Comparision of various Algorithms of Field-Programmable Gate Arrays technology for digital signal processing system

Kuldeep bhardwaj
Ph.d (ECE) scholar
Sai nath university, Ranchi (Jharkhand)

Abstract - Field-Programmable Gate Arrays (FPGAs) are revolutionizing digital signal processing as novel FPGA families are replacing ASICs and PDSPs for front-end digital signal processing algorithms. So the efficient implementation of these algorithms is critical and is the main goal of this paper. It starts with an overview of today's FPGA technology, devices, and tools for designing state-of-the-art DSP systems.

Keywords: DSP, N-tap filter, impulse invariant and time-invariance

I. INTRODUCATION

Digital filtering is one of the most powerful tools of DSP. Apart from the obvious advantages of virtually eliminating errors in the filter associated with passive component fluctuations over time and temperature, op amp drift (active filters), etc., digital filters are capable of performance specifications that would, at best, be extremely difficult, if not impossible, to achieve with an analog implementation. In addition, the characteristics of a digital filter can be easily changed under software control. Therefore, they are widely used in adaptive filtering applications in communications such as echo cancellation in modems, noise cancellation, and speech recognition.

The actual procedure for designing digital filters has the same fundamental elements as that for analog filters. First, the desired filter responses are characterized, and the filter parameters are then calculated. Characteristics such as amplitude and phase response are derived in the same way. The key difference between analog and digital filters is that instead of calculating resistor, capacitor, and inductor values for an analog filter, coefficient values are calculated for a digital filter. So for the digital filter, numbers replace the physical resistor and capacitor components of the analog filter. These numbers reside in a memory as filter coefficients and are used with the sampled data values from the ADC to perform the filter calculations.[1]

TYPES OF DIGITAL FILTERS

- Moving Average
- Finite Impulse Response (FIR)
  - Linear Phase
  - Easy to Design
  - Computationally Intensive
- Infinite Impulse Response (IIR)
  - Based on Classical Analog Filters
  - Computationally Efficient
- Lattice Filters (Can be FIR or IIR)
- Adaptive Filters

II. LITERATURE REVIEW

The real-time digital filter, because it is a discrete time function, works with digitized data as opposed to a continuous waveform, and a new data point is acquired each sampling period. Because of this discrete nature, data

samples are referenced as numbers such as sample 1, sample 2, sample 3, etc. Figure 6.1 shows a low frequency signal containing higher frequency noise which must be filtered out. This waveform must be digitized with an ADC to produce samples x(n). These data values are fed to the digital filter, which in this case is a lowpass filter. The output data samples, y(n), are used to reconstruct an analog waveform using a low glitch DAC.

Digital filters, however, are not the answer to all signal processing filtering requirements. In order to maintain real-time operation, the DSP processor must be able to execute all the steps in the filter routine within one sampling clock period, 1/fs . A fast general purpose fixed-point DSP such as the ADSP-2189M at 75MIPS can execute a complete filter tap multiply-accumulate instruction in 13.3ns. The ADSP-2189M requires N+5 instructions for an N-tap filter. For a 100-tap filter, the total execution time is approximately 1.4µs. This corresponds to a maximum possible sampling frequency of 714kHz, thereby limiting the upper signal bandwidth to a few hundred kHz. [2]

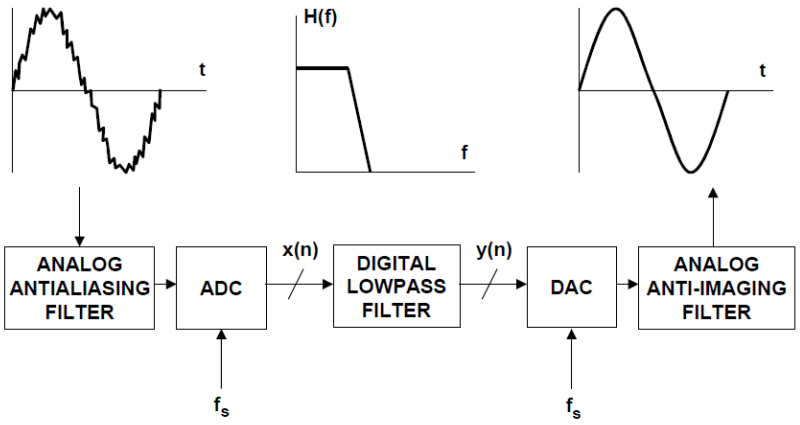

Figure 6.1

However, it is possible to replace a general purpose DSP chip and design special hardware digital filters which will operate at video-speed sampling rates. In other cases, the speed limitations can be overcome by first storing the high speed ADC data in a buffer memory. The buffer memory is then read at a rate which is compatible with the speed of the DSP-based digital filter. In this manner, pseudorealtime operation can be maintained as in a radar system, where signal processing is typically done on bursts of data collected after each transmitted pulse.

