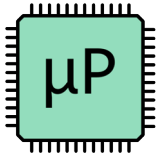
Digital Signal Processing

Lecture 15

Microprocessor-based Systems (E155)

Prof. Josh Brake



Outline

- What is digital signal processing?
- DSP in practice on the Cortex-M4
  - New features on the Cortex-M4 compared to previous iterations
  - Example: The Dot Product
  - General optimization strategies
  - Data types for DSP
  - Fractional arithmetic
- FIR Filtering Example

What is digital signal processing (DSP)?

- Filtering
- Control systems
- PID Control
- Fourier analysis

- Mathematical basis
  - Vector/Matrix computation
  - Complex numbers

Dot Product Analysis

- x[k], y[k] are arrays of 32-bit values that we want to use a 64-bit representation for z.

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

```
1  int64 dot_product (int32 *x, int32 *y, int32 N) {
2    int32 xx, yy, int32, k;
3    int64 sum = 0;
4    for(k = 0; k < N; k++) {
5      xx = *x++;
6      yy = *y++;
7      sum += xx*yy;
8    }
9    return sum;
10 }
```

Dot Product Analysis

Dot Product on Cortex-M3

Line of Code	Number of Cycles
`xx=*x++;`	2
`yy=*y++;`	2
`sum += xx * yy;`	3 - 7
Loop overhead	3
Total	10 - 14

Dot Product on Cortex-M4

Line of Code	Number of Cycles
`xx=*x++;`	2
`yy=*y++;`	1
`sum += xx * yy;`	1
Loop overhead	3
Total	7

All cycle counts are on a per sample basis.

