International Journal of Engineering Research and Advanced Technology (IJERAT)

DOI: http://dx.doi.org/10.31695/IJERAT.2018.3271

E-ISSN : 2454-6135

Volume.4, Issue 6

June -2018

Design and Comparison of Wallace Multiplier Based on Symmetric Stacking and High speed counters

B.L.Chandrika[1] Santhosh N S[2] and Amaresha S K[3]

Post Graduate Scholar[1-2] and Assistant Professor[3]

UTL technologies Ltd,

Bengaluru, India

ABSTRACT

High latency and efficient addition of multiple operands is an essential operation in any computational unit. The latency, power efficiency and area of multiplier circuits is of critical importance in the performance of processors. Multiplier circuits are an essential part of an arithmetic logic unit, or a digital signal processor system for performing filtering and convolution. The binary multiplication of integers or fixed-point numbers results in partial products that must be added to produce the final product. The addition of these partial products dominates the latency and power consumption of the multiplier. In order to combine the partial products efficiently, column compression is commonly used. Many methods have been presented to optimize the performance of the partial product summation previously. To achieve higher efficiency, more number of bits need to be reduced at a time. For this, the column compression techniques can be used. By using this, the critical path delay can be reduced and also the latency of the multiplier. When higher compression unit is used the energy of the multiplier is also reduced. In this paper, the column compression techniques are compared. The stacking circuits presented in, show an improvement over algorithmic Wallace multiplier which uses the generic equations for the column compression units. In stacking units, the 7:3 and 6:3 counters are derived from a basic 3-bit stacking circuit reducing usage of XOR gates. This reduces the usage of XOR gates and thus the critical delay. While the algorithmic units use generic equations using the generation and propagation functions of an adder.

***Key words**: Stacking circuits, Column compression, High speed counters.*

INTRODUCTION

Major applications require an efficient processor to perform the processing on an enormous amount of data. The parameters that determine the performance of a processor are speed, latency and area. These can be improved by enhancing each block of a processor. The major building blocks of a processor include a multiplier, an ALU (arithmetic and logic unit) and accumulate unit. Enhancement in the multiplier block, is one way to enhance the speed and energy of a processor. The multipliers can be majorly divided into two types: Linear Parallel which includes carry save adder, carry look ahead adder and Column compression which includes Wallace and Dadda multipliers. In general, the tree based multiplier block can be divided into three blocks: Computation of Partial Product, Partial product addition, and final addition. Partial production addition and accumulation of partial products adds to the speed, energy and area of multiplier.

Many multipliers have been proposed in the past to reduce the partial product accumulation. One such multiplier is the Wallace multiplier [1]. Wallace multiplier mainly uses the half and full adders. As the N of the multiplier size increases, usage of half and full adders increase which in turn increases latency of the multiplier. With the advances in technology, few methods have been introduced to reduce the area, power and increase the speed of a Wallace multiplier. One of them is the column compression (CC) technique [2]. The parameter for designing a multiplier such as length of multiplier and allocation of adders is decided by analyzing the number of full adder and half adders. A CC multiplier is designed with parameters obtained using above procedure and it is shown that architectures obtained from this new design technique have higher area efficiency, and have shorter interconnections than the classical multiplier. The counter based Wallace multiplier [3] introduces various column compression techniques and compares them based on area, speed and energy, The 2:2, 3:2, 4:3, 5:3, 6:3 and 7:3 are introduced here. Also 7:2 and 6:2 are introduced, using these combinations, many multipliers can be designed. In [5] and [6], the CC was built using multiplexers alone, ex-OR gates only and combination of mux and ex-OR. The CC techniques are chosen as it reduces the usage of half and full adders replacing them with XOR/mux gates. The XOR/mux gates helps to minimize the number of gates to arrive at the accurate solution. Also, the latency of XOR/mux gates is less as compared to the full adders and half adders. Hence usage of

XOR/mux gates gives a better latency over conventional method. Since, the column compression techniques can lead to excess usage of XOR gates, multiplexers, reducing the logic for compression, helps to use minimal gates possible to arrive at the solution. The higher order compressors were improved in [7] and [8].

