International Journal of Electrical and Electronics Engineers
Vol. No.8 Issue 01, January-June 2016
www.arresearchpublication.com
IJEEE
ISSN 2321 - 2055

# DEVISE AND INFERENCE OF DELAY, POWER AND AREA FOR ANALOGOUS PREFIX ADDERS

**P.S.N Bhaskar[1], K.M.Manjunath[2]**

*[1,2]Department of ECE, Alwardas Group, Andhra University, (India)*

## ABSTRACT

*Analogous Prefix Adders have been established as the most efficient circuits for binary addition. The binary adder is the critical element in most digital circuit designs including digital signal processors and microprocessor data path units. The final carry is generated ahead to the generation of the sum which leads extensive research fixated on reduction in circuit complexity and power consumption of the adder. In VLSI implementation, parallel-prefix adders are known to have the superlative performance. This paper scrutinizes four types of carry-tree adders (the Kogge-Stone, sparse Kogge-Stone, spanning tree, Brent Kung Adder) and compare them to the simple Ripple Carry Adder and Carry Skip Adder. These designs of varied bit-widths are simulated using instigated on a Xilinx version Spartan 3E FPGA. These fast carry-chain carry-tree adders support the bit width up to 256. We blazon on the area requirements and reduction in circuit complexity for a multiplicity of classical parallel prefix adder structures.*

***Keywords: Analogous Prefix Adders, Carry Tree Adders, FPGA, Logic Analyzer, Delay, Power.***

## I. FOREWORD

In Digital Computer Design adder is an imperative component and it is used in manifold blocks of its architecture. In many Computers and in various classes of processor specialization, adders are not only used in Arithmetic Logic Units [6], but also used to calculate addresses and table indices. There exist manifold algorithms to carry on addition operation ranging from simple Ripple Carry Adders to complex CLA [1].

The basic operations involved in any Digital Signal Processing systems are Multiplication, Addition and Accumulation [2]. Addition is an indispensible operation in any Digital, DSP or control system. Therefore fast and accurate operation of digital system relies on the performance of adders . Hence improving the performance of adder is the main area of research in most digital circuits.

Binary addition is a fundamental operation in most digital circuits. There are multiplicities of adders, each has certain performance. Each type of adder is selected depending on where the adder is to be used. Adders are critically imperative elements in processor chips and they are used in floating-point arithmetic units, ALUs, memory addressing, program counter updating, Booth Multipliers, ALU Designing, multimedia and communication systems, Real-time signal processing like audio signal processing, video/image processing, or large capacity data processing etc [3]. The requirements of the adder are that it is primarily fast and secondarily efficient in terms of power consumption. In VLSI implementations, parallel-prefix adders are known to have the best performance [10][13]. In this paper, designing and implementing the tree-based adders on FPGAs are described.

Tree-based adder structures are implemented on FPGA and compared with the Ripple Carry Adder (RCA) and the Carry Skip Adder (CSA)[14]. Some conclusions and suggestions are made for improving FPGA designs to empower better tree-based adder performance. Analogous prefix (or tree prefix) adders provide a good theoretical basis to make a wide range of design trade-offs in terms of delay, area and power[5][15]. Parallel Prefix Adders (PPA) is designed by considering carry look adder as a base. Similar to a CLA they employ the 3-stage structure shown in Figure.1 CLA and a PPA differs in second stage. In second stage carry signal of binary addition is generated

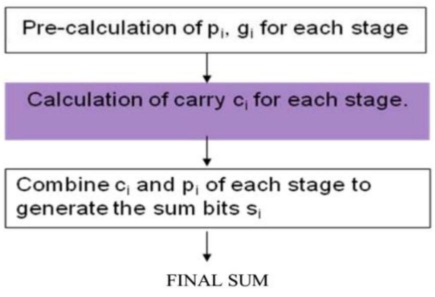

Pre-calculation of $p_i$, $g_i$ for each stage
Calculation of carry $c_i$ for each stage.
Combine $c_i$ and $p_i$ of each stage to generate the sum bits $s_i$
FINAL SUM

**Figure.1 Stages of Binary Addition**

Three stage structure of the carry look ahead and parallel prefix adder. In a PPA the prefix operator "o"[4] is introduced and the carry signal generation is treated as a prefix problem.

