JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

HIGH-SPEED AREA-EFFICIENT VLSI ARCHITECTURE USING CARRY SKIP ADDER

Chandra Shekara Reddy P, D. Indrasena Reddy
PG Scholar, Asst. Professor
SKD Engineering College

Abstract

The aim of this paper is to determine the Addition. It is one of the most basic operations performed in all computing units, including microprocessors and digital signal processors. It is also a basic unit utilized in various complicated algorithms of multiplication and division. Efficient implementation of an adder circuit usually revolves around reducing the cost to propagate the carry between successive bit positions. Multi-operand adders are important arithmetic design blocks especially in the addition of partial products of hardware multipliers. The multi-operand adders (MOAs) are widely used in the modern low-power and high-speed portable very-large-scale integration systems for image and signal processing applications such as digital filters, transforms, convolution neural network architecture. Hence, a new high-speed and area efficient adder architecture is proposed using pre-compute bitwise addition followed by carry prefix computation logic to perform the three-operand binary addition that consumes substantially less area, low power and drastically reduces the adder delay. Further, this project is enhanced by using Modified carry bypass adder to further reduce more density and latency constraints. Modified carry skip adder introduces simple and low complex carry skip logic to reduce parameters constraints. In this proposal work, designed binary tree adder (BTA) is analyzed to find the possibilities for area minimization. Based on the analysis, critical path of carry is taken into the new logic implementation and the corresponding design of CSKP are proposed for the BTA.

Keywords: *Multi-operand adders (MOAs), Binary Tree Adder (BTA), Carry Skip Adder (CSKP).*

I. INTRODUCTION

Besides technological scaling, advances in the field of computer architecture have also contributed to the exponential growth in performance of digital computer hardware. The flip-side of the rising processor performance is an unprecedented increase in hardware and software complexity. Increasing complexity leads to high development costs, difficulty with testability and verifiability, and less adaptability. The challenge in front of computer designers is therefore to opt for simpler, robust, and easily certifiable circuits. Computer arithmetic, here plays a key role aiding computer architects with this challenge. It is one of the oldest sub-fields of computer architecture. The bulk of hardware in earlier computers resided in the accumulator and other arithmetic/logic circuits. Successful operation of computer arithmetic circuits was taken for granted and high performance of these circuits has been routinely expected. This context has been changing due to various reasons. First, at very high clock rates, the interfaces between arithmetic circuits and the rest of the processor become critical. Arithmetic circuits can no longer be designed and verified in isolation. Rather an integrated design optimization is required. Second, optimizing arithmetic circuits to meet the design goals by taking advantage of the strengths of new technologies, and making them tolerant to the weakness, requires a re-

examination of existing design paradigms. Finally, incorporation of higher-level arithmetic primitives into hardware makes the design, optimization and verification efforts highly complex and interrelated. The core of every microprocessor, digital signal processor (DSP), and data processing application-specific integrated circuit (ASIC) is its data path. With respect to the most important design criteria; critical delay, chip size, and power dissipation, the data path is a crucial circuit component. The data path comprises of various arithmetic units, such as comparators, adders, and multiplier [4]. The basis of every complex arithmetic operation is binary addition. Hence, it can be concluded, that binary addition is one of the most important arithmetic operation. The hardware implementation of an adder becomes even more critical due to the expensive carry-propagation step, the evaluation time of which is dependent on the operand word length. The efficient implementation of the addition operation in an integrated circuit is a key problem in VLSI design [8]. Productivity in ASIC design is constantly improved by the use of cell-based design techniques – such as standard cells, gate arrays, and field programmable gate arrays (FPGA), and low-level and high-level hardware synthesis [13]. This asks for adder architectures which result in efficient cell-based circuit realizations which can easily be synthesized. Furthermore, they should provide enough flexibility in order to accommodate custom timing and area constraints as well as to allow the implementation of customized adders. The tasks of a VLSI chip are the processing of data and the control of internal or external system components. This is typically done by algorithms which are based on logic and arithmetic operations on data items [10]. Applications of arithmetic operations in integrated circuits are manifold. Microprocessors and DSPs typically contain adders and multipliers in their data path.. Adders, incrementers/decrementers, and comparators are often used for address calculation and flag generation purposes. ASICs use arithmetic units for the same purposes. Depending on their application, they may even require dedicated circuit components for special arithmetic operators, such as for finite field arithmetic used in cryptography, error correction coding, and signal processing.

