SESIÓN 5: UC_MOV

Los objetivos de esta práctica son:

- entrar el circuito **UC_MOV**, que forma parte de la Unidad de Control
- insistir en el diseño de la estrategia de test, en este caso en circuitos secuenciales de los que conocemos su grafo de comportamiento. El grafo de comportamiento de **UC_MOV** es más complejo que el que se vio en el caso del módulo **UC_DIR**.

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

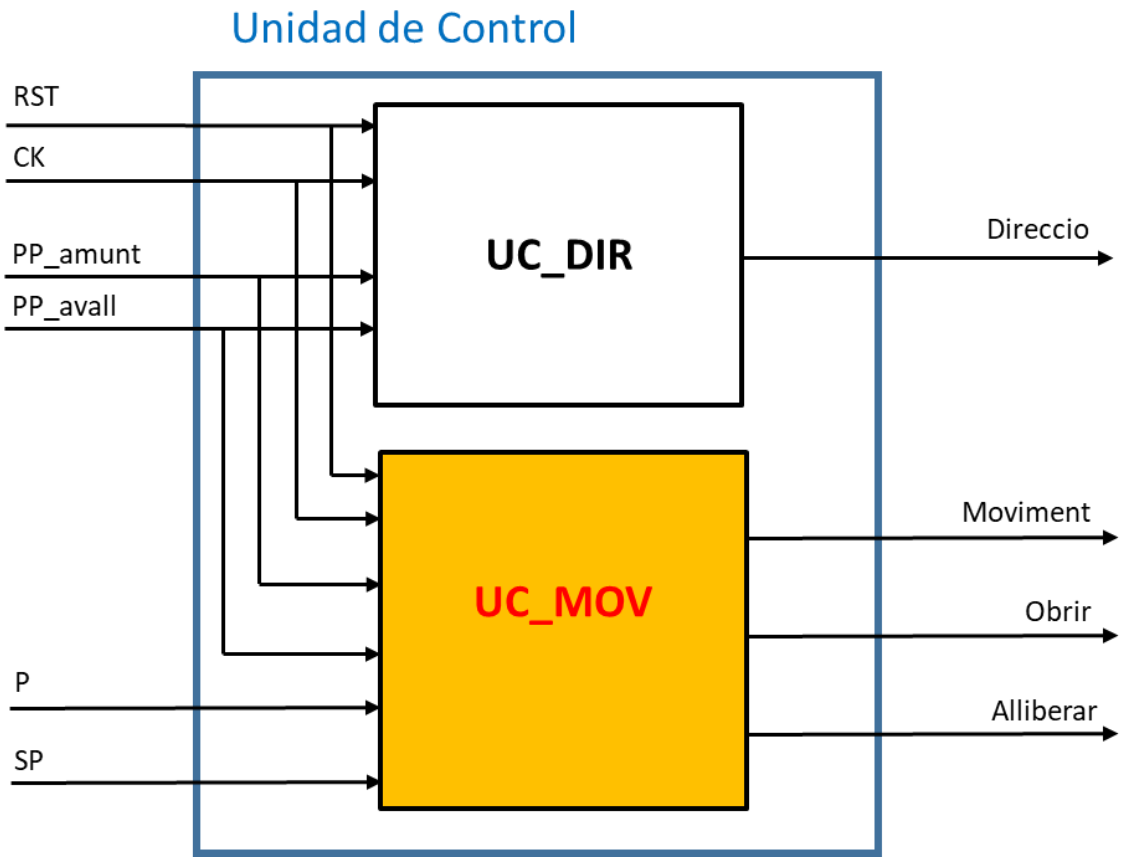
-descargar de la nube la carpeta del proyecto comprimida y moverla a **U:\FCPract**

