

SESIÓN 6: EL CONTROLADOR DEL ASCENSOR – IMPLEMENTACIÓN EN LA PLACA ALTERA DE2

En esta práctica juntaremos todos los módulos del controlador de ascensor y bajaremos el diseño a la placa Altera DE2 para comprobar su correcto funcionamiento. Como ya se explicó en la presentación de las prácticas, el controlador del ascensor está diseñado siguiendo una arquitectura Unidad de Proceso (UP) - Unidad de Control (UC). El primer paso consistirá en elaborar esos dos grandes bloques, UP y UC, a partir de los módulos realizados hasta ahora. Posteriormente, se unirán esos bloques para obtener el sistema digital completo. Una vez hecho eso se procederá a implementarlo en la FPGA de la placa Altera DE2 y se testeará. Para concluir la sesión, se hará una simulación en la que se simulará una situación concreta de funcionamiento del sistema.

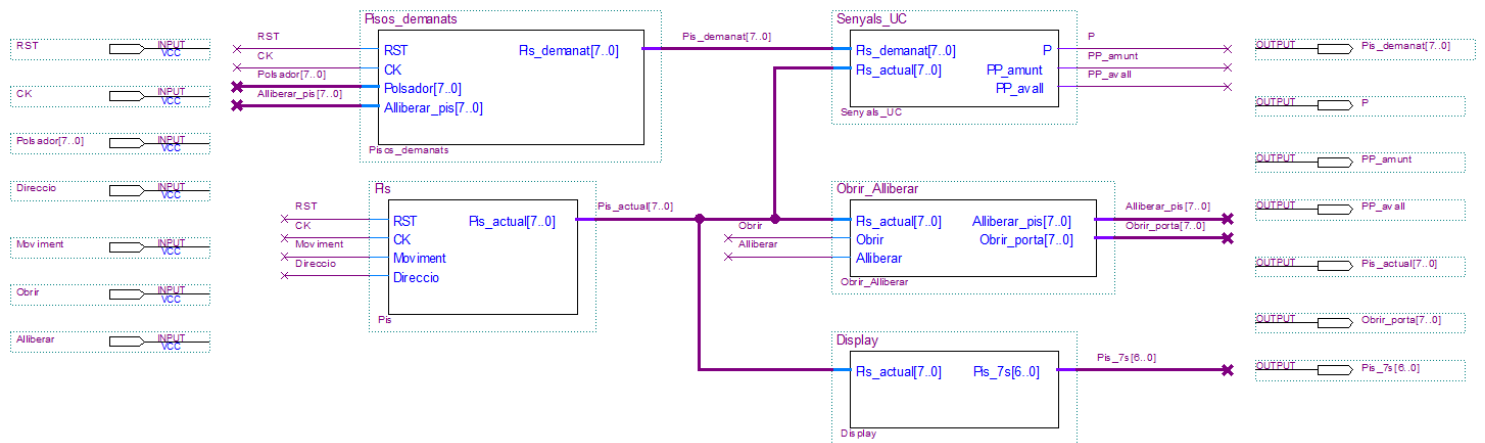
Esta práctica **obligatoriamente** se ha de realizar en un ordenador del laboratorio para evitar problemas de compatibilidad de *drivers* con los ordenadores personales de los estudiantes. Por ello antes de iniciar la sesión se tiene que:

- descargar de la nube la carpeta del proyecto comprimida y moverla a **U:\FCPract**
- una vez está la carpeta comprimida en **U:\FCPract**, descomprimir la carpeta
- borrar la carpeta comprimida

TRABAJO PREVIO: Tal como se informó durante la sesión 5, cada equipo de trabajo ha de traer ya hechos los esquemas lógicos de los módulos **UP** (Unidad de Proceso), **UC** (Unidad de Control) y **ControlAscensor** (el sistema completo) al inicio de la sesión. Se controlará si han sido realizados o no, y ello afectará a la puntuación de la sesión.

1. LA UNIDAD DE PROCESO

Construye en primer lugar el esquemático de la Unidad de Proceso entrando y conectando convenientemente los símbolos de los módulos que la componen (**Pisos_demanats**, **Senyals_UC**, **Pis**, **Obrir_Alliberar** y **Display**).



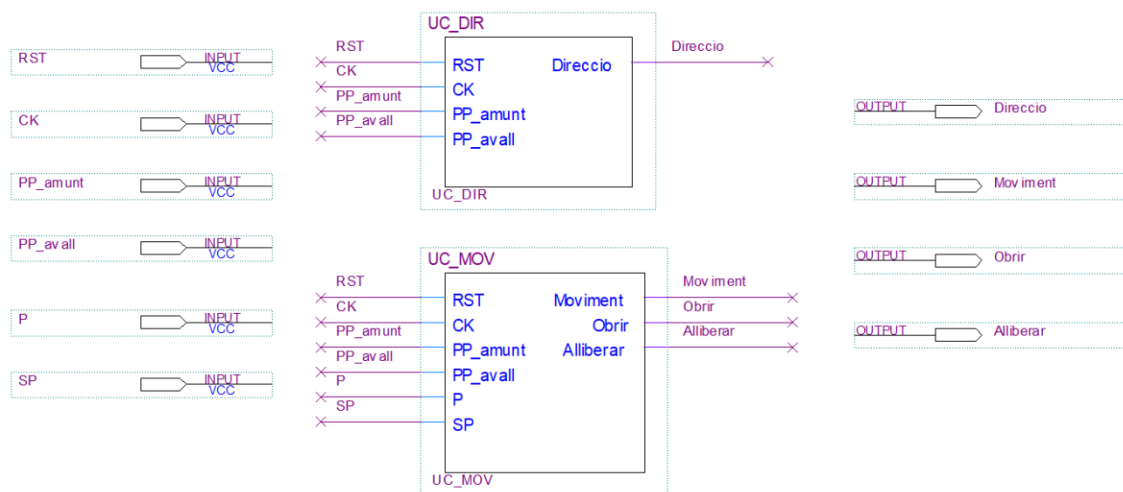
Da a cada celda como nombre de instancia el mismo nombre de la celda. Así, por ejemplo, puedes ver que la celda **Pis** tiene como nombre de instancia también **Pis**.

