IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 9, Issue 2, Ser. I (Mar. - Apr. 2019), PP 06-13
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Addition Algorithms for VLSI – A review

KapilRamGavali[1],SandeepDubey, Gaurav Shete, Sushant Gawade
DepartmentofElectronics&TelecommunicationEngineering,
Universal College of Engineering, Mumbai,India
Corresponding Author: KapilRamGavali

Abstract: *Miniaturization being the need of the hour, each system needs to have the least possible area utilized. However in the process of doing the same the speed of the system needs to be considered. As for any system addition is the most basic operation, this paper deals with the analyzing and reviewing of different addition algorithms namely, ripple carry, carry select, carry skip, carry save and carry look ahead adders by performing parallel addition of 8 unsigned numbers each of 12 bits using pipelining technique. Either of the adders above can be used depending on the applications as each of them has tradeoffs in terms of area or speed or power.*

Keywords –*VLSI, Signal Processing, Adders*

Date of Submission: 28-03-2019 Date of acceptance: 13-04-2019

I. Introduction

Even though adders seem to be the most simple of the operations to be performed for a human being, designing them under a computer can be an uphill task. This is due to various constraints while designing like area, speed, time, etc.As such, adders arepredominantly used in digital signal processors and microprocessors[1], [2],[3],[4],[5],[6].

Many designs of adders have been reported in the literature and each may have its own advantages and shortcomings. Critical constraints like area, speed or power may get improved or deteriorated depending on the design. If an adder is to be used in a system or processor, one has to decide on the circuit that is most appropriate for its architecture and design.

Ripple carry adder is one of the very simple adders used on a large scale. It employs a one bit full adder to add two one bit numbers[5][8].As such, a ripple carry adder will require as many number of full adders as the number of bits. However, ripple carry adder is not suitable in real time systems especially where time plays a critical role. A carry look ahead adder is used in such cases albeit with a large area.

The carry save adder is a layered structure employing a full adder as well as a half adder. These adders are useful in applications where the system design requires less area to be utilised by the system coupled with high speed of operation [9][10]. The conventional carry input of full adder is replaced by a third binary number. This adder is called as a carry save adder owing to the fact that the carry output of the full adder is not used immediately and rather saved.

The carry select adder divides the number into two blocks (lsb and msb) [11].In the basic carry select adder design, two additions are computed for the msb block concurrently, with one addition process assuming the carry in bit as zero and the other as one [7][12]. Use of carry select adder increasesthe speed by fast sum generation of the upper few bits [13].However, the hardware utilization also increases. This will lead to increase in area and power consumed by the circuits.

Another classic example of fast adder is the carry look-ahead adder [14]. In carry look-ahead adder, the final result is computed using two special intermediate signals viz. generate and propagate to increase the operating speed[5]. The sum and carry-out bits are obtained by XOR and AND operations respectively, by using the already obtained generate and propagate terms. This adder greatly results in increased speed of operation. Also, the speed is independent of the word size, unlike in the case of ripple carry adder.

Carry skip adders invented by C. Babbage in 1800's are faster as well as require less area and power [15]. The carry skip adder divides the words to be added into blocks, in which the sum and carry are calculated. Likewise the carry of succeeding blocks is calculated from the preceding carry. Carry skip adders have low complexity and moderate delays[16].

This paper reviews various types of adders viz. ripple carry adder, carry save adder, carry skip adder , carry save adder , carry look-ahead adder , carry select adder. Further, a comparative study has been done on the

above mentioned adders, based on area, power and speed of operation, by employing a parallel and pipelined architecture for the implementation of each.

Addition of 8 numbers, each of 12 bits has been executed using the various adders. The paper has been organized as follows. The operation of each of the adders has been explained in detail in Section II.The architecture used for the implementation of the adders has been discussed in Section III. Comparison between the adders has been done based on the results obtained, which are discussed in SectionIV. Conclusion has been given in Section V.

II. Algorithms

Ripple carry adder is one of the most basic adder s and has a simple layout[5]. The basic unit of this adder is a simple full adder. A full adder has 3 inputs viz. 2 input numbers and input carry bit and 2 outputs viz. sum and carry, as shown in Fig.1.An n bit ripple carry adder will require n full adders for its implementation.

