International Journal of
Advanced Technology and Innovative Research

www.ijatir.org

ISSN 2348–2370
Vol.07,Issue.08,
July-2015,
Pages:1397-1402

Implementation of 64-Bit Modified Wallace MAC Based On Multi-Operand Adders

MIDDE SHEKAR[1], M. SWETHA[2]

[1]PG Scholar, Siddhartha Institute of Technology & Sciences, Hyderabad, TS, India.

[2]Assistant Professor, Siddhartha Institute of Technology & Sciences, Hyderabad, TS, India.

Abstract: Although redundant addition is widely used to design parallel multioperand adders for ASIC implementations, the use of redundant adders on Field Programmable Gate Arrays (FPGAs) has generally been avoided. The main reasons are the efficient implementation of carry propagate adders (CPAs) on these devices (due to their specialized carry-chain resources) as well as the area overhead of the redundant adders when they are implemented on FPGAs. This project presents different approaches to the efficient implementation of generic carry-save compressor trees. In computing, especially digital signal processing, the multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC, or MAC unit); the operation itself is also often called a MAC or a MAC operation. Power dissipation is one of the most important design objectives in integrated circuit, after speed. Digital signal processing (DSP) circuits whose main building block is a Multiplier-Accumulator (MAC) unit. High speed and low power MAC unit is desirable for any DSP processor. This is because speed and throughput rate are always the concerns of DSP system. MAC unit consists of adder, multiplier, and an accumulator it preserves a unique mapping between input and output vector of the particular circuit. In this MAC operation is performed in two parts Partial Product Generation (PPG) circuit and Multi-Operand Addition (MOA) circuit. In the proposed scheme, we are using Modified Wallace tree multiplier which reduces the hardware complexity. As the proposed system requires less number of resources, we optimize the power consumption. In this project, a new MAC is designed based on modified Wallace tree multiplier along with Multi operand adder.

Keywords: MAC, Modified Wallace Tree Multiplier, Cpas, Xilinx ISE, Verilog.

I. INTRODUCTION

The main reasons are the efficient implementation of carry propagate adders (CPAs) on these devices (due to their specialized carry-chain resources) as well as the area overhead of the redundant adders when they are implemented on FPGAs. Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) .This paper presents different approaches to the efficient implementation of generic carry-save compressor trees on FPGAs. They present a fast critical path, independent of bit width, with practically no area overhead compared to CPA trees. Along with the classic carry-save compressor tree, we present a novel linear array structure, which efficiently uses the fast carry-chain resources. This approach is defined in a parameterizable HDL code based on CPAs, which makes it compatible with any FPGA family or vendor. A detailed study is provided for a wide range of bit widths and large number of operands. Compared to binary and ternary CPA trees, increases speed ups for 16-bit width.

II. COMPRESSORS

A multiplier is one of the key hardware blocks in most digital and high performance systems such as FIR filters, digital signal processor, microprocessors etc. With advances in technology, many researchers have tried and strive to design multipliers which offer either of the following- high speed, low power consumption, less area combination of them in multipliers, thus making them compatible for various high speed, low power, and compact VLSI implementations. However, area and speed are two conflicting constraints. Therefore, improving speed always results in larger area. The most efficient multiplier structure will vary depending on the throughput requirement of the application. The first step of the design process is the selection of the optimum circuit structure. The combined factors of low power, low transistor count and minimum delay makes the 5:2 and 4:2 compressors, the appropriate choice. In these compressors, the outputs generated at each stage are efficiently used by replacing the XOR blocks with multiplexer blocks .The select bits to the multiplexers are available much ahead of the inputs so that the critical path delay is minimized. The various adder structures in the conventional architecture are replaced by compressors. The use of two full adders would introduce a delay of 4 whereas the use of 4:2 compressors reduces the latency to 3 as shown in Fig.1. Two full adders are replaced by a single 4:2 compressor.

