



Rapid “Cross” Platform DSP with the ARM Cortex M4/M7

Eli Hughes

ehughes@wavenumber.net



#ESCsv

Presentation Scope

- In the “trenches” perspective.
- DSP Evolution, history and application rationale
- Quick introduction to the Cortex M4 & M7
- Vendors & tool-chains
- Look at some “interesting” instructions
- CMSIS-DSP library
- Performance stats
- Some fun demos

Where I got Started with ARM Cortex M4....

Active Pickguard



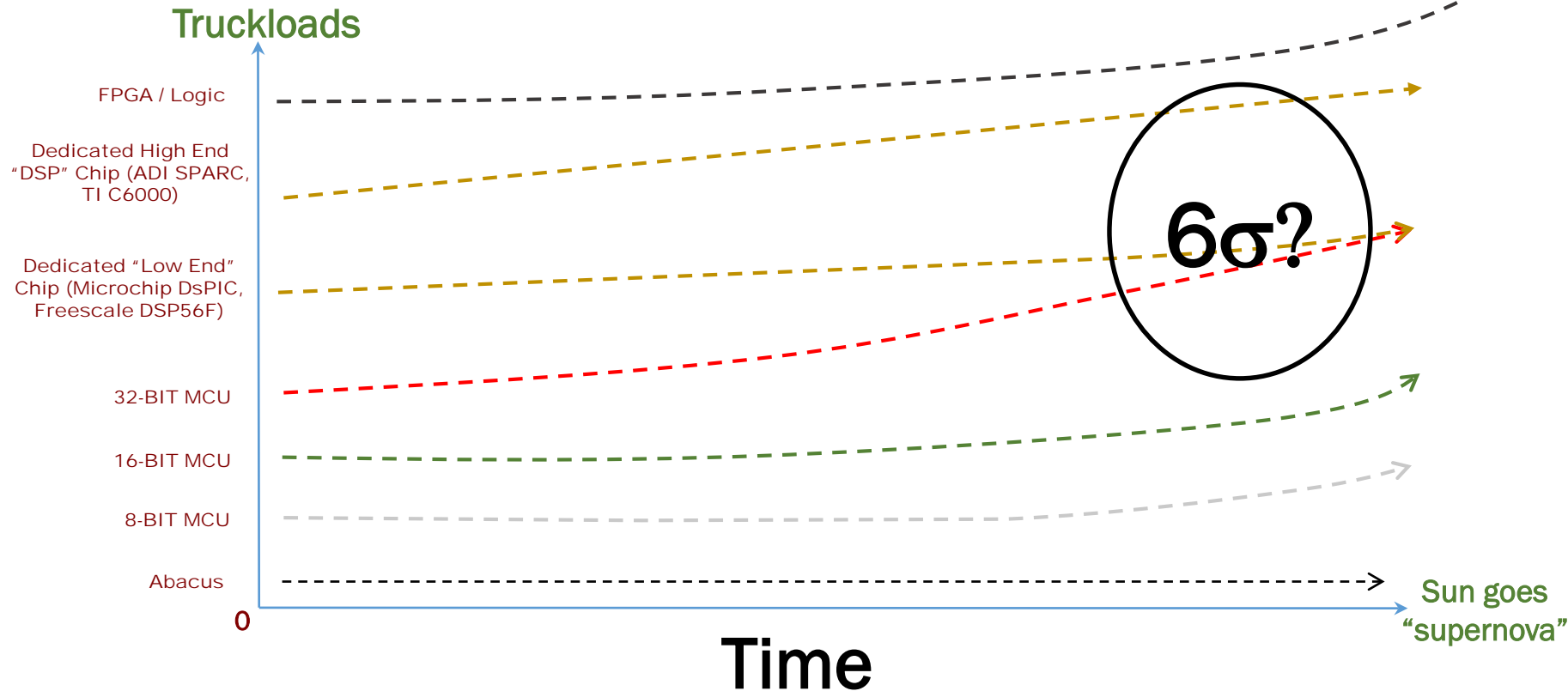
- <http://www.2pl-1.com/active-pickguard/>

Digital Signal Processing

- DSP is an application of discrete math to digital systems
- One doesn't necessarily need a special chip to implement a DSP algorithm
- I can implement DSP on an 8051.... Or an abacus.... It just might not be as fast or as sophisticated as I would like it to be
- I can also choose to implement a 2 tap FIR running at 1Hz sample rate with 4-bit data on a \$5000 FPGA

The Evolution of DSP Hardware

DSP'ometer



The 6 σ

- DSP has traditionally been relegated to specialized processors and architectures.
- There exists a large number of DSP applications that don't require an expensive FPGA or high-end specialized chip.
- Many applications also need the standard IO (USB, Ethernet, etc.)
- The M4/M7 can be useful for many applications. Keep in mind it is still general purpose RISC. Lots of register manipulation!

