ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

A Comparison of 128 bit Addition Using Ripple Carry Adder with Carry Lookahead Adder in Mentor EDA Tools

Kishore Prabhala[1] and Prof. Prabhandhakam Sangameswara Raju[2]

Research Scholar, EEE PhD, Rayalaseema University, Senior Member IEEE[1]

Principal, PLNM Degree College, Opposite Acharya Nagarjuna University Mens Hostel, Nagarjuna Nagar – 522 510, Guntur Dist., AP, India, 9177408565, prabhalakishore@gmail.com[1]

MSEE – Georgia Institute of Technology, GA, USA-1989, BSEE-Purdue University, W. Lafatette, IN, USA-1981[1]

Dept. of Electrical and Electronics Engineering, SVU Engineering College, Sri Venkateswara University, Tirupati – 517 502, AP[2]

Abstract: A complex process of adding two 128 bits has become a standard since 2018 as microprocessor have become to operate at 128 bits having a Tera byte by memory (1012) with over 1 GHz speed. There are two techniques that have been normally used so called Ripple Carry Adder and Carry Lookahead Adder. Using Mentor tools these two techniques will be explored and the results will be presented in simple fashion at 135 nm. But the technology has been rapidly moving from 135 nm to 90 nm to 65 nm.

Keywords: Addition, Ripple Carry Adder, Carry Lookahead Adder, Mentor Tools, 135nm Technology.

I. INTRODUCTION

A full adder performs addition of two inputs from A and B and another input called carry-in, Cin. There are two outputs are Sum and Carry-Out, Co. Since there are three inputs there will be eight combinations in binary and the input combinations would generate the output as truth table for a full adder (FA) shown in the table 1. A half adder does not have a carry-in as input. A half adder (HA) has two inputs and two outputs. A Karnaugh Map optimizes the equation from the truth table.

Table 1: Truth table & K-Map for Full Adder

A	B	Cin	Sum	Co
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

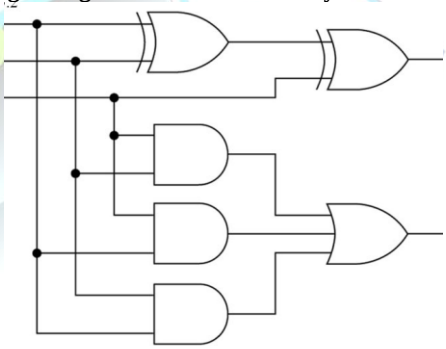
The equation for Sum from K-Map is

S = AB'Cin' + A'B'Cin + A'BCin' + ABCin
= (AB' + AB')Cin' + (A'B' + AB)Cin
= (A ⊕ B)Cin' + (A ⊙ B)Cin
= (A ⊕ B)Cin' + ((A ⊕ B)')Cin = A ⊕ B ⊕ Cin,

S = A ⊕ B ⊕ Cin, the symbol ⊕ stands for Exclusive OR, logically when inputs are not equal output will be ONE and the symbol ⊙ stands for Exclusive NOR. The logic diagram is shown in figure 1.1.

Carry out is Co = AB + ACin + BCin or AB + (A⊕B)Cin

Figure 1.1: Logic Diagram for Sum and Carry out of a Full Adder



A Verilog code for a Full adder logic is

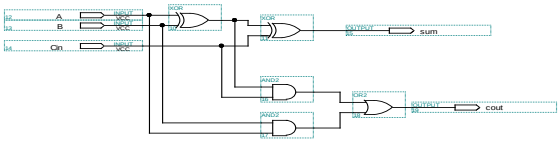
```
module fulladder(a,b,c,sum,carry);
  input a,b,c;
  output sum,carry;
  assign sum = a ^ b ^ c;
  assign carry = (a & b) | (b & c) | (c & a);
endmodule
```

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

A full adders with two exclusive OR gates is shown for the output Sum and two AND and a OR shown for Carry out in the figure 2 after the logic synthesis from Mentor Tools.

