Design and Analysis of Various Standard Multipliers Using Low Power Very Large Scale Integration (VLSI)

R. Rajeswari
UG scholar, Department of ECE,
Vel Tech Engineering College, Avadi, Chennai, Tamil Nadu, India.
Email: rvrajeswari90@gmail.com

Abstract - This paper presents a comparative study of Field Programmable Gate Array (FPGA) implementation of standard multipliers using Verilog HDL. Multiplier is a good candidate for digital signal processing (DSP) applications such as finite impulse response (FIR) and discrete cosine transforms (DCT) and so on. Significant reduction in FPGA resources, delay, and power can be achieved using Reduced Wallace multipliers with Kogge-stone adder instead of standard parallel multipliers.

Keywords - Fast Multiplier, High Speed Adder, Power-Delay Product, Field Programmable Gate Array.

1. Introduction

A binary multiplier is an electronic circuit used in digital electronics, such as computer, to multiply two binary numbers. It is built using binary adders. A variety of computer arithmetic techniques can be used to implement a digital multiplier. Most techniques involve computing a set of partial products, and then summing the partial products together. For high speed multiplications, a huge number of adders or compressors are to be used to perform the partial product addition Fast multipliers are essential parts of digital signal processing systems. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation.

The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today. In the past, multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits.

It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them. The major speed limitation in any adder is in the production of carries. Basically, carry save adder is used to compute sum of three or more n-bit binary numbers.

2. Booth Multiplier

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. We used the modified Booth encoding (MBE) scheme proposed in [15]. It is known as the most efficient Booth encoding and decoding scheme.

