Proyecto final: Sistema de semáforos

Ian G. Cañas Fdez. — 1092228; Aimée R. Duran Lapaix — 1092944; Enyer A. Valenzuela L. — 1089614
Instituto Tecnológico de Santo Domingo (INTEC)
Santo Domingo, D.N. República Dominicana
Electrónica Digital — INL349/INL349L

***Resumen*—La electrónica digital es la rama de la electrónica más moderna, pues basada en ella, estudia los sistemas electrónicos en los que se codifica la información. La electrónica digital es conocida por muchos como la introducción al mundo abstracto, pero nunca menos real.**

El presente trabajo describe el proceso de análisis y desarrollo de un sistema de 12 semáforos distribuidos en tres intersecciones, procurando el uso de las herramientas aprendidas en la asignatura para su optimización y simplificación.

***Abstract*— Digital electronics is the most modern Branch of electronics systems, since it is based on it, it studies electronics circuits in which information is encoded. Digital electronics is known to many as the introduction to abstract world, without being less real.**

Analysis process and 12 traffic lights distributed in three intersections system development are described in this work, trying to use the tools learned in the subject for its optimization and simplification.

I. Introducción

Nos hemos visto presentes ante la necesidad de diseñar un sistema de semáforos para un total de tres intersecciones. Entre esta terna tenemos un total de doce semáforos en direcciones este-oeste, norte-sur y sus viceversas como se muestra en la figura I.

Figura I. *Configuración vial.*

El sistema de semáforos debe poseer cierta sincronía de forma que mantenga el orden y la simplicidad del tránsito. Para lograr este objetivo, es necesario determinar las necesidades, especificaciones y limitaciones del sistema. Esto para no omitir ningún dato importante a la hora de hacer el análisis para próximamente programarlo en VHDL e implementarlo en la tarjeta FPGA.

II. Objetivos

- Aplicar los conocimientos de electrónica digital a un proyecto de diseño.
- Fomentar el análisis crítico de los estudiantes al trabajar en equipo.
- Diseñar un sistema de semáforos síncronos haciendo uso de máquinas de estado.

III. Problema

Se necesita que el equipo implementa un sistema de semáforos síncronos en la tarjeta FPGA Nexys A7 provista a inicios del presente trimestre. Dicho sistema debe adaptarse a las siguientes especificaciones:

- Los semáforos en la vía principal deben poseer las siguientes duraciones por color:
  - Verde: 8 segundos
  - Amarillo: 3 segundos
  - Rojo: A definir por el equipo
- Los semáforos de las vías secundarias deben poseer las siguientes duraciones por color:
  - Verde: A definir por el equipo
  - Amarillo: 2 segundos
  - Rojo: A definir por el equipo
- Los semáforos de la vía principal deben tener un *delay* de 4 segundos entre ellos.

IV. Instrumentos

Utilizaremos el software para desarrollo de código de descripción de hardware, Vivado, cuya interfaz gráfica se muestra en la figura II.

Figura II. *Interfaz de Vivado.*

Cuyo código será cargado en la tarjeta de desarrollo Nexys A7.

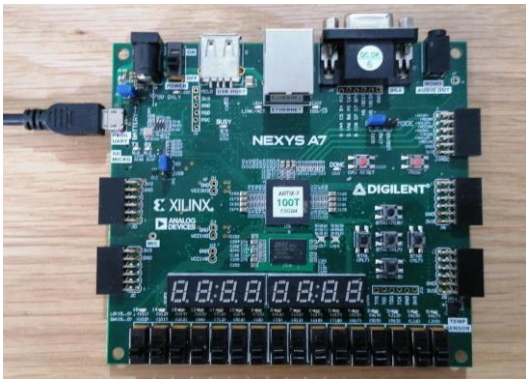


Figura III. *Tarjeta de desarrollo.*

V. Análisis

Antes de demostrar nuestro análisis, resulta preciso hacer las siguientes especificaciones de lugar:

1. No es posible tener los semáforos de ambas vías —principal y secundaria— en verde, porque podría ocasionar un accidente. En cambio, cuando los semáforos en dirección este-oeste y viceversa se encuentren en rojo, será el momento adecuado para que los semáforos de la vía secundaria se encuentren en verde o amarillo.
2. Los análisis han sido realizados tomando como centro de atención la vía principal por ser la más concurrida por los automóviles.
3. Al encontrarnos en una intersección, es posible que ambos semáforos de la misma vía se encuentren en el mismo estado en el mismo momento sin obstruir el tránsito.
4. La duración de las luces de los semáforos de la vía secundaria actúa en respuesta a los estados de los semáforos de la vía primaria.
5. Al evaluar el sistema solamente con los datos otorgados en el problema, los tiempos a ser determinados por el equipo se determinarían de forma automática por ser el tiempo necesario para que ambas vías de una intersección no se encuentren en el mismo estado.

