Journal of Engineering Sciences
Vol 10, Issue 11, Nov /2019
ISSN NO:0377-9254

A LOW AREA AND DELAY EFFICIENT ARCHITECTURE OF A PARALLEL PREFIX ADDER

[1]G.V.ALEKHYA RANI , [2]D.JAYANAYUDU
[1]PG Student, Dept of ECE, SSNEC, ONGOLE, AP, India.
[2]Associate Professor, Dept of ECE, SSNEC, ONGOLE, AP, India.

ABSTRACT- Currently, parallel prefix adders (PPA) are considered effective combinational circuits for performing the binary addition of two multi-bit numbers. These adders are widely used in arithmetic-logic units, which are parts of modern processors, such as microprocessors, digital signal processors, etc. This paper deals with Kogge-Stone adder, which is one of the fastest PPA. When performing the schematic implementation, this adder has a large hardware complexity. Therefore, in this work for reducing its hardware complexity the scheme of modified PPA has been developed. The performance parameters considered for the comparative analysis of the presented adders are: area and maximum delay obtained when schematic modeling in the environment of Xilinx ISE. As a result, when simulation of 32-bit adder, compared to Kogge-Stone adder, modified PPA have better delay.

I. INTRODUCTION

To humans, decimal numbers are easy to comprehend and implement for performing arithmetic. However, in digital systems, such as a microprocessor, DSP (Digital Signal Processor)or ASIC (Application-Specific Integrated Circuit), binary numbers are more pragmatic for a given computation. This occurs because binary values are optimally efficient at representing many values. The hardware implementation of binary addition is a fundamental architectural component in many processors, such as microprocessors, digital signal processors, mobile devices and other hardware applications. In these systems when building arithmetic logic unit (ALU), adders play an important role for performing the basic arithmetic operations, such as addition, subtraction, multiplication, division, etc. Therefore, the hardware implementation of an effective adder is necessary to increase the performance of ALU and, consequently, the processor itself as a whole. Currently, a parallel prefix adder (PPA) is considered effective adder for performing the addition of two multi-bit numbers. Circuit complexity and the speed of PPA are important parameters at the stage of efficient hardware implementation and, therefore, in recent years various types of PPA with differentcharacteristicsoftheparametershavebeende veloped.*Kogge-Stone* adder [3] is investigated, which is one of the known effective fastest PPA. *Kogge-Stone* is widely and efficiently used. Such an adder has minimum delay while performing the binary addition. However, for estimation of hardware costs this adder has a great number of logic gates and *Quine*-complexity used in the schematic implementation. Therefore, in the present work for reducing its hardware complexity a modified parallel prefix adder is developed .Then, the comparison of the two presented adders is made by the following parameters: the number of logic gates, *Quine*- complexity, as well as the delay obtained by simulation in XILINX ISE 14.4 DESIGN SUIT. Perspective architecture is proposed for schematic implementation of various PPA. And derivation of the formulas is also described for computing the hardware characteristics which are dependent on the bit width of input operands of the present adders. Parallel prefix adder(PPA)is a multi-bit carry-propagate adder which is used for parallel addition of two multi-bit numbers.PPA extends the generated and propagated logic of the carry look-ahead adder to perform addition even faster [4]. As the basic schematic structure of the various PPA, perspective architecture is analyzed, it consists of three stages [5]: pre-processing stage, prefix computation stage and final processing stage. Let consider each stage in more detail

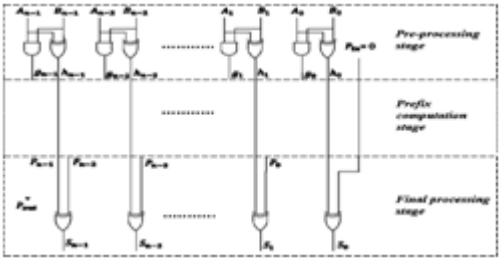

Fig.1. Architecture of the parallel prefix adder

At the pre-processing stage, carry-generate g_i and carry- propagate h_i signals are computed for each pair of input operands $A_i$and B_i . The calculation of these signals is described by the following corresponding logical equations:

$$g_i = A_i \cdot B_i;\ i = 0,1,2,\ldots,n-1 \qquad (1.1)$$

$$h_i = A_i \oplus B_i;\ i = 0,1,2,\ldots,n-1 \qquad (1.2)$$

At the prefix computation stage, group carry-generate G[i:k]and group carry-propagate H[i:k] signals are calculated for each bit by the following equations:

$$G_{[i:k]} = \begin{cases} g_i; & if \ \ i = k \\ G_{[i:j]} + H_{[i:j]} \cdot G_{[j-1:k]}; & otherwise \end{cases} \quad (1.3)$$

$$H_{[i:k]} = \begin{cases} h_i; & if \ \ i = k \\ H_{[i:j]} \cdot H_{[j-1:k]}; & otherwise \end{cases} \quad (1.4)$$

The final processing stage involves the formation of output carries and sum-values for each individual operand bit. The expression for $P_i$and S_i is defined as the following equations respectively:

$$P_i = G_{[i:0]} \quad (1.5)$$

$$S_i = h_i \oplus P_{i-1} \quad (1.6)$$

Where, P i-1 is carry-in (Pin=0). In the framework of this paper, basic schematic nodes are used for greater visibility of the given architecture when constructing the presented adders. Figure 2.2.2 shows the basic schematic nodes: a black cell, a gray cell, a white cell and a circle. These schematic nodes are implemented by the logical equations at all stages of the given architecture. The number of logic gates and Quine-complexity of each schematic node are computed for estimation of hardware costs. In this work Quine-complexity is determined by the total number of inputs of all the logic gates used in the schematic nodes. And one can also calculate the number of logic gates used in them.

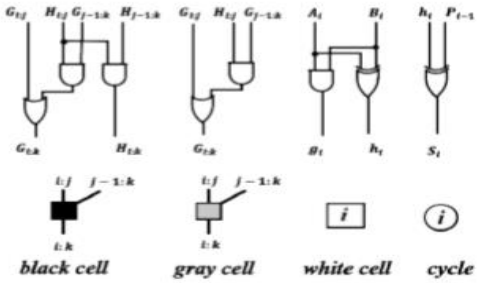

Fig.2. Basic schematic nodes

Figure.2 shows that a black cell contains 3 logic gates and Quine-complexity is 6, and a gray cell has 2 logic gates and Quine-complexity = 4. In these two cells, G[i:k] and H[i:k] signals are calculated for each bit by equations (1.3) ɢ (1.4) at the prefix computation stage. ⊄⌊ using these equations, a black cell and a gray cell receive inputs from the upper part of block spanning bits i:j and take inputs from the lower part spanning bits j-1 : k . Then, these schematic cells are combined to form a tree of generate and propagate signals for the entire block spanning bits i:k . So, the main challenge is to compute rapidly all the group generated signals G0:0 , G1:0 , G2:0, G3:0 , . . . , Gn-1:0 . These signals, along with propagated signals H0 :0 , H1:0 , H2:0 , H3:0 , . . . , Hn-1:0 are called prefixes [Error! Reference source not found.]. The network of these schematic nodes (black cell + gray cell) is a prefix tree. The white cell consists of 2 logic gates and Quine complexity = 4, it serves to calculate i g and i p signals of input operands Ai and Bi with equations (1.1) and (1.2) at the pre-processing stage. At the final stage for performing the results of the binary addition with equation (1.6) the circle consisting of one logic gate is used and Quine-complexity is equal 2.

II. EXISTING KOGGE-STONE ADDER

A. KOGGE-STONE ADDER

Kogge-stone adder is a parallel-prefix form carry look ahead adder. which has a minimum delay. Kogge-Stone adder was developed by peter M. Kogge and Harold S. Stone which they published in 1973. This adder is widely used in high performance applications.

B. 16-bit Kogge-stone adder

The scheme of a 16-bit Kogge-Stone adder is shown in figure 3.

