International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869, Volume-3, Issue-4, April 2015

Comparative Analysis of Various Adders using VHDL

Komal M. Lineswala, Zalak M. Vyas

Abstract— **In today's world, lots of work is done in signal processing and communication field. Arithmetic units are one of the widely used components that are necessary for Digital Signal Processing (DSP) application. So, arithmetic units are used for high performance throughput in every application. . Adders serves as a building block for synthesis all other arithmetic operations. Thus, design of area efficient high speed adders are one of the most essential areas of research in VLSI. This paper presents comparative analysis of various adders such as Ripple Carry Adder (RCA), Carry Look-Ahead Adder (CLA), Carry Select Adder (CSLA), Cary Save Adder (CSA), Carry Skip Adder (CSKA), and Carry Increment Adder (CIA) in terms of speed and area. These adders are designed using VHDL. Then they are simulated and synthesized using Xilinx ISE 11.1 for Spartan 3E family device with speed grade -5.**

***Index Terms*—Carry Increment Adder, Carry Look Ahead Adder, Carry Save Adder, Carry Skip Adder, Carry Select Adder, Ripple Carry Adder.**

I. INTRODUCTION

Addition is by far the most fundamental arithmetic operation. It is one of the most commonly used components for a real-time digital signal processing application from application-specific DSP to general purpose processors. Therefore, a fast operation of a digital system is greatly influenced by the performance of the adders. They are also very significant component in digital systems because of their use in other digital operations such as subtraction, multiplication and division. Hence, improving performance of the digital adder would extensively advance the execution of binary operations inside a circuit comprised of such blocks. Different adder architectures have been proposed for speeding up the addition.

II. ADDER

A. *Half Adder*

A Half Adder (HA) is a combinational circuit used for adding two bits, a_i and b_i.

Manuscript received April 12, 2015.

Komal M. Lineswala, M-Tech Student, Department of Electronics and Communication Engineering, Uka Tarsadai University, Bardoli, India
Zalak M. Vyas, M-Tech Student, Department of Electronics and Communication Engineering, Uka Tarsadai University, Bardoli, India

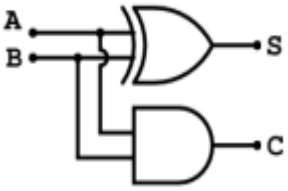


Fig. 1: Half Adder

This circuit has two outputs namely sum s_i and carry output c_o. Boolean expressions for sum and carry output are given:

$$s_i = a_i \; xor \; b_i \quad (1)$$

$$c_o = a_i \; and \; b_i \quad (2)$$

The critical path delay is one gate delay, and it corresponds to the length of any one of the two paths.

Table 1: Truth Table for Half Adder

INPUT		OUTPUT	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

B. *Full Adder*

A Full Adder (FA) is a combinational circuit used for adding three bits, a_i, b_i and c_i.

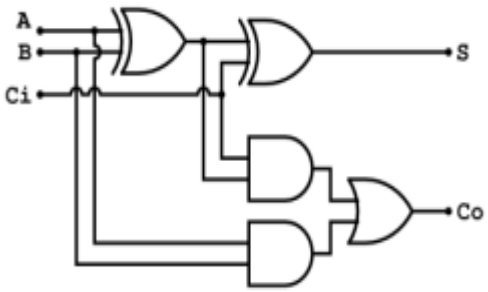


Fig. 2: Full Adder

This circuit has two outputs namely sum s_i and carry output c_o. Boolean expressions for sum and carry output are given:

$$s_i = a_i \; xor \; b_i \; xor \; c_i \quad (3)$$

$$c_o = (a_i \; and \; b_i) \; or \; (b_i \; and \; c_i) \; or \; (a_i \; and \; c_i) \quad (4)$$

A 1-bit full adder adds three 1-bit numbers, often written as A, B and C_i here A,B are the operands and C_i is a bit carried in.

Table 2: Truth Table for Full Adder

INPUT			OUTPUT	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A FA can also be constructed by cascading two HA. A and B inputs are connected to the input of first HA and the sum of first HA is connected as one input to second HA and second input to second HA is given through C_i. The final output sum of second HA is the final sum of FA and carry out of first and second HA is logically ORed to produce final carry.

