Implementation of FIR Filter using Wallace Multiplier based on Symmetric Stacking

P. Pooja
PG Scholar, Department of ECE, MVSR Engineering College, Hyderabad, India

T. Kavitha
Associate Professor, Department of ECE, MVSR Engineering College, Hyderabad, India

Abstract –A new binary counter design is proposed which groups all of the "1" bits together using symmetric bit stacking. The bit stacks are then converted to binary counters, producing 6:3 and 7:3 counter circuits eliminating exor gates on the critical path. This avoidance of exor gates results in reduction of delay and power consumption. By using these symmetric stacking based counters a Wallace multiplier is designed with the help of Xilinx 14.7 version. FIR filter is implemented by using bit stacking based Wallace multiplier which reduces the delay and power consumption of FIR filter up to 61 percent and 4.16 percent respectively when compared to normal FIR.

Index Terms— Counter, power, multiplier, delay, Wallace tree.

1. INTRODUCTION

Efficient addition of multiple operands is an essential operation in any computational unit like ALU's, DSP's etc. The binary multiplication results in partial products that must be added to produce the final product. The addition of these partial products dominates the delay 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 [2] or the improved architecture in [4]. 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.

2. RELATED WORK

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.

2.1 Counters

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. 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 EXOR gates. Therefore, much faster parallel counter architecture has been presented. A parallel 7:3 counter was presented in [5] and used to design a high speed counter-based Wallace tree multiplier in [6]. Additionally, counter designs as in [7] and [8] use multiplexers to reduce the number of XOR gates. Some of these muxes can be implemented with transmission gate logic to produce even faster designs.

In this brief, a counting method is presented that uses bit stacking circuits followed by a novel method of combining two small stacks to form larger stacks.

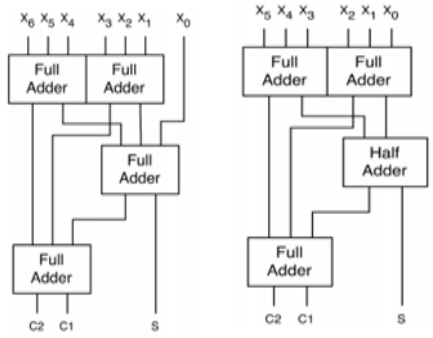

Figure 1 7:3 and 6:3 counters built from FA and HA

A 6:3 counter built using this method uses no XOR gates or multiplexers on its critical path. 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-multiplier that decreases both delay and power consumption than other counter based Wallace (CBW) designs.

2.2 Symmetric bit stacking

Stacking: From the given input bits all the "1" bits are grouped together followed by "0" bits.

The 6:3 and 7:3 counters are 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.

2.2.1 Three-Bit Stacking

Given inputs X_0, X_1, and X_2, a 3-bit stacker circuit will have three outputs Y_0, Y_1, and Y_2 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

$$\mathbf{Y_0 = X0 + X_1 + X_2}$$

$$\mathbf{Y_1 = X_0 X_1 + X_0 X_2 + X_1 X_2}$$

$$\mathbf{Y_2 = X_0 X_1 X_2}$$

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 3-bit stacking circuit is shown in Fig. 2.

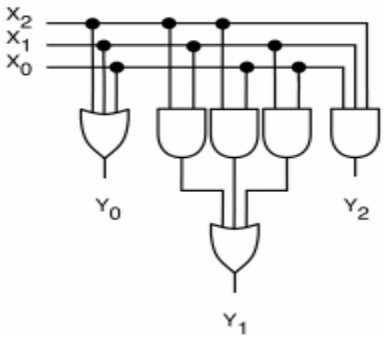

Figure 2 Three-bit stacker circuits.

2.2.2 Merging Stacks

To form a 6-bit stacking circuit 3-bit stacking circuits are used. Given six inputs X_0, X_1, X_2, X_3, X_4, X_5. These six inputs are first divided into two groups of 3 bits which are stacked using 3-bit stacking circuits. Let X_0, X_1, and X_2 be 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_1, and I_2. First, the outputs of the first stacker are reversed and the six bits are considered as H_2, H_1, H_0, I_0, I_1, and I_2.