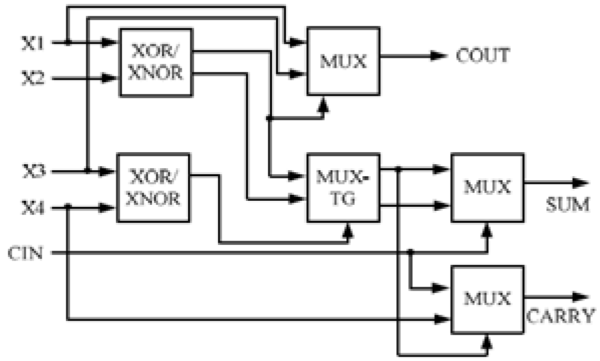


Fig.1. 4:2 Compressor.

III. Wallace Tree Multiplier

A Wallace tree multiplier is an efficient hardware implementation of a digital circuit that multiplies two integers devised by an Australian computer scientist Chris Wallace. Wallace tree reduces the no. of partial products and use carry select adder for the addition of partial products.

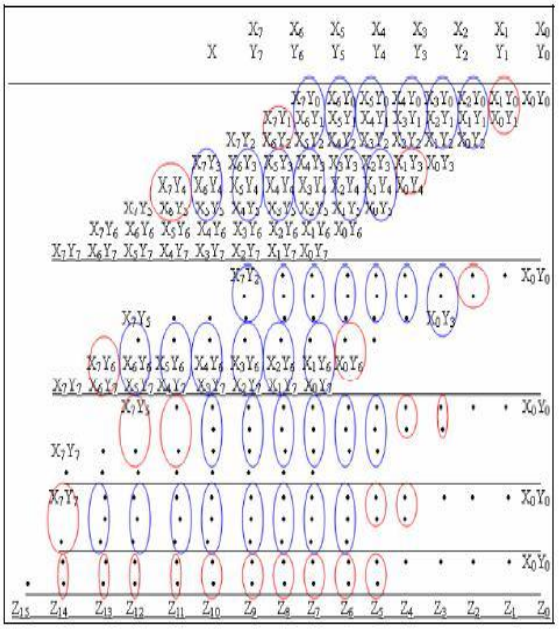

X
Y
Z_{15} Z_{14} Z_{13} Z_{12} Z_{11} Z_{10} Z_9 Z_8 Z_7 Z_6 Z_5 Z_4 Z_3 Z_2 Z_1 Z_0

Fig.2. 8 bit×8 bit wallace tree multiplier.

In this fig.2 blue circle represent full adder and red circle represent the half adder. Wallace tree has three steps. Multiply each bit of multiplier with same bit position of multiplicand. Depending on the position of the multiplier bits generated partial products have different weights. Reduce the number of partial products to two by using layers of full and half adders. After second step we get two rows of sum and carry, add these rows with conventional adders. As long as there are three or more rows with the same weight add following layers. Take any three rows with the same weights and input them into a full adder. The result will be an output row of the same weight i.e. sum and an output row with a higher weight for each three input wires i.e. carry. If there are two rows of the same weight left, input them into a half adder. If there is just one row left, connect it to the next layer. The advantage of the Wallace tree is that there are only O(log n) reduction layers (levels), and each layer has O(1) propagation delay. As making the partial products is O(1) and the final addition is O(log n), the multiplication is only O(log n), not much slower than addition (however, much more expensive in the gate count). For adding partial products with regular adders would require O(log n2) time.

IV. MODIFIED WALLACE TREE MULTIPLIER

A modified Wall ace multiplier is an efficient hardware implementation of digital circuit multiplying two integers. Generally in conventional Wallace multipliers many full adders and half adders are used in their reduction phase. Half adders do not reduce the number of partial product bits. Therefore, minimizing the number of half adders used in a multiplier reduction will reduce the complexity. Hence, a modification to the Wallace reduction is done in which the delay is the same as for the conventional Wallace reduction. The modified reduction method greatly reduces the number of half adders with a very slight increase in the number of full adders. Reduced complexity Wall ace multiplier reduction consists of three stages. First stage the N x N product matrix is formed and before the passing on to the second phase the product matrix is rearranged to take the shape of inverted pyramid. During the second phase the rearranged product matrix is grouped into non-overlapping group of three as shown in the figure 2, single bit and two bits in the group will be passed on to the next stage and three bits are given to a full adder. The number of rows in the in each stage of the reduction phase is calculated by the formula

