*11.Multiplexer(8*1)*

Verilog Code:

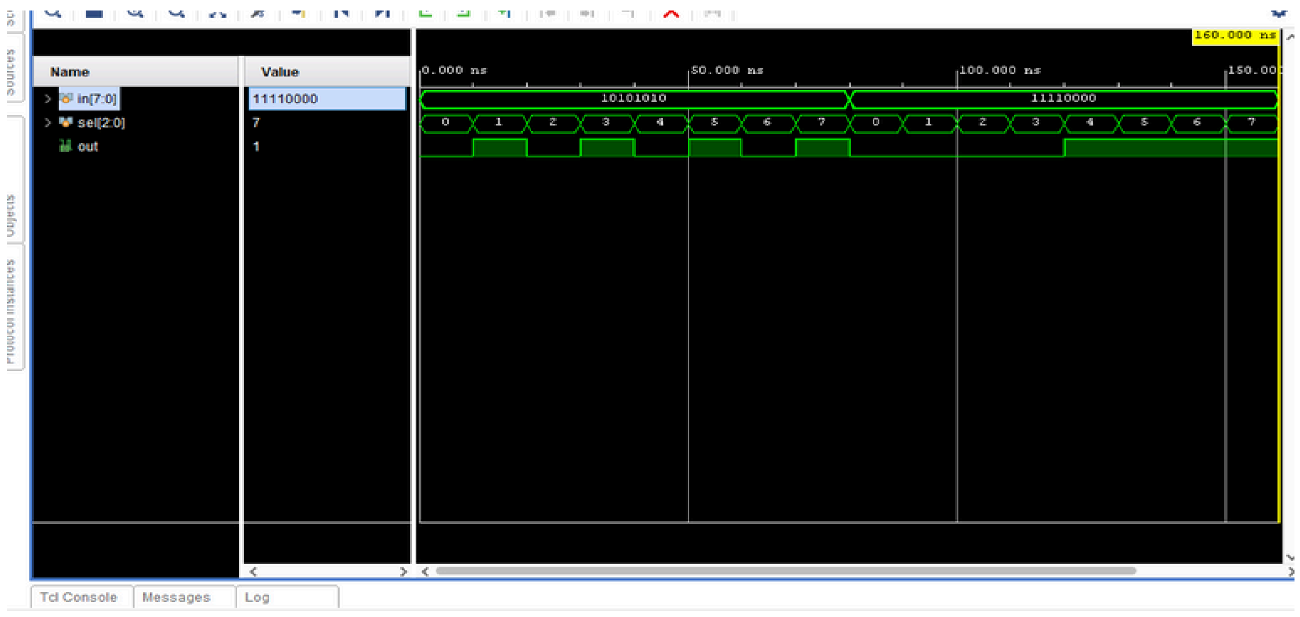
```
module       eight_x_one_mux(in,sel,out);
input [7:0] in; input [2:0] sel;
output reg out;
always @ (*) begin case(sel)
 3'b000: out=in[0];
 3'b001:out=in[1];
3'b010:out=in[2];
3'b011:out=in[3];
3'b100:out=in[4];
3'b101: out=in[5];
3'b110: out=in[6];
3'b111: out=in[7];
 default: out=0;
   endcase
 end endmodule
```

TestBench:

```
module tb3(out);
reg [7:0] in;
reg [2:0] sel;
output out;
eight_x_one_mux dut (in,sel,out);
initial
begin
in=8'b10101010;
 sel=3'b000; #10
sel=3'b001; #10
sel=3'b010; #10
sel=3'b011; #10
sel=3'b100; #10
sel=3'b101; #10
sel=3'b110; #10
sel=3'b111; #10
in=8'b11110000;
sel=3'b000; #10
sel=3'b001; #10
sel=3'b010; #10
sel=3'b011; #10
sel=3'b100; #10
sel=3'b101; #10
sel=3'b110; #10
sel=3'b111; #10
$finish;

end

endmodule
```



