Sistemas de Tempo Real: FPGAs
2013/14

Práctica 1: Introducción a Xilinx ISE

Introducción

En esta primera práctica se realizará una primera aproximación al entorno de desarrollo y *downloading* de Xilinx, llamado ISE (*Integrated System Environment*). Se recorrerán las funcionalidades básicas y se programará un primer ejemplo en VHDL.

Creación de un proyecto nuevo

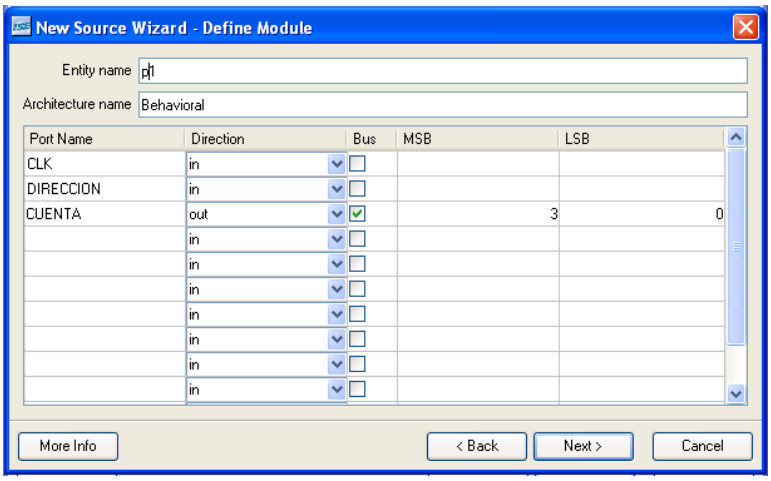
Una vez arrancado el ISE, para crear un proyecto nuevo hay que ir a:

1) **File > New Project**.
2) Teclear el nombre del proyecto (por ejemplo, "practica1")
3) Indicar el directorio donde se creará un subdirectorio con el contenido del proyecto a crear.
4) Seleccionar **HDL** en **Top-Level Source Type**.
5) Hacer click en **Next**.
6) En la siguiente pantalla del *New Project Wizard* se seleccionan las características de la FPGA para la cual se desarrollará el código. Aunque estos parámetros influyen en la generación del .bit que se escribe en la FPGA, en las prácticas a realizar en el laboratorio no van a ser determinantes (tan sólo influirán en ciertos detalles de la configuración). La FPGA que vamos a utilizar para esta práctica es una **Spartan 3E**, **XC3S500E** (para poder seleccionarla hay que dejar *Evaluation Development Board* a *None Specified* y *Product Category* a *General Purpose*). A mayores se seleccionará el encapsulado **FG320** y velocidad **-4**.
7) En **Synthesis Tool** se debe escoger **XST**, en **Simulator**, **ISim**, y en **Preferred Language**, **VHDL**. Dejar el resto de parámetros con sus valores por defecto. Darle entonces a **Next** y en la siguiente pantalla a **Finish**.

   A continuación añadiremos un nuevo módulo VHDL al proyecto:

1) Hacer click en el menú **Project** y luego en **New Source**.
2) Escoger **VHDL Module**, poner de nombre "p1", dejar marcado **Add to Project** y pulsar en **Next**.
3) En la siguiente pantalla se pueden declarar los puertos de entrada/salida. En esta primera práctica se va a implementar un contador de 4 bits que en cada ciclo de reloj incrementa o disminuye su valor en función de una señal llamada "DIRECCION". Para ello será necesario añadir en **Port Name** tres señales: **CLK** de tipo **IN**, **DIRECCION** de tipo **IN** y **CUENTA** de tipo **OUT**. La señal **CUENTA**, dado que va a ser donde se almacene

el valor del contador, debe ser de 4 bits: ha de marcarse **Bus**, se escribe 3 en **MSB** y 0 en **LSB** (esto último quiere decir que se accederá al bit más significativo con el índice 3 y al menos significativo con el 0). Finalmente, pulsar en **Next** y a continuación en **Finish**.