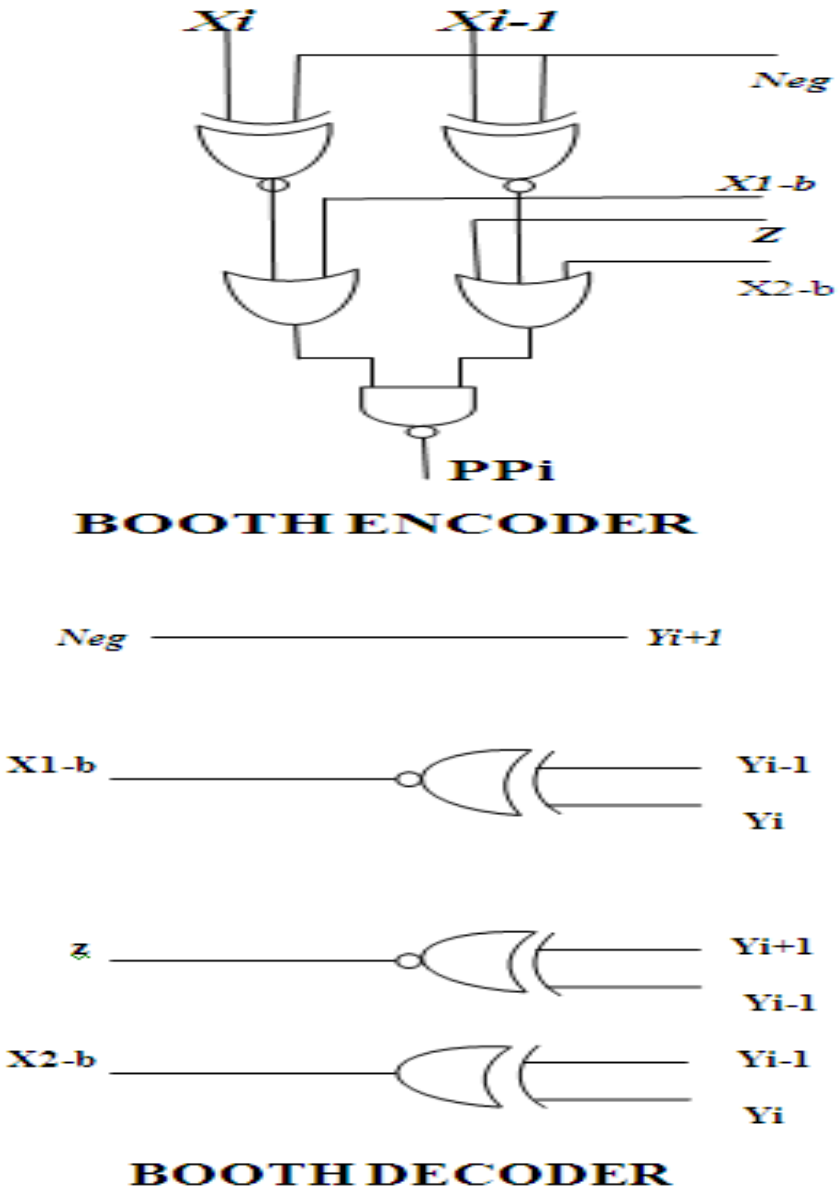

Figure.1 (a) Booth encoder (b) Booth decoder

To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. Table I shows the rules to generate the encoded signals by MBE scheme and Fig. 1 (a) shows the corresponding logic diagram. The Booth decoder generates the partial products using the encoded signals as shown in Fig. 1 (b).

The modified Booth algorithm reduces the number of partial products by half in the first step as shown in the figure 1.

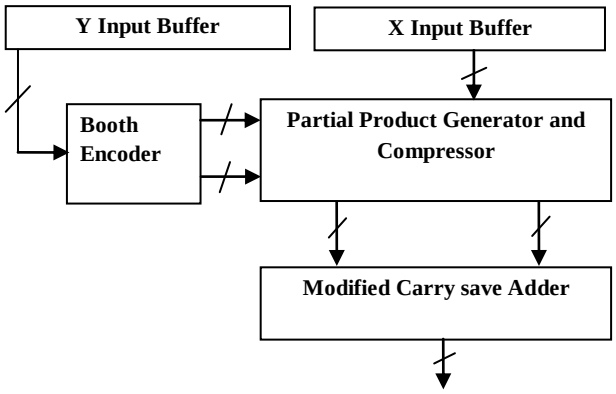

Figure.2 Generated Partial Products and Sign Extension Scheme

In the second stage, the two intermediate products are added to form the final product through the Modified Carry Save Adder. 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 recorded by the booth encoder and X is the multiplicand. PPG module and compressor form the major part of the multiplier. Modified carry save adder is the final adder used to merge the sum and carry vector from the compressor module.

For radix-4 recoding, the popular algorithm is parallel recoding or Modified Booth recoding. In parallel radix-4 recoding, Y becomes:

$$Y = \sum_{i=0}^{\frac{n}{2}-1} v_i \, 4^i = \sum_{i=0}^{\frac{n}{2}-1} (-2y_{2i+1} + y_{2i} + y_{2i-1}) \, 4^i \qquad (1)$$

The truth tables are shown in Table 1.

Table 1 Truth Table of Modified Booth Encoder Scheme

y_{2i+1}	y_{2i}	y_{2i-1}	value	neg_i	two_i	$zero_i$	cor_i
0	0	0	$+0$	0	0	1	0
0	0	1	$+X$	0	0	0	0
0	1	0	$+X$	0	0	0	0
0	1	1	$+2X$	0	1	0	0
1	0	0	$-2X$	1	1	0	1
1	0	1	$-X$	1	0	0	1
1	1	0	$-X$	1	0	0	1
1	1	1	-0	0	0	1	0

Table 1. Truth Table of Modified Booth Encoder Scheme

In our design we described Booth function as three basic operations, as 'direction', 'shift', and 'addition'. Direction determines whether the multiplicand is positive or negative, shift explains whether the multiplication operation involved shifting or not and addition means whether the multiplicand is added to partial product or not.

3. DADDA Multiplier Architecture

The Dadda Tree multiplier has the same general stages as the Wallace Tree. However, unlike the Wallace Tree, Dadda multipliers do not attempt to reduce as many partial products in each layer but rather, perform as few reductions as possible. This makes the Dadda multiplier less costly in the reduction stage but contain longer numbers in each stage, requiring larger carry-save adders (CSA).

