Journal of Computer Science & Computational Mathematics, *Volume 11, Issue 3, September 2021*
DOI: 10.20967/jcscm.2021.03.003

Implementation of Modified Booth-Wallace Tree Multiplier in FPGA

Anis Shahida Mokhtar*, Nurlisa Zahari, Chew Sue Ping, Muhazam Mustapha, Norlaili Ismail, Ahmad Sani Ismail

Department of Electric and Electronic Engineering, National Defence University of Malaysia, Sungai Besi Camp, 57000 Kuala Lumpur
*Corresponding author email: anis@upnm.edu.my

***Abstract*:** The main purpose of this paper is to present the design and implementation of the Modified Booth Algorithm introduced by Andrew Donald Booth in 1950 and the Wallace tree structure is by an Australian Computer Scientist Chris Wallace in 1964. Combination of both algorithms is to implement a versatile algorithm widely used for digital signal processing application. Due to the highly demand on the fast microprocessors, designers come out with multiple techniques to produce a high speed multipliers. Modified Booth's Algorithm have the advantages on faster multiplication process by reducing the generation of the partial products half of the number of bits of the multiplicand. The Wallace-tree multiplier itself giving a speed up in additional stage by reducing the adding partial products using the half adder and full adder instead of the long AND gate thus minimize the complexity of circuit. Combination of these two algorithms producing a new architecture of a high speed and low implementation area in one multiplier. This fulfil the requirement of high speed computer system nowadays. The algorithm was developed using Verilog HDL in Quartus II software and the result obtained from Modelsim-Altera then the design is implemented in FPGA DE2 Cyclone II to verify the result.

***Keywords*:** Booth multiplier, modified Booth multiplier, wallace tree, FPGA.

1. Introduction

Multipliers arithmetic algorithm plays important role in the performance of digital signal processing algorithm [1]. Many types of multipliers designed to match the requirement of high speed data processing [2]. Multiplication is basically an addition of the multiplicand itself to number of time of the multiplier generating levels of partial products. The critical path is determined more by the multiplier Next, product is formed by the additional of the partial products. All multipliers would having this three operation stages which is the generation of partial products, the additional of partial products and the final addition stage [3].

For this project, Modified Radix-4 Booth Algorithm as the multiplier giving full advantages in reducing the multiplication into half. A simple encoder are reduced from many operations in conventional multipliers [4] compared to 4 operations of Radix-4. The speed of multiplication can be increased by reducing the number of partial products and accelerating the accumulation of partial products. From many multipliers design of implementing high speed parallel multipliers, Booth Algorithm and Wallace-tree structure are an efficient implementation of a high speed parallel multiplier [5].

1.1 Modified Booth Algorithm

Following the standard add-shift operation of multiplier, adding the multiplicand number of times to multiplier in partial products. Big number of multiplicand increase the total partial products to be added for those large multiplier. Booth Algorithm design will be reducing the number of multiplicand multiples.

Multiplier						0	1	0	1	0	1
Multiplicand						0	0	1	0	1	0
					0	0	0	0	0	0	0
				0	1	0	1	0	1		
			0	0	0	0	0	0			
		0	1	0	1	0	1				
	0	0	0	0	0	0					
0	0	0	0	0	0						
0	0	0	1	1	0	1	0	0	1	0	

Figure 1. Conventional multiplier operation

Multiplier						0	1	0	1	0	1
Multiplicand						0	0	1	0	1	0
						1		1		1	
0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	1	0	1	0		
0	0	0	0	1	0	1	0				
0	0	0	0	1	1	0	1	0	0	1	0

Figure 2. Modified Booth algorithm operation

A modification is made in Figure 3 from the Conventional Booth Multiplier from Figure 1. For the purpose in this paper focusing Modified Radix-4 Booth Algorithm by taking groups of three bits at a time. It starts with LSB. The first block comprises only two bits of the multiplier and it assumes zero for the third bit. For multiplier to compare with the rules of encoded signal [6] in Table 1 to generate the partial products. Booth re-coding encodes multiplier bits into $[-2, 2]$.

Radix-4 Booth algorithm is given below:

1. Extend the sign bit 1 position if needed to confirm that n is even.

2. Add on a 0 to the right of the LSB of the multiplier.

Table 1. Modified Radix-4 encoding table

Multiplicand Bits, Y			Recorded Operation on Multiplier, X
Yi-1	Yi	Yi+1	
0	0	0	0X
0	0	1	+X
0	1	0	+X
0	1	1	+2X
1	0	0	-2X
1	0	1	-X
1	1	0	-X
1	1	1	0X

3. According to the value of each vector, each Partial Product will be 0, +X, –X, +2X or –2X.

4. The negative values of X are made by taking the 2's complement.

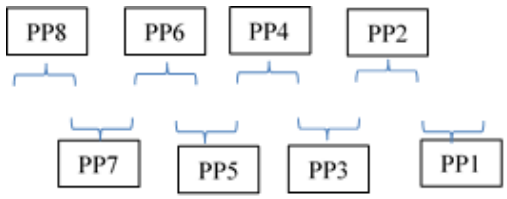

Figure 3. Radix-4 re-coding scheme

For Example:

Multiplier, X = 0000000000111100 (60)

Multiplicand, Y = 0000000100101100 (150)

PP1, PP4 = 100 encode -2X which equal to multiplier bits times -2. This is obtained by shifting 1 bit of the multiplier bits and 2's compliment multiplier bits.

