ISSN 2277-2685
IJESR/Jan-Mar. 2025/ Vol-15/Issue-1/192-201
Y. Narayanamma *et. al.*, / International Journal of Engineering & Science Research

Implementation Of Odd-Length Fir Filter Design In 3-Parallel Polyp Phase Using Brent-Kung Adder And Booth Multiplier For VLSI Applications

[1]Y. Narayanamma, [2]M. Sucharita

[1]PG Scholar, Dept. of E.C.E, Priyadarshini Institute of Technology &Science for Women, Tenali, Guntur Dt.

[2]Asst Professor, Dept. of E.C.E, Priyadarshini Institute of Technology &Science for Women, Tenali, Guntur Dt.

ABSTRACT

Digital Signal Processing (DSP) plays a crucial role in various fields, including biomedical applications, as well as voice and image processing. In DSP and signal analysis, digital filters are essential components. With the advancement of VLSI (Very-Large-Scale Integration) technology, the number of processes required to design digital filters has reduced, promoting the development of on-chip VLSI architectures for DSP applications. In DSP, Finite Impulse Response (FIR) filters are characterized by a finite duration of the impulse response, meaning they settle to zero after a specific period. This is in contrast to Infinite Impulse Response (IIR) filters, which may exhibit continuous responses due to internal feedback. This paper explores the implementation of a 3-parallel poly phase FIR filter of odd length, based on the FF Algorithm, utilizing optimized adders and multipliers to replace traditional components. Specifically, the design uses two types of multipliers Vedic and Booth multipliers and three types of adders Ripple Carry Adder, Carry Look-Ahead Adder, and Brent-Kung Adder. The focus is on reducing circuit complexity by using a modified version of the Brent-Kung Adder instead of the conventional one, thus improving circuit efficiency. The proposed design is implemented using Xilinx and simulated in Model Sim software.

Key Words: *Program Processors, Finite Impulse Response Filters, Digital Signal Processing, Very Large Scale Integration, Delays, Power Dissipation*

1. INTRODUCTION

The Digital Signal Processing (DSP) is extensively used in the biomedical fields along with the disciplines of voice and image processing. For DSP applications as well as signal analysis and estimation, digital filters are practical building blocks .With the advent of VLSI-based technology, the number of processes needed for creating digital filters has gradually decreased, which has encouraged the creation of on-chip VLSI-oriented architecture for DSP applications. A digital filter based on their impulse response is divided into two groups: First is FIR filters and second is infinite impulse response (IIR) filters. Compared to IIR filters, FIR filters are easier to use, have better stability, and are guaranteed to have linear phase characteristics. Filters have two uses: signal separation and signal restoration. Signal separation is needed when the signal has been contaminated with interference, noise or other signals. For example imagine a device for measuring the electrical activity of a baby's heart (EKG) while in the womb. The raw signal will be likely to be corrupted by the breathing and the heartbeat of the mother. A filter must be used to separate these signals so that they can be individually analyzed. Signal restoration is used when the signal has been distorted in some way. For example, an audio recording made with poor requirement may be filtered to better represent the sound as it actually occurred. Another

example is of debuting of an image acquired with an improper focused lens, or a shaky camera. These problems can be attacked with either digital or analog filters. Which is better? Analog filters are cheap, fast and have a large dynamic range both in amplitude and frequency. Digital filters in comparison are vastly superior in the level of performance that can be achieved. Digital filters can achieve thousands of times better performance than an analog filter. This makes a dramatic difference in how filtering problems are approached. With analog filters, the emphasis is on handling limitations of the electronics such as the accuracy and stability of the resistors and capacitors. In comparison digital filters are so good that the performance of the filter is frequently ignored. The emphasis shifts to the limitations of the signals and the theoretical issues regarding their processing. [1] Describes many multiplication algorithms. The most effective mathematical techniques in terms of delay and area are Vedic ones. There is a wealth of material on Vedic mathematical techniques. The most popular approach for greater bit length multipliers is the Karatsuba algorithm. But each algorithm has advantages and pitfalls of its own. The Urdhva-Tiryagbhyam method from Vedic mathematics and the Karatsuba algorithm are combined in the suggested paradigm. The greatest aspects of both algorithms can be utilised effectively to reduce both latency and area by combining them. The FIR filter used in this project was created using the multipliers stated above. The same's outcomes have been attained and scrutinised individually. The results have also been contrasted on various factors such as power consumption, device utilization area (LUT, Slices, gate count) and delay factors such as gate delay and path delay.