II. LITERATURE SURVEY

Kogge P, Stone H proposed A parallel calculation for the productive arrangement of a general class Recurrence relations . A m^{th}-request repeat issue is characterized as the calculation of the arrangement x1, x2, ..., XN, where xi = fi(xi-1, ..., xi-m) for some capacity fi. This paper utilizes a system called recursive multiplying in a calculation for illuminating a huge class of repeat issues on parallel PCs, for example, the Iliac IV. Recursive multiplying includes the part of the calculation of a capacity into two similarly complex sub functions whose assessment can be performed at the same time in two separate processors. Progressive part of every one of these subfunctions spreads the calculation over more processors. This calculation can be connected to any repeat condition of the structure xi = f(bi, g(ai, xi-1)) where f and g are capacities that fulfill certain distributive and affiliated like properties. In spite of the fact that this repeat is first request, all straight mth-request repeat conditions can be thrown into this structure. Reasonable applications incorporate direct repeat conditions, polynomial assessment, a few nonlinear issues, the assurance of the greatest or least of N numbers, and the arrangement of tridiagonal straight conditions. The subsequent calculation registers the whole arrangement x1, ..., xN in time corresponding to [log2 N] on a PC with N-overlap parallelism. On a sequential PC, calculation time is relative to N.

III. EXISTING TECHNIQUE:

Pre-Compute Bitwise Addition Followed By Carry Prefix Computation.

This method presents a new adder technique and its VLSI architecture to perform the three-operand addition in modular arithmetic. The proposed adder technique is a parallel prefix adder. However, it has four-stage structures instead three-stage structures in prefix adder to compute the addition of three binary input operands

such as bit-addition logic, base logic, PG (propagate and generate) logic and sum logic. The logical expression of all these four stages are defined as follows,

Stage-1: Bit Addition Logic:

$S_i' = a_i \oplus b_i \oplus c_i$,
$Cy_i = a_i \cdot b_i + b_i \cdot c_i + c_i \cdot a_i$

Stage-2: Base Logic:

$G_{i:i} = G_i = S_i' \cdot cy_{i-1}$, $G_{0:0} = G_0 = S_0' \cdot C_{in}$,
$P_{i:i} = P_i = S_i' \oplus Cy_{i-1}$, $P_{0:0} = P_0 = S_0' \oplus C_{in}$,

Stage-3: PG (Generate and Propagate) Logic:

$G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j}$,
$P_{i:j} = P_{i:k} \cdot P_{k-1:j}$

Stage-4: Sum Logic:

$S_i = (P_i \oplus G_{i-1:0})$, $S_0 = P_0$, $C_{out} = G_{n:0}$

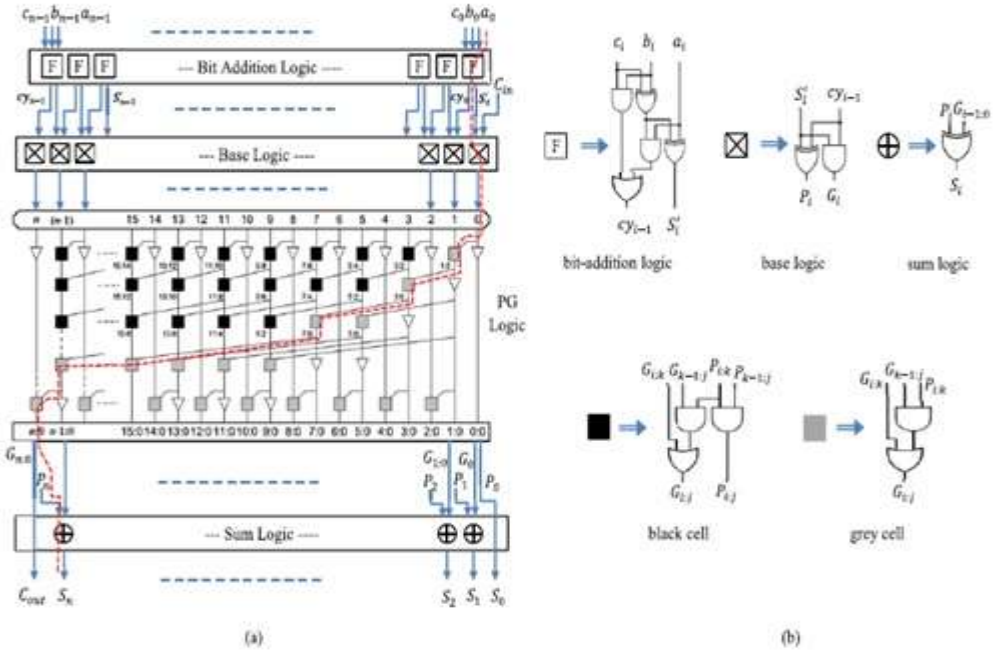

Fig 1.Three-operand adder; (a) First order VLSI architecture, (b) Logical diagram of bit addition, base logic, sum logic, black-cell and grey-cell.

Design Methodology:

The proposed VLSI architecture of the three-operand binary adder and its internal structure is shown in Fig. The new adder technique performs the addition of three n-bit binary inputs in four different stages. In the first stage (bit-addition logic), the bitwise addition of three n-bit binary input operands is performed with the array of full adders, and each full adder computes "sum (S_ i)" and "carry (cyi)" signals as highlighted in Fig. 1(a). The logical expressions for computing sum (S_ i) and carry (cyi) signals are defined in Stage-1, and the logical diagram of the bit-addition logic is shown in Fig. 1(b). In the first stage, the output signal "sum (S_ i)" bit of current full adder and the output signal "carry" bit of its right-adjacent full adder are used together to compute the generate (Gi) and propagate (Pi) signals in the second stage (base logic). The computation of G_i and P_i signals are represented by the "squared saltire-cell" as shown in Fig. 1(a) and there are n+1 number of saltire-cells in the base logic stage. The logic diagram of the saltire-cell is shown in Fig. 1(b), and it is realized by the logical expression,

Drawbacks:

More complexity with huge wiring
More gate density
Still large combinational path delay with maximum arrival time.
Less reliability

IV. PROPOSED METHOD

4.1 Carry Skip Adder (CSKA)

The carry-skip adder reduces the time needed to propagate the carry by skipping over groups of consecutive adder stages, is known to be comparable in speed to the carry look-ahead technique while it uses less logic area and less power.

4.1.1 Uniform sized adder:

A carry skip adder divides the words to be added into groups of equal size of k-bits. Carry Propagate pi signals may be used within a group of bits to accelerate the carry propagation. If all the pi signals within the group are pi=1, carry bypasses the entire group as shown in figure.

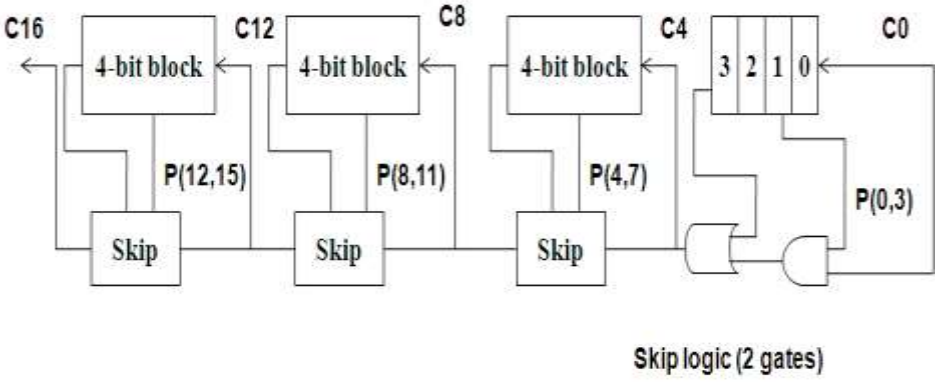

Fig 4.1: Carry skip adder

P = pi * pi+1 * pi+2 *… pi+k

In this way, the delay is reduced as compared to ripple carry adder. The worst-case carry propagation delay in a N-bit carry skip adder with fixed block width b, assuming that one stage of ripple has the same delay as one skip, can be derived:

TCSKA = (b -1)+0.5+(N/b-2)+(b -1) = 2b + N/b – 3.5 Stages

Block width tremendously affects the latency of adder. Latency is directly proportional to block width. More number of blocks means block width is less, hence more delay.

