High Speed Booth Multiplier Using Shift-Opt Algorithm

Yogeswari A*, Srilaya N*, Senthilkumar K K**
Final year students*, Associate Professor**
Department of Electronics and Communication Engineering
Prince Shri Venkateshwara Padmavathy Engineering College, Chennai 127.

***Abstract*—Multipliers play a pivotal role in the Digital Signal Processing systems and form a basic block in every ALU and MAC units. An effective multiplier is designed by considering certain parameters such as speed, power consumption, area requirement and complexity. Among various multipliers, Booth multipliers have an advantage of reducing partial product stages and operate at moderate speed. The conventional booth multiplier uses separate registers to hold multiplier, multiplicand, product and partial product values. This paper is presented to improve the computation time of the Booth multiplier using Shift-Opt algorithm. The algorithm detects the sequence of zero recoding bit pairs and non-zero recoding bit pairs of the multiplier using the Bit Pair Detector (BPD). The output obtained from the Bit Pair Detector decides which operation to perform and hop between the functional blocks. In the proposed system, the multiplier value is assigned to the product register itself for effective hardware optimization. The design of the Bit Pair Detector is simple. The algorithm can be applicable to all the radix schemes (i.e. radix-2, radix- 4, radix- 8 and so on). Thus, it will greatly reduce the delay, increase the speed of the multiplier, provides better area-time performance and accuracy. An 8-bit multiplier is designed and coded in Verilog HDL, and further it is simulated using Xilinx ISE. It is shown that for the bit width of 8, our optimization has resulted in 9.6% improvement over the existing Modified Booth multiplier in terms of delay, depending upon the input set.**

***Key words*—Radix-4, Radix-8, Shift-Opt algorithm, Bit Pair Detector, FPGA and Xilinx.**

1. INTRODUCTION

Multiplication plays a pivotal role in the Digital Signal Processing such as filters design and in Discrete Fourier Transformation (DFT). In conventional time, the multiplier performs multiplication operation by adding and shifting in sequential manner. Multiplication uses two input numbers named multiplicand and multiplier. It is done by adding one integer (multiplicand) itself for a number of times by another number (multiplier) [8]. Multiplier considers parameters such as speed, power consumption, area space and accuracy for its effective design. Certain applications require high speed multipliers for the faster computation of products. Many multipliers were designed to meet the requirements of high speed, less area, low power and less complexity.

Of all the multipliers designed, Booth multiplier proved to play a major role in high speed and low power applications. The comparison of the performance parameters of various multipliers are discussed in the table 1.1 shown below. The table 1.1 shows that the Modified Booth multiplier consumes moderate power, operates at high speed, utilizes less area and is easy to implement.

Table1.1 Comparison of Multiplier [12]

Parameter	**Array Multiplier**	**Modified Booth Multiplier**	**Wallace-Tree Multiplier**	**Modified Booth-WallaceTree Multiplier**
Area	Medium	Small	Large	Small
Critical Delay	Medium	Fast	VeryFast	Fastest
Power Consumption	Large	Medium	Large	Medium
Complexity	Simple	Complex	More complex	More complex
Implementation	Easy	Medium	Difficult	Difficult

Booth multiplier was invented by Andrew Donald Booth in 1950. Booth multiplier has a greater advantage since it reduces the number of Partial Products (PP) that has been generated. The number of partial products generated depends upon the radix scheme used for Booth multiplier. Radix scheme specifies the number of multiplier bits taken at a time for the computation of a single partial product.

A basic Booth algorithm takes two multiplier bits at a time, which is called the Radix-2 scheme. This algorithm is modified by taking three multiplier bits at a time, so called the Modified Booth algorithm or the Radix-4 Booth algorithm. The next version of the Booth algorithm takes four bits of the multiplier at a time for the PP generation. This scheme is called the Radix-8 booth algorithm. The successive algorithms use five, six, seven bits which are called the Radix-16, Radix-32 and Radix-64 Booth algorithm respectively. Higher the radix scheme, lesser is the number of partial products generated. Though higher radix schemes offer high speed, they come with a great disadvantage that they are difficult to implement and are complex in nature. Hence, in this paper the lower radix schemes namely, the radix-4 Booth algorithm and the radix-8 Booth algorithm are mainly concentrated. In this paper, an efficient algorithm named Shift-Opt algorithm is proposed which speeds up the multiplication process. Thus, the multiplier can be used for high speed applications.

The rest of the paper is organized as follows. Chapter 2 discusses the previous works on the Booth multiplier. Chapter 3 presents the existing Radix-4 and Radix-8 Booth multiplier. The proposed system is explained in Chapter 4. Chapter 5 contains the simulation results, graphs and comparison table for the existing and proposed systems. The paper is concluded in Chapter 6.

