Estimation of Speed and Area of High Speed Multiplier Designed using Booth-Wallace Unit Add Method

Vikram Singh
PG Student (M.Tech. VLSI, Deptt. of E & CE)
Laxmi Devi Institute of Engineering & Tech.
Alwar, Rajasthan
Vicky25engg11@yahoo.com

Manish Kumar Jain
Assistant Professor (Deptt. of E & CE)
Laxmi Devi Institute of Engineering & Tech.
Alwar, Rajasthan
Manishjain1977@gmail.com

***Abstract*— Multiplier is the most important element of the digital signal processing such as filtering and convolution and hence their speed and area are of prime concern. A FIR is accomplished by repetitive application of multiplication and addition, their speed becomes a major factor which determines the performance of the entire calculation. In our paper we presented a high performance multiplier and then implementing them on FIR. By comparing a few multipliers we get the best solution to optimize the speed and area.**

In this paper we presented the efficient implementation of high speed multiplier using Array, Booth, Booth - Wallace (16 bit) method. Finally the performance improvement of the proposed multipliers is validated by implementing a higher order FIR filter.

General Terms: *Finite Impulse Response (FIR), Adders, Multipliers, Filters*

Keywords*: Array, Booth Encoder, Booth-Wallace.*

I. INTRODUCTION

As we know that integration technology is growing day by day, more and more sophisticated signal processing systems are being implemented using VLSI chip. The multiplier is an essential element of the digital signal processing such as filtering and convolution and hence their power dissipation and speed are of prime concern. However speed and area are two conflicting constraints. So improving speed results always in large areas. Many research efforts have been devoted to reducing the power dissipation of different multipliers. The largest contribution to the total power consumption in a multiplier is due to generation of partial product. So in our thesis we have tried to present a new multiplication technique in which we will generate the partial products using Booth encoder and then the addition of these partial products are obtained using unit add method based on Wallace tree adder. A FIR is accomplished by repetitive application of multiplication and addition, their speed becomes a major factor which determines the performance of the entire calculation. In our paper we develop a high performance multiplier and then implementing them on FIR. By comparing a few multipliers we get the best solution to optimize the speed

II. PREVIOUS WORK

Array multiplier

In systolic multiplication, to carry out the multiplication and get the final product following steps should be followed

1. The multiplicand and multiplier are arranged in the form of array as shown in the Fig. 2.
2. Each bit of multiplicand is multiplied with each bit of multiplier to get the partial products.
3. The partial products of the same column are added along with carry generated.
4. So the resulted output by adding partial products and the carry is the final product of the two binary numbers.

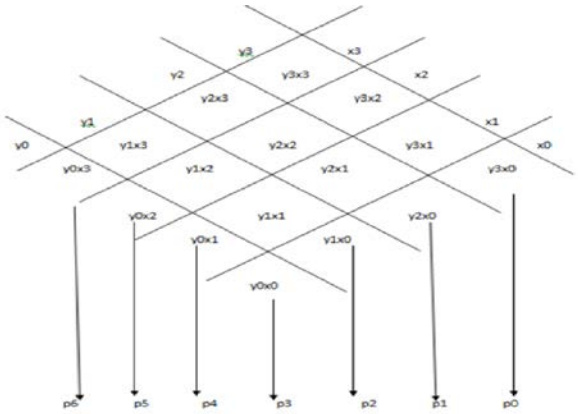

Fig. 1 Multiplicand and multiplier are arranged in the form of array [5]