Name	Value
in[7:0]	11110000
sel[2:0]	7
out	1

*12.Multiplexer(8*1) using Multiplexer(2*1)*

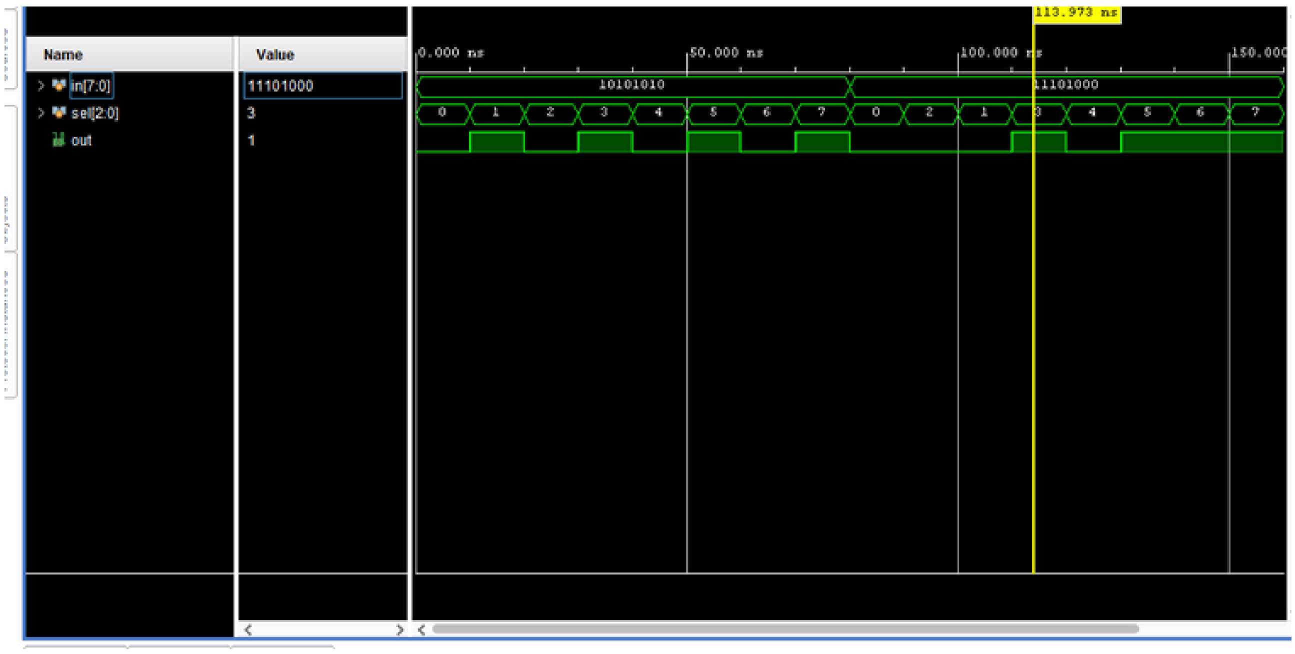
Verilog Code:

```
module x8(in,sel,out);
input [7:0]in;
input [2:0] sel;
output out;
x1_mux m1(in[0],in[1],sel[0],w1);
x1_mux m2(in[2],in[3],sel[0],w2);
x1_mux m3(in[4],in[5],sel[0],w3);
x1_mux m4(in[6],in[7],sel[0],w4);
x1_mux m5(w1,w2,sel[1],w5);
x1_mux m6(w3,w4,sel[1],w6);
x1_mux m7(w5,w6,sel[2],out);
endmodule
```

TestBench:

```
module tb2(out);
reg [7:0] in;
reg [2:0] sel;
output out;
x8 dut (in,sel,out);
initial
begin
in=8'b10101010;
 sel=3'b000; #10
sel=3'b001; #10
sel=3'b010; #10
sel=3'b011; #10
```

```
sel=3'b100; #10
sel=3'b101; #10
sel=3'b110; #10
sel=3'b111; #10
in=8'b11101000;
sel=3'b000; #10
sel=3'b010; #10
sel=3'b001; #10
sel=3'b011; #10
sel=3'b100; #10
sel=3'b101; #10
sel=3'b110; #10
sel=3'b111; #10
$finish;
end
endmodule
```



13.DeMultiplexer(1*8)

