IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
ISSN: 2278-2834, ISBN: 2278-8735. Volume 4, Issue 1 (Nov-Dec. 2012), PP 11-19
www.iosrjournals.org

Design and Implementation of FPGA based Low Power Digital FIR Filter

R.Raja Sulochana [1], Vasujadevi Midasala[2], S Nagakishore Bhavanam[3], Jeevan Reddy K[4]

[2]*Assistant Professor, Associate Professor*[4]*,*
[1, 2, 4] *Dept. of ECE, TeegalaKrishnaReddy Engineering College, Andhra Pradesh, India*
[3] *Assistant Professor, Dept. of ECE, University College of Engineering & Technology, ANU, Guntur,*

Abstract: *Finite impulse response (FIR) filters are widely used in various DSP applications. The low-power or low-area techniques developed specifically for digital filters can be found in. Many applications in digital communication (channel equalization, frequency channelization), speech processing (adaptive noise cancelation), seismic signal processing (noise elimination), and several other areas of signal processing require large order FIR filters ,since the number of multiply-accumulate (MAC) operations required per filter output increases linearly with the filter order, real-time implementation of these filters of large orders is a challenging task. This paper presents the methods to reduce dynamic power consumption of a digital Finite Impulse Response (FIR) filter these methods include low power serial multiplier and serial adder, combinational booth multiplier, shift/add multipliers, folding transformation in linear phase architecture and applied to fir filters to power consumption reduced thus reduce power consumption due to glitches is also reduced. This paper is implemented using XILINX ISE and hardware used is Spartan-3E and family is XC2S200E.*
Keywords: *Digital Filters, DSP, FIR, FPGA, Multipliers.*

I. Introduction

Recently there has been a trend to implement DSP functions using field programmable gate arrays (FPGAs). While application specific integrated circuits (ASICs) are the traditional solution to high performance applications, the high development costs and time-to-market factors prohibit the deployment of such solutions for certain cases. DSP processors offer high programmability, but the sequential execution nature of their architecture can adversely affect their throughput performance. As such, the reason for the rising popularity of the FPGA is due to the balance that FPGAs provide the designer in terms of flexibility, cost, and time-to-market. Digital filter structures, which are extensively used in applications such as speech processing, image and video processing, and telecommunications to name a few, are commonly implemented using FPGAs.

In signal processing, there are many instances in which an input signal to a system contains extra unnecessary content or additional noise which can degrade the quality of the desired portion. In such cases we may remove or filter out the useless samples. For example, in the case of the telephone system, there is no reason to transmit very high frequencies since most speech falls within the band of 400 to 3,400 Hz. Therefore, in this case, all frequencies above and below that band are filtered out. The frequency band between 400 and 3,400 Hz, which isn't filtered out, is known as the passband, and the frequency band that is blocked out is known as the stop band. This paper is organized as follows, section II deals with multiplier architecture ,section III discusses results in terms of delay, area and power, Section IV represents conclusion and Future work.

II. Multiplier Architecture

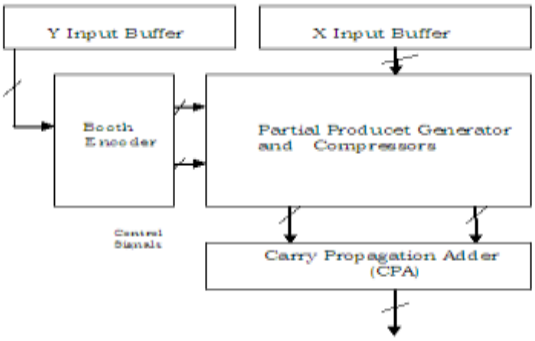


Figure 1. Block Diagram of Multiplier Architecture

A multiplier has two stages. In the first stage, the partial products are generated by the Booth encoder and the partial product generator (PPG), and are summed by compressors. In second stage, the two final products are added to form the final product through a final adder. The block diagram of traditional multiplier is depicted in Figure 1. It employs a booth encoder block, compression blocks, and an adder block. X and Y are the input buffers. Y is the multiplier which is recoded by the Booth encoder and X is the multiplicand. PPG module and compressor form the major part of the multiplier. Carry propagation adder (CPA) is the final adder used to merge the sum and carry vector from the compressor module. Each block is further explained in this chapter in detail.

