FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Sistemas Digitales

Trabajo Práctico N1

Implementación de un sistema secuencial en VHDL

Denise Gayet 100828 dgayet@fi.uba.ar

29 de septiembre de 2023

Sistemas Digitales

Trabajo Práctico N1: Implementación de un sistema secuencial en VHDL

Denise Gayet 100828 dgayet@fi.uba.ar

Índice

1. Enunciado

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

2. Introducción

Para resolver el problema dado, se propuso un diseño que consiste en una combinación entre una maquina de estados y un circuito digital secuencial. La descripción del circuito se desarrolló en VHDL y el mismo se dividió en 5 módulos: un módulo contador de 1 segundo a partir de los ciclos de clock, un contador de N segundos para controlar la transición de estados, un módulo para la maquina de estados propiamente dicha, un módulo de salida que controla las luces del semaforo, y un módulo integrador. Además, para cada módulo se describió un testbench para simular y probar su correcto funcionamiento. Por último, se sintetizó el circuito sobre la FPGA xc7a15tftg256-1 mediante la herramienta de síntesis Vivado.

2.1. Circuito implementado

El circuito implementado se puede describir como una máquina de estados. Sin embargo, para simplificar el modelo se dividió el diseño en una máquina de estados encargada de establecer el estado del sistema, y un circuito a base de contadores encargado de producir la entrada de la maquina de estados, la cual marca la transición de la misma.

Por un lado, en la figura 1 se muestra el diagrama de la maquina de estados implementada:

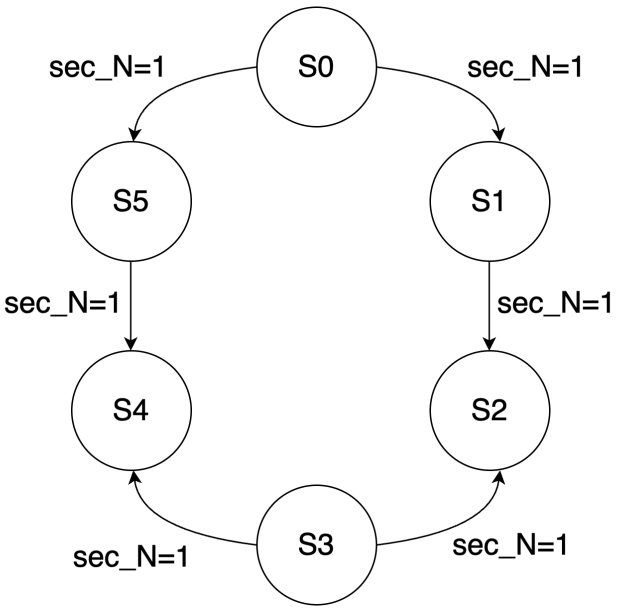

Figura 1: Diagrama de estados

La maquina de estados tiene 6 estados, definidos según las luces de ambos semáforos, que se corresponden con la tabla 1:

Estado	Semáforo 1	Semáforo 2	Tiempo de permanencia
S0	verde	rojo	30 s
S1	amarillo	rojo	3 s
S2	rojo	amarillo	3 s
S3	rojo	verde	30 s
S4	rojo	amarillo	3 s
S5	amarillo	rojo	30 s

Tabla 1: Estados de la FSM

La transición entre estados está dictada por la señal de entrada `sec_N`, la cual proviene de un contador que llega hasta 30 segundos o 3, en función del estado en que se encuentra el sistema.

Por otro lado, se diseñaron 2 contadores. El primero es un contador de 1 segundo a partir de ciclos de clock, el cual se muestra en la figura 2.

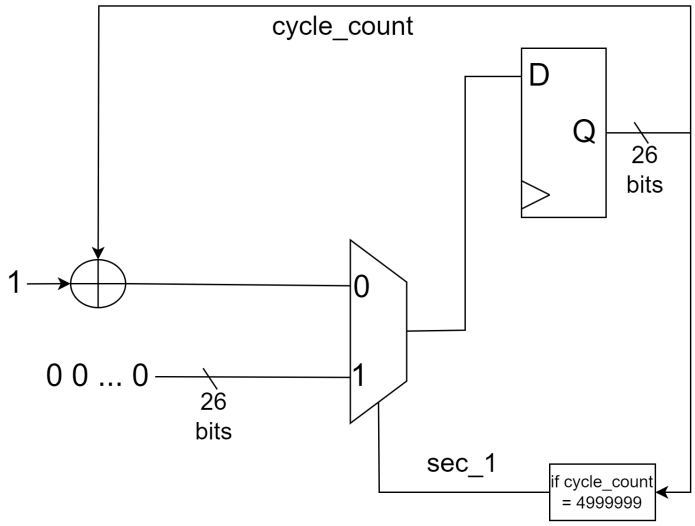

Figura 2: Contador de 1 segundo

El segundo contador depende del contador de 1 segundo, y cuenta hasta 3 o 30 dependiendo del estado. Su diagrama en bloques se muestra en la figura 3.

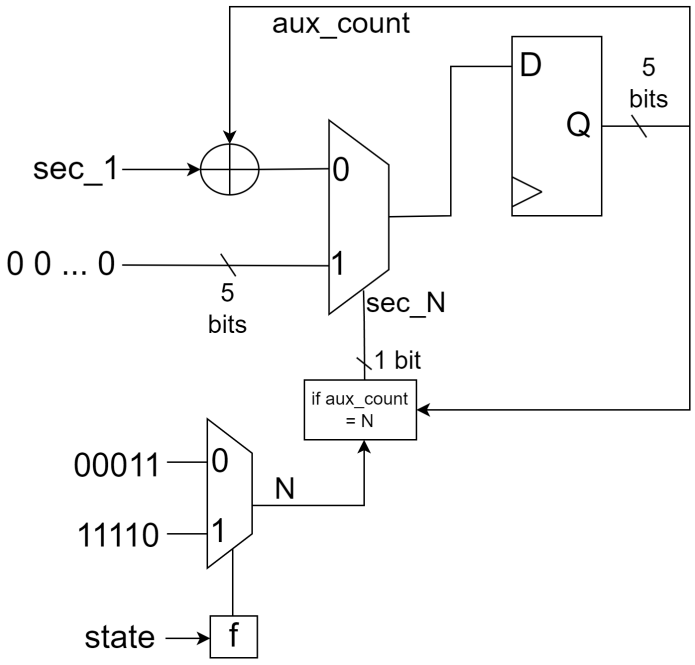

Figura 3: Contador de N segundos

Como se puede observar, el contador de N segundos recibe la señal `sec_1` y recibe `N`, el máximo número a contar.

