Cuaderno de prácticas

Laboratorio de Fundamentos de Computadores

PARTE I:
Diseño lógico usando Xilinx ISE

AUTORES:

David Atienza Alonso, Hortensia Mecha López,
Inmaculada Pardines Lence, Silvia Del Pino Gordo

Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

PRÁCTICA 0: Captura y simulación de circuitos digitales

El objetivo de esta primera práctica es la toma de contacto con las herramientas de captura y simulación de circuitos digitales que se utilizarán para la realización de las prácticas de que consta esta parte de la asignatura.

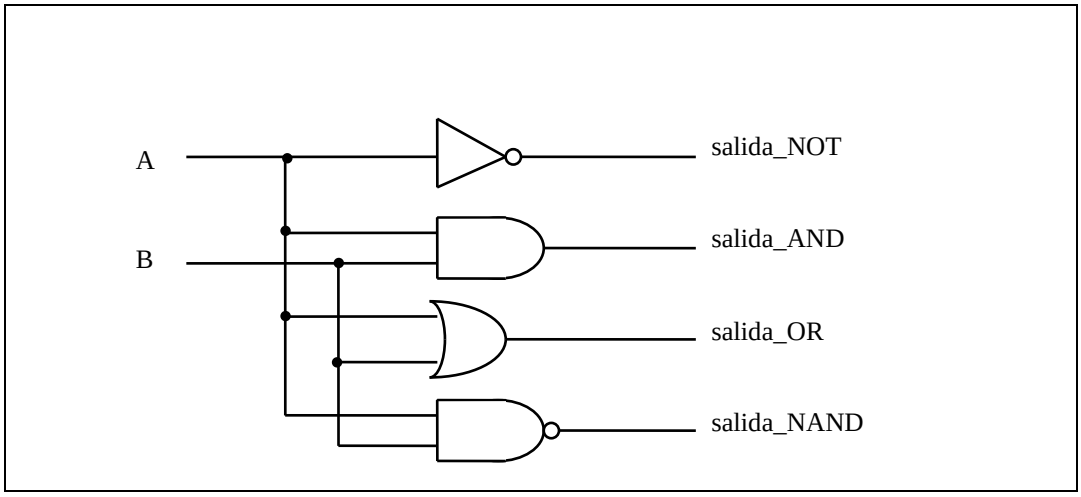
La práctica consistirá en comprobar el funcionamiento de varias puertas básicas (NOT, AND, OR y NAND). A lo largo de este guión se irán indicando todos los pasos a seguir para poder simular el funcionamiento de un circuito digital.

El proceso de realización de una práctica constará habitualmente de tres fases:

- **Diseño** del circuito.
- **Captura del esquema** del circuito.
- **Simulación** del circuito.

El alumno deberá realizar la primera fase **antes** del turno de laboratorio que le corresponda, y sólo las otras dos en el propio laboratorio. En esta primera práctica no será necesario hacer un diseño sobre el papel, previo a la captura, debido a su sencillez. Nos limitaremos a explicar el uso de la herramienta Integrated System Environment (ISE) de Xilinx, que es la aplicación que vamos a utilizar para la realización de estas prácticas, y que permiten llevar a cabo las dos últimas fases.

El circuito que vamos a capturar y simular corresponde al que se muestra en el diseño 1. Como puede verse, consta de dos entradas **A** y **B**, y cuatro salidas, **salida_NOT**, **salida_AND**, **salida_OR** y **salida_NAND**. Para comprobar el funcionamiento de estas puertas se tendrán que generar todas las combinaciones posibles de las variables de entrada y visualizar el valor de la salida para cada una de las funciones. Estos valores deben coincidir con la tabla de verdad de las funciones.



Diseño 1: Puertas lógicas simples

1. Captura del esquema del circuito

Para poder proceder a su simulación, en primer lugar hemos de capturar el esquema del circuito. Es decir, realizar una representación gráfica del diseño mediante componentes (como las puertas lógicas), conexiones entre ellos y entradas y salidas. Para ello tendremos que arrancar la herramienta Xilinx ISE y crear un nuevo proyecto. Un proyecto es un conjunto de ficheros de diseño, tales como esquemáticos, listas de conexionado, librerías de componentes, vectores de test para la simulación, etc., seleccionados para un diseño específico.

1.1 Ejecutar la aplicación

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

Xilinx ISE 9.1i.lnk

o arrancar la aplicación desde el menú Inicio seleccionando:

Inicio → Programas → Electrónica →Xilinx ISE 9.1i → Project Navigator

Nos aparecerá una ventana como la que se muestra en la Figura 1.

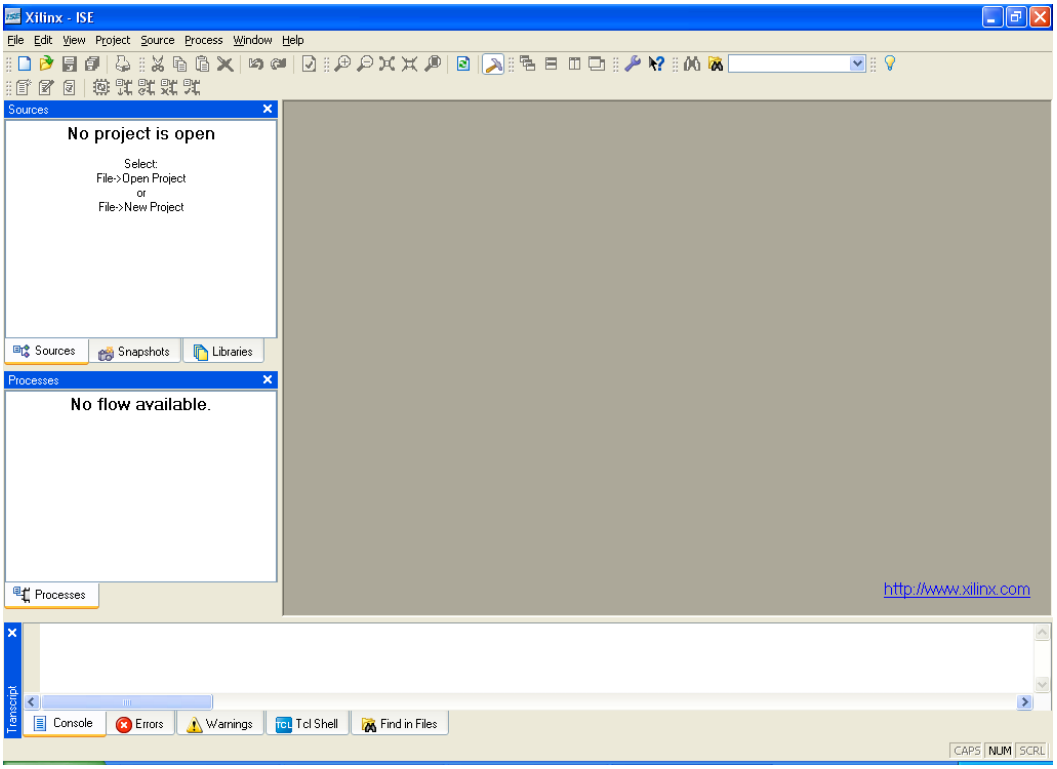


Figura 1: Ventana inicial de Xilinx ISE

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.

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**.

1.2 Acceso a la ayuda

En cualquier momento podemos acceder a la ayuda para consultar información adicional acerca del software ISE. El acceso realizar de dos formas:

- 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.

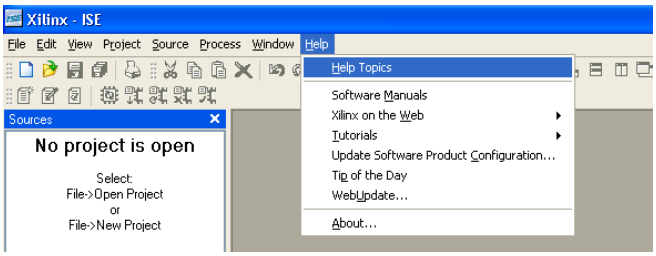


Figura 2: Ventana de ayuda en el ISE

1.3 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 3, donde debemos insertar los datos del proyecto.

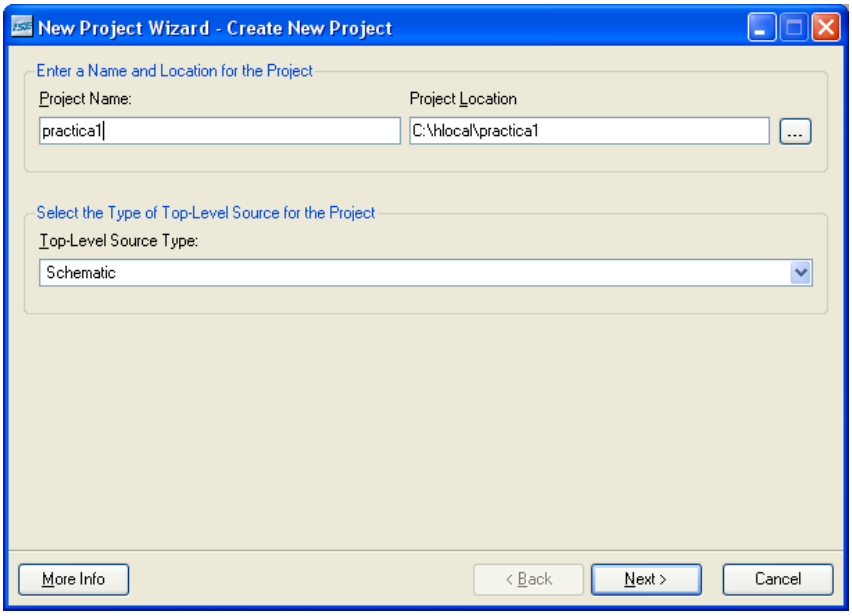


Figura 3: Datos del proyecto

1.3.1 Datos del proyecto

Project Name damos el nombre de nuestro proyecto. Por ejemplo **practica1.**

Project Location damos el camino de datos donde se guarda el proyecto. Aconsejamos utilizar **c:\hlocal\NombreProyecto** En nuestro ejemplo **c:\hlocal\practica1.**

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.