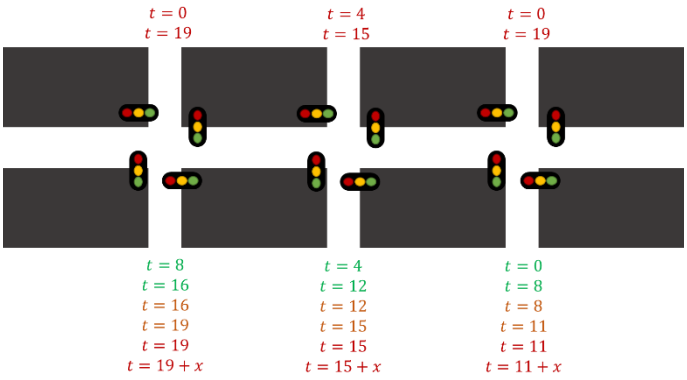


Figura IV. *Análisis de cambio I. Sin tiempo definitivo.*

Luego de evaluar las consideraciones previamente presentadas en la figura IV, se estableció un período mínimo de funcionamiento de un ciclo completo de 22 segundos, de los cuales 19 corresponden al ciclo de rojos en la primera intersección —es decir, el tiempo que dura un semáforo de dicha intersección en volver a ser rojo— en adición a al menos un segundo en rojo y dos segundos en amarillo para las vías secundarias. Debido a esto, el equipo ha considerado prudente extender el período de trabajo a 30 segundos. Al hacer esto y luego de calcular los valores correspondientes, nos encontramos ante la siguiente secuencia presente en la figura V.

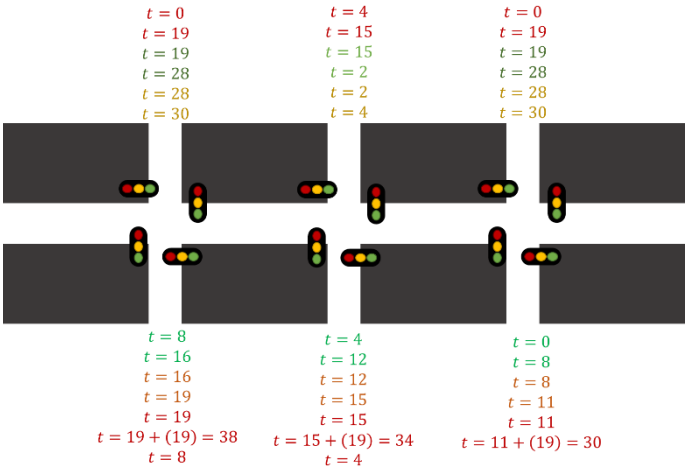


Figura V. *Análisis de cambio II. Tiempo definitivo.*

A continuación, el equipo decidió desplegar los estados del sistema en un vector de cinco valores. Los primeros tres valores representan los semáforos de la vía primaria, mientras los últimos dos valores del estado representan los semáforos de las vías secundarias. Dicho análisis se observa de manera adecuada en la siguiente figura.

t	S_1	S_2	S_3	S_4	S_5	S_6
0	V	R	R	R	V	R
2	V	R	R	R	A	R
4	V	V	R	R	R	R
8	A	V	V	R	R	R
11	R	V	V	R	R	R
12	R	A	V	R	R	R
15	R	R	V	R	V	R
16	R	R	A	R	V	R
19	R	R	R	V	V	V
28	R	R	R	A	V	A

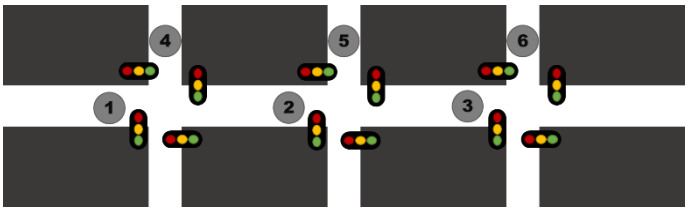


Figura VI. *Tabla de estado de semáforos en función del tiempo.*

Como los semáforos 4 y 6 siempre tienen el mismo valor, podemos unirlos obtenido como resultado la figura VII, la cual representa una forma simplificada de la figura VI presentada previamente.

t	S_1	S_2	S_3	S_4	S_5
0	V	R	R	R	V
2	V	R	R	R	A
4	V	V	R	R	R
8	A	V	V	R	R
11	R	V	V	R	R
12	R	A	V	R	R
15	R	R	V	R	V
16	R	R	A	R	V
19	R	R	R	V	V
28	R	R	R	A	V