A new architecture for 7:3 and 6:3 compressors were developed with various combination of universal gates, mux and ex-or gates. One such combination is presented in [7]. The paper also gives an algorithm way to determine the number of reduction stages and maximum rows required in next stage. Based on the algorithm and maximum rows of next stage, the number and type of compressors are computed. The various compressors or counters used use the basic universal gates and ex-or gate which helps improve the latency and area of a multiplier compared to the traditional multiplier. Since the latency is improves, the multipliers using these compressors can be used for high speed applications. In [8], these compressors were further improvised using stack circuits. It uses a basic 3-bit stacking circuit to construct a 7:3 and 6:3 compressor module.

In this paper, the conventional Wallace multiplier, high speed counters in [7] and stack based compression techniques [8] are compared for 8,16,32,64,128,256 and 512 bit. Although the algorithm used to determine the number of stages and reduction techniques is depicted in further section. The compression modules proposed in these papers are picked to compare the number of gates and latency of the multiplier.

The rest of the paper is organized as follows. Section 2 discussed the algorithm used to calculate number of stages, type and number of compressors used for a multiplier. Section 3 discusses the compression modules using high speed counters. Section 4 discusses the stacking circuits based compression module. Section 5 presents the results. The work is concluded in Section 6.

Section 2: Algorithm to calculate type and number of compressors

For designing a multiplier or any combinational unit, it is important to have a physical view of the circuit. Doted representation is used in case of multiplier. Consider 8-bit multiplier, the partial products can be represented in doted form as depicted in Fig 2 (a). These can be further depicted in pyramid shape for easier reduction as shown in Fig 2 (b).

(a) **(b)**

Fig 2 Dotted representation of partial products for 8-bit multiplier

Consider the dotted notation in figure 2 (b) this can be considered as a matrix. The transpose of this matrix (Fig 3) is used as input to compute the number of compressors required.

Fig 3 Transpose of Dotted representation in Fig 2(b)

The algorithm used to calculate number and type of compressors required for algorithmic based modules and stack based CC modules is depicted in Fig 4. This algorithm is run until the number of rows for next stage would be 2. When the number of elements in the next row is 2, the carry is propagated in same stage and the product is obtained. This might reduce the performance, but it is does not have significant effect compared to the previously proposed architectures.

