FPGA Implementation of an Efficient High Speed Wallace Tree Multiplier

Mani Kunnathettu Rajee
Student ,ECE Department
Saint GITS College
Kottayam

Tessly Thomas
Student ,ECE
Saint GITS College
Kottayam

Alphy Manuel
Student ,ECE
Saint GITS College
Kottayam

Anju Rachel Thomas
Student ,ECE
Saint GITS College
Kottayam

Riboy Cheriyan
Assoc. Professor
Saintgits College
Kottayam

ABSTRACT
High speed multiplication is one of the critical function in a range of very large scale integration (VLSI) applications. Multipliers find their importance in performing operations such as convolution, filtering and correlation in digital signal processing systems, microprocessors and graphic engines. Multiplication is an expensive and slow operation. Designing multipliers that are of low power, regular in layout and with high-speed are of substantial research interest. The speed of the multiplier can be improved by reducing the generated partial products. Of the many attempts that have been made to reduce the number of partial products generated in a multiplication process, one of the notable one is the Wallace tree multiplier. Wallace Tree CSA structures have been used to sum the partial products in reduced time. The existing Wallace tree architecture has considerable speed but due to various limitations faced, there arises the need for further improvement.

In this project, the proposed system is a modified Wallace tree multiplier. In this paper Wallace tree multiplication is investigated and evaluated. Speed of traditional Wallace tree multiplier can be improved by using compressor techniques. Here the Wallace tree is constructed by conventional method with the help of compressors such as 3:2 compressor. Therefore, minimizing the number of adders used in a multiplier reduction will reduce the delay, thereby improving the speed of multiplication.

General Terms
Multipliers ,FPGA,adders

Keywords
Compressor, full adder ,partial product,Wallace tree

1. INTRODUCTION
A multiplier can be divided into further three stages: - Partial products generation (PPG) stage is the first stage in which the multiplicand and the multiplier are multiplied bit by bit to generate the partial products. Partial products addition stage or reduction of partial products (PPR) is the second stage which is the most important as it is the most complicated and that determines the speed of the overall multiplier and the final addition stage or carry-propagate addition (CPA) using different compressors have been widely used in the high speed multipliers to lower the latency of the partial product accumulation stage. In order to employ the processor for digital signal processing applications, a modified Wallace tree multiplier which uses compressors circuits to obtain low power and high speed operation in the Arithmetic Logic Unit (ALU) . In digital CMOS design, the well-known power-delay product is commonly used to evaluate the merits of designs.

2. PARTIAL PRODUCT GENERATION
The multiplier architecture comprises of a partial product generation stage, partial product reduction stage and the final addition stage. At each step, multiply one digit of the multiplier by the full multiplicand ,add the result , shifted by the proper number of bits, to the partial product. When the multiplier digits ends, the multiplication is done. Single-digit multiplication is easy for binary numbers – binary multiplication of two bits is performed by the AND function. The computation of partial products and their accumulation into the complete product can be done in many ways, but an understanding of the basic steps in multiplication is important to a full appreciation of those improvements. A high speed multiplier is an electronic computing system used to provide multiplication processes at a very high speed by incorporating various types of logic circuits. In the high speed multiplier, a partial product is obtained from the initial addition of the multiplicand and the multiplier. Then, the partial products are added together in order to obtain final product.

The multiplier architecture proposed comprises of a partial product generation stage (PPG), partial product reduction stage (PPR) and the final addition stage or carry-propagate addition (CPA). In the partial products reduction stage the latency of the Wallace tree multiplier can be reduced by decreasing the number of adder circuits. Realization and reduction in the number of partial product addition stages are using multi-bit compressors.

3. FULL ADDER
A full adder takes in three inputs and produce a sum and carry output. It can be a single full adder or a combination of half adders performing full adder function. The following figure represent the gate structure of a full adder circuit. A full adder operation involves two XOR delays to produce its output.

