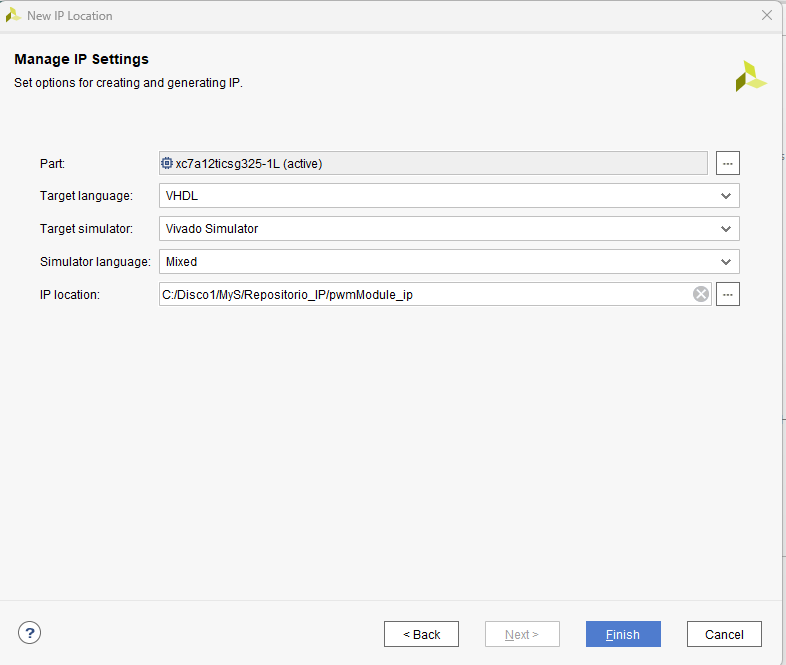
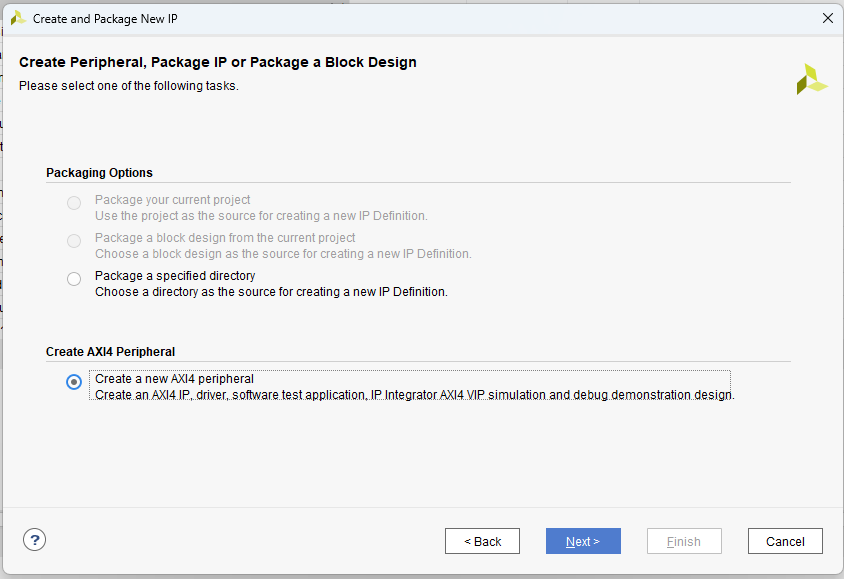
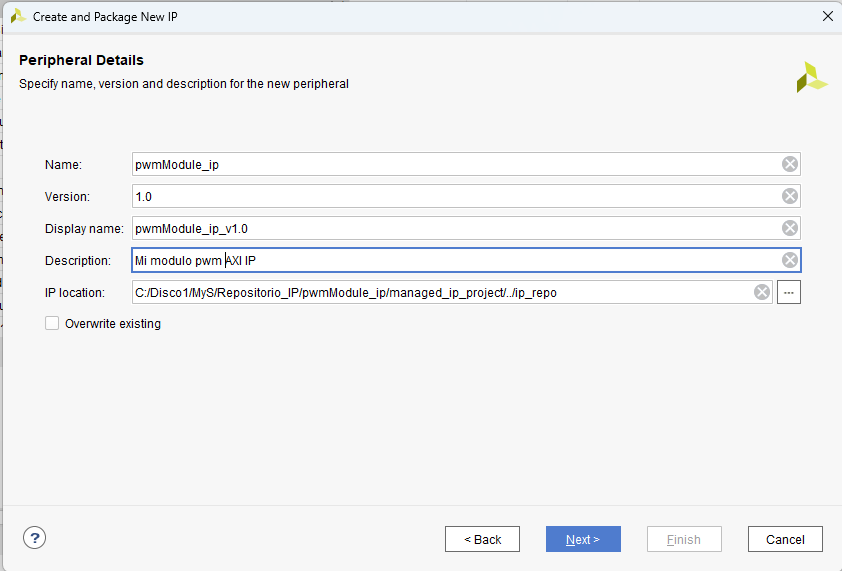
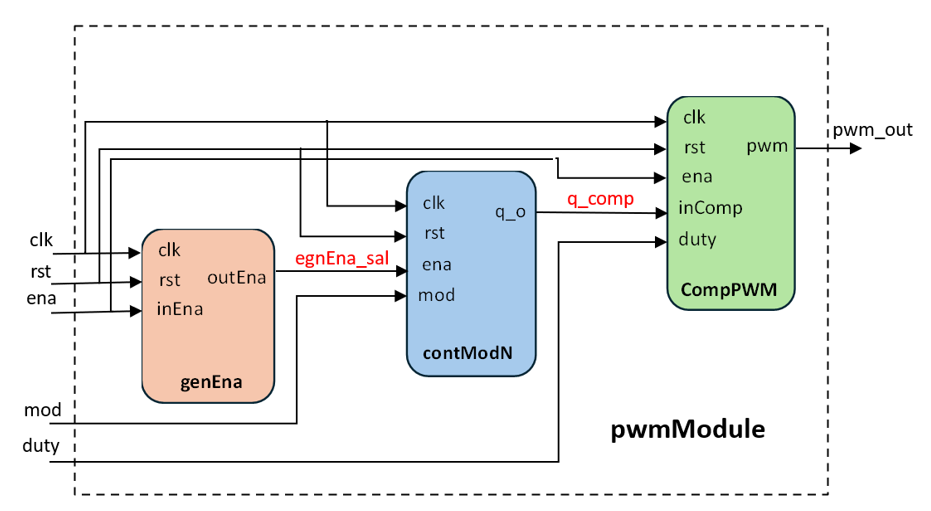
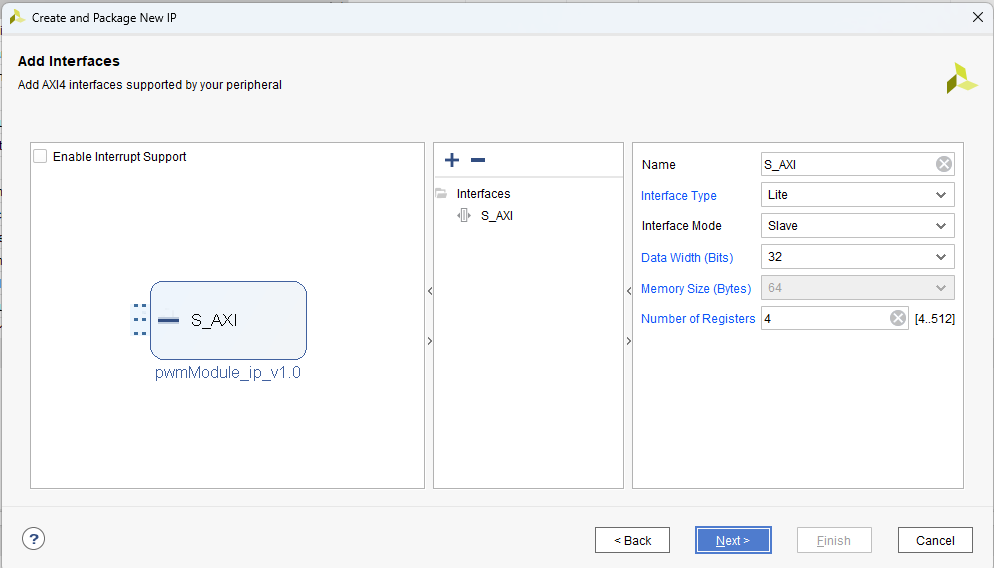
Sistema pwmModule\_ip

