International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 12, December-2015

Design and Implementation of Parallel Prefix Adder for Improving the Performance of Carry Lookahead Adder

Ravi Payal
Senior Technical Officer,
CDAC,NOIDA

Mahima Goel
Mtech-VLSI,Student
CDAC,NOIDA

Prachi Manglik
Mtech-VLSI, Student
CDAC,NOIDA

Abstract - **In today's digital world, adder plays an important role in most of the digital circuits because of their use as basic entity in other operations such as subtraction, multiplication and division. Now-a-days, where everybody is working in the direction of miniaturization,three important aspects of design i.e area, power and delay needs to be balanced optimally.So,improving the performance of basic component(adder) would greatly advance the performance of the whole digital system,which is the ultimate goal of VLSI Design.In this paper we proposed a carry lookahead adder circuit in which carry generation network is implemented in the form of prefix trees,which leads to two types of Parallel Prefix Adders, Kogge Stone Adder and Ladner Fischer adder. The HDL used for design is Verilog and code was implemented in Xilinx Spartan 3E100CP132. A modified parallel prefix adder structure is being proposed which is above the traditional adders in terms of performance and can be widely employed in industries for achieving the desired performance targets.**

Keyword: Carry lookahead adder, Parallel Prefix Adders, Kogge Stone Adder, Ladner Fischer adder,Verilog

1. INTRODUCTION:

Basically there are two types of adders, Serial adders(which performs addition bit by bit and if speed is not the constraint, a cost effective option is to use serial adders). Another type which is parallel adders performs operation parallel that is adding bits simultaneously. The basic adders like Ripple carry adders, in which full adders are connected serially, due to which carry propagation forms the longest path leading to large delay and Carry Lookahead adders in which delay is less than that of ripple carry adders as carry is generated
Before hand, but they consumes a lot of area as circuit sharing is not there. The performance of adder and thus overall circuit is very important for the design as technology keeps on advancing. So, by changing the carry generation structure of carry Lookahead adder by parallel prefix trees we can improve the performance [2].

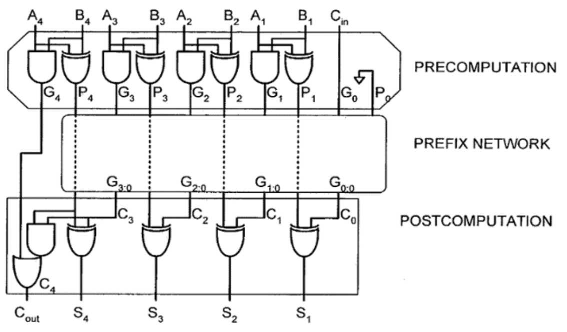


Fig.1 Three stage structure of parallel prefix adder [8]

Parallel prefix adder structure consist of three steps as shown in (fig.1), which is similar to carry Lookahead adder structure but the difference lies in 2nd stage which is about the generation of the carry signals. The process can be stated as [8]:

- Computation of carry generation and propagation: or PRE PROCESSING STAGE:The pair of operand bit(a_i,b_i) is taken and output generate bit(gi) and propagate bit(pi) is calculated. 'gi' indicates whether a carry is generated from that bit and 'pi' indicates whether a carry is propagated through that bit or not.
  p_i=a_i xor b_i
  g_i=a_i and b_i
- Calculation of the carries or CARRY GENERATION NETWORK
  Pi:j = Pi:k+1 and Pk:j
  Gi:j = Gi:k+1 or (Pi:k+1 andGk:j)

- Calculate the final sum and carry signals: or POST PROCESSING STAGE[4]:

Si= Pi xor Ci-1

Ci = Gi:0or (Cin and Pi:0)

For implementing 2nd stage or carry generation network, we need to understand prefix sum operator which is given in section 2.After that two structures of carry tree adders and there modifications are described in section 3 and 4.Section 5 contains simulation and synthesis reports and Section 6 consist of Conclusion

2.CARRY GENARATION NETWORK:

The second stage of parallel prefix adders is about constructing parallel prefix trees. That is why these adders are also known as Carry Tree adders[1].The most important advantage of tree structured adder is that the critical path of carry out signal is of the order of $\log_2 N$,wher N is the number of input bits [7].Parallel prefix operation has three components, Prefix, which is the number of inputs given to a node (here we are using prefix 2).Next is Parallel which involves the simultaneous execution of operations. This is achieved by segmentation into smaller pieces that are computed in parallel and then comes operation; it can be any arbitrary primitive operator "°" that is associative and hence parallelizable. Here we are using prefix sum operator which is associative in nature and thus parallelizable.
Prefix Sum can be understood as, if we are given an ordered set A of n elements and a binary associative operator $\oplus$,such as$A=\{a_0,a_1,a_2,\ldots\ldots a_n\}$,then we have to compute the ordered set$\{a0,(a0 \oplus a1),(a0 \oplus a1 \oplus a2), \ldots\ldots\}$.
Prefix sum operator, has applications in many other forms such as Graycodes, Parallel Algorithms, PRAM and many more. Here we are using binary addition as a prefix sum problem: if input is a pair of generate(g_i) and propagate(p_i) signals [7] ,

$$(g_n,p_n),(g_{n-1},p_{n-1})\ldots\ldots(g_0,p_0)$$

than output is a new vector of pairs,

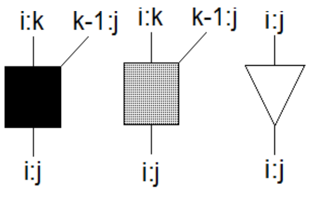
$$(G_n,P_n),(G_{n-1},P_{n-1}),\ldots(G_0,P_0)$$

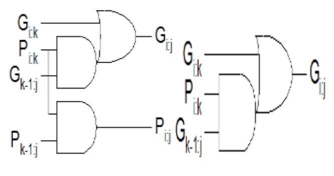
Output vector is calculated as:

$$(g_x,p_x) \circ (g_y,p_y) = \{(g_x+p_x.g_y),(p_x.p_y)\}$$

Where $(G_0,P_0) = (g_0,p_0)$

The black cell(fig.2) has two pairs of generate and propagate signals (gi, pi) and (gj, pj) as input and gives output as (G,P). The grey cell(fig.2) takes two pairs of generate and propagate signals (gi, pi) and (gj, pj) as input and computes generate signal only (G).Buffers(fig.2) are used to reduce the loading effect.[6,8]





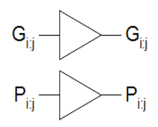


Fig. 2 Black,Grey and Buffer with their logical diagram [8]

3. KOGGE STONE ADDER

Kogge Stone adder is a parallel prefix adder, which was initially developed by Peter M.Kogge and Harold S. Stone. It generates the carry signal in $O(\log_2 N)$ time(N is the number of input bits). Using prefix sum problem which involves the use of prefix sum operator(◦) and Recursive doubling algorithm the carry generation network is constructed in the form of trees. The basic tree is constructed in prefix 2 form, where each node processes 2 inputs. The number of nodes at first stage(as shown in fig.3) is equal to the number of input bits. At first stage generate(g_i) and propagate(p_i) signals corresponding to each bit is calculated. Now, if the tree levels are numbered as k, with first level as k=0, for all further tree stages every column shifts its value to 2^{k-1} greater than their own. In the third stage (as shown in fig.3) the sum and carry are generated. All the nodes in 2nd stage(fig.3) are prefix sum operator working on 2 inputs and generating outputs [4,9].

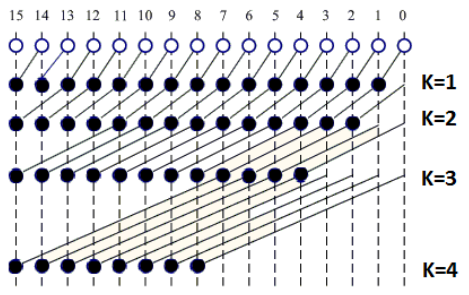


Fig.3 16 bit Kogge Stone Adder[8]

Kogge stone adder is the fastest adder but speed comes at the cost of increased area, complex wiring and increased number of cells but it has minimum number of Fan-out which is fixed to 2. [3] (table 1).

Initially, the adder designed has all the nodes present in the tree structure as black cells (fig. 2). For better performance in terms of area and delay, modifications are done. First method to improve the performance is replacing the cells with grey cells (fig.2), wherever the group propagate ($P_{i:j}$) is not required as shown in (fig.4)[3].This reduces the unwanted calculations and thereby improving the performance as shown in table .Second method suggested is removing the redundant black cells, as grey cells are used to calculate the generate bit in final stage and so they cannot be removed and appropriate routing is done after removing redundant black cells to achieve the result (fig.5). Third method suggested is based on the targeted implementation device. Since we are targeting to implement it on FPGA, an FPGA contains number of slices and each slice further contains multiplexers, Look up tables etc. So, Black and

Grey cells are designed in terms of multiplexers (fig.6) to improve the performance as shown in table 2 [11].