2.1 Booth Encoder and Partial Product Generator

Partial product generation is the very first step in binary multiplier. Partial product generators for a conventional multiplier consist of a series of logic AND gates as shown in Figure 2. If the multiplier bit is '0', then partial product row is also zero, and if it is '1', then the multiplicand is copied as it is. From the second bit multiplication onwards, each partial product row is shifted one unit to the left. In signed multiplication, the sign bit is also extended to the left.

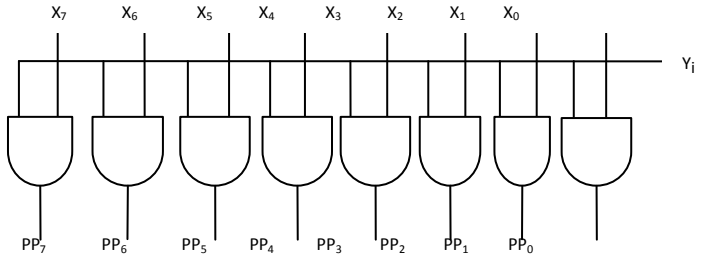


Figure 2. Partial Product generator using AND gate

2.2 Booth's Algorithm

A.D. Booth proposed Booth encoding technique for the reduction of the number of partial products [1]. This algorithm is also called as Radix-2 Booth's Recoding Algorithm. Here the multiplier bits are recoded as Zi for every ith bit Yi with reference to Yi-1 .This is based on the fact that fewer partial products are generated for groups of consecutive zeros and ones. For a group of consecutive zeros in the multiplier there is no need to generate any new partial product. We only need to shift previously accumulated group partial product one bit position to the right for every 0 in the multiplier.

The radix-2 algorithms results in these observations:

(a) Booth observed that whenever there was a large number of consecutive ones, the corresponding additions could be replaced by a single addition and a subtraction $2j + 2j\text{-}1 + \ldots\ldots\ldots\ldots + 2i+1 + 2i = 2j+1 - 2i$.

(b) The longer the sequence of ones, the greater the savings.

(c) The effect of this translation is to change a binary number with digit set [0, 1] to a binary signed-digit number with digit set [-1, 1].

The Radix-2 Booth algorithm :

Table 1. Radix-2 Booth recoding

Yi	Yi-1	Zi	Explanation
0	0	0	No string of 1s in sight
0	1	1	End of string of 1s in Y
1	0	1	Beginning of string of 1s in Y
1	1	0	Continuation of string of 1s in Y

In this algorithm the current bit is Yi and the previous bit is Yi-1 of the multiplier Yn-1 Yn-2…… Y1 Y0 are examined in order to generate the ith bit Zi of the recoded multiplier Zn-1 Zn-2…….Z1 Z2. The previous bit Yi-1 serves only as the reference bit. The recoding of the multiplier bits need not be done in any predetermined order and can be even done in parallel for all bit positions. The observations obtained from the radix-2 Booth recoding is listed below: It reduces the number of partial products which in turn reduces the hardware and delay required to sum the partial products. It adds delay into the formation of the partial products.

- It works well for serial multiplication that can tolerate variable latency operations by reducing the number of serial additions required for the multiplication.
- The number of serial additions depends on the data (multiplicand)

- Worst case 8-bit multiplicand requires 8 additions.
- 01010101 ⇔ 1 -1 1 -1 1 -1 1 -1
- Parallel systems generally are designed for worst case hardware and latency requirements. Booth-2 algorithm does not significantly reduce the worst case number of partial products.