Guarda el esquemático de la Unidad de Proceso en un fichero con el nombre **UP.bdf** en la carpeta del proyecto (**U:\FCPract\4XX_YY**). Posteriormente declara este módulo como *top-level entity* y compílalo. En principio, solo se deberían producir 8 *warnings*. Finalmente, crea un símbolo para este nuevo módulo y guárdalo en la carpeta del proyecto (**U:\FCPract\4XX_YY**).

2. LA UNIDAD DE CONTROL

Construye a continuación el esquemático de la Unidad de Control entrando y conectando convenientemente los símbolos de los dos módulos que la componen (**UC_DIR** y **UC_MOV** [este el módulo que se realizó en la sesión anterior y que consistía en una máquina de 6 estados])).

Atención: Es posible que el símbolo que generaste en la sesión 3 para UC_DIR tenga las entradas en un orden distinto al que se observa en la imagen. Asegúrate de que conectas a cada entrada de UC_DIR la señal que le corresponde.

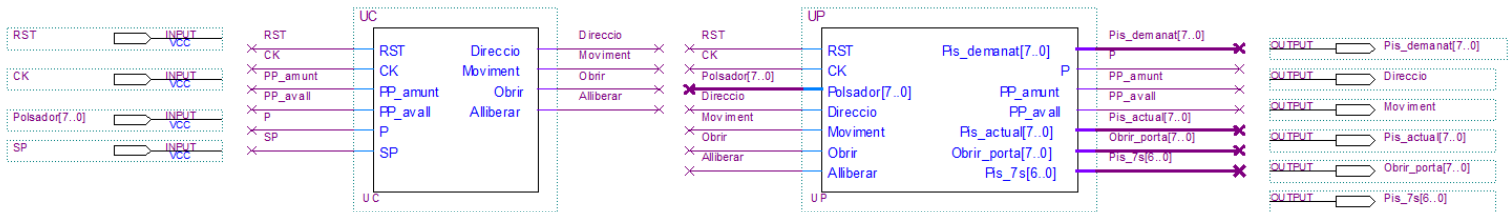


Da a cada celda como nombre de instancia el mismo nombre de la celda. Así, por ejemplo, puedes ver que la celda **UC_MOV** tiene como nombre de instancia también **UC_MOV**.

Guarda el esquemático de la Unidad de Control en un fichero con el nombre **UC.bdf** en la carpeta del proyecto (**U:\FCPract\4XX_YY**). Posteriormente declara este módulo como *top-level entity* y compílalo. En principio, solo se deberían producir 8 *warnings*. Finalmente, crea un símbolo para este nuevo módulo y guárdalo en la carpeta del proyecto (**U:\FCPract\4XX_YY**).

3. EL CONTROLADOR DEL ASCENSOR

Finalmente, crea un nuevo esquemático al que llamaremos **ControlAscensor** que englobe a la Unidad de Proceso y a la Unidad de Control.



Da a cada celda como nombre de instancia el mismo nombre de la celda. Así, por ejemplo, puedes ver que la celda **UC** tiene como nombre de instancia también **UC**.

Puedes observar que el controlador del ascensor tiene como **entradas**:

RST: la señal de reset del circuito (activa a alta, es decir, cuando vale 1 se produce el reset del circuito).

CK: la señal de reloj del circuito.

Polgador[7..0]: bus de 8 bits, cada uno de ellos vinculados a la botonera del ascensor y al botón de llamada que hay en cada piso. Cuando por parte de un usuario se produce una solicitud al ascensor para que vaya a un piso, el bit de **Polgador** correspondiente a ese piso **se pone a 1 durante un periodo de reloj**.

SP: Señal indicadora de que hay algún obstáculo que impide el cierre de la puerta del ascensor.

Asimismo, las **salidas** del controlador del ascensor son:

Pis_demanat[7..0]: bus de 8 bits que indica las solicitudes existentes al ascensor pendientes de ser atendidas. Cada bit del bus está vinculado a un piso y cuando vale 1 quiere decir que existe una solicitud para que el ascensor vaya a ese piso pendiente de ser atendida.

Direccio: Señal que indica a la parte mecánica del ascensor la dirección del movimiento del ascensor (solo es relevante si el ascensor se está moviendo).

Moviment: Señal que indica a la parte mecánica del ascensor si el ascensor se ha de mover o no.

Pis_actua[7..0]: bus de 8 bits que indica el piso en el que se encuentra el ascensor. Cada bit del bus está vinculado a un piso. De los ocho bits del bus, un bit está a uno (indicando el piso donde está el ascensor) y los siete restantes a cero.

