INGENIERÍA MECATRÓNICA

DI_CERO

DIEGO CERVANTES RODRÍGUEZ

ELECTRÓNICA DIGITAL: CIRCUITOS LÓGICOS, LENGUAJE VHDL Y VERILOG

XILINX (64-BIT PROJECT NAVIGATOR) & ADEPT

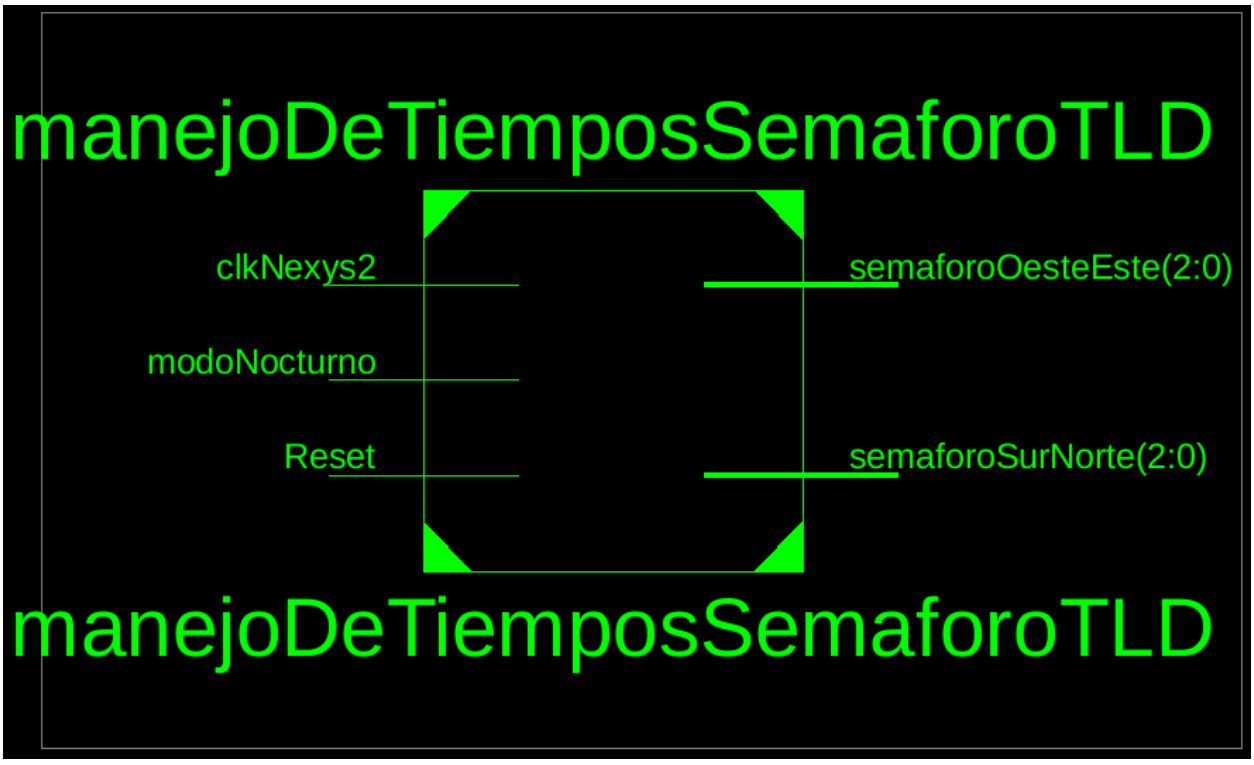
Display de 7 Segmentos: Suma, Resta, Multiplicación y Decodificador BCD

