IMPLEMENTATION OF PARALLEL MULTIPLIER USING ADVANCED MODIFIED BOOTH ENCODING ALGORITHM

[1] **K. MURALI,**
ASSISTANT PROFESSOR, DEPT OF ECE,
VIJAYA INSTITUTE OF TECHNOLOGY FOR WOMEN, ENIKEPADU, AP, INDIA.
[2] **K. PRASUNA,**
ASSISTANT PROFESSOR, DEPT OF ECE,
VIJAYA INSTITUTE OF TECHNOLOGY FOR WOMEN, ENIKEPADU, AP, INDIA.

Abstract – A 2-bit Booth encoder with Josephson Transmission Lines (JTLs) and Passive Transmission Lines (PTLs) is designed. The Booth encoding method is one of the algorithms to obtain partial products. With this method, the number of partial products decreases down to the half compared to the AND array method. The circuit area of the multiplier designed with the Booth encoder method is compared to that designed with the AND array method. The proposed 64-bit modified booth encoders are designed using Modified Booth Algorithm and Carry Look Ahead Adder. The efficiency of this project is verified by successive execution with different inputs.

Keywords: Multiplier, booth encoder, Modified Booth Encoder, Carry Look Ahead Adder.

INTRODUCTION

Low power consumption in CMOS circuits is the present research as high integration causes increases the power dissipation. Modified Booth Encoders attract much attention because of low power dissipation and high throughput. The advantages of the Modified Booth Algorithm are the operation speed and the power dissipation. The AND array method and the Booth encoding method are algorithms for partial product generation. The reduction of partial products is required for higher-bit multiplication, because the addition of partial products stage occupies a large circuit area and causes huge delay. In general, a multiplier uses Booth encoder and array of full adders (FAs), or CLA tree instead of the array of FAs. This multiplier mainly consists of the three parts: Booth encoder, a tree to compress the partial products such as CLA tree, and final stage adder. CLA tree is to add the partial products from encoder as parallel as possible. In real implementation, many (4:2) compressors are used to reduce the number of outputs in each pipeline step. The most effective way to increase the speed of a multiplier is to reduce the number of the partial products because multiplication precedes a series of additions for the partial products. To reduce the number of calculation steps for the partial products, Modified Booth Encoder has been applied mostly where CLA tree has taken the role of increasing the speed to add the partial products. The AND array method and the Booth encoding method are algorithms for partial product generation. The reduction of partial products is required for higher bit multiplication, because the addition of partial products stage occupies a large circuit area. In this brief, we present a design technique that improves speed of Modified Booth Multiplier without a power penalty. Based on the proposed technique, the maximum speed of Modified Booth Multiplier can be increased by a great factor comparing with Booth Multiplier.

EARLIER IMPLEMENTATION

A. Modified Booth Encoder (MBE)

A modification of the Booth algorithm was proposed by Mac-Sorley in which a triplet of bits is scanned instead of two bits. This technique has he advantage of reducing the number of partial products by one half regardless of the inputs.

B. Modified Booth Algorithm

The multiplier has a 8-stage structure where the PPG block, the PPA block and the FSA block consist of 2 stages, 1 stage and 5 stages respectively.

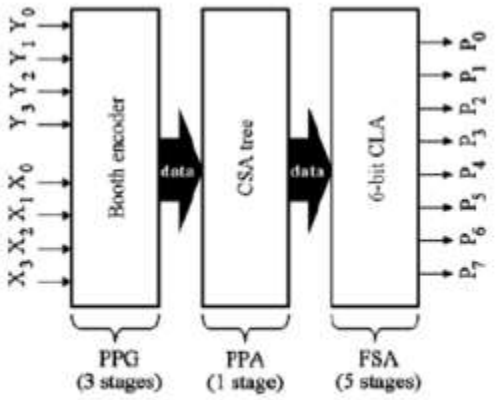


Fig. 1: Structure of Existing 4-bit Parallel Multiplier

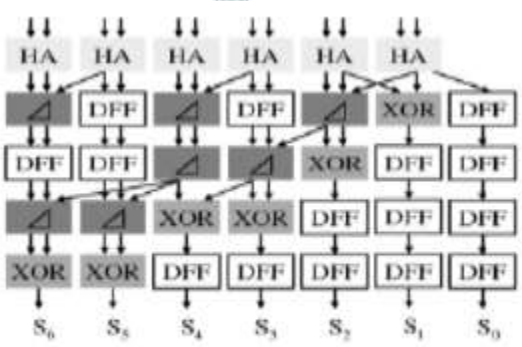


Fig.2: The Structure of Carry Look ahead Adder

When the number of partial products is reduced to sum and carry words, a final adder is required to generate the multiplication result. The number of bits of the final adder is the sum of the number of bits of the multiplier and multiplicand. Thus, the data path width is usually doubled and the delay of this stage is most severe. Normally 6-bit CLAs can be used to reduce the delay and area requirements. This adder is a practical design with reduced delay at the price of more complex hardware.