$$r_{j+1}= 2[r_j/3]+r_j\mathrm{mod}3 \quad (1)$$

$$\text{If } r_j \bmod 3 = 0, \text{ then } r_{j+1} = 2r/3 \quad (2)$$

If the value calculated from the above equation for number of rows in each stage in the second phase and the number of row that are formed in each stage of the second phase does not match, only then the half adder will be used. The final product of the second stage will be in the height of two bits and passed onto the third stage. During the third stage the output of the second stage is given to the carry propagation adder to generate the final output. Thus 64 bit modified Wallace multiplier is constructed and the total number of stages in the second phase is 10. As per the equation the number of row in each of the 10 stages was calculated and the use of half adders was restricted only to the 10th stage. The total number of half adders used in the second phase is 8 and the total number of full adders that was used during the second phase is slightly increased that in the conventional Wallace multiplier as shown in Fig.3. Since the 64 bit modified Wallace multiplier is difficult to represent, a typical l0-bit by 10-bit reduction shown in fig.4 for understanding. The modified Wallace tree shows better performance when carry save adder is used in final stage instead of ripple carry adder. The carry save adder which is used is considered to be the critical part in the multiplier because it is responsible for the largest amount of computation.

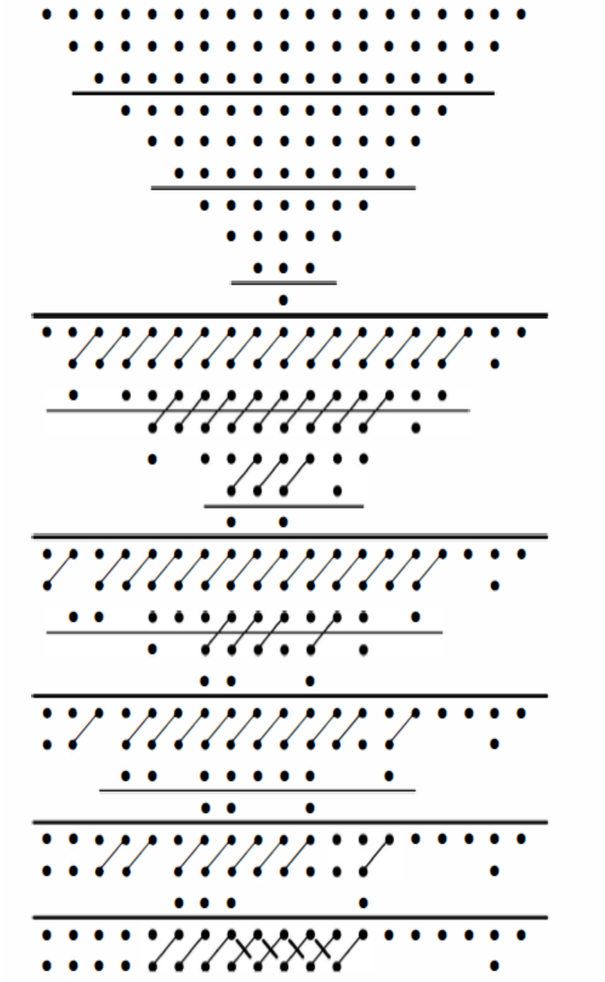


Fig.3. Modified Wallace Reduction Process.

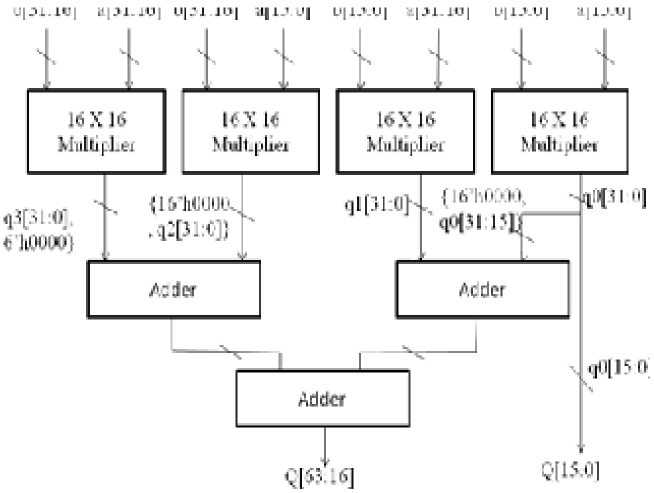


Fig.4.Block diagram of modified wallace tree multiplier.

