ENGINEERING IN ADVANCED
RESEARCHSCIENCE AND
TECHNOLOGY

ISSN 2352-8648
Vol.03, Issue.01
June-2022
Pages: -456-465

IMPLEMENTATION OF 64-BIT ALU USING MODIFIED CSLA

[1]V.S.Satish Chandra, [2]Y.Sai Sindhura, [3]M.Srinivasa Reddy ,[4]T.Ramanjaneyulu ,[5]MD. Shahabaz Sharief, [6] Mr. A. Manikanth,
[1,2,3,4,5] Under Graduate Student studying IV-B. Tech Electronics and Communication Engineering from Kallam Haranadhareddy Institute of Engineering & Technology, Guntur, A.P, India
[6]Assistant Professor of ECE Dept, Kallam Haranadhareddy Institute of Engineering & Technology, Guntur, A.P, India

Abstract:

Design of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. The CSLA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum [1]. However, the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input Cin=0 and Cout=1, then the final sum and carry are selected by the multiplexers (mux).The basic idea of this work is to use Binary to Excess-1 Converter (BEC) instead of RCA withCin=0 in the regular CSLA to achieve lower area and power consumption [2]–[4]. Thus this work uses a simple and efficient gate-level modification to significantly reduce the area and powerof the CSLA.

INTRODUCTION

Adders are of fundamental importance in a wide variety of digital systems. Many fast adders exist, but adding fast using low area and power is still challenging. The importance of a fast, low-cost binary adder in a digital system is difficult to overestimate. Not only adders used in every arithmetic operation, but they are also needed for computing the physical address in virtually every memory fetch operation in most modern CPUs. Many styles of adders exist like Ripple adders are the smallest but also the slowest. More recently, Carry-look-ahead and carry-select adders are very fast but far larger and consume much more power than ripple carry adder. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. The CSLA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum [1]. However, the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input Cin=0 and Cout=1, then the final sum and carry are selected by the multiplexers (mux).The basic idea of this work is to use Binary to Excess-1 Converter (BEC) instead of RCA with Cin=0 in the regular CSLA to achieve lower area and power consumption [2]– [4]. Thus this work uses a simple and efficient gate-level modification to significantly reduce the area and power of the CSLA. Arithmetic logical unit is a multiple operations; combinational-logic digital functions. It can perform a set of basic arithmetic operation and a set of logic operations. The ALU has a number of selection lines to select a particular operation in the unit. The selection lines are decoded with in the ALU so that K selection variables can specify upto to 2^k distinct operation.Figure shows the block diagram of modified ALU. The data input from A are combine input from B to generate an operation at output-F. The input s2 distinguish between arithmetic and logic operation. The two functions –select input s1 and s0 specify the particular arithmetic or logic operation to be generated. With three selection variables, it is possible to specify four arithmetic operations (with s2 in one state) and four logic operations (with s2 in other state) .The input and output carries have meaning only during an arithmetic operation. The input carry in the least significant position of an ALU is quite often use as a fourth selection variable that can double the number of arithmetic operation. In this way it is possible to generate four more operations for a total of eight arithmetic operations. The design of a typical ALU will be carried out in three stages. First, the design of the arithmetic section will be undertaken. Second, the design of the logic section will be considered. Finally, the arithmetic section will be modified so that it canperforms both arithmetic and logic operations

- Arithmetic Operations: Increment, Addition,Subtraction, Decrement, Transfer, Add with carry and Subtract with borrow.
- Logical Operations: AND, OR, XOR and Complement.

A. Conventional ALU

A conventional ALU use Ripple carry adder is used as the number of bit in ALU is increased. As the number of bit are increased the carry propagation delay increase.

- To avoid this delay carry select adder can be used as it calculate the sum for the two possible cases of carry Cin=1 and Cin=0in advance

