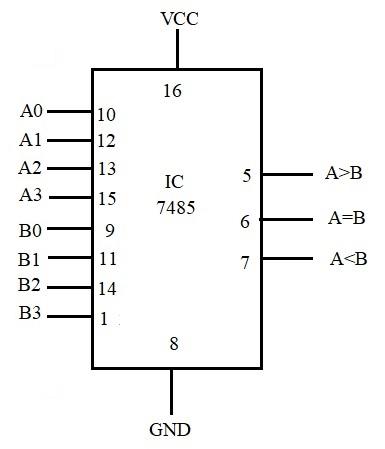
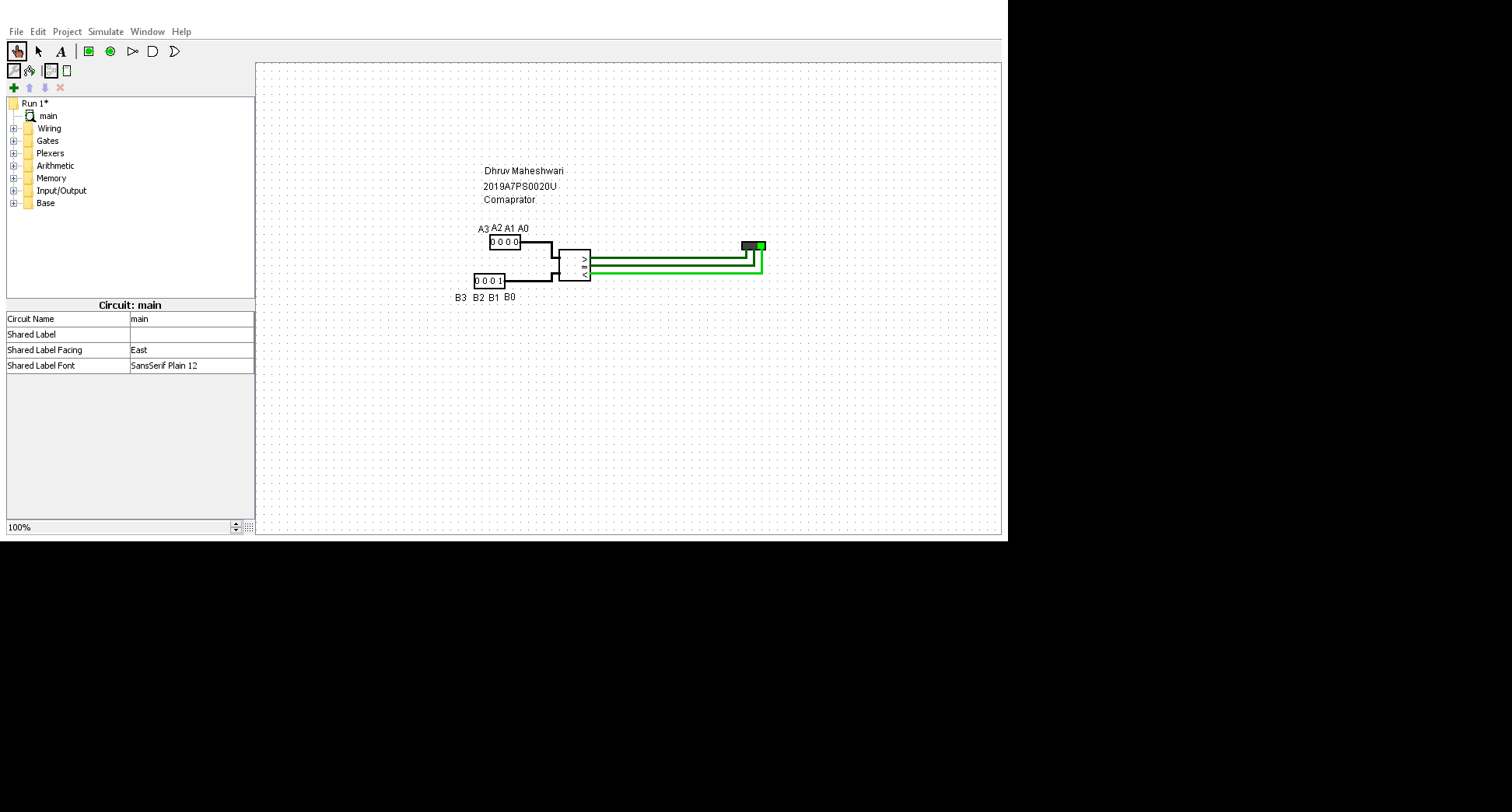
**Hardware runs**