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

-borrar la carpeta comprimida

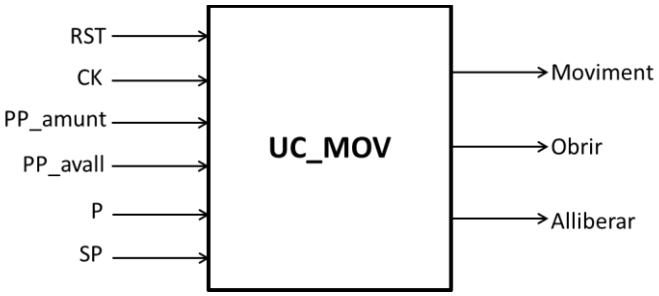
1. MÓDULO *UC_MOV*

El módulo **UC_MOV** es un circuito secuencial que genera las señales de control necesarias para controlar la parte mecánica del ascensor: *Moviment* para decirle al ascensor que se mueva o se quede parado, y *Obrir* y *Alliberar* para que el módulo **Obrir_Alliberar** (hecho en la sesión 1) controle la apertura de la puerta del piso en el que se encuentra el ascensor y *libere* (elimine) dicho piso de la lista de pisos solicitados.



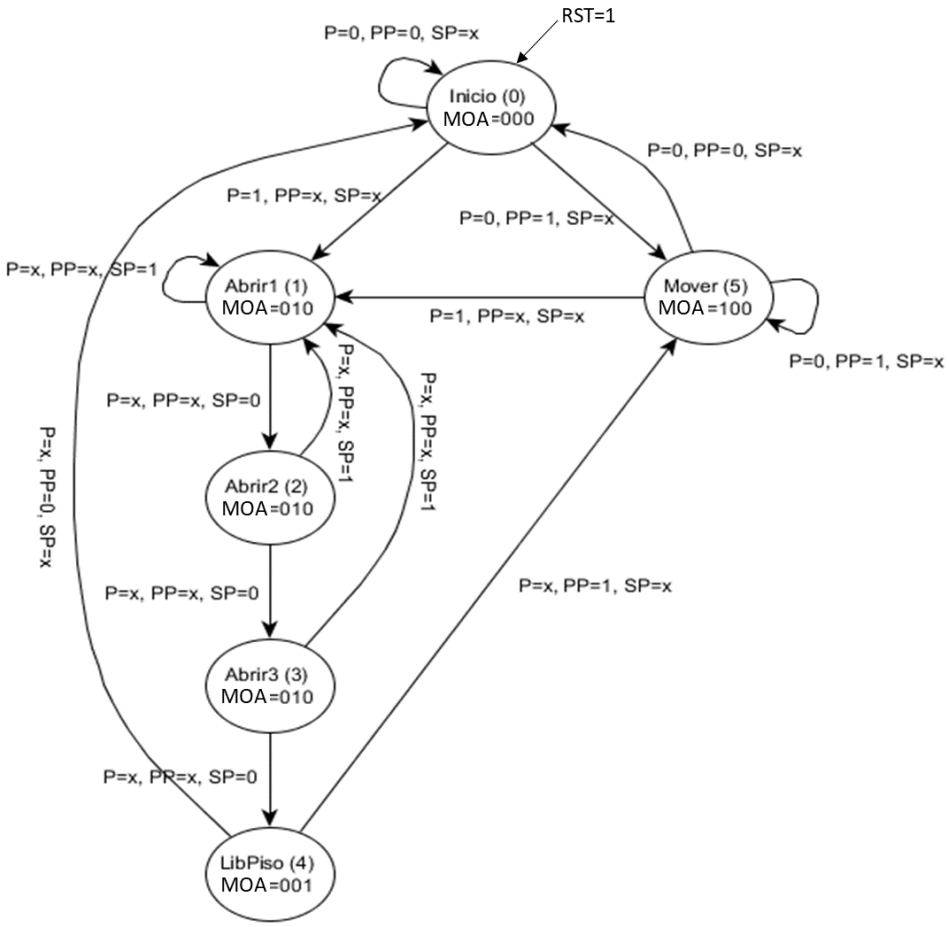
Las entradas *P*, *PP_amunt* y *PP_avall* provienen del módulo **Senyals_UC** (hecho en la sesión 2), e informan de si el piso en el que se encuentra el ascensor está solicitado (P=1), si hay pisos solicitados por encima del piso actual (*PP_amunt*=1), o si hay pisos solicitados por debajo del piso actual (*PP_avall*=1).

