International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

Performance of Improved speed pipelined floating point multiplier Architecture

Rajendra Prasad.K

Abstract*:Multiplier is a hardware component which usually covers an important chip area and must be reduced to create lots of functions in which multiplier frames shape an essential structure, including digital signal processing (DSP) systems and analytical approaches. The benefit of floating point representation across a fixed point (and integer) view is that a wider range of values can be represented. Since floating point numbers are stored in sign-magnitude type, the multiplier also requires unwritten integer numbers and standardization. The multiplier with the algorithm Revised Booth and save adder is one way to speed up the multiplier. The algorithm of Revised Booth reduces the number of incomplete products to create and is regarded as the quickest algorithm of propagation.

Keywords:Radix4 Booth Multiplier, Booth Algorithm, Partial Products Generation, Signed-Unsigned Multiplication

I. INTRODUCTION

Multiplication of digital computing systems is an important arithmetic operation. The multiplication process involves actually making partial products and then applying certain partial products to the final product. The multiplier speed therefore reduces the number of partial product and the adder speed. As the multipliers have an important impact on the system's performance, several high-performance algorithms and architectures are suggested. Floating point is a way of representing numbers and arithmetical applications in computer systems, ranging from simple computers to computers. The word floating point was taken from the idea that prior to the actual decimal point there is no fixed number of digits; that is that the decimal point can float. Although there are interpretations in which the number of digits, called fixed-point representations, are placed before and after the decimal point. In particular, depictions of floating points are slower than representations of fixed points but can accommodate a larger range of numbers [1]. In digital signal processing (DSP) technologies, such as video and communication systems, high speed Booth multipliers and pipelined Booth multipliers are used [2].Since mathematics with floating points saves a lot of computing power, many microprocessors come with a chip, which is called a floating point unit, specialized on floating point arithmetic performance. The mathematical coprocessors and numeric coprocessors are also called FPUs.

Manuscript published on November 30, 2019.
* Correspondence Author
Dr. Rajendra Prasad. K*, Assistant Professor, Department of ECE,MallaReddy Engineering College (A),Telangana India.

Floating point units areextensively included in a dynamic range of technological and engineering applications.This requires the development of faster arithmetic circuits of floating points.The division of floating points is much like the multiplication of integer. Even though floating points numbers are stored in sign-magnitude type, only unsigned integer numbers and normalization need to be handled by multipliers. Integer multiplier with the algorithm Modified Booth and save adder is one way to increase multiplier speed.

The algorithm of the modified Booth decreases the number of partial products to be produced and is known as the fastest algorithm. Several other studies have been carried out on multiplier architectures including arraying, parallel and pipeline multipliers, showing that pipeline processing is the most widely employed way to reduce digital circuit propagation delays.In this paper a design has been suggested for a fast floating point multiplier compatible with the IEEE 754 single precision specification. In order to achieve a generic design, the hardware description language of Verilog was utilized for the design of the complete multiplier unit, since it represents an enormous productivity improvement for circuit designers.

The main contributions and organization of this paper are summarized as follows: In section 2 we describe backgrounddetails of different multiplier methods adopted by the authors. Section3 discusses the proposed work. Section 4 deliberates results and discussions. Finally, in section 5, we concluded the paper.

II. BACKGROUND WORKS

In [3], the authors present an efficient implementation of an efficient Radix-4Booth 3:2 compressor configurable multiplier for both the 32 bit signed and non-signed numbers and the arithmetic floating point. Multiplication is always used in various uses of research and signal processing. Multipliers comprise one of many systems most important components. It delivers a dynamic arithmetic power, increased performance accuracy and high speed and low area usage. The model often bypasses the swapping activity of then functional output ranges dynamically. The redundant circuit can therefore be deactivated effectively, growing energy consumption and can running pace. The proposed multiplier design therefore outperforms the conventional multiplier in terms of area and speed efficiency. On the FPGA Spartan-6 XC6SLX9 board, the suggested multiplier model was introduced. In [4], the researchers proposed a new solution to BCD multipliers. The primary focus of the architecture proposed is the generation of partial products and simultaneous binary operations on two-digit columns.

