

# Relatório - Atividade: Semáforos

Maria Eduarda Soares Romana & Tiago Fernandes Soucek

2 de julho de 2025

## 1 Resumo

Esta atividade teve como objetivo o desenvolvimento de um sistema de controle para dois semáforos em VHDL, projetado para implementação em uma placa FPGA. O sistema gerencia o ciclo de luzes (verde, amarela, vermelha, onde LEDs do FPGA representam as luzes) de forma sincronizada e segura, garantindo que as vias não entrem em conflito.

O desenvolvimento do projeto abrangeu as seguintes funcionalidades e componentes principais:

1. **Controle por Máquina de Estados:** O núcleo do sistema é uma máquina de estados finitos que define os ciclos de operação para cada semáforo.
2. **Operação Sincronizada e Segura:** A lógica garante que os estados dos dois semáforos sejam opostos. Quando um semáforo está no estado verde, o outro permanece no vermelho. A transição para amarelo em um semáforo também mantém o outro em vermelho para segurança.
3. **Modos de Operação:** O sistema inclui três modos distintos de funcionamento, selecionáveis por entradas externas:
  - **Modo Normal:** Os semáforos operam com tempos pré-definidos para as luzes.
  - **Modo de Teste:** Funciona de maneira idêntica ao modo normal, porém com tempos de ciclo reduzidos para verificação rápida.
  - **Modo de Standby:** Um modo de alerta onde ambos os semáforos piscam a luz amarela de forma intermitente.

## 2 Requisitos do Projeto

Os seguintes itens foram os requisitos para o desenvolvimento do projeto:

1. Implementar dois semáforos utilizando máquinas de estado, com cada semáforo alternando entre as luzes verde, amarela e vermelha.
2. O segundo semáforo deve operar de forma inversa ao primeiro: vermelho quando o outro está verde e vice-versa.
3. Garantir que, quando um semáforo estiver em amarelo, o outro permaneça em vermelho.
4. Implementar um modo de **standby**, no qual ambos os semáforos piscam a luz amarela.
5. Implementar um modo de **teste**, que executa o ciclo normal de operação de forma acelerada.

### 3 Métodos e Técnicas

A implementação foi realizada no software Quartus Prime, e a Figura 1 apresenta o código VHDL desenvolvido para resolver o problema proposto.

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity aula8 is
6      Port (
7          clk          : in  STD_LOGIC;
8          reset        : in  STD_LOGIC;
9          modo_standby : in  STD_LOGIC;
10         modo_teste   : in  STD_LOGIC;
11         -- Sem foro 1
12         v1           : out STD_LOGIC;
13         a1           : out STD_LOGIC;
14         r1           : out STD_LOGIC;
15         -- Sem foro 2
16         v2           : out STD_LOGIC;
17         a2           : out STD_LOGIC;
18         r2           : out STD_LOGIC
19     );
20 end aula8;
21
22 architecture Behavioral of aula8 is
23
24     type estado_tipo is (S1_VERDE, S1_AMARELO, S2_VERDE, S2_AMARELO);
25     signal estado_atual : estado_tipo := S1_VERDE;
26
27     signal contador : unsigned(25 downto 0) := (others => '0');
28
29     -- Temporiza es
30     constant TEMPO_VERDE      : unsigned(25 downto 0) := to_unsigned(250000000,
31         26);
32     constant TEMPO_AMARELO    : unsigned(25 downto 0) := to_unsigned(100000000,
33         26);
34     constant TEMPO_TESTE_VERDE : unsigned(25 downto 0) := to_unsigned
35         (10000000, 26);
36     constant TEMPO_TESTE_AMARELO : unsigned(25 downto 0) := to_unsigned
37         (5000000, 26);
38     signal tempo_limite : unsigned(25 downto 0);
39
40 begin
41
42     process(clk, reset)
43     begin
44         if reset = '1' then
45             estado_atual <= S1_VERDE;
46             contador <= (others => '0');
47         elsif rising_edge(clk) then
48             if modo_standby = '1' then
49                 contador <= contador + 1; -- Deixa o contador correr para o
50                     pisca-pisca
51             else
52                 -- Define o tempo limite com base no modo e no estado atual
53                 if modo_teste = '1' then

```

