International Journal of Advanced Technology and Innovative Research
www.ijatir.org
ISSN 2348–2370
Vol.07,Issue.01,
January-2015,
Pages:0126-0133

High Speed and Power Optimized Parallel Prefix Modulo Adders using Verilog

MEDAPATI NANDINI[1], A. JAYAVANI[2]
[1]PG Scholar, Kakinada Institute of Engineering & Technology, Korangi, AP, India, Email: m.nandusri@gmail.com.
[2]Asst Prof, Kakinada Institute of Engineering & Technology, Korangi, AP, India, Email: jaya.adabala@gmail.com.

Abstract: The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor data path units. As such, extensive research continues to be focused on improving the power delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. Binary adders are one of the most essential logic elements within a digital system. In addition, binary adders are also helpful in units other than Arithmetic Logic Units (ALU), such as multipliers, dividers and memory addressing. Therefore, binary addition is essential that any improvement in binary addition can result in a performance boost for any computing system and, hence, help improve the performance of the entire system.RNS schemes can be easily implemented using Parallel-prefix adders (also known as carry-tree adders) are known to have the best performance in VLSI designs. A standard RNS is defined exclusively for positive integers. To accommodate negative integers, an implicit signed number system may be considered to be split into a positive half of the range and a negative half of the range. At the same time it is inconvenient to design applications by using these normal architectures. That's why the proposed design is developed. This paper investigates three types of carry-tree adders (the Kogge-Stone, sparse Kogge-Stone, and spanning tree adder) and compares them with the Brent kung adder and Ladner fischer. In this project Xilinx-ISE tool is used for simulation, logical verification, and further synthesizing.

Keywords: Spanning Adder, Brent Kung Adder, Kogge Stone Adder, Ladner Fischer, Sparse Kogge.

I. INTRODUCTION

Arithmetic is the oldest and most elementary branch of Mathematics. The name Arithmetic comes from the Greek word ἀριθμός (arithmos). Arithmetic is used by almost everyone, for tasks ranging from simple day to day work like counting to advanced science and business calculations. As a result, the need for faster and efficient Adders in computers has been a topic of interest over decades. Addition is a fundamental operation for any digital system, digital signal processing or control system. A fast and accurate operation of a digital system is greatly influenced by the performance of the resident adders. Adders are also very important component in digital systems because of their extensive use in other basic digital operations such as subtraction, multiplication and division. Hence, improving performance of the digital adder would greatly advance the execution of binary operations inside a circuit compromised of such blocks. The performance of a digital circuit block is gauged by analyzing its power dissipation, layout area and its operating speed. To humans, decimal numbers are easy to comprehend and implement for performing arithmetic. However, in digital systems, such as a microprocessor, DSP (Digital Signal Processor) or ASIC (Application-Specific Integrated Circuit), binary numbers are more pragmatic for a given computation. This occurs because binary values are optimally efficient at representing many values.

Binary adders are one of the most essential logic elements within a digital system. In addition, binary adders are also helpful in units other than Arithmetic Logic Units (ALU), such as multipliers, dividers and memory addressing. Therefore, binary addition is essential that any improvement in binary addition can result in a performance boost for any computing system and, hence, help improve the performance of the entire system. The major problem for binary addition is the carry chain. As the width of the input operand increases, the length of the carry chain increases. Fig 1 demonstrates an example of an 8- bit binary add operation and how the carry chain is affected. This example shows that the worst case occurs when the carry travels the longest possible path, from the least significant bit (LSB) to the most significant bit (MSB). In order to improve the performance of carry-propagate adders, it is possible to accelerate the carry chain, but not eliminate it. Consequently, most digital designers often resort to building faster adders when optimizing computer architecture, because they tend to set the critical path for most computations.

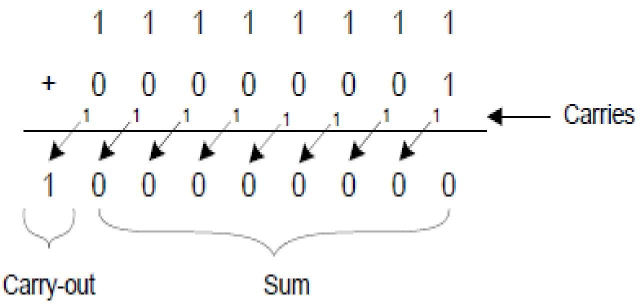


Fig1. Binary Adder Example.

II. REDUCE NUMBER SYSTEM

Reduce number system (RNS) is an ancient numerical representation system. It is recorded in one of Chinese arithmetical masterpieces, the Sun Tzu Suan Jing, in the 4th century and transferred to European known as Chinese Remainder Theorem (CRT) in the 12th century. RNS is a non-weighted numerical representation system and has carry-free property in multiplication and addition operations. In recent years, it has been received intensive study in the very large scale integration circuits (VLSI) design for digital signal processing (DSP) systems with high speed and low power consumption. Bayoumi proposed a scheme for arbitrary modulus by using two cascaded binary adders .However, the delay is the sum of the two binary adders. Several literatures constructed several modular adders with two parallel binary adders to calculate A+B and A+B+T. This method can achieve less delay but needs about twice area of binary adder. Dugdale proposed a method to construct a type of general modular adders with a reused binary adder. The shortage of this structure is that it will use two operation cycles to perform one modular addition. The area or delay of these modular adders mentioned above is twice or more than that of binary adder. In recent studies, a few modular adders with better area and delay performance are presented.