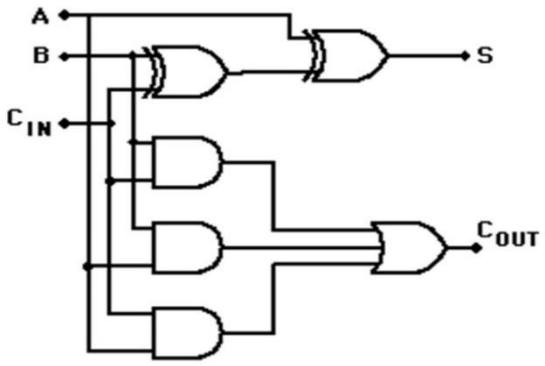

Figure1 : Full adder using logic gates[6]

4. ARRAY MULTIPLIER

Array multiplier has a simple and regular structure .The array unit, depicted in figure 2 form the basic unit of multiplier structure. Array multiplier is popular due its simplicity of design which can be done in modular fashion .The array multiplier unit takes in two inputs bits , one from multiplier and one from multiplicand, along with a sum in and carry in from the previous stages of operation . The partial product is generated and a full adder module adds the partial product with the sum in and carry in so as to produce an output sum and carry out. Which are used in the subsequent stages[4].

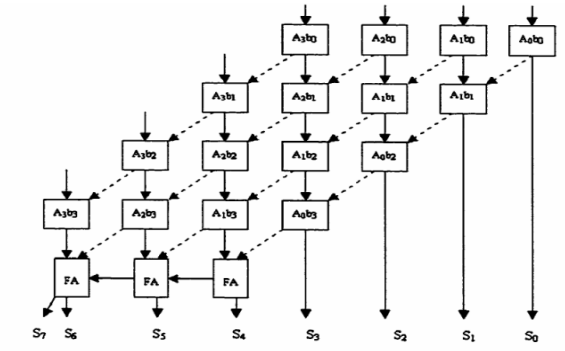

Figure 2: 4 Bit Array multiplier Structure[8]

5. WALLACE TREE MULTIPLIER

The currently existing method is a normal wallace tree multiplier.The Wallace tree multiplier is considered as faster than a simple conventional array multiplier and is considered to be an efficient implementation of a digital circuit which multiplies two integers. A Wallace tree multiplier is a parallel multiplier which use the carry save addition algorithm to reduce the propagation delay. There are many researchers working on the design of increasingly more efficient multipliers. They aim at achieving higher speed and lower power consumption and also reduced silicon area.

In the Wallace Tree algorithm, three bit signals are passed to a one bit full adder which is called the three input Wallace Tree circuit, and the output signal is supplied to the next stage full adder of the same bit, and the carry output signal is then passed to the next stage full adder of the same number of bit, and the carry output signal is then supplied to the next stage of the full adder located at a one bit higher position. Figure 4 shows the logic used for Wallace tree multiplication.

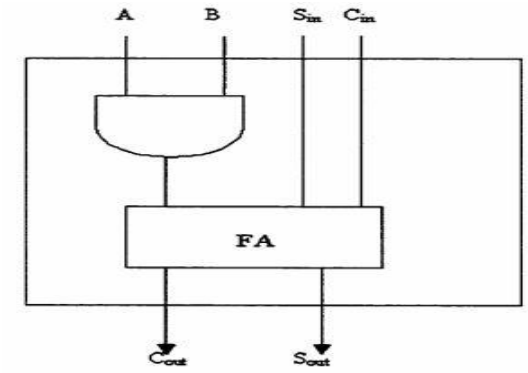

Figure 3: Basic unit of array multiplier[8]

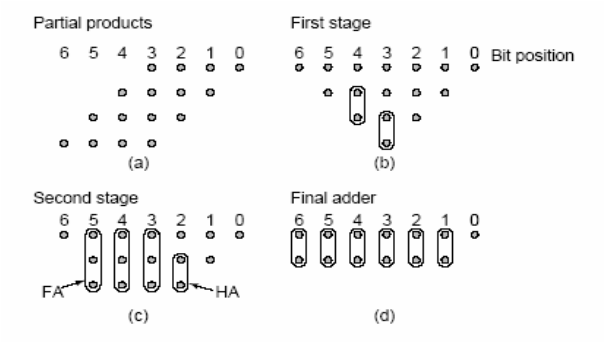

Figure 4 : Logic used for Wallace tree multiplier[2]

Figure 5 shows the block diagram of a Wallace tree multiplier. The first stage consists of 2 half adders and the second stage consists of 4 full adders. In the final stage the partial products are added using ripple carry adder.

