Computer Organization and Architecture Laboratory

Assignment 3

Group 22

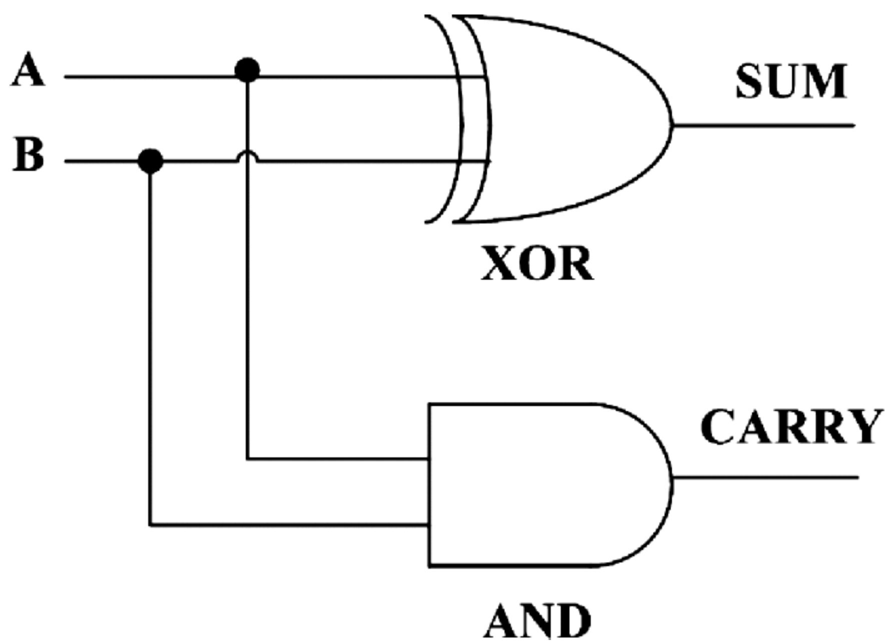
Nikhil Saraswat (20CS10039)

Amit Kumar (20CS30003)

PART 1: RCA

Half adder

Half adder is the simplest of all adder circuits. Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (s) and carry bit (c) both as output. The addition of 2 bits is done using a combination circuit called a Half adder. The input variables are augend and addend bits and output variables are sum & carry bits. A and B are the two input bits.



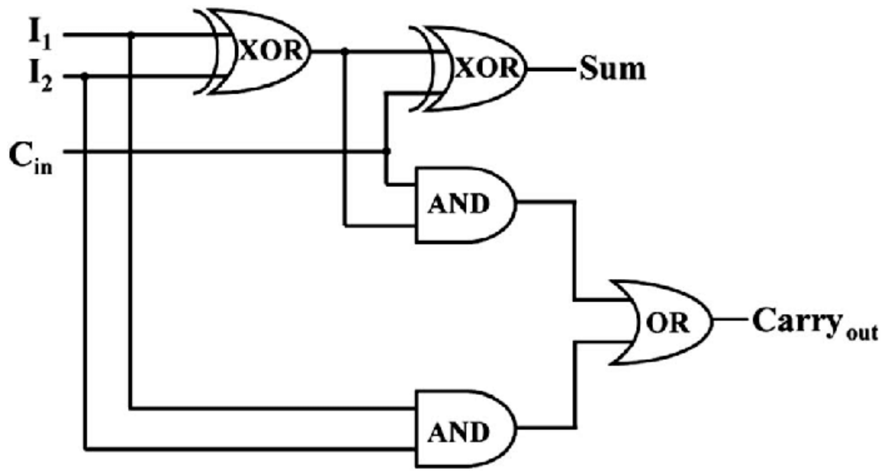
Inputs		Outputs	
A	B	Sum	C_out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean Logic:

```
c_out = a & b
sum = a ^ b
```

Full adder

A full adder circuit is central to most digital circuits that perform addition or subtraction. It is so called because it adds together two binary digits, plus a carry-in digit to produce a sum and carry-out digit. It therefore has three inputs and two outputs.



Inputs			Outputs	
I_1	I_2	C_in	Sum	C_out
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

