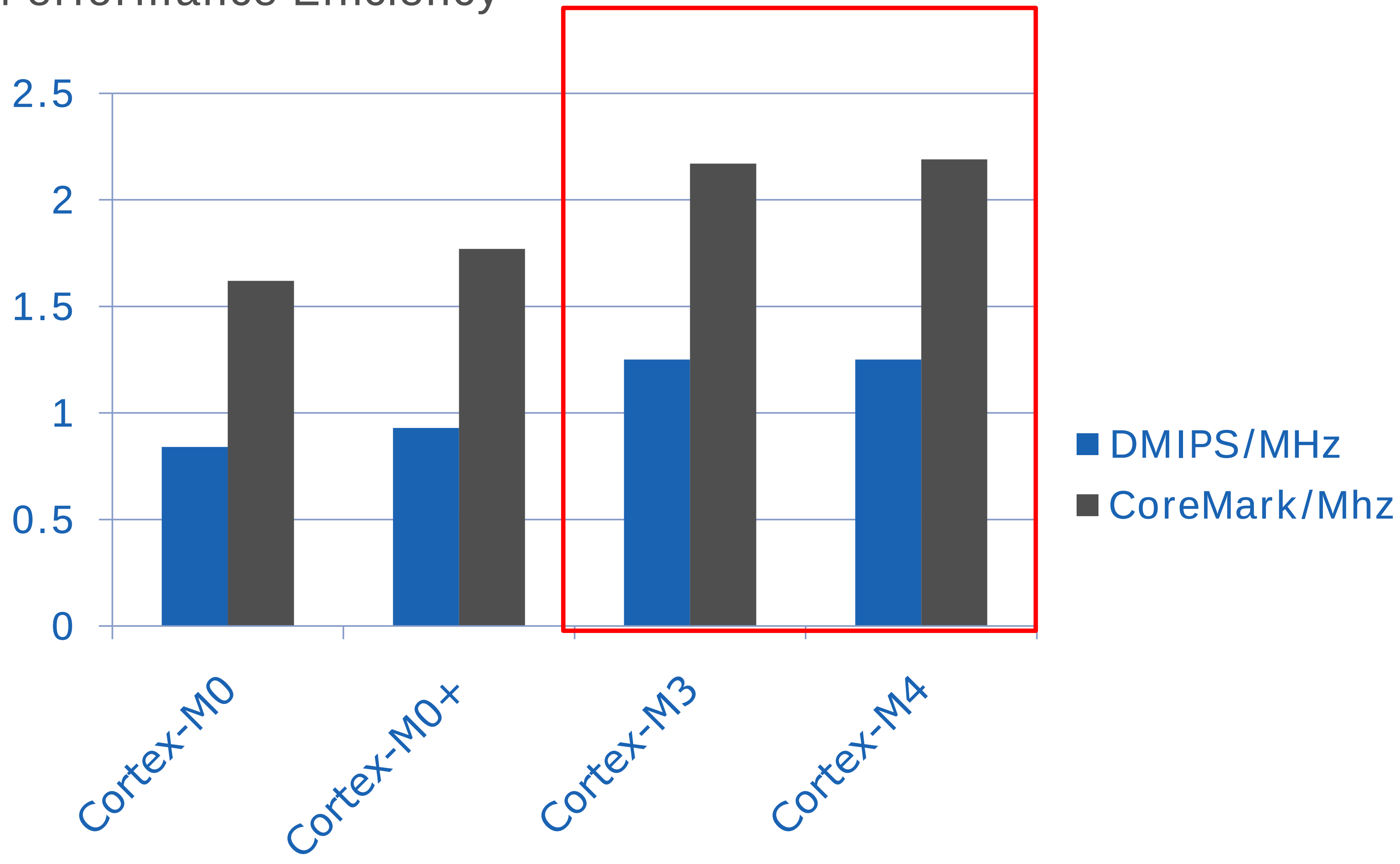
Cortex-M4 / M4F Overview

Cortex-M4 Instruction Set



Cortex-M4 / M4F Overview

Performance Efficiency



Cortex-M4 / M4F Overview

Single Cycle Multiply Accumulate Instructions

- Cortex-M4 features 32-bit hardware multiply-accumulate (MAC) unit
  - Makes digital signal processing more efficient and greatly reduces the consumption of CPU resources
  - Capable of accomplishing an operation of up to $32 \times 32 + 64 = 64$ or two operations of 16×16 in a single cycle

- Main features:
  - Wide range of multiply-accumulate instructions
  - Choice of 16 or 32 bit multiply and 32 or 64 bit accumulate
  - All instructions execute in a single cycle

Cortex-M4 / M4F Overview

MAC Instructions

OPERATION	INSTRUCTION
16 x 16 = 32	SMULBB, SMULBT, SMULTB, SMULTT
16 x 16 + 32 = 32	SMLABB, SMLABT, SMLATB, SMLATT
16 x 16 + 64 = 64	SMLALBB, SMLALBT, SMLALTB, SMLALTT
16 x 32 = 32	SMULWB, SMULWT
(16 x 32) + 32 = 32	SMLAWB, SMLAWT
(16 x 16) ± (16 x 16) = 32	SMUAD, SMUADX, SMUSD, SMUSDX
(16 x 16) ± (16 x 16) + 32 = 32	SMLAD, SMLADX, SMLSD, SMLSDX
(16 x 16) ± (16 x 16) + 64 = 64	SMLALD, SMLALDX, SMLSLD, SMLSLDX
32 x 32 = 32	MUL
32 ± (32 x 32) = 32	MLA, MLS
32 x 32 = 64	SMULL, UMULL
(32 x 32) + 64 = 64	SMLAL, UMLAL
(32 x 32) + 32 + 32 = 64	UMAAL
32 ± (32 x 32) = 32 (upper)	SMMLA, SMMLAR, SMMLS, SMMLSR
(32 x 32) = 32 (upper)	SMMUL, SMMULR

Cortex-M4 / M4F Overview

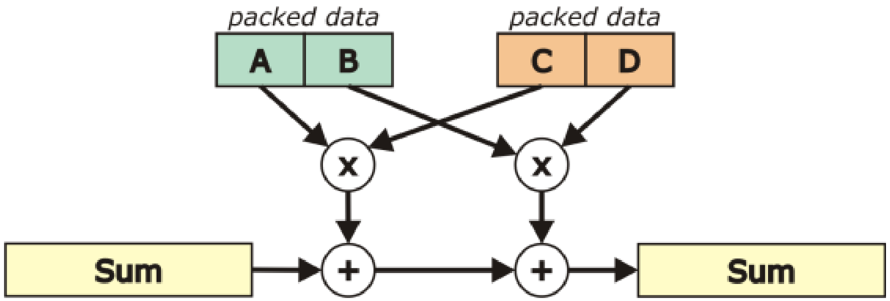
Single Instruction Multiple Data (SIMD)

- Several instructions operate on “packed” data types
  - Byte or halfword quantities packed into words
  - Allows more efficient access to packed structure types

- SIMD instructions can act on packed data:
  - Quad (4 parallel) 8-bit adds or subtracts
  - Dual (2 parallel) 16-bit adds or subtracts
  - All instructions execute in a single cycle

- SIMD extensions perform multiple operations in one cycle

Sum = Sum + (A x C) + (B x D)



SIMD techniques operate with packed data

- C Compilers won’t automatically generate SIMD instructions
  - Source code/Library must be adapted to execute them (in assembly)

Cortex-M4 / M4F Overview

Typical DSP Algorithms

- DSP operations – MAC is key operation
  - Most operations are dominated by MACs
  - These can be on 8, 16 or 32 bit operations
- FIR Filters
  - Data communications
  - Echo cancellation (adaptive versions)
  - Smoothing data
- IIR filters
  - Audio equalization
  - Motor control
- FFT
  - Audio compression
  - Spread spectrum communication
  - Noise removal

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + a_1 y[n-1] + a_2 y[n-2]$$

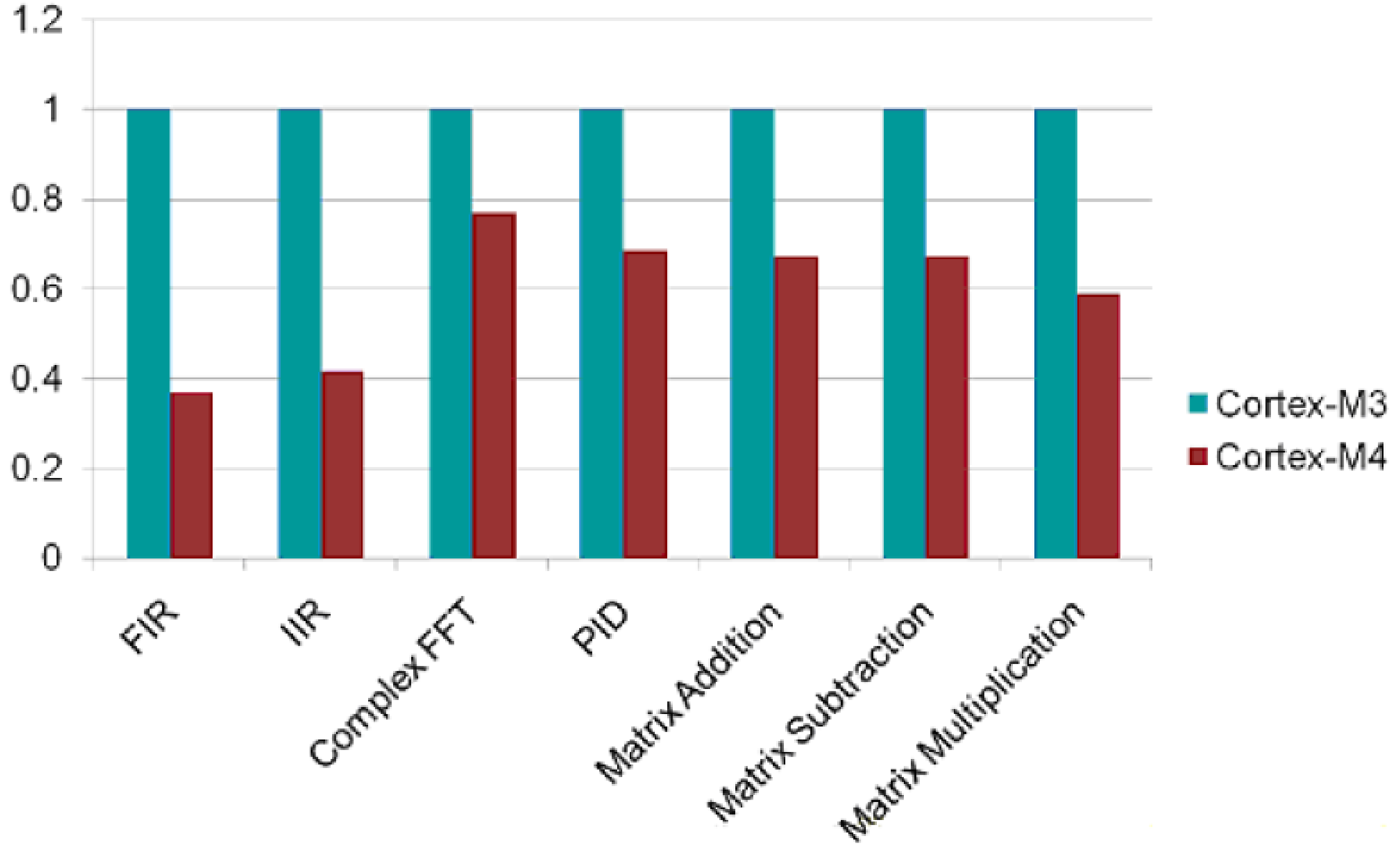
$$Y[k_1] = X[k_1] + X[k_2]e^{-j\omega}$$
$$Y[k_2] = X[k_1] - X[k_2]e^{-j\omega}$$

Atmel

Cortex-M4 / M4F Overview

Digital Signal Processing Performance

- Relative cycle count



Cortex-M4 / M4F Overview

Floating Point Unit FPU (Cortex-M4F)

- The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations.

- It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions

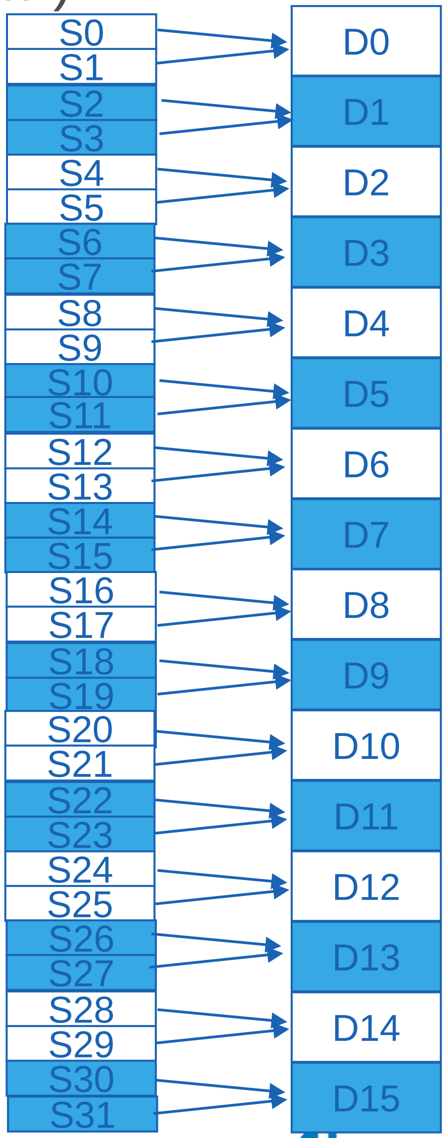
- FPU provides Floating Point computation functionality that is compliant with:
  - ANSI/IEEE Std 754-2008,
  - IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard.

Cortex-M4 / M4F Overview

FPU Registers Functional description (Cortex-M4F)

- The FPU provides an extension register file containing 32 single-precision registers. These can be viewed as:
  - Sixteen 64-bit doubleword registers, D0-D15.
  - Thirty-two 32-bit single-word registers, S0-S31.
  - A combination of registers from the above views.

- The mapping between the registers is as follows:
  - S<2n> maps to the least significant half of D<n>
  - S<2n+1> maps to the most significant half of D<n>.



Cortex-M4 / M4F Overview

Benefits of FPU (Cortex-M4F)

Float point ADD : fadd = a+b, (a= 6.1254, b=3.4587)

51 Instructions

M4 Without FPU	M4F
__aeabi_fadd: PUSH {r4-r7} EOR r2,r0,r1 LSRS r2,r2,#31 ….. ADD r1,r3,r1,LSL #1 SUBS r0,r0,r2 B 0x000003FE POP {r4-r7} BX lr	VADD.F32

1 Instruction

Float point SUBSTRACT : fsub = a-b, (a= 6.1254, b=3.4587)

68 Instructions

M4 Without FPU	M4F
__aeabi_fadd: EOR r2,r0,r1 AND r3,r2,#0x80000000 PUSH {r4-r5} ….. ADD r0,r4,r5,LSL #23 POP {r4-r5} ADD r0,r4,r3 NOP.W	VSUB.F32

1 Instruction

Cortex-M4 / M4F Overview

FPU Instructions Set (Cortex-M4F)

- Single instruction Calculation
  - Single instruction does NOT mean single cycle
  - Below few instructions of FPU with indication on cycle number.

Mnemonic	Description	Cycles
VABS.F32	Absolute value of float	1
VADD.F32	Addition floating point	1
VCMP.F32	Compare float with register or zero	1
VDIV.F32	Divide Floating point	14
VLDM.64	Load multiple doubles	1+2N, N is the number of doubles
VLDM.32	Load multiple floats	1+N, N is the number of floats
VMOV	Move immediate/float to float-register	1
VMUL.F32	Multiply float	1
VMLA.F32	Multiply then accumulate float	3

Atmel

Cortex-M4 / M4F Overview

FPU Exceptions (Cortex-M4F)

- The exception enable bits in the **FPSCR** read-as-zero, and writes are ignored.

- The processor also has six output pins
  - **FPIXC** : FPU Inexact Cumulative exception
  - **FPUFC** : FPU Underflow Cumulative Exception
  - **FPOFC** : FPU Overflow Cumulative Exception
  - **FPDZC** : FPU Division by Zero Cumulative Exception
  - **FPIDC** : FPU Input Denormal Cumulative Exception
  - **FPIOC** : FPU Invalid Operation Cumulative exception.

  that each reflect the status of one of the cumulative exception flags.

- In the SAM4X, all exception interrupt are connected on the same Instance ID and on the NVIC Interrupt.

Appendix

Appendix

Cortex-M4 Implementation Options

Cortex-M Core	MPU	FPU	ETM	ITM	JTAG / SWD	Bit-banding	SysTick Timer
SAM4S	Yes	No	No	Yes	Yes / Yes	Yes	Yes
SAM4N	Yes	No	No	Yes	Yes / Yes	Yes	Yes
SAM4E	Yes	Yes	No	Yes	Yes / Yes	Yes	Yes
SAM4L	Yes	No	No	Yes	Yes / Yes	No	Yes
SAMG51	Yes	Yes	No	Yes	Yes / Yes	Yes	Yes
SAMG53	Yes	Yes	No	Yes	Yes / Yes	Yes	Yes

Enabling Unlimited Possibilities®