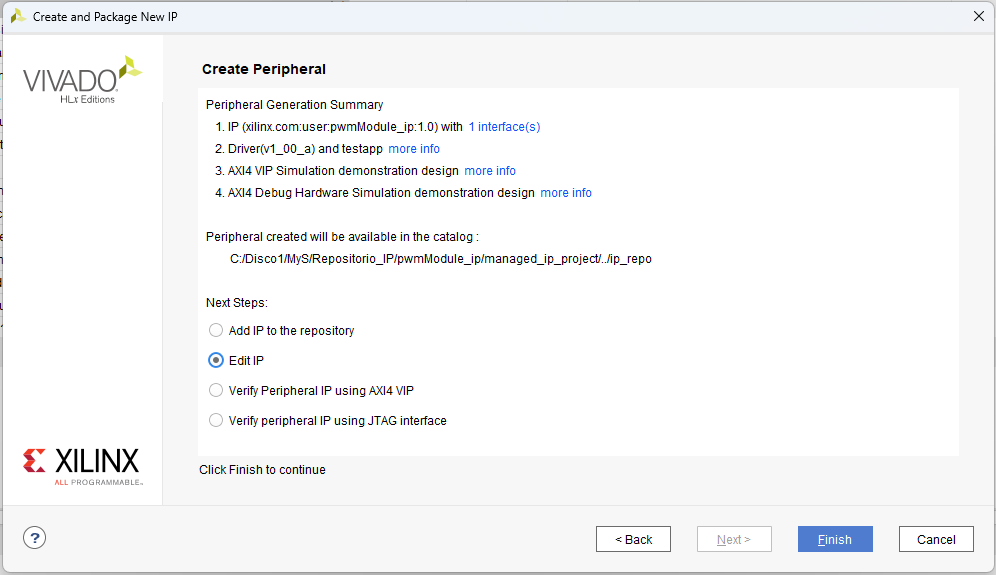




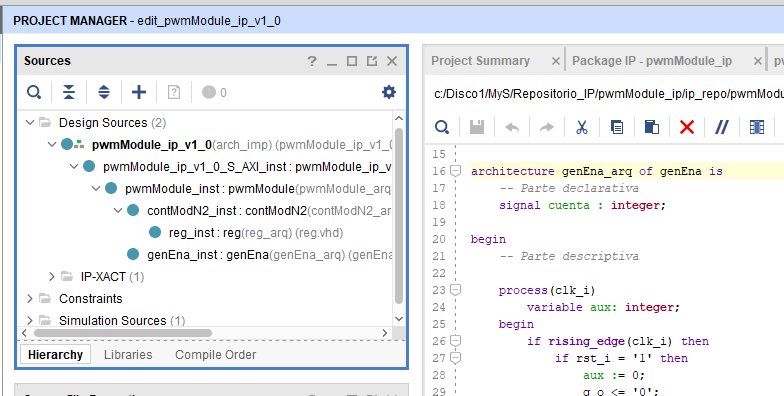




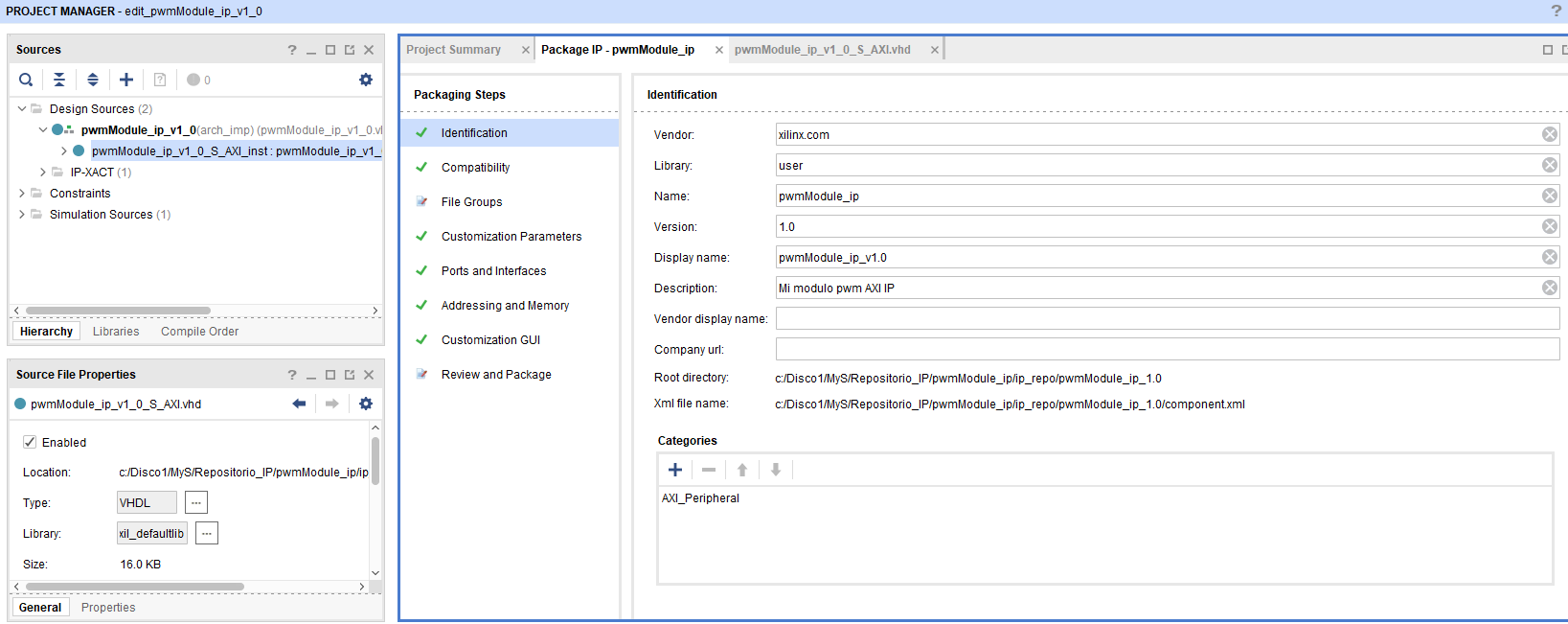




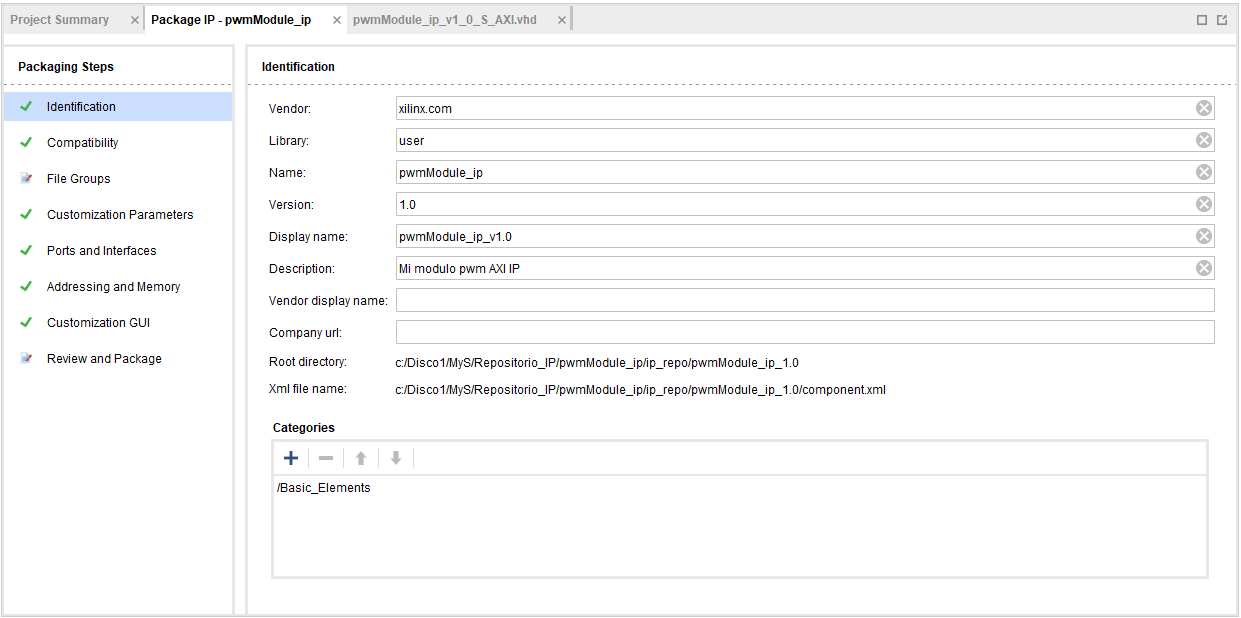
Una vez creado el wraper para mi modulo voy agregando los fuentes del contador, gen ena y reg uno a uno en el lugar que les corresponde en la jerarquía de archivos



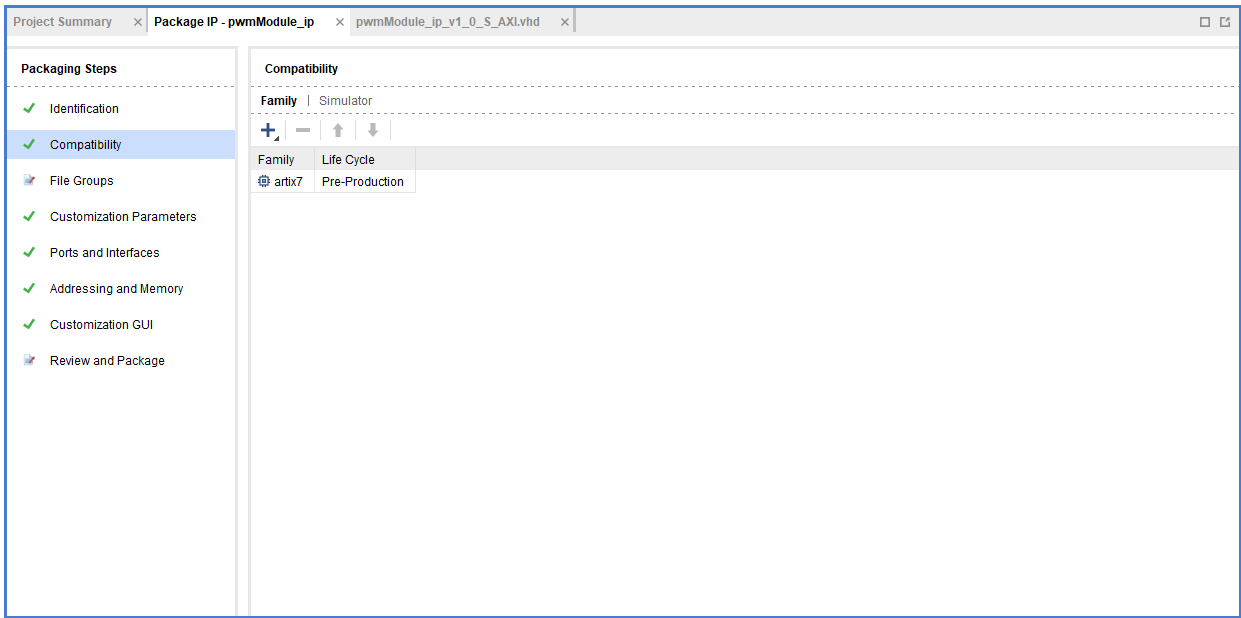
Solapa Package IP – pwmModule\_ip



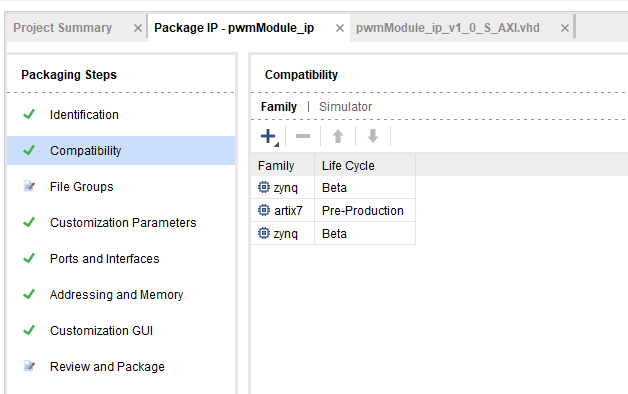
Identification🡪 cambiamos Categoría en la que se encontrará nuestro modulo pwm 🡪 Basic\_Elements y sacamos la que tenía por defecto: AXI\_Peripheral



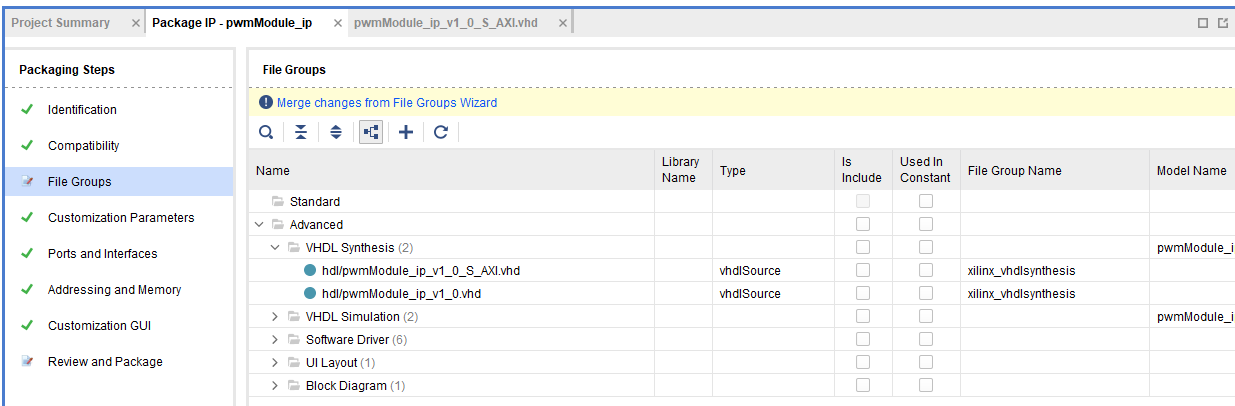
Compatibility (Configuramos compatibilidad muy importante para que después otro proyecto con nuestra placa lo pueda tomar)



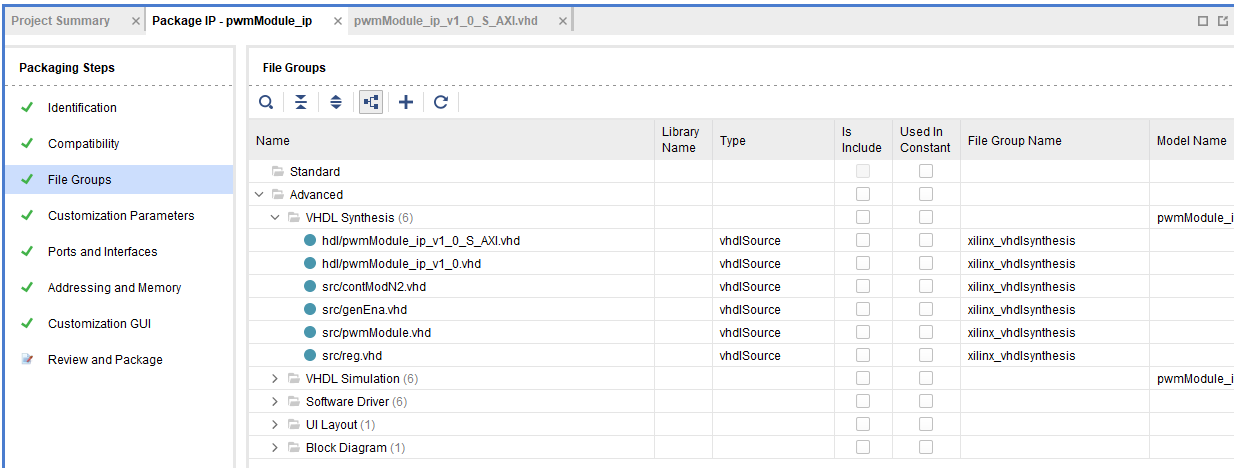
Establezco compatibilidad con zynq que incluye nuestra placa de desarrollo