Applications

- IIR Filter on a dynamic sensor / accelerometer
- A matched filter to detect an event
- Some sort of frequency domain analysis
- Real time control systems, low/mid range audio....
- Can't do it with 8051.... need USB, Ethernet, etc.
Maybe need moderate sample rates.

Application Example – “Play Node”

- Consider a “box” to implement networked sounds, sensors and lighting.
- Boxes are distributed over a large area (25m x 25m).
- A master controller communicates with the nodes to form a cool outdoor “games space”



Application Example – “Play Node”

- Requirement #1 - The worlds most rugged and inexpensive input device.... “The Acoustic Button”
(Need 4 Channels)



Application Example – “Play Node”

- Requirement #2 - Sound Playback – 44.1KHz – 4 Channels .wav or .mp3
- Requirement #3 - Embedded FAT File system on USB Thumb Drive.... USB Mass Storage Class Host
- Requirement #4 - Lighting Control for WS2812 or APA102 “smart” LEDs. Control effects on chains of 100’s of pixels at 30FPS
- Requirement #5 – CAN Bus communications

Application Example – “Play Node”

- Requirement #6 – USB Device input - Mass storage Class to configuration file system
- Requirement #7 – Misc Inputs (4-20mA)
- Requirement #8 – Instant Boot & Real Time.... No Linux
- Requirement #9 - Can't burn watts of Power

Application Example – “Play Node”

*Lots of requirements.... Some light DSP + Audio +
USB + CAN + Standard I/O*

Cortex M4 fits well..... (This case used the NXP
LPC43x Series – Dual Core M0/M4)

ARM Cortex?

- Who here is familiar with ARM Cortex?
- 3 Families
 - A *Applications Processors* - High-end cell phone/tablet chips
 - R *Real Time* - Lock-step/real-time automotive, high-end control
 - M *Microcontroller* - Focused on lower cost 32-bit. I.E. A good reason to move from 8/16-Bit.

ARM Cortex?

- Why does Cortex M Exist?

(My opinion) To fix the architectural problems in the ARM7TDMI

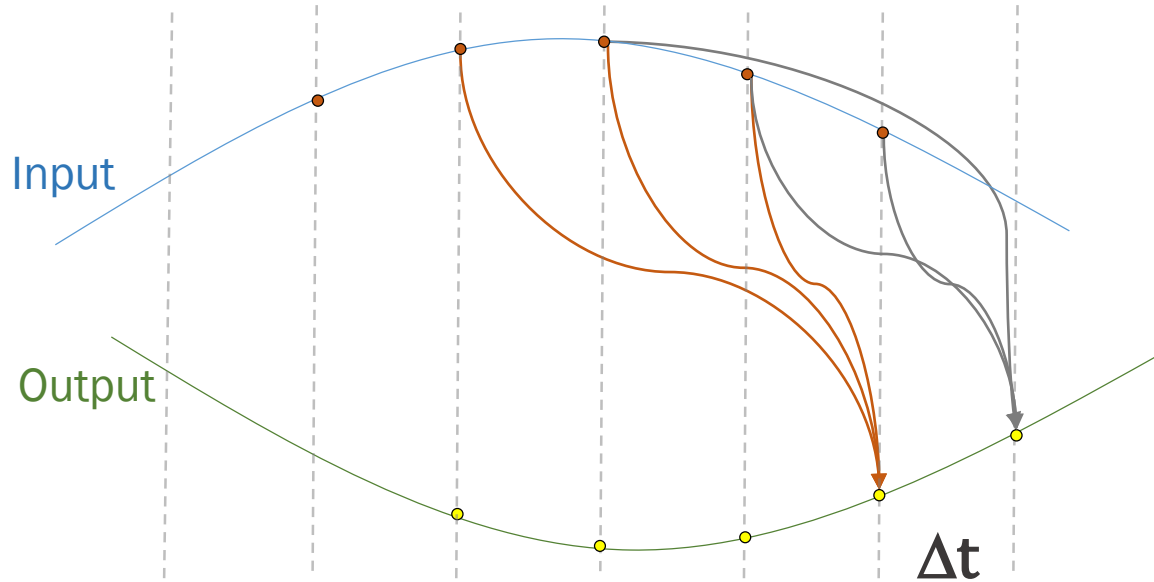
- Who here has use ARM7TDMI?

