WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Implementation and Analysis of Wallace Tree Multiplier Using Fast Adders With Xilinx

M.Ashok,C.Gopal,M.Kartheek, Mr.P.Murali

Department of Electronic and communication Engineering, Madanapalle Institute Of Technology And Science,Madanapalle,Andhra Pradesh-517325

ABSTRACT

The primary function of an electronic device is to offer a compact area with high-speed performance. The multiplier is the most sophisticated module in the digital building blocks system and the primary source of power dissipation. Approximate Computing to Multiplier Design is important in electronic applications like multimedia because it provides the quickest solutions despite its low reliability. In this project, a parallel adder, carry skip adder, carry by pass adder, and kogge stone adder strategy of 16-bit Wallace Tree approximation multiplier with 15-4 compressor, To improve reliability, the 5-3 compressor, full adder, and half adder methods are investigated. The 16-bit Wallace tree multiplier is synthesized and simulated using Xilinx ISE 14.7 software

ABSTRACT—

The primary function of an electronic device is to offer a compact area with high-speed performance. The multiplier is the most sophisticated module in the digital building blocks system and the primary source of power dissipation. Approximate Computing to Multiplier Design is important in electronic applications like multimedia because it provides the quickest solutions despite its low reliability. In this project, a parallel adder, carry skip adder, carry by pass adder, and kogge stone adder strategy of 16-bit Wallace Tree approximation multiplier with 15-4 compressor, To improve reliability, the 5-3 compressor, full adder, and half adder methods are investigated. The 16-bit Wallace tree multiplier is synthesized and simulated using Xilinx ISE 14.7 software.

1.Introduction

Microprocessors and Digital Signal Processors (DSP) play an important role in dealing with the complexity of digital signals. Approximately 95% of processors on the market are based on digital signals. Convolution, correlation, and filtering of digital signals are handled by digital signal processors. These jobs are mostly carried out via multipliers, shifters, and adders. The multiplier module is the most complicated of the three. Multipliers take longer and use more power than the other two modules. Multipliers have three phases

- Generation of partial products
- Reduction of partial products
- Final stage addition.

Reducing partial products takes a long time and a lot of power in the multiplier. Many approaches for reducing the critical route in the multiplier have been presented. The employment of compressors in the partial product reduction step is the most common. Compressors are simple circuits that use full or half adders to count the number of "ones" in the input. The partial product reduction stage necessitates the use of many compressors. In the previous 20 years, researchers introduced several compressors such as 3-2, 4-2, 5-2, and 5-3. These are only effective when the multiplier is tiny. Compressors of significant size are required for 16, 32bit multipliers. In terms of power and speed, high order compressors outperform lower order compressors. However, it High order compressors take up more space than low order compressors. All of these approaches and modules execute the precise computation and deliver the proper result. In precise computing, the module or device's accuracy is always 100 percent. But exact computing has one major drawback. Exact computation does not allow for the optimization of all circuit parameters. Exact computation, on the other hand, is not required for every application. There are some Image processing and multimedia programmes, for example, can tolerate faults and provide meaningful results. Because of their low complexity and low power consumption, inexact (approximate) computing approaches have grown in popularity. Even though it has a poor precision, inexact computing gives decent results. The values of error rate (ER), error distance (ED), and normalized error distance (NED) are significant in calculating the final output in approximation computing. The error rate is calculated by dividing the number of erroneous outputs by the total number of outputs. The arithmetic distance between an incorrect output and the correct one defined as Error Distance. The normalized error distance is the ratio of the mean error distance over all inputs to the circuit's maximum input. Several approximation strategies for adders and multipliers have been developed. Starting from the center the accuracy component of adders refers to the most significant bit (MSB) while the accuracy part of adders refers to the least significant bit (LSB). Large errors are caused by inaccurate calculation on the MSB side. In the accurate section, the standard addition rule is used, however in the incorrect part, a particular way of addition is used. When either of the adder's operand values is "0," the output "sum" value is computed normally. If both operands are "1," the "sum" value can be set to "1" from that bit position to the least significant bit. This approach is used to reduce the adder's error distance.

