

CS201P Assignment 1

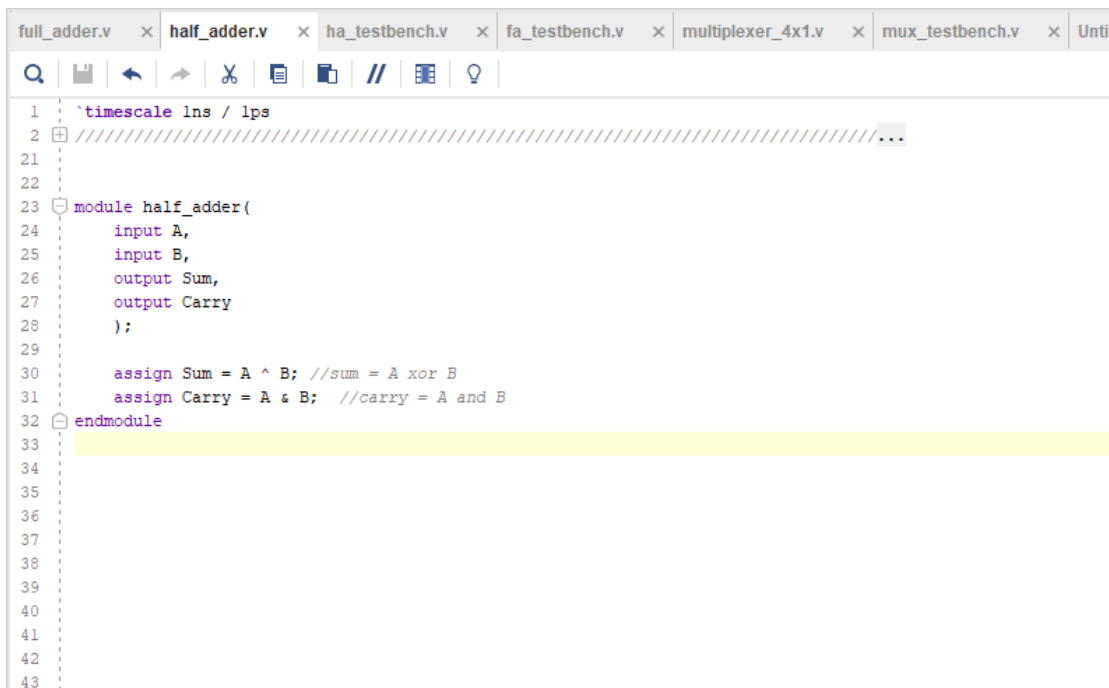
Group 23

B20218 Narmit Kumar

B20217 Monika Meena

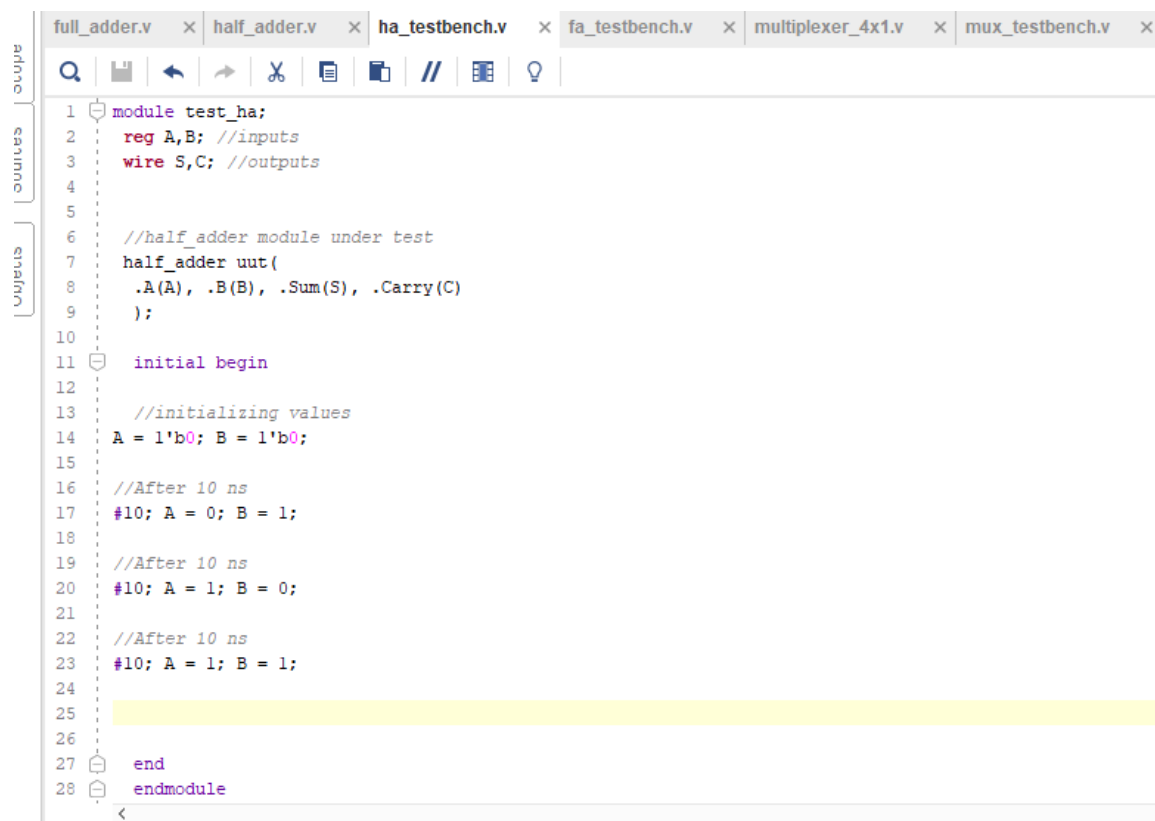
1st and 2nd Question

Half adder code:



```
1  `timescale 1ns / 1ps
2  ////////////////////////////////////////////...
21
22
23 module half_adder(
