iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** I **Month of publication:** January 2018

DOI: http://doi.org/10.22214/ijraset.2018.1104

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor :6.887
Volume 6 Issue I, January 2018- Available at www.ijraset.com

High Speed Arithmetic Logic Unit

Rita Mahajan[1], Gourav Saini[2], Deepak Bagai[3]
[1]*Assistant Professor,* [2]*ME student,* [3]*Professor, Department of Electronics and Communication Engineering, PEC University of Technology, Chandigarh, India.*

Abstract: In an ALU , adder and multiplier are the major components which define the speed of an ALU. So speed of the ALU can be enhanced by using the high speed adder and multiplier. In this paper, a high speed ALU has been discussed in which basic high speed adders and multipliers have been used. Speed of the adder and basic multiplier has also been compared with the basic adder and multiplier.
Keywords-ALU-Arithmatic Logic Unit
CSA-Carry Save Adder
VC-Virtual Carry
VS-Virtual Sum

I. INTRODUCTION

An ALU is an electronic circuit that is used to perform arithmetic and logic operations. It is the basic component of an CPU. Since ALU is the basic component of any computer, its speed matters a lot if we need high speed operations. Since in an ALU, adding and multiplying are the most basic and time consuming operations. So, if we want to improve the speed of an ALU, we should use high speed adders and multipliers. In this ALU, Carry Save Adder and Radix-4 BOOTH Multiplier have been used for the high speed operations. Its results have been compared with ripple carry adder and a Vedic multiplier which uses the fast carry save adder for its operation. Although changing the transistor level circuitry of adder[7] or multiplier[4], its speed can be increased. But, conventional multipliers and adders have been used in this paper.

II. CARRY SAVE ADDER

The carry save addition of 2 N-bit numbers results in two (N + 1)-bit numbers being produced, the virtual carry (VC) and the virtual sum (VS).But after getting VC and VS you still have to add the two values together with a convectional adder to get your final result, so only adding 2 numbers is pointless. Take this example, let's say one carry-save addition takes k*T ms, where k = number of N - bit numbers being added, and a convectional adder takes 5T ms to add 2 numbers (regardless of bit width), then if:

1) 2 numbers are added, then
Time (Carry-Save) = 2T + 5T = 7T
Time (Convectional Adder) = 5T
2) 3 numbers are added, then
Time (Carry-Save) = 3T + 5T = 8T
Time (Convectional Adder) = 5T + 5T = 10T

So carry-save addition is only useful if you have at least 3 operands to add.

So, if external carry is also there. Carry Save Adder is quite faster than Conventional Ripple Carry Adder.

III. RADIX-4 BOOTH MULTIPLIER

To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier (since there is no previous block to overlap):

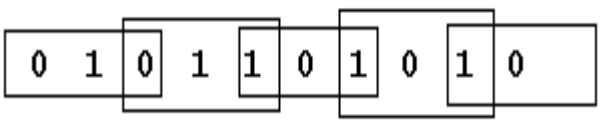


Figure 1.5: Grouping of bits from the multiplier term, for use in Booth recoding.

The least significant block uses only two bits of the multiplier, and assumes a zero for the third bit. The overlap is necessary so that we know what happened in the last block, as the MSB of the block acts like a sign bit. We then consult the table 2-3 to decide what the encoding will be.

Table 1.3 : Booth recoding strategy for each of the possible block values

Block	Partial Product
000	0
001	1 * Multiplicand
010	1 * Multiplicand
011	2 * Multiplicand
100	-2 * Multiplicand
101	-1 * Multiplicand
110	-1 * Multiplicand
111	0

Since we use the LSB of each block to know what the sign bit was in the previous block, and there are never any negative products before the least significant block, the LSB of the first block is always assumed to be 0. Hence, we would recode our example of 7 (binary 0111) as :

0 1 1 1

block 0 : 1 1 0 Encoding : * (-1)

block 1 : 0 1 1 Encoding : * (2)

In the case where there are not enough bits to obtain a MSB of the last block, as below, we sign extend the multiplier by one bit.

