

```

1 ==> mux.v <==
2
3 module mux (
4     output out,
5     input [1:0] in,
6     input sel
7 );
8     assign out = in[sel];
9 endmodule
10
11
12 module mux4 (
13     output out,
14     input [3:0] in,
15     input [1:0] sel
16 );
17     wire med1, med2;
18     mux U0 (
19         .out(med1),
20         .in (in[1:0]),
21         .sel(sel[0])
22     );
23     mux U1 (
24         .out(med2),
25         .in (in[3:2]),
26         .sel(sel[0])
27     );
28     mux U3 (
29         .out(out),
30         .in ({med2, med1}),
31         .sel(sel[1])
32     );
33 endmodule
34
35 ==> demux.v <==
36 module demux (
37     output [1:0] out,
38     input in,
39     input sel
40 );
41     assign out = in << sel;
42 endmodule
43
44 module demux4 (
45     output [3:0] out,
46     input in,
47     input [1:0] sel
48 );
49     assign out = in << ((sel[1] * 2) + sel[0]);
50 endmodule
51
52 ==> encoder.v <==
53 module encoder (
54     output out,
55     input [1:0] in,
56     input enable
57 );
58     assign out = enable ? in[1] : 1'bZ;
59 endmodule
60
61 module encoder4 (
62     output [1:0] out,
63     input [3:0] in,
64     input enable
65 );
66     assign out[0] = enable ? in[1] | in[3] : 1'bZ;
67     assign out[1] = enable ? ~in[1] & ~in[0] : 1'bZ;
68 endmodule
69
70 ==> decoder.v <==

```

```

71 module decoder (
72     output [1:0] out,
73     input sel,
74     input enable
75 );
76     wire [1:0] med;
77     and (med[0], ~sel);
78     and (med[1], sel);
79     bufifl (out[0], med[0], enable);
80     bufifl (out[1], med[1], enable);
81 endmodule
82
83 module decoder4 (
84     output [3:0] out,
85     input [1:0] sel,
86     input enable
87 );
88     wire [3:0] med;
89     and (med[0], ~sel[0], ~sel[1]);
90     and (med[1], sel[0], ~sel[1]);
91     and (med[2], ~sel[0], sel[1]);
92     and (med[3], sel[0], sel[1]);
93     bufifl (out[0], med[0], enable);
94     bufifl (out[1], med[1], enable);
95     bufifl (out[2], med[2], enable);
96     bufifl (out[3], med[3], enable);
97 endmodule
98
99 ==> tb.v <==
100 module tb ();
101     reg [1:0] mux_in;
102     reg mux_sel;
103     wire mux_out;
104
105     reg [3:0] mux4_in;
106     reg [1:0] mux4_sel;
107     wire mux4_out;
108
109     reg demux_in;
110     reg demux_sel;
111     wire [1:0] demux_out;
112
113     reg demux4_in;
114     reg [1:0] demux4_sel;
115     wire [3:0] demux4_out;
116
117     reg [1:0] enc_in;
118     reg enc_enable;
119     wire enc_out;
120
121     reg [3:0] enc4_in;
122     reg enc4_enable;
123     wire [1:0] enc4_out;
124
125     reg dec_inp, dec_ena;
126     wire [1:0] dec_out;
127
128     reg [1:0] dec4_inp;
129     reg dec4_ena;
130     wire [3:0] dec4_out;
131
132     initial begin
133         $display("MUX TESTBENCH");
134
135         mux_in = 2'b00;
136         mux_sel = 1'b0;
137         $monitor("2:1 MUX :: in = %2b, sel = %b, out = %b", mux_in, mux_sel, mux_out);
138
139         #5{mux_in, mux_sel} = 3'b001;
140         #5{mux_in, mux_sel} = 3'b010;
141         #5{mux_in, mux_sel} = 3'b011;
142         #5{mux_in, mux_sel} = 3'b100;

```

```

142 #5{mux_in, mux_sel} = 3'b101;
143 #5{mux_in, mux_sel} = 3'b110;
144 #5{mux_in, mux_sel} = 3'b111;
