ISSN 2319-6629

Volume 1, No.1, August- September 2012

International Journal of Wireless Communications and Networking Technologies

Available Online at http://warse.org/pdfs/ijwcnt02112012.pdf

High speed VLSI implementation of 256-bit Parallel Prefix Adders

[1]Srinivasasamanoj.R, [2]M. Sri Hari, [3] B. Ratna Raju

[1]PG Student, Kakinada Institute Of Engineering And Technology, Kakinada, A.P., India. pjrece@gmail.com

[2]Assistant Professor, Kakinada Institute Of Engineering And Technology, Kakinada, Andhra Pradesh, India

[3]Associate Professor, Kakinada Institute Of Engineering And Technology, Kakinada, 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. Numbers 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 128-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 adder's architectures faster and area efficient.

Keywords : Parallel Prefix Adder, Optimize Area, Fan-out, Logic Depth, Inter Connect Count.

1. INTRODUCTION

Addition is a timing critical operation in today's floating point units. In order to develop faster processing, an end-around carry (EAC) was proposed as a part of fused-multiply-add unit which performs multiplication followed by addition [5]. The proposed EAC adder was also investigated through other pre fix adders in FPGA technology as a complete adder [6]. In this thesis, we propose a 128-bit standalone adder with parallel prefix end around carry logic and conditional sum blocks to improve the critical path delay and provide flexibility to design with different adder architectures. In previous works, CLA logic was used for EAC logic. Using a modified structure of a parallel prefix 2n + 1 adder provides flexibility to the design and decreases the length of the carry path. After the architecture is tested and verified, critical path is analyzed using FreePDK45nm library. Full custom design techniques are applied carefully during critical path optimization. Critical path analysis provides fast comparison of the total delay among different architectures without designing the whole circuit and a simpler approach to size the transistors for lowest delay possible. As a final step, data path is designed as a recurring bit slice for fast layout entry. The results show that the proposed adder shows 142ps delay, 2.42mW average power dissipation, and 3,132 sq. micron area assuming there is not much routing area overhead in the estimated area.

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. 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], this 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 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 1, some background information about Parallel Prefix architecture is given. New design approach of basic operators is discussed in section 2. carry-tree adder designs discussed in section 3. Related work of prefix parallel

adders discussed in section 4. Experimental results are presented in section 5. Conclusions are drawn in section 6.

Several papers have attacked the problem of designing efficient diminished adders. The majority of them rely on the use of an inverted end around carry (IEAC) n-bit adder, which is an adder that accepts two n-bit operands and provides a sum increased by one compared to their integer sum if their integer addition does not result in a carry output. Although an IEAC adder can be implemented by using an integer adder in which its carry output is connected back to its carry input via an inverter, such a direct feedback is not a good solution. Since the carry output depends on the carry input, a direct connection between them forms a combinational loop that may lead to an unwanted race condition . To this end, a number of custom solutions have been proposed for the design of efficient IEAC adders. Considering the diminished-1 representation for modulo 2n þ 1 addition, [4], [5] used an IEAC adder which is based on an integer adder along with an extra carry look ahead (CLA) unit. The CLA unit computes the carry output which is then inverted used as the carry input of the integer adder. Solutions that rely on a single carry computation unit have also been proposed.

Although these architectures are faster than the carry look ahead ones proposed in [12], for sufficiently wide operands, they are slower than the corresponding parallel prefix integer adders because of the need for the extra prefix level. In [12], it has been shown that the recirculation of the inverted end around carry can be performed within the existing prefix levels, that is, in parallel with the carries' computation. In this way, the need of the extra prefix level is canceled and parallel-prefix IEAC adders are derived that can operate as fast as their integer counterparts, that is, they offer a logic depth of log2 n prefix levels. Unfortunately, this level of performance requires significantly more area than the solutions of [11], [12], since a double parallel-prefix computation tree is required in several levels of the carry computation unit. For reducing the area complexity of the parallel-prefix solutions, select-prefix and circular carry select IEAC adders have been proposed. Unfortunately, both these proposals achieve a smaller operating speed than the parallel-prefix ones of [12].

