# Rapid Cross Platform DSP with the ARM Cortex M4


Eli Hughes
ehughes@wavenumber.net


# Presentation Scope

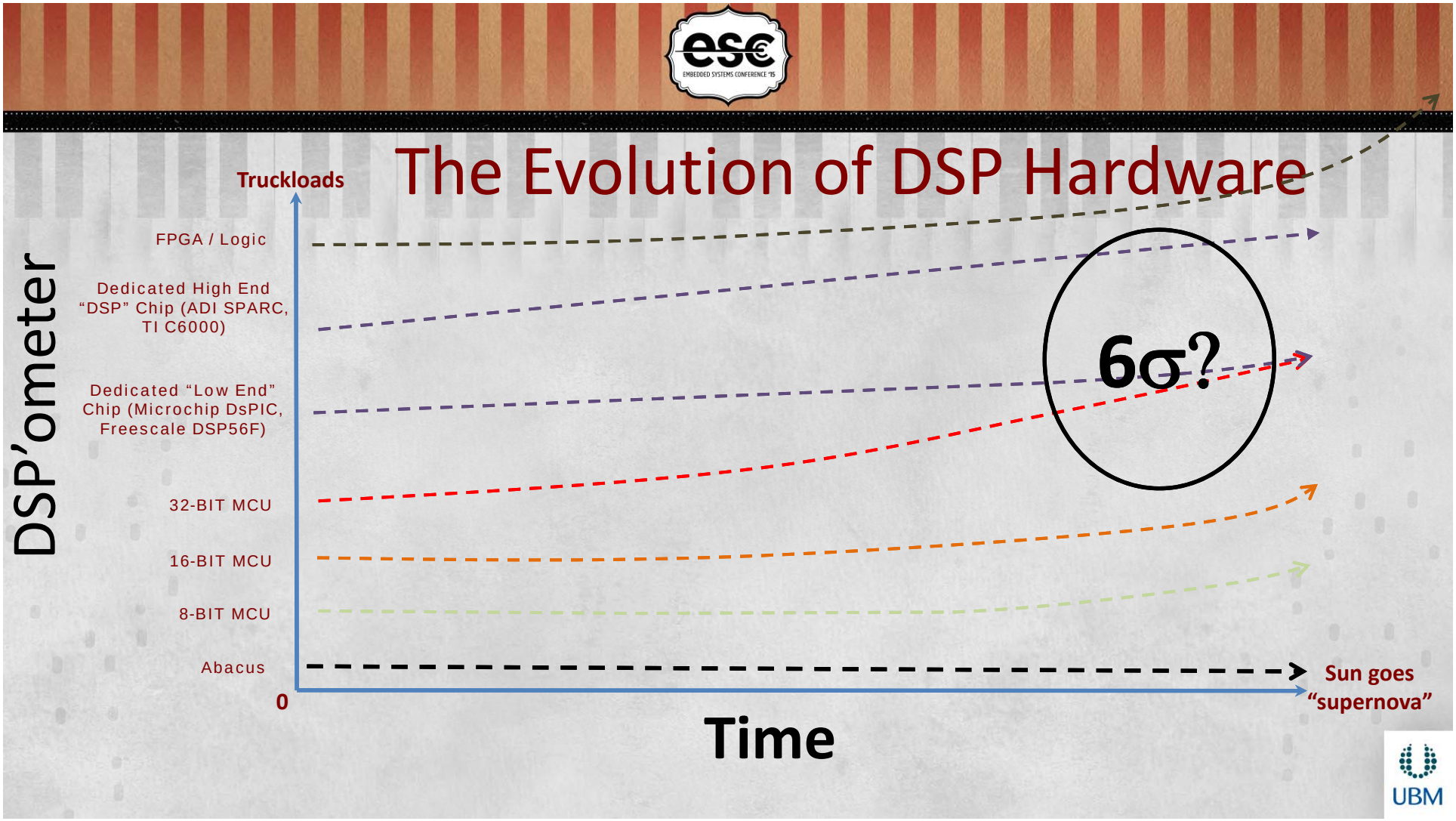
- Quick introduction to the Cortex M4
- Vendors & tool chains
- Look at some “interesting” instructions
- CMSIS-DSP library
- Performance stats
- Some fun demos

UBM

# 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 on a $5000 FPGA

# The Evolution of DSP Hardware




# The 6σ

- 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 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.