For an example of this process top module of six bit stacking circuit can be observed. Within these six bits, it is noticed that, 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 J_0, J_1, J_2 and K_0, K_1, K_2. The idea is to fill the J vector with ones first, before filling the K vector. Therefore these can be given as,

$$\mathbf{J_0 = H_2 + I_0}$$

$$\mathbf{J_1 = H_1 + I_1}$$

$$\mathbf{J_2 = H_0 + I_2}$$

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

$$\mathbf{K_0 = H_2 I_0}$$

$$\mathbf{K_1 = H_1 I_1}$$

$$\mathbf{K_2 = H_0 I_2}$$

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. It is noticed that now J_0 J_1 J_2 and K_0 K_1 K_2 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. Now J_0 J_1 J_2 and K_0 K_1 K_2 are again stacked using two more 3-bit stacking circuits. The outputs of these two circuits can then be concatenated to form the stack outputs Y_5, Y_4, Y_3, Y_2, Y_1, and Y_0.

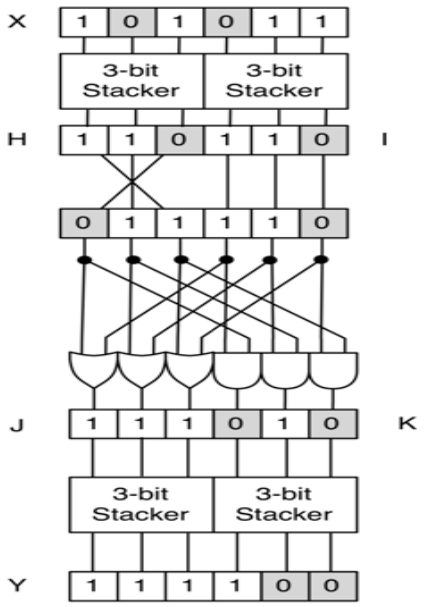

Figure 3 Six-bit stacking example.

An example of this process is shown in the Fig. 3 for an input vector containing four "1" bits in 6-bit stacking circuit. 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.

2.3 Converting bit stack to binary number

In order to implement the counter circuits stack should be converted into binary number. For fast and efficient counting intermediate values of stacking circuit are used without using bottom layers. Stack can be converted to binary number using even parity or odd parity based on given input.

2.3.1 Converting bit stack to binary number in 6:3 counter design:

In order to implement a 6:3 counter circuit, the 6-bit stack must be converted to a binary number. For a faster, more efficient count, intermediate values H, I, and K are used to quickly compute each output bit without need of 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, the parity of the outputs from the first layer of 3-bit stackers can be easily determined. Even parity occurs in the H if zero or two "1" bits appear in X_0, X_1, and X_2. Thus, H_e and I_e, which indicate even parity in the H and I bits, are given by

$$\mathbf{H_e = \overline{H_0} + H_1 \overline{H_2}}$$
$$\mathbf{I_e = \overline{I_0} + I_1 \overline{I_2}}$$

As S indicates odd parity over all of the input bits, and because the sum of two numbers with different parities is odd, S can be computed as

$$\mathbf{S = H_e \oplus I_e}$$

Although this does incur one XOR gate delay, it is not on the critical path. To compute C1, C1 equals to one when the count is 2, 3, or 6. Therefore, there are two cases. First, if there are at least two but no more than three total inputs, the intermediate H, I, and K vectors are used for this. Stacks of length two from either top level stacker, or two stacks of length one, must be checked for at least two inputs which yields $\mathbf{H_1 + I_1 + H_0 I_0}$. If inputs are not more than three then none of the K bits are set as K vector that is K vector is set if there are more than three inputs as "1", which gives $\overline{K_0 + K_1 + K_2}$.

Second, if all the six inputs are "1" it is clear that all three of both the H and I bits are set. As these are bit stacks, simply the rightmost bit in the stack is checked for this case, which yields $\mathbf{H_2 I_2}$. C_2 is to be set whenever it has at least 4-bit set. Altogether, these yields,

$$\mathbf{C_1 = (H_1 + I_1 + H_0 I_0)\ \overline{K_0 + K_1 + K_2} + H_2 I_2}$$

$$\mathbf{C_2 = K_0 + K_1 + K_2}.$$

Using these S, C_1, C_2 equations, the final 6:3 counter circuit can be constructed, as shown in Fig.4. 6:3 counters based on symmetric stacking.

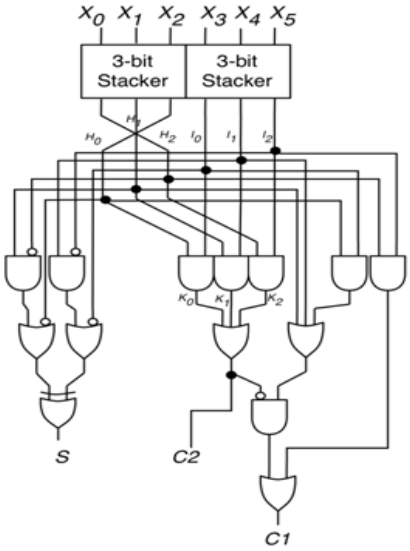

Figure 4 6:3 Counter based on symmetric stacking.

