



Facultad de Informática
Universidad Complutense de Madrid

LABORATORIO DE FUNDAMENTOS DE COMPUTADORES I

PRÁCTICA 2:

DISEÑO Y SIMULACIÓN DE UN SUMADOR BINARIO DE 2 BITS MEDIANTE VHDL

El objetivo de esta práctica es tomar contacto con la herramienta de simulación de circuitos integrados, ModelSim de Mentor Graphics y con el lenguaje de descripción de hardware VHDL. Para ello se va a implementar y simular un sumador binario de 2 bits.

DISEÑO CON VHDL

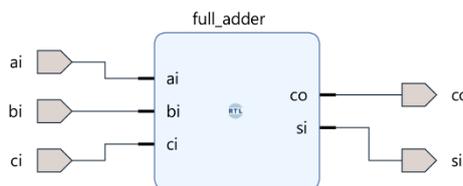
Un diseño VHDL tiene dos partes:

- Entity.
- Architecture.

La entity describe la interfaz del módulo, es decir, define las entradas, las salidas y sus tipos. En este laboratorio sólo estudiaremos dos tipos, a saber:

- Tipo bit
- Tipo bit_vector (n-1 downto 0) que define un vector de n bits.

En la siguiente figura se puede ver la caja negra que describe el interfaz del sumador total que vamos a implementar inicialmente



La declaración de entidad para esta caja negra es la siguiente:

```
ENTITY sumador IS  
    PORT(op1, op2 : IN bit; c_ent : IN bit; sum : OUT bit; c_sal : OUT bit);  
END sumador;
```

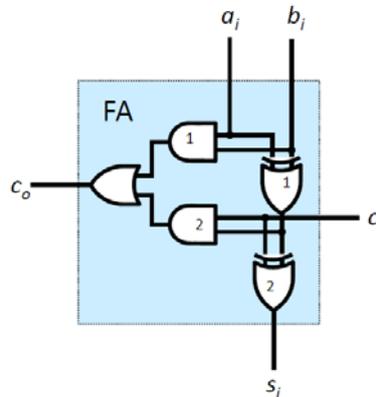
Como se puede apreciar, el módulo tiene:

- Tres entradas de datos de tipo bit.
- Dos salidas, c_sal y sum de tipo bit.

Por otro lado, la arquitectura describe la funcionalidad del módulo. Una misma entidad puede tener muchas arquitecturas diferentes. En este curso nos centraremos en dos tipos de arquitecturas:

- Descripción de la estructura del módulo mediante instancias de componentes previamente diseñadas y conectadas entre sí mediante señales.
- Descripción del comportamiento del módulo mediante expresiones lógicas.

En esta práctica vamos a realizar una descripción estructural de la arquitectura del sumador total utilizando las puertas lógicas and2, or2 y xor2 incluidas en el archivo sumador.vhd que se proporciona al alumno. En concreto ese archivo implementa la siguiente red de puertas lógicas:



A las conexiones internas de esta red las vamos a nombrar de la siguiente manera:

- sal_and1: la conexión entre and1 y or
- sal_and2: conexión entre and2 y or
- sal_xor1 la conexión entre xor1 y xor2 y and2

Visto lo anterior, la arquitectura estructural del sumador total es la siguiente:

```

ENTITY full_adder IS
    PORT(ai, bi : IN bit; ci : IN bit; si : OUT bit; co : OUT bit);
END full_adder;

--USE work.ALL;
ARCHITECTURE puertas OF full_adder IS
--declaración de componentes
    COMPONENT or2
        PORT(i1, i2 : IN bit; o : OUT bit);
    END COMPONENT;
    COMPONENT and2
        PORT(i1, i2 : IN bit; o : OUT bit);
    END COMPONENT;
    COMPONENT xor2
        PORT(i1, i2 : IN bit; o : OUT bit);
    END COMPONENT;

--declaración de señales internas
    SIGNAL sal_and1, sal_and2, sal_xor1: bit;
--empieza el cuerpo de la arquitectura
BEGIN
    i_and1 : and2 PORT MAP(ai, bi, sal_and1);
    i_xor1 : xor2 PORT MAP(ai, bi, sal_xor1);
    i_and2 : and2 PORT MAP(sal_and1, ci, sal_and2);
    i_xor2 : xor2 PORT MAP(sal_xor1, ci, si);
    i_or : or2 PORT MAP(sal_and1, sal_and2, co);
END puertas;

```

En este fragmento de código se pueden observar los siguientes elementos:

Después de la cabecera de la arquitectura aparece la declaración de los componentes que se van a utilizar en el diseño. La declaración de componentes sirve para establecer el tipo de

diseños que se van a utilizar como elemento de construcción de otro diseño. En nuestro caso se declaran tres componentes la puerta lógica or2 , la and2 y la xor2. Por ejemplo, el fragmento de código:

```
COMPONENT or2
  PORT(i1, i2 : IN bit; o : OUT bit);
END COMPONENT;
```

declara el componente or2, definido por dos entradas y una salida, de tipo bit.

