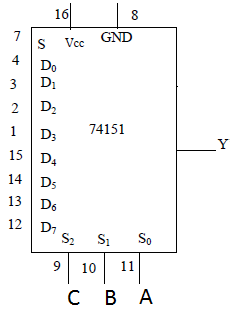
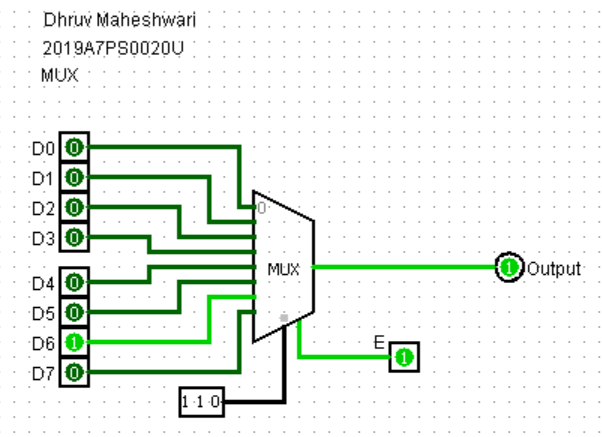
**Hardware runs**

**Run 1: Multiplexer**

**Diagram**

****

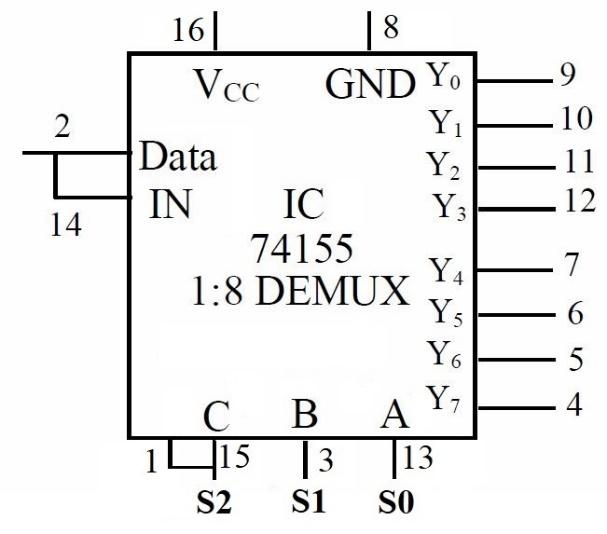
****

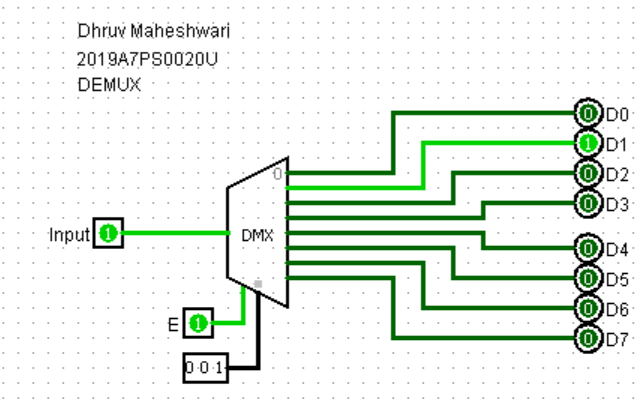
**Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S** | **C** | **B** | **A** | **Y** | **Y’** |
| 1 | 0 | 0 | 0 | D0 | ~D0 |
| 1 | 0 | 0 | 1 | D1 | ~D1 |
| 1 | 0 | 1 | 0 | D2 | ~D2 |
| 1 | 0 | 1 | 1 | D3 | ~D3 |
| 1 | 1 | 0 | 0 | D4 | ~D4 |
| 1 | 1 | 0 | 1 | D5 | ~D5 |
| 1 | 1 | 1 | 0 | D6 | ~D6 |
| 1 | 1 | 1 | 1 | D7 | ~D7 |
| 0 | x | x | x | 0 | 1 |

**Run 2: Demultiplexer**

**Diagram**





**Truth Table**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data** | **C** | **B** | **A** | **Y0** | **Y1** | **Y2** | **Y3** | **Y4** | **Y5** | **Y6** | **Y7** |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Software runs**

