International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 2, Issue 3, pp: (182-191), Month: July - September 2014, Available at: www.researchpublish.com

COMPARATIVE ANALYSIS ON POWER AND DELAY OPTIMIZATION OF VARIOUS MULTIPLIERS USING VHDL

[1]Shubhi Shrivastava, [2]Pankaj Gulhane

[1]DIMAT Raipur, Chhattisgarh, India
[2]DIMAT Raipur, Chhattisgarh, India

Abstract: **Paper shows the implementation of improved Radix 4 Booth multiplier in VHDL software. Booth multiplication allows for faster, smaller multiplication circuits through encoding the signed numbers to its 2's complement and which is also one of the standard technique used in chip designing. The algorithm reduces the number of partial product to half over "long multiplication" techniques. Proposed optimized Radix-4 Booth's multiplier modify the way it does the addition of partial products. All these multiplier designs were modelled in Verilog HDL. Paper presents comparative analysis of Radix-4 booth multiplier with different multiplier techniques. Comparison shows optimized Radix-4 multiplier is superior from other multipliers in terms of various constraints.**

Keywords: **Booth's algorithm, Radix-4 booth, Radix-2 multiplier, Vedic multiplier, Bypass multiplier, Shift and Add multiplier.**

I. INTRODUCTION

Multiplication is an important fundamental function in arithmetic operations. Multiplication dependent operations such as Multiply and Accumulate(MAC) and inner product are some of the mostly used computation currently implemented in many Digital Signal Processing (DSP) applications like convolution, Fast Fourier Transform(FFT), filtering and in microprocessors arithmetic and logic unit [1]. Since multiplication dominates the execution time of most DSP algorithms for the multiplication, so there required a high speed multiplier. Multiplication time is still the dominant factor in determining the instruction cycle time of a DSP chip.

The multiplier is a fairly large block of a computing processing system. The amount of circuitry involved is directly proportional to the square of its resolution (A multiplier of size n bits has n2 gates). For different multiplication algorithms performed in DSP applications throughput and latency are the two major concerns from delay perspective of multiplier. Latency in the DSP application is the real delay of computing a function and measure of how long the inputs to a device are stable is the final result available on output of the device. Throughput is the measure of how many number of multiplications can be performed in a particular period of time; multiplier is not only a high delay block but also a major source of power consumption. That's why if we aims to minimize power consumption, it is very important to reduce the delay by using various delay optimizations. One of the key arithmetic operations in such applications is multiplication and the development of fast multiplier circuit has been a subject of interest over decades. Reducing the time delay and power consumption are very essential requirements for many applications [2] [3]. Minimizing power consumption for digital systems involves optimization at all levels of the circuit design. This optimization includes the different technologies used to implement the digital circuits, the topology and circuit style, and the architecture for implementing the different circuits at the highest level the algorithms that are being implemented

Booth multiplication is one of the important multiplication algorithm used for bit multiplication. Large number of booth arrays is required for high speed multiplication and exponential operations which in turn require large partial sum and partial carry registers. The multiplication of two n-bit operands using a radix-2 booth recording multiplier requires approximately n / (2m) clock cycles to generate the least significant half of the final result, where m is the number of

Booth recorded in the adder stage. Because of that large propagation delay is associated with this type of case. Because of the importance of digital multipliers in DSP, it has always been an active area of research and a number of interesting multiplication algorithms have been reported in the literature [4].

The paper is subdivided into three parts : first part shows working techniques of various multipliers like Radix-2 Booth multiplier, Vedic multiplier, Bypass multiplier, Shift and Add multiplier. Second part describes the optimized Radix-4 Booth multiplier. Last part presents the comparative analysis of all multipliers in ground of hardware used, I/O power consumption and delay.

II. BOOTH'S ALGORITHM

The booth's algorithm was invented by Andrew Donald Booth in 1950 while he was doing study on crystallography at Birkbeck College in Bloomsbury, London.

Signed multiplication is a cubersome process. Through unsigned multiplication there is no need to take the sign of the number into consideration. Even though in signed multiplication the same method cannot be applied for the reason that the signed number is in a 2's compliment form which would give in an inaccurate result if multiplied in an analogous manner to unsigned multiplication [6].

