Implementation of Fast Binary Counters Based on Symmetric Stacking

[1]M.Divya, [2]M.Poornima
[1]M.tech student (VLSID), Department of ECE, VEMU Institute of Technology, P.Kothakota
[2] Associate Professor, Department of ECE, VEMU Institute of Technology, P.Kothakota

Abstract-

In this brief, a new binary counter design is proposed. Wallace tree multipliers provide a power-efficient strategy for high speed multiplication. The use of high speed 7:3 counters in the Wallace tree reduction can further improve the multiplier speed. And also we implement 128bit Vedic Wallace multiplier provide high speed and consumes less power and more efficiently. Hence in proposed method we develop 8X8 Wallace tree multiplier stacker and Vedic Wallace 128X128 bit stacker. These proposed methods have better performance improvement than 6 TO 3 Bit stacker and 7 TO 3 bit stacker. In existing method, It uses 3-bit stacking circuits, which group all of the "1" bits together, followed by a novel symmetric method to combine pairs of 3-bit stacks into 6-bit stacks. The bit stacks are then converted to binary counts, producing 6:3 counter circuits with no xor gates on the critical path. This avoidance of xor gates results in faster designs with efficient power and area utilization. In VLSI simulations, the proposed counters are 30% faster than existing parallel counters and also consume less power than other higher order counters. Additionally, using the proposed counter-based Wallace tree multiplier architectures reduces latency and power consumption for 128-bit multipliers.

Index terms – **Counter, high speed, low power, multiplier, VLSI, Wallace tree.**

I. INTRODUCTION

High speed, efficient addition of multiple operands is an essential operation in any computational unit. The speed and power efficiency of multiplier circuits is of critical importance in the overall performance of microprocessors. 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, such as the well-known row compression techniques in the Wallace tree [1] or Dadda tree [2], or the improved architecture in [3]. These methods involve using full adders functioning as counters to reduce groups of 3 bits of the same weight to 2 bits of different weight in parallel using a carry-save adder tree. Through several layers of reduction, the number of summands is reduced to two, which are then added using a conventional adder circuit. To achieve higher efficiency, larger numbers of bits of equal weight can be considered. The basic method when dealing with larger numbers of bits is the same: bits in one column are counted, producing fewer bits of different weights. For example, a 7:3 counter circuit accepts 7 bits of equal weight and counts the number of "1" bits. This count is then output using 3 bits of increasing weight.

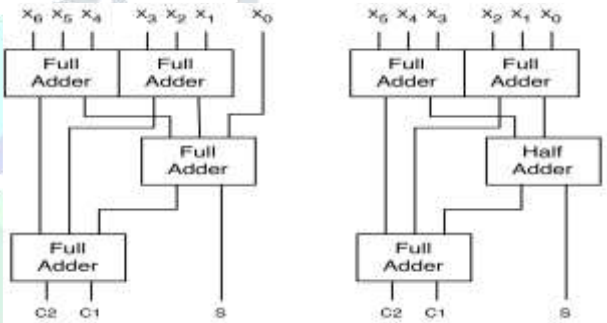


Fig. 1. A 7:3 counter and a 6:3 counter built from full and half adders.

The 7:3 and 6:3 counter circuits can be constructed using full and half adders, as shown in Fig. 1. Much of the delay in these counter circuits is due to the chains of XOR gates on the critical path. Therefore, much faster parallel counter architecture has been presented. A parallel 7:3 counter was presented in [4] and used to design a high speed counter-based Wallace tree multiplier in [5]. Additionally, counter designs as in [6] and [7] use multiplexers to reduce the number of XOR gates on the critical path. Some of these muxes can be implemented with transmission gate logic to produce even faster designs.

In this brief, we present a counting method that uses bit stacking circuits followed by a novel method of combining two small stacks to form larger stacks. A 6:3 counter built using this method uses no XOR gates or multiplexers on its critical path. VLSI simulation results
show that our 6:3 counter is at least 30% faster than existing counter designs while also using less power. Simulations were also run on full multiplier circuits for various sizes.

