Design and Simulation of Radix-8 Booth Encoder Multiplier for Signed and Unsigned Numbers

Minu Thomas[1]
[1]M-Tech student
[1]Mangalam College of Engineering Ettumannoor, Kottayam ,India

Abstract--This paper presents the design and simulation of signed-unsigned Radix-8 Booth Encoding multiplier. The Radix-8 Booth Encoder circuit generates n/3 the partial products in parallel. By extending sign bit of the operands and generating an additional partial product the signed of unsigned Radix-8 BE multiplier is obtained. The Carry Save Adder (CSA) tree and the final Carry Look ahead (CLA) adder used to speed up the multiplier operation. Since signed and unsigned multiplication operation is performed by the same multiplier unit the required hardware and the chip area reduces and this in turn reduces power dissipation and cost of a system. The simulation is done through Verilog on xiling13.3 platform which provide diversity in calculating the various parameters.

Keywords: Array multiplier, Baugh-Woolley multiplier, Braun array multiplier, CLA, CSA, Radix-8 Booth Encoding multiplier, Signed-unsigned,

I. INTRODUCTION

Enhancing the processing performance and reducing the power dissipation of the systems are the most important design challenges for multimedia and digital signal processing (DSP) applications, in which multipliers frequently dominate the system's performance and power dissipation. Multiplication consists of three major steps: 1) recoding and generating partial products; 2) reducing the partial products by partial product reduction schemes (e.g., Wallace tree)to two rows; and 3) adding the remaining two rows of partial products by using a carry-propagate adder (e.g., carry look ahead adder) to obtain the final product. There are already many techniques developed in the past years for these three steps to improve the performance of multipliers.

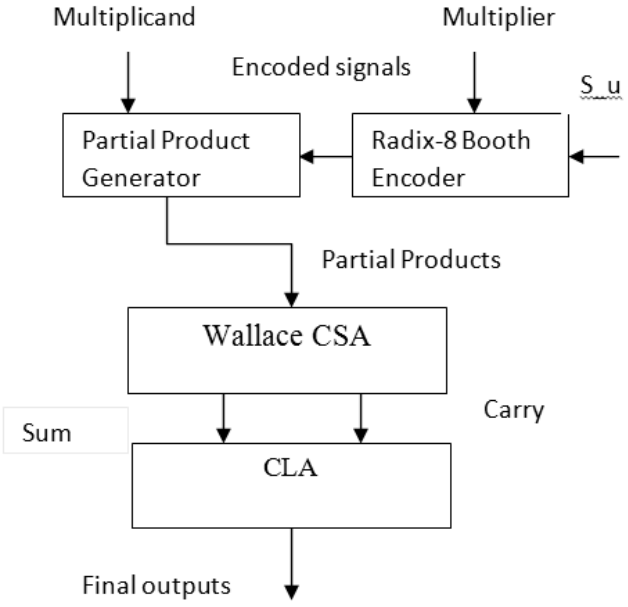
The multiplication can be performed on: 1) Signed Numbers; 2) Unsigned Numbers. Signed multiplication a binary number of either sign (two numbers whose sign may are not necessarily positive) may be multiplied. But, in signed multiplication the sign-extension for negative multiplicands is not usable for negative multipliers and there are large numbers of summands due to the large sequence of 1's in multiplier.

The conventional modified Booth encoding (MBE) generates an irregular partial product array because of the extra partial product bit at the least significant bit position of each partial product row. Therefore papers presents a simple approach to generate a regular partial product array with fewer partial product rows and negligible overhead, thereby lowering the complexity of partial product reduction and reducing the area, delay, and power of MBE multipliers. But the drawback of this multiplier is that it functions only for signed number operands. The modified-Booth algorithm is extensively used for high-speed multiplier circuits. Once, when array multipliers were used, the reduced number of generated partial products significantly improved multiplier performance. In designs based on reduction trees with logarithmic logic depth, however, the reduced number of partial products has a limited impact on overall performance. The Baugh-Wooley algorithm is a different scheme for signed multiplication, but is not so widely adopted because it may be complicated to deploy on irregular reduction trees. Again the Baugh-Wooley algorithm is for only signed number multiplication.

The array multipliers and Braun array multipliers operates only on the unsigned numbers. Thus, the requirement of the modern computer system is a dedicated and very high-speed multiplier unit that can perform multiplication operation on signed as well as unsigned numbers.

