# Experiência *7*
# *Semáforos de Trânsito I*

| | |
|---|---|
| **Disciplina:** PCS 3335 – Laboratório Digital A | |
| **Prof.:** *Bruno Albertini* | **Data:** *30/04/2020* \| *25/06/2020* |
| **Turma:** *10* | **Bancada:** *B4* |
| **Membros:** | |
| *10696510 Antonio Nigro Zamur* | |
| *11027484 Guilherme Akira Alves dos Santos* | |

## 1. Introdução

Por meio de uma lógica sequencial e uma lógica combinatória, nessa experiência irá se estruturar um sistema de semáforos de trânsito.

## 2. Objetivo

Essa atividade visa praticar os conceitos de fluxo de dados, unidade de controle, máquina de estados e diagrama de transição de estados enquanto é desenvolvido um projeto de sistema digital.

## 3. Planejamento

a. Projeto

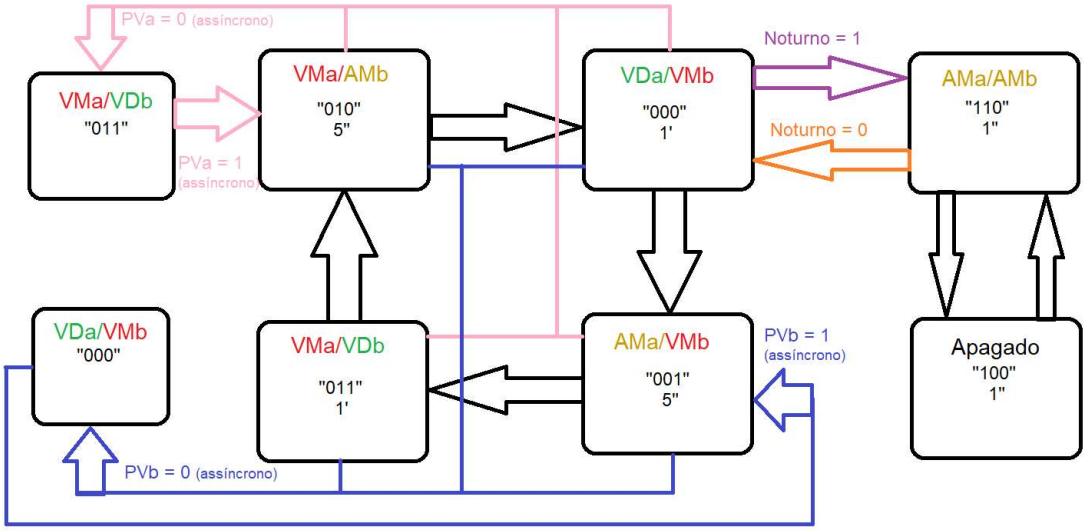
i. Descrição Funcional

Tabela 1 - Tabela verdade para o bloco de lógica combinatória

| Dec | $Q_2$ | $Q_1$ | $Q_0$ | VMa | AMa | VDa | VMb | AMb | VDb |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | X | X | X | X | X | X |
| 6 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 7 | 1 | 1 | 1 | X | X | X | X | X | X |

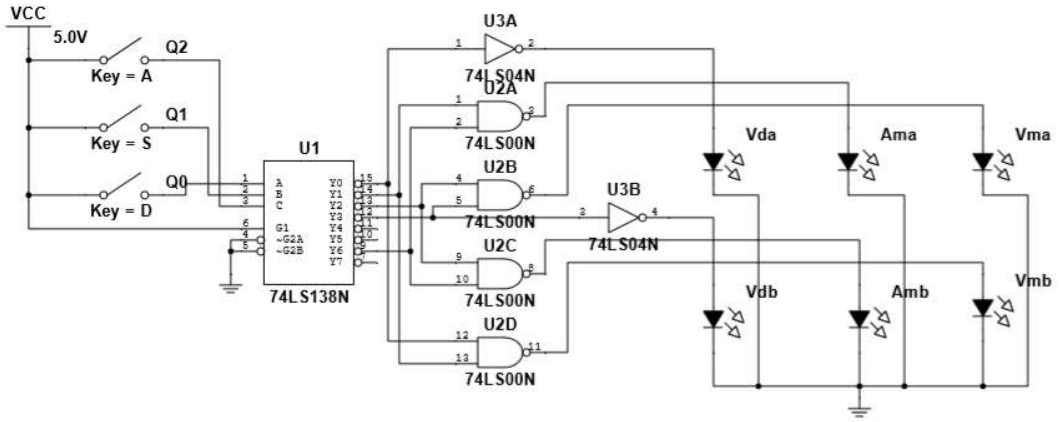
ii. Diagrama de Blocos

Figura 1 - Transição de estados da unidade de controle:




iii. Diagrama Lógico

Figura 2 - Diagrama Lógico do circuito combinatório




| Pos | Comp | Vcc | GND |
|---|---|---|---|
| U1 | 74138 | 16 | 8 |
| U2x | 7400 | 14 | 7 |
| U3x | 7404 | 14 | 7 |

Disciplina: PCS 3335 - Laboratório Digital A
Prof: Bruno Albertini Data: 30/04/2020
Projeto: Lógica Combinatória (Semáforos 1)
Turma: 10 Bancada: B4
Membros:
10696510 Antonio Nigro Zamur
11027484 Guilherme Akira Alves dos Santos

Figura 3 - Código VHDL da lógica sequencial

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity logica_sequencial is
    generic(
        short_delay : integer := 5; -- valor em segundos
        long_delay  : integer := 60 -- valor em segundos
    );
    port(
        pva, pvb, noturno, reset: in std_logic;
        clock: in std_logic;
        q: out std_logic_vector (2 downto 0)
    );
end logica_sequencial;

architecture arch_sequencial of logica_sequencial is

    signal troca_dia  :  std_logic := '0';
    signal Eatual   :  std_logic_vector (2 downto 0) := "000";
    signal seg_count  :  integer range 0 to 60 := 0;

begin

    Counter: process (reset, clock, noturno, Eatual, pva, pvb) is
    begin
        -- reset assincrono
        if reset = '1' then
            Eatual <= "100";
            seg_count <= 0;

        -- suspende contagem de tempo quando sem carro em uma via
        elsif noturno = '0' and pva = '0' and Eatual = "011" then
            seg_count <= 0;
        elsif noturno = '0' and pvb = '0' and Eatual = "000" then
            seg_count <= 0;

        -- quando ha subida de clock
        elsif clock'event and clock = '1' then
            seg_count <= seg_count + 1;

            -- transicoes noturnas
            if noturno = '1' and Eatual = "000" then
                Eatual <= "110";
                seg_count <= 0;
            elsif noturno = '1' and Eatual = "110" then
                Eatual <= "100";
                seg_count <= 0;
            elsif noturno = '0' and Eatual = "110" then
                Eatual <= "000";
                seg_count <= 0;
            elsif Eatual = "100" then
                Eatual <= "110";
                seg_count <= 0;

            -- transicoes diurnas
            elsif seg_count = (short_delay - 1) and Eatual = "001" then
                Eatual <= "011";
                seg_count <= 0;
            elsif seg_count = (short_delay - 1) and Eatual = "010" then
                Eatual <= "000";
                seg_count <= 0;
            elsif seg_count = (long_delay - 1) and Eatual = "000" then
                Eatual <= "001";
                seg_count <= 0;
            elsif seg_count = (long_delay - 1) and Eatual = "011" then
                Eatual <= "010";
                seg_count <= 0;
            end if;
        end if;
    end process;

    q <= Eatual;
end arch_sequencial;
```

Figura 4 - Código VHDL do circuito temporizador

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity circuito_temporizador is
    generic(
        freq_in  : natural := 50E6 -- valor dado em Hertz
    );
    port(
        clock_in, reset: in std_logic;
        clock_out: out std_logic
    );
end circuito_temporizador;

architecture arch_temporizador of circuito_temporizador is

    signal count    :  natural range 0 to freq_in := 0;
    signal clock_aux  :  std_logic := '0';

begin

    Counter: process (reset, clock_in) is
    begin
        -- reset assincrono
        if reset = '1' then
            count <= 0;
            clock_aux <= '0';

        -- borda de subida de clock
        elsif clock_in'event and clock_in = '1' then
            -- determina a saida do clock
            if count = (freq_in - 1) then
                count <= 0;
                clock_aux <= '0';
            elsif count >= ((freq_in / 2) - 1) then
                clock_aux <= '1';
                count <= count + 1;
            else
            clock_aux <='0';
                count <= count + 1;
            end if;

        end if;
    end process;

    clock_out <= clock_aux;
end arch_temporizador;
```

Figura 5 - Código VHDL da unidade de controle

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity unidade_controle is
    port(
        reset: in std_logic; -- sinal comum aos modulos
        pva, pvb, noturno: in std_logic; -- sinais do logica_sequencial
        clock: in std_logic; -- sinais do circuito_temporizador
        q: out std_logic_vector (2 downto 0)   -- estado
    );
end unidade_controle;

architecture arch_controle of unidade_controle is

    -- temporizador que converte a freq_in em 1 Hz
    component circuito_temporizador is
        generic(
            freq_in  : natural := 50E6 -- valor em Hertz
        );
        port(
            clock_in, reset: in std_logic;
            clock_out: out std_logic
        );
    end component circuito_temporizador;

    -- logica de transicao de estados
    component logica_sequencial is
        generic(
            short_delay : integer := 5; -- valor em segundos
            long_delay  : integer := 60 -- valor em segundos
        );
        port(
            pva, pvb, noturno, reset: in std_logic;
            clock: in std_logic;
            q: out std_logic_vector (2 downto 0)
        );
    end component logica_sequencial;

    -- sinal auxiliar para o port map
    signal clock_1hz : std_logic;

begin

    -- port map do circuito temporizador
    tempo:   circuito_temporizador port map (clock, reset, clock_1hz);
    -- port map da logica sequencial
    seq:     logica_sequencial port map (pva, pvb, noturno, reset, clock_1hz, q);

end arch_controle;
```

iv. Simulações

Figura 6 - Carta de tempo circuito combinatório

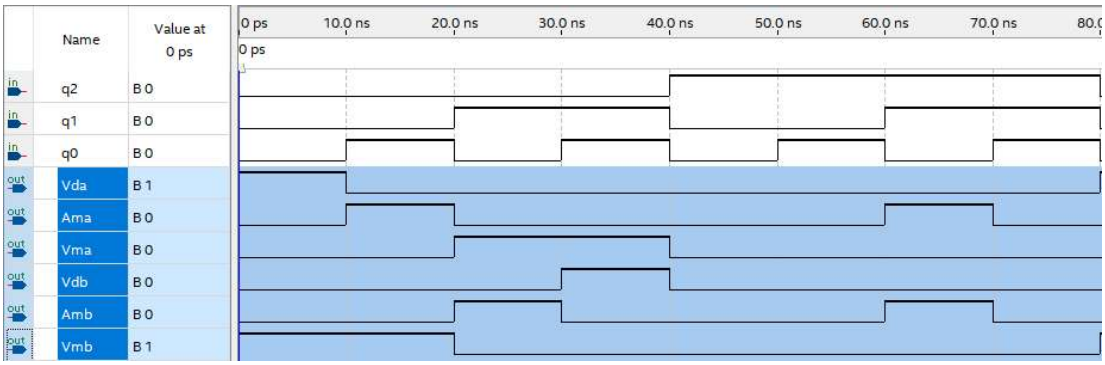


Figura 7.1 - Carta de tempo circuito sequencial: Ciclo completo

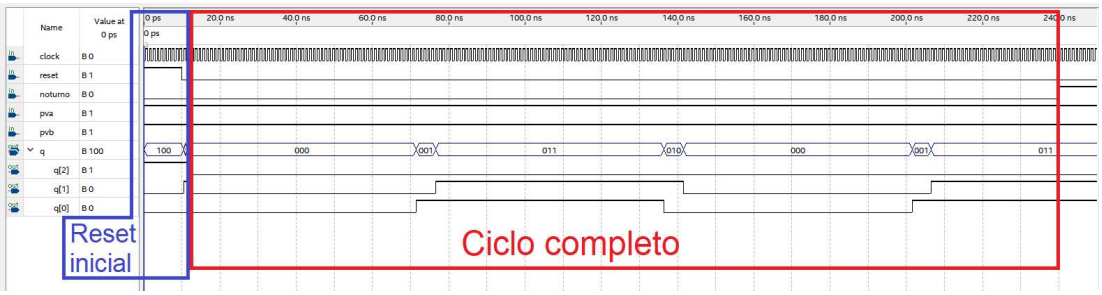



Figura 7.2 - Carta de tempo circuito sequencial: Noturno


Figura 7.3 - Carta de tempo circuito sequencial: PVa e PVb

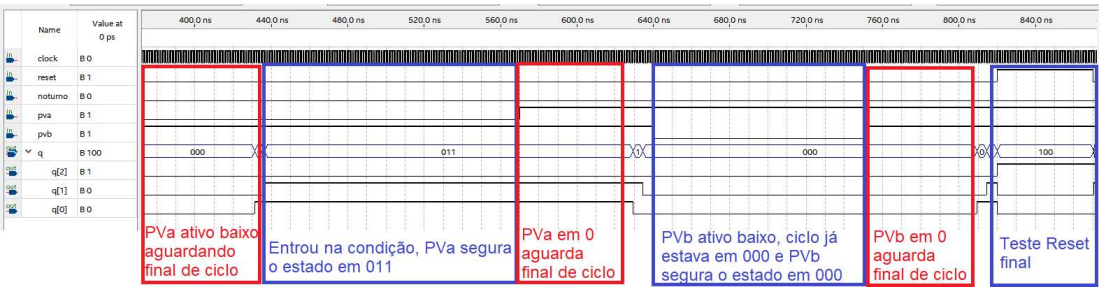
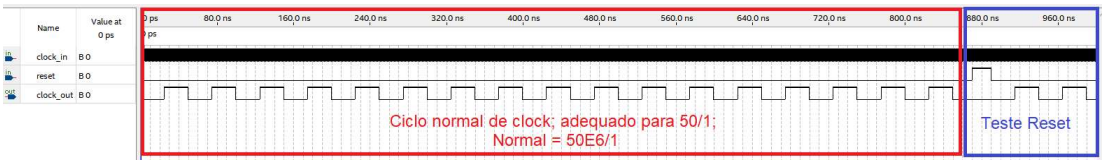



Figura 8 - Carta de tempo circuito temporizador




b. Estratégias de montagens, testes e depuração

i. Estratégia de Montagem

Para a montagem do bloco de lógica combinatória, seguiremos conforme a Figura 2, ligando o bloco de lógica sequencial nos inputs do decodificador e depois conectando as portas NAND e NOT. Posteriormente serão conectados os correspondentes LEDs.

O código unidade_controle será sintetizado e carregado na placa FPGA. Os sinais de entrada serão ligados aos devidos pinos assertados na Tabela 2. E os pinos de saída da FPGA serão ligados à entrada do bloco combinatório previamente montado.

Tabela 2 - Tabela de pinagem unidade_controle

| Variável | Tipo | Porta | Pinagem |
|---|---|---|---|
| Clock | Input | CLOCK_50 | PIN_M9 |
| Reset | Input | SW0 | PIN_U13 |
| Noturno | Input | SW1 | PIN_V13 |
| PVa | Input | SW2 | PIN_T13 |
| PVb | Input | SW3 | PIN_T12 |
| semaforos[0] | Output | LEDR0 | PIN_AA2 |
| semaforos[1] | Output | LEDR1 | PIN_AA1 |
| semaforos[2] | Output | LEDR2 | PIN_W2 |
| semaforos[3] | Output | LEDR3 | PIN_Y3 |
| semaforos[4] | Output | LEDR4 | PIN_N2 |
| semaforos[5] | Output | LEDR5 | PIN_N1 |

ii. Testes e Depuração

Para testar o bloco de lógica combinatória, serão averiguados os valores das saídas do decodificador. Comprovado seu funcionamento, serão posicionadas as portas NAND e em seguida estas serão testadas. O mesmo se aplica para a disposição das portas NOT.

Após confirmado o funcionamento dos componentes, serão conectados *switches* às entradas do decodificador. Com estes, ao chaveá-los, será

comparado o resultado obtido com a tabela verdade esperada (vide Tabela 1).

Em seguida, ao acoplar o circuito da FPGA ao circuito combinatório, serão testados alguns cenários que se assemelham a uma situação da vida real, conferindo se o resultado do circuito remete ao esperado.

4. ## Relatório

   a. Resultados obtidos

   b. Pontos positivos e negativos

   c. Lições aprendidas

## Desafio

## Referências

Apostilas e documentos de apoio do site ~/labdig do PCS.
Apostilas disponíveis na plataforma e-Disciplinas.
Tópicos da disciplina Sistemas Digitais 1.