

Práctica 0: Introducción al diseño digital con XILINX

Objetivos:

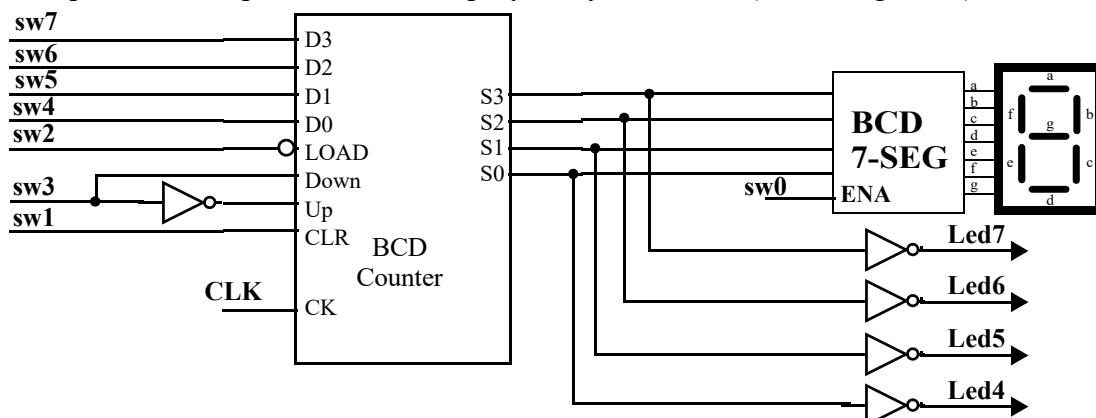
Introducir la herramienta de Xilinx “ISE Design Suite” para la gestión y diseño de proyectos electrónicos. Adquirir experiencia y competencias en herramientas CAD (Computer-Aided Design) en el ámbito del diseño electrónico.

Propuesta:

Ésta es una práctica guiada en la que se introduce a los estudiantes en las herramientas que se emplean en las diferentes etapas de diseño digital con la plataforma Nexys 3 de Digilent, y que son comunes en otros entornos de diseño CAD en este ámbito: creación del proyecto, captura de esquemáticos, simulación, síntesis e implementación. Para ello se parte de un proyecto inacabado, cuyo diagrama de bloques está incompleto, y que el estudiante completa y lleva hasta su implementación y verificación final en la plataforma de desarrollo.

Breve descripción de la tarea a realizar.

De forma resumida la propuesta es la siguiente: Inicialmente se presenta un diseño sencillo, denominado “**Practica0**”. El proyecto y los correspondientes ficheros de la herramienta Xilinx se proporcionan al estudiante como punto de partida. De esta manera se introducen la herramienta de gestión de proyectos y la herramienta de edición de esquemáticos, desde sus primeros pasos, en la opción de abrir un proyecto ya existente (ISE Design Suit).



El sistema propuesto es simple. Consta de un **contador BCD natural** (Decimal Codificado en Binario) (ver: <https://es.wikipedia.org/wiki/Contador>) cuya salida (S[3:0]) se visualiza mediante 4 **diodos led**, y mediante un **display 7-segmentos**. En este segundo caso la salida del contador BCD es recodificada mediante un **convertidor de código BDC-7segmentos**, que proporciona a su salida las 7 líneas de activación de cada uno de los led que componen cada segmento del display. Con cada flanco de la entrada de reloj (CLK), el contador se incrementa o decremente en una unidad, según indique el estado de las correspondientes entradas de control (UP/DOWN), activas a nivel alto. La velocidad de cuenta la marca la frecuencia de su entrada de reloj, que en este proyecto será seleccionable entre dos posibles valores. El contador posee también una entrada de carga LOAD, activa a nivel bajo, que permite iniciar la cuenta al valor que se indica en las líneas de entrada de dato D[3:0]. Una entrada adicional CLR, activa a nivel alto, lleva al contador al valor inicial 0. El diseño incluye también un módulo generador de relojes, que proporciona dos señales de reloj de frecuencia 1 y 4 Hz respectivamente, y otros bloques auxiliares que se emplean para controlar la visualización de datos en los displays de 7-segundos, y que serán utilizados también en prácticas y proyectos posteriores.

El proyecto no incluye en su esquemático el bloque selector de reloj, ni algunas de las conexiones entre los otros bloques. Una de las tareas que el estudiante ha de realizar en esta práctica es crear y diseñar dicho bloque, incorporarlo a proyecto “**Practica0**”, y completar el esquemático con las conexiones que faltan. A este módulo lo llamaremos **selec2a1**. Su funcionalidad es simple, posee una señal de control con la que selecciona, y deja pasar a su salida, una de sus dos entradas de dato. En este caso la señal de reloj con la que trabajará el contador BCD natural (1 o 4 Hz). El interfaz con el usuario se establece a partir de los interruptores y botones que hay en la placa de desarrollo Nexys3. Los interruptores **sw[7:1]** de la placa Nexys3 se utilizan como entradas del contador BCD, el interruptor **sw1** emplea como señal de habilitación del conversor BCD-7Seg; mientras que como entrada de control del selector se empleará el pulsador **btn0**, uno de los cinco pulsadores disponibles **btn[4:0]**.

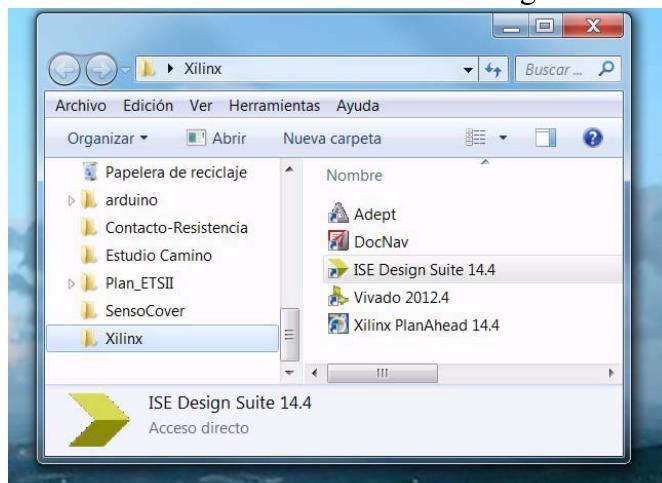
El **bloque selector** se diseña como un proyecto independiente; con lo que el estudiante aprenderá a **crear un proyecto nuevo** desde cero, a añadir una hoja de dibujo, y a dibujar en ella un diagrama lógico (**captura de esquemáticos**), seleccionando y cableando los componentes. A continuación se introduce el **procedimiento de simulación**: generación de **vectores de test**, herramienta de **simulación** y herramienta de **visualización de resultados de simulación**, con lo que aprenderá a **verificar el correcto funcionamiento** del diseño. Finalmente, este ejemplo se completa **generando un símbolo** para que el bloque selector diseñado, pueda ser considerado como un **nuevo elemento de librería**, y pueda ser **incorporado a posteriores diseños**, y en concreto, en esta práctica, al proyecto “**Practica0**”, donde será considerado como un elemento más de la librería de componentes.

Una vez incorporado a “**Practica0**”, el esquemático de este proyecto se completa añadiendo este nuevo elemento, estableciendo sus conexiones y completando el **cableado de bloques**, lo que permitirá introducir también la **creación de buses**. **Completado el esquemático, la práctica continua** ejecutando diversas herramientas, entre ellas, las de **síntesis e implementación** del sistema completo, hasta generar el **fichero de programación (*.bit)** de la FPGA incluida en la placa Nexys3. Una vez generado este fichero, el último paso consiste en conectar al ordenador la placa y ejecutar el **procedimiento de programación**, que configura la FPGA de la plataforma con el diseño. Ahora el funcionamiento del prototipo puede ser verificado interactuando con él.

Desarrollo de la práctica:

Tarea 1: Arranque y presentación del programa ISE Project Navigator.

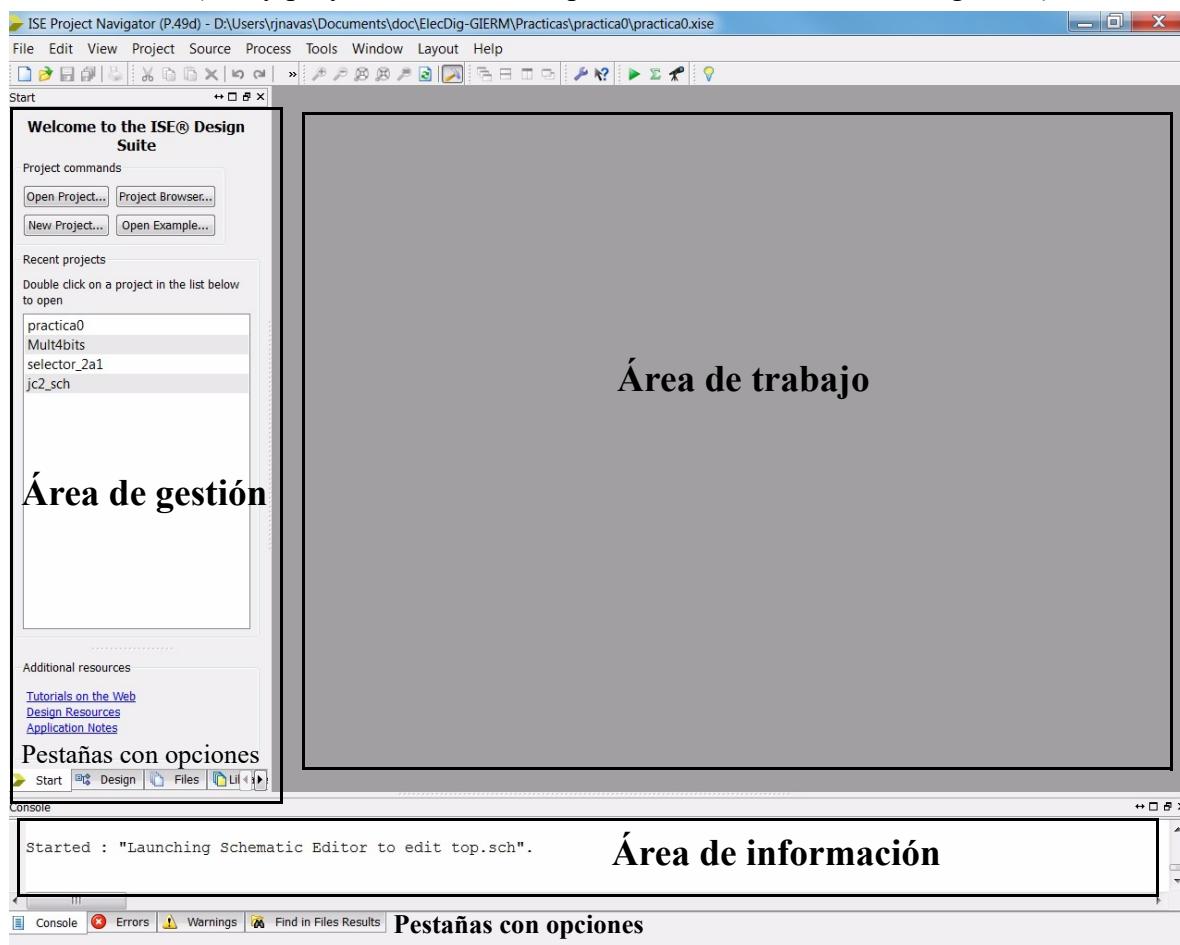
La llamada a la herramienta se hace desde el ícono “ISE Design Suite” en la carpeta Xilinx.



Este programa es una **herramienta profesional de gestión de los proyectos**, y todas las operaciones que se han de realizar durante el flujo de trabajo se ejecutan de forma controlada dentro de esta aplicación. Es de suma importancia arrancar cada sesión de trabajo ejecutando (“doble click”) el ícono “ISE Design Suite”, y no, como es habitual en otras aplicaciones windows, haciendo doble click sobre los ficheros asociados a ella. Además no es una buena práctica ejecutar simultáneamente varias copias de este programa para trabajar sobre un mismo proyecto, o varios proyectos, a la vez. Esta manera de proceder puede llevar a errores en la gestión de los muchos ficheros que conforman cada proyecto, que afectan gravemente a su integridad y a veces son difíciles de identificar.

El contenido de la pantalla que aparece al ejecutar “ISE Project Navigator” depende de si ya hay proyectos incluidos o seleccionados previamente o no. En ella cabe distinguir diversas áreas de actividad que pueden ser mostradas, o no, según convenga al usuario. Estas son: *área de trabajo*, *área de gestión* y *área de información*.

Pantalla inicial (si hay proyectos anteriores aparecen listados en el área de gestión).



En el área de gestión aparecen varias pestañas: *Start*, *Design*, *Files*, etc. La selección de cada una de ellas configura de diferente manera el área de trabajo, mostrándose diferentes aspectos del diseño. Lo mismo ocurre con el área de información, donde la selección de las diferentes pestañas cambia la información que aparece en dicha área: *Consola*, *Errores*, *Warning*.

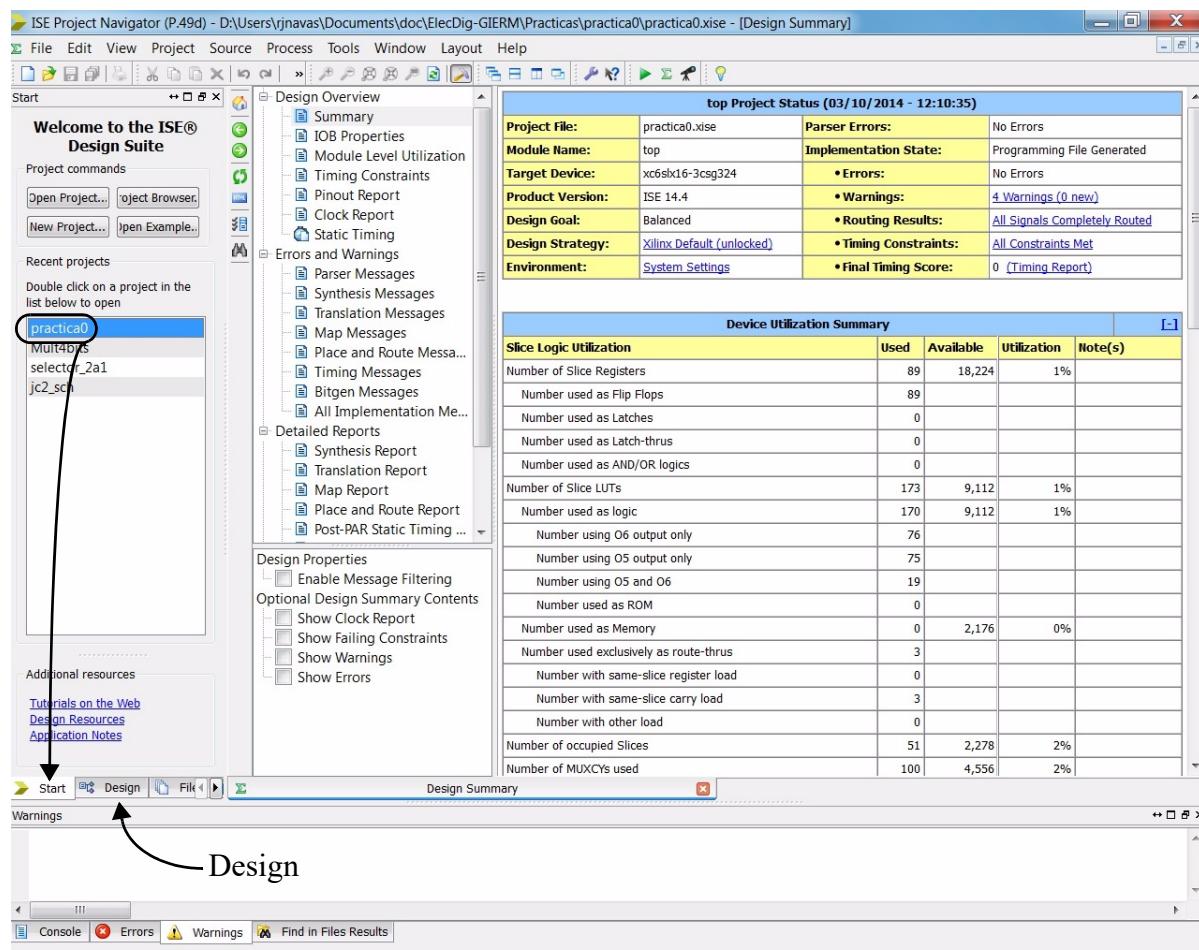
Tarea 2: Apertura y visualización de ficheros de un proyecto ya existente: practica0.

Hay dos formas de abrir un proyecto:

- Seleccionando (doble click) uno de la lista de proyectos recientes.
- Siguiendo las indicaciones tras pulsar el botón “Open Project” o “Project Browser”. Los ficheros del proyecto tienen la extensión *.xise.

Una vez abierto el proyecto, en el área de trabajo aparece información sobre su estado.

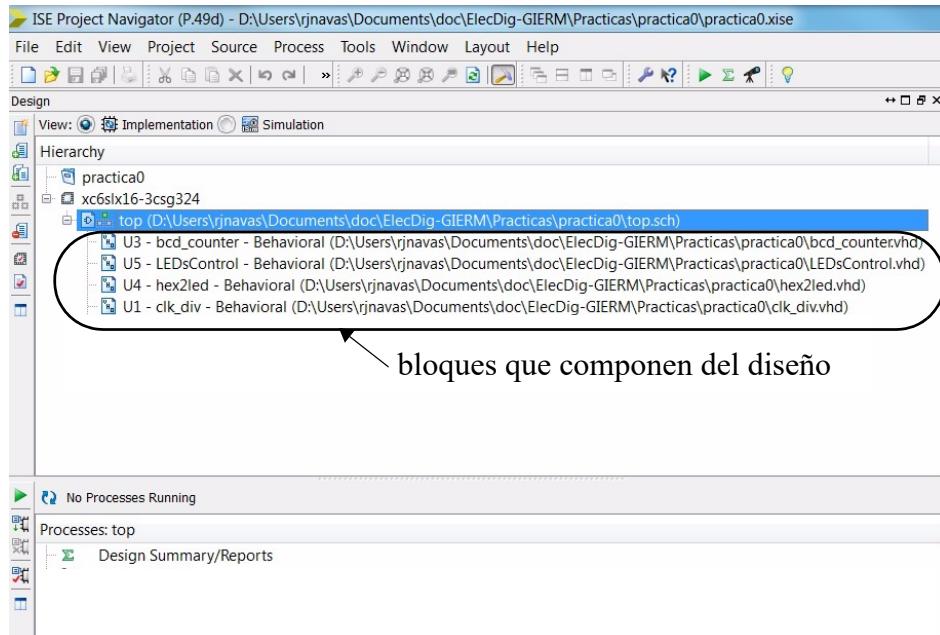
Tras doble click sobre “practica0” en el área de gestión aparece la pantalla:



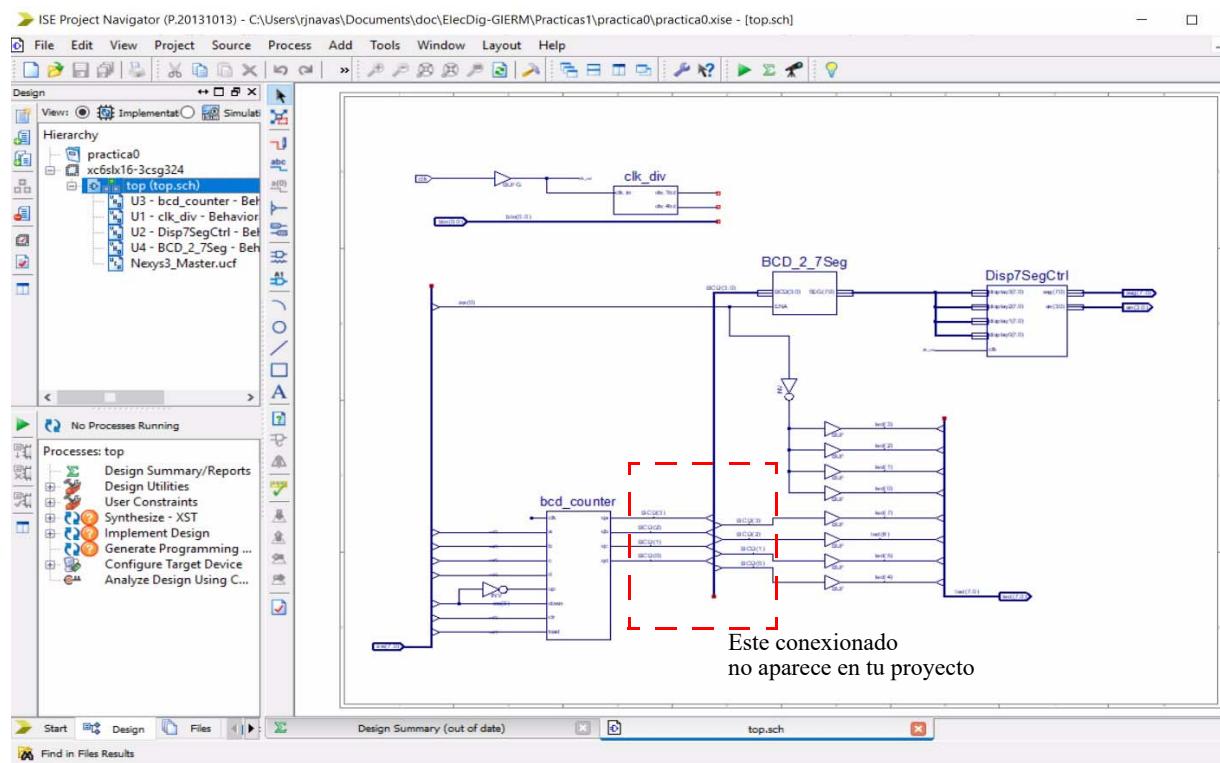
Si a continuación se selecciona la pestaña “Design” en el área de gestión, la vista de ésta cambia, y aparecen dos ventanas: *vistas* y *procesos*. En la parte superior (*view*) se proporciona información sobre la jerarquía de diseño, y los bloques funcionales que forman el proyecto; mientras que en la inferior (*process*) aparece información sobre el estado del proceso de diseño, esto es, qué fases del proceso de diseño se han completado. La información de ambas ventanas también cambia según se seleccione en la parte superior una de las opciones: *implementación* o *simulación*, cuyo uso veremos más adelante.