Compare to Traditional Discrete DSP: ADSP-21160M

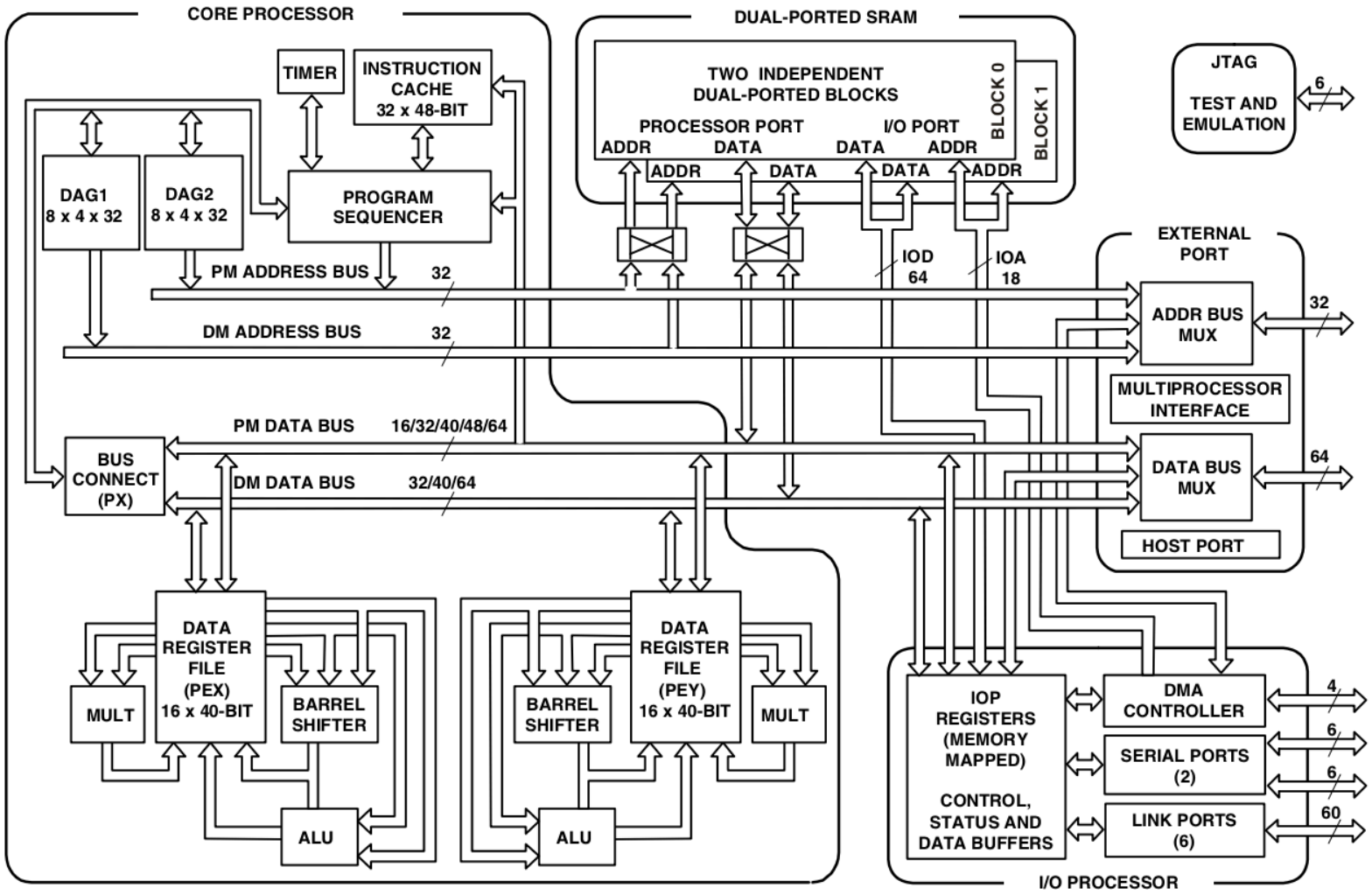


Figure 1. Functional Block Diagram

Can run dot product in N+2 instruction using straightforward optimizations!

