Projeto de Semáforos de Trânsito

Experimento #9

PCS3335 - Laboratório Digital A

30/06/2022

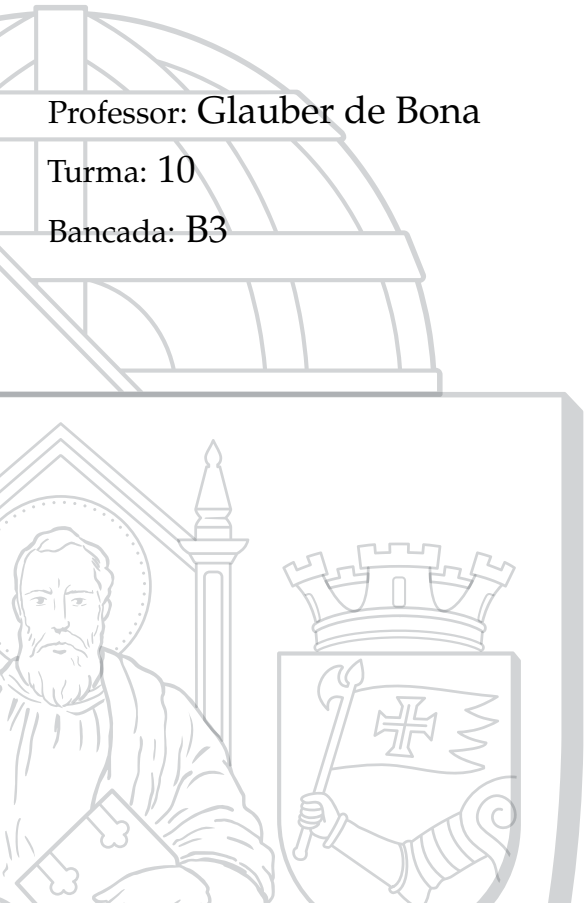
Natanael Magalhães Cardoso, 8914122

Renato Naves Fleury, 11805269

Professor: Glauber de Bona

Turma: 10

Bancada: B3



Universidade de São Paulo

Escola Politécnica

Departamento de Eng. de Computação e Sistemas Digitais

UNIVERSIDADE DE SÃO PAULO

ESCOLA POLITÉCNICA

Departamento de Eng. de Computação e Sistemas Digitais

Projeto de Semáforos de Trânsito

Natanael Magalhães Cardoso, Renato Naves Fleury

1. INTRODUÇÃO

Uma FPGA é um circuito integrado criado para ser configurado por um projetista após sua fabricação com uma linguagem de descrição de hardware. Neste projeto, será usada uma FPGA na implementação da unidade de controle de um circuito digital controlador de semáforos. Já o fluxo de dados será implementado utilizando circuitos integrados não programáveis.

2. OBJETIVOS

O objetivo deste projeto é construir um circuito digital para controle de um sistema de semáforos em um cruzamento de duas vias de trânsito com as seguintes características:

- controle de 2 semáforos (S_0 e S_1) com 6 transições de luzes no total
- tempo da duração de cada transição de sinal ajustável durante o funcionamento do semáforo (valor mínimo: 2, valor máximo: 4, ou seja 3 possibilidades)
- mesma duração para sinais iguais em semáforos diferentes

3. PLANEJAMENTO

3.1. DESCRIÇÃO FUNCIONAL

Este circuito digital foi projetado para controlar os estados de dois semáforos em um cruzamento. A Fig. 1 mostra a transição de todos os estados dos semáforos S_0 e S_1 com os respectivos tempos de transição entre cada estado, T_0, T_1 e T_2 que podem ser programados pelo usuário final. Em que T_0 é a duração do sinal verde/vermelho, T_1 é a duração do sinal amarelo/vermelho e T_2 é a duração do sinal vermelho/vermelho. Este tempo é um valor que pode ser configurado no momento da execução do circuito e é projetado para ter um valor entre 2 e 4, ou seja, o tempo de duração, na realidade, será o tempo programado acrescido de 1. Isto é, este circuito não foi projetado para "pular" estados.

