VOL. 10, NO. 17, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

MODIFIED BOOTH MULTIPLIER ARCHITECTURE USING NEW (1, 1, 1) ADDER

Anu Mehra, Priyank Kularia, Aditya Sharma, Garima Batra, Achintya Rawat and Nidhi Gaur
Amity School of Engineering and Technology Amity University, Noida, India
E-Mail: amehra@amity.edu

ABSTRACT

In this paper an alternate implementation of the modified Booth algorithm is presented where groups of the partial product terms are summed using parallel prefix adders proposed by Harris *et al*. Comparative analysis of these adders in terms of power, delay and LUTs is performed. A modified 16 bit multiplication process using Radix 4 Booth Algorithm is proposed and results with respect to Kogge Stone and New (1, 1, 1) adder are computed. Simulation results are carried out on Xilinx Vivado Version14.2 on Artix 7 Board.

Keywords: Kogge Stone Adder, modified booth architecture, adder.

INTRODUCTION

Multiplier

Multipliers are essential hardware for designing digital circuits. These systems include digital signal processors, microprocessors etc. High speed and low power consumption are the most important attributes required for high performance of multipliers [1]. Booth multiplication algorithm uses small number of additions and shift operations to perform multiplication [2]. Therefore, Booth multipliers have been a part of High speed multipliers and several modifications and applications have been proposed in literature [3].

Radix 2 involves scanning of 2 bits at a time for multiplication. The pioneer version of Booth's multiplier (Radix-2) has two limitations:

1. Fluctuation in the number of add or subtract operations is observed.
2. The algorithm becomes inconvenient in the case of isolated 1's.

In Radix-4 algorithm [4], number of steps is halved for the calculation of partial products, since scanning of 3 bits at a time is performed as compared to Radix 2 where 2 bits are scanned at a time.

Parallel Prefix Adder

Parallel prefix adders [5] [6] take inputs $\{A_{N-1}, ..., A_0\}$, $\{B_{N-1}, ..., B_0\}$ and C_{IN}, and produce a sum output $\{S_N,\ldots S_I\}$ using intermediate Generate (G) and propagate (P) prefix signals. The addition logic consists of the following 3 stages: [7]

Pre-processing

This step involves computation of generate and propagate signals corresponding to each pair of bits in A and B. These signals are given by the logic equations below:

$P_i = A_i$ XOR B_i
$G_i = A_i$ AND $B_i = A_i B_i$

Carry Look-ahead network

This step involves computation of carries corresponding to each bit. It uses group propagate and generate as intermediate signals which are given by the logic equations below:

$P_0 = A_0$ XOR B_0
$G_0 = A_0$ AND $B_0 = C_0$
Sunsequent crries C_1, C_2, C_3 etc. are
$C_1 = G_0 + P_0 C_0$
$C_2 = G_1 + P_1 C_1$
$C_3 = G_2 + P_2 C_2$ etc. In general,
$C_i = G_{i-1} + P_{i-1} C_{i-1}$

Substituting C1, C2 etc. in subsequent equations gives
$C_2 = G_1 + P_1 G_0 + C_0 P_0 P_1$
$C_3 = G_2 + G_1.P_2 + G_0.P_1.P_2 + C_0.P_0.P_1.P_2$
$C_4 = G_3 + G_2.P_3 + G_1.P_2.P_3 + G_0.P_1.P_2.P_3 + C_0.P_0.P_1.P_2.P_3$
etc.

Post processing

This is the final step and is common to all adders of this family. It involves computation of sum bits. Sum bits are computed by the logic given below: $S_i = P_i$ XOR C_{i-1}
$P_{i:j} = P_{i:k+1}$ AND $P_{k:j}$
e.g. $P_{3:0} = P_3 P_2 P_1 P_0$
$P_{3:2} = P_3 P_2$
$G_{i:j} = G_{i:k+1} + (P_{i:k+1}$ AND $G_{k:j})$
e.g. $G_{3:0} = G_{3:2}$ OR $(P_{3:2}$ AND $G_{1:0})$
$= (G_3 + P_3 G_2) + (P_3 P_2 (G_1 + P_1 G_0))$

Parallel Prefix adders include two basic cells:

Black cell

As shown in Figure-1(a) this cell is used in the logic design of adders and it produces intermediate generate and propagate terms of present and previous states.

