# Design of High Speed Low Power Parallel Prefix Adder

[1]A.Anandha Raja, [2]R.Sandeep, [3]K.P.Sandeep, [4]K.Saravanavel
[1]Assistant Professor, Dept.of ECE, SNS College of Technology, Saravanampatty, Coimbatore, India
[234] UG Scholars, Dept.of ECE, SNS College of Technology, Saravanampatty, Coimbatore, India

***Abstract :*** This paper addresses the parallel prefix adder synthesis which targets of area minimization under given bitwise timing constraints. Parallel-prefix adders offer a highly efficient solution to the binary addition problem and are well-suited for VLSI implementations. In this paper, a novel framework was introduced, which allows the design of parallel-prefix Ling adders. The proposed approach saves one-logic level of implementation compared to the parallel-prefix structures proposed for the traditional definition of carry look ahead equations and reduces the fan out requirements of the design. The proposed adder architecture was **IMPLEMENTED FOR 16-BIT, 32-BIT WIDTH OPERANDS USING XILINX 14.5 VERSION OF VHDL WITH TARGETED DEVICE OF SPARTAN 3E. THE** experimental results are compared with the basic adder variants such as Ripple Carry Adder, Carry Look ahead adder, Carry Bypass Adder, Carry Select Adder.

***IndexTerms*** **- Ripple Carry Adder (RCA), Carry Look Ahead Adder (CLA), Carry Bypass Adder (CBA), Carry select adder (CSLA), Parallel Prefix Adder (PPA), Very Large Scale Integration (VLSI). Adders, Parallel-prefix carry computation, Computer arithmetic, VLSI design.**

## I. INTRODUCTION

Binary addition is the most fundamental and frequently used arithmetic operation. A lot of work on adder design has been done so far and much architecture have been proposed. Among them, parallel prefix adders generalize carry look-ahead idea for faster carry propagation, and many regular structures, such as Brent-Kung , Kogge- Stone, and Sklansky parallel prefix adders, have been proposed. Not fixed structures, but some algorithms to generate parallel prefix adders also have been proposed. An advantage of algorithmic generation over fixed structures is that global structures of parallel prefix adders can be changed flexibly according to each context, such as bitwise input arrival times and output required times, which can result in faster or smaller circuits especially when bitwise timing constraints are not uniform.

Flexible parallel prefix adders can be synthesized through two-stage flow, that is, generation of global structures at technology-independent level and technology mapping. A global structure of a parallel prefix adder at technology-independent level can be visualized as a prefix graph whose node corresponds to some basic operation in carry propagation. Area and delay of a parallel prefix adder are roughly measured by the total number of nodes of a prefix graph and the level of nodes. By dividing two processes and using rough measures, the former process can be more tractable in formal way regardless of details of technology. These measures could be inaccurate, especially for delay. The levels of nodes correspond to load-independent delay model, so there is no consideration on effect of fan outs. But for this, we could assume gain-based technology mapping as a subsequent process to make this measure more reliable.

This paper addresses parallel prefix adder synthesis which targets area minimization under given bitwise timing constraints. This problem is captured as a problem to minimize the number of nodes of prefix graphs under timing constraints. Timing-constrained area minimization for a directed acyclic graph is inherently difficult to be solved exactly. Though there is an exact approach for delay optimization, only approaches based on local transformations have been proposed as for area minimization. They may generate good results for some examples, but we cannot judge whether there is any room for improvement since those approaches are lack of global view and cannot guarantee the optimality.

## II. BASIC ADDER MODELS

### A. RIPPLE CARRY ADDER

The Ripple carry adder is one of the simplest adders to implement. This adder takes in two N-bit inputs and produces (N+1) output bits as N-bit Sum and 1-bit carry out bit. The Ripple carry adder is built from N full adders cascaded together, with the carry out bit of one full adder is connected to the carry in bit of the next full adder. Ripple carry adder for 4-bit addition. The implementation of 16-bit Ripple carry adder based on 4-bit RCA is shown in figure 01.