**Run 3: Multiplexer**

**1**. Write the Verilog code and testbench for 2:1 Multiplexer using data flow modeling

**A: Verilog Code and testbench-** [**https://www.edaplayground.com/x/udng**](https://www.edaplayground.com/x/udng)

**Verilog:**

module mux\_2\_1(input a,b,s, output z);

assign z= (~s&a)| s&b;

endmodule

**Testbench:**

module testbench\_mux\_2\_1;

reg p,q,select\_line;

wire out;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_mux\_2\_1);

#000 p = 1; q=0; select\_line=0;

#100 p = 1; q=0; select\_line=1;

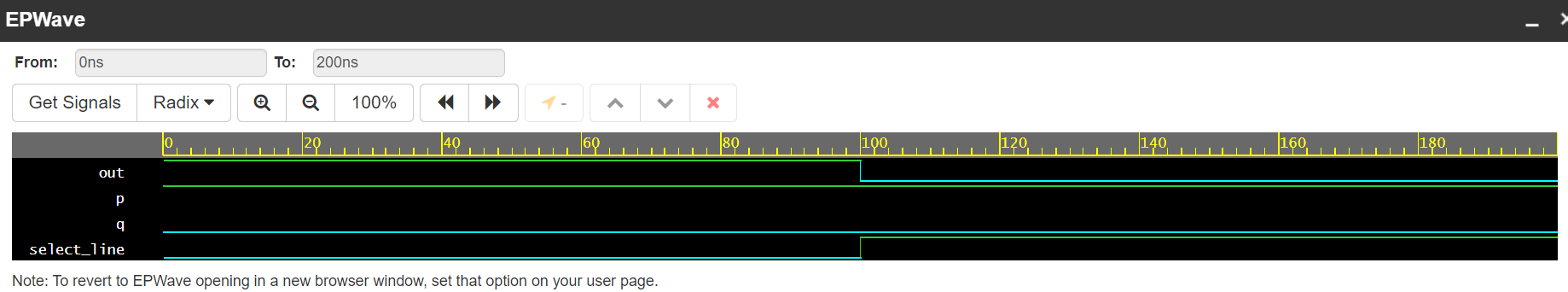
#100 $stop;

end

mux\_2\_1 U2(p,q,select\_line,out);

endmodule

**Q:** Paste the screenshot of waveform window where you get the waveforms for the above code. (online students also paste the URL)

**A: **

**2.** Write the Verilog code and testbench for 2:1 Multiplexer using behavioral modeling.

**A: Verilog Code and testbench -** [**https://www.edaplayground.com/x/qKZb**](https://www.edaplayground.com/x/qKZb)

**Verilog:**

module mux\_2\_1\_beh(input a,b,s, output reg z);

always@(a,b,s)

begin

if(s)

z=b;

else

z=a;

end

endmodule

**Testbench:**

module testbench\_mux\_2\_1;

reg p,q,select\_line;

wire out;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_mux\_2\_1);

#000 p = 1; q=0; select\_line=0;

#100 p = 1; q=0; select\_line=1;

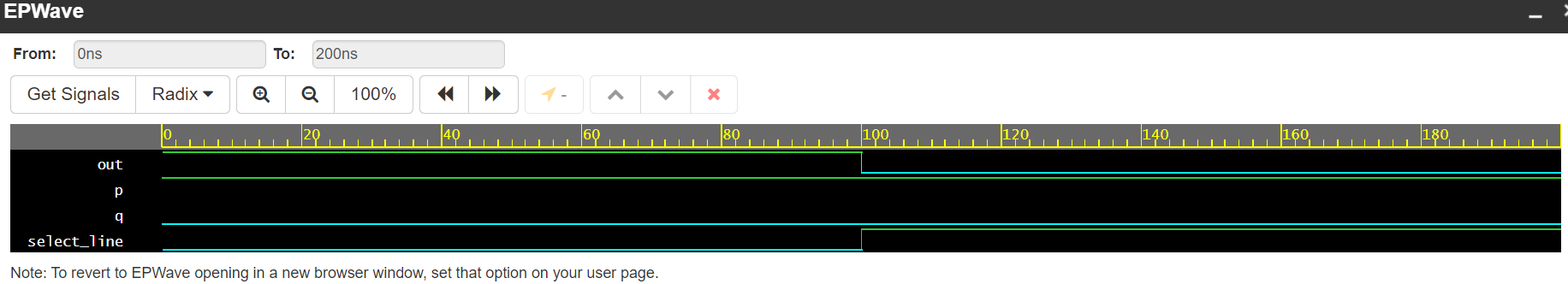
#100 $stop;

end

mux\_2\_1\_beh U2(p,q,select\_line,out);

endmodule

**Q:** Paste the screenshot of waveform window where you get the waveforms for the above code. (online students also paste the URL)

