Computação Digital

ENG 1448

Laboratório 02

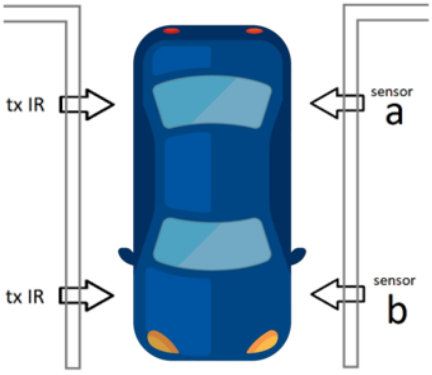
Objetivos:

- Familiarização com process
- Implementar uma máquina de estado
- Entender na prática a necessidade do tratamento de debounce

Primeira parte do laboratório.

Imagine que você seja chamado para propor uma solução de um problema comum em um estacionamento. Ainda que haja cancelas e um sistema de entrada/saída de veículos, o gerente do estacionamento reclama que volta e meia ocorre algum problema mecânico (a cancela trava, o cartão engasga e não sai...), e eles são obrigados a repetirem o processo – o que dificulta o controle de quantas vagas estão de fato ocupadas.

O estacionamento possui apenas uma porta de entrada e de saída, então 2 pares de sensores infravermelhos serão usados, de forma que seja possível saber se o carro está entrando ou saindo.

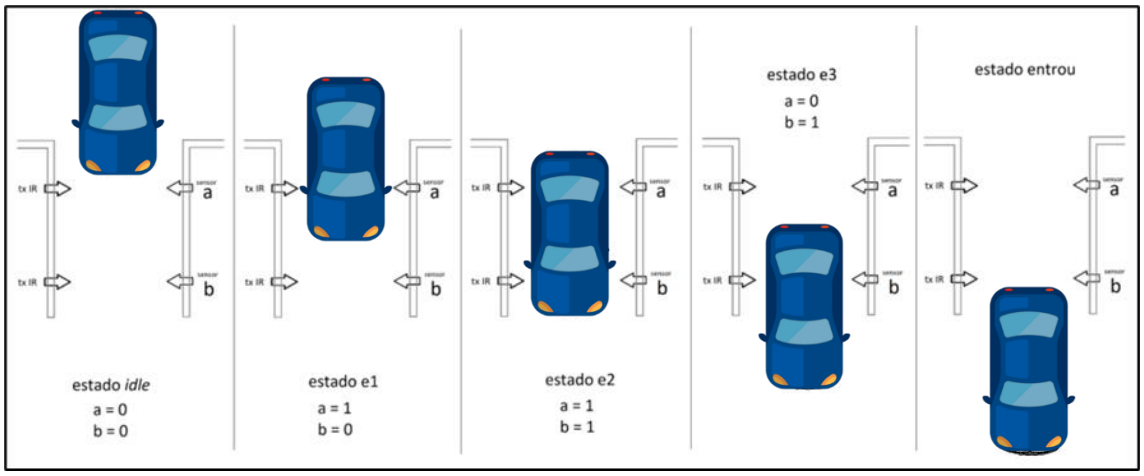


Os sensores possuem leitura “0” quando expostos ao infravermelho, o que significa dizer que vão a “1” enquanto o feixe é interrompido.

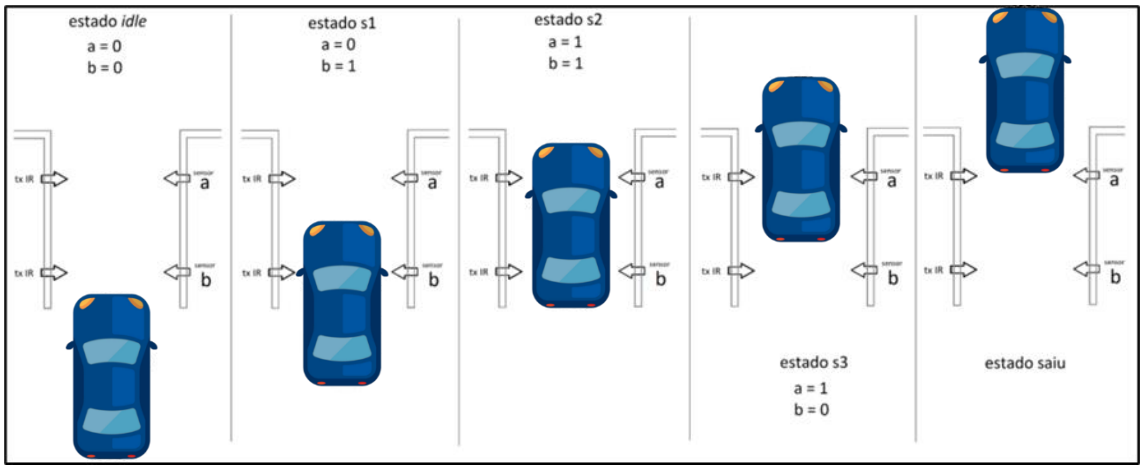
A máquina de estado se faz fundamental para:

- ✓ Se identificar se o veículo está entrando ou saindo;
- ✓ O sistema deve ser imune a uma pessoa que entre (que não bloqueará os 2 sensores ao mesmo tempo!)

Assim, temos o seguinte funcionamento para a entrada...

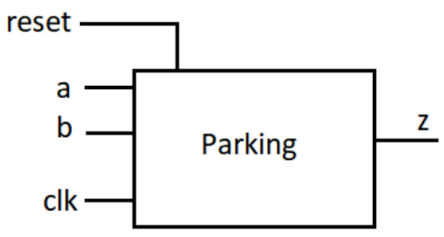


E, respectivamente, para a saída teremos....



A criação de estados é livre. A dupla (ou você, se estiver sozinho) deverá avaliar os estados necessários para o correto funcionamento.

A entidade deverá ter a seguinte cara...



A saída Z deverá dar a quantidade de veículos (nossa plaquinha possui 8 LEDs, portanto o tamanho do vetor Z será...). Não há necessidade, neste exercício, de se tratar números negativos – até mesmo porque não há como sair mais carros do que entrou.

Tenha liberdade de iniciar Z com um número baixo qualquer (2, 3 ou 4), de forma que o teste possa começar tanto com a simulação de um carro entrando, quanto saindo (isso não é importante para o testbench, mas é para o funcionamento da plaquinha).

A saída deverá ser conectada aos LEDs discretos, mostrando de uma *forma inteligível* para o rapaz que estará de plantão no estacionamento (resumidamente: o trabalhador não sabe binário, ok?).

Parte 1

1. Desenhe o projeto da máquina de estados
2. Implemente com VHDL
3. Simule usando o testbench fornecido

Parte 2

4. Implemente na plaquinha. De preferência, use push-buttons.
   Consulte o User Guide.

Se algo não funcionou, tente descobrir o motivo.

Anexo: Testbench

```
--------------------------------------------------------------------------------
-- Lab 02
-- Parking
-- Testbench
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY Parking_TB IS
END Parking_TB;

ARCHITECTURE behavior OF Parking_TB IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT Parking
        PORT(
            clk   : IN  std_logic;
            reset : IN  std_logic;
            a     : IN  std_logic;
            b     : IN  std_logic;
            z     : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;

    --Inputs
    signal clk   : std_logic := '0';
    signal reset : std_logic := '0';
    signal a     : std_logic := '0';
    signal b     : std_logic := '0';

    --Outputs
    signal z     : std_logic_vector(7 downto 0);

    -- Clock period definitions
    constant clk_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: Parking PORT MAP (
        clk   => clk,
        reset => reset,
        a     => a,
        b     => b,
        z     => z
    );

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


    -- Stimulus process
    stim_proc: process
    begin
        reset <='1';
        wait for clk_period*10;
        reset <='0';
        wait for clk_period*10;

        a <='0'; b <='0'; wait for clk_period*1;
        a <='1'; b <='0'; wait for clk_period*1;
        a <='1'; b <='1'; wait for clk_period*1;
        a <='0'; b <='1'; wait for clk_period*1;
        a <='0'; b <='0'; wait for clk_period*1;
        -- 1o carro entrou

        wait for clk_period*10;
        a <='0'; b <='0'; wait for clk_period*5;
        a <='1'; b <='0'; wait for clk_period*5;
        a <='1'; b <='1'; wait for clk_period*5;
        a <='0'; b <='1'; wait for clk_period*5;
        a <='0'; b <='0'; wait for clk_period*5;
        -- 2o carro entrou

        wait for clk_period*10;
        a <='0'; b <='0'; wait for clk_period*5;
        a <='0'; b <='1'; wait for clk_period*5;
        a <='1'; b <='1'; wait for clk_period*5;
        a <='1'; b <='0'; wait for clk_period*5;
        a <='0'; b <='0'; wait for clk_period*5;
        -- 1o carro saiu

        wait for clk_period*10;
        a <='0'; b <='0'; wait for clk_period*1;
        a <='0'; b <='1'; wait for clk_period*1;
        a <='1'; b <='1'; wait for clk_period*1;
        a <='1'; b <='0'; wait for clk_period*1;
        a <='0'; b <='0'; wait for clk_period*1;
        -- 2o carro saiu

        wait;
    end process;
END;
```