

# EE 210P – Digital Systems and Design Practicum

## Assignment 2

Name : Narmit Kumar

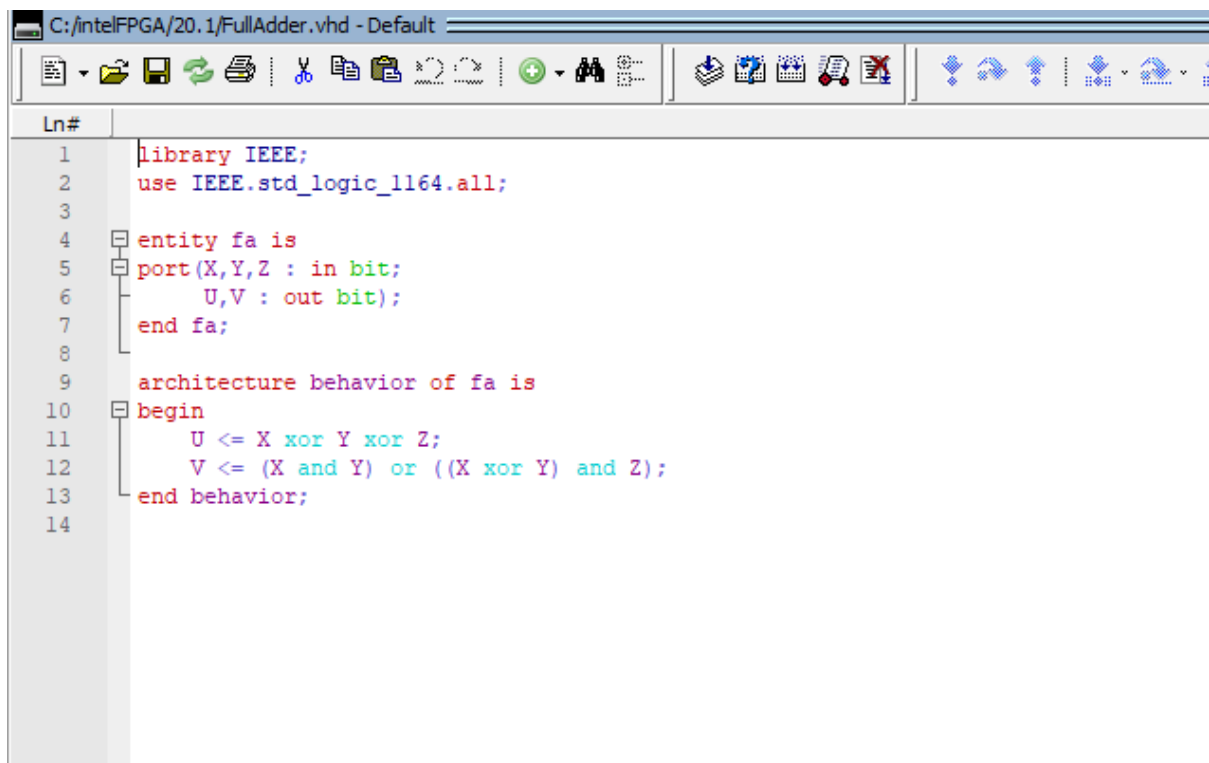
Roll No : B20218

Branch : Electrical Engineering

Q.

Soln. Code for the components :

A. Full adder –



```
C:/intelFPGA/20.1/FullAdder.vhd - Default
[Icons]
Ln#
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity fa is
5  port(X,Y,Z : in bit;
6       U,V : out bit);
7  end fa;
8
9  architecture behavior of fa is
10 begin
11     U <= X xor Y xor Z;
12     V <= (X and Y) or ((X xor Y) and Z);
13 end behavior;
14
```