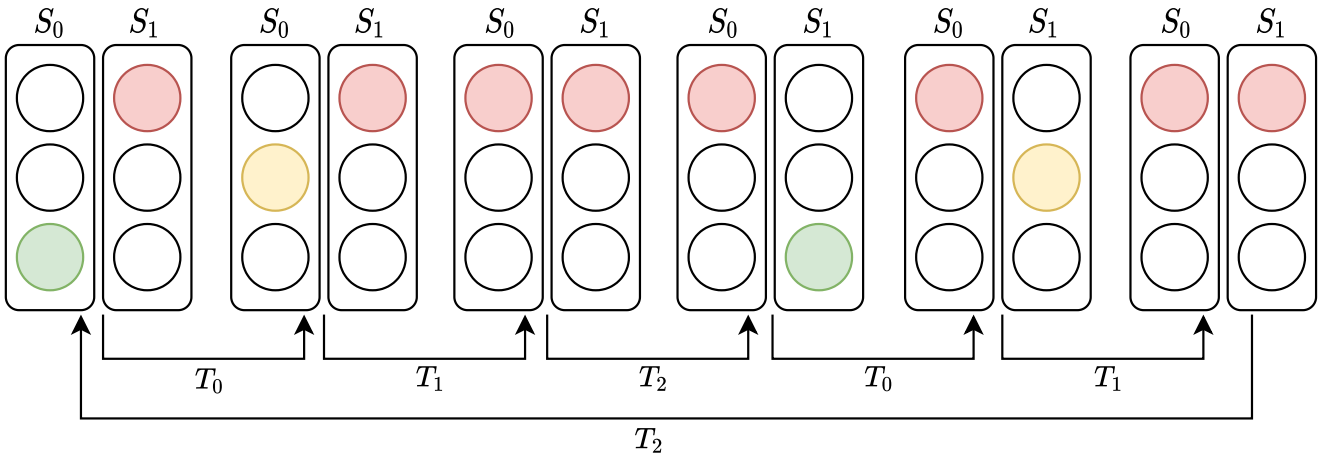


Figura 1: Transição de luzes dos semáforos S_0 e S_1 com seus respectivos tempos de transição

A Fig. 2 apresenta um esquema da disposição dos semáforos no cruzamento, indicando que o semáforo S_0 controla o fluxo Sul-Norte e o semáforo S_1 controla o fluxo Leste-Oeste.

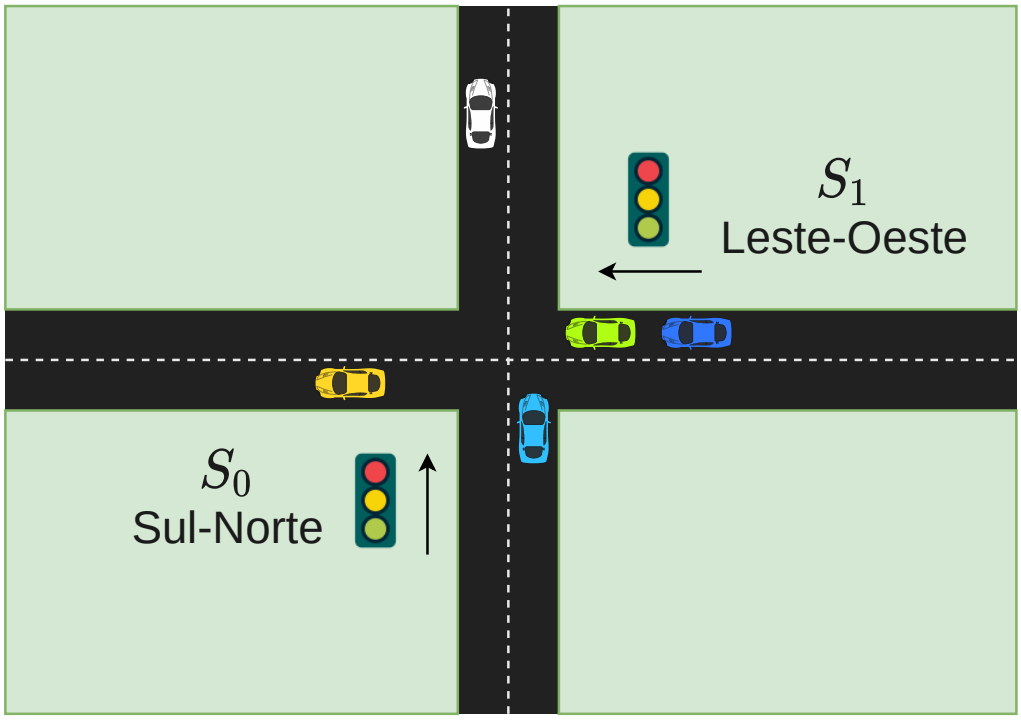


Figura 2: Esboço da disposição dos semáforos no cruzamento indicando a orientação do fluxo de veículos controlado por cada semáforo

3.2. Diagrama de Blocos

A Fig. 3 mostra o diagrama de blocos do circuito controlador de semáforos. O cirucito recebe os sinais de clock, reset e dos tempos de cada sinal e obtêm como saída um sinal de ativação para cada lâmpada dos dois semáforos. O Fluxo de Dados é composto por apenas um contador de 4 bits 74190, um multiplexador (74153) e a Unidade de Controle controla esse contador a partir do sinal load e os sinais de dados.

