

# **Microarquitecturas y Softcores**

## **Práctica 2**

### **Agregando cores IP en la lógica programable (PL)**

(Adaptado del curso “Embedded System Design Flow” de Xilinx)

## Introducción

Esta práctica lo guiará a través del proceso de ampliar el sistema de procesamiento que ya creó en la práctica anterior agregando dos IPs GPIO (General Purpose Input/Output)

## Objetivos

Después de completar esta práctica será capaz de:

- Configurar el puerto GP Master del PS para conectarse a la IP en la PL
- Agregar IP adicional a un diseño de hardware
- Establecer algunas de las configuraciones del compilador

## Procedimiento

Esta práctica está separada en pasos que consisten en sentencias generales que proveen información sobre las instrucciones detalladas que le siguen. Siga estas instrucciones detalladas para avanzar dentro de esta práctica.

Esta práctica está compuesta por 6 pasos principales: abrir el proyecto en Vivado, agregar y configurar periféricos GPIO en el sistema usando IP Integrator, conectar los puertos externos, generar el bitstream y exportar a SDK, crear una aplicación TestApp en SDK, y, finalmente, verificar el diseño en hardware.

## Descripción del Diseño

El propósito de esta práctica es extender el diseño de hardware (Figura ) creado en la práctica 1

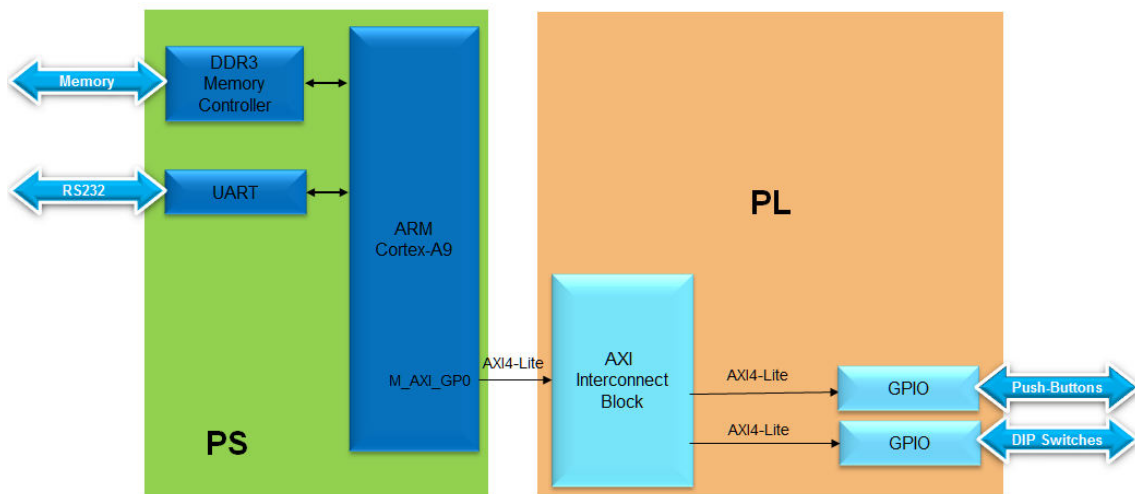
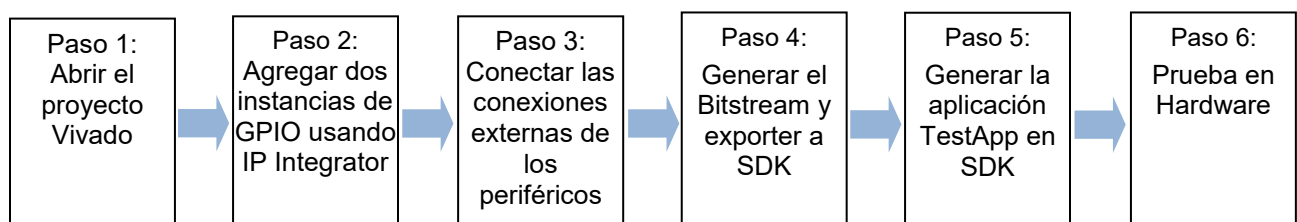


Figura 1. Extender el sistema de la práctica anterior

## Flujo General para esta práctica



## Abrir el Proyecto

## Paso 1

### 1-1. Abrir el proyecto anterior, y guardarlo como lab2. Abrir el Block Design.

- 1-1-1. Iniciar Vivado, y si es necesario, abrir el proyecto lab1.xpr que creó en la práctica anterior usando el link **Open Project** en la página Getting Started.

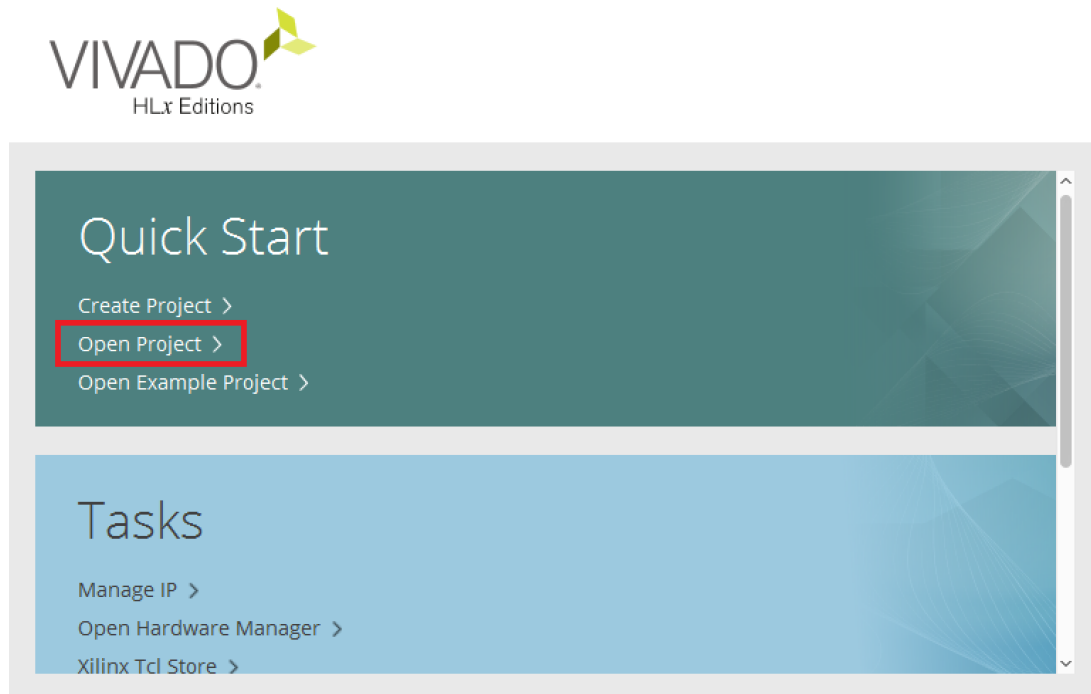


Figura 2. Abrir proyecto existente

- 1-1-2. Seleccionar **File ► Project ► Save As ...** para abrir el cuadro de diálogo *Save Project As*. Introducir **lab2** como nombre de proyecto. Asegúrese de que la opción *Create Project Subdirectory* esté marcada. Presionar **OK**.

Esto creará el directorio lab2 y guardará el proyecto y directorio asociado con el nombre lab2.

## Agregar dos Instancias de GPIO

## Paso 2

### 2-1. Habilitar la interfaz AXI\_M\_GP0, FCLK\_RESET0\_N, y FCLK\_CLK0 ports, Agregar dos instancias de un periférico GPIO del catálogo de IPs al sistema de procesador.

- 2-1-1. En el panel *Sources*, expandir *system\_wrapper*, y hacer doble-click sobre el archivo **system.bd(system\_i)** para invocar el IP Integrator. (El Block Design también puede ser abierto desde el Flow Navigator)
- 2-1-2. Doble click sobre el bloque Zynq en el diagrama para abrir la ventana de configuración del *Zynq*.
- 2-1-3. Seleccionar la página de menú **PS-PL Configuration** en la izquierda, o haga click sobre el bloque **32b GP AXI Master Ports** en la vista de diseño del bloque Zynq.