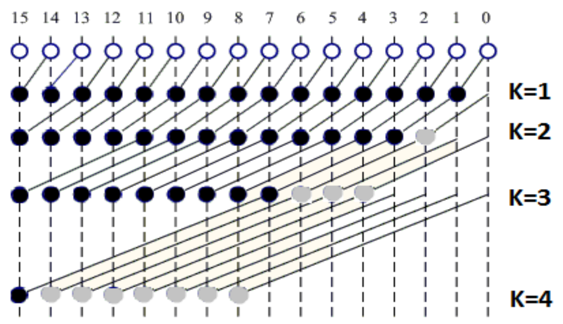


Fig. 4:16 bit modified (1) Kogge Stone Adder

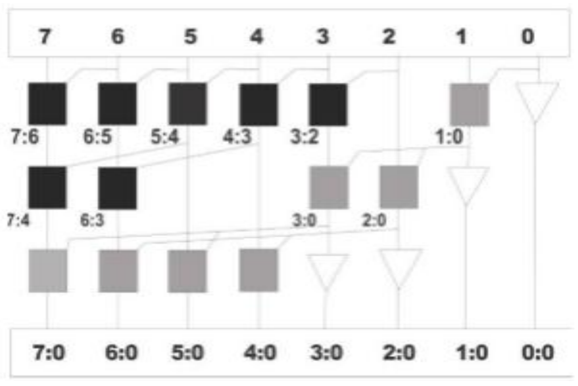


Fig.5: 8 bit modified (2) Kogge Stone Adder

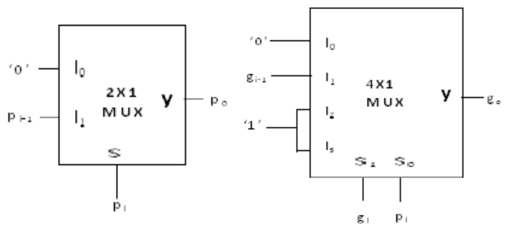


Fig.6:Multiplexers implementing prefix sum operator function modification(3) [12]

4.LADNER FISCHER ADDER

Second parallel prefix adder architecture is Ladner Fischer adder. In this also precomputation and post computation are same as the carry lookahead adder only the prefix network ischanged.This is based on Odd Even algorithm and prefix sum problem. In this adder we will divide our total no. of bits in 2 blocks. In this carry generated by the last bit will be propagated in all the inputs of second block. So starting with the process N/2 prefix blocks is further divided into two blocks and we will divide it till we get 1 bit block. This can also be understood in terms of reverse tree, for example if we consider 16 bit adder, we start by dividing into two 8-bit blocks then each 8-bit block is further divided into 4 bit blocks then further 4-bit blocks divided into 2-bit blocks as shown in fig.7 [4,9]

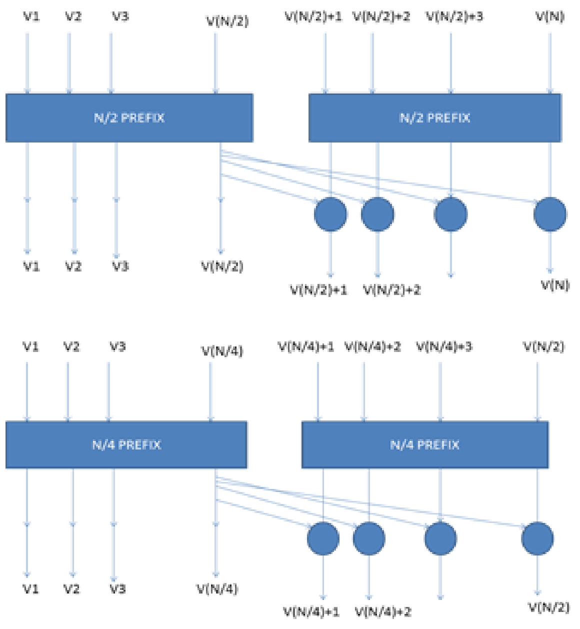

V1 V2 V3 V(N/2) V(N/2)+1 V(N/2)+2 V(N/2)+3 V(N)
N/2 PREFIX
N/2 PREFIX
V1 V2 V3 V(N/2) V(N/2)+1 V(N/2)+2 V(N)
V1 V2 V3 V(N/4) V(N/4)+1 V(N/4)+2 V(N/4)+3 V(N/2)
N/4 PREFIX
N/4 PREFIX
V1 V2 V3 V(N/4) V(N/4)+1 V(N/4)+2 V(N/2)

