IJIREEICE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

NCIET- 2020

National Conference on Innovations in Engineering & Technology

Govt., Polytechnic College, Palakkad, Kerala

Vol. 8, Special Issue 1, February 2020

High Speed and Low Area FIR Filter Implementation Based on Dadda Multiplier Design

Irshadali CM[1], Asmabi V[2]

PG Scholar Department of Electronics and Communication Engineering, Al Ameen Engineering College Kulappully, Shoranur[1]

Assistant Professor, Department of Electronics and Communication Engineering,

Al Ameen Engineering College Kulappully, Shoranur[2]

Abstract: Today every circuit has to face the power consumption issue for both portable devices aiming at large battery life and high-end circuits avoiding cooling packages and reliability issues that are too complex. It is generally accepted that during logic synthesis power tracks well with area. This means that a larger design will generally consume more power. The multiplier is an important kernel of digital signal processors. Because of the circuit complexity, the power consumption and area are the two important design considerations of the multiplier. In this paper a High Speed & low area architecture for the Dadda Multiplier is proposed. For getting the High Speed & lower area architecture, the modifications made to the conventional architecture consist of the reduction in switching activities of the major blocks of the multiplier, which includes the reduction in switching activity of the adder and counter. This architecture is used in FIR Filter Design. The simulation result for 8 bit multipliers & four tap Filters shows that the proposed low Area & Delay architecture lowers the total Area & Delay when compared to the Booth Multiplier and Dadda Multiplier architecture based Filter.

Keywords: Finite Impulse Response (FIR), Dadda Multiplier, Booth Multiplier.

I. INTRODUCTION

The basic operation in digital signal processing is filtering. This operation is widely used in many electronic devices to cancel part of signal that is redundant or damages the signal. Filters have two uses: signal separation and signal restoration. Signals which are corrupted by interference and noise require separation techniques. A device for measuring the electrical activity of a baby's heart inside the mother's womb will be corrupted by breath signal and heartbeat signal of the mother. At such times filters are used to separate the signals and analyse them individually. When signal gets distorted the process of signal restoration is used. Audio recording made with poor equipment is filtered to give better sound signal output than the original it previously produced. They can be either solved by analog or digital filters. Analog filters are cheap, fast, and have a large dynamic range in both 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 analog filters. 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. It is common in DSP to say that a filter's input and output signals are in the *time domain*. This is because signals are usually created by sampling at regular intervals of *time*. But this is not the only way sampling can take place. The second most common way of sampling is at equal intervals in *space*. Many other domains are possible; however, time and space are by far the most common. When you see the term *time domain* in DSP, remember that it may actually refer to samples taken over time, or it may be a general reference to any domain that the samples are taken in. As shown in Fig. 1.1 and 1.2, every linear filter has an impulse response and frequency response**.** Each of these responses contains complete information about the filter, but in a different form. If one of the two is specified, the other is fixed and can be directly calculated. Both of these representations are important, because they describe how the filter will react under different circumstances.

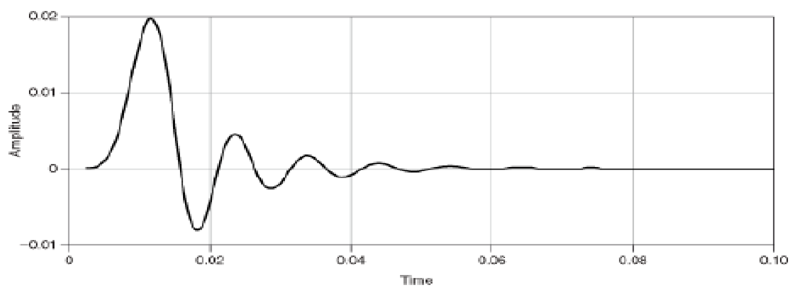


Fig. 1. Impulse response

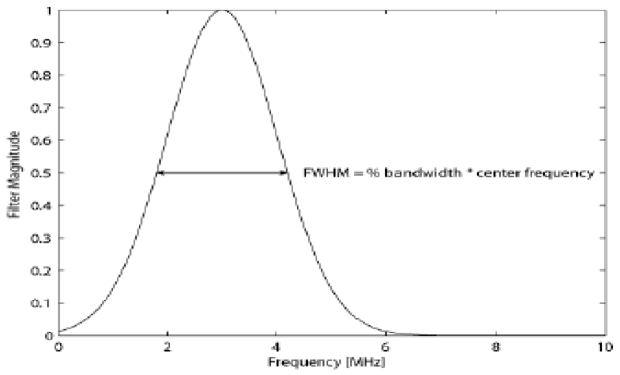


Fig. 2. Frequency response

The most straightforward way to implement a digital filter is by convolving the input signal with the digital filter's impulse response. All possible linear filters can be made in this manner. When the impulse response is used in this way, filter designers give it a special name: the filter kernel. There is also another way to make digital filters, called recursion. When a filter is implemented by convolution, each sample in the output is calculated by weighting the samples in the input, and adding them together. Recursive filters are an extension of this, using previously calculated values from the output, besides points from the input. Instead of using a filter kernel, recursive filters are defined by a set of recursion coefficients. The important point is that all linear filters have an impulse response, even if you don't use it to implement the filter. To find the impulse response of a recursive filter, simply feed in an impulse, and see what comes out. The impulse responses of recursive filters are composed of sinusoids that exponentially decay in amplitude. In principle, this makes their impulse responses infinitely long. However, the amplitude eventually drops below the round-off noise of the system, and the remaining samples can be ignored. Because of this characteristic, recursive filters are also called Infinite Impulse Response or IIR filters. In comparison, filters carried out by convolution are called Finite Impulse Response or FIR filters.

