A DESIGN OF LOW POWER MODIFIED BOOTH MULTIPLIER

M. Kiran Kumar[1], S. Anusha[2], G.Y. Rekha[3]
[1,2,3]Dept. of ECE, Anurag Group of Institutions, Hyderabad, Telangana

ABSTRACT
Multipliers are key components of many high performance systems such as FIR filters, Microprocessor, digital signal processors, etc. Modified Booth Multiplier is one of the different techniques for signed multiplication. It is used normally as the fastest multiplier. Here we designed a low power 8 bit Modified Booth multiplier has done using conventional method as well as using GDI(Gate Diffusion Input) technique. The comparative analysis of all the design for the delay, no of transistors and power has done using Cadence standard gpdk180nm Technology.
Keywords: Booth algorithm, Multiplier, GDI, Low Power, Delay.

I. INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, Microprocessor, digital signal processors, etc. Signed multiplication is a careful process. With unsigned multiplication there is no need to take the sign of number into consideration. Today we know that multiplier is using in every basic circuit. Today all the ALU system is based on a multipliers. Complete arithmetic Logical part is based on a multiplier and if multiplier is consuming so much delay then the entire product which is based on a multiplier is fail due to fail of multiplier. If multiplier have low speed then it will works slowly. Regarding this if our function is working in 2 second then it will also take some delay .Then its output will be 2second+ delay. That delay may greater then to basic performing delay. In VLSI speed of any IC is depend on power consumption, Area, delay. Some we have a complex circuit and at that time we will get increment in delay and power consumption. Power consumption is also a main power factor. If we reduce the power factor of an IC then it is showing that our product battery life is good. Today everybody is using calculators and CPU. Every company is working for low power consumption circuit so that they can deliver more long life battery as comparison to other company. If our Multiplier power consumption will be increase then heat dissipation will be increase. So it will increase leakage current. So this multiplier will be used in many ALU circuits. then all the product of this ALU have a low battery life. If we are using multiplier operations for memory allocation of mobile phone then battery of that mobile phone will not be long life because the heat dissipation, power consumption is greater than the normal range. According to Moore low in every 18 month the transistors of any IC will be doubled which is using in a IC .According to moore low after every 18 month we will get a new IC in which we will find a ore number of transistor according to previous.

								A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
								B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
						$\overline{PP_{80}}$	PP_{80}	PP_{70}	PP_{60}	PP_{50}	PP_{40}	PP_{30}	PP_{20}	PP_{10}	PP_{00}
				1	$\overline{PP_{81}}$	PP_{71}	PP_{61}	PP_{51}	PP_{41}	PP_{31}	PP_{21}	PP_{11}	PP_{01}		Neg_0
		1	$\overline{PP_{82}}$	PP_{72}	PP_{62}	PP_{52}	PP_{42}	PP_{32}	PP_{22}	PP_{12}	PP_{02}		Neg_1		
1	$\overline{PP_{83}}$	PP_{73}	PP_{63}	PP_{53}	PP_{43}	PP_{33}	PP_{23}	PP_{13}	PP_{03}		Neg_2				
									Neg_3						

Fig-1: Algorithm for Modified Booth Multiplier

II. BOOTH MULTIPLIER

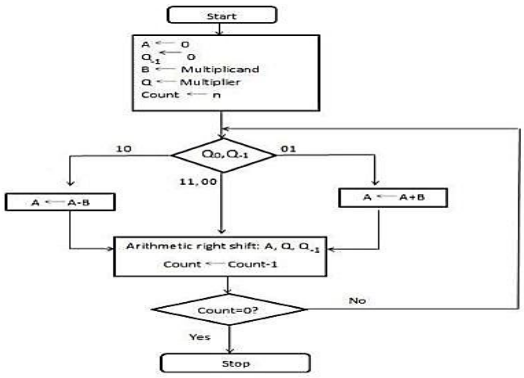

Fig-2: Flow Chat of Booth Multiplier

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward arithmetic shift on *P*. Let B and Q be the multiplicand and multiplier, respectively; and let *x* and *y* represent the number of bits in B and Q as shown in Fig-2.

1. Determine the values of *A* and *S*, and the initial value of *P*. All of these numbers should have a length equal to (*x* + *y* + 1).
   - A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (*y* + 1) bits with zeros.
   - S: Fill the most significant bits with the value of (−m) in two's complement notation. Fill the remaining (*y* + 1) bits with zeros.
   - P: Fill the most significant *x* bits with zeros. To the right of this, append the value of Q. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of *P*.
   - If they are 01, find the value of *P* + *A*. Ignore any overflow.
   - If they are 10, find the value of *P* + *S*. Ignore any overflow.
   - If they are 00, do nothing. Use *P* directly in the next step.
   - If they are 11, do nothing. Use *P* directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let *P* now equal this new value.
4. Repeat steps 2 and 3 until they have been done *y* times.
5. Drop the least significant (rightmost) bit from *P*. This is the product of **B** and Q.

