TUTORIAL DE Model*Sim*

1. INTRODUCCIÓN

El fin de este tutorial es proporcionar una guía rápida para el manejo básico de **Model*Sim***. El hilo conductor es la realización de un diseño, su compilación y su simulación. A través de este proceso se mostrara el ciclo de trabajo con **Model*Sim*** y algunas de sus características más importantes. Sin embargo, en última instancia, la referencia obligada son los manuales de la aplicación que suministra el fabricante, tanto en HTML como PDF.

Aquí se utilizará como lenguaje de descripción de *hardware* únicamente VHDL si bien **Model*Sim*** permite la realización de diseños empleando VHDL, *Verilog* o ambos, mezclando, en un mismo diseño, bloques realizados en ambos lenguajes.

La herramienta puede manejarse también en línea de comandos. Este modo no será utilizado con mucha frecuencia en este tutorial ya que resulta menos intuitivo que el modo gráfico pero es mucho más potente y eficaz para manejar grandes diseños.

2. DESCRIPCIÓN DEL DISEÑO

Como ejemplo de diseño vamos a utilizar un sumador de 8 bits con propagación de acarreo. El modelo se describirá de forma jerárquica partiendo de las puertas lógicas que componen un sumador completo de 1 bit. A partir del sumador elemental construiremos el sumador RCA de 8 bits conectando los módulos para conseguir la adecuada propagación del acarreo. La figura siguiente ilustra el esquema *hardware*.

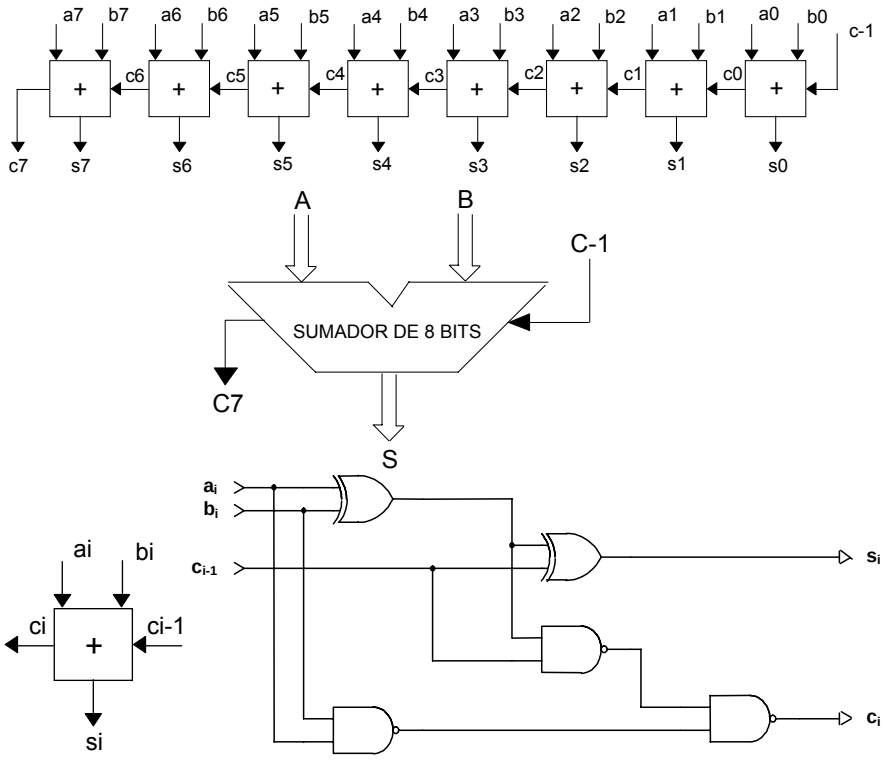


Figura 1. Esquema del sumador RCA de 8 bits.

Los modelos VHDL empleados son los siguientes:

```
ENTITY nand2 IS
	GENERIC (retardo: TIME := 2 ns);
	PORT (e1, e2: IN BIT; sal: OUT BIT);
END nand2;

ARCHITECTURE comportamiento OF nand2 IS
BEGIN
	PROCESS (e1, e2)
	BEGIN
		sal <= e1 NAND e2 AFTER retardo;
	END PROCESS;
END comportamiento;
```

Modelo 1. Puerta NAND de 2 entradas.

```
ENTITY xor2 IS
	GENERIC (retardo: TIME := 2 ns);
	PORT (e1, e2: IN BIT; sal: OUT BIT);
END xor2;

ARCHITECTURE comportamiento OF xor2 IS
BEGIN
	PROCESS (e1, e2)
	BEGIN
		sal <= e1 XOR e2 AFTER retardo;
	END PROCESS;
END comportamiento;
```

Modelo 2. Puerta XOR de 2 entradas.

```
ENTITY sum_elemental IS
	PORT (a, b, carry_in: IN BIT; suma, carry_out: OUT BIT);
END sum_elemental;

ARCHITECTURE estructural OF sum_elemental IS
--declaración de componentes
	COMPONENT xor2
		PORT (e1, e2: IN BIT; sal: OUT BIT);
	END COMPONENT;
	COMPONENT nand2
		PORT (e1, e2: IN BIT; sal: OUT BIT);
	END COMPONENT;
--declaración de señales
	SIGNAL s1, s2, s3: BIT;
--indicamos dónde se encuentra la arquitectura de los componentes
	FOR puerta1, puerta2: xor2 USE ENTITY WORK.xor2(comportamiento);
	FOR OTHERS: nand2 USE ENTITY WORK.nand2(comportamiento);
--realizamos las conexiones
BEGIN
	puerta1: xor2 PORT MAP (a,b,s1);
	puerta2: xor2 PORT MAP (s1,carry_in,suma);
	puerta3: nand2 PORT MAP (a, b, s2);
	puerta4: nand2 PORT MAP (s1, carry_in, s3);
	puerta5: nand2 PORT MAP (s2, s3, carry_out);
END estructural;
```

Modelo 3. Sumador completo de 1 bit.

```
ENTITY sumador_8bits IS
	PORT (vector_a, vector_b: IN BIT_VECTOR (7 DOWNTO 0); carry_in: IN BIT;
	vector_suma: OUT BIT_VECTOR (7 DOWNTO 0); carry_out: OUT BIT);
END sumador_8bits;

ARCHITECTURE estructural OF sumador_8bits IS
--declaración de componentes
	COMPONENT sum_elemental
		PORT (a, b, carry_in: IN BIT; suma, carry_out: OUT BIT);
	END COMPONENT;
--declaración de señales
	SIGNAL vector_carry: BIT_VECTOR (7 DOWNTO 0):="00000000";
```

```
--indicamos donde se encuentra la arquitectura de los componentes
       FOR ALL: sum_elemental USE ENTITY WORK.sum_elemental(estructural);
--realizamos las conexiones
BEGIN
       celda1: sum_elemental PORT MAP (vector_a(0), vector_b(0),
              carry_in, vector_suma(0), vector_carry(0));
       celda2: sum_elemental PORT MAP (vector_a(1), vector_b(1),
              vector_carry(0), vector_suma(1), vector_carry(1));
       celda3: sum_elemental PORT MAP (vector_a(2), vector_b(2),
              vector_carry(1), vector_suma(2), vector_carry(2));
       celda4: sum_elemental PORT MAP (vector_a(3), vector_b(3),
              vector_carry(2), vector_suma(3), vector_carry(3));
       celda5: sum_elemental PORT MAP (vector_a(4), vector_b(4),
              vector_carry(3), vector_suma(4), vector_carry(4));
       celda6: sum_elemental PORT MAP (vector_a(5), vector_b(5),
              vector_carry(4), vector_suma(5), vector_carry(5));
       celda7: sum_elemental PORT MAP (vector_a(6), vector_b(6),
              vector_carry(5), vector_suma(6), vector_carry(6));
       celda8: sum_elemental PORT MAP (vector_a(7), vector_b(7),
              vector_carry(6), vector_suma(7), vector_carry(7));
       carry_out <= vector_carry(7);
END estructural;
```

Modelo 4. Sumador RCA de 8 bits.