Utilización de plantillas en ISE

Si se ha realizado correctamente el proceso anterior se debería ver código VHDL como el de la siguiente captura (si no se ve, hacer doble click en "nombre_práctica – Behavioral (nombre_práctica.vhd)", dentro de la sección **Design**, bajo **Hierarchy**):

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

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

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

entity p1 is
    Port ( CLK : in  STD_LOGIC;
           DIRECCION : in  STD_LOGIC;
           CUENTA : out  STD_LOGIC_VECTOR (3 downto 0));
end p1;

architecture Behavioral of p1 is

begin


end Behavioral;
```

El código creado hasta ahora declara la estructura del contador, pero todavía debe indicarse la forma en que se debe comportar el contador. Para realizar esta tarea se va a hacer uso de las plantillas que dispone ISE. Para ello se debe hacer:

1) Colocar el cursor en el código, entre **Begin** y **end Behavioral**, para indicar dónde se va a añadir el código de la plantilla.
2) Ir a **Edit > Language Templates** e ir pulsando en "+" en las siguientes etiquetas: **VHDL > Synthesis Constructs > Coding Examples > Counters > Binary > Up/Down Counters**.
3) Hacer click con el botón derecho del ratón sobre **Simple Counter** y seleccionar **Use in file**. Esta acción permitirá copiar el código del contador al lugar donde se colocó el cursor en el paso (1).
4) Para que el código de la plantilla funcione es necesario editarlo, puesto que las variables de entrada/salida que se utilizan no tienen el mismo nombre que las que hemos creado anteriormente y además existe una variable que no ha sido declarada.
    a. Primero ha de crearse una variable que lleve la cuenta actual. Debe de colocarse el cursor entre la línea donde comienza a definirse la arquitectura (**architecture Behavioral[…]**) y el **Begin**, escribiendo la siguiente definición:

        signal cuenta_actual : std_logic_vector (3 downto 0) := "0000";

        Con esto lo que hemos hecho ha sido inicializar el valor actual de contador, que es un vector de 4 bits, a todo ceros.

    b. Después ha de reemplazarse **<clock>** por **CLK**, **<count_direction>** por **DIRECCION** y **<count>** por **cuenta_actual**.
    c. Por último, se debe asignar el valor de **cuenta_actual** a la variable de salida **CUENTA**. Para hacerlo, se debe poner el cursor tras la línea "**end process;**" y escribir:

        CUENTA <= cuenta_actual;

    d. Salvar el código con **File > Save**.

A mayores, para poder hacer uso de las operaciones declaradas, es necesario importar la librería STD_LOGIC_UNSIGNED. Para realizarlo, hay que añadir tras la sentencia "**library IEEE;"** la siguiente línea:

use IEEE.STD_LOGIC_UNSIGNED.ALL;

En estos momentos el código debería ser como el que sigue:

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

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

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

entity p1 is
    Port ( CLK : in  STD_LOGIC;
           DIRECCION : in  STD_LOGIC;
           CUENTA : out  STD_LOGIC_VECTOR (3 downto 0));
end p1;

architecture Behavioral of p1 is

signal cuenta_actual : std_logic_vector (3 downto 0) := "0000";

begin

process (CLK)
begin
   if CLK='1' and CLK'event then
      if DIRECCION='1' then
         cuenta_actual <= cuenta_actual + 1;
      else
         cuenta_actual <= cuenta_actual - 1;
      end if;
   end if;
end process;

CUENTA <= cuenta_actual;


end Behavioral;
```

Comprobando la sintaxis del código

Para comprobar que no nos hemos equivocado al escribir nuestro código VHDL, el Xilinx ISE tiene una utilidad que permite verificarlo (algo así como un compilador de VHDL). Para ejecutarlo, hay que seleccionar primero en la ventana **Hierarchy** el fichero VHDL que acabamos de editar. En la ventana justo de abajo (**Processes**) hay una etiqueta que pone **Synthesize – XST**, donde hay que pulsar en "+" y hacer doble click en **Check Syntax**.