We present a unique approximation adder design that may be used to greatly reduce energy consumption with a very moderate error rate. The utilized carry prediction approach, which uses the information from less significant input bits in parallel, results in a dramatically improved error rate and critical path time. An error magnitude reduction scheme is proposed to further reduce amount of error once detected with low cost. High-performance computers and data processing systems rely heavily on fast arithmetic circuits. Multipliers have been an essential and mandatory component in defining overall circuit performance when confined by power consumption and computation speed in the bulk of these applications. Compressors are an important part of the multiplier circuit since they have a big impact on the total multiplier speed.

System model

2.**FAST ADDERS**

Any digital system's most fundamental operation is addition. Various designs of different adders are covered in the following sections, with a preference for area, number of components, and speed of each device. There are several varieties of adders

- Half Adder
- Full Adder
- Parallel Adder
- Carry Skip Adder
- Carry By-Pass Adder
- Kogge Stone Adder

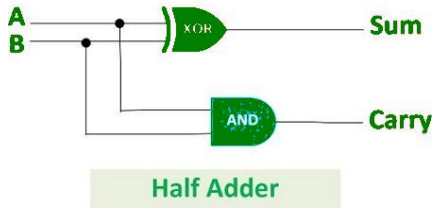
2.1 HALF ADDER

A half adder is a logic circuit that consists of one EX-OR gate and one AND gate connected together. A and B are the two inputs of the half adder circuit, which add two input digits and output a carry and a sum.

The total of the two integers is acquired by the EX-OR gate, whereas the carry is achieved by the AND gate. Carry addition will not be sent since there is no logic gate to handle it. As a result, this circuit is known as a Half Adder.

Sum = A XOR B

Carry = A AND B



2.2 FULL ADDER

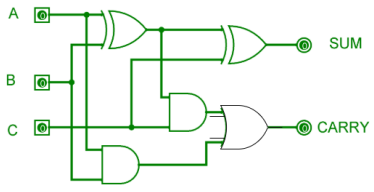
The full adder is an adder that takes three inputs and outputs two results. The first two inputs are A and B, with the third being a C-IN input. The carry output is labelled C-OUT, whereas the usual output is labelled S, which stands for SUM.

A complete adder logic is designed to accept eight inputs and combine them into a byte-wide adder while cascading the carry bit from one adder to the next.

$$SUM = (A \text{ XOR } B) \text{ XOR } Cin = (A \oplus B) \oplus Cin$$

$$CARRY\text{-}OUT = A \text{ AND } B \text{ OR } Cin(A \text{ XOR } B) = A.B + Cin(A \oplus B)$$

Fig.2.2 Full Adder logic circuit



2.3 PARALLEL ADDER

Digital circuits that add binary integers in parallel are known as parallel adders. Parallel strings of equal or different lengths. Figure 1.3 depicts the schematic diagram of a parallel adder.

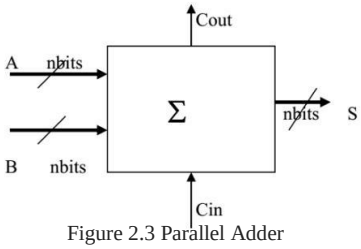


Figure 2.3 Parallel Adder

2.3.1 RIPPLE CARRY ADDER

The ripple carry adder is made up of full adder (FA) blocks that are stacked one on top of the other. One complete adder is responsible for adding two binary digits at any point during the ripple carry. The carryout from one stage is directly fed into the carry-in of the next. . A sum and a carry out are produced by each bit 6 addition. The carry out is subsequently sent to the following higher-order bit's carry in. The final result is a four-bit sum with a carry out (c4).