La siguiente imagen muestra el símbolo del módulo:



Comentemos brevemente el funcionamiento que debe tener el módulo **UC_MOV**.
El ascensor inicialmente estará parado y con la puerta cerrada. Cuando la señal *PP_amunt* o la señal *PP_avall* indiquen la existencia de solicitudes al ascensor en los pisos superiores o inferiores, el ascensor se ha de mover (hacia arriba o hacia abajo, respectivamente). Cuando el ascensor llegue a un piso y la señal *P* indique la existencia de una solicitud en ese piso, el ascensor debe detenerse y abrir la puerta.
La puerta del ascensor dispone de un sensor (una célula fotoeléctrica) que genera una señal *SP* que toma el valor 1 si detecta que alguien está pasando por la puerta o algo impide su cierre, y 0 en caso contrario. Cuando se abra la puerta porque hemos llegado a uno de los pisos solicitados, la puerta debe permanecer abierta siempre que *SP*=1, más dos ciclos de reloj completos adicionales a partir del momento en el que *SP* se pone a 0. Tras ello, se debe cerrar la puerta del ascensor y eliminar la solicitud relativa a ese piso puesto que ya ha sido atendida.

El funcionamiento descrito para el módulo **UC_MOV** se corresponde al grafo de comportamiento que se muestra en la siguiente página.



1. El circuito puede estar en 6 estados diferentes: **Inicio**, **Abrir1**, **Abrir2**, **Abrir3**, **LibPiso** y **Mover**. Para la representación de esos 6 estados se requieren 3 bits. Los estados serán codificados de la siguiente manera:

   Inicio → 000, **Abrir1** → 001, **Abrir2** → 010, **Abrir3** → 011, **LibPiso** → 100, **Mover** → 101

   Fíjate que la codificación de cada estado aparece escrita en el grafo de comportamiento **en base decimal y entre paréntesis**.

   Cada estado lleva asociado unos valores de las salidas (se trata de una máquina de Moore). Las salidas del circuito, *Moviment*, *Obrir* y *Alliberar* las hemos llamado *M*, *O* y *A* respectivamente, para facilitar su escritura dentro del nodo.

2. El estado **Inicio** es el estado inicial del circuito. Fijémonos que se ha codificado como 000. Cuando la señal *RST* toma el valor 1, el estado de **UC_MOV** se pone a 000 (Inicio).

3. Las entradas son P, PP_amunt, PP_avall y SP. En el grafo, hemos llamado PP a la suma (OR lógica) de PP_amunt y PP_avall, de modo que PP=1 cuando PP_amunt, PP_avall o ambas valen 1. Cuando escribimos que alguna de estas señales es igual a x significa que la transición se produce sea cual sea el valor (0 o 1) de dicha señal.

4. En el estado inicial (**Inicio**), todas las salidas toman el valor 0. En este estado el ascensor está parado. Si P=PP=0 significa que no hay ningún piso solicitado y, por tanto, el ascensor debe seguir parado, sin que se abran las puertas, y sin necesidad de liberar ningún piso, sea cual sea el valor del sensor de puerta SP. **UC_MOV** se mantiene en el estado Inicio.

5. Estando en el estado inicial (**Inicio**), si P=0 pero hay algún piso seleccionado (PP_amunt o PP_avall tiene el valor 1) el ascensor debe empezar a moverse, para lo cual pasa al estado **Mover** y pone la señal $Moviment$ a 1. El sentido del movimiento, hacia arriba o hacia abajo, lo determinará el módulo **UC_DIR** (hecho en la sesión 3).

6. En el estado **Mover**, si se llega a un piso que ha sido solicitado (*P* = 1) **UC_MOV** pasa al estado **Abrir1** en el que se detiene el ascensor y se abre la puerta (*M* = 0, *O* = 1).

7. Una vez abierta la puerta, si SP=1 significa que hay alguien saliendo o entrando del ascensor (o algo obstruye la puerta) y la puerta debe seguir abierta: **UC_MOV** se queda en el estado **Abrir1**. Si, por el contrario, SP=0, **UC_MOV** debe esperar dos ciclos completos de reloj con la puerta abierta; de ahí los estados **Abrir2** y **Abrir3**. Si, mientras la puerta está abierta, se detecta un nuevo SP=1, la máquina vuelve a **Abrir1** para reiniciar el conteo de los ciclos.

