

Manual de la asignatura Laboratorio de Tecnología de Computadores

Hortensia Mecha López
José Luis Risco Martín

Copyright ©2010 por Universidad Complutense de Madrid

Todos los derechos reservados.

ISBN 978-84-693-6326-3

Índice general

1. Manual del Xilinx ISE	1
1.1. Primeros pasos	1
1.1.1. Ejecutar la aplicación	1
1.1.2. Acceso a la ayuda	2
1.2. Creación de un nuevo proyecto	3
1.2.1. Datos del proyecto	4
1.2.2. Datos del dispositivo	4
1.2.3. Ficheros fuente del proyecto	5
1.3. Apertura de un proyecto existente	6
1.4. Creación del esquemático	8
1.4.1. Buses	11
1.4.2. Arrays de instancias	13
1.5. Simulación del diseño	14
1.5.1. Generación de un fichero de estímulos (Testbench)	15
1.5.2. Simulación de estímulos complejos de señales	16
1.5.3. Simulación de un Testbench	17
1.6. Generación de un símbolo a partir de un fichero VHDL	20
1.7. Generación de un fichero de restricciones *.ucf	21
1.8. Implementación y Generación del Mapa de bits.	25
1.9. Descarga del diseño en la placa XSA-3S1000	26
2. ¿Cómo ...?	27
2.1. Cómo instalar Xilinx 10.1 en MS Windows 7 64 bits	27
2.2. Cómo guardar el conjunto de señales de una simulación	27

Capítulo 1

Manual del Xilinx ISE

1.1. Primeros pasos

1.1.1. Ejecutar la aplicación

Para abrir el entorno de Xilinx ISE, pulsar el siguiente icono del escritorio:



,o arrancar la aplicación desde el menú **Inicio** seleccionando:

Inicio→**Programas**→**Electronica**→**Xilinx ISE 10.1**→**Project Navigator**

Nos aparecerá una ventana como la que se muestra en la Figura 1.1. La ventana tiene un total de cuatro ventanas secundarias:

- La ventana superior izquierda es la **ventana de fuentes**, que presenta la organización jerárquica de los elementos incluidos en el proyecto. Esta ventana tiene a su vez tres pestañas que permiten ver los módulos funcionales (**Sources**), las bibliotecas de módulos (**Libraries**) o varias instantáneas del proyecto (**Snapshots**).
- La ventana inferior izquierda es la **ventana de procesos**, que presenta las distintas operaciones que se pueden realizar sobre el objeto seleccionado en la ventana de fuentes.
- La ventana inferior es la **ventana de mensajes**, donde se muestran distintos mensajes del proceso que se está ejecutando.
- La ventana de la derecha es la **ventana para la edición** de los distintos ficheros fuentes. Es una interfaz de múltiples documentos (MDI) conocida como espacio de trabajo (workspace), y permite ver informes HTML, documentos ASCII (como código de descripción de hardware VHDL), esquemáticos, diagramas de estados, simulaciones, etc.

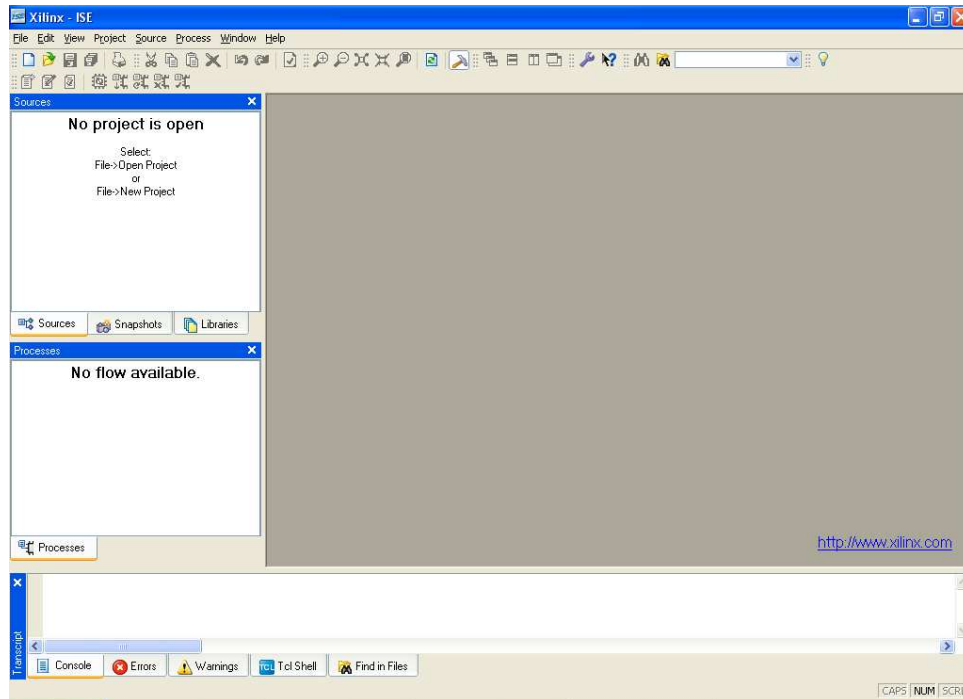


Figura 1.1: Ventana inicial de Xilinx ISE

Cada ventana puede ser redimensionada, despegada de su ubicación original, o movida a una nueva ubicación. Para restaurar la configuración de ventanas por defecto seleccionamos:

View→Restore Default Layout

1.1.2. Acceso a la ayuda

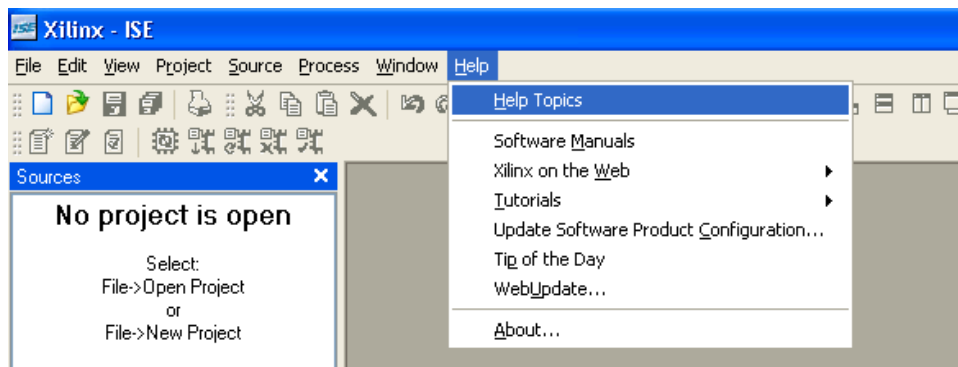


Figura 1.2: ISE Help Topics

En cualquier momento podemos acceder a la ayuda para consultar información adicional acerca del software ISE. El acceso a la ayuda se puede realizar de dos formas (véase Figura 1.2):

- Presionamos F1 para acceder a la documentación de una herramienta o función específica que hayamos seleccionado o resaltado.
- Lanzamos los contenidos generales de la ayuda desde el menú Help. Contiene información acerca de la creación y mantenimiento de nuestro flujo de diseño en ISE.

1.2. Creación de un nuevo proyecto

Para crear un nuevo proyecto pulsamos:

File→New Project

La creación de un nuevo proyecto necesita que introduzcamos 3 tipos de datos:

- Datos del proyecto.
- Datos del dispositivo donde se va a implementar
- Ficheros fuente del proyecto

En primer lugar, nos aparecerá una ventana como la de la Figura 1.3, donde debemos insertar los datos del proyecto.

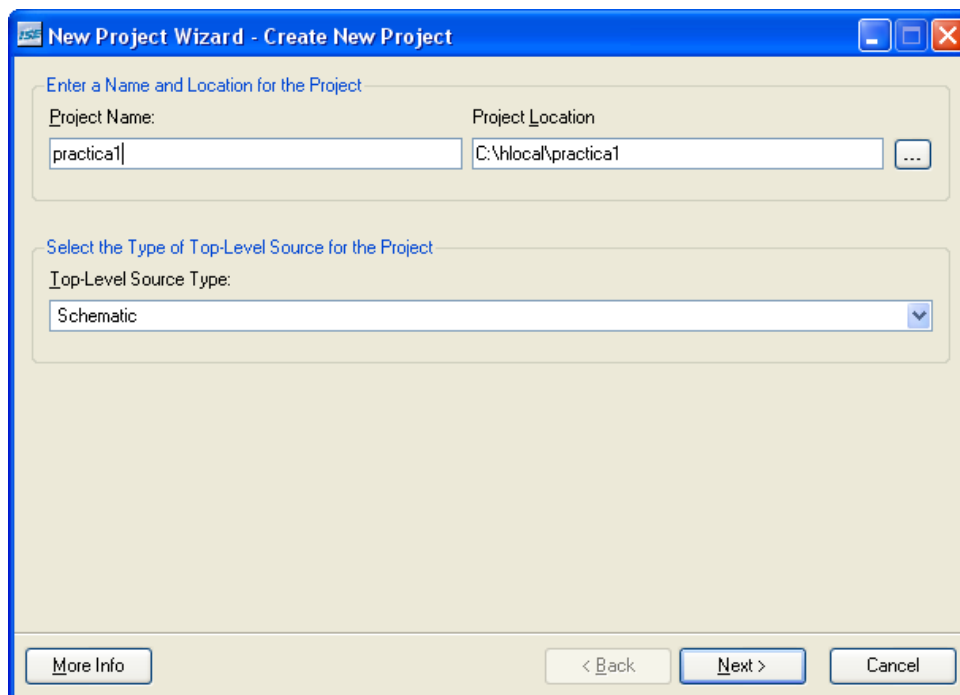


Figura 1.3: Datos del proyecto

1.2.1. Datos del proyecto

En **Project Name** damos el nombre de nuestro proyecto. Por ejemplo *practica1*. En **Project Location** damos el camino de datos donde se guarda el proyecto. Aconsejamos utilizar:

c:\hlocal\NombreProyecto

En nuestro ejemplo:

c:\hlocal\practica1

En **Top Level Source Type** de momento sólo utilizaremos el tipo **Schematic**, aunque podrían utilizarse otros métodos de descripción, como máquinas de estado o lenguajes de descripción hardware: VHDL y Verilog.

1.2.2. Datos del dispositivo

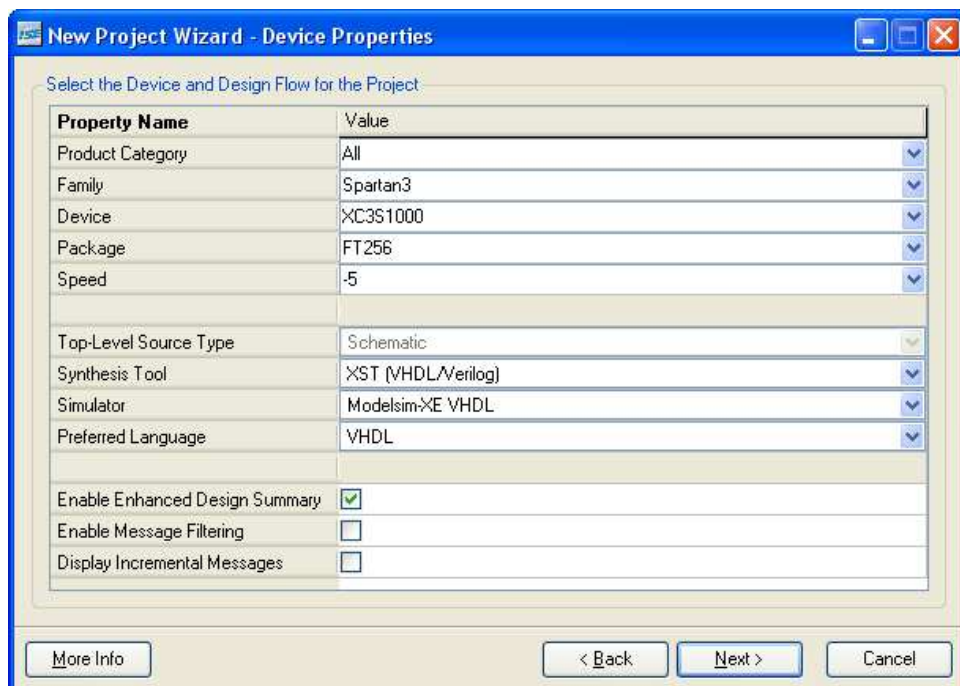


Figura 1.4: Datos del dispositivo y del flujo de diseño

Después pulsamos el botón de **Next**, lo cual da lugar a la aparición de otra ventana (como la de la Figura 1.4) que sirve para dar los datos del dispositivo sobre el que se va a realizar la implementación del diseño.

