Traffic light controller

1. Scopul Proiectului

Acest proiect implementeaza un sistem de control pentru semafoare intr-o intersectie formata dintr-un drum principal (main road) si un drum secundar (secondary road). Sistemul prioritizeaza traficul pe drumul principal, oferind in mod normal lumina verde. In momentul in care senzorul de pe drumul secundar detecteaza un vehicul, semaforul de pe drumul principal trece pe galben, apoi rosu, permitand accesul pe drumul secundar. Dupa un timp prestabilit, traficul este din nou permis pe drumul principal.

2. Arhitectura Sistemului

Designul este implementat in VHDL si contine urmatoarele componente:

- O masina de stari finite (FSM) cu 4 stari principale
- Semnale de control pentru activarea temporizatoarelor (enable)
- Un contor de 1s si un contor de secunde pentru implementarea delay-urilor
- Un testbench pentru simulare si validare

3. Descrierea FSM (Finite State Machine)

FSM-ul gestioneaza starea curenta a semafoarelor si permite tranzitia intre fazele de semafor pe baza senzorului si a timpului scurs.

Stari posibile:

1. MGRE_SRED – Semafor verde pe drumul principal, rosu pe secundar
2. MYEL_SRED – Galben pe drumul principal, rosu pe secundar
3. MRED_SGRE – Rosu pe principal, verde pe secundar
4. MRED_SYEL – Rosu pe principal, galben pe secundar

```
type FSM_States is (MGRE_SRED, MYEL_SRED, MRED_SGRE, MRED_SYEL);
```

FSM-ul face tranzitia intre aceste stari in functie de:

- prezenta unui vehicul detectat de senzor (sensor)
- semnalele de temporizare (delay_3s_M, delay_3s_S, delay_10s)

4. Semnale de control si intarzieri

Pentru a respecta durata standard a fazelor semaforului (3 secunde pentru galben, 10 secunde pentru rosu), au fost implementate mai multe semnale interne:

- RED_LIGHT_ENABLE: activeaza numaratoarea pentru rosu (10 secunde)
- YELLOW_LIGHT1_ENABLE: activeaza galbenul pe drumul principal (3 secunde)
- YELLOW_LIGHT2_ENABLE: activeaza galbenul pe drumul secundar (3 secunde)

Aceste semnale sunt activate in functie de starea FSM-ului si sunt evaluate doar cand ceasul de 1s (clk_1s_enable) este activ. Numaratorul de secunde (delay_count) este incrementat doar cand unul dintre semnalele ENABLE este activ.

Cand este atinsa valoarea dorita (2 pentru galben, 9 pentru rosu), semnalele delay_3s_M, delay_3s_S, respectiv delay_10s devin active pentru un singur ciclu de ceas, dupa care sunt resetate.

```
signal delay_10s, delay_3s_S, delay_3s_M, RED_LIGHT_ENABLE, YELLOW_LIGHT1_ENABLE, YELLOW_LIGHT2_ENABLE: std_logic := '0';
signal clk_1s_enable: std_logic; -- Semnal de activare la fiecare 1 secunda
```

5. Temporizare si generare semnal de 1s

Semnalul de baza clk (ceas de 50MHz in aplicatie reala, simulat mai rapid) este divizat intern pentru a crea un semnal de enable (clk_1s_enable) care pulseaza o data pe secunda. Acest semnal este folosit pentru a controla numaratorul delay_count, care determina durata fazelor semaforului.

6. Logica pentru semafoare

Starea curenta controleaza direct valorile de iesire pentru semafoarele de pe drumul principal (light_main) si drumul secundar (light_secondary). Valorile semaforului sunt:

- "100" – rosu
- "010" – galben
- "001" – verde

In fiecare stare, se activeaza semafoarele corespunzatoare, si se seteaza semnalele de enable doar daca este nevoie de cronometrare.

Exemplu:

- In starea MYEL_SRED, semaforul principal este galben, cel secundar ramane rosu, iar YELLOW_LIGHT1_ENABLE este activ pentru a cronometra 3 secunde.

