Práctica 1 – Introducción al diseño, simulación e implementación de circuitos mediante VHDL

El objetivo de esta práctica es familiarizarse con el entorno Vivado de Xilinx, la placa de prototipado Nexys-4 de Digilent, e iniciarse en el desarrollo de circuitos utilizando el lenguaje de descripción de hardware VHDL.

El material necesario para la realización de la práctica será el entorno Vivado versión 2016.4 y las placas de prototipado Digilent Nexys-4, todo ello disponible en el laboratorio de Fundamentos de Computadores (edificio 1G, 2S-18).

Se pretende desarrollar un semisumador y visualizar sus salidas en dos de los leds disponibles en la placa de prototipado.

1. Creación de un nuevo proyecto

El primer paso a realizar, como en cualquier entorno de desarrollo, consiste en crear un proyecto que albergue los diferentes ficheros que utilizaremos para describir el diseño hardware que deseamos implementar. Para ello, seleccionaremos la opción ***New Project*** del menú ***File***, o el icono ***Create New Project*** de ***Quick Start***.

El diálogo que aparece permitirá, en primer lugar, configurar el nombre del proyecto, dónde localizar los diferentes ficheros de los que constará el mismo, y el tipo de proyecto que deseamos crear (***RTL Project*** en nuestro caso). Ignoraremos la incorporación de los ficheros fuente, que crearemos más adelante (marcar ***Do not specify sources at this time***, o ignorar ***Add Sources***, ***Add Existing IP*** y ***Add Constraints***).

El siguiente formulario del diálogo, que se muestra en la Figura 1, precisa de información relativa al dispositivo concreto que se utilizará para realizar la implementación del diseño. Esta información no es crítica si simplemente se desea realizar una simulación comportamental del diseño, pero es fundamental si el objetivo final es realizar la implementación del mismo.

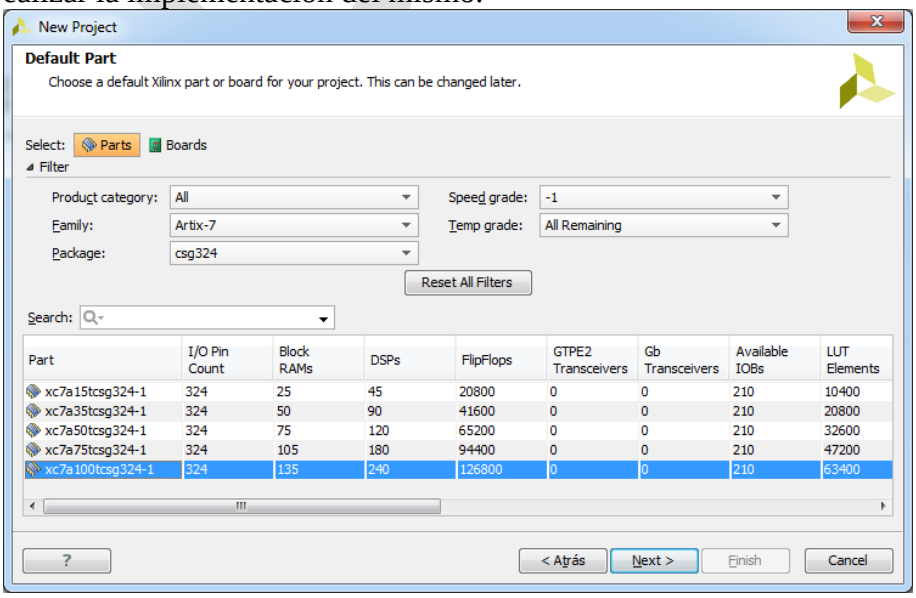

Figura 1. Selección del dispositivo en el que se implementará el diseño.

Tal y como puede apreciarse en la Figura 2, es posible obtener esta información observando la serigrafía que muestra el encapsulado de la FPGA seleccionada. La información correspondiente a la FPGA disponible en la placa de prototipado es: Familia – *Artix-7*, Dispositivo – *XC7A1000T*, Encapsulado – *CSG324* y Velocidad – *-1*.

