IIOAB JOURNAL
ISSN: 0976-3104

ARTICLE

FPGA IMPLEMENATION OF HIGH SPEED AND LOW POWER CARRY SAVE ADDER

VS. Balaji[1*], Har Narayan Upadhyay[2]

[1]*Department of Electronics & Instrumentation Engineering, INDIA*

[2]*Dept.of Electronics & Communication Engineering, School of EEE, SASTRA University, Thanjavur (TN), INDIA*

ABSTRACT

In recent years, the high speed adders play a vital role in any digital design of signal processing systems by performing much needed operations like subtraction, multiplication, and division, etc., along with the basic addition. Carry look-ahead adder (CLA), carry select adder (CSeA), carry-save adder (CSA) and carry-skip adder are the most popular high speed adders in digital signal processor and computer data path applications. This study demonstrates the efficiency of a carry-save adder with multi-operand addition using different majority voting circuit implementations in the place of carry out of a full adder. The carry is saved and propagated down to the next stage rather than passing within the same stage like in ripple carry adder (RCA). But the final sum is calculated with the help of carry ripple adder by propagating the carries appropriately. The final sum calculation stage in a carry save adder is implemented in this study using ripple carry adder (CSA-RCA), carry look-ahead adder (CSA-CLA) and carry select adder (CSA-CSeA) to compare better power, area and delay results. Addition of four 4-bit numbers, four 8-bit numbers, four 16-bit numbers, four 32-bit numbers and four 64-bit numbers are implemented on Altera EP4CE115F29C7 FPGA device using Quartus II synthesis software tool. Simulation results of adding four 32-bit numbers show that CSA-CSeA topology offers lower critical path delay as 34.823 ns compared with CSA-RCA topology (63.120 ns) and CSA-CLA topology (52.475 ns) at the expense of 221 logic elements (LEs) which is higher; the CSA-RCA topology and CSA-CLA topologies have only 192 LEs and 193 LEs respectively. It is also shown that the CSA-CLA topology has lower power dissipation as 170.54 mW compared with CLA-RCA (171.36 mW) and CSA-CSeA (174.68 mW).

KEY WORDS
Carry save adder, FPGA, high speed adders, majority function, multi-operand addition, Quartus II.

Published: 2 December 2016

***Corresponding Author**
Email:
biobala_mtech@eie.sastra.edu,
hnu@ece.sastra.edu
Tel.: +918072496254

INTRODUCTION

In digital computer arithmetic, addition and subtraction are the most basic core operations, especially for digital signal processing applications. The other two fundamental operations like multiplication and division are too performed using addition and subtraction and hence they play a very important role in processors like microprocessors, microcontrollers and digital signal processors [1]. The fast adders can be used to speed up the arithmetic operations in a processor. There are numerous ways available to perform addition but with different trade-offs. Some are good in producing low power at the cost of area and some are best in offering performance (i.e., high speed) at the cost of power and/or area [2].

Arithmetic operations are essential building blocks in any system; either the system can be designed based on a processor, or an FPGA/ASIC [3]. The data path circuitries in a microprocessor, Multiply-Accumulate (MAC) operations in a digital signal processor and high speed integrated circuits in communication systems are the few application examples which would perform arithmetic operations especially using regular full adders [4]. The k-bit ripple-carry adder is the most simplest adder structure, which adds two words having k-bits. The delay of the RCA is very high since the carry is propagated (i.e., rippled) to the full adder stages to produce a final sum. If the value of k is very large, then the delay would be more. These RCAs are not suitable in situations where the speed data processing is involved. So the demand to design fast adders grows rapidly to meet the current high speed integrated circuits trend [5] [6].

This paper is organized as follows. The Section 2 contains the carry-save adder implementations with ten different majority voting circuits in the place of a carry out of a single-bit full adder in carry-save stages. The final sum is obtained using different circuits like carry look-ahead adder and carry-select adder in addition to the conventional ripple-carry adders, which are discussed in Section 3. The Section 4 obtains simulation results with FPGA implementation along with performance metrics like resource utilization report, critical path delay and power dissipation with Quartus II synthesis software tool using very high speed integrated circuits hardware description language (VHDL).

MULTI-OPERAND ADDITION USING CARRY-SAVE ADDERS

The addition of more than two numbers would be done using carry-save adders based on the fact that a full-adder has 3 inputs and obtains 2 outputs and hence the full adder becomes the basic building block [6] [7]. The addition of four 4-bit numbers, four 8-bit numbers, four 16-bit numbers, four 32-bit numbers and four 64-bit numbers are exemplified using CSA principle in this study.

