

```

1: //-----
2: // dir_ctrl.v
3: // acassidy@jhu.edu
4: // 2010/03/31
5: //-----
6: //
7: // direction map:
8: // 00 : left
9: // 01 : up
10: // 10 : right
11: // 11 : down
12: //-----
13:
14:
15: module dir_ctrl (
16:     input clk,
17:     input rst,
18:     input game_over,
19:
20:     input L_i,
21:     input R_i,
22:
23:     output reg [1:0] dir
24: );
25:
26:     wire [1:0] LR;
27:
28:     // map input
29:     assign LR[0] = R_i;
30:     assign LR[1] = L_i;
31:
32:
33: //-----
34: // change direction based on left/right input commands (signals)
35: //
36: always @(posedge clk) begin
37:     if ((rst == 1'b1) || (game_over == 1'b1))
38:         dir <= 2'b00;
39:     else
40:         case (LR[1:0])
41:             // no turns
42:             2'b00 : dir <= dir;
43:
44:             // turn right
45:             2'b01 : dir <= dir - 1;
46:
47:             // turn left
48:             2'b10 : dir <= dir + 1;
49:
50:             // no turns
51:             2'b11 : dir <= dir;
52:         endcase
53: end // always
54:
55:
56: //-----
57: endmodule
58: //-----
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:

```

```

69: //-----
70: // xy_ctrl.v
71: // acassidy@jhu.edu
72: // 2010/03/31
73: //-----
74:
75: module xy_ctrl (
76:     input clk,
77:     input rst,
78:
79:     input [1:0] dir,
80:     output reg [7:0] X,
81:     output reg [7:0] Y,
82:     output reg game_over
83: );
84:
85:     wire [7:0] speed;
86:
87:     // assign 'speed' a constant value
88:     assign speed = 8'b00000001;
89:
90:
91: //-----
92: // update X-Y coordinates based on direction
93: //
94: always @(posedge clk) begin
95:     if ((rst == 1'b1) || (game_over == 1'b1)) begin
96:         X <= 8'b10000000;
97:         Y <= 8'b10000000;
98:     end else
99:         case (dir[1:0])
100:             // going right
101:             2'b00 : X <= X + speed;
102:
103:             // going up
104:             2'b01 : Y <= Y + speed;
105:
106:             // going left
107:             2'b10 : X <= X - speed;
108:
109:             // going down
110:             2'b11 : Y <= Y - speed;
111:         endcase
112: end // always
113:
114:
115: //-----
116: // bounds check -- end game
117: //
118: always @(posedge clk) begin
119:     if (rst == 1'b1)
120:         game_over <= 1'b0;
121:     else
122:
123:         // if X or Y is > 252 or < 2, end game
124:         if ((X > 8'b11111100) || (X < 8'b00000010) ||
125:             (Y > 8'b11111100) || (Y < 8'b00000010))
126:             game_over <= 1'b1;
127:
128: end // always
129:
130:
131: //-----
132: endmodule
133: //-----
134:
135:
136:

```

```
137: //-----
138: // snake.v
139: // acassidy@jhu.edu
140: // 2010/03/31
141: //-----
142:
143:
144: module snake (
145:     // I/O ports
146:     input clk,
147:     input rst,
148:
149:     input L_i,
150:     input R_i,
151:
152:     output [7:0] X_o,
153:     output [7:0] Y_o,
154:     output game_over
155: );
156:
157:     // internal signals
158:     wire [1:0] dir;
159:
160:
161: //-----
162: // instantiate direction control block
163: dir_ctrl dir_ctrl_blk (
164:     .clk(clk),
165:     .rst(rst),
166:     .game_over(game_over),
167:
168:     .L_i(L_i),
169:     .R_i(R_i),
170:
171:     .dir(dir)
172: );
173:
174:
175: // instantiate XY control block
176: xy_ctrl xy_ctrl_blk (
177:     .clk(clk),
178:     .rst(rst),
179:
180:     .dir(dir),
181:     .X(X_o),
182:     .Y(Y_o),
183:     .game_over(game_over)
184: );
185:
186:
187: //-----
188: endmodule
189: //-----
190:
```