Design of High Speed Efficient Modified 16-Bit Booth Multiplier using VHDL

Miss. Sayali Gaidhane[1] Prof. R. D. Kadam[2]
[1]Student [2]Assistant Professor
[1,2]Department of Electronic & Telecommunication Engineering
[1,2]BDCOE, Sewagram, Maharashtra, India

Abstract*—The Main objective of the proposed work is completely based on enhancing speed performance multiplication process using modified Booth algorithm. In this paper we have designed 16x16 bit modified booth multiplier. The proposed modified Booth multiplier has been verified and advantages over the existing multipliers. This proposed multiplier provides less delay 22.121 nsec. Many researchers had been worked and presented the modified booth multiplier with optimized delay. In this paper, it has been shown that the proposed 16 bit modified booth multiplier provides less delay in comparison with those existing research papers. Coding of the research work has been written in VHDL language whereas synthesize and simulation has been done in Xilinx ISE 14.5 simulator software. Also partial products which are generated are less as compared to conventional multiplier. No. of logic blocks required for fast multiplication process has been reduced in terms of no. of slices in comparison with previous ones.*

Keywords—Booth Multiplier, XILINX ISE, VHDL

I. INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, Microprocessors, digital signal processors, etc. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, a whole spectrum of multipliers with different area-speed constraints has been designed with fully parallel end of the spectrum and fully serial multipliers at the other end. The speed of multiplication operation is increased using several schemes such as Wallace-tree, Booth and CSA multipliers. The array multiplier is the simplest architecture and is most suitable for VLSI implementation because of its high degree of regularity. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system. The multiplier circuit is a core component of most of the present day digital signal processors. Therefore, the demand for multiplier-performance improvement is increasing. Multipliers are a major source of power dissipation. Reducing the power dissipation of multipliers is key to satisfying the overall power budget of various digital circuits and systems. For high speed FFT processor, multiplier has a major role, as it is one of the processing elements of FFT. To meet better performance of FFT, there should be high speed multiplier.

In most of the DSP applications, multiplier is the main of the system. The speed of that system is mainly depends upon the multiplier. If the multiplier is efficient for performing fast operations then the overall speed of the design automatically increases. So, there is need of high speed multiplier in every system which consists of multiplier. In some DSP applications, multiplications are carried out such as in FFT processor. Hence, if we replace the multipliers used by that system by most efficient multiplier based on Vedic mathematics then the speed of operation of that system will increase and the system will become more efficient. Hence, designing of Modified Booth Multiplier is a necessary choice.

II. BOOTH ALGORITHM

Booth's algorithm is a well-known method for 2's complement multiplication. It speeds up the process by analyzing multiple bits of multiplier at a time. This widely used scheme for two's complement multiplication was designed by Andrew D. Booth in 1951. Booth algorithm is an elegant way for this type of multiplication which treats both positive and negative operands uniformly. It allows n bit multiplication to be done using fewer than n additions or subtractions, thereby making possible faster multiplication. Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.

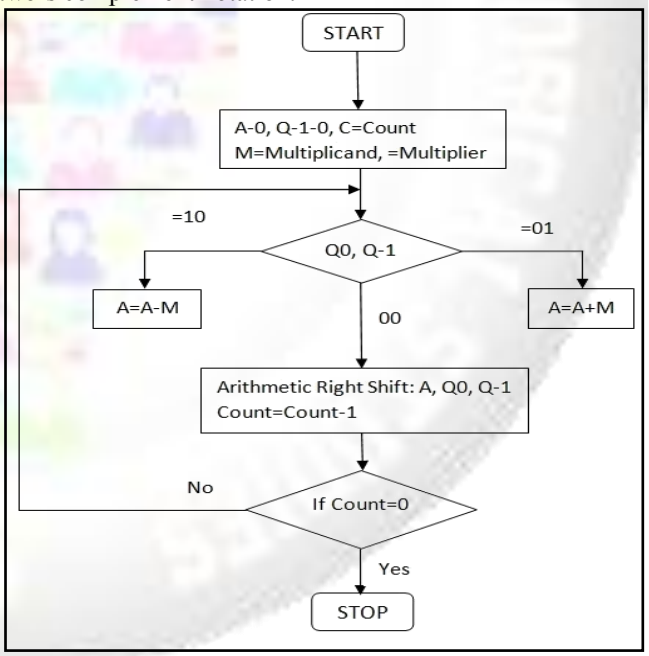

Fig. 1: Flowchart of Booth's Algorithm for Binary Multiplication

There are 2 methods that you should know before attempting Booth's algorithm. Right shift circulant and right-shift arithmetic.