Using larger CMOS gates, the critical path delay is reduced to seven basic gates. As there are no XOR gates on the critical path, this 6:3 counter outperforms existing designs. Because the proposed 6:3 counter based on bit stacking has no XOR gates on its critical path, it operates faster than all other counter designs. Thus, this method of counting via bit stacking allows construction of a counter for a substantial performance increase without increasing power consumption.

2.3.2 Converting bit stack to binary number in 7:3 counter 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 $X_6 = 1$. *S* output is computed by adding one additional XOR gate. If $X_6 = 1$, then $C_1 = 1$ if the count of $X_0 \ldots X_5$ is at least 1 but less than 3 or 5, then C1 is computed as

$$\mathbf{C_1 = (H_0 + I_0)\ J_0 \overline{J_1}\ J_2 + H_2 I_1 + H_1 I_2}$$

Also, C2 = 1 if the count of $X_0 \ldots X5$ has at least 3 number of ones. And given as $\mathbf{C_2 = J_0\ J_1\ J_2}$. Both versions of C_1 and C_2 are computed and a mux is used to select the correct version based on X_6. Note that this design therefore has muxes on the critical path. The 7:3 counter based on Symmetric stacking is shown in Fig.5. Simulations were run on the proposed 7:3 counter against the original 7:3 counters. While the proposed counter is still slightly faster than the existing counters, the improvement is not as significant and the power consumption is increased. For this reason, the proposed 6:3 counters are used to build 64-bit multiplier.

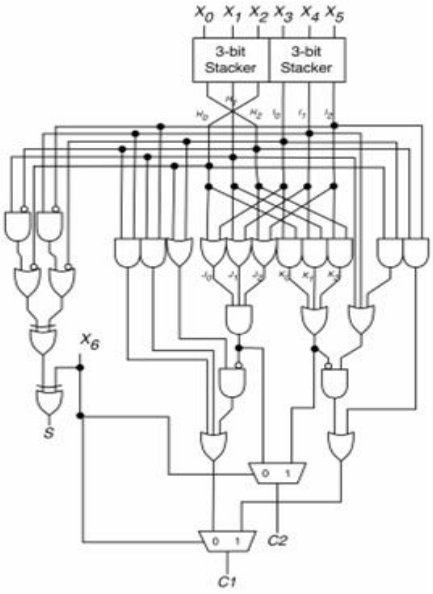

Figure 5 7:3 Counter based on symmetric stacking.

2.4 Counter based Wallace (CBW) multiplier

Partial products of left half of the array of Counter based Wallace multiplier are shifted upwards to form an inverted pyramid array and the partial product generation method is the same, no data are changed; only their vertical position is shifted. The partial products in CBW multiplier are shown in dot notation. 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. To demonstrate a use case of the proposed 6:3 counter, multiplier circuits of different sizes were constructed using different internal counters. No new multiplier design is proposed; rather, existing architectures are simulated with different internal counters. For reference, a standard Wallace multiplier was implemented for each size. Then, the counter-based Wallace multiplier was used which achieves the fewest reduction phases.

The internal 7:3 and 6:3 counters used for this CBW multiplier were varied. The 5:3 and 4:3 counters were kept the same for each multiplier, using the counter designs. Because of the efficiency of the 6-bit version of the proposed counter, for simulations using the stacker-based counter, the 6-bit version with no 7:3 counters, even though this results in one additional reduction phase for each size is used.

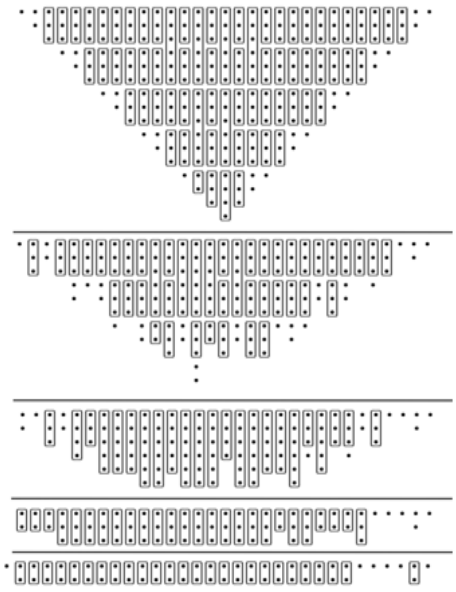

Figure 6 CBW multiplier reduction tree using 6:3 counters.

3. PORPOSED MODELLING

3.1 Finite Impulse Response

In signal processing, a Finite Impulse Response filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).

