

# **Cómo incluir tus propios drivers en los bloques IP que exportas**

Creador: David Rubio G.

Entrada: <https://soceame.wordpress.com/2024/06/30/como-incluir-tus-proprios-drivers-en-los-bloques-ip-que-exportas/>

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

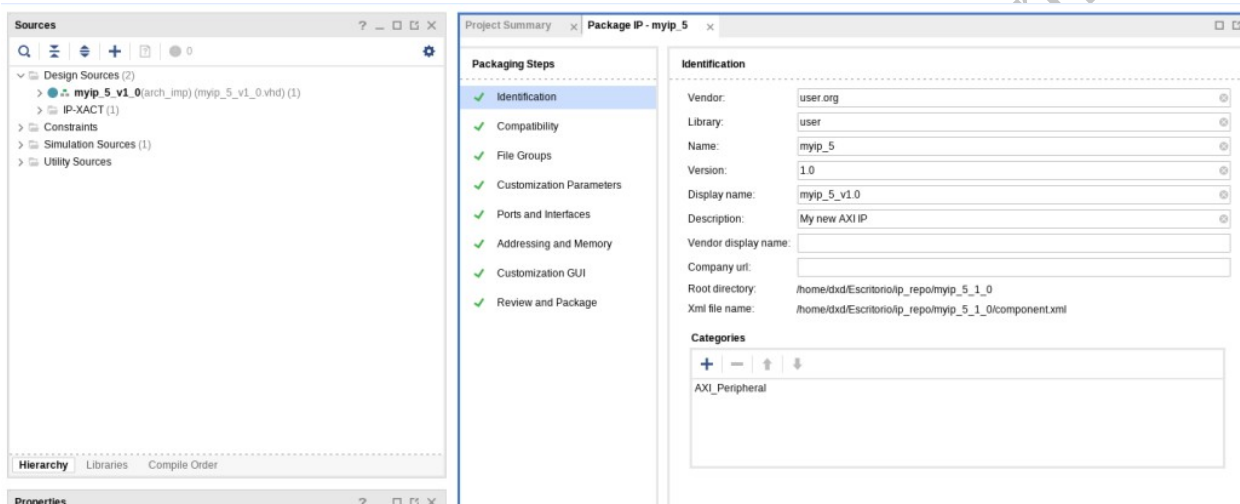
GitHub: <https://github.com/DRubioG>

Fecha última modificación: 23/02/2025

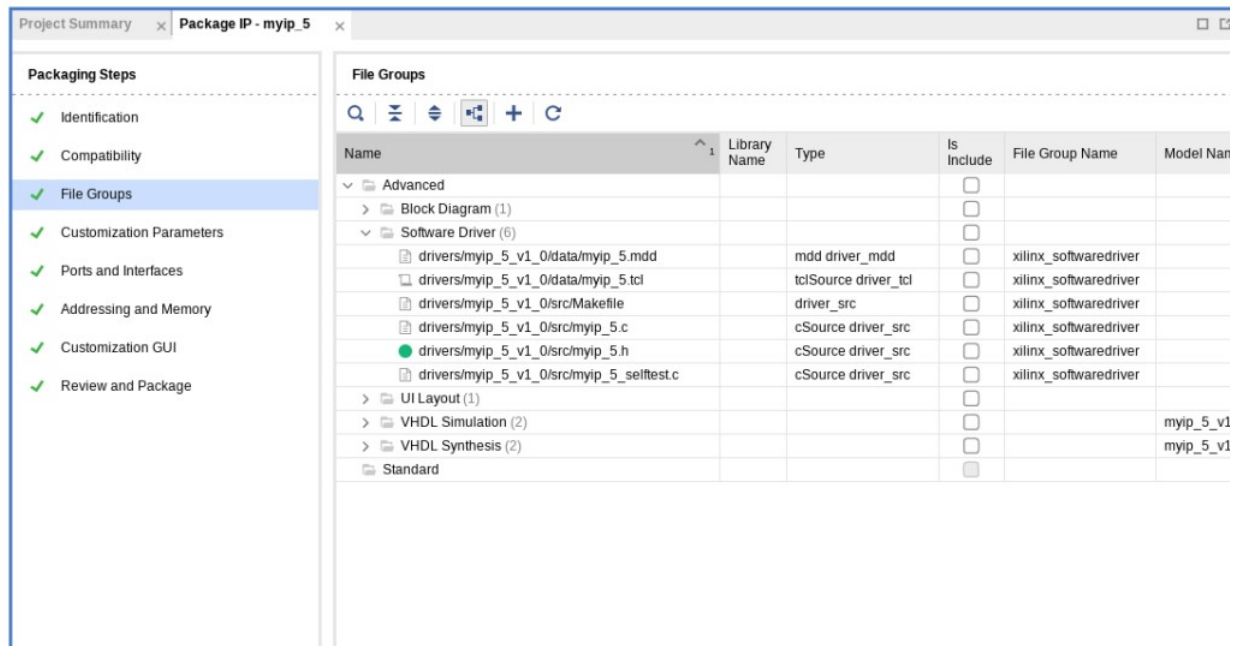
Cuando generas un bloque IP es muy recurrente el tener que generar unos drivers que suelen ir separados del bloque IP y que al generar la plataforma en Vitis tienes que importar desde el exterior. Bien, pues hay una forma de cuando hagas la exportación a Vitis los drivers que hacen funcionar el IP estén dentro a la plataforma. Te explico más detenidamente cómo se hace.

