Developing Advanced Signal Processing Software on the Cortex™-M4 Processor

Ian Johnson

Product Manager

ARM

MCUs address broad markets

Motor control
- ❑ Field oriented control
- ❑ Stepper motors
- ❑ BLDC motors etc

Connectivity
- ❑ Bluetooth
- ❑ Zigbee
- ❑ Ethernet

Audio
- ❑ MP3 players
- ❑ Wireless headsets
- ❑ Virtual surround

Smart Metering
- ❑ Gas and water
- ❑ Electricity
- ❑ Connectivity

Power management
- ❑ UPS
- ❑ Lighting systems
- ❑ AC/DC converters

Human interfaces
- ❑ EPOS terminals
- ❑ Image processing
- ❑ Audio interfaces

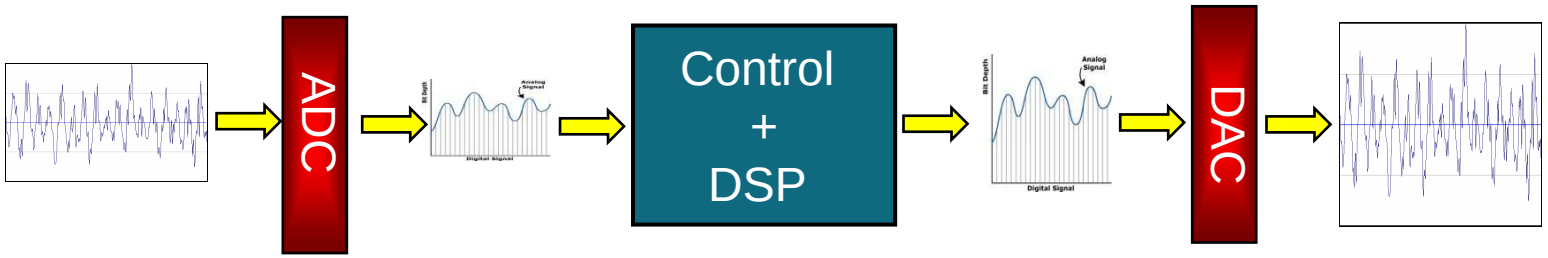
Industrial Applications
- ❑ Appliances
- ❑ Lighting
- ❑ Motion control

Medical instrumentation
- ❑ Blood pressure meters
- ❑ Glucose meters
- ❑ Defibrillators

Automotive
- ❑ ABS systems
- ❑ Chassis control
- ❑ Airbag systems

General signal processing

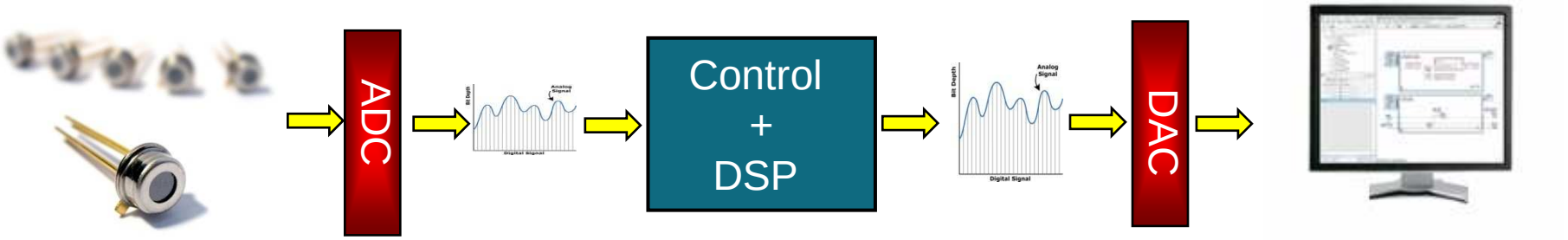
- Signal processing is almost exclusively in the digital domain
  - Real world analog signals are converted to a digital representation
  - Mathematical operations of many varieties performed on the signals
  - Digital information converted back to real world analog signal



- Digital Signal Processing heavily leans on MAC (Multiply Accumulate) operations inside large loops
- Yet most algorithms have a lot of control code as well
- Most embedded markets demanding an efficient blend of traditional RISC processing features with DSP capabilities