2. PARALLEL-PREFIX ADDITION BASICS

The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor data path units. As such, extensive research continues to be focused on improving the power delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. Reconfigurable logic such as Field Programmable Gate Arrays (FPGAs) has been gaining in popularity in recent years because it offers improved performance in terms of speed and power over DSP-based and microprocessor-based solutions for many practical designs involving mobile DSP and telecommunications applications and a significant reduction in development time and cost over Application Specific Integrated Circuit (ASIC) designs. The power advantage is especially important with the growing popularity of mobile and portable electronics, which make extensive use of DSP functions. However, because of the structure of the configurable logic and routing resources in FPGAs, parallel-prefix adders will have a different performance than VLSI implementations. In particular, most modern FPGAs employ a fast-carry chain which optimizes the carry path for the simple Ripple Carry Adder (RCA).
In this paper, the practical issues involved in designing and implementing tree-based adders on FPGAs are. An efficient testing strategy for evaluating the performance of these adders is discussed. Several tree-based adder structures are implemented and characterized on a FPGA and compared with the Ripple Carry Adder (RCA) and the Carry Skip Adder (CSA). Finally, some conclusions and suggestions for improving FPGA designs to enable better tree-based adder performance are given.

3. CARRY-TREE ADDER DESIGNS

Parallel-prefix adders, also known as carry-tree adders, pre-compute the propagate and generate signals [1]. These signals are variously combined using the fundamental carry operator (fco) [2].
(gL, pL) o (gR, pR) = (gL + pL•gR, pL • pR) (1)
Due to associative property of the fco, these operators can be combined in different ways to form various adder structures. For, example the four-bit carry-lookahead generator is given by:
c4 = (g4, p4) o [(g3, p3) o [(g2, p2) o (g1, p1)]] (2)

A simple rearrangement of the order of operations allows parallel operation, resulting in a more efficient tree structure for this four bit example:
c4 = [(g4, p4) o (g3, p3)] o [(g2, p2) o (g1, p1)] (3)

It is readily apparent that a key advantage of the tree structured adder is that the critical path due to the carry delay is on the order of log2N for an N-bit wide adder. The arrangement of the prefix network gives rise to various families of adders. For a discussion of the various carry-tree structures, see [1, 3]. For this study, the focus is on the Kogge-Stone adder [4], known for having minimal logic depth and fan out (see Fig 1(a)). Here we designate BC as the black cell which generates the ordered pair in equation (1); the gray cell (GC) generates the left signal only, following [1]. The interconnect area is known to be high, but for an FPGA with large routing overhead to begin with, this is not as important as in a VLSI implementation. The regularity of the Kogge-Stone prefix network has built in redundancy which has implications for fault-tolerant designs [5]. The sparse Kogge-Stone adder, shown in Fig 1(b), is also studied. This

hybrid design completes the summation process with a 4 bit RCA allowing the carry prefix network to be simplified.

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder is also examined [6]. Like the sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders. Also of interest for the spanning-tree CLA is its testability features [7].

4. RELATED WORK

Xing and Yu noted that delay models and cost analysis for adder designs developed for VLSI technology do not map directly to FPGA designs [8]. They compared the design of the ripple carry adder with the carry-lookahead, carry-skip, and carry-select adders on the Xilinx 4000 series FPGAs. Only an optimized form of the carry-skip adder performed better than the ripple carry adder when the adder operands were above 56 bits.

A study of adders implemented on the Xilinx Virtex II yielded similar results [9]. In [10], the authors considered several parallel prefix adders implemented on a Xilinx Virtex 5 FPGA. It is found that the simple RCA adder is superior to the parallel prefix designs because the RCA can take advantage of the fast carry chain on the FPGA.