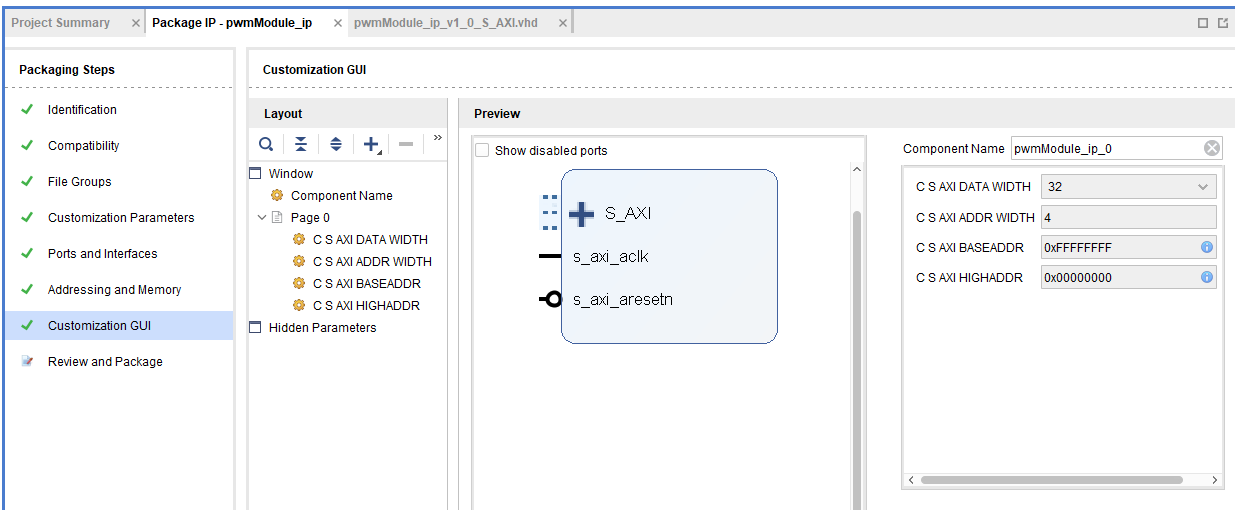
File Group agrego los archivos VHDL que faltan incluir



Le doy a Merge para incluir los archivos que faltan

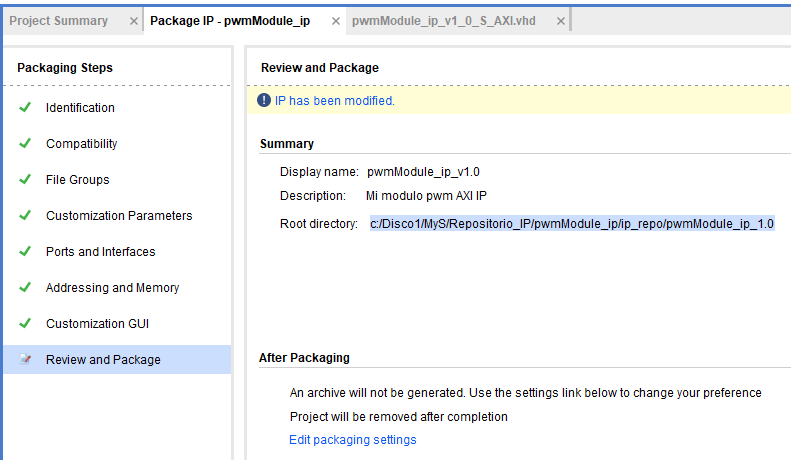


Customitation GUI



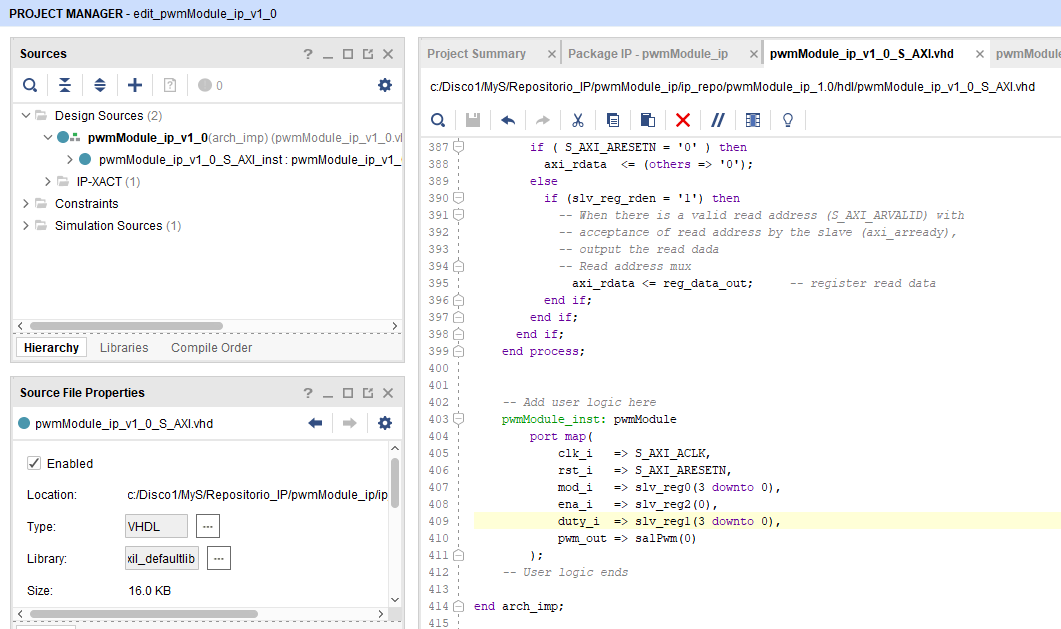
Vemos como quedaría el modulo

Review and Package (veo aquí donde se cerará mi modulo)

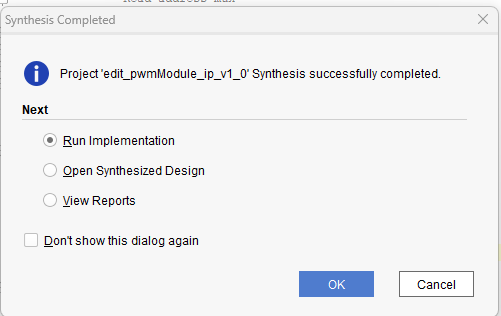


Antes de empaquetar nuestro modulo probamos si sintetiza en corriendo Run Synthesis

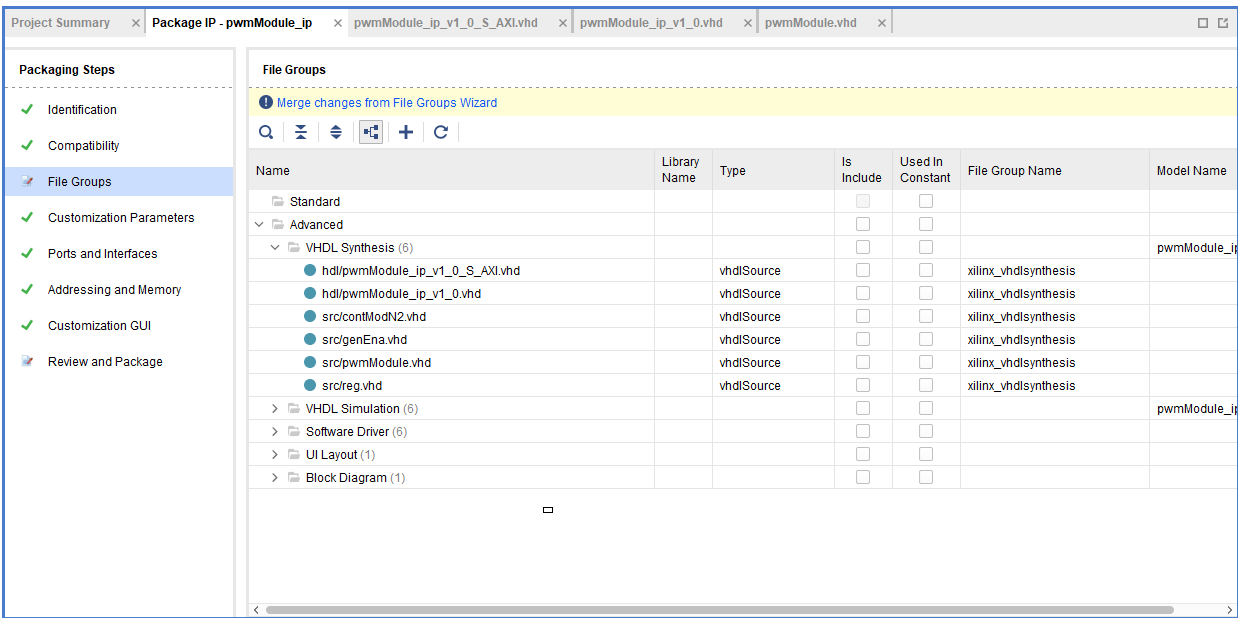
Problemas al sintetizar con el ancho de los registros AXI



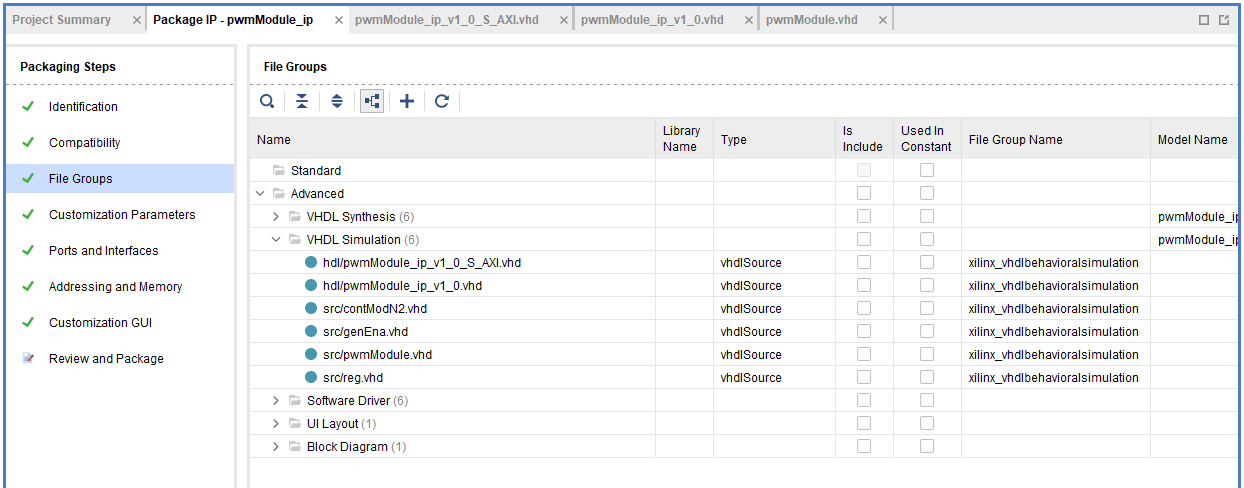
Los registros AXI por defecto tienen un ancho de 32 bits y mi modulo trabaja con 4 bits. Restringi los registros AXI a 4 bits para que sean coherentes con los que usa el modulo pwmModule y sintetizo sin errores esta vez.



