ENGINEERING IN ADVANCED RESEARCHSCIENCE AND TECHNOLOGY

UGC Care Group-I
Vol.03, Issue.01
July-2022
Pages: -700-710

IMPLEMENTATION OF RADIX 8 FIR FILTER USING BOOTH MULTIPLIER FOR REDUCED POWER CONSUMPTION

[1]DHANDUBOYINA BHEESHMA DEVI, [2]K.VEERANNABABU
[1]M. tech, Dept. of ECE, Eluru College of Engineering and Technology, ELURU, AP
[2]Assistant Professor, Dept. of ECE, Eluru College of Engineering and Technology, ELURU, AP

ABSTRACT: The increasing complexity of the DSP systems demanding higher computational performance in its architecture. But the traditional DSP arithmetic has limits in terms of speed of calculations. More over in some applications speed is more important than accuracy. Transpose form finite-impulse response (FIR) filters are inherently pipelined and support multiple constant multiplications (MCM) technique that results in significant saving of computation. However, transpose form configuration does not directly support the block processing unlike directform configuration. In this paper, we explore the possibility of realization of block FIR filter in transpose form configuration for area-delay efficient realization of large order FIR filters for both fixed and reconfigurable applications. Based on a detailed computational analysis of transpose form configuration of FIR filter, we have derived a flow graph for transpose form block FIR filter with optimized register complexity. A generalized block formulation is presented for transpose form FIR filter. We have derived a Radix-4 modified booth encoding multiplier-based architecture for the proposed transpose form block filter for reconfigurable applications. A low-complexity design using the MCM scheme is also presented for the block implementation of fixed FIR filters. As an extension of this project, the proposed structure involves significantly less area delay product (ADP) than the existing block implementation using Radix-8 modified booth encoding algorithm.

INTRODUCTION: FIR DIGITAL filters find extensive applications in mobile communication systems for applications such as channelization,channel equalization, matched filtering, and pulseshaping, due to their absolute stability and linear phase properties.The filters employed in mobile systems must be realizedto consume less power and operate at high speed. Recently, with the advent of software defined radio (SDR) technology,finite impulse response (FIR) filter research has been focusedon reconfigurable realizations. The fundamental idea of anSDR is to replace most of the analog signal processing in thetransceivers with digital signal processing in order to provide the advantage of flexibility through reconfiguration. This will enable different air-interfaces to be implemented on a single generic hardware platform to support multi standard wireless communications [1]. Wideband receivers in SDR must be realized to meet the stringent specifications of low power consumption and high speed. Re-configurability of the

receiver to work with different wireless communication standards is another key requirement in an SDR. The most computationally intensive part of an SDR receiver is the channelizer since it operates at the highest sampling rate [2]. It extracts multiple narrow band channels from a wideband signal using a bank of FIR filters, called channel filters. Using poly phase filter structure, decimation can be done prior to channel filtering so that the channel filters need to operate only at relatively low sampling rates. This can relax the speed of operation of the filters to a good extent [2]. However due to the stringent adjacent channel attenuation specifications of wireless communication standards, higher order filters are required for channelization and consequently the complexity and power consumption of the receiver will be high. As the ultimate aim of the future multi-standard wireless communication receiver is to realize its functionalities in mobile handsets, where its full utilization is possible, low power and low area implementation of FIR channel filters is inevitable. The complexity of FIR filters is dominated by the complexityof coefficient multipliers. It is well known that the common sub expression elimination (CSE) methods based on canonical signed digit (CSD) coefficients produce low complexity FIR filter coefficient multipliers [3]. The goal of CSE is to identify multiple occurrences of identical bit patterns that are present in the CSD representation of coefficients, and eliminate these redundant multiplications. A modification of the 2-bit CSE technique in [3] for identifying the proper patterns for elimination of redundant computations and to maximize the optimization impact was proposed in [4]. In [5],the technique in [3] was modified to minimize the logic depth(LD) (LD is defined as the number of adder-steps in a maximal path of decomposed multiplications [7]) and thus to improve the speed of operation.