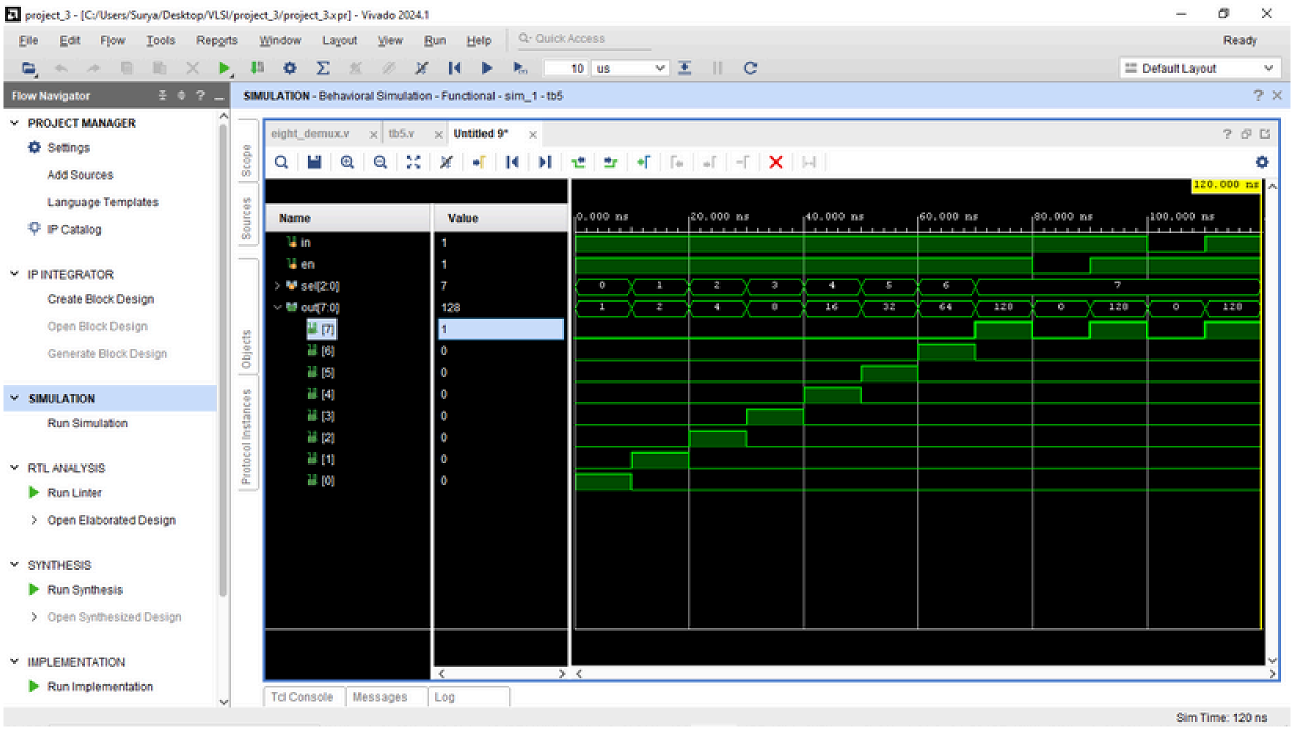
Verilog Code:

```
module eight_demux(in,en,sel,out );
input in,en;
input[2:0]sel;
output reg [7:0]out;
always @(*)
begin
out=8'b00000000;
if(en)
```

```
begin
case(sel)
3'b000: out[0]=in;
3'b001: out[1]=in;
3'b010: out[2]=in;
3'b011: out[3]=in;
3'b100: out[4]=in;
3'b101: out[5]=in;
3'b110: out[6]=in;
3'b111: out[7]=in;
default:out=8'b00000000;
endcase
end
else
out=8'b00000000;
end
endmodule
```

TestBench:

```
module tb5(out );
reg in,en;
reg[2:0]sel;
output [7:0] out;
eight_demux dut(in,en,sel,out);
initial
begin
in=1;en=1;
sel=3'b000; #10
sel=3'b001; #10
sel=3'b010; #10
sel=3'b011; #10
sel=3'b100; #10
sel=3'b101; #10
sel=3'b110; #10
sel=3'b111; #10
en=0; #10
en=1; #10
in=0; #10
in=1; #10
$finish;
end
endmodule
```