24     input A,
25     input B,
26     output Sum,
27     output Carry
28 );
29
30     assign Sum = A ^ B; //sum = A xor B
31     assign Carry = A & B; //carry = A and B
32 endmodule
33
34
35
36
37
38
39
40
41
42
43
```

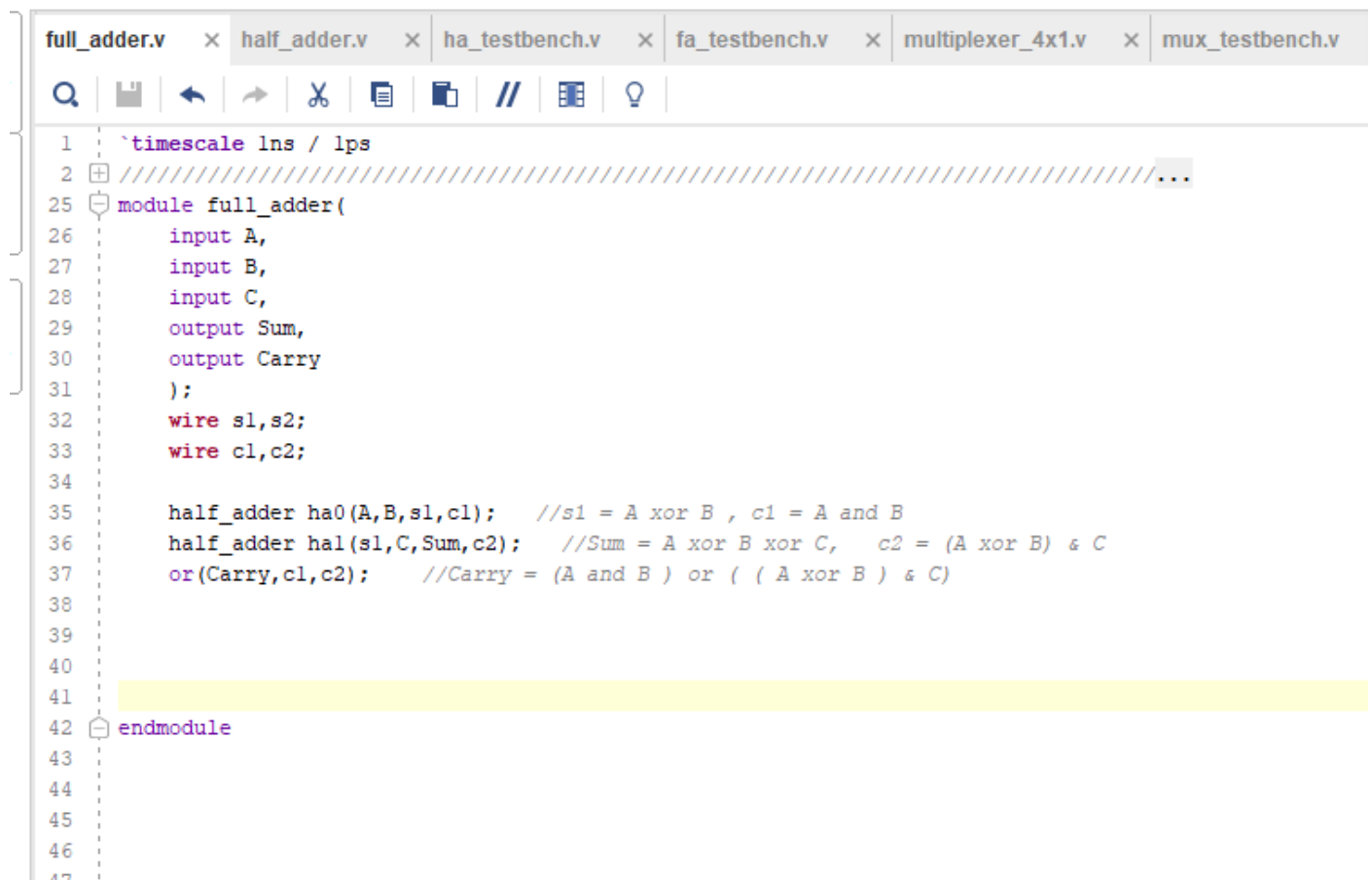
Half adder testbench:



The screenshot shows a software editor with a tabbed interface. The active tab is 'ha_testbench.v'. The code is a Verilog testbench for a half adder module. It includes a module declaration 'test_ha', input and output declarations, an instantiation of the 'half_adder' module, and an initial block with timing delays and input assignments. The editor has a sidebar on the left with icons for 'addio', 'simulate', and 'output'. The code is as follows:

```
1 module test_ha;
2     reg A,B; //inputs
3     wire S,C; //outputs
4
5
6     //half_adder module under test
7     half_adder uut(
8         .A(A), .B(B), .Sum(S), .Carry(C)
9     );
10
11     initial begin
12
13         //initializing values
14         A = 1'b0; B = 1'b0;
15
16         //After 10 ns
17         #10; A = 0; B = 1;
18
19         //After 10 ns
20         #10; A = 1; B = 0;
21
22         //After 10 ns
23         #10; A = 1; B = 1;
24
25
26
27     end
28 endmodule
```

Ful Adder code:



The screenshot shows a software editor with a tabbed interface. The active tab is 'full_adder.v'. The code is a Verilog module for a full adder. It includes a timescale declaration, input and output declarations, internal signal declarations, and logic for two half adders and an OR gate. The editor has a sidebar on the left with icons for 'addio', 'simulate', and 'output'. The code is as follows:

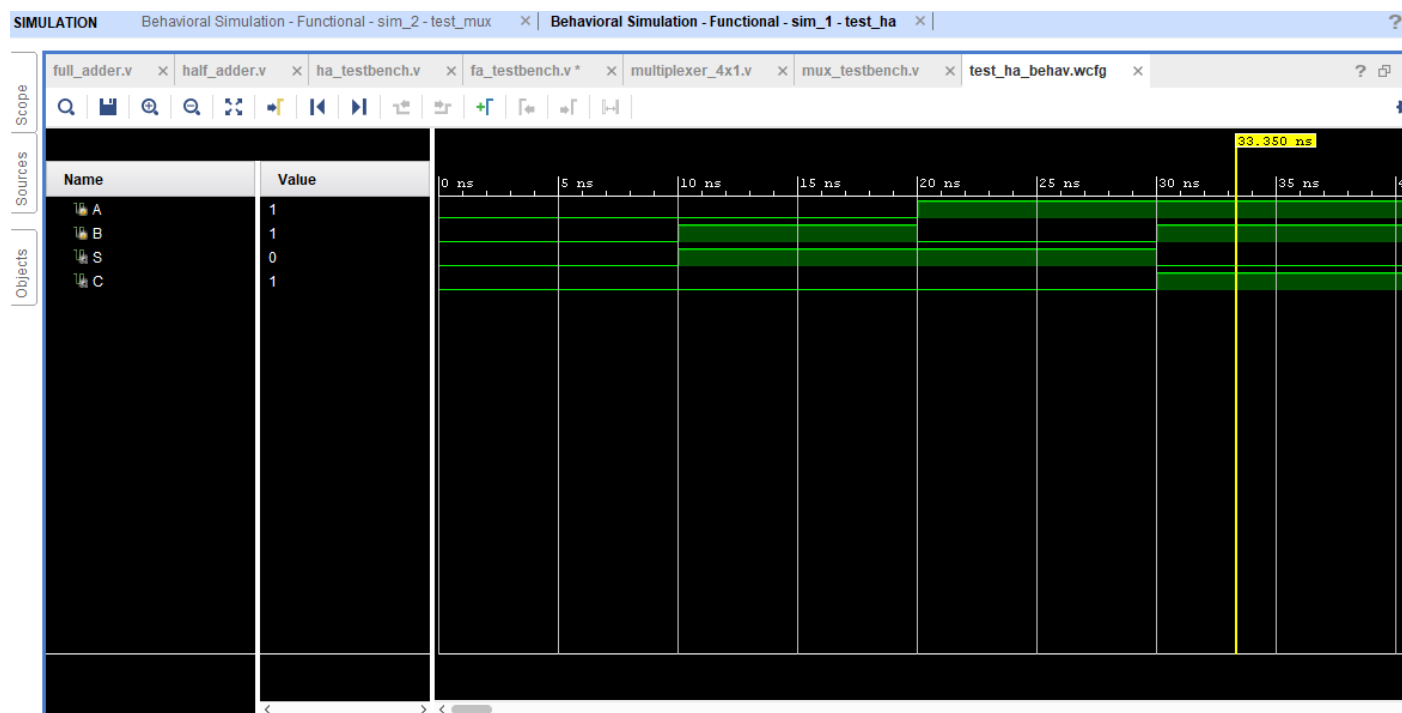
```
1 `timescale 1ns / 1ps
2 ////////////////////////////////////////...
25 module full_adder(
26     input A,
27     input B,
28     input C,
29     output Sum,
30     output Carry
31 );
32     wire s1,s2;
33     wire c1,c2;
34
35     half_adder ha0(A,B,s1,c1); //s1 = A xor B , c1 = A and B
36     half_adder ha1(s1,C,Sum,c2); //Sum = A xor B xor C, c2 = (A xor B) & C
37     or(Carry,c1,c2); //Carry = (A and B) or ((A xor B) & C)
38
39
40
41
42 endmodule
43
44
45
46
47
```

Ful Adder testbench:

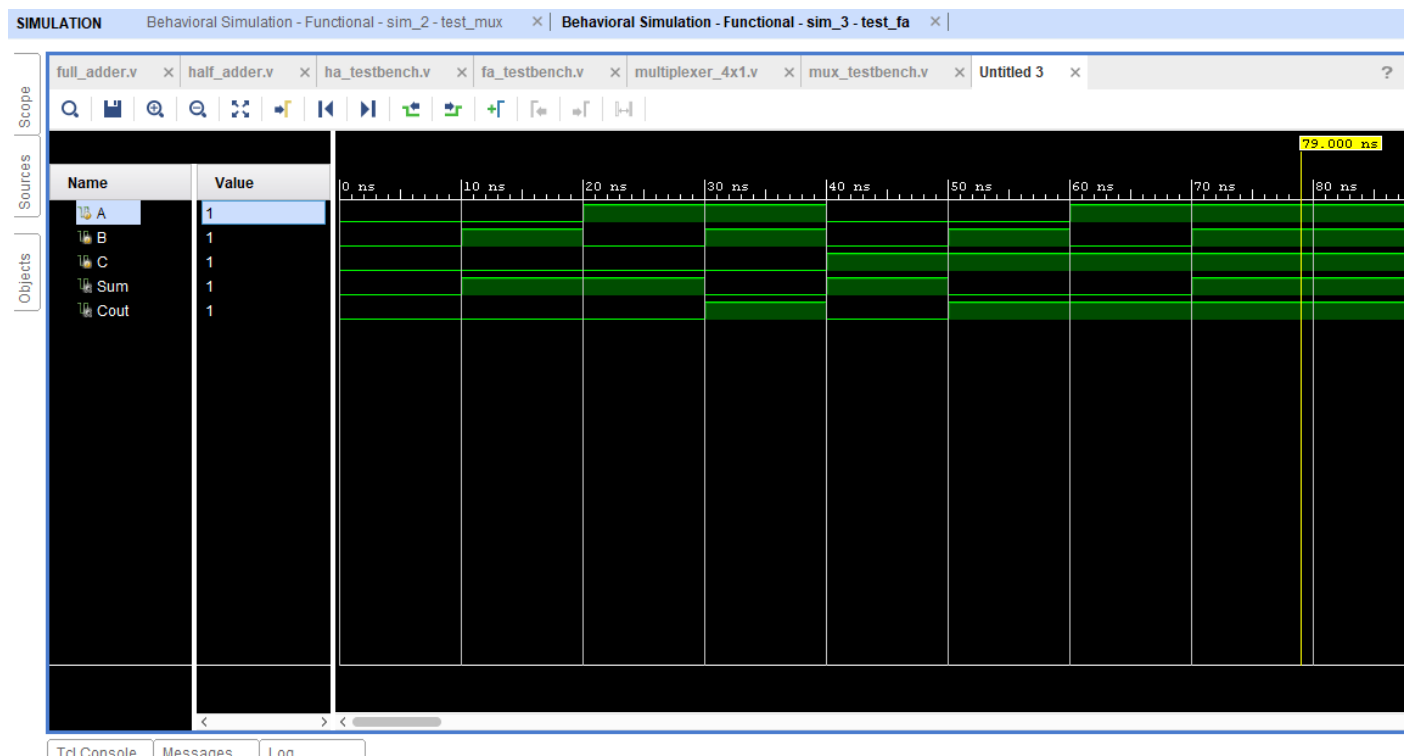
```
SIMULATION - Behavioral Simulation - Functional - sim_2 - test_mux
full_adder.v x half_adder.v x ha_testbench.v x fa_testbench.v * x multiplexer_4x1.v x mux_testbench.v x
Scope
Sources
Objects
1 `timescale 1ns / 1ps
2 ////////////////////////////////////////////...
21
22 module test_fa;
23     reg A,B,C; //inputs
24     wire Sum,Cout; //outputs
25
26
27     //full_adder under test
28     full_adder uut(
29         .A(A), .B(B), .C(C), .Sum(Sum), .Carry(Cout)
30     );
31
32     initial begin
33         C = 1'b0; A = 1'b0; B = 1'b0; //initializing values to inputs
34         #10; C = 0; A = 0; B = 1; //after 10 seconds we change our input values
35         #10; C = 0; A = 1; B = 0;
36         #10; C = 0; A = 1; B = 1;
37         #10; C = 1; A = 0; B = 0;
38         #10; C = 1; A = 0; B = 1;
39         #10; C = 1; A = 1; B = 0;
40         #10; C = 1; A = 1; B = 1;
41
42
43     end
44 endmodule
45
46
```

Outputs:

Half Adder –



Ful Adder-



3rd and 4th Question

Multiplexer 4 to 1 code:

full_adder.v x half_adder.v x ha_testbench.v x fa_testbench.v x multiplexer_4x1.v * x mux_testbe