11-digit multipliers for the production of a partial products are applied explicitly without any software translation by 4-bit binary multipliers. The binary results of the 1×1-digit multiplications are organized into the 2-digit partial products based on the column based on the most recent their two-digit positions. For the partial reduction of products, a binary decimal compressor architecture is built and used. In streamlined BCD 6-LUT adders, these diminished partial components are substituted. Parallel binary tasks and the enhanced BCD adder lead to better performance and less energy consumption. The suggested solution to Xilinx Virtex-5 and Virtex-6 FPGAs were applied with focus on increasing critical path delays.

In [5], the researchers used FPGA as an active enforcing method, but the development barrier encountered many times is constrained in capital. Optimizations seems to be the way to design big circuits, particularly the entire chip system (SOC) or chip network (NOC) on this device. There are so many ways to improve multipliers, with some improvements to conventional methods as well as their application and testability on FPGA. This paper shows the implementation of the 3x3 unsigned integer switch and add multiplier fixed-point finite duration by implementing such functional improvements The revised version leads to fewer use in the Lookup Tables (LUTs) and less delay because of lower logic levels than traditional techniques.

III. PROPOSED WORK

Fig.1 shows a single-accuracy representation of the binary format in IEEE 754, consisting of 1-bit of a sign (S), an 8-bit exponent (E) and a 23-bit fraction (M or Mantissa). An additional bit is applied to the fraction to form the significand. If there is one exponent in the MSB greater than 0 and less than 255, then the sum is said to be a fixed integer, which defines the real number by an eq.(1).

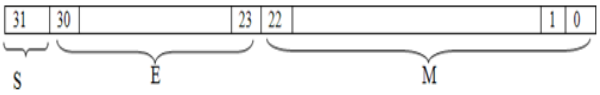

Fig.1. IEEE single precision floating point format [6]

$$Z = (-1^{S}) * 2^{(E\text{-}Bias)} * (1.M) \quad (1)$$

Where $M = m_{22}\,2^{-1} + m_{21}\,2^{-2} + m_{20}\,2^{-3} + \ldots + m_{1}\,2^{-22} + m_{0}\,2^{-23}$; Bias = 127.

Multiplication

This unit multiplies the unsigned significand and places the decimal point in the multiplication product. The consequence of the multiplication of significand is considered the intermediate product. The unsigned significand multiplication is performed on 24-bit. Multiplier output should also be treated in order to avoid impacting the entire performance of multipliers. The Carry save multiplier architecture of a 24×24 bit is used because it has progressive speed and a simple architecture. In the transmission carry save multiplier, the transmission bit is diagonally passed down (i.e. the transmitting bit is propagated to the next stage). Partial products are produced by ANDing and moving together the inputs to the correct adder. Carry save multiplier has three primary phases: The first phase is a half-adder range. The intermediate steps were complete adders. The middle step number is equivalent to the length minus two. The final stage is a range of ripple carry adders. This stage is called the phase of vector fusion as shown in Fig.2.

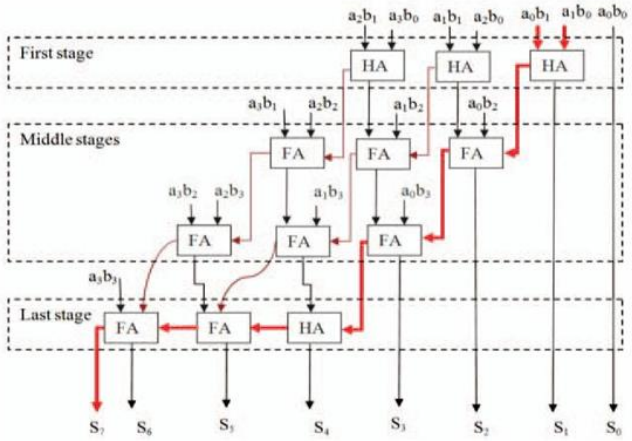

Fig. 2. 4x4 bit Carry Save multiplier [6]

Normalization:

To order to have a leading "1" to the left of the decimal point, the effect of the value multiplication (i.e. to bit 46 on the intermediate product) has to be uniform. Because the inputs are standardized, the intermediate product has the lead at bit 46 or 47.

***Floating Point Multiplier diagram*:**

Fig.3 displays the floating point multiplier block diagram. The hidden bit is appended to the manta and multiplied by the transportable save multiplier. Eventually, the outcome is averaged and the output measured.