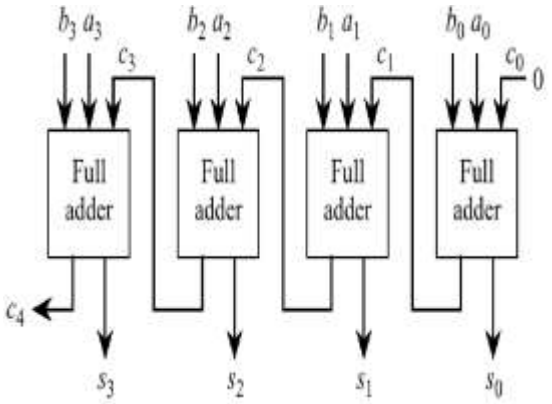


Figure 01: Ripple Carry Adder

### B. CARRY LOOK AHEAD ADDER

A Carry Look Ahead adder (CLA) is a type of adder used in digital circuits. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, ripple carry adder[16] for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits. The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits. The Kogge-Stone adder and Brent-Kung adder and Ladner- Fischer are examples of this type of adder. To reduce the computation time, engineers devised faster ways to add two binary numbers by using carry-look ahead adders. They work by creating two signals (P and G) for each bit position, based on if a carry is propagated through from a less significant bit position (at least one input is a '1'), a carry is generated in that bit position (both inputs are '1'), or if a carry is killed in that bit position (both inputs are '0'). In most cases, P is simply the sum output of a half-adder and G is the carry output of the same adder. After P and G are generated the carries for every bit position are created. Some advanced carry-look ahead architectures are the Brent-Kung adder, and the Kogge-Stone adder and Ladner-Fischer adder. The 4 bit CLA was shown in figure 02

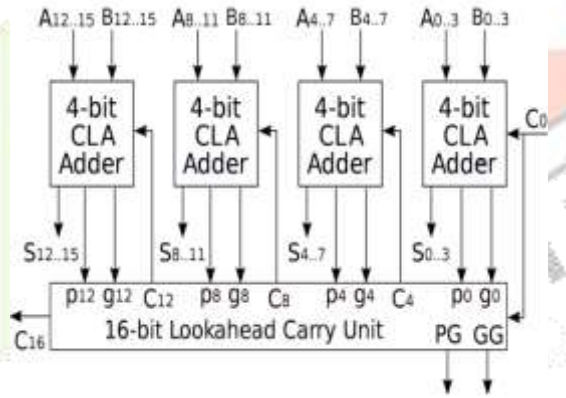


Figure 02: Carry look ahead adder

### C. CARRY SKIP ADDER

The critical path delay of the CSKA is much smaller than throne in the RCA, whereas its area and power consumption are similar to those of the RCA. In addition, the power-delay product (PDP) of the CSKA is smaller than those of the CSLA and PPA structures. In addition, due to the small number of transistors, the CSKA benefits from relatively short wiring lengths as well as a regular and simple layout. The comparatively lower speed of this adder structure, however, limits its use for high-speed applications. In this paper, given the attractive features of the CSKA structure, we have focused on reducing its delay by modifying its implementation based on the static CMOS logic. This carry skip adder was shown in figure 03.

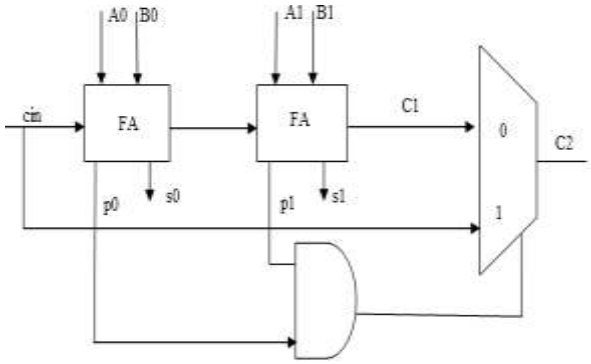


Figure 03: Carry skip adder

### D. CARRY SELECT ADDER(CSLA)

The carry select adder divides the adder into blocks that have the same input operands except forth carry in. Carry select adder perform two additions, one assuming the carry in is 1(Cin=1) andone assuming the carry in is 0 (Cin= 0), and chooses between the two results once the actual carry in is known. The logic circuit of 4-bit Carry select adder. The implementation of 16-bit Carry select adder based on 4-bit CSLA

