```
1   //*********************************TOP
    MODULE*************************************************
    *********//
2
3   //-----variable
    names::-------------------------------------------
    ----------------------------------//
4   //open_fsm and close_fsm are signals to open and
    close coming from the central game control
5   //limit_jawOpen: limit switch that indicates if jaw
    is open when depressed (similar for limit_jawClose)
6   //jaw_is_open and jaw_is_closed are "handshakes"
    back to the main game control
7
8   //---potential
    bugs:---------------------------------------------
    ----------------------------------//
9   //counter that is used to slow the motor
10  //signal may not get thru to the FPGA since there is
    no initial value for countVal
11
12  module motorFSM(clock, resetn, open_fsm, close_fsm,
    limit_jawOpen, limit_jawClose, jaw_is_open,
    jaw_is_closed, pin, current);
13
14      input clock, resetn, open_fsm, close_fsm,
    limit_jawOpen, limit_jawClose;
15      output jaw_is_open, jaw_is_closed;
16      output [3:0]pin;
17
18      wire openJaw, closeJaw, counterClock;
19      output [3:0] current;
20
21      //everytime the countDone == 1 --> rising edge
    of the counterClock
22      //counterClock used to slow down the motor
23      countdown_motor countdown_motor(.clock(clock), .
    countDone(counterClock));
24
25      controlMotor controlMotor1(.clock(clock),
26                                 .resetn(resetn),
27                                  .open_fsm(open_fsm),
28                                  .close_fsm(close_fsm),
29                                  .limit_jawOpen(
    limit_jawOpen),
30                                  .limit_jawClose(
    limit_jawClose),
```

```
31                                          .openJaw(openJaw),
32                                          .closeJaw(closeJaw),
33                                          .jaw_is_open(
    jaw_is_open),
34                                          .jaw_is_closed(
    jaw_is_closed),
35                                          .current(current));
36
37
38      datapathMotor datapathMotor1(.counterClock(
    counterClock),
39                                          .resetn(resetn),
40                                          .openJaw(openJaw),
41                                          .closeJaw(closeJaw),
42                                          .pin(pin));
43  endmodule
44
45
46
47
48
49
50
51
52
53
54
55
56
57  //reset will be connected to the reset that exists
    in all of the modules
58  //reset doesn't do anything...mostly for modelsim
    purposes
59
60  module controlMotor(clock, resetn, open_fsm,
    close_fsm, limit_jawOpen, limit_jawClose, openJaw,
    closeJaw, jaw_is_open, jaw_is_closed, current) ;
61
62      input clock, resetn, open_fsm, close_fsm;
    //open_fsm and close_fsm: signals from central game
    control that tells this module what to do
63      input limit_jawOpen, limit_jawClose; //limit
    switches to detect if jaw is open or closed
64
65      output reg openJaw, closeJaw; //to control the
    motor
66      output reg jaw_is_closed, jaw_is_open;
```

```
    //"handshake" for the central game control
67
68      //state registers
69      output reg[3:0] current;
70      reg [3:0] next;
71
72      //hold state is to allow time for the central
    game control to check game status (aka if hand is
    caught or not)
73      //should recieve signal from central game
    control to open back up the jaws
74      localparam READY = 4'd0,
75                 CLOSE = 4'd1,
76                 HOLD  = 4'd2,
77                 OPEN  = 4'd3;
78
79      //state table
80      always @(*)
81      begin: state_table
82        case(current)
83          READY: begin
84            if (close_fsm) next = CLOSE;  //goes to
    CLOSE state if close signal recieved
85            else if(open_fsm) next = OPEN; //goes to
    OPEN state if open signal recieved
86          end
87          CLOSE: begin
88            if (limit_jawClose) next = READY;
    //returns to READY if when jaw is closed
89            else if(open_fsm) next = OPEN;   //goes
    to OPEN if open signal recieved
90          end
91          OPEN: begin
92            if (limit_jawOpen) next = READY;