Addition of four 4-bit numbers

A_{0-3}, B_{0-3}, C_{0-3} and D_{0-3} are the four 4-bit numbers which are used for addition. This is accomplished with the help of full adders (boxes) as in figure to obtain the overall sum as in [Fig. 1]. The carry from each stage is propagated down instead in the same stage and hence reduce the overall delay. These are known

as "carry-save" stages and varies depend on the number of operands. Finally, the ripple-carry adder is used to obtain the final sum at the cost of delay [6] [8]. The complete schematic with full adder implementation (boxes) is shown in [Fig. 2].

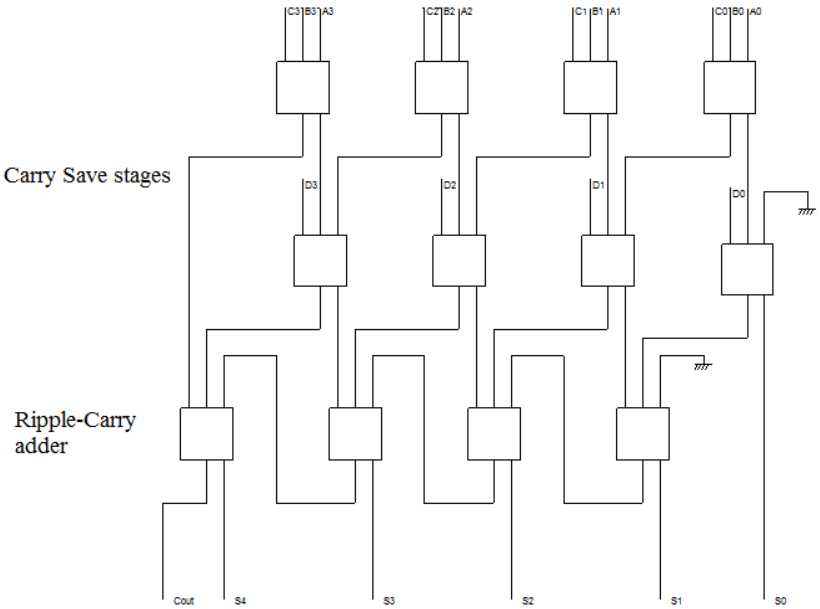


Fig. 1: Addition of four 4-bit numbers using CSA.

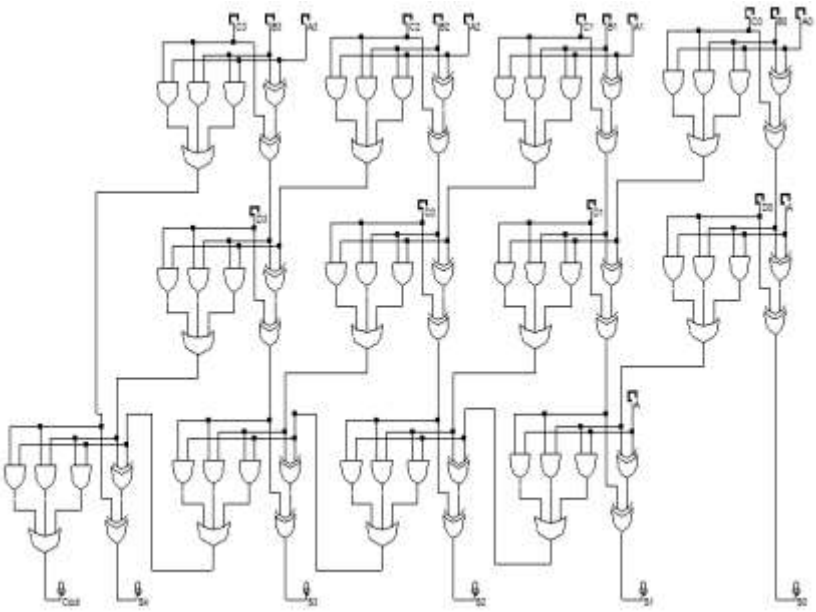


Fig. 2: Addition of four 4-bit numbers using full adders.

Parity-checking and majority-voting of full adder

The sum and carry out of a full adder can be viewed as parity-checking and majority-voting circuits respectively as in [Fig. 3]. The parity-checking of the inputs (a,b,c) is calculated as (a xor b xor c) and the majority-voting of the inputs (a,b,c) is determined as (ab or bc or ca) are known as sum and carry out respectively [9] [10].