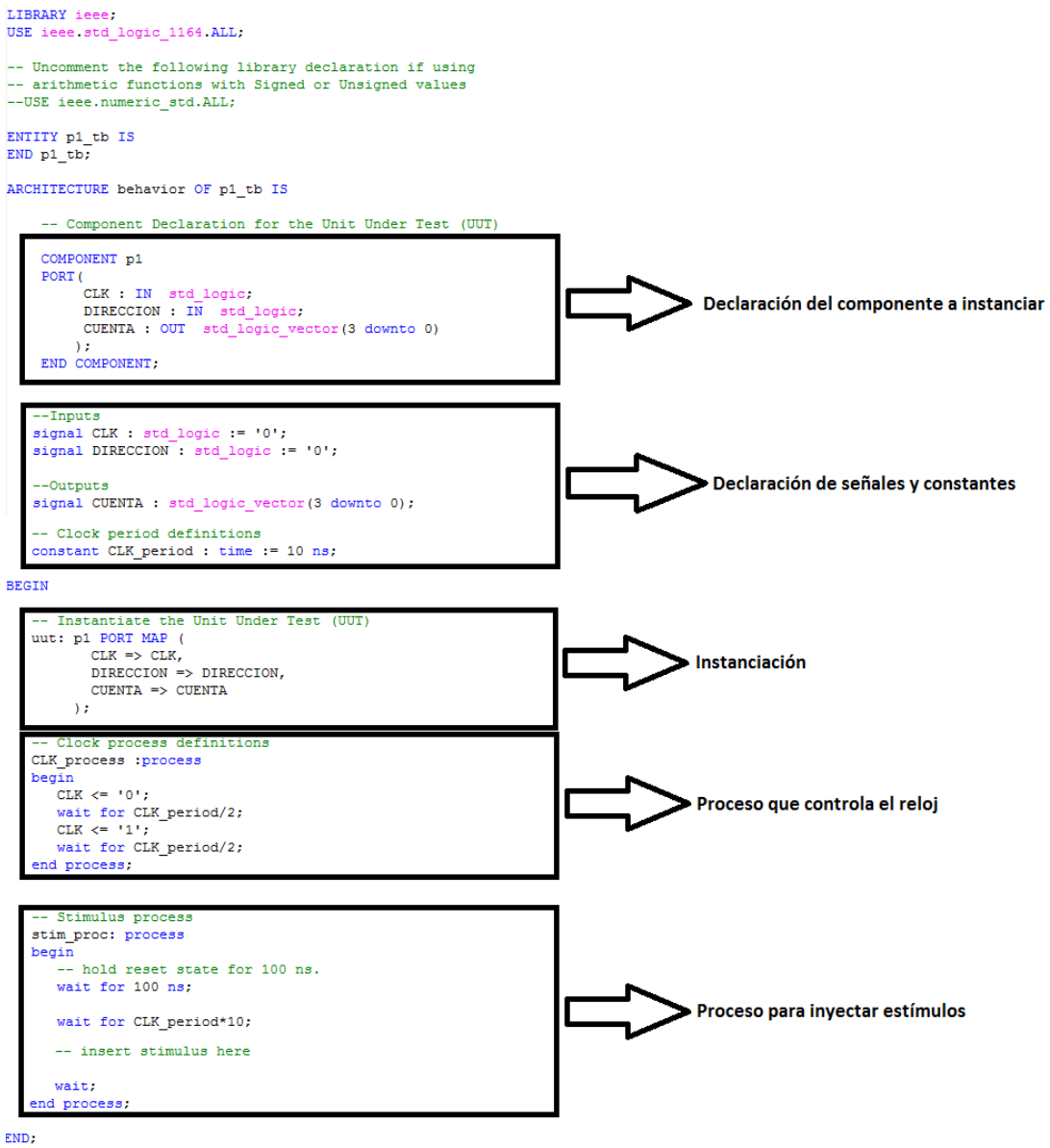
En caso de que no haya habido problemas en el proceso, en la ventana **Console** (abajo del todo) debería poner un mensaje del tipo "**Process 'Check Syntax' completed successfully**". En caso contrario, aparecerán mensajes de error o warnings.

