



## **Memoria**

*Sistemas Electrónicos para la Automatización*

**Docente:** Jorge Romero Sánchez

## ***Ejercicios básicos***

---

**Izan Amador Bustos:** [izan.amador@uma.es](mailto:izan.amador@uma.es)

*Grado en Ingeniería Electrónica, Robótica y Mecatrónica*

## Contenido

1	Especificaciones	4
2	Soluciones	4
2.1	Multiplexor	4
2.1.1	Diseño	4
2.1.2	Simulación	6
2.1.3	Síntesis	8
2.2	Sumador	9
2.2.1	Diseño	9
2.2.2	Simulación	10
2.2.3	Síntesis	12
2.3	Registro de desplazamiento	13
2.3.1	Diseño	13
2.3.2	Simulación	14
2.3.3	Síntesis	17
2.4	Contador ascendente	18
2.4.1	Diseño	18
2.4.2	Simulación	21
2.4.3	Síntesis	25
2.4.4	Verificación con Analog Discovery	26
3	Anexos	27
3.1	Multiplexor	27
3.2	Sumador	31
3.3	Registro de desplazamiento	35
3.4	Contador ascendente	40



### Lista de figuras

1	Simulación de multiplexor 2 a 1 . . . . .	7
2	Fichero CSV generado para el MUX2a1 . . . . .	8
3	Esquemático RTL del multiplexor 2 a 1 . . . . .	8
4	Simulación de sumador . . . . .	11
5	Fichero CSV generado para el Sumador . . . . .	12
6	Esquemático RTL de simulación . . . . .	12
7	Simulación de registro de desplazamiento . . . . .	16
8	Fichero CSV generado para el registro de desplazamiento . . . . .	17
9	Esquemático RTL del registro de desplazamiento . . . . .	18
10	Simulación de contador ascendente . . . . .	24
11	Fichero CSV generado para el contador ascendente . . . . .	25
12	Esquemático RTL de contador ascendente . . . . .	25
13	Conexión del Analog Discovery con la Zybo . . . . .	26
14	Verificación de contador ascendente en Waveforms . . . . .	26

### Lista de códigos

1	Entidad del Mux2a1 . . . . .	5
2	Arquitectura del Mux2a1 . . . . .	5
3	Test Bench del Mux2a1 . . . . .	6
4	Entidad del Sumador . . . . .	9
5	Arquitectura del Sumador . . . . .	9
6	Test Bench del Sumador . . . . .	10
7	Entidad del Registro de desplazamiento . . . . .	13
8	Arquitectura del Registro de desplazamiento . . . . .	14
9	Test bench del Registro de desplazamiento . . . . .	16
10	Entidad del Contador ascendente . . . . .	19
11	Arquitectura del Contador ascendente . . . . .	21
12	Testbench del Contador ascendente . . . . .	23
13	Top del del Mux2a1 . . . . .	27
14	TestBench del Mux2a1 . . . . .	31
15	Top del Sumador . . . . .	31
16	TestBench del Sumador . . . . .	35
17	Top del Registro de desplazamiento . . . . .	36
18	TestBench del Registro de desplazamiento . . . . .	40
19	Top del Contador ascendente . . . . .	42
20	TestBench del Contador ascendente . . . . .	47

## 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

Definimos el multiplexor como un circuito combinacional con  $2^n$  entradas de datos,  $n$  entradas de selección y una única salida. La entrada de control permite seleccionar una única entrada de datos para que sea transmitida a la salida.

De esta manera, la implementación de un multiplexor puede tener distintas aplicaciones como la de serializador, selector, implementador de funciones lógicas (diseñándolas a partir de conexiones con 0 y 1 de sus entradas de la expresión de álgebra de bool deseada) o transmisor de datos.

#### Codificación en VHDL

```
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 Mux2a1

En nuestro caso, se realiza la implementación de dos entradas de datos, siendo el parámetro del tamaño del dato de entrada definido en la lista de genéricos como  $n$ . Las entradas de datos y la salida se definen como vectores de la librería `std_logic` por su tamaño variable en función del parámetro, mientras que es posible definir el selector como una entrada de un único bit.

```
1 architecture Behavioral of MuxV_2a1 is
2 begin
3     process(sel, x0, x1)
4     begin
5         if sel = '1' then
6             y <= x1;
7         else
8             y <= x0;
9         end if;
10    end process;
11 end Behavioral;
```

Código 2: Arquitectura del Mux2a1

En la arquitectura se incluyen las entradas en la lista de sensibilidades y se describe el comportamiento mediante un condicional. Si la entrada de selección se encuentra a uno, se asignará a la salida en valor de la segunda entrada. Si se encuentra a cero, se asignará la primera.

### 2.1.2. Simulación

#### Test Bench

```
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 MuxV_2a1
8          generic(
9              n : integer := 8);
10         port(
11             x0 : in  std_logic_vector(n-1 downto 0);
12             x1 : in  std_logic_vector(n-1 downto 0);
13             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) :=
19         ↵ (others => 'U');
20         signal sel_interno : std_logic := 'U';
21
22     begin
23
24         DUT : MuxV_2a1
25             generic map (n)
26             port map(
27                 x0 => x0_interno,
28                 x1 => x1_interno,
29                 sel => sel_interno,
30                 y => y_interno);
31
32         [...]
33
34         variable Input_Data : BIT_VECTOR(2 downto 0) := (OTHERS => '0');
```

### Código 3: Test Bench del Mux2a1

Para la simulación, se modifica la plantilla de banco de pruebas mediante uso de ficheros proporcionada, en la que se realizan distintos cambios, siendo los más significativos:

- La instanciación del componente
- La definición de las señales internas
- El mapeado de las entradas y salidas con los señales internas

En el fragmento del [Código 3](#) podemos observar la entidad vacía necesaria para la correcta definición del banco de pruebas.

Además, se define la variable de los datos de entrada de tamaño 3.

El código completo se incluye como anexo en [Código 14](#).

### Fichero de estímulos