→ A better interrupt system, common internal elements, bit-banding, thought-out API to CPU

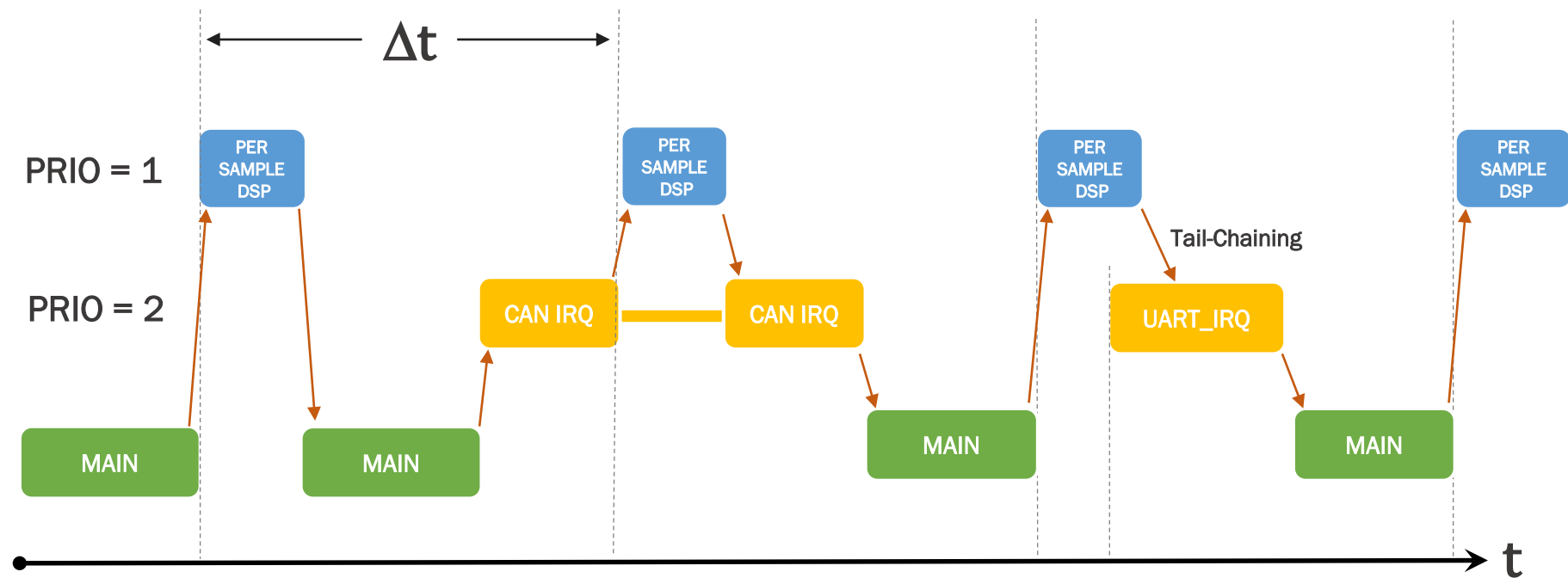
<http://community.arm.com/docs/DOC-2607>

IRQ Latency and Nesting

- The IRQ latency & nesting in the Cortex M allows for efficient “sample by sample processing”. Control loops, real time IIR, etc.



IRQ Latency and Nesting



ARM Cortex M Priorities – Lower Number is Higher Priority

Cortex M – Lots of Choice

- M0 --- Low End, Low Power 32-bit. 8-bit/16-bit replacement
- M3 --- Main Stream 32-bit
- M4 --- M3 + DSP → What We are here for today
- M7 --- More DSP under the hood through better pipeline and memory architecture.