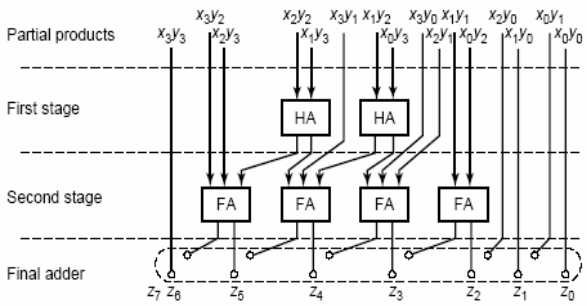

Figure 5: Block diagram of a Wallace tree multiplier[5]

6. LIMITATIONS OF THE EXISTING MULTIPLIERS

Digital multipliers are used in almost all electronic communication systems. Although a reduction in area and power dissipation results in an imminent increase in performance of the system, multiplication, being a complex, tedious and expensive process, speed will be the dominating factor in deciding the speed of computational operations. Hence it will be safe in assuming that speed of operation is the

determining factor in evaluating the performance of multiplier circuits.

An array multiplier is one of the most basic parallel multiplier circuits. It has a regular structure and its pipelined architecture is easy to design. But the major limitation of an array multiplier is its size. As operand sizes increase, arrays grow in size at a rate equal to the square of the operand size. This increase in area results in an increase in power consumption. Also the larger size will increase the propagation delay which is O(log n^2).

The Wallace tree multiplier is a adder tree comprising of carry save adders. It is used to obtain quick reduction of partial products. From various dimensions, the Wallace tree algorithm is one of the most efficient algorithms to be employed in digital multipliers. But because of its non-regularity, the layout of Wallace tree multipliers suffers from a large area wastage. Also the delay offered by the various computational units such as half adders, full adders and the final stage ripple carry adder will increase proportionally with the size of the multiplier. So it is necessary to substitute the above mentioned units with alternate computational units which provide a considerable reduction in propagation delay which is otherwise equal to O(log *n*), where *n* is the number of bits in the multiplier[2].

7. COMPRESSORS

A compressor is a device which is used to reduce the operands while adding terms of partial products in multipliers. An X:Y compressor takes X equally weighted input bits and produces Y-bit binary number. Compressors are generally designed by using XOR-XNOR gates and multiplexers. These compressors are used to minimise delay and area which results in increase in the performance of the overall system. The most widely and the simplest used compressor is the 3:2 compressor which is also known as a full adder. A 3:2 compressor has three inputs X1, X2, X3 and generates two outputs, the sum and the carry bits. The 3:2 Compressor is a combinational circuit which adds three binary inputs of one bit and returns sum and carry of one bit[1].

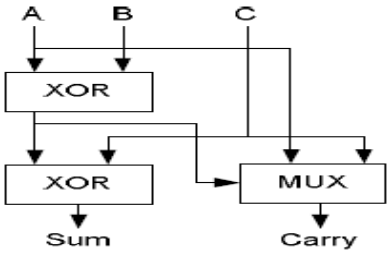

Figure 6: A 3:2 Compressor using universal gates[2]

It is used as a full adder. Figure 6 shows block diagram of the 3:2 compressor, it has three inputs X_1, X_2 and X_3 and two outputs sum and carry.