A. Right-Shift Circulant(RSC)

RSC is simply shifting the bit, in a binary string, to the right 1 bit position and take the last bit in the string and append it to the beginning of the string.

Example:
10110
After right-shift circulant now equals – 01011

B. Right-shift arithmetic (RSA)

RSAis where you add 2 binary number together and shift the result to the right 1 bit position.
Example:
0100
+0110
Result = 1010
Now shift all bits right and put the first bit of the result at the beginning of the new
String:
Result 1010
Shift 11010

According to Booth's multiplication algorithm among the two input binary numbers the one with minimum number of bit changes is considered as multiplier and the other as a multiplicand in order to reduce the time taken for calculating the multiplication product.

The steps for performing booth multiplication are as follows:

- Let the multiplicand be 'B' and multiplier be 'Q'.
- Assume initially value of 'A' and 'Q-1' is zero.
- The main step is to check last two bits.
- There will be iterations according to the number of multiplier.
- For example, if the multiplier is of 2-bit then 2 Iterations will be done, for 4-bit multiplier 4 iterations are done, and so on.
- Now, the algorithm starts, first the last two digits are checked and if the two bits are "00" or "11" the only Arithmetic Right Shift is done.
- And if the last two bits are "01", then A is added with B, and result is stored into A.
- If the last two bits are "10", then A is subtracted from B, and result is stored into A.
- Finally, the result obtained is coded in binary form which gives the desired output.
- In this way multiplication of any two numbers is performed using booth algorithm.

Example:
Multiply 14 * -5 using 5-bit numbers.
14 in binary: 01110
-14 in binary: 10010
5 in binary: 00101
-5 in binary: 11011
Result:70 in binary: 11101 11010 (10-bit result).

C. Carry save Adder

There are many cases where it is desired to add more than two numbers together. The straightforward way of adding together m numbers (all n bits wide) is to add the first two, then add that sum to the next, and so on. This require total of m − 1 additions.

Using carry save addition, the delay can be reduced further still. The idea is to take 3 numbers that we want to add together, x + y + z, and convert it into 2 numbers c + s such that x + y + z = c + s. In carry save addition, we refrain from directly passing on the carry information until the very last step. To add three numbers by hand, we typically align the three operands, and then proceed column by column in the same fashion that we perform addition with two numbers. The three digits in a row are added, and any overflow goes into the next column. Observe that when there is some non-zero carry, we are really adding four digits (the digits of x,y and z, plus the carry).

The important point is that c and s can be computed independently, and furthermore, each ci (and si) can be computed independently from all of the other c's (and s's). This achieves our original goal of converting three numbers that we wish to add into two numbers that add up to the same sum, and in time.
The same concept can be applied to binary numbers.
As an example:

```
x:    1 0 0 1 1
y:    1 1 0 0 1
z: +  0 1 0 1 1
-----------------------
s:    0 0 0 0 1
c: + 1 1 0 1 1
-----------------------------
Sum: 1 1 0 1 1 1
```

The important point is that c and s can be computed independently, and furthermore, each ci (and si) can be computed independently from all of the other c's (and s's). This achieves our original goal of converting three numbers that we wish to add into two numbers that add up to the same sum. What does the circuit to compute s and c look like? It is actually identical to the full adder, but with some of the signals renamed.

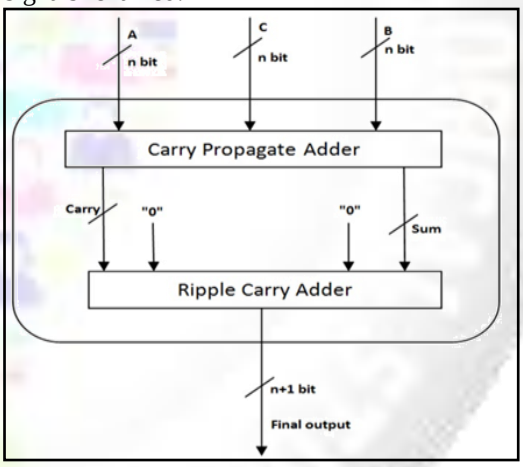

Fig. 2: Block Diagram of CSA

Figure 2 shows a carry save adder. A carry save adder simply is a combination of carry propagate adder and ripple carry adder.