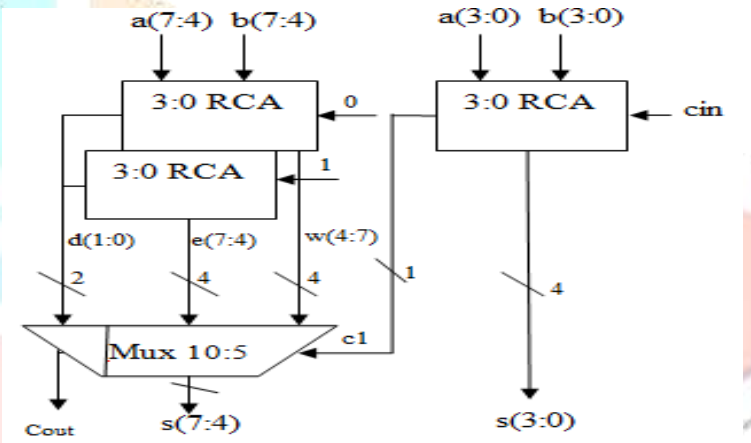


Figure 04: Carry Select Adder

## III. PROPOSED ADDERS

The PPA is like a Carry Look Ahead Adder. The production of the carriers the prefix adders can be designed in many different ways based on the different requirements. We use tree structure form to increase the speed of arithmetic operation. Parallel prefix adders are faster adders and these are faster adders and used for high performance arithmetic structures in industries. The parallel prefix addition is done in 3 steps.

1. Pre-processing stage
2. Carry generation network
3. Post processing stage

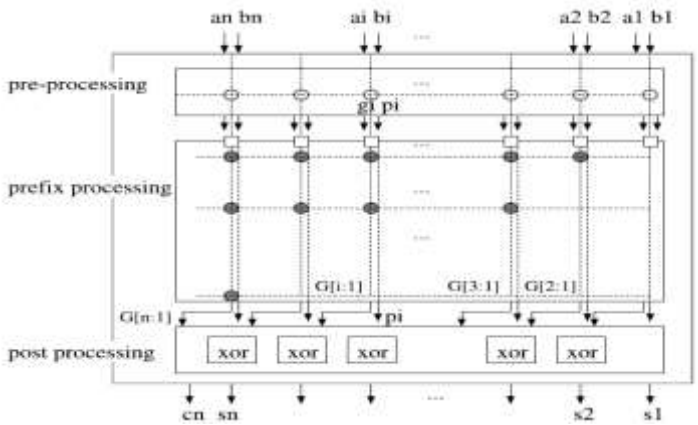


Figure 05: Structure of parallel prefix adder

**A. Pre-processing stage**

In this stage we compute, the generate and propagate signals are used to generate carry input of each adder. A and B are inputs. These signals are given by the equation 1& 2.

$$Pi=Ai\ Bi \quad (1)$$

$$Gi=Ai \cdot Bi \quad (2)$$

**B. Carry generation network**

In this stage we compute carries corresponding to each bit. Execution is done in parallel form and after the computation of carries in parallel they are divided into smaller pieces. Carry operator contain two AND gates, one OR gate. It uses propagate and generate as intermediate signals which are given by the equations 3 & 4.

$$P(i:k) = P(i:j) \cdot P(j-1:k) \quad (3)$$

$$G(i:k) = G(i:j) + (G(j-1:k) \cdot P(i:j)) \quad (4)$$

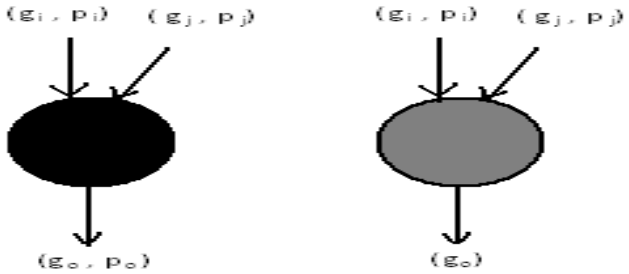


Figure 06: (a) O3black operator (b) O3gray operator

The black operator receives two sets of generate and propagate signals (gi,pi),(gj,pj), computes one set of generate and propagate signals (go , po) by the following equations:

$$g0 = gi + pigi \quad (5)$$

$$p0 = pi\ pj \quad (6)$$

The gray operator receives two sets of generate and propagate signals (gi, pi), (gj, pj), computes only one generate signal with the same equation as in equation (8).