II. EXISTING SYSTEM

A. Booth multiplier:

Booth Multiplier is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product. If the multiplier is very large, then a large number of multiplicands have to be added. In this case the delay of multiplier is determined mainly by the number of additions to be performed. If there is a way to reduce the number of the additions, the performance will get better. Booth algorithm is a method that will reduce the number of multiplicand multiples. For a given range of numbers to be represented, a higher representation radix leads to fewer digits. Since a k-bit binary number can be interpreted as K/2-digit radix-4 number, a K/3-digit radix-8 number, and so on, it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication.

1. Booth Multiplication Algorithm (radix – 4)

One of the solutions realizing high speed multipliers is to enhance parallelism which helps in decreasing the number of subsequent calculation stages. The Original version of Booth's multiplier (Radix – 2) had two drawbacks.

- The number of add / subtract operations became variable and hence became inconvenient while designing Parallel multipliers.
- The Algorithm becomes inefficient when there are isolated 1s.

These problems are overcome by using Radix 4 Booth's Algorithm which can scan strings of three bits with the algorithm given below. The design of Booth's multiplier in this project consists of Modified Booth Encoded (MBE), sign extension corrector, partial product generators and finally a Wallace Tree Adder. This Dadda Multiplier technique is to increase speed by reducing the number of partial products by half. Since an 8-bit Dadda Multiplier is used in this project, so there are only four partial products that need to be added instead of eight partial products generated using conventional multiplier. The architecture design for the modified Booths Algorithm is shown in the figure4.2

1. Modified Booth Encoder (MBE)

Modified Booth encoding is most often used to avoid variable size partial product arrays. Before designing a MBE, the multiplier B has to be converted into a Radix-4 number by dividing them into three digits respectively according to Booth Encoder Table given afterwards. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier. The figure above shows that the multiplier has been divided into four partitions and hence that mean four partial products will be generated using Dadda Multiplier approach instead of eight partial products being generated using conventional multiplier. Let's take an example of converting an 8-bit number into a Radix-4 number. Let the number be -36 = 1 1 0 1 1 1 0 0. Now we have to append a '0' tothe LSB.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
Vol. 8, Special Issue 1, February 2020

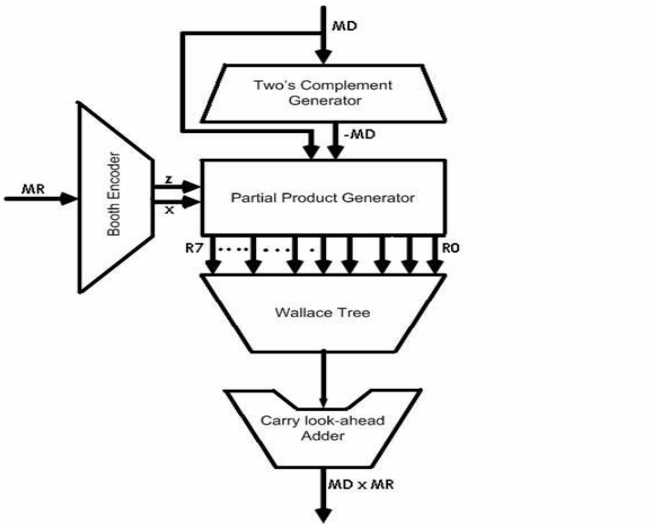


Fig. 3. Booth Algorithm architecture

Hence the new number becomes a 9-digit number, that is 1 1 0 1 1 1 0 00. This is now further encoded into Radix-4 numbers according to the following given table. Starting from right we have 0*Multiplicand, -1*Multiplicand, 2*Multiplicand, -1*Multiplicand**.**

Table. 1. Modified Booth Encoder's table to generate M, 2M, 3M control signal

B_{n+1}	B_n	B_{n-1}	Z_n	Partial Products	1M	2M	3M
0	0	0	0	0	1	1	0
0	0	1	1	1*Multiplicand	0	1	0
0	1	0	1	1*Multiplicand	0	1	0
0	1	1	2	2*Multiplicand	1	0	0
1	0	0	-2	-2*Multiplicand	1	0	1
1	0	1	-1	-1*Multiplicand	0	1	1
1	1	0	-1	-1*Multiplicand	0	1	1
1	1	1	0	0	1	1	0

Table 4.1 show B_{n+1}, $B_n$and B_{n-1}which are three bits wide binary numbers of the multiplier Bin which Bn+1 is the most significant bit (MSB) and Bn-1 is the least significant bit (LSB). Zn is representing the Radix-4 number of the 3-bit binary multiplier number. For example, if the 3-bit multiplier value is "111", so it means that multiplicand A will be 0. And it's the same for others either to multiply the multiplicand by -1, -2 and so on depending on 3-digit number. And thing to note is generated numbers are all of 9-bit. From the table 4.1, the M, 2M and 3M are the elect control signals for the partial product generator. It will determine whether the multiplicand is multiplied by 0, -1, 2 or -2. M and 2M are designed as an active low circuit which means if let's say the multiplicand should be multiplied by 1 then the M select signal will be set to low "0" whereas If the multiplicand should be multiplied by 2 then the 2M select signal will be set to low "0". The 3M is representing the sign bit control signal and its active high circuit which means if the multiplicand should be multiplied by -1 or -2, then the sign, 3Mwill be set to high "1".

