International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

DESIGN OF HIGH SPEED 64 BIT ADDER USING PARALLEL PREFIX STRUCTURE

KAVYA REGATIPALLI[1], N.VARALAKSHMI[2]
[1]PG Scholar, Dept of ECE, SIR C.V. RAMAN Institute of Technology & Science, AP, India
[2] Assistant Professor, Dept of ECE, SIR C.V. RAMAN Institute of Technology & Science, AP, India

Abstract: 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. 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.

1. INTRODUCTION

In electronics, an adder or summer is a digital circuit that performs addition of numbers. In many computers and other kinds of processors, adders are used not only in the arithmetic logic unit(s), but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar operations. Although adders can be constructed for many numerical representations, such as binary-coded decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor. Other signed number representations require a more complex adder. The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition. The value of the sum is 2C + S. The simplest half-adder design, pictured on the right, incorporates an XOR gate for S and an AND gate for C. With the addition of an OR gate to combine their carry outputs, two half adders can be combined to make a full adder.

2.LITERATURE SURVEY

Kogge P, Stone H proposed A parallel calculation for the productive arrangement of a general class Recurrence relations . A mth-demand rehash issue is described as the estimation of the plan x1, x2, ..., XN, where xi = fi(xi-1, ..., xi-m) for some limit fi. This paper uses a framework called recursive duplicating in an estimation for enlightening an enormous class of rehash issues on equal PCs, for instance, the Iliac IV. Recursive duplicating incorporates the piece of the estimation of a limit into two correspondingly complex subfunctions whose appraisal can be performed simultaneously in two separate processors. Reformist piece of all of these subfunctions spreads the estimation over more processors. This computation can be associated with any recurrent state of the construction xi = f(bi, g(ai, xi-1)) where f and g are limits that

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

satisfy certain distributive and partnered like properties. Disregarding the way that this rehash is first solicitation, all straight mth-demand rehash conditions can be tossed into this construction. Sensible applications join direct recurrent conditions, polynomial appraisal, a couple of nonlinear issues, the confirmation of the best or least of N numbers, and the plan of tridiagonal straight conditions. The ensuing computation enlists the entire course of action x1, ..., xN in time comparing to [log2 N] on a PC with N-cross-over parallelism. On a consecutive PC, computation time is comparative with N.

D. Harris proposed A Taxonomy of Parallel Prefix Networks Equal prefix frameworks are extensively used in first class adders. Frameworks in the composing address tradeoffs between number of reasoning levels, fanout, and wiring tracks. This paper shows a three-dimensional logical classification that not simply portrays the tradeoffs in existing equal prefix puts together yet moreover centers to a gathering of new frameworks. Adders using these frameworks are contemplated using the methodology for reasonable effort. The new designing is forceful in dormancy and area for specific advances.

Alvaro Vázquez, Florent de Dinechin (Multi-operand Decimal Adder Trees for FPGAs) the creative work of hardware plans for decimal number shuffling is correct now going under an outstanding activity. For most part, the strategies proposed to realize fixed and coasting point assignments are normal for ASIC structures. As such, a prompt planning or change of these techniques into a FPGA could be far from an optimal course of action. Only a few examinations have considered new procedures progressively sensible for FPGA utilization. An essential action that has not gotten sufficient thought in this setting is multi-operand BCD extension.

Performance of Parallel Prefix Adders Implemented with FPGA technology BY K. Vitoroulis and A. J. Al-Khalili

Equal Prefix Adders have been set up as the most productive circuits for paired expansion. Their ordinary construction and quick execution makes them especially appealing for VLSI execution. The traditional equal prefix viper structures that have been proposed throughout the years advance for rationale profundity, region, fan-out and interconnect check of the rationale circuits. This paper researches the exhibition of equal prefix adders carried out with FPGA innovation. We report on the space prerequisites and basic way delay for an assortment of old style equal prefix viper structures.