Fig.7 Ladner Fischer adder working

Initially all the blocks are the computation blocks in which both generate(G) and propagate(P) signals are computed(fig.8).To reduce the computations another method is used to convert black cells into grey cells as done in Kogge Stone adder design(fig.9).One more method which is proposed above is employed here by using multiplexers in place of basic gates to implement prefix sum operator problem.(fig.6)

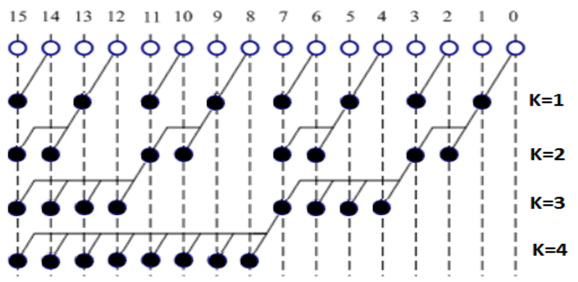


Fig.8 16 bit Ladner Fischer Adder[8]

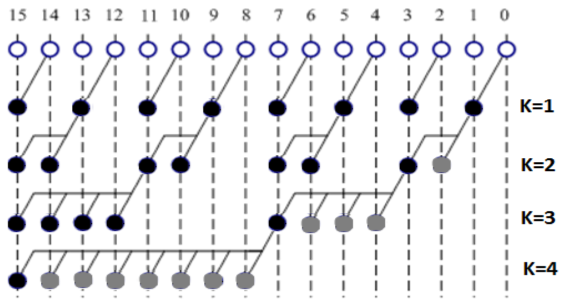


Fig.9 16 bit modified (1) Ladner Fischer Adder

Ladner Fischer adder shows many aspects in which it is better than than the fastest Kogge Stone adder (Table 1) .Here the number of cells used are less than that of Kogge Stone adder and it's any modification. So, the area required is less and leads to a lower level of wiring complexity[3]. But this structure has a variable Fan-out (N/2), where N is the number of input bits,which leads to a trade off as Fan-out is the out degree of a node, as it increases, it is favourable in terms of area, as area reduces but at the same time it increases delay, because now it has to charge more capacitances. This effect will definitely appears in our synthesis.(Table 2)

5.SIMULATION AND SYNTHESIS RESULT AND COMPARISON:

Different adders were designed in Verilog HDL using Mentor Graphics's Questasim 10.2c software for simulation and synthesis is performed on Mentor Graphics's Precision RTL plus 2012b.10.Figure 10,11,12 shows the simulated waveform and table 2 contains the synthesis report. The code was implemented on Xilinx Spartan 3E100CP132 FPGA using ISE Design Suite 14.5 . Hardware implementation of 4 bit adder is done on Digilent Basys2 Board,'a' and 'b' of 4 bit each are given on 8 input pins and cin on the push button and output 's' and 'cout' of 4 bit and 1 bit respectively are observed on LEDs. Implementation of 4 bit Kogge Stone Adder with and without carry is shown in figure 16 and Figure 17.

