Design of High Speed and Low Power Adder by using Prefix Tree Structure

V.N.SREERAMULU

***Abstract*—In the technological world development in the field of nanometer technology leads to maximize the speed and minimize the power consumption of logic circuit. This can be achieved by Parallel Prefix Tree Structure. This project investigates a 64-bit hybrid adder by using both radix-4 prefix tree structure and carry select adder for low power and high speed applications. Inorder to optimize the features of this adder, some design issues are concerned including optimal layout for CMOS group generate/propagate circuit to reduce area, design of carry bypass adder (CBA) without conflict to boost speed, carry select adder (CSA) design with speed and area efficiency. Additionally four types of parallel prefix adders Kogge Stone Adder (KSA), Spanning Tree Adder (STA), Brent Kung Adder (BKA) and Sparse Kogge Stone adder (SKA). These adders are implemented in verilog hardware description language (HDL) using Xilinx Integrated Software Environment (ISE) 14.3 Design Suite. These designs are implemented in Xilinx Artix 7 Field Programmable Gate Arrays (FPGA).The experimental results reveal that the proposed 64-bit hybrid adder is superior to other referenced adders, and has 6.71ns delay time, 9.58 mw average power.**

***Index Terms*— Carry select adder, FPGA, hybrid adder, low power, parallel prefix adder.**

I. INTRODUCTION

The binary addition is the basic arithmetic operation in any digital circuit and it becomes essential in most of the digital systems including arithmetic and logic unit (ALU),microprocessors, digital signal processing (DSP) and floating point unit (FPU). With the rapid growth in portable electronic equipments and mobile communication devices, demand to low voltage and low power technology for VLSI applications is great increasing. In general, high speed adder includes carry look ahead adder (CLA), carry select adder (CSA), carry bypass adder (CBA), conditional sum adder and later developed parallel prefix adder (PPA). Different prefix algorithms and tree topologies (BKA,STA,KSA,SKA) of PPAs have been implemented for solving delay, area, and power efficiencies. Design of an appropriate tree topology is a trade-off among the fan-out, the wiring count and the logic level [1]. Recently, several high-performance 64-bit adders have been reported in [2-3]. The high-speed 64-bit adder [2] is hybrid sparse radix-4 prefix tree and CSA based on energy-delay optimization methodology.

In this thesis, a 64-bit hybrid adder is proposed to combine both prefix tree structure (PTS) and CSA for fetching low voltage and low power features. The three stages prefix tree of the 64-bit hybrid adder computes carries, and then the CSA with add-one circuit selects sums by these carries. Otherwise, the CBA has been added at the third stage of the PTS to diminish fan-ins, fan-outs, wiring counts and transistor counts. With respect to low voltage low power method, complementary metal oxide semiconductor (CMOS) logic, transmission gate (TG) logic, and pass transistor logic (PTL) are applied in proposed design to fetch full swing operation at each node.

II. EXISTED SYSTEM

In fig.1, the first sum bit should wait until input carry is given, the second sum bit should wait until previous carry is propagated and so on. Finally the output sum should wait until all previous carries are generated. So it results in delay.

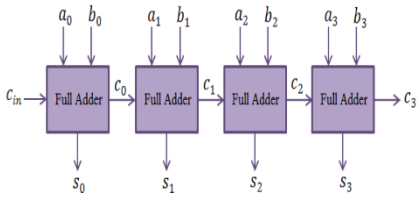

Fig.1: Four bit ripple carry adder

In order to reduce the delay in RCA (or) to propagate the carry in advance, we go for carry look ahead adder .Basically this adder works on two operations called propagate and generate The propagate and generate equations are given by.

