Test bench design example 2 - Ring counter

source [r_counter.v]

```
//-----------------------------------------------------
// Project Name : ring_counter
// File Name : r_counter.v
// Function : This is a 8 bit ring counter with synchronous active high reset and
// with active high enable signal
//-----------------------------------------------------

module r_counter (
clock , // Clock input of the design
reset , // active high, synchronous Reset input
enable , // Active high enable signal for counter
counter_out // 8 bit vector output of the counter
); // End of port list

//-------------Input
input clock, reset, enable ;
//-------------Output
output [7:0] counter_out ;

//-------------Input ports Data Type : By rule all the input ports should be wire
wire clock, reset, enable ;
//-------------Output Ports Data Type------------------
// Output port can be a storage element (reg) or a wire
reg [7:0] counter_out ;

//------------Code Starts Here-------------------------
// Since this counter is a positive edge trigged one,
// We trigger the below block with respect to positive edge of the clock.
always @ (posedge clock)
begin : COUNTER // Block Name
  // At every rising edge of clock we check if reset is active
  // If active, we load the counter output with 8'b0000_0001
  if (reset == 1'b1) begin
    counter_out <= #1 8'b0000_0001;
  end
  // If enable is active, then we increment the counter
  else if (enable == 1'b1) begin
    counter_out <= counter_out << 1;
    counter_out[0] <= counter_out[7];
  end
end // End of Block COUNTER

endmodule // End of Module counter
```

test bench [r_counter_tb.v]

```
`include "r_counter.v"
module r_counter_tb();
// Declare inputs as regs and outputs as wires

reg clock, reset, enable;
wire [7:0] counter_out;

// Connect DUT to test bench
r_counter U_counter(clock,reset,enable,counter_out);

// Initialize all variables
initial begin
  $display ("time₩t clk reset enable counter");
  $monitor ("%g₩t %b    %b      %b       %b",
          $time, clock, reset, enable, counter_out);
  clock = 1;        // initial value of clock
  reset = 0;        // initial value of reset
  enable = 0;       // initial value of enable
  #5 reset = 1;     // Assert the reset
  #10 reset = 0;    // De-assert the reset
  #10 enable = 1;   // Assert enable
  #200 enable = 0; // De-assert enable
  #5 $finish;       // Terminate simulation
end

// Clock generator
always begin
  #5 clock = ~clock; // Toggle clock every 5 ticks
end


endmodule
```