Tutorial de Xilinx ISE

Esta sección describe el entorno ISE del fabricante XILINX. Este entorno, entre otras características, incluye un simulador para el lenguaje Verilog que será utilizado en varias sesiones de laboratorio de las asignaturas CED y EdC.

1. Creación de un proyecto en Xilinx ISE

Tras iniciarse el sistema operativo, el primer paso es arrancar el entorno ISE y crear un nuevo proyecto. En el menú File hay que utilizar la opción New Project, obteniéndose la ventana mostrada en la figura 1 donde hay que escribir un nombre para el proyecto, por ejemplo *PracticaEdC2*. La herramienta creará una carpeta con ese mismo nombre y guardará en su interior todo lo que se va gene- rando a medida que vamos trabajando en ese proyecto.

Tras escribir el nombre se activa el botón Next y aparece el cuadro de diálogo mostrado en la figura 4, donde hay que establecer todas las opciones indicadas en la figura. Concretamente hay que asegurarse de establecer las siguientes opciones a su valor correcto:

- **Family:** Spartan 3E
- **Device: XC3S100E**
- **Package:** CP132
- **Speed: -4**
- **Preferred Language**: Verilog

El resto de opciones deberían estar por defecto a los mismos valores que los mostrados en la figura 2. Tras establecer las opciones correctas, utilizando el botón Next aparece una última ventana con información y un botón Finish que se pulsa para crear el proyecto. La figura 3 muestra el proyecto recién creado, sólo se muestra el nombre del proyecto y el tipo de FPGA "**xc3s100e-cp132**".

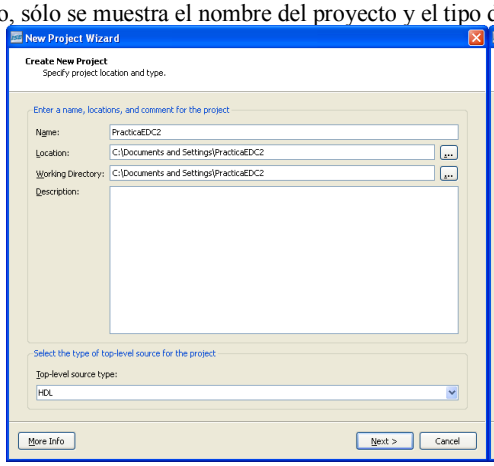


Figura 1. Ventana de creación del proyecto.

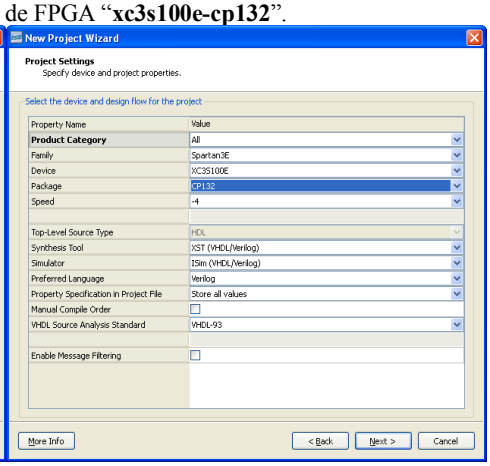


Figura 2. Ventana de propiedades del proyecto.

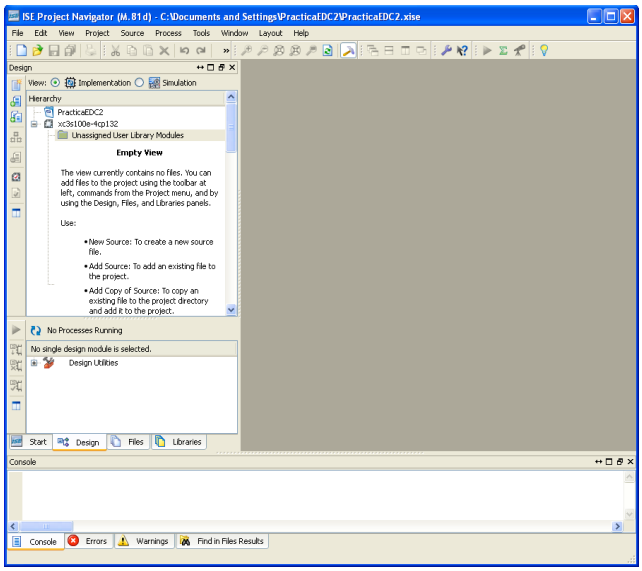


Figura 3. Vista general de un proyecto en el entorno Xilinx.

