Laboratório 7: Projeto de um sistema semafórico usando VHDL

Bernardo Hoffmann da Silva
Marcos Vinicius Pereira Veloso

1 Projeto do Circuito

Especificada as condições do problema, pode-se elaborar o grafo de transição de estados no Modelo Moore conforme segue na Figura 1, referente ao controlador principal. Nota-se que, além dos estados já citados na metodologia, representam-se também os estados intermediários (que variam apenas com a subida do clock).

Para exemplificar, considere o A como estado atual. Quando um dos sensores s_1 ou s_2 identifica um veículo na via secundária, é associado o valor 1 para a saída START_VP_AMAR, que por sua vez definirá quanto tempo o valor de TIMER permaneçerá em 0 após o próximo clock. Feito isso, passa-se para o estado B, em que as variáveis de saída start são resetadas e a contagem de TIMER começa. Com efeito, não se passa para o próximo estado enquanto TIMER for 0. Ao atingir valor 1, passa-se para o estado B' no próximo clock, que por sua vez associa o valor de saída 1 para START_VERM_SEG, que definirá o próximo tempo de valor baixo de TIMER. Com isso, passa-se para o estado C, em que as variáveis de saída START são resetadas e começa a nova contagem de tempo. Tal processo continua de forma análoga para os demais estados, definindo assim a transição de estados do problema.

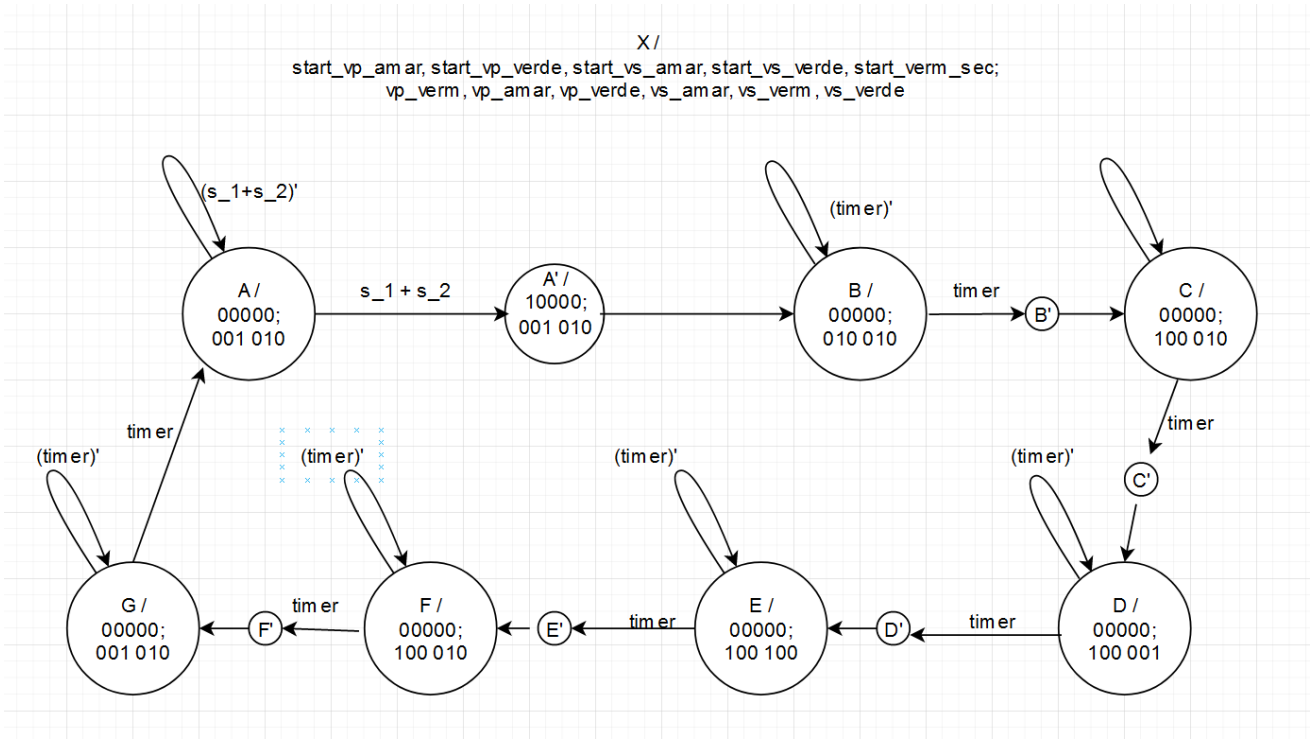

Figura 1: Grafo de transição de estados do controlador principal para o problema especificado.

