

SESIÓN 1: INTRODUCCIÓN A QUARTUS II – MÓDULO *OBRIR_ALLIBERAR* – MÓDULO *DISPLAY*

Los objetivos de esta práctica son conocer la herramienta Quartus II 13.0sp1, utilizarla para entrar el esquemático de los módulos **Obrir_Alliberar** y **Display** pertenecientes al controlador del ascensor y hacer una simulación de los mismos para verificar su correcto funcionamiento.

Previamente, es recomendable que hayáis leído las tres primeras páginas del documento “Controlador de Ascensor” que tenéis en el Campus Virtual, que resume el sistema que vamos a diseñar a lo largo de las seis sesiones de prácticas.

Tal como se ha indicado en la introducción a esta primera sesión de prácticas, todos los ficheros que generéis durante el desarrollo del proyecto a lo largo de las seis sesiones de prácticas, deberéis guardarlos en **la carpeta del proyecto** (esa carpeta es definida durante la creación del proyecto en esta primera sesión). Al finalizar cada sesión debéis:

- Comprimir la carpeta del proyecto en un fichero .zip
- Subir el fichero .zip a la nube de cada uno de los componentes del equipo de trabajo
- Borrar la carpeta del proyecto y el fichero .zip de la unidad de disco U.

Antes del inicio de cada sesión debéis tener en la nube la última versión de la carpeta del proyecto comprimida (tras el trabajo que hayáis hecho por vuestra cuenta desde la última sesión). Al inicio de la sesión de prácticas, descargaréis la carpeta del proyecto comprimida, la descomprimiréis y seguiréis desarrollando el proyecto en ella.

1. INTRODUCCIÓN

El módulo **Obrir_Alliberar** sirve para determinar cuándo se debe abrir la puerta del ascensor en un determinado piso y para indicar que una solicitud que se había hecho al ascensor para que fuera a un determinado piso ya ha sido satisfecha por el ascensor y, por tanto, la solicitud ya se puede liberar (eliminar).

Para realizar el diseño de un circuito en Quartus II, en primer lugar, es necesario crear un proyecto en Quartus II. Ello se explica detalladamente en el siguiente punto.

2. QUARTUS II 13.0

2.1 INTRODUCCIÓN AL ENTORNO DE DISEÑO DE ALTERA

Las FPGAs (*Field Programmable Gate Array*) son chips que contienen una matriz de celdas lógicas (puertas, biestables, multiplexores, etc.) y una red de líneas de conexión que corren horizontal y verticalmente, que pueden configurarse para que repliquen el circuito que deseamos. La configuración del contenido de la FPGA para que replique el circuito que deseamos se realiza a través de un “entorno de desarrollo” constituido por un paquete software que nos permite introducir el circuito (ya sea a nivel de esquema lógico o mediante lenguajes de descripción hardware como VHDL), más un dispositivo físico “programador” sobre el que ponemos la FPGA y que, conectado vía USB al ordenador, configura la FPGA de acuerdo con el circuito que hemos introducido. De esta manera podemos construir circuitos complejos dentro de un mismo chip.

En estas prácticas utilizaremos un entorno de desarrollo de ALTERA¹ llamado **QUARTUS II (versión 13.0 sp1)**. ALTERA ofrece una amplia gama de FPGAs que clasifica en familias (por ejemplo, ACEX[®] 1K, APEX[™] 20K, APEX II, ARM[®]based Excalibur[™], Cyclone[™], Cyclone II, FLEX[®] 6000, FLEX 10K[®], etc.), y en dispositivos dentro de cada familia (por ejemplo, los dispositivos EP20K200CF484C7, EP20K200CF484C8, etc. de la familia Cyclone II). En estas prácticas utilizaremos siempre el dispositivo **EP2C35F672C6** de la familia **Cyclone II**.

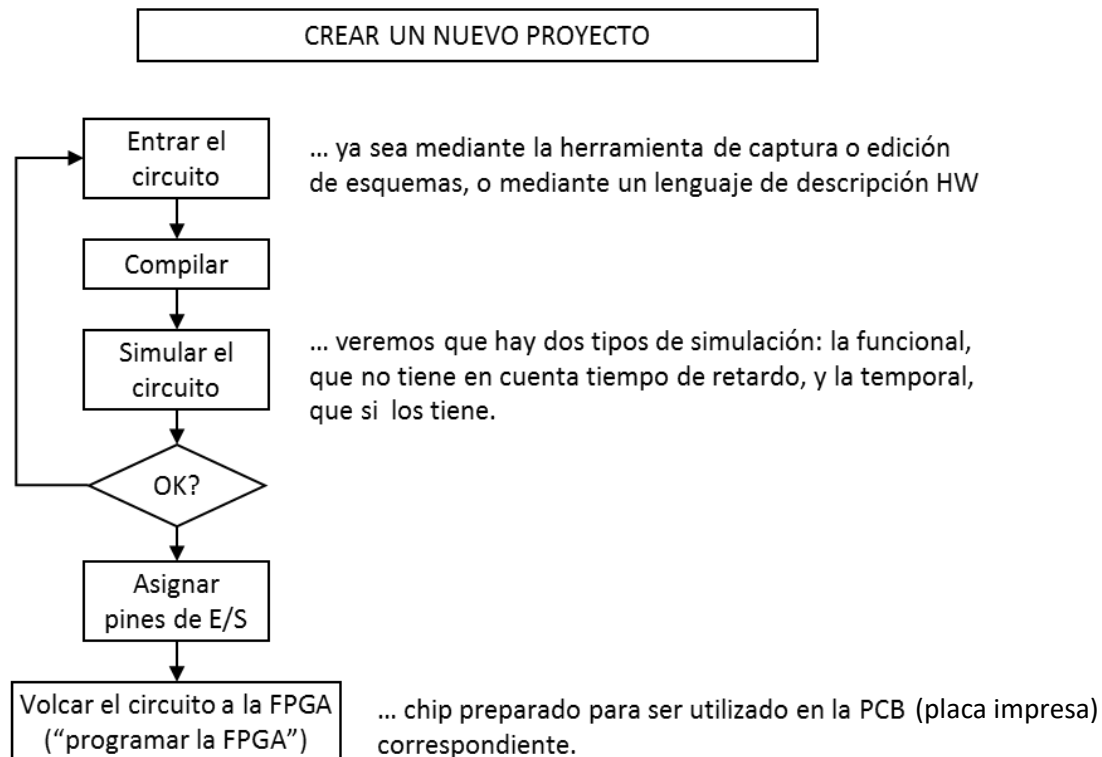
2.2 DEFINICIÓN Y DIAGRAMA DE FLUJO DE UN PROYECTO

Los circuitos que se implementan sobre FPGAs son habitualmente circuitos complejos, con miles o incluso centenares de miles de puertas lógicas, por lo que el circuito se organiza como un conjunto jerárquico de módulos que no solo hay que introducir en el ordenador sino también comprobar su funcionamiento, mediante simuladores, antes de su volcado a la FPGA. Todos los ficheros que resultan de esta complejidad se guardan en lo que ALTERA denomina un **PROYECTO**.

La figura siguiente muestra los pasos a seguir para implementar un circuito sobre una FPGA:

¹ ALTERA Corporation era una compañía dedicada al desarrollo y producción de dispositivos lógicos programables. En 2015 fue absorbida por Intel.

DIAGRAMA DE FLUJO DEL DISEÑO DE UN PROYECTO



2.2.1 CREACIÓN DE UN NUEVO PROYECTO

El primer paso es siempre crear un **proyecto** nuevo. El proyecto es una especie de base de datos que contendrá todos los ficheros que nosotros creamos al introducir esquemas o generar simulaciones, más muchos otros que genera y actualiza automáticamente Quartus. **Es muy importante, para mantener la coherencia de la estructura de datos, trabajar siempre dentro del proyecto, y no caer en la tentación de entrar o borrar ficheros directamente en el directorio del proyecto.**

Todos los ficheros que se generen en estas prácticas deben estar dentro del proyecto.

Para crear el proyecto de estas prácticas, al que llamaremos "**ControlAscensor**", haz lo siguiente:

1) Arranca Quartus II 13.0 haciendo doble clic sobre el icono del escritorio:

2) Clica en

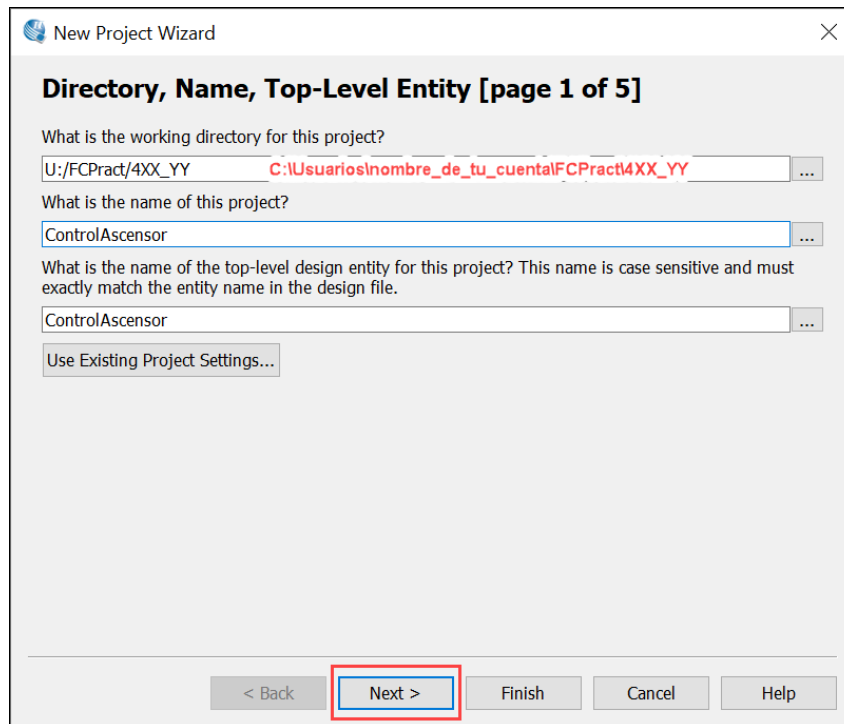


File → New Project Wizard

Si se abre una ventana con información introductoria marca la casilla *Don't show me this introduction again* para que esta información no te aparezca repetidamente y clica en *Next*.

3) Te aparece un primer formulario en el que deberás introducir el directorio (*working directory*) donde vas a guardar el proyecto, el nombre del proyecto (*name of the project*) y la celda principal del circuito (*name of the top-level design entry*). Como *working directory* has de indicar **U:/FCPract/4XX_YY** (o en caso de que uses tu propio ordenador, **C:/Usuarios/nombre_de_tu_cuenta/FCPract/4XX_YY**), donde **4XX_YY** es tu identificador de equipo de trabajo (p. e., **411_01**). Por celda principal se entiende el bloque que se encuentra en el nivel de jerarquía superior, y que vamos a llamar *ControlAscensor*.

Finalmente, clicas en *Next*.



Nota sobre la Top-Level Entity:

La *Top-Level Entity* es el bloque que se encuentra en el nivel de jerarquía más alto. Cuando dentro del proyecto tengamos diversos bloques, la manera de decirle a Quartus que queremos trabajar con uno determinado es definir este módulo como *top-level entity*. Para definir un módulo como *Top-Level Entity*: (1) clicas con el botón derecho del ratón sobre el fichero .bdf del módulo y (2) seleccionas 'Set as Top-Level Entity'.

4) A continuación, se abre un segundo formulario que permite al diseñador añadir bloques ya creados para, por ejemplo, poder reutilizar bloques diseñados en otros proyectos. En nuestro caso no hay que hacer nada, así que simplemente clicas en *Next*.

New Project Wizard

Add Files [page 2 of 5]

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
Note: you can always add design files to the project later.

File name: ...

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
-----------	------	---------	-----------------------------	-------------

Specify the path names of any non-default libraries.

5) El tercer formulario sirve para escoger el tipo de FPGA que vas a utilizar. En estas prácticas, como ya hemos dicho, trabajaremos con la familia (*Family*) **Cyclone II** y el dispositivo (*Available devices*) **EP2C35F672C6**. Al acabar, de nuevo, clicas en *Next*.

New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

Device family
Family: **Cyclone II**
Devices: All

Target device
☐ Auto device selected by the Filter
☒ Specific device selected in 'Available devices' list
☐ Other: n/a

Show in 'Available devices' list
Package: Any
Pin count: Any
Speed grade: Any
Name filter:
☒ Show advanced devices ☐ HardCopy compatible only

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit
EP2C35F672C6	1.2V	33216	475	483840	70
EP2C35F672C7	1.2V	33216	475	483840	70

Companion device
HardCopy:
☐ Limit DSP & RAM to HardCopy device resources

6) El cuarto formulario tampoco lo vamos a utilizar (permite trabajar con otras herramientas de diseño no incluidas en Quartus II). Deja todos los campos en blanco y clicas en *Next*.

EDA Tool Settings [page 4 of 5]

Specify the other EDA tools used with the Quartus II software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	<None>	<None>	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Formal Verific...	<None>		
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back **Next >** Finish Cancel Help

7) Finalmente se abre una ventana que resumen los datos del proyecto. Compruébalos; si detectas algún error clicas en *Back* hasta llegar a la información que hay que corregir y, si son correctos, clicas en *Finish*.