2.3 Modified Booth Algorithm

The radix-2 disadvantages can be eliminated by examining three bits of Y at a time rather than two. The modified Booth algorithm is performed with recoded multiplier which multiplies only +a and +2a of the multiplicand, which can be obtained easily by shifting and/or complementation. The main advantage of the modified Booth algorithm is that it reduces the partial products to n/2.The truth table for modified Booth recoding is shown below:

Table 2. Radix-4 Booth Recoding

Yi+1	Yi	Yi-1	Zi	Explanation
0	0	0	0	No string of 1s in sight
0	0	1	1	End of string of 1s
0	1	0	1	Isolated 1
0	1	1	2	End of string of 1s
1	0	0	-2	Beginning of string of 1s
1	0	1	-1	End a string, begin a new one
1	1	0	-1	Beginning of string of 1s
1	1	1	-0	Continuation of string of 1s

The following gives the algorithm for performing sign and unsigned multiplication operations by using radix-4 Booth recoding.

Algorithm: (for unsigned numbers)

- Pad the LSB with one zero
- Pad the MSB with two zeros if n is even and one zero if n is odd
- Divide the multiplier into overlapping groups of 3-bits
- Determine partial product scale factor from modified Booth-2 encoding table
- Compute the multiplicand multiplies
- Sum partial products

Algorithm: (for signed numbers)

- Pad the LSB with one zero
- If n is even don't pad the MSB (n/2 PP's)
- Divide the multiplier into overlapping groups of 3-bits
- Determine partial product scale factor from modified Booth-2 encoding table
- Compute the multiplicand multiplies
- Sum partial products

Booth recoding is fully parallel and carry free. It can be applied to design a tree and array multiplier, where all the multiples are needed at once. Radix-4 Booth recoding system works perfectly for both signed and unsigned operations.

2.4 Compressors

A Carry-Save Adder (CSA) is a set of one-bit full adders, without any carry-chaining.Therefore, an n-bit CSA receives three n-bit operands, namely a (n-1)..a (0), b (n-1)..b (0), and cin (n-1)..cin (0), and generates two n-bit result values, sum (n-1)..Sum (0) and cout (n-1).. cout (0)

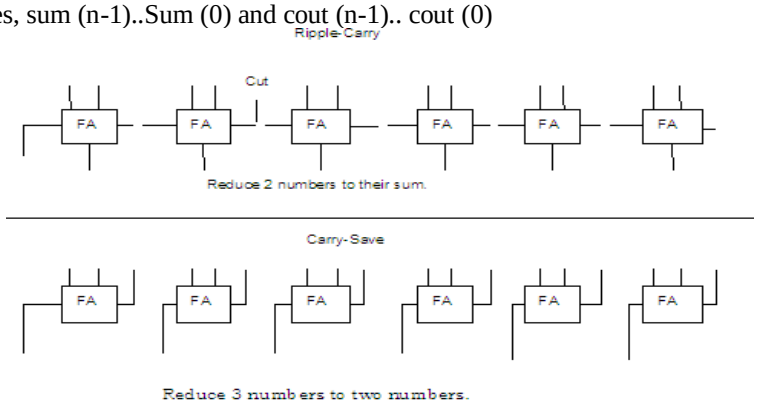


Figure 3. Carry Save Adders

A carry save adder tree can reduce n binary numbers to two numbers having the same sum in O (log n) levels. Carry save adder is also called a compressor and a Wallace Tree is constructed with CSAs. Wallace trees are CSAs in a tree structure used as a compressor. The most important application of a carry-save adder is to add the partial products in integer multiplication. From CSA separate sum and carry vector are obtained. In CSA, the output carry is not passed to the neighboring cell but is saved and passed to the cell one position down.