3.EXISTING SYSTEM

Ripple Carry Adder

RCA is a combinational logic circuit. It is one of the simplest forms of adders that is constructed by cascading multiple full adder circuits to form an N-bit adder. The carryout of each full adder is the carry-in of the next full adder in the sequence. Each carry from the previous adder gets rippled into the next adder and hence the name RCA [1]. Here, the carry-out and sum bits of any particular stage is valid only when the carry-in of the previous stage is considered for addition. Computation of the output bits is carried out sequentially but, the implementation is done in parallel since all the inputs are fed into the circuit at the same time. The propagation delay in the circuitry is an important aspect. It is the amount of time progressed between the feeding of the input and obtaining the corresponding output. The sum and carry-out of the first full adder are considered to be logical after the propagation delay of that adder is computed. Similarly, the final

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

output of an N-bit RCA is authentic only after the propagation delay of the individual full adder is summed up. This adder circuit is a combinational circuit that uses simple blocks that are connected together to perform a complex function. There are a few disadvantages of RCA circuit. RCA does not facilitate the usage of all full adders simultaneously. Each full adder has to wait compulsorily for the carry output from the previous full adder for the computation which increases the delay. Due to this, RCA works extremely slow. This leads to the search for better and faster adders.

B. Carry Look Ahead Adder

In a digital circuit the most critical parameters during the design are the power usage and its speed. In the case of digital processing systems, minimal power consumption and reliability concerns of arriving at the expected performance output can be a challenging task. Adders are the building blocks in arithmetic and logic units. Adders are also used in other arithmetic operations, such as multiplication and division, which are the most power utilizing components in the processors. In RCA, the bits to be added are available simultaneously; however, each adder block must wait for the carry to come out from its forerunner block. If we extend the number of stages, the propagation delay gets added up. As a solution, there exists an adder, i.e., CLA, which reduces the propagation delay upon the introduction of a complex adder. It solves the problem of carry propagation faced in RCA by introducing the generate bits and propagate bits. The Boolean expression for each carry output is the sum of the products (SOP). Each SOP is implemented with one level of AND gate followed by an OR gate [2]. The CLA comprises of two parts: The first part quantifies the carry for each bit and the second part adds up the input and carry bits for each bit position.

4.PROPOSED PARALLEL PREFIX ADDERS

Parallel prefix adders are faster adders which are used for high-performance arithmetic operations. It is a type of CLA mwhere carry calculation is carried out prior. Operations can be divided into three stages (a) Pre-processing stage. (b) Carry generation stage (c) Post-processing stage [14].

In every bit (i) of the two operand block, the two input signals (ai and bi) are added to the corresponding carry-in signal (carryi) to produce sum output (sumi) The equation to produce the sum output is:

$$Sumi = ai \wedge bi \wedge carryi \quad (1)$$

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 (gi) and propagate (pi) using the equations:

$$Gi = ai \wedge bi \quad (2)$$

$$Pi = ai \wedge bi \quad (3)$$

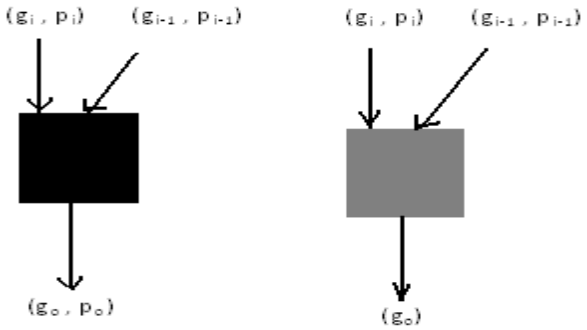
The carry in signal for any adder block is calculated by using the formula

$$Ci+1 = gi\ V\ (pi) \quad (4)$$

Where ci must be expanded to calculate ci+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.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362



(a) black operator (b) gray operator

Fig 1 Operators used in Parallel Prefix trees

The black operator receives two sets of generate and propagate signals (gi , pi),(gi-1 ,pi-1), computes one set of generate and propagate signals (go , po) by the following equations:

Go = gi V (pi ^ gi-1) (5)

Po = pi ^ pi-1 (6)

The gray operator receives two sets of generate and propagate signals (gi, pi),(gi-1 ,pi-1), computes only one generate signal with the same equation as in equation (5).

