ENGINEERING IN ADVANCED RESEARCHSCIENCE AND TECHNOLOGY

ISSN 2352-8648
Vol.03, Issue.01
June-2022
Pages: -561-569

HIGH THROUGHPUT THREE OPERAND BINARY ADDER WITH DENSITY OPTIMIZED CARRY SKIP ADDER

[1]DARAM ANITHA KUMARI, [2]K.RADHA, [3]D AJAY KUMAR
[1]PG Student, Dept. of ECE, Sir C.R. Reddy College of Engineering, Eluru, A.P
[2]Assistant professor, Dept. of ECE, Sir C.R. Reddy College of Engineering, Eluru, A.P
[3]Assistant professor, Dept. of ECE, Sir C.R. Reddy College of Engineering, Eluru, A.P

ABSTRACT: Addition 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 analysed 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 is 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 datapath. With respect to the most important design criteria; critical delay, chip size, and power dissipation, the datapath is a crucial circuit component. The datapath 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 datapath. Special circuit units for fast division and square-root operations are sometimes included as well. Adders, incrementers/decrementers, and comparators are often used for address calculation and flag generation purposes controllers. 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:

The Multi-Operand Adders are generally implemented in two methods i.e Array Adders and Adder Tree structure. In Array Adder structure, two operands are added and output is added with third operand and continues the chain of addition until to get final sum output. It requires 'K' number of adder levels for addition of 'K' operands. But in case of Adder Tree structure the number of levels to add 'K' operands is less than that of Array Adders. It groups 'K' number of operands into sets of two operands. All the sets are added parallel in one level. The sum outputs from first level again grouped into sets of two operands and perform addition. This process continues until to get two operands and added in last level to obtain final sum. In each level it reduces number of operands to half. Therefore it requires log2 K levels. The Adder Tree structure is faster than Adder Array structure with same resources consumed by both configurations. But the Array Adder is having regular routing than Adder Tree structure. The Ripple Carry Adder (RCA) or Carry Look Ahead Adder (CLA) are two general Carry Propagate Adders used in the above methods i.e. Array Adder, Adder Tree is Carry Propagate Adder. The delay of their CPA depends on bit length of operand. For N-bit operand the of RCA proportional to N and for CLA it is proportional tolog2 N. To reduce the delay these adders where implemented on FPGA by using dedicated carry chains [8]. The RCA on FPGA using fast carry chain is simpler than any other CPA topologies at an expense of high hardware cost [9]. The pipelining technique can be applied more effectively RCA [1]. The delay of Adder Tree using CPA is high due to carry propagation along the bit length. Carry Save Adder tree is used as another approach for implementing Multi-Operand Adders. Here the carry is directly propagated to next level instead of propagating in the same as in case of CPA. The advantage of Carry Save Adder (CSA) tree is utilized in ASIC implementation due to flexible routing. The critical path delay can be minimized by optimizing the interconnection between Full Adders. But to implement on FPGA the Ripple Carry Adder tree is preferred than CSA adder tree. When CSA tree is implemented on FPGA it become slower than RCA tree due to routing delay of CSA. However, a straightforward implementation on FPGAs [6] roughly requires double hardware than a carryripple adder, and does not exploit the fast carry chain to improve speed.

III. 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:

$$G_{i:i} = G_i = S'_i \cdot cy_{i-1}, \quad G_{0:0} = G_0 = S'_0 \cdot C_{in}$$
$$P_{i:i} = P_i = S'_i \oplus cy_{i-1}, \quad P_{0:0} = P_0 = S'_0 \oplus C_{in}$$

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:

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

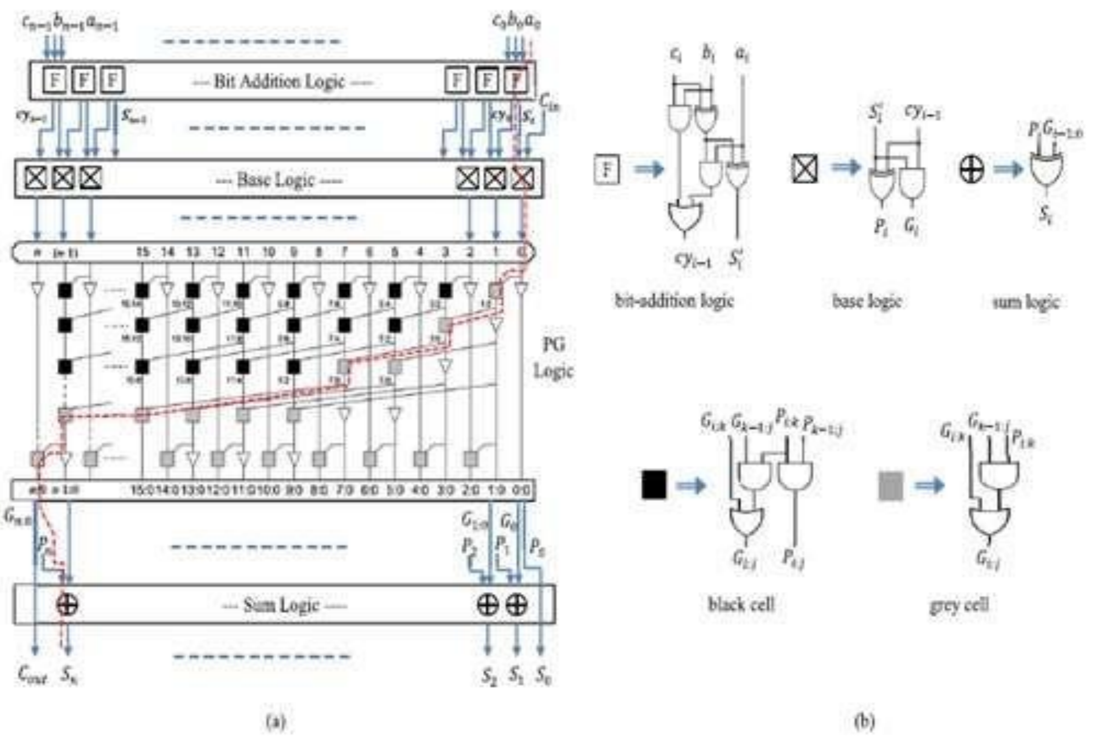


Fig1. Proposed three-operand adder; (a) First order VLSI architecture, (b) Logical diagram of bit addition, base logic, sum logic, black-cell and grey-cell.

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. 3(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. 3(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 Gi and Pi signals are represented by the "squared saltire-cell" as shown in Fig. 3(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. 3(b), and it is realized by the following logical expression,

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

The external carry-input signal (Cin) is also taken into consideration for three-operand addition in the proposed adder technique. This additional carry-input signal (Cin) is taken as input to base logic while computing the G0 (S_ 0 · Cin) in the first saltire-cell of the base logic. The third stage is the carry computation stage called "generate and propagate logic" (PG) to pre-compute the carry bit and is the combination of black and grey cell logics. The logical diagram of black and grey cell is shown in Fig. 3(b) that computes the carry generate Gi: j and propagate Pi: j signals with the following logical expression,

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

The number of prefix computation stages for the proposed adder is (log2 n+1), and therefore, the critical path delay of the proposed adder is mainly influenced by this carry propagate chain. The final stage is represented as sum logic in which the "sum (Si)" bits are computed from the carry generate Gi: j and carry propagate Pi bits using the logical expression, Si = (Pi _ Gi−1:0). The carryout signal (Cout) is directly obtained from the carry generate bit Gn:0.

IV.PROPOSED 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

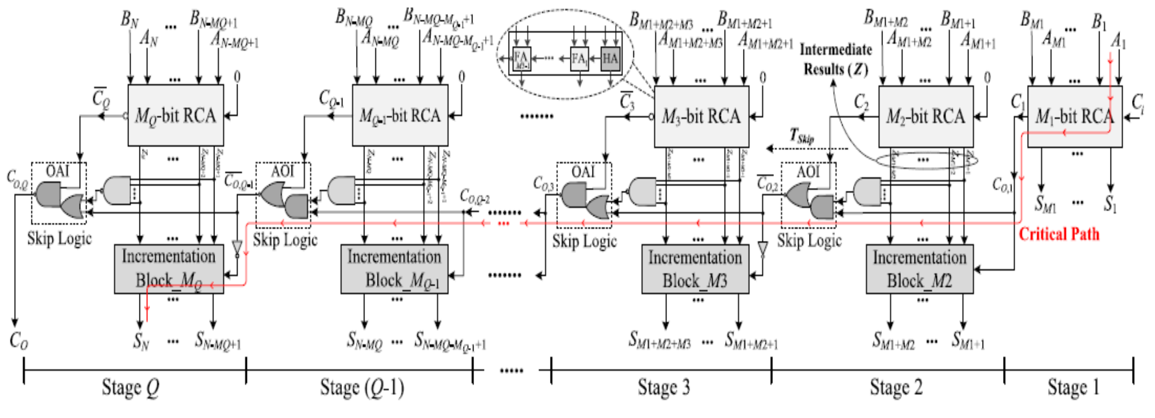
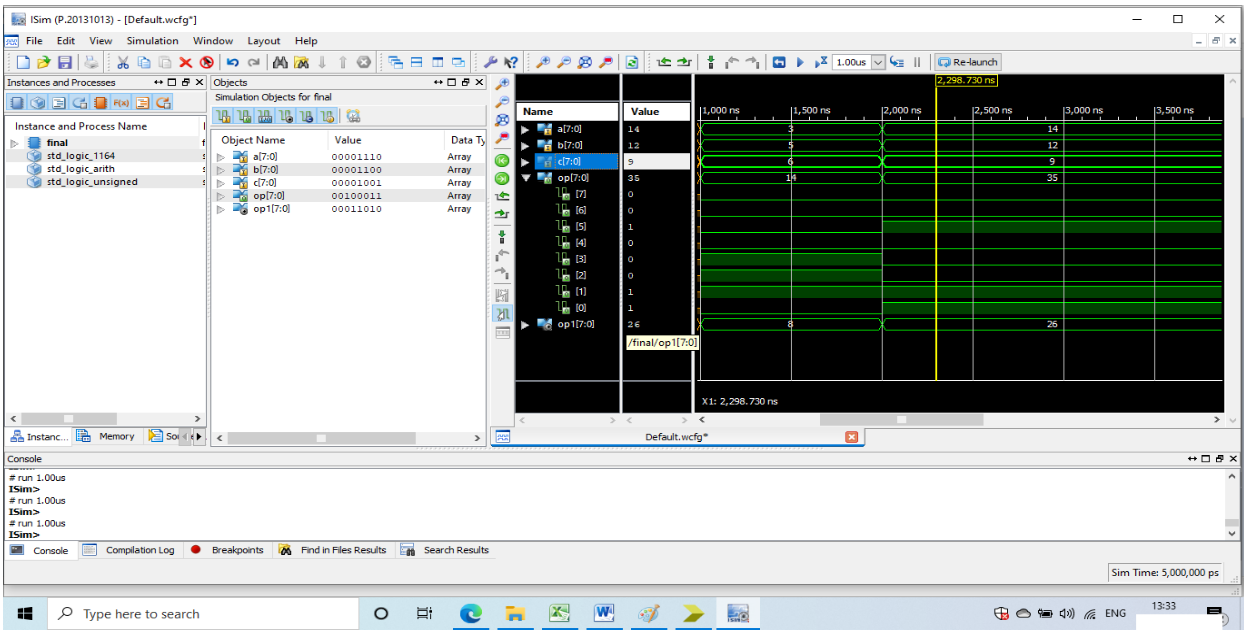


