

Diseño Digital Avanzado

Microblaze - GPIO - VIO

Dr. Ariel L. Pola

apola@fundacionfulgor.org.ar

December 13, 2021

Tabla de Contenidos

1. Introducción
2. Nuevo Proyecto
3. Creando un nuevo diseño
4. Instancias de periféricos
5. Instancia de VIO
6. Instanciar el uP en Top Level
7. Instanciar el VIO en Top Level
8. Crear la aplicación en Vitis
9. Tunel, Programar la FPGA y Ejecutar la Aplicación en Forma Remota

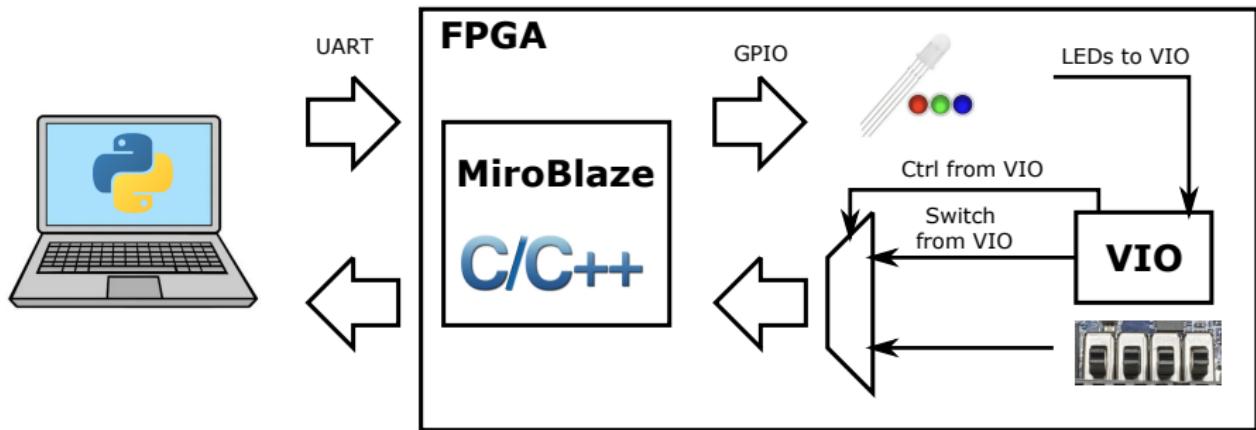


Introducción

Introducción

- El objetivo de esta presentación es detallar paso a paso la instancia y programación de un micro-embebido.
- El proyecto finaliza con el encendido de leds utilizando un script de python ejecutado en la PC y comunicándose con la FPGA utilizando el puerto UART y GPIOs hacia los leds.

Introducción

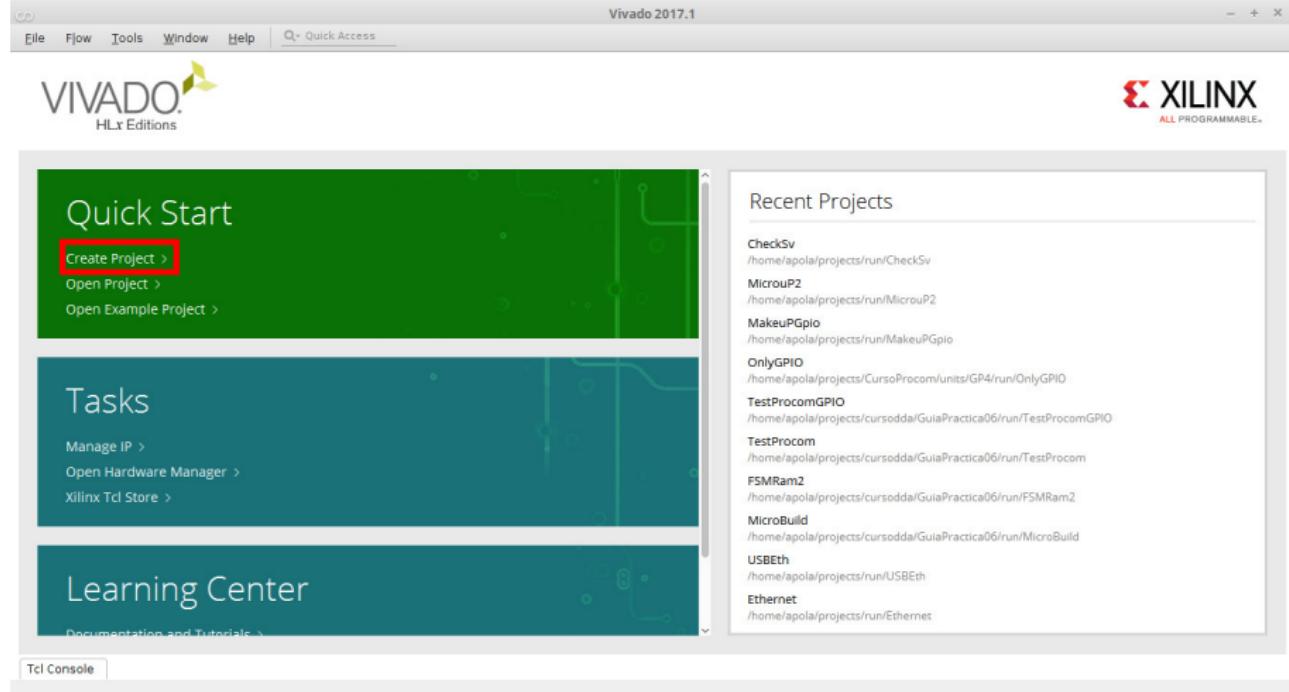


Descripción general.

Nuevo Proyecto



Nuevo Proyecto



Crear un nuevo proyecto.

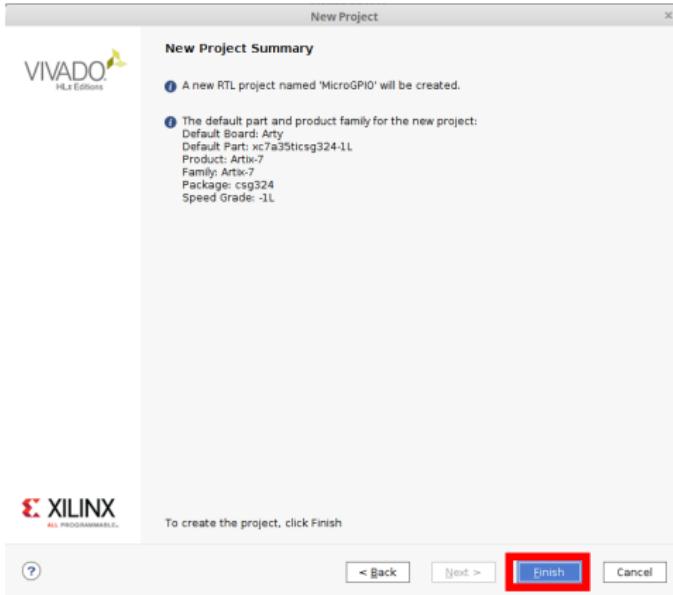
Nuevo Proyecto

The screenshot shows the "New Project" wizard in Vivado. It consists of five sequential windows:

- Create a New Vivado Project**: A welcome screen with a "Next Step" button highlighted.
- Project Name**: Shows the project name as "MicroPIO" and the project location as "/home/napolap/projects/CursoProcesamientoGP/Run". A red box highlights the project name field.
- Project Type**: Shows options for "RTL Project", "Zynq-synthesis Project", "JTAG Planning Project", "Import Project", and "Empty Project". The "RTL Project" option is selected and highlighted with a red box.
- Default Part**: Shows a list of parts under the "Arty" category. The "Arty" part is selected and highlighted with a red box. Other parts listed include "Cmx32-150", "Cmx32-250", "Nexys4", and "Nexys4 DDR".
- Summary**: A summary screen with a "Finish" button highlighted.

Definir un nombre del proyecto, directorio de trabajo y kit Arty.

Nuevo Proyecto

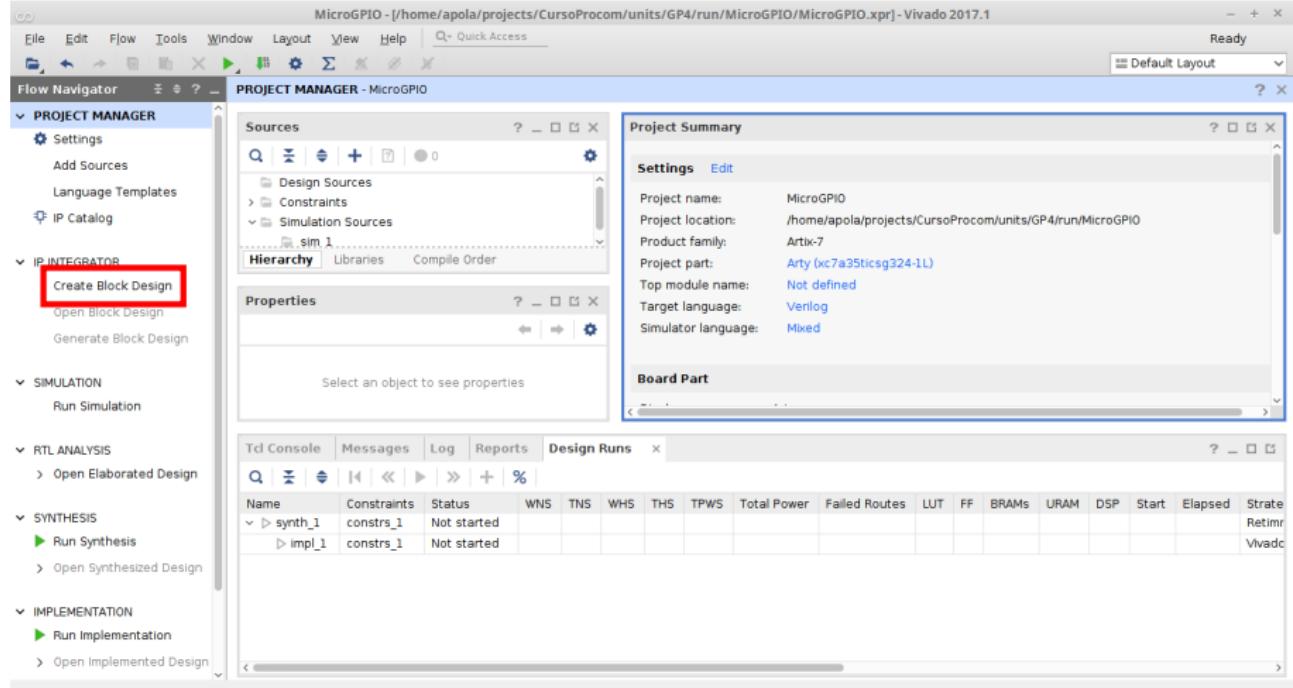


Resumen del proyecto.



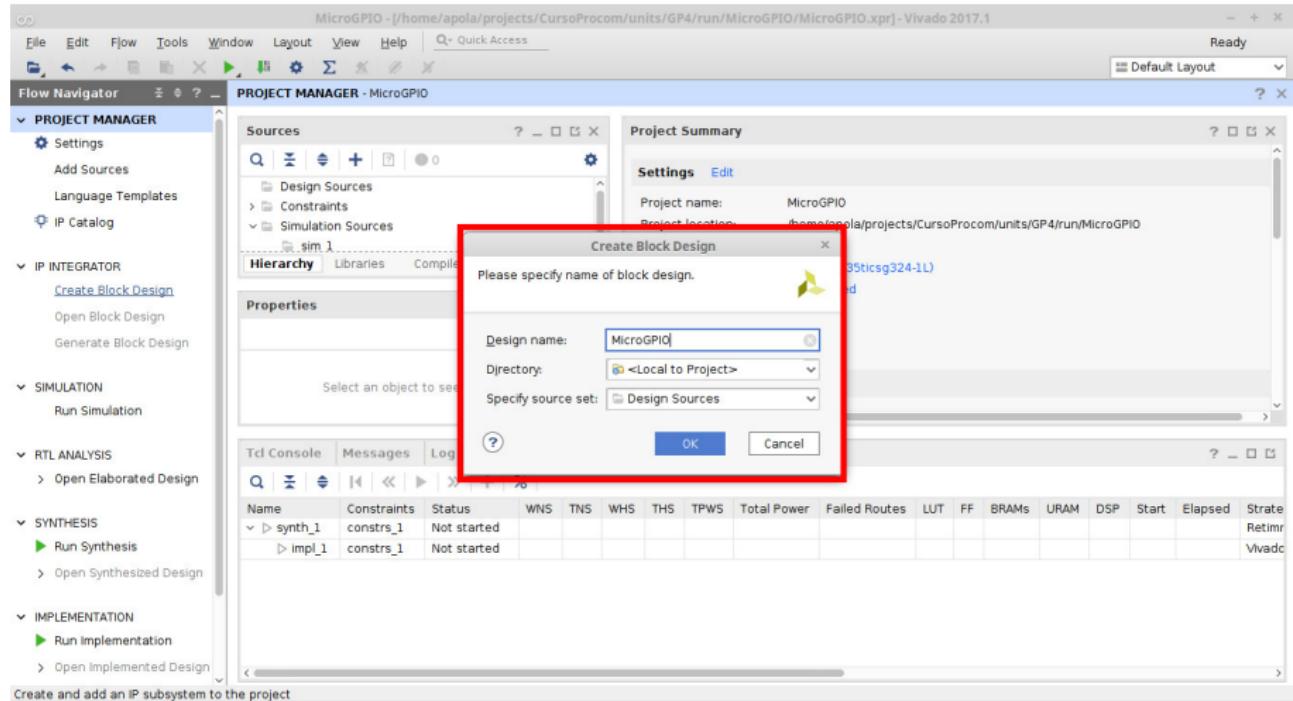
Creando un nuevo diseño

Creando un nuevo diseño



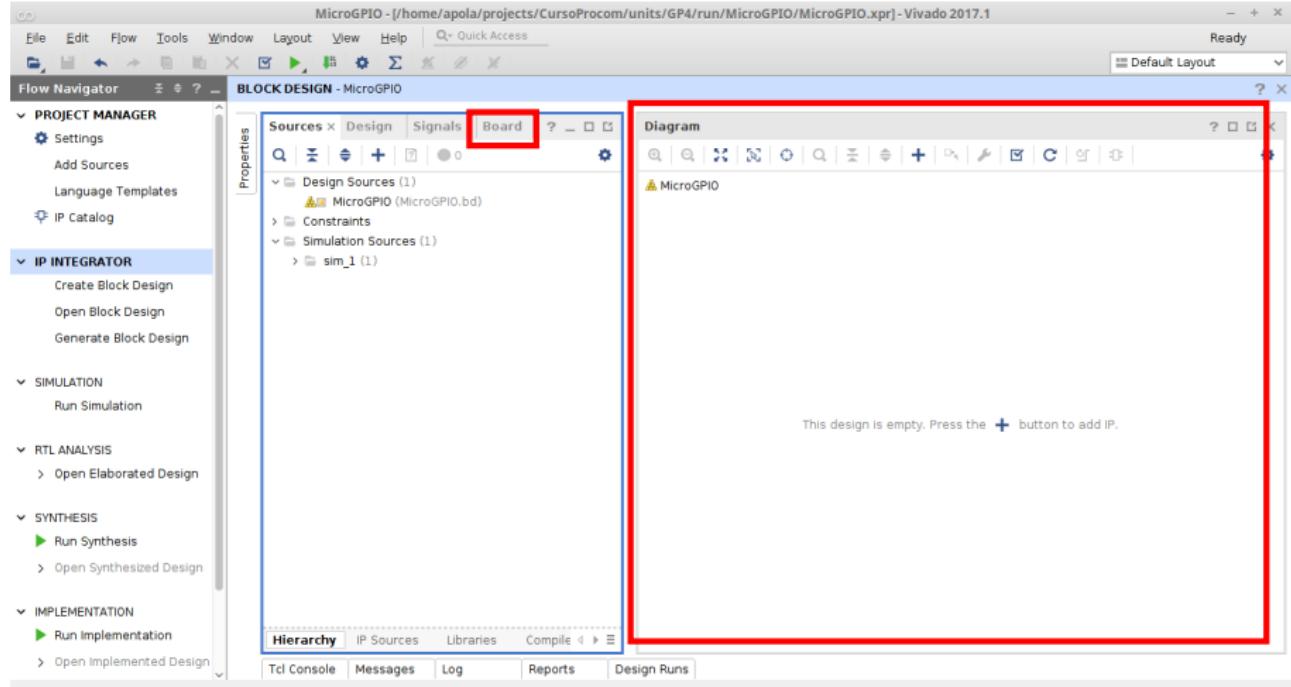
Crean un nuevo bloque.

Creando un nuevo diseño



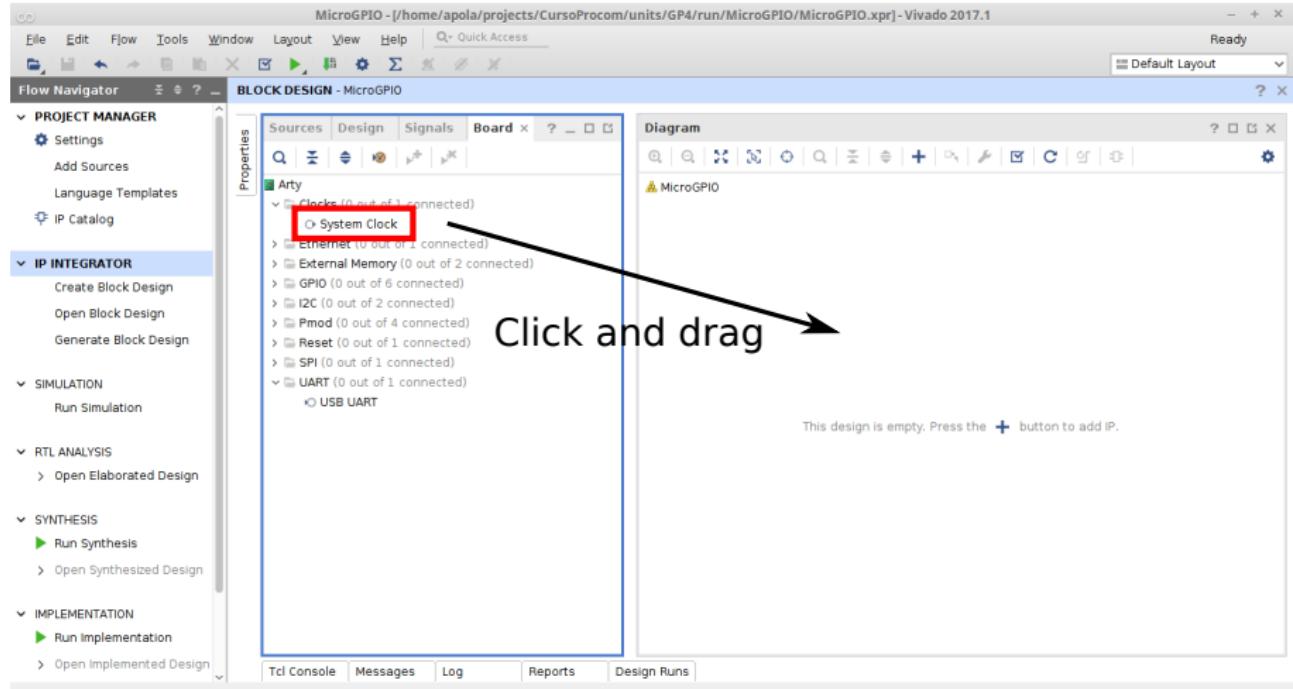
Definir el nombre del nuevo diseño y el directorio de trabajo.

Creando un nuevo diseño



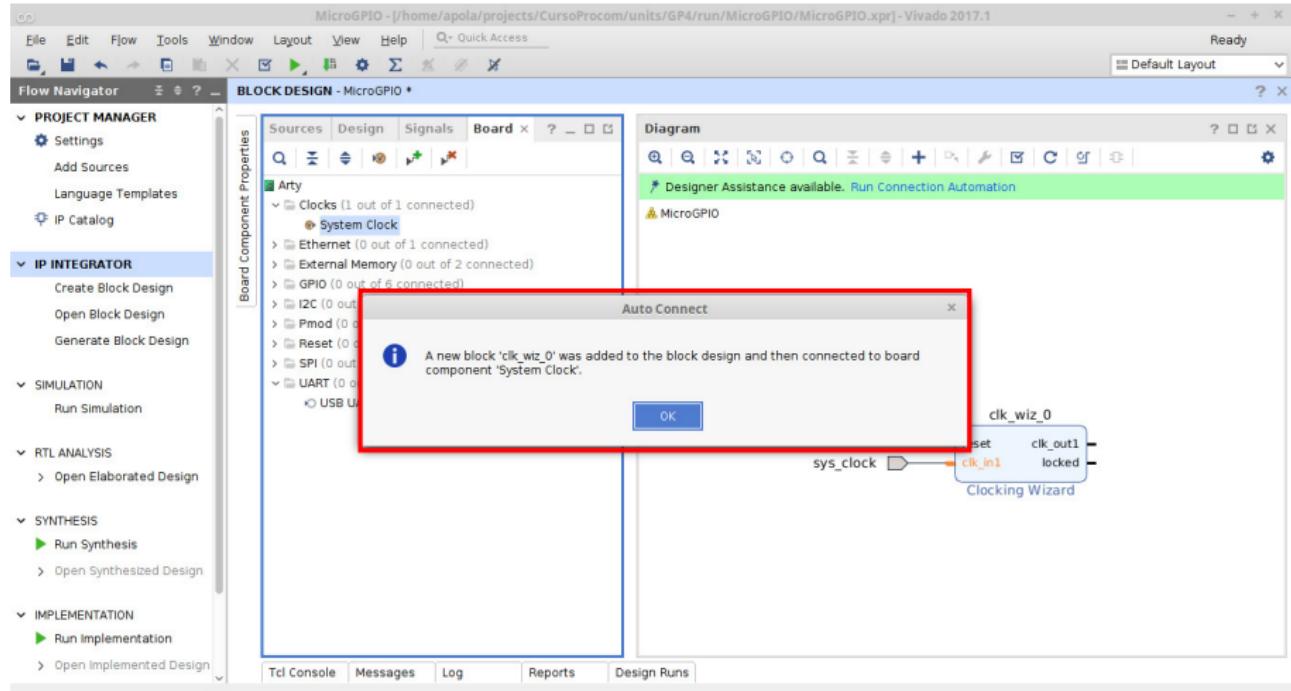
Nueva ventana de trabajo denominada “Diagrama” y en la pestaña “Board” se definen todos los componentes de la FPGA seleccionada.

Creando un nuevo diseño



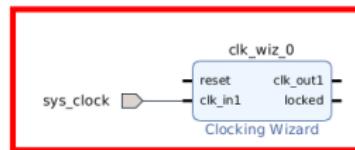
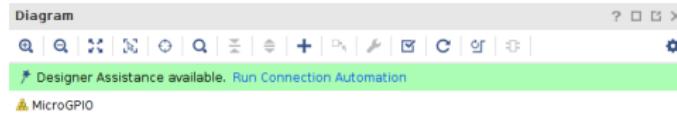
Instanciar “System Clock” haciendo click y arrastrando hacia la ventana.

Creando un nuevo diseño



Mensaje de instancia de bloque.

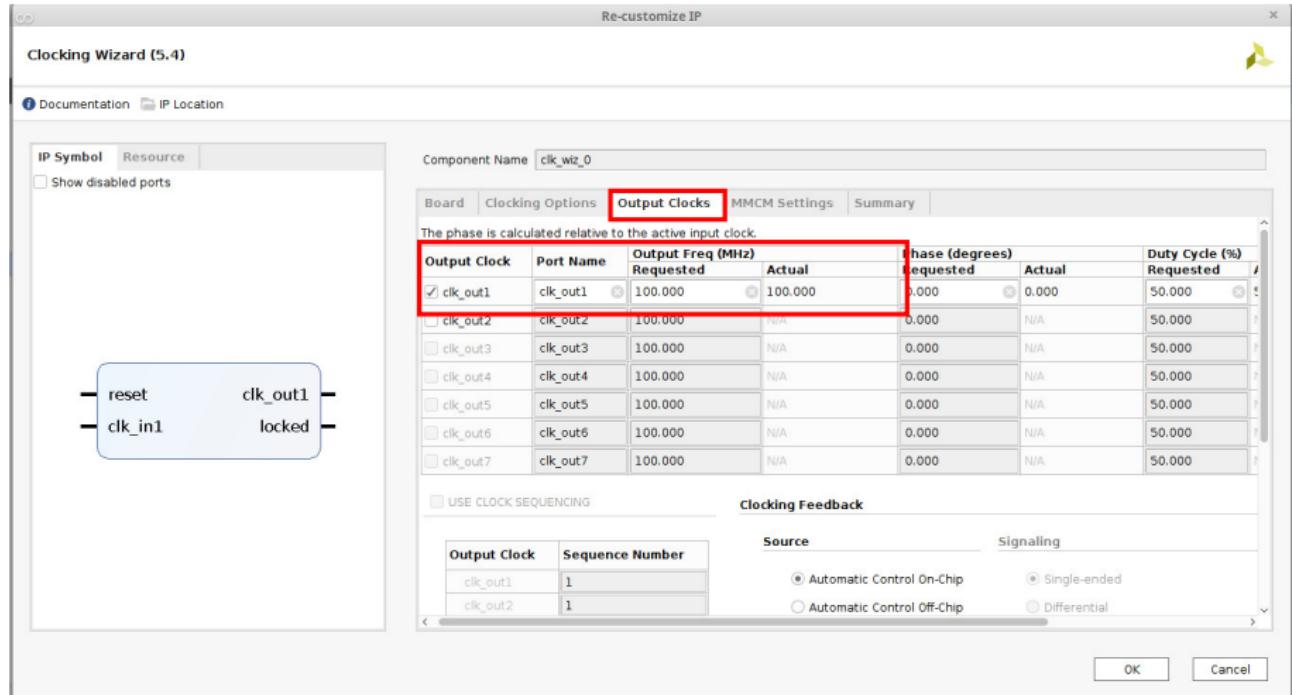
Creando un nuevo diseño



Double Click

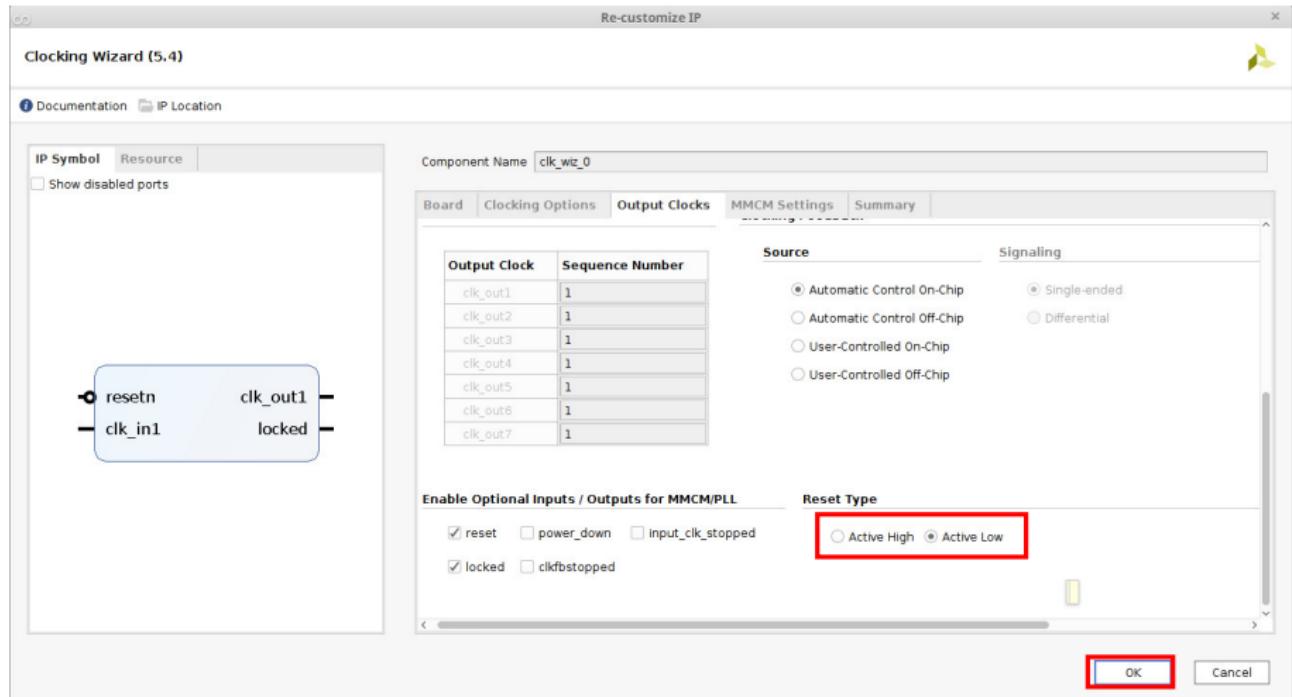
Hacer doble click sobre “Clocking Wizard”.