Tras la selección de cualquiera de los items que aparece en el árbol de jerarquía de diseño, en el área de trabajo se muestra el contenido de fichero seleccionado. Este puede ser una hoja de dibujo o “esquemático” (extensión *.sch), o un fichero de texto, si se trata de un elemento descrito en VHDL (extensión *.vhd), o un fichero de vectores de test, (test bench).

Pestaña “Design” seleccionada: área *view*: muestra los bloques que componen el diseño.



Tras la selección de la pestaña *Design* y doble click en “*top design*”, en el área de trabajo aparece la hoja de dibujo abierta: En este caso el esquema de “Practica0”.



Los bloques que componen el diseño son:

- **clk_div**: Este bloque genera dos señales de reloj de baja frecuencia a partir de la señal de reloj de 100MHZ, (entrada **ck**) que proporciona la placa Nexys3. Esta última se ha renombrado con el nombre **clk_out** a la salida del buffer de reloj (BUFG) que es obligatorio incorporar al diseño para las señales que van a ser usadas como reloj.
- **BCD_2_7Seg**: Este bloque genera el código 7-segmentos correspondiente a su entrada binaria hexadecimal. Posee una entrada de habilitación de salida (**ENA**).

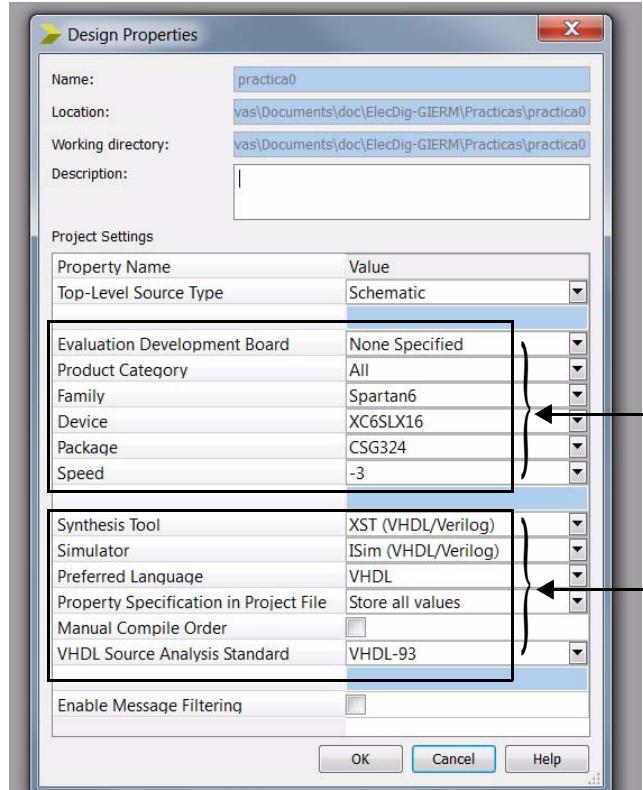
- **Disp7SegCtrl:** Este bloque implementa un procedimiento que permite visualizar información cada uno de los 4 **displays-7segmentos** de la placa Nexys3. Tiene como entrada cuatro buses, cada uno de los cuales proporciona el código 7-segmentos que se debe visualizar en cada display. Un “uno” lógico a la entrada hace que se encienda el correspondiente segmento del display. Necesita también una entrada de reloj a la que siempre debéis conectar la señal del sistema `clk_out`. Este bloque deberá incorporarse a cualquier proyecto en el que se desee utilizar los displays-7segmentos.
- **bcd_counter:** Este bloque implementa un contador “up/down” BCD natural síncrono. Donde `ck` es la entrada de reloj. Y el resto de entradas, activas a nivel alto son: **Load** que realiza la carga de un valor inicial dado por las entradas (**ABCD**), y las señales de control **Up/Down** que indican si la cuenta debe ir hacia adelante o hacia atrás.

La funcionalidad de cada uno de estos bloques está descrita en VHDL (lenguaje de descripción del hardware) y está recogida en los ficheros “`clk_div.vhd`”, “`BCD_2_7Seg.vhd`”, “`Diso7SegCtrl.vhd`” y “`bcd_counter.vhd`” respectivamente.

Como habrás notado, en el esquemático que se os ha proporcionado no aparece el conexionado enmarcado en rojo en la anterior figura. En esa zona se muestra cómo las salidas del contador se agrupan en un “bus” (mazo ordenado de cables) para facilitar su conexión al módulo **BCD_2_7Seg**, y cómo de ese “bus” se extraen la líneas que se conectan a los buffers de salida hacia los ledes de la placa Nexys3. Más adelante aprenderemos a realizar este tipo de conexiones para completar el esquemático, como aparece en la figura.

Información sobre la configuración del proyecto.

Las propiedades generales del proyecto “**Practica0**” pueden visualizarse en la opción “*Design Properties*”: La FPGA usada en la Nexys3 y las herramientas de simulación y síntesis de ISE.



Información sobre la FPGA usada en Nexys3

Información sobre las herramientas del ISE seleccionadas

Como esta herramienta puede usarse con diferentes productos de Xilinx, diferentes FPGA y herramientas de simulación, comprueba que tu proyecto está configurado como se ve en la figura.

A continuación cerraremos el proyecto. Para ello en la pantalla de inicio seleccionamos la correspondiente opción: “*Close Project*”. Si no hemos modificado nada se saldrá del programa sin más. Si se ha modificado algo, preguntara si queremos salvar el trabajo realizado. Ten cuidado con salvar o no las modificaciones realizadas.

Antes de cerrar el proyecto, si se desea tener una copia para llevarla a casa, o trabajar en otro ordenador, el procedimiento más ordenado es obtener un fichero comprimido (*.zip) que contenga todo el árbol de directorios y ficheros del proyecto. Para hacer esto, el procedimiento es el siguiente: Seleccionar en la barra de tareas de la parte superior la opción Project -> Archive. Se abrirá entonces una ventana en la que se proporciona el nombre y el path del archivo *.zip. Tras ejecutar “OK” se generará el correspondiente fichero comprimido.

Para recuperar el proyecto en otra ubicación bastará con descomprimir el fichero *.zip en un directorio, y abrir el proyecto desde “ISE Project Navigator” con el procedimiento que se inicia tras pulsar “*Open Project*” o “*Project Browser*”.

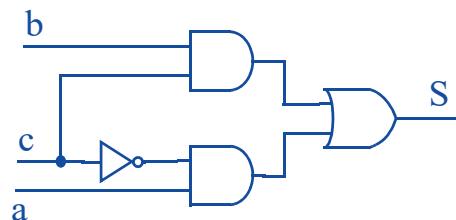
Ésta es la forma más correcta de compartir un proyecto, cualquier otro procedimiento, como la copia e intercambio de uno o varios ficheros sin más puede confundir al gestor de proyectos y llevar a errores difíciles de identificar posteriormente. Por otra parte, el gestor de proyectos se lleva mal con el trabajo en la nube, por lo que esta forma de trabajar **no está recomendada**.

Creación y diseño de un nuevo bloque: Selec_2a1

En las tareas que siguen en la Práctica 0 se propone crear un nuevo bloque, que llamaremos **Selec_2a1**, partiendo de su descripción como circuito lógico (esquemático), definiendo para él un símbolo que resuma su interfaz entrada/salida, para posteriormente incorporarlo al proyecto inicial **practica0**. El bloque **Selec_2a1** tendrá tres entradas que llamaremos *c*, *b* y *a* y una salida *S*. Su funcionalidad es simple, se recoge en la Tabla de Verdad de la figura, y puede resumirse así: Cuando la entrada de control *c* = 0 entonces la salida *S* = *a*, mientras que cuando *c* = 1, entonces *S* = *b*. Una expresión booleana de *S* en forma SDP y su diagrama lógico son los siguientes:

<i>c</i>	<i>b</i>	<i>a</i>	<i>S</i>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$S(a, b, c) = \bar{c}a + cb$$



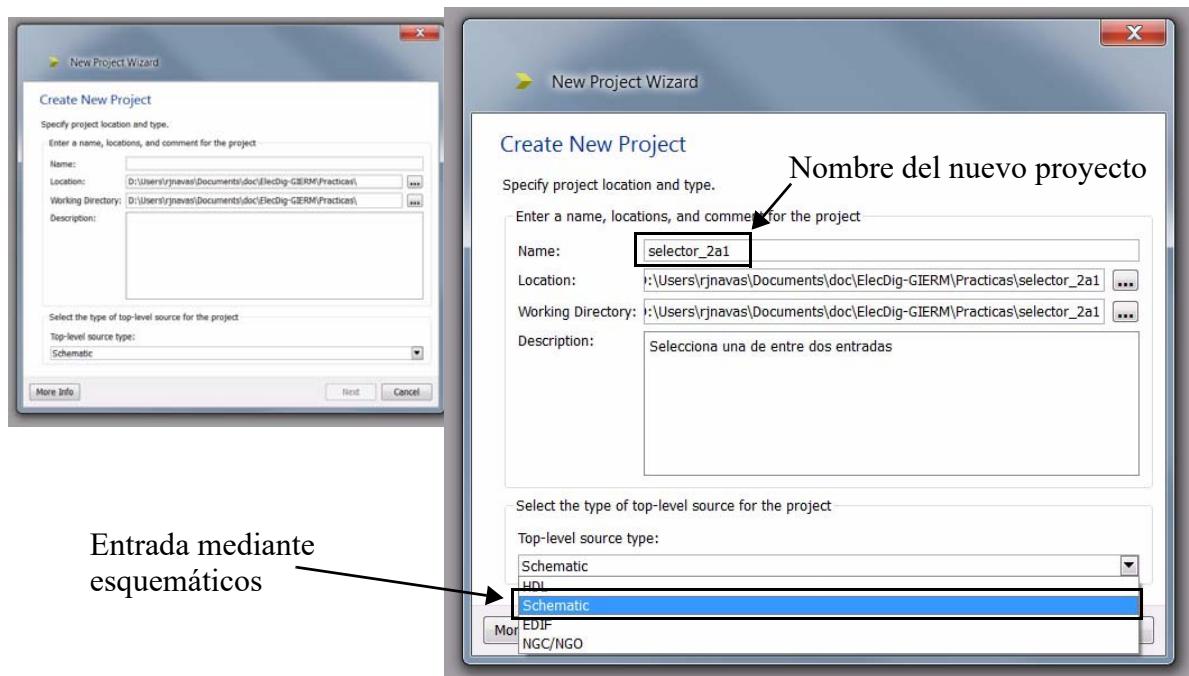
Las siguientes tareas nos van a enseñar a crear y validar este diseño antes de incorporarlo al proyecto **practica0**.

Para crear incorporar un nuevo bloque a nuestro diseño tenemos dos opciones: (1) crear un nuevo proyecto independiente que incluya dicho bloque, y posteriormente incorporarlo al proyecto **practica0**; (2) añadir un nuevo bloque a un proyecto ya existente, en este caso seleccionar en la barra de tareas de la parte superior la opción Project - > Add New Source. Esta segunda opción sería la más habitual para el ejemplo aquí presentado, puesto que se trata de un bloque que no tiene la entidad de proyecto, sin embargo, en lo que sigue utilizaremos el camino de la opción (1) para presentar así el procedimiento de creación de un nuevo proyecto y la incorporación de bloques diseñados en otras proyectos a uno dado. El camino de la opción (2) pasa por saltar la **Tarea 3** y continuar con lo que se indica en la **Tarea 4**, teniendo en cuenta que el nuevo bloque será incorporado al proyecto abierto, con lo que no será necesario incorporarlo de nuevo como más adelante se indica la primera parte del punto 1 de la **Tarea 7**. Así pues, en este tutorial, continuaremos crearemos un nuevo proyecto para contener al bloque **Selec_2a1**.

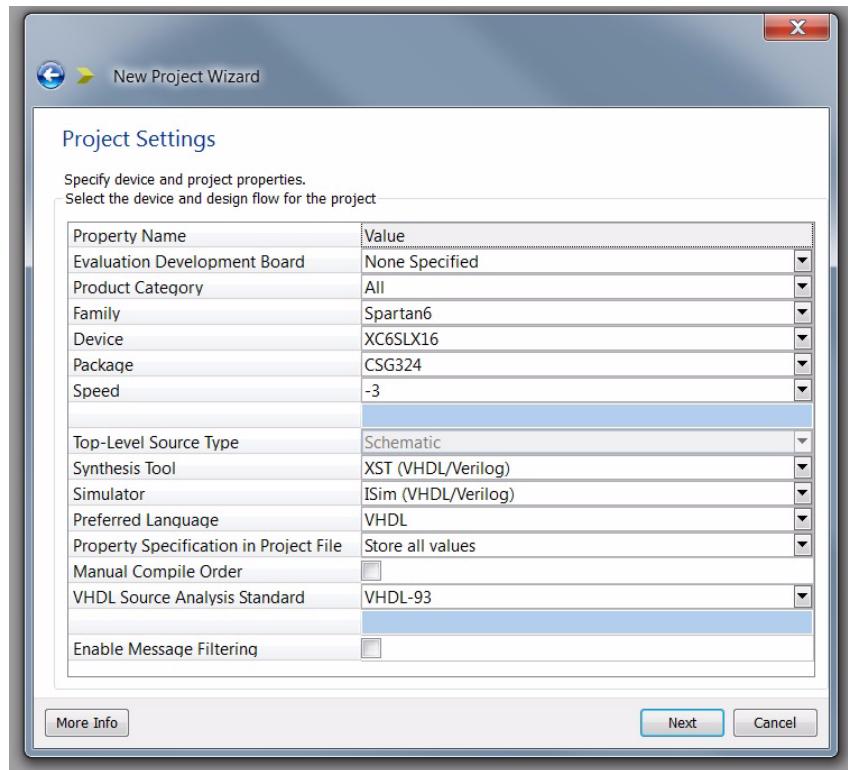
Tarea 3: Crear y configurar un nuevo proyecto. Selec_2a1.

Volvemos a la pantalla de inicio y seleccionamos la opción “*New Project*”. De esta forma se generan los siguientes cuadros de diálogo para el asistente: “*New Project Wizard*”.

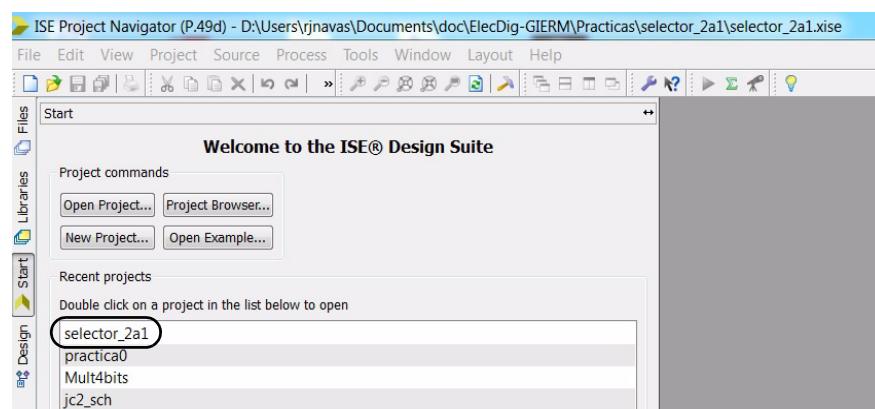
En primer lugar damos nombre a nuestro nuevo proyecto, y a continuación indicamos que el tipo de especificación del diseño, en este caso mediante hoja de dibujo “Schematic”.



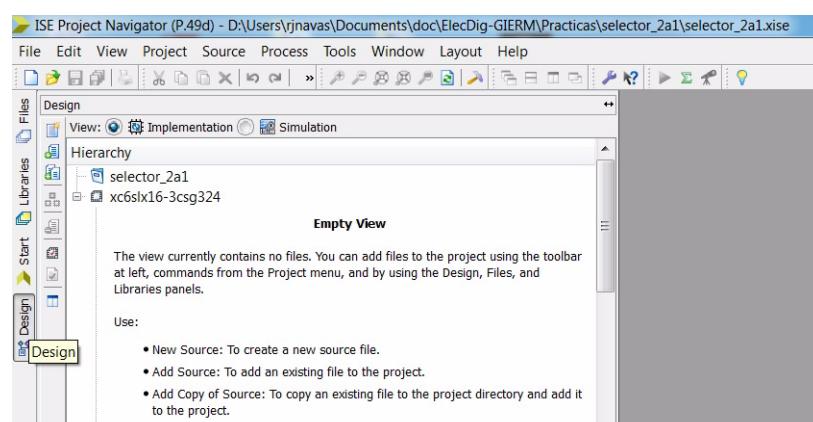
A continuación aparece una venta de configuración, dónde se establecen las propiedades: Tipo de FPGA usada, tipo de fuentes de diseño, simulador empleado, etc. Normalmente, esta pantalla estará configurada si el programa se ha usado con anterioridad en otros proyectos para la misma FPGA, como es nuestro caso. En cualquier caso, verifica que los datos que muestra corresponden con los de la siguiente imagen, de no ser así actualizalos, y por ejemplo, no se ha seleccionado el modelo de FPGA, esto dará lugar a errores durante la implementación del diseño en la FPGA:

Configuración:

A continuación, tras pulsar “Next” el programa crea el proyecto con todo su arbol de directorios y aparecerá finalmente listado en el área de gestión.

Proyecto creado

Tras seleccionar pestaña de Diseño en área de gestión. Aparece el árbol de jerarquía de diseño:



Por ahora el proyecto no contiene aún ningún elemento.

Sólo aparece el ítem “**xc6slx-3csg324**” que representa al dispositivo FPGA que se quiere configurar. El siguiente paso es añadir los elementos que componen el diseño (“sources”).

Tarea 4: Creación de nueva fuente. Dibujar esquemático. Selec_2a1.

1. Crear nueva hoja de diseño

En este caso vamos a añadir una hoja de esquemático, en la que dibujaremos el diagrama lógico del bloque **selec_2a1**.

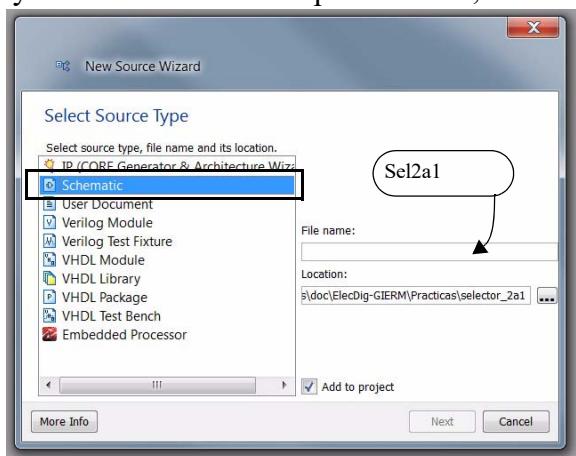
El procedimiento es el siguiente:

- Seleccionar **xc6slx16-3csg324** y *añadir nueva fuente: sel2a1.sch*.

Esto puede hacerse de varias maneras.

Una de ellas es seleccionando la opción “*New Source*” en el desplegable que aparece al seleccionar la pestaña *Project* en la barra de herramientas.

