UTFPR
Engenharia de Computação
LRCO7A - Lógica Reconfigurável

Atividade 8 - Semáforos

Aluno: Deivid da Silva Galvão RA: 2408740
Aluno: João Vitor Nakahodo Yoshida RA: 2419904
Professor orientador: Marcelo de Oliveira

Janeiro
2025

UTFPR
Engenharia de Computação
LRCO7A - Lógica Reconfigurável

Relatório

Relatório do Trabalho Prático Disciplinar apresentado como requisito parcial à obtenção de nota na disciplina de Lógica Reconfigurável do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Aluno: Deivid da Silva Galvão RA: 2408740

Aluno: João Vitor Nakahodo Yoshida RA: 2419904

Professor orientador: Marcelo de Oliveira

Janeiro
2025

Conteúdo

1 Introdução **1**

2 Conceitos dessa atividade **1**
2.1 Maquina de estados: 1
2.2 Modo Standby: 1
2.3 Modo teste: 1

3 Implementação em VHDL **2**
3.1 Código VHDL: 2
3.2 Explicação do Código: 5

4 Resultados Obtidos **5**

1 Introdução

A atividade 8 foca na implementação de semáforos de cruzamento utilizando VHDL e máquinas de estado finito (FSM). A meta é simular dois semáforos que operam de forma interdependente em um cruzamento, garantindo que quando um está verde, o oposto permanece vermelho. O código VHDL oferece modos de operação Normal, Standby e Teste, controlando os semáforos com uma máquina de estado que alterna entre vermelho, verde e amarelo. Além disso, inclui lógica para que, durante o verde de um semáforo, o outro fique vermelho, simulando o funcionamento real de um cruzamento. No modo Standby, o semáforo fica com a luz amarela piscando, enquanto no modo Teste, os ciclos são acelerados para facilitar a depuração e testes.

2 Conceitos dessa atividade

2.1 Maquina de estados:

A implementação do projeto faz uso de uma máquina de estados para gerenciar o comportamento do sistema. O diagrama abaixo ilustra essa configuração.

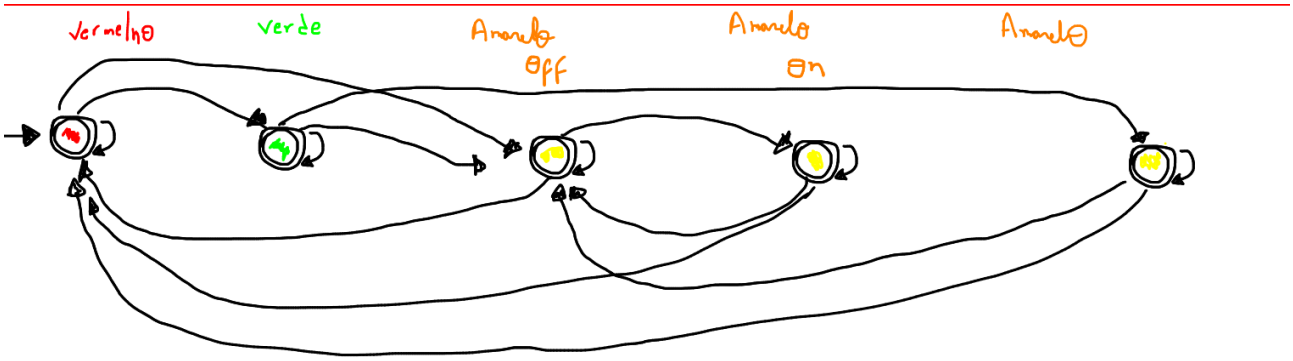

Figura 1: Desenho da maquina de estados

A máquina de estados inclui os estados vermelho, verde, amarelo , amarelo off e amarelo on. Os estados dos semáforos 1 e 2 são controlados pelos sinais dos estados. A mudança de estados ocorre na borda de subida do sinal de clock, e a frequência da máquina de estados é determinada pela constante ‘StateMachineFrequencyHz‘. Durante o modo de teste, a máquina de estados opera em uma frequência diferente, configurada pelo botão de teste.

