International Journal of Engineering Research and Development
***e**-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com*
Volume 3, Issue 6 (September 2012), PP. 62-67

Comparison of Parallel Prefix Adders Performance in an FPGA

Prof. S.V.Padmajarani[1], Dr. M.Muralidhar[2]
[1]*H.O.D., Department of E.C.E, Jagan's College of Engineering & Technology, Nellore - 524 320, Andhra Pradesh, India*
[2]*Principal, S.V.College of Engineering & Technology, Chittor– 517507, Andhra Pradesh, India*

Abstract*—Parallel prefix adder is the most flexible and widely used for binary addition. Parallel Prefix adders are best suited for VLSI implementation. Number of parallel prefix adder structures have been proposed over the past years intended to optimize area, fan-out, logic depth and inter connect count. This paper investigates the performance of different parallel prefix adders when they are implemented on an FPGA. The comparison has been done for 16-bit architectures. Xilinx Spartan 3E FPGA is used for implementation. Comparison is done for two parameters, speed and area. Speed performance comparison is with respect to delay, area comparison is with respect to number of Look up tables, slices and overall gate count.

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 many architectures have been proposed. When high operation speed is required, tree structures like parallel-prefix adders are used [1] - [10]. In [1], Sklansky proposed one of the earliest tree-prefix is used to compute intermediate signals. In the Brent-Kung approach [2], designed the computation graph for area-optimization. The KS architecture [3] is optimized for timing. The LF architecture [4], is proposed, where the fan-out of gates increased with the depth of the prefix computation tree. The HC adder architecture [5], is based on BK and KS is proposed. In [6], an algorithm for back-end design is proposed. The area minimization is done by using bitwise timing constraints [7]. In [8], which is targeted to minimize the total switching activities under bitwise timing constraints. The architecture [9], saves one logic level implementation and reduces the fan-out requirements of the design. A fast characterization process for Knowles adders is proposed using matrix representation [10].

The Parallel Prefix addition is done in three steps, which is shown in Fig. 1.

The aim of this paper is to implement and compare different parallel prefix adders proposed in the literature. The fundamental generate and propagate signals are used to generate the carry input for each adder. Two different operators black and gray are used here. The rest of the paper is organized as follows: In section II, some background information about Parallel Prefix architecture is given. In section III, various parallel prefix adders are discussed. In section IV, experimental results are presented. Conclusions are drawn in section V.

II. PRELIMINARIES

In every bit (*i*) of the two operand block, the two input signals (a_i and b_i) are added to the corresponding carry-in signal ($carry_i$) to produce sum output (sum_i)

The equation to produce the sum output is:

$$sum_i = a_i \oplus b_i \oplus carry_i \qquad \text{…. (1)}$$

Computation of the carry-in signals at every bit is the most critical and time – consuming operation. In the carry- look ahead scheme of adders (CLA), the focus is to design the carry-in signals for an individual bit additions. This is achieved by generating two signals, the generate (g_i) and propagate (p_i) using the equations:

$$g_i = a_i \wedge b_i \qquad \text{…. (2)}$$
$$p_i = a_i \oplus b_i \qquad \text{…. (3)}$$

The carry in signal for any adder block is calculated by using the formula

$$c_{i+1} = g_i \, V \, (p_i \wedge c_i) \qquad \text{…. (4)}$$

Where c_i must be expanded to calculate c_{i+1} at any level of addition.

