Acad. Year

2024-25

3º GITT

Implementación de una Cerradura Digital en VHDL

Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

Grupo 1

Silvia Mena González

Naroa García Bermejo

Sistemas Electrónicos Digitales

3º GITT

Proyecto realizado por el **Grupo 1** para la asignatura *Sistemas Electrónicos Digitales* cursada en 3º de GITT por la US. Realizado por:

Mena González, Silvia - silmengon@alum.us.es

García Bermejo, Naroa - nargarber@alum.us.es

Índice general

Introducción

En este trabajo se abordará la implementación de una cerradura digital en *VHDL*. Para ello, se diseñará en este mismo lenguaje los distintos bloques que mencionarán más adelante.

La metodología que se ha seguido para la realización de los ejercicios, es la comprobación del funcionamiento del mismo, seguido de un reset, y la ejecución de los estímulos que se piden. De este modo, en las gráficas se puede ver una simulación previa, un reset y los estímulos del ejercicio.

Para un desarrollo más profesional del proyecto se ha implementado un control de versiones mediante un repositorio en Github (enlace al repositorio). Además se han utilizado indistintamente las herramientas *VHDL Simili 3.0* como *Vivado* para el desarrollo y simulación de los diferentes bloques que componen el proyecto.

Capítulo 1

Ejercicio 1: Comprobación del código (0.5 puntos)

In this apartado se desarrollará el bloque encargado de comprobar si los dgitos introducidos coinciden with los designados como nuestro código, en nuestro caso se ha elegido la combinación (4-3-2-1) siendo 4 el dgito más significativo y 1 el menos significativo. En los siguientes apartados se puede ver tanto el código vhdl que se ha escrito para generar este bloque como el fichero de estímulos que se ha usado para comprobar su correcto funcionamiento.

1.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity codigo is
    Port (
            digit0 : in STD_LOGIC_VECTOR (3 downto 0);
            digit1 : in STD_LOGIC_VECTOR (3 downto 0);
            digit2 : in STD_LOGIC_VECTOR (3 downto 0);
            digit3 : in STD_LOGIC_VECTOR (3 downto 0);
            resultado : out STD_LOGIC);
```

```
end codigo;

architecture codigo_arq of codigo is
begin
--Resultado s lo  ser 1 si se cumple que los d gitos sean lo que
    queramos
resultado <= '1' when (digit3 = "0100" and -- digit3 = 4
                       digit2 = "0011" and -- digit2 = 3
                       digit1 = "0010" and -- digit1 = 2
                       digit0 = "0001") -- digit0 = 1
                       else '0';
end codigo_arq;
```

1.2. Código de la arquitectura del fichero de estímulos

El siguiente fichero de estímulos ha sido utilizado para generar las formas de ondas del siguiente apartado.

```
---
   ********************************************************************************

-- LIBRERIAS
--
   ********************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************************
```

```
-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_codigo IS
END    test_codigo;

--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_codigo_arq OF test_codigo IS
    --Declaracin  de componentes
    COMPONENT codigo
        Port (
           digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           digit3 : in STD_LOGIC_VECTOR (3 downto 0);
           resultado : out STD_LOGIC);
        END COMPONENT;

    SIGNAL RESULTADO : std_logic;
    SIGNAL DIGIT0,DIGIT1,DIGIT2,DIGIT3 : std_logic_vector(3
       downto 0);
```

```
constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ns

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: codigo PORT MAP(
                          digit0    => DIGIT0,
                          digit1    => DIGIT1,
                          digit2    => DIGIT2,
                          digit3    => DIGIT3,
                          resultado => RESULTADO
                        );

    --
       ================================================================

    -- Proceso para el resto de entradas; ENABLE_test
    --
       ================================================================

    tb: PROCESS
    BEGIN

        digit3 <= "0011";
```

```
        digit2 <= "0111";
        digit1 <= "1000";
        digit0 <= "1001";
        wait for 2*ciclo;

        digit3 <= "0100";
        digit2 <= "0011";
        digit1 <= "0010";
        digit0 <= "0001";
        wait for 2*ciclo;

        digit3 <= "0011";
        digit2 <= "0111";
        digit1 <= "1001";
        digit0 <= "1000";
        wait for 2*ciclo;

        digit3 <= "0000";
        digit2 <= "0000";
        digit1 <= "0000";
        digit0 <= "0000";
        wait;                    -- Espera indefinida con
           ENABLE_test='1'
    end process tb;

END test_codigo_arq;
```

1.3. Gráfica

Al aplicar el fichero de estímulos a la simulación obtenemos el siguiente diagrama de ondas, se observa que resultado sólo se pone a '1' cuando la combinación introducida coincide con la

escogida.

Se pide: rellenar el código del fichero de diseño y el de estímulos (codigo_tb.vhd) para que realice una simulación en la que las entradas tomen los valores 3789, 4321 (correcto) y 3798.

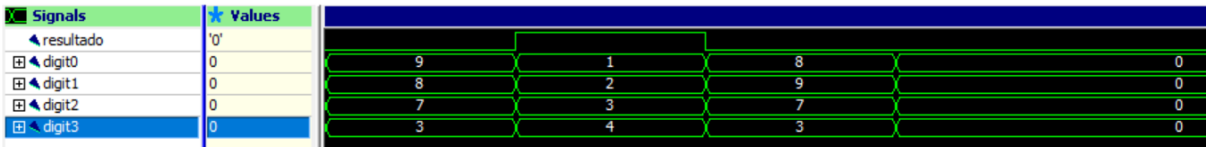

Figura 1.1: Simulación del comportamiento del bloque codigo.vhd.

Resultado de la simulación extra que se pide en el enunciado del proyecto

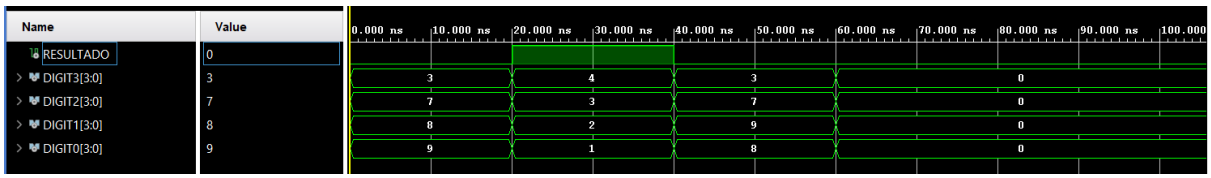

Figura 1.2: Simulación del comportamiento del bloque codigo.vhd.

Capítulo 2

Ejercicio 1: Comprobación del código (0.5 puntos)

In this apartado se desarrollará el bloque encargado de comprobar si los dgitos introducidos coinciden with los designados como nuestro código, en nuestro caso se ha elegido la combinación (4-3-2-1) siendo 4 el dgito más significativo y 1 el menos significativo. En los siguientes apartados se puede ver tanto el código vhdl que se ha escrito para generar este bloque como el fichero de estímulos que se ha usado para comprobar su correcto funcionamiento.

2.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity codigo is
    Port (
            digit0 : in STD_LOGIC_VECTOR (3 downto 0);
            digit1 : in STD_LOGIC_VECTOR (3 downto 0);
            digit2 : in STD_LOGIC_VECTOR (3 downto 0);
            digit3 : in STD_LOGIC_VECTOR (3 downto 0);
            resultado : out STD_LOGIC);
```

```
end codigo;

architecture codigo_arq of codigo is
begin
--Resultado s lo  ser 1 si se cumple que los d gitos sean lo que
    queramos
resultado <= '1' when (digit3 = "0100" and -- digit3 = 4
                       digit2 = "0011" and -- digit2 = 3
                       digit1 = "0010" and -- digit1 = 2
                       digit0 = "0001") -- digit0 = 1
                       else '0';
end codigo_arq;
```

2.2. Código de la arquitectura del fichero de estímulos

El siguiente fichero de estímulos ha sido utilizado para generar las formas de ondas del siguiente apartado.

```
---
   ********************************************************************************

-- LIBRERIAS
--
   ********************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************************
```

```
-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_codigo IS
END    test_codigo;

--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_codigo_arq OF test_codigo IS
    --Declaracin  de componentes
    COMPONENT codigo
        Port (
           digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           digit3 : in STD_LOGIC_VECTOR (3 downto 0);
           resultado : out STD_LOGIC);
        END COMPONENT;

    SIGNAL RESULTADO : std_logic;
    SIGNAL DIGIT0,DIGIT1,DIGIT2,DIGIT3 : std_logic_vector(3
       downto 0);
```

```
constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ns

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: codigo PORT MAP(
                          digit0    => DIGIT0,
                          digit1    => DIGIT1,
                          digit2    => DIGIT2,
                          digit3    => DIGIT3,
                          resultado => RESULTADO
                       );

    --
       ================================================================

    -- Proceso para el resto de entradas; ENABLE_test
    --
       ================================================================

    tb: PROCESS
    BEGIN

        digit3 <= "0011";
```

```
        digit2 <= "0111";
        digit1 <= "1000";
        digit0 <= "1001";
        wait for 2*ciclo;

        digit3 <= "0100";
        digit2 <= "0011";
        digit1 <= "0010";
        digit0 <= "0001";
        wait for 2*ciclo;

        digit3 <= "0011";
        digit2 <= "0111";
        digit1 <= "1001";
        digit0 <= "1000";
        wait for 2*ciclo;

        digit3 <= "0000";
        digit2 <= "0000";
        digit1 <= "0000";
        digit0 <= "0000";
        wait;                    -- Espera indefinida con
           ENABLE_test='1'
    end process tb;

END test_codigo_arq;
```

2.3. Gráfica

Al aplicar el fichero de estímulos a la simulación obtenemos el siguiente diagrama de ondas, se observa que resultado sólo se pone a '1' cuando la combinación introducida coincide con la

escogida.

Se pide: rellenar el código del fichero de diseño y el de estímulos (codigo_tb.vhd) para que realice una simulación en la que las entradas tomen los valores 3789, 4321 (correcto) y 3798.

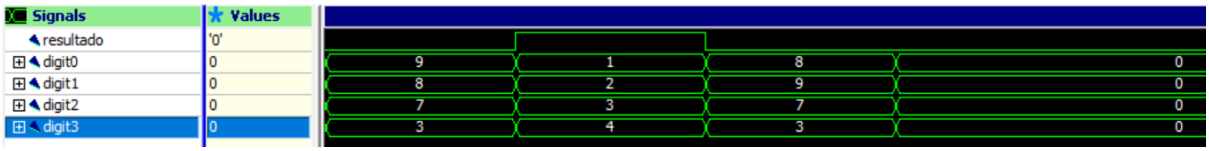

Figura 2.1: Simulación del comportamiento del bloque codigo.vhd.

Resultado de la simulación extra que se pide en el enunciado del proyecto

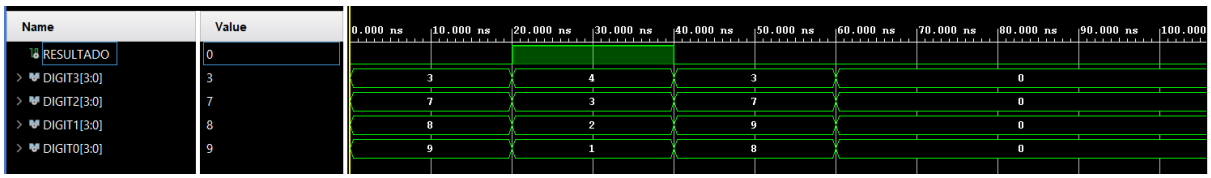

Figura 2.2: Simulación del comportamiento del bloque codigo.vhd.

Capítulo 3

Ejercicio 2: Señales para los displays de 7 segmentos (0.5 puntos)

Este bloque se encarga de multiplexar las señales y guiarlas a los displays de 7 segmentos. Estos se encargan de representar el número en los 7 segmentos del display.