2. LITERATURE SURVEY

Hardware-efficient VLSI implementation for 3-parallel linear-phase FIR digital filter of odd length by Y.C. Tsao, and K. Choi(IEEE ISCAS, 2012). Based on fast FIR algorithms (FFA), this paper proposes new 3-parallel finite-impulse response (FIR) filter structures, which are beneficial to symmetric convolutions of odd length in terms of the hardware cost. The proposed 3-parallel FIR structures exploit the inherent nature of the symmetric coefficients of odd length, according to the length of filter, (N mod 3), reducing half the number of multipliers in sub filter section at the expense of additional adders in pre-processing and post processing blocks. The overhead from the additional adders in pre-processing and post processing blocks stay fixed, not increasing along with the length of the FIR filter, whereas the number of reduced multipliers increases along with the length of the FIR filter. For example, for a 81-tap filter, the proposed A structure saves 26 multipliers at the expense of 5 adders, whereas for a 591-tap filter, the proposed structure saves 196 multipliers at the expense of 5 adders still. Overall, the proposed 3-parallel FIR structures can lead to significant hardware savings for symmetric coefficients of odd length from the existing FFA parallel FIR filter, especially when the length of the filter is large.

3.EXISTING SYSTEM

3.1 Vedic Multiplier for 8x8 bit Module

The 8x8 bit Vedic multiplier module as shown in the block diagram in Fig. 6 can be easily implemented by using four 4x4 bit Vedic multiplier modules as discussed in the previous section. Let's analyse 8x8 multiplications, say A= A7 A6 A5 A4 A3 A2 A1 A0 and B= B7 B6 B5B4 B3 B2 B1B0. The output line for the multiplication result will be of 16 bits as – S15 S14 S13 S12 S11 S10 S9 S8 S7 S6S5S4 S3 S2 S1 S0. Let's

divide A and B into two parts, say the 8 bit multiplicand A can be decomposed into pair of 4 bits AH-AL. Similarly multiplicand B can be decomposed into BH-BL. The 16 bit product can be written as: Using the fundamental of Vedic multiplication, taking four bits at a time and using 4 bit multiplier block as discussed we can perform the multiplication. The outputs of 4x4 bit multipliers are added accordingly to obtain the final product.

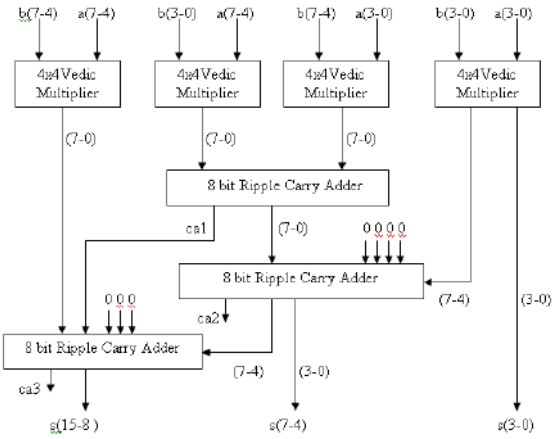

Fig: 3.1: Block Diagram of 8x8 bit Vedic Multiplier

Multiplication technique in ancient times includes multiplication of both smaller numbers and also for the larger numbers

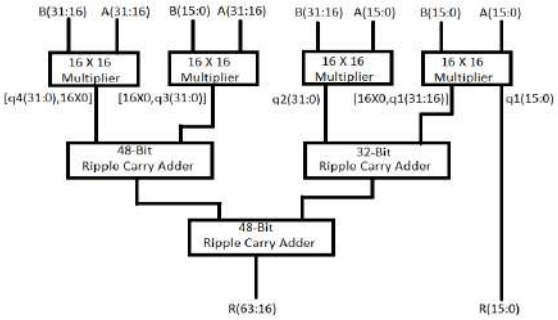

Fig .3.2: Proposed 32-Bit Vedic Multiplier Architecture

Fig.3.2 explains the multiplication architecture of the Urdhva Tiryakbhyam sutra for a 32 X 32-bit multiplication. This is developed using a 2x2 Vedic multiplier using ripple carry adders. So, here two 32-bit numbers A (31:0) and B (31:0) are taken