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3271

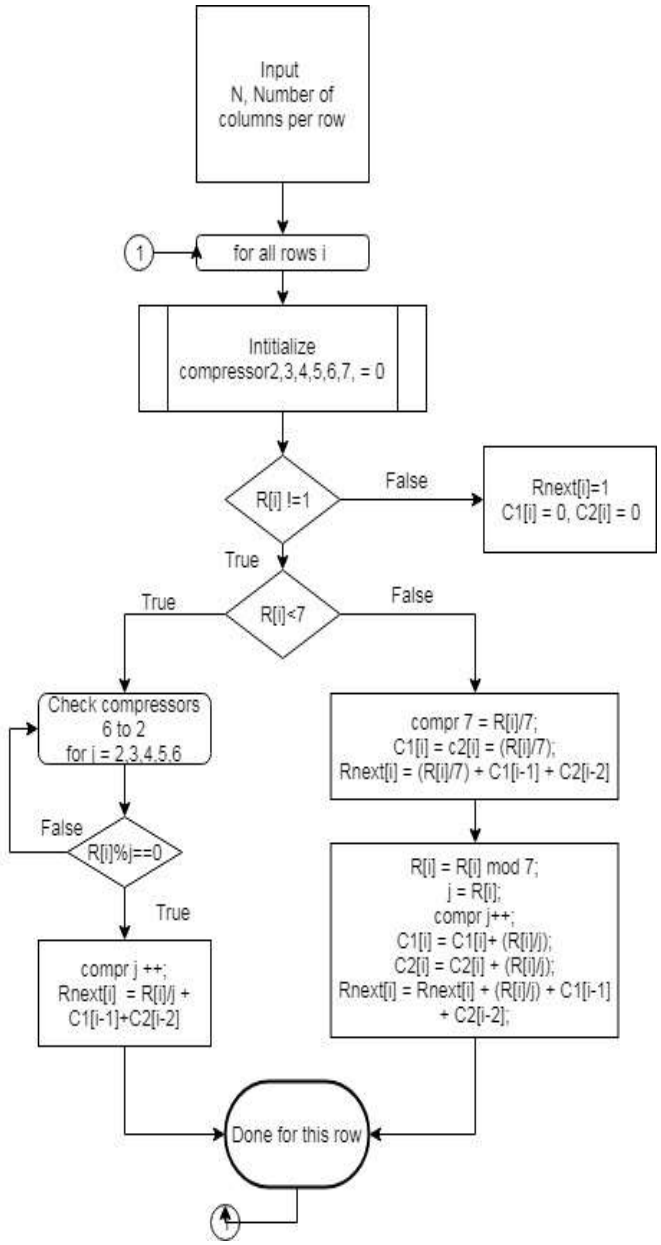


Fig 4 Algorithm to decide the type and Number of compressors required.

The physical view of reduction for 8-bit and 16-bit is depicted in Fig 5 and Fig 6 respectively. The number of stages for each bit is discussed further in further sections.

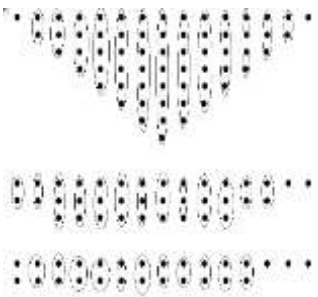
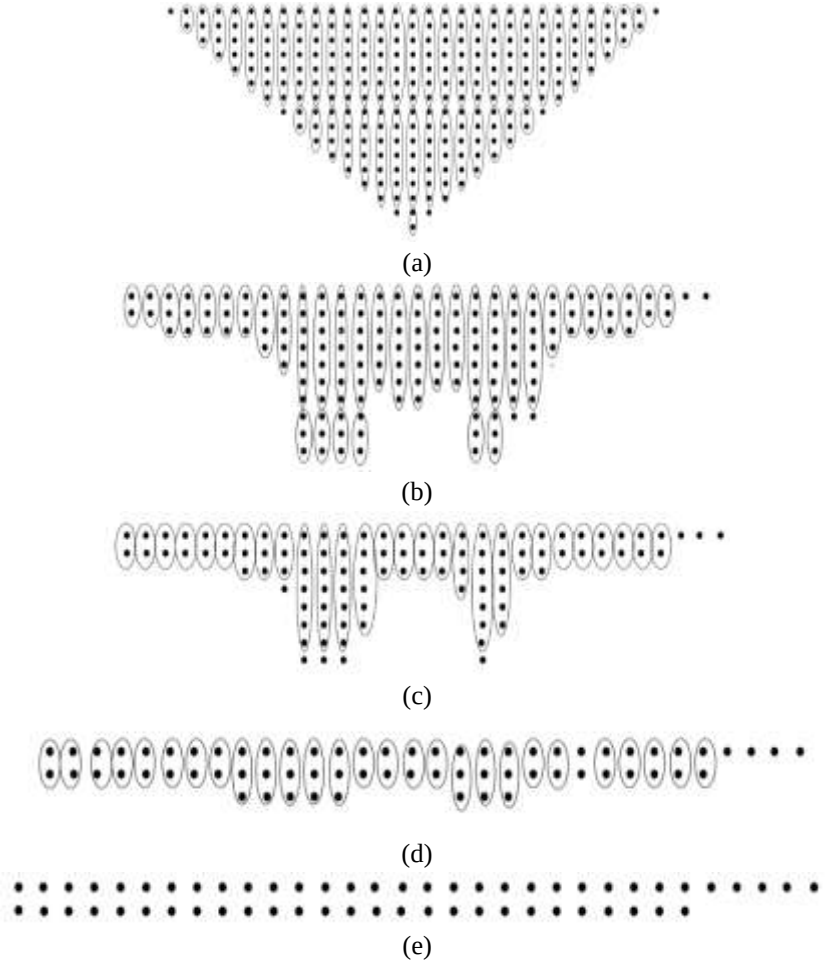