Creando un nuevo diseño



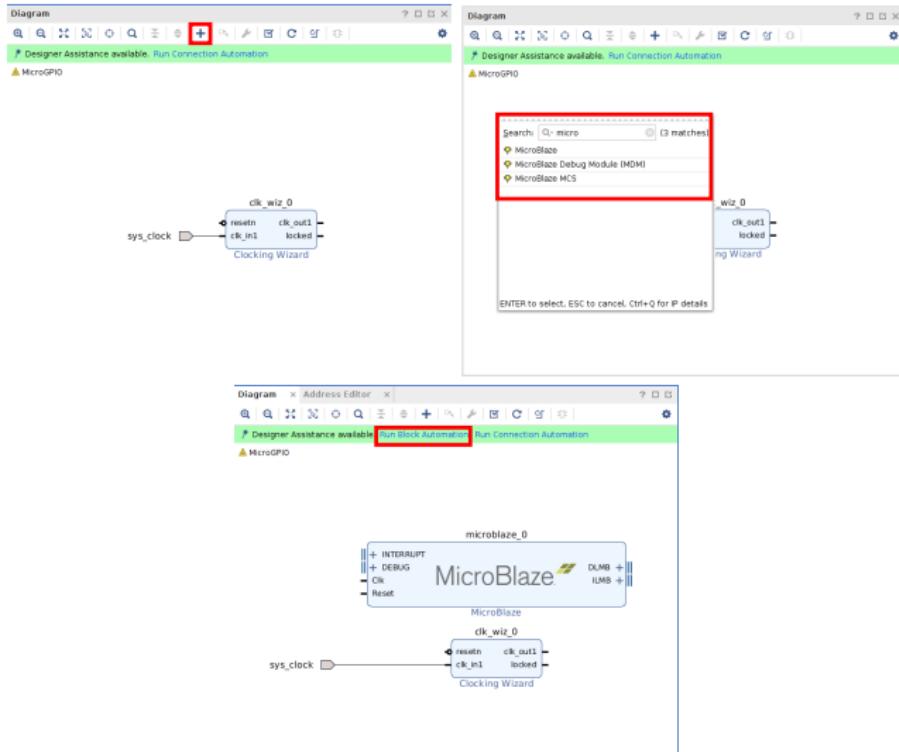
En la pestaña “Output Clocks” asignar la frecuencia de salida en 100MHz.

Creando un nuevo diseño



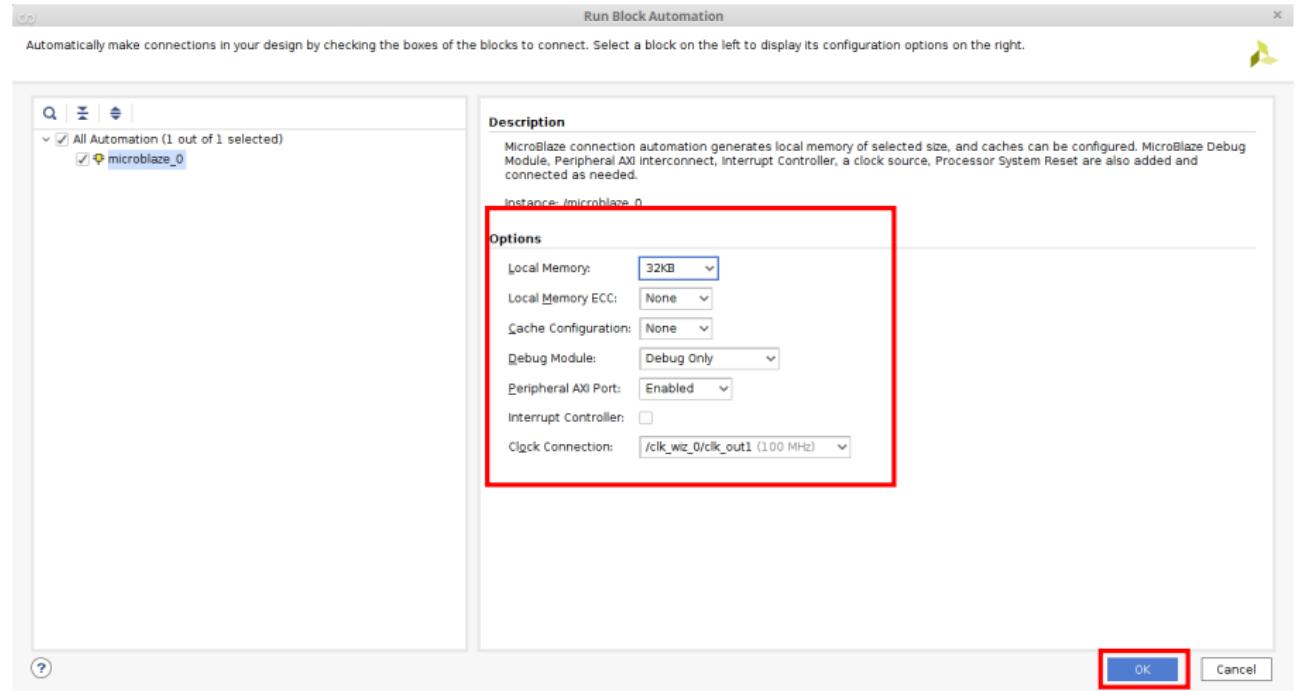
El reset del bloque tiene que ser activo por bajo.

Creando un nuevo diseño



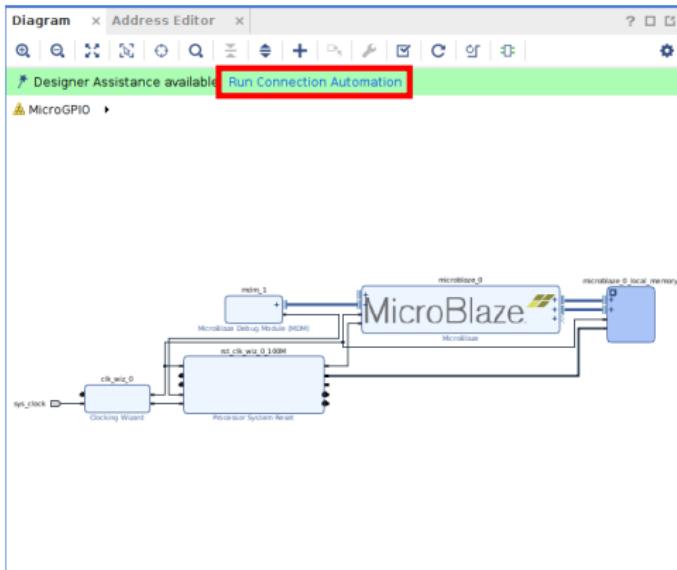
Incluir el core del MicroBlaze y ejecutar “Run Block Automation”.

Creando un nuevo diseño



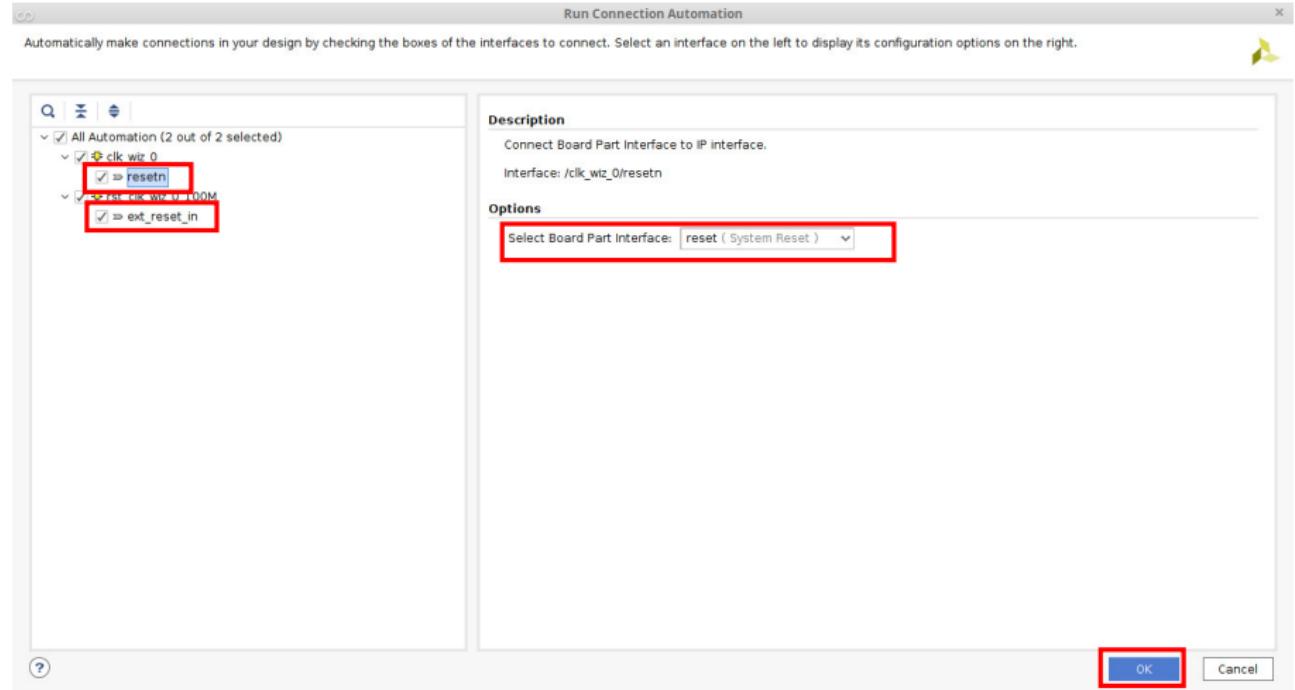
Signar los parámetros de diseño del micro.

Creando un nuevo diseño



Auto-conectar los elementos utilizando “Run Connection Automation”.

Creando un nuevo diseño

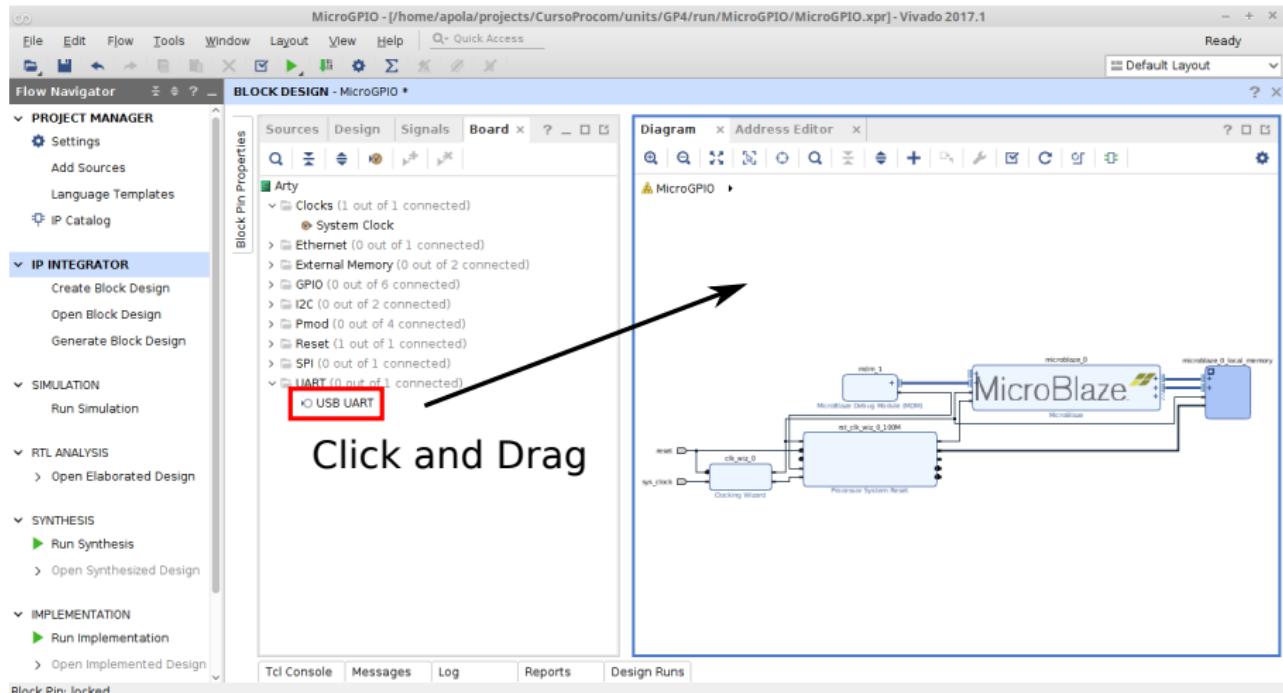


Verificar que los dos puertos de reset esten conectados al reset del sistema.

Instancias de periféricos



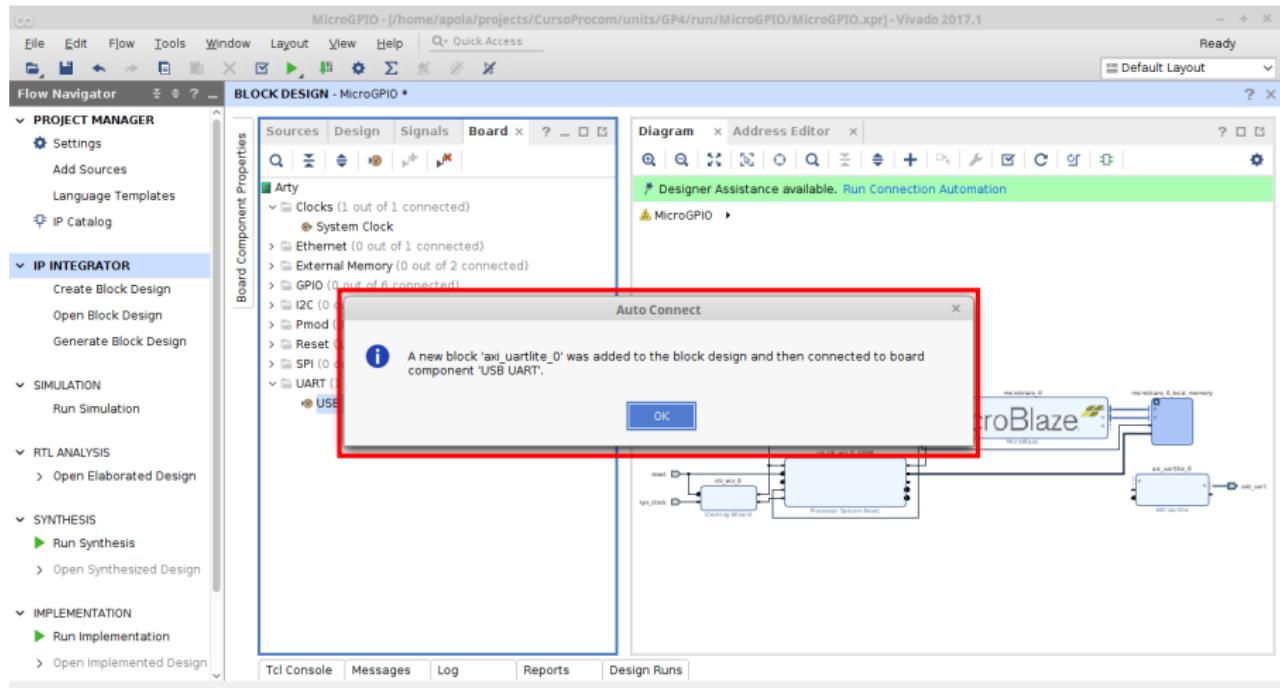
Instancias de periféricos



Click and Drag

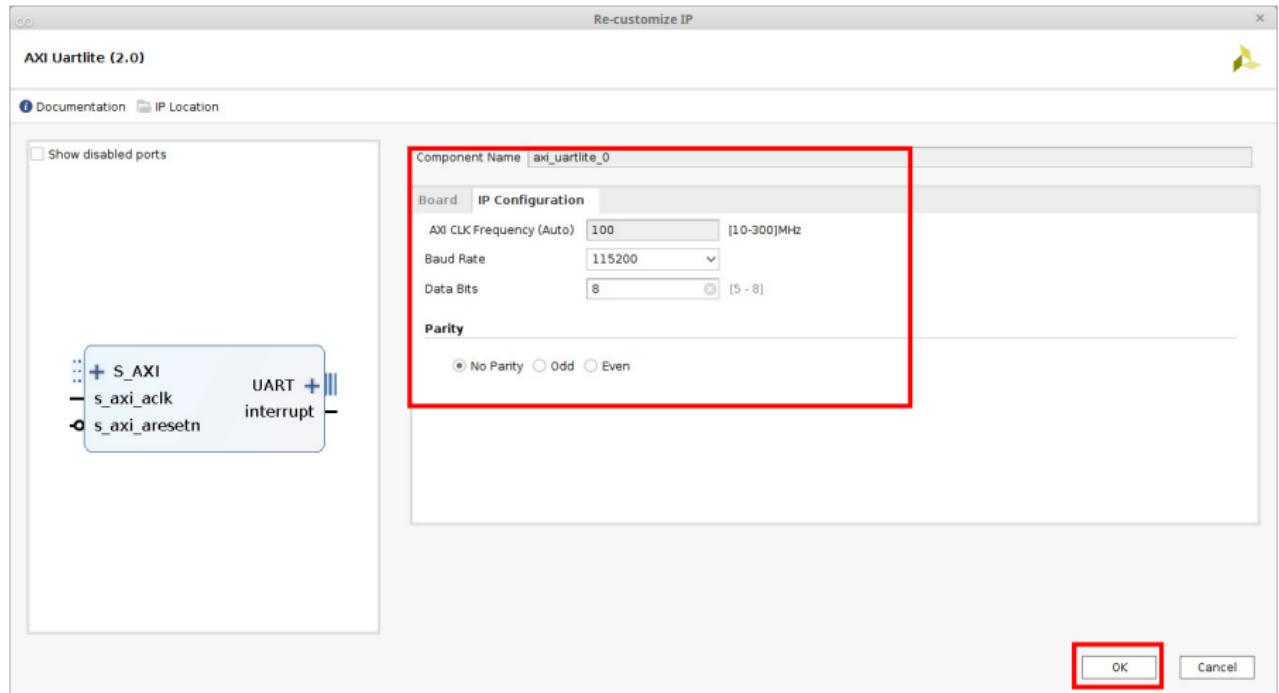
Instanciar el IP USB UART.

Instancias de periféricos



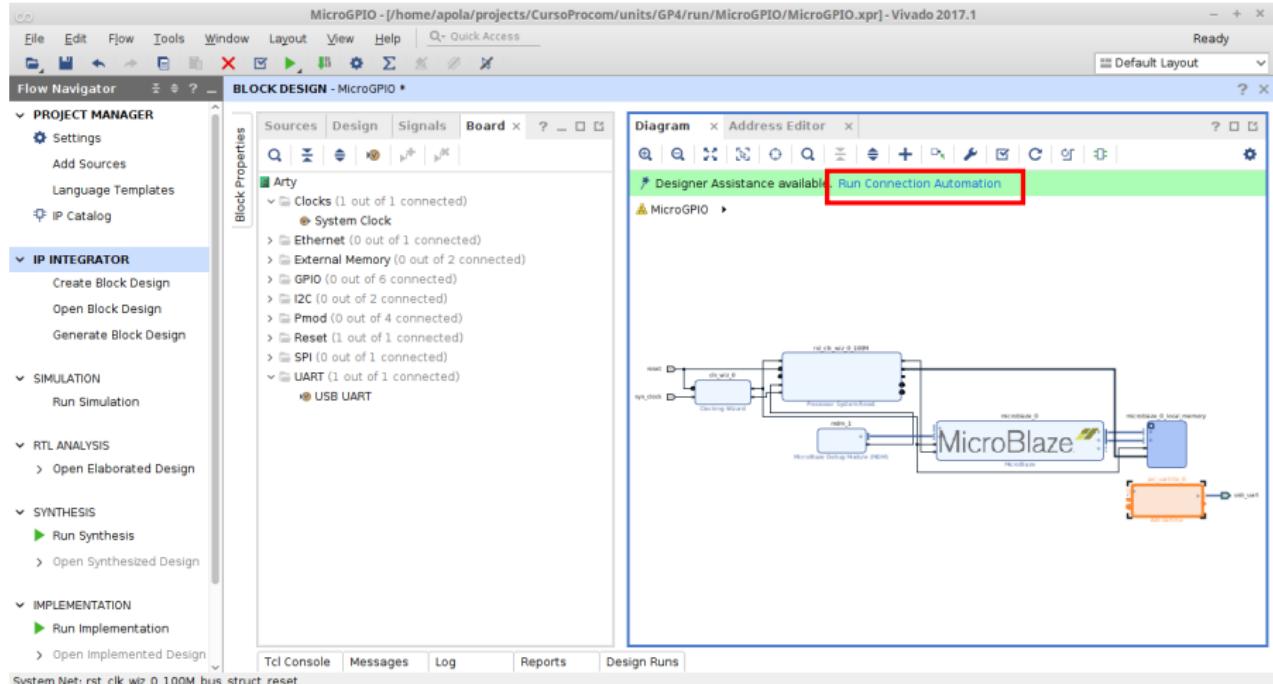
Verificar la instancia.

Instancias de periféricos



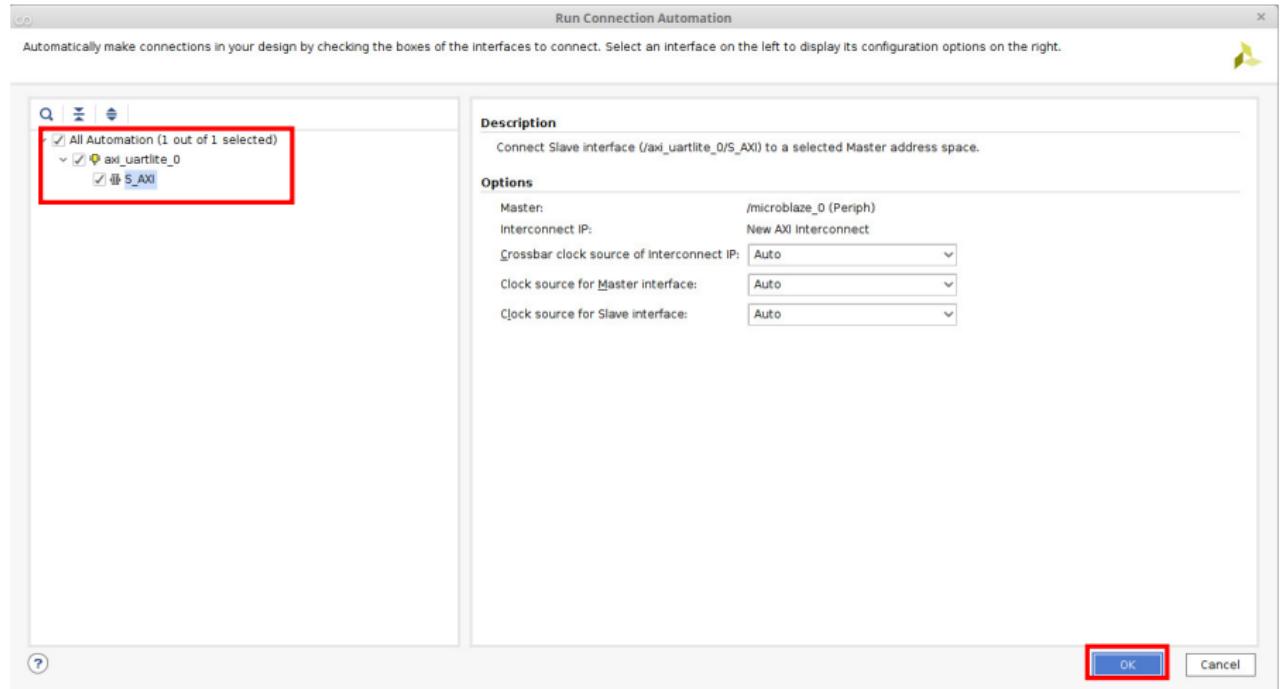
Hacer doble click sobre el periférico *UART* y configurar el *Baud Rate* en 115200.

Instancias de periféricos



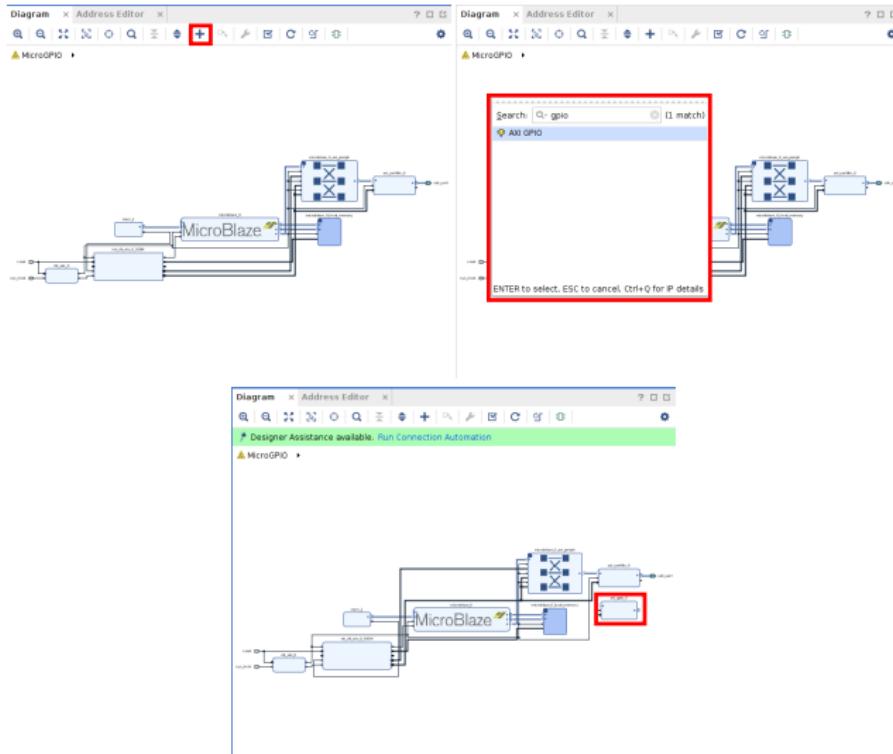
Seleccionar la conexión automática.