1.3.2 Datos del dispositivo

Después pulsamos el botón de **Next**, lo cual da lugar a la aparición de otra ventana (como la de la Figura 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 Figura 4 (Spartan3 XC3S1000 FT256). El resto de los datos sobre las herramientas de síntesis, simulador, etc, deben seleccionarse también como se muestra en la misma figura.

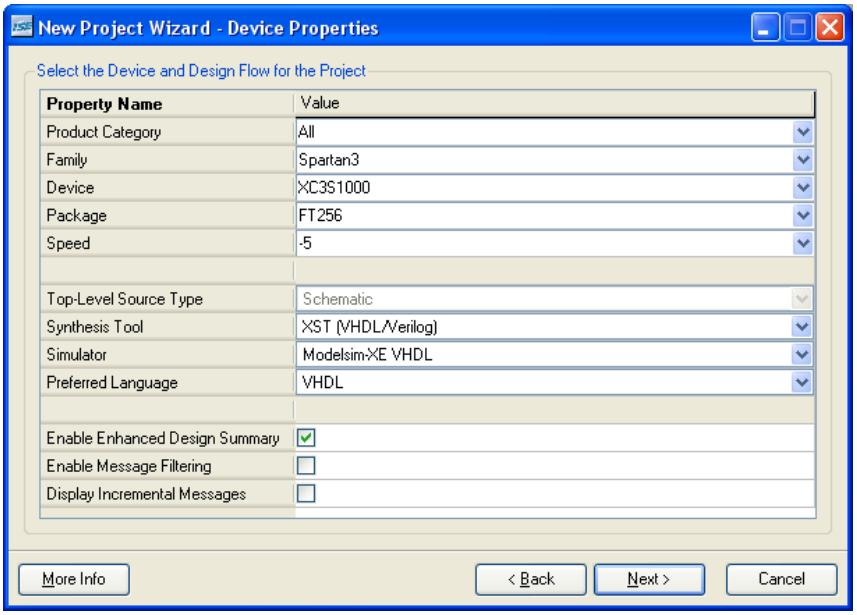


Figura 4: Datos del dispositivo y del flujo de diseño

1.3.3 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.

Volvemos a pulsar **Next** y nos aparece una ventana como la de la Figura 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.

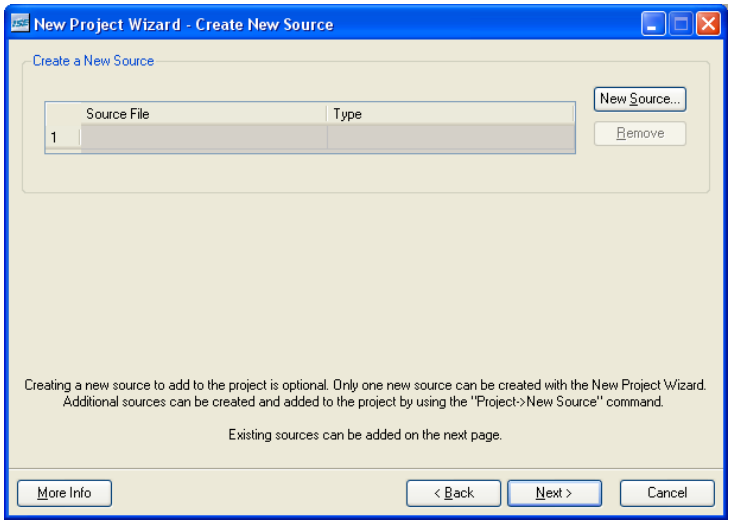


Figura 5: Datos de los nuevos ficheros fuente del proyecto

1.3.4 Ficheros nuevos de nuestro proyecto

Pulsamos **New Source** y 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**, como se muestra en la Figura 6**.**

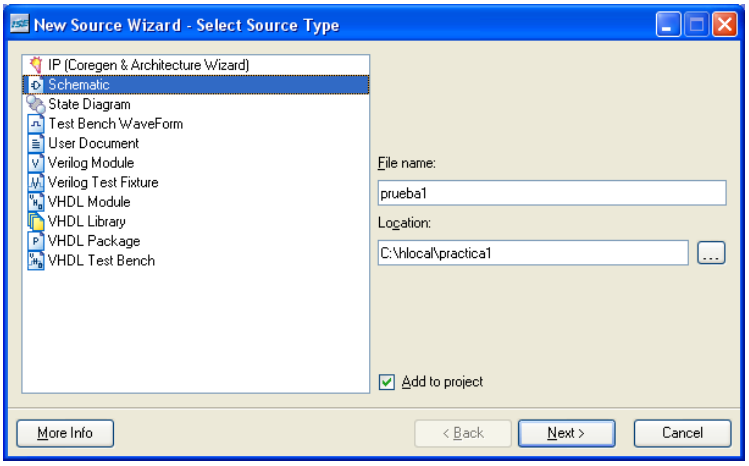


Figura 6: Datos de nuestro fichero fuente

Pulsamos **Next** y **Finish**.

Si el directorio no existe nos pregunta que si queremos crearlo. En este caso diremos **Yes**. A continuación nos aparece otra ventana, parecida a la de la Figura 5, pero con los datos de los ficheros fuente que hemos indicado que queremos crear, tal y como se muestra en la Figura 7.

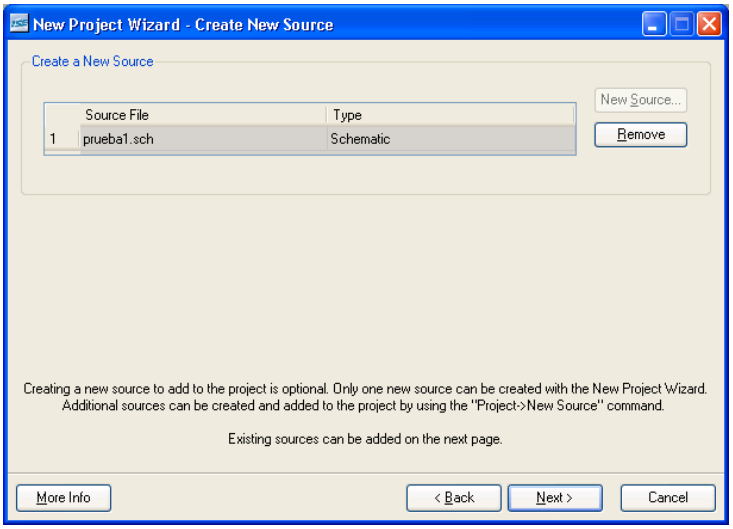


Figura 7: Datos de ficheros fuente nuevos que queramos crear en nuestro proyecto

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 8, que sirve para incluir en el proyecto otros ficheros fuente generados anteriormente. Si no vamos a incluir otros ficheros, pulsamos **Next** y **Finish**.

1.3.5 Ficheros existentes, reutilización en nuestro proyecto.

El método es exactamente el mismo del proceso de crear nuevos ficheros fuente (apartado 1.3.4), 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 esquemático de algún diseño previamente realizado, pulsaremos **Add Source.**

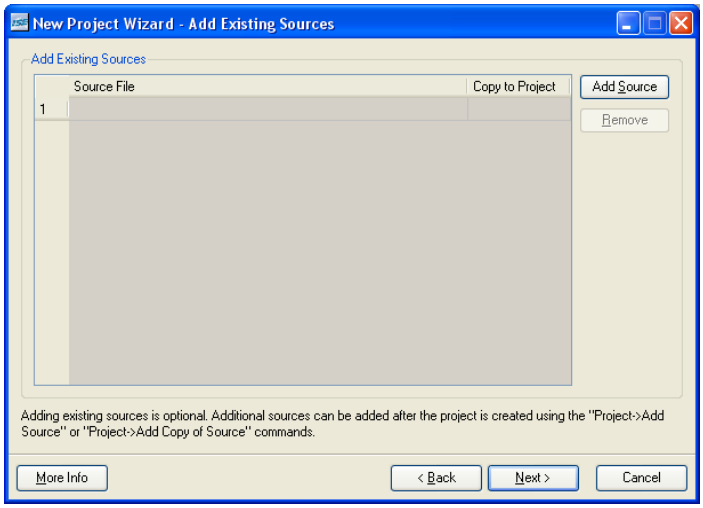


Figura 8: Datos de ficheros fuente ya creados que queramos añadir al proyecto

Seleccionamos el fichero que queremos añadir (**prueba1.sch**) y a continuación **Next**, **Finish** y **OK**. Con esto hemos terminado el proceso de Inicialización.

1.3.6 Apertura de un proyecto existente

Simplemente seleccionamos **Archive→Open Project** y seleccionamos el proyecto que queremos abrir (con la opción **Tipo ISE Project Files**) en la ventana correspondiente (ver Figura 9).

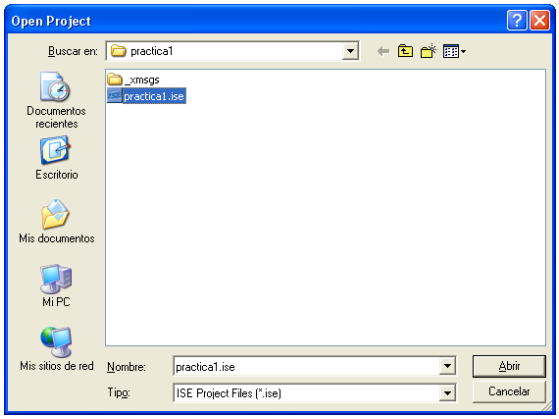


Figura 9: Apertura de un proyecto existente

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 10. Observamos que en el **workspace** aparece la pestaña "Design Summary", y que en la ventana **Sources** está seleccionada la pestaña **Sources**.

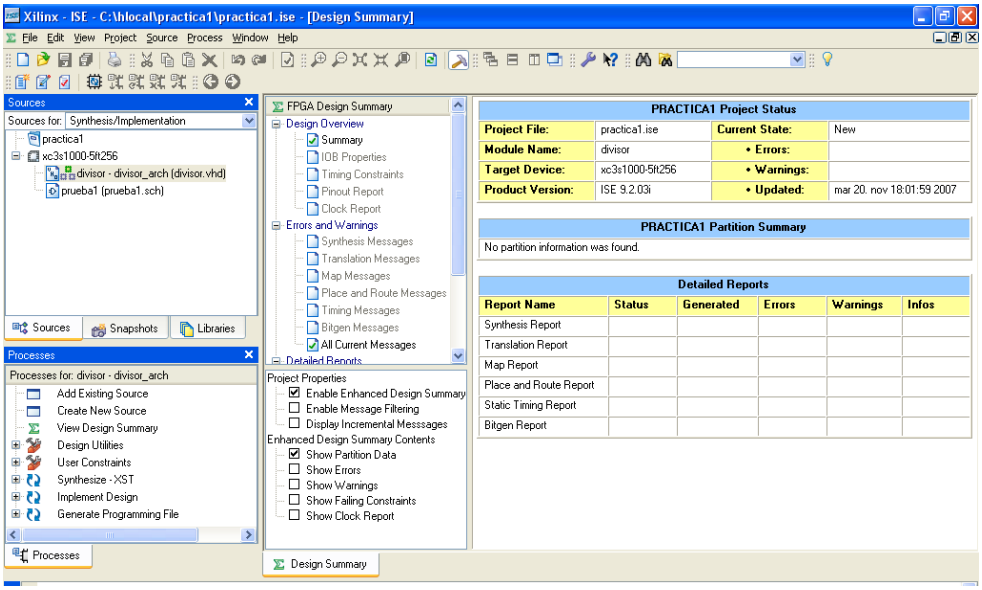


Figura 10: Entorno de Xilinx ISE

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 11.

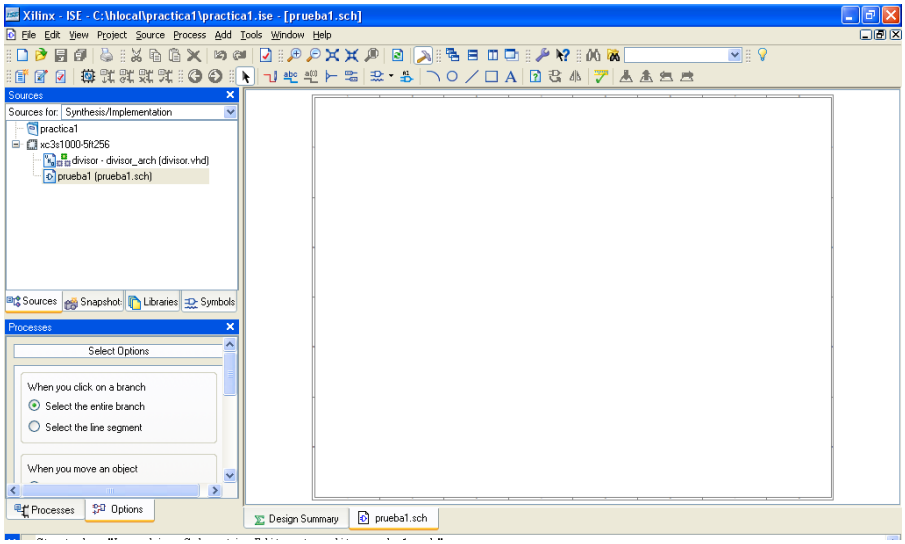


Figura 11: Entrada de esquemáticos

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 12). En la ventana para edición dibujaremos nuestro diseño.

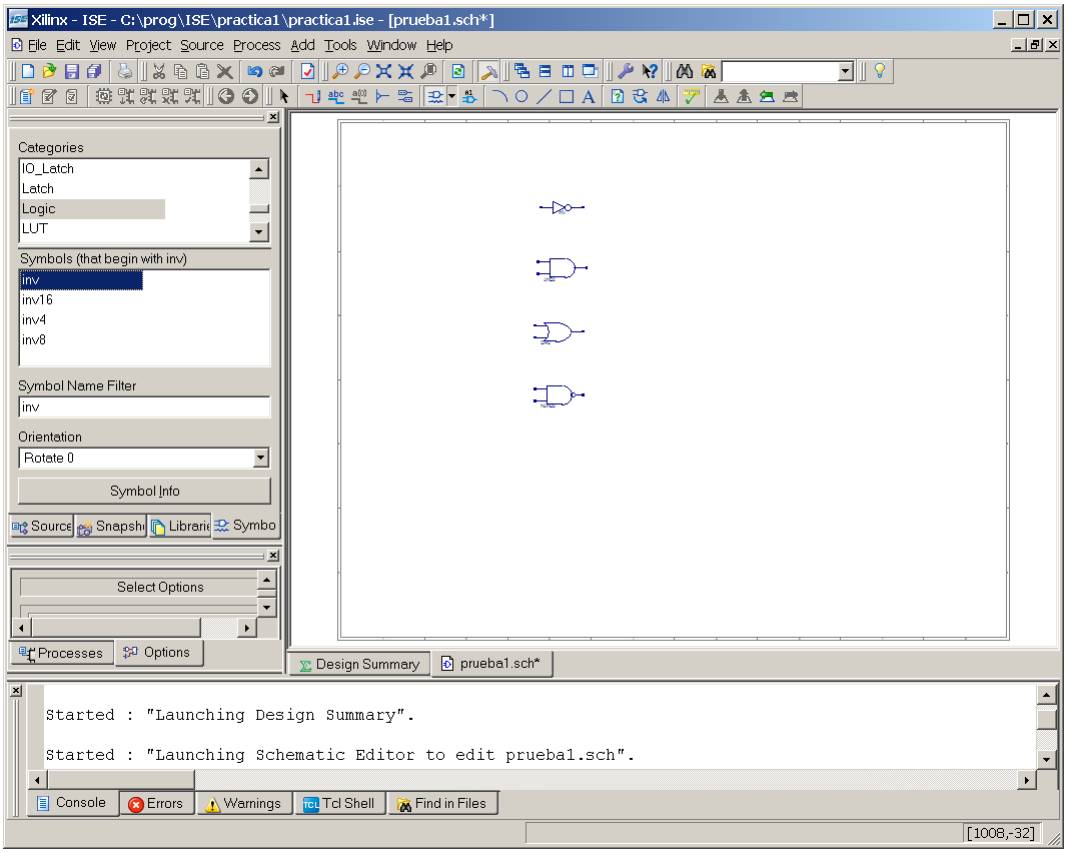


Figura 12: Entrada de esquemáticos con menú de símbolos

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 colocándolos en la ventana de edición. En el ejemplo vemos que estamos generando un esquemático que incluye distintos tipos de puertas básicas (panel de la derecha en la Figura 12) a partir de los símbolos de la categoría **Logic** (panel de la izquierda en la Figura 12): inversor (**inv**), and de 2 entradas (**and2**), or de 2 entradas (**or2**) y nand de 2 entradas (**nand2**).

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 13.

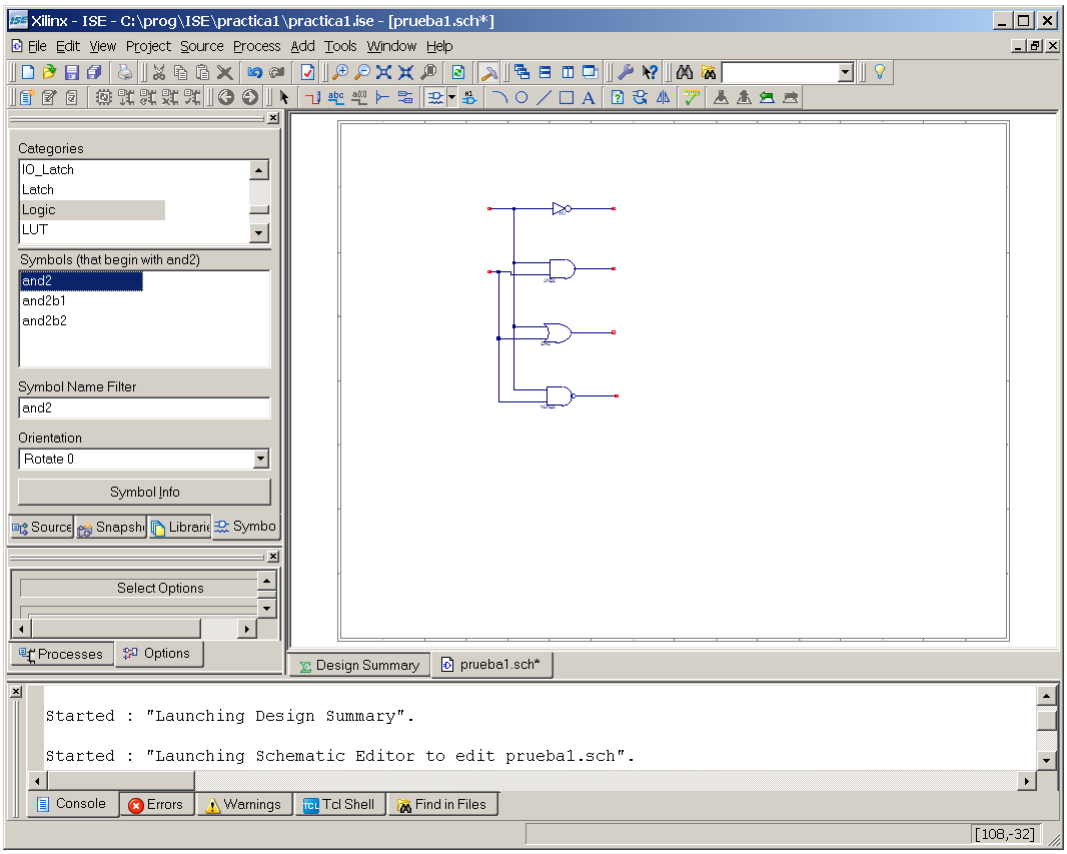


Figura 13: Puertas lógicas e interconexiones sin Buffers de E/S

En el caso que vayamos a implementar el diseño, nos falta añadir los IBUF y OBUF (categoría **IO**, símbolos **ibuff** y **obuf**), y los pines de entrada-salida con el botón  que se encuentra también en la barra de herramientas. En este caso las entradas las nombraremos como **A** y **B**, y las salidas recibirán el nombre **salida_not**, **salida_and2**, **salida_or2** y **salida_nand2**.