Kogge-Stone The Kogge-Stone tree [22] Figures 1- 5 achieves both log2N stages and fan-out of 2 at each stage. This comes at the cost of long wires that must be routed between stages. The tree also contains more PG cells; while this may not impact the area if the adder layout is on a regular grid, it will increase power consumption. Despite these cost, Kogge-Stone adder is generally used for wide adders because it shows the lowest delay among other structures.

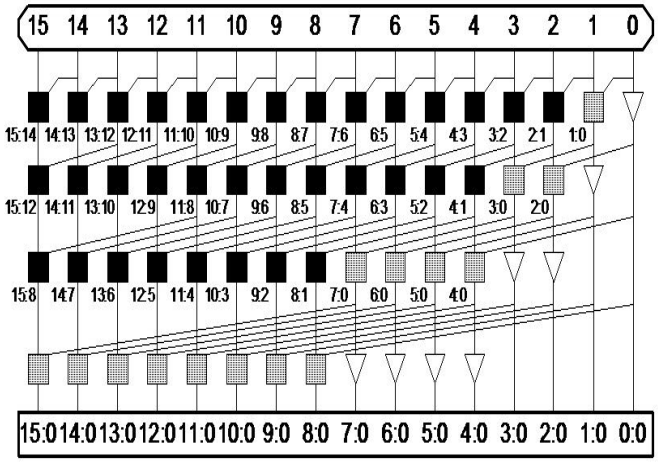

Figure 1: The Parallel Prefix addition

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder is also examined [6]. Like the sparse Kogge-Stone adder, this design terminates with a 4-bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders. Also of interest for the spanning-tree CLA is its testability features [7].

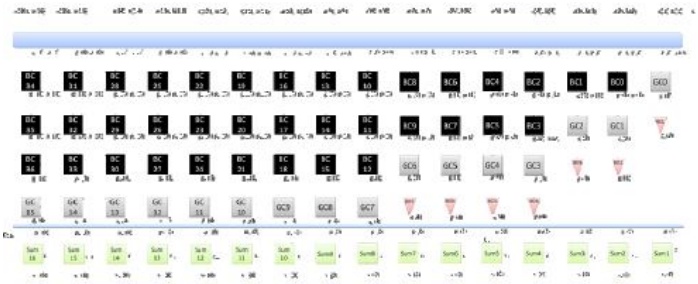

Figure 2: 128-bit Kogge-Stone adder

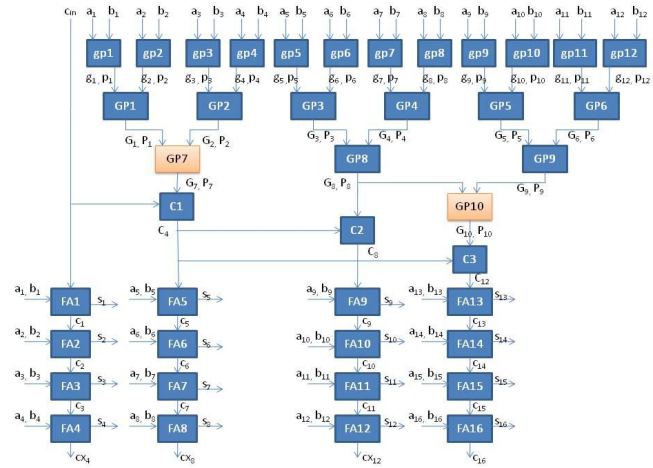

Figure 3: Spanning Tree Carry Lookahead Adder (16 bit)

This study focuses on carry-tree adders implemented on a Xilinx Spartan 3E FPGA. The distinctive contributions of this paper are two-fold. First, we consider tree-based adders and a hybrid form which combines a tree structure with a ripple-carry design. The Kogge-Stone adder is chosen as a representative of the former type and the sparse Kogge Stone and spanning tree adder are representative of the latter category. Second, this paper considers the practical issues involved in testing the adders and provides actual measurement data to compare with simulation results.