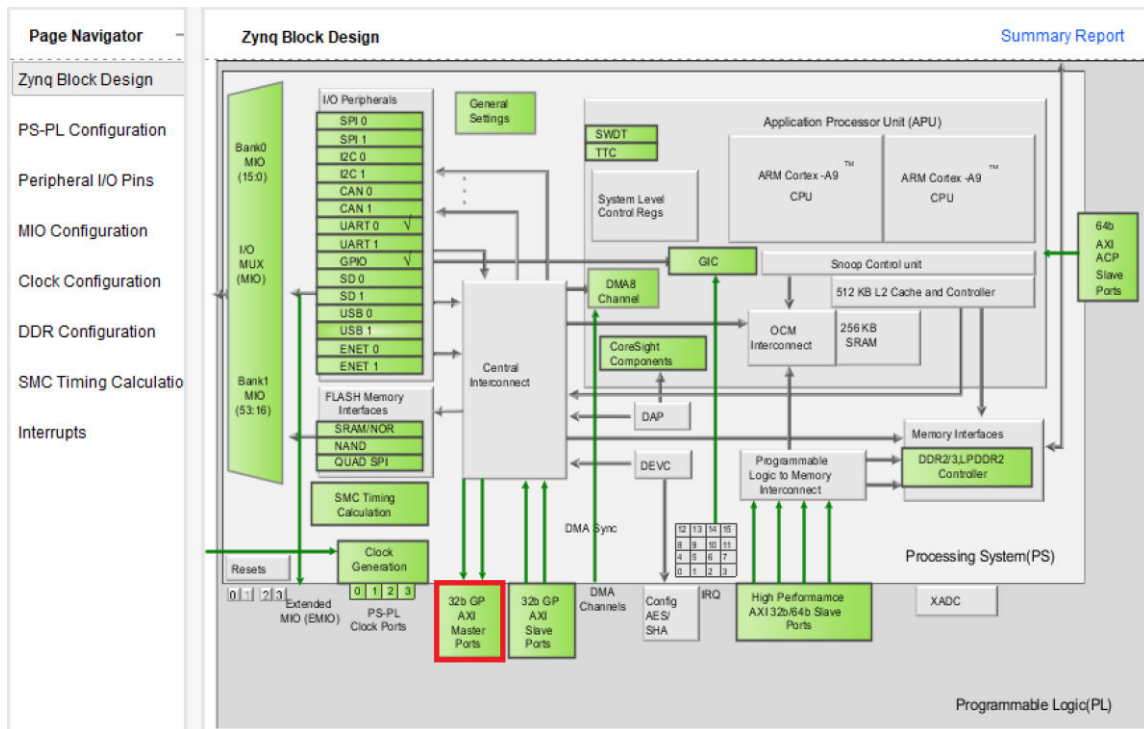


Figura 3. Configuración de los AXI Master Ports

**2-1-4.** Expandir **AXI Non Secure Enablement** ► **GP Master AXI Interface**, si es necesario, y hacer click sobre el casillero **M AXI GP0 interface** para habilitar el puerto AXI GP0.

The PS-PL Configuration window shows the configuration of the 32b Master GP. The 'AXI Non Secure Enablement' section is expanded, and the 'GP Master AXI Interface' is selected. The 'M AXI GP0 interface' is checked, enabling the General purpose AXI master interface 0.

Name	Select	Description
<b>General</b>		
AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
<b>GP Master AXI Interface</b>		
M AXI GP0 interface	<input checked="" type="checkbox"/>	Enables General purpose AXI master interface 0
M AXI GP1 interface	<input type="checkbox"/>	Enables General purpose AXI master interface 1
GP Slave AXI Interface		
HP Slave AXI Interface		
ACP Slave AXI Interface		
DMA Controller		
PS-PL Cross Trigger interface	<input type="checkbox"/>	Enables PL cross trigger signals to PS and vice-versa

Figura 4. Configuración del bloque 32b Master GP

**2-1-5.** Expandir **General** ► **Enable Clock Resets** y seleccionar la opción **FCLK\_RESET0\_N**.

**2-1-6.** Seleccionar a la izquierda la pestaña **Clock Configuration**. Expandir **PL Fabric Clocks** y seleccionar la opción **FCLK\_CLK0** (con requested clock frequency de 100 MHz) y presionar **OK**.

**2-1-7.** Notar la interfaz adicional M\_AXI\_GPO, y los puertos M\_AXI\_GPO\_ACLK, FCLK\_CLK0, y FCLK\_RESET0\_N que han sido incluidos en el bloque Zynq. Ahora puede hacer click en el botón **regenerate** para redibujar el diagrama.

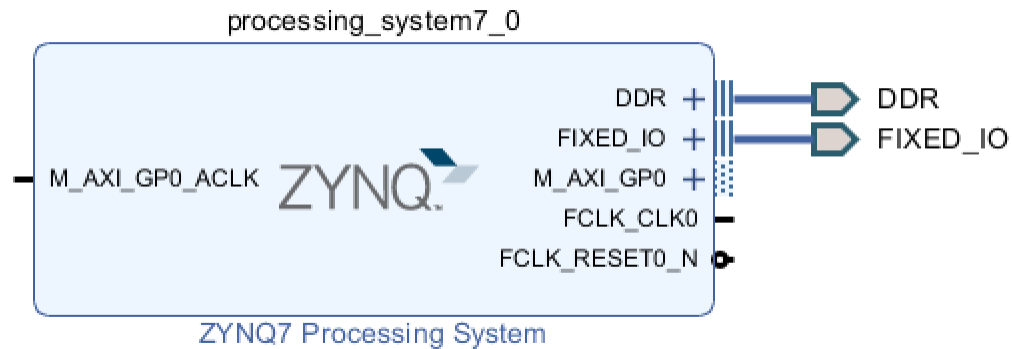


Figura 5. Sistema Zynq con interfaces AXI y de reloj

- 2-1-8. Hacer click derecho sobre el panel *Diagram* y seleccionar *Add IP* y buscar **AXI GPIO** en el catálogo

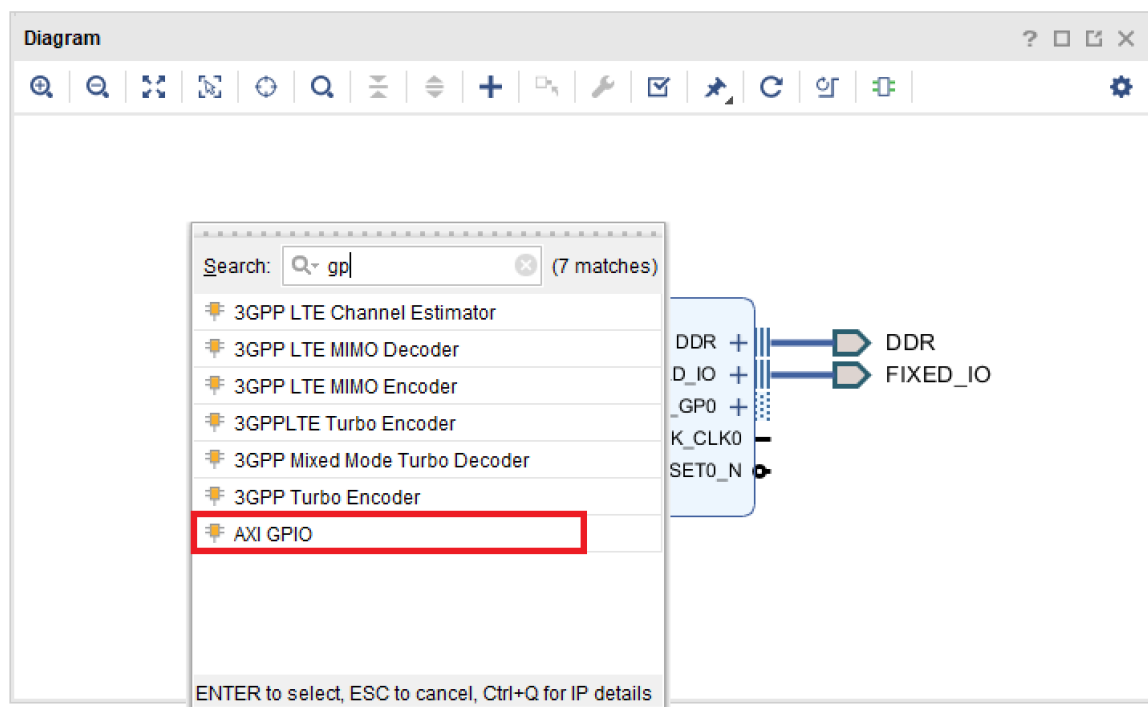


Figura 6. Agregar IP GPIO

**2-1-9.** Hacer doble-click sobre **AXI GPIO** para agregar el core al diseño. El core será agregado al diseño y el block diagram será actualizado.

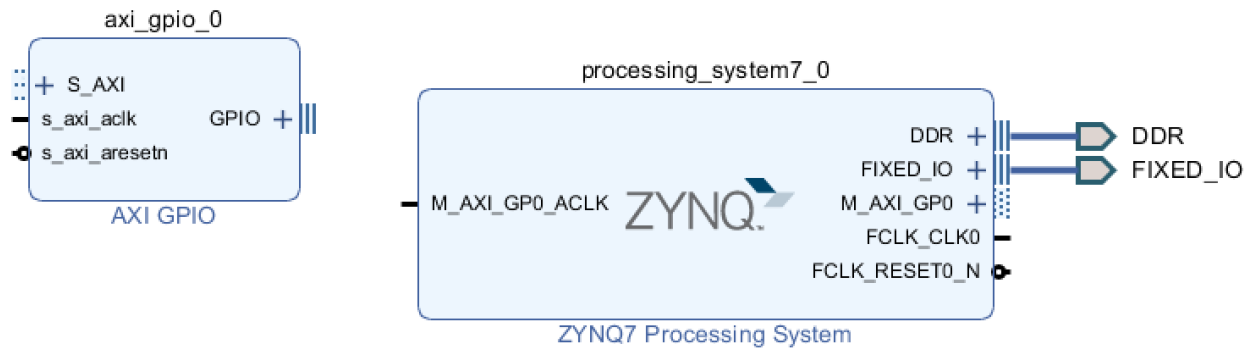


Figura 7. Sistema Zynq con el GPIO AXI agregado

**2-1-10.** Hacer click sobre el bloque **AXI GPIO** para seleccionarlo, y cambiar el nombre a **switches** en la pestaña *properties*

