Memoria

Sistemas Electrónicos para la Automatización

Docente: *Jorge Romero Sánchez*

Ejercicios básicos

***Jorge Benavides Macías**: jorge2@uma.es*

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Contenido

Lista de figuras

Lista de códigos

1. Especificaciones

Del ejercicio:

- Seleccionar dos de los circuitos combinacionales propuestos en las transparencias 7 a 10 de vuestra elección y dos de los circuitos secuenciales propuestos en las transparencias 12 a 17, también de vuestra elección.
- Explicar brevemente en un estudio previo, cómo funcionan los modelos (para la memoria).
- Generar un proyecto en Vivado (uno para cada modelo elegido).
- Generar, para la fase de simulación, un Test Bench con manejo de ficheros (modelo en Tema 5) que ilustre el funcionamiento correcto de los mismos.
- Sintetizar a nivel RTL (RTL Analysis) para comprobar que son efectivamente combinacionales ó secuenciales.

De la memoria:

- Elaborar un informe que ilustre el procedimiento de diseño, simulación y síntesis de todo el proceso seguido.
- Es posible utilizar capturas de pantalla (cronogramas de simulación, esquemáticos RTL), etc.
- Incluir en el PDF los códigos VHDL empleados y la captura de los ficheros .CSV generados y del fichero original de estímulos en función del retardo.

2. Soluciones

2.1. Multiplexor

2.1.1. Diseño

Estudio teórico

El multiplexor es un dispositivo **combinacional** que tiene *n* entradas para datos, una salida y una serie de señales de control (*c*) que dependen del número *n* de entradas. La relación entre el número de entradas y las señales de control es:

$$n = 2^c \tag{1}$$

Estructura

Como puede observarse en el Código 1 se describe un dispositivo de entradas genéricas que dependen del parámetro *n* (línea 6), el cual tiene un valor por defecto de 8 *bits*.

Cuenta con cuatro puertos, dos de entrada de tipo vector lógico (*std_logic_vector*) que en nuestro caso llamaremos datos, una salida que también es del tipo vector lógico y un selector binario, el dispositivo cumple la Ecuación 1.

La arquitectura del dispositivo, expuesta en el Código 2, es bastante sencilla, cuenta con una lista de sensibilidades la cual incluye los datos y el selector, y el núcleo del funcionamiento del dispositivo es un condicional que permite asignar una de las dos entradas a la salida.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity MuxV_2a1 is
5    generic(
6      n : integer := 8);
7    port(
8      x0  : in  std_logic_vector(n-1 downto 0);
9      x1  : in  std_logic_vector(n-1 downto 0);
10     sel : in  std_logic;
11     y   : out std_logic_vector(n-1 downto 0));
12 end MuxV_2a1;
```

Código 1: Entidad del MuxV2a1

```
architecture Behavioral of MuxV_2a1 is
begin
  process(sel, x0, x1)
  begin
    if sel = '1' then
      y <= x1;
    else
      y <= x0;
    end if;
  end process;
end Behavioral;
```

Código 2: Arquitectura del MuxV2a1

2.1.2. Simulación

Test Bench

Para realizar la simulación del dispositivo hemos usado la plantilla del campus "Test_Bench_Fichero" con una serie de modificaciones, se observa claramente en el Código 3 que se ha instanciado el componente MuxV2_1 (Código 1), se han declarado una serie de señales internas, algunas genéricas e inicializadas con el valor 'U' (*"uninitialized "*) para trabajar en simulación, y que en el *DUT* (*Device Under Test*) se ha asociado a cada puerto del componente una señal interna. Dado que es un dispositivo genérico también se ha definido una constante de tipo *integer* con un valor reducido para simulación.

En el proceso "Estímulos_Desde_Fichero" también hemos modificado el tamaño del input data, ya que solo necesitamos 3 estímulos (línea 31).

```
entity Test_Bench_Fichero is
--  Port ( );
end Test_Bench_Fichero;

architecture Comportamiento of Test_Bench_Fichero is

  component MuxV_2a1
    generic(
      n : integer := 8);
    port(
      x0  : in  std_logic_vector(n-1 downto 0);
      x1  : in  std_logic_vector(n-1 downto 0);
      sel : in  std_logic;
```

```
14        y   : out std_logic_vector(n-1 downto 0));
15    end Component MuxV_2a1;
16
17    constant n: integer := 1;
18    signal x0_interno, x1_interno, y_interno : std_logic_vector(n-1 downto 0):=
 ↪  (others => 'U');
19    signal sel_interno : std_logic := 'U';
20
21  begin
22
23    DUT : MuxV_2a1
24      generic map (n)
25      port map(
26        x0 => x0_interno,
27        x1 => x1_interno,
28        sel => sel_interno,
29        y => y_interno);
30  [...]
31      variable Input_Data   : BIT_VECTOR(2 downto 0) := (OTHERS => '0');
```

Código 3: Test Bench del Mux2a1

El código completo está definido en la Sección 3 en el Código 14.

Fichero de estímulos

Las dos entradas de datos tienen un valor aleatorio en este caso '0' y '1' para observar, sin dificultades, la diferencia y el selector va cambiando en cada paso.

```
#Fichero de Estímulos de Multiplexor 2 a 1.
#Device Name: Discovery2NI
#Nombre: Izan Amador, Jorge Benavides
#Fecha: 6 de Diciembre de 2022.
#
# Delay Time (ns) Input (x0,x1,sel)

  10 ns 010
  10 ns 011
  10 ns 010
  10 ns 011
  10 ns 010
  10 ns 011
```

Cronogramas de simulación

Se observa en la Figura 1 que la salida cambia en función de la selección, por lo tanto el dispositivo cumple con el funcionamiento esperado.

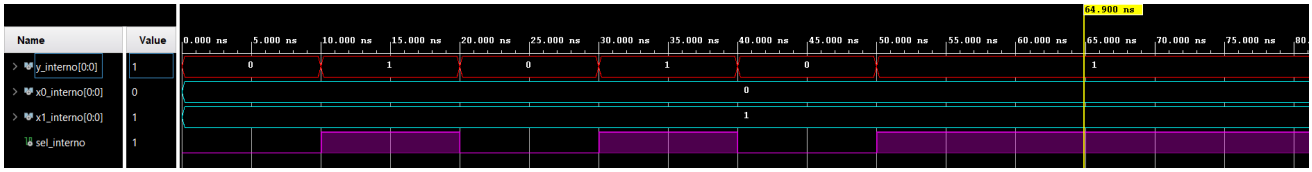

Figura 1: Simulación de multiplexor 2 a 1

Fichero CSV generado

El fichero de simulación del campus está pensado para una posible verificación con el *Analog Discovery*, la Figura 2 es el CSV generado.

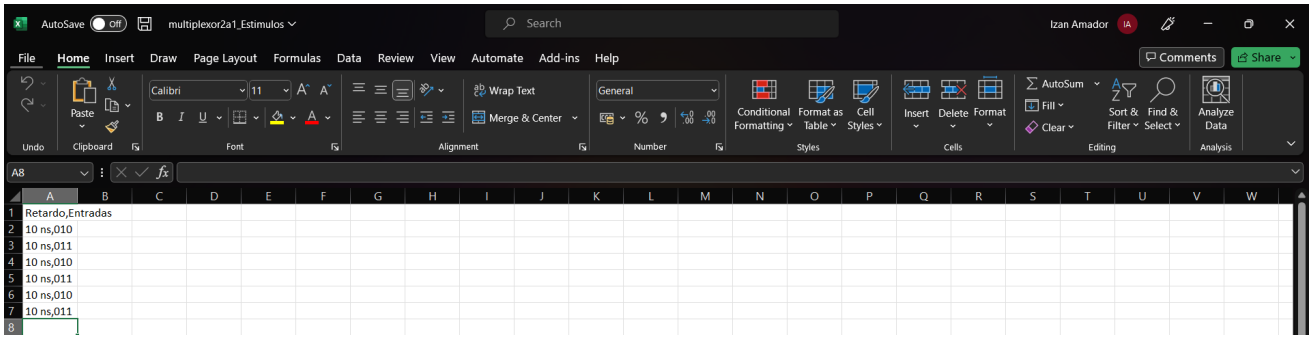

Figura 2: Fichero CSV generado para el MUX2a1

2.1.3. Síntesis

Esquemático RTL

La síntesis del dispositivo se ha realizado con éxito, es un circuito combinacional con 2 entradas de datos, una salida y el selector; las entradas de datos se han generado con el valor por defecto definido en el top del multiplexor (Código 13), 8 bits.

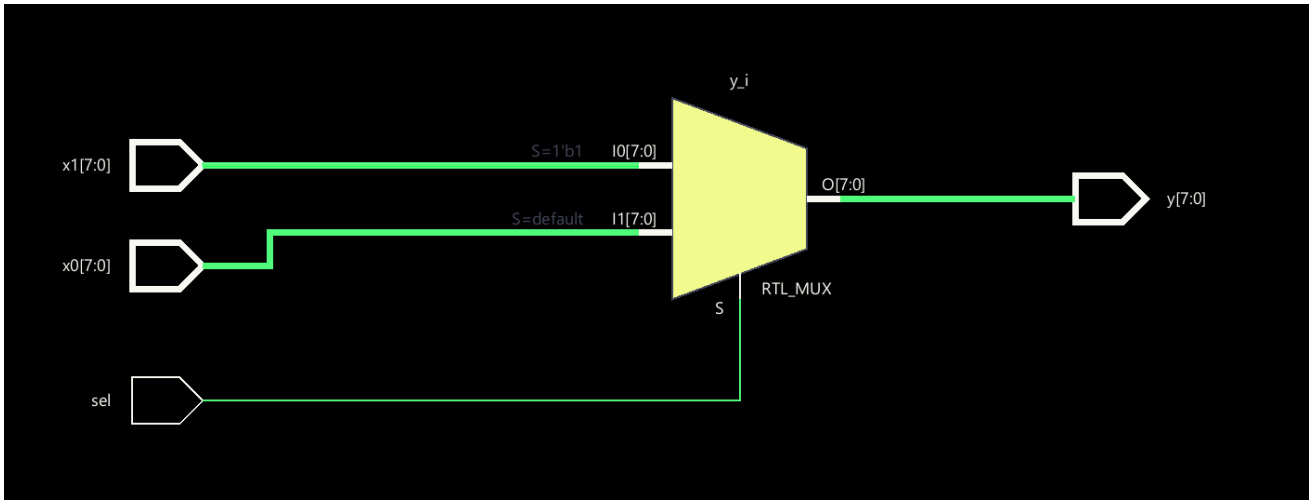

Figura 3: Esquemático RTL del multiplexor 2 a 1

2.2. Sumador

2.2.1. Diseño

Estudio teórico

El sumador es un dispositivo **combinacional** que tiene 2 entradas para los sumandos y una salida para la suma o total.

Estructura

Como se observa en el Código 4 es un dispositivo genérico que depende del parámetro n de tipo *integer*.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Sum is
6    generic(
7      n : integer := 8);
8    port(x : in  std_logic_vector(n-1 downto 0);
9         y : in  std_logic_vector(n-1 downto 0);
10        s : out std_logic_vector(n-1 downto 0));
11 end Sum;
```