Instancias de periféricos



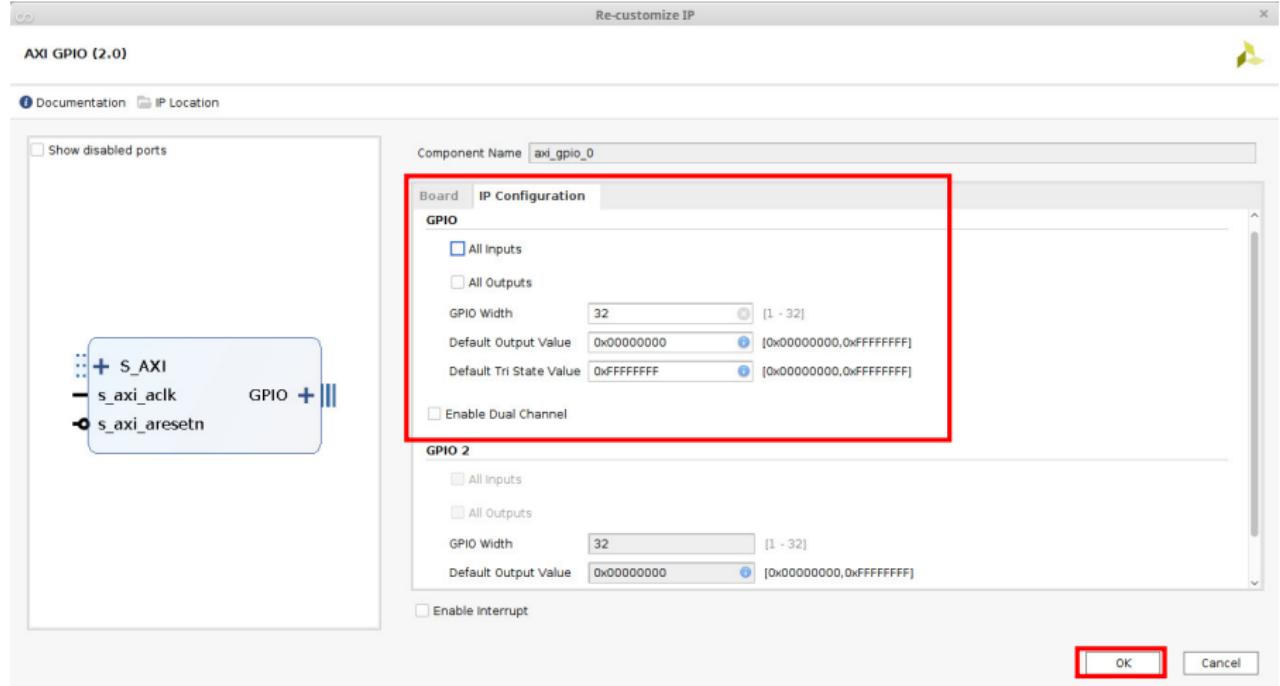
Verificar que todos los puertos estén seleccionados.

Instancias de periféricos



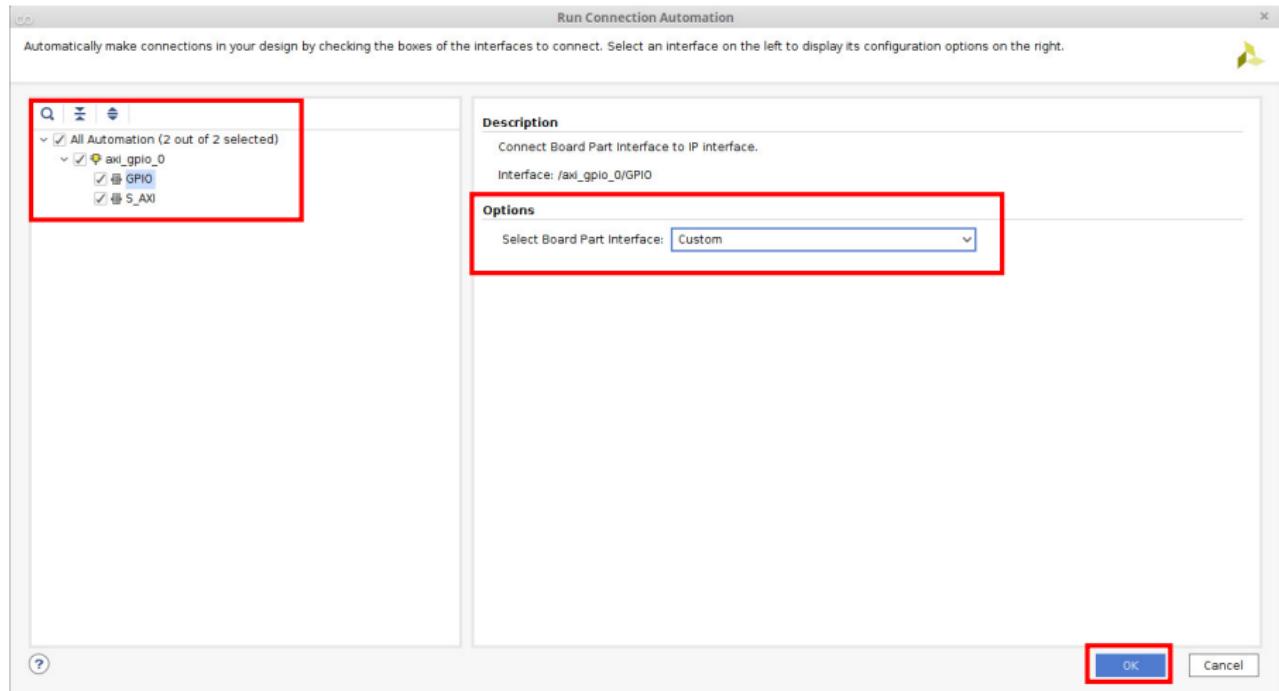
Agregar un nuevo componente GPIO y hacer doble click sobre el IP.

Instancias de periféricos



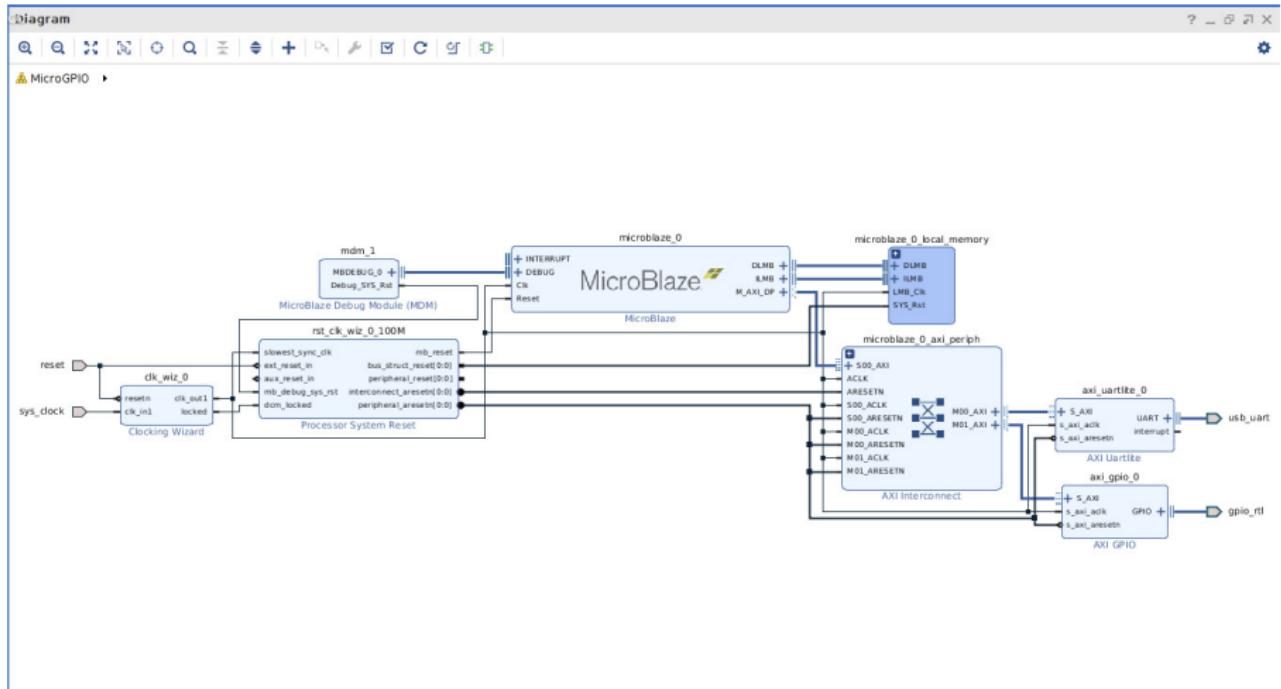
Verificar la configuración del puerto.

Instancias de periféricos



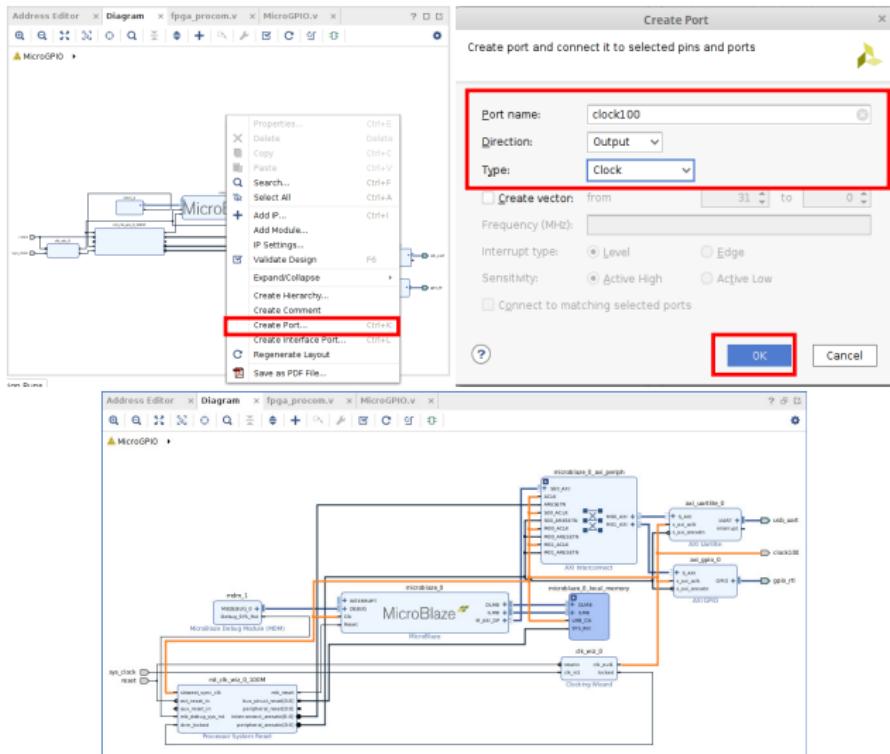
Seleccionar "Run Connection Automation" y verificar que el puerto de salida sea "Custom".

Instancias de periféricos



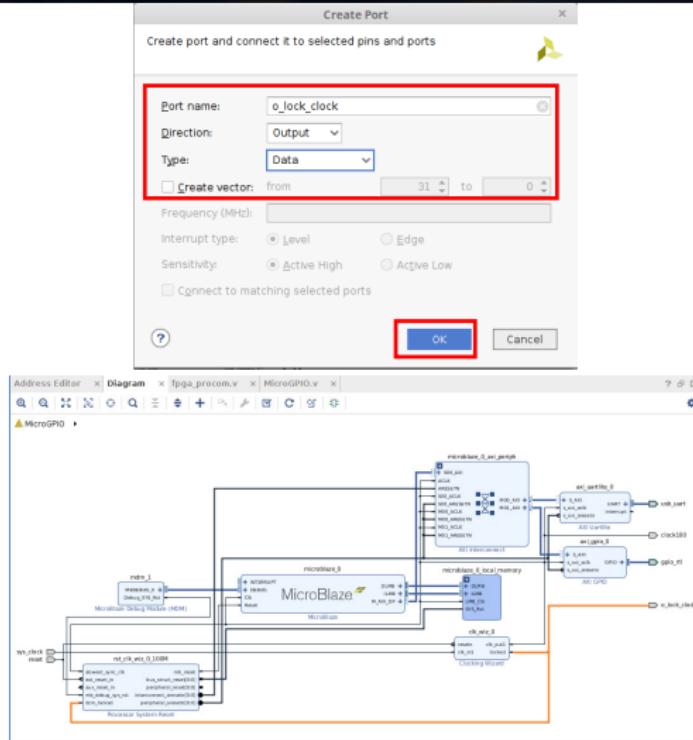
Diseño preliminar.

Instancias de periféricos



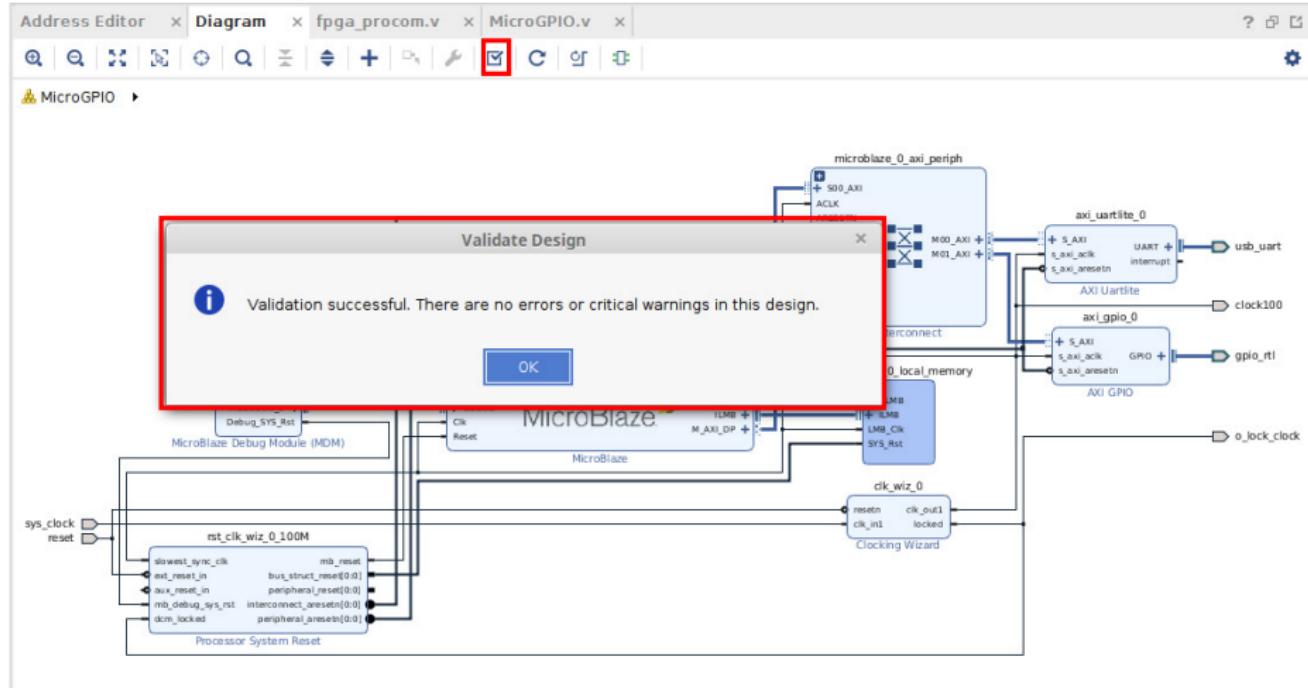
Incluir un nuevo puerto de clock de salida.

Instancias de periféricos



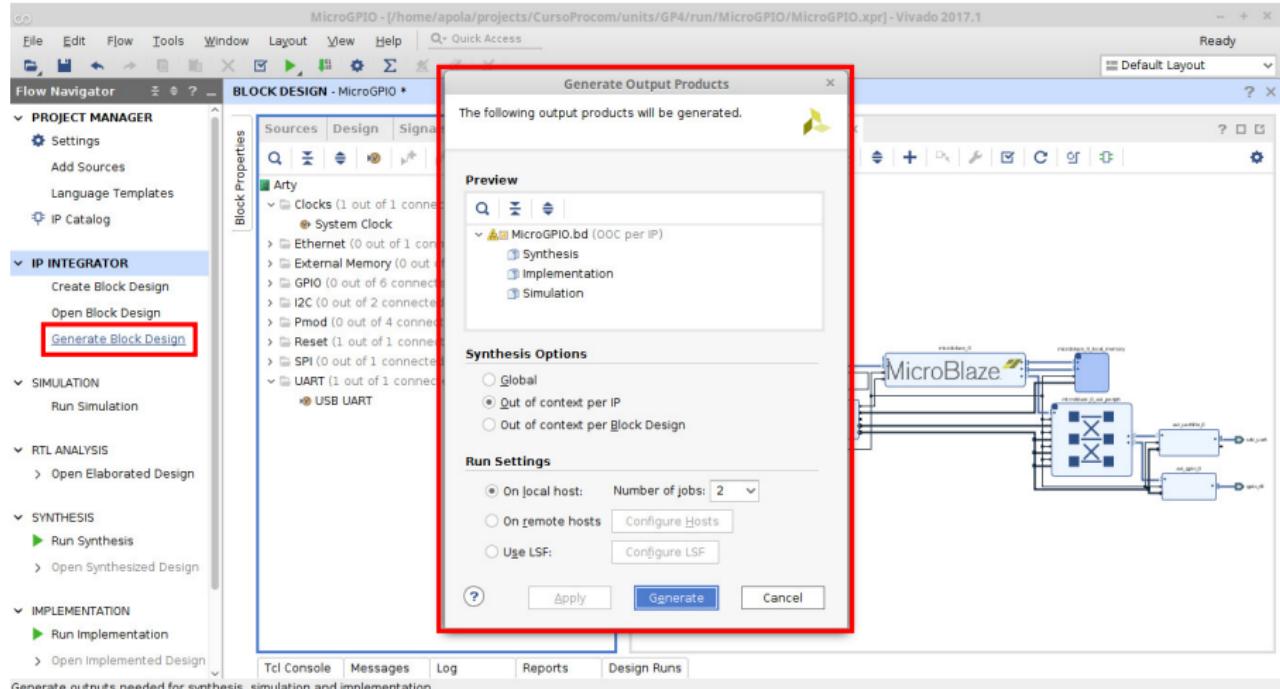
Repetir el paso anterior incluyendo un nuevo puerto de lock de clock.

Instancias de periféricos



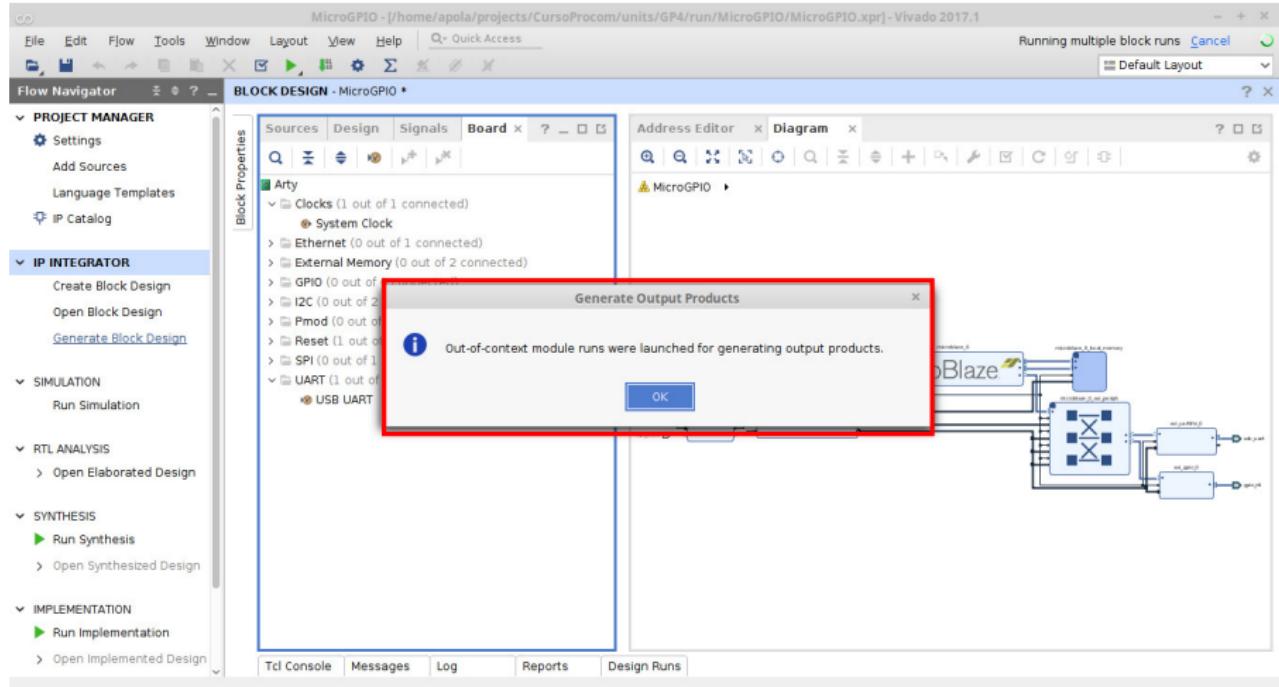
Validar el diseño.

Instancias de periféricos



Generar el diseño completo.

Instancias de periféricos

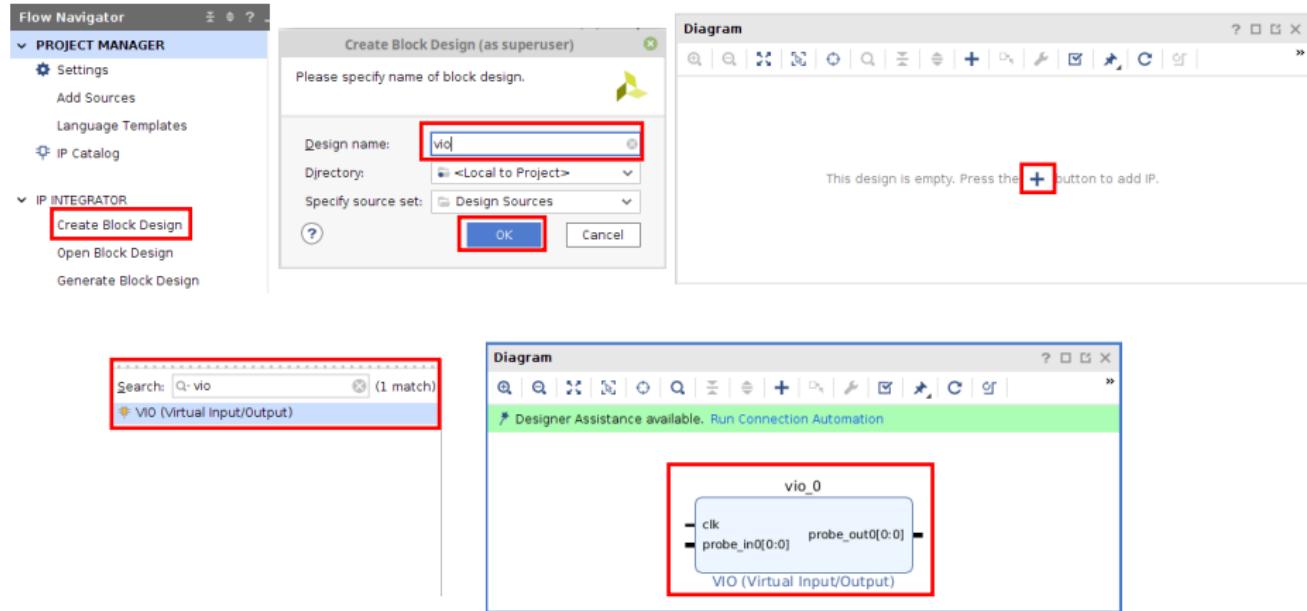


Verificar la elaboración.

Instancia de VIO

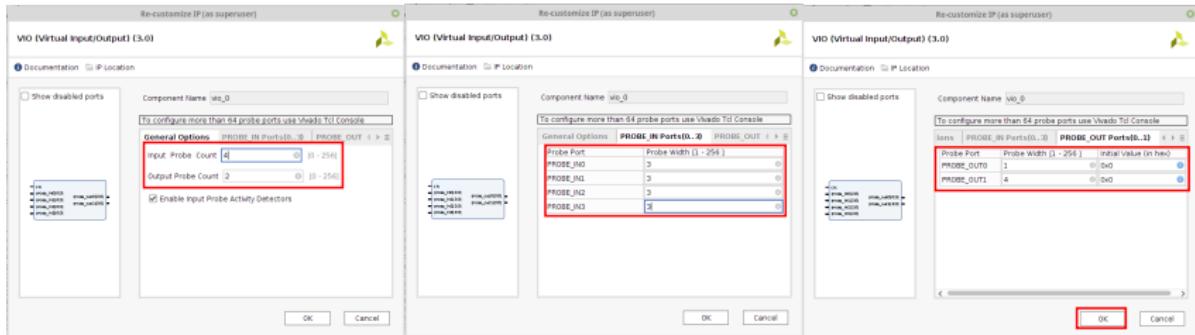


Instancia de VIO



En caso de necesitar controlar en forma remota la FPGA debemos incluir el bloque VIO a nuestro diseño. Creamos un nuevo bloque, le asignamos el nombre, agregamos un nuevo IP, seleccionamos el IP VIO y hacemos doble click sobre el IP generado.

Instancia de VIO



En el ejemplo vamos a habilitar el control externo, controlar los SWITCH y observar el valor de los leds RGB. Agregamos 4 puertos de entrada y 2 de salida. Cada puerto de entrada tiene 3 bits. En el caso de los puertos de salida se tienen 1 y 4 bits respectivamente.

Instancia de VIO

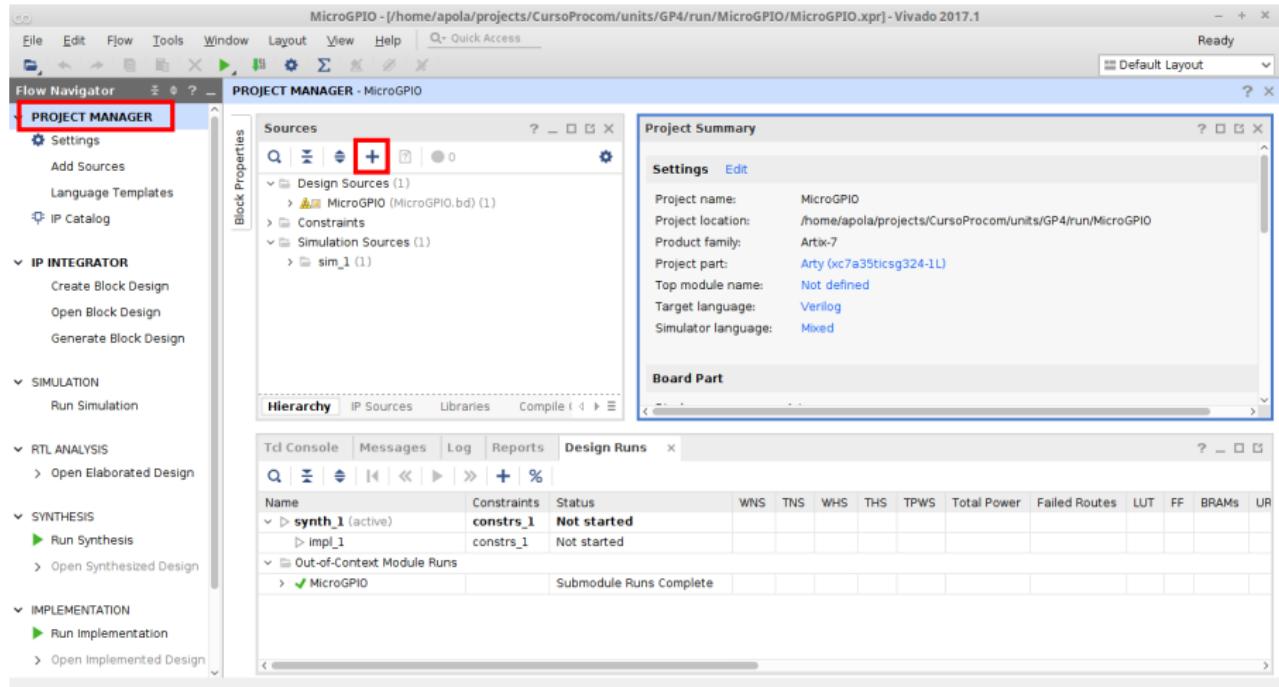
The screenshot shows the Vivado IP Integrator interface. On the left, there are two 'Diagram' windows. The top one displays a block diagram with a red box around a 'vio_0' block. The bottom one shows a similar diagram with more connections. To the right of these is a context menu with several options: 'Search...', 'Select All', 'Add IP...', 'Add Module...', 'Make External' (which is highlighted with a red box), 'Run Connection Automation...', 'Customize Block...', 'IP Documentation', 'Orientation', 'Pinning', 'IP Settings...', and 'Validate Design'. Below this is a 'Preview' window showing the generated output products: 'vio.bd (00C per IP)' with options for 'Synthesis', 'Implementation', 'Simulation', and 'Hw_Handoff'. Further down are sections for 'Synthesis Options' (with 'Out of context per IP' selected) and 'Run Settings' (with 'On local host:' selected, 'Number of jobs: 4', and a 'Generate' button highlighted with a red box).