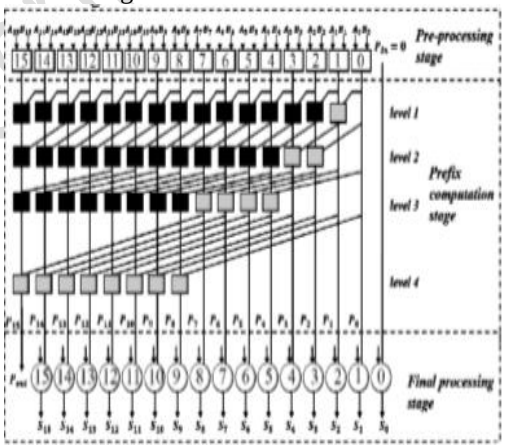

Fig. 3. Scheme of a 16-bit Kogge-Stone adder

This adder computes gi and hi signals for the pre processing stage. Then at the first level (L=l) of prefix tree, G[i :k] and H[i:k] signals of 2-bit are computed within the same time. At the second level (L=2) of prefix tree, G[i:k] and H[i:k] of 4-bit are calculated by using the result of 2-bit at level 1. Therefore, the actual carry-out value of the 4th bit would be available while the calculations at level 2 are being computed. At the third level (L=3)of prefix tree, the carry-out of the 8th bit is computed by using the 4th bit carry result. The same method adopted at level 3 is applied to get carry-out values of the 16th bit in fourth level (L=4) and etc. All other carries of bit are also computed in parallel. Finally, at the final processing stage the sums are computed from these final carry-out signals of the prefix tree.

In the prefix tree the number of levels corresponds to L= log_2^n and the number of schematic nodes (white cell + gray cell) will be (K= [n(log_2^n) − n + 1]. Quine-complexity SKQ . and the number of logic gates S KC . in the Kogge-Stone adder are given by the following equation.

$$C_{K.S} = 3n(\log_2 n) - n + 4 \quad (1.7)$$

$$Q_{K.S} = 6n(\log_2 n) - 2n + 8 \quad (1.8)$$

C. 32-bit Kogge-stone adder

Figure 4 shows the scheme of a 32-bit Kogge-Stone adder with increasing the bit width of input operands more than 16 bits.

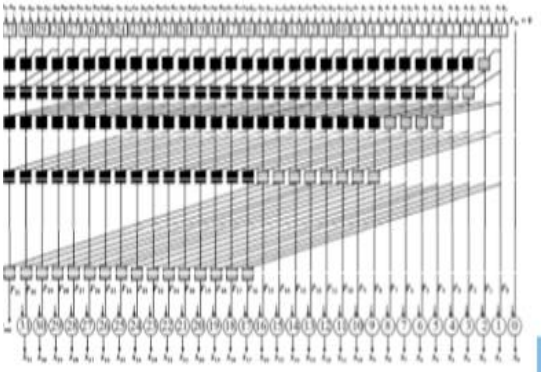

Fig.4. Scheme of a 32-bit Kogge-Stone adder

III. MODEFIED PARALLEL PREFIX ADDER

A. Modified PPA

Modified parallel prefix adder is developed for reducing the hardware complexity of Kogge-Stone adder. Figure 5 shows the scheme of a 16-bit modified parallel prefix adder.

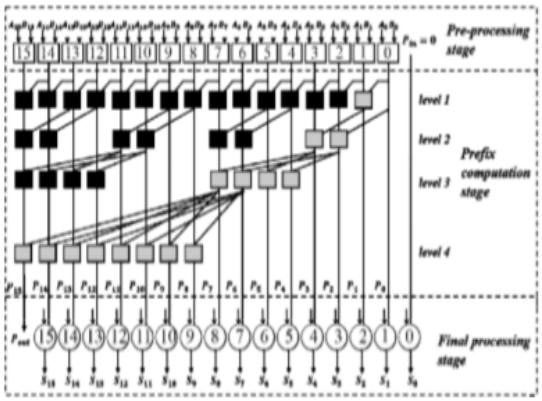

Fig.5. Scheme of a 16-bit modified parallel prefix adder

The construction of the first level of the prefix tree of this adder is similar to the construction of Kogge-Stone adder. The main structural difference begins from the second level of the prefix tree. At the second level of the prefix tree, the groups of two schematic nodes are formed, at the 3rd level - groups compose four schematic nodes and at the 4th level – groups including 8 schematic nodes, etc. This adder first computes gi and hi signals for the first stage. Then at the first level of prefix tree, G[i:k] and H[i:k] signals of 2-bit are computed at the same time, and then, it computes G[i:k] and H[i:k] signals for pairs of columns, then for blocks of 4, then for blocks of 8, then 16, and so on until the final G[i:k] signal for every column is known. Finally, at the last stage this adder computes the sums together with the generated signals obtained from the previous prefix computation stage. The number of levels of the prefix tree corresponds to (L=log_2^n) and the number of schematic nodes will be