Another option is to use a third-party dedicated DSP filter engine like the Systolix PulseDSP™ filter core. The AD7725 16-bit sigma-delta ADC has an on-chip PulseDSP filter which can do 125 million multiply-accumulates per second.

Even in highly oversampled sampled data systems, an analog antialiasing filter is still required ahead of the ADC and a reconstruction (anti-imaging) filter after the DAC. Finally, as signal frequencies increase sufficiently, they surpass the capabilities of available ADCs, and digital filtering then becomes impossible. Active analog filtering is not possible at extremely high frequencies because of op amp bandwidth and distortion limitations, and filtering requirements must then be met using purely passive components. The primary focus of the following discussions will be on filters which can run in real-time under DSP program control. As an example, consider the comparison between an analog and a digital filter shown in Figure 6.3. The cutoff frequency of the both filters is 1kHz. The analog filter is realized as a 6-pole Chebyshev Type 1 filter (ripple in passband, no ripple in stopband). In practice, this filter would probably be realized using three 2-pole stages, each of which requires an op amp, and several resistors and capacitors. The 6-pole design is certainly not trivial, and maintaining the 0.5dB ripple specification requires accurate component selection and matching.[3]

III. METHODOLOGY

A popular method for IIR filter design is to first design the analog-equivalent filter and then mathematically transform the transfer function H(s) into the z-domain, H(z). Multiple pole designs are implemented using cascaded biquad sections. The most popular analog filters are the Butterworth, Chebyshev, Elliptical, and Bessel (see Figure 6.35). There are many CAD programs available to generate the Laplace transform, H(s), for these filters.

The all-pole Butterworth (also called maximally flat) has no ripple in the passband or stopband and has monotonic response in both regions. The all-pole Type 1 Chebyshev filter has a faster rolloff than the Butterworth (for the same number of poles) and has ripple in the passband. The Type 2 Chebyschev filter is rarely used, but has ripple in the stopband rather than the passband. The Elliptical (Cauer) filter has poles and zeros and ripple in both the passband and stopband. This filter has even faster rolloff than the Chebyshev for the same number of poles. The Elliptical filter is often used where degraded phase response can be tolerated.

Finally, the Bessel (Thompson) filter is an all-pole filter which is optimized for pulse response and linear phase but has the poorest rolloff of any of the types discussed for the same number of poles.[4]

REVIEW OF POPULAR ANALOG FILTERS

_ Butterworth
_ All Pole, No Ripples in Passband or Stopband
_ Maximally Flat Response (Fastest Roll-off with No Ripple)
_ Chebyshev (Type 1)
_ All Pole, Ripple in Passband, No Ripple in Stopband
_ Shorter Transition Region than Butterworth for Given Number of Poles
_ Type 2 has Ripple in Stopband, No Ripple in Passband
_ Elliptical (Cauer)
_ Has Poles and Zeros, Ripple in Both Passband and Stopband
_ Shorter Transition Region than Chebyshev for Given Number of Poles
_ Degraded Phase Response
_ Bessel (Thompson)
_ All Pole, No Ripples in Passband or Stopband
_ Optimized for Linear Phase and Pulse Response
_ Longest Transition Region of All for Given Number of Poles

All of the above types of analog filters are covered in the literature, and their Laplace transforms, H(s), are readily available – either from tables or CAD programs. There are three methods used to convert the Laplace transform into the z-transform: impulse invariant transformation, bilinear transformation, and the matched z-transform. The resulting z-transforms can be converted into the coefficients of the IIR biquad. These techniques are highly mathematically intensive and will not be discussed further. A CAD approach for IIR filter design is similar to the Parks-McClellan program used for FIR filters. This technique uses the Fletcher-Powell algorithm. In calculating the throughput time of a particular DSP IIR filter, one should examine the benchmark performance specification for a biquad filter section. For the ADSP-21xx-family, seven instruction cycles are required to execute a biquad filter output sample. For the ADSP-2189M, 75MIPS DSP, this corresponds to $7 \times 13.3\text{ns} = 93\text{ns}$, corresponding to a maximum possible sampling frequency of 10MSPS (neglecting overhead).