2. *Partial Product Generator (PPG):* Partial product generator is the combination circuit of the product generator and the 5 to 1 MUX circuit. Product generator is designed to produce the product by multiplying the multiplicand A by 0, 1, -1, 2 or -2. A 5 to 1 MUX is designed to determine which product is chosen depending on the M, 2M, 3M control signal which is generated from the MBE. For product generator, multiply by zero means the multiplicand is multiplied by "0". Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value & multiply by "2" means just shift left the multiplicand by one place.

B. Results and discussion

Table.2 Comparison of Array Multiplier and Booth Multiplier

Type of multiplier	Parameters		
	Area	Delay	Power
Array Multiplier	58,566	72.56 ns	959 mW
Booth Multiplier	39,442	42.66 ns	619 mW

IJIREEICE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
NCIET- 2020
National Conference on Innovations in Engineering & Technology
Govt., Polytechnic College, Palakkad, Kerala
Vol. 8, Special Issue 1, February 2020

The comparison is implemented from two aspects. One is the comparison of redundancy and complexity and the other one is the comparison of implementation overhead on delay and area. The results for each solution are sketched in table below.
The motivation of Booth's multiplication scheme is to increase the speed of multiplication process. As compared to conventional methods Booth's multiplication helps to reduce the number of iteration steps and results in faster computation.
When taken into consideration the examples of Radix-2 and Radix-4 multiplication, it can be concluded that, Radix-4 Booth multiplication halves the number of partial products and helps to increase the speed of multiplication operation. This algorithm can be extended to Radix-8 for which complexity is somewhat high, but the generated partial products will reduce to 'n/3'.

C. Drawbacks of existing system

By comparing with the proposed Dadda Multiplier system we can further reduce the the various parameters like Area, Delay and power consumption

III. PROPOSED SYSTEM

Dadda method for partial product reduction uses only necessary reduction determined by the Wallace table shown in table 1 [2, 3,4]. It uses an approach to increase the number of half adders and to decrease the number of full adders required for overall reduction of partial products. The detail of Dadda technique is given in the references [2, 3, 4, 5, 9]. The Dadda method keeps as these are all right columns having height less than or equal to the necessary bits required at a stage after reduction. This is repeated at every stage. This multiplier uses a modified Dadda technique for partial product reduction. The main difference between the two lies in the partial product reduction method. Dadda's method prefers use of half adders whereas the proposed Full-Dadda multiplier uses fulladder with preference except at last stage.

A. General rules

The rules for partial products reduction used in the proposed multiplier are explained in a simple way as follow:
1) Keep to the next stage as these are all right columns having height less than or equal to the necessary bits required at a stage after reduction. These necessary bits are determined according to table 1. This is similar step as in Dadda method. It is done at every stage.
2) For the reduction of partial products, use full adders that is (3, 2) compressors preferably over the use of (2, 2) compressor (half adder). Use half adders only when full adders can't be used that is only when two bits are available for further reduction.
3) At last stage of reduction where 3 bits to 2 bits reduction is to be done; use half adder preferably. This last stage rules are also the same as of Dadda method. The height of a column at this stage is 3. Therefore, there is no need of long counting of the column bits and hence it is easy to give preference to half adder.

