ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design of High Speed 2's Complement Multiplier

Mr. Ankit Bhatt
Student of ME, ENTC,
Dept of VLSI and Embedded systems,
Matoshri College of Engineering and Research Centre, Nashik, India.

Mr. Sachin D. Pable
Associate Professor,
Dept of VLSI and Embedded systems,
Matoshri College of Engineering and Research Centre, Nashik, India.

Abstract:

The speed of multiplication operation is of great importance in digital signal processing as well as in the general purpose processors. The reduction can be achieved by Modified Booth Encoded multiplier Technique. Two's complement multipliers are used in wide range of applications like multimedia, 3D graphics, signal processing etc. The proposed method used the Radix-4 Booth Multipliers to achieve the low area and increase the speed by modifying the partial product matrix size. A multiplier is designed with 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. In the second stage, the two final products are added to form the final product through a final adder. The Multiplier design implemented using Xilinx. The results based on a rough theoretical analysis and on logic synthesis showed its efficiency in terms of both area and delay. It is compared with conventional multiplier and Radix-4 (short bit-width) Modified booth encoded Multiplier.

Index Terms:

Multiplication, Modified Booth Encoding, Partial Product Array.

I INTRODUCTION:

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier.

In signal processing applications performance strongly depends on the effectiveness of the hardware used for computing multiplications. The high interest in this field is witnessed by the large amount of algorithm and implementations of the multiplication operations. In this short bit width (8-16 bits) two's complement multipliers with single-cycle throughput and latency have emerged to be important building blocks for high performance.

Two's complement:

The two's complement of a binary number is defined as the value obtained by subtracting the number from a large power of two (specifically, from 2N for an N-bit two's complement). The two's complement of the number then behaves like the negative of the original number in most arithmetic, and it can coexist with positive numbers in a natural way. A two's-complement system, or two's-complement arithmetic, is a system in which negative numbers are represented by the two's complement of the absolute value; this system is the most common method of representing signed integers on computers. In such a system, a number is negated (converted from positive to negative or vice versa) by computing its ones' complement and adding one.

An N-bit two's-complement numeral system can represent every integer in the range −2N−1 to 2N−1-1 while ones' complement can only represent integers in the range −(2N−1−1) to 2N−1−1The two's-complement system has the advantage of not requiring that the addition and subtraction circuitry examine the signs of the operands to determine whether to add or subtract. This property makes the system both simpler to implement and capable of easily handling higher precision arithmetic. Also, zero has only a single representation, obviating the subtleties associated with negative zero, which exists in ones'-complement systems embedded processors and DSP execution cores.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

II MODIFIED BOOTH RECODED MULTIPLIERS:

In general, a radix-B = 2b MBE leads to a reduction of the number of rows to about [n/b] while, on the other hand, it introduces the need to generate all the multiples of the multiplicand X, at least from –B/2 * X to B/2 * X. As mentioned above, radix-4 MBE is particularly of interest since, for radix-4, it is easy to create the multiples of the multiplicand 0; +-X; +-2X. In particular, +-2X can be simply obtained by single left shifting of the corresponding terms +-X. It is clear that the MBE can be extended to higher radices, but the advantage of getting a higher reduction in the number of rows is paid for by the need to generate more multiples of X.

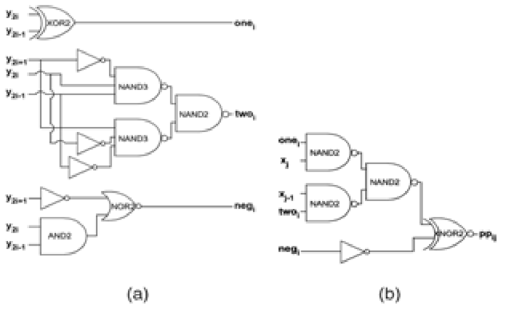


Fig 1 Gate-level diagram for partial product generation using MBE (a) MBE signals generation. (b) Partial product generation.

From an operational point of view, it is well known that the radix-4 MBE scheme consists of scanning the multiplier operand with a three-bit window and a stride of two bits (radix-4). For each group of three bits (y2i+1, y2i, y2i+1), only one partial product row is generated according to the encoding in Table 1. A possible implementation of the radix-4 MBE and of the corresponding partial product generation is shown in Fig. 1, which comes from a small adaptation of [10, Fig. 12b]. For each partial product row, Fig. 1a produces the one, two, and neg signals. These signals are then exploited by the logic in Fig. 1b, along with the appropriate bits of the multiplicand, in order to generate the whole partial product array. Other alternatives for the implementation of the recoding and partial product generation can be found in [13], [14], [15], among others.As introduced previously, the use of radix-4 MBE allows for the (theoretical) reduction of the PP rows to [n/2], with the2 possibility for each row to host a multiple of yi* X, with yi Є {0,+-1,+-2}.