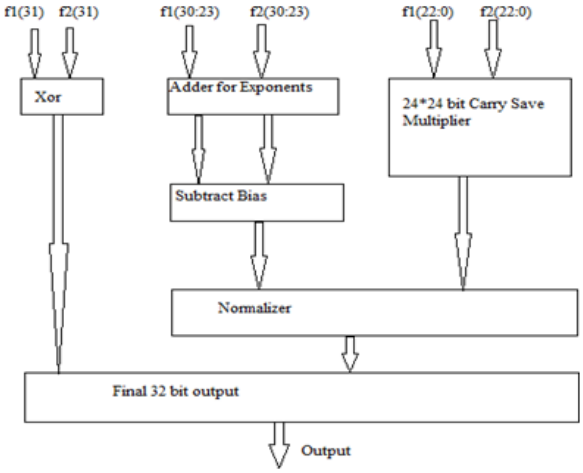

Fig.3. Floating Point Multiplier Architecture

Booth multiplier

Multiplication is a numerical procedure which is an appended method by which a number of times are applied to an integer. A number (multiplicand) will be applied to itself a number of times to create a value (product) according to a different number (multiplier). To accelerate up the transition, it is divided into the following stages:

Generation of partial products:

The Radix-8 modified algorithm is used to generate the partial products. In particular, the higher the radix, the more partial products generated that are required to quantify the component in less cycles.

High radix recoding has therefore become more and more widely used in modern high speed arithmetic units and single chip multipliers. Since the multiplier and multiplier are 24-bits, 8 partial products are generated in this algorithm.

Radix-8 Modified Booth's Algorithm

Recoding expanded to 3 bits at a time-4 bits overlapping each band. Radix-8 recoding is much like Radix-4 algorithm but we now use quartets of bits rather than triples. As a consequence, a multiplier based on radix-8 yields less partial products than a radix-4 multiplier, but the estimation of every other partial product is much more complicated. A partial component referring to an x=+3 encoding involves the estimation of 3x and therefore a total addition. Every quartet is codified with the Table.1 as a signed integer.

Table 1. Signed-digit and its equivalent quartet value

Quartet value	Signed-digit value	Quartet value	Signed-digit value
0000	0	1000	-4
0001	+1	1001	-3
0010	+1	1010	-3
0011	+2	1011	-2
0100	+2	1100	-2
0101	+3	1101	-1
0110	+3	1110	-1
0111	+4	1111	0

Partial Product Reduction

As it is clear that multiplier blocks revealed that a small number of supplements were needed to multiply a data sample with several constant coefficients. However, carry save adders are a better choice for high-speed applications than carrying propagation adder. 8 Partial products are produced using the Algorithm for Radix-8 Modified Booth. The compressors were divided by 4:2.

Carry save Adder

A Carry-Save Adder is only a set of one-bit full adders, without any communication. The key use of a carry-save adder is to measure partial products in integer multiplication. Compressors 4:2 are used as additional hold recover. It is made up of two 3:2 (full adders) compressors in the series and includes 4 XOR delays as shown in Figure 2. This compressor is suitable for the introduction of partial products due to its frequent interconnection. The structure of the 4:2 compressor compresses five bits of partial products into three. Originally two 4:2 compressors are used to reduce the amount and hold of the set of 4 partial items each. These last four partial items from the two 4:2 compressors are further decreased to manufacture and hold the final number. The compressor 4:2 consists of 2 complete adders. The last carry is protected and therefore named the carry save adder. The 4:2 compressor delay is the same as for 4 XOR gates.

Final Stage Carry Propagate Adder

Ripple Carry Adder:

The ripple carry adder is used by inserting a partial material from the carry save adders to achieve the full total and the effect. It generates a logical loop with several total adders to connect N-bit numbers. -full adder joins a C_{in}, which is the prior adder C_{out}. This form of adder is an adder with ribs, since each bit "adds" to the next adder. In the final stage, the 48-bit sum and the output obtained from the partial product accumulator are added to give the mantissa product. This unsigned adder often introduces the exponent of the first output into the exponent of the second input and excludes the incomplete portion (127) from the supplement result (i.e. A_exponent + B_ exponent-partial). The value is considered the intermediate exponent

IV. RESULTS AND DISCUSSION

In this section results are discussed using Xillinx Design suite with Modelsim simulator is used.

