International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 23 (2015) pp 43880-43884

An Efficient Low Power and Delay Optimized Floating Point Multiplier using Modified Booth's Algorithm

A. Sirisha

M. Tech Student, ECE (ES) Gudlavalleru Engineering College Gudlavalleru, Krishna Dist., AP, India-521356
e-mail: anne.sirishaecea@gmail.com

A. V. N. Tilak

Professor of ECE, Dean-P. G. Studies and R&D Gudlavalleru Engineering College Gudlavalleru, Krishna Dist., AP, India-521356
e-mail: avntilak@yahoo.com Telephone No.: 08674-273737, Fax No: 08674-273957

Abstract
This paper presents a design methodology for low power and delay optimized single precision floating-point multiplier based on Booth encoded parallel multiplier. For partial product generation, a new modified Booth encoding (MBE) scheme (Radix-8) is proposed to improve the performance of traditional MBE schemes. Addition of partial products is carried out using Modified Carry Select Adder (MCSA). This design is implemented using Xilinx 14.7 ISE Simulator in VHDL. Floating point multiplier is implemented using modified Booth's algorithm which reduced the power consumption by 22% and delay by 34%.

Keywords: Modified Booth Encoder, Radix-8, Floating-Point Multiplication, Partial products generation, Partial products accumulation, VHDL.

I. Introduction

With the recent rapid advances in multimedia and communication systems, real time signal processing like audio signal processing, video/image processing, or large-capacity data processing are increasingly being needed. The multiplier and multiplier-and-accumulator (MAC) are the essential elements of the digital signal processing such as filtering, convolution, and inner products. Most digital signal processing methods use non-linear functions such as discrete cosine transform (DCT) or discrete wavelet transform (DWT) [1]. Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and arithmetic determines the execution speed and performance of the entire calculation. Because the multiplier requires the longest delay among the basic operational blocks in digital system, the critical path is determined by the multiplier, in general.

Booth's algorithm is a smart move for multiplying numbers. For high-speed multiplication, radix-4 modified Booth's algorithm (MBA) [2] is commonly used. However, this cannot completely solve the problem due to the long critical path for multiplication. The most effective way to increase the speed of a multiplier is to reduce the number of the partial products because multiplication proceeds a series of additions for the partial products. To reduce the number of calculation steps for the partial products, radix-8 MBA has been applied.

II. Background

Booth's algorithm was invented by Andrew Donald Booth in 1950 while doing study on crystallography at Birbeck College in Bloombury, London. Booth used reception desk calculators that shift faster than adding and thus formed algorithm increases the speed [3]. Booth's algorithm is important in the study of computer architecture. Andrew Donald Booth was a British electrical engineer, computer scientist and physist who led the innovation of the magnetic drum memory for computers and invented Booth's multiplication algorithm.

III. Proposed Work

The multiplication operation is present in many parts of a digital system or digital computer, most notably in signal processing, graphics and scientific computation. With advances in technology, various techniques have been proposed to design multipliers, which offer high speed, low power consumption and lesser area, thus making them suitable for various high speed, low power and compact VLSI implementations. These three parameters i.e. power, area and speed are always traded off. The present work is devoted for the design and simulation of radix-8 Booth Encoder multiplier for signed-unsigned numbers. The radix-8 Booth encoder circuit generates n/3 partial products in parallel. By extending sign bit of the operands and generating an additional partial product the sign of unsigned radix-8 Booth encoder multiplier is obtained. The modified carry select adder is used to speed up the multiplier operation. Since signed and unsigned multiplication operation are performed by the same multiplier unit, the required hardware and the chip area reduces and this in turn reduces power dissipation and cost of the system.

3.1 Floating Point Multiplier Block Diagram

The numbers on which the multiplication must be performed are given as input to the floating point representation. Then the numbers are represented in floating point format. Next the recoding is performed according to the radix-8 MBA. Generation of partial products and accumulation are important

in multiplication process. For accumulation, modified carry select adder is used. The floating point multiplier block diagram is shown in figure 1.