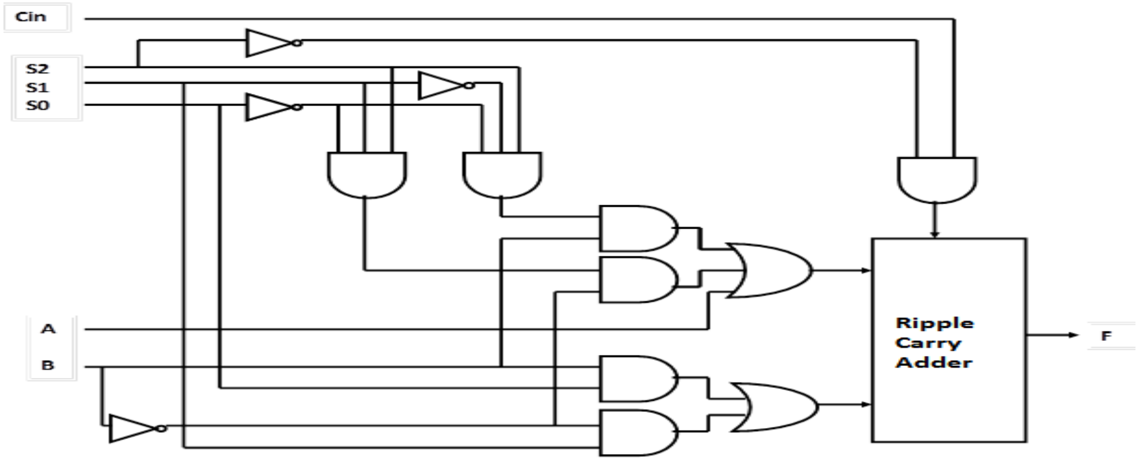


Figure-1:ALU using RCA

B. Modified ALU

In this type of ALU the additive unit in place of RCA is replaced by BEC-1 based CSLA. In conventional ALU as the carry ripples from one stage to another stage thus resulting in propagation delay of carry.The use of BEC-1 based CSLA avoids the carry propagation delay.

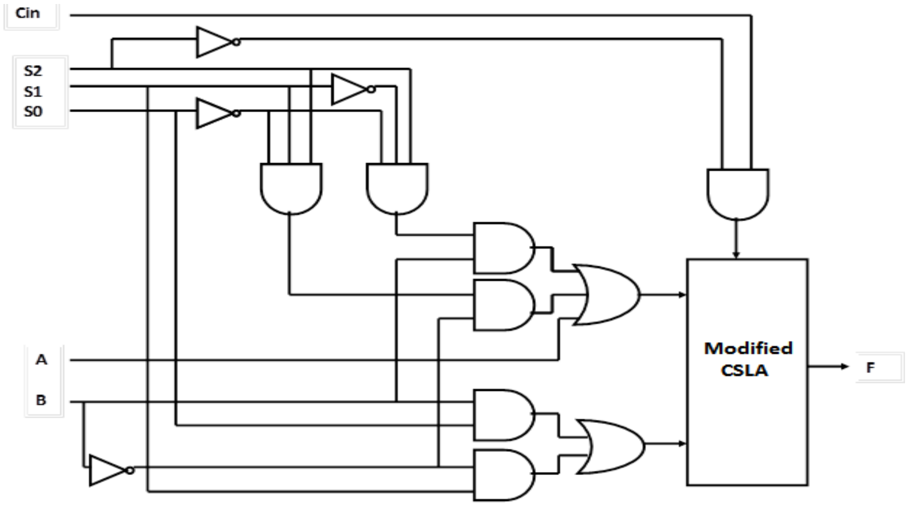


Figure-2:ALU using Modified CSLA

REGULAR CSLA

The carry-select adder (CSLA) provides a compromise between small area but longer delay ripple carry adder (RCA) and larger area with shorter delay carry look-ahead adder The existing carry-select adder generally consists of two Ripple Carry Adders (RCA) and a Multiplexer .Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two RCA).In order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one

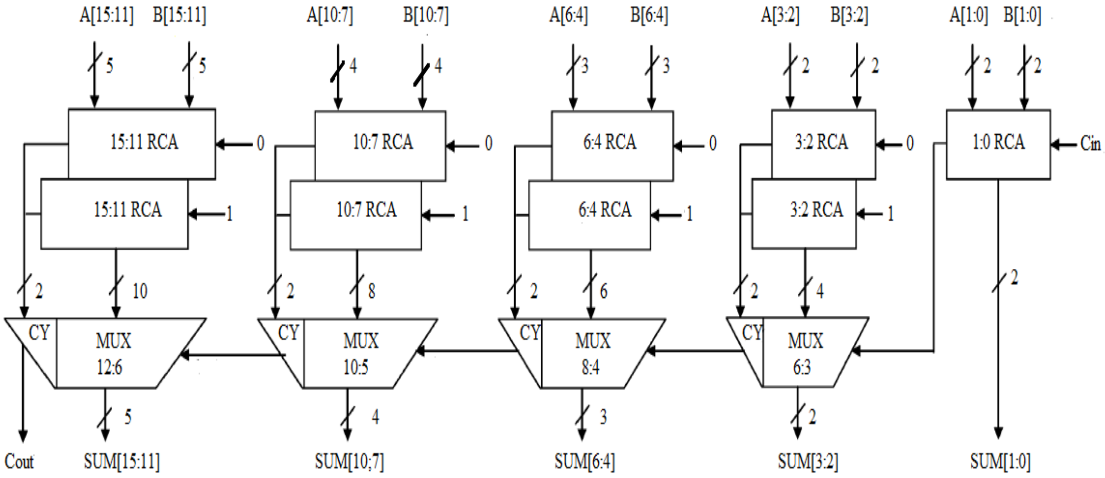


Figure-3:Regular 16-bit CSLA.

MODIFIED CSLA

As stated above the main idea behind modified model is to use BEC instead of the RCA with in order to reduce the area and powerconsumption of the regular CSLA. To replace the n-bit RCA, an n=1 bit BEC is required

- the BEC to reduce the RCA circuits.
- Here based on the carry input the MUX will be select corresponding input.
- In this design we give the MUX inputs are RCA output and BEC output.
- Compare to regular design the area of the design is less.