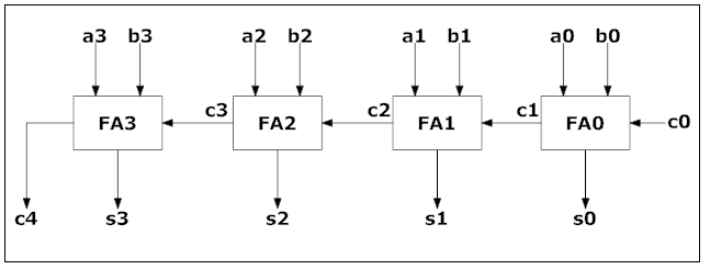

Fig. 1. Illustration of ripple carry adder

The working of a simple ripple carry adder has been demonstrated in Fig.1 to compute the addition of 2 4-bit numbers (a, b). c0 is the input carry bit.It consists of 4 full adder blocks FA3, FA2, FA1 and FA0. Addition is carried out starting from the lsb position (a0 +b0). Carry-out bit of FA0 is then as carry-in bit to the next higher FA block i.e. FA1.In short, the carry-out bit of eachith such full adder, say c_{i+1}, is given to the input carry bit, of the $(i+1)^{th}$ full adder in the next column. Thus, the carry bit propagates i.e. "ripples" through the chain of full adders and ultimately, the final sum is calculated. Hence, the name "ripple carry" adder.c4 is the resulting ultimate carry-out bit.

This type of adder is easy to implement due to easier connections between the adjacent full adders. However, the overall design is found to be relatively slow because of the delay involved in processing the carry out signal [17][8]. Also, the worst case propagation delay further goes on increasing as the number of input bits increases. The delay occurring when the intermediate carry bits travel through all the full adders in the chain is termed as the worst case delay. This a major drawback of ripple carry adders, which makes it a bad choice, especially for large input word sizes.

Carry save adders are useful whenever the addition of many words is desired. These types of addershave been proved as one of the most efficient adder designs not only in with the advantage of minimizing area but also increasing the speed of operation[9].Each block of carry save adder (CSA) is similar to a block of full adder. The only difference is that the carry-in input of full adder is replaced by a third number in the CSA block, as shown in Fig. 2.

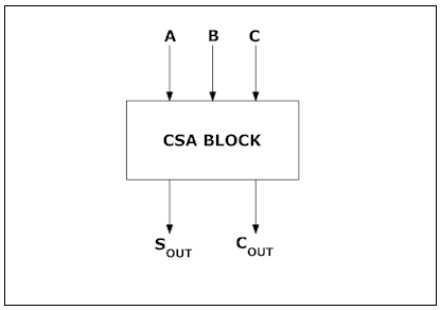

Fig. 2. Carry save adder block diagram

Basically, CSA block acts as a 3-2 reduction network. The addition of eight 12-bit numbers is considered as shown in Fig 3. Initially, a set of 3 numbers is taken together at a time; say N1, N2, N3 and N4, N5, N6. S1, C1 and S2, C2, each of 14 bits, are the sum and carry outputs obtained after computing the addition of the 2 sets of the 3 input numbers using 2 CSA blocks.

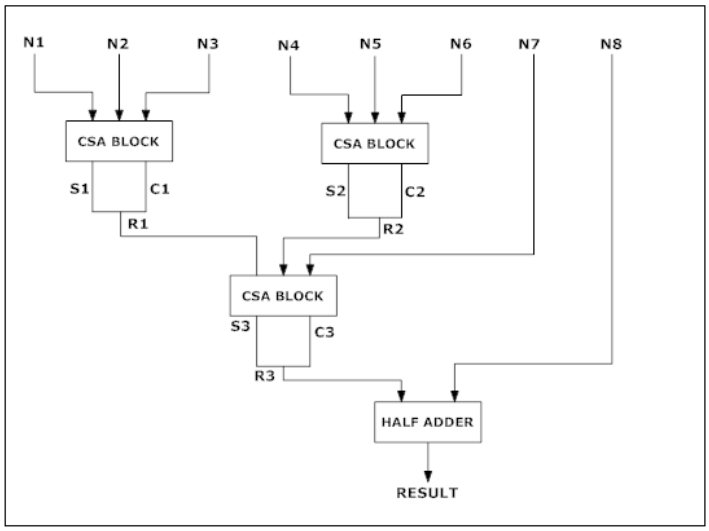

Fig. 3. Illustration of Carry save adder

Next, these intermediate sum and carry outputs are added together to obtain a two results, R1 and R2, as depicted in the figure. R1, R2 and N7 are given to another CSA block. The sum and carry obtained from this block is then combined with N8 in a similar manner described above and the final result is obtained. Owing to its high speed, these adders have been widely used in industry wherever fast arithmetic operations are required [18][10].