Focus of talk-programming

- Managing the data flow is one challenge of the system



- Input and output data can represent different signals
  - Motor position, Audio, Video, RF signal (GPS, etc.), Sensors Etc.
- Data is arriving in real-time at a fixed sample rate
  - Audio = 44.1 kHz, Video = 100 MHz, RF = MHz to GHz
- Optimal peripherals and data converters also required
- This talk purely focuses on the software programming

Characteristics of MCUs

- Easy to use architecture – programming in C
- Efficient ISA for best performance and code size
- Ultra low power architecture – sleep modes etc
- Excellent interrupt control and latency
- Deterministic operation in multitude of applications
- Low cost debug and trace
- Bit manipulation techniques
- Memory protection for separation of processes
- Excellent software ecosystem
  - Compilers, debuggers, RTOS etc
- Large number of variants with different peripheral mix

Characteristics of DSPs

- Harvard architecture
- High performance MAC
- Saturating math
- SIMD instructions for parallel computation
- Barrel shifters
- Floating point hardware
- Circular and bit-reversed addressing
- Zero overhead loops
- Load and store operations in parallel with math operations

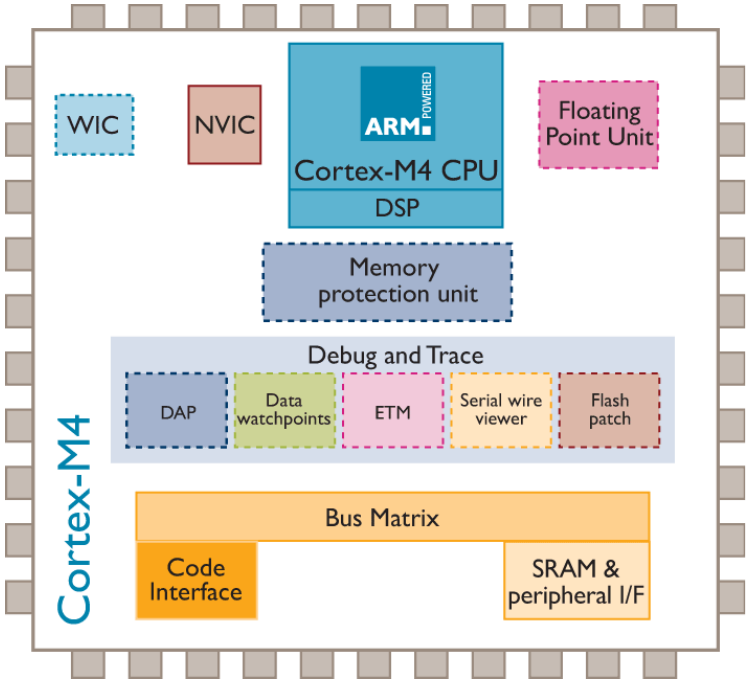
Digital signal control (DSC) - blend

Most features in DSCs today

- ★ Harvard architecture
- ★ High performance MAC
- ★ Saturating math
- ★ SIMD instructions for parallel computation
- ★ Barrel shifters
- ★ Floating point hardware
- – Circular and bit-reversed addressing
- – Zero overhead loops
- – Load store operations in parallel with math operations

Efficient CPU for DSCs

- **Cortex-M4 processor**
  - Thumb®-2 Technology
  - DSP and SIMD instructions
  - Single cycle MAC (Up to 32 x 32 + 64 -> 64)
  - Optional decoupled single precision FPU
  - Integrated configurable NVIC
- **Microarchitecture**
  - 3-stage pipeline with branch speculation
  - 3x AMBA® AHB-Lite bus Interfaces
- **Configurable for ultra low power**
  - Deep Sleep Mode, Wakeup Interrupt Controller (WIC)
  - Power down features for Floating Point Unit
- **Flexible configurations for wider applicability**
  - Configurable Interrupt Controller (1-240 Interrupts and Priorities)
  - Optional Memory Protection Unit (MPU)
  - Optional Debug & Trace



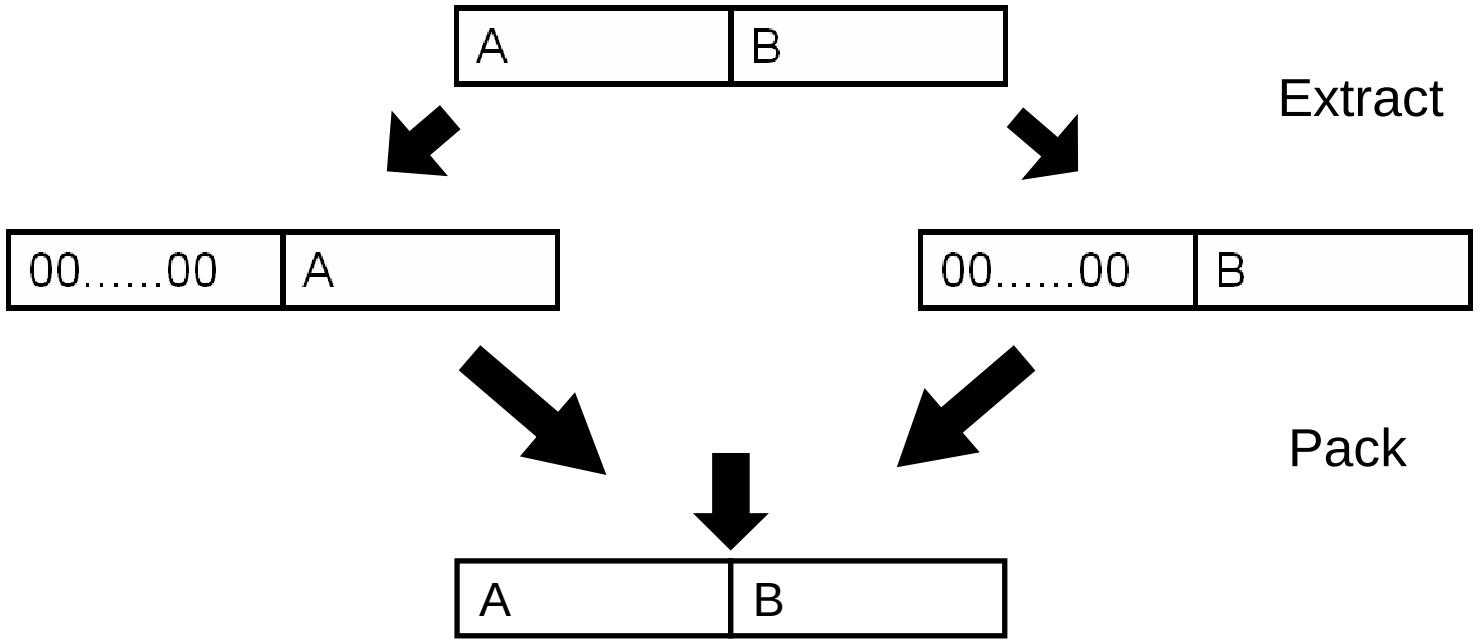
Powerful MAC instructions

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

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

Packed data types

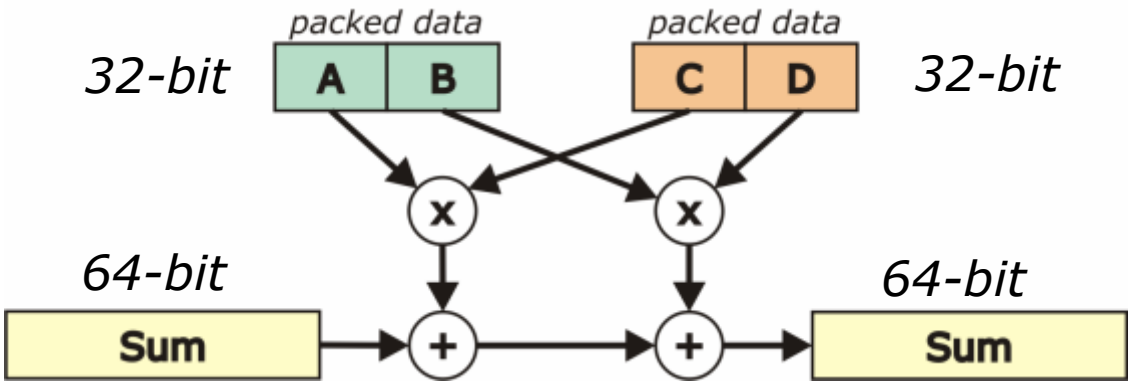
- Several instructions operate on “packed” data types
  - Byte or halfword quantities packed into words
  - Allows more efficient access to packed structure types
  - SIMD instructions can act on packed data
  - Instructions to extract and pack data