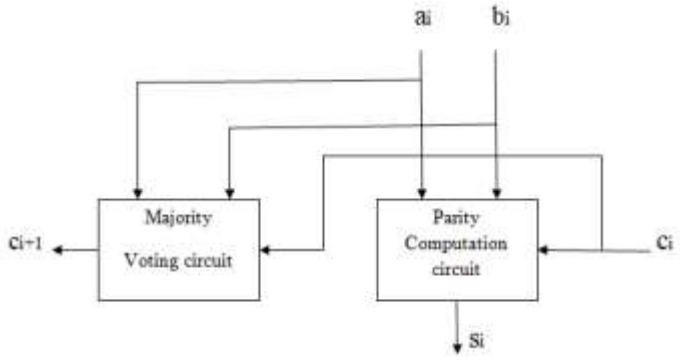


Fig.3: Single-bit full adder.

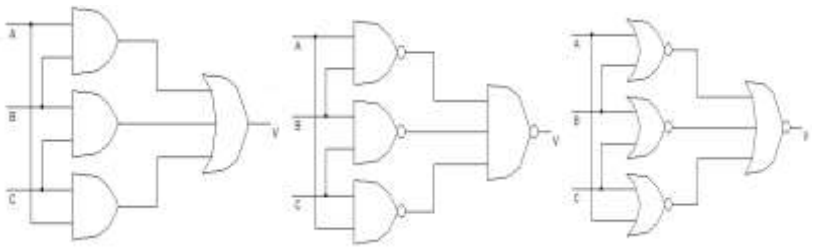
.....

Majority-voting (MV) circuits

The different majority-voting circuits are studied and are implemented in the place of a carry out of a full adder in a carry-save adder structure.

Using conventional, NAND, and NOR gates

The majority-voting circuits are implemented using conventional method, NAND gates and NOR gates as in [Fig. 4a, 4b and 4c] respectively [11]. The NAND and NOR gates are very often used in CMOS VLSI circuits for better area, power and delay trade-offs.



(a) (b) (c)

Fig: 4. Using (a) Conventional (b) NAND and (c) NOR gates.

.....

Using carry look-ahead adder fundamental

The carry out of a full adder is implemented as (a xor b) c + (ab) as in CLA concept. This is shown in [Fig.5]. The *a xor b* and *ab* terms are referred as 'propagate' and 'generate' respectively. If a $\neq$ b and carry in =1, then the carry out = 1, which is based on the propagate term. If a = b = 1, then the carry out =1, which is based on the generate term [12].

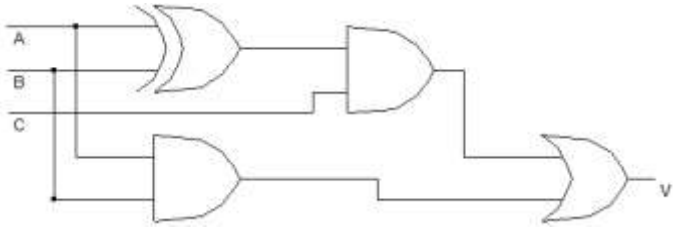
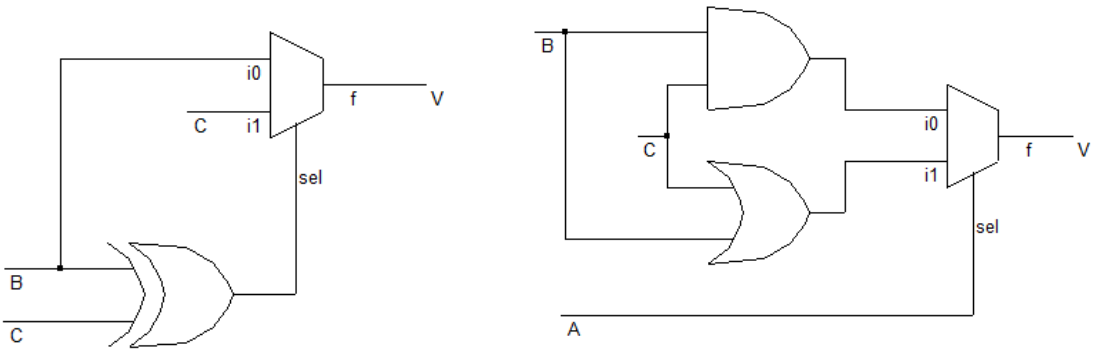


Fig. 5: MV circuit using CLA principle.

.....