Obrir_porta[7..0]: bus de 8 bits que indica si está abierta la puerta del ascensor en algún piso. Cada bit del bus está vinculado a un piso. Como máximo, solo uno de los bits puede estar a uno, indicando con ello que en ese piso la puerta del ascensor está abierta.

Pis_7s[6..0]: bus de 7 bits que controla un 7-segmentos en el que se muestra gráficamente el piso donde se halla el ascensor.

Guarda el esquemático del controlador en un fichero con el nombre **ControlAscensor.bdf** en la carpeta del proyecto (**U:\FCPract\4XX_YY**). Declara este módulo como *top-level entity* y compílalo. En principio, solo se deberían producir 8 *warnings*.

4. IMPLEMENTACIÓN EN PLACA

4.1 LA PLACA ALTERA DE2

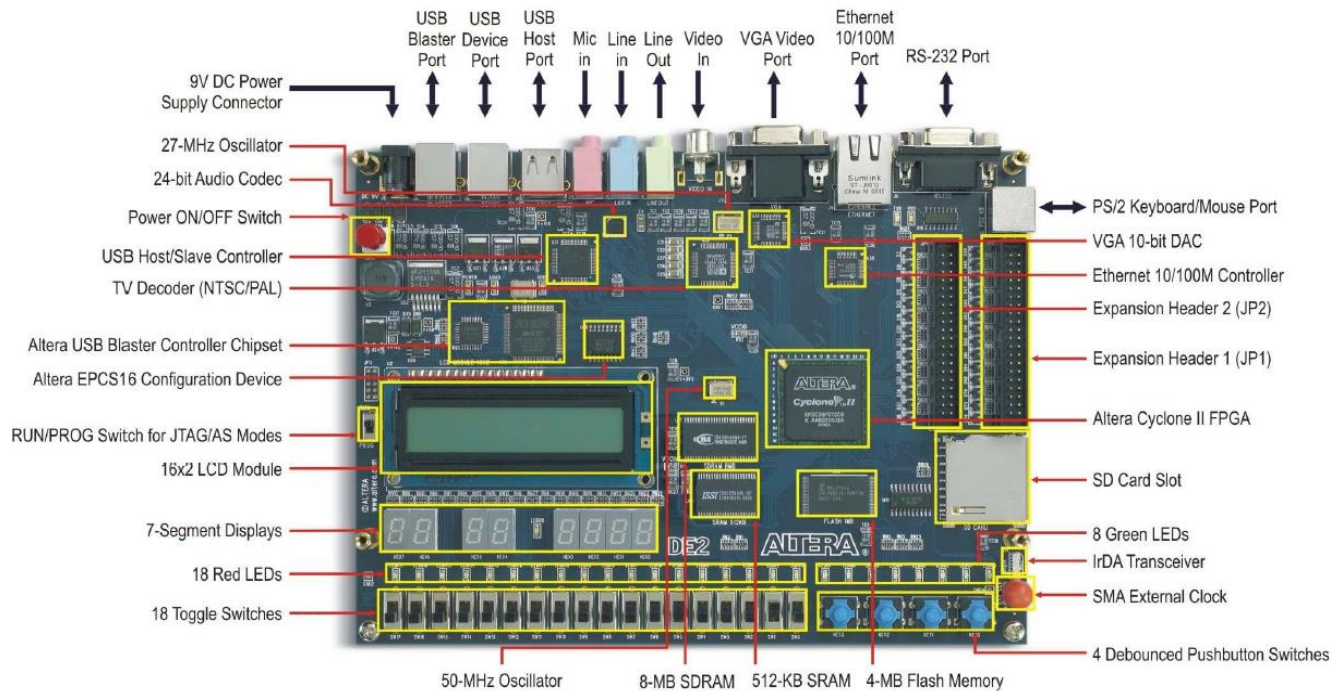
La placa de prototipado que vais a utilizar para implementar vuestro sistema y comprobarlo es la placa Altera DE2. Conectada al PC a través de un cable USB, y utilizando Quartus II, podréis:

- Programar (configurar) la FPGA de Altera de la familia Cyclone II que se halla en la placa
- Mediante la programación citada en el anterior punto, hacer uso de los dispositivos presentes en la placa que están conectados a la FPGA: *switches*, pulsadores, LEDs, 7-segmentos, etc. Con estos elementos crearéis un entorno similar al del ascensor para comprobar que el circuito desarrollado funciona correctamente.

De todo el arsenal de dispositivos que tiene la placa DE2 nosotros vamos a limitarnos a utilizar los siguientes:

- Los *switches* SW7 a SW0, para emular la botonera (*Pulsador[7..0]*).
- Si bien se puede generar una señal de reloj en la propia placa, emularemos manualmente la señal *CK* con el pulsador KEY0 para poder visualizar mejor lo que sucede pulso a pulso.
- La señal de *RST* la entraremos mediante el *switch* SW17.
- La señal de *SP* la entraremos mediante el *switch* SW16.
- El 7-segmentos HEX0 (el de más a la derecha) para visualizar **gráficamente** el piso actual (*Pis_7s[6..0]*).
- Los leds rojos LEDR7 a LEDR0 para visualizar el vector *Obrir_porta[7..0]*
- Los leds rojos LEDR15 a LEDR8 para visualizar el vector *Pis_actual[7..0]*
- El led rojo LEDR16 para visualizar la señal *Direccio*
- El led rojo LEDR17 para visualizar la señal *Moviment*
- Los leds verdes LEDG7 a LEDG0 para visualizar el vector *Pis_demanat[7..0]*