# 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?

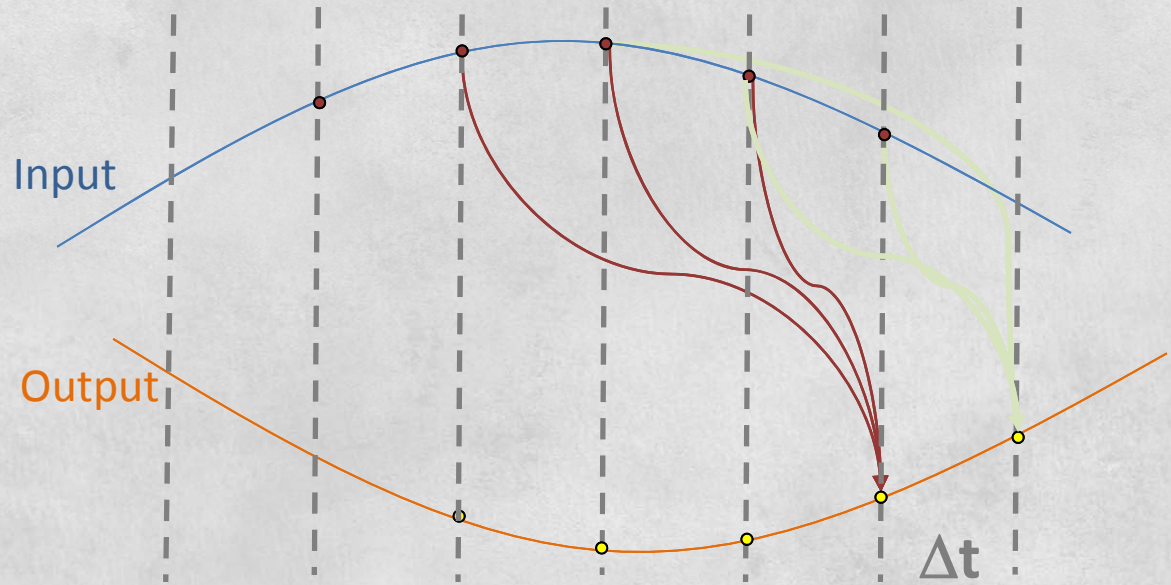
  -To fix the architectural problems in the ARM7TDMI

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

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

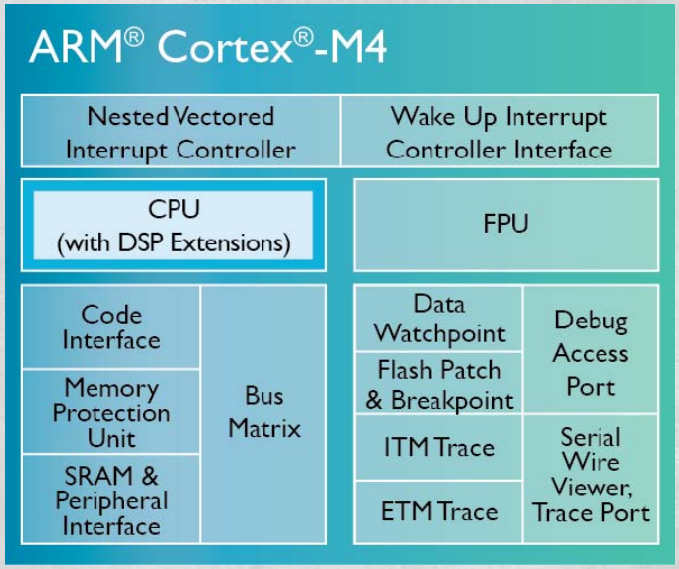
# IRQ Latency….