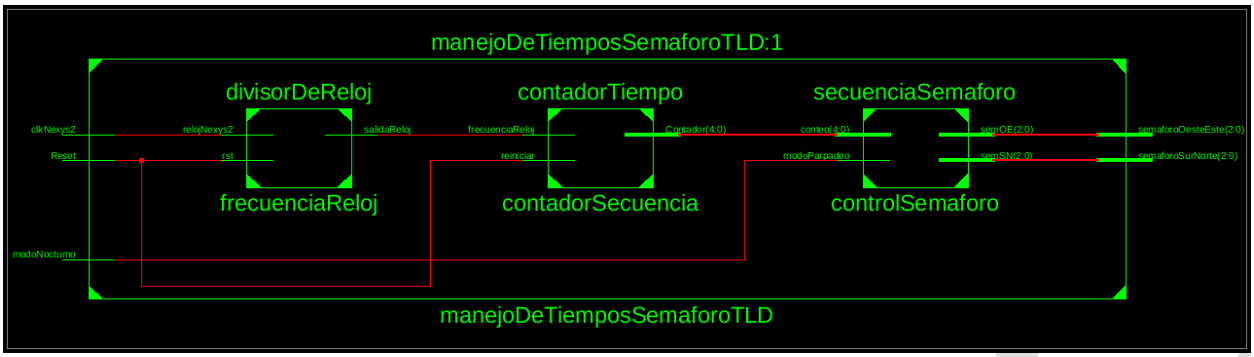
Contenido

TLD: Manejo de Tiempos, 2 Secuencias de Semáforo

El diagrama de bloques que se muestra a continuación pretende ejecutar una secuencia que se ejecute de forma indefinida, eligiendo entre dos tipos de semáforos distintos por medio de un switch para conmutar entre la secuencia normal y de modo nocturno, para ello se agregan los siguientes bloques:

- **DIV:** Un módulo divisor de reloj.
- **CONTADOR DE TIEMPO:** Un módulo contador/selector que dicte el tiempo en el que se ejecuta cada paso de la secuencia.
- **SECUENCIA SEMÁFORO:** En este módulo se indican las acciones a realizar en cada uno de los pasos de la secuencia, para ello debemos tener bien en cuenta de cuantos pasos se conforma cada secuencia y el tiempo de separación que queremos que haya entre cada paso, osea la velocidad con la que se ejecuta la secuencia.





Divisor de Reloj: Módulo DIV

Del divisor de reloj se elegirá una frecuencia de 0.745Hz con un periodo de $\frac{1}{0.745\ Hz} = 1.3422\ segundos$, que se encuentra en la coordenada 26 del vector que divide al reloj y se obtiene por medio de la fórmula porque no se encuentra en la tabla.

q(i)	BASYS 2 y NEXYS 2		NEXYS 3 y NEXYS 4	
	Frecuencia (Hz)	Periodo (s)	Frecuencia (Hz)	Periodo (s)
i	50,000,000.00	0.00000002	100,000,000.00	0.00000001
0	25,000,000.00	0.00000004	50,000,000.00	0.00000002
1	12,500,000.00	0.00000008	25,000,000.00	0.00000004
2	6,250,000.00	0.00000016	12,500,000.00	0.00000008
3	3,125,000.00	0.00000032	6,250,000.00	0.00000016
4	1,562,500.00	0.00000064	3,125,000.00	0.00000032
5	781,250.00	0.00000128	1,562,500.00	0.00000064
6	390,625.00	0.00000256	781,250.00	0.00000128
7	195,312.50	0.00000512	390,625.00	0.00000256
8	97,656.25	0.00001024	195,312.50	0.00000512
9	48,828.13	0.00002048	97,656.25	0.00001024
10	24,414.06	0.00004096	48,828.13	0.00002048
11	12,207.03	0.00008192	24,414.06	0.00004096
12	6,103.52	0.00016384	12,207.03	0.00008192
13	3,051.76	0.00032768	6,103.52	0.00016384
14	1,525.88	0.00065536	3,051.76	0.00032768
15	762.94	0.00131072	1,525.88	0.00065536
16	381.47	0.00262144	762.94	0.00131072
17	190.73	0.00524288	381.47	0.00262144
18	95.37	0.01048576	190.73	0.00524288
19	47.68	0.02097152	95.37	0.01048576
20	23.84	0.04194304	47.68	0.02097152
21	11.92	0.08388608	23.84	0.04194304
22	5.96	0.16777216	11.92	0.08388608
23	2.98	0.33554432	5.96	0.16777216
24	1.49	0.67108864	2.98	0.33554432

