International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.5, Sep-Oct. 2012 pp-3437-3441 ISSN: 2249-6645

FPGA Implementation of a New Parallel FIR Filter Structures

D. Suresh, K. Ramadevi, K. Raguram
D.suresh (E.C.E), Associate professor, Associate professor, Pragati engineering college Surampalem, East Godavari Dist ,A.P.

ABSTRACT **:** *In recent days filters with large lengths are started to use. So parallel processing is essential at any cost.In this paper proposes new parallel FIR filter structures, which are beneficial to symmetric coefficients in terms of the hardware cost, under the condition that the number of taps is a multiple of 2 or 3. The proposed parallel FIR structures use symmetric property to reducing half the number of multipliers in sub filter section at the expense of additional adders in preprocessing and post processing blocks. Exchanging multipliers with adders is advantageous because adders weigh less than multipliers in terms of silicon area; in addition, the overhead from the additional adders in preprocessing and post processing blocks stay fixed and do not increase along with the length of the FIR filter, whereas the number of reduced multipliers increases along with the length of the FIR filter. Parallel FIR filter is essential, especially when the length of the filter is large.*

Key words: *Parallel FIR, preprocessing*

I. INTRODUCTION

Finite impulse response (FIR) filters are the most popular type of filters implemented in software. This introduction will help you understand them both on a theoretical and a practical level. Filters are signal conditioners. Each functions by accepting an input signal, blocking pre-specified frequency components, and passing the original signal minus those components to the output. In a typical digital filtering application, software running on a digital signal processor (DSP) reads input samples from an A/D converter, performs the mathematical manipulations dictated by theory for the required filter type, and outputs the result via a D/A converter.

Some applications need the FIR filter to operate at high frequencies such as video processing, whereas some other applications request high throughput with a low-power circuit such as multiple-input multiple-output (MIMO) systems used in cellular wireless communication. Furthermore, when narrow transition- band characteristics are required, the much higher order in the FIR filter is unavoidable. For example, a 576-tap digital filter is used in a video ghost canceller for broadcast television, which reduces the effect of multipath signal echoes.

II. Finite Impulse Response

Filters can be classified in several different groups, depending on what criteria are used for classification. The two major types of digital filters are finite impulse response digital filters (FIR filters) and infinite impulse response digital filters (IIR).

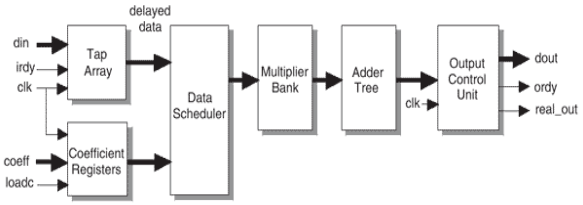


Figure.1 Digital filtering

Both types have some advantages and disadvantages that should be carefully considered when designing a filter. Besides, it is necessary to take into account all fundamental characteristics of a signal to be filtered as these are very important when deciding which filter to use. In most cases, it is only one characteristic that really matters and it is whether it is necessary that filter has linear phase characteristic or not.

Speech signal, for example, can be processed in the systems with non-linear phase characteristic. The phase characteristic of a speech signal is not of the essence and as such can be neglected, which results in the possibility to use much wider range of systems for its processing.

.

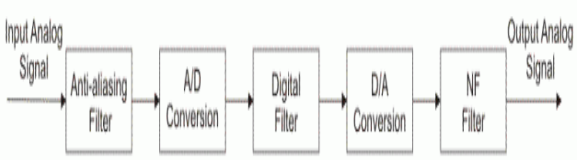


Figure 2. Digital filtering

The process of selecting the filter's length and coefficients is called filter design. The goal is to set those parameters such that certain desired stop band and pass band parameters will result from running the filter. Most engineers utilize a program such as MATLAB to do their filter design. But whatever tool is used, the results of the design effort should be the same:

A frequency response plot, like the one shown in Figure 1, which verifies that the filter meets the desired specifications, including ripple and transition bandwidth. The longer the filter (more taps), the more finely the response can be tuned With the length, N, and coefficients, float h[N] = { ... }, decided upon, the implementation of the FIR filter is fairly straightforward. Listing 1 shows how it could be done in C. Running this code on a processor with a multiply-and-accumulate instruction (and a compiler that knows how to use it) is essential to achieving a large number of taps.