2. MULTIPLIER OPTIMIZATIONS

Various optimizations have been made for the effective hardware implementation of multipliers. Radix-2 and Radix-4 Booth multiplier were studied and it has been concluded that in Radix-4 Booth multiplier, the delay and the gates were reduced and the processing speed has been increased [1]. Another study proposed the use of Approximate Redundant Binary multiplier in radix-8 Booth algorithm that gives three rows of the partial product which can be summed up and the circuit proved to be less complex and the propagation delay was decreased [2]. In order to improve the speed of the existing Modified Booth multiplier, SPST architecture has been proposed which provides a flexible arithmetic capacity and a tradeoff between output precision and power consumption [3]. Another system was proposed to increase the speed of the multiplier by switching between two functional blocks [5].

Many multipliers were compared for the performance and it is proved that the Radix-4 Booth multiplier is more efficient than other multipliers [7]. Another system proposed 16-bit approximate Radix-4 Booth multipliers with approximate factors of 12 and 14. The proposed R4ABM2 multiplier with an approximation factor of 14 has the most efficient design when considering both the power delay product and error metric NMED [14]. A review on different types of adders was made which showed that Carry Save Adder is suitable for adding three or four 8-bit numbers and Carry Look ahead adder is suitable for adding two 8-bit numbers [9]. A booth multiplier was proposed which reduces the dynamic and the static power by 45% and 65% respectively, when compared to a traditional 16-bit Booth multiplier [15].

3. EXISTING BOOTH MULTIPLIERS

Booth's algorithm is an efficient hardware implementation of a digital circuit that multiplies two binary numbers in two's complement notation. Booth multiplication is a fastest technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. The Booth multipliers are widely used in ASIC oriented products due to the higher computing speed and smaller area.

The entire Booth multiplication process is done in three steps. They are,

a) Partial Product Generation
b) Partial Product Accumulation
c) Addition of the Partial Products using the fast adders.

The generalized flow diagram of the Booth multiplier is shown in the figure 1 as follows.

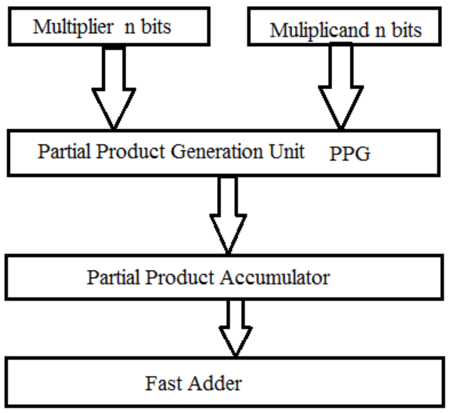

Multiplier n bits
Muliplicand n bits
Partial Product Generation Unit PPG
Partial Product Accumulator
Fast Adder

Figure 1 Generalized Booth Multiplier

A. Radix-4 Booth Multiplier

The Radix-4 Booth multiplier scheme is widely used in the existing systems. A single partial product is obtained by taking three bits of the multiplier and performing some operation on the multiplicand. The operation on the multiplicand is based on the multiplier input bits that are taken. The operation on the multiplicand for different input bits is shown in the table 2 below.

Table 2 Radix-4 Booth encoding scheme

BIT PAIR	RECODED BITS	OPERATION
000	0	Shift
001	+1	Add Md
010	+1	Add Md
011	+2	Add 2* Md
100	-2	Subtract 2* Md
101	-1	Subtract Md
110	-1	Subtract Md
111	0	Shift

In the Radix-4 Booth multiplier, the number of partial products generated is reduced by a factor of 2. For 'n' input bits, the number of PP stages is 'n/2'. For instance, if the input has 4 bits, then the number of PP generated is 2. In Radix-4, encoding operation of the multiplicands is based on multipliers bits. This technique implements smaller, faster circuits by recoding the numbers that are multiplied. It is standard technique used in chip design and provides significant improvement o ver long multiplication technique. A system proposed a speed improvement of 19% in Radix-4 booth multiplier as compared to the Radix-2 booth multiplier [11].

It has been proposed that in this multiplier, Bitwise Carry Select adder can be used in order to achieve speed maximization. The addition of partial products is carried out by using CSA (Carry Select Adder). Carry Select adder is used in order to reduce the carry propagation delay that is very high in case of ripple carry adder conventionally used by multipliers. Also, a novel hybrid adder, which has less delay and occupies less area is designed using the Carry Look-ahead adder [4]. The choice of the adders that is used for different digital applications was proposed [10].

B. Radix-8 Booth Multiplier

The Radix-8 booth multiplier takes 4 multiplier bits for the PP generation. The number of PP stages is reduced by a factor of 3. For 'n' input bits, the number of PP generated is 'n/2'. Thus, for 12 input bits, we have 4 stages. The different operations performed on the multiplicand based on the multiplier bits are shown in the table 3 as follows.