## II. THESIS

We compared the design of the ripple carry adder with the carry-look ahead, carry-skip, and carry-select adders on the Xilinx 4000 series FPGAs. Only an optimized form of the carry-skip adder performed better than the ripple carry adder when the adder operands were above 56 bits [11]. A study of adders implemented on the Xilinx Vertex II yielded similar results. The preceding authors considered several parallel prefix adders implemented on a Xilinx Vertex 5 FPGA. It is found that the unpretentious RCA adder is superior to the parallel prefix designs because the RCA can take advantage of the fast carry chain on the FPGA. This study focuses on carry-tree adders implemented on a Xilinx Spartan 3E FPGA. The distinctive contributions of this paper are two-fold. First, we consider tree-based adders and a hybrid form which combines a tree structure with a ripple-carry design. The Kogge-Stone adder is chosen as a representative of the former type and the sparse Kogge Stone and Brent Kung Adder is representative of the latter category. Second, this paper considers the practical issues involved in testing the adders and provides authentic measurement data to compare with simulation results. The previous works cited above all rely upon the synthesis reports from the FPGA place and route software for their consequences.

A 16-bit *Kogge-Stone* adder is built from 16 generate and propagate (GP) blocks, 37 black cells (BC) blocks, 16 (GC) blocks, 16 sum blocks. Kogge-Stone prefix tree is one of the adders that use fewest logic levels [16]. Gray cells are interleaved 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 tranquil to build in terms of using a program concept. The Figure.2 shown below is 16-bit (a power of 2) prefix tree and it is not problematic to extend the structure to any width if the basics are austerely followed. 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 Kogge Stone adder compared to the regular Kogge-Stone adder. The functionality of the GP block, black cell and the gray cell remains unerringly the same as in the regular Kogge-Stone adder. The sparse Kogge-Stone adder, this design cease with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is fascinating to compare the enactment of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders. The Figure.4Shown below is the Block diagram of 16-Bit Sparse Kogge-Stone Adder.

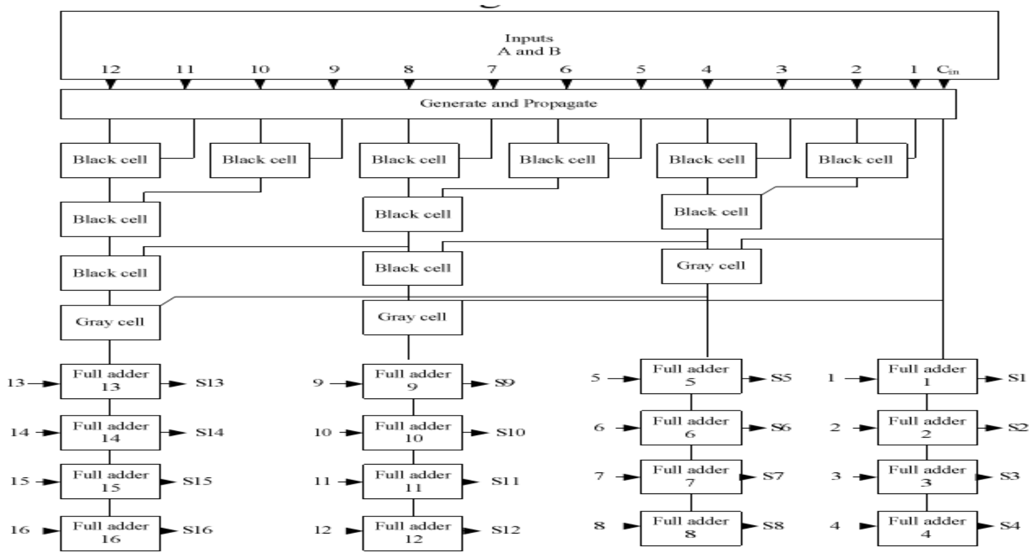


**Fig.2. 16 bit sparse kogge-Stone adder**

The 16 bit SKA uses black cells and gray cells as well as full adder blocks too. This adder computes the carries using the BC’s and GC’s and ceases with 4 bit RCA’s. Totally it uses 16 full adders. The 16 bit SKA is shown in figure 2. In this adder, first the input bits (a, b) are converted as propagate and generate (p, g). Then propagate and generate terms are given to BC’s and GC’s. The carries are propagated in advance using these cells. Later these are given to full adder blocks. Another PPA is known as STA is also tested [6]. Like the SKA, this adder also ceases with a RCA. It also uses the BC’s and GC’s and full adder blocks like SKA’s but the dissimilarity is the interconnection between them [7].The 16 bit STA is shown in the below figure 3.

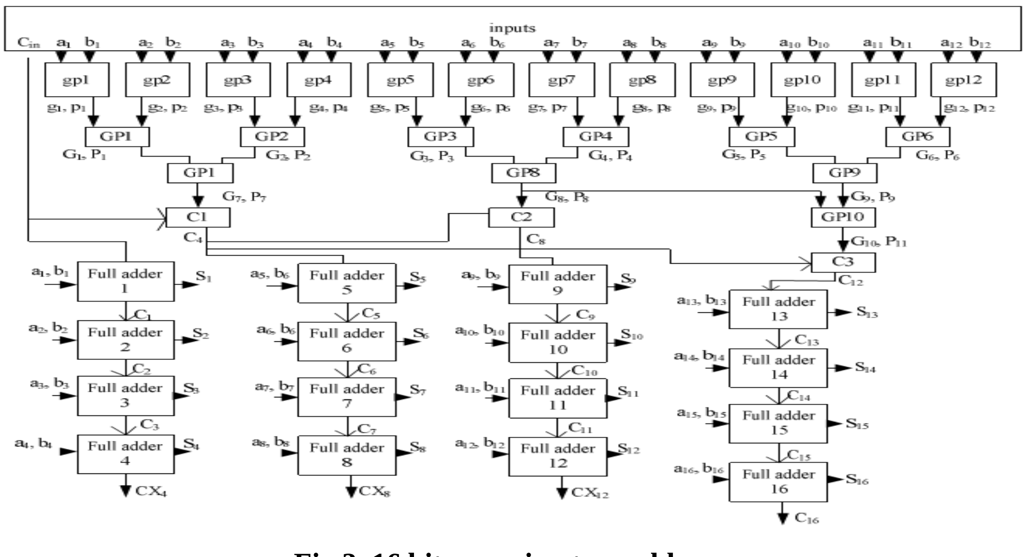


**Fig.3. 16 bit spanning tree adder**

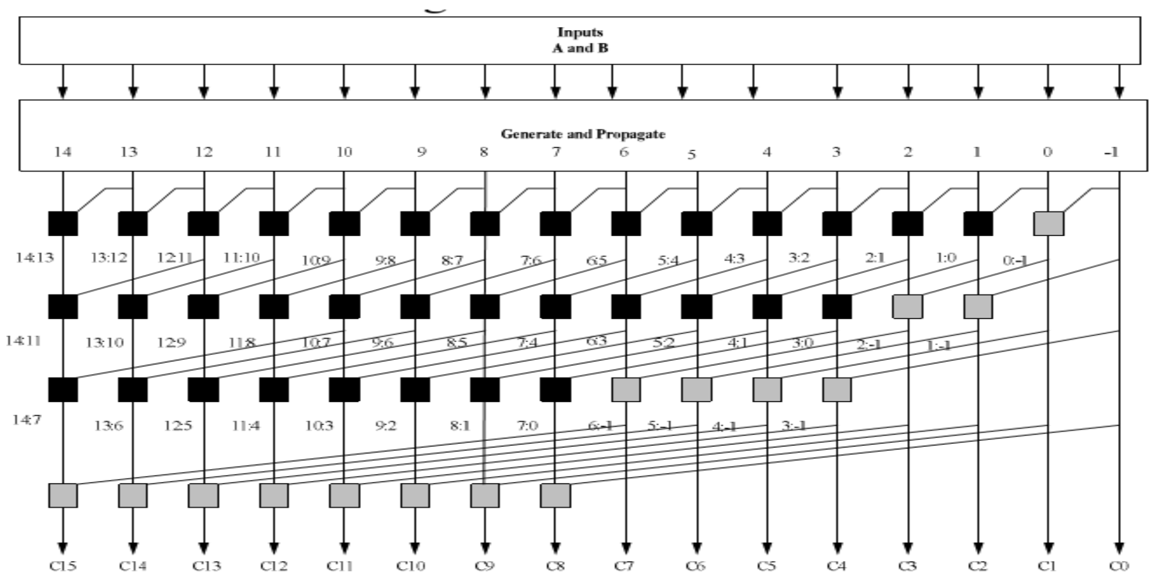


**Fig.4. 16 bit kogge stone adder**

Another carry tree known as BKA which also uses BC's and GC's but less than the KSA. So it takes less area to implement than KSA. The 16 bit BKA uses 14 BC's and 11 GC's but kogge stone uses 36 BC's and 15 GC's. So BKA has less architecture and inhabit less area than KSA. The 16 bit BKA is shown in the below figure 5.

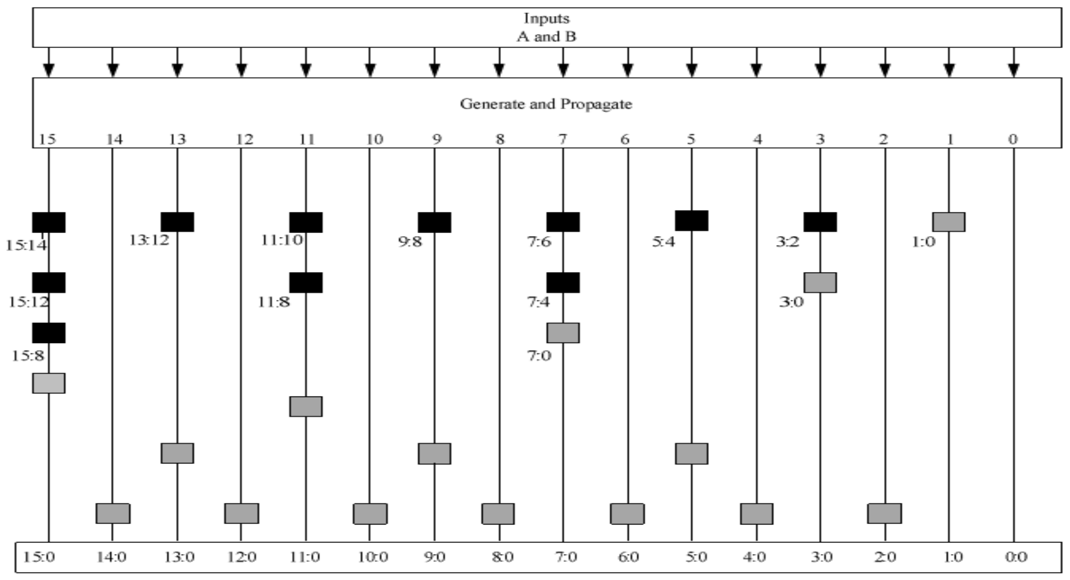


**Fig.5. 16 bit Brent Kung adder**

BKA inhabit less area than the other 3 adders called SKA, KSA, and STA. This adder uses limited number of propagate and generate cells than the other 3 adders. It takes less area to implement than the KSA and has less wiring congestion. The operation of the 16 bit brent kung adder is given below [3]. This adder uses less BC's and GC's than kogge stone adder and has the better delay performance which is observed in Agilent 1692A logic analyzer.

## III. CONSEQUENCES

Table.3 contains the results obtained. The adder abbreviations used in the table and the following deliberations are: BK for the Brent-Kung adder, KS for the Kogge-Stone adder, SK for Sparse Kogge Stone Adder, RC for Ripple Carry Adder. In the table, area is measured in Slice Look-Up Tables (LUT) units which characterize configurable logic units within the FPGA. Remarkably the synthesis tool synthesized a simple ripple carry adder regardless of the optimization strategy. The adder was implemented by configuring the slices within the FPGA as full adder components. Hence the number of lookup tables matched the operand bit-size in every case. Table.3 give the area results with the software set for area. It is apparent from these tables that the area optimization strategy produces adders which are ominously smaller compared to those produced with complexity optimization. In Table we can observe that generally the adder areas compare with the characteristics of their type. The BK exhibits the smallest size while the KS is the largest adder. This shows that in certain cases the tool optimized circuits reverse the algorithmic superiority of a design

**Table.3 Area Results Obtained With Area Optimization**

| | RC | KS | SK | BK |
|---|---|---|---|---|
| I/P Arrival Time | 18.43 | 12.72 | 12.65 | 10.82 |
| Path delay | 21.65 | 20.26 | 15.87 | 5.96 |
| Slices | 18 | 21 | 29 | 26 |
| 4 i/p LUT's | 32 | 37 | 51 | 45 |
| Bonded IOB's | 51 | 51 | 66 | 56 |

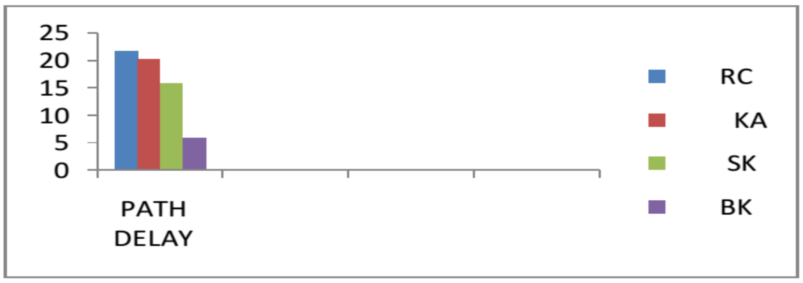
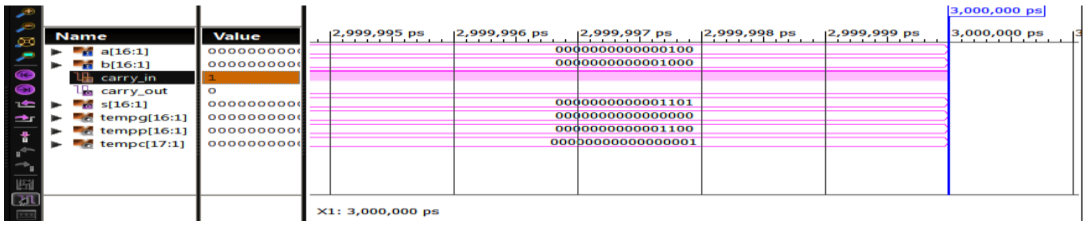


**Figure.6 Comparison of path delays for Adders**

These adders are implemented in verilog HDL in Xilinx 13.2 ISE design suite and then verified using Xilinx vertex 5 FPGA through chip scope analyzer [7], [8], [9]. And these were tested using Agilent 1692A logic analyzer.



| Name | Value | |
|---|---|---|
| a[16:1] | 000000000 | 0000000000000100 |
| b[16:1] | 000000000 | 0000000000001000 |
| carry_in | 1 | |
| carry_out | 0 | |
| s[16:1] | 000000000 | 0000000000001101 |
| tempg[16:1] | 000000000 | 0000000000000000 |
| tempp[16:1] | 000000000 | 0000000000001100 |
| tempc[17:1] | 000000000 | 00000000000000001 |

**Fig7: Look head adder result**

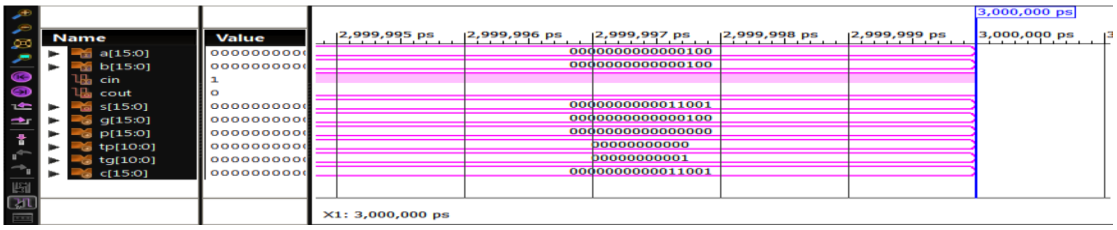


Fig8: Brentkung adder Result

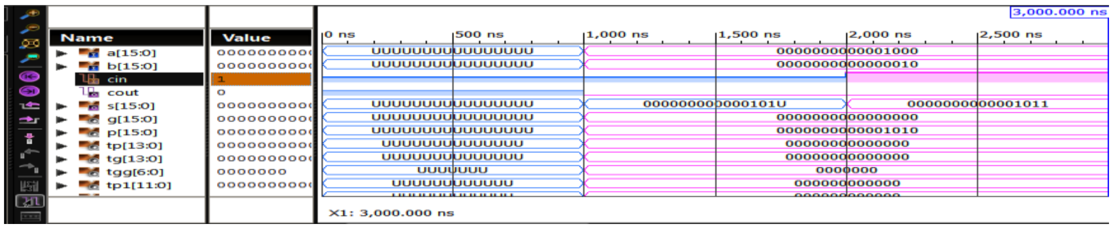


Fig9: koggestone Adder result

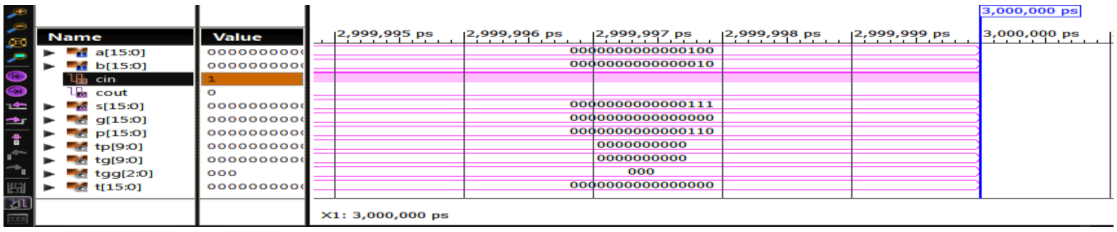


Figure10: spinning tree adder result

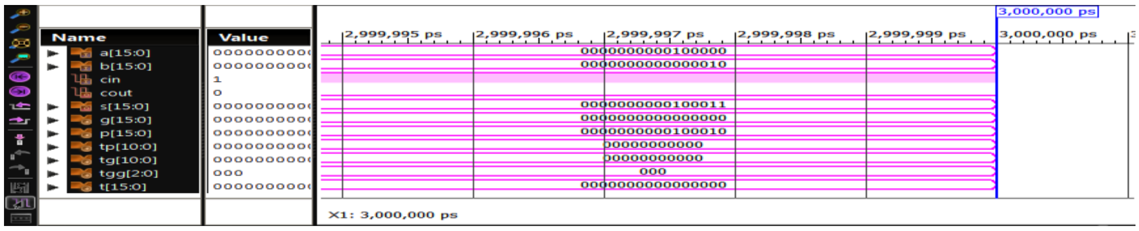


Figure11: sparse koggestone adder result

## IV. INFERENCE

Analogous-prefix adders are not as effective as the simple ripple-carry adder at low to moderate bit widths. We have indications that the carry-tree adders eventually surpass the performance of the linear adder designs at high bit-widths, expected to be in the 128 to 256 bit range. This is imperative for large adders used in precision arithmetic and cryptographic applications where the addition of numbers on the order of a thousand bits is not uncommon. Because the adder is often the critical element which determines to a large part the cycle time and power dissipation for many digital signal processing and cryptographically implementations, it would be worthwhile for future FPGA designs to include an optimized carry path to empower tree based adder designs to be optimized for place and routing. The testability and possible fault tolerant features of the Brent Kung adder are topics for future exploration

## REFERENCES

[1] A Journal by Y. Choi, "Parallel Prefix Adder Design", Proc. 17th IEEE Symposium on Computer

Arithmetic, p 90-98, 27th June 2005.

