Facultad de Informática
Universidad Complutense de Madrid

LABORATORIO DE FUNDAMENTOS DE COMPUTADORES 1

PRÁCTICA 1: DISEÑO Y SIMULACIÓN DE UNA FUNCIÓN COMBINACIONAL MEDIANTE VHDL Y LA HERRAMIENTA MODELSIM

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 la función combinacional:

$$F(x2, x1, x0) = \sum m(0,1,4,5)$$

DISEÑO CON VHDL

Un diseño VHDL tiene dos partes:

- Entity.
- Architecture.

La *entity* describe la interfaz del componente, 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 de la función combinacional que se va a implementar:

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

```
entity funcion_combinacional is

port (x2, x1, x0 :in bit;       -- entradas de datos
      f :out bit);              -- salida
end funcion_combinacional;
```

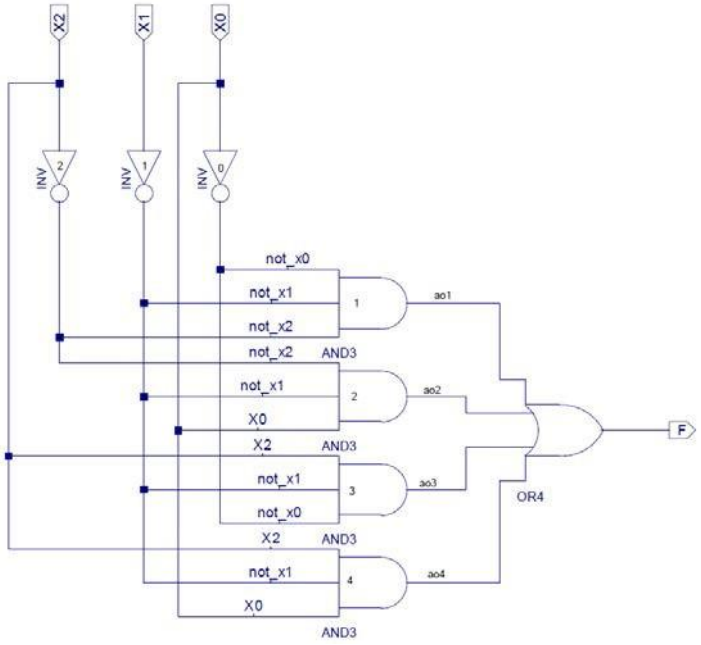
Como se puede apreciar, el componente tiene:

- Tres entradas de datos de tipo bit.
- Una salida de tipo bit.

Por otro lado, la *arquitectura* describe la funcionalidad del componente. Una misma entidad puede tener muchas arquitecturas diferentes. En este curso nos centraremos en

la arquitectura estructural en la que la descripción se realiza mediante instancias de componentes previamente diseñadas y conectadas entre sí mediante señales.

En esta práctica vamos a realizar una descripción estructural de la arquitectura del circuito combinacional utilizando las puertas lógicas and3, or4 e inv incluidas en el archivo *funcion_combinacional.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:

- not_xi: conexión entre un inversor y una puerta and
- aoi: conexión entre una puerta and y una or

Visto lo anterior, la descripción VHDL de la arquitectura estructural de la función combinacional es:

```
entity funcion_combinacional is
        port (  x2, x1, x0:     in bit;         -- entradas de datos
                f:              out bit);       -- salida
end funcion_combinacional;

architecture puertas of funcion_combinacional is
--declaración de componentes
component or4
        port(i1, i2, i3 ,i4 : in bit; o : out bit);
end component;
component and3
        port(i1, i2, i3, : in bit; o : out bit);
end component;
component inv
        port(i1 : in bit; o : out bit);
end component;

--declaración de señales internas
signal not_x2, not_x1, not_x0, ao1, ao2, ao3, ao4: bit;

--empieza el cuerpo de la arquitectura
begin
```

```
        i_inv_2:  inv port map(x2, not_x2);
        i_inv_1:  inv port map(x1, not_x1);
        i_inv_0:  inv port map(x0, not_x0);

        i_and_1: and4 port map(not_x2, not_x1,not_x0,ao1);
        i_and_2: and4 port map(not_x2,not_,x1,x0, ao2);
        i_and_3: and4 port map(x2, not_x1,not_x0,ao3);
        i_and_4: and4 port map(x2, not_x1,x0,ao4);

        i_or_1 : or4  port map(ao1,ao2,ao3,ao4,f);
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 elementos de construcción que se van a utilizar en el nuevo componente. En nuestro caso se declaran tres componentes: la puerta lógica or4, la and3 y el inversor inv. Por ejemplo, el fragmento de código:

```
component or4
        port(i1, i2,i3,i4 : in bit; o : out bit);
end component;
```

declara el componente or4, definido por cuatro 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:

```
        signal not_x2, not_x1, not_x0, ao1, ao2, ao3, ao4: 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 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 la función combinacional, vemos que ésta necesita puertas de tipo and3, or4 e inv. Estos son los tipos de componentes que tiene que utilizar. En concreto necesita cuatro instancias del componente and3, una instancia del componente or4 y tres instancias del componente inv. Estas son las instancias de las componentes que se pueden ver en el fragmento de código.

Nota: Un componente se puede instanciar tantas veces como sean necesarias

Nota: Los componentes deben estar implementados previamente a su uso. En este laboratorio se le proporcionan al alumno en el archivo *función_combinacional.vhd*.

MODELSIM

DESCARGA E INSTALACIÓN

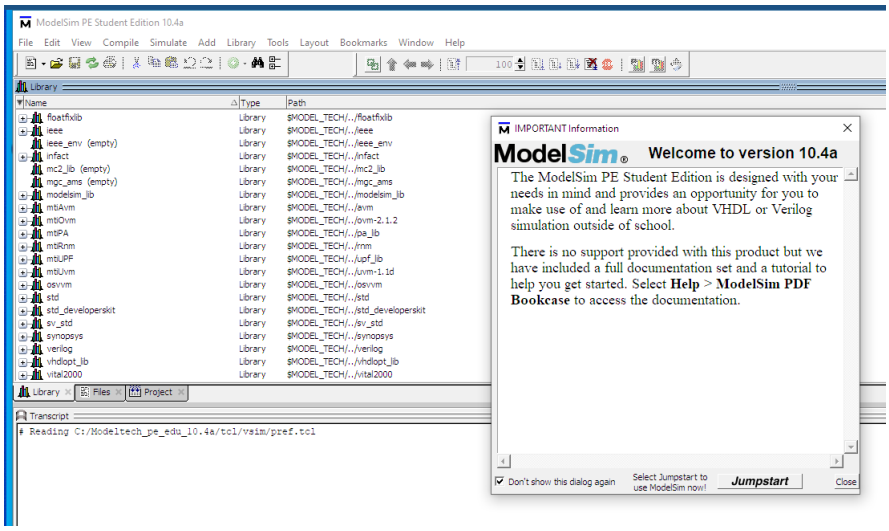
El *ModelSim* es un simulador de lenguajes de descripción de hardware. Se puede descargar una versión gratuita del mismo, llamada *ModelSim PE Student Edition*, del siguiente enlace:

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

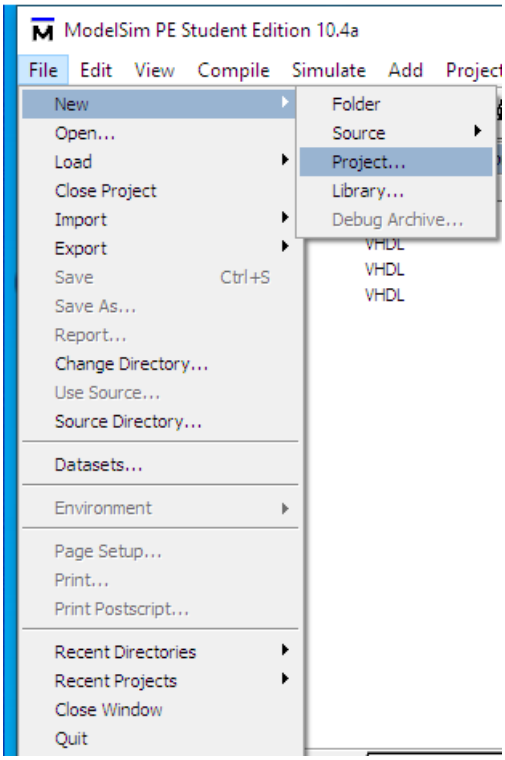
Al descargarlo se envía un correo con la descripción detallada de todos los pasos que se tienen que seguir para instalar la herramienta.