For this design, formation of partial products was done using the same method as in the Wallace Tree multipliers. As for the reduction stage, a series of generalized recursive steps Were used to determine the heights of each stage and the number of additions required to achieve each stage height as described in the following:

1. Let $d1 = 2$ and $dj+1 = [1.5 \_ dj]$. D is the height of the matrix for the *jth* stage. Repeat until the largest j stage is reached in which the original N height matrix contains at least one column that has more than d dots.

2. In the *jth stage* from the end, place (3, 2) and (2, 2) counters as required to achieve a reduced matrix. Only columns with more than dj dots as they receive carries from less significant (3, 2) and (2, 2) counters are reduced.

3. Let $j = j - 1$ and repeat step 2 until a matrix with a height of two is generated. This should occur when j= *1*.

4. Binary Multiplier Using High Speed Compressors

For higher order multiplications, a huge number of adders or compressors are to be used to perform the partial product addition. We have reduced the number of adders by introducing Special kind of adders that is capable to add five/six/seven bits per decade. These adders are called compressors. Binary counter property has been merged with the compressor property to develop high order compressors. Uses of these compressors permit the reduction of the vertical critical paths. An 8×8bit multiplier has been developed using these compressors. A 16-bit multiplier is constructed by using Wallace tree architecture, the architecture has been shown in Figure 3.

Partial products are added in five stages. Adders and different compressors are used to minimize the stage operations. Compressors and adders are used carefully so that minimum number of outputs would be generated. Consider the column number ten where ten bits are added at the first stage. These ten bits could be added by using two 5-3 compressors, but that will generate six (three of each compressor) outputs, instead of this we have used one 7-3 compressor and one full adder that generate five outputs only (three of compressor and two of full adder) that eventually decrease the number of bits for the next stage.

It is to be mentioned that each compressors (4-3 to 7-3) has three outputs of bit position jth, (j+1). Than d (j+2) th[12]. Now if a compressor is used in column no six (say) then its jth output goes to the circuit of column no six and (j+1) th bit goes to the circuit of column no seven and the rest output goes to the circuit of column no eight of the next stage. Thus our compressors reduce vertical critical path more rapidly than conventional compressors

5. Architecture for Different Compressors

The block diagrams of 4-2 and 5-3 compressors with the entire sub units are shown in Figure 4.a and Figure 4.b. Each Figure contains three circuit blocks: two adder blocks and another circuit block that performs the parallel addition operation. Here five bits are processed through one half adder and one full adder [12].