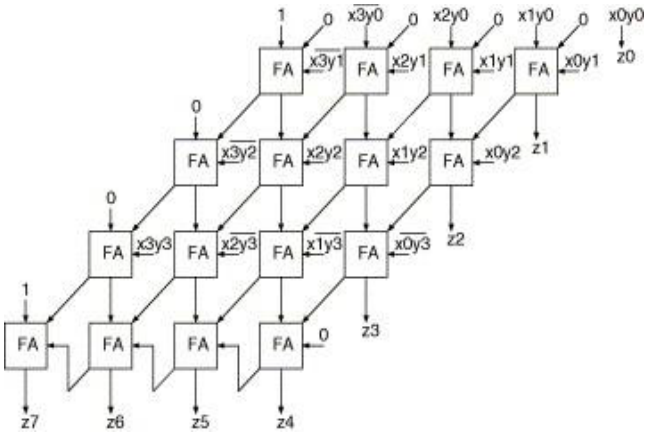

Fig. 2 Functional units of the 4 bit Systolic Multiplier [5]

Each unit is an independent processing unit. These units share information with their neighbours, after performing the needed operations on the data. Each box in the Fig. 2 represents a full adder. Inputs are vector X and Yare ANDed and the AND outputs are fed as input to full adder, which gives rise to two outputs. One is the actual multiplied output vector Z and another one is the carry generated from the addition of the AND output, which is further used by other full adders. In this way computation takes place simultaneously in the rows and columns. All the inputs are applied simultaneously, therefore registers are not required. Availability of inputs at the start of the computation and the systolic array structure helps the multiplier to perform faster compared to other multipliers.

Booth multiplication algorithm

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complement representation.

It will illustrate the booth algorithm with the following example:

Example, $2_{ten} \times (-4)_{ten}$

$0010_{two} * 1100_{two}$

STEP 1: MAKING THE BOOTH TABLE.

I. From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.

i.e., 0010 -- From 0 to 0 no change, 0 to 1 one change, 1 to 0 another change, so there are two changes on this one 1100 -- From 1 to 1 no change, 1 to 0 one change, 0 to 0 no change, so there is only one change on this one.

Therefore, multiplication of 2 x (– 4), where 2_{ten} (0010_{two}) is the multiplicand and $(-4)_{ten}$ (1100_{two}) is the multiplier.

II. Let X = 1100 (multiplier)
Let Y = 0010 (multiplicand)
Take the 2's complement of Y and call it –Y
–Y = 1110

III. Load the X value in the table.

IV. Load 0 for X-1 value it should be the previous first least significant bit of X

V. Load 0 in U and V rows which will have the product of X and Y at the end of operation.

VI. Make four rows for each cycle; this is because we are multiplying four bits numbers.

U	V	X	X-1	
0000	0000	1100	0	Load the value
				1st cycle
				2nd cycle
				3rd Cycle
				4th Cycle

TABLE 1
MAKING OF BOOTH TABLE

Step 2: Booth algorithm

Booth algorithm requires examination of the multiplier bits, and shifting of the partial product. Prior to the shifting, the multiplicand may be added to partial product, subtracted from the partial product, or left unchanged according to the following rules:

Look at the first least significant bits of the multiplier "X", and the previous least significant bits of the multiplier "X - 1".

I. 0 0 Shift only
1 1 Shift only.
0 1 Add Y to U, and shift
1 0 Subtract Y from U, and shift or add (-Y) to U and shift

II. Take U & V together and shift arithmetic right shift which preserves the sign bit of 2's complement number. Thus a positive number remains positive, and a negative number remains negative.

III. Shift X circular right shift because this will prevent us from using two registers for the X value.

U	V	X	X-1
0000	0000	1100	0
0000	0000	0110	0
0000	**0000**	**0110**	**0**

Shift only

TABLE 2
SHIFT IN BOOTH TABLE

Repeat the same step until the four cycles are completed.

U	V	X	X-1
0000	0000	1100	0
0000	0000	0110	0
0000	**0000**	**0011**	**0**

Shift only

TABLE 3
PARTIAL PRODUCTS

U	V	X	X-1
0000	0000	1100	0
0000	0000	0110	0
0000	0000	0011	0
1110	**0000**	**0011**	**0**
1111	**0000**	**1001**	**1**

Add –Y (0000 + 1110 = 1110)
Shift

TABLE 4
ADDITION OF PARTIAL PRODUCTS

U	V	X	X-1
0000	0000	*1100*	0
0000	0000	0110	0
0000	0000	0011	0
1110	0000	0011	0
1111	0000	1001	1
1111	**1000**	***1100***	**1**

Shift only

TABLE 5
AFTER FINAL SHIFT

After finishing four cycles, so the answer is shown, in the last rows of U and V which is: 11111000_{two}

Note: By the fourth cycle, the two algorithms have the same values in the Product register.