```

49         case estado_atual is
50             when S1_VERDE | S2_VERDE => tempo_limite <=
                    TEMPO_TESTE_VERDE;
51             when S1_AMARELO | S2_AMARELO => tempo_limite <=
                    TEMPO_TESTE_AMARELO;
52         end case;
53     else -- Modo Normal
54         case estado_atual is
55             when S1_VERDE | S2_VERDE => tempo_limite <=
                    TEMPO_VERDE;
56             when S1_AMARELO | S2_AMARELO => tempo_limite <=
                    TEMPO_AMARELO;
57         end case;
58     end if;
59
60     -- Mquina de estado principal
61     if contador >= tempo_limite then
62         contador <= (others => '0');
63         case estado_atual is
64             when S1_VERDE => estado_atual <= S1_AMARELO;
65             when S1_AMARELO => estado_atual <= S2_VERDE;
66             when S2_VERDE => estado_atual <= S2_AMARELO;
67             when S2_AMARELO => estado_atual <= S1_VERDE;
68         end case;
69     else
70         contador <= contador + 1;
71     end if;
72 end if;
73 end if;
74 end process;
75
76 -- Lógica de Saída para os LEDs
77 process(estado_atual, modo_standby, contador)
78 begin
79     -- Estado padrão (vermelho)
80     v1 <= '0'; a1 <= '0'; r1 <= '1';
81     v2 <= '0'; a2 <= '0'; r2 <= '1';
82
83     if modo_standby = '1' then
84         r1 <= '0'; r2 <= '0';
85         if contador(24) = '1' then -- Pisca amarelo
86             a1 <= '1'; a2 <= '1';
87         else
88             a1 <= '0'; a2 <= '0';
89         end if;
90     else
91         case estado_atual is
92             when S1_VERDE =>
93                 v1 <= '1'; r1 <= '0';
94                 r2 <= '1';
95             when S1_AMARELO =>
96                 a1 <= '1'; r1 <= '0';
97                 r2 <= '1';
98             when S2_VERDE =>
99                 r1 <= '1';
100                v2 <= '1'; r2 <= '0';
101             when S2_AMARELO =>
102                 r1 <= '1';

```

```

103             a2 <= '1'; r2 <= '0';
104         end case;
105     end if;
106 end process;
107
108 end Behavioral;

```

Listing 1: Código VHDL: Controle de Semáforos

## 4 Resultados e Considerações

Na Figura 1, é possível visualizar o esquema lógico (RTL) gerado pelo Quartus Prime após a compilação do código VHDL.

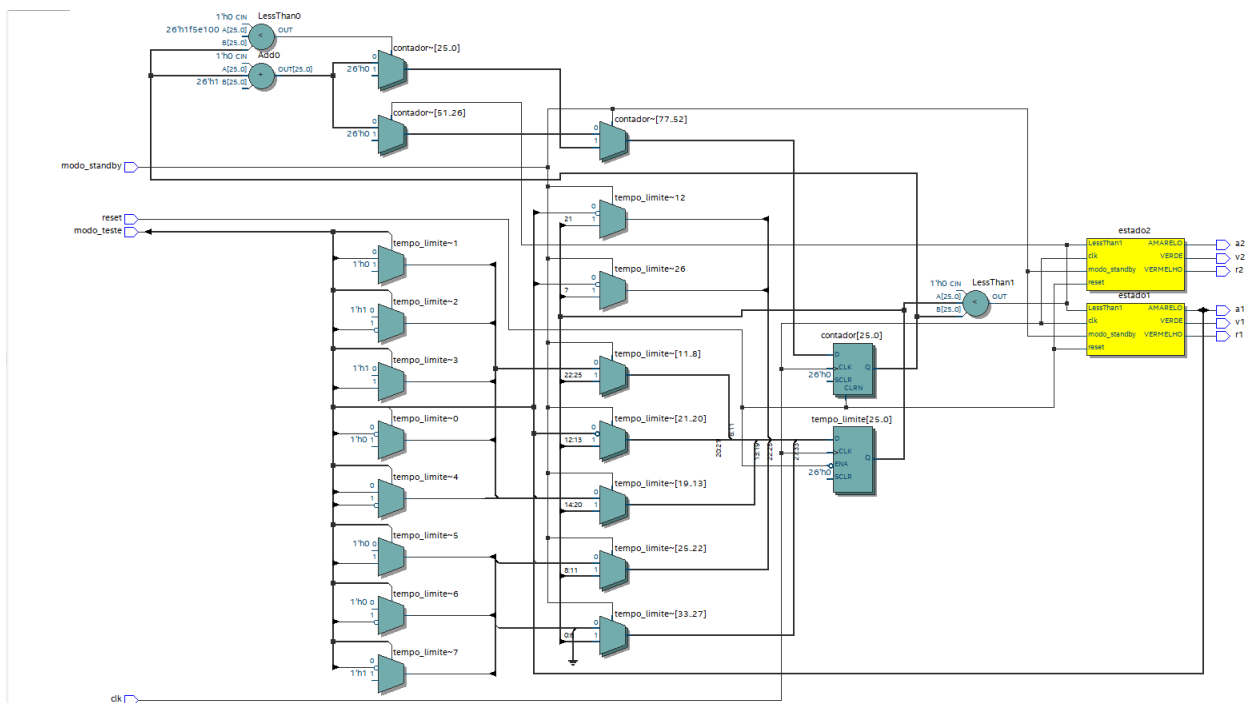


Figura 1: Esquema da distribuição das portas.

Fonte: Autoria própria.

Neste **LINK** (clique aqui) há o vídeo demonstrando o funcionamento do projeto na placa.

## 5 Conclusão

A implementação do sistema de controle de semáforos foi concluída com sucesso. Todos os requisitos funcionais, incluindo os modos de operação normal, teste e standby, foram atendidos conforme o planejado. Os testes de simulação e a implementação prática na placa FPGA confirmaram o correto funcionamento do circuito.