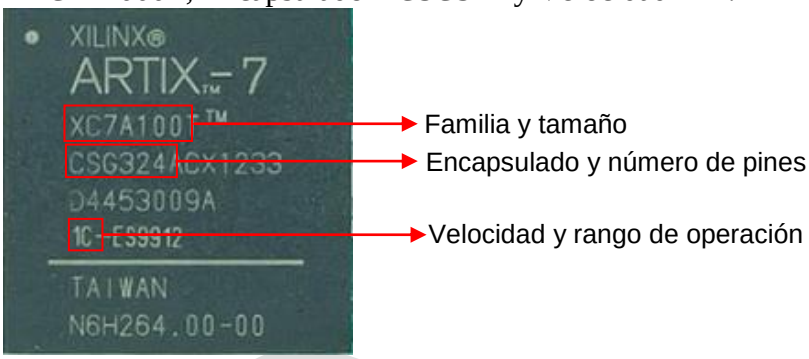

Figura 2. Características de la FPGA de Xilinx listadas en su encapsulado.

Al finalizar este proceso se generará una carpeta en el directorio seleccionado con el nombre del proyecto indicado.

2. La interfaz principal de Vivado

Básicamente Vivado es un framework que dispone de una interfaz gráfica principal para acceder a las funcionalidades proporcionadas por las diferentes herramientas integradas. Esta interfaz simplifica la interacción del usuario con las herramientas, ya que genera los scripts y/o comandos necesarios para la ejecución de las mismas de acuerdo a las preferencias establecidas en el proyecto. Es posible utilizar estas herramientas a través de línea de comandos y, en algunos casos, es la única manera de acceder a determinadas funcionalidades (más información en el documento *ug835 – Vivado Design Suite Tcl Command Reference Guide*).

La configuración básica de la interfaz, accesible desde la opción ***Default Layout*** del menú ***Layout***), se muestra en la Figura 3. Los principales componentes identificados en la figura son:

1. Barra de menú
2. Barra de herramientas principal
3. Navegador de flujo
4. Selector de distribución de componentes
5. Área de ventana de datos
6. Espacio de trabajo
7. Campo de búsqueda del menú de órdenes
8. Barra de estado del proyecto
9. Barra de estado
10. Área de ventanas de resultados