Other variants exists but these are main stream

ARM Cortex M4

M3 + DSP instructions

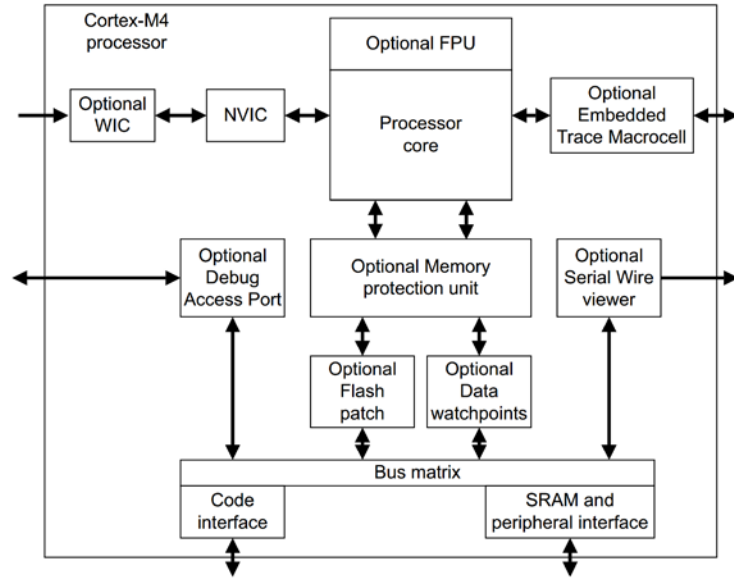


Figure 1-1 Cortex-M4 implementation

3 Stage Pipeline

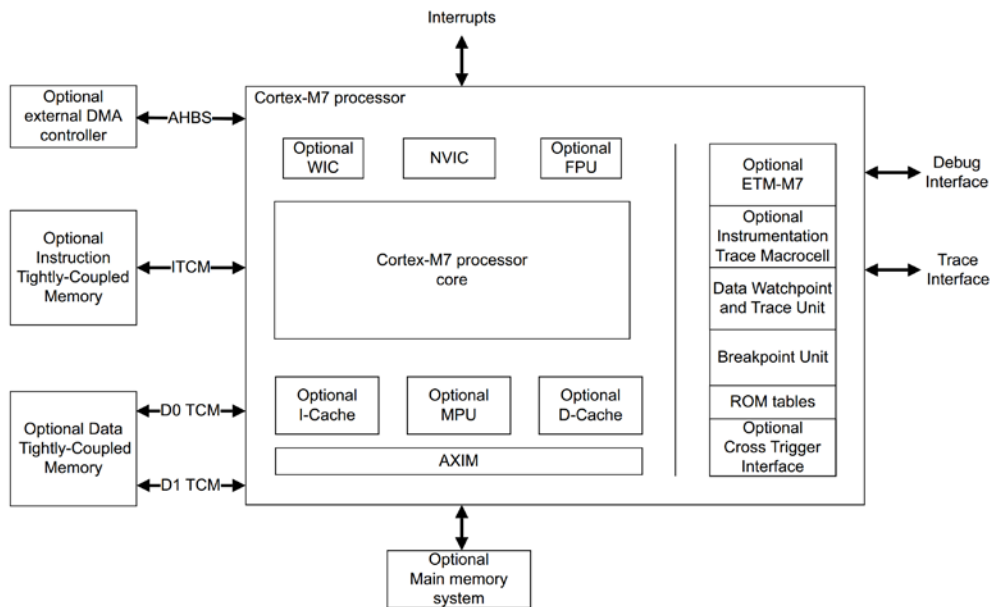
Optional 32-Bit IEEE Floating point

Some implementations have a I/D cache patched in

http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dqug.pdf

ARM Cortex M7

M4 + Better Memory Architecture & Pipeline



6 Stage Pipeline

**Optional 32-Bit or 64-bit IEEE
Floating point**

Tightly Coupled Memory

Figure 1-1 Cortex-M7 processor implementation

http://infocenter.arm.com/help/topic/com.arm.doc.dui0646a/DUI0646A_cortex_m7_dgug.pdf

The “Bible”

- <http://infocenter.arm.com/help/index.jsp>
 - Cortex M4/7 Technical Reference Manual
 - Cortex M4/7 Devices Generic User Guide

When Looking for help....