Código 4: Entidad del Sumador

Existen distintas maneras de implementar un sumador, lógica a nivel de puerta, instanciando semisumadores y concatenar sus acarreos, etc... en este caso hemos hecho uso de dos paquetes definidos en la librería *ieee* la *std_logic_1164* y *mueric_std*, como se puede observar en el Código 4 estas librerías son las que nos permite simplificar la arquitectura a un simple proceso (línea 3) del Código 5, mediante *casting* cambiamos el tipo de las entradas a *unsigned* lo que nos permite "transformar" una array de bits lógicos en un número con formato binario sin signo, por lo tanto la suma realizada siempre es de 2 números positivos, con otro *casting* a *std_logic_vector* que luego asignamos al puerto de salida.

```
1  architecture Simple of Sum is
2  begin
3    s <= std_logic_vector(unsigned(x) + unsigned(y));
4  end Simple;
```

Código 5: Arquitectura del Sumador

2.2.2. Simulación

Test Bench

Para realizar la simulación del dispositivo hemos usado la plantilla del campus "Test_Bench_Fichero" con una serie de modificaciones, se observa claramente en el Código 6 que se ha instanciado el componente Sum (Código 4), se han declarado una serie de señales internas, algunas genéricas e inicializadas con el valor 'U' (*"uninitialized "*) para trabajar en simulación, y que en el *DUT* (*Device Under Test*) se ha asociado a cada puerto del componente una señal interna. Dado que es un dispositivo genérico también se ha definido una constante de tipo *integer* con un valor reducido para simulación.

En el proceso "estímulos_Desde_Fichero" también hemos modificado el tamaño del input data, ya que solo necesitamos 4 bits para los estímulos (línea 29).

```
1  entity Test_Bench_Fichero is
2  --  Port ( );
3  end Test_Bench_Fichero;
4
5  architecture Comportamiento of Test_Bench_Fichero is
6
7    component Sum
8      generic(
9        n : integer := 8);
10     port(x : in  std_logic_vector(n-1 downto 0);
11          y : in  std_logic_vector(n-1 downto 0);
12          s : out std_logic_vector(n-1 downto 0));
13   end Component Sum;
14
15   constant n: integer := 2;
16   signal x_interno, y_interno, s_interno : std_logic_vector(n-1 downto 0):= (others
     ↪  => 'U');
17
18 begin
19
20   DUT : Sum
21     generic map (n)
22     port map(
23       x => x_interno,
24       y => y_interno,
25       s => s_interno);
26
27 [...]
```

```
28
29      variable Input_Data    : BIT_VECTOR(3 downto 0) := (OTHERS => '0');
```

Código 6: Test Bench del Sumador

Fichero de estímulos

El fichero de estímulos está repartido para las dos entradas, los bit más significativos son la 'X' y los menos significativos la 'Y'. Hemos comprobado todas las sumas posibles.

```
#Fichero de Estímulos de Sumador.
#Device Name: Discovery2NI
#Nombre: Izan Amador, Jorge Benavides
#Fecha: 6 de Diciembre de 2022.
#
# Delay Time (ns) Input (X,Y).

  10ns            0000
  10ns            0001
  10ns            0010
  10ns            0011
  10ns            0100
  10ns            0101
  10ns            0110
  10ns            0111
  10ns            1000
  10ns            1001
  10ns            1010
  10ns            1011
  10ns            1100
  10ns            1101
  10ns            1110
  10ns            1111
```

Cronogramas de simulación

Se observa en la Figura 4 que la operación "suma" se realiza correctamente para todos los caso posibles, como se puede observar hay desbordamiento ya que el dispositivo es muy sencillo, habría que modificar y tal vez realizar un proceso más complejo para evitarlo y/o gestionarlo.

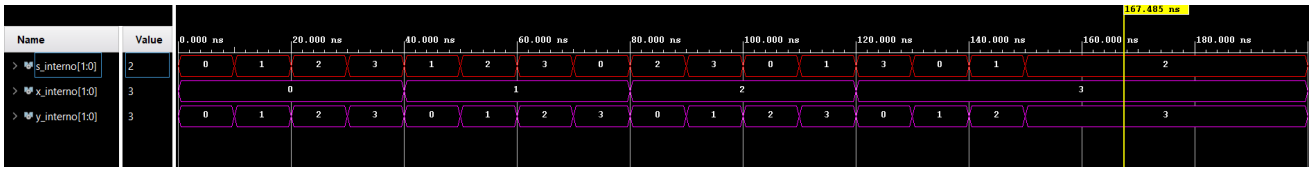

Figura 4: Simulación de sumador

Fichero CSV generado

El fichero de simulación del campus está pensado para una posible verificación con el *Analog Discovery*, la Figura 5 es el CSV generado.

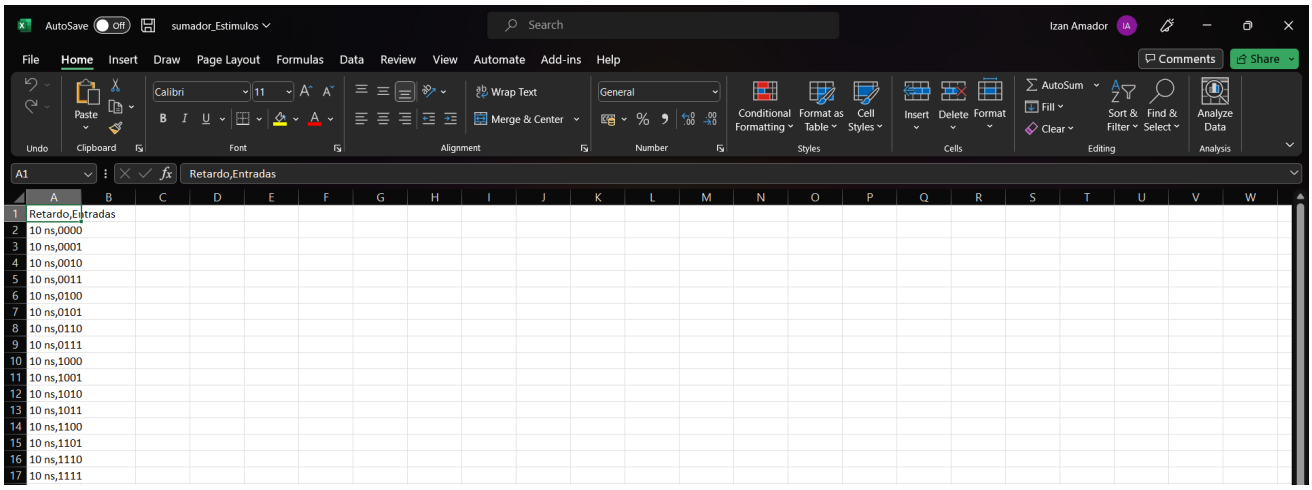

Figura 5: Fichero CSV generado para el Sumador

2.2.3. Síntesis

Esquemático RTL

La síntesis del dispositivo se ha realizado con éxito, es un circuito combinacional con 2 entradas y una salida; las entradas de datos se han generado con el valor por defecto definido en el top del sumador (Código 15), 8 bits.

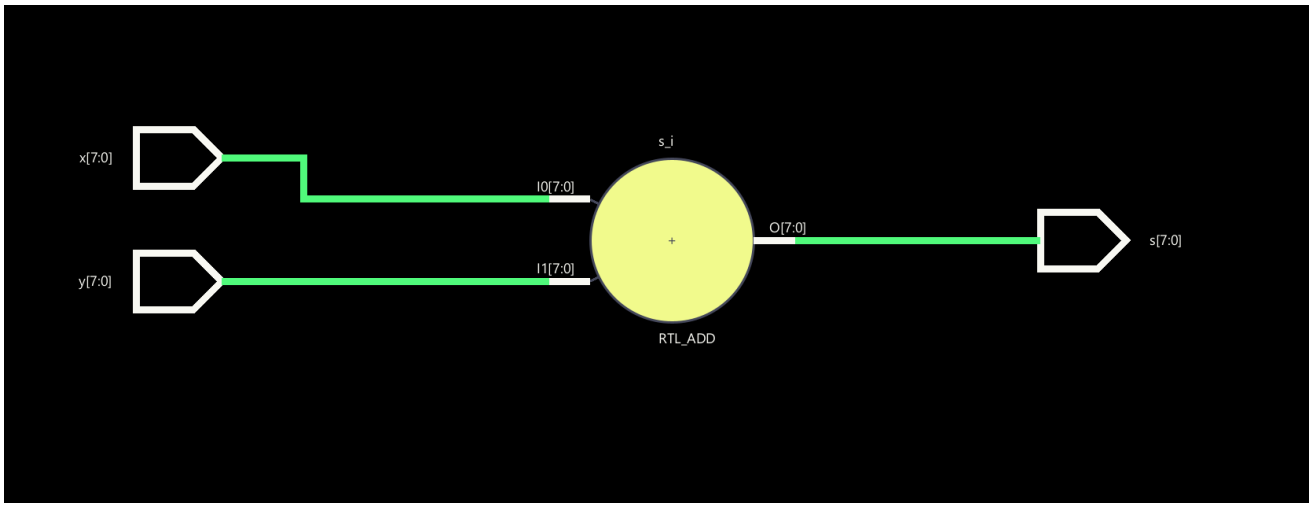

Figura 6: Esquemático RTL de simulación

2.3. Registro de desplazamiento

2.3.1. Diseño

Estudio teórico

El registro de desplazamiento es un dispositivo **secuencial** síncrono con carga en serie, desplaza a la izquierda el valor actual y carga el valor presente en el puerto 'd' en el valor menos significativo.

