

Cómo introducir tu propio modelo de PCB en Vivado

Creador: David Rubio G.

Entrada: <https://soceame.wordpress.com/2025/01/02/como-introducir-tu-propio-modelo-de-pcb-en-vivado/>

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

GitHub: <https://github.com/DRubioG>

Fecha última modificación: 24/02/2025

En esta entrada voy a explicar cómo puedes generar los ficheros necesarios para preconfigurar Vivado con tu propio modelo de PCB. La configuración es tremendamente extensa, por ello se va a hacer una descripción básica que te permita entender cómo lo puedes hacer con un pequeño ejemplo en GitHub para que lo puedas utilizar como referencia.

El crear tu propia PCB en Vivado facilita la vida a la hora de configurar los bloques, porque por ejemplo no habría que saberse los nombres de los chips o si trabajas con una Zynq no tendrías que generar un XDC para muchos pines, como podría ser el pin de reloj.

Aunque es bastante complejo para PCBs avanzadas y se basa en prueba y error, hasta que consigues que funcione, por eso el uso de un ejemplo completo como el que doy al final, para que puedas ir tocando.

Introducción

Principalmente para introducir tu propio modelo de PCB necesitas principalmente 4 ficheros.

- **board.xml**: este fichero contiene la información básica de la PCB, como el nombre, quién es el fabricante, bloques IP básicos, etc
- **part0_pins.xml**: en este fichero se pueden dejar configurados los pines de I/O de los bloques IP definidos en el punto anterior.
- **preset.xml**: este fichero es el fichero de preconfiguración de los bloques IP.
- **<imagen>.jpg**: esta es la imagen que se mostrará a la hora de seleccionar la placa.

NOTA: los ficheros *part0_pins.xml* y *preset.xml* tienen que ser previamente definidos en el fichero *board.xml*.

NOTA 2: es importante que todo esté definido, no vale dejar a medias la descripción de la PCB porque si no Vivado no la reconocerá.

Para que nos entendamos pongo un pequeño ejemplo, imaginemos que configuramos un bloque IP con la UART, bien, en el fichero *board.xml* definiré que tengo un bloque IP con una UART de dos pines (con índices 2 y 3, esto se explicará más adelante). En el fichero *part0_pins.xml* se definen los pines G21 y G22 como los pines de la UART. Y en el fichero *preset.xml* se define la configuración de 115200 baudios del bloque IP.

board.xml

Este es el fichero principal para las placas de Vivado. Este fichero tiene que contener toda la información sobre las placas o sobre los ficheros que contienen la información sobre las placas.

Este fichero se tiene que estructurar de la siguiente manera.

1. Primero viene la versión del fichero de configuración [**schema_version**] (versión 2.0, en Vivado queda reflejado el número de versión de más adelante), después el creador de la placa [**vendor**] (SoCeame, este nombre Vivado lo pone en minúscula), después el nombre de la placa [**name**] (este

nombre no aparece en ningún sitio), después el nombre que aparecerá en Vivado [**display_name**], luego la URL de la placa [**url**] (este campo es opcional, si la placa no está en internet mejor no ponerlo. También es importante, este enlace se ejecuta al darle al nombre de la placa, lo cuál puede ser un coñazo que se abra el navegador al darle al nombre cada vez que crees un proyecto) y por último el fichero de configuración previa (en caso de que lo haya) [**preset_file**] (este fichero se tiene que llamar siempre igual, preset.xml).

1.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<board schema_version="2.0" vendor="SoCeame" name="Zynq-7000" display_name="Zynq7000" url="github.com/SoCeame/Zynq7000" preset_file="preset.xml">
```

2. El siguiente punto es la imagen, en este campo se define la imagen de tu placa, también hay que darle un nombre y una descripción. El punto de *file_version* que será la versión que mostrará Vivado.