MODIFIED BOOTH MULTIPLIER:

Let A be the multiplicand and B be the multiplier for multiplication of two n-bit integer numbers which can be represented in two's complement as.,

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \quad (7)$$

$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \quad (8)$$

In Modified Booth multiplier, B in becomes.

$$B = \sum_{i=0}^{\frac{n}{2}-1} m_i 2^{2i}$$

$$= \sum_{i=0}^{\frac{n}{2}-1} (-2b_{2i+1} + b_{2i} + b_{2i-1})2^{2i} \quad (9)$$

Where $b{-}1{=}0$, and $mi \in -2,-1,0,1,2$. from the encoding results of B, The booth multipliers chooses the action -2A, -A, 0, A, or 2A to generate the partial product rows shown in Table 1.

Table -1: Truth Table for the E-Cell Of The Encoder

$Y_{i+1}\,Y_i\,Y_{i-1}$	Action	$S_0\,S_1\,S_2\,S_3$
0 0 0	+0	0 1 0 1
0 0 1	+X	1 1 0 1
0 1 0	+X	1 1 0 1
0 1 1	+2X	0 1 1 1
1 0 0	-2X	0 1 0 0
1 0 1	-X	0 0 0 1
1 1 0	-X	0 0 0 1
1 1 1	+0	0 1 0 1

The system of action is partitioned into blocks such as

- the encoder unit that is the e-cell that encodes multiplier bits (Y bits) and then it send signals for the generation of partial products;
- the partial product generator (PPG) which will decodes signals from the encoder as well as the multiplicand X

in order to generate the partial products;

- the carry-save adder matrix (CAM) will add all the partial product which obtained during previous operation, and
- The last row of full adders and half adder that is the final product adder (FPA) will add all the value from the CAM and produce the final product

Fig-3 shows the architecture for Modified Booth multiplier. The encoder (e-cell in Fig 3) where the multiplier(Y) encodes and the encoded signal and the multiplicand(X) is given to the partial product generator (g-cell in Fig 4) are the basic units of the Modified Booth multiplier. Both CAM and FPA blocks are made up of full adders as well as half adders.

III. ARCHITECTURE OF MODIFIED BOOTH MULTIPLIER:

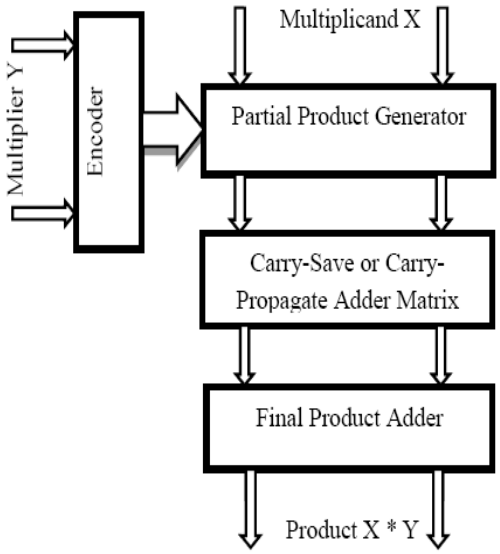

Fig-3: Architecture of Modified Booth Multiplier

BLOCK DIAGRAM OF MODIFIED BOOTH MULTIPLIER:

Fig-4 shows a block diagram of the proposed Booth multiplier implementation. This circuit takes in two 8-bit binary numbers and outputs the 16-bit product. The multiplier, X[7:0], is divided into four groupings: 0, X0, X1; X1, X2, X3; X3, X4, X5; X5, X6, X7. Each of these groupings is passed into a Booth encoder, which outputs bits corresponding to the operations described in Table 1 (x0, x1, x2, x-1). Each group of these selection bits are sent to a Booth decoder block, which outputs the appropriate partial product term based on the selected operation. These partial products are then sign extended so that sign bits are taken into account during the summing. Finally, the canonical shift and add multiplication is implemented using 12-bit carry look ahead adders (CLA). The first two bits of each partial product are passed directly to the output to account for the shifting. A standard array multiplier would typically require 8 partial products, and thus 8 adders. However, this implementation reduces the number of partial products to only four, significantly improving speed. Furthermore, the CLA provides another speed boost to the system.