The same counter-based Wallace multiplier design was used for each simulation, while the internal counter was varied. Use of the proposed counter improves multiplier efficiency for larger circuits, yielding 64- and 128-bit multipliers that are both faster and consume less power than other counter based Wallace (CBW) designs.

This paper is organized in six sections. After this introduction, in Section II, Symmetric bit stacking discussed, Section III existing method discussed of the paper. Finally, Sections IV about the proposed method explained, as well as the novel feature of the proposed method and V provides the simulation results and the conclusions, respectively

II. SYMMETRIC BIT STACKING

The proposed 6:3 counter is realized by first stacking all of the input bits such that all of the "1" bits are grouped together. After stacking the input bits, this stack can be converted into a binary count to output the 6-bit count. Small 3-bit stacking circuits are first used to form 3-bit stacks. These 3-bit stacks are then combined to make a 6-bit stack using a symmetric technique that adds one extra layer of logic.

A. Three-Bit Stacking Circuit

Given inputs $X0$, $X1$, and $X2$, a 3-bit stacker circuit will have three outputs $Y0$, $Y1$, and $Y2$ such that the number of "1" bits in the outputs is the same as the number of "1" bits in the inputs, but the "1" bits are grouped together to the left followed by the "0" bits. It is
Clear that the outputs are then formed by

$$Y0 = X0 + X1 + X2 \ldots\ldots\ldots\ldots (1)$$
$$Y1 = X0X1 + X0X2 + X1X2 \ldots. (2)$$
$$Y2 = X0X1X2 \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Namely, the first output will be "1" if any of the inputs is one, the second output will be "1" if any two of the inputs are one, and the last output will be one if all three of the inputs are "1." The $Y1$ output is a majority function and can be implemented using one complex CMOS gate. The 3-bit stacking circuit is shown in Fig. 2.

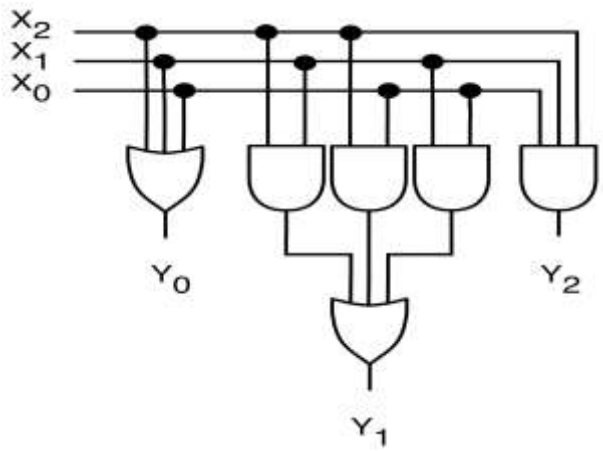


Fig. 2. Three-bit stacker circuit.

B. Merging Stacks

We wish to form a 6-bit stacking circuit using the 3-bit stacking circuits discussed. Given six inputs $X0, \ldots, X5$, we first divide them into two groups of 3 bits which are stacked using 3-bit stacking circuits. Let $X0$, $X1$, and $X2$ be stacked into signals named $H0$, $H1$, and $H2$ and $X3$, $X4$, and $X5$ be stacked into $I0$, I, and $I2$. First, we reverse the outputs of the first stacker and consider the six bits $H2$, $H1$, $H0$, $I0$, $I1$, and I2. See the top of Fig. 3 for an example of this process. We notice that within these six bits, there is a train of "1" bits surrounded by "0" bits. To form a proper stack, this train of "1" bits must start from the leftmost bit.

In order to form the proper 6-bit stack, two more 3-bit vectors of bits are formed called $J0$, $J1$, $J2$ and $K0$, $K1$, $K2$. The idea is to fill the J vector with ones first, before filling the K vector. So we let

$$J0 = H2 + I0 \ldots\ldots\ldots (4)$$
$$J1 = H1 + I1 \ldots\ldots\ldots (5)$$
$$J2 = H0 + I2 \ldots\ldots\ldots (6)$$