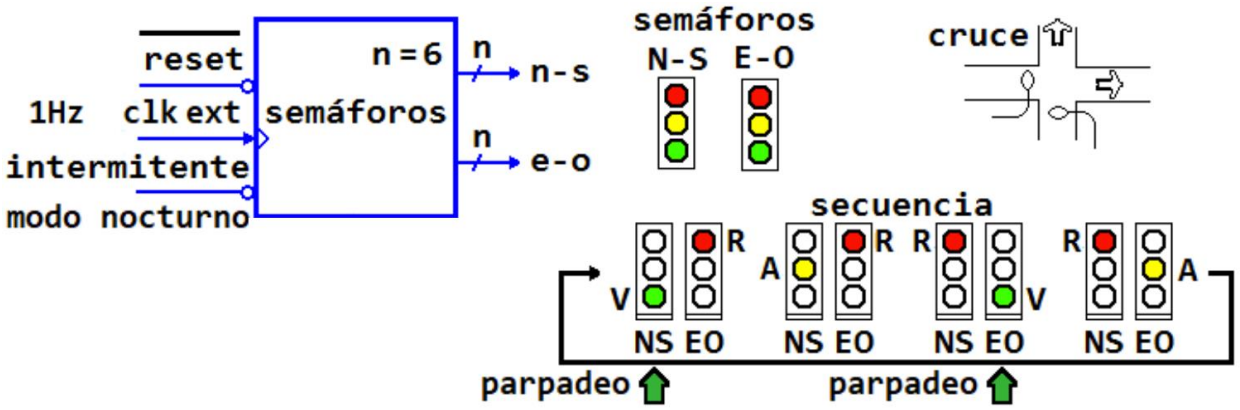
$$fi = \frac{f}{2^{i+1}}$$

Secuencia Semáforo: Modo Normal y Nocturno

En el programa se describen dos secuencias simples de un semáforo para el control de un crucero que tiene dos sentidos, de **Sur a Norte SN** (**abajo hacia arriba**) y de **Oeste a Este OE** (**derecha a izquierda**), cada secuencia se conforma de los pasos descritos a continuación, donde además se tiene que tomar en cuenta que cuando el semáforo SN se encuentre con la luz verde o amarilla encendida, el semáforo OE debe estar en rojo y viceversa:

- **Modo normal (Secuencia 1):** En esta secuencia **10s** se muestra la **luz roja**, **7s** se muestra la **luz verde** **con 3 parpadeos antes del amarillo** y **3s** se muestra **la luz amarilla**, estas secuencias están entrelazadas en los dos semáforos **SN** y **OE**, dando en total una secuencia de 20s.
- **Modo nocturno (Secuencia 2):** En esta secuencia simplemente se pone a **parpadear la luz roja** en el semáforo **SN** y **parpadea la luz amarilla** en el semáforo **OE**.

El tiempo total que abarcan ambas secuencias no es de $10 + 7 + 3 + 3 = 23\ segundos$, aunque así parezca, ya que los tiempos de encendido y apagado de los focos rojo, amarillo y verde están entrelazados en los dos semáforos al mismo tiempo.



Código VHDL:

Divisor de Reloj (DIV):

```
--1.-DIVISOR DE RELOJ:
--Este proceso sirve para dictarle al reloj en que frecuencia quiero que opere.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerías para poder usar el lenguaje VHDL.
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librería declarada para poder hacer operaciones matemáticas sin considerar el signo.

entity divisorDeReloj is
    Port ( relojNexys2 : in  STD_LOGIC;   --Reloj de 50MHz proporcionado por la NEXYS 2 en el puerto B8.
           rst : in  STD_LOGIC;           --Botón de reset.
           salidaReloj : out  STD_LOGIC); --Reloj que quiero con una frecuencia menor a 50MHz.
end divisorDeReloj;

architecture frecuenciaNueva of divisorDeReloj is
signal divisorDeReloj : STD_LOGIC_VECTOR (25 downto 0);
begin
         process(relojNexys2, rst)
         begin
                  if(rst='1') then
                           divisorDeReloj <= "00000000000000000000000000";
                  elsif(rising_edge(relojNexys2)) then
                           divisorDeReloj <= divisorDeReloj + 1;
                  end if;
         end process;
         salidaReloj <= divisorDeReloj(25);
         --La coordenada 25 corresponde a una frecuencia de 0.745Hz, obtenida con la formula.
end frecuenciaNueva;
```

