```systemverilog
// Alan Li
// 01/15/2022
// EE 371
// Lab #1

// counter takes 1-bit enter and exit as inputs and return 5-bit cout as output.
// The module functions as its name. When there is an enter signal, the cout increse by 1. When there is an exit signal, the cout decrease by 1.
// The range for the counter is from 0 to 25.

module counter(clk, reset, enter, exit, cout);

   input logic clk, reset, enter, exit;
   output logic [4:0] cout;
   logic [4:0] ps, ns;

   // Each decimal from 0 to 25 has been assigned with a 5-bit binary number.
   parameter [4:0] zero = 5'b0,
      one = 5'b1,
      two = 5'b10,
      three = 5'b11,
      four = 5'b100,
      five = 5'b101,
      six = 5'b110,
      seven = 5'b111,
      eight = 5'b1000,
      nine = 5'b1001,
      ten = 5'b1010,
      eleven = 5'b1011,
      twelve = 5'b1100,
      thirteen = 5'b1101,
      fourteen = 5'b1110,
      fifteen = 5'b1111,
      sixteen = 5'b10000,
      seventeen = 5'b10001,
      eighteen = 5'b10010,
      nineteen = 5'b10011,
      twenty = 5'b10100,
      twentyone = 5'b10101,
      twentytwo = 5'b10110,
      twentythree = 5'b10111,
      twentyfour = 5'b11000,
      twentyfive = 5'b11001;

   // 25 states for the counter. Each state represent a different number.
   // When there is a enter signal, the counter goes to next state which the number increase by one and vice versa.
   // At state 0, if there is another exit signal(which should not happen in reality), the state will stay at zero.
   // At state 25, if there is another enter signal, the state will stay at 25.
      case(ps)
         zero: if(enter)        ns = one;
               else             ns = zero;
         one: if(enter)         ns = two;
```

```
              else if(exit)     ns = zero;
              else              ns = one;
        two: if(enter)          ns = three;
              else if(exit)     ns = one;
              else              ns = two;
        three:if(enter)         ns = four;
              else if(exit)     ns = two;
              else              ns = three;
        four: if(enter)         ns = five;
              else if(exit)     ns = three;
              else              ns = four;

        /*
        five: if(exit)          ns = four;
              else              ns = five; // for demo purpose
        */


        five: if(enter)         ns = six;
              else if(exit)     ns = four;
              else              ns = five; // for lab design purpose

        six: if(enter)          ns = seven;
              else if(exit)     ns = five;
              else              ns = six;
        seven: if(enter)        ns = eight;
              else if(exit)     ns = six;
              else              ns = seven;
        eight:if(enter)         ns = nine;
              else if(exit)     ns = seven;
              else              ns = eight;
        nine: if(enter)         ns = ten;
              else if(exit)     ns = eight;
              else              ns = nine;
        ten:  if(enter)         ns = eleven;
              else if(exit)     ns = nine;
              else              ns = ten;
        eleven:if(enter)        ns = twelve;
              else if(exit)     ns = ten;
              else              ns = eleven;
        twelve:if(enter)        ns = thirteen;
              else if(exit)     ns = eleven;
              else              ns = twelve;
        thirteen:if(enter)      ns = fourteen;
              else if(exit)     ns = twelve;
              else              ns = thirteen;
        fourteen: if(enter)     ns = fifteen;
              else if(exit)     ns = thirteen;
              else              ns = fourteen;
        fifteen:if(enter)       ns = sixteen;
              else if(exit)     ns = fourteen;
              else              ns = fifteen;
        sixteen: if(enter)      ns = seventeen;
              else if(exit)     ns = fifteen;
              else              ns = sixteen;
        seventeen:if(enter)     ns = eighteen;
              else if(exit)     ns = sixteen;
              else              ns = seventeen;
```

```
         eighteen:if(enter)      ns = nineteen;
               else if(exit)     ns = seventeen;
               else              ns = eighteen;
         nineteen:if(enter)      ns = twenty;
               else if(exit)     ns = eighteen;
               else              ns = nineteen;
         twenty:if(enter)        ns = twentyone;
               else if(exit)     ns = nineteen;
               else              ns = twenty;
         twentyone:if(enter)     ns = twentytwo;
               else if(exit)     ns = twenty;
               else              ns = twentyone;
         twentytwo:if(enter)     ns = twentythree;
               else if(exit)     ns = twentyone;
               else              ns = twentytwo;
         twentythree:if(enter)   ns = twentyfour;
               else if(exit)     ns = twentytwo;
               else              ns = twentythree;
         twentyfour:if(enter)    ns = twentyfive;
               else if(exit)     ns = twentythree;
               else              ns = twentyfour;
         twentyfive:if(exit)     ns = twentyfour;
               else              ns = twentyfive;
         endcase
   end

   // When reset is pressed, the counter will reset to state 0.
   always @(posedge clk) begin
      if(reset) begin
         ps <= zero;
      end
      else begin
         cout <= ps;
         ps <= ns;
      end
   end
endmodule

// counter_testbench tests all expected behavior that the parking lot occupancy counter system in the lab may encounter
module counter_testbench();
   logic clk, reset, enter, exit;
   logic [4:0]cout;

   counter dut (.clk(clk), .reset(reset), .enter(enter), .exit(exit), .cout(cout));

   parameter CLOCK_PERIOD = 100;

   initial begin
      clk <= 0;
      forever #(CLOCK_PERIOD/2) clk <= ~clk;
   end

   // 30 cars enters, the counter will reach 25 and stay there
   // 30 cars exit(for simulation), the counter will reach 0 and stay there
   initial begin
```

```
      reset <= 1;            @(posedge clk);
      reset <= 0;            @(posedge clk);
      enter <= 1; exit <= 0; repeat(30) @(posedge clk);
      enter <= 0; exit <= 1; repeat(30) @(posedge clk);
      $stop;
   end
endmodule
```