In this way, the first three "1" bits of the train are guaranteed to fill into the J bits although they may not be properly stacked. Now to ensure no bits are counted twice, the K bits are formed using the same inputs but with the AND gates instead

$$K0 = H2\ I0 \ldots\ldots\ldots (7)$$
$$K1 = H1\ I1 \ldots\ldots\ldots (8)$$
$$K2 = H0\ I2 \ldots\ldots\ldots (9)$$

If the train of "1"s is no more than three places long, then all of the K bits will be "0" as the AND gate inputs are three positions apart. If the train is longer than three places long, then some of the AND gates will have both inputs as "1"s as the AND gate inputs are three positions apart. The number of AND gates that will have this property will be three less than the length of the train of "1"s. We notice that now $J0$ $J1$ $J2$ and $K0K1K2$ still contain the same number of "1" bits as the input in total but now J bits will be filled with ones before any of the K bits. We must now stack $J0$ $J1$ $J2$ and $K0K1K2$ using two more 3-bit stacking circuits. The outputs of these two circuits can then be concatenated to form the stack outputs $Y5$. $Y0$.

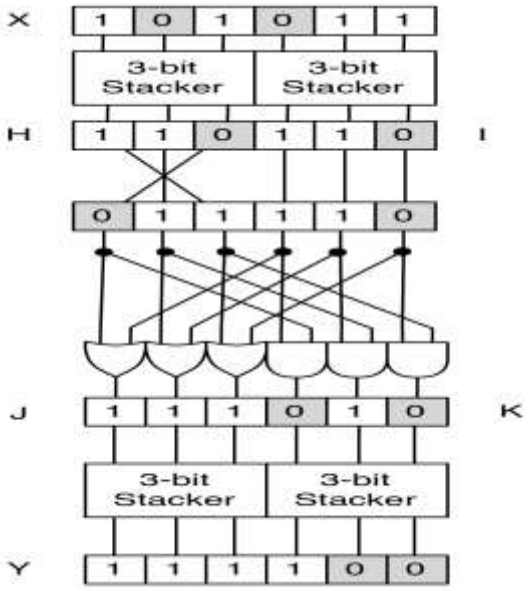


Fig. 3. Six-bit stacking example.

An example of this process is shown for an input vector containing four "1" bits in Fig. 3. In this example, first the *H* and *I* vectors are formed by stacking groups of three input bits. Then, the *H* vector is reversed, forming a continuous train of four "1" bits surrounded by zero bits. Corresponding bits are OR-ed to form the *J* vector which is full of "1" bits. Corresponding bits are AND-ed to form the *K* vector which finds exactly one overlap. Then, the *J* and *K* vectors are restacked to form the final 6-bit stack.

III.EXISTING SYSTEM

A.6:3 Counter based design

For a faster, more efficient count, we can use intermediate values *H*, *I* and *K* to quickly compute each output bit without needing the bottom layer of stackers. Call the output bits *C*2, *C*1, and *S* in which *C*2, *C*1, *S* is the binary representation of the number of "1" input bits.

To compute *S*, we note that we can easily determine the parity of the outputs from the first layer of 3-bit stackers. Even parity occurs in the *H* if zero or two "1" bits appear in *X*0, *X*1, and *X*2.

Thus, *He* and *Ie*, which indicate even parity in the *H* and *I* bits, are given by

$He = H0 + H1H2$ (10)

$Ie = I0 + I1\ I2$. (11)

As *S* indicates odd parity over all of the input bits, and because the sum of two numbers with different parities is odd, we can compute *B*0 as

$S = He \oplus Ie$. (12)