Contador/Selector (CONTADOR DE TIEMPO):

```
--2.-CONTADOR:
--En proceso se realiza el conteo de 0 a 23 del selector que indicara el comportamiento
--de las dos secuencias del semáforo: Modo normal y modo nocturno.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerías que sirven solamente para poder usar el lenguaje VHDL
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librería para poder realizar operaciones matemáticas sin considerar el signo


entity contadorTiempo is
    Port ( frecuenciaReloj : in  STD_LOGIC; --Reloj de 50MHz de la NEXYS 2.
           --Contador de 5 bits, desde 0 hasta el 31 en binario, aquí se abarcan
           --los 20 segundos de la secuencia.
           Contador : out  STD_LOGIC_VECTOR (4 downto 0);
           reiniciar: in STD_LOGIC
                        );
end contadorTiempo;

architecture Behavioral of contadorTiempo is
signal conteoAscendente : STD_LOGIC_VECTOR (4 downto 0) := "00000";
begin
         process(frecuenciaReloj, reiniciar)
         begin
                  if(rising_edge(frecuenciaReloj)) then
                           --Cuando el conteo llegue hasta 20, se reinicia el conteo del selector.
                           if(conteoAscendente >= "10100") then
                                    conteoAscendente <= "00000";
                           --Tambien si se presiona el botón de reset, el conteo se reinicia.
                           elsif(reiniciar = '1') then
                                    conteoAscendente <= "00000";
                           --Si el conteo no ha llegado a los 20 segundos o no se ha reiniciado, cuenta
                           --de 0 a 20 con un tiempo de separación de 1/0.745Hz = 1.3422 segundos.
                           else
                                    conteoAscendente <= conteoAscendente + "00001";
                           end if;
                  end if;
         end process;
         Contador <= conteoAscendente;
end Behavioral;
```

Secuencia (SECUENCIA SEMÁFORO):

```
--3.-SECUENCIA MODO NORMAL Y NOCTURNO:
--En proceso se prenden y apagan los tres leds (rojo, amarillo y verde) en las dos secuencias
--modo normal y nocturno en los dos semáforos SN (abajo hacia arriba) y OE (izquierda a derecha)
--        -Secuencia 1 (modo normal): 10s se muestra la luz roja, 7s se muestra la verde con 3 parpadeos
--         antes del amarillo y 3s se muestra la luz amarilla, estas secuencias están entrelazadas en los
--         dos semáforos SN y OE, dando en total una secuencia de 20s.
--        -Secuencia 2 (modo nocturno): En esta secuencia simplemente se pone a parpadear la luz roja en
--         el semáforo SN y la amarilla en el semáforo OE.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity secuenciaSemaforo is
    Port ( conteo : in  STD_LOGIC_VECTOR (4 downto 0);
           modoParpadeo : in  STD_LOGIC;
           semSN : out  STD_LOGIC_VECTOR (2 downto 0);
           semOE : out  STD_LOGIC_VECTOR (2 downto 0));
end secuenciaSemaforo;

architecture Behavioral of secuenciaSemaforo is
         constant verde: std_logic_vector (2 downto 0):="100";      --Semáforo en verde.
         constant amarillo: std_logic_vector (2 downto 0):="010";   --Semáforo en amarillo.
         constant rojo: std_logic_vector (2 downto 0):="001";       --Semáforo en rojo.
         constant parpadeo: std_logic_vector (2 downto 0):="000";   --Semáforos que estén parpadeando.
begin
         asignacion: process (conteo, modoParpadeo) begin
                  --Si el switch modoParpadeo esta activado, se enciende el modo Nocturno, donde simplemente
                  --parpadean los semáforos SN y OE, en SN parpadea la luz roja y en OE parpadea la luz amarilla.
                  if (modoParpadeo = '1') then
                           case(conteo) is -- Alternativa con case
                                    when "00000" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 0.
                                    when "00001" => semSN<=rojo;           semOE<=amarillo;    --segundo 1.
                                    when "00010" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 2.
                                    when "00011" => semSN<=rojo;           semOE<=amarillo;    --segundo 3.
                                    when "00100" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 4.
                                    when "00101" => semSN<=rojo;           semOE<=amarillo;    --segundo 5.
                                    when "00110" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 6.