LITERATURE SURVEY: The research paper on the design of FIR filters are published in various journals and presented in many conferences. Here the paper selected describes the design of FIR filters using VHDL or Verilog language.Some of the paper represents the modular design approach of the FIR filters and which is implemented in spartan-3E FPGA/Xilinx Virtex-5 FPGA.The evaluation result shows good area/power efficiency and flexibility by using different architectures for application. .Most papers have used microprogrammed FIR filters design approach .Abdullah A.Aljuffri, Aiman S. Badawai , Mohammad S.Bensaleh, Abdulfattah M.Odeid and SayedManzoor Qasim [1] in paper entitled "FPGA implementation of scalable micro programmed FIR filter architectures using Wallace tree and Vedic multipliers". In this paper used Wallace Tree and Vedic multipliers for implementation of 8-tap and 16-tap sequential and parallel micro programmed FIR filters architectures The designs are realized using Xilinx virtex-5 FPGA. Synplify pro tool used for synthesis, translation, mapping and place and route process and Reports are generated by CAD tool. Performance analyze base on parameter such as minimum period, slice LUTs and maximum operating frequency. The sequential FIR filters architecture designed using Wallace Tree multiplier seems to be more efficient as compared to Vedic multipliers. For 8-tap FIR filter using Wallace Tree have minimum period 11.448 ns and maximum operating frequency 87.4 MHz And for 16-tap FIR filter using Wallace Tree have minimum period 10.491 ns and maximum operating frequency 85.3 MHz .A. Aljuffri. M. M. AlNahdi, A.A.Hemaid , O. A. Alshaalan, M. S. BenSaleh, A.M. Obeid and S. M. Qasim [2], in paper entitled, "ASIC realization and performance evaluation of scalable micro-programmed FIR filter architectures using Wallace tree and Vedic

multiplier". In this paper, Wallace tree and Vedic multiplier are used for efficient realization of 8-tap and 16-tap sequential and parallel scalable micro-programmed FIR filter architectures. The designs of FIR filter are coded in VHDL. Lfoundary 150nm standard-cell based technology is used for the hardware realization of the proposed designs in ASIC. Synopsys Design Compiler is used for thegate-level synthesis. Analyze the performance based on area, Slice LUTs and critical path delays. Wallace tree multiplier using CSA (Carry Skip Adder) has minimum area and delay while Vedic using KSA (Kogge-Stone Adder) has maximum area and delay. For 8-tap FIR filter have period 6.62 ns for 8-tap filter have period 6.62 ns and area 29496 μm^2.

FIR FILTERS: FIR filters of large order to meet the stringent frequency specifications. Very often these filters need to support high sampling rate for high-speed digital communication. Filter coefficients very often remain constant and known a priori in signal processing applications. This feature has been utilized to reduce the complexity of realization of multiplications. Several designs have been suggested by various researchers for efficient realization of FIR filters (having fixed coefficients) using distributed arithmetic (DA) and multiple constant multiplication (MCM) methods. DA-based designs use lookup tables (LUTs) to store pre-computed results to reduce the computational complexity. The MCM method on the other hand reduces the number of additions required for the realization of multiplications by common sub-expression sharing, when a given input is multiplied with a set of constants. The MCM scheme is more effective, when a common operand is multiplied with more number of constants.Block-processing method is popularly used to derive high-throughput hardware structures. It not only provides throughput-scalable design but also improves the area-delay efficiency. The derivation of block-based FIR structure is straightforward when direct-form configuration is used, whereas the transpose form configuration does not directly support block processing. But, to take the computational advantage of the MCM, FIR filter is required to be realized by transpose form configuration.

DIGITAL FINITE IMPULSE RESPONSE:

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying). The impulse response (that is, the output in response to a Kronecker delta input) of an Nth-order discrete-time FIR filter lasts exactly N + 1 samples (from first nonzero element through last nonzero element) before it then settles to zero. FIR filters can be discrete-time or continuous-time, and digital or analog.

For a causal discrete-time FIR filter of order N, each value of the output sequence is a weighted sum of the most recent input values:

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] \\ &= \sum_{i=0}^{N} b_i \cdot x[n-i], \end{aligned}$$

where:

- $x[n]$ is the input signal,
- $y[n]$ is the output signal,
- N is the filter order; an Nth-order filter has $(N+1)$ terms on the right-hand side
- b_i is the value of the impulse response at the i'th instant for $0 \leq i \leq N$ of an Nth-order FIR filter. If the filter is a direct form FIR filter then b_i is also a coefficient of the filter .