El resultado es el que se muestra en la Figura 14. Una vez terminado es aconsejable salvar el diseño.

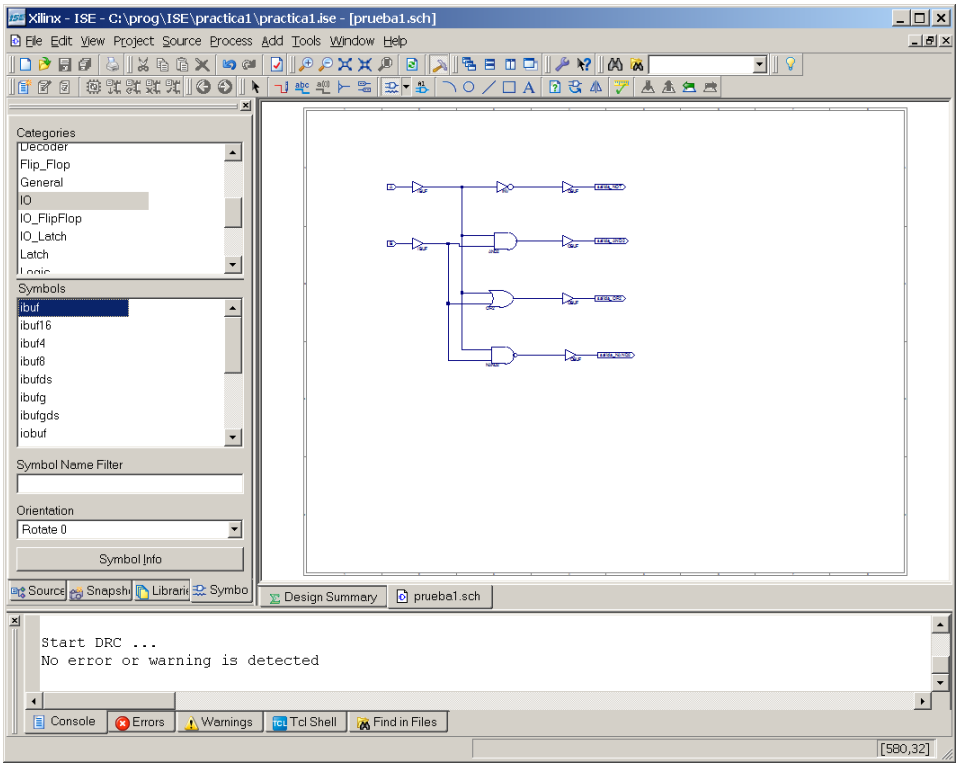


Figura 14: Puertas y entradas con Buffers de E/S

1.5 Creación del NETLIST

Para que el fichero pueda ser simulado se debe generar el listado de conexiones (*Netlist*) del mismo. 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 de mensajes nos informa de alguna advertencia o error, se debe remediar antes de continuar

Llegado a este punto estamos en disposición de pasar a simular el circuito, para esta fase se requiere (aparte del ISE) un programa adicional de simulación lógica. En este laboratorio utilizaremos el programa de simulación Mentor Modelsim. El acceso a dicho programa se encuentra plenamente integrado en el interfaz gráfico del ISE, por lo que puede ser invocado sin cambiar de entorno, como veremos a continuación.

2 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.

2.1 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 (Figura 15) 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 15.

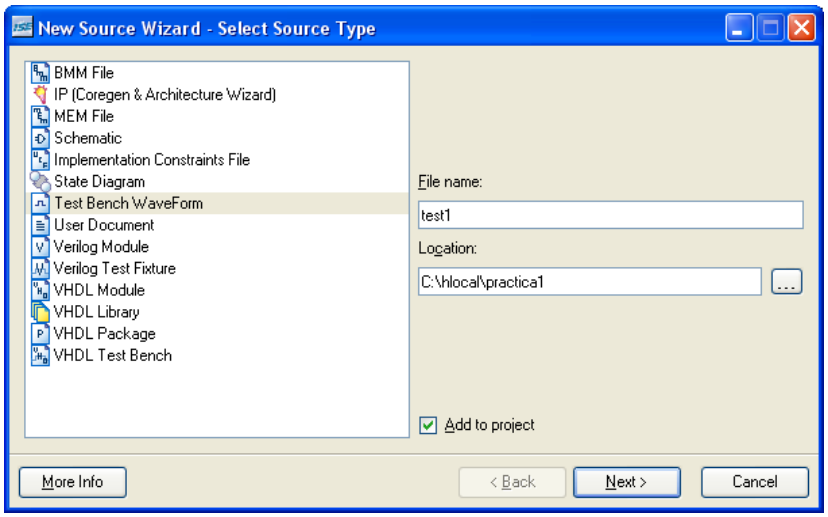


Figura 15: Generación de un testbench

Pulsamos **Next** y en la siguiente ventana (Figura 16) nos piden el nombre de la fuente con la cual queremos asociar el **testbench**. En nuestro ejemplo seleccionaremos **prueba1.**

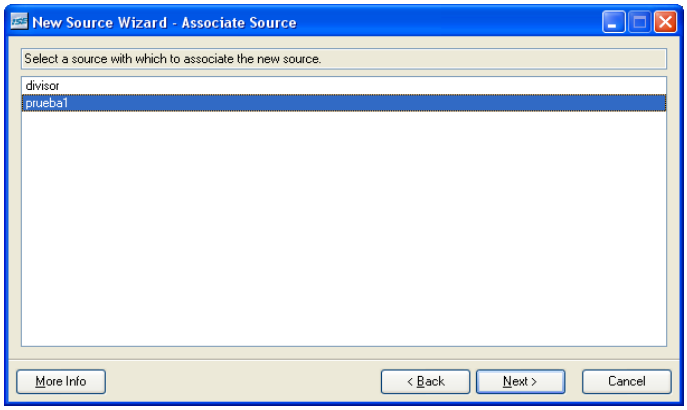


Figura 16: Fuente asociada al Testbench

Pulsamos **Next** y **Finish**.

A continuación nos aparece una ventana con datos temporales para la simulación del diseño (Figura 17), 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**.

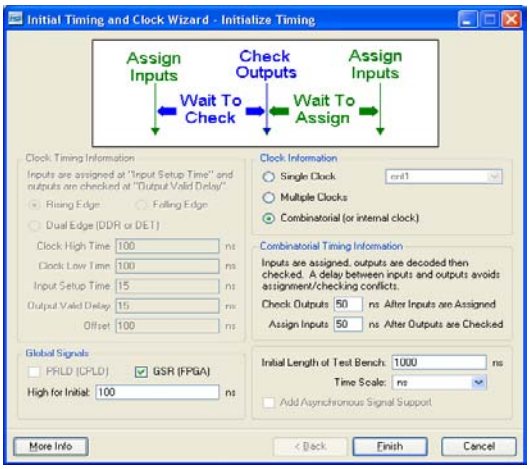


Figura 17: Datos temporales de la simulación

En la siguiente ventana (Figura 18), nos aparece el valor que se ha asignado de forma automática, de todas las entradas y salidas de nuestro circuito.

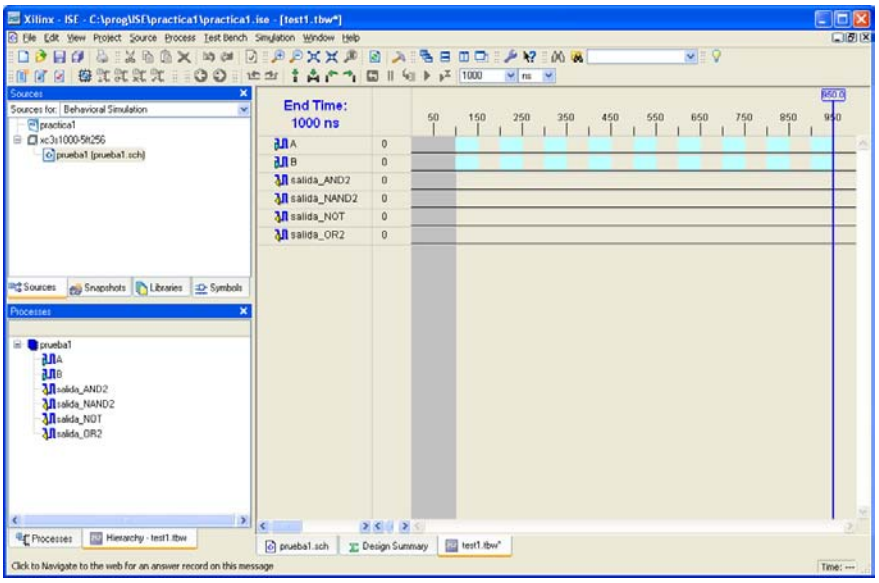


Figura 18: Formas de onda iniciales del TestBench

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. Simplemente pulsando en las correspondientes formas de onda, donde queramos cambiar su valor, se modifican automáticamente. Un posible **Testbench** final que cubre las 4 posibles combinaciones de la señales de entrada es el que se muestra en la Figura 19. Una vez terminado, salvar las formas de onda.

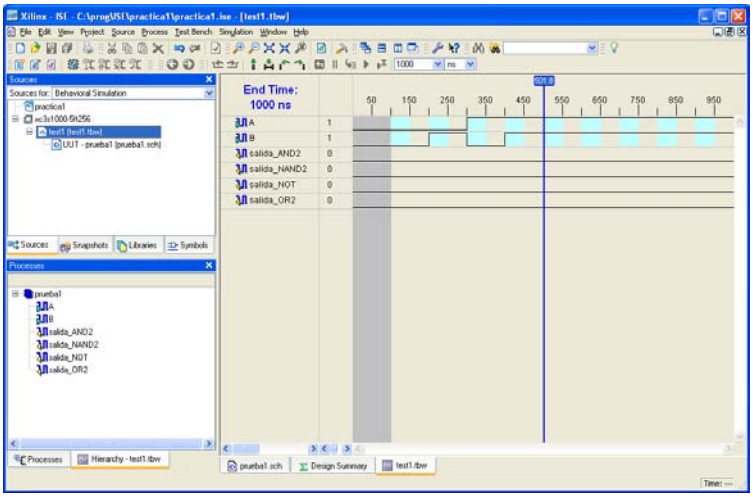


Figura 19: Formas de onda finales del TestBench

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.