3. ARRANQUE DE LA APLICACIÓN Model*Sim* XE

La versión **Model*Sim* XE** (*Xilinx Edition*) es una edición especial distribuida con el *software* de Xilinx para la implementación de FPGAs. La versión libre (**Model*Sim* XE II/Starter**) utiliza una licencia facilitada por Xilinx que ofrece una acceso limitado a las características de la herramienta.

Esta aplicación puede funcionar sobre sistemas operativos Windows 98/2000/NT 4.0/XP y Unix HP-UX 11.0, Solaris 7/8 y Linux. Este tutorial esta realizada para la versión **Model*Sim* XE II/Starter 5.7c** ejecutándose sobre Windows XP.

El arranque de la aplicación se realiza a través del menú inicio: ***Menú Inicio*** ➔ ***Programas*** ➔ ***ModelSim XE/Starter***. Después de la ventana de bienvenida aparece la ventana principal del entorno tal y como se muestra en la figura siguiente.

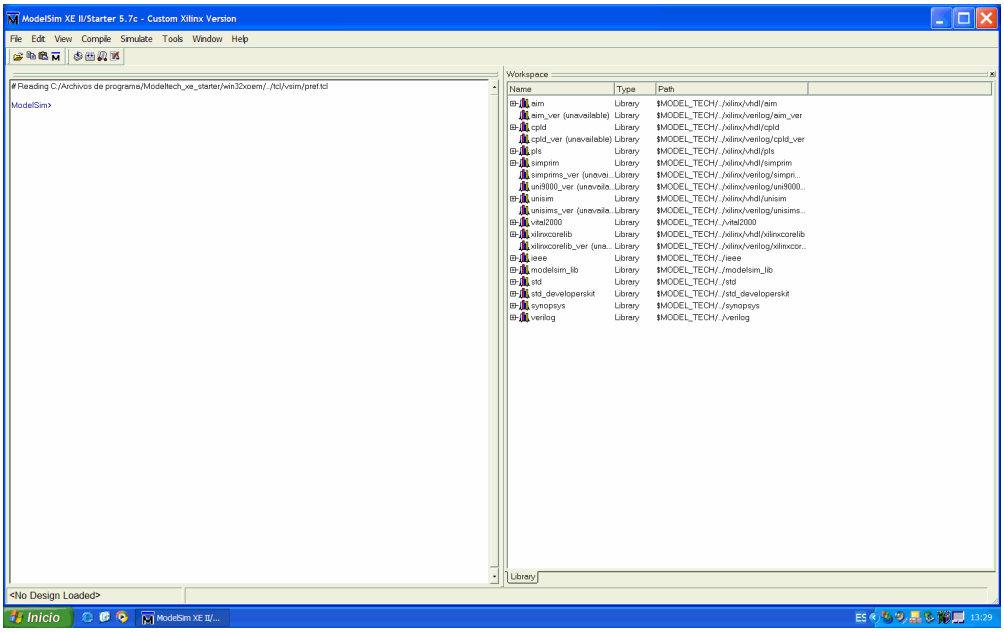


Figura 2. Aspecto inicial de **Model*Sim***.

En la ventana principal se distinguen dos partes:

- A la derecha se encuentra el ***Workspace*** (Área de trabajo). Al iniciar la aplicación únicamente se dispone de la pestaña ***Library*** (abajo) pero a medida que avanza el diseño aparecen otras nuevas (***Project***, ***Simulation***). En la pestaña ***Library***, por defecto, aparecen los nombres lógicos de las librerías de recursos disponibles.
- A la izquierda se encuentra una ventana de texto —denominada ***Transcript***— en la que se presentan los mensajes producidos por la ejecución de los comandos. En esta parte se permite teclear comandos en línea y se mantiene un histórico de todos los comandos ejecutados. Usando las teclas ↑ y ↓ se puede navegar por la lista de comandos emitidos hasta el momento para ejecutarlos de nuevo. El contenido de la ventana de trascripción (***Transcript***) puede ser guardado en un fichero para ser utilizado posteriormente como una macro.

Para cerrar la aplicación hay que seleccionar la opción ***File ➔ Quit*** y confirmar en el cuadro de diálogo correspondiente.

4. CREACION DE UN PROYECTO

En la herramienta **ModelSim**, un proyecto es un conjunto de unidades de diseño HDL —descripciones funcionales, estructurales o *testbenches*—. Un proyecto debe contener, al menos, un directorio raíz, una librería de trabajo (***work***) y un fichero de configuración (***<nombre-proyecto>.mpf***).

La librería de trabajo ***work*** es una estructura de datos que tiene aspecto de carpeta del Sistema de Ficheros pero que es propia de **Model*Sim***. Es el lugar donde se salvan los resultados de la compilación de los modelos HDL. La librería de trabajo se crea automáticamente al crear el proyecto. Si se desea compilar modelos HDL sin crear un proyecto hay que tener la precaución de crear ***work*** en el lugar adecuado mediante las opciones de menú de la herramienta (***File ➔ New ➔ Library…***).

El fichero ***<nombre-proyecto>.mpf*** es un archivo de texto que contiene variables de configuración del entorno y el valor asignado a las mismas. Por ejemplo, el orden de compilación de los ficheros HDL incluidos en el proyecto, la asignación de nombres lógicos a los ficheros de librerías de recursos empleadas en el proyecto, etc. Este fichero se crea automáticamente al crear el proyecto y su contenido puede variarse interaccionando con los menús del entorno o bien editándolo manualmente.

Creación de nuestro proyecto:

1. Iniciar la aplicación **Model*Sim* XE**.
2. Seleccionar la opción ***File ➔ New ➔ Project***. Aparecerá la ventana de dialogo de la figura siguiente. En dicha ventana se establece el nombre del proyecto, el directorio en el que desea ubicarse y el nombre de la librería por defecto.
3. En la casilla ***Project Name*** utilizar **SUMADOR** como nombre del proyecto.
4. En la casilla ***Project Location*** seleccionar el directorio en el que va a ubicar los ficheros del proyecto. Se puede ayudar del botón ***Browse...*** para moverse por el Sistema de Ficheros. Si el directorio no existe, la herramienta le pedirá confirmación para su creación. En este directorio se creará el fichero ***Sumador.mpf***. Con un editor de texto cualquiera podemos leerlo y editarlo. No es obligatorio que el nombre del directorio coincida con el nombre del proyecto; si bien, es una buena práctica hacerlos coincidir y también con el nombre de la unidad de diseño VHDL de mayor nivel dentro de la jerarquía del modelo.
5. En la casilla ***Default Library Name*** se escribe el nombre de la librería de trabajo. Por defecto aparece ***work*** y, en general, debe emplearse este nombre ya que es el asignado automáticamente por la aplicación al crear el proyecto. Esta librería se crea debajo del directorio indicado en el cuadro ***Project Location***.
6. Hacer clic en el botón OK para crear el proyecto.

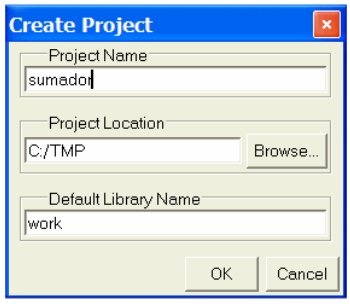


Figura 3. Ventana de dialogo pare la creación de un proyecto.

La creación del proyecto tiene las siguientes consecuencias:

1. Se crea el fichero ***Sumador.mpf*** en el directorio del proyecto.
2. Se crea la libraría ***work*** que, de momento, contiene únicamente el fichero de texto ***_info***. El aspecto de la librería ***work*** es igual al de una carpeta del Sistema de Ficheros y como tal se puede explorar.
3. Aparece una nueva pestaña ***Project***, inicialmente vacía, junto a la de ***Library*** ya existente en el área de trabajo (***workspace***).
4. Se muestra en la barra de estado de la ventana principal de **Model*Sim*** situada en la parte inferior de la misma, el nombre del proyecto activo.