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



# ARM Cortex M4

- M3 + DSP instructions + Optional hardware floating point
- 2 Variants: M4 and M4F (hardware floating point)




# The “Bible”

- **http://infocenter.arm.com/help/index.jsp**
  - **Cortex M4 Technical Reference Manual**
  - **Cortex M4 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

- Freescale, NXP, Atmel, Texas Instruments, EFM Micro, Spansion, Analog Devices, ST…..  Just about everyone except Microchip!
- 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 Implementations

- **NXP LPC43xx** -- Good “digital” implementation. Multiple cores, SGPIO, SCT, ROM bootloaders, 204MHz….
- **Freescale 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
- Make sure to look at everyone…. Lot’s of variety in packages, peripherals, etc.

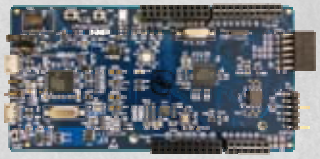
# Toolchains

- **KEIL, IAR** → If cost is not an Issue
- "**Vendor" Tools** → Freescale Codewarrior, Kinetis Design Studio, NXP LPCXpresso, TI Code Composer. Most are free or free to start. Eclipse + GCC+ Debugger. KDS is a "pure" open tool chain.
- **ATMEL Studio 6** → Microsoft Visual Studio as a front end to GCC
- **Rowley Crossworks** → Proprietary GUI front end for GCC
- Other GCC based front-ends. (CooCox, Code Sourcery….)
- Cobble together GCC, GDB, Open OCD etc. on your own https://launchpad.net/gcc-arm-embedded

# Low Cost Dev Boards

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






UBM

# 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

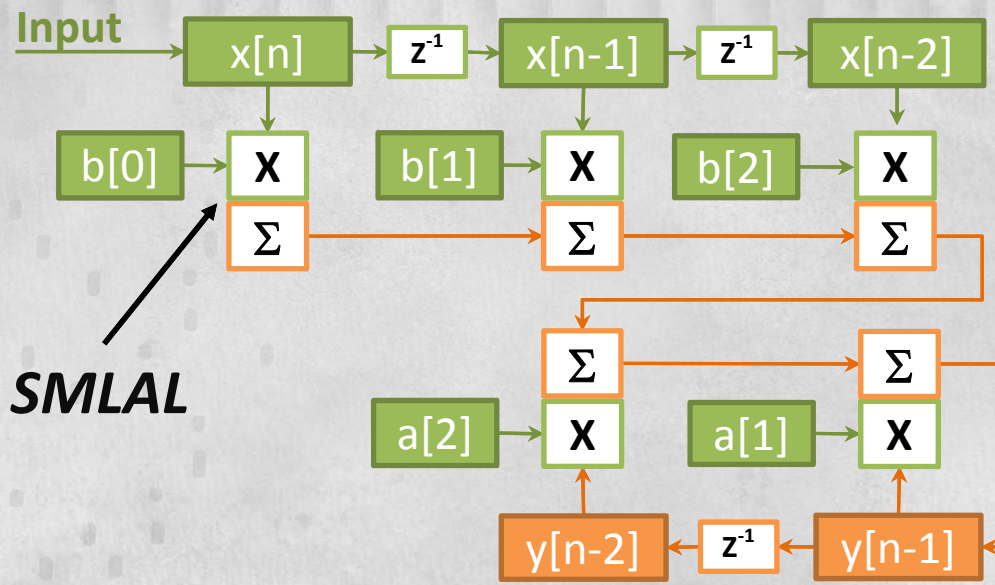
# Some Interesting Instructions

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

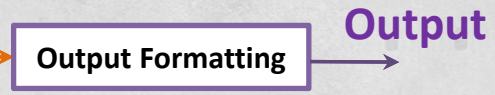
http://infocenter.arm.com/help/topic/com.arm.doc.dui0553a/DUI0553A_cortex_m4_dgug.pdf