III. EXPERIMENTAL RESULTS

16-Bit Modified Booth Multiplier
RTL View

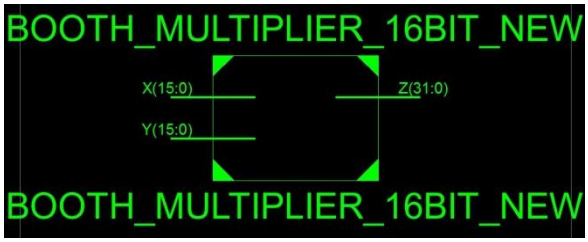

Fig. 3: RTL View of 16-Bit Modified Booth Multiplier

Figure 3 Shows the RTL (Register Transfer Logic) View of 16-Bit Modified Booth Multiplier.

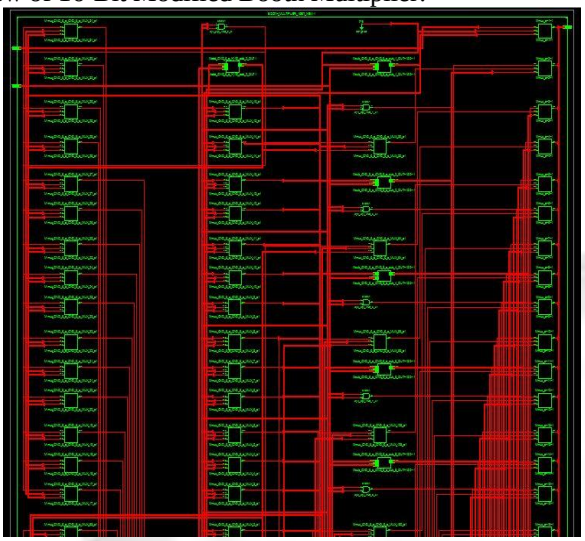

Fig. 4: Detailed RTL View of 16-Bit Modified Booth Multiplier

Figure 4 Shows the detailed RTL (Register Transfer Logic) View of 16-Bit Modified Booth Multiplier.

A. Simulation Result

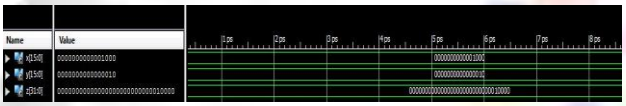

Fig. 5: Simulation Result of 16-Bit Modified Booth Multiplier in Binary form

Figure 5 Shows the detailed Simulation Result of 16-Bit Modified Booth Multiplier which gives multiplication of X=8, Y=2 i.e. Z=16 in Binary form.

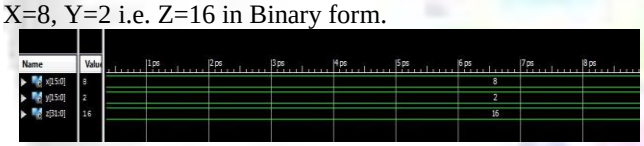

Fig. 6: Simulation Result of 16-Bit Modified Booth Multiplier in Decimal form

Figure 6 Shows the detailed Simulation Result of 16-Bit Modified Booth Multiplier which gives multiplication of X=8, Y=2 i.e. Z=16 in Decimal form.

IV. PROPOSED WORK

This research work targets the design of design of High Speed Efficient Modified 64-Bit Booth Multiplier Using VHDL. It includes; design of high speed 64-bit multiplier, Multiplier will be design using CSA, modified booth algorithm and pipelining.

Multiplication is an important fundamental function in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform (FFT), filtering and in microprocessors in its arithmetic and logic unit. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier. The following are the main objective of this project.

- To reduce delay.
- To reduce power.
- To increase the speed

The Proposed work is to design of High Speed Efficient Modified 64-Bit Booth Multiplier Using VHDL. High Speed Efficient Modified 64-Bit Booth Multiplier can be composed with Booth Encoder, Booth Decoder, Booth Multiplier and Carry Save Adder. Following figure shows a proposed block diagram of High Speed Efficient Modified 64-Bit Booth Multiplier Using VHDL.