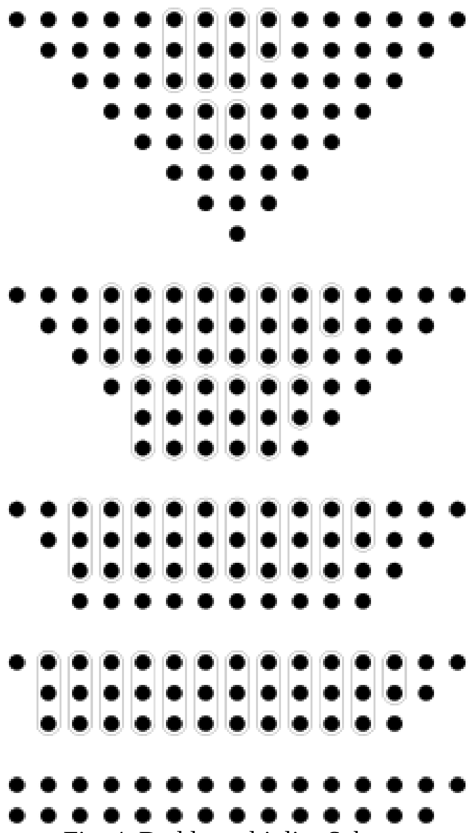
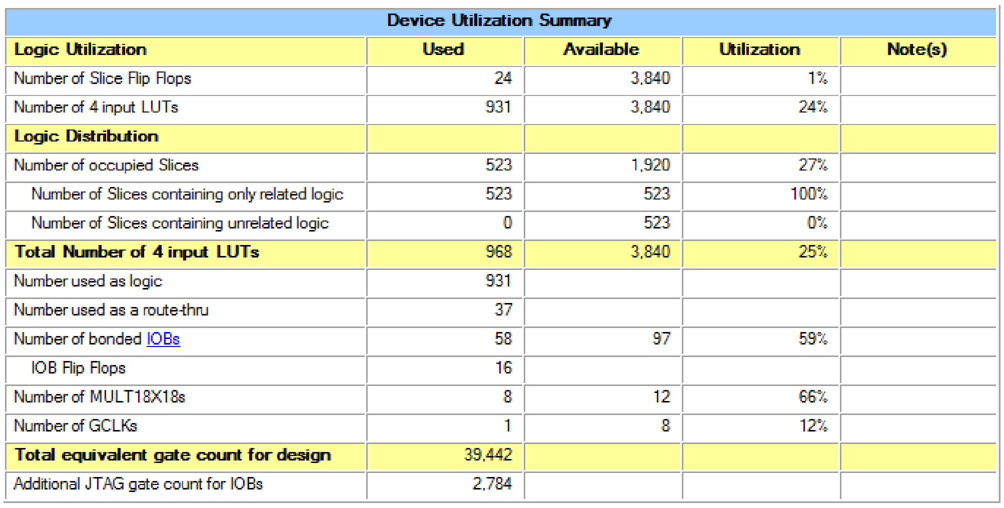


Fig. 4. Dadda multiplier Scheme

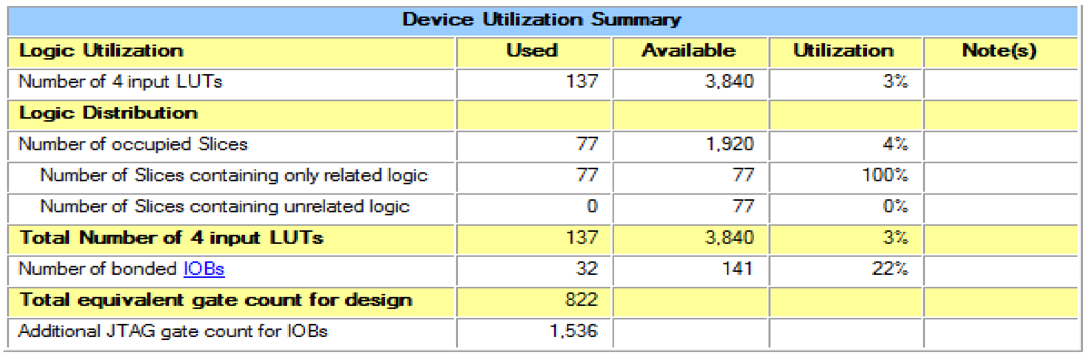
B. Results and discussion

The output waveforms and synthesis reports of both Booth Mutiplier and Dadda Multiplier observed from the Xilinx synthesis software are given below.



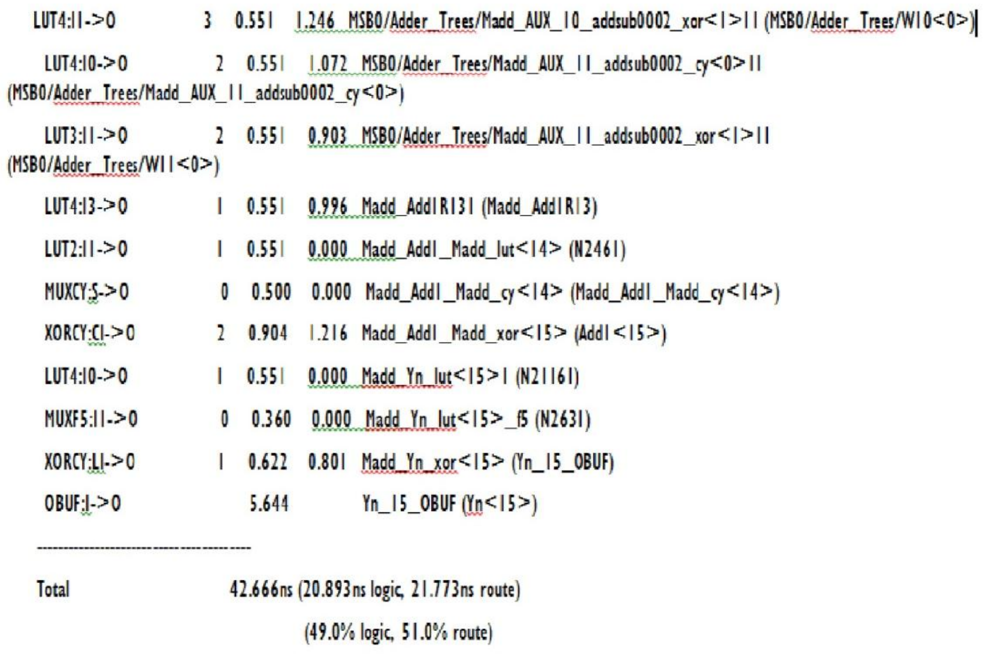
Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	24	3,840	1%	
Number of 4 input LUTs	931	3,840	24%	
Logic Distribution				
Number of occupied Slices	523	1,920	27%	
Number of Slices containing only related logic	523	523	100%	
Number of Slices containing unrelated logic	0	523	0%	
Total Number of 4 input LUTs	968	3,840	25%	
Number used as logic	931			
Number used as a route-thru	37			
Number of bonded IOBs	58	97	59%	
IOB Flip Flops	16			
Number of MULT18X18s	8	12	66%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	39,442			
Additional JTAG gate count for IOBs	2,784			

