

Evaluation of High Speed and Low Memory Parallel Prefix Adders

* A.Madhu Babu¹, K.Harikrishna²

¹ PG Student (M. Tech), Dept. of ECE, Chirala Engineering College, Chirala., A.P, India.

² Associate Professor, Dept. of ECE, Chirala Engineering College, Chirala., A.P, India.

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. Parallel-prefix adders (also known as carry-tree adders) are known to have the best performance in VLSI designs. This paper investigates three types of carry-tree adders (the Kogge-Stone, sparse Kogge-Stone, and spanning tree adder) and compares them to the simple Ripple Carry Adder (RCA). In this project Xilinx-ISE tool is used for simulation, logical verification, and further synthesizing.

Keywords: DSP, Adders, Delay, Power.

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.

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. Figure 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.

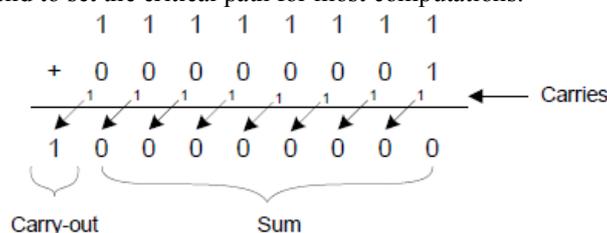


Figure 1 Binary Adder Example.

II. REVIEW OF ADDERS

FULL ADDER

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

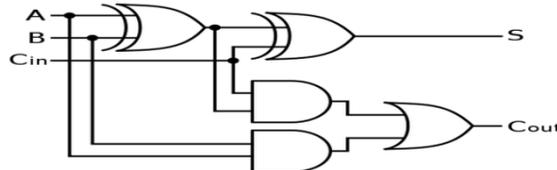


Figure 2 Full adder structure

$$S = A \oplus B \oplus C$$

$$C_{out} = A.B + (A \oplus B).C_{in}$$

PROPAGATION DELAY IN FULL ADDERS

Real logic gates do not react instantaneously to the inputs, and therefore digital circuits have a maximum speed. Usually, the delay through a digital circuit is measured in gate-delays, as this allows the delay of a design to be calculated for different devices. AND and OR gates have a nominal delay of 1 gate-delay, and XOR gates have a delay of 2, because they are really made up of a combination of ANDs and ORs.

A full adder block has the following worst case propagation delays:

- From A_i or B_i to C_{i+1} : 4 gate-delays (XOR \rightarrow AND \rightarrow OR)
- From A_i or B_i to S_i : 4 gate-delays (XOR \rightarrow XOR)
- From C_i to C_{i+1} : 2 gate-delays (AND \rightarrow OR)
- From C_i to S_i : 2 gate-delays (XOR)

Because the carry-out of one stage is the next's input, the worst case propagation delay is then:

- 4 gate-delays from generating the first carry signal ($A_0/B_0 \rightarrow C_1$).
- 2 gate-delays per intermediate stage ($C_i \rightarrow C_{i+1}$).
- 2 gate-delays at the last stage to produce both the sum and carry-out outputs ($C_{n-1} \rightarrow C_n$ and S_{n-1}).

So for an n-bit adder, we have a total propagation delay, t_p of:

$$t_p = 4 + 2(n-2) + 2 = 2n + 2$$

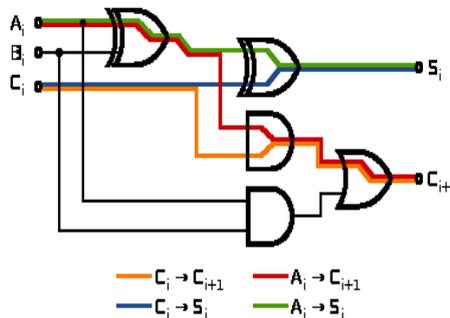


Figure 3 Delay in a full adder

This is linear in n, and for a 32-bit number, would take 66 cycles to complete the calculation. This is somewhat rather slow, and restricts the word length in our device. We would like to find ways to speed it up.

CARRY – LOOK AHEAD ADDER

A fast method of adding numbers is called carry-look ahead adder. This method doesn't require the carry signal to propagate stage by stage, causing a bottleneck. Instead it uses additional logic to expedite the propagation and generation of carry information, allowing fast addition at the expense of more hardware requirements. Rather than waiting for carry signals to ripple from the least significant to the most significant bit, CLA adders divided the inputs into groups of r bits and implement the logic equations to determine if each group will generate or propagate carry.