Scope
Sources
Objects

```

1 module mux4_1(
2     input a,b,c,d,s0,s1,
3     output out
4 );
5
6 //for 4 x 1 Multiplexer
7 // S0 = 0 S1 = 0 Output = A
8 // S0 = 1 S1 = 0 Output = B
9 // S0 = 0 S1 = 1 Output = C
10 // S0 = 1 S1 = 1 Output = D
11 //Above outputs are given by below coding expression
12 //data flow model is being used here.
13
14 assign out = s1 ? (s0 ? d : c) : (s0 ? b : a);
15
16
17 endmodule
18

```

Testbench:

SIMULATION Behavioral Simulation - Functional - sim_2 - test_mux x | Behavioral Simulation - Functional - sim_3 - test_fa x |

full_adder.v x half_adder.v x ha_testbench.v x fa_testbench.v x multiplexer_4x1.v * mux_testbench.v * x Untitled 3 x

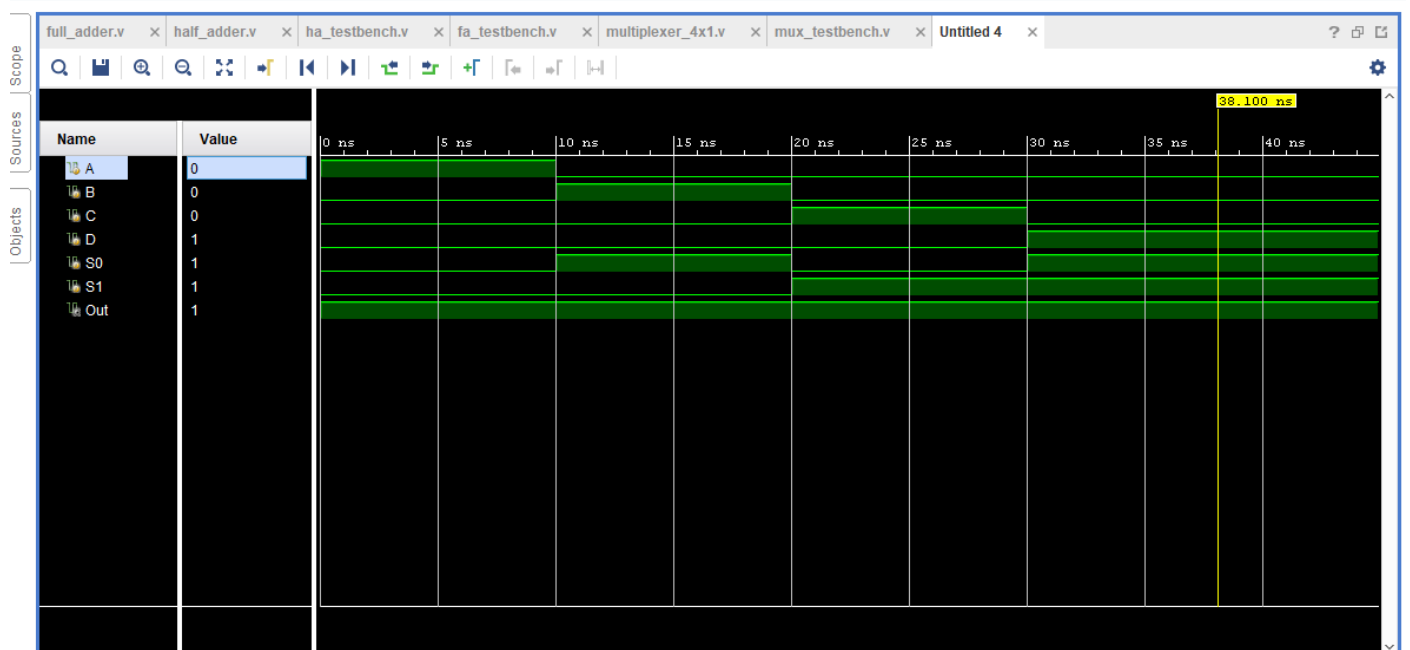
Scope Sources Objects

```