## Vivado

Lo primero que tienes que hacer es generar un bloque IP con la funcionalidad que quieres.

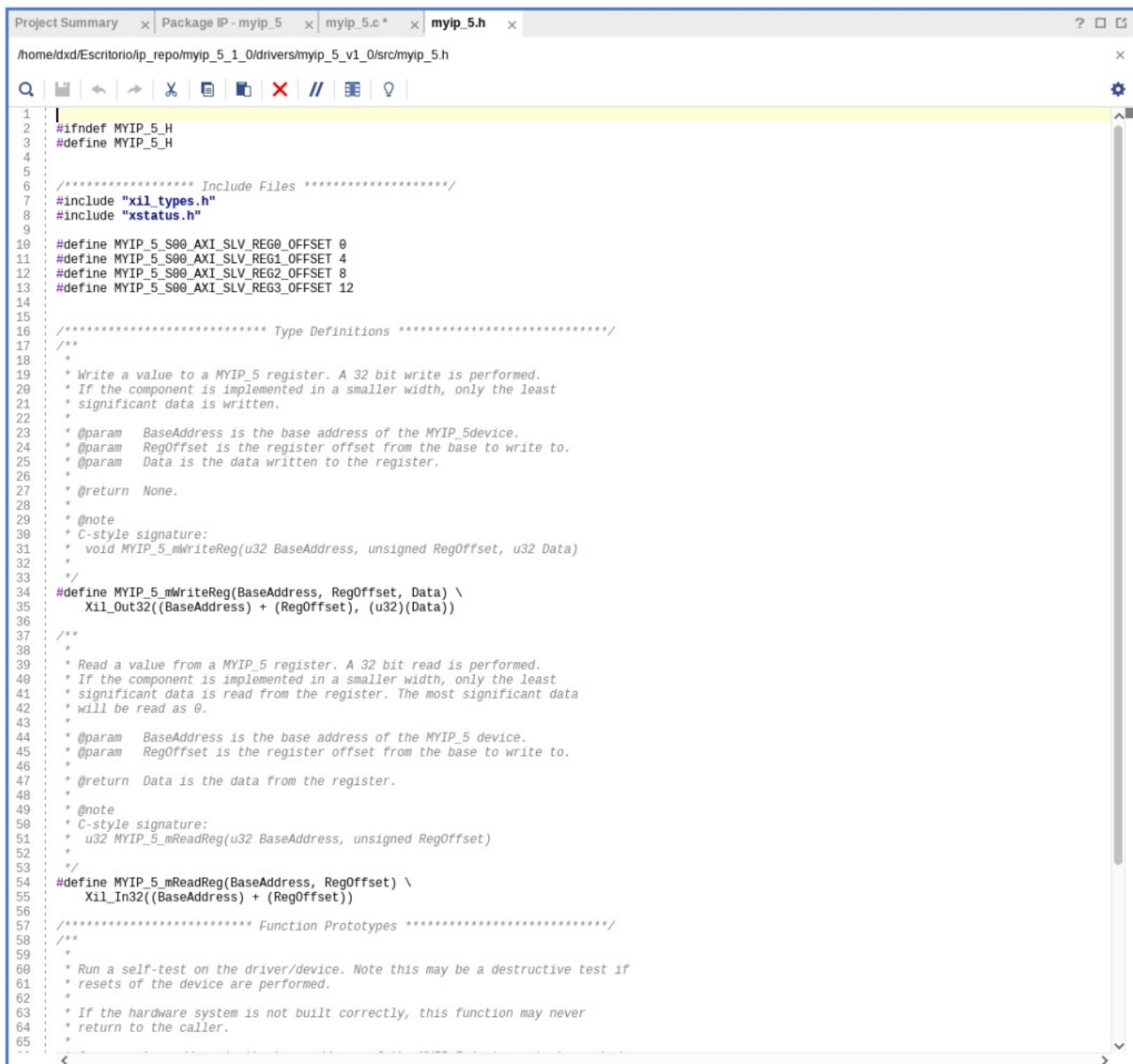


En el File Groups aparece opción Software Driver. Aquí