While it is straightforward to generate the positive terms 0, X, and 2X at least through a left shift of X, some attention is required to generate the terms -X and -2X which, as observed in Table 1, can arise from three configurations of the y2i+1 , y2i , and y2i-1 bits. To avoid computing negative encodings, i.e., -X and -2X, the two's complement of the multiplicand is generally used. From a mathematical point of view, the use of two's complement requires extension of the sign to the leftmost part of each partial product row, with the consequence of an extra area overhead. Thus, a number of strategies for preventing sign extension have been developed. For instance, the scheme in relies on the observation that $-pp = \overline{pp} + 1 = \overline{pp} -$ 1-2+4

When 4-to-2 compressors are used, which is a widely used option because of the high regularity of the resultant circuit layout for n power of two, the reduction of the extra row may require an additional delay of two XOR2 gates. However, the reduction still requires additional hardware, roughly a row of n half adders. This issue is of special interest when n is a power of two, which is by far a very common case, and the multiplier's critical path has to fit within the clock period of a high performance processor. For instance, in the design presented in for n =16, the maximum column height of the partial product array is nine, with an equivalent delay for the reduction of six XOR2 gates . For a maximum height of the partial product array of 8, the delay of the reduction tree would be reduced by one XOR2 gate. Alternatively, with a maximum height of eight, it would be possible to use 4 to 2 adders, with a delay of the reduction tree of six XOR2 gates, but with a very regular layout.

Architecture of the Modified Booth Multiplier :

Fig. shows the architecture of the commonly used modified Booth multiplier. The inputs of the multiplier are multiplicand X and multiplier Y. The Booth encoder encodes input Y and derives the encoded signals The Booth decoder generates the partial products according to the logic diagram using the encoded signals and the other input X. The Wallace tree computes the last two rows by adding the generated partial products. The last two rows are added to generate the final multiplication results using the carry look-ahead adder (CLA).

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

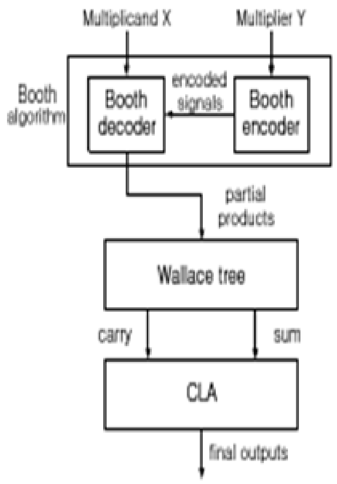


Fig: Architecture of the modified Booth multiplier

Any multiplier can be divided into three stages: Partial products generation stage, partial products addition stage, and the final addition stage.In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial products. In this stage, a second-order Booth encoding algorithm is usually used instead toreduce the number of partial products to half.

The second stage is the most important, as it is the most complicated and determines the speed of the overall multiplier.This paper focus on the optimization of the stage, which consists of the addition of all the partial products. If speed is not an issue, the partial products can be added serially, reducing the design complexity. However, in high-speed designs, the Wallace tree construction method is usually used to add the partial products in a tree-like fashion in order to produce two rows of partial products that can be added in the last stage.

Although fast, since its critical path delay is proportional to the logarithm of the number of bits in the multiplier, the Wallace tree introduces other problems such as wasted layout area and increased complexity, which we will elaborate on shortly.

III DESIGN METHODOLOGY:

The case of n * n square multipliers is quite common, as the case of n that is a power of two. Thus, we start by focusing our attention on square multipliers, and then present the extension to the general case of m * n rectangular multipliers.

3.1 Square Multipliers:

The proposed approach is general and, for the sake of clarity, will be explained through the practical case of 8 * 8 multiplications (as in the previous figures). As briefly outlined in the previous sections, the main goal of our approach is to produce a partial product array with a maximum height of [n/2] rows, without introducing any2 additional delay.Let us consider, as the starting point, the form of the simplified array as reported in Fig. 2, for all the partial product rows except the first one. As depicted in Fig. 6a, the first row is temporarily considered as being split into two sub rows, the first one containing the partial product bits (from right to left) from pp00 to pp80 bar and the second one with two bits set at "one" in positions 9 and 8. Then, the bit neg3 related to the fourth partial product row, is moved to become a part of the second sub row. The key point of this "graphical" transformation is that the second sub row containing also the bit neg3, can now be easily added to the first sub row, with a constant short carry propagation of three positions (further denoted as "3-bits addition"), a value which is easily shown to be general, i.e., independent of the length of the operands,