8. Pasados estos dos ciclos de seguridad, **UC_MOV** pasa al estado **LibPiso**, donde se da la orden al módulo **Obrir_Alliberar** de cerrar la puerta (*O*=0) y de liberar (eliminar) el piso actual de la lista de pisos solicitados (*A*=1).

9. Una vez liberado el piso, si todavía quedan pisos seleccionados (PP=1), **UC_MOV** pasa al estado **Mover** o, si no los hay, vuelve al estado **Inicio**.

1.1 ESQUEMÁTICO

Puesto que el grafo tiene 6 estados, necesitamos 3 bits para codificarlos y, por tanto, 3 flip-flops D para su almacenamiento. A los estados de dichos flip-flops los llamaremos q_2, q_1, q_0 (q_2 es el más significativo), y a sus entradas las llamaremos respectivamente D_2, D_1 y D_0. Tras hacer la correspondiente tabla de transición de estados y la tabla de salidas (no mostradas en este guion), con la ayuda de la aplicación "Analizar Circuito" de VerilUOC_Desktop, hemos calculado las funciones de entrada a los flip-flops y las funciones de las salidas:

$$D_2 = (PP\_amunt + PP\_avall).(\bar{q}_1.\bar{q}_0.\bar{P} + q_2.\bar{q}_0 + q_2.\bar{P}) + q_1.q_0 \cdot \overline{SP}$$

$$D_1 = \bar{q}_2.\bar{q}_1.q_0 \cdot \overline{SP} + q_1.\bar{q}_0 \cdot \overline{SP}$$

$$D_0 = (PP_{amunt} + PP_{avall}).(\bar{q}_0 + q_2) + \bar{q}_2.\bar{q}_0 \cdot P + q_2.q_0.P + \bar{q}_2.q_0.SP + q_1.\bar{q}_0$$

$$M = Moviment = q_2.q_0$$

$$O = Obrir = q_1 + \bar{q}_2.q_0$$

$$A = Alliberar = q_2.\bar{q}_0$$

Para que no tengas que invertir mucho tiempo en hacer el esquemático del circuito, en el Campus Virtual tienes un esquemático parcialmente realizado. Bájate el fichero UC_MOV.bdf, añádelo a tu proyecto y complétalo. En las siguientes líneas te decimos cómo hacerlo.

- En primer lugar, descarga el fichero **UC_MOV.bdf** que encontrarás en Pràctiques → Enunciats → Material Sesión 5 en el ordenador.

- Al igual que ya hiciste en la sesión 2, arrastra el fichero **UC_MOV.bdf** desde donde lo has descargado al área de trabajo del editor de esquemas de Quartus II y, a continuación, guárdalo en la carpeta de tu proyecto de Quartus, ejecutando:

File → Save As…

En el formulario que se abre debes seleccionar la carpeta del proyecto (U:\FCPract\4XX_YY).

- Finalmente debes completar el esquema al cual le falta el flip-flop 0 (y el inversor que genera *nq0*), y la implementación de las funciones de la señal *D2* y de la salida *Obrir*. Al flip-flop 0, recuerda darle como nombre de instancia **ff0** (ello se hace seleccionando el flip-flop, haciendo clic con el botón derecho del ratón sobre él, ejecutando **Properties** e introduciendo **ff0** en el campo **Instance name**). Asimismo, cuando implementes *D2*, recuerda que *PP = PP_amunt + PP_avall* y que la señal *PP* ya la tienes implementada en el circuito.

Una vez hayas completado el módulo **UC_MOV**, guárdalo en la carpeta del proyecto (U:\FCPract\4XX_YY). A continuación, defínelo como *top-level entity* y realiza su compilación (solo deberían aparecer 8 *warnings*). Si se produce algún error durante la compilación o más *warnings* de los esperados, deberás corregirlo. Finalmente, genera un símbolo para la celda **UC_MOV** y guárdalo en la carpeta del proyecto (U:\FCPract\4XX_YY).

Ahora has de comprobar que el circuito implementa correctamente el grafo de comportamiento visto en la página 4. Para ello has de realizar la simulación del circuito, aspecto sobre el que hablamos en el siguiente apartado.