Fig 5 8-bit multiplier reduction stages



(a) (b) (c) (d) (e)

Fig 6. 16 bit multiplier reduction stages

The multiplier architecture for algorithmic and stack based compression technique is the same, but the sub-modules used will be different. The algorithm to decide the number of stages and type and number of compressors used in a multiplier using algorithmic compressors and stack based compressors in depicted in Fig 4.

Section 3: Compression modules using high speed counters

The 4:3, 5:3, 6:3 and 7:3 compressors used for the multiplier architecture using high speed counters [7] are explained below with equations and circuit diagram below.

3.1. 7:3 compressors

The equation (1) gives Boolean functions for 7:3 counter for Sum, C_{out1}, and C_{out2}.

$$Sum = [(A \oplus B) \oplus (C \oplus D)] \oplus [(E \oplus F) \oplus G]$$

$$Cout1 = (w1 \oplus w2) \oplus w3$$

$$Cout2 = (w1.w2) + ((w1 \oplus w2).w3)$$

Where,

$$w1 = A.B + C.D + ((A + B).(C + D))$$

$$w2 = [((E + F).G + E.F)]$$

$$w3 = [A.B.C.D + ((A \oplus B) \oplus (C \oplus D))]. [(E \oplus F) \oplus G]$$

3.2. 6:3 compressors

The logic diagram of the 6:3 compressor is as depicted in Fig 8. The circuit of the compressor is on basis of carry look ahead adder that utilizes the concept of propagate and generate signals to enhance the latency.

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3271

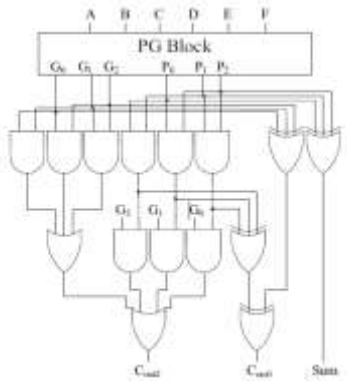


Fig 7. A 6:3 compressor for using generic equations

The propagation and generation functions for the 6:3 compressor are in Boolean equations (2).
P0 = A ⊕B P1 = C ⊕D P2 = E ⊕ F

G0 = A.B G1 = C.D G2 = E.F

The equations for Sum, Cout1, and Cout2 are:
Sum = P0 ⊕ P1 ⊕ P2
Cout1 = (P0.P1 ⊕ P0.P2 ⊕ P1.P2) ⊕ (G0 ⊕ G1 ⊕ G2)
Cout2 = (G0.G1 + G0.G2 + G1.G2) + ((P0.P1).G2) + ((P0.P2).G1) + ((P1.P2).G0)
It can be seen that it utilizes only AND, OR and XOR gates. The G vector and P vector depict the generate and propagate functions respectively.

3.3. 5:3 compressor

The logical representation of the 5:3 compressor is as depicted in Fig. 9. The 5:3 compressor utilizes the propagation and generation functions of 6:3 compressor mentioned above. The 5:3 compressor is implemented by following Boolean equations:
Sum = $P_0 \oplus P_1 \oplus E$
$Cout_1 = (G_0 \oplus G_1 \oplus H_0) \oplus (H_1 \oplus H_2) \oplus ((P_0.P_1) \oplus H_3)$
$Cout_2 = (G_0.G_1 + G_0.H_2) + (G_0.H_3) + (G_1.H_0 + G_1.H_1)$