Summary [page 5 of 5]

When you click Finish, the project will be created with the following settings:

Project directory: U:/FCPract/4XX_YY

Project name: ControlAscensor

Top-level design entity: ControlAscensor

Number of files added: 0

Number of user libraries added: 0

Device assignments:

- Family name: Cyclone II
- Device: EP2C35F672C6

EDA tools:

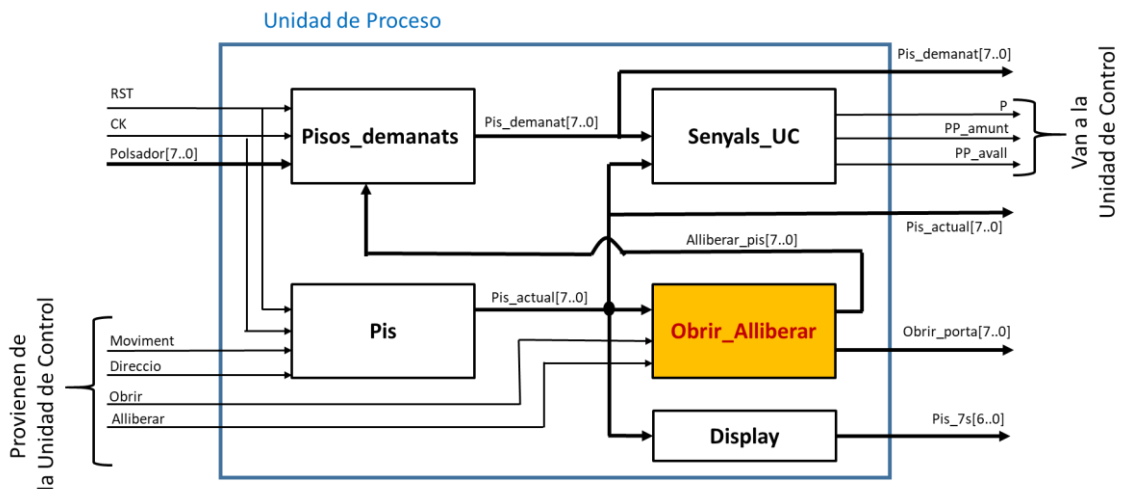
- Design entry/synthesis: <None> (<None>)
- Simulation: <None> (<None>)
- Timing analysis: ()

Operating conditions:

- Core voltage: 1.2V
- Junction temperature range: 0-85 °C

< Back Next > **Finish** Cancel Help

3. EL MÓDULO *OBRRIR_ALLIBERAR*



Tal como decíamos en la introducción, el módulo **Obrir_Alliberar** sirve para determinar cuándo se debe abrir la puerta del ascensor en un determinado piso y para indicar que una solicitud que se había hecho al ascensor ya ha sido atendida y, por tanto, puede ser liberada (eliminada).

El módulo **Obrir_Alliberar** genera los buses **Alliberar_pis[7..0]** y **Obrir_porta[7..0]** a partir del bus **Pis_actual[7..0]**, procedente del módulo **Pis**, y de la señales **Obrir** y **Alliberar** que le llegan desde la **Unidad de Control**.

Conozcamos en primer lugar las señales que recibe de entrada. El bus **Pis_actual[7..0]** indica el piso en el que se encuentra el ascensor. Cada bit del bus está asociado a un piso y si vale 1 significa que el ascensor se encuentra en ese piso (p.e., si **Pis_actual[7..0]** es 00000001 eso significa que el ascensor está en el piso cero [la planta baja]). Evidentemente, solo uno de los bits de **Pis_actual[7..0]** vale 1. Por su parte, la señal **Obrir** indica cuando se debe proceder a abrir una puerta del ascensor. Finalmente, **Alliberar** indica que se ha de eliminar una llamada hecha al ascensor previamente por un usuario puesto que esa llamada ya ha sido atendida por el ascensor.

Veamos ahora las señales de salida. **Obrir_porta[7..0]** indica si hay que abrir la puerta del ascensor en uno de los pisos. Cada bit del bus está asociado a la puerta de un piso y si vale 1 significa que esa puerta debe abrirse (si **Obrir_porta[7..0]** es 10000000 eso significa que se ha de abrir la puerta del ascensor del piso 7). Solo uno de los bits de **Obrir_porta[7..0]** puede valer 1. De igual manera, si **Obrir_porta[7..0]** es 00000000 eso significa que las puertas del ascensor en todos los pisos se hallan cerradas.

Alliberar_pis[7..0] indica si se ha de proceder a eliminar una solicitud/llamada previa de un usuario en un piso ya que esta solicitud ya ha sido atendida (el ascensor ya ha ido a ese piso). Obviamente, en ese caso es necesario eliminar ese piso de la lista de pisos solicitados. Cada bit del bus **Alliberar_pis[7..0]** está asociado a un piso. Si **Alliberar_pis[7..0]** es 00000100, eso significa que la solicitud que había hecho un usuario de que el ascensor fuera al piso 2 ya ha sido atendida y, por tanto, se debe proceder a eliminar (liberar) dicha solicitud.

La generación de las señales de salida *Obrir_porta*[7..0] y *Alliberar_pis*[7..0] es sumamente sencilla. Cada bit *i* de los buses *Obrir_porta* y *Alliberar_pis* equivale a:

$$Obrir_porta[i] = Pis_actual[i] \cdot Obrir$$

Si el ascensor está en el piso *i* y la unidad de control da la orden de abrir puerta (*Obrir* = 1), entonces se ha de abrir la puerta del ascensor en el piso *i*; en cualquier otro caso no debe abrirse la puerta del piso *i*.

$$Alliberar_pis[i] = Pis_actual[i] \cdot Alliberar$$

Si el ascensor ha ido al piso *i*, atendiendo una solicitud previa de un usuario, y la unidad de control da la orden de eliminar solicitud (*Alliberar* = 1), entonces se ha de eliminar la solicitud que había hecho el usuario para que el ascensor fuera al piso *i*; en cualquier otro caso no debe eliminarse ninguna solicitud relativa al piso *i*.

El circuito **Obrir_Alliberar** será realizado en dos pasos (será un diseño jerárquico con dos niveles de jerarquía):

1. Construye un submódulo muy sencillo, al que has de llamar **Obrir_Alliberar_1_bit**, que haga las operaciones citadas anteriormente para un valor de *i*. **Obrir_Alliberar_1_bit** tendrá tres entradas, *PA* (de *Pis_actual*), *O* (de *Obrir*) y *A* (de *Alliberar*) y dos salidas *OP* (de *Obrir_Porta*) y *AP* (de *Alliberar_Pis*). El circuito debe implementar las operaciones:

$$OP = O \cdot PA$$

$$AP = A \cdot PA$$

2. Construye el módulo **Obrir_Alliberar** completo, utilizando 8 veces el submódulo **Obrir_Alliberar_1_bit**. Una vez diseñado se deberá compilar para asegurar que no tiene ninguna incoherencia técnica y se deberá simular para verificar que su comportamiento se corresponde con el esperado.

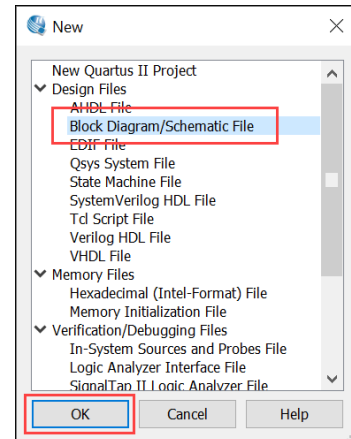
La realización de los dos pasos citados previamente implica la realización de esquemas lógicos, compilación y simulación. En los siguientes puntos del documento se explica cómo llevarlos a cabo utilizando Quartus.

4. QUARTUS: ENTRADA DEL CIRCUITO – EDITOR DE ESQUEMAS

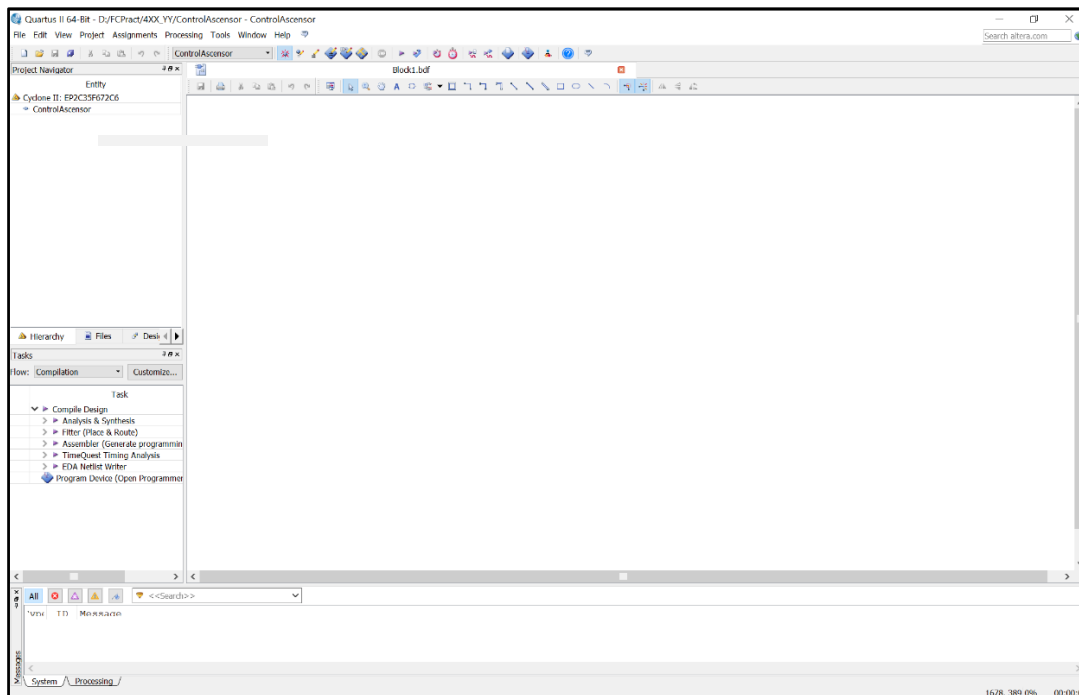
Para entrar el módulo **Obrir_Alliberar_1_bit** en el ordenador utilizaremos un editor de esquemas (también llamado herramienta de captura de esquemas). Para abrir el editor de esquemas haz:

File → New (shortcut: **Ctrl+N**)

En el menú desplegable que aparece, selecciona **Block Diagrams/Schematic File** y clicas en **OK**.



Automáticamente se abrirá una ventana del editor de esquemas:



Sobre el área de trabajo de dicha ventana tendremos que (1) ir introduciendo las puertas lógicas, (2) introducir las entradas y salidas (ports de entrada/salida) y (3) conectar las puertas entre sí y las puertas a las entradas y salidas de acuerdo a las expresiones booleanas citadas en el punto anterior:

$$OP = O \cdot PA \quad AP = A \cdot PA$$

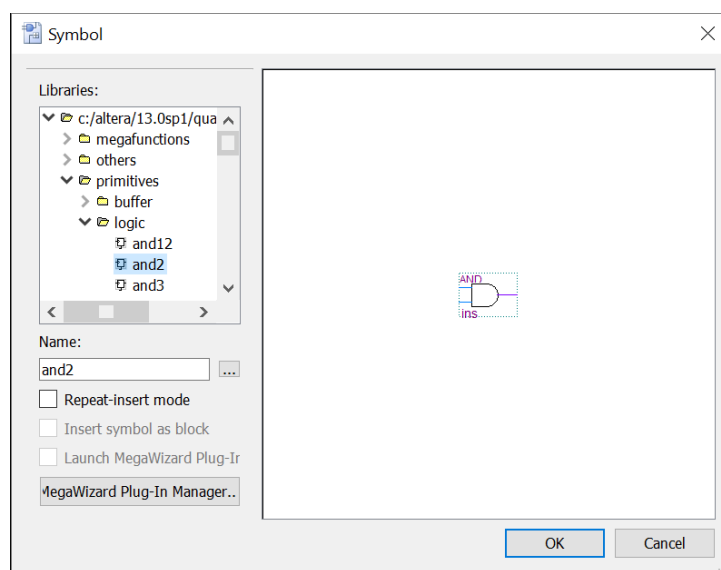
Todo ello se explica cómo hacerlo a continuación.

4.1 INTRODUCCIÓN DE LAS PUERTAS LÓGICAS

El editor de esquemas incluye diversas bibliotecas que contienen puertas y otros dispositivos lógicos. En este caso utilizaremos la biblioteca que lleva por nombre *primitives*. Para introducir una puerta:

1) Haz doble clic (botón izquierdo) sobre el área de trabajo. Se abrirá la ventana de bibliotecas (*Libraries*); clicas en *c:/altera/13.0sp1/quartus/libraries/* y abre la carpeta *primitives*.

Verás que la carpeta contiene 5 subcarpetas: *buffer* (contiene buffers), *logic* (contiene puertas lógicas), *other* (contiene una miscelánea de dispositivos), *pin* (contiene ports de entrada/salida) y *storage* (contiene biestables). Si deseas introducir, por ejemplo, una puerta AND de 2 entradas, abre la carpeta *logic*, selecciona dicha puerta y clicas *OK*.

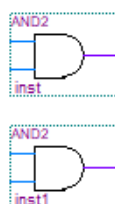


Al clicar *OK* la puerta te aparecerá en el Área de Trabajo del editor de esquemas. Si quieres introducir varias puertas iguales utiliza la opción *Repeat-insert mode* (en este caso deberás pulsar *Esc* para finalizar la introducción).

Si deseas:

- Mover una puerta: Simplemente selecciónala (clicando sobre ella con el botón izquierdo) y arrástrala hasta su destino
- Rotar una puerta: Con el ratón sobre la puerta, clicas con el botón derecho y seleccionas *Flip horizontal*, *Flip Vertical* o *Rotate by Degrees*.

Puesto que el módulo **Obrir_Alliberar_1_bit** está formado por dos puertas AND de dos entradas, procede a añadirlas tal como se muestra en la siguiente figura:

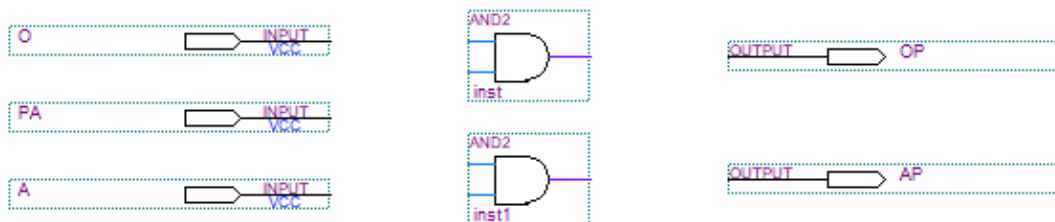


4.2 INTRODUCCIÓN DE LAS ENTRADAS Y SALIDAS

El módulo *Obrir_Alliberar_1_bit* tiene 3 entradas (PA, O y A, todas ellas de un bit) y 2 salidas (OP y AP, también de un bit). Los **ports de entrada/salida están en la biblioteca *primitives* → *pin***. Para introducir un port, moverlo o rotarlo, se sigue el mismo procedimiento explicado para introducir las puertas. Para dar nombre a los ports basta con clicar sobre ellos con el botón derecho del ratón, seleccionar *Properties* e introducir el nombre.


Nota: En general, los ports pueden hacer referencia a **señales de un solo bit** o a **buses** (señales de varios bits). En el caso de los buses, en el nombre se ha de indicar entre corchetes los índices de los bits (p. e., $Z[7..0]$ denotaría un bus llamado Z formado por ocho bits, indexados desde el 0 al 7; siendo el bit 0 el de menos peso).

Procede a añadir las entradas y salidas al circuito **Obrir_Alliberar_1_bit**, tal como se observa en la siguiente imagen:

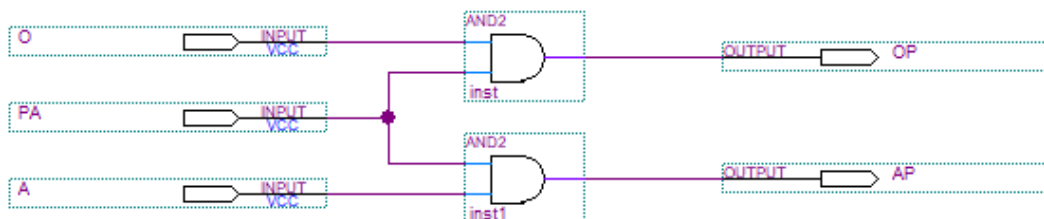


4.3 CONEXIÓN DE PUERTAS, ENTRADAS Y SALIDAS

Finalmente, solo falta conectar puertas, entradas y salidas. De manera general, las conexiones pueden consistir en **conexiones simples de un solo bit** o en **buses**. En la celda **Obrir_Alliberar_1_bit** todas las conexiones son conexiones simples de un solo bit (más adelante veremos cómo se realizan las conexiones de buses).

Para conectar dos puntos con una conexión simple, clicas en el símbolo  (*Orthogonal Node Tool*) de la botonera situada encima del área de edición, clicas con el botón izquierdo sobre el punto de inicio de la conexión y, manteniendo el botón apretado, mueves el ratón hasta el otro extremo de la conexión.

Una vez realizadas las conexiones, el circuito queda como se observa en la siguiente figura:

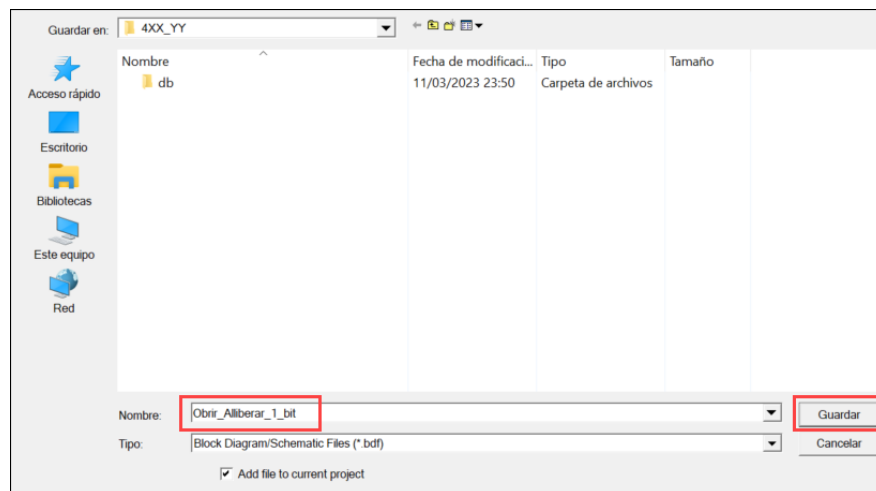


4.4 GRABACIÓN DEL MÓDULO

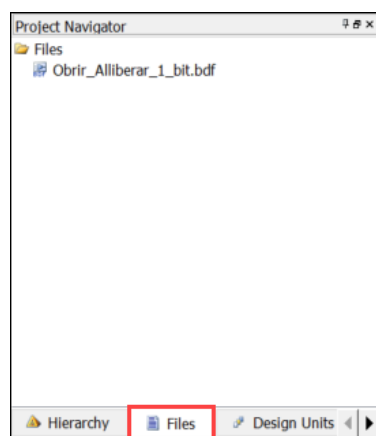
Al llegar a este punto ya has concluido el módulo **Obrir_Alliberar_1_bit**. Ahora debes guardar el esquema ejecutando:

File → Save (shortcut: **Ctrl+S**)

Se abrirá un formulario en el que **debes seleccionar la carpeta del proyecto (U:\FCPract\4XX_YY)**, indicar el nombre del módulo, en este caso **Obrir_Alliberar_1_bit.bdf**, y clicar en **Guardar**.



Si clicas en el recuadro *Project Navigator* (a la izquierda del área de trabajo) en la pestaña *Files* verás que te ha aparecido el fichero *Obrir_Alliberar_1_bit.bdf*.

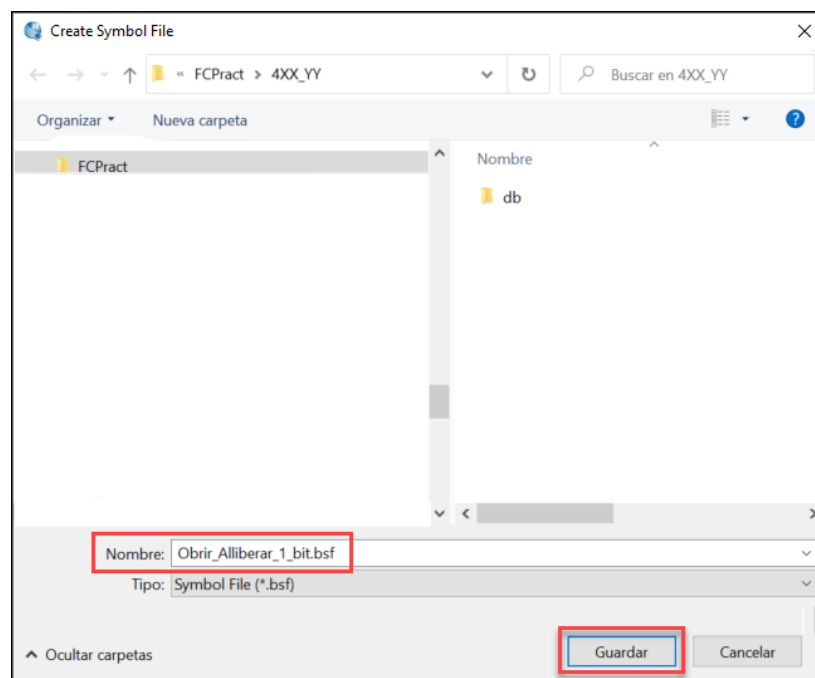


4.5 CREACIÓN DE UN SÍMBOLO PARA EL MÓDULO

Una vez hayas grabado el esquemático **Obrir_Alliberar_1_bit**, construiremos un símbolo para que, más adelante, podamos utilizar este módulo como si fuese una celda de biblioteca. Para ello basta ejecutar el comando:

File → Create/Update → Create Symbol Files for Current File

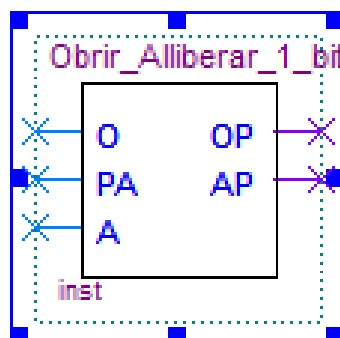
En el formulario que se abre **selecciona la carpeta del proyecto (U:\FCPract\4XX_YY)**, indica como nombre del fichero **Obrir_Alliberar_1_bit.bsf** y clicas en **Guardar**.



El símbolo ya se ha creado. Si quieres verlo, abre el fichero **Obrir_Alliberar_1_bit.bsf** con el comando:

File→Open→Indica como Tipo "Graphic Files" y selecciona el fichero Obrir_Alliberar_1_bit.bsf

En la ventana que se abre puedes observar el símbolo creado.



4.6 CREACIÓN DEL MÓDULO OBRIR_ALLIBERAR

Una vez tenemos el submódulo **Obrir_Alliberar_1_bit**, ya podemos crear el módulo **Obrir_Alliberar**, el cual contendrá 8 instancias del submódulo **Obrir_Alliberar_1_bit**. Cada una de esas instancias estará vinculada a un piso y será la encargada de generar el bit i de los buses de salida $Obrir_porta[7..0]$ y $Alliberar_pis[7..0]$, a partir del bit i del bus de entrada $Pis_actual[7..0]$ y de las señales $Obrir$ y $Alliberar$.

4.6.1 INTRODUCCIÓN DE LAS 8 INSTANCIAS DE OBRIR_ALLIBERAR_1_BIT

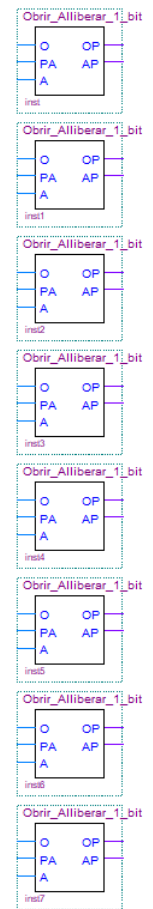
Empieza creando un esquema lógico vacío ejecutando el comando:

File → New (shortcut: **Ctrl+N**)

En el menú desplegable que aparece, selecciona **Block Diagrams/Schematic File** y clic en **OK**.

En la ventana de edición vacía que se abre, haz doble clic (botón izquierdo) sobre el área de trabajo. Se abrirá la ventana de bibliotecas (*Libraries*); clic en *Project*, selecciona *Obrir_Alliberar_1_bit*, marca la casilla *Repeat-insert mode* y clic OK.

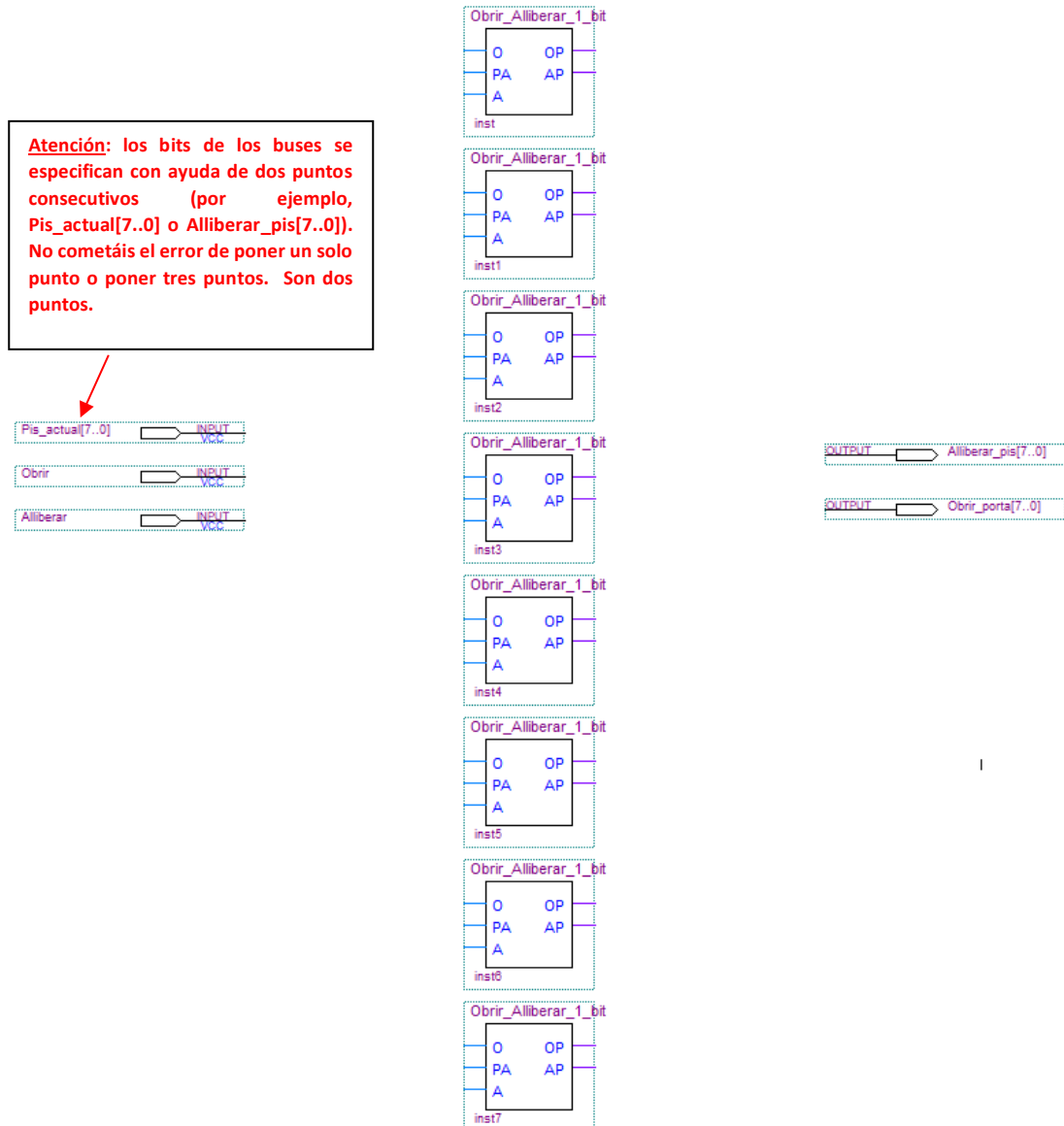
Posiciona 8 instancias del símbolo de **Obrir_Alliberar_1_bit**, una debajo de otra tal como se muestra en la siguiente imagen. Pulsa **Esc** para interrumpir la introducción repetida de instancias.



4.6.2 INTRODUCCIÓN DE LAS ENTRADAS Y SALIDAS

Como ya se indicó, el módulo **Obrir_Alliberar** tiene como entradas *Pis_actual[7..0]* (que es un bus de 8 bits), *Obrir* y *Alliberar*. Como salidas tiene *Alliberar_pis[7..0]* y *Obrir_porta[7..0]* (ambas son buses de 8 bits).

De igual manera, que ya se hizo en el módulo **Obrir_Alliberar_1_bit** se introducirán los ports de entrada y salida en el diseño, tal como se muestra en la siguiente imagen.



