Práctica 5

Uso de DCMs y simulación con tiempos

Objetivos

Realizar un ejemplo sencillo de uso de un DCM (*Digital Clock Manager*), bloque de hardware dedicado en FPGAs de Xilinx, usado para la gestión de relojes. (Para más información sobre este bloque puede consultarse la nota de aplicación de Xilinx disponible en la web del laboratorio).

Aprovechar el mismo circuito para ver un ejemplo de simulación "post-layout", entendiendo también el proceso de "retroanotación" desde un fichero SDF.

Circuito utilizado

Utilizaremos el circuito "stopwatch" de la Práctica 2, elevando su frecuencia de reloj de 50 a 125 MHz. Para ello insertaremos entre la entrada de reloj y los distintos bloques un DCM funcionando como DFS (*Digital Frequency Synthesizer*), con un factor de multiplicación / división de 5/2. La idea no es en este caso que el cronómetro funcione correctamente, sino verlo funcionar 2.5 veces más rápido de lo normal debido al cambio de la frecuencia de reloj.

Guía para la realización

Preparación del proyecto

Crear el directorio de trabajo: C:\DIE\P5_stopwatch_DCM.

Copiar dentro los directorios *rtl/* y *sim/* del cronómetro. Crear también un directorio *ise* y copiar dentro el fichero *stopwatch.ucf*.

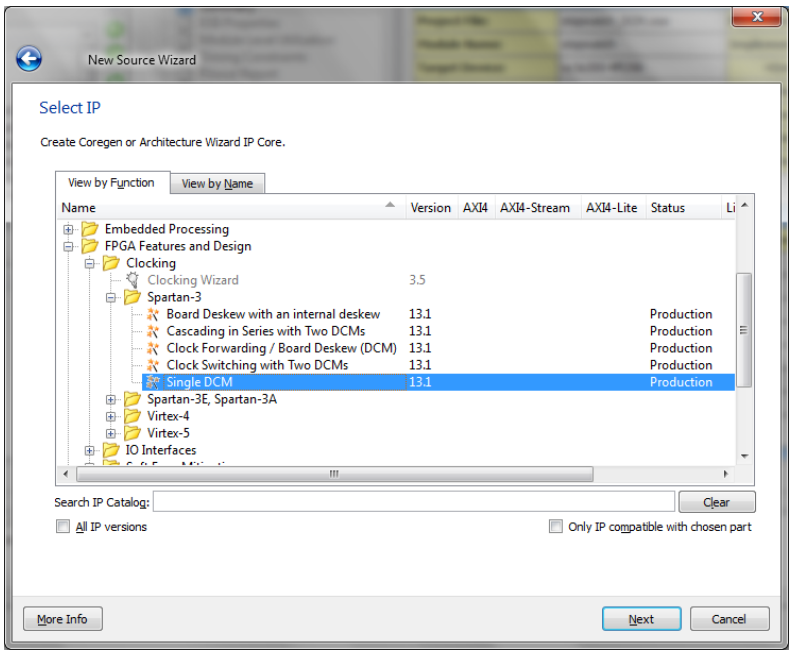
Abrir la herramienta ISE y crear un nuevo proyecto "stopwatch_DCM" ubicado en el directorio *ise* recién creado. Asegurarse de seleccionar el dispositivo de la placa: familia *Spartan3*, dispositivo *XC3S200*, encapsulado *FT256*, grado de velocidad *-4*.

Añadir todos los ficheros fuente existentes: RTL, testbench y fichero *.ucf*.