4.2 Variable Block Adder

The idea behind Variable Block Adder (VBA) is to minimize the critical path delay in the carry chain of a carry skip adder, while allowing the groups to take different sizes. In case of carry skip adder, such condition will result in more number of skips between stages.

Such an adder design is called variable block design, which is tremendously used to fasten the speed of adder. In the variable block carry skip adder design, we divided a 32-bit adder into 4 blocks or groups. The bit widths of groups are taken as: First block is of 4 bits, second is of 6 bits, third is 18 bit wide and the last group consist of most significant 4 bits.

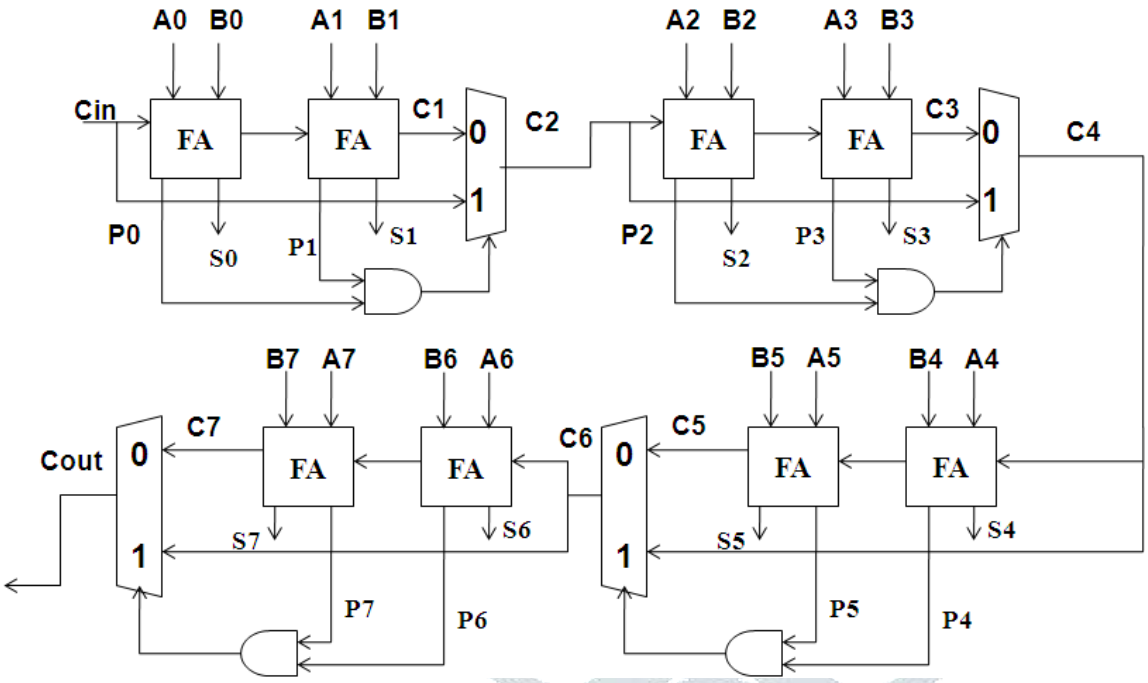

Fig 4.2 : Architectural block of 8-bit Carry skip adder

The carry skip adder provides a compromise between a ripple carry adder and a CSLA adder. The carry skip adder divides the words to be added into blocks. Within each block, ripple carry is used to produce the sum bit and the carry. The Carry Skip Adder reduces the delay due to the carry computation i.e. by skipping over groups of consecutive adder stages.

- If each Ai # Bi in a group, then we do not need to compute the new value of Ci+1 for that block; the carry-in of the block can be propagated directly to the next block.
- If Ai = Bi = 1 for some i in the group, a carry is generated which may be propagated up to the output of that group.
- If Ai = Bi = 0, a carry will not be propagated by that bit location.