Table 3 Radix-8 Booth Encoding scheme

BIT PAIR	RECODED BITS	OPERATION
0000	0	Shift
0001	+1	Add Md
0010	+1	Add Md
0011	+2	Add 2* Md
0100	+2	Add 2* Md
0101	+3	Add 3* Md
0110	+3	Add 3* Md
0111	+4	Add 4* Md
1000	-4	Subtract 4* Md
1001	-3	Subtract 3* Md
1010	-3	Subtract 3* Md
1011	-2	Subtract 2* Md
1100	-2	Subtract 2* Md
1101	-1	Subtract 1* Md
1110	-1	Subtract 1* Md
1111	0	Shift

Speeding up the multiplication using Booth algorithm can be achieved by recoding the multiplier in a higher radix than 2. Higher radix recoding means greater number of multiplier bits are inspected and eliminated per cycle resulting in less number of cycles required to obtain the product. The higher order radix scheme such as radix-16, radix-32 and so on takes five bits, six bits at a time for the computation. But higher radix schemes are more complex and are difficult to implement. Certain modifications have been made to increase the speed of the Radix-8 Booth multiplier. In a system, an approximate 2-bit adder was proposed to implement the less significant section of a recoding adder for the generation of the triple multiplicand with no carry propagation delay [6].

It is evident from the Radix-4 and Radix-8 Booth Encoding schemes that certain input bits 000/111 from the table 2 and 0000/1111 from the table 3 performs only the shifting operation on the multiplicand. In existing multipliers, these inputs are computed as every other input for obtaining the partial products. The multipliers are not used effectively since it computes PP for the inputs that have no specific operation to perform on the multiplicand. It has been proposed that modifying certain modules of the multiplier can increase speed and reduce power [13]. Hence, this paper proposes an algorithm that eliminates the unnecessary computations that are being performed.

4. PROPOSED BOOTH MULTIPLIER

4.1 Generalized Concept

The proposed system is a modification to the existing multiplier which was discussed in the chapter 3. The proposed system considers the Radix-4 booth encoding scheme. In order to eliminate the unnecessary computations those are performed, a new algorithm known as the **Shift-Opt algorithm** has been proposed. In addition to it, our method uses the product register to hold the multiplier values for the computation of PP. After the entire computation, the final product is stored in the product register. The multiplier is in the last 'n' LSB of the '2n' product register. The generalized idea about the proposed system is provided in the figure 2 below.

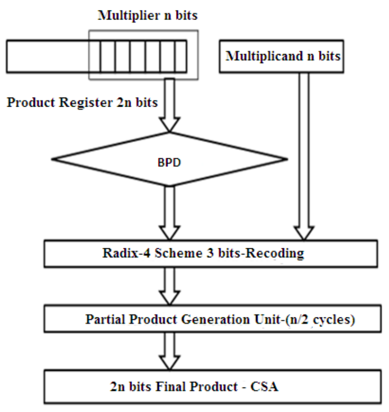

Multiplier n bits
Multiplicand n bits
Product Register 2n bits
BPD
Radix-4 Scheme 3 bits-Recoding
Partial Product Generation Unit-(n/2 cycles)
2n bits Final Product - CSA

Figure 2 Generalized Proposed System

From the figure 2, it is understood that the multiplier values are assigned to the 'n' LSB bits of the '2n' product register. For the Radix-4 Booth encoding scheme, three bits are taken at a time. The three bits are given to the Bit Pair Detector (BPD) to perform a specific function. The final product is obtained by adding the PP using the Carry Select Adder.

4.2 Shift-Opt Algorithm

To speed up the computation process, Shift-Opt algorithm has been introduced. A Bit Pair Detector is used to implement this algorithm. This algorithm is used to hop between the functional blocks. To implement this algorithm, two different functional blocks are considered. They are the Shifter Block and the Operational Block. Shifter block is used for the shifting process and the Operational Block (Op-Block) is used for the computation of Partial Products.

The last 'm' multiplier bits are checked for the presence of consecutive zeros or consecutive ones using this algorithm. The value of 'm' depends upon the type of the radix scheme used. To elaborate, the radix-2 scheme has 'm' value as 2; the radix-4 scheme takes 'm' value to be 3; the radix-8 scheme will have 'm' value to be 4 and so on. The Shift-Opt algorithm is explained below.

Algorithm

Md and Mr are the multiplicand and multiplier bits respectively and P is the product. Mr and Md are of 'n' bits each and P is of '2n' bits. The algorithm is used to identify the 'm' bits and perform the corresponding operation.

STEP 1: Obtain Inputs. *//Get n bits of Multiplier (Mr) and Multiplicand (Md)*
STEP 2: Set count = 0. *//Count is set to determine the end of the program*
STEP 3: Set the LSB of '2n' Product Register (P) as zero. *//P[0] = 0*
STEP 4: Assign 'n' LSB bits Product Register as Multiplier. *//P[1]P[2]....P[n] = Mr*
STEP 5: Fetch 'm' bits of the Product Register to Bit Pair Decoder (BPD).
STEP 6: Detect BPD output. *//BPD Output can be logic 0 or logic 1*
STEP 7: Enable Shifter block for logic 0 or the Operational block for logic 1.
STEP 8: End.