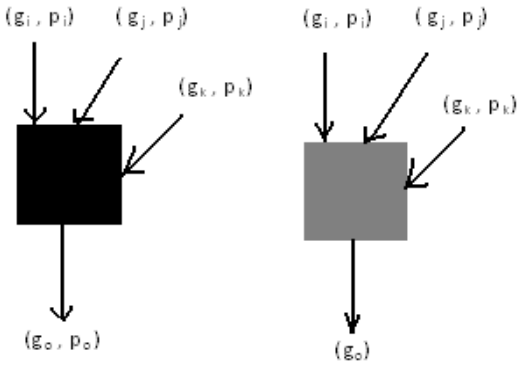


Figure 07: (a) O3black operator (b) O3gray operator

$$g0 = gi + pigj + pipjpk \quad (7)$$

$$p0 = pipjpk \quad (8)$$

The O3black operator, which takes three pairs of generate and propagate values (gi , pi),(gj, pj), (gk , pk) as inputs and produces the generate and propagate output values (go , po). The O3gray operator, which takes three pairs of generate and propagate values (gi , pi),(gj, pj), (gk , pk) as inputs and produces only one generate signal output as per equation. The implementation of these operators is done using multiplexer based design.

**C. Post processing**:

Generate each sum bit *si*

$$ci = G[i{:}1]$$

$$si = pi \oplus ci{-}1$$

Parallel prefix adders consist of the above three parts though both pre-processing and post-processing parts have fixed structures, the prefix processing part has high flexibility. Since ◦ satisfies associatively, prefix computation does not have to be done serially. How to parallelize prefix computation affects the qualities of parallel prefix adders.

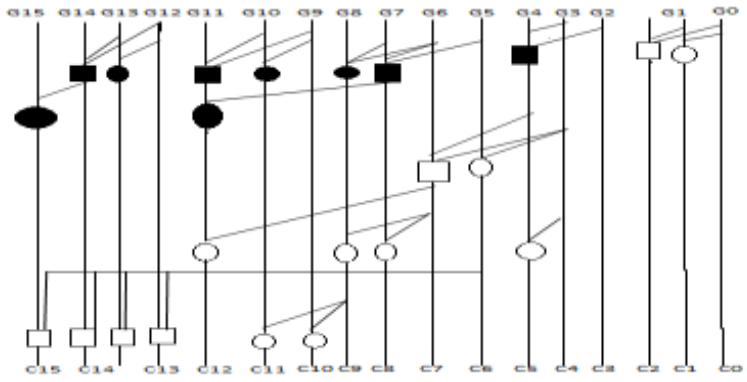

Figure 08: Hybrid parallel prefix adder for 16-bit

## IV. SIMULATION & RESULT COMPARISON

TABLE1: Results of various adders

| ADDER | POWER | AREA | DELAY |
|---|---|---|---|
| Ripple carry adder | 28 | 31 | 20.129 |
| Carry look ahead adder | 28 | 31 | 20.317 |
| Carry skip adder | 28 | 31 | 18.352 |
| Carry select adder | 28 | 31 | 14.384 |
| Proposed adder | 28 | 31 | 14.213 |