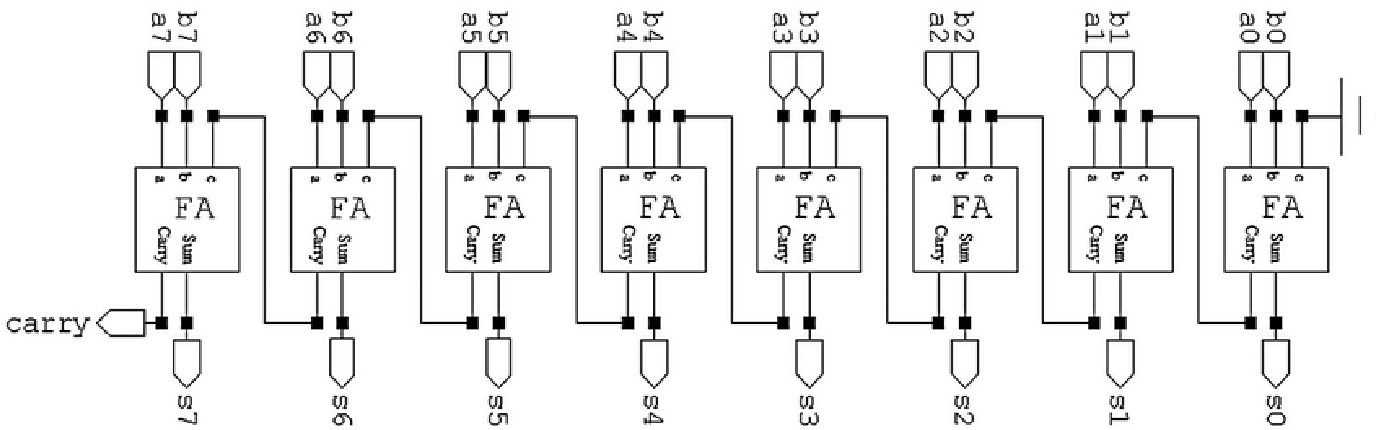
Boolean Logic:

```
sum = a ^ b ^ c_in
c_out = (a & b) | (c & (a ^ b))
```

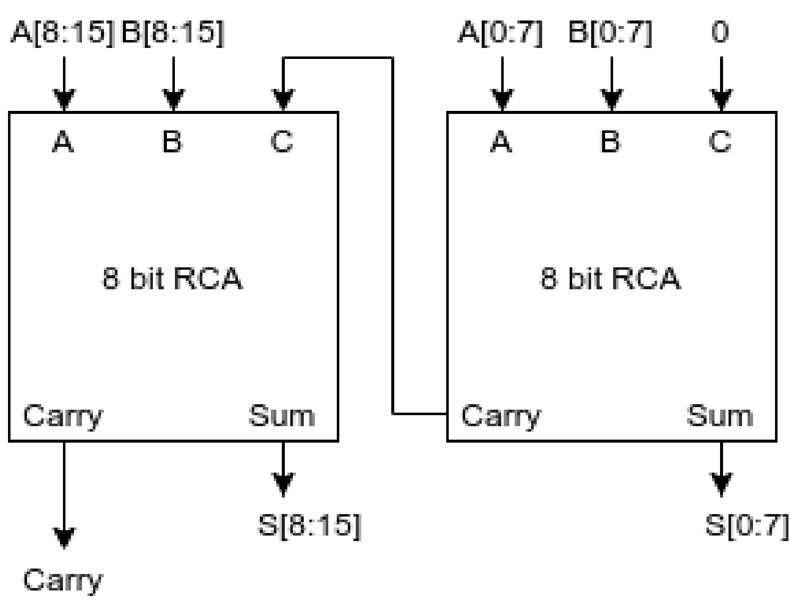
RCA

A structure of multiple full adders is cascaded in a manner to gives the results of the addition of an n bit binary sequence. This adder includes cascaded full adders in its structure so, the carry will be generated at every full adder stage in a ripple-carry adder circuit. These carry output at each full adder stage is forwarded to its next full adder and there applied as a carry input to it. This process continues up to its last full adder stage. So, each carry output bit is rippled to the next stage of a full adder. By this reason, it is named as “RIPPLE CARRY ADDER”. The most important feature of it is to add the input bit sequences whether the sequence is 4 bit or 5 bit or any.

