Design And Verification Of Low Power And Area Efficient Kogge-Stone Carry Select Adder

Mohammed Haseena Begum (M.Tech),
Nimra College Of Engineering & Technology,
Jupudi, Ibrahimpatnam (Mandal),
Vijayawada,
Krishna (Dist)-521456

V. Vamsi Mohana Krishna M.Tech,(Ph.D)
Associate Professor,
Nimra College Of Engineering & Technology,
Jupudi, Ibrahimpatnam (Mandal),
Vijayawada,Krishna (Dist)-521456

Abstract:

Adders are one of the critical elements in VLSI chips because of their variety of usages such as ALUs, floating point arithmetic units, memory addressing and program counting. For this, prefix adders are based on parallel prefix circuit theory which provides a solid theoretical basis for wide range of design trade-offs between delay, area and wiring complexity. In this paper, we propose a high speed Carry Select Adder by replacing Ripple Carry Adders with parallel prefix adders. Adders are the basic building blocks in digital integrated circuit based designs. Ripple Carry Adders (RCA) are usually preferred for addition of two multi-bit numbers as these RCAs offer fast design time among all types of adders. However RCAs are slowest adders as every full adder must wait till the carry is generated from previous full adder. On the other hand, Carry Look Ahead (CLA) adders are faster adders, but they required more area. The Carry Select Adder is a compromise on between the RCA and CLA in term of area and delay. CSLA is designed by using dual RCA: due to this arrangement the area and delay are still concerned factors. It is clear that there is a scope for reducing delay in such an arrangement. In this research, we have implemented CSLA with prefix adders. Prefix adders are tree structure based and are preferred to speed up the binary additions. This work estimates the performance of proposed design in terms of Logic and route delay. The experimental results show that the performance of CSLA with parallel prefix adder is faster and area efficient compared to conventional modified CSLA.

Index Terms-Prefix Adders, CSLA, Delay Carry Operator, Area Efficient

1. Introduction

In Digital Computer design adder is an important component and it is used in multiple blocks of its architecture. In many computers and in various classes of processor specializations adders are not only used in ALU but also used to calculate addresses and table indices. There exist multiple algorithms to carry on addition operation ranging from simple ripple carry adders to complex CLA.The basic operations involved in any Digital Signal Processing systems are Multiplication, Addition and Accumulation. Addition is an indispensible operation in any Digital, DSP or control system. Therefore fast and accurate operation of digital system depends on the performance of adders [6]. Hence improving the performance of adder is the main area of research in VLSI system design. Over the last decade many different adder architectures were studied and proposed to speed up the binary additions. The details of Ripple Carry Adder and Carry Select Adder are discussed in section II, and the implementation of proposed system is described in section III. The performance and simulation results were presented and discussed in section IV.

2. Carry Select Adder

In electronics, a **carry-select adder** is a particular way to implement an adder. Which is a logic element that computes the (n+1) bit sum of two n-bit numbers? The carry-select adder is simple but rather fast, having a gate level depth of $O(\sqrt{n})$.The carry-select adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit

numbers with a carry-select adder is done with two adders (therefore two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one. After the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known.

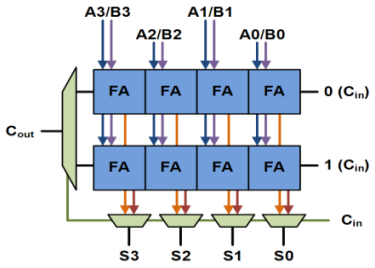

Fig2.1: Basic Building Blocks of Carry Select Adder

The number of bits in each carry select block can be uniform, or variable. In the uniform case, the optimal delay occurs for a block size of (√n). When variable, the block size should have a delay, from addition inputs A and B to the carry out, equal to that of the multiplexer chain leading into it, so that the carry out is calculated just in time. The O(√n) delay is derived from uniform sizing, where the ideal number of full-adder elements per block is equal to the square root of the number of bits being added, since that will yield an equal number of MUX delays.

2.1 Ripple Carry Adder:

The Ripple Carry Adder is used to compute addition of two N-bit numbers. It consists of N full adders to add N-bit numbers. From the second full adder, carry input of every full adder is the carry output of its previous full adder. This kind of adder is typically known as Ripple Carry Adder because carry ripples to next full adder. The layout [12] of Ripple Carry Adder is simple, which allows fast design time. The Ripple Carry Adder [9] is slowest among all the adders because every full adder must wait till the previous full adder generates the carry bit for its input. The 3-bit RCA is shown in Figure2.2. Theoretically the Ripple Carry Adder has delay of *o*(n) and area of *o*(n).