```
#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
```

Se realiza un fichero de estímulos básico en el que las entradas de datos del dispositivo permanecen constantes mientras que la entrada de selección va variando.

### Cronogramas de simulación

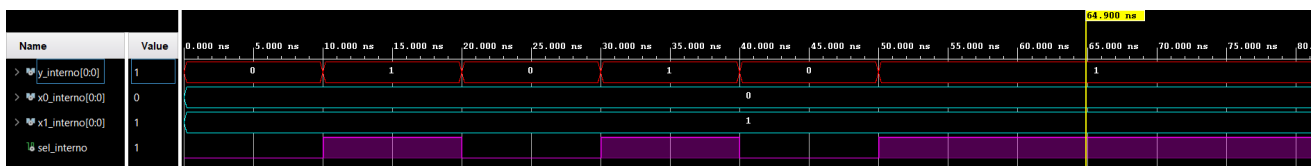
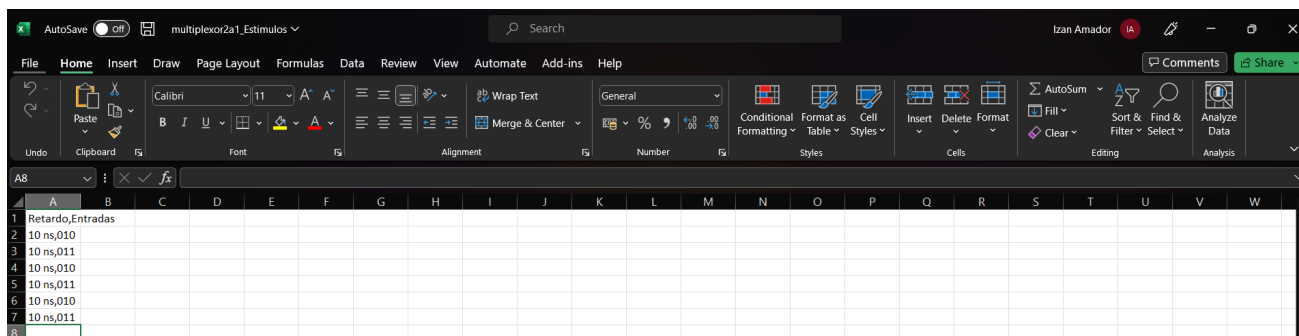


Figura 1: Simulación de multiplexor 2 a 1

En la [Figura 1](#) se aprecia como, efectivamente, el valor de la salida va cambiando en función de la entrada de la selección.

## Fichero CSV generado



Retardo	Entradas
10 ns,010	
10 ns,011	
10 ns,010	
10 ns,011	
10 ns,010	
10 ns,011	
10 ns,010	
10 ns,011	

Figura 2: Fichero CSV generado para el MUX2a1

Aunque en este ejercicio no se realiza la verificación con Analog Discovery, se genera un archivo csv con los estímulos de entrada para su lectura en Waveforms. El archivo se genera correctamente, con el mismo formato que los estímulos en el fichero de texto plano.

### 2.1.3. Síntesis

#### Esquemático RTL

Se realiza el esquemático del multiplexor deseado con la entrada de selección y las entradas de datos de 8 bits. No se aprecian elementos de circuitos secuenciales como latches, por lo que se considera correcta la síntesis.

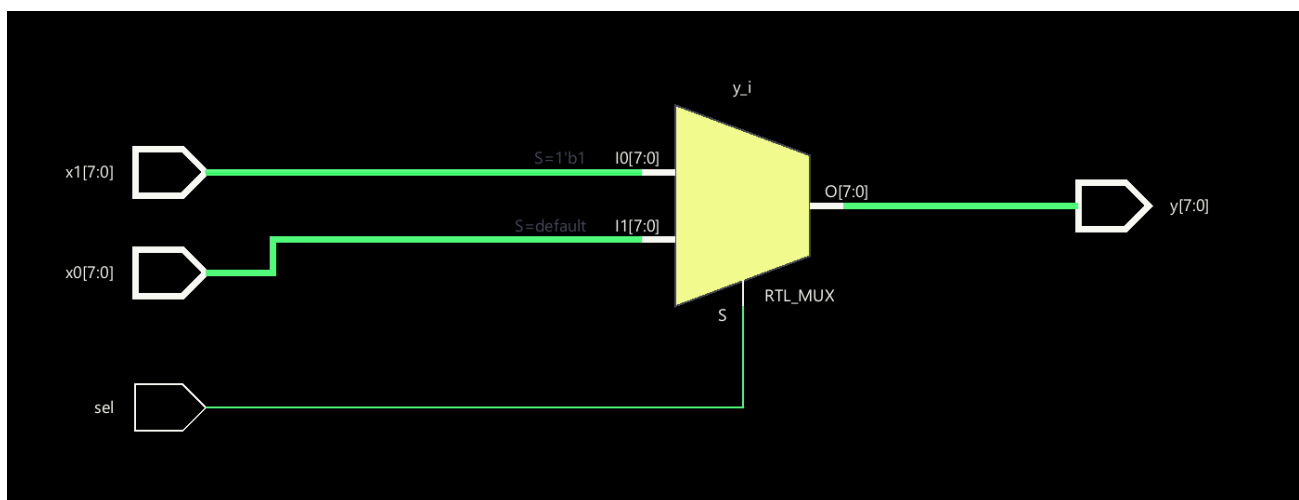


Figura 3: Esquemático RTL del multiplexor 2 a 1



## 2.2. Sumador

### 2.2.1. Diseño

#### Estudio teórico

Definimos el bloque sumador sin acarreo como un circuito que implementa la operación de adición sobre dos entradas numéricas (asumiendo que ambas son positivas). Precisamente, al no tener acarreo, no presenta la utilidad y flexibilidad de otros bloques como el semisumador o el sumador completo, que pueden ser concatenados para realizar elementos más complejos. Sencillamente implementa la operación de suma, sin realizar un trato especial al desbordamiento.

#### Codificación en VHDL

```
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

Se definen dos entradas x,y y una salida suma como vectores de la librería std\_logic en función de un genérico inicializado a 8 unidades por defecto. Además, se incluye la librería numeric para definir los tipos unsigned.

```
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

Se realiza un casting de **std\_logic\_vector** a **unsigned** y otro casting de **unsigned** a **std\_logic\_vector** antes de la asignación del valor a la salida s. Por lo tanto, el bloque interpretará los datos de entrada asumiendo que son cadenas de bits sin signo, por lo que no es posible realizar la suma de números negativos.

### 2.2.2. Simulación

#### Test Bench

```
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
17     => 'U');
18
19 begin
20
21     DUT : Sum
22         generic map (n)
23         port map(
24             x => x_interno,
25             y => y_interno,
26             s => s_interno);
27
28     [...]
```

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

Código 6: Test Bench del Sumador

Del fichero del banco de pruebas cabe destacar que el valor del tamaño de los vectores de entrada es 4 ya que cada entrada presenta dos bits. Por otra parte, el tamaño de todas las señales es el mismo ya que tanto las entradas como las salidas son de dos bits.

Se realizan los pasos necesarios para la correcta definición del banco de pruebas como son mantener la entidad vacía, la instanciación del componente y el mapeado de los puertos y los genéricos en el dispositivo bajo pruebas (DUT).

### Fichero de estímulos