2.1.1 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 20.

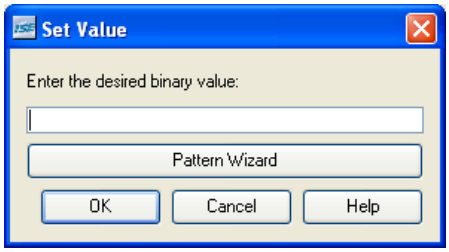


Figura 20: Dialogo para fijar valores de simulación para variables

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 21 .

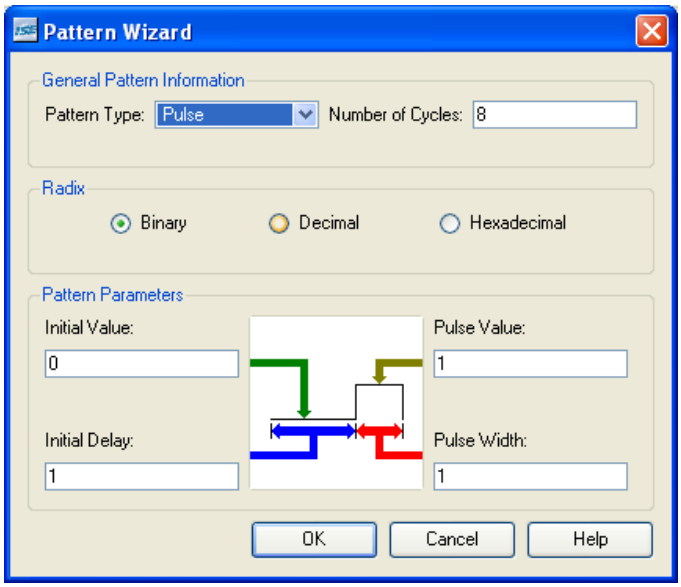


Figura 21: Formas de onda finales del TestBench

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 Value***), 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. De manera similar, si 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.

2.2 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 22). 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**.

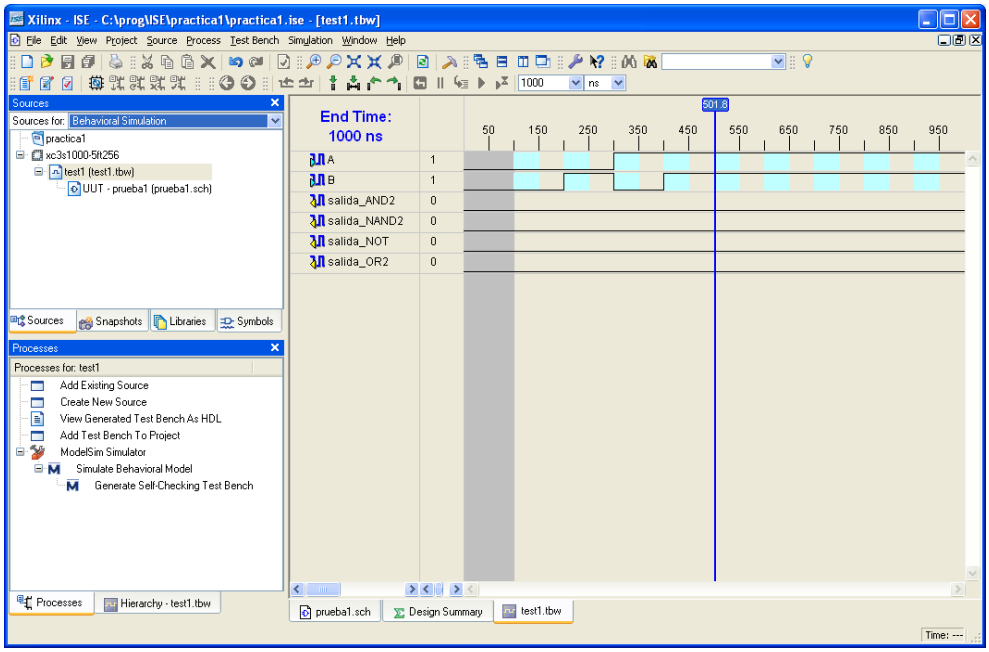


Figura 22: Inicio de la simulación utilizando ModelSim

El resultado de la simulación se presenta en la Figura 23. En algunas ocasiones hay que cerrar una ventana que aparece en el workspace, justo antes de la visualización de la simulación. Es necesario centrar la ventana de la simulación y ajustar el zoom para observar correctamente el resultado.

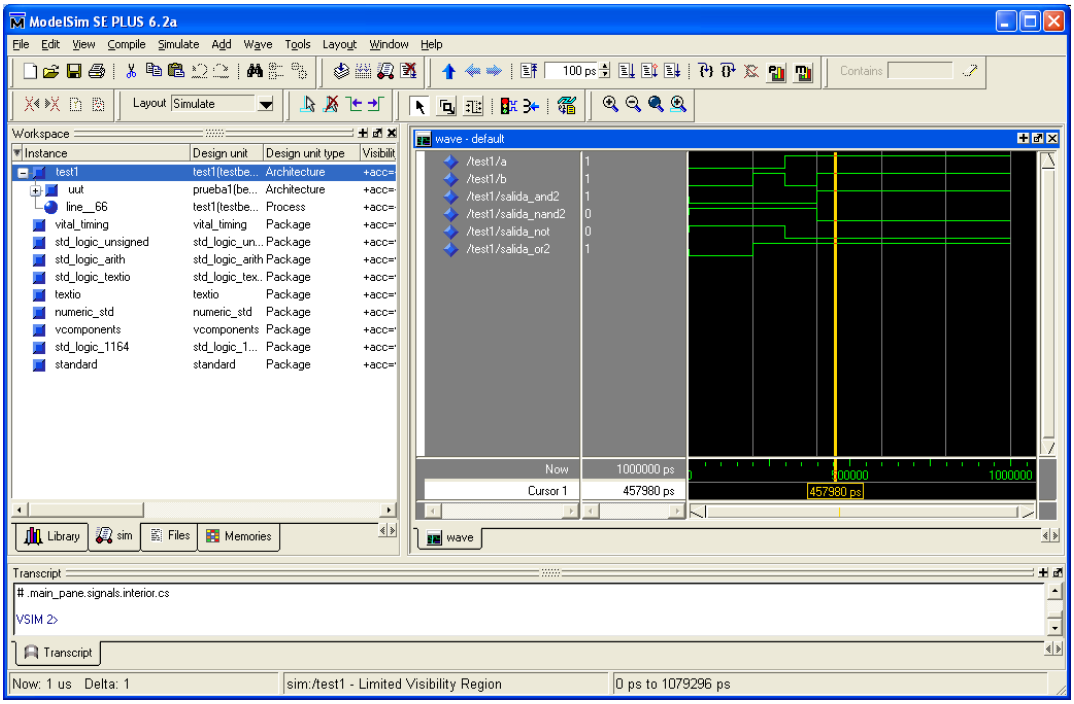


Figura 23: Resultado de la simulación del TestBench

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 2.1 y luego se vuelve a ejecutar la simulación seleccionando, en la ventana de procesos, Modelsim o algún otro simulador alternativo que se encuentre instalado.

PRÁCTICA 1: Diseño de un comparador.

OBJETIVO: Diseño completo de un circuito combinacional usando puertas lógicas.

ESPECIFICACIÓN:

Alto nivel:

Entradas:	X	X ϵ { 0, 1, 2, 3 }
	Y	Y ϵ { 0, 1, 2, 3 }
Salida:	S	S ϵ { 0, 1, 2 }

Función de alto nivel:
0 si X = Y
1 si X < Y
2 si X > Y

PASOS A SEGUIR:

1.- Codificar en binario puro las entradas y las salidas del sistema.

2.- Obtener las tablas de verdad del comportamiento del circuito en términos de la codificación realizada en 1.

3.- Especificar el circuito usando expresiones canónicas.

4.- Simplificar las expresiones canónicas usando diagramas de Karnaugh.

5.- Diseñar el circuito usando dos niveles de puertas AND/OR (símbolos **andX** y **orX** de la librería de símbolos del ISE, sustituyendo la '**X**' de los nombres de los símbolos por el número de entradas de cada puerta deseada)

6.- Capturar y simular el circuito del punto 5.

7.- Convertir el circuito del punto 5 en dos niveles de puertas NAND (símbolos **nandX** de la librería de símbolos del ISE).

8.- Captura y simulación del circuito del punto 7.

IMPLEMENTACIÓN:

Nombres de los diferentes puertos:

- Vector X: X0 y X1 (en mayúsculas).
- Vector Y: Y0 y Y1 (en mayúsculas).
- Vector S: S0 y S1 (en mayúsculas).

Nombre de los archivos:
- Paso 6: Pr11.sch
- Paso 8: Pr12.sch

NOCIONES TEÓRICAS:

- Formas canónicas de las expresiones de conmutación.
- Mapas de Karnaugh.
- Codificación en binario.
- Equivalencia de dos niveles de puertas AND - OR y dos niveles de puertas NAND.

1 TestBench exhaustivos en Xilinx ISE

En la simulación del esquemático diseñado en esta práctica debemos hacer una prueba exhaustiva de todas las posibles configuraciones de las entradas (tabla de verdad) y verificar que las salidas toman el valor esperado. Para ello debemos forzar **estímulos periódicos** en las señales de entrada, empezando por la señal menos significativa, continuamos con las siguientes doblando el período de la señal anterior, y así sucesivamente hasta la más significativa. Este tipo de estímulos se pueden introducir según las indicaciones de la Sección 2.1.1.