Fig. 5. Device utilization summary of Booth Multiplier



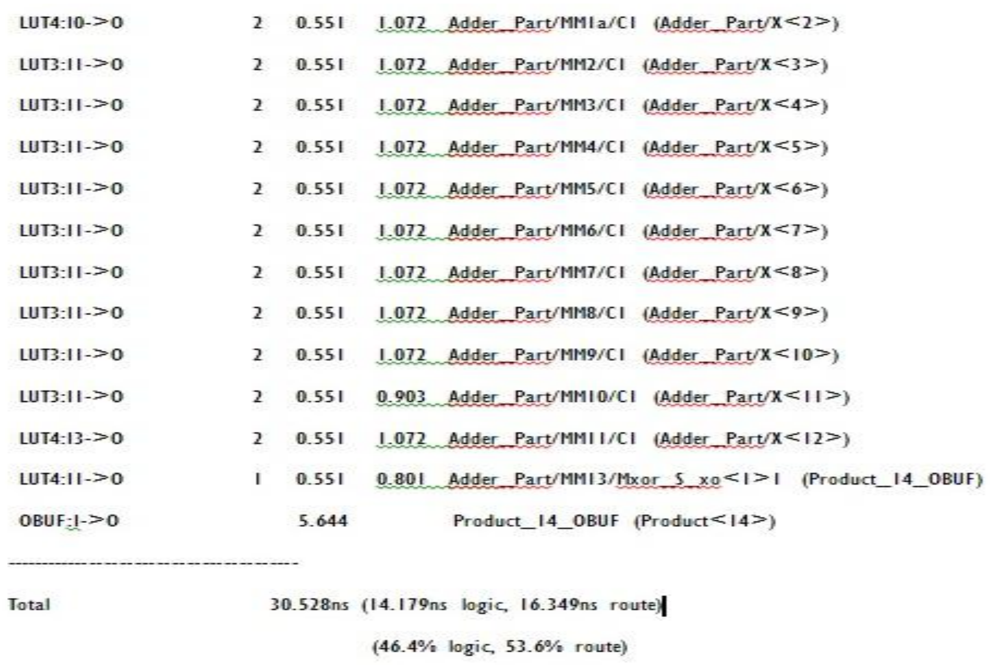
Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	137	3,840	3%	
Logic Distribution				
Number of occupied Slices	77	1,920	4%	
Number of Slices containing only related logic	77	77	100%	
Number of Slices containing unrelated logic	0	77	0%	
Total Number of 4 input LUTs	137	3,840	3%	
Number of bonded IOBs	32	141	22%	
Total equivalent gate count for design	822			
Additional JTAG gate count for IOBs	1,536			