$$\overline{qq_{90}}\,qq_{80}\,qq_{30}\,qq_{70}\,qq_{60} = 0\,0\,\overline{pp_{80}}\,pp_{70}\,pp_{60} + 0\,1\,1\,0\,neg_3$$

for square multipliers. In fact, with reference to the notation of Fig. 6, we have that As introduced above, due to the particular value of the second operand, i.e., 0 1 1 0 neg3 , in [11], we have observed that it requires a carry propagation only across the least-significant three positions, a fact that can also be seen by the implementation shown in Fig. 4.1.

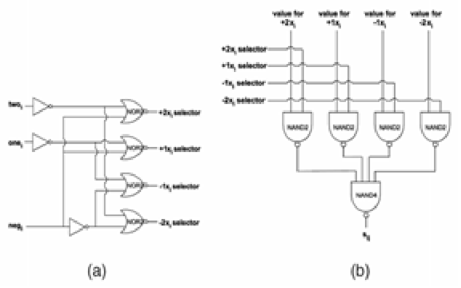


Fig.3.1 Gate-level diagram for the generation of two's complement partial product rows [9]. (a) 3-5 decoder. (b) 4-1 multiplexer.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

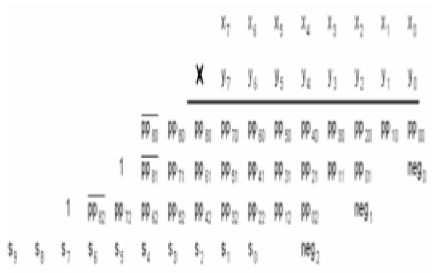


Fig. 3.2 Partial product array by applying the two's complement computation method in [9] to the last row.

It is worth observing that, in order not to have delay penalizations, it is necessary that the generation of the other rows is done in parallel with the generation of the first row cascaded by the computation of the bits qq70 qq60 in Fig. 6b. In order to achieve this, we must simplify and differentiate the generation of the first row with respect to the other rows. We observe that the Booth recoding for the first row is computed more easily than for the other rows, because the yà1 bit used by the MBE is always equal to zero. In order to have a preliminary

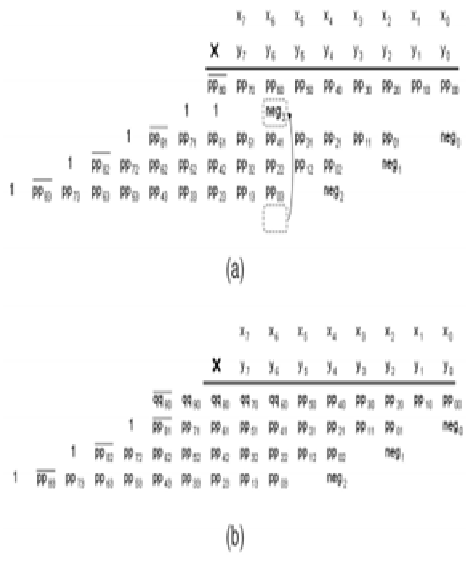


Fig. 3.3 Partial product array after adding the last negbit to the first row. (a) Basic idea. (b) Resulting array

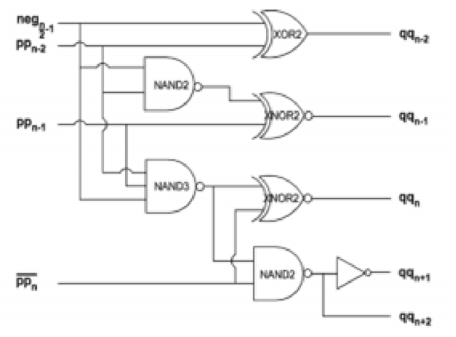


Fig.3.4 Gate-level diagram of the proposed method for adding the last negbit in the first row

Analysis which is possibly independent of technological details, we refer to the circuits in the following figures:

Fig. 3.1, slightly adapted from [10, Fig. 12], for the partial product generation using MBE;

Fig. 3.2, obtained through manual synthesis (aimed at modularity and area reduction without compromising the delay), for the addition of the last neg bit to the three most significant bits of the first row;

Fig. 3.3, obtained by simplifying Fig. 1 (since, in the first row, it is y2i-1= 0), for the partial product generation of the first row only using MBE; and

Fig. 3.4, obtained through manual synthesis of a combination of the two parts of Fig. 4.6 and aimed at decreasing the delay of Fig. 4.6 with no or very small area increase, for the partial product generation of the first row only using MBE.