The equations governing the outputs of the 3:2 compressor architecture are shown below.

$$Sum = X_1 \oplus X_2 \oplus X_3$$

$$Carry = X_1 X_2 + X_2 X_3 + X_3 X_1$$

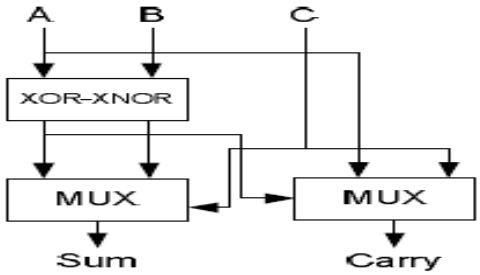

Figure 7:Block diagram of improved 3:2 compressor[2]

The 3:2 compressor architecture shown in figure 7 has less delay when compared to other architectures as one of the XOR circuits is replaced by a multiplexer circuit. In this compressor, the select bit at the multiplexer is present before the input arrives, which further reduces the delay. Thus the switching time of the transistors in the critical path is decreased. This minimizes the delay to a significant amount. This architecture denotes a critical path delay of Δ-XOR +Δ-MUX. Output functions of the modified 3:2 compressor circuit are shown by the equations given below:

$$Sum = (A \text{ XOR } B) \cdot \overline{C} + \overline{(A \text{ XOR } B)} \cdot C$$

$$Carry = (A \text{ XOR } B) \cdot C + \overline{(A \text{ XOR } B)} \cdot A$$

In this improved version a new block, the XOR-XNOR block, which facilitates the faster operation. The XOR-XNOR block takes in two inputs X and Y and produces two outputs as X XOR Y and X XNOR Y .The delay involved is only that of an XOR gate. This concept takes advantage of the CMOS design of the XOR gate which produces an XNOR output in its intermediate stage thus the two outputs are available after an XOR delay.

8. IMPROVED WALLACE TREE MULTIPLIER WITH COMPRESSORS

The block diagram shown above is the improved version of the Wallace tree multiplier incorporated using compressors. The compressors enhance the performance of the multiplier by reducing multiplier characteristics such as delay, area and power consumption. This is especially significant in the final stage of the Wallace tree structure where the ripple carry adder is present. The presence of the compressor will cause a reduction in delay for each of the adders of the ripple carry adder[5].

9. HDL USED- VERILOG

The principal feature of a hardware description language is that it contains the capability to describe the function of a piece of hardware independently of its implementation. The great advance with modern HDLs was the recognition that a single language could be used to describe the function of the design and also to describe the implementation. This allows the entire design process to take place in a single language, and thus a single representation of the design.

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits. Hardware description languages such as Verilog differ from other software programming languages because they include ways of describing the propagation time and signal strengths (sensitivity). A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output, and bidirectional port. There are different styles of modeling such as gate level, data flow, behavioral, structural modeling.

The various advantages of using Verilog HDL are as follows: HDL simulators are better than gate level simulators for mainly 2 reasons: portable model development, and the ability to design complicated test benches that react to outputs from the model under test. Finding a model for a unique component for your particular gate level simulator can be a frustrating task, with an HDL language you can always write your own model. Also most gate level simulators are limited to simple waveform based test benches which complicates the testing of bus and microprocessor interface circuits[4].

Verilog is a great low level language. Structural models are easy to design and Behavioral RTL code is pretty good. The syntax is regular and easy to remember. It is the fastest HDL language to learn and use. However Verilog lacks user defined data types and lacks the interface-object separation of the VHDL's entity-architecture model.

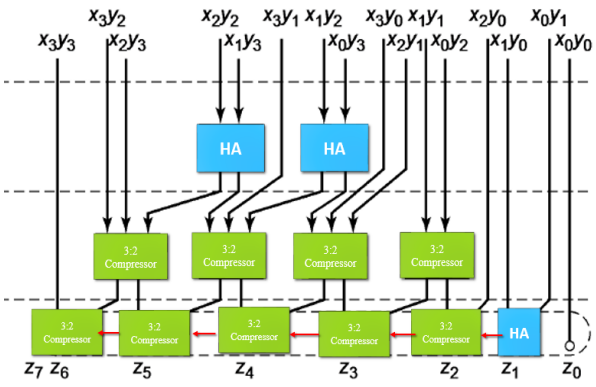

Figure 8: Improved Wallace Tree Block Diagram[1]