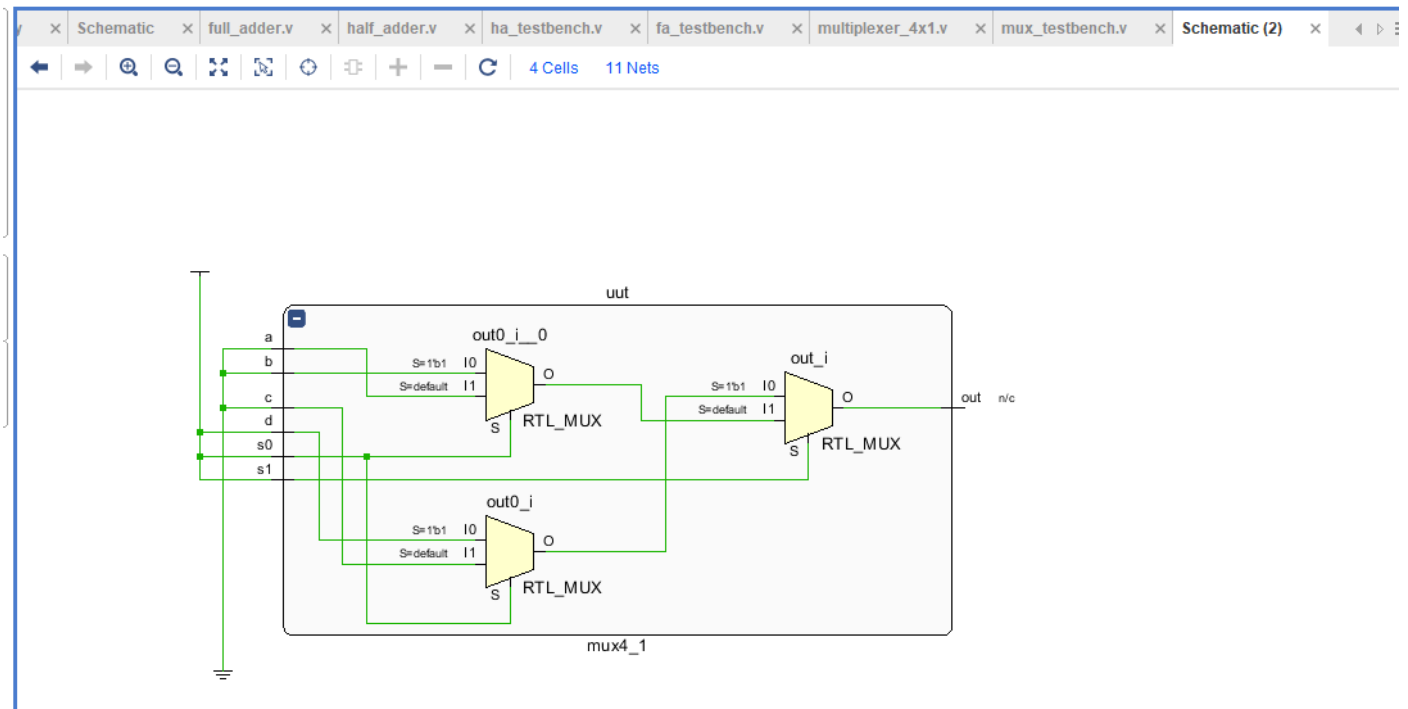
1
2
3 `timescale 1ns / 1ps
4 //////////////////////////////////////////////////...
23
24 module test_mux;
25     reg A,B,C,D,S0,S1; //inputs
26     wire Out; //outputs
27
28
29     //multiplexer under test
30     mux4_1 uut(
31         .a(A), .b(B), .c(C), .d(D), .s0(S0), .s1(S1), .out(Out)
32     );
33
34     initial begin
35         A = 1'b1; B = 1'b0; C = 1'b0; D = 1'b0; S0 = 1'b0; S1 = 1'b0; //initializing values
36         #10 S0 = 1; S1 = 0; A = 0; B = 1; C = 0; D = 0; //after 10 ns input is updated
37         #10 S0 = 0; S1 = 1; A = 0; B = 0; C = 1; D = 0;
38         #10 S0 = 1; S1 = 1; A = 0; B = 0; C = 0; D = 1;
39
40     end
41 endmodule
42
43
44
45
46

```

Output:



RTL Schematic of MUX



Synthesized Design