3.1. Código de la arquitectura del diseño

```
--
   ----------------------------------------------------------------------------------


-- Company:
-- Engineer:
--
-- Create Date: 15.04.2025 17:47:39
-- Design Name:
-- Module Name: siete_seg - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
```

```
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--
   ----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity siete_seg is
    Port ( segments : out STD_LOGIC_VECTOR (6 downto 0);
           digit_rota : in STD_LOGIC_VECTOR (3 downto 0);
           digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           digit3 : in STD_LOGIC_VECTOR (3 downto 0));
```

```
end siete_seg;

architecture Behavioral of siete_seg is
    signal bcd : std_logic_vector(3 downto 0);
begin
--Se seleciona un por defecto > 9 para que en el caso de que se
   d  no se muestre en el display
bcd <= digit0 when digit_rota = "0001" else
       digit1 when digit_rota = "0010" else
       digit2 when digit_rota = "0100" else
       digit3 when digit_rota = "1000" else
       "1111";

--La secuencia en nuestro 7 segmentos es ABCDEFGDp (mirar foto de
    la documentacin), recuerda que es activa a nivel bajo
with bcd select
segments <= "1000000" when "0000",  -- 0
            "1111001" when "0001",  -- 1
            "0100100" when "0010",  -- 2
            "0110000" when "0011",  -- 3
            "0011010" when "0100",  -- 4
            "0010010" when "0101",  -- 5
            "0000010" when "0110",  -- 6
            "1111000" when "0111",  -- 7
            "0000000" when "1000",  -- 8
            "0010000" when "1001",  -- 9
            "1111111" when others;  -- all segments OFF (default)

end Behavioral;
```

3.2. Código de la arquitectura del fichero de estímulos

```
---
   ********************************************************************

-- LIBRERIAS
--
   ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_siete_seg IS
END    test_siete_seg;

--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_siete_seg_arq OF test_siete_seg IS
```

```
    --Declaracin  de componentes
    COMPONENT siete_seg
        Port (
           digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           digit3 : in STD_LOGIC_VECTOR (3 downto 0);
           digit_rota : in STD_LOGIC_VECTOR (3 downto 0);
           segments : out STD_LOGIC_VECTOR (6 downto 0));
        END COMPONENT;

    SIGNAL SEGMENTS : std_logic_vector(6 downto 0);
    SIGNAL DIGIT0,DIGIT1,DIGIT2,DIGIT3, DIGIT_ROTA :
       std_logic_vector(3 downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ns

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: siete_seg PORT MAP(
                          digit0     => DIGIT0,
                          digit1     => DIGIT1,
```

```
                    digit2    => DIGIT2,
                    digit3    => DIGIT3,
                    segments => SEGMENTS,
                    digit_rota => DIGIT_ROTA
                );

--
   ================================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ================================================================================

tb: PROCESS
BEGIN
    -- Simulacin  inicial para comprobar que est  bien el
       c digo
    digit0 <= "0001";
    digit1 <= "0111";
    digit2 <= "0011";
    digit3 <= "0110";

    digit_rota <= "0001";
    wait for 2*ciclo;

    digit_rota <= "0010";
    wait for 2*ciclo;

    digit_rota <= "0100";
    wait for 2*ciclo;
```

```
        digit_rota <= "1000";
        wait for 2*ciclo;

        digit0 <= "0101";
        digit1 <= "0001";
        digit2 <= "0110";
        digit3 <= "1001";

        digit_rota <= "1000";
        wait for 2*ciclo;

        digit_rota <= "0100";
        wait for 2*ciclo;

        digit_rota <= "0010";
        wait for 2*ciclo;

        digit_rota <= "0001";
        wait;                   -- Espera indefinida con
           ENABLE_test='1'
    end process tb;

END test_siete_seg_arq;
```

3.3. Gráfica

Se pide simular el caso en el que digit0=5, digit1=1, digit2=6, digit3=9 y la entrada digit_rota cambiará al revés. Comenzará por 1000 -¿0100 -¿0010 -¿0001.

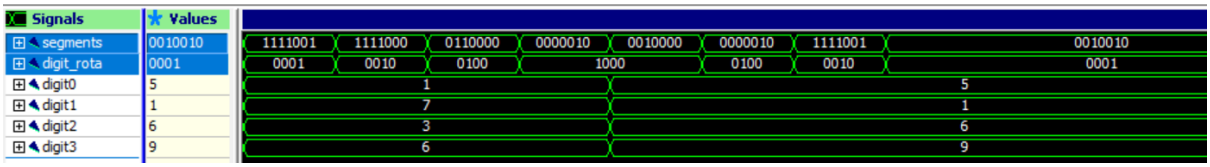

Figura 3.1: Simulación 7 segmentos

Capítulo 4

Ejercicio 2: Señales para los displays de 7 segmentos (0.5 puntos)

Este bloque se encarga de multiplexar las señales y guiarlas a los displays de 7 segmentos. Estos se encargan de representar el número en los 7 segmentos del display.

4.1. Código de la arquitectura del diseño

```
--
   ----------------------------------------------------------------------------------


-- Company:
-- Engineer:
--
-- Create Date: 15.04.2025 17:47:39
-- Design Name:
-- Module Name: siete_seg - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
```

```
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--
   ----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity siete_seg is
    Port ( segments : out STD_LOGIC_VECTOR (6 downto 0);
           digit_rota : in STD_LOGIC_VECTOR (3 downto 0);
           digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           digit3 : in STD_LOGIC_VECTOR (3 downto 0));
```

```
end siete_seg;

architecture Behavioral of siete_seg is
    signal bcd : std_logic_vector(3 downto 0);
begin
--Se seleciona un por defecto > 9 para que en el caso de que se
   d  no se muestre en el display
bcd <= digit0 when digit_rota = "0001" else
       digit1 when digit_rota = "0010" else
       digit2 when digit_rota = "0100" else
       digit3 when digit_rota = "1000" else
       "1111";

--La secuencia en nuestro 7 segmentos es ABCDEFGDp (mirar foto de
    la documentacin), recuerda que es activa a nivel bajo
with bcd select
segments <= "1000000" when "0000",  -- 0
            "1111001" when "0001",  -- 1
            "0100100" when "0010",  -- 2
            "0110000" when "0011",  -- 3
            "0011010" when "0100",  -- 4
            "0010010" when "0101",  -- 5
            "0000010" when "0110",  -- 6
            "1111000" when "0111",  -- 7
            "0000000" when "1000",  -- 8
            "0010000" when "1001",  -- 9
            "1111111" when others;  -- all segments OFF (default)

end Behavioral;
```

4.2. Código de la arquitectura del fichero de estímulos

```
---
   ********************************************************************

-- LIBRERIAS
--
   ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_siete_seg IS
END    test_siete_seg;

--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_siete_seg_arq OF test_siete_seg IS
```

```
    --Declaracin  de componentes
    COMPONENT siete_seg
        Port (
           digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           digit3 : in STD_LOGIC_VECTOR (3 downto 0);
           digit_rota : in STD_LOGIC_VECTOR (3 downto 0);
           segments : out STD_LOGIC_VECTOR (6 downto 0));
        END COMPONENT;

    SIGNAL SEGMENTS : std_logic_vector(6 downto 0);
    SIGNAL DIGIT0,DIGIT1,DIGIT2,DIGIT3, DIGIT_ROTA :
       std_logic_vector(3 downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ns

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: siete_seg PORT MAP(
                          digit0     => DIGIT0,
                          digit1     => DIGIT1,
```

```
                    digit2    => DIGIT2,
                    digit3    => DIGIT3,
                    segments => SEGMENTS,
                    digit_rota => DIGIT_ROTA
                );

--
   ================================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ================================================================================

tb: PROCESS
BEGIN
    -- Simulacin  inicial para comprobar que est  bien el
       c digo
    digit0 <= "0001";
    digit1 <= "0111";
    digit2 <= "0011";
    digit3 <= "0110";

    digit_rota <= "0001";
    wait for 2*ciclo;

    digit_rota <= "0010";
    wait for 2*ciclo;

    digit_rota <= "0100";
    wait for 2*ciclo;
```

```
        digit_rota <= "1000";
        wait for 2*ciclo;

        digit0 <= "0101";
        digit1 <= "0001";
        digit2 <= "0110";
        digit3 <= "1001";

        digit_rota <= "1000";
        wait for 2*ciclo;

        digit_rota <= "0100";
        wait for 2*ciclo;

        digit_rota <= "0010";
        wait for 2*ciclo;

        digit_rota <= "0001";
        wait;                   -- Espera indefinida con
           ENABLE_test='1'
    end process tb;

END test_siete_seg_arq;
```

4.3. Gráfica

Se pide simular el caso en el que digit0=5, digit1=1, digit2=6, digit3=9 y la entrada digit_rota cambiará al revés. Comenzará por 1000 -¿0100 -¿0010 -¿0001.

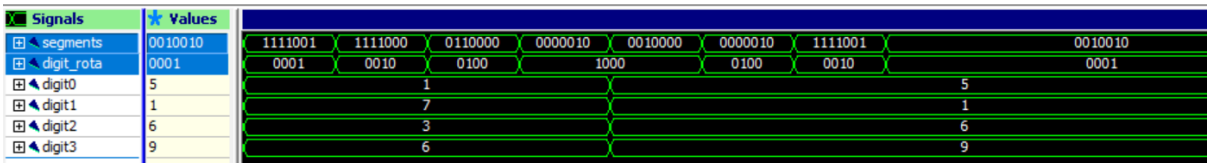

Figura 4.1: Simulación 7 segmentos

Capítulo 5

Ejercicio 3: Control de rebotes en los pulsadores (1 punto)

Los módulos `rebotes.vhd` consiguen filtrar los rebotes en la señal de elementos mecánicos como pulsadores.

5.1. Codigo arquitectura de diseño

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY rebotes IS
  GENERIC(
    N  :  INTEGER := 21); -- tamano contador (20 bits dan 10.5ms
       con reloj de 100Mhz)
  PORT(
    clk : IN  STD_LOGIC;  --input clock
    i   : IN  STD_LOGIC;  --input signal to be rebotesd
    o   : OUT STD_LOGIC); --output signal
END rebotes;
```

```
ARCHITECTURE rebotes_arq OF rebotes IS
  SIGNAL biestables: STD_LOGIC_VECTOR(1 DOWNTO 0) := "00";
                  --input flip flops
  SIGNAL reinicia  : STD_LOGIC := '0';
                                      --sync reset to zero
  SIGNAL cuenta    : STD_LOGIC_VECTOR(N-1 DOWNTO 0) := (OTHERS =>
      '0'); --counter output
  SIGNAL reset     : STD_LOGIC := '0';
BEGIN

  reinicia <= biestables(0) xor biestables(1);   --determina
     cuando se reinicia o cuenta el contador


  PROCESS(clk)
  BEGIN
    IF(clk'EVENT and clk = '1') THEN -- si hay rising_edge
      biestables(0) <= i;
      biestables(1) <= biestables(0);
      reset <= reinicia;
      IF(reinicia = '1') THEN
        CUENTA <= (others =>'0');
      ELSE
        CUENTA <= CUENTA + 1;
        IF(CUENTA(N-1) = '1') THEN
            o <= biestables(1);
        END IF;
        END IF;
      END IF;
```

```
  END PROCESS;
END rebotes_arq;
```

5.1.1. Código arquitectura del fichero de estímulos

```
--
   ********************************************************************************

-- LIBRERIAS
--
   ********************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************************

ENTITY test_rebotes IS
END    test_rebotes;

--
   ********************************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
```

```
   ****************************************************************

ARCHITECTURE test_rebotes_arq OF test_rebotes IS
    --Declaracin  de componentes
    COMPONENT rebotes
                GENERIC(
                      N : INTEGER := 3  -- Tamao  del contador
                          (ajustable segun la frecuencia del
                         reloj)
                );
                PORT(
                      clk      : IN  STD_LOGIC;  -- Reloj del
                         sistema
                      i        : IN  STD_LOGIC;  -- Entrada del
                         botn  con rebote
                      o        : OUT STD_LOGIC   -- Salida
                         debounced
                );
    END COMPONENT;

    SIGNAL CLK,I,O : std_logic;
    constant ciclo: time :=  10 ns;  -- hay que dejar un espacio
       entre 10 y ns
BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////
```

```
i_rebotes: rebotes
              GENERIC MAP (N => 2)
              PORT    MAP(
                             clk     => CLK,
                             i       => I,
                             o       => O
                          );

--
   ============================================================================

-- Proceso de la entrada de reloj. Se ejecuta indefinidamente
    ya que no tiene "WAIT;"
--
   ============================================================================