V. REGULAR CS COMPRESSOR TREE DESIGN

The classic design of a multi operand CS compressor tree attempts to reduce the number of levels in its structures as shown in Figs.5 to 10. The 3:2 counters or the 4:2 compressors are the most widely known building blocks to implement it.

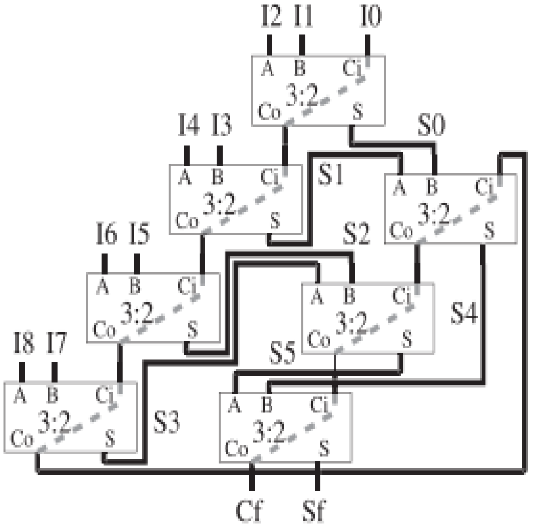


Fig.5. n-bit width cs 9:2 compressor tree based on a linear array.

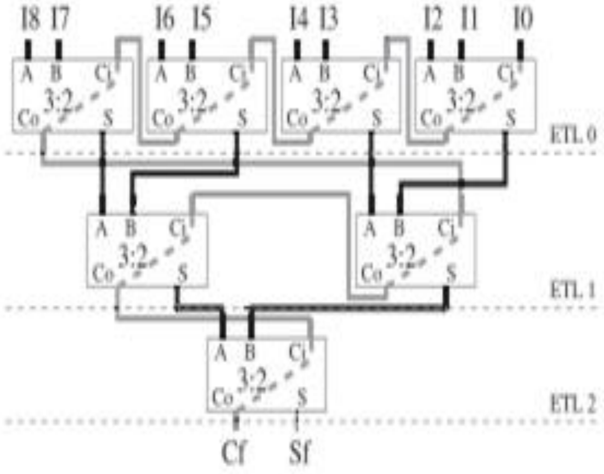


Fig.6. Time model of the proposed cs 9:2 compressor tree.

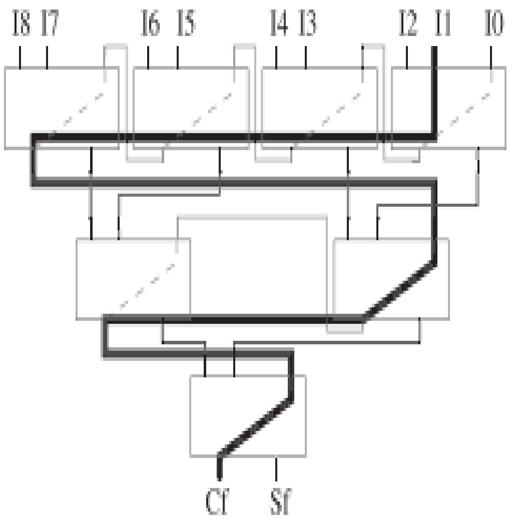


Fig.7. Critical path of the proposed 9:2 compressor tree for linear array behavior.

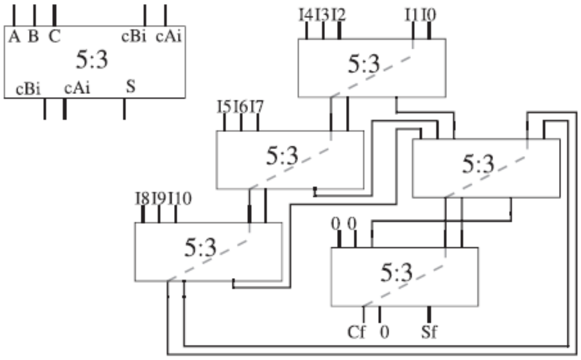


Fig.8. 5:3 compressor.