The previous works cited above all rely upon the synthesis reports from the FPGA place and route software for their results. In addition to being able to compare the simulation data with measured data using a high-speed logic analyzer, our results present a different perspective in terms of both results and types of adders as those presented in [8-10]

The ripple carry adder with the carry-lookahead, carry-skip, and carry-select adders on the Xilinx 4000 series FPGAs. Only an optimized form of the carry-skip adder performed better than the ripple carry adder when the adder operands

were above 56 bits. A study of adders implemented on the Xilinx Virtex II yielded similar results [9]. In [10], the authors considered several parallel prefix adders implemented on a Xilinx Virtex 5 FPGA. It is found that the simple RCA adder is superior to the parallel prefix designs because the RCA can take advantage of the fast carry chain on the FPGA.

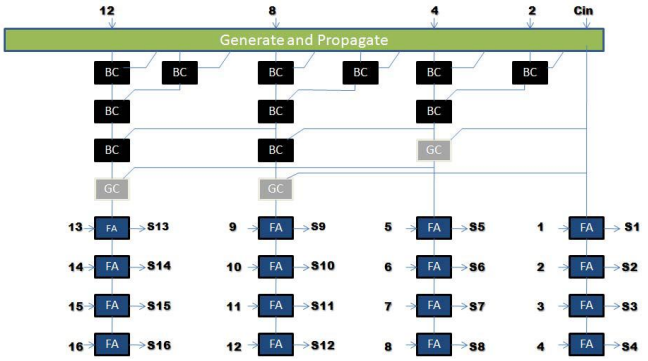

Figure 4: Sparse 128-bit Kogge-Stone adder

This study focuses on carry-tree adders implemented on a Xilinx Spartan 3E FPGA. The distinctive contributions of this paper are two-fold. First, we consider tree-based adders and a hybrid form which combines a tree structure with a ripple-carry design. The Kogge-Stone adder is chosen as a representative of the former type and the sparse Kogge Stone and spanning tree adder are representative of the latter category. Second, this paper considers the practical issues involved in testing the adders and provides actual measurement data to compare with simulation results.

The previous works cited above all rely upon the synthesis reports from the FPGA place and route software for their results. In addition to being able to compare the simulation data with measured data using a high-speed logic analyzer, our results present a different perspective in terms of both results and types of adders as those presented in [8-10].

5. RESULTS

Overall, when the delay due to routing overhead is removed, the tree adders are now closer to the simple RCA design. The RCA adder exhibits the best delay with widths up to 128 bits when the routing delay is excluded and out to 256 bits with the routing delay included. Figures depict the measured results using the TLA. A comparison between the tree adders and the RCA is given in Figure 7. The basic trends are the same: the tree adders exhibit logarithmic delay dependence on bit widths and the RCA has linear performance. An RCA as large as256 bits wide was synthesizable on the FPGA, while a Kogge-Stone adder up to 256 bits wide was implemented.