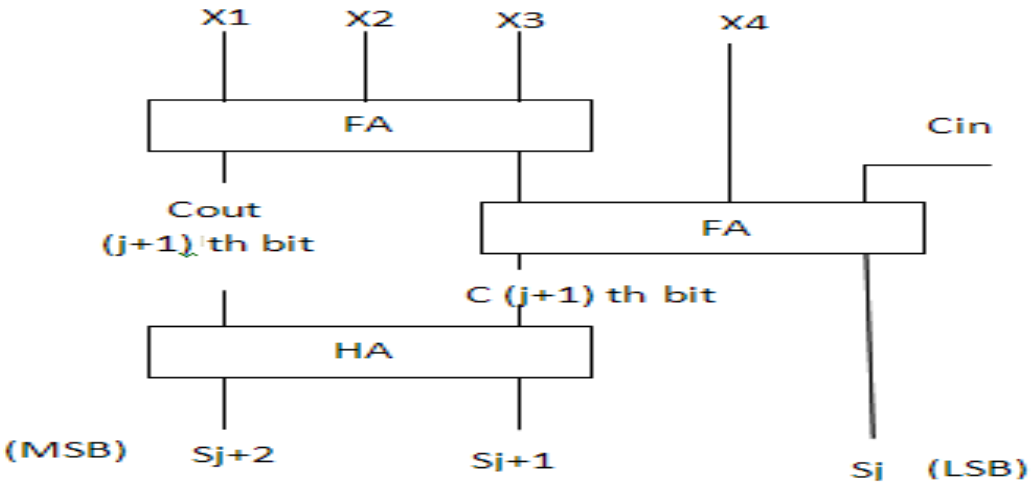

Figure.3 Modified 4-2 compressor

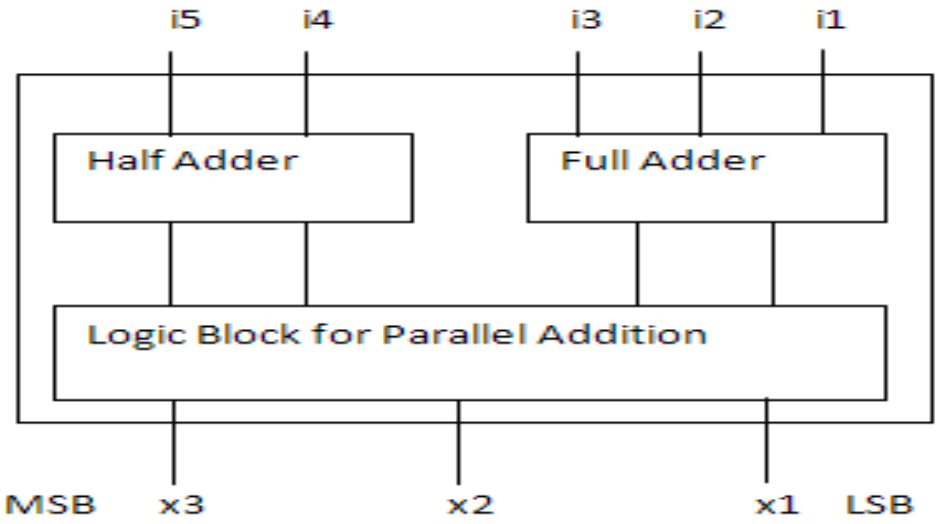

Figure.4 Block diagram of 5-3 compressor

6. Proposed Method of Reduced Complexity Wallace Multiplier

It is the modified version of Wallace multiplier. It has less half adders than the normal Wallace multiplier .The partial products are formed by N2 AND gates. The partial products are arranged in an "inverted triangle" order. The modified Wallace reduction method divides the matrix into three row groups [13].

i. Use full adders for each group of three bits in a column like the conventional Wallace reduction.
ii. A group of two bits in a column is not processed, that is, it is passed on to the next stage (in contrast to conventional method). Single bits are passed on to the next stage as in the conventional Wallace reduction.
iii. The only time half adders are used is to ensure that the number of stages does not exceed that of a conventional Wallace multiplier. For some cases, half adders are only used in the final stage of reduction.

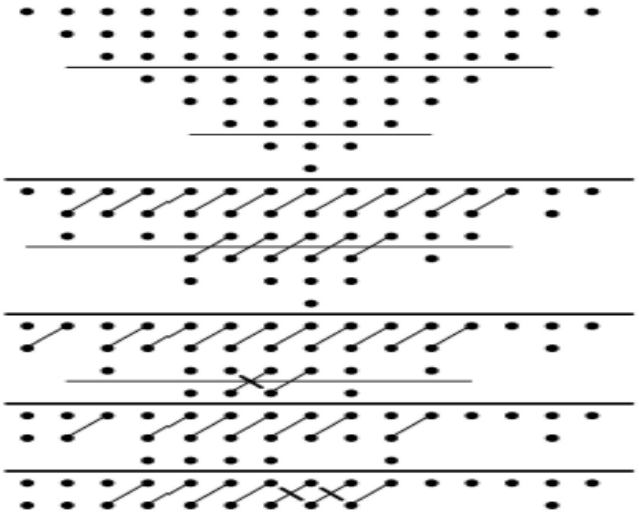

Figure 5: Reduced Complexity Wallace multiplier

7. Modified Carry save Adder

The 16-bit conventional CSA [3] has 17-half adders (H) and 15-full adders (F). Since the Ripple Carry Adder (RCA) is used in the final stage, this structure yields large carry propagation delay. To reduce the delay, the final stage of CSA is divided into 5 groups as shown in Fig 2.The first group includes n $1+\log_2$ n-bit value and other groups include $\log_2$n-bit value, where n is the bit size of the adder. The divided groups are listed as follows:

1. {c4, s [4:0]}, output s [4:0] is directly assigned as the final output.
2. {c7, x [7:5]} manipulates the partial result by considering c4 is 0.
3. {c10, x [10:8]} manipulates the partial result by considering c7 is 0.
4. {c13, x [13:11]} manipulates the partial result by considering c10 is 0
5. {X [17:14]} manipulates the partial result by considering c13 is0.

Depending on c4 of the first group, the second group Mux gives the final result without the carry propagation delay from c4 to c7; depending on c7 of the second group final result, the third group Mux gives the final result without the carry propagation delay from c7 to c10; depending on c10 of the third group final result, the fourth group Mux gives the final result without the carry propagation delay from c10 to c13 and depending on c13 of the fourth group final result, the fifth group Mux gives the final result without the carry propagation delay from c13 to s17.

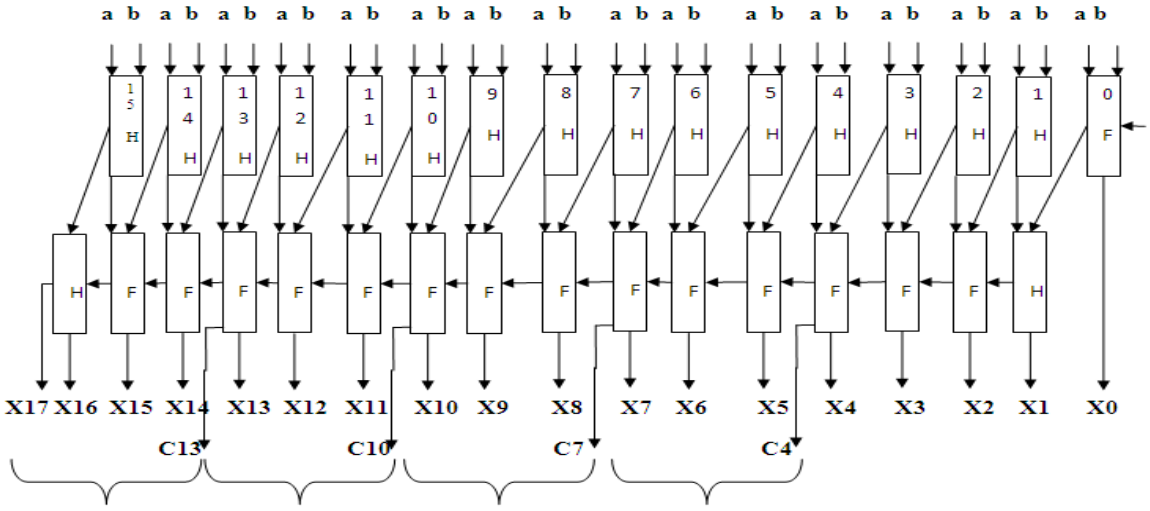

Figure6: Modified Carry save Adder

From the above diagram the term c4 to x17 propagate to following figure 7.