Thus here Booth's algorithm comes in. Booth's algorithm conserves the sign of the end result. While doing multiplication, strings of 0s in the multiplier call for only shifting. While doing multiplication, strings of 1s in the multiplier need an operation only at each end. We require adding or subtracting merely at positions in the multiplier where there is a switch from 0 to 1 or from 1 to 0. In the following flow chart we have, B=Multiplier, A=Multiplicand, m= Product [7].

A and B are registers where multiplicand and multiplier are placed respectively.B-1 is placed logically to the right of LSB(least significant bit) B0 . Q and B-1 are initially set to 0. Control logic checks the two bits B0 and B-1. If the two bits are same (00 or 11) then all of the bits of Q, B, B-1 are shifted 1 bit to the right. If both are not the same and if the combination is 10 then the multiplicand is subtracted from Q and if the combination is 01 then the multiplicand is added with Q. In both the cases results ar Q, B, B-1 are right shifted. The shifting is the arithmetic right shift operation where the left most bit namely, Q_{n-1} is not only shifted into Q_{n-2} but also remains in Q_{n-1}. This is to preserve the sign of the number in Q and B. The result of the multiplication will appear in the Q and B.

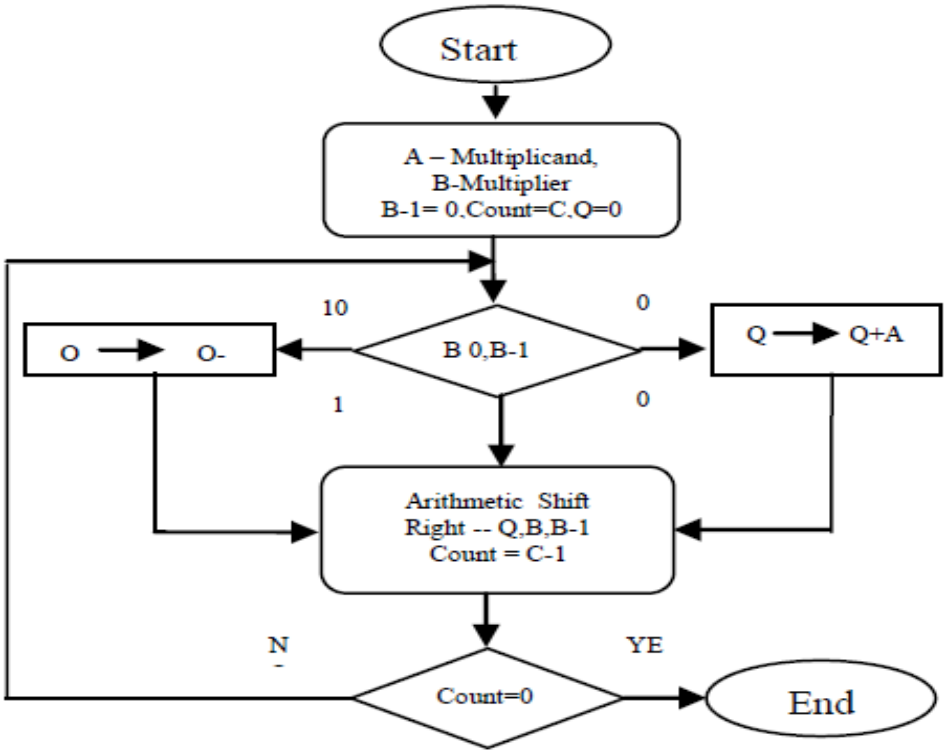


Fig.1. Flow chart of Booth's Algorithm

III. MULTIPLICATION ALGORITHMS

A. Booth Multiplication Algorithm for Radix-2

It will encode the multiplicand based on multiplier bits. In Radix -2 we will compare 2 bits at a time with overlapping method. Grouping starts from the LSB, and the first block only uses one bit of the multiplier and assumes a zero for the second bit.

Table 1: Radix-2 Booth Encoding Table

Block	Partial Product
00	0
01	1*Multiplicand
10	-1*Multiplicand
11	0

The functional operation of booth encoder is tabulated in Table 1. There are two inputs for booth encoder one is multiplicand and the other is 2 bits from multiplier, based on these two inputs it will encode the multiplicand.