Una vez creado el proyecto aparece una ventana de diálogo que nos permite seleccionar un elemento entre 4 posibles tipos para añadir al proyecto. Los elementos que pueden incorporarse a un proyecto son los siguientes:

- Un fichero nuevo: el comando ***Create New File*** permite crear un nuevo fichero de diseño VHDL.
- Un fichero existente: el comando ***Add Existing File*** permite incluir un fichero ya existente en el proyecto actual copiándolo al directorio del proyecto o referenciándolo desde su ubicación actual.
- Un directorio virtual: cuando el diseño es complejo y consta de muchos ficheros, puede resultar interesante emplear el comando ***Create New Folder*** para organizarlos en directorios y subdirectorios. Estos directorios tienen, sobre la pestaña ***Project*** del área de trabajo (***workspace***), un aspecto similar al de los directorios del sistema operativo pero no se crean realmente en el disco, sino que son internos al proyecto.
- Una configuración para simulación: El comando ***Create Simulation*** permite crear configuraciones para simulación, es decir, crear una asociación entre la unidad de diseño y las opciones con las que desea simularse.

Si cerramos el cuadro de diálogo sin seleccionar ninguna de las posibilidades descritas más arriba podemos utilizar posteriormente el menú contextual del ***workspace*** para añadir alguno de estos cuatro elementos. La figura siguiente ilustra este aspecto.

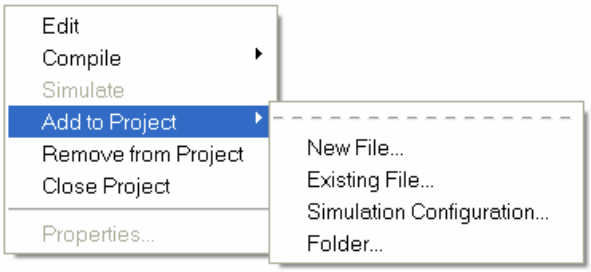


Figura 4. Menú contextual sobre ***workspace*** para añadir elementos al proyecto.

5. AÑADIR ELEMENTOS AL PROYECTO

Vamos a añadir al proyecto un nuevo fichero describiendo un circuito con VHDL. Para ello utilizaremos el menú contextual del ***workspace***. Nos aparece el cuadro de diálogo de la figura siguiente. Sobre él realizamos lo siguiente:

1. En la caja de texto ***File Name*** escribir **NAND2**.
2. Seleccionar VHDL en el cuadro combinado ***Add file as type***.
3. Seleccionar *Top Level* —puesto que no se han creado directorios virtuales en este proyecto— en ***Folder***.
4. Pulsar el botón OK.

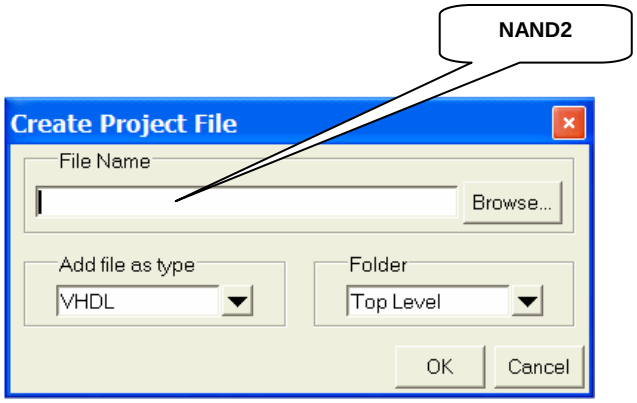


Figura 5. Cuadro de diálogo para crear ficheros en un proyecto.

Observar como, en la pestaña ***Project*** del ***workspace***, aparece el icono del fichero bajo la columna *Name* (ver figura siguiente). La información que aparece en las otras columnas es la siguiente:

- ***Status***: el símbolo **?** indica que el fichero aún no ha sido compilado; el símbolo **X** indica que el fichero se ha compilado y contiene errores; finalmente, el símbolo √ indica que el fichero ha sido compilado con éxito.
- ***Type***: indica el tipo de fichero (VHDL, *Verilog*, *Folder*, *Simulation*).
- ***Order***: indica el número de orden que ocupa el fichero en la secuencia de compilación cuando hay varios ficheros fuente.
- ***Modified***: muestra la fecha y hora en que el fichero fue modificado por última vez.

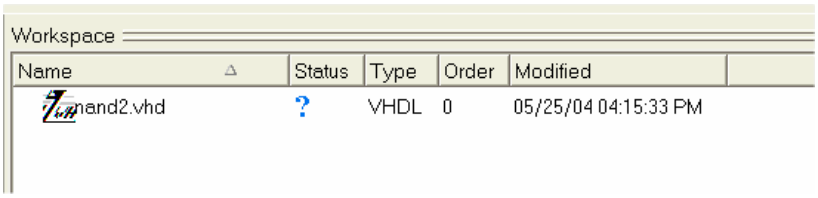


Figura 6. Detalles de los ficheros del proyecto en el ***workspace***.

Haciendo doble clic sobre un fichero del ***workspace*** se abre automáticamente una ventana del editor de texto del entorno con el fichero. La figura siguiente muestra dicho editor con el modelo VHDL de la puerta NAND de 2 entradas.

El menú contextual asociado a los ficheros permite ver sus propiedades (especialmente las relativas a la compilación) así como realizar algunas tareas tales como compilar, añadir al proyecto, eliminar, etc.

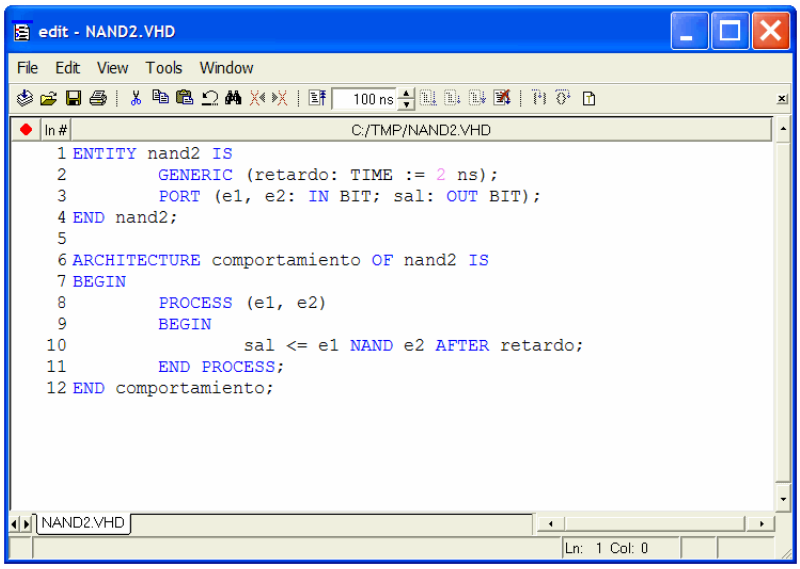


Figura 7. Ventana del editor de texto.

6. COMPILACIÓN DE LOS FICHEROS

Una vez editado el modelo de un circuito lógico descrito con algún lenguaje HDL es necesario compilarlo para poder simular su comportamiento. La compilación de nuestro modelo de NAND puede efectuarse de los siguientes modos:

- Pulsando el botón  situado en la barra de herramientas del editor.
- Pulsando el mismo botón situado en la barra de herramientas de la ventana principal.
- Mediante el menú contextual del ***workspace***.
- Tecleando **vcom** NAND2.vhd en la ventana ***Transcript***.

Si la compilación da errores estos aparecen en color rojo en la ventana de trascripción. Para ver una descripción más detallada basta con hacer doble clic en el mensaje de error. Si el resultado de la compilación es correcto aparece un mensaje en color verde indicando que ha sido un éxito.

Cuando compilamos con éxito un modelo se salvan diferentes ficheros en la librería de trabajo ***work***. Si conmutamos a la pestaña ***Library*** del ***workspace*** y pulsamos en el signo **+** de ***work*** veremos como se despliega la carpeta mostrando su contenido (NAND2). El menú contextual asociado nos proporciona acceso a las propiedades, a la posibilidad de recompilación y a la simulación.

A continuación vamos a añadir al proyecto otros tres ficheros: el modelo de la puerta XOR de 2 entradas (XOR2.vhd), el modelo estructural del sumador completo de 1 bit (SUM_ELEM.vhd) y el modelo estructural del sumador con propagación de acarreo (RCA) de 8 bits (SUMADOR8.vhd). Los modelos estructurales corresponden a descripciones de un nivel superior en la jerarquía del diseño. Son descripciones jerárquicas las que se basan en bloques compìlados con antelación. La figura siguiente muestra la ventana de diálogo encargada de añadir los nuevos ficheros. Es interesante que los ficheros de los nuevos modelos se copien en el directorio del proyecto para que las modificaciones que posteriormente puedan sufrir no afecten al funcionamiento de otros proyectos. Esto se consigue seleccionando la opción ***Copy to Project directory*** del cuadro de diálogo.