                                    when "00111" => semSN<=rojo;           semOE<=amarillo;    --segundo 7.
                                    when "01000" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 8.
                                    when "01001" => semSN<=rojo;           semOE<=amarillo;    --segundo 9.
                                    when "01010" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 10.
                                    when "01011" => semSN<=rojo;           semOE<=amarillo;    --segundo 11.
                                    when "01100" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 12.
                                    when "01101" => semSN<=rojo;           semOE<=amarillo;    --segundo 13.
                                    when "01110" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 14.
                                    when "01111" => semSN<=rojo;           semOE<=amarillo;    --segundo 15.
                                    when "10000" => semSN<=parpadeo;       semOE<=parpadeo;    --segundo 16.
```

```
                                    when "10001" => semSN<=rojo;            semOE<=amarillo;    --segundo 17.
                                    when "10010" => semSN<=parpadeo;        semOE<=parpadeo;    --segundo 18.
                                    when "10011" => semSN<=rojo;            semOE<=amarillo;    --segundo 19.
                                    when others => semSN<=parpadeo;         semOE<=parpadeo;    --otros, reinicio del conteo.
                            end case;
                    --Si el switch modoParpadeo esta activado, se enciende el modo Nocturno, donde 10s se muestra
                    --la luz roja, 7s se muestra la verde con 3 parpadeos antes del amarillo y 3s se muestra la luz
                    --amarilla, estas secuencias están entrelazadas en los dos semáforos SN y OE, dando en total
                    --una secuencia de 20s.
                    else
                            case(conteo) is -- Alternativa con case
                                    when "00000" => semSN<=rojo;            semOE<=verde;   --segundo 0: Empieza el rojo en NS.
                                    when "00001" => semSN<=rojo;            semOE<=parpadeo;--segundo 1.
                                    when "00010" => semSN<=rojo;            semOE<=verde;   --segundo 2.
                                    when "00011" => semSN<=rojo;            semOE<=parpadeo; --segundo 3.
                                    when "00100" => semSN<=rojo;            semOE<=verde;    --segundo 4.
                                    when "00101" => semSN<=rojo;            semOE<=parpadeo; --segundo 5.
                                    when "00110" => semSN<=rojo;            semOE<=verde;    --segundo 6.
                                    when "00111" => semSN<=rojo;            semOE<=amarillo; --segundo 7.
                                    when "01000" => semSN<=rojo;            semOE<=amarillo; --segundo 8.
                                    when "01001" => semSN<=rojo;            semOE<=amarillo; --segundo 9.
                                    when "01010" => semSN<=verde;           semOE<=rojo;     --segundo 10: Empieza verde en NS.
                                    when "01011" => semSN<=parpadeo;        semOE<=rojo;     --segundo 11: Empieza parpadeo NS.
                                    when "01100" => semSN<=verde;           semOE<=rojo;     --segundo 12.
                                    when "01101" => semSN<=parpadeo;        semOE<=rojo;     --segundo 13.
                                    when "01110" => semSN<=verde;           semOE<=rojo;     --segundo 14.
                                    when "01111" => semSN<=parpadeo;        semOE<=rojo;     --segundo 15.
                                    when "10000" => semSN<=verde;           semOE<=rojo;     --segundo 16.
                                    when "10001" => semSN<=amarillo;        semOE<=rojo;     --segundo 17: Empieza amarillo NS.
                                    when "10010" => semSN<=amarillo;        semOE<=rojo;     --segundo 18.
                                    when "10011" => semSN<=amarillo;        semOE<=rojo;     --segundo 19.
                                    when others => semSN<=parpadeo;         semOE<=rojo;     --otros, reinicio del conteo.
                            end case;
                    end if;
            end process asignacion;
end Behavioral;
```