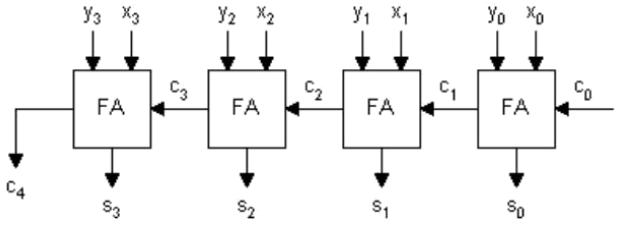


Figure 2.4 Parallel Adder: $-bit Ripple Carry Adder

2.4 CARRY SKIP ADDER

A carry-skip adder (also known as a carry-bypass adder) is a type of adder that, when compared to other adders, enhances the delay of a ripple-carry adder with minimal effort. When a basic one-level carry-ripple-adder encounters its worst-case situation.

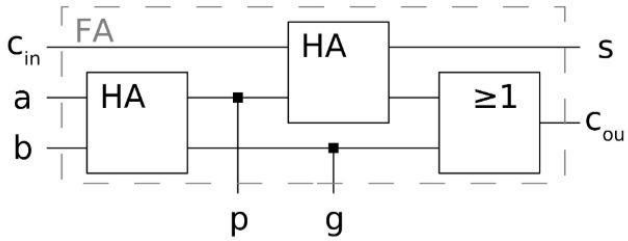


Figure 2.5 1-Bit Carry Skip Adder

A n-bit-carry-skip adder is made up of an n-bit-carry-ripple-chain, an n-input AND-gate, and a multiplexer. The n-input AND-gate is connected to each propagate bit pi given by the carry-ripple-chain. The resultant bit is used as the select bit of a multiplexer, which switches the last carry-bit cn or the carry-in signal c0 to the carry-out signal cout.

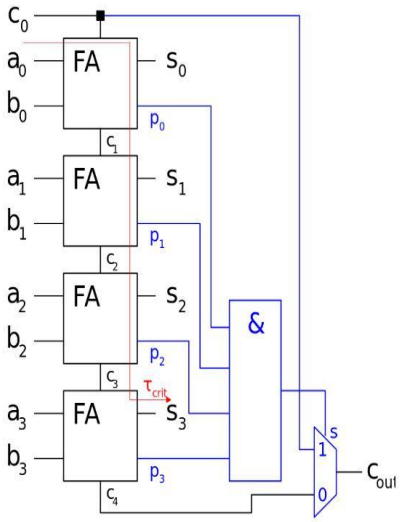


Figure 2.6 4-Bit Carry Skip Adder

2.5 CARRY BYPASS ADDER

Like a ripple-carry adder, every full adder cell must wait for the incoming carry before creating an outgoing carry. This reliance can be alleviated by speeding up the adder's operation by adding a bypass (skip). An incoming carry Ci,0=1 propagates across the entire adder chain and creates an outgoing carry C0,7=1 under the assumption that all propagation signals are 1. This information may be utilized to speed up the adder's operation, as shown in Figure 3.7. When BP = P0P1P3P4P5P6P7P8 = 1, the incoming carry is transmitted over the bypass to the next block immediately; otherwise, the carry is obtained using the traditional way. If (P0P1P3P4P5P6P7 = 1), C0,7 = Ci,0, with each group being "bypass" by a multiplexer if all of its full adders are propagating.