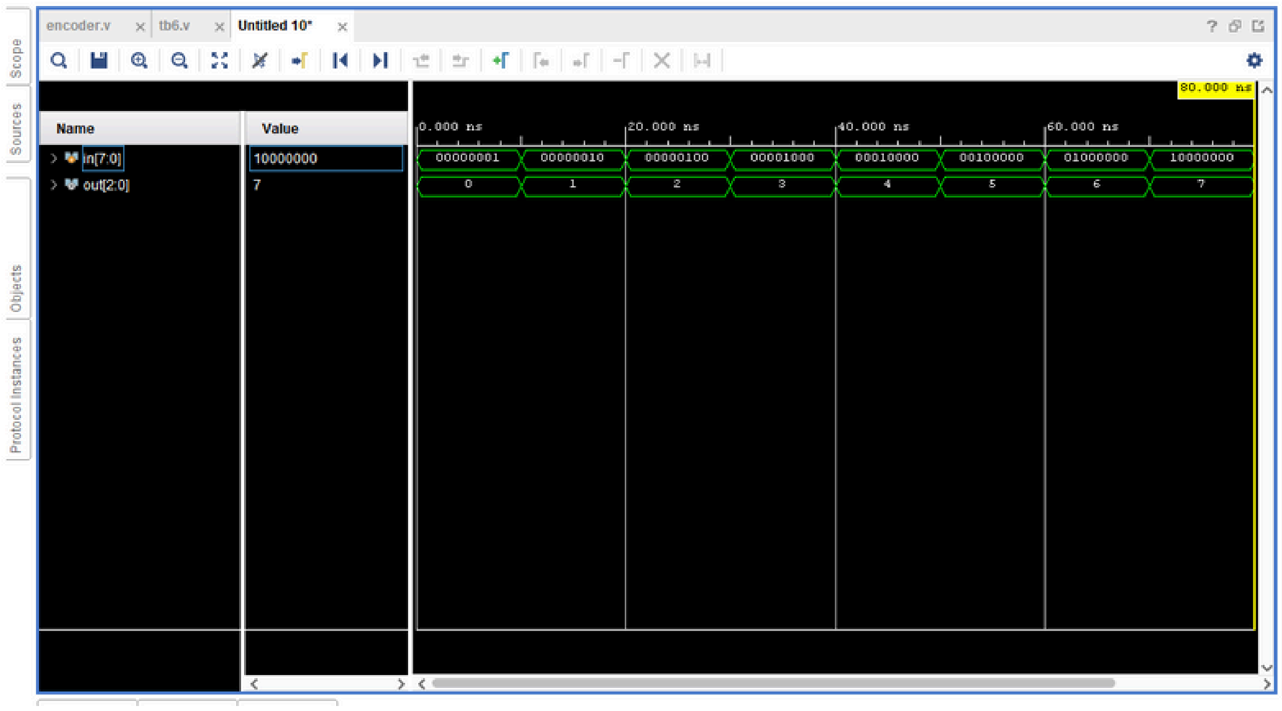
14.Encoder

Verilog Code:

```
module encoder(in,out);
input[7:0] in;
output reg[2:0] out;
always @(*)
begin
out=3'b000;
case(in)
8'b00000000: out=3'b000;
8'b00000001: out=3'b000;
8'b00000010: out=3'b001;
8'b00000100: out=3'b010;
8'b00001000: out=3'b011;
8'b00010000: out=3'b100;
8'b00100000: out=3'b101;
8'b01000000: out=3'b110;
8'b10000000: out=3'b111;
default: out=3'b000;
endcase
end
endmodule
```

TestBench:

```
module   tb6(   );   reg
[7:0]in;    wire    [2:0]
out;              encoder
dut(in,out);       initial
begin
in=8'b00000001;
#10
in=8'b00000010;
#10
in=8'b00000100;
#10
in=8'b00001000;
#10
in=8'b00010000;
#10
in=8'b00100000;
#10
in=8'b01000000;
#10
in=8'b10000000;
#10
$finish;
end
endmodule
```



encoder.v | tb6.v | Untitled 10*

Name	Value
in[7:0]	10000000
out[2:0]	7

```
0.000 ns            20.000 ns           40.000 ns           60.000 ns           80.000 ns
in[7:0]:  00000001  00000010  00000100  00001000  00010000  00100000  01000000  10000000
out[2:0]: 0         1         2         3         4         5         6         7
```