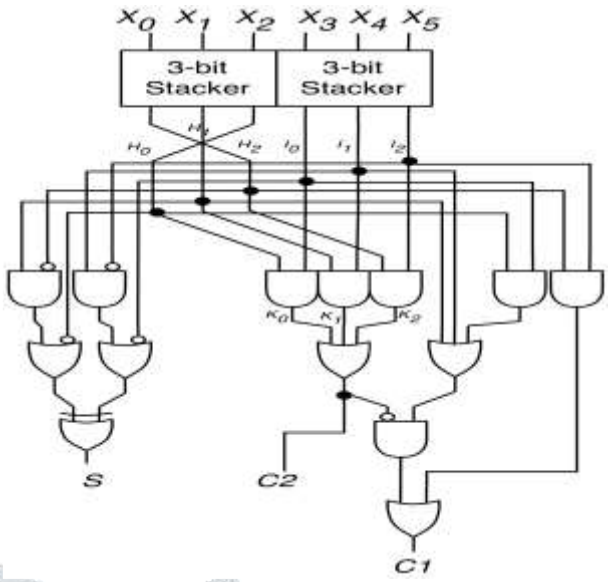


Fig. 4. A 6:3 counter based on symmetric stacking

Although this does incur one XOR gate delay, it is not on the critical path. To compute C1, we note C1 = 1 when the count is 2, 3, or 6. Therefore, there are two cases. First, we need to check if we have at least two but no more than three total inputs. We can use the intermediate H, I, and K vectors for this. To check for at least two inputs we need to see stacks of length two from either top level stacker, or two stacks of length one, which yields H1 + I1 + H0 I0. Second, we need to check if we have all six inputs as "1." We can check this by checking that all three of both the *H* and *I* bits are set. As these are bit stacks, we simply check the rightmost bit in the stack for this case, which yields H2 I2. Altogether, this yield

$$C1 = (H1 + I1 + H0\ I0)(K0 + \overline{K1} + K2) + H2\ I2. \quad (13)$$

We can easily calculate C2 as it should be set whenever we have at least 4-bit set

$$C2 = K0 + K1 + K2.$$

B.7:3 Counter based design

The symmetric stacking method can be used to create a 7:3 counter as well. The 7:3 counters are desirable as they provide a higher compression ratio. The design of the 7:3 counter involves computing outputs for *C*1 and *C*2 assuming both *X*6 = 0 (which matches the 6:3 counter) and assuming *X*6 = 1. We compute the *S* output by adding one additional XOR gate.

If *X*6 = 1, then *C*1 = 1 if the count of *X*0, . . . , *X*5 is at least 1 but less than 3 or 5, which can be computed as $C1 = (H0 + I0)J0\ J1\ J2 + H2\ I1 + H1\ I2$. (15) Also, *C*2 = 1 if the count of *X*0, . . . , *X*5 is at least 3 $C2 = J0\ J1\ J2$. Both versions of *C*1 and *C*2 are computed and a mux is used to select the correct version based on *X*6. has muxes on the critical path. The 7:3 counter designs is shown in Fig. 5.

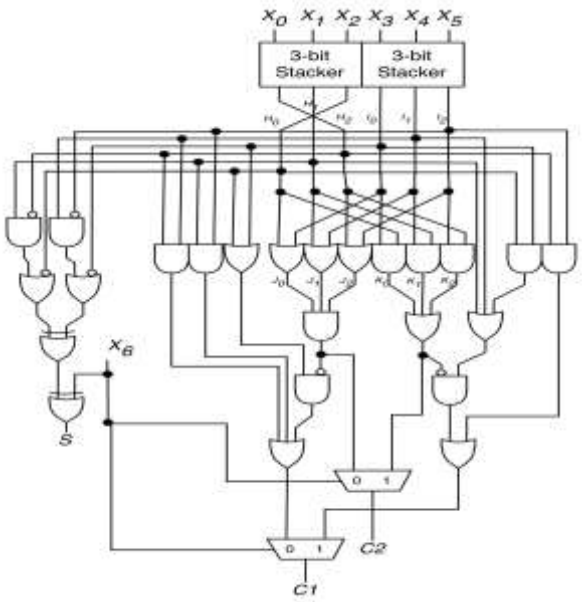


Fig. 5. A 7:3 counter based on symmetric stacking

IV. PROPOSED SYSTEM

A. Wallace tree multiplier

This section discusses the design of counter based Wallace (CBW) multiplier. The proposed algorithm uses a readjusted form of partial product tree which is rearranged as reverse pyramid [4]. Then the reduction is performed using the 7:3, 6:3, 5:3, and 4:3 counters along with the FAs and HAs. The CBW can perform the tree reduction in fewer stages as compared to the traditional Wallace multipliers [1]. Now, we will develop the equations to compute the maximum rows in each stage of CBW multiplier and total stages required for reduction process for N×N multiplier.