CREAR UN PROYECTO EN MODELSIM

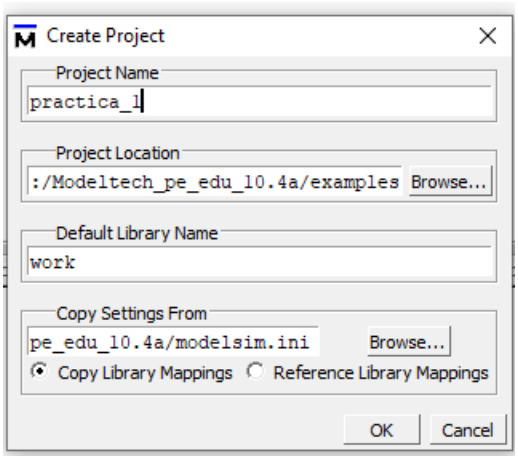
1. Lo primero es crear un directorio que vamos a llamar *fc1_practicas*.
2. En este directorio copiamos el archivo *funcion_combinacional.vhd* proporcionado al alumno.
3. Abrimos el *ModelSim*, y cerramos la ventana de bienvenida.



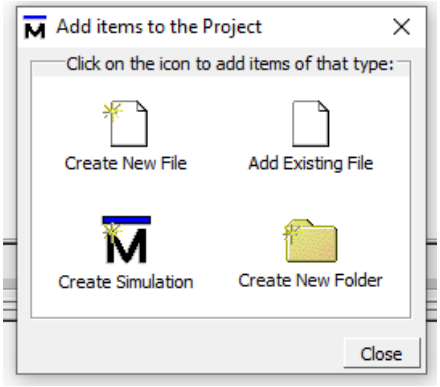
4. Seleccionamos File→New→Project…



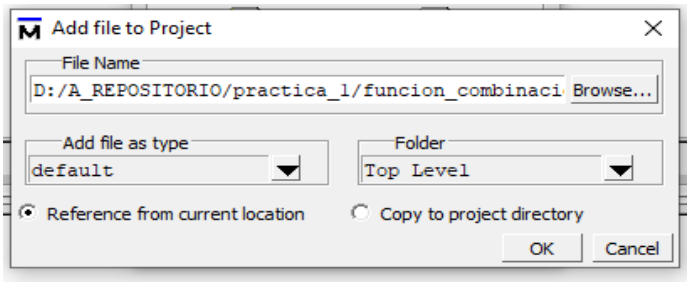
5. Rellenar la ventana que aparece con el nombre del proyecto (practica_1), buscar el directorio fc1_practicas que se ha creado con anterioridad y pulsar OK.



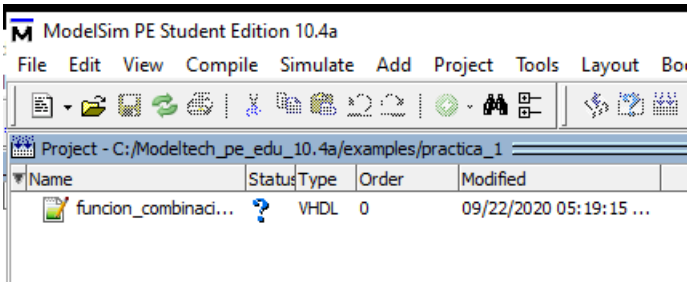
6. Seleccionar la opción Add Existing File.



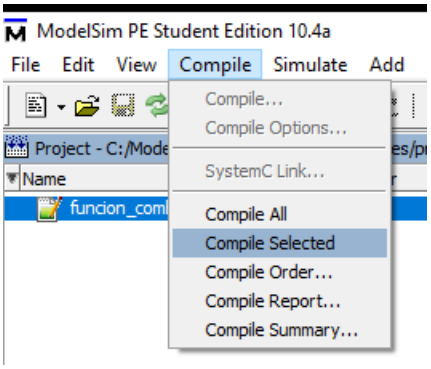
7. Seleccionar con el browser el archivo *funcion_combinacional.vhd* almacenado con anterioridad en el directorio fc1_practicas.