IV PROPOSED WORK:

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. In the second stage, the two final products are added to form the final product through a final adder.The block diagram of traditional multiplier is depicted in Figure 3.1. 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. Carry propagation adder (CPA) is the final adder used to merge the sum and carry vector from the compressor module.

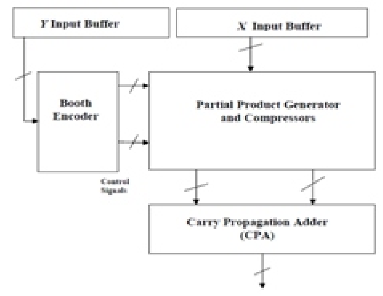

Y Input Buffer
X Input Buffer
Booth Encoder
Partial Product Generator and Compressors
Control Signals
Carry Propagation Adder (CPA)

Figure: Block Diagram of Booth Multiplier Architecture

A. Booth Encoder and Partial Product Generator:

The Booth encoder was implemented using two XOR gates and the selector using three MUXs and an inverter Careful optimization of the partial-product generation can lead to some substantial delay and hardware reduction. In the normal 8*8 multiplication 8 partial products need to be generated and accumulated. For accumulation seven adders to reduce power are required but in the case of booth multiplier only 4 partial products are required to be generated and for accumulation three adders, reduced delay required to compute partial sum and reduces the power consumption.

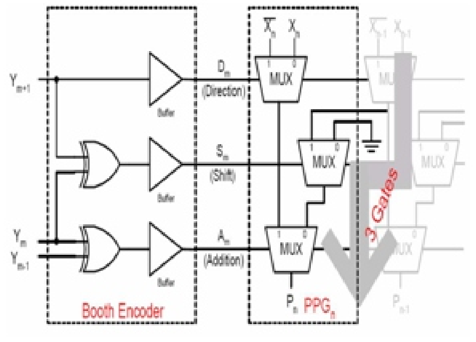


Figure 5.2 Booth encoder and PP (m=2i)

Partial product generation is the very first step in binary multiplier. Partial product generators for a conventional multiplier consist of a series of logic AND gates as shown in Figure 3.3.

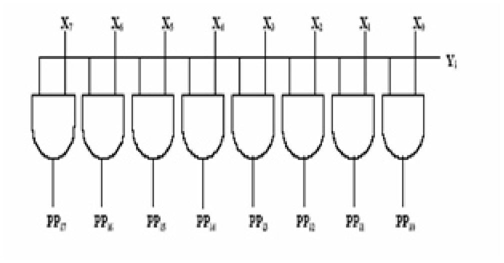


Figure 5.3 Partial Product generator using AND gates

If the multiplier bit is '0', then partial product row is also zero, and if it is '1', then the multiplicand is copied as it is. From the second bit multiplication onwards, each partial product row is shifted one unit to the left. In signed multiplication, the sign bit is also extended to the left.

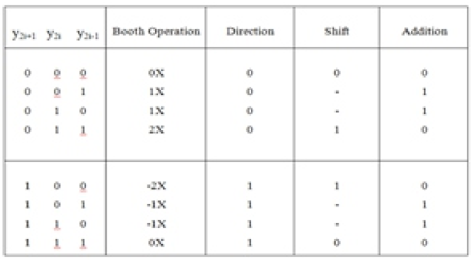
B. Carry Propagation Adder:

The final step in completing the multiplication procedure is to add the final terms in the final adder. The Carry Propagation Adder, CPA, is a final adder used to add the final carry vector to the final sum vector partial products to give the final multiplication result. This is normally called a "Vector-merging" adder. The choice of the final adder depends on the structure of the accumulation array. Various fast adders can be used as CPA. Some of them are Carry look-ahead adder, Simple carry skip adder, Multi level carry skip adder, Carry- select adder, Conditional sum adder and Hybrid adder. A Carry look-ahead Adder is an adder used in digital logic. All the carry outputs are calculated at once by specialized look-ahead logic. But requires generate and propagate signals. Simple carry skip adders looks for the cases in which carry out of a set of bits are identical to carry in. Circuits.