Another adder used to improve the speed of operation over the ripple carry adder is carry select adder. In this adder, two additions are performed simultaneously for the msb stage[11], one assuming carry input bit as 1 and the other as 0 [12].The implementation of this adder for addition of two 8 bit numbers, a and b, has been demonstrated in Fig. 4. In this type of adder, the 8 bit addition problem is split into twosub-problems, each requiring 4 bit addition. One sub-problem is of addition of lower 4 bits i.e. the lower nibble and the other, of the higher nibble.As shown in the Fig. 4,a total of 3 4-bit adders, have been used for the overall addition.Adder 1 in block A is used forthe addition of the lower nibble and adders 2 and 3 shown in block B,for that of the upper nibble. The carry delay will then depend on the output carry, generated by the output of adder used in block A. It is obvious that this output carry bit from stage1 can have only two possible values, viz. 0 or 1. Now, to minimize the time delay, two separate additions are carried out for calculating the result from upper nibble, one considering carry input as 0 in adder 2, and other considering carry input as 1 in adder 3. These additions are carried out simultaneously in block B, when the addition of the lower nibble isin progress. The carry output from block A is given to the select lines of a set of 2:1 multiplexers, which are shorted together as shown in Fig. 4.

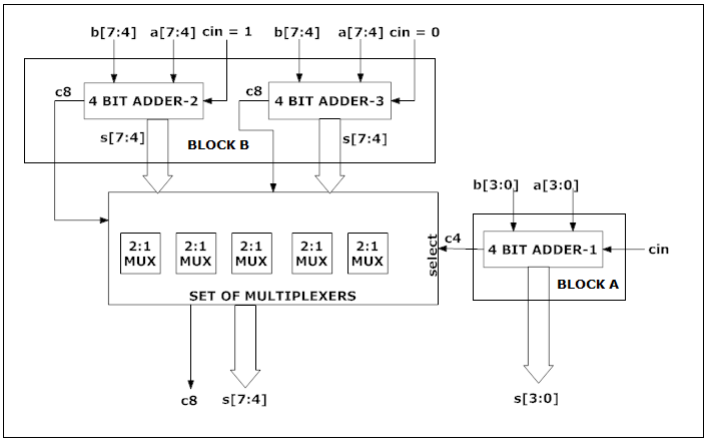

Fig. 4. Illustration of Carry select adder

5 multiplexers are required in this case. Both the sum and carry outputs of the adders 2 and 3 are given to the input of these multiplexers, and accordingly, the appropriate result from upper nibble is selected, on the basis of the signal available at the select lines of the multiplexers to determine the final sum. This design speeds up the process of addition by simultaneous calculation of the lower and upper nibbles[11],[13]. However, an

additional adder and a set of multiplexors are required, which is a major setback of this design. Carry select adders are beneficial to use only if speed is considered more important rather than area consumption.

Carry look-ahead (CLA) adders are another type of classic high speed adders which have been designed to overcome the speed limitation of ripple carry adders [14]. The delay introduced by the rippling effect of the carry bits is overcome to a large extent with the help of this adder.In general, CLA adders are used in various processing systems as they are relatively faster than ripple carry adder, but at the cost of large on-chip area utilization [5]. Two special signals – "generate" and "propagate" have been to implement the CLA adder to increase the speed of carry computation, instead of the conventional full adderblock[5]. The Boolean expressions for the generate and propagate terms are stated in Eqn. (1) and Eqn. (2) respectively.

$G_i = A_i.B_i$ (1)

$P_i = A_i \Theta B_i$ (2)

Also, the sum and carry-out terms are calculated, based on thegenerate (G_i) and propagate (P_i) terms, as shown in Eqn. (3) and Eqn. (4) respectively.