4.3 CONVENTIONAL CSKA:

The conventional structure of the CSKA consists of stages containing chain of full adders (FAs) (RCA block) and 2:1 multiplexer (carry skip logic). The RCA blocks are connected to each other through 2:1

multiplexers, which can be placed into one or more level structures [19]. The CSKA configuration (i.e., the number of the FAs per stage) has a great impact on the speed of this type of adder [23]. Many methods have been suggested for finding the optimum number of the FAs [18]–[26]. The techniques presented in [19]–[24] make use of VSSs to minimize the delay of adders based on a single level carry skip logic. In [25], some methods to increase the speed of the multilevel CSKAs are proposed. The techniques, however, cause area and power increase considerably and less regular layout. The design of a static CMOS CSKA where the stages of the CSKA have a variable sizes was suggested in [18]. In addition, to lower the propagation delay of the adder, in each stage, the carry look-ahead logics were utilized. Again, it had a complex layout as well as large power consumption and area usage. In addition, the design approach, which was presented only for the 32-bit adder, was not general to be applied for structures with different bits lengths. Alioto and Palumbo [19] propose a simple strategy for the design of a single-level CSKA. The method is based on the VSS technique where the near-optimal numbers of the FAs are determined based on the skip time (delay of the multiplexer), and the ripple time (the time required by a carry to ripple through a FA). The goal of this method is to decrease the critical path delay by considering a non-integer ratio of the skip time to the ripple time on contrary to most of the previous works, which considered an integer ratio [17], [20]. In all of the works reviewed so far, the focus was on the speed, while the power consumption and area usage of the CSKAs were not considered. Even for the speed, the delay of skip logics, which are based on multiplexers and form a large part of the adder critical path delay [19], has not been reduced.

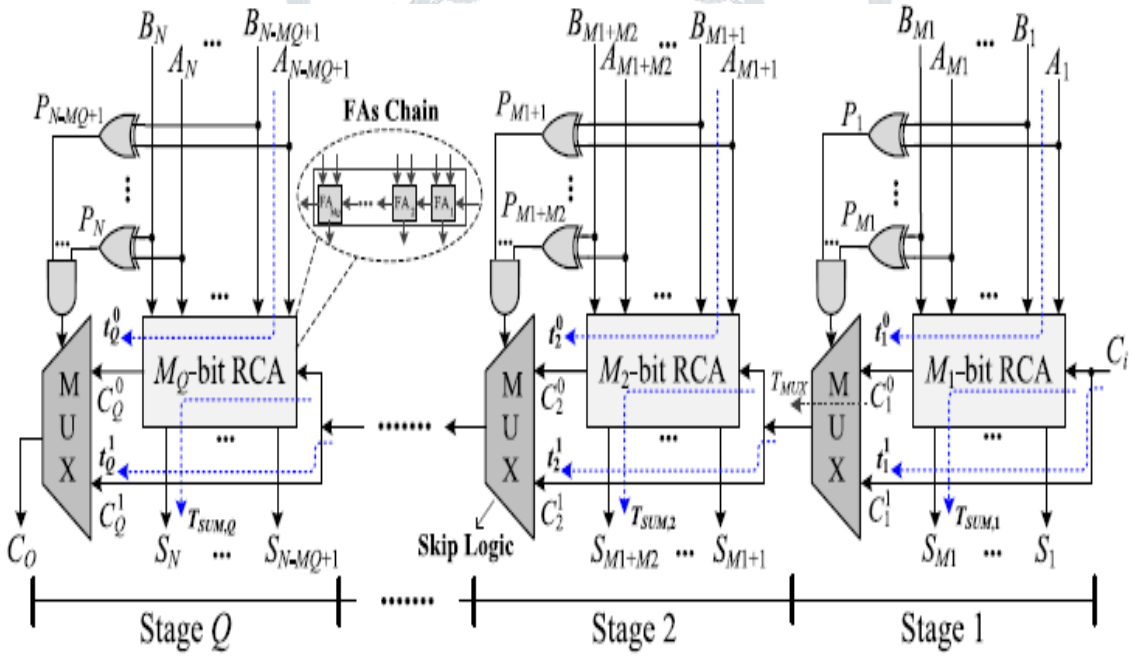

Fig 4.3 : Conventional structure of the CSKA

4.4 MODIFIED CARRY SKIP ADDER

The structure is based on combining the concatenation and the incrementation schemes [13] with the Conv-CSKA structure, and hence, is denoted by CI-CSKA. It provides us with the ability to use simpler carry skip logics. The logic replaces 2:1 multiplexers by AOI/OAI compound gates. The gates, which consist of fewer transistors, have lower delay, area, and smaller power consumption compared with those of the 2:1 multiplexer [7]. Note that, in this structure, as the carry propagates through the skip logics, it becomes complemented. Therefore, at the output of the skip logic of even stages, the complement of the carry is generated. The structure

