Performance Analysis Of Parallel Prefix Adder

Article *in* International Journal of Electrical Electronics and Data Communication · July 2015

DOI: 10.18479/ijeedc/2015/v3i7/48267

1 author:

Manjunatha Naik V
RNS Institute of Technology
1 PUBLICATION **5** CITATIONS

SEE PROFILE

PERFORMANCE ANALYSIS OF PARALLEL PREFIX ADDER

[1]MANJUNATHA NAIK V, [2]POORNIMA N

[1]Student of VLSI design & ES, Department of E & C, JSS Academy of Technical Education, Bengaluru, India
[2]Assistant professor, Department of E & C, JSS Academy of Technical Education, Bengaluru, India
E-mail: [1]manjunathanaikv@gmail.com, [2] poornimasumithra@gmail.com

Abstract—Addition is important in many of data path subsystems. The primary factor for any design is to reduce the power dissipation and to increase the performance. To achieve the performance and to reduce the power dissipation, selection of adder topology is an important thing. The use of parallel prefix adders (PPA's) increases the performance by reducing the power dissipation. This paper represents the comparison between the different prefix trees and the implementation of radix-4, 32 bit parallel prefix adder with sparseness of 4. The work involves the implementation of radix-2 and radix-4 32 bit parallel prefix adder structures by comparing power, delay, power delay product and number of computational nodes. The comparison result reveals that radix-4, 32 bit parallel prefix adder realizes minimum power and delay. The power and delay of both the adders are analyzed and compared. Cadence sim vision tool used for verilog implementation and Cadence virtuoso tool used for schematic implementation and the Encounter tool is used for the physical design of the both the adders.

Keywords —Parallel prefix adders, arithmetic and logic unit, Power delay product.

I. INTRODUCTION

Addition is important in all processing and computing devices such as ALU, and MAC design units. It is a most important in all data path sub systems. ALU is one of the applications where all the processing parameters are depends on its performance. So, design of adder is most important in consideration.

The main aim in ALU design is to reduce the adder critical path, which decides execution time in terms of delay and power. These two factors are most essential in adder design. Parallel prefix adder design is most preferable for their higher speed of operation. There are different algorithms are used in process of addition. They mainly aim on improvising the performance of PPAs by optimizing performance parameters such as Speed, Power, Area and number of gate counts. There are various topologies of prefix adders are there, they gives the comparisons among the various parallel tree adders [4-14].

These tree structures validate the benefits of each one of them with the other by making use of the performance parameters. The various tree structures are mentioned in the literature based on delay, fan-out and complexity in circuit design. Sklansky proposed a tree [4], where intermediate carry signals are computed by using tree structures. In case of the Kogge-Stone [5], it uses recursive doubling property which leads to the fan-out limited to unity at the each stage of carry merge. This uses more wires at each level of carry merge. The minimum depth prefix graphs is introduced by Ladner and Fischer [6]. If the wire of one node connecting it to the other n/2 nodes, so the fan-out will be more. To avoid these condition we need to make use of the appropriate buffering inverters are added to drive the large load. Brent-Kung [7] proposed a tree, where the lateral fan-out of each path restricted to unity. In Kogge-Stone tree [5] the logical depth is increased. Han-Carlson tree which is hybrid of both Brent-Kung and Kogge-stone adders [8]. Here, the computational node is reduced and slightly the logic depth will be increased. Some adders uses Sparseness to reduce the fan-out instead of the full parallel tree. There are plenty of sparse trees are implemented and published with sparseness of two [9], [10], [14] and four [3], [11], [12]. In [13] ling has proposed different carry generation equations to simplify the carry merge and to reduce the critical path delay.

In this work I have implemented the Radix-2 tree structure and Radix-4 structure with the sparseness of four. So, the number of logical stages, logic depth and hence, reducing the delay in computation. Then we relate both the structure and implemented it to the ALU.