Fig. 6. Device utilization summary of Dadda Multiplier



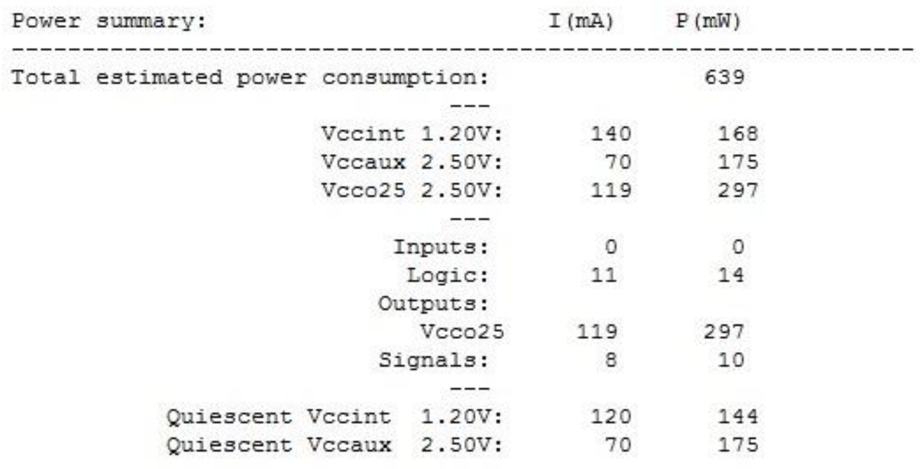
```
   LUT4:I1->O        3   0.551   1.246  MSB0/Adder_Trees/Madd_AUX_10_addsub0002_xor<1>11 (MSB0/Adder_Trees/W10<0>)
    LUT4:I0->O        2   0.551   1.072  MSB0/Adder_Trees/Madd_AUX_11_addsub0002_cy<0>11
(MSB0/Adder_Trees/Madd_AUX_11_addsub0002_cy<0>)
    LUT3:I1->O        2   0.551   0.903  MSB0/Adder_Trees/Madd_AUX_11_addsub0002_xor<1>11
(MSB0/Adder_Trees/W11<0>)
    LUT4:I3->O        1   0.551   0.996  Madd_Add1R131 (Madd_Add1R13)
    LUT2:I1->O        1   0.551   0.000  Madd_Add1_Madd_lut<14> (N2461)
    MUXCY:S->O         0   0.500   0.000  Madd_Add1_Madd_cy<14> (Madd_Add1_Madd_cy<14>)
    XORCY:CI->O       2   0.904   1.216  Madd_Add1_Madd_xor<15> (Add1<15>)
    LUT4:I0->O        1   0.551   0.000  Madd_Yn_lut<15>1 (N21161)
    MUXF5:I1->O        0   0.360   0.000  Madd_Yn_lut<15>_f5 (N2631)
    XORCY:LI->O       1   0.622   0.801  Madd_Yn_xor<15> (Yn_15_OBUF)
    OBUF:I->O              5.644          Yn_15_OBUF (Yn<15>)
    ----------------------------------------
    Total                 42.666ns (20.893ns logic, 21.773ns route)
                                   (49.0% logic, 51.0% route)
```

Fig. 7. Synthesis Delay Report of Booth Multiplier



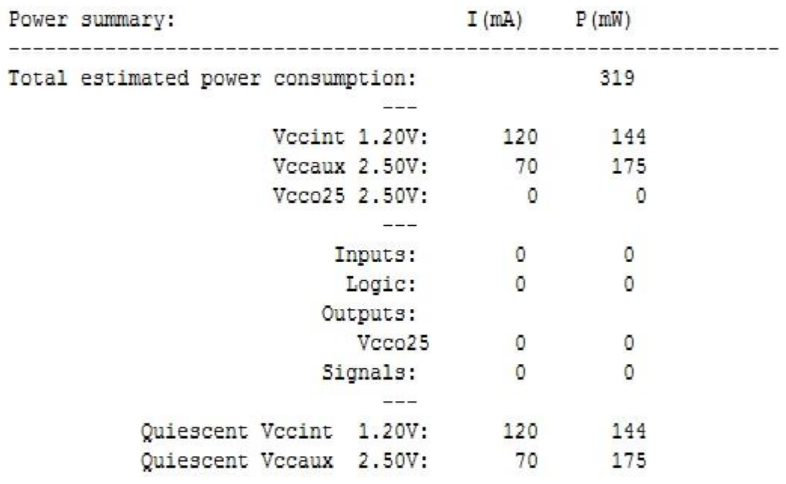
```
LUT4:I0->O          2   0.551   1.072  Adder_Part/MM1a/CI  (Adder_Part/X<2>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM2/CI  (Adder_Part/X<3>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM3/CI  (Adder_Part/X<4>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM4/CI  (Adder_Part/X<5>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM5/CI  (Adder_Part/X<6>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM6/CI  (Adder_Part/X<7>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM7/CI  (Adder_Part/X<8>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM8/CI  (Adder_Part/X<9>)
LUT3:I1->O          2   0.551   1.072  Adder_Part/MM9/CI  (Adder_Part/X<10>)
LUT3:I1->O          2   0.551   0.903  Adder_Part/MM10/CI  (Adder_Part/X<11>)
LUT4:I3->O          2   0.551   1.072  Adder_Part/MM11/CI  (Adder_Part/X<12>)
LUT4:I1->O          1   0.551   0.801  Adder_Part/MM13/Mxor_S_xo<1>1  (Product_14_OBUF)
OBUF:I->O               5.644          Product_14_OBUF  (Product<14>)
----------------------------------------
Total                  30.528ns  (14.179ns logic, 16.349ns route)
                                 (46.4% logic, 53.6% route)
```

Fig. 8. Synthesis Delay Report of Dadda Multiplier



```
Power summary:                        I(mA)     P(mW)
----------------------------------------------------------------
Total estimated power consumption:               639
                                 ---
                     Vccint 1.20V:     140       168
                     Vccaux 2.50V:      70       175
                     Vcco25 2.50V:     119       297
                                 ---
                          Inputs:        0         0
                           Logic:       11        14
                         Outputs:
                            Vcco25     119       297
                         Signals:        8        10
                                 ---
          Quiescent Vccint  1.20V:     120       144
          Quiescent Vccaux  2.50V:      70       175
```