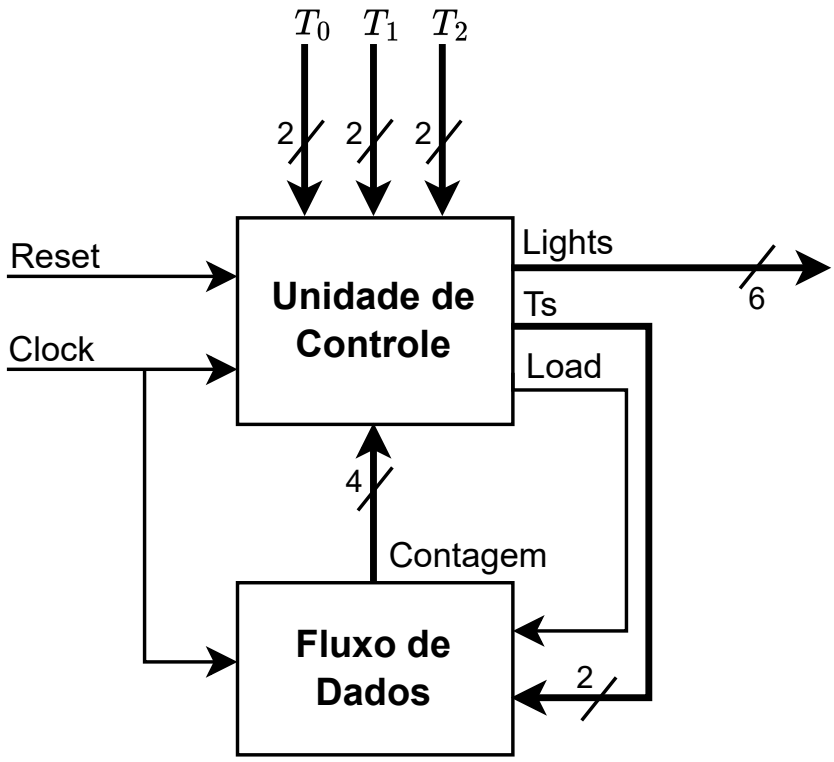


Figura 3: Diagrama de blocos do circuito controlador de semáforo

3.3. Diagrama Lógico

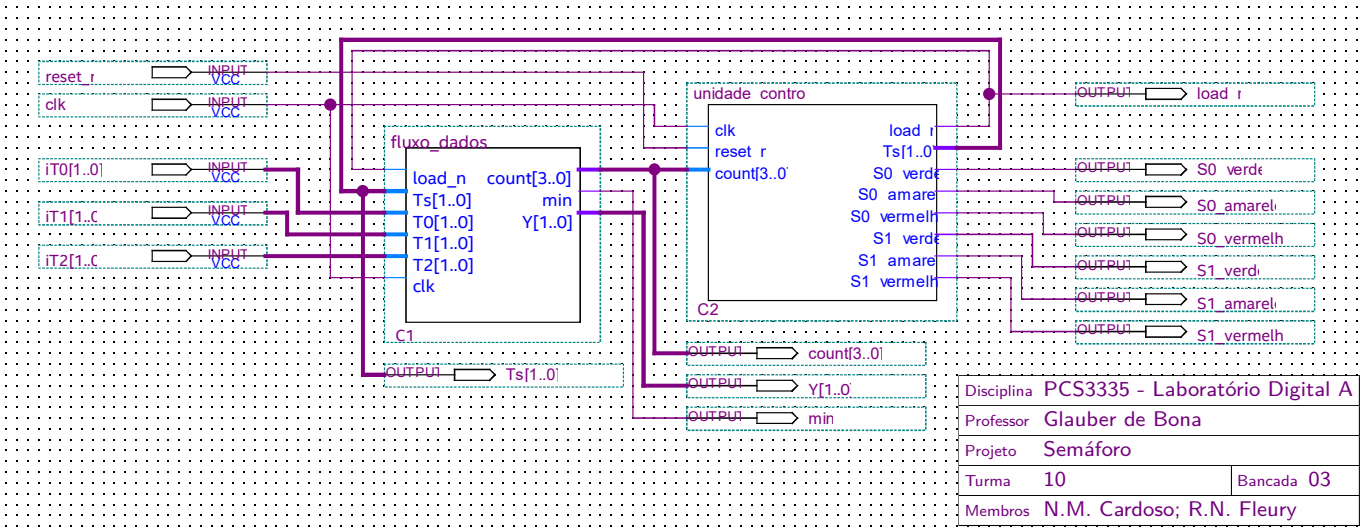


Figura 4: Diagrama lógico do circuito completo, incluindo Fluxo de Dados e Unidade de Controle

A Fig. 4 mostra a implementação do circuito digital, que é constituído de dois blocos prinicipais: o fluxo de dados e a unidade de controle. O fluxo de dados é composto por um Contador de 4 bits 74190 e um multiplexador 74153, cujo diagrama lógico é mostrado na Fig. 5 e a unidade de controle é uma máquina de estados implementada em VHDL, cuja descrição é mostrada na Listagem 1.