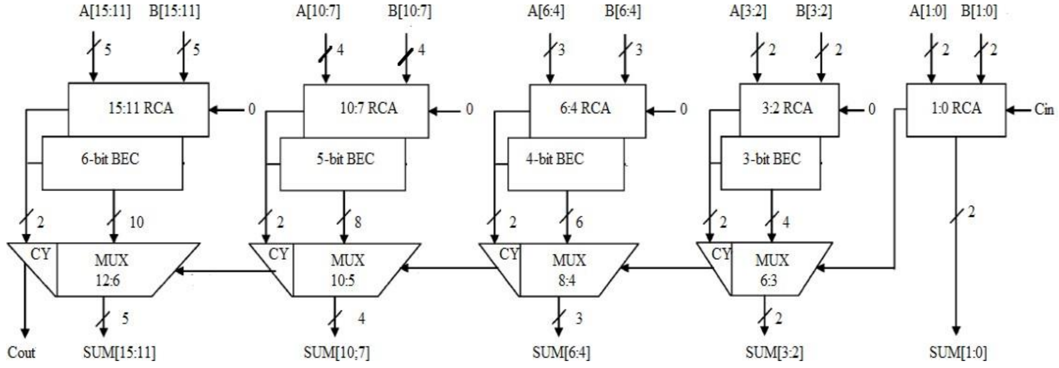


Figure-4:16-bit Modified CSLA

PROPOSED WORK:

i. Regular CSLA connected to ALU

Basically in figure 1 tells the ALU is connected to the Ripple carry adder.In this we.connected the CSLA to the ALU Block CSLA means two ripple carry adders connected in parallelly.one is RCA with carry zero another one is RCA with carry one. The below figure defines the 8-Bit ALU is connected to the Regular CSLA. Each stage output carry of full adder in RCA Block is ANDed with P [2] (third selection input). Theoutput of AND gate (one input as previous stage carry and other as P [2]) acts as a selection input for each multiplexer unit. F [7:0] indicates the output function of ALU

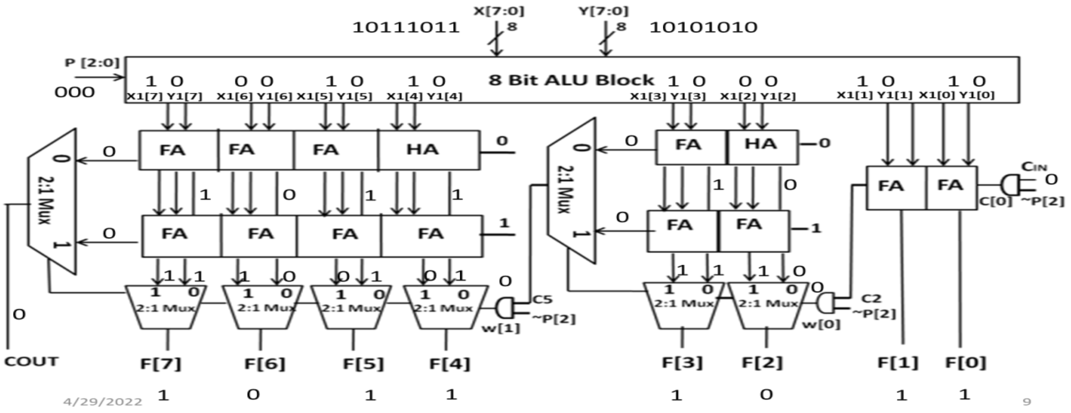


Figure-5:Regular CSLA is Connected to 8-Bit ALU Block

ii. Modified CSLA connected to ALU

Here we connected ALU Block To Modified CSLA.Modified CSLA means in Regular CSLA replace the RCA with Cin=1 to BEC-1 Converter. The BEC-1 converter is reduces the gates. Then area become efficient and resulting low power