**A: **

**3.** Write the Verilog code and testbench for 4:1 Multiplexer using structural modeling.

**A: Verilog Code and testbench -** [**https://www.edaplayground.com/x/Tnyw**](https://www.edaplayground.com/x/Tnyw)

**Verilog:**

module mux\_2\_1(input a,b,s, output z);

assign z= (~s&a)| s&b;

endmodule

module mux\_4\_1(input [3:0]I, input [1:0]S, output out);

wire w1,w2;

mux\_2\_1 U1(I[0],I[1],S[0],w1);

mux\_2\_1 U2(I[2],I[3],S[0],w2);

mux\_2\_1 U3(w1,w2,S[1],out);

endmodule

**Testbench:**

module testbench\_mux\_4\_1;

reg [3:0]Input;

reg [1:0]select\_line;

wire out;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_mux\_4\_1);

#000 Input=4'b1010; select\_line=2'b00;

#100 select\_line=2'b01;

#100 select\_line=2'b10;

#100 select\_line=2'b11;

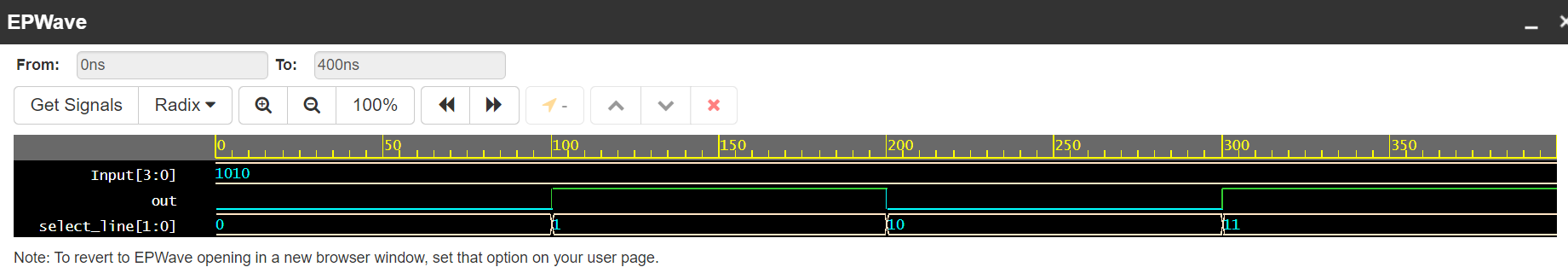
#100 $stop;

end

mux\_4\_1 U2(Input,select\_line,out);

endmodule

**Q:** Paste the screenshot of waveform window where you get the waveforms for the above code. (online students also paste the URL)

**A: **

**Run 4: Demultiplexer**

1. Write the Verilog code and testbench for 1:4 demultiplexer using behavioral modeling.

**A: Verilog Code and testbench -** [**https://www.edaplayground.com/x/GWhG**](https://www.edaplayground.com/x/GWhG)

**Verilog:**

module demux\_1\_4\_beh(input data, [1:0]s, output reg [3:0]out);

always@(s,data)

case(s)

2'b00 :

begin

out[0] = data;

out[1] = 0; out[2] = 0;out[3] = 0;

end

2'b01 :

begin

out[1] = data;

out[0] = 0; out[2] = 0;out[3] = 0;

end

2'b10 :

begin

out[2] = data;

out[0] = 0; out[1] = 0;out[3] = 0;

end

2'b11 :

begin

out[3] = data;

out[0] = 0; out[1] = 0;out[2] = 0;

end

endcase

endmodule

**Testbench:**

module testbench\_demux\_1\_4\_beh;

reg data=1;

reg [1:0]select\_line;

wire [3:0]out;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_demux\_1\_4\_beh);

#000 select\_line=2'b00;

#100 select\_line=2'b01;

#100 select\_line=2'b10;

#100 select\_line=2'b11;