Sobre el bloque modificado hacemos click con el botón derecho, seleccionamos Make External lo cual agrega los puertos exteriores y por último generamos el bloque.

Instanciar el uP en Top Level

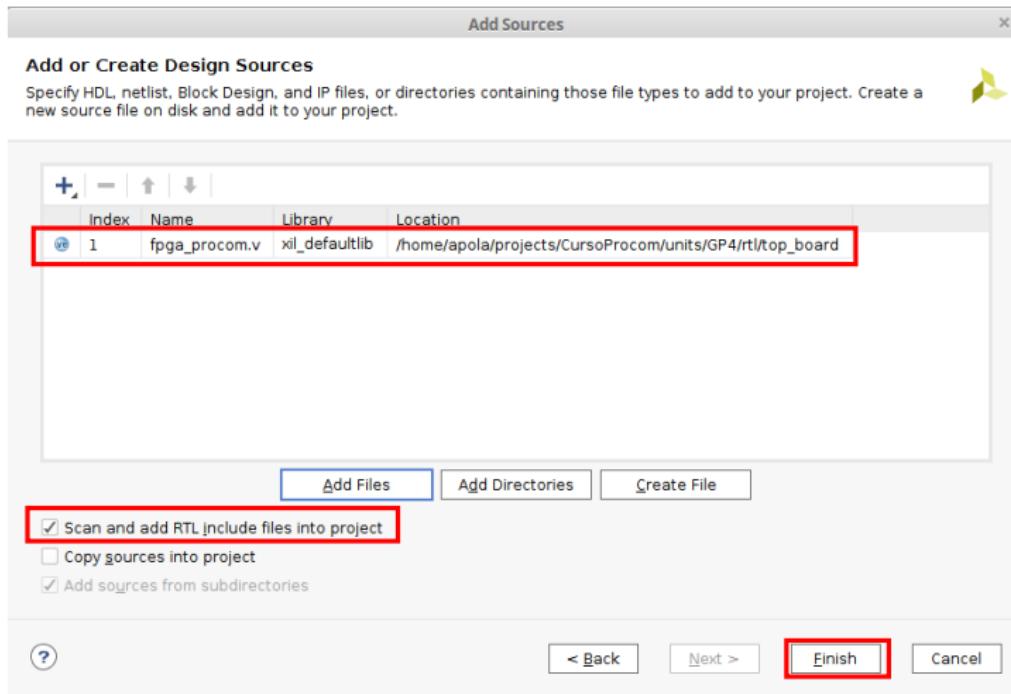


Instanciar el uP en Top Level



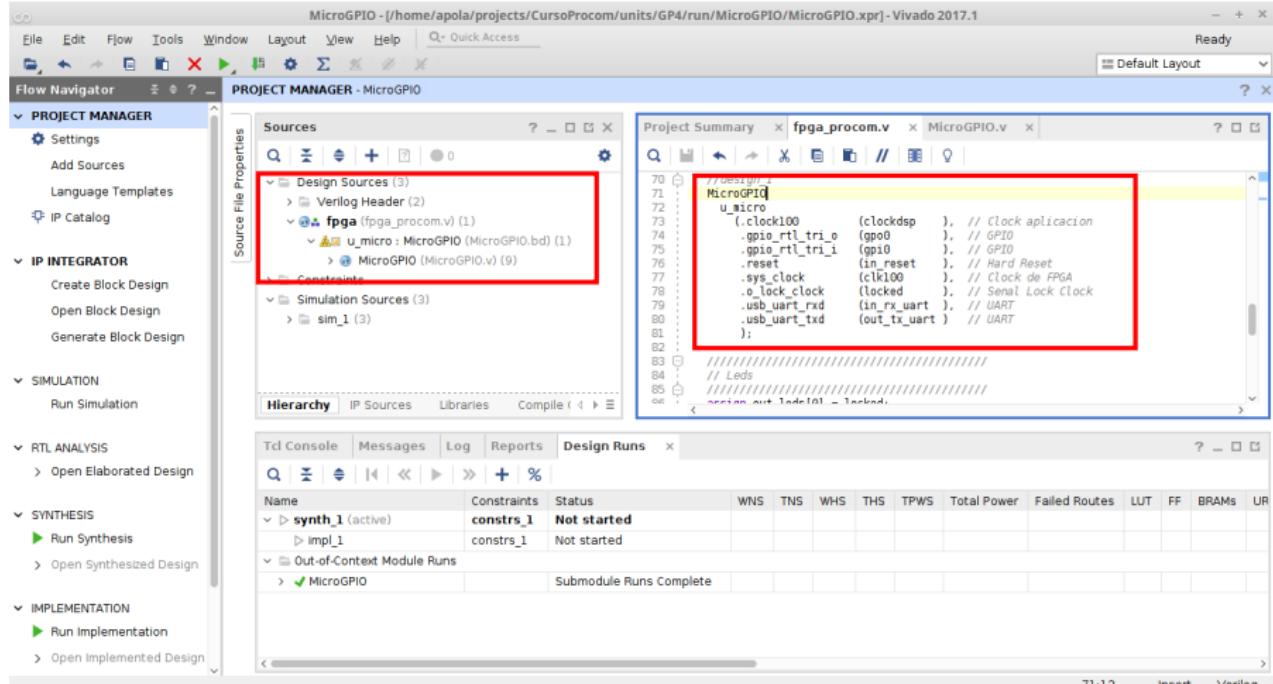
Agregar un nuevo archivo verilog.

Instanciar el uP en Top Level



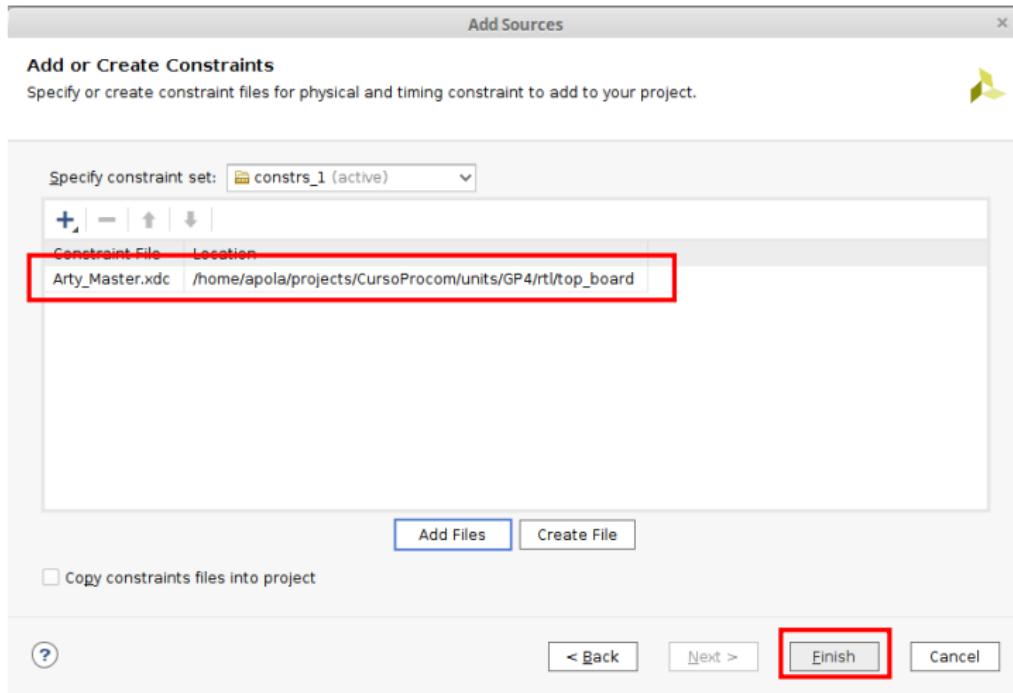
Seleccionar el archivo “fpga_dda” y asegurarse de activar el escaneo de includes.

Instanciar el uP en Top Level



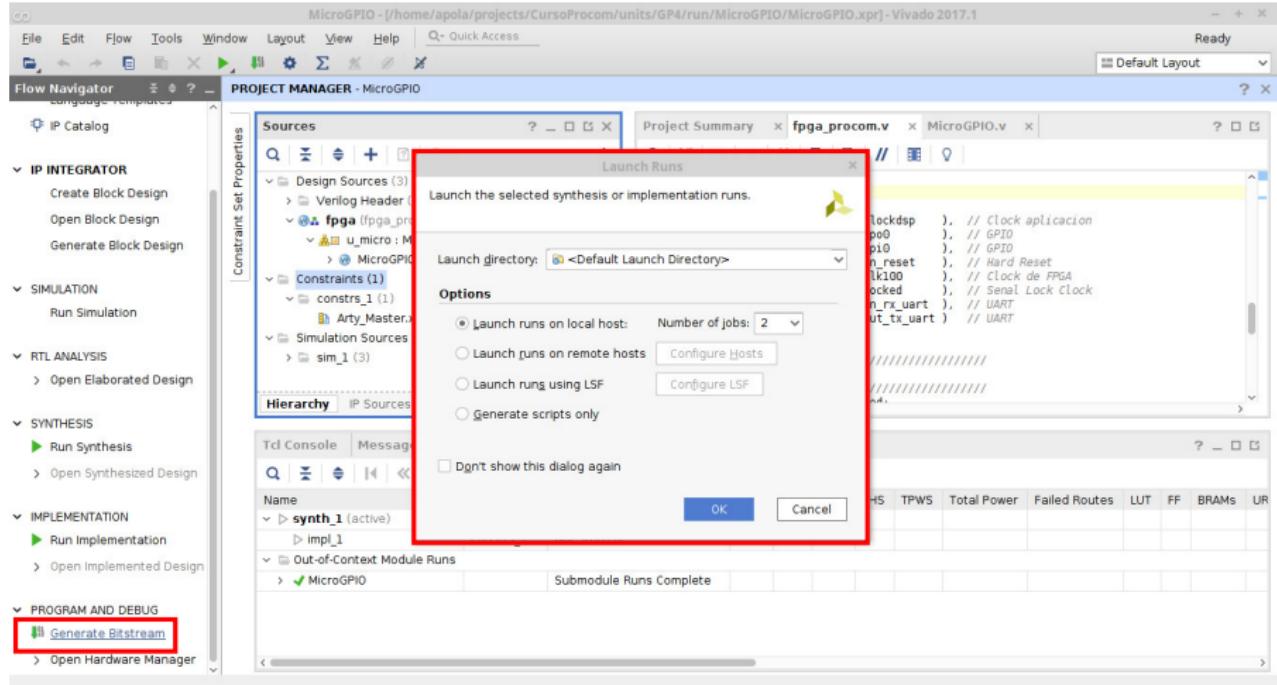
Verificar que el nombre del archivo del micro sea el mismo que se utiliza en el top level y los mismos puertos.

Instanciar el uP en Top Level



Agregar el archivo de puertos físicos.

Instanciar el uP en Top Level



Generar el binario.

Instanciar el VIO en Top Level



Instanciar el VIO en Top Level

- En caso que se esté trabajando en forma remota, vamos a necesitar incluir el VIO para controlar los switch y observar los valores de los puertos de salida.

```
Diagram x vio.v x fpga_procom.v * x
/home/apola/projects/curso/rtl/top_board/fpga_procom.v
45 //////////////////////////////////////////////////////////////////
46 // Vars
47 //
48 wire [NB_GPIO5 : 0] gpo0;
49 wire [NB_GPIO5 : 0] gpio;
50
51 wire locked;
52
53 wire soft_reset;
54 wire clock;
55 wire fromHard;
56 wire [3 : 0] fromVio;
57 :
```

Declarar en el módulo Top dos variables wire de 1 y 4 bits respectivamente. La primera nos permitirá seleccionar de donde queremos controlar el dispositivo y la segunda es para emular el comportamiento de los switch en forma virtual.

Instanciar el VIO en Top Level

The screenshot displays the Vivado IDE interface with three main windows:

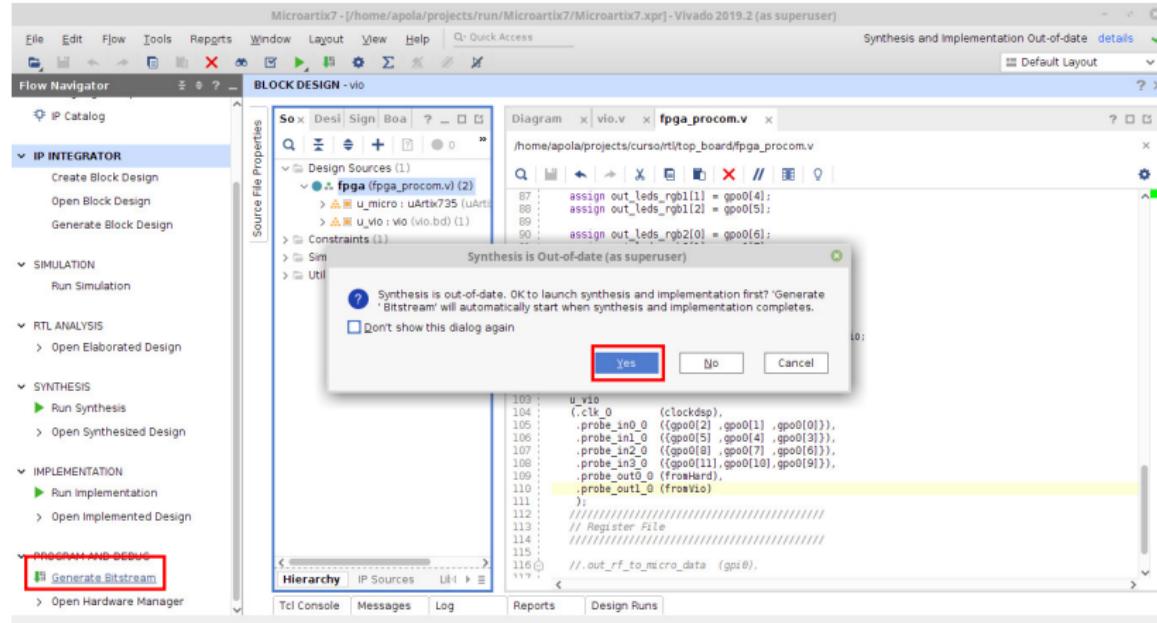
- Sources:** Shows the project structure with "Design Sources" containing "fpga (fpga_procom.v)" and "vio (vio.bd)".
- Diagram:** Shows the Verilog code for "vio.v". A red box highlights the module declaration:

```
module vio
  (clk_0,
   probe_in0_0,
   probe_in1_0,
   probe_in2_0,
   probe_in3_0,
   probe_out0_0,
   probe_out1_0);
```
- Hierarchy:** Shows the Verilog code for "fpga_procom.v". A red box highlights the instantiation of the "vio" module:

```
U_vio
  (.clk_0          (clockdsp),
   .probe_in0_0   ({gpo0[2], gpo0[1], gpo0[0]}),
   .probe_in1_0   ({gpo0[5], gpo0[4], gpo0[3]}),
   .probe_in2_0   ({gpo0[8], gpo0[7], gpo0[6]}),
   .probe_in3_0   ({gpo0[11], gpo0[10], gpo0[9]}),
   .probe_out0_0  (fromHard),
   .probe_out1_0  (fromVio)
  );
```

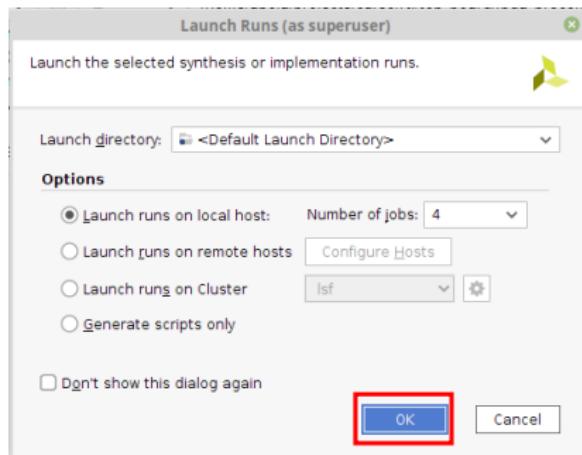
Además, debemos copiar la declaración del módulo VIO e incluirlo en el Top Level. Por último, asignamos la selección del control por medio del bloque VIO y comentamos la asignación directa de los switch.

Instanciar el VIO en Top Level



Generamos el binario nuevamente.

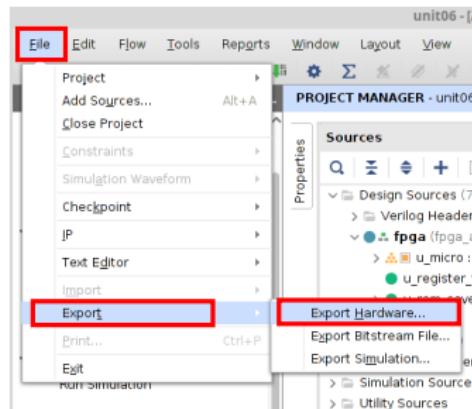
Instanciar el VIO en Top Level



Configuramos cuantos cores queremos utilizar para generar el binario.

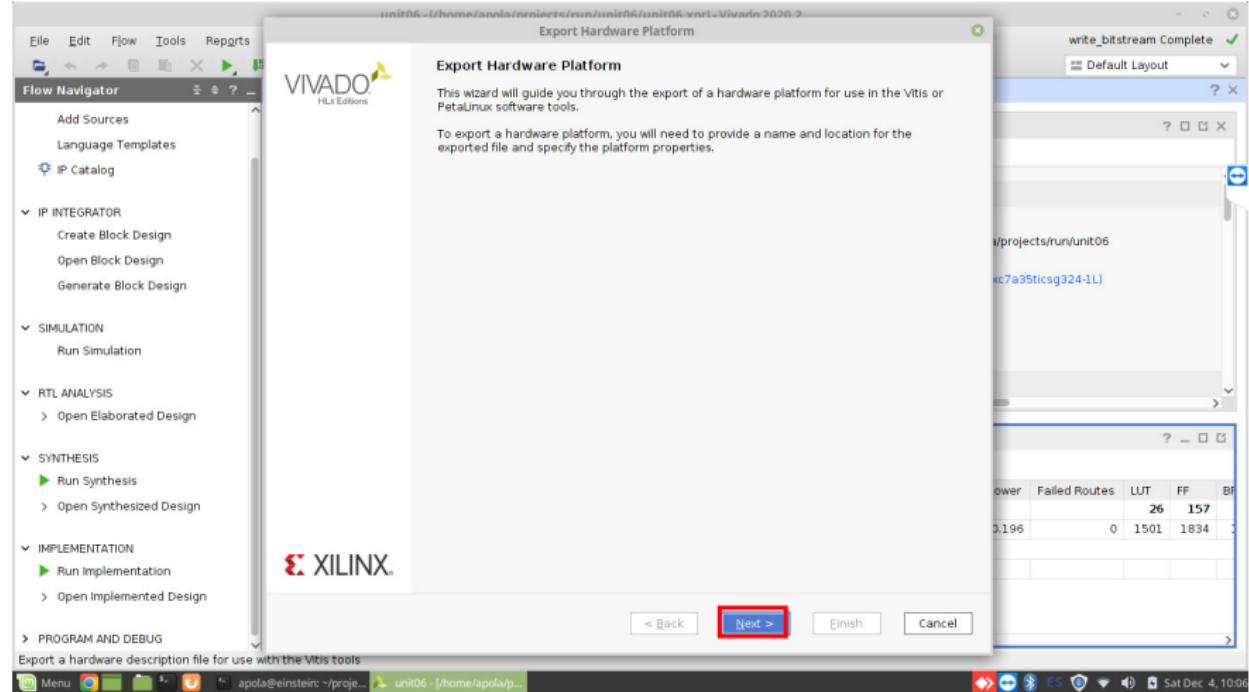
Crear la aplicación en Vitis

Crear la aplicación en Vitis



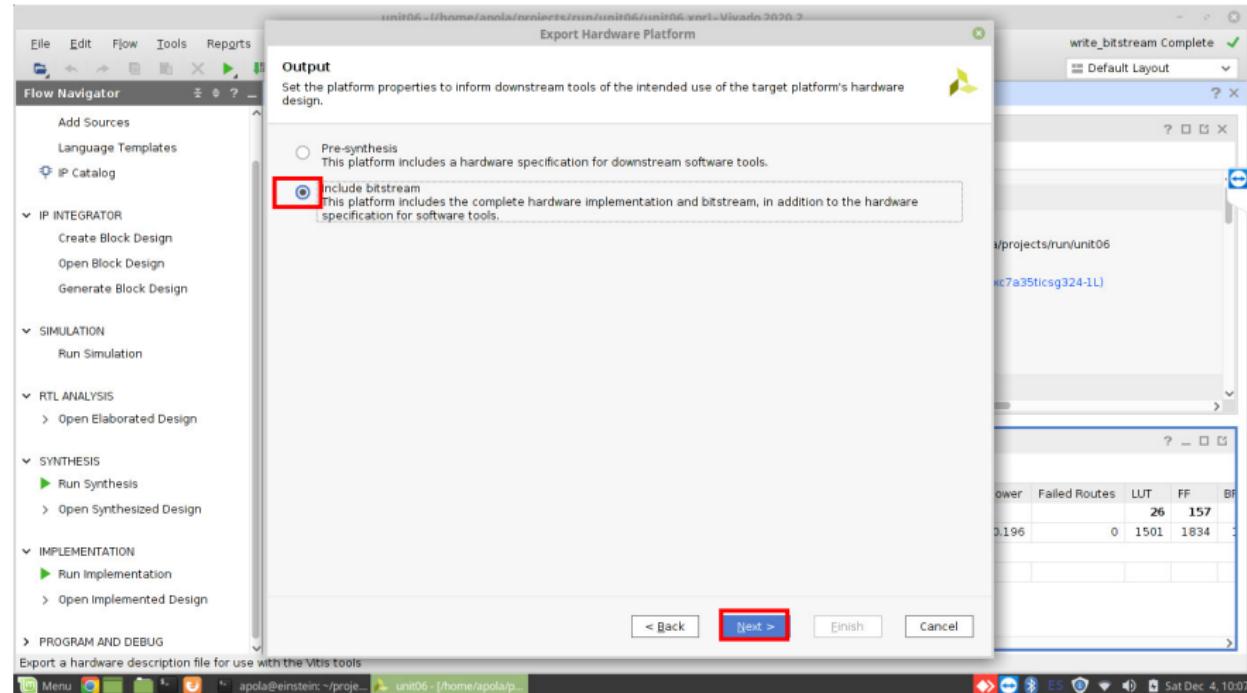
Exportar el hardware desde la opción “File” de la barra de herramientas.

Crear la aplicación en Vitis



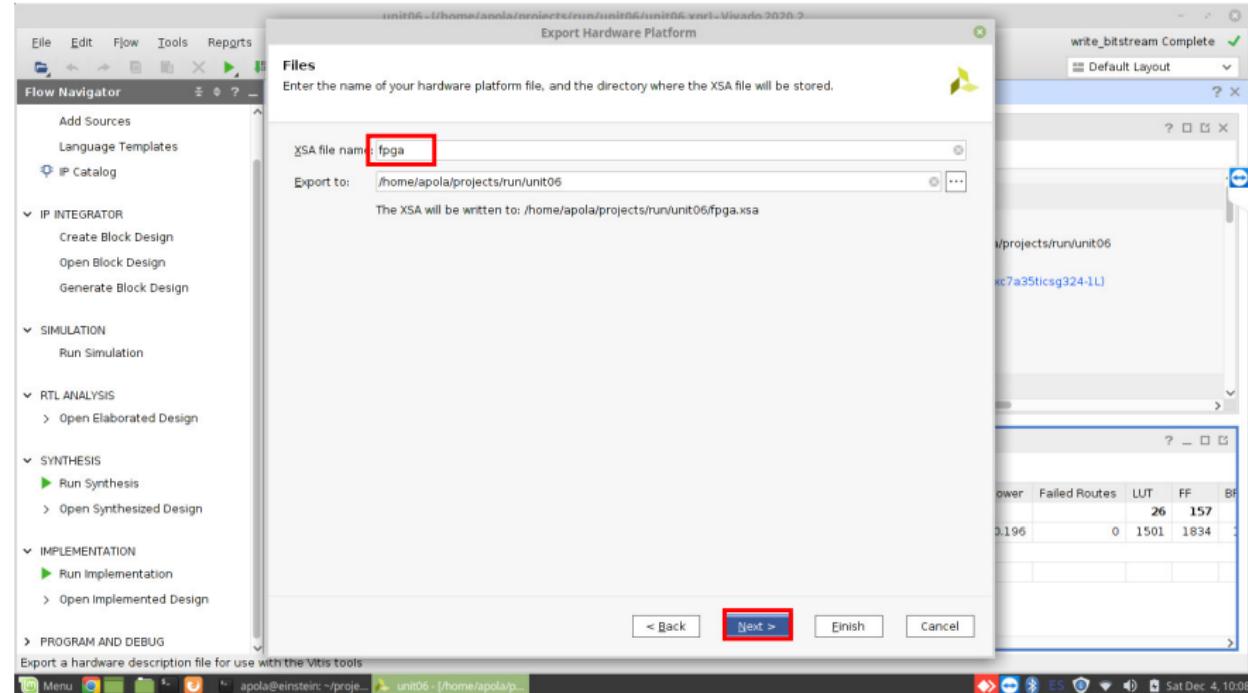
Pulsar NEXT.