II. PROPOSED SIGNED UNSIGNED RADIX-8 BOOTH ENCODER MULTIPLIER

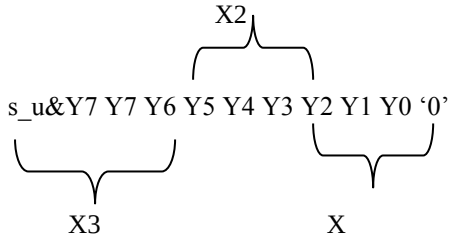
The inputs of the multiplier are multiplicand X and multiplier Y. The Booth encoder encodes input Y and derives the encoded signals The Booth decoder generates the partial products according to the truth table The Wallace tree computes the last two rows by adding the generated partial products.



The last two rows are added to generate the final multiplication results using the carry Look-ahead adder (CLA).

III. BOOTH ENCODER

Booth's algorithm involves repeatedly adding one of two predetermined values to a product P, and then performing a rightward arithmetic shift on *P*.Radix-8 Booth encoding is most often used to avoid variable size partial product arrays. Before designing Radix-8 BE, the multiplier has to be converted into a Radix-8 number by dividing them into four digits respectively according to Booth Encoder Table given afterwards. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier.



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

Table (1): Radix-8 recoding

A. SIGN EXTENSION CORRECTOR

Sign Extension Corrector is designed to enhance the ability of the booth multiplier to multiply not only the unsigned number but as well as the signed number. The working principle of sign extension that converts signed multiplier signed unsigned multiplier as follows. One bit control signal called signed-unsigned(s_u) bit is used to indicate whether the multiplication operation is signed number or unsigned number .when sign-unsign s_u=0, it indicates unsigned number multiplication and when s_u=1, it indicates signed number multiplication.

Sign unsigned	Type of operation
0	Unsigned multiplication
1	Signed multiplication

Table (2): signed unsigned radix-8 Booth Encoder operation

B. PARTIAL PRODUCT GENERATOR

A product formed by multiplying the multiplicand by one digit of the multiplier when the multiplier has more than one digit. Partial products are used as intermediate steps in calculating larger products.

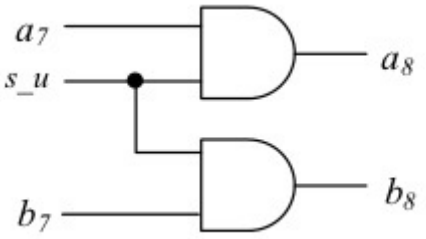


Fig (2): Logic diagram for Sign converter

Partial product generator is designed to produce the product by multiplying the multiplicand A by 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 bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by two place. Here we have an odd multiple of the multiplicand, 3Y, which is not immediately available. To generate it we need to perform this previous add: 2Y+Y=3Y. But we are designing a multiplier for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. We have tried to take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner we try to attain a better overall multiplication time, 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.

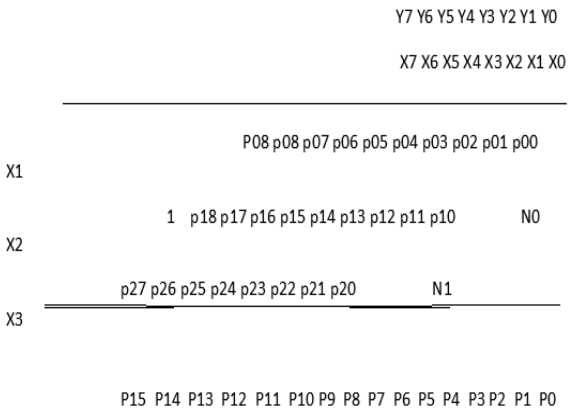


Fig (3): 8*8 Multiplier for unsigned multiplier

C. WALLACE TREE ADDER

Wallace tree has been used in this project in order to accelerate multiplication by compressing the number of partial products. This design is done using half adders; Carry save adders and the Carry Look Ahead adders to speed up the multiplication.

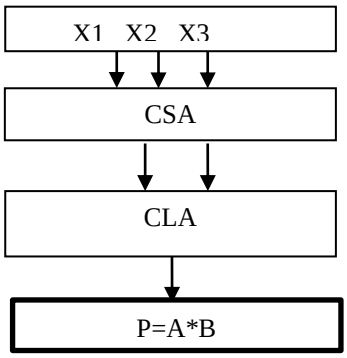


Fig (4): Partial product adder logic

In Fig 3 there are 3-partial products namely X1, X2, X3. These partial products are added by the carry save adder (CSA) and the final stage is carry look ahead (CLA) adder. CSA adder takes three inputs and produce sum and carry parallel. There is one CSA. Three partial products are added by the CSA tree and finally when there are only two outputs. Left out then finally CLA adder is used to produce final result

IV. SIMULATION RESULTS

The multiplier unit design applying the proposed radix-8 Booth encoder Multiplier for signed and unsigned numbers was specified in verilog, Simulated in Xilinx13.3.