A (15:0) X B (15:0)

A (15:0) X B (31:16)

A (31:16) X B (15:0)

A (31:16) X B (31:16)

The result from the first multiplication i.e. A (15:0) X B (15:0) is stored in q1(31:0). Out of this q1(31:0), the q1(15:0) is directly stored in the result R (15:0). The other part of q1 i.e., q1(31:16) is appended with 16 zeros before and sent to 32-Bit Ripple Carry Adder (RCA) as input. The result from the second multiplication i.e. A

(15:0) X B (31:16) is stored in q2(31:0) and given as other input to 32-Bit Ripple Carry Adder. The result from third multiplication i.e. A (31:16) X B (15:0) is stored in q3(31:0) and appended with 16 zeros before q3 and sent as input to 48-Bit Ripple Carry Adder. The result of fourth multiplication i.e. A (31:16) X B (31:16) is stored in q4(31:0) and 16 zeros are appended after q4 and sent as other input for 48-Bit Ripple Carry Adder.
Finally, result from 32-Bit RCA and 48-Bit RCA is added using a 48-Bit Ripple Carry Adder and the result R (63:16) is obtained. By combining R (63:16) and R (15:0) are combined to get the final result i.e., R (63:0).
Implementation of two different multipliers and three adders. This is followed by separate comparisons among the adders and the multipliers. From these comparisons, it can be concluded that the 3-parallel poly phase odd length FFA based FIR filter design using Brent Kung adder and Booth multiplier gives best result among multipliers and adders respectively.

- The existed system drawback is for Brent Kung adder face more complexity it is the disadvantage.

4. PROPOSED SYSTEM

It is readily apparent that a key advantage of the tree-structured adder is that the critical path due to the carry delay is on the order of *log 2N* for an N-bit wide adder. The arrangement of the prefix network gives rise to various families of adders. For a discussion of the various carry-tree structures, see [1, 3]. For this study, the focus is on the Kogge-Stone adder , known for having minimal logic depth and fan out. Here we designate BC as the black cell which generates the ordered pair in equation (1); the gray cell (GC) generates the left signal only, following. The interconnect area is known to be high, but for an FPGA with large routing overhead to begin with, this is not as important as in a VLSI implementation. The regularity of the Kogge -Stone prefix network has built in redundancy which has implications for fault-tolerant designs. The sparse Kogge -Stone adder, shown in Fig 1(b), is also studied. This hybrid design completes the summation process with a 4 bit RCA allowing the carry prefix network to be simplified.

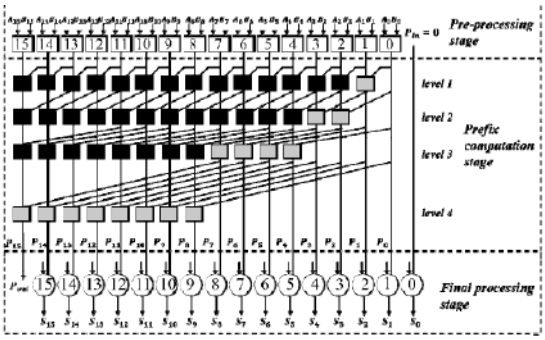

Fig 4.1: bit parallel prefix adder

The Booth multiplier is the competent multiplier which treats both positive and negative number reliably unlike regular multiplier. By and large duplication can be performed by add and shift activity, in which each multiplier bit makes one different piece of the multiplicand that must be added to the fractional item. The bigger number of multiplicand must be added when the multiplier is bigger in size; subsequently the deferral of the multiplier is high. Since the deferral relies on the quantity of expansion activity. To get better execution, we need to limit the quantity of expansion which thusly diminishes the quantity of incomplete item. The productive calculation that will limit the quantity of multiplicand is Booth calculation.