Fig, 9. Power consumption Report of Booth Multiplier



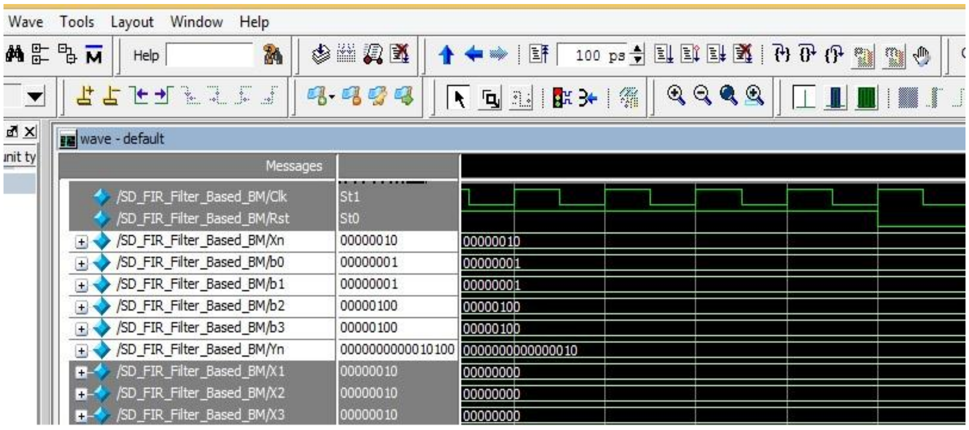
```
Power summary:                        I(mA)     P(mW)
----------------------------------------------------------------
Total estimated power consumption:               319
                                 ---
                     Vccint 1.20V:     120       144
                     Vccaux 2.50V:      70       175
                     Vcco25 2.50V:       0         0
                                 ---
                          Inputs:        0         0
                           Logic:        0         0
                         Outputs:
                            Vcco25       0         0
                         Signals:        0         0
                                 ---
          Quiescent Vccint  1.20V:     120       144
          Quiescent Vccaux  2.50V:      70       175
```

Fig. 10. Power consumption Report of Dadda Multiplier

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

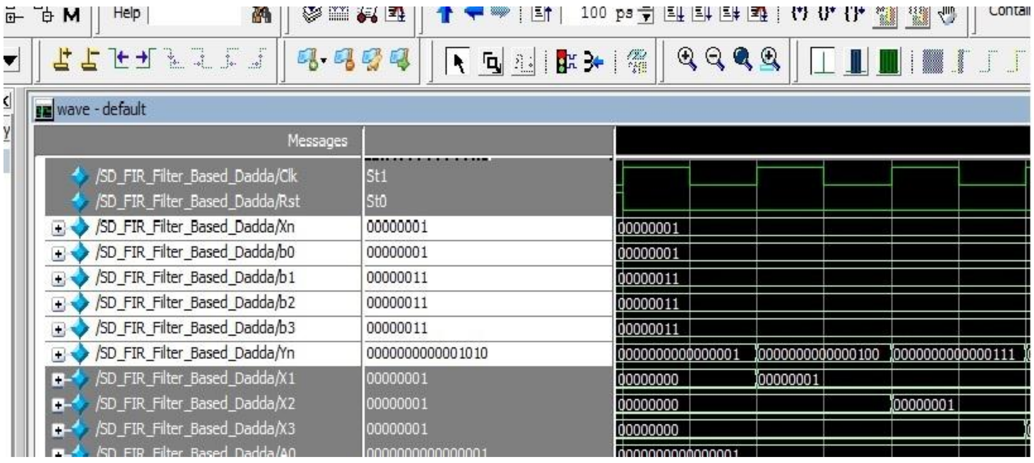
International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

NCIET- 2020
National Conference on Innovations in Engineering & Technology
Govt., Polytechnic College, Palakkad, Kerala
Vol. 8, Special Issue 1, February 2020



```
wave - default
Messages
/SD_FIR_Filter_Based_BM/Clk   St1
/SD_FIR_Filter_Based_BM/Rst   St0
/SD_FIR_Filter_Based_BM/Xn    00000010           00000010
/SD_FIR_Filter_Based_BM/b0    00000001           00000001
/SD_FIR_Filter_Based_BM/b1    00000001           00000001
/SD_FIR_Filter_Based_BM/b2    00000100           00000100
/SD_FIR_Filter_Based_BM/b3    00000100           00000100
/SD_FIR_Filter_Based_BM/Yn    0000000000010100   0000000000000010
/SD_FIR_Filter_Based_BM/X1    00000010           00000000
/SD_FIR_Filter_Based_BM/X2    00000010           00000000
/SD_FIR_Filter_Based_BM/X3    00000010           00000000
```

Fig.11. Simulation result of Booth Multiplier



```
wave - default
Messages
/SD_FIR_Filter_Based_Dadda/Clk   St1
/SD_FIR_Filter_Based_Dadda/Rst   St0
/SD_FIR_Filter_Based_Dadda/Xn    00000001           00000001
/SD_FIR_Filter_Based_Dadda/b0    00000001           00000001
/SD_FIR_Filter_Based_Dadda/b1    00000011           00000011
/SD_FIR_Filter_Based_Dadda/b2    00000011           00000011
/SD_FIR_Filter_Based_Dadda/b3    00000011           00000011
/SD_FIR_Filter_Based_Dadda/Yn    0000000000001010   0000000000000001  0000000000000100  0000000000000111
/SD_FIR_Filter_Based_Dadda/X1    00000001           00000000          00000001
/SD_FIR_Filter_Based_Dadda/X2    00000001           00000000                            00000001
/SD_FIR_Filter_Based_Dadda/X3    00000001           00000000
/SD_FIR_Filter_Based_Dadda/A0    0000000000000001   0000000000000001
```