Where, $H_0 = A.E$, $H_1 = B.E$, $H_2 = C.E$, $H_3 = D.E$

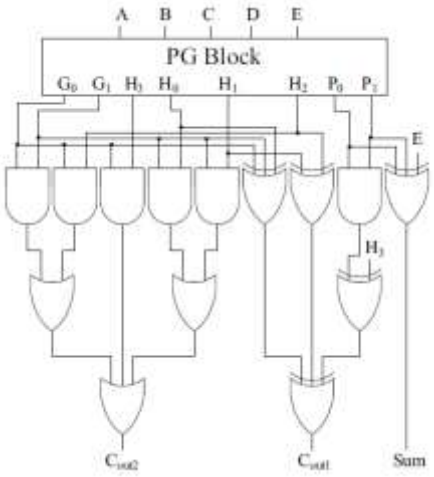


Fig 8. A 5:3 compressor for using generic equations

3.4. 4:3 compressor

The logical representation of the 4:3 compressor is as depicted in Fig. 10. The Boolean equations for the 4:3 compressor is given below:
Sum = P0 ⊕ P1
Cout1 = (P0.P1) + (G0.G1) + (G0.G1)

Cout2 = G0.G1

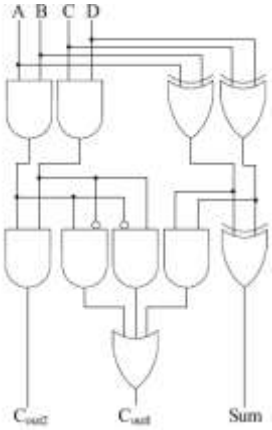


Fig 9. A 4:3 compressor using generic equations

Section 4: Stack based compression techniques

Stacking circuit can be used to reduce the number of XOR gates and lessen the critical path delay of a multiplier circuit. As the name suggests, the number of ones are stacked in the output. The stacking lies on the basic concept that the number of ones in input and output should be equal. Also, the number of ones will be followed by the zeros in the output vector. The 6-bit and 7-bit stacking circuit used here, will be using a 3-bit stacking circuit [8] as a basic building unit. The 3-bit stacking circuit is explained further in this section.

4.1. Three Bit stacking circuit

The circuit for Three-bit stacking is depicted in Fig 11. The 3-bit stacker has three inputs X_0, X_1, and X_2, and three outputs Y_0, Y_1, and Y_2. The output is depends on the number of '1' bits in input. Stacking circuit ensures that the number of '1's in input and output vector are equal. Also, the output vector had '1's grouped.

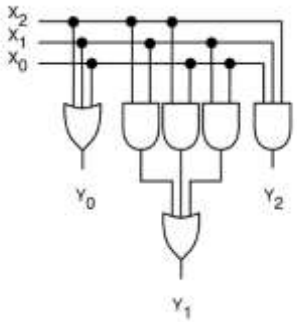


Fig 10. The Three-bit stacking circuit

Equations for the 3-bit stacker is listed below:

$Y_0 = X_0 + X_1 + X_2$

$Y_1 = X_0X_1 + X_0X_2 + X_1X_2$

$Y_2 = X_0X_1X_2$

4.2. 6-bit stacking circuit

The 6-bit stacking circuit is built using the 3-bit stacking module described above. The block diagram is depicted in Fig 12.

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3271

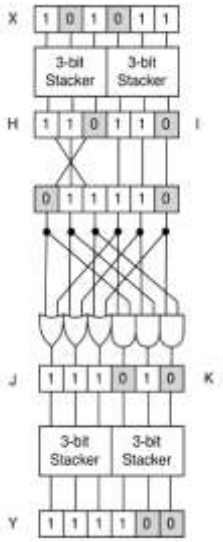


Fig 11. Steps in six-bit stacking

Consider the inputs to be $X_0 \ldots X_5$, grouping them into 3 bits each that are stacked using a three-bit stacking circuits. The X_0, X_1, and X_2 is stacked into signals named H_0, H_1, and H_2 and X_3, X_4, and X_5 be stacked into I_0, I, and I_2. Now by reversing the outputs of X_0, X_1, and X_2 stack and considering these bits H_2, H1, H0, I_0, I_1, and I_2. It is noticed that in these bits, a series of "1" bits exists, with "0" bits around it. In order to construct an appropriate stack, the series of "1" bits should start from left. To ensure it occurs, the J and K vector is formed. J vector should be filled before K vector, so we let:

$J_0 = H_2 + I_0$

$J_1 = H_1 + I_1$

$J_2 = H_0 + I_2$

The K vector is determined using same inputs using AND gates. This makes sure that no bits are counted more than once.

