Universidade Tecnológica Federal do Paraná
Engenharia da Computação
Lógica Reconfigurável

Relatório 8 - Semáforos

Aluno: Eduardo Yuji Yoshida Yamada
Professor orientador: Marcelo de Oliveira

Janeiro
2025

Conteúdo

1 Introdução **1**

2 Diagrama de Estados **2**

3 Códigos **4**

4 Diagrama RTL **9**

1 Introdução

Este relatório apresenta a implementação de dois semáforos utilizando máquinas de estado finito, garantindo o controle adequado da sinalização luminosa por meio de LEDs. Cada semáforo alterna entre os estados verde, amarelo e vermelho conforme um tempo predefinido. A lógica de funcionamento considera que, enquanto um semáforo está verde, o outro permanece vermelho, e vice-versa, mas com tempos distintos para cada um. Além disso, quando um dos semáforos está no estado amarelo, o outro permanece no vermelho para garantir uma transição segura.

Para aumentar a funcionalidade do sistema, foi implementado um modo de standby, no qual ambos os semáforos entram em estado de alerta e piscam a luz amarela continuamente. Esse modo pode ser utilizado para simular períodos noturnos ou situações em que a sinalização normal deve ser suspensa temporariamente.

A implementação foi realizada utilizando máquinas de estado, garantindo um controle preciso e determinístico da troca de sinais. O desenvolvimento deste projeto envolve conceitos fundamentais de sistemas digitais, como a modelagem de máquinas de estado, controle de tempos e manipulação de LEDs para simulação de semáforos reais.

2 Diagrama de Estados

Para o diagrama de estados feito antes da implementação do código, temos o seguinte esboço:

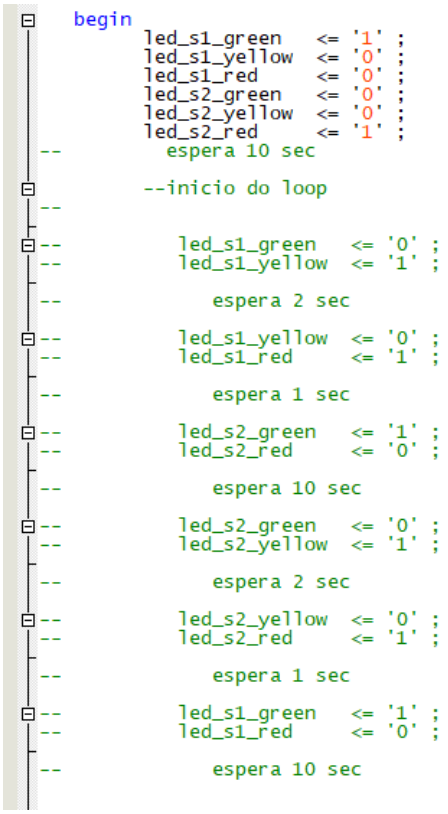

Figura 1: Diagrama de Estados

Após a implementação do código, o software *Quartus Prime* gerou esse modelo do diagrama de estados:

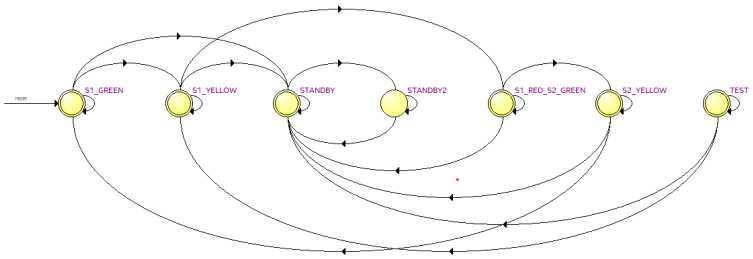

Figura 2: Diagrama de Estados Quartus

3 Códigos

Foi implementado o seguinte código para a realização da atividade:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Definicao da entidade do semaforo
entity projeto8 is
    Port (
        clk             : in  STD_LOGIC;  -- Clock de
           entrada
        reset           : in  STD_LOGIC;  -- Reset do
           sistema
        mode            : in  STD_LOGIC;  -- Modo de
           operacao (0: normal, 1: teste)
        standby_switch  : in  STD_LOGIC;  -- Ativa o
           modo Standby
        led_s1_green    : out STD_LOGIC;  -- LED verde
           do semaforo 1
        led_s1_yellow   : out STD_LOGIC;  -- LED amarelo
            do semaforo 1
        led_s1_red      : out STD_LOGIC;  -- LED
           vermelho do semaforo 1
        led_s2_green    : out STD_LOGIC;  -- LED verde
           do semaforo 2
        led_s2_yellow   : out STD_LOGIC;  -- LED amarelo
            do semaforo 2
        led_s2_red      : out STD_LOGIC   -- LED
           vermelho do semaforo 2
    );
end projeto8;

-- Definicao da arquitetura baseada em maquinas de
   estado finito (FSM)
architecture fsm of projeto8 is

    -- Definicao dos estados possiveis da FSM
    type state_type is (S1_GREEN, S1_YELLOW,
       S1_RED_S2_GREEN, S2_YELLOW, STANDBY, STANDBY2,
       TEST);
```

```
    -- Sinais para armazenar o estado atual e o proximo
       estado
    signal current_state, next_state : state_type;
    signal counter : integer := 0;  -- Contador para
       definir tempos dos estados