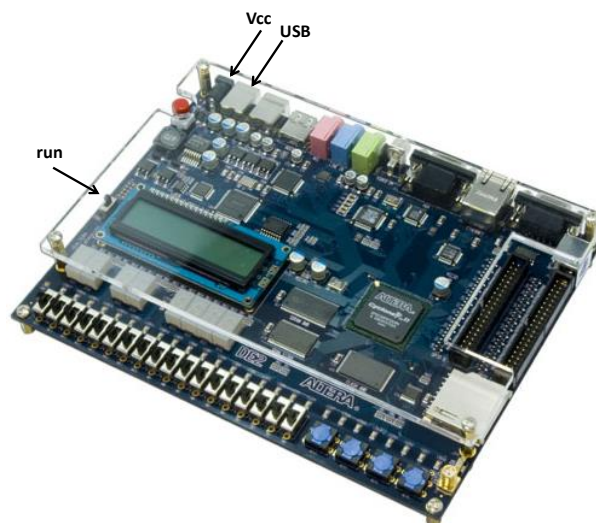
Busca dónde está cada uno de estos dispositivos en la imagen de la placa DE2 que se halla en la página siguiente.



4.2 PROGRAMACIÓN DE LA FPGA DE LA PLACA ALTERA DE2 Y COMPROBACIÓN DEL MÓDULO CONTROLADOR DEL ASCENSOR

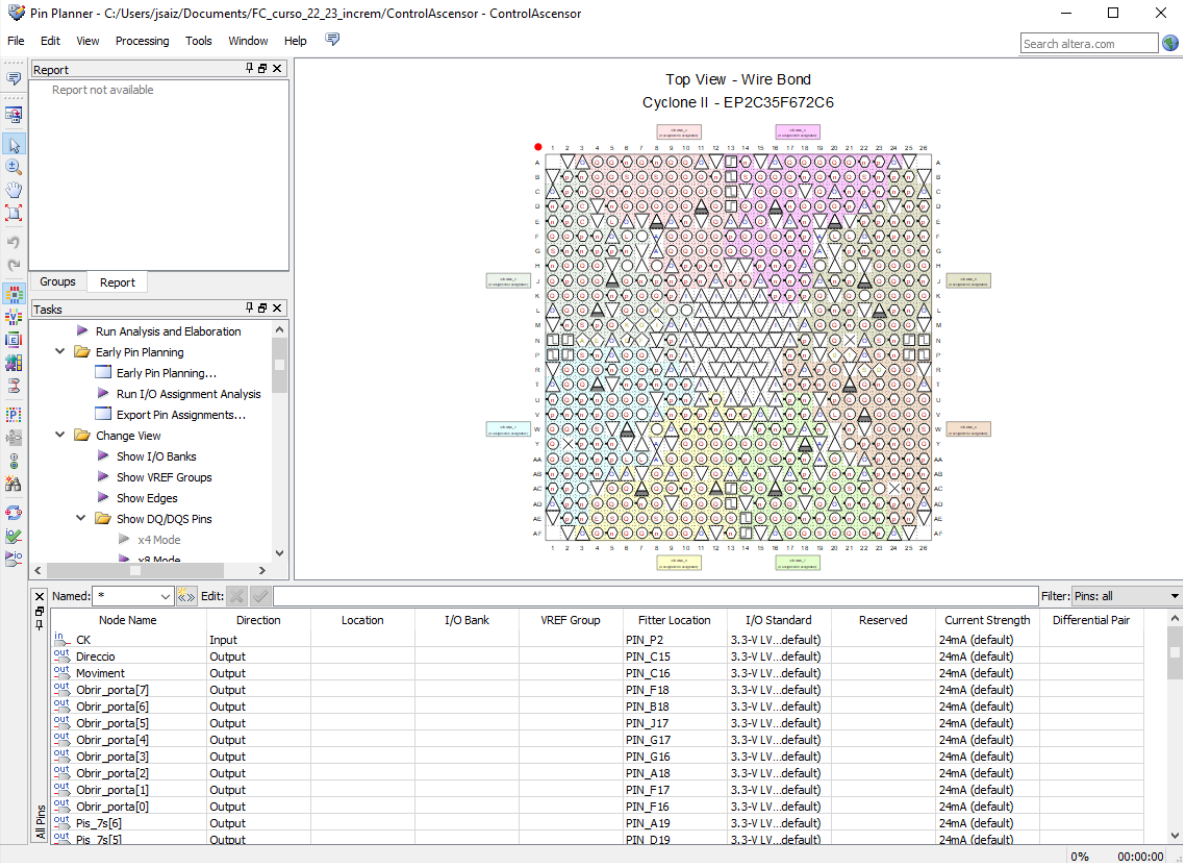
Para realizar la programación (configuración) de la FPGA de la placa Altera DE2, procede de la siguiente manera:

1. Conecta la alimentación (Vcc) y el USB de la placa DE2 tal como muestra la figura a pie de página. (**iNo lo hagas al revés!**; el USB que debes utilizar en la placa es el que está al lado de Vcc, ¿de acuerdo?).
2. Asegúrate que el *switch* RUN/PROG está en modo RUN, y pulsa el *switch* Power ON/OFF (botón rojo).