https://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-21160M_21160N.pdf

Saturating Arithmetic

- Saturate at max value instead of allowing overflow

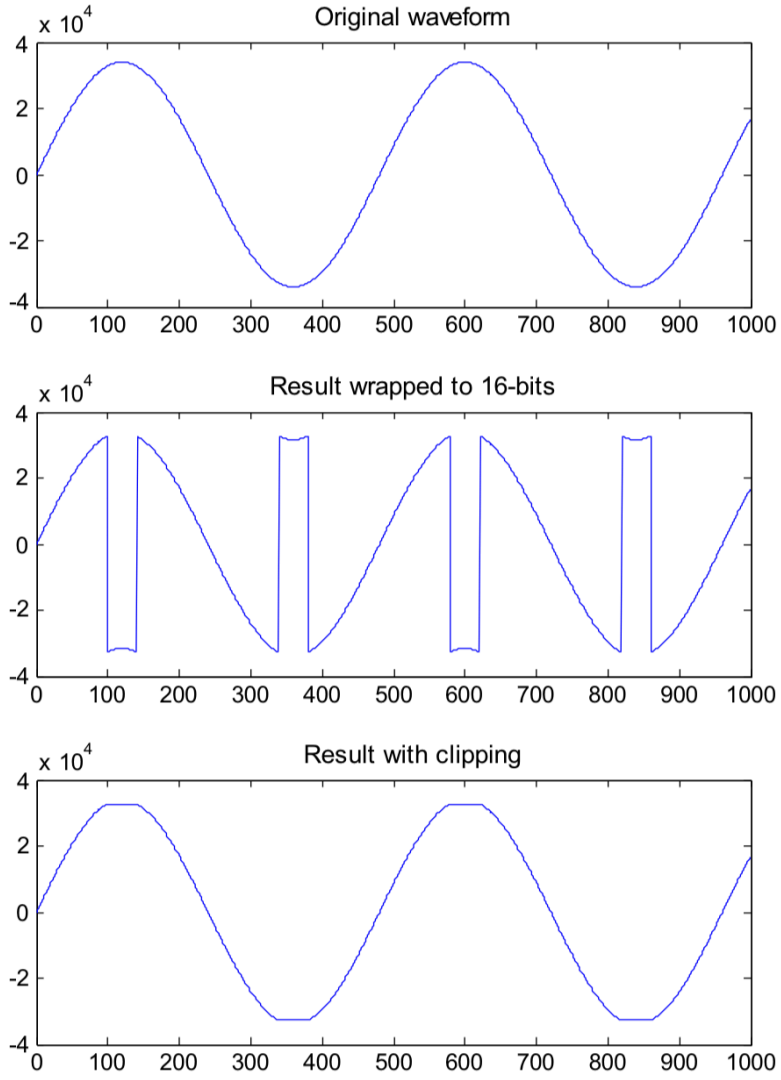


FIGURE 21.2

The effects of processing with and without saturation. The top plot shows the ideal result of processing, but it exceeds the allowable 16-bit range. The middle plot is the result wrapped to 16 bits and has severe distortion. The bottom plot is the result saturated to the allowable range and has mild distortion.

General Optimization Strategies

- Group load and store operations
- Examine the intermediate assembly code
- Enable compiler optimization
- Loop unrolling (Cortex-M4 has a 3-cycle overhead per loop iteration)
- Focus on inner loop
- Inline functions
- Count registers
- Use right amount of precision

p. 694 *The Definitive guide to ARM Cortex-M3 and Cortex-M4 Processors*

DSP Data Types

- Signed integers: 8-, 16-, 32-, 64-bit
- Unsigned integers: 8-, 16-, 32-, 64-bit
- Floating-Point:
  - float32_t (single precision 32-bit)
  - float64_t (double precision 64-bit)
- Fixed-Point/Fractional
  - q7_t (8-bit)
  - q15_t (16-bit)
  - q31_t (32-bit)
  - q63_t (64-bit)

Fixed-Point Numbers

- Recall that fixed point numbers have a fixed number of digits assigned to the integer and fractional parts of a number (as compared to floating point numbers represented in a format like IEEE 754)

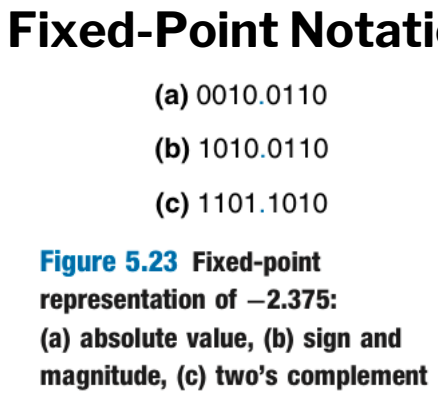


Figure 5.23 Fixed-point representation of −2.375: (a) absolute value, (b) sign and magnitude, (c) two's complement

Figure 5.23 p. 257 from *Digital Design and Computer Architecture: ARMed Edition*

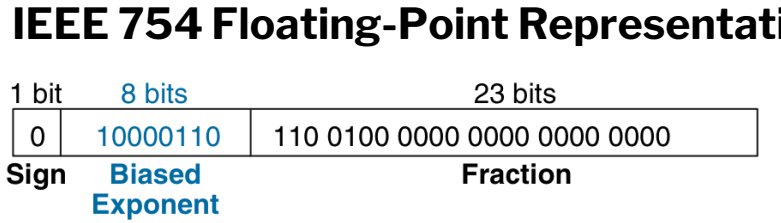


Figure 5.29 IEEE 754 floating-point notation

Figure 5.29 p. 257 from *Digital Design and Computer Architecture: ARMed Edition*

Fixed-Point Numbers

- Fractional arithmetic is a subset of fixed-point
- 8-bit fractional values in range: $\left[\frac{-2^7}{2^7}, \frac{2^7-1}{2^7}\right] = \left[-1, 1-2^{-7}\right]$
- This idea is generalized to the concept of Qm.n numbers where the number has:
  - 1 sign bit
  - m-1 integer bits
  - n fractional bits