Figure 2: Logic Diagram created for a Full Adder in Mentor tools

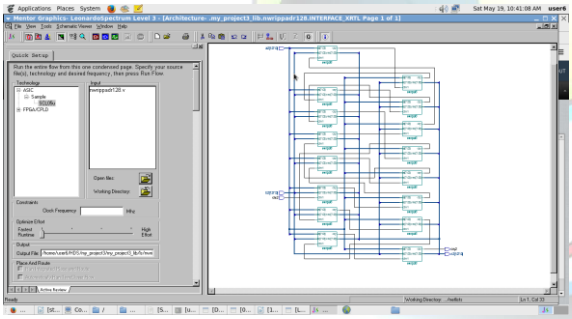


II. DESIGN OF RIPPLE CARRY ADDER WITH 128 BITS

2.1 Design of Ripple Carry Adder of 128 bits

The delay for the carry out would be based on the Carry out = AB + (A+B)Cin implemented with two AND gates in first stage and a OR gate in second stage so the total delay will be 2 gates and 8 gate delay for 4 bit RCA, 16 gate delay for 8 bit RCA, 32 gate delay for 16 bit RCA, 64 gate delay for 32 bit RCA, 128 gate delay for 64 bit RCA, so for a 128 bit RCA there will be 256 gate delay. The total logic gate count and ports is 11,274 in this design using Mentor tools.

Figure 2.1: Logic design of 128 bit RCA with 8 bit RCA of sixteen blocks



2.2: Behavioural Coding of Ripple Carry Adder of 128 bits

Verilog has been used to define behaviou of 128 bit RCA from 4 to 8 to 16 to 32 to 64 to 128 RCA and the following is the Verilog code for 128 bit Ripple Carry Adder.

Verilog Code for 128 bit Ripple Carry Adder

```
  input [127:0]a2;
  input [127:0]b2;
  input cin2;
  output [127:0]s2;
  output cry2;
  wire u2;
 newra64  z1(a2[63:0],b2[63:0],cin2,s2[63:0],u2);
 newra64  z2(a2[127:64],b2[127:64],u2,s2[127:64],cry2);
 endmodule
 module newra64 (a12,b12,cin12,sm12,cry12);
  input [63:0]a12;
  input [63:0]b12;
  input cin12;
  output [63:0]sm12;
  output cry12;
  wire u12;
 newrpa32  y1(a12[31:0],b12[31:0],cin12,sm12[31:0],u12);
 newrpa32
y2(a12[63:32],b12[63:32],u12,sm12[63:32],cry12);
 endmodule
 module newrpa32 (a11,b11,cin111,sm11,cry11);
  input [31:0]a11;
  input [31:0]b11;
  input cin111;
  output [31:0]sm11;
  output cry11;
  wire u11;
 nwrp16  y1(a11[15:0],b11[15:0],cin111,sm11[15:0],u11);
 nwrp16  y2(a11[31:16],b11[31:16],u11,sm11[31:16],cry11);
 endmodule
 module nwrp16 (a1,b1,cin11,sm1,cry1);
  input [15:0]a1;
  input [15:0]b1;
  input cin11;
  output [15:0]sm1;
  output cry1;
  wire u1;
  nwrpa8  l1(a1[7:0],b1[7:0],cin11,sm1[7:0],u1);
  nwrpa8  l2(a1[15:8],b1[15:8],u1,sm1[15:8],cry1);
  endmodule
 module nwrpa8 (a,b,cin1,sm,cry);
  input [7:0]a;
  input [7:0]b;
  input cin1;
  output [7:0]sm;
  output cry;
  wire w1;
  rpa4 p1(a[3:0],b[3:0],cin1,sm[3:0],w1);
  rpa4 p2(a[7:4],b[7:4],w1,sm[7:4],cry);
   endmodule
 module rpa4(a,b,cin,s,co);
  input [3:0]a;
  input [3:0]b;
  input cin;
  output [3:0]s;
```

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

```
output co;
wire c1,c2,c3;
  fularqa m1(a[0],b[0],cin,s[0],c1);
fularqa m2(a[1],b[1],c1,s[1],c2);
fularqa m3(a[2],b[2],c2,s[2],c3);
fularqa m4(a[3],b[3],c3,s[3],co);
endmodule
module fularqa(a,b,c,s,ca);
  input a,b,c;
  output s,ca;
  assign s=a^b^c;
  assign ca=(a&b)|(b&c)|(c&a);
endmodule
```

