

Diseño Digital Avanzado

Unidad 1 - Herramientas

Dr. Ariel L. Pola

ariel.pola@mi.unc.edu.ar

August 17, 2024

Tabla de Contenidos

1. Proyecto Leds
2. Herramienta Vivado
3. Instancia de VIO e ILA
4. Tunel y Asignación de FPGA
5. Programación y Ejecución

Proyecto Leds



Proyecto LEDS

Implementación en FPGA - Leds

Objetivo

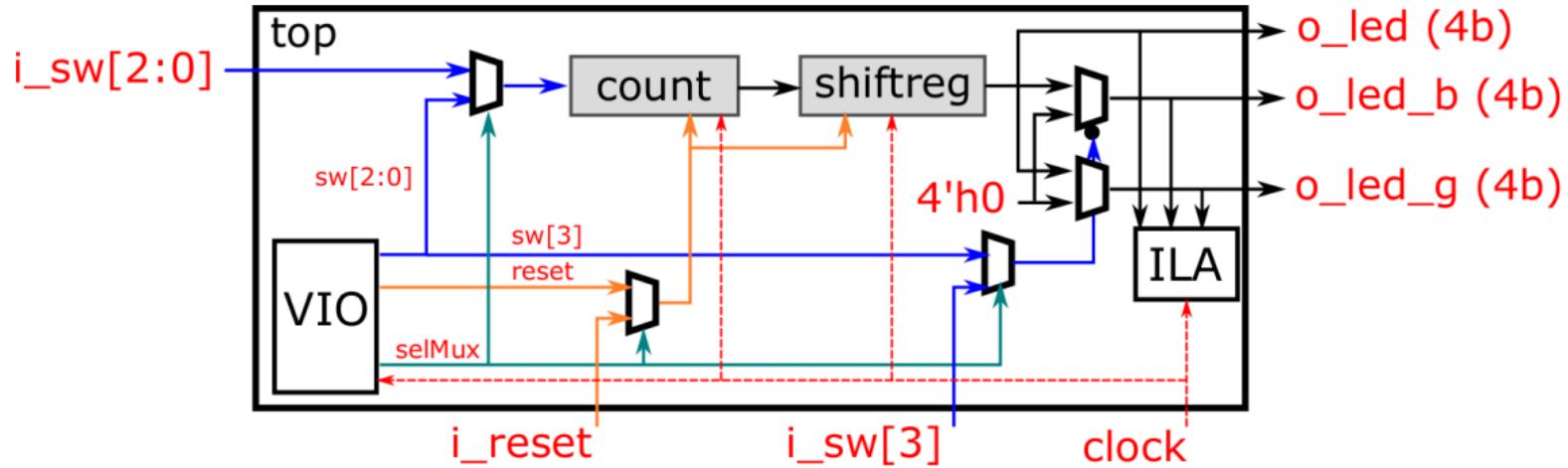
- Incluir los módulos VIO e ILA para control y debug del módulo Top.
- Los nombres en ROJO son puertos.
- *i_reset* es el reset del sistema, el cual pone a cero el contador e inicializa el shift register (SR).
- *i_sw[0]* controla el enable (1) del contador. En estado cero (0) detiene el funcionamiento sin alterar el estado actual del contador y del SR.
- El SR se desplaza únicamente cuando el contador llegó a algún límite R0-R3. La selección del límite se realiza con la señal *i_sw[2:1]*.
- La elección del límite se puede realizar en cualquier momento del funcionamiento.
- *i_sw[3]* elige el color de los leds RGB.

Entregar

- Código verilog con la instancia del VIO e ILA.
- Testbench en verilog que valide el comportamiento del diseño sin los IP cores VIO e ILA.
- Implementarlo en la FPGA.
- Realizar capturas de pantallas que muestren las formas de onda del Testbench e ILA.

Proyecto LEDS

Implementación en FPGA - Leds



Descripción

- El testbench (banco de pruebas) es un módulo que no tiene puertos declarados.
- Genera los estímulos de reloj y señales de control para modelar un escenario de prueba.
- Las variables utilizadas para estimular los puertos de entrada son declaradas como tipo **reg**.
- Las variables utilizadas para conectar los puertos de salida son declaradas como tipo **wire**.
- Lectura y cambio de estado de variables internas a los módulos
 - Definiendo las instancias se puede leer las variables internas
Ejemplo, assign tb_count = tb_shiftneds.u_shiftneds.counter;
 - Utilizando **force** se cambia el valor de una variable en una instancia. Se debe definir dentro de un bloque de procesamiento **initial**.
Ejemplo, force tb_shiftneds.u_shiftneds.o_led = 4'b0001;

Proyecto LEDS

Test Bench

Ejemplo - Generando Estímulos

```
1 'define N_LEDS 4
2 'define NB_SW 4
3
4 'timescale 1ns/100ps
5
6 module tb_shiftleds();
7
8 parameter N_LEDS = 'N_LEDS ;
9
10 wire [N_LEDS - 1 : 0] o_led ;
11 reg [NB_SW - 1 : 0] i_sw ;
12 reg ck_rst ;
13 reg CLK100MHZ;
14
15 initial begin
16     i_sw      = 4'b0000 ;
17     CLK100MHZ = 1'b0 ;
18     ck_rst   = 1'b0 ;
19     #100 ck_rst = 1'b1 ;
20     #100 i_sw   = 4'b0001 ;
21     #1000000 i_sw = 4'b0011 ;
22     #1000000 i_sw = 4'b1011 ;
23     #1000000 $finish;
24 end
```

```
1
2 always #5 CLK100MHZ = ~CLK100MHZ;
3
4 shiftleds
5 #(N_LEDS (N_LEDS) ,
6   .NB_SW (NB_SW)
7 )
8 u_shiftleds
9 (.o_led (o_led) ,
10  .i_sw (i_sw) ,
11  .ck_rst (ck_rst) ,
12  .CLK100MHZ (CLK100MHZ)
13 );
14
15 endmodule // tb_shiftleds
```

Proyecto LEDS

Test Bench

Ejemplo - Estímulos desde Archivos

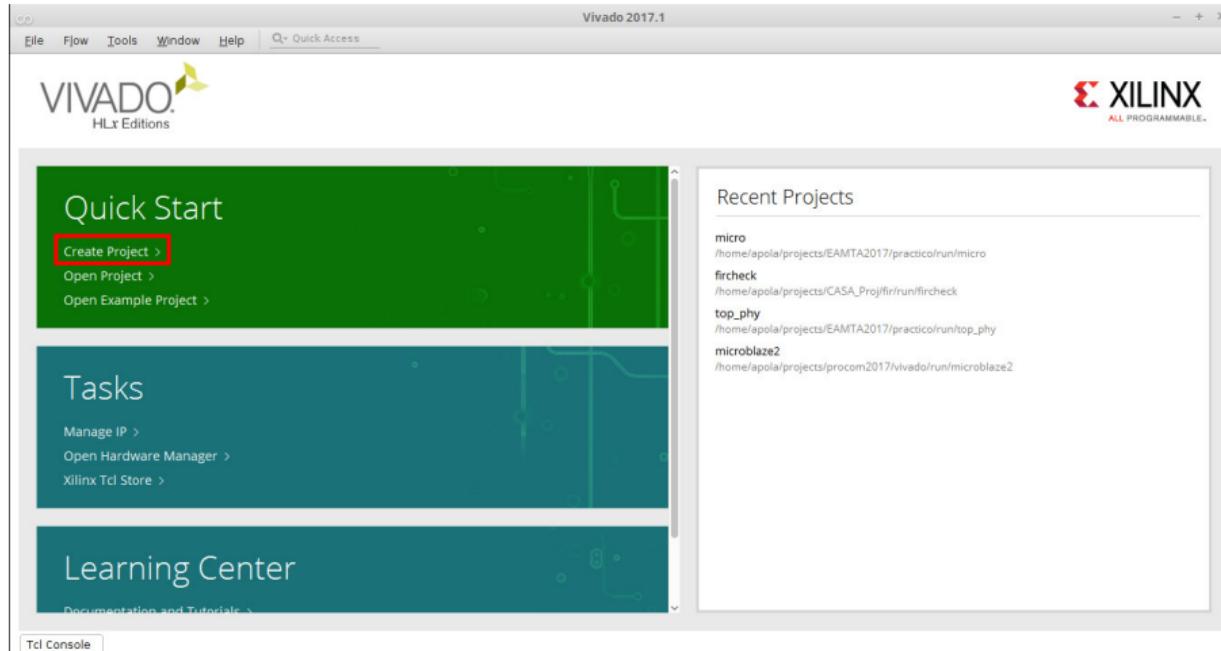
```
1  'define N_LEDS 4
2  'define NB_SW 4
3
4  'timescale 1ns/100ps
5
6  module tb_shiftleds_file();
7
8      parameter N_LEDS    = 'N_LEDS    ;
9      parameter NB_SW     = 'NB_SW     ;
10
11     wire [N_LEDS - 1 : 0] o_led    ;
12     reg [NB_SW - 1 : 0] i_sw     ;
13     reg [NB_SW - 1 : 0] sw_tmp   ;
14     reg ck_rst, CLK100MHZ, reset_tmp;
15
16     integer fid_reset,fid_sw;
17     integer code_error,code_error1;
18     integer          ptr_sw;
19
20     initial begin
21         fid_reset = $fopen("./vectors/reset.out","r");
22         if(fid_reset==0) $stop;
23         fid_sw = $fopen("./vectors/switch.out","r");
24         if(fid_sw==0) $stop;
25         CLK100MHZ = 1'b0 ;
26     end
27
28     always #5 CLK100MHZ = ~CLK100MHZ;
29
30     always@(posedge CLK100MHZ) begin
31         code_error <=
32             $fscanf(fid_reset,"%d",reset_tmp);
33         if(code_error!=1) $stop;
34
35         for(ptr_sw=0;ptr_sw<NB_SW;
36             ptr_sw = ptr_sw+1) begin
37             code_error1 <=
38                 $fscanf(fid_sw,"%d",sw_tmp[(ptr_sw+1)-1 -: 1]);
39             if(code_error1!=1) $stop;
40         end
41
42         ck_rst <= reset_tmp;
43         i_sw   <= sw_tmp;
44         $display("%d",ck_rst);
45     end
46
47     shiftleds
48         u_shiftleds
49             (.o_led      (o_led)      ,
50              .i_sw       (i_sw)       ,
51              .ck_rst    (ck_rst)    ,
52              .CLK100MHZ (CLK100MHZ));
53
54     endmodule // tb_shiftleds
```

