```
==> mux.v <==

module mux (
    output out,
    input [1:0] in,
    input sel
);
  assign out = in[sel];
endmodule


module mux4 (
    output out,
    input [3:0] in,
    input [1:0] sel
);
  wire med1, med2;
  mux U0 (
      .out(med1),
      .in (in[1:0]),
      .sel(sel[0])
  );
  mux U1 (
      .out(med2),
      .in (in[3:2]),
      .sel(sel[0])
  );
  mux U3 (
      .out(out),
      .in ({med2, med1}),
      .sel(sel[1])
  );
endmodule

==> demux.v <==
module demux (
    output [1:0] out,
    input in,
    input sel
);
  assign out = in << sel;
endmodule

module demux4 (
    output [3:0] out,
    input in,
    input [1:0] sel
);
  assign out = in << ((sel[1] * 2) + sel[0]);
endmodule

==> encoder.v <==
module encoder (
    output out,
    input [1:0] in,
    input enable
);
  assign out = enable ? in[1] : 1'bZ;
endmodule

module encoder4 (
    output [1:0] out,
    input [3:0] in,
    input enable
);
  assign out[0] = enable ? in[1] | in[3] : 1'bZ;
  assign out[1] = enable ? ~in[1] & ~in[0] : 1'bZ;
endmodule

==> decoder.v <==
```

```
module decoder (
    output [1:0] out,
    input sel,
    input enable
);
  wire [1:0] med;
  and (med[0], ~sel);
  and (med[1], sel);
  bufif1 (out[0], med[0], enable);
  bufif1 (out[1], med[1], enable);
endmodule

module decoder4 (
    output [3:0] out,
    input [1:0] sel,
    input enable
);
  wire [3:0] med;
  and (med[0], ~sel[0], ~sel[1]);
  and (med[1], sel[0], ~sel[1]);
  and (med[2], ~sel[0], sel[1]);
  and (med[3], sel[0], sel[1]);
  bufif1 (out[0], med[0], enable);
  bufif1 (out[1], med[1], enable);
  bufif1 (out[2], med[2], enable);
  bufif1 (out[3], med[3], enable);
endmodule

==> tb.v <==
module tb ();
  reg [1:0] mux_in;
  reg mux_sel;
  wire mux_out;

  reg [3:0] mux4_in;
  reg [1:0] mux4_sel;
  wire mux4_out;

  reg demux_in;
  reg demux_sel;
  wire [1:0] demux_out;

  reg demux4_in;
  reg [1:0] demux4_sel;
  wire [3:0] demux4_out;

  reg [1:0] enc_in;
  reg enc_enable;
  wire enc_out;

  reg [3:0] enc4_in;
  reg enc4_enable;
  wire [1:0] enc4_out;

  reg dec_inp, dec_ena;
  wire [1:0] dec_out;

  reg [1:0] dec4_inp;
  reg dec4_ena;
  wire [3:0] dec4_out;

  initial begin
    $display("MUX TESTBENCH");
    mux_in  = 2'b00;
    mux_sel = 1'b0;
    $monitor("2:1 MUX :: in = %2b, sel = %b, out = %b", mux_in, mux_sel, mux_out);

    #5{mux_in, mux_sel} = 3'b001;
    #5{mux_in, mux_sel} = 3'b010;
    #5{mux_in, mux_sel} = 3'b011;
    #5{mux_in, mux_sel} = 3'b100;
```

```
    #5{mux_in, mux_sel} = 3'b101;
    #5{mux_in, mux_sel} = 3'b110;
    #5{mux_in, mux_sel} = 3'b111;
    #5 mux4_in = 4'b0000;
    mux4_sel = 2'b00;
    $monitor("4:1 MUX :: in = %4b, sel = %2b, out = %b", mux4_in, mux4_sel, mux4_out);

    #5{mux4_in, mux4_sel} = 6'b000100;
    #5{mux4_in, mux4_sel} = 6'b011101;
    #5{mux4_in, mux4_sel} = 6'b101110;
    #5{mux4_in, mux4_sel} = 6'b010011;
    #5{mux4_in, mux4_sel} = 6'b010100;
    #5{mux4_in, mux4_sel} = 6'b111001;
    #5{mux4_in, mux4_sel} = 6'b011110;
    #5 $display("DEMUX TESTBENCH");

    {demux_in, demux_sel} = 2'b00;
    $monitor("1:2 DEMUX :: in = %b, sel = %b, out = %2b", demux_in, demux_sel, demux_out);

    #5{demux_in, demux_sel} = 2'b01;
    #5{demux_in, demux_sel} = 2'b10;
    #5{demux_in, demux_sel} = 2'b11;
    #5{demux4_in, demux4_sel} = 3'b000;
    $monitor("1:4 DEMUX :: in = %b, sel = %2b, out = %4b", demux4_in, demux4_sel, demux4_out);

    #5{demux4_in, demux4_sel} = 3'b001;
    #5{demux4_in, demux4_sel} = 3'b010;
    #5{demux4_in, demux4_sel} = 3'b011;
    #5{demux4_in, demux4_sel} = 3'b100;
    #5{demux4_in, demux4_sel} = 3'b101;
    #5{demux4_in, demux4_sel} = 3'b110;
    #5{demux4_in, demux4_sel} = 3'b111;
    #5 $display("ENCODER TESTBENCH");

    enc_enable = 1'b0;