La salida del contador es `sec_N`, la cual se usa como entrada a la maquina de estados para realizar las transiciones.

Por último, el módulo encargado de controlar las luces es un circuito puramente combinacional, que toma como entrada el estado del sistema.

Finalmente, abstrayéndose de la implementación circuital, en la figura 4 se muestra un diagrama en bloques del módulo integrador.

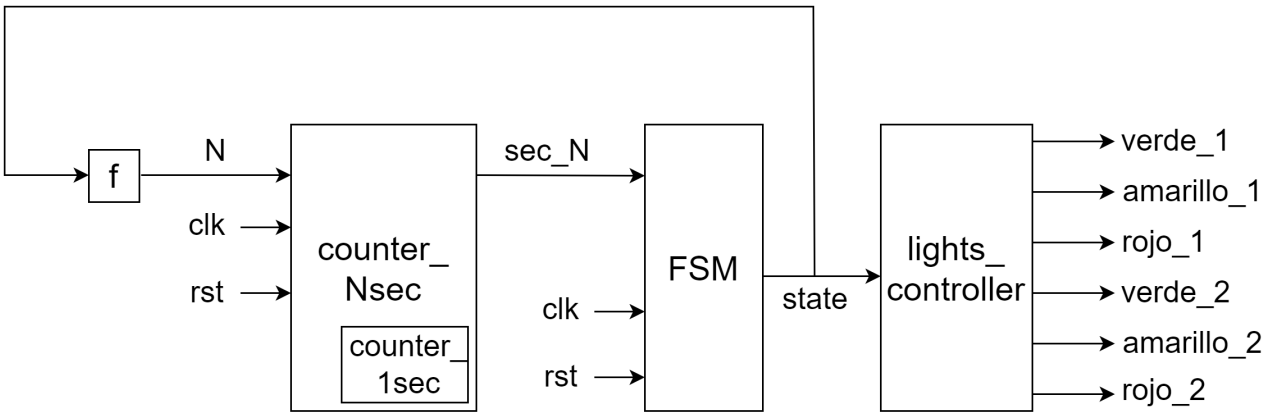

Figura 4: Diagrama del semáforo

Las entradas del semáforo son únicamente el clock y el reset, los cuales son globales (los mismos para todos los módulos), y las salidas son las mismas que las del módulo que controla las luces.

Si se tiene en cuenta que entre los contadores se necesitan 31 registros, y que, la maquina de estados necesita 3 registros dado que la cantidad de estados es 6, se deduce que en principio la cantidad de Flip Flops necesaria es 34.

3. Descripción en VHDL

3.1. Generalidades

Todos los módulos que describen circuitos secuenciales reciben como entrada la señal de clock, y transicionan en el flanco **ascendente** del mismo.

Además, dichos módulos tienen de entrada la señal de reset, que se activa al estar en alto y pone a todos los módulos en su estado inicial. El reset es **asincrónico**, es decir, funciona de manera independiente al flanco del clock. Ambas señales son globales.

3.2. Contador de 1 segundo

El módulo de este contador consiste en llevar la cuenta de ciclos de clock hasta llegar a 1 segundo. Como es un circuito secuencial, toma como entrada el clock de la FPGA y el reset. El clock es de 50 MHz, por lo tanto se necesitan 50 M ciclos para contar un segundo. Como la cuenta comienza en 0, se debe contar hasta 49999999. Cuando se llega a dicho numero, se pone en alto el flag `sec_1`, el cual actúa como selector de un multiplexor, permitiendo que se reinicie el contador. Además esta señal es la salida del contador.

A continación se muestra la descripción en VHDL.

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

entity counter_1sec is
    generic (
        Ncycles: natural := 50000000
    );
    port (
        clk_i : in std_logic;
        rst_i : in std_logic;
        sec_1 : out std_logic
    );
```