3. Asigna los *ports* de entrada y salida del módulo **ControlAscensor** a los pines de la FPGA. Has de seleccionar los pines de la FPGA que están conectados, en la placa, con los elementos (7-segmentos, pulsadores, leds y *switches*) que se especifican en la sección 4.1.

Para ello clica en *Assignments* → *Pin Planner*. Se abrirá una ventana como la siguiente:

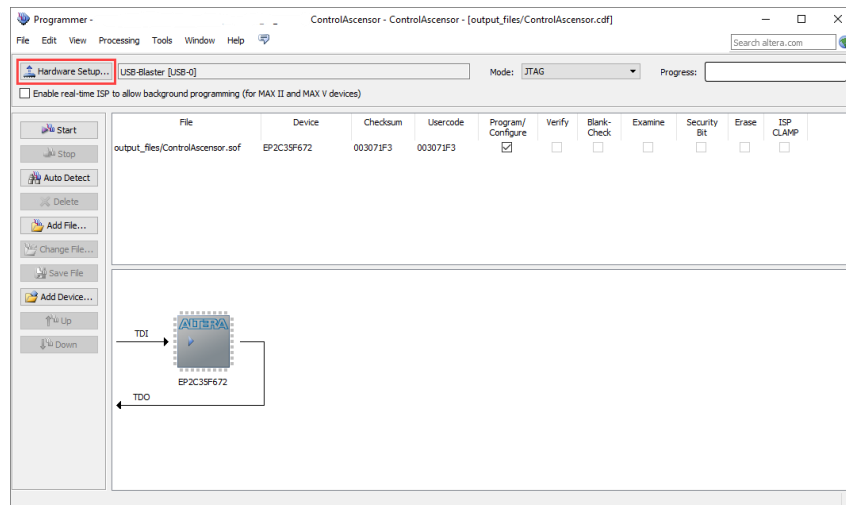


Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
in_CK	Input				PIN_P2	3.3-V LV...default		24mA (default)	
Out_Direccio	Output				PIN_C15	3.3-V LV...default		24mA (default)	
Out_Moviment	Output				PIN_C16	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[7]	Output				PIN_F18	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[6]	Output				PIN_B18	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[5]	Output				PIN_J17	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[4]	Output				PIN_G17	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[3]	Output				PIN_G16	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[2]	Output				PIN_A18	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[1]	Output				PIN_F17	3.3-V LV...default		24mA (default)	
Out_Obrir_porta[0]	Output				PIN_F16	3.3-V LV...default		24mA (default)	
Out_Pis_7s[6]	Output				PIN_A19	3.3-V LV...default		24mA (default)	
Out_Pis_7s[5]	Output				PIN_D19	3.3-V LV...default		24mA (default)	

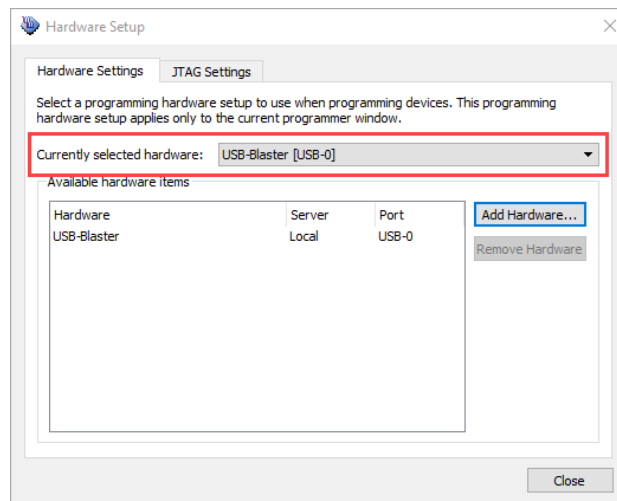
Asigna los *ports* de entrada y de salida a los pines de la FPGA de acuerdo con la distribución que se especifica en la tabla de la siguiente página, introduciendo el **identificador del pin de la FPGA** correspondiente en la columna “**Location**”.

Señal	Dispositivo	IDENT. PIN FPGA
CK	Pulsador KEY[0]	G26
RST	Switch SW[17]	V2
Pulsador[7]	Switch SW[7]	C13
Pulsador[6]	Switch SW[6]	AC13
Pulsador[5]	Switch SW[5]	AD13
Pulsador[4]	Switch SW[4]	AF14
Pulsador[3]	Switch SW[3]	AE14
Pulsador[2]	Switch SW[2]	P25
Pulsador[1]	Switch SW[1]	N26
Pulsador[0]	Switch SW[0]	N25
SP	Switch SW[16]	V1
Pis_demanat[7]	Led verde LEDG[7]	Y18
Pis_demanat[6]	Led verde LEDG[6]	AA20
Pis_demanat[5]	Led verde LEDG[5]	U17
Pis_demanat[4]	Led verde LEDG[4]	U18
Pis_demanat[3]	Led verde LEDG[3]	V18
Pis_demanat[2]	Led verde LEDG[2]	W19
Pis_demanat[1]	Led verde LEDG[1]	AF22
Pis_demanat[0]	Led verde LEDG[0]	AE22
Pis_actual[7]	Led rojo LEDR[15]	AE13
Pis_actual[6]	Led rojo LEDR[14]	AF13
Pis_actual[5]	Led rojo LEDR[13]	AE15
Pis_actual[4]	Led rojo LEDR[12]	AD15
Pis_actual[3]	Led rojo LEDR[11]	AC14
Pis_actual[2]	Led rojo LEDR[10]	AA13
Pis_actual[1]	Led rojo LEDR[9]	Y13
Pis_actual[0]	Led rojo LEDR[8]	AA14
Obrir_porta[7]	Led rojo LEDR[7]	AC21
Obrir_porta[6]	Led rojo LEDR[6]	AD21
Obrir_porta[5]	Led rojo LEDR[5]	AD23
Obrir_porta[4]	Led rojo LEDR[4]	AD22
Obrir_porta[3]	Led rojo LEDR[3]	AC22
Obrir_porta[2]	Led rojo LEDR[2]	AB21
Obrir_porta[1]	Led rojo LEDR[1]	AF23
Obrir_porta[0]	Led rojo LEDR[0]	AE23
Pis_7s[6]	7-segmentos HEX0[6]	V13
Pis_7s [5]	7-segmentos HEX0[5]	V14
Pis_7s [4]	7-segmentos HEX0[4]	AE11
Pis_7s [3]	7-segmentos HEX0[3]	AD11
Pis_7s [2]	7-segmentos HEX0[2]	AC12
Pis_7s [1]	7-segmentos HEX0[1]	AB12
Pis_7s [0]	7-segmentos HEX0[0]	AF10
Direccio	Led rojo LDR[16]	AE12
Moviment	Led rojo LDR[17]	AD12