III. COMPLEX ADDER

A. *Ripple Carry Adder*

A ripple adder that adds two N-bit operands requires N full adders. The speed varies linearly with the word length. The RCA implements the conventional way of adding two numbers[1]. In this architecture the operands are added bitwise from the least significant bits (LSBs) to the most significant (MSBs), adding at each stage the carry from the previous stage.

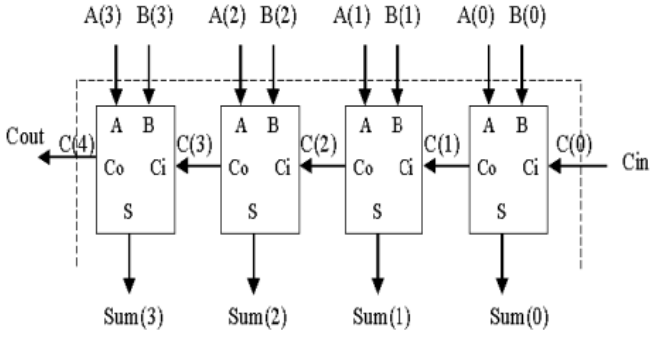


Fig. 3: 4-bit Ripple Carry Adder [**5**]

Thus the carry out from the FA at stage *i* goes into the FA at stage (*i* +1), and in this manner carry ripples from LSB to MSB (hence the name of ripple carry adder). The layout of a RCA is simple, which allows fast design time[2]. However, RCA is relatively slow, since each full adder must wait for the carry bit which is coming from the previous full adder.

B. *Carry Look-Ahead Adder*

In a carry look-ahead adder the carries entering all the bit positions of the adder are generated simultaneously by a carry look-ahead (CLA) generator; that is, computation of carries takes place in parallel with sum calculation. The speed of operation is independent of adder length. As the word length increases, the hardware organization of the addition technique gets complicated[3]. Hence adders with a large number of elements may require two or three levels of carry look-ahead stages.

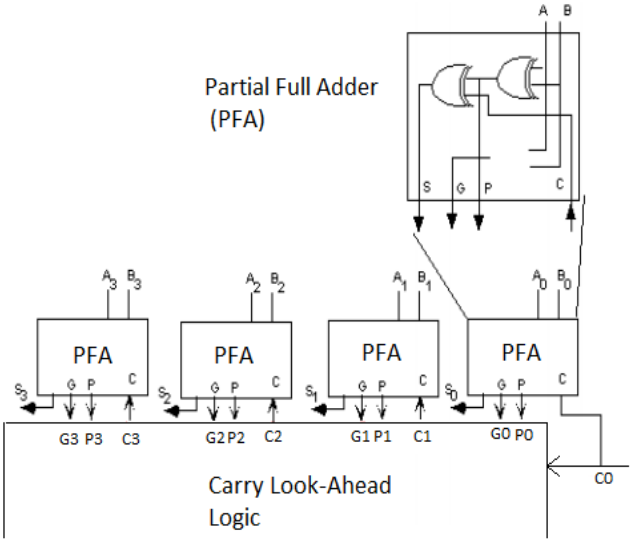


Fig. 4: 4-bit Carry Look-Ahead Adder [**5**]

A simple consideration of full adder logic identifies that a carryc_{i+1} is generated if ai=b_i=1, and a carry is propagated if either a_i or b_i is1. Carry look-ahead uses the concepts of generating (g_i) and propagating (p_i) carries. This can be written as [2]:

$$g_i = a_i \bullet b_i \quad (5)$$

$$p_i = a_i \oplus b_i \quad (6)$$

$$c_{i+1} = g_i + (p_i \bullet c_i) \quad (7)$$

$$s_i = c_i \oplus p_i \quad (8)$$

Thus a given stage generates a carry if g_i is TRUE and propagates a carry in to the next stage if p_i is TRUE. The CLA can be broken up in two modules: (1) The Partial Full Adder, PFA, which generates s_i, p_i and g_i. (2) The carry look-ahead logic, which generates the carry-out bits.

C. *Carry Select Adder*

The basic idea behind Carry Select Adder (CSLA) is to compute the result, that is, partial sum and carry out, in advance by anticipating all the possible values of carry in, that is, 0 and 1. Multiplexer stage is used to select the actual result, once the original input carry is known. Compared to RCA, CSLA have high speed and when compared to CLA, it has less hardware complexity.