PROCESS
BEGIN
    CLK<='0';     wait for ciclo/2;
    CLK<='1';     wait for ciclo/2;
END PROCESS;


--
   ============================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ============================================================================
```

```
    tb: PROCESS
    BEGIN
        I <= '0';
        wait for 5.75*ciclo;
        i <= '1';
        wait for 0.5*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 1*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 10*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 2*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 1*ciclo;
        i <= '0';



         wait;                    -- Espera indefinida
    end process tb;

END test_rebotes_arq;
```

5.1.2. Gráfica

Se pide simular un flanco de bajada de la señal de entrada con varios rebotes. En las curvas anteriores hay un par de rebotes antes de estabilizarse a nivel alto.

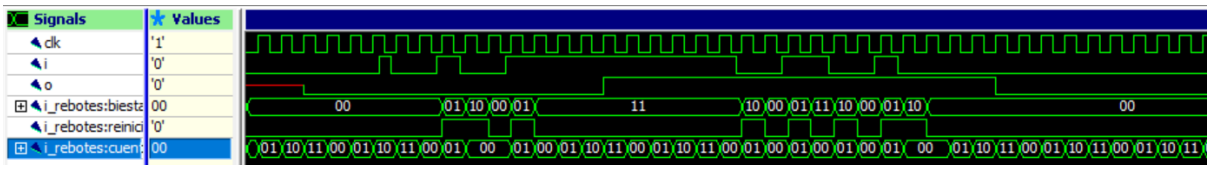

Figura 5.1: Simulación Rebotes

Capítulo 6

Ejercicio 3: Control de rebotes en los pulsadores (1 punto)

Los módulos `rebotes.vhd` consiguen filtrar los rebotes en la señal de elementos mecánicos como pulsadores.

6.1. Codigo arquitectura de diseño

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY rebotes IS
  GENERIC(
    N  :  INTEGER := 21); -- tamano contador (20 bits dan 10.5ms
       con reloj de 100Mhz)
  PORT(
    clk : IN  STD_LOGIC;  --input clock
    i   : IN  STD_LOGIC;  --input signal to be rebotesd
    o   : OUT STD_LOGIC); --output signal
END rebotes;
```

```
ARCHITECTURE rebotes_arq OF rebotes IS
  SIGNAL biestables: STD_LOGIC_VECTOR(1 DOWNTO 0) := "00";
                  --input flip flops
  SIGNAL reinicia  : STD_LOGIC := '0';
                                      --sync reset to zero
  SIGNAL cuenta    : STD_LOGIC_VECTOR(N-1 DOWNTO 0) := (OTHERS =>
      '0'); --counter output
  SIGNAL reset     : STD_LOGIC := '0';
BEGIN

  reinicia <= biestables(0) xor biestables(1);   --determina
     cuando se reinicia o cuenta el contador


  PROCESS(clk)
  BEGIN
    IF(clk'EVENT and clk = '1') THEN -- si hay rising_edge
      biestables(0) <= i;
      biestables(1) <= biestables(0);
      reset <= reinicia;
      IF(reinicia = '1') THEN
        CUENTA <= (others =>'0');
      ELSE
        CUENTA <= CUENTA + 1;
        IF(CUENTA(N-1) = '1') THEN
            o <= biestables(1);
        END IF;
        END IF;
      END IF;
```

```
  END PROCESS;
END rebotes_arq;
```

6.1.1. Código arquitectura del fichero de estímulos

```
--
   ************************************************************************

-- LIBRERIAS
--
   ************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--
   ************************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ************************************************************************

ENTITY test_rebotes IS
END    test_rebotes;

--
   ************************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
```

```
   ****************************************************************

ARCHITECTURE test_rebotes_arq OF test_rebotes IS
    --Declaracin  de componentes
    COMPONENT rebotes
                GENERIC(
                      N : INTEGER := 3  -- Tamao  del contador
                          (ajustable segun la frecuencia del
                         reloj)
                );
                PORT(
                      clk      : IN  STD_LOGIC;  -- Reloj del
                         sistema
                      i        : IN  STD_LOGIC;  -- Entrada del
                         botn  con rebote
                      o        : OUT STD_LOGIC   -- Salida
                         debounced
                );
    END COMPONENT;

    SIGNAL CLK,I,O : std_logic;
    constant ciclo: time :=  10 ns;  -- hay que dejar un espacio
       entre 10 y ns
BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////
```

```
i_rebotes: rebotes
              GENERIC MAP (N => 2)
              PORT    MAP(
                             clk     => CLK,
                             i       => I,
                             o       => O
                          );

--
   ============================================================================

-- Proceso de la entrada de reloj. Se ejecuta indefinidamente
    ya que no tiene "WAIT;"
--
   ============================================================================

PROCESS
BEGIN
    CLK<='0';     wait for ciclo/2;
    CLK<='1';     wait for ciclo/2;
END PROCESS;


--
   ============================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ============================================================================
```

```
    tb: PROCESS
    BEGIN
        I <= '0';
        wait for 5.75*ciclo;
        i <= '1';
        wait for 0.5*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 1*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 10*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 2*ciclo;
        i <= '0';
        wait for 2*ciclo;
        i <= '1';
        wait for 1*ciclo;
        i <= '0';



         wait;                    -- Espera indefinida
    end process tb;

END test_rebotes_arq;
```

6.1.2. Gráfica

Se pide simular un flanco de bajada de la señal de entrada con varios rebotes. En las curvas anteriores hay un par de rebotes antes de estabilizarse a nivel alto.

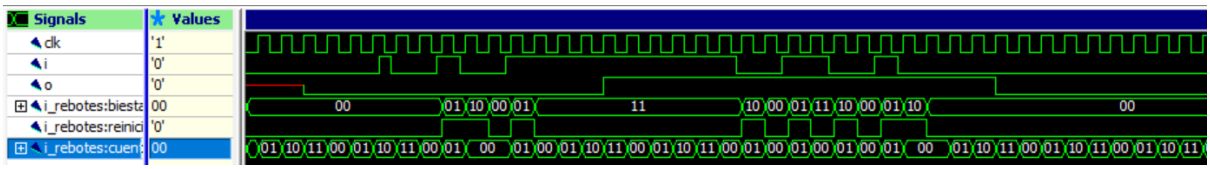

Figura 6.1: Simulación Rebotes

Capítulo 7

Ejercicio 4: Contador para los dígitos

7.0.1. Código de la arquitectura del diseño

```
--
   ----------------------------------------------------------------------------------


-- Company:
-- Engineer:
--
-- Create Date: 15.04.2025 19:32:21
-- Design Name:
-- Module Name: contador - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```
-- Additional Comments:
--
--
   ----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity contador is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           enable : in STD_LOGIC;
           up : in STD_LOGIC;
           down : in STD_LOGIC;
           cuenta : out STD_LOGIC_VECTOR (3 downto 0));
end contador;

architecture Behavioral of contador is
    signal count: unsigned(3 downto 0);
begin
```

```
cuenta <= std_logic_vector(count);

process(clk, enable, up, down, reset)
begin
--Se ha hecho suponiendo un reset asncrono, en caso de ser
    sncrono el reset debe comprobarse despus de ver si estamos
   en un flanco de subida. Preguntar Rafa
 if (reset = '1') then
    count <= "0000";
 elsif (rising_edge(clk) AND enable = '1') then
    if (up = '1') then
        count <= count + 1;
    elsif (down = '1') then
        count <= count - 1;
    end if;
end if;

end process;

end Behavioral;
```

7.0.2. Código de la arquitectura del fichero de estímulos (0.75 puntos)

```
---
   ****************************************************************************

-- LIBRERIAS
--
   ****************************************************************************

LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_contador IS
END    test_contador;

--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_contador_arq OF test_contador IS
    --Declaracin  de componentes
    COMPONENT contador
                Port (
                   clk      : in  STD_LOGIC;
                   reset    : in  STD_LOGIC;
                   enable   : in  STD_LOGIC;
                   up       : in  STD_LOGIC;
                   down     : in  STD_LOGIC;
                            cuenta  : out STD_LOGIC_VECTOR (3
                               downto 0));
```

```
    END COMPONENT;

    SIGNAL CLK,ENABLE,UP,DOWN,RESET : std_logic;
    SIGNAL CUENTA : std_logic_vector(3 downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    10 y ns

BEGIN
    --
       ////////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////////

    U1: contador PORT    MAP(
                               clk     => CLK,
                               reset   => RESET,
                               enable  => ENABLE,
                               up      => UP,
                               down    => DOWN,
                               cuenta => CUENTA
                            );

    --
       ====================================================================

    -- Proceso de la entrada de reloj. Se ejecuta indefinidamente
```

```
    ya que no tiene "WAIT;"
--
   ============================================================================

PROCESS
BEGIN
    CLK <='0';     wait for ciclo/2;
    CLK <='1';     wait for ciclo/2;
END PROCESS;

PROCESS
BEGIN
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait for 10*ciclo;
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait;
END PROCESS;


--
   ============================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ============================================================================

tb: PROCESS
BEGIN
    UP         <= '0';
    DOWN       <= '0';
    ENABLE     <= '0';
```

```
wait for 1.1*ciclo/2;     -- No tocar

-- Simulacin  de comparacin  con la dada en la memoria
for i in 0 to 1 loop
    UP <= '1';
    wait for 1.5*ciclo;
    ENABLE <= '1';
    wait for 1.1*ciclo/2;
    ENABLE <= '0';
    wait for 1.5*ciclo;
    UP <= '0';
    wait for 1.5*ciclo;
end loop;


wait until falling_edge(reset);
wait for ciclo;
--Simulacin  de lo que se nos pide aportar
for i in 0 to 2 loop
    UP <= '1';
    wait for 1.5*ciclo;
    ENABLE <= '1';
    wait for 1.1*ciclo/2;
    ENABLE <= '0';
    wait for 1.5*ciclo;
    UP <= '0';
    wait for 1.5*ciclo;
end loop;

DOWN <= '1';
wait for 1.5*ciclo;
```

```
        ENABLE <= '1';
        wait for 1.1*ciclo/2;
        ENABLE <= '0';
        wait for 1.5*ciclo;
        DOWN <= '0';
        wait for 1.5*ciclo;

        for i in 0 to 1 loop
            UP <= '1';
            wait for 1.5*ciclo;
            ENABLE <= '1';
            wait for 1.1*ciclo/2;
            ENABLE <= '0';
            wait for 1.5*ciclo;
            UP <= '0';
            wait for 1.5*ciclo;
        end loop;

        wait;                    -- Espera indefinida
    end process tb;

END test_contador_arq;
```

7.0.3. Gráfica

Se pide simular 3 pulsos de la entrada 'up', 1 pulso de la entrada 'down' y dos nuevos pulsos de 'up'.