4.6.3 CONEXIÓN DE PUERTAS, ENTRADAS Y SALIDAS

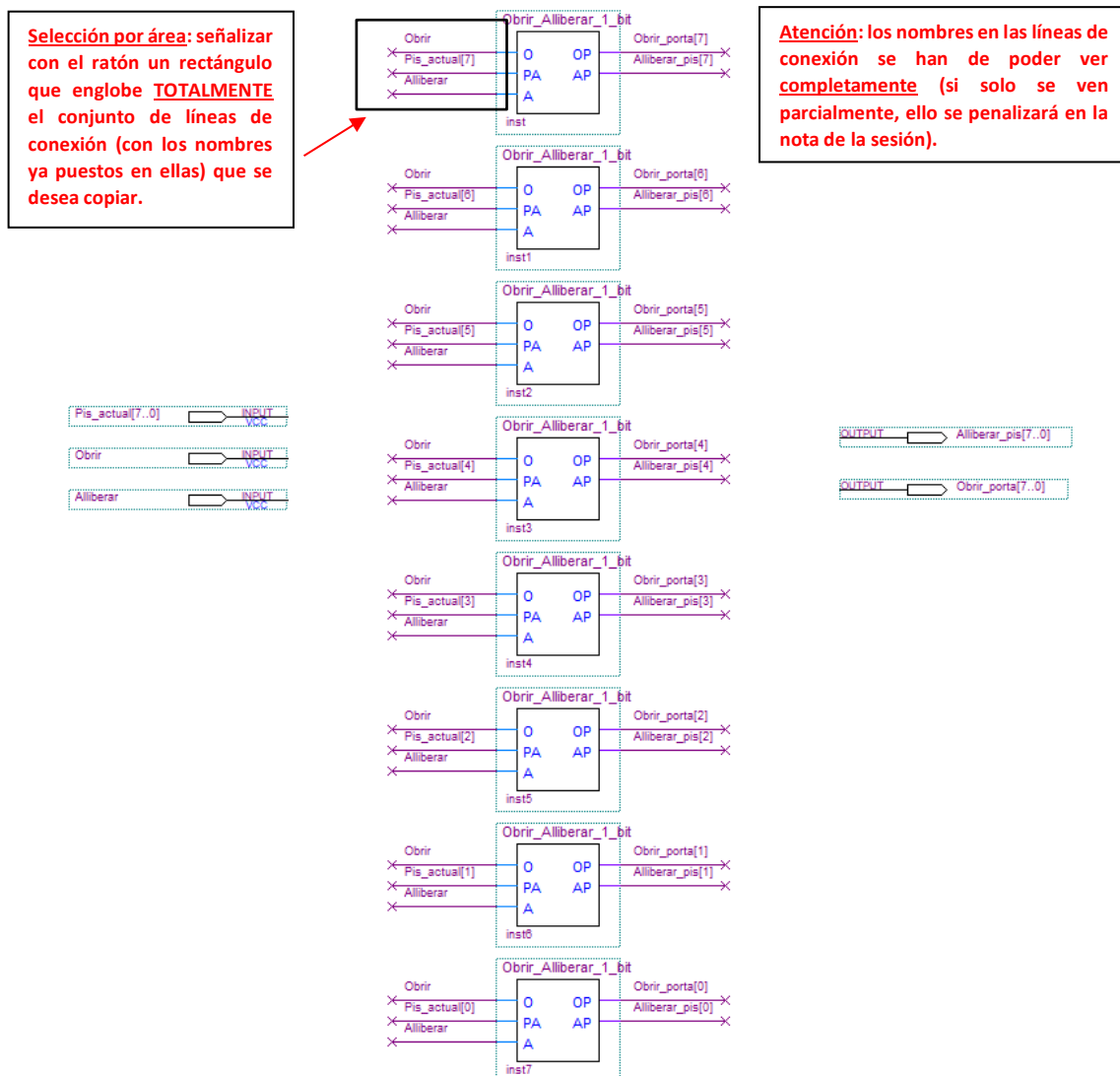
En este caso vamos a establecer las conexiones entre puertas, entradas y salidas utilizando una posibilidad muy interesante que ofrece Quartus: la **conectividad por nombre**.

Dos puntos del circuito pueden **conectarse por nombre**. Para ello basta con que los dos extremos/líneas que debe unir la conexión tengan el mismo nombre, sin necesidad de que las líneas estén conectadas. Para dar nombre a una línea se procede de la misma manera que para

dar nombre a un port: se selecciona y se edita sus propiedades. El uso de la conectividad por nombre es una opción muy recomendable porque lleva a esquemas más claros donde posteriormente es más fácil localizar los eventuales errores. Asimismo, cuando se desea **conectar una línea concreta de un bus a otro punto** es obligatorio realizar una conexión por nombre.

En la siguiente imagen se observa cómo sería el conexionado del módulo utilizando para ello la **conectividad por nombre**. Se puede ver como cada bit del bus de entrada *Pis_actual[7..0]* está conectado a la entrada *PA* de cada una de las instancias de **Obrir_Alliberar_1_bit**. Por su parte la entrada *Obrir* está conectada a todas las entradas *O* y la entrada *Alliberar* está conectada a todas las entradas *A* de las instancias. Finalmente, se puede ver que cada bit de los buses de salida *Alliberar_pis[7..0]* y *Obrir_porta[7..0]* son generados por las salidas *AP* y *OP* de cada instancia de *Obrir_Alliberar_1_bit*, respectivamente.

Se puede observar que aparecen muchas líneas de conexión con nombres repetidos o muy similares. Para agilizar la entrada de las conexiones se aconseja “copiar y pegar” grupos de líneas de conexión con el nombre ya puesto en ellas y posteriormente modificar los nombres que sea necesario. (Nota: Para copiar un grupo de líneas, primero hay que seleccionarlo. Se recomienda usar la selección por área, señalizando con el ratón un rectángulo que englobe **totalmente** el grupo).



4.6.4 GRABACIÓN DEL MÓDULO Y CREACIÓN DE UN SÍMBOLO

Puesto que ya has concluido el esquema lógico, debes guardar el esquema ejecutando:

File → Save

(shortcut: **Ctrl+S**)

Se abrirá un formulario en el que **debes seleccionar la carpeta del proyecto (U:\FCPract\4XX_YY)**, indicar el nombre del módulo, en este caso **Obrir_Alliberar.bdf**, y clicar en **Guardar**.

Finalmente, genera un símbolo para que, más adelante, puedas utilizar este módulo en la creación de otro módulo más complejo que lo contenga. Para ello basta ejecutar el comando:

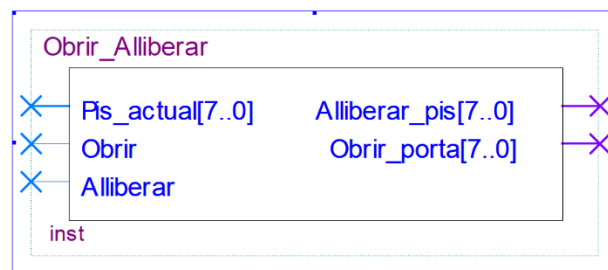
File→Create/Update→Create Symbol Files for Current File

En el formulario que se abre **selecciona la carpeta del proyecto (U:\FCPract\4XX_YY)**, indica como nombre del fichero **Obrir_Alliberar.bsf** y clicla en *Guardar*.

Si quieres ver el símbolo de **Obrir_Alliberar**, abre el fichero **Obrir_Alliberar.bsf** con el comando:

File→Open→Indica como Tipo "Graphic Files" y selecciona el fichero Obrir_Alliberar.bsf

En la ventana que se abre puedes observar el símbolo creado.

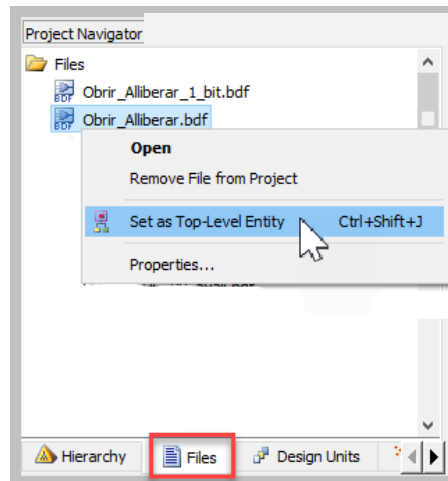


4.7 COMPILAR EL CIRCUITO

El compilador analiza el esquema y “traslada virtualmente” (el volcado real se hace mediante el módulo “programador” que veremos más adelante) el circuito a la FPGA, definiendo qué elementos lógicos de la FPGA utilizará y cómo quedará definida la matriz de interconexionado.

Para compilar el circuito necesitamos:

1) Definirlo como *Top_level entity*. Para ello (1) en la sección **Files** del **Project Navigator** clicas con el botón derecho del ratón sobre el fichero **Obrir_Alliberar.bdf** y (2) selecciona ‘Set as Top-Level Entity’.



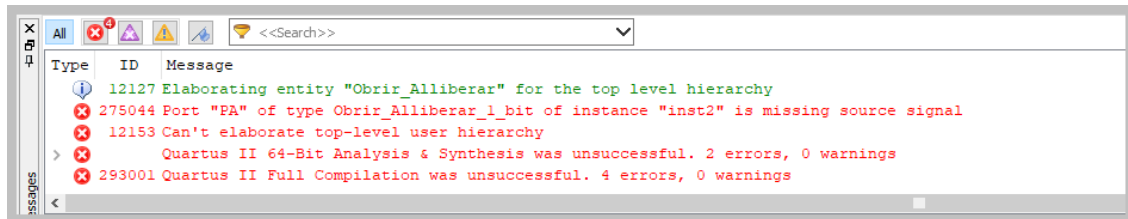
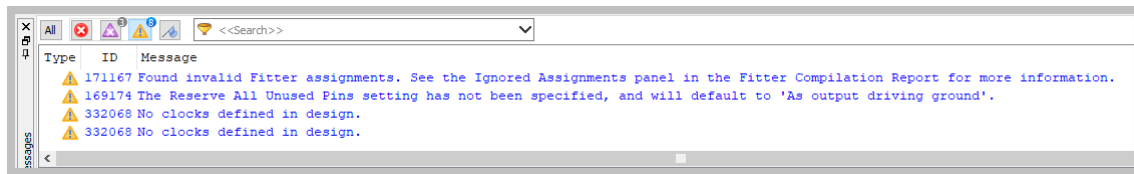
2) Lanzar el compilador clicando el botón indicado por la flecha:



(también lo puedes hacer mediante el comando **Processing → Start Compilation**)

Durante la compilación te irán apareciendo mensajes en la parte inferior de la pantalla (ver imagen al inicio de la siguiente página):

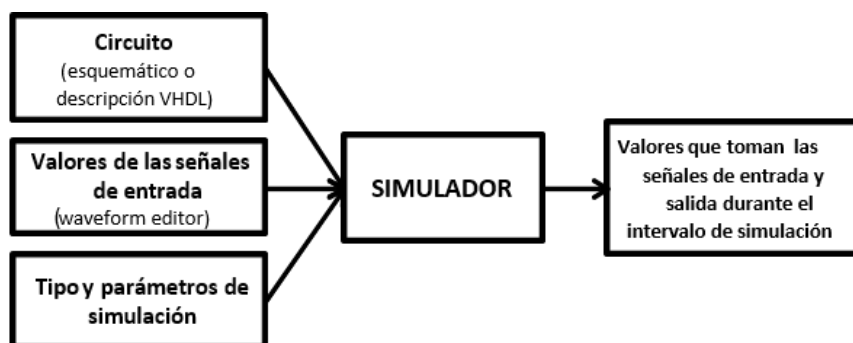
- Los mensajes en verde contienen información de los tiempos de propagación y de las capacidades. No nos han de preocupar en absoluto.
- **Warnings**, mensajes en azul: son avisos de problemas en el esquema que normalmente no ponen en peligro el funcionamiento del circuito.
- **Errors**, en rojo: son avisos de errores que **obligatoriamente deben resolverse** para poder continuar.



5. SIMULAR EL FUNCIONAMIENTO DEL MÓDULO *OBIR_ALLIBERAR*

5.1 INTRODUCCIÓN

El **simulador** es una herramienta software que calcula y visualiza los valores que tomarán las salidas del circuito en respuesta a un conjunto de señales de entrada que define el diseñador. **Eso le permite saber al diseñador si el circuito se comporta de la forma esperada o, por el contrario, no (lo cual sería un indicador de que en el circuito hay errores que deberán ser corregidos).**



Como ves en la figura previa, el simulador necesita que le suministremos (1) una descripción del circuito que queremos simular, ya sea como esquemático, ya sea a través de un lenguaje de descripción hardware, (2) los valores de las entradas, a través de un “editor de ondas” o “editor de estímulos” (*waveform editor*), y (3) ciertos parámetros que definirán el tipo y características de la simulación que queremos realizar (cuál es el fichero de formas de onda a utilizar, qué tipo de simulación queremos hacer, cuál es el intervalo de tiempo que queremos simular, etc.)

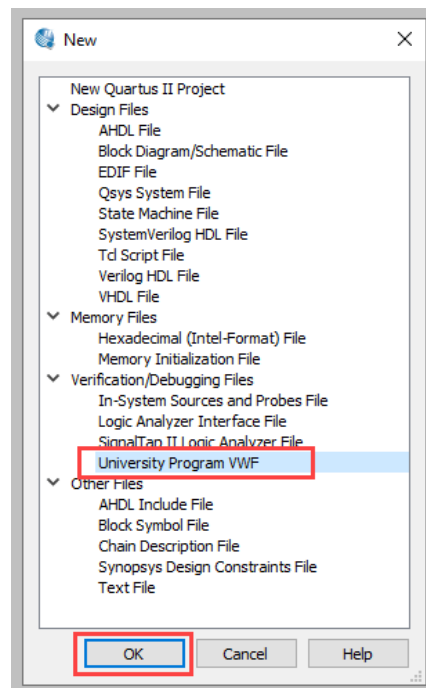
Quartus II permite realizar dos tipos de simulaciones: la simulación funcional (*Functional simulation*), que simula el funcionamiento del circuito sin tener en cuenta tiempos de respuesta, y la simulación temporal (*Timing simulation*), que sí los tiene en cuenta y, por tanto, proporciona resultados más ajustados a los reales. La simulación funcional es más sencilla de interpretar, por lo que lo que se hace habitualmente es realizar primero una simulación funcional y, una vez comprobado que el circuito es correcto (desde el punto de vista lógico), realizar una simulación temporal para asegurar que los retardos de las puertas y el resto de los dispositivos no provocan ningún error de funcionamiento.