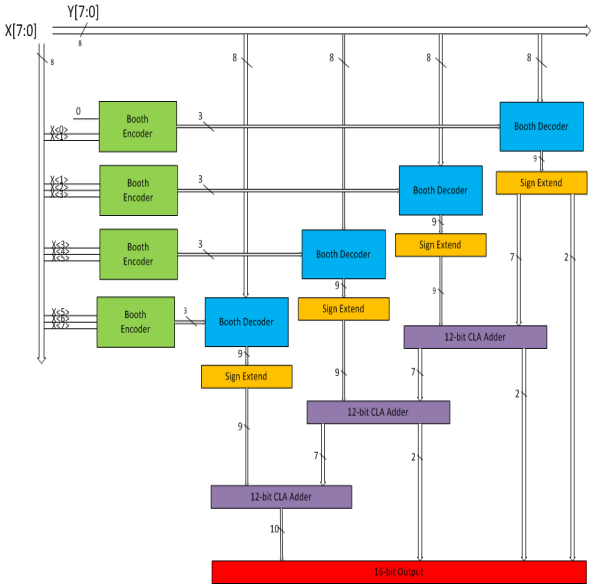

Fig-4: Block Diagram of Modified Booth Multiplier

BOOTH ENCODER:

Table 1 shows the truth table for a Booth encoder. The encoder takes inputs *xi*+1, *xi*, and *xi*−1from the multiplier bus and produces a 1 or a 0 for each operation: single, double, and negative. Fig-5 shows the booth encoder schematic Figure 6 shows the simulation results.

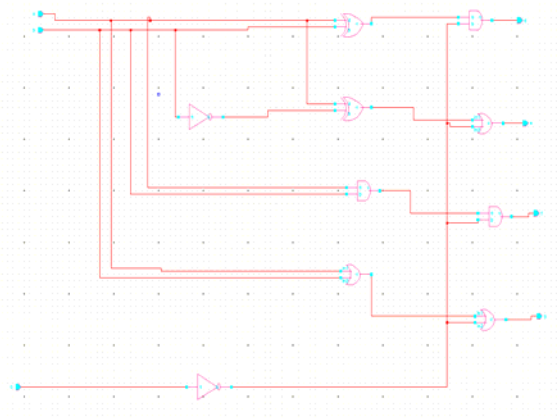

Fig-5: Booth Encoder Schematic

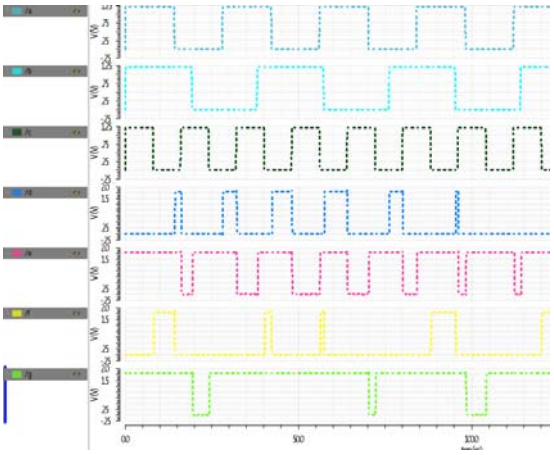

Fig-6: Booth Encoder Simulation Output

BOOTH DECODER:

Booth Decoder is designed to produce the partial products by multiplying the multiplicand, X by 0, 1, - 1, 2 or -2. The output of MBE acts as the selection inputs to the partial product generator. The partial products on each row are obtained as 1's complement numbers for negative encoding. To obtain the 2's complement number, '1' is to be added at the LSB of the obtained partial product. This operation is performed in the accumulation phase. Each partial product row is placed 2-bits to the left with respect to the previous row. Fig- 7 shows the booth decoder schematic and Fig- 8 shows the simulation result.

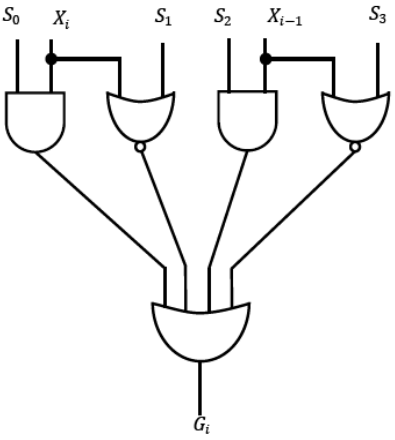

Fig-7: Booth Decoder Schematic

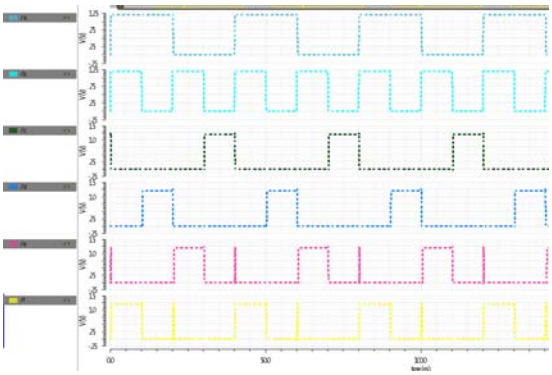

Fig-8: Booth Decoder Simulation Output

CARRY LOOK AHEAD ADDER:

A carry-look ahead adder (CLA) or fast adder is a type of adder used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, ripple carry adder for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry bit have been calculated to begin calculating its own result and carry bits (see adder for detail on ripple carry adders). The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits of the adder. Fig-9 shows the 4-bit CLA and Fig-10 shows the simulation result.

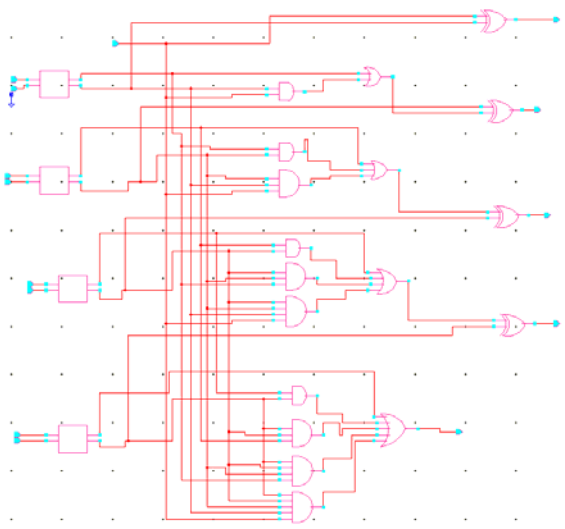

Fig-9: 4-Bit CLA Schematic

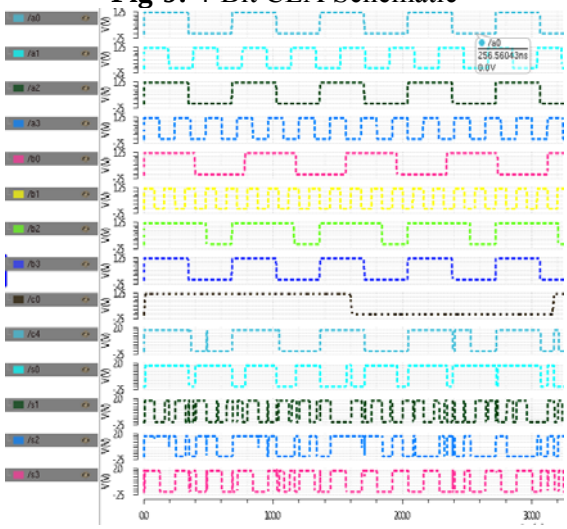

Fig-10: 4-Bit CLA Simulation Output

IV. SIMULATION RESULTS:

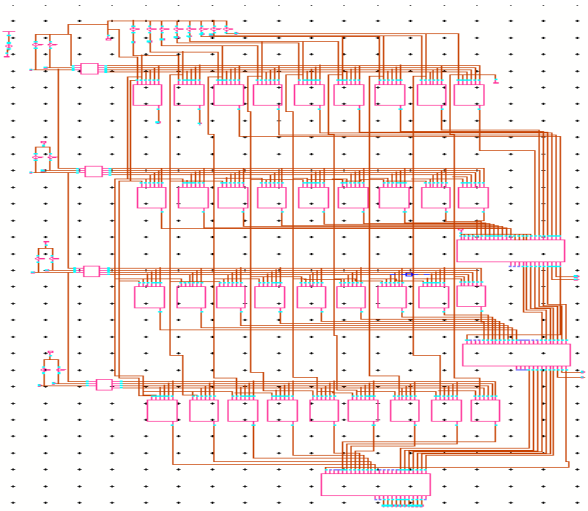

Fig-10: Schematic of 8-Bit Modified Booth Multiplier

EXAMPLE:

68*10=680

68 = 0 1 0 0 0 1 0 0 (Multiplicand)

10 = 0 0 0 0 1 0 1 0 (Multiplier)

0 0 0 -1 1 -1 1 0 Booth Recorded Multiplier

0 -1 +1 +2 Bit Pair Recorded Multiplier

1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0	(2°)*-2*68 = -136
1 1 1 1 1 1 1 0 1 1 1 1 0 0	(2²)*-1*62 = -272
0 0 0 0 0 1 0 0 0 1 0 0	(2^4)*1*68 = 1088
0 0 0 0 0 0 0 0 0 0	
0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0	680