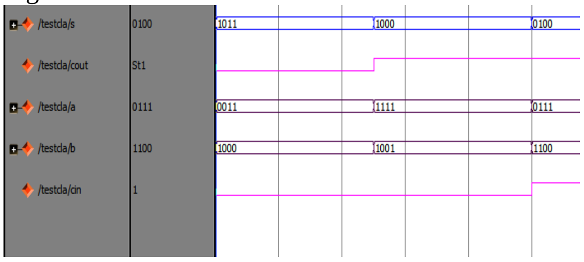


Fig.10:Simulated output of 4 bit Carry Lookahead Adder

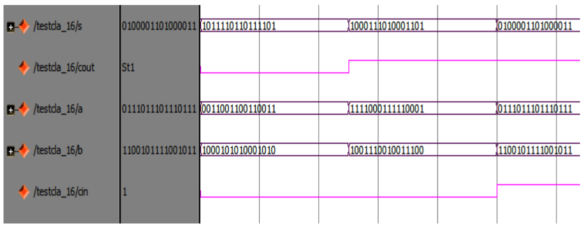


Fig.11:Simulated output of 16 bit Carry Lookahead Adder

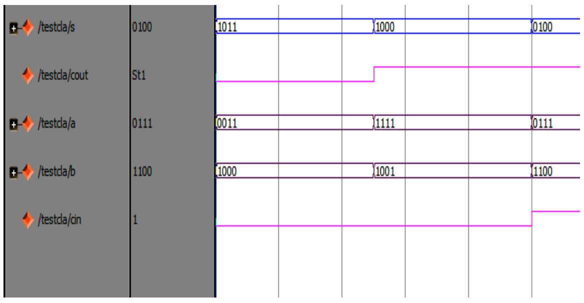


Fig.12:Simulated output of 4 bit Kogge Stone Adder

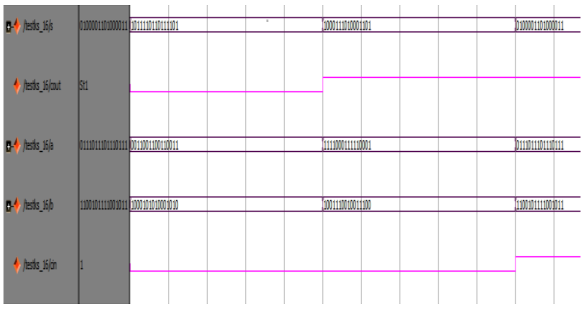


Fig.13:Simulated output of 16 bit Kogge Stone Adder

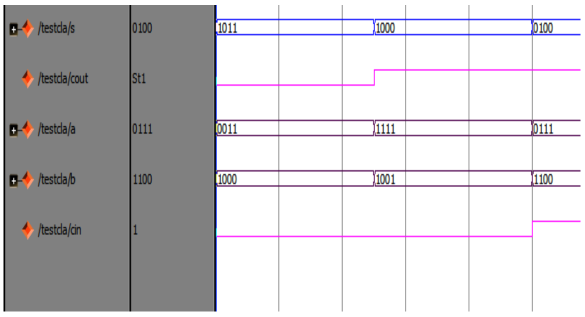


Fig.14:Simulated output of 4 bit Ladner Fischer Adder

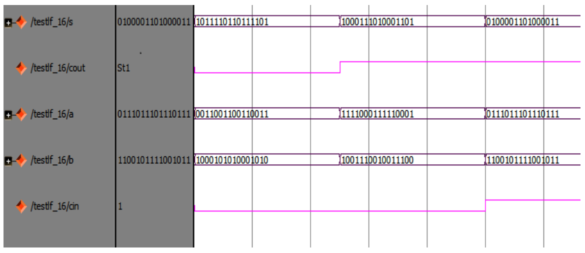


Fig.15:Simulated output of 16 bit Ladner Fischer Adder

	KOGGE STONE	LADNER FISCHER
No. of Cells	More$(N(\log_2 N1)+1)$	Less$((N/2)*\log_2 N)$
Area	Large	Small
Max Fan out	2	N/2
Wiring Complexity	High	Low

Table 1:Comparison between Kogge Stone Adder and Ladner Fischer Adder

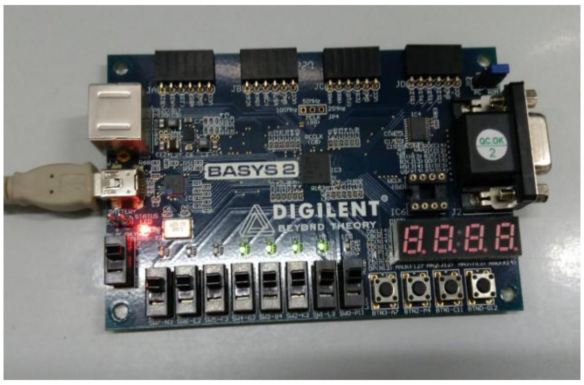


Fig.16:4bitadderBasys2boardFPGAimplementation(a=4’b1111,b=4’b1111,cin=0)

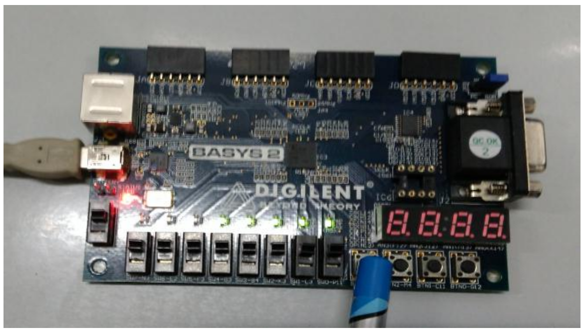


Fig.17: 4 bit adder Basys2 board FPGA implementation(a=4’b1111,b=4’b1111,cin=1)