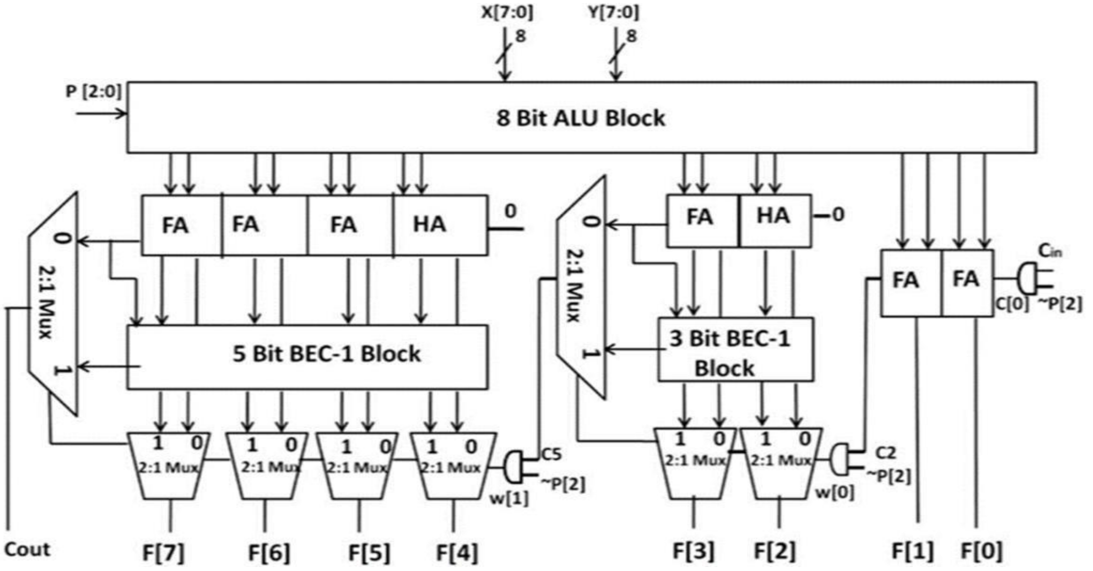


Figure-6:Modified CSLA connected to 8-Bit ALU Block

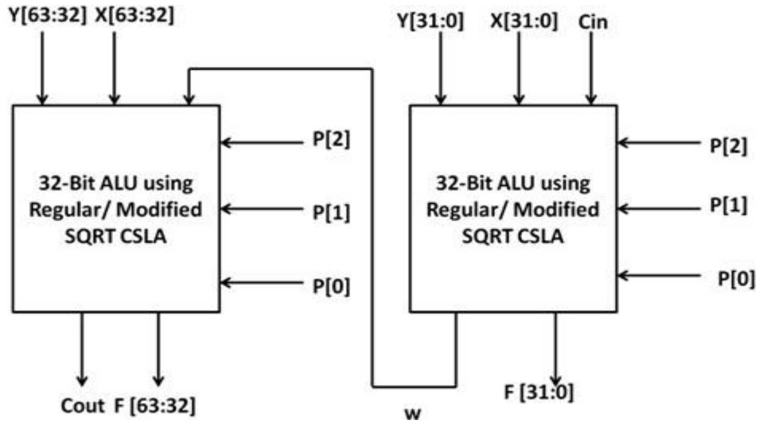
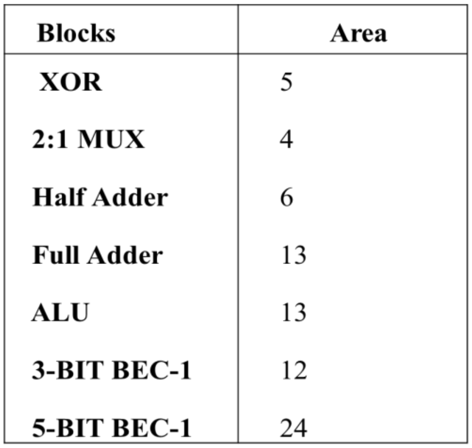

Y[63:32]
X[63:32]
32-Bit ALU using Regular/ Modified SQRT CSLA
P[2]
P[1]
P[0]
Cout
F [63:32]
Y[31:0]
X[31:0]
Cin
32-Bit ALU using Regular/ Modified SQRT CSLA
P[2]
P[1]
P[0]
F [31:0]
w

Figure 7:64-Bit ALU BLOCK

When computation for large number of bits in ALU is required, there is a need of cascading the adder circuit. These Cascaded adders however lead to Carry Propagation Delay (CPD) thereby affecting the speed of operation.if we want 64-Bit ALU is formed by cascading two 32-Bit as shown in figure 7



Blocks	Area
XOR	5
2:1 MUX	4
Half Adder	6
Full Adder	13
ALU	13
3-BIT BEC-1	12
5-BIT BEC-1	24

Table1:Gate Count for Different Blocks

iii. Regular CSLA Gate Count Calculation:-

Gate Count=225(HA+FA+MUX+ALU+AND)

Half Adder=12(2*6)

Full

Adder=156(12*13)

Multiplexer=32(8*4)

ALU=13(1*13)

AND=12(12*1)

iv. Modified CSLA Gate Count Calculation:-

Gate Count=179(HA+FA+MUX+ALU+AND+3-BiT+5-BiT)

Half Adder=12(2*6)

Full

Adder=78(6*13)

Multiplexer=32(8*

4)ALU=13(1*13)

AND=8(8*1)

