High Throughput and Density Optimized Efficient Multi Operand Adder

Vannekuti Jahnavi [1], Dr.U.Yedukondalu [2]
[1] *M. Tech, VLSI & Embedded Systems, Sri Vasavi Engineering College (A), Tadepalligudem-534101*
[2]*Associate Professor, Dept. of ECE, Sri Vasavi Engineering College (A), Tadepalligudem-534101*

***Abstract*— Basic mathematical operations like addition subtraction, multiplication and division operations are performed by many digital devices including microprocessors and digital signal processors. Even used for complicated calculations very efficiently. Here an efficient adder circuit is implemented that revolves around reducing the cost to propagate the carry among consecutive bit positions. Consequently, a novel high speed area efficient adder architecture is implemented using bitwise pre-calculations that followed by carry prefix logic to perform three operand binary addition that utilizes less area, power, and delay. Additionally, this article enhances uses modified carry bypass adder for reducing density and latency limitations. This also modifies carry skip adder that presents a simple and low complicated carry skip logic for reducing parameters limitations.**

***Index Terms:* Parallel Prefix, Brent – Kung, Carry save addition, pre-compute bitwise addition followed by carry prefix, Carry skip addition.**

I,INTRODUCTION

In addition to technological expansion, advances in computer architecture have exponentially increased the performance of digital computing hardware. The flip side of processor performance gains is an unprecedented increase in hardware and software complexity. Increased complexity leads to high development costs, difficulties in testability and verifiability, and poor adaptability. As a result, computer designers face the challenge of choosing circuits that are simpler, more dependable, and simpler to verify. First, at very high clock rates, the interface between the arithmetic circuitry and the rest of the processor becomes important. Arithmetic circuits can no longer be designed and tested in isolation. Rather, it requires complex design optimizations. Second, optimizing arithmetic circuits to exploit the strengths of new technologies and make them resilient to weaknesses to achieve design goals requires revisiting existing design paradigms. Finally, the inclusion of high-level arithmetic primitives in hardware makes designing, optimizing, and validating very complex and intertwined. The data path consists of various arithmetic devices such as comparators, adders, and multipliers [4].

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 using 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 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 usually done using algorithms based on logical and arithmetic operations on data elements [10]. Sometimes special circuit blocks are also included for fast division and square root. Adders, increasers/decrements, and comparators are often used for address calculation and controller flag generation purposes.

II.LITERATURE SURVEY

Multi-Operand Adders are commonly used in two modes namely Array Adders and Adder Tree structure. Requires ‘K’ number of adder levels to add ‘K’ operands. But in the event Adder Tree sets the number of levels to add ‘K’ operands is smaller than that of Array Adders. It combines the ‘K’ number of

operands into two operand sets. All sets are added accordingly at the same level. The plumbing method can be used most effectively RCA [1]. Adder Tree delays using CPA are high due to carrying the spread over bit lengths. The benefit of the Carry Save Adder (CSA) tree is applied to the use of the ASIC because of the flexibility. But using FPGA Ripple Carry Adder tree is more popular than the CSA tree adder. However, a direct FPGA implementation [6] requires about twice as much hardware as a ripple transfer adder and does not use fast transfer chains for increased speed. The amount from the first level was reassembled into two operands sets and performed the merger. This process continues until two operands are obtained and added to the final level to obtain the final amount. Ripple Cary Adder (RCA) or Carry Look Adder (CLA) are two common Carry Propagate Adders used in the above methods namely Array Adder, Adder Tree Carry Propagate Adder. The delay of their CPA depends on the operational minimum length. To reduce the delay of these additives where they are applied to the FPGA through dedicated chains [8]. The RCA in FPGA uses a simpler management chain than any other CPA topology at a higher hardware cost [9].