2 Descrição em VHDL

Tal como segue na metologia, é proposto o uso de duas máquinas de estado finito: Uma que rege o comportamento do TIMER conforme entradas de referência e variáveis de início de contagem, e outra que define os estados que controlam a dinâmica dos semáforos. Para a primeira máquina, denota-se o uso de código conforme segue abaixo

2.1 Código em VHDL do projeto do *timer*.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity lab7 is
6          port (START_VP_AMAR, START_VP_VERDE, START_VS_AMAR,
7          START_VS_VERDE, START_VERM_SEG, RESET, CLOCK: in std_logic;
8
9          VP_TAMAR, VS_TAMAR, TVERM_SEG : in std_logic_vector(2 downto 0);
10
11         VP_TVERDE, VS_TVERDE: in std_logic_vector(7 downto 0);
12
```

```
13          TIMER_ALARM: out std_logic);
14  end lab7;
15
16  architecture T of lab7 is
17          signal t_cnt : unsigned(7 downto 0);
18
19          begin
20          process(CLOCK,RESET)
21                  begin
22
23                  if (RESET = '1') then
24                          t_cnt <= (others => '0');
25
26                  elsif(rising_edge(CLOCK)) then
27                  if(t_cnt = "00000000") then
28                  if(START_VP_AMAR = '1') then
29                          t_cnt <= unsigned('0' & '0' & '0' & '0' & '0' & VP_TAMAR);
30                  elsif(START_VERM_SEG = '1') then
31                          t_cnt <= unsigned('0' & '0' & '0' & '0' & '0' & TVERM_SEG);
32                  elsif(START_VS_VERDE = '1') then
33                          t_cnt <= unsigned(VS_TVERDE);
34                  elsif(START_VS_AMAR = '1') then
35                          t_cnt <= unsigned('0' & '0' & '0' & '0' & '0' & VS_TAMAR);
36                  elsif(START_VP_VERDE = '1') then
37                          t_cnt <= unsigned(VP_TVERDE);
38                  end if;
39                  else t_cnt <= t_cnt - 1;
40                  end if;
41                  end if;
42          end process;
43                  TIMER_ALARM <= '1' when (t_cnt = "00000000") else '0';
44          end T;
```

2.2 Código em VHDL do projeto do controlador principal.

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use IEEE.numeric_std.all;
4
5   entity controlador is
6           port ( TIMER_ALARM, SENSOR1, SENSOR2, RESET, CLOCK: in std_logic;
7
8                   START_VP_AMAR, START_VP_VERDE, START_VS_AMAR,
```

```
9                 START_VS_VERDE, START_VERM_SEG, VP_VERM, VP_AMAR, VP_VERDE,
10                VS_AMAR, VS_VERM, VS_VERDE: out std_logic);
11
12 end controlador;
13
14 architecture controlArc of controlador is
15         type state_type is (STA,STB,STC,STD1,STE,STF,STG);
16         signal PS, NS : state_type;
17 begin
18
19 sync: process (CLOCK, NS, RESET)
20         begin
21                 if (RESET = '1') then
22                         PS <= STA;
23                 elsif(rising_edge(CLOCK)) then
24                         PS <= NS;
25                 end if;
26 end process sync;
27
28 comb: process(PS, SENSOR1, SENSOR2, TIMER_ALARM)
29
30 begin
31
32         START_VP_AMAR <= '0';
33         START_VP_VERDE <= '0';
34         START_VS_AMAR <= '0';
35         START_VS_VERDE <= '0';
36         START_VERM_SEG <= '0';
37         VP_VERM <= '0';
38         VP_AMAR <= '0';
39         VP_VERDE <= '0';
40         VS_AMAR <= '0';
41         VS_VERM <= '0';
42         VS_VERDE <= '0';
43
44         case PS is
45                 when STA =>
46                         START_VP_AMAR <= '0';
47                         START_VP_VERDE <= '0';
48                         START_VS_AMAR <= '0';
49                         START_VS_VERDE <= '0';
50                         START_VERM_SEG <= '0';
51                         VP_VERM <= '0';
52                         VP_AMAR <= '0';
```

```
53                      VP_VERDE <= '1';
54                      VS_AMAR <= '0';
55                      VS_VERM <= '1';
56                      VS_VERDE <= '0';
57
58                      if(SENSOR1 = '1') or (SENSOR2 = '1') then
59                              START_VP_AMAR <= '1';
60                              NS <= STB;
61                      else
62                              NS <= STA;
63                      end if;
64
65              when STB =>
66                      START_VP_AMAR <= '0';
67                      START_VP_VERDE <= '0';
68                      START_VS_AMAR <= '0';
69                      START_VS_VERDE <= '0';
70                      START_VERM_SEG <= '0';
71                      VP_VERM <= '0';
72                      VP_AMAR <= '1';
73                      VP_VERDE <= '0';
74                      VS_AMAR <= '0';
75                      VS_VERM <= '1';
76                      VS_VERDE <= '0';
77