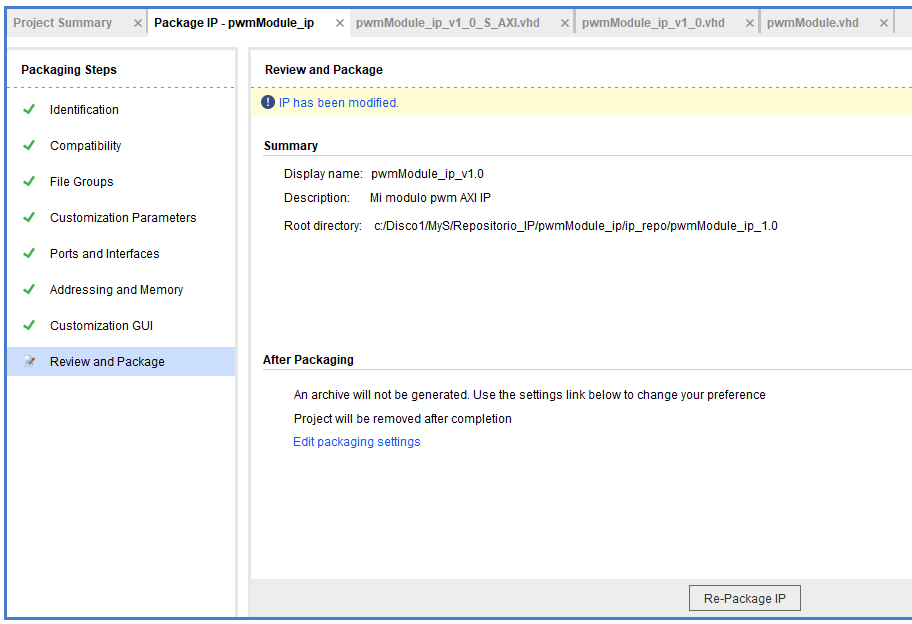
Como modifique mi archivo de entidad superior como se indica arriba, vuelvo a actualizar los archivos del ip cord haciendo Merge en File Groups



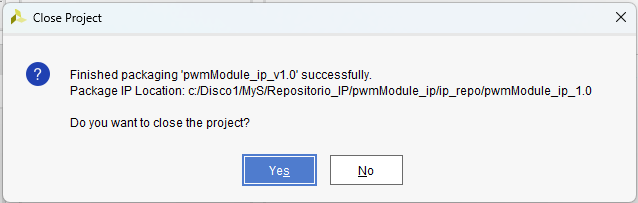
Luego de Merge queda todo ok con los archivos VHDL



Ahora empaqueto mi ip cord 🡪 Review and Packaging



Re-Packaging IP

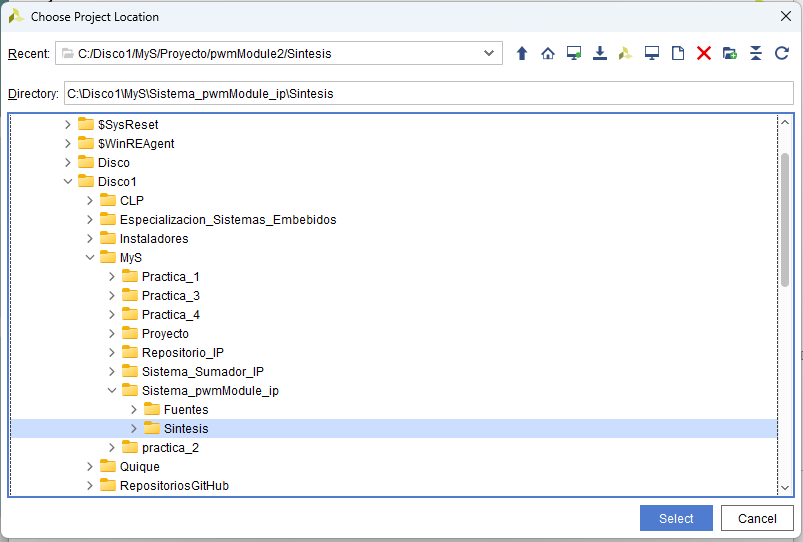


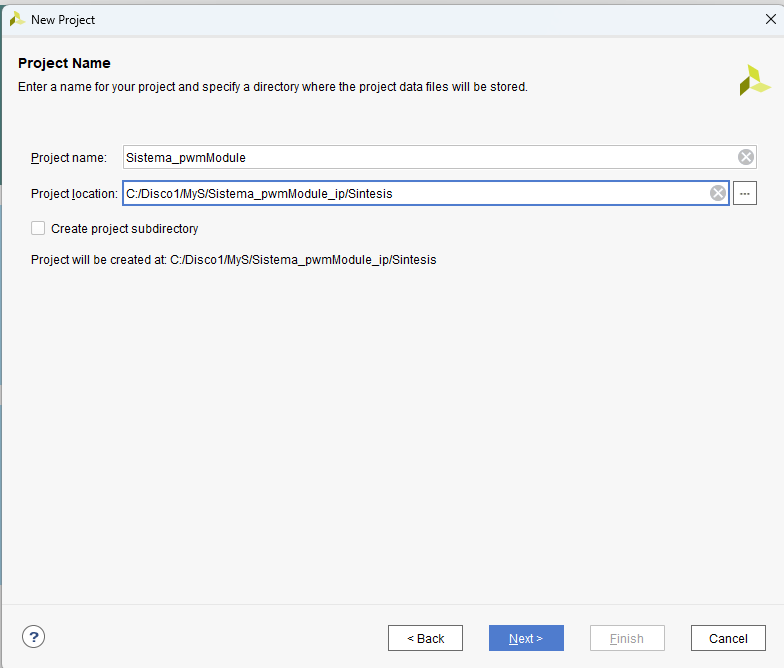
Listo, fue creado mi ip cord “pwmModule\_ip”

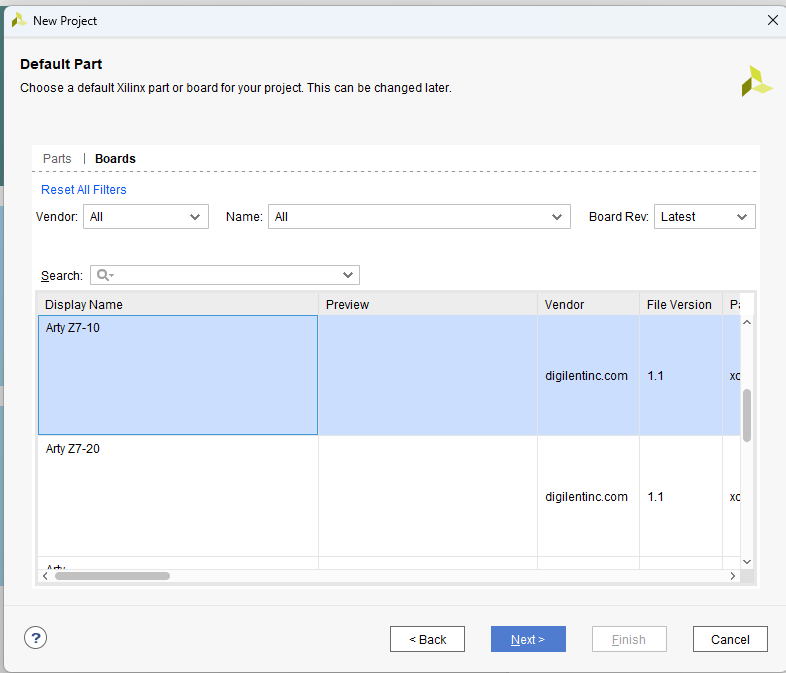
**PASO 2**

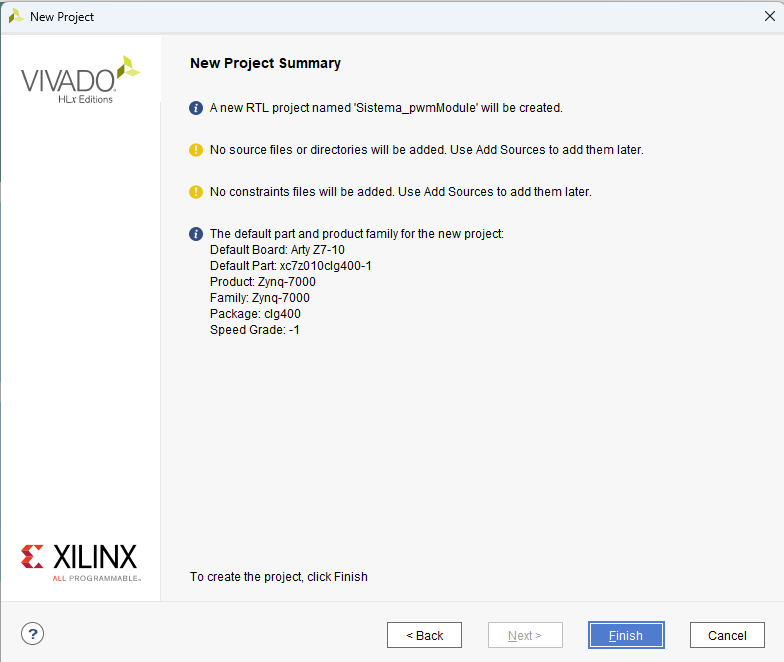
**Creando mi proyecto sistema\_pwmModule**

Creo nuevo proyecto y le doy nombre, luego le digo a vivado donde crara el proyecto:

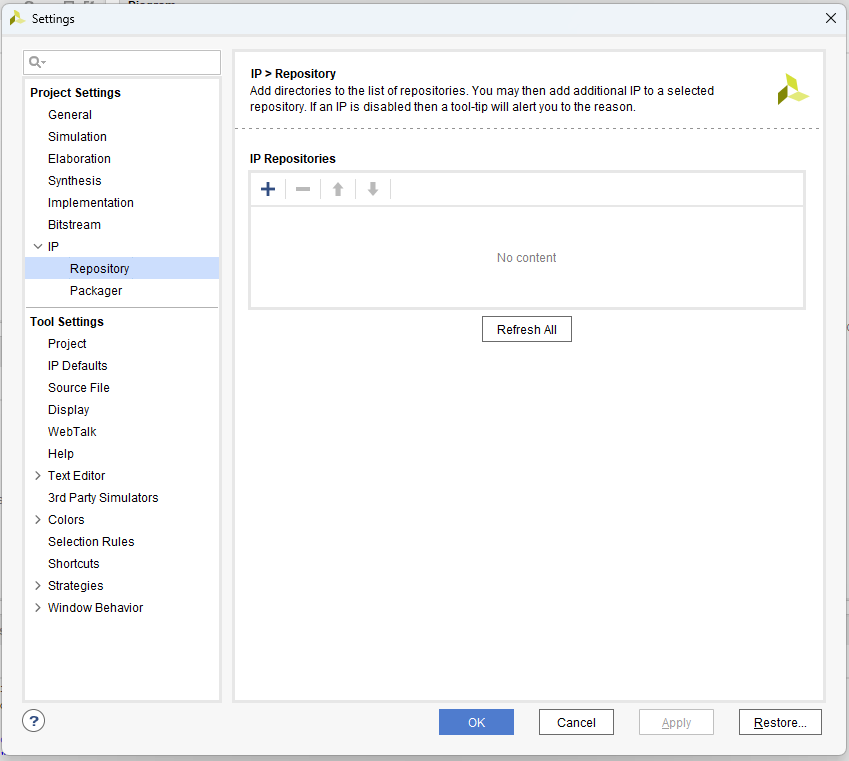


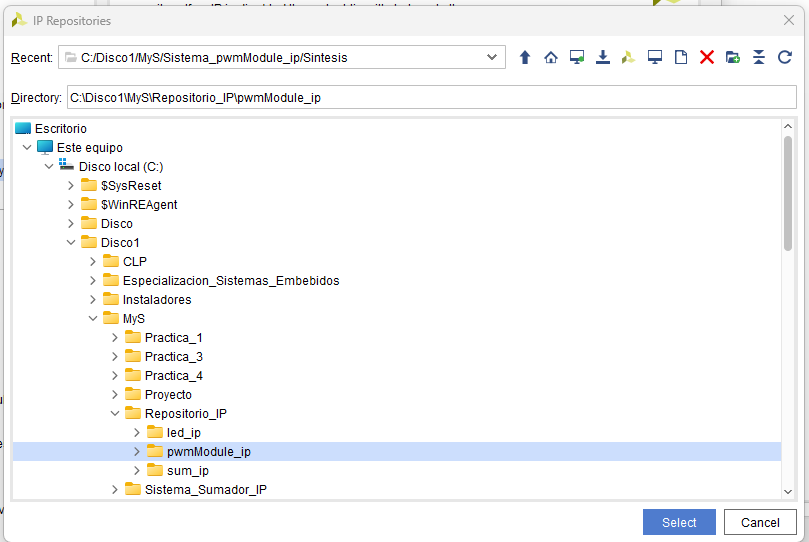


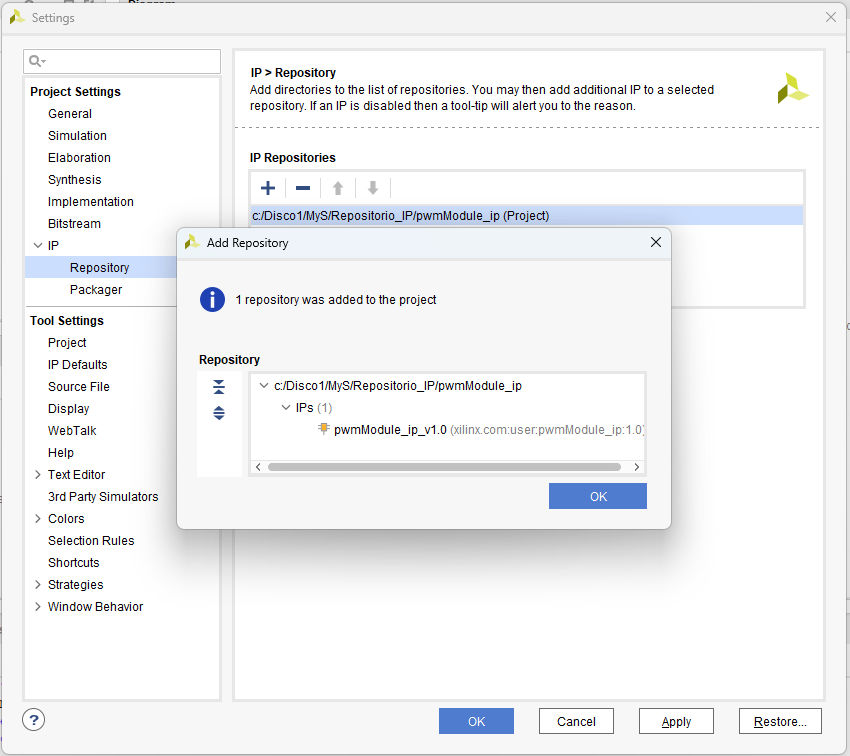


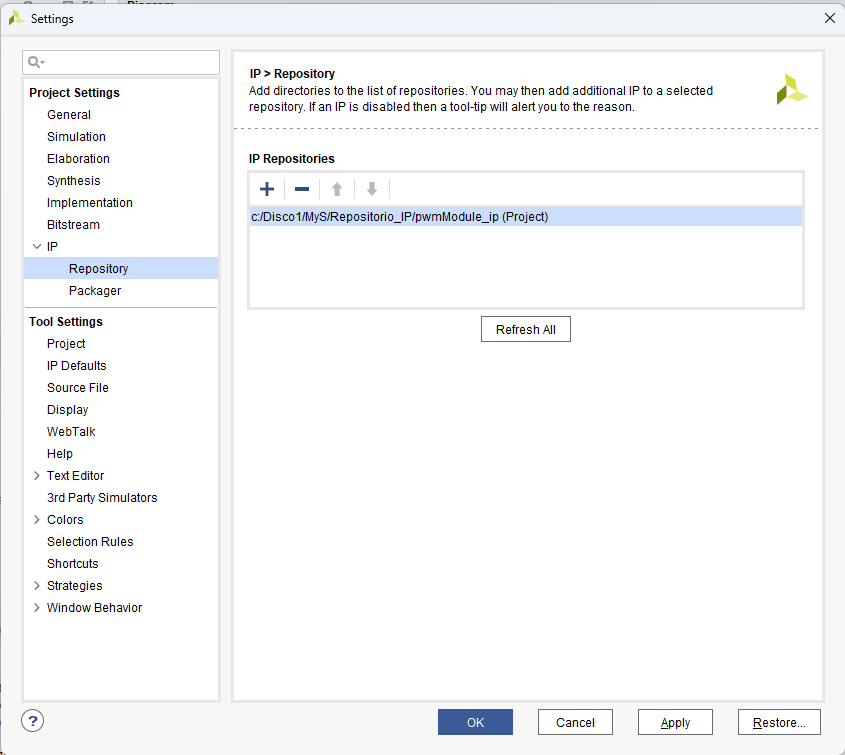


Le indico a Vivado donde encontrar el repositorio con los ip cords personalizados



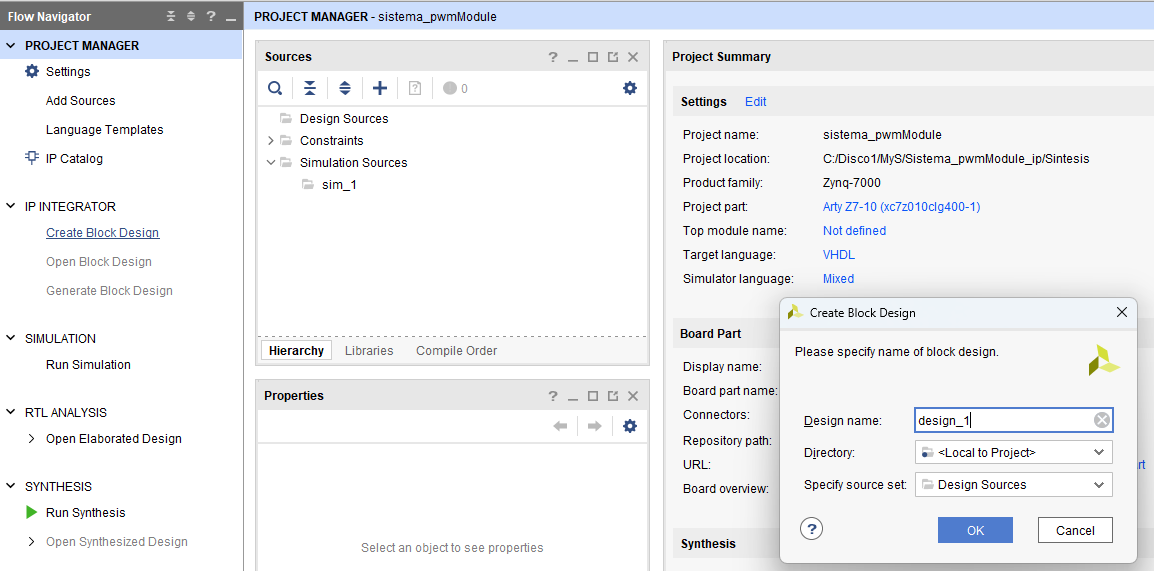


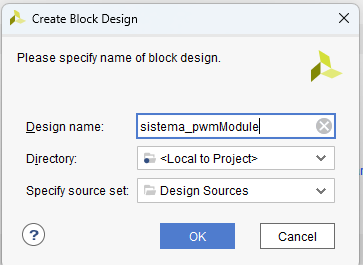




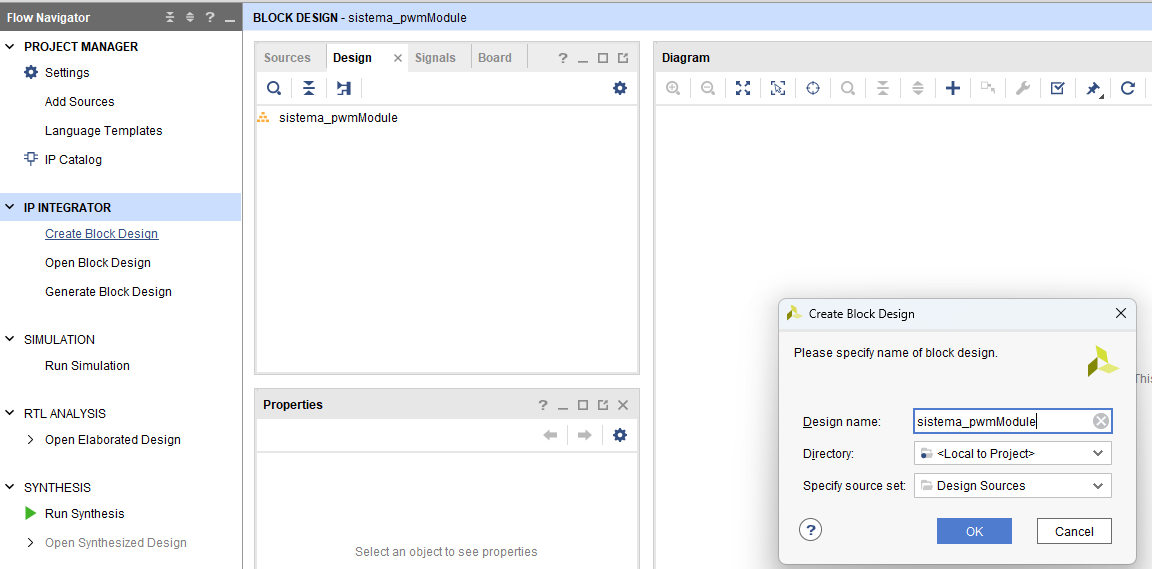
Apply y luego Ok

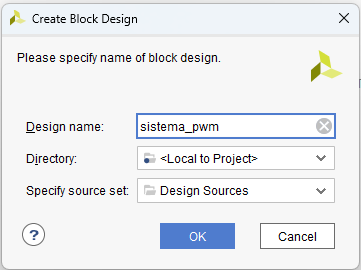
**Ahora incluyo mi ip cord personalizado al proyecto**



