SESIÓN 4: MÓDULO *PIS*

Los objetivos de esta práctica son:

- comprender qué son las megafunciones y familiarizarse con ellas
- trabajar con diversos tipos de contadores
- entrar y simular el módulo **Pis**.

Aquellos equipos que vayan a trabajar en un ordenador del laboratorio, antes de iniciar la sesión tienen que:

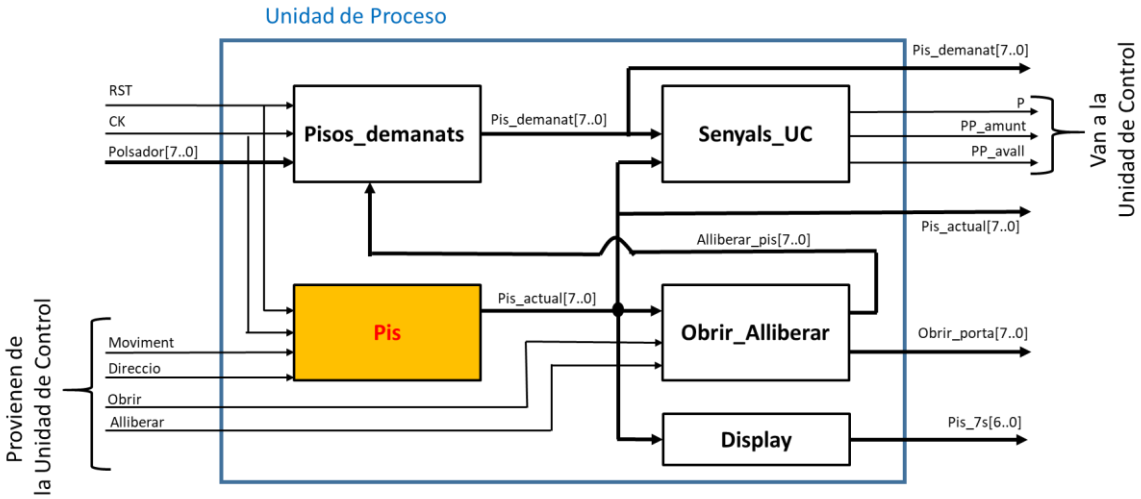
-descargar de la nube la carpeta del proyecto comprimida y moverla a **U:\FCPract**

-una vez está la carpeta comprimida en **U:\FCPract**, descomprimir la carpeta

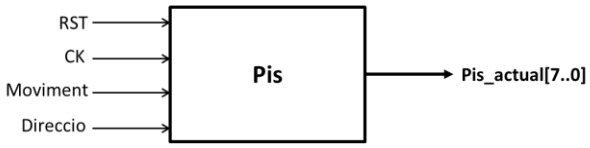
-borrar la carpeta comprimida

1. MÓDULO *PIS*

El módulo **Pis** es un módulo <u>secuencial</u> perteneciente a la Unidad de Proceso que tiene como función indicar en cada momento el piso en el que se encuentra el ascensor.



Veamos cuáles son las entradas y las salidas del módulo.



Como entradas tiene:

- *RST* cuya finalidad es resetear asíncronamente los contadores que forman parte del módulo, poniendo su valor a 0. La señal *RST* es activa a alta, es decir, debe realizar su misión cuando adopta el valor 1.

- *CK* es la señal de sincronización (reloj) que rige el funcionamiento de la parte secuencial del módulo.

- *Direccio*, señal proveniente de la Unidad de Control, que indica si el ascensor está subiendo (*Direccio*=1) o bajando (*Direccio*=0). Esta señal es generada por el módulo UC_DIR que desarrollasteis en la sesión anterior.

- *Moviment*, señal que proviene también de la Unidad de Control, e informa si el ascensor está parado (*Moviment* = 0) o moviéndose (*Moviment* = 1).

En cuanto a las salidas, hay solo una:

- *Pis_actual[7..0]*, bus formado por 8 bits que indica en qué piso se encuentra el ascensor. Cada bit está vinculado a un piso. De los ocho bits, todos están a 0 excepto uno que vale 1 y con ello indica dónde se halla el ascensor. Así, si *Pis_actual[7..0]* vale 00000100 ello significa que el ascensor está en el piso 2.

Revisemos brevemente la funcionalidad del módulo **Pis**. Tal como se ha dicho, el módulo **Pis** informa del piso en el que se halla el ascensor. Para ello, utiliza un contador bidireccional de 3 bits en el que la señal *Direccio* indicará si el ascensor está subiendo (el contador habrá de funcionar en modo ascendente) o bajando (el contador habrá de funcionar en modo descendente).