Herramienta Vivado



Herramienta Vivado

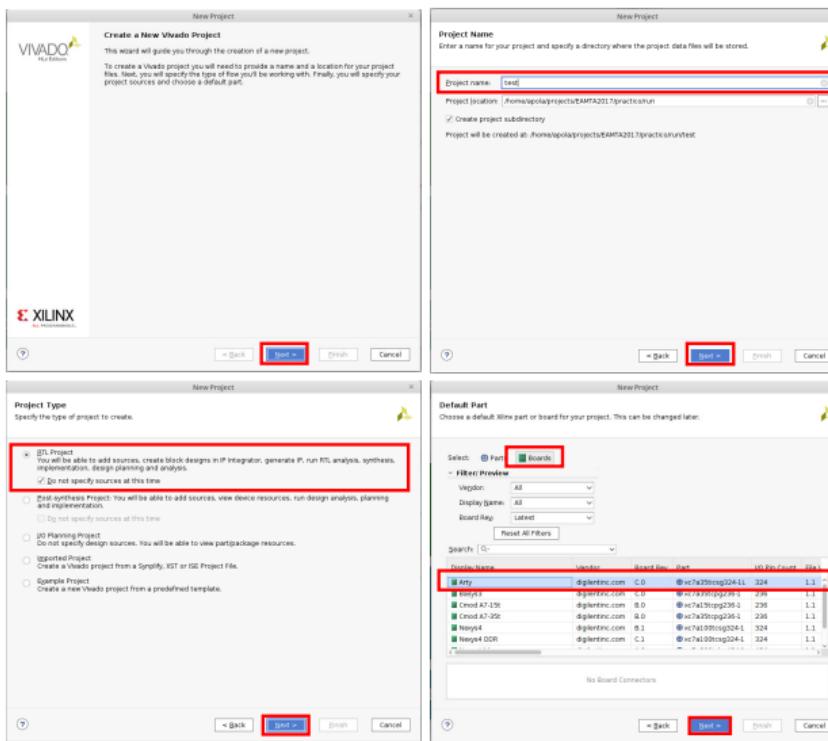
Vivado



Crea un nuevo proyecto

Herramienta Vivado

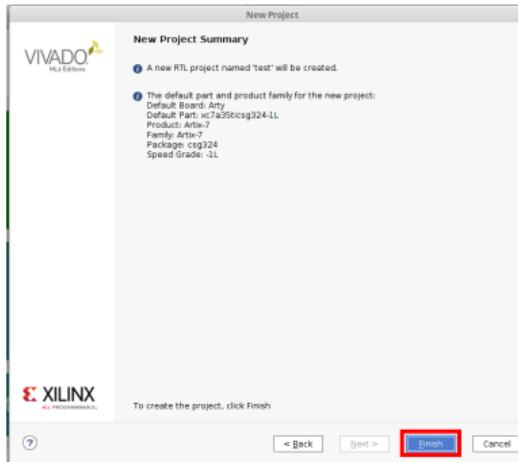
Vivado



Configura el kit de trabajo

Herramienta Vivado

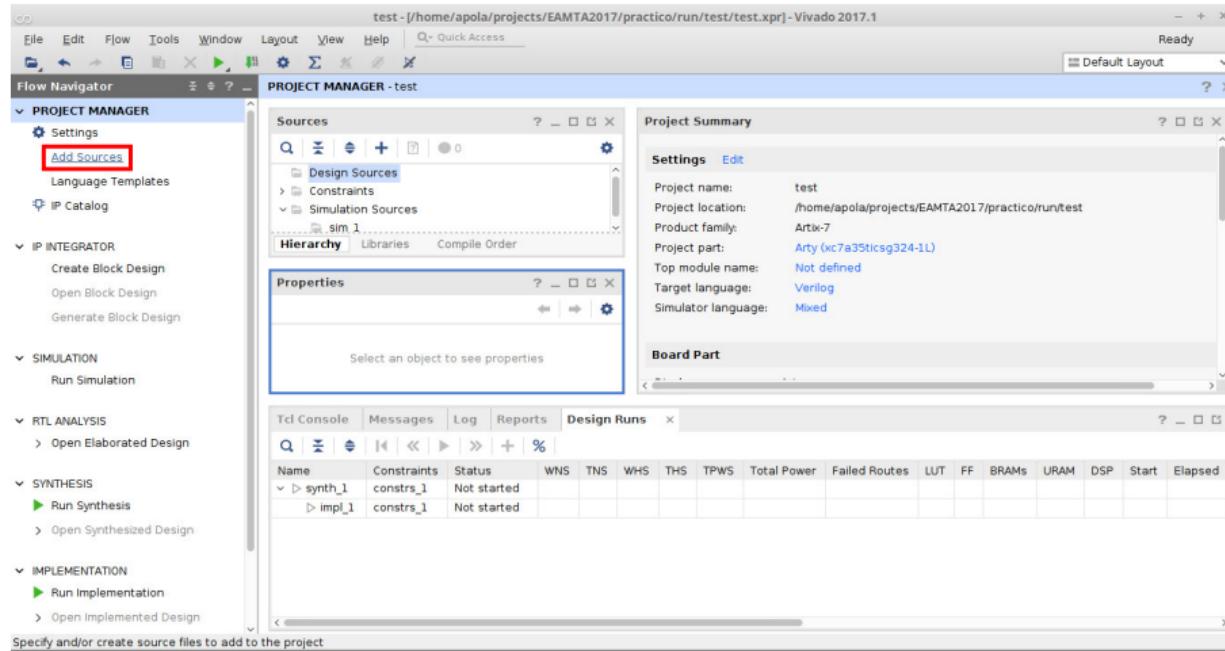
Vivado



Configura el kit de trabajo

Herramienta Vivado

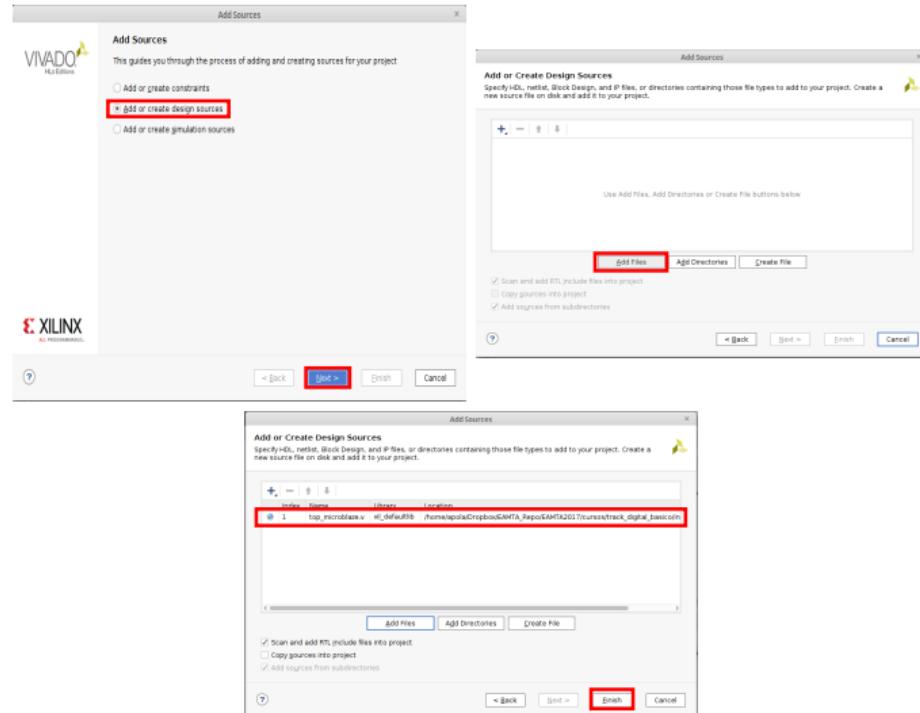
Vivado



Aggrega nuevas fuentes a *Design Sources*

Herramienta Vivado

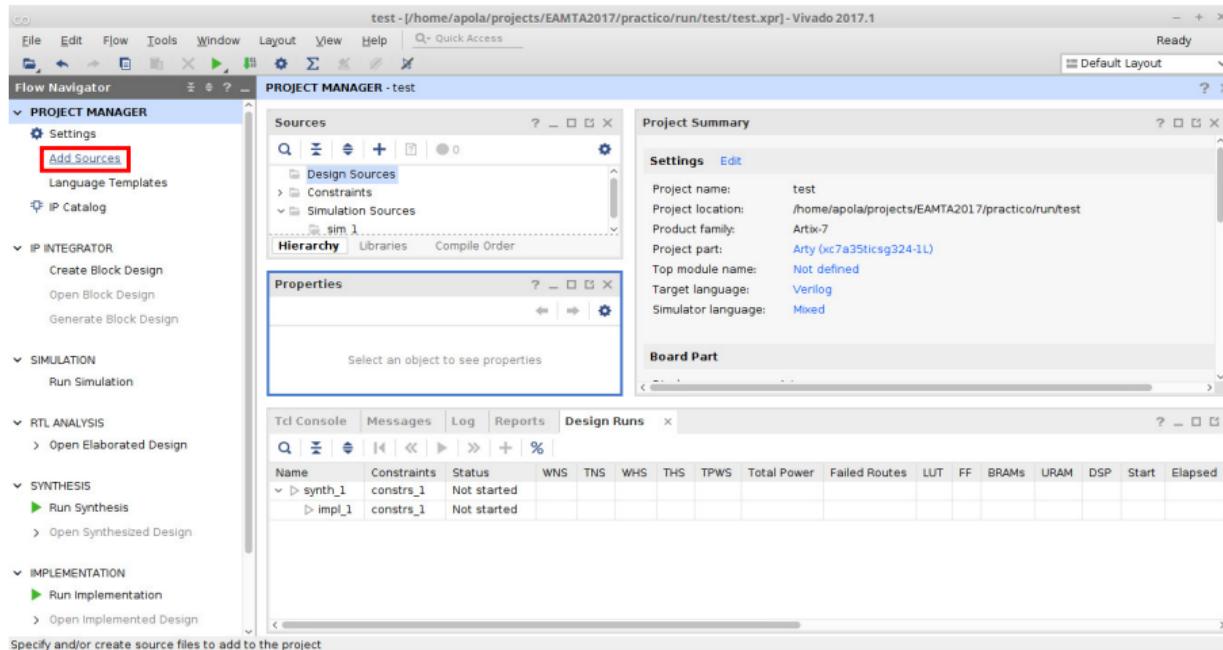
Vivado



Seleccionar todos los archivos verilog relacionados al diseño

Herramienta Vivado

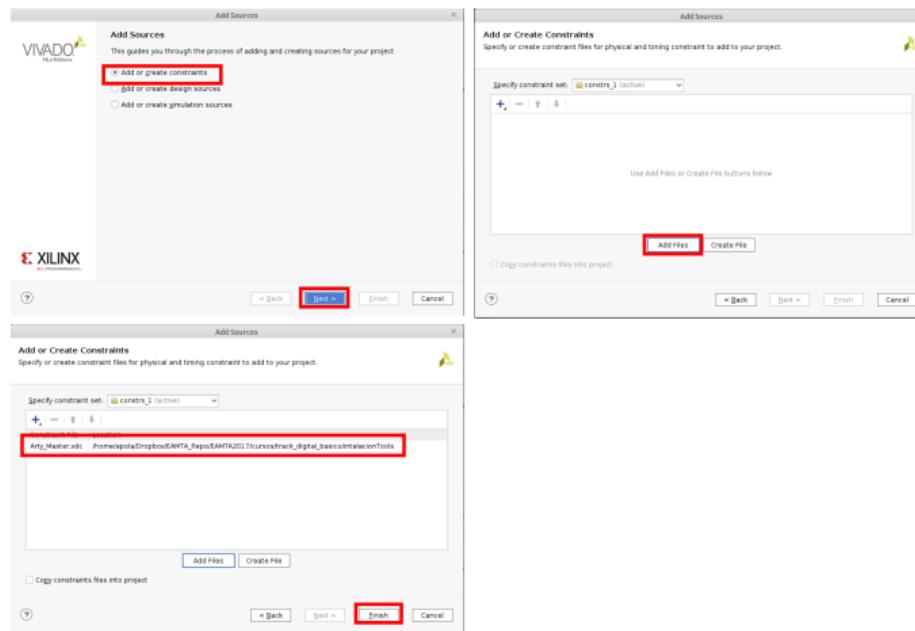
Vivado



Agrega nuevas fuentes a *Constraints*

Herramienta Vivado

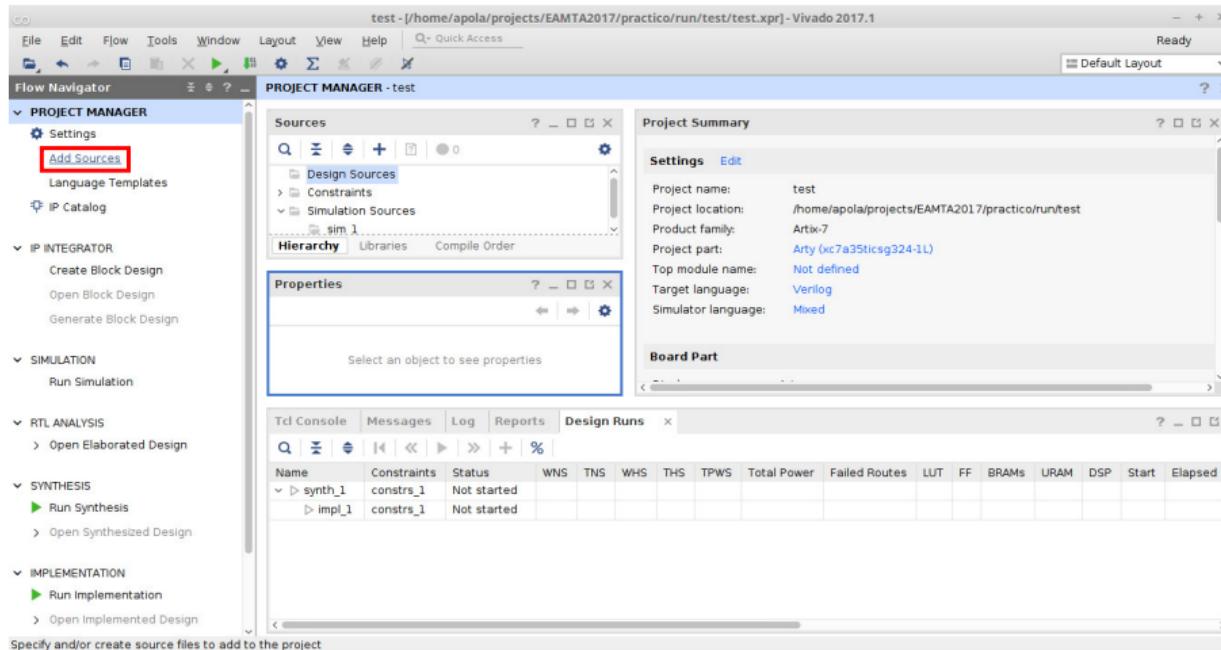
Vivado



Selecciona el archivo xdc

Herramienta Vivado

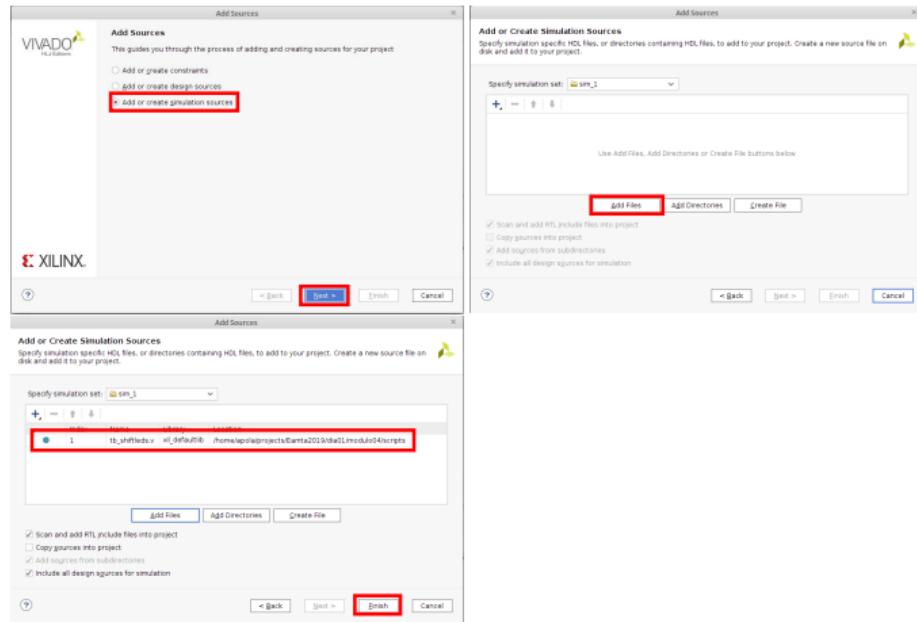
Vivado



Agrega nuevas fuentes a *Simulation Sources*

Herramienta Vivado

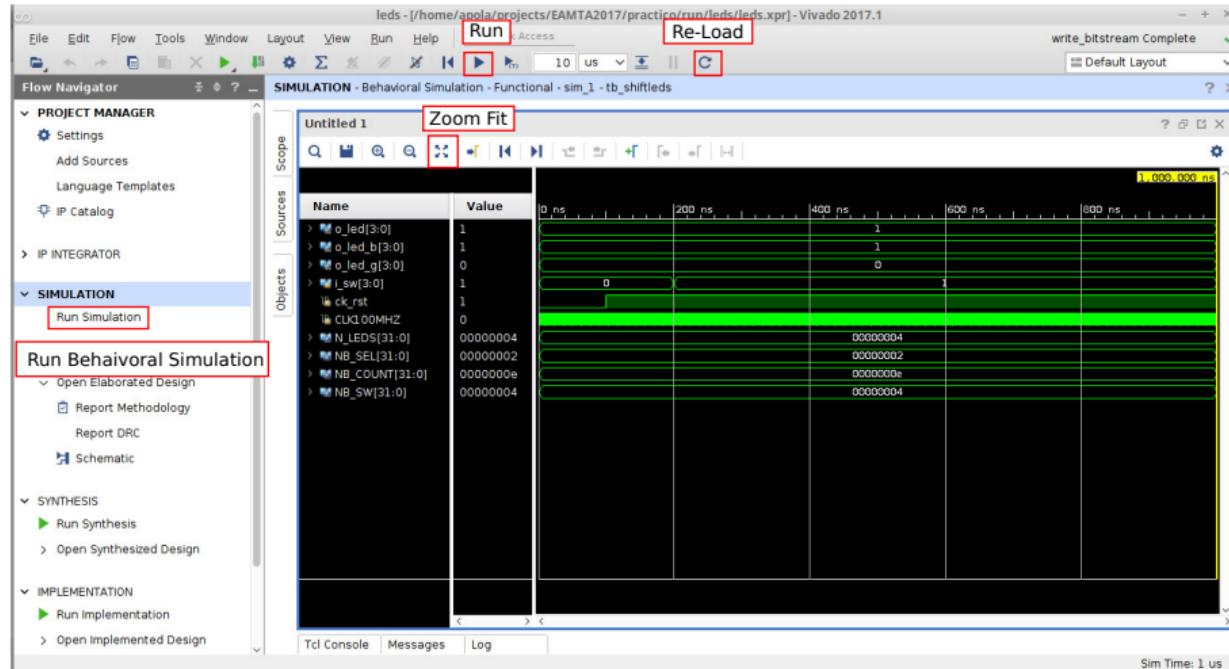
Vivado



Selecciona el archivo verilog para simulación

Herramienta Vivado

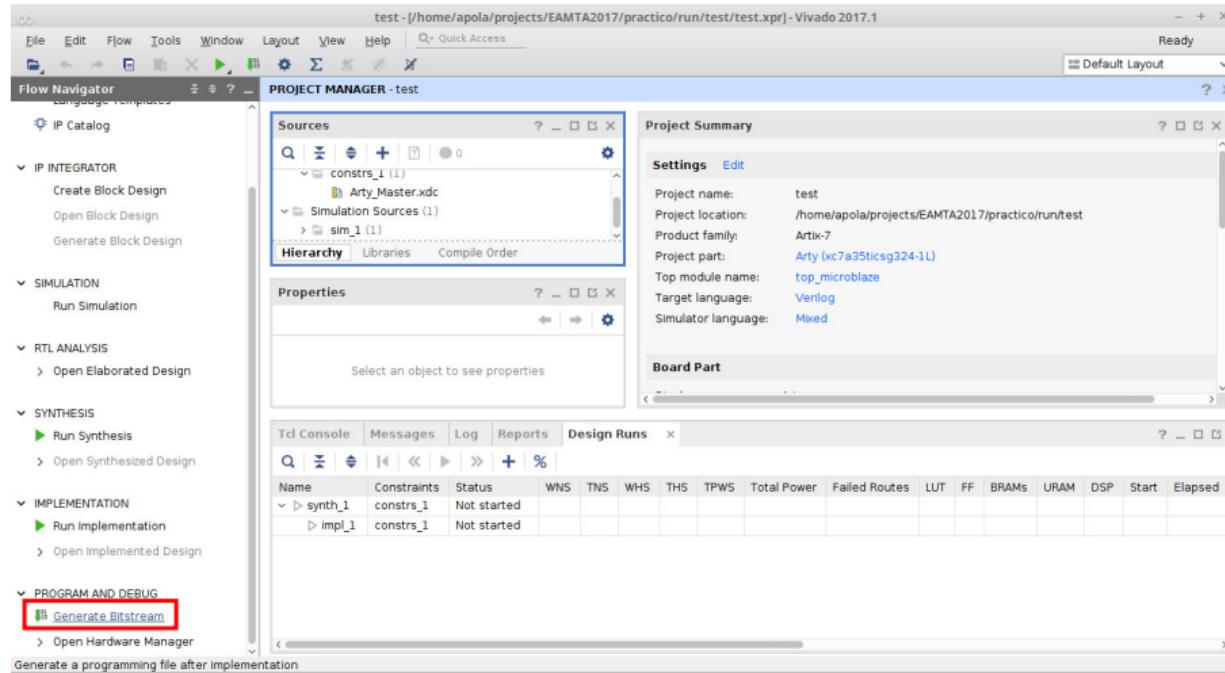
Run Simulation - Run Behavioral Simulation



