Universidad de Buenos Aires
Facultad de Ingeniería

86.41 - Sistemas Digitales

Trabajo Práctico Nº1: Implementación de un Sistema Secuencial en VHDL

Profesores:

Ing. Nicolás Álvarez

Ing. Octavio Alpago

Alumno:

Batallan, David Leonardo, *Padrón 97529*
dbatallan@fi.uba.ar

24 de mayo de 2025

Índice

1. Resumen

El presente Trabajo Practico tiene como objetivo que el alumno fije el concepto de circuito secuencial sincrónico aplicando el lenguaje de descripción de hardware VHDL.

2. Especificaciones del diseño

Implementar un circuito para controlar dos semáforos en un cruce de calles. Dicho circuito tendría 6 salidas: rojo 1, amarillo 1, verde 1, rojo 2, amarillo 2, verde 2. El tiempo en amarillo seria de 3 seg. mientras que en rojo y verde sera de 30 seg. El reloj del sistema tendría una frecuencia de aparición de 50MHz.

3. Desarrollo

3.1. Diagrama de estados

Para modelar el funcionamiento de dos semáforos se planteo un diagrama de estados en el cual, según que valor adopte, los semáforos encenderán las luces correspondientes.

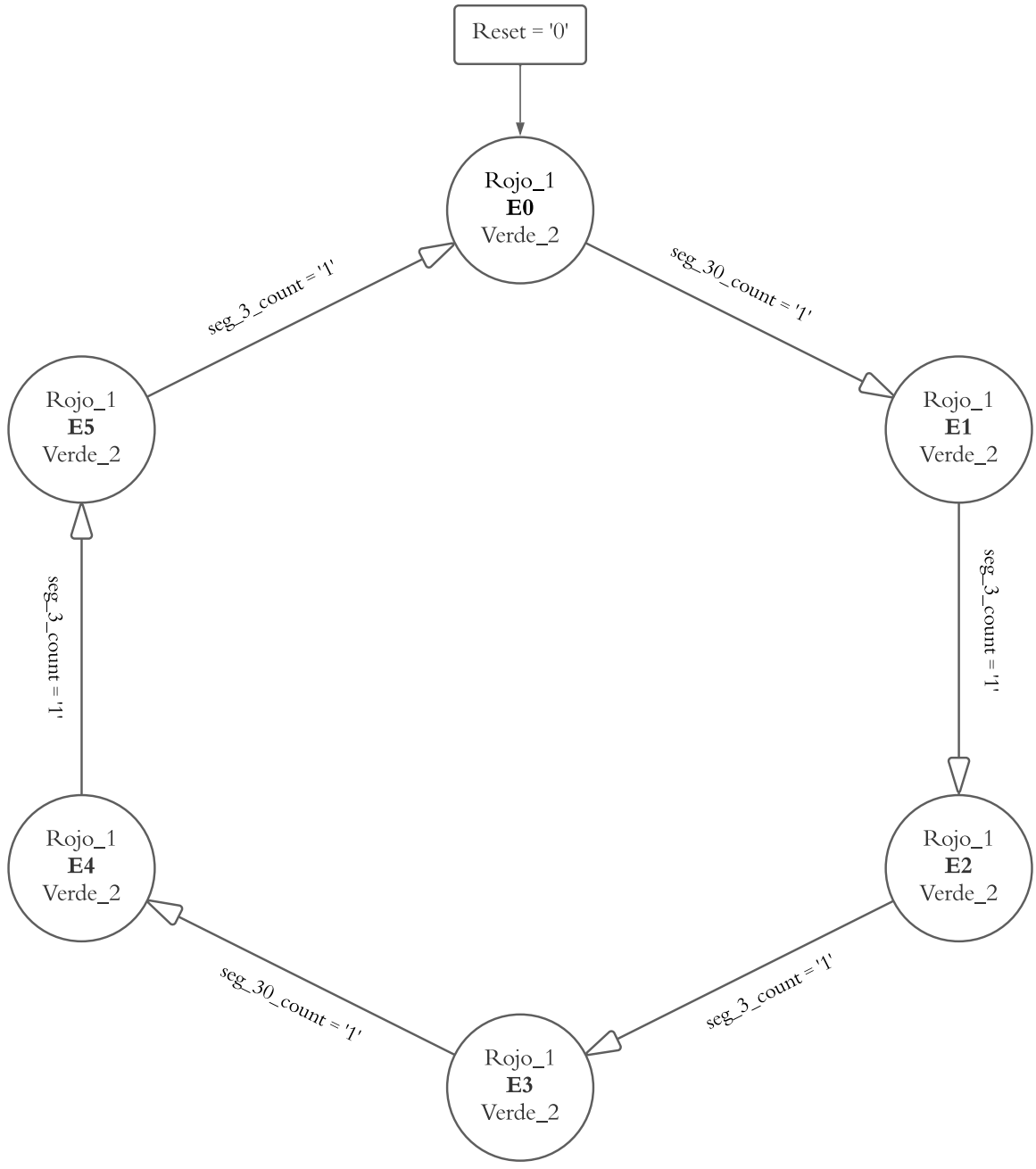

Figura 1: Diagrama de estados de las luces del semáforo

En la figura 1 se observa que en caso de habilitar un reseteo de la secuencia, se retornara al estado inicial E0. En caso contrario ira pasando de estado en estado en la medida que ocurran los lapsos de tiempo.

3.2. Diseño

El proyecto esta compuesto por un contador de 31 con habilitador, reiniciador, clock y carga. En este caso se le aplicara una frecuencia de reloj de 50 Mhz, por lo que para contar hasta 30s se deberá contar:

$$30s \cdot 50Mhz = 1500000000 \tag{1}$$

y para 3 segundo:

$$30s \cdot 50Mhz = 150000000 \tag{2}$$

Para saber cuantos bit se necesitan para poder representar un entero que cuente hasta 30s debe ser $log_2(1500000000) = 30,48$. Por lo que se necesita como mínimo una cantidad de bits igual a 31.

4. Implementación en VHDL

MUX.VHD

Archivo: mux.vhd

```
1  -- TP 1
2  -- Materia: Sistemas digitales
3  -- Alumno: Batallan David Leonardo
4  -- Padron: 97529
5
6  library IEEE;
7  use IEEE.std_logic_1164.all;