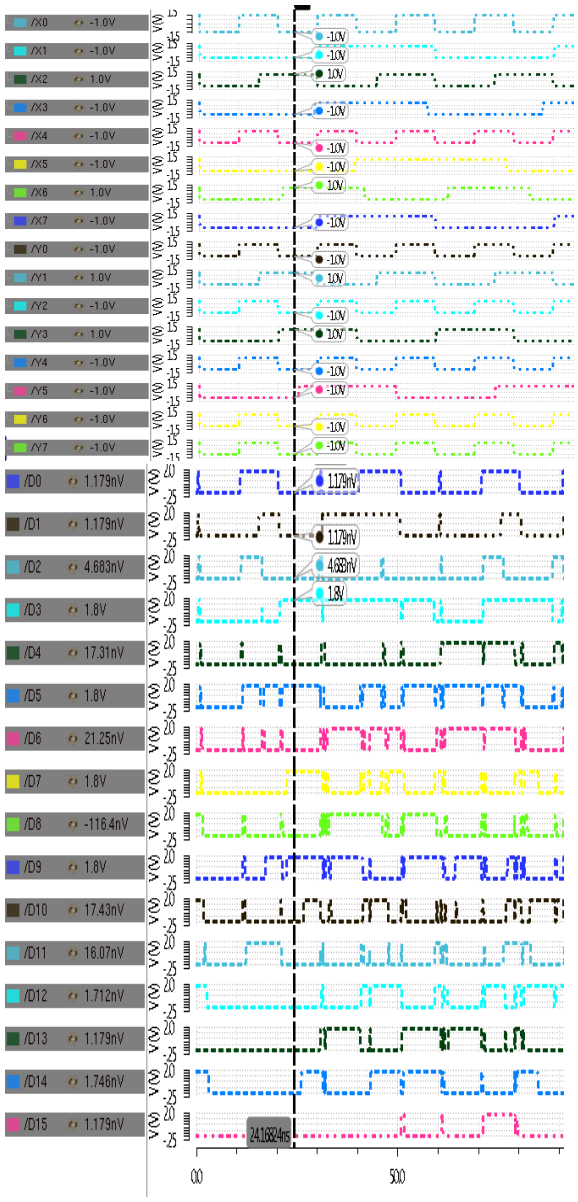

Fig-11: 8-Bit Modified Booth Multiplier Simulation Result

V. CONCLUSION:

Analysis of Modified Booth multiplier has done in Cadence RTL compiler. Radix-4

Booth Multiplier is implemented here the complete process of the implementation is giving higher speed of operation.

REFERENCES

[1] W. -C. Yeh and C. -W. Jen. 2000. "High Speed Booth encoded Parallel Multiplier Design." *IEEE transactions on computers*. 49(7): 692-701.

[2] Shiann-Rong Kuang, Jiun-Ping Wang and Cang-Yuan Guo. 2009. "Modified Booth multipliers with a Regular Partial Product Array." IEEE Transactions on circuits and systems-II. 56(5).

[3] Li-Rong Wang, Shyh-Jye Jou and Chung-Len Lee.2008. "A well-tructured Modified Booth Multiplier Design." 978-1-4244-1617-2/08/$25.00 ©2008 IEEE.

[4] Soojin Kim and Kyeongsoon Cho. 2010. "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges*."* *World academy of Science, Engineering and Technology*. P.61

[5] Magnus Sjalander and Per Larson-Edefors. 2008. "The Case for HPM-Based Baugh-Wooley Multipliers*."* *Chalmers University of Technology*, Sweden.

[6] J. Fadavi-Ardekani. 1993. "a M×N Booth Encoded Multiplier Generator Using Optimized Wallace Trees", *IEEE Trans. VLSI Systems*. 1(2).

[7] Wang G. 2004. "A unified unsigned/signed binary multiplier." *The Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*. 1: 513-516.

[8] Kim J. Y. 1999. "Multiplier to selectively perform unsigned magnitude multiplication or signed magnitude multiplication." *USpatent*. 5, 870, 322.

[9] Hwang-Cherng Chow and I-Chyn Wey. 2002. "A 3.3V 1GHz high speed pipelined Booth multiplier". *Proc. Of IEEE ISCAS*. 1: 457-460.

[10]M. Aguirre-Hernandez and M. Linarse-Aranda. "Energy-efficient high-speed CMOS pipelined multiplier". *Proc. of IEEE.*