Simulando el comportamiento del diseño

Herramienta Vivado

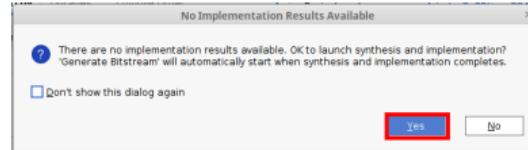
Vivado



Generar el *bitstream*

Herramienta Vivado

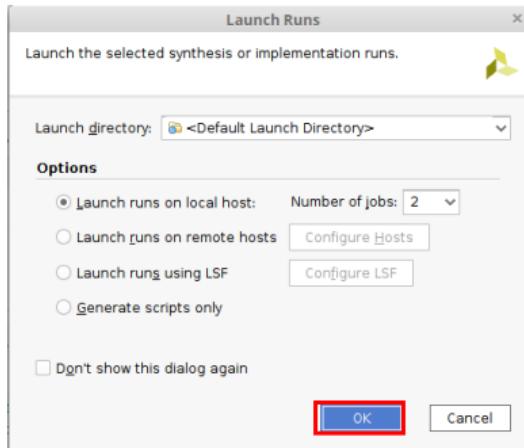
Vivado



Generar el *bitstream*

Herramienta Vivado

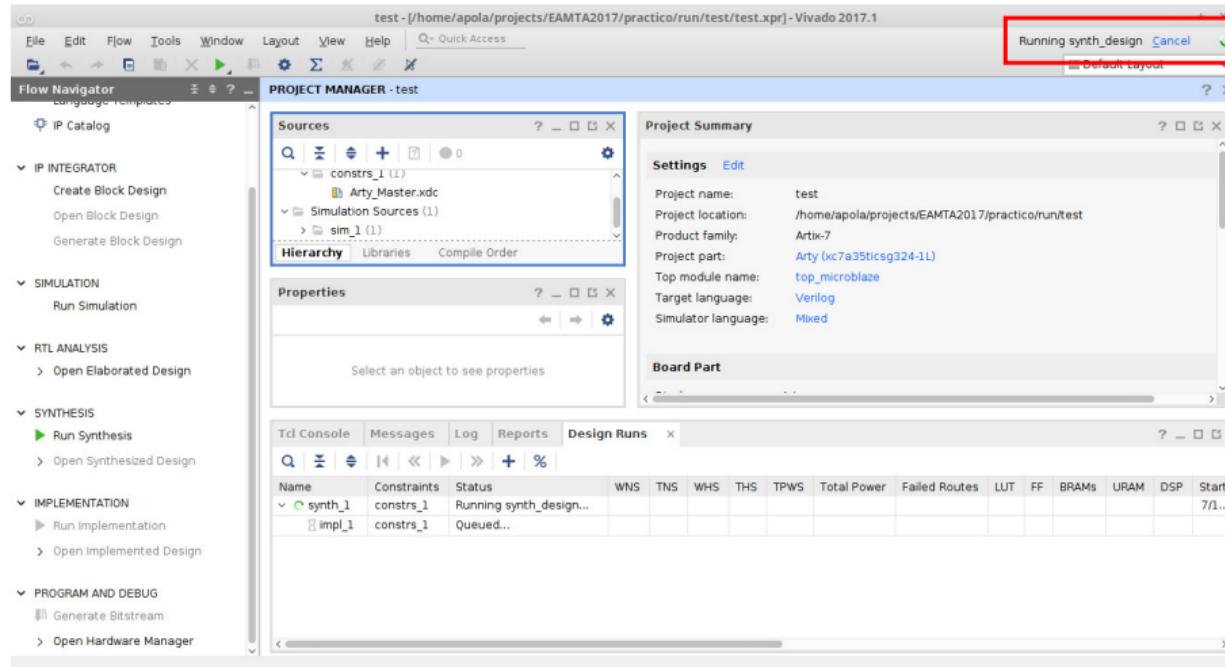
Vivado



Generar el *bitstream*

Herramienta Vivado

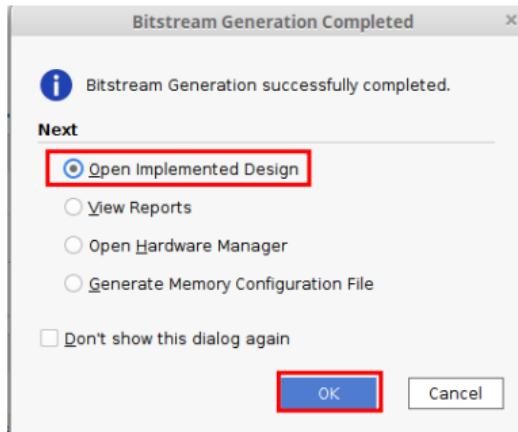
Vivado



Ejecutando tareas de implementación

Herramienta Vivado

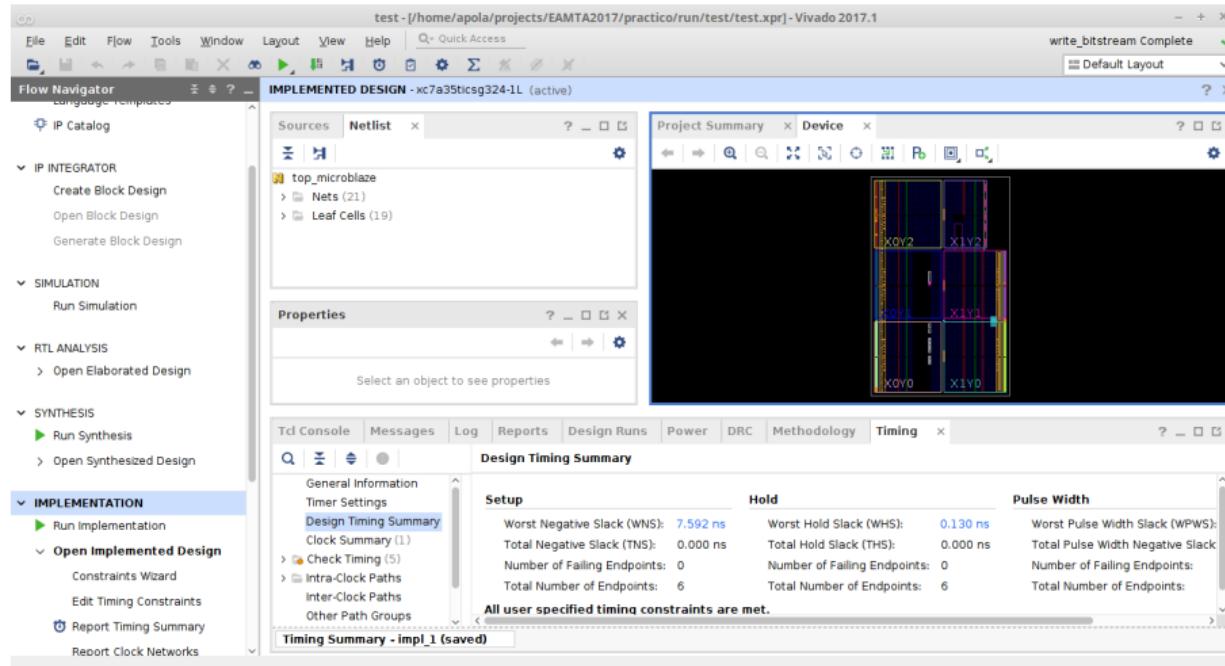
Vivado



Abriendo el modelo implementado

Herramienta Vivado

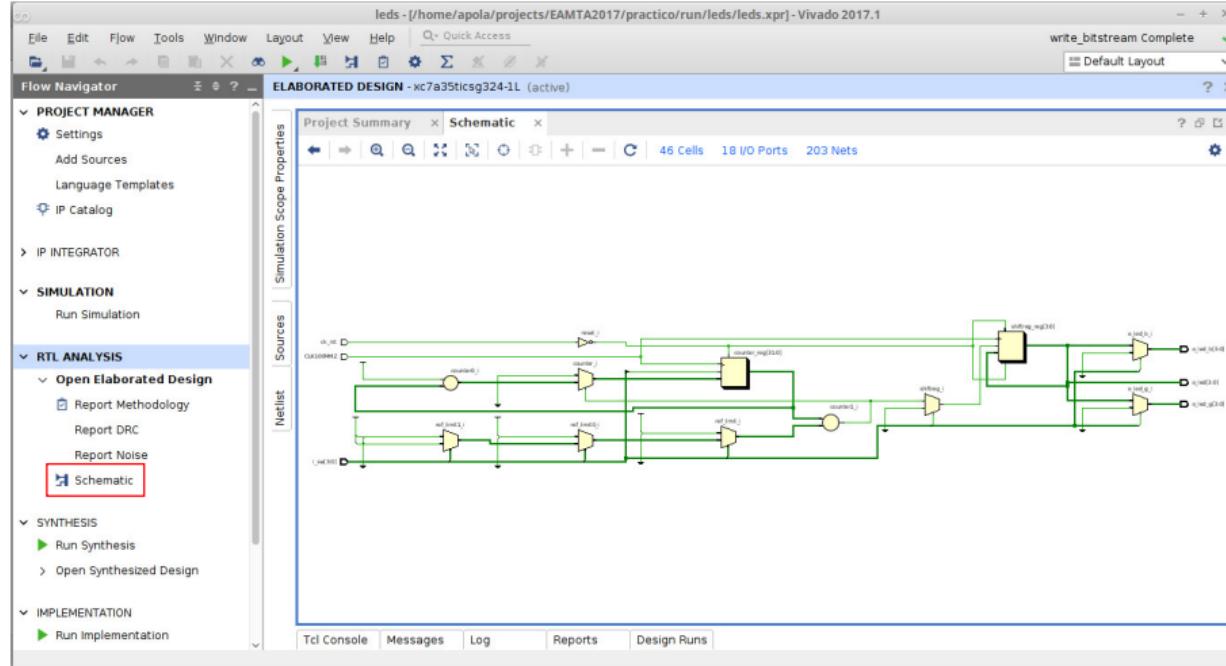
Vivado



Diseño implementado

Herramienta Vivado

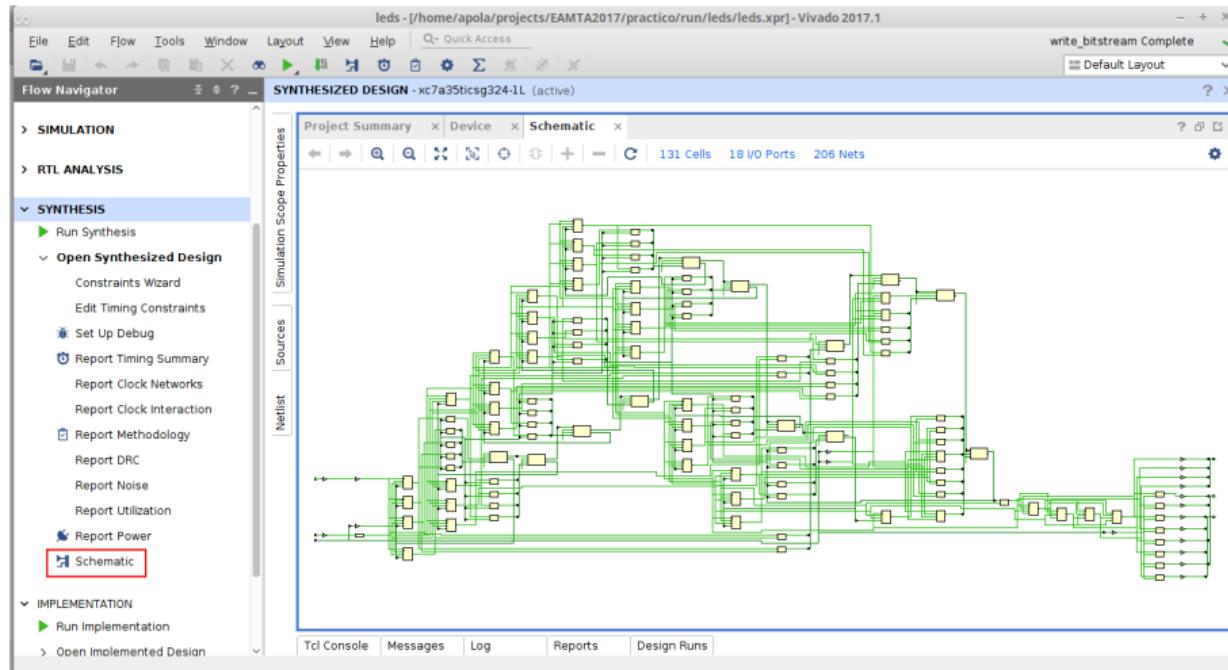
RTL Schematic



Esquemático RTL

Herramienta Vivado

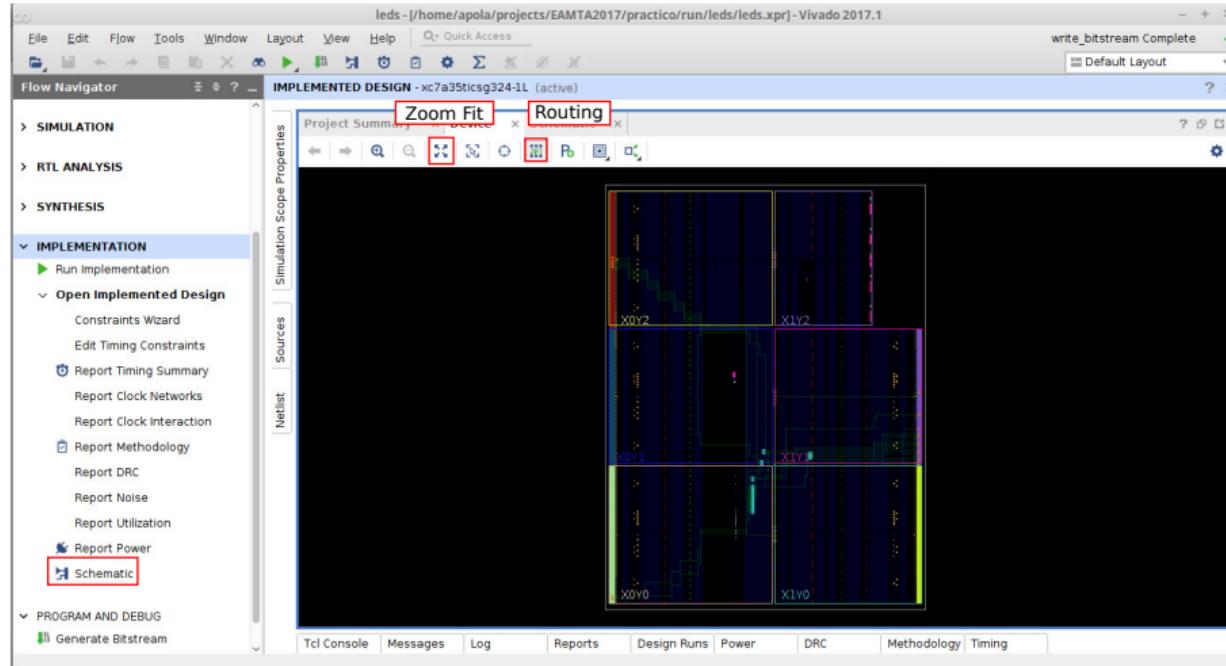
Synthesis - Schematic



Esquemático Implementación

Herramienta Vivado

Implementation - Schematic

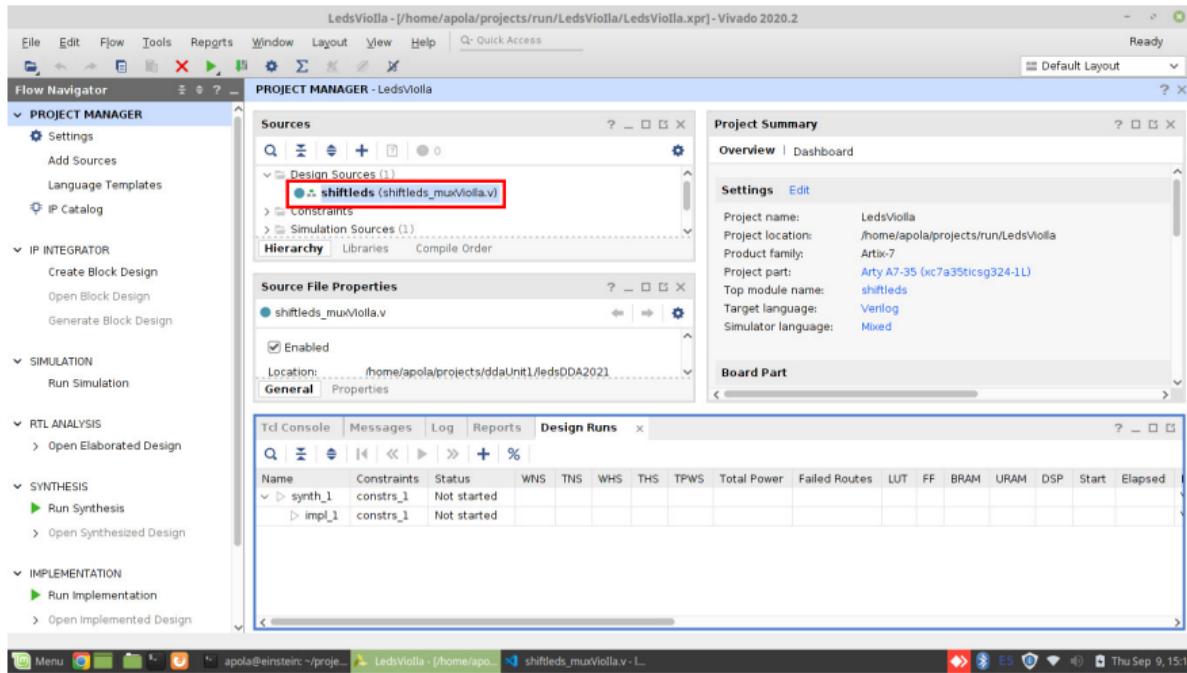


Implementación

Instancia de VIO e ILA

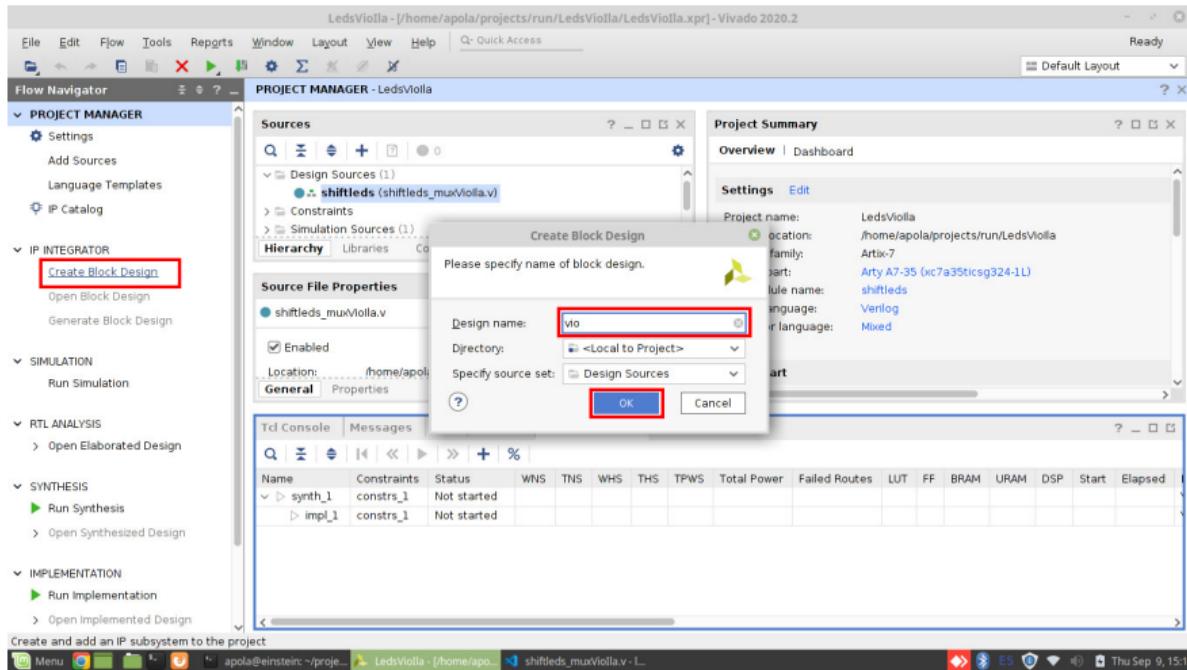


Instancia de VIO e ILA



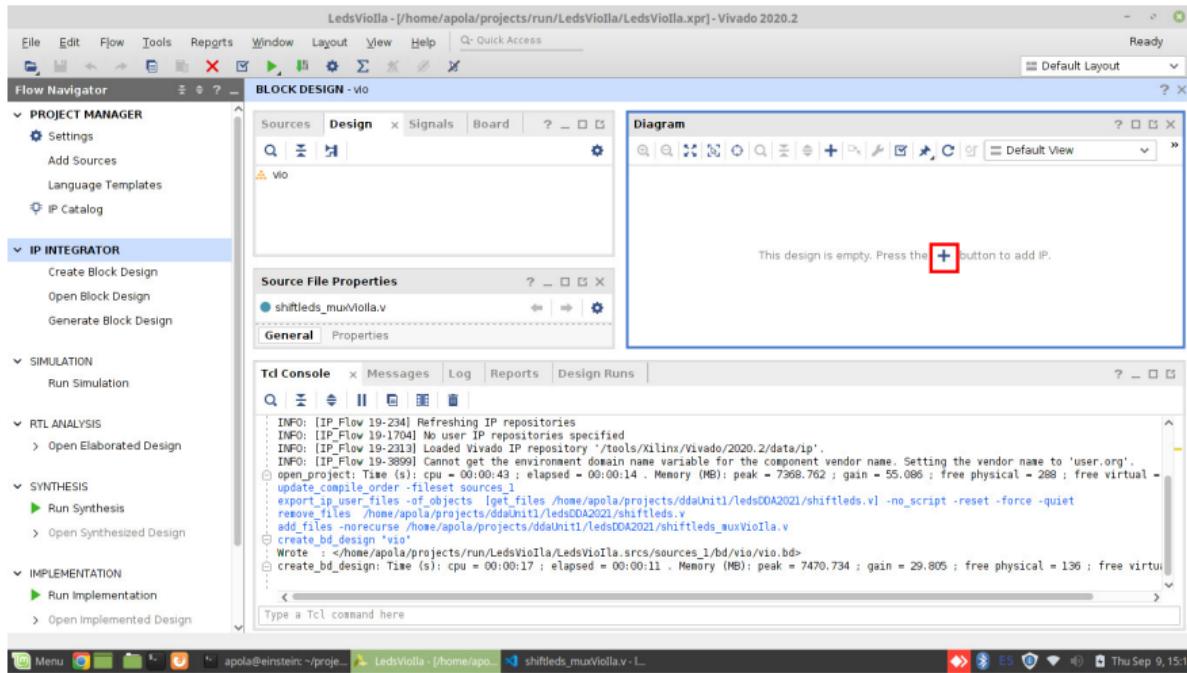
Incluir los archivos **shiftleds_muxViolla.v** y **xdc**.

Instancia de VIO e ILA



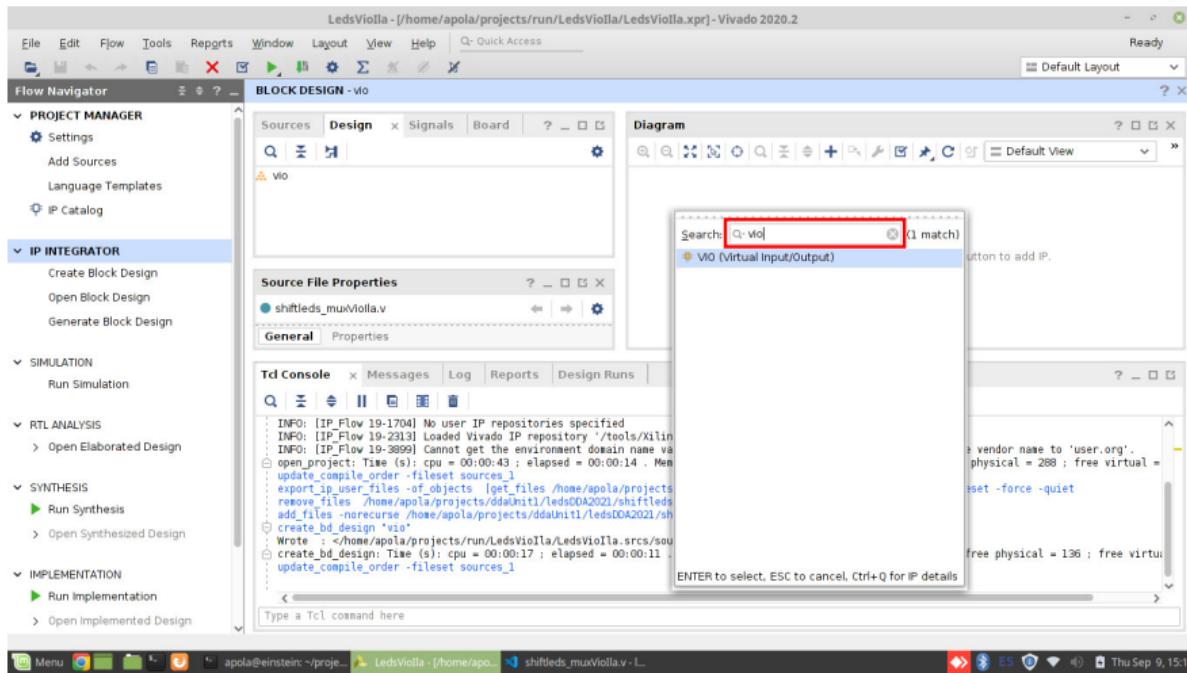
Crear el bloque VIO.

Instancia de VIO e ILA



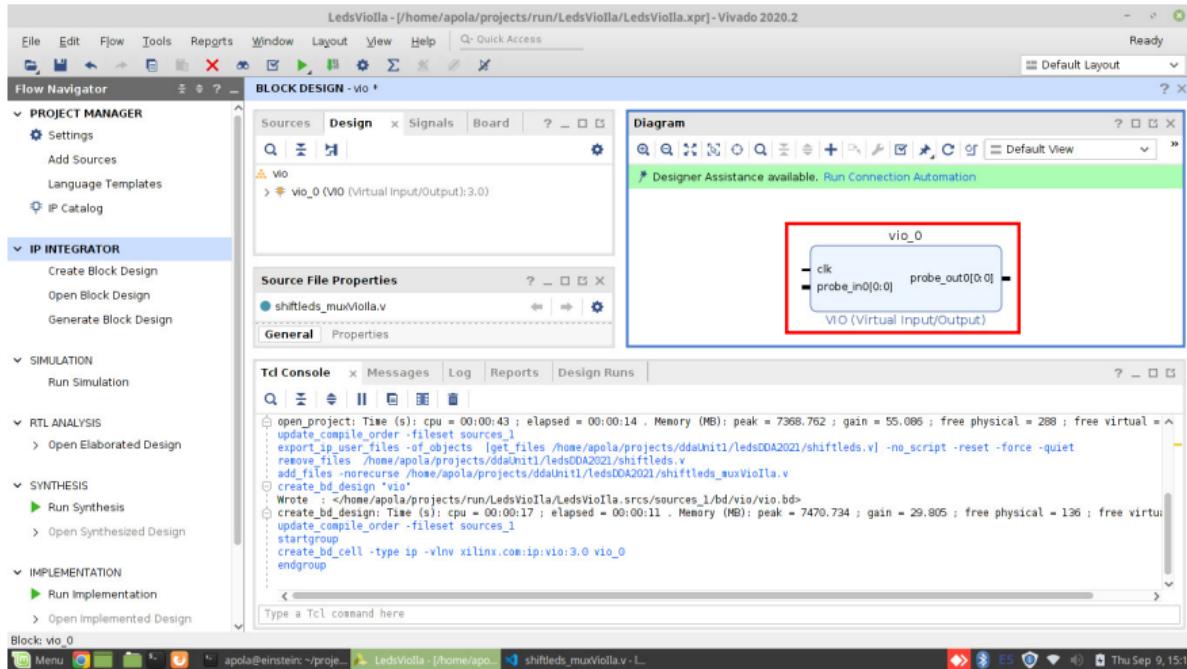
Buscar el IP VIO.