8  use IEEE.numeric_std.all;
9
10 entity mux is
11     port(
12         x0  : in std_logic; -- Entrada 0 mux
13         x1  : in std_logic; -- Entrada 1 mux
14         s   : in std_logic; -- Selectro mux
15         y   : out std_logic -- Salida
16     );
17 end mux;
18
19 architecture behavioral of mux is
20 begin
21     process(x0,x1,s)
22     begin
23         case s is
24             when '0' =>
25                 y <= x0;
26
27             when others =>
28                 y <= x1;
29         end case;
30     end process;
31 end behavioral;
```

CONTADORN.VHD

Archivo: contadorN.vhd

```
1  -- TP 1
2  -- Materia: Sistemas digitales
3  -- Alumno: Batallan David Leonardo
4  -- Padron: 97529
5
6  library IEEE;
7  use IEEE.std_logic_1164.all;
8  use IEEE.numeric_std.all;
9
10 -- Contador generico de N bits con señal del carga
11 entity counterN is
12     generic(
13         N : natural := 8
14     );
15     port(
16         rst             : in std_logic;
17         clk             : in std_logic;
18         load            : in std_logic;
19         val             : in std_logic_vector(N-1 downto 0);
20         count           : out std_logic_vector(N-1 downto 0);
21         seg_30_count    : out std_logic;
22         seg_3_count     : out std_logic
23     );
24 end counterN;
25
26 architecture behavioral of counterN is
27
28     constant N30_SEG : natural := 1499999999;
29     constant N3_SEG : natural := 149999999;
30
31     signal  aux_count : unsigned(N-1 downto 0);
32
33 begin
34
35     process(clk,rst)
36     begin
37         if rst = '1' then
38             aux_count <= (others => '0');
39         elsif clk = '1' and clk'event then
40             if load = '1' then
41                 aux_count <= unsigned(val);
42             else
43                 aux_count <= aux_count + 1;
44             end if;
45         end if;
46     end process;
47
48     count <= std_logic_vector(aux_count);
49
50     seg_30_count <= '1' when (aux_count = N30_SEG) else '0';
51
52     seg_3_count <= '1' when (aux_count = N3_SEG) else '0';
53
54
55 end behavioral;
```