4.1 Modified Booth Algorithm:

The radix-2 disadvantages can be eliminated by examining three bits of Y at a time rather than two. The modified Booth algorithm is performed with recoded multiplier which multiplies only +a and +2a of the multiplicand, which can be obtained easily by shifting and/or complementation.The main advantage of the modified Booth algorithm is that it reduces the partial

The Truth table for booth encoding is shown in Table 4.1:



y_{2i+1}	y_{2i}	y_{2i-1}	Booth Operation	Direction	Shift	Addition
0	0	0	0X	0	0	0
0	0	1	1X	0	-	1
0	1	0	1X	0	-	1
0	1	1	2X	0	1	0
1	0	0	-2X	1	1	0
1	0	1	-1X	1	-	1
1	1	0	-1X	1	-	1
1	1	1	0X	1	0	0

Booth multiplication can be functionally divided into three basic operations, called ‘Direction’, ‘Shift’, and ‘Addition’.Direction determined whether the multiplicand was positive or negative, shift explained whether the multiplication operation involved shifting or not and addition meant whether the multiplicand was added to partial products. The for binary adders to efficiently skip a carry bit over two or more bit positions with two or more carry-skip paths is called multilevel carry skip adders. In the 4-bit carry select adder there are two 4-bit adders each of which takes a different preset carry-in bit. The two sums and carry-out bits that are produced are then selected by the carry-out from the previous stage. In conditional sum adder, sum and carry outputs at the first stage assume the previous carry to be zero and sum and carry outputs at the second stage assume the previous carry to be one. For CPA we can also combine any of these adders as a hybrid adder products to n/2. For radix-4 recoding, the popular algorithm is parallel recoding or Modified Booth recoding. In parallel radix-4 recoding, Y becomes:

$$Y = \sum_{i=0}^{n/2-1} vi\ 4i = \sum_{i=0}^{n/2-1} (-2y2i+1 + y2i + y2i-1)\ 4i \qquad (5)$$

Direction, $D_{2i} = Y_{2i+1} \longrightarrow$ (6)

Shift, $S_{2i} = Y_{2i-1} \cdot (Y_{2i+} \oplus Y_{2i}) + \oplus Y_{2i-1}' \cdot (Y_{2i+1}\ Y_{2i})$

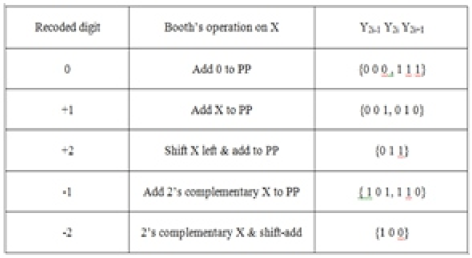
$= Y_{2i+1} \oplus Y_{2i} \longrightarrow$ (7)

Addition, $A_{2i} = Y_{2i-1} \oplus Y_{2i} \longrightarrow$ (8)

The following gives the algorithm for performing sign and unsigned multiplication operations by using radix-4 Booth recoding. Fast multipliers are imperative for high speed and low power signal processing systems and hence much thrust have been given to different design techniques.

As functional description cited above, multiplier consists of a Booth encoder, compressors, and carry propagation adders. The speed of the multiplier can be enhanced by reducing the number of partial products and thus the Booth algorithm plays a major role. In this chapter, we discuss about the related literature works for number of Booth encoder and the selector logic and the several design methods used to reduce the partial products. Booth encoding is a technique that leads to smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is the standard technique used in chip design, and provides significant improvements over the “long multiplication” technique. The widely used Booth algorithm is the radix-4 based modified Booth algorithm proposed by McSorley where it reduces the partial products into half. As the number of partial products reduces the number of CSAs required for the compression module, the height of the Wallace tree is also reduced. Booth recoding is fully parallel and carry free. It can be applied to design a tree and array multiplier, where all the multiples are needed at once. Radix-4 Booth recoding system works perfectly for both signed and unsigned operations.

Table 4 Partial Product Selections and Operations:



Recoded digit	Booth's operation on X	$Y_{2i+1}\ Y_{2i}\ Y_{2i-1}$
0	Add 0 to PP	{0 0 0 , 1 1 1}
+1	Add X to PP	{0 0 1, 0 1 0}
+2	Shift X left & add to PP	{0 1 1}
-1	Add 2's complementary X to PP	{1 0 1, 1 1 0}
-2	2's complementary X & shift-add	{1 0 0}

V IMPLEMENTATION AND RESULT:

The complete realization and results of the proposed designs using various low power multiplication techniques. All modules are realized using verilog HDL. The synthesis

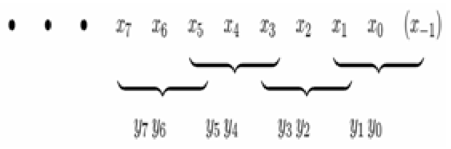

• • • x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0 (x_{-1})
$y_7\ y_6$
$y_5\ y_4$
$y_3\ y_2$
$y_1\ y_0$

