Universidad Politécnica de Madrid
ETSI de Telecomunicación
Departamento de Ingeniería Electrónica

Circuitos Electrónicos (Plan 2010)

Curso 2012-2013

Manual de referencia de la tarjeta BASYS 2

Álvaro de Guzmán Fernández

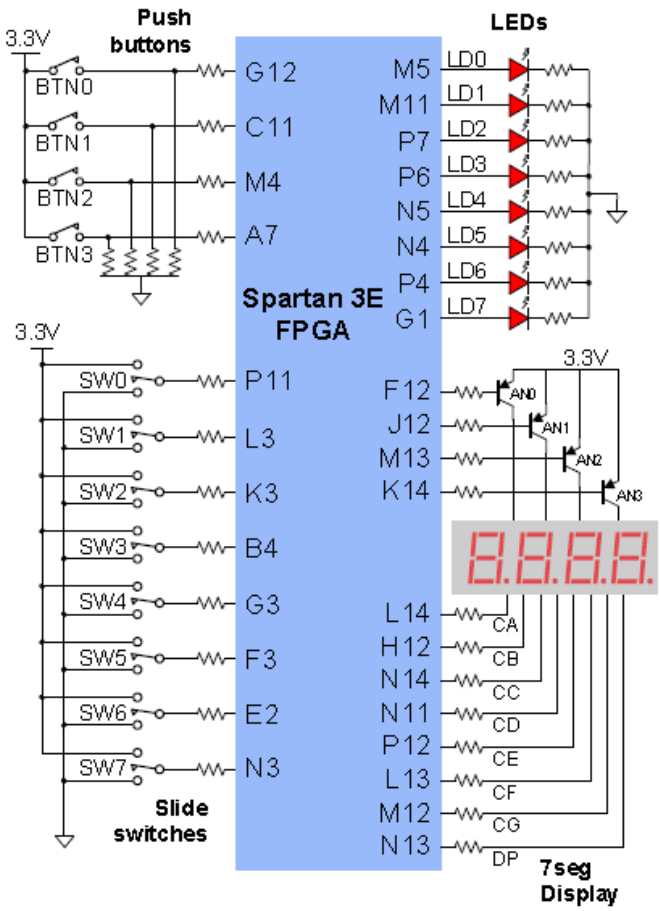
Manual de referencia de la tarjeta BASYS 2

La tarjeta BASYS 2 es una tarjeta de desarrollo que contiene una FPGA Spartan 3, fabricada por la compañía DIGILENT. Está diseñada para el aprendizaje de la síntesis de circuitos de complejidad media utilizando un entorno de desarrollo profesional.

Además de la FPGA, esta tarjeta contiene una serie de recursos que pueden ser utilizados en los diseños de los circuitos. Concretamente contiene:

- 4 pulsadores
- 8 conmutadores
- 8 LEDS
- 4 displays de 7 segmentos
- Un conector de teclado de PC
- Una salida VGA

Todos estos recursos se encuentran conectados a las patillas de la FPGA de la forma que se indica en la siguiente figura:



Además la FPGA posee una entrada conectada a un reloj de **50 MHz** que se corresponde con la patilla **B8.**

En dicha figura se han omitido intencionadamente el conector de teclado y la salida VGA por no ser necesarias para las aplicaciones que se pretenden realizar en esta asignatura.

De esta manera, el valor lógico proporcionado por el pulsador BTN0, se leerá en la patilla G12 de la FPGA. Del mismo modo, para activar el LED LD0, es necesario poner un valor lógico '1' en la patilla M5.

Además de los recursos integrados en la tarjeta, existe la posibilidad de utilizar entradas y salidas externas, las cuales también se encuentran conectadas a las patillas de la FPGA y que se encuentran disponibles en los conectores externos del entrenador presente en cada puesto del laboratorio (conectores marcados como ENTRADAS DIGITALES y SALIDAS DIGITALES). En el enunciado de cada práctica se indicarán las patillas concretas de la FPGA que se corresponden con estas entradas y salidas.