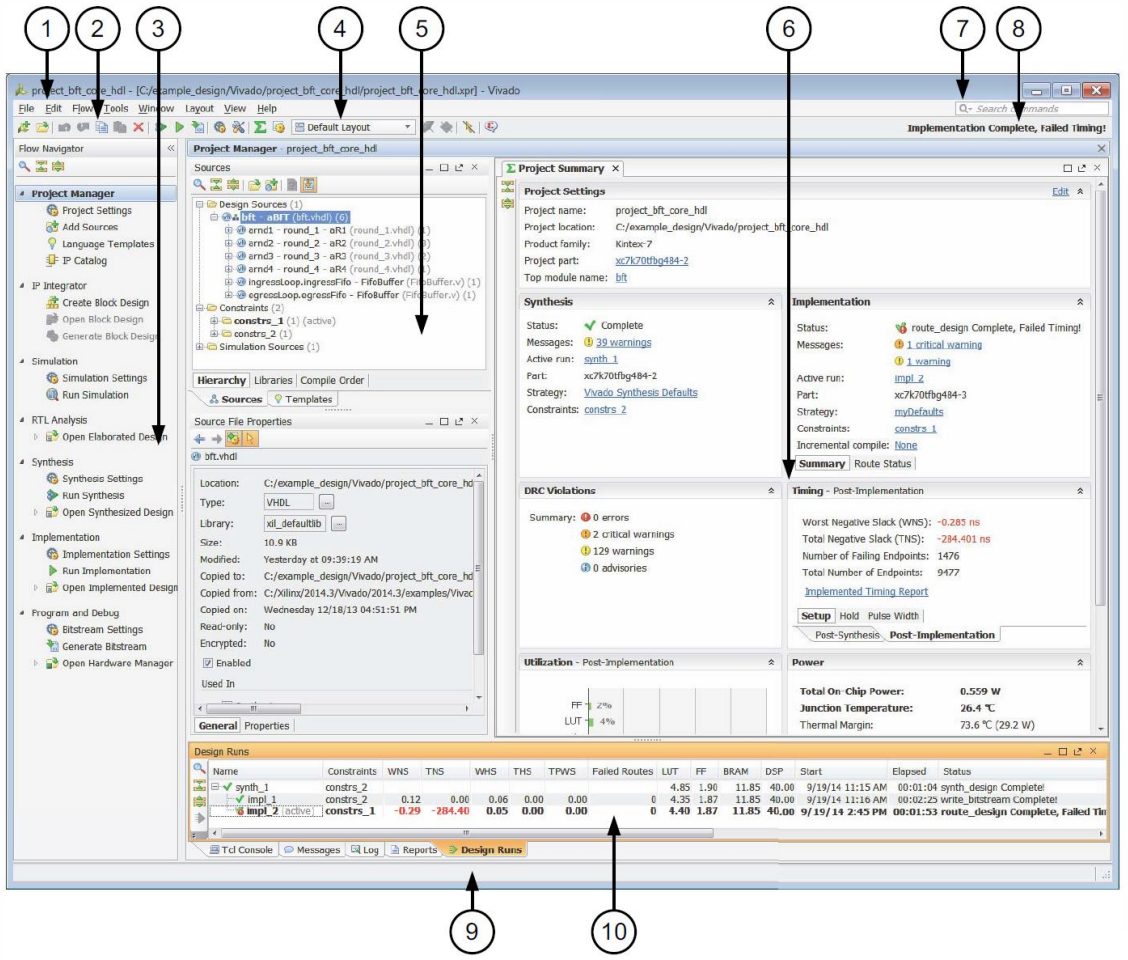

Figura 3. Layout por defecto de Vivado.

3. Incorporación de componentes al diseño

Para incorporar nuevos componentes a un diseño podemos utilizar diferentes alternativas: la opción ***Add Sources*** disponible bajo ***Project Manager*** en el ***Flow Navigator***, pulsando el botón ***Add Sources*** en la ventana ***Sources*** del área de ventana de datos, o pulsando con el botón derecho sobre la carpeta ***Design Sources*** del área de ventana de datos y seleccionando ***Add Sources*** (Alt + A).

Esta opción permite crear un nuevo elemento o añadir uno ya existente. Diversos tipos de elementos pueden incluirse en el diseño, como ficheros de restricciones, fuentes para el diseño o simulación, entre otros.

Como ejemplo de prueba, crearemos un nuevo componente (***Add or create design sources***) que modele el funcionamiento de un *semisumador*. En el asistente que aparecerá, indicaremos que el tipo de fichero a crear es *VHDL*, y le asignaremos como nombre *HalfAdder*.

El siguiente asistente, que puede verse en la Figura 4, permite indicar el nombre del componente (es recomendable que el fichero y el componente tengan el *mismo nombre*), la etiqueta asociada a la arquitectura del componente, y la lista de puertos de entrada/salida del componente (en caso de tratarse de buses debe indicarse el número asociado a los bits de mayor/menor peso). Todos los puertos generados serán de tipo *std_logic*/*std_logic_vector*. Este diálogo generará un fichero de texto plano, con extensión *.vhd*, por lo que no hay ningún problema en editarlo posteriormente.

En este caso, definiremos los dos puertos de entrada (dato *a_i* y dato *b_i*) y los dos puertos de salida (suma *s_o* y acarreo *c_o*) correspondientes a un semisumador.

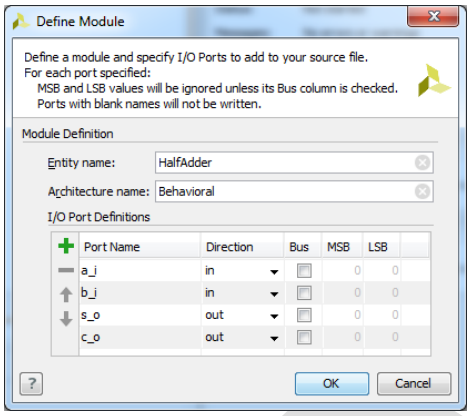

Figura 4. Declaración de a interfaz del nuevo componente.

4. Edición de componentes

Al crear un nuevo componente VHDL, el editor de texto disponible en Vivado abre automáticamente el fichero correspondiente a este componente para proceder a la definición de su arquitectura. Haciendo *doble click* sobre los diversos componentes del diseño mostrados en la pestaña ***Hierarchy*** se permite su edición.

Existen múltiples maneras de describir el funcionamiento/estructura de un semisumador. A modo ilustrativo, a continuación se detalla su tabla de verdad (Tabla 1), a partir de la cual se deducen sus ecuaciones (Ecuación 1) y estructura (Figura 6).

Tabla 1. Tabla de verdad de un semisumador.

a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$s = a \oplus b$$
$$c = a \cdot b$$

Ecuación 1. Funciones lógicas de un semisumador.

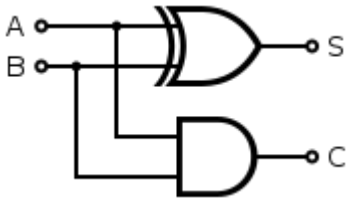

Figura 5. Diseño estructural de un semisumador.

Utilizando el editor de texto integrado en Vivado editaremos el componente recién creado para especificar el funcionamiento del semisumador. Un ejemplo, dado que puede realizarse de múltiples formas diferentes, se muestra en el Código 1.

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

entity HalfAdder is
    Port ( a_i : in  STD_LOGIC;     -- first operand
           b_i : in  STD_LOGIC;     -- second operand
           s_o : out  STD_LOGIC;    -- sum
           c_o : out  STD_LOGIC);   -- carry
end HalfAdder;

architecture Behavioral of HalfAdder is
```

```
begin

      s_o <= a_i xor b_i;   -- a + b
      c_o <= a_i and b_i;   -- carry out

end Behavioral;
```

Código 1. Descripción de flujo de datos de un semisumador.

La sintaxis del código se irá comprobando conforme se modela el componente. En la pestaña ***Messages*** del área de ventanas de resultados podremos comprobar los diferentes mensajes de error que pudieran aparecer. Es conveniente recordar que el proceso se ejecutará sobre el elemento que se tenga seleccionado en la pestaña ***Hierarchy***, no sobre el que se esté editando en ese momento.

5. Simulación de componentes

Para validar la descripción del componente recién creado, es necesario realizar una simulación del mismo. Para ello, se precisa de un fichero adicional (*testbench*) encargado de proporcionar al modelo del componente los estímulos requeridos para activar sus entradas y poder comprobar los valores generados en sus salidas. Este fichero se creará como un nuevo elemento de diseño de tipo simulación (***Add or create simulation sources***). Este fichero será también de tipo VHDL y le asociaremos un nombre descriptivo como *HalfAdder_tb*. Al tratarse de un fichero de simulación, su interfaz no dispone de ningún puerto asociado, por lo que se dejará en blanco cuando aparezca el asistente correspondiente. Este nuevo elemento deberá ser editado para incluir la declaración de la interfaz del componente a simular, e introducir los estímulos correspondientes a las diferentes combinaciones de entrada para poder validar que el sistema se comporta de la forma esperada. Esto se consigue asignando valores a las señales que se han conectado a los puertos de entrada del componente (ver Código 2) y dejando transcurrir un tiempo prudencial (dependerá del diseño) antes de modificar dichos valores. Un ejemplo de un posible fichero de estímulos se muestra en Código 2.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY HalfAdder_tb IS
END HalfAdder_tb;

ARCHITECTURE Behavioral OF HalfAdder_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT HalfAdder
    PORT(
         a_i : IN  STD_LOGIC;
         b_i : IN  STD_LOGIC;
         s_o : OUT  STD_LOGIC;
         c_o : OUT  STD_LOGIC
        );
    END COMPONENT;
```

```
    --Inputs
    signal a_i, b_i : STD_LOGIC := '0';
    --Outputs
    signal s_o, c_o : STD_LOGIC;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: HalfAdder
    PORT MAP (
        a_i => a_i,
        b_i => b_i,
        s_o => s_o,
        c_o => c_o
    );

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

        -- a= '0', b = '0'
        wait for 10 ns;

        a_i <= '1'; -- a = '1', b = '0'
        wait for 10 ns;

        b_i <= '1'; -- a = '1', b = '1'
        wait for 10 ns;

        a_i <= '0'; -- a = '0', b = '1'
        wait for 10 ns;

        wait; -- end simulation

    end process;

END Behavioral;
```

Código 2. Fichero de estímulos para la simulación de un semisumador.

Los procesos relacionados con la simulación de componentes se encuentran bajo ***Simulation*** en el ***Flow Navigator***. En estos momentos, debido a que todavía no se ha comenzado el proceso de implementación del componente, la única opción disponible al pulsar ***Run Simulation*** será ***Run Behavioral Simulation***. El resultado de esta simulación puede verse en la Figura 6.

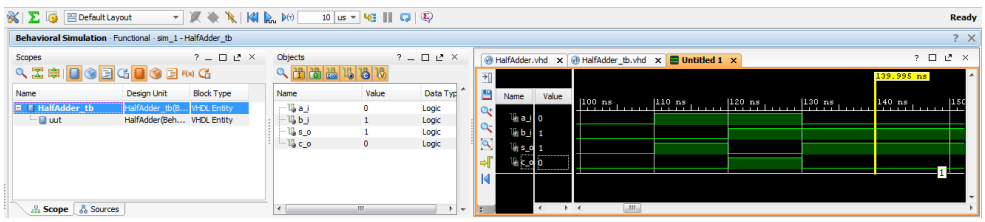

Figura 6. Simulación comportamental de un semisumador.

Si el resultado de la simulación es correcto, entonces puede procederse a continuar con el proceso de implementación del diseño sobre la placa de prototipado. En caso contrario, es necesario modificar el modelo realizado para corregir los posibles problemas detectados.

6. Implementación del diseño

A continuación se presentarán los pasos que deben seguirse para implementar el componente diseñado en la placa de prototipado disponible, de una manera rápida y sencilla. El detalle de las diversas herramientas, procesos y opciones de configuración disponibles se irán descubriendo poco a poco a lo largo del curso.

6.1. Síntesis del diseño

El primer paso hacia la implementación del diseño, conocido como síntesis (***Synthsesis***), extrae del modelo aquellos componentes que el sintetizador reconoce como bloques o macros (sumadores, RAM, o máquinas de estado, entre otros) y que pueden generar una implementación más eficiente. El resto de lógica se tratará como *glue logic* (pequeña lógica sin función clara identificada que permite la conexión entre los grandes bloques bien definidos).

Para su ejecución, simplemente es necesario hacer click sobre el proceso ***Run Synthesis*** bajo ***Synthesis*** en el ***Flow Navigator***.

Los informes generados en cada uno de los procesos pueden consultarse en cualquier momento desde la pestaña ***Reports***.

6.2. Especificación de restricciones de asignación de pines

En este momento se dispone del modelo del componente que se desea implementar. Sin embargo, el entorno de desarrollo dispone de libertad completa para tomar las decisiones que estime oportunas a la hora de realizar esta implementación. Por ejemplo, la FPGA seleccionada dispone de hasta 210 pines que pueden utilizarse como entrada/salida. Así pues, ¿cuáles de todos estos pines deben utilizarse como entrada/salida para el diseño realizado? El diseñador debe suministrar esta información al entorno de desarrollo para realizar correctamente esta asignación de pines.

En la página 18 del manual de referencia de la placa de desarrollo (*Nexys4 DDR FPGA Board Reference Manual*) se listan los diferentes elementos que conforman la interfaz de entrada/salida básica de la FPGA disponible en la placa. Entre los elementos de entrada se encuentran *5 botones* y *16 interruptores*, mientras que los elementos de salida los constituyen *16 leds*, *2 leds tricolor* y *8 displays de 7 segmentos*. Para introducir datos en el semisumador a implementar en la FPGA podrían utilizarse 2 interruptores (a_i y b_i) y las salidas podrían visualizarse por medio de 2 leds (s_o y c_o). Por ejemplo, los interruptores ***SW0*** y ***SW1*** están conectados a los pines ***J15*** y ***L16***, respectivamente, mientras que los leds ***LD0*** y ***LD1*** están conectados a los pines ***H17*** y ***K15***, respectivamente.

Para indicar qué pines se desean conectar a qué puertos, es preciso crear un nuevo tipo de elemento en el diseño conocido como fichero de restricciones (***Xilinx Design Constraints***). Existen varias maneras de realizar este proceso.

A la primera de ellas se accede pulsando sobre ***Open Synthesized Design*** bajo ***Synthesis*** en el ***Flow Navigator***. Una vez abierto el diseño sintetizado, debemos cambiar la vista a ***I/O Planning***, con lo que se mostrará la distribución de pines en el encapsulado del dispositivo (ver Figura 7). Desde la pestaña ***I/O Ports*** es posible asignar los pines del diseño arrastrándolos y dejándolos caer (*drag & drop*) sobre los representados en la interfaz gráfica (muy poco práctico), o editando el campo ***Package Pin*** disponible (el campo ***I/O Std*** debe fijarse a *LVCMOS33*).

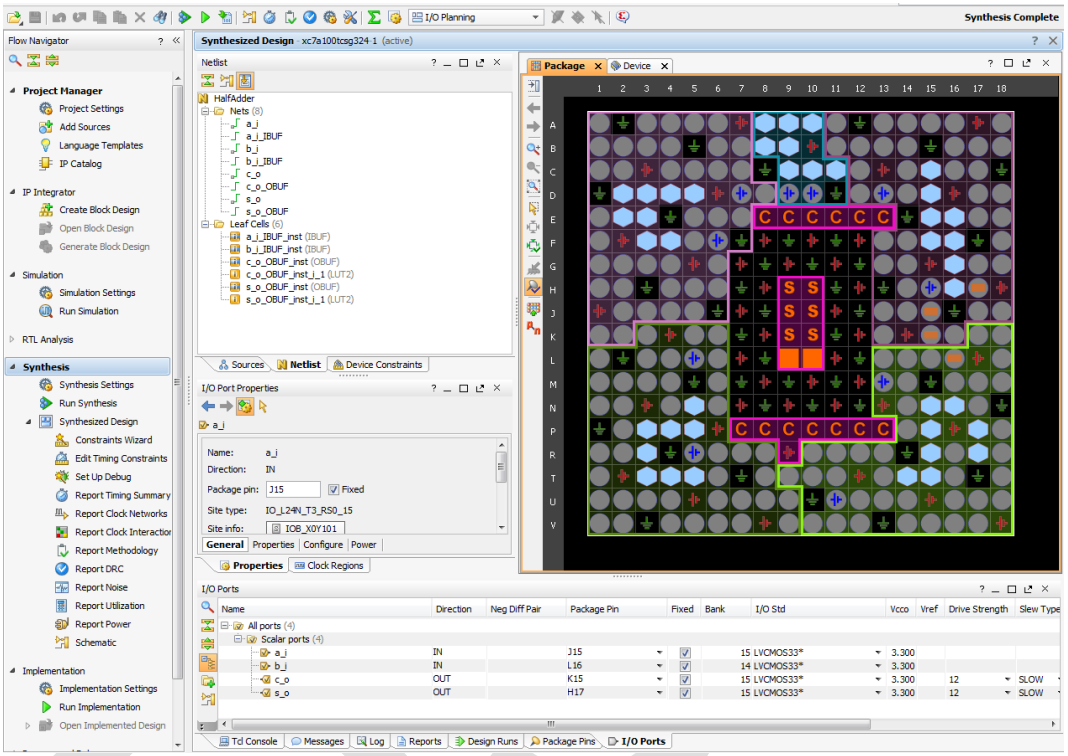

Figura 7. Asignación de pines desde la interfaz gráfica.

Esto creará un nuevo fichero de restricciones (extensión *.xdc*) con las localizaciones indicadas y se asociará al diseño. Este fichero de texto plano puede visualizarse desde el ***Project Manager***.

La segunda forma de introducir estas restricciones, y mucho más cómoda y rápida una vez se conoce el formato, consiste en crear directamente un nuevo elemento de tipo restricciones, y editarlo directamente con el editor de texto existente. El fabricante de la placa de prototipado proporciona un fichero con las restricciones necesarias para utilizar todos los pines de entrada/salida de usuario en la FPGA disponible en la placa (*Nexys4DDR_Master.xdc*), por lo que es muy simple asociar cualquiera de los bits de entrada/salida a los pines disponibles.

El fichero resultante de asignar los pines seleccionados se lista en el Código 3.

```
#Switches
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports a_i]
set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports b_i]

#Leds
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports s_o]
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports c_o]
```

Código 3. Fichero de restricciones de asignación de pines.