Hiasat proposed a class of modular adders in which any regular Carry Look-Ahead(CLA)—based binary adder can be used in the final stage. However, it needs an extra CLA unit to get the carry-out bit of A+B+T before the final CLA addition. As a result, the structure does not reduce the delay significantly. The ELMMA algorithm proposed by Platelet al. uses two carry computation modules for A+B and A+B+T in which some carry computation units can be shared. The area reduction of this scheme dominated by the form of T. In the worst case, almost two independent carry generation modules are needed. Patelet al. also proposed several algorithms which can generate carries fast. A new number representation for modulo addition is proposed. However, its outputs are represented in special format. Thus, the extra area and delay are needed to perform the conversion from the special representation to binary number representation or all operations should be performed in this number representation format in RNS-based systems. On the other hand, the complexity of the special modular adder is much less than that of general modular adder, since the structure of the special modular adder can be further optimized according to the modulus.

III. BINARY ADDER SCHEMES

Adders are one of the most essential components in digital building blocks, however, the performance of adders become more critical as the technology advances. The problem of addition involves algorithms in Boolean algebra and their respective circuit implementation. Algorithmically, there are linear-delay adders like ripple-carry adders (RCA), which are the most straightforward but slowest. Adders like carry-skip adders (CSKA), carry-select adders (CSEA) and carry-increment adders (CINA) are linear-based adders with optimized carry-chain and improve upon the linear chain within a ripple-carry adder. Carry-lookahead adders (CLA) have logarithmic delay and currently have evolved to parallel-prefix structures. Other schemes, like Ling adders, NAND/NOR adders and carry-save adders can help improve performance as well.

A Full Adder is a combinational circuit that performs the arithmetic sum of three input bits. It consists of three inputs and two outputs. Three of the input variables can be defined as A, B, Cin and the two output variables can be defined as S, Cout. The two input variables that we defined earlier A and B represents the two significant bits to be added. The third input Cin represents the carry bit. We have to use two digits because the arithmetic sum of the three binary digits needs two digits. The two outputs represents S for sum and Cout for carry. For designing a full adder circuit, two half adder circuits and an OR gate is required. It is the simplest way to design a full adder circuit. To design a full adder two XOR gates, two AND gates, one OR gate is required

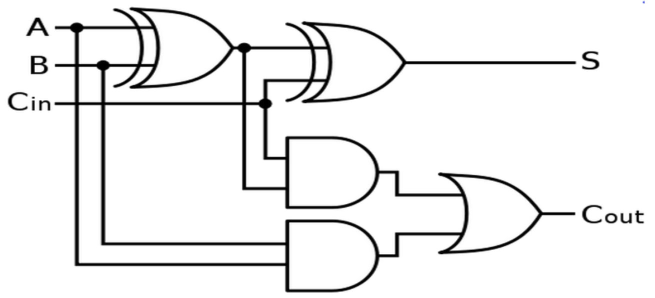


Fig 2. Full adder structure.

$$S = A \wedge B \wedge C \quad (1)$$

$$c_{out} = A.B + (A \wedge B).C_{in} \quad (2)$$

A. Parallel Prefix Adders

Parallel prefix adders employs 3- stage structure of carry look ahead adder The improvement is in the carry generation stage which is the most intensive one. The below figure3 shows the structure of ling adder. The ling adder uses predetermined propagate and generate in 1st stage of design. The 2nd stage uses the carry calculation paralleled to reduce time .the 3rd stage is the simple adder block to calculate the sum

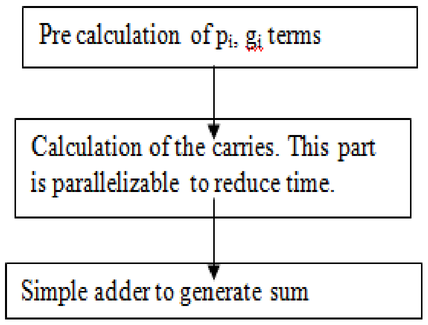

Pre calculation of p_i, g_i terms
Calculation of the carries. This part is parallelizable to reduce time.
Simple adder to generate sum

Fig 3. Structure of the parallel prefix adder.

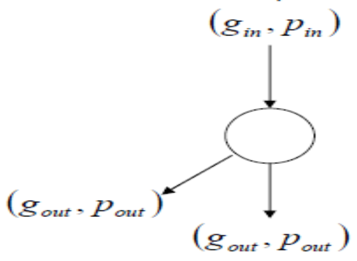
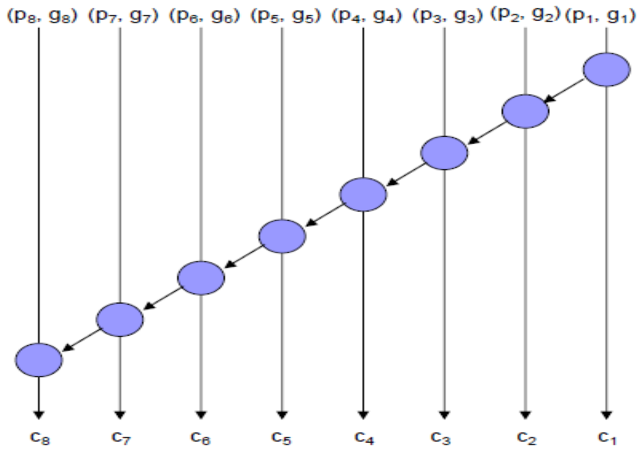


Fig 4. Processing component structure.

