

D.S.P corner

How STM32F4/F7 fulfill Digital Signal Processing requirements

Examples of DSP Applications

2

■ Typical applications:

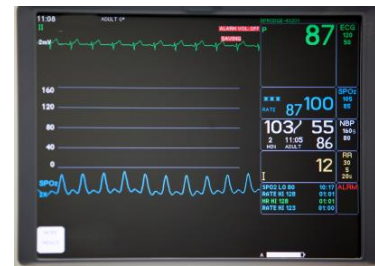
■ Audio player functionality:

- MP3 audio, HiFi Audio, equalizer...
- Voice Coding /decoding
- Hand free system



■ Sensor data processing:

- Sensor HUB



■ Medical devices:

- Heart rate monitor, Breath Help



■ Control applications:

- Motor control

■ Industrial applications,...

- Filtering, Radar (Doppler)
- Vibration analyzer



Analog portfolio provides
a lot of functions

- **Some common & low cost ...**
 - Resistor, Capacitors,
 - Inductors, Delay Line
 - Diode & transistor
 - Comparator
 - OpAmp
- **From which you can build**
 - RC, LC & RLC passive filters
 - With OpAmp you can add, subtract
make active filters
 - Also Sinewave generators
 - Analog Multiplier (Gilbert cell)

Digital processing provides a lot of
instructions or hardware blocs ...

- **Some replicate straightforwardly analog**
 - Absolute value
 - Compare
 - Multiply by constant
 - Delay line (RAM)
 - Multiply 2 variables (fast HW multiplier)
- **Some needs programs to replicate**
 - using combination of instructions, delay line
to replicate analog blocs made of
OpAmp, resistors, inductors or capacitors
- **And you can also implement much more
complex functions that would be impossible
in analog**
e.g: voice encoding, recognition

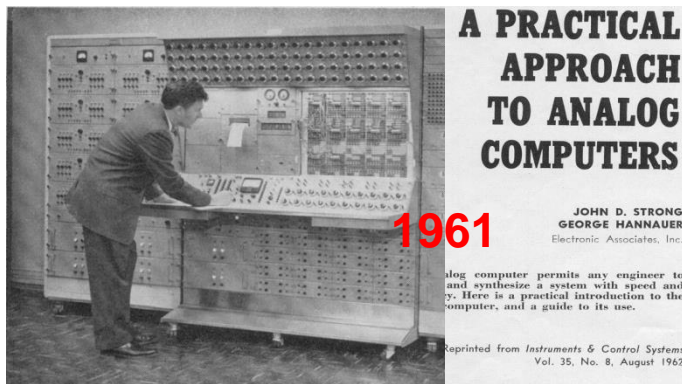


FIG. 1. PACE 231R analog computer.



Digital Signal Processing provides

- **Lot of advantages**

- Flexible → parameters change just by SW
- Easier upgrades
- Highly linear (within dynamic range)
- Complex algorithms fit in single chip
- Insensitive to component tolerances, aging, environmental conditions
- Better control over accuracy
- Make possible Data channel multiplexing in communication