The first stage of an $N \times N$ multiplier has N rows. We need to find the maximum rows in subsequent stages until we are left with only two rows. Let us assume that the maximum rows in stage$i-1$ are 16 and the number of rows in each column are same. In order to perform the reduction at column c, two 7:3 compressors are used which will operate on 14 rows. The remaining two rows are reduced by using a 2:2 compressor.

This will reduce the rows in column c from sixteen to three. Similarly, the columns $c-1$ and $c-2$ are reduced by using two 7:3 and one 2:2 compressor. The three compressors used at column $c-1$ will produce three *Cout*1 bits which are added to the column c of the stage$i$. This will increase the rows in column c of stage$i$ from 3 to 6. The two 7:3 compressors at column $c-2$ will produce two *Cout*2 bits which are also added to the column c of stage$i$. Hence, the rows in column c of stage$i$ will increase from 6 to 8.

It can be seen from the above example that the 2:2 compressor at column $c-2$ does not produce a *Cout*2 bit so it has no effect on column c. The compression is performed mainly by using 7:3 compressors, the other compressors are used only if the rows in a column are not exact multiple of seven. There will be one unprocessed row if the rows in column c are equal to $(n \times 7) + 1$, where n is any positive integer.

Based on the observations of above example, we can obtain the rows in stage$i$ by adding

1) The number of traditional and proposed compressors at column c and $c-1$ of stage$i-1$.

2) The number of proposed compressors (7:3, 6:3, 5:3, and 4:3) at column $c-2$ of stage$i-1$.

3) The unprocessed row at column c of stage$i-1$.

The dot notation [11] is used to represent the partial product tree of the CBW multiplier discussed in this section as shown in Fig. 5. The right most column is called column 0. The counters in each column are represented by the boxes around the dot products. The box enclosing seven, six, five, four, three, and two dots represents 7:3, 6:3, 5:3, 4:3, 3:2, and 2:2 counters, respectively. The stages are separated by a thick horizontal line. The architecture of CBW multiplier is based on the intelligent use of high speed counters. The Fig. 6 shows the dot diagram of a 16 ×16 CBW multiplier.