SEMAFOROS.VHD

Archivo: semaforos.vhd

```
1  -- TP 1
2  -- Materia: Sistemas digitales
3  -- Alumno: Batallan David Leonardo
4  -- Padrón: 97529
5
6  library IEEE;
7  use IEEE.std_logic_1164.all;
8  use IEEE.numeric_std.all;
9
10 entity semaforos is
11         port (
12                 rst                : in std_logic;
13                 clk                : in std_logic;
14                 rojo_1                : out std_logic;
15                 amarillo_1        : out std_logic;
16                 verde_1                : out std_logic;
17                 rojo_2                : out std_logic;
18                 amarillo_2        : out std_logic;
19                 verde_2                : out std_logic
20         );
21 end semaforos;
22
23 architecture behavioral of semaforos is
24
25         -- Definición de constantes
26         constant N_counter        : natural := 31;
27         constant OCHO                : unsigned(3 downto 0) := to_unsigned(8, 4);
28
29         -- Definición del tipo de dato "t_state"
30         type t_state is (R1_V2, R1A1_V2, V1_A2, V1_R2, A1_R2A2);
31
32         -- Señales
33         constant val        : std_logic_vector(N_counter-1 downto 0) := (others => '0');
34         signal state        : t_state;
35         signal mux_out        : std_logic;
36         signal sel_mux        : std_logic;
37         signal seg_30        : std_logic;
38         signal seg_3        : std_logic;
39         signal count        : unsigned(3 downto 0);
40         signal d     : unsigned(3 downto 0);
41
42 begin
43
44         -- Instancia del MUX
45         mux: entity work.mux
46                 port map (
47                         x0 => seg_30,
48                         x1 => seg_3,
49                         s  => sel_mux,
50                         y  => mux_out
51                 );
52
53         -- Instancia del contador
54         contador: entity work.counterN
55                 generic map (N => N_counter)
56                 port map (
57                         rst   => rst,
58                         clk   => clk,
59                         load  => mux_out,
60                         val   => val,
61                         count => open,
```

```
62                      seg_30_count => seg_30,
63                      seg_3_count  => seg_3
64              );
65
66      -- Separación de lógica de incremento
67      process(count)
68      begin
69              if count = OCHO then
70                      d <= (others => '0');
71              else
72                      d <= count + 1;
73              end if;
74      end process;
75
76
77      -- Máquina de estados
78      fsm: process(clk, rst)
79      begin
80              if rst = '1' then
81                      state <= R1_V2;
82                      count <= (others => '0');
83
84              elsif rising_edge(clk) then
85                      case state is
86
87                              when R1_V2 =>
88                                      if seg_3 = '1' then
89                                              if count = OCHO then
90                                                      state <= R1A1_V2;
91                                                      count <= d;
92                                              else
93                                                      count <= d;
94                                              end if;
95                                      end if;
96
97                              when R1A1_V2 =>
98                                      if seg_3 = '1' then
99                                              state <= V1_A2;
100                                     end if;
101
102                             when V1_A2 =>
103                                     if seg_3 = '1' then
104                                             state <= V1_R2;
105                                     end if;
106
107                             when V1_R2 =>
108                                     if seg_3 = '1' then
109                                             if count = OCHO then
110                                                     state <= A1_R2A2;
111                                                     count <= d;
112                                             else
113                                                     count <= d;
114                                             end if;
115                                     end if;
116                             when A1_R2A2 =>
117                                     if seg_3 = '1' then
118                                             state <= R1_V2;
119                                     end if;
120
121                     end case;
122             end if;
123     end process;
124
125
126     -- Lógica de control del MUX
```

```
127         sel_mux <= '1' when (        state = R1_V2 or state = R1A1_V2 or
128                                  state = V1_A2 or state = V1_R2 or
129                                  state = A1_R2A2) else '0';
130 
131         -- Salidas semáforo 1
132         rojo_1        <= '1'         when         (state = R1_V2 or state = R1A1_V2) else '0';
133         amarillo_1  <= '1' when (state = R1A1_V2 or state = A1_R2A2) else '0';
134         verde_1         <= '1' when         (state = V1_A2 or state = V1_R2) else '0';
135 
136         -- Salidas semáforo 2
137         rojo_2          <= '1' when         (state = V1_R2 or state = A1_R2A2) else '0';
138         amarillo_2  <= '1' when (state = V1_A2 or state = A1_R2A2) else '0';
139         verde_2         <= '1' when         (state = R1_V2 or state = R1A1_V2) else '0';
140 
141 end behavioral;
142 
```

5. Simulación

Para poder realizar la simulación, se cambio el tiempo de clock a un periodo de 1, y en lugar de contar hasta 1500000000 y 150000000 para obtener los 30 y 3 segundos respectivamente, solo se cuenta hasta 30 y 3 segundos. La simulación se realizó con el siguiente testbeanch.

TB_SEMAFOROS.VHD

Archivo: tb_semaforos.vhd

```
1  -- Trabajo Practico Nº1: Implementación de un Sistema Secuencial en VHDL
2  -- Materia: Sistemas digitales
3  -- Alumno: Batallan David Leonardo
4  -- Padron: 97529
5  
6  library IEEE;
7  use IEEE.std_logic_1164.all;
8  use IEEE.numeric_std.all;
9  
10 entity tb_semaforos is
11 end tb_semaforos;
12 
13 architecture behavioral of tb_semaforos is
14 
15     constant SIM_TIME : time := 200000 ms;
16     constant N_TB : natural := 31;
17 
18     signal tb_rst           : std_logic;
19     signal tb_clk           : std_logic := '0';
20     signal tb_rojo_1        : std_logic;
21     signal tb_amarillo_1    : std_logic;
22     signal tb_verde_1       : std_logic;
23     signal tb_rojo_2        : std_logic;
24     signal tb_amarillo_2    : std_logic;
25     signal tb_verde_2       : std_logic;
26 
27 begin
28 
29     tb_rst <= '0', '1' after 1 ns, '0' after 20 ns;
30     tb_clk <= not tb_clk after 500 ms; -- Clock con freq : 50 MHz
31 
32 
33 
34     I1: entity work.semaforos(behavioral)
```

```
35      port map(
36          rst         => tb_rst,
37          clk         => tb_clk,
38          rojo_1      => tb_rojo_1,
39          amarillo_1  => tb_amarillo_1,
40          verde_1     => tb_verde_1,
41          rojo_2      => tb_rojo_2,
42          amarillo_2  => tb_amarillo_2,
43          verde_2     => tb_verde_2
44      );
45
46  end behavioral;
```

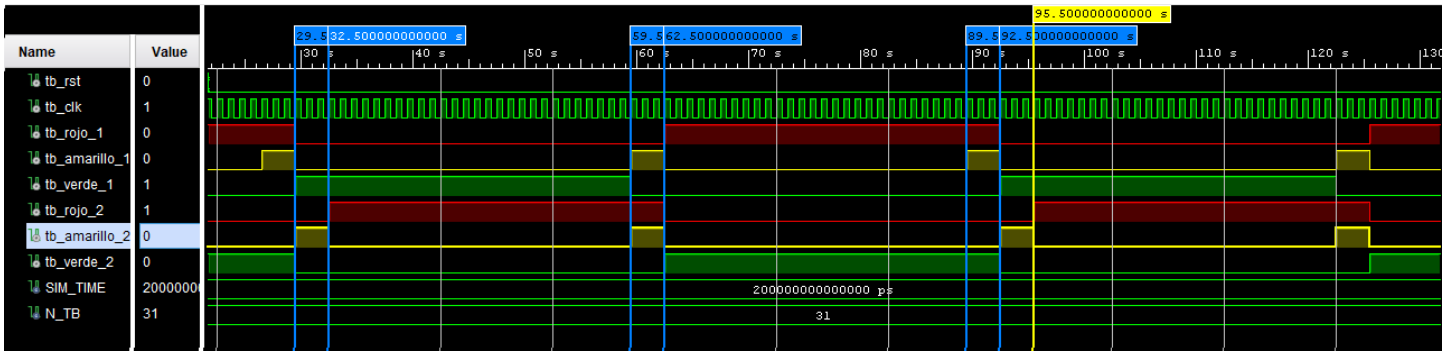

Figura 2: Simulación de la implementación

6. Síntesis

Finalmente para realizar la síntesis de la implementación, se utilizo el programa Vivado sobre el dispositivo FPGA xc7a15tftg256-1.

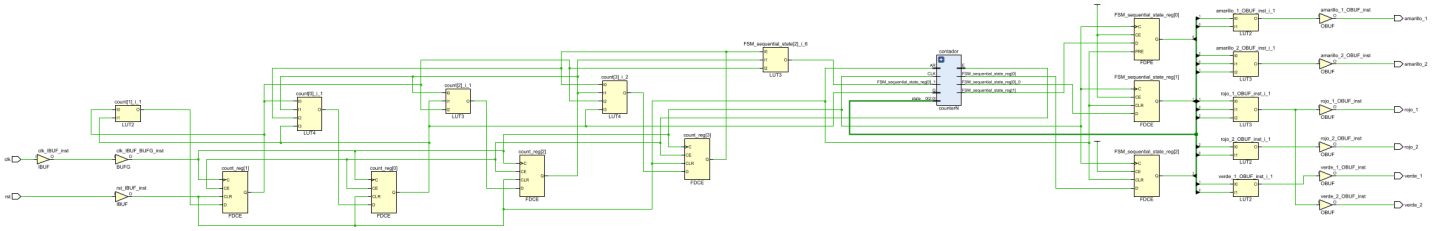

Figura 3: Sintesis de la implementación

Resource	Utilization	Available	Utilization %
LUT	69	10400	0.66
FF	67	20800	0.32
IO	9	170	5.29

Figura 4: Recursos utilizados

En las figura 3 y 4 se muestran una representación esquemática y la cantidad de recursos utilizados al realizar la implementación.

7. Links

Repositorio en GitHub