En este caso hay que introducir los datos de la familia de FPGAs con la que vamos a trabajar (Spartan 2, Spartan 3, Virtex, etc), los datos del dispositivo en particular y de su empaquetamiento. Todos estos datos pueden leerse sobre el circuito integrado correspondiente a la FPGA

que estamos usando, y para el caso en particular de las FPGAs del laboratorio son los que se muestran en la (Spartan3 XC3S1000 FT256). El resto de los datos sobre las herramientas de síntesis, simulador, etc, deben seleccionarse también como se muestra en la Figura 1.4.

1.2.3. Ficheros fuente del proyecto

Los ficheros fuentes del proyecto pueden ser de dos tipos:

- Ficheros nuevos, que vamos a crear en nuestro proyecto.
- Ficheros existentes, creados para otros proyectos y que vamos a reutilizar en nuestro proyecto.

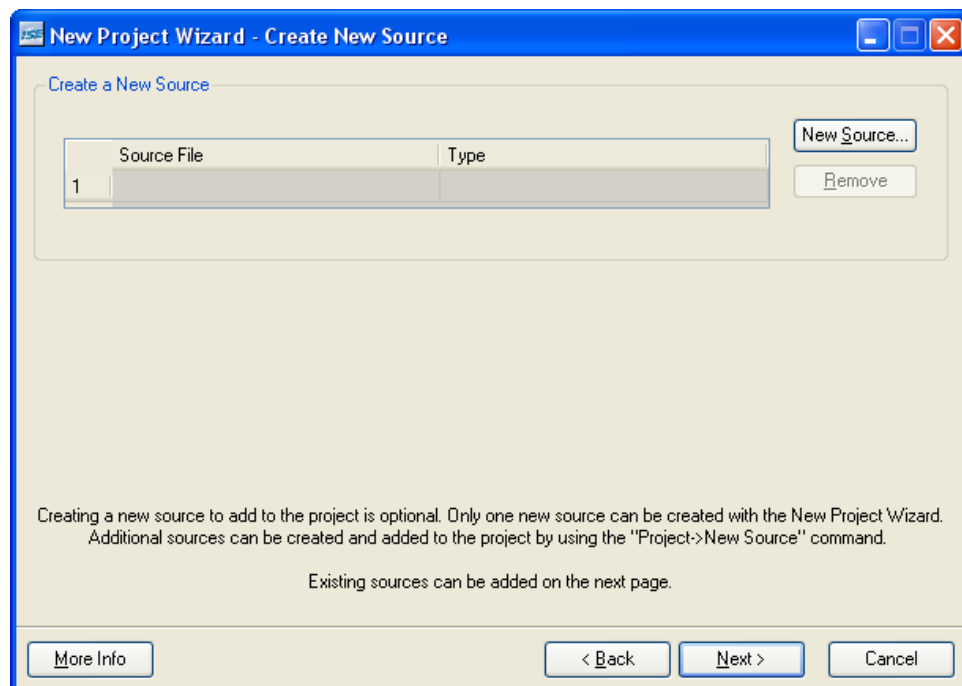


Figura 1.5: Datos de los nuevos ficheros fuente del proyecto

Volvemos a pulsar **Next** y nos aparece una ventana como la de la Figura 1.5. Esta ventana sirve para determinar todos los ficheros fuente que vamos a crear en nuestro proyecto. En las prácticas iniciales nuestro proyecto estará formado por un único diseño en formato esquemático, y por tanto sólo utilizaremos un fichero fuente. Pero en prácticas más complejas puede utilizarse diseño jerárquico, formado por varios módulos conectados entre sí, cada uno de ellos sobre un fichero fuente diferente, e incluso en un formato distinto.

Ficheros nuevos de nuestro proyecto

Pulsamos **New Source** y nos aparece la ventana de la Figura 1.6. Allí damos el nombre del fichero (por ejemplo *prueba1*), el camino de datos donde vamos a almacenar el diseño (se puede

dejar el que sale por defecto), y el tipo correspondiente, que en nuestro caso será **Schematic**.

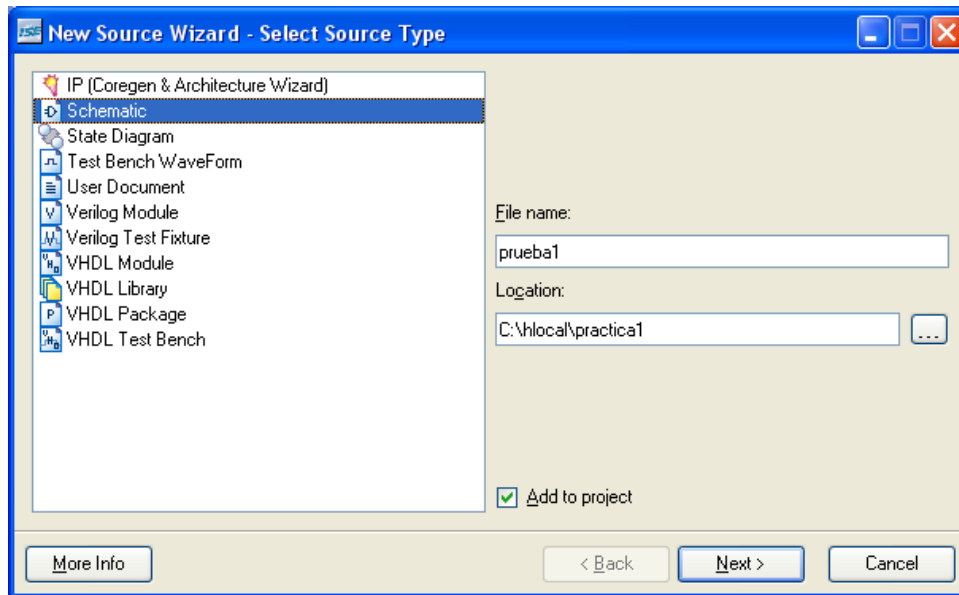


Figura 1.6: Datos de nuestro fichero fuente

Pulsamos **Next** y **Finish**. Si el directorio no existe nos pregunta si queremos crearlo. En este caso diremos **Yes**. A continuación nos aparece otra ventana, parecida a la de la Figura 1.5, pero con los datos de los ficheros fuente que hemos indicado que queremos crear, tal y como se muestra en la Figura 1.7.

Si nuestro proyecto va a constar de más de un módulo nuevo, repetiríamos este proceso tantas veces como sea necesario. Cuando hayamos terminado de dar los nombres de todos los ficheros que vamos a crear, pulsamos **Next**, y aparece una ventana como la de la Figura 1.8, que sirve para incluir en el proyecto otros ficheros fuente generados anteriormente. Si no vamos a incluir otros ficheros, pulsamos **Next** y **Finish**.

Ficheros existentes, que vamos a reutilizar en nuestro proyecto

El método es exactamente el mismo del proceso de crear nuevos ficheros fuente (apartado anterior), pero en este caso para añadir diseños existentes (de otros proyectos) tanto en formato de esquemático, como en otros lenguajes de descripción como VHDL. Por ejemplo, si queremos añadir un divisor de frecuencias como el que se proporciona en clase, pulsaremos **Add Source**.

Seleccionamos el fichero del divisor de frecuencias (divisor.vhd) y a continuación **Next**, **Finish** y **OK**. Con esto hemos terminado el proceso de inicialización.

1.3. Apertura de un proyecto existente

Simplemente seleccionamos:

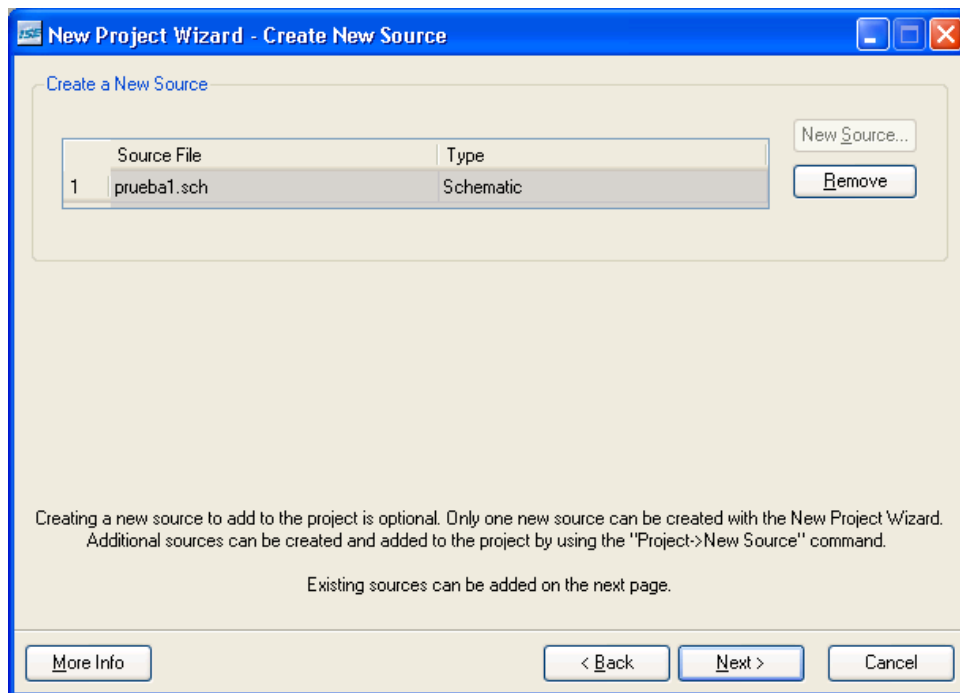


Figura 1.7: Datos de los ficheros fuente nuevos que queramos crear en nuestro proyecto

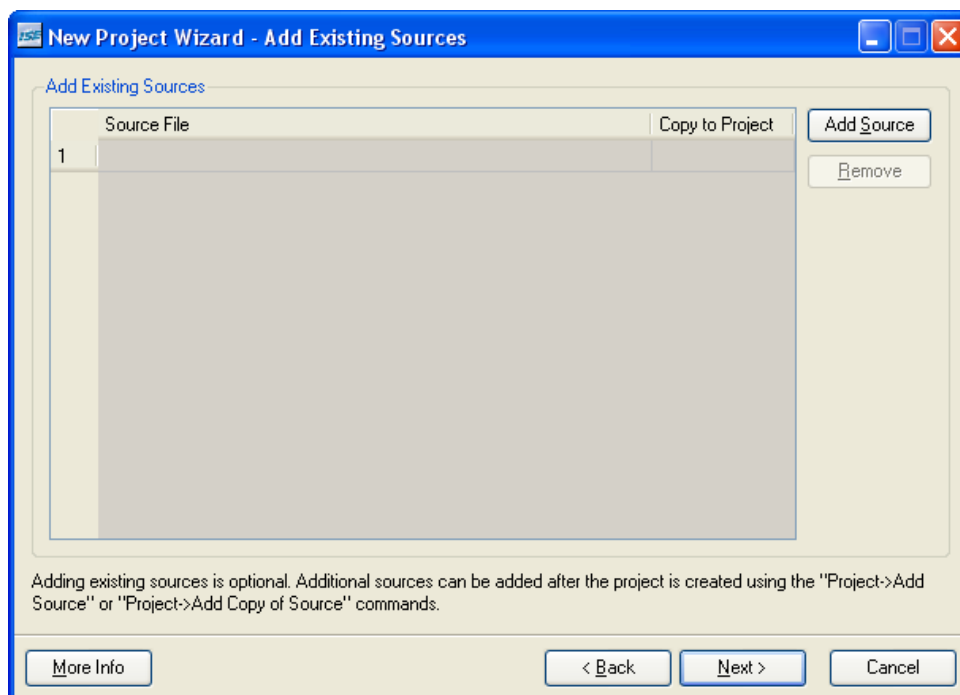


Figura 1.8: Datos de ficheros fuente ya creados que queramos añadir a nuestro proyecto

Archive→Open Project

y seleccionamos el proyecto que queremos abrir (con la opción **Tipo ISE Project Files**) en la ventana correspondiente (ver Figura 1.9).