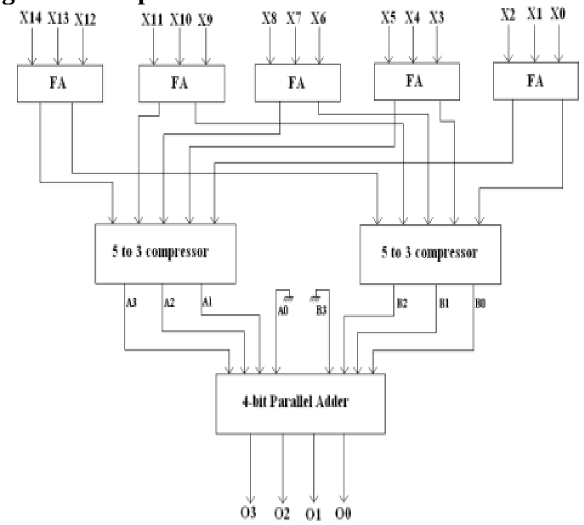


Fig.9. 15:4 compressor.

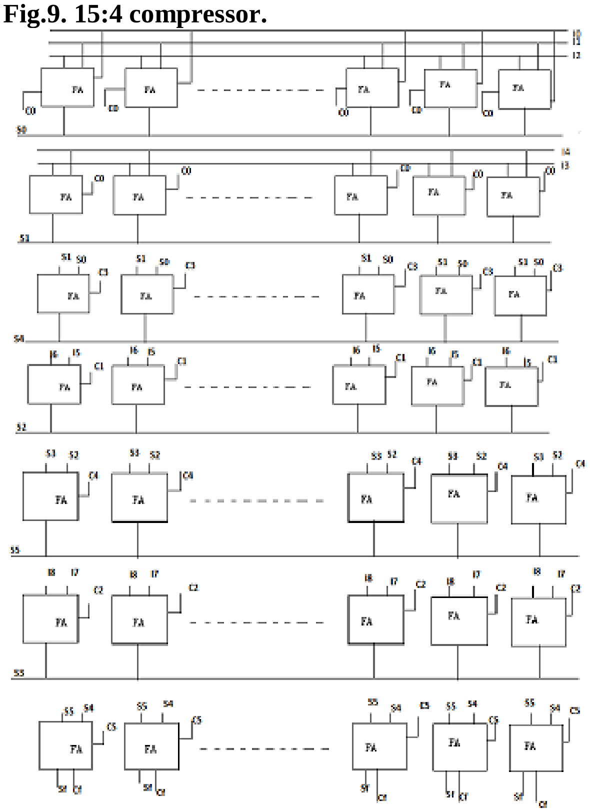


Fig.10. Transformation of n-bit width 9:2 linear array compressor tree.

VI. ARCHITECTURE OF MAC UNIT

Multiplier-Accumulator (MAC) operation is an important operation for many DSP and video processing applications. On FPGAs, multi-input addition has traditionally been implemented using trees of carry-propagate adders. This approach has been used because the traditional look up table (LUT) structure of FPGAs is not amenable to compressor trees, which are used to implement multi-input addition and parallel multiplication in ASIC technology. In prior work, we developed a greedy heuristic method to map compressor trees onto the general logic of an FPGA. Although redundant addition is widely used to design parallel multi operand adders for ASIC implementations, the use of redundant adders on Field Programmable Gate Arrays (FPGAs) has generally been avoided. MAC unit is an inevitable component in many digital signal processing (DSP) applications involving multiplications and/or accumulations. MAC unit is used for high performance digital signal processing systems. The DSP applications include filtering, convolution, and inner products. Most of digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) or discrete wavelet transforms (DWT). Because they are basically accomplished by repetitive application of multiplication and addition, the speed of the multiplication and addition arithmetic determines the execution speed and performance of the entire calculation.

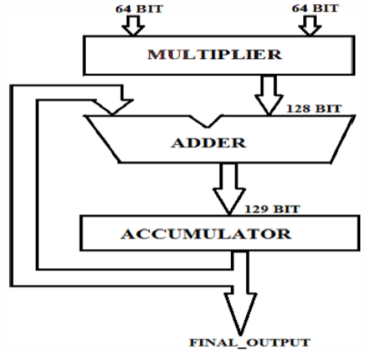


Fig.11. Architecture of 64 Bit Mac.