The state diagram of the Radix-2 Booth multiplier is shown in Figure 2. Here we have four different types of states. For 00, 11 states we can perform multiplication of multiplicand with zero. For 01 state, we can multiply multiplicand with one whereas for 10 states, we can multiply multiplicand with -1.

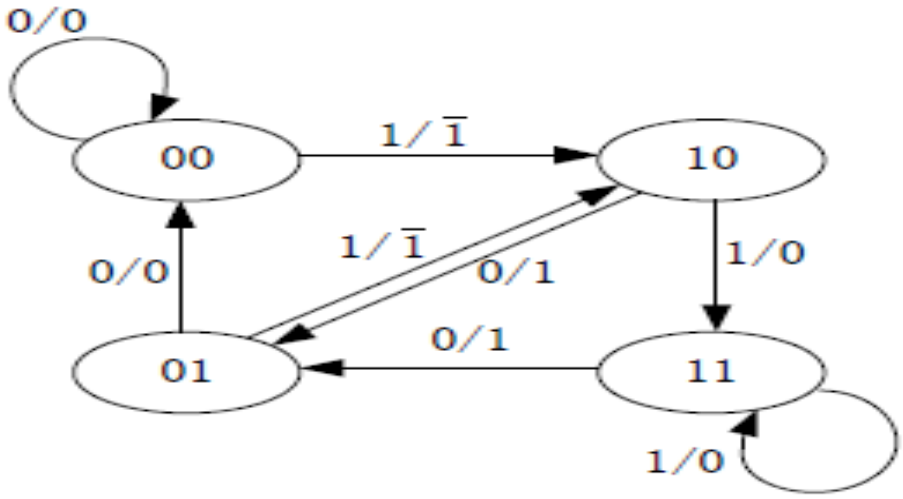


Fig.2. State diagram for Radix-2 Multiplier

B. Vedic Multiplication

Vedic mathematics is part of four Vedas (books of wisdom). It is part of Sthapatya- Veda (book on civil engineering and architecture), which is an upa-veda (supplement) of Atharva Veda. Several mathematical terms including geometry (plane, co-ordinate), arithmetic, quadratic equations, trigonometry, factorization and even calculus are explained in it.

His Holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884- 1960) comprised all this work together and gave its mathematical explanation while discussing it for various applications. Swamiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae) after extensive research in Atharva Veda. These formulae are not going to be found in present text of Atharva Veda because these formulae were constructed by Swamiji. Vedic mathematics is not only considered as a mathematical wonder but also it is logical proved. That's why it has such a high degree of eminence which cannot be disapproved by any one. Because of these phenomenal characteristics, Vedic math has already crossed the boundaries of India and has become an interesting topic of research internationally. Vedic math deals with different basic as well as complex mathematical operations. Especially, different methods of basic arithmetic are extremely simple and powerful [2] [3].

The word "Vedic" is derived from the word "Veda" which means the store-house of all knowledge. Vedic mathematics is basically based on 16 Sutras (or aphorisms) dealing with various branches of mathematics like arithmetic, algebra, geometry etc [8]. These Sutras along with their brief meanings are enlisted below alphabetically.[5]

1. (Anurupye) Shunyamanyat – If one is in ratio, then the other is zero.
2. Chalana-Kalanabyham – Differences and Similarities.
3. Ekadhikina Purvena – By one more than the previous One.
4. Ekanyunena Purvena – By one less than the previous one.
5. Gunakasamuchyah – The factors of the sum is equal to the sum of the factors.
6. Gunitasamuchyah – The product of the sum is equal to the sum of the product.
7. Nikhilam Navatashcaramam Dashatah – All from 9 and last from 10.
8. Paraavartya Yojayet – Transpose and adjust.
9. Puranapuranabyham – By the completion or noncompletion.
10. Sankalana- vyavakalanabhyam – By addition and by subtraction.
11. Shesanyankena Charamena – The remainders by the last digit.
12. Shunyam Saamyasamuccaye – When the sum is the same that sum is zero.
13. Sopaantyadvayamantyam – The ultimate and twice the penultimate.
14. Urdhva-tiryagbhyam – Vertically and crosswise.
15. Vyashtisamanstih – Part and Whole.
16. Yaavadunam – Whatever the extent of its deficiency.