IV. PROPOSED WORK

Wallace Tree

In a Wallace tree multiplier the addition of bits within one column is rearranged still further. Let's first re-visit the columns involved in the computation of a product. Each column is characterised by the inputs to that column, and the outputs from that column.

The inputs to a column are the bits of the partial product (Booth or non-Booth encoded) plus the carry bits from one column to the right plus the sum bits that are generated within the column. The outputs from a column are the carry bits to the column one to the left plus the last two sum bits in that column that are passed to the CLA. All bits from partial products are available at the same time. So in the Wallace tree multiplier we use a tree of adder cells. The carry in comes CSA fashion from the previous column. The carryout goes from CSA fashion to the next column. In comparison to the basic array multiplier, the delay from partial products to final sum bits in a column is O(ln(n)) rather than O(n).

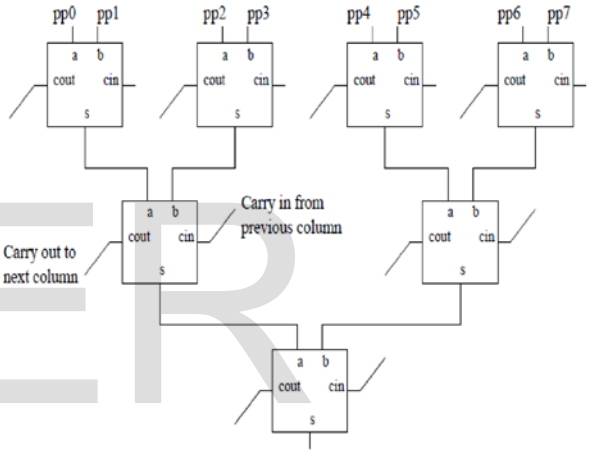

Fig. 3 Wallace tree using full adder [7]

The speed is limited by the number of full adders that the first partial product has to go through. Wallace noticed that A+B+C+D=(A+B)+(C+D). The carry save adder approach adds the partial products sequentially: the first two partial products are added before the third partial product is added. A+B+C+D require three full adder delays. Partial product A and B, C and D can be added at the same time. Their results are then added together. This requires only two full adder delays. Figure shows the difference in two approaches.

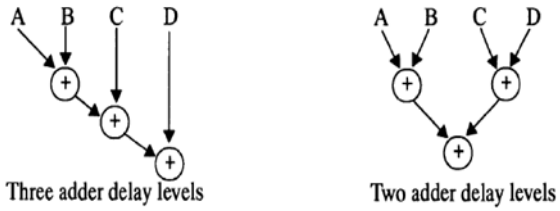

Fig. 4 Parallel adder tree [7]

So on the basis of this we use a unit adder instead of using full adder. Each unit adder adds four data inputs and one carry input. It generates one sum bit and two carry outputs.

Fig. 5 shows the unit adders organization for adding eight partial products. Note that the first two rows of unit adders add partial products. The first row unit adders add partial products P4, P3, P2 and P1. The second row of unit adders add partial product P8, P7, P6 and P5. The third row of unit adders adds the sum outputs from the first two rows and the carry bits from the unit adders in the right column.