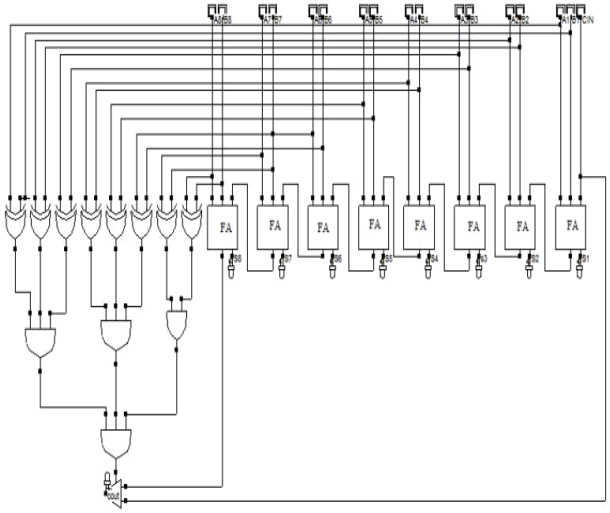


Figure 2.7 Carry Bypass Adder

2.6 KOGGE STONE ADDER

It is basically a parallel prefix adder. This sort of adder excels at making the quickest additions depending on design time. It's well-known for its unique and quick additions depending on design time. The propagate signals "Pi" and generate signals "Gi" are computed using the ith bit of the provided input. Similarly, these generated signals produce output carry signals.

Pre processing

Pi = Ai XOR Bi

Gi = Ai AND Bi

Generation of carry

G = Gi OR (Pi AND Pj)

P = Pi AND Pj

Final computation.

Si = Pi XOR Ci-1 Ci = Gi

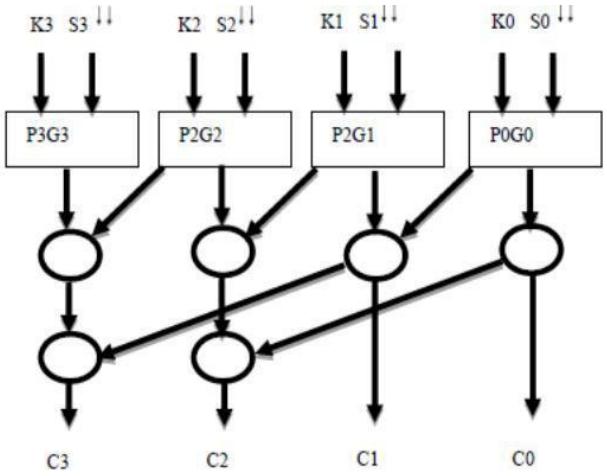


Figure 2.8 Structure of Kogge Stone Adder

3.MULTIPLIER

In most digital signal processing (DSP) systems, a multiplier is one of the most important hardware components. Digital filtering, digital communications, and spectrum analysis are examples of DSP applications where a multiplier plays a key role (Ayman. Aetal (2001)). Because many modern DSP applications are aimed at portable, battery-powered systems, power dissipation becomes a major design consideration. Because multipliers are complicated circuits that must often function at a high system clock rate, lowering the multiplier's delay is critical to the entire design's success.

Multiplications are quite costly and slow down the entire process. The speed with which a multiplication operation can be performed is often a determining factor in the performance of many computer problems. Consider two unsigned binary values, X and Y, each of which is M bits wide. It is

helpful to define X and Y in binary form to explain the multiplication process.

A single two-input adder is the simplest technique to achieve multiplication. The multiplication tasks M cycles using an N-bit adder for inputs that are M and N bits wide. This shift-and-add multiplication algorithm sums up M partial products. Each partial product is created by multiplying the multiplicand by a multiplier bit and shifting the result in the basis of the multiplier bit's position. Binary multiplication works similarly to long-hand decimal multiplication in that it adds shifted copies of the multiplicand dependent on the value and location of each multiplier bit. Binary multiplication is, in reality, considerably easier to accomplish than decimal multiplication. Each digit of a binary number can only have a value of 0 or 1, hence it depends on the value of the binary number. The partial products can only be a duplicate of the multiplicand or 0 because of the multiplier bit.This is merely an AND function in digital logic.

A method akin to manually computing a multiplication is a faster way to implement multiplication. The complete partial product is created all at once and stored in an array. The final product is computed using a multi-operand addition. The procedure is depicted in Figure 4.1.