Instancia de VIO e ILA



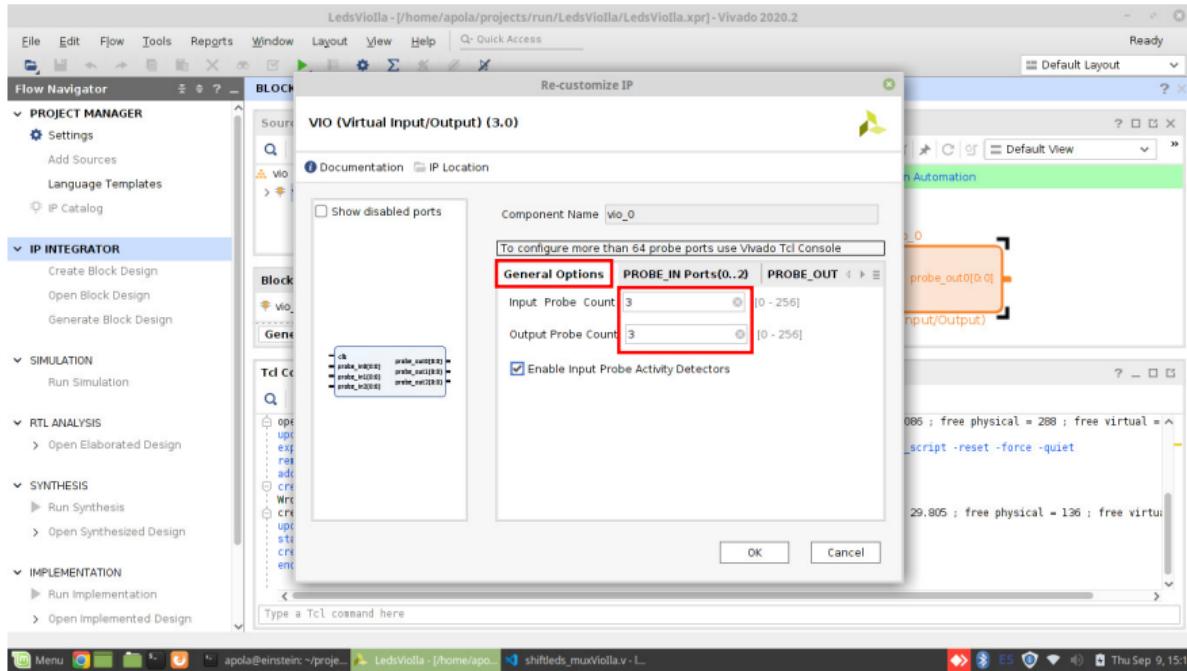
Buscar el IP VIO.

Instancia de VIO e ILA



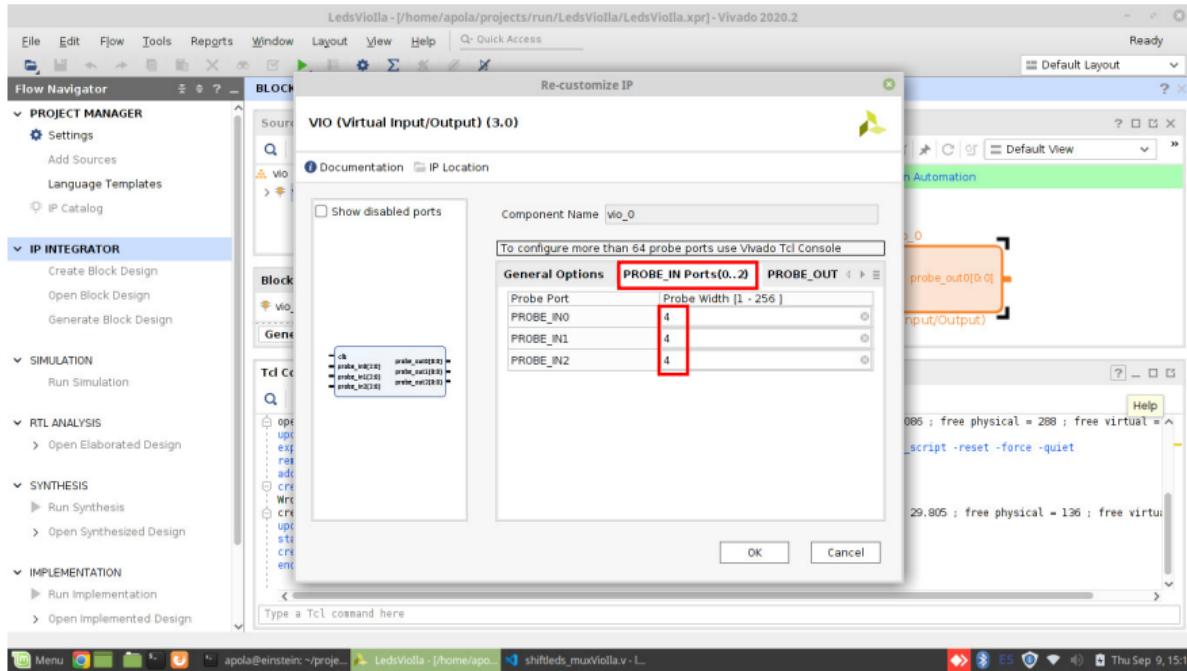
Doble click sobre el IP para entrar a la configuración.

Instancia de VIO e ILA



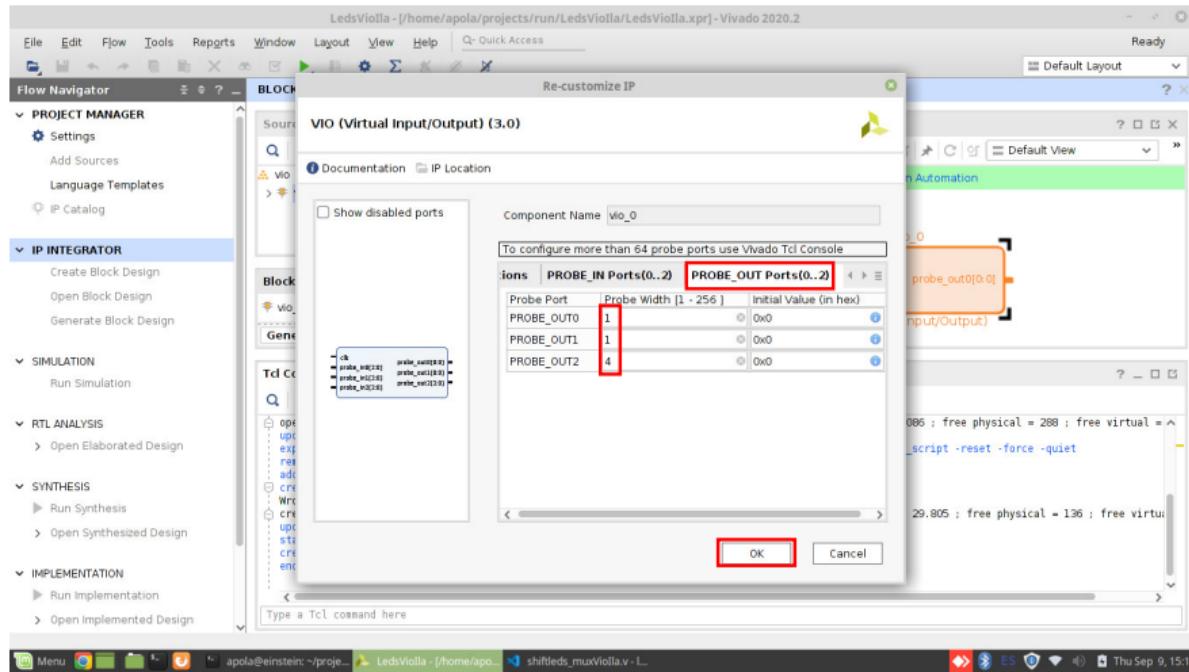
Asignar el número de puertos.

Instancia de VIO e ILA



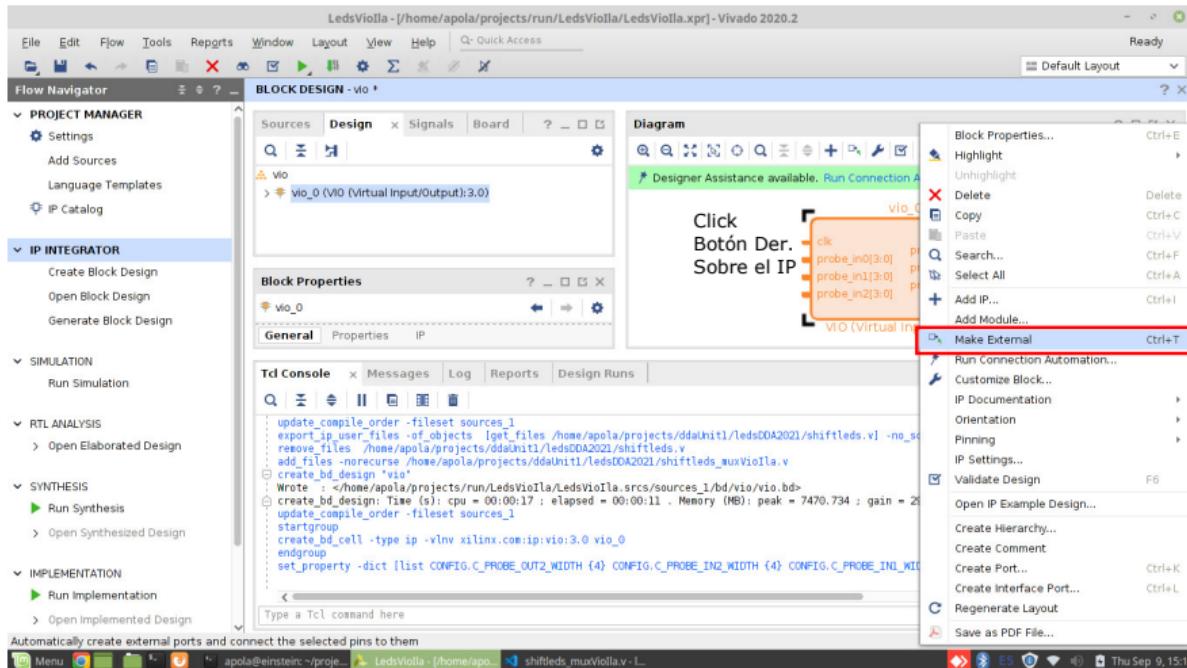
Asignar el número de bits por cada puerto.

Instancia de VIO e ILA



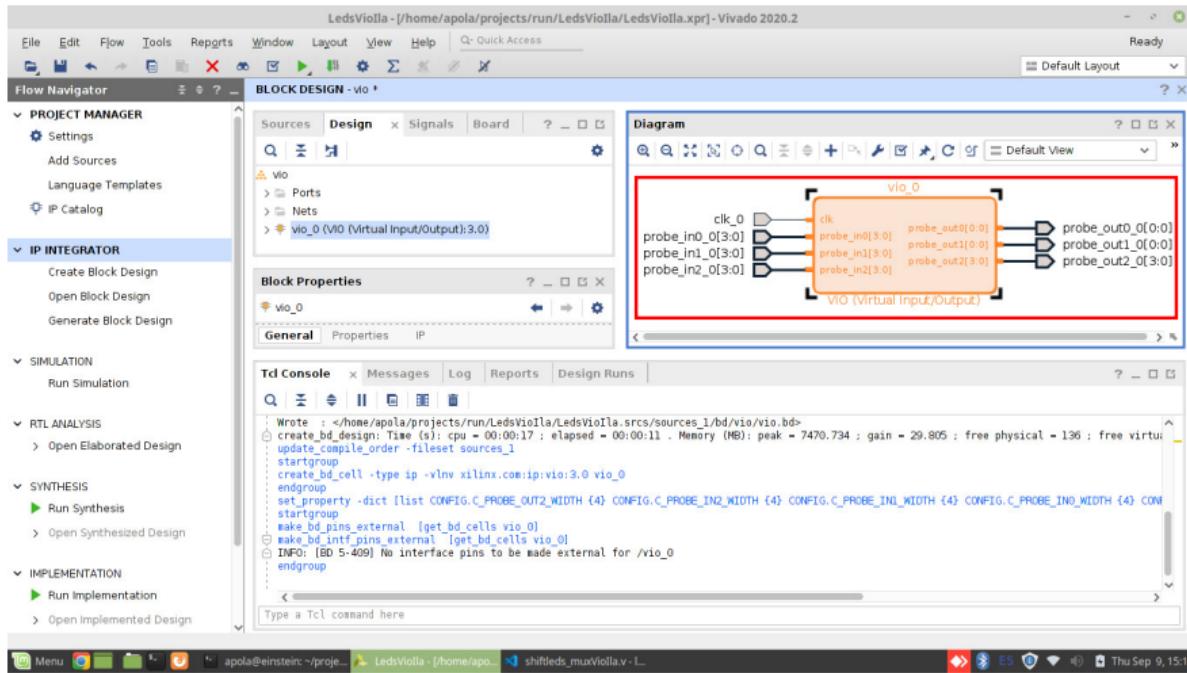
Asignar el número de bits por cada puerto.

Instancia de VIO e ILA



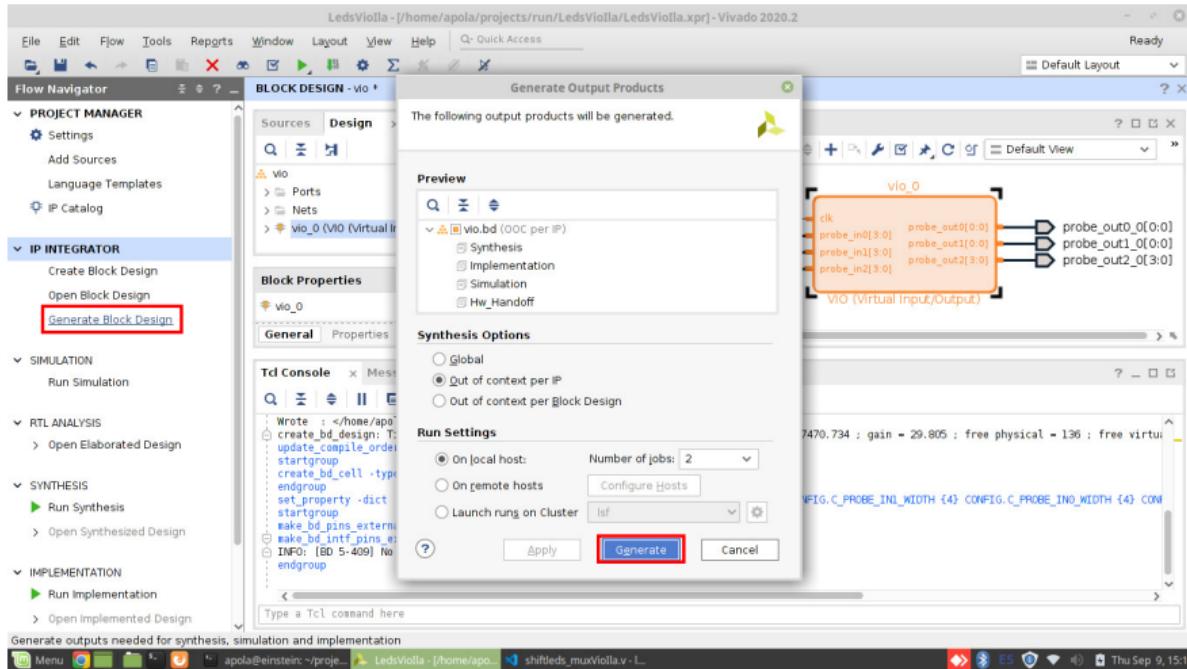
Conectar los puertos.

Instancia de VIO e ILA



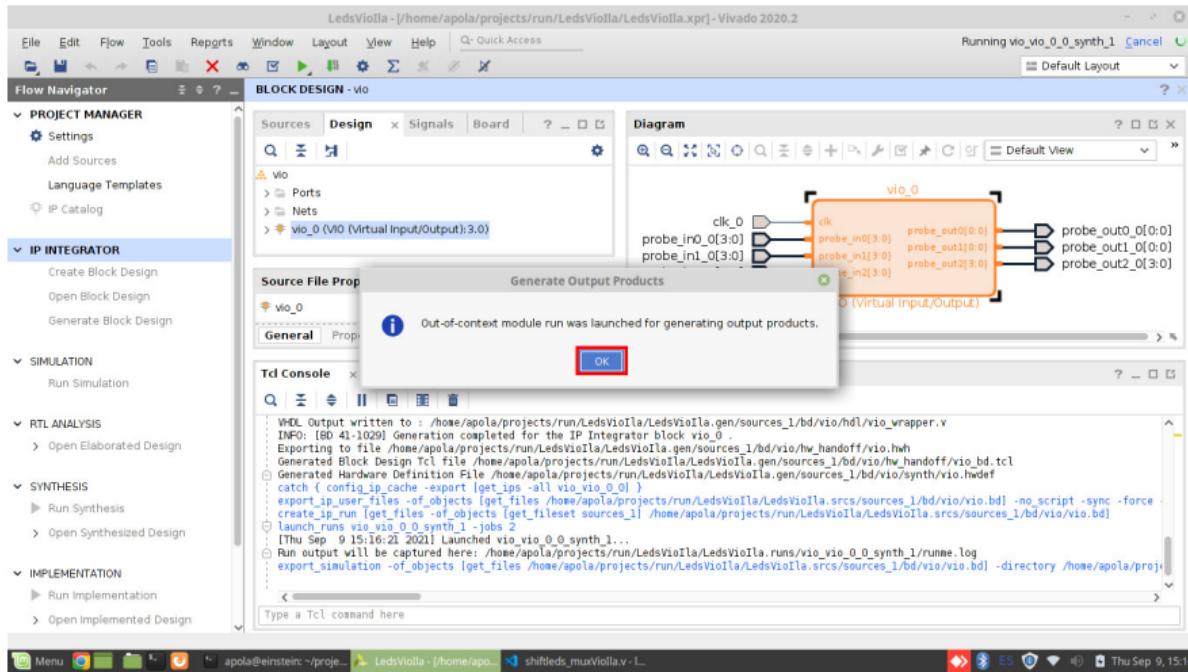
IP conectado.