The parallel prefix graph for representation of prefix addition is shown as Fig 5.



(p_8, g_8) (p_7, g_7) (p_6, g_6) (p_5, g_5) (p_4, g_4) (p_3, g_3) (p_2, g_2) (p_1, g_1)
c_8 c_7 c_6 c_5 c_4 c_3 c_2 c_1

Fig 5. Parallel prefix adder structure.

Some of the parallel prefix adders are Ladner-Fisher adder, Brent-Kung adder, Kogge-Stone adder, sparse kogge stone adder, Spanning carry look ahead adder

B. Kogge-Stone Prefix Tree

Kogge-Stone prefix tree is among the type of prefix trees that use the fewest logic levels. A 16-bit example is shown in Figure 3.17. In fact, Kogge-Stone is a member of Knowles prefix tree. The 16-bit prefix tree can be viewed as Knowels [1,1,1,1]. The numbers in the brackets represent the maximum branch fan-out at each logic level. The maximum fan-out is 2 in all logic levels for all width Kogge-Stone prefix trees. The key of building a prefix tree is how to implement Equation according to the specific features of that type of prefix tree and apply the rules described in the previous section. Gray cells are inserted similar to black cells except that the gray cells final output carry outs instead of intermediate G/P group. The reason of starting with Kogge-Stone prefix tree is that it is the easiest to build in terms of using a program concept. The example in Figure 6 is 16-bit (a power of 2) prefix tree. It is not difficult to extend the structure to any width if the basics are strictly followed. For the Kogge-Stone prefix tree, at the logic level 1, the inputs span is 1 bit (e.g. group (4:3) take the inputs at bit 4 and bit 3). Group (4:3) will be taken as inputs and combined with group (6:5) to generate group (6:3) at logic level 2. Group (6:3) will be taken as inputs and combined with group (10:7) to generate group (10:3) at logic level 3, and so on so forth. The number cells for a Kogge-Stone prefix tree can be counted as follows. Each logic level has n-m cells, where m = $2^{l\ \text{level}\ -\ 1}$. That is, each logic level is missing m cells. That number is the sum of a geometric series starting from 1 to n/2 which totals to n-1. The total number of cells will be nlog $_2$n subtracting the total number of cells missing at each logic level , which winds up with nlog $_2$n-n +1. When n = 16, the area is estimated as 49.

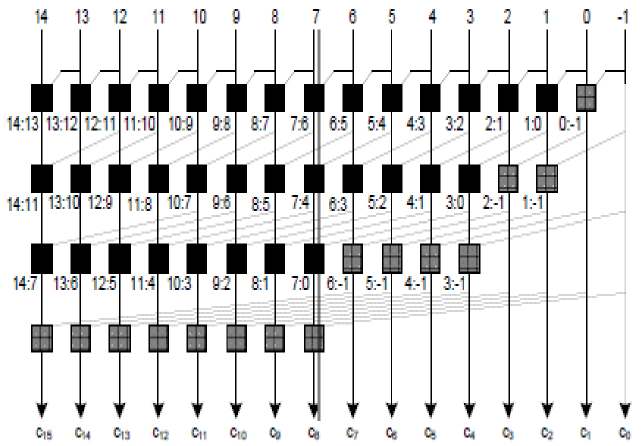


Fig 6. Block diagram of Kogge-Stone Prefix Tree.

C. Sparse kogge stone adder

The sparse Kogge-Stone adder consists of several smaller ripple carry adders (RCAs) on its lower half, a carry tree on its upper half. It terminates with RCAs. The number of carries generated is less in a sparse KoggeStone adder compared to the regular Kogge-Stone adder. The functionality of the GP block, black cell and the gray cell remains exactly the same as in the regular Kogge-Stone adder. The sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders.

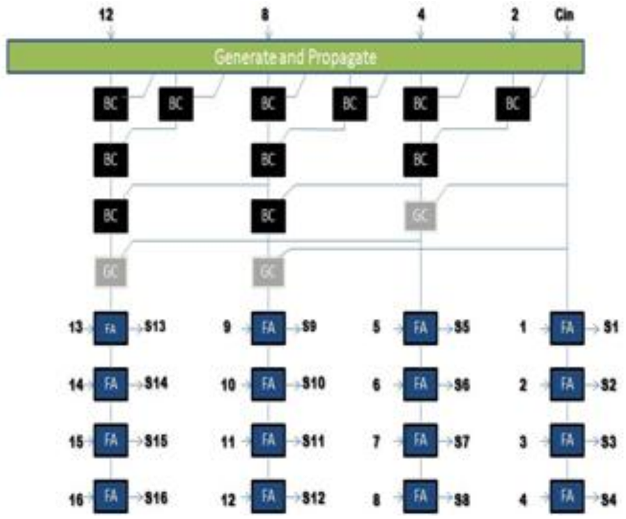


Fig 7. Block diagram of Sparse Kogge-Stone Adder.