Pi=Aixor Bi (1)

G i= A iand B i (2)

ISSN: 2278 – 7798

For 4 bit CLA, the propagated carry equations are given as

$C1 = G0 + P0\,C0$ (3)

$C2 = G1 + P1\,G0 + P1\,P0\,C0$ (4)

$C3 = G2 + P2\,G1 + P2\,P1\,G0 + P2\,P1\,P0\,C0$ (5)

$C4 = G3 + P3\,G2 + P3P2\,G1 + P3P2P1G0 + P3\,P2\,P1\,P0\,C0$ (6)

Equations (3),(4),(5) and (6) are observed that, the carry complexity increases by increasing the adder bit width. So designing higher bit CLA becomes complexity. In this way, for the higher bit of CLA's, the carry complexity increases by increasing the width of the adder. So results in bounded fan-in rather than unbounded fan-in, when designing wide width adders. In order to compute the carries in advance without delay and complexity, there is a concept called Parallel prefix approach.

The PPA's pre-computes generate and propagate signals are presented in [4]. Using the fundamental carry operator, these computed signals are combined in [5].Thefundamental carry operator is denoted by the symbol "o",

$(gL, pL)\ o\ (gR, pR) = (gL + pL\,.g\,R,\ p\,L\,.\,p\,R)$ (7)

For example, 4 bit CLA carry equation is given by,

$C4 = (g4, p4)\ o\ [(g3, p3)\ o\ [(g4, p4)\ o\ (g3, p3)]]$ (8)

For example, 4 bit PPA carry equation is given by,

$C4 = [(g4, p4)\ o\ (g3, p3)]\ o\ [(g4, p4)\ o\ (g3, p3)$ (9)

From equations (8) and (9) it is clear that, the CLA takes three steps to generate the carry, but PPA takes two steps to generate the carry. PPA's basically consists of three stages during addition process. They are

1. Pre computation
2. Prefix stage
3. Final computation

The structure of PPA is shown in fig.2

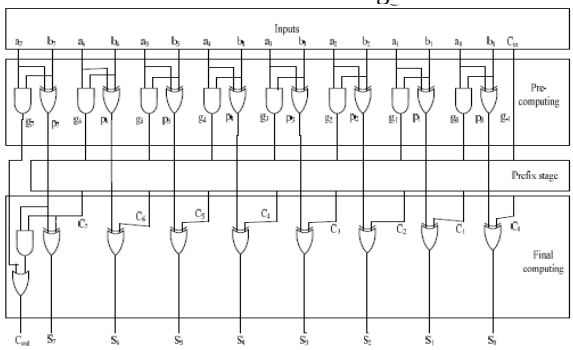

Fig.2: PPA structure with carry save notation

A. Pre computation:

In pre computation stage, propagates and generates are computed for the given inputs using the given equations (1) and (2).

B. Prefix stage

In the prefix stage, group generate/propagate signals are computed at each bit using the given equations. The black cell (BC) generates the ordered pair in equation (7), the gray cell (GC) generates only left signal, following [4].

$G\,i:k = G\,i:k + P\,i:k\,.Gj\text{-}1:k$ (10)

$Pi:k = Pi:k\,.Pj\text{-}1:k$ (11)

More practically, the above two equations can be expressed by using a symbol"o" and it can be written as,

$Gi:k : Pi:k = (Gi:j : Pi:j)\ o\ (G\,j\text{-}1:k : P\,j\text{-}1:k)$ (12)

C. Final computation

In the final computation, the sum and carryout are the final output.

$Si = Pi\,.Gi\text{-}1:\text{-}1$ (13)

$Cout = Gn:\text{-}1$ (14)

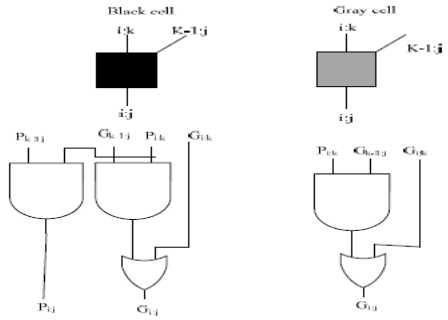

Fig.3: Black and Gray cells

Where "-1" is the position of carry-input. The generate/propagate signals can be grouped in different ways to get the same correct carries. Based on different ways of grouping the generate/propagate signals, different prefix architectures can be obtained. They are

i. Sparse Kogge Stone adder(SKA)
ii. Spanning Tree Adder (STA)
iii. Kogge Stone Adder (KSA)
iv. Brent Kung Adder (BKA)

