Cómo generar un fichero .xdc (Actualizado y ampliado)

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2021/01/01/como-generar-un-fichero-xdc-actualizado-y-ampliado/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

En la anterior versión de esta entrada (<- pinchando aquí la tienes) se hablaba de cómo generar un fichero XDC, para ello se necesitaban dos líneas, una que asignaba a cada puerto un banco de alimentación(LVCMOS33, LVCMOS18, etc) y otra en la que se asignaba el pin de la placa. Pues bien, existe otro método más **práctico** de hacer las dos cosas en una sola línea, lo que facilita poder crear fichero .xdc *en los que solo hay que cambiar el nombre de la señal que viene por defecto*.

El sistema es muy sencillo, en vez de escribir:

```
set_property IOSTANDAR <Banco de alimentación> [get_ports <nombre puerto>];
set_property PACKAGE_PIN <nombre del pin> [get_ports <nombre del puerto>];
```

Se tiene que escribir:

```
set_property -dict { PACKAGE_PIN <nombre del pin> IOSTANDARD <Banco de
alimentación> } [get_ports <nombre del puerto>];
```

Ejemplo

Para entender mejor cómo hacer este nuevo .xdc se va a optimizar el ejemplo de la otra vez.

En el ejemplo anterior se tenían dos puertos, dos_bits[0] y dos_bits[1], y para ello se usaban las siguientes cuatro líneas.

```
set_property IOSTANDAR LVCMOS33 [get_ports {dos_bits[0]}];  #pin 7
set_property IOSTANDAR LVCMOS33 [get_ports {dos_bits[1]}]; #pin 8

set_property PACKAGE_PIN K18 [get_ports {dos_bits[0]}];  #pin 7
set_property PACKAGE_PIN K17 [get_ports {dos_bits[1]}];  #pin 8
```

Pues con el nuevo formato de XDC quedaría de la siguiente manera:

```
set_property -dict { PACKAGE_PIN K18 IOSTANDARD LVCMOS33 } [get_ports
{dos_bits[0]}];
set_property -dict { PACKAGE_PIN K19 IOSTANDARD LVCMOS33 } [get_ports
{dos_bits[1]}];
```