SIMD operations

SIMD extensions perform multiple operations in one cycle

$$Sum = Sum + (A \times C) + (B \times D)$$



SIMD techniques operate with packed data

SIMD arithmetic operations

Prefix / Instruction	S Signed	Q Signed Saturating	SH Signed Halving	U Unsigned	UQ Unsigned Saturating	UH Unsigned Halving
ADD8	SADD8	QADD8	SHADD8	UADD8	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

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

Floating point hardware

- IEEE 754 standard compliance

- Single-precision floating point math key to some algorithms
  - Add, subtract, multiply, divide, MAC and square root
  - Fused MAC – provides higher precision

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

Main DSP operations

- Finite impulse response (FIR) filters
  - Data communications
  - Echo cancellation (adaptive versions)
  - Smoothing data
- Infinite impulse response (IIR) filters
  - Audio equalization
  - Motor control
- Fast Fourier transforms (FFT)
  - Audio compression
  - Spread spectrum communication
  - Noise removal

Mathematical details

- FIR Filter

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

- IIR or recursive filter

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

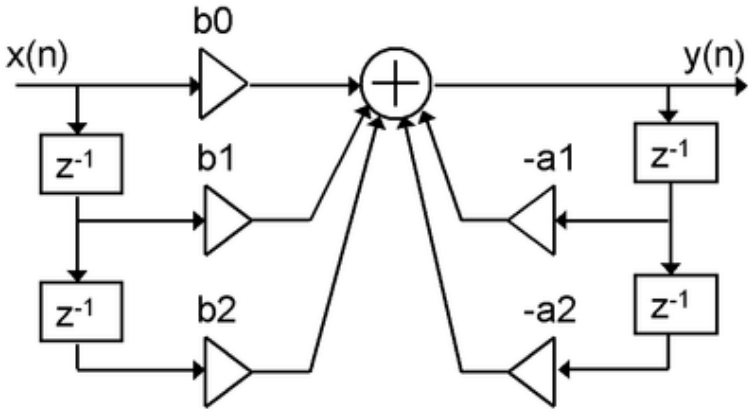
- FFT Butterfly (radix-2)

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

Most operations are dominated by MACs
These can be on 8, 16 or 32 bit operations

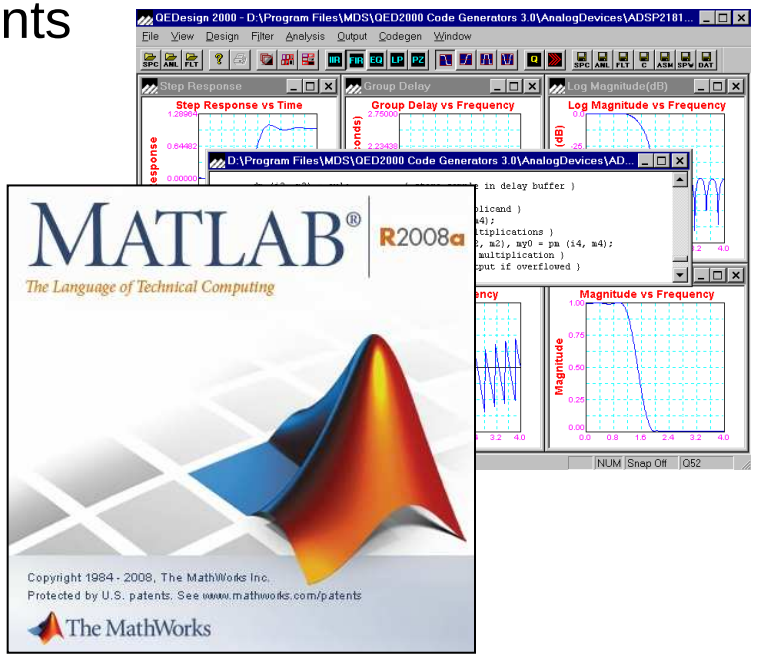
Computing Coefficients

- Variables in a DSP algorithm can be classified as “coefficients” or “state”
  - Coefficients – parameters that determine the response of the filter (e.g., lowpass, highpass, bandpass, etc.)
  - State – intermediate variables that update based on the input signal