Nótese que encima del nombre del proyecto aparecen dos iconos *Implementation* y *Simulation*, (ver figura 3) cada uno con distintas vistas del mismo proyecto. Debemos procurar estar siempre en modo de **simulación** para evitar problemas con el entorno ISE. También se recomienda utilizar la entrada de menú Layout → Restore Default Layout en caso de no ver correctamente las ventanas o los controles de *ISE Project Navigator* (o del entorno de Simulación *Isim* que usará más adelante*)*.

2. *Añadir ficheros al proyecto*

El primer paso tras la creación del proyecto es añadir los ficheros Verilog (diseños y testbenchs) al proyecto. Para añadir ficheros al proyecto se puede puede utilizar la opción de menú Project o, pulsar el botón derecho del ratón en la zona en blanco de la vista del proyecto, eligiendo entre:

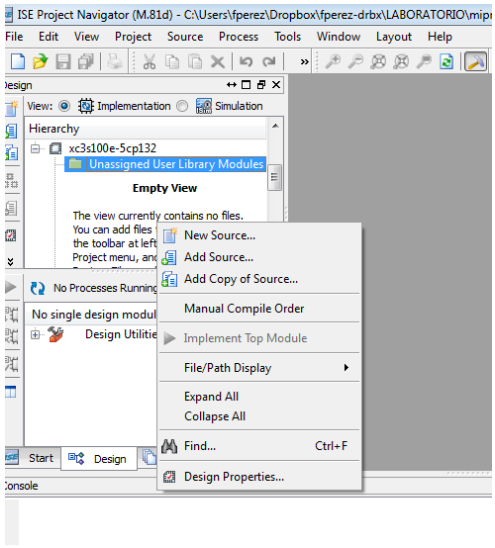


Figura 4. Incorporando ficheros al proyecto.

- New Source crea un nuevo fichero vacío en el que habrá que escribir el código del diseño.
- Add Copy of Source crea un nuevo fichero dentro del proyecto que inicialmente será una copia del fichero fuente seleccionado, previamente creado. Así las modificaciones serán propias a este proyecto y el fichero fuente original no se modificará. La copia residirá dentro de la carpeta del proyecto.
- Add Source no crea un nuevo fichero dentro del proyecto, utiliza el propio fichero fuente seleccionado sin crear ninguna copia. Así las modificaciones afectarán al fichero fuente en su ubicación original, es decir, el fichero residirá en la misma carpeta dónde esté almacenado previamente.

Si partimos de un proyecto en blanco, elegiremos New Source, y seleccionamos el tipo de fichero que queremos crear:

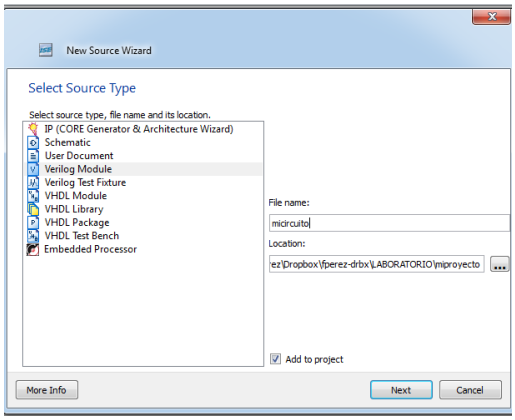


Figura 5.Elección del nuevo fichero a crear .

Normalmente elegiremos Verilog Module (módulo Verilog que especifica un circuito) o Verilog Test (fichero testbench de simulación de un módulo).

Si hemos pedido la creación de un módulo verilog, la herramienta nos solicita los terminales (ports) del nuevo módulo, debiendo indicar si son entradas o salidas, y si son líneas individuales o buses.

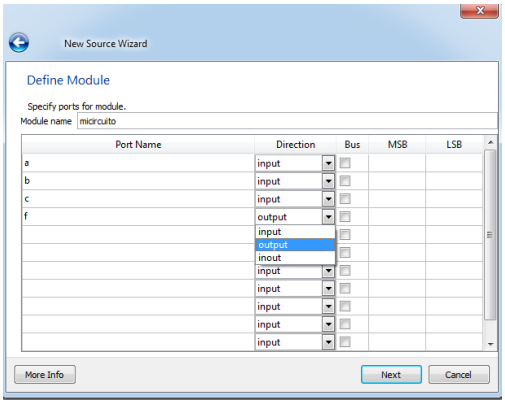


Figura 6. Elección del nuevo fichero a crear .