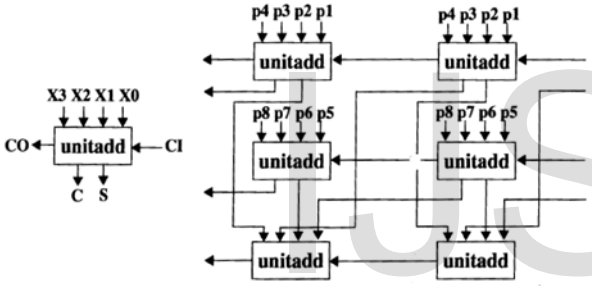

Fig. 5 Unit adder organization for eight partial products [7]

Booth- Wallace tree multiplier

Now we can put the suggestion by booth and Wallace together. The partial products are generated with booth recoder. The partial products are added with the Wallace tree adder, similar to the case of carry save adder approach. The last row of carry and sum outputs is added together with the carry skewed to the left by one position. Fig. 6 shows a 16×16 multiplier, using booth recoder and Wallace tree adders. The multiplicand comes from the left to go into eight booth recoders. Each recoder takes 3 bits from the multiplier with '0' appended at the right end. The recoder has a 17-bit output. Each recoder output is shifted to its correct position, sign extended, and zero filled in the right end. There are three rows of unit adders. Each row has 32 unit adders. The carry and sum outputs of the last row are added with the carry output bits shifted left one bit position to add with the sum bits. The output of the adders forms the product.

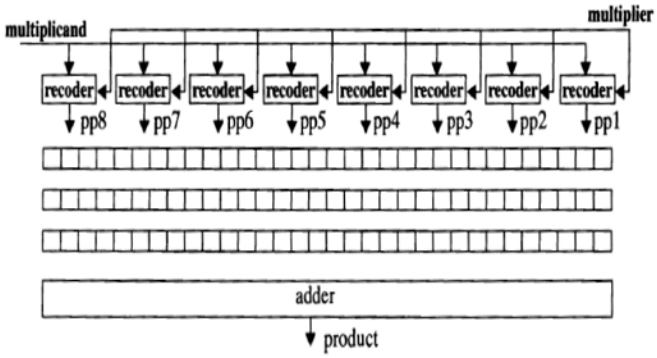

Fig. 6 Booth- Wallace tree 16×16 multiplier [7]

V. RESULTS

We have done the coding in VHDL of multipliers discussed above. The VHDL codes of different multipliers are synthesized on Xilinx ISE 12.1 and simulated using Xilinx ISE simulator. The device utilization summary obtained from synthesis report is used to compare the different multipliers.

As we know that LUTs is proportional to the area occupied by the multipliers on VLSI chip and path delay is inversely proportional to the speed of the multiplier.

Then these multipliers are implemented separately with FIR filters using computation technique like FFT, DFT. These coding are also written in VHDL language and simulate it to get the RTL circuit of each system. Also get the lookup table, where we get the exact no of i/p, o/p/ no of slices requirement etc for the system. Xilinx Power Estimator is used to analyze & determine the power consumption of the system. These results are given below.

Synthesis report of different multipliers

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	70	960	7%
Number of 4 input LUTs	121	1920	6%
Number of bonded IOBs	32	66	48%

TABLE 6

SYNTHESIS REPORT OF ARRAY MULTIPLIER
Maximum combinational path delay: 26.048ns

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	50	768	6%
Number of 4 input LUTs	94	1536	6%
Number of bonded IOBs	33	63	52%

TABLE 7

SYNTHESIS REPORT OF BOOTH MULTIPLIER
Maximum combinational path delay: 16.552ns

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of bonded IOBs	64	63	101%
Number of MULT18X18s	1	4	25%

TABLE 8

SYNTHESIS REPORT OF BOOTH-WALLACE MULTIPLIER
Maximum combinational path delay: 13.024ns

Synthesis report of FIR designed using different multipliers

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	62	768	8%
Number of Slice Flip Flops	38	1536	2%
Number of 4 input LUTs	115	1536	7%
Number of bonded IOBs	26	63	41%
Number of GCLKs	1	8	12%