2.2 Modo Standby:

No modo Standby, a máquina de estados é pausada, controlada pelo sinal ‘pause‘. Quando o ‘pause‘ está em ’0’, a máquina de estados permanece no estado atual, impedindo transições.

2.3 Modo teste:

Durante o modo de teste, a máquina de estados opera em uma frequência diferente, simulando condições específicas para testar o sistema. Esse comportamento é controlado pelo sinal ‘teste‘.

3 Implementação em VHDL

3.1 Código VHDL:

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

entity prog8 is
    generic (
        FrequenciaClockHz : integer := 50_000_000
    );
    Port (
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        pause : in STD_LOGIC;
        teste : in STD_LOGIC;
        semaforo_1 : out STD_LOGIC_VECTOR(2 downto 0);
        semaforo_2 : out STD_LOGIC_VECTOR(2 downto 0)
    );
end prog8;

architecture prog8 of prog8 is
    type EstadoSemaforo is (Vermelho, Amarelo, Verde, AmareloOff, AmareloOn);
    signal estado_atual_1, proximo_estado_1 : EstadoSemaforo;
    signal estado_atual_2, proximo_estado_2 : EstadoSemaforo;
    constant FrequenciaMaquinaEstadoHz : integer := 1;
begin
    process(clk, reset)
        variable contador : integer := 0;
        variable contadorTempo : integer := 0;
        variable PassosMaquinaEstado : integer := FrequenciaClockHz / 2;
    begin
        if teste = '0' then
            PassosMaquinaEstado := FrequenciaClockHz;
            contadorTempo := 0;
        else
            contadorTempo := 3;
            PassosMaquinaEstado := FrequenciaClockHz / 2;
        end if;

        if reset = '0' then
            estado_atual_1 <= Vermelho;
            estado_atual_2 <= Verde;
            contador := 0;
        elsif rising_edge(clk) then
```

```
        if contador < PassosMaquinaEstado then
            contador := contador + 1;
        else
            estado_atual_1 <= proximo_estado_1;
            estado_atual_2 <= proximo_estado_2;
            contador := 0;
        end if;
    end if;
end process;

process(estado_atual_1, estado_atual_2)
    variable flag_1 : integer := 1;
    variable flag_2 : integer := 0;
begin
    if pause = '0' then
        case estado_atual_1 is
            when Vermelho =>
                if flag_1 = 1 then
                    proximo_estado_1 <= Verde;
                else
                    proximo_estado_1 <= Vermelho;
                end if;
            when Verde =>
                proximo_estado_1 <= Amarelo;
                flag_2 := 0;
            when Amarelo =>
                proximo_estado_1 <= Vermelho;
                flag_2 := 1;
            when others =>
                proximo_estado_1 <= Vermelho;
        end case;

        case estado_atual_2 is
            when Vermelho =>
                if flag_2 = 1 then
                    proximo_estado_2 <= Verde;
                else
                    proximo_estado_2 <= Vermelho;
                end if;
            when Verde =>
                proximo_estado_2 <= Amarelo;
                flag_1 := 0;
            when Amarelo =>
                proximo_estado_2 <= Vermelho;
                flag_1 := 1;
            when others =>
                proximo_estado_2 <= Verde;
        end case;
    else
        case estado_atual_1 is
```

```
                when AmareloOff =>
                    proximo_estado_1 <= AmareloOn;
                when others =>
                    proximo_estado_1 <= AmareloOff;
            end case;

            case estado_atual_2 is
                when AmareloOff =>
                    proximo_estado_2 <= AmareloOn;
                when others =>
                    proximo_estado_2 <= AmareloOff;
            end case;
        end if;
    end process;

    process(estado_atual_1)
    begin
        case estado_atual_1 is
            when Vermelho =>
                semaforo_1 <= "001";
            when Verde =>
                semaforo_1 <= "100";
            when Amarelo =>
                semaforo_1 <= "010";
            when AmareloOff =>
                semaforo_1 <= "000";
            when AmareloOn =>
                semaforo_1 <= "010";
        end case;
    end process;