Después de una pantalla resumen, la aplicación abre la ventana de jerarquía del proyecto, donde aparecerá el nuevo módulo y el esqueleto de su especificación verilog, que deberemos completar:

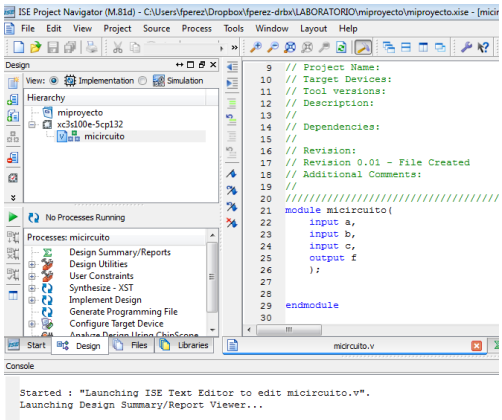


Figura 7. Módulo creado e incorporado al proyecto

Podemos incorporar también ficheros previamente creados, en este caso, elegimos la opción Add Source seleccionando uno o más ficheros a añadir al proyecto. Hay que confirmar los ficheros que son para implementación y cuales son exclusivamente para simulación (cómo los ficheros de *testbench* suministrados). En la figura 6 se muestran un ejemplo donde se solicita añadir varios ficheros así como la asociación realizada en cada uno de ellos para que la simulación opere correctamente (elija *All* y *Simulation* según se indica).

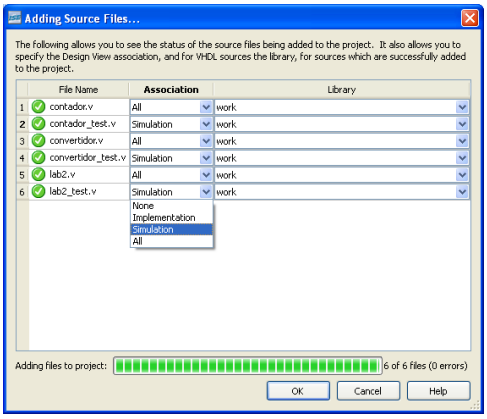


Figura 8. Ventana de inclusión de ficheros al proyecto.

Una vez añadidos los ficheros, éstos son mostrados en la vista del proyecto de forma jerárquica y, componiendo el árbol de proyecto de ficheros en función de que un fichero necesite de otro fichero, esto se puede visualizar en la figura 6. El fichero que incluye a los demás será el primer fichero del árbol de proyecto.

Para editar o ver cualquiera de los ficheros del proyecto, sólo hay que seleccionarlo con el ratón en el árbol de proyecto y pulsar el botón izquierdo del ratón dos veces.

3. Simulación y verificación de un diseño

La simulación nos permite verificar el correcto funcionamiento de una unidad/módulo diseñado, para los casos que se plantean en el fichero de testbench. Éstos podrán ser más o menos completos según el caso y si encontramos algún problema, nos permite indagar la causa del mismo antes de pasar a modificar el código. Efectuando correcciones y simulaciones se consigue solucionar todos los problemas que pueda tener el diseño.

Tras añadir un fichero de testbench, éste no se muestra en la vista de *implementación*, únicamente aparece en la vista *simulación*. Esto es debido a que dichos ficheros contienen información para realizar una simulación/verificación del diseño lógico, pero no se usa para realizar una implementación de la unidad (la última etapa del proceso de diseño). También puede comprobar como en la vista de simulación aparecen las unidades en un orden jerárquico distinto al de implementación.
En la siguiente figura se muestra un ejemplo de varios módulos de un proyecto, con sus módulos de test asociados.
.

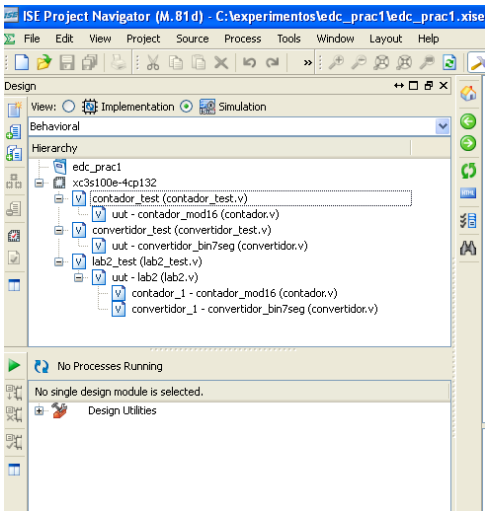


Figura 9. Vista jerárquica de un proyecto

Así, para simular una unidad debemos resaltar el nombre de la unidad/fichero de testbench a simular y abajo en la caja titulada *Processes* desplegar la entrada *ISim Simulator* para ejecutar la orden *Simulate Behavioral Model* pulsando el ratón dos veces, como muestra la figura 10.