Si *Moviment* vale 1 el ascensor se mueve en la dirección indicada por *Direccio*. Si *Moviment* vale 0 el ascensor permanece parado (y el valor de *Direccio* es irrelevante). Cuando el ascensor se mueve requiere de un tiempo para ir de un piso al siguiente. Para saber cuándo el ascensor llega al piso siguiente, **en un caso real habría un sensor que detectaría la llegada del ascensor al piso y emitiría una señal indicadora de ello**. Para no complicar el diseño del controlador del ascensor con el uso de más señales de control, **se va a hacer una importante simplificación**: se va a considerar que el ascensor tarda un tiempo **fijo** en moverse de un piso al siguiente, un tiempo simbólico (ficticio) de cuatro ciclos de reloj, el cual será contabilizado por un contador ascendente de 2 bits que formará parte del controlador, en concreto del mismo módulo **Pis**.

Por lo expuesto en el párrafo anterior, cuando *Moviment* = 1 y *Direccio* = 1, cada cuatro ciclos de reloj el piso actual aumenta en 1. De igual manera, cuando *Moviment* = 1 y *Direccio* = 0, el piso actual disminuye en 1.

Para concluir, se ha de recordar que siempre que *RST* valga 1, los dos contadores se inicializarán asíncronamente con el valor 0.

1.1 DISEÑO

La funcionalidad expuesta en el punto anterior la vamos a conseguir de la siguiente manera:

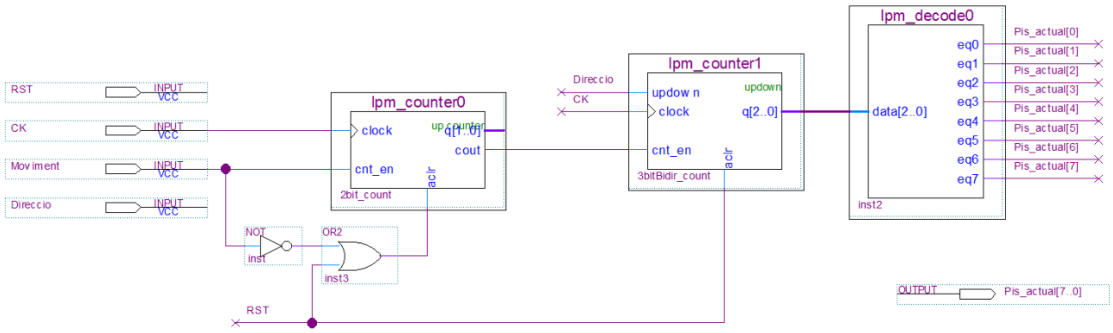


Figura 1

3bitBidir_count es un contador bidireccional de 3 bits que, dependiendo de la señal *Direccio*, cuenta hacia arriba (*Direccio*=1) o hacia abajo (*Direccio*=0). Este contador nos dirá en cada momento el piso en el que se encuentra el ascensor. El ascensor tarda 4 ciclos de reloj en moverse de un piso al siguiente y, por tanto, este contador **no** debe cambiar de estado cada ciclo de reloj, sino cada 4 ciclos. Para ello añadimos un contador de 2 bits (**2bit_count**), que tiene una salida *cout* que utilizaremos como entrada de *cnt_en* (count_enable) del contador bidireccional. La señal que sale por *cout*[1] se pone a 1 cuando el estado del contador es 11. Sólo entonces (esto es, cada cuatro ciclos de *CK*) el contador bidireccional recibirá un 1 por su entrada *cnt_en* que le permitirá aumentar en 1 su valor cuando llegue el siguiente flanco de reloj.

Cuando *Moviment*=0, el ascensor permanece parado. En el esquemático puedes ver cómo cuando *Moviment*=0 la entrada *cnt_en* (cout_enable) del contador recibe un 0 y, por tanto, el contador deja de contar. La señal *Moviment* llega, además, a la entrada *aclr* (clear asíncrono), para que el contador se ponga a 00. Fíjate que se ha incluido un inversor porque *aclr* debe recibir un 1 cuando *Moviment* = 0.

Por último, la salida del contador bidireccional (que no es más que el piso actual, pero descrito como un número binario de 3 bits) se lleva a un decodificador de 3-a-8 (**Decoder**) que genera el bus *Pis_actual[7..0]* (el cual ya conocéis desde la primera sesión de prácticas).

[1] *cout* suele recibir muy frecuentemente el nombre de *TC*, Terminal Count, indicando que el contador ha llegado a su valor más alto.