The organization of paper is as follows. Section 2 describes the Prefix computation Structure. Section 3 tells about the various design choices. Section 4 presents various Prefix trees. Section 5 gives implementation of Radix-4 Han-Carlson Adder. Section 6 is the application of the prefix adder in ALU design. Section 7 gives the simulation result, inference and Comparison of various prefix trees and Section 8 Concludes the paper.

II. OVERVIEW OF THE PREFIX ADDER

Parallel prefix adder which uses Parallel Computation and Parallel Execution. It has three different stages for addition [6]. For the operand X and Y with width m, the figure below shows the structure of Prefix addition.

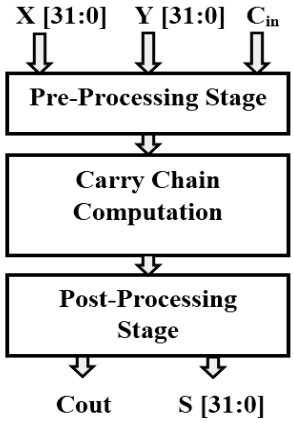

Fig.1. Structure of Prefix tree Adder.

A. Pre-Processing Stage:

The Pre-Processing Stage computes the all the Generate and Propagate signals of all the input bits of operand X and Y. The equations corresponding to Pre-Processing stage is as shown below:

$$G_i = X_i \cdot Y_i \quad (1)$$

$$P_i = X_i + Y_i \quad (2)$$

Where $0 \leq i \leq m-1$, here m represents the number of bits.

B. Carry Chain Computation:

In carry chain computation, the carry merge operation performed. It is a most hungry product in all prefix tree adders. The Group generate and Group Propagate signals are calculated for j^{th} bit to i^{th} bit using the below equation:

$$G_{i:j} = G_{i:k} + G_{k-1:j} P_{i:k.} \quad (3)$$

$$P_{i:j} = P_{i:k} P_{k-1:j.} \quad (4)$$

Where, $i > k > j$.

C. Post-Processing Stage:

The final product can be obtained by using an XOR gate or by using Conditional Sum Adders. In conditional sum adder the corresponding sums can be selected based on the carry in to the present stage.

While, designing the PPAs, there are various topologies which includes full tree or sparse trees. The validation of both the radix-2 and Radix-4 prefix tree are done by embedding it to the application of ALU design.

III. DESIGN CHOICES.

The group Generate and Propagate signals produced during the carry merge stage is modeled in to prefix trees [10], that are depends on the following parameters.

- *Radix/Valency/Prefix*: It identifies the number of bits to be merged at each logic stages.
- *Logic depth:* It defines maximum number of logic stages in carry graph.
- *Fan-out:* Highest number of nodes driven by a node in a stage.
- *Wire tracks:* Maximum number of wires routed across the bit pitch between two successive levels of the carry merge.

IV. RADIX-2 PREFIX ADDERS.

The Radix-2 Prefix adders have wide range of tree structures these can be validated through delay, fan-out and wiring complexity. The main block needed in the prefix adders are Carry chain computation or carry merge stage. It decides the performance parameters. The Black dots in the carry merge represents the Black cell which can hold both group Generate and Propagate signals. The rest of dots represents the Grey cells. Prefix adders are divided into six categories as mentioned in the literature [4-14]. Among them I have chosen only three of them, because of the high performance parameters.

A. 32 bit Brent-Kung Adder.

It computes the valency for the group of two bits and again these can be used to for the valency of four bit groups, which intern used for eight bit group & so on. The valencies are fan back to find carries in each bit. The tree requires $2(\log_2 N-1)$ logical levels or stages for N bit logic implementation. The fan out is limited to two at each stages. The figure below shows the dot diagram of 32-bit Brent-Kung Adder.