2.5 Carry Propagation Adder

The final step in completing the multiplication procedure is to add the final terms in the final adder. The Carry Propagation Adder, CPA, is a final adder used to add the final carry vector to the final sum vector partial products to give the final multiplication result. This is normally called a "Vector-merging" adder. The choice of the final adder depends on the structure of the accumulation array. Various fast adders can be used as CPA. Some of them are Carry look-ahead adder, Simple carry skip adder, Multi level carry skip adder, Carry-select adder, Conditional sum adder and Hybrid adder. A Carry look-ahead Adder is an adder used in digital logic. All the carry outputs are calculated at once by specialized look-ahead logic. But requires generate and propagate signals. Simple carry skip adders looks for the cases in which carry out of a set of bits are identical to carry in. Circuits for binary adders to efficiently skip a carry bit over two or more bit positions with two or more carry-skip paths is called multilevel carry skip adders. In the 4-bit carry select adder there are two 4-bit adders each of which takes a different preset carry-in bit. The two sums and carry-out bits that are produced are then selected by the carry-out from the previous stage. In conditional sum adder, sum and carry outputs at the first stage assume the previous carry to be zero and sum and carry outputs at the second stage assume the previous carry to be one. For CPA we can also combine any of these adders as a hybrid adder.

III. RELATED WORK

Fast multipliers are imperative for high speed and low power signal processing systems and hence much thrust have been given to different design techniques. As explained in Chapter 2 multipliers consists of a Booth encoder, compressors, and carry propagation adders. The speed of the multiplier can be enhanced by reducing the number of partial products and thus the Booth algorithm plays a major role. In this chapter, we discuss about the related literature works for number of Booth encoder and the selector logic and the several design methods used to reduce the partial products.Booth encoding is a technique that leads to smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is the standard technique used in chip design, and provides significant improvements over the "long multiplication" technique. The widely used Booth algorithm is the radix-4 based modified Booth algorithm proposed by McSorley where it reduces the partial products into half. As the number of partial products reduces the number of CSAs required for the compression module, the height of the Wallace tree is also reduced.

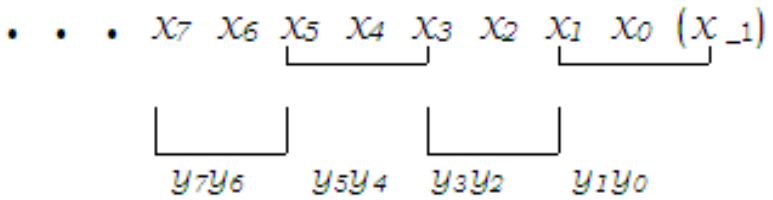


Figure 4. Modified Booth recoding pattern

Modified Booth algorithm's basic idea is that the bits Yi and Yi-1 are recoded into Zi and Zi-1, while, Yi-2 serves as reference bit. In a separate step, Yi-2 and Yi-3 recoded into Zi-2 and Zi-3 with, Yi-4 serving as reference bit. This signifies that the modified Booth's encoding partitions input Y into a group of 3-bits with 1-bit overlap and generates the following five signed digits, 2, 1, 0, -1 and -2. Encoding on the each group reduces the number of partial products by factor of 2.

Operations on the encoded digits performed with multiplier input X is illustrated in the following table.

Table 3. Partial Product Selections and Operations

Recorded digit	Booth's operation on X	Y2i-1Y2iY2i+1
0	Add 0 to PP	(000,111)
+1	Add X to PP	(001,010)
+2	Shift X left & add to PP	(011)
-1	Add 2's complementary X to PP	(101,110)
-2	2's complementary X & shift-add	(100)

3.1 Booth Encoder and PPG proposed by Ohkubo

Ohkubo, et al., developed a CMOS multiplier using pass transistor multiplexer.