has a considerable lower propagation delay with a slightly smaller area compared with those of the conventional one. Note that while the power consumptions of the AOI (or OAI) gate are smaller than that of the multiplexer, the power consumption of the proposed CI-CSKA is a little more than that of the conventional one. This is due to the increase in the number of the gates, which imposes a higher wiring capacitance (in the noncritical paths). Now, we describe the internal structure of the proposed CI-CSKA shown in Fig. 2 in more detail. The adder contains two N bits inputs, A and B, and Q stages. Each stage consists of an RCA block with the size of Mj (j = 1, . . . , Q). In this structure, the carry input of all the RCA blocks, except for the first block which is Ci , is zero (concatenation of the RCA blocks). Therefore, all the blocks execute their jobs simultaneously. In this structure, when the first block computes the summation of its corresponding input bits (i.e., SM1, . . . , S1), and C1, the other blocks simultaneously compute the intermediate results [i.e., {ZK j+Mj , . . . , ZK j+2, ZK j+1} for K j = _j−1 r=1 Mr (j = 2, . . . , Q)], and also Cj signals. In the proposed structure, the first stage has only one block, which is RCA. The stages 2 to Q consist of two blocks of RCA and incrementation. The incrementation block uses the intermediate results generated by the RCA block and the carry output of the previous stage to calculate the final summation of the stage.

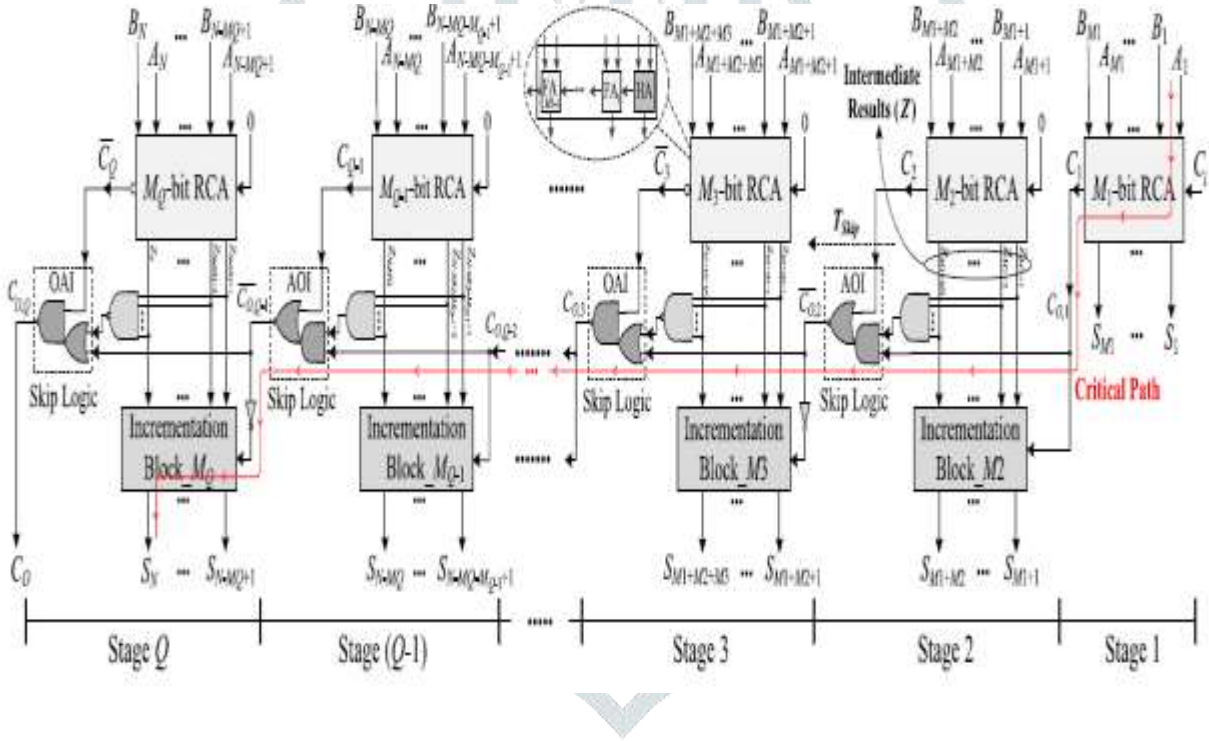

Fig 4.4 : CI-CSKA structure.

