Implementation trade-offs of digital FIR filters

By Ed Rocha

U.S. Air Force photo taken by Tech. Sgt. Jason Tudor

High-precision FIR filters are used in myriad medical, military, and high-volume consumer applications. However, given the choice of stand-alone Microcontroller Units (MCUs), DSPs, FPGAs, or dedicated Finite Impulse Response (FIR) ICs, only the latter balances cost, power, size, and precision performance.

FIR filters form the basis of wireless systems in medical devices, industrial control, consumer electronics, and cellular infrastructure. In fact, they are one of the most common types of digital filters. These linear time-invariant style filters rely solely on current and past input samples and not on past outputs, making the resultant signal directly proportional to the number of taps (summation series) and a coefficient set used to calculate the output.

FIR filters can be calculated relatively easily using several different IC implementations, including microcontrollers, general purpose DSP chips, FPGAs, and purpose-built dedicated FIR devices. While each type has advantages and disadvantages, in systems requiring the combination of high precision, low cost, reasonable size, and power, purpose-built FIRs plus low-cost MCUs represent the best of all worlds. Let's examine each option in detail.

Implementation comparison

Several implementation choices are available to designers. Each has its benefits and risks (Table 1). The first implementation is that of a microcontroller with a built-in hardware multiplier and enough data RAM to store data samples to be used in the computation. The major benefits are that this is a small, low-power implementation. It is also highly integrated. The microcontroller not only implements the FIR filter, but can also function as the overall system controller. The disadvantage is that a microcontroller, even after adding a hardware multiplier on-chip, still has very limited computational abilities and is only a viable solution for relatively simple FIR filters at low sample rates. Also, adding resources to a microcontroller like a hardware multiplier and memory significantly increases the cost of the chip.

The second implementation is that of a DSP. The major benefit of using a DSP is that it provides a large amount of computation horsepower and a virtually infinite amount of flexibility in design and algorithm choices. The DSP can also handle medium to large FIR filters up to very high sample frequencies. The downside of using DSPs, though, is that they are large devices that burn a lot of power. They are also relatively expensive and possibly overkill in systems that run at lower sample frequencies.

The third choice is that of an FPGA. The major benefit of an FPGA is the ability to implement algorithms in hardware without losing versatility. FPGAs can support an almost infinite number of algorithmic choices and provide the highest performance of all. The downside of using FPGAs is that they are also large

FIR basis	Pro	Con
Microcontroller (MCU)	SWaP, integration, CPU decision making	Implements only simple FIR, limited computation, cost increases with complexity
Dedicated DSP	Flexibility, nearly limitless horsepower, implements large FIRs and high sample rates	Power hungry, expensive, often overkill in some systems
FPGA	H/W implementation yet still versatile, very flexible algorithms	Large and power hungry, requires skilled IC designers, and cost per chip is high
Purpose-built FIR chip (such as the SavFIRe)	Fast time-to-market, programming ease, low cost	Fixed functionality, requires host controller

Table 1

devices that burn more power than any of the other alternatives. They also require skilled engineers and considerable design time to implement. The cost per chip is the highest of all competing solutions for a single filter such as the one described earlier.

The final implementation choice is a Simple and Versatile FIR Engine or *SavFIRe*. The dedicated FIR chip is the easiest to program and represents the quickest time-to-market of all solutions. It is the least expensive of all the solutions and is by far the smallest and most power-efficient method discussed. The downside of using this part is that it is a dedicated FIR filter and cannot be reused or reconfigured to be anything else. Using SavFIRe requires some sort of host controller to run the system. However, an MCU with filter data storage can cost as low as $1. The combination of an inexpensive, low-power microcontroller along with the SavFIRe chip represents the best total system solution in terms of power, size, time-to-market, and cost.

Example FIR filter for comparison

To make a reasonable comparison between the different platforms for implementing the FIR filter, it helps to have an example. This way we can not only compute the ability of each platform to perform the task, but also estimate comparisons between the platforms' size, power, and cost.

The FIR filter we will design is a notched low-pass filter and has the specifications shown in Figure 1.

- » System sample frequency = 1 KHz
- » Notch center frequency = 60 Hz
- » Top notch BW = 50 Hz
- » Bottom notch BW = 38 Hz
- » Notch attenuation = 60 dB
- » Pass-band upper frequency = 250 Hz
- » Pass-band lower frequency = 260 Hz
- » Pass-band ripple = 0.1 dB
- » Stop-band attenuation = 60 dB

The approximate number of taps for this filter is 467.

Microcontroller implementation

A good example of a microcontroller that can handle a considerable FIR load is Texas Instruments' MSP430F169. It has both a built-in MAC unit and multiple DMA channels to assist in moving data and coefficients to and from data memory. Another significant specification is its 2 KB of memory. This is important because the microcontroller must have enough memory to buffer data samples equal to the number of taps.

Figure 2 shows the data flow of the FIR filter. The steps needed to calculate each output of the FIR filter are as follows:

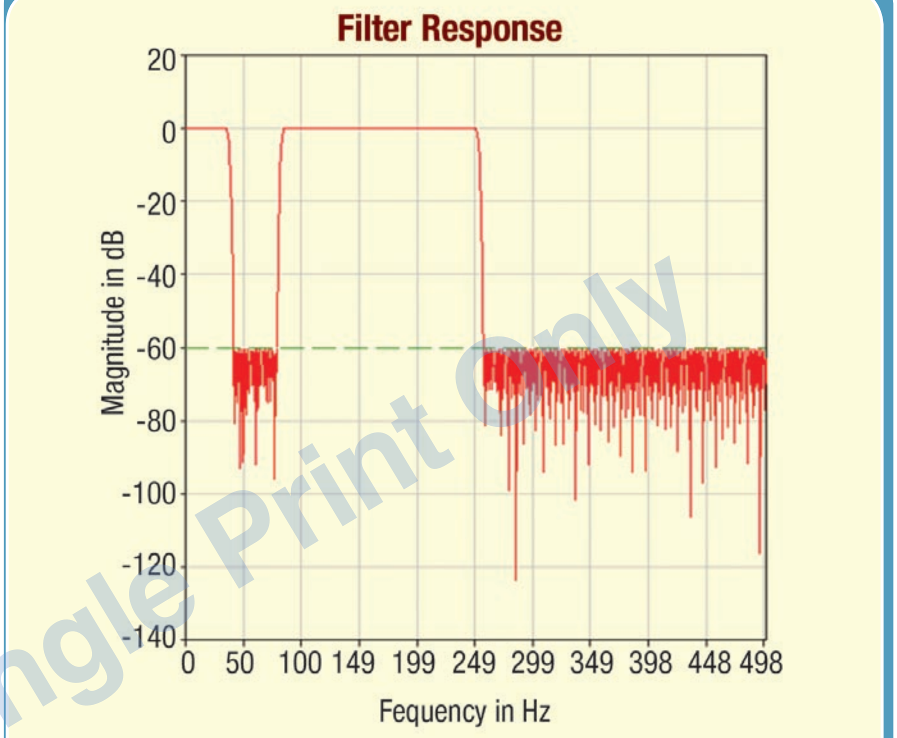

Figure 1

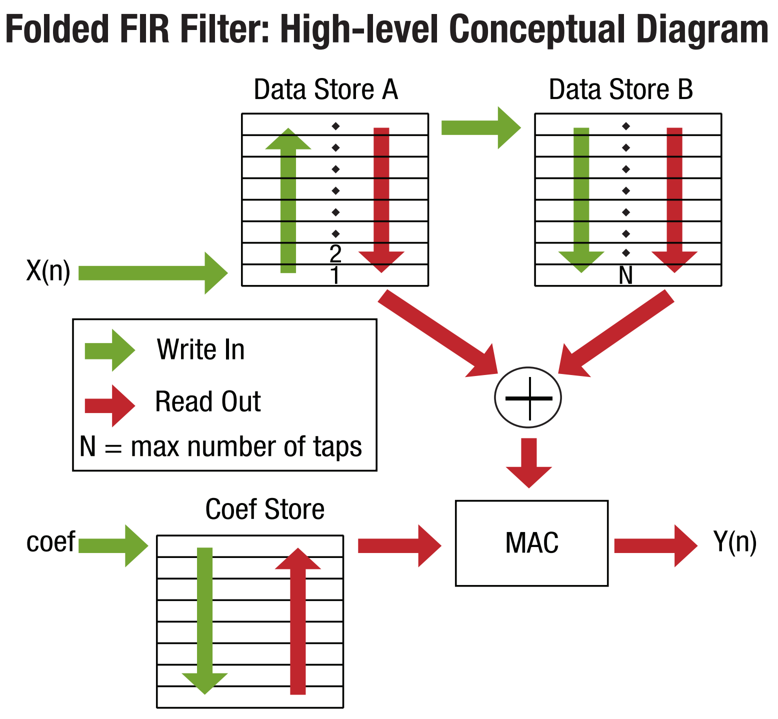

Figure 2

» Get input sample from ADC
» Loop N-1 times: Move two input samples from memory to MAC; move coefficient from memory to MAC; perform MAC operation
» Retrieve output from MAC and store to memory

» Table 2 represents some benchmarks for the MSP430F169. Using the data in the table, we can estimate the number of clock cycles needed to calculate a 467-tap filter. After some interpolation, we come up with about 3,200 clock cycles. This indicates a maximum sample rate of 2,500 Hz. We are only sampling at 1,000 Hz. If we add a bit of overhead to retrieve samples and deliver results to memory or perform any type of analysis on the samples, it is fair to assume that our filter will result in approximately a 50 percent loading on the CPU. Table 2 specifies the benchmarks of the MSP430F169.

Number of Taps, N	Number of CPU Clocks	Max Sampling Rate (KHz) (100% CPU loading)
32	310	25
50	410	19
100	730	11

Table 2

	Compute Power	Flexibility	Size	Power	Cost	Time-to-Market
Micro-controller		✓		✓		
DSP	✓	✓				
FPGA	✓	✓				
SavFIRe	✓		✓	✓	✓	✓

Table 3

The MSP430 family only has a 16 x 16 multiplier in its MAC unit. This means that designers are limited to 16-bit data and 16-bit coefficients if they are to achieve the performance just mentioned. In many systems, both data and coefficients can be larger than 16 bits, which could easily result in a performance degradation of 10x or more.

DSP implementation

The steps involved in FIR calculations are very similar for a DSP compared to a microcontroller. Differences occur in the memory usage where the DSP treats the memory as a circular buffer with pointers that automatically update. Also, because of the high amount of parallelism and pipelining in a DSP, most DSPs can execute the entire MAC operation including fetching and storing data in one clock cycle per MAC operation. There is a slight bit of overhead associated with the beginning and end of the computation loops, but this is negligible for a large number of taps.

Using a Texas Instruments lower-end, fixed-point DSP, the TMS320C55x, let's look at our previous filter design. This DSP is capable of 160 MIPS with one 32 x 32 multiply each clock cycle. This equates to 160M/467 taps per sample = 342 KSps at full speed; this is more than adequate for our example application.

FPGA implementation

The available hardware resources in any given FPGA vary widely depending on the FPGA vendor, part family, and the size of the device chosen.

For the 467-tap FIR in our example, let's use the smallest FPGA in the Cyclone II family from Altera. The EP2C5 has about 4.6 K logic elements, 26 M4K blocks (4 Kb RAMs), and 13 embedded multipliers that are each 18 x 18 bits wide. To implement the 467-tap filter in our example, there are many different architectural choices available using the resources in the FPGA.

The simplest architecture is to use a single multiplier and run it 467/2=234 times for each input sample. The sample rate in this example is only 1 KHz, which implies that our multiplier would have to run at:

1 KHz * 467/2 = 234 KHz

This is easily achievable in today's FPGAs and represents a minimal hardware resource usage.

The multipliers in most low-cost FPGAs are 18 x 18 as mentioned earlier. That means to process larger than 18-bit data or 18-bit coefficients or both, designers will need to use four of these multipliers to create a 36 x 36 multiplier.

Let's assume a typical industry worst case of 24-bit data words and 32-bit coefficients. For a 467-tap filter, there must be enough memory to buffer up 467 data samples and to store 467/2 coefficients. Ignoring some of the implementation details of the FPGA RAM architecture, this implies that a data RAM of 12 Kb and a coefficient RAM of 8 Kb are needed. Translating this into the M4K blocks of the Cyclone II FPGAs would mean a total of five M4K blocks.

SavFIRe implementation

Quickfilter Technologies has developed a SavFIRe chip that is a dedicated FIR filter, supporting up to 512 taps with 32-bit coefficients and anywhere from 12- to 24-bit data words. The QF1D512 supports sample rates from 1 Hz up to 500 KHz. A block diagram of the chip is provided in Figure 3.

The main advantages that come from a dedicated IC all stem from the ability to design a chip specifically for the purpose in mind and optimize all of the associated characteristics of the device. For example, this device is 3 x 3 mm in size; 16 of these will fit in the same footprint

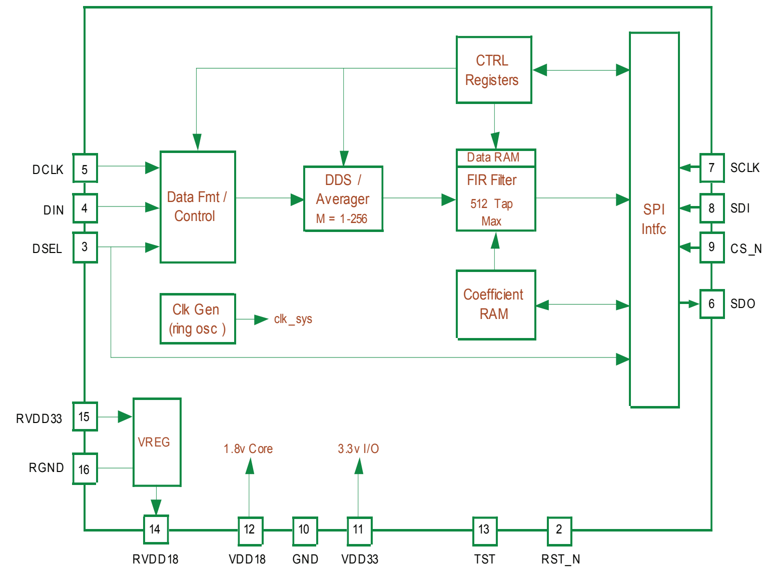

Figure 3

area of an FPGA configured for the same function. Power and cost can also be fully optimized for the function at hand.

The disadvantage of using an IC tailored for a specific function like FIR filters is that it is not very flexible for use in applications outside of FIR filtering. Another requirement is that SavFIRe cannot be used autonomously in a system. There must be some sort of host controller to configure the chip and run sample data through it.

For comparison to the other implementation methods, let's use the same 467-tap FIR example from above. This filter can be created with 32-bit coefficients and use from 12- to 24-bit data samples with the current architecture in SavFIRe. Also, 1 KHz is no problem since SavFIRe works from 1 Hz to 500 KHz.

So what about design effort? There is also a development package, QF1D512-DK, comprising a development board and software development tool. The software tool enables a designer to design a filter by simply specifying the filter parameters. Once the parameters are entered, the software generates all the filter coefficients and necessary register values to get the chip up and running, making design time absolutely minimal. It literally takes minutes for an inexperienced designer to design a high-precision filter.

Another advantage to using SavFIRe is power consumption. The average power consumption for SavFIRe running a 467-tap filter at 1 KHz is about 50 μW. This is orders of magnitude lower than any competing solution. The power scales directly with the sample rate, giving an equivalent power of 0.5 mW at 10 KHz and 5 mW at 100 KHz.

Implementation comparison

We have outlined four popular choices for implementing a high-precision digital FIR filter. All of these implementations have their bright spots as well as their downfalls, which are summarized in Table 3. Again, the SavFIRe choice appears to come out ahead.

Ed Rocha *is a consultant and the owner of Ed Rocha Consulting. He consulted on the architecture and development of Quickfilter's software development tools for both the QF1D512 and QF4A512. He can be contacted at erocha@quickfilter.net.*

Quickfilter Technologies
214-547-0460
www.quickfiltertech.com