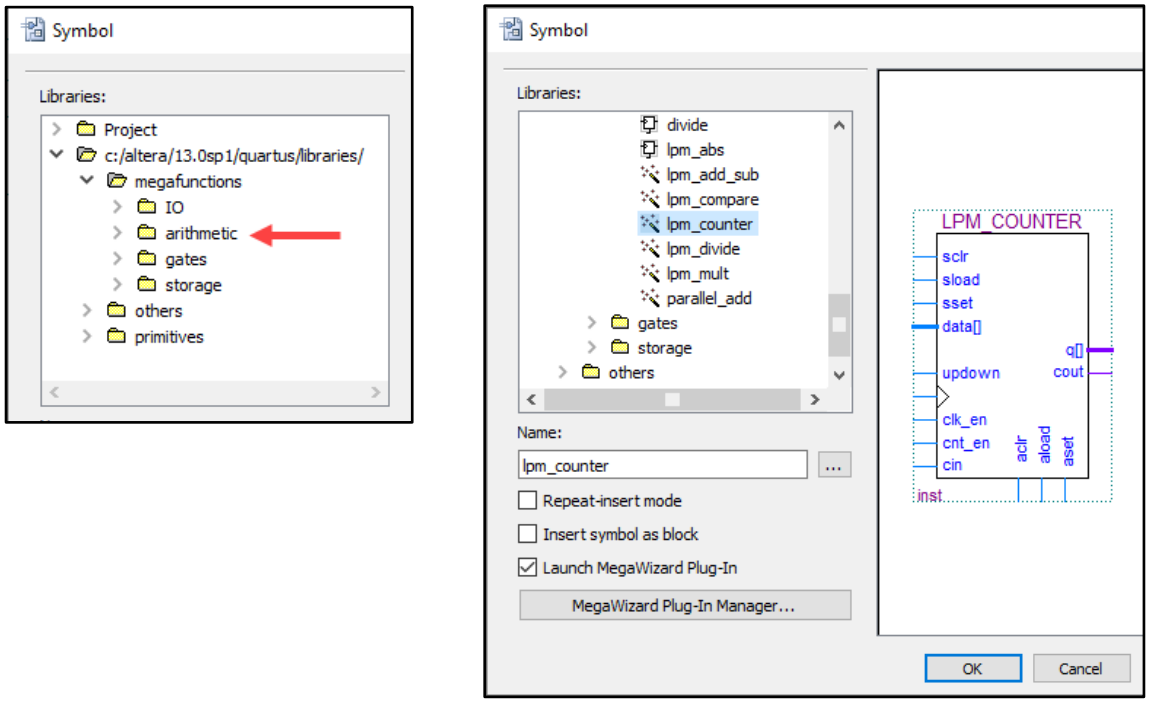
1.2 ENTRAR EL ESQUEMÁTICO

¿Cómo generamos los dos contadores y el decodificador?

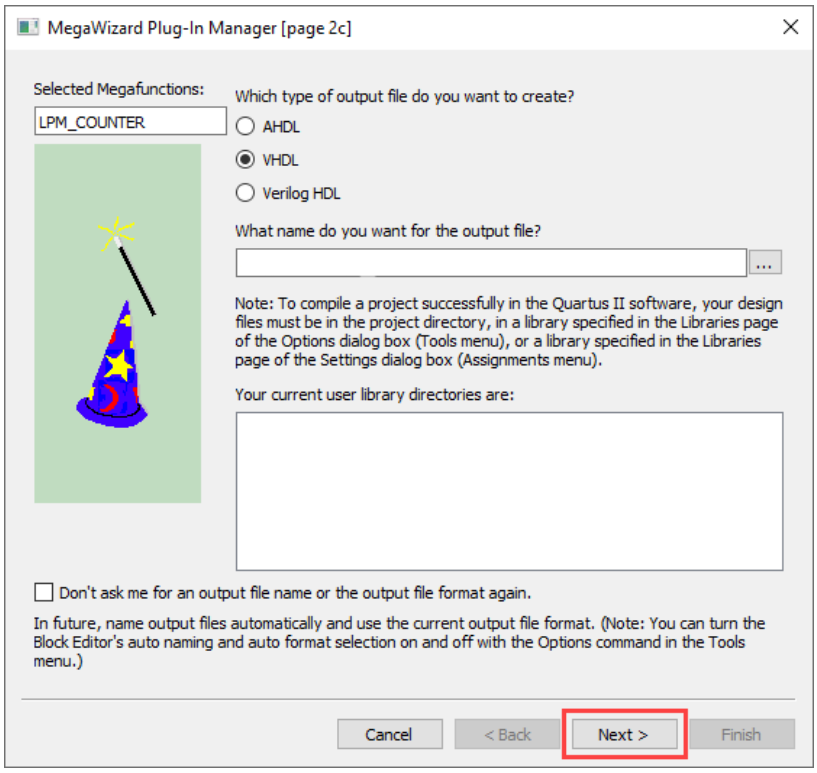
Para ello vamos a utilizar la biblioteca de *megafunciones* de Quartus. Las *megafunciones* son circuitos de una cierta complejidad en los que podemos definir muchas de sus características. Quartus sabe interpretar dichas características y generar su estructura interna de una manera automática y transparente al usuario.