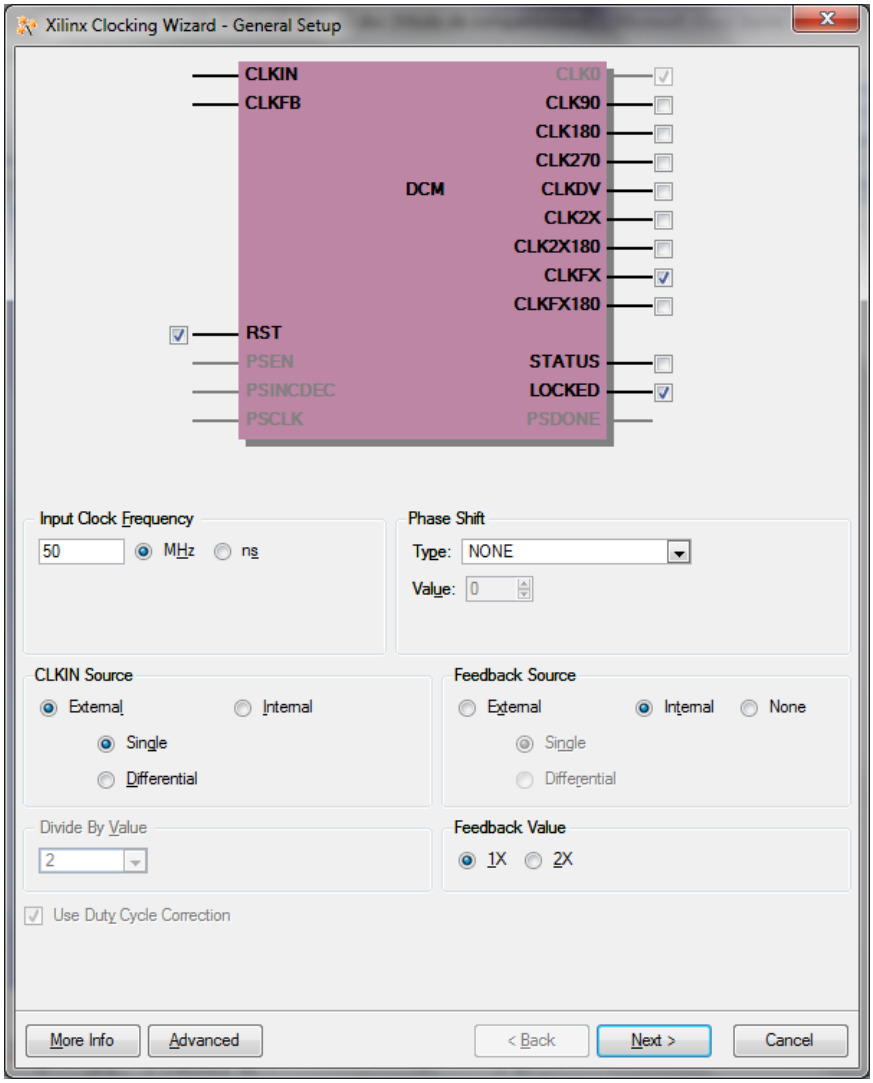
Añadir un DCM

Añadir un nuevo elemento al diseño usando *Project* | *New Source…* . Elegir el tipo *IP (Core Generator & Architecture Wizard)*, el nombre "clockgen_dcm" y el directorio por defecto *ise/ipcore_dir*.

En la ventana *New Source Wizard* que se abre, seleccionar, como se muestra en la siguiente figura: *FPGA Features and Design* | *Clocking* | *Spartan-3* | *Single DCM*.