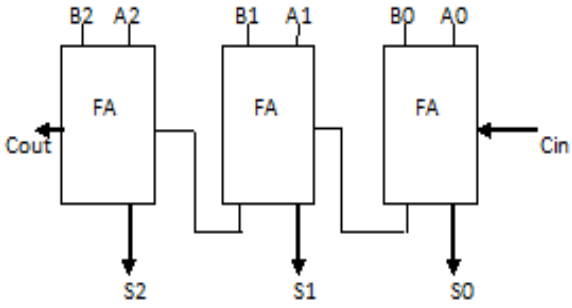

Figure 2.2: 3-bit Ripple Carry Adder

2.2. Multiplexer:

Multiplexer is a combinational circuit which has many inputs and single output. Depending on the select input combination the content on one of the selected input line is transferred on to the output line. It is also widely known as many to one circuit, data selector and universal parallel to serial converter. The 6:3 Multiplexer is shown in Figure 2.3.

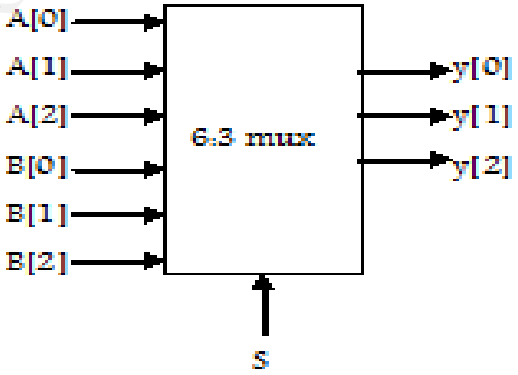

Figure 2.3:6:3 multiplexer

4. Fused Multiply-Add Operation

A fused multiply-add(FMA) unit performs the multiplication A B followed immediately by an addition of the product and a third operand C so that the result T is calculated as Eqn. 1.1 in a single indivisible step [2]. Such a unit is capable of performing multiply only by setting C = 0 and add(or subtract) only by setting, for example, B=1. T = A *B + C = M + C (1.1) An advantage of a fused multiply add unit, compared to separate multiply and adder, arises when executing floating-point operations since

rounding is performed only once for the result of T = A*B+C rather than twice(for the multiply and then for the add) [13]. Since rounding may introduce computation errors, reducing the number of rounding operations a effects positively the overall error.

The input of the operands is calculated at the CSA (Carry-Save Adder) multiplier tree and the magnitude of the operands is not known prior to addition to determine which operand has greater value. Since floating point is a sign magnitude operation, the result of the adder should be in two's complement form [14]. Therefore, an adder is needed to produce two separate results

5. Parallel Prefix Adders

A parallel prefix circuit is a combinational circuit with n inputs x1, x2,xn producing the outputs x1,x20x1, :::; xn, xn1:::x1 where is the associatively binary operation. The first stage of the adder generates individual P and G signals. The remaining stages constitute the parallel prefix circuit with the fundamental carry operation serving as the associative binary operation. This part of the adder can be designed in many different ways. To describe parallel-prefix adders, first of all the parallel-prefix or dot operator needs to be defined:

(GI, PI'). (GX, PX') = (GI + PI'GX, PI'PX')

Here the Ps and Gs can be either single or group propagate and generate signals. The Ps and Gs with index X are from a lower significance level than those with index.

Although computing carry-propagate addition can use generate and propagate signals, its implementation in VLSI can be quite inefficient due to the number of wires that have to be connected together. Parallel-prefix adders solve this problem by making the wires shorter with simple gate structures to aid in the passing of groups of carries to the next weight.