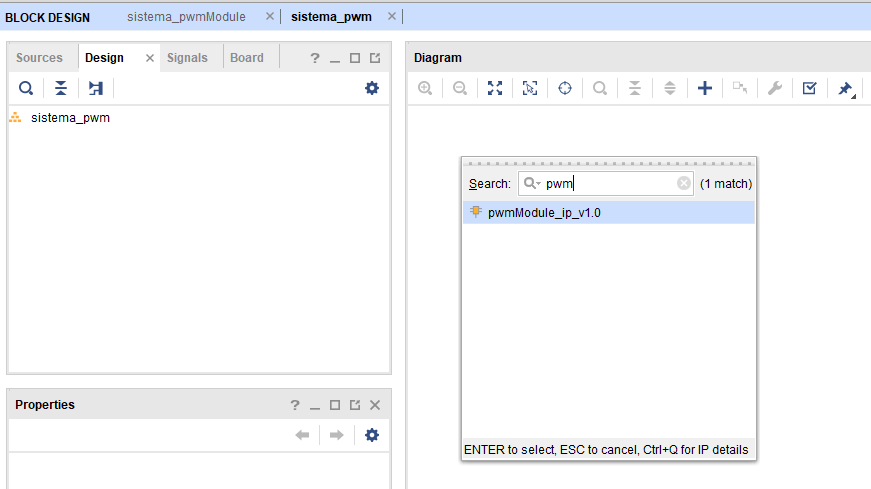


Create Block Design

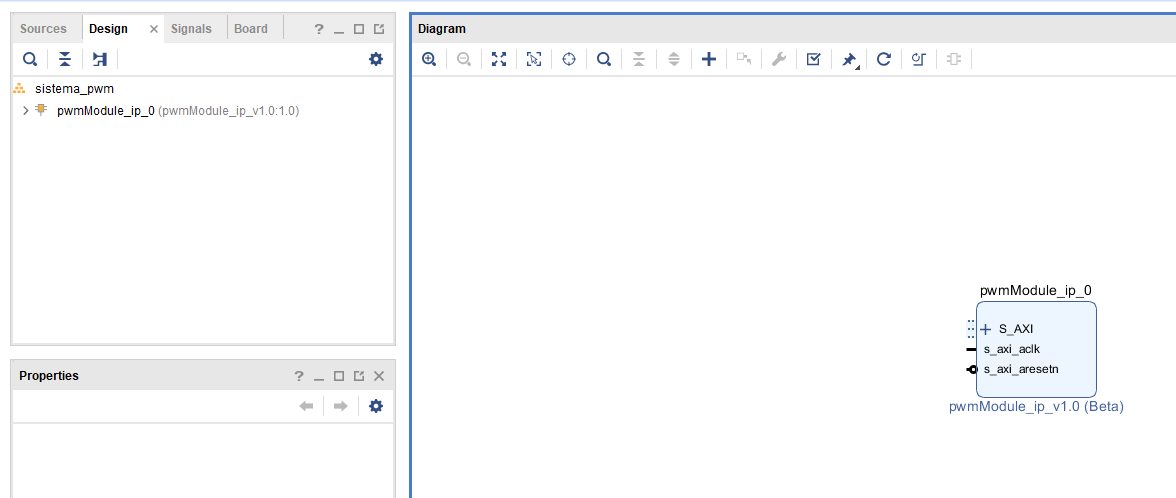




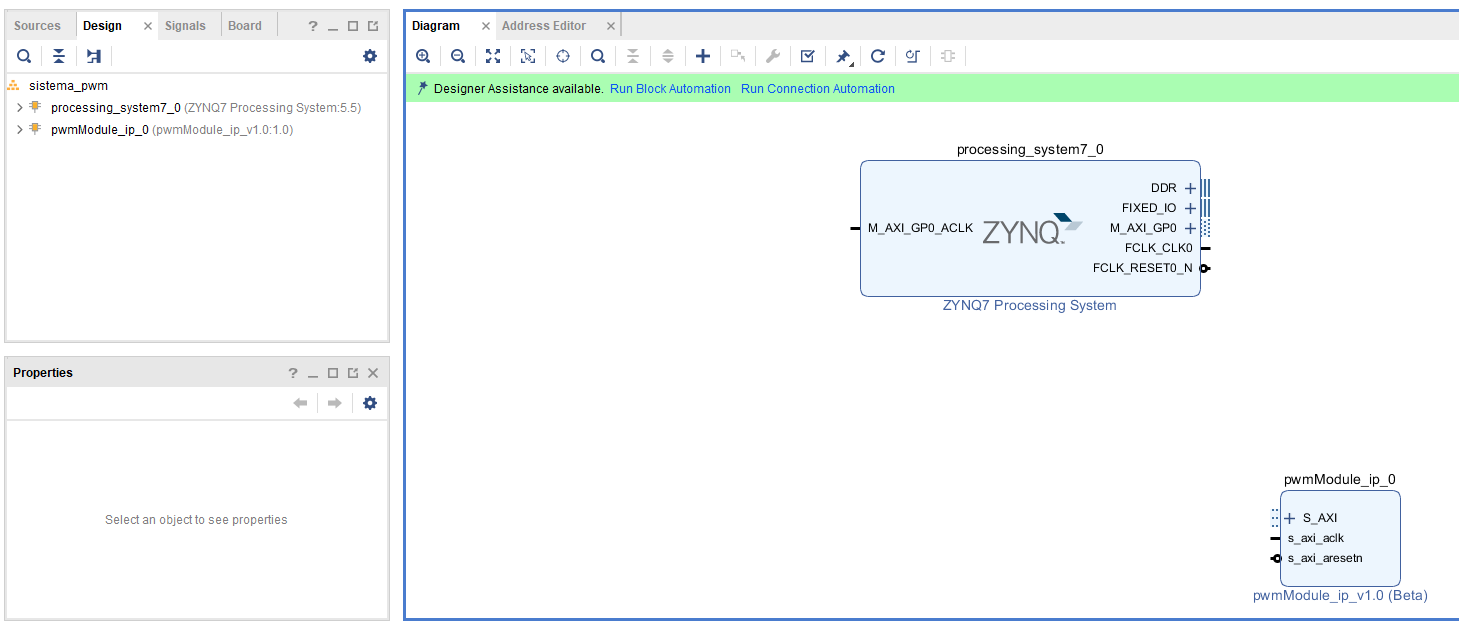
Le doy a “**+**” para agregar ip cord y lo busco en el listado



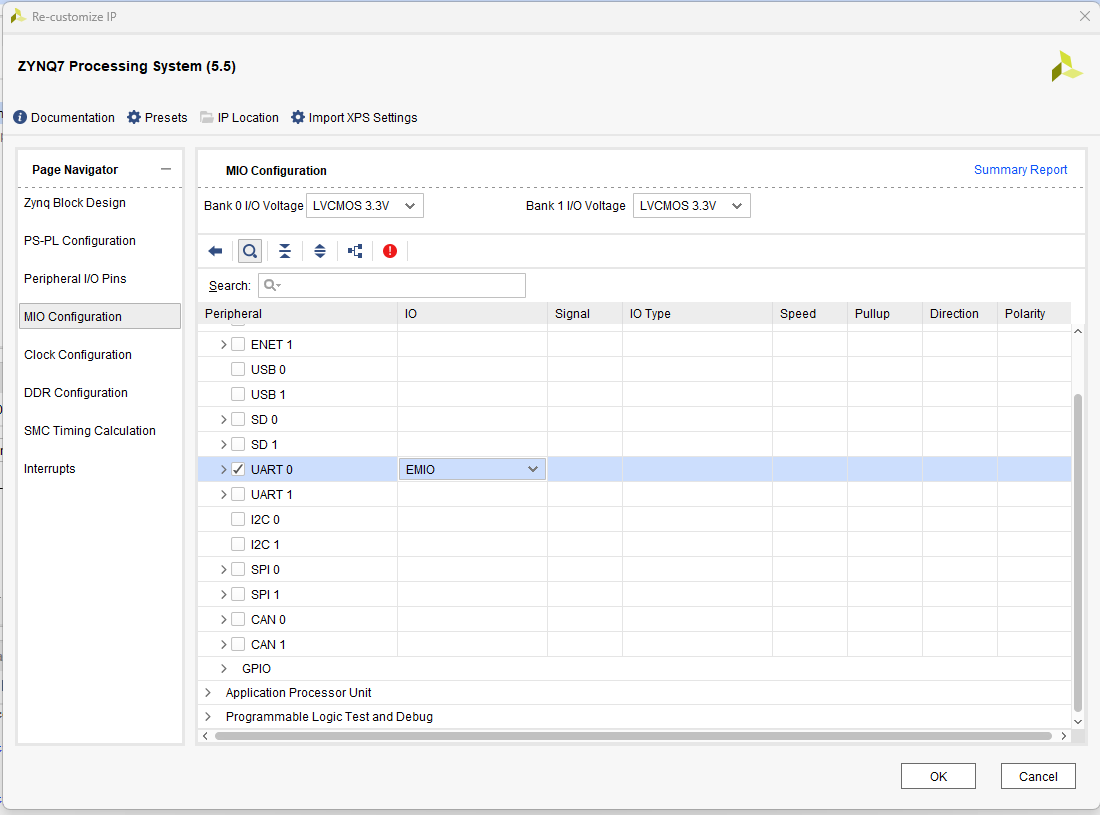
Enter



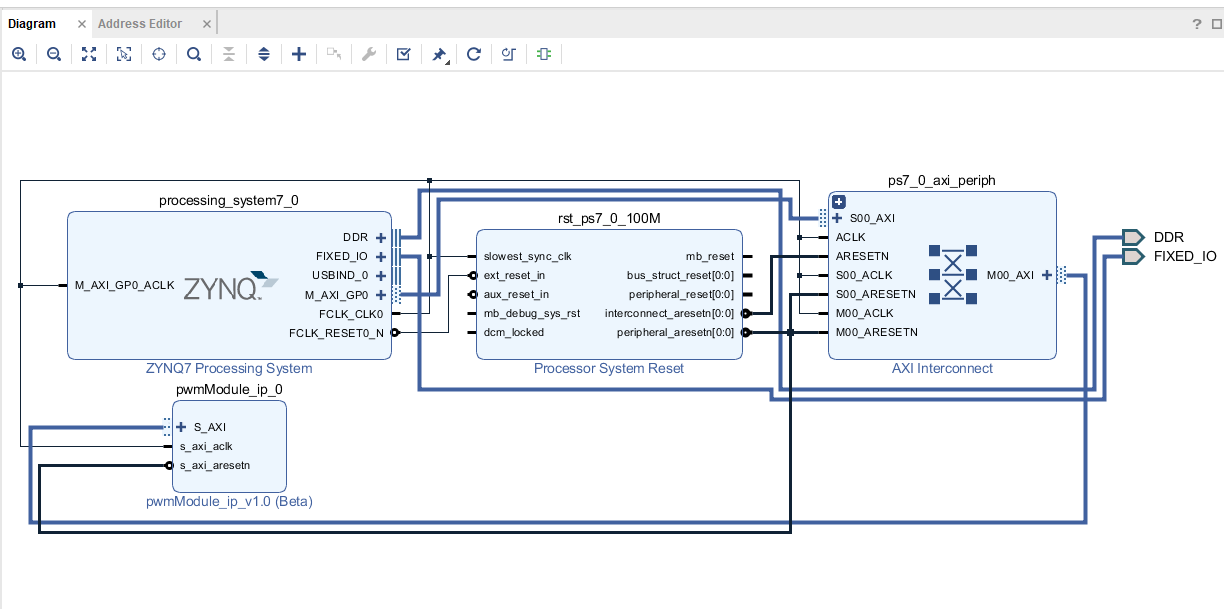
Agrego el ip cord del micro “zynq”

