Jurnal Kejuruteraan SI 4(1) 2021: 161-165
https://doi.org/10.17576/jkukm-2021-si4(1)-20

Implementation of Booth Multiplier Algorithm using Radix-4 in FPGA

(Pelaksanaan Algoritma Pendaraban Booth menggunakan Radix-4 di dalam FPGA)

Anis Shahida Mokhtar[a,*], Chew Sue Ping[a], Muhamad Faiz Md Din[a], Nazrul Fariq Makmor[a] &
Muhammad Asyraf Che Mahadi[b]

[a]*Department of Electrical and Electronics Engineering, Faculty of Engineering, National Defence University of Malaysia, Sg. Besi Camp, 57000 Kuala Lumpur, Malaysia*

[b]*Royal Signal Regiment, Malaysia Armed Forces, Kuala Lumpur*

[*]*Corresponding author: anis@upnm.edu.my*

Received 25 March 2021, Received in revised form 13 April 2021
Accepted 30 August 2021, Available online 30 September 2021

ABSTRACT

This paper presents the performance of Radix-4 Modified Booth Algorithm. Booth algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. Multiplier is a fundamental component in general-purpose microprocessors and in digital signal processors. With advances in technology, researchers design multipliers which offer high speed, low power, and less area implementation. Booth multiplier algorithm is designed to reduce number of partial products as compared to conventional multiplier. The proposed design is simulated by using Verilog HDL in Quartus II and implemented in Cyclone II FPGA. The result shows that the average output delay is 20.78 ns. The whole design has been verified by gate level simulation.

Keywords: Booth multiplier; Radix 4; FPGA; digital arithmetic

ABSTRAK

Kertas kerja ini membentangkan prestasi Radix-4 Algorithma Booth yang diubah suai. Kerangka algoritma ini adalah algoritma pendaraban yang menggandakan dua nombor binari yang ditanda aras dalam notasi pelengkap dua. Secara umum, pengganda merupakan komponen utama di dalam mikropemproses dan pemproses isyarat digital. Dengan kemajuan teknologi terkini, penyelidik merekabentuk pendarab yang berkelajuan tinggi, penggunaan kuasa yang rendah, dan hanya memerlukan ruangan kerja yang kecil. Algoritma Pendaraban Booth ini juga direkabentuk untuk mengurangkan nombor dari hasil darab separa berbanding pendaraban sedia ada. Rekabentuk yang dicadangkan telah disimulasikan menggunakan Verilog HDL di dalam Quartus II dan dilaksanakan ke atas Cyclone II FPGA. Hasil simulasi menunjukkan purata masa kelewatan keluaran adalah 20.78 ns. Keseluruhan rekabentuk telah disahkan melalui simulasi logik gate.

Kata kunci: Kerangka pengganda; Radix-4; FPGA; aritmatik digital

INTRODUCTION

Multipliers are used in many different places in microprocessor design. Multipliers are also very important and frequently used in Digital Signal Processing (DSP) to run complex high-speed calculations. Booth multiplication is used greatly to increase the speed of the multiplier by encoding the numbers that are multiplied. There are many types of Modified Booth Algorithm (MBA) multiplier using different radix. The purpose of MBA Radix-2 is to reduce the partial product of the multiplication operation and the MBA Radix-4 is used to cut the partial product of MBA Radix-2 by half (Kaur, S. & Manna, M.S. 2013). Booth algorithm was created by A.D Booth (Booth, A.D. 1951) forms the signed number's base multiplication algorithm that are simple to fulfill at hardware level and have potential to speed up signed multiplication (Booth, A.D. 1951). Booth's algorithm contained the multiplier y, value z, leaving multiplicand. The way to express the signed digits, there is a special notion that is called signed digit (SD). In (SD) encoding +1 and 0 are expressed as 1 and 0. On the other hand, -1 is expressed as 1. The operation must follow the operation table called Booth recoding for Radix-2 (Dhumal, A.K. & P.S.S.S. 2016).

BOOTH ALGORITHM MULTIPLIER RADIX-4

With a specific end goal to cut the quantity of halfway items considerably, the Radix-4 booth encoding was made. In radix-4 operation, it observes three bits at a time by using overlay method. It begins with LSB. Modified Booth multiplication is the strategy that takes into consideration littler, speedier circuit by recoding the numbers which are increased. It gives significant change over expanded multiplication procedure (Thomas& Tech 2014).