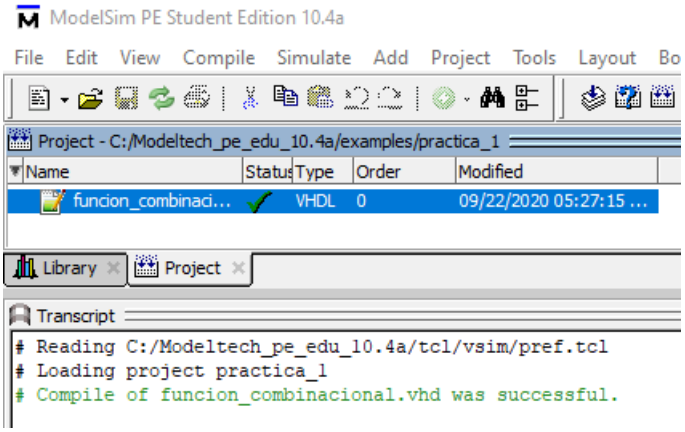
8. En la ventana Project aparece el archivo *funcion_combinacional.vhd* con una interrogación azul, que indica que todavía no se ha compilado.



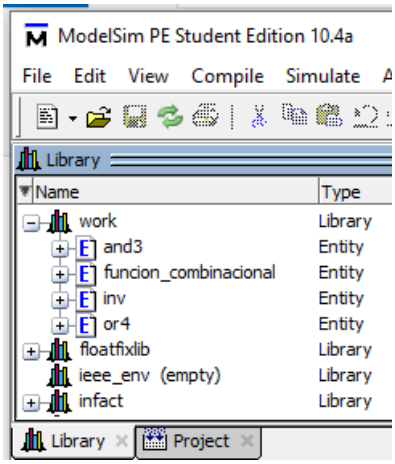
9. Para compilar, seleccionar el archivo y después seleccionar Compile→Compile Selected.



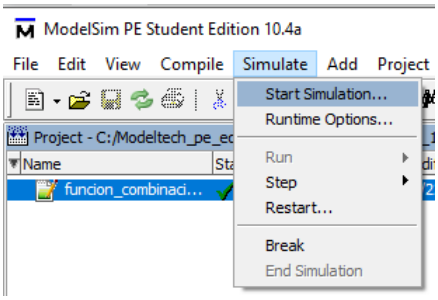
Si la compilación ha sido correcta la interrogación se ve sustituida por una "v" verde.



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 VHDL compiladas. Nótese que en work aparecen las cuatro entidades contenidas en el archivo funcion_combinacional.vhd:

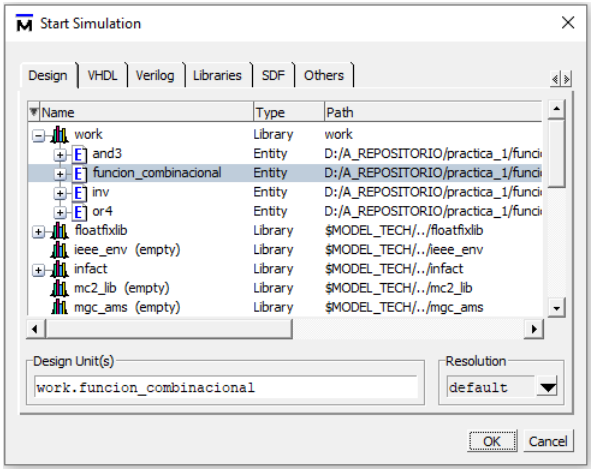


10. Para simular el archivo seleccionar *Simulate →Start* Simulation.

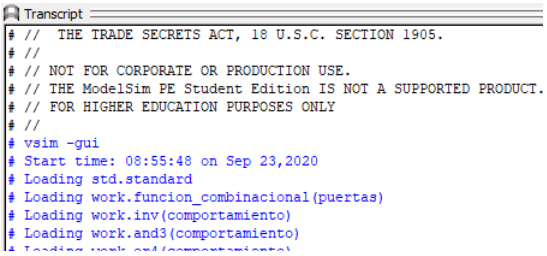


En la ventana Start Simulation, desplegamos el contenido del directorio Work, seleccionamos el