D. Spanning carry look ahead adder

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder is like the sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders.

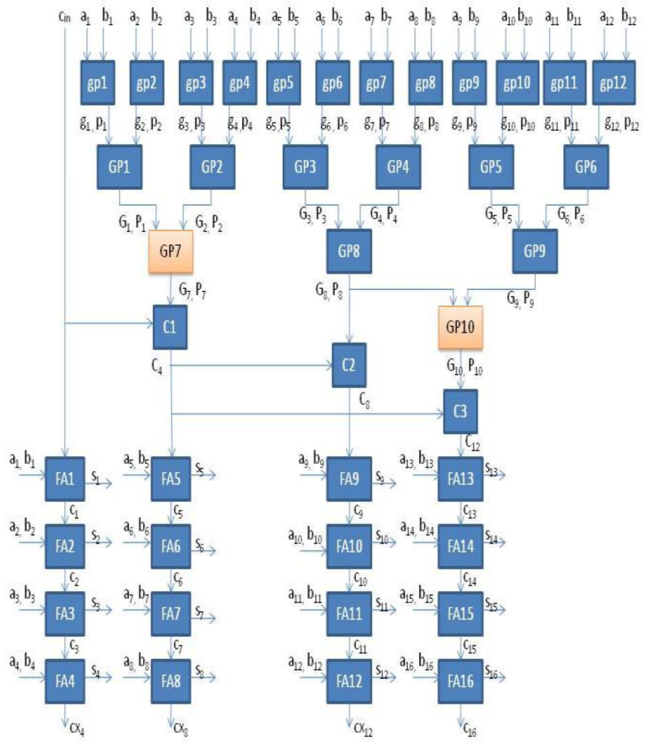


Fig 8. Block diagram of spanning tree carry look ahead adder.

E. Brent Kung Adder

The Brent Kung adder is the advanced version of Kogge stone adder. Kogge Stone adder generates intermediate carries shown in the block diagram is very attractive for high-performance applications. However, it comes at the cost of area and power. A simpler tree structure could be formed if only the carry at every power of two positions is computed as proposed by Brent and Kung [7]. Figure.9 shows a 16-bit prefix tree of their idea. An inverse carry tree is added to compute intermediate carries. Its wire complexity is much less than that of Kogge Stone adder. The number of slices, LUT, IOB Bounds is less compared to Kogge Stone Adder, Sparse Kogge stone Adder and Spanning Tree Carry Look Adder. When the usage of components reduces, the circuit complexity reduces, cost also gets reduces. so according to application either of the adder are used. The large number of levels in Brent Kung Adder (BKA) however reduces its operational speed. BKA is also power efficient because of its lowest area delay with large number of input bits . The delay of BKA is equal to (2*log2n)-2 which is also the number of stages for the “o” operator. The BKA has the area (number of “o” operators) of (2*n)-2-log2n where n is the number of input bits [1]. The BKA is known for its high logic depth with minimum area characteristics .High logic depth here means high fan-out characteristics.

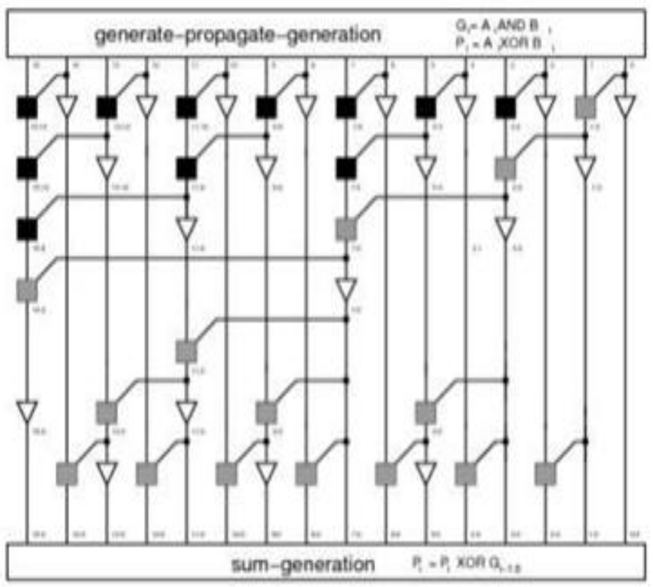


Fig 9. Block Diagram of Brent Kung Adder.

F. Lander-Fischer Adder

Addition operation is the main operation in digital signal processing and control systems. The fast and accuracy of a processor or system depends on the adder performance. Ripple carry adder is used for the addition operation i.e., if N-bits addition operation is performed by the N-bit full adder. In ripple carry adder each bit full adder operation consists of sum and carry, that carry will be given to next bit full adder operation, that process is continuous till the Nth bit operation. The N-1th bit full adder operation carry will be given to the Nth bit full adder operation present in the ripple carry adder. For 16-bit ripple carry adder, the first bit carry is given to second bit full adder, second bit carry is given to the third bit full adder, similarly the operation is continue till fifteenth bit carry is given to sixteenth bit full adder. The addition operation is performed from least significant bit to most significant bit in ripple carry adder. In ripple carry adders each bit wait for the last bit operation. In parallel prefix adders instead of waiting for the carry propagation of the first addition, the idea here is to overlap the carry propagation of the first addition with the computation in the second addition, and so forth, since repetitive additions will be performed by a multi operand adder. The Ladner-Fischer is the parallel prefix adder used to perform the addition operation. It is looking like tree structure to perform the arithmetic operation. Ladner-Fischer adder is used for high performance addition operation. The Ladner-Fischer adder consists of black cells and gray cells. Each black cell consists of two AND gates and one OR gate. Ladner-Fischer adder consists of three stages. They are pre-processing stage, carry generation stage, post-processing stage.