    -- Constantes que definem os tempos de cada estado
    constant GREEN_TIME  : integer := 100000000; --
       Tempo no sinal verde
    constant YELLOW_TIME : integer := 200000000; --
       Tempo no sinal amarelo
    constant RED_TIME    : integer := 10000000;  --
       Tempo no sinal vermelho
    constant TEST_SPEED  : integer := 20000000;  --
       Tempo reduzido para modo de teste

begin

    -- Processo de controle da FSM
    process (clk, reset)
    begin
        if reset = '1' then  -- Se reset for ativado,
           reinicia para o estado inicial
            current_state <= S1_GREEN;
            counter <= 0;
        elsif rising_edge(clk) then  -- A cada ciclo de
           clock, verifica transicao de estado
            if counter = 0 then  -- Se contador zerou,
               muda de estado
                current_state <= next_state;
                if mode = '1' then  -- Se estiver no
                   modo de teste, usa tempos menores
                    counter <= TEST_SPEED;
                else
                    -- Define o tempo para o proximo
                       estado conforme o estado atual
                    case next_state is
                        when S1_GREEN => counter <=
                           GREEN_TIME;
                        when S1_YELLOW => counter <=
                           YELLOW_TIME;
                        when S1_RED_S2_GREEN => counter
                           <= GREEN_TIME;
```

```
                    when S2_YELLOW => counter <=
                       YELLOW_TIME;
                    when others => counter <=
                       GREEN_TIME;
                end case;
            end if;
        else
            counter <= counter - 1;  -- Decrementa o
                contador a cada ciclo
        end if;
    end if;
end process;

-- Processo para definir o proximo estado e os LEDs
   correspondentes
process (current_state, mode, standby_switch)
begin
    if standby_switch = '1' then  -- Se o modo
       standby for ativado
        -- LEDs piscam em amarelo
        led_s1_green  <= '0'; led_s1_yellow <= '1';
           led_s1_red <= '0';
        led_s2_green  <= '0'; led_s2_yellow <= '1';
           led_s2_red <= '0';
        next_state <= STANDBY;
    else
        case current_state is
            when S1_GREEN =>  -- Semaforo 1 verde,
               semaforo 2 vermelho
                led_s1_green  <= '1'; led_s1_yellow
                   <= '0'; led_s1_red <= '0';
                led_s2_green  <= '0'; led_s2_yellow
                   <= '0'; led_s2_red <= '1';
                next_state <= S1_YELLOW;

            when S1_YELLOW =>  -- Semaforo 1 amarelo
               , semaforo 2 continua vermelho
                led_s1_green  <= '0'; led_s1_yellow
                   <= '1'; led_s1_red <= '0';
                led_s2_green  <= '0'; led_s2_yellow
                   <= '0'; led_s2_red <= '1';
                next_state <= S1_RED_S2_GREEN;
```

```
when S1_RED_S2_GREEN =>  -- Semaforo 1
   vermelho, semaforo 2 verde
    led_s1_green  <= '0'; led_s1_yellow
       <= '0'; led_s1_red <= '1';
    led_s2_green  <= '1'; led_s2_yellow
       <= '0'; led_s2_red <= '0';
    next_state <= S2_YELLOW;

when S2_YELLOW =>  -- Semaforo 1
   vermelho, semaforo 2 amarelo
    led_s1_green  <= '0'; led_s1_yellow
       <= '0'; led_s1_red <= '1';
    led_s2_green  <= '0'; led_s2_yellow
       <= '1'; led_s2_red <= '0';
    next_state <= S1_GREEN;

when STANDBY =>  -- Estado de standby (
   piscando amarelo)
    led_s1_green  <= '0'; led_s1_yellow
       <= '1'; led_s1_red <= '0';
    led_s2_green  <= '0'; led_s2_yellow
       <= '1'; led_s2_red <= '0';
    next_state <= STANDBY2;

when STANDBY2 =>  -- Alternancia no modo
    standby (desliga os LEDs por um
   instante)
    led_s1_green  <= '0'; led_s1_yellow
       <= '0'; led_s1_red <= '0';
    led_s2_green  <= '0'; led_s2_yellow
       <= '0'; led_s2_red <= '0';
    next_state <= STANDBY;

when TEST =>  -- Estado de teste (
   transicoes rapidas entre estados)
    led_s1_green  <= '1'; led_s1_yellow
       <= '0'; led_s1_red <= '0';
    led_s2_green  <= '0'; led_s2_yellow
       <= '0'; led_s2_red <= '1';
    next_state <= S1_YELLOW;

when others =>  -- Estado padrao caso
   haja erro
```

```
                    next_state <= S1_GREEN;
            end case;
        end if;
    end process;

end fsm;
```

4 Diagrama RTL

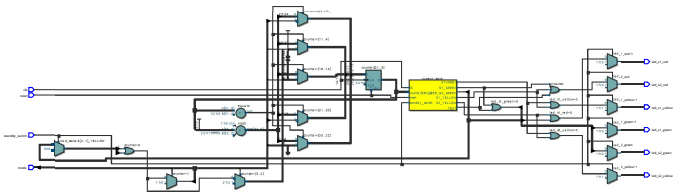

Figura 3: RTL Viewer