B. 4 bit full adder.

```
C:/intelFPGA/20.1/4bitadder.vhd - Default
Ln#
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity adder_4bit is
5  port(x,y : in bit_vector(3 downto 0);
6       cin : in bit;
7       sum : out bit_vector(3 downto 0);
8       cout : out bit);
9  end adder_4bit;
10
11  architecture datal of adder_4bit is
12  signal cary : bit_vector(2 downto 0);
13
14
15  component fa is
16  port(X,Y,Z : in bit;
17       U,V : out bit);
18  end component;
19
20  begin
21  a0 : fa port map (x(0),y(0), cin,sum(0),cary(0));
22  a1 : fa port map (x(1),y(1), cary(0),sum(1),cary(1));
23  a2 : fa port map (x(2), y(2), cary(1), sum(2), cary(2));
24  a3 : fa port map (x(3), y(3), cary(2), sum(3), cout);
25
26  end datal;
27
```

# ALU :

Here, ALU has been coded using structural model. The component used is a 4 bit full adder for executing the 4 operations of ALU and rest are simple operations which are executed normally without the use of any component. We have two 4 bit inputs A and B for our ALU, a 3 bit input MS, a single bit input Ci ( carry in ) , 4 bit output S (sum) and single bit output C out (carry out) . 8 combinations of MS are possible and thus this ALU can perform 8 operations.

MS			Operation
0	0	0	$A + B + C_{in}$
0	0	1	$A + B' + 1$
0	1	0	$A + 1$
0	1	1	$A - 1$
1	0	0	A
1	0	1	A AND B
1	1	0	A OR B
1	1	1	A'