```
<images>
  <image name="imagen.jpg" display_name="Zynq" sub_type="board">
    <description>Zynq-7000 Board File Image</description>
  </image>
</images>
<compatible_board_revisions>
  <revision id="0">d</revision>
</compatible_board_revisions>
<file_version>1.0</file_version>
<description>Version</description>
```

3. El siguiente paso que viene es la definición de interfaces y definiciones. Todo esto tiene que estar dentro de una etiqueta component.

- El component tiene que comenzar por una etiqueta component donde figure principalmente el fichero *part0_pins* para los pines de la interfaz y el nombre del chip que se va a utilizar (*xc7z010clg400-1*).
- Después vienen las interfaces específicas. Para explicarlo mejor se va a utilizar como ejemplo un bloque AXI GPIO con 8 leds.
 - Para definir los leds, primer definimos el tipo de interfaz que se va a utilizar (master) después el nombre de la interfaz (leds_8bits, este nombre es el que figurará en el puerto del bloque GPIO), después se crea la interfaz para un bloque AXI GPIO, luego el *of_component* que se puede dejar con el mismo que el nombre, y por último el nombre que va a tener el preset que configura el bloque IP al introducirlo.
 - En la etiqueta *preferred_ip* se configura el nombre que va a tener el bloque IP que se meta.
 - Luego se define el *port_map* donde se le pone nombre a la interfaz y se define el tipo. También se define el tipo de pines y el tamaño de la interfaz (de izquierda a derecha).

- Lo siguiente es definir el *pin_map* donde se le pone un índice, relativo a la interfaz, y un nombre a la interfaz.

```
<component name="part0" display_name="Zynq7000" type="fpga" part_name="xc7z010clg400-1" pin_map_file="part0_pins.xml" vendor="SoC-eame" spec_url="github.com/DRubioG">
  <description>FPGA part on the board</description>
  <interfaces>
    <interface mode="master" name="leds_8bits" type="xilinx.com:interface:gpio_rtl:1.0" of_component="leds_8bits" preset_proc="leds_8bits_preset">
      <port_maps>
        <port_map logical_port="TRI_O" physical_port="leds_8bits_tri_o" dir="out" left="7" right="0">
          <pin_maps>
            <pin_map port_index="0" component_pin="leds_8bits_tri_o_0"/>
            <pin_map port_index="1" component_pin="leds_8bits_tri_o_1"/>
            <pin_map port_index="2" component_pin="leds_8bits_tri_o_2"/>
            <pin_map port_index="3" component_pin="leds_8bits_tri_o_3"/>
            <pin_map port_index="4" component_pin="leds_8bits_tri_o_4"/>
            <pin_map port_index="5" component_pin="leds_8bits_tri_o_5"/>
            <pin_map port_index="6" component_pin="leds_8bits_tri_o_6"/>
            <pin_map port_index="7" component_pin="leds_8bits_tri_o_7"/>
          </pin_maps>
        </port_map>
      </port_maps>
    </interface>

    <interface mode="master" name="sws_5bits" type="xilinx.com:interface:gpio_rtl:1.0" of_component="sws_5bits" preset_proc="sws_5bits_preset">
      <port_maps>
        <port_map logical_port="TRI_I" physical_port="sws_5bits_tri_i" dir="in" left="4" right="0">
          <pin_maps>
            <pin_map port_index="0" component_pin="sws_5bits_tri_i_0"/>
            <pin_map port_index="1" component_pin="sws_5bits_tri_i_1"/>
            <pin_map port_index="2" component_pin="sws_5bits_tri_i_2"/>
            <pin_map port_index="3" component_pin="sws_5bits_tri_i_3"/>
            <pin_map port_index="4" component_pin="sws_5bits_tri_i_4"/>
          </pin_maps>
        </port_map>
      </port_maps>
    </interface>

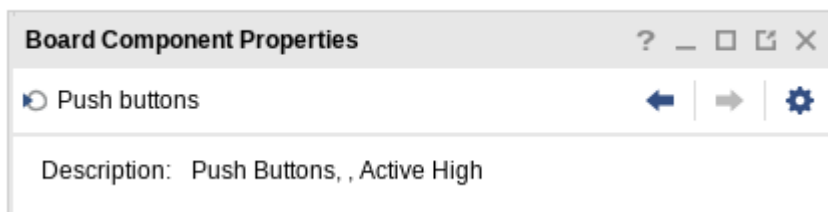
    <interface mode="master" name="btns_3bits" type="xilinx.com:interface:gpio_rtl:1.0" of_component="btns_3bits" preset_proc="btns_3bits_preset">
      <port_maps>
        <port_map logical_port="TRI_I" physical_port="btns_3bits_tri_i" dir="in" left="2" right="0">
          <pin_maps>
            <pin_map port_index="0" component_pin="btns_3bits_tri_i_0"/>
            <pin_map port_index="1" component_pin="btns_3bits_tri_i_1"/>
            <pin_map port_index="2" component_pin="btns_3bits_tri_i_2"/>
          </pin_maps>
        </port_map>
      </port_maps>
    </interface>