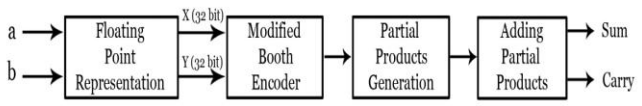

Figure 1: Floating-point multiplier block diagram

3.1.1 Floating-Point Representation

Floating point is denoting a mode of representation of numbers as two sequences of bits, one representing the digits in number and the other an exponent which determines the position of the radix point. The IEEE has standardized the computer representation for binary floating point numbers. Single-precision floating-point format is a format that occupies 4 bytes (32 bits) in computer memory and represents a wide dynamic range of values by using a floating point [3]. Representation of single precision binary format is shown in figure 2, starting from MSB as one-bit sign (S), an 8-bit exponent (E) and a twenty-three-bit fraction (M).

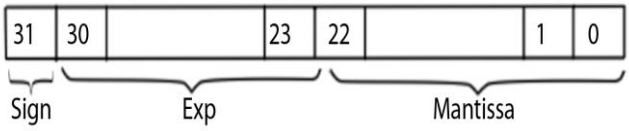

Figure 2: IEEE-754 single precision format

$Z = (-1^{s}) * 2^{(E-Bias)} * (1.M)$
Sign= (31^{st} bit of X) XOR (31^{st} bit of Y)
Bias is 127 for single precision floating point format
Exponent = $E_1 + E_2 - Bias$
Multiplication = $(1.M_1 * 1.M_2)$
M_1 is the mantissa of X
M_2 is the mantissa of Y

3.1.2 Modified Booth Encoder

Modified Booth's algorithm is twice as fast as Booth's algorithm. Modified Booth encoding algorithm is an efficient way to reduce the number of partial products by grouping consecutive bits in one of the two operands to form the signed multiples. The operand that is Booth encoded is called the multiplier and the other operand is called the multiplicand.
Radix-8 Booth recoding applies the same algorithm as that of radix-4[5], but here quartets of bits are taken instead of triplets as shown in the figure 3. Each quartet is codified as a signed digit using Table 1. Radix-8 algorithm reduces the number of partial products to n/3, where n is the number of multiplier bits [6]. Thus it allows a time gain in the partial products summation.

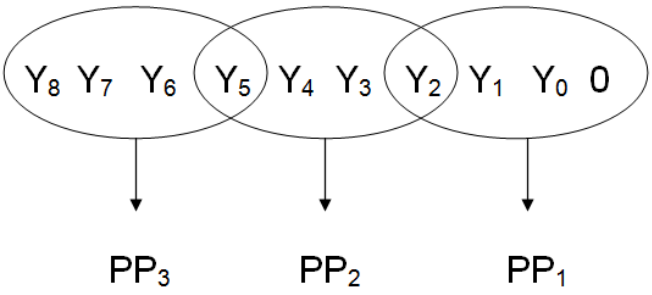

Figure 3: Grouping of bits in radix-8 method

Table 1: Recoding table of radix-8 MBA

Multiplier Bits Y_{i+2}	Y_{i+1}	Y_i	Y_{i-1}	Recoded Operation on multiplicand, X
0	0	0	0	0X
0	0	0	1	+X
0	0	1	0	+X
0	0	1	1	+2X
0	1	0	0	+2X
0	1	0	1	+3X
0	1	1	0	+3X
0	1	1	1	+4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-1X
1	1	1	0	-1X
1	1	1	1	0X

3.1.3 Partial Products Generation

The algorithm and flowchart for partial products generation of radix-8 modified Booth encoder are given below.

a) Algorithm for radix-8 modified Booth encoder

The number of subsequent calculation stages can be decreased by enhancing the parallelism operation. So, one of the solutions of realizing high speed multiplier is to enhance parallelism operation. The radix-4 Booth multiplier is the modified version of the conventional version of the Booth algorithm (Radix-2), which has two drawbacks. They are:

(i) Inconvenient in designing parallel multipliers because the number of add-subtract operations and the number of shift operations becomes variable.
(ii) When there are isolated 1's, the algorithm becomes inefficient.

These problems can be overcome by using radix-8 modified Booth's multiplier.