Para generar el contador de 2 bits haremos lo siguiente:

1) Llamaremos a la *megafunción* **lpm_counter** abriendo la librería *megafunctions→arithmetic* (ver figura), seleccionando el módulo **lpm_counter** y clicando en ***OK***.

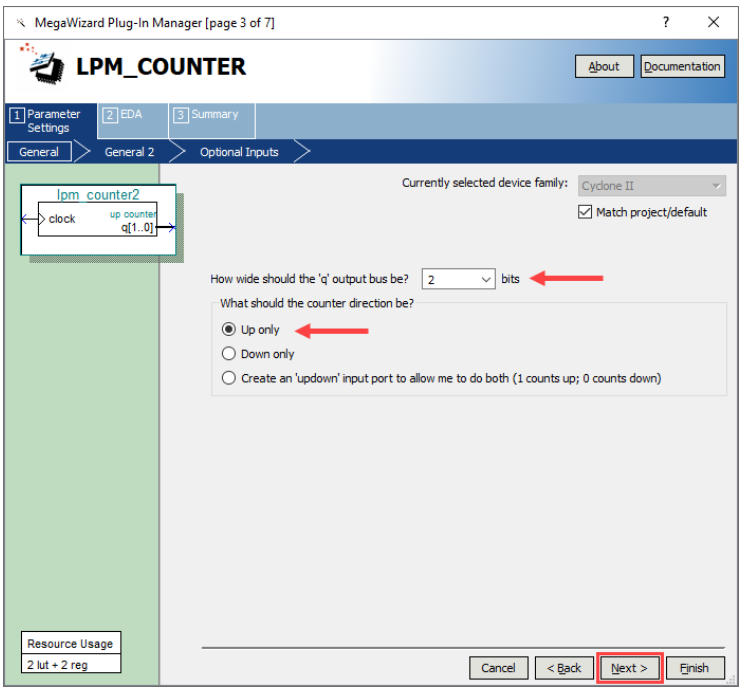


2) Aparecerán una serie de menús, el gestor de *megafunciones*, que nos permitirán personalizar nuestro contador. La primera imagen será algo como esto:

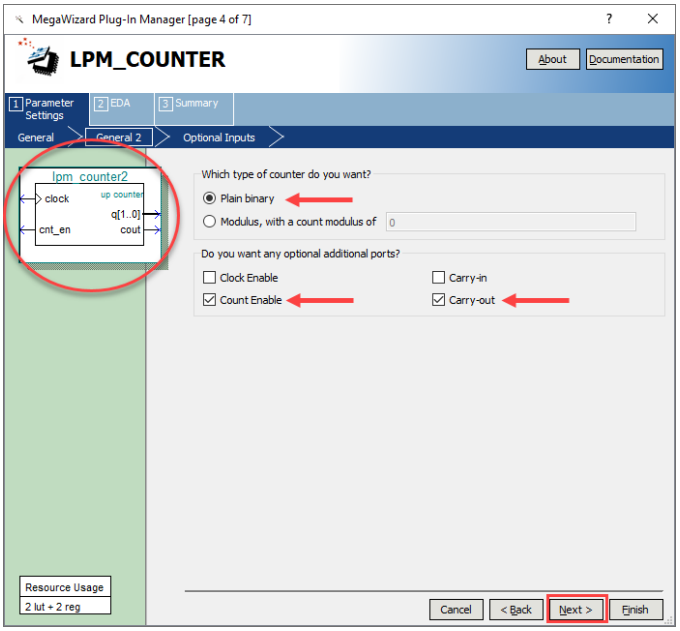


Asegúrate que en esta pantalla esté seleccionado **VHDL** (es el lenguaje de descripción de hardware que utilizaremos) y clica en ***Next***.

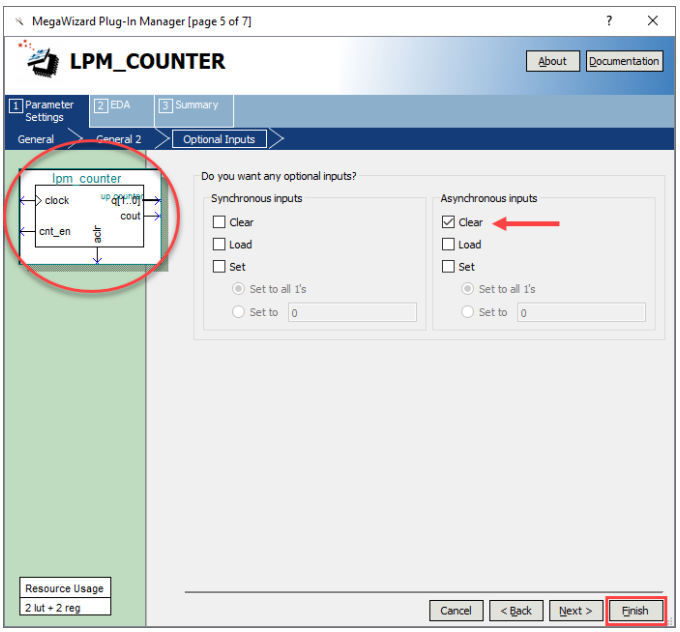
3) A continuación, te aparecerá una nueva pantalla en la que deberás definir el tipo de contador que quieres usar (ascendente, descendente o bidireccional) y número de bits del contador. En este caso selecciona **2 bits** y **Up only** (ascendente) en los sitios que te indican las flechas y clica en ***Next***.