    <interface mode="slave" name="sys_clock" type="xilinx.com:signal:clock_rtl:1.0" of_component="sys_clock" preset_proc="sys_clock_preset">
      <port_maps>
        <port_map logical_port="CLK" physical_port="sys_clk" dir="in">
          <pin_maps>
            <pin_map port_index="0" component_pin="sys_clk"/>
          </pin_maps>
        </port_map>
      </port_maps>

      <parameters>
        <parameter name="frequency" value="125000000" />
      </parameters>
    </interface>
  </interfaces>
</component>
```

- Lo siguiente que se define son las definiciones que aparecerán en el campo *Board Component Properties*.

```
<component name="btns_3bits" display_name="Push buttons" type="chip" sub_type="push_button" major_group="General Purpose Input or Output">
  <description>Push Buttons, , Active High</description>
</component>
```



Modelo completo:

```
<components>

<component name="part0" display_name="Zynq7000" type="fpga" part_name="xc7ze10clg400-1" pin_map_file="part0_pins.xml" vendor="SOC-eame" spec_url="github.com/DRubio06">

  <description>FPGA part on the board</description>
  <interfaces>
    <interface mode="master" name="leds_8bits" type="xilinx.com:interface:gpio_rtl:1.0" of_component="leds_8bits" preset_proc="leds_8bits_preset">
      <port_maps>
        <port_map logical_port="TRI_O" physical_port="leds_8bits_tri_o" dir="out" left="7" right="0">
          <pin_maps>
            <pin_map port_index="0" component_pin="leds_8bits_tri_o_0"/>
            <pin_map port_index="1" component_pin="leds_8bits_tri_o_1"/>
            <pin_map port_index="2" component_pin="leds_8bits_tri_o_2"/>
            <pin_map port_index="3" component_pin="leds_8bits_tri_o_3"/>
            <pin_map port_index="4" component_pin="leds_8bits_tri_o_4"/>
            <pin_map port_index="5" component_pin="leds_8bits_tri_o_5"/>
            <pin_map port_index="6" component_pin="leds_8bits_tri_o_6"/>
            <pin_map port_index="7" component_pin="leds_8bits_tri_o_7"/>
          </pin_maps>
        </port_map>
      </port_maps>
    </interface>

    <interface mode="master" name="sws_5bits" type="xilinx.com:interface:gpio_rtl:1.0" of_component="sws_5bits" preset_proc="sws_5bits_preset">
      <port_maps>
        <port_map logical_port="TRI_I" physical_port="sws_5bits_tri_i" dir="in" left="4" right="0">
          <pin_maps>
            <pin_map port_index="0" component_pin="sws_5bits_tri_i_0"/>
            <pin_map port_index="1" component_pin="sws_5bits_tri_i_1"/>
            <pin_map port_index="2" component_pin="sws_5bits_tri_i_2"/>
            <pin_map port_index="3" component_pin="sws_5bits_tri_i_3"/>
            <pin_map port_index="4" component_pin="sws_5bits_tri_i_4"/>
          </pin_maps>
        </port_map>
      </port_maps>
    </interface>

    <interface mode="master" name="btms_3bits" type="xilinx.com:interface:gpio_rtl:1.0" of_component="btms_3bits" preset_proc="btms_3bits_preset">
      <port_maps>
        <port_map logical_port="TRI_I" physical_port="btms_3bits_tri_i" dir="in" left="2" right="0">
          <pin_maps>
            <pin_map port_index="0" component_pin="btms_3bits_tri_i_0"/>
            <pin_map port_index="1" component_pin="btms_3bits_tri_i_1"/>
            <pin_map port_index="2" component_pin="btms_3bits_tri_i_2"/>
          </pin_maps>
        </port_map>
      </port_maps>
    </interface>

    <interface mode="slave" name="sys_clock" type="xilinx.com:signal:clock_rtl:1.0" of_component="sys_clock" preset_proc="sys_clock_preset">
      <port_maps>
        <port_map logical_port="CLK" physical_port="sys_clk" dir="in">
          <pin_maps>
            <pin_map port_index="0" component_pin="sys_clk"/>
          </pin_maps>
        </port_map>
      </port_maps>

      <parameters>
        <parameter name="frequency" value="125000000" />
      </parameters>
    </interface>
  </interfaces>
</component>

<component name="sys_clock" display_name="System clock" type="chip" sub_type="system_clock" major_group="Clock Sources">
  <description>System Clock, 50 MHz</description>
</component>

<component name="btms_3bits" display_name="Push buttons" type="chip" sub_type="push_button" major_group="General Purpose Input or Output">
  <description>Push Buttons, , Active High</description>
</component>

<component name="leds_8bits" display_name="leds" type="chip" sub_type="led" major_group="General Purpose Input or Output">
  <description>Leds, , Active High</description>
</component>

<component name="sws_5bits" display_name="Push buttons" type="chip" sub_type="switch" major_group="General Purpose Input or Output">
  <description>Switches, , Active High</description>
</component>
</components>
```

4. El siguiente paso es la definición del JTAG, que no haría falta tocarla.

```
<jtag_chains>
  <jtag_chain name="chain1">
    <position name="0" component="part0"/>
  </jtag_chain>
</jtag_chains>
```


5. La última parte va referenciada a los pines del fichero `part0_pins.xml`. En esta parte se ordenan los interfaces en un orden absoluto de todas las interfaces. Esto se entenderá mejor en la definición de `part0_pins.xml`.

- Por ejemplo, los leds van definidos de la posición 3 a la posición 10.

```
<connections>

<connection name="part0_btms_3bits" component1="part0" component2="btms_3bits">
| <connection_map name="part0_btms_3bits_1" c1_st_index="0" c1_end_index="2" c2_st_index="0" c2_end_index="2"/>
</connection>

<connection name="part0_leds_8bits" component1="part0" component2="leds_8bits">
| <connection_map name="part0_leds_8bits_1" c1_st_index="3" c1_end_index="10" c2_st_index="0" c2_end_index="7"/>
</connection>


<connection name="part0_sws_5bits" component1="part0" component2="sws_5bits">
| <connection_map name="part0_sws_5bits_1" c1_st_index="11" c1_end_index="15" c2_st_index="0" c2_end_index="4"/>
</connection>

<connection name="part0_sys_clock" component1="part0" component2="sys_clock">
| <connection_map name="part0_sys_clock_1" c1_st_index="16" c1_end_index="16" c2_st_index="0" c2_end_index="0"/>
</connection>

</connections>
```

Ejemplo: https://github.com/DRubioG/Ejemplo_fichero_PCB_Vivado/blob/main/Zynq-7000/board.xml

Toda la configuración anterior dejaría una placa de la siguiente manera.

Display Name	Preview	Vendor	File Version	Part	I/O Pin Count	Board Rev	Available IC
Zynq7000		soceame	1.0	xc7z010clg400-1	400	d	100

part0_pins.xml

Este fichero define los pines de entrada y salida de la interfaz creada en el `board.xml`, con la tensión del pin.

En este fichero lo primero que se tiene que definir es el `part_info`, que es el nombre del chip al que se le van a asignar los pines.

Después se tiene que crear las etiquetas `pins`, estás etiquetas están referencias por un índice, es es el índice que se ha definido en el `board.xml`, después va el nombre de la interfaz seguido del número del pin en el bloque IP, después la tensión del pin y por último, el pin al que se asigna en la FPGA/SoC.

- Por ejemplo los leds que iban en posición absoluta del 3 al 10, aquí quedan definidos.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<part_info part_name="xc7z010clg400-1">
<pins>
  <pin index="0" name ="btns_3bits_tri_i_0" iostandard="LVCMOS33" loc="K16"/>
  <pin index="1" name ="btns_3bits_tri_i_1" iostandard="LVCMOS33" loc="J16"/>
  <pin index="2" name ="btns_3bits_tri_i_2" iostandard="LVCMOS33" loc="J15"/>

  <pin index="3" name ="leds_8bits_tri_o_0" iostandard="LVCMOS33" loc="H20"/>
  <pin index="4" name ="leds_8bits_tri_o_1" iostandard="LVCMOS33" loc="G19"/>
  <pin index="5" name ="leds_8bits_tri_o_2" iostandard="LVCMOS33" loc="G20"/>
  <pin index="6" name ="leds_8bits_tri_o_3" iostandard="LVCMOS33" loc="H15"/>
  <pin index="7" name ="leds_8bits_tri_o_4" iostandard="LVCMOS33" loc="G15"/>
  <pin index="8" name ="leds_8bits_tri_o_5" iostandard="LVCMOS33" loc="K14"/>
  <pin index="9" name ="leds_8bits_tri_o_6" iostandard="LVCMOS33" loc="J14"/>
  <pin index="10" name ="leds_8bits_tri_o_7" iostandard="LVCMOS33" loc="N15"/>

  <pin index="11" name ="sws_5bits_tri_i_0" iostandard="LVCMOS33" loc="N16"/>
  <pin index="12" name ="sws_5bits_tri_i_1" iostandard="LVCMOS33" loc="L14"/>
  <pin index="13" name ="sws_5bits_tri_i_2" iostandard="LVCMOS33" loc="L15"/>
  <pin index="14" name ="sws_5bits_tri_i_3" iostandard="LVCMOS33" loc="M14"/>
  <pin index="15" name ="sws_5bits_tri_i_4" iostandard="LVCMOS33" loc="M15"/>

  <pin index="16" name ="sys_clk" iostandard="LVCMOS33" loc="H16"/>
</pins>
</part_info>
```

Ejemplo: https://github.com/DRubioG/Ejemplo_fichero_PCB_Vivado/blob/main/Zynq-7000/part0_pins.xml

preset.xml

Este es el último fichero que se necesita para configurar la PCB, también es el más complejo porque hay que saber que parámetros hay que tocar. Por ello se recomienda guiarse por los otros ficheros de PCBs que incluye Vivado para saber sobre qué parámetros hay que configurar.

NOTA: este fichero también incluye la opción de lanzar TCLs que contengan la información de la configuración del bloque IP.

Por ejemplo para configurar el reloj de 125MHz se tiene que hacer de la siguiente forma.

```
<ip_preset preset_proc_name="sys_clock_preset">
  <ip vendor="xilinx.com" library="ip" name="clk_wiz" ip_interface="clk_in1">
    <user_parameters>
      <user_parameter name="CONFIG.PRIM_IN_FREQ" value="125"/>
      <user_parameter name="CONFIG.PRIM_SOURCE" value="Single_ended_clock_capable_pin"/>
      <user_parameter name="CONFIG.RESET_TYPE" value="ACTIVE_LOW"/>
    </user_parameters>
  </ip>
  <ip vendor="xilinx.com" library="ip" name="clk_wiz" ip_interface="clk_in2">
    <user_parameters>
      <user_parameter name="CONFIG.USE_INCLK_SWITCHOVER" value="true"/>
      <user_parameter name="CONFIG.SECONDARY_IN_FREQ" value="125"/>
      <user_parameter name="CONFIG.SECONDARY_SOURCE" value="Single_ended_clock_capable_pin"/>
      <user_parameter name="CONFIG.RESET_TYPE" value="ACTIVE_LOW"/>
    </user_parameters>
  </ip>
</ip_preset>
```

NOTA: Los bloques de la Zynq, se llaman *processing_system* y son extremadamente complejos de configurar de primeras por eso se recomienda buscar ayuda en los ficheros que incluye Vivado.

Ejemplo: https://github.com/DRubioG/Ejemplo_fichero_PCB_Vivado/blob/main/Zynq-7000/preset.xml

Imagen

La imagen puede ser un JPG o un PNG, sin importar el tamaño. Aunque Vivado reescalará la imagen, entonces, se recomienda que mantenga unas dimensiones próximas a un rectángulo.

Pasos finales

Una vez tengamos los ficheros creados, solo necesitamos seguir una estructura de ficheros como la siguiente:

```
- <nombre_placa> (todo junto)
|-<versión de la placa> (algo como 1.0)
||-board.xml
||-preset.xml
||-part0_pins.xml
||-<imagen>
```

NOTA: a veces también se puede encontrar que la carpeta de versión está dentro de otra carpeta, eso es más para placa comerciales, aunque con el formato que se ha explicado anteriormente también es válido y más simple.

Ahora esta carpeta que has creado la tienes que llevar al directorio donde están las placas de Vivado. En esta entrada tienes la ruta donde se guardaría en Linux.

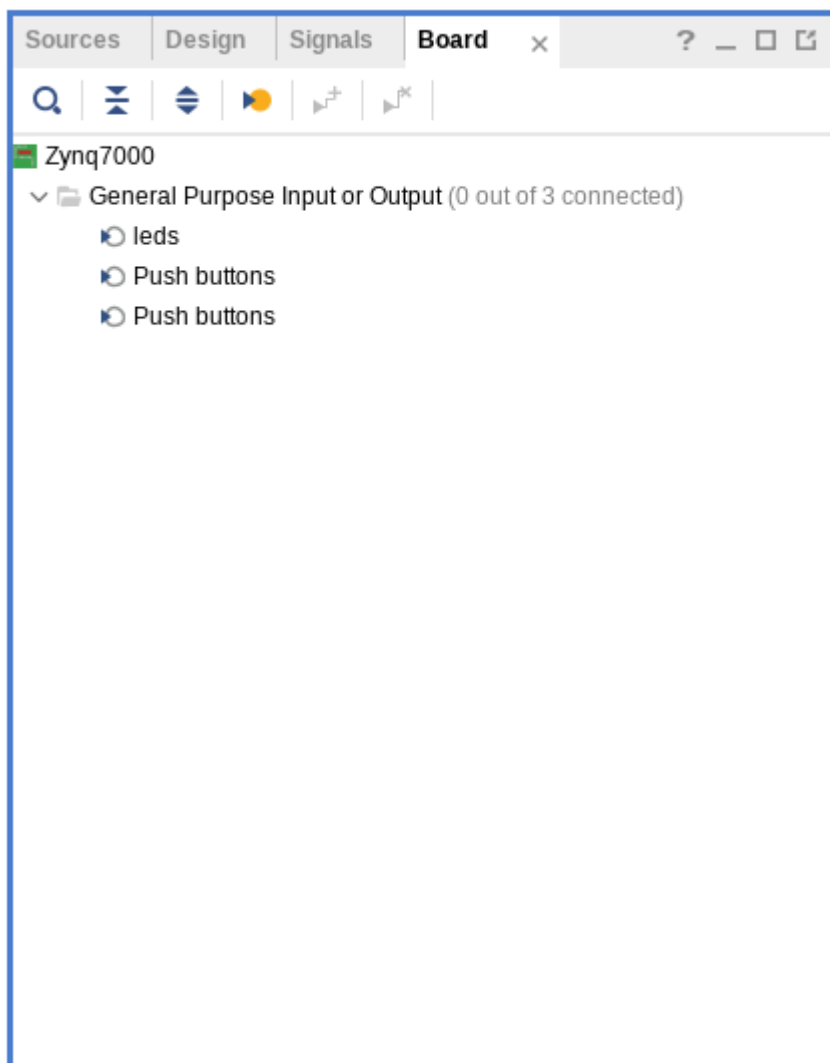
<https://soceame.wordpress.com/2024/06/10/como-introducir-los-ficheros-de-nuevas-placas-en-vivado-para-linux/>

En Windows: <Vivado>\<Vivado_version>\data\boards\board_files

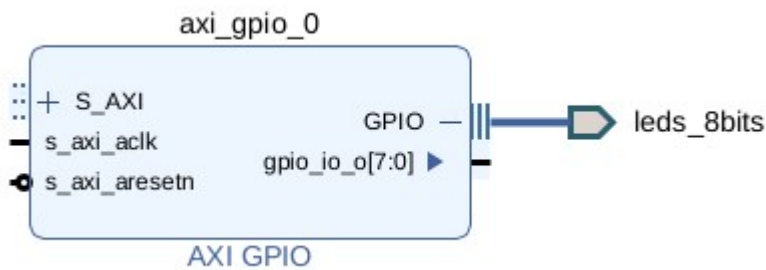
Ahora cuando abras Vivado podrás acceder a la nueva configuración de la placa que has creado.

Display Name	Preview
Zynq7000	

Además si creas un *Block Design* aparecerá un apartado llamado *Board* que incluirá los bloques que has preconfigurado.



Y al incluir los bloques aparecerá ya con la interfaz predefinida en el *board.xml*, lo que hará que no sea necesario incluir estos pines en el XDC porque estos pines ya han sido predefinidos en el *part0_pins.xml*.



Si la placa que has diseñado es comercial, puedes subir estos ficheros a este repositorio de Xilinx en GitHub para que se incluya directamente en cualquier versión de Vivado:

<https://github.com/Xilinx/XilinxBoardStore>

Ejemplo

En este repositorio hay dos ejemplos, el del blog y el de la Pynq-Z1 que puede llegar a ser de mucha ayuda para ficheros de PCBs semicomplejos.

https://github.com/DRubioG/Ejemplo_fichero_PCB_Vivado