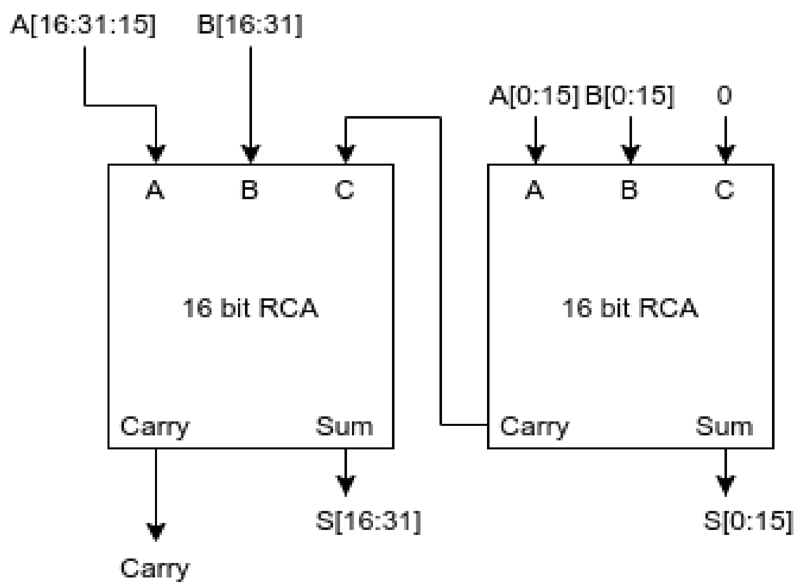
8-bit RCA



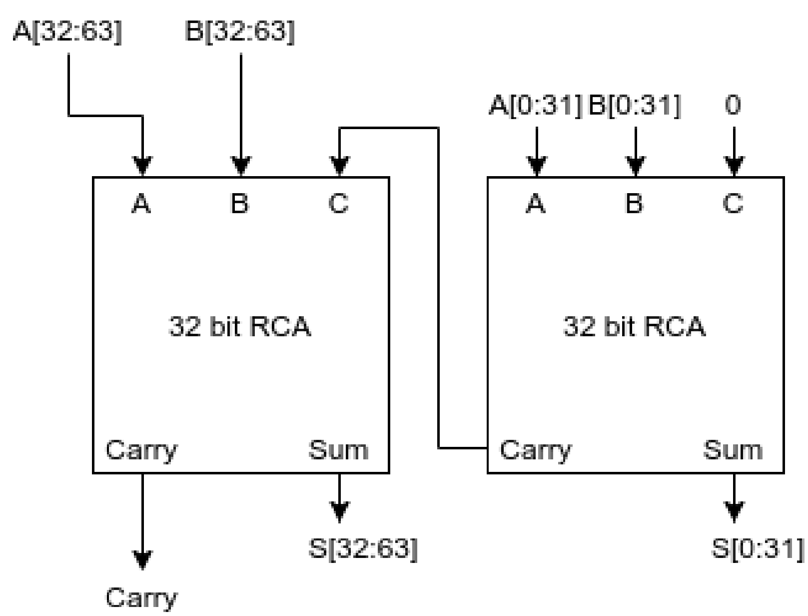
16-bit adder



32-bit adder



64-bit adder

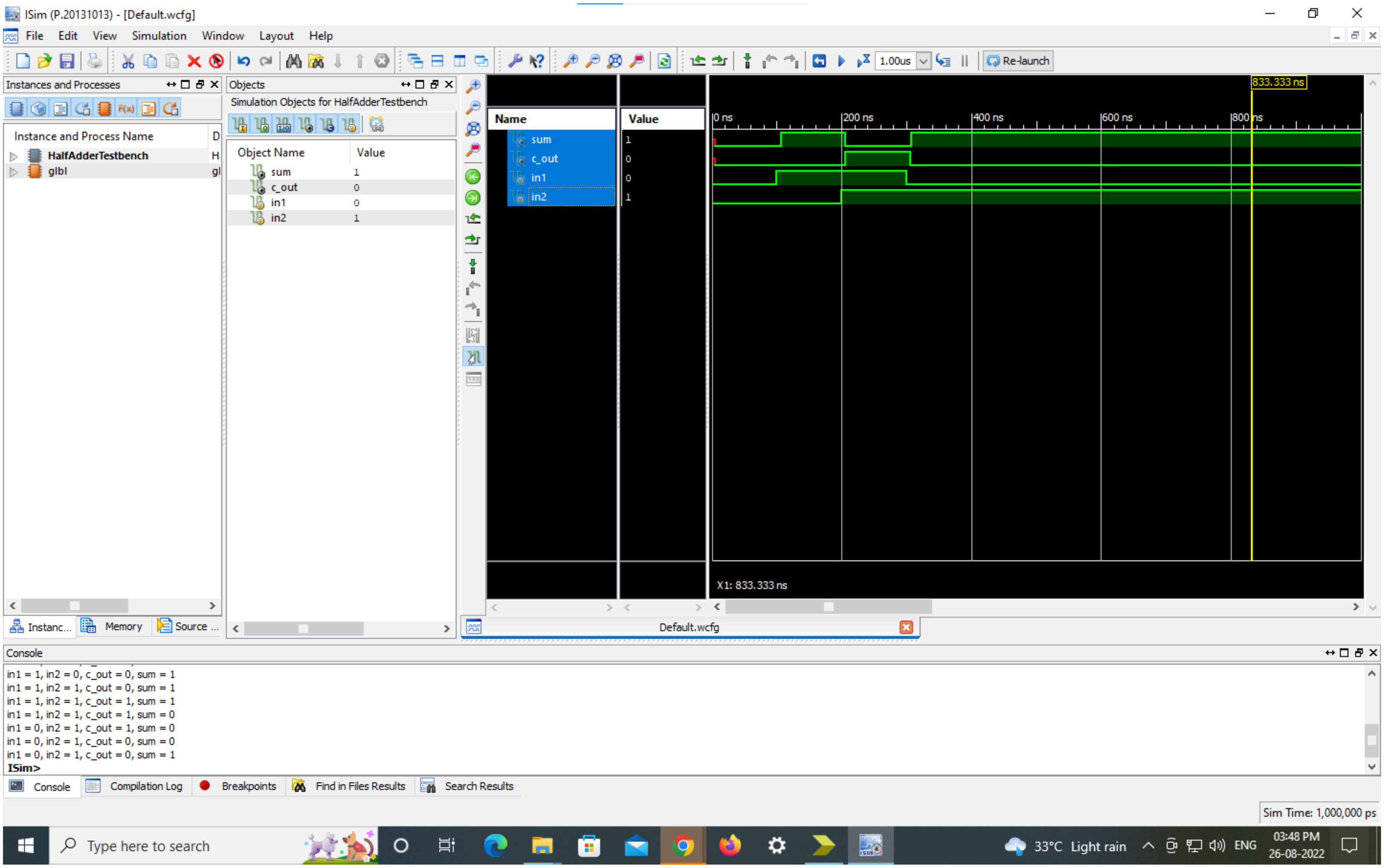


SYNTHESIS SUMMARY

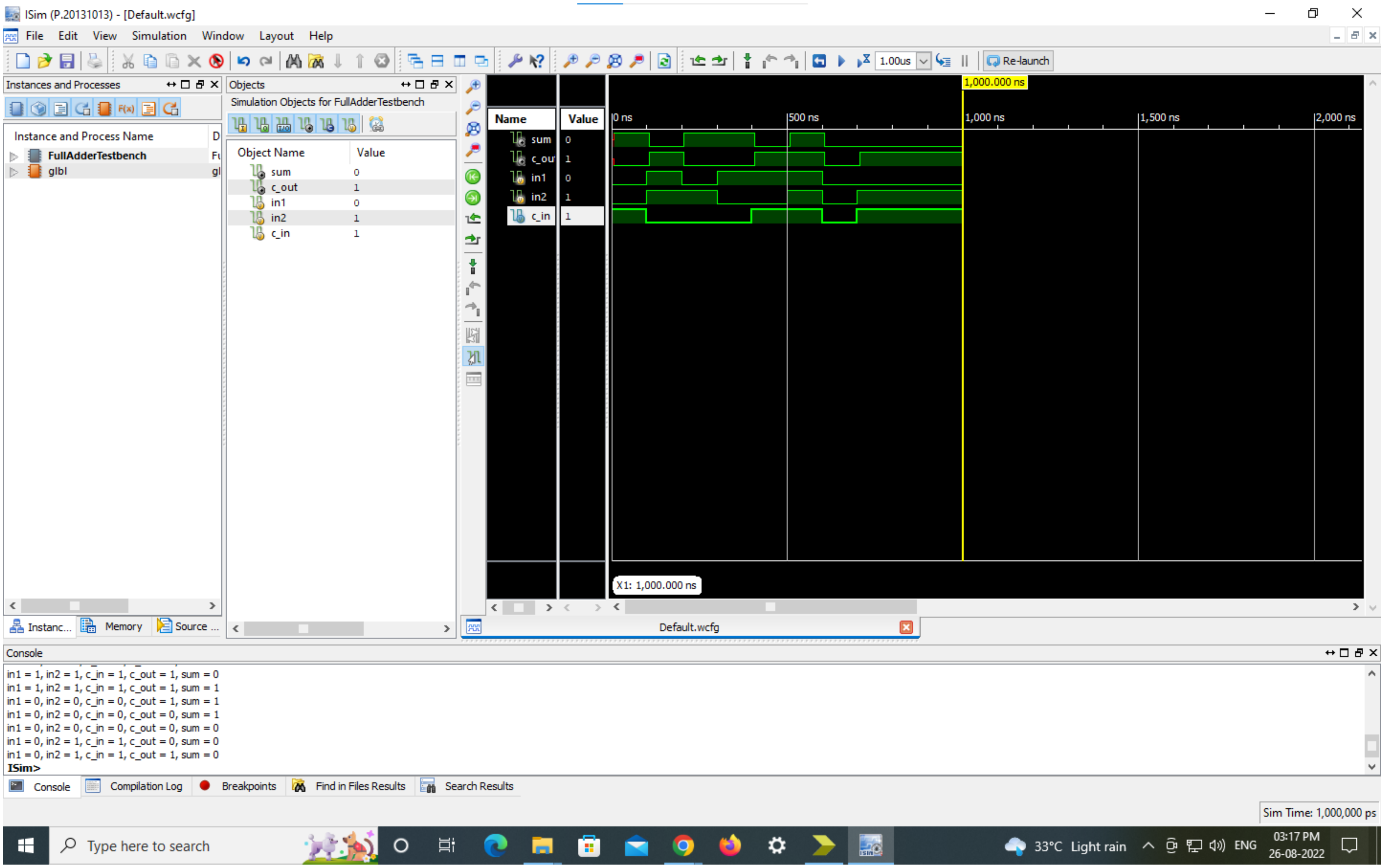
Circuit	Delay (in ns)	Logic Levels	Number of Slice LUTs	Number of bonded IOBs
8-bit RCA	3.471	6	12 / 63400	26 / 210
16-bit RCA	6.167	10	24 / 63400	50 / 210
32-bit RCA	11.559	18	48 / 63400	98 / 210
64-bit RCA	22.343	34	96 / 63400	194 / 210

Post Route Simulation Screen Shots:

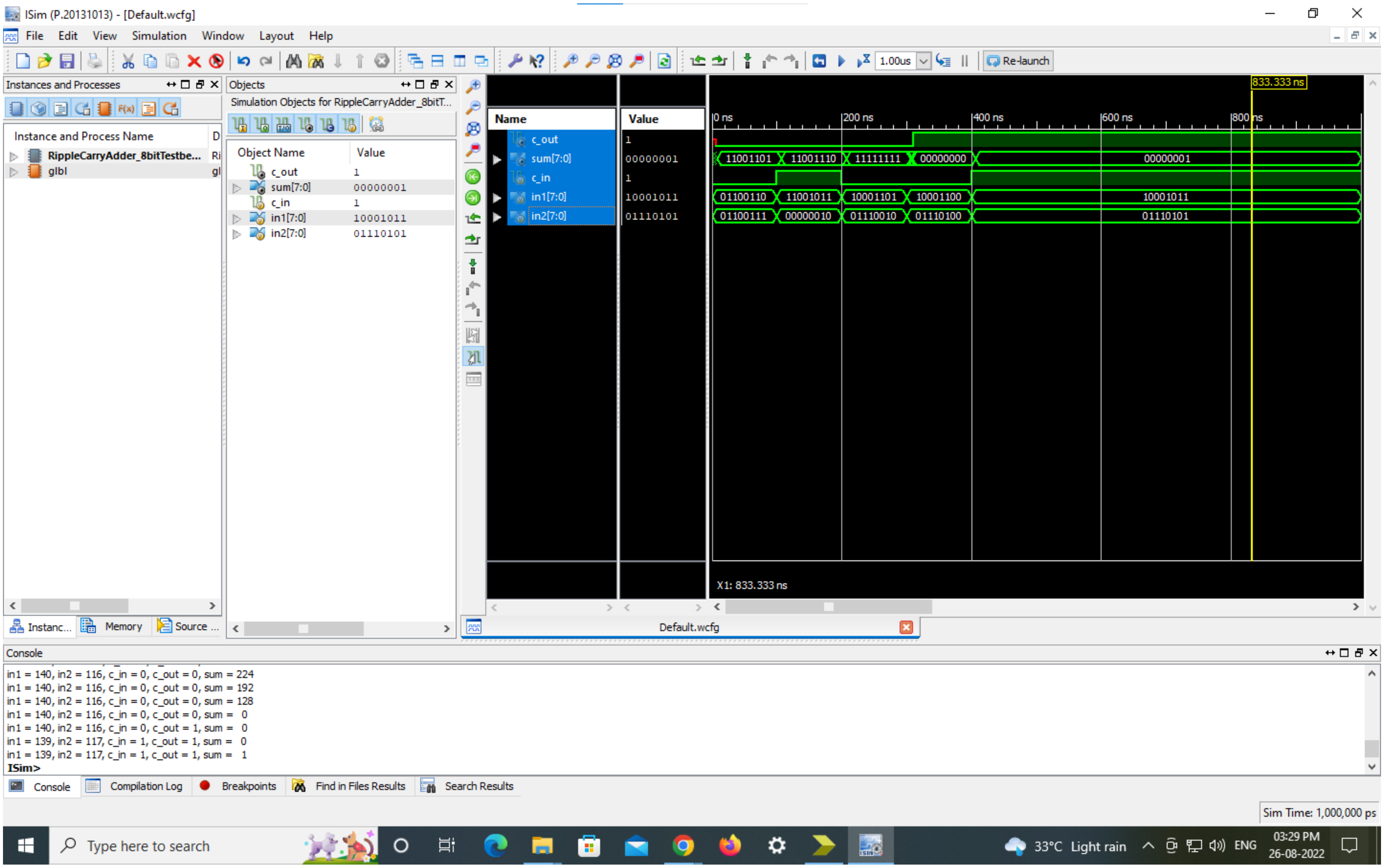
Half adder



Full adder

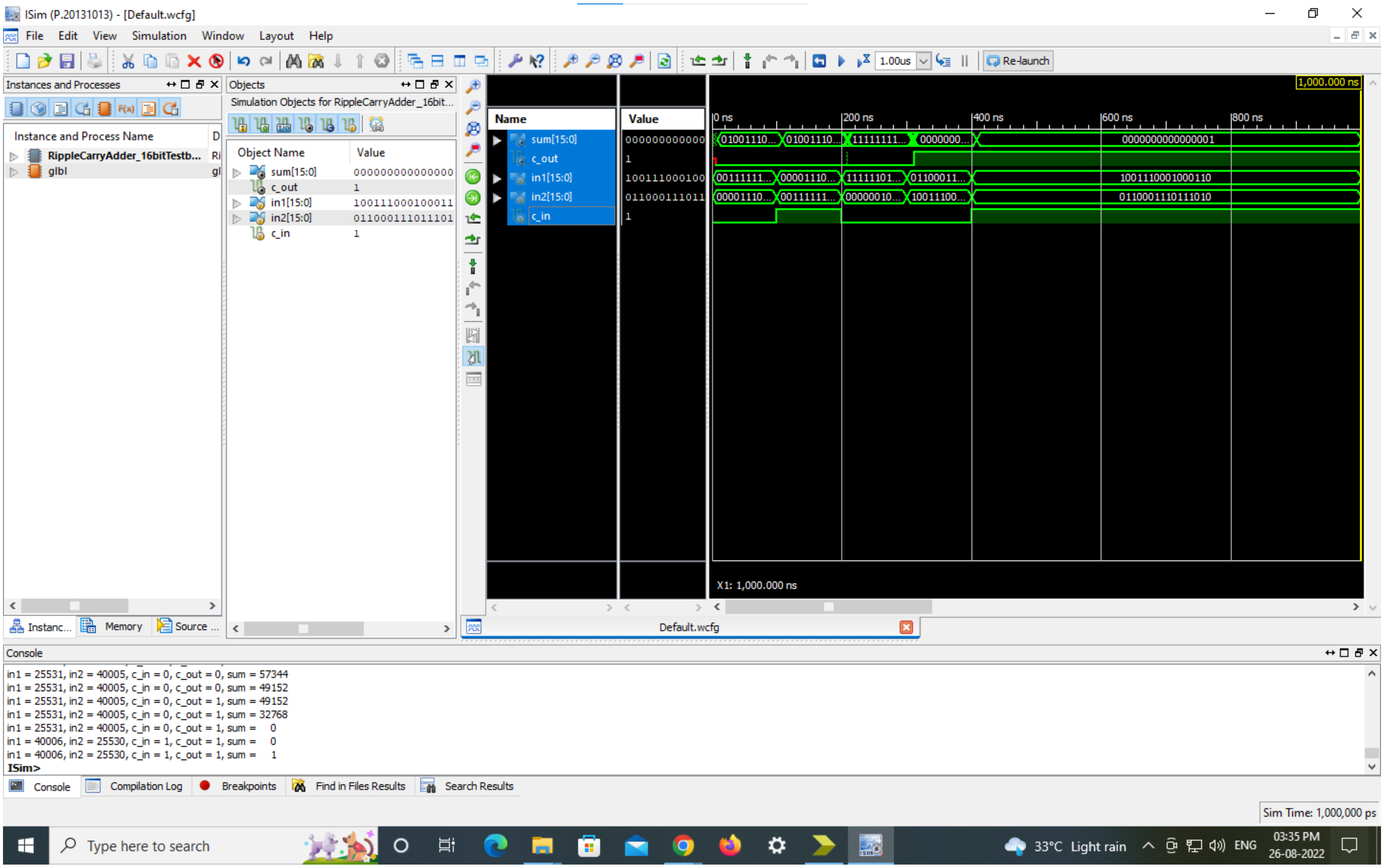

ISim (P.20131013) - [Default.wcfg]
Instances and Processes
Objects
Simulation Objects for FullAdderTestbench
Object Name
Value
sum 0
c_out 1
in1 0
in2 1
c_in 1
1,000.000 ns
X1: 1,000.000 ns
Default.wcfg
Console
in1 = 1, in2 = 1, c_in = 1, c_out = 1, sum = 0
in1 = 1, in2 = 1, c_in = 1, c_out = 1, sum = 1
in1 = 0, in2 = 0, c_in = 0, c_out = 1, sum = 1
in1 = 0, in2 = 0, c_in = 0, c_out = 0, sum = 1
in1 = 0, in2 = 0, c_in = 0, c_out = 0, sum = 0
in1 = 0, in2 = 1, c_in = 1, c_out = 0, sum = 0
in1 = 0, in2 = 1, c_in = 1, c_out = 1, sum = 0
ISim>
Sim Time: 1,000,000 ps