Using Multiplexers, XOR and OR gates

The MV circuit can be designed using multiplexer and XOR gate as in Figure 6a. If A ≠ B, then the majority becomes the third input; otherwise, the output is either B or A. The majority function can also be computed using Mux, AND and OR gates as in [Fig. 6b]. If the first input bit is equal to zero, then the MV output is computed as B and C, else it is determined as A or B [12].



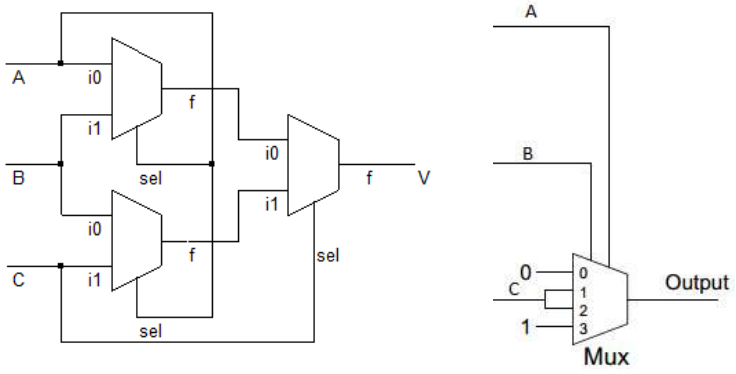
(a) (b)

Fig.6: Using (a) Mux and XOR gate and (b) Mux, AND and OR gates.

.....

Using 2-to-1 and 4-to-1 Multiplexers

The MV circuit can be designed only using 2-to-1 multiplexers as in [Fig.7a]. The [Fig. 7b] shows the majority function schematic using 4-to-1 multiplexer. If A = B = 0, then the output becomes as '0' and if A = B = 1, then the output is '1'. If A ≠ B, then the majority output becomes as the third input 'C' [12].



(a) (b)

Fig. 7: Using (a) 2-to-1 Muxes and (b) 4-to-1 Muxes.

.....

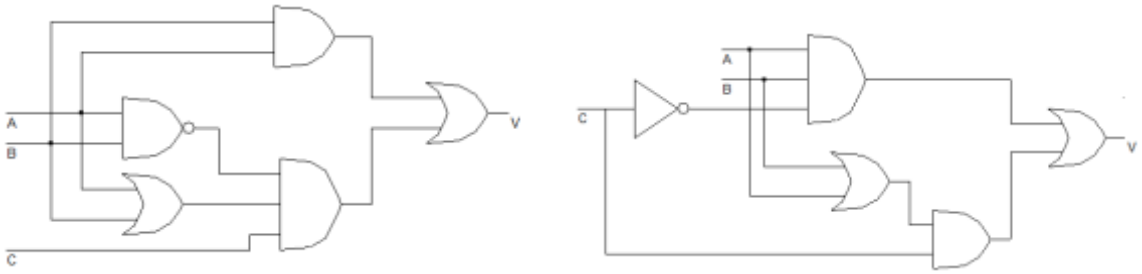
Using AND, NAND, OR and Inverter gates

The carry out of a full can also be design using the equation (1) and the schematic is shown in [Fig. 8a] using AND, NAND, OR and Inverter gates which is referred to as ANOI in this study.

$$V = (\overline{AB})(A+B)C+AB \quad (1)$$

The MV circuit output can also be determined using the equation (2) and the schematic is shown in [Fig. 8b] using AND, OR and Inverter gates which is referred to as AOI in this study.

$$V = (AB\overline{C})+(A+B)C \quad (2)$$



(a) (b)

Fig. 8: Using (a) AND, NAND, OR and Inverter (b) AND, OR and Inverter

FINAL SUM COMPUTATION METHODS OF CSA

Multi-operand addition using CSA has two important stages as "carry-save stages" and "final stage" to compute the overall sum. This section exemplifies the final stage using ripple-carry adder, carry look-ahead adder and carry select adder to determine the final output sum.

Sum output using RCA

In this study, the final sum output using RCA is referred in this study as "CSA-RCA". The carry from the previous stages are considered here and propagated within the last stage to find the final sum output value. The general schematic of *k*-bit RCA is shown in [Fig. 9]. It is apparent that this circuit offers more delay due to the rippling effect of the carry which is shown in the diagram [2] [4].