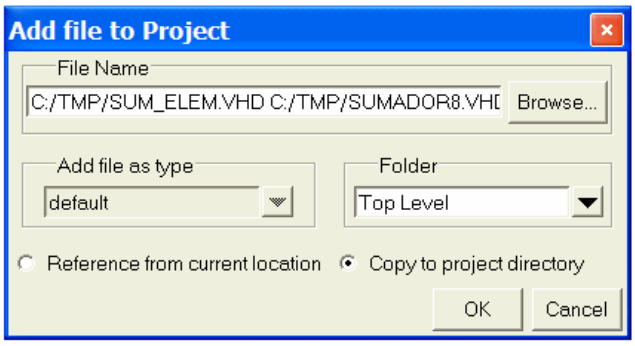


Figura 8. Cuadro de diálogo para añadir ficheros existentes al proyecto.

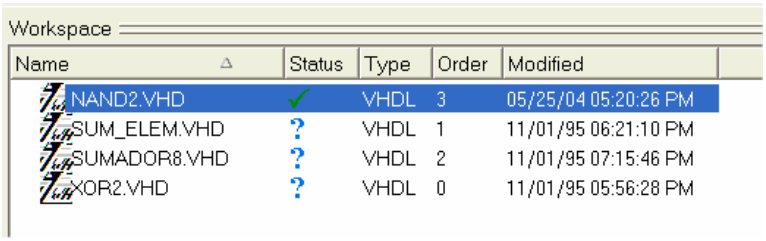


Figura 9. Situación en la que queda el ***workspace*** una vez añadidos el resto de ficheros del sumador RCA de 8 bits al proyecto.

Ahora es necesario compilar el resto de modelos. En un diseño jerárquico —como es nuestro caso— antes de compilar un diseño deben estar compilados los precedentes en la jerarquía ya que, en caso contrario, se producen errores. En el ejemplo que estamos siguiendo el orden sería: NAND, XOR, sumador elemental (sumador completo de 1 bit) y, finalmente, sumador RCA de 8 bits.

Para compilar el nuevo fichero se puede:

- Seleccionar el fichero —haciendo clic con el ratón sobre el mismo en área de trabajo— y seleccionando el comando ***Compile ➔ Compile Selected*** del menú contextual. En este caso se compilara únicamente el fichero seleccionado.
- Activar el comando ***Compile ➔ Compile All*** del mismo menú. En este caso se compilaran todos los ficheros.
- Seleccionar el comando ***Compile ➔ Compile Out-of-Date*** del menú contextual. Con este comando se compilan únicamente los ficheros que no hayan sido compilados con éxito anteriormente, es decir, todos los que no presenten el icono √ en la columna *Status*. Esta opción es especialmente útil cuando el diseño se compone de muchos ficheros y se efectúan modificaciones en algunos de altos.

Si se compila manualmente uno a uno cada fichero, es responsabilidad del usuario hacerlo en el orden adecuado. Si se manda compilar todo, ya sea indiscriminadamente ya sea seleccionando solamente lo no compilado con antelación, el orden que tome la aplicación no tiene por qué coincidir con el jerárquico y se pueden producir errores en algunos ficheros. En ese caso puede ser necesario modificar el orden de compilación con el comando ***Compile ➔ Compile Order...*** del menú contextual asociado al ***workspace***. Otra solución, menos elegante, puede consistir en activar la compilación indiscriminada reiteradamente hasta que no quede ningún modelo por compilar.

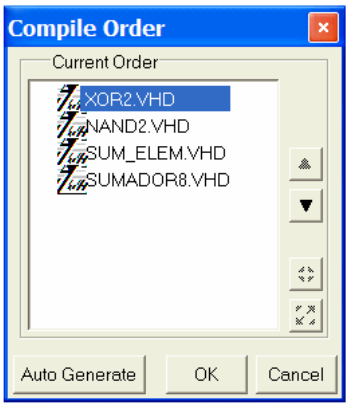


Figura 10. Ventana para configurar el orden de compilación.

Una vez compilados los ficheros sin errores, podemos activar la pestaña ***Library*** del área de trabajo y pulsar el signo **+** que aparece a la izquierda de la librería ***work*** para desplegar su contenido. Si todo es correcto debemos ver las carpetas correspondientes a los 4 ficheros compilados colgando de ***work***.

Una vez que los ficheros están compilados, puede efectuarse la simulación del diseño.

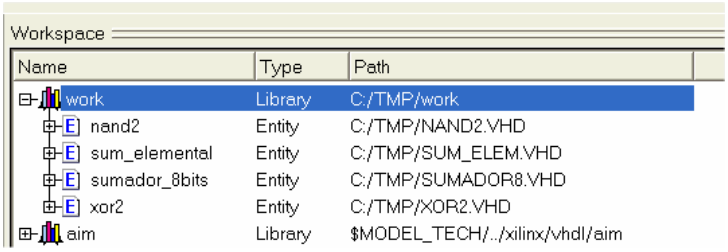


Figura 11. Aspecto de ***work*** con los ficheros compilados.

7. SIMULACIÓN DEL BLOQUE NAND2.VHD

Como ejemplo vamos a pasar a describir cómo se realiza la simulación de uno de los modelos jerárquicamente más bajos, el de la puerta NAND de 2 entradas. Para ello podemos lanzar la simulación utilizando el menú contextual asociado con el área de trabajo en su pestaña ***Library*** (recomendado), la barra de menú de la aplicación principal o el icono correspondiente de la barra de herramientas de la aplicación principal.

Una vez iniciada la simulación sin problemas podemos apreciar los siguientes cambios:

- Aparecen dos pestañas nuevas en el ***workspace*** junto a las ya existentes. Una es ***sim***. Esta pestaña muestra la estructura del diseño además de las librerías de recursos utilizadas. La otra es ***Files*** que recoge información de los ficheros utilizados para el modelo bajo simulación y de las librerías asociadas al mismo.
- Se muestra el nombre de la unidad de diseño cargada y el tiempo de simulación —inicialmente a cero— en la barra de estado de la ventana principal de **ModelSim** situada en la parte inferior de la pantalla.
- El *prompt* de la ventana de trascripción pasa de ser ***ModelSim>*** a ser ***VSIM>***

La herramienta **ModelSim** permite la ejecución de la simulación de unidades de diseño HDL de forma interactiva, ejecutando la simulación paso a paso o de forma continua así como introducir puntos de ruptura en el código HDL que detengan la ejecución de la misma, tal como sucede con las herramientas de depuración de los lenguajes de programación de alto nivel.

Además del simulador propiamente dicho, la herramienta dispone de un **analizador de cobertura** que indica qué porcentaje de líneas de código HDL de los diferentes ficheros se han ejecutado en la simulación y un ***profiler*** que indica a qué parte del código se dedica mas tiempo de simulación.

Para interaccionar con el simulador y visualizar los resultados de la simulación, la herramienta dispone de un conjunto de ventanas que, de forma resumida, presentan la siguiente funcionalidad:

- Ventana ***dataflow***: permite visualizar, de modo grafico la conectividad entre los elementos (procesos, sentencias de asignación concurrente, etc.) del modelo HDL y rastrear eventos.
- Ventana ***list***: muestra, en modo texto, los valores de las señales y variables en un formato tabular.
- Ventana ***process***: muestra la lista de procesos junto con información sobre su estado.
- Ventana ***signal***: muestra la lista de señales de la unidad de diseñó seleccionada.
- Ventana ***source***: ventana similar a la de edición, permite visualizar el código fuente HDL de los ficheros del diseño.
- Ventana ***structure***: muestra de forma grafica la estructura de la jerarquía del diseño de modo similar al mostrado en el ***workspace***. De hecho, esta ventana no resulta muy útil porque las funciones que permite se pueden realizar en la propia área de trabajo.
- Ventana ***variables***: permite visualizar el valor de las variables, constantes y genéricos.
- Ventana ***wave***: permite visualizar la forma de onda de las señales y variables (diagramas de tiempos o cronogramas).