4) En el siguiente paso hay que especificar si el contador debe contar en binario hasta 2^n-1 (Plain binary) o hasta un valor predeterminado (Modulus), y si debe incorporar una señal *Count Enable* que congele el conteo cuando *Count Enable* = 0. La señal *Carry-out* tiene la funcionalidad de "terminal count" (*TC*), es decir, toma el valor 1 cuando el contador llega a su valor máximo. En la figura siguiente te hemos marcado las características deseadas (contador "Plain binary", con salida *TC*, y con entrada *Count Enable*). No necesitamos el resto de señales. Fíjate que conforme vamos definiendo las señales que vamos a utilizar, van apareciendo como entradas o salidas en el dibujo de la esquina superior izquierda. Finalmente, pulsa el botón **Next**.

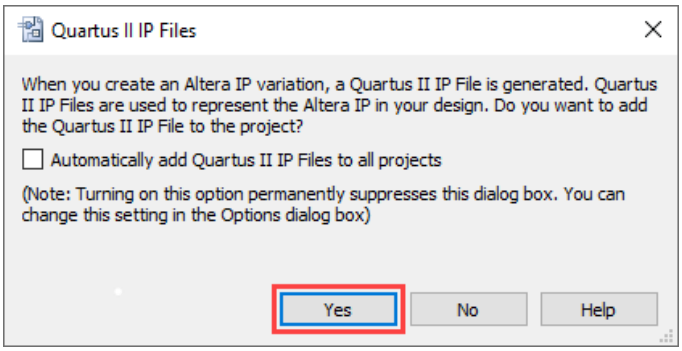


5) A continuación le diremos si queremos entradas de Reset (Clear), Set o Load, síncronas o asíncronas. Nosotros utilizaremos sólo el *Clear* asíncrono:



Las ventanas restantes no nos interesan, así que puedes hacer clic directamente en ***Finish***. Se muestra una ventana con un resumen del contador. En esa ventana haz de nuevo clic en ***Finish***.

Finalmente aparece una ventana en la que se nos pregunta si queremos añadir el *Quartus II IP File* al proyecto. Responde afirmativamente pulsando el botón ***Yes***.



Al completar el diseño, el módulo nos aparecerá en el editor de esquemas.

Para generar el contador bidireccional de 3 bits seguiremos el mismo procedimiento, con las siguientes diferencias:

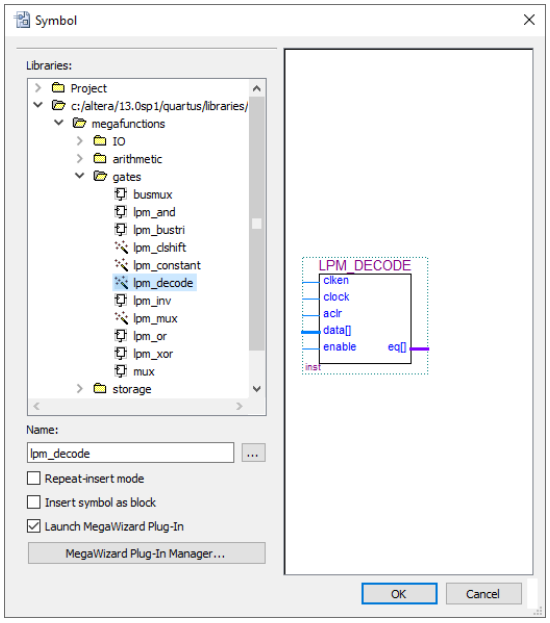
a) En el paso 3 le hemos de decir que el contador será de 3 bits y que incorpore una entrada "*updown*" porque se trata de un contador bidireccional (utiliza la última opción del cuadro "What should the counter direction be?")

b) En el paso 4 le hemos de decir que el contador es de tipo "Plain binary", y marcar *Count Enable* (en este caso no necesitamos para nada la salida *cout*).