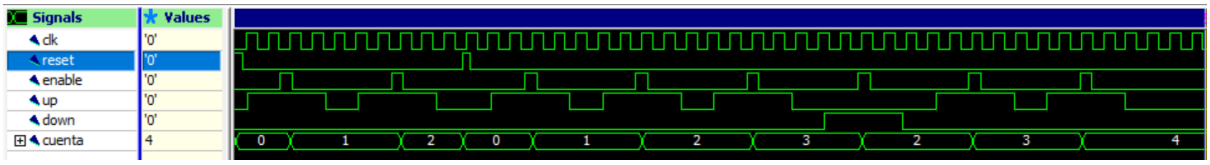

Figura 7.1: Simulación bloque Contador

Capítulo 8

Ejercicio 4: Contador para los dígitos

8.0.1. Código de la arquitectura del diseño

```
--
   ----------------------------------------------------------------------------------

-- Company:
-- Engineer:
--
-- Create Date: 15.04.2025 19:32:21
-- Design Name:
-- Module Name: contador - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
```

```
-- Additional Comments:
--
--
   ----------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity contador is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           enable : in STD_LOGIC;
           up : in STD_LOGIC;
           down : in STD_LOGIC;
           cuenta : out STD_LOGIC_VECTOR (3 downto 0));
end contador;

architecture Behavioral of contador is
    signal count: unsigned(3 downto 0);
begin
```

```
cuenta <= std_logic_vector(count);

process(clk, enable, up, down, reset)
begin
--Se ha hecho suponiendo un reset asncrono, en caso de ser
    sncrono el reset debe comprobarse despus de ver si estamos
   en un flanco de subida. Preguntar Rafa
 if (reset = '1') then
    count <= "0000";
 elsif (rising_edge(clk) AND enable = '1') then
    if (up = '1') then
        count <= count + 1;
    elsif (down = '1') then
        count <= count - 1;
    end if;
end if;

end process;

end Behavioral;
```

8.0.2. Código de la arquitectura del fichero de estímulos (0.75 puntos)

```
---
   ****************************************************************************

-- LIBRERIAS
--
   ****************************************************************************

LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;

--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_contador IS
END    test_contador;

--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_contador_arq OF test_contador IS
    --Declaracin  de componentes
    COMPONENT contador
                Port (
                   clk      : in  STD_LOGIC;
                   reset    : in  STD_LOGIC;
                   enable   : in  STD_LOGIC;
                   up       : in  STD_LOGIC;
                   down     : in  STD_LOGIC;
                            cuenta  : out STD_LOGIC_VECTOR (3
                               downto 0));
```

```
    END COMPONENT;

    SIGNAL CLK,ENABLE,UP,DOWN,RESET : std_logic;
    SIGNAL CUENTA : std_logic_vector(3 downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    10 y ns

BEGIN
    --
       ////////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////////

    U1: contador PORT    MAP(
                               clk     => CLK,
                               reset   => RESET,
                               enable  => ENABLE,
                               up      => UP,
                               down    => DOWN,
                               cuenta => CUENTA
                            );

    --
       ====================================================================

    -- Proceso de la entrada de reloj. Se ejecuta indefinidamente
```

```
    ya que no tiene "WAIT;"
--
   ============================================================================

PROCESS
BEGIN
    CLK <='0';     wait for ciclo/2;
    CLK <='1';     wait for ciclo/2;
END PROCESS;

PROCESS
BEGIN
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait for 10*ciclo;
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait;
END PROCESS;


--
   ============================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ============================================================================

tb: PROCESS
BEGIN
    UP         <= '0';
    DOWN       <= '0';
    ENABLE     <= '0';
```

```
wait for 1.1*ciclo/2;     -- No tocar

-- Simulacin  de comparacin  con la dada en la memoria
for i in 0 to 1 loop
    UP <= '1';
    wait for 1.5*ciclo;
    ENABLE <= '1';
    wait for 1.1*ciclo/2;
    ENABLE <= '0';
    wait for 1.5*ciclo;
    UP <= '0';
    wait for 1.5*ciclo;
end loop;


wait until falling_edge(reset);
wait for ciclo;
--Simulacin  de lo que se nos pide aportar
for i in 0 to 2 loop
    UP <= '1';
    wait for 1.5*ciclo;
    ENABLE <= '1';
    wait for 1.1*ciclo/2;
    ENABLE <= '0';
    wait for 1.5*ciclo;
    UP <= '0';
    wait for 1.5*ciclo;
end loop;

DOWN <= '1';
wait for 1.5*ciclo;
```

```
        ENABLE <= '1';
        wait for 1.1*ciclo/2;
        ENABLE <= '0';
        wait for 1.5*ciclo;
        DOWN <= '0';
        wait for 1.5*ciclo;

        for i in 0 to 1 loop
            UP <= '1';
            wait for 1.5*ciclo;
            ENABLE <= '1';
            wait for 1.1*ciclo/2;
            ENABLE <= '0';
            wait for 1.5*ciclo;
            UP <= '0';
            wait for 1.5*ciclo;
        end loop;

        wait;                    -- Espera indefinida
    end process tb;

END test_contador_arq;
```

8.0.3. Gráfica

Se pide simular 3 pulsos de la entrada 'up', 1 pulso de la entrada 'down' y dos nuevos pulsos de 'up'.

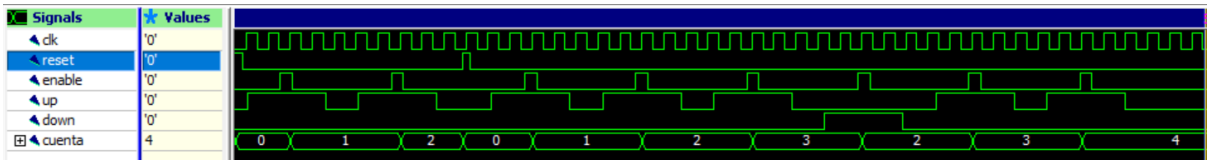

Figura 8.1: Simulación bloque Contador

Capítulo 9

Ejercicio 5: Iluminación de los displays mediante rotación

9.1. Iluminación Básica (0.75 puntos)

9.1.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity rotacion is
        GENERIC (
            N20: INTEGER := 20    -- contador que fija el tiempo
               de iluminacin  de cada display
        );
    Port ( clk,reset        : in  STD_LOGIC;
           digit_rota       : out STD_LOGIC_VECTOR (3 downto 0)
         );
end rotacion;
```

```
architecture rotacion_arq of rotacion is
    signal contador_rotacion : unsigned(N20-1 downto 0) := (
       others => '1');  -- unsigned(19 downto 0)
    SIGNAL rotacion_selecc : UNSIGNED(1 DOWNTO 0);
    signal fin_rotacion : std_logic;

begin

   -- proceso para darle valores (asignar) la seal  '
      contador_rotacion'
   --Tambien se gestiona aqu el reset asncrono
   process(clk, reset)
   begin
        --Para evitar latches
        contador_rotacion <= contador_rotacion;

        if(reset = '1') then
                contador_rotacion <= (others => '0');
                fin_rotacion <= '0';

        elsif (rising_edge(clk)) then

                contador_rotacion <= contador_rotacion + 1;

                --Esto hay que ver si se puede poner de una forma
                    ms mona realmente estamos viendo si en el
                   siguiente ya desborda
                if((contador_rotacion + 1) = 0) then
                        fin_rotacion <= '1';
                else
                        fin_rotacion <= '0';
```

```
                end if;
        end if;
    end process;


    -- seal  auxiliar con los dos bits ms significativos del
       contador
    rotacion_selecc(1 DOWNTO 0) <=  contador_rotacion( (N20-1)
       downto (N20-2) );

    -- Proceso para asignar la salida 'digit_rota[]'
    PROCESS (rotacion_selecc)
    BEGIN

        case rotacion_selecc is
            when "00" =>  digit_rota <= "0001";
            when "01" =>  digit_rota <= "0010";
            when "10" =>  digit_rota <= "0100";
            when "11" =>  digit_rota <= "1000";
            when others =>  digit_rota <= "1111";
        end case;

     END PROCESS;

end rotacion_arq;
```

9.1.2. Código de la arquitectura del fichero de estímulos

```
---
    ********************************************************************

-- LIBRERIAS
```

```
--
   ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;



--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_rotacion IS
END    test_rotacion;



--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los estmulos)
--
   ********************************************************************

ARCHITECTURE test_rotacion_arq OF test_rotacion IS
    --Declaracin  de componentes
```

```
    COMPONENT rotacion
        GENERIC (
              N20: INTEGER := 20   -- Paramatrizable para
                 cualquier frecuencia de reloj
        );
        Port (
           clk,reset        : in  STD_LOGIC;
           digit_rota       : out STD_LOGIC_VECTOR (3 downto 0)
        );
    END COMPONENT;

    SIGNAL CLK,RESET: std_logic;
    SIGNAL DIGIT_ROTA : std_logic_vector(3 downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: rotacion
           GENERIC MAP (N20 => 3)
           PORT MAP(
               clk           => CLK,
```

```
            reset          => RESET,
            digit_rota     => DIGIT_ROTA
        );
--
   ============================================================================

-- Proceso de la entrada del reset inicial
--
   ============================================================================


PROCESS
BEGIN
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait for 10*ciclo;
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait;
END PROCESS;

 --
    ============================================================================

-- Proceso de la simulacin  de un reloj de entrada
--
   ============================================================================

PROCESS
BEGIN
    CLK <='0';     wait for ciclo/2;
    CLK <='1';     wait for ciclo/2;
END PROCESS;
```

```
END test_rotacion_arq;
```

9.1.3. Gráfica

Se pide añadir una señal interna llamada fin_rotacion que se pondrá a 1 -durante un ciclo de reloj- cuando contador_rotacion desborda y pasa a 0. Habrá que indicarle al simulador que muestre esta señal interna.

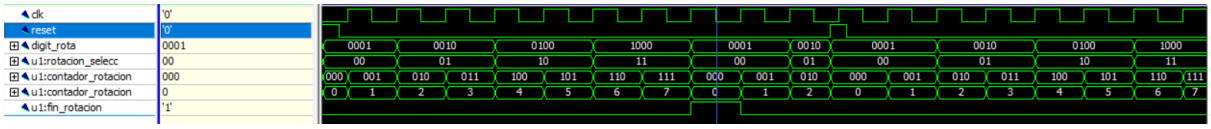

Figura 9.1: Simulación Iluminación Básica

9.2. Iluminación Temporizada (1.5 puntos)

9.2.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity rotacion is
        GENERIC (
            N20: INTEGER := 20;    -- 20 bits para el contador de
               rotacin
            N10: INTEGER := 10;    -- 10 bits para el contador de
               parpadeo
            NP: INTEGER := 2    -- Constante para el  c lculo  de
               parpadeos
        );
    Port ( clk,reset        : in  STD_LOGIC;
```

```
            contador_activo: in std_logic_vector(3 downto 0);
            ini_parpadeo: in std_logic;
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
          );
end rotacion;

architecture rotacion_arq of rotacion is
    signal contador_rotacion : unsigned(N20-1 downto 0) := (
       others => '1');  -- unsigned(19 downto 0)
    signal contador_parpadeo : unsigned(N10-1 downto 0) := (
       others => '1');  -- unsigned(9 downto 0)
    SIGNAL rotacion_selecc : UNSIGNED(1 DOWNTO 0);
    signal fin_rotacion : std_logic;

begin

   -- proceso para darle valores (asignar) la seal  '
      contador_rotacion'
   --Tambien se gestiona aqu el reset asncrono
   process(clk, reset, ini_parpadeo, fin_rotacion)
   begin
        --Para evitar latches
        contador_rotacion <= contador_rotacion;
        contador_parpadeo <= contador_parpadeo;

        if(reset = '1') then
                contador_rotacion <= (others => '0');
                contador_parpadeo <= (others => '0');
                fin_rotacion <= '0';
                fin_parpadeo <= '0';
```

```
    elsif (rising_edge(clk)) then

            contador_rotacion <= contador_rotacion + 1;

            -- Vemos si al sumarle uno ms desborda
            if((contador_rotacion + 1) = 0) then
                    fin_rotacion <= '1';
            else
                    fin_rotacion <= '0';
            end if;

            if(contador_parpadeo(N10-1) = '0') then
                    fin_parpadeo <= '0';

                    if(fin_rotacion = '1'  AND ini_parpadeo =
                        '1') then
                            contador_parpadeo <=
                               contador_parpadeo + 1;
                    end if;
            else
                    fin_parpadeo <= '1';
            end if;
    end if;
end process;


-- seal  auxiliar con los dos bits ms significativos del
   contador
rotacion_selecc(1 DOWNTO 0) <=  contador_rotacion( (N20-1)
   downto (N20-2) );
```

```
    -- Proceso para asignar la salida 'digit_rota[]'
    PROCESS (rotacion_selecc)
    BEGIN

         case rotacion_selecc is
             when "00" =>  digit_rota <= "0001";
             when "01" =>  digit_rota <= "0010";
             when "10" =>  digit_rota <= "0100";
             when "11" =>  digit_rota <= "1000";
             when others =>  digit_rota <= "1111";
         end case;

     END PROCESS;

end rotacion_arq;
```

