FSM semafor u Verilog-u u Vivado simulatoru

Primer objedinjuje: grananje (`if`, `case`), sekvencijalnu i kombinacionu logiku, testbench i tumačenje talasnih oblika.

- `case` bira ponašanje u zavisnosti od stanja FSM-a.
- `if` proverava kada treba preći u sledeće stanje.
- `always @(posedge clk or posedge rst)` opisuje registre i reset.
- `always @(*)` opisuje kombinacionu logiku: next_state i izlaze.
- `forever #5 clk = ~clk;` u testbench-u pravi clock.
- `$monitor` i waveform služe da proverimo da li dizajn radi kako očekujemo.

Verilog nije “C sa drugačijom sintaksom”, već opis hardvera kroz stanja, registre, ulaze, izlaze i vreme.

Struktura projekta

Jedan RTL modul + jedan testbench.

tb_traffic_light

generiše clk
postavlja rst
zaustavlja
simulaciju

traffic_light

FSM stanja:
S_RED
S_GREEN
S_YELLOW

brojač
+
logika
prelaza

Ulazi iz testbench-a:
clk, rst

Glavni RTL modul: tri logičke celine

- 1) Registrovanje stanja i brojača
  `always @(posedge clk or posedge rst)`
- 2) Računanje sledećeg stanja
  `always @(*)` + `case(state)`
- 3) Generisanje izlaza
  `always @(*)` + aktivno svetlo

Jasnije razdvajamo memoriju stanja od kombinacione logike.

```
always @(posedge clk or posedge rst) begin
    if (rst) begin
        state   <= S_RED;
        counter <= 0;
    end else begin
        state <= next_state;
        if (state != next_state)
            counter <= 0;
        else
            counter <= counter + 1;
    end
end

always @(*) begin
    next_state = state;
    case (state)
        S_RED:    if (counter == 4) next_state = S_GREEN;
        S_GREEN:  if (counter == 4) next_state = S_YELLOW;
        S_YELLOW: if (counter == 2) next_state = S_RED;
        default:  next_state = S_RED;
    endcase
end
```

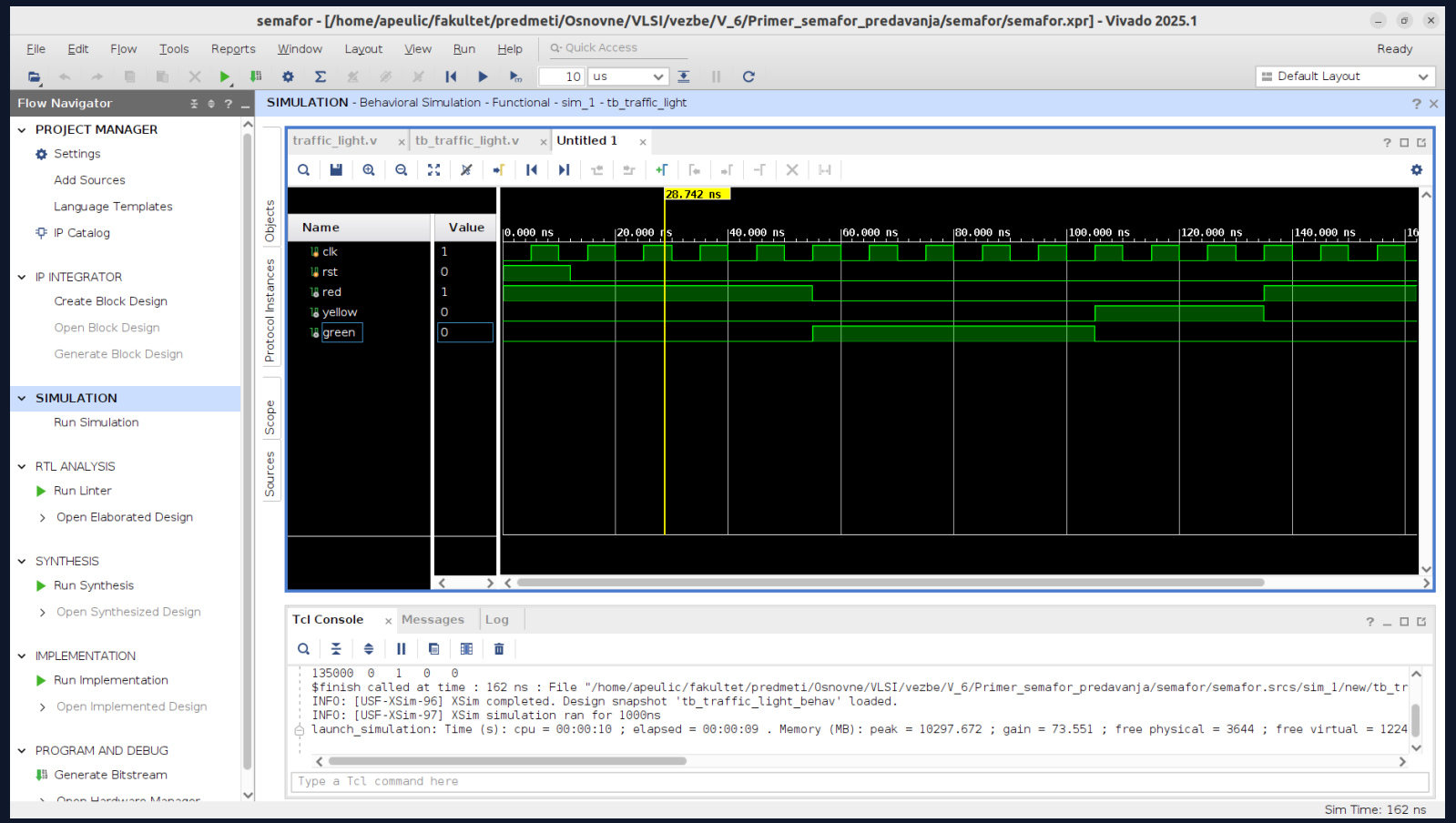
Testbench i vremenska kontrola

```
initial begin
    clk = 0;
    forever #5 clk = ~clk;
end

initial begin
    rst = 1;
    #12 rst = 0;
    #150;
    $finish;
end

initial begin
    $monitor("%0t %b %b %b %b",
             $time, rst, red, yellow, green);
end
```

- `forever #5` pravi clock periode 10 ns.
- `rst = 1`, pa posle `#12` prelazi na 0: sistem kreće iz poznatog stanja.
- `$finish` zatvara simulaciju da ne traje beskonačno.
- `$monitor` je koristan za tekstualnu proveru i poređenje sa waveform-om.

testbench opisuje scenario simulacije, a ne hardver koji će se sintetisati.

Tumačenje talasnih oblika u Vivado-u



- Na početku je `rst = 1`, pa je aktivno crveno svetlo.
- Posle otpuštanja reseta FSM ostaje neko vreme u stanju `S_RED`.
- Zatim prelazi na `S_GREEN`, pa na `S_YELLOW`.
- Na kraju se ciklus vraća u `S_RED` i ponašanje se ponavlja.

Predlozi za mini-zadatke posle časa

Zadatak 1

Promeniti kod tako da `S_GREEN` traje 6 taktova umesto 4.

Zadatak 2

Dodati novo stanje `S_RED_YELLOW` pre prelaza na zeleno.

Zadatak 3

Dodati izlaz `blink` koji treperi dok je aktivno žuto svetlo.