

Microarquitecturas y Softcores

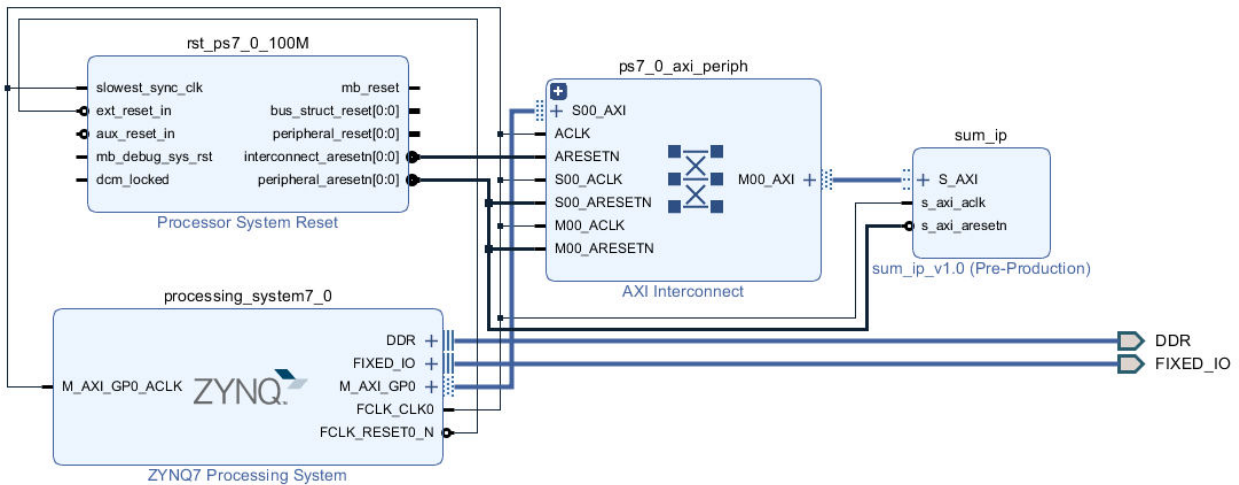
Tutorial

Agregado de una IP sumador al Sistema

Introducción

Este tutorial lo guiará a través del proceso de crear y agregar un periférico que realiza la operación de suma (con operandos de 32 bits) al sistema de procesamiento usando IP Packager de Vivado. La interfaz a utilizar será AXI4 Lite.

La siguiente imagen muestra el esquema general del sistema a implementar.



Creación de la IP

Paso 1

1-1. Usar el template de periférico esclavo provisto axi_lite y el código fuente del custom IP para crear un custom IP.

1-1-1. Abrir Vivado seleccionando **Start ► Xilinx Design Tools ► Vivado 2018.1**

1-1-2. Hacer click sobre **Manage IP** y seleccionar New IP Location y clicar **Next** en la ventana **New IP Location**.

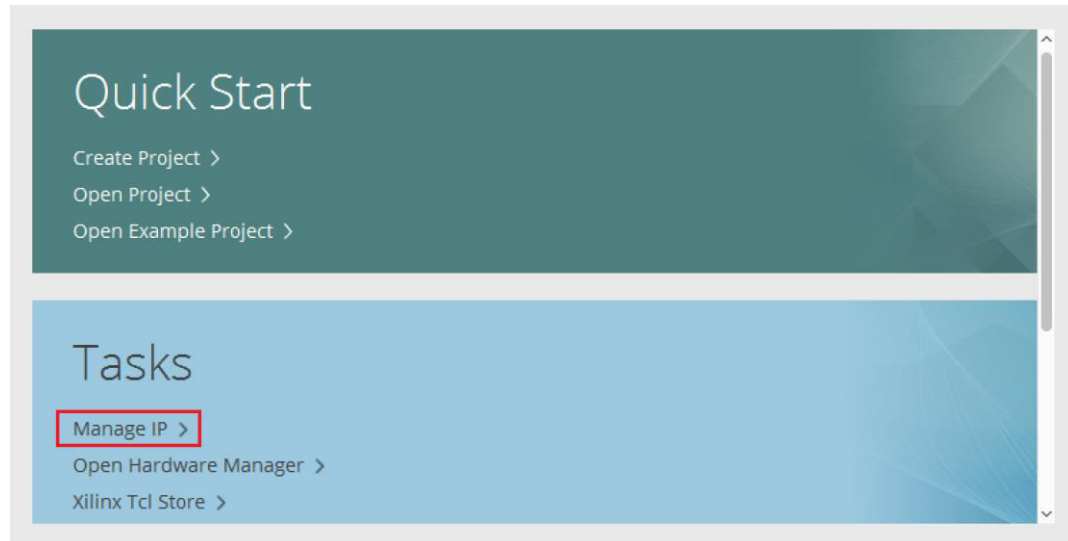


Figura 1. Manage IP

1-1-3. Seleccionar la placa correspondiente, **VHDL** como Target Language, y **Mixed** como Simulator language. Para el campo IP Location elegir una ubicación como por ejemplo el lugar donde se encuentran todas las prácticas y crear una carpeta sum_ip. Presionar **Finish**. Si el directorio no existe se solicitará la creación. Aceptar.

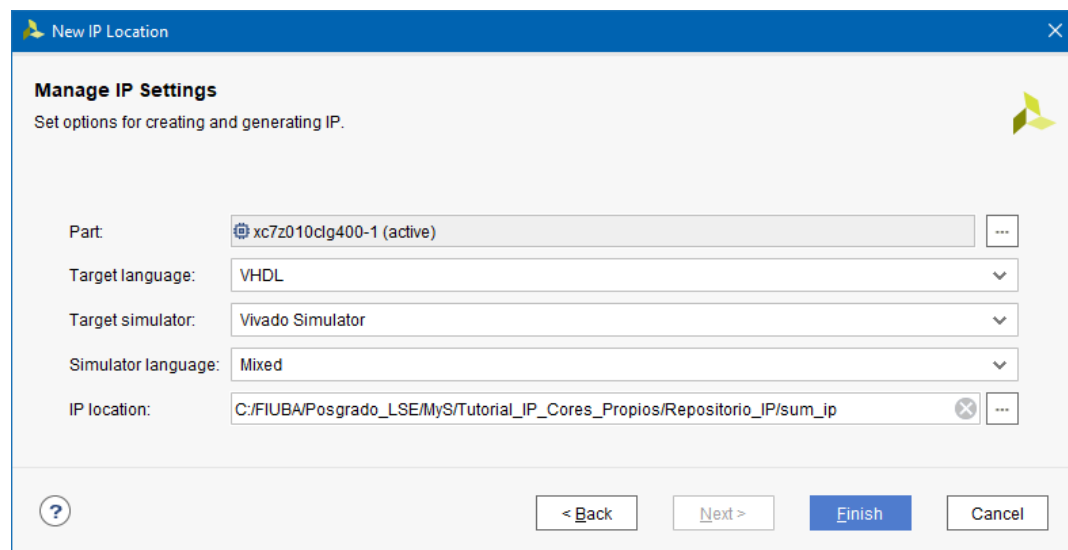


Figura 2. Configuraciones varias de la IP

1-2. Run the Create and Package IP Wizard

1-2-1. Seleccionar **Tools ► Create and Package New IP ...**

1-2-2. En la ventana presionar **Next**

1-2-3. Seleccionar *Create a new AXI4 peripheral*, y preionar **Next**

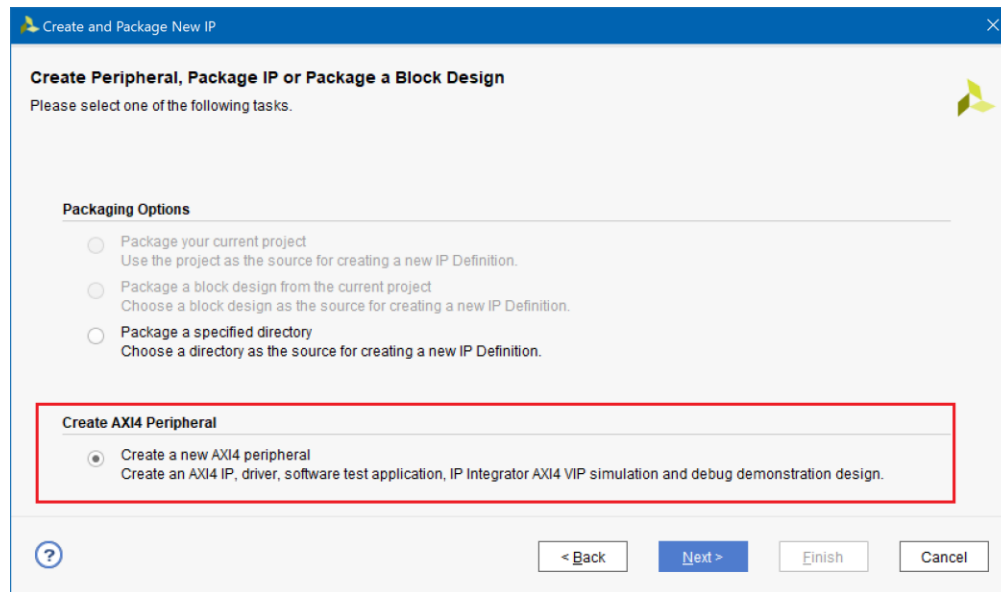
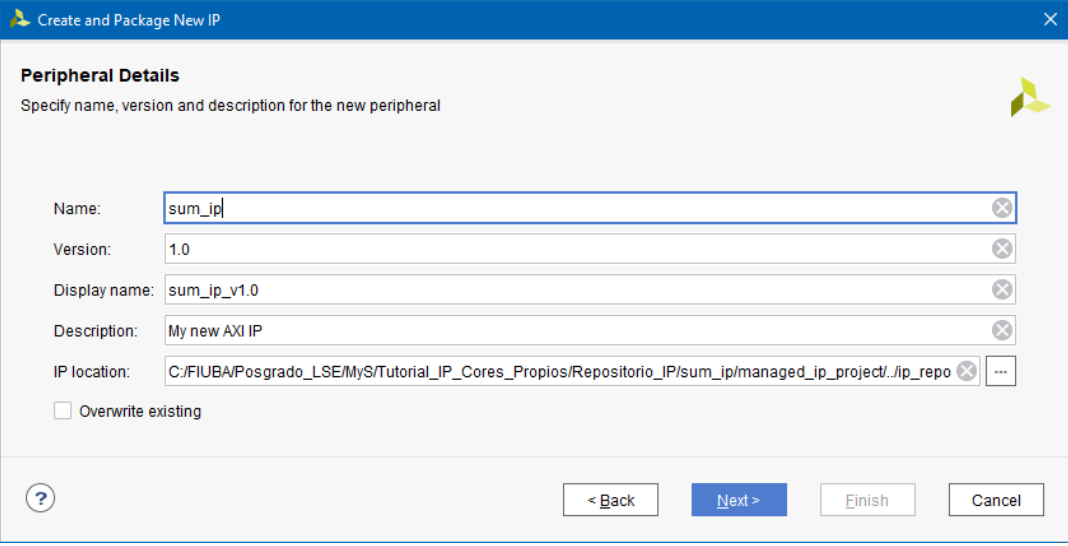


Figura 3. Selección de un periférico con conexión al bus AXI

1-2-4. Completar los detalles para la IP

Name: **sum_ip**

Display Name: **sum_ip_v1_0**



The screenshot shows the 'Create and Package New IP' dialog box with the 'Peripheral Details' tab selected. The dialog has a blue header bar with the title and a close button. Below the header, there's a subtitle 'Specify name, version and description for the new peripheral'. The main area contains several input fields: 'Name' (sum_ip), 'Version' (1.0), 'Display name' (sum_ip_v1.0), 'Description' (My new AXI IP), and 'IP location' (C:/FIUBA/Posgrado_LSE/MyS/Tutorial_IP_Cores_Propios/Repositorio_IP/sum_ip/managed_ip_project/.ip_repo). There's also an 'Overwrite existing' checkbox which is unchecked. At the bottom, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Create and Package New IP

Peripheral Details
Specify name, version and description for the new peripheral

Name: sum_ip

Version: 1.0

Display name: sum_ip_v1.0

Description: My new AXI IP

IP location: C:/FIUBA/Posgrado_LSE/MyS/Tutorial_IP_Cores_Propios/Repositorio_IP/sum_ip/managed_ip_project/.ip_repo

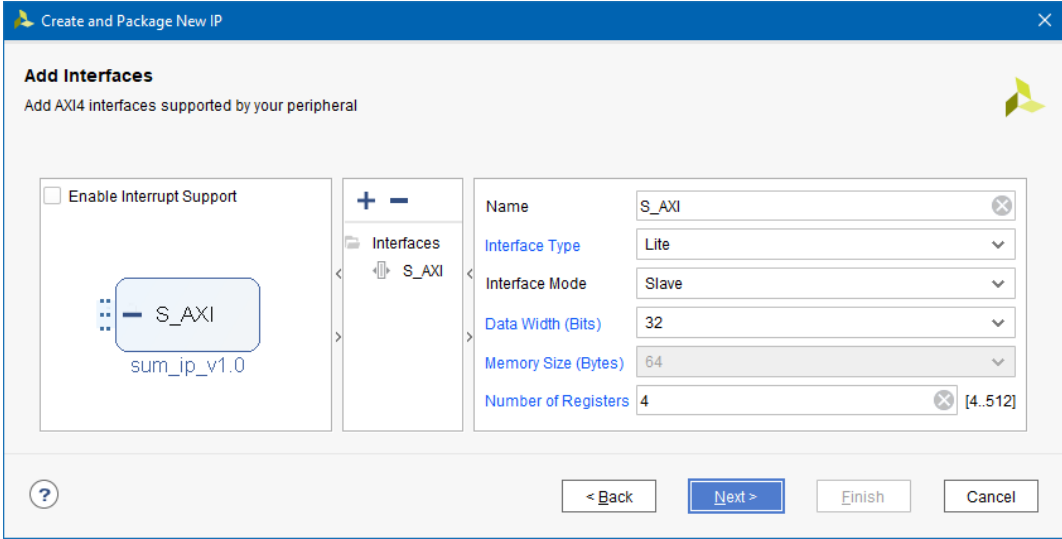
☐ Overwrite existing

< Back Next > Finish Cancel

Figure 4. Actualización de los datos del periférico

1-2-5. Presionar Next

1-2-6. Cambiar el nombre de la interfaz a S_AXI. Dejar el resto de las configuraciones por defecto y hacer click en Next.



The screenshot shows the 'Create and Package New IP' dialog box with the 'Add Interfaces' tab selected. The dialog has a blue header bar with the title and a close button. Below the header, there's a subtitle 'Add AXI4 interfaces supported by your peripheral'. The main area is divided into three sections. On the left, there's a checkbox 'Enable Interrupt Support' which is unchecked. In the center, there's a tree view showing 'Interfaces' with 'S_AXI' selected. On the right, there's a configuration table for the selected interface. The table has the following values: Name (S_AXI), Interface Type (Lite), Interface Mode (Slave), Data Width (Bits) (32), Memory Size (Bytes) (64), and Number of Registers (4). At the bottom, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Create and Package New IP

Add Interfaces
Add AXI4 interfaces supported by your peripheral

☐ Enable Interrupt Support

Interfaces

- S_AXI

Name	S_AXI
Interface Type	Lite
Interface Mode	Slave
Data Width (Bits)	32
Memory Size (Bytes)	64
Number of Registers	4 [4..512]

< Back Next > Finish Cancel

Figure 5. Renombrando de la interfaz AXI

1-2-7. Seleccionar Edit IP y hacer click en Finish (se abrirá un Nuevo proyecto Vivado)

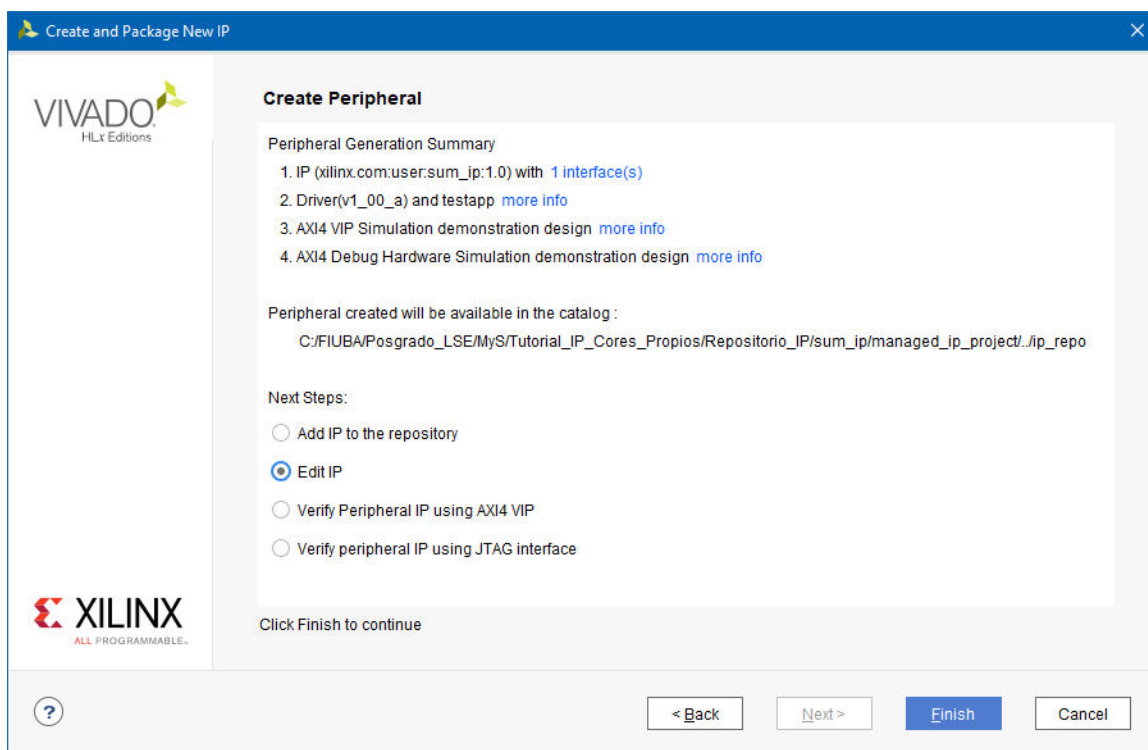


Figure 6. Selección para editar la IP

1-3. Inclusión de nuestra IP en el código

1-3-1. En el panel sources hacer doble click sobre el archivo **sum_ip_v1_0.vhd**.

Este archivo contiene el código HDL para la interfaz seleccionada arriba. El archivo top level contiene un módulo que implementa la lógica de interfaz AXI, y un ejemplo de diseño para escribir hacia y leer desde un número de registros especificados arriba. Esta plantilla puede ser usada como base para crear la IP personalizada..

1-3-2. En el panel sources expandir **sum_ip_v1_0** y abrir **sum_ip_v1_0_S_AXI.vhd**

1-3-3. Insertar el siguiente código en la parte declarativa de la arquitectura

```
component sum_user_logic is
  port(
    clk_i: in std_logic;
    opA_i: in std_logic_vector(31 downto 0);
    opB_i: in std_logic_vector(31 downto 0);
    res_o: out std_logic_vector(31 downto 0)
  );
end component;
```

Figura 7. Declaración del componente sumador

- 1-3-4.** Asimismo agregue la siguiente declaración de señal que será usada para conectar la salida del core sumador.

```
signal salSum: std_logic_vector(31 downto 0);
```

- 1-3-5.** En la parte descriptiva de la arquitectura realizar la instancia del componente sumador (puede guiarse con las siguientes palabras "-- Add user logic here"

```
-- Add user logic here
inst_sum: sum_user_logic
  port map(
    clk_i => S_AXI_ACLK,
    opA_i => slv_reg0,
    opB_i => slv_reg1,
    res_o => salSum
  );

-- User logic ends

end arch_imp;
```

Figura 8. Instancia del componente sumador

- 1-3-6.** Modificar slv_reg2 por salSum en el process encargado de la lectura de los registros

```
-- Implement memory mapped register select and read logic generation
-- Slave register read enable is asserted when valid address is available
-- and the slave is ready to accept the read address.
slv_reg_rden <= axi_arready and S_AXI_ARVALID and (not axi_rvalid) ;

process (slv_reg0, slv_reg1, salSum, slv_reg3, axi_araddr, S_AXI_ARESETN, slv_reg_rden)
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
  -- Address decoding for reading registers
  loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
  case loc_addr is
    when b"00" =>
      reg_data_out <= slv_reg0;
    when b"01" =>
      reg_data_out <= slv_reg1;
    when b"10" =>
      reg_data_out <= salSum;
    when b"11" =>
      reg_data_out <= slv_reg3;
    when others =>
      reg_data_out <= (others => '0');
  end case;
end process;
```

Figura 9. Modificación del código para poder leer la salida del sumador

- 1-3-7.** Guardar el archivo presionando Ctrl + S.

- 1-3-8.** Hacer click sobre *Add Sources* en el panel Flow Navigator, seleccionar *Add or Create Design Sources*, hacer click en *Next*, luego click en **Add Files...**, navegar donde se encuentren los archivos auxiliares, seleccionar el archivo **sum_user_logic.vhd** y hacer click en **OK**, y luego en **Finish** para agregarlo.
- 1-3-9.** Hacer click sobre **Run Synthesis** y guardar si se lo solicita. (Esto es para verificar que el diseño sintetiza correctamente antes de empaquetar la IP. Si en este paso uno estuviera realizando un diseño propio se debería realizar una simulación para verificar la funcionalidad antes de proceder a los siguientes pasos).
- 1-3-10.** Verifique la pestaña de Mensajes buscando errores para corregirlos, si fuera necesario, antes de seguir adelante. Cuando la Síntesis se complete satisfactoriamente haga click en **Cancel**.

1-4. Empaquetar la IP (Package)

1-4-1. Hacer click sobre la pestaña **Package IP – sum_ip**

The screenshot shows the 'Package IP - sum_ip' window. The 'Packaging Steps' list on the left includes: Identification (checked), Compatibility (checked), File Groups, Customization Parameters (checked), Ports and Interfaces (checked), Addressing and Memory (checked), Customization GUI (checked), and Review and Package. The 'Identification' section on the right contains the following fields:

- Vendor: xilinx.com
- Library: user
- Name: sum_ip
- Version: 1.0
- Display name: sum_ip_v1.0
- Description: My new AXI IP
- Vendor display name: (empty)
- Company url: (empty)
- Root directory: c:/FIUBA/Posgrado_LSE/MyS/Tutorial_IP_Cores_Propios/Repositorio_IP/sum_ip/ip_repo/sum_ip_1.0
- Xml file name: c:/FIUBA/Posgrado_LSE/MyS/Tutorial_IP_Cores_Propios/Repositorio_IP/sum_ip/ip_repo/sum_ip_1.0/component.xml

The 'Categories' section shows a list with the following items:

- AXI_Peripheral

Figure 10. Package IP

1-4-2. Hacer click sobre **File Groups** y luego sobre *Merge changes from File Groups Wizard*

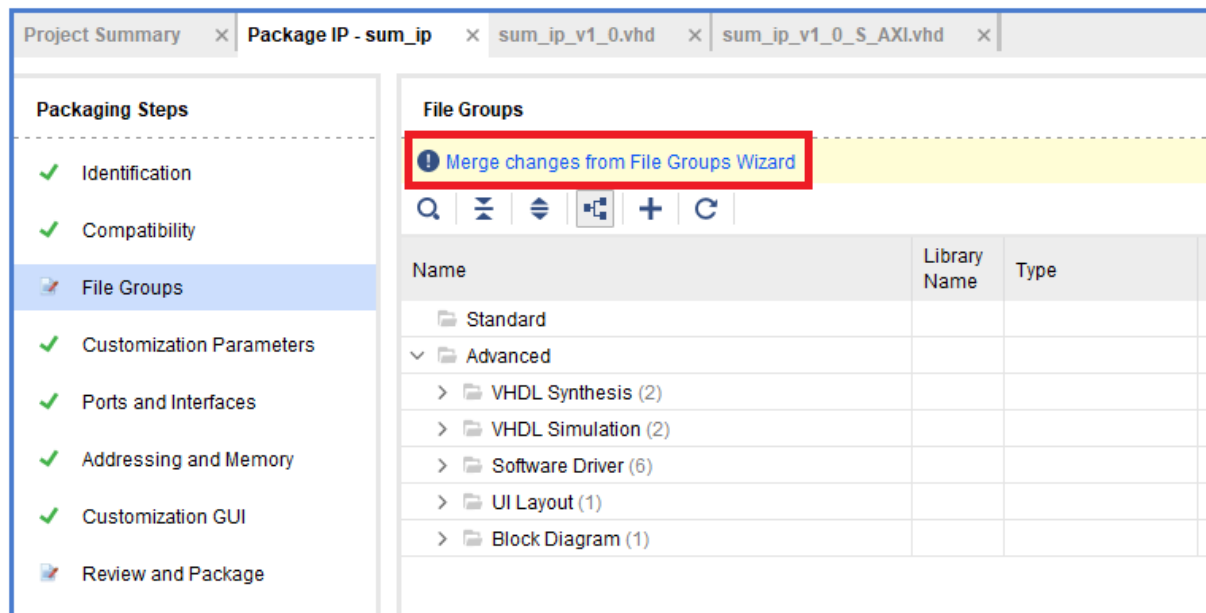


Figure 11. Actualización del file group

Esto se realiza para actualizar el IP Packager con los cambios realizados al IP y al archivo *sum_user_logic.vhd* que fue agregado al proyecto.

1-4-3. Expandir *VHDL Synthesis* y notar que ha sido incluido el **sum_user_logic.vhd**

1-4-4. Seleccionar **Review and Package**, y notar que el path donde la IP será creada.

1-4-5. Hacer click **Re-Package IP**. Luego hacer click en **Yes** (el proyecto se cerrará al finalizar).

1-4-6. En la ventana original de Vivado hacer lick en **File ► Close Project**

Modificar la configuración del proyecto

Paso 2

2-1. Abrir o crear el proyecto donde se quiere agregar la IP. Configurar el proyecto para que apunte al repositorio IP creado.

2-1-1. Teniendo en cuenta los laboratorios ya realizados crear un sistema base con procesador dejando sólo la UART 0.

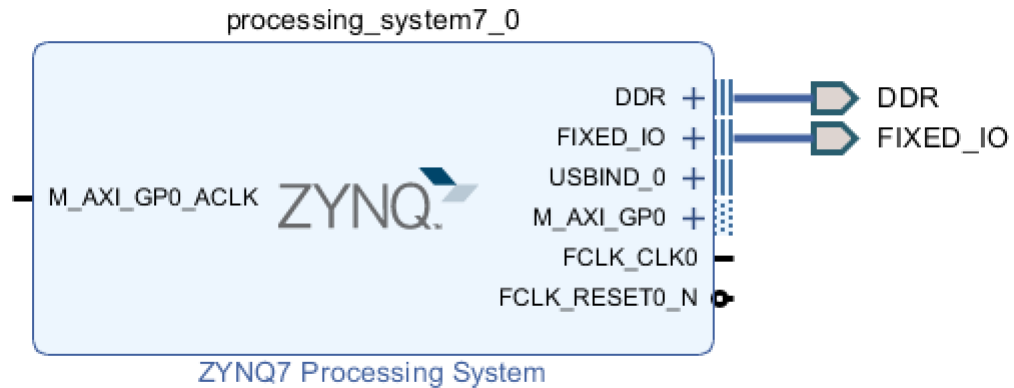


Figura 12. Vista del sistema base con procesador

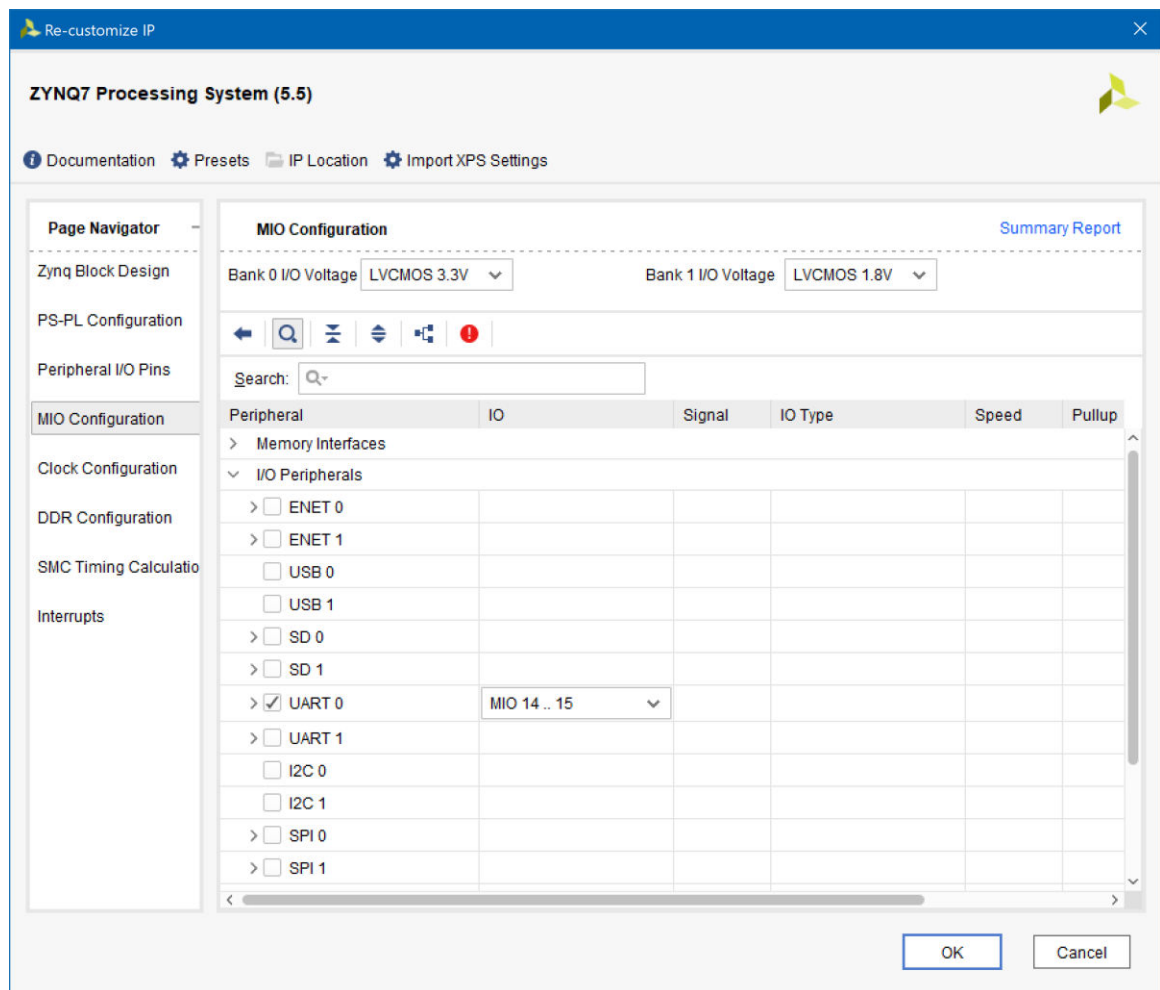


Figura 13. Configuración del PS

- 2-1-2. Hacer click en Settings en el panel Flow Navigator.
- 2-1-3. Seleccionar **IP ► Repository** en el panel izquierdo de la ventana de configuración del proyecto.
- 2-1-4. Hacer click en el símbolo **+** y navegar hasta el directorio `sum_ip` donde se estableció el repositorio. Aparecerá un mensaje indicando que se agregó un repositorio al proyecto. Presionar **OK**.

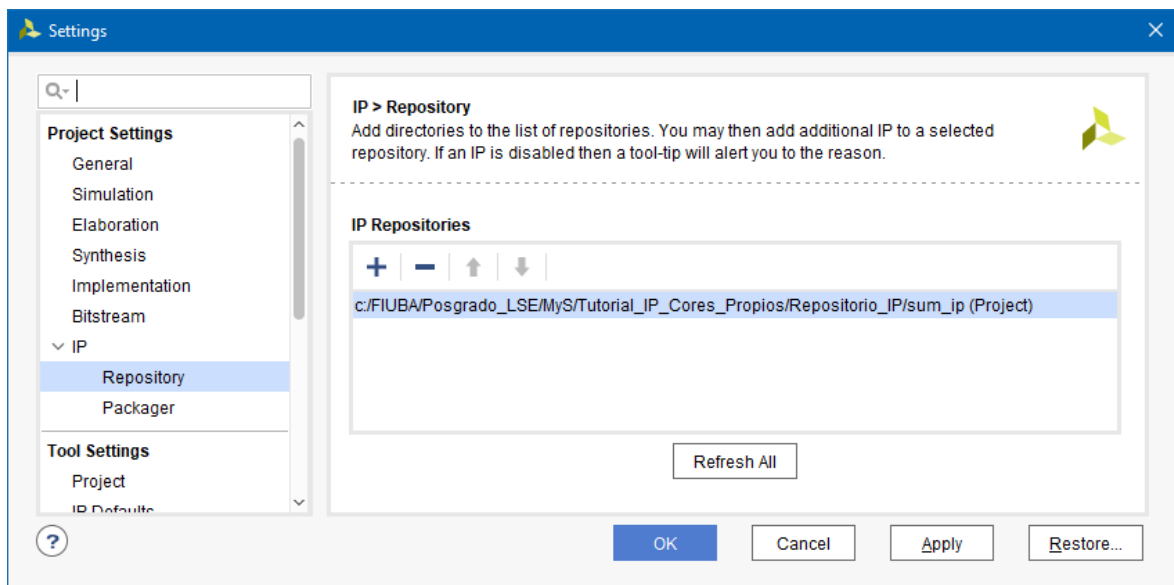


Figura 14. Ubicación del repositorio de IP

2-1-5. Hacer click en **OK**.

Agregar la IP personalizada

Paso 3

3-1. Agregar sum_ip al diseño y conectarla al AXI4 Lite interconnect en el IPI.

- 3-1-1. Hacer click sobre **Open Block Design** debajo de **IP Integrator** en el panel Flow Navigator (en caso de ser necesario).
- 3-1-2. En la ventana Diagram hacer botón derecho y seleccionar **Add IP ...** y buscar sum_ip_v1.0 en el catálogo colocando sum en el campo de búsqueda.
- 3-1-3. Hacer doble click sobre sum_ip_v1.0 para agregarla al diseño.
- 3-1-4. Seleccionar la ip y cambiar el nombre de instancia a sum_ip en **Block Properties**.
- 3-1-5. Hacer doble click sobre el bloque para abrir la configuración si es que se desea observar la configuración existente.

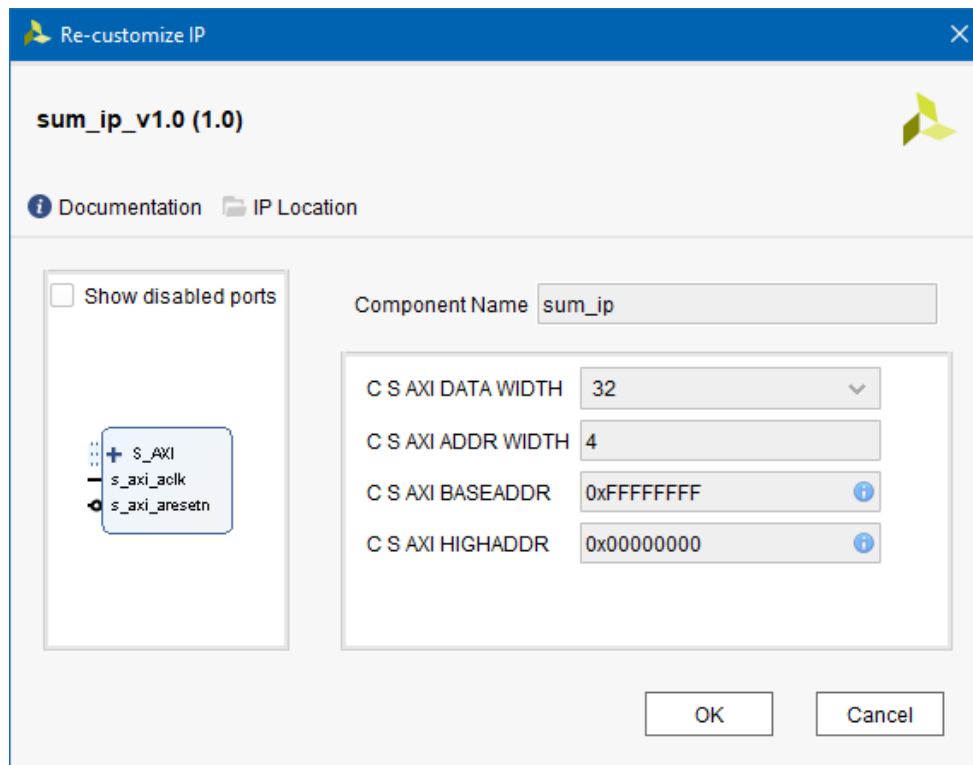


Figura 15. Configuración de la IP

- 3-1-6. Hacer click en **Run Connection Automation**, seleccionar /sum_ip/S_AXI y hacer click en **OK** para realizar la conexión automática desde el AXI Interconnect a la IP.

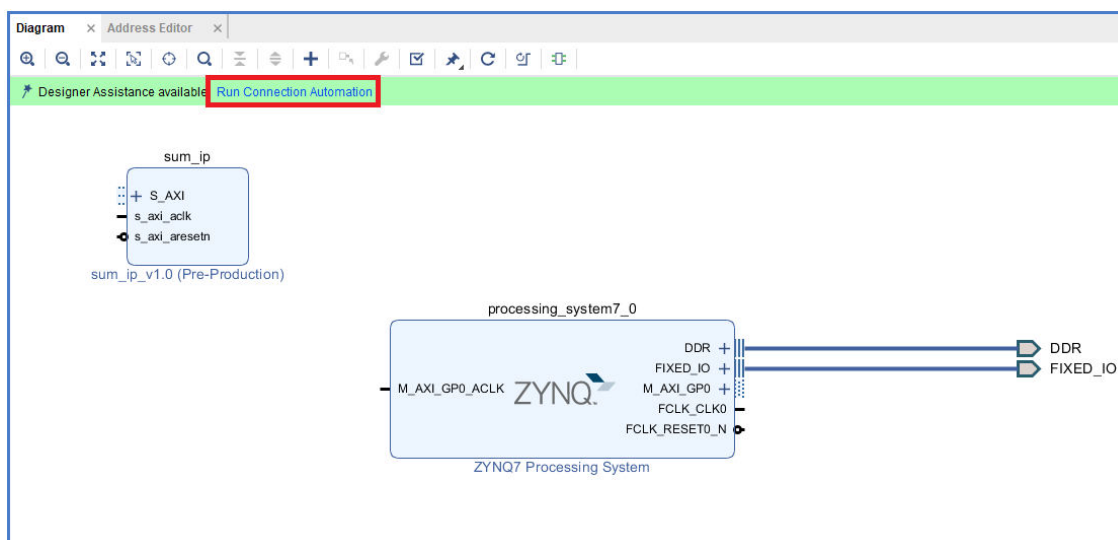


Figura 16. Selección de la conexión automática

3-1-7. Hacer click en el botón **Regenerate Layout** para redibujar el diagrama. Luego validar el diseño ☒ (F6). Luego de estos pasos se debería tener el siguiente sistema:

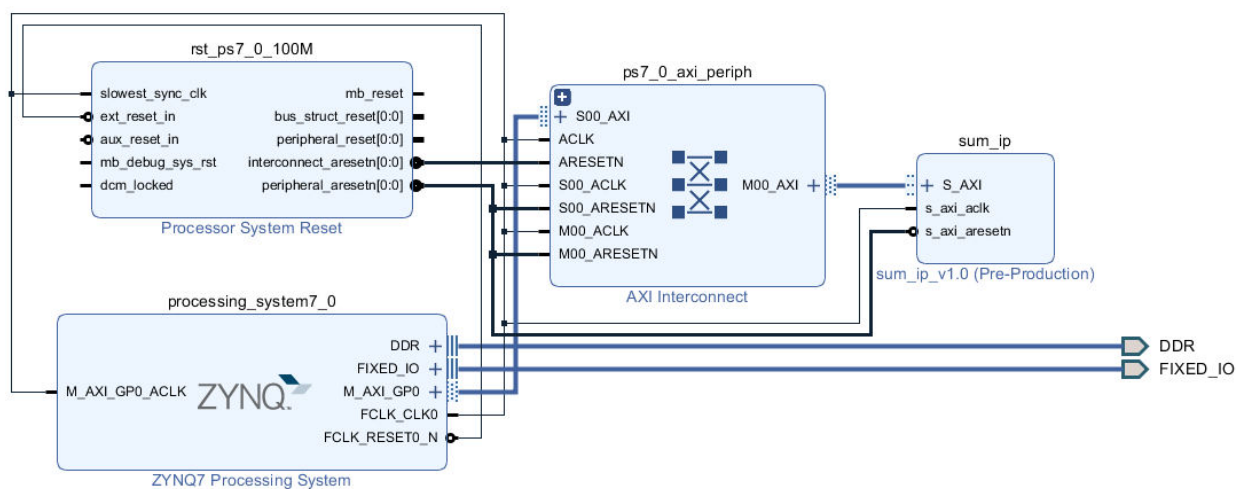


Figura 17. Sistemade procesamiento con el agregado de la IP propia

Generar el Top-Level y Export el diseño de hardware al SDK Paso 4

4-1. Generar las salidas de IP Integrator, el HDL top-level, y ejecutar el SDK exportando el hardware.

4-1-1. En el panel *Sources*, hacer click-derecho sobre *system.bd*, y seleccionar **Generate Output Products ...** y presionar **Generate** para generar los archivos de implementación, simulación y síntesis para el diseño (también puede hacer click sobre **Generate Block Design** en el panel Flow Navigator para hacer lo mismo). Presionar **OK**.

4-1-2. Hacer click-derecho nuevamente sobre *system.bd*, y seleccionar **Create HDL Wrapper...** para generar el modelo top-level de VHDL. Dejar seleccionada la opción *Let Vivado manager wrapper and auto-update*, y presionar **OK**.

Se creará el archivo *system_wrapper.vhd* y se agregará al proyecto. Hacer doble-click sobre el archivo para ver el contenido en el panel auxiliar.

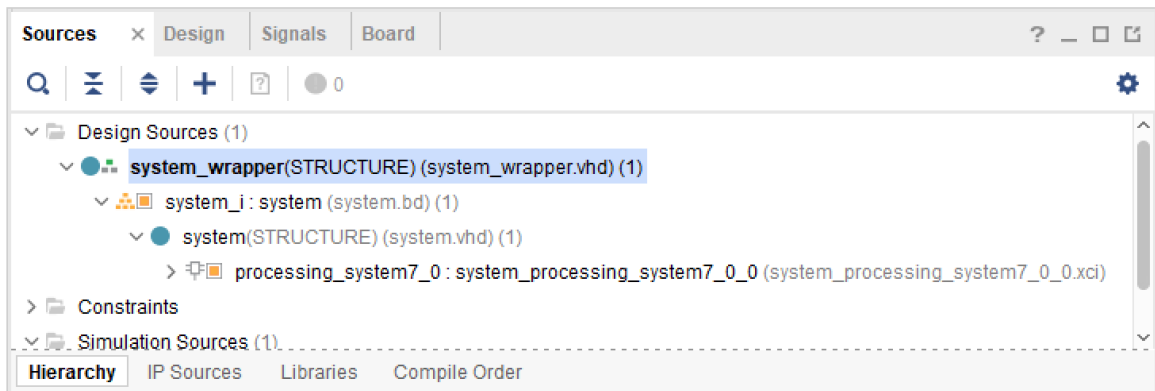



Figura 18. Archivo Wrapper HDL generado y agregado al proyecto

4-1-3. Notar que el archivo está configurado como el módulo principal (*Top module*) en el diseño, indicado por el icono .

4-1-4. Hacer click en **Generate Bitstream** y presionar **OK**. Hacer click en **Cancel** cuando se pregunte si se desea abrir el diseño implementado (*Open the Implemented Design*).

4-1-5. Seleccionar **File ► Export ► Export hardware** (marcar la opción “*Include bitstream*”) y presionar **OK**. (Guardar el proyecto si es requerido). En este momento se crea la carpeta *Nombre_del_proyecto.sdk*.

4-1-6. Seleccionar **File ► Launch SDK** dejando la configuración por defecto, y presionar **OK**.

El SDK debería estar abierto en este momento. Si sólo el panel de Bienvenida es visible, cierre o minimice este panel para ver el *Project Explorer* y el panel *Preview*. Debería haberse creado un proyecto de plataforma de hardware automáticamente, y el directorio *system_wrapper_hw_platform_0* debería existir en el panel *Project Explorer*.

4-1-7. Seleccionar **File ► New ► Application Project**.

- 4-1-8. Ingresar **sum** como nombre del proyecto, y para Board Support Package, elegir **Create New** sum_bsp (debería ser la única opción).

The screenshot shows the 'New Project' dialog box in the SDK. The dialog is titled 'New Project' and 'Application Project'. It contains the following fields and options:

- Project name:** sum
- ☒ **Use default location**
- Location:** C:\FIUBA\Posgrado_LSE\MyS\Tutorial_IP_Cores_Propios\Pro
- Choose file system:** default
- OS Platform:** standalone
- Target Hardware**
 - Hardware Platform:** system_wrapper_hw_platform_0
 - Processor:** ps7_cortexa9_0
- Target Software**
 - Language:** C (selected), C++
 - Compiler:** 32-bit
 - Hypervisor Guest:** N/A
 - Board Support Package:** Create New (selected), Use existing
 - sum_bsp** (text input field)

The 'Finish' button is highlighted in blue.

Figura 19. Creación del proyecto de aplicación

- 4-1-9. Hacer click en **Next**, y seleccionar **Empty Application** y hacer click en **Finish**.
- 4-1-10. Expandir sum en la vista de proyecto y hacer botón-derecho en la carpeta src y seleccionar **Import**. También se puede optar por escribir el código desde cero dentro del IDE.

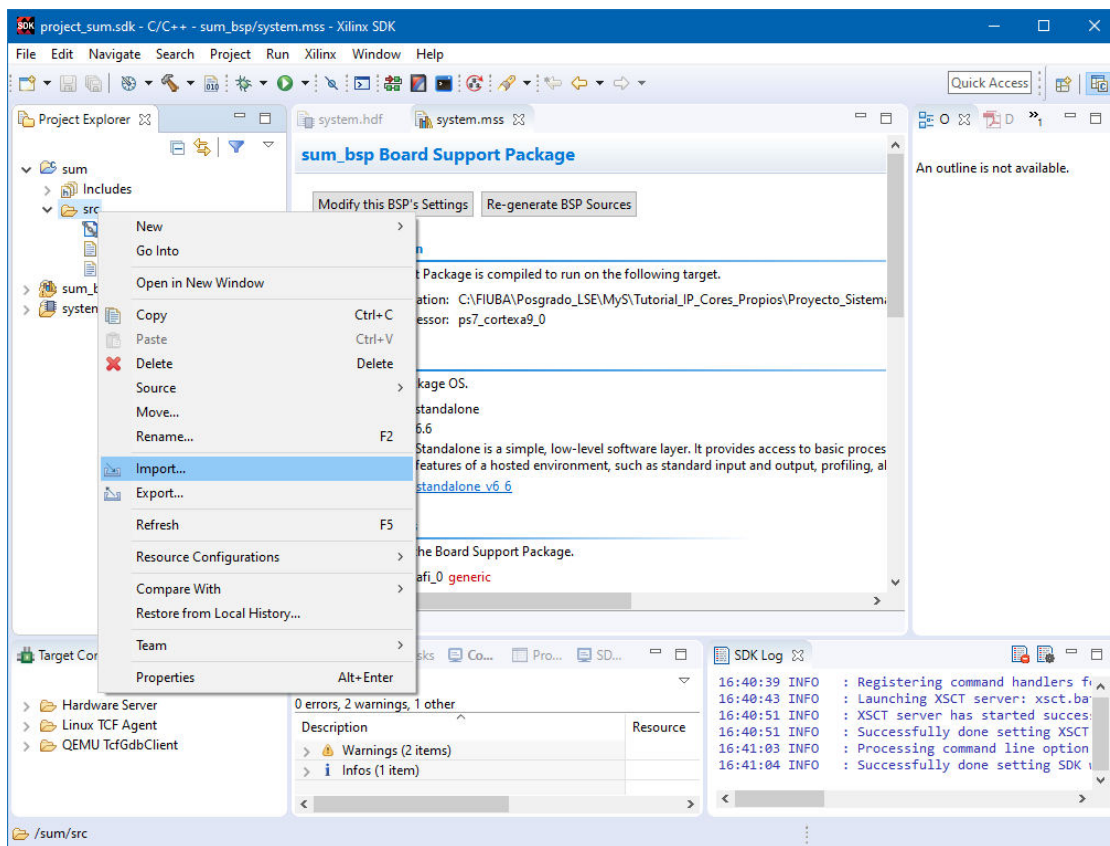


Figura 20. Importar el código C

4-1-11. Expandir la categoría **General** y hacer doble click **File System**.

4-1-12. Navegar hasta la ubicación de la carpeta donde se encuentra el archivo sum.c y hacer click en **OK**.

4-1-13. Seleccionar el archivo sum.c y hacer click en **Finish** para agregarlo al proyecto.

```
#include "xparameters.h"
#include "xil_io.h"
#include "sum_ip.h"

//=====

int main (void) {

    int res;
    int opA = 1256;
    int opB = 800;

    xil_printf("-- Inicio del programa para validar el uso de IP cores propios --\r\n");

    SUM_IP_mWriteReg(XPAR_SUM_IP_S_AXI_BASEADDR, SUM_IP_S_AXI_SLV_REG0_OFFSET, opA);
    SUM_IP_mWriteReg(XPAR_SUM_IP_S_AXI_BASEADDR, SUM_IP_S_AXI_SLV_REG1_OFFSET, opB);
    res = SUM_IP_mReadReg(XPAR_SUM_IP_S_AXI_BASEADDR, SUM_IP_S_AXI_SLV_REG2_OFFSET);

    xil_printf("Cuenta: %d + %d = %d\r\n", opA, opB, res);

}
```

Figura 21. Código C

Verificación en la placa

Paso 5

- 5-1. Conectar la placa con el cable micro-usb.**
Establecer una comunicación serie con alguna terminal (puede usar la terminal que brinda el SDK).
 - 5-1-1.** Asegurarse de que el cable micro-USB esté conectado entre la placa y la PC.
 - 5-1-2.** Configurar la terminal con 115200 como velocidad. Conectarse al puerto correspondiente.

- 5-2. Programar la FPGA seleccionando Xilinx ► Program FPGA y asignando el archivo system_wrapper.bit. Ejecutar la aplicación TestApp y verificar la funcionalidad.**
 - 5-2-1.** Seleccionar **Xilinx ► Program FPGA**.
 - 5-2-2.** Hacer click en el botón **Program** para programar la FPGA.
 - 5-2-3.** Seleccionar **sum** en el Project Explorer, hacer click derecho y seleccionar **Run As ► Launch on Hardware (GDB)** para descargar la aplicación, ejecutar ps7_init, y ejecutar sum.elf.

En la terminal se debería observar lo siguiente:

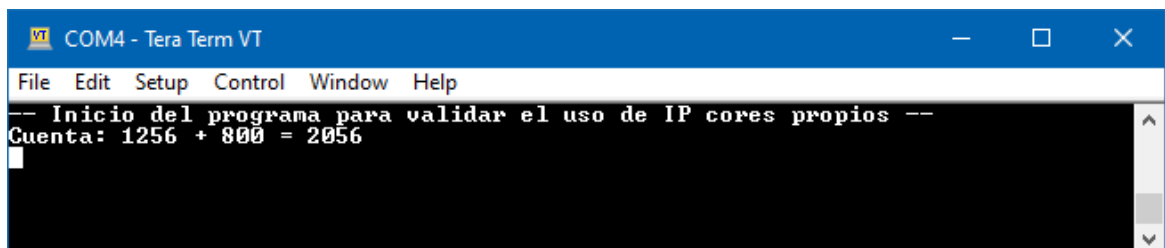


Figura 22. Resultado de la ejecución del código en el procesador