La figura siguiente muestra un aspecto de la aplicación con todas las ventanas abiertas dispuestas para la simulación de la NAND de 2 entradas (antes de comenzar la simulación).

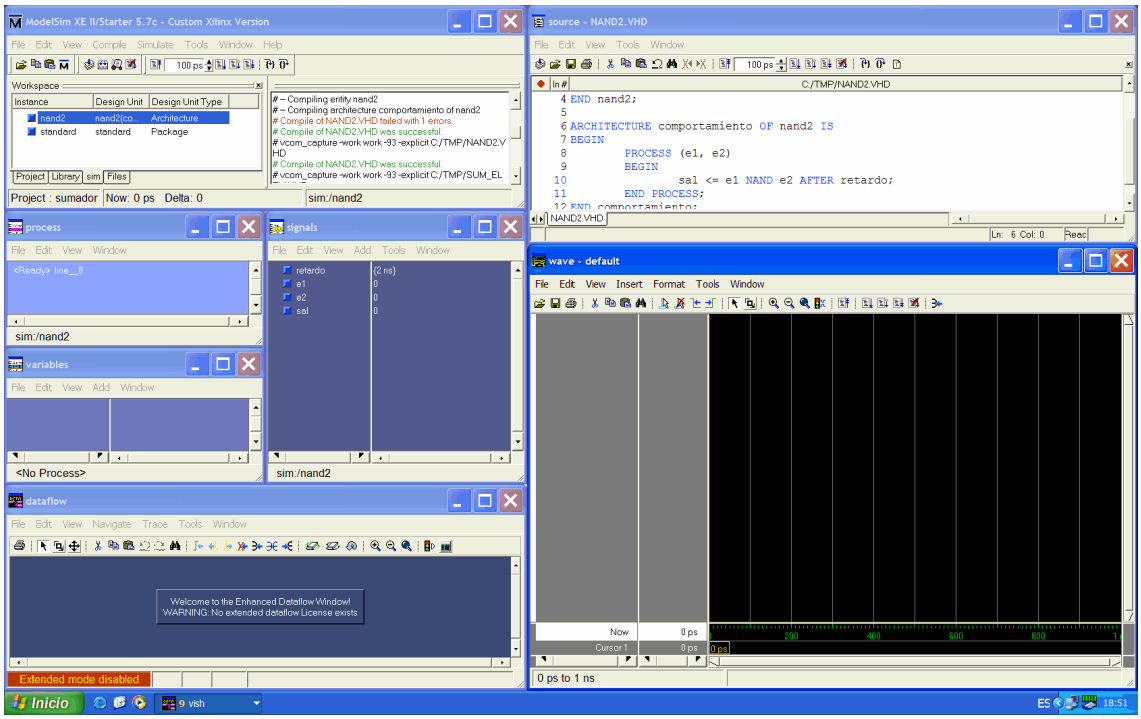


Figura 12. Aspecto de la aplicación con todas las ventanas abiertas esperando el comienzo de una simulación de la puerta NAND de 2 entradas.

La utilidad detallada de las ventanas se puede consultar en la ayuda proporcionada por la propia aplicación. Aquí tan sólo se sugieren algunas actividades con el fin de familiarizarse con el manejo de la misma. Por ejemplo, realizar las siguientes operaciones:

1. Activar el comando ***View*** ➔ ***Wave*** del menú de la ventana principal de **ModelSim** para arrancar la ventana *wave*. Observar que aparece la ventana vacía. El siguiente paso es incluir en ella las señales cuyo aspecto se desea visualizar durante la simulación. Para ello puede procederse de las siguientes formas:
   a. Seleccionar, en la estructura del diseño que aparece en el ***workspace***, la unidad de diseño cuyas señales se desea incluir (en este caso únicamente nand2) y pulsar el botón derecho del ratón para seleccionar el comando ***Add*** ➔ ***Add to Wave***. De este modo, se incluyen todas las señales de la unidad de diseño seleccionada en la ventana de onda (***wave***).
   b. Activar la ventana ***signal*** —seleccionando el comando ***View*** ➔ ***Signals*** del menú de la ventana principal de **ModelSim**—. En esta ventana se visualizan todas las señales de la unidad de diseño seleccionada en la estructura que aparece en el área de trabajo. Sobre ella se puede escoger las señales deseadas y después, pulsando el botón derecho, activar el comando ***Add to Wave*** ➔ ***Selected Signals*** del menú contextual para incluirlas. Observar que desde el comando ***Add to Wave*** también se pueden incluir todas las señales del bloque sin necesidad de seleccionarlas previamente —comando ***Add to Wave*** ➔ ***Signals in Region***—, o bien, todas las señales de todos los bloques del diseño —comando ***Add to Wave*** ➔ ***Signals in Design***—.
   c. Tecleando en la ventana ***transcript*** el comando ***add wave*** con las opciones deseadas. A continuación se muestran algunos ejemplos:
      - Para incluir todas las señales de nand2:
        ```
        add wave sim:/nand2/*
        ```
      - Para incluir la serial e1 de nand2:
        ```
        add wave sim:/nand2/e1
        ```
      - Para incluir todas las señales del diseño:
        ```
        add wave -r /*
        ```
2. Seleccionar el comando ***View*** ➔ ***Signals*** del menú de la ventana principal para activar la ventana ***Signal***.
3. Pulsar sobre ella el botón derecho del ratón y activar el comando ***Add to Wave*** ➔ ***Signals in Design*** del menú contextual pare incluir todas las señales del diseño. La ventana ***wave*** tendrá el aspecto mostrado en la figura siguiente.

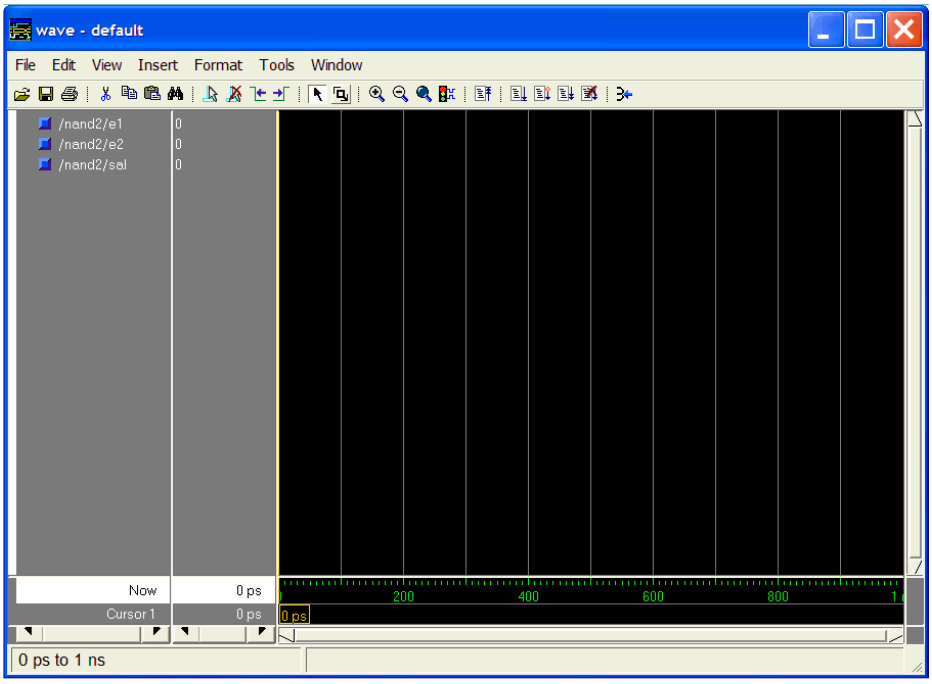


Figura 13. Aspecto de la ventana ***wave*** para la nand2.

4. Ajustar el tamaño de la ventana para poder visualizar el nombre completo de las señales que aparecen en su columna izquierda. Como puede apreciar las señales se referencian con su nombre jerárquico completo. La "profundidad" del nombre que se presenta puede configurarse para una mejor visualización.
5. Activar el comando ***Tools ➔ Window Preferences...*** de la barra de menús de la ventana ***wave*** y escribir 1 en el casilla ***Display Signal Path*** del cuadro de dialogo que aparece (ver figura). Con esto limitamos a 1 la “profundidad” de la ruta de las señales.