Multiplication-and-accumulate operations are typical for digital filters. Therefore, the functionality of the MAC unit enables high-speed filtering and other processing typical for DSP applications. Since the MAC unit operates completely independent of the CPU, it can process data separately and thereby reduce CPU load. The application like optical communication systems which is based on DSP, require extremely fast processing of huge amount of digital data. The Fast Fourier Transform (FFT) also requires addition and multiplication. 64 bit can handle larger bits and have more memory as shown in Fig.11. A MAC unit consists of a multiplier and an accumulator containing the sum of the previous successive products. The MAC inputs are obtained from the memory location and given to the multiplier. A multiplier is one of the key hardware blocks in most digital and high performance systems such as FIR filters, micro processors and digital signal processors etc. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the whole system and also it is occupying more area consuming.

VII. SIMULATION RESULTS

Simulation results of this paper is shown in bellow Figs.12 to 17.

A. Schematic Diagrams of different Compressors

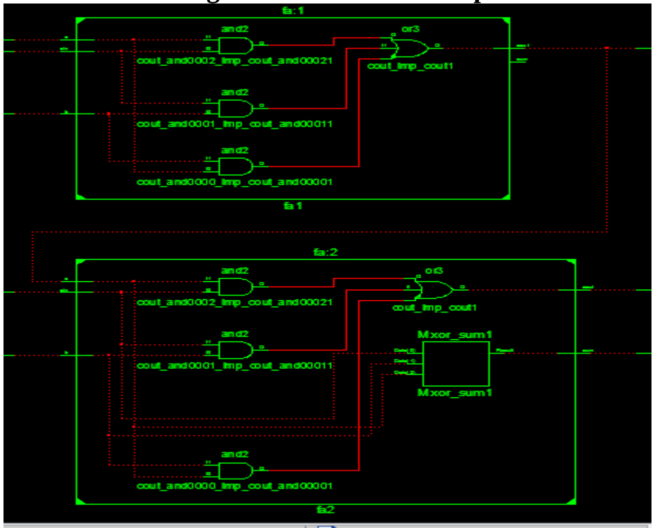

fa:1
and2
or3
cout_and0002_imp_cout_and00021
cout_imp_cout1
and2
cout_and0001_imp_cout_and00011
and2
cout_and0000_imp_cout_and00001
fa1
fa:2
and2
or3
cout_and0002_imp_cout_and00021
cout_imp_cout1
and2
Mxor_sum1
cout_and0001_imp_cout_and00011
Mxor_sum1
and2
cout_and0000_imp_cout_and00001
fa2

Fig.12. Schematic diagram of 5:3 compressor.

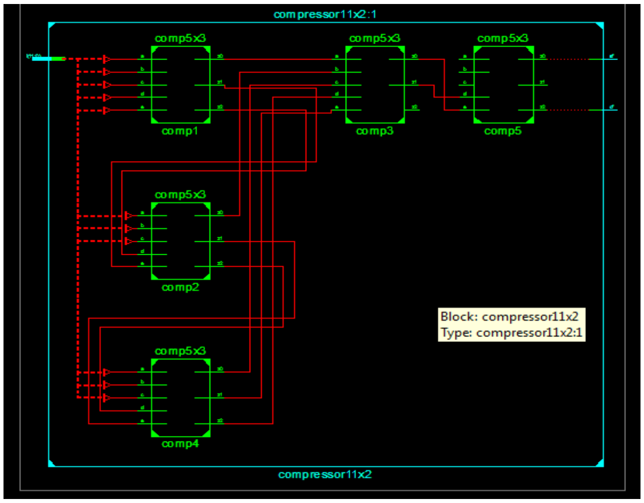


Fig .13. Schematic diagram of 11:2 compressor.