These methods and ideas can be directly applied to plain, spherical geometry, trigonometry, calculus, conics (both integral and differential) and applied mathematics of different kinds. As we mentioned earlier, all these Sutras were reconstructed from ancient Vedic texts early in the last century. Many different Sub-sutras were also discovered at the same time. The beauty of Vedic mathematics lies in the fact that it reduces the otherwise cumbersome-looking calculations in conventional mathematics to a very simple form. May be because the Vedic formulae are claimed to be based on the natural principles on which the human mind is working. This is a very interesting field of study and presents some effective algorithms which can be applied to various branches of engineering such as computing and digital signal processing[1] [4].

The multiplier architecture can be generally classified into three different categories. First one is the serial multiplier which emphasizes on hardware and minimum amount of chip area. Second is parallel multiplier (array and tree) which carries out high speed arithmetical operations. But its drawback is the relatively larger chip area consumption. Third one is the serial- parallel multiplier which serves as a good trade-off between the times consuming serial multiplier and the area consuming parallel multipliers.

C. Bypass Multiplier

In Bypass Multiplier there are two different type of technique used Row Bypassing and Column Bypassing Techniques.

i) Row Bypassing Technique

The Row bypassing scheme disables the operation in some rows in order to save switching power consumption. To understand the row bypassing technique, let take an example of an unsigned 4×4 multiplication. In fig 2, if bit b_j is 0, all products in row j ($a_i b_j$ for $0 \leq i \leq n-1$) are 0. As a result, the addition in corresponding row can be bypassed. For example, let b1 of fig 3 be 0. For this case, the output from the first CSA row can be fed directly to the third CSA row and the second CSA row is disabled, thus by disabling the second CSA the switching activities are reduced row which results in low power dissipation.

Since the rightmost FA in the second row is disabled, so it does not execute the addition and because of that output is not correct. In order to remove this problem, extra circuit must be added in the row bypassing technique.

$$
\begin{array}{cccccccc}
 & A = & a_3 & a_2 & a_1 & a_0 & & \\
\times & B = & b_3 & b_2 & b_1 & b_0 & & \\
\hline
 & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & & \\
 & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & & \\
a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & & \\
a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & & \\
\hline
P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0
\end{array}
$$

Fig 3: Example of 4×4 multiplication

ii) Column Bypassing Technique

In this technique, if the corresponding bit in the multiplicand is 0, the operations in a column can be disabled. There are two advantages of this technique. Firstly, it eliminates the extra correcting circuit. Second, the modified FA is simpler than that used in the row-bypassing multiplier[9] [10]. To understand the column bypassing technique, let take an example of 4×4 multiplication as shown in Figure 4, which executes 1010×1111.