The real carry for the lower half bits (0 to 31) is obtained from Eq. 6 that represents the operation of the white hexagon node. The carry for the upper half bits (32 to 63) is obtained from the Eq. 5 represented by the black hexagon node. The final sum is calculated in the last stage using the half sum bits computed in the pre-processing stage and carry bits computed in the prefix calculation stage. The sum is given by the following equation.

Sumi = ai ^ carryi (7)

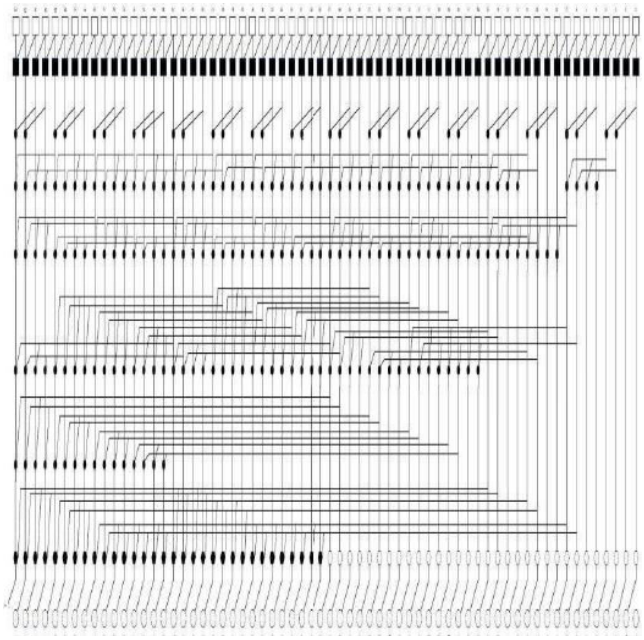


Fig. .2. Proposed 64-bit parallel prefix adder structure

The proposed 64-bit parallel prefix adder structure, as shown in Fig. 1, has seven levels. Generate, propagate, halfsum, carry, and sum calculations at different levels form a complete 64-bit adder. The white square node, which forms the pre-processing stage of the parallel prefix addition gives out the generate, propagate, and half sum for each individual bit. The black square node in the second row gives out the intermediate propagate and intermediate generate bits by considering the present propagate and generate bits, and its previous propagate and generate bits.

The black circle nodes, white and black hexagon nodes, along with the output of the black square nodes compute the real carry *Ci* at each bit. This process is the prefix calculation stage. The last row is the post-processing stage, which is the final stage. Here for each bit, XOR operation is performed on the half-sum bit calculated at the pre-processing stage with the carry of the previous bit, i.e., *Ci-1,* which is computed at the end of the prefix calculation stage. The

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

final stage output gives the 64-bit sum and 1-bit carry *Cout*. Considering the 15th bit of the 64-bit adder as an example, the first row in the adder structure consists of the white square node having the inputs *a15* and *b15*. This row is called the pre-processing stage where the generate, propagate, and half-sum bit is computed by the following equations.

$$gen_{15} = a_{15} \text{ AND } b_{15} \quad (8)$$

$$pro_{15} = a_{15} \text{ OR } b_{15} \quad (9)$$

$$d_{15} = a_{15} \text{ XOR } b_{15} \quad (10)$$

The second row in the structure has black square nodes which compute the intermediate generate and propagate bits. Here the intermediate generate and propagate bits of the 15th bit is computed by,

$$G^*_{15} = gen_{15} \text{ OR } gen_{14} \quad (11)$$

$$P^*_{15} = pro_{14} \text{ AND } pro_{13} \quad (12)$$

The third row in the structure is a black circle node that takes four inputs, where two inputs are intermediate generate, and two inputs are intermediate propagate bits. Here at the 15th bit, the inputs will be *G15:14, P14:13, G13:12,* and *P12:11*. Now *G15:12* and *P14:11* after the black circle logic computation is given by

$$G_{15:12} = G_{15:12} \text{OR } (G_{13:12} \text{ AND } P_{14:13}) \quad (13)$$

$$P_{14:11} = P_{4:13} \text{ AND } P_{12:11} \quad (14)$$

The fourth row in the structure is the black circle node. The node considers the input *G15:12, P14:11, G11:8,* and *P10:7*. After a similar black circle logic operation, the node computes the group generate and group propagate bits, i.e., (*G15:8, P14:7*). The fifth row is the black circle node similar to the fourth row. It has four inputs *G15:8, P14:7, G7:0,* and *P6:-1* and at the output, we have the group generate bit and group propagate bit computation *G15:0* and *P14:-1* respectively. In the eighth row, the white hexagon computes the real carry of each bit. It has two inputs, *H15* and *pro15*. Its logical computation is performed as

$$C_{15} = H_{15} \text{ AND } pro_{15} \quad (15)$$

After obtaining the real carry at the prefix calculation stage, we compute the sum. The *Sum15* is given as,

$$Sum_{15} = d_{15} \oplus C_{14} \quad (16)$$

5.SIMULATION RESULTS

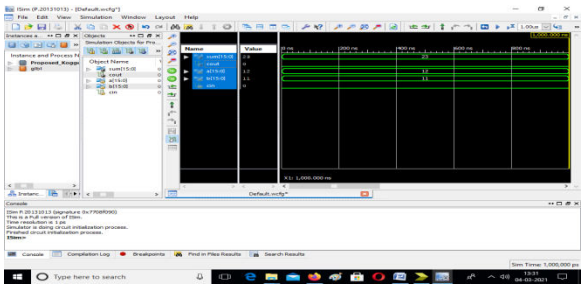


Fig.3 Output wave form

Fig 3 show the simulation result of proposed adder a, b, cin as the input and s and cout as the output

6. CONCLUSION & FUTURE SCOPE

A detailed design and a structure of the 64-bit parallel prefix adder is proposed in this paper. Here the Ling adder is used in the design of the parallel prefix adder to further reduce the gate count. The 64-bit adder structure implementation will help in faster execution of the larger bit-width operations. The faster execution comes at the cost of area and power consumption. In the future, a 128-bit parallel prefix adder can be designed by building upon the proposed adder structure. The reduction of the logic elements involved can always be achieved, which will further decrease the area and

power consumed. This adder can be used for any signal processing application which needs fast execution and can compromise on the power and area.

REFERENCES

[1]. K. Vitoroulis and A. J. Al-Khalili, "Performance of Parallel Prefix Adders Implemented with FPGA technology," IEEE Northeast Workshop on Circuits and Systems, pp. 498-501, Aug. 2007.

[2]. D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, "Easily Testable Cellular Carry Lookahead Adders," Journal of Electronic Testing: Theory and Applications 19, 285-298, 2003.

[3]. S. Xing and W. W. H. Yu, "FPGA Adders: Performance Evaluation and Optimal Design," IEEE Design & Test of Computers, vol. 15, no. 1, pp. 24-29, Jan. 1998.

[4]. M. Bečvář and P. Štukjunger, "Fixed-Point Arithmetic in FPGA," Acta Polytechnica, vol. 45, no. 2, pp. 67-72, 2005.

[5]. P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. on Computers, Vol. C-22, No 8, August 1973.

[6]. P. Ndai, S. Lu, D. Somesekhar, and K. Roy, "Fine-Grained Redundancy in Adders," Int. Symp. on Quality Electronic Design, pp. 317-321, March 2007.

[7]. T. Lynch and E. E. Swartzlander, "A Spanning Tree Carry Lookahead Adder," IEEE Trans. on Computers, vol. 41, no. 8, pp. 931-939, Aug. 1992.

[8]. N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson–Addison-Wesley, 2011.

[9]. R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, pp. 260-264, 1982.

[10]. D. Harris, "A Taxonomy of Parallel Prefix Networks," in Proc. 37th Asilomar Conf. Signals Systems and Computers, pp. 2213–7, 2003.