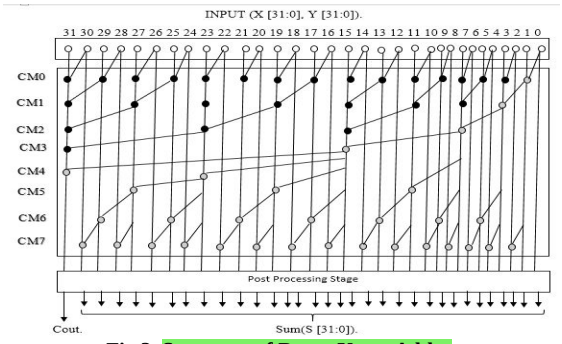

Fig.2. Structure of Brent-Kung Adder.

B. 32 bit Kogge-stone Adder.

The Kogge-Stone Adder achieves both $(\log_2 N)$ stages and the fan out of two. It uses too much of wires between the logic cells. Thus, the complexity of wiring increases. The tree contains more Propagate and generate cells. The area is not impacted on this design whereas power consumption increases. Despite of this drawbacks the Kogge Stone adder is most widely used. The figure below shows the dot diagram of 32 bit Kogge-Stone Adder.

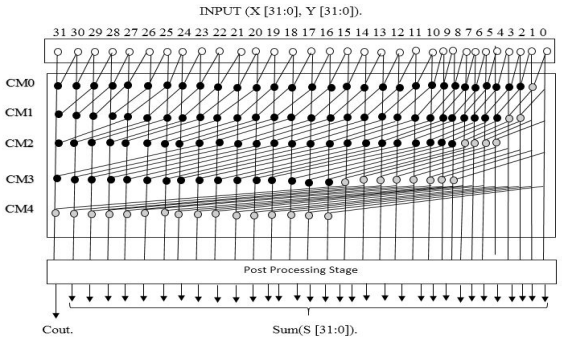

Fig.3. Structure of Kogge-Stone Adder.

C. 32 bit Han-Carlson Adder.

It is a network in which it comprises of both Kogge-Stone and Brent-Kung Adders. In which it performs Kogge-Stone on odd numbered bits, whereas this may uses the one or more ripple to the even positions so as to avoid the bit ambiguity and to reduce the data path delay, the Buffers are used at each of the long wire networks.

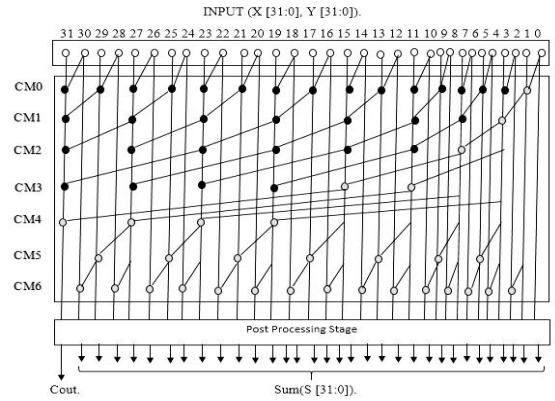

Fig.4. Structure of Hybrid Han-Carlson Adder.

V. RADIX-4, SPARSE-4, HAN-CARLSON ADDER.

Any carry graph can be converted to sparse tree and can have sparseness greater than two. Rather than using the full tree, sparse tree has been used. The sparse tree separates the critical path and non-critical path. Instead of producing carry at each bit. The sparse tree adder produced every 4th bit as shown in the dot diagram. It reduces the fan out, and reduces the interconnect length between the stages of the carry merge tree.

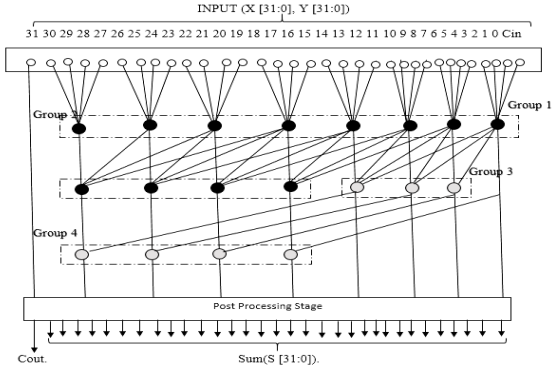

Fig.5. Structure of Radix-4, Sparse tree Han-Carlson Adder.