- **But have limitations**

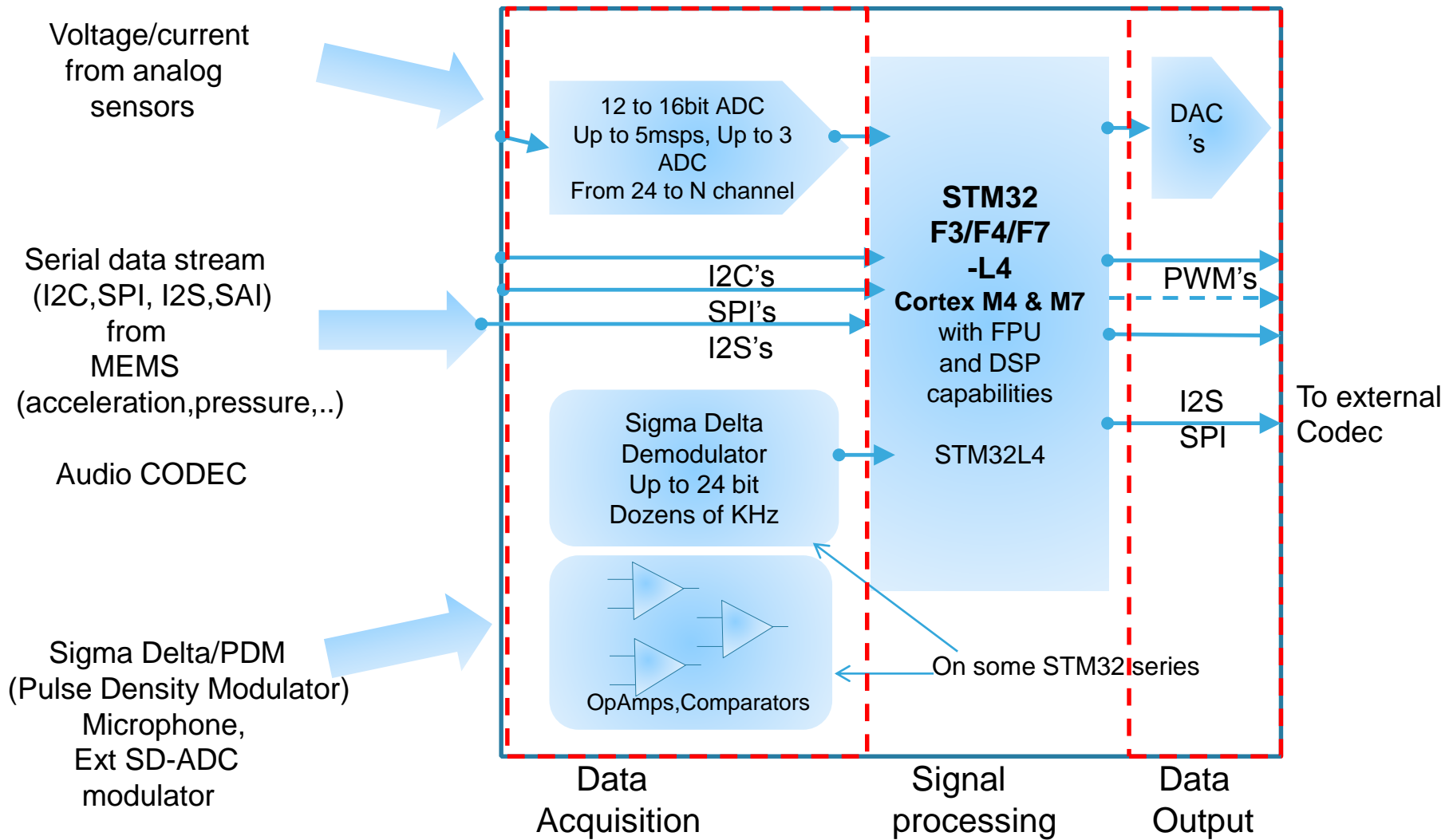
- Discrete-time artifacts (aliasing)
- Finite word-length effect (computation noise) (DSP is not fully “noiseless”)
- Processor speed limitation for wide band/High frequency ANALOG can work on GHz!!
- Consumption in some case
- Clock and switching can cause interference

- **And still need analog**

- Front end “pure” analog part for :
 - Amplification /Impedance matching
 - Anti- aliasing (see next)
- ADC (external or internal) with ad-hoc resolution/accuracy
- Smoothing-Amplifying output

STM32: On-chip resources for DSP

5



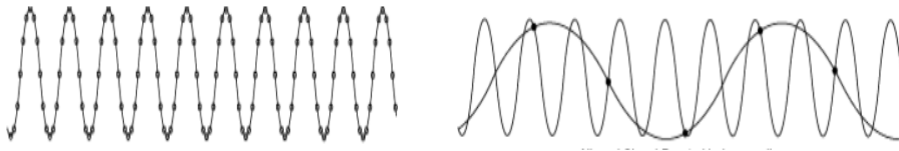
DSP constraints/limitations: Shannon rule