## CODE :

```

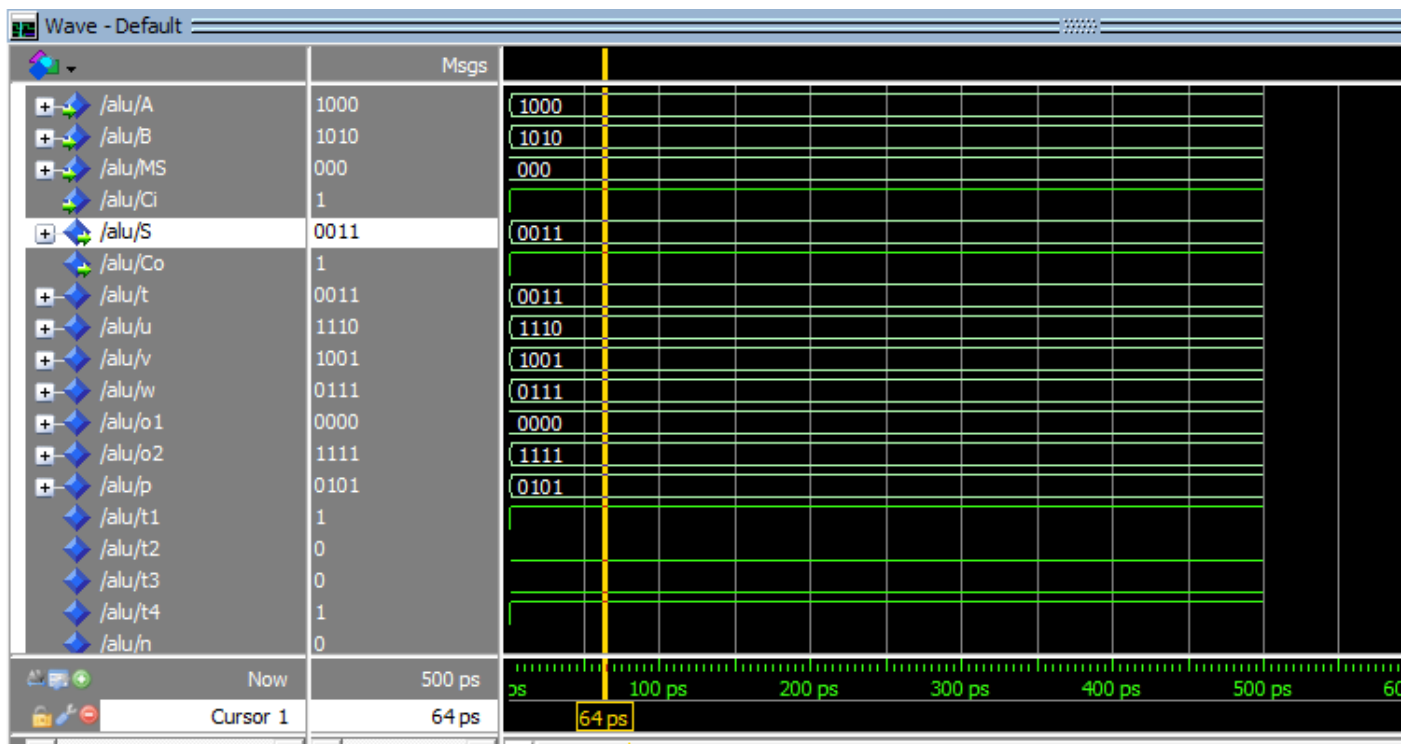
C:/intelFPGA/20.1/ALU.vhd - Default
Ln#
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use ieee.std_logic_arith.all;
5
6  entity ALU is
7  port(A,B: in bit_vector(3 downto 0);
8      MS: in bit_vector(2 downto 0);
9      Ci: in bit;
10     S: out bit_vector(3 downto 0);
11     Co: out bit);
12  end ALU;
13
14  architecture structure of ALU is
15
16  component adder_4bit is
17  port(x,y : in bit_vector(3 downto 0);
18      cin : in bit;
19      sum : out bit_vector(3 downto 0);
20      cout : out bit);
21  end component;
22
23  signal t,u,v,w,o1,o2,p: bit_vector(3 downto 0);
24  signal t1,t2,t3,t4,n,o: bit;
25
26
27  begin
28  p<= NOT B;
29  n<='0';
30  o<= '1';
31  o1<= B"0000";
32  o2<= B"1111";
33  alu1: adder_4bit port map(A,B,Ci,sum=>t,cout=>t1); --Performing operation A + B + Cin
34  alu2: adder_4bit port map(A,p,o,sum=>u,cout=>t2); --Performing operation A + B' + 1
35  alu3: adder_4bit port map(A,o1,o,sum=>v,cout=>t3); --Performing operation A + 1
36  alu4: adder_4bit port map(A,o2,n,sum=>w,cout=>t4); -- Performing operation A - 1 ( adding 2's compliment of 1 to A)
37  process(MS,t,u,v,w,t1,t2,t3,t4,n,o1)
38  begin
39  case MS is
40  when "000" =>
41  S<=t;
42  Co<=t1;
43  when "001" =>
44  S<=u;
45  Co<=t2;
46  when "010"=>
47  S<=v;
48  Co<=t3;
49  when "011"=>
50  S<=w;
51  Co<=t4;
52  when "100" => --Operation A
53  S <= A;
54  Co<=n;
55  when "101" => --Operation A and B
56  S <= A and B;
57  Co<=n;
58  when "110" => --Operation A or B
59  S <= A or B;
60  Co<=n;
61  when "111" => --Operation A'
62  S <= not A;
63  Co<=n;
64  when others =>
65  S <= o1;
66  Co<=n;
67  end case;
68  end process;
69  end structure;
70

```

Simulations :