It comprises of many blocks of RCA to generate partial sum for c_{in}=0 and 1. The carry out is calculated from the last stage. Advantage of using this adder with non-uniform RCA blocks is that it consumes less area and fastens the speed of execution.

To further reduce the area and power consumption, modified CSLA has been developed which make use of single RCA and a Binary to Excess-1 converter (BEC). BEC is used in place of c_{in}=1. N-bit RCA is replaced by N+1 bit BEC. Boolean expression for 4-bit BEC is given by[4]:

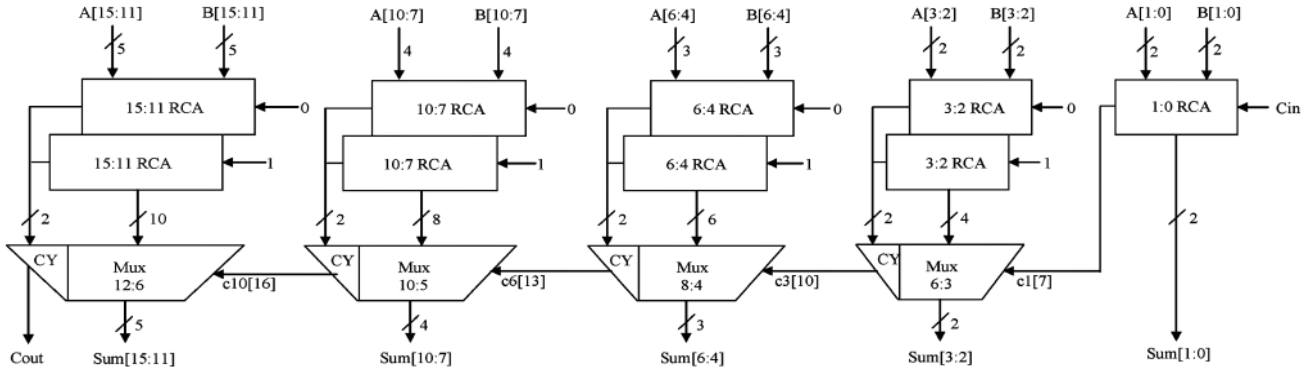


Fig. 5: Regular 16-bit Carry Select Adder [**4**]

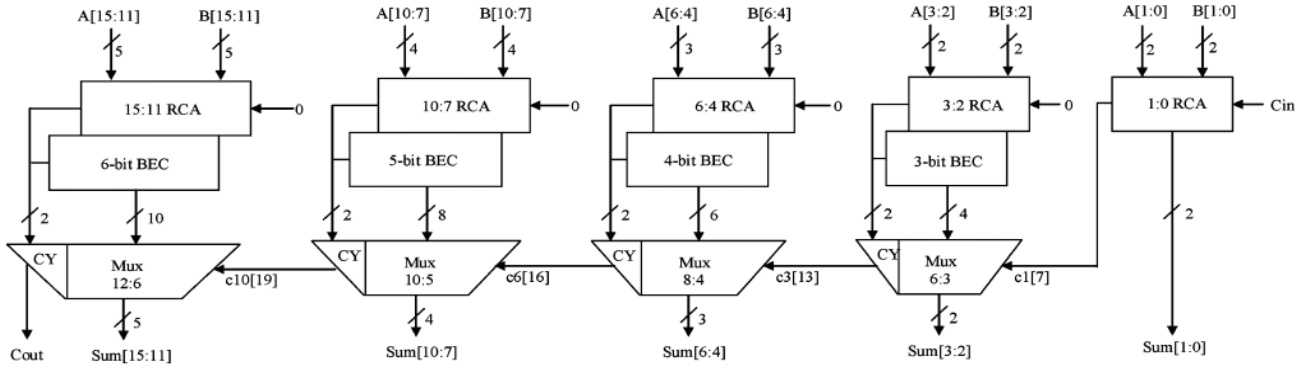


Fig. 6: Modified 16-bit Carry Select Adder [**4**]

$$X_0 = B_0 \tag{9}$$

$$X_1 = B_0 \, xor \, B_1 \tag{10}$$

$$X_2 = B_0 \, xor \, (B_1 \, and \, B_2) \tag{11}$$

$$X_3 = B_0 \, xor \, (B_1 \, and \, B_2 \, and \, B_3) \tag{12}$$

D. Carry Increment Adder

In Carry Select Adder[5], instead of computing two partial sums for each group and selecting the correct one, only one partial sum is calculated and incremented if necessary, according to the input carry.