    process(estado_atual_2)
    begin
        case estado_atual_2 is
            when Vermelho =>
                semaforo_2 <= "001";
            when Verde =>
                semaforo_2 <= "100";
            when Amarelo =>
                semaforo_2 <= "010";
            when AmareloOff =>
                semaforo_2 <= "000";
            when AmareloOn =>
                semaforo_2 <= "010";
        end case;
    end process;
end prog8;
```

3.2 Explicação do Código:

O código VHDL é implementado dentro de uma entidade chamada ‘prog8‘ e utiliza bibliotecas da IEEE para operações lógicas e aritméticas. A entidade ‘prog8‘ possui sinais de entrada para clock, reset, pause e teste, e sinais de saída para os dois semáforos (‘semaforo 1‘ e ‘semaforo 2‘). Um parâmetro genérico, ‘FrequenciaClockHz‘, define a frequência do clock usada no projeto (50 MHz por padrão).

Dentro da arquitetura ‘prog8‘, é declarado um tipo enumerado ‘EstadoSemaforo‘ que inclui os estados ‘Vermelho‘, ‘Amarelo‘, ‘Verde‘, ‘AmareloOff‘ e ‘AmareloOn‘. Sinais são usados para controlar os estados atuais e próximos dos semáforos (‘estado atual 1‘, ‘proximo estado 1‘, ‘estado atual 2‘, ‘proximo estado 2‘). A constante ‘FrequenciaMaquinaEstadoHz‘ é definida como 1 Hz.