6

- The Nyquist/Shannon sampling theorem state that:

“All information of a signal is kept when it is sampled at a frequency which is at least **TWICE** the max frequency of the input signal”

Aliasing is what you see in Western movies... when the wheels seems to turn wrong



- Need to “cut “ (filter) frequencies above $F_s/2$ (so called **anti-aliasing filter**)

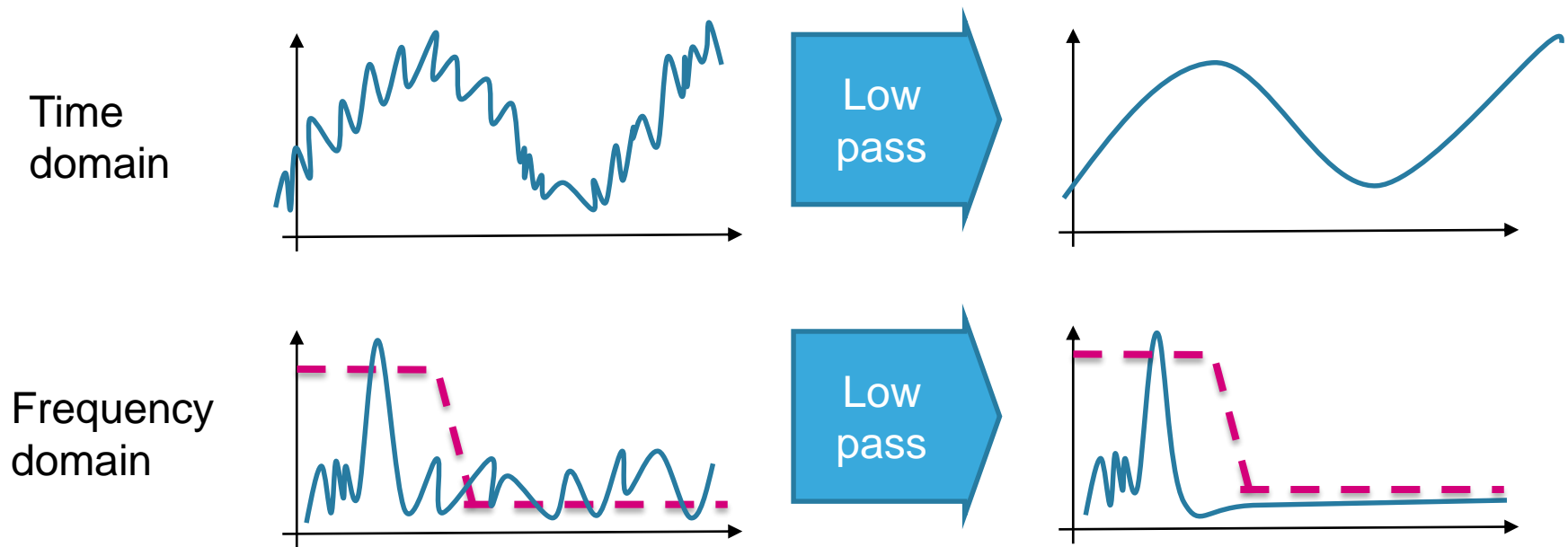
Example : Phone line: Voice bandwidth is limited to **300 to 3400 Hz** , **$F_s = 8\text{KHz}$**

- In practice, taking a ratio of only two lead to need a very sharp filter (need a lot of OpAmps, capacitors) ; **higher ratio allow “smoother” filter**
- A consequence is also that DSP processing must last less than sampling period
Implies that DSP task are often at highest priority level (regardless RTOS allocation)