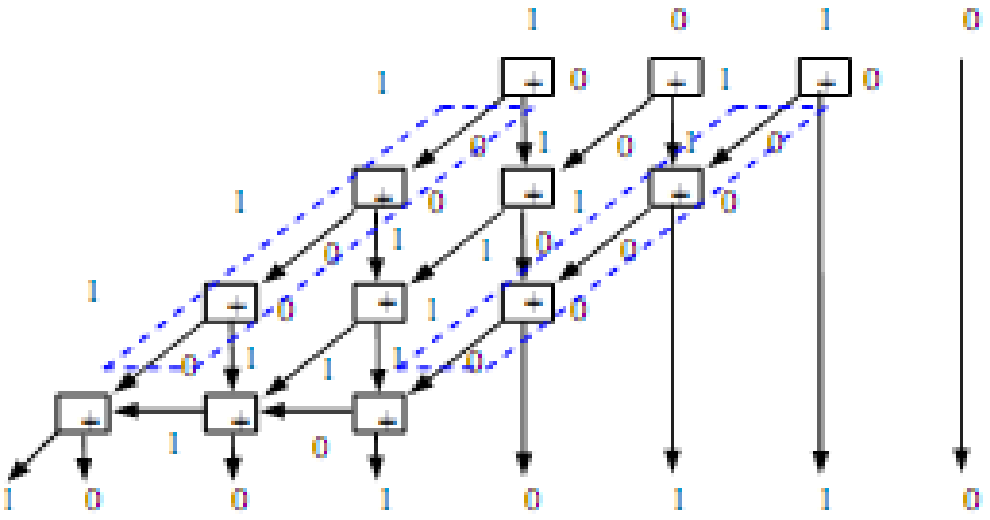


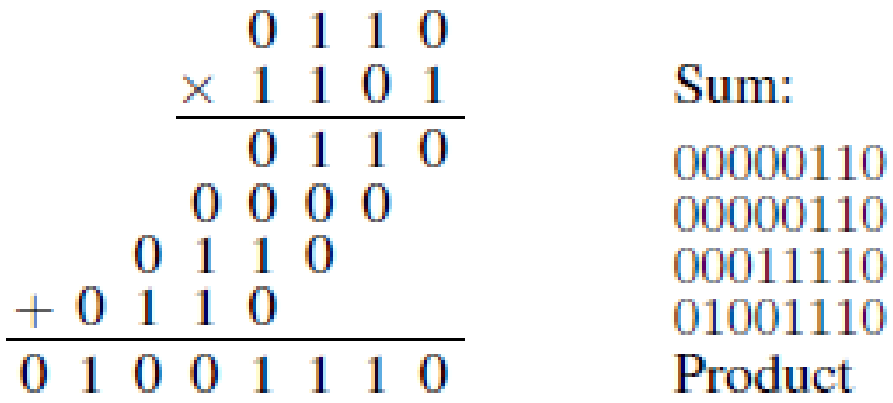
Fig 4: Example of column bypassing

Out of the three input bits, in the first and third diagonals (enclosed by dashed lines), two input bits are 0: the "carry" bit from its upper right FA, and the partial product $a_i b_j$ .Thus, the output carry bit of the FA is 0, and the output sum bit is equal to the third bit, which is the "sum" output of its upper FA.

D. Shift and Add Multiplier

Shift-and-add multiplication is similar to the multiplication performed by pencil and paper. This method adds the multiplicand *X* to itself *Y* times, where *Y* shows the multiplier.For multiplication of two numbers by paper and pencil, the algorithm has to take the digits of the multiplier one at a time from right to left, multiply the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

Example: Multiplication of 6 and 13, i.e. $6 \times 13 = 78$.



Method to avoid adder arrays. Shift register for partial product and multiplier with each cycle,

a) Partial product increases by one digit

b) Multiplier is reduced by one digit

MSBs of partial product and multiplicand are aligned in each cycle, multiplicand is not shifted.

1. Load multiplier into lower half of shift register (the upper half is to be zeroed)

2. Test LSB of the shift register

3. If LSB is set

- then add multiplicand to the upper half of the shift register
- else add nothing (make sure carry-bit is cleared!)

4. Perform right shift including carry on full shift register

5. Repeat process till point 2, as long as multiplier part of shift register is not empty.

6. After termination, the shift register (both halves!) contains the product.

Process is easy to implement in software.

E. Booth Multiplication Algorithm for Radix-4

Radix-4 Booth algorithm scans 3 bits of the strings with the algorithm given below:

1. Extending the sign bit 1 position if require, to make sure that n is even only.
2. Append a 0 to the right side of the least significant bit of the multiplier.
3. According to the value of each vector, Partial Product will be 0, +Y, –Y, +2Y,–2Y. The negative values of y are considered by taking the 2‘s complement to the Booth recode the multiplier term, we have to consider the bits in groups of three, in a way that each group overlaps with the previous group by one bit. Grouping starts from the LSB and the first group only uses 2 bits of the multiplier [11].

Let us take an example:

Multiplicand is (001011)

Multiplier is (010011)

Now we will consider the group of three bits for Multiplier [12] .

Recoding for radix-4 booth multiplier will be done according to the following table which is as below:

Table 2: Radix-4 Booth Encoding Table

Groups	Partial products
000	0
001	1*multiplicand
010	1*multiplicand
011	2*multiplicand
100	-2*multiplicand
101	-1*multiplicand
110	-1*multiplicand
111	0

Now according to the Table 2: We get to know that

(010) -1

(001) - 1

(110) - (-1)

So Multiplicand is multiplied with the three encoded digits which are 1, 1 and –1.

(i) –1 * (001011)

= 001011

And now 1111 is added with the result because of the negative sign. So final result of multiplication of -1 is 1111001011, here negative term sign is extended.

(ii) 1 * 001011= 001011

(iii) 1 * 001011 = 001011

(iv) 00001 are then get added with these 3 resultants as the error correction of the negation.