The Booth algorithm, examines the multiplier word 'y' and searches for 0's since these have no effect on the sum. This may be used to encode groups of bits in 'y' to produce a control digit that specifies the operation to be performed on the multiplicand 'x'.

Encoding digit since yk has a value of '0' or '1'. The yk can have decimal values of +2,+1,0,-1,-2. To compute the product x*y using radix-4 Booth multiplier, we divide 'y' into three bit segments that overlap by one bit. The last zero on the right has been added for y-1 = 0. Each group gives a value of yk that determines an operation. The product is computed by providing a dual-word size register that holds the sum after every operation is completed. The Table I summarizes the meaning of the encoded values.

y_{2k+1}	y_{2k}	y_{2k-1}	E_k	Effect on sum
0	0	0	0	Add '0'
0	0	1	+1	Add 'x'
0	1	0	+1	Add 'x'
0	1	1	+2	Shift 'x' left, add
1	0	0	-2	Take 2's (x), shift left, add
1	0	1	-1	Add 2's(x)
1	1	0	-1	Add 2's(x)
1	1	1	0	Add 0

Table I: multiplier in block

The Procedure for implementing the Booth Algorithm is as follows

Step 1: Formation of Booth recoding table consider the multiplier in block of three, such that each block overlaps the previous block by one bit and form the table using Table I.

Step 2: Booth Algorithm

The multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the Booth rule and then shifted. Radix-4 booth encoding reduces the number of multiplier bits so that the number of partial products is also reduced, which reduces the delay as compared to the normal multiplication process. Using this encoding process, the length of the multiplier is changed from n to n/2. So, the number of partial products is also reduced from n to n/2 as shown in Fig. After the generation of partial products serially, the generated partial products are added in the serial manner only. In this sense, the circuit contains two registers of length m+n. One register is named pp and the second register is a product.

Initially, these registers, pp, and products, contain zero. After generating the first partial product, it is stored in pp(register). Here the length of the partial product is m which has to be extended by sign extinction and converted to (m + n) bits. Next, the content of these two registers is added using a ripple carry adder (RCA) which gives pp + product [m+n-1,0]. Then next partial product is generated and stored in pp and add with the product. Now it will be added like pp + product [m+n-1,1]. Since product [0] will no longer contribute to generating any carry so the addition of partial products will be [3]-

$$product = pp + product[m + n - 1,0] \quad \text{.....(1)}$$

$$product = pp + product[m + n - 1,1] \quad \text{.....(2)}$$

$$product = pp + product[m + n - 1,2] \quad \text{.....(3)}$$

$$product = pp + product[m + n - 1,3] \quad \text{.....(4)}$$

Partial products are added until it reduces to two final rows. After the addition of all the partial products, the final two rows are added using Brent Kung Adder. Initially, for LSB bit addition, they carry is zero in the Brent Kung Adde). Brent Kung Adder takes one bit from each row, add them using full adder and generates sum and carry. The sum output is saved and carry is passed to the next full adder. Brent Kung Adder requires less area but it takes more time to add the bits because it depends on the previous carry generated

It is readily apparent that a key advantage of the tree-structured adder is that the critical path due to the carry delay is on the order of *log 2N* for an N-bit wide adder. The arrangement of the prefix network gives rise to various families of adders. For a discussion of the various carry-tree structures, see [1, 3]. For this study, the focus is on the parallel prefix, known for having minimal logic depth and fanout. Here we designate BC as the black cell which generates the ordered pair in equation (1); the gray cell (GC) generates the left signal only, following . The interconnect area is known to be high, but for an FPGA with large routing overhead to begin with, this is not as important as in a VLSI implementation. The regularity of the parallel prefix has built in redundancy which has implications for fault-tolerant designs. The sparse parallel prefix adder, shown in Fig 1(b), is also studied. This hybrid design completes the summation process with a 4 bit RCA allowing the carry prefix network to be simplified