Instancia de VIO e ILA



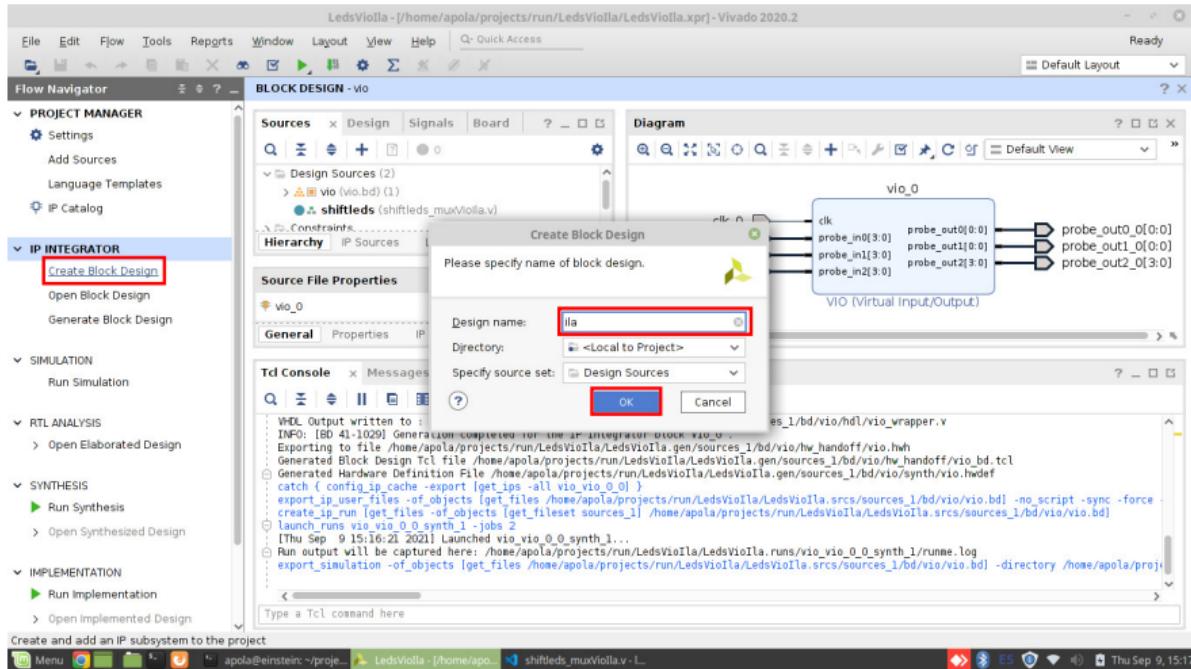
Generar el IP y compilarlo.

Instancia de VIO e ILA



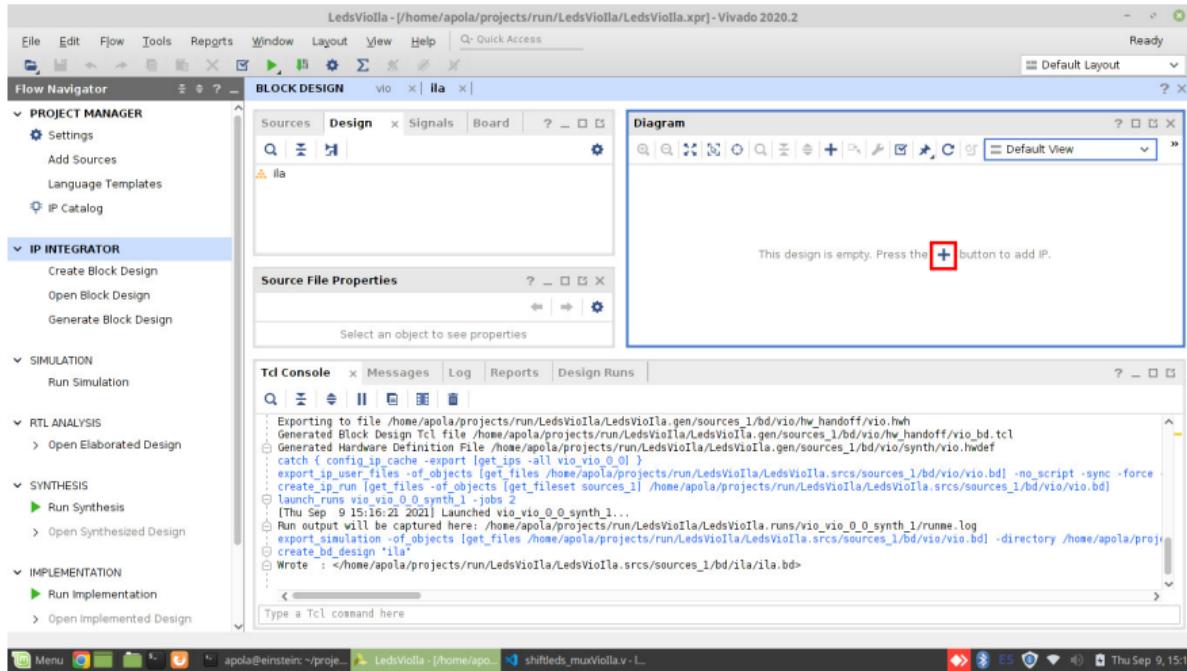
Generar el IP y compilarlo.

Instancia de VIO e ILA



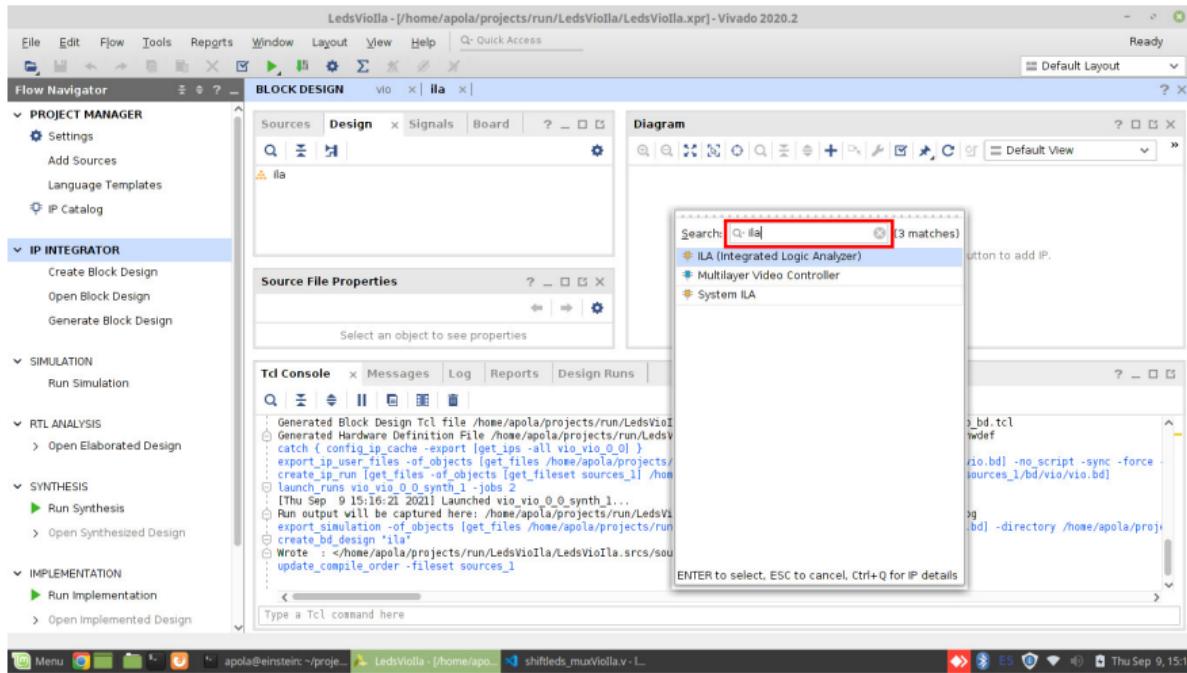
Crear el bloque ILA.

Instancia de VIO eILA



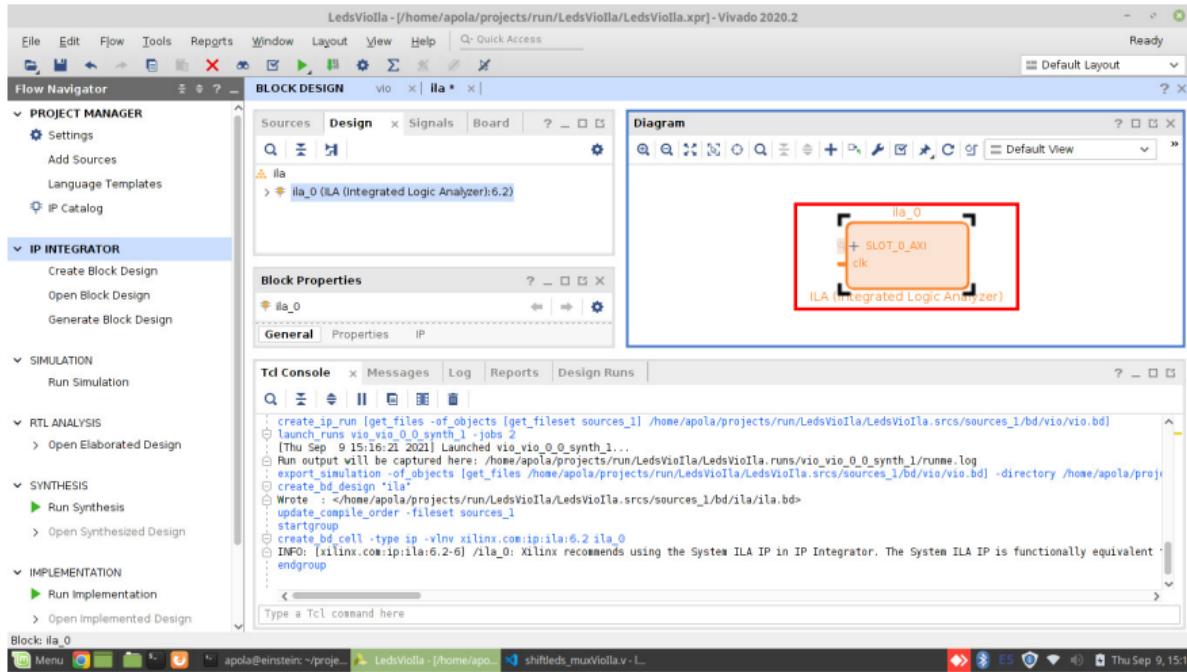
Buscar el IP ILA.

Instancia de VIO eILA



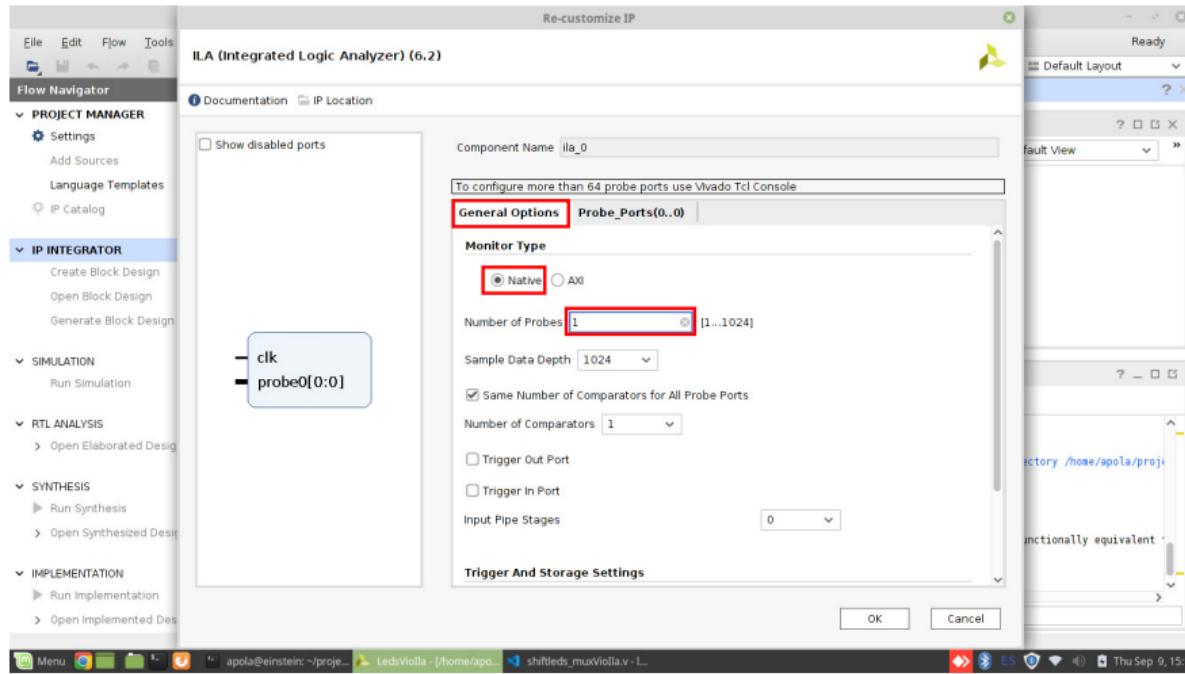
Buscar el IP ILA.

Instancia de VIO e ILA



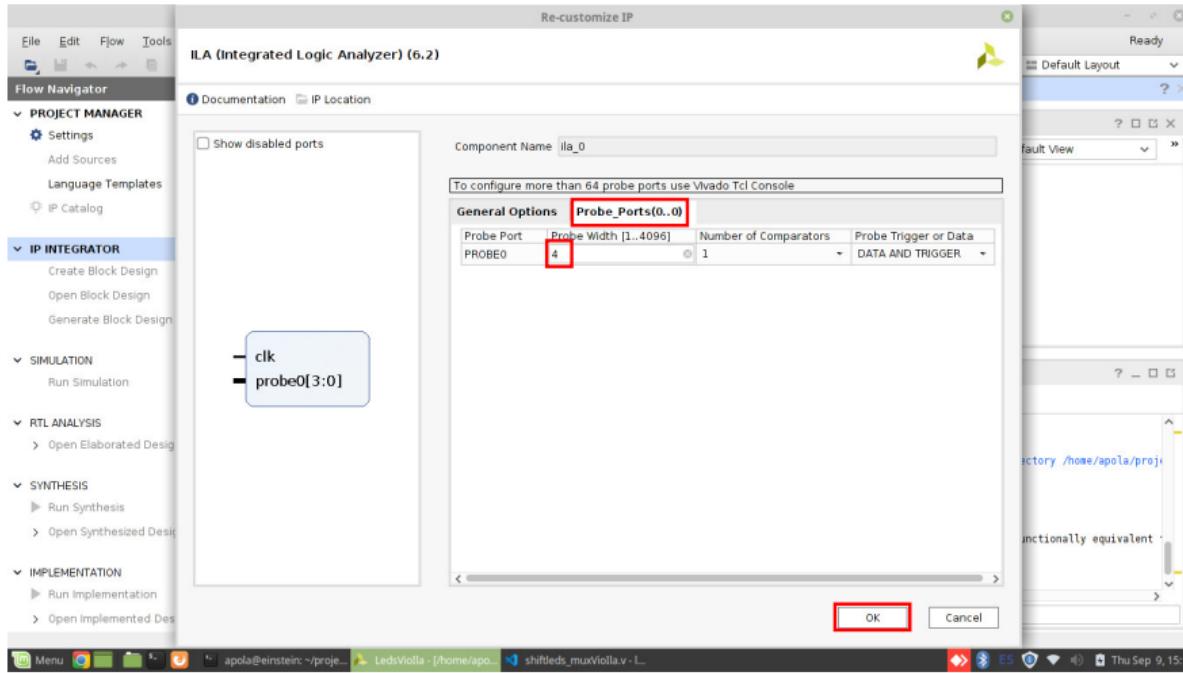
Doble click sobre el IP para entrar a la configuración.

Instancia de VIO eILA



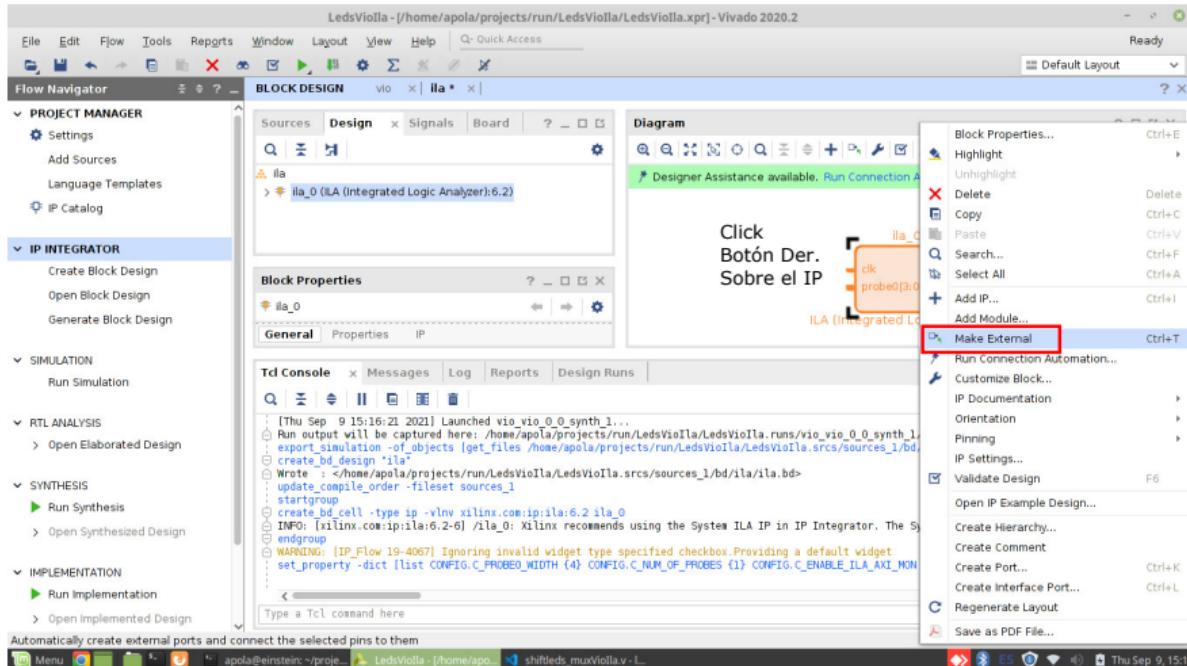
Asignar el tipo de puerto y el número de puntas de prueba.

Instancia de VIO eILA



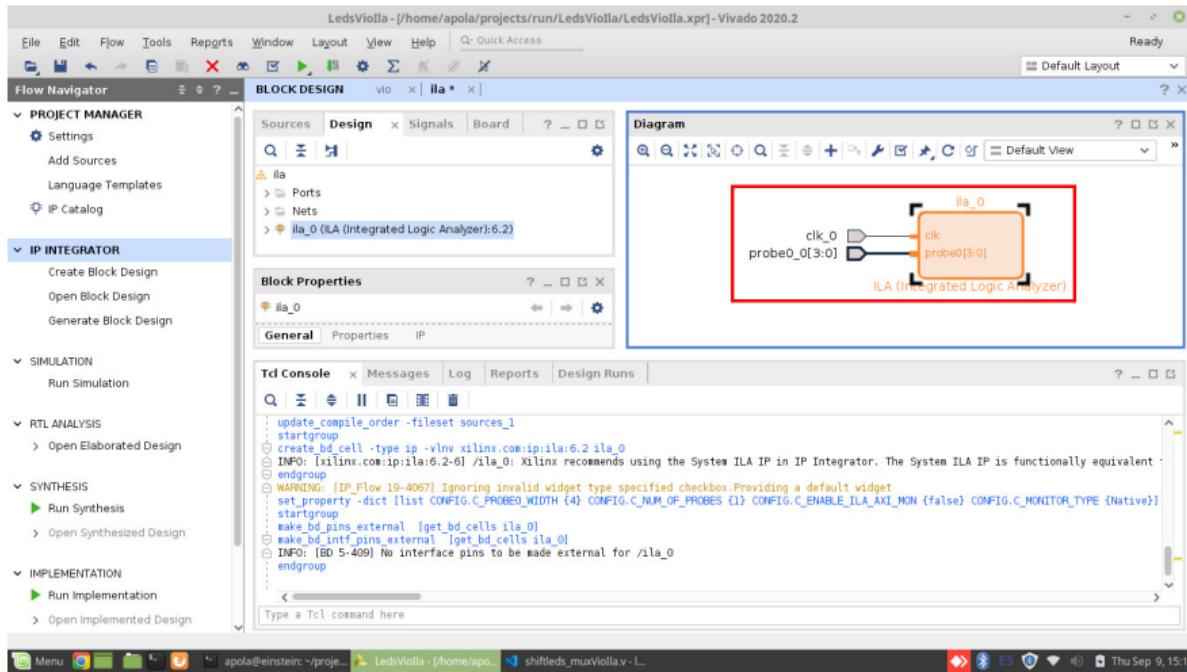
Asignar el número de bits.