A. Ideal low-pass filter

FIR filters are digital filters with finite impulse response. They are also known as non-recursive digital filters as they do not have the feedback (a recursive part of a filter), even though recursive algorithms can be used for FIR filter realization

B. Window Method for FIR Filter Design

The window method for digital filter design is fast, convenient, and robust, but generally suboptimal. It is easily understood in terms of the convolution theorem for Fourier transforms, making it instructive to study after the Fourier theorems and windows for spectrum analysis.

$$[-N, N]$$

We would expect to be able to truncate it to the interval, for some sufficiently large N, and obtain a pretty good FIR filter which approximates the ideal filter. This would be an example of using the window method with the rectangular window. We saw in §4.3 that such a choice is optimal in the least-squares sense, but it designs relatively poor audio filters. Choosing other windows corresponds to tapering the ideal impulse response to zero instead of truncating it. Tapering better preserves the shape of the desired frequency response, as we will see. By choosing the window carefully, we can manage various trade-offs so as to maximize the filter-design quality in a given application. Window functions are always time limited. The window method always designs a finite-impulse-response (FIR) digital filter (as opposed to an infinite-impulse-response (IIR) digital filter). By the dual of the convolution theorem, point wise multiplication in the time domain corresponds to convolution in the frequency domain.

C. FIR And IIR Digital Filter Design

Based on combining ever increasing computer processing speed with higher sample rate processors, Digital Signal Processors (DSP's) continue to receive a great deal of attention in technical literature and new product design. The following section on digital filter design reflects the importance of understanding and utilizing this technology to provide precision stand alone digital or integrated analog/digital product solutions. By utilizing DSP's capable of sequencing and reproducing hundreds to thousands of discrete elements, design models can simulate large hardware structures at relatively low cost. DSP techniques can perform functions such as Fast-Fourier Transforms (FFT), delay equalization, programmable gain, modulation, encoding/decoding, and filtering.

• Filter weighting functions (coefficients) can be calculated on the fly, reducing memory requirements

• Algorithms can be dynamically modified as a function of signal input.

DSP represents a subset of signal-processing activities that utilize A/D converters to turn analog signals into streams of digital data. A stand-alone digital filter requires an A/D converter (with associated anti-alias filter), a DSP chip and a PROM or software driver. An extensive sequence of multiplication's and additions can then be performed on the digital data. In some applications, the designer may also want to place a D/A converter, accompanied by a reconstruction filter, on the output of the DSP to create an analog equivalent signal. A digital filter solution offering a 90 dB attenuation floor and a 20 kHz bandwidth can consist of up to 10 circuits occupying several square inches of circuit-board space and costing hundreds of dollars.

Digital filters process digitized or sampled signals. A digital filter computes a quantized time-domain representation of the convolution of the sampled input time function and a representation of the weighting function of the filter. They are realized by an extended sequence of multiplications and additions carried out at a uniformly spaced sample interval. Simply said, the digitized input signal is mathematically influenced by the DSP program. These signals are passed through structures that shift the clocked data into summers (adders), delay blocks and multipliers. These structures change the mathematical values in a predetermined way; the resulting data represents the filtered or transformed signal. It is important to note that distortion and noise can be introduced into digital filters simply by the conversion of analog signals into digital data, also by the digital filtering process itself and lastly by conversion of processed data back into analog.

When fixed-point processing is used, additional noise and distortion may be added during the filtering process because the filter consists of large numbers of multiplications and additions, which produce errors, creating truncation noise. Increasing the bit resolution beyond 16-bits will reduce this filter noise.

Instead of using a commercial DSP with software algorithms, a digital hardware filter can also be constructed from logic elements such as registers and gates, or an integrated hardware block such as an FPGA (Field Programmable Gate Array). Digital hardware filters are desirable for high bandwidth applications; the trade-offs are limited design flexibility and higher cost.