```
14 end;
15
16 architecture behavioral of counter_1sec is
17     signal cycle_count : unsigned(25 downto 0);
18 begin
19
20     sec_1 <= '1' when (cycle_count=Ncycles-1) else '0';
21
22     process(clk_i, rst_i)
23     begin
24         if rst_i='1' then
25             cycle_count <= (others => '0');
26         elsif rising_edge(clk_i) then
27             if (cycle_count = Ncycles-1) then
28                 cycle_count <= (others => '0');
29             else
30                 cycle_count <= cycle_count + 1;
31             end if;
32         end if;
33     end process;
34 end;
```

Cabe aclarar que la asignación de la señal `sec_1` se realiza en concurrencia con el proceso secuencial definido por `process()`, con lo cual, se pone en alto en el flanco ascendente de la señal `cycle_count` (al llegar a `Ncycles-1`).

3.3. Contador de N segundos

Este módulo toma como entrada el clock de la placa, el reset, y un valor N que indica la cantidad de segundos que se deben contar. Dentro del módulo se instancia el contador de 1 segundo y se toma la señal `sec_1` para incrementar la cuenta de la señal `aux_cuenta`, la cual se reinicia al llegar al valor $N - 1$.

La salida es la señal `sec_N`, la cual vale '1' cuando el contador llega al máximo y '0' en otro momento. Dicha salida luego alimenta a la maquina de estados para realizar la transición.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity counter_Nsec is
6      -- N is for the amount of bits of the output, so max. number possible is: 2**5 -1
7      port (
8          clk_i: in std_logic;
9          rst_i: in std_logic;
10         Ncount : in unsigned(4 downto 0);
11         sec_N: out std_logic
12     );
13 end;
14
15 architecture behavioral of counter_Nsec is
16     constant NCYCLES: natural := 5;
17     signal aux_count: unsigned(4 downto 0);
18     signal is_1sec: std_logic;
19 begin
20
21     sec_N <= '1' when ((aux_count = Ncount-1) AND (is_1sec='1')) else '0';
22     process (clk_i, rst_i)
23     begin
24         if rst_i = '1' then
25             aux_count <= (others => '0');
26         elsif clk_i='1' and clk_i'event then
27             if (is_1sec = '1') then
28                 if (aux_count = Ncount-1) then
```

```
                    aux_count <= (others => '0');
                else
                    aux_count <= aux_count + 1;
                end if;
            end if;
        end if;
    end process;

    I_C1sec: entity work.counter_1sec(behavioral)
        generic map (
            Ncycles => NCYCLES
        )
        port map (
            clk_i => clk_i,
            rst_i => rst_i,
            sec_1 => is_1sec
        );
end;
```

Como se puede observar, a `NCYCLES` se le asigna el valor 5, cuando en realidad debería ser 50 M. Esto se debe a que es más simple para verificar el sistema durante la simulación. De todos modos, para sintetizar el diseño se utilizó el valor real.

3.4. Maquina de estados

Como se explicó previamente, los estados de la FSM son 6 y estan descriptos en la tabla 1. Para describir dichos estados se utilizó un tipo personalizado `t_state`, definido en un package externo.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

package t_types is
    type t_state is (S0, S1, S2, S3, S4, S5);
end package t_types;
```

En cuanto a la descripción de la máquina de estados, ésta tiene como entrada el clock y el reset global, y la señal `sec_N` que sale del contador. Dicha señal es la encargada de controlar la transición de los estados de la FSM: cuando está en alto y ocurre un flanco ascendente del clock, se transiciona al próximo estado.

El estado inicial de la máquina es `S0`, que se corresponde con el **semaforo 1** en verde y el **semaforo 2** en rojo, tal como se indica en la tabla 1.

La salida de la FSM es el estado `state`, el cual se utiliza posteriormente en el módulo de salida, responsable de controlar las luces del semáforo.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.t_types.all;

entity fsm is
    port (
        clk_i       : in std_logic;
        rst_i       : in std_logic;
        sec_N      : in std_logic;
        state       : out t_state
    );
end fsm;

architecture behavioral of fsm is
    signal aux_state : t_state;

begin
    state <= aux_state;
```

```
20
21      process(clk_i, rst_i)
22      begin
23          if rst_i = '1' then
24              aux_state <= S0;
25          elsif rising_edge(clk_i) then
26              if sec_N = '1' then
27                  case aux_state is
28                      when S0 =>
29                          aux_state <= S1;
30                      when S1 =>
31                          aux_state <= S2;
32                      when S2 =>
33                          aux_state <= S3;
34                      when S3 =>
35                          aux_state <= S4;
36                      when S4 =>
37                          aux_state <= S5;
38                      when S5 =>
39                          aux_state <= S0;
40                  end case;
41              end if;
42          end if;
43      end process;
44  end behavioral;
```

3.5. Módulo de salida

La función de salida es un circuito puramente combinacional que toma como entrada el estado del sistema y tiene como salida las 6 señales propuestas en el enunciado: verde 1, amarillo 1, rojo 1, verde 2, amarillo 2 y rojo 2, las cuales se corresponden con las luces del semáforo 1 y semáforo 2 respectivamente.

De todos modos, aunque éste sea un circuito combinacional, también está sincrónizado con el clock, ya que depende únicamente del estado (el cual únicamente puede tener un cambio en un flanco ascendente del clock).

```
1   library IEEE;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4   use work.t_types.all;
5
6   entity lights_controller is
7       port (
8           state       : in t_state;
9           green_1     : out std_logic;
10          yellow_1    : out std_logic;
11          red_1       : out std_logic;
12          green_2     : out std_logic;
13          yellow_2    : out std_logic;
14          red_2       : out std_logic
15      );
16  end;
17
18  architecture behavioral of lights_controller is
19  begin
20
21      green_1 <= '1' when (state=S0) else '0';
22      yellow_1 <= '1' when ((state=S1) OR (state=S5)) else '0';
23      red_1 <= '1' when ((state=S2) OR (state=S3) OR (state=S4)) else '0';
24
25      green_2 <= '1' when (state=S3) else '0';
26      yellow_2 <= '1' when ((state=S2) OR (state=S4)) else '0';
27      red_2 <= '1' when ((state=S0) OR (state=S1) OR (state=S5)) else '0';
28
```

```
29
30 end behavioral;
```