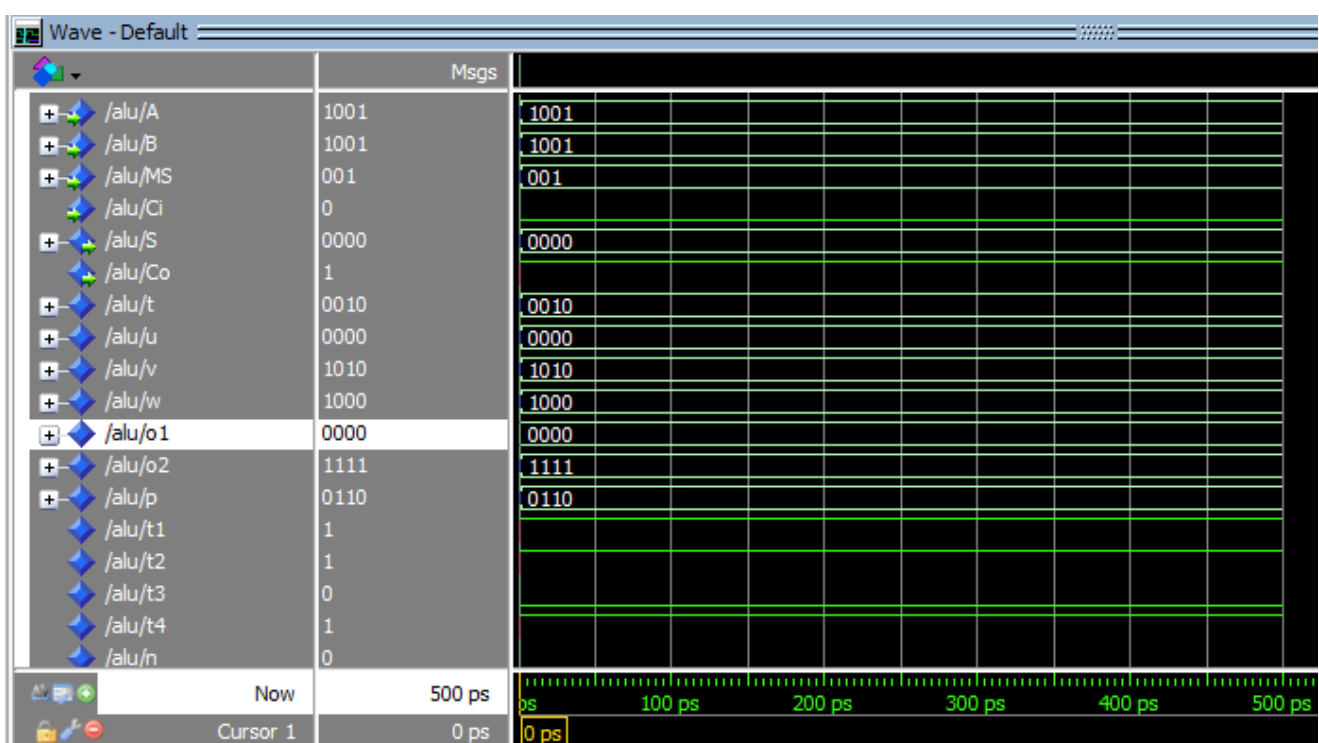
Operation 1 :  $A + B + C_{in}$

Here, we will put  $A = 1000$  ,  $B = 1010$  and  $C_{in} = 1$  which will give  $S = 0011$  and  $Co = 1$ . Hence operation is verified.