7. Testbench (tb_traffic_light_controller)

Testbench-ul simuleaza functionarea controlerului prin:

- Generarea semnalului de ceas (clk)

```
clk_process : process
begin
    clk <= '0';
    wait for clk_period/2;
    clk <= '1';
    wait for clk_period/2;
end process;
```

- Resetarea initiala a sistemului (rst_n <= '0')
- Activarea senzorului (sensor <= '1') dupa un anumit timp
- Deactivarea senzorului pentru a observa revenirea la starea initiala

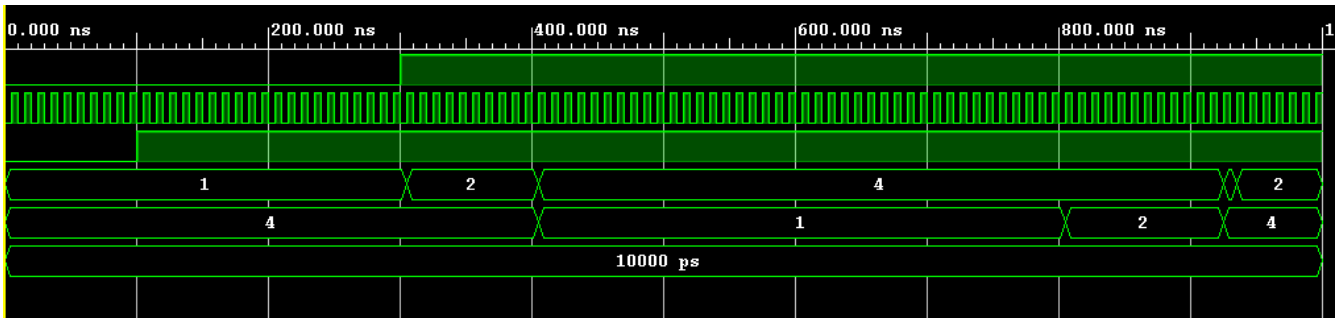
```
stim_proc: process
begin
    rst_n <= '0';        -- Resetare initiala
    sensor <= '0';       -- Fara vehicule pe drumul secundar
    wait for clk_period * 10;

    rst_n <= '1';        -- Se scoate resetarea
    wait for clk_period * 20;

    sensor <= '1';       -- Vehicul detectat pe drumul secundar
    wait for clk_period * 100;

    sensor <= '0';       -- Niciun vehicul detectat
    wait;
end process;
```

8.Rezultatul simularii



- **Intervale si comportamente observate:**

0 ns – 200 ns:

- sensor = 0 → Nu e detectat niciun vehicul
- light_main = 001 → Verde pe drumul principal
- light_secondary = 100 → Rosu pe drumul secundar
- → Aceasta este starea initiala corecta (MGRE_SRED), conform FSM-ului

200 ns –300 ns:

- sensor = 1 → Apare o masina pe drumul secundar
- FSM-ul trece in starea MYEL_SRED:
  - light_main = 010 → Galben pe drumul principal
  - light_secondary = 100 → Rosu pe drumul secundar
- → Este declansata temporizarea pentru galben (3 secunde simulate)

300 ns – 600 ns:

- Se activeaza starea MRED_SGRE:
  - light_main = 100 → Rosu pe drumul principal
  - light_secondary = 001 → Verde pe drumul secundar
- → Traficul este permis pe drumul secundar timp de 10 secunde simulate

600 ns – 700 ns:

- FSM-ul trece in MRED_SYEL:
  - light_main = 100 → Rosu pe drumul principal
  - light_secondary = 010 → Galben pe drumul secundar
- → Se cronometreaza 3 secunde de galben pe drumul secundar

700 ns – sfarsit:

- FSM-ul revine la MGRE_SRED:
  - light_main = 001 → Verde pe drumul principal
  - light_secondary = 100 → Rosu pe drumul secundar
- → Se revine in starea initiala.