

Crear un bloque IP propio

Creador: David Rubio G.

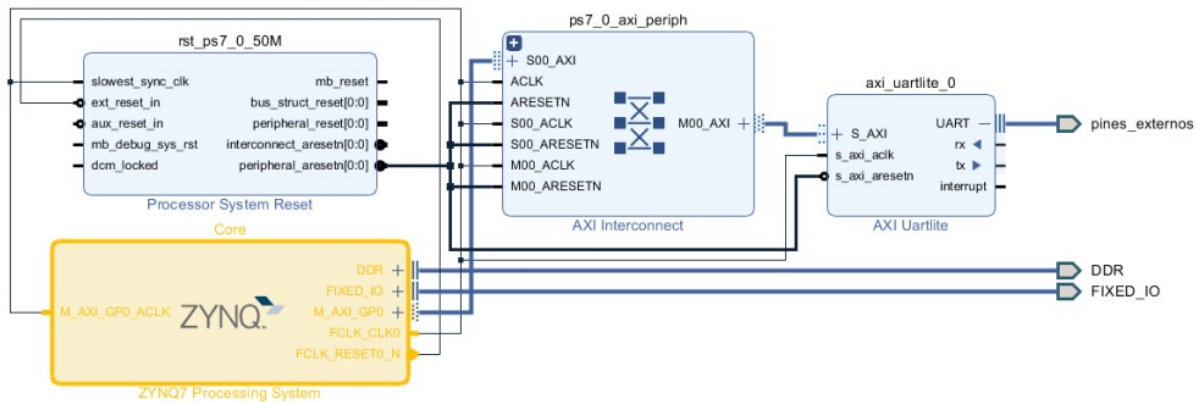
Entrada: <https://soceame.wordpress.com/2024/06/12/crear-un-bloque-ip-propio/>

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

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

Fecha última modificación: 22/02/2025

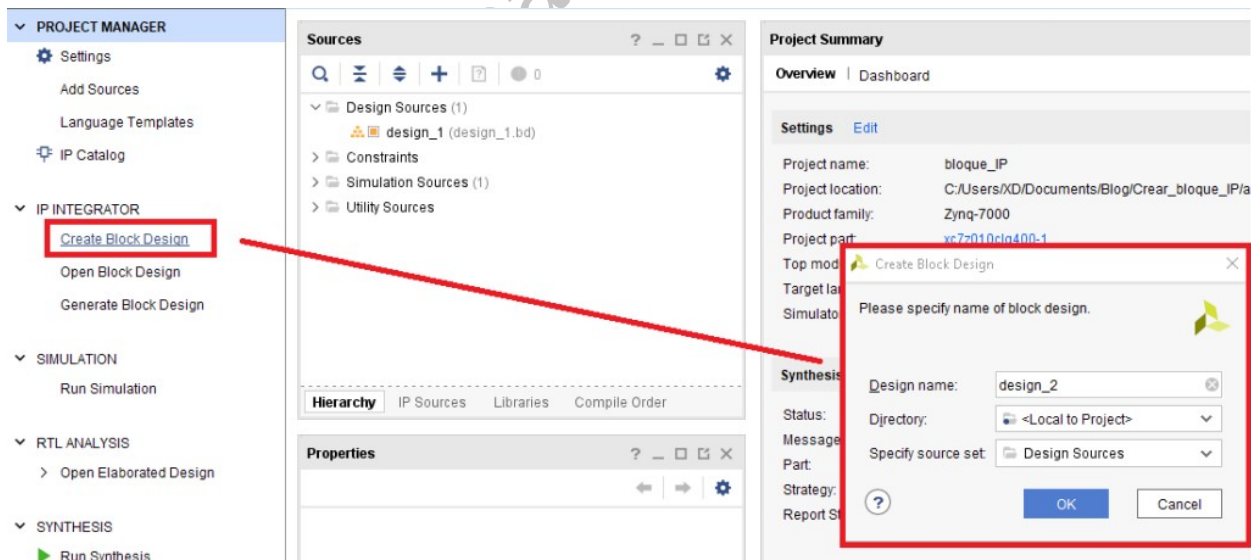
Un bloque IP es una forma que tiene Xilinx de encapsular código HDL para luego ejecutar sistemas interconectados, como por ejemplo conectar un Core(microprocesador) a un sistema que se comunica por UART usando los pines externos.



