Design of High Speed Efficient Modified 64-Bit Booth Multiplier using VHDL

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

Abstract—*This paper targets the Design, Synthesis and Simulation of 64-bit Modified Booth Multiplier using VHDL. Further we have introduced CSA (Carry save Adder) into the proposed design to increase the speed and performance of the proposed design. CSA is one way of improving the overall processing performance of a multiplier. Here, CSA with modified techniques are used to increase the speed of the Booth Multiplier; also it can perform fast operations as compared to normal multiplier. Design, synthesis and simulation of 64-bit Modified Booth Multiplier have been done using XILINX ISE 14.5 software. Coding of the proposed design has been done in VHDL (Very high Speed Integrated Circuit Hardware Description Language). After taking synthesis of the design it is found that 64-bit booth multiplication can be done in 7.361ns time and 135.847 MHz frequency with only 143mW power consumption*.

Keywords—*Modified Booth Multiplier, XILINX ISE, VHDL*

I. INTRODUCTION

Multipliers are most commonly used in various electronic applications e.g. Digital signal processing in which multipliers are used to perform various algorithms like FIR, IIR etc. Earlier, the major challenge for VLSI designer was to reduce area of chip by using efficient optimization techniques to satisfy MOORE'S law. Then the next phase is to increase the speed of operation to achieve fast calculations like, in today's microprocessors millions of instructions are performed per second. Speed of operation is one of the major constraints in designing DSP processors and today's general-purpose processors. However area and speed are two conflicting constraints. So improving speed results always in larger areas. Now, as most of today's commercial electronic products are portable like Mobile, Laptops etc. that require more battery back-up. Therefore, lot of research is going on to reduce power consumption. So, in this paper it is tried to find out the best solution to achieve low power consumption, less area required and high speed for multiplier operation

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, design of FFT system using Vedic mathematics is a necessary choice.

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.

II. MODIFIED BOOTH MULTIPLIER

One of the many solutions of realizing high speed multipliers is enhancing parallelism which helps in decreasing the number of subsequent calculation levels. The original version of Booth algorithm (Radix-2) had two particular drawbacks. They were: The number of add-subtract operations and shift operations become variable and causes inconvenience in designing parallel multipliers. The algorithm becomes inefficient when there are isolated 1's. These problems are overwhelmed by using modified Radix4 Booth algorithm which scans strings of three bits using the algorithm given below: 1) Lengthen the sign bit 1 position if necessary to ensure that n is even. 2) Add a 0 to the right of the LSB of the multiplier. 3) Corresponding to the value of each vector, each Partial Product will be 0, +M, -M, +2M or -2M.

The negative values of M are made by taking its 2's complement. The multiplication of M is done by shifting M by one bit to the left (in case it's multiplied with 2). Thus, in any case, in designing an n-bit parallel multiplier, only n/2 partial products are generated.

i+1	*I*	*i-1*	*add*
0	0	0	0*M
0	0	1	1*M
0	1	0	1*M
0	1	1	2*M
1	0	0	-2*M
1	0	1	-1*M
1	1	0	-1*M
1	1	1	0*M

Table 1: Modified Booth's Recoding Table

A multiplier generator that creates a smaller number of partial products will allow the partial product summation to be efficient and use less hardware. The simple multiplication generator can be extended to reduce the number of partial products by grouping the bits of the multiplier into pairs, and selecting the partial products from the set of 0, M, 2M or their complements, where M is the multiplicand. This reduces the number of partial products, by a factor two but also generates some extra-bits for the sign extension and the 2's complementation. All partial products set can be produced using simple shifting and complementing. The multiplier is partitioned into overlapping groups of 3 bits, and each group is decoded to select a single partial product as per the selection table 3.2 shown below. Each partial product is shifted 2 bit positions with respect to its neighbors. The number of partial products have been reduced to half of total number of multiplier bits. In general the three will be n/2 products, where n is the operand length. The multiply by 2 can be obtained by a simple left shift of the multiplicand and negative of number obtained from its two's complement form.

III. CARRY SAVE ADDER

The carry-save adder reduces the addition of 3 numbers to the addition of 2 numbers. The propagation delay is 3 gates regardless of the number of bits. The carry-save unit consists of n full adders, each of which computes a single sum and carries bit based solely on the corresponding bits of the three input numbers. The entire sum can then be computed by shifting the carry sequence left by one place and appending a 0 to the front (most significant bit) of the partial sum sequence and adding this sequence with RCA produces the resulting n + 1-bit value. This process can be continued indefinitely, adding an input for each stage of full adders, without any intermediate carry propagation. These stages can be arranged in a binary tree structure, with cumulative delay logarithmic in the number of inputs to be added, and invariant of the number of bits per input. The main application of carry save algorithm is, well known for multiplier architecture is used for efficient CMOS implementation of much wider variety of algorithms for high speed digital signal processing .CSA applied in the partial product line of array multipliers will speed up the carry propagation in the array. The design schematic of Carry Save Adder is shown in Figure.