Instancia de VIO eILA



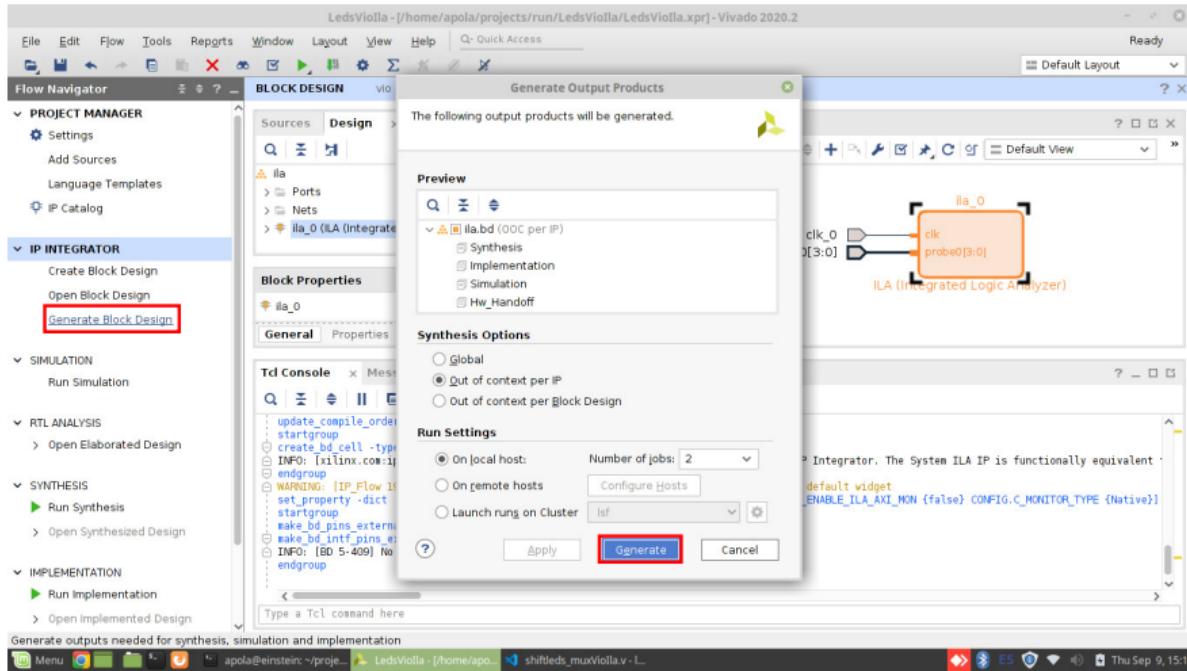
Conegar los puertos.

Instancia de VIO eILA



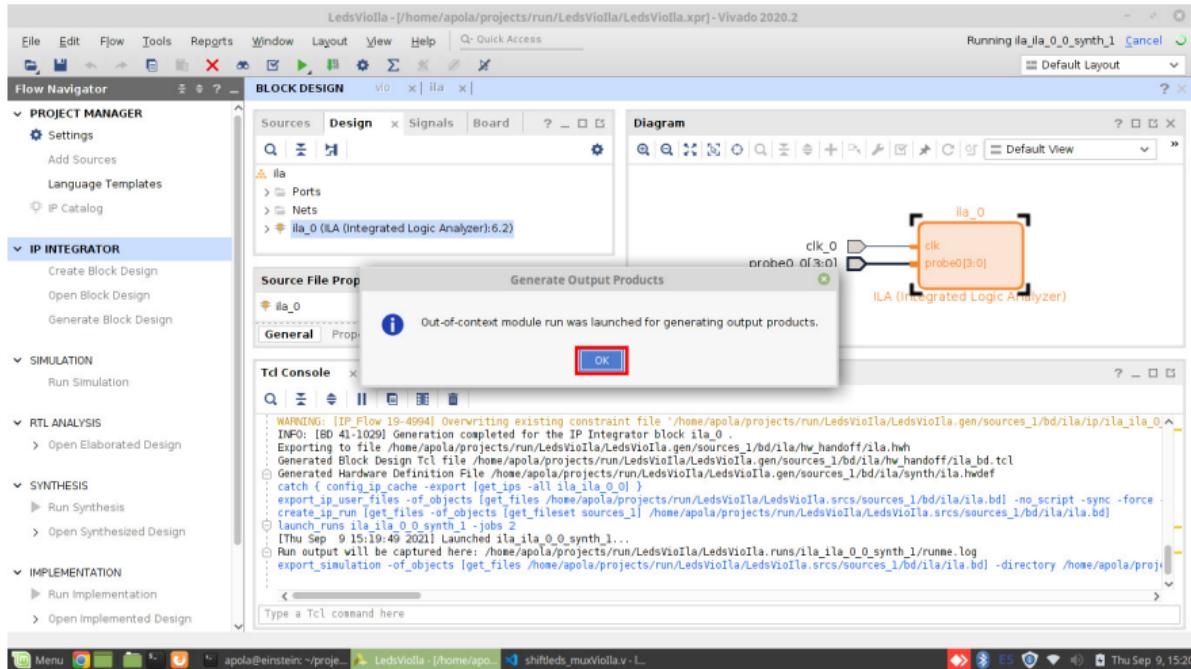
IP conectado.

Instancia de VIO eILA



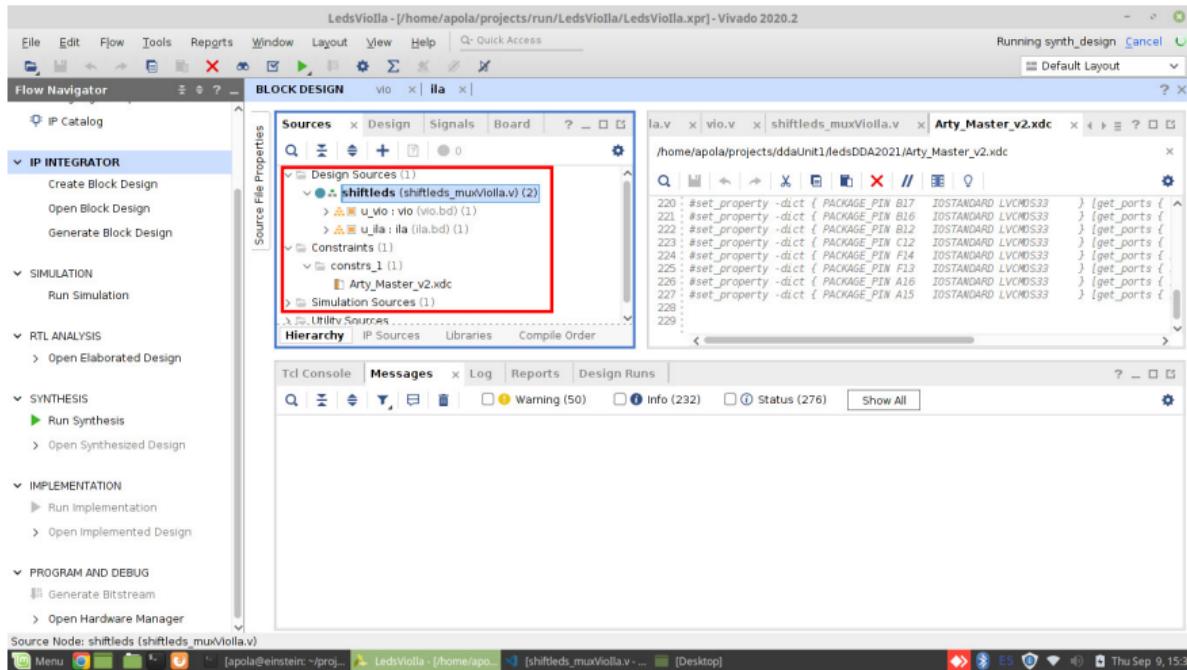
Generar el IP y compilarlo.

Instancia de VIO e ILA



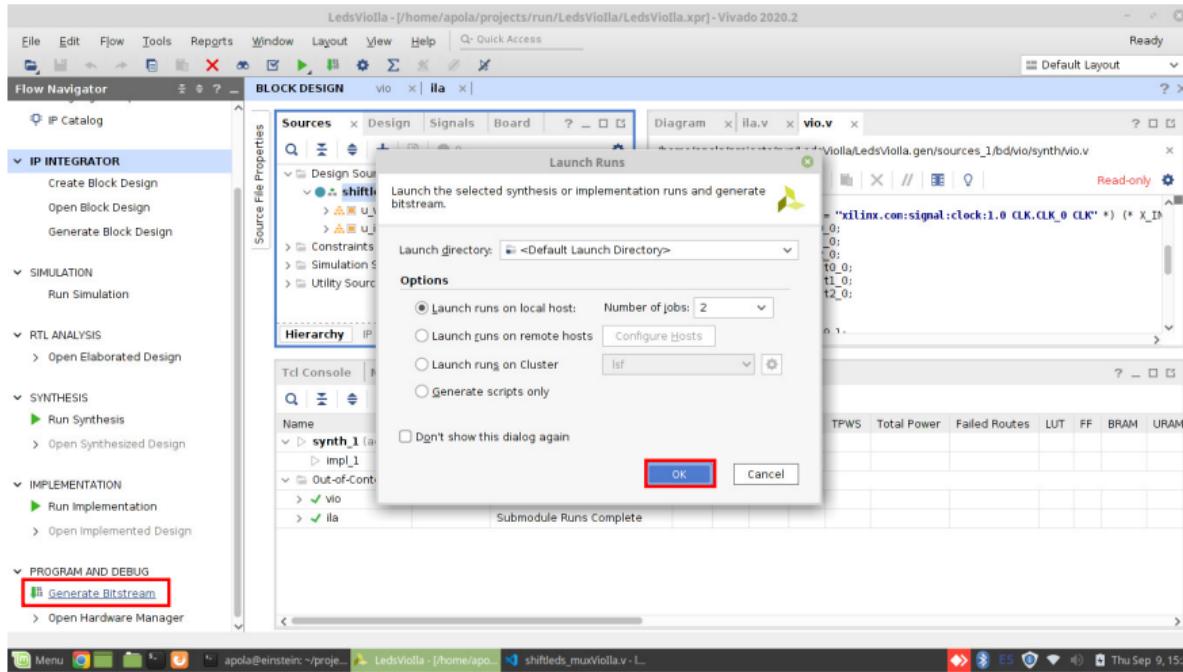
Generar el IP y compilarlo.

Instancia de VIO eILA



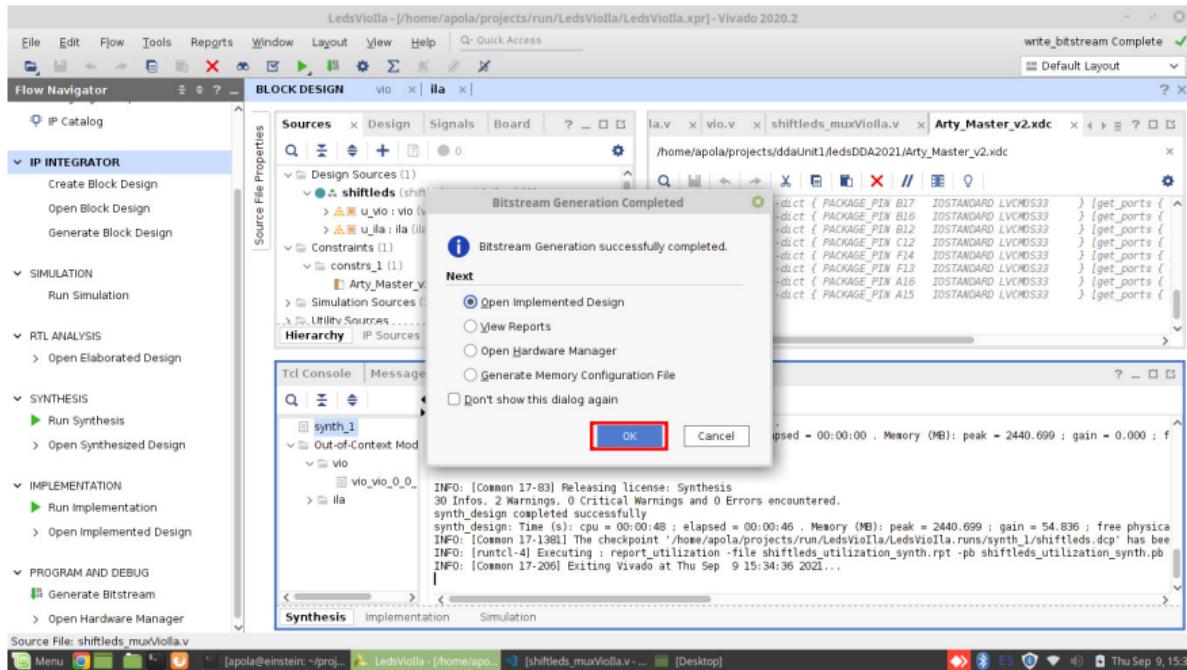
Jerarquía de archivos.

Instancia de VIO e ILA



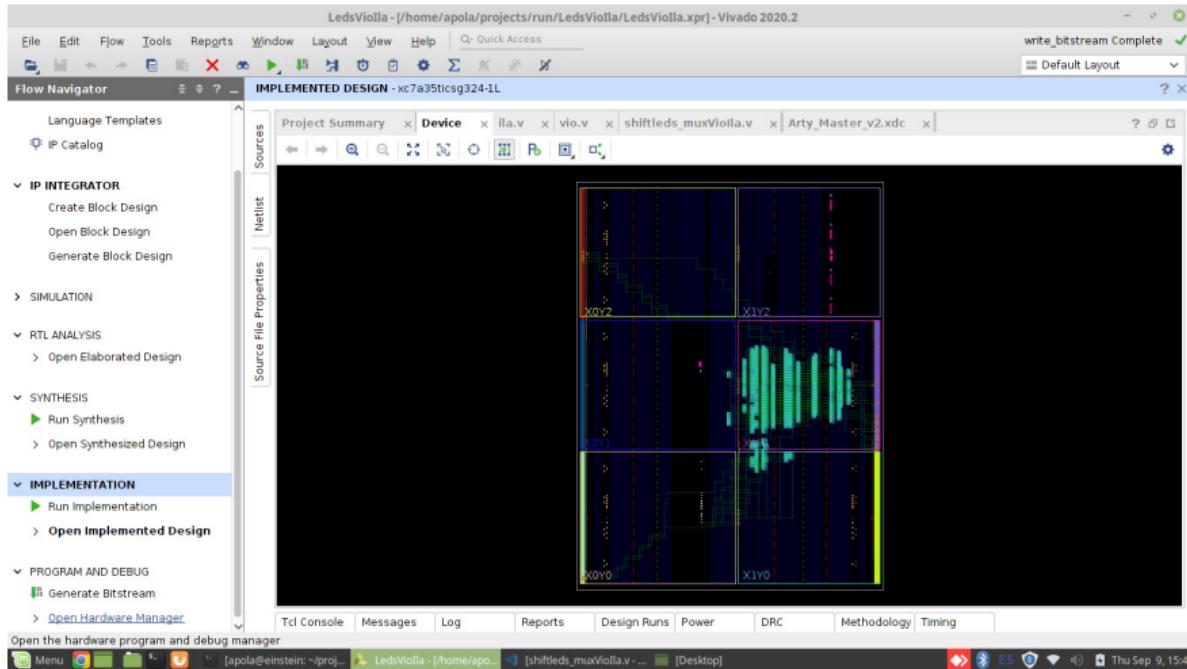
Crear el bitstream.

Instancia de VIO e ILA



Revisar el diseño.

Instancia de VIO eILA



Revisar el diseño.

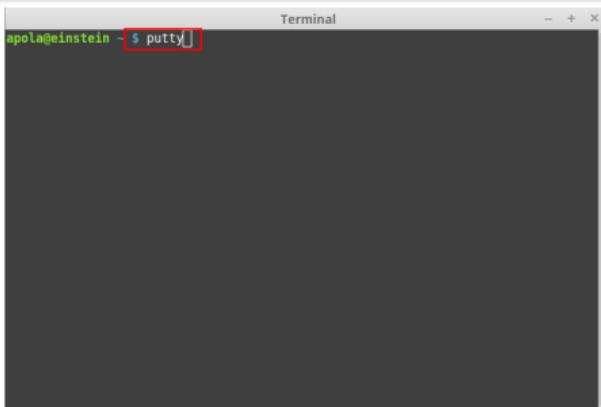
Tunel y Asignación de FPGA



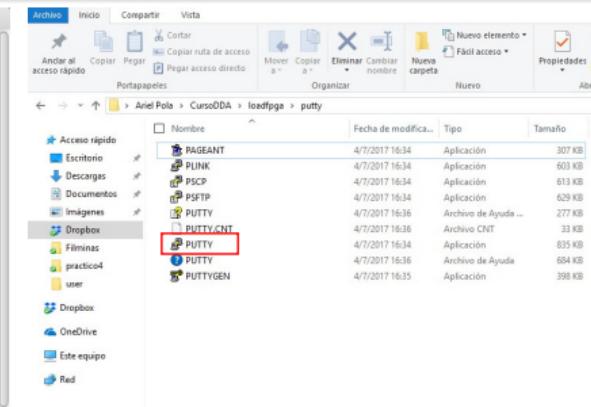
Tunel y Asignación de FPGA

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.

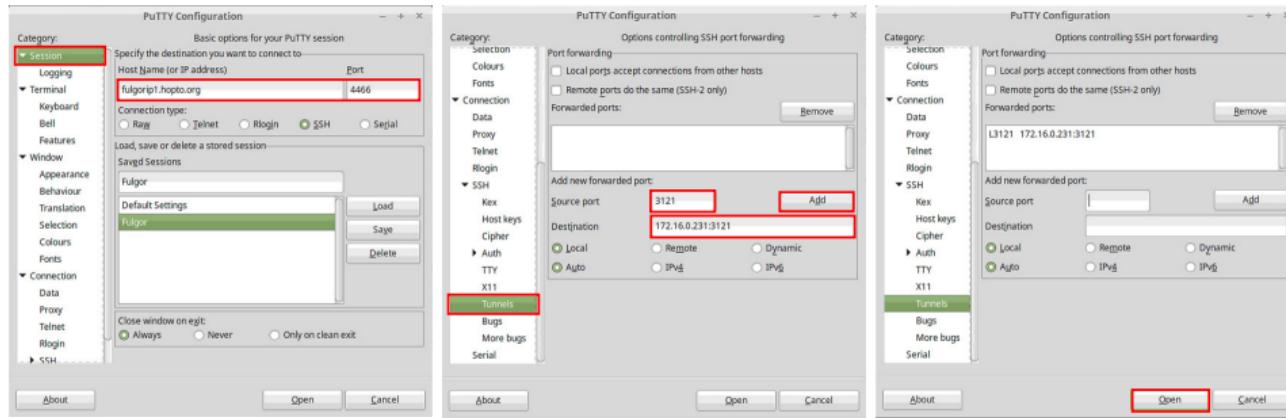


Linux



Windows

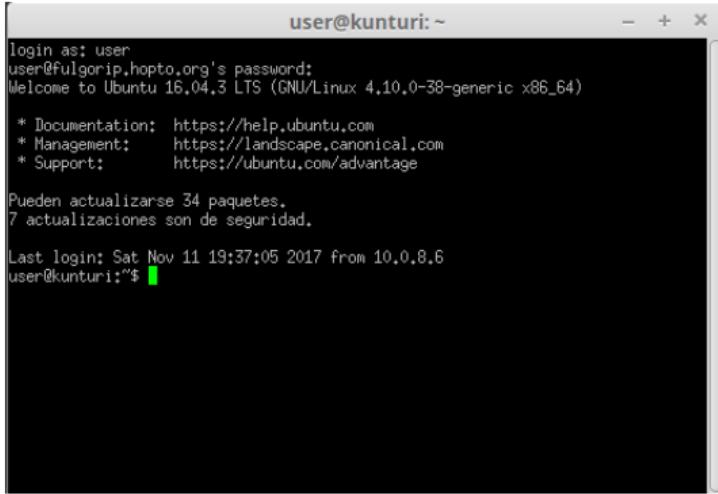
Tunel y Asignación de FPGA



- En la categoría **Session**, setear el host: *fulgorip.hopto.org* o *fulgorip1.hopto.org* y el puerto: **4466**.
- En la categoría **Connection** –> **SSH** –> **Tunnels** incluir el *forwarded* del puerto y agregarlo a la lista.
 - Source Port: **3121**
 - Destination: **172.16.0.231:3121**
- 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.