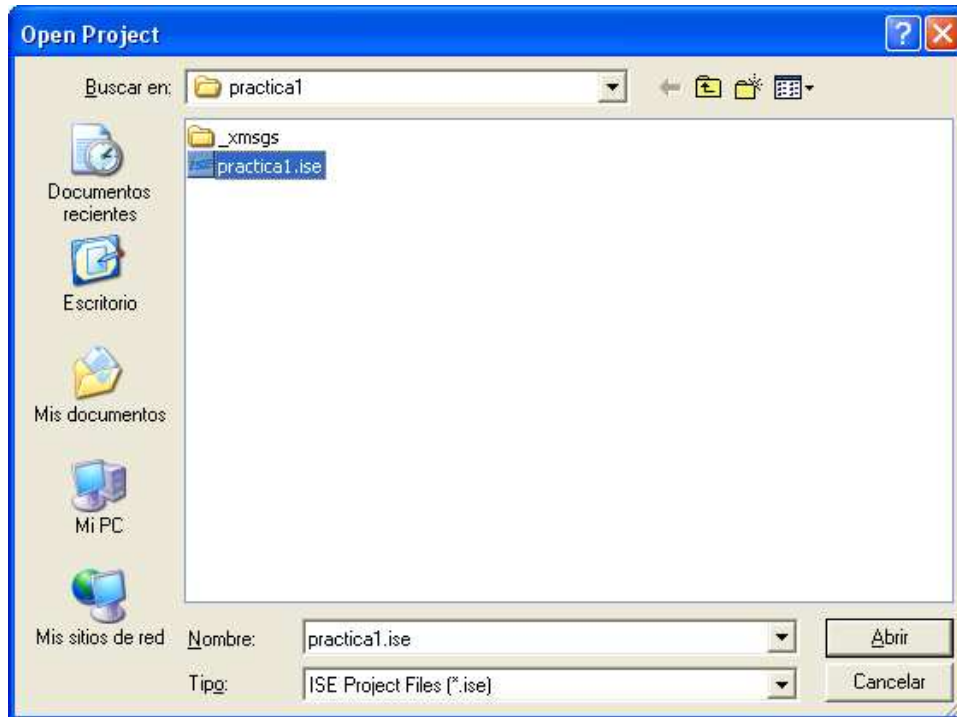


Figura 1.9: Apertura de un proyecto existente

1.4. Creación del esquemático

Una vez terminado el proceso de inicialización, o bien el de abrir un proyecto existente, debe aparecer una ventana como la que se muestra en la Figura 1.10. Observamos que en el workspace aparece la pestaña **Design Summary**, y que en la ventana **Sources** está seleccionada la pestaña **Sources**.

En la ventana de fuentes, con la pestaña **Sources** seleccionada, observamos la lista desplegable etiquetada **Sources for:**, que nos permite seleccionar fuentes para distintas etapas del proceso de diseño (para síntesis, para simulación lógica, simulación después de la colocación, etc.). Nos situaremos en la selección **Synthesis/Implementation**, y dentro del conjunto de módulos para síntesis, seleccionamos el esquemático que queremos editar (por ejemplo, *prueba1*), y con el botón derecho del ratón pulsamos **Open**. El espacio de trabajo (o ventana de edición) cambia entonces a un editor de esquemáticos que presenta la Figura 1.11.

En la ventana de fuentes nos aparece una nueva pestaña **Symbols**, y al seleccionarla se muestra la biblioteca de módulos prediseñados que podemos utilizar en nuestro diseño (Figura 1.12). En la ventana para edición dibujaremos nuestro diseño.

Para elegir los símbolos utilizamos la pestaña **Symbols** de la ventana de fuentes, y a través del menú **Categories** y el menú **Symbols** iremos seleccionando los elementos de nuestro diseño y

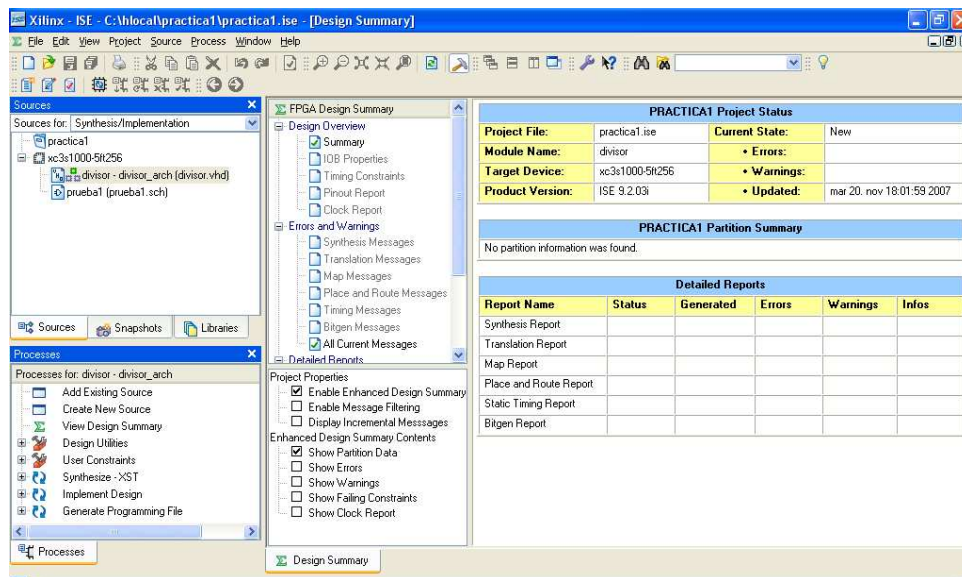


Figura 1.10: Entorno de Xilinx ISE

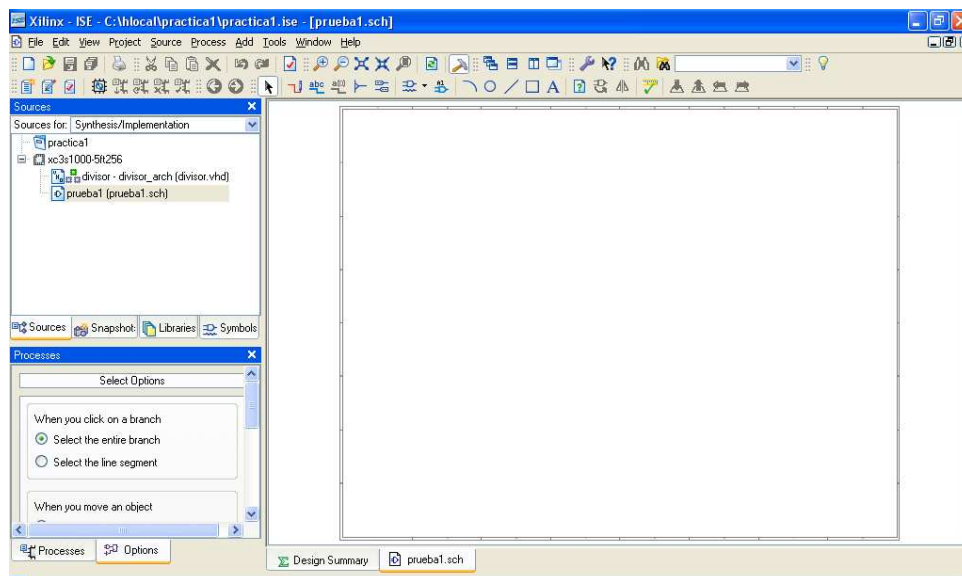



Figura 1.11: Entrada de esquemáticos

colocándolos en la ventana de edición. En el ejemplo vemos que estamos generando una and de 4 entradas a partir de puertas and de 2 entradas.

Para realizar las conexiones utilizaremos el botón AddWire  que se encuentra en la barra de herramientas.

Una vez realizadas las conexiones el diseño debe quedar como se muestra en la Figura 1.13.

En el caso que vayamos a implementar el diseño, nos falta añadir los IBUF y OBUF (categoría

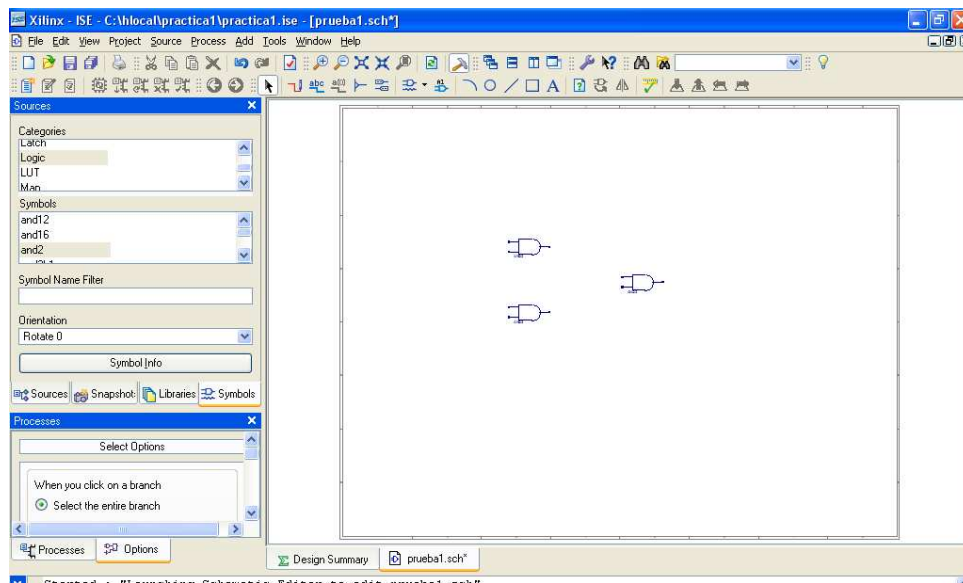


Figura 1.12: Entrada de esquemáticos con menú de símbolos

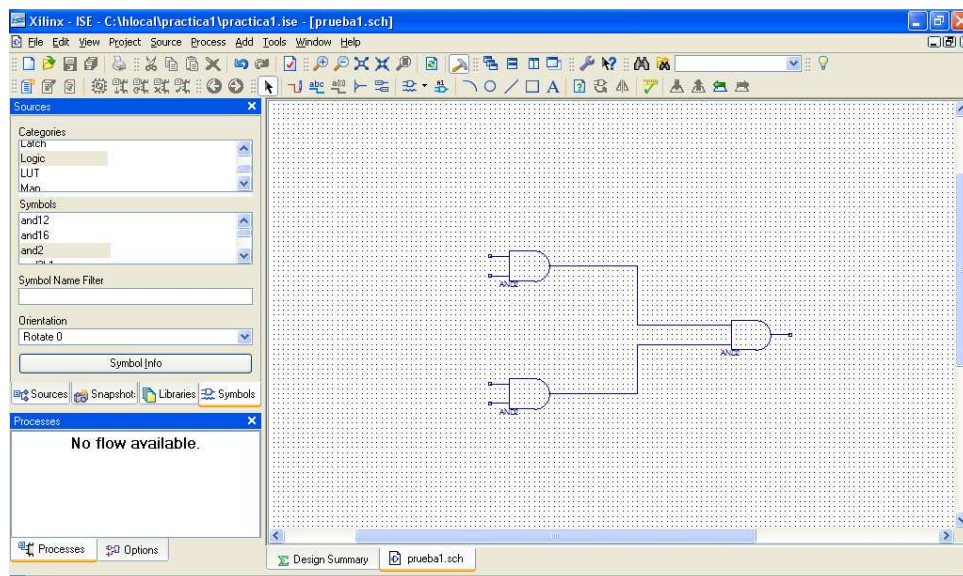



Figura 1.13: And de 4 entradas sin Buffers de E/S

IO, símbolos ibuf y obuf), y los pines de entrada salida con el botón  que se encuentra también en la barra de herramientas. El resultado es el que se muestra en la Figura 1.14. Una vez terminado es aconsejable salvar el diseño.