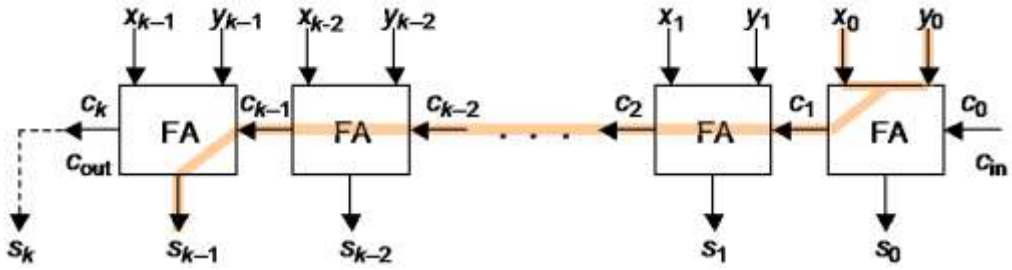


Fig. 9: *k*-bit ripple-carry adder.

Sum output using CLA

The last stage is replaced with carry look-ahead adder circuit and is referred to as "CSA-CLA". The carry out is calculated in advance based on the "propagate" and "generate" terms instead of depending on the previous carry. Since the rippling effect is reduced, this circuit should offer better performance compared with the RCA combination. The schematic of 4-bit CLA is shown in [Fig. 10]. The VHDL portion of a CLA stage with the addition of four 8-bit numbers is shown in Figure 11. If $a_0 = b_0 = 1$ (i.e., $g_0 = 1$), then the c1 is computed as '1' based on the *generate* term. If $p_0 = 1$ and $c_0 = 1$, then the c1 is computed as '1' based on the *propagate* term. The CLA circuit is constructed based on the following equation (3).

$$C_{i+1} = (A_i \oplus B_i)C_i + A_iB_i = P_iC_i + G_i \quad (3)$$

For a 4-bit CLA circuit, these carry terms can be computed as,

$$C_1 = P_0C_0 + G_0$$

$$C_2 = P_1C_1 + G_1 = P_1(P_0C_0+G_0) + G_1$$

$$C_3 = P_2C_2 + G_2 = P_2(P_1P_0C_0+P_1G_0+G_1) + G_2$$

$$C_4 = P_3C_3 + G_3 = P_3(P_2P_1P_0C_0+P_2P_1G_0+P_2G_1) + G_3 \quad (4)$$

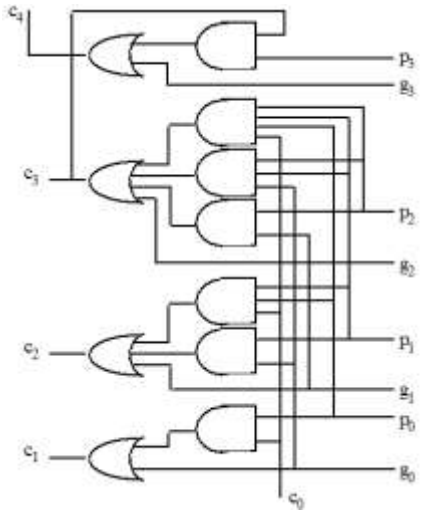


Fig.10: A 4-bit carry-look ahead adder.

.....

```
p0 <= m16 xor m17; g0 <= m16 and m17; c11 <= (p0 and cin1) or g0; s1 <= p0 xor cin1;
p1 <= m18 xor m19; g1 <= m18 and m19; c21 <= (p1 and c11) or g1; s2 <= p1 xor c11;
p2 <= m20 xor m21; g2 <= m20 and m21; c31 <= (p2 and c21) or g2; s3 <= p2 xor c21;
p3 <= m22 xor m23; g3 <= m23 and m23; c41 <= (p3 and c31) or g3; s4 <= p3 xor c31;
p4 <= m24 xor m25; g4 <= m24 and m25; c51 <= (p4 and c41) or g4; s5 <= p4 xor c41;
p5 <= m26 xor m27; g5 <= m26 and m27; c61 <= (p5 and c51) or g5; s6 <= p5 xor c51;
p6 <= m28 xor m29; g6 <= m28 and m29; c71 <= (p6 and c61) or g6; s7 <= p6 xor c61;
p7 <= m15 xor m30; g7 <= m15 and m30; cout <= (p7 and c71) or g7; s8 <= p7 xor c71;
```

Fig. 11: Final sum using ripple-carry adder in VHDL.

.....

Sum output using CSeA

The carry-select adder can also be used in the last stage to produce the final sum and this topology is referred to as “CSA-CSeA”. This is used especially when adding more number of bits. The basic schematic of a CSeA is shown in [Fig. 12]. The [Fig. 13] shows the VHDL portion of the addition of four 8-bit numbers using carry-select adder stage [5] [6].