$C_{i+1} = G_i + P_i.C_i$ (3)

$S_i = P_i \Theta C_i$. (4)

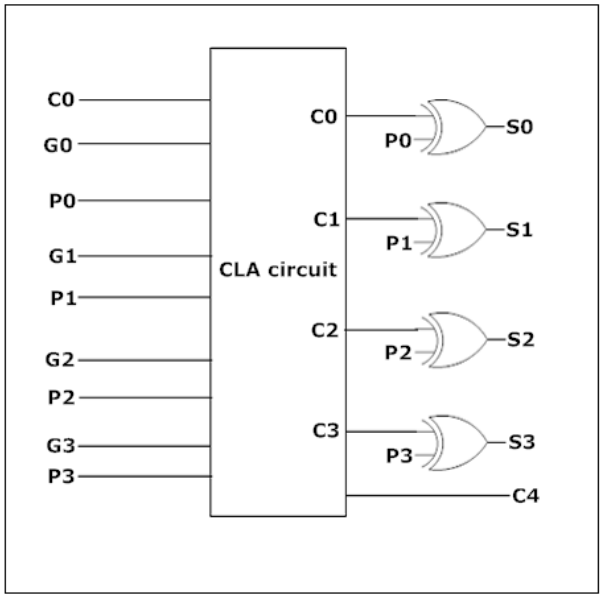

Fig. 5. Illustration of Carry look ahead adder

In general, the main objective of the CLA adder is to calculate the values of G_i and G_i for each and every input bit(A_i and B_i). The sum (S_i) and carry-out (C_{i+1}) bits are then calculated based on the above obtained values of G_i,P_i and the input carry bits (c_i). Addition of two 4-bit numbers (say, A[3:0] and B[3:0]) have been considered below,as an instance to explain the working of this particular algorithm, with the aid of block diagram shown in Fig. 5. The generate (G0,G1,G2,G3) and propagate (P0,P1,P2,P3) terms are obtained using "AND" and "XOR" operations, as mentioned above. C0 is considered as the input carry bit. The carry out terms are calculated as given below in Eqns. (5), (6), (7) and (8).

$C1 = G0 + P0.C0$ (5)

$C2 = G1 + P1.C1$ (6)

$C3 = G2 + P2.C2$ (7)

$C4 = G3 + P3.C3$ (8)

On substituting (5) in (6) and simplifying, the expression for C2 can be modified as shown below in Eqn. (9).

$C2 = G1 + P1.(G0 + P0.C0)$. (9)

Similarly, expressions forC3 and C4 can be modified in a similar manner. It is clear from these equations that all the carry-out bits can be made available simultaneously, unlike in the case of ripple carry adder where each carry-out bit depends on the carry-out bit of its preceding full adder in the LSB position. This fact itself is responsible for faster operation of CLA adder.Finally, the resultant sum can be obtained as shown in Fig. 5, based on the expressions already discussed above.

In a ripple carry adder, the computation of the final carry-out bit depends on the carry-out bit of each of the full adder blocks. Also, the carry out bit of each full adder block depends on the incoming carry bit. This dependency can be avoided by the use of carry skip adder(CSK) and thus the speed of operation can be improved by a large extent, with lesser requirement of area and power as well[15][16]. The algorithm has been illustrated in Fig. 6 for the addition of 2 4-bit numbers. The generate (G_i) and propagate (P_i) terms are calculated

in the same fashion, as in the case of CLA adder. Each "adder" block in Fig. 6 involves the use of necessary circuitry to generate the sum and carry-out bits, by making use of the generate and propagate terms. The Boolean expressions for sum and carry-out bits of each such adder block are same as those in CLA adder. The "carry-skip"circuit is the distinguishing feature of CSK adder, which increases the operating speed, mainly with the help of block propagate signal. The Boolean expressions for the block propagate signal (BP) and carry-out bit (COUT) are mentioned below, in Eqn. (10) and Eqn. (11) respectively.

$BP = P0.P1.P2.P3$ (10)

$COUT = C4 + (BP).C0$ (11)

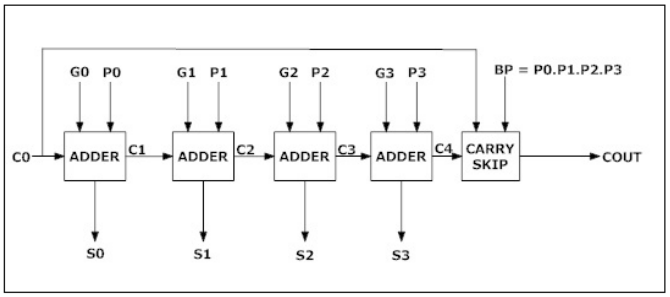

Fig. 6. Illustration of Carry skip adder

Thus, when the BP signal is 1, the incoming carry is directly given to the next block as COUT by "skipping" or bypassing the 4 adder blocks depicted in the figure. If the BP signal is 0, then the carry-out term COUTis computed by the normal procedure. The "carry-skip" block can be simply implemented by a 2:1 mux, using the BP signal on the select line and C0,C4 inputs to the multiplexer. It can also be implemented otherwise with the help of basic gates, to obtain the desired Boolean expression.