A continuación, se declaran las señales internas que conectan las diferentes puertas lógicas entre sí, en concreto se declaran las señales sal_and1, sal_and2 y al_xor1.

```
SIGNAL sal_and1, sal_and2, sal_xor1 : bit;
```

Ya dentro del cuerpo de la arquitectura aparecen las instancias de las componentes que vamos a utilizar. Como hemos visto un componente es el tipo de diseño que utilizamos como elemento de construcción, y una instancia es el elemento de construcción propiamente dicho. Si nos fijamos en la figura que contiene la red de puertas lógicas que implementa el sumador total vemos que esta necesita puertas de tipo and2, or2 y xor2. Esto son los tipos de componente que tiene que utilizar. En concreto necesita dos puertas and, dos puertas xor y una puerta and. Estos son las instancias de las componentes que se pueden ver en el fragmento de código.

Nota: Los componentes deben estar implementados previamente a su uso. En este laboratorio se le proporcionaran al alumno en el archivo full_adder.

MODELSIM

DESCARGA E INSTALACIÓN

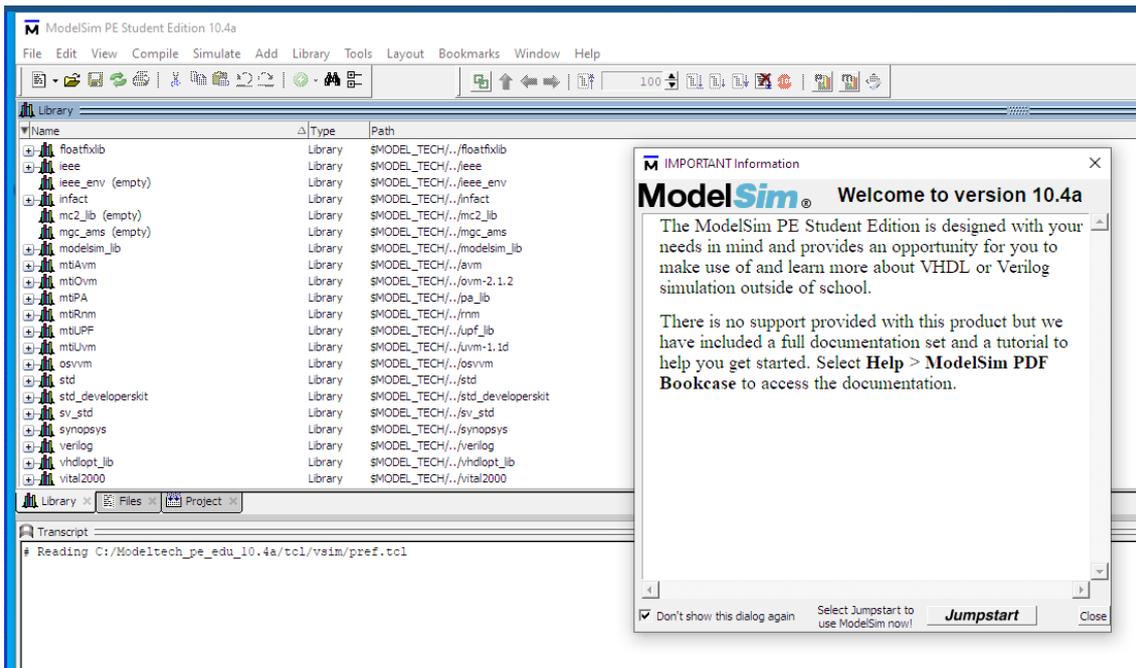
El ModelSim es un simulador de lenguajes de descripción de hardware. Nos podemos descargar una versión gratuita del mismo, llamada ModelSim PE Student Edition, del siguiente enlace:

https://www.mentor.com/company/higher_ed/modelsim-student-edition

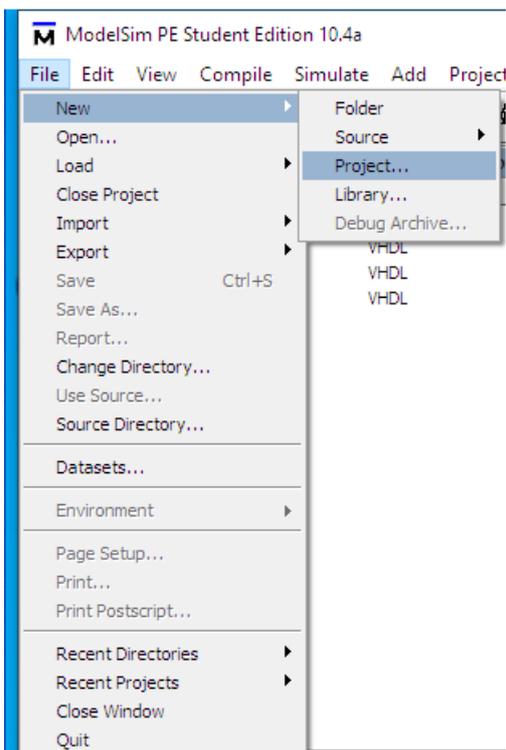
cuando os lo descarguéis os enviarán un correo con una descripción detallada de todos los pasos que tenéis que seguir para instalar la herramienta

CREAR UN PROYECTO EN MODELSIM

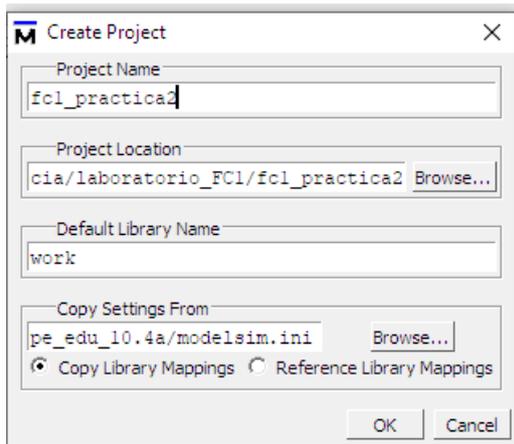
1. Lo primero es crear un directorio que vamos a llamar fc1_practica2.
2. En este directorio copiamos el archivo full_adder.vhd proporcionado al alumno
3. abrimos el ModelSim, y cerramos la ventana de bienvenida



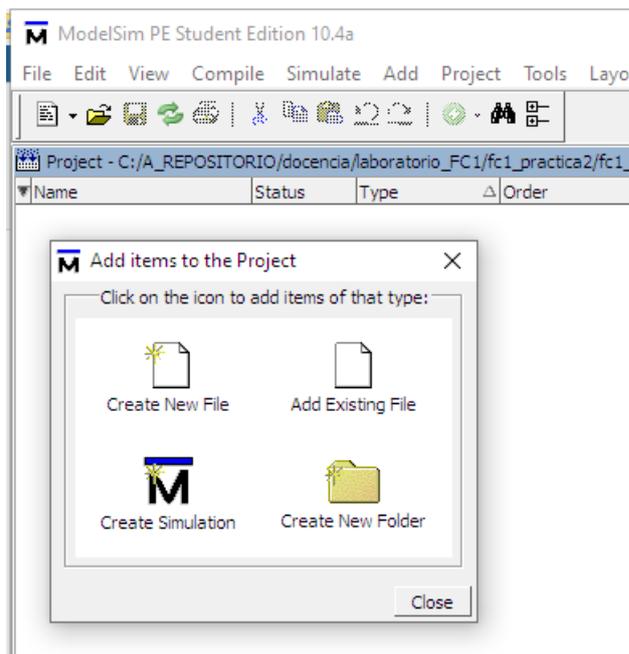
4. Seleccionamos File→New→Project...



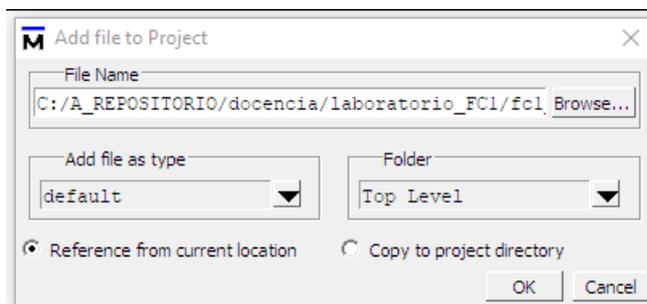
5. Se rellena la ventana que aparece con el nombre del proyecto (fc1_practica2) y se busca el directorio fc1_practica2 que hemos creado con anterioridad, y pulsas OK