4. **Compila de nuevo el circuito.** Solo debería haber 7 *warnings*.
5. Programa la FPGA: Clica en **Tools** → **Programmer**. Se abrirá una ventana como la siguiente:

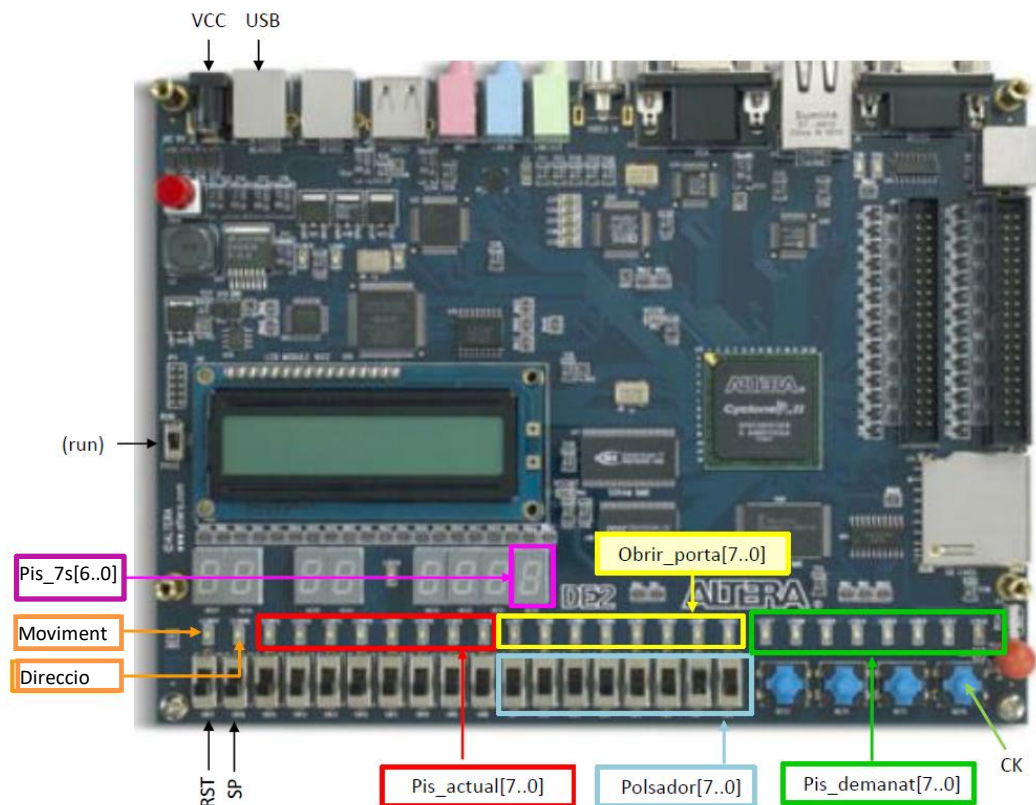


Clica en **Hardware Setup** y en **Currently selected hardware** selecciona **USB_Blaster**. Clica en **Close**.



A continuación, en la ventana del *Programmer* haz clic en **Start** y el hardware del módulo **ControlAscensor** se replicará en la FPGA.

6. Comprueba que el controlador funciona correctamente emulando el comportamiento de un usuario que entra en el ascensor y selecciona un piso, y de otros usuarios que llaman al ascensor desde diversos pisos. Para ello te puede resultar útil consultar la imagen de la siguiente página en la que se muestra la disposición de los pulsadores, *switches*, leds y 7-segmentos que vas a usar durante la emulación.



