Cod Design Semafor

```
module semafor (
input     clk              , // semnal de ceas,
input     rst_n            , // semnal de reset asincron activ 0
input     buton            , // buton
output    semafor_masini   , // output1
output    semafor_pietoni    // output2
);

reg [1:0] stare_prezenta;
reg [1:0] stare_viitoare;
reg [8:0] counter       ;
reg buton_apasat        ;
reg ok1                 ;
reg ok2                 ;

initial begin
counter      <= 8'b0;
buton_apasat <= 1'b0;
ok1 <= 1'b0;
ok2 <= 1'b0;
end

// coduri stari (unice)
localparam stare1    = 2'b00; // verde pietoni , rosu   masini
localparam stare2    = 2'b11; // rosu  pietoni , verde  masini
localparam stare3    = 2'b10; // rosu  pietoni , galben masini

always @(posedge clk)
counter <= counter + 1;
// modeleaza registrul de stare
always @(posedge clk or negedge rst_n)
if (~rst_n) begin
            stare_prezenta <= stare1;  // stare initiala
            counter        <= 8'b0  ;
end
else        stare_prezenta <= stare_viitoare;
// modeleaza circuitul combinational de stare
always @(*)
case (stare_prezenta)
stare1 :  begin
          ok1 <= 1'b1;
          ok2 <= 1'b0;
          buton_apasat <= 1'b0;
          if(counter == 'd30) begin
              counter        <= 'b0;
              stare_viitoare <= stare3;
          end
          end
stare2 :  begin
          ok2 <= 1'b1;
          ok1 <= 1'b0;
          if(buton && !buton_apasat) begin
              buton_apasat <= 1'b1;
              counter <= 8'b0;
          end
          else if (buton_apasat == 1'b1 && counter == 'd60) begin
                   counter        <= 8'b0;
                   stare_viitoare <= stare3;
          end
```

```
            end
stare3 :    begin
            buton_apasat <= 1'b0;
            if(counter == 'd5) begin
                counter         <= 'b0;
                if(ok1 == 1'b1)
                stare_viitoare <= stare2;
                else if(ok2 == 1'b1)
                stare_viitoare <= stare1;
            end
            end
default     :  stare_viitoare <= stare1;  // s_final
endcase

// modeleaza circuitul combinational de iesire
assign semafor_masini    = (stare_prezenta == stare2  );
assign semafor_pietoni   = (stare_prezenta == stare1  );

endmodule // semafor
```

Cod Test Semafor

```
`timescale 1s/1s
module semafor_test;
wire      clk             ;
wire      rst_n           ;
reg       buton           ;
wire      semafor_masini  ;
wire      semafor_pietoni;


ck_rst_tb #(
.CK_SEMIPERIOD ('d5)
) i_ck_rst_tb (
.clk     (clk   ),
.rst_n   (rst_n )
);

initial begin
  buton <= 1'bx;
  @(negedge rst_n);
  buton <= 1'b0;
  @(posedge rst_n);
  @(posedge clk);
  buton <= 1'b1;
  #1;
  buton <= 1'b0;
  #5;
  buton <= 1'b1;
  #15;
  buton <= 1'b0;
  #5;
  buton <= 1'b1;
  #10;
  buton <= 1'b0;
  #5;
  buton <= 1'b1;
  #7;
```

```
  buton <= 1'b0;
  #5;
  buton <= 1'b1;
  #25;
  buton <= 1'b0;
  #5;
  buton <= 1'b1;
  #10;
  buton <= 1'b0;
  #3;
  buton <= 1'b1;
  #50;
  buton <= 1'b0;
  $display ("%M %0t INFO: Final simulare.", $time);
  $stop;
end

semafor i_semafor (
.clk        (clk        ),
.rst_n      (rst_n      ),
.buton      (buton        ),
.semafor_masini    (semafor_masini    ),
.semafor_pietoni   (semafor_pietoni   )

);

endmodule // semafor_test
```

Cod ck_rst_tb

```
`timescale 100ms/100ms

module ck_rst_tb #(
parameter CK_SEMIPERIOD = 'd10        // semi-perioada semnalului de ceas
)(
output reg              clk         , // ceas
output reg              rst_n         // reset asincron activ 0
);
initial
begin
  clk = 1'b0;             // valoare initiala 0
  forever #CK_SEMIPERIOD  // valoare complementata la fiecare semi-perioada
    clk = ~clk;
end

initial begin
  rst_n <= 1'b1;    // initial inactiv
  @(posedge clk);
  rst_n <= 1'b0;    // activare sincrona
  @(posedge clk);
  @(posedge clk);
  rst_n <= 1'b1;    // inactivare dupa doua perioade de ceas
  @(posedge clk);   // ramane inactiv pentru totdeauna
end

endmodule // ck_rst_tb
```

Cod sim.do

```
vlib work
vmap work work

vlog  ../../hdl/semafor.v
vlog  ../hdl/semafor_test.v
vlog  ../hdl/ck_rst_tb.v

vsim -novopt work.semafor_test
do wave_fsm.do

run -all
```

Explicatii:

In codul de design, avem modulul semafor, initializam in primul initial begin cu 0 unele valori, facem counter-ul sa creasca cu +1 mereu, avem acel always cat timp reset ul este activ in 0, dupa care avem cazurile pentru fiecare stare: Verde pietoni cu Rosu Masini, Rosu Pietoni cu Verde Masini, Rosu Pietoni cu Galben Masini, ne folosim de ok-uri in starea 3 pentru trecerile din galben in rosu/verde, avem apoi cele 2 assign-uri pentru semafor_masini si semafor_pietoni care sunt active cand starile 2, respectiv 1 sunt 1.

In testbench, am introdus o perioada pentru buton si am completat introducerea si codul pentru design, am modificat apoi si in sim.do, iar sub acest text las si o poza cu rezultatul.