6.3. Implementación

Los procesos involucrados en el proceso de implementación propiamente dicho se encargan de asociar el resultado de la síntesis con las restricciones especificadas, distribuir todos los elementos requeridos entre los componentes hardware disponibles en el dispositivo programable, y realizar su interconexión. Todo ello se lanza a ejecución pulsando ***Run Implementation*** bajo ***Implementation*** en el ***Flow Navigator***.

6.4. Verificación de los resultados de la implementación

Además de revisar los informes que devuelven cada uno de los procesos involucrados en la implementación del diseño, es conveniente verificar estos resultados por medio de simulaciones. Por ello, es posible simular el modelo del sistema correspondiente a cada una de sus fases de implementación: ***Run Behavioral Simulation***, ***Run Post-Synthesis Functional Simulation***, ***Run Post-Synthesis Timing Simulation***, ***Run Post-Implementation Functional Simulation*** y ***Run Post-Implementation Timing Simulation***.

6.5. Generación del fichero de programación

El paso final del proceso consiste en generar el fichero de configuración que se descargará al dispositivo para programar sus bloques lógicos e interconexiones para que se comporte como el diseño descrito en el modelo.

Para ello es necesario pulsar sobre ***Generate Bitstream*** bajo ***Program and Debug*** en el ***Flow Navigator***. Esto generará un fichero con extensión *.bit*, denominado *bistream*, que representa el estado de todos los elementos programables del dispositivo.

7. Verificación sobre el dispositivo

Ya solo resta descargar el fichero de configuración sobre el dispositivo FPGA disponible en la placa de desarrollo para validar su correcto funcionamiento.

Primeramente es necesario conectar la placa al equipo de desarrollo utilizando el cable USB disponible en la caja (conector ***PROG***), y activar el interruptor de alimentación (***POWER*** – ON).

A continuación, se pulsará ***Open Hardware Manager*** bajo ***Program and Debug*** en el ***Flow Navigator***. Esto simplemente ejecutará el gestor de hardware (dispositivos conectados), y se procederá a pulsar sobre ***Open Target*** y seleccionar ***Auto Connect*** para leer las características del dispositivo conectado a través del cable USB. A continuación podremos seleccionar y descargar el fichero de configuración deseado por medio de ***Program Device***.

Una vez terminada correctamente la configuración, se encenderá el led ***DONE*** en la placa, con lo que se podrá proceder a verificar el correcto funcionamiento del diseño. Puede borrarse la configuración descargada en todo momento pulsando el botón ***PROG*** de la placa. Puede descargarse una nueva configuración en todo momento sin necesidad de borrar la anterior, que será sobrescrita.

8. Ejercicio: Display 7 segmentos

Se pide diseñar un componente que permita visualizar un dato hexadecimal (4 bits) en los displays de 7 segmentos disponibles en la placa de prototipado (consultar páginas 18-19 del manual de referencia).

Para ello, se utilizarán 4 interruptores como dato de entrada (hexadecimal – *0..9A..F*, y 8 de los interruptores restantes como señal de habilitación para los displays, de tal forma que se muestre el número introducido en todos aquellos displays que tengan su señal de habilitación activa (ver ejemplo en Figura 8).

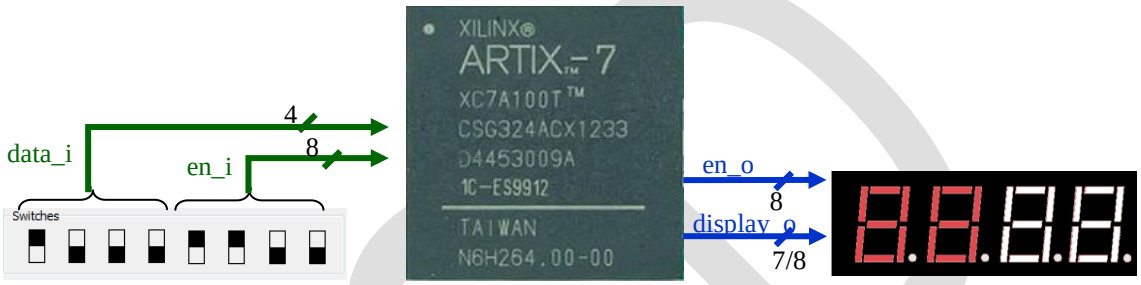

Figura 8. Ejemplo de utilización del sistema a desarrollar.