$$(k = [(\frac{n}{2}\log_2 n - 2) + \frac{n}{2} + 1]).$$

The number of logic gates C Modified PPA and Quine-complexity Q Modified PPA of this modified adder are calculated using the following equation

$$C_{Modified\ PPA} = [\frac{3n}{2}(\log_2 n + 1)] + 2n - 2 \quad (1.9)$$

$$Q_{Modified\ PPA} = 3n(\log_2 n + 1) + 4n - 4 \quad (1.10)$$

B. 32-bit Modified PPA

The scheme of a 32-bit modified parallel prefix adder is shown in Figure 6 with increasing the bit width of input operands more than 16 bits.

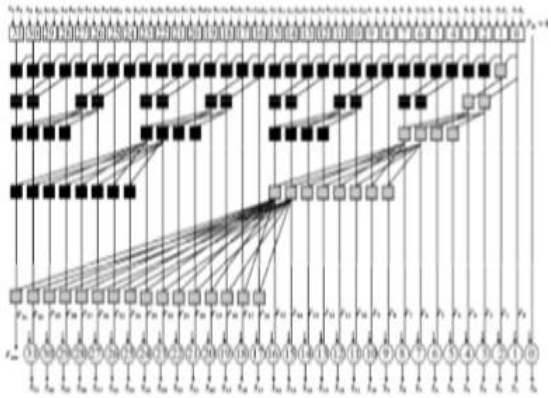

Fig.6. Scheme of a 32-bit modified parallel prefix adder

IV. SIMULATION RESULTS

The corresponding simulation results of the existing(16,32-bit) and proposed(16,32-bit) kogge-stone adder are shown below. Inputs a,b,cin are supplied from test bench we can apply any number of inputs and can observe the respective output to verify whether the design is working properly or not.

The above fig 7 shows the simulation results of existing 16-bit Kogge stone adder with 3 inputs a,b of size 16 bits and cin and two outputs SUM and Count which generates the sum and count value based on given inputs.

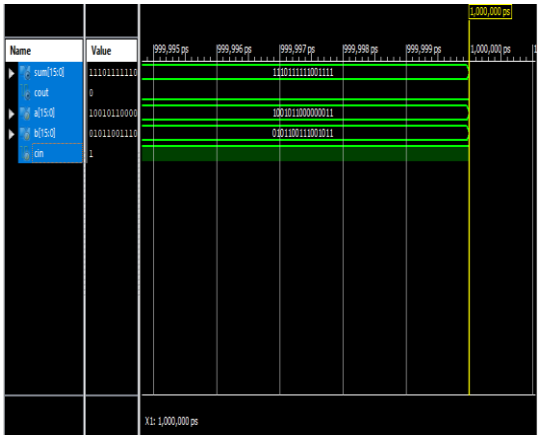

Fig.7. simulation output of existing 16-bit kogge stone adder

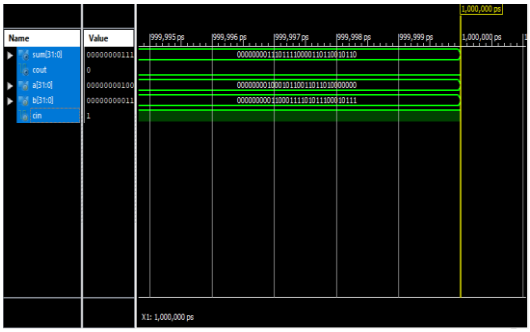

Fig.8. simulation output of existing 32-bit kogge-stone adder

The above fig 8 shows the simulation results of existing 32-bit Kogge stone adder with 3 inputs a,b of size 32 bits and cin and two outputs SUM of size 32 bits and Count which generates the sum and count value based on given inputs.