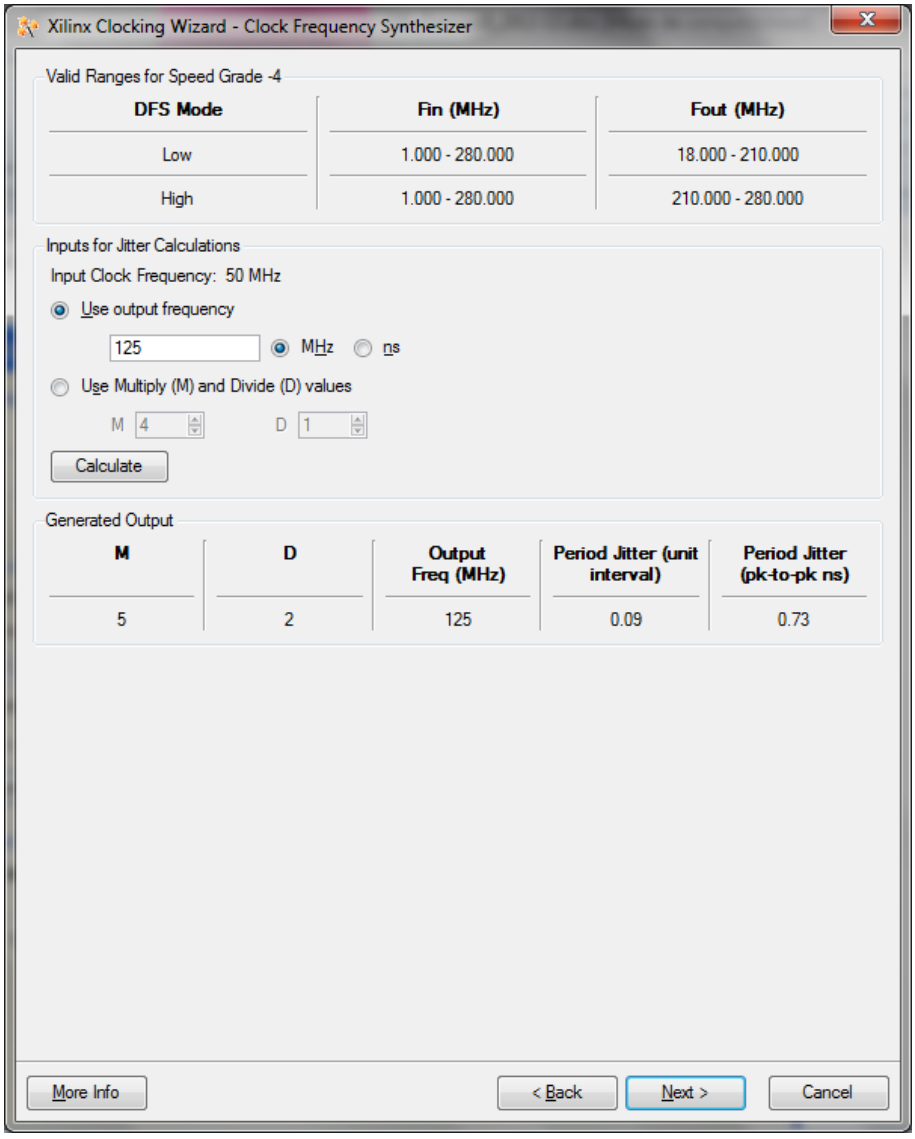


Al darle a *Next* nos aparece un cuadro de diálogo en el que seleccionaremos las opciones por defecto: lenguaje VHDL, sintetizador XST y la FPGA anteriormente seleccionada para el proyecto. Tras esto aparece el *Xilinx Clocking Wizard*. En la primera pantalla seleccionaremos el pin CLKFX, que es la salida del DFS (*Digital Frequency Synthesizer*), e introduciremos la frecuencia de entrada para nuestro sistema, 50 MHz.



Dando a *Next* pasamos a una ventana donde se pueden cambiar los buffers que se conectarán en las salidas del DCM. Dejaremos esta opción por defecto.

Dando otra vez a *Next* entramos en la ventana de configuración del sintetizador de frecuencia. Poniendo 125 MHz en la casilla *Use output frequency* y dando al botón *Calculate*, podemos ver abajo que se nos calculan automáticamente los valores de multiplicación (M) y división (D) del DFS (frecuencia de salida en CLKFX = frecuencia de entrada en CLKIN * M / D). Además se nos reporta el *jitter* estadístico pico a pico que se estima para el reloj generado, tanto en valor absoluto como en fracción del ciclo de reloj.



Continuando nos aparecerá el *Summary* de la configuración y al dar a *Finish* se nos generarán los ficheros que describen la configuración seleccionada. Si miramos en el subdirectorio *ise/ipcore_dir* del proyecto veremos que ha aparecido un fichero "clockgen_dcm.xaw", que describe en formato binario el bloque DCM seleccionado. También podemos ver que se ha generado una descripción VHDL del bloque, el fichero *clockgen_dcm.vhd* que es el mismo que podemos ver en la herramienta seleccionando *clockgen_dcm* en el panel *Design* y ejecutando el proceso *View HDL Source*.

Examinar esta descripción VHDL del bloque generado: podemos ver que en realidad nuestro bloque es un componente DCM propiamente dicho más diversos buffers, tanto en

las salidas como en la entrada. Además en este caso hemos seleccionado (en la primera pantalla por la que pasamos, véase arriba la opción *Feedback Source: Internal*) que se haga una realimentación, de CLK0 a CLKFB, que en general permite alinear la salida del DFS y otras salidas del DCM, como el propio CLK0, con la señal de reloj de entrada.

⇨ Hacer con papel y lápiz o bolígrafo un esquema simple del bloque "clockgen_dcm", pintando los buffers y el conexionado entre ellos, la primitiva DCM y los puertos del bloque.