(1) Fixed-Point DSP and FIR (Finite Impulse Response) Implementations: Fixed-Point DSP processors account for a majority of the DSP applications because of their smaller size and lower cost. The Fixed-Point math requires programmers to pay significant attention to the number of coefficients utilized in each algorithm when multiplying and accumulating digital data to prevent distortion caused by register overflow and a decrease of the signal-to-noise ratio caused by truncation noise. The structure of these algorithms uses a repetitive delay-and-add format that can be represented as "DIRECT FORM-I STRUCTURE",

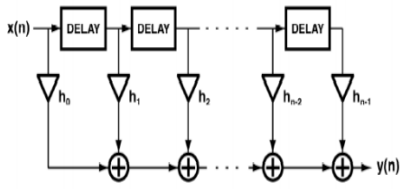


Figure 3 Transposed direct form FIR Filter

FIR (Finite Impulse Response) filters are implemented using a finite number "n" delay taps on a delay line and "n" computation coefficients to compute the algorithm (filter) function. The above structure is non-recursive, a repetitive delay-and-add format, and is most often used to produce FIR filters. This structure depends upon each sample of new and present value data. FIR filters can create transfer function that have no equivalent in linear circuit technology.

***III. Window Technique*:**

The simplest technique is known as "Windowed" filters. This technique is based on designing a filter using well-known frequency domain transition functions called "windows". The use of windows often involves a choice of the lesser of two evils. Some windows, such as the Rectangular, yield fast roll-off in the frequency domain, but have limited attenuation in the stop-band along with poor group delay characteristics. Other windows like the Blackman, have better stop-band attenuation and group delay, but have a wide transition-band (the band-width between the corner frequency and the frequency attenuation floor). Windowed filters are easy to use, are scalable (give the same results no matter what the corner frequency is) and can be computed on-the-fly by the DSP.

IV. The Equiripple Technique

An Equiripple or Remez Exchange (Parks-McClellan) design technique provides an alternative to windowing by allowing the designer to achieve the desired frequency response with the fewest number of coefficients. This is achieved by an iterative process of comparing a selected coefficient set to the actual frequency response specified until the solution is obtained that requires the fewest number of coefficients. Though the efficiency of this technique is obviously very desirable, there are some concerns.

- For equiripple algorithms some values may converge to a false result or not converge at all. Therefore, all coefficient sets must be pre-tested off-line for every corner frequency value.
- Application specific solutions (programs) that require signal tracking or dynamically changing performance parameters are typically better suited for windowing since convergence is not a concern with windowing.
- Equiripple designs are based on optimization theory and require an enormous amount of computation effort. With the availability of today's desktop computers, the computational intensity requirement is not a problem, but combined with the possibility of convergence failure; equiripple filters typically cannot be designed on-the-fly within the DSP.

Analog filters beyond 10 poles are very difficult to realize and tend to be noisy

V. Digital to Analog Conversion (D/A)

As with input signals to A/D converters, waveforms created by D/A converters also exhibit errors. For each input digital data point, the D/A holds the corresponding value until the next sample period. Therefore, the output waveform exists as a sequence of steps. This output, a kind of "sample-and-hold" – is known as a "first-order hold." In non-reconfigurable filters, these coefficients are constant and shift operation is done by hardwiring. The long tree of adders in multiplier implementation increases switching activity and physical capacitance and then power consumption.