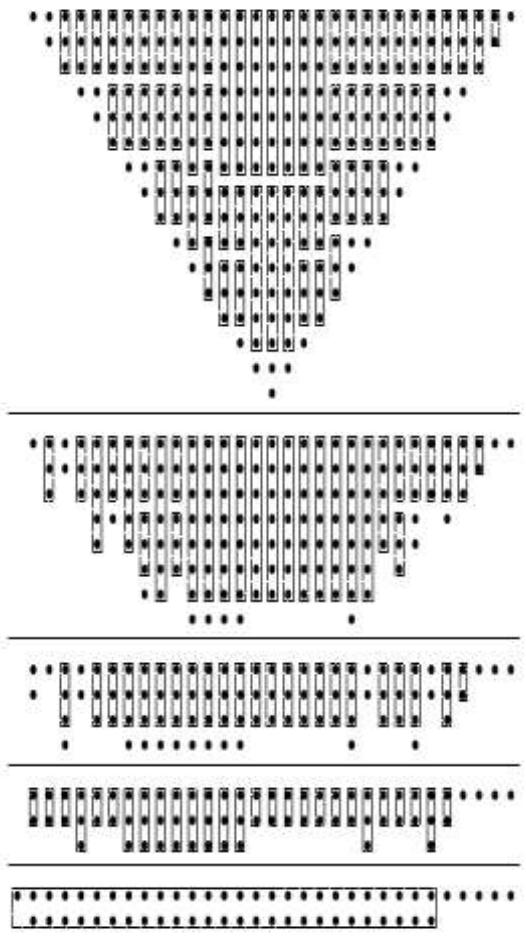


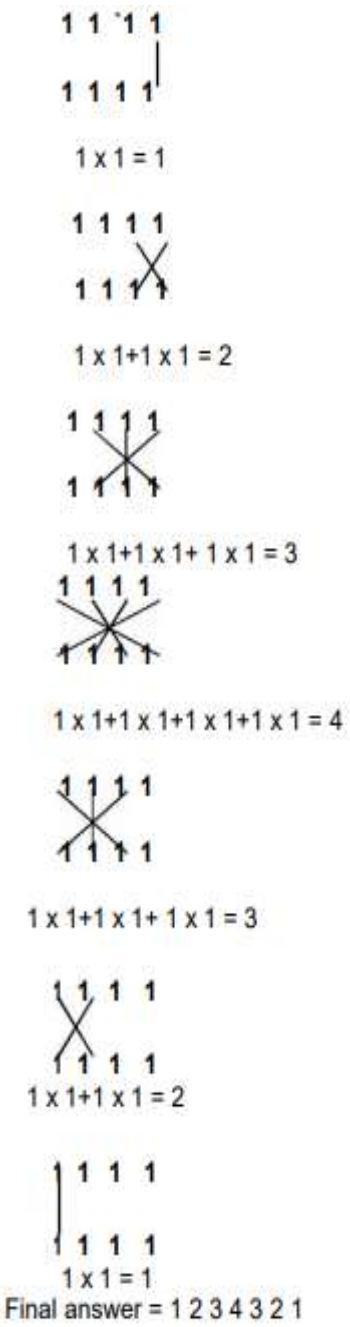
Fig. 6: Dot Diagram of CBW Multiplier

B. Vedic Wallace tree multiplier

Multiplication methods are extensively discussed in Vedic mathematics. Various tricks and short cuts are suggested by VM to optimize the process. These methods are based on concept of 1 Multiplication using deficits and excess 2 Changing the base to simplify the operation. Various methods of multiplication proposed in VM a) UrdhvaTiryagBhyam - vertically and crosswise b) Nikhilam navatashcharamam Dashatah: All from nine and last from ten

Urdhva – Triyakbhyam is the general formula applicable to all cases of multiplication and also in the division of a large number by another large number. It means vertically and crosswise. We discuss multiplication of two, 4 digit numbers with this method [8-9]. Ex.1. the product of 1111 and 1111 using Triyakbhyam (vertically and crosswise) is given below