TABLE 1. Operation table for Booth recoding for Radix-2

a	b	Partial product
0	0	0
0	1	M
1	0	-M
1	1	0

METHODOLOGY

In order to decrease the number of partial products by half, MBA Radix-4 encoding was formed. Radix-4 method compare three bits of multiplicand at time using overlap method. It starts with LSB of multiplicand. The first overlap comprises of only two bits of the multiplier and it assumes zero for the third bit. The operation must refer to the encoding table of Radix-4 as shown in Table 2. MBM multiplication is the technique that lets for reduced, quicker circuit by recoding multiplication of the number. It provides major improvement over long multiplication technique. Figure 1 shows the flow chart of the Radix-modified Booth algorithm multiplier.

MODIFIED BOOTH ALGORITHM MULTIPLIER RADIX-4

In order to decrease the number of partial products by half, MBA Radix-4 encoding was formed. Radix-4 method compare three bits of multiplicand at time using overlap method. It starts with LSB of multiplicand. The first overlap comprises of only two bits of the multiplier and it assumes zero for the third bit. The operation must refer to the encoding table of Radix-4 as shown in Table 2. MBM multiplication is the technique that lets for reduced, quicker circuit by recoding multiplication of the number. It provides major improvement over long multiplication technique. Figure 1 shows the flow chart of the Radix-modified Booth algorithm multiplier.

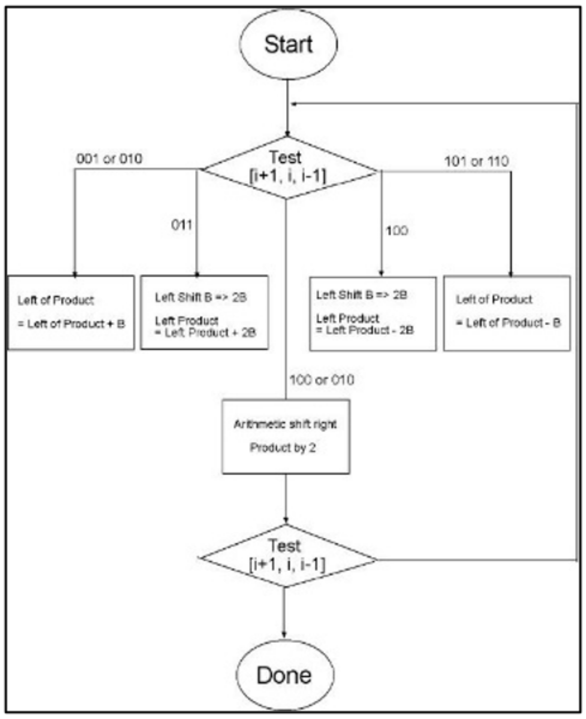

Start
Test [i+1, i, i-1]
001 or 010
011
100
101 or 110
Left of Product = Left of Product + B
Left Shift B => 2B
Left Product = Left Product + 2B
Left Shift B => 2B
Left Product = Left Product - 2B
Left of Product = Left of Product - B
100 or 010
Arithmetic shift right Product by 2
Test [i+1, i, i-1]
Done

FIGURE 1. Flowchart of Radix-4 modified Booth algorithm multiplier

TABLE 2. The encoding table of Radix-4

Multiplicand bits			Partial Product
i+1	i	i-1	PP
0	0	0	0
0	0	1	m
0	1	0	m
0	1	1	2m
1	0	0	-2m
1	0	1	-m
1	1	0	-m
1	1	1	0

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 0 to LSB of the multiplicand.

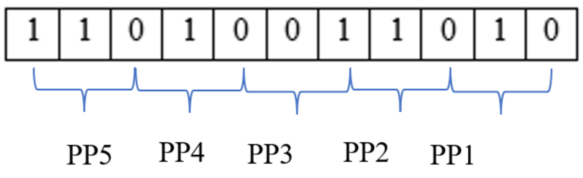
3. According to the value of each vector, each Partial Product will be 0, +m, –m, +2m or –2m which m is multiplier.

