International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 7, September - 2012

# DESIGN & FPGA IMPLEMENTATION OF RECONFIGURABLE FIR FILTER ARCHITECTURE FOR DSP APPLICATIONS

MAHESH BABU KETHA*, CH.VENKATESWARLU ** KANTIPUDI RAGHURAM**
*ECE Department Pragati Engineering*
*College, Surampalem, India*

## Abstract

*Multi standard wireless communication systems require the reconfigurable FIR filters with low complexity architectures. The complexity of FIR filters is dominated by the coefficient multipliers. A new hardware efficient reconfigurable FIR filter architecture is proposed in this paper based on the proposed binary signed sub coefficient method. Using the proposed coefficient representation method, the hardware requirements for multiplexer units are reduced dramatically with respect to typical methods. ALTERA QUARTUS II synthesis results of the designed filter architecture show 39% area reduction in the resources usage and 15% power reduction over previously reported two state of the art reconfigurable architectures.*

***Keywords***: Finite-impulse response digital filters, Reconfigurability

## 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 one 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.

An analog filter, by contrast, operates directly on the analog inputs and is built entirely with analog components, such as resistors, capacitors, and inductors. There are many filter types, but the most common are low pass, high pass, band pass, and band stop. A low pass filter allows only low frequency signals (below some specified cut-off) through to its output, so it can be used to eliminate high frequencies. A low pass filter is handy, in that regard, for limiting the uppermost range of frequencies in an audio signal; it's the type of filter that a phone line resembles. A high pass filter does just the opposite, by rejecting only frequency components below some threshold.

## II. Finite Impulse Response

A finite impulse response (FIR) filter is a filter structure that can be used to implement almost any sort of frequency response digitally. An FIR filter is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. Figure 1 shows the basic block diagram for an FIR filter of length N. The $h_k$ values are the coefficients used for multiplication, so that the output at time n is the summation of all the delayed samples multiplied by the appropriate coefficients.

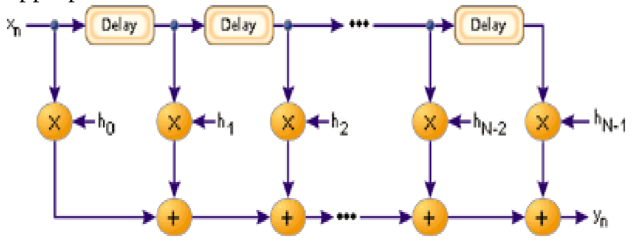


Figure1. The logical structure of an FIR filter

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 show 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. Low pass Filter Design Specifications*

Typical design parameters for a low pass filter are shown in Fig.2

The pass-band ripple is typically larger than the stop-band ripple because it is a deviation about 1 instead of 0. In summary, the pass-band ripple is an allowed gain deviation, while the stop-band ripple is an allowed ``leakage" level.

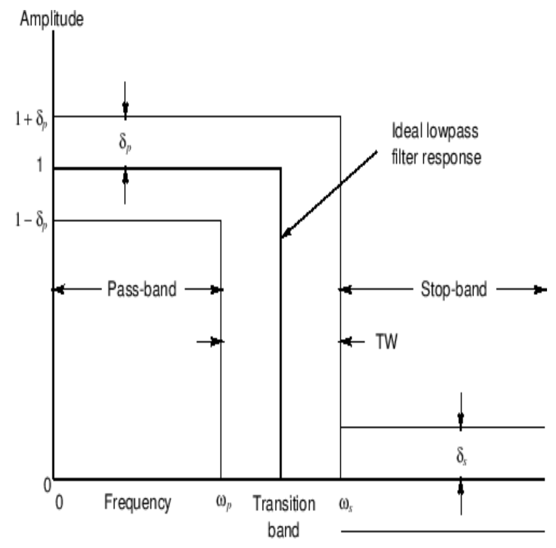


Figure 2. Illustrations of typical low pass filter specification in the frequency domain.

In terms of these specifications, we may define an optimal FIR low pass filter of a given length to be one which minimizes the stop-band and pass-band ripple (weighted relatively as desired) for given stop-band and pass-band edge frequencies. Such optimal filters are often designed in practice by Chebyshev methods, as we encountered already in the study of windows for spectrum analysis (§3.10,§3.13). Optimal Chebyshev FIR filters will be discussed further below (in §4.5.2), but first we look at simpler FIR design methods and compare to optimal Chebyshev designs for reference. An advantage of the simpler methods is that they are more suitable for interactive, real-time, and/or signal-adaptive FIR filter design.