Filter is a block designed to pass only certain frequencies of the signal

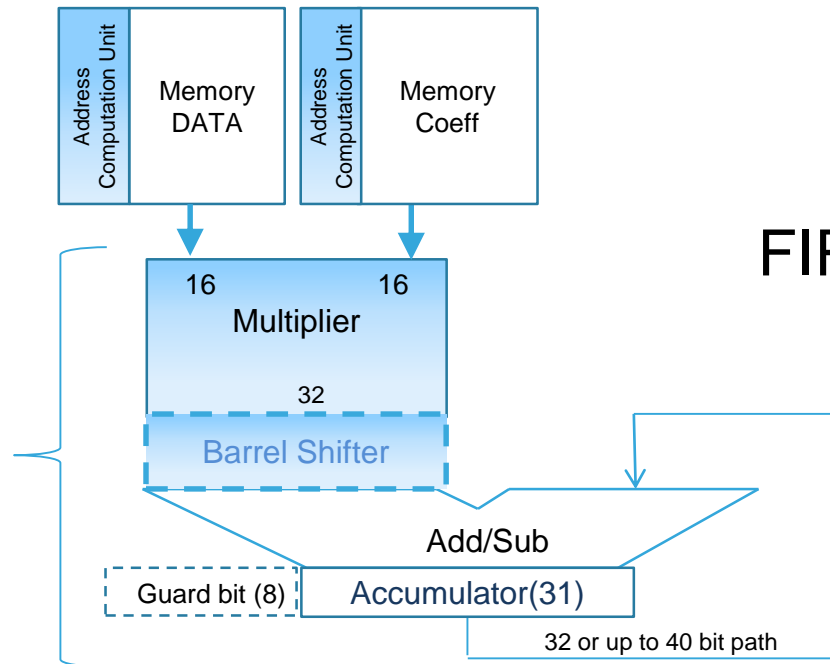
- Low Pass, High Pass, Band Pass or custom

Low pass example



Finite Impulse Response filter (FIR)

8



Finite Impulse Response Filter

- Sum of N last weighted samples of a signal

$$\text{FIR} : y(n) = \sum_0^N a(i) * x(n - i)$$

- 2 memory read accesses And 2 address Updates
- Multiplication
- Addition to previous result (accumulation)
 - And loop for next tap

DSP makes intensive use of MAC operation
Basic benchmark is MMAC/s (Million of MAC/S)

- Most operations are **dominated** by MACs(16 or 32 bit operations)

- FIR Filter

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

- IIR or recursive filter

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] \\ + a_1y[n-1] + a_2y[n-2]$$

- FFT Butterfly (radix-2)

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

- Correlation, convolution

$$(x * y)_n = \sum_{m=0}^{N-1} x[m]y[n-m]$$

How to improve MMAC/s.... ?

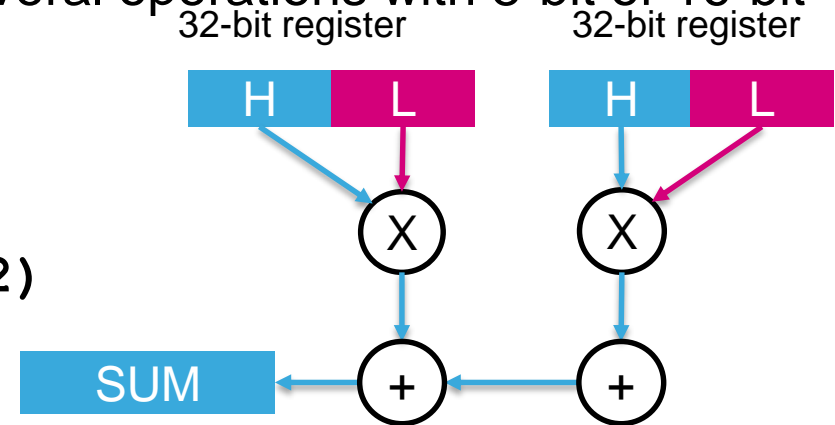
2 MAC per cycle with SIMD

10

- SIMD instructions : **Single Instruction Multiple Data (Cortex M4 & M7)**
- Allows to do simultaneously several operations with 8-bit or 16-bit data format