8 – bit RCA

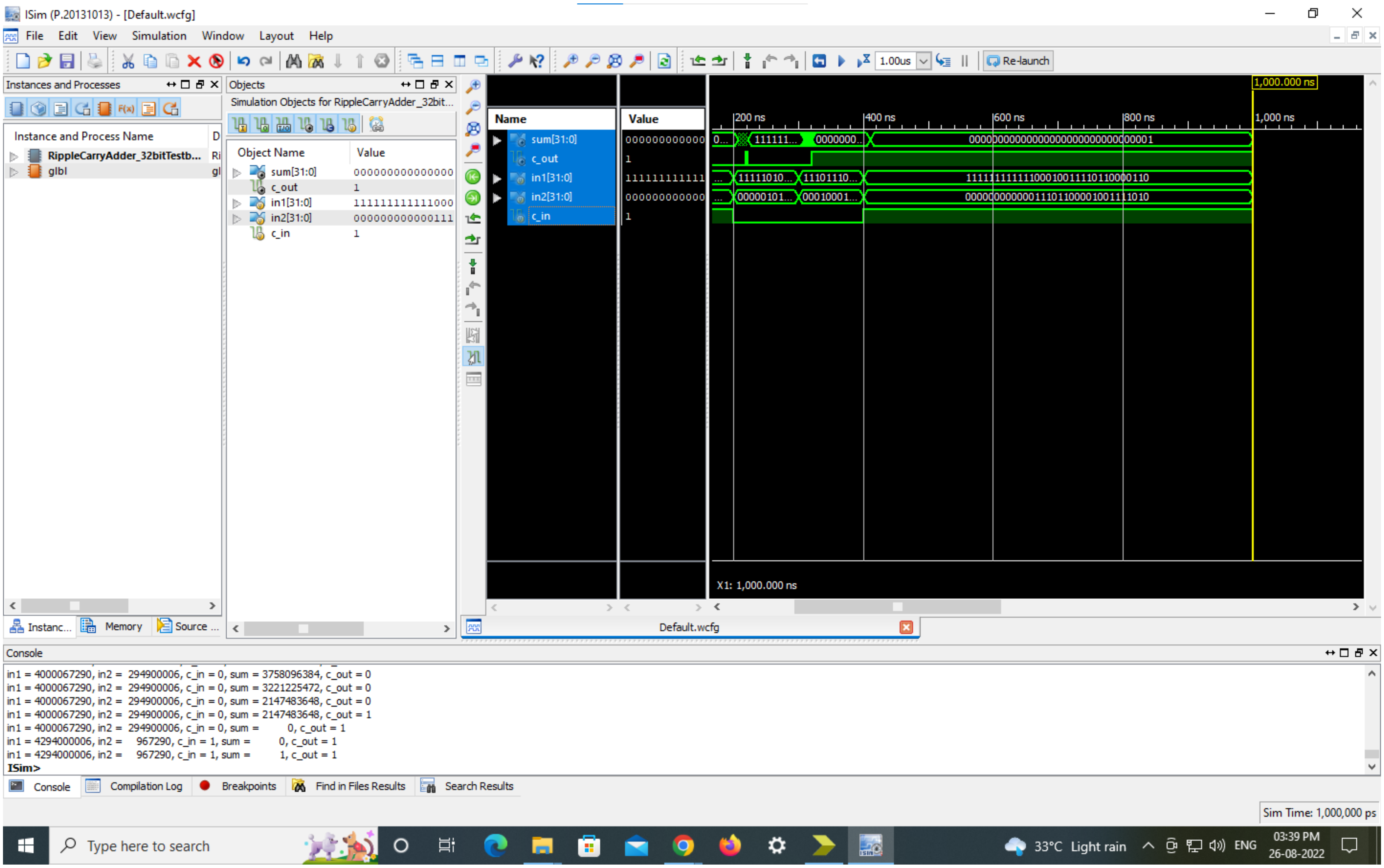


in1 = 140, in2 = 116, c_in = 0, c_out = 0, sum = 224
in1 = 140, in2 = 116, c_in = 0, c_out = 0, sum = 192
in1 = 140, in2 = 116, c_in = 0, c_out = 0, sum = 128
in1 = 140, in2 = 116, c_in = 0, c_out = 0, sum = 0
in1 = 140, in2 = 116, c_in = 0, c_out = 1, sum = 0
in1 = 139, in2 = 117, c_in = 1, c_out = 1, sum = 0
in1 = 139, in2 = 117, c_in = 1, c_out = 1, sum = 1