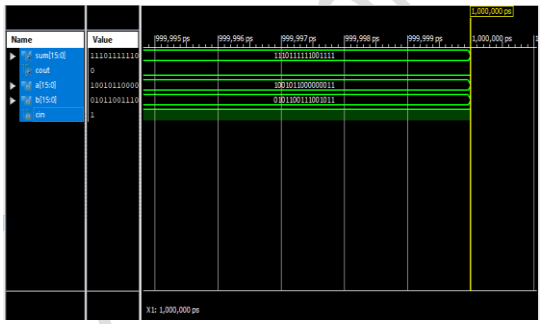

Fig.9. simulation output of proposed 16-bit kogge-stone adder

The above fig 9 shows the simulation results of proposed 16-bit Kogge stone adder with 3 inputs a,b of size 16 bits and cin and two outputs SUM of size 16 bits and Count which generates the sum and count value based on given inputs.

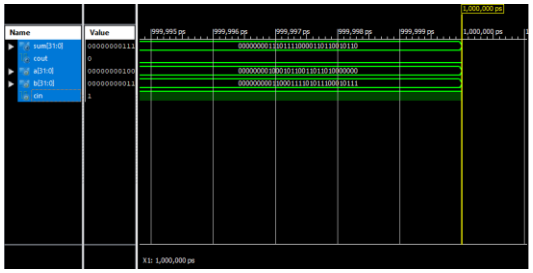

Fig.10. simulation output of propose 32-bit kogge-stone adder

The above fig 10 shows the simulation results of proposed 32-bit Kogge stone adder with 3 inputs a,b of size 32 bits and cin and two outputs SUM of size 32 bits and Count which generates the sum and count value based on given inputs. The simulation result is obtained when simulated the code in XILINX ISE 14.7.

V. CONCLUSION

Analysis of the perspective architecture for constructing various multi-bit PPA schemes; derivation of formulas for estimating the hardware complexity of multi-bit PPA ,schematic implementation of the standard 16-bit and 32-bit Kogge-Stone adders and schematics implementation of 16-bit and 32-bit modified parallel prefix adders. Then, a comparative analysis of parameters and simulation results of the presented adders have been carried out. As a result, researches have shown, that the modified parallel prefix adder proposed in the work has an advantage in terms of hardware complexity in comparison with the known structure of KoggeStone adder. Additionally, in terms of speed the proposed parallel prefix adder has the advantage over group-prefix and carry-look a head adders, and famous as parallel prefix adders Sklansky and Brent-Kung.

REFERENCES

[1] Geeta Rani, Sachin Kumar. “Delay Analysis of Parallel-Prefix Adders”. International Journal of Science and Research (IJSR), ISSN: 2319-7064, Impact Factor (2012): 3.358. Volume 3 Issue 6, June, 2014. pp. 2339.

[2] Reto Zimmermann. Binary Adder Architectures for Cell-Based VLSI and their Synthesis. Thesis for the degree of Doctor of technical sciences. Zurich. 1997. pp. 5-7.

[3] Sunil.M, Ankith.R.D, Manjunatha.G.D and Premananda.B.S. Design and implementation of faster parallel prefix Kogge Stone adder. International Journal of Electrical and Electronic Engineering & Telecommunications 2014. ISSN 2319 – 2518. Vol. 3, No. 1, January 2014. pp. 116.

[4] Athira.T.S, Divya.R, Karthik.M, Manikandan.A. Design of Kogge-Stone for fast

addition. Proceedings of 34th IRF International Conference, 26th February 2017, Bengaluru, India. ISBN: 978-93-86291-639. pp. 27-28.
[5] CH. Sudha, Rani, CH. Ramesh. Design and Implementation of High Performance Parallel Prefix Adders. International Journal of Innovative Research in Computer and Communication Engineering. An ISO 3297: 2007 Certified Organization. Vol.2, Issue 9, September 2014. pp. 5900.
[6] Bazarova S. B. M., Mantatov B.V. Adders: Methodical instructions for laboratory work. Publishing house of the SSCU. Ulan-Ude. 2006. pp. 810.
[7] David Money Harris, Sarah L. Harris David. Digital Design and Computer Architecture. 2nd Edition. Avenue South, New York, 2013. pp. 237-238.
[8] Vibhuti Dave. High-speed multi operand addition utilization flag bits. Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering, Chicago. Illinois. May 2007. pp. 38-39.