Crear la aplicación en Vitis



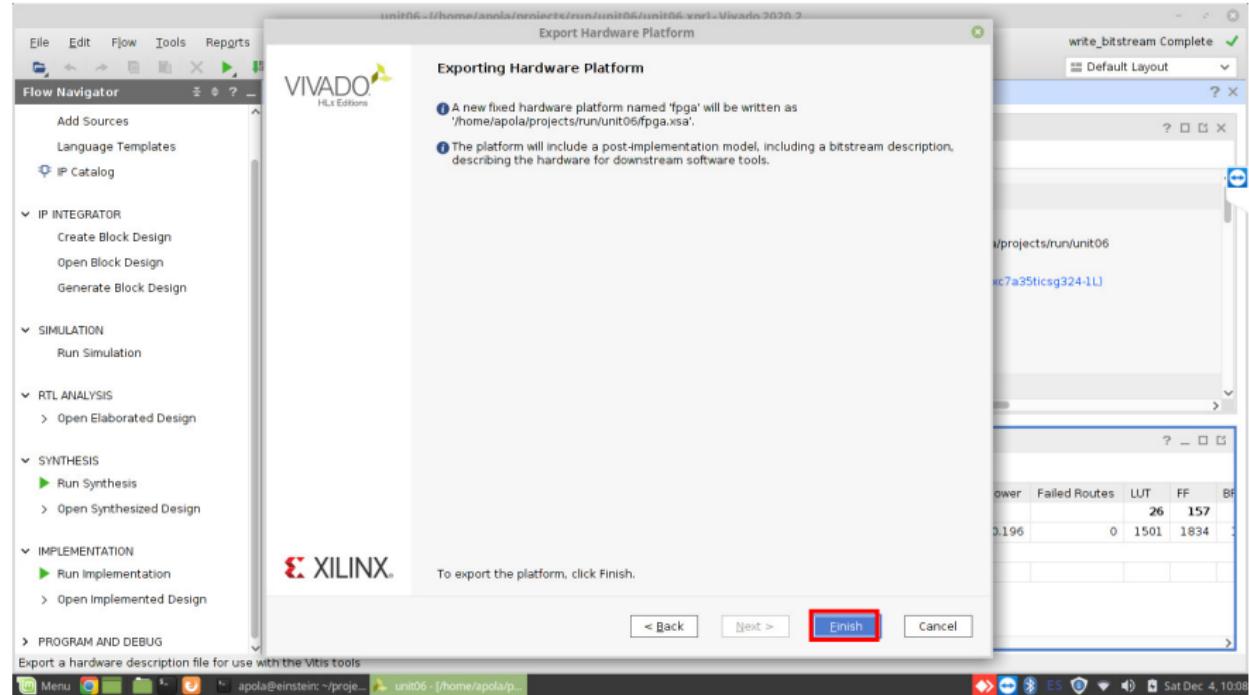
Incluir el bitstream.

Crear la aplicación en Vitis



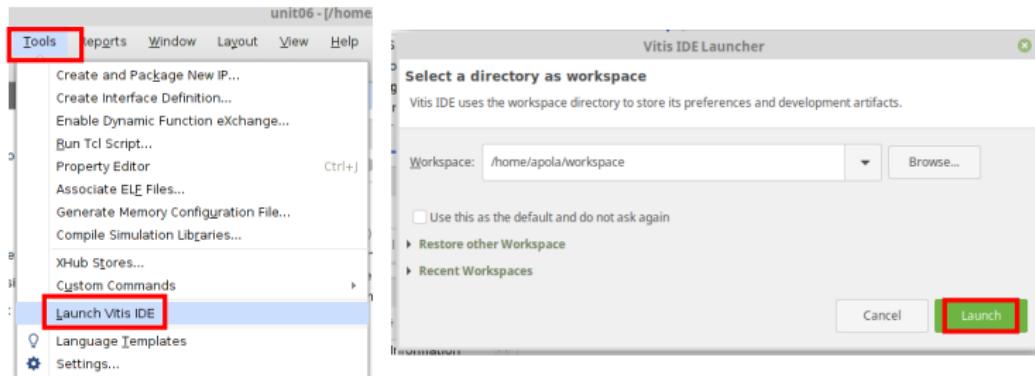
Definir el nombre del XSA (dejarlo por defecto).

Crear la aplicación en Vitis



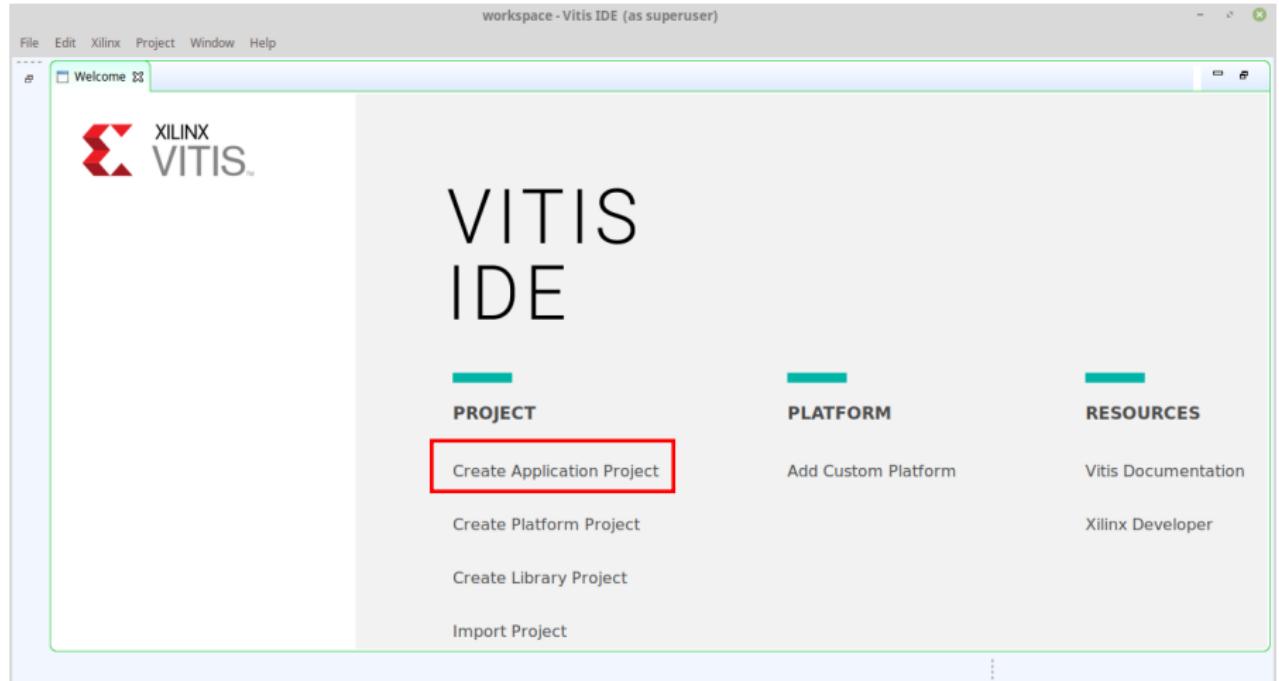
Pulsar FINISH.

Crear la aplicación en Vitis



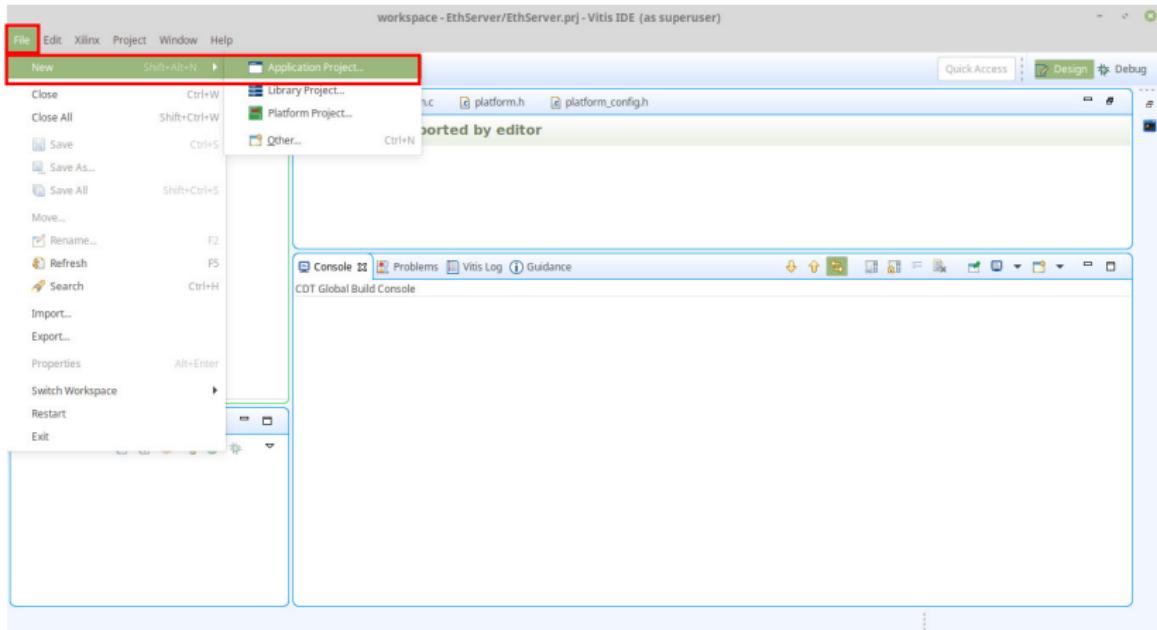
Ejecutar Vitis desde Vivado, seleccionando la opción Launch Vitis IDE.

Crear la aplicación en Vitis



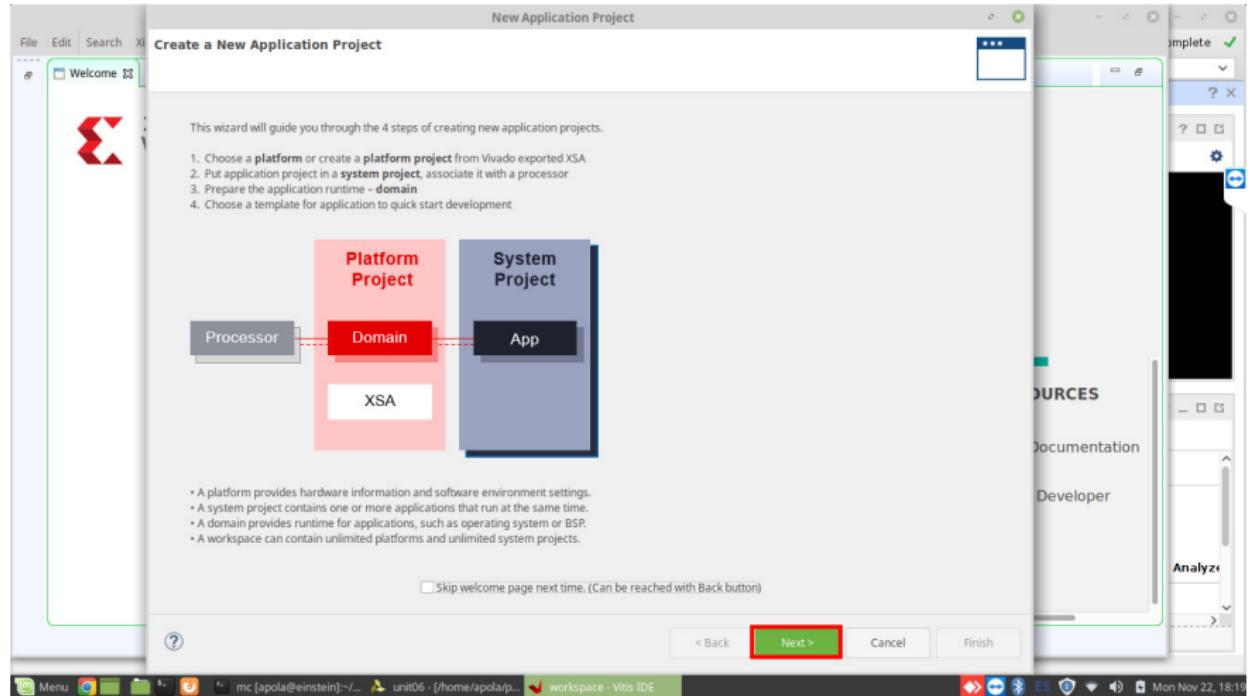
Entorno de trabajo en Vitis. Opción de inicio I: En la pagina de inicio seleccionar Create Application Project.

Crear la aplicación en Vitis



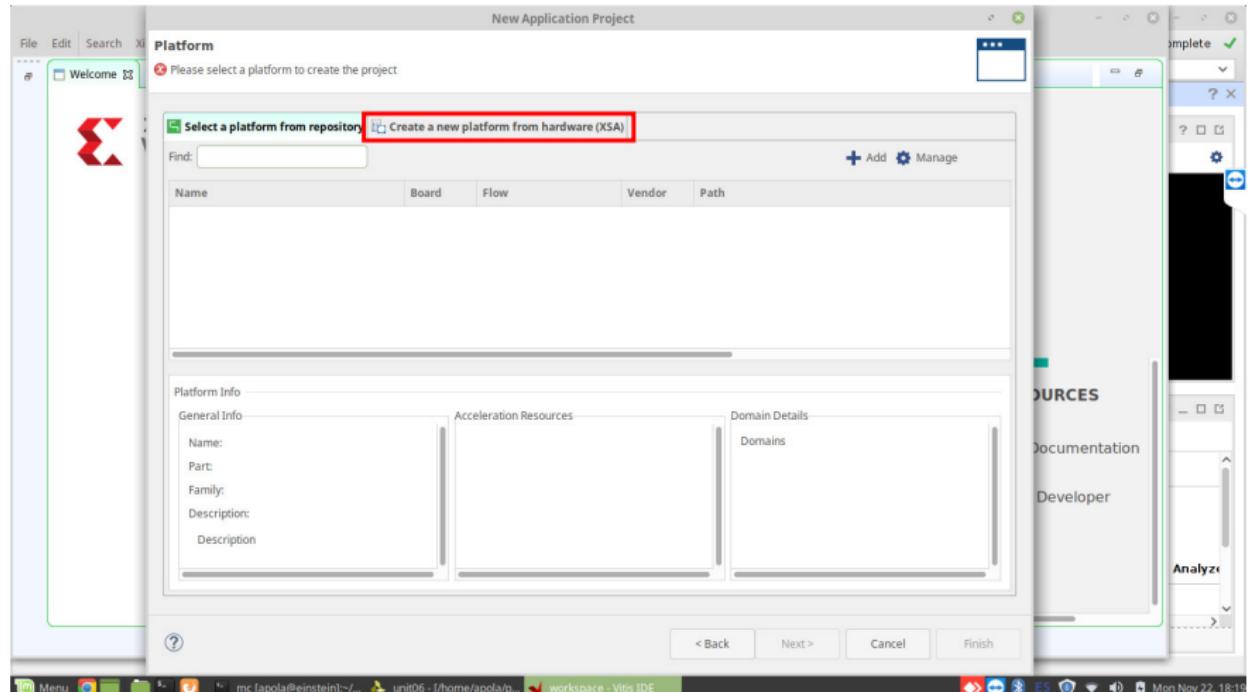
Entorno de trabajo en Vitis. Opción de inicio II: También puede iniciarse la nueva aplicación desde File-New-Application Project.

Crear la aplicación en Vitis



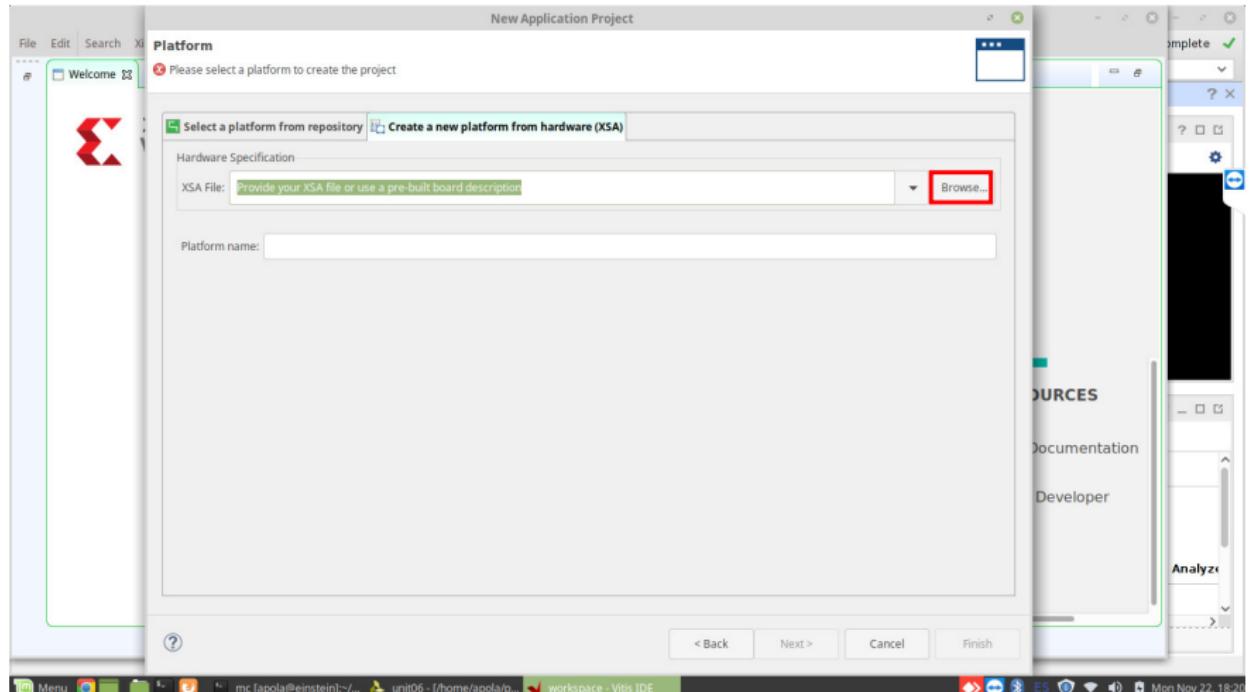
Crear un nuevo proyecto.

Crear la aplicación en Vitis



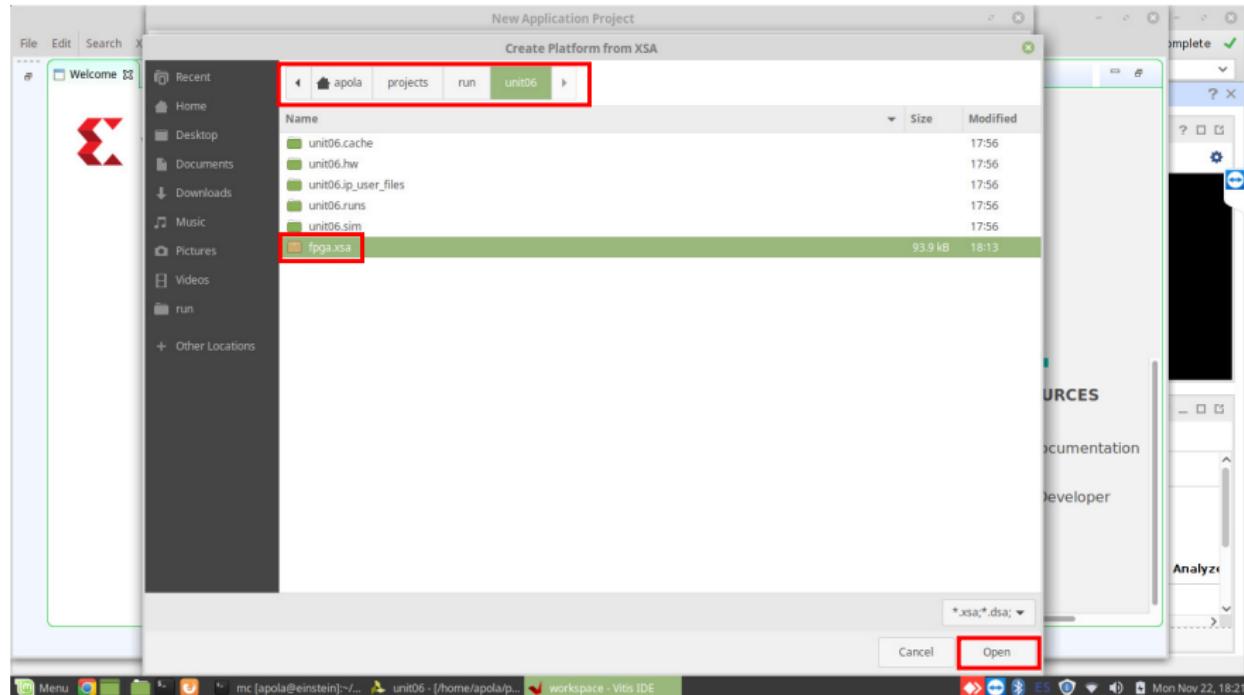
Crear una nueva plataforma importando el archivo XSA generado con el Vivado.

Crear la aplicación en Vitis



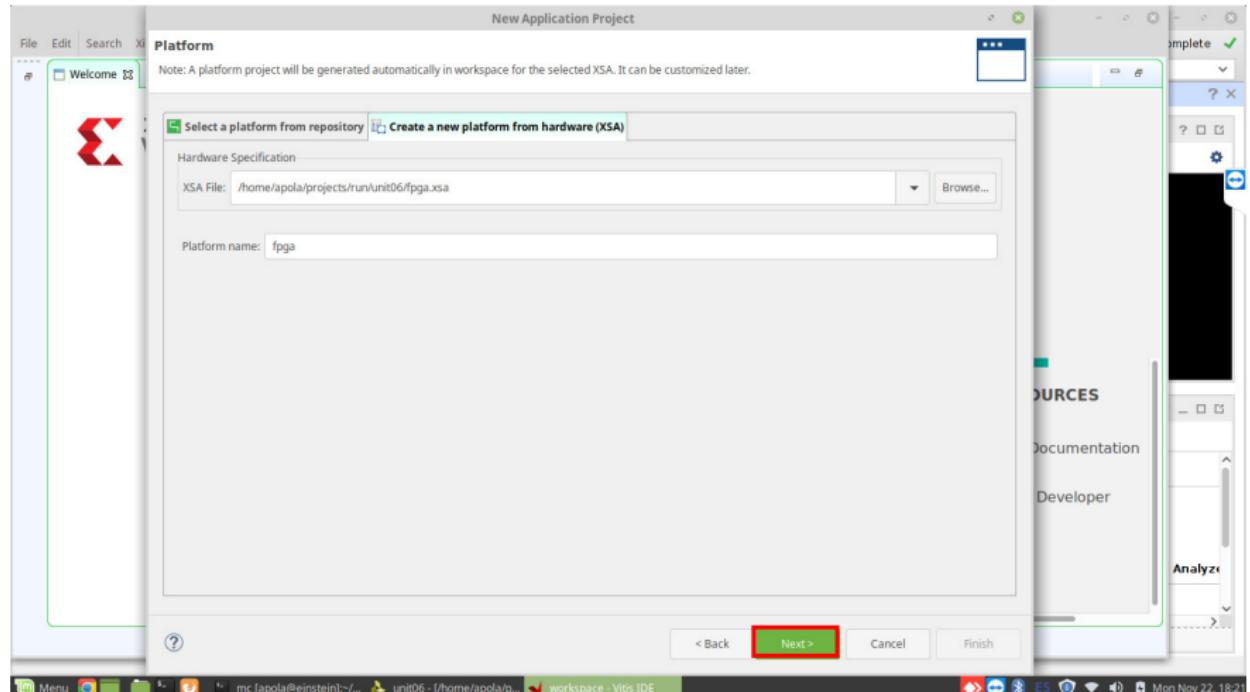
Crear una nueva plataforma importando el archivo XSA generado con el Vivado.

Crear la aplicación en Vitis



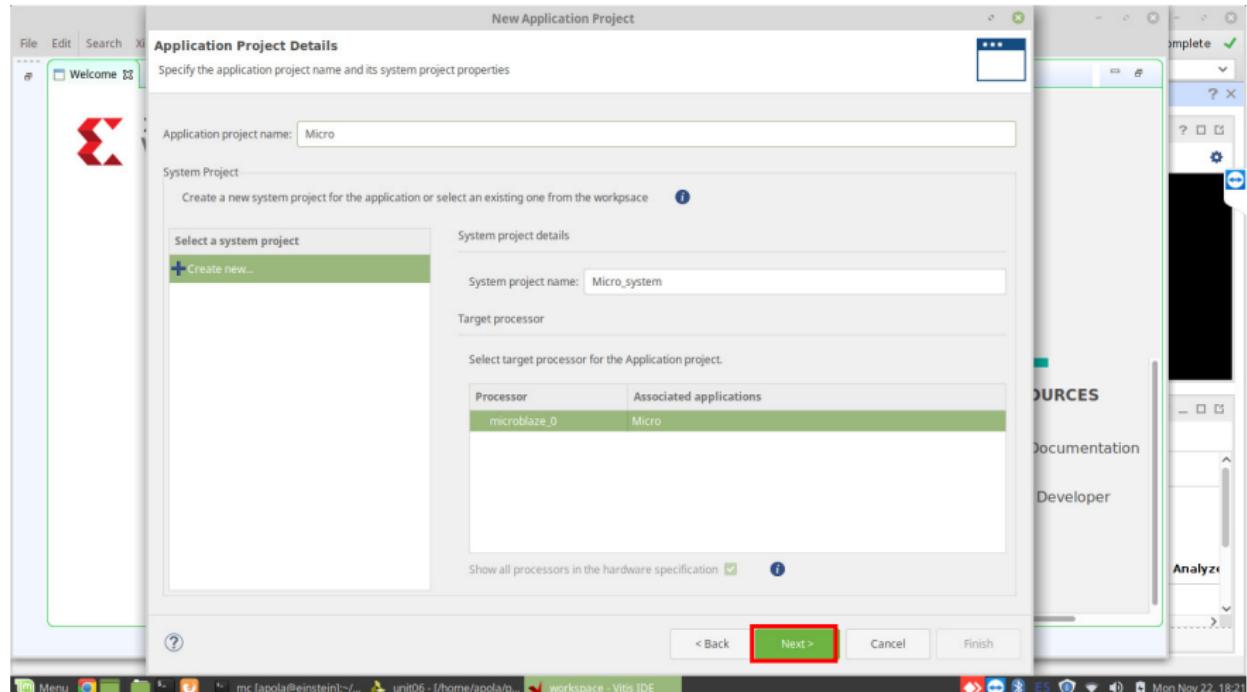
Crear una nueva plataforma importando el archivo XSA generado con el Vivado.

Crear la aplicación en Vitis



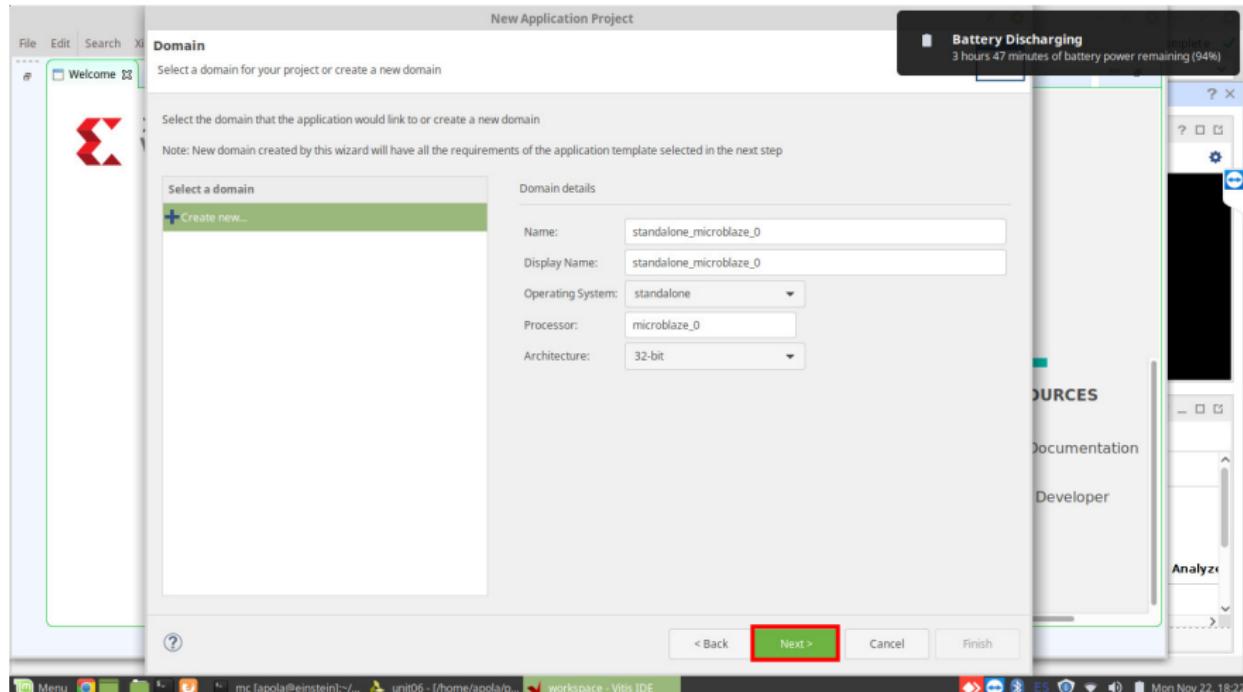
Crear una nueva plataforma importando el archivo XSA generado con el Vivado.