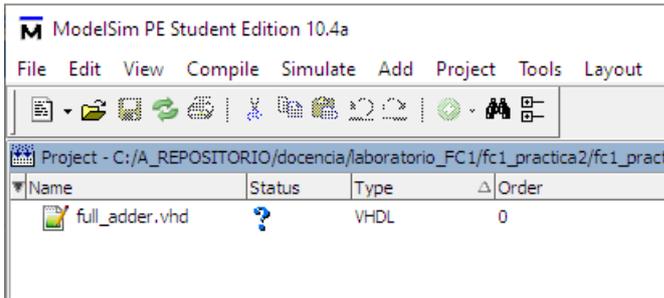
6. De la ventana que aparece seleccionas la opción Add Existing File



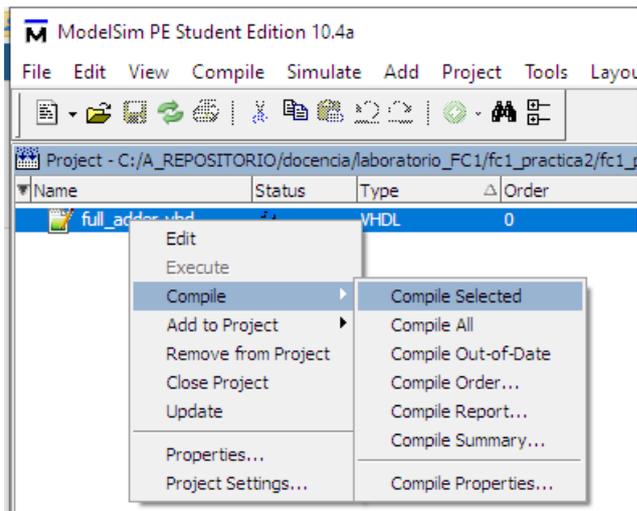
7. Seleccionar con el browser el archivo full_adder.vhd que has almacenado con anterioridad en el directorios fc1_practica2



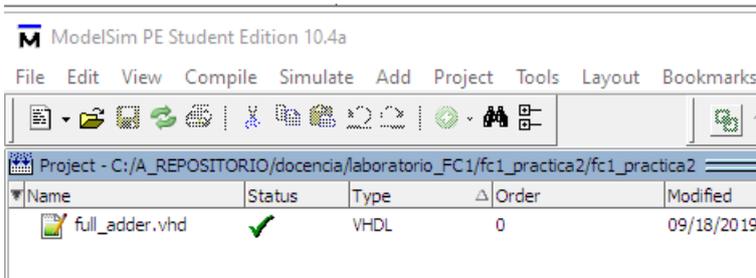
8. En la ventana proyecto aparece el archivo full_adder.vhd con una interrogación azul, lo que indica que todavía no se ha compilado



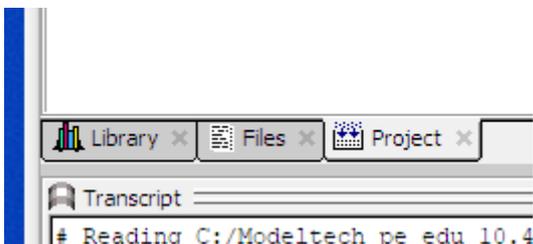
9. Para compilar, seleccionar Compile → Compile Selected



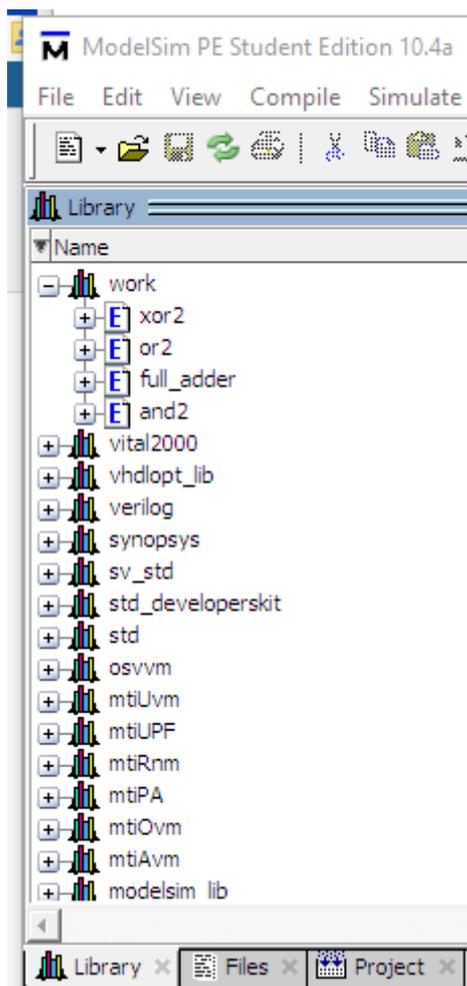
Si la compilación ha sido correcta la interrogación se ve sustituida por una v verde