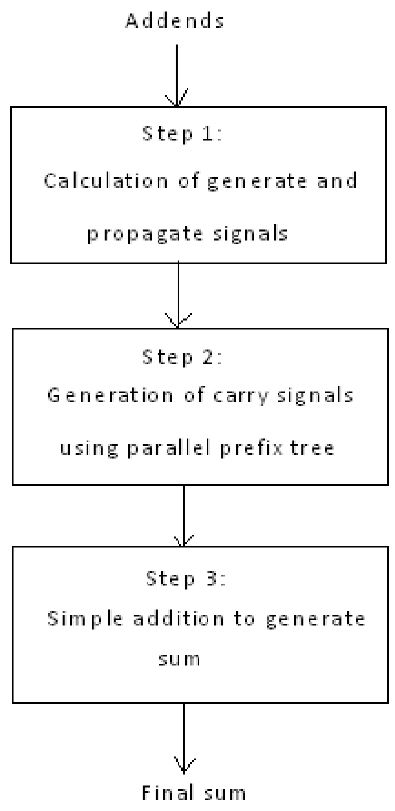
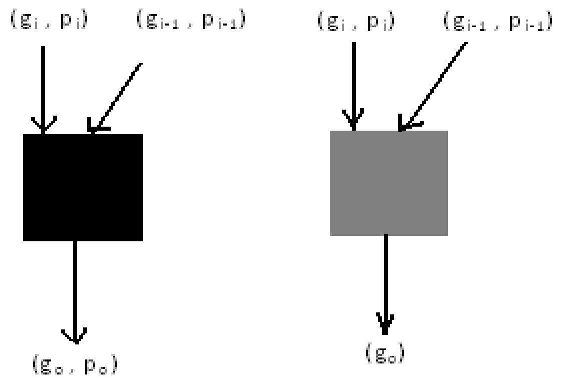

Addends
Step 1:
Calculation of generate and
propagate signals
Step 2:
Generation of carry signals
using parallel prefix tree
Step 3:
Simple addition to generate
sum
Final sum

Fig. 1 Addition procedure using Parallel Prefix tree structures

Parallel Prefix adders compute carry-in at each level of addition by combining generate and propagate signals in a different manner. Two operators namely *black* and *gray* are used in parallel prefix trees are shown in Fig 2(a), 2(b) respectively.



(a) black operator (b) gray operator

Fig. 2 Operators used in Parallel Prefix trees

The black operator receives two sets of generate and propagate signals (g_i , p_i),(g_{i-1} , p_{i-1}), computes one set of generate and propagate signals (g_o , p_o) by the following equations:

$$g_o = g_i \vee (p_i \wedge g_{i-1}) \quad .... (5)$$
$$p_o = p_i \wedge p_{i-1} \quad .... (6)$$

The gray operator receives two sets of generate and propagate signals (g_i, p_i), (g_{i-1} , p_{i-1}), computes only one generate signal with the same equation as in equation (5).

III. PARALLEL PREFIX ADDERS

The following Parallel prefix adders are considered for the implementation:

- Brent-Kung Adder(BK Adder)
- Skalansky Adder(Sk Adder)
- Kogge-Stone Adder(KS Adder)
- Han-Carlson Adder(HC Adder)
- Ladner-Fischer Adder(LF Adder)
- Knowles Adder(Kn Adder)

The architecture of 16-bit BK adder, Skalansky adder, KS adder, HC adder, LF adder and Knowles adder are shown in Fig . (3), (4), (5),(6),(7) and (8) respectively .

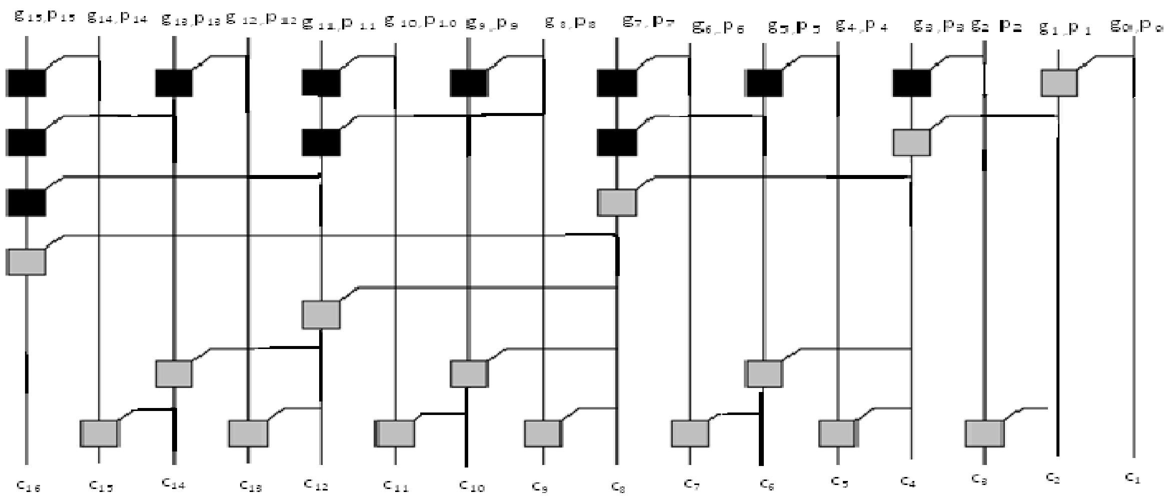


Fig. 3 Brunt-Kung adder

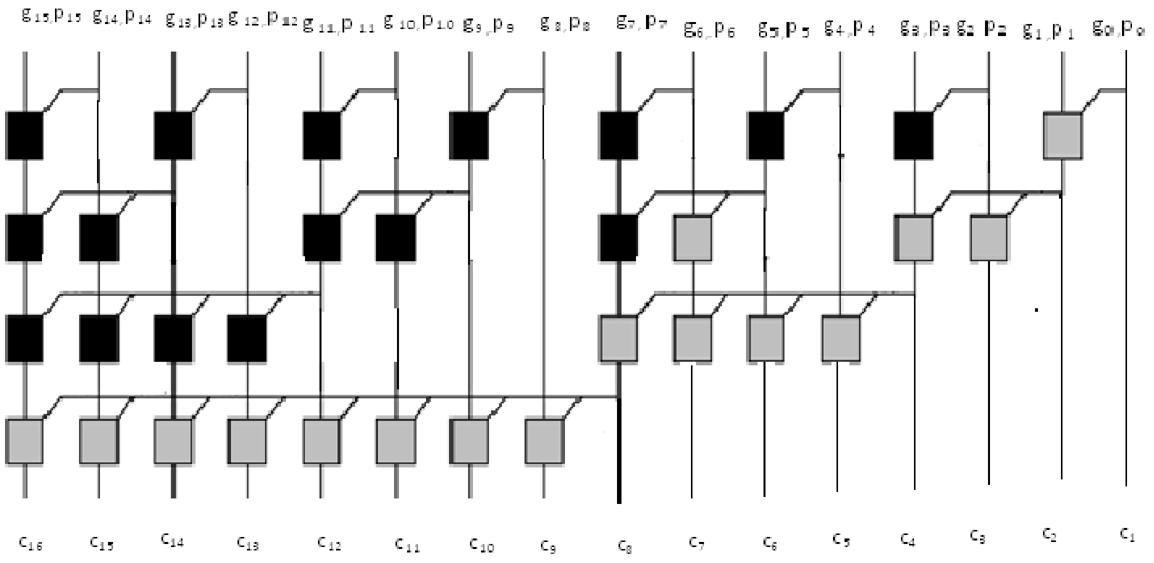


Fig. 4 Skalnsky adder

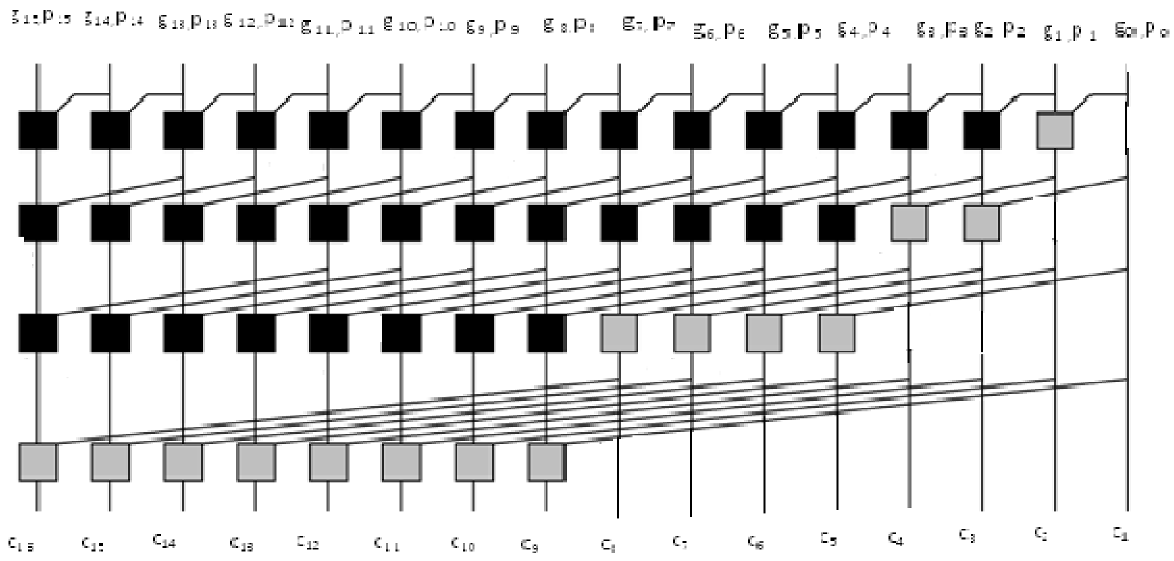

g15,p15 g14,p14 g13,p13 g12,p12 g11,p11 g10,p10 g9,p9 g8,p8 g7,p7 g6,p6 g5,p5 g4,p4 g3,p3 g2,p2 g1,p1 g0,p0