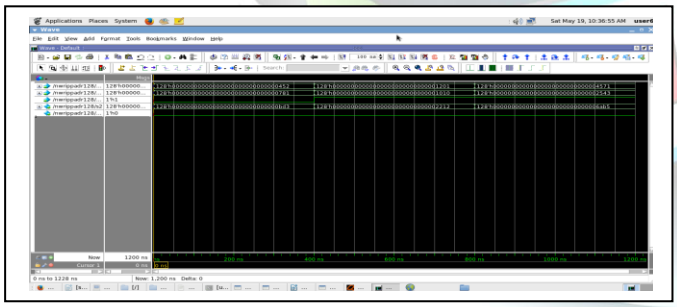
2.3 Simulation of Ripple Carry Adder of 128 bits

Mentor Logical Simulation begins with a set of inputs to the 128 bit RCA and verifies the outputs of 128 bit of Sum and a Carryout have to be checked and make sure they are correct. So a set of test benches are created to find the validity of the design.

A simple test bench is shows two different numbers at A and B inputs with Carry-in and find the output, as shown in figure 2.2.

A = 452, B = 781 and Cin = 0, Output is BD3, Carry out = 0,
A = 1201, B = 1010 and Cin = 1, Output is 2212, Carry out = 0,
A = 4571, B = 2543 and Cin = 1, Output is 6AB5, Carry out = 0

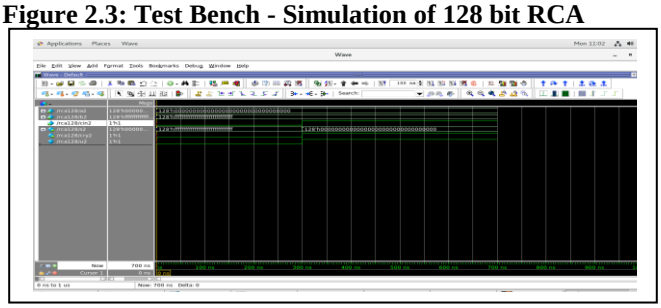
Figure 2.2: Test Bench - Simulation of 128 bit RCA



A standard test bench is check all Fs as an input with a 1 at B or Carry-in so Carry out moves at each adder stage of 128 adders in 128 bit RCA. Each input has to checked for One and Zero as shown in figure 2.3.

A = 0, B = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF and Cin = 0,
Output is FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF, Carry out = 0
A = 0, B = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF and Cin = 1, Output is 0, Carry out = 1.

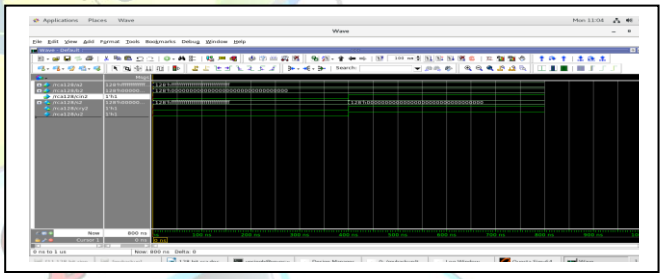
Figure 2.3: Test Bench - Simulation of 128 bit RCA



Another test bench is to check for Zero and One for inputs from previous check.

A = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF, B = 0 and Cin = 0, Output is FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF, Carry out = 0
A = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF, B = 0, and Cin = 1, Output is 0, Carry out = 1, shown in figure 2.4.

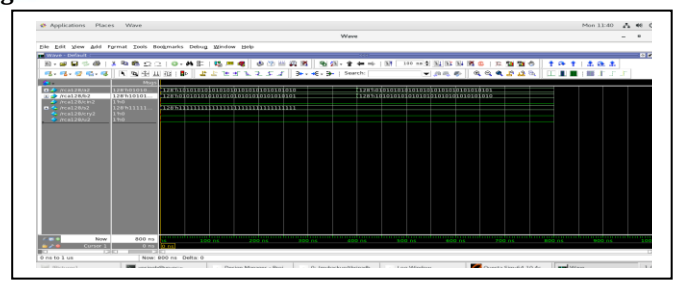
Figure 2.4: Test Bench - Simulation of 128 bit RCA



The test bench is alternative Ones and Zeros at one input and Zeros with Ones at other input.

A = 1010 1010 1010 1010 1010 1010 1010 1010 , B = 0101 0101 0101 0101 0101 0101 0101 0101 and Cin = 0, Output is 1111 1111 1111 1111 1111 1111 1111 1111, Carry out = 0
A = 0101 0101 0101 0101 0101 0101 0101 0101, B = 1010 1010 1010 1010 1010 1010 1010 1010 and Cin = 0, Output is 1111 1111 1111 1111 1111 1111 1111 1111, Carry out = 0, shown in figure 2.5.

