```
/*
   Ben Davis
   1/11/23
   EE 371
   Lab 1

   This module counts up to 25 for the parking lot meter.
   It takes in three inputs, a reset, a plus one (inc), and
   a minus one (dec). It returns the number of cars in the lot
   as a 5 bit output (out).

*/

module counter_25 (clk, rst, inc, dec, out);

   input logic clk, rst, inc, dec; //clock, reset, increment, decrement
   output logic [4:0] out; //number of cars inside lot

   enum {s0, s1, s2, s3, s4, s5, s6, s7, s8 //states to count the # of cars
         , s9, s10, s11, s12, s13, s14, s15
         , s16, s17, s18, s19, s20, s21, s22
         , s23, s24, s25} ps, ns;

   //always ff block to reset the state to zero
   //or to move to the next state if need be
   always_ff @(posedge clk) begin
      if(rst) begin
         ps <= s0;
      end else begin
         ps <= ns;
      end
   end

   //assigning each bit. each bit is one in specific states
   // example: first bit is only true in odd states
   assign out[0] = ((ps==s1)|(ps==s3)|(ps==s5)|(ps==s7)|(ps==s9)|
                    (ps==s11)|(ps==s13)|(ps==s15)|(ps==s17)|(ps==s19)|
                    (ps==s21)|(ps==s23)|(ps==s25));

   assign out[1] = ((ps==s2)|(ps==s3)|(ps==s6)|(ps==s7)|(ps==s10)|(ps==s11)|
                    (ps==s14)|(ps==s15)|(ps==s18)|(ps==s19)|(ps==s22)|(ps==s23));

   assign out[2] = ((ps==s4)|(ps==s5)|(ps==s6)|(ps==s7)|(ps==s12)|(ps==s13)|(ps==s14)|
                    (ps==s15)|(ps==s20)|(ps==s21)|(ps==s22)|(ps==s23));

   assign out[3] = ((ps==s8)|(ps==s9)|(ps==s10)|(ps==s11)|(ps==s12)|(ps==s13)|
                    (ps==s14)|(ps==s15)|(ps==s24)|(ps==s25));

   assign out[4] = ((ps==s16)|(ps==s17)|(ps==s18)|(ps==s19)|(ps==s20)|(ps==s21)|
                    (ps==s22)|(ps==s23)|(ps==s24)|(ps==s25));

   //the block to tell each state if it goes to the
   //next or previous state
   always_comb begin
      case(ps)

         s0: if(inc) ns <= s1;
               else ns <= s0;

         s1: if(inc) ns <= s2;
               else if(dec) ns <= s0;
               else ns <= s1;

         s2: if(inc) ns <= s3;
               else if(dec) ns <= s1;
               else ns <= s2;

         s3: if(inc) ns <= s4;
               else if(dec) ns <= s2;
               else ns <= s3;

         s4: if(inc) ns <= s5;
               else if(dec) ns <= s3;
```

```
74                  else ns <= s4;
75
76              s5: if(inc) ns <= s6;
77                  else if(dec) ns <= s4;
78                  else ns <= s5;
79
80              s6: if(inc) ns <= s7;
81                  else if(dec) ns <= s5;
82                  else ns <= s6;
83
84              s7: if(inc) ns <= s8;
85                  else if(dec) ns <= s6;
86                  else ns <= s7;
87
88              s8: if(inc) ns <= s9;
89                  else if(dec) ns <= s7;
90                  else ns <= s8;
91
92              s9: if(inc) ns <= s10;
93                  else if(dec) ns <= s8;
94                  else ns <= s9;
95
96              s10: if(inc) ns <= s11;
97                  else if(dec) ns <= s9;
98                  else ns <= s10;
99
100             s11: if(inc) ns <= s12;
101                 else if(dec) ns <= s10;
102                 else ns <= s11;
103
104             s12: if(inc) ns <= s13;
105                 else if(dec) ns <= s11;
106                 else ns <= s12;
107
108             s13: if(inc) ns <= s14;
109                 else if(dec) ns <= s12;
110                 else ns <= s13;
111
112             s14: if(inc) ns <= s15;
113                 else if(dec) ns <= s13;
114                 else ns <= s14;
115
116             s15: if(inc) ns <= s16;
117                 else if(dec) ns <= s14;
118                 else ns <= s15;
119
120             s16: if(inc) ns <= s17;
121                 else if(dec) ns <= s15;
122                 else ns <= s16;
123
124             s17: if(inc) ns <= s18;
125                 else if(dec) ns <= s16;
126                 else ns <= s17;
127
128             s18: if(inc) ns <= s19;
129                 else if(dec) ns <= s17;
130                 else ns <= s18;
131
132             s19: if(inc) ns <= s20;
133                 else if(dec) ns <= s18;
134                 else ns <= s19;
135
136             s20: if(inc) ns <= s21;
137                 else if(dec) ns <= s19;
138                 else ns <= s20;
139
140             s21: if(inc) ns <= s22;
141                 else if(dec) ns <= s20;
142                 else ns <= s21;
143
144             s22: if(inc) ns <= s23;
145                 else if(dec) ns <= s21;
146                 else ns <= s22;
```

```
147
148             s23: if(inc) ns <= s24;
149                  else if(dec) ns <= s22;
150                  else ns <= s23;
151
152             s24: if(inc) ns <= s25;
153                  else if(dec) ns <= s23;
154                  else ns <= s24;
155
156             s25: if(dec) ns <= s24;
157                  else ns <= s25;
158
159          endcase
160       end
161    endmodule
162
163    module counter_25_testbench();
164
165       logic clk, rst, inc, dec; //repeating logic variables
166       logic [4:0] out;
167
168       //test counter_25 module
169       counter_25 dut (.clk, .rst, .inc, .dec, .out);
170
171       // clock setup
172       parameter clock_period = 100;
173
174       initial begin
175          clk <= 0;
176          forever #(clock_period /2) clk = ~clk;
177       end // of clock setup
178
179       //an instance where it counts up to five with a
180       //space after the first increment, and then
181       //decrements twice
182       initial begin
183          rst <= 0; inc <= 1; dec <= 0; @(posedge clk);
184          rst <= 0; inc <= 0; dec <= 0; @(posedge clk);
185          rst <= 0; inc <= 1; dec <= 0; @(posedge clk);
186          rst <= 0; inc <= 1; dec <= 0; @(posedge clk);
187          rst <= 0; inc <= 1; dec <= 0; @(posedge clk);
188          rst <= 0; inc <= 1; dec <= 0; @(posedge clk);
189          rst <= 0; inc <= 0; dec <= 1; @(posedge clk);
190          rst <= 0; inc <= 0; dec <= 0; @(posedge clk);
191          $stop;
192       end
193    endmodule
```