X = 0000000000111100

2X = 0000000001111000

-2X = 1111111110001000

PP2 = 011 +2X which equal to multiplier bits times 2. This is obtained by shifting 1 bit of the multiplier bits.

X = 0000000000111100

2X = 0000000001111000

PP3, PP5 = 010 encode +X which equal to multiplier bits times by 1. For this encode just take the value of the multiplier.

X = 0000000000111100

PP6, PP7, PP8 = 000 encode 0X which equal to multiplier bits times 0.

X = 0000000000111100

0X = 0000000000000000

After doing the operations and obtaining the partial products, the additions of partial products are done to obtain the answer.

In designing Modified Booth multiplier in this project are using 16 bits of multiplier and multiplicand. Which resulting in 16 partial products for conventional multiplier but reduced to half in this Modified Booth Algorithm leaving only 8 partial products to be added later.

Multiplier	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
Multiplicand	0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0
PP1	1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0
PP2	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
PP3	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
PP4	1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0
PP5	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
PP6	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
PP7	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
PP8	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Result	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0

Figure 4. Modified Radix-4 calculation

1.2 Wallace Tree Structure

The Wallace tree structure proposed by Wallace in 1964 [7]. The idea to speed up the additional stage by reducing the partial products to be added by taking a group of three rows of partial products. The additional of partial products generated from the Modified Booth Algorithm [1] can be added in sequence for standard additional operation. This logic is easy to implement but it this method causing large delay of time.

Thus, for this algorithm, the partial products are solved by using the Wallace tree structure method. The Wallace tree speed up the additional stage by reduced the level or partial product leaving two rows to be added on the last stage. The final results of multiplications are from the addition of the last two rows of partial products.

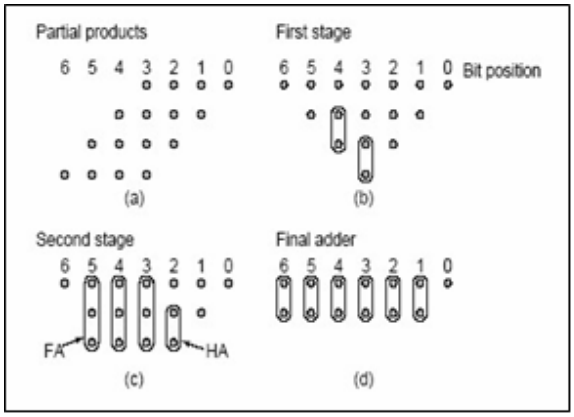

Figure 5. Additional steps of partial products in wallace tree

Figure 5 shows steps to implement Wallace tree structure for additional of partial products. First is the generation of the partial products which is from the Modified Booth Algorithm multiplier. Next the bits of partial products are reduced to rows by taking group of three rows to produce next rows of sum and products. Three bits signal are passed to a one [8] bit full adder input Wallace Tree circuit, and the output signal is supplied to the next stage full adder of the same bit position producing sum and carry. The new rows of sums and carries will be repeated until the last two rows of sum and carry left. Last step is the architecture of adders to sum up the last two rows for the final products [9].

2. Results and analysis

The algorithm was developed using Verilog HDL in Quartus II and simulated in Modelsim-Altera.

Figure 6 shows the simulation result for 60×150 by taking three types of multiplier using Modelsim-Altera to analyse the delays of the output

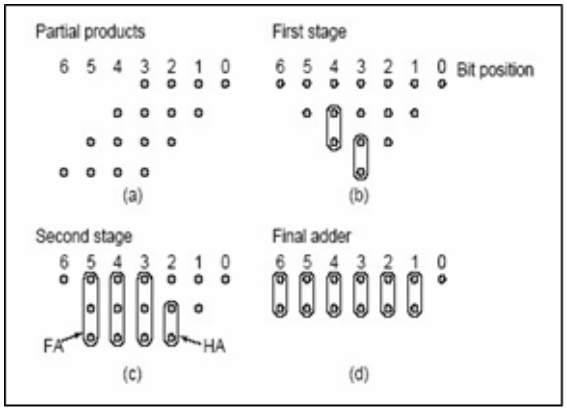

Figure 6. Simulation result from Modelsim

The result for simulation were recorded in Table 3 to compare the difference delay from the designed Modified Booth-Wallace with the conventional multiplier.