III.PRE-COMPUTE BITWISE ADDITION FOLLOWED BY CARRY PREFIX COMPUTATION

This Proposed method implements a novel adder method and VLSI design to implement a three-operand addition in modular calculation. A parallel prefix adder is proposed which has four-stages instead three-stages in prefix adder to calculate the addition of three binary input operands such as addition logic, base logic, PG (propagate and generate) logic and sum logic bitwise. The four stages are defined by using logical expression 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:

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

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:

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

The proposed three-function binary adder structure and its internal structure are shown in Fig.1 The new adder method enables the addition of three n-bit binary inputs in four different categories. In the first phase (bit-addition logic), the gradual addition of three n-bit binary operands is done with a series of full add-ons, and each full adder counts “total (S_ i)” and “carry” (cyi)” signals as highlighted in Fig. 1 (a). Logical computing sum (S_i) and (cyi) signals are described in Section 1, and a logical drawing of bit-addition logic is shown in Fig. 1 (b)

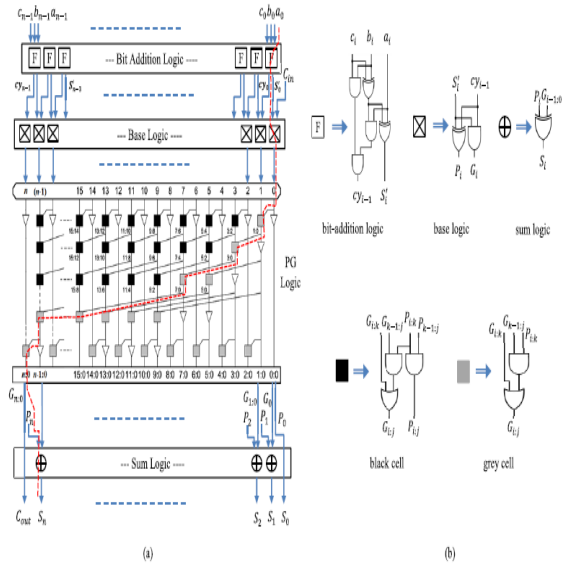

Fig 1. Proposed three-operand adder; (a) First order architecture, (b) Logical diagram of bit addition, base logic, sum logic, grey-cell, and black-cell.

In the first phase, the outgoing signal “sum (S_ i)” the full current adder and the sub-signal “thwala” of its right-hand adder are used together to calculate the production (Gi) and distribution (Pi) signals in the second phase (logic base). The Giant and Pi signals are represented by a "square saltire-cell" as shown in Fig. 1 (a) and there is a n + 1 number of saltire cells in the basic sensory phase. A logical diagram of the saltire cell is shown in Fig. 1 (b) and is characterized by the following logical expression.

$$G_{i:i} = G_i = S_i' \cdot cy_{i-1};$$
$$P_{i:i} = P_i = S_i' \oplus cy_{i-1}$$

An external input signal (Cin) is also considered to add three functions to the proposed adder system. This additional input signal (Cin) is the basic input device while the computer G0 (S_ 0 · Cin) is inserted into the primary saltire cell core logic. The third phase is the calculation phase called “generate and propagate logic” (PG) to calculate the carry bit and is a combination of black and gray cell thinking. A logical diagram of the black and gray cell is shown in Fig. 3 (b) combining carry generate Gi: j and transmitting Pi: j signals with the following logical expression,

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

Since the number of prefix calculation stages in the proposed adder is (log2 n + 1), the critical path delay of the proposed adder is primarily affected by this carry propagation chain. The final stage is expressed as total logic, the "total (Si)" bits are derived from the carry-generated Gi: j, and the carry-propagated Pi bits are calculated using the formula Si = (Pi _ Gi-1: 0). increase. The carry-out signal (Cout) is obtained directly from the carry-generated bit Gn: 0.