1.2 SIMULACIÓN

En este caso vas a tener que definir tú solo el conjunto de vectores de simulación que te permita comprobar razonablemente que el circuito funciona correctamente. Existen muchas maneras de hacerlo, pero, en el caso de circuitos secuenciales en los que conozcas el grafo de comportamiento, puedes seguir las siguientes indicaciones:

Obligatoriamente debes comprobar, al menos, que …

1. **El circuito se inicializa correctamente**. En el caso que nos ocupa, los tres flip flops deben ponerse a 0 cuando RST=1. Compruébalo en la simulación.

2. Las formas de onda que definas deben garantizar, al menos, que (a) **el circuito pase por todos los estados**, y (b) **que el circuito pase por todas sus transiciones** (**atención**: para que la simulación no sea excesivamente larga, cuando una de las transiciones se pueda hacer con diversas combinaciones de valores de las entradas, solo es necesario que compruebes la transición con una de esas combinaciones).

Teniendo en cuenta lo que se indica en los puntos anteriores tienes que definir una sucesión de combinaciones de valores a las entradas que haga que el circuito pase por todos los estados al menos una vez y haga todas las transiciones entre los estados al menos una vez. Evidentemente la simulación habrá de empezar con un reset (RST = 1) para que el circuito secuencial se inicialice en el estado 0 (los tres flip-flops habrán de adoptar el valor 0). Veamos cómo **podría empezar** la simulación (cada valor aplicado a las entradas ha de durar un periodo de reloj, de flanco de bajada a flanco de bajada de reloj):

RST = 1 — Ponemos la máquina en el estado **Inicio** (0)

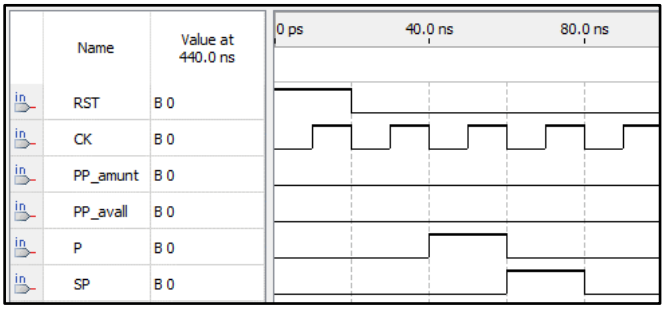
RST = 0 (a partir de aquí RST = 0), PP_amunt = 0, PP_avall = 0, P = 0, SP = 0 — ↑ **Inicio** (0)