This algorithm greatly reduces the time required for the computation of the final product since it leaves the unwanted operations that are being performed in the existing multipliers. The Bit Pair Detector circuit is specially designed to implement this algorithm.

4.3 Bit Pair Detector

A BPD consists of the basic logic gates namely the AND gate, the NOT gate, the XNOR gate andthe OR gates. The structure of the Bit Pair Detector is shown in the figure 3 below.

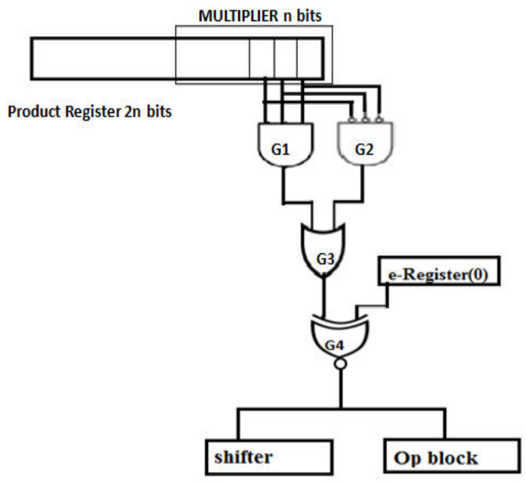


Figure 3 Bit Pair Detector circuit

Specifications G1- AND gate G2-NAND gate G3-OR gate G4-XNOR gate e- Register having value 0

Let us assume the outputs of the AND, NAND, OR and XNOR gates as x1, x2, x 3 and Y respectively. It can be noted that the final output of the BPD (Y) can be either 0 or 1. It is suitable to say that the output Y can be 0 only for the input bits 000 or 111 (for the Radix-4 scheme). Thus, when the 'm' bits of the multiplier has consecutive zeros or consecutive ones, the output Y is 0. Thus, the shifter block is enabled for Y=0. The other inputs (001,010,011,100,101,110) gives the output Y=1. This enables the operational block (Op block).

4.4 System Flowchart

The entire system is considered as two different functional blocks. The Shifter block encloses the operations that are to be performed for the consecutive zeros or consecutive ones inputs. The Op block encloses the operations that are to be performed for the other inputs. Hence, the flowcharts for the Shifter block and the Operational block are shown separately.

The figure 4 shows the flow chart of the shifter functional block. Initially, m bits are taken at a time and given to the Bit Pair Detector. Here, radix-4 scheme is considered thus the 'm' value is taken as three bits. The counter value is initiated as 0. When the circuit detects the sequence of consecutive zeroes or ones, then the output obtained from the Bit Pair Detector is logic 0(Y=0). The shifter block is enabled and then multiplicand value is left shifted by 2 bits in the multiplicand register (Shifting of the multiplicand bits also depend on the radix scheme used. For radix-4, 2 bits are shifted; for radix-8, 3 bits are shifted and so on). Then the counter is incremented. The counter is checked for the count value, which must be less than or equal to n/2. (This is because for Radix-4 scheme, only n/2 PP is required). If it is less than n/2, the next set of inputs will be fetched and the same process is done. If the counter value is equal to n/2, then the product register will be cleared and the final product will be stored.

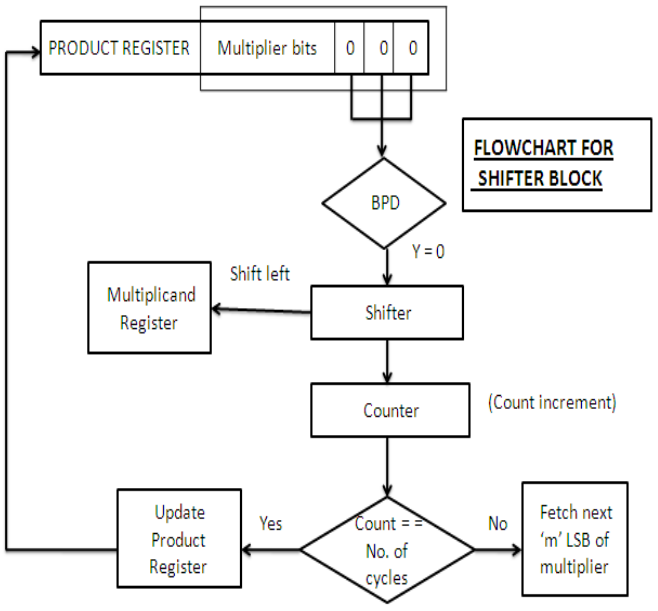


Figure 4 Flowchart for Shifter Block

The flowchart for the Operational block is shown in the figure 5 below. Initially, m bits are taken at a time and given to the Bit Pair Detector. When the circuit detects the sequence other than consecutive zeroes or ones, then the output obtained from the Bit Pair Detector is logic 1(Y=1). The operational block is enabled where the recoded multiplier bits are obtained. (For example, for input 001, the recoded multiplier value is +1). Then the PP is computed using the PP generation unit. This unit contains adders, shifters and PP register of 2n bits for holding the intermediate PP. After every stage of PP computation, the counter is incremented. If the counter value is equal to n/2, then the product register is cleared and the final product is stored in the product register. If the counter value is less than n/2, then the multiplicand is shifted by 2 bits (For Radix-4 scheme) and the next set of inputs is fetched by the BPD and the same process continues.