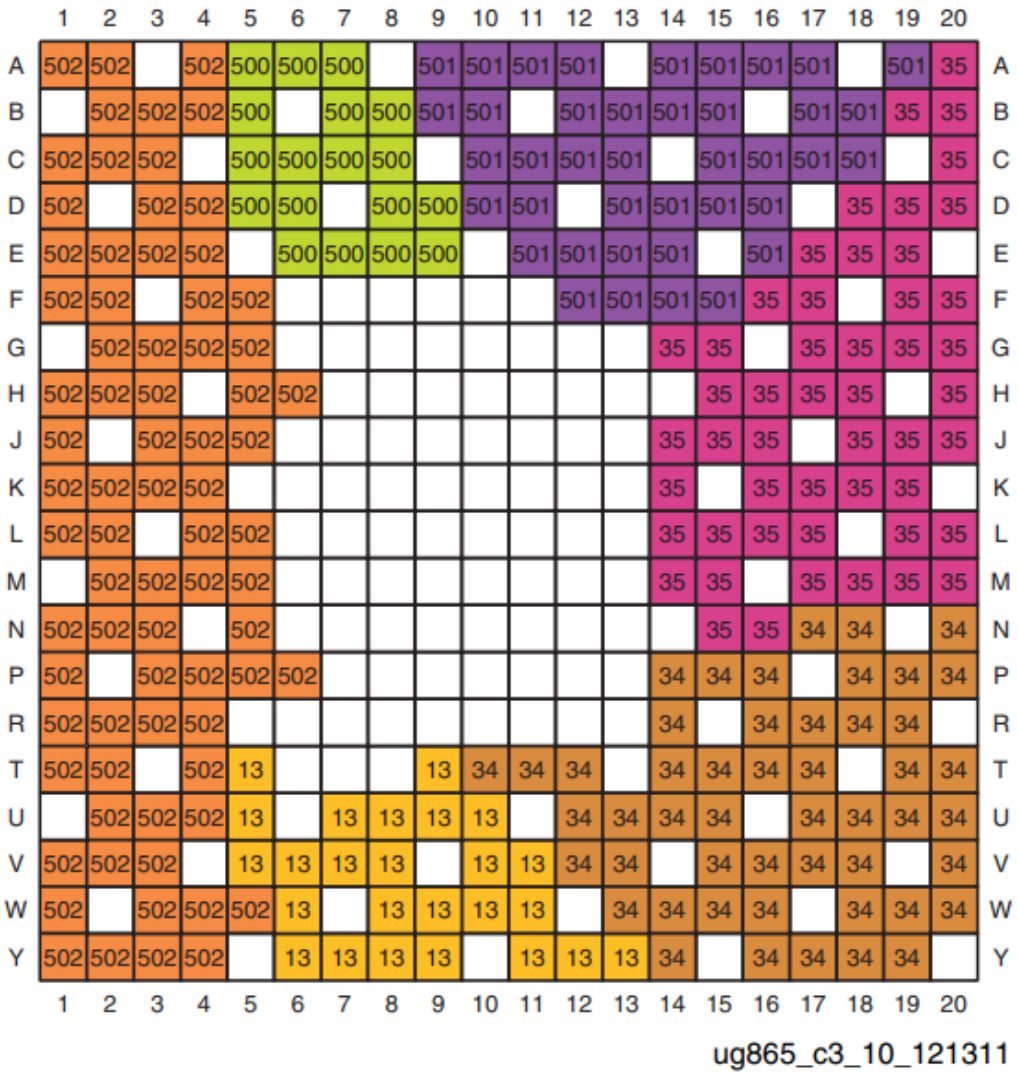
Con este formato se ahorra una línea por puerto, por lo que cualquier modificación de pines es más sencilla, y además permite generar ficheros XDC genéricos que permitan tener un único XDC para evitar tener que ir continuamente al datasheet para saber que pines tienes que tocar.

Aquí dejo los dos formatos de XDC para una placa QMTECH para que se vea más claramente las diferencias.

- Antigua versión
- Nueva versión (esta versión es extremadamente genérica, por lo que es más sencillo cambiar los pines)

Otra versión de hacer un XDC. ***Poner todo el banco de alimentación de pines a la misma tensión***

Otra forma de hacer los XDC es poner todo el banco de pines al mismo banco de alimentación. Para entender esto hay que saber que los pines del chip están agrupados en bancos, por ejemplo, en una Zynq-7000 tiene todos los pines de la memoria ram en un mismo banco y los pines de salida los tiene divididos en dos bancos distintos. *[Adjunto imagen de un XC7Z010-CLG400]*



En la imagen se pueden ver todas las agrupaciones de pines **Nota: los pines que faltan son pines no conectados(NC), alimentaciones, masas, etc **

Bien, pues estas agrupaciones se pueden poner a la misma tensión para ello se recurre a lo explicado en la versión anterior, pero en vez de poner cada pin a una tensión se pone todo el banco a la misma. Todo sigue el siguiente esquema

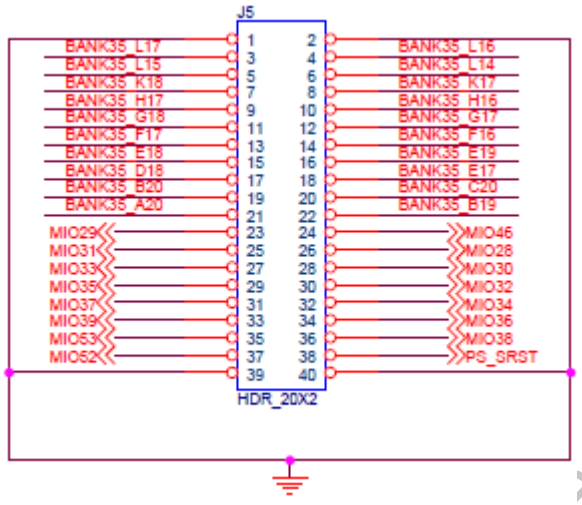
```
set_property PACKAGE_PIN <nombre del pin> [get_ports <nombre del puerto>];
```

```
set_property IOSTANDAR <Banco de alimentación> [get_ports -of_objects
[get_iobanks <número de banco de pines>]];
```

La primera parte es la misma forma de poner a un puerto un pin de salida. La segunda parte es la que pone todo el banco al mismo banco de alimentación.