5.2 EDITOR DE FORMAS DE ONDA (WAVEFORM EDITOR)

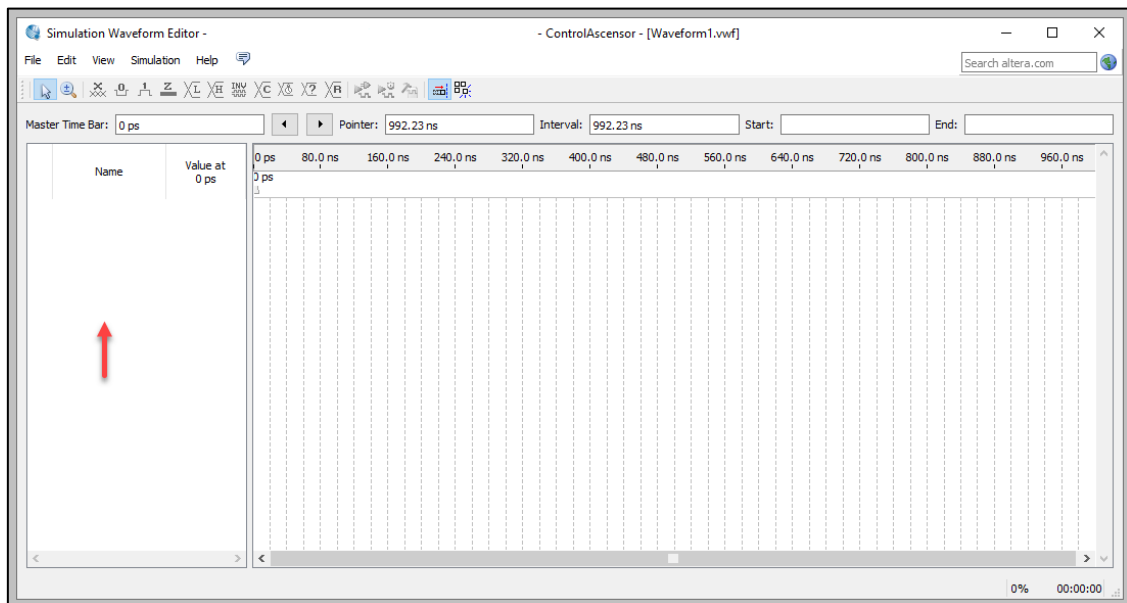
Lo primero que hay que hacer cuando queremos simular un esquemático es generar el llamado “fichero de vectores de simulación” (también llamado “fichero de estímulos” o “fichero de formas de onda”). Para ello, ejecuta la orden:

File → New

En el menú desplegable que aparece, selecciona **University Program VWF** y clicla en **OK**.



Como consecuencia del comando, se abre la ventana del editor de formas de onda:

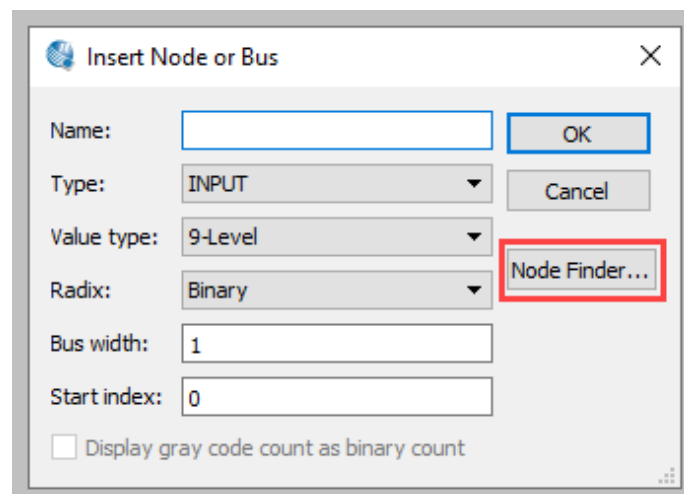


Por defecto, el editor de formas de onda trabaja con franjas temporales de 10 ns (ver las líneas verticales discontinuas que aparecen en el área de edición) y con un tiempo de simulación de 1 μ s. Para la simulación del módulo Obrir_Alliberar no requeriremos de hacer una simulación tan larga. Con **500 ns** será suficiente. Asimismo, el uso de franjas temporales de **20 ns** resultará más adecuado. Por ello, se modificará ambos parámetros con los comandos:

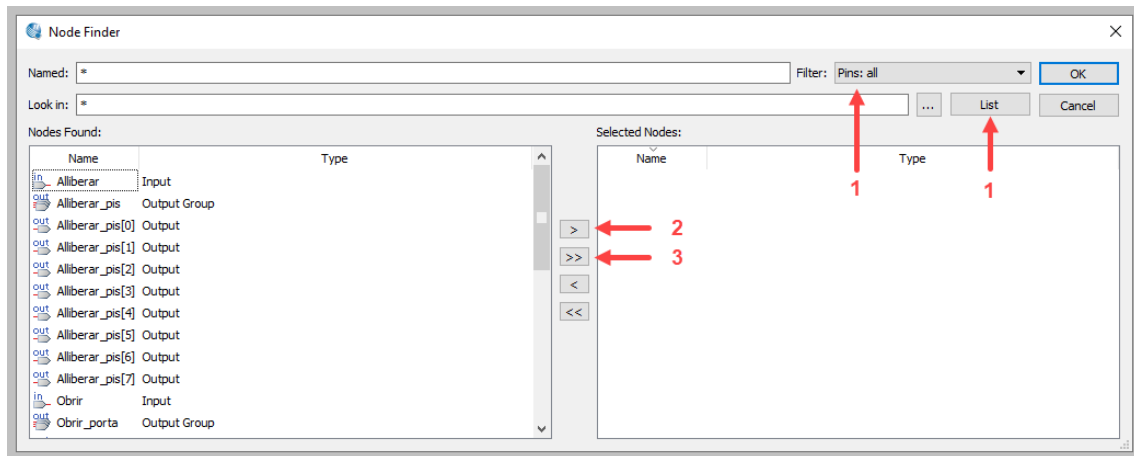
Edit → Grid size... → 20 ns

Edit → Set End Time... → 500 ns

Para introducir las entradas y sus valores, haz doble clic donde marca la flecha en la figura anterior. Te aparecerá el siguiente formulario en el cual has de clicar en **Node Finder...**.



Se abrirá una nueva ventana desde donde podrás seleccionar los nodos de entrada (*Pis_actual*, *Obrir* y *Alliberar*) y los nodos de salida (*Alliberar_pis* y *Obrir_porta*):



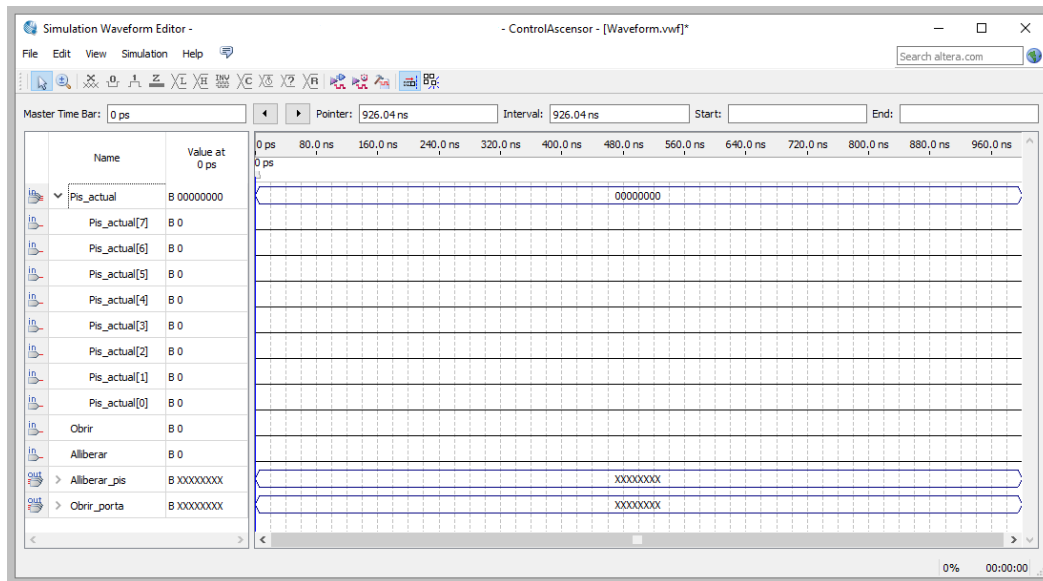
Selecciona la opción “**Pins: all**” en Filter, y clicas en “**List**” (flechas 1). En la sub-ventana “**Nodes Found**” aparecerán todos los nodos de entrada y salida de *Obrir_Alliberar*. Selecciona el nodo *Pis_actual* y clicas en > (flecha 2). Verás que el nodo seleccionado te aparece en la ventana de la derecha (**Selected nodes**). Repite el mismo proceso para los nodos *Obrir*, *Alliberar*, *Alliberar_pis* y *Obrir_porta*. Con ello ya tendrás seleccionados todos los nodos que serán visualizados en la simulación.

Nota: En general, para visualizar los buses puedes optar entre seleccionar el bus entero (p. e., *Pis_actual*) o seleccionar cada uno de los bits del bus (p. e., *Pis_actual*[7], *Pis_actual*[6],...). No selecciones nunca a la vez el nombre del bus y todas sus líneas porque en ese caso se generan un montón de *warnings* diciendo que has seleccionado la misma línea más de una vez. El icono >> (flecha 3) sirve para visualizar todos los nodos encontrados.

Una vez seleccionados los nodos que te interesan, clicas en **OK**. Te volverá a aparecer la ventana **Insert Node or Bus**. En el campo **Radix** selecciona **Binary** (de esa forma el valor de los buses se mostrará en binario en la ventana de simulación) y vuelve a clicar en **OK**. Los nodos se incorporarán a la ventana de simulación.

Vuelve al simulador y verás todos los nodos de entrada y de salida. *Pis_actual* y *Alliberar_pis* y *Obrir_porta*, al ser buses, se visualizan en modo compacto. Para desplegarlos basta con clicar en el icono > que acompaña al nombre del bus.

En la figura de la siguiente página puedes ver los nodos en el simulador. Como ejemplo, hemos desplegado los 8 bits del bus *Pis_actual*[7..0]. Observa que todas las entradas están a 0 y todas las salidas están fijadas a un valor indeterminado.



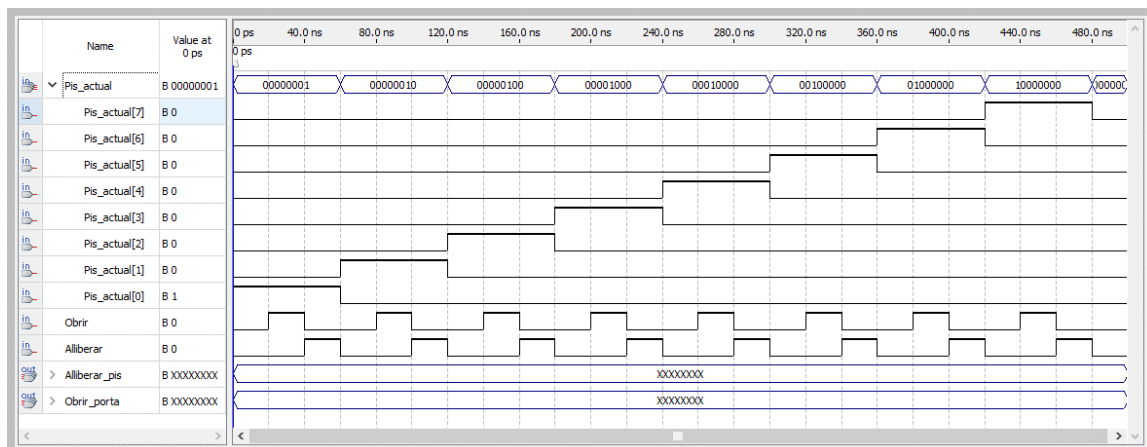
Lo siguiente que hemos de hacer es definir el valor de las entradas en cada instante de tiempo.

Estrategia de test

Para comprobar que el circuito funciona haremos lo siguiente. Se establecerá inicialmente que el ascensor está en el piso 0 ($Pis_actual = 00000001$), después el ascensor sube al piso 1 ($Pis_actual = 00000010$), de ahí al piso 2 ($Pis_actual = 00000100$) y así sucesivamente hasta llegar al piso 7 ($Pis_actual = 10000000$). Mientras que el ascensor está en cada uno de los pisos se comprobará que:

- Si la entrada *Obrir* vale 1, el bit de la salida *Obrir_porta* correspondiente a ese piso se pone a 1; en caso contrario, se mantiene a 0.
- Si la entrada *Alliberar* vale 1, el bit de la salida *Alliberar_pis* correspondiente a ese piso se pone a 1; en caso contrario, se mantiene a 0.

La figura muestra qué estímulos de entrada habrán de ser definidos. Se puede ver como cuando el ascensor está en cada piso, las señales *Obrir* y *Alliberar* adoptan los valores 00, 10 y 01.



En el siguiente punto se explica cómo introducir los valores de los diversos nodos de entrada.

¿Cómo entrar los valores a los nodos de entrada?

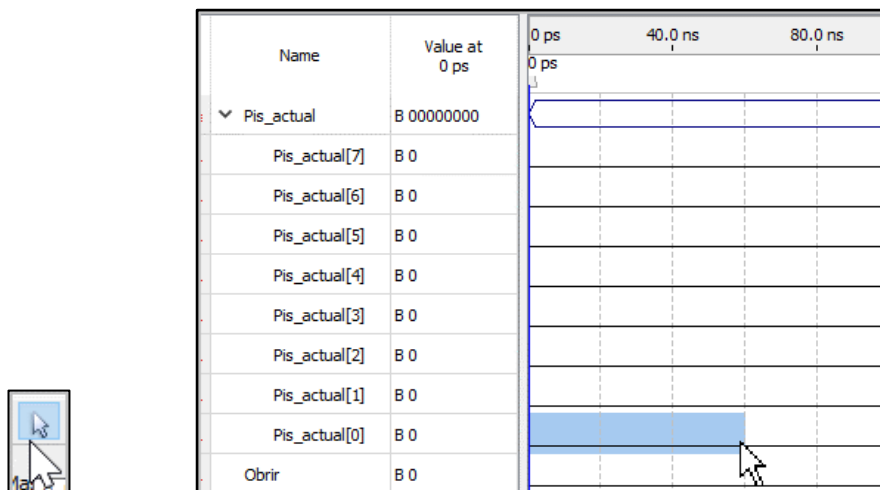
Ahora que ya sabemos qué queremos comprobar, la pregunta es ¿cómo pongo los valores de las entradas a 0 o a 1 en cada instante de tiempo? O, dicho de otra manera, ¿cómo entro las formas de onda?