	Delay(ns)								
	Original			Modified 1			Modified2		
	4bit	8 bit	16 bit	4bit	8 bit	16 bit	4bit	8 bit	16 bit
Carry Lookahead	7.287	8.268	10.086	-	-	-	-	-	-
Kogge Stone	6.375	6.426	9.243	6.216	6.338	9.230	5.149	6.314	8.277
Ladner Fischer	6.102	6.361	9.219	6.108	6.217	9.175	5.027	6.147	8.312

	Area																	
	Original						Modified 1						Modified2					
	4bit		8 bit		16 bit		4bit		8 bit		16 bit		4bit		8 bit		16 bit	
	L	S	L	S	L	S	L	S	L	S	L	S	L	S	L	S	L	S
Carry Lookahead	9	5	32	16	60	30	-	-	-	-	-	-	-	-	-	-	-	-
Kogge Stone	8	4	31	16	77	39	5	3	11	6	27	14	5	3	13	7	29	15
Ladner Fischer	8	4	26	13	28	56	5	3	10	5	24	12	5	3	12	6	28	14

Table 2:Comparison betweenCarry Lookahead Adder, Kogge Stone Adder and Ladner Fischer Adder

6.CONCLUSION:

The proposed adders are efficient in delay and area comparison to the basic Carry Lookahead adder and the traditional tree adders. Among the Kogge Stone adder and Ladner Fischer adder, when the number of bits are less, Ladner Fischer performance is better than the Kogge Stone adder's performance as shown in this paper but as the number of bit increases and reaches to a very large number Kogge Stone becomes faster than Ladner Fischer adder, because of variable fan out of Ladner Fischer adder.

REFRENCES:

[1] Neil H. E. Weste, David Money Harris,CMOS VLSI esign,4thedition,Pearson-Addison-Wesley
[2] KonstantinosVitoroulis,Asim J. Al-Khalili",Performance of Parallel Prefix Adders implemented with FPGA technology",Circuits and Systems,2007.NEWCAS 2007,IEEE
[3] T.KIRAN KUMAR "Design of High Speed 128 bit Parallel Prefix Adders "Int. Journal of Engineering Research and Applications www.ijera.com ISSN : 2248-9622, Vol. 4, Issue 11(Version 3), November 2014, pp.112-115
[4] SwaroopGhosh, Patrick Ndai, Kaushik Roy. "A Novel Low Overhead Fault Tolerant Kogge-Stone Adder Using Adaptive Clocking",Proceedings -Design, Automation and Test in Europe, DATE (2008) 366-371;
[5] A.V Pradeep Chandra,K.Jayaprasada Rao and T.Sivaram Gupta,"Design of Parallel Prefix Adders", School of Electronics Engineering(SENSE),VIT University, Vellore
[6] K.Babulu, Y.Gowthami,"Implementation and Performance Evaluation of Prefix Adders using FPGAs" IOSR Journal of VLSI and Signal Processing,IOSR-JVSP)ISSN: 2319 – 4200, ISBN No. : 2319 – 4197 Volume 1, Issue 1 (Sep-Oct. 2012), PP 51-57
[7] Hoe,D.H.K;Martinez,C.;Vundavalli,S.J "Design and characterization of parallel prefix adders using FPGAs"System theory(SSST),2011 IEEE 43rd Southeastern Symposium on Year 2011
[8] Geeta Rani,"Delay Analysis of Parallel-Prefix Adders",International Journal of Science, Engineering and Technology Research (IJSETR),July 14
[9] CH SudhaRani,CH.Ramesh,"Design and Implementation of high performance parallel prefix adders", IJIRCCE,vol.2,Issue 9,September 2014
[10] Matthew Ziegler, Mircea Stan, "Optimal Logarithmic Adder structures with a fan-out of two for minimizing area delay product," IEEE 2001.
[11] JianhuaLiuZhu, Haikun, Chung-Kuan Cheng, John Lillis, "Optimum prefix Adders in a Comprehensive Area,Timing and power Design Space"., Proceeding of the 2007 Asia and South pacific Design Automation conference.Washington, pp.609-615,jan 2007.
[12] TaekoMatsunaga and YusukaMatsunaga., "Timing-Constrained Area minimization Algorithm for parallel prefix adders", IEICE TRANS, Fundamentals, vol.E90-A, No.12 Dec, 2007.
[13] Ravi Payal, "Simulation and Synthesis Model for the Addition of Single Precision Floating Point Numbers Using VERILOG" International Journal of Engineering Research & Technology(IJERT), Vol.2 - Issue 9 (September - 2013)