```systemverilog
/*
   Ben Davis
   1/11/23
   EE 371
   Lab 2, Task 2

   This module counts up to 15 for the 32x4 RAM.
   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 4 bit output (out).

*/

module counter_31 (clk, rst, out);

   input logic clk, rst; //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, s26, s27, s28, s29
         , s30, s31} 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


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

         s0: begin out <= 5'b00000; ns <= s1; end

         s1: begin out <= 5'b00001; ns <= s2; end

         s2: begin out <= 5'b00010; ns <= s3; end

         s3: begin out <= 5'b00011; ns <= s4; end

         s4: begin out <= 5'b00100; ns <= s5; end

         s5: begin out <= 5'b00101; ns <= s6; end

         s6: begin out <= 5'b00110; ns <= s7; end

         s7: begin out <= 5'b00111; ns <= s8; end

         s8: begin out <= 5'b01000; ns <= s9; end

         s9: begin out <= 5'b01001; ns <= s10; end

         s10: begin out <= 5'b01010; ns <= s11; end

         s11: begin out <= 5'b01011; ns <= s12; end

         s12: begin out <= 5'b01100; ns <= s13; end

         s13: begin out <= 5'b01101; ns <= s14; end

         s14: begin out <= 5'b01110; ns <= s15; end

         s15: begin out <= 5'b01111; ns <= s16; end

         s16: begin out <= 5'b10000; ns <= s17; end
```

```

            s17: begin out <= 5'b10001; ns <= s18; end

            s18: begin out <= 5'b10010; ns <= s19; end

            s19: begin out <= 5'b10011; ns <= s20; end

            s20: begin out <= 5'b10100; ns <= s21; end

            s21: begin out <= 5'b10101; ns <= s22; end

            s22: begin out <= 5'b10110; ns <= s23; end

            s23: begin out <= 5'b10111; ns <= s24; end

            s24: begin out <= 5'b11000; ns <= s25; end

            s25: begin out <= 5'b11001; ns <= s26; end

            s26: begin out <= 5'b11010; ns <= s27; end

            s27: begin out <= 5'b11011; ns <= s28; end

            s28: begin out <= 5'b11100; ns <= s29; end

            s29: begin out <= 5'b11101; ns <= s30; end

            s30: begin out <= 5'b11110; ns <= s31; end

            s31: begin out <= 5'b11111; ns <= s0; end

         endcase
      end
   endmodule

   module counter_31_testbench();

      logic clk, rst; //repeating logic variables
      logic [4:0] out;

      //test counter_25 module
      counter_31 dut (.clk, .rst, .out);

      // clock setup
      parameter clock_period = 100;

      initial begin
         clk <= 0;
         forever #(clock_period /2) clk = ~clk;
      end // of clock setup

      //an instance where it counts up to five with a
      //space after the first increment, and then
      //decrements twice
      initial begin
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         rst <= 0; @(posedge clk);
         $stop;
      end
   endmodule
```