## III. 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. It can be effectively combined with the frequency sampling method, as we will see in §4.6below. 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. 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. This means there is always a finite integer is thus always time-limited, as needed for practical implementation. 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.

## IV. PROPOSED SYSTEM

The proposed coefficient representation technique uses pipelining technique in order to increase the performance and we also focus on increase the filter order i.e filter order to save more area as compared to existing methods. This is accomplished by searching the common sub expressions in coefficients terms before filter implementation and sharing them to eliminate extra computational complexity. In programmable and reconfigurable filter, the coefficients are not fixed and it is not easy to find the common sub expressions for newly applied coefficients. The conventional FIR filter implementation approaches (CSD, CSE …) cannot be usable and the dedicated multipliers are required for each coefficient multiplication.

### *A. 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.

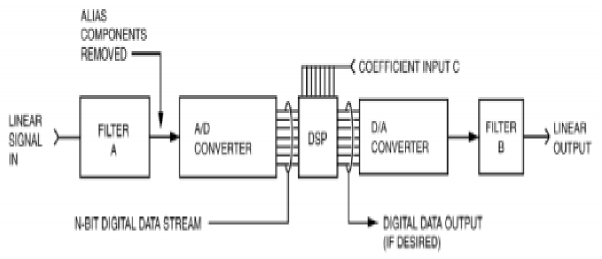


Figure3: A typical digital filter configuration.

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 .The structure of these algorithms uses a repetitive delay-and-add format that can be represented as "DIRECT
FORM-I STRUCTURE",

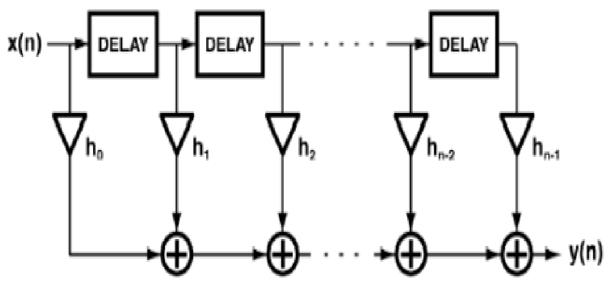


Figure 4 Transposed direct forms I FIR Filter

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 functions that have no equivalent in linear circuit technology.

In applications where linear phase is critical and long phase delay cannot be tolerated, a linear active Bessel or a constant delay filter may be a better selection. Two very different design techniques are commonly used to develop digital FIR filters:

1. The Window Technique
2. The Equiripple Technique.

*i. 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.

*ii. 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

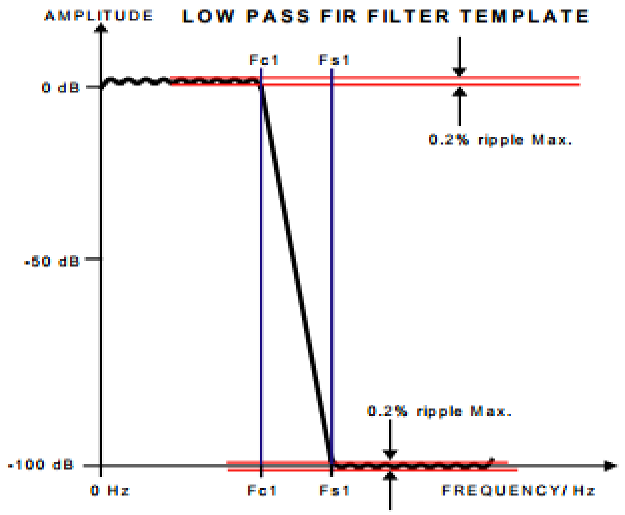


Figure5: low pass Equiripple FIR filter

(2) The Floating-Point DSP and IIR (Infinite Impulse Response) Implementations

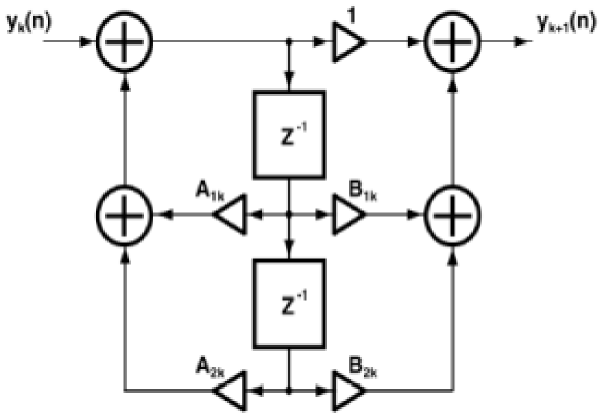


Figure6: Bi-quad digital filter structures

**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. Some techniques have been proposed to minimize the number of required adders for multiplier implementation. Canonical signed digit (CSD) coefficient representation [1] and Common Sub expression Elimination (CSE) [2] methods are well known approaches which produce FIR filter coefficient multipliers with low complexity. In fixed coefficients