An overview of the adder's 4 stages is shown in the figure 2.5 with stage 1 and the top and stage 4 at the bottom. In stage 1 the local generate and propagate signals for each bit are created. In stage 2 these signals are combined into 9 bit block signals. In stage 4 the carry into each block is calculated and these signals begin traveling back up the adder tree. In stage 3 the carry into local carry signals to calculate the final sum bits. Each group is created, and into each bit is created. Finally, stage 1 uses the propagate and generate blocks to give the sum as the output. The carry look ahead shows good performance for the lower order bits that is it shows better performance for sum of less number of bits. The 32-bit carry look ahead adder shows better performance than 64-bit carry look ahead adder. Even the 64-bit adder shows better performance than 128 bit adder. As the bit size increases the performance decreases in the carry look ahead adder.

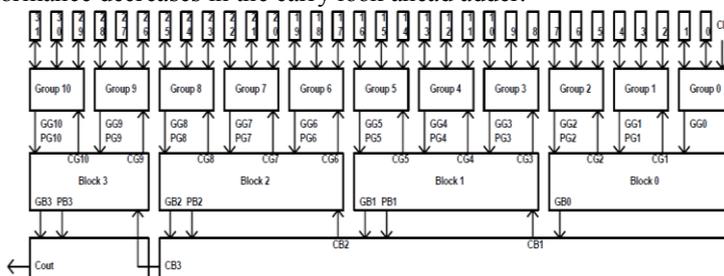


Figure 4 Simple Adder Tree Structure

III. Parallel Prefix Adders

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

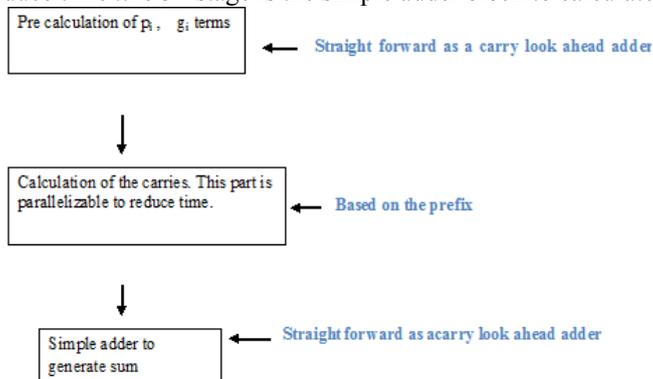


Figure 5 Structure of the parallel prefix adder

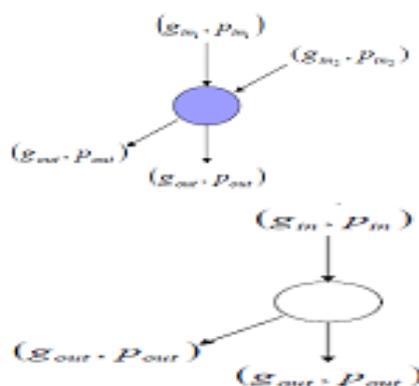


Figure 6 Buffer & Processing component structure

The parallel prefix graph for representation of prefix addition is shown as

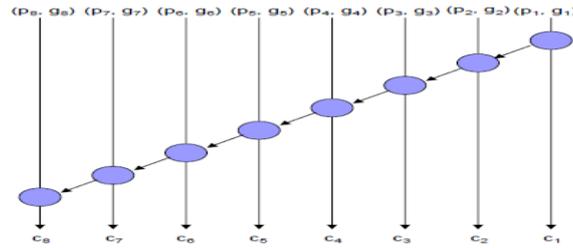


Figure 7 Parallel prefix adder structure

Some of the parallel prefix adders are :

- Kogge-Stone adder
- Sparse kogge stone adder
- Spanning carry lookahead adder

Kogge stone adder

The Kogge–Stone adder is a parallel prefix form carry look-ahead adder. It generates the carry signals in $O(\log n)$ time, and is widely considered the fastest adder design possible. It is the common design for high-performance adders in industry.

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 2.11. In fact, Kogge-Stone is a member of Knowles prefix tree. The 16-bit prefix tree can be viewed as Knowles [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 8 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.

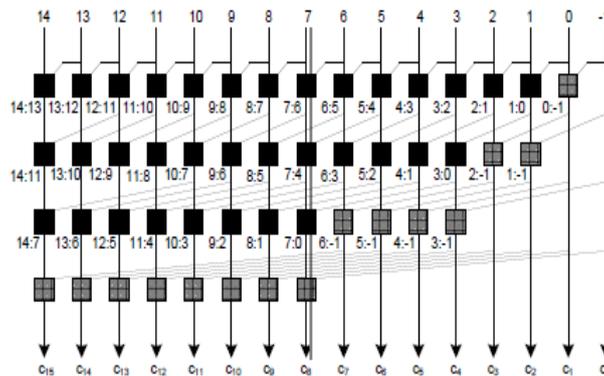


Figure 8 A 16-bit Kogge-Stone Prefix Tree

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.