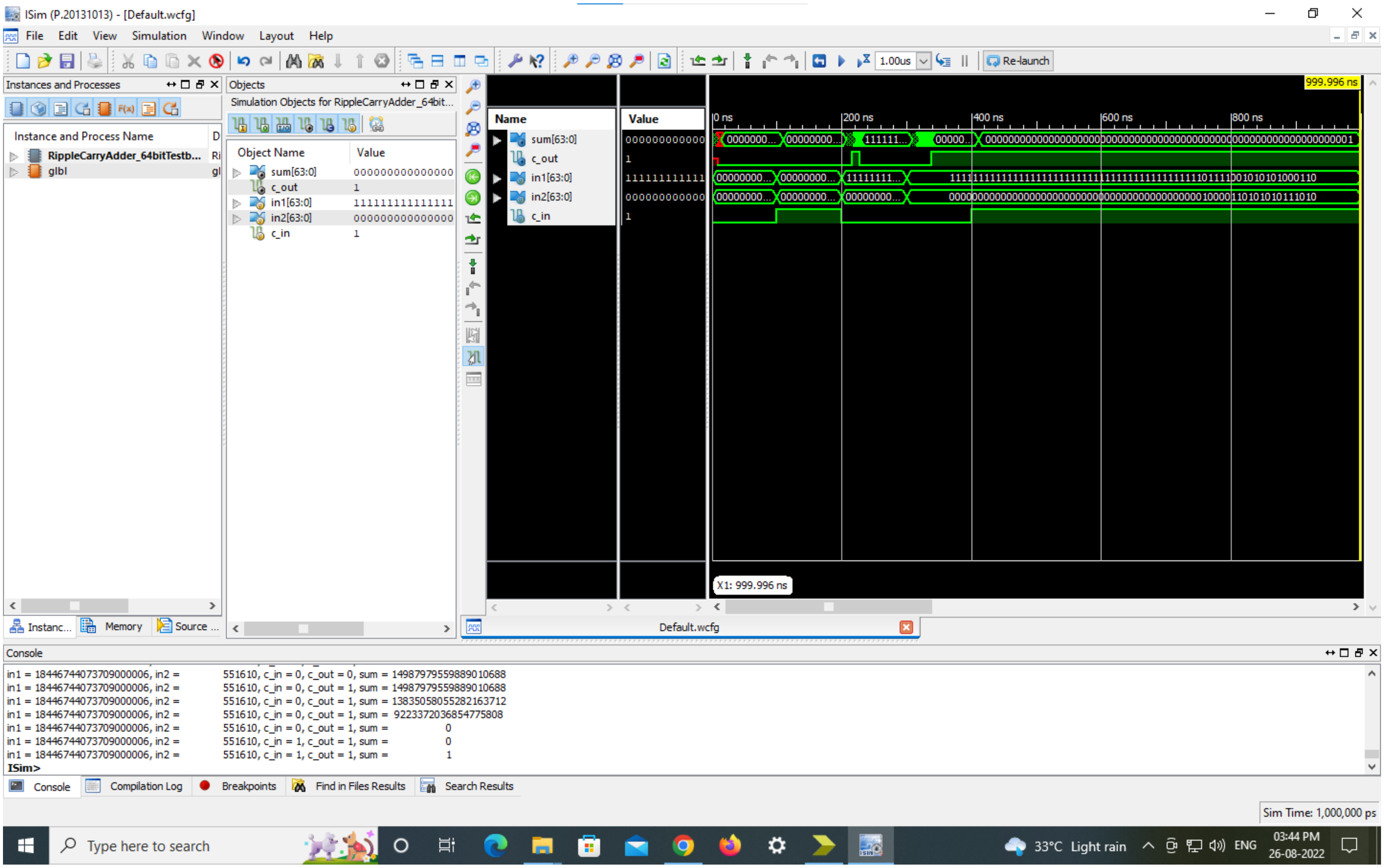
16 – bit RCA


ISim (P.20131013) - [Default.wcfg]
in1 = 25531, in2 = 40005, c_in = 0, c_out = 0, sum = 57344
in1 = 25531, in2 = 40005, c_in = 0, c_out = 0, sum = 49152
in1 = 25531, in2 = 40005, c_in = 0, c_out = 1, sum = 49152
in1 = 25531, in2 = 40005, c_in = 0, c_out = 1, sum = 32768
in1 = 25531, in2 = 40005, c_in = 0, c_out = 1, sum = 0
in1 = 40006, in2 = 25530, c_in = 1, c_out = 1, sum = 0
in1 = 40006, in2 = 25530, c_in = 1, c_out = 1, sum = 1
ISim>
Sim Time: 1,000,000 ps

32 – bit RCA


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Instances and Processes
Objects
Simulation Objects for RippleCarryAdder_32bit...
Instance and Process Name
RippleCarryAdder_32bitTestb...
glbl
Object Name
Value
sum[31:0]
c_out
in1[31:0]
in2[31:0]
c_in
Name
Value
1,000.000 ns
X1: 1,000.000 ns
Default.wcfg
Console
in1 = 4000067290, in2 = 294900006, c_in = 0, sum = 3758096384, c_out = 0
in1 = 4000067290, in2 = 294900006, c_in = 0, sum = 3221225472, c_out = 0
in1 = 4000067290, in2 = 294900006, c_in = 0, sum = 2147483648, c_out = 0
in1 = 4000067290, in2 = 294900006, c_in = 0, sum = 2147483648, c_out = 1
in1 = 4000067290, in2 = 294900006, c_in = 0, sum = 0, c_out = 1
in1 = 4294000006, in2 = 967290, c_in = 1, sum = 0, c_out = 1
in1 = 4294000006, in2 = 967290, c_in = 1, sum = 1, c_out = 1
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps

64 – bit RCA


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Instances and Processes
Objects
Simulation Objects for RippleCarryAdder_64bit...
Object Name
Value
sum[63:0]
c_out
in1[63:0]
in2[63:0]
c_in
Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
999.996 ns
X1: 999.996 ns
Default.wcfg
Console
in1 = 18446744073709000006, in2 = 551610, c_in = 0, c_out = 0, sum = 14987979559889010688
in1 = 18446744073709000006, in2 = 551610, c_in = 0, c_out = 1, sum = 14987979559889010688
in1 = 18446744073709000006, in2 = 551610, c_in = 0, c_out = 1, sum = 13835058055282163712
in1 = 18446744073709000006, in2 = 551610, c_in = 0, c_out = 1, sum = 9223372036854775808
in1 = 18446744073709000006, in2 = 551610, c_in = 0, c_out = 1, sum = 0
in1 = 18446744073709000006, in2 = 551610, c_in = 1, c_out = 1, sum = 0
in1 = 18446744073709000006, in2 = 551610, c_in = 1, c_out = 1, sum = 1
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps

Q. How can you use the above circuit, to compute the difference between two n-bit numbers?

Ans: (a - b) can be written as (a + (-b)).We observe that (-b) is the 2's complement of b. So, (-b) can be written as (~b + 1).Now, ~b is 1's complement of b. so (a - b) is RCA(a, ~b, 1) where 1 is the carry in and a and ~b arethe two inputs.

In the circuit, we can connect a switch to the carry in of the adder, as well as XOR gates to each of b's input bits. When the switch is turned on, the carry in bit becomes 1 and all the bits of b are simultaneously flipped (XOR with 1 flips the bit value). In this manner, we shall obtain a-b as output. When the switch is turned off, nothing changes and we obtain a+b in output.

Post Route of Difference between two 64 bit numbers