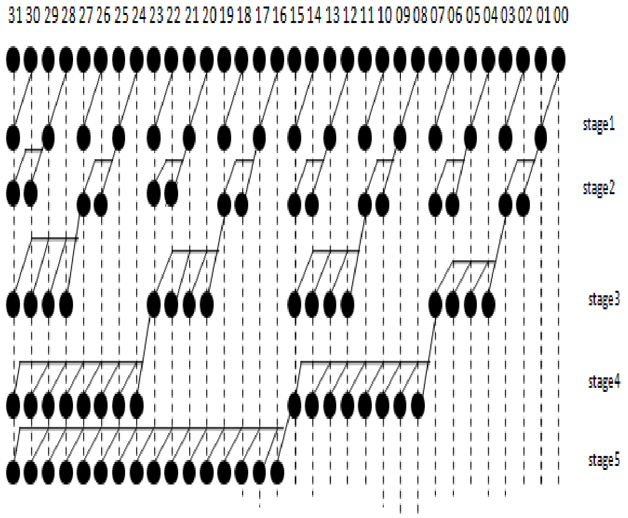


Fig 10. Block Diagram of LANDER FISCHER adder.

IV. IMPLEMENTATION

A. Kogge-Stone adder

Kogge-Stone prefix tree is among the type of prefix trees that use the fewest logic levels. In fact, Kogge-Stone is a member of Knowles prefix tree.

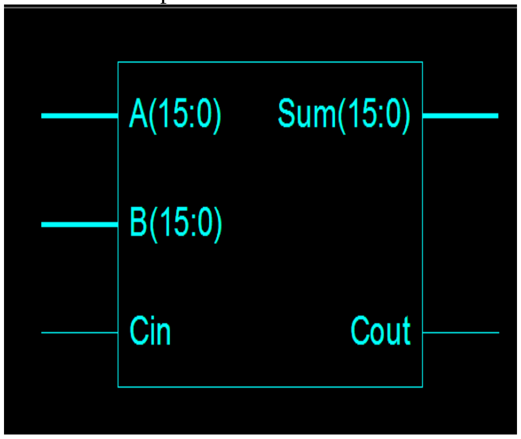


Fig 11. Block view of 16 bit Kogge stone Adder.

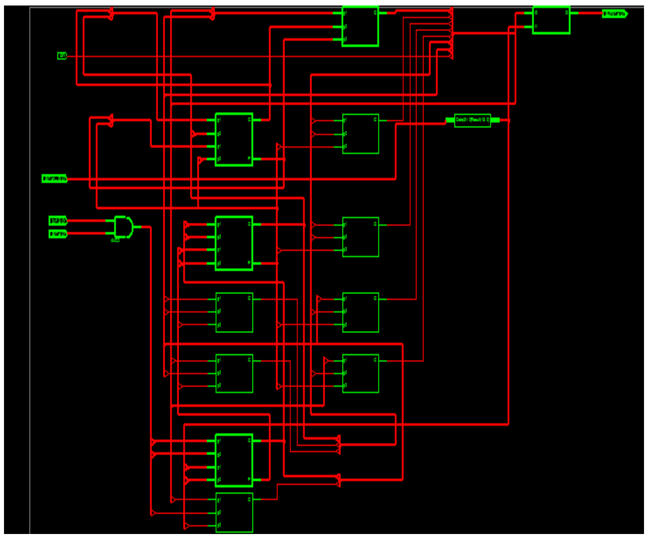


Fig 12. RTL View of 16 bit kogge stone adder

B. Sparse Kogge-Stone adder

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder is like the sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders.

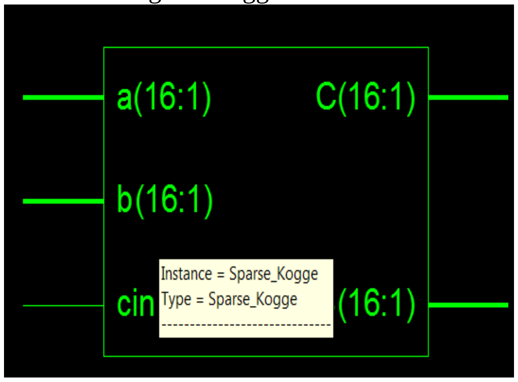


Fig 13. Block view of 16-bit sparse kogge-stone Adder.