VHDL is good for designing behavioral models and incorporates some of the modern object oriented techniques. Its syntax is strange and irregular, and the language is difficult to use. Structural models require a lot of code that interferes with the readability of the model. **C++**as a hardware modeling language is excellent choice for high-level behavioral analysis of a system (like evaluating different data flow architectures in a microprocessor). However C++ lacks the basic hardware concepts like knowledge of strengths, connections, and concurrent execution which complicates model generation for lower level simulations.

10. SIMULATION TOOL XILINX ISE14.2

Xilinx ISE (Integrated Software Environment) 14.2 is also a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. It is an advanced version of Xilinx 8.1 with many additional features from the synthesis reports for various parameters such as propagation delay, number of LUT's used etc. is obtained.

11. ARTIX 7

FPGAs are programmable semiconductor devices that are based around a matrix of Configurable Logic Blocks (CLBs) connected through programmable interconnects. FPGAs can be programmed to the desired application or functionality requirements. FPGAs allow designers to change their designs very late in the design cycle– even after the end product has been manufactured and deployed in the field. In addition, Xilinx FPGAs allow for field upgrades to be completed remotely, eliminating the costs associated with re-designing or manually updating electronic systems.

The FPGA used is ARTIX 7 from XILINX. The various advantages of this FPGA includes lowest total power per logic cell, best-in-class performance for DDR3, DSP, parallel and serial I/O, low cost, small footprint packaging, part of a broad cost effective all programmable gate arrays.

12. ADVANTAGES AND LIMITATIONS

a. Reduced delay: With use of compressors the delay involved for addition reduced from two XOR delays to a single XOR delay and an AND delay per adder stage

b. Minimised area: As the number of LUTs used is reduced, area consumed will also be less.

c. Low power consumption: Area and power are directly related .So as area decreases power consumption also reduces.

d. Improved performance: Due to improved speed at reduced power the overall efficiency has improved.

Limitations:
Irregular topology:- In an irregular or parallel topology, they do not have a regular pattern for connecting the counters. This makes it difficult to make a layout the Wallace tree multiplier[7].

13. RESULTS AND DISCUSSION

After completing the comparison of array multiplier, normal Wallace tree multiplier and improved Wallace tree multiplier using compressor, it has been proved that using compressor could increase the speed of the multiplier. Performance measures of multipliers are obtained by synthesizing individual codes using Xilinx ISE 14.2From the table 1it can see that delay of 4 bit conventional Wallace tree multiplier is 11.109 ns. It has a percentage improvement of 23.05 over array multiplier which is having a delay of 14.437 ns. Whereas in case of 4 bit Wallace tree multiplier using compressor delay is 8.437 ns and has an improvement of 41.56% over array multiplier. Similarly in 8 bit Wallace tree multiplier using compressor delay is 11.314 ns .Here delay is reduced by 71.84% as compared to 8 bit array multiplier, having delay of 30.353 ns.

Table 1: Synthesis results using Xilinx 14.2

Sl.no	Multiplier	Bits	Total delay (ns)	% improvement over array multiplier
1	Array	4	14.437	23.05%
2	Wallace tree	4	11.109	41.56%
3	Improved Wallace tree	4	8.437	
4	Array	8	30.353	12.69%
5	Wallace tree	8	26.501	71.84%
6	Improved Wallace tree	8	11.314	

Table 2 shows the synthesis result of number of slice LUTs used in 4 and 8 bit multipliers. Slice LUTs used in a 4 bit array multiplier is 31, in a conventional Wallace tree multiplier it is 26 and 22 in Wallace tree using compressor. Number of LUTs used in Wallace tree multiplier using compressor is reduced by 29.03% , compared to array multiplier. In the case of an 8 bit array multiplier, 235 slice LUTs are used, for a normal Wallace tree it is 205 and in the case of an improved Wallace tree multiplier with compressor, the number of slice LUTs used are 128, a significantly lower value consumption as well. This results in a reduction to the number of slice LUTs used by 45.53%, leading to a significant reduction in power As the number of LUTs reduces area also reduces considerably. Area and power are directly related. As a result power reduces, so area also get reduced.