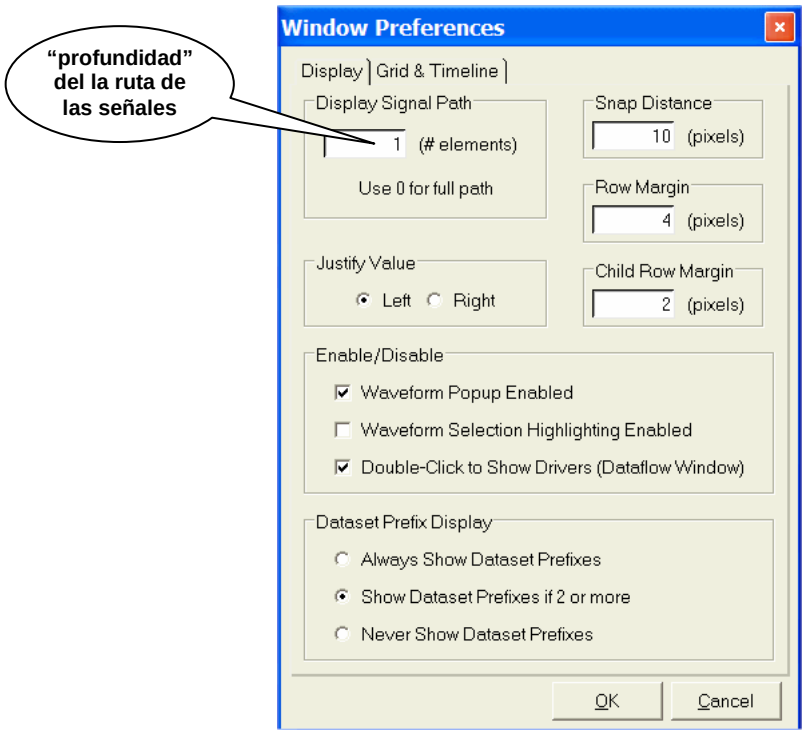


Figura 14. Aspecto de la ventana de preferencias de ***wave***.

6. Pulsar el botón OK y observar el nuevo aspecto de la ventana ***wave***. Ahora se presenta tan sólo el nombre de la señal.

Es posible ajustar los tamaños de cada una de las secciones de la ventana de onda. También es posible cambiar el orden en el que aparecen las señales: basta con seleccionar y arrastrar a la nueva posición. Se pueden eliminar señales seleccionándolas y pulsando la tecla *Suprimir* o bien con el comando adecuado del menú contextual.

El menú contextual de cada señal ofrece la posibilidad de acceder a la declaración de la señal (marcada en la ventana de código fuente), a expresar la señal en función de una determinada base (hexadecimal, decimal, etc.), a modificar el formato y a ver las propiedades de la misma, entre otras opciones disponibles. Entre las propiedades de la señal, además de la base y del formato, encontramos el color tanto de la línea como del texto asociado.

Las señales de tipo bus (compuestas de varias líneas) se presentan bajo el nombre común pero disponen de un símbolo **+** a la izquierda que actúa como un botón. Cuando se pulsa el botón **+** la señal se expande en todas sus componentes que se representan como líneas sueltas dentro de la ventana de onda.

8. EJECUCIÓN DE LA SIMULACIÓN

Una vez configurado el aspecto deseado en la ventana ***wave*** se puede iniciar la simulación. La ejecución de la simulación puede controlarse de forma cómoda con los botones que se encuentran tanto en la ventana principal como en las ventanas ***wave*** y ***source***.

- ***Run***: ejecuta la simulación durante el tiempo indicado en el cuadro 100 ps de la ventana principal aunque la simulación puede detenerse antes si se han introducido puntos de ruptura o por la ejecución de alguna sentencia del *testbench*. El tiempo seleccionado por defecto es de 100ps.
- ***Continue Run***: continúa la simulación en curso hasta agotar el tiempo de simulación por defecto.
- ***Run-All***: ejecuta la simulación durante tiempo indefinido aunque puede detenerse si se han incluido puntos de ruptura o si se interrumpe con el botón ***break***. En la barra de estado de la ventana principal se presenta el tiempo de simulación.
- ***Break***: detiene la simulación en curso.
- ***Restart***: reinicia la simulación. Carga de nuevo el modelo binario, sitúa el tiempo de simulación en cero y, opcionalmente, puede mantener el formato dado a algunas ventanas, los puntos de ruptura, etc.
- ***Step***: ejecuta la simulación hasta la siguiente sentencia.
- ***Step Over***: ejecuta funciones y procedimientos en un solo paso.

Vamos a realizar algunas de estas operaciones sobre el modelo de la NAND2 que estamos simulando:

1. Pulsar el botón (***run***) en la ventana principal o en la ventana ***wave*** para comenzar la ejecución de la simulación. Se simulara el funcionamiento del modelo durante 100ps, que es el tiempo de simulación definido por defecto, y la ventana ***wave*** presentara un aspecto similar al mostrado en la figura siguiente. Se puede modificar el aspecto de la ventana actuando sobre los botones de *zoom* . Para modificar el tiempo de simulación por defecto basta escribir el nuevo valor sobre el cuadro 100 ps.

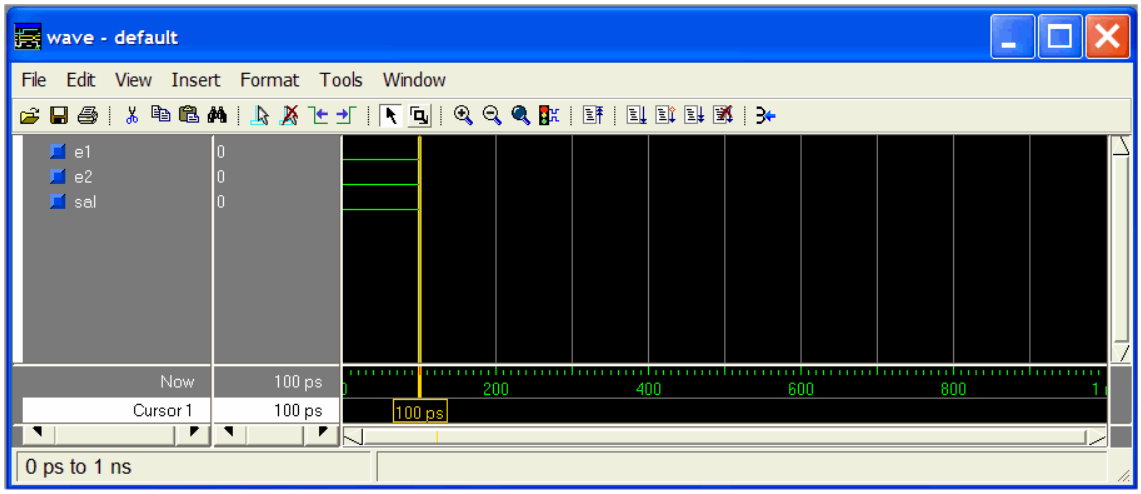
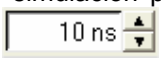


Figura 15. Aspecto de la ventana ***wave*** cuando se han simulado 100ps.

