COA Lab Verilog Assignment 1
Group 17

Aditya Choudhary (20CS10005)
Rishi Raj (20CS30040)

Problems :-

1. A Half Adder is a combinational circuit, which takes in two input bits, a and b, and produces the sum bit, s and the carry-out bit, c. Write the truth-table for the assignments of s and c. Write the verilog code for half adder and provide snapshots of working.
2. A Full Adder is a combinational circuit, which takes in three input bits, a, b, and in addition a carry-in bit c 0 , and produces the sum bit, s and the carry-out bit, c. Write the truth-table for the assignments of s and c in the Full-Adder. Write the verilog code for the Full Adder and provide snapshots of working.
3. Cascade 8 Full adders and create an 8-bit adder. These type of adders are called Ripple Carry Adders. Likewise create 16, 32, and 64 bit adders, and observe the longest delays in the circuits.
4. How can you use the above circuit, to compute the difference between two n-bit numbers?

Solutions :-

1. **Half Adder solution:**
   Let the input bits to the half adder be a and b. Let sum s be the sum bit and carry c be the carry bit.

Truth Table of Half-Adder			
a	*b*	*s*	*c*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From this table we get :-

S = a XOR b

C = a AND b

Half adder is a fundamental module as it contains no other module
Verliog file name :- “halfadder.v”
Testbench file name :- “halfadder_tb.v”

2. **Full Adder Solution:**
   Let a and b be the input bits, along with the carry bit c. Let s be the sum bit and c be the carry bit.

Truth Table of Full-Adder				
a	*b*	*ci*	*s*	*c*
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

   From this table, we get

   S = a^b^c

   C = ab + ci.(a^b)

   We have created full adder using 2 half adders, hence we have included "halfadder.v" in the verilog file.
   Verilog file name :- "fulladder.v"
   Testbench file name :- "fulladder_tb.v"

3. **Ripple carry Adders**
   a. 8-bit ripple carry adder :-
      We have created the 8 bit ripple carry adder by using 8 full adders. The carry out of 1 full adder is connected to the carry in of the next full adder and so on. The first full adder has 0 as the carry in.
      Verilog file name :- "rippleadder_8bit.v"
      Testbench file name :- "rippleadder_8bit_tb.v"

   b. 16-bit ripple carry adder :-
      Is created using 2-8 bit ripple carry adders. The carry out of one ripple carry adder is connected to the carry in of the next ripple carry adder. The first ripple carry adder has input carry as 0.
      Verilog file name :- "rippleadder_16bit.v"

Testbench file name :- “rippleadder_16bit_tb.v”

c. 32-bit ripple carry adder :-
Is created using 2-16 bit ripple carry adders. The carry out of one ripple carry adder is connected to the carry in of the next ripple carry adder. The first ripple carry adder has input carry as 0.
Verilog file name :- “rippleadder_32bit.v”
Testbench file name :- “rippleadder_32bit_tb.v”

d. 64-bit ripple carry adder :-
Is created using 2-32 bit ripple carry adders. The carry out of one ripple carry adder is connected to the carry in of the next ripple carry adder. The first ripple carry adder has input carry as 0.
Verilog file name :- “rippleadder_64bit.v”
Testbench file name :- “rippleadder_64bit_tb.v”

4. Subtraction using the given circuit :-

We can use the following relation -

$$a - b = a + (-b)$$

We will calculate the 2s complement of b and add it to a using our adder circuits. The result of the computation will be a 2s complement value as well. If the result is positive, the output will be the same as the value of the answer, but if the answer is negative, the output will be the 2s complement of the answer, and the output carry of the adder will be set.

We can develop the circuit as follows :-
We can keep a control bit as input of our adder, if the control bit is set, then we perform subtraction on the numbers, and if it is 0 then we perform addition on the numbers. We assume that the input to the adder are positive signed numbers.

