# Contador ascendente y descendente en VHDL

El objetivo es diseñar y simular el siguiente bloque contador en VHDL.

Debe ser capaz de contar 4 bits ascendente y descendentemente con un control de sentido.





1. Creamos un nuevo proyecto en el ISE Project Navigator<sup>1</sup>

File -> New Project

Se abrirá la siguiente interfaz.

🍃 New Project	Wizard	
Create New Pro	ect and type.	
Enter a name, locati	ns, and comment for the project contador4bits	
Location:	C:\practicas\contador4bits	
Working Directory: Description:	C:\practicas\contador4bits	
Select the type of to Top-level source typ HDL	p-level source for the project e:	×
More Info		Next Cancel

### Fig. 2 Nuevo Proyecto

Ponemos un nombre que no contenga espacios por ejemplo "contador4bits".

<sup>&</sup>lt;sup>1</sup> Para el presente tutorial se ha utilizado la versión 14.2 de Xilinx ISE Design Suite. Con licencia ISE Web Pack.

Elegimos la locación y el directorio de trabajo (Dejar las ubicaciones por defecto)

Presionamos "next".

pecify device and project properties.		
pecify device and project properties.		
elect the device and design now for the p	project	
Property Name	Value	
Evaluation Development Board	Spartan-3E Starter Board	
Product Category	All	
Family	Spartan3E	
Device	XC3S500E	
Package	FG320	
Speed	-4	
Top-Level Source Type	HDL	
Synthesis Tool	XST (VHDL/Verilog)	
Simulator	ISim (VHDL/Verilog)	
Preferred Language	VHDL	
Property Specification in Project File	Store all values	
Manual Compile Order		
	VUDL 02	
VHDL Source Analysis Standard	VIIUL-95	

2. En la siguiente ventana debemos especificar la tarjeta en la que vamos a desarrollar el proyecto.

Fig. 3 Especificaciones de tecnología

En nuestro caso contamos con la tarjeta Spartan 3E Starter Board que contiene las siguiente FPGA:

Family: Spartan3E

Device: XC3S500E

Package: FG320

#### Speed: -4

\*Los datos de arriba se pueden obtener directamente del chip FPGA incluido en la tarjeta.

Top-Level Source Type: HDL

Synthesis Tool: XST (VHDL/Verilog)

Simulator: ISim (VHDL/Verilog)

Preferred lenguaje: VHDL

Presionamos "next".

3. Aparecerá la siguiente ventana que simplemente detalla el proyecto creado.

	×
S New Project Wizard	
Project Summary	
Project Navigator will create a new project with the following specifications.	
Project:	*
Project Name: contador4bits1	
Project Path: C:\practicas\contador4bits1	
Working Directory: C:\practicas\contador4bits1	
Description:	
Top Level Source Type: HDL	
Devices	
Evaluation Development Board, Spartan_3E Starter Board	
Device Family: Spartan3F	
Device: xc3s500e	E
Package: fg320	
Speed: -4	
Top-Level Source Type: HDL	
Synthesis Tool: XST (VHDL/Verilog)	
Simulator: ISim (VHDL/Verilog)	
Preferred Language: VHDL	
Property Specification in Project File: Store all values	
Manual Compile Order: false	
VHDL Source Analysis Standard: VHDL-93	
Maggaga Filtering, disabled	-
More Info Finish	Cancel

Fig. 4

Presionamos "Finish".

4. En el tab "design" de la izquierda hacemos doble clic sobre el ícono de nuestro proyecto y creamos un nuevo recurso (New Source), como en la fig 5.

JSE Project Navigator (P.28xd) - C:\practicas\contador4bits1\contado	or4
File Edit View Project Source Process Tools Window	La
🗋 🏓 🖶 🖓 🖓 🖬 🛱 🗶 👘 🔂 😭	8
Design ↔ □ ₽ ×	
View: 💿 🏟 Implementation 🔘 🞆 Simulation	
Hierarchy ^	
Add Source	
The second secon	
Manual Compile Order	
x Fil	
File/Path Display	
No single Expand All	
Collapse All	
Find Ctrl+F	
Design Properties	
Start Tiles Consign	
Source Libraries	
I work	
Console	
Errors 🔬 Warnings 🦝 Find in Files Results	
A dat - wanni sanna ka klas nasi sak	

Fig. 5

5. En la ventana emergente seleccionamos el tipo de recurso a crear, como queremos diseñar el contador en vhdl, escogemos un nuevo recurso de tipo "VHDL module".



Fig. 6

En "File name" le damos un nombre al módulo VHDL a crear, igualmente no debe contener espacios por ejemplo "sumador".

Clic en "next".

6. En la siguiente ventana podemos ingresar las entradas y salidas que tendrá la entidad. (consulte la Fig. 1).

	00							x
	🕒 🛛 New So	urce Wizard						
s	Define Modu	ıle						
	Specify ports for	module.						
1	Entity name	sumador						
	Architecture name	Behavioral						
ł		Port Name	Direction	ı	Bus	MSB	LSB	<u>^</u>
1	reset		in	•				
	clk		in	•				
	sentido		in	•				≡
	Q		out	•	<b>v</b>	3	0	
			in	•				
			in	•				
l			in	•				
•			in	•				
1			in	•				-
	More Info					Next	Cance	

Declaramos como entradas en la columna Port Name:

Reset -> in

clk -> in

sentido -> in

Q -> out -> bus -> MSB 3 LSB 0

En salida "Q", seleccionamos la opción de "Bus" debido a que es la salida del contador de 4 bits, por ende debe tener de 0 a 3 bits.

A continuación damos click en "next", y luego en "finish" en la ventana resumen.

# 7. Ahora está preparado el archivo Vhdl para su edición.



#### Fig. 8

En el bloque de código de las librerías debemos incluir las siguiente sentencia "use IEEE.std\_logic\_unsigned.all;" esto para poder trabajar con sumas y restas sin signo.

Quedaría el código superior así:

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.std\_logic\_unsigned.all;

8. La declaración de la entidad debe quedar de la siguiente manera:

\_\_\_\_\_

\_\_\_\_\_

entity contador is Port ( reset : in STD\_LOGIC;

# clk : in STD\_LOGIC; sentido : in STD\_LOGIC := '0'; Q : out STD\_LOGIC\_VECTOR (3 downto 0));

### end contador;

\*recordar que las líneas de código anteriores pueden generarse automáticamente siguiendo el proceso del paso 6, o pueden escribirse directamente en el código obviando el paso 6. Sin embargo estas líneas pueden modificarse en cualquier momento.

# 9. Luego describimos la arquitectura de la entidad, esta arquitectura será de tipo comportamental o "behavioral".

architecture Behavioral of contador is

```
signal counter : STD_LOGIC_VECTOR (3 downto 0) := "0000";
```

begin

```
process(reset,clk)
```

begin

Arquitectura comportamental.

En cada pulso de reloj se incrementa una unidad a la señal contador

10. El código complete quedaría así:

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.std\_logic\_unsigned.all;

entity contador is Port ( reset : in STD\_LOGIC; clk : in STD\_LOGIC; sentido : in STD\_LOGIC := '0'; Q : out STD\_LOGIC\_VECTOR (3 downto 0)); end contador;

architecture Behavioral of contador is

signal counter : STD\_LOGIC\_VECTOR (3 downto 0) := "0000"; begin

process(reset,clk)

begin

```
if reset='1' then
    counter <= "0000";
elsif clk'event and clk = '1' then
    if sentido = '1' then
    counter <= counter + 1;
    else
        counter <= counter - 1;
    end if;
Q <= counter;
end if;</pre>
```

end process;

end Behavioral;

### 11. SIMULACIÓN

Una vez terminado nuestro código podemos realizar la simulación para constatar el correcto funcionamiento.

Creamos un nuevo recurso como en el paso 4, con excepción de que esta vez necesitamos un recurso de tipo "VHDL test bench".





Le asignamos un nombre al nuevo recurso, click en "next".

En la ventana siguiente se nos pide que asociemos un recurso a este "test bench", seleccionamos el recurso correcto, en este caso solo existe un recurso.

New Source Wizard	x
Associate Source	
Select a source with which to associate the new source.	
More Info Next Cance	2

Fig. 10

A continuación clic en "next" y luego en "finish".

Ahora tenemos un archivo de tipo VHDL pero con las señales, puertos y temporizaciones necesarias para una simulación.

\*Seleccionando la interfaz "simulation" en la barra de la izquierda podemos visualizar el archivo de simulación.

JSE Project Navigator (P.28xd) - C:\practicas\contador4t	tsincremento\contador4bitsincremento.xise - [test3.vhd]	- 0 <b>X</b>
File Edit View Project Source Process Too	Window Layout Help	_ 8 ×
※ ある X 前 る X を こ	୬ ୬ ୬ ୬ ≥ 🔀 🧏 🗄 🗉 🖙 🖋 🕨 🗴 🗶 👂	
Design + 🗆 🗗 🗙 🚽	38 ARCHITECTURE behavior OF test3 IS	*
Wew:	39 40 Component Declaration for the Unit Under Test (UUT) 41 42 COMPONENT contrador	
contador4bitsincremento	43 PORT (	
📅 🖻 🚺 xc3s500e-4fg320	44 reset : IN std_logic;	
test3 - behavior (test3.vhd)	45 clk: IN std logic; 45 mentide: IN std logic;	
en uut - contador - benavioral (contat	47 Q: OUT std logic vector (3 downto 0)	
2	48 );	
· · · · · · · · · · · · · · · · · · ·	49 END COMPONENT;	
No Processes Running	51	
Processes: test3 - behavior	52 Inputs	=
🖲 🖆 🎾 ISim Simulator	<pre>53 signal reset : std_logic := '0';</pre>	
Real Check Syntax	54 signal clk: std_logic := '0';	
Simulate Behavioral Works	56	
	57Outputs	
	58 signal Q : std_logic_vector(3 downto 0);	
	60 Clock period definitions	
> Start 👔 Files 📲 Design	61 constant clk_period : time := 10 ns;	
	62	
	63 BEGIN	
Source Libraries	65 Instantiate the Unit Under Test (UUT)	-
🚛 🐵 🚺 work	< III	•
	Design Summary (Programming File Generated) 🗵 📋 contador.vhd* 🗵 📄 contador.ucf 🗵 📄 test3.vhd 区	
Console		↔□₽×
Process "Configure Target Device" lau	nched successfully	*
Country , Without hims TCP, Tout Reliance		
Started . Daunching ISE lext Editor	o Euro desus vila .	*
< III.		•
🙆 Errors 🔔 Warnings 🐹 Find in Files Results		
		Ln 38 Col 1 VHDL

Fig. 11

Seleccionamos el archivo "VHDL test bench" recién creado, chequeamos la sintaxis en la barra de la izquierda y luego procedemos a correr la simulación.

**12.** Al terminar de correr el proceso "Simulate Behavioral Model", se abrirá otro programa llamado "Isim".

ISim (P.28x	d) - (C	efault.wcfg]			Common sur-	-	-	the second little	A		-		-		
📷 File Edit	Vie	w Simulation Window Layo	ut Help												_ 8 ×
🗋 🆻 🔒	5	X 🗈 🗋 🗙 🚷 🛤 🖓	AA 😹 🕹 🕆 🐼	5 H H	🖻 🎤 K?	P P 🔊	۵ 🔎	12 2r 🕴	1 🗠 👌 🖸	▶	.00us 🔻 🐓	🗉    🗔 Re-l	aunch		
	۲					13.006787 us									*
Simulation Ob	۲														
uu »		Name	Value	· · · · ·	12 us	<u> </u>	14 us		10	6 us		18 us		20 us	
Object Name	۶	Le reset	0												
lig reset	R	sentido	0												
le cik	6	▶ 🛃 q[3:0]	0001	X 00	10 )	0001	0000	1111	) 1110		1101	) 1100	) 1011	/ 1100	X 1101
⊳ 🔏 q[3:0]	1	🔓 clk_period	10000 ps							10000 ps					
🖞 clk_pe	<b>d</b> r														
	÷														
	4														
	-														
	ाः जि														
				X1: 13.0067	787 us										
	_	+	+ F	•											F v
* E +	22		Default.w	cfg			×								
Console															⇔⊡₽×
# run 1.00us															~
# run 1.00us															
ISim> #run 1.00us															_
ISim>			1.7**												*
Console		Compilation Log 🛛 🗧 Breakpoints	Find in Files Results	M Searc	h Results										
														Sim	Time: 22 000 000 ns



En el mismo podemos interactuar con las señales de entrada y observar la simulación y comportamiento de nuestro circuito.

Para modificar las señales de entradas se procede de la siguiente manera:





Damos click derecho en cualquier señal de entrada, por ejemplo en "sentido". Y seleccionamos "force constant".

Force Selected Signal	? ×										
Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.											
Signal Name:	/test3/sentido										
Value Radix	Binary 💌										
Force to Value:	1										
Starting at Time Offset:	0										
Cancel after Time Offset:											
OK Cancel	Apply Help										

Fig. 14

En la ventana emergente damos un valor de '0' o '1', en "Force to value".

\*Según nuestro ejemplo si la señal "sentido" está en '0' el contador descenderá, caso contrario '1', el contador ascenderá.

								13.006	5787 u	IS																				
7	Name	Value			12 u	ıs					14 us					16	6 us		. 1				18 us					20 us		
	le reset	0																												
	U <sub>lo</sub> cik	0																								L				
	🌡 sentido	0																												
	• 📲 q[3:0]	0001	$\supset$	00	010		00	01	X	00	opo	X	1	111	1	1110	)	X	110	1	Х	11	00	X	10	11	1	100	11	)1
	🔓 clk_period	10000 ps															10000	ps												

## Fig. 15

En la figura 15 se puede observar el funcionamiento del contador y el cambio de la señal sentido a los 18us.

Para correr la simulación y observar las señales se debe dar clic en el ícono que se observa en Fig 16 e interactuar con los íconos de zoom out y zoom in.



Fig. 16

**Creado por:** Tuesman Daniel Castillo Ing. Escuela de Electrónica y Telecomunicaciones UTPL Universidad Técnica Particular de Loja UTPL