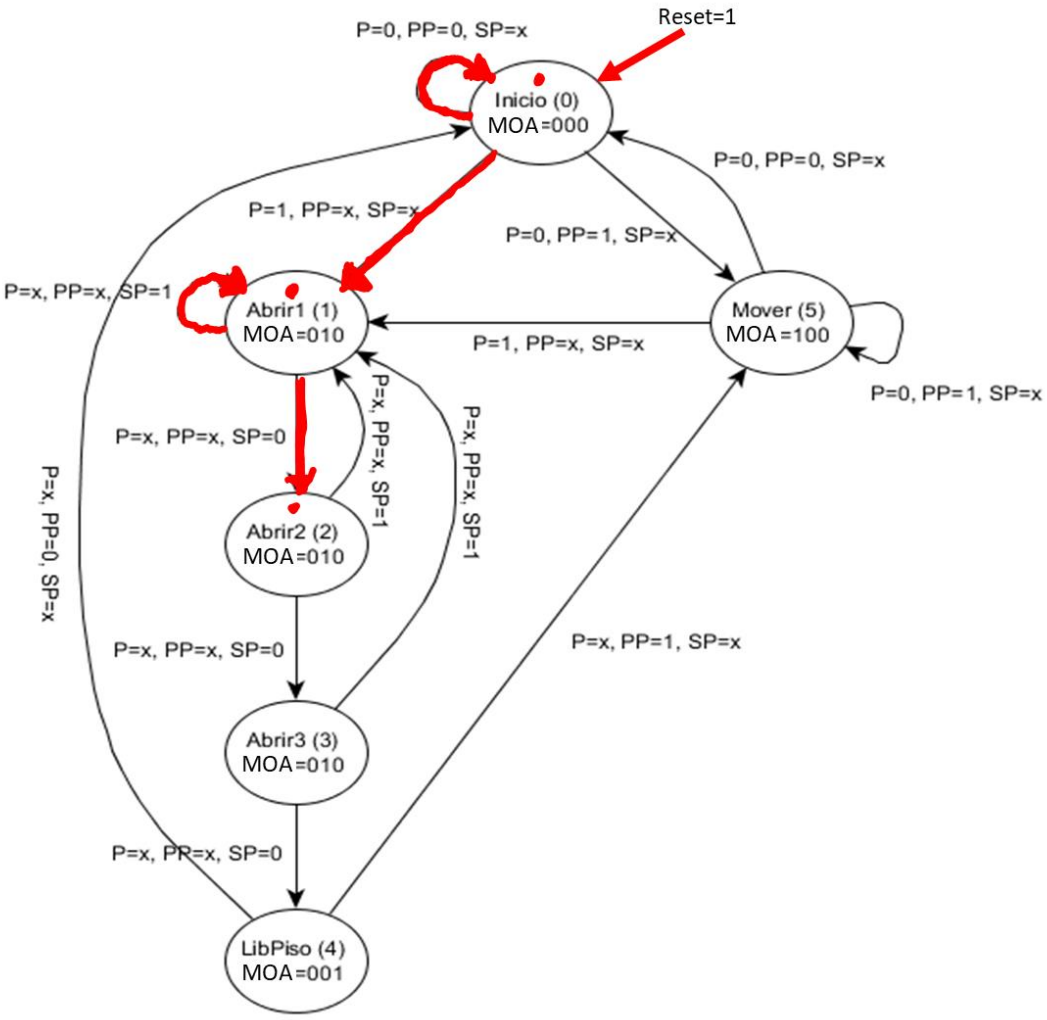
PP_amunt = 0, PP_avall = 0, P = 1, SP = 0 — ↑ **Abrir1** (1)

PP_amunt = 0, PP_avall = 0, P = 0, SP = 1 — ↑ **Abrir1** (1)

PP_amunt = 0, PP_avall = 0, P = 0, SP = 0 — ↑ **Abrir2** (2)

Las cinco líneas anteriores corresponden a los estímulos aplicados durante los cinco primeros periodos de reloj y con ellos se habría pasado por los estados **Inicio** (cuya codificación es 0), **Abrir1** (1) y **Abrir2** (2) (siempre tras el flanco de subida de reloj [↑]). Asimismo, se habrían realizado las transiciones entre estados marcadas en rojo (ver grafo en página siguiente). A continuación, se puede ver una captura de esos estímulos en el fichero de *waveforms*.



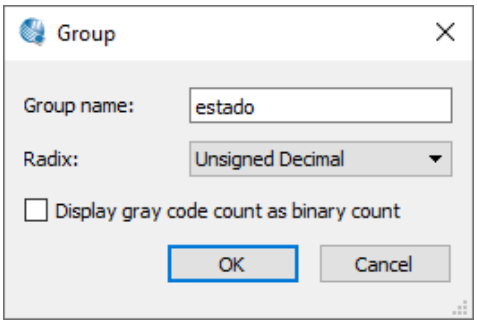


Tú tienes que ampliar la secuencia de cinco combinaciones de valores en las entradas que hemos mostrado en la página anterior a una secuencia que permita pasar por todas las transiciones entre estados y evidentemente pasar por todos los estados.