Módulo TLD:

```
--4.-TLD (Top Level Design) Semáforo: En el programa se describen dos secuencias simples de un semáforo para el control
--de un crucero que tiene dos sentidos, de Sur a Norte SN (abajo hacia arriba) y de Oeste a Este OE (derecha a
--izquierda), en cada secuencia se tienen los siguientes tiempos, donde además se tiene que tomar en cuenta que
--cuando el semáforo SN se encuentre con la luz verde o amarilla encendida, el semáforo OE debe estar en rojo y
--viceversa:
--                  - Secuencia 1 (modo normal): 10s se muestra la luz roja, 7s se muestra la verde con 3 parpadeos antes del
--                    amarillo y 3s se muestra la luz amarilla, estas secuencias están entrelazadas en los dos semáforos SN y OE,
--                    dando en total una secuencia de 20s.
--                  - Secuencia 2 (modo nocturno): En esta secuencia simplemente se pone a parpadear la luz roja en el semáforo SN
--                    y la amarilla en el semáforo OE.
--El tiempo total que abarcan ambas secuencias no es de 10 + 7 + 3 + 3 = 23 s aunque así parezca, ya que los tiempos de
--encendido y apagado de los focos rojo, amarillo y verde están entrelazados en los dos semáforos.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

--Entidad: Las entradas y salidas en este módulo son las que van a entrar y salir del diagrama de bloques global.
entity manejoDeTiemposSemaforoTLD is
    Port ( Reset : in  STD_LOGIC;
                        clkNexys2 : in  STD_LOGIC;
                        modoNocturno : in  STD_LOGIC;
           semaforoSurNorte : out  STD_LOGIC_VECTOR (2 downto 0);
           semaforoOesteEste : out  STD_LOGIC_VECTOR (2 downto 0));
end manejoDeTiemposSemaforoTLD;

architecture Behavioral of manejoDeTiemposSemaforoTLD is
--Conexiones internas del diseño TLD.
signal reloj : STD_LOGIC;
signal controlSecuencia : STD_LOGIC_VECTOR (4 downto 0);

begin
          --INSTANCIAS:
          --Debo darle un nombre a cada instancia que cree, indicar el nombre de la entidad del módulo que quiero instanciar,
          --usar la palabra reservada port map(); y dentro de su paréntesis asignarle a todas las entradas y salidas del
          --módulo instanciado una entrada, salida o signal de este módulo separadas por comas una de la otra, esto hará que
          --lo que entre o salga del otro módulo, entre, salga o se guarde en este.
          --La sintaxis que debemos usar es la siguiente:

          --NombreInstancia            :                  entity work.Entidad_Del_Modulo_Que_Queremos_Instanciar     port map(
          --                  Entrada_Del_Modulo_Instanciado => Entrada_En_Este_Modulo,
          --                  Salida_Del_Modulo_Instanciado  => Salida_En_Este_Modulo,

          --                  Entrada_Del_Modulo_Instanciado => Salida_En_Este_Modulo,
          --                  Salida_Del_Modulo_Instanciado  => Entrada_En_Este_Modulo,

          --                  Entrada_Del_Modulo_Instanciado => Signal_En_Este_Modulo,
          --                  Salida_Del_Modulo_Instanciado  => Signal_En_Este_Modulo
          --);

          --INSTANCIA DEL MODULO divisorDeReloj para obtener la frecuencia en la que quiero que se cree el contador/selector.
          frecuenciaReloj : entity work.divisorDeReloj port map(
                            relojNexys2 => clkNexys2,
                            --La entrada relojNexys2 del divisorDeReloj se asigna a la entrada clkNexys2 de este módulo.
                            rst => Reset,
                            --La entrada rst del divisorDeReloj se asigna a la entrada Reset de este módulo.
```

```
                        salidaReloj => reloj
                        --La salida salidaReloj del divisorDeReloj se asigna a la signal reloj de este módulo.
                );
        --INSTANCIA DEL MODULO contadorTiempo para obtener el conteo que controla la secuencia del semáforo.
        contadorSecuencia : entity work.contadorTiempo port map(
                        frecuenciaReloj => reloj,
                        --A la entrada frecuenciaReloj del contadorSelector se le asigna el valor de la signal reloj.
                        Contador => controlSecuencia,
                        --La salida contador del contadorTiempo se asigna a la signal controlSecuencia de este módulo.
                        reiniciar => Reset
                        --La entrada rst del divisorDeReloj se asigna a la entrada Reset de este módulo.
                );
        --INSTANCIA DEL MODULO secuenciaSemaforo para recibir el contador e iniciar la secuencia del semaforo.
        controlSemaforo : entity work.secuenciaSemaforo port map(
                        conteo => controlSecuencia,
                        --A la entrada conteo de secuenciaSemaforo se le asigna el valor de la signal controlSecuencia.
                        modoParpadeo => modoNocturno,
                        --La entrada modoParpadeo de secuenciaSemaforo se asigna a la entrada modoNocturno de este módulo.
                        semSN => semaforoSurNorte,
                        --La salida semNS de secuenciaSemaforo se asigna a la salida semaforoNorteSur de este módulo.
                        semOE => semaforoOesteEste
                        --La salida semEO de secuenciaSemaforo se asigna a la salida semaforoNorteSur de este módulo.
                );
end Behavioral;
```

Código UCF:

```
//ENTRADAS DEL DIAGRAMA TLD:
net "Reset" loc = "R17";                // Reset para el reloj BTN3.
net "clkNexys2" loc = "B8";             // Reloj de 50MHz.
net "modoNocturno" loc = "G18";         // Activación del modo nocturno SW0.

//SALIDAS DEL DIAGRAMA TLD:
//Semaforo Sur Norte.
net "semaforoSurNorte[2]" loc = "R4";   // LD7 semáforo NS luz verde.
net "semaforoSurNorte[1]" loc = "F4";   // LD6 semáforo NS luz amarilla.
net "semaforoSurNorte[0]" loc = "P15"; // LD5 semáforo NS luz roja.
//Semforo Oeste Este.
net "semaforoOesteEste[2]" loc = "K15"; // LD2 semáforo EO luz verde.
net "semaforoOesteEste[1]" loc = "J15"; // LD1 semáforo EO luz amarilla.
net "semaforoOesteEste[0]" loc = "J14"; // LD0 semáforo EO luz roja.
```

Simulación Secuencia Semáforo: