Prof. S.V.Padmajarani, Dr. M.Muralidhar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue4, July-August 2012, pp.1524-1528

A New Approach To Implement Parallel Prefix Adders In An FPGA

Prof. S.V.Padmajarani*, Dr. M.Muralidhar **
*(H.O.D., Department of E.C.E,Jagan's College of Engineering & Technology, Nellore- 524 320, Andhra Pradesh, India
** (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 presents a new approach to redesign the basic operators used in parallel prefix architectures. The number of multiplexers contained in each Slice of an FPGA is considered here for the redesign of the basic operators used in parallel prefix tree. This new design is implemented with 16-bit width operands of various parallel prefix adders on Xilinx Spartan FPGA. The experimental results indicate that the new approach of basic operators make some of the parallel prefix adders architectures faster and area efficient.

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 fig1. 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 aim of this paper is to propose a new approach for the basic operators and make use of these operators in various parallel prefix adders to evaluate their performance with newly redesigned operators

The rest of the paper is organized as follows: In section II, some background information about Parallel Prefix architecture is given. New design approach of basic operators is discussed in section III. Experimental results are presented in section IV. Conclusions are drawn in section V.

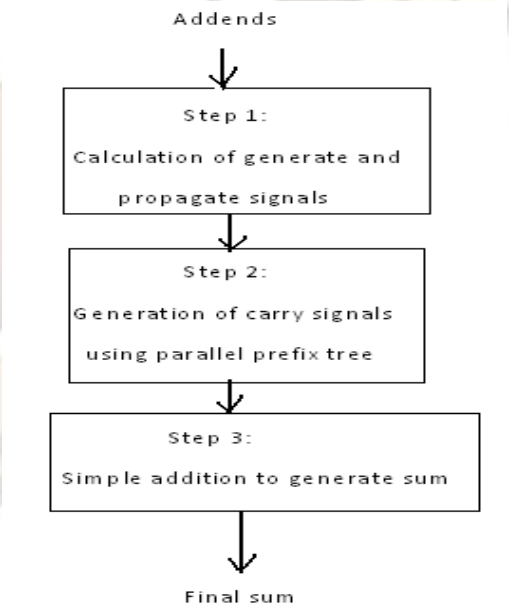


Fig 1. Addition procedure using Parallel Prefix tree structures

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:

$$sum_i = a_i \oplus b_i \oplus carry_i \quad - (1)$$

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:

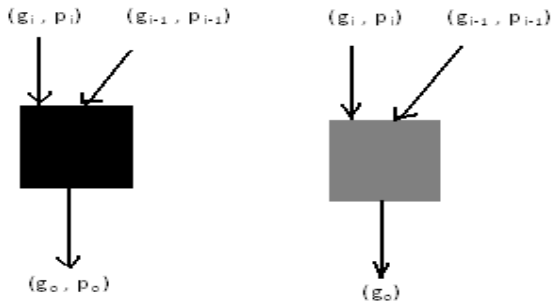
$$g_i = a_i \wedge b_i \quad - (2)$$
$$p_i = a_i \oplus b_i \quad - (3)$$

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

$$c_{i+1} = g_i \vee (p_i) \quad - (4)$$

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

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), fig 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:

$$g_o = g_i \vee (p_i \wedge g_{i-1}) \quad - (4)$$

$$p_o = p_i \wedge p_{i-1} \quad - (5)$$

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 (4).

The logic diagram of black operator and gray operator is shown in fig 3(a), fig 3(b) respectively.

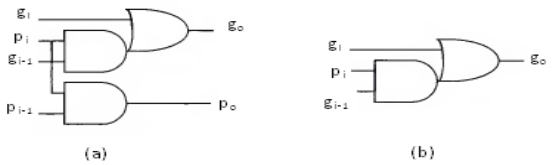


Fig 3. Logic diagram of (a) black operator (b) gray operator

III. NEW APPROACH

An FPGA contains number of Slices, each Slice contains number of multiplexers, Look up tables, logic gates, flip-flops, etc. The black and gray operators are redesigned by using multiplexers in place of gate level design. The redesign for black operator is shown below in fig 4.

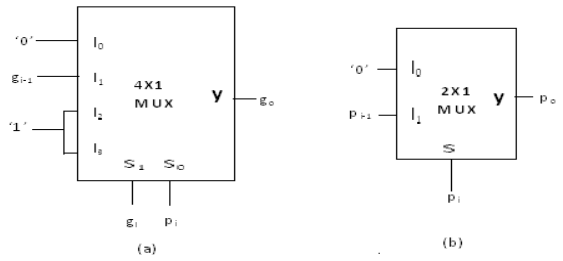


Fig 4. New design approach to design a black operator (a) for calculation of g_0 (b) for calculation of p_0

The black operator computes g_o value by using the inputs g_i , p_i and g_{i-1}, as shown in fig 4(a), the p_o value is computed by using the inputs p_i , p_{i-1} as shown in fig 4(b). The redesign is based on the multiplexer design of a truth table given in table 1.

Table 1 Truth Table of black operator

g_i	p_i	g_o	p_0
0	0	0	0
0	1	g_{i-1}	p_{i-1}
1	0	1	0
1	1	1	p_{i-1}

The table 1 indicates the operation of black operator for possible input combinations. The gray operator computes only g_o by receiving the inputs g_i , p_i and g_{i-1} and it is similar to the computation of g_o

as in black operator, therefore fig4(a) is used to implement gray operator.

In this paper the following Parallel Prefix adders are considered for the implementation with the newly redesigned operators:

- 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 Brent-Kung adder is shown in fig (5).

IV. EXPERIMENTAL RESULTS

The BK adder, Skalansky adder, KS adder, HC adder, LF adder, Knowles adder are designed with old and new approaches of basic operators using VHDL software. These adders are simulated using Xilinx 9.1 version choosing the device number XC3S500E. The following parameters are noted for their performance evaluation:

- Delay (ns)
- Utilization of Look up tables
- Utilization of Slices
- Overall gate count

The results are tabulated in table(2),(3),(4),(5) and (6).

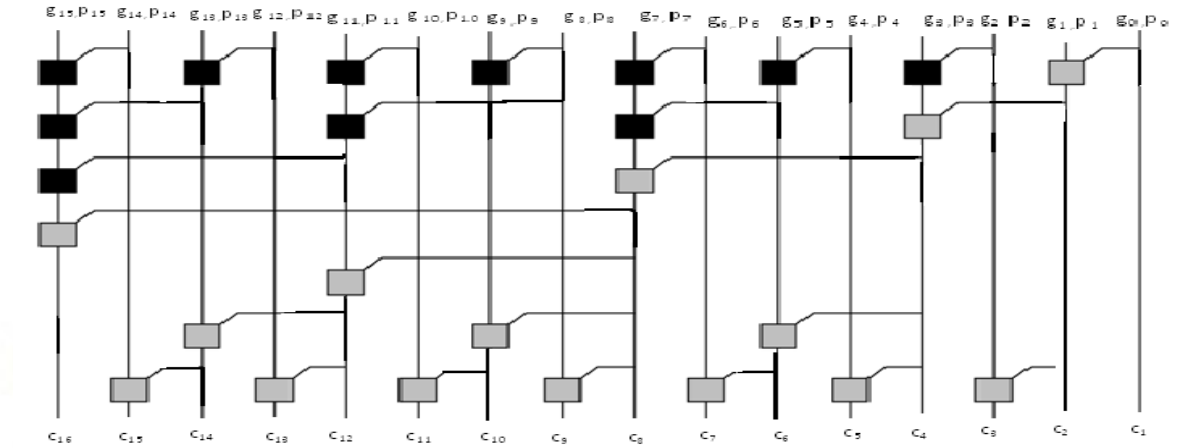


Fig 5. Brunt-Kung adder

Table 2 Delay comparison of old and new methods(in nsec)

Adder/method	BK adder	Skalansky adder	KS adder	HC adder	LF adder	Knowles adder
Old	15.568	17.548	15.429	13.739	16.644	16.477
New	15.357	13.452	12.339	16.793	13.561	12.340

Table 3 Logic Levels comparison of old and new methods

Adder/method	BK adder	Skalansky adder	KS adder	HC adder	LF adder	Knowles adder
Old	12	14	12	11	13	13
New	12	10	9	13	10	9

Table 4 Look up table utilization comparison of old and new methods

Adder/method	BK adder	Skalansky adder	KS adder	HC adder	LF adder	Knowles adder
Old	39	37	61	51	41	67
New	47	50	82	51	48	80

Table 5 Slices utilization comparison of old and new methods

Adder/method	BK adder	Skalansky adder	KS adder	HC adder	LF adder	Knowles adder
Old	23	22	33	29	23	36
New	26	27	45	28	26	42

Table 6 Over all gate count comparison of old and new methods

Adder/method	BK adder	Skalansky adder	KS adder	HC adder	LF adder	Knowles adder
Old	264	240	390	321	270	423
New	297	315	501	324	309	486

V CONCLUSIONS

This paper presents a new approach for the basic operators of parallel prefix tree adders. In Skalansky, KS,LF, Knowles adders the delay is reduced by this new approach, in BK adder there is no much difference with this new approach and in the case of HC adder the delay is increased. The same can be understood with reference to number of logic levels of implementation, as the logic levels are more delay increases.

The area requirement can be considered from the utilization of LUTs, Slices and over all gate count. The BK adder occupies less area compared to other adders, but does not show much difference with new approach. Skalansky, LF adders occupies slightly more area in new approach compared to old method. KS and Knowles adders occupies more area in new approach. HC adder shows almost no difference with new approach.

The HC adder is the fast adder but occupies large area in old approach. The new approach makes Skalansky adder faster and occupies less area than HC adder. The KS and Knowles adders are proven even faster than Skalansky adder but they occupy large area compared to Skalansky adder. Therefore the Skalansky adder is resulted as the best adder in terms of speed and area characteristics with new approach. Finally it can be concluded that the new approach for the redesign of basic operators will speed up the addition process of parallel prefix addition with some area overhead.

The performance of these adders can be estimated for high bit-widths. This can be further used in Cryptographic applications, where the addition of more number of bits is necessary. The new approach for the parallel prefix adders can also be used to speed up the addition process in FIR filter and arithmetic operations like multipliers, 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, , pp.404-409*, 2008

[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), PP 90-98*,2005

[11] John F.Wakerly, *Digital Design Principles and Practices*, 4th Edition, Pearson Education, 2009.