SUM = _SMLALD (C, S, SUM)

SUM += (C1*S1) + (C2*S2)

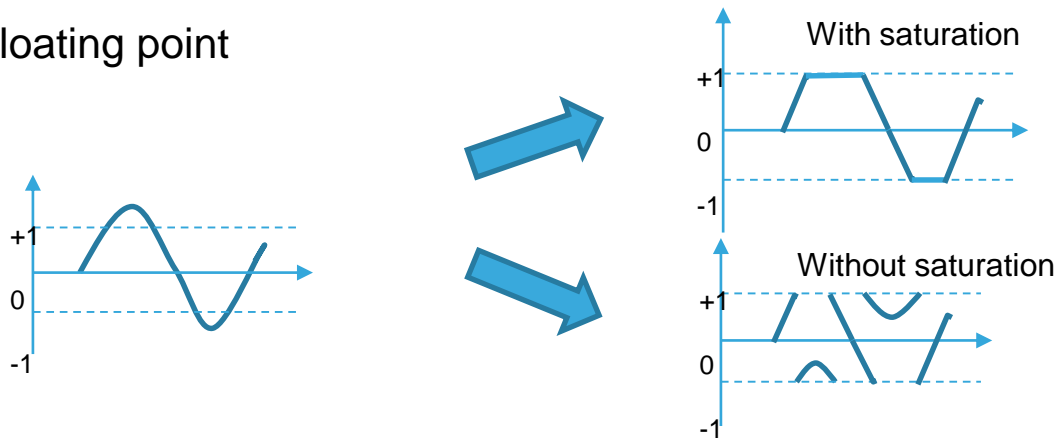


- **Benefits**
 - Parallelizes operations (**2x to 4x speed gain**)
 - Less Load operations
 - Maximizes register file use (1 register holds 2 or 4 values)
- **But Instruction are not standard “C compiler”**
 - Set of “intrinsic” functions extension (also used in “true” DSP)
 - Thanks to CMSIS “DSP” Library, can be avoided for most of DSP tasks

- Finite word length (e.g. 16bit) implies **quantization noise**

$$\begin{array}{rclcl}
 & \text{smallest } x & = -1 & = & 1.1111111111111111 \\
 q_{15_t} \quad x & = 0 & = & = & 0.0000000000000000 \\
 & \text{biggest } x & = 1-2^{-16} & = & 0.1111111111111111
 \end{array}$$

- Multiply 2 numbers lower than 1 \rightarrow result is smaller
 - Example in decimal : $0.1234 \times 0.0001 = 0.00001234 = 0.0000$!! if keep only 4
- Adding 2 numbers smaller than 1 can give a result > 1
 - Need “**saturation**” arithmetic
 - Do pre-scaling to avoid saturation
 - Use floating point



Fixed-Point

- $32 \times 32 + 32$ or $32 \times 32 + 64$ MAC takes 1 cycle
- SIMD instructions accelerate 16-bit and 8-bit math
- Limited to 16-integer registers in total; 3 are reserved by the compiler.

Floating-Point

- MAC takes at 1 cycle
- No SIMD
- Provide an additional 32 floating-point registers on top of the 16 integer registers. This facilitates loop unrolling and reduce stack usage

General Guidelines

- If your data types are 8- or 16-bit integers then you are better off with a fixed-point implementation.
- If input data type is 32-bits then only very simple algorithms will be more efficient in fixed-point.
- Floating-point is much easier to develop with – especially for complicated algorithms.

How to improve MMAC/S...?