9.2.2. Código de la arquitectura del fichero de estímulos

```
---
    ********************************************************************

-- LIBRERIAS
--
    ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;
```

```
--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_rotacion IS
END    test_rotacion;



--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_rotacion_arq OF test_rotacion IS
    --Declaracin  de componentes
    COMPONENT rotacion
        GENERIC (
              N20: INTEGER := 20;   -- 20 bits para el contador
                 de rotacin
              N10: INTEGER := 10;   -- 10 bits para el contador
                 de parpadeo
              NP: INTEGER := 2   -- Constante para el  clculo  de
                 parpadeos
        );
```

```
        Port (
            clk,reset       : in  STD_LOGIC;
            contador_activo: in std_logic_vector(3 downto 0);
            ini_parpadeo: in std_logic;
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
        );
    END COMPONENT;

    SIGNAL CLK, RESET, INI_PARPADEO, FIN_PARPADEO: std_logic;
    SIGNAL DIGIT_ROTA, CONTADOR_ACTIVO : std_logic_vector(3
       downto 0);

constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: rotacion
            GENERIC MAP (N20 => 2, N10 => 4)
            PORT MAP(
                clk           => CLK,
                reset         => RESET,
                ini_parpadeo => INI_PARPADEO,
```

```
            fin_parpadeo => FIN_PARPADEO,
            contador_activo => CONTADOR_ACTIVO,
            digit_rota    => DIGIT_ROTA
        );
--
   ============================================================================

-- Proceso de la entrada del reset inicial
--
   ============================================================================


PROCESS
BEGIN
    reset <='1';    wait for ciclo/3;
    reset <='0';    wait for 10*ciclo;
    reset <='1';    wait for ciclo/3;
    reset <='0';    wait;
END PROCESS;

 --
    ============================================================================

-- Proceso de la simulacin  de un reloj de entrada
--
   ============================================================================

PROCESS
BEGIN
    CLK <='0';    wait for ciclo/2;
    CLK <='1';    wait for ciclo/2;
```

```
    END PROCESS;

    --
       ============================================================================

    -- Proceso para el resto de entradas; ENABLE_test
    --
       ============================================================================

    tb: PROCESS
    BEGIN
     contador_activo <= (others => '0');
     ini_parpadeo    <= '0';
     wait for 1.1*ciclo/2;

     ini_parpadeo <= '1';
     contador_activo <= "1000";
     --Descomentar esta seccin  para comprobar que al
        desactivarse la sean  ini_parpadeo para de incrementarse
         el contador
     -- de parpadeo

     --wait for 20*ciclo;
     --ini_parpadeo <= '0';

     --wait for 20*ciclo;
     --ini_parpadeo <= '1';
     wait;

    end process tb;
END test_rotacion_arq;
```

9.2.3. Gráfica

Se pide realizar la simulación que se muestra pero solo hasta que la señal fin_parpadeo se pone a 1. El valor de contador_activo=2 y la constante NP=2 son irrelevantes, por ahora. Las señales contador_rotacion, rotacion_selecc y digit_rota se mostrarán en formato binario.

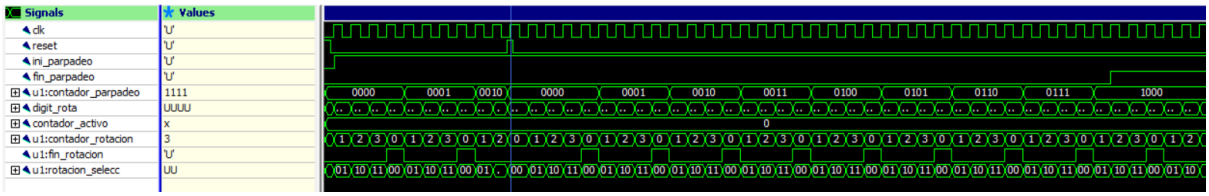

Figura 9.2: Simulación Iluminación Completa

9.3. Iluminación completa (1 punto)

9.3.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity rotacion is
        GENERIC (
            N20: INTEGER := 20;    -- 20 bits para el contador de
               rotacin
            N10: INTEGER := 10;    -- 10 bits para el contador de
               parpadeo
            NP: INTEGER := 2    -- Constante para el  c lculo  de
               parpadeos
        );
    Port ( clk,reset        : in  STD_LOGIC;
           contador_activo: in std_logic_vector(3 downto 0);
           ini_parpadeo: in std_logic;
```

```
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
          );
end rotacion;

architecture rotacion_arq of rotacion is
    signal contador_rotacion : unsigned(N20-1 downto 0) := (
       others => '1');  -- unsigned(19 downto 0)
    signal contador_parpadeo : unsigned(N10-1 downto 0) := (
       others => '1');  -- unsigned(9 downto 0)
    SIGNAL rotacion_selecc : UNSIGNED(1 DOWNTO 0);
    signal fin_rotacion : std_logic;

begin

   -- proceso para darle valores (asignar) la seal  '
      contador_rotacion'
   --Tambien se gestiona aqu el reset asncrono
   process(clk, reset, ini_parpadeo, fin_rotacion)
   begin
        --Para evitar latches
        contador_rotacion <= contador_rotacion;
        contador_parpadeo <= contador_parpadeo;

        if(reset = '1') then
                contador_rotacion <= (others => '0');
                contador_parpadeo <= (others => '0');
                fin_rotacion <= '0';
                fin_parpadeo <= '0';

        elsif (rising_edge(clk)) then
```

```
                contador_rotacion <= contador_rotacion + 1;

                -- Vemos si al sumarle uno ms desborda
                if((contador_rotacion + 1) = 0) then
                        fin_rotacion <= '1';
                else
                        fin_rotacion <= '0';
                end if;

                if(contador_parpadeo(N10-1) = '0') then
                        fin_parpadeo <= '0';

                        if(fin_rotacion = '1'  AND ini_parpadeo =
                            '1') then
                                contador_parpadeo <=
                                   contador_parpadeo + 1;
                        end if;
                else
                        fin_parpadeo <= '1';
                end if;
        end if;
end process;


-- seal  auxiliar con los dos bits ms significativos del
   contador
rotacion_selecc(1 DOWNTO 0) <=  contador_rotacion( (N20-1)
   downto (N20-2) );

-- Proceso para asignar la salida 'digit_rota[]'
```

```
PROCESS (rotacion_selecc)
BEGIN

     case rotacion_selecc is
         when "00" =>
             digit_rota <= "0001";
             if(contador_activo = "0001" AND ini_parpadeo =
                '1' AND contador_parpadeo(contador_parpadeo'
                high-NP) = '0' AND contador_parpadeo(N10-1) =
                '0') then
                     digit_rota <= (others => '0');
             end if;
         when "01" =>
             digit_rota <= "0010";
             if(contador_activo = "0010" AND ini_parpadeo =
                '1' AND contador_parpadeo(contador_parpadeo'
                high-NP) = '0' AND contador_parpadeo(N10-1) =
                '0') then
                     digit_rota <= (others => '0');
             end if;
         when "10" =>
             digit_rota <= "0100";
             if(contador_activo = "0100" AND ini_parpadeo =
                '1' AND contador_parpadeo(contador_parpadeo'
                high-NP) = '0' AND contador_parpadeo(N10-1) =
                '0') then
                     digit_rota <= (others => '0');
             end if;
         when "11" =>
             digit_rota <= "1000";
```

```
                if(contador_activo = "1000" AND ini_parpadeo =
                    '1' AND contador_parpadeo(contador_parpadeo'
                    high-NP) = '0' AND contador_parpadeo(N10-1) =
                    '0') then
                        digit_rota <= (others => '0');
                end if;
            when others =>
                digit_rota <= "1111";
        end case;



    END PROCESS;

end rotacion_arq;
```

9.3.2. Código de la arquitectura del fichero de estímulos

```
---
   ************************************************************************

-- LIBRERIAS
--
   ************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;
```

```
--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_rotacion IS
END    test_rotacion;



--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_rotacion_arq OF test_rotacion IS
    --Declaracin  de componentes
    COMPONENT rotacion
        GENERIC (
              N20: INTEGER := 20;    -- 20 bits para el contador
                 de rotacin
              N10: INTEGER := 10;    -- 10 bits para el contador
                 de parpadeo
              NP: INTEGER := 2    -- Constante para el  clculo  de
                 parpadeos
        );
```

```
        Port (
            clk,reset       : in  STD_LOGIC;
            contador_activo: in std_logic_vector(3 downto 0);
            ini_parpadeo: in std_logic;
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
        );
    END COMPONENT;

    SIGNAL CLK, RESET, INI_PARPADEO, FIN_PARPADEO: std_logic;
    SIGNAL DIGIT_ROTA, CONTADOR_ACTIVO : std_logic_vector(3
       downto 0);

constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: rotacion
            GENERIC MAP (N20 => 2, N10 => 4, NP => 2)
            --GENERIC MAP (N20 => 2, N10 => 4, NP => 3)
            PORT MAP(
                clk             => CLK,
                reset           => RESET,
```

```
            ini_parpadeo => INI_PARPADEO,
            fin_parpadeo => FIN_PARPADEO,
            contador_activo => CONTADOR_ACTIVO,
            digit_rota    => DIGIT_ROTA
        );
--
   ================================================================================

-- Proceso de la entrada del reset inicial
--
   ================================================================================


PROCESS
BEGIN
    reset <='1';    wait for ciclo/3;
    --reset <='0';    wait for 10*ciclo;
    --reset <='1';    wait for ciclo/3;
    reset <='0';    wait;
END PROCESS;

 --
    ================================================================================

-- Proceso de la simulacin  de un reloj de entrada
--
   ================================================================================

PROCESS
BEGIN
    CLK <='0';    wait for ciclo/2;
```

```
    CLK <='1';    wait for ciclo/2;
END PROCESS;

--
   ============================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ============================================================================

tb: PROCESS
BEGIN
 contador_activo <= "0000";
 ini_parpadeo    <= '0';
 wait for 1.1*ciclo/2;

 ini_parpadeo <= '1';
 contador_activo(3) <= '1';
 --Descomentar esta seccin  para comprobar que al
    desactivarse la sean  ini_parpadeo para de incrementarse
     el contador
 -- de parpadeo

 --wait for 20*ciclo;
 --ini_parpadeo <= '0';

 --wait for 20*ciclo;
 --ini_parpadeo <= '1';
 wait;

end process tb;
```

```
END test_rotacion_arq;
```

9.3.3. Gráfica

Se pide simular el circuito con N20=2, N10=4 y NP=3

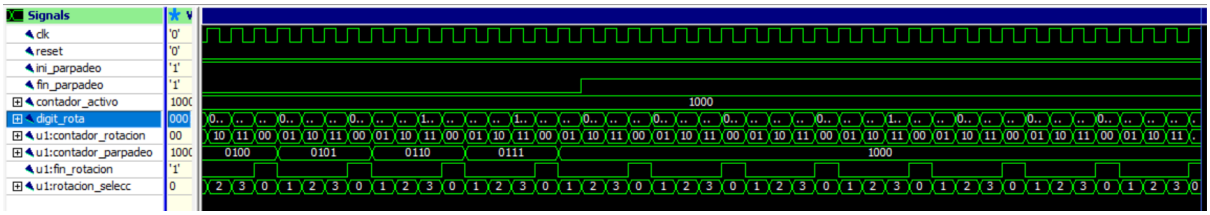

Figura 9.3: Simulación Rotación Completa

Capítulo 10

Ejercicio 5: Iluminación de los displays mediante rotación

10.1. Iluminación Básica (0.75 puntos)

10.1.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity rotacion is
        GENERIC (
            N20: INTEGER := 20    -- contador que fija el tiempo
               de iluminacin  de cada display
        );
    Port ( clk,reset        : in  STD_LOGIC;
           digit_rota       : out STD_LOGIC_VECTOR (3 downto 0)
         );
end rotacion;
```

```
architecture rotacion_arq of rotacion is
    signal contador_rotacion : unsigned(N20-1 downto 0) := (
       others => '1');  -- unsigned(19 downto 0)
    SIGNAL rotacion_selecc : UNSIGNED(1 DOWNTO 0);
    signal fin_rotacion : std_logic;

begin

   -- proceso para darle valores (asignar) la seal  '
      contador_rotacion'
   --Tambien se gestiona aqu el reset asncrono
   process(clk, reset)
   begin
        --Para evitar latches
        contador_rotacion <= contador_rotacion;

        if(reset = '1') then
                contador_rotacion <= (others => '0');
                fin_rotacion <= '0';

        elsif (rising_edge(clk)) then

                contador_rotacion <= contador_rotacion + 1;

                --Esto hay que ver si se puede poner de una forma
                    ms mona realmente estamos viendo si en el
                   siguiente ya desborda
                if((contador_rotacion + 1) = 0) then
                        fin_rotacion <= '1';
                else
                        fin_rotacion <= '0';
```

```
                end if;
        end if;
    end process;


    -- seal  auxiliar con los dos bits ms significativos del
       contador
    rotacion_selecc(1 DOWNTO 0) <=  contador_rotacion( (N20-1)
       downto (N20-2) );

    -- Proceso para asignar la salida 'digit_rota[]'
    PROCESS (rotacion_selecc)
    BEGIN

        case rotacion_selecc is
            when "00" =>  digit_rota <= "0001";
            when "01" =>  digit_rota <= "0010";
            when "10" =>  digit_rota <= "0100";
            when "11" =>  digit_rota <= "1000";
            when others =>  digit_rota <= "1111";
        end case;

     END PROCESS;

end rotacion_arq;
```