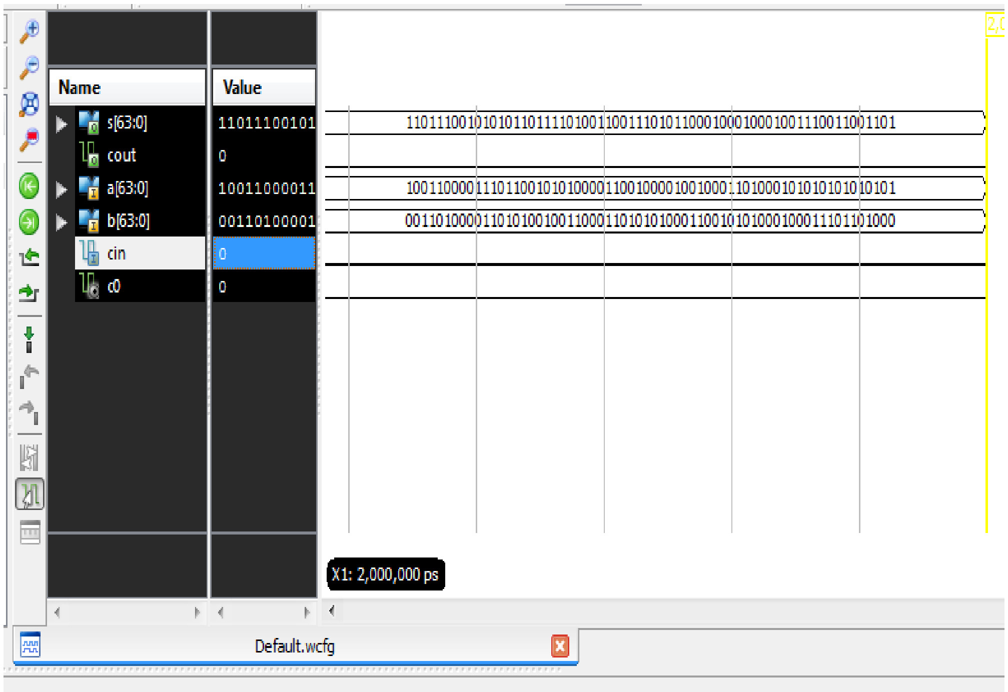
3-Bit BEC-1=12(1*12)

5-Bit BEC-1=24(1*24)

- It can be observed that the numbers of basic gates used are reducing from 225 to 179.

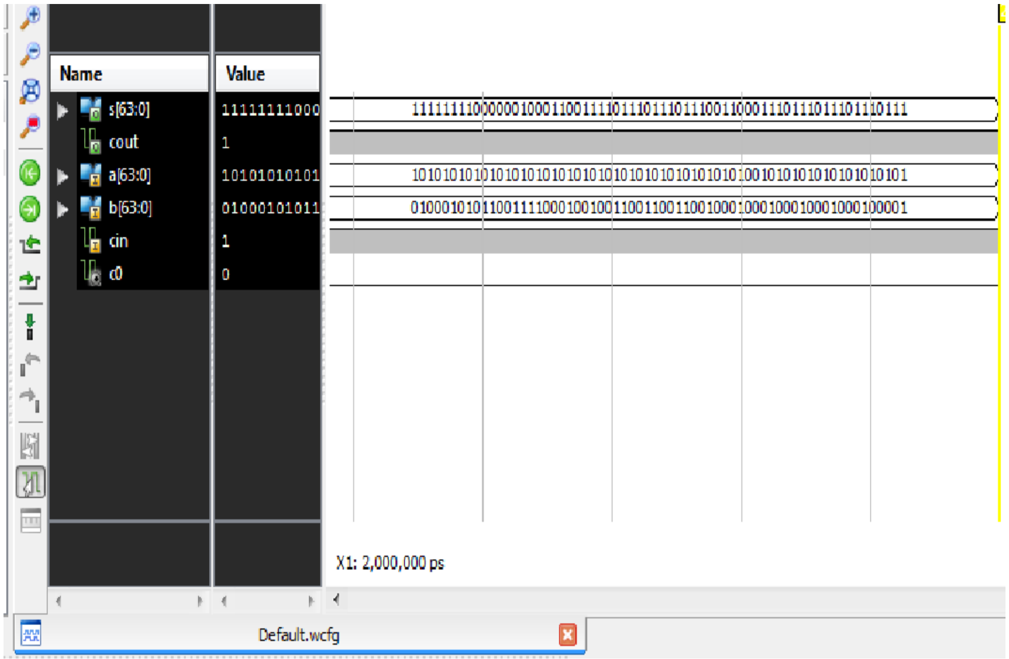
SIMULATION RESULTS

The implementation of 64-BIT Alu using Modified carry select adder is successfully done. are coded using Verilog HDL and simulated using I-SIM Simulator. The simulated waveforms for ALU using regular CSLA, Modified CSLA are shown in Figure 8, Figure 9 respectively.The software using for this project is Xilinx ISE 12.2 Design Suite



```
Name        Value
s[63:0]     11011100101     1101110010101011011110100110011101011000100010001001110011001101
cout        0
a[63:0]     10011000011     1001100001110110010101000011001000010010001101000101010101010101
b[63:0]     00110100001     0011010000110101001001100011010101000110010101000100011101101000
cin         0
c0          0
X1: 2,000,000 ps
Default.wcfg
```

