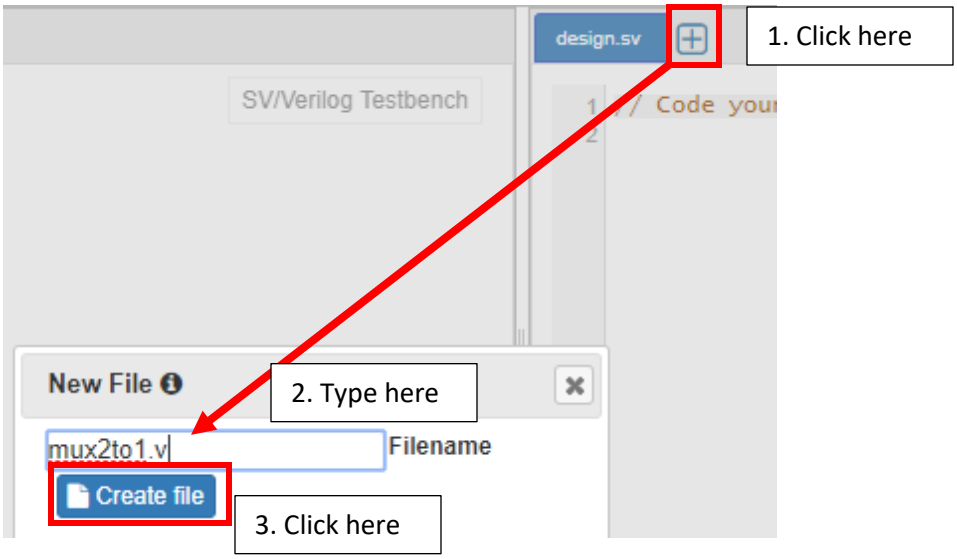
CECS 225: MUX TREE LAB

OBJECTIVE: Create a **4 to 1 mux** using multiple **2 to 1 multiplexers**

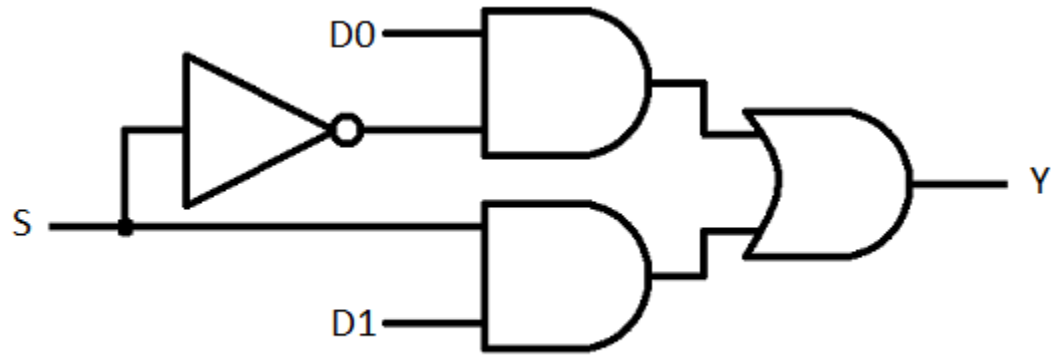
PROCEDURE: In a new project space on EDA playground, create a new source file named mux2to1.v



Create a module skeleton for your mux 2 to 1 using the exact same module name, input variables, and output variable as shown in the image below.


```
1  module mux2to1(input d0, d1, s,
2                 output y);
3
4
5  endmodule
6
```

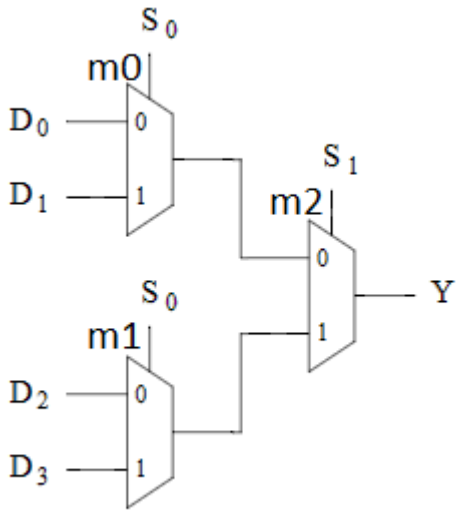
Use image below to complete the module skeleton and make a 2 to 1 multiplexer.



CECS 225: MUX TREE LAB

The design.sv file will contain what is known as your top level module. In hierarchical design, the module that connects together all other modules will be known as the top level module.

Create a top level module which interconnects multiple instances of the mux2to1 according to the block diagram below:



D0, D1, D2, and D3 are data inputs

S0 and S1 are select lines (control inputs)

m0, m1, and m2 are instance labels (like the variables used to name logic gates when using gate level primitives)

Y is the output

Use the following module skeleton to get started:

```
`include "mux2to1.v"     //import mux2to1
module mux4to1(input d3,d2,d1,d0,s1,s0,
               output y);
  wire mux1out, mux0out;

  mux2to1
  m0(                ), //1st mux2to1 instance
  m1(                ), //2nd mux2to1 instance
  m2(                   ); //3rd mux2to1 instance
endmodule
```

Pass variables into the port list parameter positions where the order of parameters must match the definition of the mux2to1 module.

Once you have modeled the circuit, its proper functionality can be verified using the tester on the following page.

CECS 225: MUX TREE LAB

SV/Verilog Testbe

```
module mux4to1tester;
  reg d3,d2,d1,d0,s1,s0;
  wire y;

  mux4to1 dut(d3,d2,d1,d0,s1,s0,y);
  integer i;
  initial begin
    $display("Mux 4 to 1 tester!");
    d0 = 1'b1;    d1 = 1'b0;
    d2 = 1'bx;    d3 = 1'b1;
    $display("\tTest case\td3\td2\td1\td0\t|\ts1\ts0\t|\ty");
    for(i = 0; i < 4; i = i +1)
       begin
         {s1,s0}=i;
         #1 $display("%d\t\t%b\t%b\t%b\t%b\t|\t%b\t%b\t|\t%b",i,d3,d2,d1,d0,s1,s0,y);
       end
    $finish;
  end
endmodule
```

Correct test results will appear as follows:

```
[2017-09-26 00:51:39 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
Mux 4 to 1 tester!
        Test case       d3      d2      d1      d0      |       s1      s0      |       y
          0             1       x       0       1       |       0       0       |       1
          1             1       x       0       1       |       0       1       |       0
          2             1       x       0       1       |       1       0       |       x
          3             1       x       0       1       |       1       1       |       1
Done
```

WHAT TO TURN IN: Once your mux4to1 module is completed and proper test results are produced:

- Copy the contents of your **mux4to1** module to a file named **mux4to1.txt**
- Upload **mux4to1.txt** to the beachboard dropbox for **mux4to1_lab**