CK → Pulsador KEY[0]

RST → switch SW[17]

SP → switch SW[16]

Polsador[7..0] → switches SW[7] .. SW[0]

Pis_demanat[7..0] → leds verdes LEDG[7] .. LEDG[0]

Obrir_porta[7..0] → leds rojos LEDR[7]..LEDR[0]

Pis_actual[7..0] → leds rojos LEDR[15]..LED[8]

Pis_7s[6..0] → 7-segmentos situado más a la derecha

Moviment → led rojo LEDR[17]

Direccio → led rojo LEDR[16]

5. SIMULACIÓN DE UN CASO CONCRETO















Realizar una simulación exhaustiva del controlador del ascensor requeriría mucho tiempo. Por ello, nos vamos a conformar con simular un caso concreto. El caso es el siguiente:

- Inicialmente se resetea el sistema.
- Tras unos ciclos de reloj se produce una solicitud para que el ascensor vaya al piso 6 (*Polsador*[6] = 1).
- Cuando el ascensor está ascendiendo y va por el piso 5 se produce una solicitud para que el ascensor vaya al piso 2 (*Polsador*[2] = 1).
- Una vez que el ascensor ya ha llegado al piso 6 y ha abierto la puerta, un usuario A se da cuenta de que otro usuario B llega corriendo para intentar coger el ascensor. Para evitar que el ascensor se vaya sin el usuario B, el usuario A se pone en medio de la puerta bloqueando su cierre (*SP* = 1) durante unos ciclos de reloj. [Nota: el bloqueo de la puerta se realizará en el estado 3 de **UC_MOV**].
- Una vez que el usuario B entra al ascensor, el bloqueo de la puerta finaliza (*SP* = 0).

Para realizar la simulación, crea un fichero de estímulos vacío y añádele:

- las entradas (*RST*, *CK*, *Polsador* y *SP*) utilizando para ello el filtro **Pins:Input**. El bus *Polsador* se debe visualizar en formato binario (seleccionalo, clicas con el botón derecho del ratón y ejecuta **Radix → Binary**).
- El estado de los flip-flops del módulo **UC_MOV** utilizando para ello el filtro **Registers: post-fitting**. Los flip-flops aparecen con el nombre "**UC:UC|UC_MOV:UC_MOV|ff2**", "**...ff1**" y "**...ff0**", respectivamente. Forma un grupo con los flip-flops ff2, ff1 y ff0 (**respetar este orden**) y denomina a ese grupo **estado_UC_MOV**. Para formar el grupo selecciona ff2, ff1 y ff0, haz clic con el botón derecho y ejecuta el comando **Grouping -> Group**. En el formulario que aparece indica como **Group name** estado_UC_MOV y como **Radix** Unsigned Decimal.
- las salidas (*Direccio*, *Moviment*, *Pis_demanat*, *Obrir_porta*, *Pis_actual* y *Pis_7s*) utilizando para ello el filtro **Pins:Output**. Los buses *Pis_demanat*, *Obrir_porta*, *Pis_actual* y *Pis_7s* se deben visualizar en formato binario (**Radix → Binary**).

Tras ello, las señales en fichero de *waveforms* deberían quedar así:

	Name	Value at 0 ps
	RST	B 1
	CK	B 0
	> Pulsador	B 00000000
	SP	B 0
	▼ estado_UC_MOV	U X
	UC:UC UC_MOV:UC_MOV ff2	U U
	UC:UC UC_MOV:UC_MOV ff1	U U
	UC:UC UC_MOV:UC_MOV ff0	U U
	Direccio	B X
	Moviment	B X
	> Pis_demanat	B XXXXXXXX
	> Obrir_porta	B XXXXXXXX
	> Pis_actual	B XXXXXXXX
	> Pis_7s	B XXXXXXXX

En esta simulación vas a definir la señal de reloj (CK) como una señal de periodo 40 ns. Asimismo, especifica que la duración máxima de la simulación será de 4 μ s. Esto se puede establecer con el comando **Edit → Set End Time...** .

Ahora ya puedes especificar los estímulos de entrada de acuerdo con lo indicado en el caso que será simulado y que ha sido explicado al inicio de esta sección (recuerda que cada estímulo aplicado dura un periodo de reloj, de flanco de bajada a flanco de bajada del reloj):

- reseteo inicial del circuito ($RST = 1$)
- tras unos periodos de reloj, $Pulsador[6] = 1$ (solicitud para que el ascensor vaya al piso 6).
- cuando el ascensor está ascendiendo y va por el piso 5 (recuerda que el ascensor tarda cuatro ciclos en llegar al piso siguiente), $Pulsador[2] = 1$ (solicitud para que el ascensor vaya al piso 2).
- cuando el ascensor está parado en el piso 6, bloqueo de la puerta durante varios ciclos de reloj ($SP = 1$) cuando UC_MOV está en **Abrir3**. Para ello lo mejor es que de momento no pongas $SP=1$. Después cuando hagas por primera vez la simulación, anota en qué instante de tiempo **UC_MOV** se halla en el estado **Abrir3** y retoca el fichero de estímulos poniendo $SP = 1$ a partir de ese instante de tiempo, durante varios ciclos de reloj.
- Una vez que el usuario B entra al ascensor, el bloqueo de la puerta finaliza ($SP = 0$).

Guarda el fichero de estímulos de entrada con el nombre **ControlAscensor.vwf** en la carpeta del proyecto (U:\FCPract\4XX_YY).

A continuación, simula funcionalmente el circuito.