Estructura

El Código 7 describe un dispositivo genérico, cuenta con 5 puertos, de los cuales 4 de ellos son de entrada, como se ha comentado en la sección anterior es síncrono por lo tanto tiene una señal de reloj *clk*, además cuenta con un *reset* que pone el registro a 0 (realmente 'n' ceros ya que es genérico), una señal *des* que activa o desactiva el desplazamiento y una entrada binaria *d* que carga un elemento en la posición menos significativa.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity Reg_Des is
5    generic(n : integer := 4);
6    port(
7      d          : in  std_logic;
8      q          : out std_logic_vector(n-1 downto 0);
9      reset, des : in  std_logic;
10     clk        : in  std_logic);
11 end Reg_Des;
```

Código 7: Entidad del Registro de desplazamiento

La arquitectura tiene un estilo de máquina de estados ya que cuenta con dos procesos uno combinacional y otro secuencial, el combinacional es una simple asignación de una variable temporal en la salida; el proceso secuencial cuenta con una lista de sensibilidad, *reset* y *clk*, como he comentado antes el *reset* pone todo a '0' y los cambios son síncronos así que se producen con el flanco de subida del reloj, además de la señal de activación *des*.

```
1  architecture A of Reg_Des is
2    signal temp : std_logic_vector(n-1 downto 0);
3  begin
4    process(clk, reset)
5    begin
6      if reset = '1' then
7        temp <= (others => '0');
```

```
8     elsif rising_edge(clk) then
9       if des = '1' then
10        for i in temp'high downto 1 loop
11          -- 'high Atributo para detectar el indice mayor de un array
12          temp(i) <= temp(i-1);
13        end loop;
14        temp(0) <= d;
15      end if;
16    end if;
17  end process;
18  q <= temp;
19 end A;
```

Código 8: Arquitectura del Registro de desplazamiento

2.3.2. Simulación

Test Bench

Para realizar la simulación del dispositivo hemos usado la plantilla del campus "Test_Bench_Fichero" con una serie de modificaciones, se observa claramente en el Código 9 que se ha instanciado el componente Reg_Des (Código 7), se han declarado una serie de señales internas, algunas genéricas (para simplificar) e inicializadas con el valor 'U' ("*uninitialized* ") y otras a '0' para simplificar el trabajo en simulación, y que en el *DUT* (*Device Under Test*) se ha asociado a cada puerto del componente una señal interna. Dado que es un dispositivo genérico también se ha definido una constante de tipo *integer* con un valor reducido para simulación.

En el proceso "estímulos_Desde_Fichero" también hemos modificado el tamaño del input data, ya que solo necesitamos 1 bit el estímulo (línea 29).

Las señales de control, *reset, clk* y *des*, las he definido como un proceso dentro del test bench ya definirlas en el fichero de estímulos no aportaba prácticamente ninguna ventaja. El fichero de estímulos solo contiene el dato de entrada.

Por simplificar el código, la señal *des* está a '1' todo el tiempo de simulación.

```
1  entity Test_Bench_Fichero is
2  --  Port ( );
3  end Test_Bench_Fichero;
4
5  architecture Comportamiento of Test_Bench_Fichero is
6
7    component Reg_Des
8      generic(n : integer := 4);
```

```
9     port(
10      d          : in  std_logic;
11      q          : out std_logic_vector(n-1 downto 0);
12      reset, des : in  std_logic;
13      clk        : in  std_logic);
14  end Component Reg_Des;
15
16  constant semiperiodo : time := 10 ns;
17  constant periodo : time := 2*semiperiodo;
18  constant n: integer := 2;
19  signal q_interno : std_logic_vector(n-1 downto 0):= (others => 'U');
20  signal d_interno, reset_interno, des_interno : std_logic := 'U';
21  signal clk_interno : std_logic := '0';
22
23 begin
24
25  DUT : Reg_Des
26    generic map (n)
27    port map(
28      d => d_interno,
29      q => q_interno,
30      reset => reset_interno,
31      des => des_interno,
32      clk => clk_interno);
33
34  clock_gen: process (clk_interno) is
35  begin
36    if clk_interno = '0' then
37      clk_interno <= '1' after semiperiodo,
38                     '0' after periodo;
39    end if;
40  end process clock_gen;
41
42  des_interno <= '1';
43
44  reset: process
45  begin
46    reset_interno <= '0';
47    wait for 2*periodo;
```

```
48      reset_interno <= '1';
49      wait for 1*periodo;
50      reset_interno <= '0';
51      wait;
52    end process reset;
53
54  [...]
55
56      variable Input_Data    : BIT_VECTOR(0 downto 0) := (OTHERS => '0');
```

Código 9: Test bench del Registro de desplazamiento

Fichero de estímulos

La entrada del fichero de estímulos es la d del dispositivo.

```
#Fichero de Estímulos de Registro de Desplazamiento.
#Device Name: Discovery2NI
#Nombre: Izan Amador, Jorge Benavides
#Fecha: 6 de Diciembre de 2022.
#
# Delay Time (ns) Input (D).