```
#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
```

En el fichero de estímulos se codifican los dos bits más significativos como los valores de  $x$ , y los menos significativos como los valores de  $y$ . Se realizan todas las combinaciones posibles espaciadas mediante 10 nanosegundos.

### Cronogramas de simulación

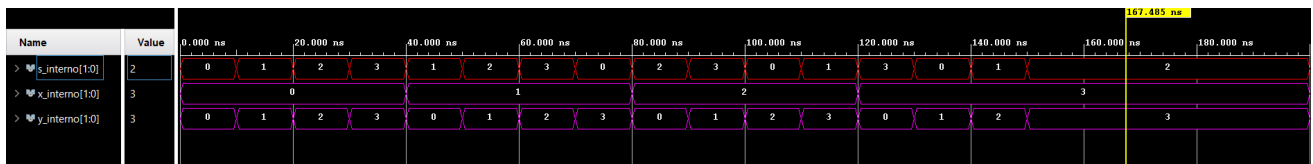
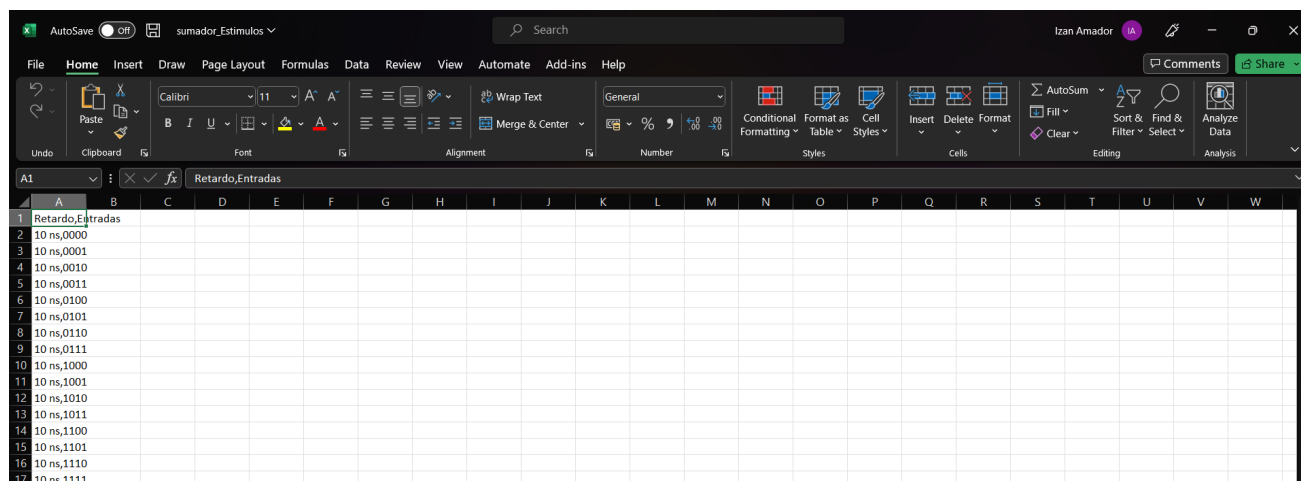


Figura 4: Simulación de sumador

Se observa como se realiza la operación de suma correctamente, en este caso se muestran los datos en formato decimal para una comprensión más sencilla de los mismos. Las entradas se representan en magenta y la salida en rojo. Finalmente, puede observarse como se produce desbordamiento en los últimos casos.

## Fichero CSV generado



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

Figura 5: Fichero CSV generado para el Sumador

Se escribe correctamente el fichero en formato de valores separados entre comas coincidiendo con el fichero de estímulos de texto plano.

### 2.2.3. Síntesis

#### Esquemático RTL

Se sintetiza con éxito el esquemático con dos entradas y una salida de 8 bits, por lo que coincide con el diseño inicial planteado.

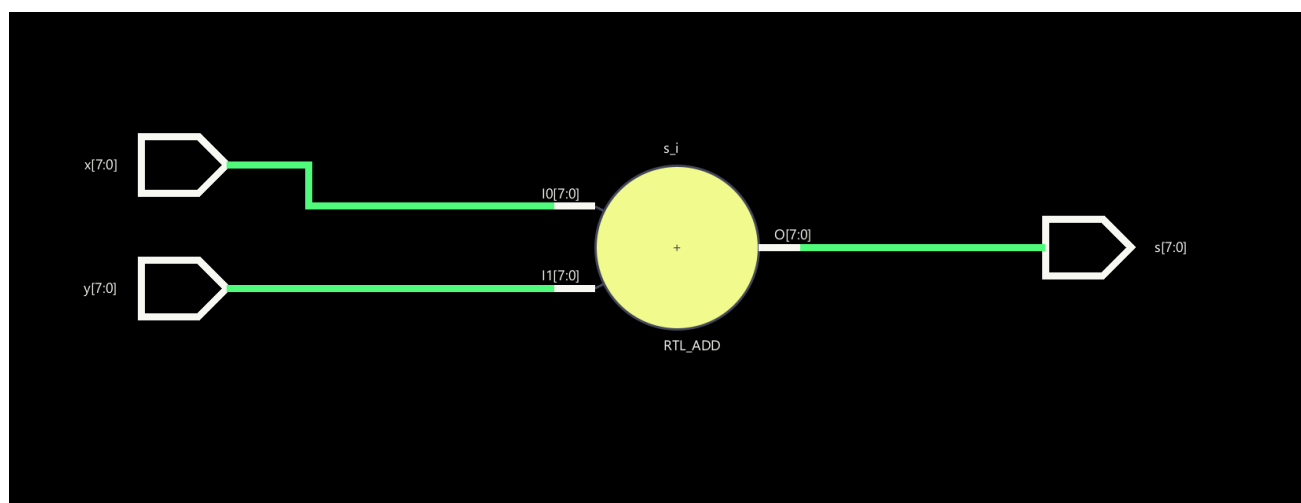


Figura 6: Esquemático RTL de simulación

## 2.3. Registro de desplazamiento

### 2.3.1. Diseño

#### Estudio teórico

Se define el registro de desplazamiento como un circuito de **carácter secuencial síncrono** formado por biestables que se conectan en cascada capaz de desplazar un dato a la derecha o a la izquierda, realizando las funciones de división y multiplicación respectivamente.

En nuestro caso, el dispositivo desplaza hacia la izquierda los datos y los carga también empezando por el bit menos significativo.

El funcionamiento del circuito depende de una entrada *d* que determina si se va a desplazar el dato o no, y otra entrada *des* en la que se carga el dato a desplazar. En el flanco de subida del reloj, se actualizan los valores almacenados, descartando el primer bit más significativo, desplazando hacia la izquierda el menos significativo e introduciendo el nuevo dato a cargar.

#### Codificación en VHDL