Desde el panel de procesos también podemos acceder a un "Instantiation Template". Utilizar esta plantilla para conectar el DCM en nuestro diseño, "stopwatch.vhd", interponiéndolo entre el puerto de entrada de reloj, "Clk" y los pines de reloj de todos los bloques. Para ello podemos renombrar la señal "Clk" como "clk2x5" (nueva señal a declarar) en la arquitectura (ojo, sólo en la parte derecha de los "port maps") y luego instanciar y conectar el DCM, haciendo el siguiente "port mapping":

- CLKIN_IN => Clk
- RST_IN => se puede conectar a masa ('0')
- CLKFX_OUT => clk2x5
- CLKIN_IBUFG_OUT, CLK0_OUT, LOCKED_OUT => sin conectar (open)

Comprobar el funcionamiento del DCM

A continuación realizaremos una simulación funcional del circuito. [Nota para aquellos que quieran trabajar en casa: hay que tener en cuenta que como esta vez hemos usado en el VHDL instanciaciones de primitivas de Xilinx (por ejemplo el DCM), será necesario tener compiladas las librerías de Xilinx y configurado ModelSim para encontrarlas, consúltense en la página web del laboratorio las notas a este respecto].

Cambiar el testbench para que se simulen tan sólo 2000 nanosegundos (tras la incialización y pulsado del botón *StartStop*). Lanzando una simulación funcional podremos ver que el reloj "clk2x5" dentro de la instancia del cronómetro (normalmente llamada"uut") es efectivamente 5/2 = 2.5 veces más rápido que "Clk".

Pasaremos ahora a realizar la implementación del diseño. Antes de lanzar este proceso, nos aseguraremos de que estén presentes y sin comentar las líneas del fichero UCF correspondientes a la constraint de reloj:

```
NET "Clk" TNM_NET = CLK_GROUP;
TIMESPEC TS_CLK = PERIOD CLK_GROUP 20 ns;
```

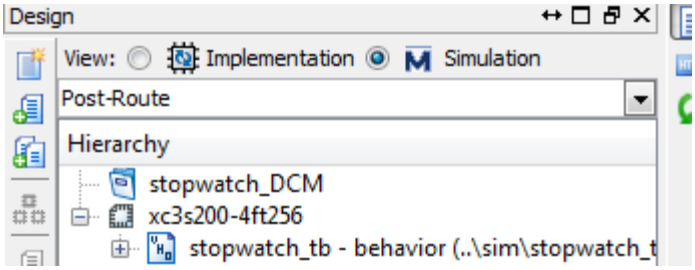
Lanzar la implementación del diseño y al terminar lanzar el análisis de tiempos, asegurándose de que el tipo de reporte sea "verbose" y no sólo de errores (cambiar las propiedades de *Generate Post-Place & Route Static Timing* y lanzar *Analyze Post-Place & Route Static Timing*). Comprobar que ha aparecido automáticamente una *constraint* referida al reloj de salida del DCM (*constraint* derivada de la de su reloj de entrada). En este caso el periodo es de 8 ns y el "slack" podemos ver que es reducido, pero se satisfacen los requerimientos.

Una vez comprobado que no ha habido ningún problema en la implementación, generar el fichero de configuración y descargar el diseño a la placa.

Podremos ver por ejemplo que por cada 25 "segundos" mostrados ahora en el cronómetro transcurren realmente sólo 10 segundos.

Simulación con tiempos

A continuación haremos una simulación con la estimación de tiempos generada por la herramienta tras realizar el *Place and Route*. Para ello en la herramienta seleccionaremos "*Simulation*" en el panel del diseño y a continuación, en la lista desplegable, "*Post-Route*":



Para poder simular necesitaremos comentar en el fichero del testbench la presencia del *generic* "FAST_SIMULATION", porque en la netlist este *generic* no existe. (Otra opción más elegante sería usar una *configuration* VHDL apropiada, pero no haremos esto). Antes de lanzar la simulación comprobar también en el código del testbench que estamos usando una frecuencia de 50 MHz para el reloj que damos a nuestro diseño.

Seleccionar el testbench y en el panel de procesos hacer doble clic en *ModelSim Simulator / Simulate Post-Place & Route Model*.

Obsérvese que los ficheros necesarios para realizar la simulación son depositados en el directorio *ise\netgen\par* : una "netlist" VHDL y un fichero SDF (Standard Delay Format). El fichero SDF contiene las descripciones de los retardos en cada punto del circuito, y es usado por herramientas como los simuladores para "retroanotar" (*back-annotate*) la netlist con los retardos.