Maximum	=	01111111	=	$1-2^{-7}$
Smallest positive	=	00000001	=	2^{-7}
Zero	=	00000000	=	0
Smallest negative	=	11111111	=	-2^{-7}
Minimum	=	10000000	=	-1

p. 683 *The Definitive guide to ARM Cortex-M3 and Cortex-M4 Processor*

Fixed-Point Review

Represent the number -5 in Q4.1 format

$$-2^3\times 1 + 2^2\times 0 + 2^1\times 1 + 2^0\times 1 + 2^{-1}\times 0 = -5$$

11010

Integer bits Fractional bits

What is the decimal equivalent of 0111.101 in Q4.3 representation?

$$-2^3\times 0 + 2^2\times 1 + 2^1\times 1 + 2^0\times 1 + 2^{-1}\times 1 + 2^{-2}\times 0 + 2^{-3}\times 1 = 4 + 2 + 1 + \frac{1}{2} + \frac{1}{8} = 7.625$$

Fractional Number Representation

- What is the decimal value of the q7_t number 10111010?

1 0 1 1 1 0 1 0

$$-2^0\times 1 + 2^{-1}\times 0 + 2^{-2}\times 1 + 2^{-3}\times 1 + 2^{-4}\times 1 + 2^{-5}\times 0 + 2^{-6}\times 1 + 2^{-7}\times 0$$

$$-1 + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{64} = \text{-0.546875}$$

$$\left[S, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\right]$$

Fractional Arithmetic

- Why is the data type q7_t instead of q8_t?
  - A: The MSb is an implied sign bit
- Fractional arithmetic is useful for multiplication since it ensures the result remains in the range [-1, 1).

$$\frac{I_1}{2^{N-1}} \times \frac{I_2}{2^{N-1}} = \frac{I_1 I_2}{2^{2N-2}}$$

$I_1 I_2$ is a standard integer multiplication

Result is 2N-bit number which is in the format Q2.(2N-2)

Fractional Number Multiplication Example

- What is 10010000 x 00000100?

$-2^0 + 2^{-3}$= -0.875 2^{-5}= 0.03125

(-0.875)×(0.03125) = -0.02734375

Treat as normal integer multiplication, then divide by 2^{2N-2} $-448_{10} = 1111111001000000_2$

$$\frac{I_1}{2^{N-1}} \times \frac{I_2}{2^{N-1}} = \frac{(-112)\times 4}{2^{2N-2}} = \frac{(-112)\times 4}{2^{16-2}} = -\frac{448}{2^{14}} = -0.02734375$$

This is in general a Q2.(2N-2) number

11.11111001000000_2

Fractional Number Multiplication Example

- What is 10010000 x 00000100?

$-2^0 + 2^{-3}$= -0.875 2^{-5}= 0.03125

$$(-0.875)\times(0.03125) = -0.02734375$$

Treat as normal integer multiplication, then divide by 2^{2N-2} $-448_{10} = 1111111001000000_2$

$$\frac{I_1}{2^{N-1}} \times \frac{I_2}{2^{N-1}} = \frac{(-112)\times 4}{2^{2N-2}} = \frac{(-112)\times 4}{2^{16-2}} = -\frac{448}{2^{14}} = -0.02734375$$

This is in general a Q2.(2N-2) number

11.1111001000000_2

Finite Impulse Response (FIR) Filters

- FIR benefits compared to IIR
  1. Inherently stable
  2. Linear phase can be accomplished by making the coefficient symmetric
  3. Simple design formulas
  4. Well-behaved even when implemented with fixed-point

Recall: Infinite impulse response filters are recursive; rely on previous output values as well as previous inputs.