Las formas de onda que deben fijarse en el TestBench de esta práctica mediante la definición de patrones se muestran en la Figura 24, donde podemos observar que aparecen ordenadamente todos los casos de la tabla de verdad.

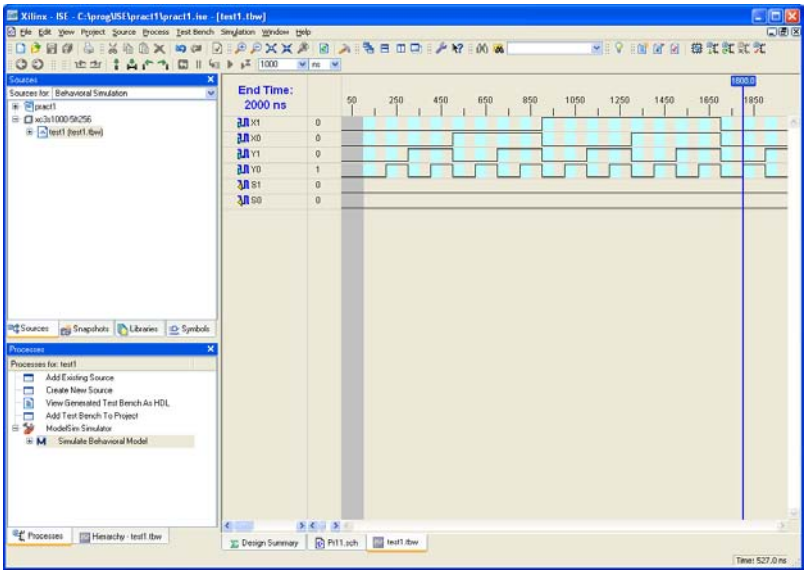


Figura 24: TestBench de la simulación de la tabla de verdad

Las ondas resultantes de esta simulación en Modelsim se muestran en la Figura 25, donde podemos observar que aparecen ordenadamente todos los casos de la tabla de verdad.

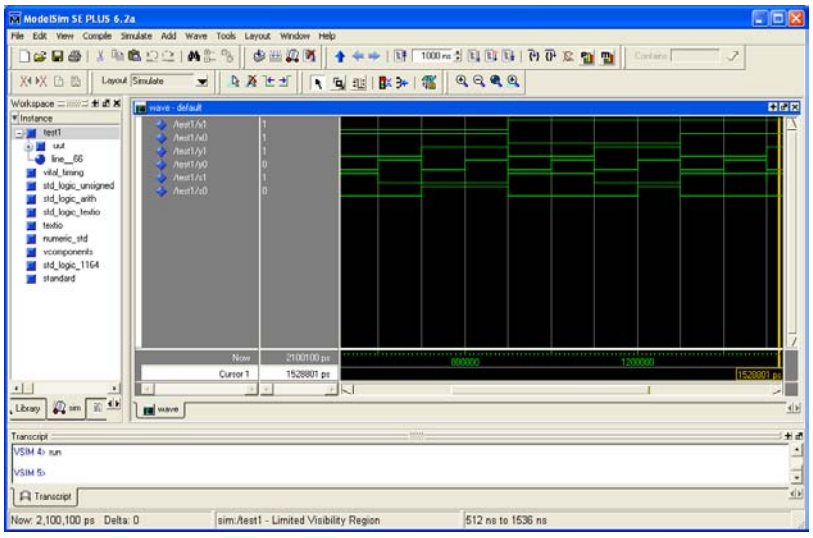


Figura 25: Resultados de la simulación en Modelsim del comparador

PRÁCTICA 2: Diseño de un sumador.

OBJETIVO: Diseño modular y jerárquico de circuitos combinacionales.

ESPECIFICACIÓN:

Alto nivel:

Diseño de un sumador de números positivos, codificados en binario puro de 4 bits; poseerá una línea de C_{in} (para el acarreo de entrada) y otra de C_{out} (para el acarreo de salida).

Binaria:

Entradas: $X, Y \in \{0, 1\}^4$ / X, Y en binario puro
$C_{in} \in \{0, 1\}$

Salidas: $S \in \{0, 1\}^4$ / S en binario puro
$C_{out} \in \{0, 1\}$

Función:

$S = (X+Y+C_{in}) \bmod 16$
$C_{out} = 1$ si $X+Y+C_{in} > 15$
0 en caso contrario

PASOS A SEGUIR:

1.- Implementar el circuito sumador de un bit usando puertas NAND y XOR (símbolos **nandX** y **xorX** de la librería de símbolos del ISE, sustituyendo la '**X**' de los nombres de los símbolos por el número de entradas de cada puerta deseada)).

.

2.- Captura y simulación del sumador de un bit.

3.- Interconectar los cuatro sumadores de un bit entre sí y con puertos externos.

4.- Capturar y simular el circuito completo.

5.- Diseñar jerárquicamente en dos niveles el sumador completo.

6.- Capturar el diseño jerárquico.

IMPLEMENTACIÓN:

Nombres de los diferentes puertos:

- Entradas del sumador de un bit: X, Y y Ci.
- Salida del sumador de un bit: Z y Co.
- Vectores de entrada del sumador de 4 bits X=(X0,X1,X2,X3), Y=(Y0,Y1,Y2,Y3).
- Vector de salida del sumador de 4 bits S=(S0,S1,S2,S3).
- Entrada y salida de acarreo del sumador de 4 bits: Cin y Cout.

Nombre de los archivos:
- Paso 2: Pr21.SCH.
- Paso 4: Pr22.SCH.
- Paso 6: Pr23.SCH.

1 Nociones teóricas

El elemento de suma más sencillo es el llamado **semisumador**, que tiene dos entradas, los dos bits a sumar, y dos salidas, el bit de suma y el bit de acarreo. A partir de este módulo podemos desarrollar el **sumador completo** de un bit, que tiene tres entradas: los dos bits a sumar y un acarreo de entrada, y dos salidas: el bit de suma y el bit de acarreo de salida.

La forma más sencilla de implementar un sumador de números codificados en binario con *n* bits es mediante el "**encadenamiento**" de *n* sumadores completos de un bit. Este método se corresponde con el método de suma "a mano" de dos números en binario. Los sumadores se disponen en batería, sumando cada uno de ellos cada par bits de igual peso de las entradas, y se conectan los bits de acarreo de forma que el acarreo de salida del sumador de peso *i* sea el acarreo de entrada del sumador de peso *i+1*. Finalmente, el acarreo de entrada del sumador de peso 0 se pone al valor lógico 0, y el acarreo

de salida del sumador de mayor peso (*n-1*) constituye el acarreo de salida de la suma total.

2 Nociones de Xilinx ISE

2.1 Diseño jerárquico en Xilinx ISE

Para realizar un diseño jerárquico necesitamos crear un símbolo a partir de un diseño ya existente. El símbolo consistirá en un nuevo componente donde el usuario sólo verá un bloque con el mismo número de entradas y de salidas que el esquema original, que englobará a éste y realizará su misma función. Este símbolo que se crea, pasa a formar parte de la librería de componentes del proyecto, y podrá utilizarse como un dispositivo más. Como ejemplo, vamos a generar un símbolo llamado **prueba1** que representa el conjunto de puertas lógicas básicas de la práctica 0 (**prueba1.sch**). Para crearlo, una vez terminado el diseño vamos a la ventana de fuentes (vista **Synthesis/Implementation**) y seleccionamos el fichero fuente que contiene el conjunto de puertas lógicas. 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 26.

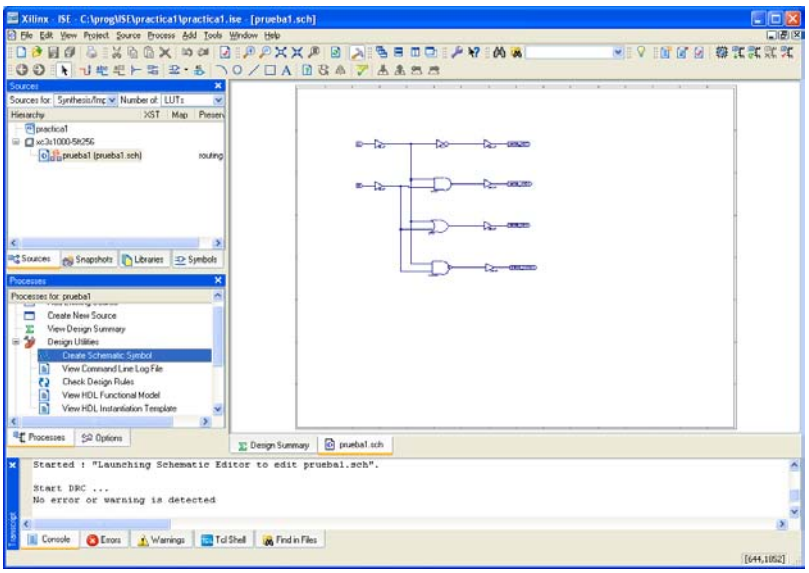


Figura 26: Generación de un símbolo asociado a un esquemático

Para ver que la operación se ha realizado de forma satisfactoria, y que tenemos el símbolo disponible en la librería, se accede a la ventana de fuentes y seleccionamos la pestaña **Symbol**. En la categoría c:/hlocal/practica1 aparece el nuevo símbolo **prueba1**, como se muestra en la Figura 27.