Crear la aplicación en Vitis



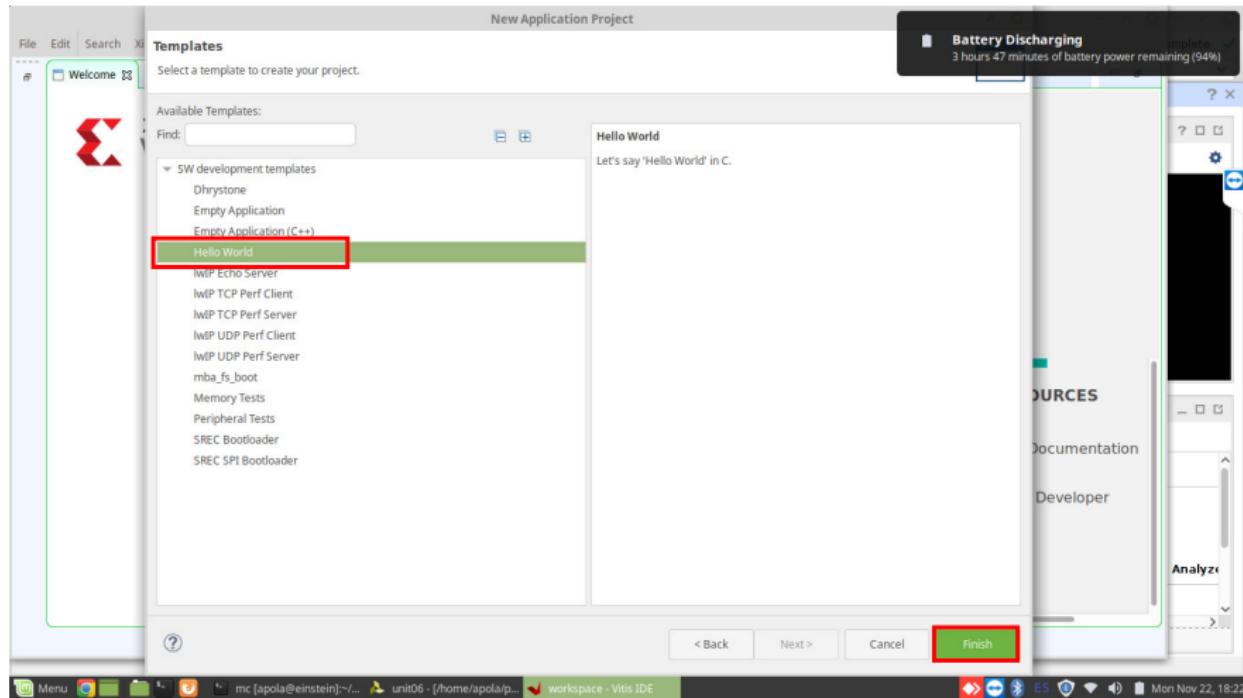
Asignar el nombre de la Aplicación (Ej. Micro).

Crear la aplicación en Vitis



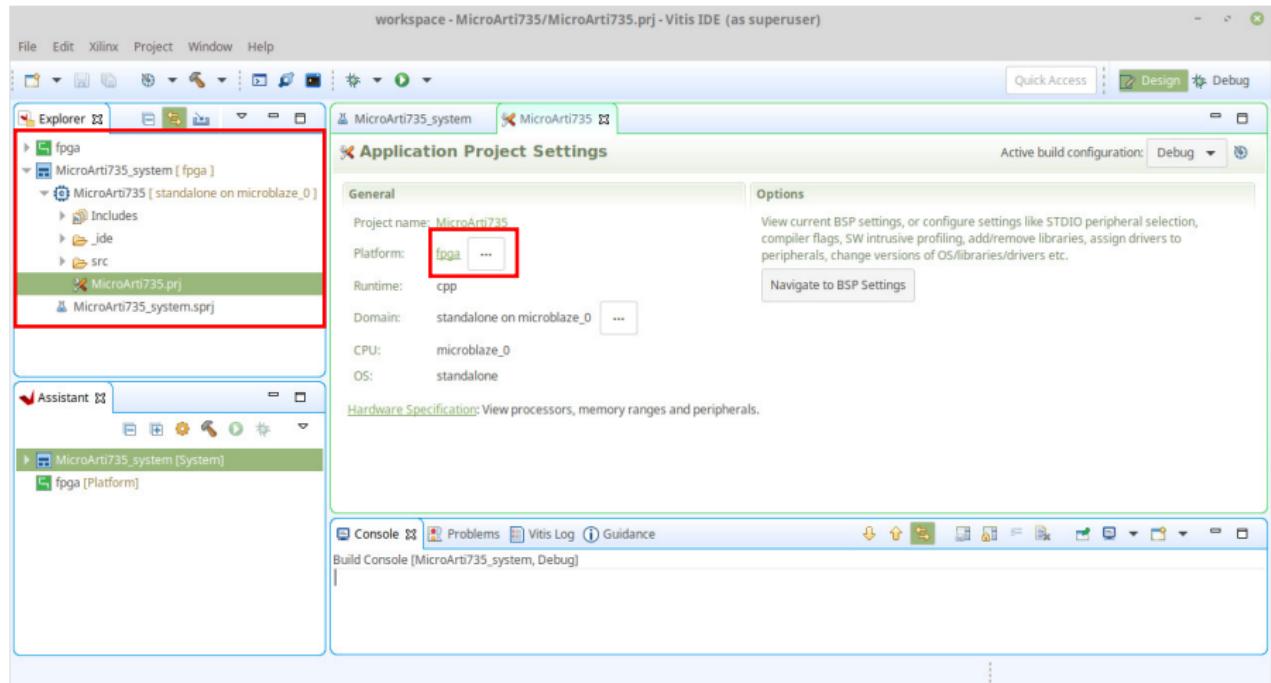
Verificar las características del Micro.

Crear la aplicación en Vitis



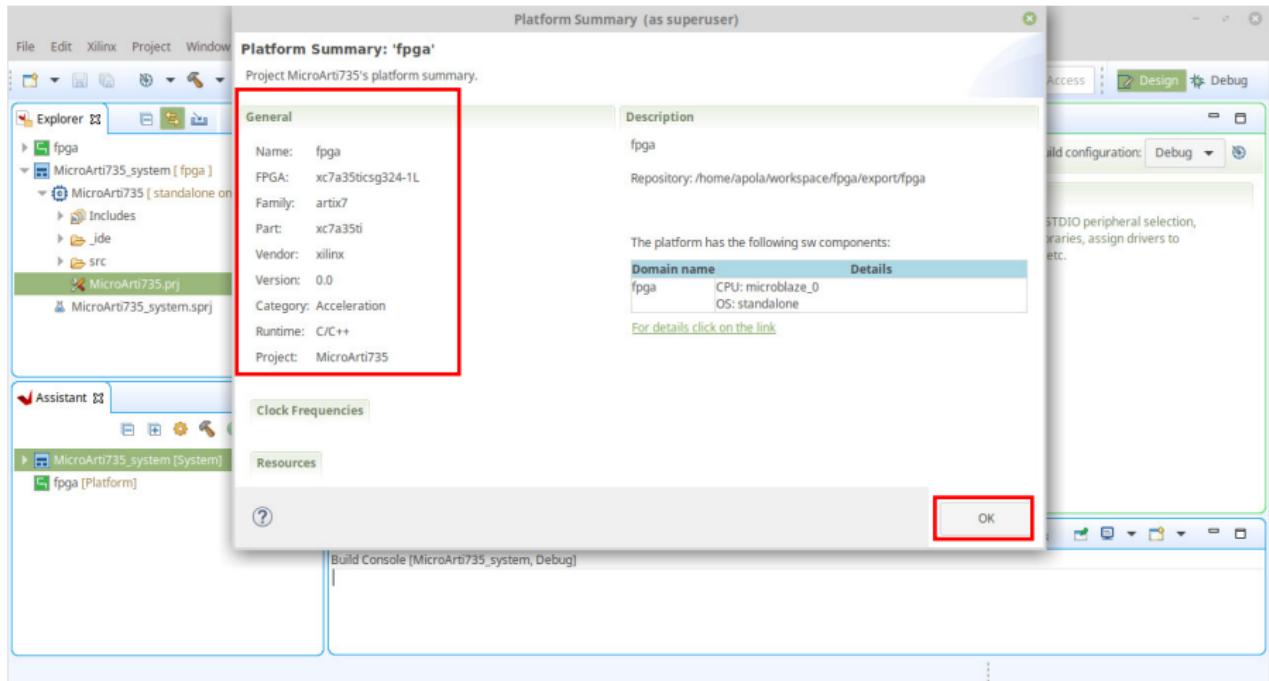
Asignar el proyecto Hello Word.

Crear la aplicación en Vitis



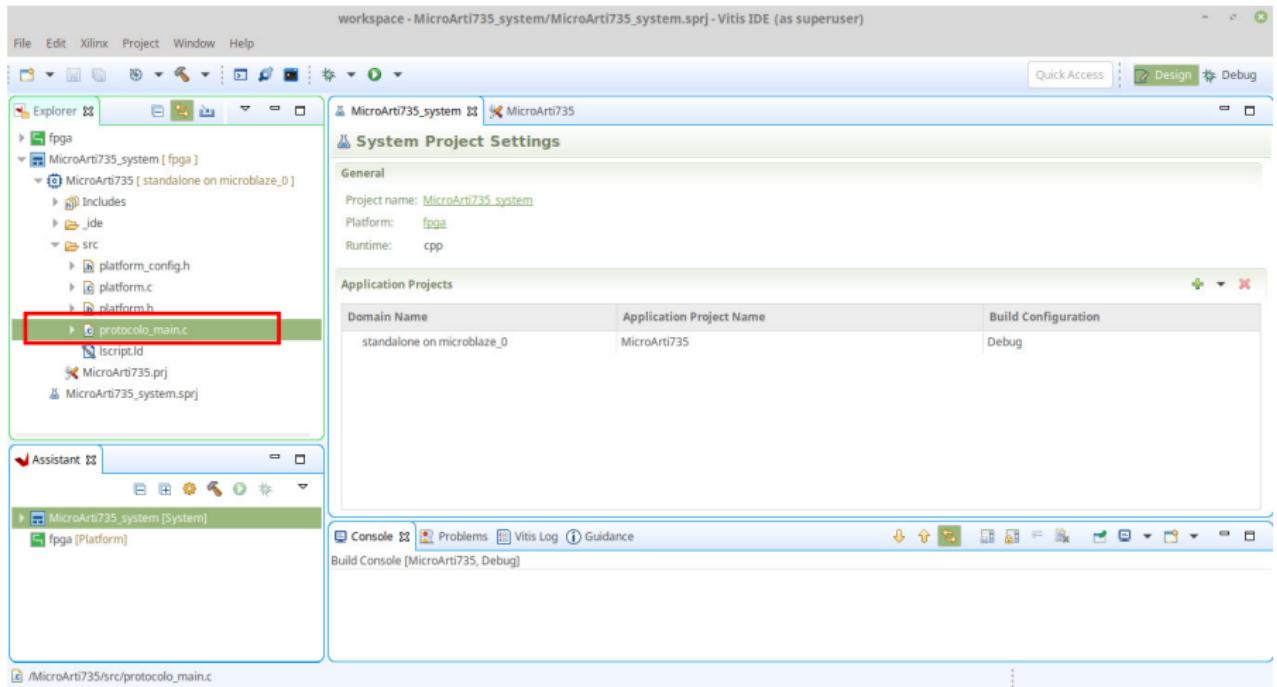
Se generan dos carpetas con el detalle del hardware y la aplicación en C seleccionada.
Se puede verificar las características técnicas del diseño haciendo click en el nombre del hardware.

Crear la aplicación en Vitis



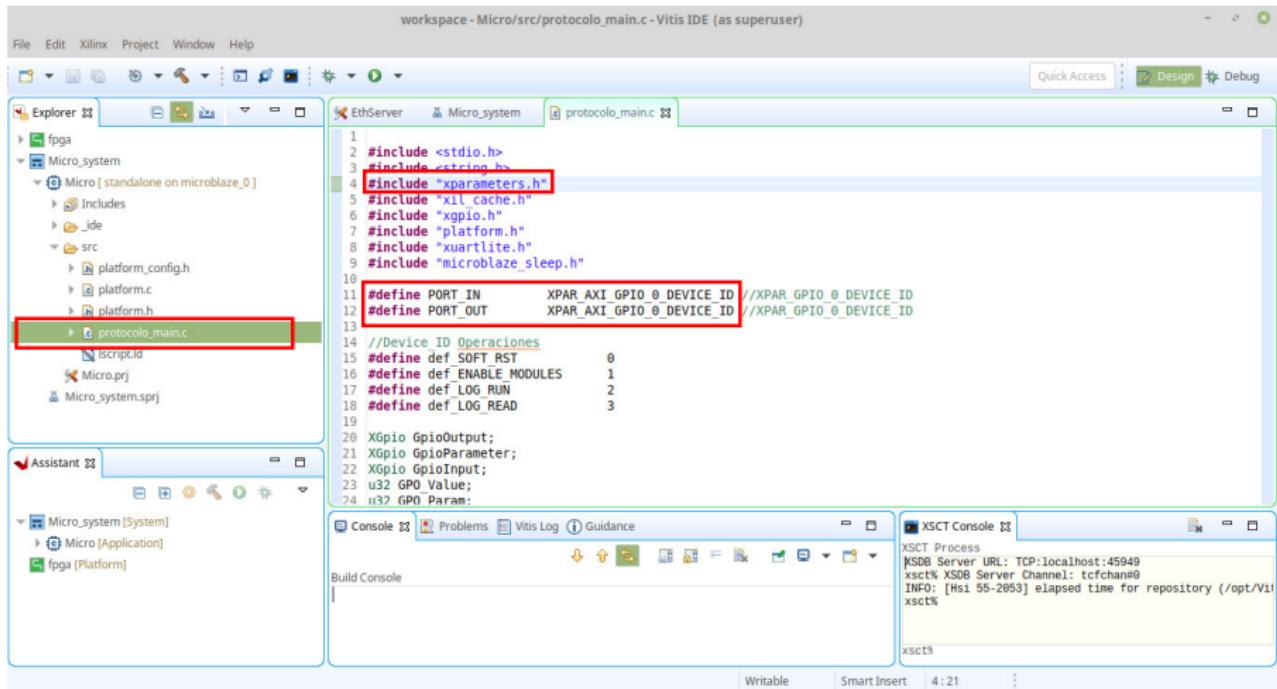
Detalles de la aplicación.

Crear la aplicación en Vitis



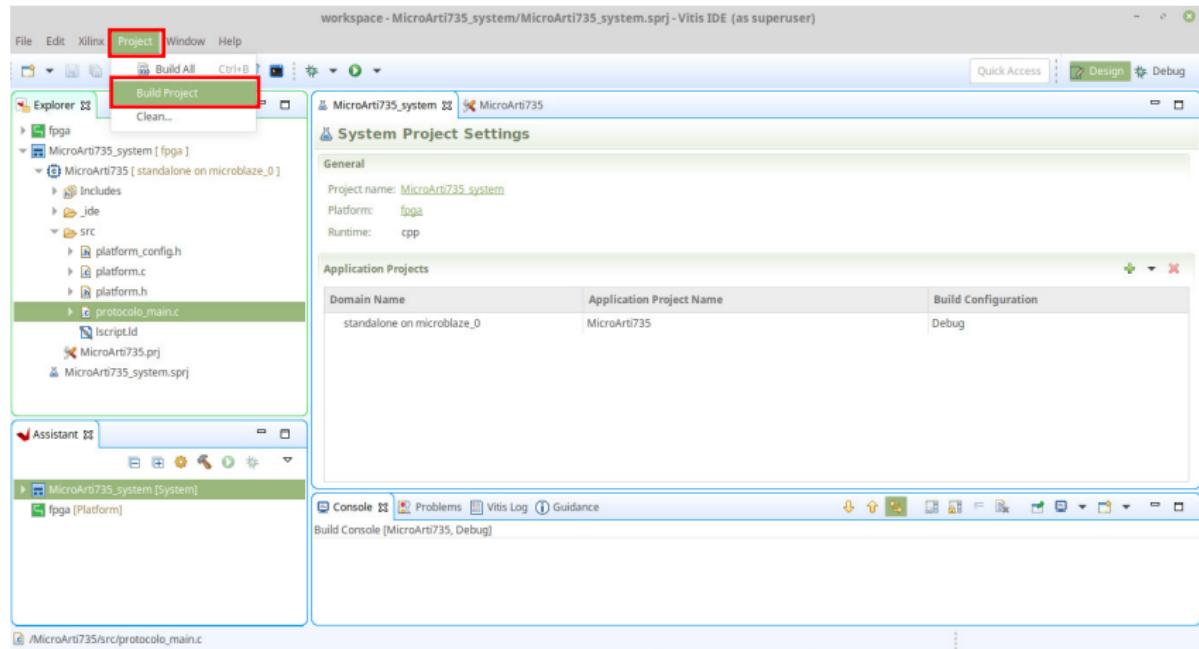
Reemplazar el archivo “Hello Word” con “protocolo_main” desde el dirección del proyecto.
Ej. “/home/apola/workspace/MicroArti735/src/”

Crear la aplicación en Vitis



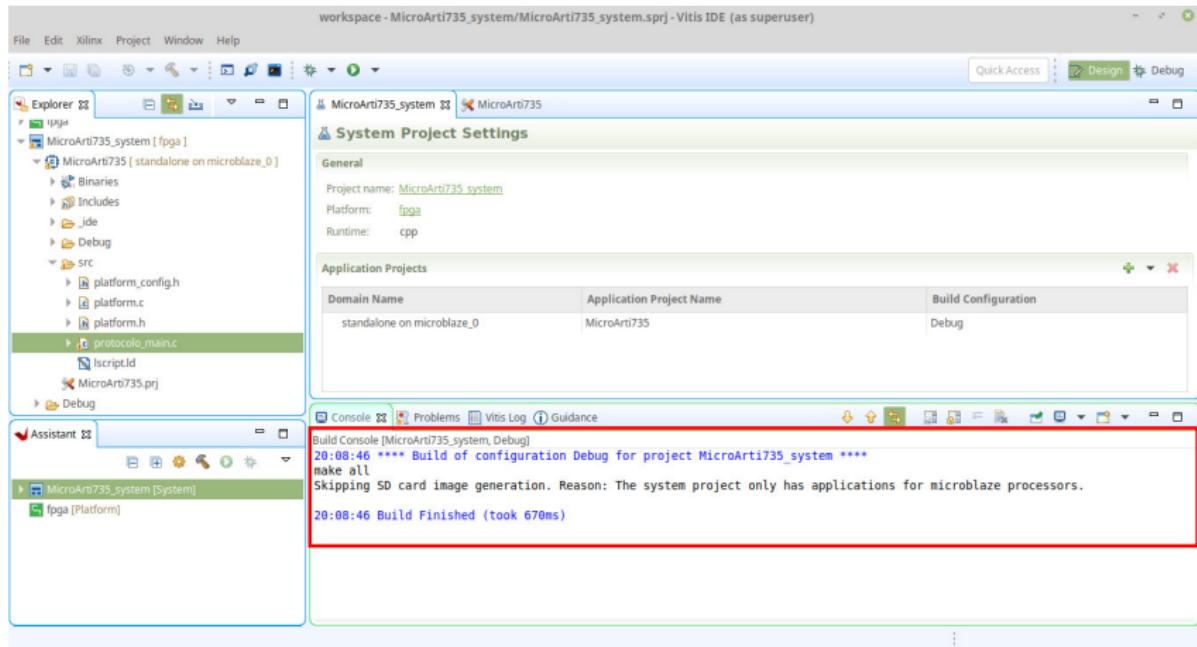
Verificar la definición de los puertos GPIO desde el archivo “xparameters.h”. En caso de error en la compilación, debemos buscar en el archivo “xparameters.h” la definición que contenga las palabras claves XPAR, AXI, GPIO, DEVICE e ID.

Crear la aplicación en Vitis



Compilar el proyecto (Build Project) o compilar todo Build All).

Crear la aplicación en Vitis



Verificar el estado de la compilación.

Tunel, Programar la FPGA y Ejecutar la Aplicación en Forma Remota



Tunel y Asignación de FPGA

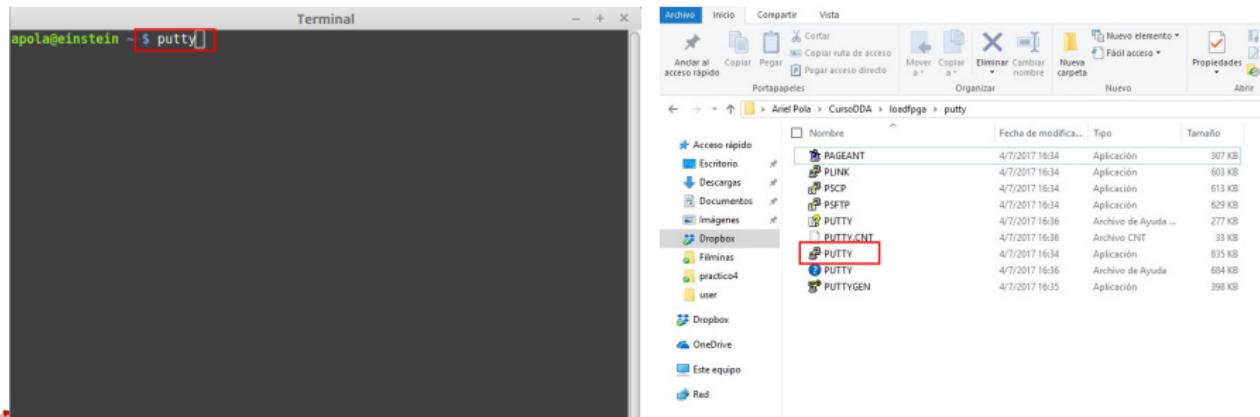
Acceso Remoto a Servidor

- Instalar la herramienta Putty. En el caso de Windows se tendrán los ejecutables descargados desde

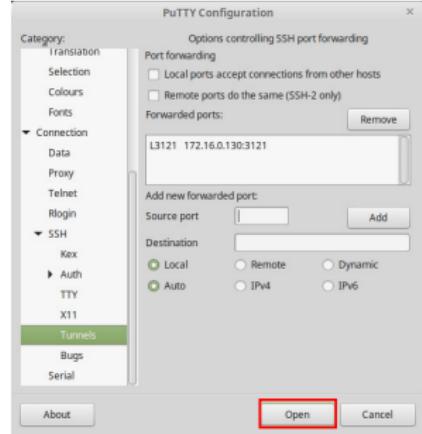
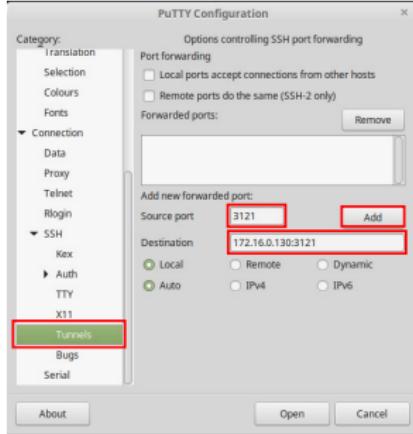
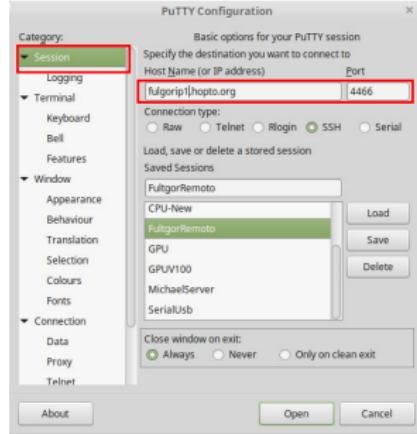
<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>.

- Ejecutar

- **Linux:** En un terminal ejecutar el programa Putty.
- **Windows:** Hacer doble click sobre el ícono de Putty.



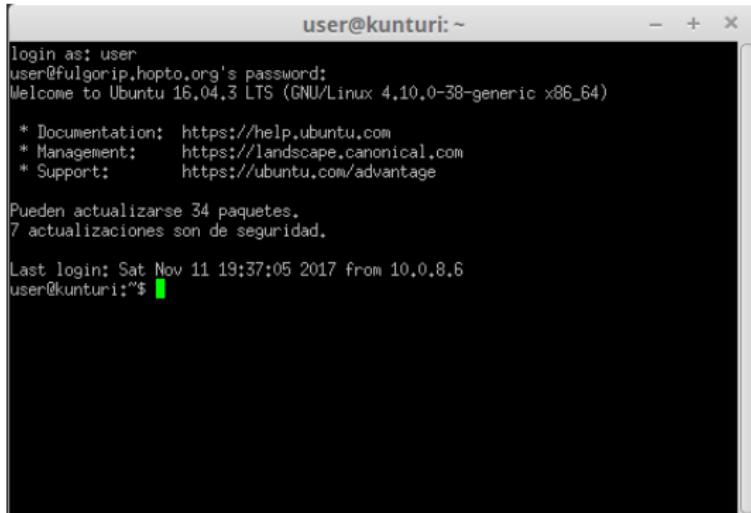
Tunel y Asignación de FPGA



- (a) En la categoría **Session**, setear el host: *fulgorip.hopto.org* o *fulgorip1.hopto.org* y el puerto: *4466*.
- (b) En la categoría **Connection** –> **SSH** –> **Tunnels** incluir el *forwarded* del puerto y agregarlo a la lista.
- (c) Abrir el tunel.

Tunel y Asignación de FPGA

- Para el acceso se solicita usuario (**user**) y contraseña (**Cai1keaw**).
- Este acceso es para todos por igual.