The parallel prefix adders are more flexible and are used to speed up the binary additions. Parallel prefix adders are obtained from Carry Look Ahead (CLA) structure. We use tree structure form to increase the speed of arithmetic operation. Parallel prefix adders are fastest adders and these are used for high performance arithmetic circuits in industries. The construction of parallel prefix adder [5] involves three stages

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

5.1. Pre-possessing stage

In this stage we compute, generate and propagate signals to each pair of inputs A and B. These signals are given by the logic equations 1&2:

$P_i = A_i$ xor B_i..... (1)

$G_i = A_i$ and B_i..... (2)

5.2. Carry generation network:

In this stage we compute carries corresponding to each bit. Execution of these operations is carried out in parallel [5]. After the computation of carries in parallel they are segmented into smaller pieces. It uses carry propagate and generate as intermediate signals which are given by the logic equations 3&4:

CPi:j = Pi:k+1 and Pk:j.....(3)

CGi:j = Gi:k+1 or (Pi:k+1 and Gk:j).....(4)

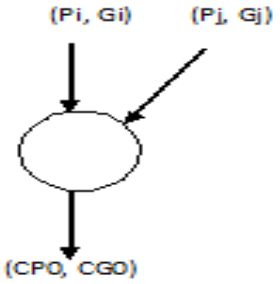

Figure 5.1: Carry operator

The operations involved in this figure are given as:

CP0 = Pi and Pj..... (5)

CG0 = (Pi and Gj) or Gi..... (6)

5.3. Post processing:

This is the final step to compute the summation of input bits. It is common for all adders and the sum bits are computed by logic equation 7&8:

Ci-1 = (Pi and Cin) or Gi..... (7)

Si = Pi xor Ci-1..... (8)

Here the Ps and Gs can be either single or group propagate and generate signals. The Ps and Gs with index X are from a lower significance level than those with index I. In state-of-the-art designs conventional domino and static CMOS logic are used to implement the dot-operator cells. Alternative and more efficient implementations of the dot-operator cells are presented in Paper 6 where a dynamic pass-transistor logic style is employed. The new cells dissipate less power and use fewer transistors than their corresponding domino implementations and the speed penalty is small to none.

Fewer and smaller transistors means smaller area and thus the power-dissipating and performance-killing interconnect capacitances (described later in this chapter) of tree adders can be reduced. If NMOS transistors are used as precharge transistors, the proposed cells have a delay significantly lower than a corresponding domino implementation at the cost of a less robust design. In a "real" application keepers also have to be introduced which would remove some of the speed gained from the design, but this is also true for domino circuits. If dot operators are located in a tree in such a way that an output carry from the tree has been affected by all levels of lower significance, a parallel-prefix adder is obtained. Some examples of the most common tree structures are presented in following figures. They are the Brent-Kung adder, the Kogge-Stone adder, the Ladner-Fischer adder, and the Han-Carlson adder.

6. Kogge-Stone Adder

Kogge-Stone the Kogge-Stone tree 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.

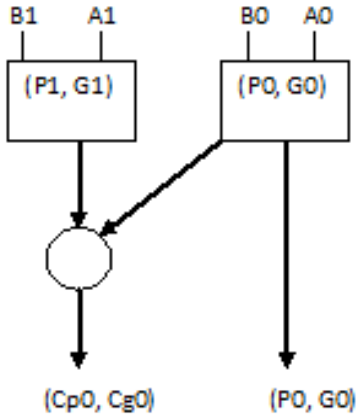

Figure 6.1: 2-bit Kogge-Stone Adder.

Kogge-stone adder is a parallel prefix form of Carry Look-ahead Adder. Kogge-Stone adder can be represented as a parallel prefix graph consisting of carry operator nodes. The time required to generate carry signals in this prefix adder is *o*(log n). It is the fastest adder with focus on design time and is the common choice for high performance adders in industry. The Kogge-Stone adder concept was developed by Peter M. Kogge and Harold S. Stone which was published in 1973. The better performance of Kogge-Stone adder is because of its minimum logic depth and bounded fan-out. On the other side it occupies large silicon area. The construction of 2, 3, 4, 5-bit Kogge-Stone adder are shown in 5.2&5.3 Figures.