Methodology of Parallel Calculation

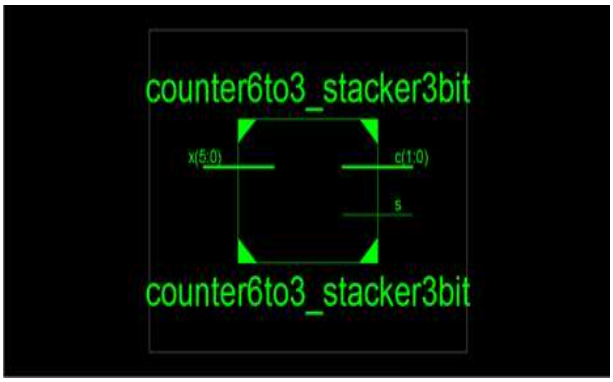


V. SIMULATION RESULTS

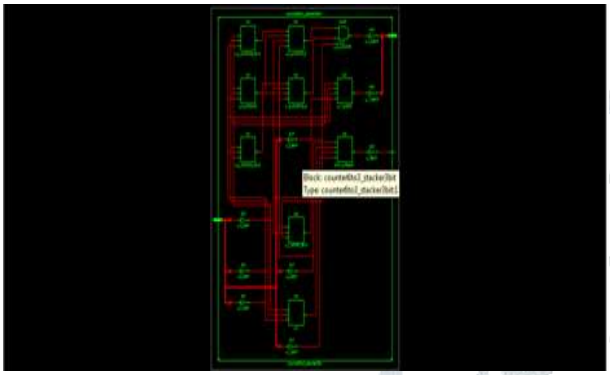
1. EXISTING SYSTEM

A. 6 TO 3 STACKER 3 BIT

- ***RTL For Counter 6 To 3- Stacker 3- Bit***



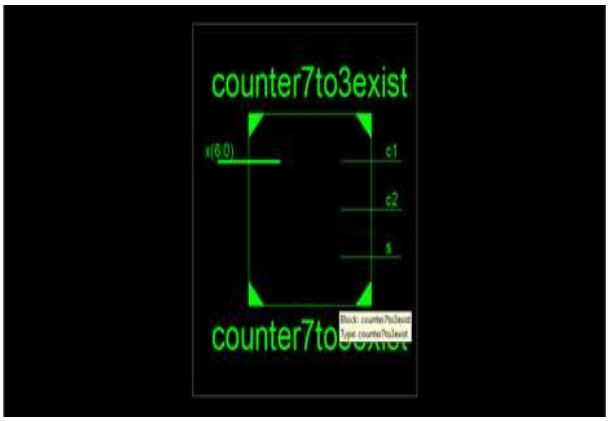
- **Technology Schematic for Counter 6to3-Stacker**



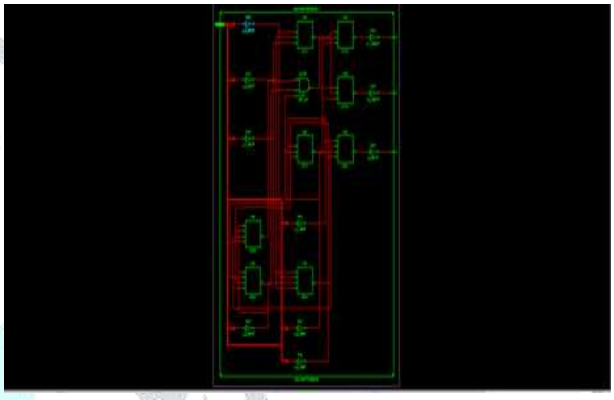
- **Simulation Results of Counter 6 To 3 Stacker-3-Bit**

B. COUNTER 7 T0 3

- **RTL For Counter 7 To 3**



- **Technology Schemetic For Counter 7 To 3**

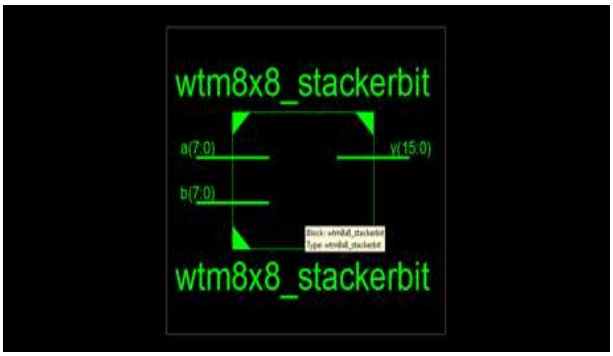


- **Simulation Results of Counter 7 To 3 Stacker-3-Bit**

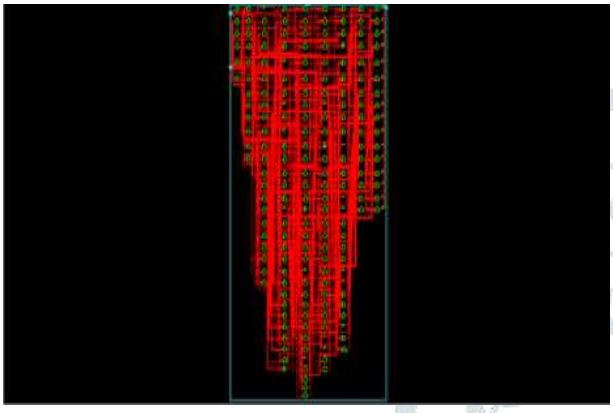
2. PROPOSED METHOD

A. WALLACE MULTIPLIER STACKER

- ***RTL for 8x8 Wallace Tree Multiplier***



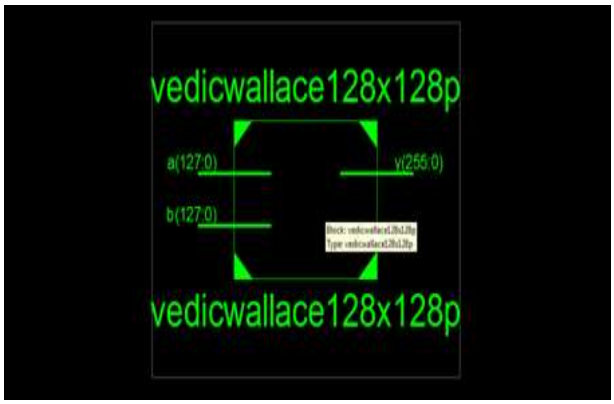
- **Technology Schematic for *8x8 Wallace Tree Multiplier***