```
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

Cabe destacar de la entidad los dos tipos de entradas. En primer lugar las entradas de control *clk*, *reset*, *des* se realizarán mediante procesos, mientras que *d* (el dato a cargar) se cargará mediante el fichero de texto plano.

```
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

La arquitectura presenta dos procesos. El proceso secuencial con la lista de sensibilidad de *clk* y *reset* y el proceso combinacional de la asignación de *temp* a *q*.

### 2.3.2. Simulación

#### Test Bench

```
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  [...]
```

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

Código 9: Test bench del Registro de desplazamiento

En el banco de pruebas se realizan tres procesos distintos: uno para el reloj, otro para reset y una asignación constante de *des* a 1 para realizar desplazamiento durante toda la simulación. Se definen las constantes necesarias para los mismos y las señales internas con la inicialización al valor indefinido además de continuar con la estructura necesaria para el correcto funcionamiento del banco de prueba explicada en los ejercicios anteriores.

Se define la constante *n* a 2 para facilitar la simulación y entendimiento del dispositivo, en lugar del valor definido por defecto de 4.

### Fichero de estímulos

```
#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
```

En el fichero de estímulos se introducen los datos únicamente para *d* ya que las señales del control secuencial se realizan mediante procesos en el banco de pruebas.

### Cronogramas de simulación

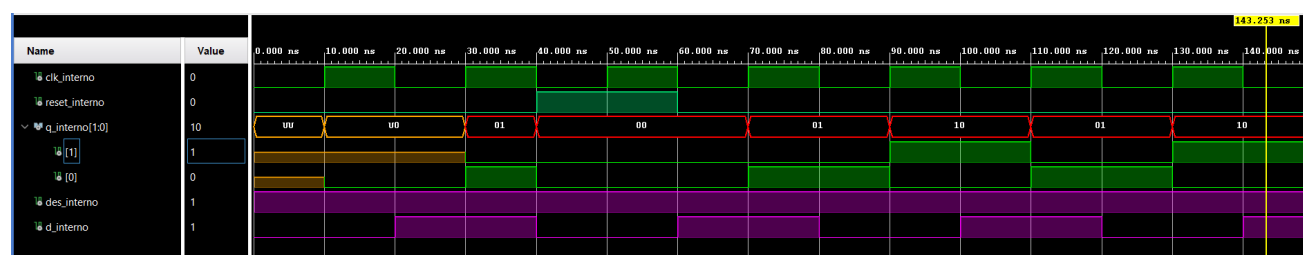


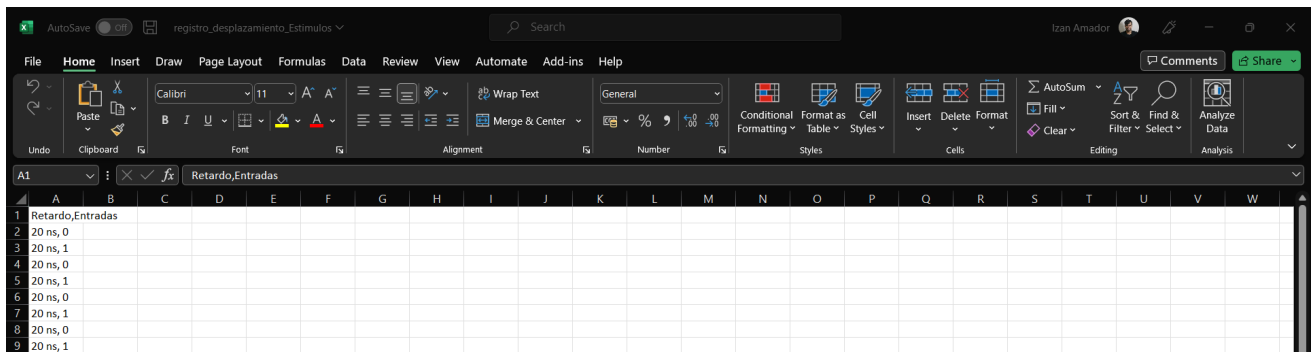
Figura 7: Simulación de registro de desplazamiento



En el cronograma puede apreciarse como se mantiene *desinterno* constante, por lo que siempre que haya un flanco de subida se desplaza el dato. Se realiza un reset y se aprecia el funcionamiento requerido, se carga el dato en el bit menos significativo, se descarta el bit más significativo y se realiza el desplazamiento del bit menos significativo al más significativo en el flanco de subida.

### Fichero CSV generado

Se observa como en el fichero de valores separados mediante comas se almacenan los estímulos del fichero de texto plano correctamente.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Retardo,Entradas																						
2	20 ns, 0																						
3	20 ns, 1																						
4	20 ns, 0																						
5	20 ns, 1																						
6	20 ns, 0																						
7	20 ns, 1																						
8	20 ns, 0																						
9	20 ns, 1																						

Figura 8: Fichero CSV generado para el registro de desplazamiento

### 2.3.3. Síntesis

#### Esquemático RTL

Se realiza la síntesis del diseño con éxito ya que el esquemático presenta las entradas y salidas deseadas. El tamaño de  $q$  es de 4 debido a que el genérico en el top tiene ese valor, mientras que en la simulación se usa el parámetro reducido a 2.

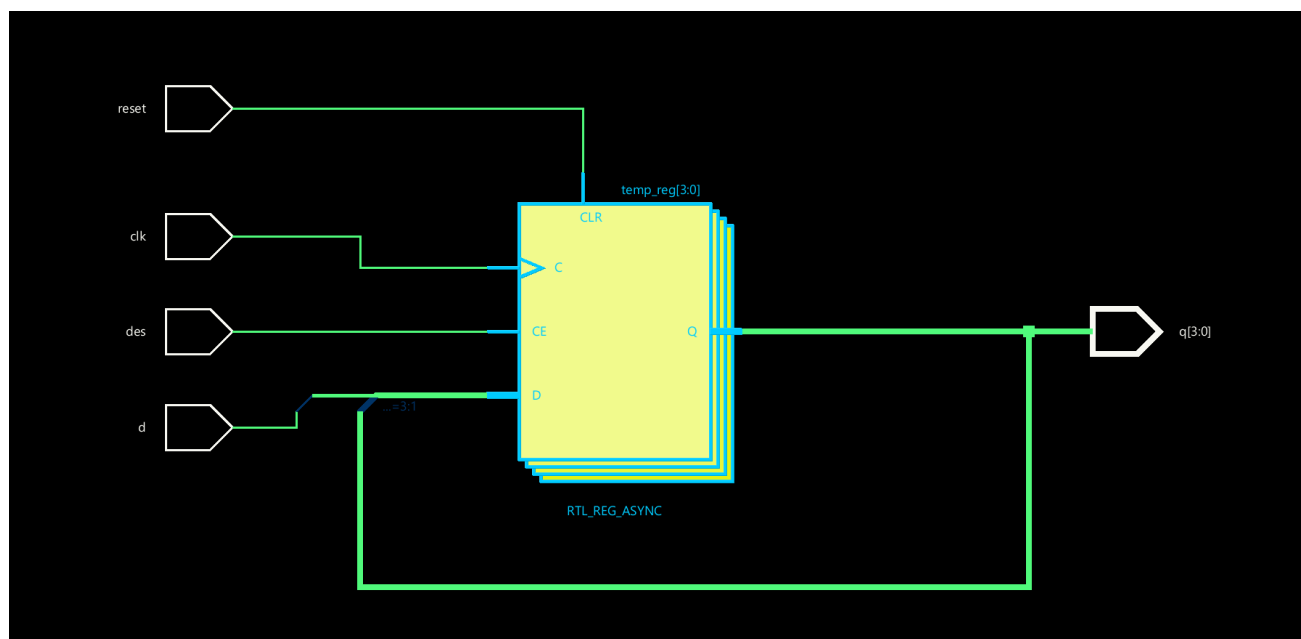


Figura 9: Esquemático RTL del registro de desplazamiento

## 2.4. Contador ascendente

### 2.4.1. Diseño

#### Estudio teórico

Se define el contador ascendente como un circuito secuencial síncrono capaz de realizar una cuenta con incrementos en unidades a un valor almacenado en memoria cada flanco de subida de reloj hasta un número determinado en el que se finaliza la cuenta.

Para el diseño del sistema se utilizarán elementos sincronizadores para controlar los posibles rebotes provenientes de los botones físicos al implementar el sistema.

Además, se incorporará la señal de activación de reloj con el objetivo de ralentizar y controlar la ejecución del sistema en su verificación mediante el módulo Analog Discovery.

#### Estructura de la entidad