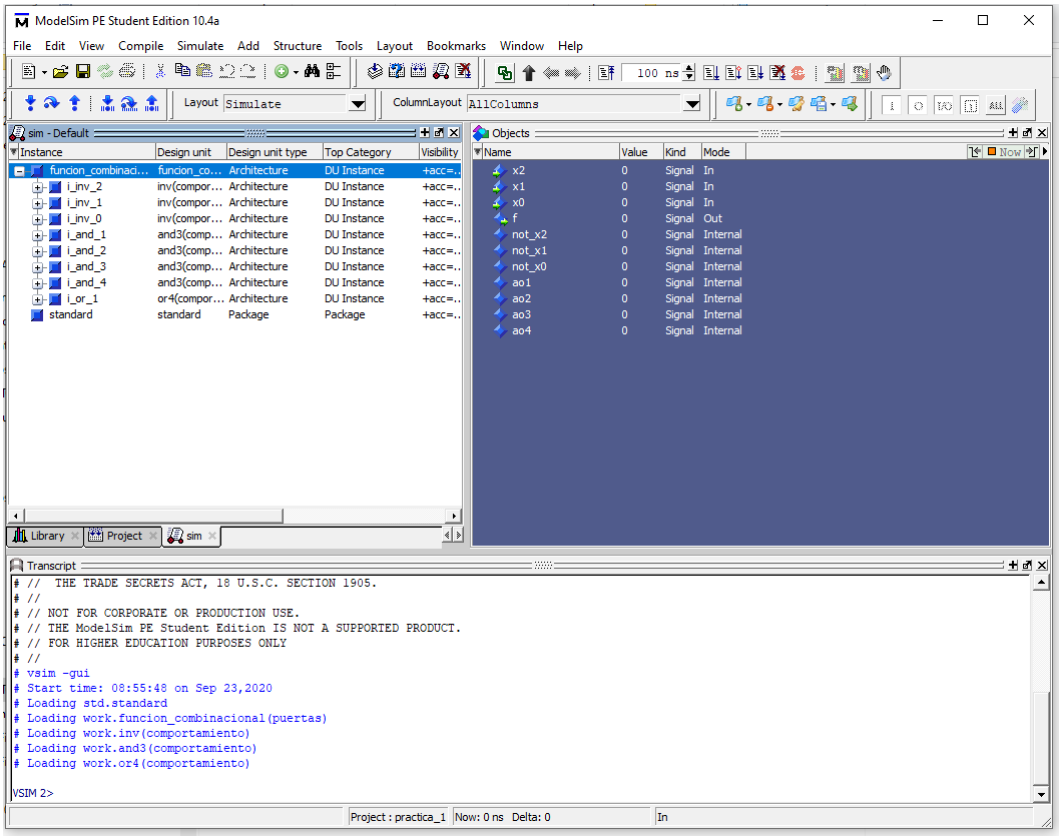
archivo funcion_combinacional.vhd y pulsamos OK.



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



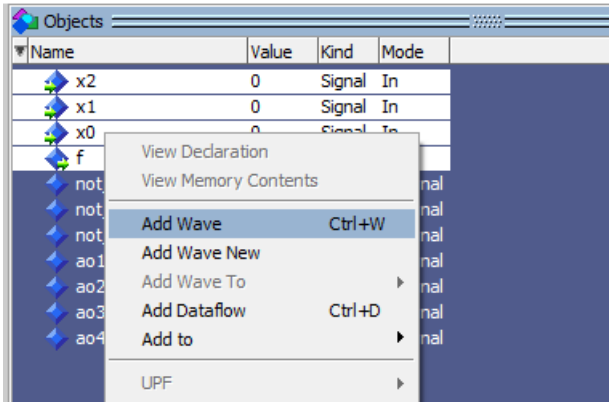
Además, se abre una nueva ventana bajo la pestaña sim:



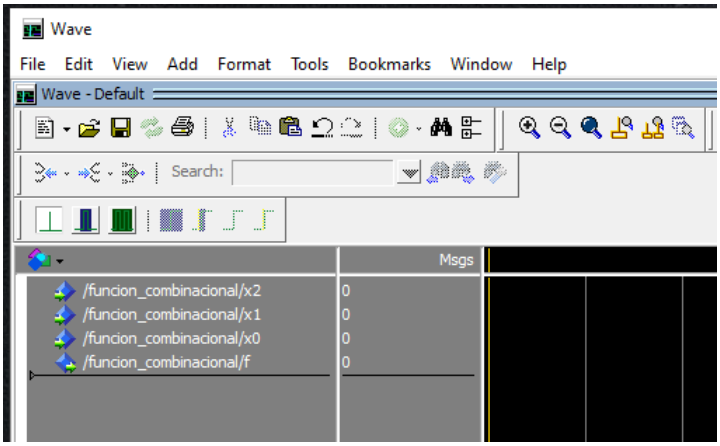
En la parte superior izquierda está el panel Instance en el que se incluye la jerarquía del diseño con sus instancias, en este caso de funcion_combinacional cuelgan las instancias i_and1, i_and2,...,etc. En la parte superior derecha está el panel Objects que incluye todas la entradas, salidas y señales

internas del diseño. En ocasiones esta ventana no se abre automáticamente, para abrirla seleccionar View->Objects.