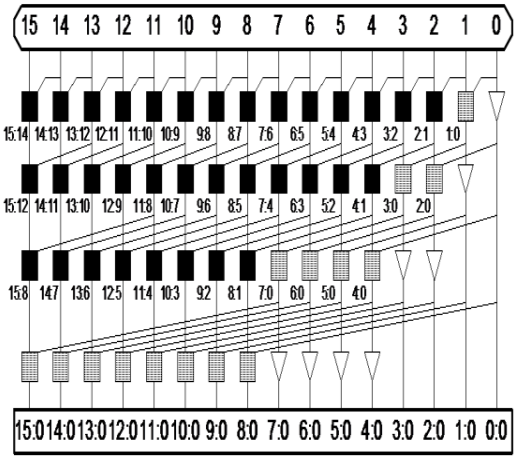

Figure 6.2: Structure of Kogge-Stone adder

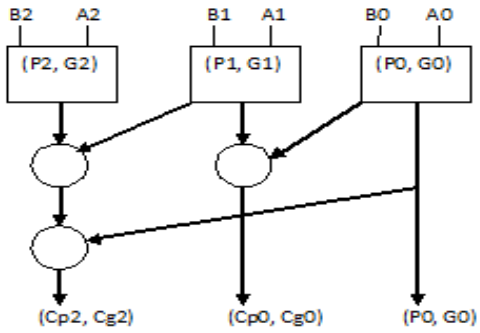

Figure 6.3: 3-bit Kogge-Stone Adder

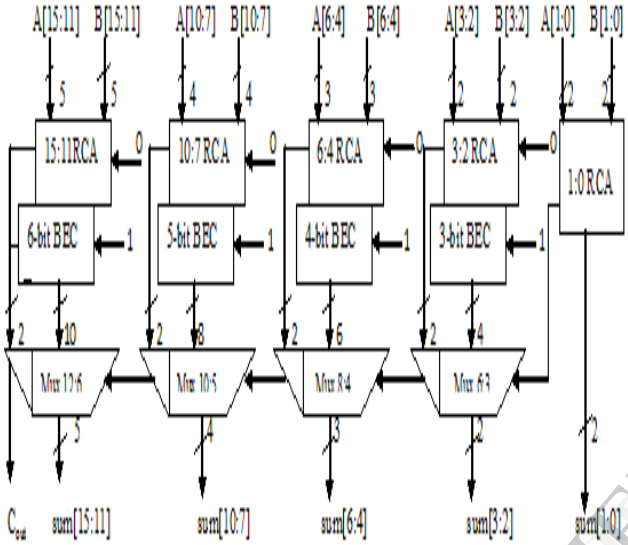

Figure 6.4: Modified 16-b KS CSLA.

7. Simulation and Comparisons

Adder type	No.of bits	Logic delay (ns)	Route delay (ns)	Total delay (ns)
Ripple carry Adder	2 bit 5 bit	5.499 7.335	1.338 2.685	6.837 10.020
Kogge-Stone Adder	2 bit 5 bit	5.165 6.723	0.96 2.331	6.125 9.034

Table 6.1: Comparison between Ripple carry and Kogge-stone adders.

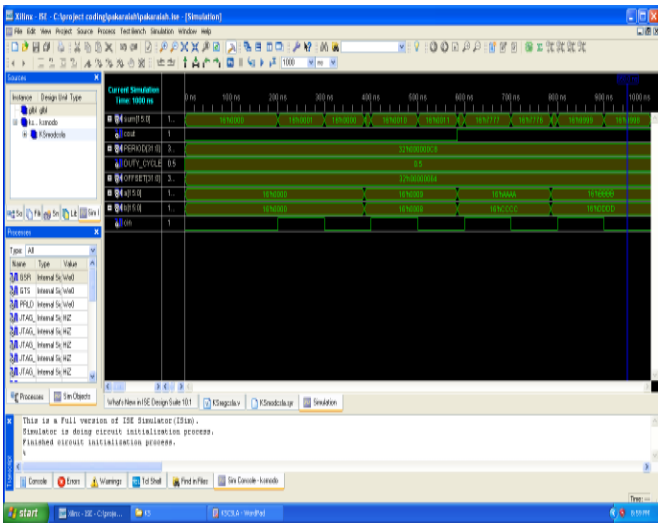

Figure6.5: Simulated output of 16-bit KS CSLA

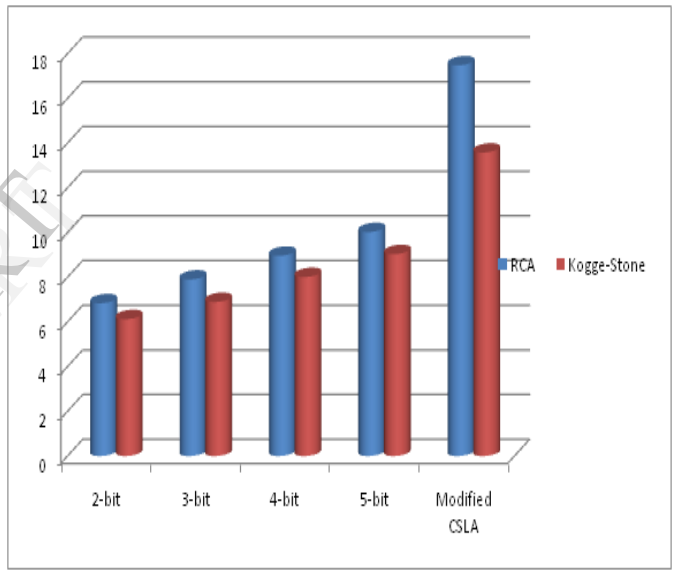

Figure6.6:Total delay values for different Adders

8. Conclusion

Both measured and simulation results from this study have shown that parallel-prefix adders are not as effective as the simple ripple-carry adder at low to moderate bit widths. This is not unexpected as the Xilinx FPGA has a fast carry chain which optimizes the performance of the ripple carry adder. However, contrary to other studies, we have indications that the carry-tree adders eventually surpass the performance of the linear adder designs at high bit-widths, expected to be in the 128 to 256 bit range. This is important for large adders used in precision arithmetic and cryptographic applications where the addition of numbers on the order of a thousand bits is

not uncommon. Because the adder is often the critical element which determines to a large part the cycle time and power dissipation for many digital signal processing and crypto graphical implementations, it would be worthwhile for future FPGA designs to include an optimized carry path to enable tree based adder designs to be optimized for place and routing. This would improve their performance similar to what is found for the RCA.

7. Acknowledgement

Mohammed Haseena Begum would like to thank Mr. V.Vamsi Mohana Krishna, Associate professor, in the Department of ECE who had been guiding throughout the project and supporting me in giving technical ideas about the paper and motivating me to complete the work efficiently and successfully.

8. References

[1] B. Parhami, Computer *Computer Arithmetic–Algorithm and Hardware Designs*, Oxford University Press, 2000.[2] M. Snir, "Depth-size trade-offs for parallel prefix computation," in *Journal of Algorithms* 7, pp.185–201, 1986. [3] J. Sklansky, "Conditional sum addition logic", in *IRE Tran. Electron. Computer.*, vol. EC-9, no. 6, pp.226-231, June 1960 [4] Y-C Lin and C-C Shih, "A new class of depth-size optimal parallel prefix circuits," in *The Journal of Supercomputing*, 14, pp.39–52, 1999. [5] S. Lakshmivarahan, C. M. Yang, and S. K. Dhall, "On a new class of optimal parallel prefix circuits with (size depth)=2n−2 and ⌈log n⌉ ≤ depth ≤ (2⌈log n⌉−3)," in *Proc. of the 1987 International Conference on Parallel Processing"*, pp.58–65, 1987. [6] Reto Zimmermann, "Non-heuristic optimization and synthesis of parallel prefix Adders," in *International Workshop on Logic and Architecture Synthesis (IWLAS'96)*, Grenoble, December 1996. [7] F. E. Fich, "New bounds for parallel prefix circuits," in *Proc. of the 15th Annu. ACM Sympos. Theory of Comput.*, 1983, pp.100–109. [8] S. Lakshmivarahan, S. K. Dhall, *Parallel Computing: Using the Prefix Problem*, Oxford University Press, 1994. [9] Reto Zimmer mann, Java applet for adder synthesis, http://www.iis.ee.ethz.ch/ zimmi/applets/prefix.html [10] Haikun Zhu, Chung-Kuan Cheng, Ronald Graham, "Constructing Zerodeficiency Parallel Prefix Adder of Minimum Depth," ACM Transactions on Design Automation of Electronic Systems, submitted [11] B. Ram Kumar, Harish M Kittur, "Low –Power and Area-Efficient Carry Select Adder", IEEE transaction on very large scale integration (VLSI) systems, vol.20, no.2, pp.371-375, Feb 2012.[12] Youngjoon Kim and Lee-Sup Kim, "A low power carry select adder with reduced area", IEEE International Symposium on Circuits and Systems, vol.4, pp.218-221, May 2001.