Cortex M7 versus M4 improvements

- Remember “M7 architecture session” ?
- CORTEX-M7 Enhancements compared to Cortex-M4
 - Read data from memory in parallel with MAC operation
 - Branch in 1 cycle

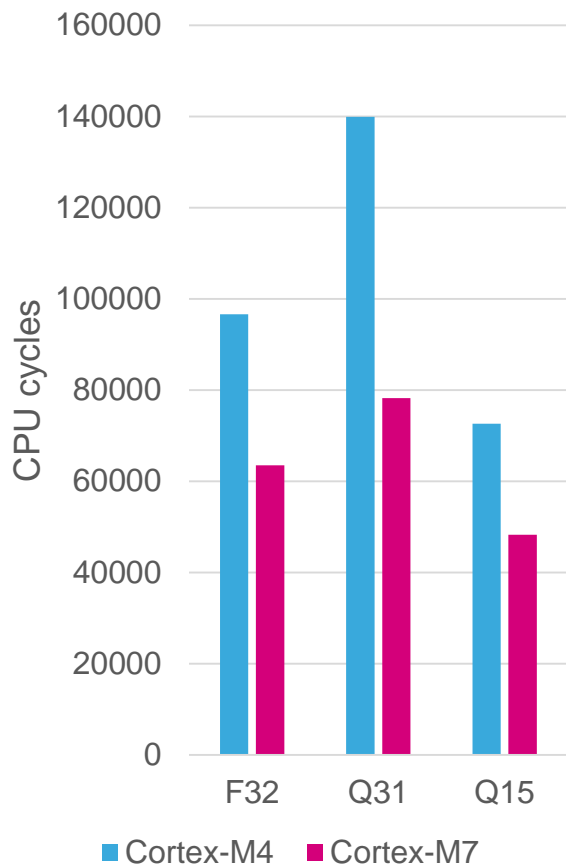
	Cortex-M3	Cortex-M4	Cortex-M7	Traditional DSP
Single cycle MAC		Fixed-point only	Fixed and floating-point	Y
Floating-point		Y	Y	Y
Fractional and saturating math		Y	Y	Y
SIMD operations		Y	Y	Y
Load and store in parallel with math			Y	Y
Zero overhead loops			Y	Y
Accumulator with guard bits				Y
Circular and bit-reversed addressing				Y

All those features make CORTEX-M7 a very powerful
“MAC CRUNCHER”