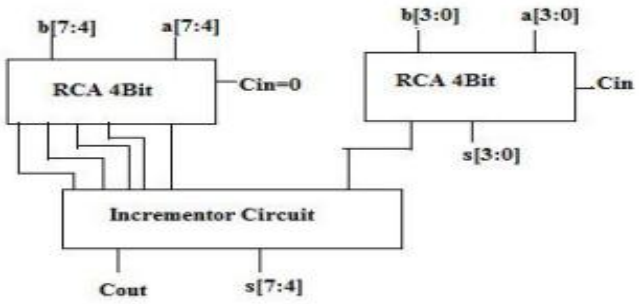


Fig. 7: 4-bit Carry Increment Adder [**5**]

Thus the second adder and the multiplexers in the carry-select scheme can be replaced by a much smaller incremental circuit and the modified architecture is Carry Increment Adder (CIA)[6]. For example, an 8-bit CIA comprises of two 4-bit RCA. The first block of RCA adds first 4-bits to produce 4-bit partial sum and a carry output.

Thus, first 4-bit of sum of CIA is directly obtained from first block of RCA. And the carry output of first RCA block is given as input to the c_{in} of incremental circuit. Incremental circuit consists of Half Adders. Hence, the partial sum obtained from the second RCA block is given to incremental circuit.

E. Carry Skip Adder

When addition of large number of bits is to take place, Carry skip adder is used as it is faster in speed than ripple carry adder. However a carry-skip adder consists of a simple ripple carry-adder with a special speed up carry chain.

In an N-bit carry skip adder, N-bits are divided into groups of k bits. The adder propagates all the carries simultaneously through the group's[7].Each group i compute P_i using the following relationship:

$$P_i = p_{i+3} \bullet p_{i+2} \bullet p_{i+1} \bullet p_i \tag{13}$$

Equation (6) is used to compute p_i for each bit location i. The strategy is that, if any group generates a carry, it passes it to the next group; but if the group does not generate its own carry owing to the arrangements of individual bits in the block, then it simply bypasses the carry from the previous block to its next block. This bypassing of a carry is handled by P_i. The carry skip adder can also be designed to work on unequal groups. Using the individual propagate values, the output from the AND gate is ORed with C_{i+4} to produce a stage output of [8]:

$$carry = c_{i+4} + \left(P_{i+4} \bullet c_i\right) \quad (14)$$

If P_i =0, then the carry-out of the group is determined by the value of c_{i+4}. However, if P_i =1 when the carry-in bit is c_i =1, then the group carry-in is automatically sent to the next group of adders. The name "carry-skip" is due to the fact that if the condition $P_i \bullet C_i$ is true and then the carry-in bit skips the block entirely.

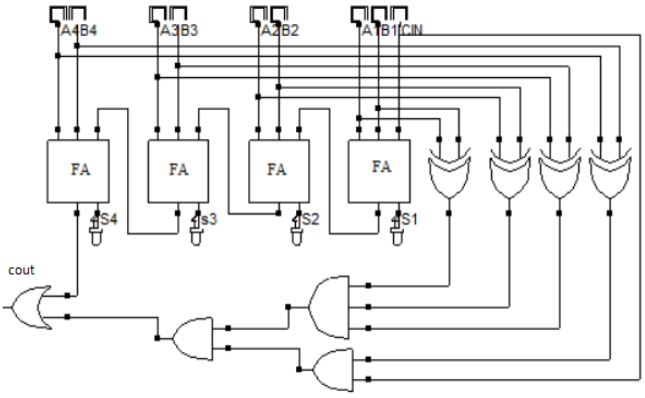


Fig. 8: 4-bit Carry Skip Adder [**6**]

F. *Carry Save Adder*

All the adders discussed above are used for adding two operands, and then propagate carries from one bit position to the next in computing the final sum and are collectively known as carry propagate adders (CPA's). But when three or more operands are to be added in a single cycle using two-operand adders, the time consuming carry-propagation must be repeated several times. If the number of operands is k, then carries have to propagate (k-1) times [9].