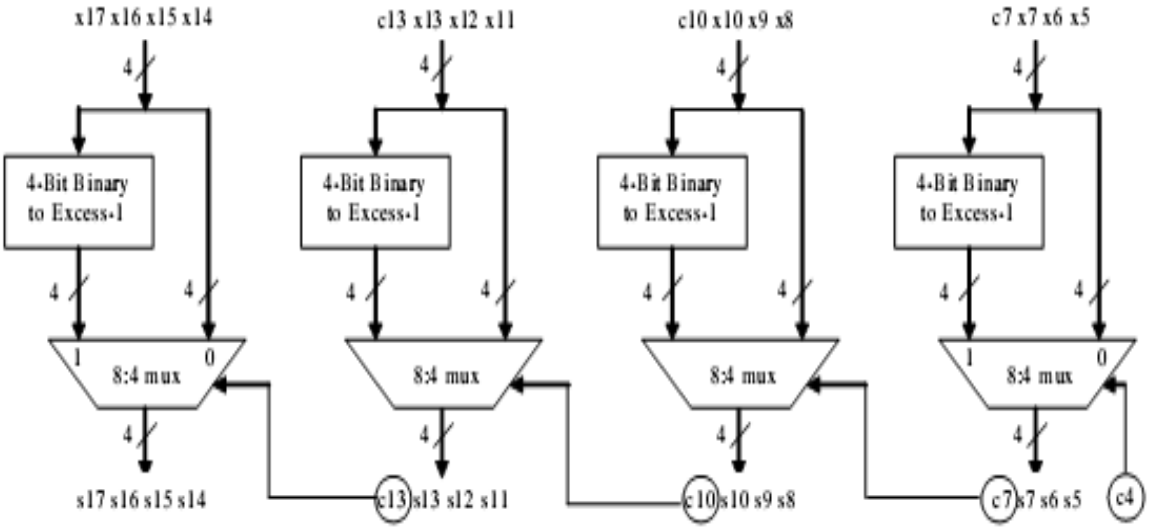

Figure 7: Propagates from above figure

The main advantage of this logic is that each group computes the partial results in parallel and the multiplexers are ready to give the final result "immediately" with the minimum delay

of the Mux. When the Cin of each group arrives, the final result will be determined "immediately". Thus the maximum delay is reduced in the carry propagation path. This same logic has been used for 32 and 64-bit adder structures to achieve higher speeds. The area indicates the total cell area of the design and the total power is sum of leakage power, internal power, net power and dynamic power. The proposed result shows that the Modified Carry Save Adder (MCSA) [3] has reduced area, delay and consumes lesser power than CSA. In the above diagram,instead of using normal multiplexer,go for proposed architecture by using adders for that getting better speed and area.The proposed multiplexer shown in below.

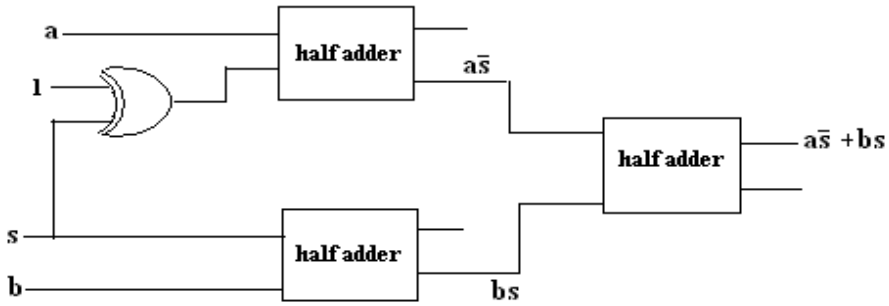

Figure 7: Proposed Multiplexer

8. Results and Conclusion

We design all multipliers using Hardware Description Language (HDL) for 8-bit unsigned data. To get power and speed report, we use XILINX ISE 10.1 as synthesis tool and Model-Sim XE III 6.3c for simulation. FPGA-Spartan III is used for implementation. The MCSA adder with fixed block size was implemented in 8x8 multiplier architecture. It gives better result than conventional adders in terms of speed and power consumption as shown in the Table-2. The Simulation Result of the proposed multiplier is shown in figure 8.