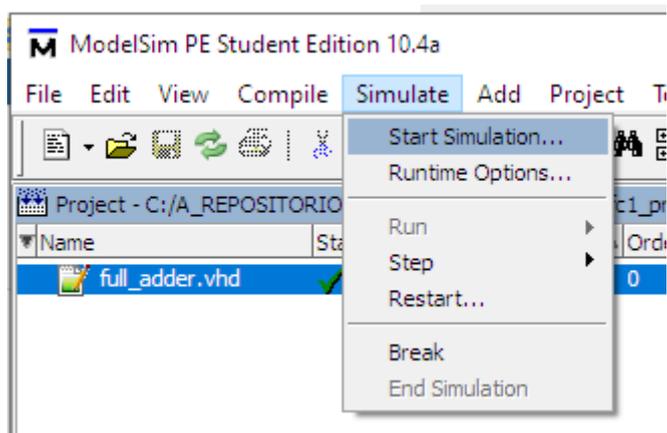
Además, seleccionando la pestaña Library



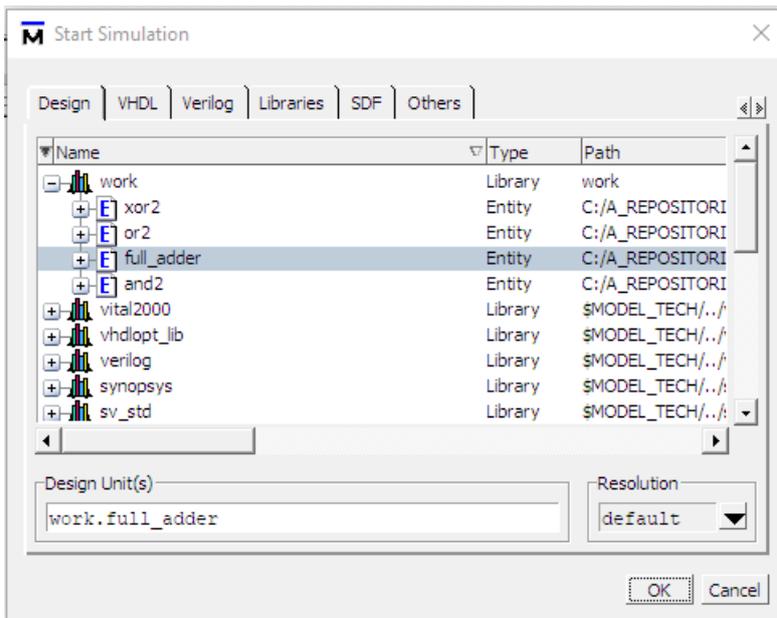
Se abre una ventana en la que aparece entre otras la librería work en la que se incluyen las entidades vhd compiladas. Darse cuenta que en work aparecen las cuatro entidades contenidas en el archivo full_adder.vhd.



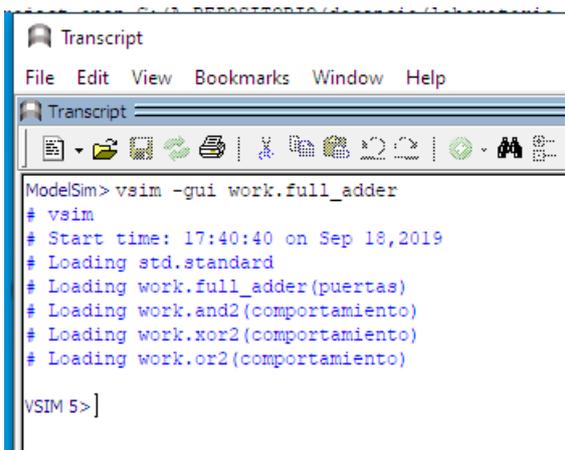
10. Para simular el archivo seleccionar Simulate→Start Simulation