Figure 2.5: Test Bench - Simulation of 128 bit RCA



ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

3. Design and Simulation of 128 bit adder with CLA

A 4 bit CLA has been explained with details for the design of 128 bit CLA in the next section with figure 3.1 and 3.2 and simulated to verify the logic. A total of thirty two 4 bit CLAs have been used.

3.1 Logic Design of Carry Lookahead Adder of 128 bits

Carry Look Ahead (CLA) is new technique in a adder design based on the principle of looking carry of lower four bits or eight bits. This adder reduces the carry delay by reducing the number of gates through which a carry signal must propagate. This adder design brings two internal inputs known as "Generation" (G) and "Propagation" (P) at each stage. Carry is generated when inputs A and B are "High" without carry-in. Similarly if A or B is High the carry is propagated if Carry-in is "High". The generation values and propagation values are computed based on the carry out equation which is

Sum = A ⊕ B ⊕ Carryin, Cout = AB + Cin(A+B) eq. 2.1

Generate, G = AB eq. 2.2

Propagate, P = A + B eq. 2.3

For a 4 bit CLA, the Generate and Propagate terms are

$G_0 = A_0B_0$ and $P_0 = A_0+B_0$, $G_1 = A_1B_1$ and $P_1 = A_1+B_1$, $G_2 = A_2B_2$, and $P_2 = A_2+B_2$, and $G_3 = A_3B_3$, and $P_3 = A_3+B_3$

Substituting the each term from 0 stage to 2 stage in the 3rd stage, the final carry-out can be obtained.

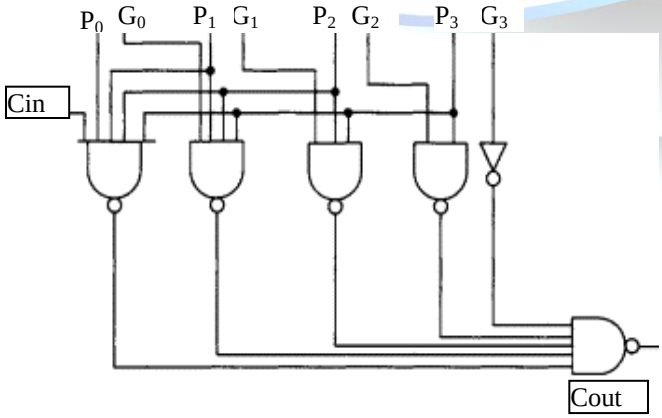
C[1] = G0 + P0Cin eq. 2.4

C[2] = G1 + P1 G0 + P1 P0 Cin eq. 2.5

C[3] = G2 + P2G1+P2P1G0+P2P1P0Cin eq. 2.6

Cout=G3+P3G2+P3P2G1+P3P2P1G0+P3P2P1P0Cin eq. 2.7

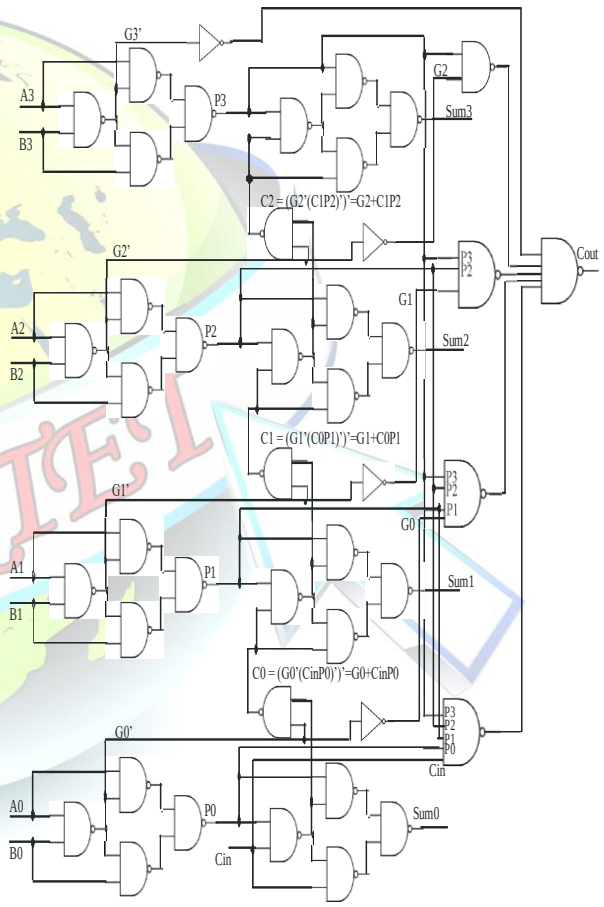
Figure 3.1: Carry Lookahead Logic (CLL) for Carry Out for a 4 bit adder