FIR Filter with Shift Register

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

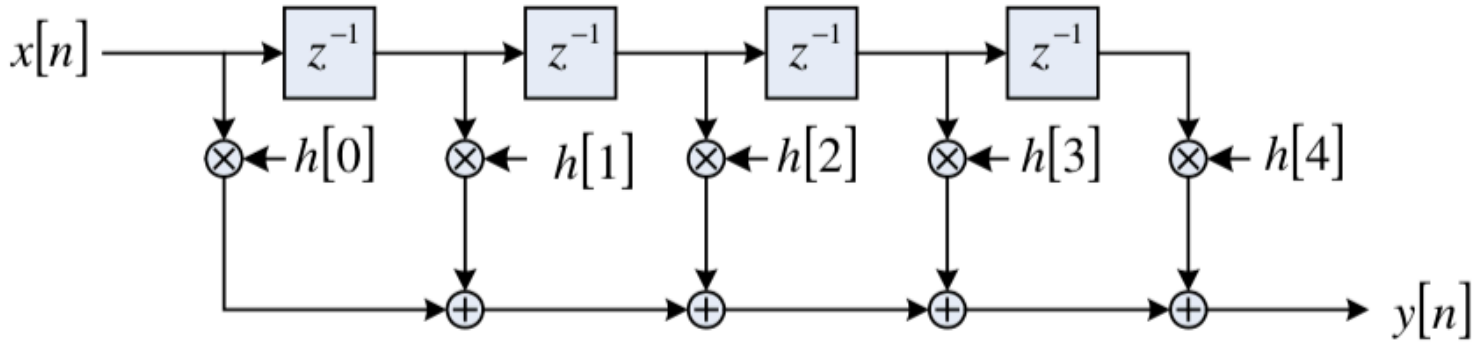


FIGURE 21.11

FIR filter implemented using a shift register. In practice this is rarely used since the state variables have to be shifted right whenever a new sample arrives

Figure 21.11 p. 710 *The Definitive guide to ARM Cortex-M3 and Cortex-M4 Processors*

FIR Filter with Circular Buffer

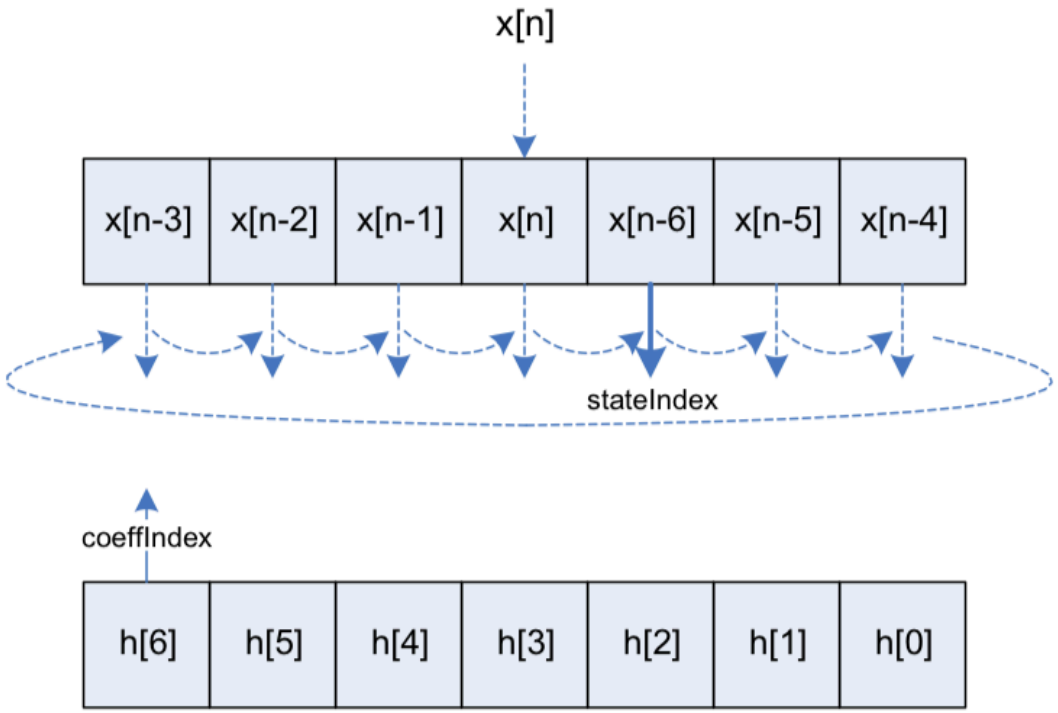
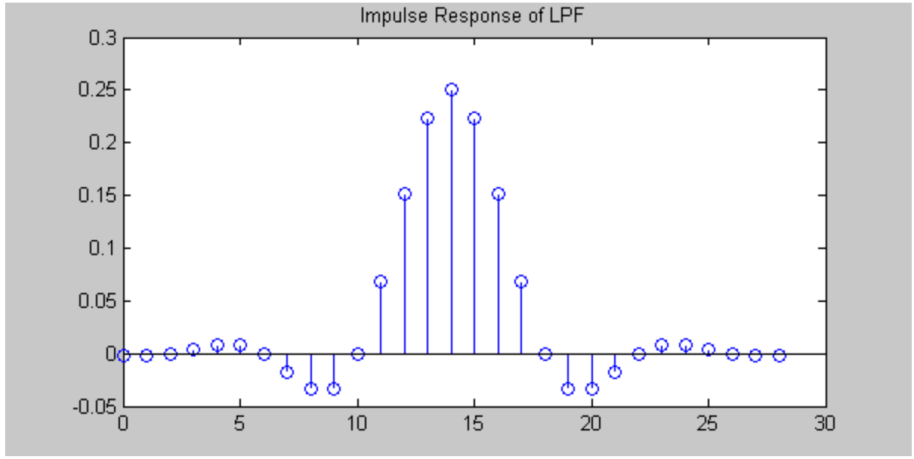