So, a better option is to use that first reduces the three numbers to two and then any CPA adds the two numbers to compute the final sum. From the timing and area perspective, the CSA is one of the most efficiently and widely used techniques for speeding up digital designs of signal processing systems dealing with multiple operands for addition and multiplication[2].

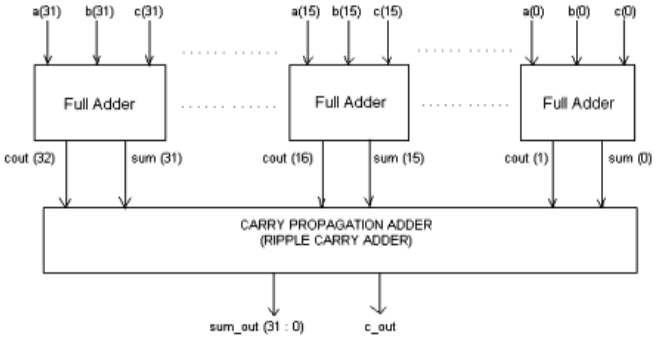


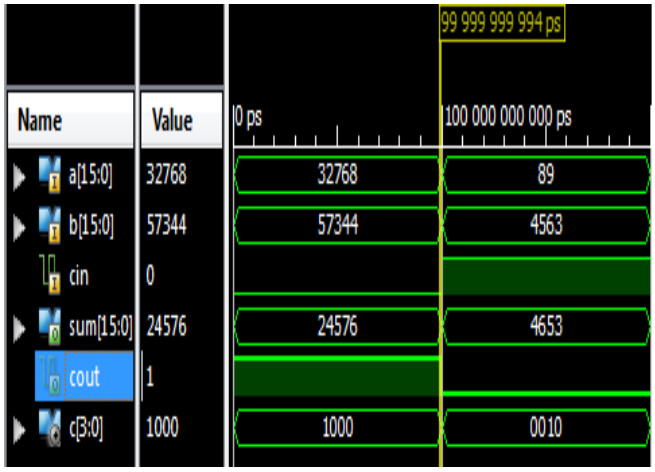
Fig. 9: 32-bit Carry Save Adder

In carry save addition, carry is propagated in last step, while in all the other steps a partial sum and a sequence of carries is computed separately. Thus, the basic CSA accepts three n-bit operands and generates two n-bit results, an n-bit partial sum and an n-bit carry. A second CSA accepts these two bit sequences and another input operand, and generates a new partial sum and carry. A CSA is therefore, capable of reducing the number of operands to be added from 3:2, so it is also called 3:2 compressor.

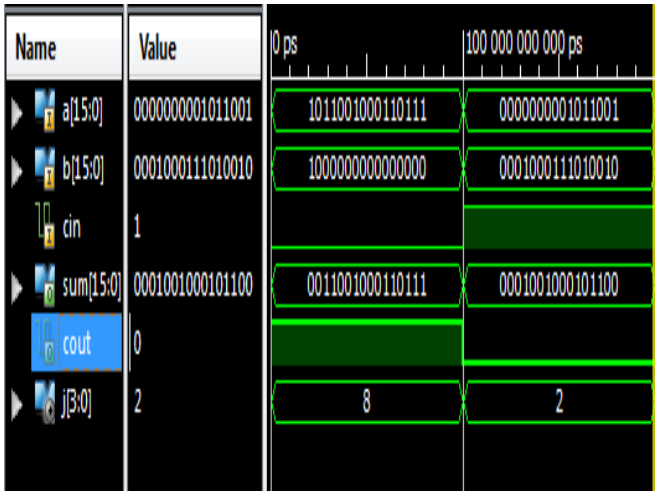
IV. SIMULATION RESULTS

The RTL code of each adder has been written in VHDL and Xilinx ISE 11.1 is used to simulate and synthesize the design. The adders use 16-bit values as shown in simulation waveforms.