Table 2. Comparison of Delay and Power of Proposed multiplier

Various Multipliers	**Delay(ns)**	**Power(w)**
Reduced Complexity Wallace multiplier(MCSA)	22.657	0.157
Dadda multiplier	25.859	0.624
Multiplier using counter	25.491	0.173
Booth multiplier	29.981	0.376

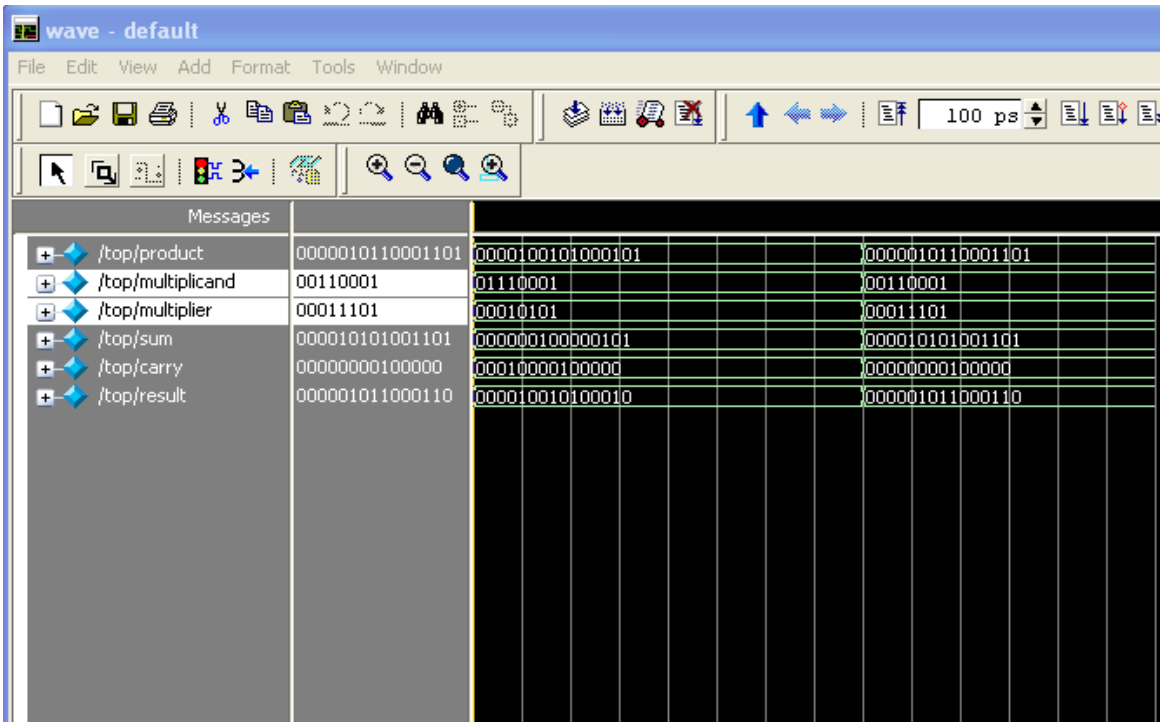

Figure8.Reduced Wallace multiplier with Modified Carry save Adder (MCSA)