**Run 1: Comparator**

**Diagram**



****

**Truth Table**

**The MSB is taken as sign bit**

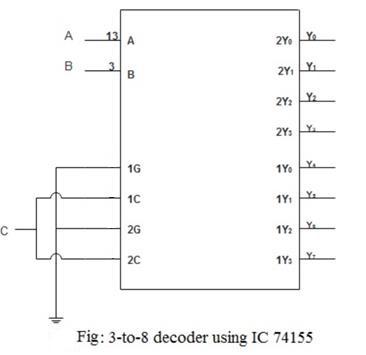
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **A>B** | **A=B** | **A<B** |
| 0000 | 1111 | 1 | 0 | 0 |
| 1111 | 0000 | 0 | 0 | 1 |
| 0000 | 0000 | 0 | 1 | 0 |
| 1111 | 1111 | 0 | 1 | 0 |
| 0101 | 1000 | 1 | 0 | 0 |
| 1010 | 0011 | 0 | 0 | 1 |
| 1000 | 0101 | 0 | 0 | 1 |
| 0011 | 1010 | 1 | 0 | 0 |
| 0001 | 1000 | 1 | 0 | 0 |
| 1001 | 1010 | 0 | 0 | 1 |

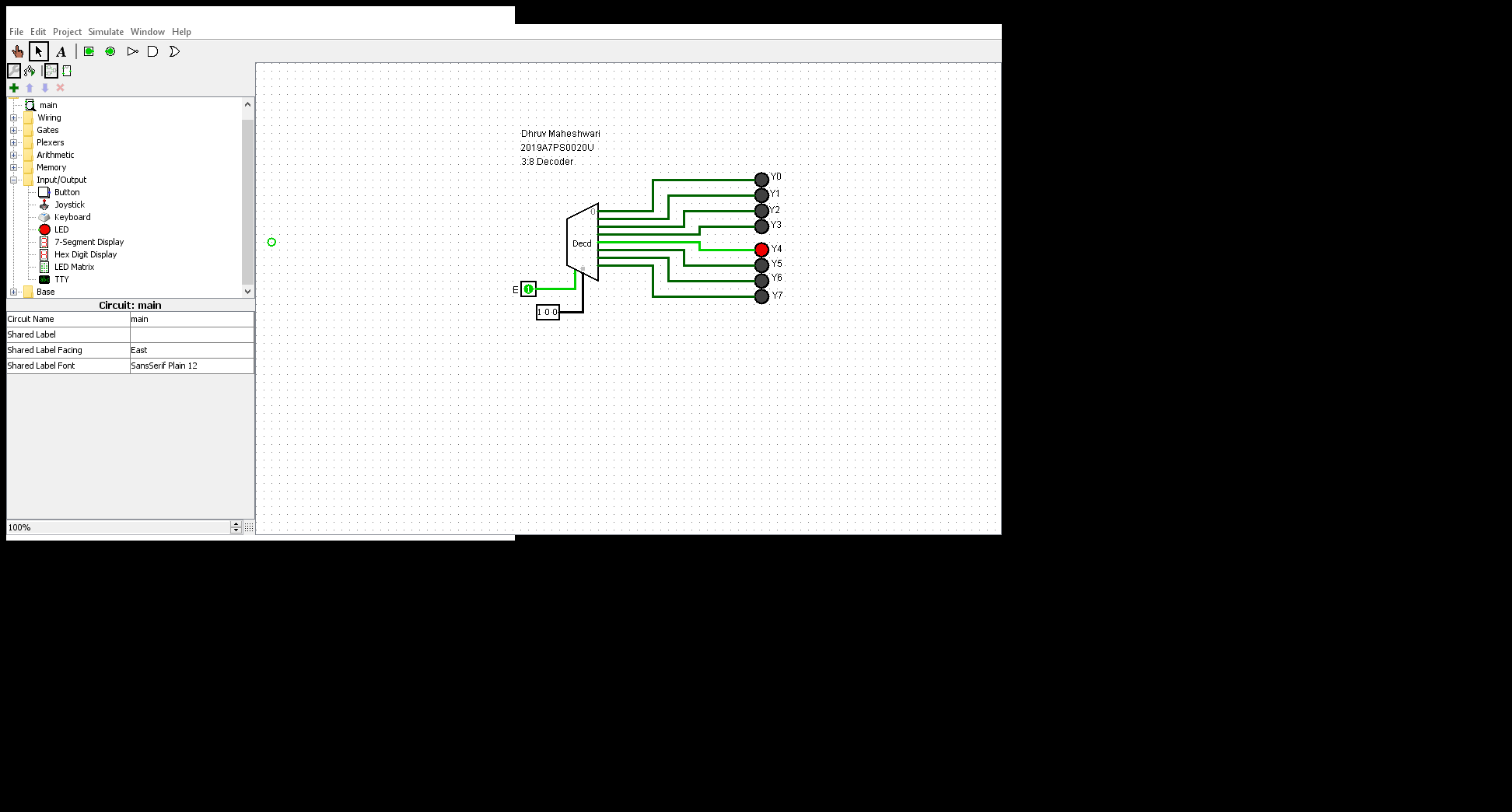
**Q:** Name the gates that can be used as one-bit comparators.

**A: not, and, xor**

**Run 2: Decoder**

**Diagram**



****

**Truth Table**

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

**Q: Are the outputs of the decoder active low or active high?**

**A: Active high**

**Q: What external gate would have been used if the IC were to be active high?**

**A: no external gate is required, as they are active high only.**

**Software runs**

**Run 3: Comparator**

1. Write the Verilog code and testbench for 4-bit comparator using data flow modeling. (Hint: Use >, < and = = to compare the two numbers).

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

**Verilog:**

module four\_bit(input [3:0]a, [3:0]b, output reg L,E,G);

assign L=(a<b);

assign E=(a==b);

assign G=(a>b);

endmodule

**Testbench:**

module testbench\_four\_bit;

reg [3:0]p;

reg [3:0]q;

wire L,E,G;

initial

begin

$dumpfile ("dump.vcd");

$dumpvars (1, testbench\_four\_bit);

#00 p = 4'b1000; q = 4'b1111;

#10 p = 4'b1000; q = 4'b1000;

#10 p = 4'b1000; q = 4'b0111;

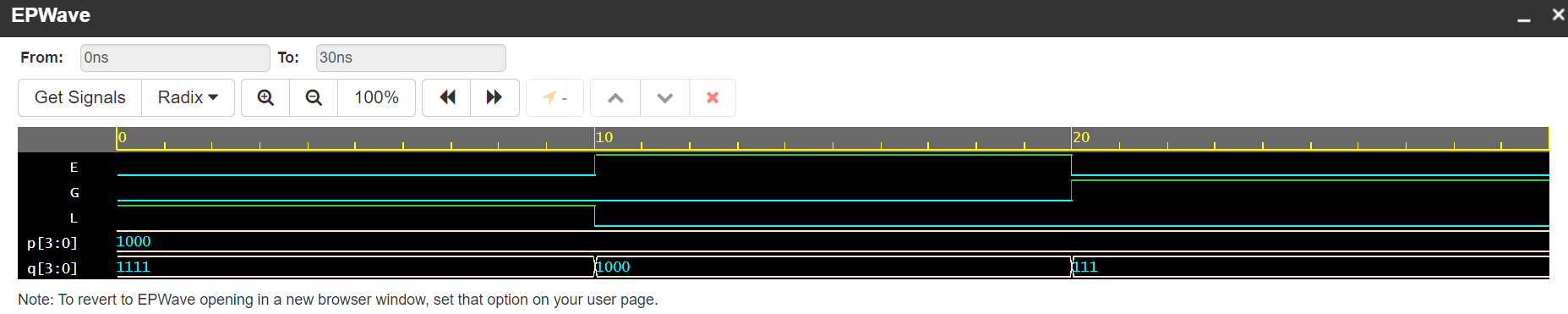
#10 $stop;

end

four\_bit U1(p,q,L,E,G);

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: Encoders and Decoders**

1. Write the verilog code and testbench for 2:4 decoder using data flow modeling, with active high output. (Hint: part of code is written below, A is input and D is output.)