```

1 entity top is
2   generic(counter_size : integer := 4;    -- tamaño del contador
3         filter_size : integer := 32);    -- tamaño 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

Las entradas de *ce*, *reset*, y *load* se habilitarán en el fichero .xdc para poder ser accionadas mediante la interfaz física de la FPGA.

El tamaño del filtro es de 4 unidades para la simulación para que el cronograma sea más legible y de 32 unidades para la implementación para su correcto funcionamiento con el hardware y las pruebas realizadas.

El funcionamiento de *jcín* es el siguiente: al activar la entrada *load* se carga su valor almacenado en la salida *jcout*. También se define una entrada de reloj *clk*, la salida *jdout* y una señal de salida de final de cuenta *fdc*.

El tamaño del contador se inicializa a 4 por simplicidad de la simulación.

```
1 architecture Behavioral of top is
2
3   component Sincronizador
4     generic (m : integer := 1);           -- tamaño del filtro
5     Port(
6       I      : in  std_logic;
7       CKE    : out std_logic;
8       reset  : in  std_logic;
9       clk    : in  std_logic);
10  end component Sincronizador;
11
12  component Contador_Asc
13    generic(n : integer := 8);
14    port(din      : in  std_logic_vector(n-1 downto 0);
15         dout     : out std_logic_vector(n-1 downto 0);
16         reset, ce, load : in  std_logic;
17         fdc      : out std_logic;
18         clk      : in  std_logic);
19  end component Contador_Asc;
20
21  signal reset_interno : std_logic := 'U';
```

```
22  signal ce_interno      : std_logic := 'U';
23  signal load_interno    : std_logic := 'U';
24
25  begin
26
27  Sincronizador_reset : Sincronizador generic map(filter_size)
28    port map(
29      I      => reset,
30      CKE     => reset_interno,
31      reset  => '0',
32      clk    => clk
33    );
34  Sincronizador_ce : Sincronizador generic map(filter_size)
35    port map(
36      I      => ce,
37      CKE     => ce_interno,
38      reset  => '0',
39      clk    => clk
40    );
41  Sincronizador_load : Sincronizador generic map(filter_size)
42    port map(
43      I      => load,
44      CKE     => load_interno,
45      reset  => '0',
46      clk    => clk
47    );
48
49  Contador : Contador_Asc generic map(counter_size)
50    port map(
51      din  => jc_in,
52      reset => reset_interno,
53      ce   => ce_interno,
54      load => load_interno,
55      clk  => clk,
56      dout => jd_out,
57      fdc  => fdc
58    );
59
```

```
60 end Behavioral;
```

### Código 11: Arquitectura del Contador ascendente

En el [Código 11](#), la arquitectura, se instancian los componentes del sincronizador (que posee el antirebotes) y el contador ascendente. Se definen las señales internas necesarias inicializadas como indefinidas y se mapean las señales internas con las señales y entradas correspondientes.

En el sincronizador, el genérico m se encuentra a 1 para su correcta identificación del componente, ya que así se encontraba definido inicialmente.

#### 2.4.2. Simulación

##### Test Bench

```
1  entity Test_Bench is
2  -- Port ( );
3  end Test_Bench;
4
5  architecture Comportamiento of Test_Bench is
6
7      component top
8          generic(counter_size : integer := 4;    -- tamaño del contador
9                  filter_size  : integer := 32); -- tamaño del filtro
10         port(jc_in  : in  std_logic_vector (counter_size-1 downto 0);
11              reset  : in  std_logic;
12              ce     : in  std_logic;
13              load   : in  std_logic;
14              clk    : in  std_logic;
15              jd_out : out std_logic_vector (counter_size-1 downto 0);
16              fdc    : out std_logic
17              );
18     end Component top;
19
20     constant semiperiodo : time := 10 ns;
21     constant periodo    : time := 2*semiperiodo;
22     constant counter_size: integer := 4;
23     constant filter_size : integer := 4;
24     signal jc_in_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
25     ↪ 'U');
26     signal reset_interno, fdc_interno : std_logic := 'U';
```

```
26  signal clk_interno, ce_interno, load_interno : std_logic := '0';
27  signal jd_out_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
    ↪  'U');
28  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     variable Input_Data : BIT_VECTOR(3 downto 0) := (OTHERS => '0');
```

Código 12: Testbench del Contador ascendente

Del banco de pruebas cabe destacar la reducción del tamaño del filtro y del contador para realizar una simulación más ligera y visible, además de la creación de los procesos internos de *reset*, *load* y *clockgen*.

La intención de este banco de pruebas no es comprobar la funcionalidad de la señal de carga, ya que se realizará la verificación con el módulo Analog Discovery de esta funcionalidad.

De esta manera, se pretende realizar un ciclo de cuenta completo hasta que la señal de final de cuenta se active.

### Fichero de estímulos

```
#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

```

En el fichero de simulación no se encuentran los valores equiespaciados con objeto de un testeo más aleatorio en Analog Discovery. Al no comprobar en el banco de pruebas la función de carga, no serán relevantes la diferencia de tiempos entre los distintos valores.