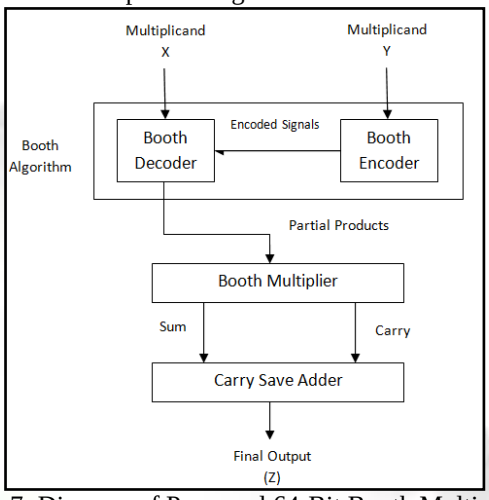

Fig. 7: Diagram of Proposed 64-Bit Booth Multiplier

V. CONCLUSION

In this paper we have implemented High Speed Efficient Modified 16-Bit Booth Multiplier Using VHDL. Design, synthesis and simulation of Modified 16-Bit Booth Multiplier will be done using XILINX ISE 14.5. Coding of the proposed design will be done in VHDL. The combinational path delay obtained is 22.121 nsec. Future work is likely to achieve the design, synthesis and simulation of Booth Multiplier, CSA adder and finally the High Speed Efficient Modified 64-Bit Booth Multiplier Using VHDL. Therefore, Modified 64-Bit Booth Multiplier using VHDL is the aim of this research work.

REFERENCES

[1] Nyamatulla Patel, Vidyashri M. Bastawadi, Suparna R. Daddimani, "Design of High Speed Hardware Efficient Modified Booth Multiplier Using HDL", Journal of Advances in Science and Technology Vol. 14, Issue No. 1, June-2017, ISSN 2230-9659.

[2] Shaik Meerabi, Krishna Prasad Satamraju, "Design and Implementation of 64-Bit Multiplier Using CLAA and CSLA", International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013).

[3] Sweta Khatri, GhanshyamJangid, "FPGA Implementation of 64-bit fast multiplier using barrel shifter", International Journal for Research In Applied Science And Engineering TechnoloGy (I JRAS ET), Vol. 2 Issue VII, July 2014.

[4] Deepthi, Rani, Manasa, "Performance Analysis of a 64-bit signed Multiplier with a Carry Select Adder Using VHDL", IJCSNS International Journal of Computer Science and Network Security, VOL.15 No.11, November 2015.

[5] EPPILI JAYA, "POWER, AREA AND DELAY COMPARISION OF DIFFERENT MULTIPLIERS",

International Journal of Science, Engineering and Technology Research (IJSETR) Volume 5, Issue 6, June 2016.

[6] Bai A. Akahori, M. Tanaka, A. Sekiya, A. Fujimaki, and H.Hayakawa (1996). "Design and demonstration of SFQ pipelined multiplier," IEEE Trans. Appl. Supercond, vol. 13, no. 2, pp. 559–562.

[7] Cherkauer B. and Friedman E. (1996). "A Hybrid Radix-4/Radix-8 Low Power, High SpeedMultiplier Architecture for Wide Bit Widths", In IEEE International Symposium on Circuits and Systems, volume 4, pages 53–56.

[8] Da Huang, AfsanehNassery (2002). "Encoding Radix 4 8 bit multipliers", final project report, pp. 5- 6.

[9] David Villager, VojinGoklobdzija (2000). "Analysis of booth encoding in parallel multipliers using compressor for reduction of partial products", University of California, pp. 1-2.

[10] Hsin-Lei Lin, Robert C., Chang, Ming-Tsai Chan (2003), "Design of a Novel Radix-4 Booth Multiplier", pp. 14-17.

[11] Miss. SayaliGaidhane, Prof. R. D. Kadam, "Survey of High Speed Efficient Modified 64-Bit Booth Multiplier using VHDL", International Journal for Research in Technological Studies| Vol. 5, Issue 3, February 2018.