Figure 6 and Table 3 showed the result when Modified Booth Wallace multiplier implement into FPGA DE2 board. 16 bits input binary used onto the switches and product of 32 bits binary shown on 7 segments in Hexadecimal.

Modified Booth-Wallace multiplier algorithm run using Quartus II software then simulated on Modelsim-Altera to see the delay. The delay results are taken to analyze the speed of the multiplier to generate output of the multiplication as shown in Figure 6.

From the results of the simulation on Table 4, Modified Booth-Wallace multiplier had faster processing time compared to conventional multiplier, Modified Booth and Wallace tree structure itself. Modified Booth Algorithm multiplier reduced the number of partial products contributes to the less area of logic gates implement in circuit while Wallace structure lessen the circuitry structure of conventional adders thus both algorithm when combined producing one fast multiplier because of the less logic gates used in the multiplier to be implement in circuits.

tiplier, Modified Booth-Wallace algorithm was chosen as the fastest multiplier for Digital Signal Processing.

3. Conclusion

The Modified Booth-Wallace multiplier generates $n/2$ partial products hence decreasing the number of partial products generated with Wallace structure reducing the additional process of partial products. Combination of both algorithms producing a high speed multiplier with less area implementation on circuit. The simulation results and analysis are implemented using Altera Quartus II and Modelsim. The hardware implementation of the multiplier is implemented using FPGA DE2 Altera Cyclone II board. Taking the interest in a high speed and reduced area with power consumption mul-

References

[1] M. Othman, M. M. Ali *et al.*, "High performance parallel multiplier using wallace-booth algorithm," in *ICONIP'02. Proceedings of the 9th International Conference on Neural Information Processing. Computational Intelligence for the E-Age (IEEE Cat. No. 02EX575)*. IEEE, 2002, pp. 433–436.

[2] A. Chandrakasan, W. J. Bowhill, and F. Fox, "High-speed vlsi arithmetic units: Adders and multipliers," 2001.

[3] S. Asif and Y. Kong, "Performance analysis of wallace and radix-4 booth-wallace multipliers," in *2015 Electronic System Level Synthesis Conference (ESLsyn)*. IEEE, 2015, pp. 17–22.

[4] S.-R. Kuang, J.-P. Wang, and C.-Y. Guo, "Modified booth multipliers with a regular partial product array," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 5, pp. 404–408, 2009.

[5] M. J. Rao and S. Dubey, "A high speed and area efficient booth recoded wallace tree multiplier for fast arithmetic circuits," in *2012 Asia Pacific conference on postgraduate research in microelectronics and electronics*. IEEE, 2012, pp. 220–223.

[6] O. L. MacSorley, "High-speed arithmetic in binary computers," *Proceedings of the IRE*, vol. 49, no. 1, pp. 67–91, 1961.

[7] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on electronic Computers*, no. 1, pp. 14–17, 1964.

[8] A. Cooper, "Parallel architecture modified booth multiplier," in *IEE Proceedings G-Electronic Circuits and Systems*, vol. 135, no. 3. IET, 1988, pp. 125–128.

[9] J. Fadavi-Ardekani, "M* n booth encoded multiplier generator using optimized wallace trees," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 2, pp. 120–125, 1993.

[10] G. C. Ram, D. S. Rani, R. Balasaikesava, and K. B. Sindhuri, "Design of delay efficient modified 16 bit wallace multiplier," in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, 2016, pp. 1887–1891.

[11] N. Sharma and R. Sindal, "Modified booth multiplier using wallace structure and efficient carry select adder," *International Journal of Computer Applications*, vol. 68, no. 13, 2013.

Table 2. Calculation of Modified Booth Wallace Multiplier

Multiplier																0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
Multiplicand																0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0
PP1	1														1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0
PP2													0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0		
PP3											0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0					
PP4								1	1	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0							
PP5	2						0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0									
PP6						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
PP7	3			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
PP8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
sum1											0	0	0	1	1	1	1	1	1	1	0	1	1	0	1	0	1	0	0	0	
carry1											0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0		
sum2						0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0						
carry2						0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0							
sum3			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
carry3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
sum4						0	0	0	1	1	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	0
carry4						0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	
sum5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0							
carry5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
sum6						0	0	0	1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	0	0	1	0	1	0	0	0
carry6					0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	
carry5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
prod_sum			0	0	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0
prod_carry		0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0		
result		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0

Table 3. Simulation Result

Type of Multiplier	Delay (ps)	Difference delay (ps)
Conventional Multiplier *(mult)*	18117	-
Booth Multiplier *(check)*	24084	5697
Modified Booth Multiplier *(prod)*	21847	3730

Table 4. Comparison Types of Multiplier

	Conventional Multiplier [10]	Modified Booth Algorithm [11]	Wallace tree structure [10]	Modified Booth-Wallace
Delay (ns)	61.39	24.23	36.350	18.23
No of LUTs	844	1292	1000	713