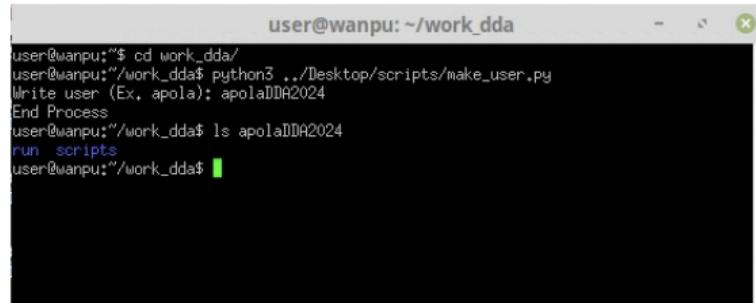
Last login: Sat Nov 11 19:37:05 2017 from 10.0.8.6
user@kunturi:~$
```

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 `python3 ../Desktop/scripts/make_user.py`

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



```
user@wanpu:~/work_dda
user@wanpu:~$ cd work_dda/
user@wanpu:~/work_dda$ python3 ../Desktop/scripts/make_user.py
Write user (Ex. apola): apolaDDA2024
End Process
user@wanpu:~/work_dda$ ls apolaDDA2024
run scripts
user@wanpu:~/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.

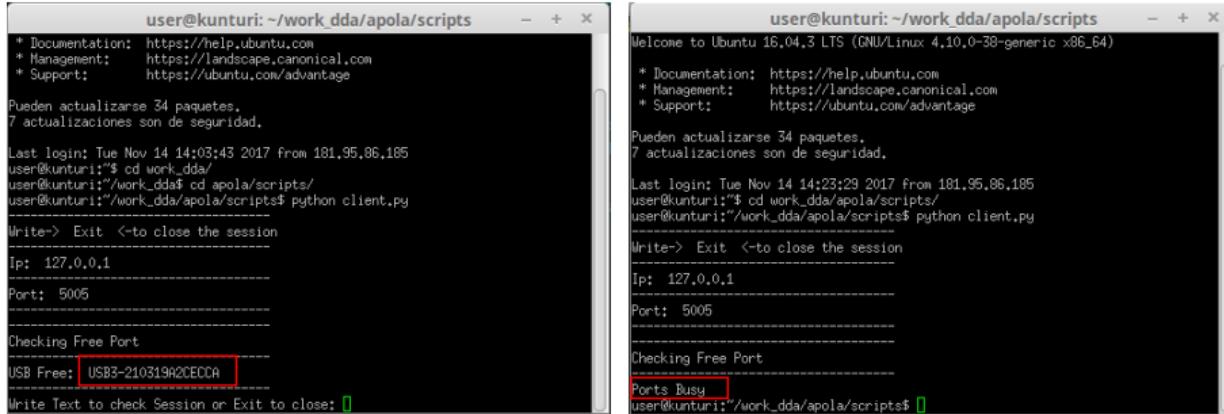
1 *cd apola/scripts*
2 *python client.py*

- 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.

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: USB3-210319A2CE0A
-----
Write Text to check Session or Exit to close: 0