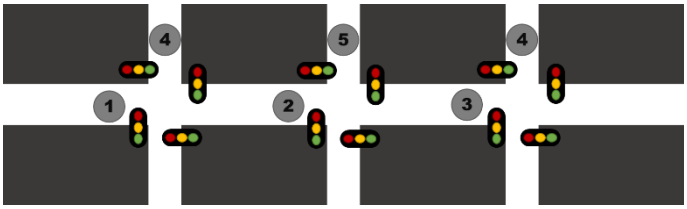


Figura VII. *Tabla de estado de semáforos simplificada.*

VI. Diseño

Luego de ejecutar el análisis anterior, el equipo arribó al siguiente resultado: una máquina de estados de Moore, cuya entrada es un bus lógico de 5 bits que representaría el tiempo y se limitaría al $t = 30\ s$. En la figura VIII se muestra la máquina de estados que representa este sistema, la cual ha sido representada con una relación unidireccional, debido a que un semáforo no puede retroceder desde su estado hacia un estado anterior, sino que volverá a dicho estado una vez haya completado un ciclo.

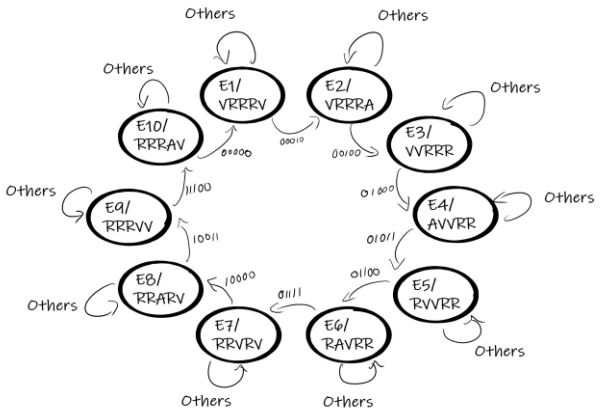


Figura VIII. *Máquina de estados.*

A continuación, para mejor comprensión del código, en la figura IX se presenta el diagrama de flujo del algoritmo empleado. En este se muestra mejor la importancia de nuestra única entrada, el reloj, debido a que gracias a esta es que se asignan los diez estados con los que cuenta nuestro sistema para luego proceder a imprimirlo con leds.

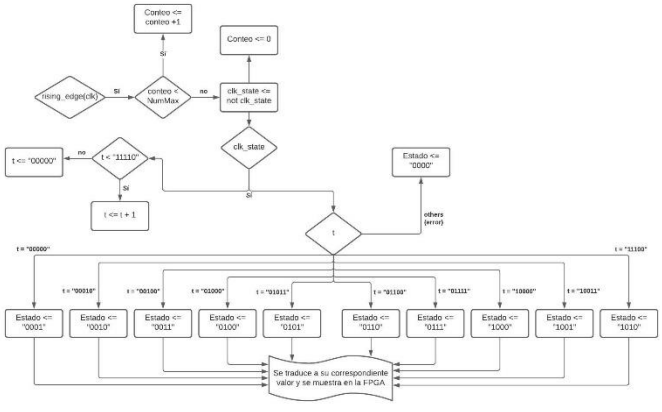


Figura IX. *Diagrama de Flujos.*

Para la implementación en la FPGA solo serán utilizados los leds y el display de 7 segmentos señalados en la figura X.

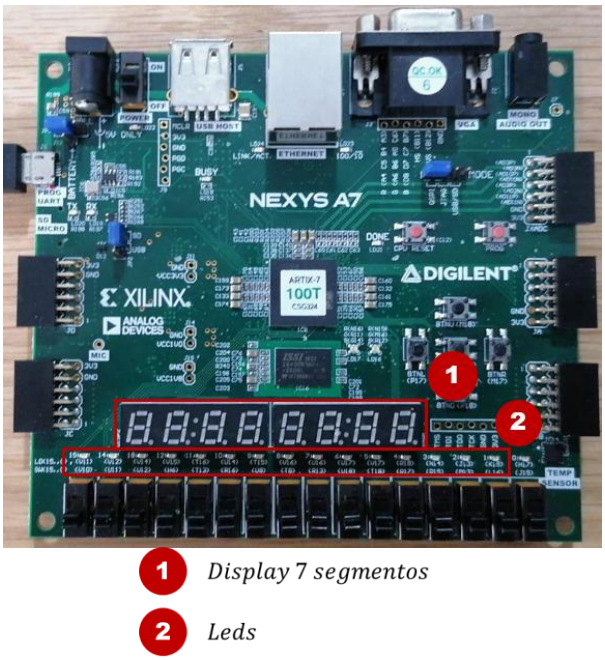


Figura X. *Componentes utilizados de la tarjeta.*

VII. Código y testbench

Una vez decidido el funcionamiento esperado de nuestro sistema de semáforos fue tiempo de migrarlo a VHDL para poder implementarlo en la FPGA. Se realizó un código principal, el cual posteriormente sería implementado, sin embargo, para comprobar que dicho código obtenía los resultados esperados, se realizo un testbench de dicho código.