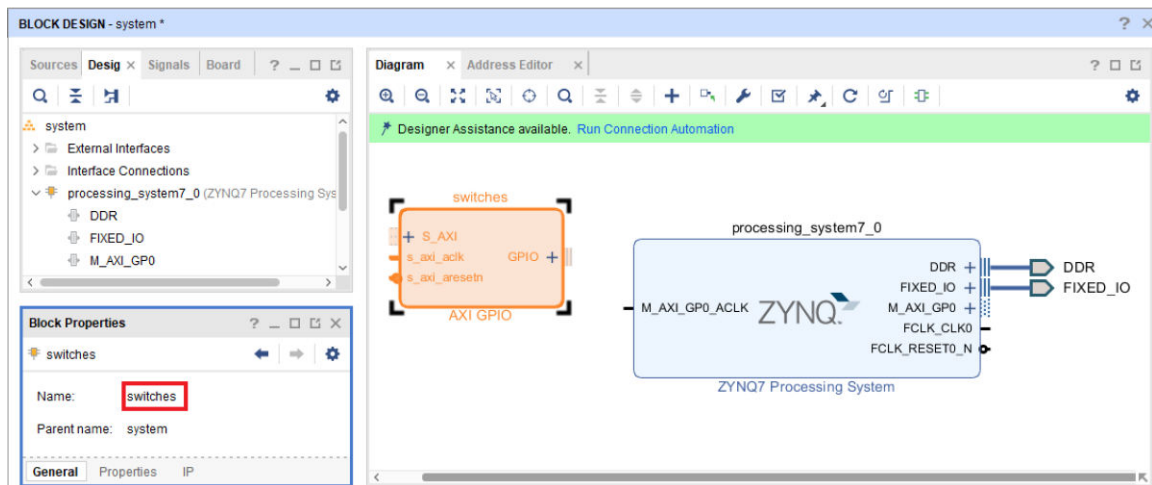


Figura 8. Cambio del nombre por defecto del AXI GPIO

**2-1-11.** Hacer doble click sobre el bloque **AXI GPIO** para abrir la ventana de personalización.

**2-1-12.** En **Board Interface** seleccionar **sws2 bits** para GPIO.

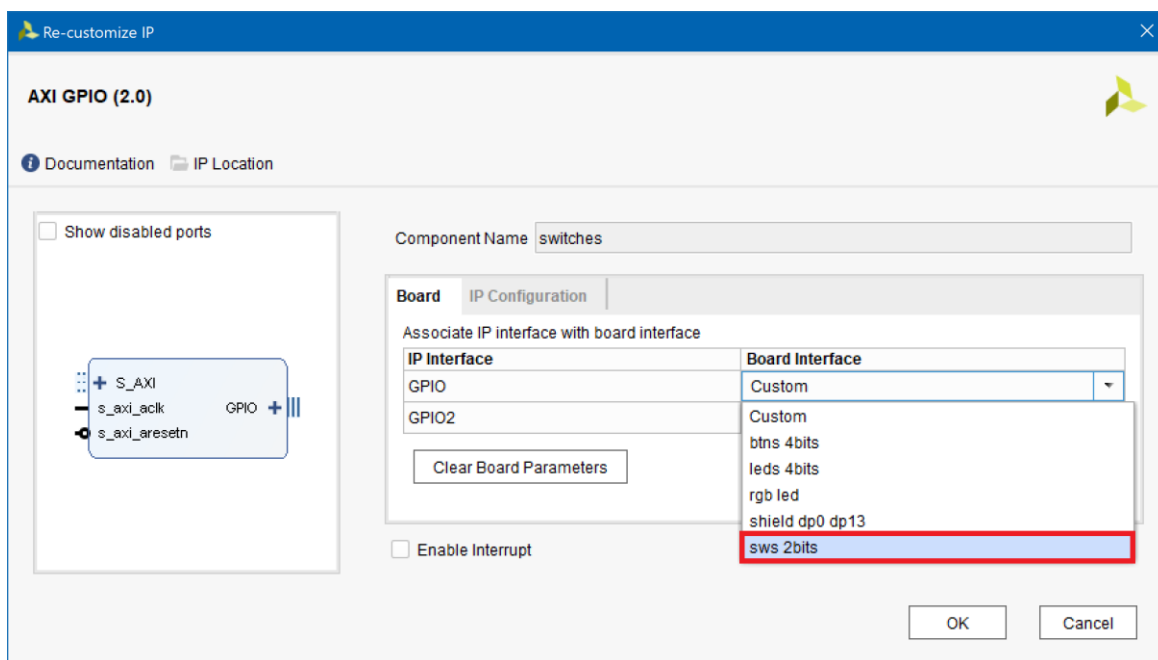


Figura 9. Configurando la interfaz GPIO

**2-1-13.** Hacer click en la pestaña de configuración de la IP, y notar que el ancho ha sido configurado para utilizar los switches existentes en la placa Arty Z7-010 (2)

Notar que el periférico puede ser configurado para dos canales, pero, ya que solo queremos usar uno solo sin interrupciones, deje *Enable Interrupt* y *Enable Dual Channel* sin seleccionar.

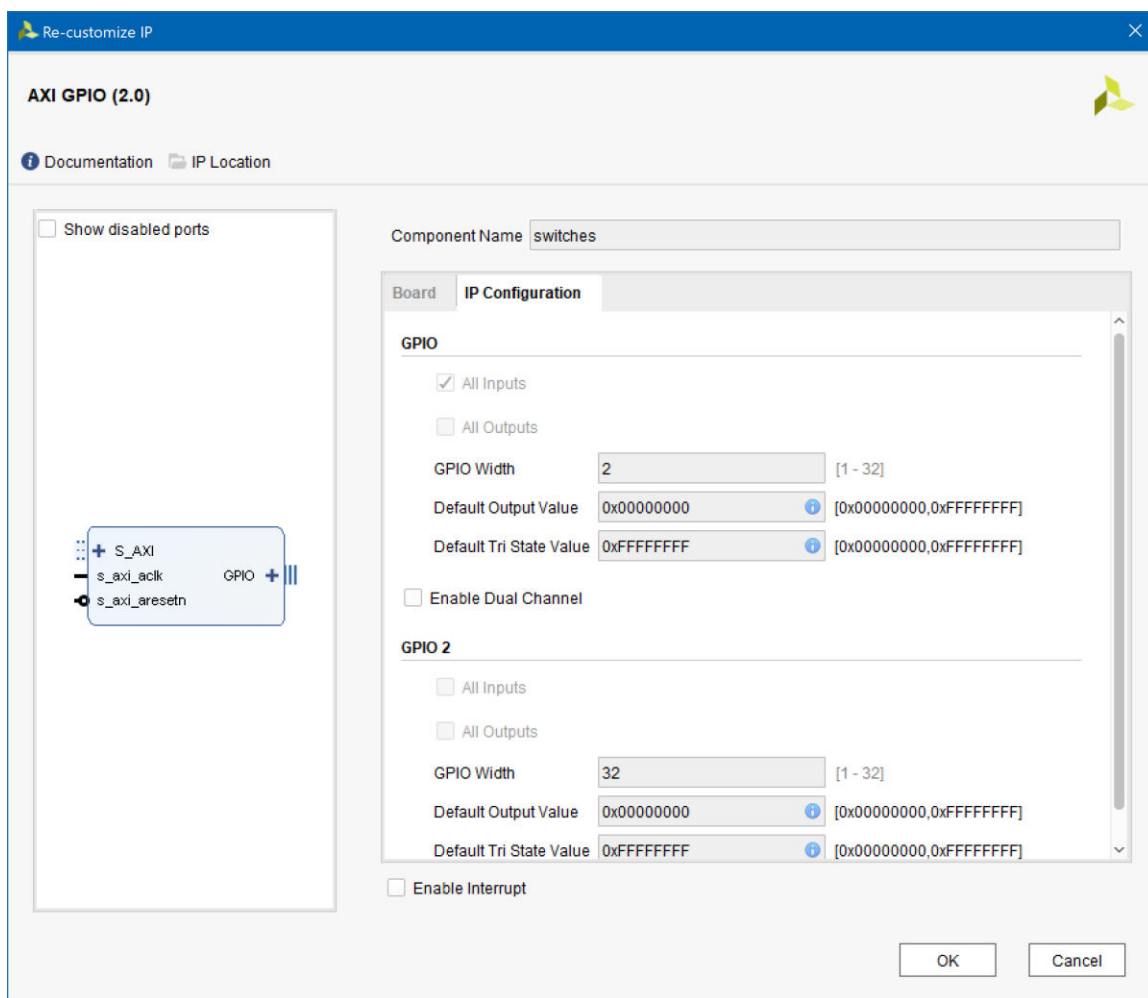


Figura 10. Configurando la instancia GPIO

2-1-14. Hacer click en **OK** para guardar y cerrar la ventana de personalización.

2-1-15. Notar que el *Designer assistance* está disponible. Hacer click sobre **Run Connection Automation**, y seleccionar **/switches/S\_AXI**



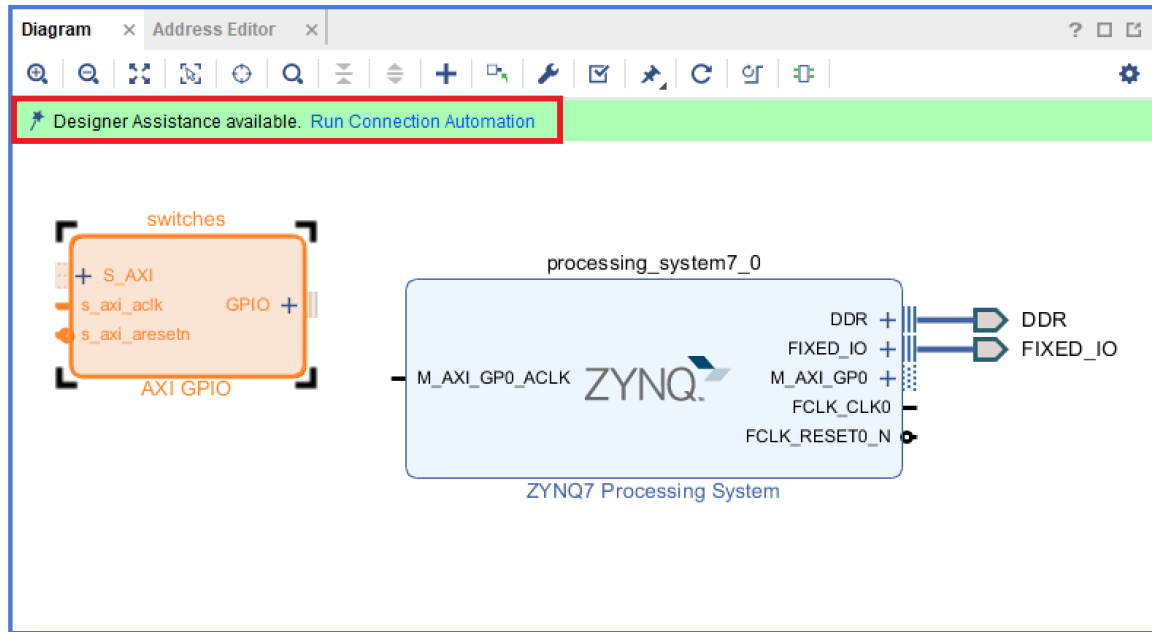


Figura 11. Disponibilidad del asistente al diseñador

**2-1-16.** Hacer click en **OK** cuando se pregunte, dentro de la ventana **Run Connection Automation**, para conectar automáticamente las interfaces maestras y esclavas.

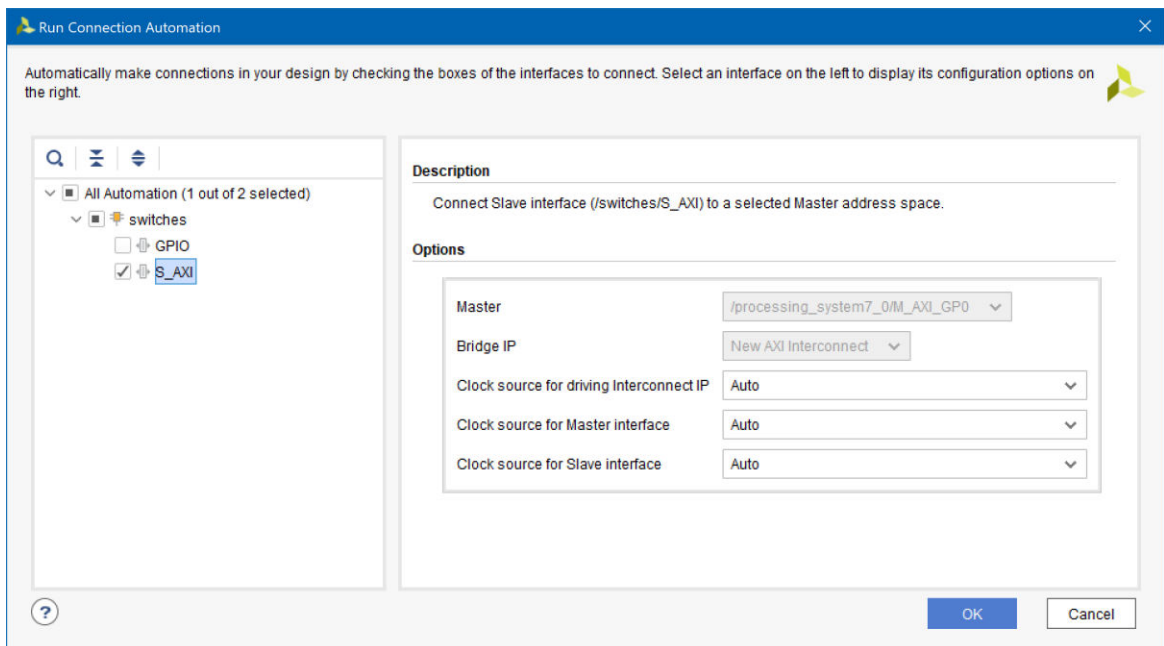


Figura 12. Ejecutar la conexión automática

**2-1-17.** Notar que dos bloques adicionales, *Processor System Reset*, y *AXI Interconnect* han sido automáticamente agregados al diseño. (Los bloques pueden ser arrastrados para reordenarlos o el diseño puede ser redibujado)

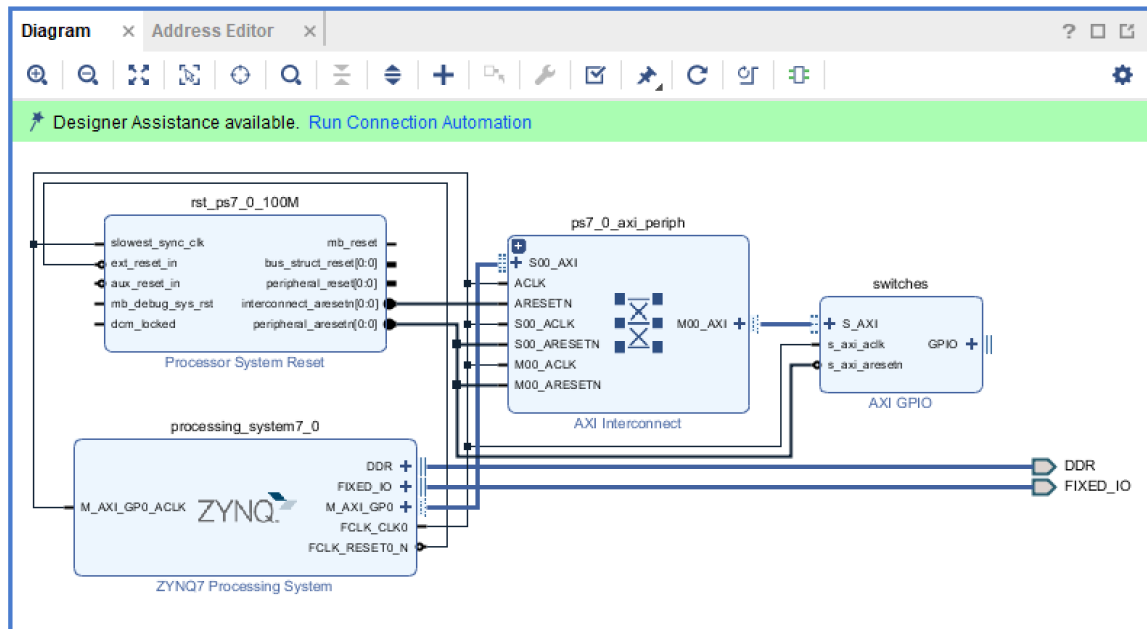


Figura 13. Diseño con switches conectado automáticamente

**2-1-18.** Agregar otra instancia del periférico *GPIO* (**Add IP**). Nómbrelo como **buttons**.

**2-1-19.** Hacer doble click sobre el bloque IP, seleccionar la interfaz *GPIO btns (btns 4bits)* y presionar **OK**.

En este punto la conexión automática podría ser corrida, o el bloque podría ser conectado a mano. Esta vez la conexión se realizará manualmente para experimentar un nuevo método.

**2-1-20.** Hacer doble click sobre la interconexión AXI y cambiar a 2 el número de *Master Interfaces* y hacer click en **OK**.

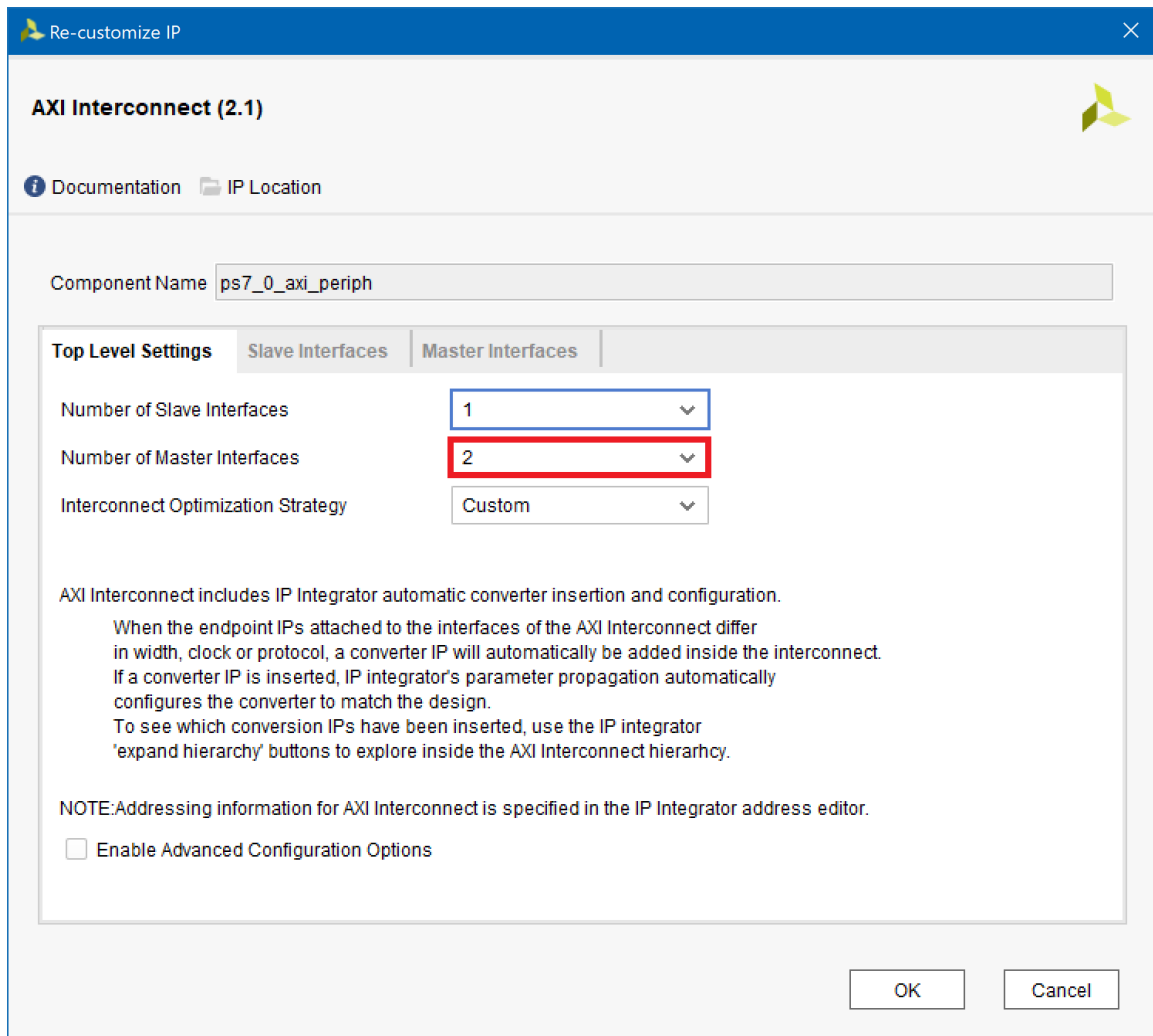


Figura 14. Agregar master port para AXI Interconnect

- 2-1-21.** Hacer click sobre el puerto `s_axi` de los botones del bloque AXI GPIO, y arrastrar el puntero hacia el bloque de interconexión AXI. Debería aparecer el mensaje *Found 1 interface*, y también un tilde verde al lado del puerto `M01_AXI` en la interconexión AXI indicando que ese es un puerto válido para conectarse. Arrastrar el puntero a ese puerto y soltar el botón del mouse para concretar la conexión.

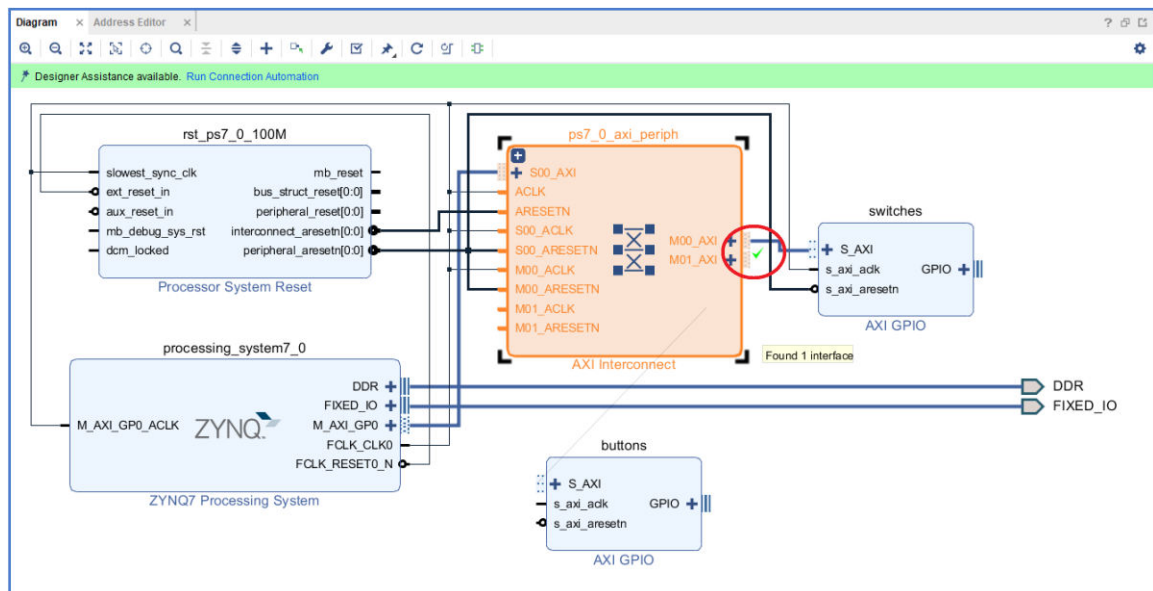


Figura 15. Conexión usando drag and drop

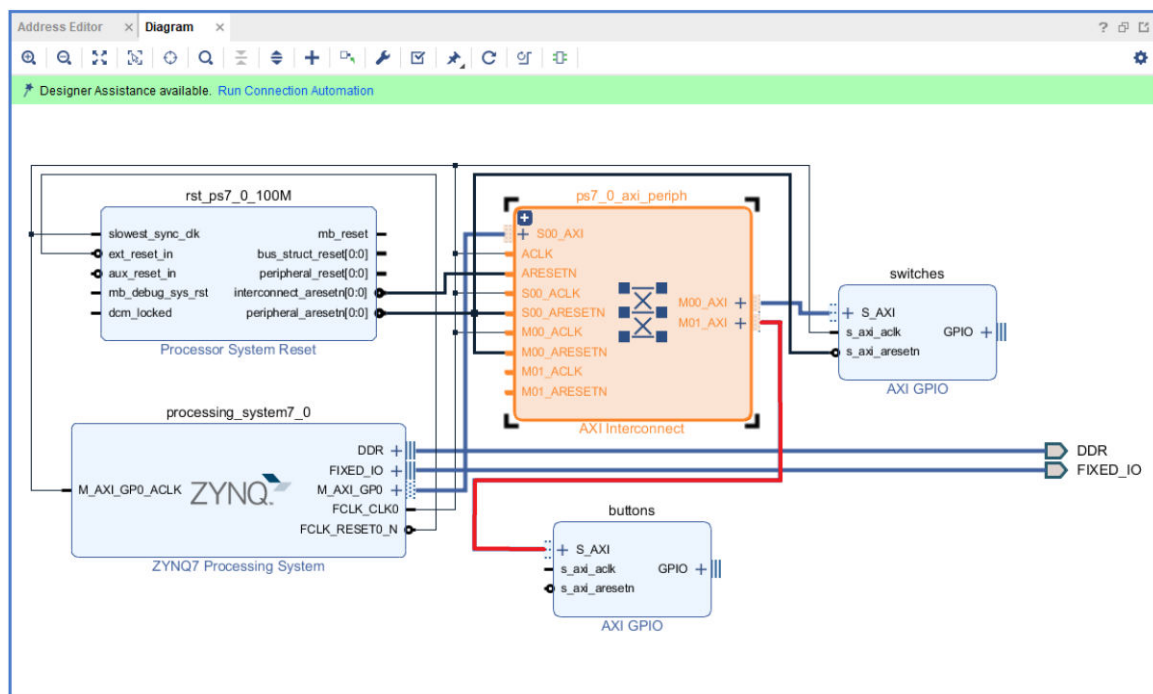


Figura 16. Conexión realizada

2-1-22. Conectar los siguientes puertos de la misma manera:

*buttons s\_axi\_aclk* → Zynq7 Processing System **FCLK\_CLK0**

*buttons s\_axi\_aresetn* → Processor System Reset **peripheral\_aresetn**

*AXI Interconnect M01\_ACLK* → Zynq7 Processing System **FCLK\_CLK0**

*AXI Interconnect M01\_ARESETN* → Processor System Reset **peripheral\_aresetn**

El diagrama debería verse similar a la siguiente imagen:

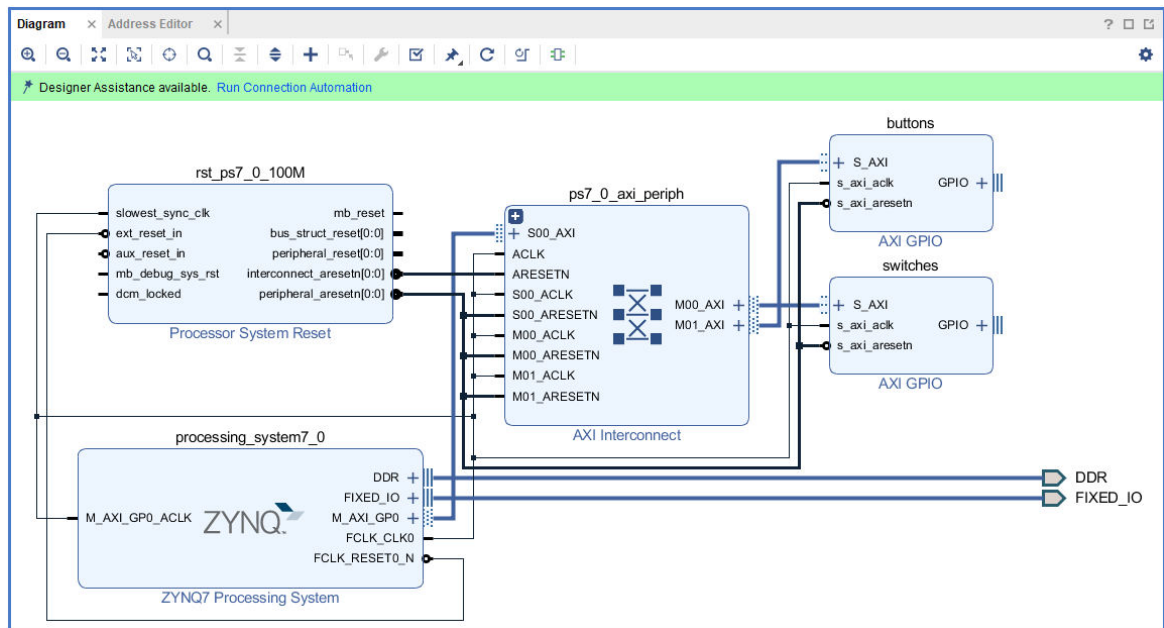


Figura 17. Esquema del sistema después del agregado de los periféricos

**2-1-23.** Hacer click sobre la pestaña *Address Editor*, y expandir **processing\_system7\_0 ► Data ► Unmapped Slaves** si es necesario

**2-1-24.** Notar que a los *switches* se les ha asignado automáticamente una dirección, pero a los botones no (ya que estos últimos fueron conectados manualmente). Hacer click-derecho sobre *buttons* y seleccionar **Assign Address** o hacer click sobre el botón

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ])					
switches	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
Unmapped Slaves (1)					
buttons	S_AXI	Reg			

Figura 18. Puertos no mapeados a memoria

Notar que ambos periféricos son asignados en el rango de direcciones de 0x40000000 a 0x7FFFFFFF (rango de GP0).

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ])					
switches	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
buttons	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF

Figura 19. Mapa de memoria de los periféricos

## Crear la conexión externa de los periféricos GPIO

## Paso 3

**3-1. Las instancias de los botones y de los switches serán conectadas a los correspondientes pines de la placa. Esto puede ser realizado manualmente o usando *Designer Assistance*. La información para concretar este paso está disponible en los manuales de usuario de la placa usada.**

**3-1-1.** En la vista *Diagram*, notar que el *Designer Assistance* está disponible. De todos modos vamos a crear manualmente los puertos y la conexión.

**3-1-2.** Hacer Click-derecho sobre el puerto *GPIO* de la instancia de los switches y seleccionar **Make External** para crear el puerto externo. Esto creará un Puerto externo denominado *GPIO\_0* y conectado al periférico. Ya que Vivado tiene información de la placa utilizada, las restricciones de pines serán automáticamente aplicadas al puerto.

**3-1-3.** Seleccionar el puerto *GPIO\_0* y, en el panel *properties*, cambiar el nombre a **switches**.

El ancho de la interfaz será determinada automáticamente por el bloque de upstream.

**3-1-4.** Para los botones (**buttons**) GPIO, hacer click sobre el link *Run Connection Automation*. Presionar **OK**.

**3-1-5.** Seleccionar el Puerto externo creado y cambiar su nombre a **buttons**

**3-1-6.** Ejecutar *Design Validation* (**Tools ► Validate Design**) y verificar que no hay errores.

El diseño debería verse similar al diagram mostrado a continuación

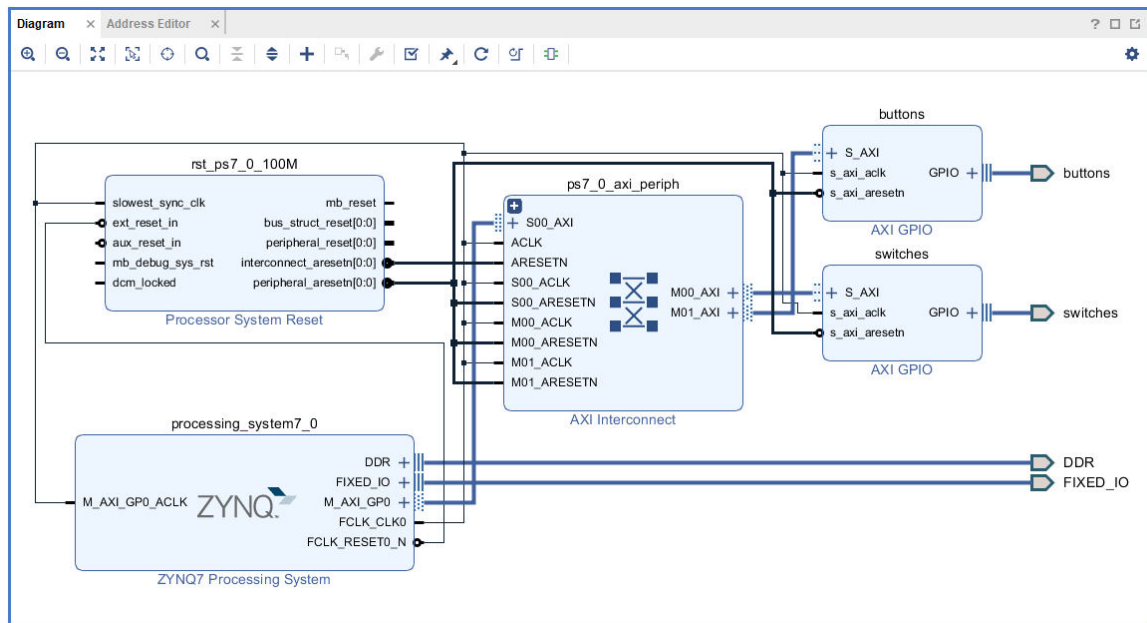


Figura 20. Diseño completo

### 3-2. Sintetizar el diseño, abrir el layout de I/O Planning, y verificar las restricciones usando la herramienta I/O planning.

- 3-2-1. En el Flow Navigator, hacer click en **Run Synthesis**. (Seleccionar **Save** si así se lo requiere). Presionar **OK**. Cuando la síntesis se haya completado, seleccionar **Open Synthesized Design** y **OK**.
- 3-2-2. Seleccionar **I/O Planning** del menu *Layout* (arriba a la derecha del entorno de desarrollo)

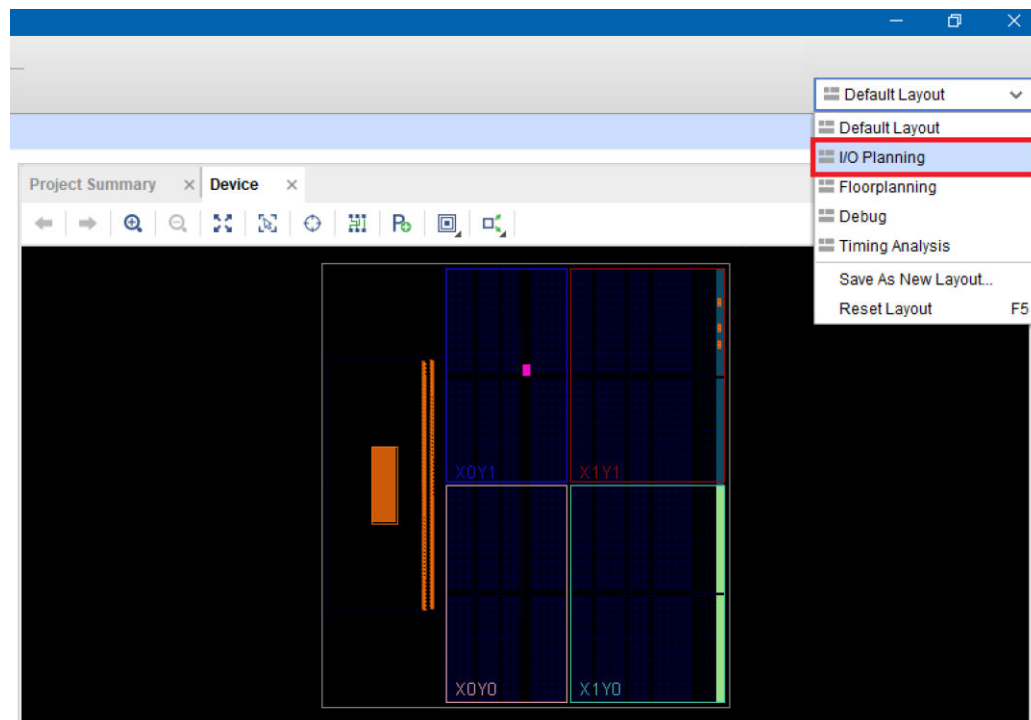


Figura 21. Cambiar a la vista IO planning

- 3-2-3.** En la pestaña I/O ports, expandir los dos iconos GPIO, y expandir *buttons\_tri\_i*, y *switches\_tri\_i*, y notar que los puertos han sido automáticamente asignados a ubicaciones de pines, junto con los otros puertos *FixedIO* en el diseño, y un LVCMOS33 ha sido aplicado. Si no se hubiera aplicado automáticamente, las restricciones de pines pueden ser incluidas en un archivo de restricciones, o incluidas manualmente o modificadas a través de la pestaña de I/O Ports.