The Wallace Tree

```
      1 0 1 0 1 0   Multiplicand (N bits)
    x     1 0 1 1   Multiplier   (M bits)
      1 0 1 0 1 0
    1 0 1 0 1 0     Partial Products
  0 0 0 0 0 0
1 0 1 0 1 0
1 1 1 0 0 1 1 1 0   Product
```

Figure 1 Example of manual multiplication

This collection of operations can be directly implemented in hardware The resulting structure is referred to as an array multiplier, and it combines three functions: partial product generation, partial product accumulation, and final addition

3.1 MULTIPLIER DESIGN

The design of a 16 x 16 multiplier is presented in this section. Using the proposed four 15-4 compressors, four approximate multipliers are designed. In addition, one precise multiplier and four approximate multipliers are taken into account.
Approximate multipliers based on the proposed approximate 15-4 compressors are compared to correct 16 16 multipliers based on accurate 15-4 compressors, as well as other multipliers based on other approximate compressors. The design of a 16bit multiplier utilising a 15-4 compressor is shown in the diagram, where each dot represents one partial product.

In the partial product reduction, six 15-4 compressors are utilised to create one multiplier, and then four multipliers are designed. Rectangular boxes in the illustration denote the employment of 15-4 and 4-2 compressors in the multiplier. From the 13th column onwards, 15-4 compressors are utilised in the multiplier.

There are only thirteen partial products in column 13 of the multiplier. To utilise the 15-4 compressor, two zeros are added to that column. In the 14th column, one "0" is also added. For partial product reduction, in addition to the 15-4 compressors in the multiplier, other accurate compressors such as 4-2 and half, full adders are utilised. In the 13th, 14th, and 15th columns of multipliers, approximate compressors are utilised. In most cases, using approximation compressors would result in a higher mistake rate. In the multiplier, a 15-4 approximation compressor is employed. 1.Similarly, design half adder (1), full adder (2), 5-3 compressor (3) and 15-4 approximate compressors (4) are used in multiplier 1, 2, 3 and 4 respectively. In approximate multiplier, all approximate 15-4 compressors are used along with accurate 2-2 and 3-2 compressors. Approximate 15-4 compressors, 5-3 compressors, half and full adders are used in second and third stage of partial product reduction tree. Parallel adders, carry skip adders, carry bypass adders, and kogge stone adders are employed to calculate the final result at the last step.

3.2 WALLACE TREE MULTIPLIER

The partial-sum adders can also be reorganised in a tree-like pattern, lowering the critical path as well as the number of adder cells required. The Wallace tree multiplier is the name of the structure provided, and its implementation is depicted in figure 4.2. For bigger multipliers, the tree multiplier results in significant hardware savings. The propagation latency is also minimised. While the Wallace multiplier is significantly quicker than the carry-save structure for high multiplier word lengths, it has the disadvantage of being irregular, which makes efficient layout design more difficult.