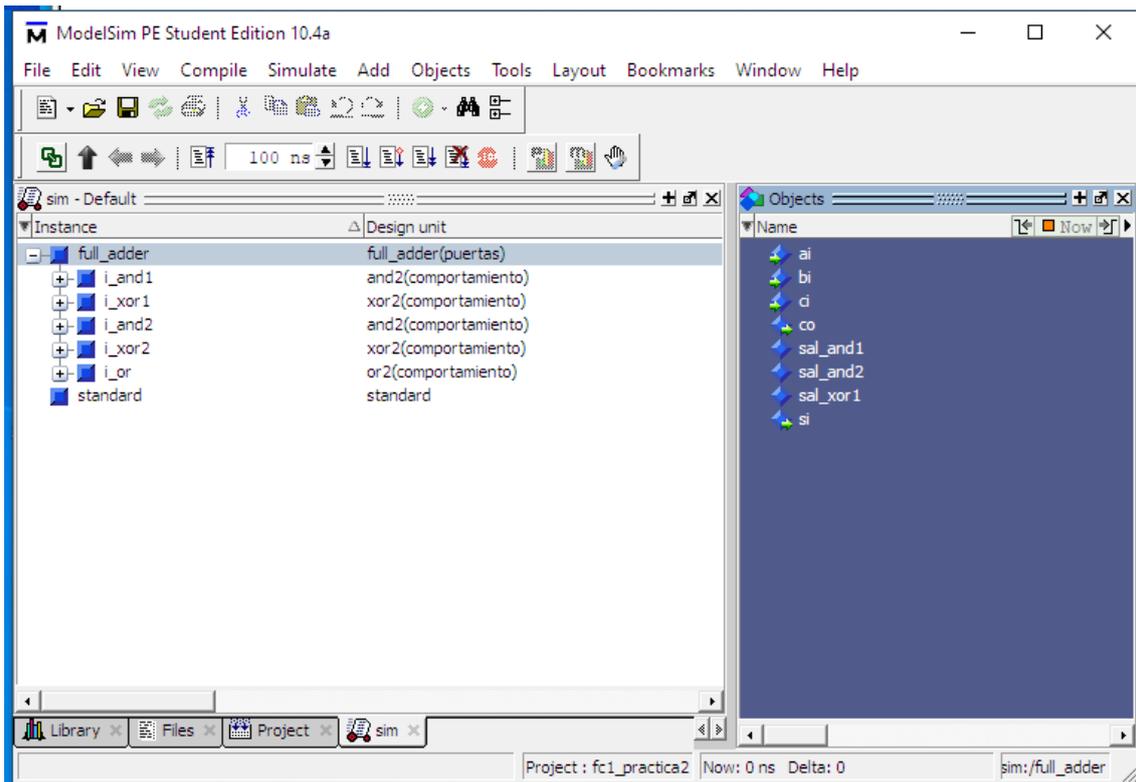
Aparece la ventana Start Simulation, despliega el contenido del directorio work y selecciona el archivo full_adder y ok



En la ventana transcript aparece la siguiente información indicando que se han cargado correctamente todas las entidades necesarias del diseño