$K0 = H2\ I0$

$K1 = H1\ I1$

$K2 = H0\ I2$.

It is observed that the J vector and K vector contain an equal number of "1" bits as input but the *J* vector is filled with "1"s before the *K* vector. J vector and K vector are stacked using two additional three-bit stacking circuits. The outputs of this is concatenated to get the outputs $Y5 \ldots Y0$.

The Boolean equations to calculate the sum(S) and carry(C_1 and C_2) uses the values of vectors *H*, *I*, and *K* can be used to calculate the output bit instead using the last layer of stack.

He = H0 + H1H2

Ie = I0 + I1 I2.

$S = He \oplus Ie$.

$C_1 = (H1 + I1 + H0\ I0)(K0 + K_1 + K2) + H2\ I2$.

$C_2 = K0 + K1 + K2$.

Using the equations above the circuit for a 6-bit stacking circuit can be drawn as shown in Fig 13 below.

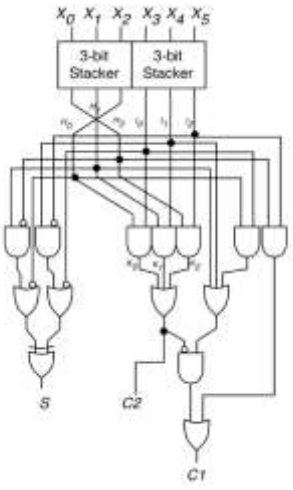


Fig 12. 6:3 compressor module depicted w.r.t gates

4.3. 7-bit stacking module

The symmetric stacking procedure can be utilized to build a 7:3 compressor. These are efficient as they give an elevated compression ratio. Implementation of the 7:3 compressor takes into account the outputs for C_1 and C_2 assuming both $X6 = 0$ (same as the 6:3 compressor) and assuming $X6 = 1$. The equations for sum and carry are mentioned below.

Sum (S): It is done by using an additional XOR gate.

He = H0 + H1H2

Ie = I0 + I1 I2.

$S = He \oplus Ie \oplus X_6$

Carry1 : C_1:

$X6 = 0$, C_1 = (H1 + I1 + H0 I0) (K0 + K_1 + K2) + H2 I2.

$X6 = 1$, $C_1 = (H0 + I0)J0\ \overline{J}\ 1\ J2 + H2\ I1 + H1\ I2$

Carry2 : C_2:

$X6 = 0$, C_2 = K0 + K1 + K2.

$X6 = 1$, $C2 = J0\ J1\ J2$

The C_1 and C_2 are computed in both cases and a mux is used based on $X6$. This design has multiplexers on the critical path. 7:3 compressor design is depicted in Fig. 14.

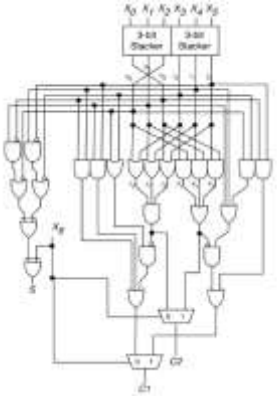


Fig 13. A 7:3 compressor using stacking circuit

Section 5: Results

Number of stages for conventional Wallace multiplier and counter-based, stack-based compression module is depicted in Table 1. It can be observed that the number of stages for a counter based and stack based multiplier is less compared to conventional Wallace multiplier. The latency of a multiplier mainly depends on the number of stages required, the latency is listed in Table 2. A 500 Megahertz clock is used to design these multipliers.