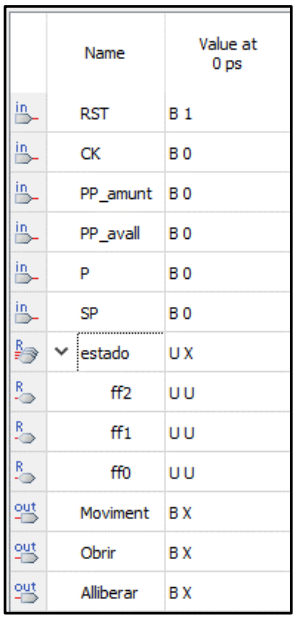
Una vez tengas pensada la secuencia completa de combinaciones de valores en las entradas que utilizarás, …

- Introduce el conjunto de vectores de test en el editor de formas de onda (*waveform editor*) y guárdalo en el fichero **UC_MOV.vwf** en la carpeta del proyecto (U:\FCPract\4XX_YY). Para ello ten en cuenta lo siguiente:
  - En el fichero de *waveforms*, has de añadir las entradas *RST*, *CK*, *PP_amunt*, *PP_avall*, *P* y *SP*, los flip-flops *ff2*, *ff1* y *ff0*, y las salidas *Moviment*, *Obrir* y *Alliberar*.
  - Recuerda que para seleccionar las entradas y salidas puedes usar el filtro **Pins: all** y para seleccionar los flip-flops el filtro **Registers: post-fitting**.
  - Forma un grupo con los flip-flops ff2, ff1 y ff0 (respeta este orden) y denomina a ese grupo **estado**. 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 y como **Radix** Unsigned Decimal. Esto permitirá que durante la simulación podamos ver fácilmente en qué estado se encuentra el circuito secuencial (estado mostrará la codificación del estado expresada en base decimal). Finalmente pulsa el botón **OK**.



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



- La señal CK debe ser una señal periódica de 20 ns.

- Simula funcionalmente el circuito
- Comprueba si funciona bien y, en caso contrario, detecta y corrige los fallos. Si detectas fallos tendrás que modificar convenientemente el esquemático. Recuerda que tras modificarlo deberás compilarlo de nuevo. Para comprobar que el circuito funciona correctamente, comprueba primero que todas las transiciones de estados se realizan correctamente. Si detectas un error en una transición fíjate qué flip-flop adopta un valor erróneo. Muy probablemente, el error se halle en el subcircuito que genera la función de estado siguiente que conectamos a su entrada D. Una vez hayas comprobado que todas las transiciones de estados se realizan correctamente,

comprueba que el valor de las salidas en cada estado es el correcto. Si detectas que una salida es errónea, muy probablemente el error esté en el subcircuito que genera dicha salida.

- Cuando estés seguro de que el circuito funciona bien, guarda el resultado de la simulación en un fichero con el nombre **UC_MOV_sim.vwf** en la subcarpeta simulation\qsim del proyecto **(U:\FCPract\4XX_YY\simulation\qsim)**. A continuación, añade dicho fichero al proyecto. Tras ello, se debería ver el fichero simulation/qsim/UC_MOV_sim.vwf en la sección *Files* del *Project Navigator*.

2. ANÁLISIS DE SIMULACIONES

En la carpeta de material para la sesión 5 encontrarás dos ficheros con los resultados de la simulación de dos implementaciones distintas del módulo **UC_MOV**. Cada una de esas implementaciones tenía un error distinto. Por eso, los ficheros se llaman **err1_sim.vwf** y **err2_sim.vwf**. Analiza esos ficheros con los resultados de la simulación y determina cuál es el **primer** comportamiento anómalo que se observa en cada uno de ellos. Asimismo, reflexiona y deduce en qué parte concreta del circuito podría estar el error que provoca el comportamiento anómalo observado.

Entrega correspondiente a esta quinta sesión

La entrega consistirá en un fichero .zip cuyo nombre será **FC-S5-4XX_YY.zip**, donde **4XX_YY** es el identificador del equipo de trabajo (por ejemplo, **FC-S5-411_01.zip**), que ha de contener cuatro ficheros:

- el fichero **UC_MOV.bdf**

- el fichero **UC_MOV_sim.vwf**

- un fichero PDF en el que se indique el **primer** comportamiento anómalo detectado en el análisis de los ficheros **err1_sim.vwf** y **err2_sim.vwf** (el primer comportamiento anómalo de cada fichero), y se explique en qué parte del circuito puede encontrarse el error que provoca cada uno de ellos.

- un fichero denominado **autoría.txt** con el nombre y el NIU del autor/autores de la entrega.

La entrega del fichero **FC-S5-4XX_YY.zip** se ha de realizar a través del Campus Virtual.