145 #5 mux4_in = 4'b0000;
146 mux4_sel = 2'b00;
147 $monitor("4:1 MUX :: in = %4b, sel = %2b, out = %b", mux4_in, mux4_sel, mux4_out);
148
149 #5{mux4_in, mux4_sel} = 6'b000100;
150 #5{mux4_in, mux4_sel} = 6'b011101;
151 #5{mux4_in, mux4_sel} = 6'b101110;
152 #5{mux4_in, mux4_sel} = 6'b010011;
153 #5{mux4_in, mux4_sel} = 6'b010100;
154 #5{mux4_in, mux4_sel} = 6'b111001;
155 #5{mux4_in, mux4_sel} = 6'b011110;
156 #5 $display("DEMUX TESTBENCH");
157
158 {demux_in, demux_sel} = 2'b00;
159 $monitor("1:2 DEMUX :: in = %b, sel = %b, out = %2b", demux_in, demux_sel, demux_out);
160
161 #5{demux_in, demux_sel} = 2'b01;
162 #5{demux_in, demux_sel} = 2'b10;
163 #5{demux_in, demux_sel} = 2'b11;
164 #5{demux4_in, demux4_sel} = 3'b000;
165 $monitor("1:4 DEMUX :: in = %b, sel = %2b, out = %4b", demux4_in, demux4_sel, demux4_out);
166
167 #5{demux4_in, demux4_sel} = 3'b001;
168 #5{demux4_in, demux4_sel} = 3'b010;
169 #5{demux4_in, demux4_sel} = 3'b011;
170 #5{demux4_in, demux4_sel} = 3'b100;
171 #5{demux4_in, demux4_sel} = 3'b101;
172 #5{demux4_in, demux4_sel} = 3'b110;
173 #5{demux4_in, demux4_sel} = 3'b111;
174 #5 $display("ENCODER TESTBENCH");
175
176 enc_enable = 1'b0;
177 $monitor("2:1 ENCODER :: enable = %b, in = %2b, out = %b", enc_enable, enc_in, enc_out);
178
179 #5{enc_enable, enc_in} = 3'b101;
180 #5{enc_enable, enc_in} = 3'b110;
181 #5 enc4_enable = 1'b0;
182 $monitor("4:1 ENCODER :: enable = %b, in = %4b, out = %2b", enc4_enable, enc4_in, enc4_out);
183
184 #5{enc4_enable, enc4_in} = 5'b10001;
185 #5{enc4_enable, enc4_in} = 5'b10010;
186 #5{enc4_enable, enc4_in} = 5'b10100;
187 #5{enc4_enable, enc4_in} = 5'b11000;
188 #5 $display("DECODER TESTBENCH");
189
190 dec_ena = 1'b0;
191 $monitor("1:2 DECODER :: enable = %b, in = %b, out = %2b", dec_ena, dec_in, dec_out);
192
193 #5{dec_ena, dec_in} = 2'b10;
194 #5{dec_ena, dec_in} = 2'b11;
195 #5 dec4_ena = 1'b0;
196 $monitor("2:4 DECODER :: enable = %b, in = %2b, out = %4b", dec4_ena, dec4_in, dec4_out);
197
198 #5{dec4_ena, dec4_in} = 3'b100;
199 #5{dec4_ena, dec4_in} = 3'b101;
200 #5{dec4_ena, dec4_in} = 3'b110;
201
202 #5{dec4_ena, dec4_in} = 3'b111;
202 #30 $finish(0);
203 end
204
205 mux U0 (
206     .out(mux_out),
207     .in (mux_in),
208     .sel(mux_sel)
209 );
210
211 mux4 U1 (
212     .out(mux4_out),

```

```

213         .in (mux4_in),
214         .sel(mux4_sel)
215     );
216
217     demux U2 (
218         .out(demux_out),
219         .in (demux_in),
220         .sel(demux_sel)
221     );
222
223     demux4 U3 (
224         .out(demux4_out),
225         .in (demux4_in),
226         .sel(demux4_sel)
227     );
228
229     encoder U4 (
230         .out(enc_out),
231         .in(enc_in),
232         .enable(enc_enable)
233     );
234
235     encoder4 U5 (
236         .out(enc4_out),
237         .in(enc4_in),
238         .enable(enc4_enable)
239     );
240
241     decoder U6 (
242         .out(dec_out),
243         .sel(dec_inp),
244         .enable(dec_ena)
245     );
246
247     decoder4 U7 (
248         .out(dec4_out),
249         .sel(dec4_inp),
250         .enable(dec4_ena)
251     );
252
253     initial begin
254         $dumpfile("mux.vcd");
255         $dumpvars(0, tb);