Comprueba si el controlador del ascensor funciona bien para lo cual tendrás que verificar si el comportamiento del circuito ante los diversos estímulos de entrada se corresponde con lo esperado (ver texto en azul):

- Inicialmente se resetea el sistema. Como consecuencia de ello:
 - o el controlador debería indicar que el ascensor está en el piso 0 (la planta baja) (*Pis_actual[7..0] = '00000001'*)
 - o todas las puertas deben estar cerradas (*Obrir_porta[7..0] = '00000000'*)
 - o no hay ninguna solicitud pendiente (*Pis_demanat[7..0] = '00000000'*)
 - o la máquina de estados del módulo UC_MOV debería estar en su estado inicial (el estado **Inicio** cuya codificación es 0)
 - o el ascensor debe estar parado (*Moviment = 0*) y la dirección deber ser hacia abajo (*Direccio = 0*)
- Tras unos ciclos de reloj se produce una solicitud para que el ascensor vaya al piso 6 (*Polsador[6] = 1*). Como consecuencia de ello:
 - o La solicitud debe reflejarse en *Pis_demanat* que ha de valer '01000000'.
 - o la máquina de estados del módulo **UC_MOV** debería pasar al estado **Mover** cuya codificación es 5.
 - o el ascensor debe ponerse en movimiento (*Moviment = 1*) y la dirección deber ser hacia arriba (*Direccio = 1*).
 - o El ascensor empieza a subir pisos (recuerda que, cada cuatro ciclos de reloj, sube un piso), lo cual se refleja en la salida *Pis_actual*.
- Cuando el ascensor va por el piso 5 se produce una solicitud para que el ascensor vaya al piso 2 (*Polsador[2] = 1*). Como consecuencia de ello:
 - o La solicitud debe reflejarse en *Pis_demanat* que ha de valer '01000100'
 - o El ascensor sigue su camino hacia el piso 6.
- Una vez que el ascensor ya ha llegado al piso 6 y ha abierto la puerta, un usuario A se da cuenta de que otro usuario B llega corriendo para intentar coger el ascensor. Para evitar que el ascensor se vaya sin el usuario B, el usuario A se pone en medio de la puerta bloqueando su cierre (*SP = 1*) durante unos ciclos de reloj. [Nota: el bloqueo de la puerta se realizará en el estado **Abrir3** (cuya codificación es 3) de **UC_MOV**]. Como consecuencia de ello:
 - o **UC_MOV** irá del estado 3 al estado **Abrir1** cuya codificación es 1 y permanecerá allí mientras *SP* valga 1.
 - o El ascensor permanecerá parado (*Moviment = 0*) y con la puerta abierta (*Obrir_porta[6] = 1* mientras dure el bloqueo).
- Una vez que el usuario B entra al ascensor, el bloqueo finaliza (*SP = 0*). Como consecuencia de ello:
 - o UC_MOV pasará por los estados **Abrir2** (cuya codificación es 2), **Abrir3** (3) y **LibPiso** (4). En **LibPiso** la puerta del piso 6 es cerrada (*Obrir_porta[6] = 0*) y la solicitud relativa al piso 6 es eliminada (*Pis_demanat[6] = 0*).
 - o Después, **UC_MOV** pasa al estado **Mover** (5) y el ascensor se pone en movimiento (*Moviment = 1*) en dirección hacia abajo (*Direccio = 0*).

- Debido a su movimiento, el ascensor llegará al piso 2 (*Pis_actual* = '00000100'), en el que se detiene y en el que abrirá la puerta (*Moviment* = 0 y *Obrir_porta*[2] = 1), pasando por los estados **Abrir1** (1), **Abrir2** (2) y **Abrir3** (3).
- Y ya para concluir, en el estado **LibPiso** (4) la puerta es cerrada (*Obrir_porta*[2] = 0) y la solicitud relativa al piso 2 es eliminada (*Pis_demanat*[2] = 0). Al no haber más solicitudes **UC_MOV** pasa al estado **Inicio** (0) y el ascensor se queda parado en el piso 2 (*Pis_actual* = '00000100').

Si se detecta algún comportamiento anómalo se deberá determinar el origen del mismo y corregirlo, modificando convenientemente el esquemático del módulo erróneo. Recuerda que tras modificarlo deberás compilar de nuevo el módulo **ControlAscensor**. Si, por el contrario, el circuito se comporta de la forma esperada, guarda el resultado de la simulación funcional en un fichero con el nombre **ControlAscensor_functional_sim.vwf** en la subcarpeta *simulation\qsim* del proyecto (**U:\FCPract\4XX_YY\simulation\qsim**).

Finalmente, realiza una simulación temporal (*timing simulation*) para comprobar que los retardos de los diversos elementos del circuito no alteran el correcto funcionamiento del circuito. Guarda el resultado de la simulación temporal en un fichero con el nombre **ControlAscensor_timing_sim.vwf** en la subcarpeta *simulation\qsim* del proyecto (**U:\FCPract\4XX_YY\simulation\qsim**).

A continuación, añade ambos ficheros .vwf al proyecto. Tras ello, se deberían ver dichos ficheros en la sección *Files* del *Project Navigator*.

Entrega correspondiente a esta sexta sesión

La entrega consistirá en un fichero .zip cuyo nombre será **FC-S6-4XX_YY.zip**, donde **4XX_YY** es el identificador del equipo de trabajo (por ejemplo, **FC-S6-411_01.zip**), que ha de contener **cuatro** ficheros:

- el fichero **Proyecto_4XX_XX.zip** que contendrá la carpeta de todo el proyecto comprimida
- el fichero **ControlAscensor_functional_sim.vwf**
- el fichero **ControlAscensor_timing_sim.vwf**
- un fichero denominado **autoría.txt** con el nombre y el NIU del autor/autores de la entrega.

La entrega del fichero **FC-S6-4XX_YY.zip** se ha de realizar a través del Campus Virtual.