Figure 3 shows the definitions of cells that are used in prefix structures, including BC and GC. For analysis of various parallel prefix structures, see [4], [5] & [6].

i. Sparse Kogge Stone adder (SKA)

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 terminates with 4 bit RCA's. Totally it uses 16 full adders. The 16 bit SKA is shown in figure 4.In this adder, first the input bits (a, b) are converted as propagate and generate (p, g).Then propagate and generateterms are given to BC's and GC's. The carries are propagated in advance using these cells. Later these are given to full adderblocks.

ii. Spanning Tree Adder (STA)

Like the SKA, this adder also terminates with a RCA. It also uses the BC's and GC's and full adder blocks like SKA's but the difference is the interconnection between them [9].The 16 bit STA is shown in the below figure 5.

ISSN: 2278 – 7798

iii. Kogge Stone Adder (KSA)

KSA is another of prefix trees that use the fewest logic levels. A 16-bit KSA is shown in Figure 6. The 16 bit kogge stone adder uses BC's and GC's and it won't use full adders. The 16 bit KSA uses 36 BC's and 15 GC's. And this adder totally operates on generate and propagate blocks. So the delay is less when compared to the previous SKA and STA. The 16 bit KSA is shown in figure 6.In this KSA, there are no full adder blocks like SKA and STA [7] & [8].