Síntesis de circuitos mediante la tarjeta BASYS 2

La síntesis de circuitos se realizará mediante el lenguaje de descripción hardware VHDL. Se compilará utilizando el entorno ISE de XILINX que se encontrará disponible en el ordenador de cada puesto del laboratorio. Este entorno es capaz de crear un archivo para la configuración de la FPGA a partir del código VHDL que se escriba (archivo de *"bit stream"* con extensión .bit). Dicho archivo debe ser cargado en la tarjeta BASYS 2. Esto hace que el hardware interno de la FPGA se configure para seguir las especificaciones hardware descritas.

Para volcar el contenido del archivo en la FPGA y configurarla es necesario utilizar el programa ADEPT de DIGILENT, el cual también estará disponible en el ordenador de cada puesto.

Por tanto, la secuencia de síntesis de un circuito deberá seguir necesariamente los siguientes pasos:

1. Escribir un código en VHDL que describa el hardware que queremos sintetizar. Recuerde que ESTAMOS SINTETIZANDO HARDWARE y que por tanto el VHDL NO ES UN LENGUAJE DE PROGRAMACIÓN, SINO UN LENGUAJE DE DESCRIPCIÓN HARDWARE.
2. Compilar dicho código y generar el archivo de "bit stream". Evidentemente el archivo no se generará si el programa tiene errores.
3. Una vez generado el fichero de "bit stream", deberá utilizarse el programa ADEPT para volcarlo en la FPGA.
4. En este momento, la FPGA se convierte en un circuito que deberá realizar la tarea que haya sido descrita mediante el VHDL. Si no se corresponde con lo esperado, deberá modificar el código y volver al punto 1 de esta secuencia.

La estructura básica de un diseño VHDL

Un diseño VHDL consiste en la especificación de un hardware concreto que se quiere sintetizar en el interior de la FPGA. Para escribir el código VHDL se utilizará el editor del entorno ISE, y serán necesarios obligatoriamente los siguientes archivos:

1. Uno o varios archivos conteniendo el código VHDL, (dependiendo de su complejidad a veces es más sencillo separar el código en varios ficheros independientes). En estos archivos se declararán entidades ("entity") cuyas entradas y salidas tendrán nombres arbitrarios elegidos por el desarrollador.
2. Un archivo de asociaciones donde se le dice al compilador qué patilla de la FPGA se corresponde con cada entrada y salida declaradas en las entidades que componen el circuito. Este fichero es IMPRESCINDIBLE para generar el "bit stream". La sintaxis empleada en este fichero es muy sencilla y se describirá más adelante.

Ejemplo 1: Encendido de un LED mediante un conmutador.

El objetivo de este ejemplo es demostrar cómo se sintetiza un circuito que sea capaz de reflejar en un LED el valor lógico proporcionado por un conmutador. Se utilizarán los recursos de la tarjeta BASYS 2.

Se realizará con dos ficheros: uno de código VHDL y otro de asociaciones.

FICHERO PRUEBA_LED.VHD (fichero de código VHDL)

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


entity circuito is
  port (
    CONMUT : in   STD_LOGIC;    -- Conmutador conectado a una entrada
    LED    : out   STD_LOGIC;   -- LED conectado a una salida
        );
end circuito;

architecture a_circuito of circuito is

LED<=CONMUT;

end a_circuito;
```

FICHERO DE ASOCIACIONES: ASOCIACIONES.UCF

```
NET "CONMUT" LOC = "P11";
NET "LED" LOC = "M5";
```

Con estos dos ficheros se puede ya compilar el código y obtener un "bit stream". En el fichero VHDL declaramos una entidad llamada "circuito" con una entrada llamada CONMUT y una salida llamada LED. En su descripción funcional, decimos que dicha entrada se corresponde con la salida (de esta manera el valor del LED variará según el valor digital del conmutador). Es por tanto un circuito combinacional. En el fichero de asociaciones indicamos cuáles son las patillas de la FPGA que se asocian físicamente con la entrada y la salida de la entidad. Fíjese

que se asocian exactamente con las patillas donde se encuentran conectados los recursos de la tarjeta BASYS 2.

Ejemplo 2: Utilización del reloj de la FPGA.

La FPGA posee una entrada conectada a un reloj de 50 MHz en la patilla B8. Este reloj puede ser utilizado para el diseño de circuitos secuenciales síncronos. En este ejemplo utilizaremos el reloj para dividir su frecuencia mediante un contador visualizando la salida en un LED.

Utilizaremos un proceso en cuya lista de sensibilidad se coloca la señal de reloj. Con cada flanco activo se incrementará un contador. Queremos que el LED parpadee con un periodo encendido/apagado de 500 ms cada uno.

Se utilizarán igualmente dos ficheros, uno VHDL y otro de asociaciones.

FICHERO DIV_RELOJ.VHD (fichero de código VHDL)

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

entity circuito is
    Port ( CLK : in  STD_LOGIC;    -- Reloj de entrada a la FPGA
           SAL : out  STD_LOGIC);  -- salida del circuito para conectar al LED
end circuito;

architecture a_circuito of circuito is
signal contador : STD_LOGIC_VECTOR (31 downto 0);
signal flag : STD_LOGIC;
begin
  PROC_CONT : process (CLK)
    begin
    if CLK'event and CLK='1' then
      contador<=contador + '1';
      if contador=25000000 then   -- el reloj tiene un periodo de 20 ns (50 MHz)
        flag<=not flag;           -- tras 25e6 cuentas habrán transcurrido 500 ms
        SAL<=flag;
        contador<=(others=>'0');
      end if;
    end if;
    end process;
end a_circuito;
```

FICHERO DE ASOCIACIONES: ASOCIACIONES.UCF

```
NET "CLK" LOC = "B8"
NET "SAL" LOC = "M5"
```

En este caso se trata de un circuito secuencial con un proceso que sincroniza el sistema.

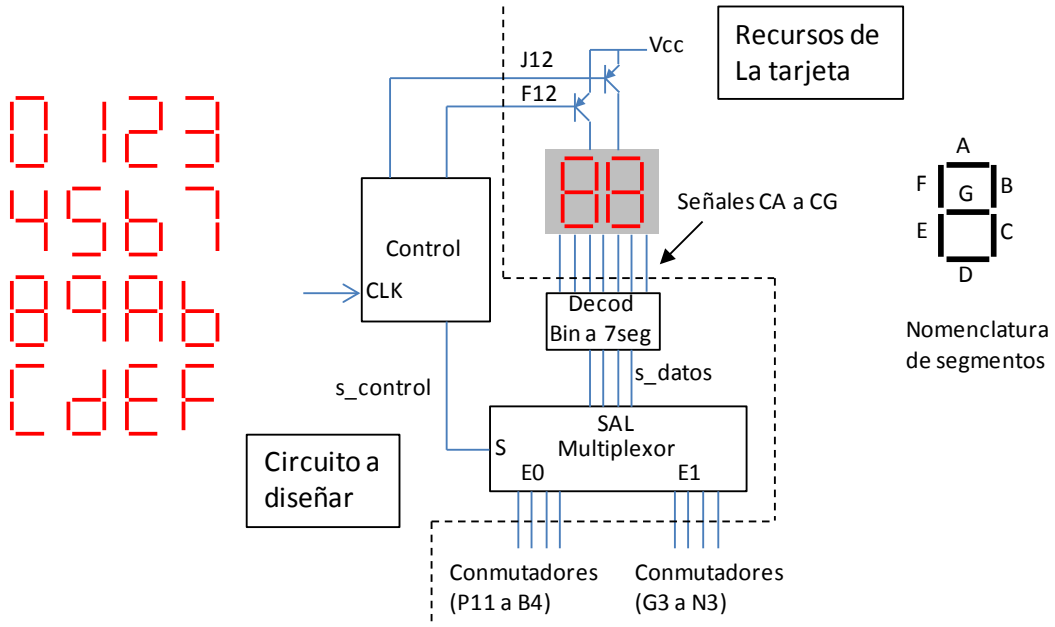
Esta es la estructura básica cuando se trata de un circuito de muy poca complejidad como los presentados en los ejemplos 1 y 2. A continuación se muestra un ejemplo más complejo con varios ficheros VHDL.

Ejemplo 3: Visualización de dos cifras diferentes en dos displays. Las cifras binarias se introducen mediante los conmutadores.

En este caso queremos visualizar dos cifras diferentes en dos de los displays disponibles en la tarjeta BASYS 2. El problema de este ejemplo reside en el hecho de que los valores binarios para excitar los distintos segmentos son compartidos por los 4 displays simultáneamente. Existen, no obstante, 4 señales (las correspondientes a las salidas F12,J12,M13 y K14) que controlan la activación independiente de cada uno de los displays. Para visualizar cifras diferentes en cada uno, es necesario activar los segmentos correspondientes a la cifra de uno de los displays junto con su señal de activación, a continuación hacer lo mismo con el siguiente y así sucesivamente. Si esta secuencia se repite más de 25 veces por segundo, el ojo no es capaz de percibir el parpadeo, pudiendo representarse varias cifras diferentes.

Para este ejemplo necesitaremos los siguientes elementos (ver figura):

- Un decodificador de binario a 7 segmentos.
- Un multiplexor de dos entradas de 4 bits para seleccionar la cifra a visualizar
- Un elemento de control que realice el refresco periódico de los displays.



Visualizaremos todos los valores entre 0000 y 1111 utilizando para ello cifras hexadecimales tal como se muestra en la figura. Utilizaremos los recursos de la tarjeta: 8 conmutadores (4 para la primera cifra y 4 para la segunda), dos displays y el reloj para controlar la visualización.

En este caso, la manera más apropiada de describir este hardware consiste en realizar un fichero independiente por cada elemento, con descripciones funcionales de dichos elementos y un fichero maestro con una descripción estructural de interconexión entre ellos. Además también es necesario escribir el fichero de asociaciones.

El ejemplo constaría por tanto de 4 ficheros VHDL más uno de asociaciones.

FICHERO decod7s.vhd

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity decod7s is
    Port ( D : in  STD_LOGIC_VECTOR (3 downto 0); -- entrada de datos en binario
           S : out  STD_LOGIC_VECTOR (0 to 6));   -- salidas para los segmentos
end decod7s;

architecture a_decod7s of decod7s is

begin

with D select S <=
"0000001" when "0000",
"1001111" when "0001",
"0010010" when "0010",
"0000110" when "0011",
"1001100" when "0100",
"0100100" when "0101",
"1100000" when "0110",
"0001111" when "0111",
"0000000" when "1000",
"0001100" when "1001",
"0001000" when "1010",
"1100000" when "1011",
"0110001" when "1100",
"1000010" when "1101",
"0110000" when "1110",
"0111000" when "1111",
"1111111" when others;

end a_decod7s;
```

Este primer fichero contiene la descripción funcional de un decodificador de binario a 7 segmentos. Debe tenerse en cuenta que según las conexiones de la figura los segmentos se encienden con un “0” en la patilla correspondiente.

FICHERO MUX.vhd

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

entity MUX is
    Port ( E0 : in  STD_LOGIC_VECTOR (3 downto 0);  -- Entrada de datos 0
           E1 : in  STD_LOGIC_VECTOR (3 downto 0);  -- Entrada de datos 1
           S : in  STD_LOGIC;                       -- Entrada de control
           SAL : out  STD_LOGIC_VECTOR (3 downto 0)); -- Salida
end MUX;

architecture a_MUX of MUX is

begin

with S select SAL<=
E0 when '0',
E1 when '1',
E0 when others;

end a_MUX;
```

Este fichero describe un multiplexor de 2 entradas de 4 bits y una salida. La señal de control es S.

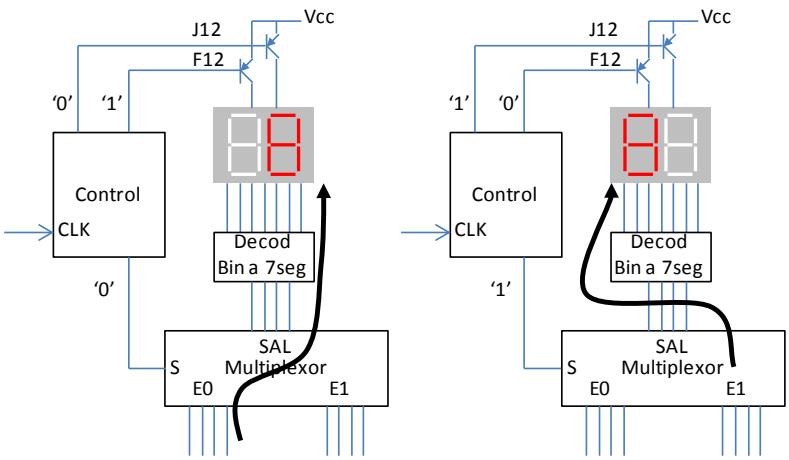
FICHERO control.vhd

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

entity control is
    Port ( CLK : in  STD_LOGIC;          -- reloj de la FPGA
           DSP0 : out  STD_LOGIC;        -- señal para activar el display 0
           DSP1 : out  STD_LOGIC;        -- señal para activar el display 1
           SCONTROL : out  STD_LOGIC);   -- señal para controlar el mux.
end control;

architecture a_control of control is
signal contador : STD_LOGIC_VECTOR (31 downto 0);
signal salida : STD_LOGIC;
begin

process (CLK)
begin
if CLK'event and CLK='1' then    -- el periodo del reloj es de 20 ns
  contador<=contador + '1';      -- por tanto 500.000 cuentas se corresponden
  if contador=500000 then        -- con el transcurso de 10 ms.
    salida<=not salida;
        contador<=(others=>'0');
  end if;
end if;
end process;

SCONTROL<=salida;
DSP0<=salida;
DSP1<=not salida;

end a_control;
```

Este fichero describe un circuito secuencial de control. Un proceso divide la frecuencia del reloj utilizando un contador. Cada 10 ms invierte el valor lógico de la señal ‘salida’.

En paralelo con este proceso la salida SCONTROL toma el valor de la señal ‘salida’, mientras que las salidas DSP0 y DSP1 toman valores contrarios dependiendo de la señal ‘salida’.

Con este código se describe un elemento que alterna el valor de SCONTROL cada 10 ms. Al mismo tiempo invierte los valores de DSP0 y DSP1 que son las señales que activarán los displays. De esta manera uno de ellos está activo cuando se selecciona una entrada del multiplexor. El otro se activa cuando se selecciona la otra entrada. (Tenga en cuenta que según la figura, los displays se activan cuando la señal de control DSP correspondiente es ‘0’).



El circuito de control alterna
estas dos configuraciones cada 10 ms

FICHERO circuito.vhd

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

entity circuito is
    Port ( CIFRA0 : in  STD_LOGIC_VECTOR (3 downto 0); -- entrada de cifra 0
           CIFRA1 : in  STD_LOGIC_VECTOR (3 downto 0); -- entrada de cifra 1
           SEG7 : out  STD_LOGIC_VECTOR (0 to 6);  -- salidas para los displays
           DISP1 : out  STD_LOGIC;                 -- salida para activar display 1
           DISP2 : out  STD_LOGIC;                 -- salida para activar display 2
           CLK : in  STD_LOGIC);                   -- reloj de la FPGA
end circuito;

architecture a_circuito of circuito is             -- en este caso se trata de una

component decod7s                                  -- descripción estructural
  Port ( D : in  STD_LOGIC_VECTOR (3 downto 0);
           S : out  STD_LOGIC_VECTOR (0 to 6));
end component;

component MUX
  Port ( E0 : in  STD_LOGIC_VECTOR (3 downto 0);
           E1 : in  STD_LOGIC_VECTOR (3 downto 0);
           S : in  STD_LOGIC;
           SAL : out  STD_LOGIC_VECTOR (3 downto 0));
end component;

component control
  Port ( CLK : in  STD_LOGIC;
           DSP0 : out  STD_LOGIC;
           DSP1 : out  STD_LOGIC;
           SCONTROL : out  STD_LOGIC);
end component;

signal s_control : STD_LOGIC;
signal s_datos   : STD_LOGIC_VECTOR (3 downto 0);

begin

U1 : control
  port map (
           CLK => CLK,
                       DSP0 => DISP1,
                       DSP1 => DISP2,
                       SCONTROL => s_control
           );

U2 : MUX
  port map (
           E0 => CIFRA0,
                       E1 => CIFRA1,
                       S => s_control,
                       SAL => s_datos
           );

U3 : decod7s
  port map (
           D => s_datos,
                       S => SEG7
           );

end a_circuito;
```

Este es el fichero maestro que describe la interconexión entre los distintos elementos. Se describe un circuito principal cuyas entradas son las dos cifras binarias de 4 bits y la señal de reloj. Sus salidas son las señales de activación de los displays, y los valores de los segmentos.