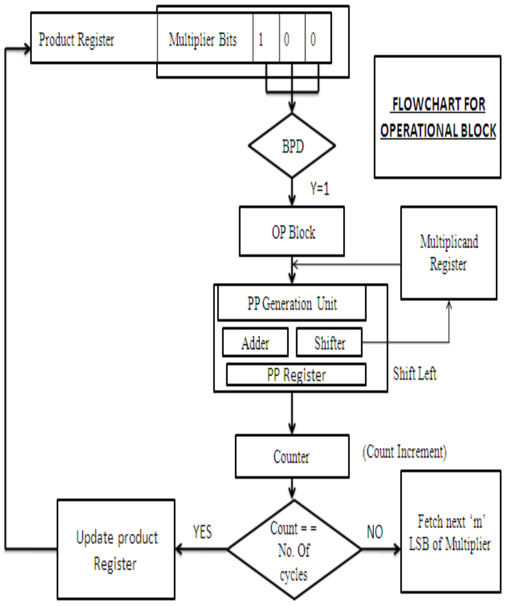


Figure 5 Flowchart for Operational Block

5. RESULTS AND DISCUSSUION

The results for the proposed systemare obtained by coding in ModelSim and by implementing in Xilinx. The outputs for the individual modules are simulated in ModelSim. Firstly, the output for the Bit Pair Detector is obtained for various inputs. This is shown in the figure 6.

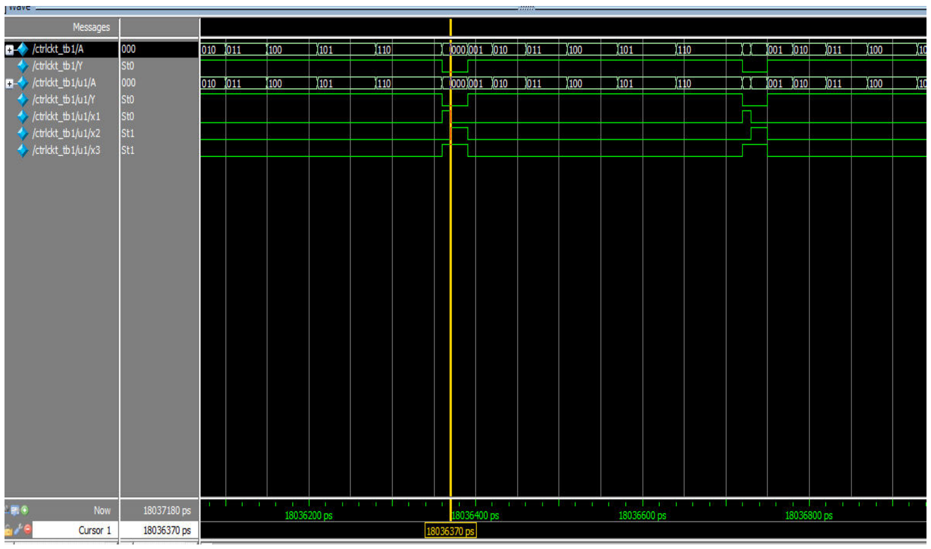


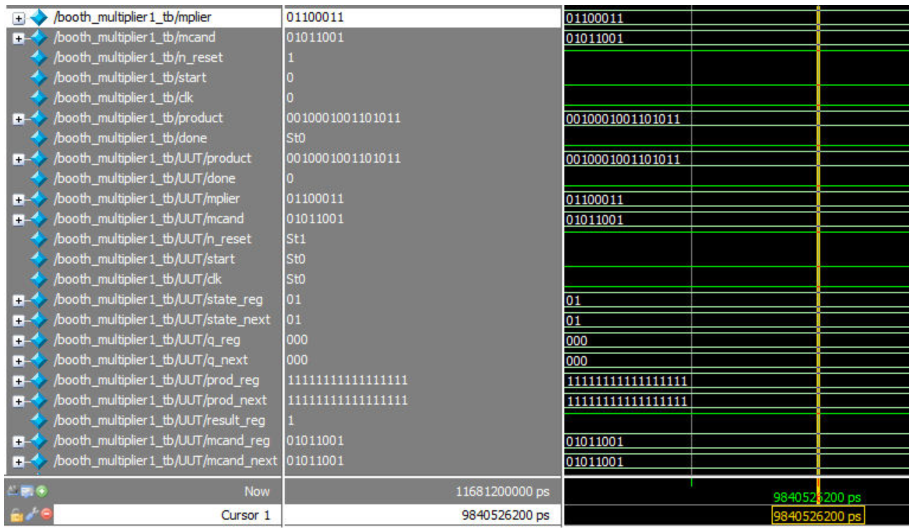
Figure 6 Simulation output for the BPD for different multiplier inputs (Radix-4 Booth algorithm)