1 1 1 1 1 1 0 1 0 0 here also negative term sign is extended

0 0 1 0 1 1

0 0 1 0 1 1

0 0 0 0 1 here is error correction for negation.

0 0 1 1 0 1 0 0 0 1 here we have discarded the carried high bit.

The state diagram of the Radix-4 Booth multiplier is shown in Fig.5. It consists of eight different types of states and during these states we can obtain the outcomes, which are multiplication of multiplicand with 0,-1 and -2 consecutively [13]. The pictorial view of the state diagram presents various logics to perform the Radix-4 Booth multiplication in different states as per the different encoding technique.

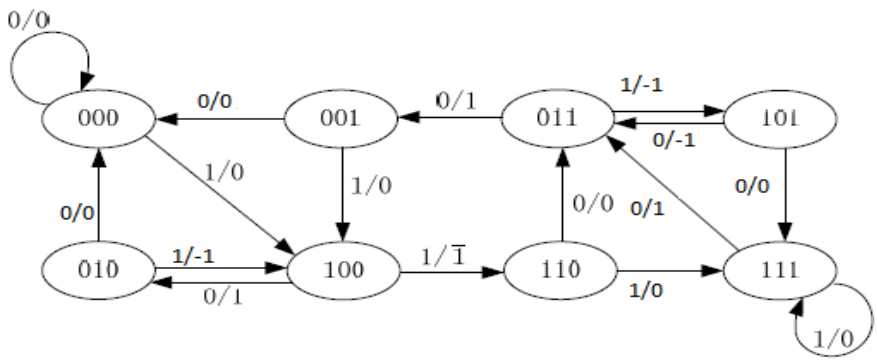


Fig.5. State diagram for Radix-4 Multiplier

In our proposed method we used the same hardware in Radix-4 booth multiplier to optimize the multiplier. In optimization table the 8 conditions of partial products covered in 4 conditions. The optimized Radix-4 booth multiplier encoding table shows in Table 3.

Table 3: Optimized Radix-4 Booth Encoding Table

Groups	Partial products
001 or 010	1*multiplicand
011	2*multiplicand
100	-2*multiplicand
101 or 110	-1*multiplicand

IV. SIMULATION RESULTS

Figure:6 below shows the simulation output of the Booth radix-4 multiplier algorithm for the multiplication of the two bytes.

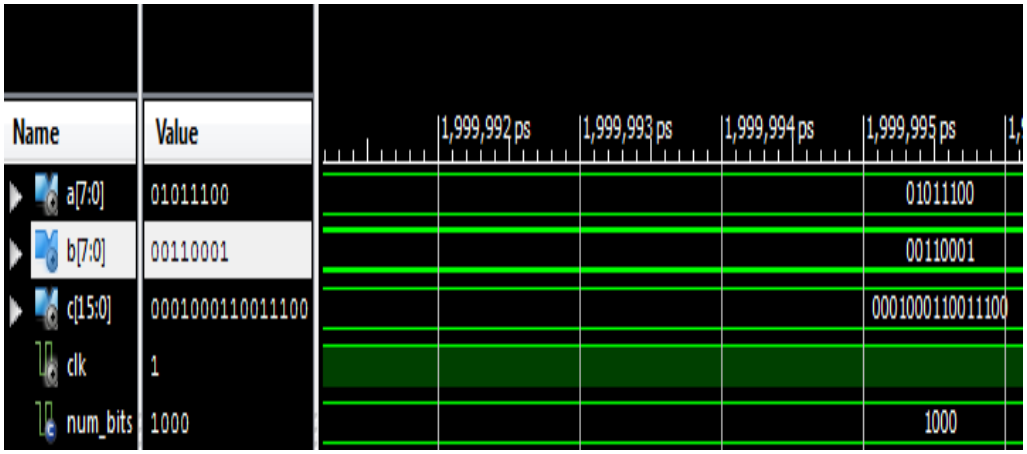


Fig. 6: Optimized booth radix-4 multiplier algorithm

Table 3 shows the total delay required by the different multiplier techniques for the multiplication. Table 4 shows the total power required by the different multiplier techniques for the multiplication [14] and Table 5 shows the total units required by Proposed Radix-4 Booth multipliers in multiplication.