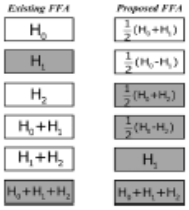


Fig.4 Implementation of coefficient

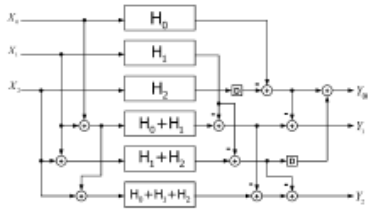


Fig.5 Parallel FIR filter architecture

VI. Proposed Reconfigurable Fir Filter Architecture

To utilize the symmetry of coefficients, the main idea behind the proposed structures is actually pretty intuitive, to manipulate the polyphase decomposition to earn as many subfilter blocks as possible which contain symmetric coefficients so that half the number of multiplications in the single subfilter block can be reused for the multiplications of whole taps, which is similar to the fact that a set of symmetric coefficients would only require half the filter length of multiplications in a single FIR filter. Therefore, for an N-tap 4-parallel FIR filter the total amount of saved multipliers would be the number of subfilter blocks that contain symmetric coefficients times half the number of multiplications in a single subfilter block decomposition to earn as many subfilter blocks as possible which contain symmetric coefficients so that half the number of multiplications in the single subfilter block can be reused for the multiplications of whole taps, which is similar to the fact that a set of symmetric coefficients would only require half the filter length of multiplications in a single FIR filter. Therefore, for an N-tap 3-parallel FIR filter the total amount of saved multipliers would be the number of subfilter blocks that contain symmetric coefficients times half the number of multiplications in a single subfilter block . As can be seen from the example above, two of three subfilter blocks from the proposed two-parallel FIR filter structure,H0+H1 and H0-H1, are with symmetric coefficients now, as (8), which means the subfilter block can be realized by Fig. 4, with only half the amount of multipliers required. Each output of multipliers responds to two taps. Note that the transposed direct-form FIR filter is employed. Compared to
the existing FFA two-parallel FIR filter structure, the proposed FFA structure leads to one more subfilter block which contains symmetric coefficients. However, it comes with the price of the increase of amount of adders in preprocessing and postprocessing blocks. In this case, two additional adders are required for L==2.Add/Sub control block. This block uses the sign bit of each sub-coefficient, and control the add/sub block. To implement the multiplication by zero for each subcoefficient, the multiplexer blocks are followed by AND gates, which is controlled by Mux control block. Three full add/sub bocks are used to combine the partial products of subcoefficients.

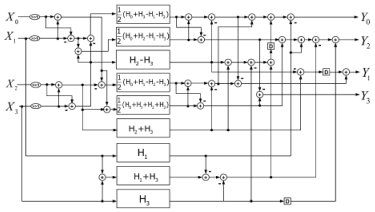


Figure. 6 Proposed parallel FIR filter architecture using four input.

VII. IMPLEMENTATION OF ALGORITHM

A primary objective of this project was to develop a synthesizable model for the AES128 encryption algorithm. Synthesis is the process of converting the register transfer level (RTL) representation of a design into an optimized gate-level netlist. This is a major step in ASIC design flow that takes an RTL model closer to a low-level hardware implementation.

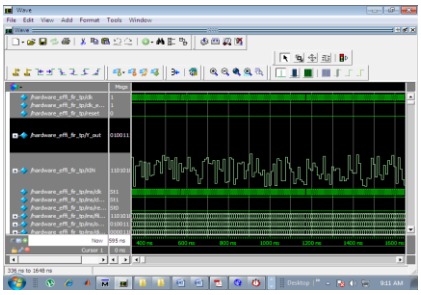


Figure7 .Simulated output.

A. Synthesis Timing Result

The synthesis tool optimizes the combinational paths in a design. In General, four types of combinational paths can exist in any design: [3]

1- Input port of the design under test to input of one internal flip-flip
2- Output of an internal flip-flip to input of another flip-flip
3- Output of an internal flip-flip to output port of the design under test
4- A combinational path connecting the input and output ports of the design under test

The last DC command in the script developed in previous section, instructs the tool to report the path with the worst timing. In this case, the path with the worst timing is a combinational path of type two. The delay associated with this path is the summation of delays of all combinational gates in the path plus the *Clock-To-Q* delay of the originating flip-flop, which was calculated as 24.09ns.

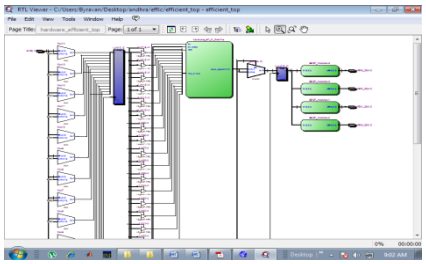
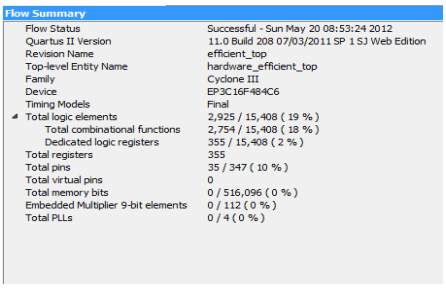