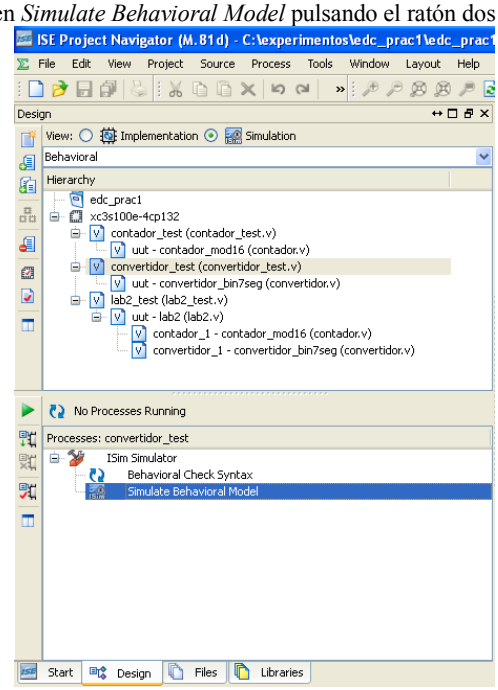


Figura 10. Selección de simulación para el convertidor binario 7 segmentos.

Si no hay errores en el diseño, se abrirá una nueva ventana con el simulador *ISim* y ejecutará una simulación durante un corto periodo de tiempo (habitualmente 1s), deteniéndose la simulación en ese punto, salvo que el testbench detenga la simulación con antelación (con la orden *$finish*).

Si no hay errores en el código se abrirá el simulador *ISim* donde, para ver las formas de onda, hay que utilizar la pestaña DEFAULT.WCFG . Es aconsejable utilizar en este momento el icono (ZOOM TO FULL VIEW) para tener una vista completa de toda la simulación.

En esta vista, mostrada en la figura 11, se dispone de una ventana con las formas de onda a la derecha en fondo negro y varias señales representadas con sus valores simulados, que deben ser las entradas y salidas de nuestra unidad (al menos aquellas que aparecen en el testbench). Si hacemos pulsamos el ratón sobre el gráfico de formas de ondas, se sitúa un cursor amarillo indicando datos exactos en ese instante de tiempo (nótese como cambian los valores de las señales al situarse en distintos instantes). Para las señales de varios bits podemos cambiar la codificación pulsando el botón derecho del ratón sobre el nombre de la señal y, accediendo en el menú flotante a la opción *RADIX* (binario, hexadecimal, decimal, etc.).

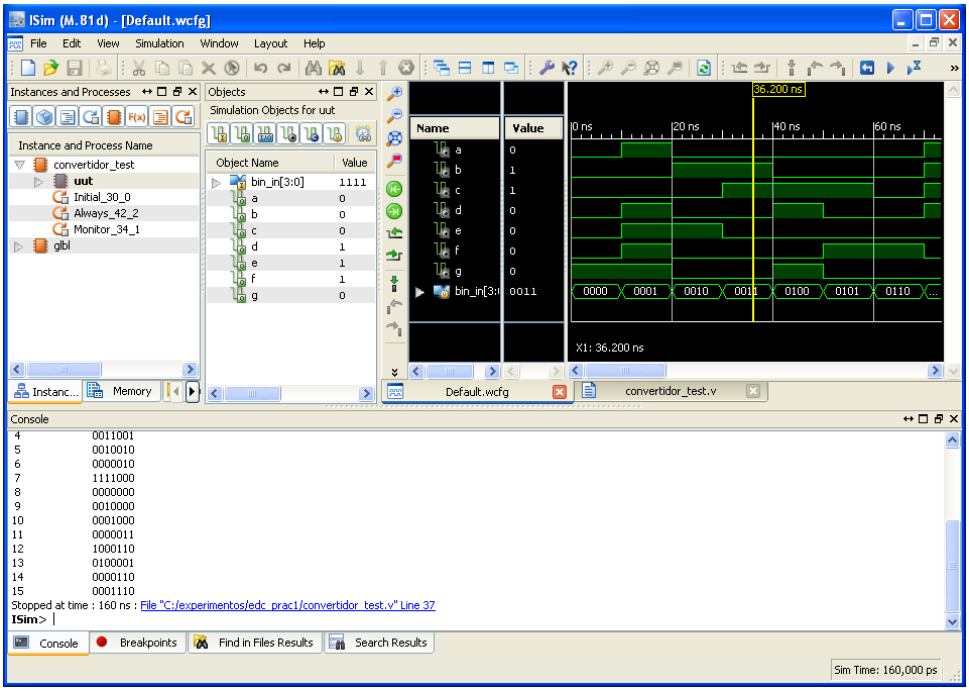


Figura 11. Simulación de un convertidor binario 7 segmentos con ISim.