Now, if the control bit is set, we want to flip all the bits of B and add 1 to it (2s complement), hence we can XOR all the bits of B with the control bit and send this control bit to the initial carry bit of the adder.

$$B[i] = B[i] \text{ XOR control}$$

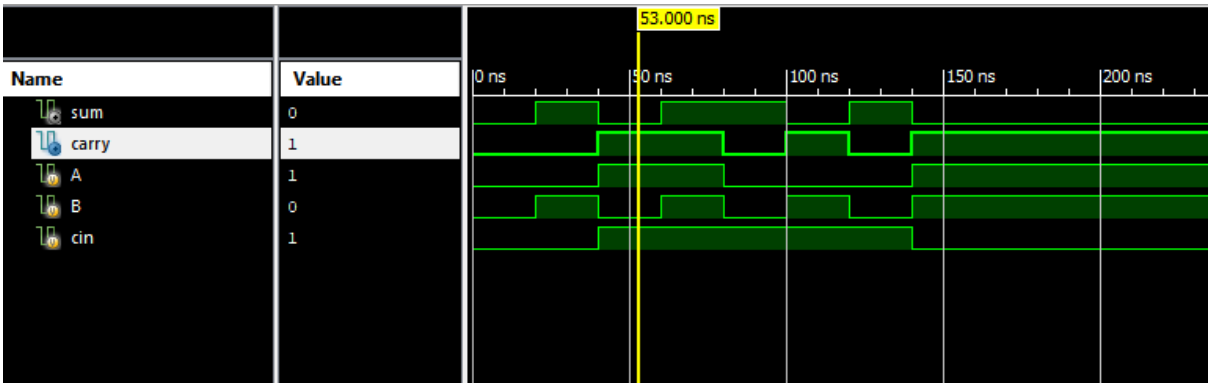
If the control bit is set, all the bits of B are flipped and the control bit, which is set is sent to the initial carry of the adder, which eventually adds 1 to B, hence getting us its 2s complement.

Here is an 8 bit verilog implementation of a conditional adder.
Verilog file name :- “add_sub.v”
Testbench file name :- “add_sub_tb.v”