IV.PROPOSED CARRY SKIP ADDER

The structure is based on a combination of integration and expansion schemes [13] with the Conv-CSKA structure, therefore, defined by the CI-CSKA. It gives us the ability to use simple skip logic. Logic replaces the 2: 1 multiplier with AOI / OAI composite gates. Gates, consisting of fewer transistors, have lower latency, space, and lower power consumption compared to those of the 2: 1 multiplexer [7]. Note that, in this structure, as the load is propagated in skip logics, it is perfected. Therefore, at the skip logic output of the equal sections, a carrying case is produced. The structure has a very low distribution delay in a small area compared to the conventional one. Note that although AOI (or OAI) power consumption is lower than that of multiplexer, the power consumption of the proposed CI-CSKA is slightly higher than normal. This is due to the increase in the number of gates, which sets the wiring capacitance higher (in less important ways). Now, we describe the internal structure of the proposed CI-CSKA shown in Fig. 2 in detail. The adder contains two N inputs, sections A and B, and Q. Each stage contains an RCA block with Mj size (j = 1,..,Q).

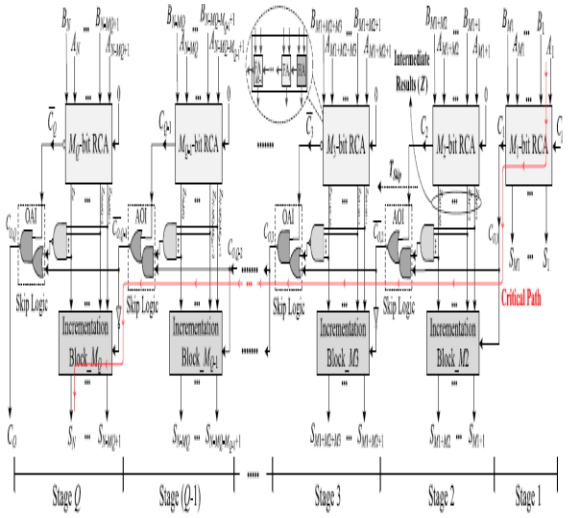

Fig 2. CI-CSKA

In this structure, the carrying element of all RCA blocks, apart from the first Ci block, is zero (a combination of RCA blocks). Therefore, all blocks perform their functions simultaneously. In this structure, where the first block calculates the sum of its corresponding input bits (i.e., SM1,..,S1), and C1, the other blocks simultaneously calculate the median results. In the proposed building, the first phase has only one block, which is the RCA. Sections 2 to Q contain two blocks of RCA and climb. The incrementation block uses a structure.

Here are intermediate results produced by the RCA block and the output of the previous stage load to calculate the final depth of the stage. The internal structure of the incrementation block, which contains a series of half-adders (HAs), is shown in Fig. 2. In addition, note that, to minimize the significant delay, by calculating the output of the stage load, the bearing. expansion block output is not used. As shown in Fig. 2, skip logic determines the output of the j th phase (CO, j) based on the intermediate effects of the j j phase and the output of the previous phase (CO, j-1) as well as the output of the corresponding RCA block (Cj). When determining CO, j, these conditions may combine. If Cj is equal to one, CO, j will be one. On the other hand, if Cj is equal to zero, if the output of the median output is one (zero), the value of CO, j will be equal to CO, j − 1 (zero). The reason for using both AOI and OAI

gates as a combination of skip logics is the flexible function of these gates in standard cell libraries.

In this way the need for an inverter gate, which increases power consumption and delay, is eliminated. As shown in Fig.2, If AOI is used as an escape concept, the next escape concept should use the OAI gate. In addition, another point to be made is that the use of the proposed escape structure in the Conv-CSKA building increases the delays of the most critical route. This stems from the fact that, in Conv-CSKA, the escape concept (AOI or OAI integrated gateway) cannot exceed zero load input until zero load input spreads to the subsequent RCA block. To solve this problem, in the proposed structure, we used an RCA block with zero load insertion (using the coupling method). In this way, since the RCA block does not need to wait for the load of the prior phase load, the output carrier blocks are calculated accordingly. As mentioned earlier, the use of vertical AOI and OAI gates (six transistors) compared to a 2:1 vertical multiplexer (12 transistors), leads to a decrease in space usage and a delay in the transition concept.