Fig. 12. Simulation result of Dadda Multiplier

Table. 13. The overall comparison of proposed and existing systems

Sl.No	Parameters	Booth Multiplier (Existing System)	Dadda Multiplier (Proposed System)
1	Delay	42.66 ns	30.52 ns
2	Power consumption	639 mW	319 mW
3	Area (Gate count)	39442	822

III. COMPARISON OF PARAMETERS

In this section, we compare the overall performance of the proposed Dadda Multiplier codes with the existing Booth Multiplier codes. The comparison of parameters like area, delay, power consumption are sketched in table below.

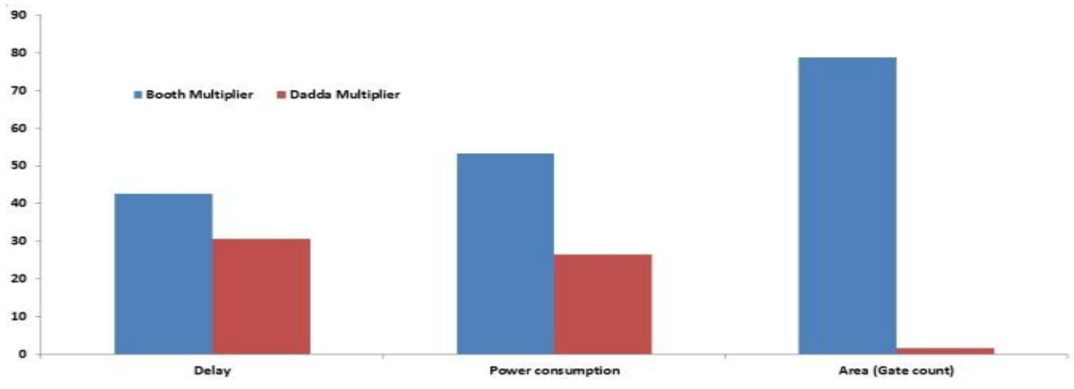


Fig. 13. Comparative Analysis Graph

V. CONCLUSION

The proposed architecture Area & Delay Efficiency when compared to a Booth Multiplier the proposed architecture makes use of bit width control logic and a low Area & Delay. This architecture is used in FIR Filter Design. The simulation result for 8 bit multipliers & four tap Filters shows that the proposed low Area & Delay architecture lowers the total Area & Delay when compared to the Dadda Multiplier architecture based Filter. The design can be verified using Modelsim 6.4c with Verilog HDL code and Area & Delay is analyzed using Xilinx software. From Table and Comparison Graph, proposed architecture can attain less area and less Delay when compared to the Dadda Multiplier s.

REFERENCES

[1] A.Chandrakasan and R. Brodersen, "Low Power CMOS Digital Design", IEEE J. Solid State Circuits, Vol.27, no.4, pp 473-484, Apr 1992
[2] N.Y.Shen and O.T.C.Chen."Low power multipliers by minimizing switching activities of partial products" in Proc. IEEE Int.Symp.Circuits Syst., May 2002, Vol.4, pp 93-96.
[3] O.T.Chen, S.Wang and Y, W.Wu "Minimization of switching activities of partial products for designing low power multipliers" IEEE Trans. Very Large Scale Integer .(VLSI)Syst. , Vol .11, No-3, pp418-433, June 2003
[4] B.Parhami Computer arithmetic algorithms and Hardware designs 1 st ed.Oxford U.K. Oxford Univ, Press 2000.
[5] K.H.Chen and Y.S.Chu , "A low power multiplier with spurious power suppression technique" , IEEE Trans. Very Large Scale Integr .(VLSI)Syst. , Vol.15 , no-7,pp846-850, July 2007.
[6] K.H.Chen K.C.Chao,J.I.Guo,J.S.Wang and Y,S,Chu. "An efficient spurious power suppression technique (SPST) and its applications on MPEG-4 AVC/H.264 transform coding design" in Proc.IEEE Int.Symp.Low Power Electron.Des . 2005 pp 155-160
[7] M.Mottaghi Dastjerdi ,A.afzali Kusha,m.Pedram "BZFAD A Low Power Low Area Multiplier Based on Booth Architecture " IEEE Trans. Very Large Scale Integr .(VLSI)Syst., Vol.17, no-2,pp302-306, Feb. 2009
[8] Ercegovac M.D. & Huang Z. (Mar 2006) "High performance low power left to right Booth Multiplier design" IEEE Trans. Comput., Vol-54, no-2, pp 272-283.
[9] Chen K.H., Chen Y.M. and Chu Y.S. (May 2007) "A Versatile Multimedia functional Unit Design Using the Spurious Power suppression Technique" in Proc.IEEE Asian Solid State Circuits Conf., pp111-114.
[10] V.P.Nelson, H.T.Nagle, B.D.Carroll and J.I.David Digital logic circuits analysis and design PH Hall ,1996