O processo principal é sensível aos sinais de clock (‘clk‘) e reset (‘reset‘). Ele utiliza variáveis locais para controlar as transições de estado. Se o reset for ativado, os semáforos são configurados para ‘Vermelho‘ e ‘Verde‘, respectivamente. Durante cada borda de subida do clock, um contador é incrementado até atingir um valor de limite (‘PassosMaquinaEstado‘), momento em que os estados são atualizados.

O processo de transição de estados controla as mudanças dos semáforos entre os estados ‘Vermelho‘, ‘Verde‘ e ‘Amarelo‘ com base em condições específicas. Quando o sinal ‘pause‘ está em ’0’, os semáforos transitam normalmente entre os estados. Quando ‘pause‘ está ativado, os semáforos alternam entre os estados ‘AmareloOff‘ e ‘AmareloOn‘.

Dois processos separados controlam as saídas dos semáforos (‘semaforo 1‘ e ‘semaforo 2‘) com base nos estados atuais. Para cada estado, o valor apropriado é atribuído às saídas, representando as luzes ‘Vermelho‘, ‘Verde‘ e ‘Amarelo‘. Em resumo, o código implementa uma lógica de controle de semáforos que alterna entre os estados de forma a simular o funcionamento real de um cruzamento, incluindo modos de pausa e teste para maior flexibilidade.

4 Resultados Obtidos

Diagrama RTL gerado pelo Quartus Prime de acordo com o codigo VHDL.

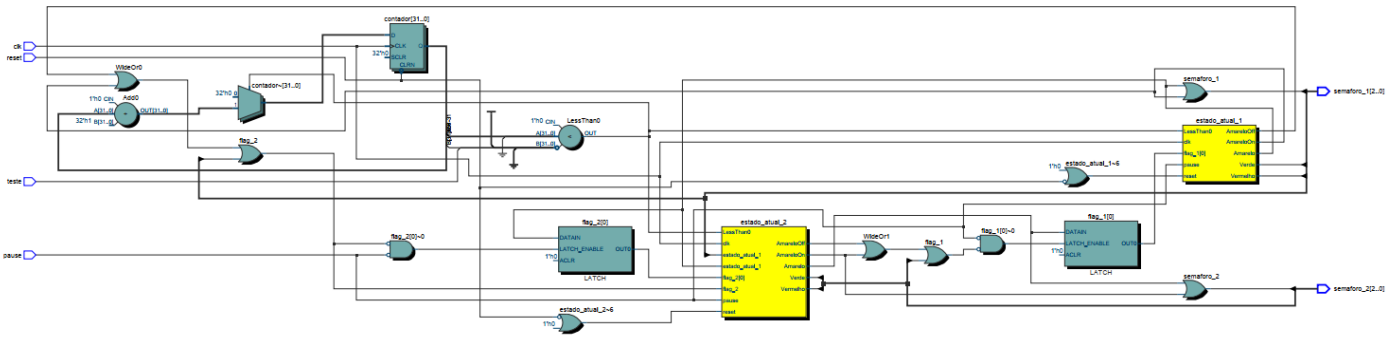

Figura 2: Diagrama RTL

Figura 3: Placa em StandBy

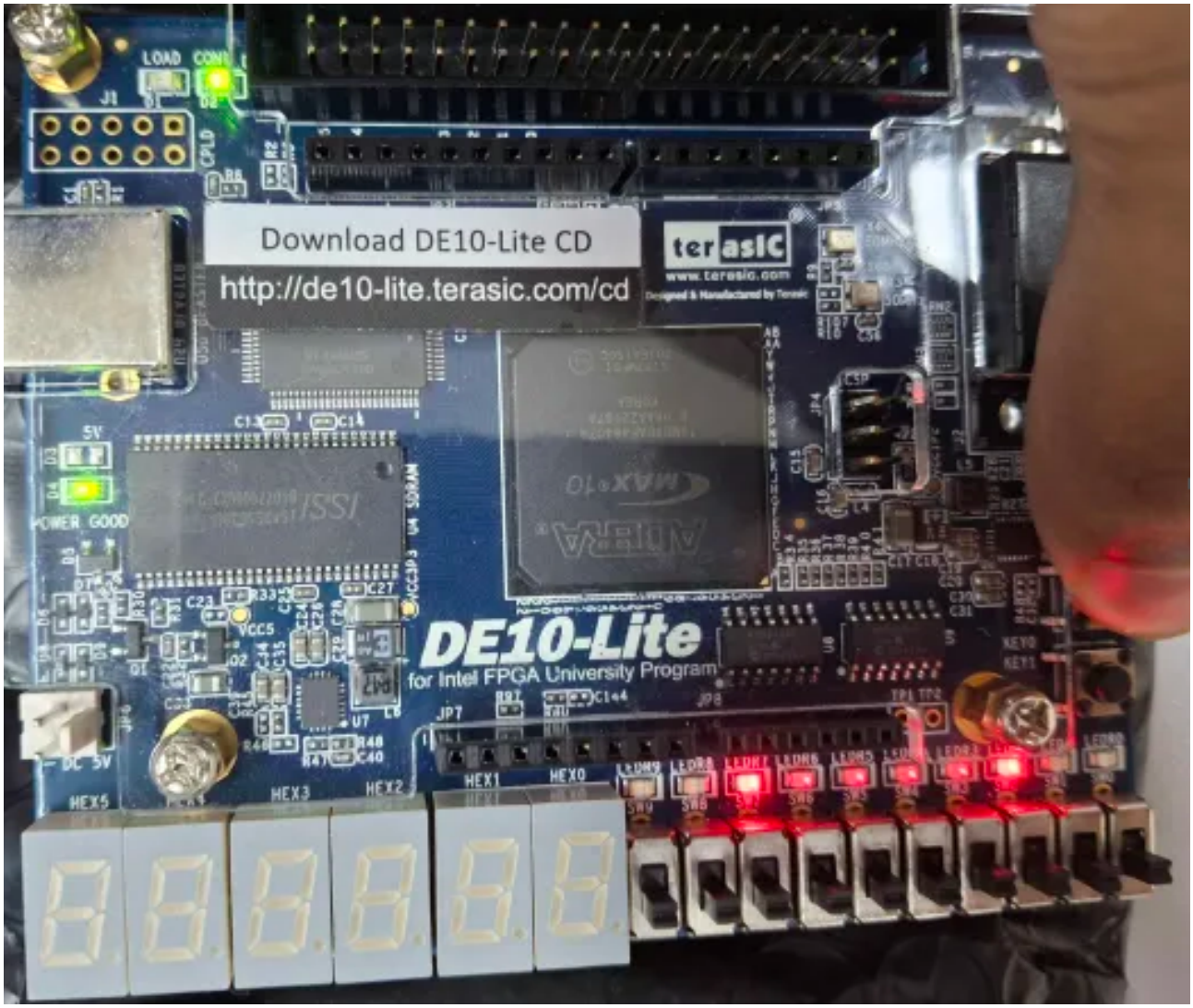

Figura 4: Semafaro 1 "verde- Semafaro 2 vermelho