78                      if(TIMER_ALARM = '0') then
79                              NS <= STB;
80                      else
81                              START_VERM_SEG <= '1';
82                              NS <= STC;
83                      end if;
84
85
86              when STC =>
87                      START_VP_AMAR <= '0';
88                      START_VP_VERDE <= '0';
89                      START_VS_AMAR <= '0';
90                      START_VS_VERDE <= '0';
91                      START_VERM_SEG <= '0';
92                      VP_VERM <= '1';
93                      VP_AMAR <= '0';
94                      VP_VERDE <= '0';
95                      VS_AMAR <= '0';
96                      VS_VERM <= '1';
```

```
97                      VS_VERDE <= '0';
98
99                      if(TIMER_ALARM = '0') then
100                             NS <= STC;
101                     else
102                             START_VS_VERDE <= '1';
103                             NS <= STD1;
104                     end if;
105
106
107             when STD1 =>
108
109                     START_VP_AMAR <= '0';
110                     START_VP_VERDE <= '0';
111                     START_VS_AMAR <= '0';
112                     START_VS_VERDE <= '0';
113                     START_VERM_SEG <= '0';
114                     VP_VERM <= '1';
115                     VP_AMAR <= '0';
116                     VP_VERDE <= '0';
117                     VS_AMAR <= '0';
118                     VS_VERM <= '0';
119                     VS_VERDE <= '1';
120
121                     if(TIMER_ALARM = '0') then
122                             NS <= STD1;
123                     else
124                             START_VS_AMAR <= '1';
125                             NS <= STE;
126                     end if;
127
128             when STE =>
129
130                     START_VP_AMAR <= '0';
131                     START_VP_VERDE <= '0';
132                     START_VS_AMAR <= '0';
133                     START_VS_VERDE <= '0';
134                     START_VERM_SEG <= '0';
135                     VP_VERM <= '1';
136                     VP_AMAR <= '0';
137                     VP_VERDE <= '0';
138                     VS_AMAR <= '1';
139                     VS_VERM <= '0';
140                     VS_VERDE <= '0';
```

```
141
142                     if(TIMER_ALARM = '0') then
143                             NS <= STE;
144                     else
145                             START_VERM_SEG <= '1';
146                             NS <= STF;
147                     end if;
148
149             when STF =>
150
151                     START_VP_AMAR <= '0';
152                     START_VP_VERDE <= '0';
153                     START_VS_AMAR <= '0';
154                     START_VS_VERDE <= '0';
155                     START_VERM_SEG <= '0';
156                     VP_VERM <= '1';
157                     VP_AMAR <= '0';
158                     VP_VERDE <= '0';
159                     VS_AMAR <= '0';
160                     VS_VERM <= '1';
161                     VS_VERDE <= '0';
162
163                     if(TIMER_ALARM = '0') then
164                             NS <= STF;
165                     else
166                             START_VP_VERDE <= '1';
167                             NS <= STG;
168                     end if;
169
170             when STG        =>
171
172                     START_VP_AMAR <= '0';
173                     START_VP_VERDE <= '0';
174                     START_VS_AMAR <= '0';
175                     START_VS_VERDE <= '0';
176                     START_VERM_SEG <= '0';
177                     VP_VERM <= '0';
178                     VP_AMAR <= '0';
179                     VP_VERDE <= '1';
180                     VS_AMAR <= '0';
181                     VS_VERM <= '1';
182                     VS_VERDE <= '0';
183
184                     if(TIMER_ALARM = '0') then
```

```
185                                 NS <= STG;
186                         else
187                                 NS <= STA;
188                         end if;
189
190                 when others =>
191                         START_VP_AMAR <= '0';
192                         START_VP_VERDE <= '0';
193                         START_VS_AMAR <= '0';
194                         START_VS_VERDE <= '0';
195                         START_VERM_SEG <= '0';
196                         VP_VERM <= '0';
197                         VP_AMAR <= '0';
198                         VP_VERDE <= '0';
199                         VS_AMAR <= '0';
200                         VS_VERM <= '0';
201                         VS_VERDE <= '0';
202
203         end case;
204         end process comb;
205 end controlArc;
```