Echaremos un vistazo a la simulación y comprobaremos que ahora aparecen retardos en las transiciones. Si intentamos mirar el diseño por dentro la labor se nos complica, pues lo que vemos ahora no es el RTL sino la netlist resultante del proceso de implementación. Por ejemplo, a continuación sacaremos en la ventana de ondas el bit 1 del contador de *freq_divider*. Para ello hacer un "*undock"* de la ventana de ModelSim donde se muestra la jerarquía del diseño, seleccionar en el árbol la instancia del diseño ("uut") y utilizar la herramienta de búsqueda (botón con icono con unos prismáticos) para encontrar los elementos "freq_divider_<contador>_1", donde <contador> es el nombre que le hayamos dado a la cuenta en el bloque "freq_divider". Hacer doble clic sobre la línea que tenga este nombre terminado en el número "1", sin sufijos adicionales, y mandar al visor de ondas la región correspondiente. Simular por ejemplo 2000 ns (siempre avanzando el tiempo hasta más allá del momento en que retiremos el reset en nuestro testbench) y comprobar el retardo entre el pin de reloj (CLK) y el de salida de dato (O) del flip-flop correspondiente.

- ⇨ ¿Cuál es el valor de este retardo? [Para medir retardos en el visor de ondas se puede hacer lo siguiente: a) Añadir un segundo cursor con *Add (| To Wave) | Cursor*; b) Colocar ambos cursores exactamente en los flancos de interés moviéndolos justo tras el flanco mientras está seleccionada en la parte izquierda del visor de ondas la señal correspondiente (observar como el cursor "salta" al flanco); c) En la parte inferior de la ventana de ondas nos aparece la medida de la separación temporal entre ambos cursores].

Abrir en un editor de texto el fichero SDF y buscar la célula con el mismo nombre que hemos usado arriba (por ejemplo, si nuestra señal de cuenta se llama "count" podemos buscar la cadena "freq_divider_count_1)", incluyendo el paréntesis para discriminar esta célula de otras).

- ⇨ Copiar en un fichero de texto aparte la descripción que hay en el SDF de esta célula y tratar de encontrar alguna relación con lo que hemos visto en las ondas de la simulación (aunque no se haya estudiado el formato SDF, se puede intuir el significado de algunas cosas).

Podemos también comprobar que una cosa es el reloj que entra a la FPGA y otra lo que le llega a los flip-flops.

- ⇨ Añadir a las ondas la instancia del DCM usado por el bloque "clockgen_dcm" (aparecerá con nombre <*nombre_instancia_en_stopwatch*>*_DCM_INST*) y medir y anotar el retardo entre los siguientes relojes. Hacer estas medidas en la zona final de la simulación realizada para dar tiempo al DCM para que se estabilice:
  - Del reloj *Clk* de entrada a la FPGA a la pata CLKIN del DCM.
  - Entre las patas CLKIN y CLKFB del DCM.
  - De la pata CLKFX del DCM a la pata de reloj del flip flop examinado anteriormente.
  - Del reloj *Clk* de entrada a la FPGA al reloj usado en el flip-flop.

En este caso tenemos un DCM de por medio, pero siempre habrá una diferencia entre el reloj a la entrada de la FPGA y el que le llega a los flip-flops, debida al retardo del buffer de entrada y de la red de distribución del reloj.

Finalmente, probar a simular *post-layout* el circuito al doble de frecuencia (modificando el periodo en el testbench, a 10 ns para tener 100 MHz de entrada a la FPGA).

- ⇨ ¿Qué ocurre al simular al doble de frecuencia?

Entrega de la práctica

Antes de la fecha límite especificada en la web del laboratorio deberá **enseñarse al profesor** la placa funcionando con el DCM, una simulación *post-layout* y las anotaciones realizadas para las diversas cuestiones planteadas en el enunciado (marcadas con "⇨").

Por otra parte, **deberá entregarse por web un archivo zip con el diseño realizado**, antes de la fecha límite especificada en la web del laboratorio. Entregar el directorio "P5_stopwatch_DCM" completo.

Todos los ficheros del código fuente deberán entregarse con una cabecera adecuada, incluyendo siempre el nombre de los autores.

El archivo zip deberá tener la siguiente nomenclatura:

```
<{DIE_L | DIE_X | DIE_J}><numero_pareja_2digitos>_P5.zip
```

(Ej.: `DIE_J02_P5.zip` para pareja 2 de DIE jueves)

No será necesario presentar ninguna memoria.

Se recuerda que los alumnos deberán comprender el diseño completo y familiarizarse con él.