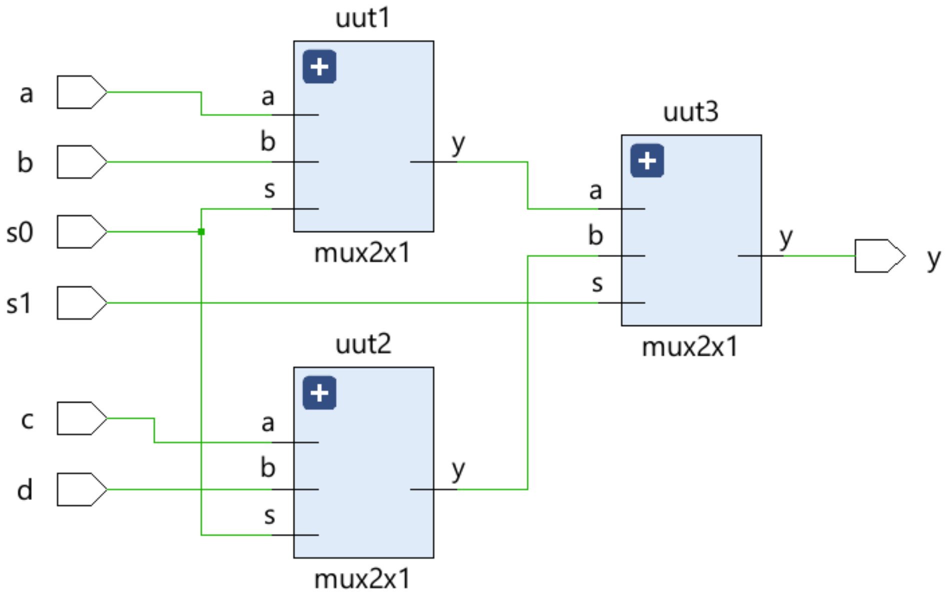
2. Design a 4:1 multiplexer using 2:1 multiplexers.

Code for 2x1 MUX

```
module mux2x1(
input  a,b,s,
output y );
assign y = (s)?b:a;
endmodule
```

Code for 4x1 MUX

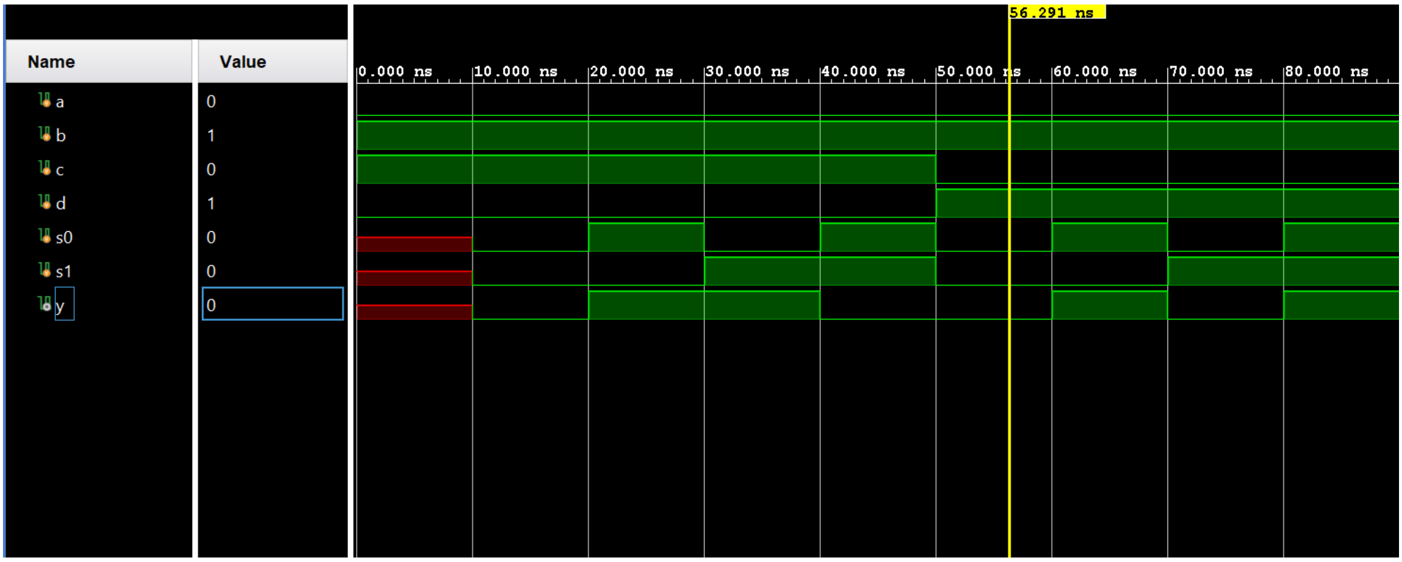
```
module mux4x1(
input a,b,c,d,s0,s1,
output y
    );
wire w1, w2;
mux2x1 uut1(
.a(a), .b(b), .s(s0), .y(w1)
);
mux2x1 uut2(
.a(c), .b(d), .s(s0), .y(w2)
);
mux2x1 uut3(
.a(w1), .b(w2), .s(s1), .y(y)
);
endmodule
```



TestBench:

```
`timescale 1ns / 1ps


module tb_mux4x1();

reg a,b,c,d,s0,s1;

wire y;


mux4x1 uut(.a(a), .b(b), .c(c), .d(d), .s0(s0), .s1(s1), .y(y));

initial begin

   a=0; b=1; c=1; d=0; #10

   s0=0; s1=0; #10

   s0=1; s1=0; #10

   s0=0; s1=1; #10

   s0=1; s1=1; #10


   a=0; b=1; c=0; d=1; s0=0; s1=0; #10

   s0=1; s1=0; #10

   s0=0; s1=1; #10

   s0=1; s1=1; #10

   $stop;

end

endmodule
```

Simulation Results:



Name	Value
a	0
b	1
c	0
d	1
s0	0
s1	0
y	0

- **We learnt how to instantiate components.**
- In Verilog, reg is used for variables that hold values and can be assigned within an initial or always block. Since the testbench needs to assign values to the inputs of the mux4x1, we declare a, b, c, d, s0, s1 as reg.
- Wire is used for signals that are driven by the instantiated module (i.e., mux4x1 in this case). The output y is computed by combinational logic inside mux4x1, so we declare it as a wire.

`reg`	Testbench	Stores values for driving inputs
`wire`	Testbench	Reads values from the module output
`input`	Design module	Receives signals from the testbench
`output`	Design module	Sends computed values to the testbench

The timescale directive in Verilog specifies the time unit and time precision for the simulation.

`timescale 1ns / 1ps

Breaking It Down:

1. 1ns (Time Unit)
   - This means that all delay values (like #10) are interpreted in nanoseconds.
   - Example: #10 means 10 nanoseconds.
2. 1ps (Time Precision)
   - This defines the smallest resolution of time the simulator will consider.
   - Here, 1 picosecond is the smallest time step that the simulator will track.