Cuando una entrada o salida de un componente se cablea directamente a una entrada o salida del circuito principal, se asigna directamente en el “port map”. Si se trata de interconexiones

entre elementos entonces deben utilizarse señales adicionales declaradas también en el fichero.

FICHERO DE ASOCIACIONES

```
NET "CLK" LOC = "B8"; # RELOJ DE LA FPGA

NET "SEG7<0>" LOC = "L14"; #Segmento a
NET "SEG7<1>" LOC = "H12"; #Segmento b
NET "SEG7<2>" LOC = "N14"; #Segmento c
NET "SEG7<3>" LOC = "N11"; #Segmento d
NET "SEG7<4>" LOC = "P12"; #Segmento e
NET "SEG7<5>" LOC = "L13"; #Segmento f
NET "SEG7<6>" LOC = "M12"; #Segmento g

NET "CIFRA0<0>" LOC = "P11"; # CONMUTADOR 0
NET "CIFRA0<1>" LOC = "L3";
NET "CIFRA0<2>" LOC = "K3";
NET "CIFRA0<3>" LOC = "B4";  # CONMUTADOR 3

NET "CIFRA1<0>" LOC = "G3"; # CONMUTADOR 4
NET "CIFRA1<1>" LOC = "F3";
NET "CIFRA1<2>" LOC = "E2";
NET "CIFRA1<3>" LOC = "N3";  # CONMUTADOR 7

NET "DISP1" LOC = "F12"; # ACTIVACIÓN DISP 1
NET "DISP2" LOC = "J12"; # ACTIVACIÓN DISP 2
```

Por último, las entradas y salidas del circuito principal deben asociarse con los recursos de la tarjeta. Esto se hace mediante el correspondiente fichero de asociaciones que se muestra. La nomenclatura SEG7<N> se corresponde con el bit N de la salida SEG7. Lo mismo sucede con CIFRA0 y CIFRA1.

Con este circuito se representan las cifras hexadecimales correspondientes a los conmutadores en los dos displays de la derecha. Sin embargo los otros dos presentan también cifras como consecuencia de no estar desactivados. Un posible ejercicio consiste en modificar el archivo de descripción estructural (cricuito.vhd) y el fichero de asociaciones para que se mantengan desactivados los dos displays no utilizados.

Utilización del entorno ISE

Para realizar una síntesis mediante el entorno ISE debe crearse un proyecto. El proyecto incluirá todos los ficheros correspondientes al diseño incluido el fichero de asociaciones.

Vaya al menú FILE -> NEW PROJECT. Aparecerá una ventana donde tendrá que escribir el nombre del nuevo proyecto. El entorno creará una carpeta dentro de la ubicación "Location" que aparece en dicha ventana. En dicha carpeta almacenará toda la información del proyecto, incluidos los ficheros VHDL, el fichero de asociaciones y el "bit stream" una vez obtenido. Asegúrese también que en la casilla de selección "Top level Source Type" aparece "HDL".

En la siguiente pantalla deberá ajustar los siguientes valores:

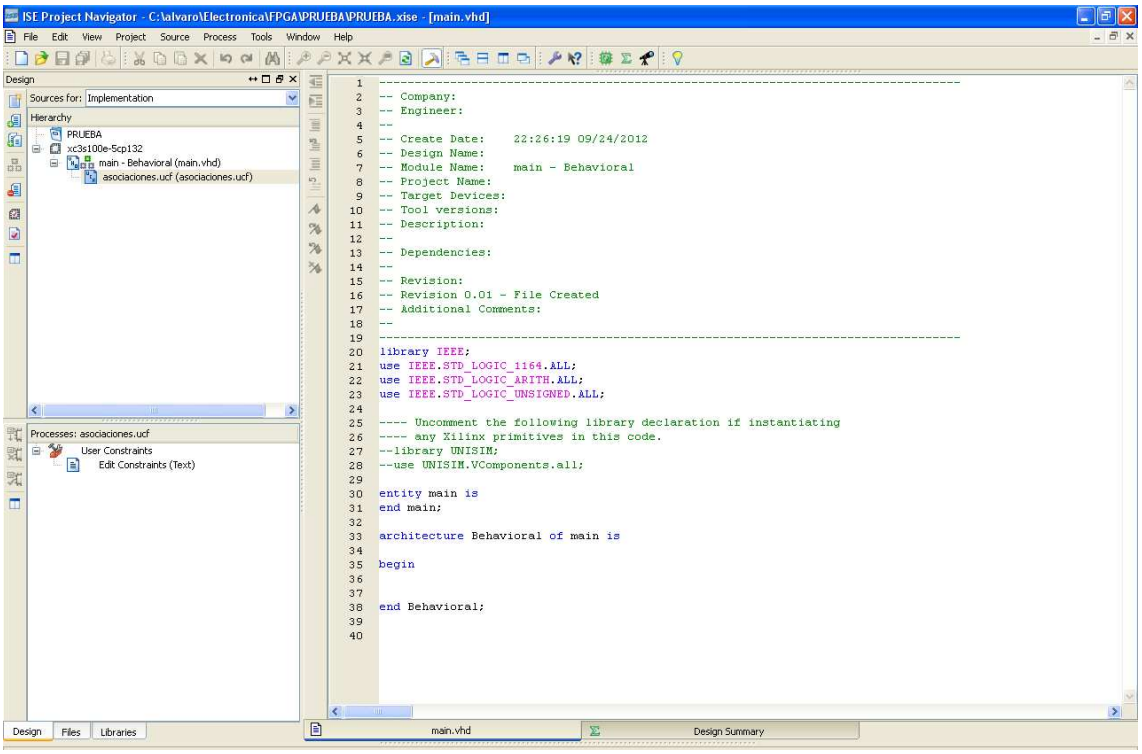
Family: **Spartan 3E**
Device: **XC3S100E**
Package: **CP132**
Speed: **-5**
Synthesis tool: **XST (VHDL/Verilog)**
Simulator: **ISim (VHDL/Verilog)**
Preferred Language: **VHDL**

Por último sáltese las siguientes pantallas pulsando en "Next". En la última aparecerá el botón "Finish".

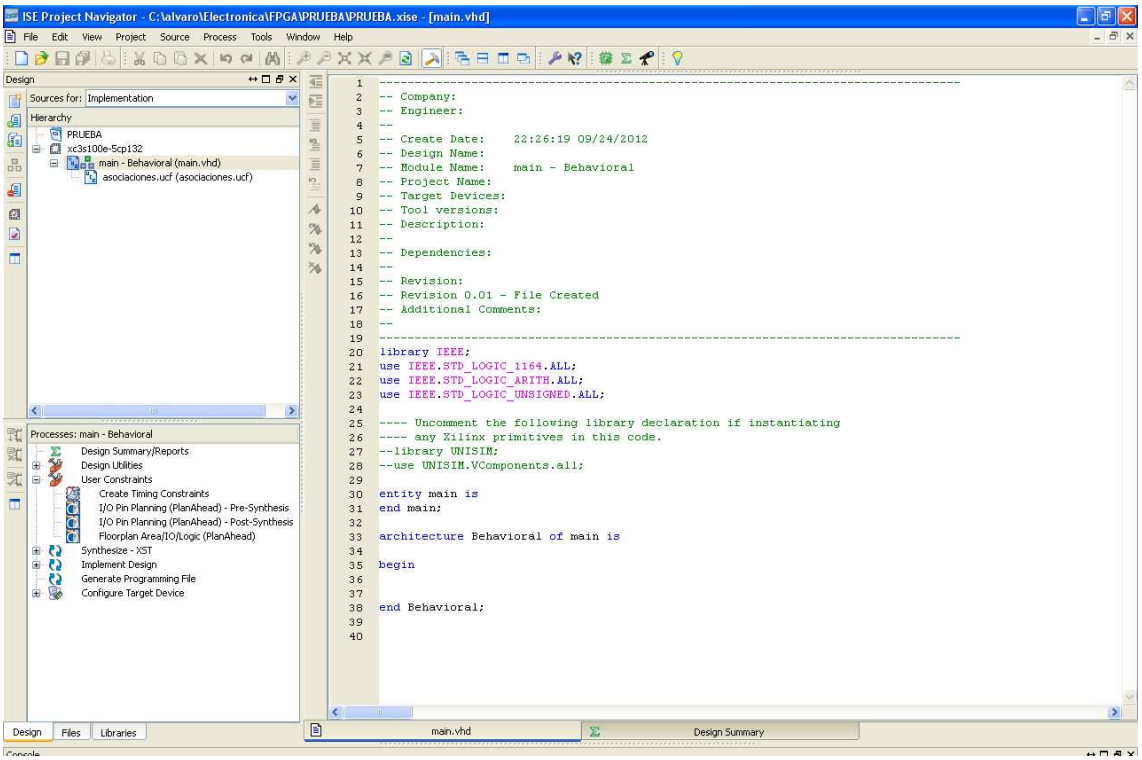
Una vez creado el proyecto pueden añadirse ficheros nuevos pulsando con el botón derecho del ratón sobre el icono "xc3s100e-5cp132" que aparece en la parte izquierda de la pantalla. En el menú que se despliega seleccione "New Source".