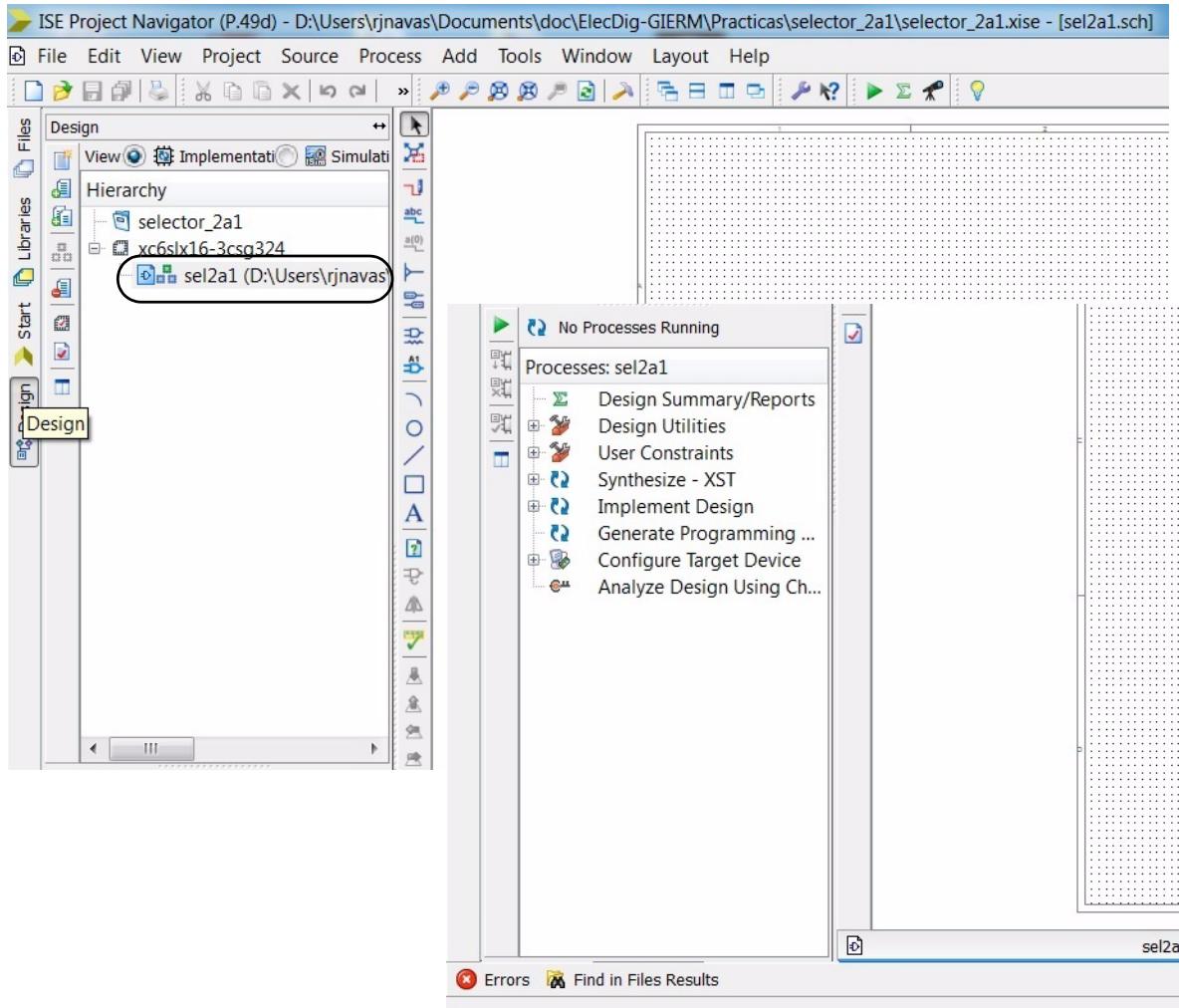
Así aparece el cuadro de diálogo “*New Source Wizard*”, que se completa con el nombre del fichero de esquemático y con la selección del tipo de fuente, en este caso “*Schematic*”.



Tras pulsar “*Next*”. Se crea una hoja de dibujo vacía en el área de trabajo en la que reproducimos el diagrama lógico del circuito **selec_2a1**:



Nótese cómo en la parte superior del área de gestión (zona view) se ha creado un nuevo elemento **sel2a1.sch**; mientras que en la parte inferior (zona process) ha aparecido un árbol con los diferentes procesos que hay que completar hasta llegar a la implementación. El cuadradito de color verde que aparece delante del nombre del fichero indica que éste es ahora el fichero más alto en la jerarquía de diseño.



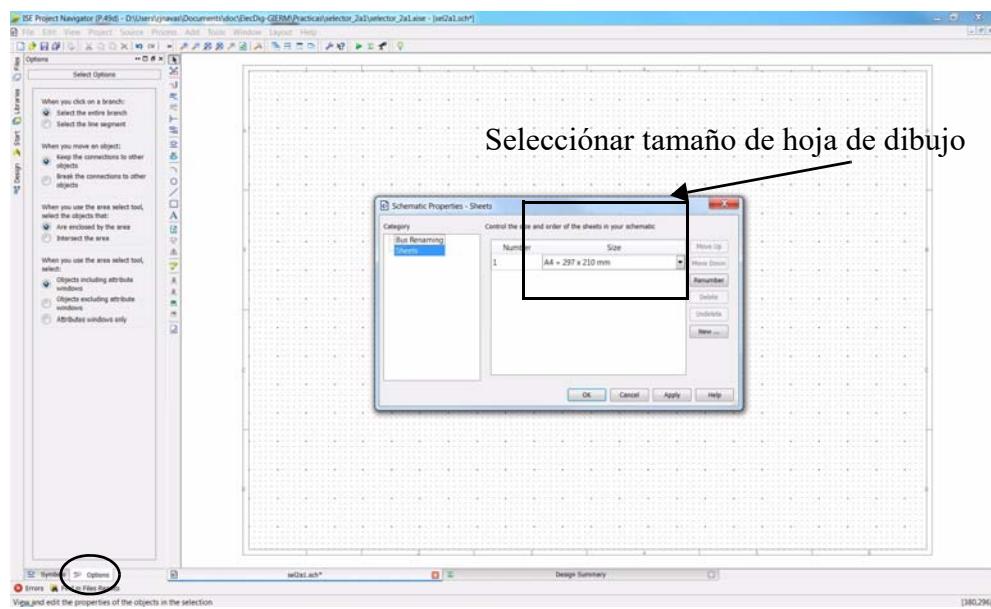
2. Descripción del circuito. Dibujar un esquemático.

Antes de comenzar a dibujar conviene verificar que el tamaño de la hoja de dibujo es adecuado para contener el número de elementos del diagrama que se pretende dibujar.

Para ver el tamaño actual de la hoja de dibujo y en su caso modificarlo puede seguirse el siguiente procedimiento.

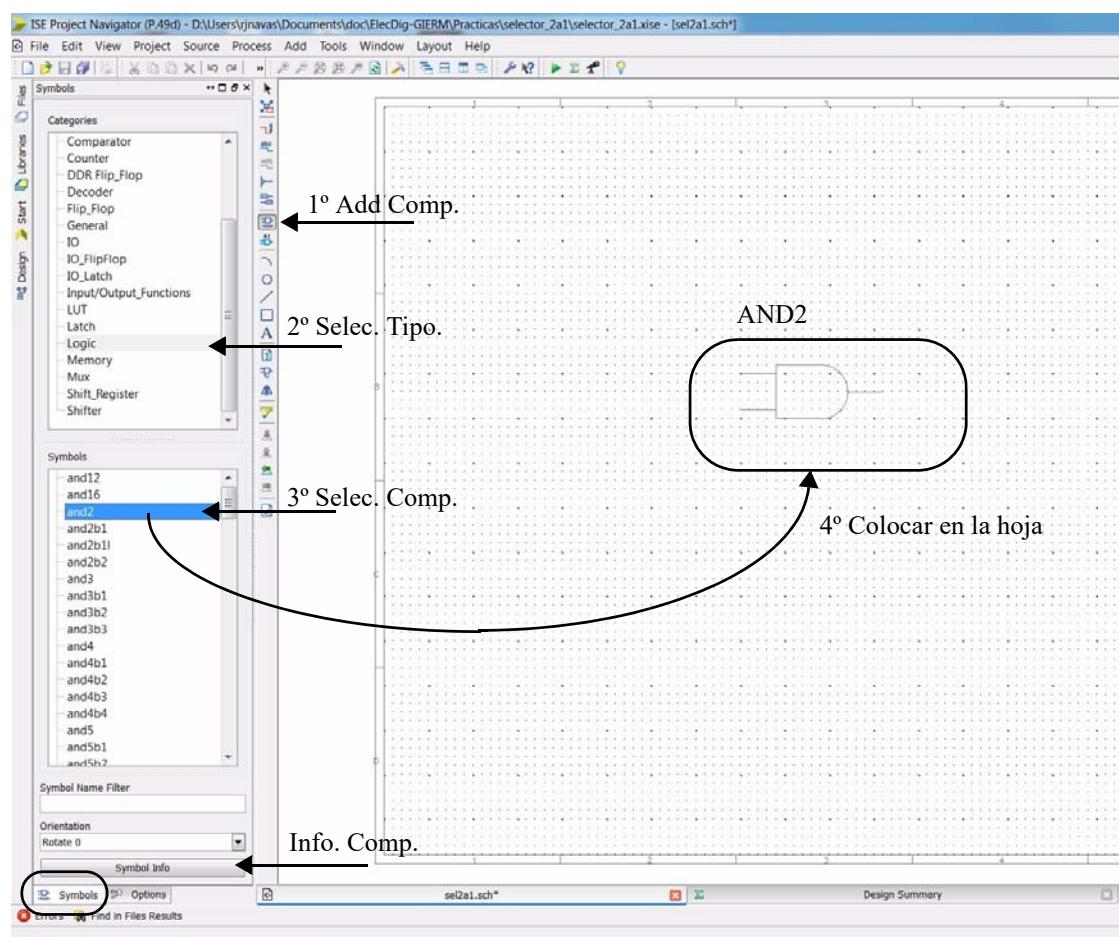
Cambio de propiedades de la hoja de dibujo: Pulsar Raton Botón derecho, seleccionar la Opción: *Object Properties* -> Seleccionar tamaño de la hoja de dibujo. Seleccionado A4.

En este caso, como nuestro diagrama es pequeño, puede que no haya problema con el tamaño de hoja por defecto, pero en otra ocasión puede ser necesario trabajar con una hoja de mayor tamaño.



Las acciones a realizar son:

1. Dibujar el diagrama lógico del bloque **selec_2a1**: Añadir componentes. Puertas Lógicas.



1º Añadir componente de librería: *Seleccionar Add comp.*

2º Seleccionar en el desplegable librerías el tipo adecuado. En este caso “Logic”

3º Seleccionar un elemento concreto: En este caso puerta AND2.

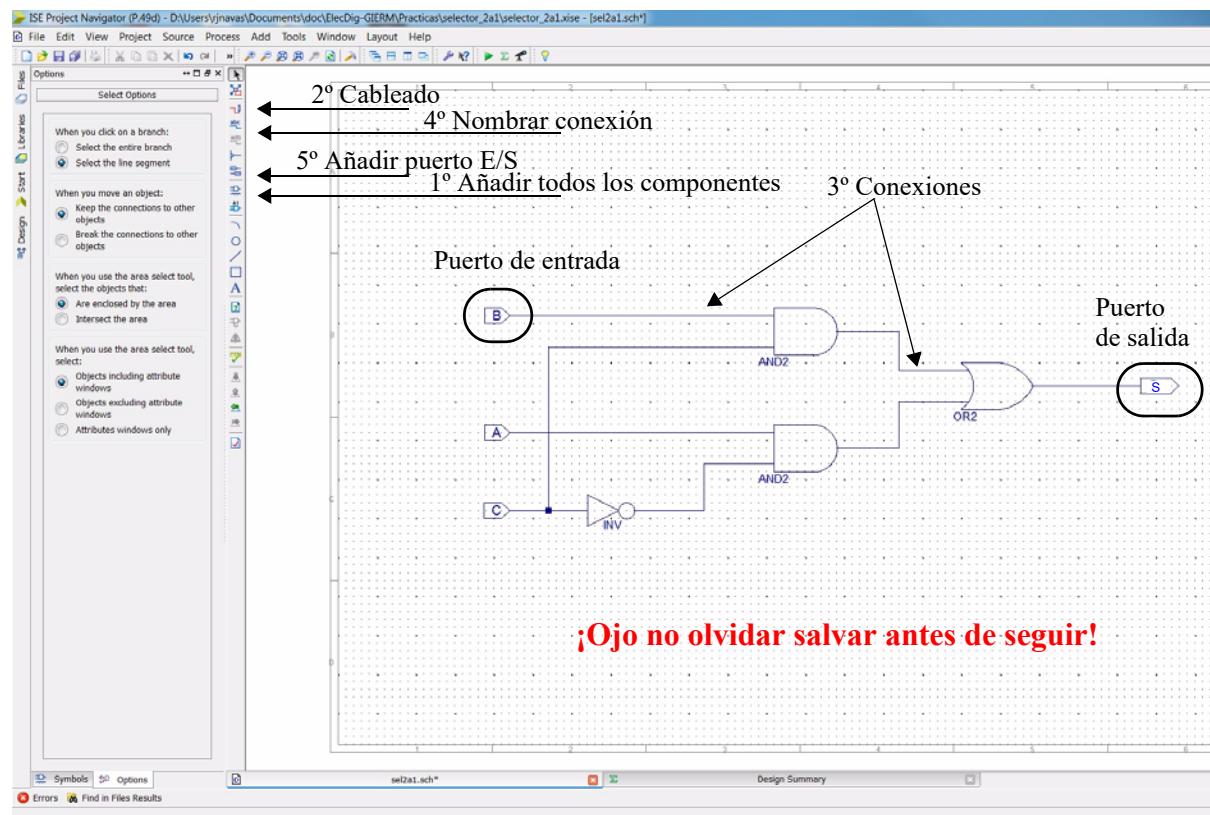
4º Posicionar el elemento en la hoja de dibujo. Morver ratón y “click”.

La información completa sobre la funcionalidad cada elemento de la librería puede visualizarse seleccionando el correspondiente elemento y la pestaña Symbols.

2. Dibujar diagrama lógico: Cableado y conexiones. Declaración de puertos entrada/salida.

- 1º Antes de establecer las conexiones, conviene ubicar todos los componentes del diseño.
- 2º Seleccionar cableado.
- 3º Realizar las conexiones.
- 4º Poner nombre a cada conexión.
- 5º Añadir componentes E/S, para indicar cuáles son los terminales (puertos) entrada/salida.

Ojo: No olvides salvar los ficheros modificados antes de avanzar en el flujo de trabajo.



Como en todo editor de dibujo, este editor permite muchas acciones, como mover, borrar, copiar, etc. tanto conexiones como componentes, y con diferentes opciones, que sería largo de exponer aquí. Siéntete libre para explora estas posibilidades cuando edites tu dibujo.

Además, para evitar marañas de cables en diseños complejos, está permitido **establecer una conexión entre cables** simplemente **asignándoles el mismo nombre**. Esta facilidad tiene un pequeño inconveniente. **Debes tener especial cuidado cuando borres una conexión a la que hayas dado nombre previamente**. En este caso asegúrate de que has borrado la línea de conexión completa. Si solo borras un trocito del cable, el resto de la conexión seguirá llamándose como la habías nombrado, y las partes nombradas seguirán conectadas. Si quieras borrar una conexión completa pero has olvidado borrar alguno de los trozos, y con posterioridad nombras otra conexión empleando el mismo nombre del cable que habías borrado, esto dará lugar a errores, cuyos mensajes pueden resultar confusos, y cuyo origen resulta a veces difícil de localizar. (Observa las Opciones de Selección la izquierda)

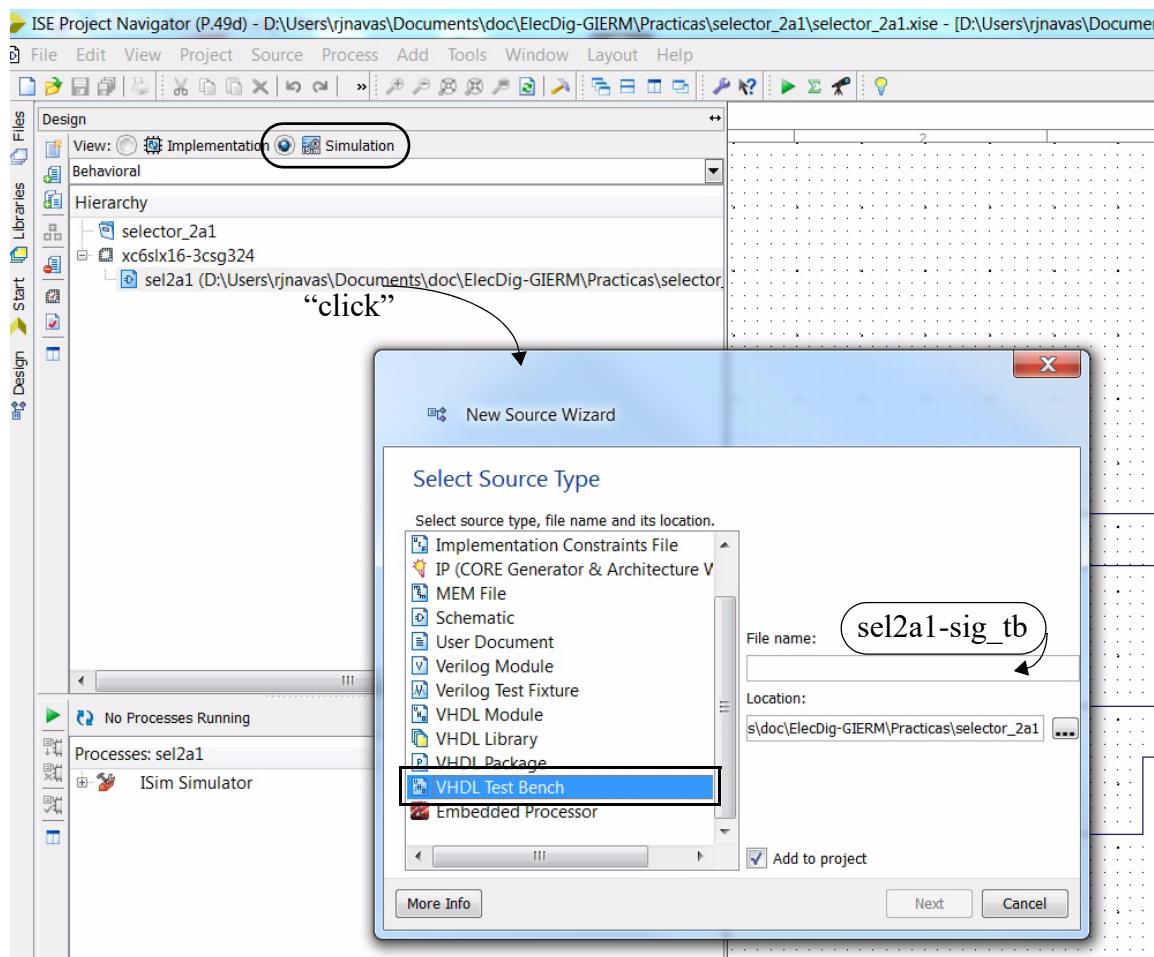
Tarea 5: Simulación: verificación funcional del diseño. Selec_2a1.

Antes de dar por bueno un diseño y utilizarlo como elemento de librería en un nuevo proyecto, o continuar con los procesos de síntesis e implementación en la placa Nexys3, es conveniente asegurarse de que su funcionamiento es el que se espera. Para ello resulta útil, y a veces imprescindible, **simular su funcionalidad**. En general, con la herramienta de simulación trataremos de ejercitar el bloque diseñado aplicando a su entrada una secuencia de valores y comprobando que su respuesta es la esperada para todos los valores y circunstancias posibles de sus entradas. En el caso de un bloque combinacional como **selec_2a1** bastará con verificar que el diagrama dibujado en el esquemático responde como se recoge en su tabla de verdad. La secuencia de valores de entrada con que se ejercita un sistema para validar su funcionalidad se denomina **vectores de test**.

1. Creación de fichero de Vectores de Test.

Para excitar el circuito en una circunstancia particular es necesario añadir al proyecto un nuevo elemento denominado fichero de *vectores de test*. Esto puede hacerse de varias maneras. Una de ellas es seleccionando la opción “New Source” en el desplegable que aparece al seleccionar la pestaña *Project* en la barra de herramienta.

A continuación aparece el cuadro de diálogo:



En él se completa el campo de *nombre* del nuevo fichero y se selecciona una fuente de tipo “*VHDL Test Bench*”.

En nuestra herramienta éste es un fichero especificado en lenguaje VHDL, (extensión *.vhd) que contiene una descripción de los estímulos que se desea aplicar al sistema. (Es conveniente que el nombre de este fichero sea fácilmente identifiable con el bloque que se desea simular, y es práctica común añadir al nombre los caracteres “_tb” para indentificar el tipo de fichero del que se trata y distinguirlo de otros ficheros con la misma extensión *.vhd que describen módulos en HDL).