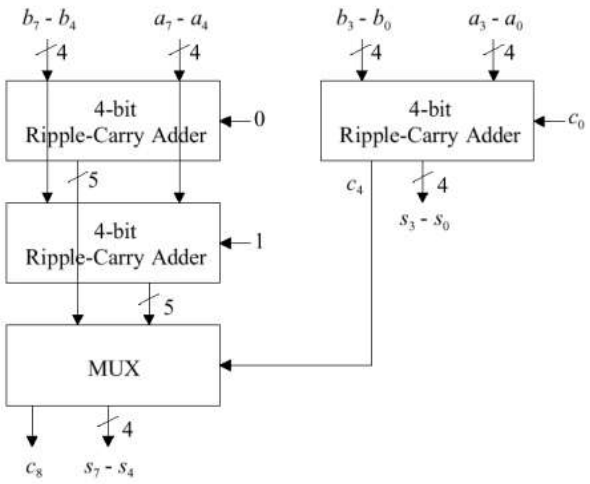


Fig.12: A 8-bit carry-select adder.

```
s4 <= ((not c4) and s50) or (c4 and s51);
s5 <= ((not c4) and s60) or (c4 and s61);
s6 <= ((not c4) and s70) or (c4 and s71);
s7 <= ((not c4) and s80) or (c4 and s81);
c8 <= ((not c4) and c80) or (c4 and c81);
```

Fig.13: Final sum using carry-select adder in VHDL.

RESULTS AND DISCUSSION

This paper demonstrate the addition of four 4-bit, 8-bit, 16-bit, 32-bit and 64 numbers and thus obtain a comparative analysis with the performance metrics like power, delay and area in each design. All the circuits are implemented on Altera FPGA EP4CE115F29C7 device (i.e., DE2-115) using Quartus II synthesis software tool with VHDL to obtain simulation results [13] [14]. The timing verification is done using the Time Quest timing analyzer, which is a part of Quartus II software bundle. The functional and timing verification results are shown in [Fig. 14] and [Fig. 15] respectively for the addition of four 32-bit numbers.

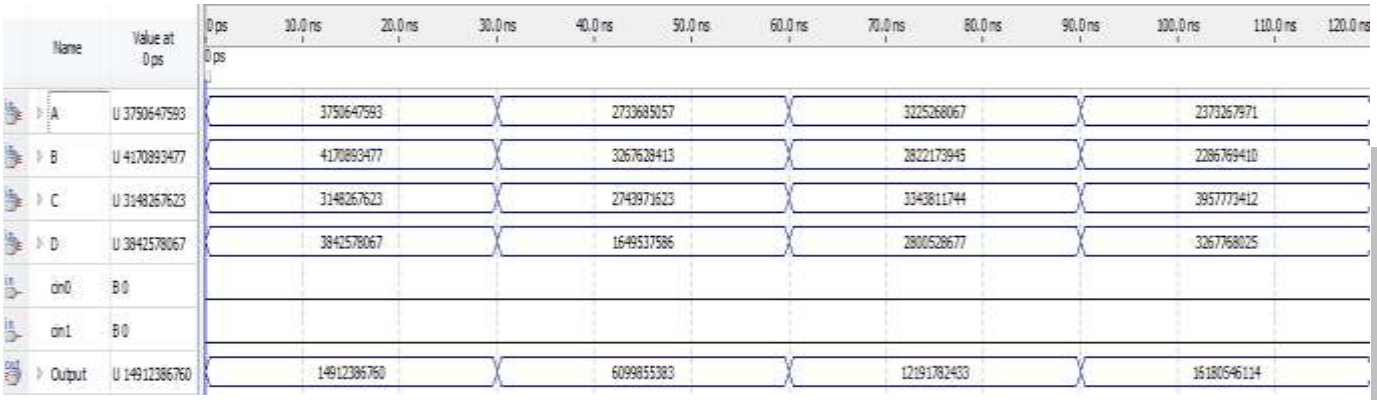


Fig. 14: Functional verification for the addition of four 32-bit numbers.