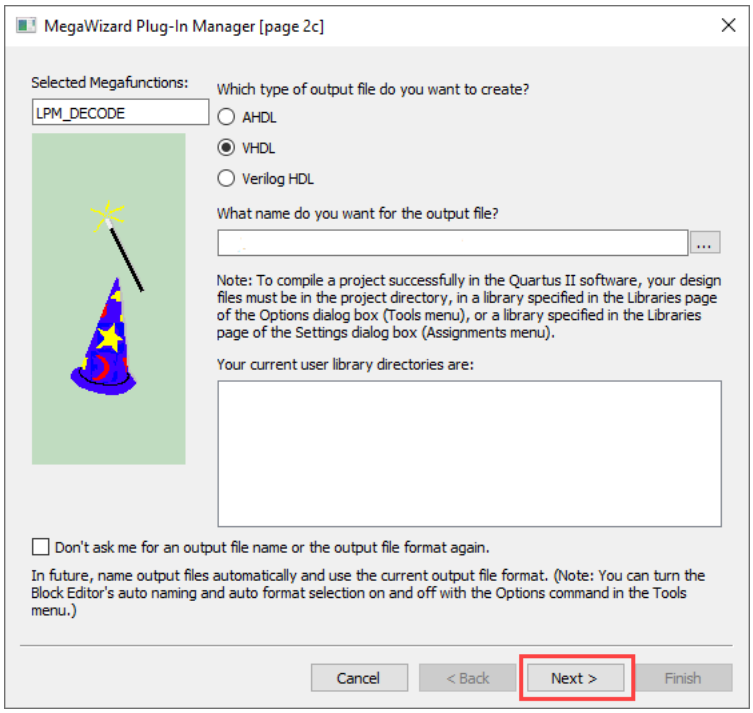
c) El paso 5 lo replicaremos tal cual, es decir, introduciremos un *clear* asíncrono.

Para generar el decodificador vamos a tener que utilizar una *megafunción* diferente:

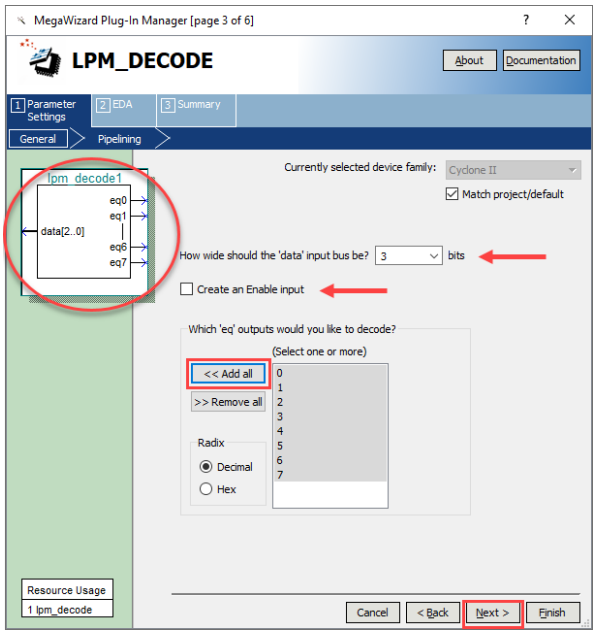
1) Llamaremos a la *megafunción* **lpm_decode** abriendo la librería *megafunctions→gates* (ver figura), seleccionando el módulo **lpm_decode** y clicando en ***OK***.



2) El paso 2 es idéntico al seguido con los contadores

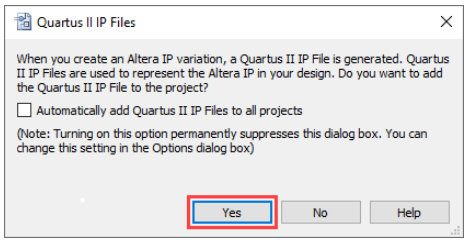


3) A continuación le diremos el número de entradas (3). Nuestro decodificador no tendrá, y por tanto **no** seleccionaremos, ninguna entrada de *Enable*. Asimismo, se indicará que queremos que se generen las 8 salidas del decodificador **(<< Add all**). A continuación, pulsa el botón **Next**.



4) Y eso es todo, el resto de las ventanas hemos de dejarlas tal cual. Clica en **Finish**. Se muestra una ventana con un resumen del decodificador. En esa ventana haz de nuevo clic en ***Finish***.

Finalmente aparece una ventana en la que se nos pregunta si queremos añadir el *Quartus II IP File* al proyecto. Responde afirmativamente pulsando el botón ***Yes***.



Al completar el diseño, el módulo nos aparecerá en el editor de esquemas.