RADIX-4 HAND CALCULATION

For example, Multiplier (M) is 10 and Multiplicand (Q) is -179.

10(M)	=	01010
-M	=	10101(1's complement)
	+	1
		10110 (2's complement)
-179 (Q)	=	11010011010

The operation must refer to Q. The 0 must be added at the LSB. So, the Q becomes 11010011010.



By using overlap of three bits method, the partial products are:

PP1 =	010	(M)
PP2 =	110	(-M)
PP3 =	001	(M)
PP4 =	010	(M)
PP5 =	110	(-M)

Then, refer the partial product bits from Table 2.

			0	0	0	0	0	0	0	1	0	1	0	M
		×	1	1	0	1	0	0	1	1	0	1	0	Q
	0	0	0	0	0	0	0	0	0	1	0	1	0	PP1
	1	1	1	1	1	1	1	0	1	1	0			PP2
	0	0	0	0	0	1	0	1	0					PP3
	0	0	0	1	0	1	0							PP4
+	1	0	1	1	0									PP5
1	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**-1790**

RESULTS AND DISCUSSION

From Figure 2, it shows the input a as the multiplicand and b as a multiplier. It shows that there are 10 inputs and outputs. The delay between input and output of every operation are in range of 19 ns to 22 ns due to the sequence of the input and output. As a conclusion, the delay times are in range 19 ns to 22 ns and average of delay is 20.78 ns. This algorithm also implements in FPGA for hardware testing. Figure 3 shows the output in FPGA Altera DE2 Cyclone II. After the Radix-4 Verilog are downloaded to the board through JTAG programmer, the output can be shown through the seven-segment section of the board. Table 4 shows the input and output of Radix-4 implemented to the FPGA board.

TABLE 4. Input and output of Radix-4 on FPGA board

No	Input		Output	Relay
	a	b		
1	5500	-15	-82500	20.64
2	345	-27	-9315	19.32
3	432	-564	-243648	20.48
4	446	-43	-19178	22.50
5	786	-56	-44016	20.94
6	124	-77	-9548	21.25
7	45	-882	-39690	22.21
8	345	-23	-7935	19,33
9	113	-53	-5989	20.50
10	196	-74	-14504	20.29
	Average			20.78

The input of multiplier is 10 and the value of the multiplicand are fixed to -179. Based on Figure 3, the input is 10 which is 1010 in binary are shown using switch. Output shown in seven segment FFFFF902 in Hexadecimal.

From the result in Table 3, it shows that the delay time taken for MBM Radix-4 to operate is in the range of 19 ns to 22 ns and the average of the 10 samples operation is 20.78 ns. The processing time of MBM Radix-4 is faster than MBM Radix -2 due to the partial product of MBM Radix-4 is half of MBM Radix-2. The difficulties to run this project, firstly is to construct the Verilog HDL code of MBM Radix-4. This is because, we must have the knowledge of the operation of the MBM Radix-4 beforehand and the next step is to learn about the module of the operation. It takes time to learn the module of the operation in MBM Radix-4.

Secondly is to implement the Verilog codes to the FPGA board. Before the Verilog codes are ready to be implemented, it must be free from any errors. If the codes are not successful and there are errors, it will refer back to the Verilog HDL code. If the code has errors, the lines of the Verilog HDL code must be checked and constructed until it is free from errors. After the Verilog HDL code is successful, pin assignment of the input refers to the FPGA pin table. Then, the code must be downloaded through USB Blaster and JTAG mode. The difficulty happened when the USB Blaster was not detected by the PC because the USB Blaster was not updated.