It can be seen from the output that only for the input bits of 000 and 111, the output Y is 0. For the other sets of inputs, the output Y remains 1. Thus, the Shifter block is enabled for Y=0 and the Operational block is enabled for Y=1.

The output simulated with the given inputs using ModelSim is shown in the figures 7 and 8. The figure 7 shows the values of inputs and outputs, along with the other waveforms. It can be noted that in the figure 7, for the inputs 01100011 (decimal equivalent is 99) and

01011001 (decimal equivalent is 89), the output obtained is 0010001001101011 (decimal equivalent is 8811), which is the product of the two given inputs. Also, the figure 8 shows the values of the three LSBs of the multiplier. These are given to the BPD circuit and output is obtained.

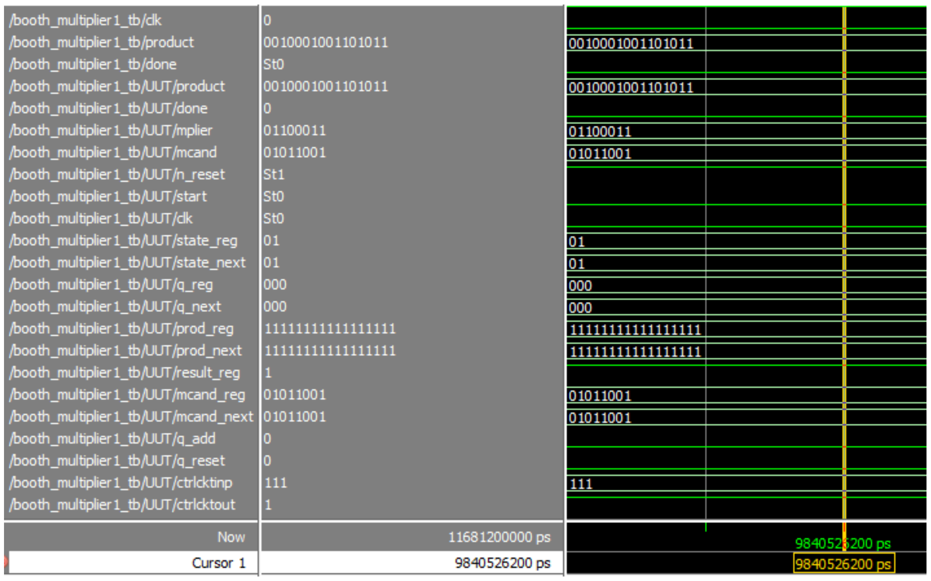
From the figure 8, the parameter ‘ctrlcktinp’ denotes the LSB of the multiplier and ‘ctrlcktout’ represents the output of the BPD circuit. Other inputs such as clk, n_reset and done are given to determine the clock input, start of computation and the end of the program respectively. In order to differentiate the present state and the next state, FSM is used in the code. The FSM is useful in determining the IDLE and the BUSY states of the program. During the IDLE state, no operation takes place. The operation takes place only at the BUSY state. The entire process takes less time for computation of the partial products when compared with the existing system.



```
/booth_multiplier1_tb/mplier              01100011            01100011
/booth_multiplier1_tb/mcand               01011001            01011001
/booth_multiplier1_tb/n_reset             1
/booth_multiplier1_tb/start               0
/booth_multiplier1_tb/clk                 0
/booth_multiplier1_tb/product             0010001001101011    0010001001101011
/booth_multiplier1_tb/done                St0
/booth_multiplier1_tb/UUT/product         0010001001101011    0010001001101011
/booth_multiplier1_tb/UUT/done            0
/booth_multiplier1_tb/UUT/mplier          01100011            01100011
/booth_multiplier1_tb/UUT/mcand           01011001            01011001
/booth_multiplier1_tb/UUT/n_reset         St1
/booth_multiplier1_tb/UUT/start           St0
/booth_multiplier1_tb/UUT/clk             St0
/booth_multiplier1_tb/UUT/state_reg       01                  01
/booth_multiplier1_tb/UUT/state_next      01                  01
/booth_multiplier1_tb/UUT/q_reg           000                 000
/booth_multiplier1_tb/UUT/q_next          000                 000
/booth_multiplier1_tb/UUT/prod_reg        1111111111111111    1111111111111111
/booth_multiplier1_tb/UUT/prod_next       1111111111111111    1111111111111111
/booth_multiplier1_tb/UUT/result_reg      1
/booth_multiplier1_tb/UUT/mcand_reg       01011001            01011001
/booth_multiplier1_tb/UUT/mcand_next      01011001            01011001
Now                                       11681200000 ps      9840526200 ps
Cursor 1                                  9840526200 ps       9840526200 ps
```