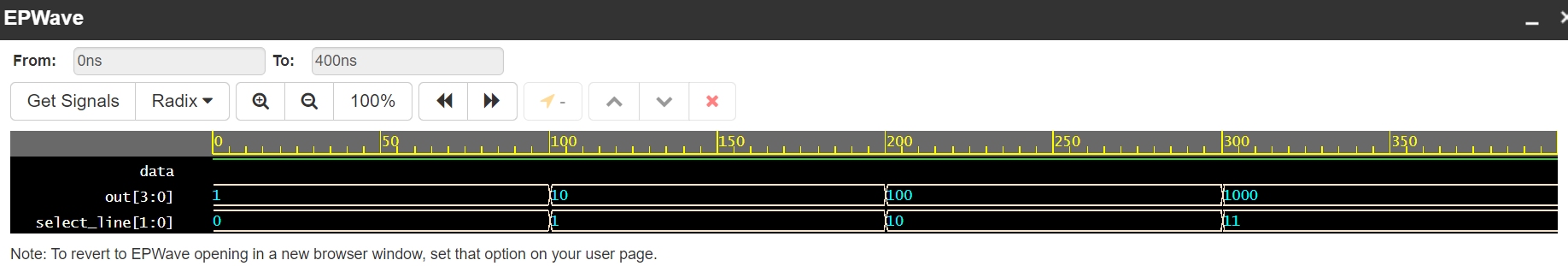
#100 $stop;

end

demux\_1\_4\_beh U1(data,select\_line,out);

endmodule

**Q:** Paste the screenshot of waveform window where you get the waveforms for the above code. (online students also paste the URL)

**A: **

1. Write the Verilog code and testbench for 1:8 demux using data flow modeling.

**A: Verilog Code and testbench-** [**https://www.edaplayground.com/x/mXmi**](https://www.edaplayground.com/x/mXmi)

**Verilog:**

module demux\_1\_8(input data, [2:0]lines, output [7:0] out);

assign out[0]=(data & ~lines[2] & ~lines[1] & ~lines[0]);

assign out[1]=(data & ~lines[2] & ~lines[1] & lines[0]);

assign out[2]=(data & ~lines[2] & lines[1] & ~lines[0]);

assign out[3]=(data & ~lines[2] & lines[1] & lines[0]);

assign out[4]=(data & lines[2] & ~lines[1] & ~lines[0]);

assign out[5]=(data & lines[2] & ~lines[1] & lines[0]);

assign out[6]=(data & lines[2] & lines[1] & ~lines[0]);

assign out[7]=(data & lines[2] & lines[1] & lines[0]);

endmodule

**Testbench:**

module testbench\_demux\_1\_8;

reg data=1;

reg [2:0]select\_line;

wire [7:0]out;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_demux\_1\_8);

#000 select\_line=3'b000;

#100 select\_line=3'b001;

#100 select\_line=3'b010;

#100 select\_line=3'b011;

#100 select\_line=3'b100;

#100 select\_line=3'b101;

#100 select\_line=3'b110;

#100 select\_line=3'b111;

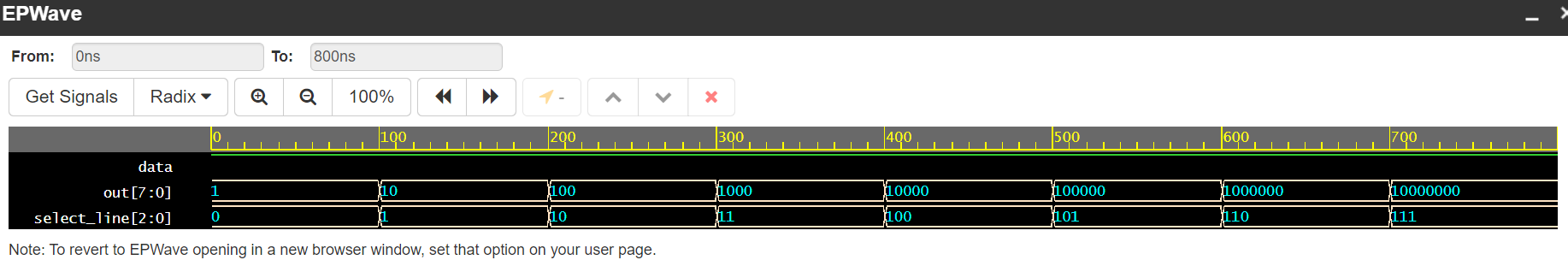
#100 $stop;

end

demux\_1\_8 U1(data,select\_line,out);

endmodule

**Q:** Paste the screenshot of waveform window where you get the waveforms for the above code. (online students also paste the URL)

**A: **