1) If n is even, then the sign bit 1 position is extended.
2) A 0 bit is appended to the right of the LSB of the multiplier.
3) Each partial product will be 0, + X,-X, +2X,-2X,-3X, +3X,-4X, +4X.

b) Flow Chart of radix-8 modified Booth's algorithm

Here X is multiplicand, Y is multiplier, count is initialized to n. Four bits are grouped and according to the corresponding binary values, particular operation is performed as shown in figure 4.A product formed by multiplying the multiplicand by one digit of the multiplier is called the partial product. Partial products are used as intermediate steps in calculating larger products. Partial product generator is designed to produce the product by multiplying the multiplicand with 0, 1,-1, 2,-2,-3, 4, 3, 4. For product generator, multiply by zero means the multiplicand is multiplied by "0". Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place. Multiply by "-4" is to shift left two bits the two's complement of the multiplicand value and multiply by "4" means just shift left the multiplicand by two places.

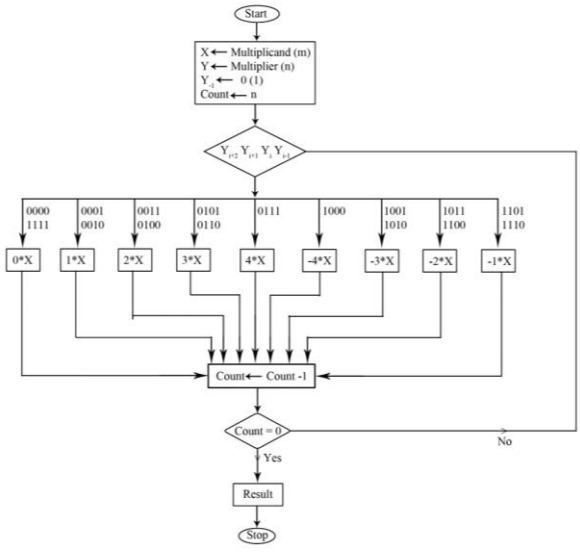

Figure 4: Radix-8 flow chart

Here an odd multiple of the multiplicand exists, 3Y, which is not immediately available. To generate it the previous addition: 2Y+Y=3Y is to be performed. But the multiplier is being designed for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. One should take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner a better overall multiplication time can be attained, or at least comparable to the time we could obtain using radix-4 architecture (with the additional advantage of using a less number of transistors). To generate 3Y with 8-bit words we only have to add 2Y+Y, that is to add the number with the same number shifted one position to the left.

3.1.4 Adding Partial Products

Modified carry select adder is used for exponent addition of the given two floating-point numbers. MCSA is as shown in the figure 5. Modified Carry Select Adder (MCSA) is one of the fastest adders used in many data processing processors to perform fast arithmetic functions. The CSLA is used in computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum. Adding two n-bit numbers with a carry select adder is done with two adders (two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being 'zero'(C_{in}=0) and the other assuming 'one'(C_{in}=1). After the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known. Area consumed is more due to the use of dual RCA's. To reduce the area and power consumption, Modified CSA is used. BEC (Binary to Excess-1 Convertor) is used instead of RCA with C_{in} =1.

The basic idea of this work is to use Binary to Excess-1 Converter (BEC) instead of RCA with C_{in}=1 in the regular CSLA to achieve lower area and power consumption [8]. The 4-bit BEC is shown in figure 6. The main advantage of this BEC logic comes from the lesser number of logic gates than the n-bit Full Adder (FA) structure.