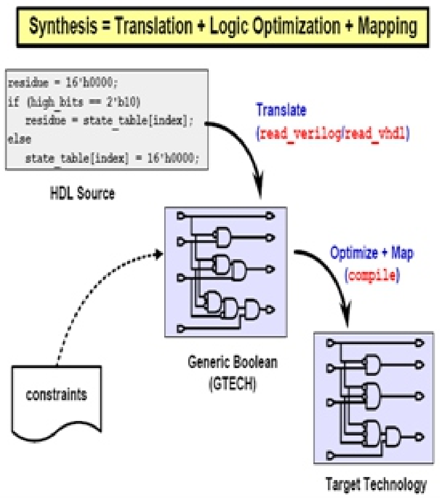
Figure 5.4 Modified Booth recoding pattern

Modified Booth algorithm’s basic idea is that the bits Yi and Yi-1 are recoded into Zi and Zi-1, while, Yi-2 serves as reference bit.

In a separate step, Yi-2 and Yi-3 recoded into Zi-2 and Zi-3 with, Yi-4 serving as reference bit. This signifies that the modified Booth's encoding partitions input Y into a group of 3-bits with 1-bit overlap and generates the following five signed digits, 2, 1, 0, -1 and -2. Encoding on the each group reduces the number of partial products by factor of 2. Operations on the encoded digits performed with multiplier input X is illustrated in Table 3.3. and power analysis of all modules are done by using of Xilinx ISE 12.1.The XST(Xilinx Synthesis Technology) Synthesizer of the ISE Tool converts HDL Model in to gate level netlist. The output results of the design are captured in the form of waveforms and netlist files and Time, Area (in the form of LUTs).

Design Flow:

There are wide varieties of ICs that can have their logic functions programmed into them after they had manufactured. Most of these devices used technology that also allows the function to be reprogrammed. FPGA is such a kind of IC where the user can program the functions realized by each logic cell and the connection between the cells.



```
residue = 16'h0000;
if (high_bits == 2'b10)
   residue = state_table[index];
else
   state_table[index] = 16'h0000;
```

Fig Synthesis and Procedure

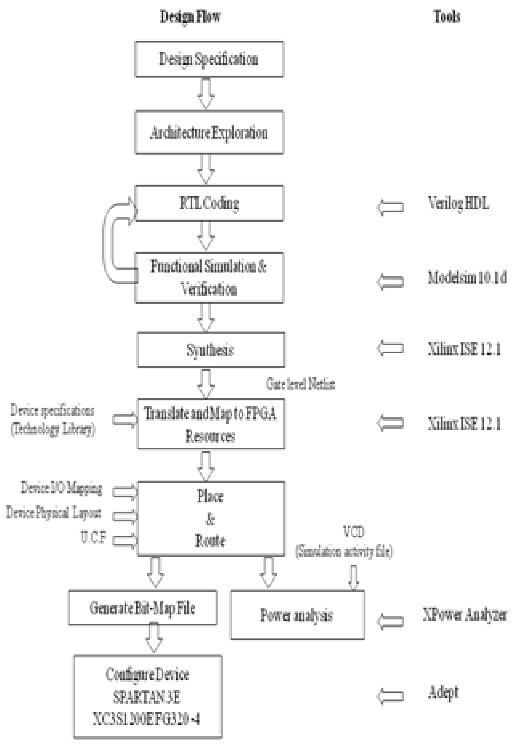


Fig FPGA Design Flow

The process of synthesis is described as translation + logic optimization + mapping. In terms of the Cadence or Synopsys tools, translation is performed by the read_vhdl/read_verilog commands. Logic optimization and mapping are performed by the synthesize/compile command. This process is illustrated in the figure below.

VI. RESULT:

Proposed booth multipliers using 8 bit The Figure shows the pin diagram of a16 tap FIR filter using shift and add multiplier. The 'clk' signal is used to time all the operations while the 'rst' signal is used to initialize the outputs and the internal registers. The multiplier of 8 bit and multiplicand of 8 bit are input. The output is product of 16 bit. The 'start' is used to start the operation.