0 0 1 1 1

block 0 : 1 1 0 Encoding : * (-1)

block 1 : 0 1 1 Encoding : * (2)

block 2 : 0 0 0 Encoding : * (0)

The previous example can then be rewritten as:

0 0 1 0 1 1 , multiplicand

0 1 0 0 1 1 , multiplier

1 1 -1 , booth encoding of multiplier

1 1 1 1 1 1 0 1 0 0 , negative term sign extended

0 0 1 0 1 1

0 0 1 0 1 1

0 0 0 0 1 , error correction for negation

0 0 1 1 0 1 0 0 0 1 , discarding the carried high bit

One possible implementation is in the form of a Booth recorder entity, such as the one in figure , with its outputs being used to form the partial product:

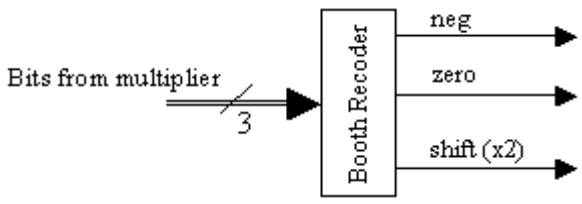


Figure 1.6 : Booth Recoder and its associated inputs and outputs.

In figure

1) The zero signal indicates whether the multiplicand is zeroed before being used as a partial product

2) The shift signal is used as the control to a 2:1 multiplexer, to select whether or not the partial product bits are shifted left one position.

3) Finally, the negative signal indicates whether or not to invert all of the bits to create a negative product (which must be corrected by adding "1" at some later stage)

The described operations for booth recoding and partial product generation can be expressed in terms of logical operations if desired but, for synthesis, it was found to be better to implement the truth tables in terms of VHDL case and if/then/else statements.

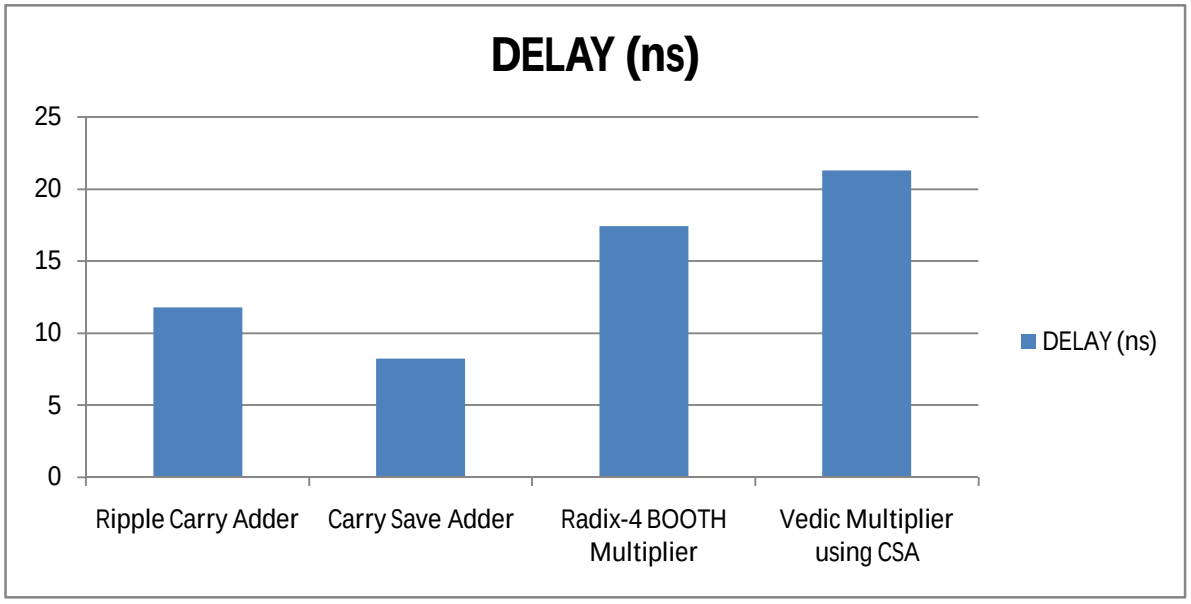
IV. VEDIC MULTIPLIER USING CARRY SAVE ADDER

Since we know that multiplier is nothing, but a process of repeated additions. So, a multiplier can be designed by using a suitable adder. In this kind of Vedic multiplier, fastest adder, that is, Carry Save Adder has been used to maximize its speed as much as possible. This kind of multiplier is very useful when area of the circuit is taken into consideration, since same CSA that has been used as adder is used in the multiplication process also. But in case we need high speed operations, BOOTH multiplier is faster than the Vedic multiplier so formed.