2.3 Código em VHDL do projeto do semáforo completo.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity semaforoFINAL is
6          port (SENSOR1s, SENSOR2s, RESETs, CLOCKs: in std_logic;
7
8          VP_TAMARs, VS_TAMARs, TVERM_SEGs : in std_logic_vector(2 downto 0);
9
10         VP_TVERDEs, VS_TVERDEs: in std_logic_vector(7 downto 0);
11
12
13         VP_VERMs, VP_AMARs, VP_VERDEs,
14         VS_AMARs, VS_VERMs, VS_VERDEs: out std_logic);
15
16 end semaforoFINAL;
17
18 architecture semafArc of semaforoFINAL is
19
```

```
component controlador is
        port ( TIMER_ALARM, SENSOR1, SENSOR2, RESET, CLOCK: in std_logic;

        START_VP_AMAR, START_VP_VERDE, START_VS_AMAR,
        START_VS_VERDE, START_VERM_SEG, VP_VERM, VP_AMAR, VP_VERDE,
        VS_AMAR, VS_VERM, VS_VERDE: out std_logic);

end component;

component lab7 is
        port (START_VP_AMAR, START_VP_VERDE, START_VS_AMAR,
        START_VS_VERDE, START_VERM_SEG, RESET, CLOCK: in std_logic;

        VP_TAMAR, VS_TAMAR, TVERM_SEG : in std_logic_vector(2 downto 0);

        VP_TVERDE, VS_TVERDE: in std_logic_vector(7 downto 0);

        TIMER_ALARM: out std_logic);
end component;

signal aux1, aux2, aux3, aux4, aux5, aux6: std_logic;

begin
CONTport: controlador port map (START_VP_AMAR => aux1, START_VP_VERDE => aux2,
START_VS_AMAR => aux3, START_VS_VERDE => aux4, START_VERM_SEG => aux5,
TIMER_ALARM => aux6, SENSOR1 => SENSOR1s, SENSOR2 => SENSOR2s, RESET => RESETs,
CLOCK => CLOCKs, VP_VERM => VP_VERMs, VP_AMAR => VP_AMARs, VP_VERDE => VP_VERDEs,
VS_AMAR => VS_AMARs, VS_VERM => VS_VERMs, VS_VERDE => VS_VERDEs);