Verificamos que el esquemático no contiene errores lógicos seleccionando:

Tools→Check Schematic

La ventana de mensajes debería informar que no se detectan errores o advertencias. Si la ventana

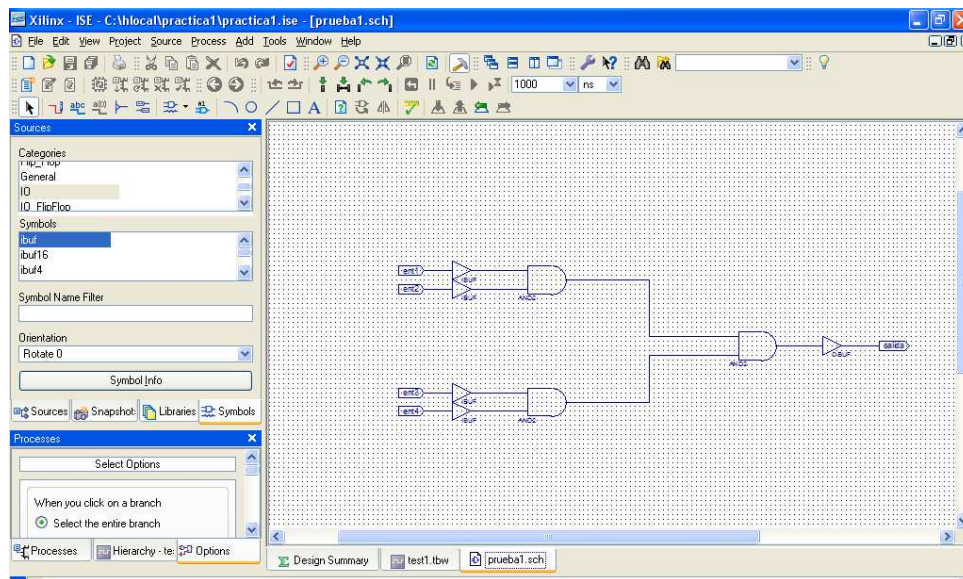


Figura 1.14: And de 4 entradas con Buffers de E/S

de mensajes nos informa de alguna advertencia o error, se debe remediar antes de continuar.

1.4.1. Buses

Cuando en un diseño utilizamos señales de más de un bit, es conveniente emplear buses. Lo primero que hay que hacer es crear los terminales de los buses. Para ello, al pinchar dos veces sobre los pines de entrada-salida, nos aparecerá una ventana como la que se muestra en la Figura 1.15 donde tendremos que indicar el nombre, el tipo de terminal y el número de bits del bus, según el formato:

NOMBRE(NUM-BITS)

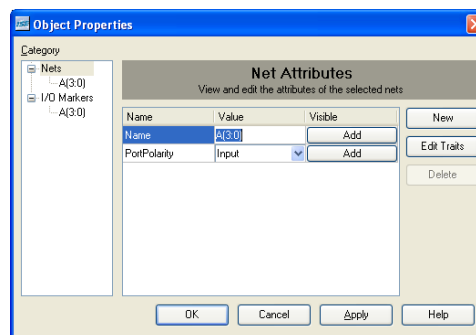


Figura 1.15: Pantalla de definición de propiedades del bus

Como último paso, hay que renombrar los cables de conexión de las entradas de las puertas correspondientes con los nombres de los componentes del bus que se desean. Para ello, pulsamos con el botón derecho sobre el cable que queramos renombrar y elegimos la opción de renombrarlo

(**Rename Selected Net**), de modo que nos aparece una ventana tal y como se muestra en la Figura 1.16.

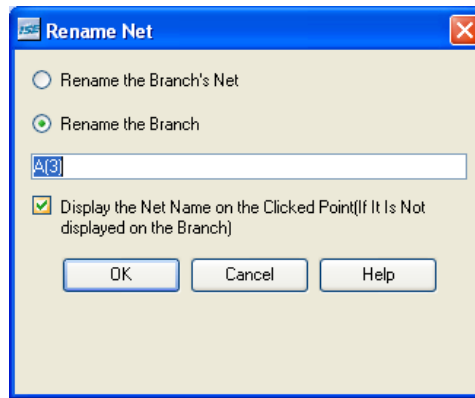


Figura 1.16: Pantalla para renombrar un cable de conexión

Después sólo hay que incluir el nombre del componente del bus que queramos utilizar como cable de conexión para dicha puerta según el formato:

NOMBRE(BIT)

o bien:

NOMBRE(BIT-INICIO:BIT-FIN)

en el caso de ser un rango de bits). Como ejemplo, el resultado final con un bus de entrada para un esquemático que contiene una puerta AND de 4 entradas quedaría como aparece en la Figura 1.17.

Otra forma de crear un bus es dibujando un cable aislado y renombrándolo como:

NOMBRE(BIT-INICIO:BIT-FIN)

, creando a continuación un pin de entrada/salida en uno de sus extremos. El resultado sería el mismo que el mostrado en la Figura 1.17.

También se pueden crear buses concatenando varias señales, separándolas por comas y sin espacios. En el ejemplo que aparece en la Figura 1.17 se ha formado un bus de 8 bits con el sub-bus de 6 bits **datos(7:2)**, y las señales *a* y *b*.

Si se quiere conectar 2 buses entre sí, se puede utilizar un buffer (BUF) instanciado tantas veces como el ancho del bus. Para instanciar *n* veces un componente se pincha sobre su nombre con el botón derecho y se selecciona **Object Properties**. En la ventana que aparece, en el atributo **InstName** se pone un nombre (como queramos llamarlo) y el número de veces que queremos instanciarlo. En el ejemplo de la Figura 1.17, hemos conectado 2 buses de 8 bits: el formado por la concatenación de **datos(7:2)**, con las señales *a* y *b* con el bus **datos2(7:0)**. Como queremos unir 2 buses de 8 bits, el componente BUF hay que instanciarlo 8 veces, por lo que en el atributo **InstName** hemos puesto **mibuffer(7:0)**.

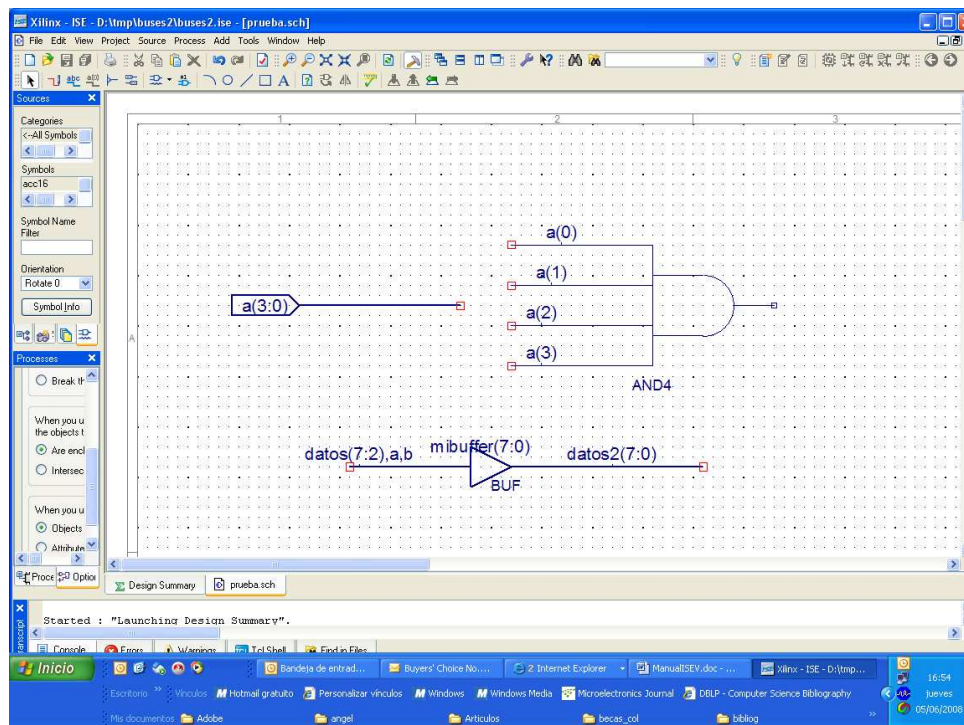


Figura 1.17: Uso de buses

1.4.2. Arrays de instancias

Podemos generar automáticamente un array de instancias. Al final del proceso sólo se muestra un símbolo en el esquemático (normalmente en negrita). El procedimiento es el que sigue:

1. Seleccionamos el símbolo del que deseamos crear múltiples instancias.
2. Seleccionamos:

Edit → Rename → Rename Selected Instance

3. En el cuadro de diálogo **Rename Instance**, seleccionamos **Iterated Instance Name**.
4. En el campo **Base Name** introducimos el nombre base o prefijo, por ejemplo: MUX.
5. En el campo **Starting Value** introducimos el número de la primera instancia, por ejemplo 8.
6. En el campo **Ending Value**, escribimos el número de la última instancia, por ejemplo 15.
7. Seleccionamos **Display Instance Name** para visualizar el nombre en el esquemático.
8. Pinchamos **OK**.

También se pueden crear arrays de instancias renombrando el símbolo por medio de una notación de bus (por ejemplo MUX(8:15)).

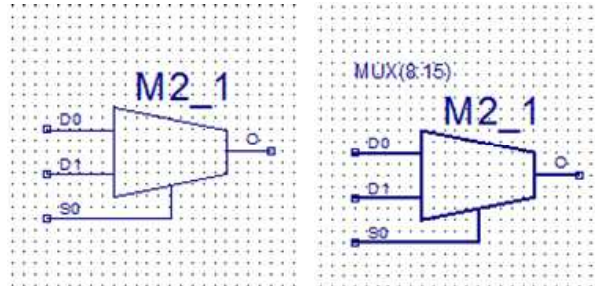


Figura 1.18: Símbolo antes y después de ser renombrado

El resultado es que hemos creado una instancia que contiene un array de símbolos, a pesar de que sólo se muestra uno de ellos en el esquemático. En el ejemplo de multiplexor mostrado en la Figura 1.18, MUX(8:15) representa un banco de ocho multiplexores 2 a 1 con los nombres MUX(8) a MUX(15), o lo que es lo mismo, un multiplexor 2 a 1 de 8 bits.

Finalmente, al conectar un bus con uno de los pines del array, la señal enésima del bus se conecta con el pin de la enésima instancia del array. El ancho del bus debe coincidir con el número de instancias. Por otro lado, si se conecta un escalar (cable simple) a uno de los pines del array, el escalar se replica a todas las instancias.

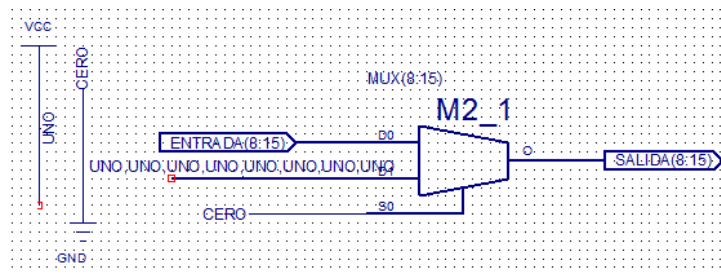


Figura 1.19: Ejemplos de conexión a múltiples instancias

1.5. Simulación del diseño

En primer lugar, en la ventana de fuentes seleccionamos la pestaña **Sources** y la opción **Behavioral Simulation** en la lista desplegable **Sources for:**, de forma que se visualicen las fuentes de nuestro proyecto que se pueden simular, y elegimos aquella que queremos simular (en nuestro caso *prueba1.sch*). Antes de realizar la simulación, es necesario generar un **testbench** con las entradas con que queremos estimular a nuestro diseño para comprobar que funciona. Es decir, la simulación sólo va a realizarse para el conjunto de estímulos que coloquemos en el testbench. Para circuitos simples, el conjunto de estímulos puede ser el total de todas las posibles entradas de un diseño, pero para módulos muy complejos puede interesarnos realizar la simulación sólo para un conjunto de posibles entradas.

1.5.1. Generación de un fichero de estímulos (Testbench)

Para generar el conjunto de estímulos, seleccionamos en el menú superior:

Project→New Source

y en la ventana que nos aparece (1.20) seleccionamos:

Tipo de fuente: Test Bench Waveform

y damos un nombre al fichero donde vamos a crearlo, por ejemplo test1, como se muestra en la Figura 1.20.