aparecen los ficheros <nombre del IP>.h y .c.



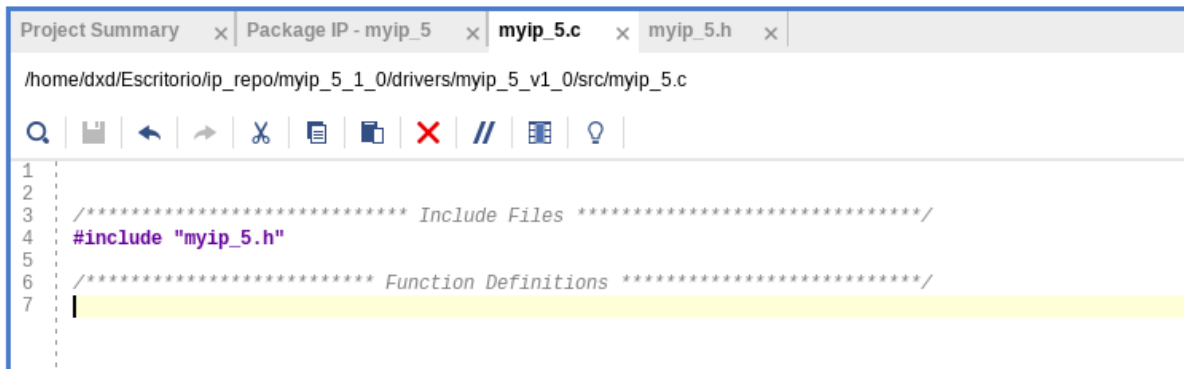
Estos son los drivers propios del IP, que puedes modificar o sustituir por los drivers que quieres.