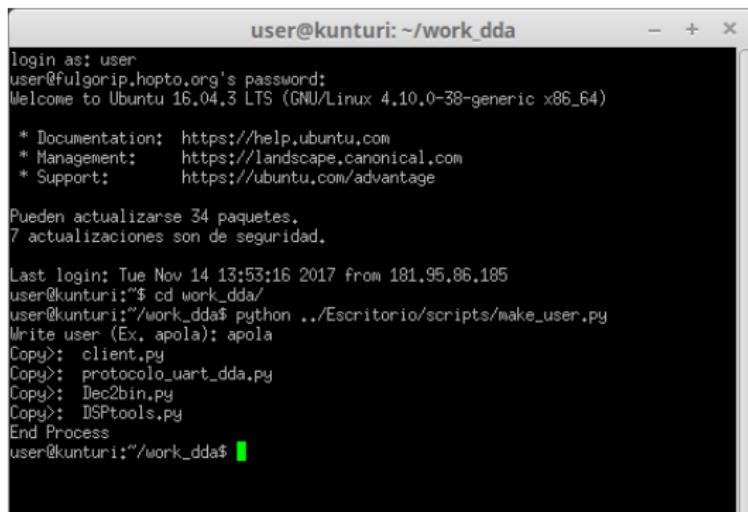
```
user@kunturi:~  
login as: user  
user@fulgorip.hopto.org's password:  
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-38-generic x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
Pueden actualizarse 34 paquetes.  
7 actualizaciones son de seguridad.  
  
Last login: Sat Nov 11 19:37:05 2017 from 10.0.8.6  
user@kunturi:$
```

Tunel y Asignación de FPGA

- Para generar el entorno de trabajo debemos ejecutar la siguiente linea de comando dentro de la carpeta **work_dda** agregando el usuario personal con la inicial del nombre y el apellido completo (Ej. apola).

- 1 *cd work_dda*
- 2 *python ..//Escritorio/scripts/make_user.py*

- El script copia unos archivos y arma el árbol de carpetas.



```
user@kunturi:~/work_dda
login as: user
user@fulgorip.hopto.org's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-38-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

Pueden actualizarse 34 paquetes,
7 actualizaciones son de seguridad.

Last login: Tue Nov 14 13:53:16 2017 from 181.95.86.185
user@kunturi:~$ cd work_dda/
user@kunturi:~/work_dda$ python ..//Escritorio/scripts/make_user.py
Write user (Ex. apola): apola
Copy> client.py
Copy> protocolo_uart_dda.py
Copy> Dec2bin.py
Copy> DSPtools.py
End Process
user@kunturi:~/work_dda$
```

Tunel y Asignación de FPGA

- Acceder a la carpeta <**user**>/**scripts**/ y ejecutar **client.py** para solicitar un puerto USB y el ID de la placa FPGA disponible.

1 *cd apola/scripts*

2 *python client.py*

- El script retorna el USB: **USB3** y el ID de la FPGA: **210319A2CECCA**.
- En caso de no haber disponible ninguna placa, el script retorna que los puertos estan ocupados.
- Esta ventana y el script no se tienen que cerrar hasta no terminar de usar la FPGA, ya que se libera el puerto y dará acceso a otro usuario.
- En las siguientes figuras se observa el caso de asignación correcta y puertos ocupados.

Tunel y Asignación de FPGA

Nota: Copia de archivos o carpetas

- Usamos el comando **pscp** para copiar archivos entre sesiones.

- Remoto a local

```
pscp -P 4466 user@fulgorip1.hopto.org:/home/user/work_dda/<user>/<file> ./
```

- Local a remoto

```
pscp -P 4466 ./<file> user@fulgorip1.hopto.org:/home/user/work_dda/<user>/
```

Tunel y Asignación de FPGA

```
user@kunturi:~/work_dda/apola/scripts - + ×
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

Pueden actualizarse 34 paquetes,
7 actualizaciones son de seguridad.

Last login: Tue Nov 14 14:03:43 2017 from 181.95.86.185
user@kunturi:~$ cd work_dda/
user@kunturi:~/work_dda$ cd apola/scripts/
user@kunturi:~/work_dda/apola/scripts$ python client.py

Write-> Exit <-to close the session

Ip: 127.0.0.1
-----
Port: 5005
-----

Checking Free Port
USB Free: USB3-210319A2CECA
-----
Write Text to check Session or Exit to close: █
```

```
user@kunturi:~/work_dda/apola/scripts - + ×
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-38-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

Pueden actualizarse 34 paquetes,
7 actualizaciones son de seguridad.

Last login: Tue Nov 14 14:23:29 2017 from 181.95.86.185
user@kunturi:~$ cd work_dda/apola/scripts/
user@kunturi:~/work_dda/apola/scripts$ python client.py

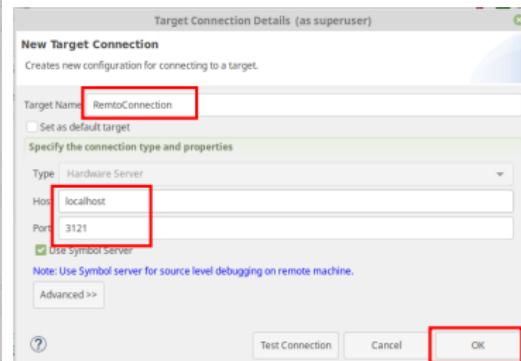
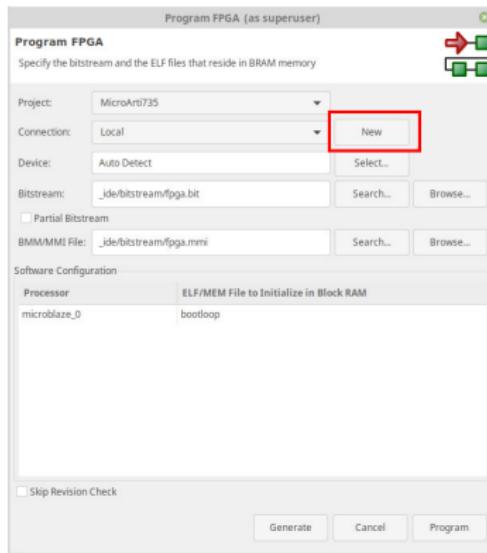
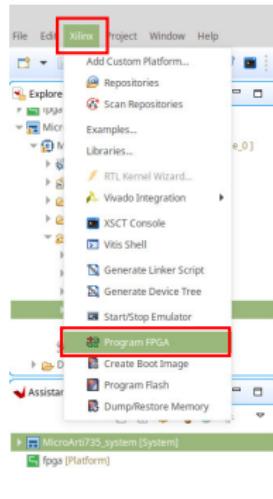
Write-> Exit <-to close the session

Ip: 127.0.0.1
-----
Port: 5005
-----

Checking Free Port
Ports Busy
user@kunturi:~/work_dda/apola/scripts$ █
```

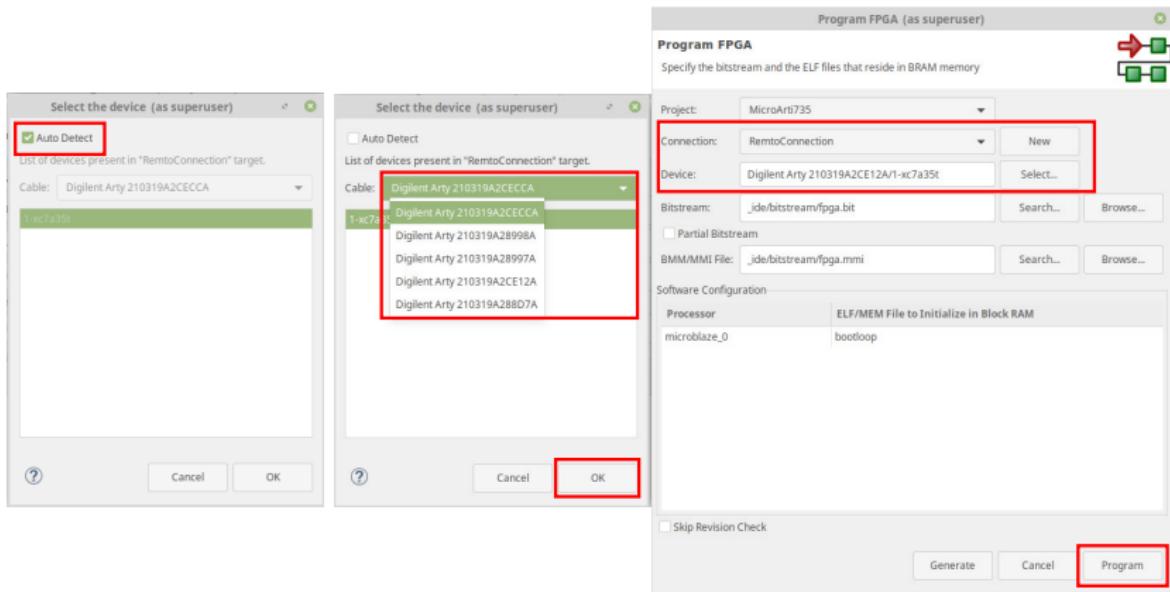
Las figuras muestran la asignación correcta de puerto (izq.) y los puertos ocupados (der.).

Programar la FPGA



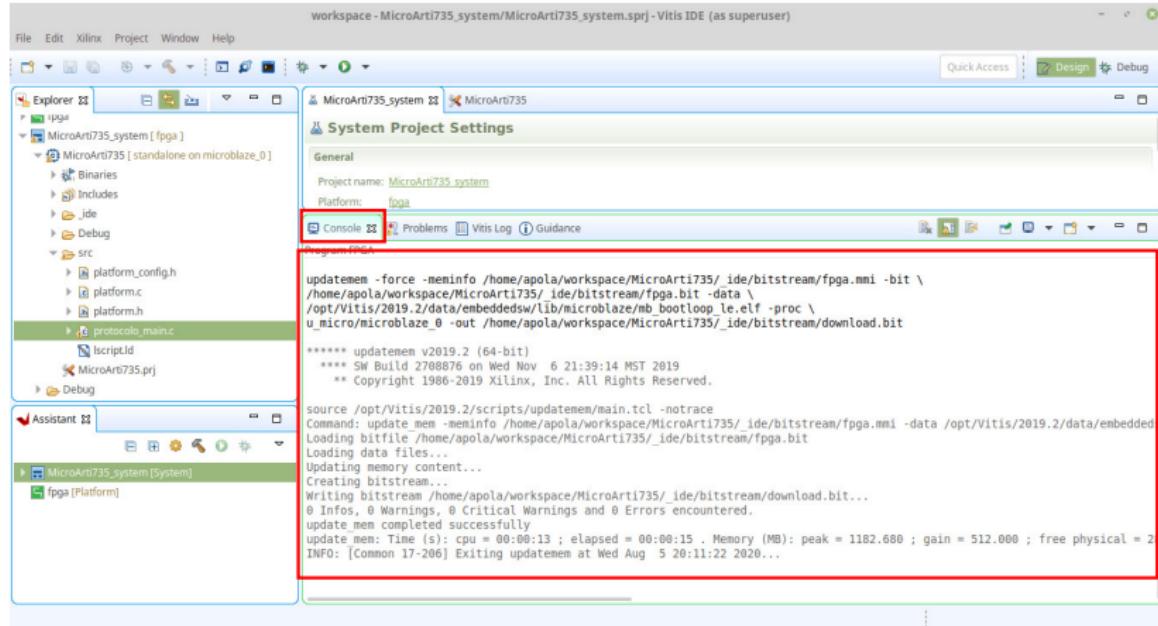
Programar la FPGA. Crear un nuevo Server, dar un nombre y colocar la dirección IP del server remoto. Para nuestro caso, el cual realiza un forward del puerto 3121 se debe colocar LOCALHOST como dirección de host.

Programar la FPGA



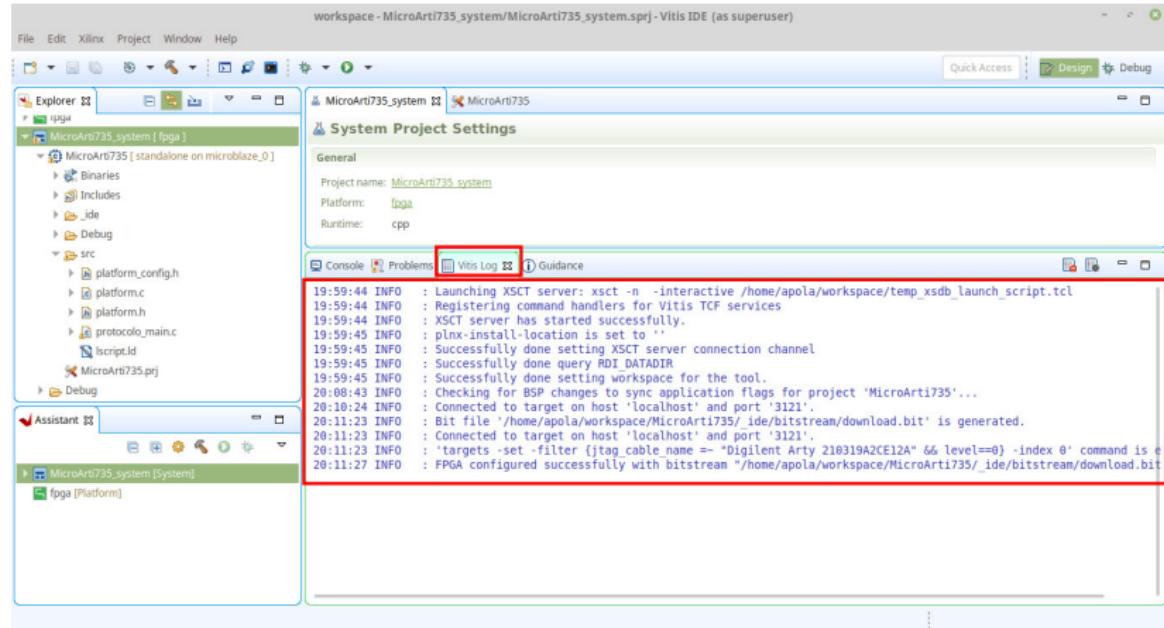
Desactivar AutoDetect, seleccionar el ID de la FPGA y verificar la información.

Programar la FPGA



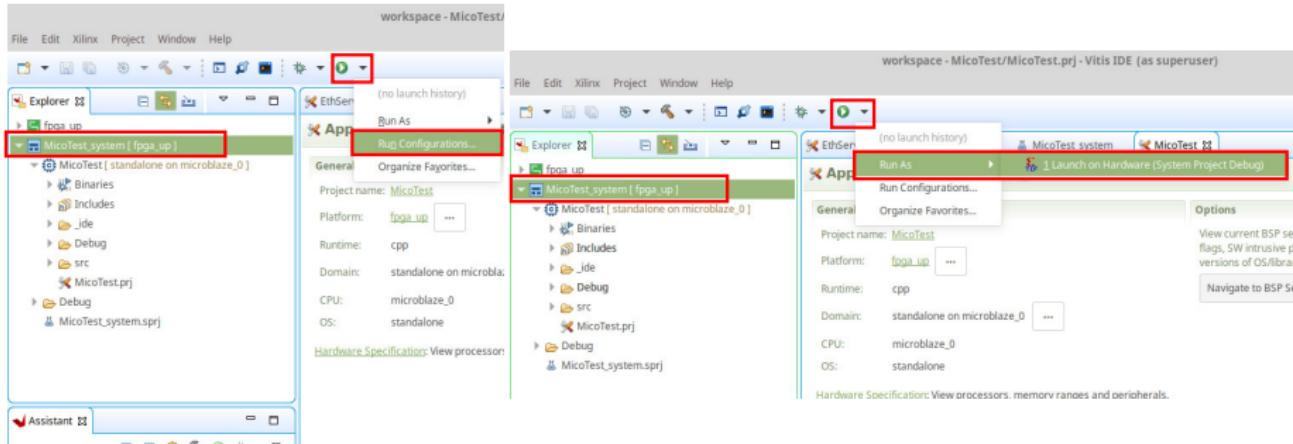
En la “consola” se observará la generación de un nuevo binario.

Programar la FPGA



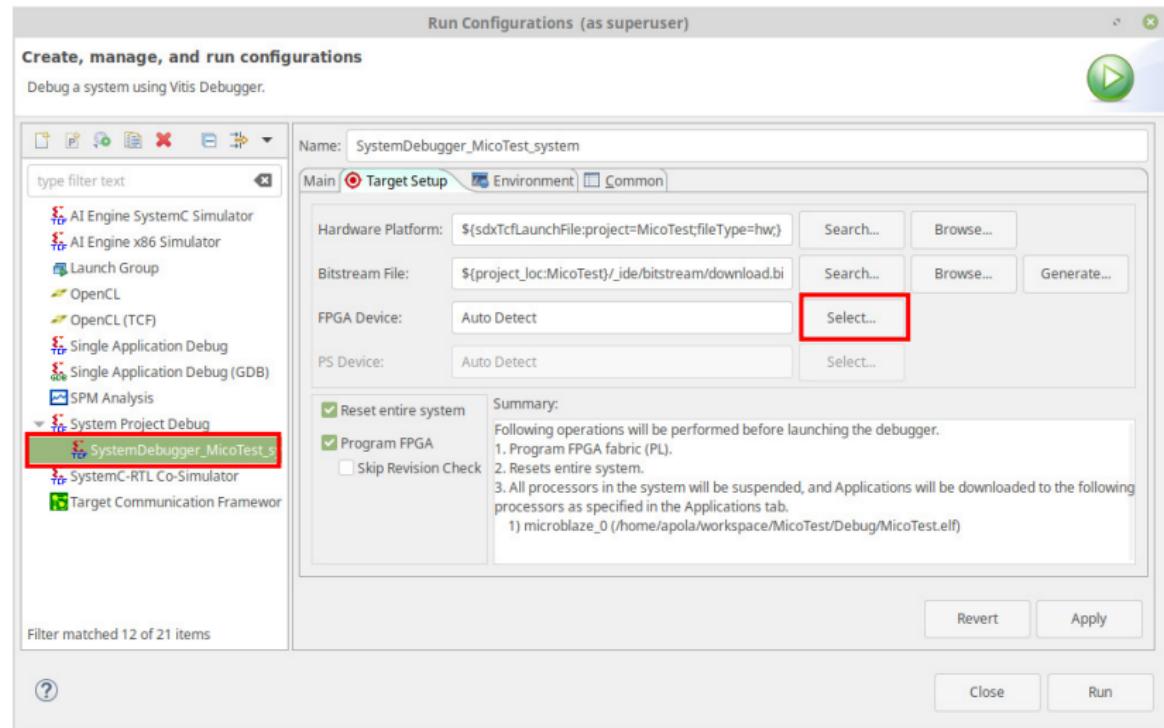
En "Vitis Log" se detallará la programación.

Ejecutar la Aplicación



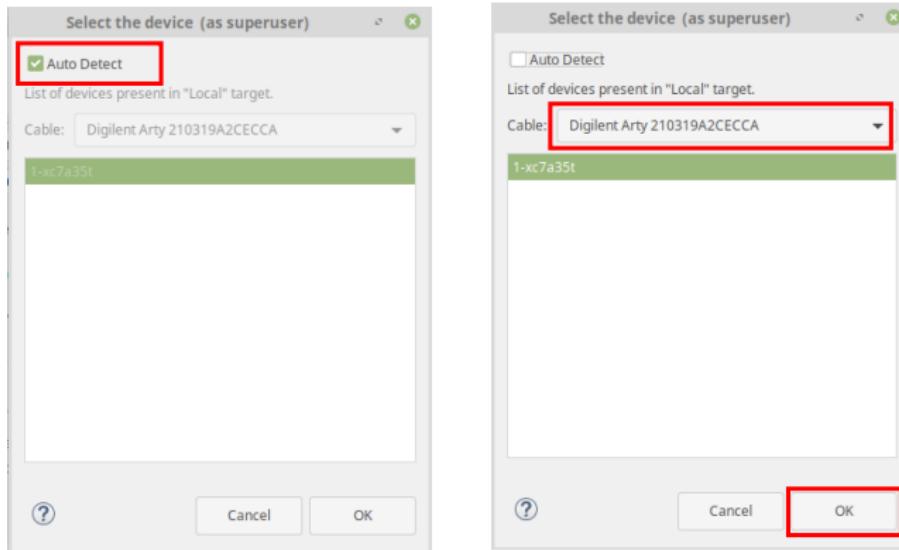
Antes de ejecutar la aplicación es necesario setear el dispositivo que se quiere correr. Para ello necesitamos acceder a Run Configuration. **Nota: En caso que no aparezcan las opciones que se detallan en la figura de la siguiente pagina, debemos ejecutar al menos una vez la aplicación. En este caso debemos seguir los pasos de la figura de la derecha.**

Ejecutar la Aplicación



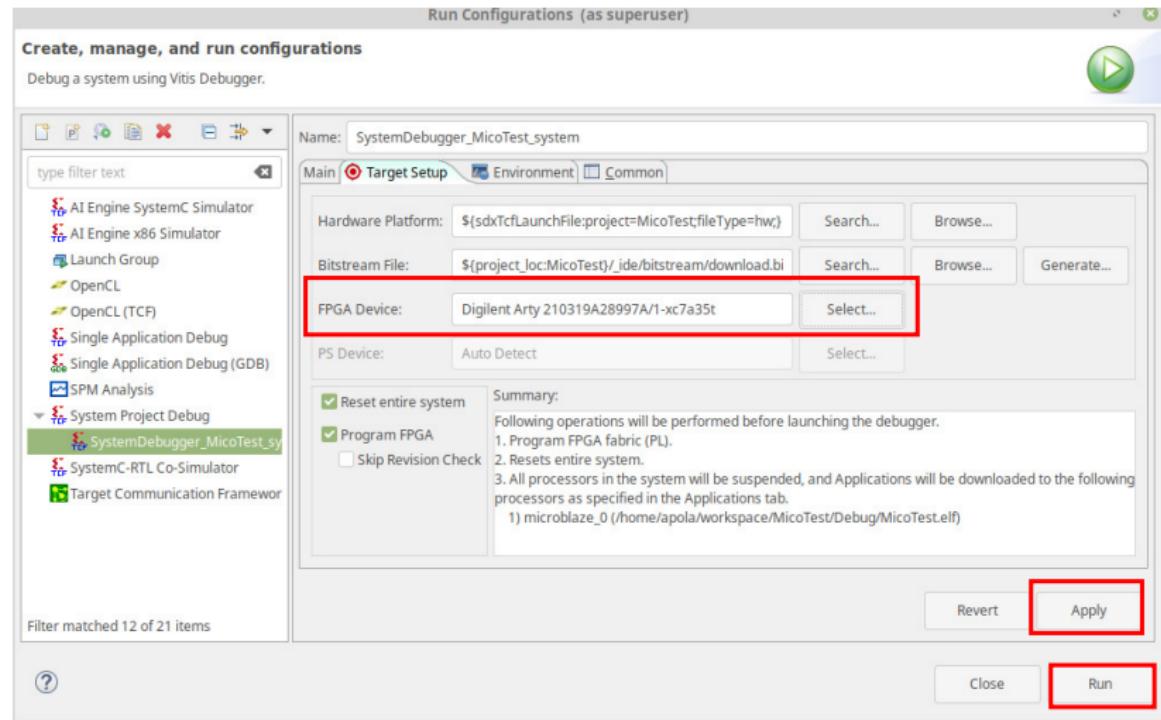
En la opción **SystemDebugger**, cambiar el dispositivo (**FPGA Device**).

Ejecutar la Aplicación



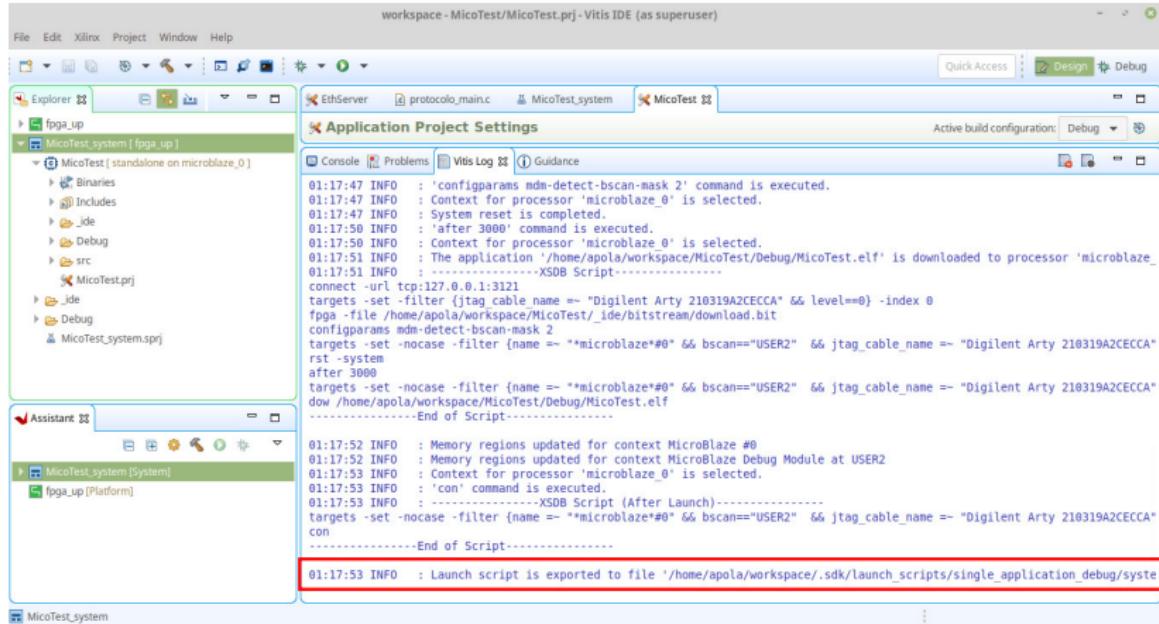
Seleccionar una FPGA.

Ejecutar la Aplicación



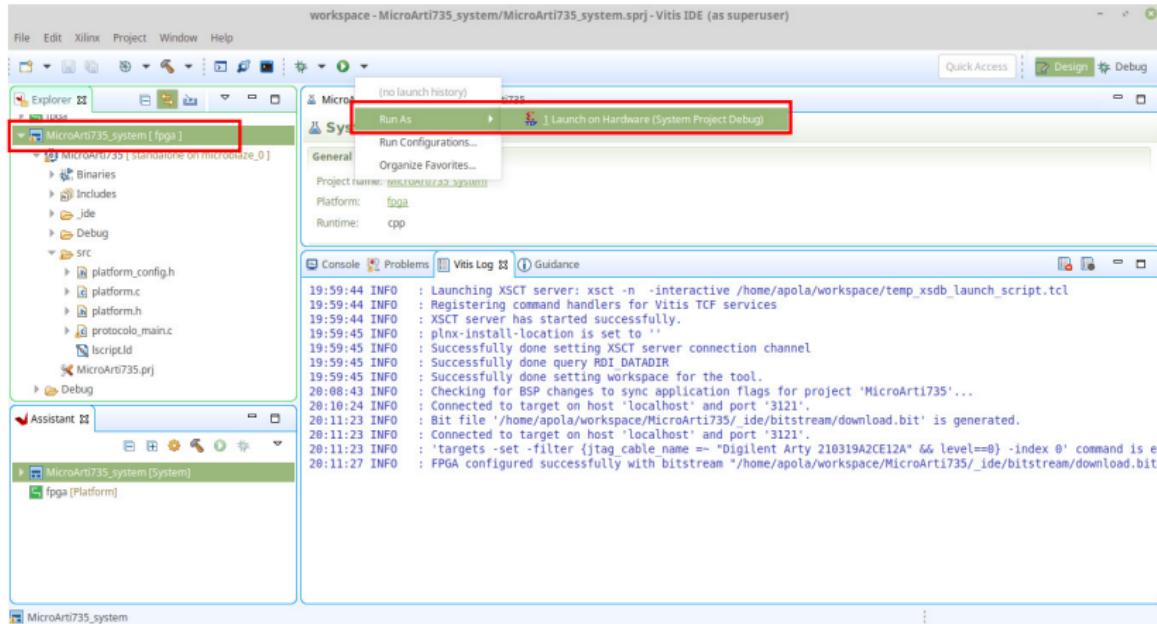
Verificar el ID, aplicar los cambios y ejecutar.