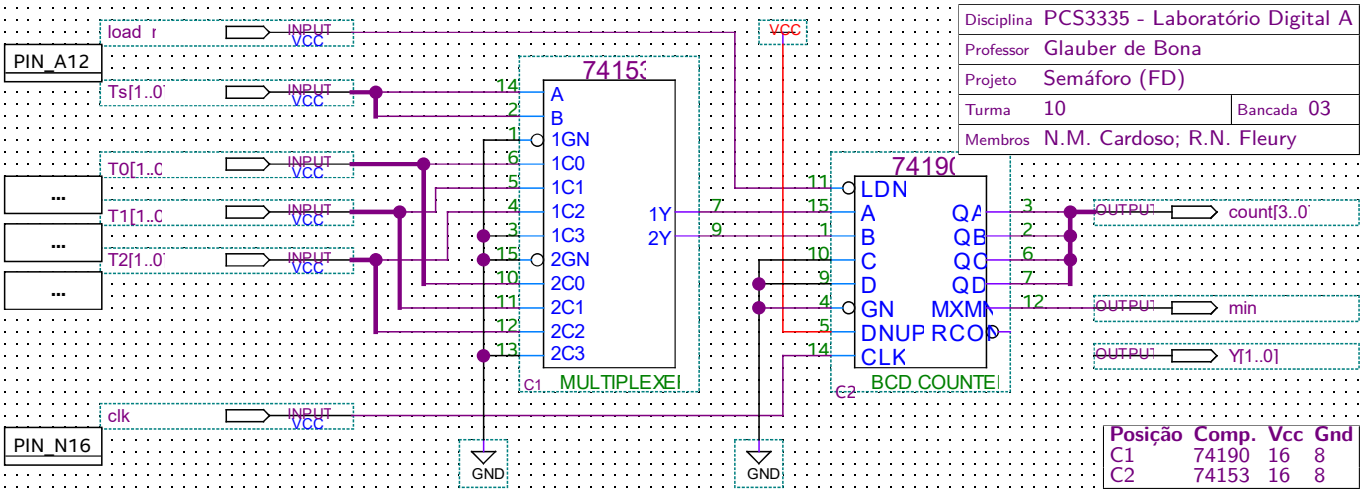


Figura 5: Diagrama lógico do Fluxo de Dados

3.4. Diagrama de Estados

A máquina de estados implementada na unidade de controle é composta por 12 estados. Os estados de carregamento $E0_L$ a $E5_L$ tem duração de 1 clock e são acionados a cada troca de luzes do semáforo. Já os estados de contagem $E0_C$ a $E5_C$ representam a contagem dos 3 tempos de mudança de luzes dos semáforos mostrados na Fig. 1.

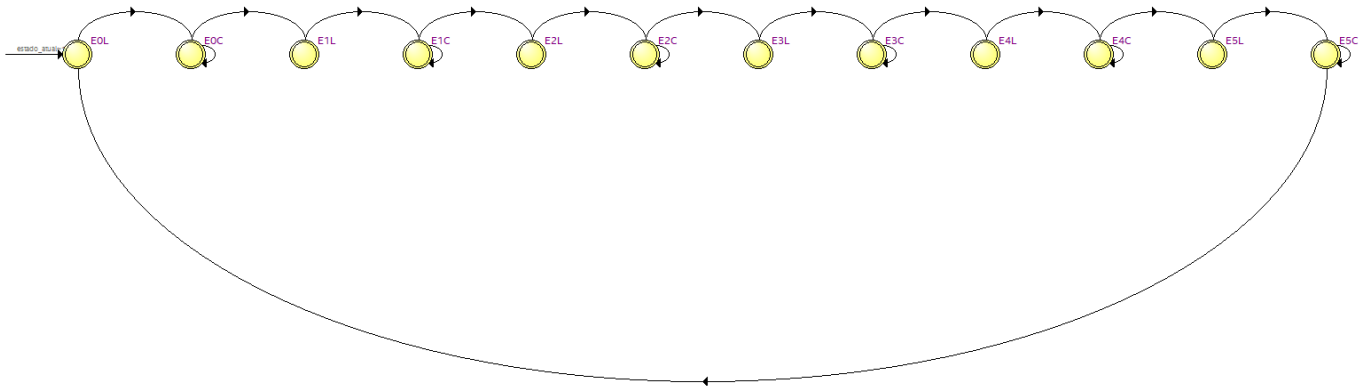


Figura 6: Diagrama de estados da unidade de controle

3.5. Simulação

As Figs. 7, 8 e 9 mostram as cartas de tempos do Fluxo de Dados, da Unidade de Controle e do Circuito Completo, respectivamente, obtidas pela simulação do circuito.

3.6. Testes

As tabelas de teste foram criadas a partir da simulação e possui os valores esperados para os sinais de saída e depuração. As Tabelas 1, 2 e 3 mostram as tabelas de testes para o Fluxo de Dados, a Unidade de Controle e o Circuito Completo, respectivamente.