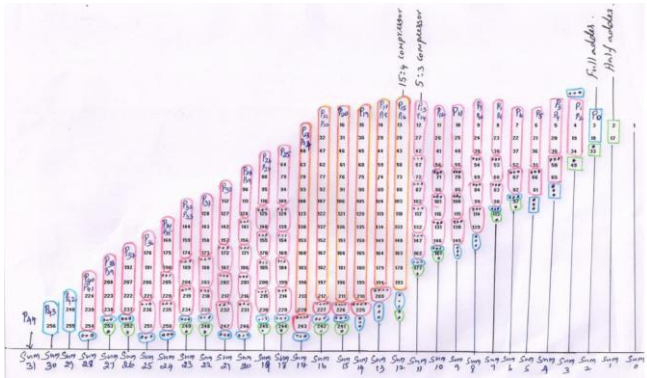


Figure 4.2 Structure of Wallace tree multiplier

4.SIMULATIONS AND RESULTS

6.1 MULTIPLIER

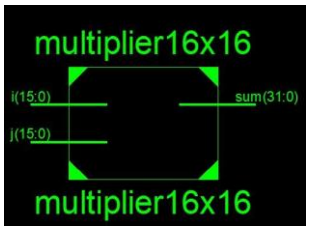


Figure 6.1 Top Module of Multiplier

6.2 MULTIPLIER USING PARALLEL ADDER

The below fig: 6.2 shows the area utilization for parallel adder. Here, totally there are 5720 LUT‟s (Look Up Table) available. On that we use only 682 LUT‟s. It consumes more area than compared to the Kogge stone adder and also it has higher delay than Kogge stone adder. In the diagram below, the delay will be displayed 6.3. Therefore, the fig: 6.4 explains about the RTL schematic view for parallel adder and fig: 6.5 explains about the waveform for the parallel adder.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	682	5720	11%
Number of fully used LUT-FF pairs	0	682	0%
Number of bonded IOBs	64	102	62%

Figure 6.2 Area Utilization for Parallel Adder

Figure 6.5 Waveform for Parallel Adder

6.3 MULTIPLIER USING CARRY SKIP ADDER

The below fig:6.6 to fig:6.9 explains about the area, delay, waveform and RTL schematic view for Carry Skip Adder using Multiplier. The total available LUT's are 5720 but the carry skip adder consumes only 546 LUT‟s. In comparison to the Kogge Stone adder, it has a longer delay and consumes a larger area.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	546	5720	9%
Number of fully used LUT-FF pairs	0	546	0%
Number of bonded IOBs	64	102	62%

Figure 6.6 Area Utilization for Carry Skip Adder

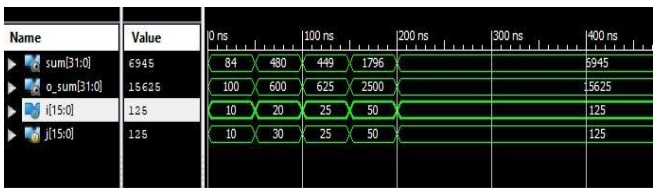


Figure 6.7 Waveform for Carry Skip Adder

6.4 MULTIPLIER USING CARRY BYPASS ADDER

The below fig: 6.10 to fig: 6.13 explains about the area, delay, waveform and RTL schematic view for Carry Bypass Adder using Multiplier. The total available LUT"s are 5720 but the carry bypass adder consumes only 519 LUT"s. It has same area as Kogge Stone adder but it has higher delay than Kogge Stone Adder.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	519	5720	9%
Number of fully used LUT-FF pairs	0	519	0%
Number of bonded IOBs	64	102	62%

Figure 6.10 Area Utilization for Carry Bypass Adder

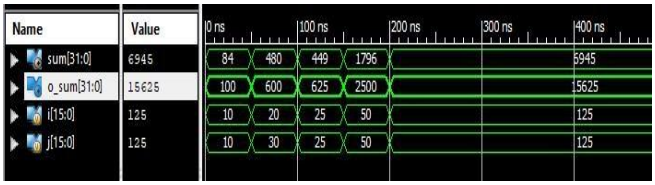


Figure 6.12 Waveform for Carry Bypass Adder

6.5 MULTIPLIER USING KOGGE STONE ADDER

The fig: 6.14 to fig: 6.17 shows the area, delay, waveform and RTL schematic view for Kogge Stone Adder. When compared to other adders, it uses less space and has a shorter latency. As a consequence, Kogge Stone Adder is the best when compared to other adders.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	519	5720	9%
Number of fully used LUT-FF pairs	0	519	0%
Number of bonded IOBs	64	102	62%