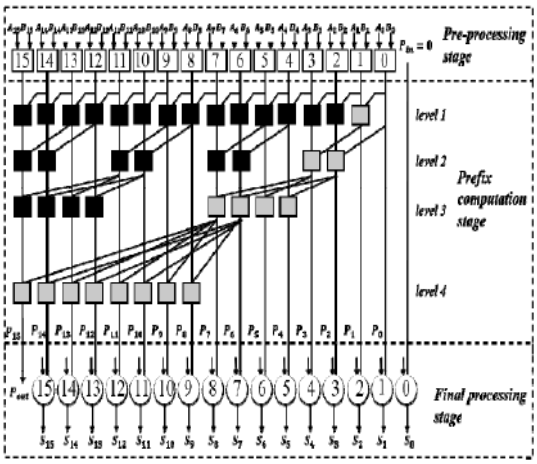

Fig .4.2 Modified 16-bit parallel prefix adder

The construction of the first level of the prefix tree of this adder is similar to the construction of parallel prefix adder. The main structural difference begins from the second level of the prefix tree. At the second level of the prefix tree, the groups of two schematic nodes are formed, at the 3rd level – groups compose four schematic nodes and at the 4th level – groups including 8 schematic nodes, etc. This adder first computes gi and hi signals for the first stage. Then at the first level of prefix tree, Gi:k and Pi:k signals of 2-bit are computed at the same time, and then, it

computes Gi:k and Pi:k signals for pairs of columns, then for blocks of 4, then for blocks of 8, then 16, and so on until the final Gi:k signal for every column is known. Finally, at the last stage this adder computes the sums together with the generated signals obtained from the previous prefix computation stage. The number of levels of the prefix tree corresponds to (log2 n) and the number of schematic nodes

5. SIMULATION RESULTS

The implementation of the odd-length FIR filter design in 3-parallel polyp phase using the Brent-Kung adder and Booth multiplier for VLSI applications demonstrates significant improvements in performance, particularly in terms of area, speed, and power efficiency. The use of the Brent-Kung adder optimizes the addition process, reducing the critical path delay, while the Booth multiplier enhances the speed of multiplication operations