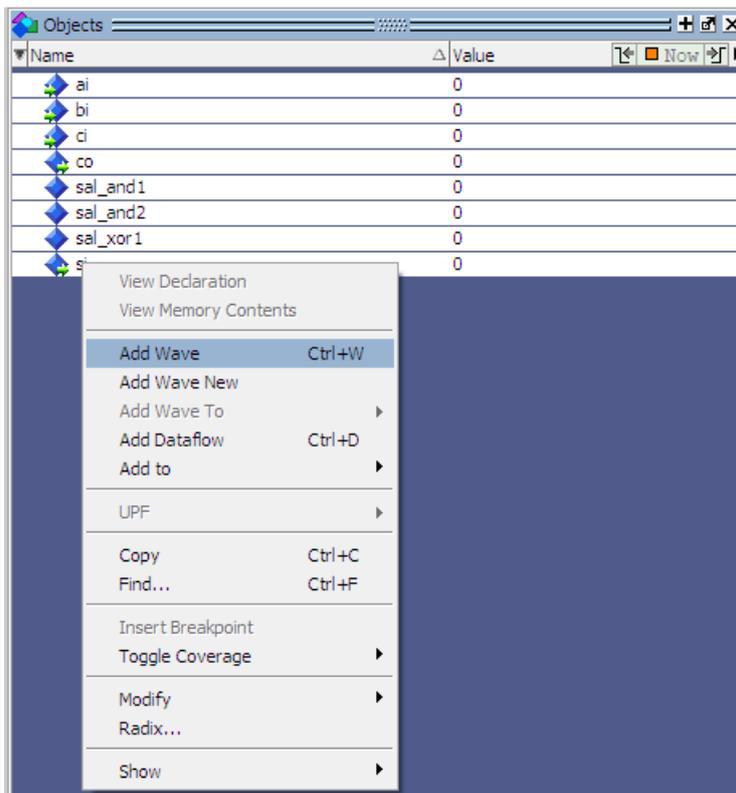


Además, se abre una nueva ventana bajo la pestaña Sim

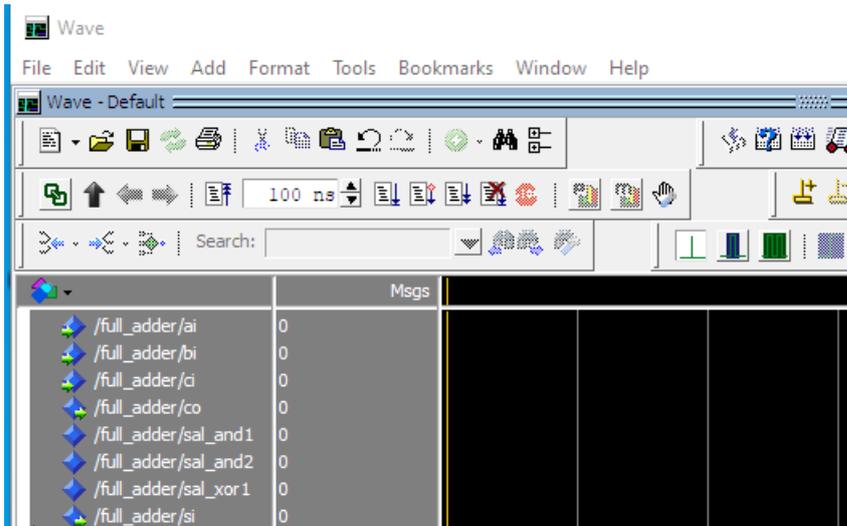


En la parte izquierda de esta ventana aparecen la jerarquia del diseño con sus instancias, en este caso de full_adder cuelgan las instancias i_and1, i_and2,..etc. en la parte de la derecha aparece la ventana objetos que incluye todas la entradas, salidas y señales internas del diseño. En ocasiones esta ventana no se abre automáticamente, para abrila seleccionar view→Objects

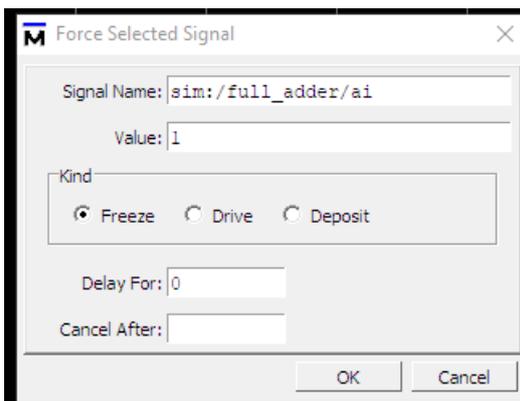
11. Selección de señales que se quieren simular. Selecciona todas las señales que se quieren simular, pincha derecha y selecciona add wave



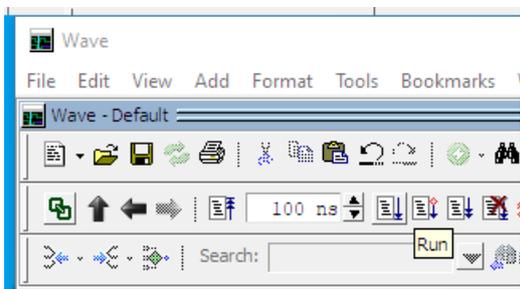
Como resultado se abre la venta de ondas en las que se van a observar los cronogramas de la simulación



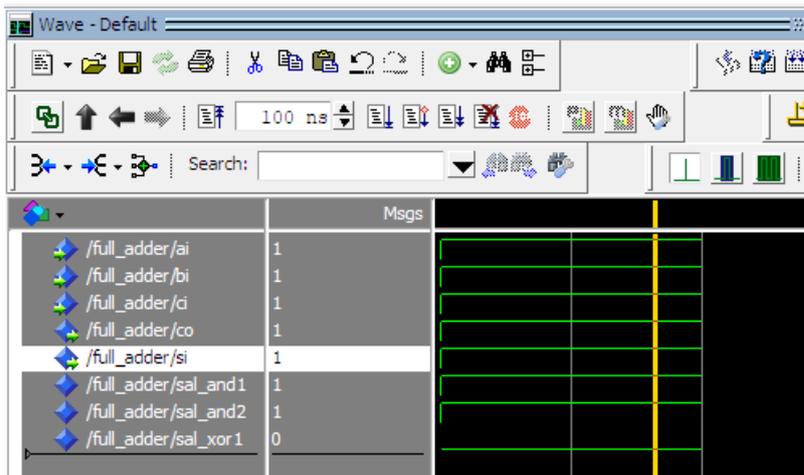
12. Añadir señales de entrada al simulador. En nuestro caso las señales de entrada al sumador son ai, bi, ci, por lo tanto debemos dar valores a estas entradas para ver si el diseño es correcto. Por ejemplo vamos a dar los valores 1,1,1 a estas tres entradas y vamos a comprobar en el cronograma que las dos salidas si y co se ponen a 1. Para ello selecciona con la derecha la señal ai y selecciona force y en value pon un uno