- The Direct Form 1 Biquad has 5 coefficients and 4 state variables

- Coefficients may be computed in a number of different ways
  - Simple design equations running on the MCU
  - External tools such as MATLAB or QED Filter Design



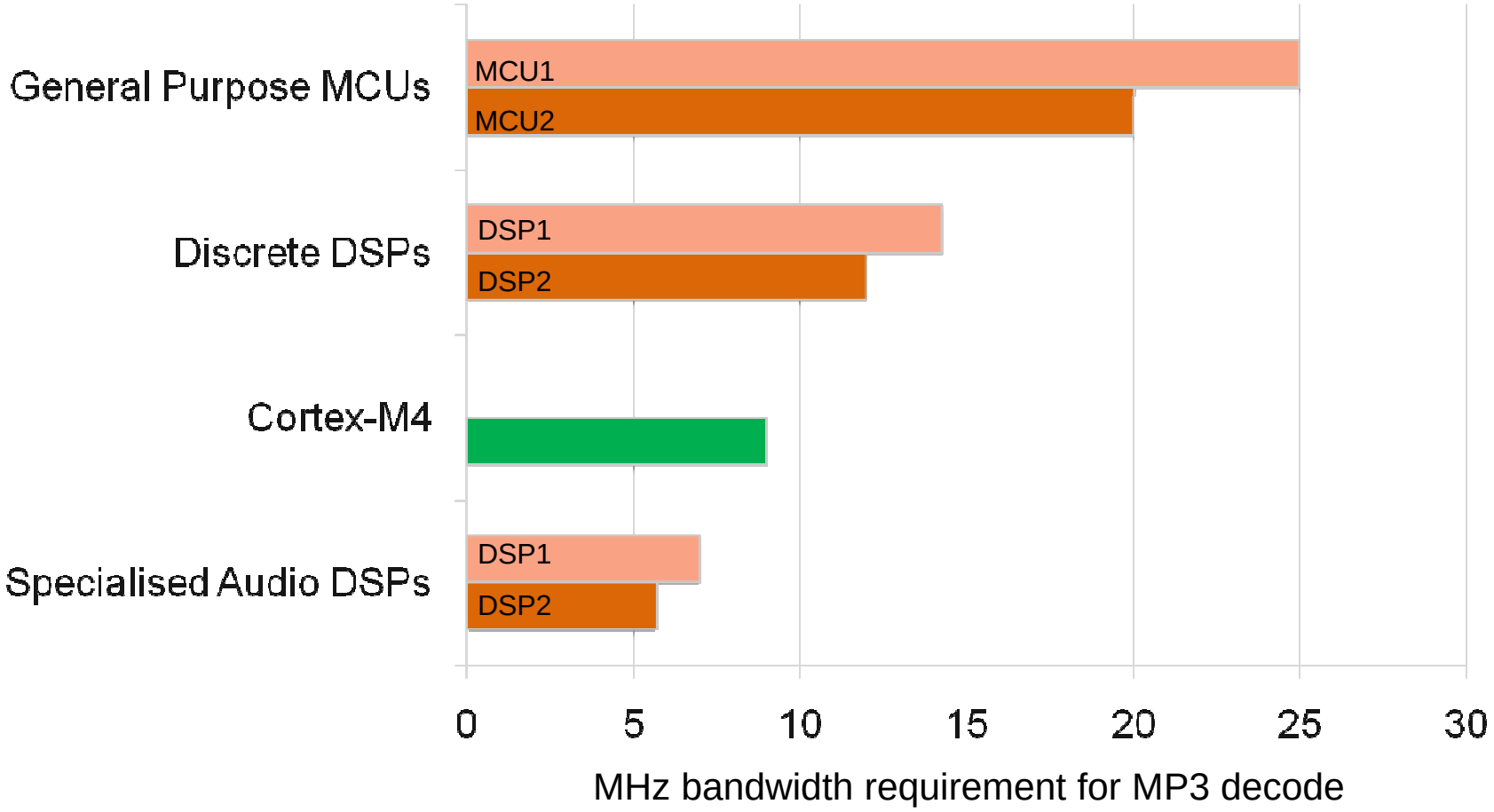
IIR – single cycle MAC benefit

	Cortex-M3 cycle count	Cortex-M4 cycle count
`xN = *x++;`	2	2
`yN = xN * b0;`	3-7	1
`yN += xNm1 * b1;`	3-7	1
`yN += xNm2 * b2;`	3-7	1
`yN -= yNm1 * a1;`	3-7	1
`yN -= yNm2 * a2;`	3-7	1
`*y++ = yN;`	2	2
`xNm2 = xNm1;`	1	1
`xNm1 = xN;`	1	1
`yNm2 = yNm1;`	1	1
`yNm1 = yN;`	1	1
`Decrement loop counter`	1	1
`Branch`	2	2

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

- Only looking at the inner loop, making these assumptions
  - Function operates on a block of samples
  - Coefficients b0, b1, b2, a1, and a2 are in registers
  - Previous states, x[n-1], x[n-2], y[n-1], and y[n-2] are in registers
- Inner loop on Cortex-M3 takes 27-47 cycles per sample
- Inner loop on Cortex-M4 takes 16 cycles per sample

Example – MP3 playback



Keeping programming simple

- Complex hardware needs to be easy to program
  - Assembly optimization is hard work
- Interfacing with easy to use tools is very important
- All Cortex-M processors can be fully programmed in C
  - Quicker learning curve for faster application development
  - Easy to maintain, reuse and port
- Reusing code essential for faster delivery