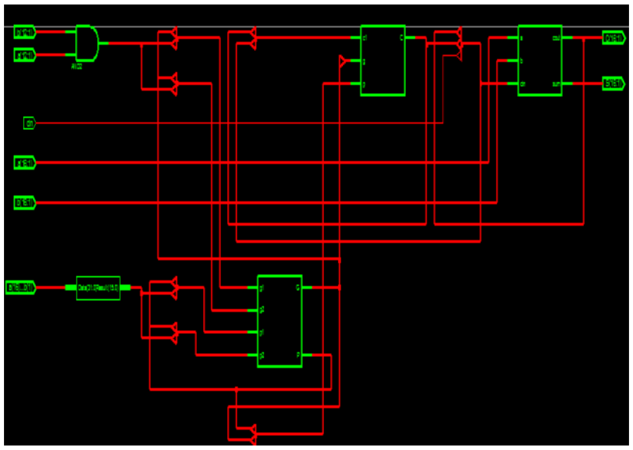


Fig 14. RTL View of 16-bit Sparse-kogge stone Adder.

C. Spanning Tree Carry Look ahead Adder

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder is like the sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders.

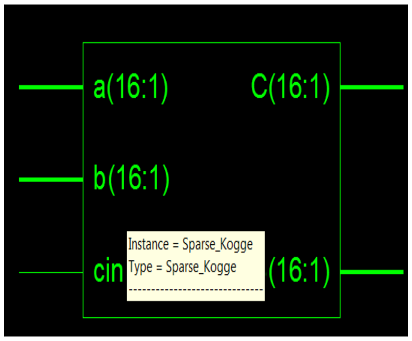


Fig 15. Block view of 16-bit Spanning Tree Carry Look ahead Adder.

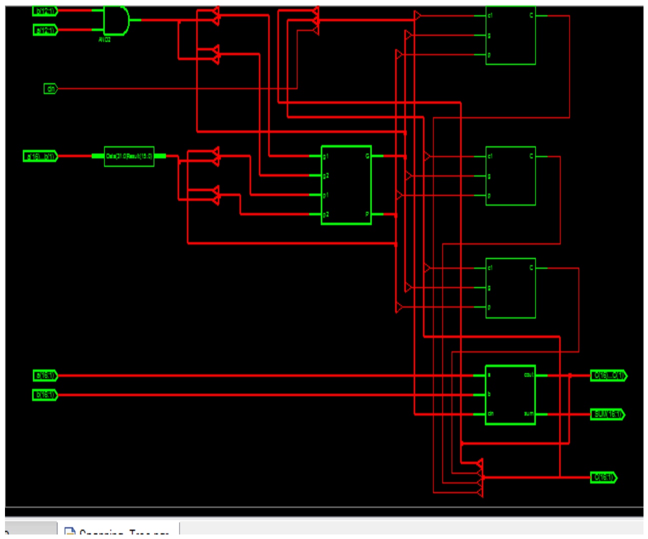


Fig16. RTL View of 16-bit Spanning Tree Carry Lookahead Adder.

D. Lander-Fischer Adder

The Ladner-Fischer adder consists of black cells and gray cells.

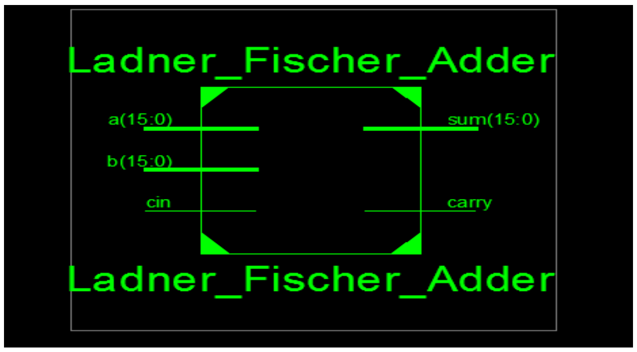


Fig 17. Block view of 16-bit LANDER-FISCHER Adder.

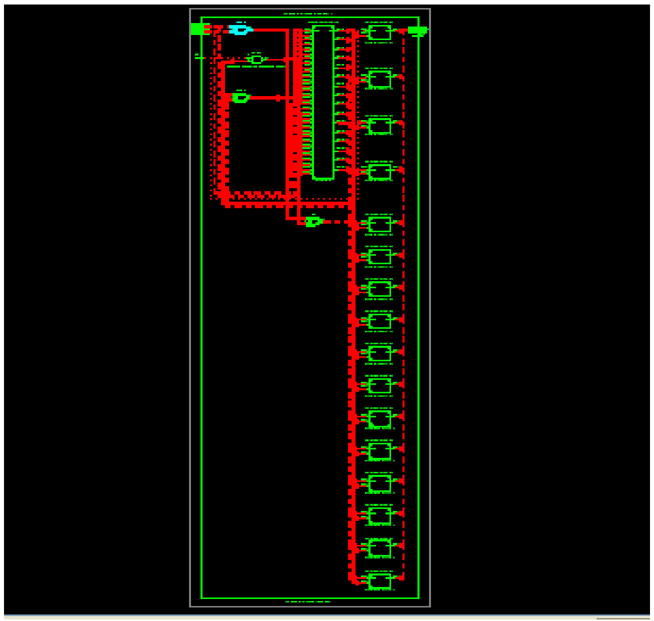


Fig 18. RTL View of 16-bit LANDER-FISCHER Adder.

Each black cell consists of two AND gates and one OR gate. Ladner-Fischer adder consists of three stages. They are pre-processing stage, carry generation stage, post-processing stage.