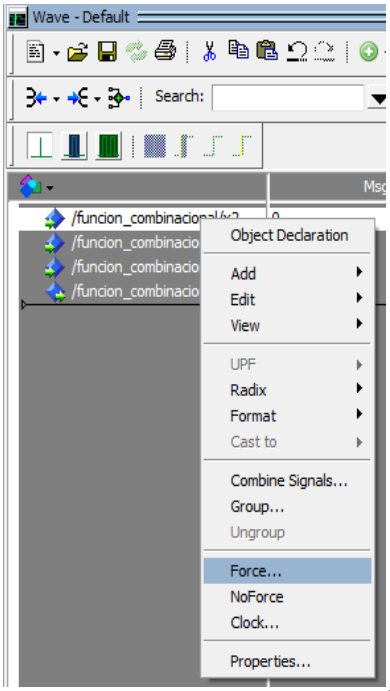
11. Seleccionar las señales que se quieren simular. Para ello se pincha con el botón derecho y se selecciona Add Wave.



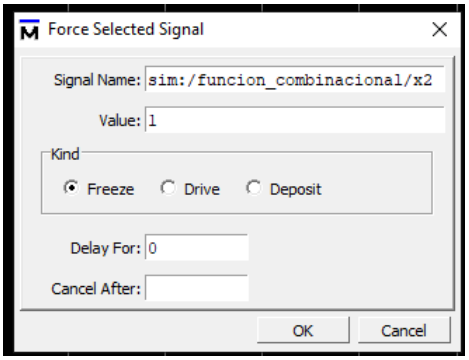
Como resultado se abre la ventana de ondas en la que se van a observar los cronogramas de la simulación:



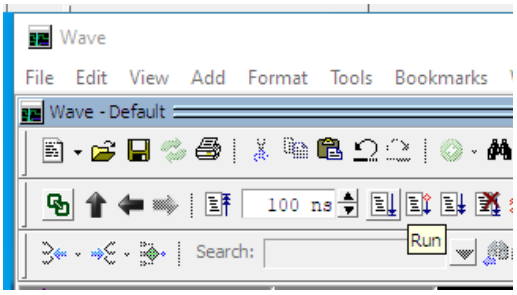
12. Añadir valores a las señales de entrada. En nuestro caso las señales de entrada al circuito son x2, x1, x0, por lo que se debe dar valores a estas entradas para comprobar si el diseño es correcto. Por ejemplo, se van a dar los valores 1,0,0 a las entradas y vamos a comprobar en el cronograma que la salida vale 1, puesto que la entrada corresponde al mintérmino 4 de la función. Para ello seleccionamos la señal x2, y tras pulsar el botón derecho del ratón, seleccionamos Force,



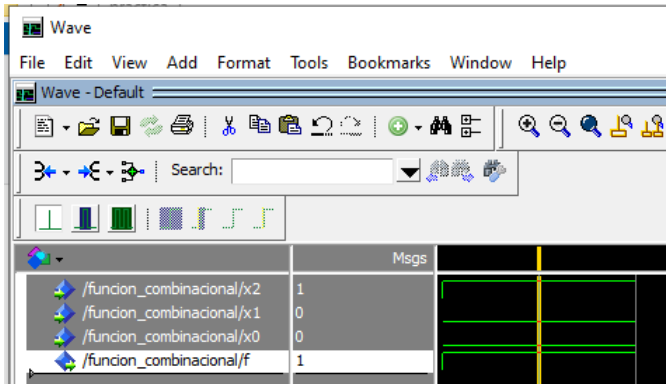
poniendo un 1 en el campo Value



Hacemos lo mismo para el resto de las entradas (con los valores especificados antes) y seleccionamos el icono Run.



Tras ello, en la ventana de ondas aparece las señales simuladas:



Se puede ver que f=1 con los valores forzados.

DESARROLLO DE LA PRÁCTICA

1. Crear un proyecto que incluya el fichero funcion_combinacional.vhd proporcionado al alumno, compilarlo y simularlo comprobando que funciona correctamente. Para ello el alumno debe simular todas las posibles combinaciones de entrada, según aparecen en esta tabla de verdad:

X2	X1	X0	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

2. Suponiendo que G(x2,x1,x0)=∑m(0,1,4,5,6), completar la plantilla que contiene el archivo Karnaugh.vhd de manera que implemente la función combinacional optimizada mediante mapas de Karnaugh