- Silicon vendors generally do not provide the ARM documentation for their implementation.
- The vendor reference manuals document vendor peripherals and implementation notes. I.E. IRQ slot assignments.
- Details on the core CPU is found in the ARM technical reference manuals...
- Good news is that ARM provides abstraction to the core functions. Easy to move to other vendors.

Lots of Silicon Available

- NXP, Texas Instruments, Silicon Labs (formerly EFM Micro), Spansion, Analog Devices, ST..... Just about everyone Even Microchip now that they purchased Atmel!
- (Microchip was proud of their PIC32MX with MIPS core... PIC32MZ was a DISASTER). Future of M4/M7 from Atmel Parts?
- We can get everything from a 3mm x 3mm package to a BGA256 between the vendors. Lots of variety in IO, peripherals, etc...
- Speeds from 20MHz to 240MHz

Some Notable M4 Implementations

- **NXP LPC43xx and LPC541xx** – Good “digital” implementations
Multi-cores, SGPIO, SCT, ROM bootloaders, up to 204MHz....
- **Freescale (NXP) Kinetis** - Great ADCs. Embedded wireless options.
Lot's of choice across ARM Cortex M4.
- **Analog Devices** – Motor controller / power analysis variants. SINC3
Filters... Great ADCs, direct connection to isolated ADCs.
- **TI MSP432** - Marketing play for MSP430 16-bit series. M4 + older
MSP430 Peripherals.
- Make sure to look at everyone (.... Lot's of variety in packages,
peripherals, etc)

Some M7 Implementations

- **ST Micro** - STM32F7 Lots of good Audio Interfaces, external SDRAM, TFT Display Driver. Up to 216MHz
- **Microchip (Atmel....)** SAM S/E/V70. Lots of peripherals... very similar in Spec to STM32F7. Up to 300Mhz...
- **NXP (Freescale....)** Kinetis KV5 → Real Time / Motor Control. 240Mhz

Right now this is it.... M7 is new

Software Toolchains

- KEIL & IAR → If cost is not an Issue. Floating licenses (the only real option in my option) are very expensive.
 - “Vendor” Tools
 - Kinetis Design Studio
 - NXP LPCXpresso (Formerly Code Red)
 - TI Code Composer
 - Silicon Labs Simplicity Studio
- *Eclipse + GCC+ Debugger is the common element*
- ATMEL Studio 6 → Microsoft Visual Studio as a front end to GCC

MCU Xpresso
Spring of 2017

Software Toolchains

- **Rowley Crossworks** → Proprietary GUI front end for GCC
- **Atollic** → A well put together Eclipse + GCC + Debugger. Free and Paid options
- <http://gnuarmclipse.github.io/> open Source Eclipse + GCC + Open OCD Debugger
- **Altium/Tasking** → Eclipse + Proprietary Compiler. Well respected.... Good tools for static code analysis

Software Toolchains

- Cobble together GCC, GDB, Open OCD etc. on your own
<https://launchpad.net/gcc-arm-embedded>
- **Note: The GCC ARM tools are being worked on by ARM employees!**
- Commercial Tools (Keil, IAR) does not necessarily mean fast/better compiling. There are a lot of details to consider..... Middleware, etc.
- I prefer Eclipse based tools. The extensibility beats IAR and KEIL every time (IMHO). Lots of plugins (both free and commercial) for automating everything.

Low Cost Dev Boards

- Freescale FRDM \$(15-\$60)
- NXP LPCXpresso (\$25-\$40)
- Atmel Xplained (\$30-\$40)
- ST Discovery (\$15 - \$50)
- TI Launch Pad (\$15-\$30)



Let's Talk Bare Metal

- Assembly language programming not required BUT.....
 - You should take a look at the instruction set
 - Some real gems in the manual....
 - You **need** to think about YOUR algorithm and how it will map.
 - You **need** to be familiar with your compiler and optimization levels.
 - You need to be familiar with the documentation

<http://infocenter.arm.com/help/index.jsp>

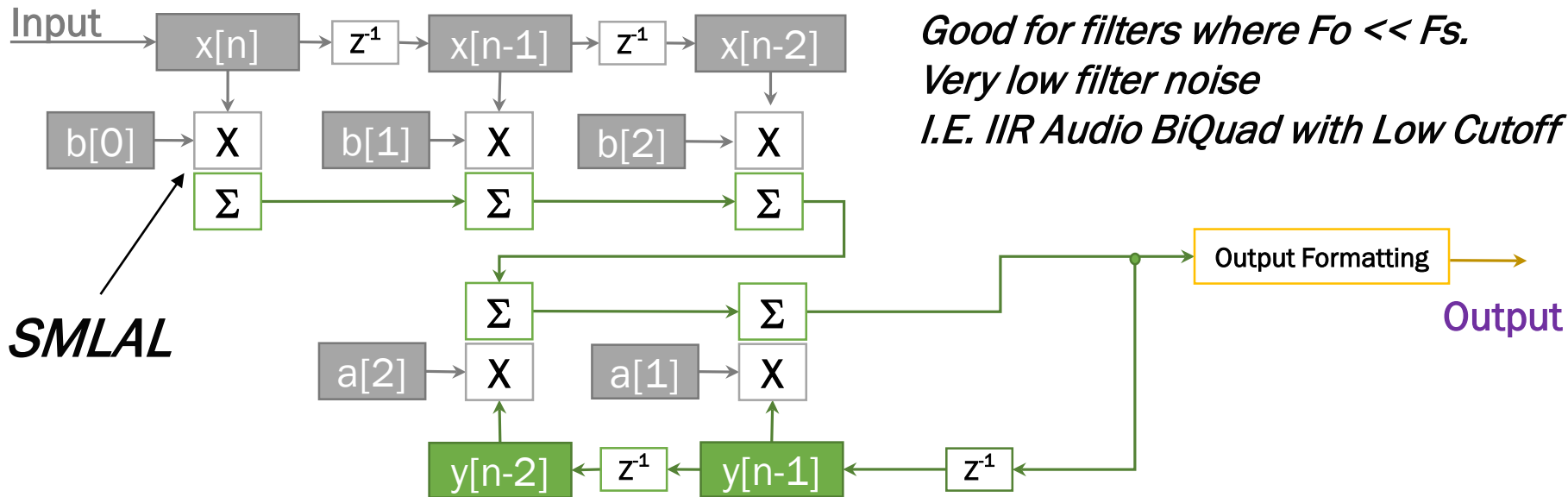
Some Interesting Instructions

- Integer: **UMLAL** and **SMLAL**
- Multiply, with optional Accumulate, using 32-bit operands and producing a 64-bit result.

http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf

Using these we can have IIR filters with 64-bit precision in the accumulation / State Variables.

High Precision Filter Using SMLAL



Gray = 32-bit

Green = 64-bit

Purple = User Defined

Some Interesting Instructions

- SMUAD and SMUSD

Signed Dual Multiply Add and Signed Dual Multiply Subtract

16-bit FFT Butterfly?

$$\begin{aligned} y_0 &= x_0 + x_1 \omega_n^k \\ y_1 &= x_0 - x_1 \omega_n^k, \end{aligned} \quad \omega_n^k = e^{-\frac{2\pi i k}{n}}$$

Fast FIR,IIR → You can process multiple 16-bit taps at one time

Some Interesting Instructions

- VLMA, VLMS ,VNMLA, VNMLS, VNMUL

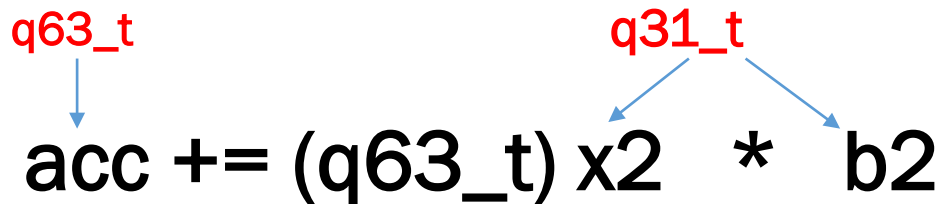
Multiplies two floating-point values, and accumulates or subtracts the results . Floating-point multiply with negation followed by add or subtract.

For floating point filters. Note that with **SMLAL** we can get more precision with the integers!