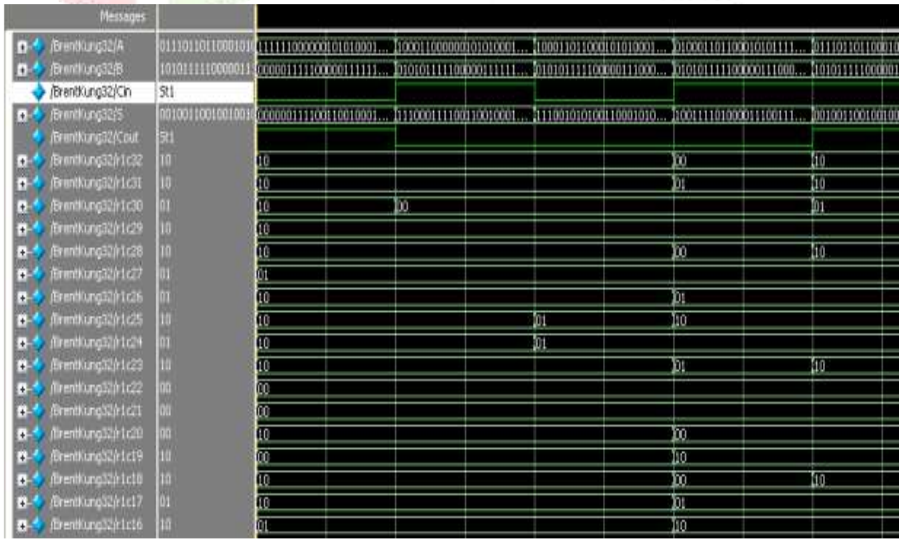

Figure 09: Simulated output of 32-bit hybrid adder

## V. CONCLUSION

This paper presents the implementation of carry look-ahead adder, carry bypass adder, carry select adder and the proposed hybrid parallel prefix adder for 16-bit and 32-bit addition. To evaluate the performance of these adders, Xilinx 14.5 version of VHDL is used with the targeted device of Spartan 3E family of device XC3S500E. The proposed adders are faster because of less delay and area efficient compared to other basic adders. Among these three prefix adders Ladner-Fischer adder has better performance compared to remaining adders. The performance comparisons between these adders are measured in terms of area and delay. It would be interesting to investigate the design of the 128 and 256 bit adders. These adders are popularly used in VLSI implementations.

## REFERENCES


**[1]** J. Skalansky, "conditional sum additions logic", IRE Transactions, Electronic Computers, vol. EC – 9, pp. 226 - 231, June 1960.

**[2]** Y.Choi and E.E.Swartz lander, Ir, "Parallel Prefix adder design with matrix representation",, in Proc.17th IEEE symposium on computer Arithmetic (ARITH), PP 90-98,2005.

**[3]** Kogge P, Stone H, "A parallel algorithm for the efficient solution of a general class Recurrence relations", IEEE Trans. Computers, vol.C-22, No.8, pp. 786-793, Aug.1973.

**[4]** GiorgosDimitrakopoulos and DimitricNikolos, "High Speed Parallel –Prefix VLSI Ling Adders", IEEE Trans on computers, Vol.54, No.2, Feb 2005.

**[5]** Han T, Carlson D, "Fast area-efficient VLSI adders", Proc.8th.symp.Comp.Arit.pp.49-56, Sep.1987.

**[6]** TaekoMatsunaga and Shinji Kimura, YusukaMatsunaga, "Synthesis of parallel prefix adders considering switching activities", IEEE International Conference on computer design,

**[7]** Reto Zimmermann. Binary Adder Architectures for Cell-Based VLSI an their Synthesis. Hartung-Gorre, 1998.

**[8]** Y. Choi, "Parallel Prefix Adder Design," Proc. 17th IEEE Symposium on Computer Arithmetic, pp 90-98, 27th June 2005.

**[9]** D. Harris, "A taxonomy of parallel prefix networks," in Signals, Systems and Computers,2003. Conference Record of Thirty Seventh Asilomar Conference on, 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]** K.Vitoroulis and A. J. Al-Khalili "Performance of Parallel Prefix Adders Implemented with FPGA technology," IEEE Northeast Workshop on Circuits and Systems, pp. 498-501, Aug. 2007.

**[13]** D. H. K. Hoe, C. Martinez, and J. Vundavalli, "Design and Characterization of Parallel Prefix Adders using FPGAs, IEEE 43rd Southeastern Symposium on System Theory, pp. 170-174, March 2011.

**[14]** T. Matsunaga, S. Kimura, and Y. Matsunaga."Power-conscious syntheses of parallel prefix adders under bitwise timing constraints," Proc. the Workshop on Synthesis And System Integration of Mixed Information technologies(SASIMI), Sapporo, Japan, October 2007, pp. 7–14.