Figure-8:Simulation of 64-Bit Regular-CSLA

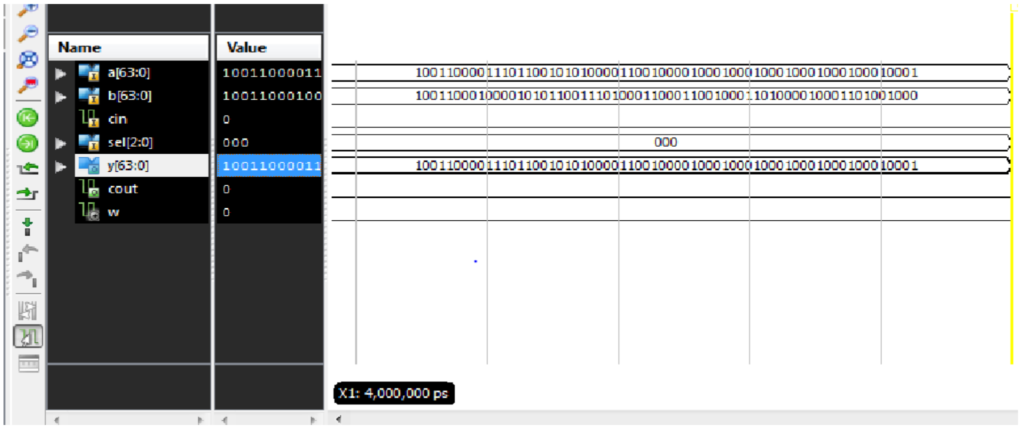


```
Name        Value
s[63:0]     11111111000     1111111100000001000110011110111011101110011000111011101110111 0111
cout        1
a[63:0]     10101010101     1010101010101010101010101010101010101010100101010101010101010101
b[63:0]     01000101011     0100010101100111100010010011001100110010001000100010001000100001
cin         1
c0          0
X1: 2,000,000 ps
Default.wcfg
```

Figure-9:Simulation of 64-Bit Modified-CSLA

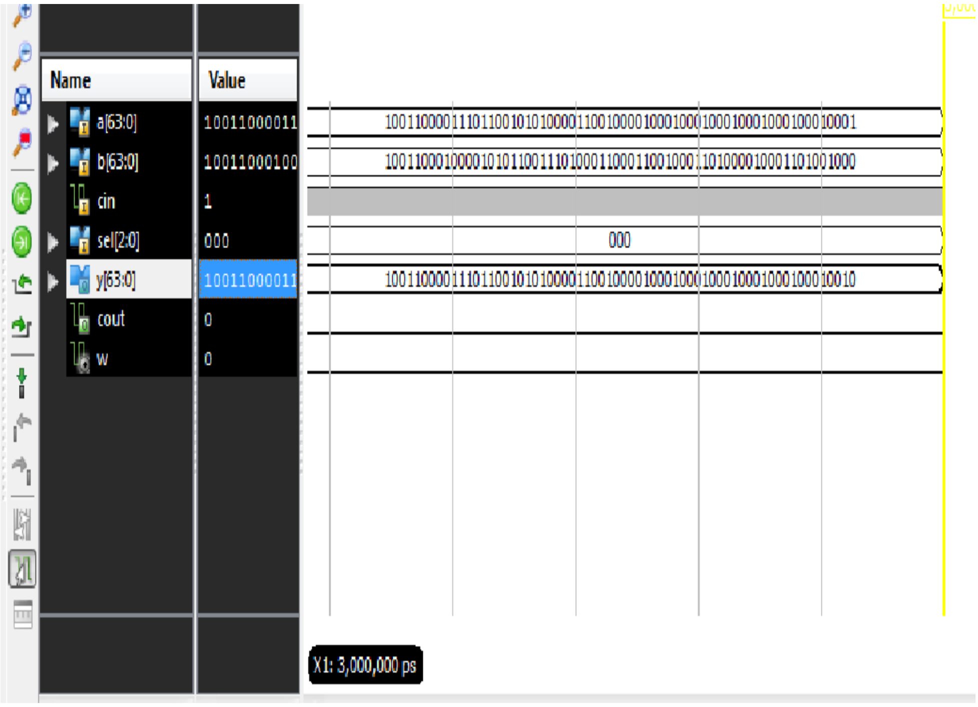
Table-2: The operations carried out by ALU during Simulation