EXISTING METHOD:

The data-flow graphs (DFG-1 and DFG-2) of transpose form FIR filter for filter length N = 6, as shown in below figure, for

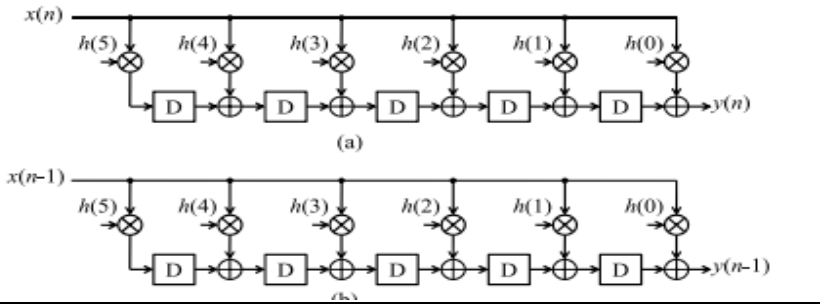

Fig:1 DFG of transpose form structure for N = 6. (a) DFG-1 for output y(n). (b) DFG-2 for output y(n − 1).

ccs	M_1	M_2	M_3	M_4	M_5	M_6
1	$x(n-5)h(5)$	$x(n-5)h(4)$	$x(n-5)h(3)$	$x(n-5)h(2)$	$x(n-5)h(1)$	$x(n-5)h(0)$
2	$x(n-4)h(5)$	$x(n-4)h(4)$	$x(n-4)h(3)$	$x(n-4)h(2)$	$x(n-4)h(1)$	$x(n-4)h(0)$
3	$x(n-3)h(5)$	$x(n-3)h(4)$	$x(n-3)h(3)$	$x(n-3)h(2)$	$x(n-3)h(1)$	$x(n-3)h(0)$
4	$x(n-2)h(5)$	$x(n-2)h(4)$	$x(n-2)h(3)$	$x(n-2)h(2)$	$x(n-2)h(1)$	$x(n-2)h(0)$
5	$x(n-1)h(5)$	$x(n-1)h(4)$	$x(n-1)h(3)$	$x(n-1)h(2)$	$x(n-1)h(1)$	$x(n-1)h(0)$
6	$x(n)h(5)$	$x(n)h(4)$	$x(n)h(3)$	$x(n)h(2)$	$x(n)h(1)$	$x(n)h(0)$

(a)

ccs	M_1	M_2	M_3	M_4	M_5	M_6
1	$x(n-6)h(5)$	$x(n-6)h(4)$	$x(n-6)h(3)$	$x(n-6)h(2)$	$x(n-6)h(1)$	$x(n-6)h(0)$
2	$x(n-5)h(5)$	$x(n-5)h(4)$	$x(n-5)h(3)$	$x(n-5)h(2)$	$x(n-5)h(1)$	$x(n-5)h(0)$
3	$x(n-4)h(5)$	$x(n-4)h(4)$	$x(n-4)h(3)$	$x(n-4)h(2)$	$x(n-4)h(1)$	$x(n-4)h(0)$
4	$x(n-3)h(5)$	$x(n-3)h(4)$	$x(n-3)h(3)$	$x(n-3)h(2)$	$x(n-3)h(1)$	$x(n-3)h(0)$
5	$x(n-2)h(5)$	$x(n-2)h(4)$	$x(n-2)h(3)$	$x(n-2)h(2)$	$x(n-2)h(1)$	$x(n-2)h(0)$
6	$x(n-1)h(5)$	$x(n-1)h(4)$	$x(n-1)h(3)$	$x(n-1)h(2)$	$x(n-1)h(1)$	$x(n-1)h(0)$

(b)

Fig:2 (a) DFT of multipliers of DFG shown in Fig. 1(a) corresponding to output y(n). (b) DFT of multipliers of DFG shown in Fig. 1(b) corresponding to output y(n − 1). Arrow: accumulation path of the products. a block of two

successive outputs {y(n), y(n − 1)} that are derived from (2). The product values and their accumulation paths in DFG-1 and DFG-2 of Fig. 1 are shown in data- flow tables (DFT-1 and DFT-2) of Fig. 2. The arrows in DFT-1 and DFT-2 of Fig. 2 represent the accumulation path of the products. We find that five values of each column of DFT-1 are same as those of DFT-2 (shown in gray color in Fig. 2). These redundant computation of DFG-1 and DFG-2 can be avoided using nonoverlapped sequence of input blocks, as shown in Fig. 2. DFT-3 and DFT-4 of DFG-1 and DFG-2 for nonoverlapping input blocks are, respectively, shown in Fig. 2(a) and (b). As shown in Fig. 2(a) and (b), DFT-3 and DFT-4 do not involve redundant computation. It is easy to find that the entries in gray cells in DFT-3 and DFT-4 of Fig. 2(a) and (b) correspond to the output y(n), whereas the other entries of DFT-3 and DFT-4 correspond to y(n−1).