Table 2: Synthesis results showing slice LUTs used

Sl. No	Multiplier	Bits	Slice LUT Used	% improvement over array multiplier
1	Array	4	31	
2	Wallace tree	4	26	16.12%
3	Improved Wallace tree	4	22	29.03%
4	Array	8	235	
5	Wallace tree	8	205	12.76%
6	Improved Wallace tree	8	128	45.53%

14. CONCLUSION

The simulation and synthesis of multiplier is done in Xilinx ISM 14.2 and functionally tested in Modelsim with different test cases. The implementation is followed using 3:2 compressor.The combinational path delay is 11.314 ns for an 8 bit Wallace tree multiplier and area reduction is 29.03% compared to an array multiplier. By the synthesis on ARTIX 7 FPGA the performance of Wallace tree multiplier is improved. The results prove that the proposed architecture is more efficient than the existing one in terms of delay and power consumption. This approach may be well suited for multiplication of numbers with more than 16 bit size for high speed applications.The power of the proposed multiplier can be explored to implement high performance multiplier in VLSI applications. Wallace tree multiplier using compressors such as 4:2, 5:2 and 7:2 can be designed to further reduce the delay. The work can be extended by employing a method known as pipelining to improve the multiplier in terms of time delay and power consumption

15. ACKNOWLEDGMENT

First of all, we thank God Almighty for His abundant grace and mercy which enabled us in the finalization of this project.The support and help of a few people not only enabled us to complete our project successfully, but also made it a worthwhile experience. We thank the management for providing all the infrastructure and facilities.We are grateful toDr.M.C.Philipose, principal, SAINTGITS College of Engineering for providing us the best facilities and atmosphere for the development and implementation of our project. We thank Prof. Susan Abe, HOD, Electronics and Communication, for her encouragement and support; also we thank Er.Ashwin P V, Project Coordinator for his valuable suggestions and support.We are greatly indebted to our main project guide Er.Riboy Cherian, Asso. Prof, Dept. of Electronics and Communication for his invaluable suggestions and ideas without which this project would have been a dream.Finally, we thank all the staff members of our department for their timely help and guidance.

16. REFERENCES

[1] S. Karthick , S. Karthika , S. Valarmathy,Design and analysis of low powercompressors' , *International* Journal Of Advanced Research In Electrical, Electronics And Instrumentation Engineering, Vol. 1, Issue 6, December 2012.

[2] K. Gopi Krishna, B. Santhosh , V.Sridhar, Design of Wallace Tree Multiplier using Compressors ,International Journal Of Engineering Sciences & Research Technology

[3] Samir Palnitkar ,'*VerilogHDL:A guide to Digital Design and Synthesis',* 2nd edition , Prentice Hall , USA.

[4] Oscar Gustafsson , Andrew G. Dempster ,Lars Wanhammar . *Multiplier Blocks Using carry Save Adders*. Department of Electrical Engineering, Linkoping University, Sweden and University of Westminster, UK

[5] Moises E Robinson and Earl Swartzlander ,jr .. *A Reduction Scheme To Optimize The Wallace Multiplier*, Department of Electrical and Computer Engineering, University of Texas, USA.

[6] Michael D. Ciletti, *Advanced Digital Design with Verilog HDL*, Prentice Hall of India.

[7] Neil H. E. Weste, David Money Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, Pearson Education India

[8] V. Carl Hamacher, Zvonko G. Vranesic, Safwat G. Zaky,*ComputerOrganisation* , McGraw-Hill, 2002.

[9] V G KiranKumar ,H.RNagesh, *Introduction to VLSI Design* ,Pearson.Education

[10] xilingallprogrammable http://www.xilinx.com/training/fpga