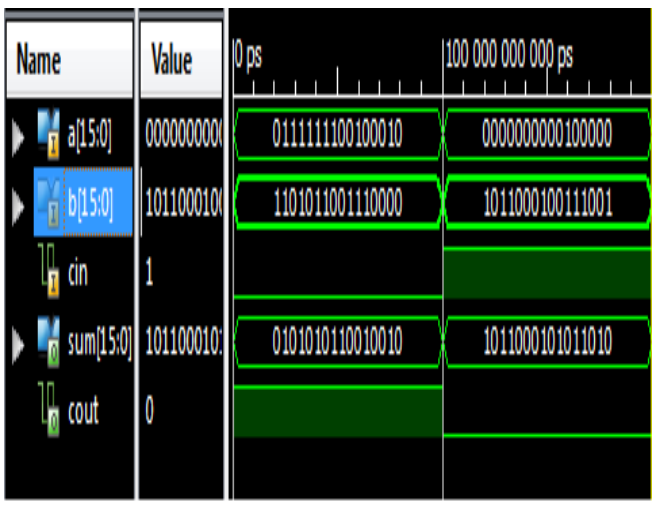
A. *Simulation Result of Ripple Carry Adder*



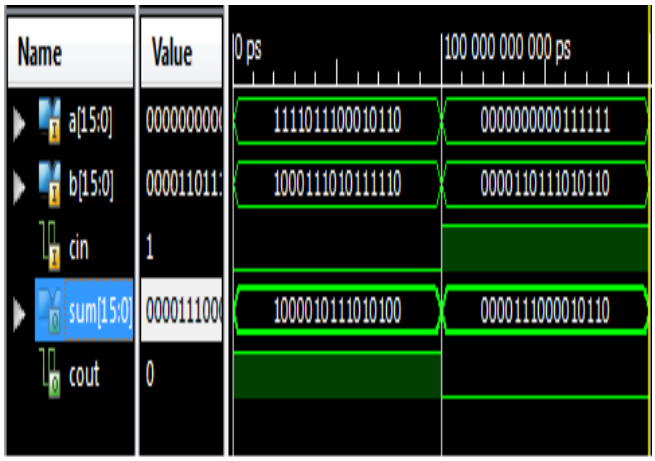
B. *Simulation Result of Carry Look-Ahead Adder*



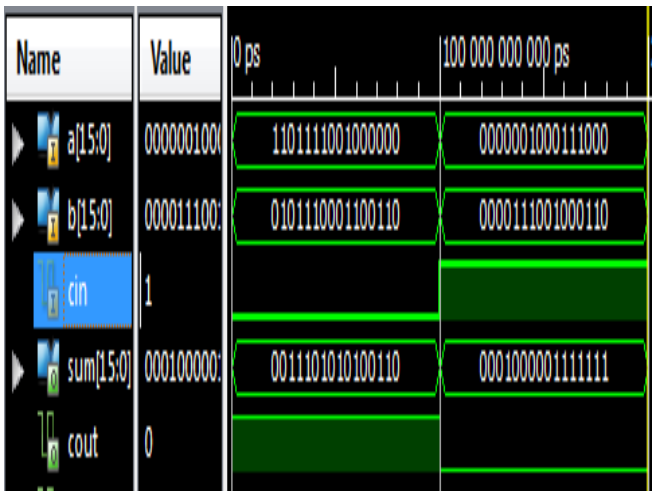
C. *Simulation Result of Carry Select Adder*



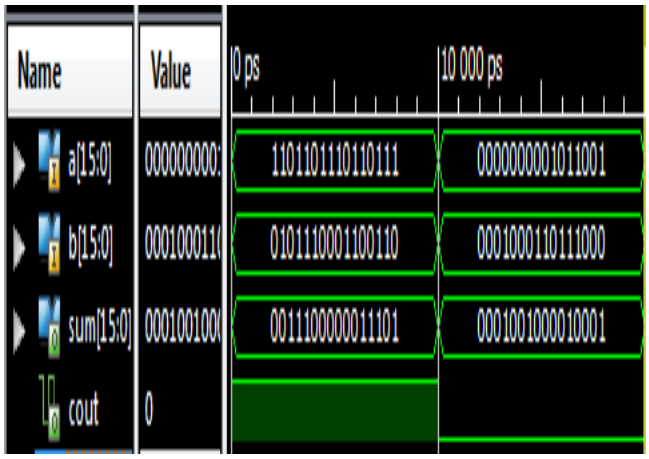
D. Simulation Result of Carry Skip Adder



E. Simulation Result of Carry Increment Adder



F. Simulation Result of Carry Save Adder



V. PERFORMANCE ANALYSIS

The performance analysis of various adders is done with respect to area and delay. It has been observed that RCA, CLA, CSLA, and CIA have better performance in terms of area (no. of slices). RCA and CLA occupies less memory (in KB) compared to all other adders. CSLA and CIA have lowest delay compared to other adders. However, CSA provides better area optimization when three or more operands are used and also there is increase in its speed of operation. From all the above analysis, it can be said that CIA provides optimized results in terms of area and delay compared to other adder topologies.