FIR filter application, the CSE is an efficient approach to share the common sub expressions. The conventional FIR filter implementation approaches (CSD, CSE …) cannot be usable and the dedicated multipliers are required for each coefficient multiplication. Several methods have been proposed in [3-7] to implement the reconfigurable FIR filters. In [3] a fully programmable multiply accumulates (MAC) based filter processor has been proposed which is suffering from large delay of long data path, large area and power consumption requirements.

An interesting CSD based reconfigurable FIR filter architectures have been proposed in [4] and [5], where its area and power consumption are also large. Two other efficient reconfigurable FIR filter implementation approaches have been presented in [6] and [7]. These works have been focused on the implementing of FIR filter by partitioning the filter coefficient into fixed groups (sub-coefficients) and pre-computing the products of input data with these constant fixed sub coefficients. These partial products are distributed inside the chip instead of input data and properly selected by filter tap multiplexers to compose the desired coefficient multiplication.

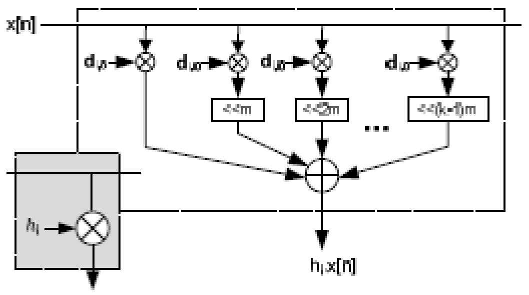


Fig. 7 Implementation of coefficient multiplication using the coefficient partitioning approach

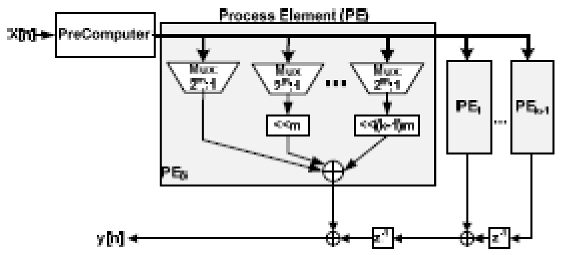


Fig. 8 Reconfigurable FIR filter architecture

## VI. Proposed Reconfigurable Fir Filter Architecture

The proposed coefficient representation technique uses signed digit to represent each sub-coefficients. In conventional coefficient partitioning method, the main coefficient may be assumed signed value, but the sub-coefficients are not signed. For m-bit word length sub-coefficients case, their values are in 0 ~ 2m-1 range.

Eight partial products are calculated by pre-computer block using shift/sum operation and distributed to each tap's PE block. In practical implementation just four ×1, ×3, ×5, ×7 partial products are implemented in pre-computer block and other products (×2, ×4, ×6, ×8) are composed by simple hardwire shift operation of above four partial products inside PE block. The required four sub-coefficients to compose the desired coefficient are selected by four 8:1 multiplexers, which are controlled by Mux control block. This block uses $h_{i,j}$ bits of each sub coefficient to control the selection bits of multiplexers. Note that it is need to eight partial products (×1, ×3, ×5, ×7, ×9, ×11, ×13 and ×15 in practical implementation) and four 16:1 multiplexers in conventional reconfigurable FIR filter architecture. The selected four partial products in PE block, after hardwire shift operation are combined by add/sub operation while controlled by 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 sub coefficient, 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 sub coefficients.

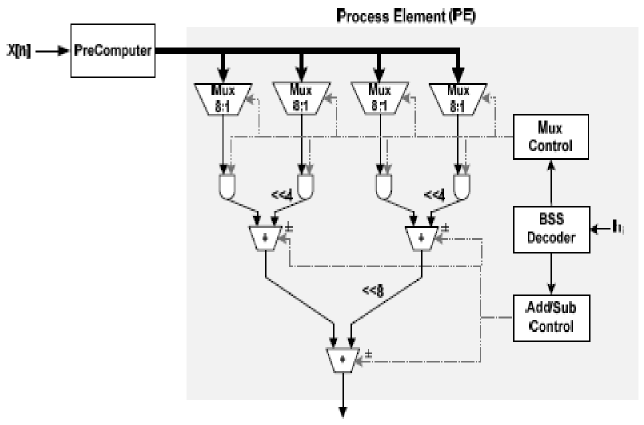


Fig. 9 Proposed reconfigurable FIR filter architecture using 4bit BSS representation

## 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 net list. This is a major step in ASIC design flow that takes an RTL model closer to a low-level hardware implementation.