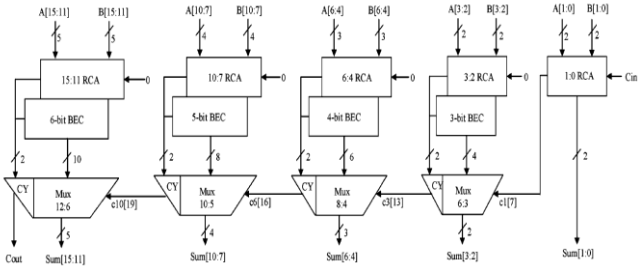

Figure 5: Modified carry select adder

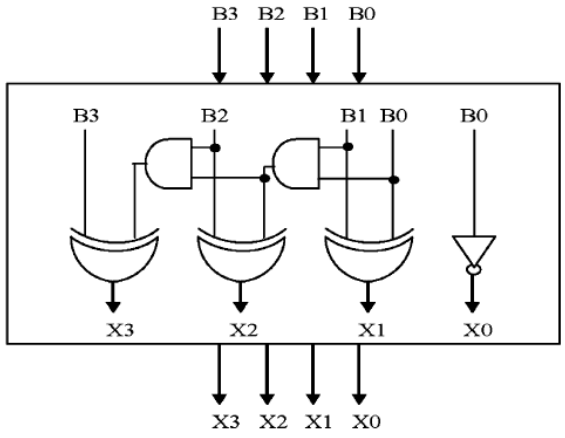

Figure 6: 4-bit BEC

Figure 7 illustrates how the basic function of the CSLA is obtained by using the 4-bit BEC together with the MUX. One input of the 8:4 MUX gets (B3, B2, B1, and B0) as one input and BEC output as second input. This produces two possible partial results in parallel and the MUX is used to select either the BEC output or the direct inputs according to the control signal C_{in}. The importance of the BEC logic stems from the large silicon area reduction when the CSLA is designed with large number of bits.

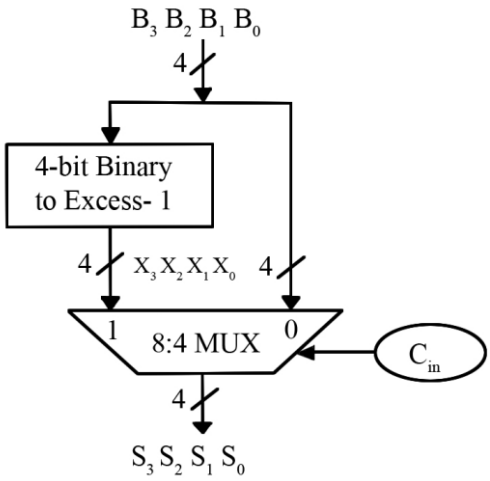

Figure 7: 4-bit BEC with 8:4 MUX

The Boolean expressions of the 4-bit BEC shown in figure 6 are listed as below (note the functional symbols ~ NOT, & AND, ^XOR).

$X_0 = \sim B_0$
$X_1 = B_0 \wedge B_1$
$X_2 = B_2 \wedge (B_0 \& B_1)$
$X_3 = B_3 \wedge (B_0 \& B_1 \& B_2)$

The truth table for BEC is shown in table 2.

Table 2: Truth table for BEC [8]

Binary Bits	Excess -1 Bits
$B_3\ B_2\ B_1\ B_0$	$X_3\ X_2\ X_1\ X_0$
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	1 0 1 0
1 0 1 0	1 0 1 1
1 0 1 1	1 1 0 0
1 1 0 0	1 1 0 1
1 1 0 1	1 1 1 0
1 1 1 0	1 1 1 1
1 1 1 1	0 0 0 0

IV. Results

The simulation result of radix-8 MBA is shown in figure 8.

Inputs:
x = 00000000000000000010111010110101
y = 00000000000000001110001011101000

Output:
x y = 01000000100000010001000111101111

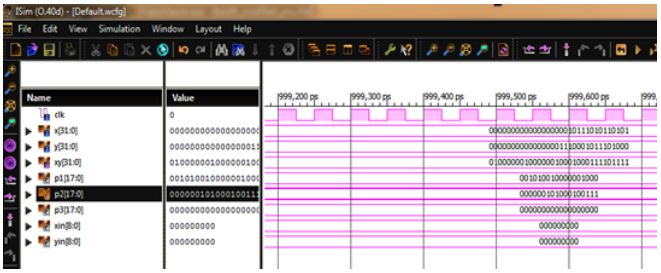

Figure 8: Simulation result of radix-8 modified Booth's algorithm

Simulation is done with Xilinx 14.7 using VHDL. Here X, Y are the 32-bit inputs, where X is the multiplicand and Y is the multiplier. XY is the output. Three partial products p1, p2, p3 are generated.

Power Analysis

Power Analysis is done using Xilinx Power Analyzer. Power is reduced by using modified Booth encoding scheme. For Dadda multiplier the power consumption is 2.202W as shown in figure 9. For radix-8 MBA the power consumption is 1.718W as shown in figure 10.