Verilog code is written to generate the required hardware and to produce the partial product. For CSA adder and CLA adder after the successful compilation the RTL view generated is shown in Fig 5

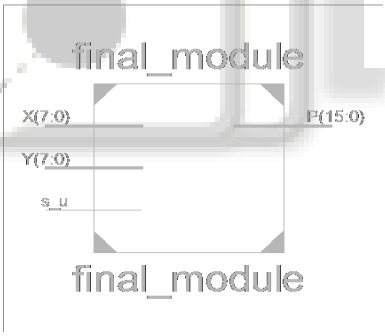


Fig (5): RTL view of 8*8 signed unsigned multiplier

Fig 6 shows the simulation result of signed unsigned numbers .when control signal s_u =0, the 8bit operands are considered as unsigned and the product of 11111111(255)*11111111(255)=1111111000000001(65025).and when the control signal s_u=1,the 8bit operand considered as signed and the product of 11111111(-1)*11111111(-1)=0000000000000001(+1)

Word size	Multiplier type	Delay(ns)
8 bit	Radix-4 signed unsigned booth multiplier	23.080
8 bit	Radix-8 signed unsigned booth multiplier	23.046

Table (3): Comparison to other multiplier

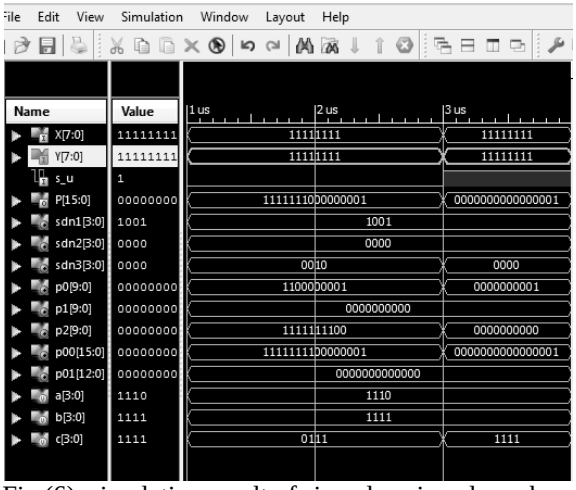


Fig (6): simulation result of signed unsigned numbers

V. CONCLUSION

It has been performed the design, and simulation of a 8x8 bit, radix-8, multiplier unit for signed and unsigned numbers using Xilinx 13.3 platform. In all multiplication operation product is obtained by adding partial products. thus the final speed of the multiplier circuit depends on the speed of the adder circuit and the number of partial products generated .radix-8 booth encoded technique used then there are only 3 partial products and only one CSA and CLA is required to produce the final product

REFERENCES

[1] W. –C. Yeh and C. –W. Jen, "High Speed Booth encoded Parallel Multiplier Design," IEEE transactions on computers, vol. 49, no. 7, pp. 692-701, July 2000.

[2] Shiann-Rong Kuang, Jiun-Ping Wang, and Cang-Yuan Guo, "Modified Booth multipliers with a Regular Partial Product Array," IEEE Transactions on circuits and systems-II, vol 56, No 5, May 2009.

[3] Li-Rong Wang, Shyh-Jye Jou and Chung-Len Lee, "A well-structured Modified Booth Multiplier Design" 978-1-4244-1617-2/08/$25.00 c2008 IEEE.

[4] Soojin Kim and Kyeongsoon Cho "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges" World Academy of Science, Engineering andTechnology2010.

[5] Magnus Sjalander and Per Larson-Edefors. "The Case for HPM-Based Baugh-Wooley Multipliers," Chalmers University of Technology, Sweden, March 2008.

[6] Z Haung and M D Ercegovac, "High performance Low Power left to right array multiplier design" IEEE trans.Computer, vol 54 no3, page 272-283 Mar 2005.

[7] Hsing-Chung Liang and Pao-Hsin Huang, "Testing Transition Delay Faults in Modified BoothMultipliers by Using C-testable and SIC Patterns"IEEE2007, 1-4244-1272-2/07.

[8] Aswathy Sudhakar, and D. Gokila, "Run-Time Reconfigurable Pipelined Modified Baugh-Wooley Multipliers," Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 3 Number 2 (2010) pp. 223–235.

[9] Myoung-Cheol Shin, Se-Hyeon Kang, and In-Cheol Park, "An Area-Efficient Iterative Modified-Booth

Multiplier Based on Self-Timed Clocking," Industry, and Energy through the project System IC 2010,and by IC Design Education Center (IDEC).

[10] Leandro Z. Pieper, Eduardo A. C. da Costa, Sergio J. M. de Almeida,"Efficient Dedicated Multiplication Blocks for2´s Complement Radix-2m ArrayMultipliers," JOURNAL OF COMPUTERS, VOL. 5, NO. 10, OCTOBER 2010.