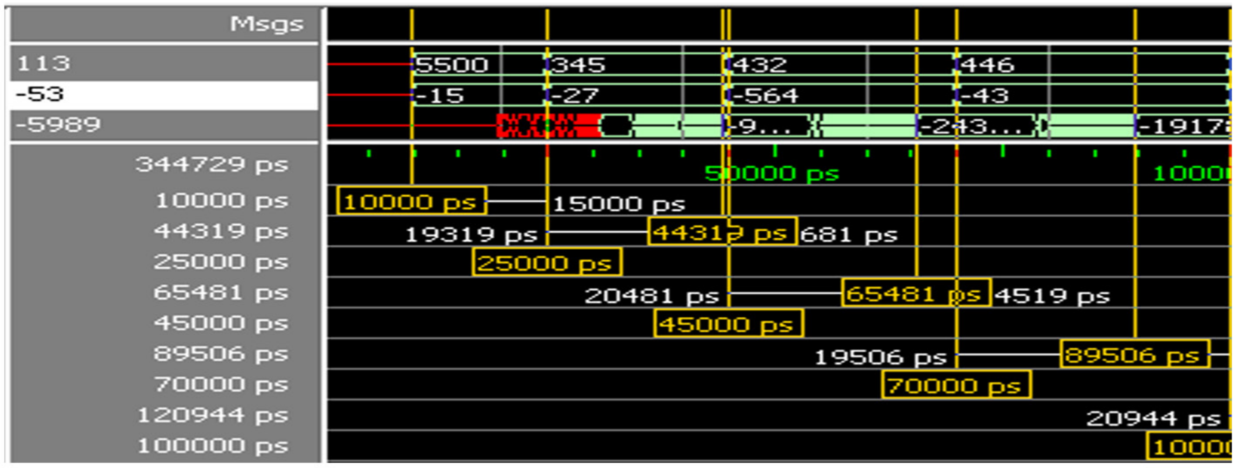


FIGURE 2. Modelism output of Radix-4

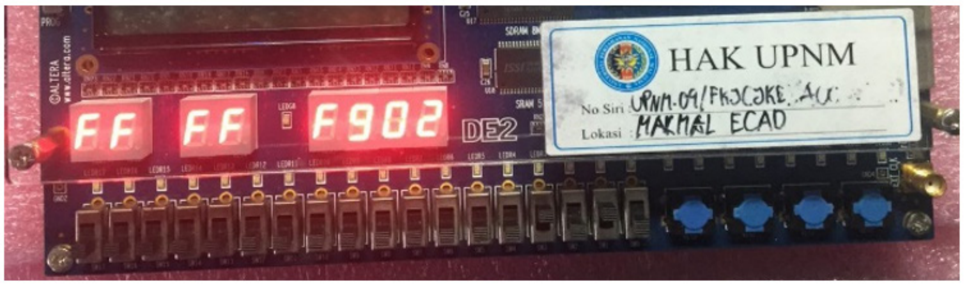


FIGURE 3. Radix-4 Booth algorithm on FPGA

CONCLUSION

In conclusion, Radix-4 modified Booth algorithm was successfully built using Verilog HDL and implemented in FPGA. The simulation was realized using Quartus II. The multiplier reduces the partial product to n/2 which contributes to the decrease of the delay.

DECLARATION OF COMPETING INTEREST

None

REFERENCES

Booth, A.D. 1951. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics* 2: 236-240.

Brown, S. & Rose, J. 1996. Architecture of FPGAs and CPLDs: A tutorial. IEEE Des. *Test Comput*. 13(2): 42–57.

Catanzaro, B.C. 2005. Department of Electrical and Computer Engineering Brigham Young University. Higher Radix Floating-Point Representations for FPGA-Based Arithmetic.

Dhumal, A.K. & P. S. S.S. 2016. Comparison between Radix-2 and Radix -4 based on Booth algorithm. *Ijarcce* 5(12): 498–500.

Tech, A.R.V.M. & Sri, M.S.S. 2014. Design and implementation of FPGA Radix-4 Booth multiplication algorithm. International Journal of *Research in Computer and Communication Technology* 3(9): 1067–1074.

Feedback, S. 2016. MAX 10 FPGA Device Architecture.

Kaur, S. & Manna, M.S. 2013. Implementation of Modified Booth Algorithm (Radix 4) and its comparison with Booth Algorithm (Radix-2). Adv. Electron. Electr. Eng. 3(6): 683–690.

Kelly, L.S.S. 2012. Performance comparison review of Radix-based multiplier designs. 4th International Conference on Intelligent and Advanced Systems 854-859.

Thomas, M. & Tech, M. 2014. Design and simulation of Radix-8 Booth encoder multiplier for signed and unsigned numbers. *IJIRSTV1I1008* 1(1): 1–10.