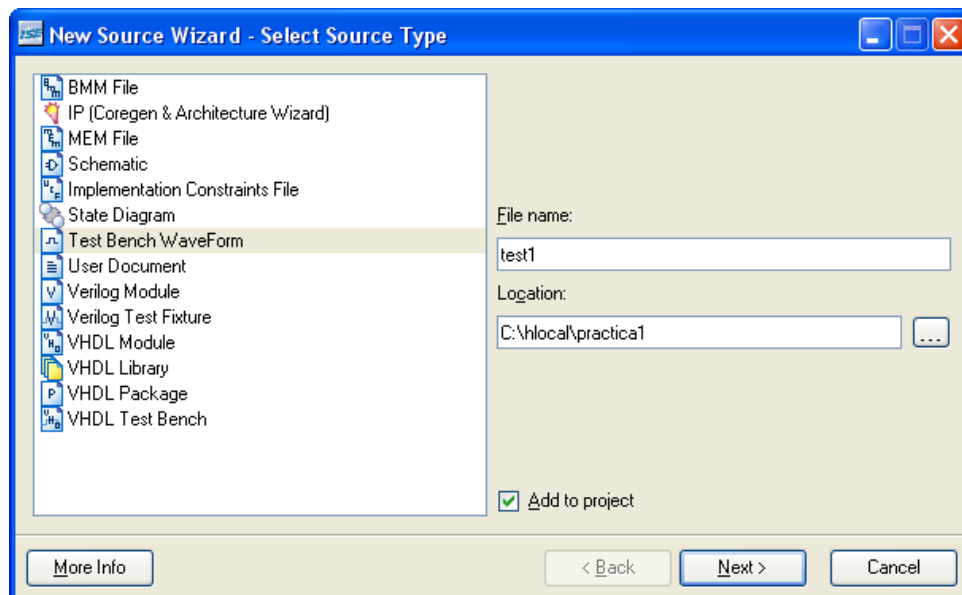


Figura 1.20: Generación de un test bench

Pulsamos **Next** y en la siguiente ventana (Figura 1.21) nos piden el nombre de la fuente con la cual queremos asociar el testbench. En nuestro ejemplo seleccionaremos *prueba1*.

Pulsamos **Next** y **Finish**. A continuación nos aparece una ventana con datos temporales para la simulación del diseño (Figura 1.22), como por ejemplo la forma del reloj, el tiempo de simulación, etc.

En este ejemplo, como se trata de un circuito combinacional, seleccionaremos la opción **Combinatorial** y dejaremos el resto de opciones con su valor por defecto. Para continuar pulsaremos **Finish**.

En la siguiente ventana, que se muestra en la Figura 1.23, nos aparece el valor que se ha asignado de forma automática, de todas las entradas y salidas de nuestro circuito.

Es necesario modificar el conjunto de estímulos que aparece inicialmente, de la forma que consideremos más adecuada, para probar el máximo número de combinaciones de entrada posibles.

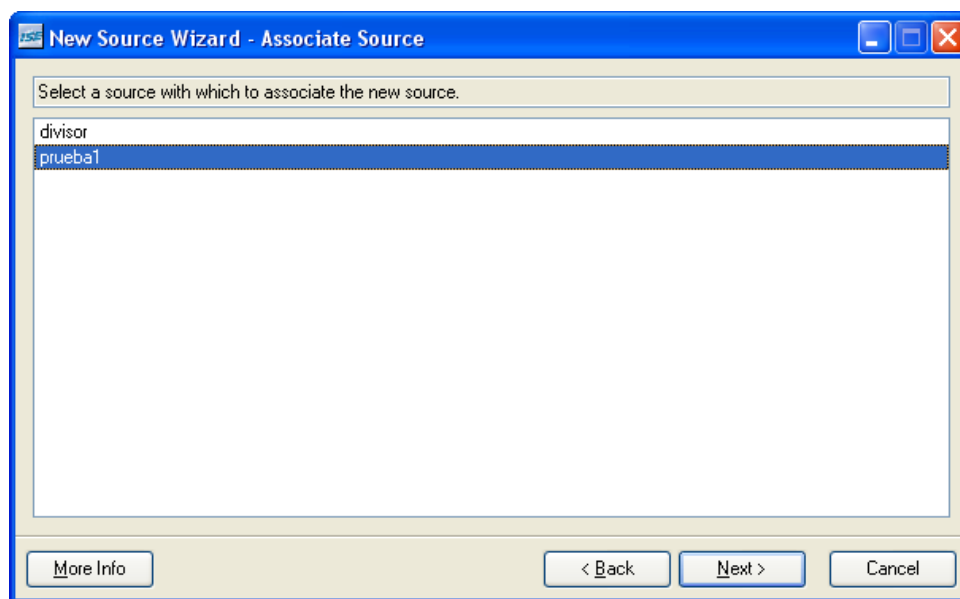


Figura 1.21: Fuente asociada al Testbench

Simplemente pulsando en las correspondientes formas de onda, donde queramos cambiar su valor, se modifican automáticamente. Un posible Testbench final es el que se muestra en la Figura 1.24. Una vez terminado, salvar las formas de onda.

Se puede observar que en la ventana de fuentes, con la selección **Behavioral Simulation**, ha aparecido la nueva fuente **test1.tbw**, en cuyo interior se encuentra nuestro esquemático *prueba1.sch*. El módulo *test1.tbw* es realmente el que podemos simular, puesto que tiene definidas las entradas. Si seleccionamos este módulo, y en la ventana de procesos seleccionamos la pestaña **Process**, veremos que entre los procesos asociados a la fuente *test1.tbw* aparece el de simulación.

1.5.2. Simulación de estímulos complejos de señales

Una manera alternativa de definir los estímulos de una cierta señal es pulsando con el botón derecho sobre su forma de onda y seleccionar entonces la opción de **Set Value**, lo que produce la aparición del cuadro de diálogo que se muestra en la Figura 1.25.

Donde se puede seleccionar la opción **Pattern Wizard**, que permite definir cómo varían las señales de entrada o tipo de patrón de estímulos (**Pattern Type**), la duración del estímulo en ciclos (**Number of cycles**), y con qué parámetros se debe aplicar el patrón (**Pattern Parameters**), según se muestra en la Figura 1.26.

De las opciones que aparecen en la ventana, sólo utilizaremos inicialmente el patrón de tipo Pulsos (**Pulse**). Este patrón nos permite darle un valor inicial a cada señal (**Initial Value**), el retardo que dicho valor inicial se debe mantener (**Initial Delay**), e indicar cuál será el valor del pulso (**Pulse Value**) y el número de ciclos de reloj que quiere mantenerse la señal a dicho valor (**Pulse Width**). De este modo, para crear un estímulo que dure 8 ciclos desde la posición actual de simulación, para una señal que cambie su valor cada ciclo de reloj y tenga como valor inicial un 0 lógico, debemos definir la señal como se muestra en la Figura 1.26. De manera similar, si

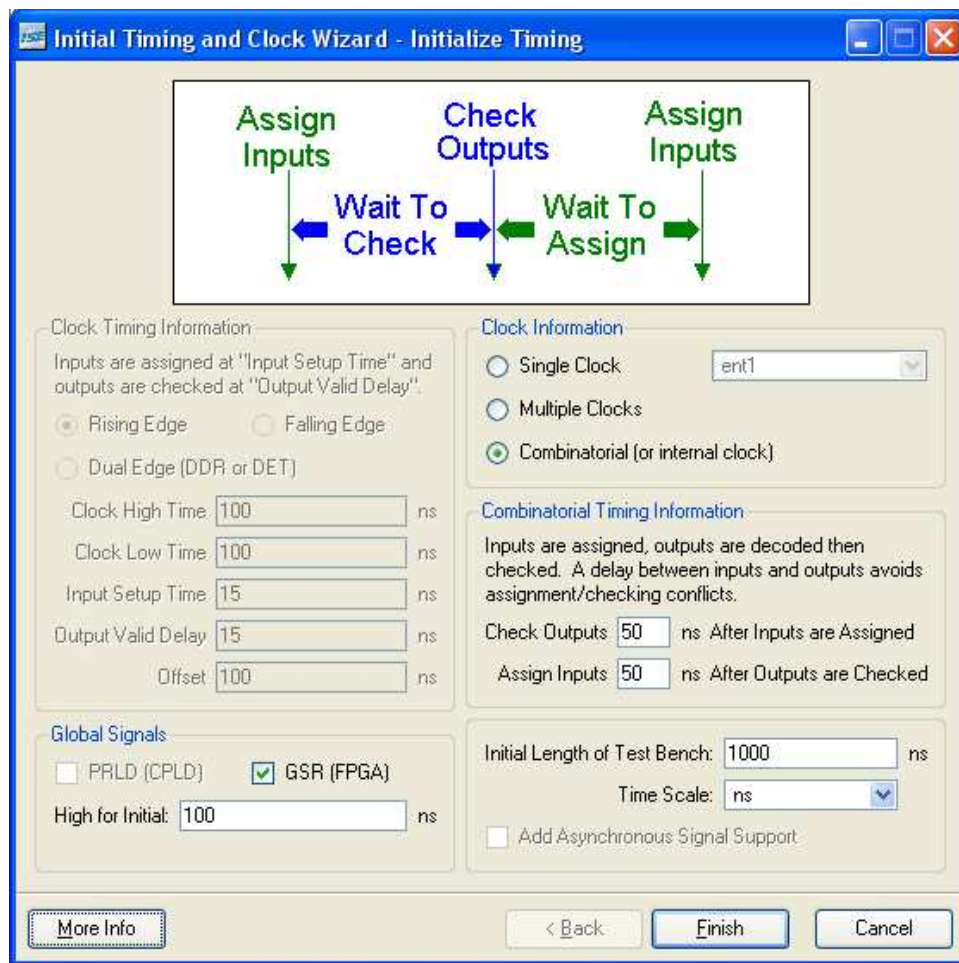


Figura 1.22: Datos temporales de la simulación

se quisiera generar una señal que cambie su valor lógico, basta con fijar su retardo inicial y su ancho de pulso a 2, y así sucesivamente con potencias de dos para las señales con mayor peso.

1.5.3. Simulación de un Testbench

En la ventana de fuentes, seleccionamos la pestaña **Sources**, y dentro de las fuentes la correspondiente al Testbench (*test1.tbw*). En la ventana de procesos seleccionamos la pestaña **Process**, para que nos aparezcan todos los procesos asociados a la fuente *test1* (según se muestra en la Figura 1.27). Entonces mandamos ejecutar la simulación pulsando dos veces sobre:

ModelSim Simulator→**Simulate Behavioral Model**

, o bien seleccionado:

ModelSim Simulator→**Simulate Behavioral Model**

y pulsando el botón derecho y seleccionando **Run**.