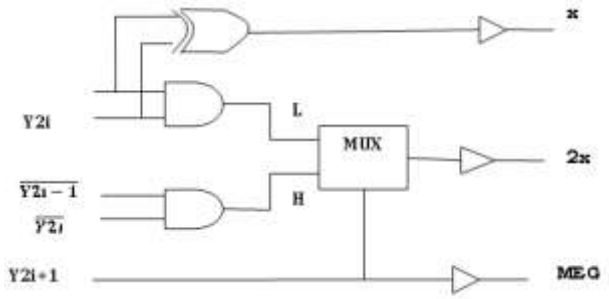


Figure 5. Booth encoder

There were three control signals for complement, shifting and direction. The complement signal was generated by XOR function and the Shift by the AND and MUX operation. The partial products were obtained by the NAND and XOR operations.

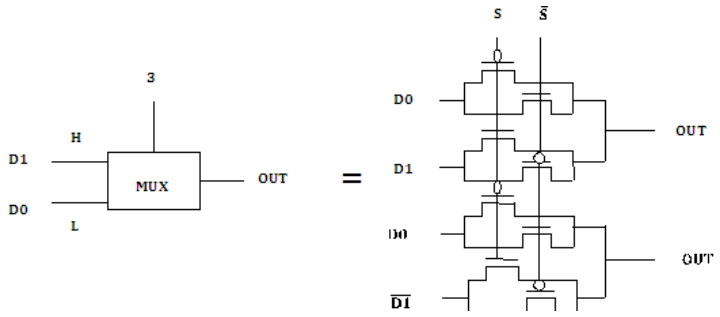


Figure 6. Pass- transistor multiplexer circuit

The multiplexer used in Booth encoder itself used 8 transistors which used separate transistors to design nMOS and pMOS.

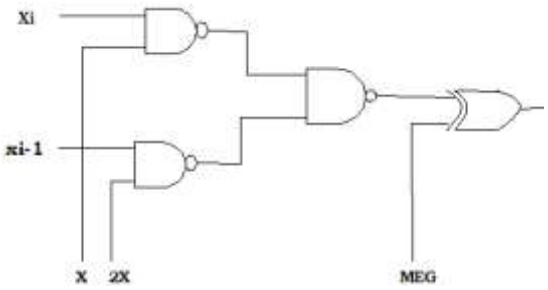


Figure 7. Partial Product Generator

The PPG was implemented using NAND and XOR gates. Here the inputs were the control signals generated by the Booth encoder and these signals were used to output the data inputs Xi and Xi-1.

Ohkubo, et al., work provided a speed advantage over conventional CMOS circuits because the critical path gate stages were minimized using pass transistor multiplexer. The drawback of Ohkubo's work was that it consisted of more transistors and it produced unnecessary glitches by the partial product generator. According to his design; any change in the value of the partial products also caused a change all along the multiplier array, and the final adder. This energy dissipation associated with the glitches in the modified Booth algorithm was an important portion of the total energy dissipation of the whole multiplier and the issue has been dealt by Fried in his work. The total number of transistors for the encoder and selector logic added up to 48 transistors which occupied a large amount of space.

3.2 Booth Encoder and PPG proposed by Goto

Goto, et al., was successful in reducing the number of transistors when compared with Ohkubo's work . In Goto's work, there were two control signals used for generation of sign of the partial product: Mj (for negative) and PLj (for positive). The modified Booth Selector required four multiplexers which consumed a large area.

Booth encoder and PPG module constitute one third part of the entire multiplier design. In fact, Goto's work used the multiplicand as the select signals in the selector, which was very different from the conventional method which used the encoded signals as the select signals. However, encoded signals ran through the two multiplexers in series, thus incurred more delay than some other multipliers which were developed in later periods. Five gates were needed on the critical path. The truth table for the Booth encoding as per Goto's work is given in Table.