SYSTEM DESCRIPTION

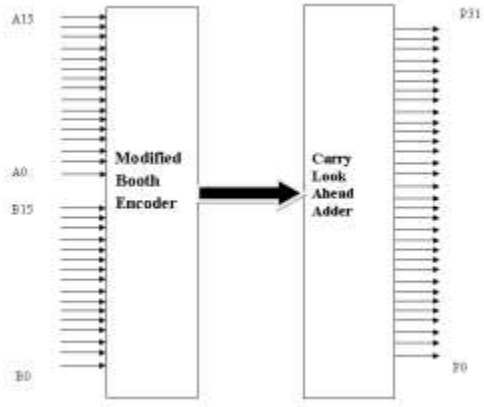


Fig. 3: 64 -bit Parallel Multiplier

Modified Booth Algorithm:

Multiplication consists of three steps: 1) the first step to generate the partial products; 2) the second step to add the generated partial products until the last two rows are remained; 3) the third step to compute the final multiplication results by adding the last two rows. The modified Booth algorithm reduces the number of partial products by half in the first step. This project has used the modified Booth encoding (MBE) scheme proposed in [2]. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, - 1, 0, 1, 2}. Table I shows the rules for obtaining the product. It is possible to reduce the number of partial products by half, by using the technique of radix 4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to

obtain the same results. Multiplication involves the generation of partial product, one of the each digit in the multiplier. These partial products are then summed to produce the final product. The partial products are easily defined. When the multiplier bit is 0, the partial product is 0. When the multiplier is 1, the partial product is the multiplicand. The total product is produced by summi.ng the partial products. For this operation, each successive partial product is shifted one position to the left relative to the preceding partial product. The multiplication of two n-bit binary integers, results in a product of up to 2^n bits in length.

Table 1: Processing of Modified Booth Multiplier

Q1	Q0	Q-1	Action
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

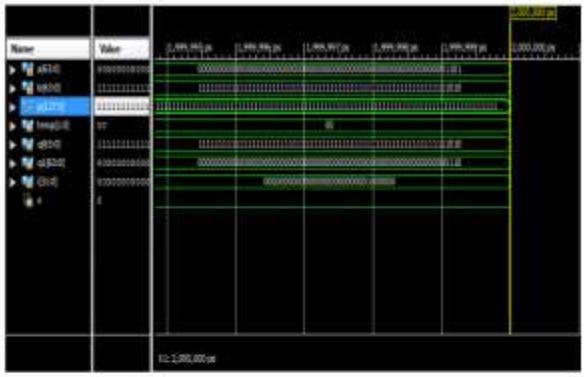
SIMULATION RESULT

8-Bit Booth Multiplier

16-Bit Booth Multiplier

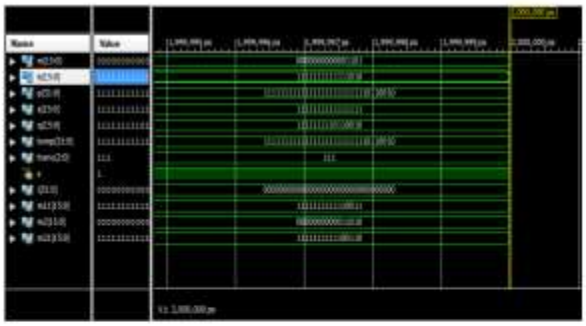
32-Bit Booth Multiplier

64-Bit Booth Multiplier



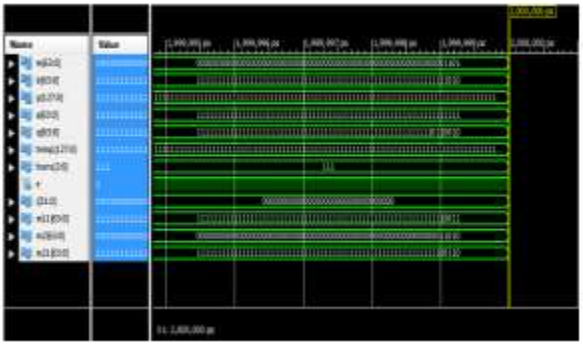
8-Bit Modified Booth Multiplier

16-Bit Modified Booth Multiplier



32-Bit Modified Booth Multiplier

64-Bit Modified Booth Multiplier



RESULTS AND DISCUSSIONS

Comparison of Time Delay for Booth and Modified Booth Multiplier

Multiplier Type	No. of bits	Time delay(ns)
Booth Multiplier	8 Bit	8.45
	16 Bit	17.06
	32 Bit	33.55
	64 Bit	66.52
Modified Booth Multiplier	8 Bit	6.62
	16 Bit	16.51
	32 Bit	27.92
	64 Bit	51.18

Table 2: Comparison of time delay for Booth and Modified Booth Multipliers

By observing above comparison it is clear that Modified Booth Multiplier takes less time for calculation than that of Booth Multiplier for 8-bit, 16-bit, 32-bit and 64-bit. So Modified Booth Multipliers are more efficient compared to Booth Multiplier.

FUTURE SCOPE AND CONCLUSION

The 64- bit multiplication is executed successfully and got the correct results. The project has reduced the number of partial products considerably, and increased the speed of execution. The program has given correct results for number of times. The project code is much efficient in terms of number of bits and in reducing the amount complexity accomplished. This project can be further modified to obtain the results for the numbers with the numbers of bits greater than 64. They can also be used for designing SFQ logic circuits.

REFERENCES

[1] 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.

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

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

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

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

[6] J. Sklansky (1960). "Conditional-sum addition logic," IRE Trans. Electron. Comput., vol. 9, pp. 226–231.

[7] L. Rubinfield (1975). "A proof of the modified booth's algorithm for multiplication,"IEEE tans. Compt., vol. C-24, no. 10, pp. 1014–1015.

[8] M. J. Schulte and E. E. Swartzlander (1993). Jr "Truncated multiplication with correction constant," VLSI Signal Processing VI, pp. 388–396.

[9] Pezaris S. D. (1971). "A 40ns 17-bit by 17-bit Array Multiplier", IEEE Transactions on Computers, VOL C-20, pp. 6-7.

[10] Srimani P. K. (1981). "Generalised proof of modified Booth"s algorithm", Comput. &Elect. Eng., pp. 7-9.