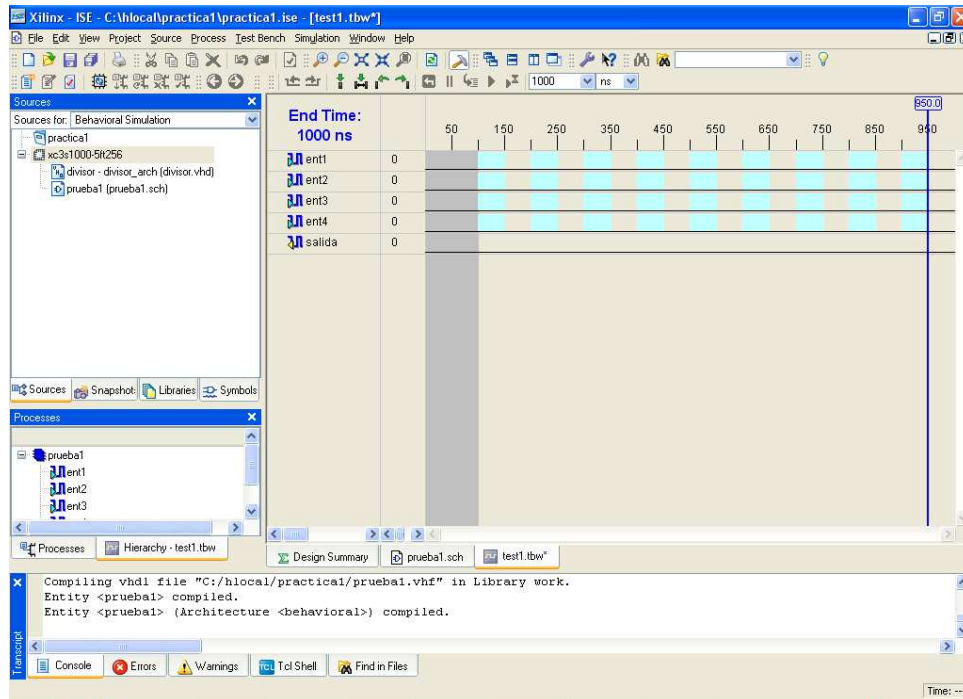


Figura 1.23: Formas de onda iniciales del TestBench

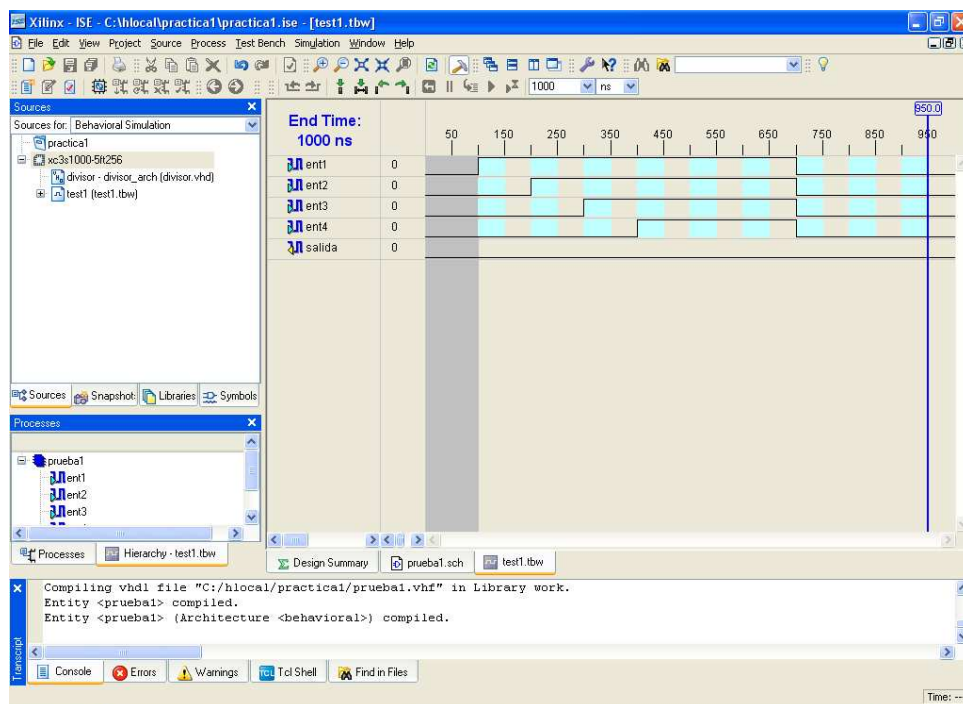


Figura 1.24: Formas de onda finales del TestBench

El resultado de la simulación se presenta en la Figura 1.28. En algunas ocasiones hay que cerrar una ventana que aparece en el workspace, justo antes de la visualización de la simulación. Es

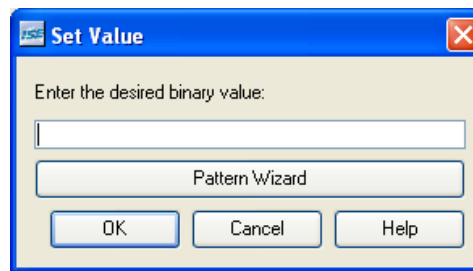


Figura 1.25: Diálogo para fijar valores de simulación para variables

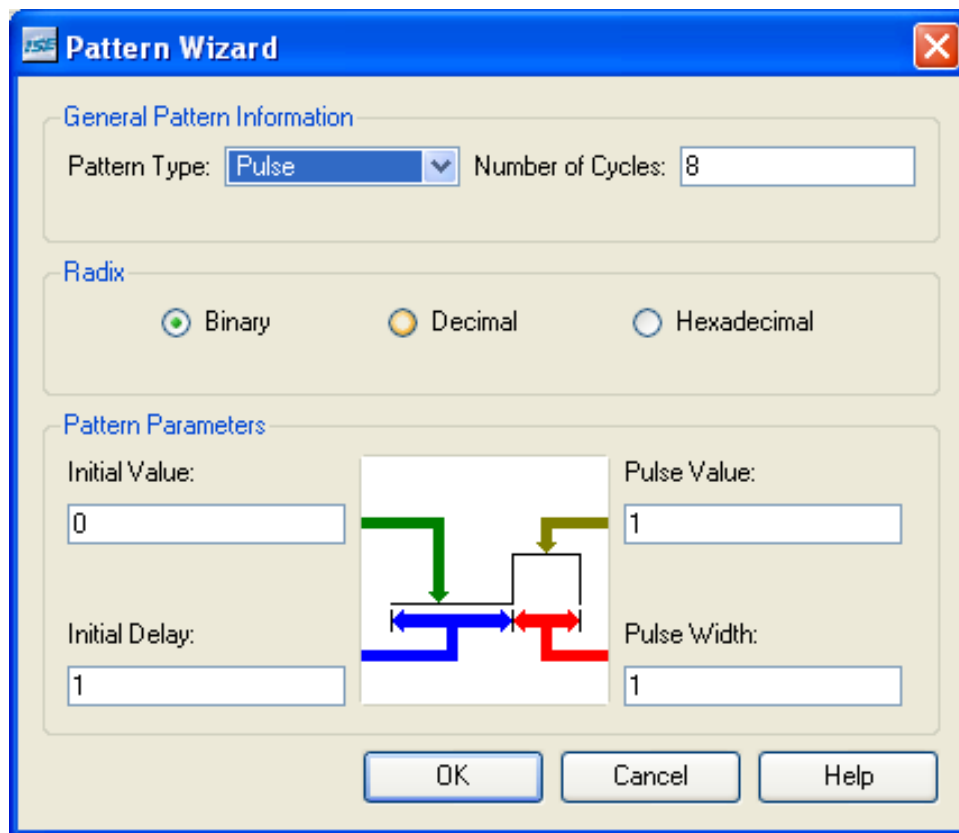


Figura 1.26: Formas de onda finales del TestBench

necesario centrar la ventana de la simulación y ajustar el zoom para observar correctamente el resultado.

Si consideramos que es mejor cambiar el valor de los estímulos de entrada, habrá que repetir el proceso de edición del testbench. Si no está abierto el fichero con las formas de ondas, primero pinchamos sobre la fuente correspondiente (*test1.tbw*) y con el botón derecho del ratón pulsar **Open**. Una vez abierto, se modifican las formas de onda pinchando sobre ellas como se ha explicado en el apartado 1.5.1 y luego se vuelve a ejecutar la simulación seleccionando, en la ventana de procesos, el simulador que se desee.

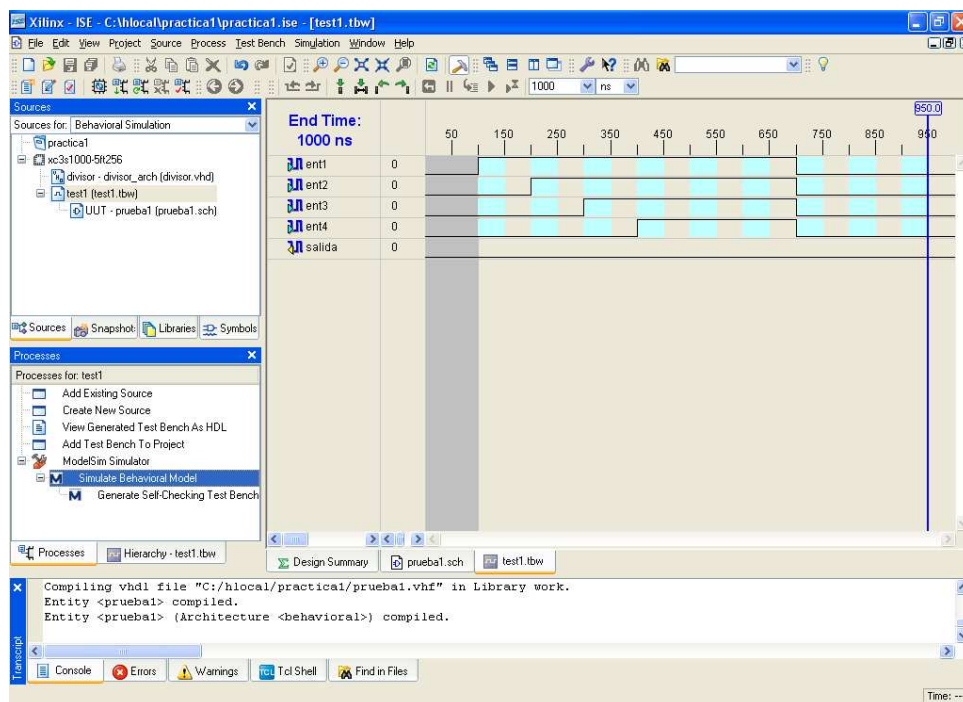


Figura 1.27: Inicio de la simulación utilizando el ModelSim

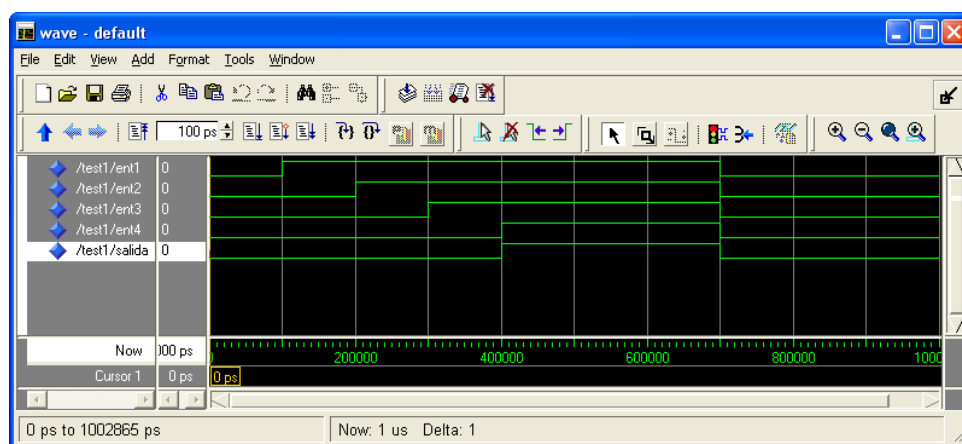


Figura 1.28: Resultado de la Simulación del TestBench

1.6. Generación de un símbolo a partir de un fichero VHDL

Si queremos realizar un diseño jerárquico, y utilizar un módulo dentro de otro, primero debemos generar los símbolos correspondientes. En nuestro ejemplo vamos a generar un símbolo del divisor de frecuencias (*divisor.vhd*) y a incluirlo en nuestro esquemático *prueba1.sch*.

En primer lugar vamos a la ventana de fuentes (seleccionar la vista **Synthesis/Implementation**) y seleccionamos el fichero fuente del divisor de frecuencias (*divisor.vhd*). En la ventana de procesos seleccionamos la pestaña **Process**, y dentro de Design Utilities pulsamos dos veces sobre

Create Schematic Symbol, según se muestra en la Figura 1.29.

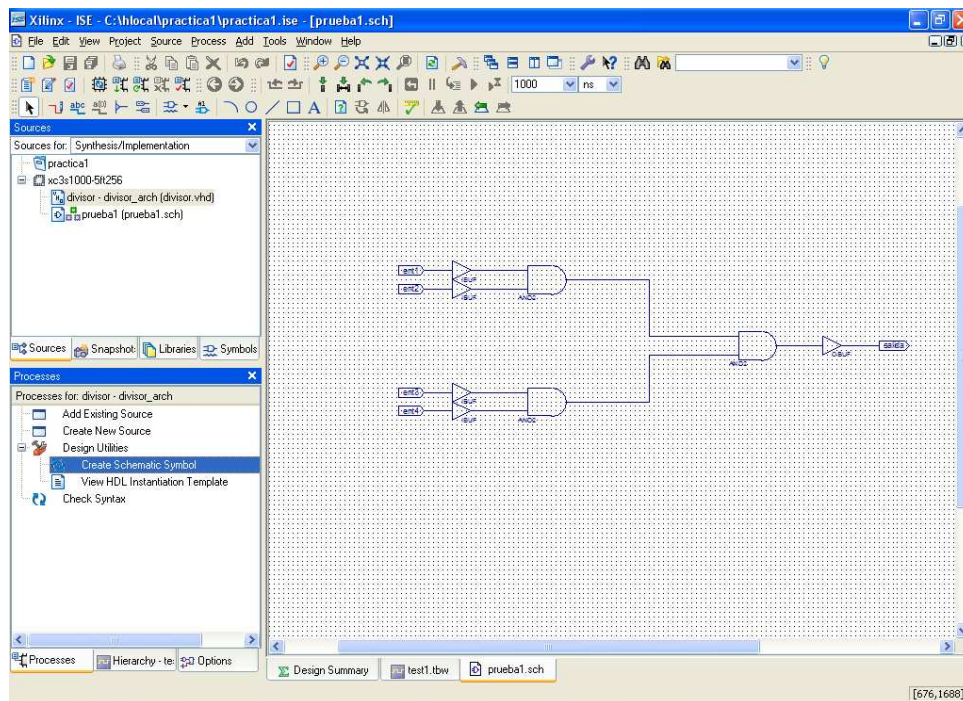


Figura 1.29: Generación de un símbolo asociado a un módulo

Si ahora vamos a la ventana de fuentes y seleccionamos la pestaña **Symbol**, vemos que en la categoría `c:/hlocal/practical1` aparece el símbolo del divisor. Podemos añadirlo a nuestro esquemático `prueba1.sch` tal y como se muestra en la Figura 1.30.