10.1.2. Código de la arquitectura del fichero de estímulos

```
---
    ********************************************************************

-- LIBRERIAS
```

```
--
   ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;



--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_rotacion IS
END    test_rotacion;



--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los estmulos)
--
   ********************************************************************

ARCHITECTURE test_rotacion_arq OF test_rotacion IS
    --Declaracin  de componentes
```

```
    COMPONENT rotacion
        GENERIC (
              N20: INTEGER := 20   -- Paramatrizable para
                 cualquier frecuencia de reloj
        );
        Port (
           clk,reset        : in  STD_LOGIC;
           digit_rota       : out STD_LOGIC_VECTOR (3 downto 0)
        );
    END COMPONENT;

    SIGNAL CLK,RESET: std_logic;
    SIGNAL DIGIT_ROTA : std_logic_vector(3 downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: rotacion
           GENERIC MAP (N20 => 3)
           PORT MAP(
               clk           => CLK,
```

```
            reset          => RESET,
            digit_rota     => DIGIT_ROTA
        );
--
   ================================================================================

-- Proceso de la entrada del reset inicial
--
   ================================================================================


PROCESS
BEGIN
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait for 10*ciclo;
    reset <='1';     wait for ciclo/3;
    reset <='0';     wait;
END PROCESS;

 --
    ================================================================================

-- Proceso de la simulacin  de un reloj de entrada
--
   ================================================================================

PROCESS
BEGIN
    CLK <='0';     wait for ciclo/2;
    CLK <='1';     wait for ciclo/2;
END PROCESS;
```

```
END test_rotacion_arq;
```

10.1.3. Gráfica

Se pide añadir una señal interna llamada fin_rotacion que se pondrá a 1 -durante un ciclo de reloj- cuando contador_rotacion desborda y pasa a 0. Habrá que indicarle al simulador que muestre esta señal interna.

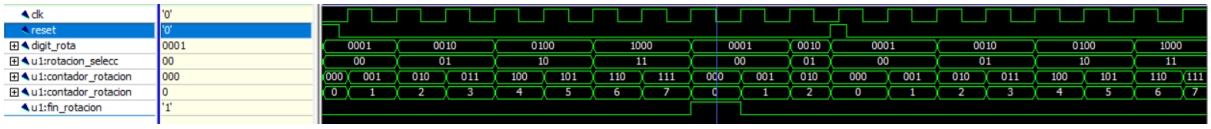

Figura 10.1: Simulación Iluminación Básica

10.2. Iluminación Temporizada (1.5 puntos)

10.2.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity rotacion is
        GENERIC (
            N20: INTEGER := 20;    -- 20 bits para el contador de
               rotacin
            N10: INTEGER := 10;    -- 10 bits para el contador de
               parpadeo
            NP: INTEGER := 2    -- Constante para el  c lculo  de
               parpadeos
        );
    Port ( clk,reset        : in  STD_LOGIC;
```

```
            contador_activo: in std_logic_vector(3 downto 0);
            ini_parpadeo: in std_logic;
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
          );
end rotacion;

architecture rotacion_arq of rotacion is
    signal contador_rotacion : unsigned(N20-1 downto 0) := (
       others => '1');  -- unsigned(19 downto 0)
    signal contador_parpadeo : unsigned(N10-1 downto 0) := (
       others => '1');  -- unsigned(9 downto 0)
    SIGNAL rotacion_selecc : UNSIGNED(1 DOWNTO 0);
    signal fin_rotacion : std_logic;

begin

   -- proceso para darle valores (asignar) la seal  '
      contador_rotacion'
   --Tambien se gestiona aqu el reset asncrono
   process(clk, reset, ini_parpadeo, fin_rotacion)
   begin
        --Para evitar latches
        contador_rotacion <= contador_rotacion;
        contador_parpadeo <= contador_parpadeo;

        if(reset = '1') then
                contador_rotacion <= (others => '0');
                contador_parpadeo <= (others => '0');
                fin_rotacion <= '0';
                fin_parpadeo <= '0';
```

```
    elsif (rising_edge(clk)) then

            contador_rotacion <= contador_rotacion + 1;

            -- Vemos si al sumarle uno ms desborda
            if((contador_rotacion + 1) = 0) then
                    fin_rotacion <= '1';
            else
                    fin_rotacion <= '0';
            end if;

            if(contador_parpadeo(N10-1) = '0') then
                    fin_parpadeo <= '0';

                    if(fin_rotacion = '1'  AND ini_parpadeo =
                        '1') then
                            contador_parpadeo <=
                               contador_parpadeo + 1;
                    end if;
            else
                    fin_parpadeo <= '1';
            end if;
    end if;
end process;


-- seal  auxiliar con los dos bits ms significativos del
   contador
rotacion_selecc(1 DOWNTO 0) <=  contador_rotacion( (N20-1)
   downto (N20-2) );
```

```
    -- Proceso para asignar la salida 'digit_rota[]'
    PROCESS (rotacion_selecc)
    BEGIN

         case rotacion_selecc is
             when "00" =>  digit_rota <= "0001";
             when "01" =>  digit_rota <= "0010";
             when "10" =>  digit_rota <= "0100";
             when "11" =>  digit_rota <= "1000";
             when others =>  digit_rota <= "1111";
         end case;

     END PROCESS;

end rotacion_arq;
```

10.2.2. Código de la arquitectura del fichero de estímulos

```
---
    ************************************************************************

-- LIBRERIAS
--
    ************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;
```

```
--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_rotacion IS
END    test_rotacion;



--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_rotacion_arq OF test_rotacion IS
    --Declaracin  de componentes
    COMPONENT rotacion
        GENERIC (
              N20: INTEGER := 20;    -- 20 bits para el contador
                 de rotacin
              N10: INTEGER := 10;    -- 10 bits para el contador
                 de parpadeo
              NP: INTEGER := 2    -- Constante para el  clculo  de
                 parpadeos
        );
```

```
        Port (
            clk,reset       : in  STD_LOGIC;
            contador_activo: in std_logic_vector(3 downto 0);
            ini_parpadeo: in std_logic;
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
        );
    END COMPONENT;

    SIGNAL CLK, RESET, INI_PARPADEO, FIN_PARPADEO: std_logic;
    SIGNAL DIGIT_ROTA, CONTADOR_ACTIVO : std_logic_vector(3
       downto 0);

constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: rotacion
           GENERIC MAP (N20 => 2, N10 => 4)
           PORT MAP(
               clk           => CLK,
               reset         => RESET,
               ini_parpadeo => INI_PARPADEO,
```

```
            fin_parpadeo => FIN_PARPADEO,
            contador_activo => CONTADOR_ACTIVO,
            digit_rota    => DIGIT_ROTA
        );
--
   ============================================================================

-- Proceso de la entrada del reset inicial
--
   ============================================================================


PROCESS
BEGIN
    reset <='1';    wait for ciclo/3;
    reset <='0';    wait for 10*ciclo;
    reset <='1';    wait for ciclo/3;
    reset <='0';    wait;
END PROCESS;

 --
    ============================================================================

-- Proceso de la simulacin  de un reloj de entrada
--
   ============================================================================

PROCESS
BEGIN
    CLK <='0';    wait for ciclo/2;
    CLK <='1';    wait for ciclo/2;
```

```
    END PROCESS;

    --
       ============================================================================

    -- Proceso para el resto de entradas; ENABLE_test
    --
       ============================================================================

    tb: PROCESS
    BEGIN
     contador_activo <= (others => '0');
     ini_parpadeo    <= '0';
     wait for 1.1*ciclo/2;

     ini_parpadeo <= '1';
     contador_activo <= "1000";
     --Descomentar esta seccin  para comprobar que al
        desactivarse la sean  ini_parpadeo para de incrementarse
         el contador
     -- de parpadeo

     --wait for 20*ciclo;
     --ini_parpadeo <= '0';

     --wait for 20*ciclo;
     --ini_parpadeo <= '1';
     wait;

    end process tb;
END test_rotacion_arq;
```

10.2.3. Gráfica

Se pide realizar la simulación que se muestra pero solo hasta que la señal fin_parpadeo se pone a 1. El valor de contador_activo=2 y la constante NP=2 son irrelevantes, por ahora. Las señales contador_rotacion, rotacion_selecc y digit_rota se mostrarán en formato binario.

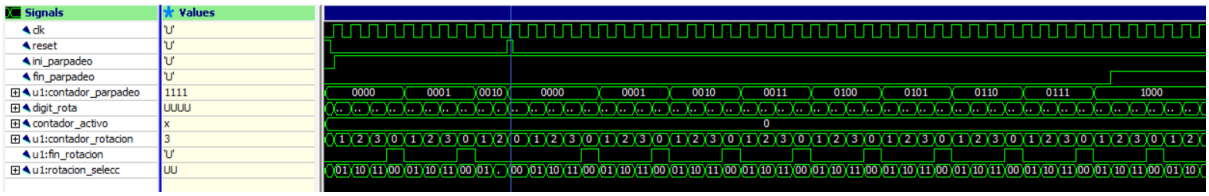

Figura 10.2: Simulación Iluminación Completa

10.3. Iluminación completa (1 punto)

10.3.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity rotacion is
        GENERIC (
            N20: INTEGER := 20;    -- 20 bits para el contador de
               rotacin
            N10: INTEGER := 10;    -- 10 bits para el contador de
               parpadeo
            NP: INTEGER := 2    -- Constante para el  c lculo  de
               parpadeos
        );
    Port ( clk,reset        : in  STD_LOGIC;
           contador_activo: in std_logic_vector(3 downto 0);
           ini_parpadeo: in std_logic;
```

```
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
          );
end rotacion;

architecture rotacion_arq of rotacion is
    signal contador_rotacion : unsigned(N20-1 downto 0) := (
       others => '1');  -- unsigned(19 downto 0)
    signal contador_parpadeo : unsigned(N10-1 downto 0) := (
       others => '1');  -- unsigned(9 downto 0)
    SIGNAL rotacion_selecc : UNSIGNED(1 DOWNTO 0);
    signal fin_rotacion : std_logic;

begin

   -- proceso para darle valores (asignar) la seal  '
      contador_rotacion'
   --Tambien se gestiona aqu el reset asncrono
   process(clk, reset, ini_parpadeo, fin_rotacion)
   begin
        --Para evitar latches
        contador_rotacion <= contador_rotacion;
        contador_parpadeo <= contador_parpadeo;

        if(reset = '1') then
                contador_rotacion <= (others => '0');
                contador_parpadeo <= (others => '0');
                fin_rotacion <= '0';
                fin_parpadeo <= '0';

        elsif (rising_edge(clk)) then
```

```
                contador_rotacion <= contador_rotacion + 1;

                -- Vemos si al sumarle uno ms desborda
                if((contador_rotacion + 1) = 0) then
                        fin_rotacion <= '1';
                else
                        fin_rotacion <= '0';
                end if;

                if(contador_parpadeo(N10-1) = '0') then
                        fin_parpadeo <= '0';

                        if(fin_rotacion = '1'  AND ini_parpadeo =
                            '1') then
                                contador_parpadeo <=
                                   contador_parpadeo + 1;
                        end if;
                else
                        fin_parpadeo <= '1';
                end if;
        end if;
end process;


-- seal  auxiliar con los dos bits ms significativos del
   contador
rotacion_selecc(1 DOWNTO 0) <=  contador_rotacion( (N20-1)
   downto (N20-2) );

-- Proceso para asignar la salida 'digit_rota[]'
```

```
PROCESS (rotacion_selecc)
BEGIN

     case rotacion_selecc is
         when "00" =>
             digit_rota <= "0001";
             if(contador_activo = "0001" AND ini_parpadeo =
                '1' AND contador_parpadeo(contador_parpadeo'
                high-NP) = '0' AND contador_parpadeo(N10-1) =
                '0') then
                     digit_rota <= (others => '0');
             end if;
         when "01" =>
             digit_rota <= "0010";
             if(contador_activo = "0010" AND ini_parpadeo =
                '1' AND contador_parpadeo(contador_parpadeo'
                high-NP) = '0' AND contador_parpadeo(N10-1) =
                '0') then
                     digit_rota <= (others => '0');
             end if;
         when "10" =>
             digit_rota <= "0100";
             if(contador_activo = "0100" AND ini_parpadeo =
                '1' AND contador_parpadeo(contador_parpadeo'
                high-NP) = '0' AND contador_parpadeo(N10-1) =
                '0') then
                     digit_rota <= (others => '0');
             end if;
         when "11" =>
             digit_rota <= "1000";
```

```
                if(contador_activo = "1000" AND ini_parpadeo =
                    '1' AND contador_parpadeo(contador_parpadeo'
                    high-NP) = '0' AND contador_parpadeo(N10-1) =
                    '0') then
                        digit_rota <= (others => '0');
                end if;
            when others =>
                digit_rota <= "1111";
        end case;




    END PROCESS;

end rotacion_arq;
```