A. Código para implementación

En primer lugar, se procedió a hacer la importación de las librerías como ya es costumbre. En este caso importamos las librerías *std_logic_1164*, *numeric_std*, *std_logic_arith* y *std_logic_unsigned* debido a que fueron las consideradas necesarias para implementar el código de manera exitosa según lo planeado anteriormente. Se utilizaron cinco puertos, donde una sola señal era una entrada mientras las cuatro restantes eran señales de salida. La entrada anteriormente mencionada es una señal de reloj, al tiempo que las salidas son: *disp*, el vector que nos indicará cuáles segmentos va a ser necesario encender en el *display* de siete segmentos para mostrar el mensaje deseado, *num7seg*, otro vector que nos señala el valor numérico a ser impreso por la señal *display*, *ledmodo*, una señal que nos indica cuándo el reloj ha hecho un cambio útil para el funcionamiento y, por último, *sem*, un vector que comprende los valores de los cinco semáforos para proceder a imprimirlos.

```
1  -- Librerías.
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.numeric_std.ALL;
5  use IEEE.std_logic_arith.ALL;
6  use IEEE.std_logic_unsigned.ALL;
7
8  -- Puertos utilizados.
9  entity PF is
10   Port (
11         clk: in std_logic; -- Reloj
12         Disp: out std_logic_vector (7 downto 0); -- Vector que establecerá cuáles displays encender
13         Num7Seg: out std_logic_vector (6 DOWNTO 0); -- Código del número correspondiente al 7 Segmentos
14         LEDMODO: out std_logic;
15         Sem: out std_logic_vector (14 downto 0)
16         );
17 end PF;
```

Figura XI. *Código VHDL. Parte I.*

A continuación, procedemos con la creación de las señales auxiliares. Tenemos nuestras señales constantes *NumMax* y *NumMax2*, las cuales llevan un conteo junto con *conteo* y *conteo2* de los flancos de subida de nuestra señal del reloj. La señal *clk_state* descrita a continuación es la que permite que el *display* cambie de estado. La señal *Estado* es la que indica en cuál etapa se encuentra el código en el momento, permitiendo así a las señales de salida tomar su valor adecuado. Por último, tenemos el arreglo *Lights*, en el cual indicamos los valores de nuestros semáforos representados con los caracteres R, A y V.

```
21  ------------------------------------------------------------------
22  -- Creación de señales.
23      constant NumMax: INTEGER := 49_999_999;
24      constant NumMax2: INTEGER := 49999;
25
26      signal conteo: INTEGER range 0 to NumMax;
27      signal conteo2: INTEGER;
28
29      signal clk_state: std_logic; -- Permite el cambio de display.
30      signal clk_state100: std_logic;
31
32      signal t: std_logic_vector (4 downto 0) := "00000";
33      signal cambio, cambio2: std_logic_vector (2 downto 0);
34      signal Estado: std_logic_vector (3 downto 0) := "0001";
35
36      --Declaración del tipo asociado a los estados.
37      type Color IS (R, A, V);
38      type Colores is array (0 to 4) of Color;
39      signal Lights: Colores;
40  ------------------------------------------------------------------
```

Figura XII. *Código VHDL. Parte II.*

A esto le sucede la función RAV7Seg, la cual transforma los caracteres 'R', 'A', 'V' en valores para el display de siete segmentos. Esta recibe el color y hace uso de una sentencia *with/select* para hacer la asignación previamente mencionada.

```
42  -- Función que va a presentar letra en 7 segmentos.
43  function RAV7Seg (
44      BCD: Color) -- Código binario del dígito.
45      return std_logic_vector is
46
47      variable Cod7Seg: std_logic_vector (6 downto 0);
48
49      begin
50      with BCD select -- Selección de la combinación de ánodos en el display.
51          Cod7Seg :=  ("0001000") WHEN (A), -- A
52                      ("1111010") WHEN (R), -- R
53                      ("1000001") WHEN (V), -- V
54                ("1111111") WHEN OTHERS;
55      return Cod7Seg; -- Devolución del respectivo valor.
56  end function;
```