Si ahora seleccionamos la ventana de fuentes y la pestaña **Sources**, veremos que el divisor de frecuencias está dentro del módulo `prueba1`, puesto que se trata de un diseño jerárquico (Figura 1.31).

1.7. Generación de un fichero de restricciones *.ucf

Asociado a nuestro diseño debemos generar un fichero de restricciones *.ucf, que sirve para indicar en qué patillas se van a colocar todas las entradas y salidas de nuestro circuito. Por ejemplo, podemos querer que la entrada de reset quede conectada a un switch determinado de la placa, el reloj a la salida del oscilador, y la salida del circuito a uno de los leds. En el fichero `modelo3.0.ucf` de la página web de la asignatura tenemos los puertos asociados a todas las posibles entradas y salidas de la tarjeta que vamos a utilizar en el laboratorio. Por ejemplo, podemos ver que en dicho fichero, en el bloque marcado como **switches** la placa extendida, aparece una entrada como la siguiente:

```
#NET DIPSW<1> LOC=P12;
#NET DIPSW<2> LOC=J1;
#NET DIPSW<3> LOC=H1;
```

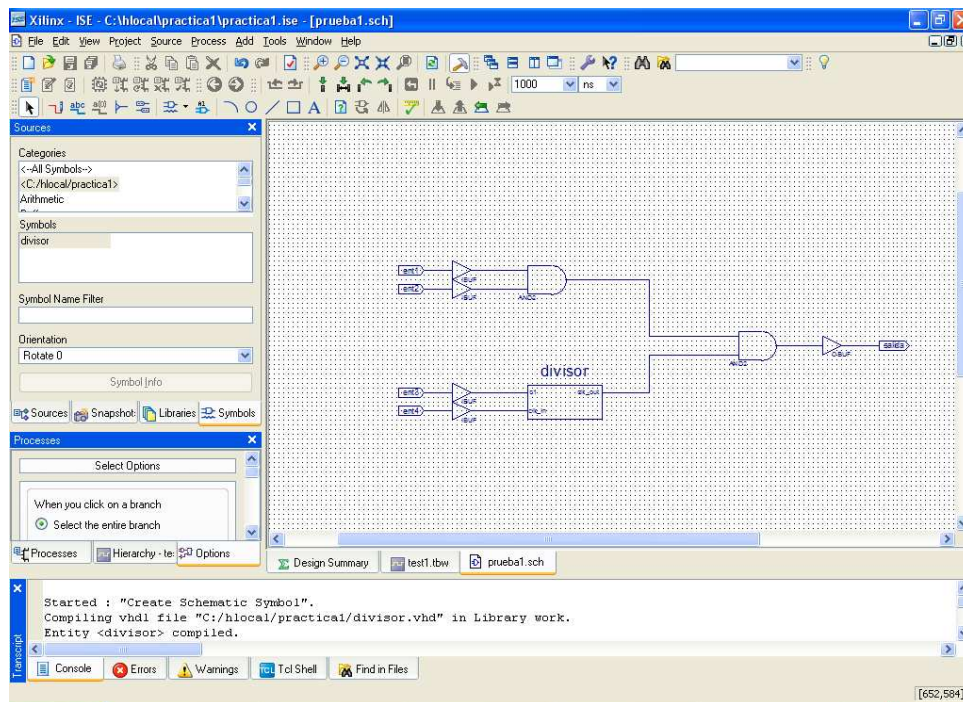


Figura 1.30: Esquemático con el divisor de frecuencias

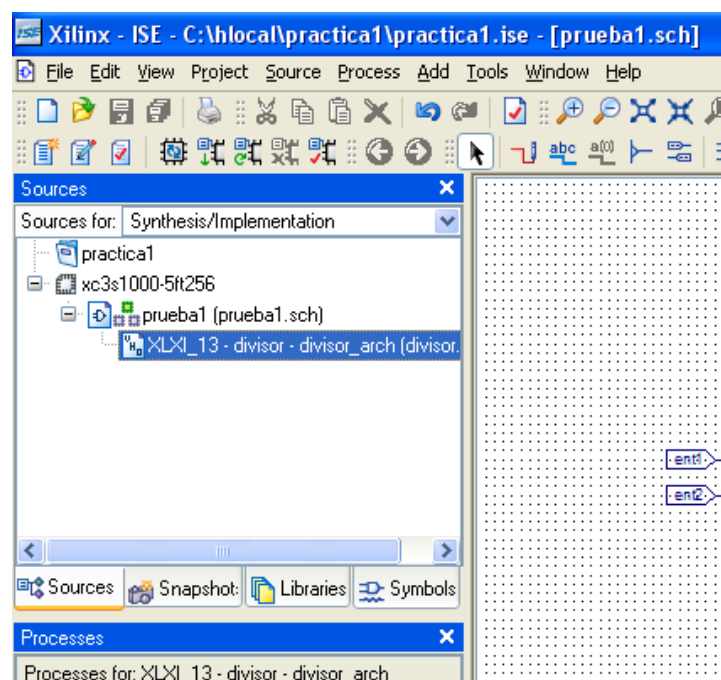


Figura 1.31: Diseño jerárquico

• • •

Eso quiere decir que el switch 1 está conectado a la entrada P12 de la FPGA, el switch2 a la entrada J1, el switch3 a la H1, etc.

Si queremos utilizar el **switch 1** de la tarjeta extendida para controlar la entrada ent1, tenemos que copiar esta línea en el fichero *.ucf de nuestro diseño, descomentarla y cambiar el nombre **DIPSW<1>** por el de nuestra señal **ent1**, de forma que la línea correspondiente a esta restricción quedará:

```
NET ent1 LOC=P12;
```

Y repetir este proceso para todas las entradas y salidas de nuestro diseño. Hay que tener en cuenta que si se nos olvida alguna entrada o salida, y no podemos la restricción correspondiente, el software no va a dar ningún error, y va a conectar esa entrada/salida a cualquier pin de la FPGA, que puede que no esté conectado a ningún switch ó led, con lo cual la depuración no será posible.

Para generar un fichero de restricciones *.ucf e incluirlo en el diseño se procede de la siguiente forma, que se muestra en la Figura 1.32.

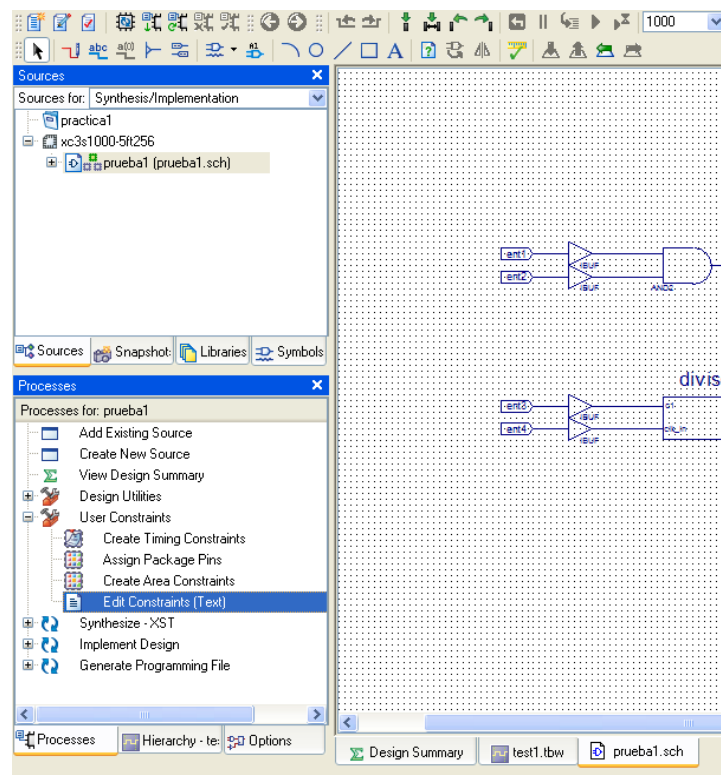


Figura 1.32: Generación del fichero de restricciones

Nos colocamos en la ventana de fuentes y, con la opción Sources for Síntesis/Implementation seleccionamos el fichero correspondiente al módulo de mayor jerarquía. En nuestro caso prueba1.sch. A continuación vamos a la ventana de procesos y ejecutamos:

User Constraints(Edit constraints (Text))

A la pregunta de si queremos crear un nuevo fichero diremos **Yes**. Y en la ventana de edición nos aparecerá el fichero vacío. En dicha ventana meteremos las restricciones correspondientes a nuestro circuito, según se muestra en la Figura 1.33.

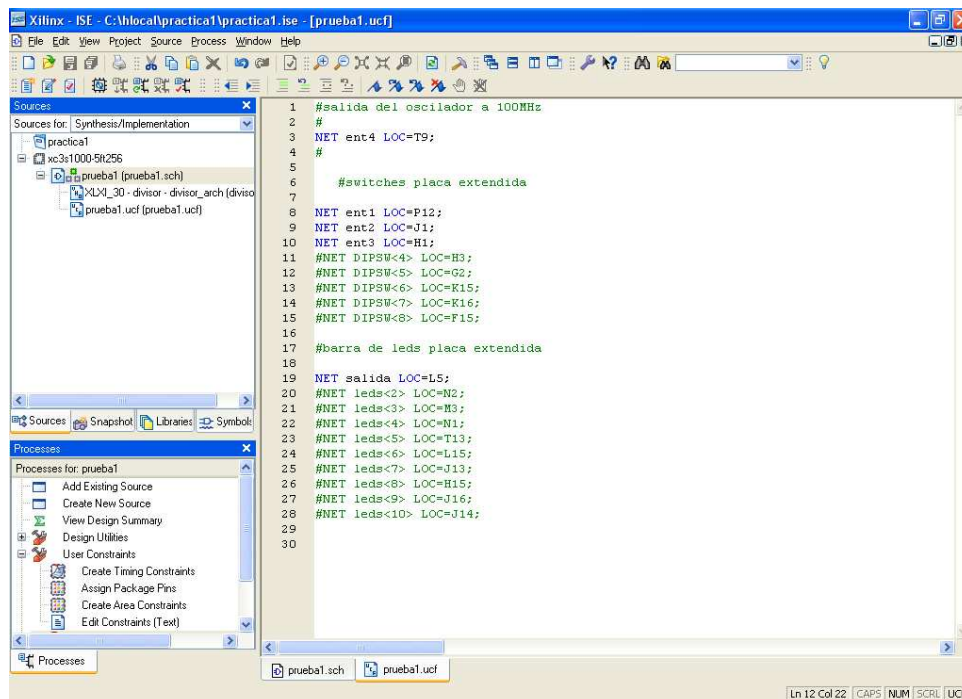


Figura 1.33: Fichero de restricciones

Podemos observar que las salidas que aparecen sin comentar son las siguientes:

```

#salida del oscilador a 100MHz conectada a ent4
#
NET ent4 LOC=T9;
#
# switches placa extendida.
# Los tres primeros se conectan a ent1, ent2 y ent3

```

```

NET ent1 LOC=P12;
NET ent2 LOC=J1;
NET ent3 LOC=H1;

```

#barra de leds placa extendida. El primer led se conecta a la salida

```

NET salida LOC=L5;

```

Una vez terminada la edición hay que acordarse de salvar el fichero.

También es posible generar el fichero de restricciones fuera del entorno de ISE, por ejemplo editando directamente el fichero modelo que hay en la página web de la asignatura, y modificando

las líneas correspondientes a las entradas y salidas que vamos a utilizar. En este caso los pasos a seguir son los siguientes:

1. Se genera el fichero *.ucf con cualquier editor, pero con cuidado de que la extensión sea ucf. Supongamos que lo hemos llamado como el fichero original, **modelo3.0.ucf** y lo tenemos en el directorio de nuestro circuito:

c:\hlocal\practical

2. En el menú superior pulsamos:

Project→Add Source

3. Seleccionamos el fichero correspondiente a las restricciones, que en nuestro ejemplo es modelo3.0.ucf y pulsamos **Abrir** y **OK**. En la ventana de fuentes aparece el nuevo fichero modelo3.0.ucf.
4. Si por cualquier motivo deseamos modificar el fichero de restricciones, hay que seleccionarlo en la ventana de fuentes y en la ventana de procesos pulsar:

User Constraints→Edit constraints (Text)

5. En la ventana de edición aparecerá el fichero correspondiente. Hacemos las modificaciones que queramos y luego lo salvamos.