Additionally, apart from the first RCA block, the load input for all other blocks is zero, so, in these blocks, the first adder cell in the RCA series is HA.

V.RESULTS ANALYSIS

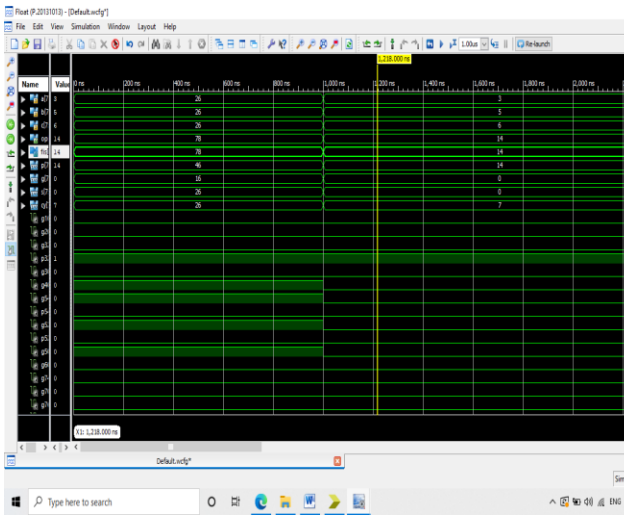

Fig 3. Existing method result with three operand addition using one three operand adder

Above figure 3 shows base paper method simulation result taken in XILINX-ISE 14.7 with three operand addition using one three operand adder.

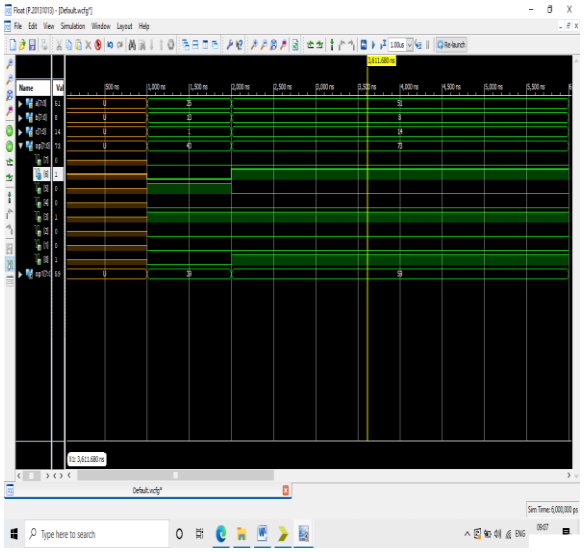

Fig 4. Proposed Skip Effect

The above figure 4 shows the proposed skip effect taken from XILINX-ISE 14.7 and three operand combinations using two operand add-ons.

VI.CONCLUSION AND FUTURE SCOPE

In this paper, an adder method that works well in high-speed environments and its VLSI design is proposed to make three operand sequences for operand for accurate computation. The proposed three-function adder method is a coherent start that uses four-phase structures to calculate the addition of three input operands. The innovation of this proposed construction is the reduction of latency and space in the initial calculation phases in PG logic and bit-addition logic. As an extension of this concept, a vertical structure of the CMOS CSKA called CI-CSKA was proposed, showing higher speed and lower power consumption compared to conventional ones. Speed enhancement has been accomplished by changing the structure using the techniques of integration and expansion. Furthermore, AOI and OAI are compound gates used in the form of carrying skip logics.

In future work, further research will be conducted using the two-way FP structure in a three-function FP adder and the use of other obsolete FP presentations. The use of advanced techniques in the design phase of the project (i.e., obsolete LZD, customization, and rotation) will result in faster construction, although higher local costs are expected.

REFERENCE

[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

[22] Suhas B. Shirol, S. Ramakrishna and Rajashekar B. Shettar, "Design and Implementation of Adders and Multiplier in FPGA Using ChipScope: A Performance Improvement", Information and Communication Technology for Competitive Strategies pp 11-19, 31 August 2018