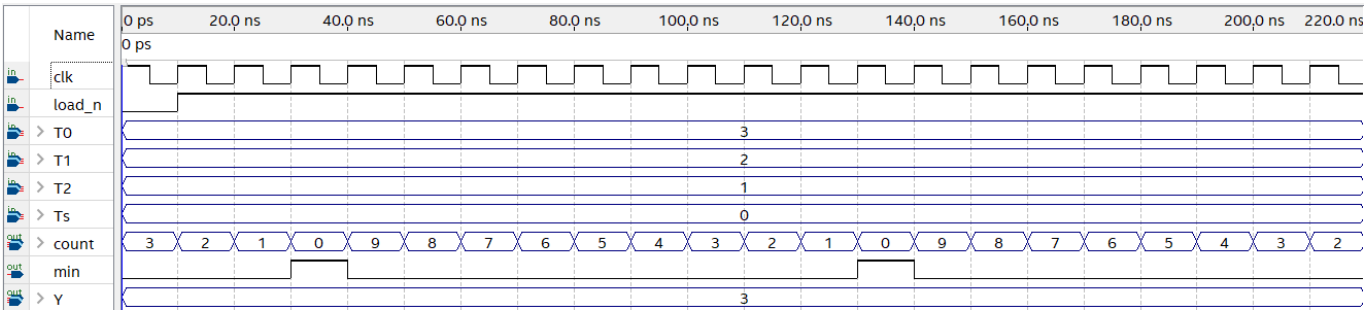


Figura 7: Carta de tempos do Fluxo de Dados

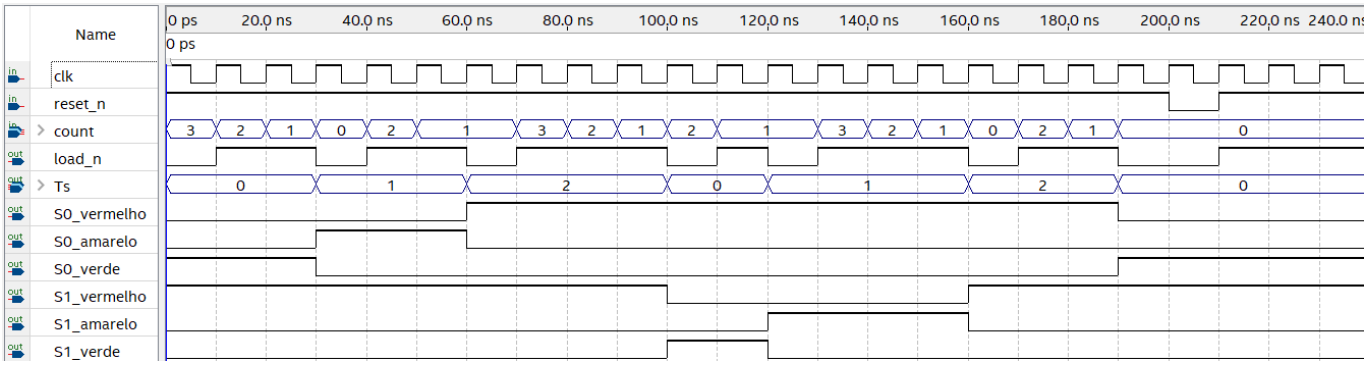


Figura 8: Carta de tempos da Unidade de Controle

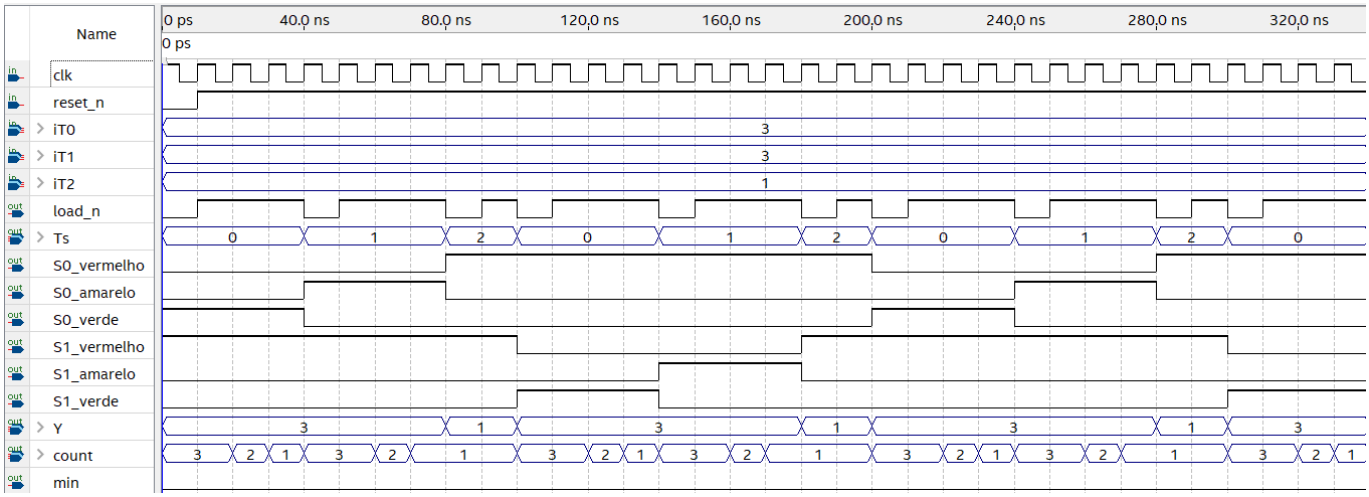


Figura 9: Carta de tempos do Circuito Completo, incluindo Fluxo de Dados e Unidade de Controle