## Cronogramas de simulación

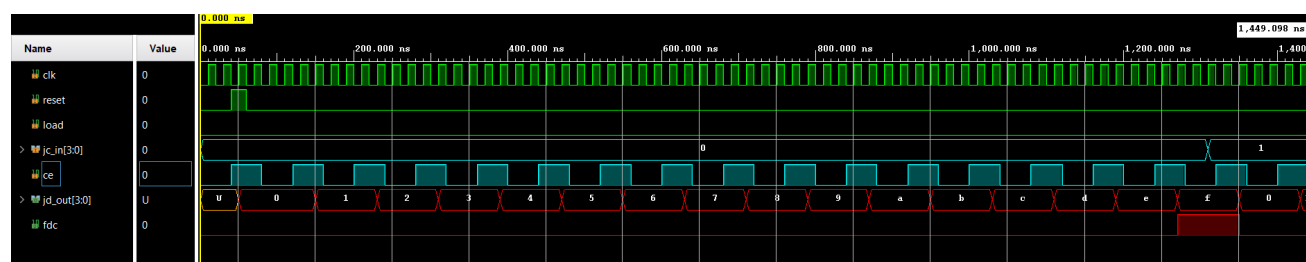


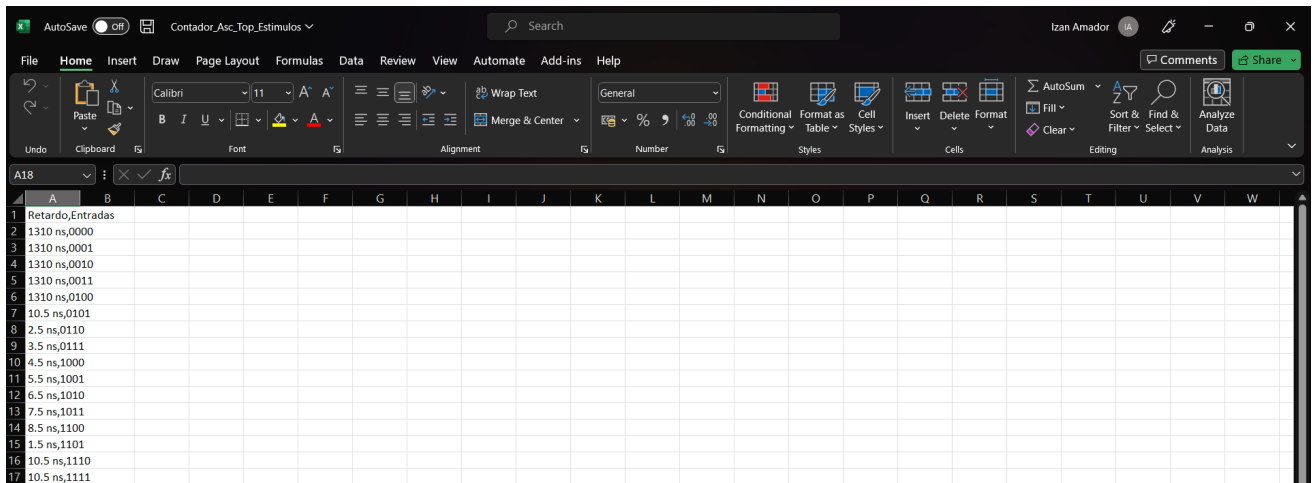
Figura 10: Simulación de contador ascendente

En el cronograma de la simulación se aprecia efectivamente un ciclo de cuenta completo. La salida del contador se encuentra indefinida hasta la entrada del reset y va incrementándose por unidades hasta llegar al valor de final de cuenta en el que se activa su salida. Se comprueba también el funcionamiento de la entrada ce simulando al usuario accionando el botón correspondiente.

## Fichero CSV generado

En el fichero de valores separados por comas se transcriben los valores de los estímulos del fichero de texto plano, por lo que se comprueba su correcto funcionamiento.





Row	Column A	Column B	Column C	Column D	Column E	Column F	Column G	Column H	Column I	Column J	Column K	Column L	Column M	Column N	Column O	Column P	Column Q	Column R	Column S	Column T	Column U	Column V	Column W
1	Retardo, Entradas																						
2	1310 ns, 0000																						
3	1310 ns, 0001																						
4	1310 ns, 0010																						
5	1310 ns, 0011																						
6	1310 ns, 0100																						
7	10.5 ns, 0101																						
8	2.5 ns, 0110																						
9	3.5 ns, 0111																						
10	4.5 ns, 1000																						
11	5.5 ns, 1001																						
12	6.5 ns, 1010																						
13	7.5 ns, 1011																						
14	8.5 ns, 1100																						
15	1.5 ns, 1101																						
16	10.5 ns, 1110																						
17	10.5 ns, 1111																						