V. SIMULATION RESULTS

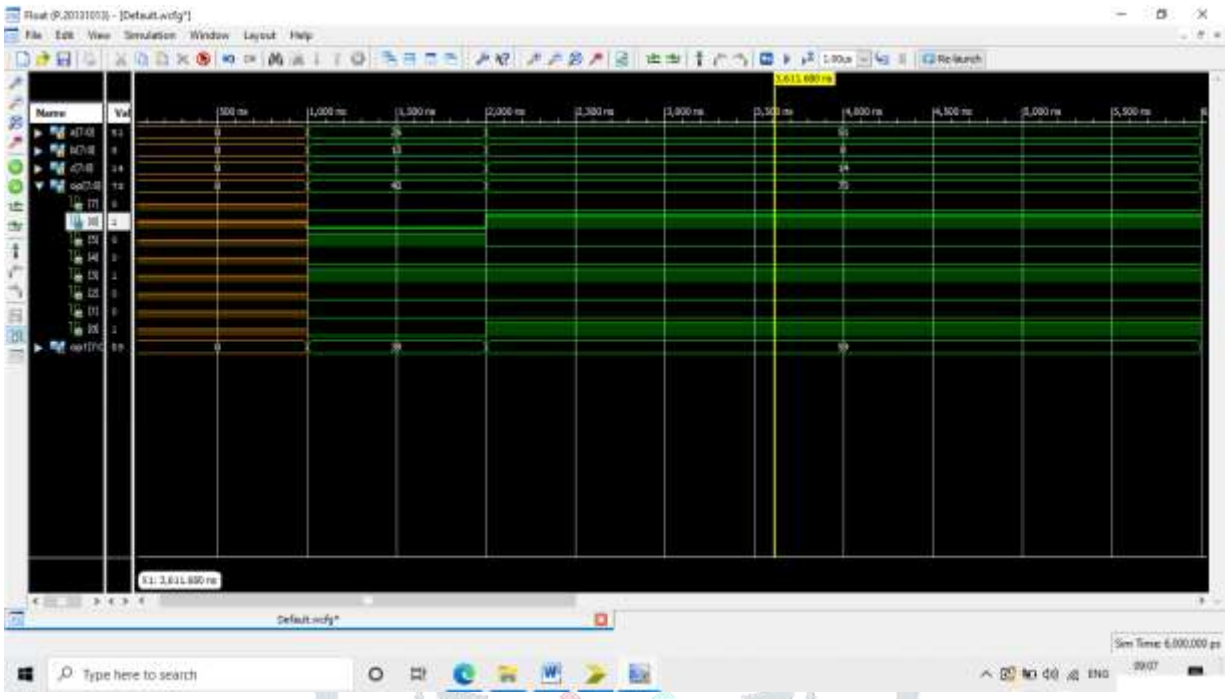

Fig 5 : Output Waveform.

Fig 5 shows the output waveform of proposed adder where we performed addition of 16 bit input data of a b and cin and gives the output is 16 bit sum and Cout

Table 1 Comparison Table

SYSTEM	SPEED(hz)	Delay(ns)
Existing method	45.5 mhz	21.848 ns
Proposed method	73.8 mhz	13.546 ns

VI. CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION:

In this paper, a high-speed area-efficient adder technique and its VLSI architecture is proposed to perform the three operand binary addition for efficient computation. The proposed three-operand adder technique is a parallel prefix adder that uses four-stage structures to compute the addition of three input operands. The novelty of this proposed architecture is the reduction of delay and area in the prefix computation stages in PG logic and bit-addition logic. As an extension of this concept, a static CMOS CSKA structure called CI-CSKA was proposed, which exhibits a higher speed and lower energy consumption compared with those of the conventional one. The speed enhancement was achieved by modifying the structure through the concatenation

and incrementation techniques. In addition, AOI and OAI compound gates were exploited for the carry skip logics.

6.2 FUTURE SCOPE:

In the future work, further research will be done by applying dual-path FP architecture to the three-operand FP adder and using other redundant FP representations. Use of improved techniques in the termination phase of the design (i.e., redundant LZD, normalization, and rounding) would lead to faster architectures, though higher area costs are expected.

VII. REFERENCES

[1] S. Yu and E. E. Swartzlander, "DCT implementation with distributed arithmetic", IEEE Transactions on Computers, vol. 50, no. 9, pp. 985–991, Sept. 2001.