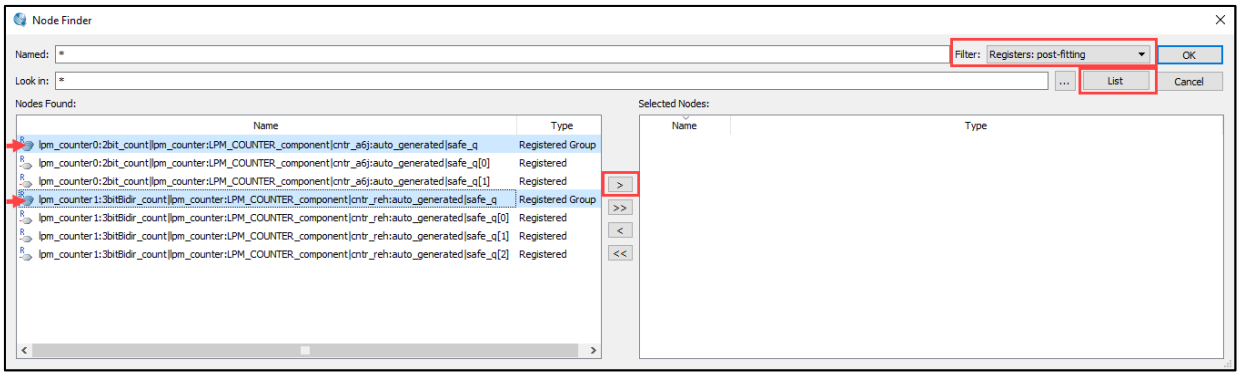
Con esto tendrás los tres módulos en el editor de esquemas y sólo te faltará introducir las entradas y salidas y conectar todo tal como te indicaba la **figura 1**. Fíjate que has de dar los nombres de instancia **2bit_count** y **3bitBidir_count** al contador de 2 bits y al contador de 3 bits respectivamente.

Una vez tengas todo el esquemático, guárdalo en el fichero **Pis.bdf** en la carpeta del proyecto (U:\FCPract\4XX_YY). A continuación, **crea un símbolo para el módulo Pis** que necesitaremos cuando montemos el controlador completo. Finalmente, **compílalo** y asegúrate de que no hay errores.

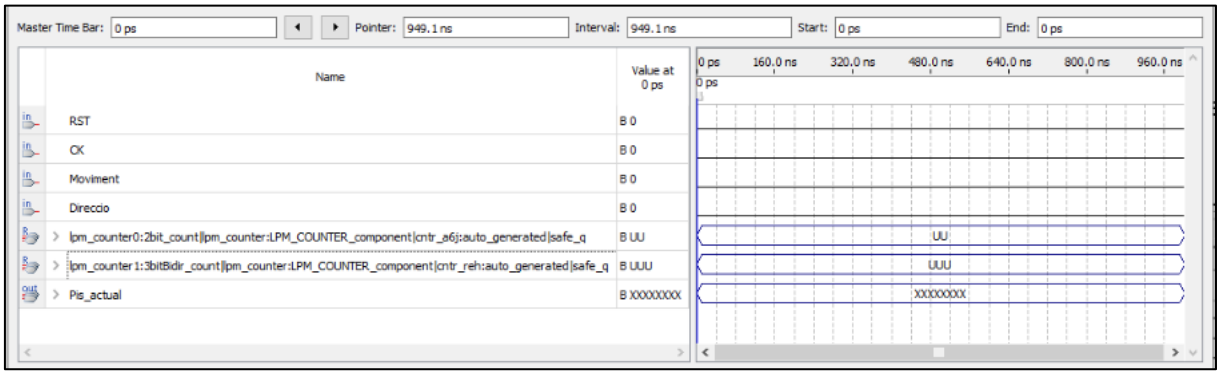
1.3 SIMULACIÓN

A diferencia de prácticas anteriores, en esta práctica vas a ser tú el que diseñe la estrategia de test y genere las formas de onda de la simulación. El conjunto de vectores de simulación ha de ser lo suficientemente completo como para garantizar (razonablemente) que el circuito funciona correctamente.

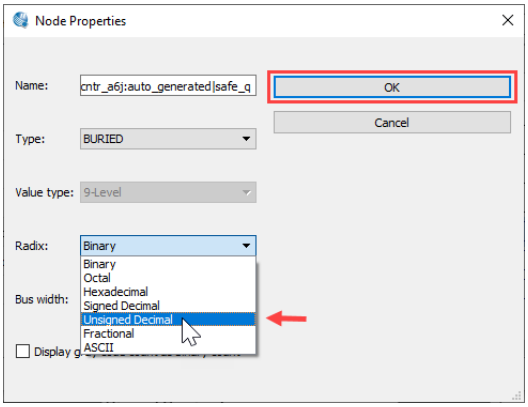
Crea un fichero de estímulos vacío y añádeles todas las entradas (*RST*, *CK*, *Moviment* y *Direccio*) y la salida (el bus *Pis_actual*). Asimismo, para facilitar el seguimiento de la simulación se debe añadir al fichero de estímulos la salida de ambos contadores. Para hacerlo, cuando introduzcas las señales a simular en el editor de formas de onda, selecciona "**Registers: post fitting**" en el desplegable **Filter**, pulsa el botón **List** y, en la ventana de la izquierda, selecciona las salidas de los contadores que verás con unos nombres larguísimos y poco intuitivos, tal como puedes observar en la figura siguiente.