Table 4: Comparison table of total delay by different multipliers

Sr. no.	Algorithm	Total Delay
1	Proposed Booth Radix- 4 Multiplier	2.203ns
2	Booth Radix-2 Multiplier	4..261ns
3	Vedic Multiplier	27ns
4	Bypass Multiplier	7.24ns
5	Shift and Add Multiplier	43.42ns

Table 5: Comparison table of total power by different multipliers

Sr. no.	Algorithm	Total Power
1	Proposed Booth Radix- 4 Multiplier	27mw
2	Booth Radix-2 Multiplier	43mW
3	Vedic Multiplier	63.74mw
4	Bypass Multiplier	70.6mw
5	Shift and Add Multiplier	71.8mw

Table 6: Table of total units used by Proposed Radix-4 Booth multipliers

Sr. no.	Algorithm	Total Units Used
1	Number of Slice Flip Flops	35
2	Number of 4 input LUTs	595
3	Number of bonded IOBs	33

V. CONCLUSION

From the above simulation results successful implementation of the optimized Booth radix-4 multiplier is showed. The high speed implementation of such a multiplier has wide range of applications in image processing, arithmetic logic unit and VLSI signal processing. The comparison tables for power, delay in the different multiplier techniques shows that the Booth radix-4 technique gives the best and appropriate results.

REFERENCES

[1] Wallace, C.S., "A suggestion for a fast multiplier," IEEE Trans. Elec. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[2] Booth, A.D., "A signed binary multiplication technique," Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236– 240, 1951.

[3] Jagadguru Swami Sri Bharath, Krsna Tirathji, "Vedic Mathematics or Sixteen Simple Sutras From The Vedas", Motilal Banarsidas, Varanasi(India),1986.

[4] A.P. Nicholas, K.R Williams, J. Pickles, "Application of Urdhava Sutra", Spiritual Study Group, Roorkee (India), 1984.

[5] Shripad Kulkarni, "Discrete Fourier Transform (DFT) by using Vedic Mathematics"Papers on implementation of DSP algorithms/VLSI structures using Vedic Mathematics, 2006, www.edaindia.com, IC Design portal.

[6] Laxman S, Darshan Prabhu R, Mahesh S Shetty ,Mrs. Manjula BM, Dr. Chirag Sharma, "FPGA Implementation of Different Multiplier Architectures", International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 6, June 2012

[7] Dr. Ravi Shankar Mishra,Prof. Puran Gour,Braj Bihari Soni, "Design and Implements of Booth and Robertson's multipliers algorithm on FPGA." International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622.

[8] Jeganathan Sriskandarajah, "Secrets of Ancient Maths: Vedic Mathematics", Journal of Indic Studies Foundation, California, pages 15 and 16.

[9] Jin –Tai Yan, Zhi –Wei Chen "Low Cost Low Power Bypassing –Based Multiplier Design", Dept. of Computer Science. & Information., Chung-Hua Univ., Hsinchu, Taiwan, IEEE, pp2338 - 2341., 2010.

[10] Sunjoo Hong, Taehwan Roh and Hoi-Jun Yoo, "a 145w 8×8 parallel multiplier based on optimized bypassing architecture", department of electrical engineering, Korea advanced institute of science and technology (KAIST), Daejeon, Republic of Korea, IEEE, pp.1175-1178, 2011.

[11] Sandeep Shrivastava*, Jaikaran Singh* and Mukesh Tiwari*, "Implementation of Radix-2 Booth Multiplier and Comparison with Radix-4 Encoder Booth Multiplier," International Journal on Emerging Technologies 2(1): 14-16(2011) ISSN : 0975-8364

[12] Ravindra P Rajput , M.N.Shanmukha Swamy, "High speed Modified Booth Encoder multiplier for signed and unsigned numbers," IEEE International Conference On Modelling and Simulation , 2012

[13] Kelly Liew Suet Swee, Lo Hai Hiung, "Performance Comparision Review of Radix Based Multiplier Designs", IEEE,International Conference On Intelligent and Advance Systems (ICIAS2012) , 2012

[14] Prabhu, A.S., Elakya , "Design of Modified Low Power Booth Multiplier", IEEE International Conference On Computing, Communication and Applications (ICCCA), 2012