Tras este procedimiento se genera un fichero VHDL que sirve de plantilla. En él hay una sección “cabecera” en la que aparecen referenciadas las variables de entrada y salida del bloque; una sección de “declaración de variables auxiliares”; y una sección en la que sólo queda completar el código correspondiente a la secuencia de estímulos que se desea aplicar al circuito. La figura muestra el fichero de vectores de test, generado automáticamente, para el bloque **sel2a1**.

```

ISE Project Navigator (P.49d) - D:\Users\rjnavas\Documents\doc\ElecDig-GIERM\Practicas\selector_2a1\selector_2a1xise - [sel2a1-sig_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design Files View: Implementation Simulation
Libraries Behavioral
Hierarchy
  selector_2a1
    xc6slx16-3csg324
      sel2a1_sel2a1_sch_tb - behavior
Design Utilities
No Processes Running
No single design module is selected.
  Design Utilities
  Cabecera
  No es necesario modificar
  Variables auxiliares,
  a veces hay que añadir o modificar variables
  Asignación de entrada /salida,
  a veces hay que añadir o modificar esta asignación
  En esta sección
  se escriben los vectores de test
D:\Users\rjnavas\Documents\doc\ElecDig-GIERM\Practicas\selector_2a1\sel2a1.sch
sel2a1-sig_tb.vhd
  Errors Find in Files Results

```

The screenshot shows the ISE Project Navigator interface with the file 'sel2a1-sig_tb.vhd' open. The code is as follows:

```

1 -- VHDL test bench created from schematic D:\Users\rjnavas\Documents\doc\ElecDig-GIERM\Practicas\sele
2 --
3 -- Notes:
4 -- 1) This testbench template has been automatically generated using types
5 -- std_logic and std_logic_vector for the ports of the unit under test.
6 -- Xilinx recommends that these types always be used for the top-level
7 -- I/O of design in order to guarantee that the testbench will bind
8 -- correctly to the timing (post-route) simulation model.
9 -- 2) To use this template as your testbench, change the filename to any
10 -- name of your choice with the extension .vhd, and use the "Source->Add"
11 -- menu in Project Navigator to import the testbench. Then
12 -- edit the user defined section below, adding code to generate the
13 -- stimulus for your design.
14 --
15 LIBRARY ieee;
16 USE ieee.std_logic_1164.ALL;
17 USE ieee.numeric_std.ALL;
18 LIBRARY UNISIM;
19 USE UNISIM.Vcomponents.ALL;
20 ENTITY sel2a1_sel2a1_sch_tb IS
21   END sel2a1_sel2a1_sch_tb;
22 ARCHITECTURE behavioral OF sel2a1_sel2a1_sch_tb IS
23   COMPONENT sel2a1
24     PORT( C : IN STD_LOGIC;
25           A : IN STD_LOGIC;
26           B : IN STD_LOGIC;
27           S : OUT STD_LOGIC);
28   END COMPONENT;
29   SIGNAL C : STD_LOGIC;
30   SIGNAL A : STD_LOGIC;
31   SIGNAL B : STD_LOGIC;
32   SIGNAL S : STD_LOGIC;
33 BEGIN
34   UUT: sel2a1 PORT MAP(
35     C => C,
36     A => A,
37     B => B,
38     S => S
39   );
40   tb : PROCESS
41   BEGIN
42     WAIT; -- will wait forever
43   END PROCESS;
44   --
45   -- *** Test Bench - User Defined Section ***
46   tb : PROCESS
47   BEGIN
48     WAIT; -- will wait forever
49   END PROCESS;

```

Annotations on the right side of the code:

- Cabecera**: Brackets around the first 13 lines of code.
- No es necesario modificar**: Brackets around the first 13 lines of code.
- Variables auxiliares, a veces hay que añadir o modificar variables**: Brackets around lines 31 to 34.
- Asignación de entrada /salida, a veces hay que añadir o modificar esta asignación**: Brackets around lines 38 to 43.
- En esta sección se escriben los vectores de test**: Brackets around lines 46 to 49.

Si en la sección “cabecera” no aparecen automáticamente los nombres de las entradas y salidas de tu módulo, es que este fichero no se ha creado correctamente. Probablemente no has seguido bien las indicaciones de creación, o hay un error en el esquemático que confunde al programa.

Por cada conjunto de vectores de test que se quiera ejercitar se tendrá que generar un nuevo fichero, al que queda asociada una copia del bloque que se desea simular (UUT Unit Under Test).

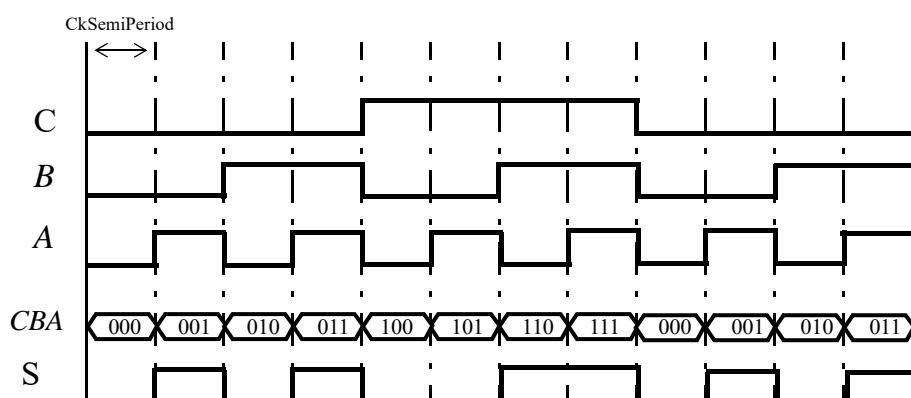
El bloque **sel2a1** es un circuito combinacional simple, posee tres entradas A, B y C y una salida S. Por tanto la manera mejor y más completa de verificar su funcionamiento es ejercitarse con todas las combinaciones posibles de sus variables de entrada.

Como el simulador trata de emular el comportamiento en el tiempo del sistema, los vectores de test deberán estar definidos como una secuencia temporal, en la que aplicaremos sucesivamente cada una de las ocho combinaciones posibles de las tres variables de entrada. Como el circuito es combinacional, (sistema sin memoria) la respuesta será la misma independientemente del orden que cada combinación de las variables de entrada ocupe en esa secuencia. Sin embargo, por conveniencia, en los sistemas combinacionales, la secuencia más cómoda será la que reproduzca su tabla de verdad. Aunque, en el caso de simular un circuito secuencial, habrá que tener en cuenta todas las posibles secuencias de entrada.

Para planificar la secuencia de entrada es útil dibujar un cronograma. Esto es un diagrama temporal que muestre cómo evolucionan las señales de entrada con el tiempo, e incluso que recoja cuál es la respuesta esperada de sistema. Esto ayudará a describir y visualizar cómo se van a aplicar los vectores de test y a escribir el correspondiente fichero. Y, por supuesto, a validar su correcto funcionamiento, dado que las mayoría de las herramientas de simulación permiten visualizar el comportamiento de los sistemas en forma de cronograma.

Para un circuito combinacional, reproducir una secuencia que haga un barrido por todos las posibles combinaciones de entrada es simple. En la figura se muestra el cronograma que reproduce la tabla de verdad del circuito **selec_2a1** en particular, y en general de cualquier circuito combinacional con tres entradas y una salida. Lo único que cambiará en cada caso será la forma de onda de S.

Cronograma



Nótese que en este cronograma la secuencia de entrada de la señal A es una señal cuadrada de periodo $2 \cdot \text{CkSemiPeriod}$. Que la secuencia de entrada de la señal B es una señal cuadrada de periodo $4 \cdot \text{CkSemiPeriod}$; y que la secuencia de la entrada C es una señal cuadrada de periodo $8 \cdot \text{CkSemiPeriod}$.

La línea etiquetada CBA, es un recurso gráfico, usado en muchos simuladores, para indicar que las señales de entrada se han agrupado en formato “bus”, que resulta útil para interpretar grupos de señales.

Una secuencia como ésta, que reproduce la tabla de verdad de un circuito combinacional, puede especificarse de distintas maneras mediante el lenguaje VHDL, por lo que se pueden escribir varios ficheros de test diferentes, para obtener el mismo resultado. A continuación se muestran cuatro ejemplos.

El primero de ellos (**Ejemplo1**) es el más directo, pero más de “fuerza bruta”. Describe la evolución de las entradas transición a transición, estableciendo un tiempo mínimo de espera entre ellas. El segundo (**Ejemplo2**) aprovecha la periodicidad de las variaciones de cada una de las variables de entrada en la tabla de verdad, por lo que asigna a cada entrada una señal de reloj de frecuencia una múltiplo de otra, para en conjunto generar la secuencia tabla de verdad requerida. El tercero (**Ejemplo3**) especifica las secuencia temporal como en el Ejemplo 1 pero utiliza variables intermedias para asignar cada combinación de entrada. El cuarto (**Ejemplo4**) utiliza un “bucle for” del lenguaje VHDL. Cualquiera de las cuatro opciones es válida para el propósito que aquí se tiene. La conveniencia de una u otra en una situación concreta la decide el diseñador, aunque el **Ejemplo2** y el **Ejemplo4** resultan más apropiados cuando el número de entradas es elevado.

Ejemplo 1

```
-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
    C <= '0';
    B <= '0';
    A <= '0';
    wait for 50ns;
    C <= '0';
    B <= '0';
    A <= '1';
    wait for 50ns;
    C <= '0';
    B <= '1';
    A <= '0';
    wait for 50ns;
    C <= '0';
    B <= '1';
    A <= '1';
    wait for 50ns;
    C <= '1';
    B <= '0';
    A <= '0';
    wait for 50ns;
    C <= '1';
    B <= '0';
    A <= '1';
    wait for 50ns;
    C <= '1';
    B <= '1';
    A <= '0';
    wait for 50ns;
    C <= '1';
    B <= '1';
    A <= '1';
    wait for 50ns;
    C <= '1';
    B <= '1';
    A <= '0';
    wait for 50ns;
    C <= '1';
    B <= '1';
    A <= '1';
    wait for 50ns;
    C <= '1';
    B <= '1';
    A <= '1';

    WAIT; -- will wait forever
END PROCESS;
```

Ejemplo 2

```
-- *** Test Bench - User Defined Section ***
-- Forma de onda de la señal A
clock1: process
begin
    A <= '0';
    loop
        wait for CkSemiPeriod;
        A <= not A;
    end loop;
end process clock1;

-- Forma de onda de la señal B
clock2: process
begin
    B <= '0';
    loop
        wait for (2*CkSemiPeriod);
        B <= not B;
    end loop;
end process clock2;

-- Forma de onda de la señal C
clock3: process
begin
    C <= '0';
    loop
        wait for (4*CkSemiPeriod);
        C <= not C;
    end loop;
end process clock3;

-- *** End Test Bench - User Defined Section ***
END;
```

En el **Ejemplo 1**, se van cambiando los valores de las variables de entrada cada 50 ns en este caso, por asignación directa del valor lógico requerido en cada instante.

En el **Ejemplo 2**, se emplean tres procesos que, en VHDL, se ejecutan simultáneamente, y que generan tres señales periódicas, de frecuencia decreciente. La de frecuencia más alta, corresponde a la entrada A, de periodo 100ns. La entrada B de frecuencia mitad que la anterior y la entrada C de frecuencia mitad que la anterior, esto es, un cuarto de la frecuencia asignada a A. Así es posible reproducir de forma cíclica el cronograma que se muestra en la anterior página. En este caso es necesario declarar la variable auxiliar *CkSemiPeriod* y su valor (50 ns en este ejemplo) en la zona de **variables auxiliares del fichero**.

La línea VHDL a incluir en dicha sección es:

```
constant CkSemiPeriod : TIME := 50 ns;
```

En el **Ejemplo 3**: Se procede como en el Ejemplo 1, asignando valores a las variables de entrada cada 50 ns. Sin embargo, en este caso, por comodidad, se ha introducido una señal intermedia, **INPUT[3:0]**, que es un vector de tres componentes. Cada una de las cuales se asocia a una de las entradas A, B y C. Esta señal permite asignar valores a las entradas de forma conjunta, como muestra el código VHDL que aquí se reproduce. La variable **INPUT** se declara como señal en la correspondiente sección de la plantilla, mientras que la asignación y correspondencia de cada elemento del vector **INPUT** a las entradas se realiza en la definición “sel2a1 PORT MAP()”. Nota que también, en esta sección se ha anulado la declaración de las señales intermedias A, B y C anteponiéndoles la marca de comentario “--” ya que en este caso no se utilizan.

Ejemplo 3

```
-- SIGNAL C    : STD_LOGIC;
-- SIGNAL A    : STD_LOGIC;
-- SIGNAL B    : STD_LOGIC;
SIGNAL S      : STD_LOGIC;
SIGNAL INPUT  : std_logic_vector(2 downto 0);

BEGIN
  UUT: sel2a1 PORT MAP(
    C => INPUT(2),
    A => INPUT(0),
    B => INPUT(1),
    S => S
  );
-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
  INPUT <= "000";
  wait for 50 ns;
  INPUT <= "001";
  wait for 50 ns;
  INPUT <= "010";
  wait for 50 ns;
  INPUT <= "011";
  wait for 50 ns;
  INPUT <= "100";
  wait for 50 ns;
  INPUT <= "101";
  wait for 50 ns;
  INPUT <= "110";
  wait for 50 ns;
  INPUT <= "111";
  wait for 50 ns;
  INPUT <= "000";
  WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

Declaración de variables

Asignación de las entradas/salidas a variables intermedias

Asignación de la secuencia valores de valores de entrada que queremos simular

El **Ejemplo4**, utiliza también una variable auxiliar **INPUT[2:0]** como en el **Ejemplo3**. Pero en este caso las distintas combinaciones de las variables de entrada se asignan mediante un bucle sentencia “for” del lenguaje VHDL, que se ejecuta con la periodicidad deseada (variable PERIOD). El índice entero de bucle (**i**) se hace corresponder con el valor en binario natural de cada una de las combinaciones posibles de las variables de entrada. Este entero es asignado a la variable **INPUT** empleando una función de librería **ieee.std_logic_arith.ALL** de VHDL, que permite convertir formatos (enteros a binario en este caso), como se señala en el ejemplo. A continuación, cada uno de los bit del vector **INPUT[2:0]** se asignan a las señales C, B y A.

Nota que una nueva librería (`ieee.std_logic_arith.ALL`) se ha incluido en la cabecera. Se han declarado las señales auxiliares A, B, C y S, y la constante PERIOD en la zona de declaración de variables, y se han conectado con los puertos entrada/salida del bloque. En la sección definida por el usuario se asignan secuencialmente las ocho combinaciones posibles de las variables de entrada. Para ello, se han definido, dos variables auxiliares: el índice entero `i`, que se hace variar entre 0 y 7, y el vector auxiliar `INPUT[2:0]`, que se utilizan para la conversión del índice entero a vector binario con la función de conversión (`CONV_STD_LOGIC_VECTOR(<valor>, <número de bits>)`). Cada uno de los bits de este vector se asigna despues a las señales C, B y A. Nota tambien que la sentencia `wait for PERIOD`; es la que permite espaciar el tiempo deseado los cambios en las entradas.

Se recomienda al estudiante que en esta tarea pruebe a generar los cuatro ficheros de vectores de test y compruebe que con cada uno de ellos es posible verificar el correcto funcionamiento del circuito.

Ejemplo 4

```

15 LIBRARY ieee;
16 USE ieee.std_logic_1164.ALL;
17 USE ieee.numeric_std.ALL;
18 USE ieee.std_logic_arith.ALL;          Librería Arith
19 LIBRARY UNISIM;
20 USE UNISIM.Vcomponents.ALL;
21 ENTITY sel2a1_sel2a1_sch_tb_ej4 IS
22 END sel2a1_sel2a1_sch_tb_ej4;
23 ARCHITECTURE behavioral OF sel2a1_sel2a1_sch_tb_ej4 IS
24 COMPONENT sel2a1
25   PORT( C : IN STD_LOGIC;
26         A : IN STD_LOGIC;
27         B : IN STD_LOGIC;
28         S : OUT STD_LOGIC);
29 END COMPONENT;
30 SIGNAL C : STD_LOGIC;                Declaración
31 SIGNAL B : STD_LOGIC;                de variables
32 SIGNAL A : STD_LOGIC;                auxiliares
33 SIGNAL S : STD_LOGIC;
34 constant PERIOD : Time := 50 ns;

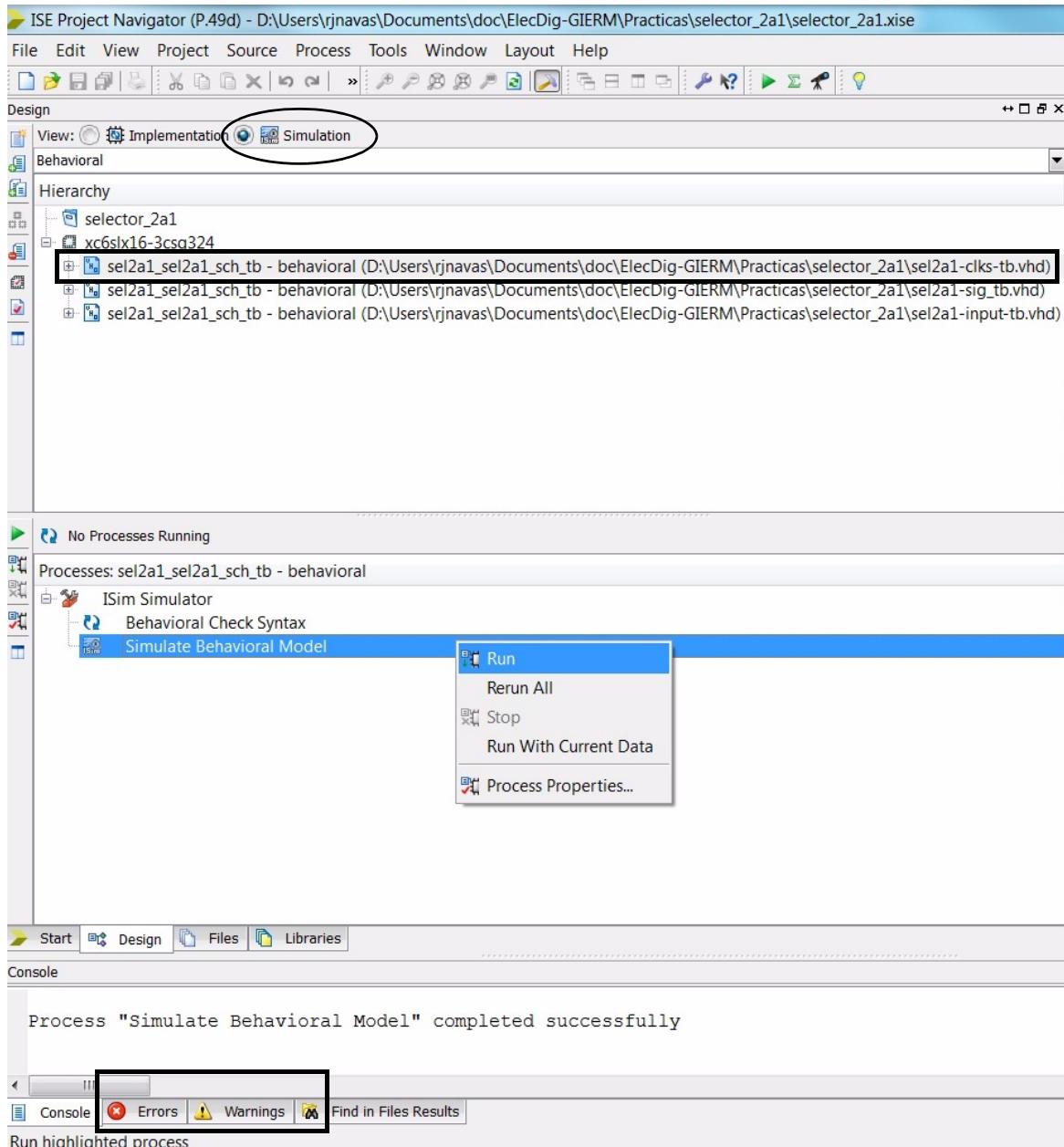
35
36 BEGIN
37   UUT: sel2a1 PORT MAP(            Asignación entrada/salida
38     C => C, B => B, A => A,        a variables intermedias
39     S => S
40 );
41 -- *** Test Bench - User Defined Section ***
42 tb : PROCESS
43   variable i : integer;
44   variable INPUT : std_logic_vector(2 downto 0);
45 BEGIN
46   for i in 0 to 7 loop
47     INPUT := CONV_STD_LOGIC_VECTOR(i,3);
48     C <= INPUT(2); B <= INPUT(1); A <= INPUT(0);
49     wait for PERIOD;
50   end loop;
51   INPUT := "000";
52   C <= INPUT(2);B <= INPUT(1);A <= INPUT(0);
53   WAIT;
54 END PROCESS;
55 -- *** End Test Bench - User Defined Section ***
56 END;

```

2. Ejecutar proceso de simulación.

Los pasos a seguir son los siguientes:

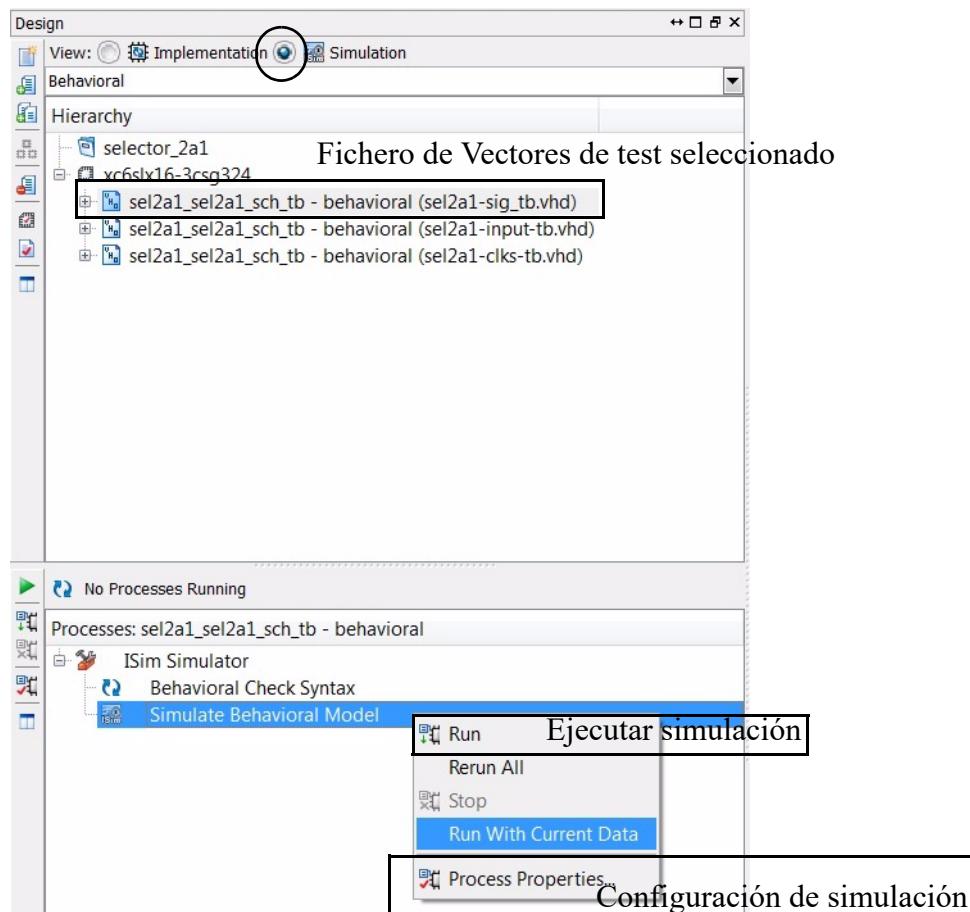
- Se selecciona la opción *Simulation* en la zona *view*.
- Se selecciona el fichero de test que se desea simular
- Se ejecuta el comando **Run** en la zona *process*



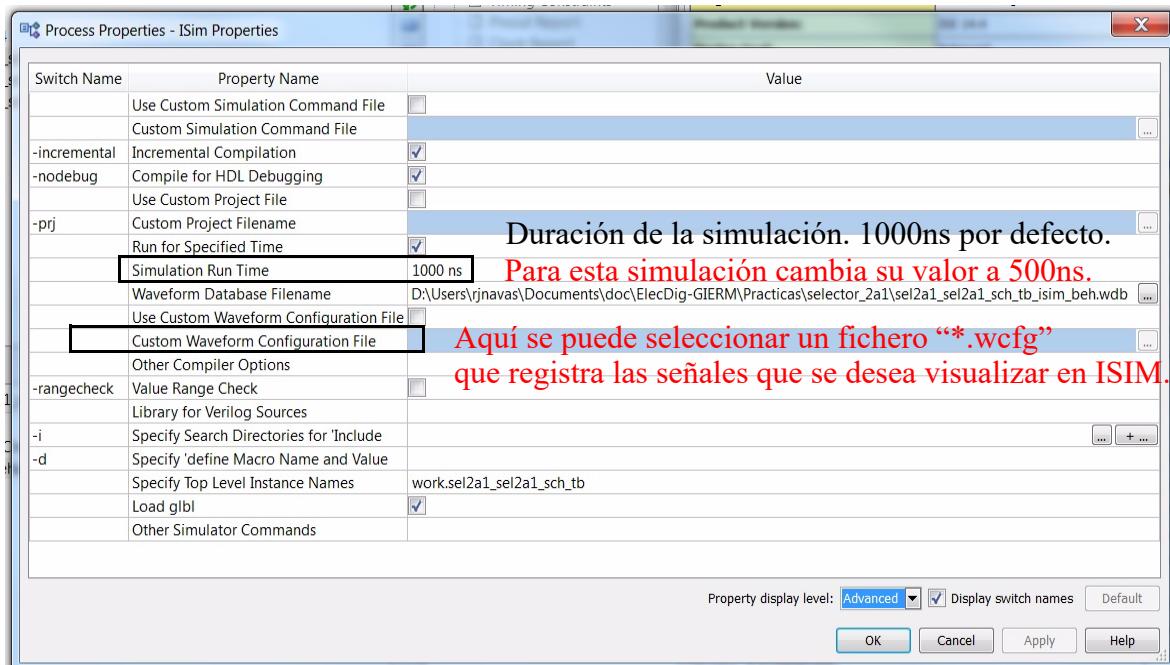
Si todo va bien, esto es, si no hay errores, ni en el esquemático, ni en el fichero de test, se lanza automáticamente la herramienta **ISim**, que permite ver los resultados de la simulación.

En caso contrario habrá que seleccionar la pestaña *Errors* e ir al *área de información* para ver que ha ido mal. Los *errores* o *warnings* más frecuentes están relacionados con errores en el dibujo: conectividad entre componentes (mal uso de entradas y salidas), asignación de nombres a cables o buses; o bien de errores sintácticos en la escritura de los vectores de test.

Algunos de los parámetros de simulación pueden seleccionarse y/o modificarse en el cuadro de diálogo, de preferencias de simulación “Process Properties”, antes de lanzar la simulación.



El cuadro de diálogo que aparece al seleccionar “Process Properties” es el siguiente:



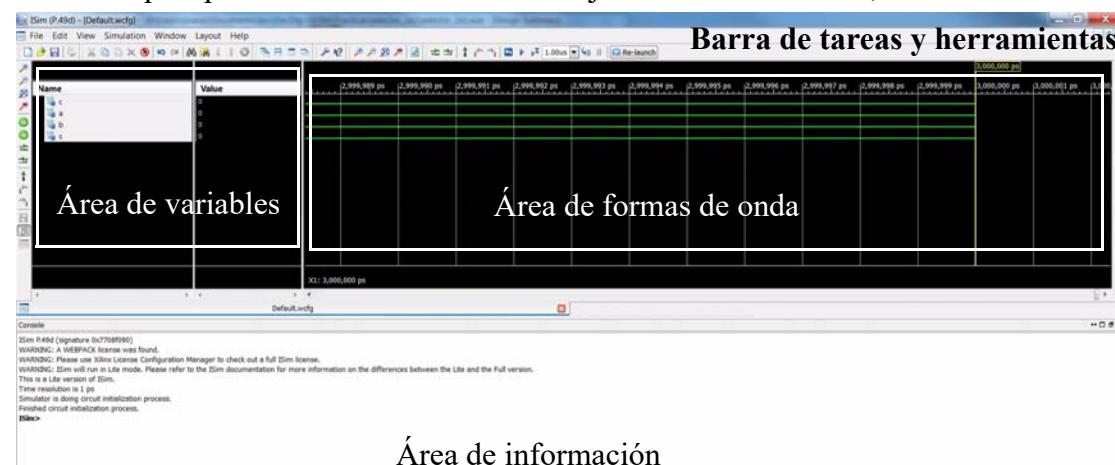
Es importante elegir adecuadamente este **tiempo de simulación** para ejercitar todos los vectores de test. La siguiente ecuación puede ser de utilidad.

$$t_{sim} \geq N \times CkSemiperiod$$

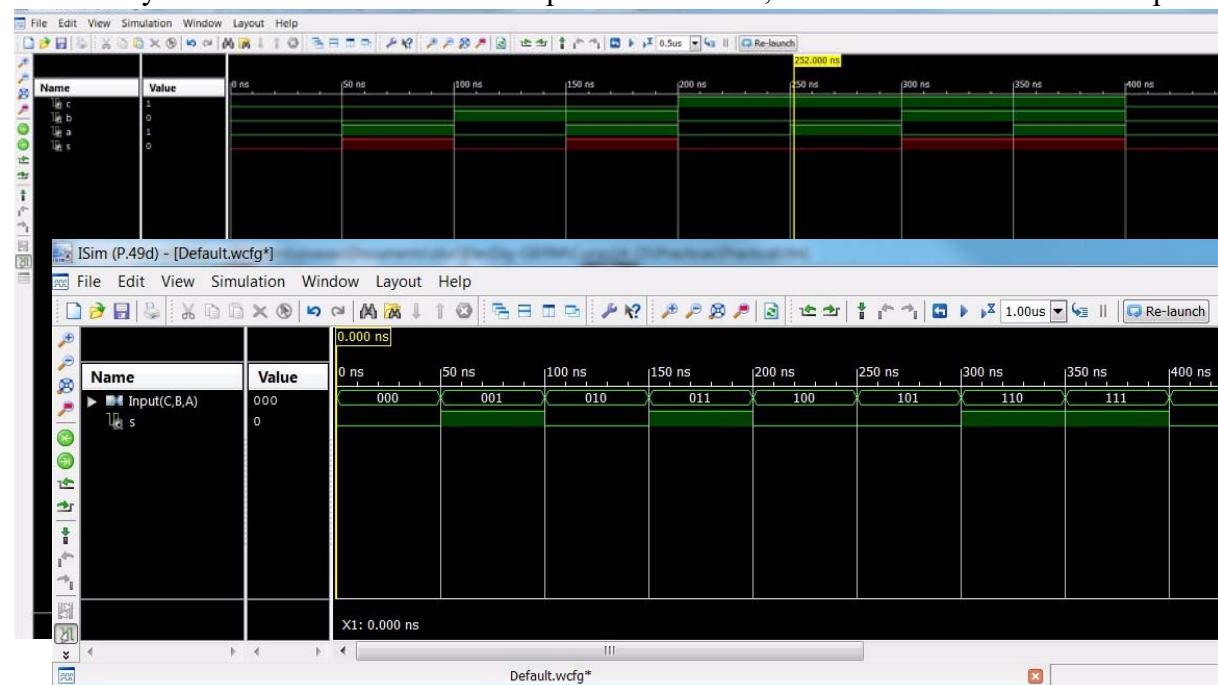
donde N es el número mínimo de vectores de test, o número de combinaciones de las variables de entrada.

3. Revisión del resultado de la simulación.

La herramienta **ISim** permite mostrar y analizar los resultados de la simulación. La siguiente pantalla es la que aparece automáticamente tras ejecutarse el simulador, si todo ha ido bien.



La figura muestra dos pantallas que resultan cuando se selecciona adecuadamente las señales a visualizar y se hace un zoom sobre el tiempo de simulación, haciendo un escalado de tiempos.



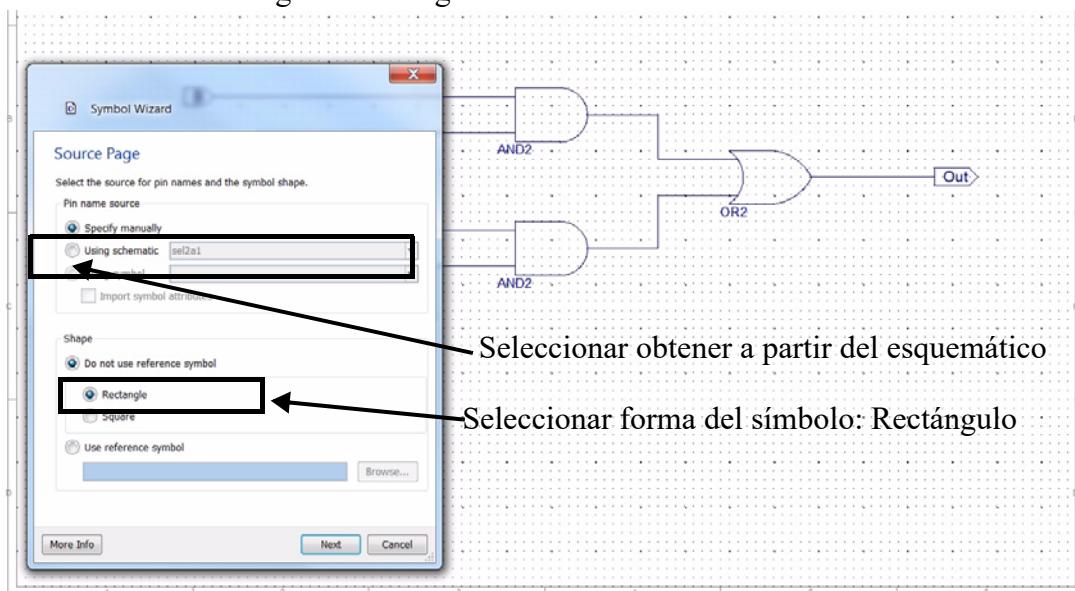
En la imagen de la parte superior se muestran las entradas C, B y A y la salida S como señales separadas, en el orden adecuado, siguiendo el cronograma previo, para verificar fácilmente y de forma visual el correcto funcionamiento. Se observa que cuando la entrada C está a cero, la salida S sigue los cambios de la variable A, mientras que cuando C está a uno, la salida sigue a la variable B. Como corresponde al cronograma del bloque **sel2a1**.

En la pantalla inferior las señales de entrada se han agrupado en una representación tipo bus, “virtual bus”, que permite una representación en binario (OCTAL o HEXA), del valor de la combinación de las variables de entrada. La configuración de esta pantalla se puede guardar en el correspondiente fichero “*.wcfg”.

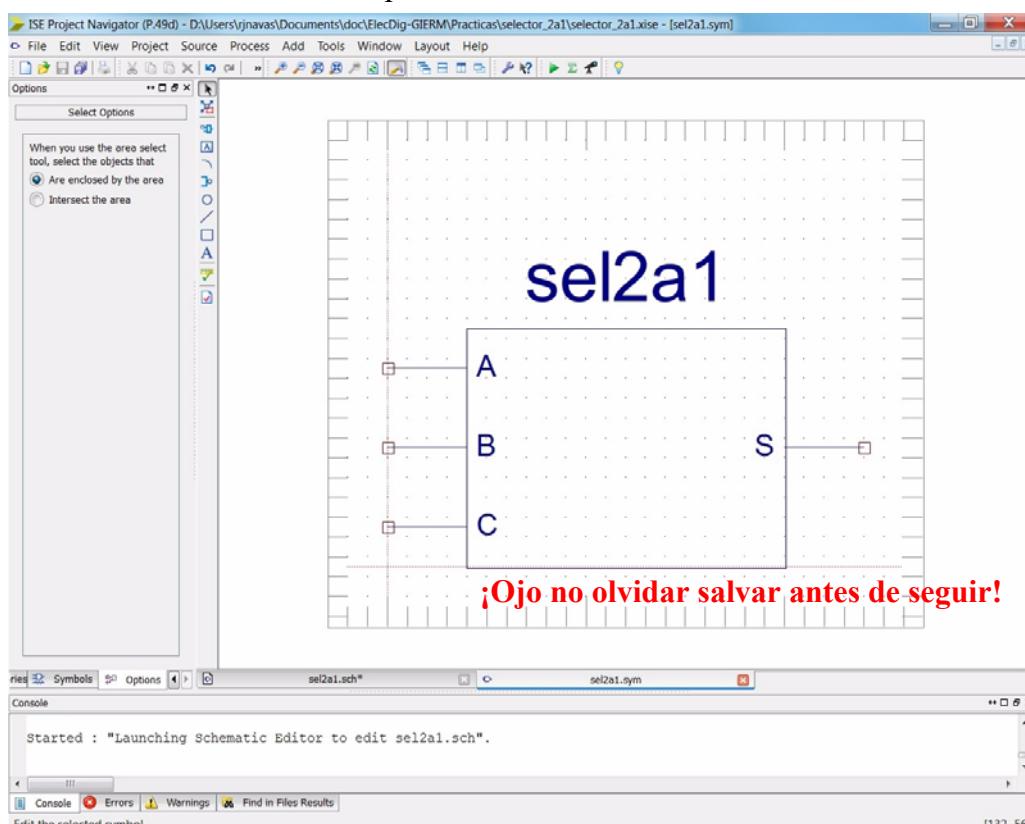
Para aprender más detalles sobre el uso de ISIM se recomienda revisar el documento **ug682.pdf** que se referencia en la bibliografía. Especialmente el Cap 3: *ISim GUI and Debugging the Design*. Muestra el resultado de tu simulación al profesor y él te dará alguna indicación adicional sobre las diferentes opciones y herramientas de **ISim**.

Tarea 6: Crear un Símbolo para selec_2a1.

