```
/*
   Ben Davis
   3/5/24
   EE 371
   Lab 6, Task 2

   This module keeps record of when rush hour begins
   and when it ends. If rush hour starts, the output
   hours are defaulted to 15, an hour not in the work day.
   It uses inputs from parking spot presences and also
   the parking lot's clock.
*/

module rush_hour (
   input logic clk, //clock
   input logic rst, //reset
   input logic [3:0] hour, //current hour
   input logic car1, //car presence 1
   input logic car2, //car presence 2
   input logic car3, //car presence 3
   output logic [3:0] rush_start, //hour rush hour starts
   output logic [3:0] rush_end //hour rush hour ends
);

   enum {off, start, middle, finish, done} ps, ns;
   logic full; //full lot
   logic empty; //empty lot
   logic day_end; //the day has ended

   assign day_end = (hour == 4'b1000); //if the day is over

   assign full = (car1 && car2 && car3); //if the lot is full
   assign empty = ((!car1) && (!car2) && (!car3)); // if the lot is empty

   // flip flop to progress the state to ns or to reset
   always_ff @(posedge clk) begin
      if(rst) begin
         ps <= off;
      end else begin
         ps <= ns;
      end
   end

   //state progressions and logic
   always_comb begin
      case(ps)
            off:  begin
                     if(full) ns <= start;
                     else if(day_end) ns <= done;
                     else ns <= off;
                  end
         start:   begin
                     ns <= middle;
                  end
         middle:  begin
                     if(empty) ns <= finish;
                     else if(day_end) ns <= done;
                     else ns <= middle;
                  end
         finish:  begin
                     ns <= done;
                  end
         done:    begin
                     ns <= done;
                  end
      endcase
   end

   // flip flop to update rush hour start if it begins
   always_ff @(posedge clk) begin
      if(rst) begin
         rush_start <= 4'b1110; //initial value is 14
      end else if((ps == off) && (full)) begin
```

```
            rush_start <= hour;
         end else if (day_end && (ps == off)) begin
            rush_start <= 4'b1111; //if no rush hour, value is 15
         end
      end

      // flip flop to track rush hour end if it ends
      always_ff @(posedge clk) begin
         if(rst) begin
            rush_end <= 4'b1110;
         end else if((ps == middle) && (empty)) begin
            rush_end <= hour;
         end else if (day_end && (ps == off)) begin
            rush_end <= 4'b1111;
         end
      end
   endmodule
   // testbench
   module rush_hour_tb ();
      logic clk, rst, car1, car2, car3;
      logic [3:0] hour, rush_start, rush_end;

      rush_hour dut (.*);

      //clock setup
      parameter clock_period = 100;
      initial begin
         clk <= 0;
         forever #(clock_period /2) clk <= ~clk;
      end //of clock setup

      //tests a case where rush hour starts hr 2 and ends hr 4
      initial begin
         rst <= 0; hour <= 4'b0000; car1 <= 0; car2 <= 0;
                   car3 <= 0; @(posedge clk);
         rst <= 0; hour <= 4'b0000; car1 <= 1; car2 <= 0;
                   car3 <= 0; @(posedge clk);
         rst <= 0; hour <= 4'b0001; car1 <= 1; car2 <= 0;
                   car3 <= 0; @(posedge clk);
         rst <= 0; hour <= 4'b0010; car1 <= 1; car2 <= 1;
                   car3 <= 1; @(posedge clk);
         rst <= 0; hour <= 4'b0011; car1 <= 1; car2 <= 0;
                   car3 <= 0; @(posedge clk);
         rst <= 0; hour <= 4'b0100; car1 <= 0; car2 <= 0;
                   car3 <= 0; @(posedge clk);
         rst <= 0; hour <= 4'b0101; car1 <= 0; car2 <= 0;
                   car3 <= 0; @(posedge clk);
         $stop;
      end
   endmodule
```