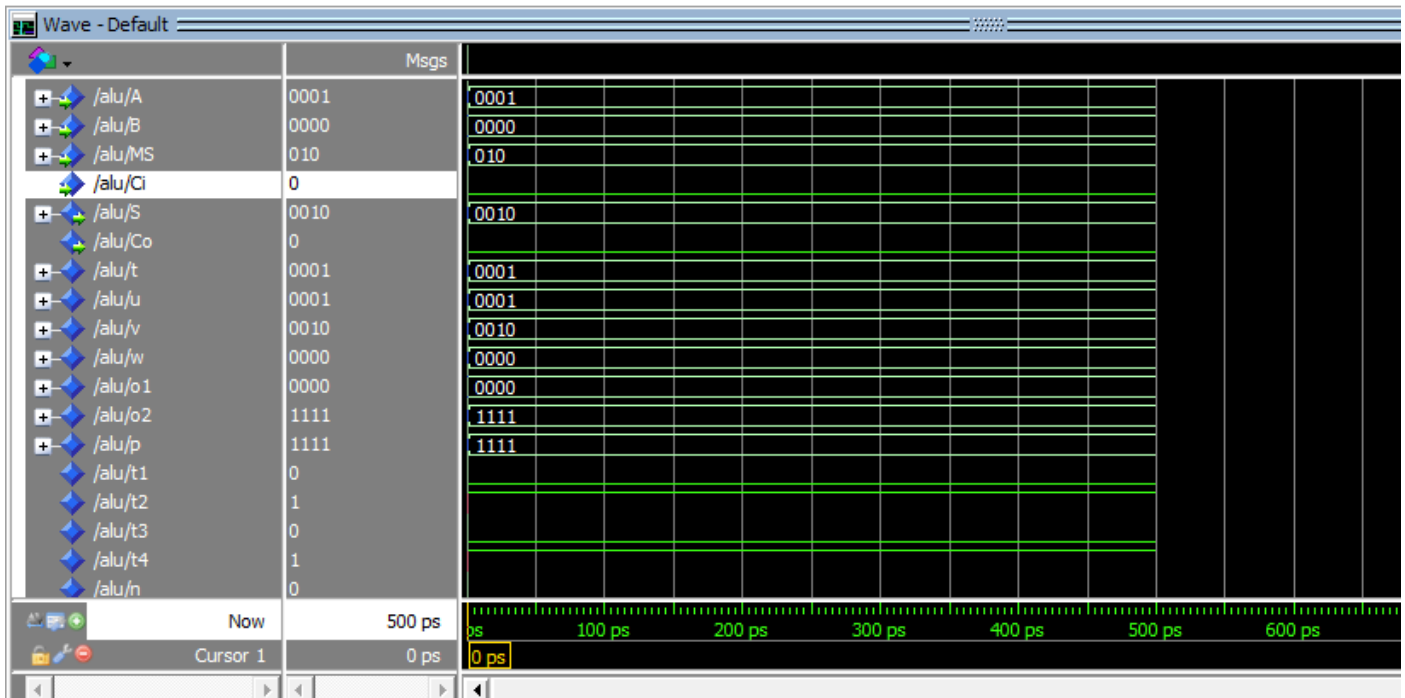
Operation 2 :  $A + B' + 1$

Here, we will put  $A = 1001$  ,  $B = 1001$  which will give  $S = 0000$  and  $Co = 1$ . Hence operation is verified from the below simulation.