Figure 7 Simulation output of the proposed multiplier with clock, reset signals

The following figure 8 shows the output of the proposed system along with the control circuit.



```
/booth_multiplier1_tb/clk                 0
/booth_multiplier1_tb/product             0010001001101011    0010001001101011
/booth_multiplier1_tb/done                St0
/booth_multiplier1_tb/UUT/product         0010001001101011    0010001001101011
/booth_multiplier1_tb/UUT/done            0
/booth_multiplier1_tb/UUT/mplier          01100011            01100011
/booth_multiplier1_tb/UUT/mcand           01011001            01011001
/booth_multiplier1_tb/UUT/n_reset         St1
/booth_multiplier1_tb/UUT/start           St0
/booth_multiplier1_tb/UUT/clk             St0
/booth_multiplier1_tb/UUT/state_reg       01                  01
/booth_multiplier1_tb/UUT/state_next      01                  01
/booth_multiplier1_tb/UUT/q_reg           000                 000
/booth_multiplier1_tb/UUT/q_next          000                 000
/booth_multiplier1_tb/UUT/prod_reg        1111111111111111    1111111111111111
/booth_multiplier1_tb/UUT/prod_next       1111111111111111    1111111111111111
/booth_multiplier1_tb/UUT/result_reg      1
/booth_multiplier1_tb/UUT/mcand_reg       01011001            01011001
/booth_multiplier1_tb/UUT/mcand_next      01011001            01011001
/booth_multiplier1_tb/UUT/q_add           0
/booth_multiplier1_tb/UUT/q_reset         0
/booth_multiplier1_tb/UUT/ctrlcktinp      111                 111
/booth_multiplier1_tb/UUT/ctrlcktout      1
Now                                       11681200000 ps      9840526200 ps
Cursor 1                                  9840526200 ps       9840526200 ps
```

Figure 8 Simulation output of the proposed multiplier with the BPD inputs and outputs

After obtaining the simulation results from the ModelSim software, the code has been implemented in the Xilinx software to determine the performance of the multipliers. The RTL schematic for the proposed system is shown in the figure 9. The various inputs are mcand (multiplicand), mplier (multiplier), clk (clock), n_reset (reset) and start (start of execution of the program). The outputs are product and done (determines the end of the program). The detailed RTL schematic is shown in the figure 10.

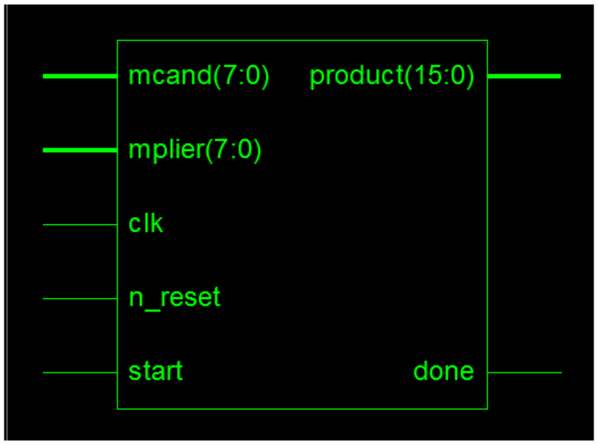


Figure 9 RTL schematic of the proposed model

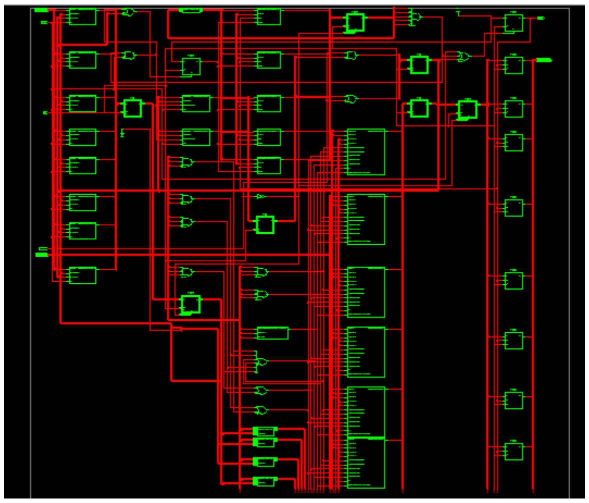


Figure 10 Detailed RTL schematic of the roposed model

The performance of the proposed multiplier which uses the Shift-Opt algorithm has been compared with the existing systemas shown in the table below. The parameters determining the performance of the multipliers such as logic delay, route delay, total delay and the memory used has been listed.