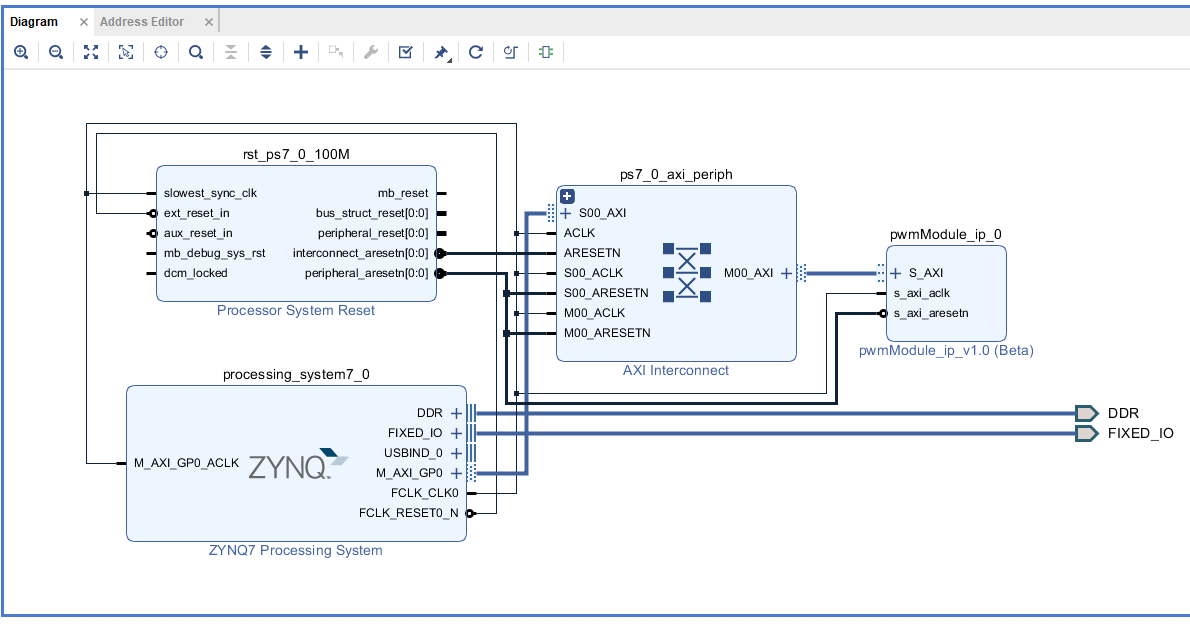


Des tildamos ENET 0, USB 0, SD 0, dejamos UART 0 y le damos ok

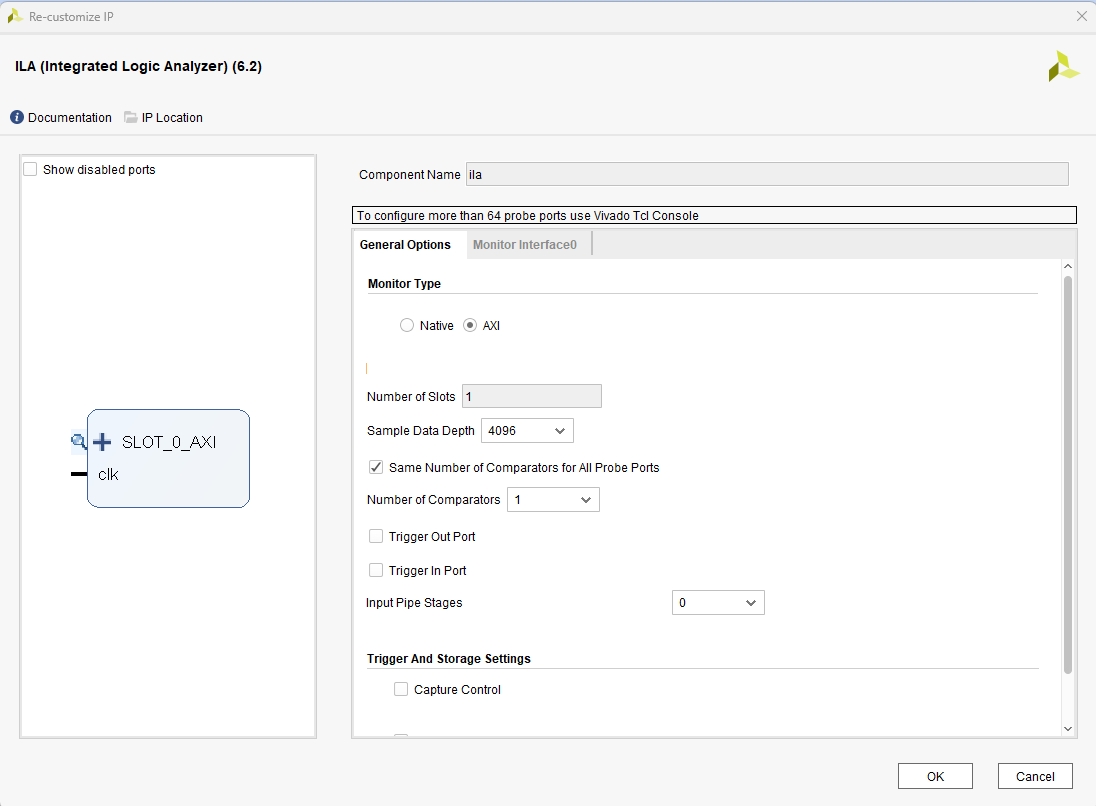


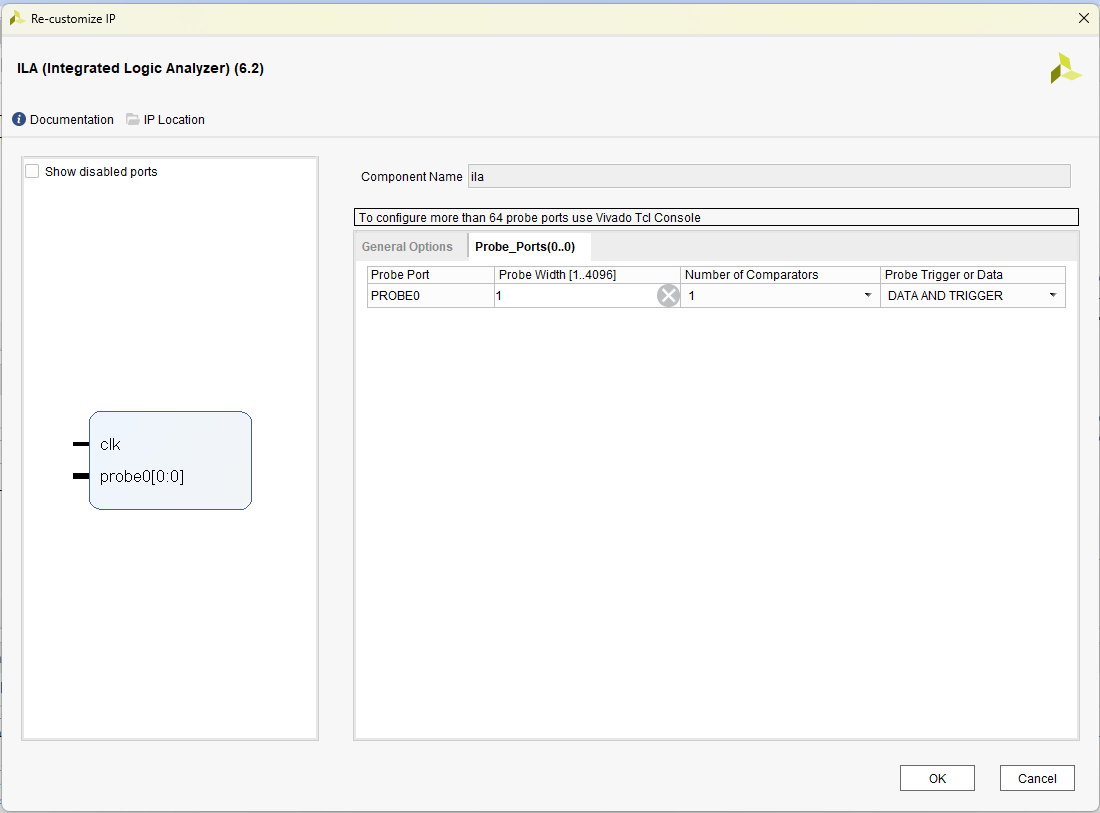
Le damos a “Run Block Automation” y luego a “Run connection automation”