10.3.2. Código de la arquitectura del fichero de estímulos

```
---
   ************************************************************************

-- LIBRERIAS
--
   ************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;
```

```
--
   ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
   tiene)
--
   ********************************************************************

ENTITY test_rotacion IS
END    test_rotacion;



--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_rotacion_arq OF test_rotacion IS
    --Declaracin  de componentes
    COMPONENT rotacion
        GENERIC (
              N20: INTEGER := 20;    -- 20 bits para el contador
                 de rotacin
              N10: INTEGER := 10;    -- 10 bits para el contador
                 de parpadeo
              NP: INTEGER := 2    -- Constante para el  clculo  de
                 parpadeos
        );
```

```
        Port (
            clk,reset       : in  STD_LOGIC;
            contador_activo: in std_logic_vector(3 downto 0);
            ini_parpadeo: in std_logic;
            fin_parpadeo: out std_logic;
            digit_rota      : out STD_LOGIC_VECTOR (3 downto 0)
        );
    END COMPONENT;

    SIGNAL CLK, RESET, INI_PARPADEO, FIN_PARPADEO: std_logic;
    SIGNAL DIGIT_ROTA, CONTADOR_ACTIVO : std_logic_vector(3
       downto 0);

constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: rotacion
            GENERIC MAP (N20 => 2, N10 => 4, NP => 2)
            --GENERIC MAP (N20 => 2, N10 => 4, NP => 3)
            PORT MAP(
                clk             => CLK,
                reset           => RESET,
```

```
            ini_parpadeo => INI_PARPADEO,
            fin_parpadeo => FIN_PARPADEO,
            contador_activo => CONTADOR_ACTIVO,
            digit_rota    => DIGIT_ROTA
        );
--
   ================================================================================

-- Proceso de la entrada del reset inicial
--
   ================================================================================


PROCESS
BEGIN
    reset <='1';    wait for ciclo/3;
    --reset <='0';    wait for 10*ciclo;
    --reset <='1';    wait for ciclo/3;
    reset <='0';    wait;
END PROCESS;

 --
    ================================================================================

-- Proceso de la simulacin  de un reloj de entrada
--
   ================================================================================

PROCESS
BEGIN
    CLK <='0';    wait for ciclo/2;
```

```
    CLK <='1';    wait for ciclo/2;
END PROCESS;

--
   ============================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ============================================================================

tb: PROCESS
BEGIN
 contador_activo <= "0000";
 ini_parpadeo    <= '0';
 wait for 1.1*ciclo/2;

 ini_parpadeo <= '1';
 contador_activo(3) <= '1';
 --Descomentar esta seccin  para comprobar que al
    desactivarse la sean  ini_parpadeo para de incrementarse
     el contador
 -- de parpadeo

 --wait for 20*ciclo;
 --ini_parpadeo <= '0';

 --wait for 20*ciclo;
 --ini_parpadeo <= '1';
 wait;

end process tb;
```

```
END test_rotacion_arq;
```

10.3.3. Gráfica

Se pide simular el circuito con N20=2, N10=4 y NP=3

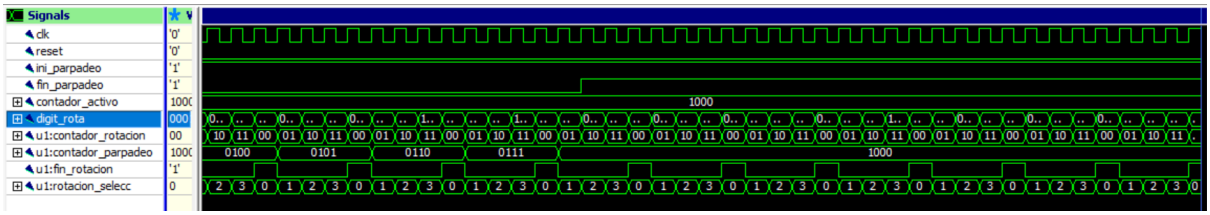

Figura 10.3: Simulación Rotación Completa

Capítulo 11

Ejercicio 6: Control del sistema

11.1. Control básico del sistema (1.5 puntos)

11.1.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity control is
    Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;
        arriba     : in STD_LOGIC;
        abajo      : in STD_LOGIC;
        izquierda  : in STD_LOGIC;
        derecha    : in STD_LOGIC;
     --   fin_parpadeo     : in  STD_LOGIC;
     --   ini_parpadeo     : out  STD_LOGIC;
        enable_contador : out STD_LOGIC_VECTOR (3 downto 0);
                                -- un pulso de 1 ciclo de reloj
```

```
            para que cuente solo 1
        contador_activo : out STD_LOGIC_VECTOR (3 downto 0)
    );
end control;

architecture control_arq of control is
    signal regDesplazaContadorActivo : std_logic_vector(3 downto
       0);
    TYPE ESTADOS IS (reposo, izquierda_derecha_enable,
       espera_boton_pulsado, arriba_abajo_enable);  -- ,
       parpadeando);
    SIGNAL estado_s, estado_c : ESTADOS;

begin
    contador_activo <= regDesplazaContadorActivo;

    PROCESS (clk,reset)
    BEGIN
        IF (reset='1') THEN                  -- Reset activo a
           nviel alto
            estado_s  <=reposo;
            regDesplazaContadorActivo <= "1000";



        ELSIF (rising_edge(clk)) THEN
            estado_s  <= estado_c;

            -- actualiza regDesplazaContadorActivo para que rote
               a derecha o izquierda
            IF (estado_s = izquierda_derecha_enable) THEN
```

```
                IF(izquierda = '1') then
                        regDesplazaContadorActivo <=
                           regDesplazaContadorActivo(2
                           downto 0) &
                           regDesplazaContadorActivo(3);
                ELSIF(derecha = '1') then
                        regDesplazaContadorActivo <=
                           regDesplazaContadorActivo(0) &
                           regDesplazaContadorActivo(3
                           downto 1);
                END IF;
        END IF;


    END IF;
END PROCESS;



PROCESS (estado_s,arriba,abajo,derecha,izquierda) --,
   fin_parpadeo)
BEGIN
    -- ---------------------------------
    -- Valores por defecto
    -- ---------------------------------
    estado_c        <= estado_s;
    enable_contador <= "0000";  -- Se activarA en un estado
       transitorio de un ciclo de reloj
  --  ini_parpadeo    <= '0';   -- no se usa en la versiOn
     basica
    -- ---------------------------------
```

```
CASE estado_s IS
    WHEN reposo =>
         if(derecha = '1' OR izquierda = '1') then
                estado_c <= izquierda_derecha_enable;
         elsif(arriba = '1' OR abajo = '1') then
                estado_c <= arriba_abajo_enable;

         else
                estado_c <= reposo;
         end if;

    WHEN izquierda_derecha_enable =>

                 estado_c <= espera_boton_pulsado;

    WHEN espera_boton_pulsado =>
        --Si se pulsa cualquier  b o t n  nos mantenemos en
           el estado
         if (arriba = '1' OR abajo = '1' OR arriba = '1'
           OR abajo = '1') then
                estado_c <= espera_boton_pulsado;
         else
                estado_c <= reposo;
         end if;

    WHEN arriba_abajo_enable =>
          enable_contador  <= regDesplazaContadorActivo ;
          estado_c <= espera_boton_pulsado;

    WHEN OTHERS =>
```

```
                estado_c <= estado_c;

        END CASE;
    END PROCESS;

end control_arq;
```

11.1.2. Código de la arquitectura del fichero de estímulos

```
---
    ********************************************************************

-- LIBRERIAS
--
    ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;




--
    ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
    tiene)
--
    ********************************************************************

ENTITY test_control IS
END    test_control;
```

```
--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_control_arq OF test_control IS
    --Declaracin  de componentes
    COMPONENT control
      Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;
        arriba     : in STD_LOGIC;
        abajo      : in STD_LOGIC;
        izquierda  : in STD_LOGIC;
        derecha    : in STD_LOGIC;
     --    fin_parpadeo     : in  STD_LOGIC;
     --    ini_parpadeo     : out  STD_LOGIC;
        enable_contador : out STD_LOGIC_VECTOR (3 downto 0);
                                -- un pulso de 1 ciclo de reloj
           para que cuente solo 1
        contador_activo : out STD_LOGIC_VECTOR (3 downto 0)
      );
    END COMPONENT;

    SIGNAL CLK,ARRIBA,ABAJO,IZQUIERDA,DERECHA,RESET: std_logic;
                      -- ,INI_PARPADEO,FIN_PARPADEO
```

```
    SIGNAL ENABLE_CONTADOR,CONTADOR_ACTIVO : std_logic_vector(3
       downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: control PORT MAP(
           clk        => CLK,
           reset      => RESET,
           arriba     =>  ARRIBA,
           abajo      =>  ABAJO,
           izquierda  =>  IZQUIERDA,
           derecha    =>  DERECHA,
       --    ini_parpadeo     =>  INI_PARPADEO,
       --    fin_parpadeo     =>  FIN_PARPADEO,
           enable_contador => ENABLE_CONTADOR,   -- un pulso de 1
               ciclo de reloj para que cuente solo 1
           contador_activo  => CONTADOR_ACTIVO);


    --
```

```
   ================================================================

-- Proceso de la entrada de reloj. Se ejecuta indefinidamente
    ya que no tiene "WAIT;"
--
   ================================================================

PROCESS
BEGIN
    CLK <='0';     wait for ciclo/2;
    CLK <='1';     wait for ciclo/2;
END PROCESS;

GenReset: process
begin
    RESET<= '1';      wait for 9*ciclo/4;
    RESET<= '0';      wait;
end process GenReset;

--
   ================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ================================================================

tb: PROCESS
BEGIN
    ARRIBA    <= '0';
    ABAJO     <= '0';
    DERECHA   <= '0';
```

```
        IZQUIERDA <= '0';
--        FIN_PARPADEO <= '0';
        wait for 13*ciclo/4;
        ARRIBA   <= '1';
        wait for 1.5*ciclo;
        ARRIBA <= '0';
        wait for 3*ciclo;

        ABAJO  <= '1';
        wait for 1.5*ciclo;
        ABAJO <= '0';
        wait for 3*ciclo;

        IZQUIERDA <= '1';
        wait for 1.5*ciclo;
        IZQUIERDA <= '0';
        wait for 3*ciclo;

        DERECHA   <= '1';
        wait for 1.5*ciclo;
        DERECHA <= '0';
        wait;
    end process tb;