	Function	Select			
s2	s1	s0	Cin	Output Equal	Function
0	0	0	0	F=A	Transfer
0	0	0	1	F=A+1	Increment
0	0	1	0	F=A+B	Addition
0	0	1	1	F=A+B +1	ADD with Carry
0	1	0	0	F=A-B-1	Sub with Borrow
0	1	0	1	F=A-B	SUBTRACT
0	1	1	0	F=A-1	Decrement
0	1	1	1	F=A	Transfer
1	0	0	X	F=A|B	OR
1	0	1	X	F=A^B	XOR
1	1	0	X	F=A&B	AND
1	1	1	X	F=A'	COMPLEMENT



```
Name        Value
a[63:0]     10011000011   1001100001110110010101000011001000010001000100010001000100010001
b[63:0]     10011000100   1001100010000101011001110100011000110010001101000010001101001000
cin         0
sel[2:0]    000           000
y[63:0]     10011000011   1001100001110110010101000011001000010001000100010001000100010001
cout        0
w           0
```

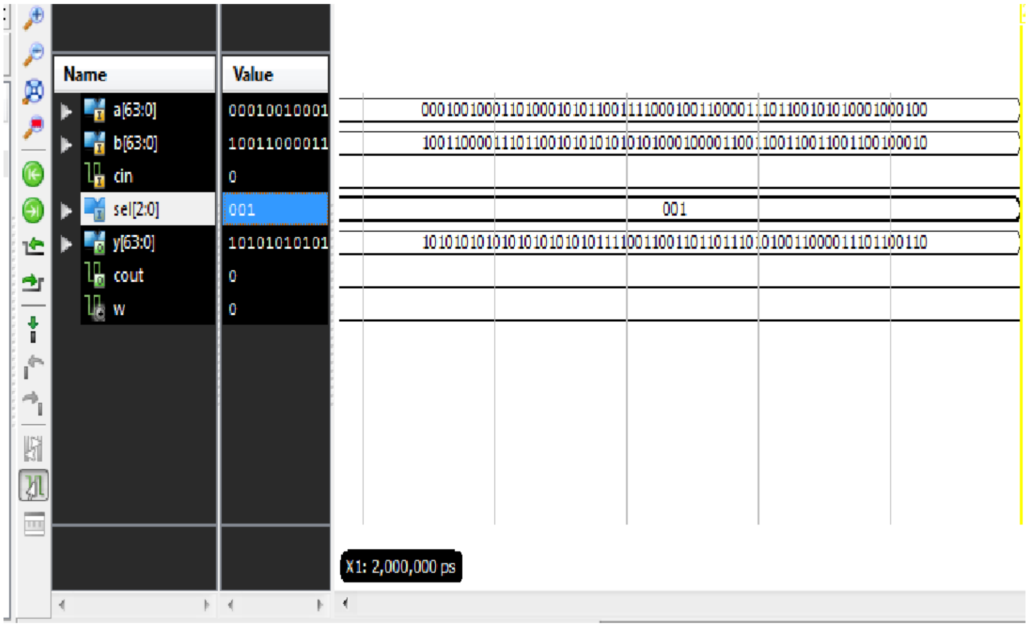
Fig-10:Simulated waveform of 64-Bit ALU performing TRANSFER operation.



Name	Value	
a[63:0]	10011000011	10011000011101100101010000110010000100010001000100010001000110001
b[63:0]	10011000100	1001100010000101011001110100011000110010001101000010001101001000
cin	1	
sel[2:0]	000	000
y[63:0]	10011000011	10011000011101100101010000110010000100010001000100010001000110010
cout	0	
w	0	

X1: 3,000,000 ps

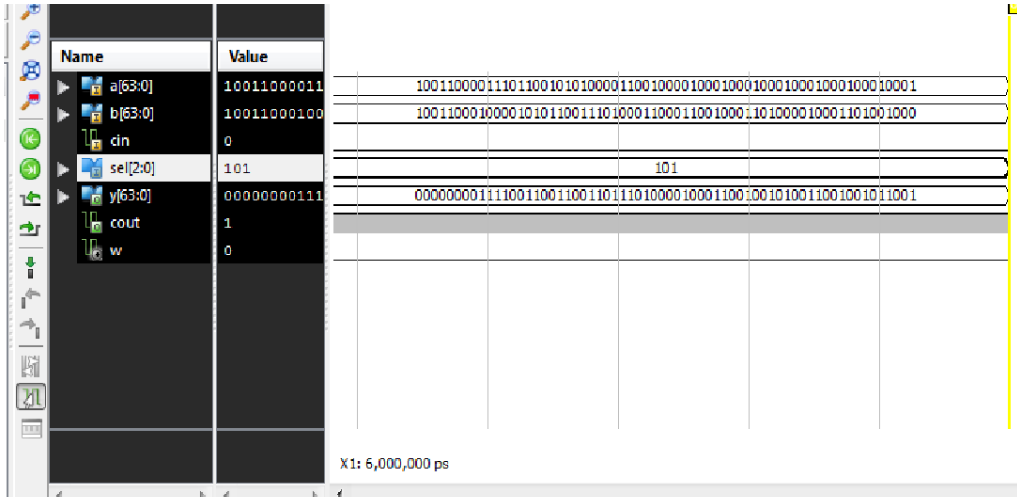
Fig-11:Simulated waveform of 64-Bit ALU performing INCREMENT operation.