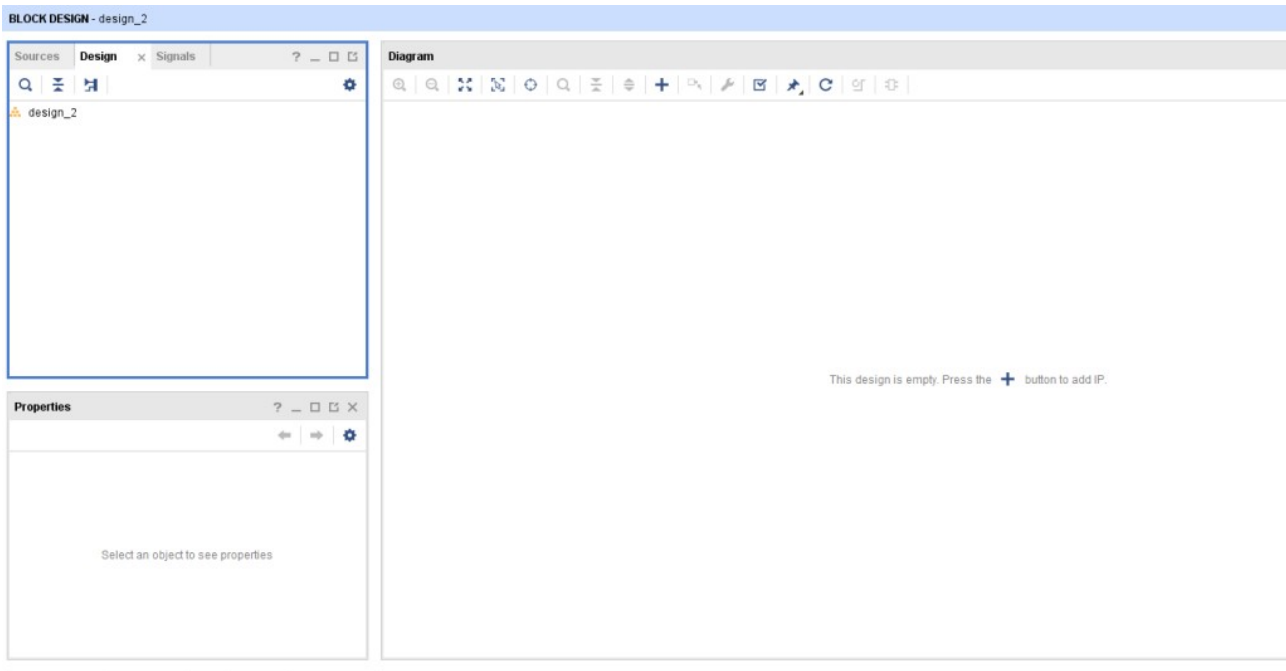
Gracias a esta forma de encapsular se pueden crear sistemas complejos usando una interfaz gráfica, además de hacer que todo el sistema tenga el mismo reloj y el mismo reset.

Cómo crear un bloque IP

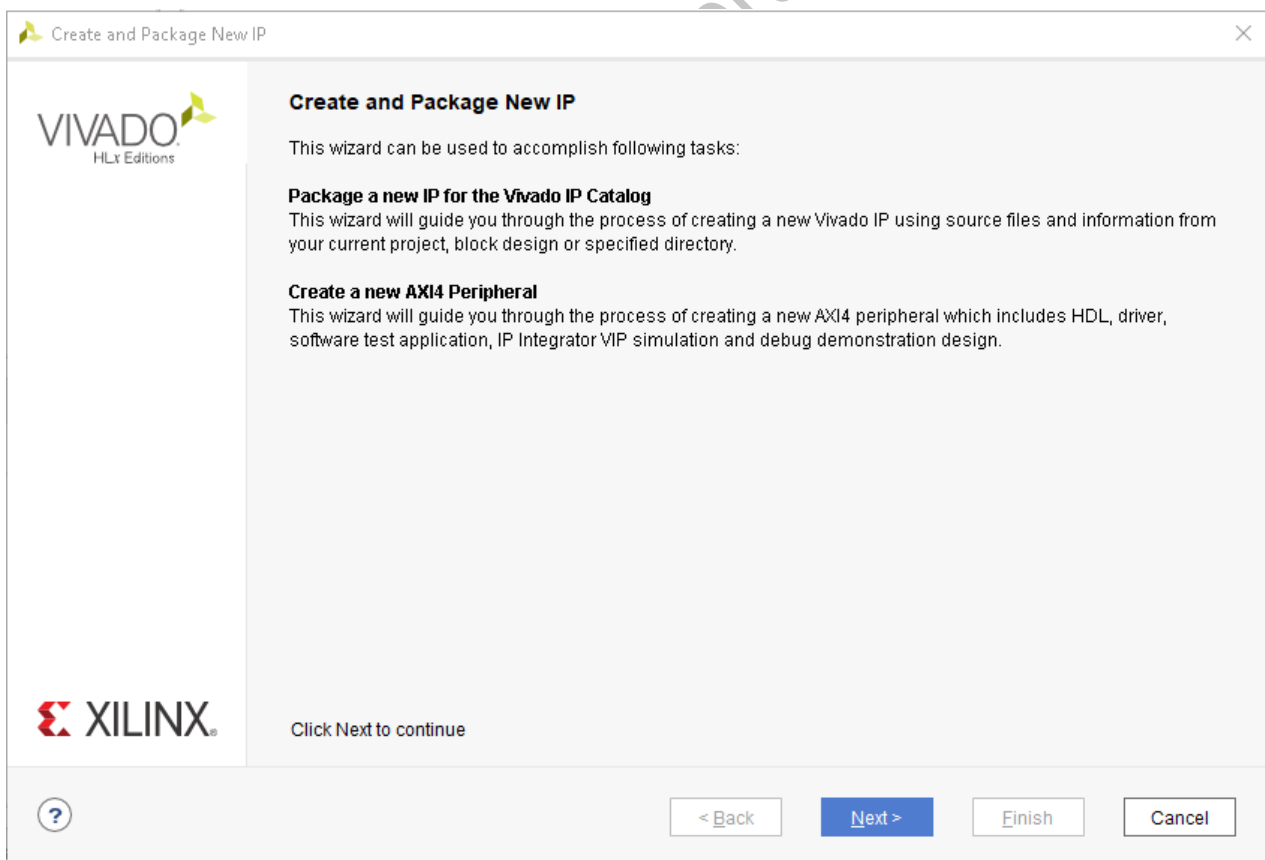
Para crear un bloque IP lo primero que hay que hacer es tener creado un Block Design, para ello en la pestaña IP INTEGRATOR se crea un Design



Y una vez abierto se abre el editor dónde se plasmará el diagrama de bloque(Diagram)



Ahora se crea un bloque IP, para ello en *Tools* hay una opción llamada *Create and Package New IP*. Pinchando en esa opción aparecen las pestañas en las que se configuran los bloques IP



En la pestaña anterior se informa de las diferentes opciones que existen para crear un bloque IP. En la siguiente pestaña se elige la opción para crear el bloque IP.

Create and Package New IP

Create Peripheral, Package IP or Package a Block Design

Please select one of the following tasks.

Packaging Options

- ☒ Package your current project
Use the project as the source for creating a new IP Definition.
- ☐ Package a block design from the current project
Choose a block design as the source for creating a new IP Definition.
Select a block design: design_2
- ☐ Package a specified directory
Choose a directory as the source for creating a new IP Definition.

Create AXI4 Peripheral

- ☐ Create a new AXI4 peripheral
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

? < Back Next > Finish Cancel

AXI4 : es la arquitectura de buses que utiliza Xilinx para interconectar bloques.

Para más información dejo el enlace del fabricante

<https://developer.arm.com/documentation/ihl0022/latest/>

Si se lo que se pretende es crear un nuevo bloque se tiene que elegir la opción **Create AXI4 Peripheral**. También se puede encapsular un proyecto de Vivado como un bloque ILA, encapsular un Block Design o encapsular un directorio con código HDL.

Si se elige esa opción se tiene que configurar los siguientes campos:

- *Name*: Este es el nombre del IP que se va crear
- *Version*: Esta es la versión del IP que se va crear
- *Display Name*: Este es nombre que se mostrará cuando se inserte el IP en el diagrama de bloques
- *Description*: Esta es la descripción que se va a dar del IP
- *IP location*: Esta es la ubicación en la que se va a crear el IP
- *Overwrite*: Esta es la casilla para sobrescribir el IP en caso de que ya exista

The screenshot shows a Windows-style dialog box titled "Create and Package New IP". Inside, the "Peripheral Details" tab is active, with the instruction "Specify name, version and description for the new peripheral". The form contains the following fields:

- Name:** "Este es mi IP" (highlighted with a red border)
- Version:** "1.0" (highlighted with an orange border)
- Display name:** "NOMBRE_IP" (highlighted with a yellow border)
- Description:** "Este es el IP creado para SoC-eame" (highlighted with a green border)
- IP location:** "C: /..ip_repo" (highlighted with a purple border, with a file explorer icon to the right)
- Overwrite existing:** An unchecked checkbox.

At the bottom, there is a help icon (?), and four buttons: "< Back", "Next >" (highlighted in blue), "Finish", and "Cancel".

Con los detalles del periférico ya configurado se pasa a la siguiente pestaña *Add Interfaces* en la que se definirán los detalles técnicos del IP.

Entre los detalles técnicos se tiene:

- El número de interfaces de entrada, por norma general con una es suficiente
- *Name*: El nombre de la interfaz de entrada, es opcional cambiarlo
- *Interface Type*: El tipo de interfaz, con el tipo Lite es suficiente para muchos proyectos debido a que no se utilizará el Burst
- *Interface Mode*: El modo de la interfaz, si va a ser **Master** de otras interfaces o va a ser **Slave** de otras interfaces
- *Data Width (Bits)*: Ancho de datos de la interfaz, por norma general se suele utilizar de 32 bits
- *Number of Registers*: Número de registros, este parámetro es muy importante porque es el número de registros que el Core podrá leer o escribir del bloque IP, con 4 suele ser suficiente si se saben administrar todos los bits pero si el número de bits a trabajar supera los 128 bits o los cuatro registros hay que aumentarlos.

The screenshot shows the 'Create and Package New IP' dialog box with the 'Add Interfaces' tab selected. The dialog has a title bar with a yellow logo and a close button. The main area is divided into three sections. The left section has a checkbox 'Enable Interrupt Support' and a diagram of an interface block labeled 'Entrada' with 'NOMBRE_IP' below it. The middle section is a tree view showing 'Interfaces' with a sub-item 'Entrada'. The right section contains configuration fields: 'Name' (Entrada), 'Interface Type' (Lite), 'Interface Mode' (Slave), 'Data Width (Bits)' (32), 'Memory Size (Bytes)' (64), and 'Number of Registers' (4). At the bottom are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

Create and Package New IP

Add Interfaces
Add AXI4 interfaces supported by your peripheral

☐ Enable Interrupt Support

Interfaces
Entrada

Entrada
NOMBRE_IP

Name: Entrada

Interface Type: Lite

Interface Mode: Slave

Data Width (Bits): 32

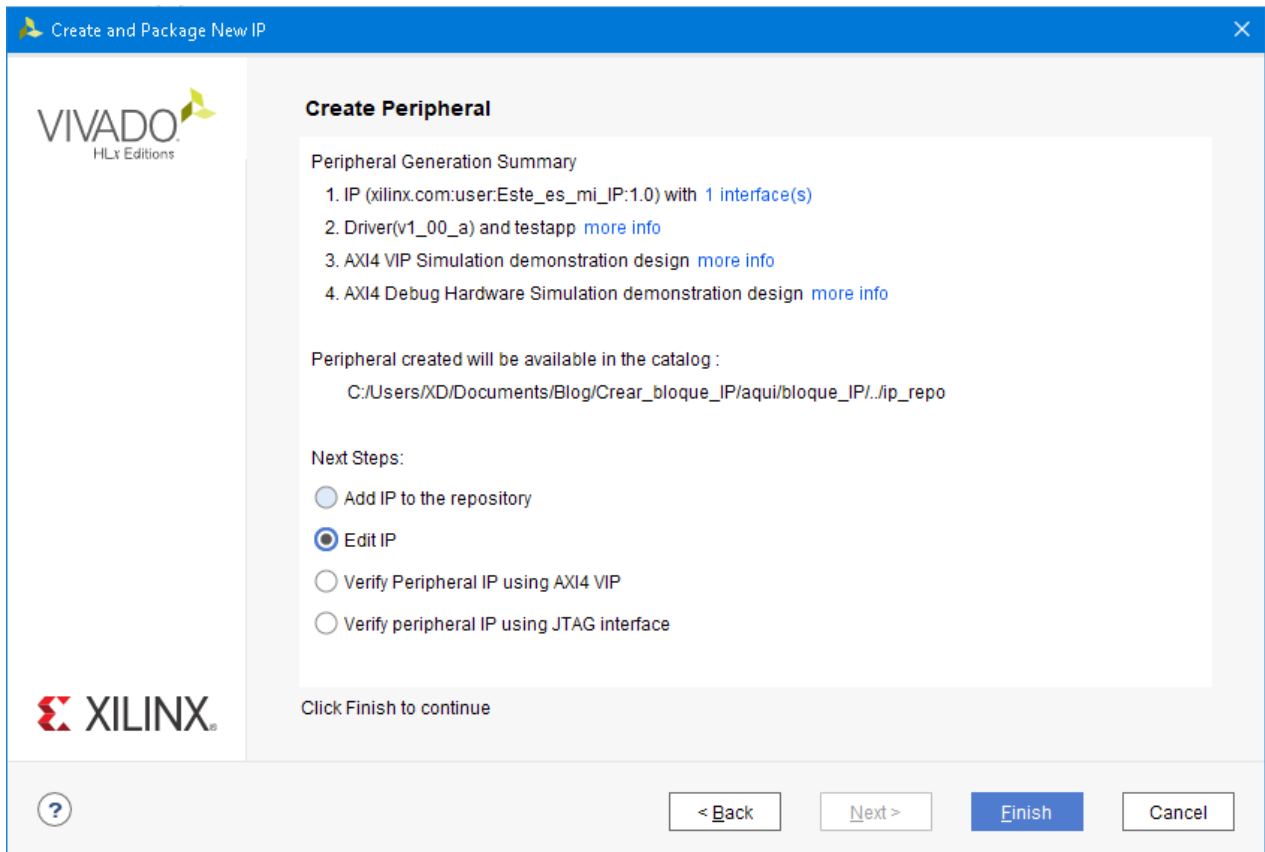
Memory Size (Bytes): 64

Number of Registers: 4 [4..512]

? < Back Next > Finish Cancel

Una vez la interfaz está creada se le dice al sistema que es lo que se quiere hacer.

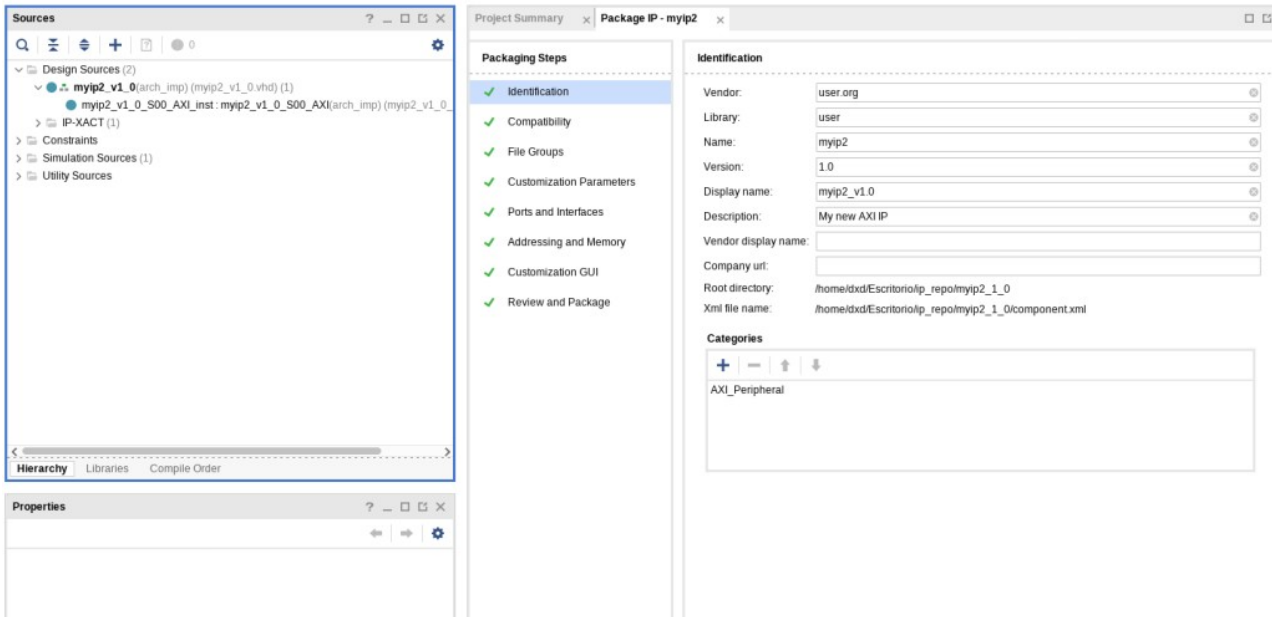
Si lo que se quiere es programarlo se marca la opción *Edit IP*



Y una vez se ha terminado la configuración se abre una pestaña en Vivado en la que se implementará el IP. Esta nueva pestaña es un **entorno aparte** del Diagrama de bloques, por lo que se puede simular solo el IP que se está creando, para no interferir con los demás.

Programar un IP

Para poder crearlo en la pestaña que se abre aparecen dos ficheros VHDL, el top es el IP y el bottom es el IPIF.



Si se abren lo que se ve es una interfaz del bus AXI que no es necesario tocarla, lo único que hace falta tocar del fichero IP son las conexiones de puertos. Todo lo demás va al IPIF, en el IPIF se declaran todos los módulos útiles que se comunican por AXI.

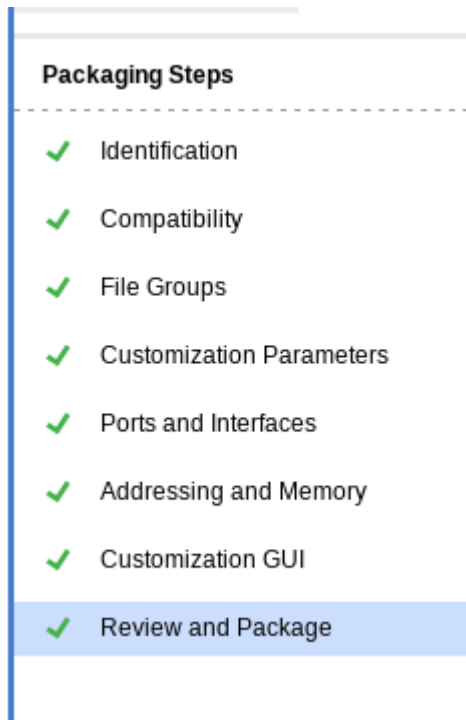
La comunicación AXI se realiza por registros que son estas señales del IPIF.

```
----- Signals for user logic register space example -----  
----- Number of Slave Registers 4 -----  
signal slv_reg0 :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);  
signal slv_reg1 :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);  
signal slv_reg2 :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);  
signal slv_reg3 :std_logic_vector(C_S_AXI_DATA_WIDTH-1 downto 0);
```

Estas señales son las que se asignan como entrada o salida de los módulos que definamos. Por lo que si escribimos en el bit 0 del registro cero, atacamos el bit 0 de la señal slv_reg0 (las señales se pueden cambiar de nombre, pero cuidado con el orden en el que recogen los datos del bus AXI). Es importante tener en cuenta que hay señales que se pueden estar escribiendo desde dos sitios distintos, es importante revisarlo y comentar aquellos del IPIF en los que esta circunstancia se dé.

Una vez hallamos terminado de diseñar el bloque IP **se recomienda sintetizarlo** desde la propia ventana para comprobar que todo este correcto.

Cuando se tenga se vuelve a la pestaña de inicio del bloque IP y se actualizan todos los campos. Todo tiene que aparecer en verde para indicar que todo está correcto.



Y se pulsa Re-Package IP. Y se cierra la pestaña, a partir de este momento ya se puede llamar al bloque IP en el Block design.

Si el bloque IP es para un sistema programable (Zynq, Microblaze) se le adjudicará una dirección de memoria. Esta dirección de memoria será la dirección de memoria en la que comienza el registro 0.

Para atacarlos bits se puede utilizar las funciones Xil_IO.h

```
#include "xil_io.h"
```

```
...
```

```
// escribir en un bit de una dirección de 32 bits
```

```
Xil_Out32(<dirección registro(mirar xparameters.h>, 0x1);
```

```
// para leer
```

```
uint32_t dato = Xil_In32(<dirección del registro>;
```