Y haz lo mismo para bi y ci y selecciona el icono run

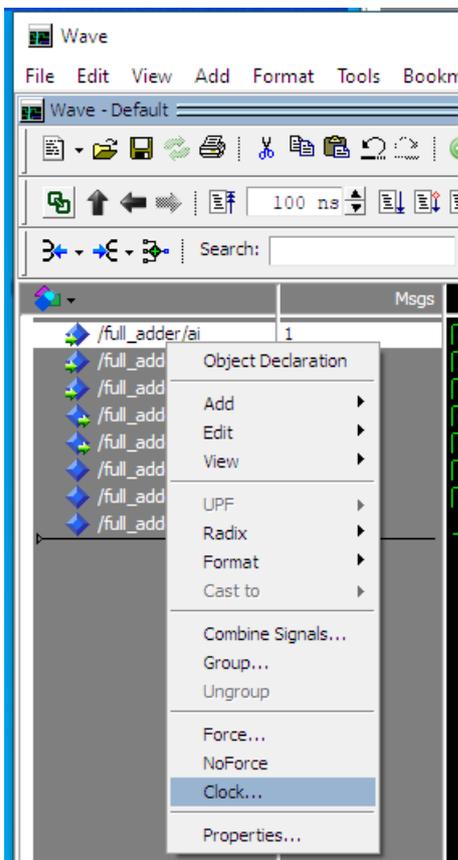


Y en la ventana aparece

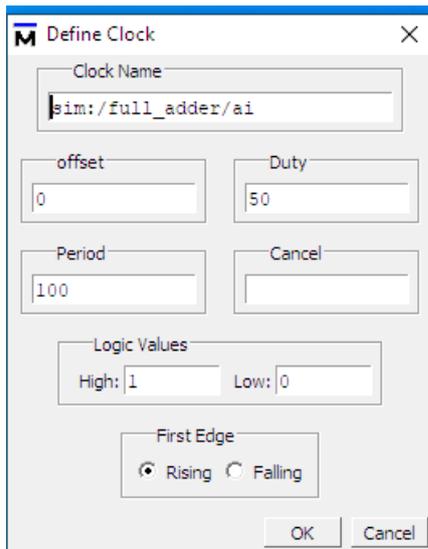


Se puede ver que si y co valen 1 cuando ai, bi y ci valen 1.

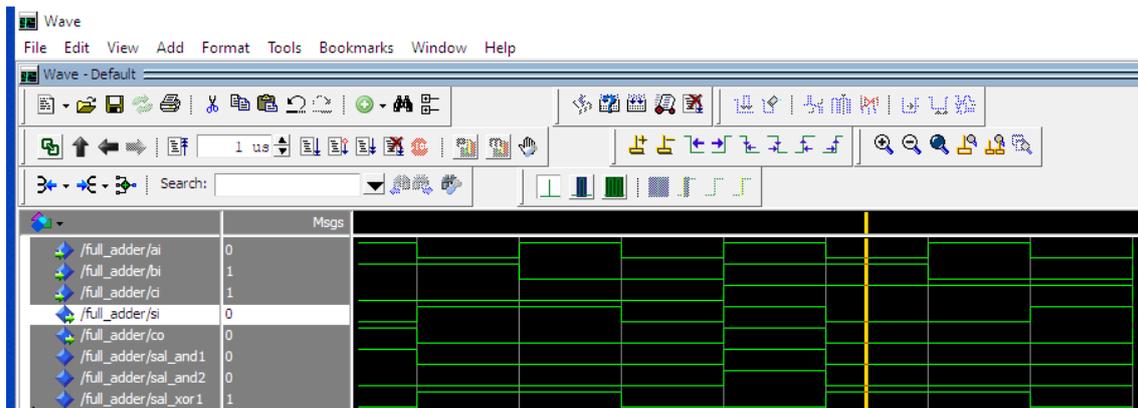
13. Generar relojes para la simulación. Existe una forma de generar todas las posibles entradas del sumador y es utilizando relojes. Pinchar la entrada ai con la derecha y seleccionar clock



Aparece la ventana



Deja todo como está y selecciona ok. Haz lo mismo para bi y si pero cambiando el periodo, en el primer caso a 200 y en el segundo a 400.



DESARROLLO DE LA PRÁCTICA

1. Crear un proyecto que incluya el fichero full_adder.vhd proporcionado al alumno, compilarlo y simularlo comprobando que el sumador funciona correctamente.
2. Modificar el archivo full_adder de manera que implemente un sumador binario de dos bits mediante una arquitectura estructural que utilice que utilice el full_adder.vhd del apartado anterior como componente, siguiendo la estructura explicada en las transparencias.