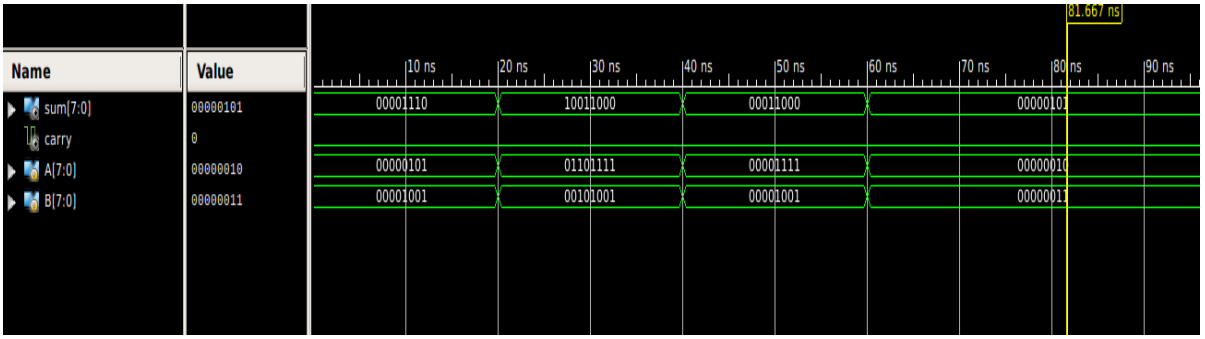
SNAPSHOTS OF SIMULATION :-

Half Adder :-

Full Adder :-

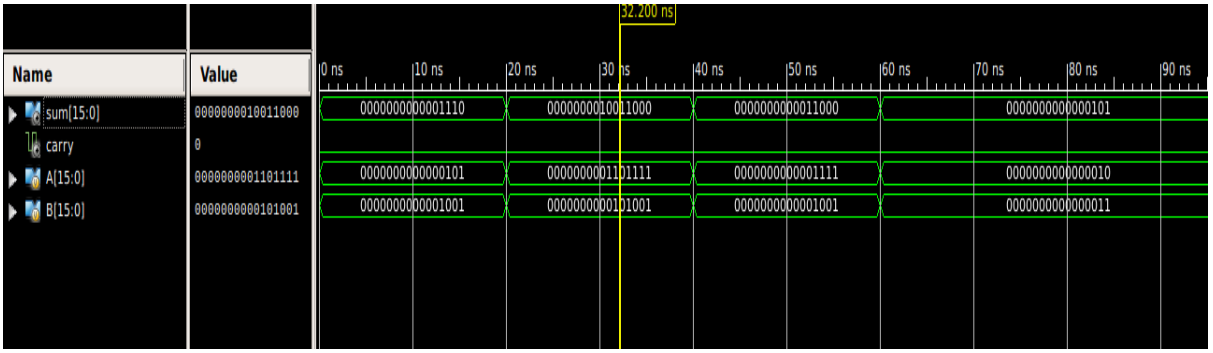


8 bit RCA :-



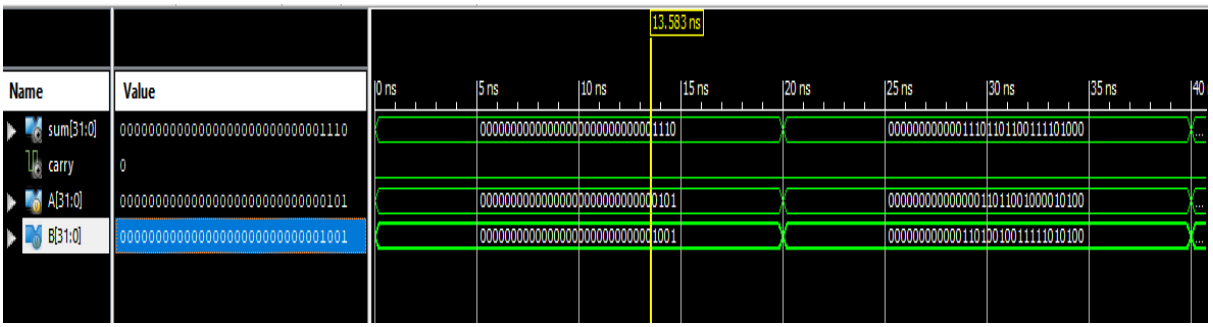
Critical Path Delay :- 3.471 ns

16 bit RCA :-



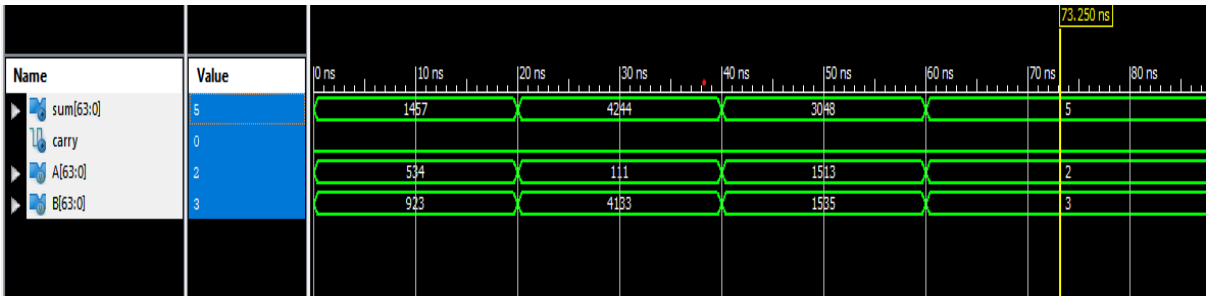
Critical Path Delay :- 6.167 ns

32 bit RCA :-



Critical Path Delay :- 11.559 ns

64 bit RCA :-



Critical Path Delay :- 22.343 ns

Conditional Adder :-