FIGURE 21.12

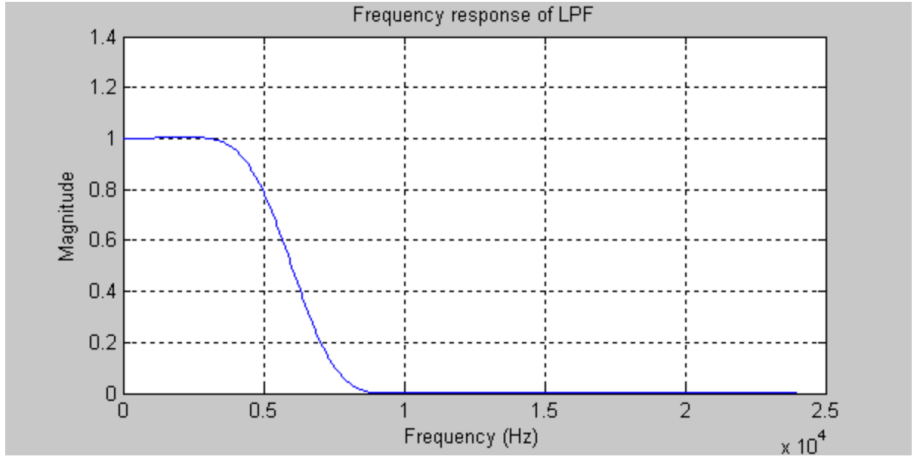
FIR filter implemented using a circular buffer for the state variables (top). The stateIndex pointer advances to the right and then circularly wraps when it reaches the end of the buffer. The coefficients are accessed in linear order

Figure 21.12 p. 710 *The Definitive guide to ARM Cortex-M3 and Cortex-M4 Processors*