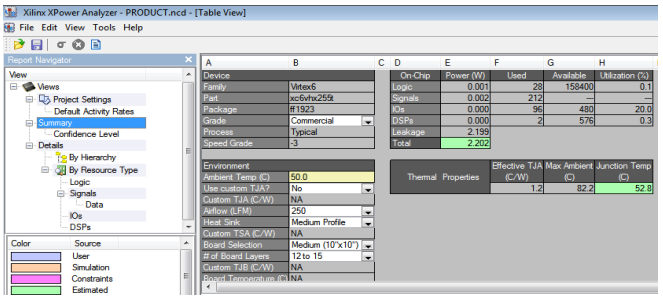

Figure 9: Power report of Dadda multiplier

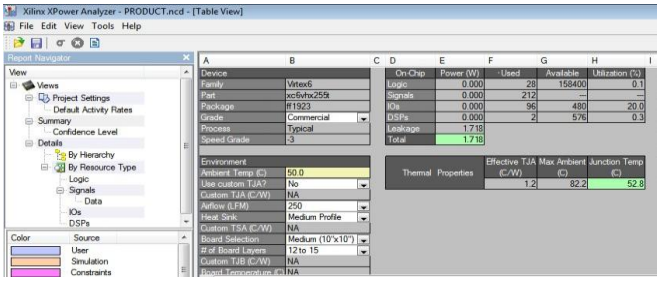

Figure 10: Power report of radix-8 modified Booth's algorithm

Timing Analysis

Delay is reduced by using modified carry select adder. For Dadda Multiplier with carry save adder, the time consumed is 5.473 ns. For radix-8 modified Booth algorithm with modified carry select adder, the time consumed is 3.613 ns.

V. Conclusion

A new radix-8 modified booth multiplier architecture to execute the multiplication-accumulation operation is implemented, which is the key operation for digital signal processing and multimedia information processing. By removing the independent accumulation process that has the largest delay and merging it to the compression process of the partial products, a reduction in power consumption by 22 % and delay by 34% is obtained.

VI. References

[1] C. S. Wallace, "A Suggestion for a Fast multiplier", IEEE Transactions Electronic Computers., vol. EC-13, no.1, pp. 14-17, Feb. 1964.

[2] Shubhi Shrivastava, Pankaj Gulhane, "Optimized Model of Radix-4 Booth Multiplier in VHDL", International Journal of Emerging Technology and Advanced Engineering, vol. 4, issue 9, September 2014.

[3] Deepali Chandel, Gagan Kumawat, Pranay Lahoty, Vidhi Vart Chandrodaya, Shailendra Sharma, "Booth Multiplier: Ease of Multiplication", International Journal of Emerging Technology and Advanced Engineering, vol. 3, issue 3, pp. 326-330, March 2013.

[4] Jeevan, Narender.S, Reddy. C.V. K & Sivani.K, "A High Speed Binary Floating Point Multiplier using Dadda Algorithm", Proceedings of IEEE Conference on computing, 2013, pp. 455-460, 2013

[5] Mr.Hemantkumar, H. Nikhare, Prof.Ashish Singhadia, "A detailed review on architectures for 2-DWT by using radix-4 Booth multiplier", International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 3, issue 5, pp. 80-85, March 2015.

[6] Ruchi Sharma, "Analysis of Different Multiplier with Digital Filters Using VHDL Language", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, vol. 2, issue-1, pp. 45-48, October 2012.

[7] A B. Pawar, "Radix-2 Vs Radix-4 High Speed Multiplier" International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, issue 3, pp. 329-333, March 2015.

[8] Prof. Mary Joseph, Renji Narayanan, "16 Bit Carry Select Adder with Low Power and Area", International Journal on Recent and Innovation Trends in Computing and Communication, vol. 2, issue 5, pp. 1223-1225, May 2014.

Biographical Sketch

A. Sirisha received B. Tech degree in Electronics and Communication Engineering from Sri Sunflower College of Engineering and Technology, Lankapalli, AP, India. She is pursuing M. Tech (Embedded Systems) in Gudlavalleru Engineering College, Gudlavalleru.

A.V. N. Tilak has obtained his B.E., M. Tech. and Ph.D. from MIT Manipal, IIT Kanpur, and IIT Madras respectively. His areas of interest are Microelectronics, Digital Design, and Low Power VLSI Design. Dr. Tilak is a member of IEEE, Fellow IETE, Fellow IE(I), and Life member of ISTE.