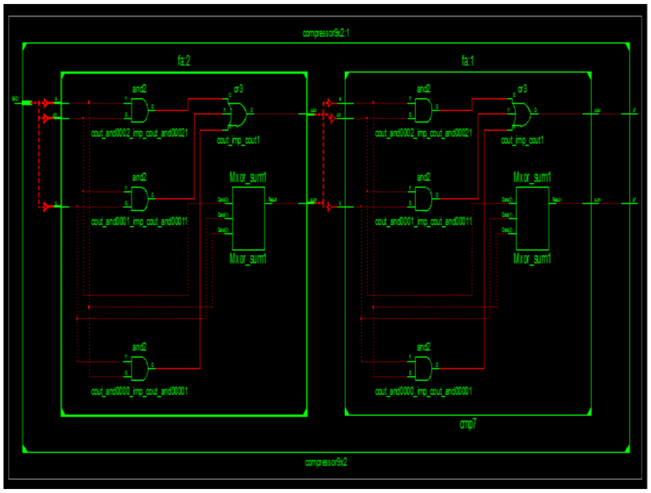

compressor9x2:1
fa2
fa1
and2
or3
cout_and0002_imp_cout_and00021
cout_imp_cout1
and2
Mxor_sum1
cout_and0001_imp_cout_and00011
Mxor_sum1
and2
cout_and0000_imp_cout_and00001
and2
or3
cout_and0002_imp_cout_and00021
cout_imp_cout1
and2
Mxor_sum1
cout_and0001_imp_cout_and00011
Mxor_sum1
and2
cout_and0000_imp_cout_and00001
cmp7
compressor9x2

Fig.14. Schematic diagram of 9:2 compressor.

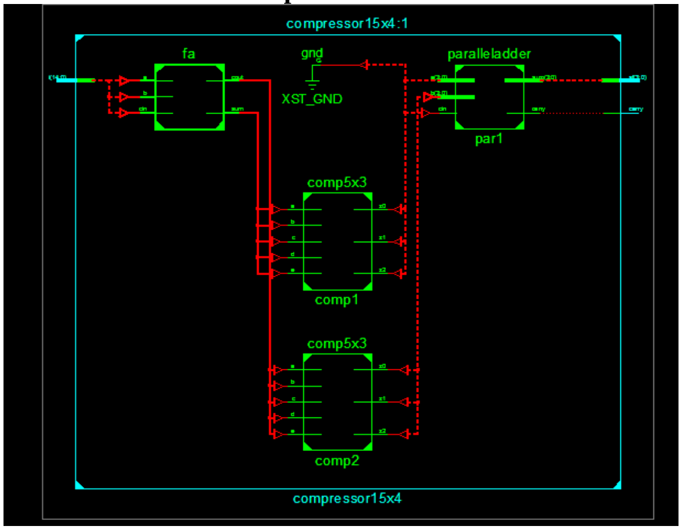


Fig.15. Schematic diagram of 15:4 compressor.

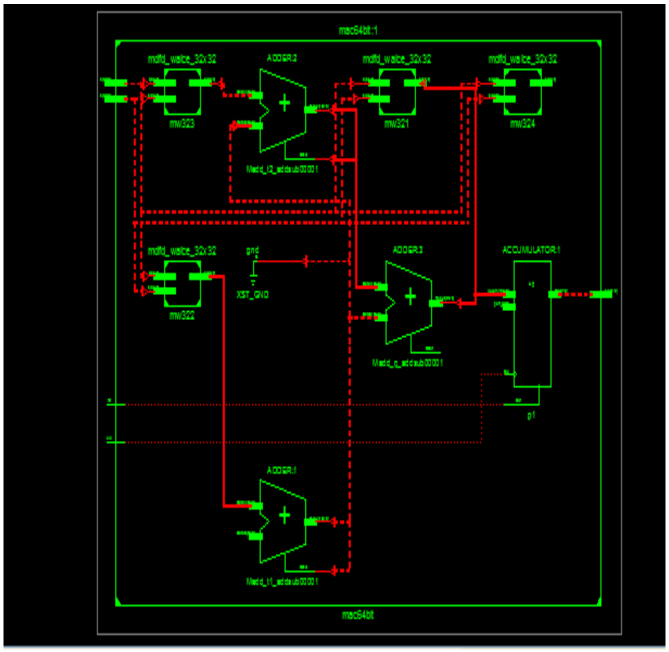
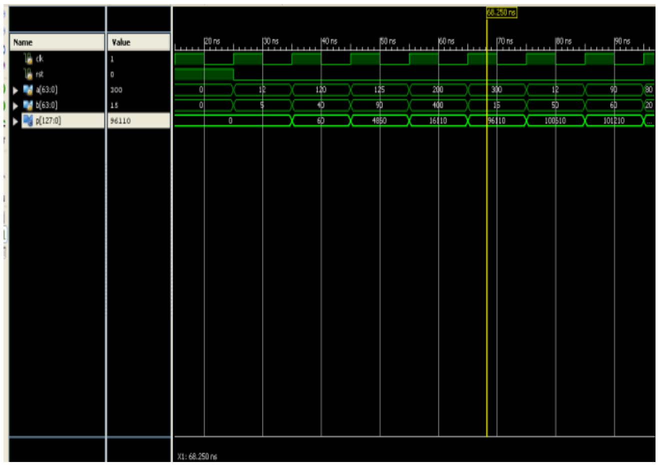