Table 4. Booth encoding

Inputs		Usual			Sign select			
b_{j+1} b_j b_{j-1}	Func	X_j	$2x_j$	M_j	X_j	$2x_j$	pLj	M_j
0 0 0	0	0	0	0	0	1	0	0
0 0 1	+A	1	0	0	1	0	1	0
0 1 0	+A	1	0	0	1	0	1	0
0 1 1	+2A	0	1	0	0	1	1	0
1 0 0	-2A	0	1	1	0	1	0	1
1 0 1	-A	1	0	1	1	0	0	1
1 1 0	-A	1	0	1	1	0	0	1
1 1 1	0	0	0	1	0	1	0	0

P=A x B (j=0,2,4.....n-4,n-2)

$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i$

$B = -b_{n-1}2^{n-1} + \sum_{j=0}^{n-2} b_j 2^j$

PL_j: Positive partial product
M_j: Negative partial product
X_j: Not Doudled
$2X_j$: Doubled

Here inputs are bj+1, bj and bj-1. The Booth encoder had four outputs and the selector had two outputs. The design also used a number of inverters which resulted in power consumption.

3.3 Booth Encoder and PPG proposed by Fried

The unnecessary glitches from Goto's design were eliminated by Fried's design of a new two-gate-delay implementation of the Booth encoder and partial product generator. He proposed two approaches to eliminate the unnecessary glitches in the Booth algorithm. One was to latch all the partial products and allow them to change only after steady-state was reached in the encoder and partial product generator. This was achieved by using a clock derivative from the global clock, whose duty cycle was defined according to the slowest path in the Booth implementation.

However, this approach required large area and dissipates a lot of energy by itself. The second approach was to synchronize all the paths in Booth encoder and partial product generator.

3.4 Booth Encoder and PPG proposed by Gro□schädl

The partial product generator developed by Groβschädl was used for two different types of operands; integers and binary polynomials. For integer mode it was done by modified Booth recoding technique and for polynomial by a digital serial polynomial multiplier.

3.5 Booth Encoder and PPG proposed by Cho

In 2003, Cho, et al., developed a new Booth encoder and the selector with a fewer number of components. They developed a new encoder based on the modified Booth algorithm. Table 6. shows the truth table of the operations developed by Cho.

Table 5. Truth Table for Booth encoding

Ym+1	Ym	Ym-1	Booth Op.	Dir.	Sht.	Add.
0	0	0	0x	0	0	0
0	0	1	1x	0	-	1
0	1	0	1x	0	-	1
0	1	1	2x	0	1	0
1	0	0	-2x	1	1	0
1	0	1	-1x	1	-	1
1	1	0	-1x	1	-	1
1	1	1	-0x	1	0	0

In their design they described Booth function as three basic operations, which they called 'direction', 'shift', and 'addition' operation.

Direction determined whether the multiplicand was positive or negative, shift explained whether the multiplication operation involved shifting or not and addition meant whether the multiplicand was added to partial products. The expressions for Booth encoding were stated below as :

Direction,

Dm = Ym+1;

Shift, Sm = Ym-1 • (Ym+1 ⊕ Ym) + Ym-1' • (Ym+1 ⊕ Ym) = Ym+1 ⊕ Ym;

Addition, Am = Ym-1 ⊕ Ym;

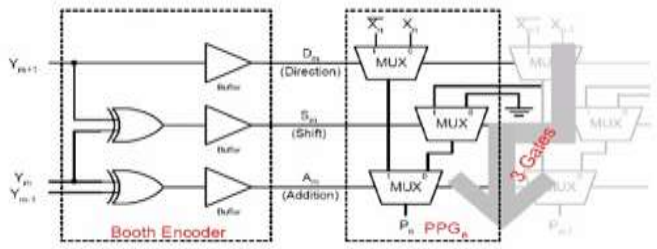


Figure 8. Booth encoder and PPG

The Booth encoder was implemented using two XOR gates and the selector using 3 MUXes and an inverter which counted to a total of 40 transistors. Careful optimization of the partial-product generation can lead to some substantial delay and hardware reduction. Keeping this in mind, some designs are proposed in Chapter 4.