FIR Filter Design: CMSIS Demo

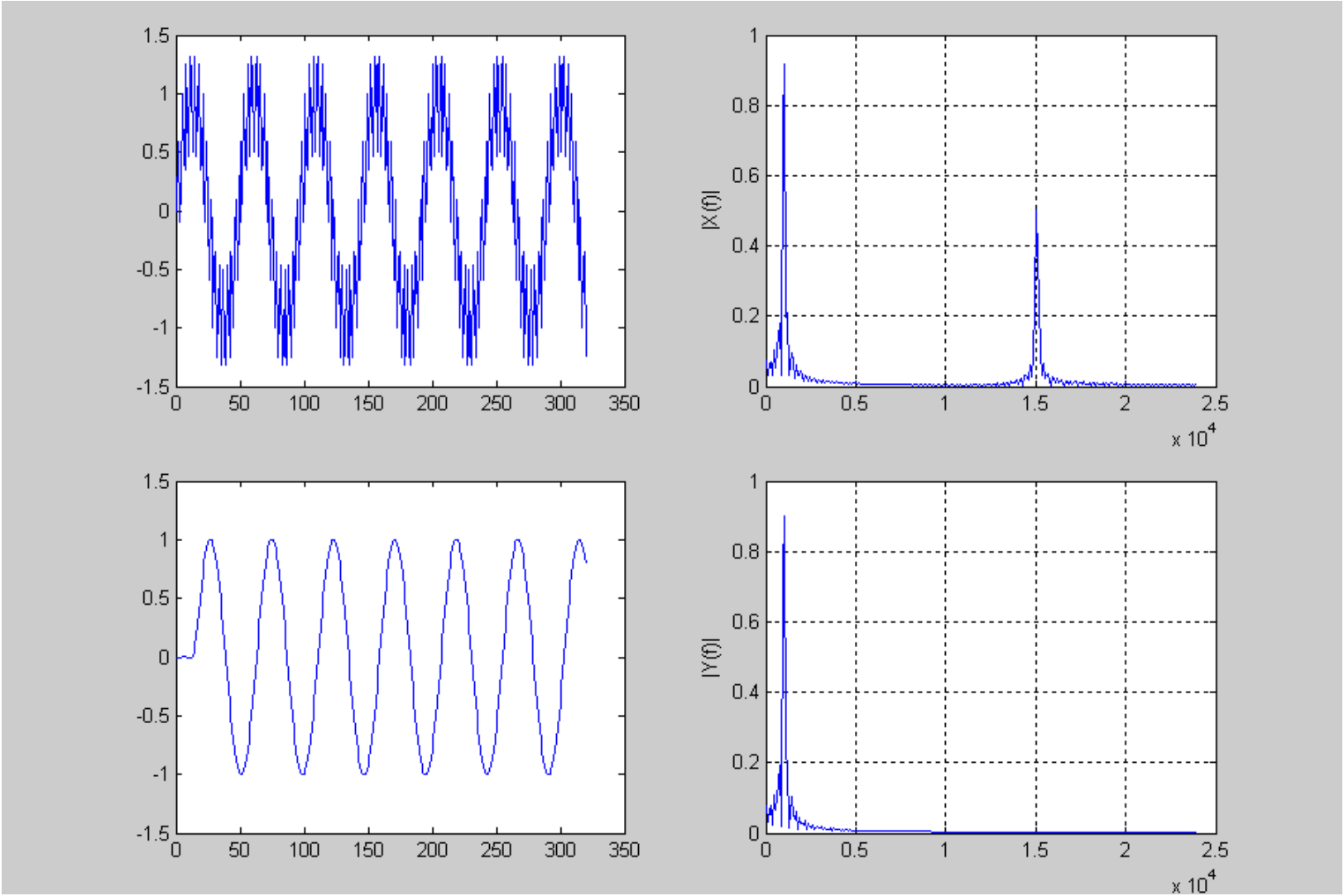


The frequency response of the filter is shown next. The passband gain of the filter is 1.0 and it reaches 0.5 at the cutoff frequency 6 kHz.



https://www.keil.com/pack/doc/CMSIS/DSP/html/group__FIRLPF.html

FIR Filter CMSIS Demo



https://www.keil.com/pack/doc/CMSIS/DSP/html/group__FIRLPF.html

Up Next

- Wednesday: How to Pick an MCU and Board Bring-up
- Lab 7: The Internet of Things
  - No checkoffs this week
  - Final project proposal due date unchanged

Lecture Feedback

- What is the most important thing you learned in class today?
- What point was most unclear from lecture today?

https://forms.gle/Ay6MkpZ6x3xsW2Eb8