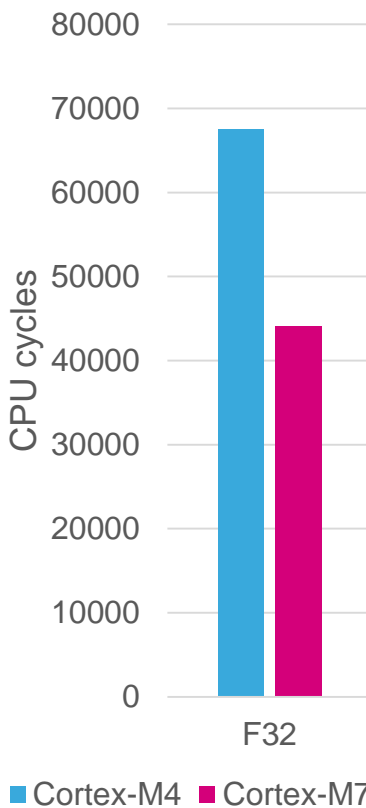
Performance comparison: M4 – M7

14

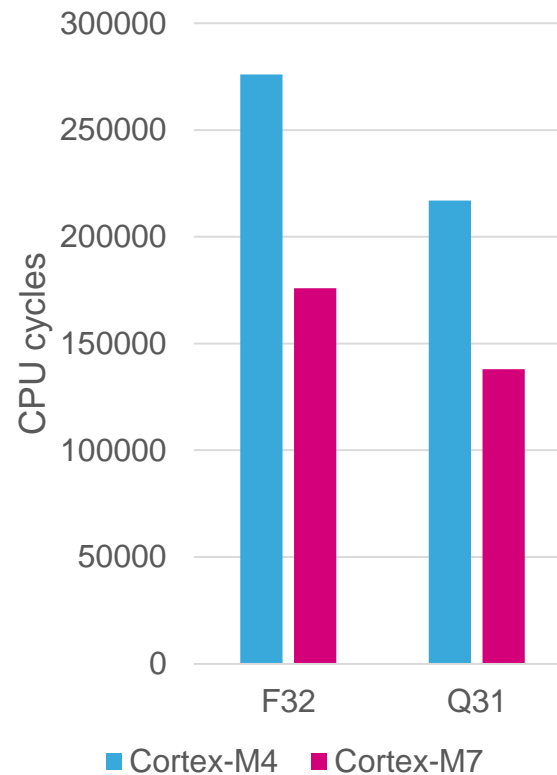
FFT



BiQuad 1024, 4 stages



FIR filter 1024, 100 taps



- A library of common DSP functions written in C using intrinsics
 - Source and object (.lib) files provided
 - Separate functions for operating on 8, 16 and 32-bit ints and 32-bit floats
- A collection of 61 algorithms including:
 - Basic maths: vector multiply, add, subtract, scale, shift, negate...
 - Statistics: root mean square, mean, standard deviation...
 - Fast maths: sine, cosine, square root...
 - Complex maths: conjugate, dot product, magnitude, multiply by real...
 - Filters: FIR, IIR, convolution, correlation..
 - Matrix algebra: addition, multiplication, scale...
 - Transforms: Fast Fourier, discrete cosine...
 - Controller: PID motor control, (Inverse)Park transform, (Inverse)Clarke transform...
 - Interpolation: linear and bilinear...
 - Support functions: type conversion, copy, fill...
- Provided free of charge by ARM



CMSIS 2.0.0: DSP Software Library on Cortex-M3 M4 and M4F (1/2)

16

- **Basic Math Functions**

- Vector Absolute Value
- Vector Addition, Subtraction
- Vector Dot Product
- Vector Multiplication
- Vector Negate
- Vector Offset
- Vector Scale
- Vector Shift

- **Fast Math Functions**

- Cosine, Sine
- Square Root

- **Complex Math Functions**

- Complex Conjugate
- Complex Dot Product
- Complex Magnitude
- Complex Magnitude Squared
- Complex-by-Complex Multiplication
- Complex-by-Real Multiplication

- **Filtering Functions**

- Biquad Cascade IIR Filters Using Direct Form I Structure
- Biquad Cascade IIR Filters Using a Direct Form II Transposed Structure
- High Precision Q31 Biquad Cascade Filter
- Convolution
- Partial Convolution
- Correlation
- Finite Impulse Response (FIR) Decimator
- Finite Impulse Response (FIR) Filters
- Finite Impulse Response (FIR) Lattice Filters
- Finite Impulse Response (FIR) Sparse Filters
- Infinite Impulse Response (IIR) Lattice Filters
- Least Mean Square (LMS) Filters
- Normalized LMS Filters
- Finite Impulse Response (FIR) Interpolator

CMSIS 2.0.0: DSP Software Library on Cortex-M3 M4 and M4F (2/2)

17

• Matrix Functions

- Matrix Addition, Substraction
- Matrix Initialization
- Matrix Inverse
- Matrix Multiplication
- Matrix Scale
- Matrix Transpose

• Transform Functions

- Complex FFT Functions
- DCT Type IV Functions
- Real FFT Functions

• Controller Functions (Motor Control)

- Sine Cosine
- PID Motor Control
- Vector Clarke/Inverse Clarke Transform
- Vector Park/Inverse Park Transform

Statistics Functions

- Maximum
- Mean Minimum
- Power
- Root mean square (RMS)
- Standard deviation
- Variance

Support Functions

- Convert 16-bit Integer value
- Convert 32-bit Integer value
- Convert 32-bit floating point value
- Convert 8-bit Integer value
- Vector Copy
- Vector Fill

Interpolation Functions

- Linear Interpolation
- Bilinear Interpolation

Need more info ?

18

For more info contact:

enrico.marinoni@avnet.eu

(Digital FAE for STM - MCU, WireLess (IoT), MEMS, PLM, etc)

roberto.rossetti@avnet.eu

(B.D.M.)





Thank you

www.st.com/stm32