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

**Verilog**

module decoder (A,B,E,D);

input A,B,E;

output [3:0] D;

assign D[0] =(~A & ~B & E);

assign D[1] =(~A & B & E);

assign D[2] =( A & ~B & E);

assign D[3] =( A & B & E);

endmodule

**Testbench**

module testbench\_decoder;

reg p,q,e;

wire [3:0]s;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_decoder);

#000 p = 0; q=0; e=1;

#100 p = 0; q=1; e=1;

#100 p = 1; q=0; e=1;

#100 p = 1; q=1; e=1;

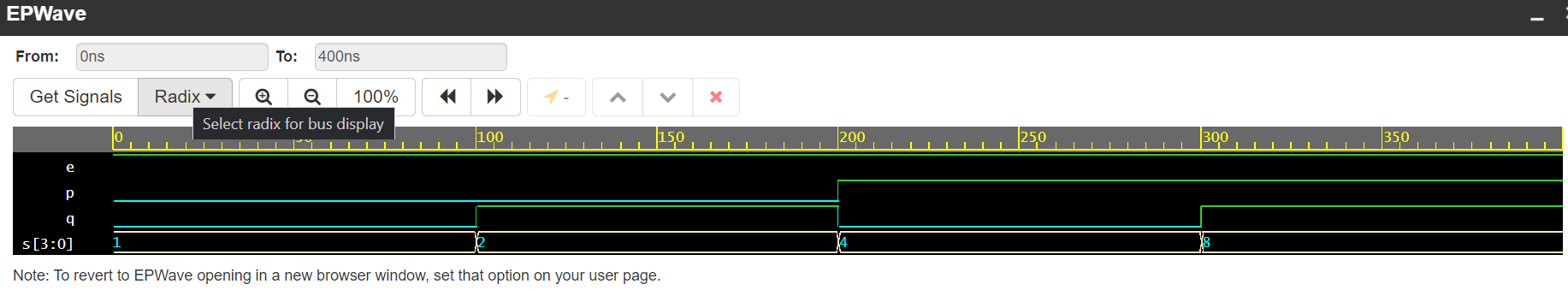
#100 $stop;

end

decoder U2(p,q,e,s);

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 2:4 decoder using data flow modeling, with active high output and active low enable pin. (Hint: use 1-bit enable input pin in all the assign statements like *D[3] = (A&B&~En)*).

**A: Verilog Code and testbench-** [**https://www.edaplayground.com/x/ead2**](https://www.edaplayground.com/x/ead2)

**Verilog**

module decoder\_active\_low\_enable (A,B,E,D);

input A,B,E;

output [3:0] D;

assign D[0] =(~A & ~B & ~E);

assign D[1] =(~A & B & ~E);

assign D[2] =( A & ~B & ~E);

assign D[3] =( A & B & ~E);

endmodule

**Testbench**

module testbench\_decoder;

reg p,q,e;

wire [3:0]s;

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1,testbench\_decoder);

#000 p = 0; q=0; e=0;

#100 p = 0; q=1; e=0;

#100 p = 1; q=0; e=0;

#100 p = 1; q=1; e=0;

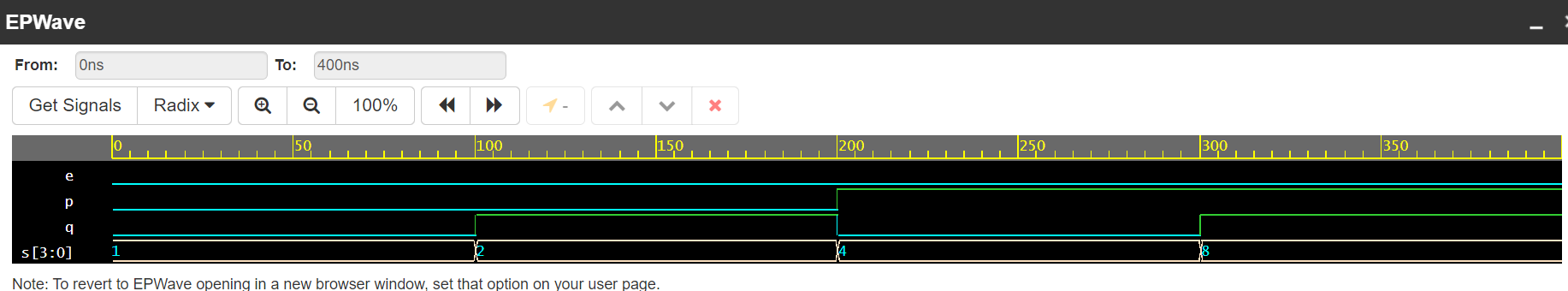
#100 $stop;

end

decoder\_active\_low\_enable U2(p,q,e,s);

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-bit (4-to-2) encoder using behavioral modeling. (Hint: Part of code is given below).