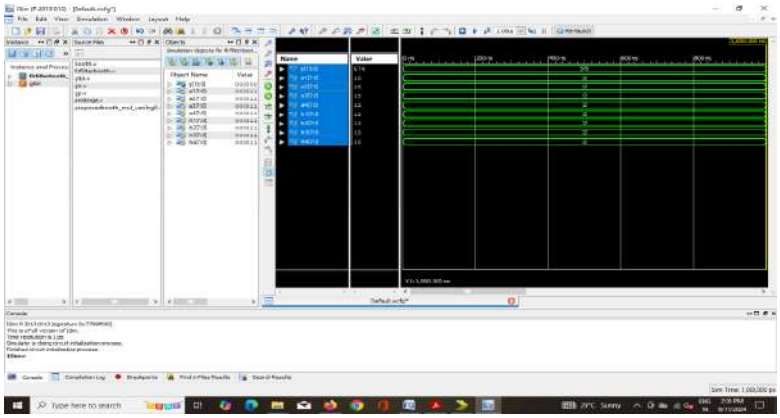

Fig 5.1 Simulation Result

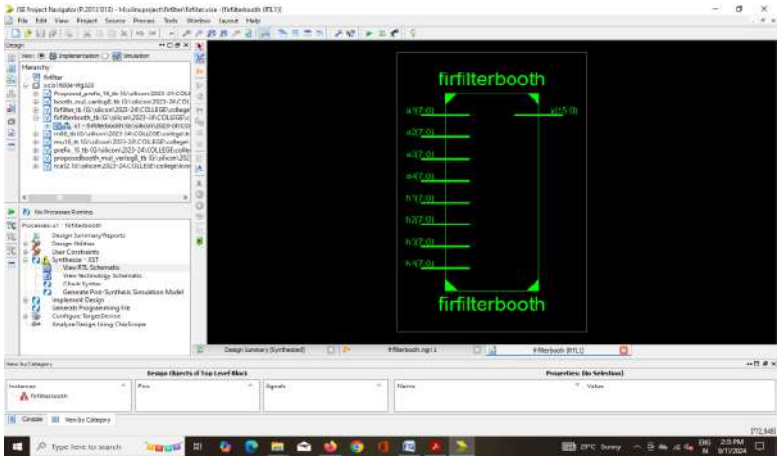

Fig 5.2: Block Diagram

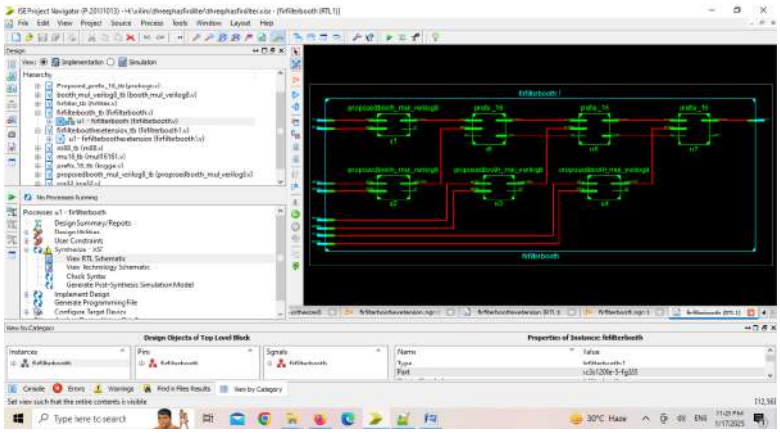

Fig 5.3: RTL Schematics

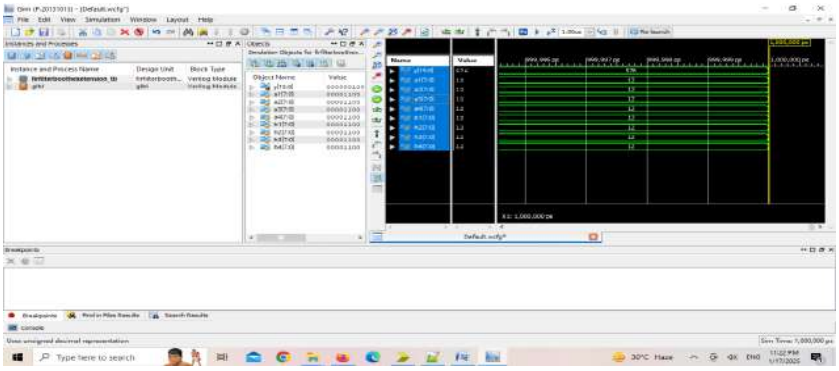

Fig 5.4: Extension Simulation Result

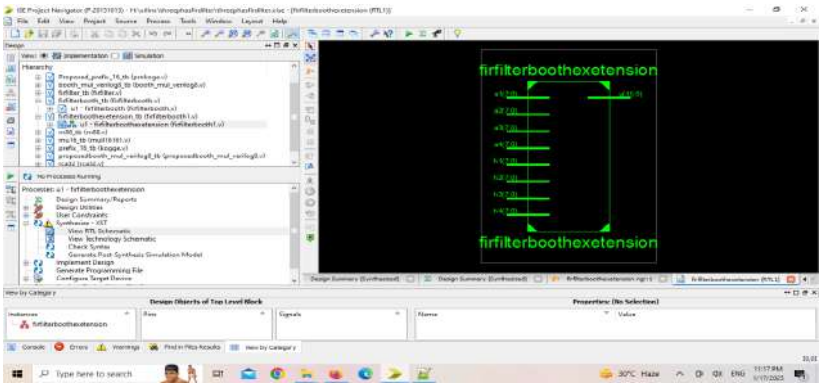

Fig 5.5: Extension block diagram

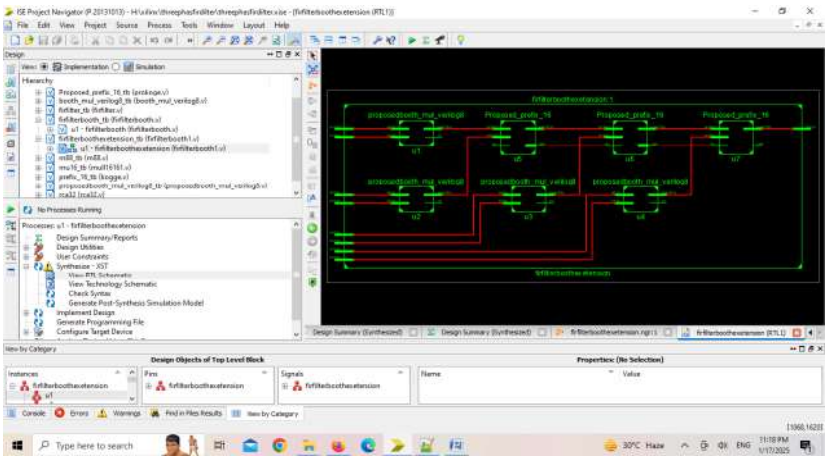

Fig 5.6: Extension RTL

	PROPOSED	EXTENSION
AREA	1065	986
DELAY	42.935NS	41.134NS
SPEED	23.29MHZ	24.31MHZ

Table.2: Comparison Proposed and Extension

6. CONCLUSION AND FUTURE WORK

The parallel FIR filter plays very important role in Digital Signal Processing. With the advancement in the technology, less chip area and low power consumption is the primary focus of FIR filters design using VLSI technology. The parallel FIR filter contains three units: adder, multiplier, and delay. The multiplier is the slowest among all the units, adder also contributes in slowing the process. To get efficient multiplier and adder, the 3- parallel polyphase FFA based odd length FIR filter has been designed using two different multipliers and three different adders. In this research paper the presented idea was able to achieve adequate results. It is evident from the results that chip area and power dissipation of 3-Parallel polyphase odd length FIR filter designed using Brent Kung adder and Booth multiplier is significantly reduced, hence making the system faster.

REFERENCES

[1] K. K. Parhi,VLSI Digital Signal Processing System : Design and Implementation (Wiley, New York, 1999)

[2] L. K. Phimu and M. Kumar, "Design and implementation of area efficient 2-parallel filters on FPGA using image system" in proc. ICECDS, 2017

[3] 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, 1991.

[4] D. A. Parker, and K. K. Parhi, "Low-area/power parallel FIR digital filter implementation," J. VLSI Signal Process. Syst, vol. 17, 1997.