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.

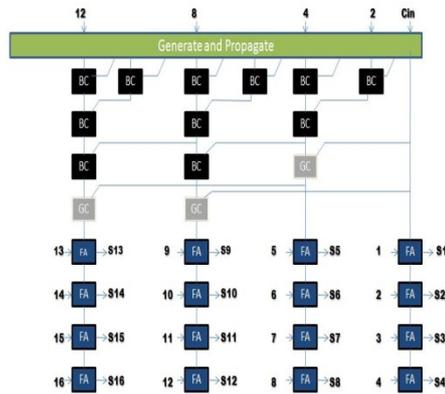


Figure 9 Block diagram of 16-Bit Sparse Kogge-Stone Adder[16].

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.

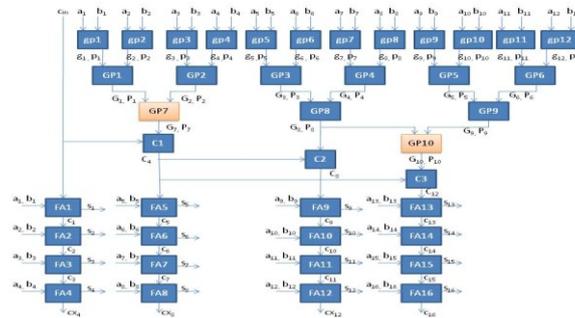


Figure 10 16 bit Spanning tree carry look ahead adder

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.

IV. Simulation Results

RIPPLE CARRY ADDER

The simplest way of doing binary addition is to connect the carry-out from the previous bit to the next bit's carry-in. Each bit takes carry-in as one of the inputs and outputs sum and carry-out bit and hence the name ripple-carry adder.

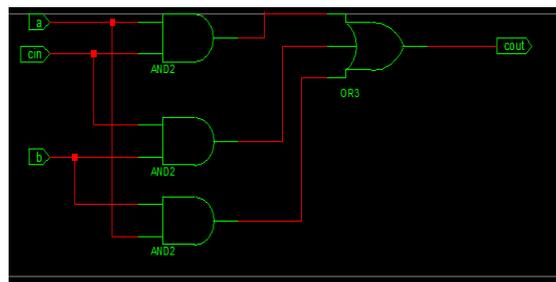


Figure 11 RTL view of 16 bit Ripple carry adder.

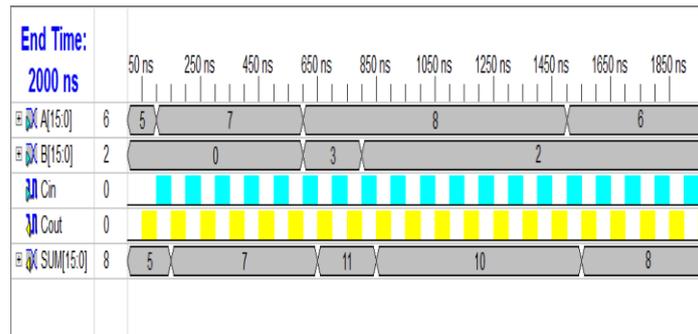


Figure 12 Simulated wave form of 16 bit ripple carry adder.

16 bit 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.



Figure 13 RTL View of 16 bit Kogge stone adder

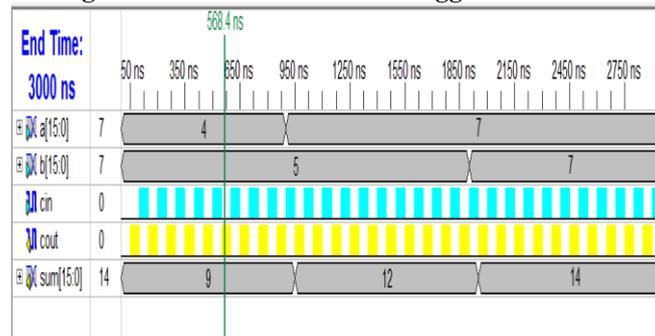


Figure 14 Simulated wave form of Kogge stone adder.

sparse 16 bit 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.