Figura XIII. *Código VHDL. Parte III.*

Procedemos a declarar el proceso *CambiaEstados*, dicho proceso determina cuándo ha pasado el tiempo adecuado para cambiar de estado según la señal *clk_state* lo indique. Al asignar el valor del estado también aprovechamos y cambiamos la señal *t* para que el tiempo siga corriendo.

```
58  ------------------------------------------------
59  begin
60
61  -- Máquina de estado; responde al estado actual y la entrada.
62  CambiaEstados: process(clk_state)
63  begin
64      if clk_state'event and clk_state = '1' then
65          case Estado is
66              when "1010" =>
67                  if t = "00000" then
68                      Estado <= "0001";
69                      t <= t + 1;
70                  elsif t = "11101" then
71                      t <= "00000";
72                      Estado <= "1010";
73                  else
74                      t <= t + 1;
75                      Estado <= "1010";
76                  end if;
77
78              when "0001" =>
79                  if t = "00010" then
80                      Estado <= "0010";
81                  else
82                      Estado <= "0001";
83                  end if;
84
85                  t <= t + 1;
86
87              when "0010" =>
88                  if t = "00100" then
89                      Estado <= "0011";
90                  else
91                      Estado <= "0010";
92                  end if;
93
94                  t <= t + 1;
95
96              when "0011" =>
97                  if t = "01000" then
98                      Estado <= "0100";
99                  else
100                     Estado <= "0011";
101                 end if;
102
103                 t <= t + 1;
104
105             when "0100" =>
106                 if t = "01011" then
107                     Estado <= "0101";
108                 else
109                     Estado <= "0100";
110                 end if;
111
112                 t <= t + 1;
113
114             when "0101" =>
115                 if t = "01100" then
116                     Estado <= "0110";
117                 else
118                     Estado <= "0101";
119                 end if;
120
121                 t <= t + 1;
```

```
123             when "0110" =>
124                 if t = "01111" then
125                     Estado <= "0111";
126                 else
127                     Estado <= "0110";
128                 end if;
129
130                 t <= t + 1;
131
132             when "0111" =>
133                 if t = "10000" then
134                     Estado <= "1000";
135                 else
136                     Estado <= "0111";
137                 end if;
138
139                 t <= t + 1;
140
141             when "1000" =>
142                 if t = "10011" then
143                     Estado <= "1001";
144                 else
145                     Estado <= "1000";
146                 end if;
147
148                 t <= t + 1;
149
150             when "1001" =>
151                 if t = "11100" then
152                     Estado <= "1010";
153                 else
154                     Estado <= "1001";
155                 end if;
156
157                 t <= t + 1;
158
159             when others =>
160                 t <= "00000";
161                 Estado <= "0000";
162
163         end case;
164
165
166     end if;
167 end process;
```

Figura XIV. *Código VHDL. Parte IV.*

Agregando a lo anterior, se asigna a través de una sentencia *with/select* los cinco valores del array *Lights* que representan todos los semáforos de nuestro sistema. Asimismo, se asignan los valores del vector *Sem* que luego procederán a imprimirse a través de los leds de la FPGA. Inmediatamente, se describe el reloj mediante un LED que indica el tempo (tic… tac…) de un segundo.

```
169 -- "Traduce" los estados a los valores
170 -- que le corresponde en un array de datos.
171 with Estado select
172     Lights <=   (V, R, R, R, V) when "0001",
173                 (V, R, R, R, A) when "0010",
174                 (V, V, R, R, R) when "0011",
175
176                 (A, V, V, R, R) when "0100",
177                 (R, V, V, R, R) when "0101",
178                 (R, A, V, R, R) when "0110",
179
180                 (R, R, V, R, V) when "0111",
181                 (R, R, A, R, V) when "1000",
182
183                 (R, R, R, V, V) when "1001",
184                 (R, R, R, A, V) when "1010",
185
186                 (R, R, R, R, R) when others;
187 LEDMODO <= clk_state; -- Muestra un tic-tac del segundero.
```

Figura XV. *Código VHDL. Parte V.*

Por último, indicamos los procesos que van a generar las frecuencias pertinentes, como la de cada segundo y la de cambio de display para el efecto visual.

```
Reconteo: process(CLK) -- Manda una señal cada medio segundo al modo de tic-tac.
begin
    if CLK'event and CLK = '1' then
        if conteo < NumMax then
            conteo <= conteo + 1;
        else
            clk_state <= not clk_state;
            conteo <= 0;
        end if;
    end if;
end process;

ReconteoDisplay: process(clk) -- Manda una señal cada 10 ms, para el cambio de display utilizado para la ilusión óptica.
begin
    if clk'event and clk = '1' then
        if conteo2 < NumMax2 then
            conteo2 <= conteo2 + 1;
        else
            clk_state100 <= not clk_state100;
            conteo2 <= 0;
        end if;
    end if;
end process;

cambiador: process(clk_state100) -- Hacer per se el conteo solicitado para tener dichos displays encendidos.
begin
    if clk_state100'event and clk_state100 = '1' then
        if Cambio < "101" then
            Cambio <= Cambio + 1;
        else
            Cambio <= "001";
        end if;
    end if;
end process;
```

```vhdl
-- Muestra varios dígitos en los displays 7 segmentos de una forma cíclica.
Muestra_display: process(cambio)
begin
    if Cambio = "000" then
        Disp <= "11111101";
    elsif Cambio = "001" then
        Disp <= "11111011";
        Num7Seg <= RAV7Seg(Lights(4));
    elsif Cambio = "010" then
        Disp <= "11110111";
        Num7Seg <= RAV7Seg(Lights(3));
    elsif Cambio = "011" then
        Disp <= "11101111";
        Num7Seg <= RAV7Seg(Lights(2));
    elsif Cambio = "100" then
        Disp <= "11011111";
        Num7Seg <= RAV7Seg(Lights(1));
    elsif Cambio = "101" then
        Disp <= "10111111";
        Num7Seg <= RAV7Seg(Lights(0));
    elsif Cambio = "110" then
        Disp <= "11111110";
    else
        Disp <= "01111111";
    end if;
end process;
```