  20 ns 0
  20 ns 1
  20 ns 0
  20 ns 1
  20 ns 0
  20 ns 1
  20 ns 0
  20 ns 1
```

Cronogramas de simulación

Se observa en la Figura 7 que el desplazamiento se realiza correctamente, que en la inicialización los valores de salida son “no inicializados” y que el *reset* pone a cero el registro.

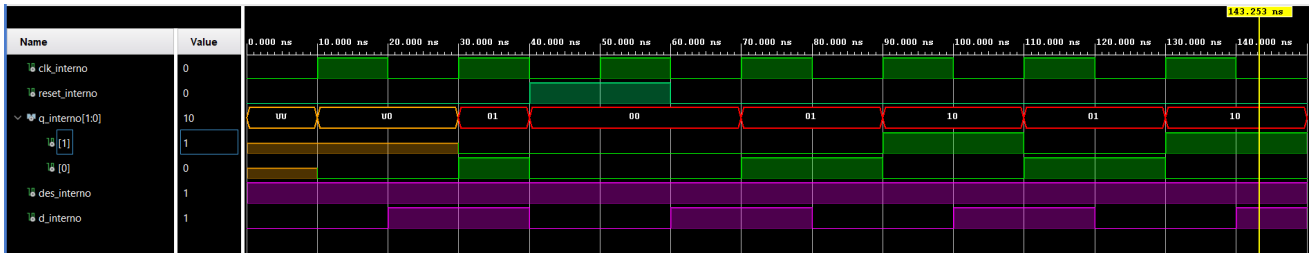

Figura 7: Simulación de registro de desplazamiento

Fichero CSV generado

El fichero de simulación del campus está pensado para una posible verificación con el *Analog Discovery*, la Figura 8 es el CSV generado.

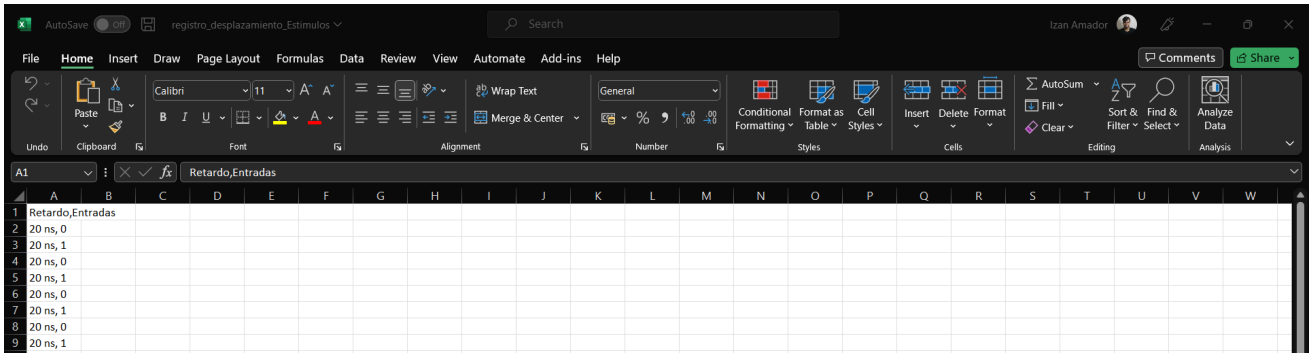

Figura 8: Fichero CSV generado para el registro de desplazamiento

2.3.3. Síntesis

Esquemático RTL

La síntesis del dispositivo se ha realizado con éxito, es un circuito secuencial, que no presenta ningún *latch*, con 4 entradas y una salida; el registro se ha generado con el valor por defecto definido en el top del registro de desplazamiento (Código 17), 4 bits.

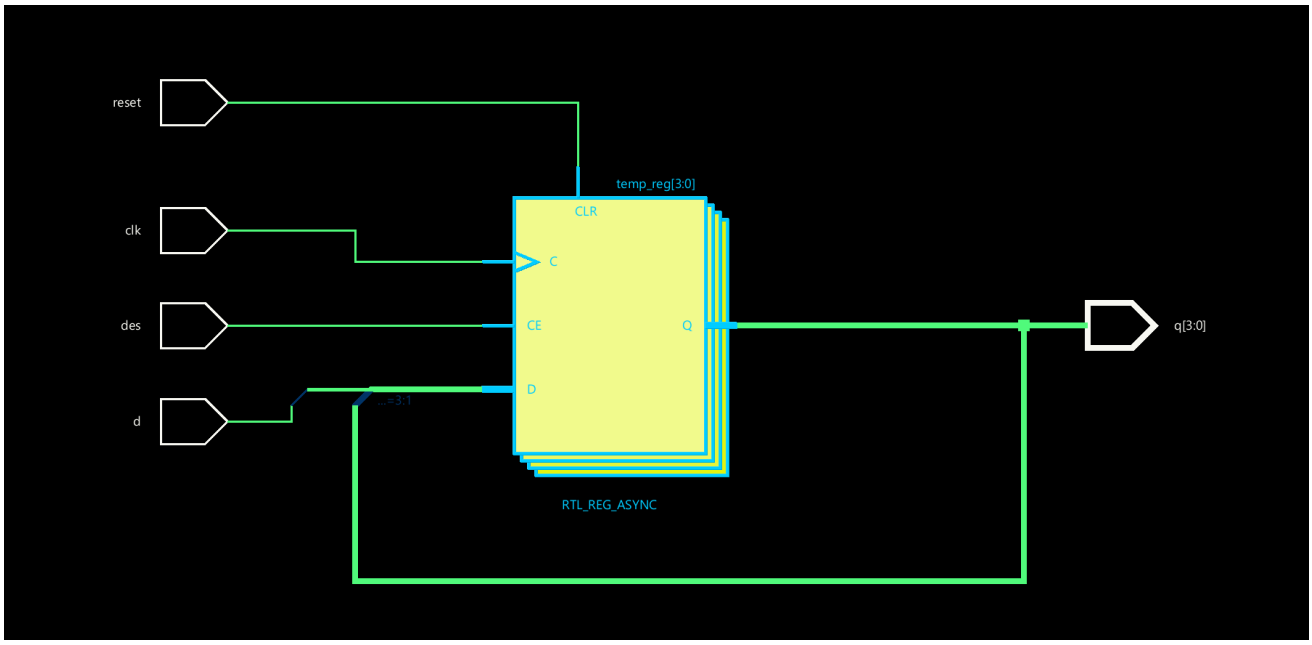

Figura 9: Esquemático RTL del registro de desplazamiento

2.4. Contador ascendente

2.4.1. Diseño

Estudio teórico

El contador es un dispositivo **secuencial** que evoluciona siguiendo una secuencia circular de estados, la transición se produce cuando recibe un pulso en la entrada. Si lo particularizamos para el contador ascendente, conceptualmente, tenemos un generador de secuencias de 0 a N de la forma:

$$0- > 1- > 2- > 3- > 4- > 5- > 6- > 7- > ...- > N$$

Cuando llega al final el final de cuenta muestra una señal de que ha terminado y empieza otra vez.

Estructura

El Código 10 describe un dispositivo genérico, cuenta con dos parámetros genéricos que definen el tamaño del contador y del filtro[1]. Este dispositivo cuenta con 5 puertos de entrada de los cuales solo uno es genérico, el valor que se cargará en el dispositivo cuando se active la señal binaria *load*; dado que es un dispositivo síncrono cuenta con su respectiva señal de reloj, *clk*, señal de *reset* y de activar cuenta, *ce*.

Las salidas muestran el final de cuenta y el valor actual de la cuenta.

```
1  entity top is
2    generic(counter_size : integer := 4;     -- tamano del contador
3            filter_size  : integer := 32);   -- tamano del filtro
4    port(jc_in   : in  std_logic_vector (counter_size-1 downto 0);
5         reset   : in  std_logic;
6         ce      : in  std_logic;
7         load    : in  std_logic;
8         clk     : in  std_logic;
9         jd_out  : out std_logic_vector (counter_size-1 downto 0);
10        fdc     : out std_logic
11        );
12 end entity top;
```

Código 10: Entidad del Contador ascendente

La arquitectura expuesta en el Código 11 es mucho más compleja que en otro circuitos lógicos presentados en esta memoria, ya que este dispositivo ha sido verificado con el *Analog Discovery*, se ha instanciado un "sincronizador" (diseñado en sesiones prácticas anteriores) y el "Contador_Asc" per se. Además, se han definido una serie de señales internas para la interconexión de los dispositivos.

[1] Tamaño del *debouncer* para los botones

```
architecture Behavioral of top is

  component Sincronizador
    generic (m : integer := 1);          -- tamano del filtro
    Port(
      I     : in  std_logic;
      CKE   : out std_logic;
      reset : in  std_logic;
      clk   : in  std_logic);
  end component Sincronizador;

  component Contador_Asc
    generic(n : integer := 8);
    port(din             : in  std_logic_vector(n-1 downto 0);
         dout            : out std_logic_vector(n-1 downto 0);
         reset, ce, load : in  std_logic;
         fdc             : out std_logic;
         clk             : in  std_logic);
  end component Contador_Asc;

  signal reset_interno : std_logic := 'U';
  signal ce_interno    : std_logic := 'U';
  signal load_interno  : std_logic := 'U';

begin

  Sincronizador_reset : Sincronizador generic map(filter_size)
    port map(
      I     => reset,
      CKE   => reset_interno,
      reset => '0',
      clk   => clk
      );

  Sincronizador_ce : Sincronizador generic map(filter_size)
    port map(
      I     => ce,
      CKE   => ce_interno,
      reset => '0',
```

```
40        clk   => clk
41        );
42
43    Sincronizador_load : Sincronizador generic map(filter_size)
44      port map(
45        I     => load,
46        CKE   => load_interno,
47        reset => '0',
48        clk   => clk
49        );
50
51    Contador : Contador_Asc generic map(counter_size)
52      port map(
53        din  => jc_in,
54        reset => reset_interno,
55        ce => ce_interno,
56        load => load_interno,
57        clk  => clk,
58        dout => jd_out,
59        fdc =>  fdc
60        );
61
62  end Behavioral;
```

Código 11: Arquitectura del Contador ascendente

2.4.2. Simulación

Test Bench

Para realizar la simulación del dispositivo hemos usado la plantilla del campus “Test_Bench_Fichero” con una serie de modificaciones, se observa claramente en el Código 12 que se ha instanciado el componente top (Código 10), se han declarado una serie de señales internas, algunas genéricas e inicializadas con el valor ‘U’ (*“uninitialized ”*) y otras a ‘0’ para simplificar el trabajo en simulación, y que en el *DUT* (*Device Under Test*) se ha asociado a cada puerto del componente una señal interna. Dado que es un dispositivo genérico también se ha definido una constante de tipo *integer* con un valor reducido para simulación.

Las entradas o puertos de entrada han sido estimulado mediante procesos, simplificación de fichero de estímulos e independencia del mismo. Un proceso para el *clk*, otro para el *reset*, otro para el *load* el cual no realiza ninguna función ya que está toda la simulación a ‘0’, por lo tanto no carga ningún valor, y otro para el *ce* el cual habilita la cuenta.

Es importante destacar que el tamaño del filtro para simulación es de 4 bits ya que usar un valor muy alto puede entorpecer la simulación y no se notaría la diferencia.

En este fichero también hemos modificado el tamaño del vector de estímulos, en esta ocasión es de 4 bits, así que la respuesta esperada es:

$$0- > 1- > 2- > 3- > 4- > 5- > 6- > 7- > 8- > 9- > a- > b- > c- > d- > e- > f- > 0- > ...$$

Aunque parezca un testbench un poco escueto, es perfecto para la síntesis y el despliegue en la plaza Zybo ya que el fichero de estímulos se transformará en un .csv y los botones simulados pasarán a ser reales lo que hace una simulación más fructífera.

```
entity Test_Bench is
--  Port ( );
end Test_Bench;

architecture Comportamiento of Test_Bench is

  component top
    generic(counter_size : integer := 4;     -- tamano del contador
            filter_size  : integer := 32);  -- tamano del filtro
    port(jc_in  : in  std_logic_vector (counter_size-1 downto 0);
         reset  : in  std_logic;
         ce     : in  std_logic;
         load   : in  std_logic;
         clk    : in  std_logic;
         jd_out : out std_logic_vector (counter_size-1 downto 0);
         fdc    : out std_logic
         );
  end Component top;

  constant semiperiodo : time := 10 ns;
  constant periodo : time := 2*semiperiodo;
  constant counter_size: integer := 4;
  constant filter_size : integer := 4;
  signal jc_in_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
 ↪  'U');
  signal reset_interno, fdc_interno : std_logic := 'U';
  signal clk_interno, ce_interno, load_interno : std_logic := '0';
  signal jd_out_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
 ↪  'U');
begin
```

```
29
30    DUT : top
31    generic map (counter_size, filter_size)
32      port map(
33        jc_in  => jc_in_interno,
34        reset  => reset_interno,
35        ce     => ce_interno,
36        load   => load_interno,
37        clk    => clk_interno,
38        jd_out => jd_out_interno,
39        fdc    => fdc_interno);
40
41  clock_gen: process (clk_interno) is
42    begin
43      if clk_interno = '0' then
44        clk_interno <= '1' after semiperiodo,
45                       '0' after periodo;
46      end if;
47    end process clock_gen;
48
49    reset: process
50    begin
51      reset_interno <= '0';
52      wait for 2*periodo;
53      reset_interno <= '1';
54      wait for 1*periodo;
55      reset_interno <= '0';
56      wait;
57    end process reset;
58
59
60    load: process (load_interno) is
61      begin
62        if load_interno = '0' then
63          load_interno <= '1' after 200*periodo,
64                          '0' after 201*periodo;
65        end if;
66      end process load;
67
```

```
68
69    ce: process (ce_interno) is
70      begin
71        if ce_interno = '0' then
72          ce_interno <= '1' after 2*periodo,
73                        '0' after 4*periodo;
74        end if;
75      end process ce;
76
77  [...]
78
79      variable Input_Data    : BIT_VECTOR(3 downto 0) := (OTHERS => '0');
```

Código 12: Testbench del Contador ascendente

Fichero de estímulos

El fichero de estímulos son números en binario del de '0' al '15' con distintos tiempos de retraso„ que se cargarán aleatoriamente cuando el usuario (en este caso nosotros) pulse el botón *load*.