Otros controles importantes de *ISim* se encuentra en dos bloques situados a la izquierda con los que se puede navegar por todas las unidades que componen el diseño. Utilizando estos dos bloques se pueden buscar señales y componentes internos de cualquier módulo para poder mostrar sus formas de ondas, pero, para ello habría que reiniciar la simulación tras añadirlas a la vista de simulación.

Por último, en la barra de iconos superior hay varios iconos que permiten hacer ampliaciones y reducciones de escala en la imagen. Además de estos iconos, varios iconos verdes que hay a continuación sirven para navegar por la forma de onda y, los iconos azules, controlan la simulación utilizándose para continuar o detener el proceso.

Los iconos verdes se deben usar para buscar o centrarse en una parte concreta de las formas de onda, por ejemplo, *Previous Transition* y *Next Transition* nos permiten ir al anterior/siguiente flanco de una señal seleccionada previamente en la ventana de formas de onda.
Los iconos azules controlan el flujo de ejecución de la simulación permitiendo: borrar la simulación actual volviendo al instante cero (*Restart*), continuar la simulación indefinidamente (*Run All*), continuar un tiempo de simulación determinado deteniéndose automáticamente (*Run*), ejecutar la simulación línea a línea de Verilog (*Step*) y, por último, detener una simulación en ejecución (*Break*).

4. Implementación del sistema digital en la placa de FPGA Basys2

Repasemos brevemente el proceso que hemos seguido hasta ahora:

- hemos especificado nuestro diseño en verilog y hemos depurado errores
- lo hemos simulado con un fichero testbench y hemos comprobando que funciona según nuestra especificación.

Finalmente se implementará el sistema digital realizado en un dispositivo programable tipo FPGA incluida en la placa de desarrollo Basys2. Esta placa tiene como elemento fundamental un dispositivo programable FPGA (sobre el que vamos a volcar nuestro diseño lógico), pero también incluye dispositivos fisicos que podemos conectar a la FPGA, como periféricos de entrada o salida (pulsadores, interruptores, LEDs, displays, etc.)

Para ello es necesario incluir en nuestro proyecto otro fichero donde se indican las conexiones entre las entradas y salidas del módulo top y los pads del chip FPGA que van conectados a los componentes de la placa. Este fichero se llama *basys2.ucf* y ya contiene la asignación de las conexiones del sistema a los componentes de la placa.

Los pasos a seguir son los siguientes:

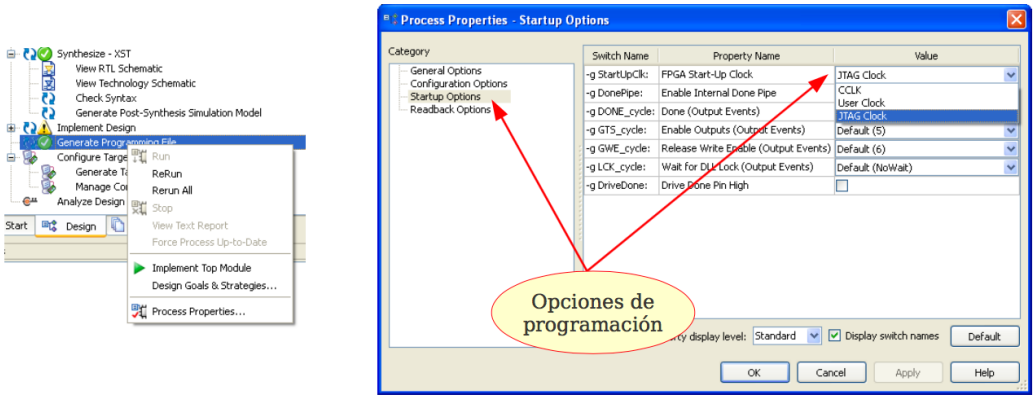
- Cambiar el proyecto ISE del modo simulación al modo implementación.
- Añadir el fichero de asociaciones *basys2.ucf* y modificarlo para especificar adecuadamente los terminales del circuito especificado a los componentes de la placa de la FPGA (ver apéndice de este documento).
- Generar el fichero de programación del proyecto
- Programar la placa FPGA con el programa Adept

La implementación se realiza seleccionando el módulo principal de nuestro diseño en el árbol de proyecto.