Figura XVI. *Código VHDL. Parte VI.*

B. *Testbench*

En primer lugar, creamos un componente PF que cuenta con los mismos puertos que nuestro código principal (*clk, Disp, Num7Seg, LEDMODO y Sem*) y, de igual forma, un conjunto de señales locales con las cuales instanciaríamos dicho componente.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
end testbench;

architecture tb of testbench is

-- Componente del código principal
component PF is
port(
    clk     :  in std_logic;
    LEDMODO : out std_logic;
    Num7Seg : out std_logic_vector (6 DOWNTO 0);
    Disp    : out std_logic_vector (7 downto 0);
    Sem     : out std_logic_vector (14 downto 0)
    );
end component;

-- Señales que contribuirán al mapeo más adelante
signal clk_in, ledmodo_out: std_logic;
signal num7seg_out  : std_logic_vector (6 downto 0);
signal disp_out     : std_logic_vector (7 downto 0);
signal sem_out      : std_logic_vector (14 downto 0);

signal stop : boolean := false; -- Señal para parar el tiempo
constant period: TIME := 10NS; -- Tiempo para tener un ciclo de reloj

begin
  -- Instancia del componente
  DUT: PF port map(clk_in, ledmodo_out, num7seg_out, disp_out, sem_out);
```

Figura XVII. *Testbench. Parte I.*

Luego se procede a crear el proceso que genera los pulsos de reloj en un período total de 10 nanosegundos. Inmediatamente después tenemos nuestro proceso *state_check* el cual evalúa las salidas del vector *sem* en los tiempos indicados en la figura 7.

```vhdl
-- Proceso que genera los cambios en el reloj
clock_gen : process
begin
    while not stop loop
        clk_in <= '0';
        wait for period/2;
        clk_in <= '1';
        wait for period/2;
    end loop;
    wait;
end process clock_gen;
```

```vhdl
state_check: process
  begin

    assert(sem_out <= "001100100100001") report "Estado 1 incorrecto" severity error;
    wait for period*2;

    assert(sem_out <= "101001001000010") report "Estado 2 incorrecto" severity error;
    wait for period*2;

    assert(sem_out <= "100100100001001") report "Estado 3 incorrecto" severity error;
    wait for period*4;

    assert(sem_out <= "100100001001010") report "Estado 4 incorrecto" severity error;
    wait for period*3;

    assert(sem_out <= "100001001100100") report "Estado 5 incorrecto" severity error;
    wait for period*1;

    assert(sem_out <= "100100001100100") report "Estado 6 incorrecto" severity error;
    wait for period*3;

    assert(sem_out <= "100100001100001") report "Estado 7 incorrecto" severity error;
    wait for period*1;

    assert(sem_out <= "100100010100001") report "Estado 8 incorrecto" severity error;
    wait for period*3;

    assert(sem_out <= "100100100001001") report "Estado 9 incorrecto" severity error;
    wait for period*9;

    assert(sem_out <= "100100100010001") report "Estado 10 incorrecto" severity error;