Si abres el .h tiene el código de la declaración de las funciones.



```
1 |
2 | #ifndef MYIP_5_H
3 | #define MYIP_5_H
4 |
5 |
6 | /***** Include Files *****/
7 | #include "xil_types.h"
8 | #include "xstatus.h"
9 |
10 | #define MYIP_5_S00_AXI_SLV_REG0_OFFSET 0
11 | #define MYIP_5_S00_AXI_SLV_REG1_OFFSET 4
12 | #define MYIP_5_S00_AXI_SLV_REG2_OFFSET 8
13 | #define MYIP_5_S00_AXI_SLV_REG3_OFFSET 12
14 |
15 | /***** Type Definitions *****/
16 | /**
17 | *
18 | * Write a value to a MYIP_5 register. A 32 bit write is performed.
19 | * If the component is implemented in a smaller width, only the least
20 | * significant data is written.
21 | *
22 | * @param BaseAddress is the base address of the MYIP_5 device.
23 | * @param RegOffset is the register offset from the base to write to.
24 | * @param Data is the data written to the register.
25 | *
26 | * @return None.
27 | *
28 | * @note
29 | * C-style signature:
30 | * void MYIP_5_mWriteReg(u32 BaseAddress, unsigned RegOffset, u32 Data)
31 | *
32 | */
33 | #define MYIP_5_mWriteReg(BaseAddress, RegOffset, Data) \
34 |     Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))
35 |
36 | /**
37 | *
38 | * Read a value from a MYIP_5 register. A 32 bit read is performed.
39 | * If the component is implemented in a smaller width, only the least
40 | * significant data is read from the register. The most significant data
41 | * will be read as 0.
42 | *
43 | * @param BaseAddress is the base address of the MYIP_5 device.
44 | * @param RegOffset is the register offset from the base to write to.
45 | *
46 | * @return Data is the data from the register.
47 | *
48 | * @note
49 | * C-style signature:
50 | * u32 MYIP_5_mReadReg(u32 BaseAddress, unsigned RegOffset)
51 | *
52 | */
53 | #define MYIP_5_mReadReg(BaseAddress, RegOffset) \
54 |     Xil_In32((BaseAddress) + (RegOffset))
55 |
56 | /***** Function Prototypes *****/
57 | /**
58 | *
59 | * Run a self-test on the driver/device. Note this may be a destructive test if
60 | * resets of the device are performed.
61 | *
62 | * If the hardware system is not built correctly, this function may never
63 | * return to the caller.
64 | *
65 | */
```

Y en el .c el código ejecutable, esto aparece vacío porque no tiene función alguna definida.



```
1
2
3  /***** Include Files *****/
4  #include "myip_5.h"
5
6  /***** Function Definitions *****/
7  |
```

**NOTA:** también aparece un fichero *selftest.c* al que tampoco hay que hacerle mucho caso. Al final es un fichero que contiene las funciones para testear el bloque IP. Estas funciones se tienen que utilizar dónde se quiera hacer las pruebas.

Bien, pues para importar drivers que se han creado para este IP se tiene que dar clic derecho en *Software Driver* > *Add files* (los ficheros tienen que estar si creados, porque no deja crearlos sobre la marcha).

Para añadirlos aparece una pestaña como esta que se tiene que rellenar con los drivers que se quieren importar.

**NOTA:** los driver por defecto que crea Vivado al crear un bloque IP están en la siguiente ruta: `../ip_repo/<nombre IP>/drivers/<nombre IP>/src` . **En esta ruta se recomienda importar el código de los drivers.**

**NOTA 2:** al añadir los drivers desde fuera aparece abajo una casilla que pone «*Copy sources into IP Directory*», esta casilla lo que hace es una copia del driver y la mete dentro del IP, al meterlo crea una carpeta llamada «src» en el directorio principal del IP.

## Vitis

Una vez hemos generado el Bitstream y hemos exportado el hardware, creamos una plataforma en Vitis basándonos en el ese hardware.

Al hacer eso automáticamente Vitis importa los drivers del IP. Los puedes localizar en la ruta: `<plataforma>/ps7_cortexa9_0/standalone_ps7_cortexa9_0/bsp/ps7_cortexa9_0/libsrc/<nombre del IP>/src`



Para utilizar los drivers en el código solo es necesario declarar el bloque .h del IP

```
#include <stdio.h>
#include "platform.h"
#include "myip_5.h"
```

Y para usar las funciones internas solo hace falta llamarlas con los parámetros necesarios

```
MYIP_5_Reg_SelfTest( p);
```

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