    //returns to READY if when jaw is opened
93          end
94          default: next = READY;
95        endcase
96      end
97
98      //datapath controls
99      always @ (*)
100     begin: enable_signals
101       closeJaw = 'd0;
102       openJaw = 'd0;
103       jaw_is_closed = 'd0;
104       jaw_is_open = 'd0;
```

```
105
106             case (current)
107                 READY: begin
108                     closeJaw = 'd0;
109                     openJaw = 'd0;
110                     jaw_is_closed = 'd0;
111                     jaw_is_open = 'd1;
112                 end
113
114                 CLOSE: begin
115                     closeJaw = 'd1;
116                     openJaw = 'd0;
117                     jaw_is_closed = 'd0;
118                     jaw_is_open = 'd0;
119                 end
120
121                 HOLD: begin
122                     closeJaw = 'd0;
123                     openJaw = 'd0;
124                     jaw_is_closed = 'd1;
125                     jaw_is_open = 'd0;
126                 end
127
128                 OPEN: begin
129                     closeJaw = 'd0;
130                     openJaw = 'd1;
131                     jaw_is_closed = 'd0;
132                     jaw_is_open = 'd0;
133                 end
134             endcase
135         end
136
137         //current state registers
138         always @ (posedge clock)
139             begin
140                 if (!resetn) current <=READY;
141                 else current <= next;
142             end
143
144     endmodule
145
146
147     //there will be a feedback loop to see which pins
        were last triggered
148     module datapathMotor(counterClock, resetn, openJaw,
        closeJaw, pin);
149
```

```
150         input counterClock, resetn, openJaw, closeJaw;
151         output reg [3:0]pin;
152 
153         reg [3:0]even_odd;
154 
155         // //output to pins also stored in wires
156         // wire [3:0]p;
157 
158         always @ (posedge counterClock) begin
159             even_odd <= even_odd + 1;
160 
161             if (!resetn) begin
162                 pin[3:0] <= 'd0;
163             end
164 
165 
166 
167             else if (closeJaw) begin
168                 case (pin[3:0])
169                     4'b1100: pin = 4'b0110;
170                     4'b0110: pin = 4'b0011;
171                     4'b0011: pin = 4'b1001;
172                     4'b1001: pin = 4'b1100;
173                     default: pin = 4'b1100;
174                 endcase
175             end
176 
177 
178 
179             else if (openJaw) begin
180                 if(even_odd == 0 || even_odd == 3 ||
    even_odd == 6 || even_odd == 9 || even_odd == 12)
    begin
181                     case (pin[3:0])
182                         4'b0011: pin = 4'b0110;
183                         4'b0110: pin = 4'b1100;
184                         4'b1100: pin = 4'b1001;
185                         4'b1001: pin = 4'b0011;
186                         default: pin = 4'b0011;
187                     endcase
188                end
189             end
190 
191             //when the motor shouldnt be stimulated
192             else begin
193                 case (pin[3:0])
194                     4'b1100: pin = 4'b0000;
```

```
195                     4'b0110: pin = 4'b0000;
196                     4'b0011: pin = 4'b0000;
197                     4'b1001: pin = 4'b0000;
198                     default: pin = 4'b0000;
199                 endcase
200             end
201
202         end
203     endmodule
204
205     //motor clock....values need to be changed
206     //potential bugs in here...not sure if resetn is
        needed? **********************
207     module countdown_motor(clock, countDone);
208         input clock;
209         output reg countDone;
210
211         reg [32:0]countVal;
212
213         always @(posedge clock) begin
214             if (countVal == 'd0) begin
215                 countVal <= 'd100000; //change values as
        needed
216                 countDone <= 1;
217             end
218
219             else if(countVal != 'd0) begin
220                countVal <= countVal - 1;
221                countDone <= 0;
222             end
223         end
224     endmodule
225
```