Para poder reutilizar un bloque en un diseño o incorporarlo a nuevos proyectos, resulta conveniente generar un símbolo que lo represente. Esto puede hacerse fácilmente con la herramienta “*Symbol Wizard*” que puede ejecutarse en el desplegable “*Tools*” de la pantalla de inicio. Los cuadros de diálogo son los siguientes:



Tras un cuadro de diálogo que indica las entradas y salidas del bloque y su posición respecto al formato del símbolo seleccionado, se presenta el resultado en el *Editor de símbolos*.

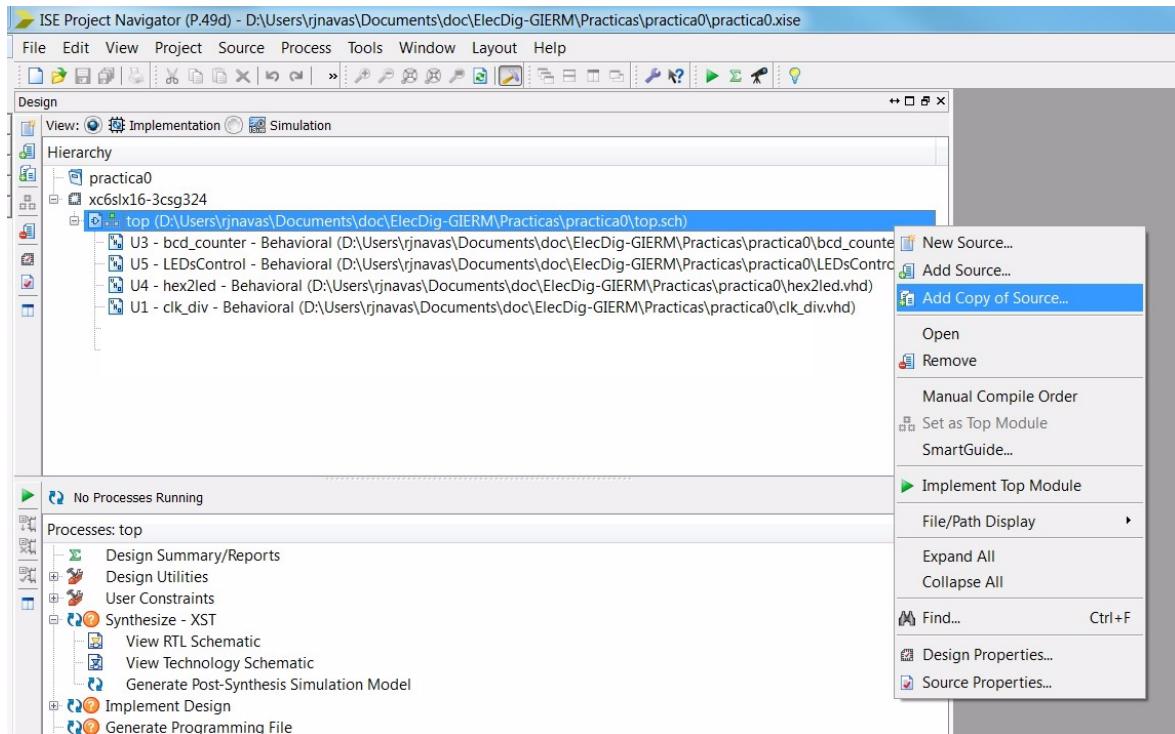


De esta forma, cuando este bloque sea incorporado a otro proyecto veremos nuestro bloque en la librería de componentes.

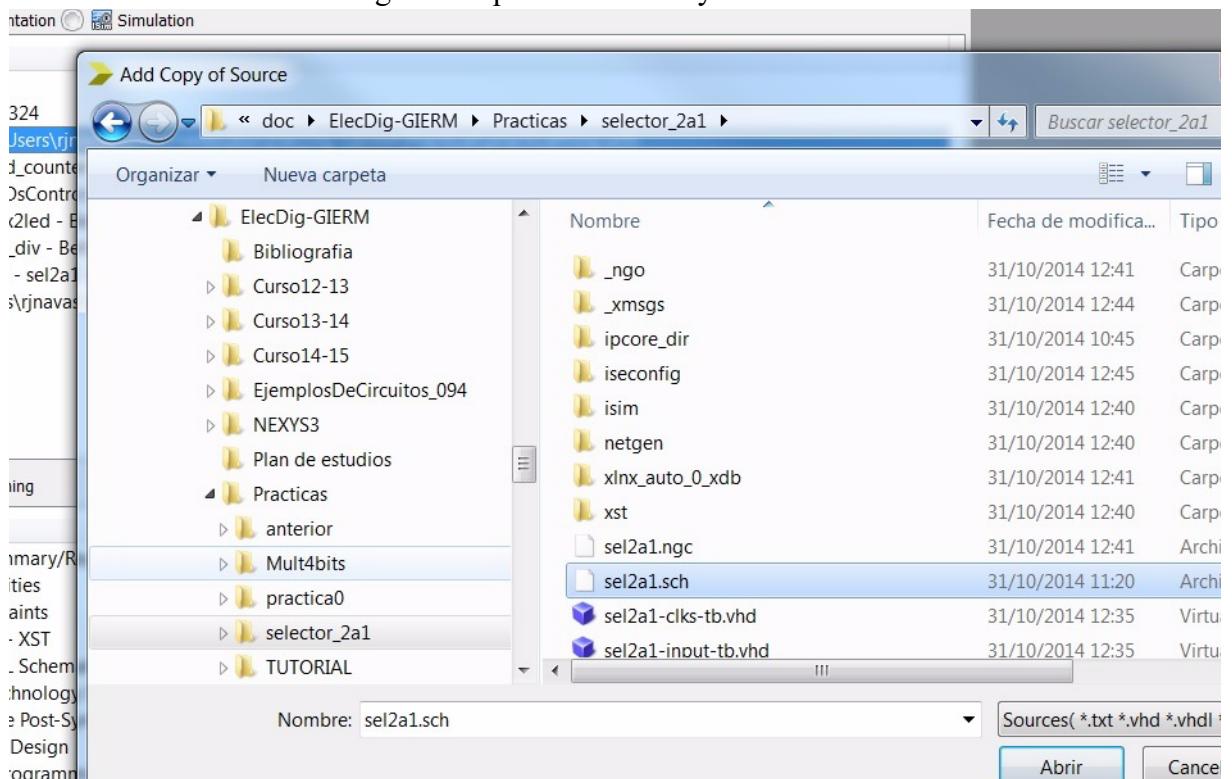
Tarea 7: Vuelta al proyecto “Practica0”:

1. Incorporación del bloque Selec_2a1

Asumimos que hemos completado y guardado el proyecto **selc_2a1**, y queremos ahora incorporar dicho elemento al proyecto “**Practica0**”. Abrimos pues ese proyecto, seleccionamos la pestaña “*Design*”, que nos lleva al árbol de jerarquía del proyecto, y a continuación seleccionamos la opción “Add Copy of Source” en el desplegable.



Esto abre un cuadro de diálogo en el que buscaremos y seleccionaremos el fichero **sel2a1.sch**.

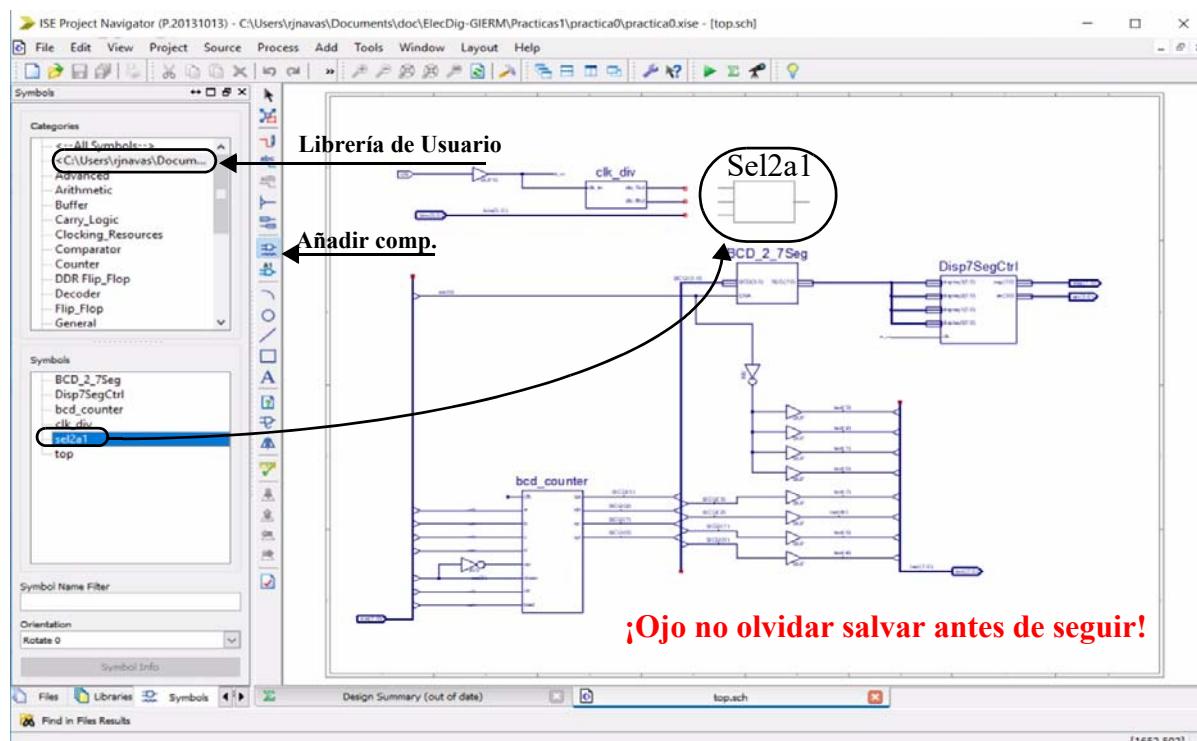


Este procedimiento incorpora una copia del elemento sel2a1 al presente proyecto, como podemos comprobar en la ventana “view”. Sin embargo, todavía este elemento no está incorporado al árbol de jerarquía, puesto que aún no se ha añadido a ninguna hoja de dibujo del proyecto. Cabe señalar que esta copia puede ser modificada, y salvada, aunque los cambios introducidos no se actualizarán en el fichero original.

Otra forma de incorporar elementos ya diseñados a un proyecto es seleccionar en el desplegable la opción “Add Source”. Se abre el mismo cuadro de diálogo que antes, para seleccionar el elemento. Sin embargo, en este caso, cualquier modificación en el elemento añadido, quedará también registrada en la fuente original. Así pues, **es importante distinguir entre estas dos formas de incorporar fuentes a un proyecto, pues tienen consecuencias distintas sobre los diseños ya hechos.**

Por su parte la opción “Add New Source” permite incorporar una nueva hoja de dibujo y por tanto un nuevo bloque de diseños. Este es el camino de la opción (2) indicada más arriba, antes del comienzo de la **Tarea 3**, en este tutorial.

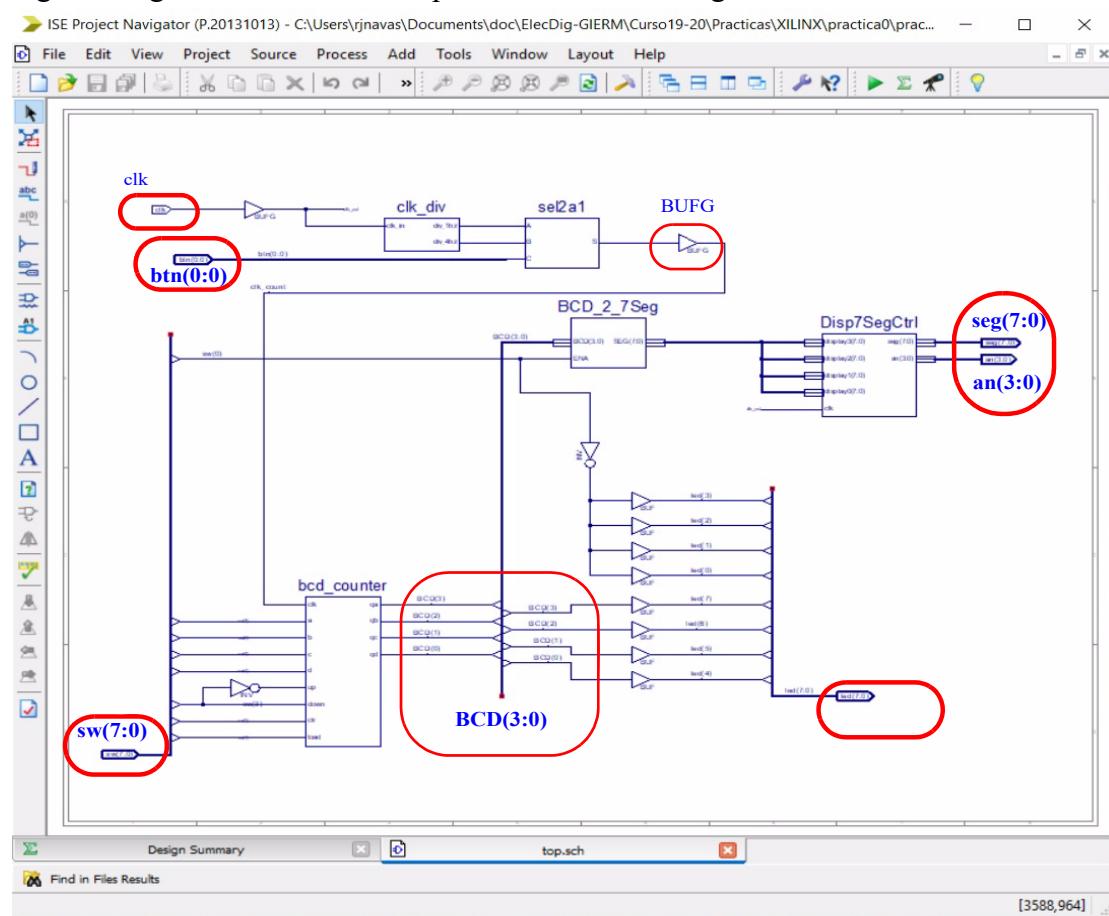
Dado que habíamos generado un símbolo para él, el elemento añadido también aparece como un nuevo componente que es posible seleccionar en la librería de componentes y añadir a una hoja de dibujo, como ilustra la figura.



2. Completar cableado.

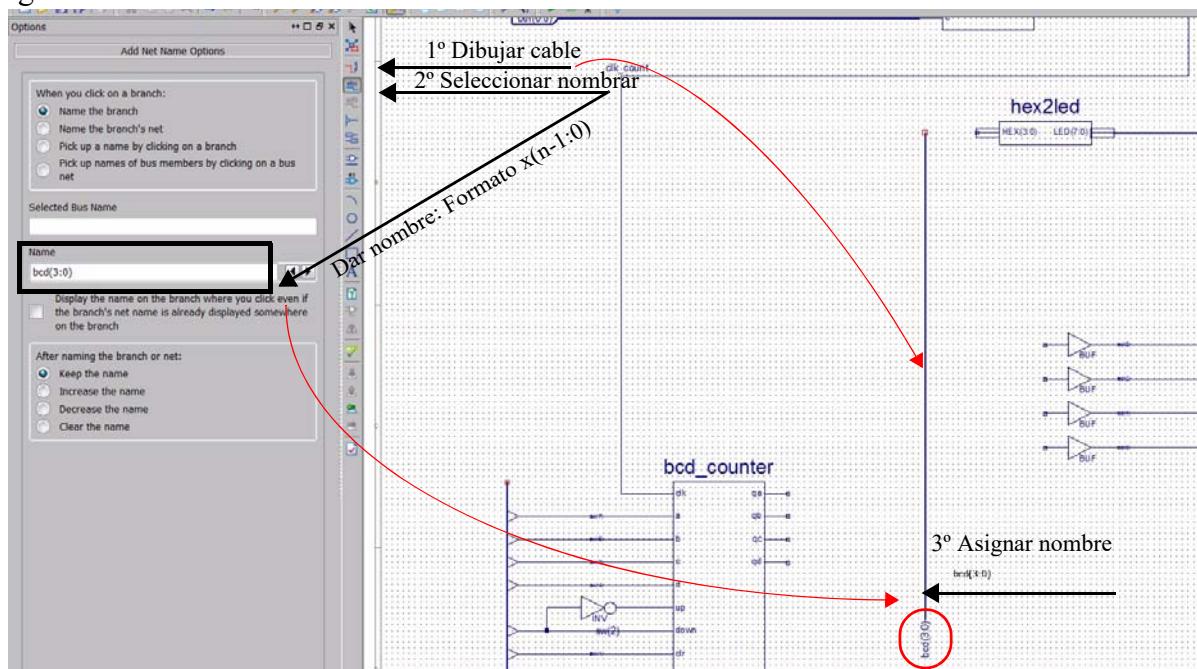
A continuación completamos el esquemático de la “**Practica0**” cableando el nuevo componente. Recordemos que la idea era utilizarlo para seleccionar la frecuencia del reloj que controla la cuenta del contador BCD. Nótese que además de los cables se ha añadido un nuevo **buffer de reloj** (BUFG) entre la salida de **sel2a1** y la entrada de reloj (**clk_count**) del contador. Esto es necesario para indicar que esa línea lleva información de reloj, y que en su implementación en la FPGA debe ser cuidada, además de evitar los avisos al respecto de la herramienta (warning). La entrada de control del selector se ha conectado al conector de entrada **btn(0:0)**. Nótese que aunque es una sola línea se ha nombrado con formato de bus. La razón es que así lo exige el correcto funcionamiento del programa para reconocer la conexión con el interfaz externo.

El diagrama lógico final debe ser el que se muestra en la figura.



3. Creación de buses: Agrupar señales en un bus. “Pinchar” un bus para sacar una señal.

Para completar el esquemático, tal y como aparece en la figura anterior, será necesario aprender a crear buses y a extraer una señal de un bus. El procedimiento general es el siguiente:

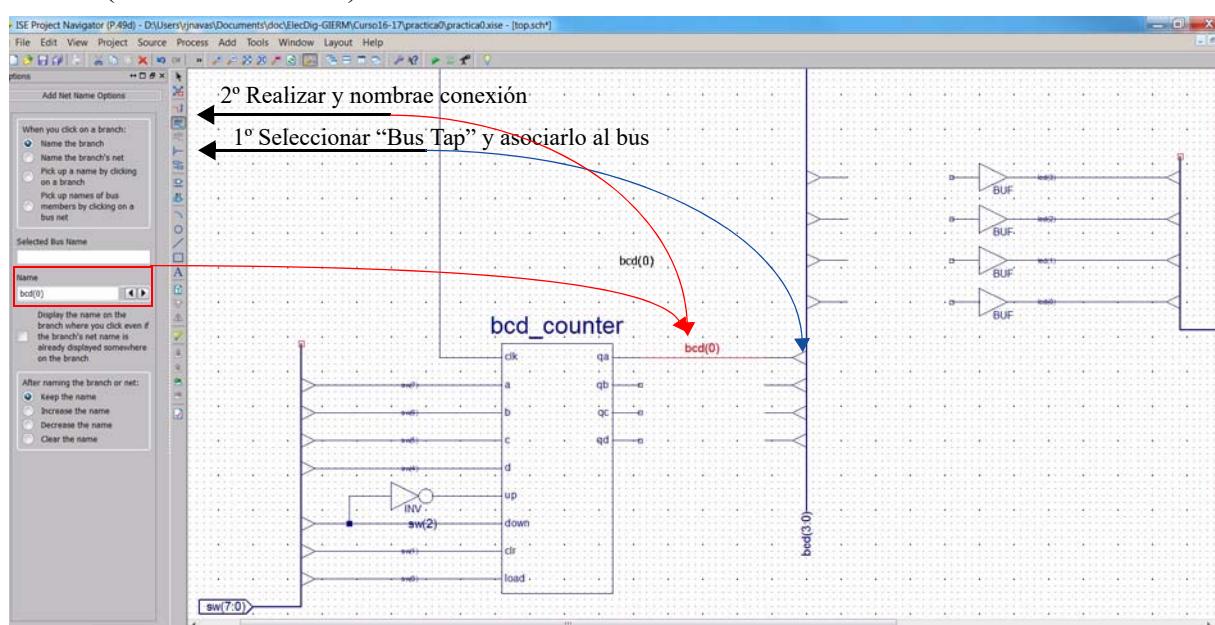


- 1º Seleccionar dibujar cable y trazar la línea correspondiente.
