```
/* Name: Eugene Ngo
   Date: 3/7/2023
   Class: EE 371
   Lab 6: Parking Lot 3D Simulation
*/

// addrIter iterates through the different addresses on the RAM
// this is used in the datapath unit, once the parking lot is in the
// state of showing RAM data. It iterates through the integers
// 0 - 7, cycling back to 0 once it reaches 7. It is reset
// via the reset input, SW[9].
`timescale 1 ps / 1 ps
module addrIter (inc, clk, reset, out);

   // Basic I/O. Inc enables incrememeting the address,
   // this is triggered once parking lot is at the end of day.
   // The output is a 4 bit number used to access RAM memory.

   input logic inc, clk, reset;
   output logic [3:0]  out;

   // Sequential logic for counting up and counting down depending on the input.
   always_ff @(posedge clk) begin
      // reset if reset switch is flipped.
      if (reset) begin
         out <= 4'b0000;
      end
      // incremenet when the counter is not at max (7).
      else if (inc & out < 4'b1000) begin //increment when not at max
         out <= out + 4'b0001;
      end
      // reset to zero if the value ever exceeds max (7)
      else if (out > 4'b0111) begin
         out <= 4'b0000;
      end
      // keep the value at out if none of the other conditions are met.
      else
         out <= out; // hold value otherwise
   end // always_ff

endmodule

// addrIter_testbench tests all expected, unexpected, and edgecase behaviors
// to ensure the module iterates through addresses 0 to 7, making sure the value
// output value is updated properly.
module addrIter_testbench();

   // Same I/O as addrIter()
   logic inc, clk, reset;
   logic [3:0] out;
   logic CLOCK_50;

   addrIter dut (.inc, .clk(CLOCK_50), .reset, .out);

   // Setting up the clock.
   parameter CLOCK_PERIOD = 100;
   initial begin
      CLOCK_50 <= 0;
      forever #(CLOCK_PERIOD/2) CLOCK_50 <= ~CLOCK_50; // toggle the clock forever
   end // initial

   initial begin
      reset <= 1;                 @(posedge CLOCK_50); // reset
      reset <= 0;                 @(posedge CLOCK_50); // inc past max limit
      inc <= 0;                   repeat(7) @(posedge CLOCK_50); // dec past min limit
      inc <= 1;                   repeat(30) @(posedge CLOCK_50); // dec past min limit
      $stop;
   end
endmodule // counter_testbench
```