Linear-Phase-Folding Architecture Fir Filter Based Booth Multiplier Booth Multiplier If the phase of the filter is linear, the symmetrical architecture can be used to reduce the multiplier operation. Comparing Fig.1 and Fig.4, the number of multipliers can be reduced half after adopting the symmetrical architecture. But number of adders remains constant and it is the basic model to develop the proposed architecture.

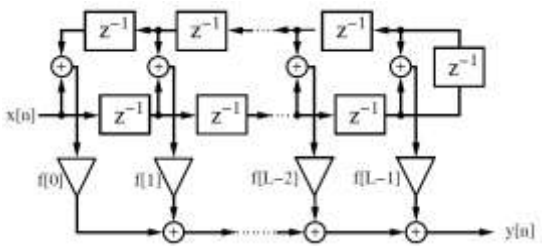
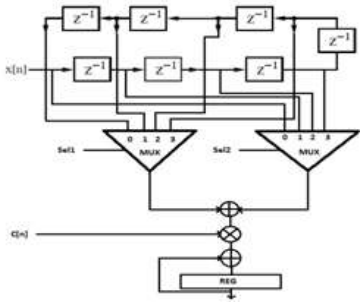


Figure 9. Linear-phase filter with reduced number of multipliers.



Many algorithm transformation techniques are available for optimum implementation of the digital signal processing algorithms. Reducing the implementation area is important for complex algorithms, such as the receiver equalizer in the metal link digital communications. For example Folded architectures provide a trade-off between the hardware speed and the area complexity. The folding transformation can be used to design time-multiplexed architectures using less silicon area. Power consumption can be even reduced with the folding transformation. Thus folding is a technique to reduce the silicon area by time multiplexing many operations (e.g. multiply & add) into single function units Folding introduces registers Computation time increased. Fig.5 show linear phase folding architecture fir filter.

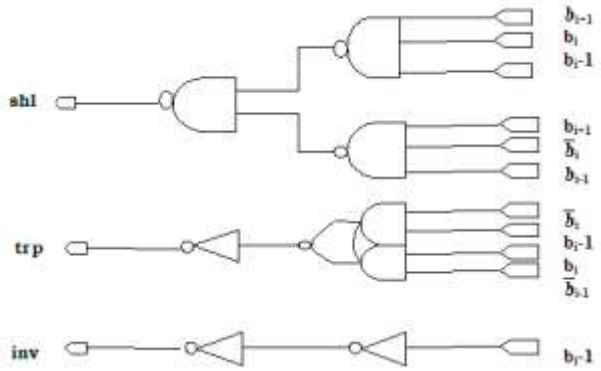


Figure 10. Booth encoder

For encoding multiplier was partitioned into overlapping groups of three bits (bi+1, bi, bi-1) with i= 0, 2, 4, 6, …. Each group had its own encoder circuit which produced the control signal inv (invert), trp (transport, and shl (shift left). When control signal inv = 1 then the PP is negative. When control signal trp =1 means the PP = ± A (no shift left). When shl = 1, a one bit left-shift was performed. The PP= 0 was generated by trp = shl = 0.

IV. Results

Table 3. Power consumption of proposed filters

Architecture	Delay(ns)	No. of LUT's	Power(mw)
1.Mac Fir Filter Based Booth Multiplier	5.20	510 out of 9312 5%	0.87
2. Linear-Phase-Folding Architecture Fir Filter Based Booth Multiplier	7.336	541 out of 9312 5%	0.90
3. Mac Fir Filter Based Low Power Serial Multiplier And Serial Adder	6.547	533 out of 9312 5%	0.88
4. FIR Filter Based Shift/ Add Multiplier	9.181	8 out of 9312 0%	0.86