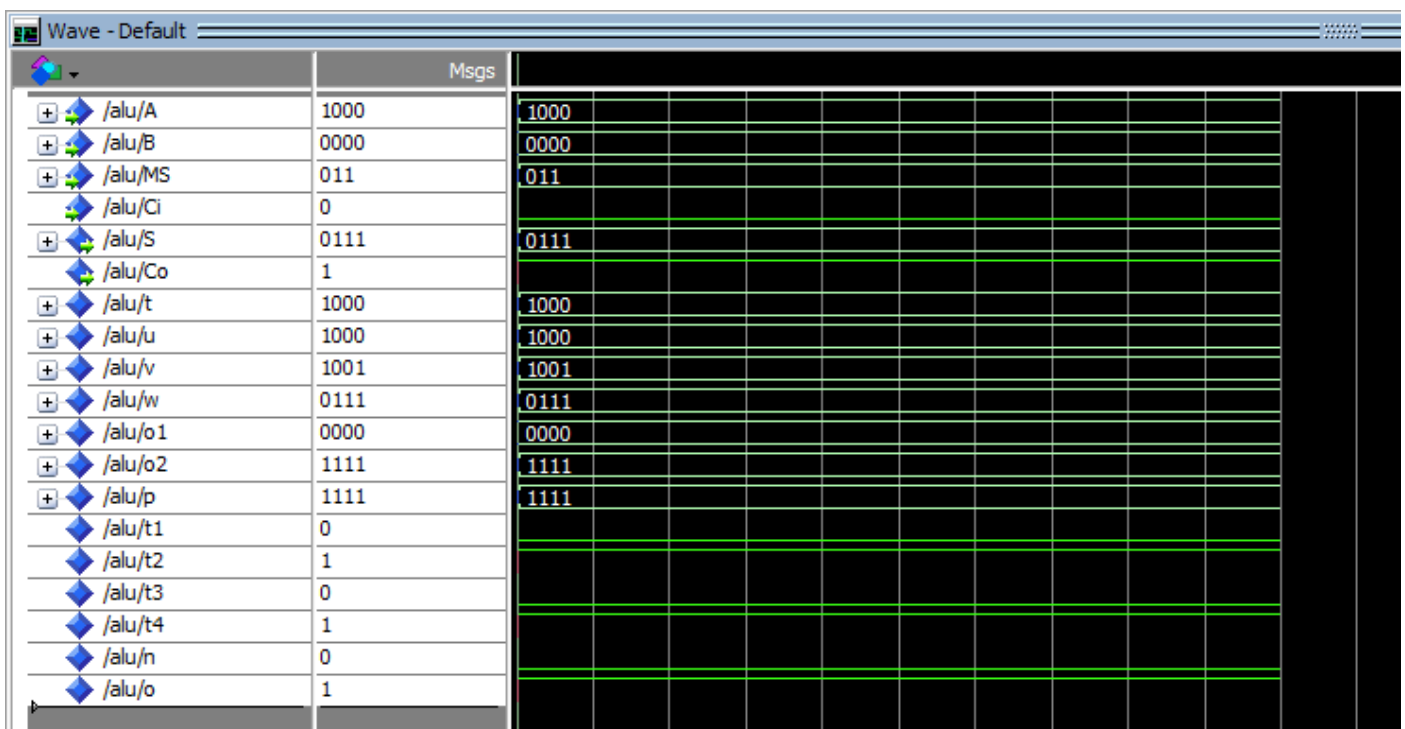
### Operation 3 : A + 1

Here, we will put A = 0001 and output should be S = 0010. Hence, operation is verified from the below observation.