```
#Fichero de Estímulos de Contador_Asc.
#Device Name: Discovery2NI
#Nombre: Izan Amador, Jorge Benavides
#Fecha: 30 de Noviembre de 2022.
#
# Delay Time (ns) Input (din(3:0)).

    1310 ns 0000
    1310 ns 0001
    1310 ns 0010
    1310 ns 0011
    1310 ns 0100
    10.5 ns 0101
    2.5  ns 0110
    3.5  ns 0111
    4.5  ns 1000
    5.5  ns 1001
    6.5  ns 1010
    7.5  ns 1011
    8.5  ns 1100
    1.5  ns 1101
    10.5 ns 1110
    10.5 ns 1111
```

Cronogramas de simulación

El cronograma generado con el test bench y el fichero de estímulos es bastante sencillo, la señal de *reset* inicializa el sistema y empieza a contar desde 0 hasta que llega al final y el final de cuenta (*fdc*) envía un pulso.

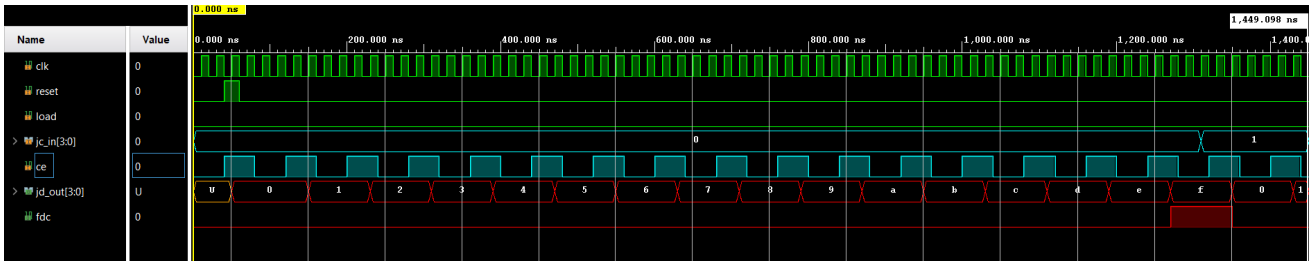

Figura 10: Simulación de contador ascendente

Fichero CSV generado

Se ha genderdo un CSV que ha diferencia de los dispositivos anteriores tiene una importancia ya que es el fichero que usaremos en el *Analog Discovery* para estimular nuestro diseño en la placa Zybo.

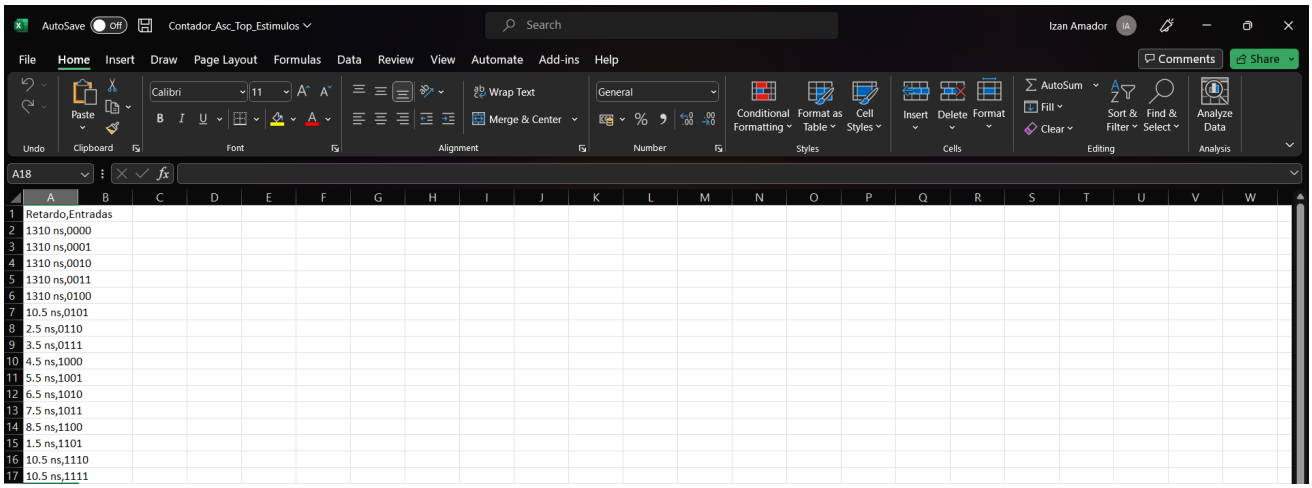

Figura 11: Fichero CSV generado para el contador ascendente

2.4.3. Síntesis

Esquemático RTL

La síntesis del dispositivo se ha realizado con éxito, es un circuito secuencial, que no presenta ningún *latch*, con 5 entradas y dos salidas; como hemos comentado en secciones anteriores cuenta con 3 sincronizadores y el contador (Código 19), 4 bits.

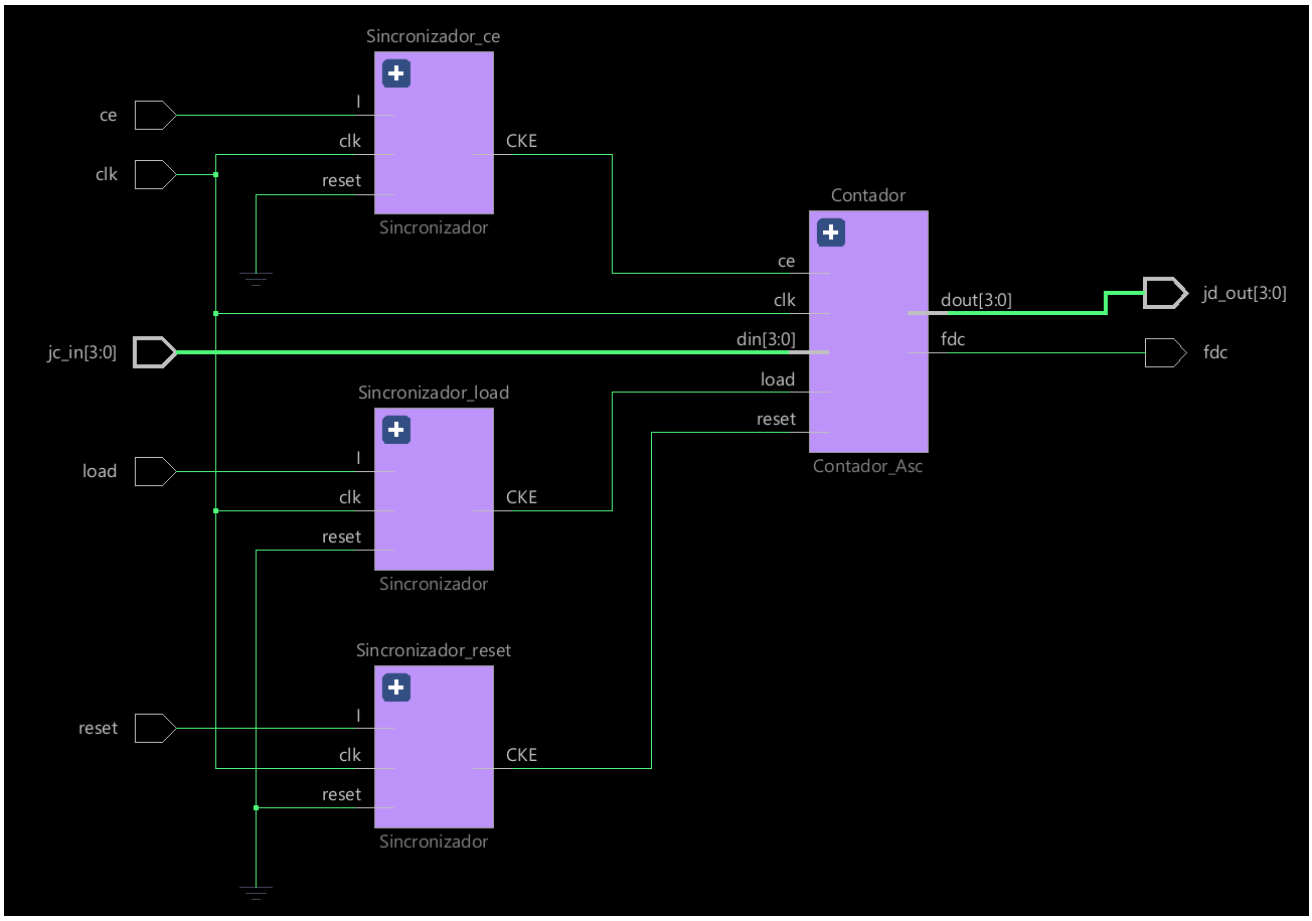

Figura 12: Esquemático RTL de contador ascendente

2.4.4. Verificación con Analog Discovery

El siguiente paso en el proceso de diseño, simulación y síntesis es la verificación en hardware como se muestra en la Figura 13, hemos modificado el *.xdc* de la Zybo para habilitar los puertos pmod #4 y #5 para realizar las conexiones.

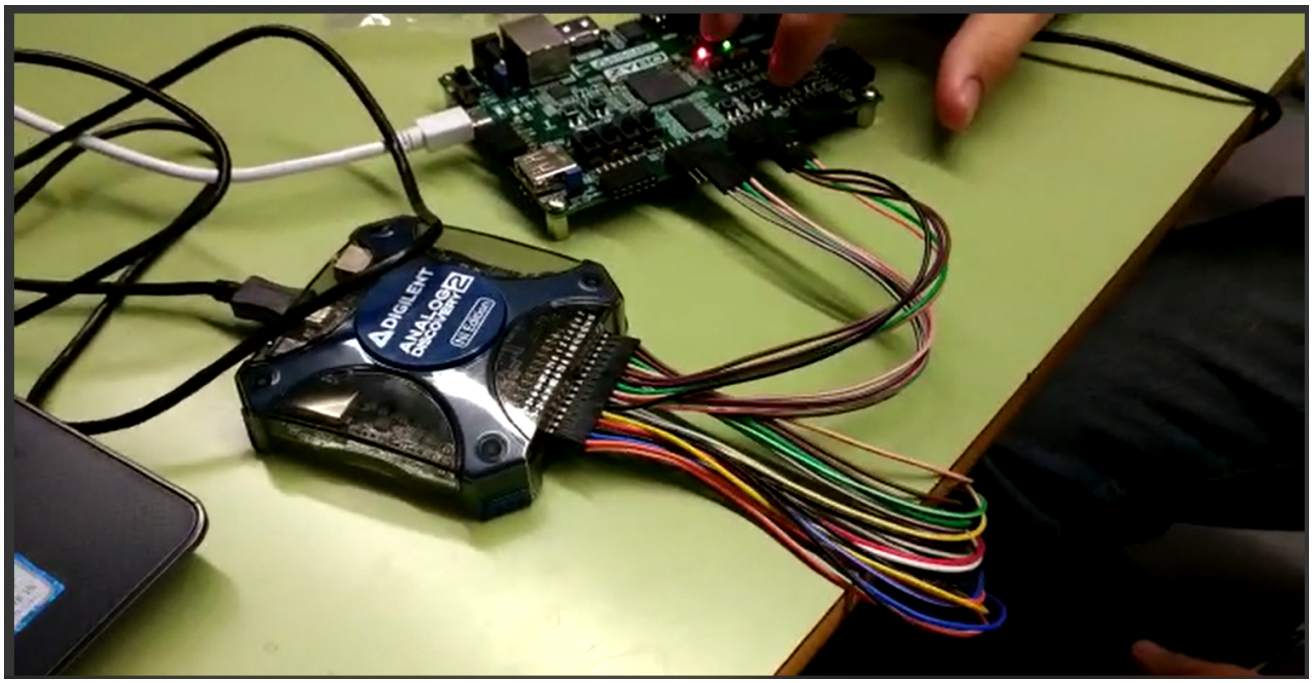

Figura 13: Conexionado del Analog Discovery con la Zybo

Resultado de la verificación

La verificación en hardware al igual que los procesos anteriores ha sido todo un éxito y es mucho más completa porque podemos activar la cuenta a placer, ya que la activación del contador la gestionamos nosotros con un botón, no hay ningún tipo de rebote y

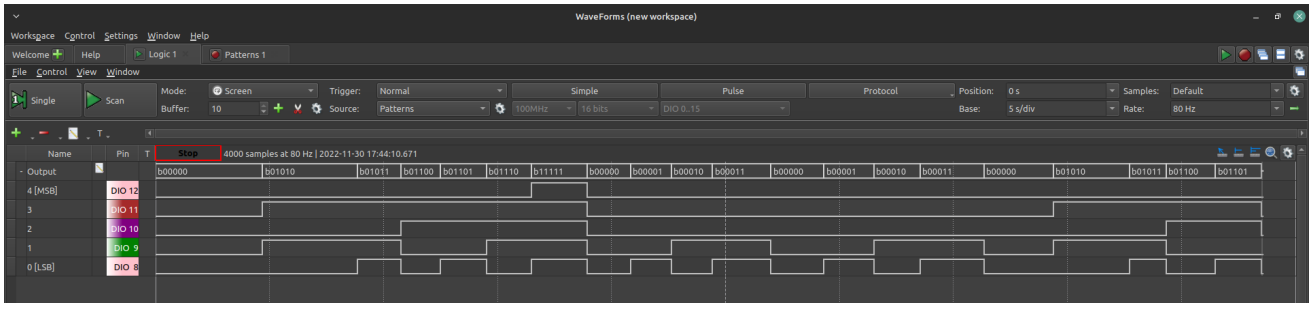

Figura 14: Verificación de contador ascendente en Waveforms

3. Anexos

3.1. Multiplexor

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

entity MuxV_2a1 is
  generic(
    n : integer := 8);
  port(
    x0  : in  std_logic_vector(n-1 downto 0);
    x1  : in  std_logic_vector(n-1 downto 0);
    sel : in  std_logic;
    y   : out std_logic_vector(n-1 downto 0));
end MuxV_2a1;

architecture Behavioral of MuxV_2a1 is
begin
  process(sel, x0, x1)
  begin
    if sel = '1' then
      y <= x1;
    else
      y <= x0;
    end if;
  end process;
end Behavioral;
```