LAB7port : lab7 port map (START_VP_AMAR => aux1, START_VP_VERDE => aux2,
START_VS_AMAR => aux3,START_VS_VERDE => aux4,START_VERM_SEG => aux5,
TIMER_ALARM => aux6, RESET => RESETs, CLOCK => CLOCKs, VP_TAMAR => VP_TAMARs,
VS_TAMAR => VS_TAMARs, TVERM_SEG => TVERM_SEGs, VP_TVERDE => VP_TVERDEs,
VS_TVERDE => VS_TVERDEs);
end semafArc;
```

3 Resultados de simulação temporal

Feita a implementação denotada na seção anterior, prossegue-se com a análise temportal da resposta do semáforo dada as entradas de sensores definidas. Com isso, pode-se denotar o correto funcionamento da MEF que rege o TIMER e da MEF principal, conforme segue nas Figuras 2 e 3.

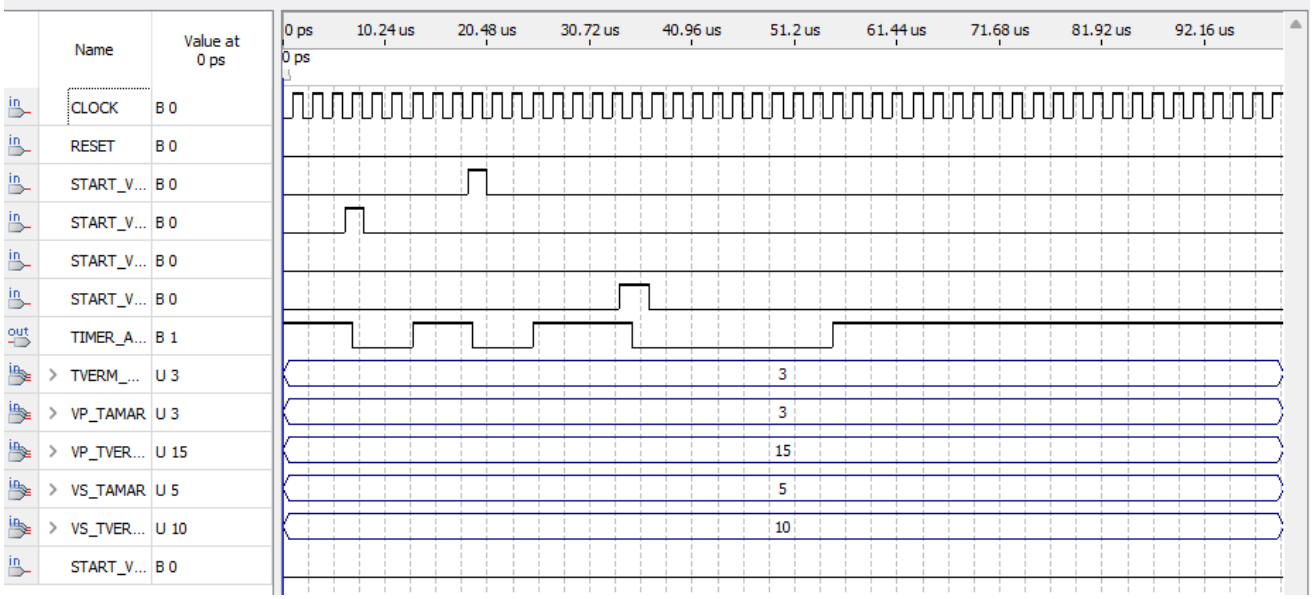