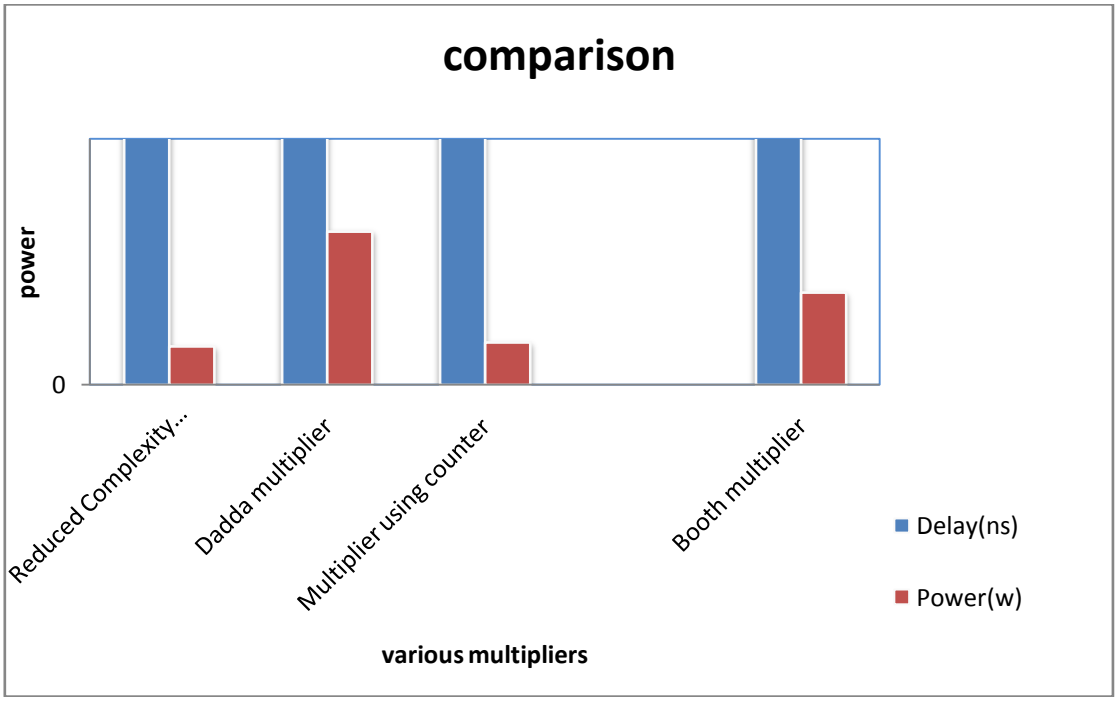

Figure 9: comparisons graph various multipliers with their delay and power

References

[1] C.Y.H. Lee. "A Performance Comparison Study on Multiplier Designs," B.Eng. Project Report, University Technology PETRONAS, 2009.

[2] Chris Y.H. Lee, Lo Hai Hiung, Sean W.F. Lee, Nor Hisham Hamid "A Performance Comparison Study on Multiplier Designs".

[3] Soojin Kim and Kyeongsoon Cho "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges" World Academy of Science, Engineering and Technology 61 2010

[4] Habibi and P.A. Wintz. "Fast Multipliers," IEEE Trans. on Computers, vol. 19, pp. 153-157, 1970.

[5] B.Ramkumar,Harish M Kittur, P.Mahesh Kannan, "ASIC Implementation of Modified Faster Carry Save Adder", European Journal of Scientific Research, ISSN 1450-216X Vol.42 No.1, pp.53-58, 2010.

[6] S. Shah, A.J. Al-Khalili and D. Al-Khalili. "Comparison of 32-bit Multipliers for Various Performance Measures," in the 12thInternational Conference on Microelectronics, pp. 75-80, 2000.

[7] P.C.H. Meier, R.A. Rutenbar and L.R. Carley, "Exploring Multiplier Architecture and Layout for Low Power,' in IEEE Custom Integrated Circuits Conf., pp. 513-516, 1996.

[8] Kousuke TARUMI, Akihiko HYODO, Masanori MUROYAMA, Hiroto YASUURA, "A design method for a low power digital FIR filter in digital wireless communication systems," 2004.

[9] K.C. Bickerstaff and E.E. Swartzlander, Jr. "Analysis of Column Compression Multipliers," in 15thIEEE Symp. On Computer Arithmetic, pp. 33-39, 2001.

[10] Dandapat, S. Ghosal, P. Sarkar, D. Mukhopadhyay"A 1.2-ns16×16-BitBinaryMultiplier Using High Speed Compressors" International Journal of Electrical, Computer, and Systems Engineering 4:3 2010.

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

[12] L. Dadda. "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, 1965.

[13] K.A.C. Bickerstaff, M. Schulte, and E.E. Swartzlander, Jr., "Reduced Area Multipliers," Intl. Conf. on Application-Specific Array Processors, pp. 478-489, 1993.

[14] S.Rajaram , Mrs.K.Vanithamani "Improvement of Wallace multipliers using Parallel prefix adders"Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011).