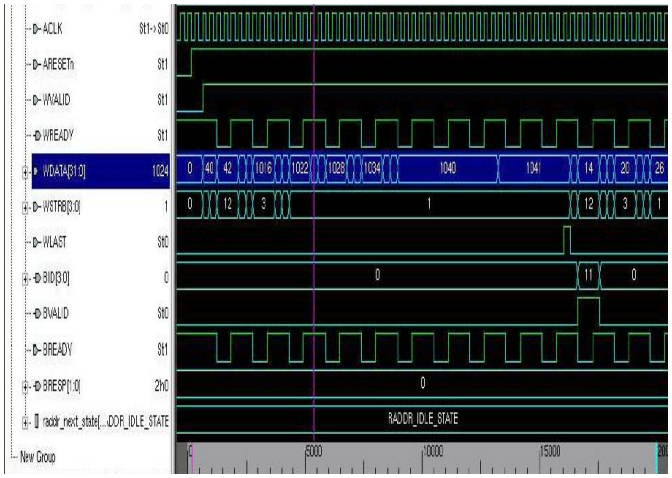

Figure 5: Carry Lookahead Adder 128-bit simlation

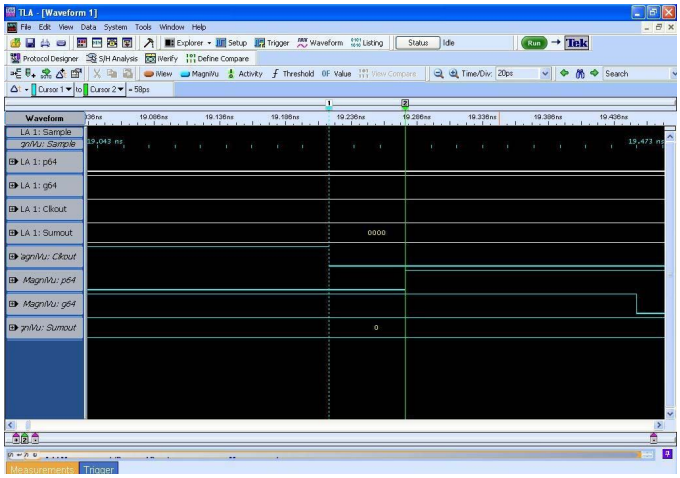

Figure 6: Screen shot of a delay measurement for a128-bitadder using MagniVu timing (blue traces) on theTLA 7012.

The carry-skip adders are compared with the Kogge-Stoneadders and the RCA in Figure 8. Carry skip adders with skip of four and eight were implemented.

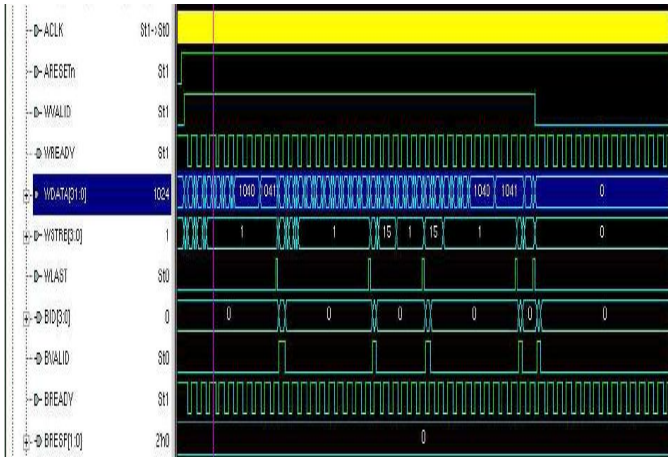

Figure 7: Sparse 128-bit Kogge-Stone adder simulation results

The poor performance of the carry skip adders is attributable to the significant routing overhead incurred by this structure.

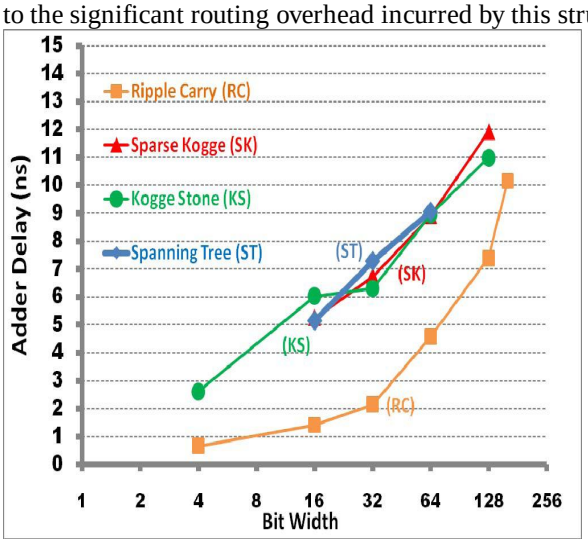

Figure 8: Measured results for the parallel-prefix adder designs compared with the RCA