3.6. Semáforo

Este módulo es el que integra los 4 descriptos previamente, instanciándolos de manera concurrente. Tiene como entrada el clock y el reset y como salida las señales de los semáforos. En este mismo módulo se asigna el valor de N para el contador de N segundos con un multiplexor cuyo selector depende del estado de la FSM. Como se definió previamente, N tiene que valer 30 (segundos) únicamente cuando alguno de los dos semáforos está en verde, es decir, en el estado S0 o S3. En cualquier otro estado, N vale 3.

```
1  library IEEE;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use work.t_types.all;
5
6  entity semaphore is
7      port (
8          clk_i : in std_logic;
9          rst_i : in std_logic;
10         green_1      : out std_logic;
11         yellow_1     : out std_logic;
12         red_1        : out std_logic;
13         green_2      : out std_logic;
14         yellow_2     : out std_logic;
15         red_2        : out std_logic
16     );
17 end semaphore;
18
19 architecture behavioral of semaphore is
20     constant SECS_G : unsigned := to_unsigned(30, 5);
21     constant SECS_YR: unsigned := to_unsigned(3, 5);
22     signal state : t_state;
23     signal is_Nsec : std_logic;
24     signal N : unsigned(4 downto 0);
25
26 begin
27     N_SEL: N <= SECS_G when ((state=S0) OR (state=S3)) else SECS_YR;
28
29     NC_I: entity work.counter_Nsec(behavioral)
30         port map (
31             clk_i => clk_i,
32             rst_i => rst_i,
33             Ncount => N,
34             sec_N => is_Nsec
35         );
36
37     L_CONT_I: entity work.lights_controller(behavioral)
38             port map(
39                 state => state,
40                 green_1 => green_1,
41                 yellow_1 => yellow_1,
42                 red_1 => red_1,
43                 green_2 => green_2,
44                 yellow_2 => yellow_2,
45                 red_2 => red_2
46             );
47
48     FSM_I: entity work.fsm(behavioral)
49             port map(
50                 clk_i => clk_i,
51                 rst_i => rst_i,
52                 sec_N => is_Nsec,
53                 state => state
```

```
54                 );
55 end behavioral;
```

3.7. Testbench

Este módulo se usa a modo de banco de pruebas, por lo que dicha entidad no posee puertos, es decir, no tiene entradas ni salidas. Su utilidad es únicamente durante la simulación, no forma parte de la sintetización del diseño.

En la arquitectura de este módulo se instancia el semaforo y se describe una señal de clock (con una frecuencia de 50 MHz) y de reset.

A continuación se muestra el *testbench* del sistema completo, pero a lo largo del trabajo se utilizaron distintos bancos de prueba para probar los módulos de manera independiente.

```
1  library IEEE;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use work.t_types.all;
5
6  entity semaphore_tb is
7  end;
8
9  architecture behavioral of semaphore_tb is
10 signal clk : std_logic := '1';
11 signal rst : std_logic := '1';
12
13 signal rojo_1 : std_logic;
14 signal amarillo_1 : std_logic;
15 signal verde_1 : std_logic;
16
17 signal rojo_2 : std_logic;
18 signal amarillo_2 : std_logic;
19 signal verde_2 : std_logic;
20
21 begin
22     clk <= not clk after 1 ns;
23     rst <= '0' after 2 ns;
24
25     SEMAPHORE: entity work.semaphore(behavioral)
26         port map (
27         clk_i       => clk,
28         rst_i       => rst,
29         green_1     => verde_1,
30         yellow_1    => amarillo_1,
31         red_1       => rojo_1,
32         green_2     => verde_2,
33         yellow_2    => amarillo_2,
34         red_2       => rojo_2
35         );
36
37 end behavioral;
```

4. Simulación

A continuación se muestran los resultados obtenidos de realizar la simulación de comportamiento sobre el *testbench* del semáforo.

Como se aclaró previamente, para poder simular el sistema se tomaron 5 flancos ascendentes del clock para contar 1 segundo, en vez de 50,000,000.

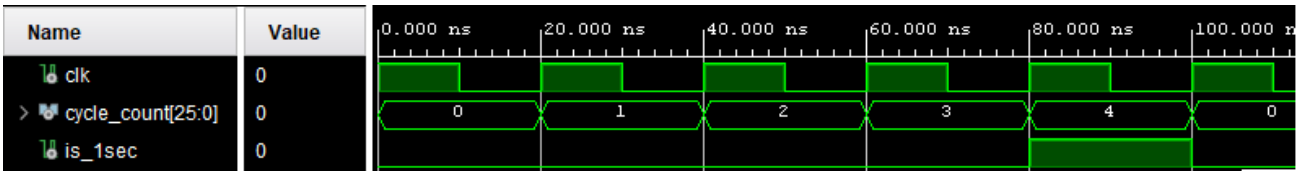

Figura 5: Waveform del contador de 1 segundo con 5 ciclos de reloj

Como se puede observar, el clock simulado (`clk`) con el testbench tiene un período de 20 ns, es decir una frecuencia de 50 MHz. A su vez, el contador de ciclos (`cycle_count`) cuenta 5 flancos ascendentes de reloj y una vez que llega al 5to ciclo se pone en alto la señal `is_1sec` y en el próximo flanco de reloj se reinicia el contador. Por lo tanto, 1 segundo se corresponde con 100 ns en la simulación.

En la figura 6 se muestra el funcionamiento del contador de N segundos. El registro de la cuenta se lleva en `aux_count`. Se puede observar que su valor se incrementa cada vez que se reinicia el contador de cyclos (`cycle_count`). Una vez que llega al valor máximo (en este caso 30-1) y que se pone en alto la señal `sec_1`, se pone en alto la señal `sec_N` y se reinicia el contador. Como `sec_N` controla la maquina de estados, en el próximo flanco ascendente de reloj cambia el estado del sistema, por lo que también cambia el valor de N (de 30 a 3). Luego, dado que el valor de N es 3, se cuenta hasta 3 segundos y se reinicia `aux_count`, demostrando así el correcto funcionamiento del contador.