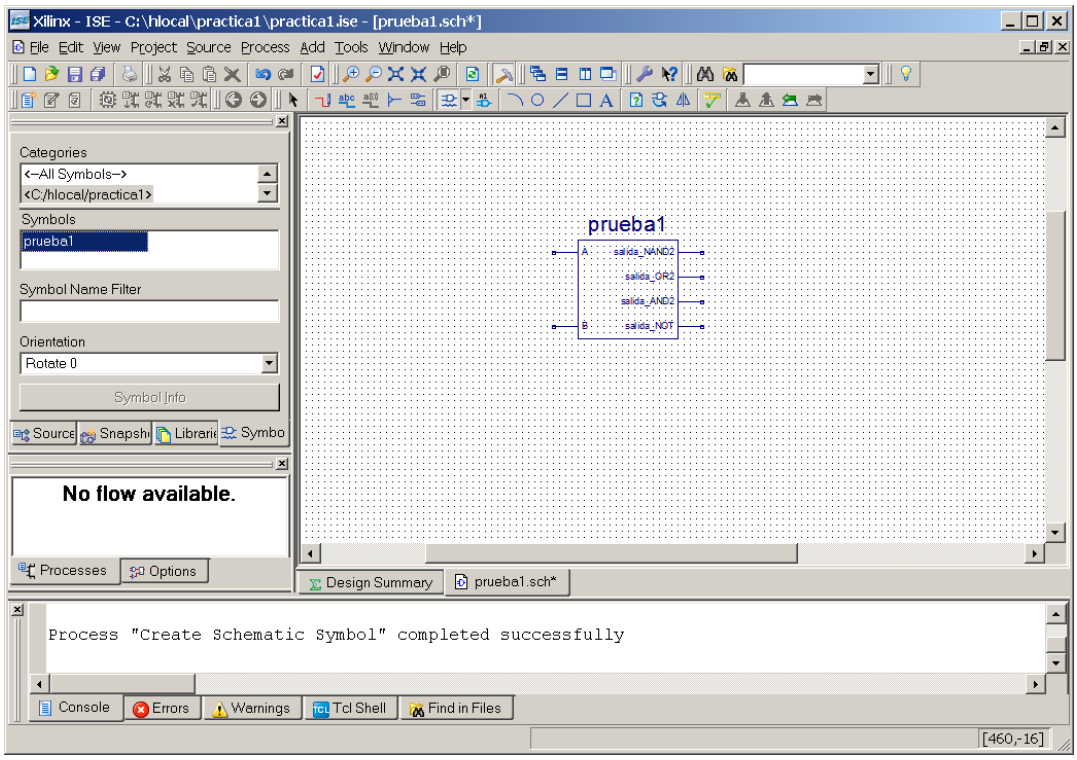


Figura 27: Esquemático del nuevo símbolo prueba1

De este modo, el módulo **prueba1** puede añadirse a un nuevo esquemático (por ejemplo, **jerarquico.sch**) como cualquier símbolo de las librerías del ISE. Tras añadir el módulo **prueba1** al nuevo esquemático, al seleccionar la pestaña **Sources**, vemos que el nuevo módulo está dentro del esquemático jerarquico.sch al ser un diseño jerárquico, como se muestra en la Figura 28.

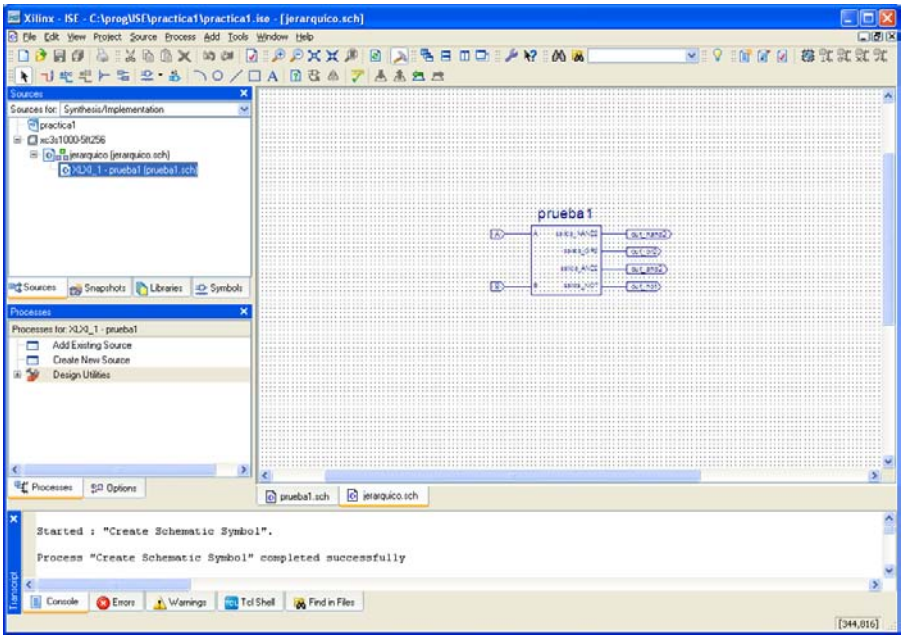


Figura 28: Diseño jerárquico de esquemáticos

La realización de símbolos es una herramienta muy útil a la hora de diseñar circuitos más complicados y que además tengan partes que se repitan. Facilita el diseño jerárquico y estructurado, permitiendo una mayor claridad a la hora de presentar un circuito y entender su funcionamiento.

2.2 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 crear los pines de entrada-salida con el botón  de la barra de herramientas, nos aparecerá una ventana como la que se muestra en la Figura 29, donde tendremos que indicar el nombre, el tipo de terminal y el número de bits del bus, según el formato **NOMBRE(NUM-BITS)**.

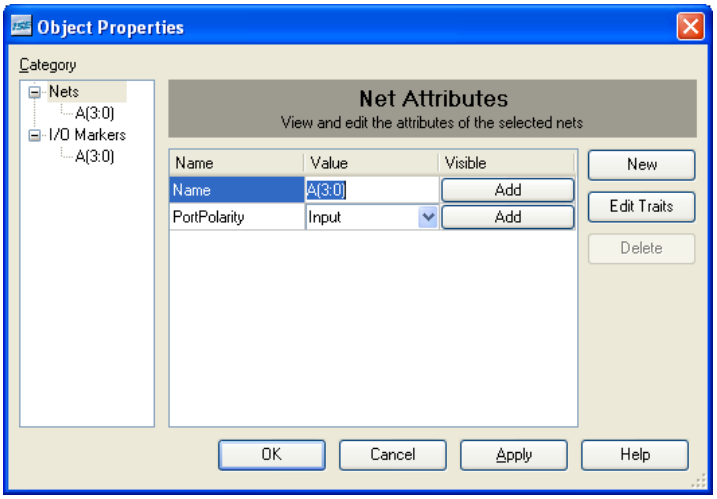


Figura 29: Pantalla de definición de propiedades del bus

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 30.

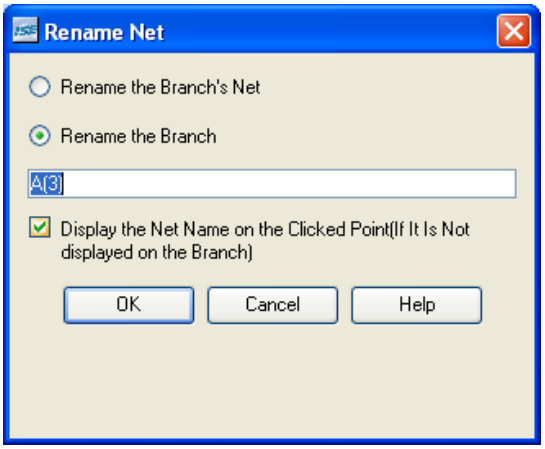


Figura 30: Pantalla para renombrar un cable de conexión

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 **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 31.

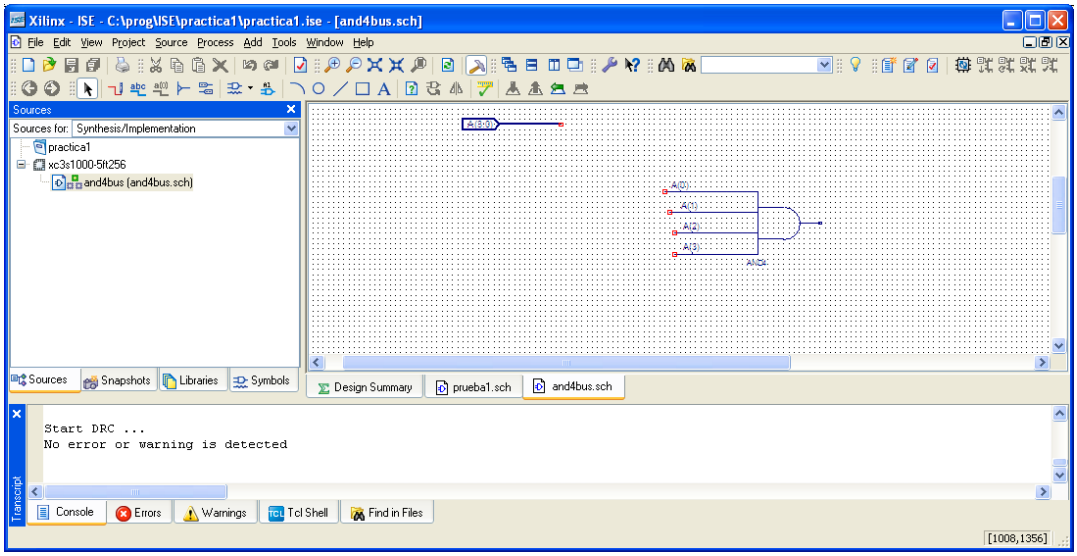


Figura 31: Uso de un bus de entrada para una puerta and4

PRÁCTICA 3: Diseño de un generador de secuencia ascendente / descendente.

OBJETIVO: Diseño completo de un circuito secuencial usando biestables.

ESPECIFICACIÓN:

Alto nivel:

Diseño de un circuito secuencial capaz de generar dos secuencias binarias distintas de valores por su salida S = (s_2, s_1, s_0), en función de una entrada de control C:

Si C = 0 la secuencia será 0, 3, 4, 2, 5, 1, 6, 7, 0 ...
Si C = 1 la secuencia será 0, 7, 6, 1, 5, 2, 4, 3, 0 ...

En el momento en que varíe el valor de la entrada de control el circuito proseguirá contando en la otra secuencia desde el punto en donde se encontraba.

PASOS A SEGUIR:

1.- Obtener el diagrama de estados del sistema como máquina de Moore.

2.- Codificar los estados con el valor de la salida que les corresponde, de forma que la salida sea directamente el estado.