c16 c15 c14 c13 c12 c11 c10 c9 c8 c7 c6 c5 c4 c3 c2 c1

Fig. 5 Kogge-Stone adder

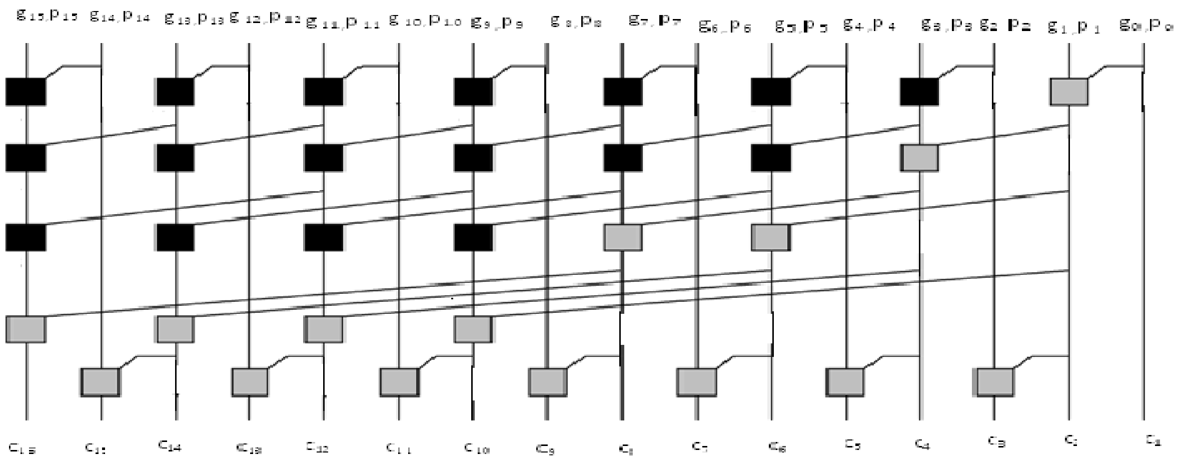

g15,p15 g14,p14 g13,p13 g12,p12 g11,p11 g10,p10 g9,p9 g8,p8 g7,p7 g6,p6 g5,p5 g4,p4 g3,p3 g2,p2 g1,p1 g0,p0
c16 c15 c14 c13 c12 c11 c10 c9 c8 c7 c6 c5 c4 c3 c2 c1