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	D
DDR_12642 (71)	INOUT					✓	502	(Multiple)*	1.500	(Multiple)	
FIXED_IO_12642 (59)	INOUT					✓	(Multiple)	(Multiple)*	(Multiple)	(Multiple)	
GPIO_41639 (2)	IN					✓	35	LVCMOS33*	3.300		
switches_tri_i (2)	IN					✓	35	LVCMOS33*	3.300		
switches_tri_i[1]	IN	sws_2bits_tr...			M19	✓	35	LVCMOS33*	3.300		
switches_tri_i[0]	IN	sws_2bits_tr...			M20	✓	35	LVCMOS33*	3.300		
Scalar ports (0)											
GPIO_43611 (4)	IN					✓	35	LVCMOS33*	3.300		
buttons_tri_i (4)	IN					✓	35	LVCMOS33*	3.300		
buttons_tri_i[3]	IN	btns_4bits_tr...			L19	✓	35	LVCMOS33*	3.300		
buttons_tri_i[2]	IN	btns_4bits_tr...			L20	✓	35	LVCMOS33*	3.300		
buttons_tri_i[1]	IN	btns_4bits_tr...			D20	✓	35	LVCMOS33*	3.300		
buttons_tri_i[0]	IN	btns_4bits_tr...			D19	✓	35	LVCMOS33*	3.300		
Scalar ports (0)											
Scalar ports (0)											

Figura 22. Restricciones de los pines para la placa ArtyZ7-010

## Generar el Bitstream y Exportar a SDK

## Paso 4

- 4-1.** Generar el bistream, y exportar el hardware junto con el bitstream generado al SDK.



- 4-1-1. Hacer click sobre **Generate Bitstream**, y presionar **Yes** si se solicita lanzar la implementación (Hacer click en **Yes** si se solicita guardar el diseño).
- 4-1-2. Hacer click en **Cancel** en el cuadro de diálogo *Bitstream Generation succesfully completed*.
- 4-1-3. Exportar el hardware seleccionando **File ► Export ► Export Hardware** y hacer click en **OK**. En esta oportunidad, hay hardware en la lógica programable (PL) y un bitstream ha sido generado y debería ser incluido en el proceso de exportación al SDK.

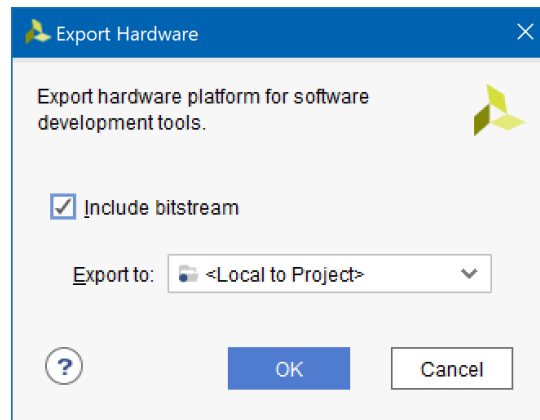


Figura 23. Exportar el diseño

- 4-1-4. Hacer click en **Yes** para sobre escribir el módulo de hardware.
- 4-1-5. Iniciar el SDK haciendo click en **File ► Launch SDK** y presionando **OK**

## Generar la Aplicación TestApp en SDK

## Paso 5

- 5-1. **Cerrar los proyectos de la práctica anterior. Generar un proyecto de plataforma de software con la configuración por defecto y un proyecto de software (nombre por defecto, standalone\_0).**
  - 5-1-1. En el SDK, hacer botón-derecho sobre el proyecto *mem\_test* de la práctica anterior y seleccionar **Close Project**.
  - 5-1-2. Hacer lo mismo para *mem\_test\_bsp* y *system\_wrapper\_hw\_platform\_0*. Se debería ver algo similar a lo mostrado por la figura siguiente.

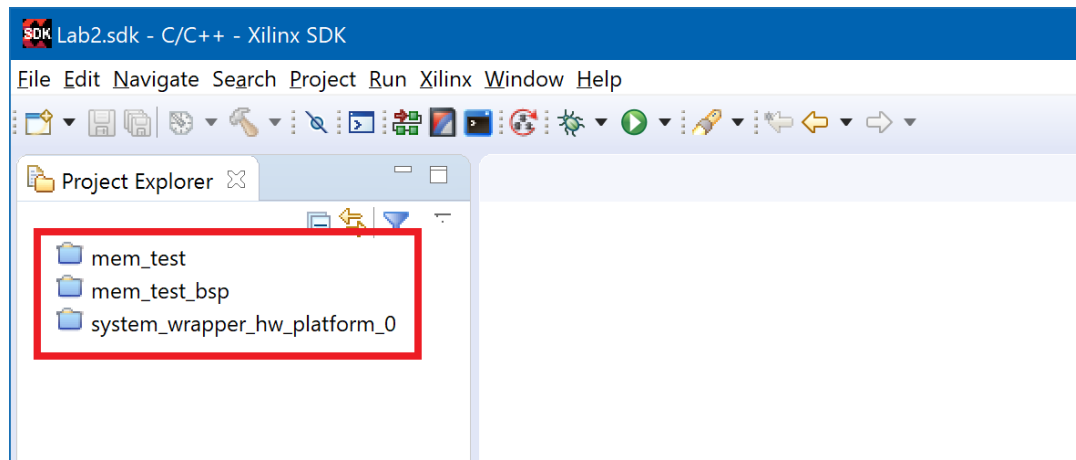


Figura 24

- 5-1-3. Desde el menú *File* seleccionar **File ► New ► Board Support Package**
- 5-1-4. Hacer click en **Finish** con el *standalone* OS seleccionado y nombre del proyecto por defecto *standalone\_bsp\_0*.
- 5-1-5. Hacer click en **OK** para generar el *board support package* nombrado *standalone\_bsp\_0*.
- 5-1-6. Desde el menú *File* seleccionar **File ► New ► Application Project**.
- 5-1-7. Nombrar el proyecto como **TestApp**, seleccionar *Use existing* board support package, seleccionar **standalone\_bsp\_0** y hacer click en **Next**.

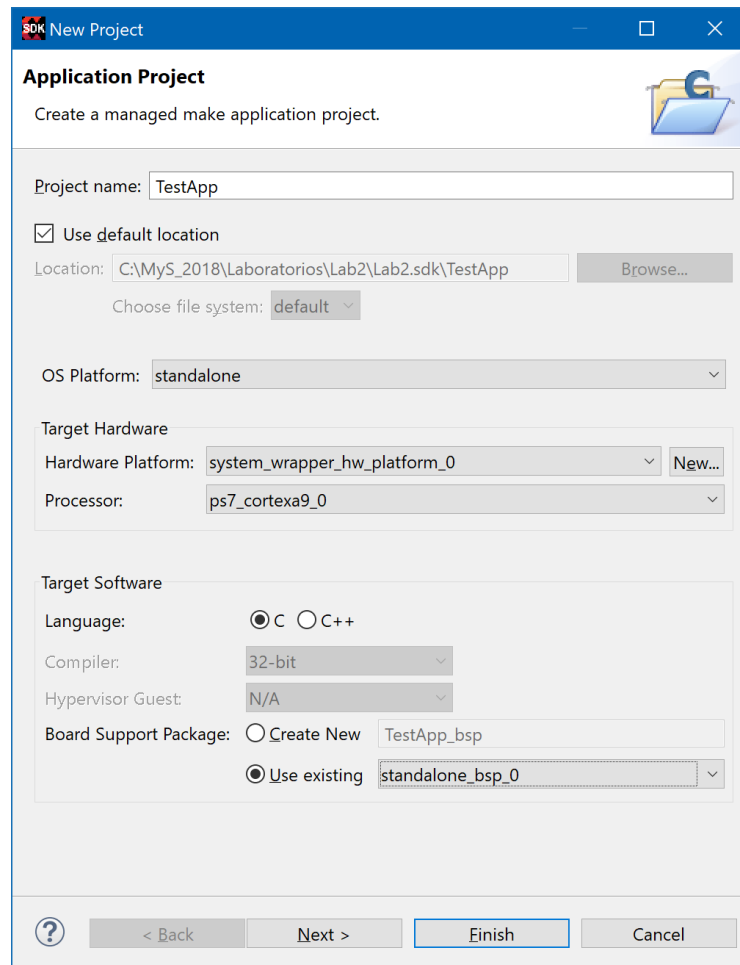


Figura 25. Configuración del proyecto de aplicación

**5-1-8.** Seleccionar **Empty Application** y hacer click en **Finish**.

Esto creará un proyecto de aplicación nuevo usando el *board support package* creado antes.

**5-1-9.** El generador de librería correrá en background y creará el archivo **xparameters.h** en el directorio **lab2\lab2.sdk\standalone\_bsp\_0\ps7\_cortexa9\_0\include**.

**5-1-10.** Expandir **TestApp** en la vista de proyecto, botón derecho del mouse sobre la carpeta **src**, y seleccionar **Import**.