Debe seleccionar la opción Generate Programming File (figura 12a) y seguir los siguientes pasos:

- Pulsando el botón derecho del ratón sobre la opción Generate Programming File aparecerá un menú flotante como el mostrado en la figura 12a, seleccionando la opción de menú de Process Poperties aparecerá el diálogo mostrado en la figura 12b.
- En el diálogo hay que elegir la categoría Startup Options y cambiar el valor del primer campo FPGA Start-Up Clock de *CCLK* a *Jtag-Clock*. Tras aceptar los cambios con el botón *OK* se volverá a la ventana principal de ISE.

- Pulsando dos veces botón izquierdo del ratón sobre Generate Programming File se ejecuta el proceso completo y se genera el fichero de programación. Tal y como se muestra en la figura 11, si el proceso ha terminado con éxito aparecerá un indicador verde, en caso contrario un aspa rojo indicando la existencia de errores. En caso de existir errores debe corregirlos y repetir el proceso.



(a) *(b)*

Figura 12. Opciones de generación del fichero de programación: (a) menú desplegable, (b) diálogo con opciones

- El último paso consiste en programar la placa de desarrollo con un fichero que se ha generado tras el proceso del paso anterior. Concretamente, en este proceso debe haberse generado un fichero de programación de nuestro proyecto con extensión *.bit*, que debe ser transferirlo por la conexión USB a la FPGA. Para ello siga los siguientes pasos:

1. Compruebe en la placa el modo de programación establecido, para ello fíjese en la figura 7 donde está situado el conmutador *Modo de Programación*. Asegúrese que la placa está configurada en modo *PC (jumper* azul en la parte superior derecha de la placa*)*.
2. Conecte el puerto USB de la placa de desarrollo y conmute la alimentación de la placa.
3. Inicie el programa Adept desde el menú de inicio (menú Digilent → Adept, icono Adept) y aparecerá el programa mostrado en la figura 13. Con este programa se puede transferir el fichero de programación (*bitstream*) a la FPGA. El programa *Adept* permite programar los componentes de la placa Basys2, estos son, una FPGA y una PROM. Se programará la FPGA, por tanto, se debe utilizar el botón Browse indicado en la figura y seleccionar el fichero *.bit*. (Hay buscar la carpeta del proyecto ISE en el que está trabajando, allí encontrará el resultado de la síntesis en un fichero con extensión "**.bit**") - Una vez seleccionado este fichero se activará el botón Program *y* bastará con pulsarlo para la placa se programe y tengamos nuestro sistema funcionando.

Figura 13. Programación de placas Digilent con Adept.

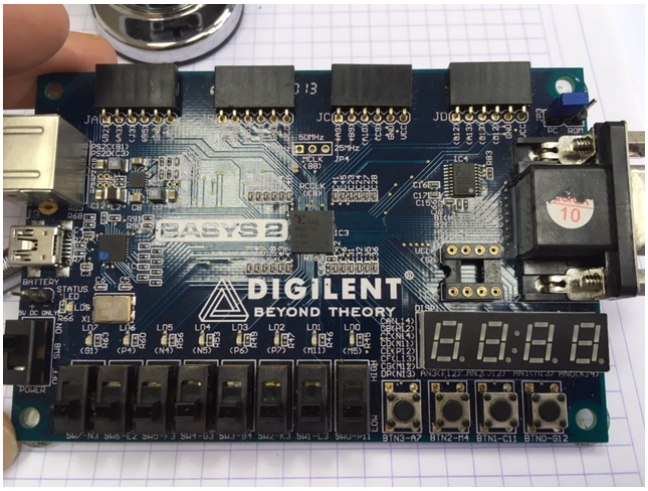
NOTA:
La FPGA es un dispositivo lógico programable volátil, por lo que se pierde la programación de la misma (y por tanto, nuestra implementación), cuando se retira la alimentación de la placa. Si quiere comprobarlo, apague y encienda la placa y podrá observar cómo nuestro circuito deja de funcionar.