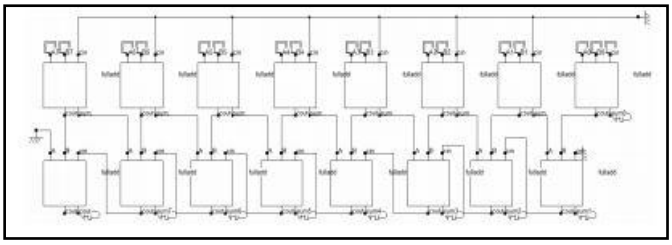

Fig. 1: Schematic of Carry save Adder

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 requires a total of m − 1 additions, for a total gate delay of O(m lg n) (assuming lookahead carry adders). Instead, a tree of adders can be formed, taking only O(lg m · lg n) gate delays. 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, and do this in O(1) time. The reason why addition cannot be performed in O(1) time is because the carry information must be propagated. In carry save addition, we refrain from directly passing on the carry information until the very last step. We will first illustrate the general concept with a base 10 example. 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).

IV. PROPOSED 64-BIT MODIFIED BOOTH MULTIPLIER

The block diagram of 64-Bit Modified Booth Multiplier is shown in the figure below.

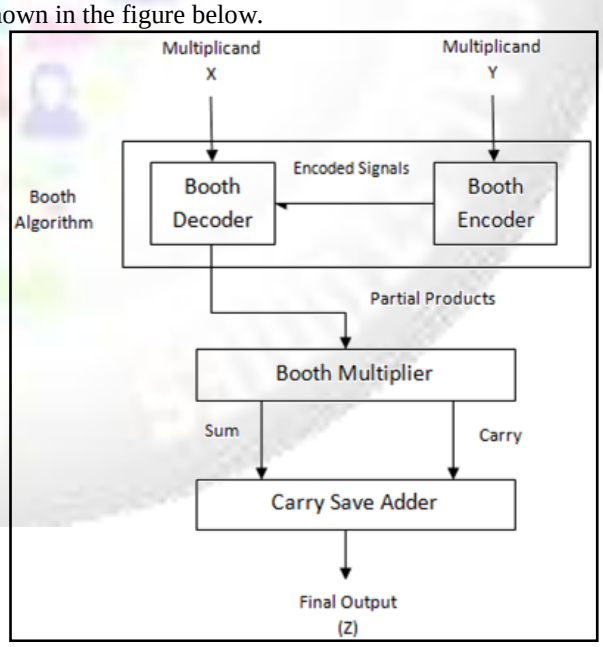

Fig. 2: Block Diagram of 64-Bit Modified Booth Multiplier

For designing of 64-bit Modified Booth multiplier; Booth encoder, Booth decoder, booth multiplier and carry save adder are necessary. These four modules are the essential for the design of 64-bit Modified Booth multiplier. For the 64-bit Modified Booth multiplier design, the most

important field booth multiplier since the performance of 64-bit Modified Booth multiplier is depend Booth multiplier unit. In this research work we have designed the 64-bit Modified Booth multiplier using Pipelining.

V. EXPERIMENTAL RESULTS

A. 64-Bit Modified Booth MultiplierRTL View

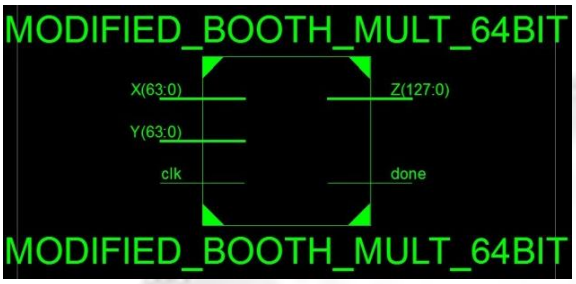

Fig. 3: RTL View of 64-Bit Modified Booth Multiplier

Figure 3 shows the detail RTL view of 64-bit Modified Booth Multiplier. It shows the register transfer logic view of the 64-bit Modified Booth Multiplier. It consists of CSA adder, MUX unit and full adder.