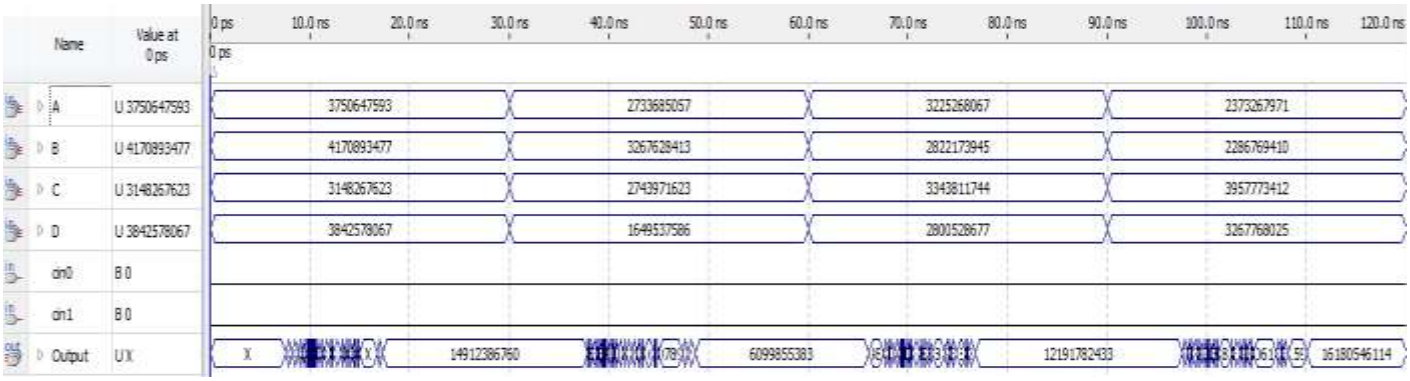


Fig. 15: Timing verification for the addition of four 32-bit numbers.

The utilization summary of LEs, critical path delay and power dissipation results for the addition of four 64-bit numbers are shown in [Fig. 16, 17 and 18] respectively. It is apparent that the CSA-CSeA method produce lower delay at the cost of area. For example with Mux-XOR majority voting, the CSA-CSeA approach produces 72.623 ns which is lower as compared with CSA-CLA (97.479 ns) and CSA-RCA (133.033 ns) at the cost of more LEs (384). Among the ten majority-voting circuits, it is also shown that the Mux-and-OR majority voting circuit offers lower delay as 100.965 ns and lower power dissipation as 187.79 mW using CSA-RCA approach.

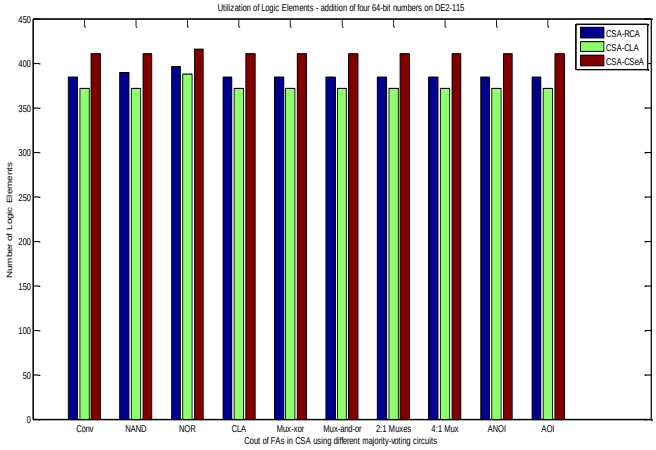


Fig. 16: Utilization of LEs for the addition of four 64-bit numbers.

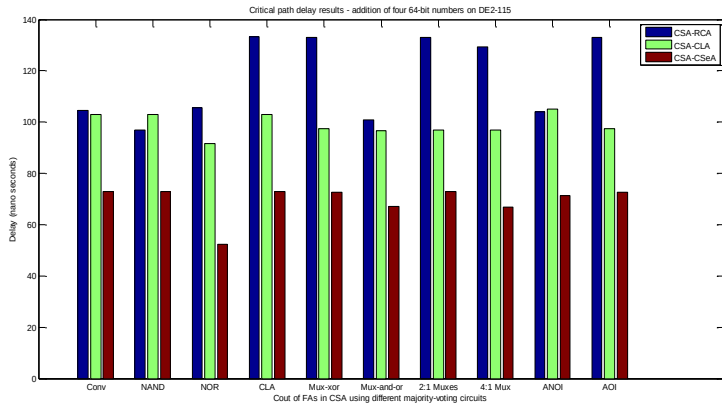


Fig. 17: Critical path delay for the addition of four 64-bit numbers.