No obstante, tenemos la posibilidad de guardar el diseño en una PROM de la placa (memoria ROM programable no volátil) para que, en caso de pérdida de alimentación, pueda restablecerse el diseño de la programación de la FPGA. Para ello:

1. Estando la placa en modo PC (jumper azul de la placa en la posición izquierda), seleccione el fichero de programación con extensión **".bit"** y programe la PROM, al igual que hemos hecho con la FPGA.
2. Cambie el jumper azul a modo PROM.

3. Compruebe que apagando y encendiendo la placa, nuestra implementación sigue funcionando.

APÉNDICE: <u>Arquitectura de la placa BASYS2 y fichero de asociaciones UCF</u>



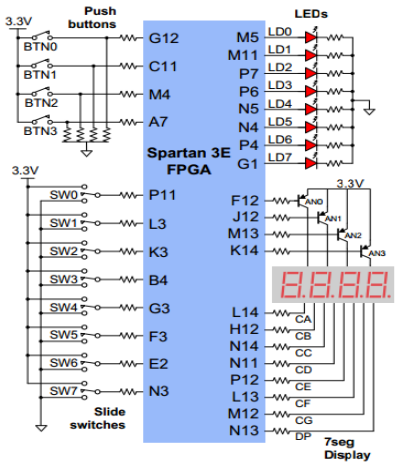


Figure 6. Basys2 I/O circuits

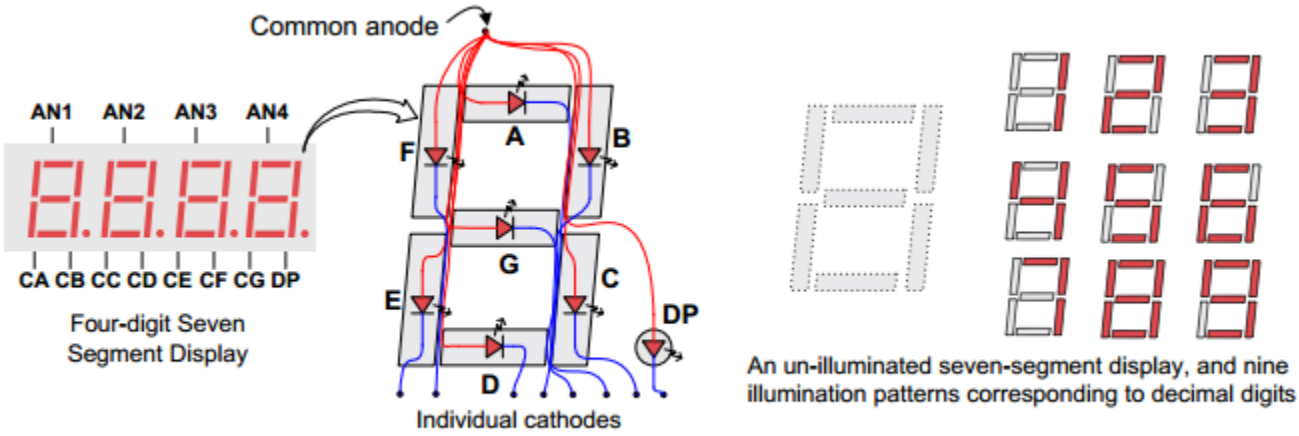


Figure 7. Seven-segment display

El fabricante de la placa, suministra un fichero de asociaciones "basys2.ucf" que tendremos que modificar para asignar las entradas y salida de nuestro módulo verilog, a pines (patitas o terminales) de la FPGA Spartan 3E que, a su vez, están conectados a elementos físicos de la placa (LEDs, switches, pulsadores, displays 7 segmentos, etc).

Pase a modo "Implementation", añada al proyecto el fichero de asociaciones **"basys2.ucf"** y edítelo. Todas las líneas deben empezar por // (comentarios), salvo las líneas correspondientes a los dispositivos físicos que vamos a utilizar. Podemos asociar, por ejemplo, las entradas (variables a,b,c) a los switches SW7,SW6, SW5, la salida (función f) al LED LD7, etc.

```
# This file is a general .ucf for Basys2 rev C board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins
// . . .

# Pin assignment for LEDs
NET "f" LOC = "G1" ; # Bank = 3, Signal name = LD7
//NET "G" LOC = "P4" ; # Bank = 2, Signal name = LD6
//NET "ld<5>" LOC = "N4" ;  # Bank = 2, Signal name = LD5
//NET "ld<4>" LOC = "N5" ;  # Bank = 2, Signal name = LD4
// . . .

# Pin assignment for SWs
NET "a" LOC = "N3";  # Bank = 2, Signal name = SW7
NET "b" LOC = "E2";  # Bank = 3, Signal name = SW6
NET "c" LOC = "F3";  # Bank = 3, Signal name = SW5
//NET "D" LOC = "G3";  # Bank = 3, Signal name = SW4
//NET "datb<3>" LOC = "B4";  # Bank = 3, Signal name = SW3
// . . .
```