Ejemplo

El ejemplo es el mismo que se ha usado anteriormente, pero rescatando la imagen del datasheet, ¿por qué? porque es ahí dónde figura el banco de pines a los que pertenecen los pines de salida.



Cómo se ve en la imagen los pines K17 y K18 pertenecen al banco 35, por lo que el .xdc quedaría de la siguiente manera.

```
set_property PACKAGE_PIN K18 [get_ports {dos_bits[0]}];  #pin 7
set_property PACKAGE_PIN K17 [get_ports {dos_bits[1]}];  #pin 8

set_property IOSTANDAR LVCMOS33 [get_ports -of_objects [get_iobanks 35]
]; #todos los pines del banco a 3.3V
```

Cómo generar un reloj en un xdc para una FPGA

Para generar un reloj se necesitan varias cosas, la primera es saber a que pin está unido el oscilador de la placa, por ejemplo, este pin en una Nexys 4 DDR es un pin de tipo MRCC con el nombre E9. *[Datasheet de la Nexys 4 DDR]*. Otra cosa que hay que saber es el periodo del reloj, el periodo es la inversa de la frecuencia ($1/f_{clk}$). Con todo esto ya se puede generar el reloj en el XDC.

El reloj en el XDC tiene la siguiente estructura:

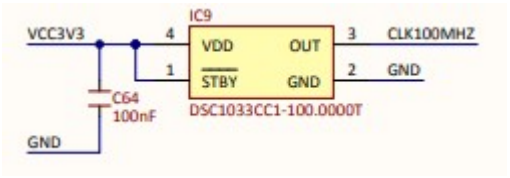
```
set_property -dict { PACKAGE_PIN <pin del reloj> IOSTANDARD <banco de
alimentación> } [get_ports <puerto de entrada del reloj>];

create_clock -period <periodo del reloj en ns> [get_ports <puerto de entrada del
reloj>];
```

La primera parte es la declaración del pin de entrada del reloj. (Aquí se usa para declarar la versión explicada en la parte superior de esta entrada). En la segunda parte es cuando se crea el reloj. En ella se configura el periodo del reloj (por defecto tiene un ciclo del 50%) y el nombre del puerto del reloj.

Ejemplo

En este ejemplo se va a crear el reloj para una Nexys 4 DDR de 10 ns de periodo, con el puerto de entrada del reloj llamado ***«CLK»***

Para empezar primero hay que localizar a que pin va el oscilador de la placa



Como se puede comprobar va al pin E3.

Una vez se sabe el nombre del pin, el del puerto y el periodo se crea el .xdc. Como banco de alimentación se toma el LVCMOS33.

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports CLK];

create_clock -period 10.00 [get_ports CLK];
```

Con esto ya se podría generar el reloj de entrada.

Otras configuraciones

Para crear un reloj también se puede configurar el ciclo de trabajo de la señal de reloj y darle un nombre. Por ejemplo, si en el ejemplo anterior se quisiese un reloj de un ciclo de trabajo del 70% y un nombre como «senal_reloj» solo habría que añadir los siguientes parámetros:

```
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports CLK];

create_clock -add -name senal_reloj -period 10.00 -waveform {0 7}[get_ports
CLK];
```

El ciclo de trabajo se puede configurar de otras formas: {1 8}, {2 9}, etc. Recordando siempre que el ciclo de trabajo elegible es en *ns* **(NO en porcentaje)**, o lo que es lo mismo en el ejemplo «{1 8}» es un ‘0’ hasta 1ns, ‘1’ desde 1ns hasta 8ns y después ‘0’ hasta los 10ns.

Para más información sobre los XDC está la documentación oficial de Xilinx.