  - Cortex-M processors fully upwards compatible
  - Programming standards essential for code reuse

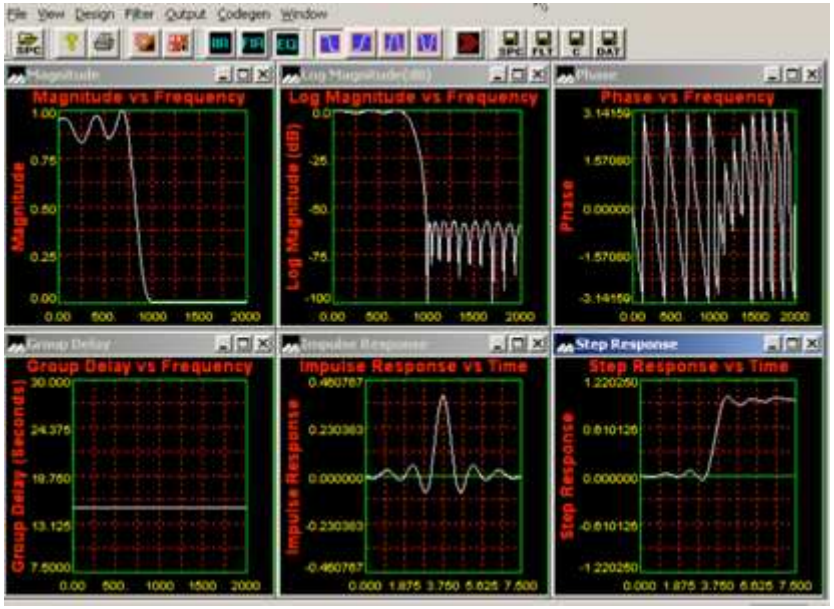
Cortex Microcontroller Standard (CMSIS)

- Cortex Microcontroller Software Interface Standard
  - Abstraction layer for all Cortex-M processor based devices
  - Developed in conjunction with silicon, tools and middleware partners

- Benefits to the embedded developer
  - Consistent software interfaces for silicon and middleware vendors
  - Simplifies re-use across Cortex-M processor-based devices
  - Reduces software development cost and time-to-market
  - Reduces learning curve for new Cortex microcontroller developers

Cortex-M4 CMSIS extensions

- Cortex-M4 support available today in ARM Compiler and Keil-MDK
  - C Compiler intrinsic functions for Cortex-M4 extended Instructions
  - Optimized Floating Point Library using FPU CPU Instructions
  - Complete µVision Debugger support; including Instruction Set Simulation
- CMSIS - Expanded with Cortex-M4 Features (Intrinsic Functions)
  - Every CMSIS compliant C Compiler supports Cortex-M4 extensions
- Optimized Library using CMSIS
  - Designed to make DSP programs easy to develop for MCU users
  - **General Functions**
    math, trigonometric, control functions (building blocks)
  - **Digital Filter Algorithms**
    for filter design utilities and DSP toolkits (MATLAB, LabVIEW, etc.)