3.- Construir la tabla de transición de estados del generador de secuencias.

4.- Diseñar el sistema usando biestables tipo D disponibles en la librería de símbolos (pestaña ***Symbols***) con el nombre **fdc**.

5.- Captura y simulación del circuito diseñado en el punto 4.

6.- Diseñar un módulo combinacional traductor de código que convierta la secuencia que genera un contador convencional en la que nosotros deseamos.
Utilizar para la implementación de la expresión de conmutación del bit más significativo un multiplexor de 8 a 1, que se encuentra en la librería con el nombre **m8_1e**.
Utilizar para la implementación de la expresión de conmutación del segundo bit un decodificador de 3 a 8, que se encuentra en la librería con el nombre **d3_8e**, y las puertas OR (**orX**) necesarias.
Utilizar para la implementación de la expresión de conmutación del bit menos significativo dos niveles de multiplexores de 4 a 1, que se encuentra en la librería con el nombre **m4_1e**.

7.- Captura y simulación del circuito combinacional diseñado en el punto anterior.

8.- **Opcional**: Implementar un contador convencional de 0 a 7. Crear un símbolo a partir de este contador y otro a partir del traductor de código implementado con multiplexores. Conectar los dos símbolos. Captura y simulación del circuito resultante.

IMPLEMENTACIÓN:

Nombres de los diferentes puertos:
- Línea de entrada de control: C.
- Vector de salida: S=(S2,S1,S0).
- Inicialización a 0 de los biestables (Q=0): CLR=1
- Señal de reloj: CLK

Nombre de los archivos:
- Paso 5: Pr31.SCH.
- Paso 7: Pr32.SCH.
- Paso 8: Pr33.SCH.

1 Nociones teóricas

1.1 Multiplexores

Un multiplexor de 2^n a 1 es un módulo combinacional con 2^n entradas de datos (*x*), n entradas de control (*s*), una entrada de capacitación (*E*), y una salida *y* que verifica la siguiente expresión de conmutación:

$$y = E \cdot \sum_{i=0}^{2^{n-1}} x_i \cdot m_i(s_{n-1} \dots s_0)$$

Las entradas de control seleccionan cuál de las entradas de datos se ofrece como salida. Por lo tanto, las líneas de control representan la codificación binaria de la entrada de datos a seleccionar. Por ejemplo, en un multiplexor de 16 a 1 (16 líneas de datos x_{15} a x_0, cuatro líneas de control s_3 a s_0, y una salida *y*), la asignación de las líneas de control al vector 0101 selecciona como salida la línea de datos x_5.

La idea de la síntesis usando multiplexores consiste en utilizar las variables de entrada de la función a implementar como entradas de control del multiplexor, y fijar cada entrada de datos del multiplexor al valor lógico asociado al mintérmino correspondiente.

1.2 Descodificadores

Un descodificador de n a 2^n, es un módulo combinacional con n entradas de datos (*x*), una entrada de capacitación (*E*), y 2^n salidas (*y*) que verifican la siguiente expresión de conmutación:

$$y_i = E \cdot m_i(x_{n-1} \dots x_0)$$

Las entradas de datos representa la codificación en binario del número de la línea de salida que se activa, es decir, que toma el valor lógico 1, mientras que el resto de salidas permanecen a nivel lógico 0. Por ejemplo, en un descodificador de 4 a 16 (4 líneas de entrada de datos x_3 a x_0, y 16 líneas de salida y_{15} a y_0), la asignación de las entradas al vector 1001 activará la línea y_9, permaneciendo el resto de líneas inactivas.

Para implementar funciones de conmutación con descodificadores utilizaremos las variables de entrada de la función como entradas de datos del descodificador. Podemos observar que cada línea de salida representa un mintérmino diferente de la función, de este modo la función podrá implementarse como suma lógica (OR) de sus correspondientes mintérminos.

1.3 Máquinas de estados

Las máquinas de Moore y de Mealy son dos métodos alternativos de modelar sistemas secuenciales, ofreciendo dos maneras de implementar el mismo sistema. La diferencia entre ambos radica en el modo que tienen de generar la salida: mientras que en la máquina de Mealy cualquier cambio que se produzca en la entrada se manifiesta en la salida, aún cuando no se modifique su estado, en la de Moore la salida sólo cambiará cuando el estado de la máquina cambie.

Formalmente podemos expresarlo:

Máquina de Mealy	Máquina de Moore
z(t) = H(s(t), x(t)) s(t+1) = G(s(t), x(t))	z(t) = H(s(t)) s(t+1) = G(s(t), x(t))

1.3.1 Biestables D

Un biestable (flip-flop) es un circuito secuencial de dos estados, cuya salida binaria de un bit corresponde con su estado. Como cualquier sistema secuencial puede ser especificado usando las funciones de salida y de transición.

El biestable síncrono tipo D, posee dos entradas binarias de un bit: una de reloj CLK y una de datos D, así mismo posee una salida binaria de un bit: Q, verificando:

Q(t)	D(t)	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

2 Nociones de Xilinx ISE

2.1 Biestable D

El biestable tipo D (**fdc** en ISE) modelado en la librería, es un biestable disparado por flanco de subida (cuando el reloj cambia de 0 a 1, el biestable muestrea la entrada en ese instante y decide su transición, el resto de las modificaciones de la entrada no le afectan).

Este componente, aparte de las entradas síncronas referidas anteriormente, posee una entrada asíncrona (modifica el estado del flip-flop, independientemente del valor del reloj y de la entrada D): la entrada de clear, CLR. Con valor 0 no tiene efecto sobre el comportamiento del biestable, pero cuando toma el valor 1 modifica inmediatamente (asíncronamente) el estado del mismo, poniéndolo a 0. Se puede observar su comportamiento temporal en la Figura 32 .

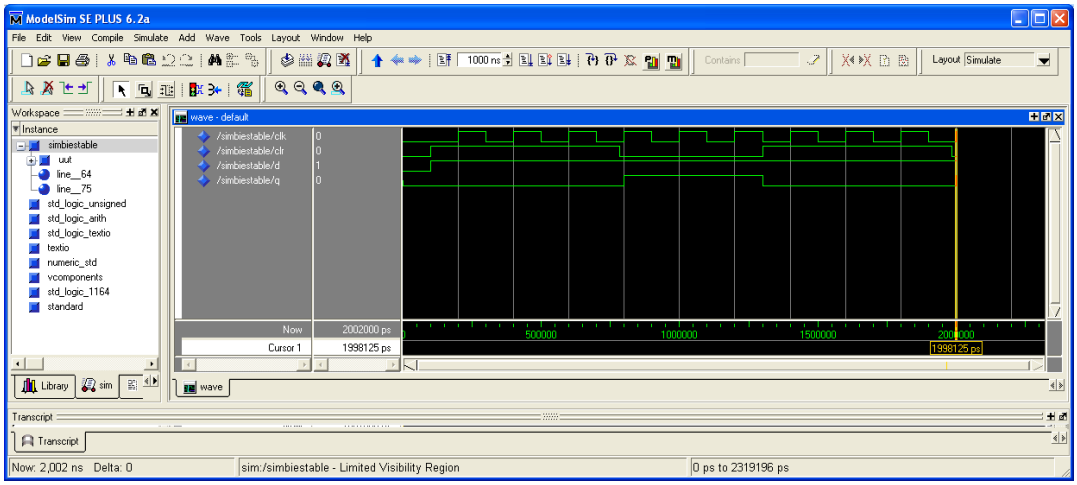


Figura 32: Simulación en Modelsim del biestable fdc

PRÁCTICA 4: Diseño de un reconocedor de secuencias.

OBJETIVO: Diseño de un circuito secuencial modelando su comportamiento como máquinas de Moore y Mealy.

ESPECIFICACIÓN:

Alto nivel:

Diseño de un sistema síncrono que posee una entrada, x, por la que recibe bits en serie, y una salida binaria, z, que tomará el valor lógico 1 cuando los últimos cuatro bits de entrada formen la secuencia 1101, y 0 en otro caso.

PASOS A SEGUIR:

1.- Obtener el diagrama de estados del sistema, suponiendo que es un reconocedor solapado, como máquina de Moore.

2.- Obtener el diagrama de estados del sistema, suponiendo que es un reconocedor solapado, como máquina de Mealy.

3.- Obtener las tablas de salida y transición de estados de los sistemas especificados en los puntos 2 y 3.

4.- Diseñar el sistema especificado en el punto 1 usando biestables tipo JK, que aparecen en la librería del ISE con el nombre **fjkc**.

5.- Captura y simulación del circuito diseñado en el punto 4.

6.- Diseñar el sistema especificado en el punto 2 usando biestables tipo D.

7.- Captura y simulación del circuito diseñado en el punto 6.

OTRA IMPLEMENTACIÓN OPCIONAL:

Estudiar la manera de implementar el sistema con un registro de desplazamiento y un módulo combinacional reconocedor.

Un registro de desplazamiento de ***n*** bits es un módulo que almacena los ***n*** últimos bits introducidos. Se implementa fácilmente encadenando biestables D.

El módulo combinacional reconocedor debe verificar que la configuración almacenada en el registro corresponda con la secuencia a reconocer.

IMPLEMENTACIÓN:

Nombres de los diferentes puertos:
- Línea de entrada: X.
- Línea de salida: S.
- Inicialización a 0 de los biestables (Q=0): CLR=1
- Señal de reloj: CLK

Nombre de los archivos:
- Paso 5: Pr41.SCH.
- Paso 7: Pr42.SCH.
- Implementación opcional: Pr43.SCH.

1 Nociones teóricas

1.1 Biestable JK

El biestable síncrono tipo JK, posee tres entradas binarias de un bit: una de reloj CK y dos de datos J y K. Así mismo, posee una salida binaria de un bit: Q, verificando:

Q(t)	J(t)	K(t)	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Su tabla de excitación es:

Q(t)	Q(t+1)	J(t)	K(t)
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

La entrada asíncrona de CLEAR funciona de manera análoga a la del biestable **fdc** de la práctica 3.