Figura 11: Fichero CSV generado para el contador ascendente

### 2.4.3. Síntesis

#### Esquemático RTL

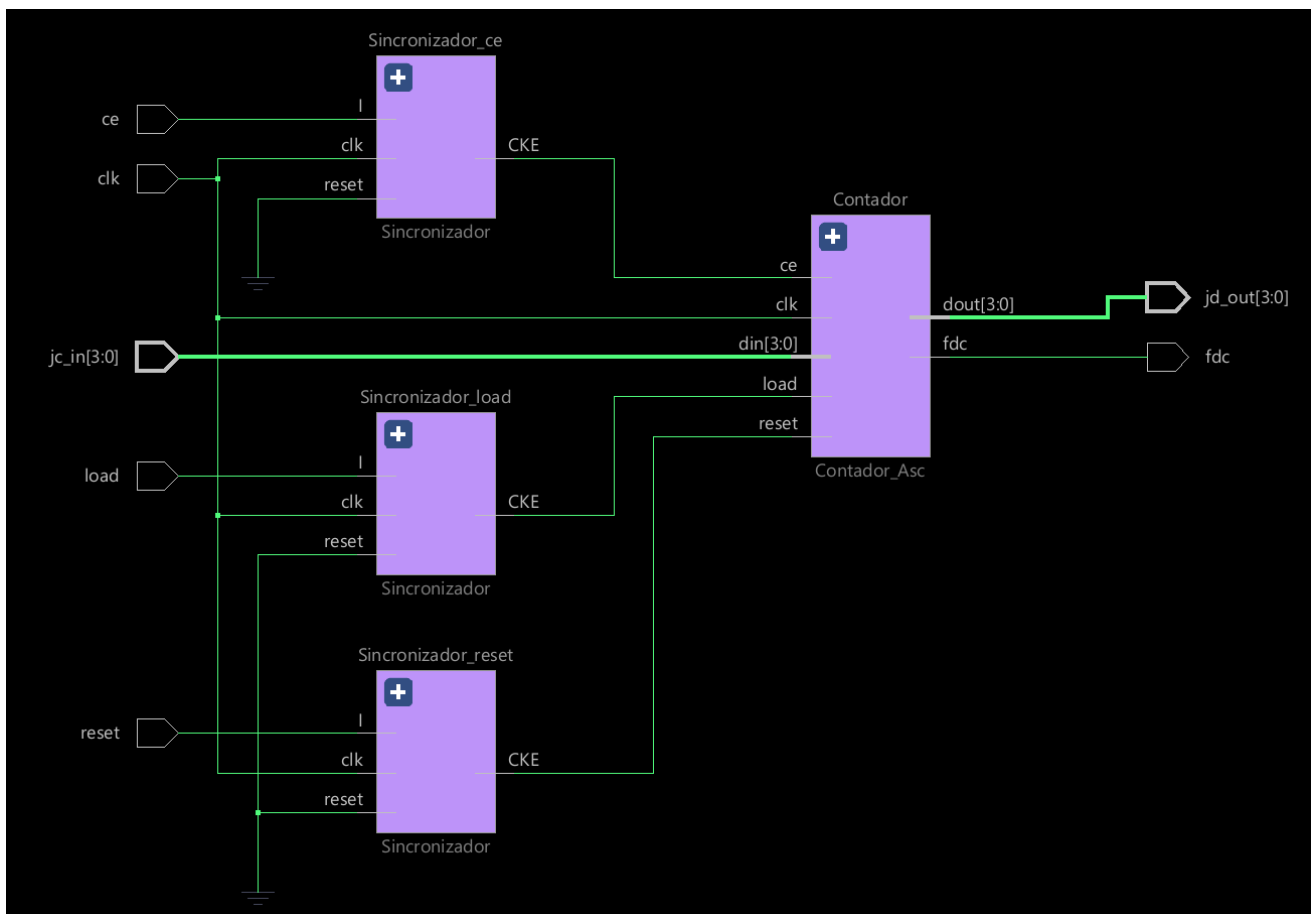


Figura 12: Esquemático RTL de contador ascendente

Se comprueba que el esquemático RTL generado es correcto (4 bits de entrada y salida, y un bit en las demás) tanto en el tamaño de las entradas y salidas como en la estructura del mismo. Cuenta con un sincronizador por cada entrada que se acciona mediante un botón en la interfaz de la FPGA y las señales de reloj son comunes.

#### 2.4.4. Verificación con Analog Discovery

##### Conexionado Analog Discovery

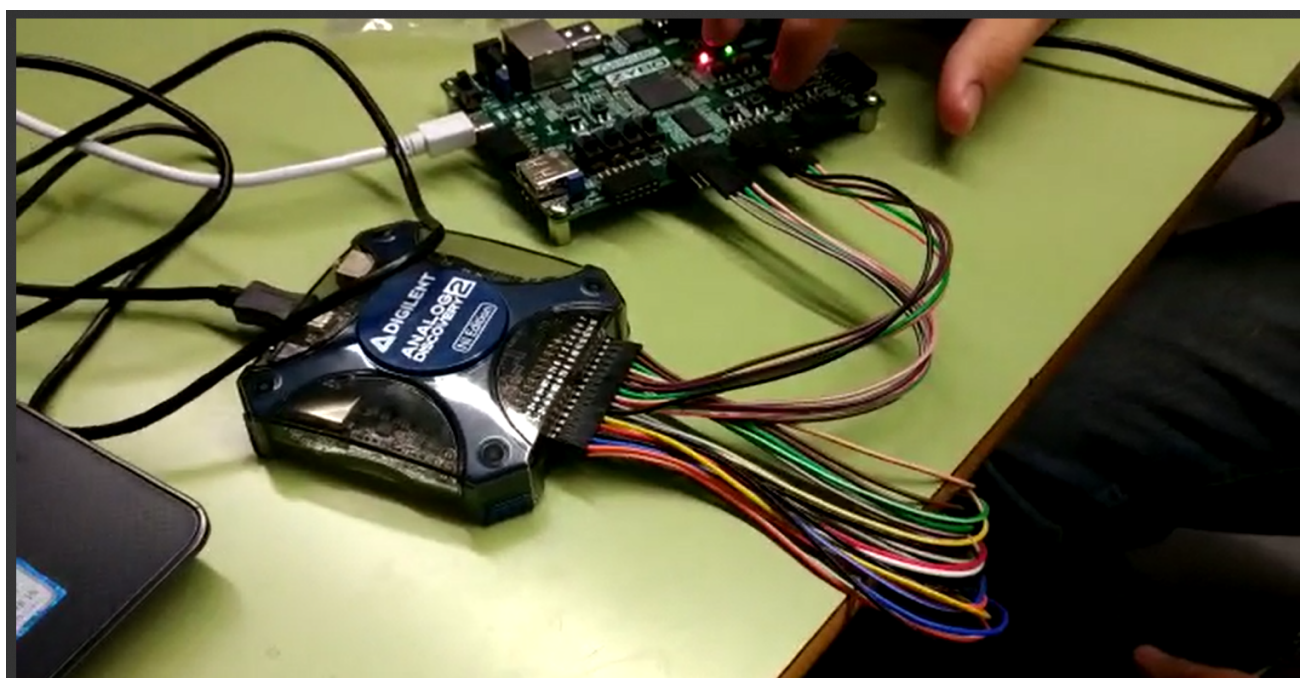


Figura 13: Conexión del Analog Discovery con la Zybo

Se realiza el conexionado de las salidas por los pines de los PMODS correspondientes con el conexionado de tierras comunes entre el Módulo Analog Discovery 2 y la placa de desarrollo Zybo.

##### Resultado de la verificación

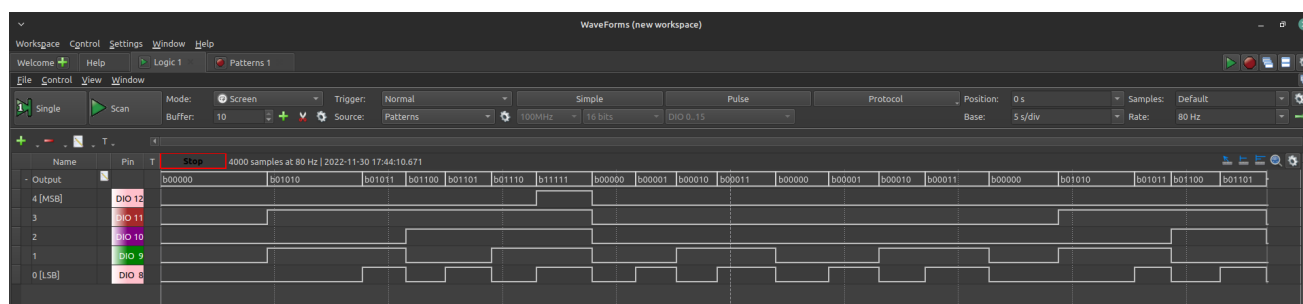


Figura 14: Verificación de contador ascendente en Waveforms

En el resultado de la verificación se puede apreciar como inicialmente el contador se encuentra a cero y se le carga el valor de cuenta binario 1010 desde el que continua la cuenta hasta el final de la misma, cuando se activa el bit de final de cuenta. Los avances del contador se producen cuando el usuario acciona el botón asociado a *ce*. Al final de la simulación se prueba un reset para comprobar el reseteo del sistema poniendo la salida a 0.

De esta manera se cierra el proceso de diseño, simulación y síntesis de un, obteniendo resultados coherentes con todos las etapas del proceso.

### 3. Anexos

#### 3.1. Multiplexor

```
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;
13
14 architecture Behavioral of MuxV_2a1 is
15 begin
16     process(sel, x0, x1)
17     begin
18         if sel = '1' then
19             y <= x1;
20         else
21             y <= x0;
22         end if;
23     end process;
24 end Behavioral;
```

Código 13: Top del del Mux2a1

```
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: 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 estímulos 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 estímulos 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); -- Comprobaci3n de que se trata de un
↪         texto que representa
91         -- el retardo, si no es as3 leemos la siguiente l3nea.
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 simult3nea 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
9 -- Target Devices: Zybo
10 -- Tool Versions: Vivado 2022.1
11 -- Description: basic test bench for a simple adder.
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20
21
22
23 library IEEE;
24 use IEEE.STD_LOGIC_1164.ALL;
25 use STD.textIO.ALL;                -- Se va a hacer uso de ficheros.
26
27 -- Uncomment the following library declaration if using
28 -- arithmetic functions with Signed or Unsigned values
29 --use IEEE.NUMERIC_STD.ALL;
30
31 -- Uncomment the following library declaration if instantiating
32 -- any Xilinx leaf cells in this code.
33 --library UNISIM;
34 --use UNISIM.VComponents.all;
35
36 entity Test_Bench_Fichero is
37 -- Port ( );
38 end Test_Bench_Fichero;
39
```





```
40 architecture Comportamiento of Test_Bench_Fichero is
41
42     component Sum
43         generic(
44             n : integer := 8);
45         port(x : in std_logic_vector(n-1 downto 0);
46             y : in std_logic_vector(n-1 downto 0);
47             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
52     => 'U');
53
54 begin
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 estímulos 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 estímulos 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); -- Comprobación de que se trata de un
   ↳ texto que representa
99     -- el retardo, si no es así leemos la siguiente línea.
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 simultánea 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 estímulos y los tiempos de
105     ↪ retardo para el semisumador.
106     file_open(Input_File,
107     ↪ "C:\Users\izana\Documents\GitHub\SEA\Estimulos\registro_desplazamiento_Estimulos.txt",
108     ↪ read_mode);
109
110     -- registro_desplazamiento.csv contiene los estímulos y los tiempos de retardo para
111     ↪ el Analog Discovery 2.
112     file_open(Output_File,
113     ↪ "C:\Users\izana\Documents\GitHub\SEA\CSV\registro_desplazamiento_Estimulos.csv",
114     ↪ write_mode);
115
116     -- Titles: Son para el formato EXCEL *.CSV (Comma Separated Values):
117     write(Std_Out_Line, string'("Retardo"), right, 7);
118     write(Std_Out_Line, Coma, right, 1);
119     write(Std_Out_Line, string'("Entradas"), right, 8);
120
121     Output_Line := Std_Out_Line;
122
123     writeline(output, Std_Out_Line);
124     writeline(Output_File, Output_Line);
125
126     while (not endfile(Input_File)) loop
127
128         readline(Input_File, Input_Line);
129
130         read(Input_Line, Delay, Correcto); -- Comprobación de que se trata de un
131     ↪ texto que representa
132         -- el retardo, si no es así leemos la siguiente línea.
133         if Correcto then
134
135             read(Input_Line, Input_Data); -- El siguiente campo es el vector de
136     ↪ pruebas.
```

```

129      -- Der a Izq
130
131      d_interno <= TO_STDLOGICVECTOR(Input_Data)(0);
132
133      -- De forma simultánea 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;    -- tamaño del contador
3              filter_size  : integer := 32);  -- tamaño 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);           -- tamaño 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(
56     I      => load,
57     CKE    => load_interno,
58     reset  => '0',
59     clk    => clk
60   );
61
62 Contador : Contador_Asc generic map(counter_size)
63   port map(
64     din  => jc_in,
65     reset => reset_interno,
66     ce   => ce_interno,
67     load => load_interno,
68     clk  => clk,
69     dout => jd_out,
70     fdc  => fdc
71   );
72
73 end Behavioral;
```

Código 19: Top del Contador ascendente

```
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: Contador Ascendente
7 -- Module Name: Test_Bench_top - Behavioral
8 -- Project Name: contador_ascendente
9 -- Target Devices: Zybo
```



```
10  -- Tool Versions: Vivado 2022.1
11  -- Description: basic upwards counter
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  -- 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;    -- tamaño del contador
43              filter_size  : integer := 32); -- tamaño 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 =>
59 ↪ 'U');
60 signal reset_interno, fdc_interno : std_logic := 'U';
61 signal clk_interno, ce_interno, load_interno : std_logic := '0';
62 signal jd_out_interno : std_logic_vector(counter_size-1 downto 0) := (others =>
63 ↪ 'U');
64 begin
65
66 DUT : top
67 generic map (counter_size, filter_size)
68 port map(
69     jc_in  => jc_in_interno,
70     reset  => reset_interno,
71     ce     => ce_interno,
72     load   => load_interno,
73     clk    => clk_interno,
74     jd_out => jd_out_interno,
75     fdc    => fdc_interno);
76
77 clock_gen: process (clk_interno) is
78 begin
79     if clk_interno = '0' then
80         clk_interno <= '1' after semiperiodo,
81             '0' after periodo;
82     end if;
83 end process clock_gen;
84
85 reset: process
86 begin
87     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                := ',';
```

```
125
126 begin
127
128 -- Contador_Asc_Top_Estimulos.txt contiene los estímulos 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 estímulos 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); -- Comprobación de que se trata de un
   ↳ texto que representa
148     -- el retardo, si no es así leemos la siguiente línea.
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 simultánea 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);
165
166      wait for Delay;
167      end if;
168      end loop;
169
170      file_close(Input_File);          -- Cerramos el fichero de entrada.
171      file_close(Output_File);        -- Cerramos el fichero de salida.
172      wait;
173      end process Estimulos_Desde_Fichero;
174
175
176      end Comportamiento;
```

Código 20: TestBench del Contador ascendente