Código 13: Top del multiplexor

```
----------------------------------------------------------------------------------
-- Company: Universidad de Málaga
-- Engineer: Izan Amador, Jorge L. Benavides
--
-- Create Date: 23.11.2022 17:47:41
-- Design Name: registro_desplazamiento
-- Module Name: Test_Bench_Fichero- Behavioral
-- Project Name: mux2a1
```

```
9  -- Target Devices: Zybo
10 -- Tool Versions: Vivado 2022.1
11 -- Description: basic multiplexer block
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 ----------------------------------------------------------------------------------
19
20 -- Uncomment the following library declaration if instantiating
21 -- any Xilinx leaf cells in this code.
22 --library UNISIM;
23 --use UNISIM.VComponents.all;
24
25 entity Test_Bench_Fichero is
26 --  Port ( );
27 end Test_Bench_Fichero;
28
29 architecture Comportamiento of Test_Bench_Fichero is
30
31   component MuxV_2a1
32     generic(
33       n : integer := 8);
34     port(
35       x0  : in  std_logic_vector(n-1 downto 0);
36       x1  : in  std_logic_vector(n-1 downto 0);
37       sel : in  std_logic;
38       y   : out std_logic_vector(n-1 downto 0));
39   end Component MuxV_2a1;
40
41   constant n: integer := 1;
42   signal x0_interno, x1_interno, y_interno : std_logic_vector(n-1 downto 0):=
     ↪  (others => 'U');
43   signal sel_interno : std_logic := 'U';
44
45 begin
46
```

```
47    DUT : MuxV_2a1
48      generic map (n)
49      port map(
50        x0 => x0_interno,
51        x1 => x1_interno,
52        sel => sel_interno,
53        y => y_interno);
54
55    Estimulos_Desde_Fichero : process
56
57      file Input_File  : text;
58      file Output_File : text;
59
60      variable Input_Data   : BIT_VECTOR(2 downto 0) := (OTHERS => '0');
61      variable Delay        : time                   := 0 ms;
62      variable Input_Line   : line                   := NULL;
63      variable Output_Line  : line                   := NULL;
64      variable Std_Out_Line : line                   := NULL;
65      variable Correcto     : Boolean                := True;
66      constant Coma         : character              := ',';
67
68
69    begin
70
71  -- multiplexor2a1_Estimulos.txt contiene los estÃmulos y los tiempos de retardo para
     ↪  el semisumador.
72      file_open(Input_File,
     ↪  "C:\Users\izana\Documents\GitHub\SEA\Estimulos\multiplexor2a1_Estimulos.txt",
     ↪  read_mode);
73  -- multiplexor2a1_Estimulos.csv contiene los estÃmulos y los tiempos de retardo para
     ↪  el Analog Discovery 2.
74      file_open(Output_File,
     ↪  "C:\Users\izana\Documents\GitHub\SEA\CSV\multiplexor2a1_Estimulos.csv",
     ↪  write_mode);
75
76  -- Titles: Son para el formato EXCEL *.CSV (Comma Separated Values):
77      write(Std_Out_Line, string'("Retardo"), right, 7);
78      write(Std_Out_Line, Coma, right, 1);
79      write(Std_Out_Line, string'("Entradas"), right, 8);
```

```
80
81    Output_Line := Std_Out_Line;
82
83    writeline(output, Std_Out_Line);
84    writeline(Output_File, Output_Line);
85
86    while (not endfile(Input_File)) loop
87
88      readline(Input_File, Input_Line);
89
90      read(Input_Line, Delay, Correcto);  -- ComprobaciÃ³n de que se trata de un
    ↪ texto que representa
91      -- el retardo, si no es asÃ leemos la siguiente lÃnea.
92      if Correcto then
93
94        read(Input_Line, Input_Data);  -- El siguiente campo es el vector de
    ↪ pruebas.
95        -- Der a Izq
96
97        x0_interno <= TO_STDLOGICVECTOR(Input_Data)(2 downto 2);
98        x1_interno <= TO_STDLOGICVECTOR(Input_Data)(1 downto 1);
99        sel_interno <= TO_STDLOGICVECTOR(Input_Data)(0);
100
101       -- De forma simultÃ¡nea lo volcaremos en consola en csv.
102       write(Std_Out_Line, Delay, right, 5);  -- Longitud del retardo, ej. "20 ms".
103       write(Std_Out_Line, Coma, right, 1);
104       write(Std_Out_Line, Input_Data, right, 2);  --Longitud de los datos de
    ↪ entrada.
105
106       Output_Line := Std_Out_Line;
107
108       writeline(output, Std_Out_Line);
109       writeline(Output_File, Output_Line);
110
111       wait for Delay;
112     end if;
113   end loop;
114
115   file_close(Input_File);             -- Cerramos el fichero de entrada.
```

```
116     file_close(Output_File);            -- Cerramos el fichero de salida.
117     wait;
118   end process Estimulos_Desde_Fichero;
119
120
121 end Comportamiento;
```