Name	Value	
a[63:0]	00010010001	000100100011010001010110011100010011000011101100101010001000100
b[63:0]	10011000011	1001100001110110010101010101010001000011001100110011001100100010
cin	0	
sel[2:0]	001	001
y[63:0]	10101010101	101010101010101010101010111100110011011011101010011000011101100110
cout	0	
w	0	

X1: 2,000,000 ps

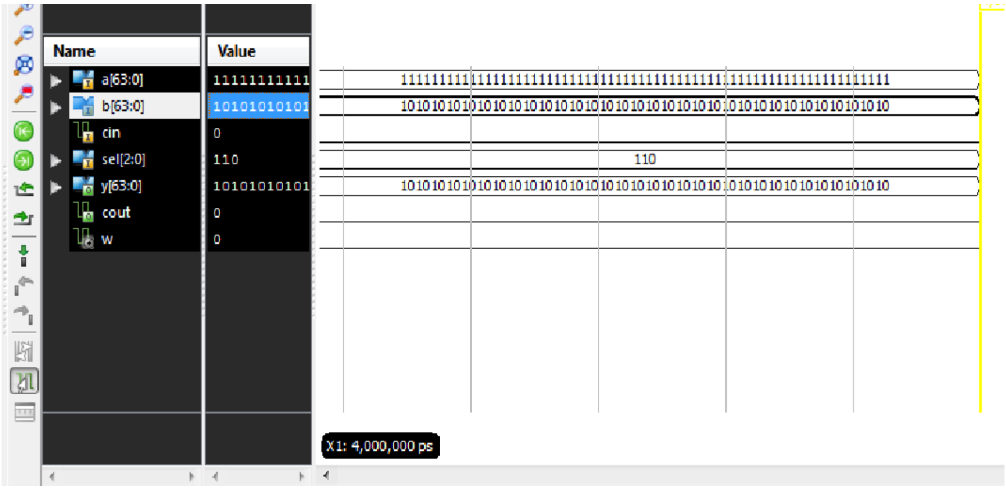
Fig-12: Simulated waveform of 64-BIT ALU performing ADDITION operation.



Name	Value	
a[63:0]	10011000011	1001100001110110010101000011001000010001000100010001000100010001
b[63:0]	10011000100	1001100010000101011001110100011000110010001101000010001101001000
cin	0	
sel[2:0]	101	101
y[63:0]	00000000111	0000000011110011001100110111010000100011001001010011001001011001
cout	1	
w	0	

X1: 6,000,000 ps

Fig-13:Simulated waveform of 64-Bit ALU performing XOR operation.



Name	Value	
a[63:0]	11111111111	1111111111111111111111111111111111111111111111111111111111111111
b[63:0]	10101010101	1010101010101010101010101010101010101010101010101010101010101010
cin	0	
sel[2:0]	110	110
y[63:0]	10101010101	1010101010101010101010101010101010101010101010101010101010101010
cout	0	
w	0	

X1: 4,000,000 ps

Fig-14:Simulated waveform of 64-Bit ALU performing AND operation

CONCLUSIONS

The first phase of the work primarily focuses on the designing of carry select adder. A simple approach is proposed In this report to reduce the area andpower of SQRT CSLA architecture. The reduced number of gates ofthis work offers the great advantage in the reduction of area and also thetotal power. Next the ALU is designed by using this carry select adder. All the designs are written in Verilog and simulated on ISIM simulator. The use of BEC-1 unit in modified CSLA reduces the number of gates. BEC-1 unit is simply circuit that add one, which replaces the RCA with Cin=1. The ALU performing both logical and arithmetic operation is designed and simulated. ALU up to 64-bit is designed.

REFERENCES

[1] B. Ramkumar and H.M. Kittur “Low-Power and Area-Efficient Carry Select Adder” IEEE Trans. VLSI, vol. 20, no. 2, pp.371–375, february 2012.

[2] O. J. Bedrij, “Carry-select adder” IRE Trans. Electron. Comput, pp.340–344, 1962.

[3] B. Ramkumar, H.M. Kittur, and P.M. Kannan “ASIC implementation of modified faster carry save adder” Eur. J. Sci. Res,vol. 42, no. 1, pp.53–58, 2010.

[4] T.Y. Ceiang and M.J. Hsiao, “Carry-select adder using single ripple carry adder” Electron.Lett., vol. 34, no. 22, pp. 2101–2103, Oct. 1998.

[5] Y. Kim and L.S. Kim, “64-bit carry-select adder with reduced area” Electron.Lett., vol. 37, no. 10, pp. 614–615, May 2001.

[6] J.M. Rabaey, Digtal Integrated Circuits—A Design Perspective.Upper Saddle River, NJ:Prentice-Hall, 2001.

[7] Y. He, C.H. Chang, and J. Gu, “An area efficient 64-bit square root carry-select adder for Low power applications” in Proc.IEEE Int. Symp.Circuits Syst., 2005, vol. 4, pp. 4082–4085.