$G_{i:j}$= $G_{i:k}$ OR ($P_{i:k}$ and $G_{k-1:j}$)

$Pi_{:j}$= $P_{i:k}$ AND$P_{k-1:j}$

Grey cell

As shown in Figure-1(b) this cell is used in the logic design of adders and it produces intermediate generate of present and previous states.

$G_{i:j}$= $G_{i:k}$ OR ($P_{i:k}$ and $G_{k-1:j}$)

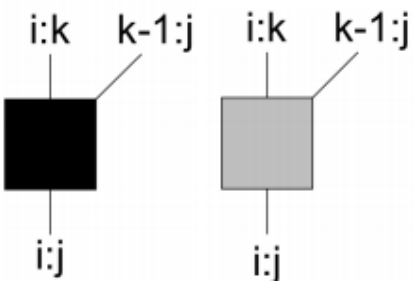

Figure-1. (a) Black cell; (b) Grey cell.

Contributions

We have proposed a modified Booth multiplier which reduces the delay to a greater extent for 16 bit multiplication. This makes the circuit faster and appealing to high speed digital electronic designs. The principle and working of modified approach is described in detail in Part II.

Organization of Paper

Paper comprised of five sections including introduction. Section II describes our technique Modified Booth Multiplier Architecture. In section III block diagram of our design is presented with explanation of each block. In section IV simulation results are analysed and compared with existing results. Conclusion of paper is discussed in section IV.

MODIFIED BOOTH ARCHITECTURE (MBA)

This includes dividing the multiplier into three digits correspondingly on the basis of the Booth encoder Table (Table-1). In Radix-4 booth encoder, first step is to append extra '0'to the Least Significant Bit (LSB) of the multiplier. When we add an extra '0' it becomes a 9 bit number. Now starting from the Least Significant Bit, three bits are consequently grouped e.g. first group is '0'and A_7 and A_6 and second group is A_6 and A_5 and A_4 where and represents concatenation. After checking the sequence from Table-1, the relevant encoding is chosen.

Table-1. Radix-4 booth algorithm encoding.

BN+1	BN	BN-1	Operation	1M	2M	3M
0	0	0	0	1	1	0
0	0	1	1 * M	0	1	0
0	1	0	1 * M	0	1	0
0	1	1	2 * M	1	0	0
1	0	0	-2 * M	1	0	1
1	0	1	-1 * M	0	1	1
1	1	0	-1 * M	0	1	1
1	1	1	0	1	1	0

BLOCK DIAGRAM OF MBA

MBA consists of six blocks as shown in Figure-2: Sign Extension Generator, Control Unit, and Partial Product Generator.

SIGN EXTENSION GENERATOR

It comprises of sign extension chip and sign control chip which is clock based. Sign extension corrector improves the capability of the booth multiplier by not only multiplying the unsigned numbers but also the signed numbers.

Sign Extension Control Unit (CHIP6) is used to control the result from CHIP5 and stores them into particular and correct order and then fed the result to partial product rearrange unit.

Table-2. Sign extension table when $a_{7=0/1.}$

A7		Bn+1	Bn	Bn-1	E(A7=0)	E(A7=1)
0	1	0	0	0	0	0
0	1	0	0	1	0	1
0	1	0	1	0	0	1
0	1	0	1	1	0	1
0	1	1	0	0	1	0
0	1	1	0	1	1	0
0	1	1	1	0	1	0
0	1	1	1	1	0	0