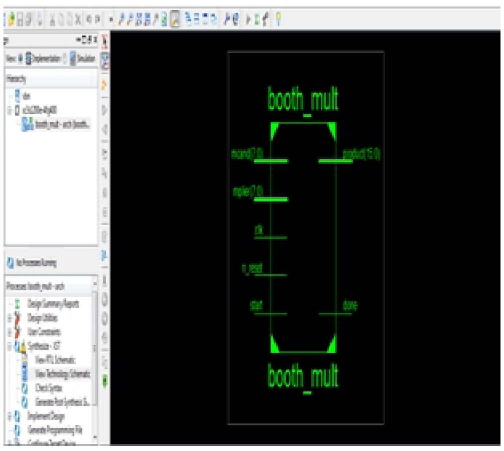


Figure: TOP Module of proposed booth multipliers using 8 bit

Simulation Results of proposed booth multipliers using 8 bit The Figure shows the working of a proposed booth multipliers using 8 bit with different combinations of multiplier and multiplicand. Multiplier is 6 and multiplicand is 2 to get product is 12.

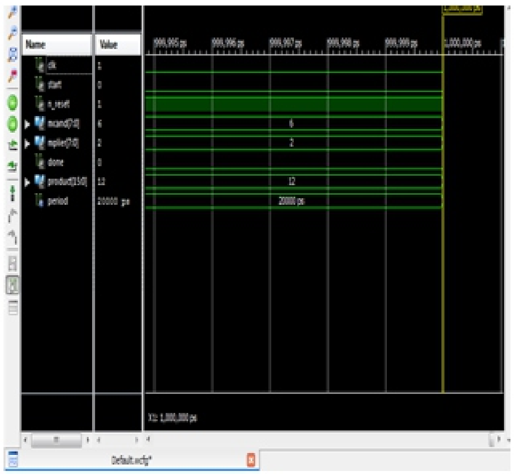


Figure : Waveform of proposed booth multipliers using 8 bit

Synthesis Results of proposed booth multipliers using 8 bit The RTL Schematic View of the proposed booth multipliers using 8 bit is shown in Figure 4.21.

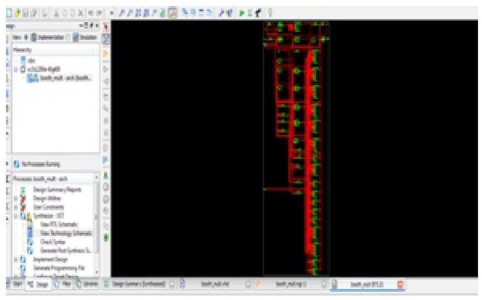


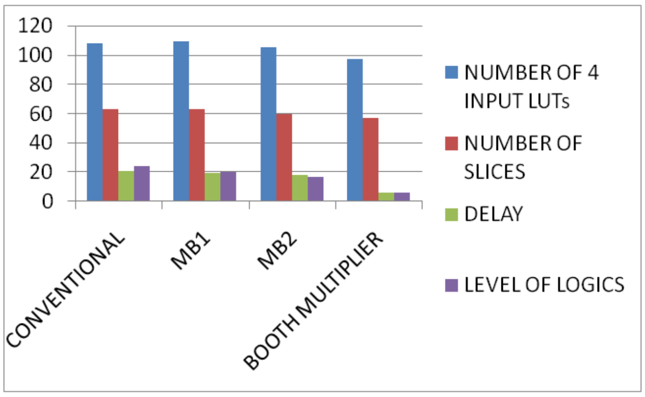
Figure : RTL Schematic View of proposed booth multipliers using 8bit

Performance Characteristics:

The Table 4.1 shows the performance comparison of various multipliers. It can be concluded that the proposed booths multiplier takes very less time over the various multiplication techniques. The least power consumption is obtained in proposed multiplier compared to all other multiplier architectures.

Multipliers Design using 8 bit	Number of slices	Number of 4 input LUT	Number of logic	Total delay (ns)	Total power
Conventional Multiplier	5067	31.86	34.197	547.152	931.65
modified radix-4-1 multiplier	4847	30.22	21.645	346.32	1396.144
modified radix-4-2 multiplier	2143	2.49	7.508	31473.536	331.645
Proposed Multiplier	1880	2.38	7.960	636.8	298.99

Result Graph:



VII CONCLUSION:

Two's complement n x n multipliers using radix-4 Modified Booth Encoding produce [n/2] partial products but due to the2 sign handling, the partial product array has a maximum height of [n/2] + 1. This paper a partial product array with a maximum height of [n/2], without 2 introduces any extra delay in the partial product generation stage. With the extra hardware of a (short) 3-bit addition, and the simpler generation of the first partial product row, we have been able to achieve a delay within the bound of the delay of a standard partial product row generation.. This is of special interest for all multipliers, and especially for single-cycle short bit-width multipliers for high performance embedded cores, where short bit-width multiplications are common operations improve both the performance and area requirements of square multiplier designs. This approach also applies with minor modifications to general Modified Booth Encoding multipliers.