III. Architecture

Any process can be implemented in either two ways viz. serial or parallel manner. In serial computation which is shown in Fig. 7, the complete process is executed in a single clock cycle itself. Hence, care needs to be taken to ensure that the duration of this clock cycle is large enough to facilitate the execution of the process within a single clock cycle. This reduces the speed of the computation and is a major drawback of serial architecture[19].

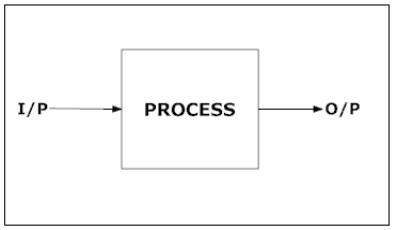

Fig. 7. Illustration of serial computation

Parallel architecture involves relatively large area as compared to serial architecture but, it greatly boosts the speed of operation of the overall system. In parallel architecture, the entire process is divided into various sub-processes as illustrated in Fig. 8.

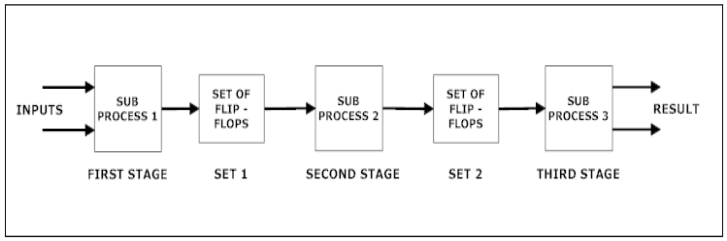

Fig. 8. Illustration of parallel and pipelined architecture

Each of the sub-processes is active and execute in parallel. The sub-process 1 receives a certain set of inputs, which gets computed in the first clock cycle and transfers the result to the succeeding flip-flop for temporary storage, until the arrival of the next clock pulse. At the immediate next clock pulse, the outputs of the

flip-flopsare transferred as inputs to the sub-process 2, and the computation is initiated in this stage. However, the first stage in this point is not idle and takes up the next set of inputs to compute. If the process is divided into n stages, then neither of the stage is idle after n-1 clock pulses. Thus, the parallel and pipelined approach reduces the clock time period,thereby increasing the operating speed of the overall system [19].Hence, parallel and pipelined architecture has been used to implement each of the above adders, having different number of pipeline stages ranging from 3 for ripple carry adder to 12 for carry select adder.

IV. Results

The synthesis results of the above described adders have been obtained using Xilinx ISE Design Suite (version 13.2) on Spartan 3E FPGA. The criticalfactors to be compared for the different adders are area, power and speed. Area or device utilization is characterised by the number of slices utilized on the FPGA to implement the design. Speed can be commented upon by knowing the maximum frequency at which the adder can work. Power simply refers to the amount of power consumed by a particular adder, when it is implemented on the development board mentioned above. The various adders mentioned in this paper have been implemented using a parallel and pipelined architecture and can be easily compared with the help of the following graphs, based on the above 3 factors.

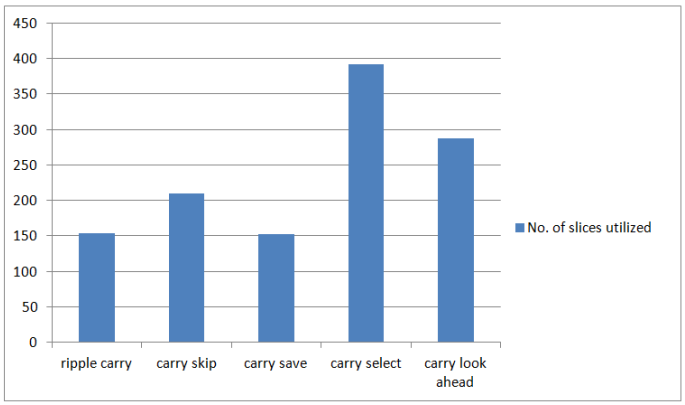

Fig. 9. Number of slices utilized

From the bar graph depicted in Fig. 9, it can be seen that carry save adder occupies the lowest number of slices. i.e. 154. Ripple carry adder occupies nearly the same number of slices (154) as that of carry save adder. The carry skip adder and carry look-ahead adders occupy relative much more area. They occupy 210 and 287 slices respectively. The carry select adder occupies the largest number of slices viz. 392.