IIR FILTER DESIGN TECHNIQUES

_ Impulse Invarient Transformation Method
_ Start with H(s) for Analog Filter
_ Take Inverse Laplace Transform to get Impulse Response
_ Obtain z-Transform H(z) from Sampled Impulse Response
_ z-Transform Yields Filter Coefficients
_ Aliasing Effects Must be Considered

_ Bilinear Transformation Method
_ Another Method for Transforming H(s) into H(z)
_ Performance Determined by the Analog System's
Differential Equation
_ Aliasing Effects do not Occur
_ Matched z-Transform Method
_ Maps H(s) into H(z) for filters with both poles and zeros
_ CAD Methods
_ Fletcher-Powell Algorithm
_ Implements Cascaded Biquad Sections

THROUGHPUT CONSIDERATIONS FOR IIR FILTERS
_ Determine How Many Biquad Sections (N) are Required to
Realize the Desired Frequency Response
_ Multiply this by the number of instruction cycles per Biquad
for the DSP and add overhead cycles (5N + 2 cycles for the
ADSP-21xx series, for example).
_ The Result (plus overhead) is the Minimum Allowable
Sampling Period (1 / fs) for Real-Time Operation
Choosing between FIR and IIR filter designs can be somewhat of a challenge, but a few basic guidelines can be given. Typically, IIR filters are more efficient than FIR filters because they require less memory and fewer multiply-accumulates are needed. IIR filters can be designed based upon previous experience with analog filter designs. IIR filters may exhibit instability problems, but this is much less likely to occur if higher order filters are designed by cascading second-order systems.

On the other hand, FIR filters require more taps and multiply-accumulates for a given cutoff frequency response, but have linear phase characteristics. Since FIR filters operate on a finite history of data, if some data is corrupted (ADC sparkle codes, for example) the FIR filter will ring for only N–1 samples. Because of the feedback, however, an IIR filter will ring for a considerably longer period of time. If sharp cutoff filters are needed and processing time is at a premium, IIR elliptic filters are a good choice. If the number of multiply/accumulates is not prohibitive, and linear phase is a requirement, then the FIR should be chosen.

COMPARISON BETWEEN FIR AND IIR FILTERS

IIR FILTERS	FIR FILTERS
More Efficient	Less Efficient
Analog Equivalent	No Analog Equivalent
May Be Unstable	Always Stable
Non-Linear Phase Response	Linear Phase Response
More Ringing on Glitches	Less Ringing on Glitches
CAD Design Packages Available	CAD Design Packages Available
No Efficiency Gained by Decimation	Decimation Increases Efficiency

Figure 6.38

There are two fundamental types of digital filters: finite impulse response (FIR) and infinite impulse response (IIR). As the terminology suggests, these classifications refer to the filter's impulse response. By varying the weight of the coefficients and the number of filter taps, virtually any frequency response characteristic can be realized with an FIR filter. As has been shown, FIR filters can achieve performance levels which are not possible with analog filter techniques (such as perfect linear phase response). However, high performance FIR filters generally require a large

number of multiply-accumulates and therefore require fast and efficient DSPs. On the other hand, IIR filters tend to mimic the performance of traditional analog filters and make use of feedback. Therefore their impulse response extends over an infinite period of time. Because of feedback, IIR filters can be implemented with fewer coefficients than for an FIR filter. Lattice filters are simply another way to implement either FIR or IIR filters and are often used in speech processing applications. Finally, digital filters lend themselves to adaptive filtering applications simply because of the speed and ease with which the filter characteristics can be changed by varying the filter coefficients[4

IV. FINDINGS

The most elementary form of an FIR filter is a moving average filter as shown in Figure 6.6. Moving average filters are popular for smoothing data, such as in the analysis of stock prices, etc. The input samples, x(n) are passed through a series of buffer registers (labeled z–1, corresponding to the z-transform representation of a delay element). In the example shown, there are four taps corresponding to a fourpoint moving average. Each sample is multiplied by 0.25, and these results are added to yield the final moving average output y(n). The figure also shows the general equation of a moving average filter with N taps. Note again that N refers to the number of filter taps, and not the ADC or DAC resolution as in previous sections.