Figure 8.RTL Schematic report

By considering the setup time of the destination flip-flop in this path, which is 0.85ns, the 40MHz clock signal satisfies the worst combinational path delay. The delays of combinational gates, setup time of flip-flops and *Clock-To-Q* values are derived from the LSI_10k library file that was used for the mapping step during synthesis

B. Synthesis Area Result

The synthesis area report shows the total number of cells and nets in the netlist. It also uses the area parameter associated with each cell in the LSI_10K library file, to calculate the total combinational and sequential area of the netlist. The total area of the gate level netlist is unknown since it depends on total area of the interconnects, which itself is a function of the wiring load model used in physical design. The total cell area in the netlist is reported as 22978 units, which is the sum of combinational and sequential areas.



Flow Summary

Flow Status	Successful - Sun May 20 08:53:24 2012
Quartus II Version	11.0 Build 208 07/03/2011 SP 1 SJ Web Edition
Revision Name	efficient_top
Top-level Entity Name	hardware_efficient_top
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	2,925 / 15,408 (19 %)
Total combinational functions	2,754 / 15,408 (18 %)
Dedicated logic registers	355 / 15,408 (2 %)
Total registers	355
Total pins	35 / 347 (10 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	0 / 112 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 9.Flow summary report

To enforce the synthesis tool to create the most compact netlist, the area of the gate level netlist was constrained to zero during the synthesis process. As a result, the only constraint violation, which is expected, is related to the area as shown bellow:

C. Performance Report

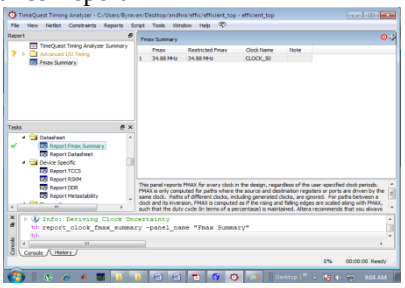


Figure10 .Fmax. summary report for slow corner.

VIII. CONCLUSION

The proposed new structure exploits the nature of even symmetric coefficients and save a significant amount of multipliers at the expense of additional adders. Since multipliers outweigh adders in hardware cost, it is profitable to exchange multipliers with adders. Moreover, the number of increased adders stays still when the length of FIR filter becomes large, whereas the number of reduced multipliers increases along with the length of FIR filter. Consequently, the larger the length of FIR filters is, the more the proposed structures can save from the existing FFA structures, with respect to the hardware cost. Overall this paper proved that for larger filter length area consumption of proposed filter is far better than any other existing method.

REFERENCES

[1] D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implementations," *J. VLSI Signal Process. Syst.*, vol. 17, no. 1, pp. 75–92, 1997.

[2] J. G. Chung and K. K. Parhi, "Frequency-spectrum-based low-area low-power parallel FIR filter design," *EURASIP J. Appl. Signal Process.*, vol. 2002, no. 9, pp. 444–453, 2002.

[3] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999.

[4] Z.-J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. Signal Process.*, vol. 39, no. 6, pp. 1322–1332, Jun. 1991.

[5] J. I. Acha, "Computational structures for fast implementation of L-path and L-block digital filters," *IEEE Trans. Circuit Syst.*, vol. 36, no. 6, pp. 805–812, Jun. 1989.

[6] C. Cheng and K. K. Parhi, "Hardware efficient fast parallel FIR filter structures based on iterated short convolution," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 8, pp. 1492–1500, Aug. 2004.

[7] C. Cheng and K. K. Parhi, "Furthur complexity reduction of parallel FIR filters," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2005)*, Kobe, Japan, May 2005.

[8] C. Cheng and K. K. Parhi, "Low-cost parallel FIR structures with - stage parallelism," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 2, pp. 280–290, Feb. 2007.

[9] I.-S. Lin and S. K. Mitra, "Overlapped block digital filtering," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 8, pp. 586–596, Aug. 1996.

[10] *"Design Compiler User Guide,"* ver. B-2008.09, Synopsys Inc., Sep. 2008.