END test_control_arq;
```

11.1.3. Gráfica

Se pide simular un pulso en la entrada izquierda y, al cabo de unos ciclos, un pulso en la entrada derecha. Esto hará que el registro de desplazamiento rote en un sentido y el contrario de forma que la salida contador_activo también lo hará.

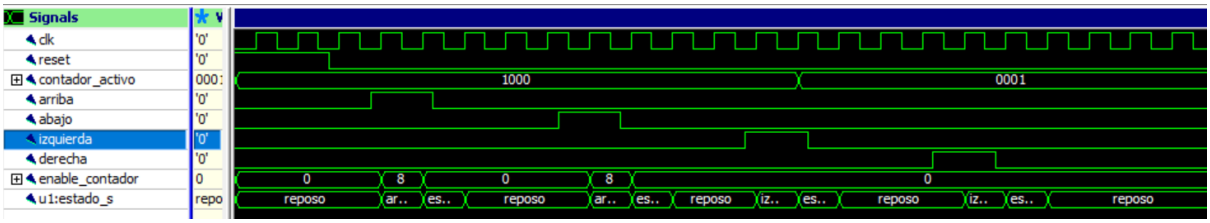

Figura 11.1: Simulación Control Básico

11.2. Control con parpadeo (1.5 puntos)

11.2.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity control is
    Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;
        arriba     : in STD_LOGIC;
        abajo      : in STD_LOGIC;
        izquierda  : in STD_LOGIC;
        derecha    : in STD_LOGIC;
        fin_parpadeo     : in  STD_LOGIC;
        ini_parpadeo     : out  STD_LOGIC;
        enable_contador : out STD_LOGIC_VECTOR (3 downto 0);
        contador_activo : out STD_LOGIC_VECTOR (3 downto 0)

    );
end control;

architecture control_arq of control is
    signal regDesplazaContadorActivo : std_logic_vector(3 downto
       0);
    TYPE ESTADOS IS (reposo, izquierda_derecha_enable,
       espera_boton_pulsado, arriba_abajo_enable, parpadeando);
```

Capítulo 12

Ejercicio 6: Control del sistema

12.1. Control básico del sistema (1.5 puntos)

12.1.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity control is
    Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;
        arriba     : in STD_LOGIC;
        abajo      : in STD_LOGIC;
        izquierda  : in STD_LOGIC;
        derecha    : in STD_LOGIC;
     --   fin_parpadeo     : in  STD_LOGIC;
     --   ini_parpadeo     : out  STD_LOGIC;
        enable_contador : out STD_LOGIC_VECTOR (3 downto 0);
                                -- un pulso de 1 ciclo de reloj
```

```
            para que cuente solo 1
        contador_activo : out STD_LOGIC_VECTOR (3 downto 0)
    );
end control;

architecture control_arq of control is
    signal regDesplazaContadorActivo : std_logic_vector(3 downto
       0);
    TYPE ESTADOS IS (reposo, izquierda_derecha_enable,
       espera_boton_pulsado, arriba_abajo_enable);  -- ,
       parpadeando);
    SIGNAL estado_s, estado_c : ESTADOS;

begin
    contador_activo <= regDesplazaContadorActivo;

    PROCESS (clk,reset)
    BEGIN
        IF (reset='1') THEN                  -- Reset activo a
           nviel alto
            estado_s  <=reposo;
            regDesplazaContadorActivo <= "1000";



        ELSIF (rising_edge(clk)) THEN
            estado_s  <= estado_c;

            -- actualiza regDesplazaContadorActivo para que rote
               a derecha o izquierda
            IF (estado_s = izquierda_derecha_enable) THEN
```

```
                IF(izquierda = '1') then
                        regDesplazaContadorActivo <=
                           regDesplazaContadorActivo(2
                           downto 0) &
                           regDesplazaContadorActivo(3);
                ELSIF(derecha = '1') then
                        regDesplazaContadorActivo <=
                           regDesplazaContadorActivo(0) &
                           regDesplazaContadorActivo(3
                           downto 1);
                END IF;
        END IF;


    END IF;
END PROCESS;



PROCESS (estado_s,arriba,abajo,derecha,izquierda) --,
   fin_parpadeo)
BEGIN
    -- ---------------------------------
    -- Valores por defecto
    -- ---------------------------------
    estado_c        <= estado_s;
    enable_contador <= "0000";  -- Se activarA en un estado
       transitorio de un ciclo de reloj
  --  ini_parpadeo    <= '0';   -- no se usa en la versiOn
     basica
    -- ---------------------------------
```

```
CASE estado_s IS
    WHEN reposo =>
         if(derecha = '1' OR izquierda = '1') then
                estado_c <= izquierda_derecha_enable;
         elsif(arriba = '1' OR abajo = '1') then
                estado_c <= arriba_abajo_enable;

         else
                estado_c <= reposo;
         end if;

    WHEN izquierda_derecha_enable =>

                 estado_c <= espera_boton_pulsado;

    WHEN espera_boton_pulsado =>
        --Si se pulsa cualquier  b o t n  nos mantenemos en
           el estado
         if (arriba = '1' OR abajo = '1' OR arriba = '1'
           OR abajo = '1') then
                estado_c <= espera_boton_pulsado;
         else
                estado_c <= reposo;
         end if;

    WHEN arriba_abajo_enable =>
          enable_contador  <= regDesplazaContadorActivo ;
          estado_c <= espera_boton_pulsado;

    WHEN OTHERS =>
```

```
                estado_c <= estado_c;

        END CASE;
    END PROCESS;

end control_arq;
```

12.1.2. Código de la arquitectura del fichero de estímulos

```
---
    ********************************************************************

-- LIBRERIAS
--
    ********************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;




--
    ********************************************************************

-- ENTIDAD     (entradas/salidas, el fichero de simulacin  no
    tiene)
--
    ********************************************************************

ENTITY test_control IS
END    test_control;
```

```
--
   ********************************************************************

-- ARQUITECTURA   (descripcin  de los  estmulos)
--
   ********************************************************************

ARCHITECTURE test_control_arq OF test_control IS
    --Declaracin  de componentes
    COMPONENT control
      Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;
        arriba     : in STD_LOGIC;
        abajo      : in STD_LOGIC;
        izquierda  : in STD_LOGIC;
        derecha    : in STD_LOGIC;
     --    fin_parpadeo     : in  STD_LOGIC;
     --    ini_parpadeo     : out  STD_LOGIC;
        enable_contador : out STD_LOGIC_VECTOR (3 downto 0);
                                -- un pulso de 1 ciclo de reloj
           para que cuente solo 1
        contador_activo : out STD_LOGIC_VECTOR (3 downto 0)
      );
    END COMPONENT;

    SIGNAL CLK,ARRIBA,ABAJO,IZQUIERDA,DERECHA,RESET: std_logic;
                      -- ,INI_PARPADEO,FIN_PARPADEO
```

```
    SIGNAL ENABLE_CONTADOR,CONTADOR_ACTIVO : std_logic_vector(3
       downto 0);


constant ciclo: time := 10 ns;  -- hay que dejar un espacio entre
    1 y ms

BEGIN
    --
       ////////////////////////////////////////////////////////////////

    -- Se crea el componente U1 y se conecta a las  s e ales
        internas de la arquitectura
    --
       ////////////////////////////////////////////////////////////////

    U1: control PORT MAP(
           clk        => CLK,
           reset      => RESET,
           arriba     =>  ARRIBA,
           abajo      =>  ABAJO,
           izquierda  =>  IZQUIERDA,
           derecha    =>  DERECHA,
       --    ini_parpadeo     =>  INI_PARPADEO,
       --    fin_parpadeo     =>  FIN_PARPADEO,
           enable_contador => ENABLE_CONTADOR,   -- un pulso de 1
               ciclo de reloj para que cuente solo 1
           contador_activo  => CONTADOR_ACTIVO);


    --
```

```
   ================================================================

-- Proceso de la entrada de reloj. Se ejecuta indefinidamente
    ya que no tiene "WAIT;"
--
   ================================================================

PROCESS
BEGIN
    CLK <='0';     wait for ciclo/2;
    CLK <='1';     wait for ciclo/2;
END PROCESS;

GenReset: process
begin
    RESET<= '1';      wait for 9*ciclo/4;
    RESET<= '0';      wait;
end process GenReset;

--
   ================================================================

-- Proceso para el resto de entradas; ENABLE_test
--
   ================================================================

tb: PROCESS
BEGIN
    ARRIBA    <= '0';
    ABAJO     <= '0';
    DERECHA   <= '0';
```

```
        IZQUIERDA <= '0';
--        FIN_PARPADEO <= '0';
        wait for 13*ciclo/4;
        ARRIBA   <= '1';
        wait for 1.5*ciclo;
        ARRIBA <= '0';
        wait for 3*ciclo;

        ABAJO  <= '1';
        wait for 1.5*ciclo;
        ABAJO <= '0';
        wait for 3*ciclo;

        IZQUIERDA <= '1';
        wait for 1.5*ciclo;
        IZQUIERDA <= '0';
        wait for 3*ciclo;

        DERECHA   <= '1';
        wait for 1.5*ciclo;
        DERECHA <= '0';
        wait;
    end process tb;

END test_control_arq;
```

12.1.3. Gráfica

Se pide simular un pulso en la entrada izquierda y, al cabo de unos ciclos, un pulso en la entrada derecha. Esto hará que el registro de desplazamiento rote en un sentido y el contrario de forma que la salida contador_activo también lo hará.

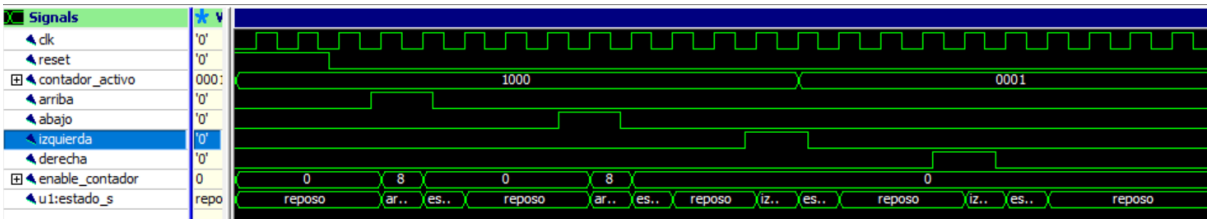

Figura 12.1: Simulación Control Básico

12.2. Control con parpadeo (1.5 puntos)

12.2.1. Código de la arquitectura del diseño

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity control is
    Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;
        arriba     : in STD_LOGIC;
        abajo      : in STD_LOGIC;
        izquierda  : in STD_LOGIC;
        derecha    : in STD_LOGIC;
        fin_parpadeo     : in  STD_LOGIC;
        ini_parpadeo     : out  STD_LOGIC;
        enable_contador : out STD_LOGIC_VECTOR (3 downto 0);
        contador_activo : out STD_LOGIC_VECTOR (3 downto 0)

    );
end control;

architecture control_arq of control is
    signal regDesplazaContadorActivo : std_logic_vector(3 downto
       0);
    TYPE ESTADOS IS (reposo, izquierda_derecha_enable,
       espera_boton_pulsado, arriba_abajo_enable, parpadeando);
```

Capítulo 13

Ejercicio 7: Diseño Completo

Para la realización de este ejercicio, se ha generado el fichero `.bit` que se añade junto con el resto de archivos. Además, se adjunta una imagen de la simulación:

Se pide implementar el diseño completo en Vivado y dejar en la documentación el fichero 'bitstream' generado. El fichero de restricciones lo tenéis en la documentación

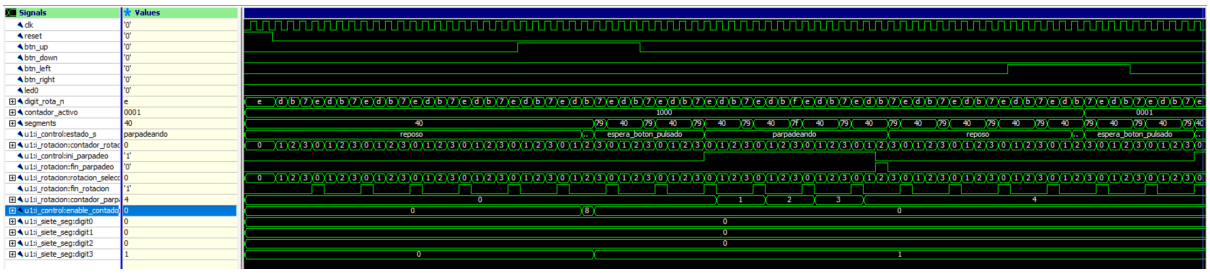

Figura 13.1: Simulación Proyecto Completo