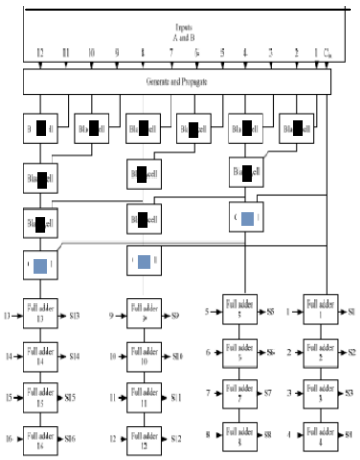

Fig.4:16-bit Sparse KoggeStone adder

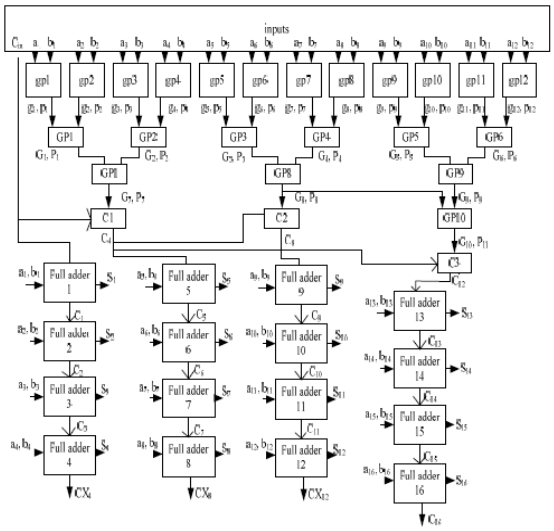

Fig.5:16-bit Spanning Tree adder

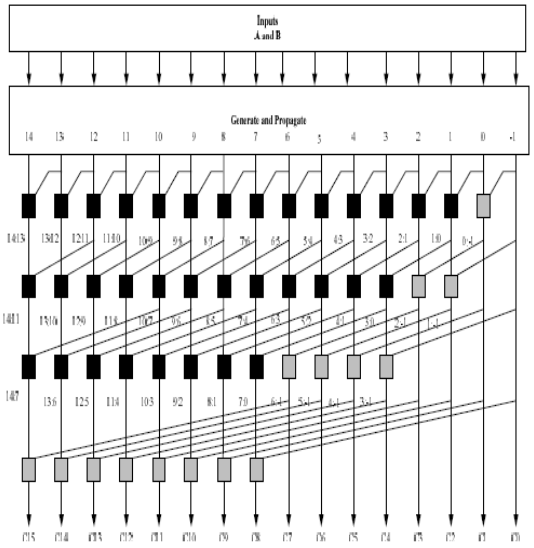

Fig.6:16-bit Kogge Stone adder

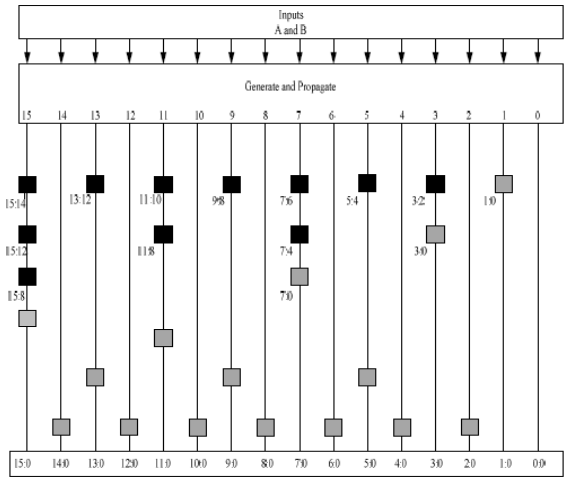

Fig.7:16-bit Brent Kung adder

iv. Brent Kung Adder (BKA)

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 occupies less area than KSA. The 16 bit BKA is shown in the below figure 7.

III. PROPOSED 64-BIT HYBRID ARCHITECTURE

The new hybrid adder, as shown in Figure 8, is made up of three modules including the generate/propagate generation (GPG), the prefix tree structure (PTS) and the CSA with add-one circuit. The initial stage of the GPG generates individual generate and propagate signals for each bit position. The middle stage of the PTS computes some specific carries to the next stage for the CSA. The final stage of the CSA with add-one circuit selects the proper sum as the output. I take

ISSN: 2278 – 7798

advantage of the initial stage of sharing generate and propagate signals in both PTS and CSA blocks to reduce the hardware overheads and to achieve more compact area.

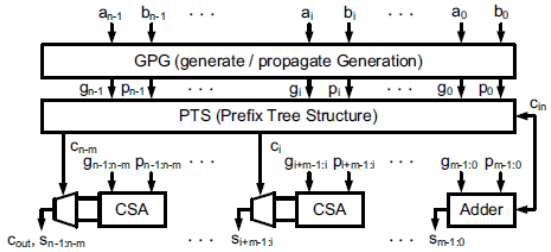

Fig.8: Block diagram of proposed hybrid architecture

A. Proposed 64-bit hybrid adder

For a PPA, the carry computation approach dominates the overall performance. In order to fetch faster addition, many parallel prefix tree topologies have been developed to give a good trade-offs among speed, area and power. The more efficient way is implementing parallelizable prefix computation by taking advantage of the associative operator "*o*" [10].The generate (*g*) and propagate (*p*) signals can be defined as follow:

$(g, p)\ o\ (g', p') = (g + pg', pp')$ (12)

Giving a series of bits i…j…k, the group generate/propagate pair ($g\ i{:}k$, $p\ i{:}k$) can be expressed in terms of input signals p *j* and g *j* from bit position i to k, respectively. Therefore,

$(g\ i{:}k, p\ i{:}k) = (gi,pi,)\ o...o\ (gj{+}1, p\ j{+}1)\ o\ (g\ j, p\ j)\ o(g\ j{-}1, p\ j{-}1)\ o…o\ (g\ k, p\ k)$ (13)

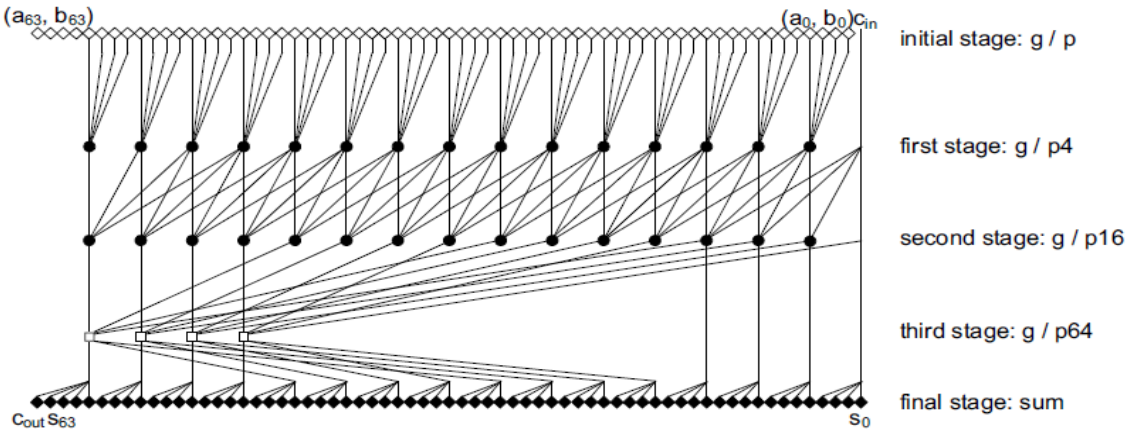

Fig.9: Proposed 64-bit hybrid adder

The proposed 64-bit hybrid adder is designed based on hybrid PTS and CSA as shown in Figure 9. For a group of four bits, a radix-4 64-bit hybrid adder is only composed of three stages. According to the relationship between stage and radix in a prefix tree [2, 11], the fewer stages depend on higher radix adopted, and the least stages are applied in prefix adder results in the potential higher speed and lowest power consumption. A 4-bit CSA is taken in my consideration is due to the delay time of a 4-bit carry ripple adder (CRA) module is slightly less than that of PTS. Consequently, every four-bit of PTS computes a carry for corresponding 4-bit CSA module.

To achieve the goal of high speed adder, carry-in signal (*cin*) directly connects to the second stage. The proposed structure differs greatly from other architectures in which carry-in signal are linked to the first stage. In such scheme, the critical path of proposed design has reduced load capacitance. As a result, the total performance of the adder will be boosted.

In the proposed 64-bit hybrid adder, the generate signal (*gi*) and propagate signal (*pi*) are generated at the initial stage. The radix-4 PTS, the middle stage, computes carries tothe CSA modules in final stage. For the first and second stages, every module, denoted as black circle in Figure 9, utilizes CMOS logic to complete the intermediate carry. To diminish the fan-out at the second stage as the fan-in at the third stage, CBA is adopted at the third stage for terminal carry output. This turns out to greatly reduce the wiring counts, so the third stage owns compact area and lower power consumption. And CBAs skip over long carry ripple to reduce the delay time with fewer transistor numbers. At the final stage, the 4-bit CSA quickly produce sums through MUXs when carry signal arrived.

The group generate/propagate (*g/p*) functions for each group of four bits, depicted as black circles of the first and the second stages in Fig. 9, are expressed as:

$g_{i+3:i} = g_{i+3} + p_{i+3}\ g_{i+2} + p_{i+3}\ p_{i+2}\ g_{i+1} + p_{i+3}\ p_{i+2}\ p_{i+1}\ g_i$ (14a)

$p_{i+3:i} = p_{i+3} p_{i+2} p_{i+1} p_i$ (14b)

ISSN: 2278 – 7798

IV. EXPERIMENT RESULTS

The proposed 64-bit hybrid adder of RTL schematic, view technological schematic diagrams, the summary reports and output wave forms are shown in below.

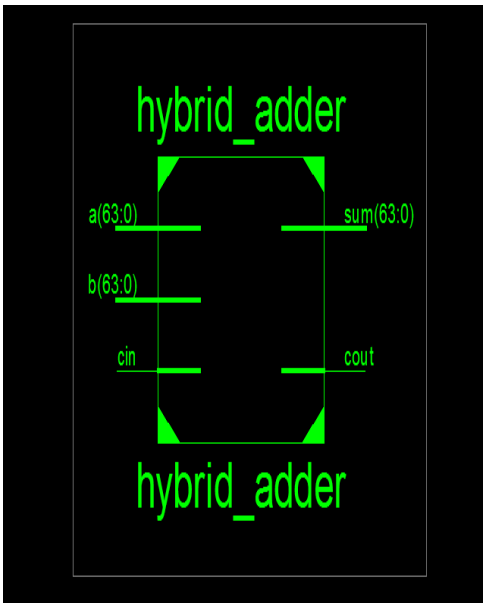

Figure 10: View RTL Schematic

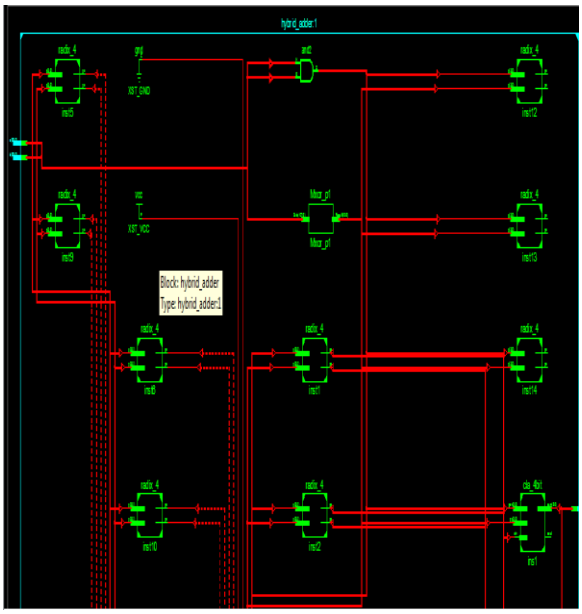

Fig.11: Internal structure of RTL Schematic

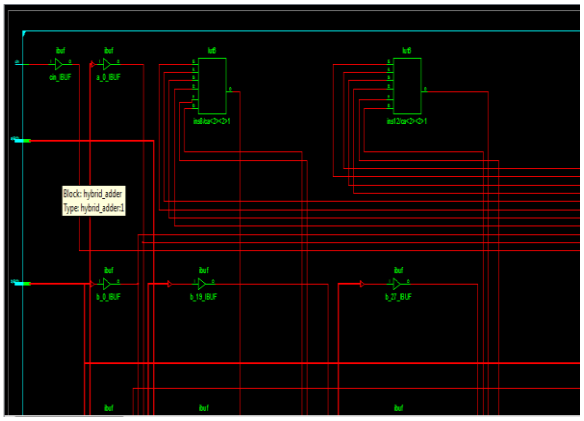

Fig.12: Internal structure of technology Schematic

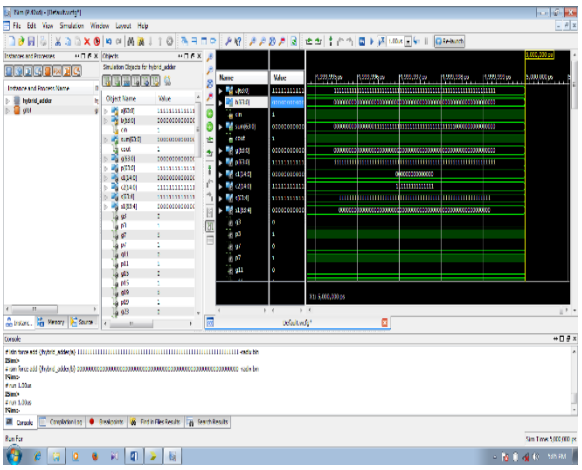

Fig13: Wave forms

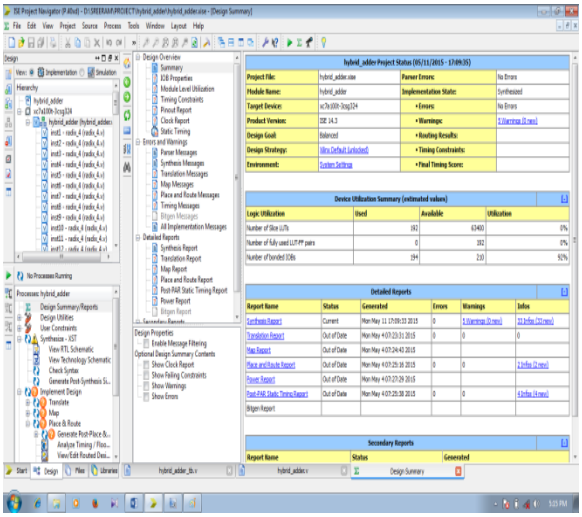

Fig.14: Design summary report

ISSN: 2278 – 7798

TABLE: Comparisons

S. No.	Adder Name (16bit)	Total Delay (ns) in Xilinx ISE 14.3	Power in Xilinx ISE 14.3 (mw)	Device Utilization (IOBs)	Percentage of utilization
1	Sparse Kogge Stone	4.46	42.38	50	24
2	Spanning Tree	3.98	42.38	50	24
3	Kogge Stone	4.53	42.38	50	24
4	Brent Kung	4.12	42.38	50	24
5	Proposed(64bit)adder	6.71	49.68	194	92

V. CONCLUSION

The proposed hybrid adder is composed of the radix-4 prefix tree structure and the CSA to benefit the high speed. The experimental results show that the proposed 64-bit hybrid adder has 6.71ns delay, consumes 8.58mw power and utilization 194 look at tables (LUTs). As a results, my design achieves superior speed, area and power features and therefore this adder plays a crucial role in ALUs, Processors, FPUs etc., compare to other designs.

REFERENCES

[1] D. Harris, "A Taxonomy of Parallel Prefix Networks," *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 2213-2217, 9-12 Nov. 2003.

[2] R. Zlatanovici, S. Kao and B. Nikolic, "Energy–Delay Optimization of 64-Bit Carry-Lookahead Adders With a 240 ps 90 nm CMOS Design Example," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 2, pp. 569-583, Feb. 2009