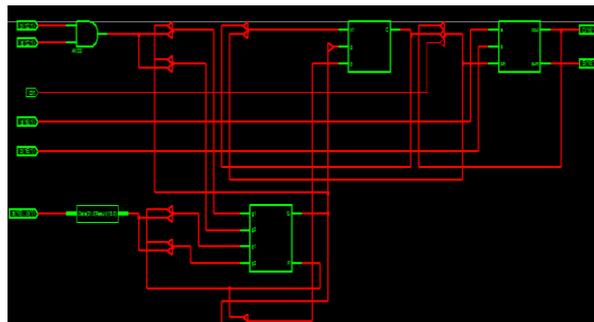


Figure 15 RTL View of 16-bit Sparse-Kogge stone Adder

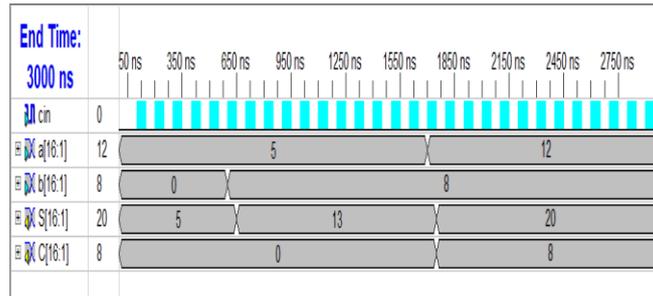


Figure 16 Simulated wave form of 16 bit sparse-Kogge stone adder

16-bit 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.

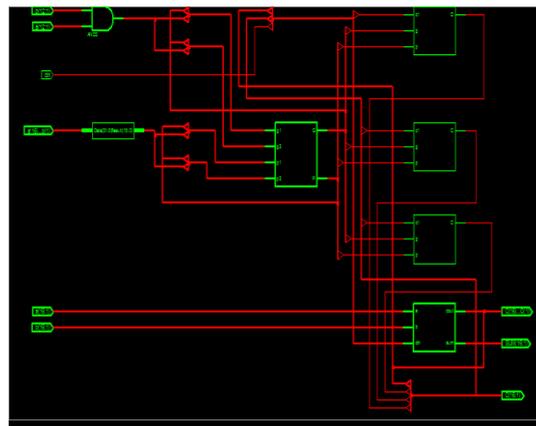


Figure 17 RTL View of 16-bit Spanning Tree Carry Look ahead Adder

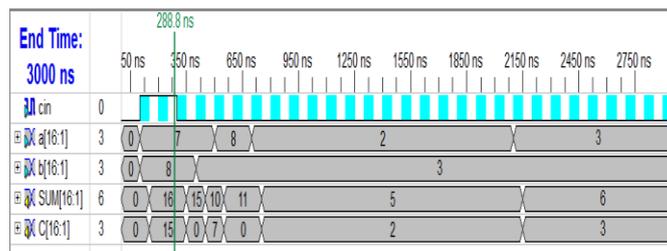


Figure 18 Simulated waveform of 16 bit spanning carry look head adder.

Performance Comparative Analysis

The below comparison table is drawn from the analysis of four adders in terms of delay, memory and LUT's drawn from synthesis report.

Table 1 Comparison briefs of simulated adders

ADDER NAME	Memory(KB)	Delay(ns)	LUT'S
KOGGE STONE	186972	25.084	36
SPARSE KOGGE STONE	187036	19.502	51
SPANNING TREE	186716	28.050	32
RIPPLE CARRY ADDER	189372	31.947	38

V. Conclusion:

The Adders namely ripple carry adder, Kogge stone adder, sparse Kogge stone adder and spanning tree adder are discussed in detail. VERILOG code was written for all the modules. Each individual modules was tested for its correct functionality. This project has resulted in the development of Adders Design with reduced delay and memory advantage. The Delay measurement and memory analysis of the adders is being done.

From synthesis report the adders designed kogge stone adder , sparse kogge stone adder and spanning tree adder is being compared with the normal ripple carry adder in terms of delay and memory. From the synthesis report the delay is less for SPARSE KOGGE STONE adder and the memory is less for SPANNING TREE is concluded. In future all the proposed architectures are designed using parallel prefix adders are used in the design of MAC unit.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

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-Kung 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 parrellel 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 comparison of 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.

Authors Profile:



A. Madhu Babu is Pursuing his M. Tech from Chirala Engineering College, Chirala in the department of Electronics & Communications En-gineering (ECE) with specialization in VLSI & Embedded systems.



K. Hari Krishna is working as an Associate Professor in the department of Electronics & Communication Engin-eering in Chirala Engineering College, Chirala. She completed masters from JNTUK. She has over 8 years of teaching and two years industrial experience.