El formato es:

```
NET “nombre_entrada_o_salida_modulo” LOC  = “Nombre pin FPGA”
```

Si en el fuente verilog, la entrada o salida se ha definido tipo array (por ejemplo, [3:0] sw), la sintaxis de asociación de cada bit tendrá esta sintaxis (es sólo un ejemplo):

```
NET “sw<0>” LOC  = “P11”; #Bank =2, Signal name =SW0
```

Evidentemente, debemos asociar entradas de módulos a interfaces de entrada de la placa (pulsadores o interruptores) y salidas de módulos a interfaces de salida de la placa (leds, segmentos de visualización).

Por último, si queremos generar el reloj del sistema digital mediante un pulsador (por ejemplo, para que cada vez que pulsemos el botón 0, se genere un pulso de reloj), o simplemente queremos detectar flancos, debemos indicarlo en el fichero UCF de la siguiente manera:

```
NET “btn<0>” LOC = “G12” #Bank =0, Signal name = BTN0
NET “btn<0>” CLOCK_DEDICATED_ROUTE =FALSE
```

```
# This file is a general .ucf for Basys2 rev C board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
# NET "seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
# NET "seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
# NET "seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
# NET "seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
# NET "seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
# NET "seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
# NET "seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
# NET "dp" LOC = "N13"; # Bank = 1, Signal name = DP

#NET "an<3>" LOC = "K14"; # Bank = 1, Signal name = AN3
#NET "an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2
#NET "an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1
#NET "an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0

# Pin assignment for LEDs
#NET "Led<7>" LOC = "G1" ; # Bank = 3, Signal name = LD7
#NET "Led<6>" LOC = "P4" ; # Bank = 2, Signal name = LD6
#NET "Led<5>" LOC = "N4" ;  # Bank = 2, Signal name = LD5
#NET "Led<4>" LOC = "N5" ;  # Bank = 2, Signal name = LD4
#NET "Led<3>" LOC = "P6" ; # Bank = 2, Signal name = LD3
#NET "Led<2>" LOC = "P7" ; # Bank = 3, Signal name = LD2
#NET "Led<1>" LOC = "M11" ; # Bank = 2, Signal name = LD1
#NET "Led<0>" LOC = "M5" ;  # Bank = 2, Signal name = LD0

# Pin assignment for SWs
#NET "sw<7>" LOC = "N3";  # Bank = 2, Signal name = SW7
#NET "sw<6>" LOC = "E2";  # Bank = 3, Signal name = SW6
#NET "sw<5>" LOC = "F3";  # Bank = 3, Signal name = SW5
#NET "sw<4>" LOC = "G3";  # Bank = 3, Signal name = SW4
#NET "sw<3>" LOC = "B4";  # Bank = 3, Signal name = SW3
#NET "sw<2>" LOC = "K3";  # Bank = 3, Signal name = SW2
#NET "sw<1>" LOC = "L3";  # Bank = 3, Signal name = SW1
#NET "sw<0>" LOC = "P11";  # Bank = 2, Signal name = SW0

#NET "btn<3>" LOC = "A7";  # Bank = 1, Signal name = BTN3
#NET "btn<2>" LOC = "M4";  # Bank = 0, Signal name = BTN2
#NET "btn<1>" LOC = "C11"; # Bank = 2, Signal name = BTN1
#NET "btn<0>" LOC = "G12"; # Bank = 0, Signal name = BTN0
#NET "btn<0>" CLOCK_DEDICATED_ROUTE = FALSE;

# ETC
```

Fichero de asociaciones (hay que quitar comentarios en las asociaciones usadas

Para finalizar, genere el fichero de programación del proyecto depurando errores si fuera necesario; para ello, marque el fichero Verilog raíz (de mayor jerarquía o “top module”), y en la ventana de herramientas seleccione “Generate Programming File”. (Si no usamos el reloj de la placa, porque lo generemos nosotros, o porque se utilice un reloj externo, no olvides seleccionar JTAG clock en “process properties/startup options) Con ello se generará un fichero de programación con extensión .bit que utilizaremos para programar la FPGA con ayuda de la aplicación ADEPT, suministrada por Digilent, fabricante de la placa Spartan 3E .