[5] J. Selvakumar and Vidhyacharan Bhaskar, "Efficient complexity reduction technique for parallel FIR digital Filter based on Fast FIR algorithm,"International Journal of Computer Applications ,Vol. 55, 2012

[6] Y.C. Tsao, and K. Choi,,"Hardware-efficient VLSI implementation for 3-parallel linear-phase FIR digital filter of odd length " in proc. IEEE ISCAS, 2012.

[7] Y.C. Tsao, and K. Choi,,"Hardware-efficient VLSI implementation for 3-parallel linear-phase FIR digital filter of odd length " in proc. IEEE ISCAS, 2012.

[8] Q. Tian, Y. Wang, G. Liu, X. Liu, J. Diao, and Hui Xu "Hardware-efficient parallel FIR filter structure based on modified Cook-Toom algorithm "in proc. IEEE APCCAS, 2018

[9] K Anjali Rao, Abhishek Kumar, Neetesh Purohit, "Efficient implementation for 3-parallel linear- phase FIR digital odd length filters " in proc. IEEE CICT, 2020

[10] A.Kumar, S. Yadav and N. Purohit, "Exploiting coefficient symmetry in conventional polyphase FIR filters," IEEE Access, vol.7, 2019

[11] S.Y. Park, and Pramod K. Meher, "Efficient FPGA and ASIC realizations of DA-based reconfigurable FIR digital filter," IEEE Trans. Circuit and Syst.II, vol.61, no. 7, 2014.

[12] T. Vamshi Krishna, Niveditha S, Mamatha G. N, Sunil M. P. "Simulation study of brent Kung adder using cadence tool," International

Journal of Advanced Research, Ideas, and Innovations in Technology, 2018.

[13] D. K. Kahar and H. Mehta, "High-speed Vedic multiplier used Vedic mathematics, " in Proc. ICICCS, 2018.

[14] Rajesh K., Reddy G. "FPGA implementation of multiplier accumulator unit using Vedic multiplier and reversible gates " in proc. ICISC, 2019

[15] S. Nagaria, A. Singh, and V. Niranjan "Efficient FIR filter design using Booth multiplier for VLSI applications " in proc. IEEE CPCT (GUCON), 2018