Tabela 1: Tabela de testes do Fluxo de Dados

Entradas						Saídas		Dep.
clk	$\overline{\text{load}}$	T_0	T_1	T_2	T_s	min	count	Y
↑	0	11	10	01	00	0	0011	11
↑	1	11	10	01	00	0	0010	11
↑	1	11	10	01	00	0	0001	11
↑	1	11	10	01	00	1	0000	11
↑	1	11	10	01	00	0	1001	11
↑	1	11	10	01	00	0	1000	11
↑	1	11	10	01	00	0	0111	11
↑	1	11	10	01	00	0	0110	11
↑	1	11	10	01	00	0	0101	11

Tabela 2: Tabela de testes no Circuito Completo

Entradas			Saídas							
clk	$\overline{\text{rst}}$	count	$\overline{\text{load}}$	T_s	S_0	S_0	S_0	S_1	S_1	S_1
↑	1	0011	0	00	0	0	1	1	0	0
↑	1	0011	1	00	0	0	1	1	0	0
↑	1	0010	1	00	0	0	1	1	0	0
↑	1	0001	1	00	0	0	1	1	0	0
↑	1	0011	0	01	0	1	0	1	0	0
↑	1	0011	1	01	0	1	0	1	0	0
↑	1	0010	1	01	0	1	0	1	0	0
↑	1	0001	1	01	0	1	0	1	0	0
↑	1	0001	0	10	1	0	0	1	0	0
↑	1	0001	1	10	1	0	0	1	0	0
↑	1	0011	0	00	1	0	0	0	0	1
↑	1	0011	1	00	1	0	0	0	0	1
↑	1	0010	1	00	1	0	0	0	0	1
↑	1	0001	1	00	1	0	0	0	0	1
↑	1	0011	0	01	1	0	0	0	1	0
↑	1	0011	1	01	1	0	0	0	1	0
↑	1	0010	1	01	1	0	0	0	1	0
↑	1	0001	1	01	1	0	0	0	1	0
↑	1	0001	0	10	1	0	0	1	0	0
↑	1	0001	1	10	1	0	0	1	0	0
↑	0	×	0	01	0	0	1	1	0	0

Tabela 3: Tabela de testes no Circuito Completo

Entradas					Depuração				Saídas					
clk	$\overline{\text{rst}}$	T_0	T_1	T_2	$\overline{\text{load}}$	T_s	count	Y	S_0	S_0	S_0	S_1	S_1	S_1
↑	1	11	11	01	0	00	0011	11	0	0	1	1	0	0
↑	1	11	11	01	1	00	0011	11	0	0	1	1	0	0
↑	1	11	11	01	1	00	0010	11	0	0	1	1	0	0
↑	1	11	11	01	1	00	0001	11	0	0	1	1	0	0
↑	1	11	11	11	0	01	0011	11	0	1	0	1	0	0
↑	1	11	11	11	1	01	0011	11	0	1	0	1	0	0
↑	1	11	11	11	1	01	0010	11	0	1	0	1	0	0
↑	1	11	11	11	1	01	0001	11	0	1	0	1	0	0
↑	1	11	11	01	0	10	0001	01	1	0	0	1	0	0
↑	1	11	11	01	1	10	0001	01	1	0	0	1	0	0
↑	1	11	11	01	0	00	0011	11	1	0	0	0	0	1
↑	1	11	11	01	1	00	0011	11	1	0	0	0	0	1
↑	1	11	11	01	1	00	0010	11	1	0	0	0	0	1
↑	1	11	11	01	1	00	0001	11	1	0	0	0	0	1
↑	1	11	11	11	0	01	0011	11	1	0	0	0	1	0
↑	1	11	11	11	1	01	0011	11	1	0	0	0	1	0
↑	1	11	11	11	1	01	0010	11	1	0	0	0	1	0
↑	1	11	11	11	1	01	0001	11	1	0	0	0	1	0
↑	1	11	11	01	0	10	0001	01	1	0	0	1	0	0
↑	1	11	11	01	1	10	0001	01	1	0	0	1	0	0
↑	0	11	11	01	0	01	×	×	0	0	1	1	0	0

3.7. LEVANTAMENTO DOS MATERIAIS NECESSÁRIOS

Tabela 4: Unidades requeridas para cada CI

Slot	Operação	CI	Un. Requeridas	Un. Disponíveis
1	Contador	74190	1	1
2	Multiplexador	74153	2	2

Para garantir que o circuito projetado respeite as restrições de montagem, fizemos um levantamento dos recursos necessários para este circuito mostrado na Tabela 4. Ela mostra a quantidade de unidades lógicas requeridas para cada CI utilizado. As especificações de cada CI foi obtido pelos respectivos *datasheets*.