V. Conclusion AND Future Work

In this paper, we proposed four new architectures, Mac Fir Filter Based Booth Multiplier, Linear-Phase-Folding Architecture Fir Filter Based Booth Multiplier, Mac Fir Filter Based Low Power Serial Multiplier and Serial Adder, FIR Filter Based Shift/ Add Multiplier, and the delay of these architectures are respectively 5.20ns, 7.336ns, 6.457ns and 9.181 respectively. And power of these architecture are respectively 0.87mw, 0.90mw, 0.88mw and 0.86mw are respectively. By comparing all the architectures from the table, we can say that Mac Fir Filter Based Booth Multiplier has less delay i.e. 5.20 when compared to other architectures. At the same time we can see lower power consumption i.e. 0.86 in FIR Filter Based Shift/ Add Multiplier. Where the others are varied with 0.01 mw which don't imply much on the other architectures. But the same case is not followed in delay timings, where the delay values are 9.181ns, 7.336ns & 6.547ns.

The present work on the new multiplier architecture can be further extended in various directions. The design can be simulated to check the power consumption. Other methods can be incorporated with this to further improve the delay. In order to completely analyze the performance, the circuit can be extended to chip level where the delays due to wiring, interconnects and PAD are included.

Acknowledgement

R. Raja Sulochana would like to thank Associate Prof K. Jeevan Reddy, who had been guiding through out to complete the work successfully, and would also like to thank the HOD, ECE Department and other Professors for extending their help & support in giving technical ideas about the paper and motivating to complete the work effectively & successfully.

References

[1] Huang, Z. and M.D. Ercegovac, 2002. Signal gating for low-power array multiplier design. In: IEEE International Symposium on Circuits and Systems, ISCAS'2002, IEEE Computer Society, Washington DC., USA, pp: 489-492. Doi: 10.1109/ISCAS.2002.1009884.

[2] Manish Bhardwaj, R. Min and A.P. Chandrakasan, 2001. Quantifying and enhancing power awareness of VLSI systems. IEEE Trans. VLSI Syst.,9: 757-772.

[3] Meier, P.C.H., R.A. Rutenber and L.R. Carley,1999. Inverse polarity techniques for highspeed/low-power multipliers. In: International Symposium on Low Power Electronics and Design, ISLPEAD'1999, IEEE Computer Society,Washington DC., USA, pp: 264-266.

[4] Kim, S. and M.C. Papaefthymiou, 2000.Reconfigurable low energy multiplier formultimedia system design. In: Proceedings of IEEEComputer Society Workshop on VLSI, 2000, IEEEComputer Society, Washington DC., USA,pp: 129-134. Doi: 10.1109/IWV.2000.844541.

[5] Sinha, A., A. Wang and A. Chandrakasan, 2002. Energy scalable system design. IEEE Trans. VLSI Syst., 10: 135-145.

[6] Di, J., J.S. Yuan and R.F. DeMara, 2006.Improving power-awareness of pipelined array multipliers using 2-dimensional pipeline gating and its application to FIR design. Integration the VLSI Journal.,39(2):9 112.doi:10.1016/j.vlsi.2004.08.2

[7] Di, J. and J.S. Yuan, 2003. Power-aware pipelined multipliers design based on 2-dimensional pipeline gating, Proceedings of 13th ACM Great Lakes Symposium on VLSI 2003,Washington, DC, USA, 64-67.

[8] Hoang, Q.D., B.R. Zeydel and V.G. Oklobdzija, 2006. Energy optimization of pipelined digital systems using circuit sizing and supply scaling. IEEE Trans. VLSI Syst., 14: 122-134.

[9] Lee, K.H. and C.S. Rim, 2000. A hardware reduced multiplier for low power design. In: Proceedings of the 2nd IEEE Asia-Pacific Conference on ASICs, 2000, IEEE Computer Society, Washington DC., USA, pp: 331-334. Doi: 10.1109/APASIC.2000.896975.

[10] Parhi, K.K., 1999. VLSI Digital Signal Processing Systems. John Willey and Sons Inc., USA.