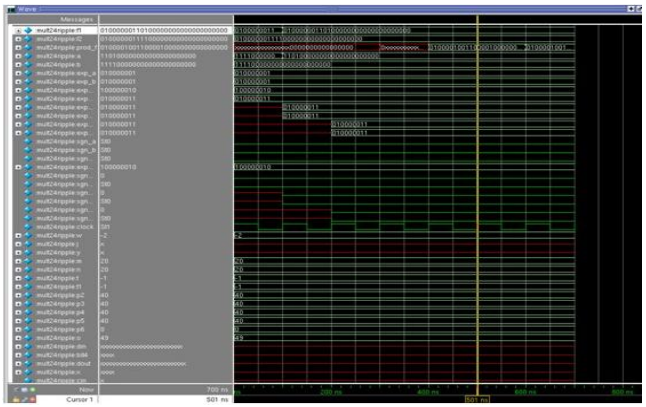

Fig.4. Simulated output of 5-stage Pipelined Floating Point Multiplier

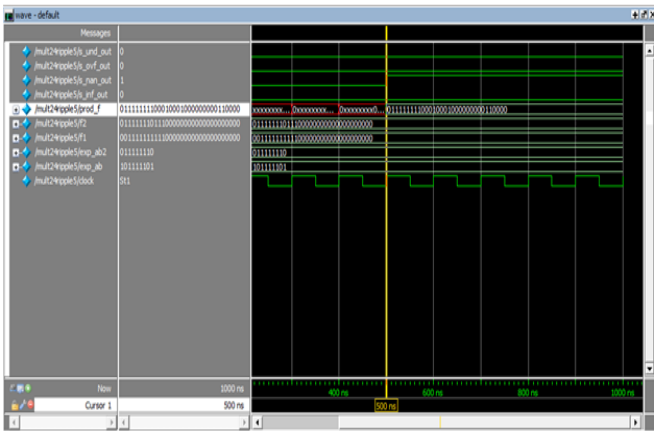

Fig.5.Simulated output of Floating Point Multiplier with exceptions

V. CONCLUSION

The booth multiplier technique increases energy and logic region utilization considerably more with a minimal delay than traditional architecture. It leads to high efficiency by addition of pipeline phases, but at the expense of area and energy. Booth's Recoding Strategy can be used for partial products generation. The Algorithm of the Radix-2 Booth, Radix-4 and Radix-8 revised Booths for the processing of partial products was related. Because the multiplier and multiplicand consists of 24-bit, 8 partial products are produced using Radix-8 Modified Booth's Algorithm. The use of carry save adders reduces these 8 partial products. Furthermore, Ripple Carry Adder reduces the partial products acquired by carrying saving additives. The result of the significance (intermediate product) multiplication is normalized to be a leading "1" just to the left of the decimal point.

REFERENCES

1. Anna Jain, Baisakhy Dash, Ajit Kumar Panda, Muchharla Suresh, “FPGA design of a fast 32-bit floating point multiplier unit”, 2012 IEEE International Conference on Devices, Circuits and Systems, Coimbatore, India, March 15-16, pp 545 – 547,2012.
2. Narsampallin Bhargavi, Ms. Shatabdi Nandi, D. Devi Lavanya, ”Design and Performance Analysis of Multiplier using Wallace-Booth Algorithm”, in proc. IJSER, Volume 5, Issue 8,ISSN 2229-5518, August2014.
3. Shuli Gao, Dhamin Al-Khalili, J. M. Pierre Langlois, Noureddine Chabini, “Efficient Realization of BCD Multipliers Using FPGAs”, International Journal of Reconfigurable Computing Volume 2017.
4. Kishore Shinde, A.K. Kureshi, “Hardware implementation of configurable booth multiplier on FPGA”, Proceedings of 49th IRF International Conference, 21st February 2016.
5. Aneela Pathan, Tayab D Memon, “An Optimized 3x3 Shift and Add Multiplier on FPGA”, Proceedings of 2017 14th International Bhurban Conference on Applied Sciences & Technology (IBCAST), IEEE January, 2017.
6. Al-Ashrafy, M. Salem, A. Anis, W., Mentor Graphics, “An efficient implementation of floating point multiplier”, IEEE Electronics, Communications and Photonics Conference(SIECPC), Saudi International , April 24-26 , pp 1 – 5, 2011.

AUTHOR PROFILE

Dr. Rajendra Prasad.K is presently working asAssistant Professor in the Department of Electronicsand communication Engineering, MallaReddy Engineering College (A),Telangana India. His areas ofinterests include VLSI system Design, Low power VLSI.