Posiciona las señales en el fichero de estímulos en el siguiente orden: primero las entradas del módulo (*RST*, *CK*, *Moviment* y *Direccio*), después la salida del contador de 2 bits, después la salida del contador de 3 bits y finalmente la salida del módulo *Pis_actual*, tal como se muestra en la siguiente imagen. Este orden facilitará posteriormente la legibilidad y el seguimiento del resultado de la simulación.



Hay otro aspecto a tener en cuenta respecto a la legibilidad de los resultados de la simulación: el formato con el que se presenta el contenido de los buses. Te recomiendo que los buses correspondientes a la salida de ambos contadores se muestren en formato de número en base 10 no signado (**Unsigned decimal**), mientras que el bus *Pis_actual* se muestre en formato binario (**Binary**). Para fijar el formato, simplemente se selecciona el bus, y después haciendo clic con el botón derecho del ratón se ejecuta **Properties**. En el formulario que se abre, en el campo **Radix** se escoge el formato de visualización deseado para el bus.



Ahora llega el momento de definir los valores que se van a aplicar a las entradas (recuerda que solo se aplican valores a las entradas, nunca a las salidas ni a los nodos internos). Cualquier valor que se aplique a una entrada tiene que durar al menos un periodo de reloj (*CK*), **de flanco de bajada a flanco de bajada del reloj**. Evidentemente, también es posible aplicar un valor durante varios periodos consecutivos de reloj, empezando en un flanco de bajada y terminando en otro flanco de bajada del reloj.

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 µs**. Esto se puede establecer con el comando **Edit → Set End Time...** .

A la hora de preparar los estímulos de entrada para verificar el correcto funcionamiento del módulo, si lo deseas, puedes inspirarte en la siguiente estrategia de test:

1) Comprueba que ambos contadores se ponen a 0 cuando *RST*=1. Parece lógico empezar la simulación reseteando el circuito para así poder comprobar en los resultados de la simulación si los dos contadores se ponen asíncronamente a 0.
2) Mantén la señal *Direccio* a 1 un número de ciclos suficiente para poder comprobar que, cada 4 ciclos de *CK*, el ascensor va subiendo desde el piso 0 hasta el piso 7 (unos 32 ciclos en total).
3) Mantén la señal *Moviment* a 1 para que el ascensor "se mueva", pero en algún momento, mientras *Direccio*=1, pon *Moviment*=0 para comprobar que mientras *Moviment* = 0 el piso donde está el ascensor no cambia de valor. Comprueba, además, que el contador de 2 bits vuelve al estado inicial 0 mientras *Moviment*=0.

(*la estrategia de test continúa en la siguiente página*)

4) Repite este proceso cuando *Direccio*=0. Mientras *Direccio*=0 tendrás que comprobar que el ascensor baja y, de nuevo, que cuando pongas *Moviment*=0 el ascensor se para (el piso queda bloqueado).

Cuando hayas concluido la elaboración de los estímulos de entrada, guarda el fichero de ondas con el nombre **Pis.vwf** en la carpeta del proyecto (U:\FCPract\4XX_YY).

A continuación, realiza una simulación funcional. Analiza los resultados de la simulación para comprobar si el circuito se comporta correctamente. Si el circuito se comporta de la forma esperada, guarda el resultado de la simulación funcional en un fichero con el nombre **Pis_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 en un fichero con el nombre **Pis_timing_sim.vwf** en la subcarpeta simulation\qsim del proyecto **(U:\FCPract\4XX_YY\simulation\qsim)**.

Los ficheros con el resultado de la simulación (**Pis_functional_sim.vwf** y **Pis_timing_sim.vwf**) no se considera que formen parte del proyecto y por ese motivo no aparecen en la sección *Files* del *Project Navigator*. Para añadirlos al proyecto se ejecutará el comando:

Project → Add/Remove Files in Project…

y se seguirá el mismo procedimiento seguido en sesiones anteriores a tal fin.

Si has hecho correctamente el proceso ahora se deberían ver los ficheros simulation/qsim/Pis_functional_sim.vwf y simulation/qsim/Pis_timing_sim.vwf en la sección *Files* del *Project Navigator*.

Entrega correspondiente a esta cuarta sesión

La entrega consistirá en un fichero .zip cuyo nombre será **FC-S4-4XX_YY.zip**, donde **4XX_YY** es el identificador del equipo de trabajo (por ejemplo, **FC-S4-411_01.zip**), que ha de contener cinco ficheros:

- el fichero **Pis.bdf**

- el fichero **Pis.vwf**

- el fichero **Pis_functional_sim.vwf**

- el fichero **Pis_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-S4-4XX_YY.zip** se ha de realizar a través del Campus Virtual.