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

# High Precision Filter Using SMLAL




***Good for filters where Fo << Fs.***
***Very low filter noise***
***I.E. IIR Audio BiQuad with Low Cutoff***




**Green = 32-bit** **Orange = 64-bit** **Purple = User Defined**

# Some Interesting Instructions

- **SMUAD and SMUSD**

Signed Dual Multiply Add and Signed Dual Multiply Subtract

**16-bit FFT Butterfly**

$$y_0 = x_0 + x_1\omega_n^k$$
$$y_1 = x_0 - x_1\omega_n^k,$$

$$\omega_n^k = e^{-\frac{2\pi ik}{n}}$$

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

# Some Interesting Instructions

- **VLMA, VLMS ,VNMLA, VNMLS, VNMUL**

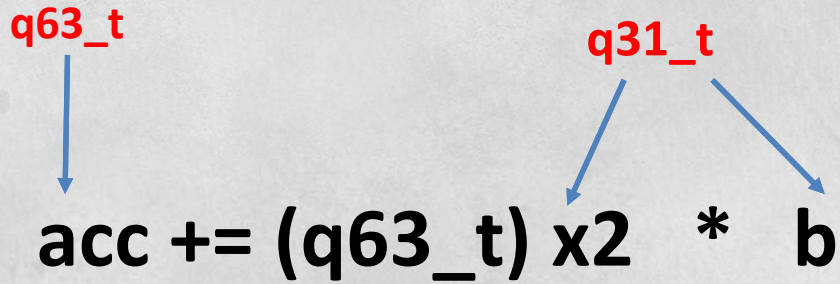
Multiplies two floating-point values, and accumulates or subtracts the results . Floating-point multiply with negation followed by add or subtract.

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?

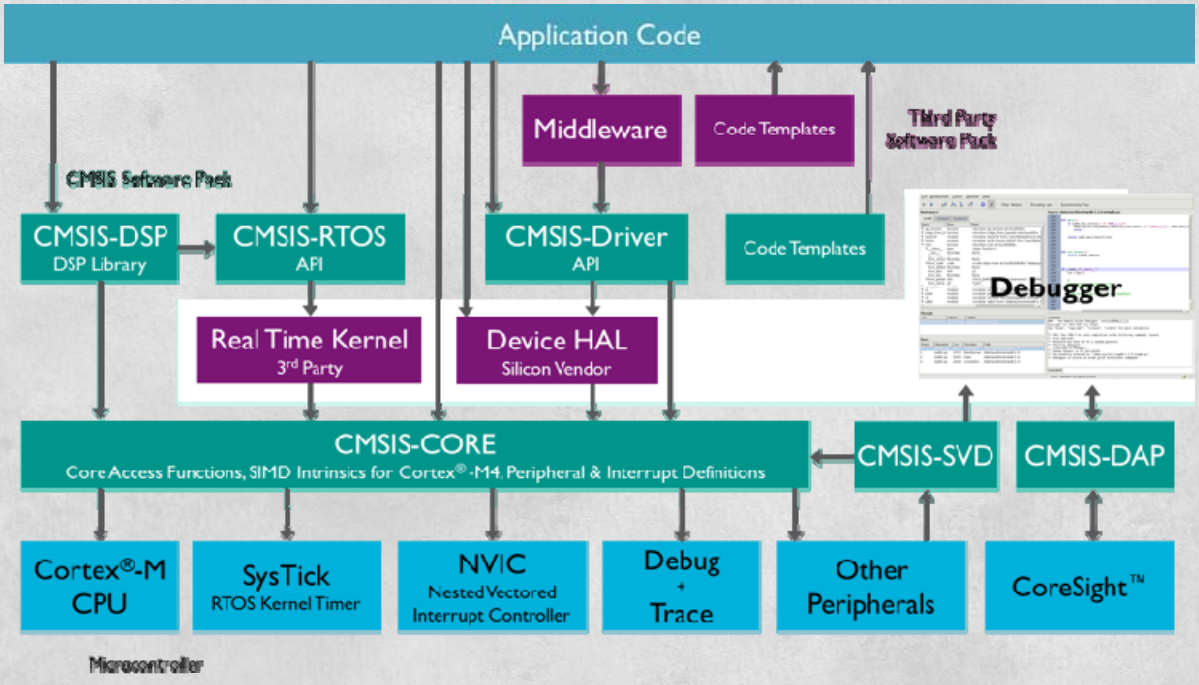
- Sometimes it is useful. I have found the SSAT instruction useful in many cases.
- Some of the instructions can be inferred in C code