- 2º Seleccionar nombrar y asignar nombre en formato bus $x(n-1:0)$, donde n es el número de líneas del bus. En este caso llamamos **bcd(3:0)**, dado que en este bus queremos agrupar las cuatro líneas de salida del contador BCD qa, qb, qc y qd. De esta forma cada línea individual agrupada queda nombrada como bcd(3), bcd(2), bcd(1) y bcd(0), y así se llamaran cuando nos refiramos a cada línea individualmente.
- 3º Asociar el nombre al bus. Seleccionar con el cursor y hacer “click” sobre el cable que se quiere nombrar. Al asignar el nombre, la línea dibujada, inicialmente delgada, pasa a ser más gruesa, indicando que se trata de un bus.

Tanto para agrupar señales en un bus, como para extraer una línea de un bus se utilizan los elementos llamados “Bus Tap”. (Tienen forma de triángulo, con una línea de conexión en uno de sus vértice. Esta línea se conecta a la línea individual sobre la que se opera, mientras que el lado opuesto del triangulo se apoya sobre el bus al que se asocia).

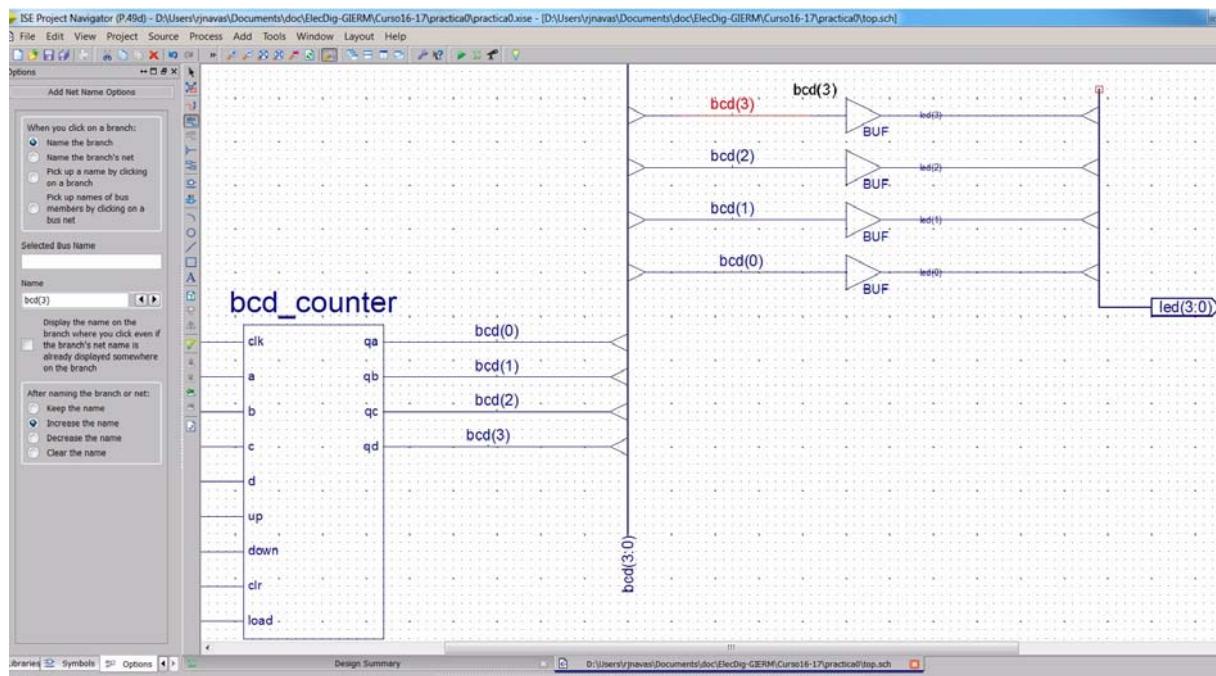
El procedimiento para crear conexiones a un bus puede resumirse en los siguientes pasos:

- 1º Seleccionar y añadir “Bus Tap”. Se añade uno por cada línea que se agrupa o se extrae del bus. Estos elementos colocan sobre el bus y se pueden orientar a derecha, izquierda, arriba o abajo, para colocarlos donde mejor convenga a la hora de realizar las conexiones.
- 2º Cablear cada “Bus Tap” a la correspondiente línea que se le quiere asociar y nombrar dicha línea con el nombre de la línea de bus a la que se asocia. En este ejemplo bcd(3), bcd(2), bcd(1) y bcd(0). Nótese que la conexión se entenderá de agrupamiento o extracción de una línea del bus según se trate de una conexión a una línea que provenga de una salida de componente (agrupación en bus) o vaya a la entrada de otro componente (extracción del bus).



El procedimiento se repite hasta completar el diseño.

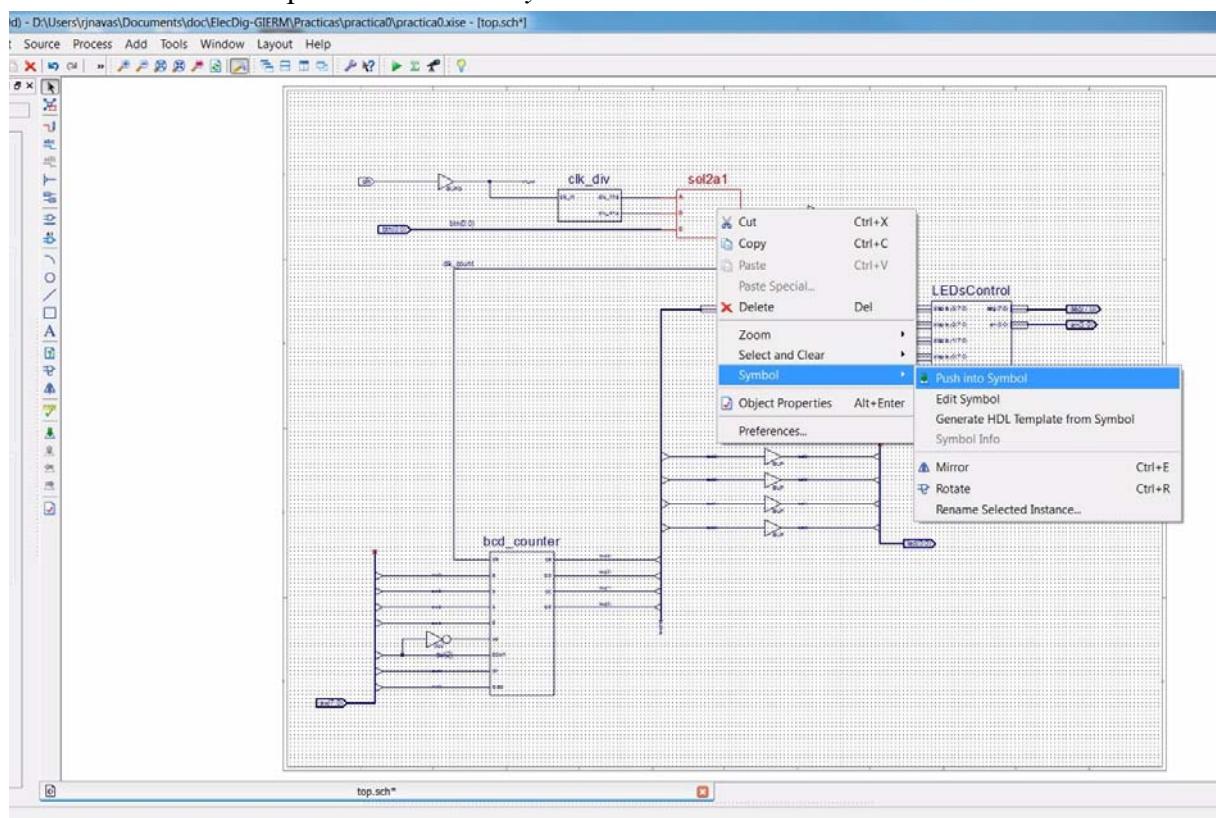
El resultado final es el que aparece en la siguiente figura.



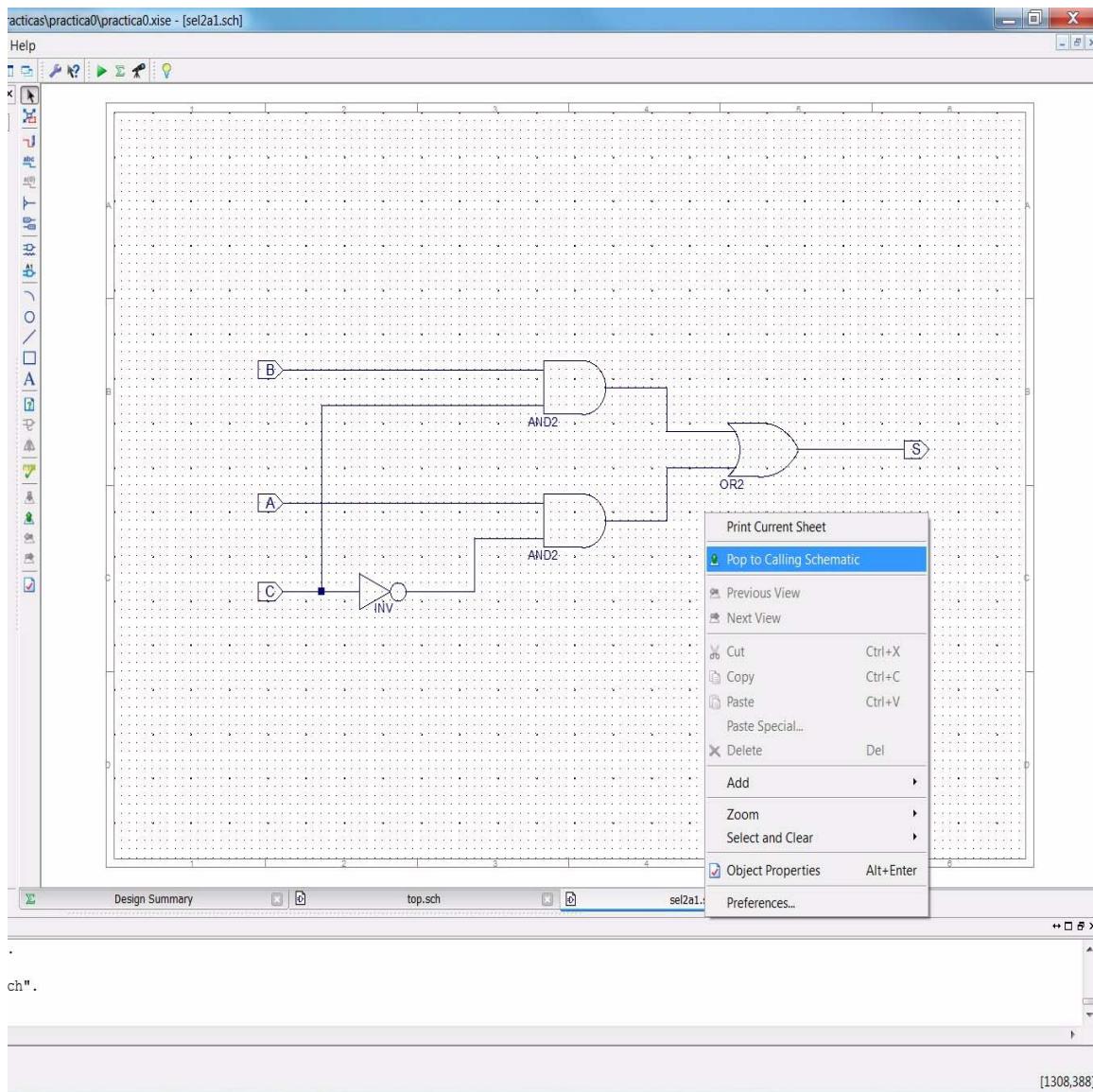
Para completar el diseño, no olvides conectar el bus **bcd(3:0)** a la entrada del bloque **hex2led**.

4. Navegar por la jerarquía de diseño.

El contenido de cualquier bloque de la jerarquía de diseño puede verse seleccionando en el dibujo actual el símbolo correspondiente y en el desplegable que aparece al pulsar el botón derecho del ratón la opción “*Push into Symbol*”.



Por ejemplo, de esta forma, desde la hoja de dibujo practica0.sch, podemos ver el contenido del bloque sel2a1. Al ejecutar esta acción, bajamos en la jerarquía y se abre la hoja de esquemático que contiene su diagrama lógico.



De la misma manera puede regresarse al nivel de jerarquía superior. Seleccionando en el desplegable, que aparecerá de la misma manera la opción “*Pop to calling Schematic*”.

Si el símbolo seleccionado corresponde a un elemento especificado mediante una descripción VHDL, entonces en el área de trabajo se abrirá el fichero de texto que contiene dicha descripción.

Tarea 8: Incorporación al proyecto del fichero de configuración de pines FPGA:

Antes de lanzar los procedimientos de síntesis e implementación, que llevarán a la generación del fichero de programación de la FPGA (*.bit), es necesario incorporar al proyecto el fichero **Nexys3-Master.ucf**. Este es un fichero proporcionado por el fabricante y que contiene una relación de todos los pines que constituyen el interfaz entrada/salida de la FPGA utilizada en la placa Nexys3. Este fichero se incorpora siguiendo el procedimiento “Add New Source” descrito con anterioridad.

Una copia de este fichero debe ser incorporada a todo proyecto que vaya a ser implementado en la placa Nexys3. Y debe ser modificado de manera que queden configuradas y asignadas apropiadamente las señales del interfaz que van a ser utilizadas.

La figura muestra un extracto de este fichero, tal y como debe quedar en el proyecto “Practica0”. Las líneas que comienzan con el símbolo “#” son consideradas como comentario.