Código 14: TestBench del Mux2a1

3.2. Sumador

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Sum is
6    generic(
7      n : integer := 8);
8    port(x : in  std_logic_vector(n-1 downto 0);
9         y : in  std_logic_vector(n-1 downto 0);
10        s : out std_logic_vector(n-1 downto 0));
11 end Sum;
12
13 architecture Simple of Sum is
14 begin
15   s <= std_logic_vector(unsigned(x) + unsigned(y));
16 end Simple;
```

Código 15: Top del sumador

```
1  ----------------------------------------------------------------------------------
2  -- Company: Universidad de Málaga
3  -- Engineer: Izan Amador, Jorge L. Benavides
4  --
5  -- Create Date: 23.11.2022 17:47:41
6  -- Design Name: sumador
7  -- Module Name: Test_Bench_Fichero - Behavioral
8  -- Project Name: sumador
```

```
-- Target Devices: Zybo
-- Tool Versions: Vivado 2022.1
-- Description: basic test bench for a simple adder.
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use STD.textIO.ALL;                     -- Se va a hacer uso de ficheros.

-- 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 Test_Bench_Fichero is
--  Port ( );
end Test_Bench_Fichero;

architecture Comportamiento of Test_Bench_Fichero is

  component Sum
    generic(
      n : integer := 8);
    port(x : in  std_logic_vector(n-1 downto 0);
         y : in  std_logic_vector(n-1 downto 0);
         s : out std_logic_vector(n-1 downto 0));
```

```
48    end Component Sum;
49
50    constant n: integer := 2;
51    signal x_interno, y_interno, s_interno : std_logic_vector(n-1 downto 0):= (others
      ↪  => 'U');
52
53  begin
54
55    DUT : Sum
56      generic map (n)
57      port map(
58        x => x_interno,
59        y => y_interno,
60        s => s_interno);
61
62    Estimulos_Desde_Fichero : process
63
64      file Input_File  : text;
65      file Output_File : text;
66
67      variable Input_Data   : BIT_VECTOR(3 downto 0) := (OTHERS => '0');
68      variable Delay        : time                   := 0 ms;
69      variable Input_Line   : line                   := NULL;
70      variable Output_Line  : line                   := NULL;
71      variable Std_Out_Line : line                   := NULL;
72      variable Correcto     : Boolean                := True;
73      constant Coma         : character              := ',';
74
75
76    begin
77
78  -- sumador_Estimulos.txt contiene los estÃmulos y los tiempos de retardo para el
      ↪  semisumador.
79      file_open(Input_File,
      ↪  "C:\Users\izana\Documents\GitHub\SEA\Estimulos\sumador_Estimulos.txt",
      ↪  read_mode);
80
81  -- sumador_Estimulos.csv contiene los estÃmulos y los tiempos de retardo para el
      ↪  Analog Discovery 2.
```

```
82      file_open(Output_File,
    ↪  "C:\Users\izana\Documents\GitHub\SEA\CSV\sumador_Estimulos.csv", write_mode);
83
84  -- Titles: Son para el formato EXCEL *.CSV (Comma Separated Values):
85      write(Std_Out_Line, string'("Retardo"), right, 7);
86      write(Std_Out_Line, Coma, right, 1);
87      write(Std_Out_Line, string'("Entradas"), right, 8);
88
89      Output_Line := Std_Out_Line;
90
91      writeline(output, Std_Out_Line);
92      writeline(Output_File, Output_Line);
93
94      while (not endfile(Input_File)) loop
95
96        readline(Input_File, Input_Line);
97
98        read(Input_Line, Delay, Correcto);  -- ComprobaciÃ³n de que se trata de un
    ↪  texto que representa
99        -- el retardo, si no es asÃ leemos la siguiente lÃnea.
100       if Correcto then
101
102         read(Input_Line, Input_Data);  -- El siguiente campo es el vector de
    ↪  pruebas.
103         -- Der a Izq
104
105         x_interno <= TO_STDLOGICVECTOR(Input_Data)(3 downto 2);
106         y_interno <= TO_STDLOGICVECTOR(Input_Data)(1 downto 0);
107
108         -- De forma simultÃ¡nea lo volcaremos en consola en csv.
109         write(Std_Out_Line, Delay, right, 5);  -- Longitud del retardo, ej. "20 ms".
110         write(Std_Out_Line, Coma, right, 1);
111         write(Std_Out_Line, Input_Data, right, 2);  --Longitud de los datos de
    ↪  entrada.
112
113         Output_Line := Std_Out_Line;
114
115         writeline(output, Std_Out_Line);
116         writeline(Output_File, Output_Line);
```

```
117
118         wait for Delay;
119       end if;
120     end loop;
121
122     file_close(Input_File);              -- Cerramos el fichero de entrada.
123     file_close(Output_File);             -- Cerramos el fichero de salida.
124     wait;
125   end process Estimulos_Desde_Fichero;
126
127
128 end Comportamiento;
```

Código 16: TestBench del sumador

3.3. Registro de desplazamiento

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity Reg_Des is
5    generic(n : integer := 4);
6    port(
7      d          : in  std_logic;
8      q          : out std_logic_vector(n-1 downto 0);
9      reset, des : in  std_logic;
10     clk        : in  std_logic);
11 end Reg_Des;
12
13 architecture A of Reg_Des is
14   signal temp : std_logic_vector(n-1 downto 0);
15 begin
16   process(clk, reset)
17   begin
18     if reset = '1' then
19       temp <= (others => '0');
20     elsif rising_edge(clk) then
21       if des = '1' then
```

```
22          for i in temp'high downto 1 loop
23            -- 'high Atributo para detectar el indice mayor de un array
24            temp(i) <= temp(i-1);
25          end loop;
26          temp(0) <= d;
27        end if;
28      end if;
29    end process;
30    q <= temp;
31  end A;
```

Código 17: Top del registro de desplazamiento

```
1   ----------------------------------------------------------------------------------
2   -- Company: Universidad de Málaga
3   -- Engineer: Izan Amador, Jorge L. Benavides
4   --
5   -- Create Date: 23.11.2022 17:47:41
6   -- Design Name: registro_desplazamiento
7   -- Module Name: Test_Bench_Fichero- Behavioral
8   -- Project Name: registro_desplazamiento
9   -- Target Devices: Zybo
10  -- Tool Versions: Vivado 2022.1
11  -- Description: basic shift register block.
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  -- Basic shift register
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use STD.textIO.ALL;                     -- Se va a hacer uso de ficheros.
25
26  -- Uncomment the following library declaration if using
```

```
27 -- arithmetic functions with Signed or Unsigned values
28 --use IEEE.NUMERIC_STD.ALL;
29
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34
35 entity Test_Bench_Fichero is
36 --  Port ( );
37 end Test_Bench_Fichero;
38
39 architecture Comportamiento of Test_Bench_Fichero is
40
41   component Reg_Des
42     generic(n : integer := 4);
43     port(
44       d          : in  std_logic;
45       q          : out std_logic_vector(n-1 downto 0);
46       reset, des : in  std_logic;
47       clk        : in  std_logic);
48   end Component Reg_Des;
49
50   constant semiperiodo : time := 10 ns;
51   constant periodo : time := 2*semiperiodo;
52   constant n: integer := 2;
53   signal q_interno : std_logic_vector(n-1 downto 0):= (others => 'U');
54   signal d_interno, reset_interno, des_interno : std_logic := 'U';
55   signal clk_interno : std_logic := '0';
56
57 begin
58
59   DUT : Reg_Des
60     generic map (n)
61     port map(
62       d => d_interno,
63       q => q_interno,
64       reset => reset_interno,
65       des => des_interno,
```

```
66      clk => clk_interno);
67
68    clock_gen: process (clk_interno) is
69    begin
70      if clk_interno = '0' then
71        clk_interno <= '1' after semiperiodo,
72                       '0' after periodo;
73      end if;
74    end process clock_gen;
75
76    des_interno <= '1';
77
78    reset: process
79    begin
80      reset_interno <= '0';
81      wait for 2*periodo;
82      reset_interno <= '1';
83      wait for 1*periodo;
84      reset_interno <= '0';
85      wait;
86    end process reset;
87
88    Estimulos_Desde_Fichero : process
89
90      file Input_File  : text;
91      file Output_File : text;
92
93      variable Input_Data   : BIT_VECTOR(0 downto 0) := (OTHERS => '0');
94      variable Delay        : time                   := 0 ms;
95      variable Input_Line   : line                   := NULL;
96      variable Output_Line  : line                   := NULL;
97      variable Std_Out_Line : line                   := NULL;
98      variable Correcto     : Boolean                := True;
99      constant Coma         : character              := ',';
100
101
102   begin
103
104 -- registro_desplazamiento_Estimulos.txt contiene los estÃmulos y los tiempos de
    ↪  retardo para el semisumador.
```

```
105     file_open(Input_File,
        ↪ "C:\Users\izana\Documents\GitHub\SEA\Estimulos\registro_desplazamiento_Estimulos.txt",
        ↪ read_mode);
106
107 -- registro_desplazamiento.csv contiene los estÃmulos y los tiempos de retardo para
        ↪ el Analog Discovery 2.
108     file_open(Output_File,
        ↪ "C:\Users\izana\Documents\GitHub\SEA\CSV\registro_desplazamiento_Estimulos.csv",
        ↪ write_mode);
109
110 -- Titles: Son para el formato EXCEL *.CSV (Comma Separated Values):
111     write(Std_Out_Line, string'("Retardo"), right, 7);
112     write(Std_Out_Line, Coma, right, 1);
113     write(Std_Out_Line, string'("Entradas"), right, 8);
114
115     Output_Line := Std_Out_Line;
116
117     writeline(output, Std_Out_Line);
118     writeline(Output_File, Output_Line);
119
120     while (not endfile(Input_File)) loop
121
122       readline(Input_File, Input_Line);
123
124       read(Input_Line, Delay, Correcto);  -- ComprobaciÃ³n de que se trata de un
        ↪ texto que representa
125       -- el retardo, si no es asÃ leemos la siguiente lÃnea.
126       if Correcto then
127
128         read(Input_Line, Input_Data);  -- El siguiente campo es el vector de
        ↪ pruebas.
129         -- Der a Izq
130
131         d_interno <= TO_STDLOGICVECTOR(Input_Data)(0);
132
133         -- De forma simultÃ¡nea lo volcaremos en consola en csv.
134         write(Std_Out_Line, Delay, right, 5);  -- Longitud del retardo, ej. "20 ms".
135         write(Std_Out_Line, Coma, right, 1);
136         write(Std_Out_Line, Input_Data, right, 2);  --Longitud de los datos de
        ↪ entrada.
```

```
137
138         Output_Line := Std_Out_Line;
139
140         writeline(output, Std_Out_Line);
141         writeline(Output_File, Output_Line);
142
143         wait for Delay;
144       end if;
145     end loop;
146
147     file_close(Input_File);              -- Cerramos el fichero de entrada.
148     file_close(Output_File);             -- Cerramos el fichero de salida.
149     wait;
150   end process Estimulos_Desde_Fichero;
151
152
153 end Comportamiento;
```