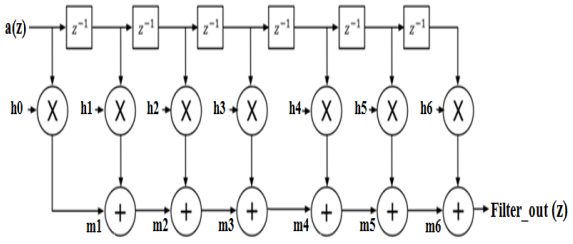

Figure 7 The logical structure of an FIR filter

A low pass FIR filter is shown in Fig.7. Here a(z) is input and filter_out (z) is output. b0, b1, b2, b3 are coefficients. Input values a(z) are multiplied with coefficients h0, h1, h2, h3, h4, h5, h6. Similar to MAC these product values are added to the feedbacks which are represented as m1, m2, m3, m4, m5, m6. And finally output is given as filter_out (z).

4. RESULTS AND DISCUSSIONS

Wallace Multiplier	% of Reduced DELAY	% of Reduced POWER
8 bit	19.64%	1.23%
16 bit	11.18%	1.82%
32 bit	15.43%	2.46%
64 bit	15.97%	3.57%

Table 1 Delay and Power comparisons of Multipliers

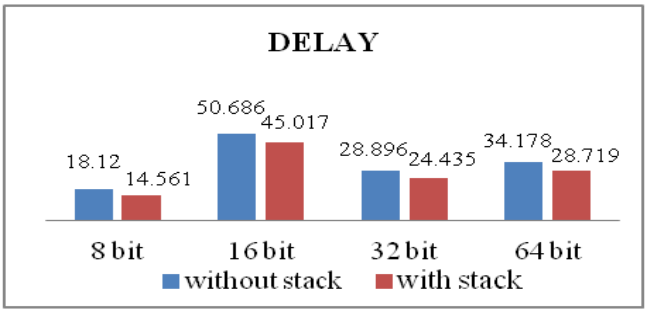

Figure 8 Graphical Representation of Delay comparisons of Multipliers

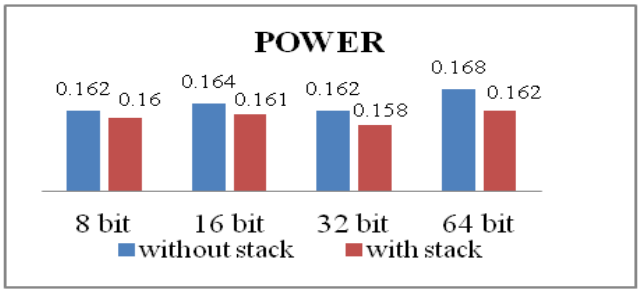

Figure 9 Graphical Representation of Power comparisons of Multipliers

Method	FIR without stacking	FIR with stacking	% Reduction
Delay(ns)	34.827ns	13.526ns	61%
Power(W)	0.168W	0.161W	4.19%