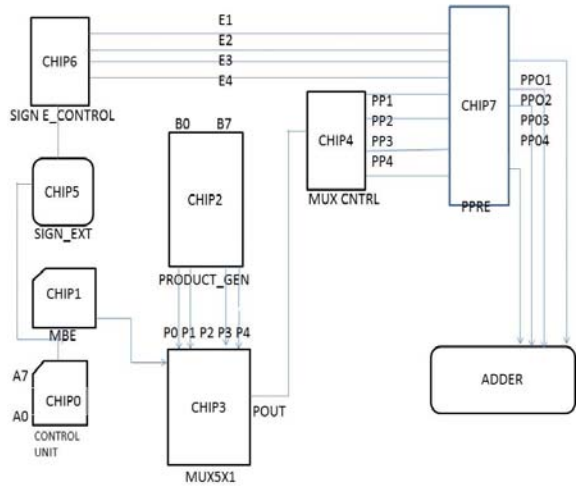

Figure-2. Block diagram of proposed multiplier.

CONTROL UNIT

The control unit (CHIP0) automatically fetches the combination form the multiplier and delivers it to MBE. MBE then decide on the base of result of control unit which operation is to be carried out.

PARTIAL PRODUCT GENERATOR

Partial product generator block consists of CHIP2, CHIP3 and CHIP4 of the MBA. Product generator (CHIP2) is structured to produce the multiple products of multiplicand which are 0*K, 1*K, 2*K, -1*K and -2*K. The product generator helps to generate the multiplicands by-product and to make decision of choosing partial product faster.

ADDER ARCHITECTURES

Kogge Stone Adder

Kogge stone adder (Figure-3) is a type of parallel prefix adder. It is a highly used adder in industries as it is considered the fastest adder which provides high performance. It gives lower fan-out. This helps in improving performance. The limitation of this adder is that it ingests large silicon area [8].

New(1,1,1) Adder

The parallel prefix designs can be sorted with the help of three dimensional distribution in l,f and t, where l represents the number of logic levels(logic levels = L+1), f means fanout ($2^f + 1$) and t corresponds to wiring tracks (2^l). This adder is a new member to the family of parallel prefix adders. Here l, f and t are greater than 0 [9].

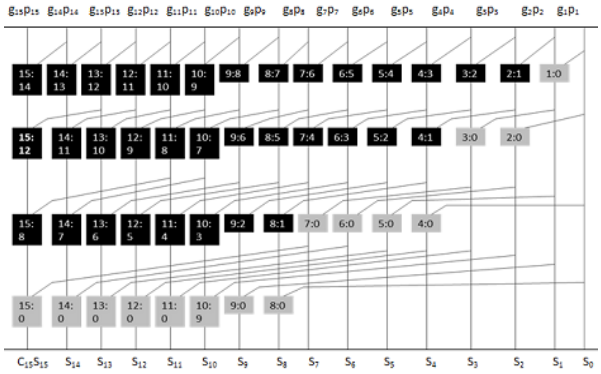

Figure-3. Kogge Stone Adder architecture.

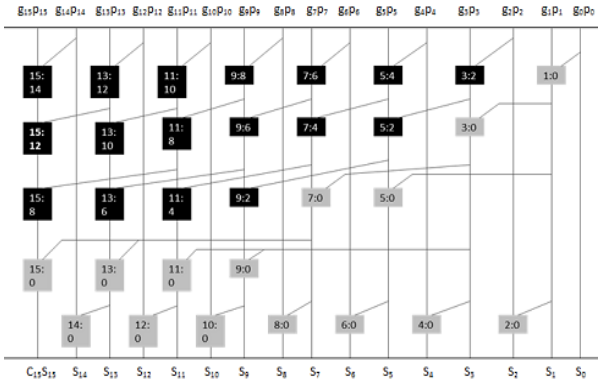

Figure-4. New (1, 1, 1) adder architecture.

SIMULATION RESULTS

Radix 4 Modified Booth Architecture (MBA) was implemented first using Kogge Stone which is more prevalent and then, using New (1, 1, 1) adder. The simulations were carried out in Xilinx VIVADO version 14.2 on Artix 7 Board. Comparative analysis between them is given in Table-2. Table-3 depicts the comparison of adder architectures which are implemented in the heart of Multiplier architectures. Simulation waveforms for Kogge Stone Multiplier architecture is given in Figure-5. Figure-6 gives simulation waveform of MBA multiplier.