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

# 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's will support doubles in future)


# 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.

**http://www.support.code-red-tech.com/CodeRedWiki/CM4_FloatingPoint**

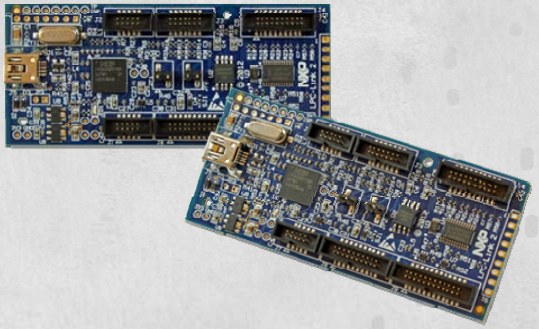
# Building the CMSIS Library

- **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.

**http://www.support.code-red-tech.com/CodeRedWiki/CM4_FloatingPoint**

# CMSIS Library Performance

- LPC-LINK-2 as an evaluation board for NCP LPC4370 @204MHz – Triple Core (CortexM4 + 2 Cortex M0).
- RAM only Device.  Can get close to Ideal performance



- LPCXPRESSO4337 – eval board ($25) for LPC4337.  Dual Core (Cortex M4F + Cortex M0).  Flash based.
- These are high end implementations of the M4.  We can use them to develop scaling equations for RAM based and FLASH based execution

# 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
- LPC43xx Timer 0 can run at core rate. 32-bit
- Reset Timer0, Start Timer0 → Stop Timer0→ Read Timer0 value
  - This is our “overhead” in clock cycles to measure function execution
- Reset Timer0 →Start Timer 0->Execute DSP Code→Stop Timer 0 →Read Timer0 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
- Look at different optimization Levels (O0,O1,O2,O3,Og,Os) and SoftABI vs HardABI

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

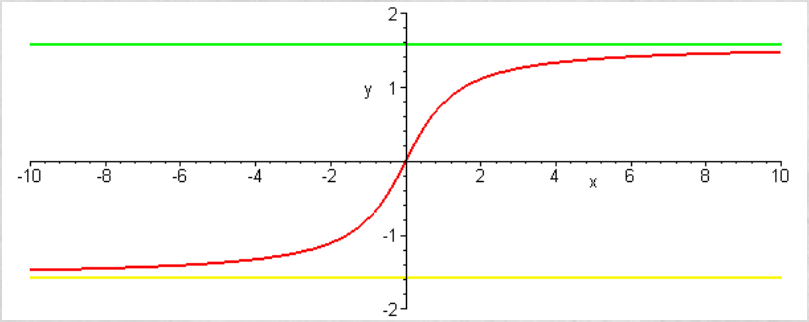
## Active Pickguard

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

# Some Fun Applications/Demonstrations

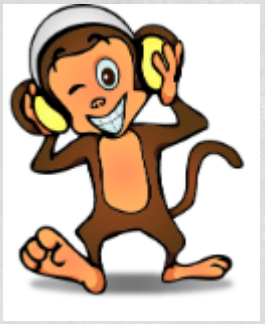
## MonkeyJam

- **https://community.freescale.com/docs/DOC-100149**



# Some Fun Applications/Demonstrations

## MonkeyListen



- **https://community.freescale.com/docs/DOC-100207**


# Questions and Discussion

UBM