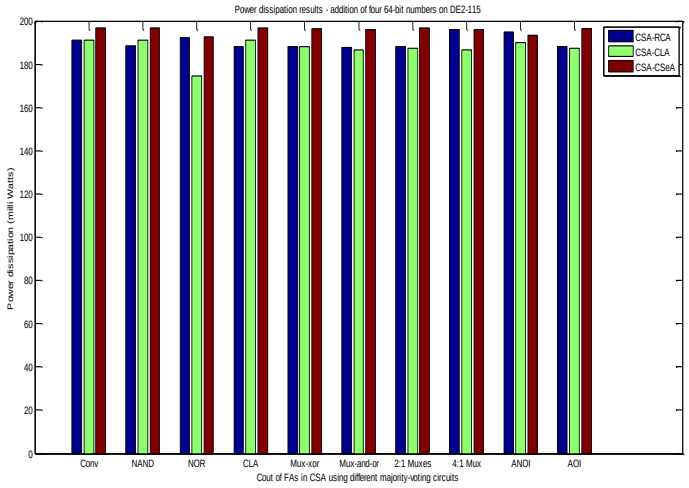


Fig. 18: Power dissipation results for the addition of four 64-bit numbers.

The research findings are summarized as follows. Among the different final stage circuits, the CSA-CSeA approach offers a lower delay, the CSA-CLA approach produces low power dissipation and the CSA-RCA approach consumes less number of logic elements. Among the majority-voting circuits, NAND based circuit offers lower delay and Mux-OR based circuit produces a better power dissipation.

CONCLUSION

This paper exemplifies the carry-save adder utility for the multi-operand addition with plenty of recipes with the majority-voting circuits instead of a regular full adder. Each design trade-off would be different from others. The best optimum circuit can be chosen depending on the applications and requirements. This work further can be extended to by modifying the final sum calculation stage using 4:2 compressor circuits to enhance the performance and better results. Similarly, the parity-circuit portion in a regular full adder can be considered to modify like using two half adders for example.

CONFLICT OF INTEREST
There is no conflict of interest.

ACKNOWLEDGEMENTS
We thank SASTRA University for providing financial support for this research work under the Research and Modernization fund – R&M/0034/SEEE-014/2013-14.

FINANCIAL DISCLOSURE
None

REFERENCE

[1] Dejan Markovic, Robert W.Broderson. DSP Architecture Design Essentials. US: Springer publishers, 2012.
[2] Uwe Meyer-Baese. Digital Signal Processing using Field Programmable Gate Arrays. Springer-Verlag Berlin Heidelberg, 2007.
[3] James E.Stine. Digital Computer Arithmetic Datapath Design using Verilog HDL. Springer publishers, 2012.
[4] Mark Owen. Practical Signal Processing. Cambridge University press, 2007.
[5] Real-time fault-tolerance with hot-standby topology for conditional sum adder. Atin Mukerjee and Anindya Sundar Dhar. 2015. Microelectronics Reliability, 55: 704-712.
[6] An implementation of area and power efficient digital FIR filter for hearing aid applications. Balaji, V.S. and Har Narayan Upadhyay. 2015. Journal of Optoelectronics and Advanced Materials, 9: 657-662.
[7] Uyemura JP.[2006] Introduction to VLSI Circuits and Systems. India: Wiley India publishers.
[8] Kiat-Seng Yeo, Kaushik Roy.[2009] Low Voltage, Low Power VLSI Subsystems. India: Mc-Graw Hill Education India
[9] FPGA Implementation of Fast Adder. Hamid MM.Kamboh and Shoab A.Khan. s.l: IEEE Press, 2012. International Conference on Computing and Convergence Technology, pp. 1324-1327.
[10] An Area Efficient and Low Power Multiplier using Modified Carry Save Adder for Parallel Multipliers. S.Murugeswari and S.Kaja Mohideen. s.l: Springer: International Joint Conference AIM/CCPE 2012, Communications in Computer and Information Science, pp. 446-450.
[11] CMOS VLSI Design of Low Power SRAM Cell Architectures with new TMR: A Layout Approach. Elamaran, V. and Har Narayan Upadhyay. 2015. Asian Journal of Scientific Research, 8:466-477.
[12] Majority function computation using different voter circuits – a comparative study. V Elamaran, et al. 2015. International Journal of Pharmacy and Technology, 7: 9764-9773.
[13] B Stephen Brown, V Zvonko. [2012]Fundamentals of Digital Logic with VHDL Design. Mc-Graw Hill publishers,
[14] R Woods, J McAllister, Dr. Ying Yi and Gaye Lightbody[2008]. FPGA-based Implementation of Signal Processing Systems. Wiley-Blackwell publishers,