NOTA: En la figura anterior se puede observar que los valores de las señales no corresponden a la tabla de verdad de la función lógica NAND. Esto se debe a que el retardo asociado a la sentencia de asignación de señal de nuestro modelo es de 2ns mientras que el tiempo de simulación asociado al comando ***run*** por defecto es de 100ps. Hasta que la simulación no haya evaluado los primeros 2ns no veremos el valor correcto en la señal de salida del modelo. Para solucionar esto es conveniente dar un tiempo de simulación del mismo orden que los retardos del circuito. Proponemos, pues, tiempos de simulación para comando ***run*** de 10ns. Introduciremos en el cuadro lo siguiente: 

Si ejecutamos la simulación bajo estas condiciones tendremos el cronograma siguiente:

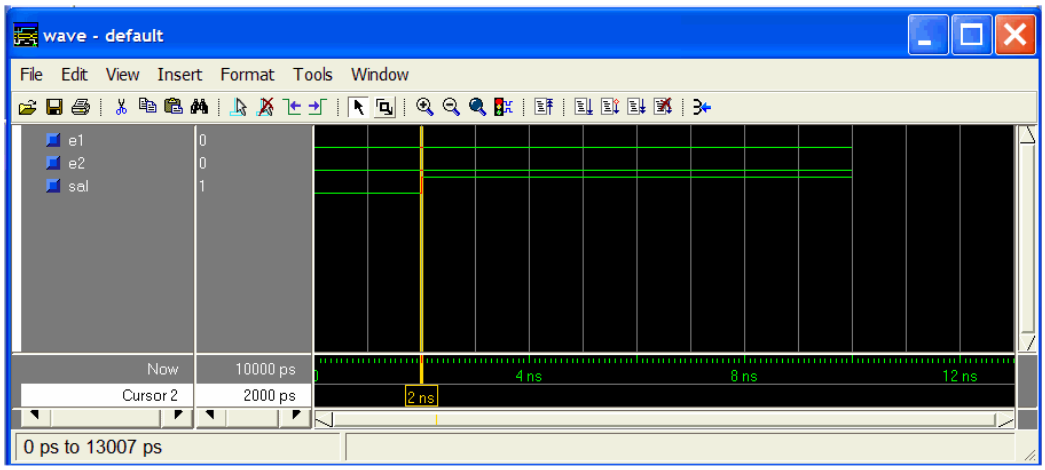


Figura 16. Aspecto de la ventana ***wave*** con un tiempo de simulación por defecto de 10ns.

Ahora si que se cumple la tabla de verdad de la función lógica y se comprueba, gracias al cursor, que la asignación de tiempos se ha producido en el instante de simulación igual a 2ns.

2. Introducir un punto de ruptura en la sentencia de asignación de señal siguiente:

```
sal <= e1 NAND e2 AFTER retardo;
```

El punto de ruptura se puede introducir desde la ventana ***source*** o desde el editor de los ficheros, marcando con el ratón a la izquierda del número de línea del código. Aparecerá un rombo de color rojo que indica la inserción de un punto de ruptura en la línea. La figura siguiente ilustra lo explicado.

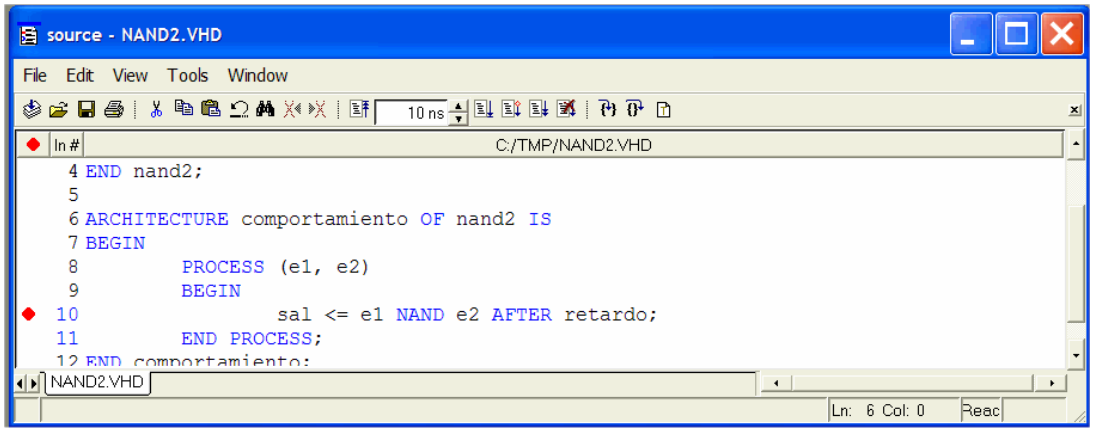


Figura 17. Aspecto de la ventana ***source*** con el punto de ruptura introducido.

3. Pulsar ahora el icono (***run***) en la ventana principal, en ***wave*** o en ***source***. La simulación se debería ejecutar hasta que se alcance el punto de ruptura. Sin embargo, observamos que avanza el tiempo de simulación, se construye el cronograma para 10ns más pero no se alcanza el punto de ruptura. Esto se debe a que no ha cambiado ninguna señal de la *lista de sensibilización* y, por tanto, no se ha ejecutado el proceso asociado con ella en el que tenemos la sentencia de asignación de señal de dicho punto de ruptura. Para que adopten diferentes valores las señales del modelo hay que **forzar** el cambio desde la ventana ***signals***. La figura siguiente muestra el cronograma antes de forzar valores. Para ver todo el cronograma hay que usar el comando ***View ➔ Zoom ➔ Zoom Full*** o el icono correspondiente ( ).

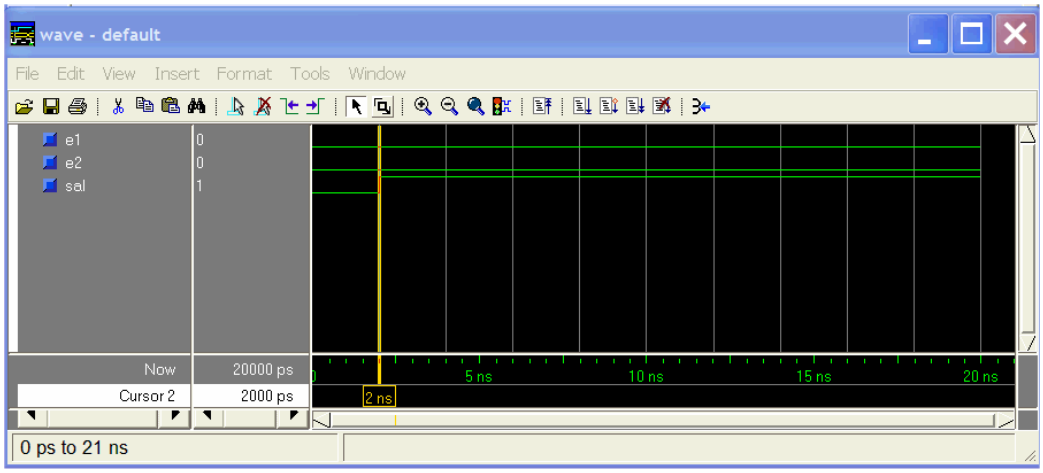


Figura 18. Aspecto de la ventana ***wave*** con 20ns de simulación.

4. Seleccionar la señal *e1* en la ventana ***signals***. En la barra de menú buscar el comando ***Edit ➔ Force…***. Asignar el valor 1 a esta señal. Como podemos ver en la figura siguiente, el cuadro de diálogo para forzar valores en las señales permite establecer 3 modos de asignación distintos así como una cierta temporalización.

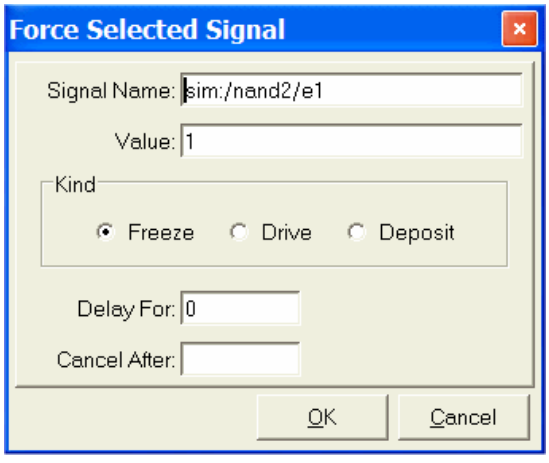


Figura 19. Cuadro de diálogo de la ventana ***signals*** para forzar valores en las señales del modelo bajo simulación.