Código 18: TestBench del registro de desplazamiento

3.4. Contador ascendente

```
1  entity top is
2    generic(counter_size : integer := 4;     -- tamano del contador
3            filter_size  : integer := 32);   -- tamano del filtro
4    port(jc_in   : in  std_logic_vector (counter_size-1 downto 0);
5         reset   : in  std_logic;
6         ce      : in  std_logic;
7         load    : in  std_logic;
8         clk     : in  std_logic;
9         jd_out  : out std_logic_vector (counter_size-1 downto 0);
10        fdc     : out std_logic
11        );
12 end entity top;
13
14 architecture Behavioral of top is
15
16   component Sincronizador
```

```
17      generic (m : integer := 1);          -- tamano del filtro
18      Port(
19        I     : in  std_logic;
20        CKE   : out std_logic;
21        reset : in  std_logic;
22        clk   : in  std_logic);
23    end component Sincronizador;
24
25    component Contador_Asc
26      generic(n : integer := 8);
27      port(din             : in  std_logic_vector(n-1 downto 0);
28           dout            : out std_logic_vector(n-1 downto 0);
29           reset, ce, load : in  std_logic;
30           fdc             : out std_logic;
31           clk             : in  std_logic);
32    end component Contador_Asc;
33
34    signal reset_interno : std_logic := 'U';
35    signal ce_interno    : std_logic := 'U';
36    signal load_interno  : std_logic := 'U';
37
38  begin
39
40    Sincronizador_reset : Sincronizador generic map(filter_size)
41      port map(
42        I     => reset,
43        CKE   => reset_interno,
44        reset => '0',
45        clk   => clk
46        );
47    Sincronizador_ce : Sincronizador generic map(filter_size)
48      port map(
49        I     => ce,
50        CKE   => ce_interno,
51        reset => '0',
52        clk   => clk
53        );
54    Sincronizador_load : Sincronizador generic map(filter_size)
55      port map(
```

```
      I     => load,
      CKE   => load_interno,
      reset => '0',
      clk   => clk
      );

  Contador : Contador_Asc generic map(counter_size)
    port map(
      din  => jc_in,
      reset => reset_interno,
      ce => ce_interno,
      load => load_interno,
      clk  => clk,
      dout => jd_out,
      fdc =>  fdc
      );

end Behavioral;
```

Código 19: Top del contador ascendente

```
----------------------------------------------------------------------------------
-- Company: Universidad de Málaga
-- Engineer: Izan Amador, Jorge L. Benavides
--
-- Create Date: 23.11.2022 17:47:41
-- Design Name: Contador Ascendente
-- Module Name: Test_Bench_top - Behavioral
-- Project Name: contador_ascendente
-- Target Devices: Zybo
-- Tool Versions: Vivado 2022.1
-- Description: basic upwards counter
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- Configured for verification with Analog Discovery Module
```

```
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use STD.textIO.ALL;                    -- Se va a hacer uso de ficheros.
25
26 -- Uncomment the following library declaration if using
27 -- arithmetic functions with Signed or Unsigned values
28 --use IEEE.NUMERIC_STD.ALL;
29
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34
35 entity Test_Bench is
36 --  Port ( );
37 end Test_Bench;
38
39 architecture Comportamiento of Test_Bench is
40
41   component top
42     generic(counter_size : integer := 4;    -- tamano del contador
43             filter_size  : integer := 32);  -- tamano del filtro
44     port(jc_in  : in  std_logic_vector (counter_size-1 downto 0);
45          reset  : in  std_logic;
46          ce     : in  std_logic;
47          load   : in  std_logic;
48          clk    : in  std_logic;
49          jd_out : out std_logic_vector (counter_size-1 downto 0);
50          fdc    : out std_logic
51          );
52   end Component top;
53
54   constant semiperiodo : time := 10 ns;
55   constant periodo : time := 2*semiperiodo;
56   constant counter_size: integer := 4;
57   constant filter_size : integer := 4;
```

```
58    signal jc_in_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
      ↪  'U');
59    signal reset_interno, fdc_interno : std_logic := 'U';
60    signal clk_interno, ce_interno, load_interno : std_logic := '0';
61    signal jd_out_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
      ↪  'U');
62  begin
63
64    DUT : top
65    generic map (counter_size, filter_size)
66      port map(
67        jc_in  => jc_in_interno,
68        reset  => reset_interno,
69        ce     => ce_interno,
70        load   => load_interno,
71        clk    => clk_interno,
72        jd_out => jd_out_interno,
73        fdc    => fdc_interno);
74
75    clock_gen: process (clk_interno) is
76    begin
77      if clk_interno = '0' then
78        clk_interno <= '1' after semiperiodo,
79                       '0' after periodo;
80      end if;
81    end process clock_gen;
82
83    reset: process
84    begin
85      reset_interno <= '0';
86      wait for 2*periodo;
87      reset_interno <= '1';
88      wait for 1*periodo;
89      reset_interno <= '0';
90      wait;
91    end process reset;
92
93
94    load: process (load_interno) is
```

```
95      begin
96        if load_interno = '0' then
97          load_interno <= '1' after 200*periodo,
98                          '0' after 201*periodo;
99        end if;
100     end process load;
101
102
103   ce: process (ce_interno) is
104     begin
105       if ce_interno = '0' then
106         ce_interno <= '1' after 2*periodo,
107                       '0' after 4*periodo;
108       end if;
109     end process ce;
110
111
112   Estimulos_Desde_Fichero : process
113
114     file Input_File  : text;
115     file Output_File : text;
116
117     variable Input_Data   : BIT_VECTOR(3 downto 0) := (OTHERS => '0');
118     variable Delay        : time                   := 0 ms;
119     variable Input_Line   : line                   := NULL;
120     variable Output_Line  : line                   := NULL;
121     variable Std_Out_Line : line                   := NULL;
122     variable Correcto     : Boolean                := True;
123     constant Coma         : character              := ',';
124
125
126   begin
127
128 -- Contador_Asc_Top_Estimulos.txt contiene los estÃmulos y los tiempos de retardo
    ↪  para el semisumador.
129     file_open(Input_File,
    ↪  "C:\Users\izana\Documents\GitHub\SEA\Estimulos\Contador_Asc_Top_Estimulos.txt",
    ↪  read_mode);
130 -- Contador_Asc_Top_Estimulos.csv contiene los estÃmulos y los tiempos de retardo
    ↪  para el Analog Discovery 2.
```

```
131     file_open(Output_File,
    ↪  "C:\Users\izana\Documents\GitHub\SEA\CSV\Contador_Asc_Top_Estimulos.csv",
    ↪  write_mode);
132
133 -- Titles: Son para el formato EXCEL *.CSV (Comma Separated Values):
134     write(Std_Out_Line, string'("Retardo"), right, 7);
135     write(Std_Out_Line, Coma, right, 1);
136     write(Std_Out_Line, string'("Entradas"), right, 8);
137
138     Output_Line := Std_Out_Line;
139
140     writeline(output, Std_Out_Line);
141     writeline(Output_File, Output_Line);
142
143     while (not endfile(Input_File)) loop
144
145       readline(Input_File, Input_Line);
146
147       read(Input_Line, Delay, Correcto);  -- ComprobaciÃ³n de que se trata de un
    ↪  texto que representa
148       -- el retardo, si no es asÃ leemos la siguiente lÃnea.
149       if Correcto then
150
151         read(Input_Line, Input_Data);  -- El siguiente campo es el vector de
    ↪  pruebas.
152         -- Der a Izq
153
154         jc_in_interno <= TO_STDLOGICVECTOR(Input_Data)(3 downto 0);
155
156         -- De forma simultÃ¡nea lo volcaremos en consola en csv.
157         write(Std_Out_Line, Delay, right, 5);  -- Longitud del retardo, ej. "20 ms".
158         write(Std_Out_Line, Coma, right, 1);
159         write(Std_Out_Line, Input_Data, right, 2);  --Longitud de los datos de
    ↪  entrada.
160
161         Output_Line := Std_Out_Line;
162
163         writeline(output, Std_Out_Line);
164         writeline(Output_File, Output_Line);
```