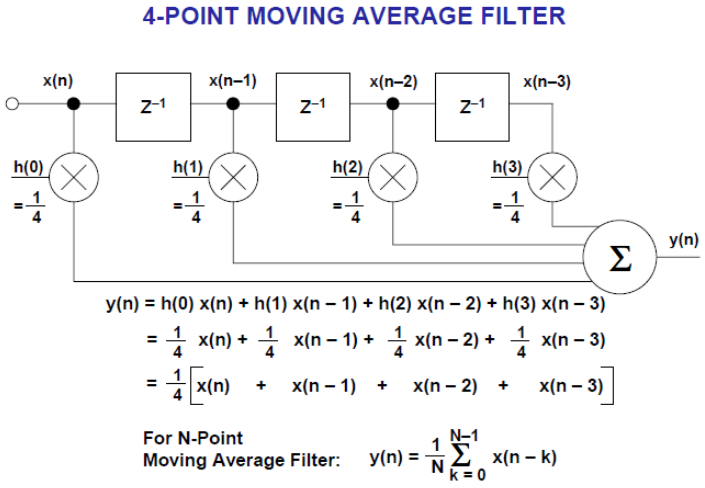

Figure 6.6

Since the coefficients are equal, an easier way to perform a moving average filter is shown in Figure 6.7. Note that the first step is store the first four samples, x(0), x(1), x(2), x(3) in a register. These quantities are added and then multiplied by 0.25 to yield the first output, y(3). Note that the initial outputs y(0), y(1), and y(2) are not valid because all registers are not full until sample x(3) is received. When sample x(4) is received, it is added to the result, and sample x(0) is subtracted from the result. The new result must then be multiplied by 0.25. Therefore, the calculations required to produce a new output consist of one addition, one subtraction, and one multiplication, regardless of the length of the moving average filter.

The step function response of a 4-point moving average filter is shown in Figure 6.8. Notice that the moving average filter has no overshoot. This makes it useful in signal processing applications where random white noise must be filtered but pulse response preserved. Of all the possible linear filters that could be used, the moving average produces the lowest noise for a given edge sharpness. This is illustrated in Figure 6.9, where the noise level becomes lower as the number of taps are increased. Notice that the 0% to 100% risetime of the pulse response is equal to the total number of taps in the filter multiplied by the sampling period.

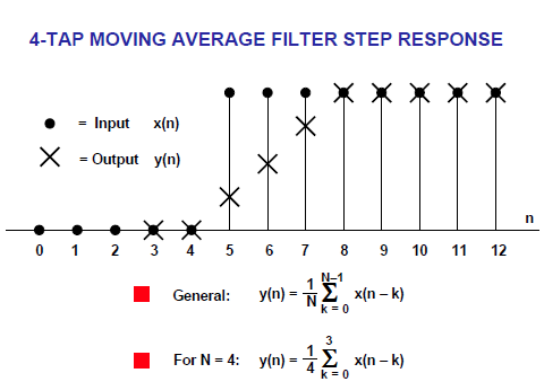

Figure 6.8

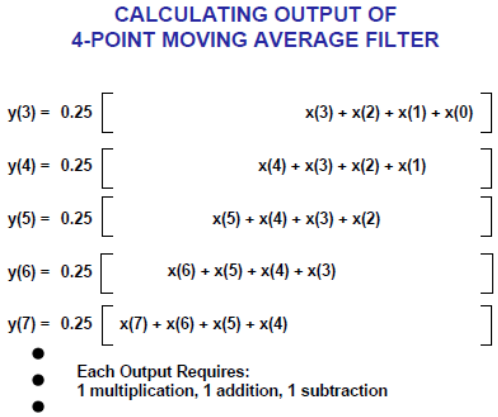

Figure 6.7

V. CONCLUSION

An introduction to digital filters has been presented. The main utility of the analysis methods presented is in ascertaining how a given filter will affect the spectrum of a signal passing through it. Some of the concepts introduced were linearity, time-invariance, filter impulse response, difference equations, transient response, steady-state response, transfer functions, amplitude response, phase response, phase delay, group delay, linear phase, minimum phase, maximum phase, poles and zeros, filter stability, and the general use of complex numbers to represent signals, spectra, and filters. Additionally, practical filtering in matlab has been discussed.

However, this is still only the beginning. With these foundations there is an unlimited number of avenues of investigation into applications of digital filters.

REFERENCES

[1] Steven W. Smith, **The Scientist and Engineer's Guide to Digital Signal Processing**, Second Edition, 1999, California Technical Publishing, P.O. Box 502407, San Diego, CA 92150. Also available for free download at: http://www.dspguide.com or http://www.analog.com
[2] C. Britton Rorabaugh, **DSP Primer**, McGraw-Hill, 1999. 3. Richard J. Higgins, **Digital Signal Processing in VLSI,** Prentice-Hall, 1990.
[3] A. V. Oppenheim and R. W. Schafer, **Digital Signal Processing**, Prentice- Hall, 1975.
[4] L. R. Rabiner and B. Gold, **Theory and Application of Digital Signal Processing**, Prentice-Hall, 1975.
[5] John G. Proakis and Dimitris G. Manolakis, **Introduction to Digital Signal Processing**, MacMillian, 1988.