Figura 2: Simulação temporal do *timer*.

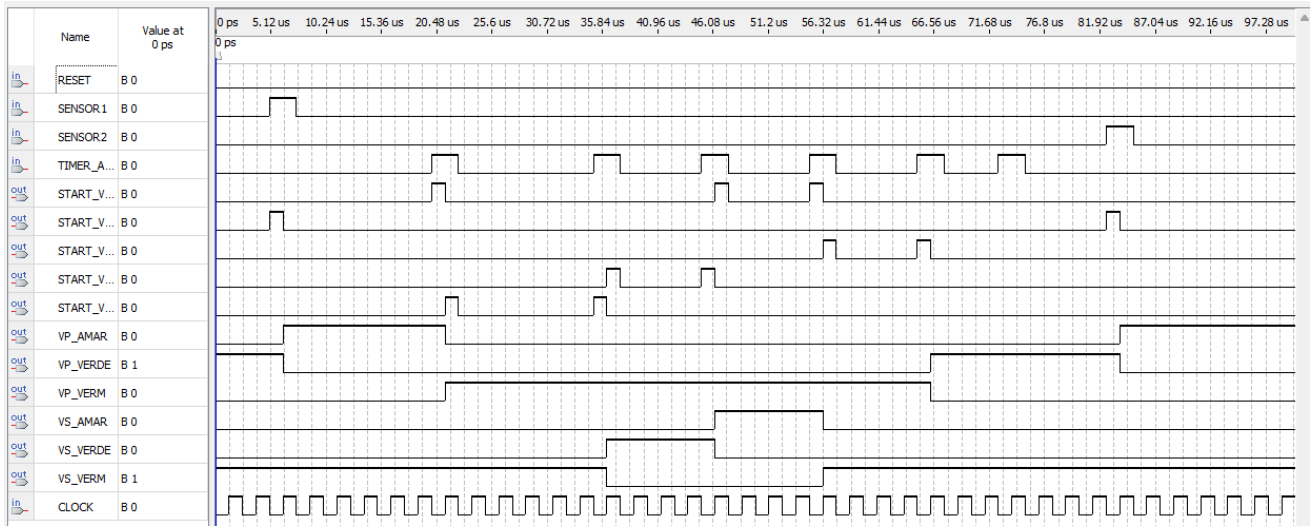

Figura 3: Simulação temporal do controlador principal.

Por fim, nota-se que a implementação completa, descrita tal como no código especificado na seção anterior, atende ao comportamento esperado para o semáforo, como é denotado na Figura 4, em que se é possível ver a transição de estados da máquina após a subida do *SENSOR1s*. É possível visualizar a mudança dos valores da saída dos semáforos, cujas transições ocorrem conforme o intervalo de tempo especificado pela entrada da máquina de estado finito.

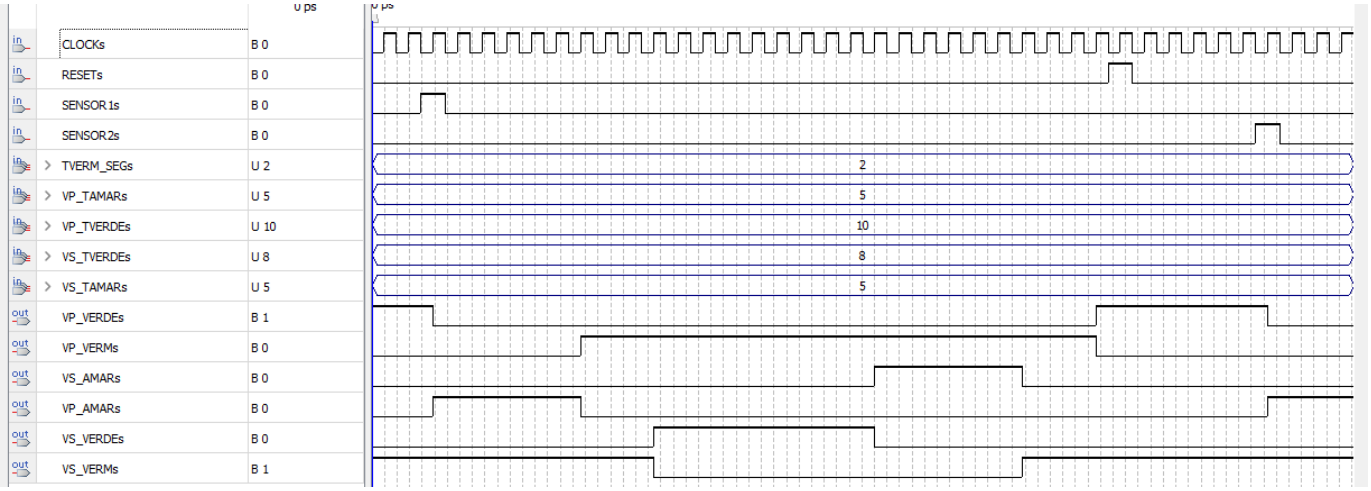

Figura 4: Diagrama de temporização para o semáforo completo.