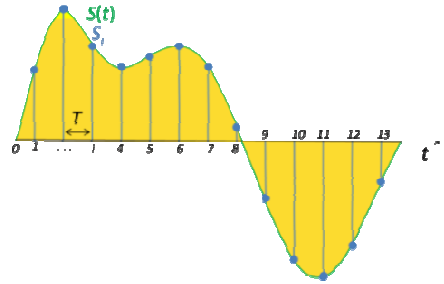
CMSIS-DSP Library

- C Source Code, optimized for Cortex-M4
- For CMSIS compliant C Compilers (ARM/Keil, IAR, GCC)

- Basic Math Functions
  - Vector Multiplication
  - Vector Subtraction
  - Vector Addition
  - Vector Scale
  - Vector Shift
  - Vector Offset
  - Vector Negate
  - Vector Absolute
  - Vector Dot Product
- Fast Math Functions
  - Cosine
  - Sine
  - Square root of number
- Complex Math Functions
  - Complex conjugate
  - Complex dot product
  - Complex magnitude
  - Complex magnitude squared
  - Complex by complex multiplication
  - Complex by real multiplication
- Filters
  - Biquad Cascade IIR Filters Using Direct form I Structure
  - Finite Impulse Response (FIR) Filters
  - Convolution
  - Partial Convolution
  - Correlation
  - Finite Impulse Response (FIR) Decimation
  - Finite Impulse Response (FIR) Lattice Filters
  - Infinite Impulse Response (IIR) Lattice Filters
  - Biquad Cascade IIR 32x64 filter using Direct form I structure
  - Biquad Cascade IIR Filters Using a Direct form II Transposed Structure
  - Finite Impulse Response (FIR) Sparse Filters
  - Finite Impulse Response (FIR) Interpolation
  - Least Mean Square FIR Filter
  - Least Mean Square Normalized FIR Filter

- Matrix Functions
  - Matrix Addition
  - Matrix Initialization
  - Matrix Scale
  - Matrix Subtraction
  - Matrix Multiplication
  - Matrix Inverse
  - Matrix Transpose
- Transforms
  - Complex FFT Functions
  - Real FFT Functions
  - DCT Type IV Function
- Controller Functions
  - SineCosine
  - PID Motor Control
  - Vector park transform
  - Vector Inversepark transform
  - Vector Clarke transform
  - Vector Inverse Clarke transform
- Statistical Functions
  - Power
  - Root mean square (RMS)
  - Standard deviation
  - Variance
  - Maximum
  - Minimum
  - Mean
- Support Functions
  - Vector Copy
  - Vector Fill
  - Convert 8-bit Interger value
  - Convert 16-bit Interger value
  - Convert 32-bit Interger value
  - Convert 32-bit floating point value
- Interpolator Functions
  - Linear Interpolate Function
  - Bilinear Interpolate Function