**5-1-11.** Expandir la categoría **General** y hace doble click sobre **File System**.

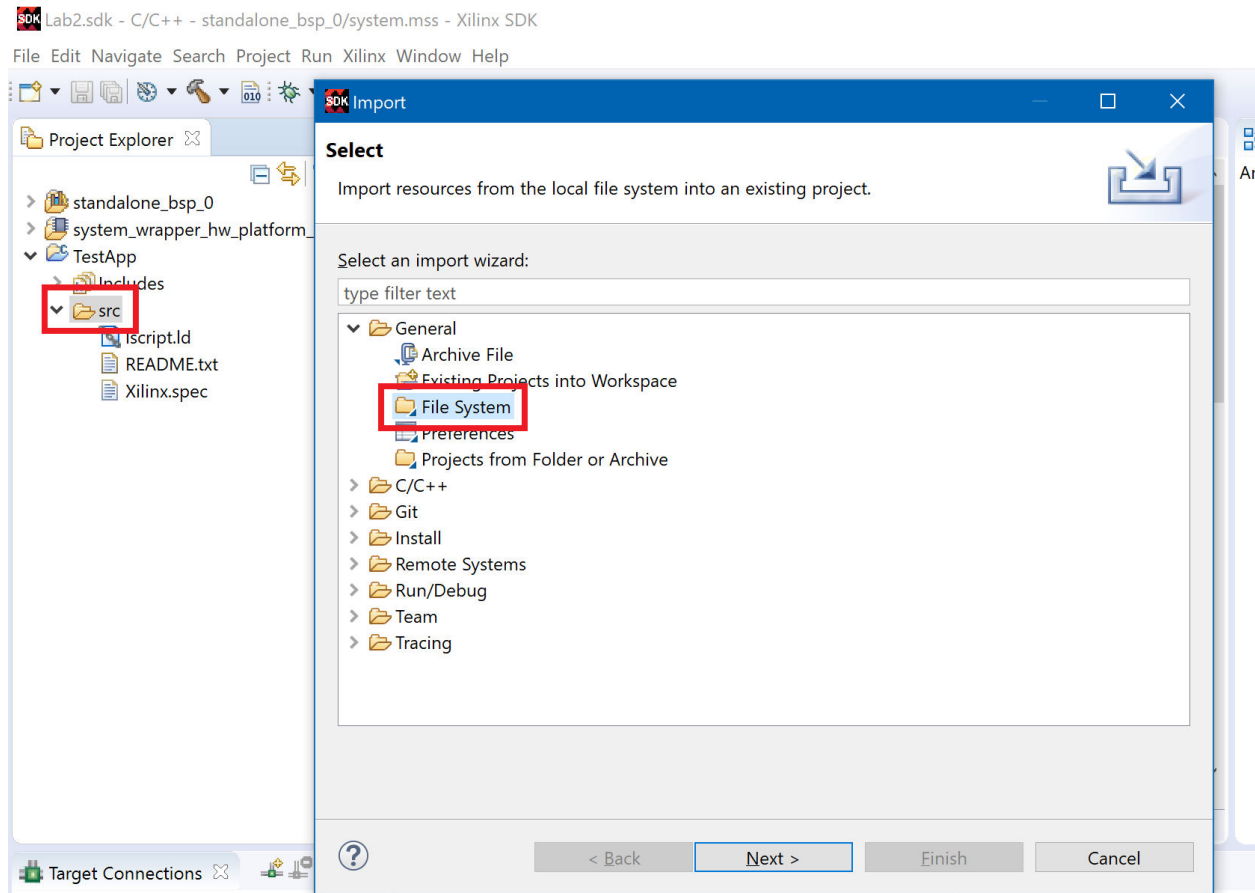


Figura 26. Importar archivo fuente

**5-1-12.** Navegar hasta el directorio donde se encuentra el archivo lab2.c (existente en el campus)

**5-1-13.** Seleccionar **lab2.c** y presionar **Finish**.

En la figura siguiente se muestra un pedazo de código del mismo.

```

#include "xparameters.h"
#include "xgpio.h"

//=====

int main (void)
{
    XGpio dip, push;
    int psb_check, dip_check;

    xil_printf("-- Start of the Program --\r\n");

    XGpio_Initialize(&dip, XPAR_SWITCHES_DEVICE_ID);
    XGpio_SetDataDirection(&dip, 1, 0xffffffff);

    XGpio_Initialize(&push, XPAR_BUTTONS_DEVICE_ID);
    XGpio_SetDataDirection(&push, 1, 0xffffffff);

    while (1)
    {
        psb_check = XGpio_DiscreteRead(&push, 1);
        xil_printf("Push Buttons Status %x\r\n", psb_check);
        dip_check = XGpio_DiscreteRead(&dip, 1);
        xil_printf("DIP Switch Status %x\r\n", dip_check);

        sleep(1);
    }
}

```


Figura 27. Pedazo de código fuente


## Prueba en Hardware

## Paso 6

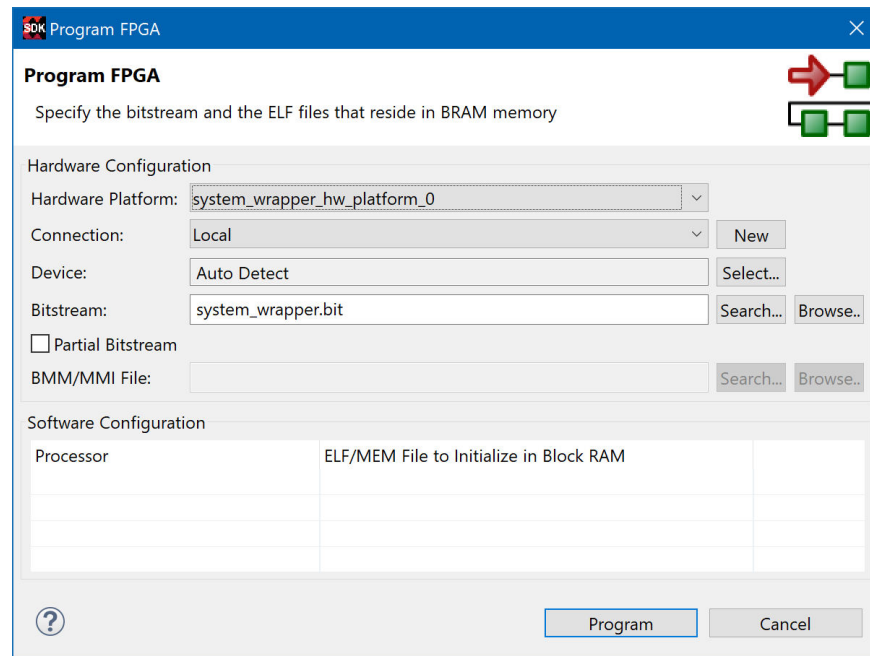
### 6-1. Conectar la placa con un cable micro-usb. Establecer una comunicación serial usando la terminal de SDK (pestaña Terminal).

6-1-1. Asegurarse que un cable micro-USB esté conectado entre la placa y la PC.

6-1-2. Seleccionar la pestaña  **Terminal**. Si no está visible seleccione **Window ► Show view ► Terminal**.

6-1-3. Hacer click sobre el icono  y, si es requerido, seleccionar el Puerto COM apropiado (depende de su computadora), y configurarlo con los parámetros tal cual lo hizo en la práctica anterior.

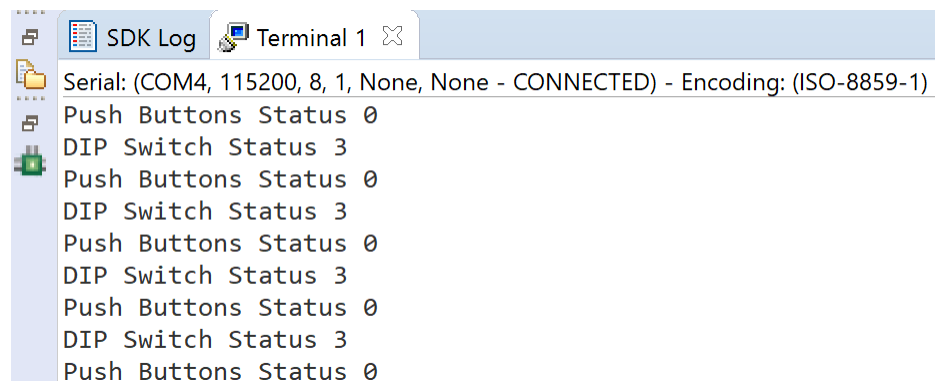
### 6-2. Programar la FPGA seleccionando Xilinx Tools ► Program FPGA y asignando el archivo system.bit. Ejecutar la aplicación TestApp y verificar la funcionalidad

**6-2-1. Seleccione Xilinx Tools ► Program FPGA***Figura 28. Programando la FPGA*

**6-2-2.** Hacer click sobre **Program** para descargar el bitstream de hardware. Cuando la FPGA esté siendo programada el led DONE (de color verde) se apagará, y se encenderá cuando la FPGA esté finalmente programada.

**6-2-3.** Seleccionar **TestApp** en el *Project Explorer*, click-derecho y seleccionar **Run As ► Launch on Hardware (GDB)** para descargar la aplicación, ejecutar ps7\_init, y ejecutar TestApp.elf

**6-2-4.** Debería verse algo parecido a la siguiente salida de consola que se muestra en la imagen (en este caso se encuentran puestos a 1 los dos switches)

*Figura 29. Salida de la terminal del SDK*

**6-2-5.** Cerrar el SDK y Vivado seleccionando **File ► Exit** en cada programa

**6-2-6.** Desconectar la placa

## Conclusión

En esta práctica fueron agregados periféricos GPIO del catálogo IP y conectados al sistema de procesamiento a través de la interfaz 32b Master GP0. Los periféricos fueron configurados y se establecieron conexiones externas. Se creó un proyecto de aplicación TestApp y la funcionalidad fue verificada después de haber descargado el bitstream y ejecutado el programa.