Figure 6.14 Area Utilization for Kogge Stone Adder

Figure 6.15 Waveform for Kogge Stone Adder

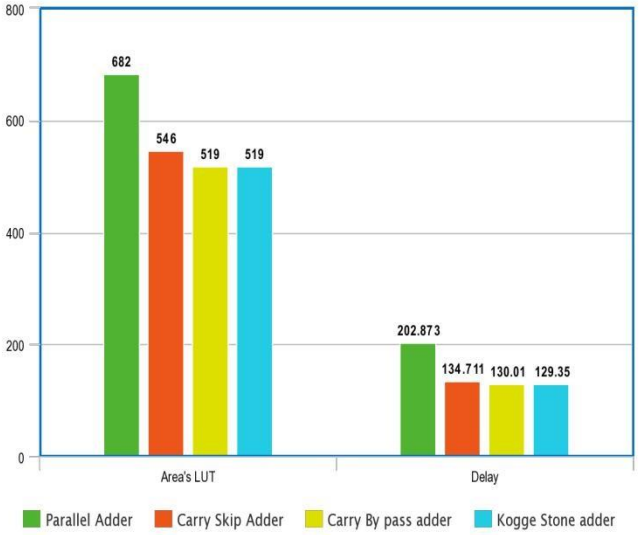


Figure 6.18 Chart analysis for types of multiplier using adder

Conclusion

The approximate 16-bit Wallace tree multiplier was simulated in Xilinx ISE 14.7. The proposed Kogge stone adder multiplier is contrasted to a multiplier with the same design but a different adder. A 16x16 multiplier with a 15-4 compressor and a Kogge stone adder is faster than a multiplier with a

parallel adder, a carry skip adder, a carry By-pass adder, and a Kogge Stone adder, as can be deduced. The suggested multiplier's performance can be increased in the future and used in applications such as image and video processing.

REFERENCES

[1] "Marimuthu R, Member, IEEE, Elsie Rezinold Y, and P.S Mallick, Senior Member, IEEE"Design and Analysis of multiplier using approximate 15-4 compressor, IEEE Access, 2019

[2] C. S. Wallace, A Suggestion for a Fast Multiplier, IEEE Transactions on Computers, 13, 1964,14-17.

[3] K. Bhardwaj, P. S. Mane, and J. Henkel, „„Power-and area-efficient approximate Wallace tree multiplier for error-resilient system, ""in Proc.15thInt. Symp. Quality Electron. Design (ISQED), Mar. 2014, pp. 263–269.

[4] C.-H. Lin and I.-C. Lin"High accuracy approximate multiplier with error correction", " in Proc. IEEE 31st Int. Conf. Comput. Design (ICCD), Oct. 2013, pp. 33–38.

[5] D. R. Gandhi, and N. N. Shah, Comparative Analysis for Hardware Circuit Architecture of Wallace Tree Multiplier, IEEE International Conference on Intelligent Systems and Signal Processing, Gujarat, 2013, 1-6.

[6] R. Menon and D. Radhakrishnan, „„High performance 5:2 compressor architectures, "" Proc. IEE-Circuits, Devices Syst., vol. 153, no. 5, pp. 447–452, Oct. 2006.

[7] R. Marimuthu, M. Pradeepkumar, D. Bansal, S. Balamurugan, and P. S. Mallick, „„Design of high speed and low power 15-4 compressor, " in Proc. Int.

Conf. Commun. Signal Process. (ICCSP), Apr. 2013, pp. 533– 536.

[8] Teffi Francis, Tera Joseph and Jobin K Antony., "Modified MAC Unit for low power high speed DSP application using multiplier with bypassing technique and optimized adders", IEEE-31661, 4th ICCCNT, 2013.

[9] Yezerla, Sudheer Kumar, and B. Rajendra Naik. "Design and Estimation of delay, power and area for Parallel prefix adders."

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

[11] Abdoreza Pishvaie, Ali Jahanian, "Improved CMOS (4:2) compressor designs for parallel multipliers", vol. 38, pages: 1703-1716, Nov 2012.

[12] Shima Mehrabi, Reza Faghih Mirzaee, Keivan Navi, "Design, analysis, and implementation of partial product reduction phase by using wide m: 3 (4≤m≤10) compressors", International Journal of HighPerformance Systems Architecture, Vol.4, page: 231-241, 2013.