Carry generation from each stage is calculated from the above equations and finally carry out at the end of 4 bits which shown in as Carry Lookahead Logic in the figure 3.1.

Full adder has been defined with four NAND gates as EXOR in two stages with Generate (G) coming from first NAND (it is G' or G Bar) and Propagate (P) coming from first EXOR output, A and B as inputs. This is shown in figure 3.2. The input carry-in will feed the second EXOR gate designed after four NANDs in the first stage shown in the bottom of the figure 3.2.

Figure 3.2: Carry Lookahead Adder with Carry Out for a 4 bit adder



A 4 bit CLA has nine inputs: Four A0-3, Four B0-3 and a Carry-in. It has with five outputs; Four Sum S0-3 and Carry-out. This Carry out becomes input as Carry-in in the next stage. Two 4 bit CLAs have been used to build an 8 bit, two 8 bit CLAs used for 16 bit, two 16 bit CLAs used for 32 bit, two 32 bit CLAs used for 64 bit and two 64 bit CLAs used for 128 bit CLA as shown in figure 3.3 and figure 3.4.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

Figure 3.3: Logic design of 128 bit CLA with 4 bit CLA of sixteen blocks, first page

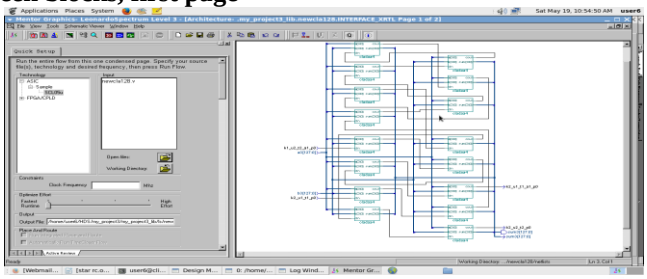
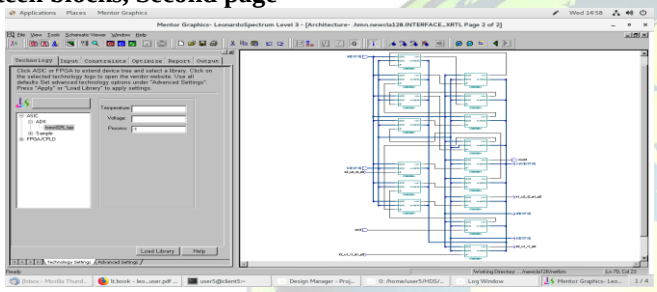


Figure 3.4: Logic design of 128 bit CLA with 4 bit CLA of sixteen blocks, Second page