BOOTH MULTIPLIER:

Booth's Multiplication Algorithm is a Multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeckcollege in Bloomsbury, London. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

RADIX-4 MODIFIED BOOTH ALGORITHM:

Booth multiplication algorithm consists of three major steps as shown in structure of Booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product. modified Booth algorithm & for multiplication, we must know about each block of Booth algorithm for multiplication process. It is possible to reduce the number partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that , instead of shifting and adding for every column of multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by+ (or) - 1,+ (or) -2,or 0,to obtain the same results.Radix-4 Booth encoder performs the process of encoding the multiplicand based on multiplier bits. It will compare 3 bits at a time with overlapping technique. Grouping starts from the LSB, and the first block only uses two bits of the multiplier and assumes a zero for the third bit. Radix-4 Booth algorithm is given below: 1. Extend the sign bit 1 position if necessary to ensure that n is even. 2. Append a 0 to the right of the LSB of the multiplier.

Bit position			Operation
i+1	I	i-1	
0	0	0	0*M
0	0	1	1*M
0	1	0	1*M
0	1	1	2*M
1	0	0	-2*M
1	0	1	-1*M
1	1	0	-1*M
1	1	1	0*M

DRAWBACKS:

1. More combinational path delay

2. Required more logic gates to design these FIR filters

3.less throughput with more flickers

RADIX-8 MODIFIED BOOTH ALGORITHM: The Booth algorithm consists of repeatedly adding one of two predetermined values to a product P and then performing an arithmetic shift to the right on P.

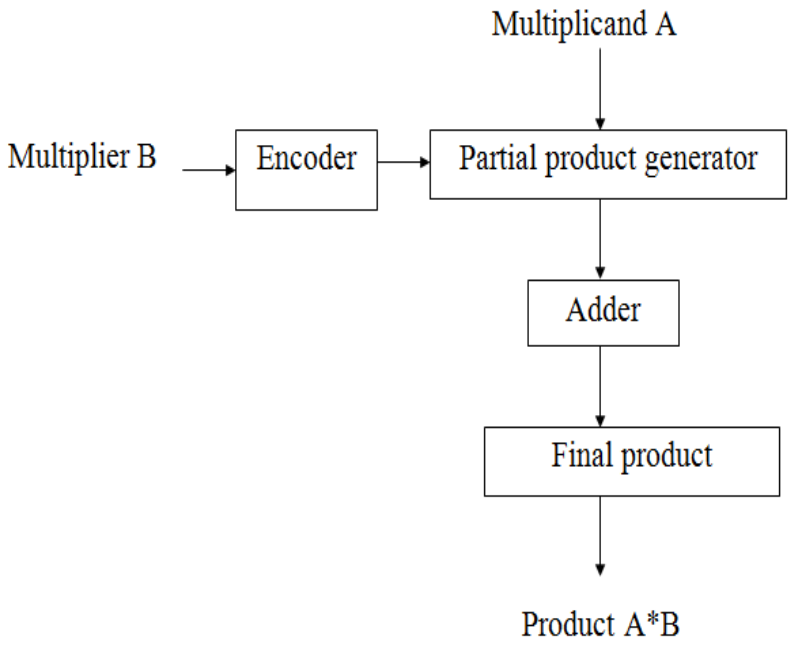

Fig3. Booth algorithm

The multiplier architecture consists of two architectures, i.e., Modified Booth. By the study of different multiplier architectures, we find that Modified Booth increases the speed because it reduces the partial products by half. Also, the delay in the multiplier can be reduced by using Wallace tree. The energy consumption of the Wallace Tree multiplier is also lower than the Booth and the array. The characteristics of the two multipliers can be combined to produce a high-speed and low-power multiplier. The modified stand-alone multiplier consists of a modified recorder (MBR). MBR has two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The operation of BE is to decode the multiplier signal, and the output is used by BS to produce the partial product. Then, the partial products are added to the Wallace tree adders, similar to the carry-save-adder approach. The last transfer and sum output line are added by a carry look- ahead adder, the carry being stretched to the left by positioning.