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

**Verilog**

module encoder\_4\_2(input [3:0]D, output reg [1:0]Y);

always@(D)

case(D)

4'h1: Y=2'b00;

4'h2: Y=2'b01;

4'h4: Y=2'b10;

4'h8: Y=2'b11;

default: Y=2'bxx;

endcase

endmodule

**Testbench**

module testbench\_encoder\_4\_2;

reg [3:0]inp;

wire [1:0]out;

initial

begin

$dumpfile ("dump.vcd");

$dumpvars (1, testbench\_encoder\_4\_2);

#00 inp = 4'b0001;

#10 inp = 4'b0010;

#10 inp = 4'b0100;

#10 inp = 4'b1000;

#10 inp = 4'b0000;

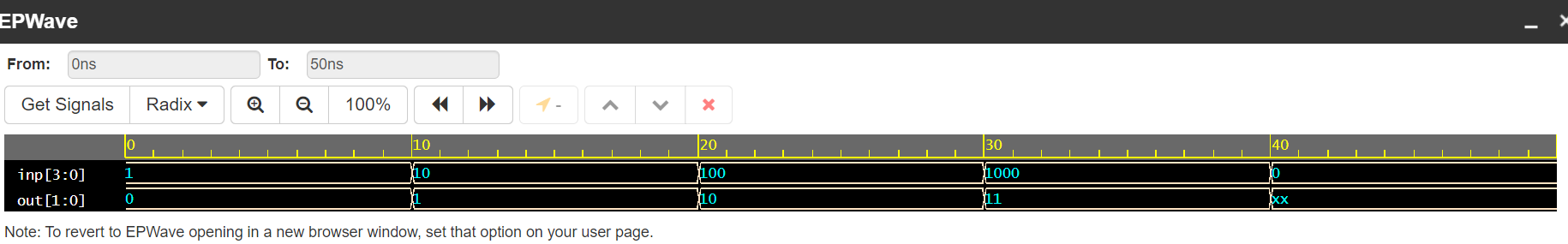
#10 $stop;

end

encoder\_4\_2 U1(inp,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:**

4. Write the Verilog code and testbench for 8:3 encoder using behavioral modeling. (Hint: use **case** statements)

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

**Verilog**

module encoder\_8\_3(input [7:0]D, output reg [2:0]Y);

always@(D)

case(D)

8'h1: Y=3'b000;

8'h2: Y=3'b001;

8'h4: Y=3'b010;

8'h8: Y=3'b011;

8'h10: Y=3'b100;

8'h20: Y=3'b101;

8'h40: Y=3'b110;

8'h80: Y=3'b111;

default: Y=2'bxx;

endcase

endmodule

**Testbench**

module testbench\_encoder\_8\_3;

reg [7:0]inp;

wire [2:0]out;

initial

begin

$dumpfile ("dump.vcd");

$dumpvars (1, testbench\_encoder\_8\_3);

#00 inp = 8'b00000001;

#10 inp = 8'b00000010;

#10 inp = 8'b00000100;

#10 inp = 8'b00001000;

#10 inp = 8'b00010000;

#10 inp = 8'b00100000;

#10 inp = 8'b01000000;

#10 inp = 8'b10000000;

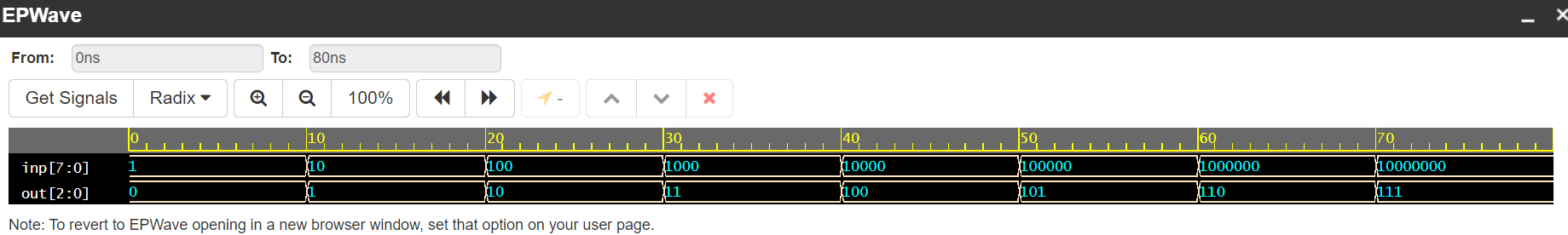
#10 $stop;

end

encoder\_8\_3 U1(inp,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:**