Ejecutar la Aplicación



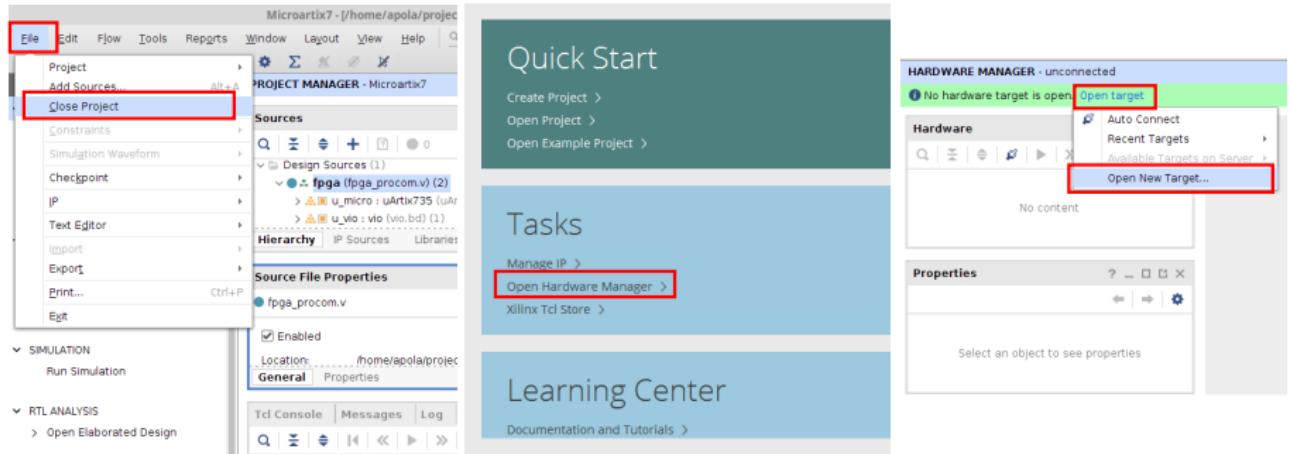
El mensaje Launch verifica que se esta corriendo la aplicación en el microProcesador.

Ejecutar la Aplicación



En caso de necesitar ejecutar varias veces la aplicación y una vez que se han ejecutado los pasos anteriores, basta con lanzar nuevamente la aplicación.

Cargar parámetros del VIO



Volver a la ventana del Vivado, cerrar el proyecto, abrir el controlador de hardware y seleccionar un nuevo target.

Cargar parámetros del VIO

Open New Hardware Target (as superuser)

Open Hardware Target

This wizard will guide you through connecting to a hardware target.

To connect to a remote hardware target, provide the host name and IP port of the remote machine on which the instance of a Vivado Hardware Server is running.

Open New Hardware Target (as superuser)

Hardware Server Settings

Select local or remote hardware server, then configure the host name and port settings. Use Local server if the target is attached to the local machine; otherwise, use Remote server.

Connect to: Remote server (target is on remote machine)

Remote Server

Host name: Port: [default is 3121]

Click Next to launch and/or connect to the fw_server (port 3121) application on the remote machine You.

XILINX

Open New Hardware Target (as superuser)

Next >

Cancel

Open Select Hardware Target

Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the frequency or select a different target.

Hardware Targets

Type	Name	JTAG Clock Frequency
xilinx_tcf	Digilent/210319A288D7A	15000000
xilinx_tcf	Digilent/210319A28998A	15000000
xilinx_tcf	Digilent/210319A28997A	15000000
xilinx_tcf	Digilent/210319A2C2CECA	15000000
xilinx_tcf	Digilent/210319A2C8E12A	15000000

Add Xilinx Virtual Cable (XVC)

Hardware Devices (For unknown devices, specify the Instruction Register (IR) length)

Name	ID Code	IR Length
xc7z25i_0_1	03620093	6

Hardware server: localhost:3121

?

< Back

Next >

Cancel

Open Open Hardware Target Summary

Hardware Server Settings

- Server: localhost:3121

Target Settings

- Target: xilinx_tcf@Digilent/210319A28998A
- Frequency: 15000000

XILINX

To connect to the hardware described above, click Finish

?

< Back

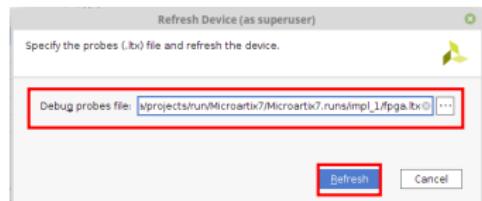
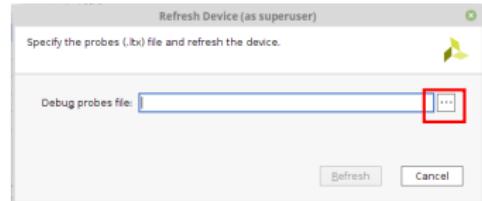
Next >

Finish

Cancel

Crear un nuevo servidor cuya IP sea LOCALHOST y seleccionar la FPGA con el ID que previamente el servidor les entrega.

Cargar parámetros del VIO



Debemos agregar el archivo con extensión **.Itx** que se generó cuando creamos el VIO en los pasos anteriores. Se encuentra dentro de la carpeta

nameProject > .runs/impl_1/ < nameBinario > .Itx

Diseño Digital Avanzado

Cargar parámetros del VIO

The screenshot shows two windows from a software interface for managing VIO parameters.

Left Window: Add Probes

- Header: Add Probes
- Search bar: Search: Q:
- Section: Probes for hw_vio_1 (6)
- List:
 - > u_vio/probe_in0_0[2:0]
 - > u_vio/probe_in1_0[2:0]
 - > u_vio/probe_in2_0[2:0]
 - > u_vio/probe_in3_0[2:0]
 - > u_vio/vio_0_probe_out0
 - > u_vio/vio_0_probe_out1[3:0]

Right Window: hw_vios Configuration

Name	Type	Activity	Direction	VIO
> u_vio/probe_in3_0[2:0]	[H] 1	Input	hw_vio_1	
> u_vio/probe_in2_0[2:0]	[H] 1	Input	hw_vio_1	
> u_vio/probe_in0_0[2:0]	[H] 1	Input	hw_vio_1	
> u_vio/probe_in1_0[2:0]	[H] 1	Input	hw_vio_1	
u_vio/vio_0_probe_out0	[B] 1	Output	hw_vio_1	
> u_vio/vio_0_probe_out1[3:0]	[H] 0	Output	hw_vio_1	

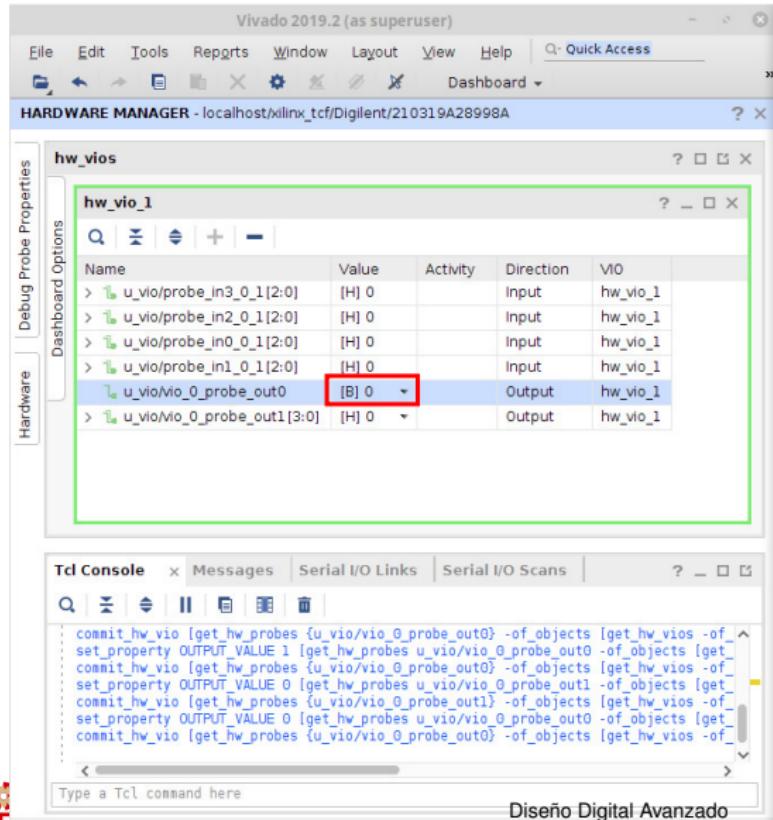
Agregamos los puertos de entrada y salida que queremos controlar y observar.

Asignar el Puerto USB



En un nuevo terminal de PUTTY, ejecutamos el script Python_UART_FPGA.py asignando el puerto UART.

VIO y Python



```
user@kunturi:~/work_dda/apola/scripts$ python Python_UART_FPGA.py
```

VIO y Python

The screenshot shows two windows side-by-side. On the left is the Vivado 2019.2 Hardware Manager, specifically the 'hw_vios' tab for a device named 'hw_vio_1'. A red box highlights the value '1' for the input probe 'u_vio/probe_in3_0_1[2:0]'. On the right is a terminal window titled 'user@kuntrui:~/work_dda/apola/scripts'. It displays the command 'python Python_VART_FPGA.py' and the prompt 'Ingres un comando:[0,1,2,3]'. The terminal window has a red box around its title bar.

hw_vios

Name	Value	Activity	Direction	VIO
> u_vio/probe_in3_0_1[2:0]	[H] 1		Input	hw_vio_1
> u_vio/probe_in2_0_1[2:0]	[H] 1		Input	hw_vio_1
> u_vio/probe_in0_0_1[2:0]	[H] 1		Input	hw_vio_1
> u_vio/probe_in1_0_1[2:0]	[H] 1		Input	hw_vio_1
u_vio/vio_0_probe_out0	[B] 0		Output	hw_vio_1
u_vio/vio_0_probe_out1[3:0]	[H] 0		Output	hw_vio_1

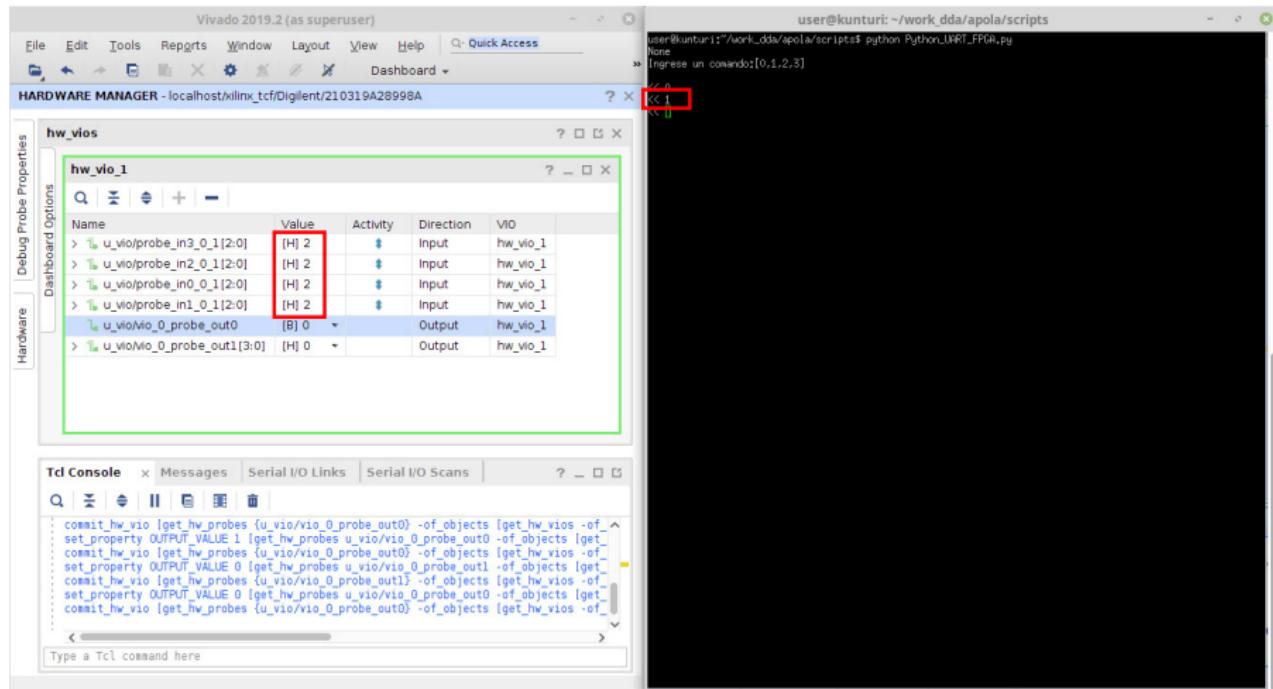
Tcl Console

```
commit_hw_vio [get_hw_probes {u_vio/vio_0_probe_out0}] -of_objects [get_hw_vios -of _set_property OUTPUT_VALUE 1 [get_hw_probes {u_vio/vio_0_probe_out0} -of_objects [get_commit_hw_vio [get_hw_probes {u_vio/vio_0_probe_out0} -of_objects [get_hw_vios -of _set_property OUTPUT_VALUE 0 [get_hw_probes {u_vio/vio_0_probe_out1} -of_objects [get_commit_hw_vio [get_hw_probes {u_vio/vio_0_probe_out1} -of_objects [get_hw_vios -of _set_property OUTPUT_VALUE 0 [get_hw_probes {u_vio/vio_0_probe_out0} -of_objects [get_commit_hw_vio [get_hw_probes {u_vio/vio_0_probe_out0}] -of_objects [get_hw_vios -of _
```

Type a Tcl command here

Elegimos 0 y activa el color Rojo (1).

VIO y Python



Elegimos 1 y activa el color Verde (2).

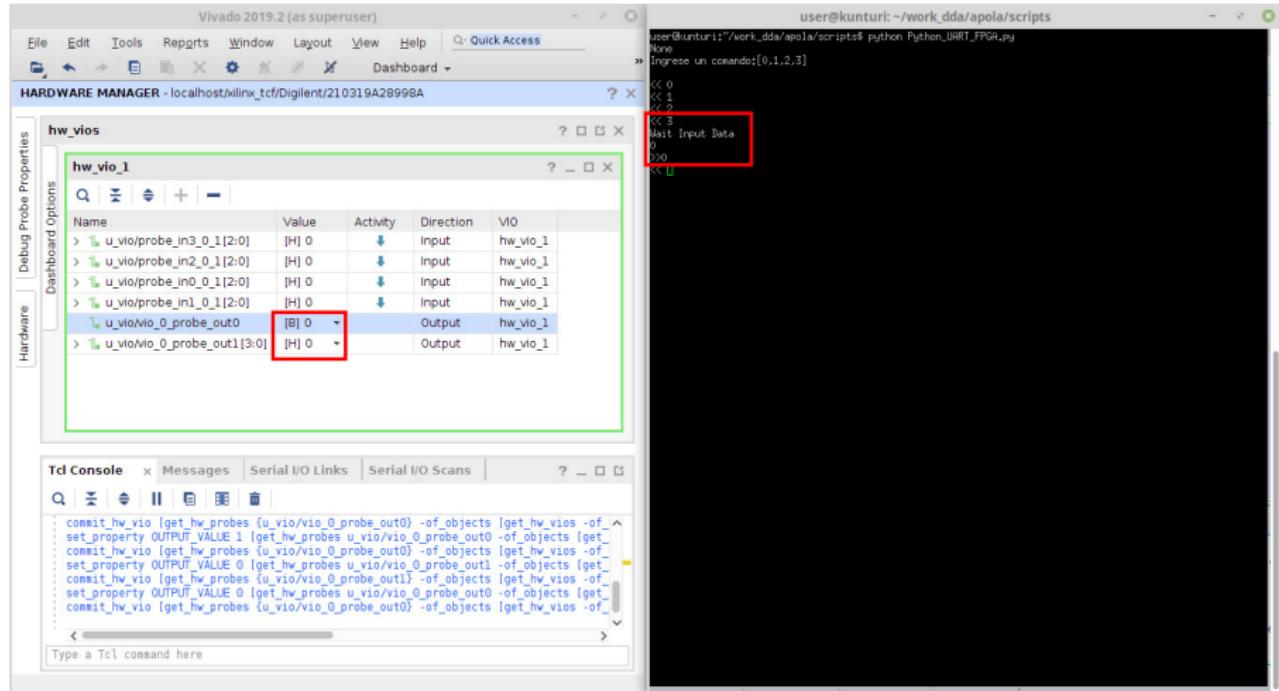
VIO y Python

The screenshot shows the Vivado 2019.2 interface with the following components:

- Hardware Manager:** A window titled "hw_vios" showing a table of I/O probes. One row, "u_vio/probe_in3_0_1[2:0]", has its value set to [H] 4, which is highlighted with a red box.
- Tcl Console:** A terminal window displaying a series of TCL commands for managing VIO probes. The commands involve setting properties like OUTPUT_VALUE for specific probe objects.
- Terminal:** A separate terminal window showing the command "python Python_UART_FPGA.py" and the prompt "Ingresé un comando:[0,1,2,3]".

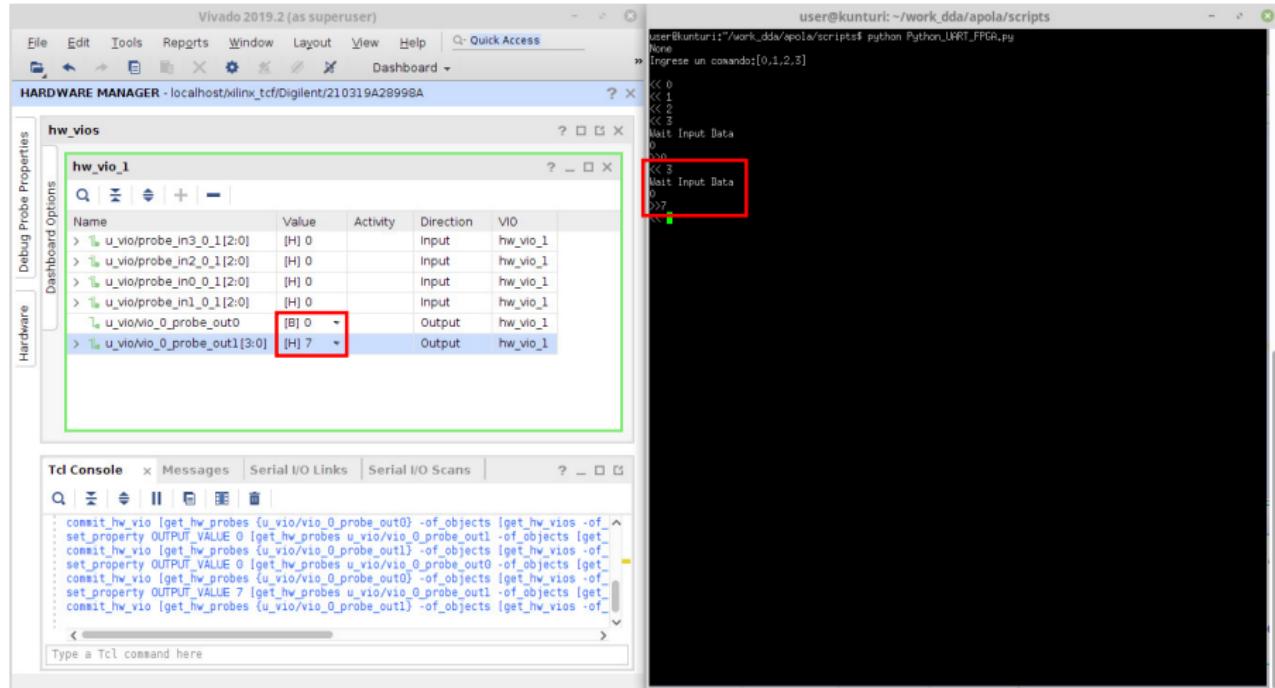
Elegimos 2 y activa el color Azul (4).

VIO y Python



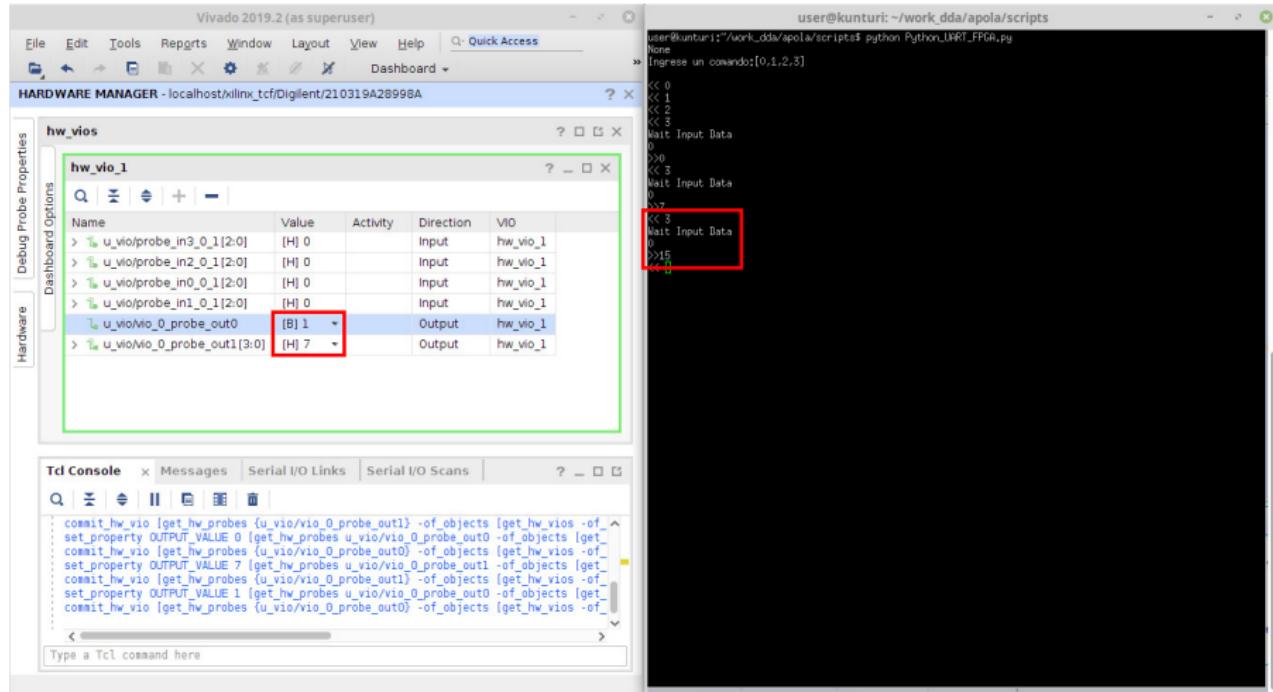
Elegimos 3 para leer el estado de los Switch. Con la salida out0: 0 habilitamos el control desde el VIO y el python tiene que leer el estado de la salida out1 (Python >> 0).

VIO y Python



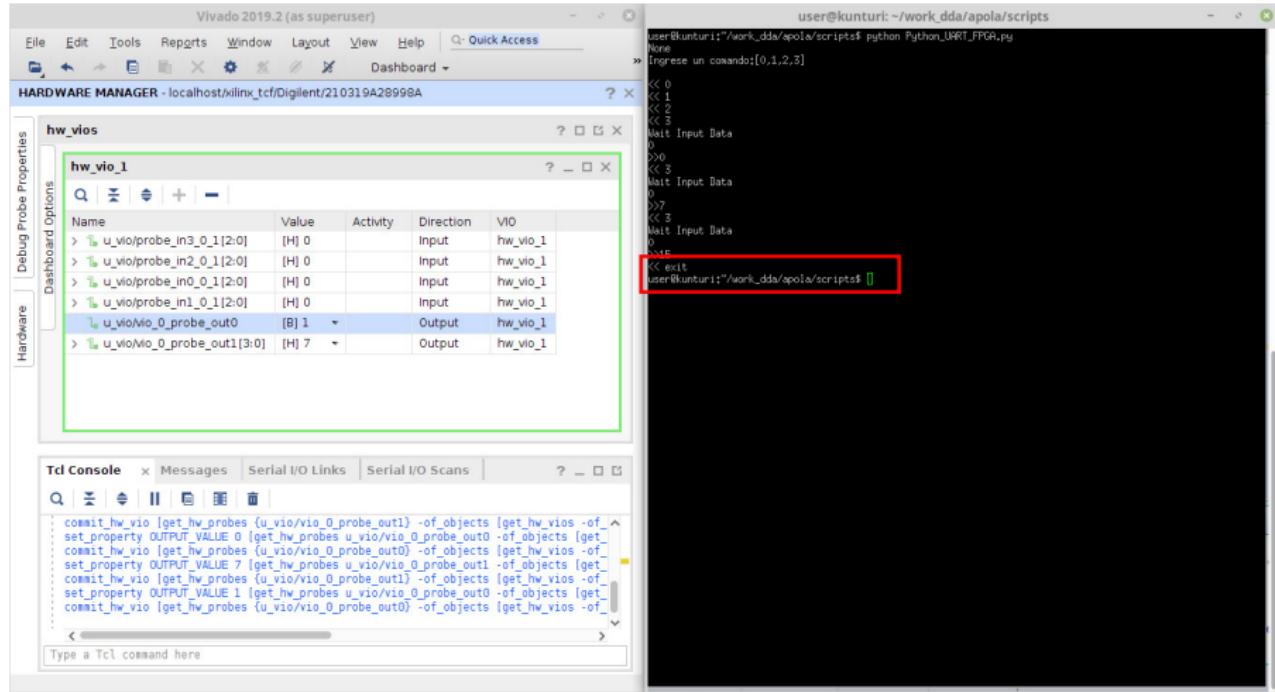
Elegimos 3 para leer el estado de los Switch. Con la salida out0: 0 habilitamos el control desde el VIO y el python tiene que leer el estado de la salida out1 (Python >> 7).

VIO y Python



Elegimos 3 para leer el estado de los Switch. Con la salida out0: 1 habilitamos el control desde el VIO y el python tiene que leer el estado de los switchs (Python >> 15).

VIO y Python



Salimos del script.

Cierre de Sesiones

Cerrando el Tunel

- Liberando el puerto USB
 - Para liberar el puerto USB asignado escribir la palabra **Exit**.
- Cerrando los dos tuneles
 - Para cerrar ambas sesiones escribir la palabra **exit**.

```
user@kunturi: ~/work_dda/apola/scripts - + x
* Support: https://ubuntu.com/advantage

Pueden actualizarse 27 paquetes,
0 actualizaciones son de seguridad.

Last login: Wed Nov 15 21:11:43 2017 from 200.126.231.123
user@kunturi: "$ cd work_dda/apola/scripts/
user@kunturi: ~/work_dda/apola/scripts$ python client.py

Write-> Exit <-to close the session
-----
Ip: 127.0.0.1
-----
Port: 5005
-----
Checking Free Port
-----
USB Free: USB3-210319A2CECCA
-----
Write Text to check Session or Exit to close: Exit
Echo: Exit
Close Session
user@kunturi: ~/work_dda/apola/scripts$ exit
```

```
user@kunturi: ~/work_dda/apola/scripts - + x
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-38-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

Pueden actualizarse 27 paquetes,
0 actualizaciones son de seguridad.

Last login: Wed Nov 15 22:33:00 2017 from 200.126.231.123
user@kunturi: "$ cd work_dda/apola/scripts/
user@kunturi: ~/work_dda/apola/scripts$ python protocolo_uart_dda.py 3
RESET ON
MODULES OFF
LOG RUN OFF
RESET OFF
ENB MODULES
LOG
LOG RUN OFF
LOG RUN ON
LOG READ
4096
End Script
user@kunturi: ~/work_dda/apola/scripts$ exit
```

Las figuras muestran como liberar el USB asignado y el cierre de los tuneles abiertos.