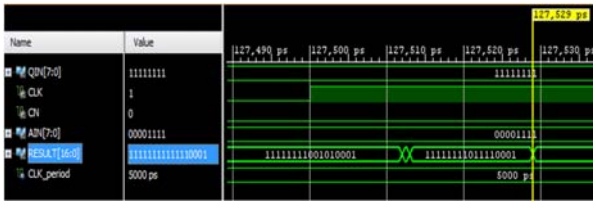

Figure-5. MBA using new (1, 1, 1) adder.

Table-3. Comparison of modified booth architecture using Kogge Stone and new (1, 1, 1) adder.

Multi with adder	LUTs	registers	Delay (ns)	Power (mW)		
				Total	Total	Total
m-Kogge stone	101	100	127.83	164	164	164
m-new(1,1,1)	92	100	127.52	165	165	165

Table-4. Comparison of Kogge Stone and new (1, 1, 1) adder architectures.

Adder	LUTs	registers	Delay (ns)	Power (mW)		
				Total	Total	Total
m-Kogge stone	46	50	5.916	163	43	120
m-new(1,1,1)	32	50	6.042	163	43	120

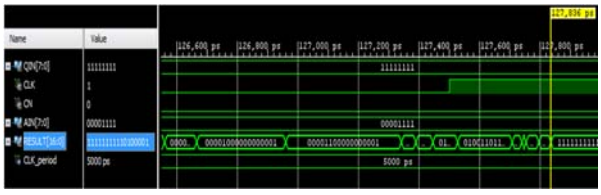

Figure-6. Multiplier using Kogge Stone Adder.

CONCLUSIONS

From simulation results as shown in Table-3, number LUTs used in implementation of MBA with New (1, 1, 1) Adder is about 9% lower as compared to Booth using Kogge Stone. The power consumption and delay are comparable in both cases with MBA being slightly faster. Implementation with parallel processing in larger CPUs etc. will lead to lesser area without degrading the results of Booth using Kogge Stone.

REFERENCES

[1] Jagadeswar Rao M, Sanjay Dubey, "A high speed and Area efficient Booth Recoded Wallace Tree Multiplier for fast Arithmetic Circuits", Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PRIMEASIA), 2012, pp. 220-223, 5th -7th December 2012.

[2] Rahul D Kshirsagar, Aishwarya E.V., Ahire Shashank Viswanath, P Jayakrishnan, "Implementation of Pipelined Booth Encoded Wallace Tree Multiplier Architecture. 2013 ICGCE, 2013, pp. 199-204.

[3] T. S. Kaur, Susman and M.S. Manna, "Implementation of modified Booth Algorithm (Radix 4) and its comparison with Booth Algorithm (Radix 2)," in Advances in Electronic and Electric Engineering. Vol. 3 No. 6, pp. 683-690.

[4] Hsin-Lei Lin, Design of a Novel Radix-4 Booth Multiplier, the 2004 IEEE Asia –pacific Conference on Circuit and Systems, December 2005.

[5] Y. Choi, "Parallel Prefix Adder Design," Proc. 17th IEEE Symposium on Computer Arithmetic, pp. 90-98, 27th June 2005.

[6] M. Snir, "Depth-size trade-offs for parallel prefix computation," in Journal of Algorithms. 7, pp. 185-201, 1986.

[7] R. Zmermann, Binary Adder Architectures for Cell-Based VLSI .and their Synihesis, ETH Dissertation 12480, Swiss Federal Institute of Technology, 1997.

[8] P. Kogge and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence relation," IEEE transactions on computers. Vol. C-22, no. 8, pp. 786-793, August 1973.

[9] I. Sutherland, R Sproull, and D. Harris, Logical Efori, San Francisco: Morgan Kauhann, 1999.