TABLE 9

SYNTHESIS REPORT OF FIR DESIGNED USING ARRAY MULTIPLIER

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	54	768	7%
Number of Slice Flip Flops	41	1536	2%
Number of 4 input LUTs	99	1536	6%
Number of bonded IOBs	26	63	41%
Number of GCLKs	1	8	12%

TABLE 10

SYNTHESIS REPORT OF FIR DESIGNED USING BOOTH MULTIPLIER

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	73	768	9%
Number of Slice Flip Flops	64	1536	4%
Number of 4 input LUTs	90	1536	5%
Number of bonded IOBs	50	63	79%
Number of GCLKs	1	8	12%

TABLE 11

SYNTHESIS REPORT OF FIR DESIGNED USING BOOTH- WALLACE MULTIPLIER

Comparison of FIR designed using different multipliers

Logic Utilization	Array	Booth	Booth-Wallace
Number of Slices	62	54	73
Number of Slice Flip Flops	38	41	64
Number of 4 input LUTs	115	99	90

Number of bonded IOBs	26	26	50
Number of GCLKs	1	1	1
Maximum Combinational Path Delay	26.048ns	16.552ns	13.024ns

TABLE 12
COMPARISON OF FIR DESIGNED USING DIFFERENT MULTIPLIERS

VI. CONCLUSION

In this paper, a unit add method based on Wallace tree was proposed for multiplication in which partial products are generated using booth encoder. The device utilization summary obtained from synthesis report, shows that Booth multiplier has less number of LUTs as compare to array multiplier which shows Booth has less area in comparison to the array multiplier.

By simulating the proposed Booth- Wallace multiplier we found that Maximum combinational path delay in this multiplier is 13.024ns, which is very less as compare to other multipliers. This technique performs the multiplication of 16 bit. When we compare the path delay and LUTs of Booth encoder and Wallace unit adder with other multipliers, we found that this method is better than other multipliers in terms of speed and area. So by using Booth-Wallace multiplier we can achieve the fast and efficient multiplication.

REFERENCES

[1] Anand Kumar., "Fundamentals of Digital Circuits", *Prentice Hall of India*, 2008.

[2] Volnei A. Pedroni., "Circuit Design using using VHDL", *MIT Press*, 2004.

[3] Morris Mano,"Digital Design, Third edition", *Prentice Hall of India*, 2000.

[4] Marcelo Fonseca, Eduardo da Costa et al., "Design of a Radix-2 Hybrid Array Multiplier Using Carry Save Adder", *in SBCCI'05 proc. of the annual symposium on integrated circuits and system design*, pp. 172-177, 2005.

[5] Bairu K. Saptalakar et al., "Design and Implementation of VLSI Systolic Array Multiplier for DSP Applications", *in International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)*, Vol. 2 Issue 3, pp. 156-159,2013.

[6] S. Karunakaran et al., "Area and Power Efficient VLSI Architecture for FIR Filter using Asynchronous Multiplier", *in British Journal of Science*, Vol. 2, pp. 61-77, 2011.

[7] C. S. Wallace., "A Suggestion for a Fast Multiplier", *in IEEE Trans. on Electronic Computers*, vol. 13, pp. 14-17, 1964.

[8] D. Baran et al., "Energy Efficient Implementation of Parallel CMOS Multipliers with Improved Compressors", *in publication of IEEE conference on Low-Power Electronics and Design (ISLPED)*, pp. 147-152, 2010.

[9] Soojin Kim et al., "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges", *in World Academy of Science, Engineering and Technology*, 2010.

[10] Dong-Wook Kim, Young-Ho Seo., "A New VLSI Architecture of Parallel Multiplier-Accumulator based on Radix-2 Modified Booth Algorithm", *in IEEE Trans. On Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 201-208, 2010.

[11] Raminder Preet Pal Singh et al., "Performance Analysis of 32-Bit Array Multiplier with a Carry Save Adder and with a Carry-Look-Ahead Adder", *in International Journal of Recent Trends in Engineering*, Vol 2, No. 6, 2009.