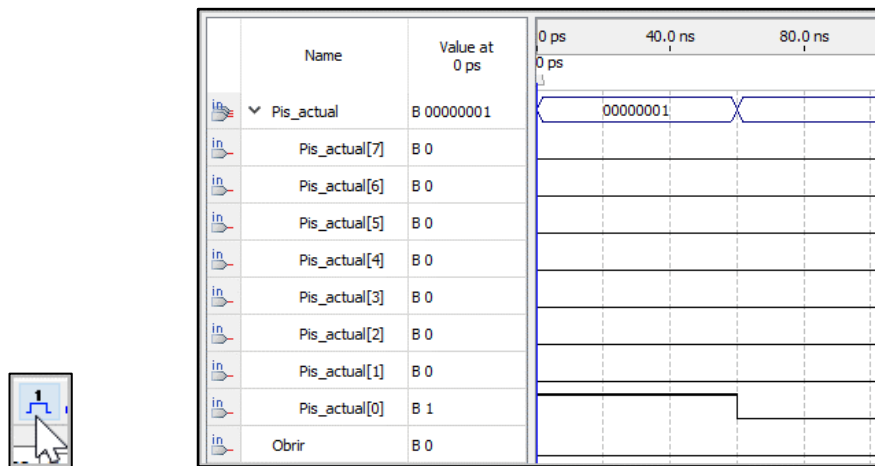
En la parte superior de editor de formas de onda verás el siguiente menú con la mayoría de comandos que se requieren para introducir los valores de las entradas:



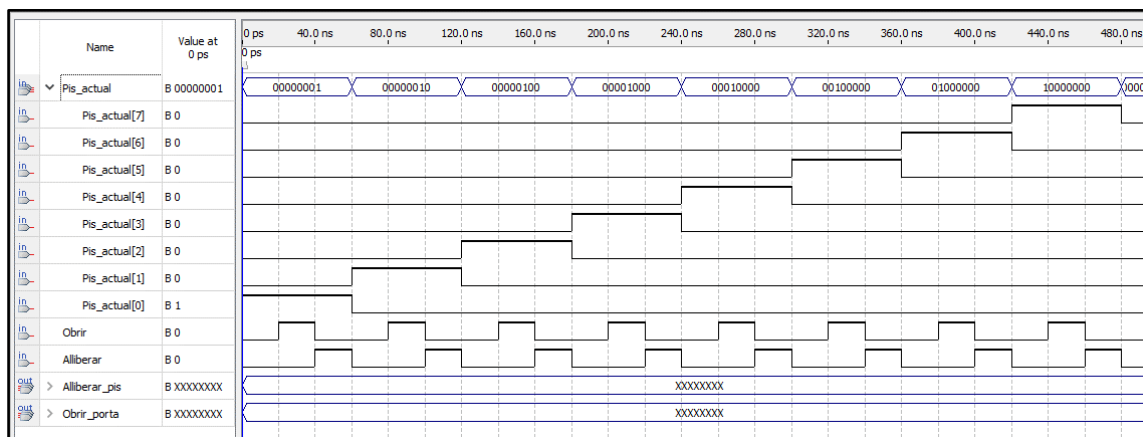
- 1: Selecciona el intervalo de tiempo de una señal de entrada en el que se desea forzar un valor
- 2: Zoom
- 3: Fuerza un 0 en el intervalo seleccionado
- 4: Fuerza un 1 en el intervalo seleccionado
- 5: Genera una señal periódica (señal de reloj)
- 6: "Snap to grid". Si está activa, la zona seleccionada se ajusta a un intervalo de tiempo de los marcados (mediante líneas verticales) en el área de edición de formas de onda.

Por ejemplo, para forzar un 1 en un intervalo de 60 ns en la señal *Pis_actual[0]*, clics en el icono marcado por la flecha 1, selecciona el intervalo de 60 ns con el botón izquierdo, y clics a continuación en el icono marcado por la flecha 4 (ver imágenes en esta y la siguiente página).





Siguiendo el procedimiento indicado, introduce los valores correspondientes a las entradas *Pis_actual*, *Obrir* y *Alliberar* que habían sido definidos al establecer la estrategia de test.



Finalmente, guarda el fichero de simulación:

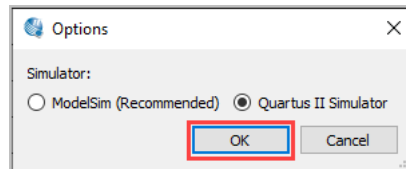
File → Save (shortcut: **Ctrl+S**)

En el formulario que se abre **selecciona la carpeta del proyecto (U:\FCPract\4XX_YY)** e introduce como nombre del fichero **Obrir_Alliberar.vwf**.

Simulación del circuito

En este punto ya estamos en condiciones de simular el circuito. En primer lugar, seleccionaremos el simulador propio de Quartus II (*qsim*) como el simulador que vamos a utilizar:

Simulation → Options → Seleccionar Quartus II Simulator → Pulsar OK

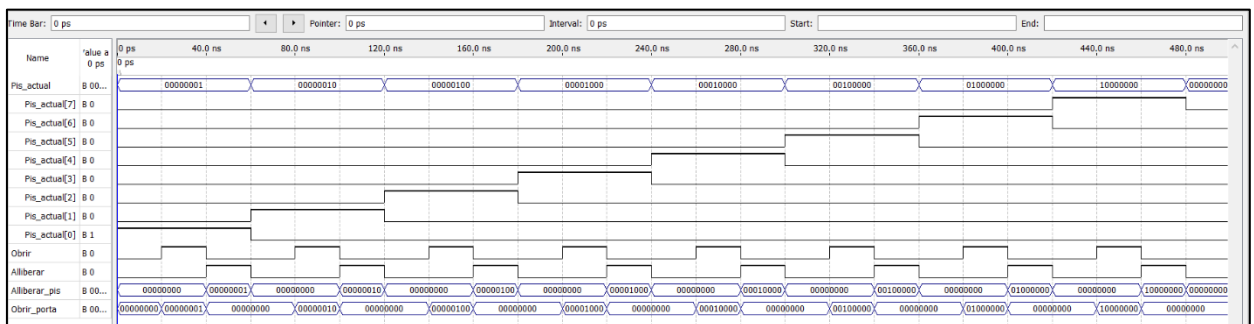


En este caso vamos a realizar una **simulación funcional**, es decir, una simulación en la que las que se considera que las puertas y otros dispositivos se comportan idealmente sin tener ningún retardo asociado. Para ello, basta con clicar, en la barra superior, en el icono que muestra la figura siguiente:



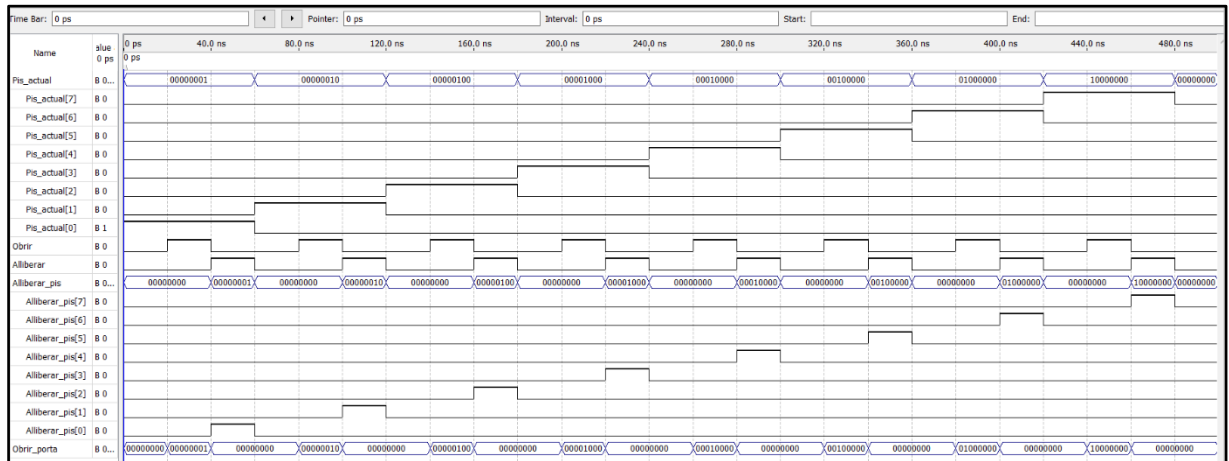
(también lo puedes hacer mediante el comando **Simulation → Run functional Simulation**)

El resultado de la simulación aparece en una ventana como la siguiente, en la que puedes agrandar o reducir la escala de tiempos con el icono de zoom, y en la que puedes moverte a lo largo de la escala de tiempo con la barra de la parte inferior.

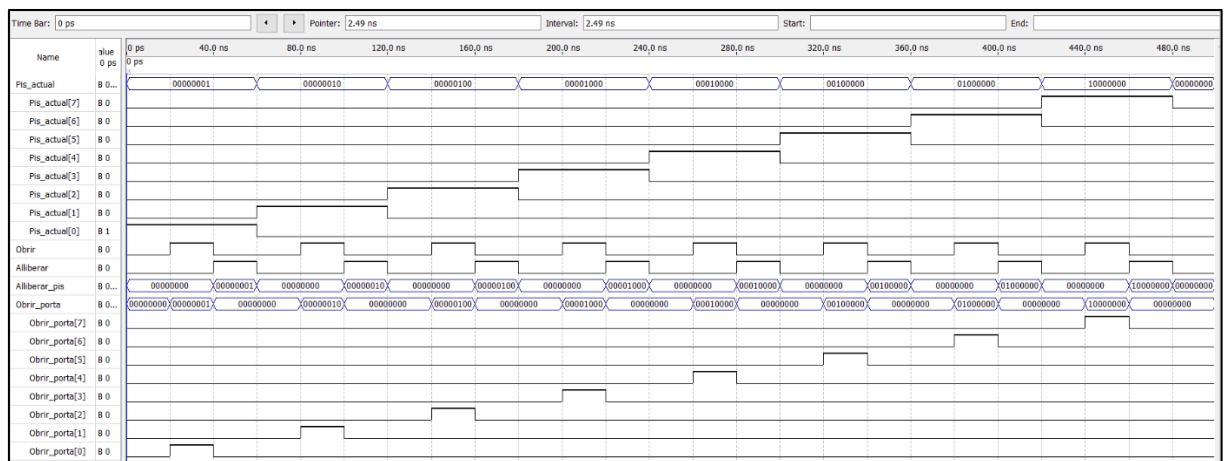


Lo único que falta por hacer es comprobar que los valores de las salidas que da el simulador coinciden con los valores esperados en todos los casos (en la siguiente página te doy algunas indicaciones al respecto). Si no es así, es que el circuito es incorrecto. Identificar las condiciones en las que las salidas no coinciden con las esperadas te dará pistas de donde está el error.

Para comprobar que el resultado es correcto te aconsejo que primero compruebes el correcto funcionamiento de la salida *Alliberar_pis[7..0]* (cada bit correspondiente a un piso solo puede valer 1 cuando el ascensor esta en ese piso y la señal *Alliberar* vale 1).



A continuación, comprueba el correcto comportamiento de la salida *Obrir_porta[7..0]* (cada bit correspondiente a un piso solo puede valer 1 cuando el ascensor está en ese piso y la señal *Obrir* vale 1).




Cuando acabes la simulación, guarda sus resultados ejecutando el comando:

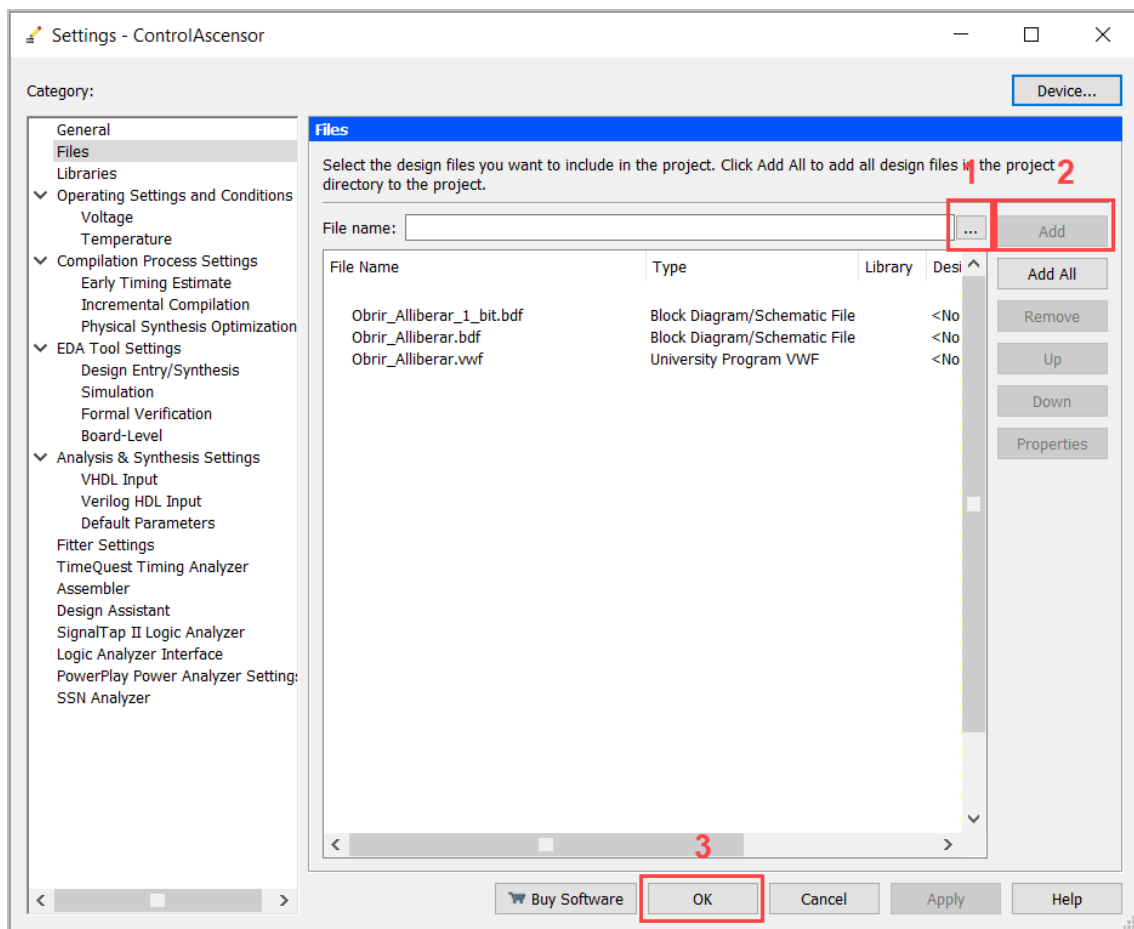
File → Save As ...