***Floating Point Note:** Your toolchain may **not** have the FPU initialized in the C startup routines.*

Do I need to write assembly code?

- Sometimes it is useful. I have found the SSAT instruction useful in many cases.
- Some of the instructions can be inferred in C code

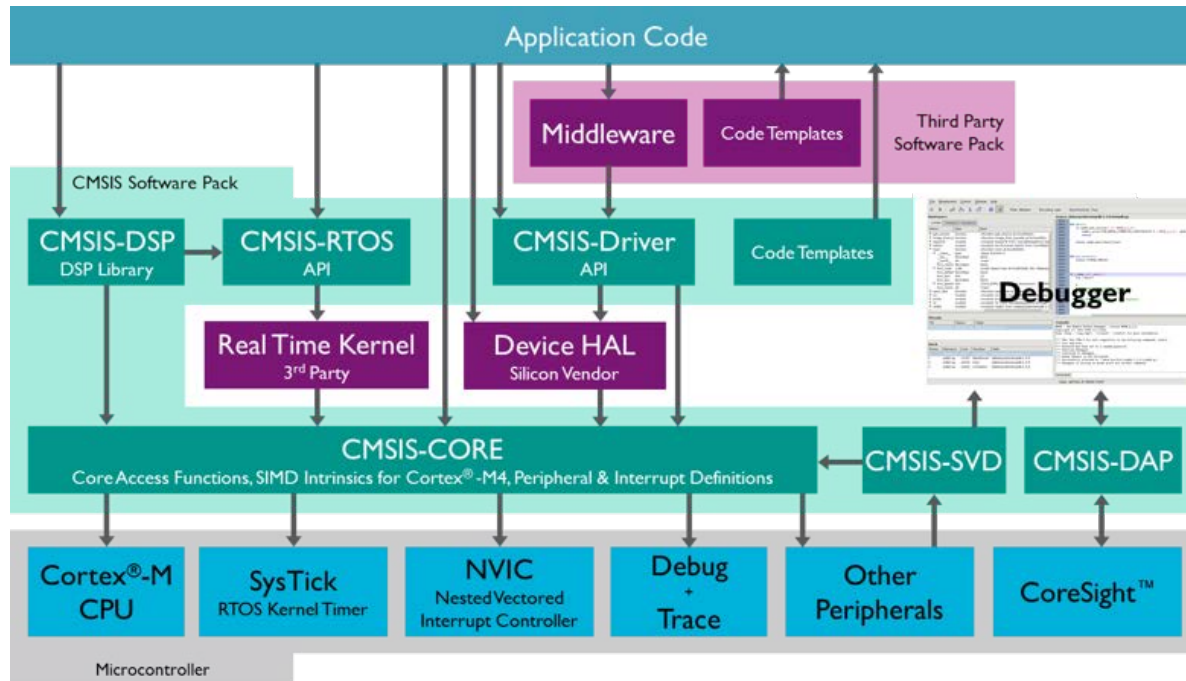

`acc += (q63_t) x2 * b2`

Will compile to an SMLAL (with the right optimization levels!)

This is true for both GCC AND Commercial Compilers!

Let's Take a look.....

Cortex Microcontroller Software Interface Standard CMSIS



Kicking the CMSIS Tires

- CMSIS Code can be acquired here:

<http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php>

Open source.... Easy to look at.

Let's look at the package

Some Library Fundamentals

- Fixed Point Data types

- Q0.7

- Q0.15

- Q0.31

- Q0.63

- Defined data types used a fixed point scaling normalized from -1 to 1
 - Some functions output other fixed point scalings (FFT)
 - *most* functions have a Q0.15 and Q0.31 version for data inputs. Some use internal resolutions at Q0.63 or Q1.62 for high precision
 - Documentation does a good job an indicating when you need to scale, saturation, etc.

- Floating Point – 32-Bit IEEE 754.

- Some M7 implementations support double precision floating point

Filtering Functions

- IIR, FIR, Interpolation, Decimation.....
- Post shift to implement coefficients > 1
- Pay attention to coefficients
- Let's look into the docs

Frequency Domain

- Real & complex FFTs
- DCT
- Bit reversed ordering options
- Let's look in the documentation..... Lots to talk about...