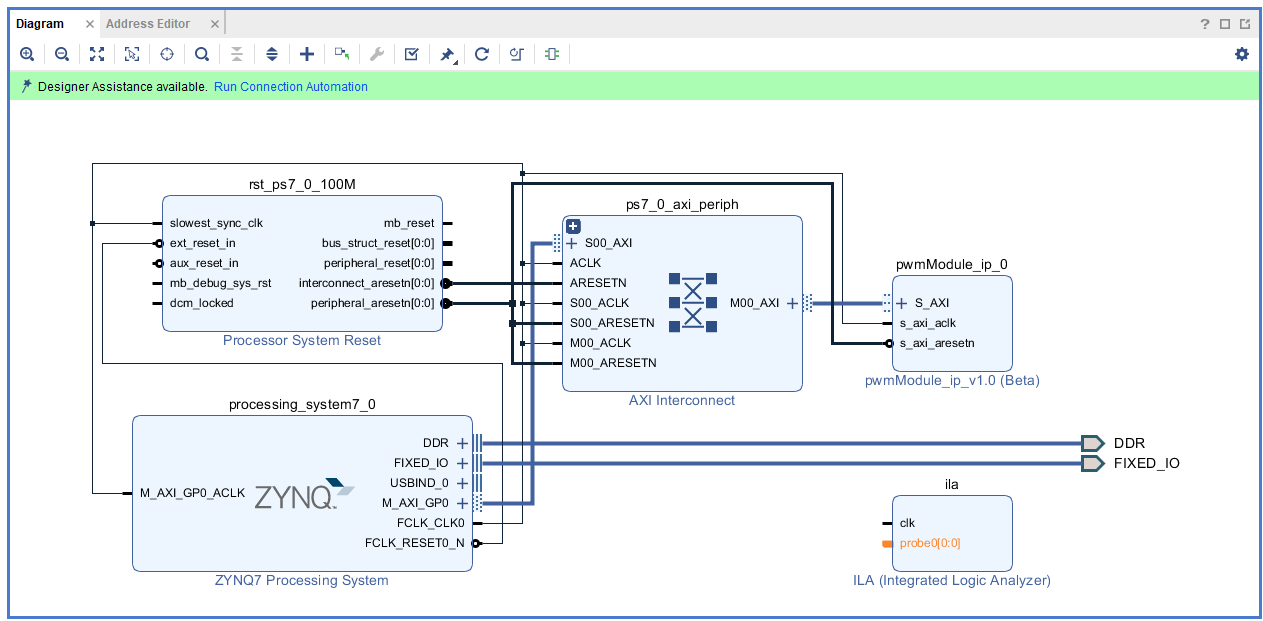
Regenremamos

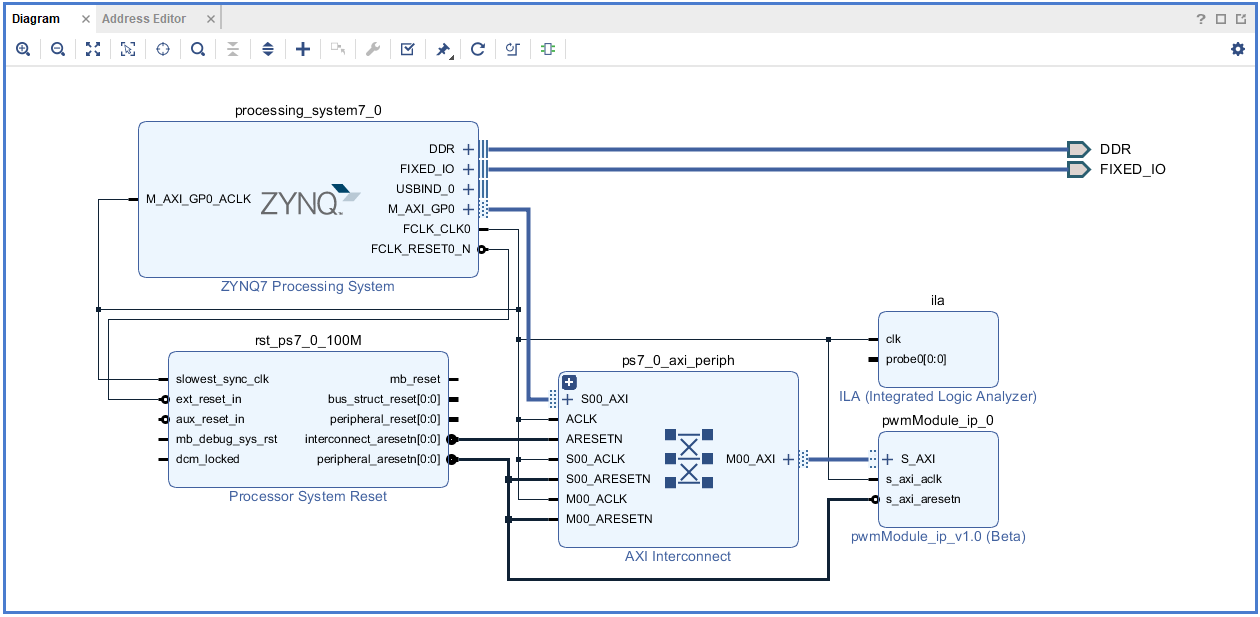


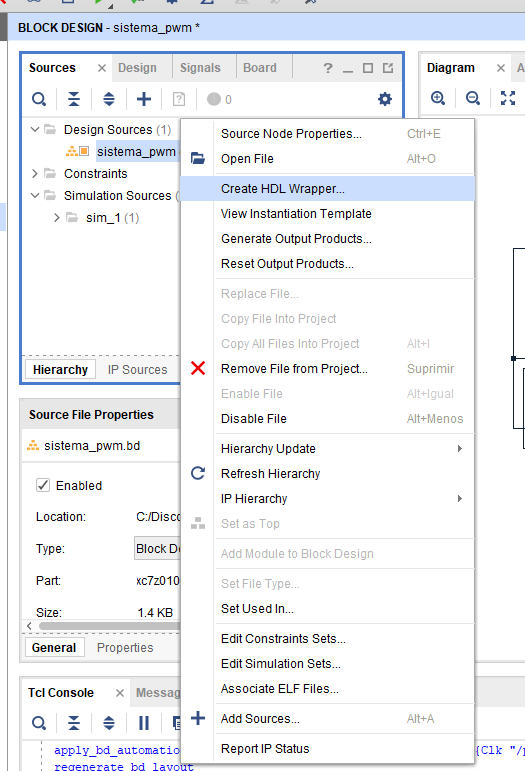
Ahora falta agregar nuestro ILA para ver la salida del PWM

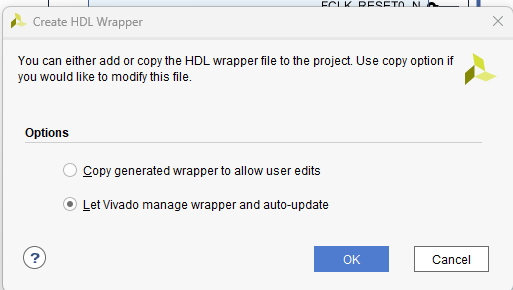




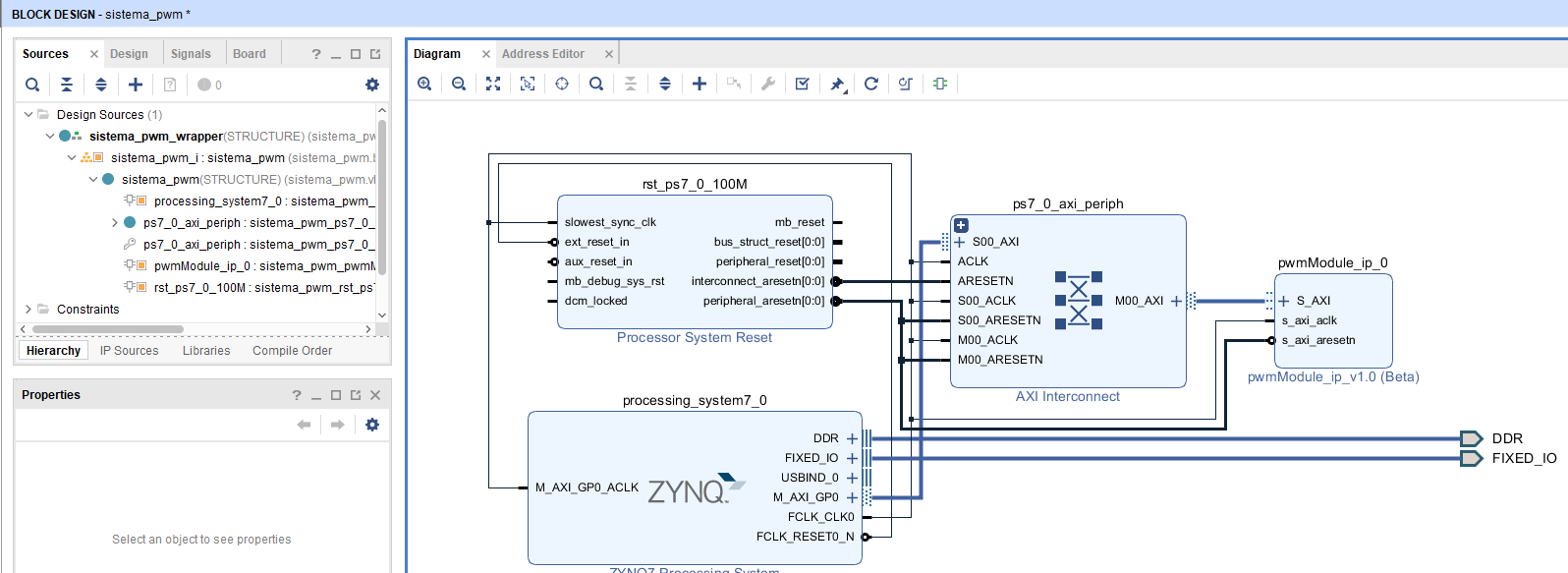








Me da error al crear el wrapper, relacionado con el ILA así que elimino el ILA del proyecto y repito los pasos de creación del wrapper y me da ok



Ahora agregar el ILA. (PENDIENTE)

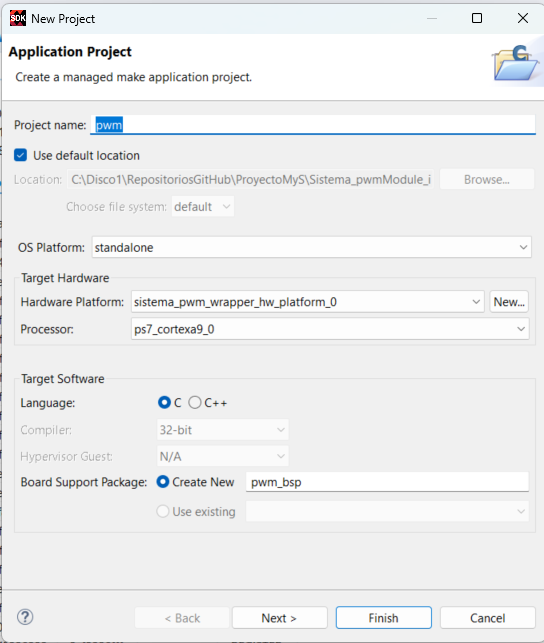
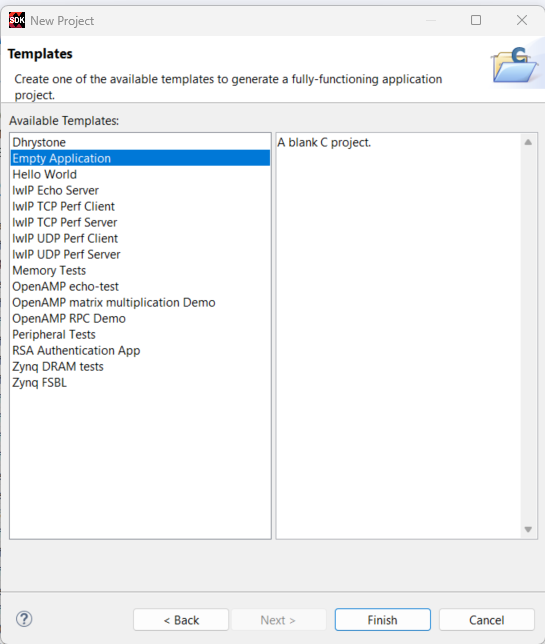
Copio sum.c a la carpeta de mi proyecto pwm y lo usaré de base para mi proyecto.

Genero el bistream 🡪 Se genero ok

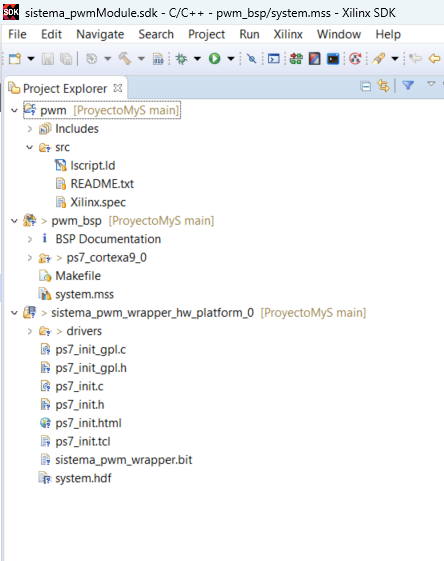
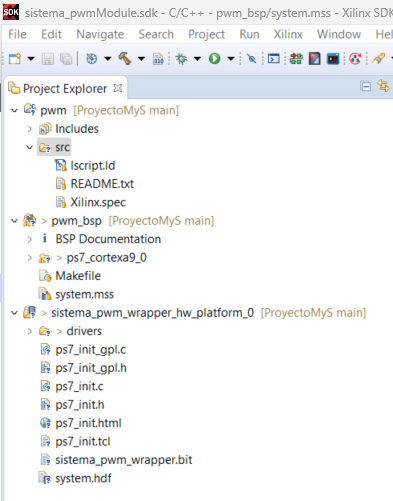
Hago export hardware

Lanzo el SDK

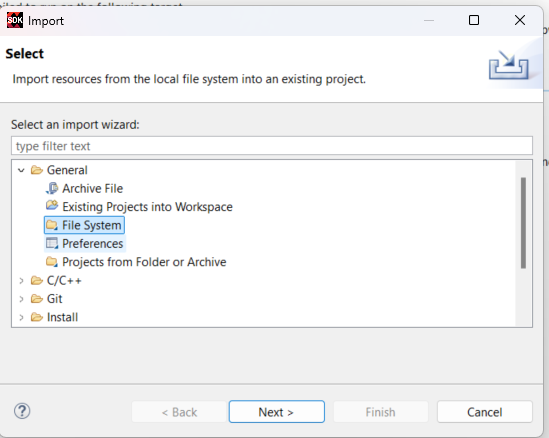
Seleccionar File ► New ►Application Project

Next ventana de la izquierda y seleccionar empty applicatio y finish

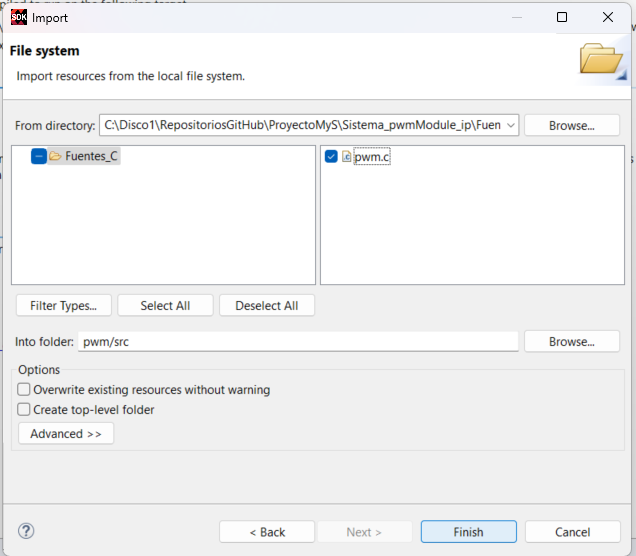
 

Import sobre pwm 🡪 src



Ingresamos en System file

Navegar hasta la ubicación de la carpeta donde se encuentra el archivo pwm.c y hacer click en OK.



Finish