Fig.16. By using above compressor we designed a MAC unit.



Name	Value
clk	1
rst	0
a[63:0]	300
b[63:0]	15
p[127:0]	96110

Fig.17. Waveform.

VIII. CONCLUSION

Efficiently implementing MAC on FPGA, in terms of area and speed, is made possible by using the specialized carry-chains of these devices in a novel way. Similar to what happens when using ASIC technology, the proposed CS linear array compressor trees lead to marked improvements in speed compared to CPA approaches and, in general, with no additional hardware cost. Furthermore, the proposed high-level definition of CSA arrays based on CPAs facilitates ease-of-use and portability, even in relation to future FPGA architectures, because CPAs will probably remain a key element in the next generations of FPGA. AAs compare to conventional multiplier number of hardware components are less there by area over head can be reduced cost is less. In future we can extend it to implement as ALU. The functionality is verified through XILINX ISE using VERILOG HDL.

IX. REFERENCES

[1].Young-Ho Seo and Dong-Wook Kim, "New VLSI Architecture of Parallel Multiplier-Accumulator Based on Radix-2 Modified Booth Algorithm," IEEE Transactions on very large scale integration (vlsi) systems, vol. 18, no. 2,february 2010.

[2]. Ron S. Waters and Earl E. Swartzlander, Jr., "A Reduced Complexity Wallace Multiplier Reduction, " IEEE Transactions On Computers, vol. 59, no. 8, Aug 2010.

[3]. C. S. Wallace, "A suggestion for a fast multiplier," iEEE Trans. ElectronComput., vol. EC-13, no. I, pp. 14-17, Feb. 1964.

[4]. Shanthala S, Cyril Prasanna Raj, Dr.S.Y.Kulkarni, "Design and VLST Implementation of Pipelined Multiply Accumulate Unit," IEEE International Conference on Emerging Trends in Engineering and Technology, ICETET-09.

[5] B. Cope, P. Cheung, W. Luk, and L. Howes, “Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case Study,” IEEE Trans. Computers, vol. 59, no. 4, pp. 433-448, Apr. 2010.

[6] S. Dikmese, A. Kavak, K. Kucuk, S. Sahin, A. Tangel, and H. Dincer, “Digital Signal Processor against Field Programmable Gate Array Implementations of Space-Code Correlator Beamformer for Smart Antennas,” IET Microwaves, Antennas Propagation, vol. 4, no. 5, pp. 593-599, May 2010.

[7] F. Schneider, A. Agarwal, Y.M. Yoo, T. Fukuoka, and Y. Kim, “A Fully Programmable Computing Architecture for Medical Ultrasound Machines,” IEEE Trans. Information Technology in Biomedicine, vol. 14, no. 2, pp. 538-540, Mar. 2010..

[8] J.S. Kim, L. Deng, P. Mangalagiri, K. Irick, K. Sobti, M. Kandemir, V. Narayanan, C. Chakrabarti, N. Pitsianis, and X. Sun, “An Automated Framework for Accelerating Numerical Algorithms on Reconfigurable Platforms Using Algorithmic/Architectural Optimization,” IEEE Trans. Computers, vol. 58, no. 12, pp. 1654-1667, Dec. 2009.

[9] H. Lange and A. Koch, “Architectures and Execution Models for Hardware/Software Compilation and their System-Level Realization,” IEEE Trans. Computers, vol. 59, no. 10, pp. 1363- 1377, Oct. 2010.