Table 1. Number of stages comparison

N-bit	Stages – Conventional Wallace multiplier	Stages – Counter based and stack based
8	4	2
16	6	4
32	8	6
64	10	8
128	12	10
256	14	11
512	16	13

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3271

Table 2. Latency comparison

N-bit	Conventional multiplier	Stack based modules	High-Speed counter modules
8	5.6	5.7	5.8
16	10.9	11	11.2
32	11.8	11.2	12.5
64	14.7	12.9	14

From the table it is evident that the latency of counter based and stack based is less than the conventional multiplier. Hence it can be concluded that the latency can be improved with this algorithm. The number of half adders and full adders required for a Wallace multiplier is listed in Table 2. Also the corresponding number of gates required are depicted.

Table 3. Number of half adders and full adders for conventional multiplier

N	Number of HA	Number of FA	Total number of gates
8	15	38	144
16	52	200	704
32	156	96	600
64	430	3850	12410

The number and type of compressors used for high speed counters and stack-based compressors are listed in Table 3. The total number of gates required for the conventional and multipliers based on stack-based and high speed counters is listed in Table 4.

Table 4. Number and type of compressors required

N	2:2 compressors	3:2 compressors	4:3 compressors	5:3 compressors	6:3 compressors	7:3 compressors
8	6	11	2	2	2	3
16	40	35	8	5	9	38
32	137	84	27	18	43	226
64	409	187	75	59	109	1125

From Table 2 and Table 3 it is evident that the number of gates will increase for the algorithm in Figure1. But the reduction in latency will improve the performance of the multiplier. To reduce the number of gates, fewer number of high order compressors can be used but there might be slight compromise in latency of the multiplier. The algorithm proposed in [7] ensure fewer number of higher-order compressors are used. From [7] and [8], it is observed that the power is also reduced using these compression modules.

Section 6: Conclusion

The conventional Wallace multiplier performance can be improved by using column compression techniques. The higher order compression modules ensures the number of stages are reduced in turn reducing the latency of the multiplier. In this paper, a generic algorithm is presented to calculate number and type of compressors required. It is observed that the number of stages

using this algorithm is less than those compared to the conventional Wallace multiplier. The column compression techniques can be used to reduce the area further by using more low-order compression modules and less number of higher-order compression modules. It can be seen that the latency has been improved for stack-based and high –speed counters compared to conventional multipliers.

REFERENCES

[1] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[2] Z. Wang, G. A. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," *IEEE Trans. Comput.*, vol. 44, no. 8, pp. 962–970, Aug. 1995.

[3] S. Asif and Y. Kong, "Analysis of different architectures of counter based Wallace multipliers," in Proc. 10th Int. Conf. Comput. Eng.Syst. (ICCES), Dec. 2015, pp. 139–144

[4] Hussain, R.K.Sah, M.Kumar Performance comparison of wallace multiplier architectures, IJIRSET Vol4, Issue 1 , Jan 2015

[5] S. Veeramachaneni, K. M. Krishna, L. Avinash, S. R. Puppala, and M. B. Srinivas, "Novel architectures for high-speed and low-power 3-2, 4-2 and 5-2 compressors," in Proc. 20th Int. Conf. VLSI Design Held Jointly 6th Int. Conf. Embedded Syst. (VLSID), Jan. 2007, pp. 324–329.

[6] K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in Proc. Conf. Rec. 35th Asilomar Conf. Signals, Syst. Comput., vol. 1. Nov. 2001, pp. 129–133.

[7] S. Asif and Y. Kong, "Design of an algorithmic wallace multiplier using high speed counters," in Proc. IEEE Comput. Eng. Syst. (ICCES), Dec. 2015, pp. 133–138

[8] Christopher Fritz and Adly T. Fam, "Fast Binary Counters Based on Symmetric Stacking," IEEE Trans. VLSI, Dec. 2017

DOI : http://dx.doi.org/10.31695/IJERAT.2018.3271