Fig. 6 Han-Carlson adder

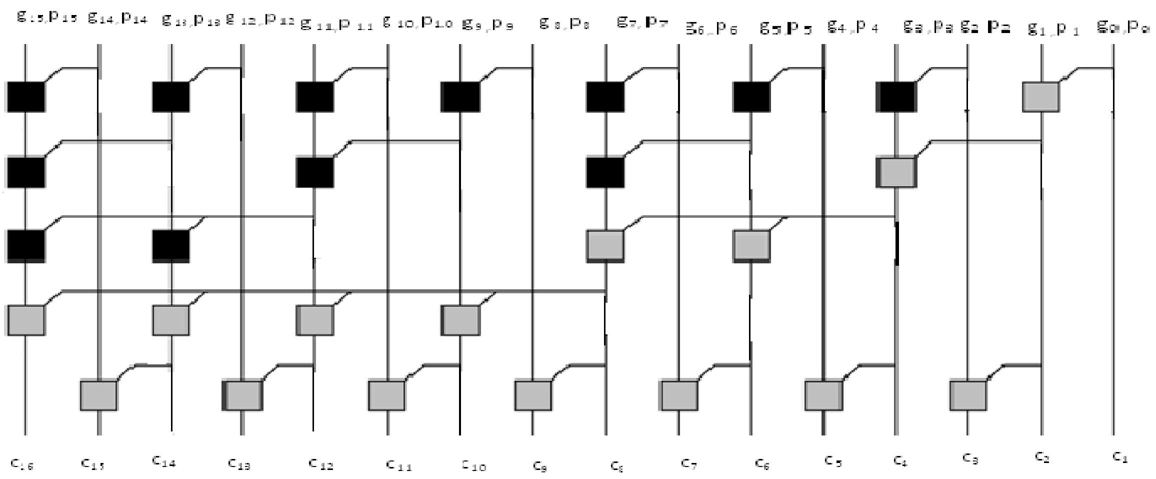

g15,p15 g14,p14 g13,p13 g12,p12 g11,p11 g10,p10 g9,p9 g8,p8 g7,p7 g6,p6 g5,p5 g4,p4 g3,p3 g2,p2 g1,p1 g0,p0
c16 c15 c14 c13 c12 c11 c10 c9 c8 c7 c6 c5 c4 c3 c2 c1