3.8. MONTAGEM E DEPURAÇÃO

O circuito será montado e depurado por partes. A Tabela 5 mostra a correlação dos sinais de depuração e dos LEDS da placa de montagem do fluxo de dados e a Tabela 6 mostra a designação dos pinos da unidade de controle. Os sinais de saída (luzes do semáforo) serão mostrados nos LEDS da FPGA.

Tabela 5: Tabela de correlação dos sinais de depuração com os LEDS da placa de montagem do Fluxo de Dados

Sinal	Endereço da Porta	Número do LED	Tipo E/S
Q_A	74163:14	LED0	Saída
Q_B	74163:13	LED1	Saída
Q_C	74163:12	LED2	Saída
Q_D	74163:11	LED3	Saída
O_A	74163:3	LED4	Entrada
O_B	74163:4	LED5	Entrada
O_C	74163:5	LED6	Entrada
O_D	74163:6	LED7	Entrada

Tabela 6: Tabela de designação de pinos da Unidade de Controle para a placa FPGA DE0-CV com Cyclone V 5CEBA4F23C7

Sinal	Código Interface	Código FPGA	Tipo E/S
clk	GPIO_0_D0	N16	Entrada
$\overline{\text{reset}}$	GPIO_0_D1	B16	Entrada
$\overline{\text{end}}$	GPIO_0_D3	C16	Entrada
$T_0[0]$	GPIO_0_D4	D17	Entrada
$T_0[1]$	GPIO_0_D5	K20	Entrada
$T_0[2]$	GPIO_0_D6	K21	Entrada
$T_0[3]$	GPIO_0_D7	K22	Entrada
$T_1[0]$	GPIO_0_D8	M20	Entrada
$T_1[1]$	GPIO_0_D9	M21	Entrada
$T_1[2]$	GPIO_0_D10	N21	Entrada
$T_1[3]$	GPIO_0_D11	R22	Entrada
$T_2[0]$	GPIO_0_D12	R21	Entrada
$T_2[1]$	GPIO_0_D13	T22	Entrada
$T_2[2]$	GPIO_0_D14	N20	Entrada
$T_2[3]$	GPIO_0_D15	N19	Entrada
$\overline{\text{load}}$	GPIO_1_D1	A12	Saída
offset[0]	GPIO_1_D4	A13	Saída
offset[1]	GPIO_1_D5	B13	Saída
offset[2]	GPIO_1_D6	C13	Saída
offset[3]	GPIO_1_D7	D13	Saída
S_0 vermelho	LEDR2	W2	Saída
S_0 amarelo	LEDR1	AA1	Saída
S_0 verde	LEDR0	AA2	Saída
S_1 vermelho	LEDR5	Y3	Saída
S_1 amarelo	LEDR4	N2	Saída
S_1 verde	LEDR3	N1	Saída

4. RESULTADOS

A implementação do Fluxo de Dados e da Unidade de Controle foram realizadas com sucesso na placa de montagem e na FPGA, respectivamente. Cada um desses componentes foram montados e testados separadamente e os resultados foram comparados com as Tabelas 1 e 2. Após concluídos os testes unitários, o Fluxo de Dados e a Unidade de Controle foram devidamente conectadas e testadas de acordo com a Tabela 3. Todas as saídas do circuito estão de acordo com o esperado e nenhuma alteração do planejamento foi realizada durante a montagem. Os resultados dos testes no Circuito Completo estão anotados na coluna "Check" da Tabela 7.

Tabela 7: Resultado dos testes no Circuito Completo

Entradas					Depuração				Saídas						
clk	$\overline{\text{rst}}$	T_0	T_1	T_2	$\overline{\text{load}}$	T_s	count	Y	S_0	S_0	S_0	S_1	S_1	S_1	Check
↑	1	11	11	01	0	00	0011	11	0	0	1	1	0	0	✓
↑	1	11	11	01	1	00	0011	11	0	0	1	1	0	0	✓
↑	1	11	11	01	1	00	0010	11	0	0	1	1	0	0	✓
↑	1	11	11	01	1	00	0001	11	0	0	1	1	0	0	✓
↑	1	11	11	11	0	01	0011	11	0	1	0	1	0	0	✓
↑	1	11	11	11	1	01	0011	11	0	1	0	1	0	0	✓
↑	1	11	11	11	1	01	0010	11	0	1	0	1	0	0	✓
↑	1	11	11	11	1	01	0001	11	0	1	0	1	0	0	✓
↑	1	11	11	01	0	10	0001	01	1	0	0	1	0	0	✓
↑	1	11	11	01	1	10	0001	01	1	0	0	1	0	0	✓
↑	1	11	11	01	0	00	0011	11	1	0	0	0	0	1	✓
↑	1	11	11	01	1	00	0011	11	1	0	0	0	0	1	✓
↑	1	11	11	01	1	00	0010	11	1	0	0	0	0	1	✓
↑	1	11	11	01	1	00	0001	11	1	0	0	0	0	1	✓
↑	1	11	11	11	0	01	0011	11	1	0	0	0	1	0	✓
↑	1	11	11	11	1	01	0011	11	1	0	0	0	1	0	✓
↑	1	11	11	11	1	01	0010	11	1	0	0	0	1	0	✓
↑	1	11	11	11	1	01	0001	11	1	0	0	0	1	0	✓
↑	1	11	11	01	0	10	0001	01	1	0	0	1	0	0	✓
↑	1	11	11	01	1	10	0001	01	1	0	0	1	0	0	✓
↑	0	11	11	01	0	01	×	×	0	0	1	1	0	0	✓