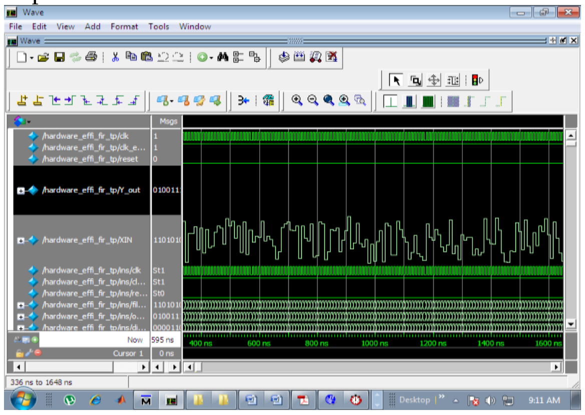

Wave
File Edit View Add Format Tools Window
/hardware_effi_fir_tp/clk
/hardware_effi_fir_tp/reset
/hardware_effi_fir_tp/Y_out
/hardware_effi_fir_tp/XIN
Now
Cursor 1
400 ns
600 ns
800 ns
1000 ns
1200 ns
1400 ns
1600 ns

Figure 10.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]

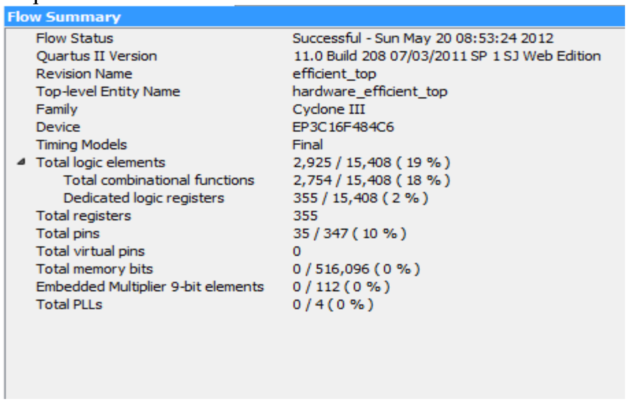
1- Input port of the design under test to input of one internal flip-flip
2- Output of an internal flip-flop to input of another flip-flip
3- Output of an internal flip-flop to output port of the design under test
4- A combinational path connecting the input and output ports of the design under test

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.

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 net list. 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 net list. The total area of the gate level net list 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 net list 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 11.Flow summary report

To enforce the synthesis tool to create the most compact net list, the area of the gate level net list 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:.

CONCLUSION

A new hardware efficient reconfigurable FIR filter architecture has been presented in this paper using proposed coefficient representation. Filter coefficients has been partitioned to smaller sub coefficients based on proposed binary signed sub coefficients. The partial products of all possible sub coefficients and input data have been calculated in pre-computer block and results are distributed on filter taps to compose the coefficient multiplication. Two reconfigurable FIR filters based on 4-bit partitioning were designed and synthesized.

## REFERENCES:

[1] M. Potkonjak, M. B. Srivastava, and A. Chandrakasan, "Efficient substitution of multiple constant multiplications by shifts and additions using iterative pairwise matching" in *Proc. 31st ACM/IEEE Design Automation Conf.*, 1994, pp. 189–194.

[2] I. C. Park and H. J. Kang, "FIR filter synthesis algorithms for minimizing the delay and the number of adders," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 8, pp. 770–777, Aug. 2001.

[3]. R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE*

*Trans. Circuits Syst. II*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[4]. Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner, "FPGA Implementation of High Speed FIR Filters
Using Add and Shift Method", International Conference on Computer Design, (ICCD), pp 308-313, 2006.

[5]. R. Mahesh and A. P. Vinod, "Reconfigurable low cmplexity FIR filters for software radio receivers,"
in Proc. 17th IEEE Int. Symp. Personal Indoor Mobile Radio Commun. (PIMRC), Helsinki, Finland, Sep.
2006, pp. 1–5.