The figure above shows the carry graph of 32 bit prefix adder with sparseness of four, in structure the empty circle represents bit generate and propagate, top eight black dots which are shown as group one carry merge stage, groups the bit generate and propagate signals four by four to produce the first level group propagate and group generate signals using the equations shown below:

$$P_j = P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot P_i. \quad (5)$$

$$G_j = G_{i+3} + P_{i+3} \cdot G_{i+2} + P_{i+3} \cdot P_{i+2} \cdot G_{i+1} + P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot G_i \quad (6)$$

Here, j=i/4, and j= 0, 1,….7.

The next four black dots which are shown as group two carry merge further groups the signals from group one four by four and produces its second level Group propagate and group generate signals as shown below:

$$PGP_j = GP_{i+3} \cdot GP_{i+2} \cdot GP_{i+1} \cdot GP_i. \quad (7)$$

$$PGG_j = GG_{i+3} + GP_{i+3} \cdot GG_{i+2} + GP_{i+3} \cdot GGP_{i+2} \cdot GG_{i+1} + GP_{i+3} \cdot GP_{i+2} \cdot GP_{i+1} \cdot GG_i. \quad (8)$$

The carry signal {c7, c11, and c15} are produced by the carry merge block which is shown as group three. In figure, the carries {c19, c23, c27, c31} are produced from carry merge blocks which are shown as group four. By grouping the signals from group two and group three using the equation:

$$C_{19} = GG_0 + GP_0 \cdot C_3. \quad (9)$$

$$C_{23} = GG_1 + GP_1 \cdot C_7. \quad (10)$$

$$C_{27} = GG_2 + GP_2 \cdot C_{11}. \quad (11)$$

$$C_{31} = GG_3 + GP_3 \cdot C_{15}. \quad (12)$$

Then, by using appropriate number of conditional sum generators in the last stage all the sum bits can be produced. The non-critical section of the adder which works in parallel with the critical section consists of four bit conditional sum generators. As shown in figure, a four bit conditional sum generator produces the pair of sum bits, assuming an input carry of zero and one respectively

VI. ALU IMPLEMENTATION

To validate the efficient adder design I implemented the ALU which can perform faster than that of any other prefix adders. The same adder used for the subtraction with some modification made. There are certain operations performed with ALU design these functions can be shown in table below. The figure 6 shows the functional block diagram of ALU.

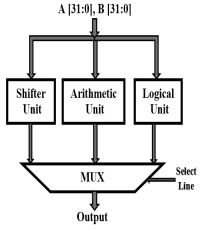

Fig. 6.ALU Block diagram.

Table .1 . ALU Functions.

Select line	Operation
0001	A + B + Cin
0010	A + B
0011	A – B – Cin
0100	A – B
0101	A * B
0110	A<< i
0111	A >>i
1000	A & B
1001	A + B
1010	A ^ B
1011	~ A

VII. SIMULATION RESULTS AND DISCUSSION

The figure below shows the simulation result which is performed for various adders.

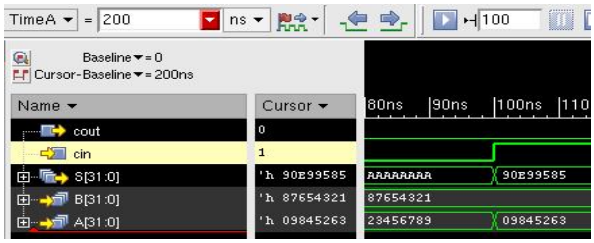

Fig.7. Simulation result of 32 bit adder.

The graph below shown in fig.8 represents the performance parameters i.e., delay and power between various adders.

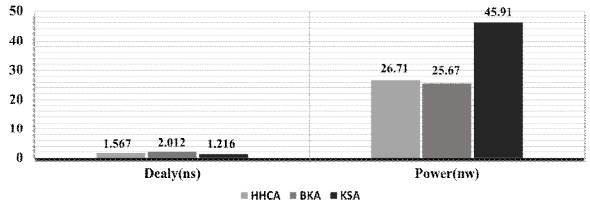

Fig.8. Comparisons of delay and power for various Radix-2 Prefix adders.