[3] A. Neve, H. Schettler, T. Ludwig and D. Flandre,"Power-Delay-product Minimization in High-Performance 64-bit Carry-Select Adders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 235-244, Mar. 2004.

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

[4] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEETrans. Comput., vol. C-31, pp. 260-264, 1982.

[5] D. Harris, "A Taxonomy of Parallel Prefix Networks," in Proc. 37th Asilomar Conf. Signals Systems and Computers, pp. 2213–7, 2003.

[6] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the EfficientSolution of a General Class of Recurrence Equations," IEEE Trans. OnComputers, Vol. C-22, No 8, August 1973.

[7] D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, "EasilyTestable Cellular Carry Lookahead Adders," Journal of ElectronicTesting: Theory and Applications 19, 285-298, 2003.

[8] T. Lynch and E. E. Swartzlander, "A Spanning Tree Carry LookaheadAdder," IEEE Trans. on Computers, vol. 41, no. 8, pp. 931-939, Aug.1992.

[9] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260-264, Mar. 1982.

[10] B. R. Zeydel, D. Baran and V. G. Oklobdzija, "Energy-Efficient Design Methodologies: High-Performance VLSI Adders," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 6, pp. 1220-1233, Jun.2010.

[11] T. Uehara and W. M. Vancleemput, "Optimal Layout of CMOS Functional Arrays," *IEEE Transactions on Computers*, vol. C-30, no. 5, pp. 305-312, May 1981.

First Author: VN.SREERAMULU(PG Scholar)
(Department of Electronics and Communication Engineering, Sreenivasa Institute of Technology and Management Studies (SITAMS), Chittoor,AP,India.

ISSN: 2278 – 7798