```systemverilog
// An Nguyen, Darren Do
// 3/8/2024
// EE 371
// Lab #6, Task #2

// counter takes in 2 input logic 'clk' and 'reset', in addition to 'en'. It
// outputs a 3-bit logic 'out'. clk represents the clock used and reset represents the input
// needed to reset the system to state 0. 'en' is used to signify that parking lot operational
// hours has ended and we need to cycle through the RAM. 'Out' represents working hour, starting
// from 0 -> 8. With this clock, it should cycle through the RAM at roughly 1 second/transition.
module counter(clk, reset, en, out);

   input logic clk, reset;
   input logic en;
   output logic [2:0] out;
   // intermediate logic to keep track of the current hour
   logic [2:0] count;

   // if reset is called, or en is NOT enabled, hold the count @ '0'.
   // Else, keep increasing count.
   always_ff @(posedge clk) begin
      if (reset | ~en)
         count <= 3'b000;
      else
         count <= count + 3'b001;
   end

   // used a temp variable to keep track of the count before assigning it to the
   // final output.
   assign out = count;

endmodule

// counter_testbench tests the functionality of the counter module. For this testbench,
// we care about the case that the counting portion of our 'counter' should only work when
// en is '1', else it stays in the '0' count. We also care that after it counts up to '7',
// it should be able to cycle back to '0'.
module counter_testbench();
   logic clk, reset, en;
   logic [2:0] out;

   counter dut1 (.*);

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

   end

   initial begin
      // Initialize our inputs
      reset <= 1; en <= 0; @(posedge clk);
      reset <= 0; en <= 1; @(posedge clk);
      // Let it cycle
      for (int i = 0; i < 8; i++) begin
         @(posedge clk);
      end
      $stop;
   end

endmodule

```