Verificando el comportamiento del diseño: Test Benchs

Antes de hacer downloading del diseño realizado en una FPGA es aconsejable simularlo. Para ello existen los **Test Bench**, los cuales permiten crear "estímulos" de entrada para ver cómo se

comporta nuestro diseño. Para crear un **Test Bench** adecuado para esta práctica se ha de hacer lo siguiente:

1) Seleccionar el fichero VHDL con el código en la ventana **Hierarchy**.
2) Ir a **Project > New Source**, seleccionar **VHDL Test Bench**, darle un nombre (por ejemplo, 'p1_tb'), dejar marcado **Add to Project** y pulsar en **Next**, **Next** y **Finish**.
3) Aparecerá en el área dedicada a la implementación el código VHDL del **Test Bench**:



En esta captura previa del código están resaltadas cinco secciones:

- *Declaración del componente a instanciar*. Declara la interfaz del módulo VHDL del contador.
- *Declaración de señales y constantes*. Se declaran e inicializan las señales de entrada/salida del módulo. A mayores, se define una constante que indicará el

período del reloj del sistema. Por defecto, este valor se fija en 10 ns. **Modificar el período para que la frecuencia del reloj sea de 25 MHz.**

- *Instanciación*. Se crea una instancia del componente definido previamente.
- *Proceso que controla el reloj*. Controla el cambio de flanco de la señal de reloj, haciendo que éste permanezca con el valor '0' durante la mitad del ciclo y con el valor '1' la otra mitad.
- *Proceso para inyectar estímulos*.

Esta último proceso será el que modificaremos para poner a prueba el contador creado. En concreto, vamos a verificar simplemente cómo influye la señal **DIRECCION** en el conteo. Para ello, vamos a hacer que cambie a '1' cuando transcurran 300 ns y que vuelva a '0' con 900 ns. Esto se realizará modificando el proceso donde se inyectan los estímulos de la manera que se ve en la siguiente captura:

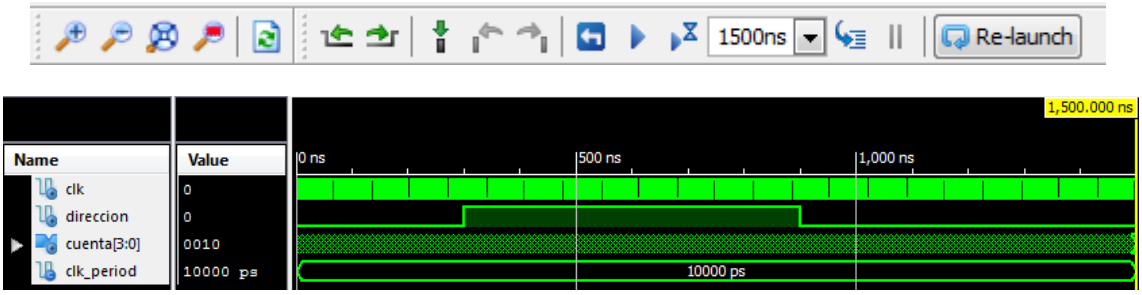
```
-- Stimulus process
stim_proc: process
begin
   -- hold reset state for 100 ns.
   wait for 300 ns;

   DIRECCION <= '1';

   wait for 600 ns;
   DIRECCION <= '0';

   wait for CLK_period*10;

   -- insert stimulus here

   wait;
end process;
```

Para verificar que el código del Test Bench se ha introducido correctamente es necesario:

1) En el área **Design**, escoger el radio button **Simulation** para el parámetro *View.*
2) Seleccionar en **Hierarchy** el fichero .vhd que contiene el código VHDL del Test Bench.
3) En Processes, dentro de **ISim Simulator**, hacer doble click sobre **Behavioral Check Syntax**.

Simulación

Si la verificación del código VHDL ha sido exitosa, podemos pasar a simular el Test Bench. Esto se realiza haciendo doble click sobre **Simulate Behavioral Model** (bajo **ISim Simulator** en la ventana **Processes**). Se abrirá el software de simulación **Isim**, pero no se verán los resultados de simulación deseados debido a la escala temporal: es necesario jugar con los botones del display y la longitud temporal para visualizar al menos los primeros 900 ns (para cambiar la escala a nanosegundos, pulsar con el botón derecho sobre la escala del display).



Haciendo zoom puede observarse como el contador, que inicialmente está a 0 y con **DIRECCION** igual a 0, empieza a disminuir de 1 en 1 con cada pulso de reloj. Al llegar a los 300 ns, **DIRECCION** pasa a valer 1, con lo que el contador empieza a subir de 1 en 1, hasta que se llega a los 900 ns, momento en que **DIRECCION** vuelve a valer 0, volviendo a decrementarse el contador de 1 en 1.