Semafor cu buton pentru pietoni

Modul design SEMAFOR

```
module semafor #(
parameter WIDTH = 6     // numar de biti pentru numarator de secvente
)
(
input   clk       , // ceas, 1 MHz
input   rst_n     , // reset asincron activ 0
input   buton     , //buton pietoni
output  rosu      , // rosu masini
output  galben    , //galben masini
output  verde     , //verde masini
output  rosu_p    , //rosu pietoni
output  verde_p     //verde pietoni
);

reg [WIDTH  -1:0]   num_secunde       ;
reg [WIDTH  -1:0]   num_secunde_val   ;
wire                num_secunde_zero  ;
wire                load_sec          ; // se incarca numaratorul cu
valoarea secundelor ciclului

reg [3      -1:0]   stare_prez        ;
reg [3      -1:0]   stare_viitoare    ;

reg                 buton_apasat      ;

localparam  ROSU_INIT     = 3'b000;
localparam  ROSU_CNT      = 3'b001;
localparam  GALBEN_INIT   = 3'b010;
localparam  GALBEN_CNT    = 3'b011;
localparam  VERDE_INIT    = 3'b100;
localparam  VERDE_CNT     = 3'b101;

// numarator de secunde presetabil
always @(posedge clk or negedge rst_n)
if (~rst_n)   num_secunde <= {WIDTH{1'b1}}; else
if (load_sec) num_secunde <= num_secunde_val; else
              num_secunde <= num_secunde - 1;

assign num_secunde_zero = ~|num_secunde;   // SAU-NU (=0)

// logica de stare
always @(*)
case (stare_prez)
ROSU_CNT    : stare_viitoare =  num_secunde_zero ?  VERDE_INIT  : ROSU_CNT;
VERDE_INIT  : stare_viitoare =  VERDE_CNT;
VERDE_CNT   : stare_viitoare =  num_secunde_zero ? (buton_apasat ?
GALBEN_INIT : VERDE_CNT) : VERDE_CNT;
GALBEN_INIT : stare_viitoare =  GALBEN_CNT;
GALBEN_CNT  : stare_viitoare =  num_secunde_zero ?  ROSU_INIT   :
GALBEN_CNT;
default     : stare_viitoare =  ROSU_CNT;      // ROSU_INIT
```

```
endcase

always @(*)
if(stare_prez == VERDE_CNT) begin
    if(buton) buton_apasat = 1; end
else buton_apasat = 0;

// registru de stare
always @(posedge clk or negedge rst_n)
if (~rst_n)   stare_prez <= ROSU_INIT; else
              stare_prez <= stare_viitoare;

// logica de iesire
always @(*)
  case (stare_prez)
  VERDE_INIT  : num_secunde_val =  'd60;
  GALBEN_INIT : num_secunde_val =  'd5 ;
  ROSU_INIT   : num_secunde_val =  'd30;
  default     : num_secunde_val =  'd0;
  endcase

//modeleaza circuitul combinational de iesire
assign load_sec   = (stare_prez == ROSU_INIT)   |
                    (stare_prez == GALBEN_INIT) |
                    (stare_prez == VERDE_INIT) ;

assign rosu   = (stare_prez == ROSU_INIT)     |
                (stare_prez == ROSU_CNT)      ;
assign galben = (stare_prez == GALBEN_INIT)   |
                (stare_prez == GALBEN_CNT)    ;
assign verde  = (stare_prez == VERDE_INIT)    |
                (stare_prez == VERDE_CNT)     ;
assign rosu_p = verde | galben;
assign verde_p= rosu;

endmodule
```

Modul de test SEMAFOR

```
module semafor_test;
parameter WIDTH  =6 ;

wire clk       ;
wire rst_n     ;
reg  buton     ;
wire rosu      ;
wire galben    ;
wire verde     ;
wire rosu_p    ;
wire verde_p   ;

ck_rst_tb #(
.CK_SEMIPERIOD ('d10)
) i_ck_rst_tb (
.clk    (clk   ),
```

```
.rst_n  (rst_n )
);

initial begin
    buton <= 1'bx;
    @(negedge rst_n);
    @(posedge rst_n);
    @(posedge clk);
    buton <= 1'b0;
    repeat (65) @(posedge clk);
    buton <= 1'b1; //activam butonul
    @(posedge clk);
    buton <= 1'b0;
    repeat (80) @(posedge clk);
    buton <= 1'b1;
    @(posedge clk);
    buton <= 1'b0;
end


semafor #(
.WIDTH (WIDTH)
)  i_semafor (
.clk     (clk    ),
.rst_n   (rst_n  ),
.buton   (buton  ),
.rosu    (rosu   ),
.galben  (galben ),
.verde   (verde  ),
.rosu_p  (rosu_p ),
.verde_p (verde_p)
);

Endmodule
```

Modul clock si reset

```
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
```

Fisierul sim.do

```
vlib work
vmap work work

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

vsim -novopt work.semafor_test
do wave.do

run -all
```

Fisierul wave.do

```
onerror {resume}

quietly WaveActivateNextPane {} 0

add wave -noupdate /semafor_test/i_semafor/clk

add wave -noupdate /semafor_test/i_semafor/rst_n

add wave -noupdate /semafor_test/i_semafor/buton

add wave -noupdate /semafor_test/i_semafor/rosu

add wave -noupdate /semafor_test/i_semafor/verde

add wave -noupdate /semafor_test/i_semafor/galben

add wave -noupdate /semafor_test/i_semafor/rosu_p

add wave -noupdate /semafor_test/i_semafor/verde_p

add wave -noupdate /semafor_test/i_semafor/stare_prez

add wave -noupdate /semafor_test/i_semafor/stare_viitoare

TreeUpdate [SetDefaultTree]

WaveRestoreCursors {{Cursor 1} {1519 ns} 0}

configure wave -namecolwidth 173

configure wave -valuecolwidth 59
```

```
configure wave -justifyvalue left
configure wave -signalnamewidth 1
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
configure wave -timelineunits ns
update
WaveRestoreZoom {0 ns} {2368 ns}
```