En el formulario que aparece **selecciona la subcarpeta simulation\qsim del proyecto (U:\FCPract\4XX_YY\simulation\qsim)** e indica como nombre del fichero **Obrir_Alliberar_sim.vwf**. Finalmente pulsa el botón **Abrir**.

El fichero con el resultado de la simulación (**Obrir_Alliberar_sim.vwf**) no se considera que forme parte del proyecto y por ese motivo no aparece en la sección *Files* del *Project Navigator*. Para añadirlo al proyecto se ejecutará el comando:

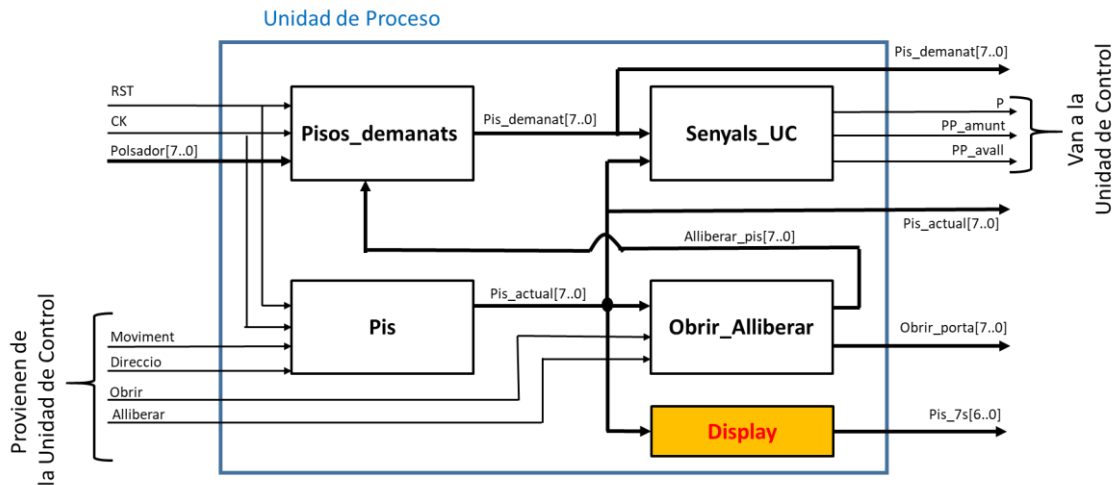
Project → Add/Remove Files in Project...

- (1) Pulsa el botón  . En la ventana que se abre sitúate en la subcarpeta **U:\FCPract\4XX_YY\simulation\qsim** y selecciona el fichero **Obrir_Alliberar_sim.vwf** (atención: para poderlo ver tendrás que escoger el filtro **All files(*.*)**). Finalmente, pulsa el botón **Abrir**.
- (2) Pulsa el botón **Add**.
- (3) Pulsa el botón **OK**.



Si has hecho correctamente el proceso ahora se debería ver el fichero **simulation/qsim/Obrir_Alliberar_sim.vwf** en la sección *Files* del *Project Navigator*.

6. EL MÓDULO *DISPLAY*

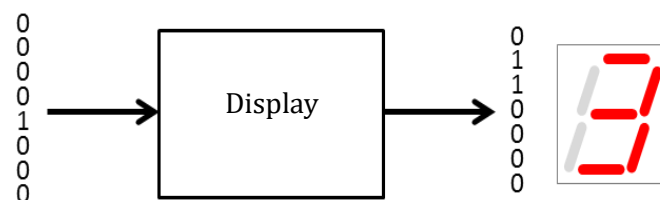


El módulo **Display** visualiza sobre un 7-segmentos el piso actual. Acepta como entrada un vector (un bus) de 8 bits, $Pis_actual[7..0]$, y genera 7 salidas que activan cada una de ellas uno de los segmentos del 7-segmentos. Las salidas son un vector (bus) de 7 bits $Pis_7s[6..0]$.



Como ya sabéis, los bits del Pis_actual están todos a 0 salvo el correspondiente al piso en el que se encuentra el ascensor, que toma el valor 1. Por ejemplo, si el ascensor está en el piso 0 (planta baja), $Pis_actual[7..0] = 00000001$; si se encuentra en el piso 1, $Pis_actual[7..0] = 00000010$; si se encuentra en el piso 2, $Pis_actual[7..0] = 00000100$, y así sucesivamente.

Los segmentos (LEDs) del 7-segmentos de la placa de prototipado Altera DE2 funcionan con lógica negativa; es decir, para que un LED se ilumine es necesario que reciba un 0. Si recibe un 1, el LED se mantiene apagado. Por ejemplo, si el ascensor está en el piso 3 ($Pis_actual[7..0] = 00001000$), se tienen que iluminar los LEDs 0, 1, 2, 3 y 6 y, por tanto, $Pis_7s[6..0]$ debe tomar el valor 0110000:



Se ha utilizado la utilidad *Proyecto→AnalizarCircuito* de VerilUOC_Desktop para crear la tabla de verdad del módulo **Display** y para simplificar las funciones resultantes. Las funciones booleanas que describen el módulo **Display** son las siguientes:

$$\begin{aligned}Pis_7s[6] &= \overline{Pis_actual[6]} \cdot \overline{Pis_actual[5]} \cdot \overline{Pis_actual[4]} \cdot \overline{Pis_actual[3]} \cdot \overline{Pis_actual[2]} = \\&\quad \overline{Pis_actual[6] + Pis_actual[5] + Pis_actual[4] + Pis_actual[3] + Pis_actual[2]} \\Pis_7s[5] &= \overline{Pis_actual[6]} \cdot \overline{Pis_actual[5]} \cdot \overline{Pis_actual[4]} \cdot Pis_actual[0] = \\&\quad \overline{Pis_actual[6] + Pis_actual[5] + Pis_actual[4] + Pis_actual[0]} \\Pis_7s[4] &= \overline{Pis_actual[6]} \cdot \overline{Pis_actual[2]} \cdot \overline{Pis_actual[0]} = \\&\quad \overline{Pis_actual[6] + Pis_actual[2] + Pis_actual[0]} \\Pis_7s[3] &= \overline{Pis_actual[6]} \cdot \overline{Pis_actual[5]} \cdot \overline{Pis_actual[3]} \cdot \overline{Pis_actual[2]} \cdot \overline{Pis_actual[0]} = \\&\quad \overline{Pis_actual[6] + Pis_actual[5] + Pis_actual[3] + Pis_actual[2] + Pis_actual[0]} \\Pis_7s[2] &= Pis_actual[2] \\Pis_7s[1] &= \overline{Pis_actual[7]} \cdot \overline{Pis_actual[4]} \cdot \overline{Pis_actual[3]} \cdot \overline{Pis_actual[2]} \cdot \overline{Pis_actual[1]} \cdot \overline{Pis_actual[0]} = \\&\quad \overline{Pis_actual[7] + Pis_actual[4] + Pis_actual[3] + Pis_actual[2] + Pis_actual[1] + Pis_actual[0]} \\Pis_7s[0] &= \overline{Pis_actual[7]} \cdot \overline{Pis_actual[6]} \cdot \overline{Pis_actual[5]} \cdot \overline{Pis_actual[3]} \cdot \overline{Pis_actual[2]} \cdot \overline{Pis_actual[0]} = \\&\quad \overline{Pis_actual[7] + Pis_actual[6] + Pis_actual[5] + Pis_actual[3] + Pis_actual[2] + Pis_actual[0]}\end{aligned}$$

A partir de este punto se trata simplemente de dibujar el esquemático, entrarlo en Quartus y simularlo para comprobar su correcto funcionamiento.

6.1 REALIZAR EL ESQUEMÁTICO DEL MÓDULO

Para entrar el esquemático del módulo **Display** crea un nuevo módulo ejecutando el comando:

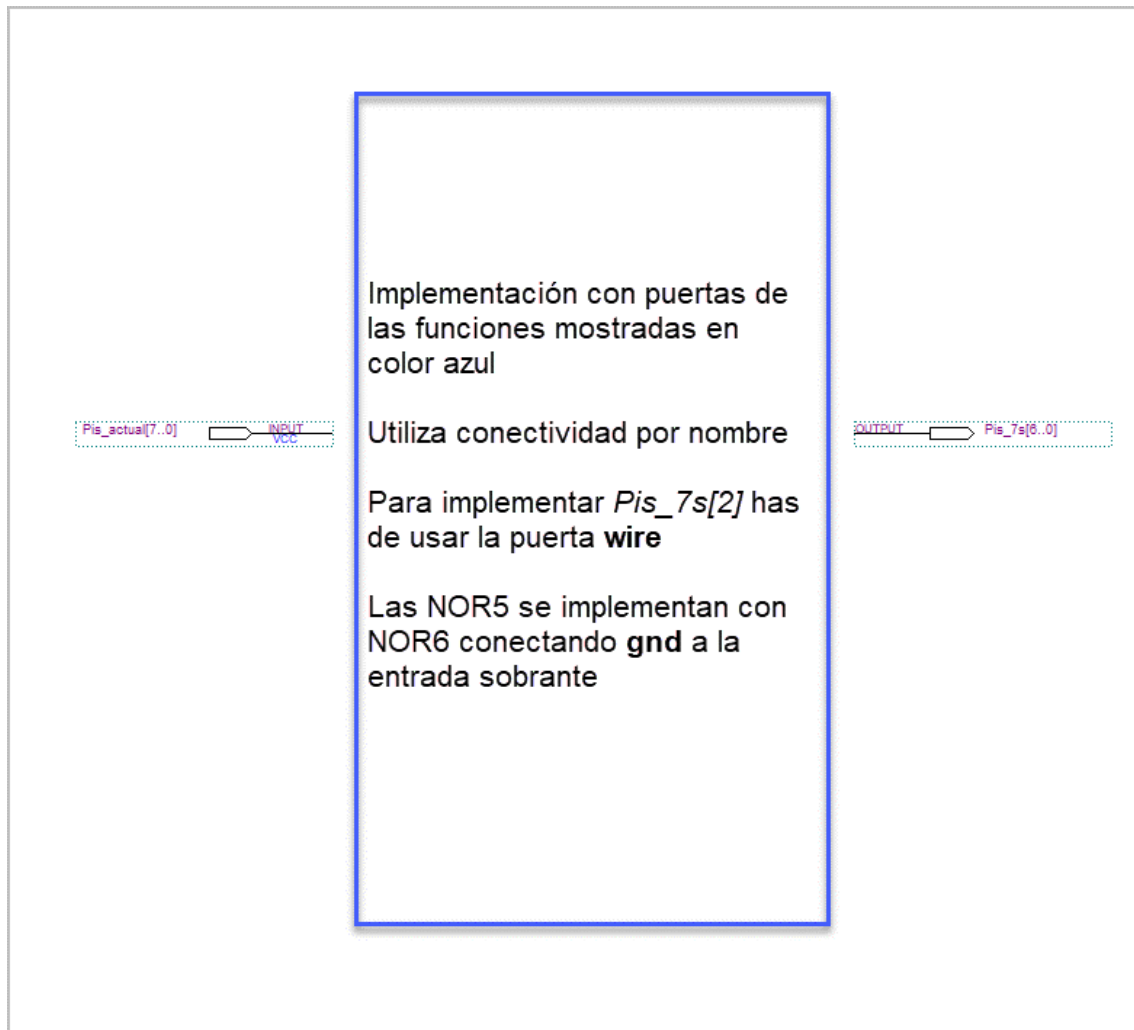
File → New

En el menú desplegable que aparece, selecciona *Block Diagrams/Schematic File* y clicla en *OK*.

Con lo que has aprendido haciendo el módulo **Obrir_Alliberar** crea el esquema del módulo **Display** el cual implementa las expresiones booleanas mostradas en el punto anterior (por simplicidad del esquema, usaremos las expresiones NOR mostradas en azul).

Si te fijas hay una salida (*Pis_7s[2]*) que coincide con una entrada (*Pis_actual[2]*). Quartus no permite que una entrada esté directamente conectada a una salida por motivos técnicos. Para soslayar esta situación, Quartus te obliga a introducir una “puerta lógica ficticia” (esto es, que sólo le sirve para que el software correspondiente entienda que se trata de una conexión entrada-salida correcta), la puerta **wire**, la cual hallarás en la carpeta **primitives → buffer**.

Asimismo, has de tener en cuenta que al no existir una puerta NOR de 5 entradas en la biblioteca, la operación NOR de 5 entradas se habrá de implementar mediante una puerta NOR de 6 entradas, conectando un 0 lógico a su última entrada. Para ello se utiliza la celda **gnd** que hallarás en **primitives** → **other**.



Puesto que ya has concluido el esquema lógico, debes guardar el esquema ejecutando:

File → Save

En el formulario que se abre **selecciona la carpeta del proyecto (U:\FCPract\4XX_YY)** e indica el nombre del módulo, en este caso **Display.bdf**.


Finalmente, genera un símbolo para que, más adelante, puedas utilizar este módulo en la creación de otro módulo más complejo que lo contenga. Para ello basta ejecutar el comando:

File→Create/Update→Create Symbol Files for Current File

En el formulario que se abre **selecciona la carpeta del proyecto (U:\FCPract\4XX_YY)**, da el nombre **Display.bsf** al símbolo, y clicas en **Guardar**.