3.2 Verilog Code for 128 bit Carry Lookahead Adder

```
module newcla128 (a3,b3,cin3,sum3,cout3);
        input[127:0] a3,b3;
        input cin3;
        output [127:0] sum3;
        output cout3;
        wire p0,cout3;
        wire [127:0]sum3;
         newclar64 k1(a3[63:0],b3[63:0],cin3,sum3[63:0],p0);
         newclar64
k2(a3[127:64],b3[127:64],p0,sum3[127:64],cout3);
endmodule
module newclar64 (a23,b23,cin23,sum23,cout23);
        input[63:0] a23,b23;
        input cin23;
        output [63:0] sum23;
        output cout23;
        wire p0,cout23;
        wire [63:0]sum23;
         nwclaa32
u1(a23[31:0],b23[31:0],cin23,sum23[31:0],p0);
         nwclaa32
u2(a23[63:32],b23[63:32],p0,sum23[63:32],cout23);
endmodule
module nwclaa32 (a2,b2,cin2,sum2,cout2);
        input[31:0] a2,b2;
        input cin2;
        output [31:0] sum2;
        output cout2;
        wire p0,cout2;
        wire [31:0]sum2;
        nwclra16 t1(a2[15:0],b2[15:0],cin2,sum2[15:0],p0);
        nwclra16
t2(a2[31:16],b2[31:16],p0,sum2[31:16],cout2);
endmodule

module nwclra16(a12,b12,cin12,sum12,cout12);
        input[15:0] a12,b12;
        input cin12;
        output [15:0] sum12;
        output cout12;
        wire p0,cout12;
        wire [15:0]sum12;
nwclra8 a1(a12[7:0],b12[7:0],cin12,sum12[7:0],p0);
nwclra8a2(a12[15:8],b12[15:8],p0,sum12[15:8],cout12);
endmodule
module nwclra8 (a1,b1,cin1,sum1,cout1);
        input[7:0] a1,b1;
        input cin1;
        output [7:0] sum1;
        output cout1;
        wire p0,cout1;
        wire [7:0]sum1;
        cladaa4 c1(a1[3:0],b1[3:0],cin1,sum1[3:0],p0);
        cladaa4 c2(a1[7:4],b1[7:4],p0,sum1[7:4],cout1);
endmodule
module cladaa4 (a,b,cin,sum,cout);
        input[3:0] a,b;
        input cin;
        output [3:0] sum;
        output cout;
        wire p0,p1,p2,p3,g0,g1,g2,g3,c0,c1,c2,c3,c4;
        assign p0=a[0]^b[0];
        assign p1=a[1]^b[1];
        assign    p2=a[2]^b[2];
        assign    p3=a[3]^b[3];
        assign   g0=a[0]&b[0];
        assign     g1=a[1]&b[1];
        assign     g2=a[2]&b[2];
        assign     g3=a[3]&b[3];
```

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

```
        assign   c0=cin;
        assign  c1=g0|(p0&cin);
        assign     c2=g1|(p1&g0)|(p1&p0&cin);
        assign
c3=g2|(p2&g1)|(p2&p1&g0)|(p1&p1&p0&cin);
        assign
c4=g3|(p3&g2)|(p3&p2&g1)|(p3&p2&p1&g0)|(p3&p2&p1&p0
&cin);
        assign   sum[0]=p0^c0;
        assign     sum[1]=p1^c1;
        assign     sum[2]=p2^c2;
        assign     sum[3]=p3^c3;
        assign   cout=c4;
   endmodule
```

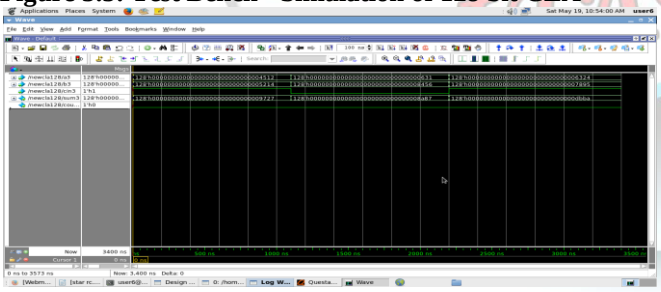
3.3 Simulation of Carry Lookahead Adder of 128 bits

Logical Simulation in Mentor initiated through a set of logical conditions at the 128 inputs of A and B along with Carryin in the 128 bit CLA and these logical conditions verifies the outputs of 128 bit for Sum and a Carryout have to be checked to make sure they are correct. So a set of test benches are created to find the validity of the design.

A simple test bench is shows two different numbers at A and B inputs with Carry-in and find the output, shown in figure 3.5.

A = 4512, B = 5214 and Cin = 1, Output is 9727, Carry out = 0
A = 631, B = 456 and Cin = 0, Output is 1087, Carry out = 0
A = 6324, B = 7895 and Cin = 1, Output is DBBA, Carry out = 0

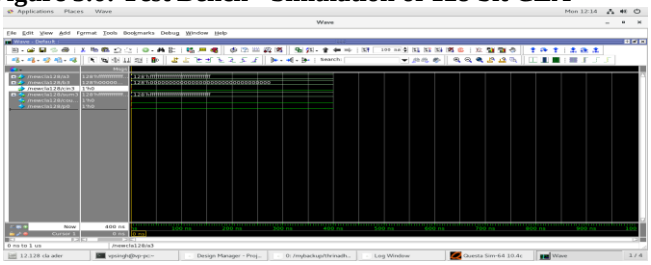
Figure 3.5: Test Bench - Simulation of 128 bit CLA



Another test bench shows two different numbers at A and B inputs with Carry-in and find the output. One has all ONEs and other has all ZEROs, shown in figure 3.6.