5. Conclusão

Neste experimento, usamos a separação Fluxo de Dados e Unidade de Controle em um projeto de semáforos. O Fluxo de Dados foi montado em uma placa de montagem usando CIs TTL e a Unidade de Controle foi implantado em uma FPGA com descrição em VHDL. A conexão entre placa de montagem e FPGA foi feita usando conversores de tensão. Os sinais obtidos pelos testes no circuto estão de acordo com as tabelas de testes previamente elaboradas.

APÊNDICE

A. DESCRIÇÕES VHDL

Listing 1: Descrição da Unidade de Controle

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5
6  entity unidade_controle is
7    port (
8      clk, reset_n: in std_logic;
9      count: in std_logic_vector(3 downto 0);
10     -- count_min: in std_logic;
11     load_n: out std_logic; -- sinal de controle do contador
12     Ts: out std_logic_vector(1 downto 0); --sinal de controle do mux
13     S0_verde, S0_amarelo, S0_vermelho: out std_logic;
14     S1_verde, S1_amarelo, S1_vermelho: out std_logic
15   );
16 end unidade_controle;
17
18 architecture unidade_controle_arch of unidade_controle is
19   type estado_tipo is (E0L, E0C, E1L, E1C, E2L, E2C, E3L, E3C, E4L, E4C, E5L,
     ↪ E5C);
20   signal estado_atual, prox_estado: estado_tipo;
21   signal luzes: std_logic_vector(5 downto 0);
22 begin
23   sincrono: process(clk, reset_n, prox_estado)
24   begin
25     if reset_n='0' then
26       estado_atual <= E0L;
27     elsif clk'event and clk='1' then
28       estado_atual <= prox_estado;
29     end if;
30   end process;
31
32   circ_prox_estado: process(estado_atual, count)
33   begin
```

```
34      case estado_atual is
35        when E0L =>
36          prox_estado <= E0C;
37          Ts <= "00";
38          luzes <= "100001";
39          load_n <= '0';
40        when E0C =>
41          Ts <= "00";
42          luzes <= "100001";
43          load_n <= '1';
44          if count = "0001" then
45            prox_estado <= E1L;
46          else
47            prox_estado <= E0C;
48          end if;
49        when E1L =>
50          prox_estado <= E1C;
51          Ts <= "01";
52          luzes <= "010001";
53          load_n <= '0';
54        when E1C =>
55          Ts <= "01";
56          luzes <= "010001";
57          load_n <= '1';
58          if count = "0001" then
59            prox_estado <= E2L;
60          else
61            prox_estado <= E1C;
62          end if;
63        when E2L =>
64          prox_estado <= E2C;
65          Ts <= "10";
66          luzes <= "001001";
67          load_n <= '0';
68        when E2C =>
69          Ts <= "10";
70          luzes <= "001001";
71          load_n <= '1';
72          if count = "0001" then
73            prox_estado <= E3L;
74          else
```

```
75            prox_estado <= E2C;
76          end if;
77        when E3L =>
78          prox_estado <= E3C;
79          Ts <= "00";
80          luzes <= "001100";
81          load_n <= '0';
82        when E3C =>
83          Ts <= "00";
84          luzes <= "001100";
85          load_n <= '1';
86          if count = "0001" then
87            prox_estado <= E4L;
88          else
89            prox_estado <= E3C;
90          end if;
91        when E4L =>
92          prox_estado <= E4C;
93          Ts <= "01";
94          luzes <= "001010";
95          load_n <= '0';
96        when E4C =>
97          Ts <= "01";
98          luzes <= "001010";
99          load_n <= '1';
100         if count = "0001" then
101           prox_estado <= E5L;
102         else
103           prox_estado <= E4C;
104         end if;
105       when E5L =>
106         prox_estado <= E5C;
107         Ts <= "10";
108         luzes <= "001001";
109         load_n <= '0';
110       when E5C =>
111         Ts <= "10";
112         luzes <= "001001";
113         load_n <= '1';
114         if count = "0001" then
115           prox_estado <= E0L;
```

```
116         else
117           prox_estado <= E5C;
118         end if;
119     end case;
120   end process;
121
122   S0_verde <= luzes(5);
123   S0_amarelo <= luzes(4);
124   S0_vermelho <= luzes(3);
125   S1_verde <= luzes(2);
126   S1_amarelo <= luzes(1);
127   S1_vermelho <= luzes(0);
128 end architecture;
```

Universidade de São Paulo

Escola Politécnica

Departamento de Eng. de Computação e Sistemas Digitais