Table 1 Comparison of the existing and the proposed systems

PARAMETERS	EXISTING SYSTEM	PROPOSED SYSTEM
Total delay	7.169ns	7.073ns
Logic delay	5.194ns (72.4%)	1.975ns (69.1%)
Route delay	2.186ns	1.975ns

	(30.9%)	(27.6%)
Memory	158912kb	159936kb

6. CONCLUSION

Multipliers are required in almost every application. It is a part of every computer. Designing the multipliers with high speed is a great area of study. Thus, various methods have been proposed for the multipliers to operate at high speed and consume low power. Our system proves to provide a high speed multiplier for certain inputs. If the multiplier has many consecutive zeros or consecutive ones (1000, 1000000, 111111, 11111111, etc.,) in it, then the performance can be improved greatly. But if the multiplier has alternate zeros or ones (010, 1010, 101010, 1010101010, etc.,) in it, the performance cannot be improved. This forms the worst case of our system. Our system achieved higher speed and consumed less power with improved area-time performance when compared with the existing multipliers. The multiplier that we proposed improves the delay performance of the existing system by 9.6%. Thus, it is faster than the existing multipliers.

REFERENCES

[1] Ashwini K. Dhumal1 , Prof. S.S. Shirgan2 (2016)'Comparison between Radix-2 and Radix -4 based on Booth Algorithm 'International Journal of Advanced Research in Computer and Communication Engineering(IJARCCE), Vol. 5, Issue 12.

[2] Chandrika Sowmini D., Lavanya A., Jagadeesh Sai D., Raja C., Aditya M. (2019) "Design and Analysis of Multipliers using Radix-8 Booth Encoding Technique for Low Power and Area Consumption:"International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-9, Issue-2.

[3] Chinababu Vanama, Sumalatha M. (2013) 'Implementation of High Speed Modified Booth Multiplier and Accumulator (Mac) Unit', IOSR Journal of Electronics and Communication Engineering, Volume 8, Issue 5, PP 17-25.

[4] Divya Govekar and Ameeta Amonkar (2017) "Design and Implementation of High Speed Modified Booth Multiplier using Hybrid Adder IEEE TRANSACTIONS ON COMPUTERS Volume: 66, Issue: 8.

[5] Duncan J. M. Moss, David Boland, and Philip H. W. Leong.(2019) "A Two-Speed, Radix-4, Serial–Parallel Multiplier" IEEE TRANSACTIONS on Very Large Scale Integration(VLSI) Systems,Vol.27,No.4.

[6] Honglan Jiang, Jie Han, Fie Qiao (2015)"Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation:" IEEE TRANSACTIONS ON COMPUTERS.

[7] Jasbir Kaur, Sumit Kumar (2016)'Performance Comparison of Higher Radix Booth Multiplier using 45nm technology' International Journal of Innovative Research in Science, Engineering and Technology, Vol. 5, Issue .

[8] Niharika (2015) "Comparative Analysis of Booth Multiplier using Radix-2 and Radix-4 Technique using VHDL" International Journal of Research and Development in Applied Science and Engineering (IJRDASE)

[9] Rashmi Rahul Kulkarni (2015) "Comparison among Different Adders:" IOSR Journal of VLSI and Signal Processing, Volume 5, Issue 6, pp.01-06.

[10] Sakshi Rajput ,Priya Sharma , Gitanjali and Garima (2013) "High Speed and Reduced Power – Radix-2 Booth Multiplier IJCEM International Journal of Computational Engineering & Management (IJCEM), Vol. 16 Issue 2.

[11] Sakthivel B, M.Nivedhaa, A.Manimegalai, M.Vignesh and N.Satheesh(2018)."Performance Comparison of Radix-2 and Radix-4 by Booth Multiplier" International Journal of Recent Trends in Engineering & Research (IJRTER) Conference on Electronics, Information and Communication Systems (CELICS'18)

[12] Tarun Kumar B.V.N, Aravind Chitiprolu, Hemanth Kumar Reddy G, Ms.SonaliAgrawal(2020) "Analysis of High Speed Radix-4 Serial Multiplier" 3rd International Conference on Smart Systems and Inventive Technology (ICSSIT)

[13] Venkata Dharani.B, Sneha M. Joseph, Sanjeev Kumar and Durgesh Nandan (2020) "Booth Multiplier: The Systematic Study" 3rd In and Cyber Physical Engineering (ICCCE)

[14] Wen-Chang Yeh and Chein-Wei Jen(2000) "High-Speed Booth Encoded Parallel Multiplier Design", IEEE TRANSACTIONS on Computers.

[15] Yen-Jen Changi1, Yu-Cheng Cheng1, Shao-Chiliao2 and Chun-Huo Hsiao (2020) "A Low Power Radix-4 Booth Multiplier With Pre-Encoded Mechanism:" IEEE Access.