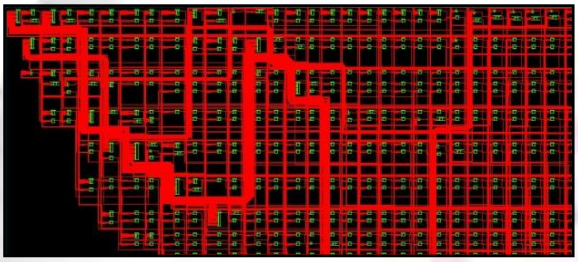

Fig. 4: Detailed RTL View of 64-Bit Modified Booth Multiplier

Figure 4 shows the detail RTL view of 64-bit Modified Booth Multiplier. It shows the register transfer logic view of the 64-bit Modified Booth Multiplier. It consists of CSA adder, MUX unit and full adder.

B. Simulation Result

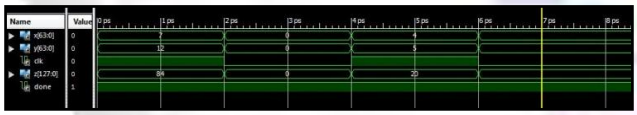

Fig. 5: Simulation Result of 64-Bit Modified Booth Multiplier in Binary form

Simulation result for 64-bit Modified Booth Multiplier is shown in figure 5. It has inputs x and y for multiplier and multiplicand, clk for clock. Output is z which represents multiplication of x and y, done for operation has been executed.

C. Result Analysis

The Proposed 64-bit Modified Booth Multiplier has found less delay, low power and high frequency. The device utilization of the design shows that the design and be upgrade and integrate on the same device.

Parameter	Proposed Design
Delay (ns)	7.361
Frequency (MHz)	135.847
Power (mW)	143

Table 2:Parameters (VIRTEX 7)

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	390	408000	0%
Number of fully used LUT-FF pairs	2557	204000	1%
Number of bonded IOBs	380	2567	14%

Table 3: Device Utilization Table (VIRTEX 7)

Table 3 shows Device Utilization Table for the device, VIRTEX 7 for the design of proposed 64-bit Modified Booth Multiplier using VHDL.

Parameters	Ref [1] (CLA)	Ref [2] Xilinx 10.1 (CLA & CSLA)	Ref [2] Xilinx 10.1 (CLA & CSLA)	Ref [2] Xilinx 10.1 (CLA & CSLA)	Ref [3] ISE Simulator (FPGA) (Vedic Maths)	Ref [4] Xilinx 10.2 (CLA & CSLA)	Proposed Work Xilinx 14.5i (CSA)
Delay (nsec)	51.18	210.881	224.370	260.910	315.725	172.02	7.361
Power (mW)	---	203	203	203	---	---	143
Frequency (MHz)	19.53	4.74	4.456	3.832	3.167	5.813	135.847

Table 4: Comparison Table

Table 4 shows Comparison Table of proposed 64-bit Modified Booth Multiplier using VHDL with different previous papers.

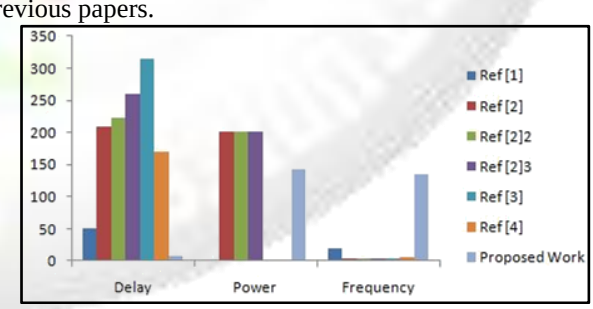

Figure 6: Comparison graph

Figure 6 shows Comparison graph of proposed 64-bit Modified Booth Multiplier using VHDL with different previous papers.

VI. CONCLUSION & FUTURE SCOPE

Been designed. VHDL language has been used to describe the design. The design, synthesis and simulation of 32-bit Modified booth multiplier based on pipelining using VHDL

have been described. Also, design, synthesis and simulation of 64-bit Modified booth multiplier based on pipelining using VHDL have been presented. Synthesis results are shown by RTL view of the design whereas simulation results are shown by the graphical representation of the design. Using this design any multiplication can be possible of 64-bits. After taking synthesis of the design it is found that 64-bit booth multiplication can be done in 7.361ns time and 135.847 MHz frequency with only 143mW power consumption.

If we suppose to implement this 64-bit Modified booth multiplier based on pipelining using FPGA technology it is possible to upgrade the system with new features as per user requirements. In future it can also be upgraded to 128-bit using same standards and it can be implemented on FPGA kit also. The hardware complexity can be reduced and integration of different circuits in a single chip can be possible on FPGA kit. In the near future applications like the ATMs, mobile phones, portable gaming kits can be implemented.

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 TECHNOLO GY (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 Speed

[8] Multiplier Architecture for Wide Bit Widths", In IEEE International Symposium on Circuits and Systems, volume 4, pages 53–56.

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

[10] 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.