[2] A book on Vlsi Digital Signal Processing Systems: Design And Implementation page 201-250 by Keshab K. Parhi

[3] A book on “Real time business processing” by Prentice Hall page no: 781-892.

[4] A Journal by Kogge P, Stone H, “A parallel algorithm for the efficient solution of a general class Recurrence relations,” IEEE Trans. Computers, Vol.C-22, pp 786-793,Aug. 1973.

[5] A Journal by R. Zimmermann, “Non-heuristic operation and synthesis of parallel-prefix adders,” in International workshop on logic and architecture synthesis, December 1996,pp. 123-132.

[6] A Journal by C.Nagendra, M. J. Irwin, and R. M. Owens, “Area -Time-Power tradeoffs in parallel adders”, Trans. Circuits Syst. II, vol.43, pp.Applications 19, 285-298, 2003.

[6] A computation, Journal on ACM La.Jolla CA Volume 27 by R. Ladner and M. Fischer, “Parallel of ACM.La.Jolla CA,Vol.27,pp.831-838,October 1980.

[7] A Journal by Reto Zimmermann on Binary Adder Architectures for Cell-Based VLSIan their Synthesis. Hartung-Gorre, 1998.

[8] A Journal by Y. Choi, “Parallel Prefix Adder Design,” Proc. 17th IEEE Symposium onComputer Arithmetic,pp 90-98, 27th Jun2005.

[9] A conference report on, “A taxonomy ofparallel prefix networks,” in Signals, Systems and Computers,2003. Conference Record of Thirty Seventh Asilomar Conference D. Harrison, vol. 2, the Nov. 2003,pp.2217.

[10] N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson Addison-Wesley, 2011.

[11] H. Ling, High-speed binary adder," IBM Journal of Research and Development, vol. 25,no. 3, pp. 156 March 1981.

[12] A paper on K.Vitoroulis and A. J. Al-Khalili “Performance of Parallel Prefix Adder Implemented with FPGA technology,” IEENortheast Workshop on Circuits and Systems,pp.451-575, Aug. 2007

[13] A paper by D. H. K. Hoe, C. Martinez, and J. Vundavalli, “Design and Characterization ofanalogous Prefix Adders using FPGAs, ”IEE43rd Southeastern Symposium on System Theory, pp. 167-178 ,2011

[14] T. Matsunaga, S. Kimura, and Y. Matsunaga.“Power-conscious syntheses of parallel prefix adders under bitwise timingconstraints,” Proc. the Workshop on Synthesis And System Integration of Mixed Information technologies(SASIMI), Sapporo,Japan,October 2011,pp. 11–24.

[15] A book by F. E. Fich, “New bounds for parallel Prefixcircuits,” in Proc. of the 15thAnnu. ACM Symposia. Theory of Computer., 1983,pp.100–109.

[16] A journal by D. Gizopoulos, M. Psarakis, A. Paschalis,and Y.Zorian, “Easily TestableCellular Carry Look ahead Adders,” Journal ofElectronic Testing: Theory and Applications 19, 275-312.

## BIBLOGRAPHY

| | |
|---|---|
| | **P.S.N Bhaskar** received his M.Tech in VLSI from the Andhra University and is working in the Department of Electronics & Communication Engineering in AlwardasGroup in Visakhapatnam. He has a vast 14 years teaching experience He is an Life member of IRED,SDIWC, SCIEI,WAYS ISEIS,EA,IAENG,ISQEM,ISDS, and MISTE. His prime area of interest is on “Floor planning of algorithm Techniques” in VLSI design. |
| | **K.M.Manjunath**: is an assistant professor & currently working in Yogananda Institute of Technology and Science. He is perused his B.Tech from SITAMS in 2005 and received his M.Tech degree in Electronics and Communication Engineering in the year 2011. As he is in the field of teaching for a long time he has 8 years of teaching experience in VLSI, communication systems, electronic devices and circuits, Pulse and digital circuits. He subject of interest is on VLSI design. |