REFERENCES:

1)M.D. Ercegovac and T. Lang, Digital Arithmetic. Morgan Kaufmann Publishers, 2003.

2)S.K. Hsu, S.K. Mathew, M.A. Anders, B.R. Zeydel, V.G.Oklobdzija, R.K. Krishnamurthy, and S.Y. Borkar, “A 110GOPS/ W 16-Bit Multiplier and Reconfigurable PLA Loop in 90-nm CMOS,” IEEE J. Solid State Circuits, vol. 41, no. 1, pp. 256-264, Jan.2006.

3)H. Kaul, M.A. Anders, S.K. Mathew, S.K. Hsu, A. Agarwal, R.K.Krishnamurthy, and S. Borkar, “A 300 mV 494GOPS/W Reconfi-gurable Dual-Supply 4-Way SIMD Vector Processing Accelerator in 45 nm CMOS,” IEEE J. Solid State Circuits, vol. 45, no. 1, pp. 95-101, Jan. 2010.

4)M.S. Schmookler, M. Putrino, A. Mather, J. Tyler, H.V. Nguyen, C.Roth, M. Sharma, M.N. Pham, and J. Lent, “A Low-Power, High-Speed Implementation of a PowerPC Microprocessor Vector Extension,” Proc. 14th IEEE Symp. Computer Arithmetic, pp. 12-19,1999.

5)F. Lamberti, N. Andrikos, E. Antelo, and P. Montuschi,“Speeding-Up Booth Encoded Multipliers by Reducing the Size of Partial Product Array,” internal report, http://arith.polito.it/ir_mbe.pdf, pp. 1-14, 2009.

6)L. Dadda, “Some Schemes for Parallel Multipliers,” Alta Frequenza,vol. 34, pp. 349-356, May 1965.

7)C.S. Wallace, “A Suggestion for a Fast Multiplier,” IEEE Trans.Electronic Computers, vol. EC-13, no. 1, pp. 14-17, Feb. 1964.D.E. Shaw, “Anton: A Specialized Machine for Millisecond-ScaleMolecular Dynamics Simulations of Proteins,” Proc. 19th IEEE Symp. Computer Arithmetic, p. 3, 2009.

8)J.-Y. Kang and J.-L. Gaudiot, “A Simple High-Speed Multiplier Design,” IEEE Trans.Computers, vol. 55, no. 10, pp. 1253-1258, Oct.2006.

9)O.L. MacSorley, “High Speed Arithmetic in Binary Computers,”Proc. IRE, vol. 49, pp. 67-91, Jan. 1961 .

10) J.-Y. Kang and J.-L. Gaudiot, “A Fast and Well-Structured Multiplier,” Proc. Euromicro Symp. Digital System Design, pp. 508-515, Sept. 2004.Z. Huang and M.D. Ercegovac, “High-Performance Low-Power Left-to-Right Array Multiplier Design,” IEEE Trans. Computers,vol. 54, no. 3, pp. 272-283, Mar. 2005.

11) R. Zimmermann and D.Q. Tran, “Optimized Synthesis of Sum-of-Products,” Proc. Conf. Record of the 37th Asilomar Conf. Signals,Systems and Computers, vol. 1, pp. 867-872, 2003.

12)V.G. Oklobdzija, D. Villeger, and S.S. Liu, “A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach,” IEEE Trans.Computers, vol. 45, no. 3, pp. 294-306, Mar. 1996.

13)P.F. Stelling, C.U. Martel, V.G. Oklobdzija, and R. Ravi, “Optimal Circuits for Parallel Multipliers,” IEEE Trans. Computers, vol. 47,no. 3, pp. 273-285, Mar. 1998.

14)J.-Y. Kang and J.-L. Gaudiot, “A Logarithmic Time Method for Two’s Complementation,” Proc. Int’l Conf. Computational Science, pp. 212-219, 2005.

15)K. Hwang, Computer Arithmetic Principles, Architectures, and Design.Wiley, 1979.

16)R.Hashemian and C.P.Chen, “A New Parallel Technique for Design of Decrement/Increment and Two’s Complement Circuits,” Proc. 34th Midwest Symp. Circuits and Systems, vol. 2,pp. 887-890, 1991.

17)D. Gajski, Principles of Digital Design. Prentice-Hall, 1997.STMicroelectronics, “130nm HCMOS9Cell Library,”http://www.st.com/stonline/products/technologies/soc/evol.htm, 2010.