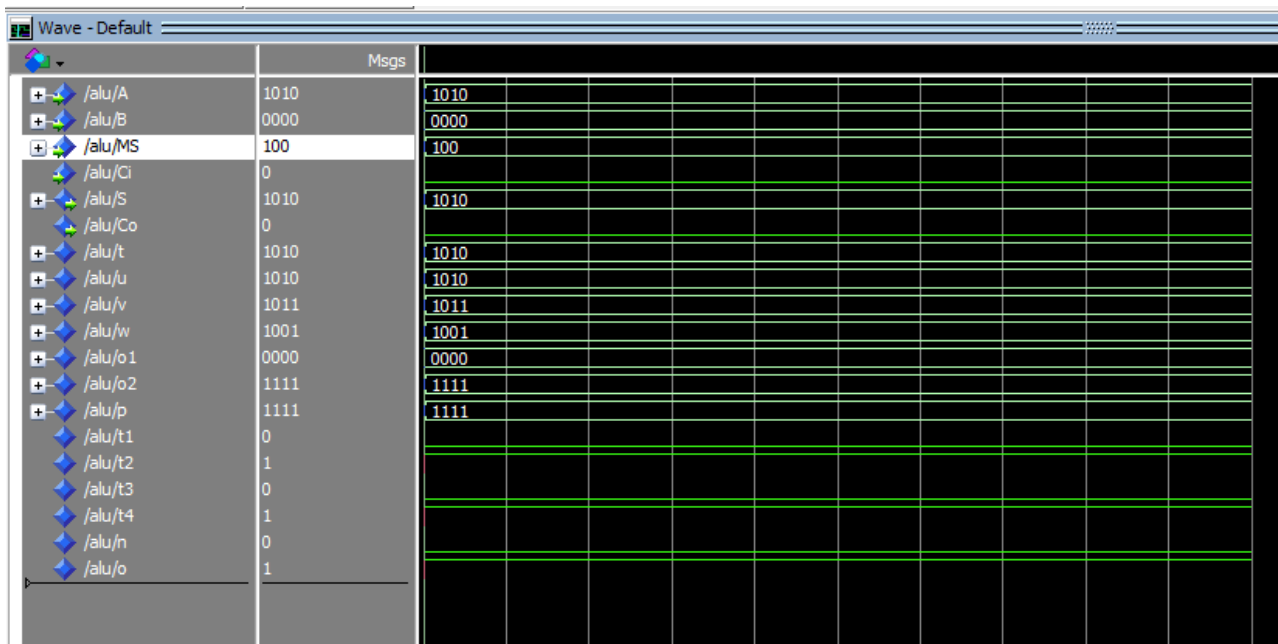
### Operation 4 : A - 1

Here, we will put A = 1000 and output should be S = 0111. Hence, operation is verified from the below observation.



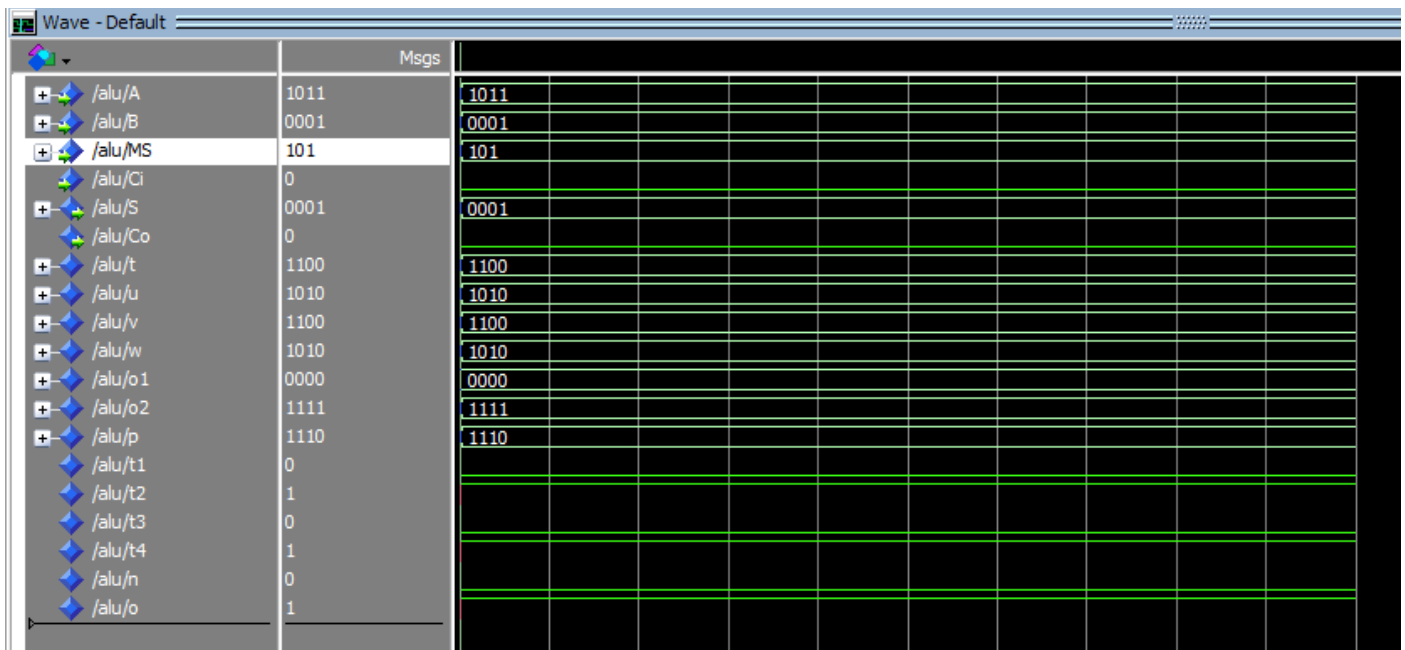
### Operation 5: A

Here, we will put  $A = 1010$  and output should be  $A$  itself, i.e.  $S = A$ . Hence, the operation is verified by the following simulation.