    stop <= true;
    wait;
end process state_check;
end tb;
```

Figura XVIII. *Testbench. Parte II.*

VIII. Resultados

Previo a la implementación física del código presentado previamente, era necesario evaluar las respuestas obtenidas a través del testbench. Los resultados esperados eran, con solo la señal del reloj como entrada, tener los estados cambiando según el tiempo indicado anteriormente como adecuado. Los resultados obtenidos al ejecutar el análisis de ondas fueron los siguientes:

Figura XIX. *Estudio de ondas,* $0 \leq t \leq 14$

Figura XX. *Estudio de ondas,* $14 \leq t \leq 28$

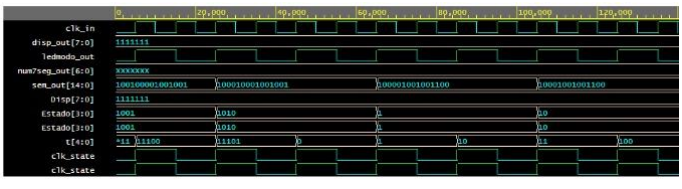


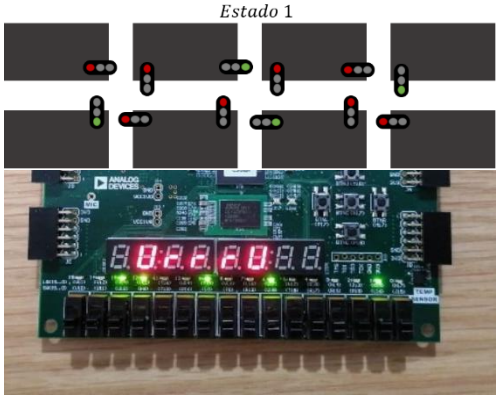
Figura XXI. *Estudio de ondas,* $28 \leq t \leq 30, 0 \leq t \leq 4$

Nota: Al terminar un ciclo con período de 30 segundos, el mismo vuelve a iniciar.

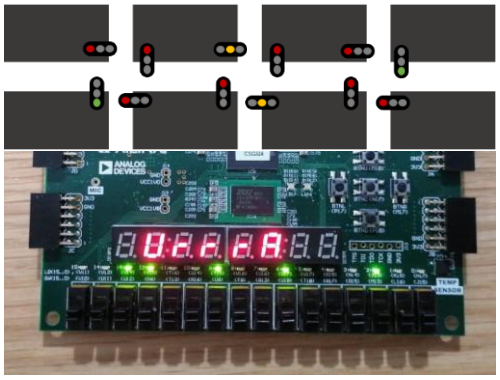
En primera instancia, en las figuras XIX a XXI parece que la simulación se desfasa por un segundo con respecto al tiempo. Sin embargo, luego de analizar el código se demostró que esto no es así. Lo que realmente sucede es que, al asignar un nuevo estado, inmediatamente se hace el cambio a la señal t, lo cual provoca el efecto mencionado anteriormente. El equipo decidió desestimar esta situación por no tener ningún efecto erróneo hacia los resultados.

Luego de tomar esta prueba como exitosa, se procedió a implementar y describir en físico el código probado, donde se obtuvieron los resultados mostrados a continuación.

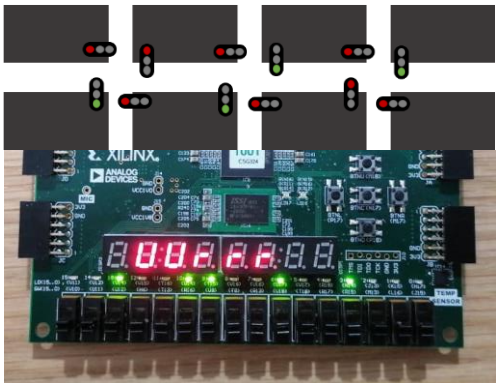
Estado 1



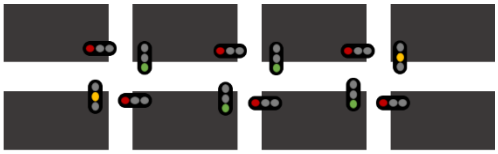
Estado 2

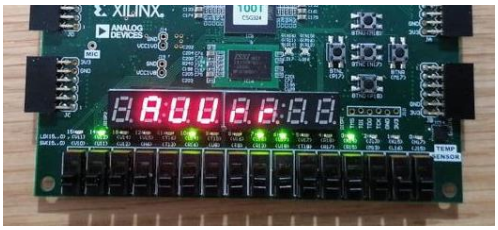


Estado 3

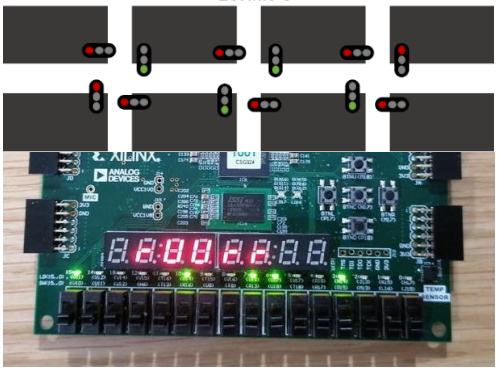


Estado 4

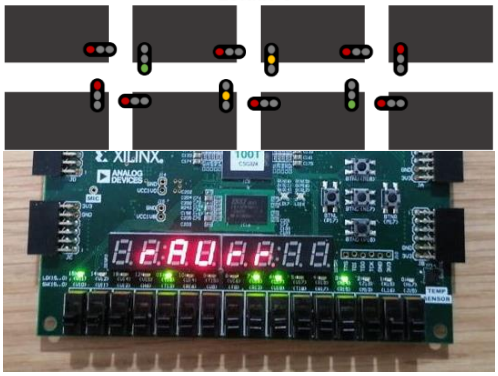




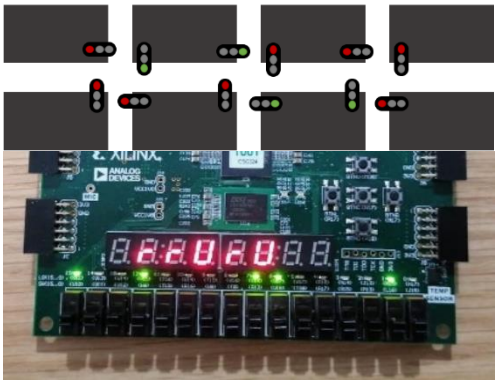
Estado 5



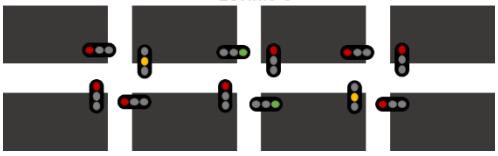
Estado 6



Estado 7



Estado 8



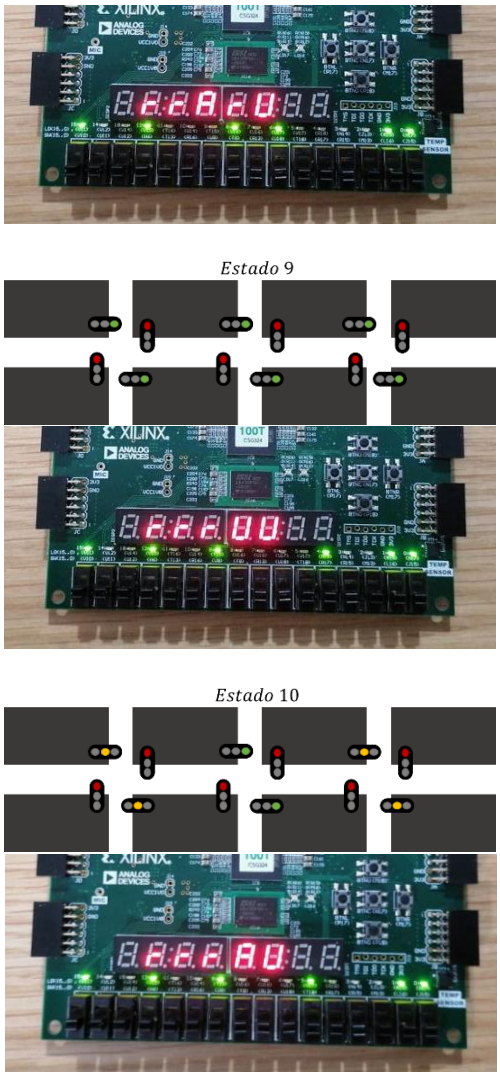


Figura XXII. *Resultados en tarjeta FPGA.*

IX. Conclusiones

Habiendo pasado por el proceso e implementación del sistema presentado, se puede comprobar sin temor a equivocación la importancia del funcionamiento y comprensión de las máquinas de estados, ya que permiten el uso de las memorias o historial de cambios en caso de ser preciso.

El equipo fue capaz de crear un sistema de doce semáforos separados en tres intersecciones. Dichos semáforos se encontraban en sincronía, trabajando a la par para optimizar el sistema de tránsito en dicho entorno. Los semáforos funcionan a base solamente del reloj integrado en nuestra tarjeta FPGA.

Se pudo observar un comportamiento tal cual el esperado, las respuestas de estado respecto a la entrada (el tiempo) fueron muy acertadas, especialmente con los valores predichos mediante el testbench. Además, se pudo observar cómo el sistema que inicialmente se imaginaba de 12 semáforos pudo ser reducido importantemente, al alcance de 5 semáforos.

X. Referencias bibliográficas

[1] UNIVERSIDAD AUTÓNOMA DE ENTRE RÍOS, «Tema 1 - Introducción a los sistemas digitales,» [En línea]. Available: http://quegrande.org/apuntes/grado/1G/FCG/teoria/10-11/tema_1_-_introduccion_a_los_sistemas_digitales.pdf. [Último acceso: 07 2021].

[2] M. Sánchez-Élez, Introducción a la programación en VHDL, Madrid, Facultad de informática: Universidad Complutense de Madrid, 2014.

[3] Digilent, Nexys 4 FPGA Board Reference Manual, Pullman. : Digilent, 2016.

[4] E. A. Lee, Introduction to embedded systems, 2 ed., MIT Press, 2017.

XI. Anexos

El video del funcionamiento del sistema presentado, así como los códigos utilizados se encuentran publicados en OneDrive. En caso de estar interesado, usted podrá acceder a ellos a través del siguiente enlace: https://estintecedu-my.sharepoint.com/personal/1092228_est_intec_edu_do/_layouts/15/onedrive.aspx?originalPath=aHR0cHM6Ly9lc3RpbnRlY2VkdS1teS5zaGFyZXBvaW50LmNvbS86ZjovZy9wZXJzb25hbC8xMDkyMjI4X2VzdF9pbnRlY19lZHVfZG8vRXNXNjNmUnFvOFZPbUJmWVc0Z1hpWTRCY203eTFqWnVkY2Yzc1NuR3RqcG1LQT9ydGltZT1rTE1iUHNKQjJVZw&id=%2Fpersonal%2F1092228_est_intec_edu_do%2FDocuments%2FGRUPO%20F%20-%20PF%20-%20ED