Table.2. Parameters comparison between both the adders.

Parameters	Radix-2 HCA	Radix-4 HCA
Cells	220	128
Area	1157	914
Power	26.713679 uw	25.032432 uw
Delay	1.567 ns	4.320 ns

Table.3. ALU implementation results.

Parameters	ALU with Hybrid HCA	ALU with Sparse Tree
Area	11767	11641
Gates	90721	90006
Power	6.885 mW	6.865 mW

CONCLUSIONS

By the simulation result i came to know that both the adders are functionally verified. And by synthesis and physical design I came to know that Radix-4, Sparse-4 Han Carlson Adder is more efficient than that of Radix-2 Hybrid Han Carlson Adder.

ACKNOWLEDGMENT

Working on this project has been great opportunity for me to gain valuable experience outside the academic world. I would like to take this opportunity to express my deepest gratitude to Mrs. Poornima N, Dept. of ECE, JSSATE and all supporting staff members and my family and friends.

REFERENCES

[1] Poornima N, and V. S. Kanchana Bhaaskaran, “Power-Delay Optimized 32 bit Radix-4, Sparse-4 Prefix Adder,” doi 10.1109/icsip.2014.38, pp.202–204.

[2] Neil H E Weste, David Harris and Ayan Banerjee, ”CMOS VLSI Design: A System Perspective”, 3rd Edition, 2008, pp. 303-360.

[3] Stefania Perri and Pasquale Corsonello,” Efficient Implementations of Radix-4 Parallel-Prefix Trees” ISBN: 978-1-61208-150-2, pp.2-4, 2011.

[4] J. Sklansky, ”Conditional-sum addition logic,” IRE Transactions on Electronic Computers, vol. 9, pp. 226-231, 1960.

[5] P. M. Kogge and H. S. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence equations,” IEEE Transactions on Computers, vol. C-22, no. 8, pp. 786-793, 1973.

[6] R. E. Ladner and M. J. Fischer, ”Parallel prefix computation,” Journal of the ACM, vol. 27, no. 4, pp. 831-838, 1980.

[7] R. P. Brent and H. T. Kung,” A Regular Layout for Parallel Adders,” IEEE Transactions on Computers, vol. C-31, no. 3, pp.260-264, 1982.

[8] T. Han and D. Carlson, “Fast area-efficient VLSI adders,” in Proceedings of IEEE Symposium on Computer Arithmetic, pp.49-56, May 1987.

[9] S. Mathew, R. Krishnamurthy, M. Anders, R. Rios, K. Mistry, and K.Soumyanath,”Sub-500 ps 64b ALUs in 0.18 m SOI/bulk CMOS: Design and scaling trends,” in IEEE ISSCC Dig. Tech. Papers, Feb.2001, pp. 318-319.

[10] S. Kao, R. Zlatanovici, and B. Nikolic, “A 240 ps 64b carry-look ahead adder in 90 nm CMOS,” in IEEE ISSCC Dig. Tech. Papers, Feb. 2006, pp.438-439.

[11] S. Naffziger, “A sub-nanosecond 0.5 m 64b adder design,” in IEEEISSCC Dig. Tech. Papers, Feb. 1996, pp. 210-211.

[12] Y. Shimazaki, R. Zlatanovici, and B. Nikolic, “A shared-well dual supply voltage 64-bit ALU,” IEEE J. Solid-State Circuits, vol. 39, Mar. 2004, pp.494-500.

[13] H. Ling, ”High speed binary adder,” IBM J. Res. Develop., vol. 25, no.3, May 1981, pp. 156-166.

[14] Sreenivaas Muthyala Sudhakar, Kumar P. Chidambaram and Earl E. Swartz lander Jr.”Hybrid Han-Carlson Adder”, in MWSCAS-2012 page.818-821.

★★★