    $monitor("2:1 ENCODER :: enable = %b, in = %2b, out = %b", enc_enable, enc_in, enc_out);

    #5{enc_enable, enc_in} = 3'b101;
    #5{enc_enable, enc_in} = 3'b110;
    #5 enc4_enable = 1'b0;
    $monitor("4:1 ENCODER :: enable = %b, in = %4b, out = %2b", enc4_enable, enc4_in, enc4_out);

    #5{enc4_enable, enc4_in} = 5'b10001;
    #5{enc4_enable, enc4_in} = 5'b10010;
    #5{enc4_enable, enc4_in} = 5'b10100;
    #5{enc4_enable, enc4_in} = 5'b11000;
    #5 $display("DECODER TESTBENCH");

    dec_ena = 1'b0;
    $monitor("1:2 DECODER :: enable = %b, in = %b, out = %2b", dec_ena, dec_inp, dec_out);

    #5{dec_ena, dec_inp} = 2'b10;
    #5{dec_ena, dec_inp} = 2'b11;
    #5 dec4_ena = 1'b0;
    $monitor("2:4 DECODER :: enable = %b, in = %2b, out = %4b", dec4_ena, dec4_inp, dec4_out);

    #5{dec4_ena, dec4_inp} = 3'b100;
    #5{dec4_ena, dec4_inp} = 3'b101;
    #5{dec4_ena, dec4_inp} = 3'b110;
    #5{dec4_ena, dec4_inp} = 3'b111;
    #30 $finish(0);
  end

  mux U0 (
      .out(mux_out),
      .in (mux_in),
      .sel(mux_sel)
  );

  mux4 U1 (
      .out(mux4_out),
```

```
      .in (mux4_in),
      .sel(mux4_sel)
  );

  demux U2 (
      .out(demux_out),
      .in (demux_in),
      .sel(demux_sel)
  );

  demux4 U3 (
      .out(demux4_out),
      .in (demux4_in),
      .sel(demux4_sel)
  );

  encoder U4 (
      .out(enc_out),
      .in(enc_in),
      .enable(enc_enable)
  );

  encoder4 U5 (
      .out(enc4_out),
      .in(enc4_in),
      .enable(enc4_enable)
  );

  decoder U6 (
      .out(dec_out),
      .sel(dec_inp),
      .enable(dec_ena)
  );

  decoder4 U7 (
      .out(dec4_out),
      .sel(dec4_inp),
      .enable(dec4_ena)
  );

  initial begin
    $dumpfile("mux.vcd");
    $dumpvars(0, tb);
  end


endmodule

==> output.txt <==
MUX TESTBENCH
VCD info: dumpfile mux.vcd opened for output.
2:1 MUX :: in = 00, sel = 0, out = 0
2:1 MUX :: in = 00, sel = 1, out = 0
2:1 MUX :: in = 01, sel = 0, out = 1
2:1 MUX :: in = 01, sel = 1, out = 0
2:1 MUX :: in = 10, sel = 0, out = 0
2:1 MUX :: in = 10, sel = 1, out = 1
2:1 MUX :: in = 11, sel = 0, out = 1
2:1 MUX :: in = 11, sel = 1, out = 1
4:1 MUX :: in = 0000, sel = 00, out = 0
4:1 MUX :: in = 0001, sel = 00, out = 1
4:1 MUX :: in = 0111, sel = 01, out = 1
4:1 MUX :: in = 1011, sel = 10, out = 0
4:1 MUX :: in = 0100, sel = 11, out = 0
4:1 MUX :: in = 0101, sel = 00, out = 1
4:1 MUX :: in = 1110, sel = 01, out = 1
4:1 MUX :: in = 0111, sel = 10, out = 1
DEMUX TESTBENCH
1:2 DEMUX :: in = 0, sel = 0, out = 00
1:2 DEMUX :: in = 0, sel = 1, out = 00
1:2 DEMUX :: in = 1, sel = 0, out = 01
```

```
1:2 DEMUX :: in = 1, sel = 1, out = 10
1:4 DEMUX :: in = 0, sel = 00, out = 0000
1:4 DEMUX :: in = 0, sel = 01, out = 0000
1:4 DEMUX :: in = 0, sel = 10, out = 0000
1:4 DEMUX :: in = 0, sel = 11, out = 0000
1:4 DEMUX :: in = 1, sel = 00, out = 0001
1:4 DEMUX :: in = 1, sel = 01, out = 0010
1:4 DEMUX :: in = 1, sel = 10, out = 0100
1:4 DEMUX :: in = 1, sel = 11, out = 1000
ENCODER TESTBENCH
2:1 ENCODER :: enable = 0, in = xx, out = z
2:1 ENCODER :: enable = 1, in = 01, out = 0
2:1 ENCODER :: enable = 1, in = 10, out = 1
4:1 ENCODER :: enable = 0, in = xxxx, out = zz
4:1 ENCODER :: enable = 1, in = 0001, out = 00
4:1 ENCODER :: enable = 1, in = 0010, out = 01
4:1 ENCODER :: enable = 1, in = 0100, out = 10
4:1 ENCODER :: enable = 1, in = 1000, out = 11
DECODER TESTBENCH
1:2 DECODER :: enable = 0, in = x, out = zz
1:2 DECODER :: enable = 1, in = 0, out = 01
1:2 DECODER :: enable = 1, in = 1, out = 10
2:4 DECODER :: enable = 0, in = xx, out = zzzz
2:4 DECODER :: enable = 1, in = 00, out = 0001
2:4 DECODER :: enable = 1, in = 01, out = 0010
2:4 DECODER :: enable = 1, in = 10, out = 0100
2:4 DECODER :: enable = 1, in = 11, out = 1000
```