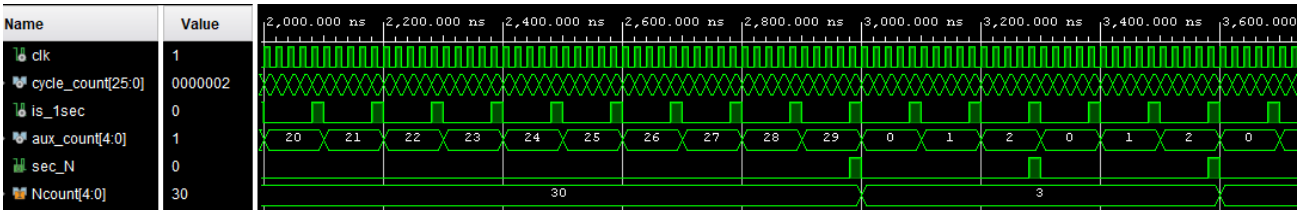

Figura 6: Waveform del contador de N segundos

Abstrayéndose de los contadores y analizando las transiciones en función de la señal `sec_N`, se muestra a continuación el funcionamiento de la maquina de estados.

El estado inicial de la máquina es S0 (correspondiente a verde_1 en '1' y rojo_2 en '1'), con lo cual, el tiempo de permanencia en dicho estado es de 30 segundos. En el caso de la simulación, la transición se debe dar a los 3 μs. Luego se tiene que transicionar a S1 en el cual el tiempo de permanencia es de 3 segundos, por lo que N cambia de valor a 3.

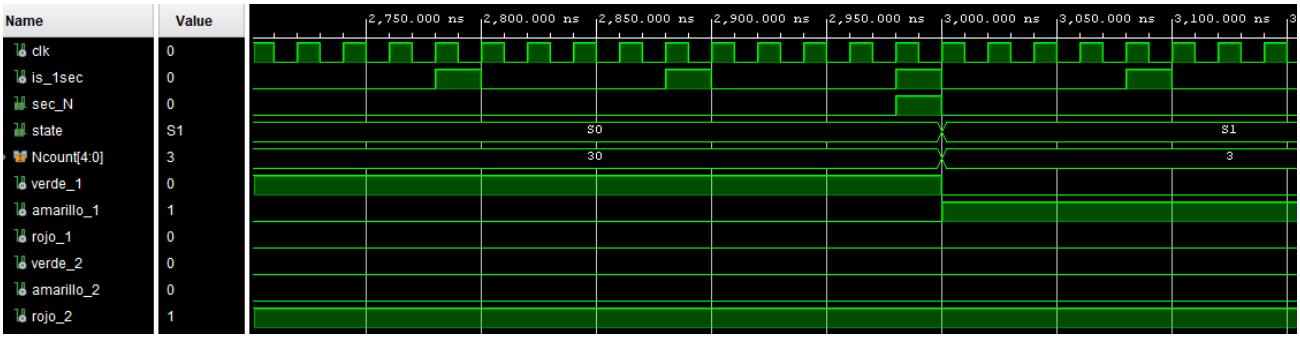

Figura 7: Transición de S0 a S1

Se puede observar que un flanco de reloj antes de 3 μs se pone en alto la señal `sec_N`. Por lo tanto, en 3 μs la maquina de estados recibe dicha señal, y transiciona de S0 a S1 (correspondiente a amarillo_1 en '1' y rojo_2 en '1'). En el mismo flanco de reloj, cambia de valor N, dado que en S1 el tiempo de permanencia es 3.

Luego, 3 segundos despues (0,3 μs en la simulación), se transiciona al estado S2 cuando la maquina de estados recibe nuevamente `sec_N` en alto. Sin embargo, en este caso no cambia el valor de N, dado que el tiempo de permanencia en el estado S2 sigue siendo de 3 segundos.

En este estado los valores de los semáforos se corresponden con rojo_1 en '1' y amarillo_2 en '1'.

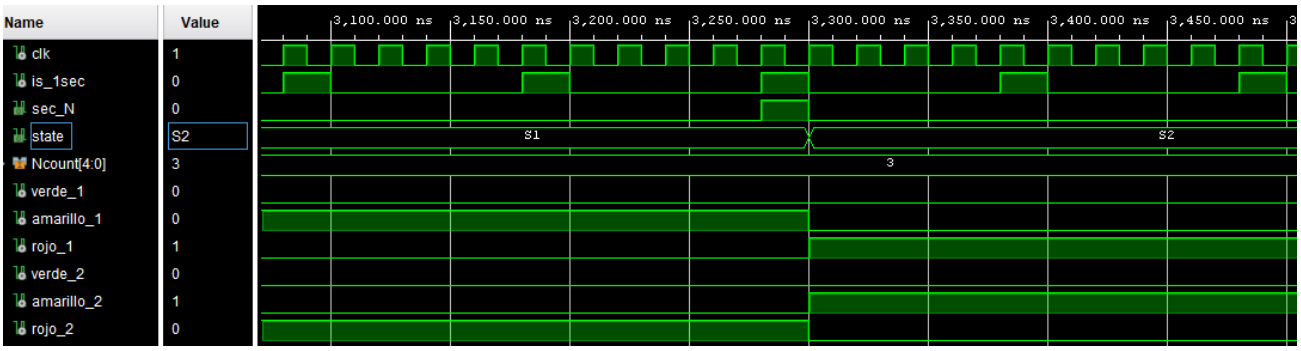

Figura 8: Transición de S1 a S2

Como el N se mantiene en el estado S2, 3 segundos después (o lo equivalente en la simulación) se transiciona al estado S3 dado que se puso en alto la señal sec_N en el flanco ascendente de reloj anterior.

El estado S3 se corresponde con rojo_1 en '1' y verde_2 en '1', por lo tanto, como uno de los semáforos está en verde, N cambia de valor a 30 segundos nuevamente.

En el momento en que se hace la transición al estado S3, en la simulación tienen que haber pasado lo equivalente a 36 segundos, es decir, 3,6 μs.

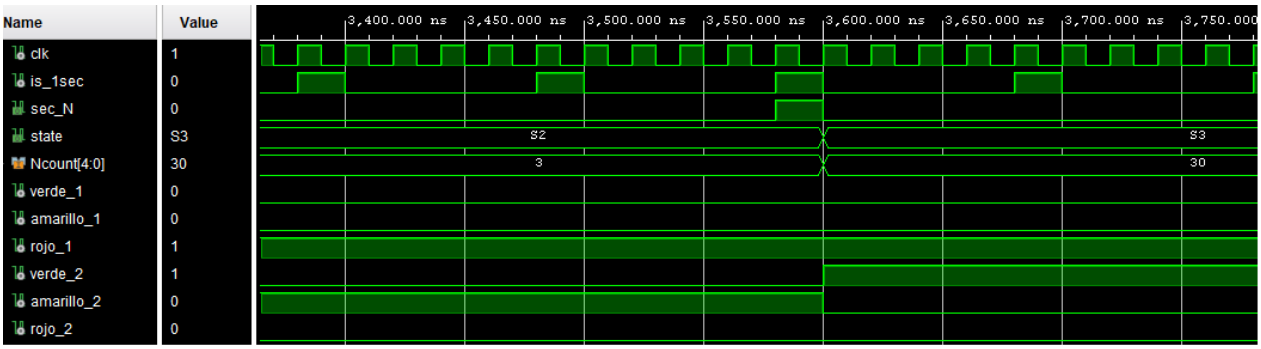

Figura 9: Transición de S2 a S3

Como en el estado S3 N nuevamente pasó a ser 30 segundos, la transición al próximo estado, S4, se da en 6,6 μs. Como se explicó anteriormente, dicha transición la provoca la señal `sec_N`, la cual se pone en 1 un flanco de reloj antes. Al pasar al estado S4, las luces del semáforo se corresponden con rojo_1 en '1' y amarillo_1 en '1'. Dado que ninguno de los semáforos esta en verde, nuevamente cambia el valor de N a 3 segundos, en el mismo flanco que transiciona la maquina de estados.

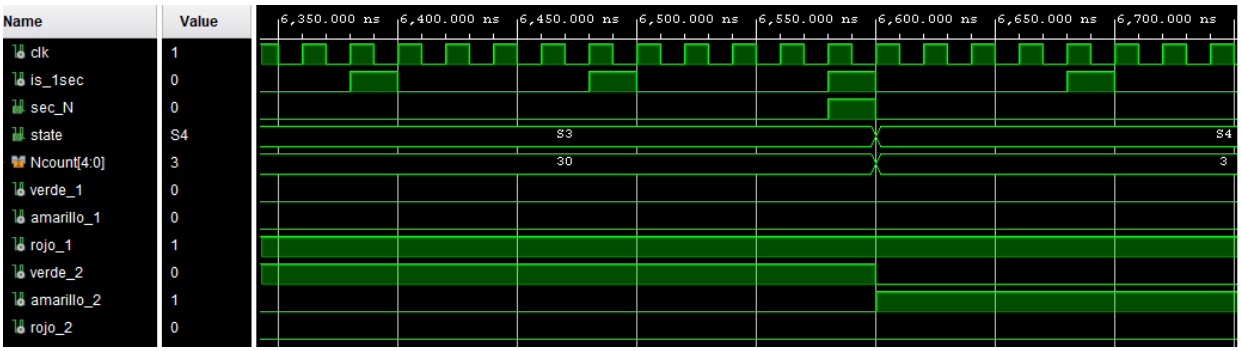

Figura 10: Transición de S3 a S4

La transición al estado S5 se produce entonces a los 6,9 μs, ya que mientras el sistema se encuentra en el estado S4, N vale 3.

Al cambiar a S5, el semaforo 1 pasa a tener amarillo_1 en '1' y el semaforo 2 pasa a estar en rojo, es decir, rojo_2 en '1'. Como ninguno de los dos está en verde, el valor de N continúa siendo 3 segundos.

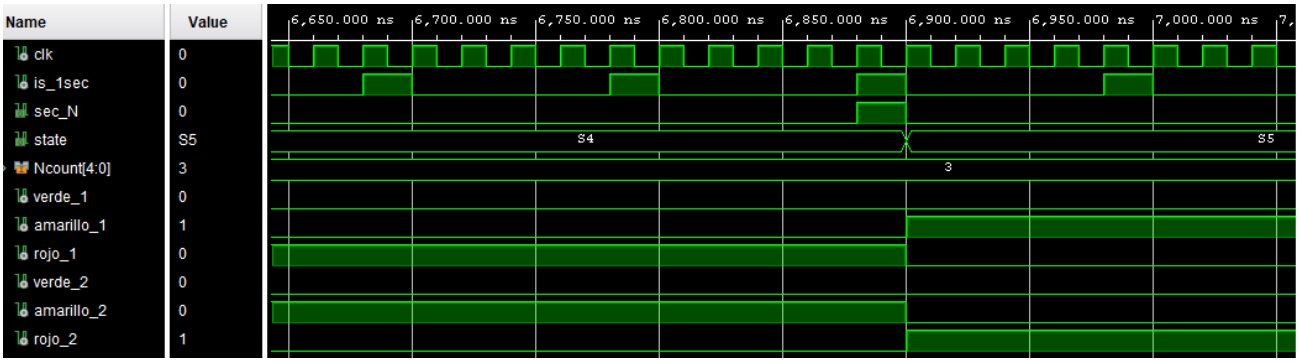

Figura 11: Transición de S4 a S5

Dado que el valor de N se mantuvo en 3 segundos al pasar a S5, la próxima transición se da en el tiempo 7,2 μs. Del estado S5 se retorna nuevamente al estado S0 y el valor de N vuelve a ser 30 segundos ya que el semáforo 1 tiene verde_1 en '1' (y el semáforo de rojo_2 en '1'), completando así todas las transiciones de la maquina de estados.

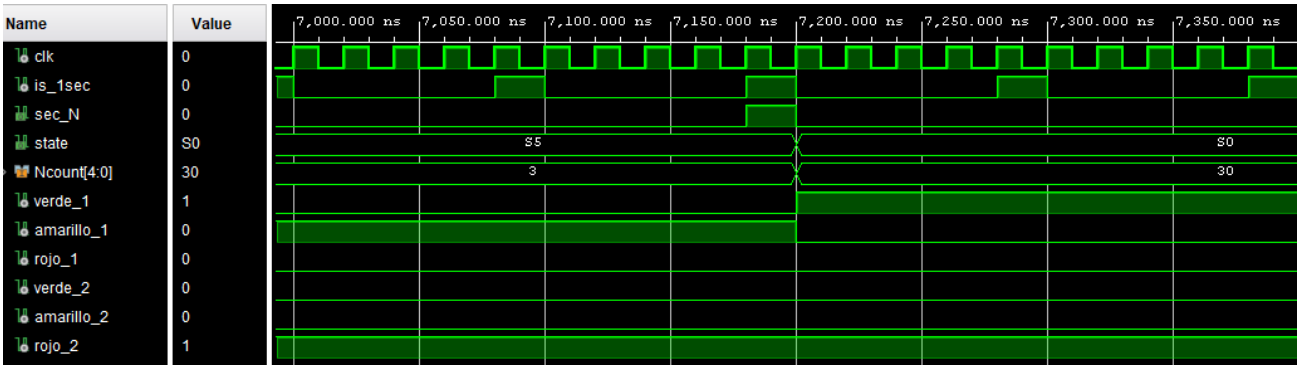

Figura 12: Transición de S5 a S0

5. Síntesis

La síntesis del diseño se hizo sobre la FPGA xc7a15tftg256-1, utilizando Vivado. Esta herramienta de síntesis permite observar el esquemático del diseño tanto a nivel de Look Up Tables (LUTs) y Flip Flops (FFs) como a un nivel de abstracción llamado Register Transfer (RTL), el cual modela el circuito en términos del flujo de las señales entre registros y operaciones lógicas.

En primer lugar se muestra el esquemático RTL del circuito en la figura 13.

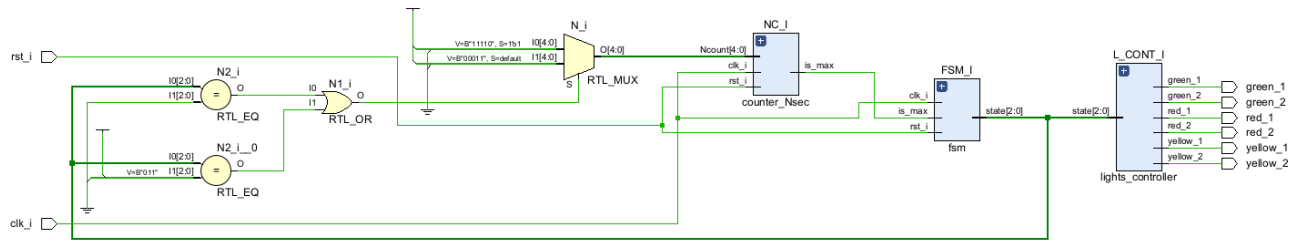

Figura 13: Esquemático RTL del semáforo

Se pueden observar los tres bloques principales del sistema (el contador `counter_Nsec`, la máquina de estados `fsm` y la función de salida `lights_controller`) y el circuito combinacional que define el valor de N. Mediante el multiplexor `RTL_MUX`

se puede seleccionar el valor 30 ó 3 (descrito como un unsigned de 5 bits). El selector del MUX depende del estado.

El contador de N segundos toma el N (máximo valor a contar), el clock y el reset y tiene como salida la señal `is_max`. Luego, la máquina de estado tiene como entrada el clock, el reset y la señal `is_max`, y como salida el estado del sistema, el cual es la entrada de la función de salida que controla las luces de los semáforos. Las salidas del sistema son las 6 luces de los 2 semáforos, tal como se describió en el diseño inicial.

A continuación se muestra el esquemático del contador de N segundos y el contador de 1 segundo.

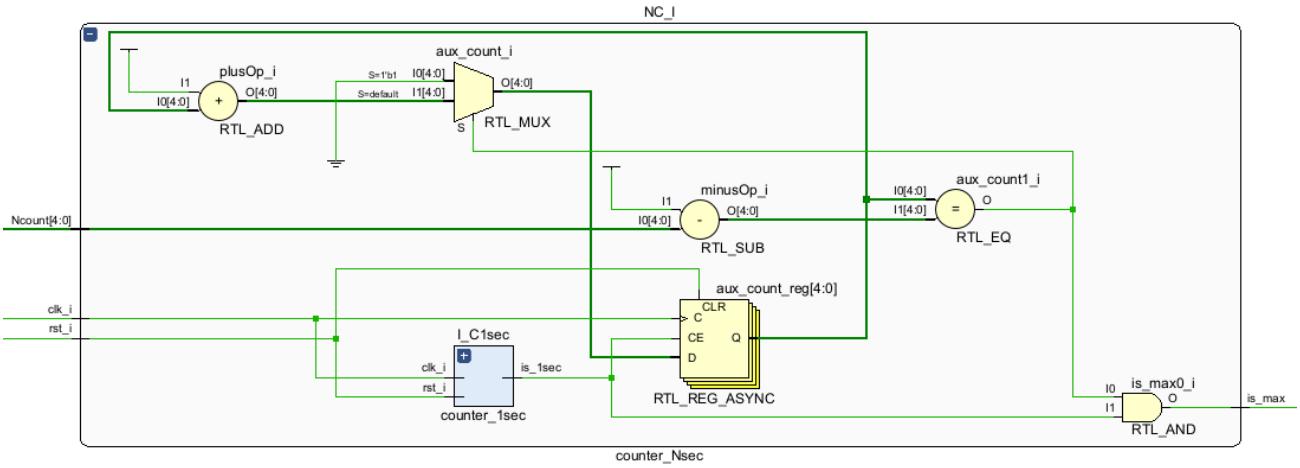

Figura 14: Esquemático RTL del contador de N segundos

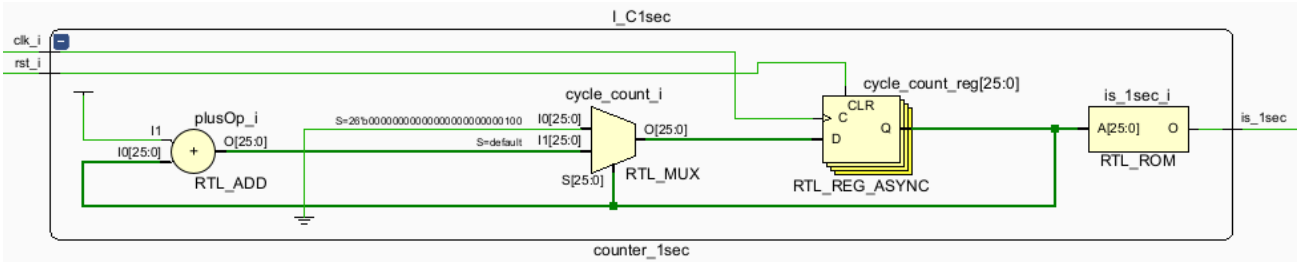

Figura 15: Esquemático RTL del contador de 1 segundo

Ambos diagramas coinciden con los diseños propuestos en la introducción.

Para observar el esquemático de la máquina de estados, se muestra el esquemático de la síntesis en términos de LUTs y FFs.

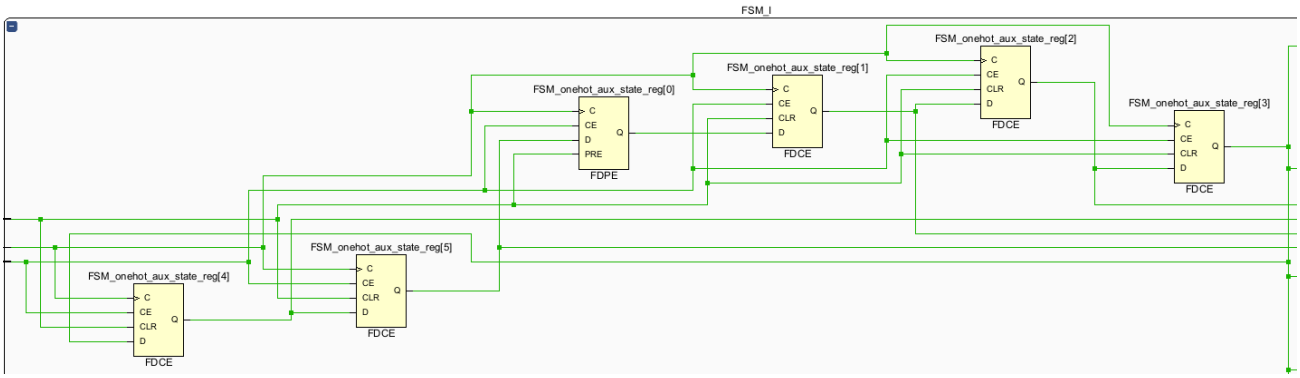

Figura 16: Sector de FFs del esquemático de la máquina de estados

Dado que el esquemático de la máquina de estado es de un tamaño considerable, se optó por recortarlo y mostrar el sector

de los registros para analizarla.

En la figura 16 se puede observar, en primer lugar, que el estado esta códificado en formato one-hot, es decir, se utilizan 6 bits para representar 6 estados y cada estado tiene un único bit en alto.

Además, se observa que los estados están conectados en serie, es decir, la salida (Q) del registro `FSM_onehot_aux_state_reg[0]` está conectada a la entrada (D) del registro `FSM_onehot_aux_state_reg[1]`, cuya salida está conectada a la entrada de `FSM_onehot_aux_state_reg[2]` y así sucesivamente, hasta conectar la salida del registro `FSM_onehot_aux_state_reg[5]` con la entrada de `FSM_onehot_aux_state_reg[0]`.

Entonces, el reg i guarda el valor anterior del reg $i-1$, por lo que cuando el reg $i-1$ pasa de '1' a '0', el reg i pasa a guardar el valor '1'. Esto es equivalente a pasar del estado $i-1$ al estado i.

Por último, se muestra a continuación el reporte de utilización de LUTs y FFs del circuito.

Name	Constraints	Status	LUT	FF	BRAM	URAM	DSP	IO	Part
> ✓ synth_1	constrs_1	synth_design Complete!	29	37	0	0	0	8	xc7a15tftg256-1

Figura 17: Reporte de LUTs y FFs

Se puede ver que se utilizaron 37 FFs en vez de los 34 predichos en un principio. Esto se puede deber a que en el diseño en papel se había tomado una codificación binaria de los estados (necesitando 3 FFs) pero la herramienta de síntesis utilizó una codificación one-hot (necesitando 6 FFs).

6. Conclusión

Durante este trabajo se pudo asimilar los conocimientos de VHDL introducidos en clase y se logró establecer la relación entre la descripción de hardware mediante un lenguaje y su equivalente en un diagrama circuital. Además, se afianzaron los conceptos clave a la hora de describir un cirucito secuencial. A la vez, dado que se llegó a sintetizar el diseño en una FPGA, se pudo obtener un primer acercamiento a las herramientas de síntesis como es Vivado, el cual tiene funcionalidades que facilitan el diseño y el análisis del circuito. Por último, se pudo valorar la importancia de modularizar los componentes del circuito de manera correcta para poder simplificar el diseño del circuito y luego reutilizar dichos componentes a futuro.