```

Project Navigator (P.49d) - D:\Users\jnavas\Documents\doc\ElecDig-GIERM\Practicas\practica0\practica0.xise - [Nexys3_Master.ucf]
Edit View Project Source Process Tools Window Layout Help
File Project Source Process Tools Window Layout Help
1 ## This file is a general .ucf for Nexys3 rev B board
2 ## To use it in a project:
3 ## - remove or comment the lines corresponding to unused pins
4 ## - rename the used signals according to the project
5
6 # Clock signal
7 NET "clk"          LOC = "V10" | IOSTANDARD = "LVCMS33"; #Bank = 2, pin name = IO_L30N_GCLK0_USERCCLK,
8 Net "clk" TNM_NET = sys_clk_pin;
9 TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;
108
109 # 7 segment display
110 NET "seg<0>"    LOC = "T17" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L51P_M1DQ12,
111 NET "seg<1>"    LOC = "T18" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L51N_M1DQ13,
112 NET "seg<2>"    LOC = "U17" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L52P_M1DQ14,
113 NET "seg<3>"    LOC = "U18" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L52N_M1DQ15,
114 NET "seg<4>"    LOC = "M14" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L53P,
115 NET "seg<5>"    LOC = "N14" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L53N_VREF,
116 NET "seg<6>"    LOC = "L14" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L61P,
117 NET "seg<7>"    LOC = "M13" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L61N,
118
119 NET "an<0>"     LOC = "N16" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L50N_M1UDQSN,
120 NET "an<1>"     LOC = "N15" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L50P_M1UDQS,
121 NET "an<2>"     LOC = "P18" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L49N_M1DQ11,
122 NET "an<3>"     LOC = "P17" | IOSTANDARD = "LVCMS33"; #Bank = 1, Pin name = IO_L49P_M1DQ10,
123
124 # Leds
125 NET "Led<0>"   LOC = "U16" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L2P_CMPCLK,
126 NET "Led<1>"   LOC = "V16" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L2N_CMPMOSI,
127 NET "Led<2>"   LOC = "U15" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L5P,
128 NET "Led<3>"   LOC = "V15" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L5N,
129 #NET "Led<4>"   LOC = "M11" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L15P,
130 #NET "Led<5>"   LOC = "N11" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L15N,
131 #NET "Led<6>"   LOC = "R11" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L16P,
132 #NET "Led<7>"   LOC = "T11" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L16N_VREF,
133
134
135 # Switches
136 NET "sw<0>"   LOC = "T10" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L29N_GCLK2,
137 NET "sw<1>"   LOC = "T9" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L32P_GCLK29,
138 NET "sw<2>"   LOC = "V9" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L32N_GCLK28,
139 NET "sw<3>"   LOC = "M8" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L40P,
140 NET "sw<4>"   LOC = "N8" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L40N,
141 NET "sw<5>"   LOC = "U8" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L41P,
142 NET "sw<6>"   LOC = "V8" | IOSTANDARD = "LVCMS33"; #Bank = 2, Pin name = IO_L41N_VREF,
143 NET "sw<7>"   LOC = "T5" | IOSTANDARD = "LVCMS33"; #Bank = MISC, Pin name = IO_L48N_RDWR_B_VREF_2,
144
145
146 # Buttons
147 NET "btn<0>"  LOC = "B8" | IOSTANDARD = "LVCMS33"; #Bank = 0, Pin name = IO_L33P,
148 #NET "btn<1>"  LOC = "A8" | IOSTANDARD = "LVCMS33"; #Bank = 0, Pin name = IO_L33N,
149 #NET "btn<2>"  LOC = "C4" | IOSTANDARD = "LVCMS33"; #Bank = 0, Pin name = IO_L1N_VREF,
150

```

Se ha quitado este símbolo de las líneas que corresponden a las señales que constituyen los puertos entrada/salida del proyecto: La señal de reloj de la placa “clk”, los displays 7-segmentos “seg y an”, los Ledes “led”, los interruptores “sw” y el pulsador 0 “btn0”.

Tarea 9: Síntesis e implementación: generación fichero de programación del Proyecto0

Estos dos procesos se ejecutan secuencialmente.

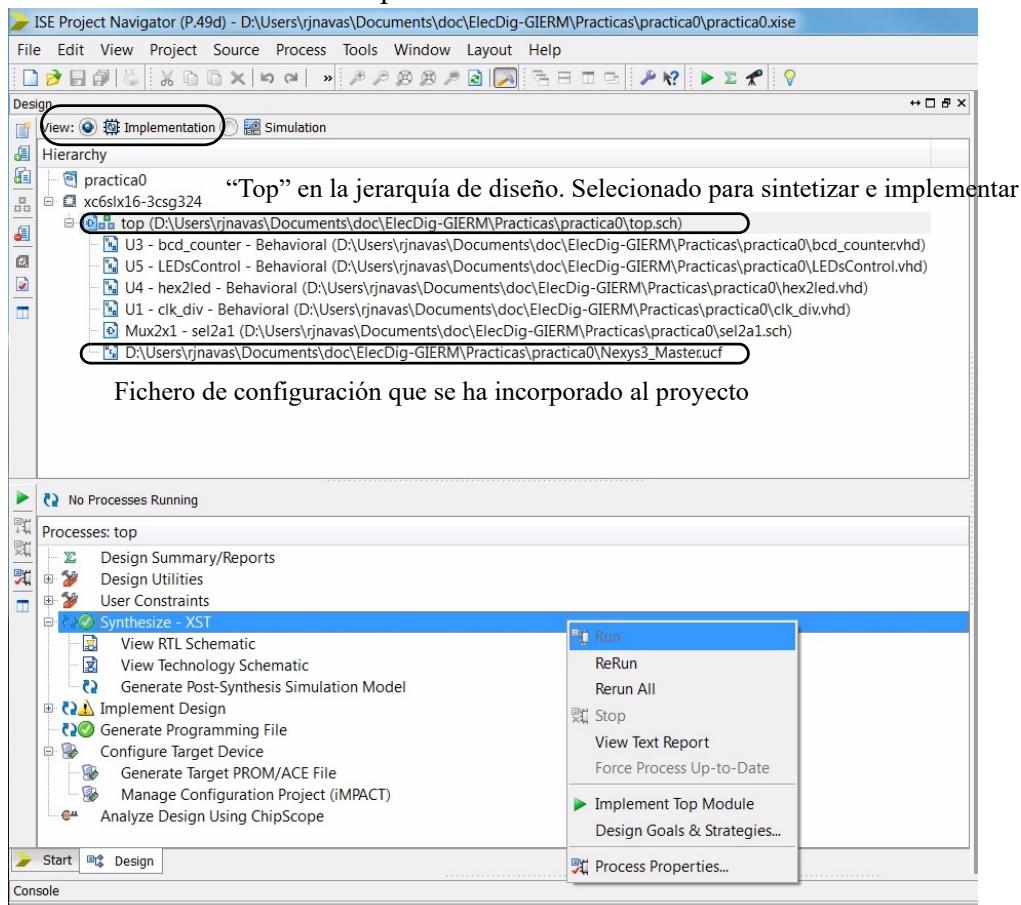
El proceso de **Síntesis** es el encargado de trasladar el diseño que hemos realizado, a un formato que representa la forma concreta en la que el sistema va a ser realizado físicamente en la FPGA de la placa Nexys3. Genera un modelo (modelo post-síntesis) para ser utilizado en una **simulación temporal** del circuito, en la que podemos verificar que no hay problemas en él debido al retardo de propagación de las señales.

Inicialmente, en nuestras prácticas, vamos a ejecutar directamente el procedimiento de **Implementación**. Este proceso se encarga automáticamente de revisar si se ha ejecutado el procedimiento de **Síntesis** y si no es así, lo ejecutará antes de pasar a la **Implementación** y finalmente a la generación del fichero de programación.

El proceso de **Generating Programming File** se encarga de generar un fichero con el mismo nombre que el elemento raíz de la jerarquía de diseño (top) y con la extensión “.bit”. Este fichero contiene la información necesaria para programar la FPGA. En la siguiente sección veremos como se carga este fichero en la FPGA.

Durante la ejecución de estos procesos pueden aparecer errores y/o alertas (“warning”). Los primeros impiden continuar el proceso y habrán de ser revisados y corregidos; los segundos no abortan el procedimiento, aunque según sea su causa, pueden dar lugar o no a problemas en la implementación, por lo que es conveniente, al menos, identificarlos.

La figura muestra como lanzar estos procedimientos.

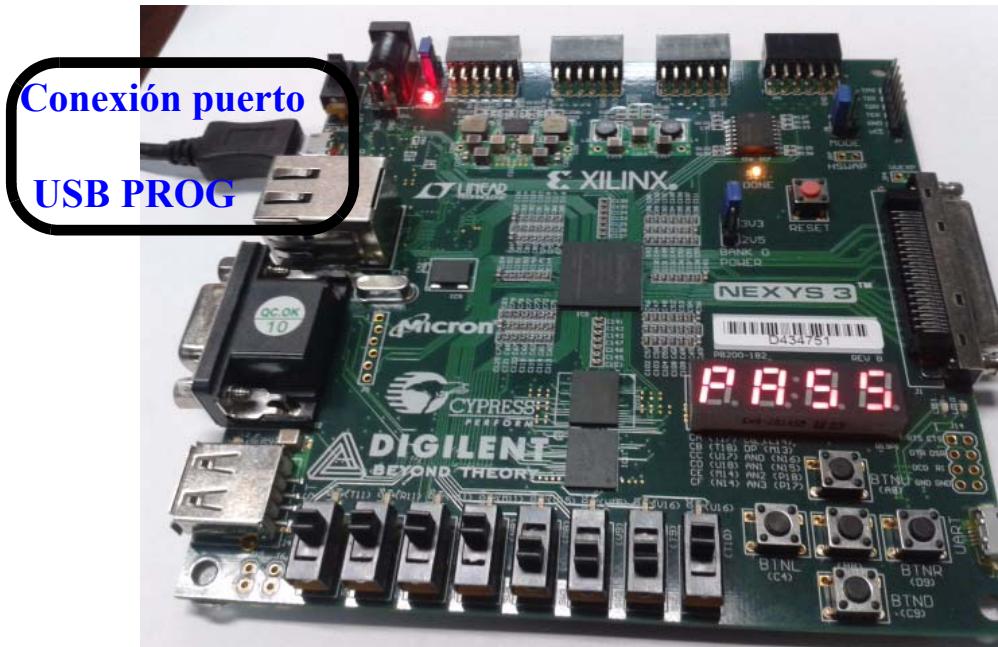


En primer lugar se selecciona el elemento raíz (top) de la jerarquía de diseño que se va a implementar en la ventana *view* del *área de gestión*, en la que está marcada la opción *Implementación*. A continuación se lanza el proceso en la ventana *process*. Una manera de hacer esto es seleccionar el proceso que se desea ejecutar y seleccionar la opción run en el desplegable de opciones que se abre pulsando el botón derecho del ratón. Los posibles errores se indican en el *área de información*.

Nota: En nuestras prácticas y proyectos no vamos ha realizar simulaciones temporales, esto es simulaciones que tengan en cuenta el retardo de propagación de las señales en los bloques considerados. Asumiremos que este retardo es pequeño, comparado con el periodo de la señal de reloj con la que se trabaja. Por ello solo realizarémos simulaciones de comportamiento.

Tarea 10: Programación de la placa Nexys3 con el Proyecto0. Verificación.

La figura muestra la plataforma Nexys 3 una vez conectada al PC, por tanto, alimentada y sin programar. La conexión se realiza mediante un cable USB a uno cualquiera de los puertos disponibles en él.



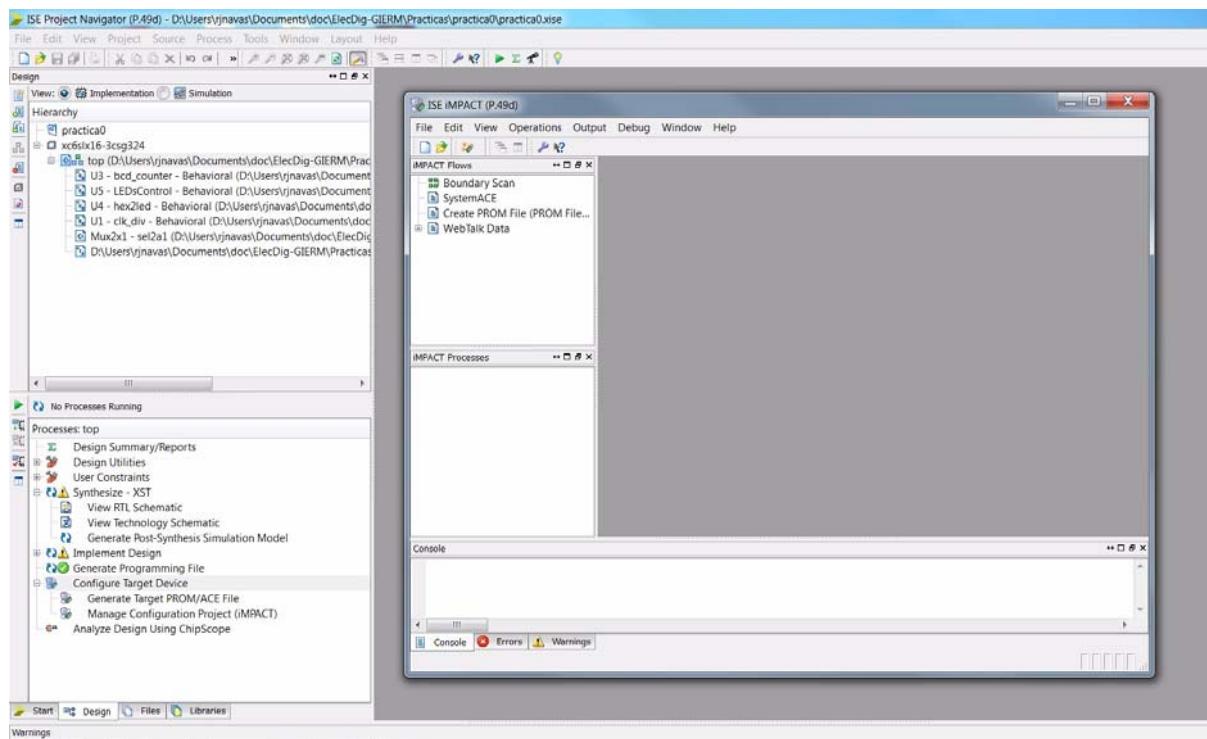
Nota que la conexión a la placa se realiza por el conector mini-USB, y en el conector de la placa marcado como USB PROG. Cuando la FPGA está sin programar los displays muestra la palabra PASS o números.

1. Iniciar programación.

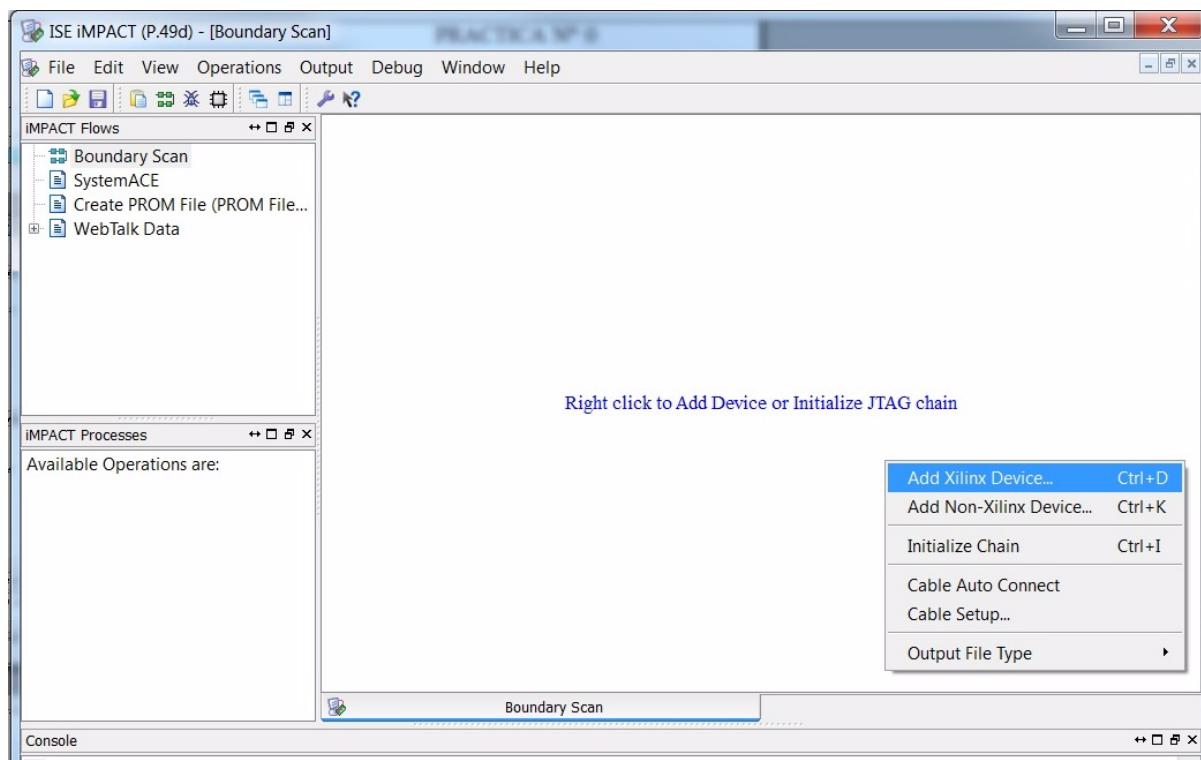
El último paso es ejecutar el procedimiento “**Manage Configuration Project**” en “Configure Target Device”. Este procedimiento vuelca el contenido del fichero de programación (top.bit) en la FPGA de la placa Nexys3.

Tras seleccionar y ejecutar este procedimiento aparece una secuencia de cuadros de diálogo que hay que ir completando. Los principales pasos se muestran la siguiente serie de figuras.

Tras la ejecución aparece.

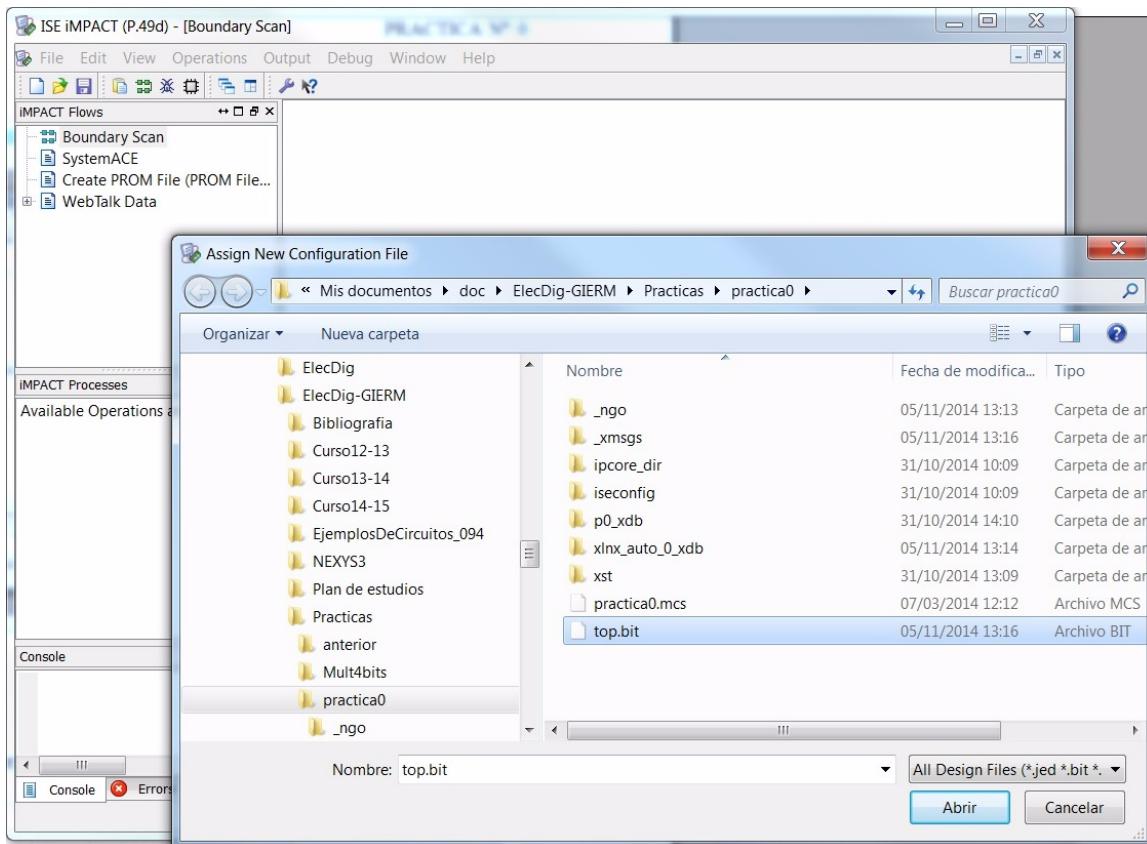


Tras ejecutar “Boundary Scan” aparece

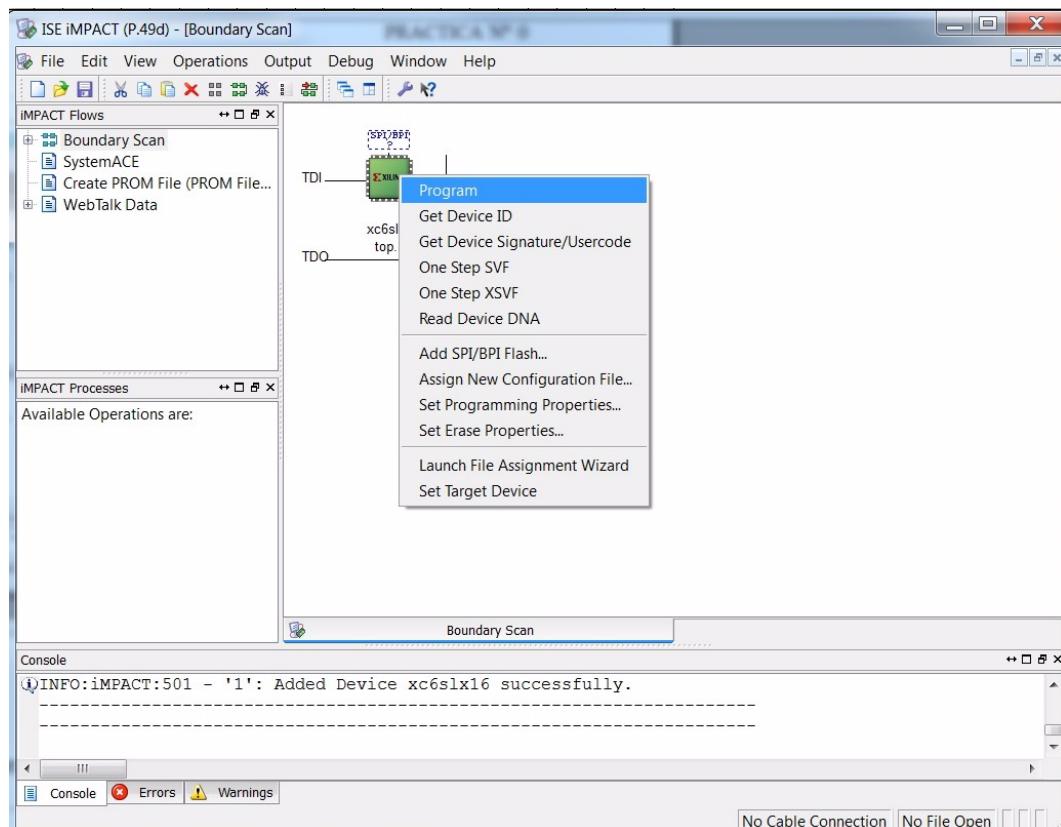


Se selecciona *Add Xilinx Device* en el desplegable.

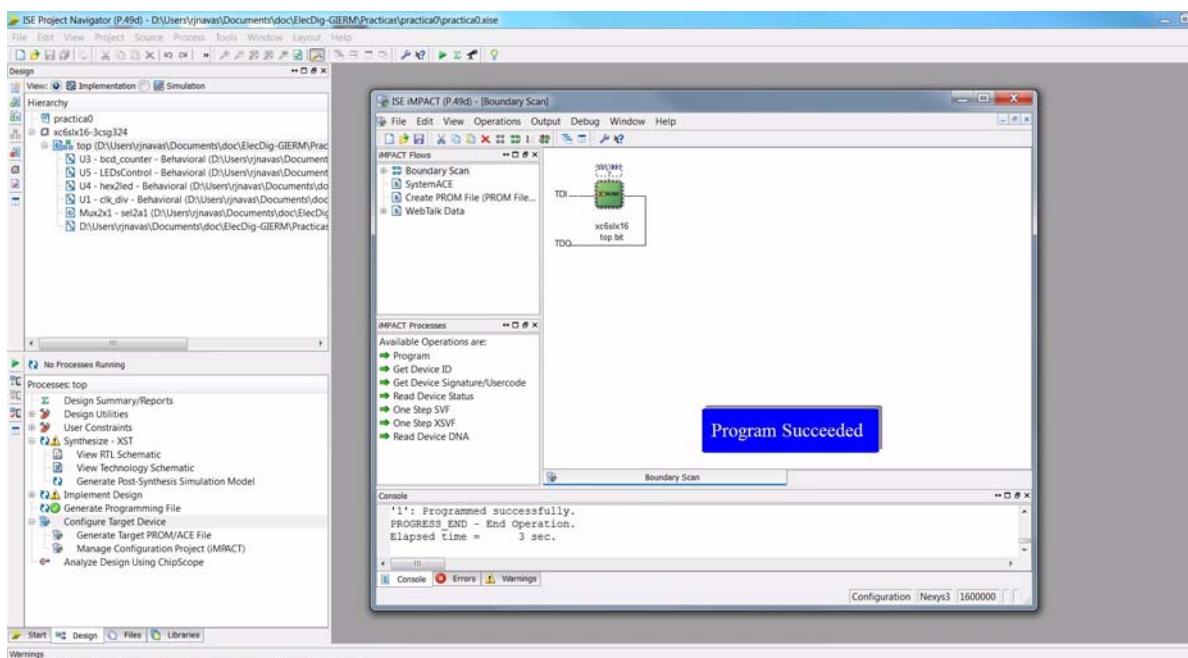
Se selecciona el fichero de programación en el directorio del proyecto.



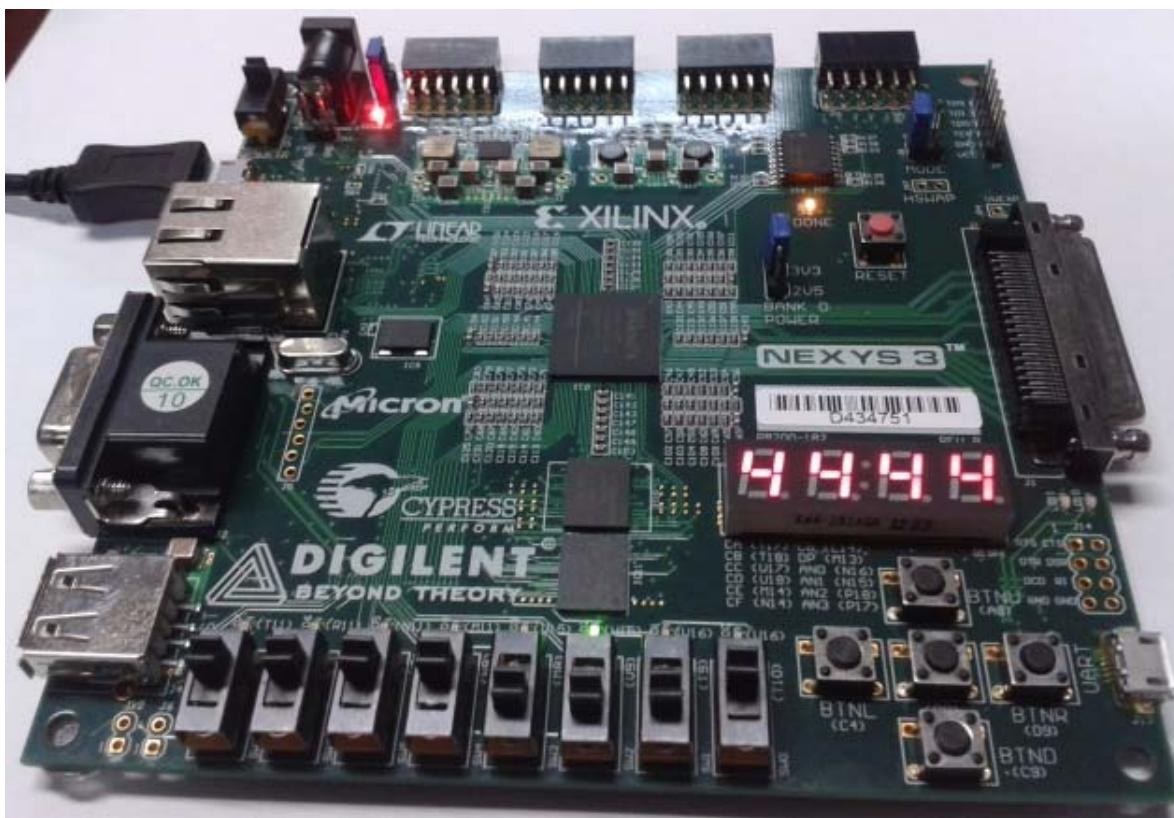
Se selecciona la opción *Program* en el desplegable que aparece al seleccionar el ícono del dispositivo.



2. Indicación de Programación terminada



Placa programada ejecutando en diseño “Practica0”.



3. Verificación del funcionamiento

Actuando sobre los interruptores y botones: Comprobar la funcionalidad: Correcta visualización de cuenta: up/down. Carga de valor inicial etc. Control de velocidad de cuenta con btn0.

Tareas y Presentación de resultados:

Esta práctica se ha dividido en 10 tareas, para desarrollarlas durante las tres primeras sesiones de laboratorio de este curso:

Tarea 1: Arranque y presentación del programa ISE Project Navigator

Tarea 2: Apertura y visualización de ficheros de un proyecto ya existente: practica0

Tarea 3: Crear y configurar un nuevo proyecto selec_2a1

Tarea 4: Creación de nueva fuente. Dibujar el esquemático selec_2a1

Tarea 5: Simulación: verificación funcional del diseño selec_2a1

Tarea 6: Crear un Símbolo para selec_2a1

Tarea 7: Vuelta al proyecto “Practica0”: Incorporación de selec_2a1.

Tarea 8: Incorporación al proyecto del fichero de configuración de pines FPGA

Tarea 9: Síntesis e implementación: generación fichero de programación del Proyecto0

Tarea 10: Programación de la placa Nexys3 con el Proyecto0. Verificación in situ.

Se recomienda el siguiente esquema temporal para la ejecución de las tareas: durante la primera sesión deberías completar sin dificultad las tareas 1 a 4, y haber comenzado la tarea 5; para la segunda sesión deberías poder completar las tareas 5, 6, y 7; y durante la tercera el resto de tareas junto con la presentación de resultados.

Es importante que prepares cada sesión de laboratorio leyendo con detenimiento el material proporcionado, y despejando la mayor cantidad de dudas.

Recuerda que el objetivo de esta práctica es que des tus primeros pasos en el entorno de desarrollo de Xilinx, diseño, simulación hasta llegar a implementar el proyecto propuesto en la placa de desarrollo y comprobar su funcionamiento.

Los resultados que hay que presentar al profesor en esta práctica son:

1º Mostrar el esquemático del bloque selec_2a1, así como las simulaciones realizadas para validar su funcionalidad (utilizando al menos dos de los métodos propuestos para escribir los vectores de test).

2º Describir el proyecto0, navegando por su jerarquía de diseño e indicando la funcionalidad del conjunto y de cada uno de los bloques que lo componen.

3º Realizar una demostración práctica del funcionamiento del sistema implantado en la plataforma Nexys 3.

Bibliografía:

- Documentación Nexys 3 Spartan-6 FPGA Board. <http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,897&Prod=NEXYS3>
- Documentos de la carpeta XLINX DOC de la asignatura de campus virtual.
<http://industriales.cv.uma.es>
Especialmente los documentos **ise_tutorial_ug695.pdf**, **ug682.pdf** para el entorno de desarrollo y **xapp.199.pdf** para escribir vectores de test.