Table .Quartet coded signed-digit table

Quartet value	Signed-digit value
0000	0
0001	+1
0010	+1
0011	+2
0100	+2
0101	+3
0110	+3
0111	+4
1000	-4
1001	-3
1010	-3
1011	-2
1100	-2
1101	-1
1110	-1
1111	0

Here we have a multiplication multiplier, 3Y, which is not immediately available. To Generate it, we must run the previous addition operation: 2Y + Y = 3Y. But we are designing a multiplier for specific purposes and then the multiplier belongs to a set of previously known numbers stored in a memory chip. We have tried to take advantage of this fact, to relieve the radix-8 bottleneck, that is, 3Y generation. In this way, we try to obtain a better overall multiplication time or at least comparable to the time, we can obtain using a radix-4 architecture (with the added benefit of using fewer transistors). To generate 3Y with 21-bit words you just have to add 2Y + Y,ie add the number with the same number moved to a left position. A product formed by multiplying it with a multiplier digit when the multiplier has many digits. Partial products are calculated as intermediate steps in the calculation of larger products. The partial product generator is designed to produce the product multiplying by multiplying A by 0, 1, -1, 2, -2, -3, -4, 3, 4. Multiply by zero implies that the product is "0 ". Multiply by" 1 "means that the product remains the same as the multiplier. Multiply by "-1" means that the product is the complementary form of the number of two. Multiplying with "-2" is to move left one as this rest as per table.

SIGN EXTENSION CORRECTOR:

The Sign Extension Corrector is designed to increase the Booth multiplier capacity by multiplying not only the unsigned number but also the signed number. The principle of the sign extension that converts the signed multiplier not signed as follows. When unsign is signalled s_u = 0, it indicates the multiplication of the unsigned number and when s_u = 1, it shows the multiplication of the signed number. When a bit signal is called unsigned bit (s_u), it is indicated whether the multiplication operation is an unsigned number or number.

Table.5. Sign extension corrector

Sign-unsign	Type of operation
0	Unsigned multiplication
1	Signed multiplication

Example:

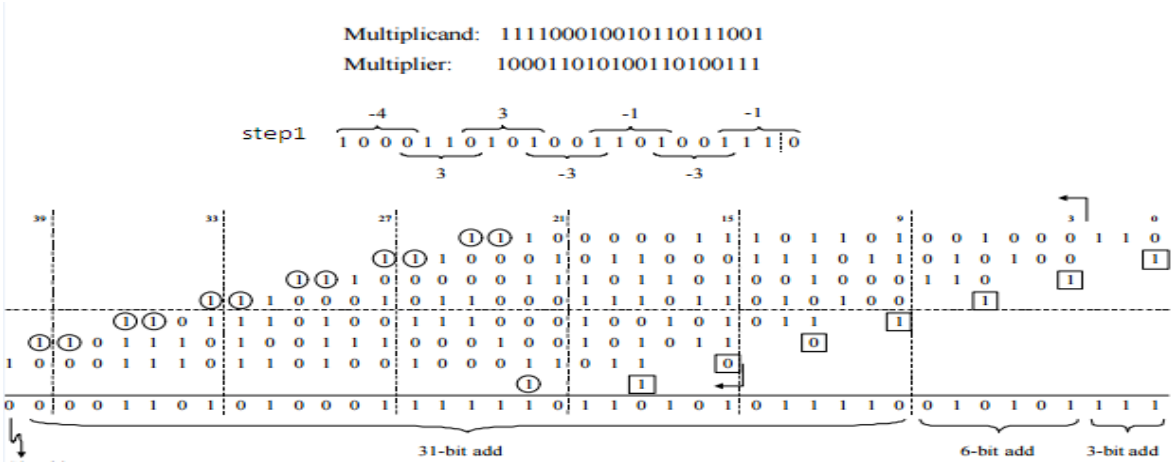

Fig4. Example of modified booh algorithem

RESULT:

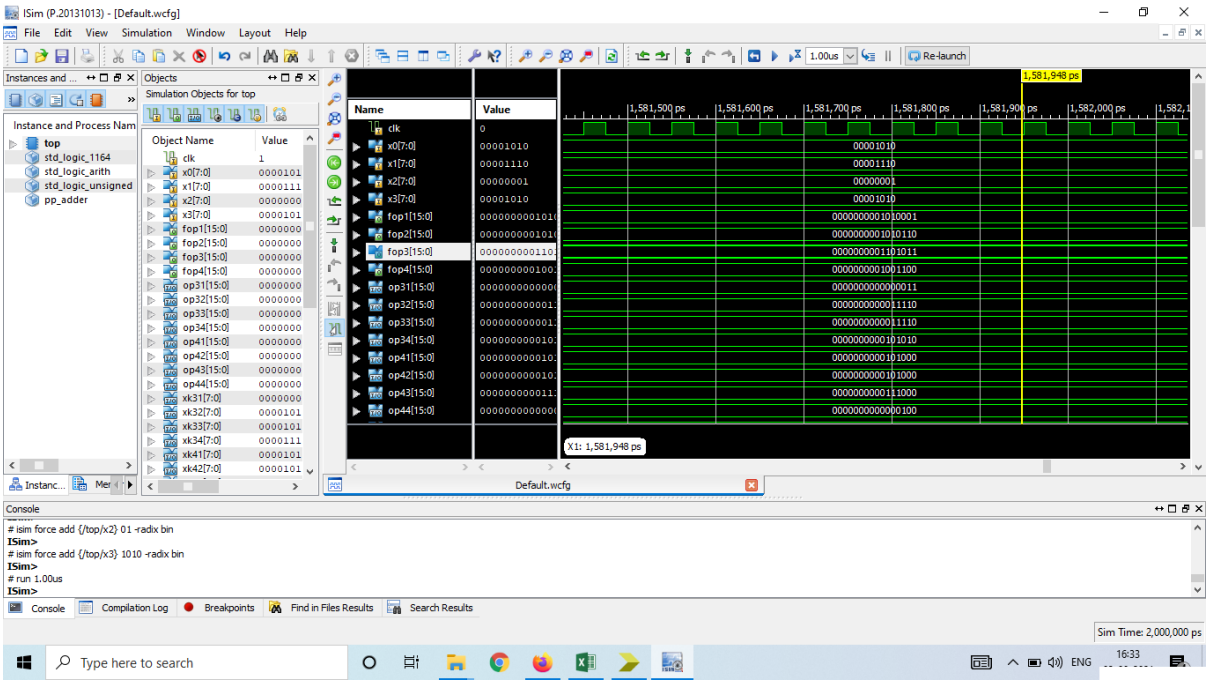

Fig5: proposed FIR filter output

Above snap shot is 4 tap FIR filter with 4 inputs and 4 coefficient constants.

Coefficient1=”00000001”, Coefficient2=” 00000010”,Coefficient3=”00000011”, Coefficient4=”00000100”; Input1=”00001010”, Input2=”00001110”,Input3=”00000001”,Input4=”00001010”;