A = FF..FF, B = 00..00 and Cin = 0, Output is FF..FF, Carry out = 0.

Figure 3.6: Test Bench - Simulation of 128 bit CLA



III. RESUTLS

Delays are calculated based on the figure 2.1, logic design for RCA along with figure 3.2 and 3.3, logic design for CLA of 128 bits and simulation has been done in Mentor EDA tools. The simulation results of these two adders are compared in terms of delay and area and shown in table 4.1. Gate count has been increased in CLA because at each adder stage Generate and Propagate logic has to be built which is less than 2 percent. But the delay got reduced by over 40 percent from 22 ns to 12.9 ns in this 64 bit design.

Table 4.1: Area and Delay comparison of RCA and CLA

Name of the Adder Design	Logic Gate Count	Delay
128 bits		
Ripple Carry Adder	11,274	32.5 ns
Carry Look Ahead Adder	13,414	19.3 ns
64 bits		
Ripple Carry Adder	5,461	22.5 ns
Carry Look Ahead Adder	5,574	12.9 ns

IV. CONCLUSION

CLA is 40 percent faster than RCA and it will boast the performance of any microprocessor though its higher gate count. This has been done with 135 nm technology in Mentor tool. At 19 ns, the One GHz microprocessor has to wait for 20 cycles to finish a 128 bit addition. One can do the design in 90 nm and 65 nm to reduced the delay as microprocessors are going over 50 GHz speed and a clock cycle of 20 ps.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 8, August 2019

REFERENCES

1. Morris Mano "Digital Design". Pearson Education Asia. 3rd Ed, 2002.
2. Raj Kumar Mistri, Rahul Ranjan, Anuradha Kumari Choudhari, and Babita Kumari, "IC Layout Design of 4-bit Magnitude Comparator using Electric VLSI Design System", IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), Volume 7, Issue 2, Ver. I, PP 67-73, e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197, www.iosrjournals.org
3. Shun-Wen Cheng, "A high-speed magnitude comparator with small transistor count" in proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 1168- 1171, 2003.
4. Suman Deb and S. Chaudhary, "High-Speed Comparator Architectures for Fast Binary Comparison", IEEE International Conference on Emerging Applications of Information Technology, pp.454-457, 2012.
5. Kishore Prabhala, Haritha Dasari, and Thrinadh Komatipalli, "Performance Comparison of 64-Bit Adders", IJEDR, Volume 6, Issue 2, June 2018, ISSN: 2321 9939
6. Anjuli and Satyajit Anand, "2-Bit Magnitude Comparator design using different logic styles," International Journal of Engineering Science Invention, Vol. 2, No. 1, pp.13-24, 2013.
7. Kishore Prabhala and Prof. Prabhandhakam Sangameswara Raju, "A CMOS Design of 64 bit ALU using Mentor Tools", IJSART - Volume 5 Issue 4 –April 2019, ISSN [ONLINE]: 2395-1052
8 C.H. Huang and J.S. Wang, "High-performance and power-efficient CMOS comparators," IEEE J. Solid-State Circuits, vol. 38, no.2, pp. 254–262, Feb. 2003.

BIOGRAPHY

Kishore Prabhala is a research Scholar in EEE PhD, Rayalaseema University and also Senior Member IEEE. He published eight papers in CMOS VSLI design in India after leaving USA in 1994 working at Motorola, MMI and National Semiconductor from 1981. Currently, he is the Principal, PLNM Degree College, Opposite Acharya Nagarjuna University Mens Hostel, Nagarjuna Nagar – 522 510, Guntur Dist., AP, India. He received a MSEE from Georgia Institute of Technology, GA, USA in 1989 and BSEE from Purdue University, W.Lafatette, IN, USA in 1981.

Prof. Prabhandhakam Sangameswara Raju is a Professor in Dept. of Electrical and Electronics Engineering, SVU Engineering College, Sri Venkateswara Univeristy, Tirupati – 517 502, Chitoor Dist., AP. He received M.Tech. and Ph.D. from SVU Engineering College. He has been teaching PG course for last 25 years and guided ove 52 projects. He published over 70 papers. Currently there are 8 students pursuing Ph.D.