VI. CONCLUSION

This paper presents different adders that are modeled using VHDL. From the performance analysis, it can be concluded that Carry Increment Adder has better performance in terms of area and delay compared to other adders. However, the selection of adder is done on the basis of application.

Table 3: Simulation and Synthesis Results for Area and Delay of all 16-Bit Adders

S.No	Parameters	Ripple Carry Adder	Carry Look-Ahead Adder	Carry Select Adder	Carry Skip Adder	Carry Increment Adder	Carry Save Adder
1	No. of Slices	18/960	18/960	25/960	18/960	19/960	26/960
2	Levels of Logic	18	18	13	18	13	17
3	Memory Usage (KiloBytes)	20,5928	20,5928	20,8744	20,6952	20,7016	20,7528
4	Logic Delay (ns)	14.067	14.067	11.007	14.067	11.007	13.455
	Route Delay (ns)	7.623	7.623	5.455	7.311	5.439	8.263
	Total Delay (ns)	21.690	21.690	16.462	21.378	16.446	21.718

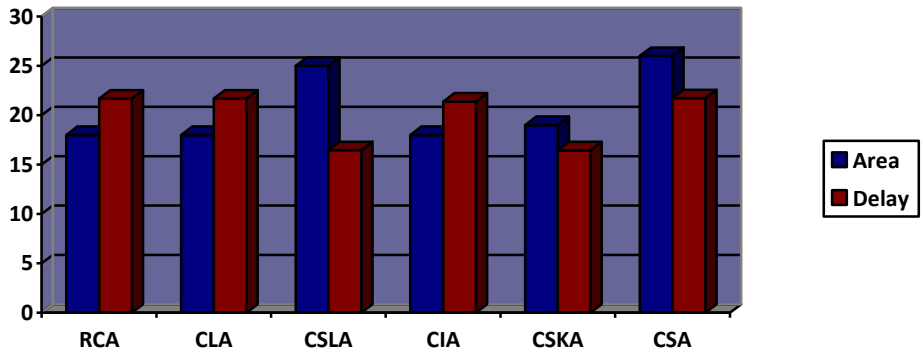


Fig. 10: Area and Delay Comparison of 16-Bit Adder

REFERENCES

[1] Parhami, Behrooz. Computer Arthmetic: Algorithms and Hardware Design, Oxford University Press, Inc., New York, 2000.
[2] Shoab Ahmed Khan, Digital Design Of Signal Processing Systems: A Practical Approach, John Wiley & Sons, Ltd, United Kindom, 2011.
[3] Prashant Gurjar, Rashmi Solanki, Mahendra Vucha, and Pooja Kansliwal, "VLSI Implementation of Adders for High Speed ALU," India Conference (INDICON), 2011 Annual IEEE, Hyderabad, pp.1 – 6, Dec. 2011.
[4] B. Ramkumar, and Harish M Kittur, "Low-Power and Area-Efficient Carry Select Adder," IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 20, pp. 371-375, February 2012.
[5] Maroju SaiKumar, and Dr. P. Samundiswary, "Design and Performance Analysis of Various Adders using Verilog," International Journal of Computer Science and Mobile Computing, vol. 2, pp. 128-138, September 2013.
[6] Themozhi.G, And Thenmozhi.V. "Propagation Delay Based Comparison Of Parallel Adders." Journal of Theoretical and Applied Information Technology, vol. 67, pp. 306-315, September 2014.
[7] M. Lehman, and N. Burla, "Skip techniques for high speed carry propagation in binary arithmetic circuits," IRE Transactions on Electronic Computer, 1961, vol. 10, pp. 691-698.
[8] A.K.Verma, P. Brisk, and P. Ienne, "Data flow transformations to maximize the use of carry save representation in arithmetic circuits," IEEE Transactions on Computer Aided Design of Integrated Circuits and System,. vol. 27, pp. 1761-1774, 2008.
[9] Israel Korean, Computer Arithmetic Algorithms, A K Peters,Ltd, South Avenue, Natick, 2002.