6.2 COMPILAR EL CIRCUITO

Vamos ahora a compilar el módulo **Display**. Recordemos que, como ya hiciste con `Obrir_Alliberar`, para compilar el módulo necesitas:

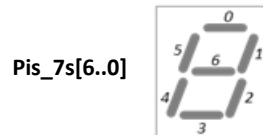
- 1) Definirlo como *Top_level entity*. Para ello (1) en la sección **Files** del **Project Navigator** clicas con el botón derecho del ratón sobre el fichero **Display.bdf** y (2) selecciona '*Set as Top-Level Entity*'.
- 2) Lanzar el compilador mediante el botón 

Durante la compilación te irán apareciendo mensajes en la parte inferior de la pantalla. Si se produce algún error, deberás analizar el error y resolverlo.

6.3 SIMULACIÓN DEL MÓDULO

Como ya sabes, la simulación tiene como objetivo verificar que el módulo diseñado se comporta correctamente, es decir, de la forma esperada. ¿Qué significa eso en el caso del módulo **Display**? Pues que en función del piso en donde está el ascensor, lo cual viene indicado por la entrada *Pis_actual[7..0]*, los valores de la salida *Pis_7s[6..0]* enciendan los leds pertinentes del 7-segmentos para mostrar el número del piso.

Recordemos que el 7-segmentos funciona con lógica negativa (para que un led se encienda ha de recibir un 0 lógico) y que la salida *Pis_7s[6..0]* controla los leds de la siguiente manera:



El comportamiento correcto del módulo quedaría resumido en la siguiente tabla que muestra los ocho pisos en donde se puede hallar el ascensor:

Ascensor en piso 0 →	<i>Pis_actual[7..0]</i> = 00000001 →	<i>Pis_7s[6..0]</i> = 1000000 →	
Ascensor en piso 1 →	<i>Pis_actual[7..0]</i> = 00000010 →	<i>Pis_7s[6..0]</i> = 1111001 →	
Ascensor en piso 2 →	<i>Pis_actual[7..0]</i> = 00000100 →	<i>Pis_7s[6..0]</i> = 0100100 →	
Ascensor en piso 3 →	<i>Pis_actual[7..0]</i> = 00001000 →	<i>Pis_7s[6..0]</i> = 0110000 →	
Ascensor en piso 4 →	<i>Pis_actual[7..0]</i> = 00010000 →	<i>Pis_7s[6..0]</i> = 0011001 →	
Ascensor en piso 5 →	<i>Pis_actual[7..0]</i> = 00100000 →	<i>Pis_7s[6..0]</i> = 0010010 →	
Ascensor en piso 6 →	<i>Pis_actual[7..0]</i> = 01000000 →	<i>Pis_7s[6..0]</i> = 0000010 →	
Ascensor en piso 7 →	<i>Pis_actual[7..0]</i> = 10000000 →	<i>Pis_7s[6..0]</i> = 1111000 →	

Vamos ahora a realizar una simulación en la que comprobaremos que para cada uno de los ocho valores posibles de *Pis_actual[7..0]* se genera el valor adecuado en *Pis_7s[6..0]*.

Los pasos que se siguen para realizar la simulación son análogos a los que seguiste para simular el módulo `Obrir_Alliberar`. Por ello no son explicados de forma detallada. Si tienes alguna duda consulta la parte del guion donde se explica la simulación del módulo `Obrir_Alliberar`.

Creación del fichero de vectores de simulación

El primer paso es crear el fichero de vectores de simulación. Para ello, ejecuta la orden:

File → New

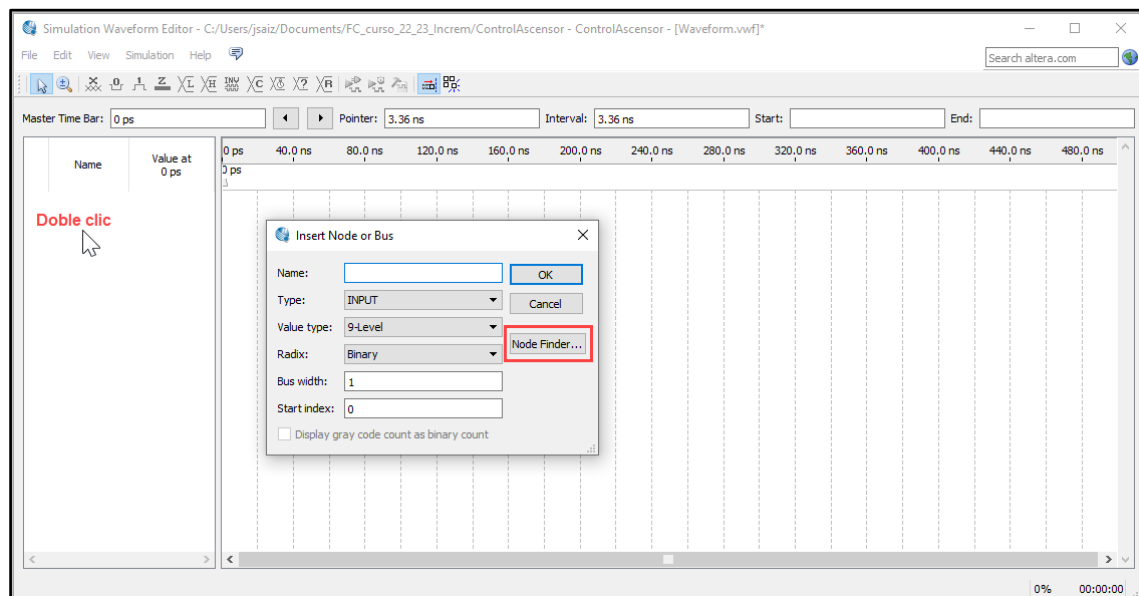
En el menú desplegable que aparece, selecciona **University Program VWF** y clic en **OK**.

Establece la anchura de las franjas temporales del editor de formas de onda y la duración de la simulación:

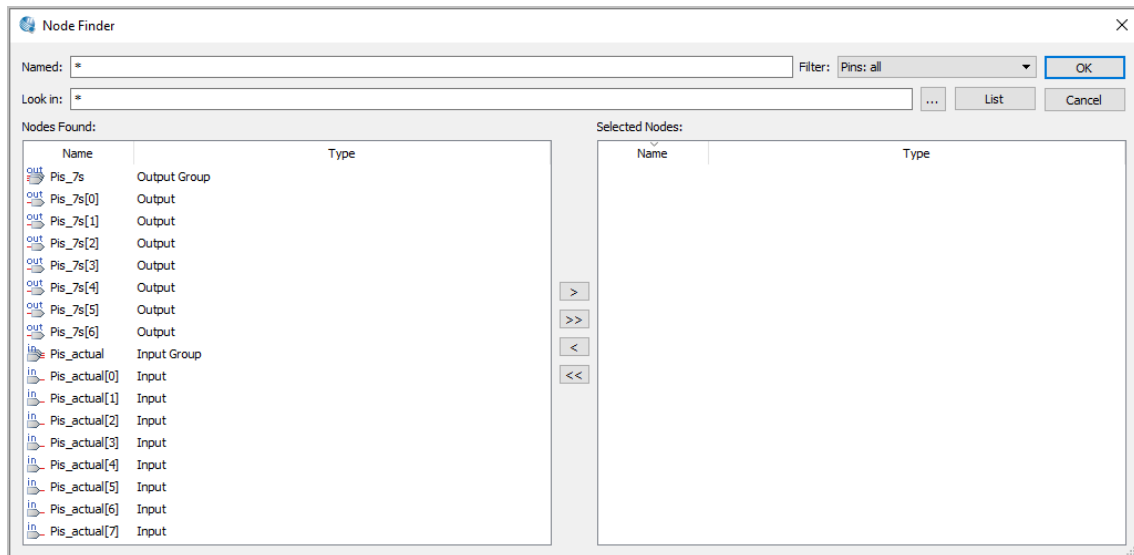
Edit → Grid size... → 20 ns

Edit → Set End Time... → 500 ns

Para introducir las entradas y sus valores, haz doble clic debajo de la columna *Name*. En el formulario que aparece, clic en el botón **Node Finder...**.



Se abrirá una nueva ventana desde donde podrás seleccionar los nodos de entrada (*Pis_actual*) y los nodos de salida (*Pis_7s*):



Selecciona la opción “**Pins: all**” en **Filter**, y clicas en **List**. En la subventana “**Nodes Found**” aparecerán todos los nodos de entrada y salida de **Display**. Selecciona el nodo *Pis_actual* y clicas en > (flecha 2). Repite el mismo proceso para el nodo *Pis_7s*. Finalmente clicas en **OK**.

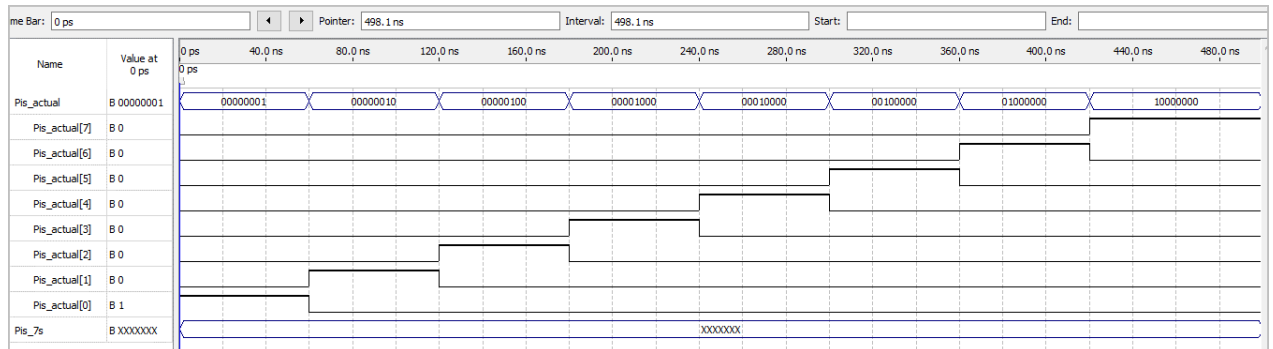
Te volverá a aparecer la ventana **Insert Node or Bus**. En el campo **Radix** selecciona **Binary** (de esa forma el valor de los buses se mostrará en binario en la ventana de simulación) y vuelve a clicar en **OK**. Los nodos se incorporarán a la ventana de simulación.

Lo siguiente que hemos de hacer es definir el valor de *Pis_actual[7..0]* en cada instante de tiempo. Pero antes comentemos en qué consistirá la estrategia de test.

Estrategia de test

Para comprobar que el circuito funciona, la estrategia que se seguirá es muy simple. Se establecerá inicialmente que el ascensor está en el piso 0 (*Pis_actual* = 00000001), después el ascensor sube al piso 1 (*Pis_actual* = 00000010), de ahí al piso 2 (*Pis_actual* = 00000100) y así sucesivamente hasta llegar al piso 7 (*Pis_actual* = 10000000). Mientras que el ascensor está en cada uno de los pisos se comprobará que la salida *Pis_7s* adopta el valor requerido para que se enciendan los leds del 7-segmentos correspondientes al número del piso. En la página 33 se indicaban cuáles eran esos valores en formato binario.

Siguiendo el procedimiento que ya conoces para dibujar las formas de onda, introduce la totalidad de los valores correspondientes a la entrada *Pis_actual*.



Finalmente, guarda el fichero de simulación:


File → Save

En el formulario que se abre, **selecciona la carpeta del proyecto (U:\FCPract\4XX_YY)** e introduce como nombre del fichero **Display.vwf**.

Simulación del circuito

Selecciona el simulador propio de Quartus II (*qsim*) como el simulador que vamos a utilizar:

Simulation → Options → Seleccionar Quartus II Simulator → Pulsar OK

Realiza una simulación funcional. Para ello clica en el botón .

El resultado de la simulación aparece en una nueva ventana, en la que puedes agrandar o reducir la escala de tiempos con el icono de zoom, y en la que puedes moverte a lo largo de la escala de tiempo con la barra de la parte inferior.

Lo único que falta por hacer es comprobar que los valores de la salida *Pis_7s* que da el simulador coinciden con los valores esperados en todos los casos (están indicados en la página 33). Si no es así, ello significa que el circuito es incorrecto. En tal caso, para detectar el origen del error, puede ser útil ver qué bit de *Pis_7s* presenta un valor erróneo y revisar la puerta del módulo *Display* que genera ese bit, así como las conexiones de esa puerta.

Cuando acabes la simulación, guarda sus resultados ejecutando el comando:

File → Save As ...

En el formulario que aparece **selecciona la subcarpeta *simulation\qsim* del proyecto (U:\FCPract\4XX_YY\simulation\qsim)** e indica como nombre del fichero **Display_sim.vwf**. Finalmente pulsa el botón **Abrir**.

El fichero con el resultado de la simulación (**Display_sim.vwf**) no se considera que forme parte del proyecto y por ese motivo no aparece en la sección *Files* del *Project Navigator*. Para añadirlo al proyecto se ejecutará el comando:

Project → Add/Remove Files in Project...

y se seguirá el mismo procedimiento que se siguió para añadir el fichero **Obrir_Alliberar_sim.vwf** al proyecto.

Si has hecho correctamente el proceso ahora se debería ver el fichero [simulation/qsim/Display_sim.vwf](#) en la sección *Files* del *Project Navigator*.

7 CERRAR EL PROYECTO

Cuando hayas acabado, cierra correctamente el proyecto mediante la orden:

File → Close Project

Cuando quieras reabrir el proyecto haz:

File → Open Project *ControlAscensor.qpf* (bindkey **Ctrl+J**)

Llegado a este punto, **recuerda comprimir la carpeta 4XX_YY en un fichero .zip y súbela a tu nube**. Al inicio de la siguiente sesión de prácticas la deberás descargar y descomprimir para seguir desarrollando el proyecto.

Entrega correspondiente a esta primera sesión

La entrega consistirá en un fichero .zip cuyo nombre será **FC-S1-4XX_YY.zip**, donde **4XX_YY** es el identificador del equipo de trabajo (por ejemplo, **FC-S1-411_01.zip**), que ha de contener seis ficheros:

- el fichero **Obrir_Alliberar_1_bit.bdf**
- el fichero **Obrir_Alliberar.bdf**
- el fichero **Obrir_Alliberar_sim.vwf**
- el fichero **Display.bdf**
- el fichero **Display_sim.vwf**
- un fichero denominado **autoría.txt** con el nombre y el NIU del autor/autores de la entrega.

La entrega del fichero **FC-S1-4XX_YY.zip** se ha de realizar a través del Campus Virtual.