256     end
257
258
259 endmodule
260
261 ==> output.txt <==
262 MUX TESTBENCH
263 VCD info: dumpfile mux.vcd opened for output.
264 2:1 MUX :: in = 00, sel = 0, out = 0
265 2:1 MUX :: in = 00, sel = 1, out = 0
266 2:1 MUX :: in = 01, sel = 0, out = 1
267 2:1 MUX :: in = 01, sel = 1, out = 0
268
269 2:1 MUX :: in = 10, sel = 0, out = 0
270 2:1 MUX :: in = 10, sel = 1, out = 1
271 2:1 MUX :: in = 11, sel = 0, out = 1
272 2:1 MUX :: in = 11, sel = 1, out = 1
273
274 4:1 MUX :: in = 0000, sel = 00, out = 0
275 4:1 MUX :: in = 0001, sel = 00, out = 1
276 4:1 MUX :: in = 0111, sel = 01, out = 1
277 4:1 MUX :: in = 1011, sel = 10, out = 0
278 4:1 MUX :: in = 0100, sel = 11, out = 0
279 4:1 MUX :: in = 0101, sel = 00, out = 1
280 4:1 MUX :: in = 1110, sel = 01, out = 1
281 4:1 MUX :: in = 0111, sel = 10, out = 1
282
283 DEMUX TESTBENCH
284 1:2 DEMUX :: in = 0, sel = 0, out = 00
285 1:2 DEMUX :: in = 0, sel = 1, out = 00
286 1:2 DEMUX :: in = 1, sel = 0, out = 01

```

```
284 1:2 DEMUX :: in = 1, sel = 1, out = 10
285 1:4 DEMUX :: in = 0, sel = 00, out = 0000
286 1:4 DEMUX :: in = 0, sel = 01, out = 0000
287 1:4 DEMUX :: in = 0, sel = 10, out = 0000
288 1:4 DEMUX :: in = 0, sel = 11, out = 0000
289 1:4 DEMUX :: in = 1, sel = 00, out = 0001
290 1:4 DEMUX :: in = 1, sel = 01, out = 0010
291 1:4 DEMUX :: in = 1, sel = 10, out = 0100
292 1:4 DEMUX :: in = 1, sel = 11, out = 1000
293 ENCODER TESTBENCH
294 2:1 ENCODER :: enable = 0, in = xx, out = z
295 2:1 ENCODER :: enable = 1, in = 01, out = 0
296 2:1 ENCODER :: enable = 1, in = 10, out = 1
297 4:1 ENCODER :: enable = 0, in = xxxx, out = zz
298 4:1 ENCODER :: enable = 1, in = 0001, out = 00
299 4:1 ENCODER :: enable = 1, in = 0010, out = 01
300 4:1 ENCODER :: enable = 1, in = 0100, out = 10
301 4:1 ENCODER :: enable = 1, in = 1000, out = 11
302 DECODER TESTBENCH
303 1:2 DECODER :: enable = 0, in = x, out = zz
304 1:2 DECODER :: enable = 1, in = 0, out = 01
305 1:2 DECODER :: enable = 1, in = 1, out = 10
306 2:4 DECODER :: enable = 0, in = xx, out = zzzz
307 2:4 DECODER :: enable = 1, in = 00, out = 0001
308 2:4 DECODER :: enable = 1, in = 01, out = 0010
309 2:4 DECODER :: enable = 1, in = 10, out = 0100
310 2:4 DECODER :: enable = 1, in = 11, out = 1000
```