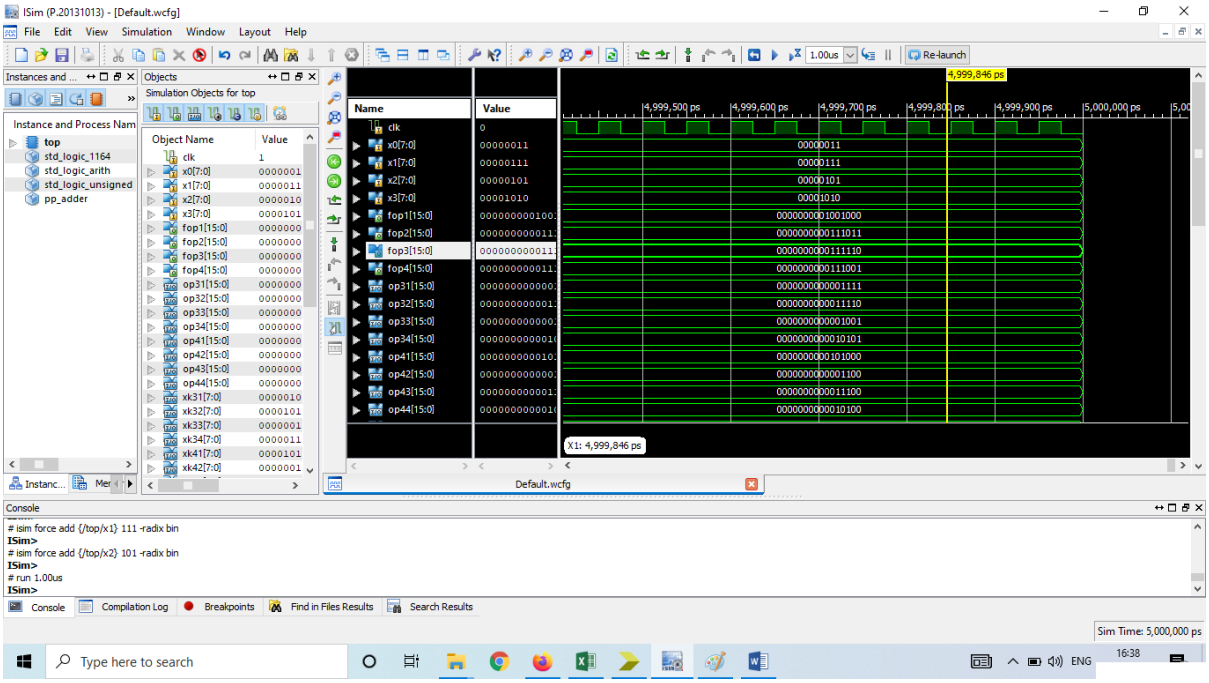

Fig6: proposed FIR filter output

Above snap shot is 4 tap FIR filter with 4 inputs and 4 coefficient constants.

Coefficient1=”00000001”, Coefficient2=” 00000010”,Coefficient3=” 00000011”, Coefficient4=” 00000100”; Input1=”00000011”, Input2=”00000111”,Input3=”00000101”,Input4=”00001010”;

CONCLUSION: In this paper, we have explored the possibility of realization of block FIR filters in transpose form configuration for area delay efficient realization of both fixed and reconfigurable applications. A generalized block formulation is presented for transpose form block FIR filter, and based on that we have derived transpose form block filter for reconfigurable applications. We have presented a scheme to identify the MCM blocks for Radix-8 modifed booth encoding in the proposed block FIR filter for fixed coefficients to reduce the computational complexity

FUTURE SCOPE: 1. Research can be extended in design of FIR filter using various optimization techniques ACO, PSO etc. 2. In further work FIR filters can be design using evolutionary algorithms etc. we can design according to that recent technique and low power processor can be designed

REFERENCES:

[1] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Englewood Cliffs, N.J., USA: Prentice Hall, 1993.

[2] A. Sibille, C. Oestges and A. Zanella, *MIMO: From Theory to Implementation*, New York, NY, USA: Academic, 2010.

[3] N. Kanekawa, E. H. Ibe, T. Suga and Y. Uematsu, *Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro- Magnetic Disturbances*, New York, NY, USA: Springer Verlag, 2010.

[4] M. Nicolaidis, "Design for soft error mitigation," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 405–418, Sep. 2005.

[5] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM J. Res. Develop.*, vol. 28, no. 2, pp. 124–134, Mar. 1984.

[6] A. Reddy and P. Banarjee "Algorithm-based fault detection for signal processing applications," *IEEE Trans. Comput.*, vol. 39, no. 10, pp. 1304–1308, Oct. 1990.

[7] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in *Proc. IEEE IOLTS*, 2008, pp. 192–194.

[8] Z. Gao, W. Yang, X. Chen, M. Zhao and J. Wang, "Fault missing rate analysis of the arithmetic residue codes based fault-tolerant FIR filter design," in *Proc. IEEE IOLTS*, 2012, pp. 130–133.

[9] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14,no. 4, pp. 336–348, Apr. 2006.

[10] Y.-H. Huang, "High-efficiency soft-error-tolerant digital signal processing using fine-grain subword-detection processing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no 2, pp. 291–304, Feb. 2010.

[11] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Structural DMR: A technique for implementation of soft-error-tolerant FIR filters," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 58, no. 8, pp. 512–516, Aug. 2011.

[12] P. Reviriego, S. Pontarelli, C. Bleakley and J. A. Maestro, "Area efficient concurrent error detection and correction for parallel filters," *IET Electron. Lett.*, vol. 48, no 20, pp. 1258–1260, Sep. 2012.

[13] Z. Gao *et al.*, "Fault tolerant parallel filters based on error correction codes," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 2, pp. 384–387, Feb. 2015.

[14] R. W. Hamming, "Error correcting and error detecting codes," *Bell Sys. Tech. J.*, vol. 29, pp. 147–160, Apr. 1950.

[15] J. Park, W. Jeong, H. M. Meimand, Y. Wang, H. Choo, and K. Roy, "Computation sharing programmable FIR filter for low-power and high-performance applications," IEEE J. Solid-State Circuits, vol. 39, no. 2, pp. 348–357, Feb. 2004.