Fig. 7 Ladner-Fisher adder

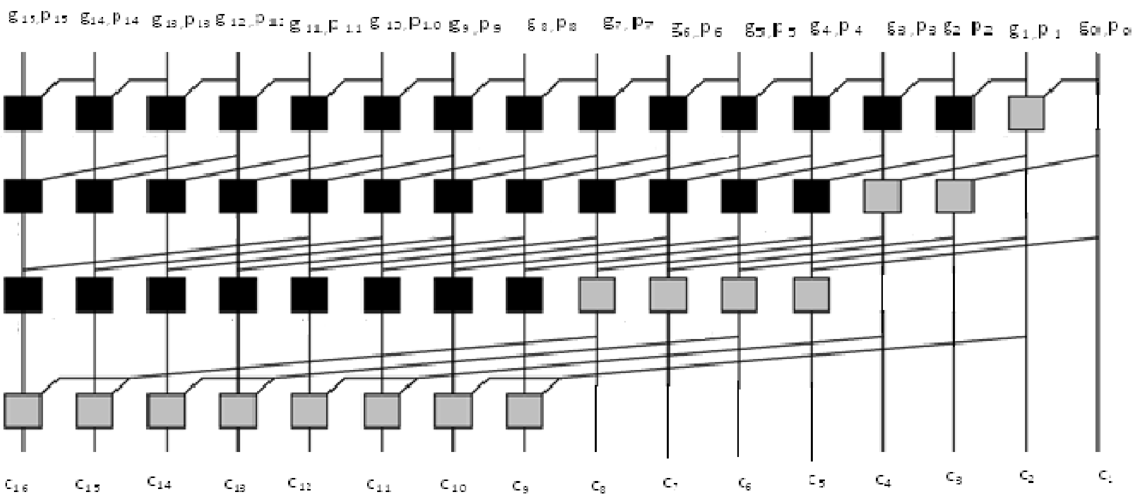


Fig. 8 Knowles adder

IV. EXPERIMENTAL RESULTS

The VHDL code is developed for above said adders and are simulated using Xilinx 9.1 version choosing the device number XC3S500E. The performance of the Parallel Prefix adders which can be evaluated by the Delay(ns), number of Logic Levels of implementation, utilization of Look up tables, Slices and the over all Gate Count on the targeted device is tabulated in Table I.

Table I : Comparison of 16-bit parallel prefix adders

Adder	Delay(nsec)	Logic Levels	Look-up tables	slices	Gate count
BK adder	15.568	12	39	23	264
SK adder	17.548	14	37	22	240
KS adder	15.429	12	61	33	390
HC adder	13.739	11	51	29	321
LF adder	16.644	13	41	23	270
Kn adder	16.477	13	67	36	423

V. CONCLUSIONS

The experimental results shows that the Skalansky adder is slower when compared to other adders. The delay is slightly less in LF adder and Knowles adder but the area occupied by LF adder is almost 50% are occupied by Knowles adder. The BK adder and KS adder has better in speed performance compared to LF and Knowles adder, KS adder occupies more area than BK adder. HC adder is shown as the fastest adder than any adder which is considered here, and does not occupy large area compared to KS adder and Knowles adder.

Finally it is concluded that the HC adder is the fastest and area efficient adder when implemented in an FPGA. The Skalansky adder occupies very less area compared to other adders.

The performance of these adders can be estimated for high bit-widths. This is important for Cryptographic applications, where the addition of more number of bits is necessary. It is also important to make use of efficient adder for high speed arithmetic operations, and in FIR filters etc.

REFERENCES

[1]. Skalansky "conditional sum additions logic" IRE Transactions, Electronic Computers, vol, EC – 9, pp, 226 - 231, June 1960.
[2]. 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.
[3]. Brent R, Kung H, "A regular layout for parallel adders". IEEE Trans, computers, Vol.C-31, no.3, pp 260-264, March1982.
[4]. Ladner R, Fischer M,"Parallel prefix computation ", J.ACM, vol.27, no. 4, pp 831-838, Oct.1980.
[5]. Han T, Carlson D, "Fast area-efficient VLSI adders", Proc.8th.symp.Comp.Arit.pp.49-56, Sep.1987.

[6]. Jianhua LiuZhu, 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.
[7]. Taeko Matsunaga and Yusuka Matsunaga., “Timing-Constrained Area minimization Algorithm for parallel prefix adders”, IEICE TRANS, Fundamentals, vol.E90-A, No.12 Dec, 2007.
[8]. Taeko Matsunaga and Shinji Kimura, Yusuka Matsunaga, “Synthesis of parallel prefix adders considering switching activities”,IEEE International Conference on computer design, 2008, pp.404-409.
[9]. Giorgos Dimitrakopoulos and Dimitric Nikolos, “High Speed Parallel –Prefix VLSI Ling Adders”, IEEE Trans on computers, Vol.54, No.2, Feb 2005
[10]. V.Choi and E.E.Swartz lander, Ir, “Parallel Prefix adder design with matrix representation”,, in Proc.17th IEEE symp, comput.Arithmatic (ARITH), 2005, PP 90-98.
[11]. John F.Wakerly, Digital Design Principles and Practices, 4th Edition, Pearson Education, 2009.