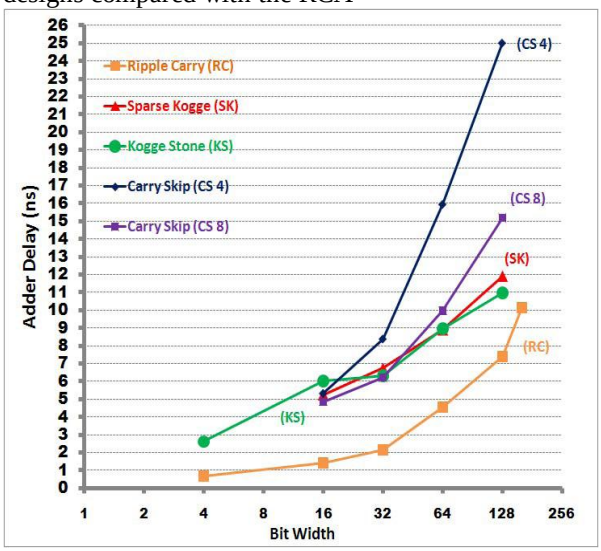

Figure 9: Measured results for the carry-skip adders compared to the RCA and Kogge-Stone adders

Table 1: Delay Results for the Kogge-Stone Adders

N	**Synth. Predict**	**Route Delay**	**Route Fitted**	**Delay *tKS***	**Delay *tRCA***
4	4.343	1.895	1.852	4.300	1.817
16	6.113	2.441	2.614	6.286	2.429
32	7.607	3.323	3.154	7.438	3.245
64	8.771	3.875	3.800	8.696	4.877
128	10.038	4.530	4.552	10.060	8.141
256	11.458	5.404	5.410	11.530	14.669

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.

6. CONCLUSION

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.

REFERENCES

1. Skalansky. **Conditional sum additions logic**, IRE Transactions, Electronic Computers, vol, EC – 9, pp, 226 - 231, June 1960.

2. Kogge P and 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 and Kung H. **A regular layout for parallel adders**. IEEE Trans, computers, Vol.C-31, No.3, pp 260-264, March1982.

4. Koren. **Computer arithmetic algorithms,** A K Peters, Ltd., 2002.

5 R. P. Brent and H. T. Kung. **A regular layout for parallel adders**, IEEE Trans. Computers, Vol. 31, No.3, pp. 260–264, March 1982.

6. P. M. Kogge and H. S. Stone. **A parallel algorithm for the efficient solution of a general class of recurrence equations**, IEEE Trans. Computers, Vol. 22, No. 8, pp. 786–793, August 1973.

7. J. Sklansky. **Conditional sum addition logic**, IRE Trans. Electron. Compute, Vol. 9, No. 6, pp. 226–231, 1960.

8. J. Liu, S. Zhou, H. Zhu, and C.-K. Cheng. **An algorithmic approach for generic parallel adders**, in ICCAD, November 2003, pp. 734–730.

9. R. Zimmermann. **Non-heuristic optimization and synthesis of parallel-prefix adders**, International Workshop on Logic and Architecture Synthesis, December 1996, pp. 123–132.

10. T. Matsunaga and Y. Matsunaga. **Timing-constrained area minimization algorithm for parallel prefix adders**, IEICE Transactions on Fundamentals, Vol. E90-A, No. 12, pp. 2770–2777, December 2007.

11. 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.

12. R. Bryant. **Graph-based algorithms for boolean function manipulation**, IEEE Trans. Computers, Vol. 35, No. 8, pp. 677–691, August 1986.

13. M. Pedram. **Power minimization in ic design: Principles and applications**, ACM Trans. on Design Automation of Electronic Systems, Vol. 1, No. 1, pp. 3–5, January 1996.