[2] T.-S. Chang, C. Chen, and C.-W. Jen, "New distributed arithmetic algorithm and its application to IDCT," IEE Proceedings Circuits, Devices and Systems, vol. 146, no. 4, pp. 159–163, Aug. 1999.

[3] T.-S. Chang and C.-W. Jen, "Hardware-efficient implementations for discrete function transforms using LUT-based FPGAs," IEE Proceedings Circuits, Devices and Systems, vol.146, no. 6, pp. 309–315, Nov. 1999.

[4] F. de Dinechin, H. D. Nguyen and B. Pasca, Pipelined FPGA Adders, LIP Research Report no. ensl00475780, Apr. 2010.

[5] J. Hormigo, M. Ortiz, F. Quiles, F. J. Jaime, J. Villalba and E.L. Zapata, Efficient Implementation of CarrySave Adders in FPGAs, 20th IEEE international Conference on Application-Specific Systems, Architectures and Processors, pp. 207–210, Jul. 2009.

[6] P. M. Martinez, V. Javier, and B. Eduardo, On the design of FPGA-based Multioperand Pipeline Adders, XII Design of Circuits and Integrated System Conference, 1997.

[7] H. Parandeh-Afshar, P. Brisk, and P. Ienne, "Efficient Synthesis of Compressor Trees on FPGAs," in Asia and South Pacific Design Automation Conference (ASPDAC). IEEE, 2008, pp. 138–143. [8] Xilinx Inc., Virtex-6 User Guide, 2009, http://www.xilinx.com/. [9] S. Xing and W. H. Yu, FPGA Adders: Performance Evaluation and Optimal Design, IEEE Design and Test of Computers, vol. 15, no. 1, pp. 24–29, Jan.- Mar. 1998.

[10] R. D. Kenney and M. J. Schulte, "High-Speed Multioperand Decimal Adders", IEEE Transactions on Computers, vol. 54, no. 8, pp. 953-963, Aug. 2005.

[11] J. Villalba, J. Hormigo, J. M. Prades and E. L. Zapata, "On–line Multioperand Addition Based on On–line Full Adders∗", in Proc. Int. Conf. on ApplicationSpecific Systems, Architecture Processors (ASAP'05), pp. 322-327, 2005

[12] M. Ortiz, F. Quiles, J. Hormigo, F. J. Jaime, J. Villalba, and E. L. Zapata, "Efficient Implementation of CarrySave Adders in FPGAs," in IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP), 2009, pp. 207– 210.

[13] W. Kamp, A. Bainbridge-Smith, and M. Hayes, "Efficient Implementation of Fast Redundant Number Adders for Long Word- Lengths in FPGAs," in 2009 International Conference on Field- Programmable Technology (FPT). IEEE, 2009, pp. 239–246. [14] J. Hormigo, J. Villalba, and E. L. Zapata, "Multioperand Redundant Adders on FPGAs," submitted to IEEE Transactions on Computers, vol. 62, no. 10, pp. 2013– 2025, 2013.

[15] S. D. Thabah; M. Sonowal and P. Saha ,"EXPERIMENTAL STUDIES ON MULTI-OPERAND ADDERS", INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS VOL. 10, NO. 2, JUNE 2017

[16] S. Singh and D. Waxman, "Multiple Operand Addition and Multiplication", IEEE Transactions on Computers, vol. C-22, no. 2, pp. 113-120, Feb. 1973. [17] C. Wallace, "A Suggestion for a Fast Multiplier," IEEE Transactions on Electronic Computers, no. 1, pp. 14– 17, 1964.

[18] L. Dadda, "Some Schemes For Parallel Multipliers," Alta Frequenza, vol. 45, no. 5, pp. 349–356, 1965.

[19] A. Omondi and B. Premkumar, Residue Number Systems: Theory and Implementation. Imperial College Press, 2007.

[20] A. R. Meo, "Arithmetic Networks and Their Minimization Using a New Line of Elementary Units," submitted to IEEE Transactions on Computers and currently under review, vol. C-24, no. 3, pp. 258– 280, 1975.

[21] K.A.C. Bickerstaff, M. Schulte, and E.E. Swartzlander, "Reduced area multipliers," in Application-Specific Array Processors, 1993