E. Brent-Kung Adder

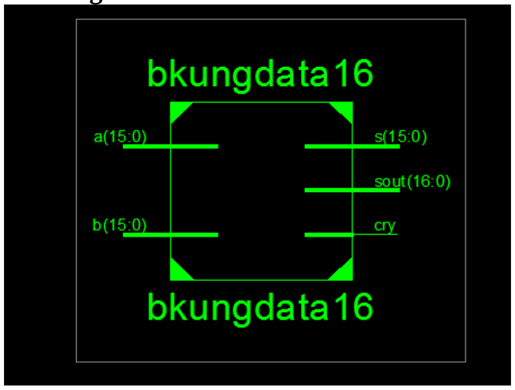


Fig 19. Block view of Brent-Kung adder 16-bit Adder.

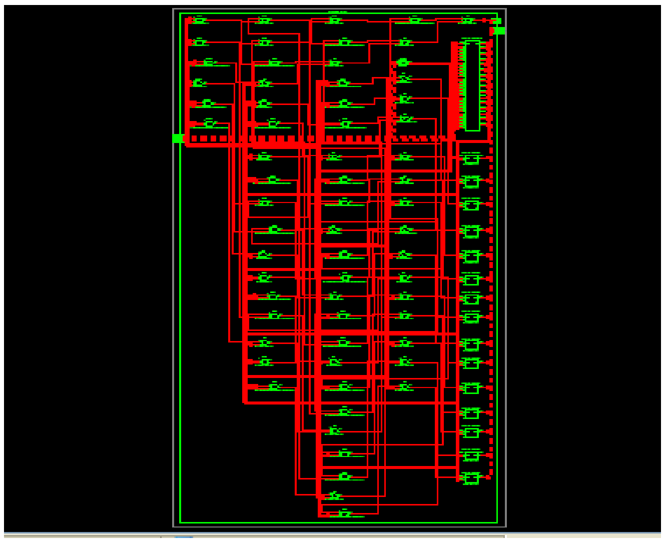


Fig 20. RTL View of 16-bit Brent-Kung Adder.

The Brent Kung adder is the advanced version of Kogge stone adder. Kogge Stone adder generates intermediate carries shown in the block diagram is very attractive for high-performance applications. However, it comes at the cost of area and power. A simpler tree structure could be formed if only the carry at every power of two positions is computed as proposed by Brent and Kung [7].

V. RESULTS

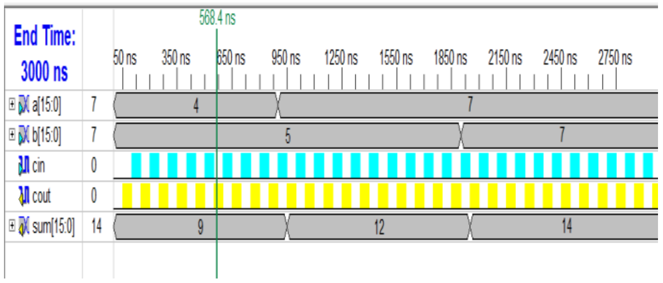


Fig 21. simulated wave form of kogge stone adder.

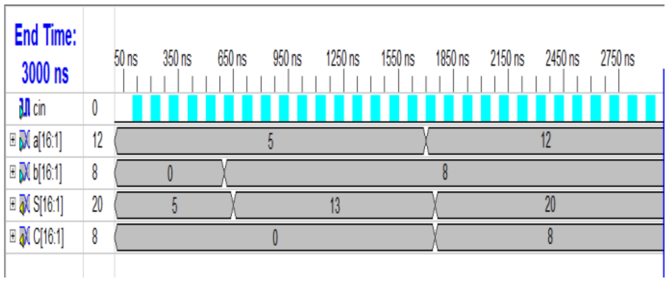


Fig22. Simulated wave form of sparse-kogge stone adder

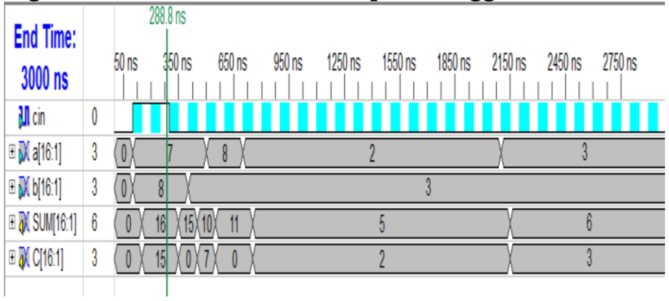


Fig 23. Simulated waveform of spanning tree carry alook head adder.

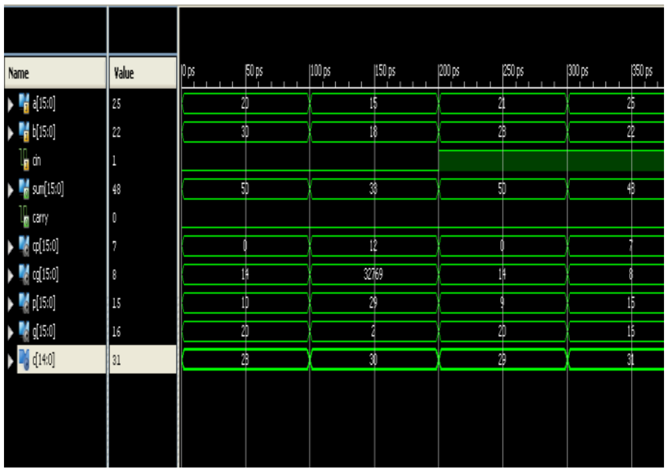


Fig 24. Simulated waveform of LANDER-FISCHER adder.