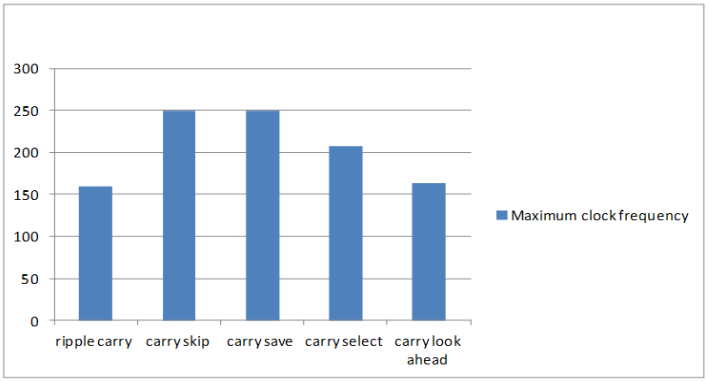

Fig. 10. Maximum clock frequency (MHz)

The bar graph illustrated in Fig.10 showcases the comparison of the speed of the operation of the various adders. From the graph, it is clear that both carry skip adder and carry save adder can provide the maximum clock speed of 249.186 MHz. Next to these adders are carry select adder and carry look-ahead adders, with maximum clock frequency of 206.612MHzand 164.123 MHzrespectively. The ripple carry adder provides the worst speed of 160.102MHz.

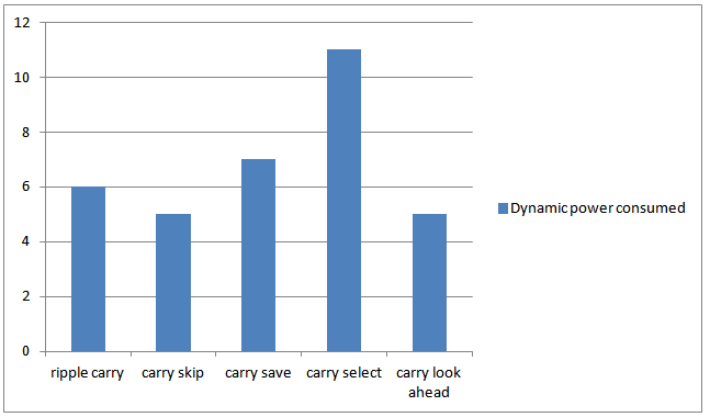

Fig. 11. Power consumed (in mW)

The bar graph shown in Fig. 11 compares the dynamic power consumed by the different types of adders in milliWatts (mW).Carry skip adder and Carry look-ahead adder consume a minimum power of 86 mW each. Next to them are ripple carry adder and carry save adders, which consume 87 and 88 mW respectively. The Carry select adder consumes a maximum dynamic power of 92 mW amongst all the adders discussed above.

V. Conclusion

From the results observed in the previous section, it can be commented that carry skip adder performs better than the other adders as far as the maximum operating frequency and the power consumption are concerned. The carry save adder has a speed equivalent to that of the carry skip adder and occupies the minimum area while consuming a moderate power. Also the ripple carry adder shares the advantage of occupying minimal area, but at the cost of low speed of operation.

Thus, it can be concluded that a trade-off exists between the area occupied, speed of operation and the power consumption in the various adder designs. Any adder can be given a preference considering which parameter is of epitome. It is now the designers' discretion to choose an adder to suit the requirements of the architecture. Thus, a suitable type of adder has to be chosen, by keeping in view the requirements of the overall system, in which the adder has to be implemented.

References

[1]. R.P.P. Singh, Parveen Kumar, Balwinder Singh, "Performance Analysis of fast Adders using VHDL", 2009 International Conference on Advances in Recent Technologies in Communication and Computing, October 2009,ISBN: 978-0-7695-3845-7,pp. 189 – 193.

[2]. Mariano Aguirre, Monico Linares, "An Alternative Logic Approach to Implement High-Speed Low-Power Full Adder Cells", SBCCI '05, Proceedings of the 18th annual symposium on Integrated circuits and system design,September 4-7 2005, ISBN: 1-59593-174-0, pp. 166 – 171.