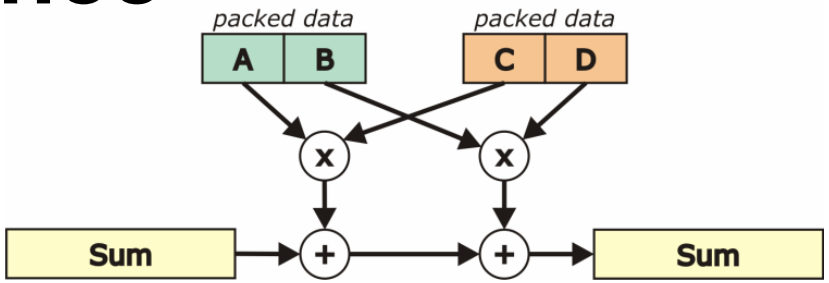
CMSIS-DSP Library



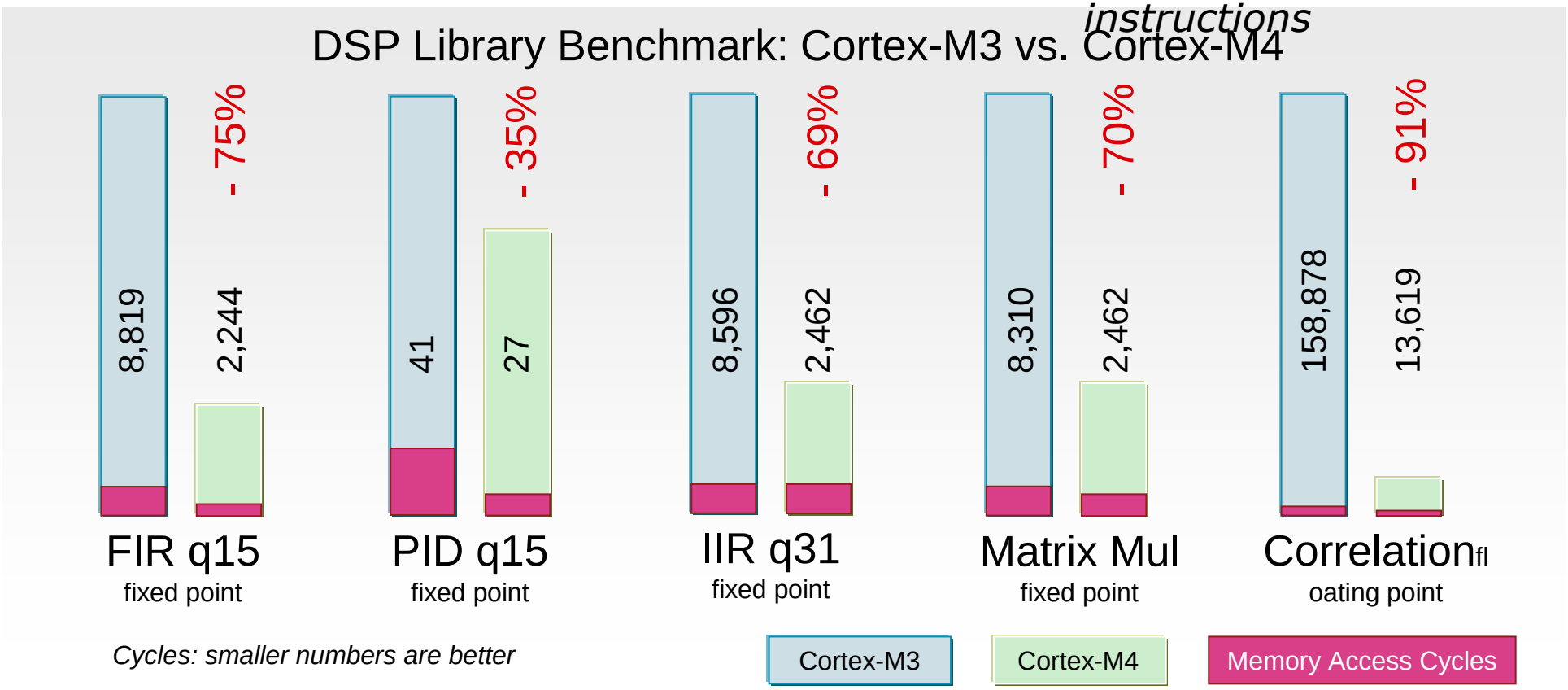
- Enabling product that is free-of-charge
  - For silicon vendors, tool partners, and end-user's
- Extends application range of Cortex-M3/M4 MCUs to high-performance, low power signal processing
  - Can be used on existing Cortex-M3 microcontrollers
- Generic, vendor independent, designed to be a standard
  - Works with any good RTOS message passing system (i.e. Keil RTX)
  - But can be used stand-alone
  - For CMSIS compliant C Compilers (ARM/Keil, IAR, GCC, etc.)

DSP Library Performance

- Cortex-M4 SIMD + FPU
  - Fix point: ~2x faster
  - Floating point: ~10x faster



On Cortex-M4: uses SIMD & FPU instructions



Conclusion

- Signal processing needs in low cost MCUs are increasing
  - Motor control, industrial automation, automotive, audio etc
- MCUs and DSCs with signal processing features coming
  - Single cycle MAC, SIMD arithmetic, saturation, floating point h/w etc
- High performance signal processing with Cortex-M4 DSCs
  - MP3 decode within 10MHz requirement possible on low cost devices
- Signal processing is not as hard as you might think !
  - Can be programmed fully in C
  - Software standards like CMSIS make programming easier