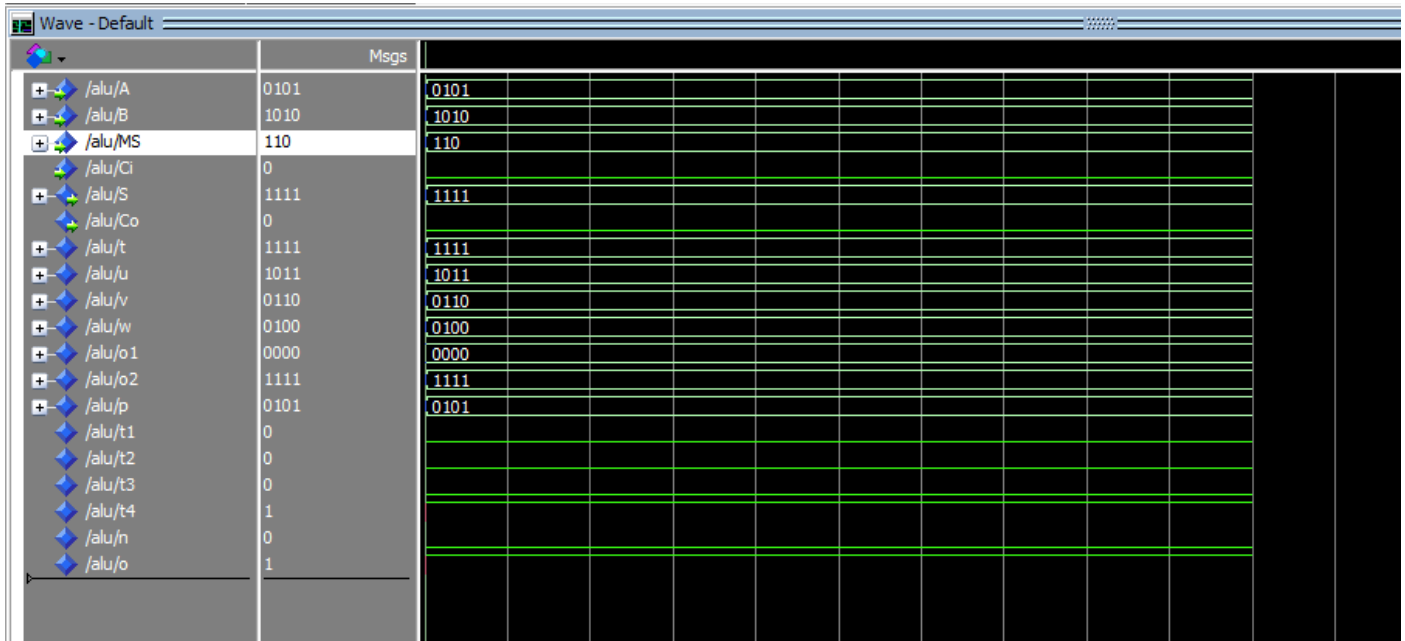
### Operation 6 : A AND B

Here, we will put  $A = 1011$  and  $B = 0001$ , output should be  $S = 0001$ . Hence, the operation is verified by the following simulation.



### Operation 7 : A OR B

Here, we will put A = 1010 and B=0101 , output will be S=1111. Hence, the operation is verified by the following simulation.



### Operation 8: A'

Here, we will put A = 1011 and output should be S=0100. Hence, the operation is verified by the following simulation.