[3]. LuJunming, Shu Yan, Lin Zhenghui and Wang, Ling, "A Novel 10-Transisitor Low-Power High-speed Full Adder Cell", 6th International Conference on Solid-State and Integrated Circuit Technology, October 22-25 2001, ISBN: 0-7803-6520-8, Volume 2, pp. 1155 - 1158.

[4]. Ayman A. Fayed and Magdy A. Bayoumi, "A Low Power 10-Transistor Full Adder Cell for Embedded Architectures",The 2001 IEEE International Symposium on Circuits and Systems, May 6-9 2001, ISBN: 0-7803-6685-9, Volume 4, pp.226 – 229.

[5]. Fatemeh Karami H,Ali K. Horestani, "New Structure for Adder with Improved Speed, Area and Power", International Conference on Networked Embedded Systems for Enterprise Applications, December 8-9 2011, ISBN: 978-1-4673-0495-5, pp. 1-6.

[6]. Habib Ghasemizadeh Tamar et al, "High Speed Area Reduced 64-bit Static Hybrid Carry-Lookahead/Carry-Select Adder", International Conference on Electronics, Circuits and Systems, December 11-14 2011, ISBN: 978-1-4577-1846-5, pp. 460 – 463.

[7]. Kuldeep Rawat, Tarek Darwish. and Magdy Bayoumi, "A Low Power and reduced area carry select adder", Midwest Symposium on Circuits and Systems, August 4-7 2002, ISBN: 0-7803-7523-8, Volume 1,pp. 467 – 470.

[8]. Suhaili S et al, "An Investigation of Signed bit Adder with VHDL", International Symposium on Informtion Technology, August 26-28 2008, ISBN: 978-1-4244-2328-6, Volume 4, pp. 1-6.

[9]. Youngtae Kim and Taewhan Kim, "Accurate Exploration of Timing and Area Trade-offs in Arithmetic Optimization using Carry-Save-Adders", Asia and South Pacific Design Automation Conference, January 30 2001 - February 2 2001, ISBN: 0-7803-6633-6, pp. 622 – 627.

[10]. Taewhan Kim, William Jao, and Steve Tjiang, "Circuit Optimization Using Carry–Save–Adder Cells", IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, October 1998, Volume 17, Issue 10, pp. 974 – 984.

[11]. G.A. Ruiz, M. Granda, "An area-efficient static CMOS carry-select adder based on a compact carry look-ahead unit", Microelectronics Journal 35, September 2 2004, pp. 939–944.

[12]. J. Hennesey and D.A.Patterson, Computer Architecture – A Quantitative Approach, San Mateo, CA: Morgan Kaufmann, 1990, Appendix A.

[13]. Hiroyuki Morinaka et al, "A 64bit Carry Look-ahead CMOS Adder using Modified Carry Select", IEEE 1995 Custom Integrated Circuits Conference, May 1-4 1995, ISBN: 0-7803-2584-2, pp. 585 – 588.

[14]. A. Weinberger and J.L.Smith,"A logic for high speed Addition", National Bureau of Standards, Circulation 591, p.3-12, 1958.

[15]. The Origins ofDigital Computers**,** 3rd ed., B. Randell, Ed. New York:Springer-Verlag, 1982.
[16]. Min Cha, Earl E. Swartzlander, "Modified Carry Skip Adder for Reducing First Block Delay",43rd IEEE Midwest Symp. on Circuits and Systems,August 2000, ISBN: 0-7803-6475-9, Volume 1, pp. 346 – 348.
[17]. K. Gupta, N. Pandey, M. Gupta, "A Novel Active Shunt-Peaked MCML-based High Speed Four-Bit Ripple-CarryAdder",International Conference on Computer & Communication Technology, September 17-19 2010, ISBN: 978-1-4244-9033-2, pp. 285 – 289.
[18]. Junhyung Umand Taewhan Kim, "An Optimal Allocation ofCarry-Save-Adders in Arithmetic Circuits", IEEE Transactions on Computers, March 2001, Volume 50, Issue 3,ISSN: 0018-9340, pp. 215-233.
[19]. Piyush S. Kasat, Devendra S. Bilaye et al, "Multiplication Algorithms for VLSI- A Review", International Journal on Computer Science and Engineering, ISSN: 0975-3397, Volume 4, Issue 11, November 2012, pp. 1761 – 1765.

Reju John" Addition Algorithms for VLSI – A review" IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) , vol. 9, no. 2, 2019, pp. 06-13.