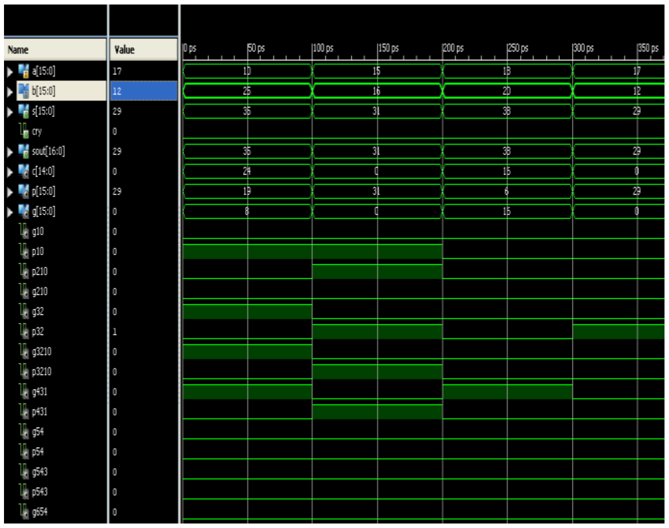


Fig 25. Simulated waveform of Brent-Kung adder.

A. Performance Comparative Analysis

The below comparison table is drawn from the analysis of five adders in terms of delay, power memory and number of 4 input LUTS which are used to compare the performance from synthesis report.

TABLE 1: COMPARATIVE ANALYSIS

ADDER	4 INPUT LUTS USED	DELAY (ns)	MEMORY (kb)	POWER(mw)
Spanning adder	39	19.115	1,33,636	0.31796
Brent kung adder	37	17.484	1,40,804	0.30166
Kogge stone adder	94	12.499	1,35,684	0.76637
Ladner fischer	32	21.690	1,33,636	0.26089
Sparse kogge	57	15.502	1,48,932	0.46472

VI. CONCLUSION

The Adders namely Brent kung adder, Ladner fischer adder, Spanning adder, Kogge stone adder and sparse Kogge stone adder are discussed in detail. Each individual module was tested for its correct functionality. This project has resulted in the development of Adders Design with reduced delay and power advantage. The Delay measurement and power analysis of the adders is being done and these designed kogge stone adder and sparse kogge stone adders power, delay and area is being compared with the other adder. From synthesis report the all the adders are compared in terms of delay, power memory and number of 4 input LUTS. From the comparison table Kogge stone adder has less combinational delay with 12.499 ns, Ladner fischer adder has less amount of power consumption with 0.26089 mw and uses less amount of 4 input LUTS.In future all the proposed architectures are designed using parallel prefix adders are used in the design of MAC unit. CMOS Implementation of adders in cadence is also being done for better performance. The architectures proposed are of combinational circuits, there is a tendency of causing struck at faults so the self checking circuits are to be designed in the future work to detect and correct the errors obtains in the adders.

VII. REFERENCES

[1] B. Ramkumar and Harish M Kittur , “ Low-Power and Area-Efficient Carry Select Adder “, IEEE Transactions on very large scale integration (VLSI) systems, vol.20, no.2,pp.371-375, Feb .2012.

[2] Dilip P. vasudevan, parag k.lala, james patrik parkerson, “ Self- checking carry-select adder design based on two rail en-

coding",IEEE Trans, CIRCUITS AND SYSTEMS—I , December 2007.
[3] http://en.wikibooks.org/wiki/Digital_Circuits/Adders.html
[4] Yu-Ting Pai and Yu-Kumg Chen, "The fastest carry look ahead adder," IEEE trans . International Workshop on Electronic Design, Test and Applications,vol.22. Feb .2004
[5] http://umunhum.stanford.edu/~farland/notes.html.
[6] Dayu Wang, Xiaoping Cui, Xiaojing Wang, "Optimized design of Parallel Prefix Ling Adder,"IEEE Trans. pp.941-944, oct.2011.
[7] B.Bhaskar, M.Kanagasabapathy,V.S.Kanchana Bhaaskaran, "A hybrid adiabatic parllel prefix addition scheme for low power", IEEE Trans , International Conference on Recent Trends in Information Technology, pp.389-393 , june 2011.
[8] G. W. Hanson, Fundamentals of Nanoelectronics. Englewood Cliffs,NJ: Prentice-Hall, 2008.
[9] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "Performance comparisonof quantum-dot cellular automata adders," in Proc. IEEE Int. Symp.Circuits Syst., 2005, pp. 2522–2526.
[10] Architecture of FPGAs and CPLDs: A Tutorial – Computer...www.eecg.toronto.edu/~jayar/pubs/brown/survey.pdfFile Format: PDF/Adobe Acrobat - Quick View.
[11]FPGADesignFlowOverviewwww.xilinx.com/itp/xilinx10/.../ise_c_fpga_design_flow_overview.h...FPGA Design Flow Overview.
[12]The Full-Adderhyperphysics.phy-astr.gsu.edu/hbase/electronic/fulladd.html.
[13] N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson–Addison-Wesley, 2011.