V. SPEED AND PROPAGATION DELAY

Propagation Delay is the time taken by an electronic circuit to propagate the signal from the input to output. In a digital circuit, it may include the delay of various gates, flip-flops, etc. Speed of any circuit is inversely proportional to the propagation delay of the circuit. It implies that more the propagation delay, slower is the speed of the circuit and vice versa.

CIRCUIT	DELAY (ns)
RIPPLE CARRY ADDER	11.807 ns
CARRY SAVE ADDER	8.234 ns
RADIX-4 BOOTH MULTIPLIER	17.412 ns
VEDIC MULTIPLIER USING CSA	21.284 ns



The delay for Ripple Carry Adder is calculated to be 11.807 ns which is more than that of Carry Save Adder which has a delay of 8.234 ns. The delay for Radix-4 BOOTH Multiplier is 17.412 ns which is less than Vedic Multiplier which has a delay of 21.284 ns.

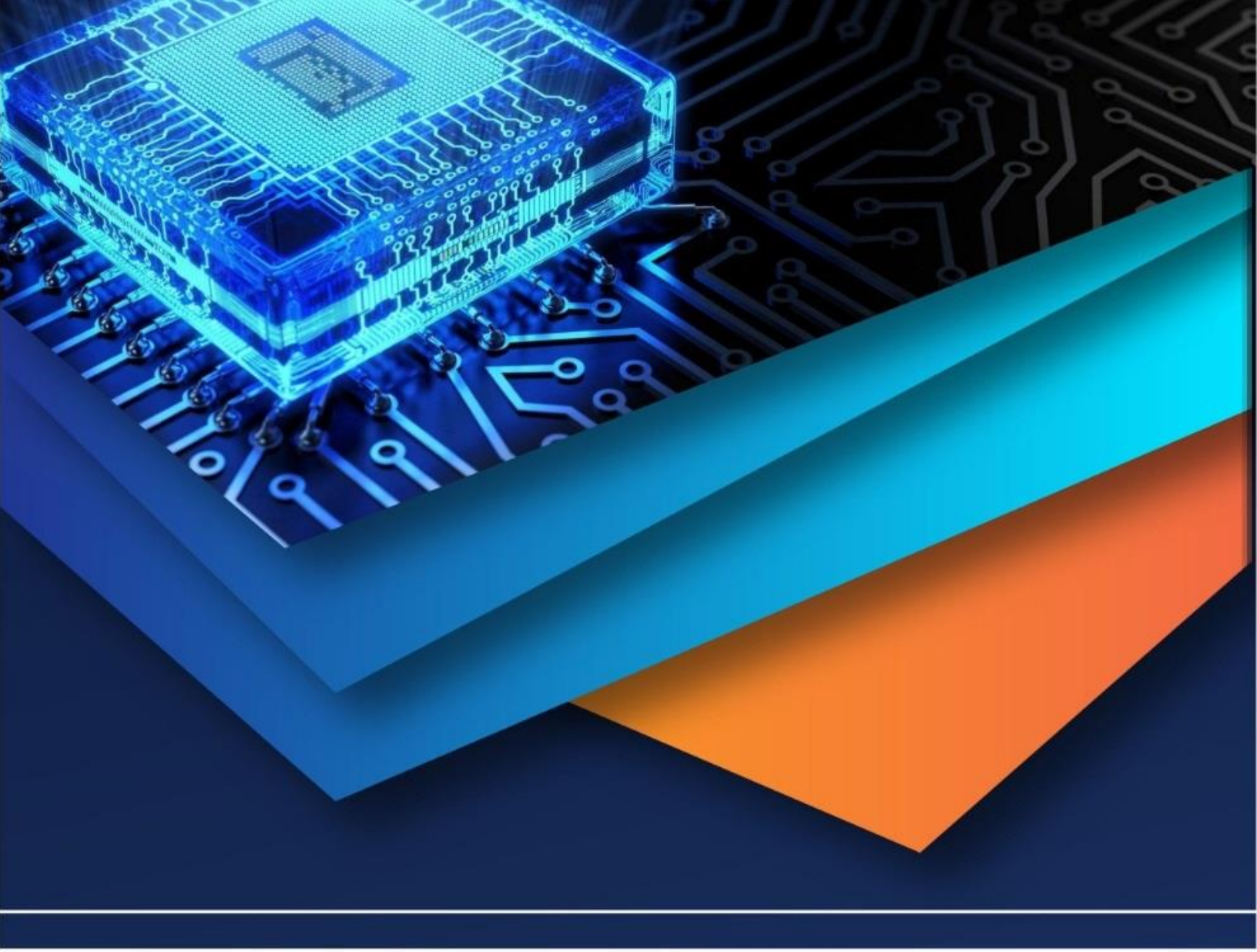
VI. CONCLUSIONS

So it is evident from the results that the delay for Carry Save Adder is 8.234 ns less than that of Ripple Carry Adder which has a delay of 11.807 ns. Also, the delay of Radix-4 BOOTH Multiplier is 17.412 ns which is less than that of Vedic Multiplier which has a delay of 21.284 ns. So the speed of Radix-4 BOOTH Multiplier is more than that of Vedic Multiplier. Similarly CSA is faster than

RCA. So in case, we want to make a high speed ALU, we can use CSA as adder and Radix-4 BOOTH Multiplier as multiplier. It again agrees with fact that CSA is the faster adder[2] and Radix-4 BOOTH Multiplier is the fastest multiplier[8].

REFERENCES

[1] Prakash, P. and Saxena, A.K., 2009, October. Design of Low Power High Speed ALU Using Feedback Switch Logic. In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on (pp. 899-902). IEEE.