Fig2: CI-CSKA structure.

intermediate results generated by the RCA block and the carry output of the previous stage to calculate the final summation of the stage. The internal structure of the incrementation block, which contains a chain of half-adders (HAs), is shown in Fig. 4. In addition, note that, to reduce the delay considerably, for computing the carry output of the stage, the carry output of the incrementation block is not used. As shown in Fig. 2, the skip logic determines the carry output of the j th stage (CO, j) based on the intermediate results of the j th stage and the carry output of the previous stage (CO, j−1) as well as the carry output of the corresponding RCA block (Cj). When determining CO, j , these cases may be encountered. When Cj is equal to one, CO, j will be one. On the other hand, when Cj is equal to zero, if the product of the intermediate results is one (zero), the value of CO, j will be the same as CO, j−1 (zero). The reason for using both AOI and OAI compound gates as the skip logics is the inverting functions of these gates in standard cell libraries. This way the need for an inverter gate, which increases the power consumption and delay, is eliminated. As shown in Fig., if an AOI is used as the skip logic, the next skip logic should use OAI gate. In addition, another point to mention is that the use of the proposed skipping structure in the Conv-CSKA structure increases the delay of the critical path considerably. This originates from the fact that, in the Conv-CSKA, the skip logic (AOI or OAI compound gates) is not able to bypass the zero carry input until the zero carry input propagates from the corresponding RCA block. To solve this problem, in the proposed structure, we have used an RCA block with a carry input of zero (using the concatenation approach). This way, since the RCA block of the stage does not need to wait for the carry output of the previous stage, the output carries of the blocks are calculated in parallel. As mentioned before, the use of the static AOI and OAI gates (six transistors) compared with the static 2:1 multiplexer (12 transistors), leads to decreases in the area usage and delay of the skip logic. In addition, except for the first RCA block, the carry input for all other blocks is zero, and hence, for these blocks, the first adder cell in the RCA chain is a HA.

RESULTS:

Proposed simulation:



Object Name	Value	Data Ty
a[7:0]	00001110	Array
b[7:0]	00001100	Array
c[7:0]	00001001	Array
op[7:0]	00100011	Array
op1[7:0]	00011010	Array

Name	Value
a[7:0]	14
b[7:0]	12
c[7:0]	9
op[7:0]	35
[7]	0
[6]	0
[5]	1
[4]	0
[3]	0
[2]	0
[1]	1
[0]	1
op1[7:0]	26

```
# run 1.00us
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Fig3: Proposed simulation output

Above simulation snapshot shows three operand binary adder using proposed method with a, b, c inputs and op as output sum of those inputs. One bunch of inputs are a=14, b=12, c=9; yielded output op=35 in decimal format. Addition among operands a, b, and c was takes place using two "two operand" "modified carry skip adder" s. Another bunch of inputs are also tested in same screen shot; those are a=3, b=5, c=6; yielded output op=14 in decimal format. Remaining variable 'OP1' represents intermediate signals of proposed designed method.

Table1: Comparison table

Parameter	Existing	Proposed
Area (Gate count)	414	116
Delay(ns)	16.29	8.403

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

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.