- **Simulation Results Schematic for *8x8 Wallace Tree Multiplier***

B. VEDIC WALLACE 128 BIT STACKER

- ***RTL for Vedic Wallace 128X128 bit stacker***



- ***Simulation Results for Vedic Wallace 128X128 bit stacker***

VI. CONCLUSION

In this paper, we proposed a generic algorithm that can be used to construct the Counter Based Wallace (CBW) multiplier of any size. Here we have developed 8X8 Wallace tree multiplier stacker and also a highly efficient method of multiplication – "Urdhva Tiryakbhyam Sutra" based on Vedic mathematics. It is a method for hierarchical multiplier design which clearly indicates the computational advantages offered by Vedic methods.

Wallace tree multipliers provide a power-efficient strategy for high speed multiplication. The use of high speed 7:3 counters in the Wallace tree reduction can further improve the multiplier speed. And also we implement 128bit Vedic Wallace multiplier provide high speed and consumes less power and more efficiently. Hence in proposed method we have developed 8X8 Wallace tree multiplier stacker and Vedic Wallace 128X128 bit stacker.

REFERENCESS

[1] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, February 1964.

[2] 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.

[3] H. Eriksson, P. Larsson-Edefors, M. Sheeran, M. Sjalander, D. Johansson, and M. Scholin, "Multiplier reduction tree with logarithmic logic depth and regular connectivity," in *Proc.IEEE Int. Symp. Circuits and Systems, 2006. ISCAS*, Island of Kos, May 2006, pp. 5–8.

[4] R. S. Waters and E. E. Swartzlander, "A reduced complexity Wallace multiplier reduction," *IEEE Transactions on Computers*, vol. 59, no. 8, pp. 1134–1137, August 2010.

[5] C. C. Foster and F. Stockton, "Counting responders in an associative memory," *Computers, IEEE Transactions on*, vol. C-20, no. 12, pp. 1580–1583, Dec 1971.

[6] E. Swartzlander, "Parallel counters," *Computers, IEEE Transactions on*, vol. C-22, no. 11, pp. 1021–1024, Nov 1973.

[7] C. Vinoth, V. Bhaaskaran, B. Brindha, S. Sakthikumaran, V. Kavinilavu, B. Bhaskar, M. Kanagasabapathy, and B. Sharath, "A novel low power and high speed wallace tree multiplier for risc processor," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 1, April 2011, pp. 330–334.

[8] K. Prasad and K. Parhi, "Low-power 4-2 and 5-2 compressors," in *Signals, Systems and Computers, 2001. Conference Record of the Thirty- Fifth Asilomar Conference on*, vol. 1, Nov 2001, pp. 129–133 vol.1.

[9] C.-H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power cmos 4-2 and 5-2 compressors for fast arithmetic circuits," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 10, pp. 1985–1997, Oct 2004.

[10] M. Mehta, V. Parmar, and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits," in *Computer Arithmetic, 1991. Proceedings., 10th IEEE Symposium on*, Jun 1991, pp. 43–50.

[11] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[12] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, August 1973.

[13] J. Sklansky, "Conditional-sum addition logic," *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, 1960.

[14] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, March 1982.

ABOUT AUTHORS

1. Miss M.Divya received B.Tech degree from VEMU Institute of Technology, P.Kothakota, Chittoor Dist, A.P. Currently She is Pursuing M.Tech degree (VLSID) in Department of ECE in VEMU Institute of Technology P.Kothakota, Chittoor Dist, A.P, and India.

Her interested areas are VLSI, FPGA etc.

2. Mrs M.Poornima received B.Tech degree from SKIT, Srikalhasti, Chittoor Dist, A.P, and India. and M.tech degree received from JNTUA ,Anantapuramu, Anantapur Dist, A.P, and India.

Her interested areas are Digital Image Processing.