Elija VHDL module para crear un nuevo archivo VHDL. Escriba un nombre para el archivo y pase a la siguiente pantalla. En ella puede declarar las entradas y las salidas del módulo (sección port dentro de la declaración entity). Si lo desea puede pasar esta pantalla y declarar las entradas y salidas manualmente más tarde. Tras esta pantalla aparecerá un editor con el esqueleto principal del código VHDL ya escrito. Puede escribir en el archivo y grabar los cambios con el icono de un diskette en la parte superior de la pantalla.

Para el archivo de asociaciones elija "Implementation Constraints File" y escriba un nombre. Se creará un archivo con extensión UCF. Para acceder a él, selecciónelo en la parte superior izquierda de la pantalla, luego pulse con el ratón el icono "Edit Constraints (Text)" en la parte inferior izquierda (ver figura).



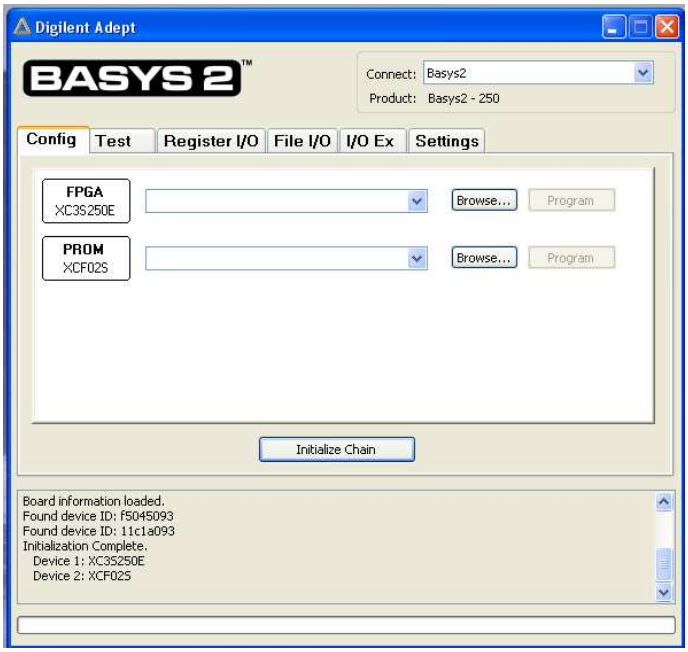
Para compilar el código y generar el “bit stream” , seleccione el fichero principal del proyecto y haga doble click en “Generate Programming File” en la parte inferior izquierda. Vea la figura siguiente:



Evidentemente tendrá que depurar todos los errores existentes en el código hasta que se pueda generar un fichero “bit stream”. En el momento en que todo esté correcto aparecerá un icono verde a la izquierda de la etiqueta “Generate Programming File”.

Volcado del “bit stream” sobre la FPGA para sintetizar el circuito

Para ello necesita el programa ADEPT de DIGILENT. Arranque dicho programa:



Mediante el botón “Browse…” situado en la línea superior puede seleccionar el archivo que desea cargar. El archivo debe tener extensión .bit

A continuación haga click en el botón “Program”. Acepte la ventana que aparece y el fichero será transferido a la FPGA configurándola de tal manera que siga el circuito descrito en su diseño.