1.8. Implementación y Generación del Mapa de bits.

Para generar el mapa de bits que se cargará en la memoria de configuración de la FPGA, seleccionamos en la ventana de fuentes el fichero correspondiente al módulo de mayor jerarquía. En nuestro caso *prueba1.sch*. A continuación vamos a la ventana de procesos y seleccionamos:

Generate Programming File

y con el botón derecho del ratón pulsamos **Run**, tal y como se muestra en la Figura 1.34. Durante un cierto periodo de tiempo la herramienta realizará las fases de síntesis e implementación, y dentro de ésta última todos los pasos se realizan automáticamente:

1. Translate: se compila la netlist a un formato propio.
2. Map: se realizan optimizaciones para minimizar el número de puertas.
3. Place&Route: se asignan las puertas a CLBs y las conexiones a líneas de interconexión concretas y por último generará un mapa de bits (archivo *.bit) que se coloca en el directorio del proyecto (en nuestro caso prueba1.bit).

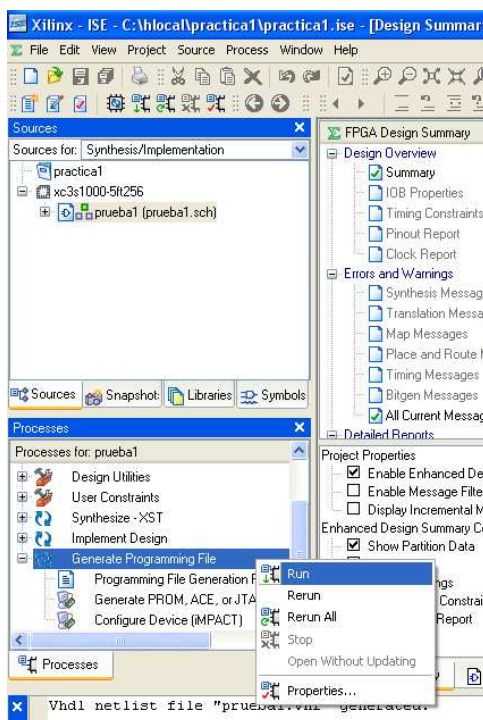


Figura 1.34: Generación del mapa de bits

1.9. Descarga del diseño en la placa XSA-3S1000

Desde el menú de Windows seleccionamos:

Programas→Electrónica→XStools→GXSLoad

Después seleccionamos el tipo de tarjeta: XSA-3S1000. Arrastramos el *.bit a la ventana FPGA y pulsamos **Load**.

Una vez volcado el mapa de bits, colocamos las entradas ent1, ent2 a 1 y ent3 a 0 (switches 1, 2 y 3 de la placa extendida) y comprobamos que un parpadeo aparece representado el led 1.

En las placas XSA-3S1000 debe realizarse un paso adicional para que el puerto paralelo no fuerce la salida:

Programas→Electrónica→XStools→GXSPort

Ponemos todos los datos a 1 y pulsamos SET.

Capítulo 2

¿Cómo ... ?

2.1. Cómo instalar Xilinx 10.1 en MS Windows 7 64 bits

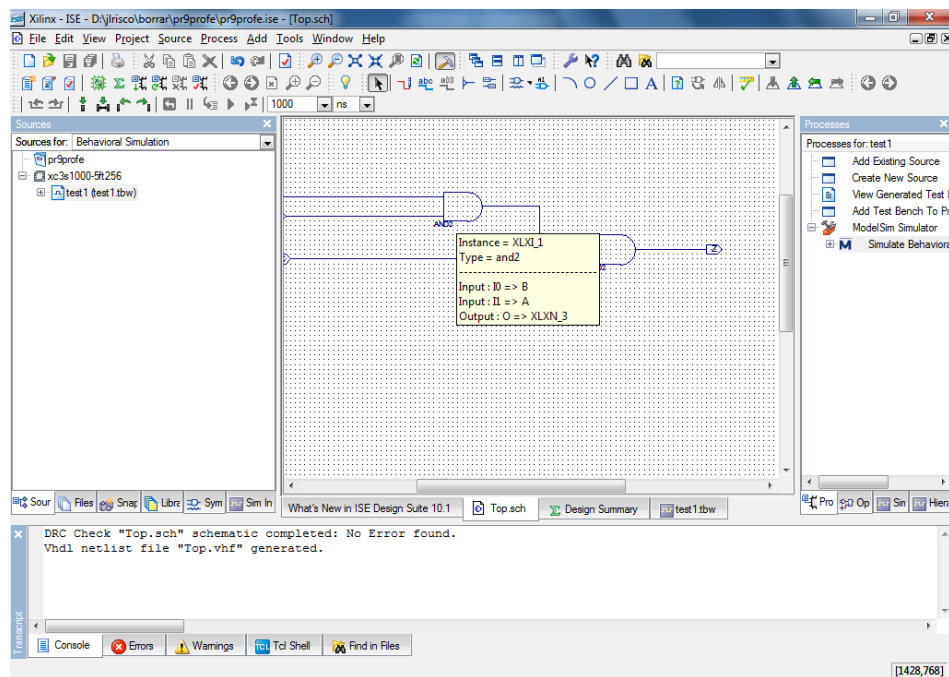
En Windows 7 64 bits no es posible usar el ISI Simulator, pero nos podemos descargar una versión gratuita (y tan limitada como el ISI) de ModelSim. Los pasos son los siguientes:

1. Descargar los archivos de instalación de Xilinx (no el instalador web)
2. Ejecutar el instalador de Xilinx con permisos de administrador. Su ruta es “Carpeta con archivos de instalación/bin/nt/setup.exe”
3. Descargar las actualizaciones del ISE
4. Descargar ModelSim XE (Xilinx Edition) desde la web de Xilinx. Tendréis que usar vuestra Xilinx account, creada para bajar el ISE
5. Instalar ModelSim XE. Seleccionad en el instalador ModelSim XE Starter y Full VHDL
6. Solicitar una licencia para ModelSim XE usando el asistente “Licensing Wizard” que se instala con ModelSim.

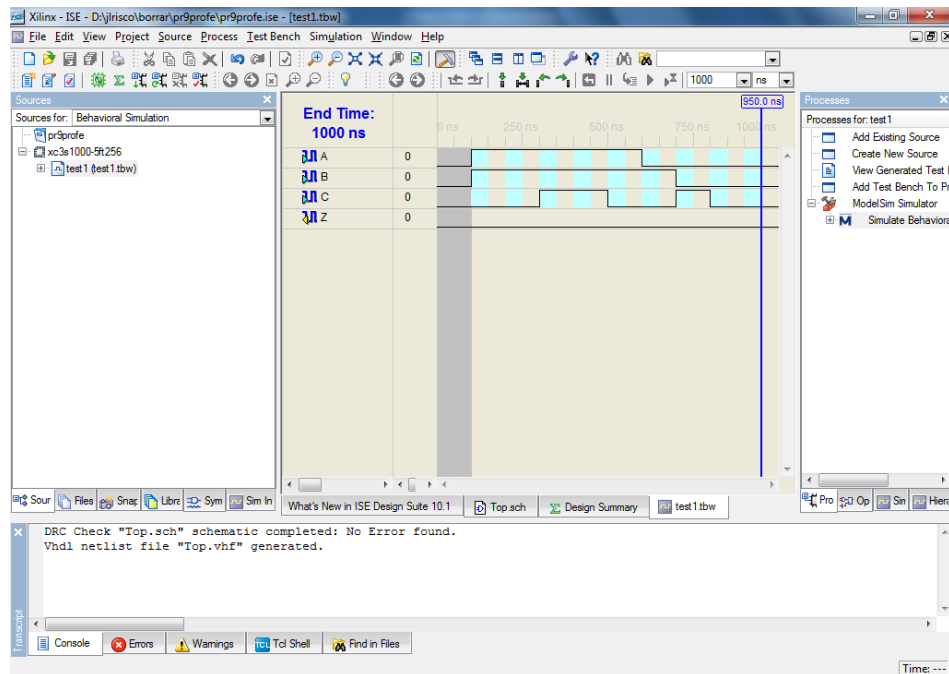
Cuando creéis el proyecto, seleccionad ModelSim XE VHDL como simulador. Es importante que no mováis ni eliminéis el archivo de licencia license.dat de ModelSim. Si lo hacéis, ModelSim dejará de funcionar, y os tocará pedir otra licencia. Os recomiendo que lo guardéis en la carpeta de instalación de ModelSim.

2.2. Cómo guardar el conjunto de señales de una simulación

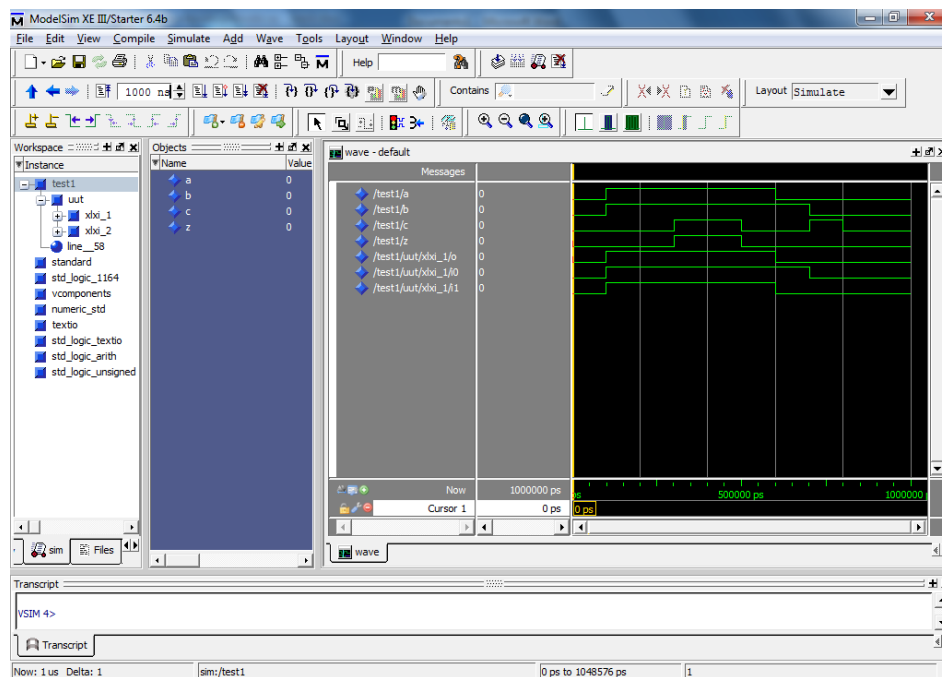
Cuando una práctica es relativamente grande, conviene realizar muchas depuraciones sobre el mismo conjunto de señales para comprobar que todo va bien. Para poder “cargar” un conjunto de señales a visualizar hay que cumplir lo siguiente: Las entradas y salidas del sistema original no deben cambiar. Procedemos de la siguiente forma: En primer lugar simulamos nuestro diseño. Es muy simple e incluye dos puertas AND, una de ellas se llama XLXI.1.



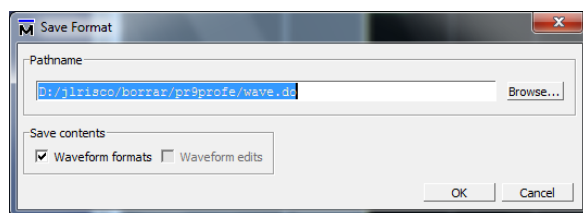
Creamos el testbench en la forma habitual:



Y lanzamos la simulación. En una de las ventanas del ModelSim aparecen las instancias de nuestro diseño. Podemos coger cualquiera de ellas y arrastrarla a la ventana de simulación. En este caso arrastramos XLXI_1:



Ahora podemos guardar las señales. Pulsamos en el disco, del cual se debe leer "Save Format ...", y guardamos el archivo.



Más tarde, si detectamos un error en nuestra práctica y corregimos algo interno (i.e. el conjunto de entradas y salidas sigue siendo el mismo, así como el nombre de los componentes), podemos volver a ejecutar la simulación desde Xilinx, y una vez el ModelSim esté abierto, pulsar File→Close, y File→Load..., cargando el archivo anterior (wave.do). Obtendremos una nueva simulación, pero con el conjunto de señales que definimos inicialmente (se puede pulsar restart y run 1000 ns, para mayor seguridad):