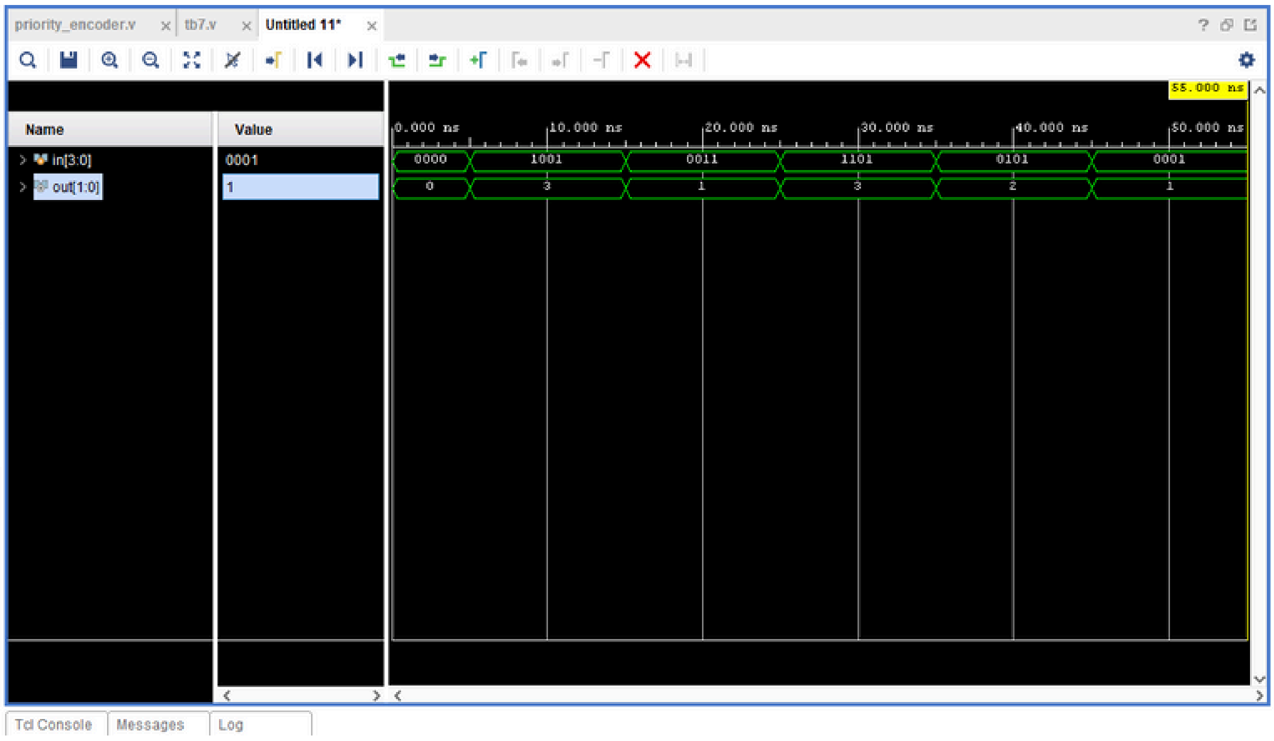
15.Priority Encoder

Verilog Code:

```
module priority_encoder(in,out );
input [3:0] in;
output [1:0] out;
assign out[1]= in[2]|in[3];
assign out[0]=in[3]|((~in[2])&in[0]);
endmodule
```

TestBench:

```
module tb7( );
reg [3:0] in;
wire [1:0] out;
priority_encoder uut(in,out);
initial
begin
in=4'b0000; #5
in=4'b1001; #10
in=4'b0011; #10
in=4'b1101; #10
in=4'b0101; #10
in=4'b0001; #10
$finish;
end
endmodule
```



~Aditya Mittha
~DAY 3