Table 2 Delay and Power comparisons of FIR Filter

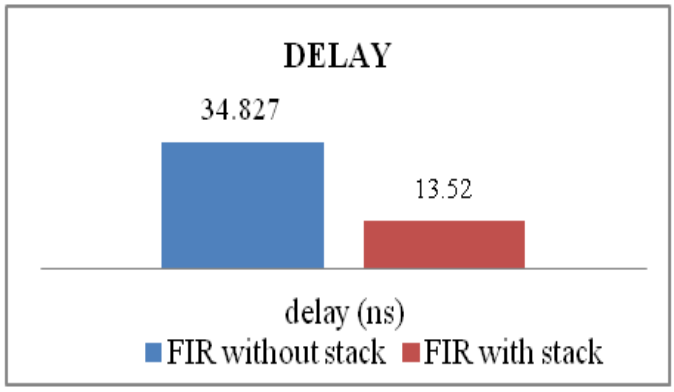

Figure 10 Graphical Representation of Delay comparisons of FIR Filter

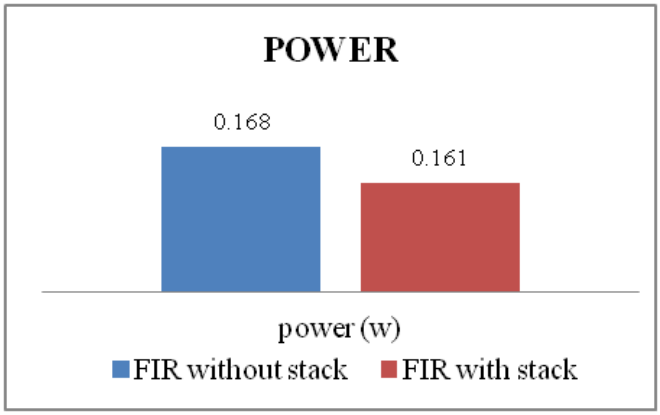

Figure 11 Graphical Representation of Power comparisons of FIR Filter

5. CONCLUSION

A counter based symmetric bit stacking Wallace Multiplier is implemented in FIR Filter. Whereas the existing Wallace multiplier uses full adders and half adders functioning as counters; these counter circuits use chains of XOR gates on critical path which result in more delay. To reduce the delay and increase speed of the multiplier, counter circuits are replaced with symmetric stacking. Then 8- bit, 16-bit, 32-bit, 64-bit Wallace multiplier are implemented using counters based on symmetric stacking which results in less delay and power consumption.

The delay comparisons for 8-bit, 16-bit, 32-bit, 64-bit Wallace Multiplier with symmetric stacking are reduced to 19.64%, 11.18%, 15.43%, and 15.97% respectively when compared with Wallace multiplier without symmetric stacking. The power comparisons for 8-bit, 16-bit, 32-bit, 64-bit Wallace Multiplier with symmetric stacking are reduced to 1.23%, 1.82% , 2.46%, 3.57% respectively when compared with Wallace multiplier without symmetric stacking.

A Low pass FIR filter is designed using Wallace multiplier based on symmetric stacking. This FIR filter uses symmetric stacking based Wallace multiplication in place of normal multiplication. The delay comparison of FIR filter based on Wallace Multiplier with symmetric stacking is reduced to 61% when compared to normal FIR filter. The power comparison of FIR filter based on Wallace Multiplier with symmetric stacking is reduced to 4.16% when compared to normal FIR filter.

REFERENCES

[1] Christopher Fritz and Adly T. Fam, "Fast Binary Counters Based on Symmetric Stacking," IEEE Transactions On Very Large Scale Integration (VLSI) Systems, 1063-8210 © 2017 IEEE
[2] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[3] L. Dadda, "Some schemes for parallel multipliers," Alta Freq., vol. 34, pp. 349–356, May 1965.
[4] 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.
[5] M. Mehta, V. Parmar, and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits," in Proc. 10th IEEE Symp. Comput. Arithmetic, Jun. 1991, pp. 43–50.
[6] 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.
[7] S. Veeramachaneni, L. Avinash, M. Krishna, and M. B. Srinivas, "Novel architectures for efficient (m, n) parallel counters," in Proc. 17th ACM Great Lakes Symp. VLSI, 2007, pp. 188–191.
[8] 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.
[9] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Trans. Comput., vol. 45, no. 3, pp. 294–306, Mar. 1996.
[10] 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.
[11] J. Gu and C.-H. Chang, "Low voltage, low power (5:2) compressor cell for fast arithmetic circuits," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), vol. 2. Apr. 2003, pp. 661–664.
[12] 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.
[13] I. Koren, Computer Arithmetic Algorithms, 2nd ed. Natick, MA, USA: A. K. Peters, 2002.
[14] M. Rouholamini, O. Kavehie, A.-P. Mirbaha, S. J. Jasbi, and K. Navi, "A new design for 7:2 compressors," in Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl., May 2007, pp. 474–478.
[15] A. Dandapat, S. Ghosal, P. Sarkar, and D. Mukhopadhyay, "A 1.2-ns 16 × 16-bit binary multiplier using high speed compressors," Int. J. Elect. Electron. Eng., vol. 4, no. 3, pp. 234–239, 2010
[16] D. Radhakrishnan, "Low-voltage low-power CMOS full adder," IEE Proc.-Circuits, Devices Syst., vol. 148, no. 1, pp. 19–24, Feb. 2001.
[17] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, "Design of high-speed lowpower 3-2 counter and 4-2 compressor for fast multipliers," Electron. Lett., vol. 34, no. 4, pp. 341–343, Feb. 1998.

Authors

P.Pooja received her B.tech. Degree in Electronics and Communication Engineering from Sri Indu College of Engg. & Tech., Affiliated to JNTUH and pursuing M.E in Embedded Systems and VLSI Design from MVSR Engineering College, Affiliated to Osmania University, Hyderabad, India. Her area of interests includes VLSI.

T.Kavitha is an Associate Professor at M.V.S.R Engineering College, Hyderabad in ECE Department. She received her B.E degree in Electronics and Communication Engineering from M.V.S.R Engineering College, Hyderabad and M.Tech degree in Digital Systems and Computer Electronics from J.N.T.U.H, Hyderabad. Her research interest is Signal Processing and Communications.