Building the CMSIS Library

- CMSIS includes Keil uVision build example projects.
- You should learn how to build the library from scratch.
 - Lots of options –> optimization levels, compiler flags, etc.

Building the CMSIS Library

- **SoftABI**

- Single precision floating point operations are implemented in hardware and hence provide a large performance increase over code that uses traditional floating point library calls, but when calls are made between functions any floating point parameters are passed in ARM (integer) registers or on the stack.
- SoftABI is the 'most compatible' as it allows code that is not built with hardware floating point usage enabled to be linked with code that is built using software floating point library calls.



Building the CMSIS Library

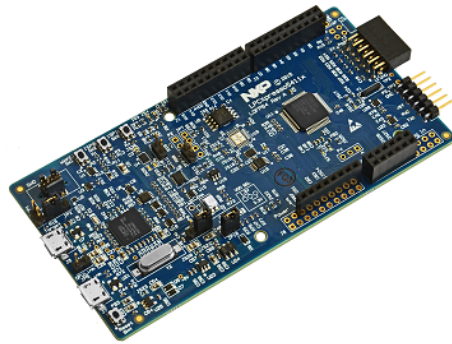
- **HardABI**

- Single precision floating point operations are implemented in hardware, and floating point registers are used when passing floating point parameters to functions.
- HardABI will provide the highest absolute floating point performance, but is the 'least compatible' as it means that all of the code base for a project (including **all** library code) must be built for HardABI.



M4 CMSIS Library Performance

- LPCXpresso 54114 – Dual Core M4/M0
- These are high end implementations of the M4. Lots of RAM and some sections are on separate connections to the bus matrix
- We can use them to develop scaling equations for RAM based and FLASH based execution profiles.



M7 CMSIS Library Performance

- TWR-KV58F220M
- Commercially available dev board (not a lot of boards compared to M4)
- Has I/D Cache as well as OCM, ITCM and DTCM



CMSIS Library Performance – Test Methodology

- We want to get a measure of “clock cycles” to execute code. (not time).
We can use this to scale to other CPUs
- The Internal SysTick timer runs at the same rate as the core. 24-Bits
- Reset SysTick → Start SysTick → Stop SysTick → Read SysTick value
 - This is our “overhead” in clock cycles to measure function execution
- Reset SysTick → Start SysTick → Execute DSP Code → Stop SysTick → Read SysTick and subtract “overhead”
- Between the RAM based and Flash based platforms, we should get some real world profiles.

CMSIS Library Performance – Test Methodology

- Real and complex FFT at varying block sizes for Q15, Q31 and float.
- IIR Filter – high precision Q0.31, Q0.15 and float
- FIR Filter – high precision Q0.31 , Q0.15 and float
- Eli's per sample high precision Q0.31 BiQuad
- A hand written Goertzel's algorithm
- Look at different optimization Levels (O0,O1,O2,O3,Og,Os) and SoftABI vs HardABI
- M4 and M7 are binary compatible (if we don't use doubles!) I link to the same libraries for all projects (GCC)
- GCC & KEIL for M4 GCC for M7

Code available on GITHUB

<https://github.com/ehughes/ESC-M4>

CMSIS Library Performance – Test Methodology

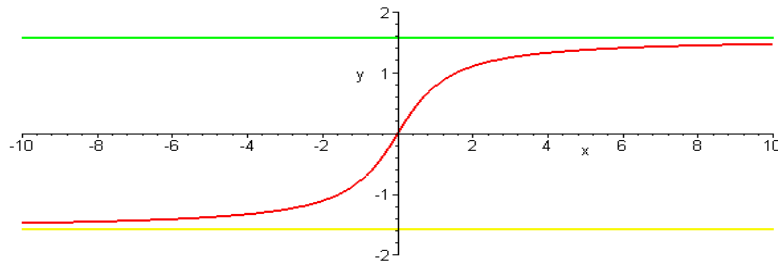
- Now for the results.... Let's look at Eli's spreadsheet!

Some Fun Applications/Demonstrations

MonkeyJam



- <https://community.freescale.com/docs/DOC-100149>



Some Fun Applications/Demonstrations

MonkeyListen



- <https://community.freescale.com/docs/DOC-100207>

Questions and Discussion

Thanks for Coming!

Eli Hughes
ehughes@wavenumber.net