[2] Gurjar, P., Solanki, R., Kansliwal, P. and Vucha, M., 2011, December. VLSI implementation of adders for high speed ALU. In India Conference (INDICON), 2011 Annual IEEE (pp. 1-6). IEEE.
[3] Ramalatha, M., Dayalan, K.D., Dharani, P. and Priya, S.D., 2009, July. High speed energy efficient ALU design using Vedic multiplication techniques. In Advances in Computational Tools for Engineering Applications, 2009. ACTEA'09. International Conference on (pp. 600-603). IEEE.
[4] Yeh, W.C. and Jen, C.W., 2000. High-speed Booth encoded parallel multiplier design. IEEE transactions on computers, 49(7), pp.692-701.
[5] Kuang, S.R., Wang, J.P. and Guo, C.Y., 2009. Modified booth multipliers with a regular partial product array. IEEE Transactions on Circuits and Systems II: Express Briefs, 56(5), pp.404-408.
[6] Harata, Y., Nakamura, Y., Nagase, H., Takigawa, M. and Takagi, N., 1987. A high-speed multiplier using a redundant binary adder tree. IEEE Journal of Solid-State Circuits, 22(1), pp.28-34.
[7] Noll, T.G., 1991. Carry-save architectures for high-speed digital signal processing. Journal of VLSI Signal Processing, 3(1-2), pp.121-140.
[8] Broker, H.J., Cook, R.S., O'connor, J. and Xu, N.S., International Business Machines Corporation, 1993. High speed multiplier. U.S. Patent 5,253,195.
[9] Darley, H.M., Niehaus, J.A. and Ovens, K.M., Texas Instruments Incorporated, 1992. High speed multiplier. U.S. Patent 5,115,408.
[10] Zhu, N., Goh, W.L. and Yeo, K.S., 2009, December. An enhanced low-power high-speed adder for error-tolerant application. In Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on (pp. 69-72). IEEE.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY