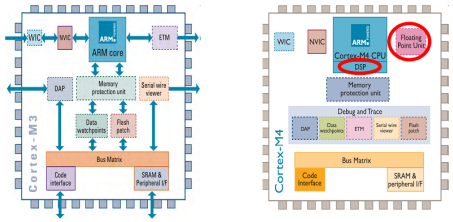
ELE408 Section 1. Cont'd

Introduction to ARM Cortex-M4 Architecture

Cortex M4 Highly efficient 1

Cortex™-M3 → Cortex™-M4

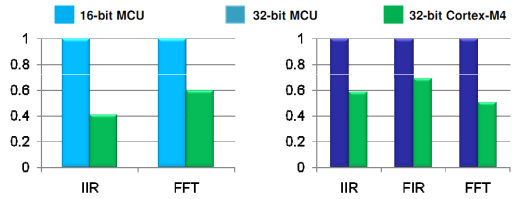


Key Cortex™-M4 enhancements over Cortex-M3:

DSP instructions, SIMD instructions, optional floating point unit

Cortex M4 Highly efficient 2

The Cortex™-M4 is ~2X more efficient on most DSP tasks than leading 16 and 32 bit MCU devices with DSP extensions

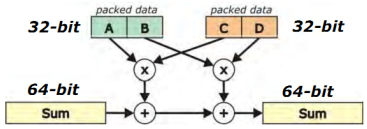


Cycle counts on DSP tasks compared, smaller is better

SIMD(Single instruction Multiple data) Operations

SIMD extensions perform multiple operations in one cycle

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



SIMD techniques operate with packed data

Cortex™-M4 Single Cycle MAC

OPERATION	INSTRUCTIONS
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

All the above operations are single cycle on the Cortex-M4 processor

Cortex™-M4 SIMD arithmetic

Prefix / Instr	S Signed	Q Signed Saturating	SH Signed Halving	U Unsigned	UQ Unsigned Saturating	UH Unsigned Halving
ADD8	SADD8	QADD8	SHADD8	USADD8	UQADD8	UHADD8
SUB8	SSUB8	QSUB8	SHSUB8	USUB8	UQSUB8	UHSUB8
ADD16	SADD16	QADD16	SHADD16	UADD16	UQADD16	UHADD16
SUB16	SSUB16	QSUB16	SHSUB16	USUB16	UQSUB16	UHSUB16
ASX	SASX	QASX	SHASX	UASX	UQASX	UHASX
SAX	SSAX	QSAX	SHSAX	USAX	UQSAX	UHSAX
USAD8	Unsigned Sum of Absolute Difference (8 bits)					
USADA8	Unsigned Sum of Absolute Difference and Accumulate (8 bits)					

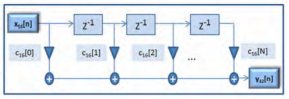
ASX
1. Exchanges halfwords of the second operand register
2. Adds top halfwords and subtracts bottom halfwords

SAX
1. Exchanges halfwords of the second operand register
2. Subtracts top halfwords and adds bottom halfwords

DSP Example: Cortex™-M4 FIR

$$y_{32}[n] = \sum_{i=0}^{N} x_{16}[n-i] * c_{16}[i]$$

y_{32} – filter ouput (32 bit); N - filter order
x_{16} –16 bit input data; c_{16} –16 bit filter coeff



Cortex-M3 Code Segment:

```
FIR_LOOP:
LDR    R2,[R0],#4    ;(2) Load input x16
LDR    R3,[R1],#4    ;(2) Load coeff c16
SXTH   R4, R2        ;(1) Extract x16[n-i]
ASR    R2, R2,#16    ;(1) Extract x16[n-i-1]
SXTH   R5, R3        ;(1) Extract c16[i]
ASR    R3, R3,#16    ;(1) Extract c16[i+1]
MLA    R6, R4, R5    ;(2) y32 += x16[n-i]*c16[i]
MLA    R6, R2, R3    ;(2) y32 += x16[n-i-1]*c16[i+1]
SUBS   R7, R7, #2    ;(1) loop count -= 2
BNE    FIR_LOOP      ;(2)

Note:
1. In these examples, FIR_LOOP is unrolled by 2
2. This example assumes number of taps is even.
```

Cortex-M4 Code Segment:

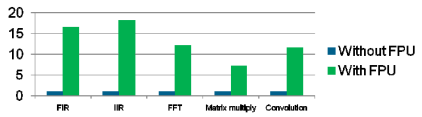
```
FIR_LOOP:
LDR    R2,[R0],#4 ;(1) Load input x16[n-1],x16[n-i-1]
LDR    R3,[R1],#4 ;(2) Load coeff c16[i], c16[i+1]
SUBS   R5, R5, #2 ;(1) loop count -= 2
SMLAD  R4, R2, R3 ;(1) y32 +=x16[n-1,n-i-1]*c16[i,i+1]
BNE    FIR_LOOP   ;(2)
```

Processor	Kernel cycles	Total Cycles	Number of Instructions	Register usage
Cortex-M3	8	15	10	7
Cortex-M4	1	7	5	5
Advantage	8x	~2.2x	2x	1.4x

Example (non-binding) from Ittiam Systems, a leading provider of signal processing software on ARM platforms

Cortex™-M4 Single Precision Floating Point

- Floating point benefits
  - Extended range, Highly accurate measurements
- Cortex™-M4 FPU
  - IEEE 754 standard compliant
  - Single-precision floating point math
- Graph below shows Cortex™-M4 single precision floating point algorithm performance normalized to "Without FPU"

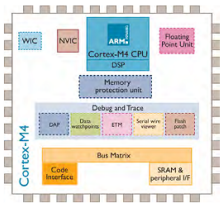


Cortex™-M4 Floating Point Unit

- Single-precision floating point math
  - Add, subtract, multiply, divide, MAC and square root
  - Fused MAC

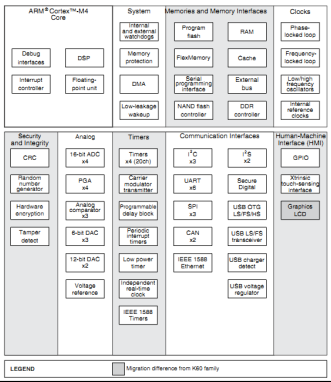
OPERATION	CYCLE COUNT
Add/Subtract	1
Divide	14
Multiply	1
Multiply Accumulate (MAC)	3
Fused MAC	3
Square Root	14

Kinetis Cortex-M4 Core Enhancements

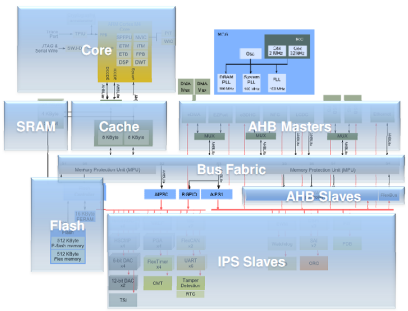


- Up to 32-channel DMA
  - Reduced CPU loading
  - Faster system throughput
- Cross bar switch
  - Concurrent multi-master bus accesses
- Up to 8KB of instruction and 8KB of data cache
  - Optimized bus bandwidth and flash execution performance
- Independent flash banks
  - Concurrent code execution and firmware updating
  - No performance degradation or complex coding routines

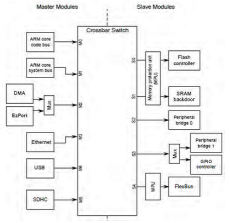
K70 Block Diagram



Kinetis Bus Structure



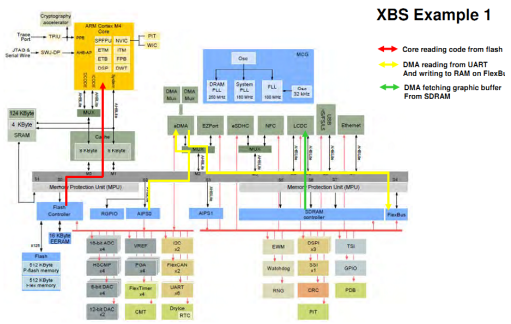
Cross Bar Switch Configuration



- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
  - Slave arbitration attributes configured on a slave by slave basis
- 32 -bit wide and supports byte, 2 byte, 4 byte, and 16 byte burst transfers
- Operates at a 1-to-1 clock frequency with the bus masters
- Low-power park mode support

UNIVERSITY of Rhode Island

XBS Example



UNIVERSITY of Rhode Island