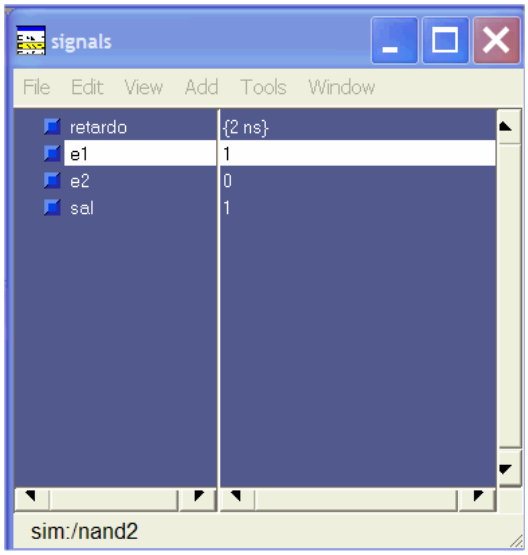


Figura 20. Ventana ***signals*** con los valores en curso.

5. Al pulsar ahora el icono  (***run***) la simulación se ejecuta hasta alcanzar el punto de ruptura. El cronograma de la ventana ***wave*** representa el valor 1 de la señal *e1* pero no avanza más el tiempo de simulación hasta que no se ejecute la sentencia de asignación de señal en la que nos hemos detenido. Observar cómo en la ventana ***source*** aparece una flecha azul indicando la siguiente línea a ejecutar.

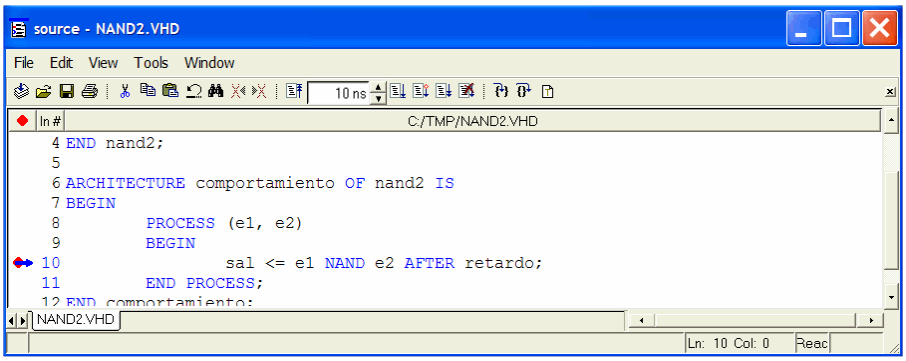


Figura 21. Ventana ***source*** con el punto de ruptura indicado por una flecha azul.

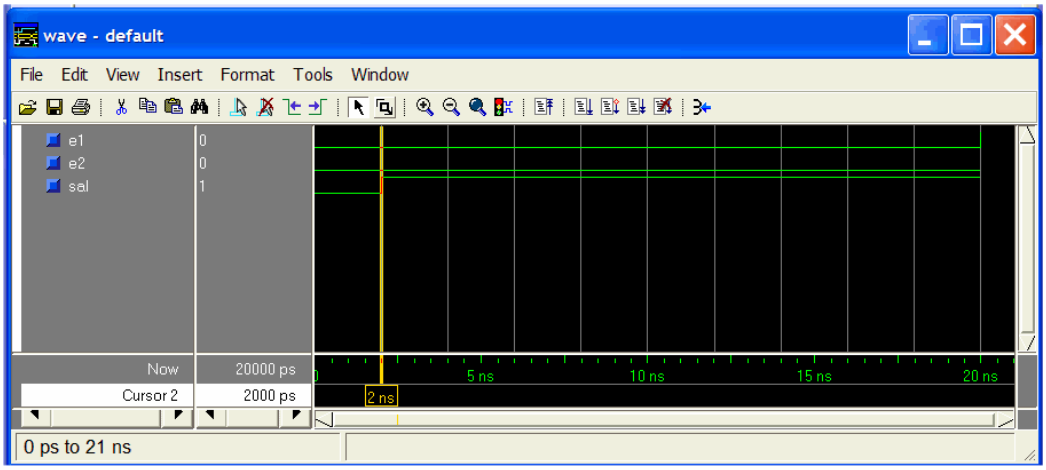


Figura 22. Aspecto de la ventana ***wave*** habiéndose detenido la simulación en el punto de ruptura indicado.

6. Pulsar el icono  (***continue run***). Ahora avanza el tiempo de simulación en la ventana ***wave***. La figura siguiente ilustra como queda.

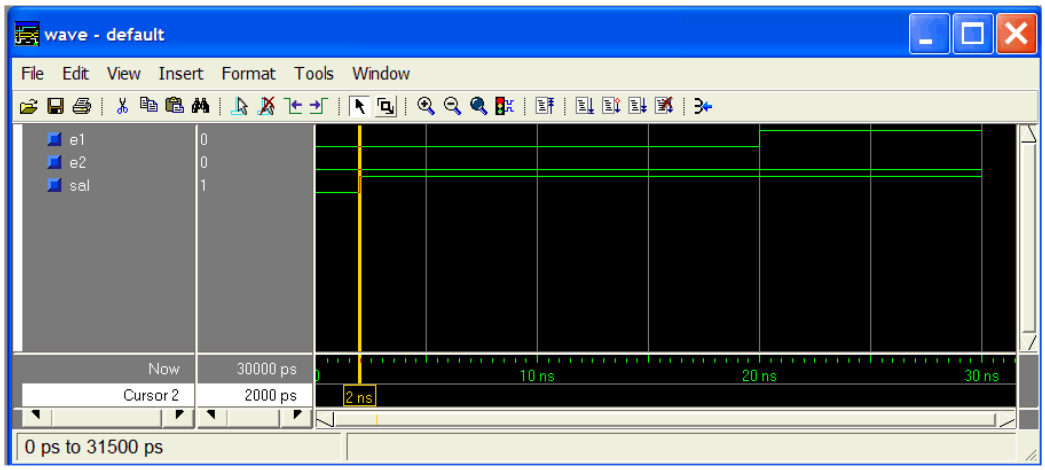


Figura 23. Aspecto de la ventana ***wave*** después de reanudar la ejecución durante 10ns más.

7. Forzar ahora la señal *e2* a 1 para comprobar el correcto funcionamiento del modelo con una nueva combinación de la tabla de verdad. La figura ilustra como queda el cronograma.

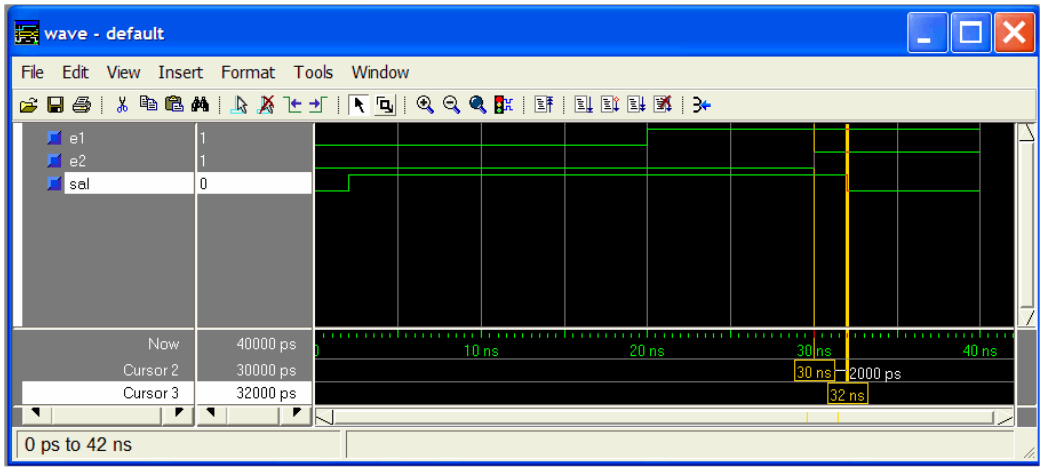


Figura 24. Aspecto de la ventana ***wave*** después de reanudar la ejecución durante 10ns más con nuevos valores en las señales de entrada.

Como se puede ver en la figura anterior, podemos introducir más de un cursor sobre la ventana de onda. De esta manera se pueden medir intervalos de tiempo entre eventos. El comando se emite utilizando la orden ***Insert*** ➔ ***Cursor*** de la barra de menú de la ventana de onda (***wave***). Los cursores se pueden arrastrar con el ratón y al soltarlos se aproximan de manera automáticamente al evento más cercano. Cuando hay más de un cursor sobre la ventana de onda, se indican los intervalos de tiempo entre ellos.