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

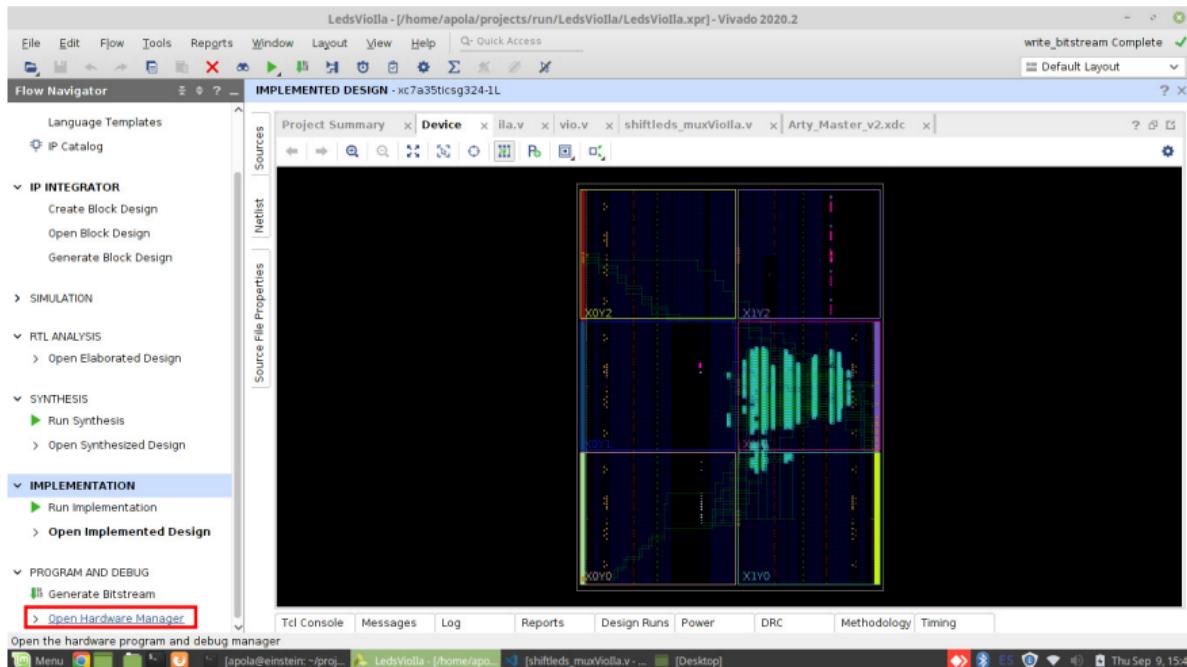
Write-> Exit <-to close the session
-----
Ip: 127.0.0.1
-----
Port: 5005
-----
Checking Free Port
-----
Ports Busy
user@kunturi:~/work_dda/apola/scripts$ 0
```

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

Programación y Ejecución

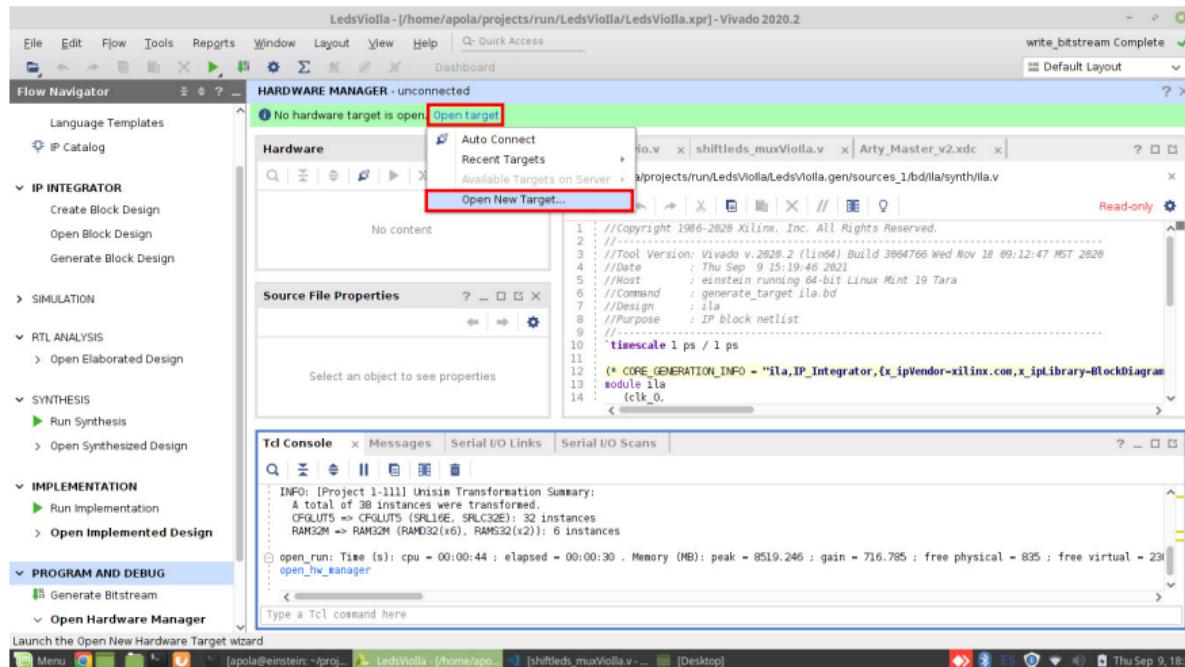


Programación y Ejecución



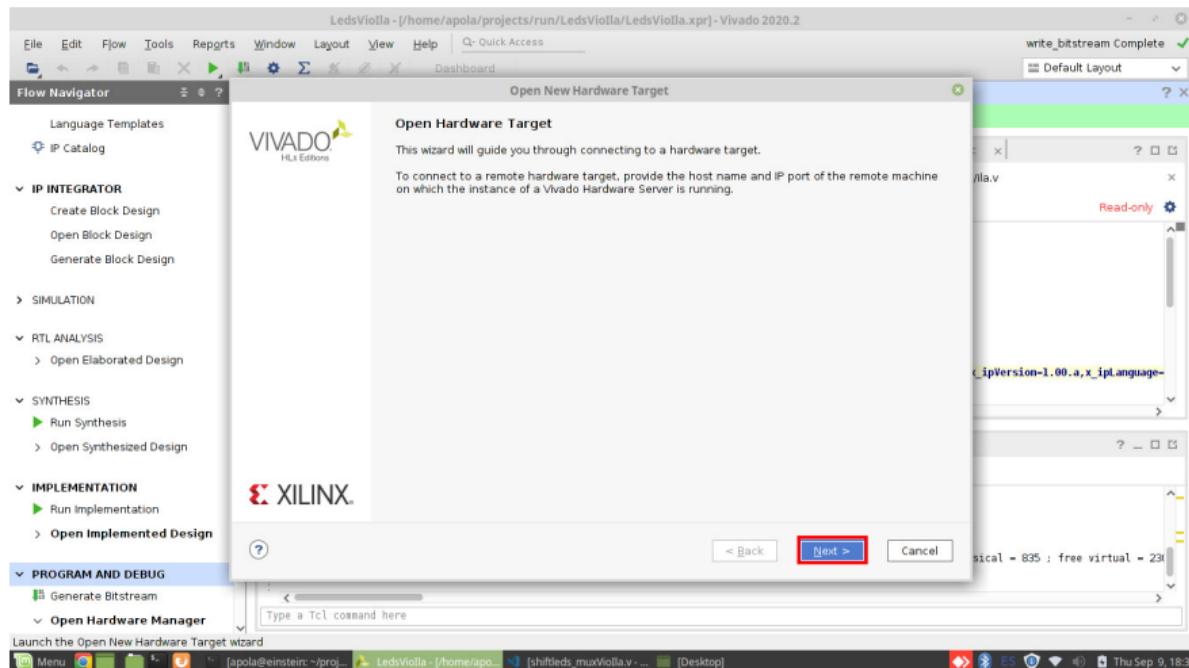
Open Hardware Manager.

Programación y Ejecución



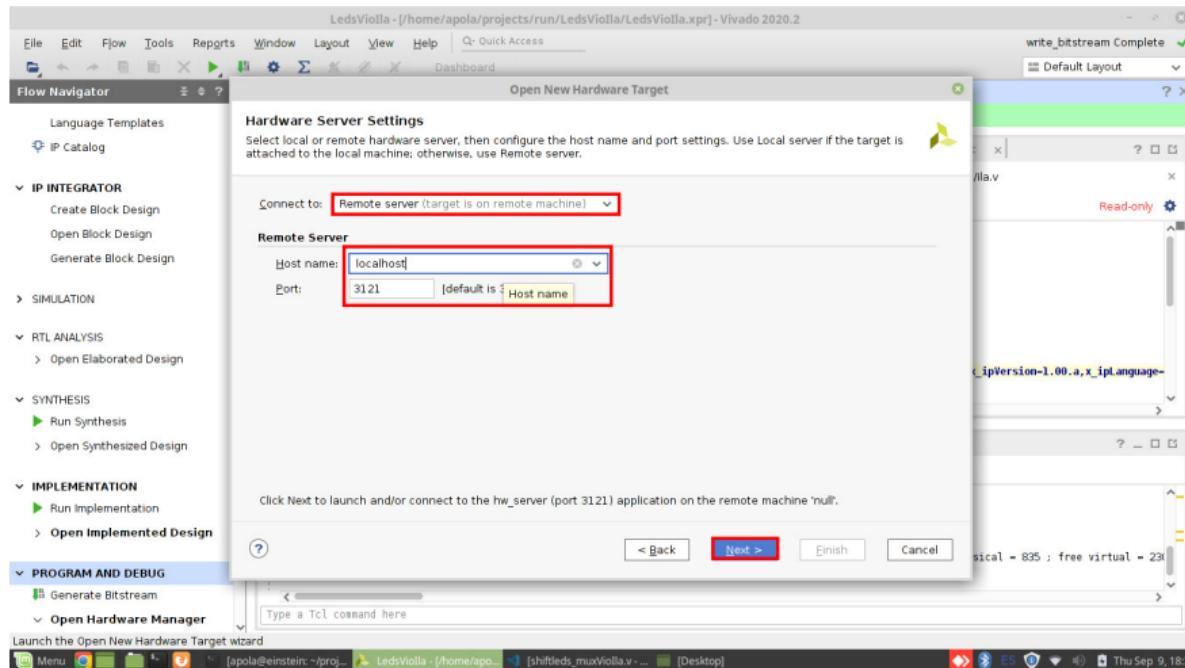
Abrir el dispositivo.

Programación y Ejecución



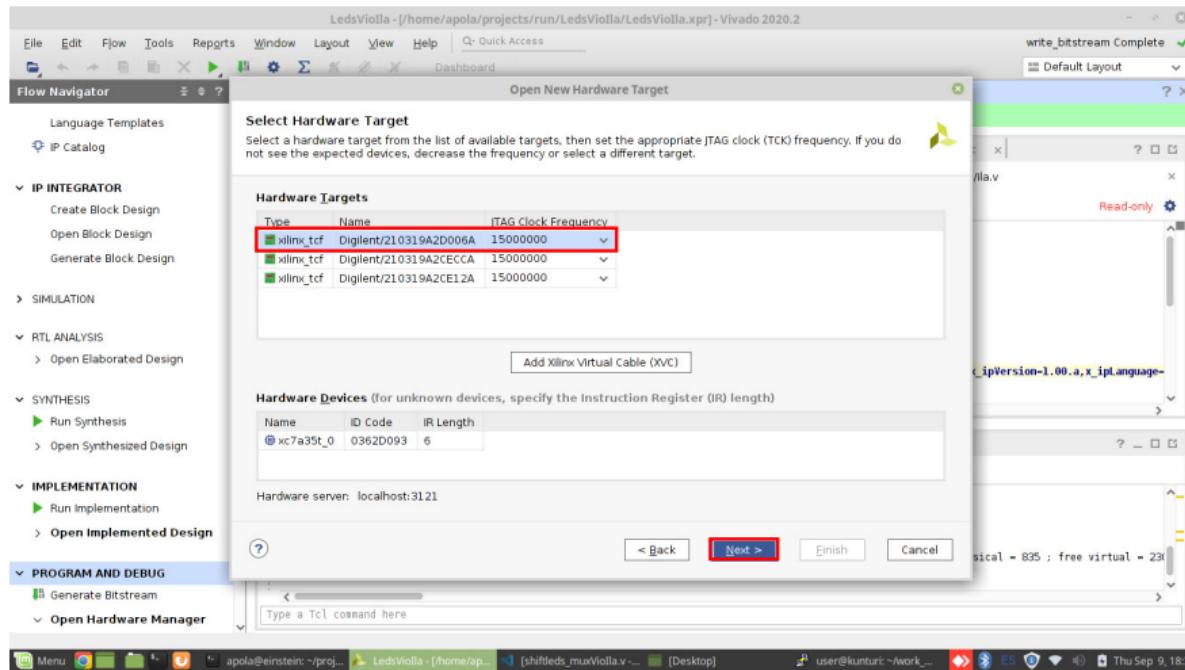
Abrir el dispositivo.

Programación y Ejecución



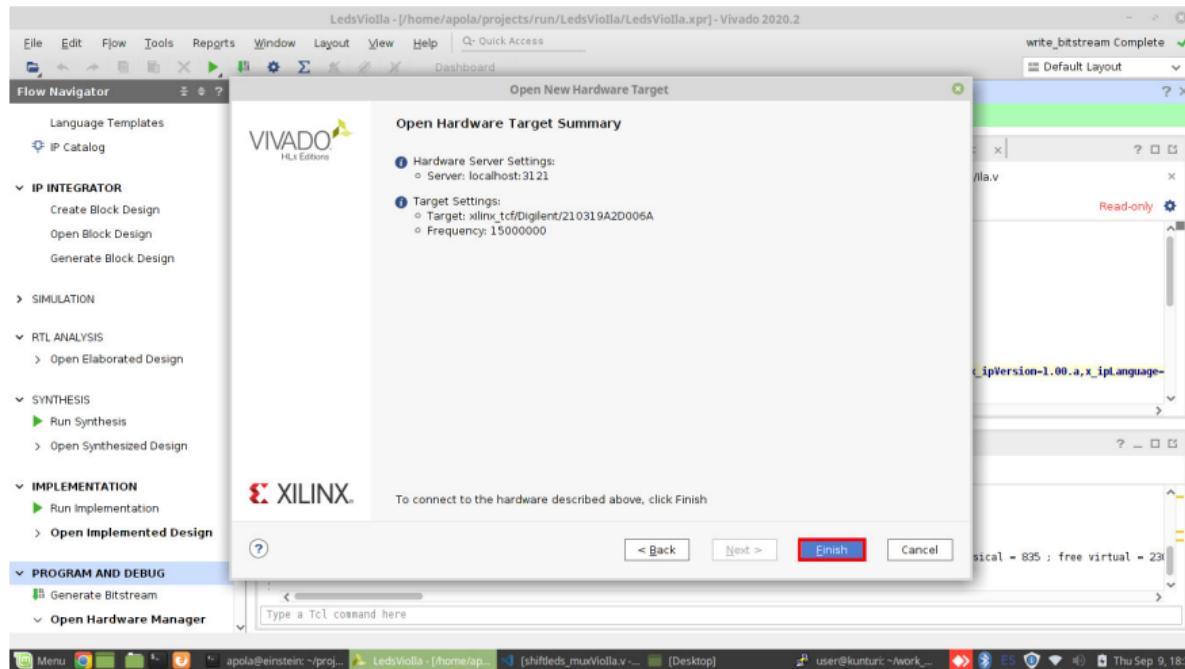
Seleccionar el servidor remoto y asignar el nombre del Host (localhost).

Programación y Ejecución



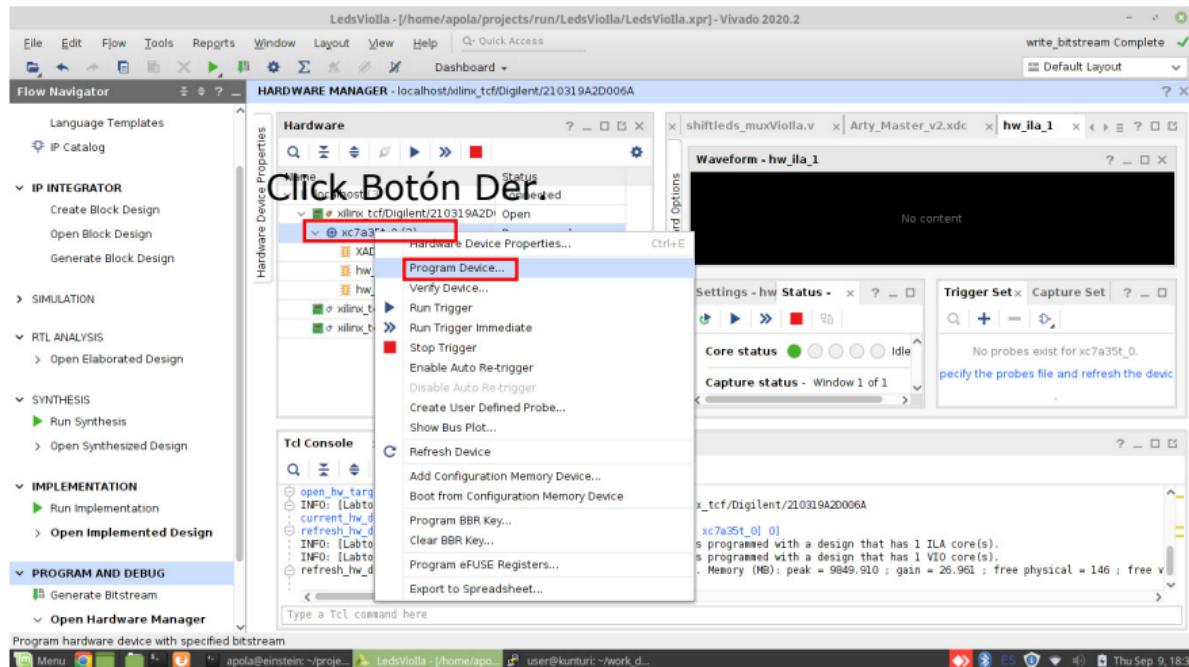
Seleccionar el ID previamente asignado por la conexión remota.

Programación y Ejecución



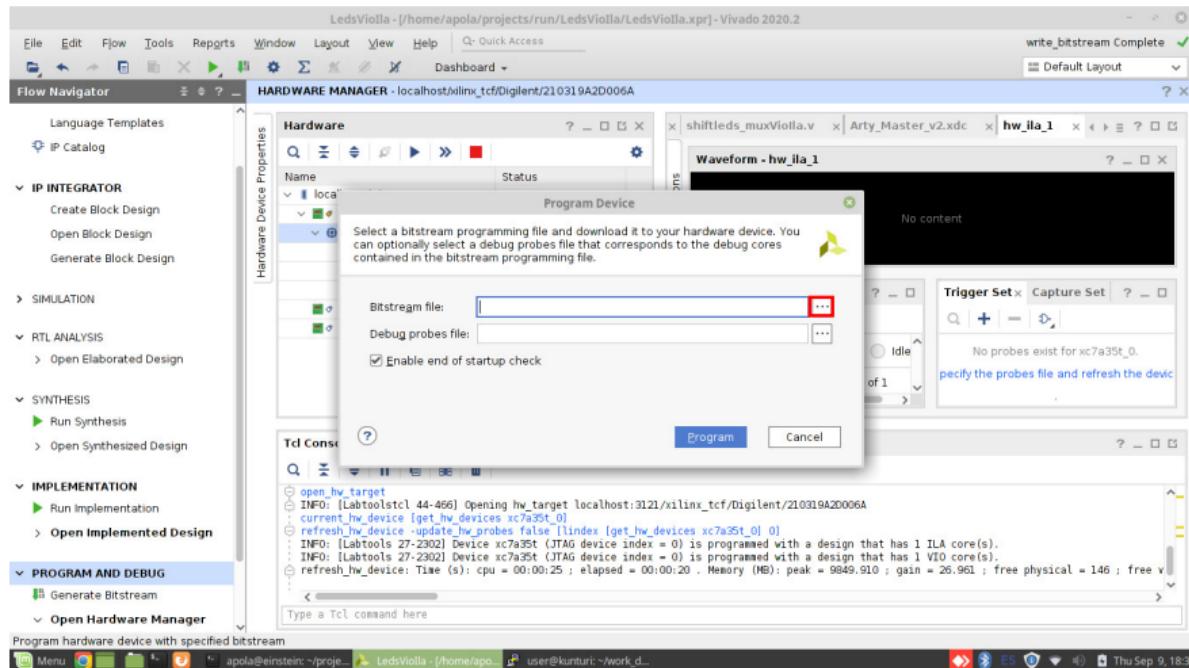
Finalizar configuración.

Programación y Ejecución



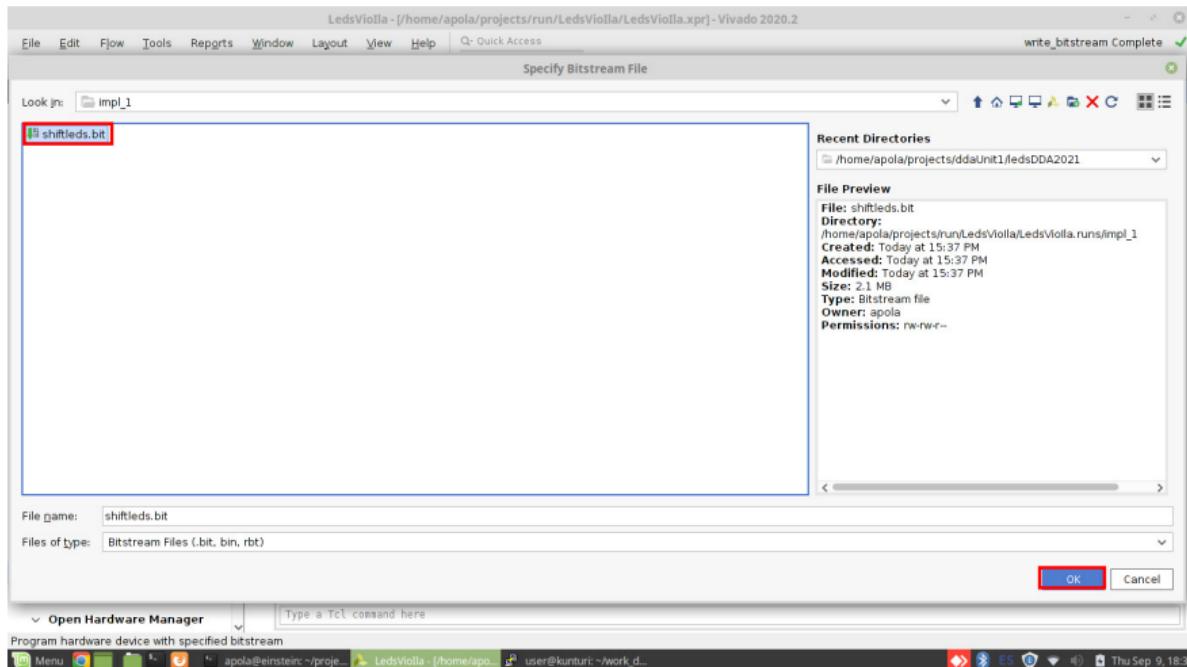
Programar la FPGA.

Programación y Ejecución



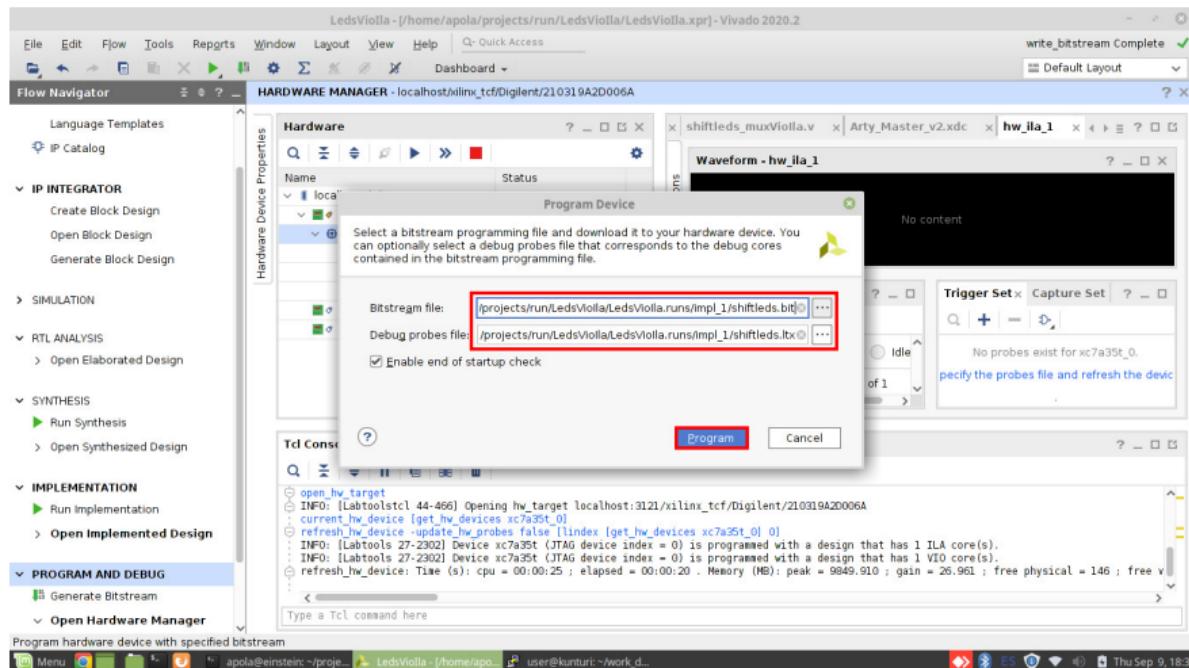
Buscar el archivo binario.

Programación y Ejecución



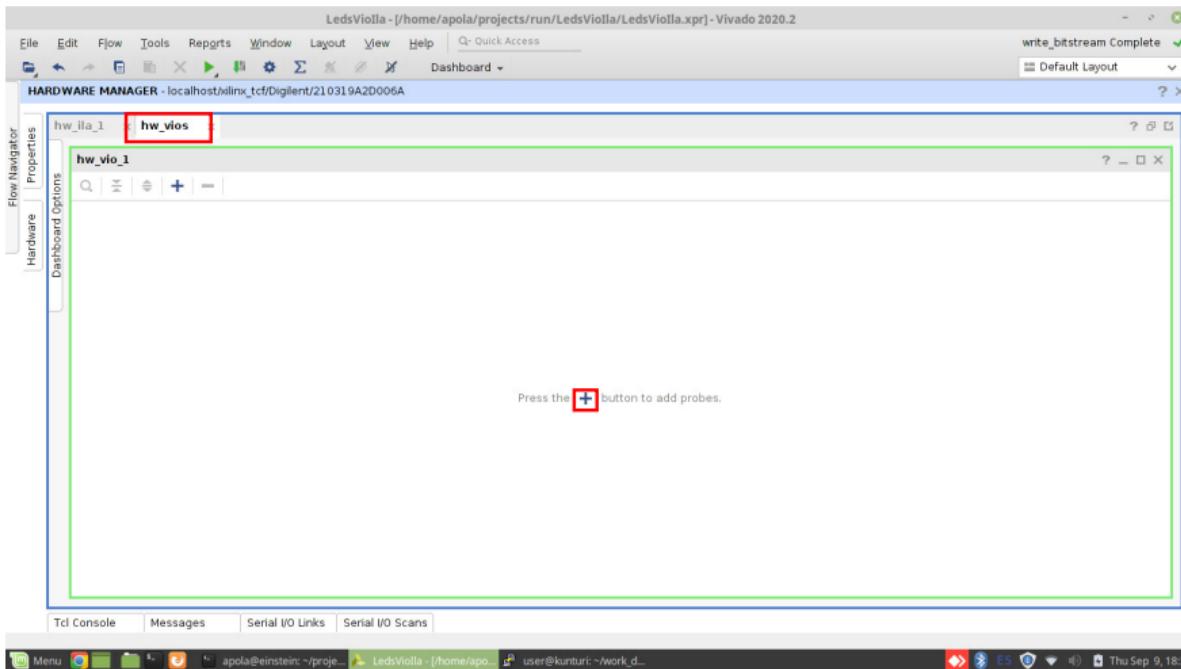
Buscar el archivo binario.

Programación y Ejecución



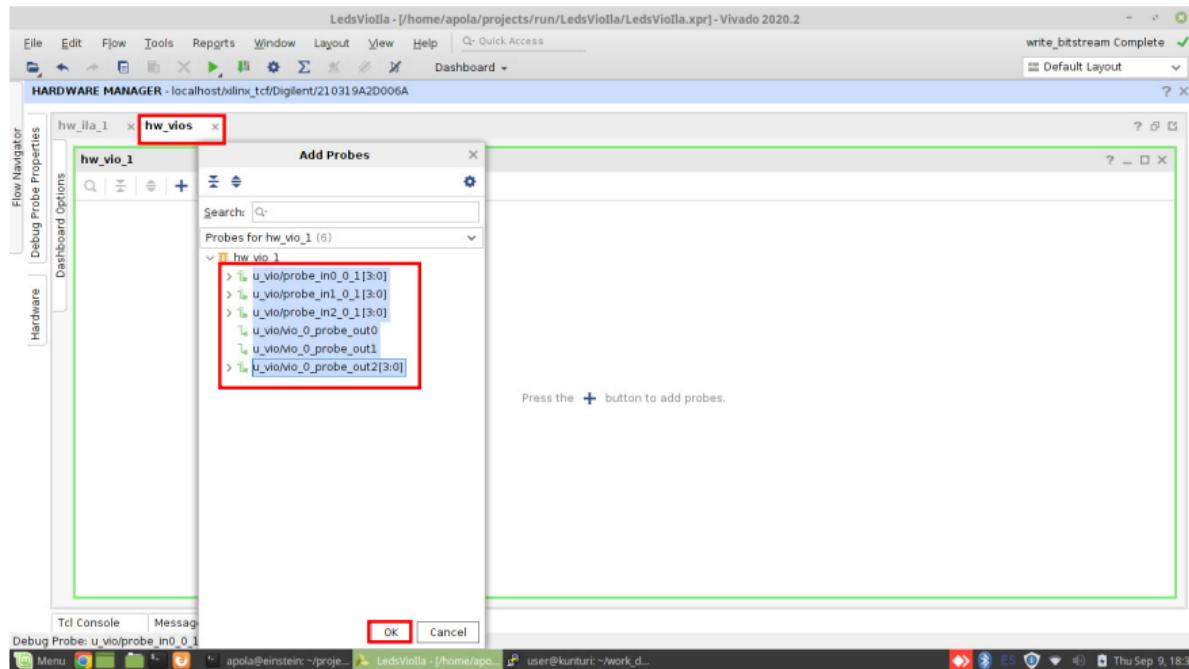
Verificar que se asigne el binario y el archivo de puntas de prueba.

Programación y Ejecución



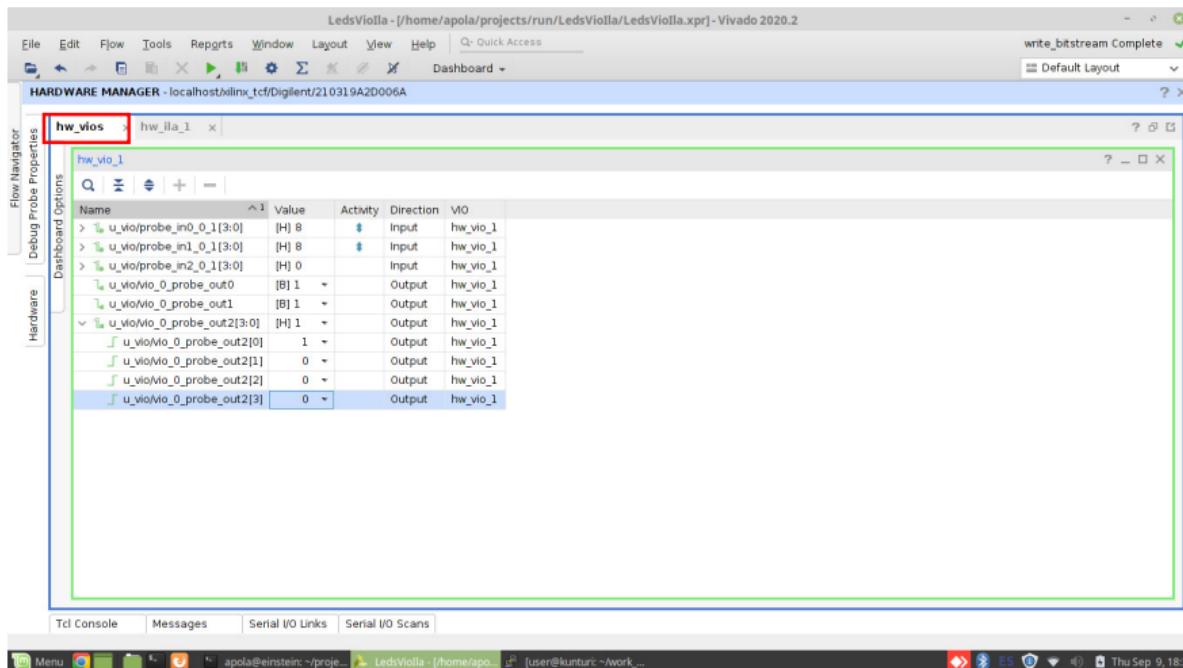
Seleccionar los puertos de control.

Programación y Ejecución



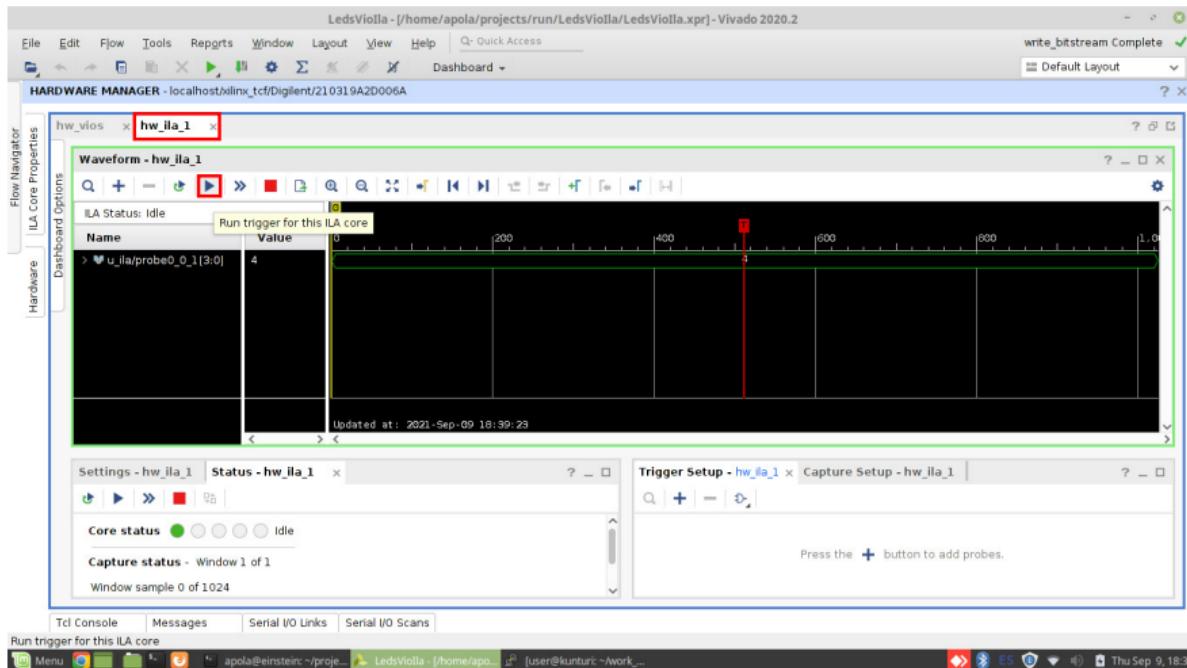
Seleccionar los puertos de control.

Programación y Ejecución



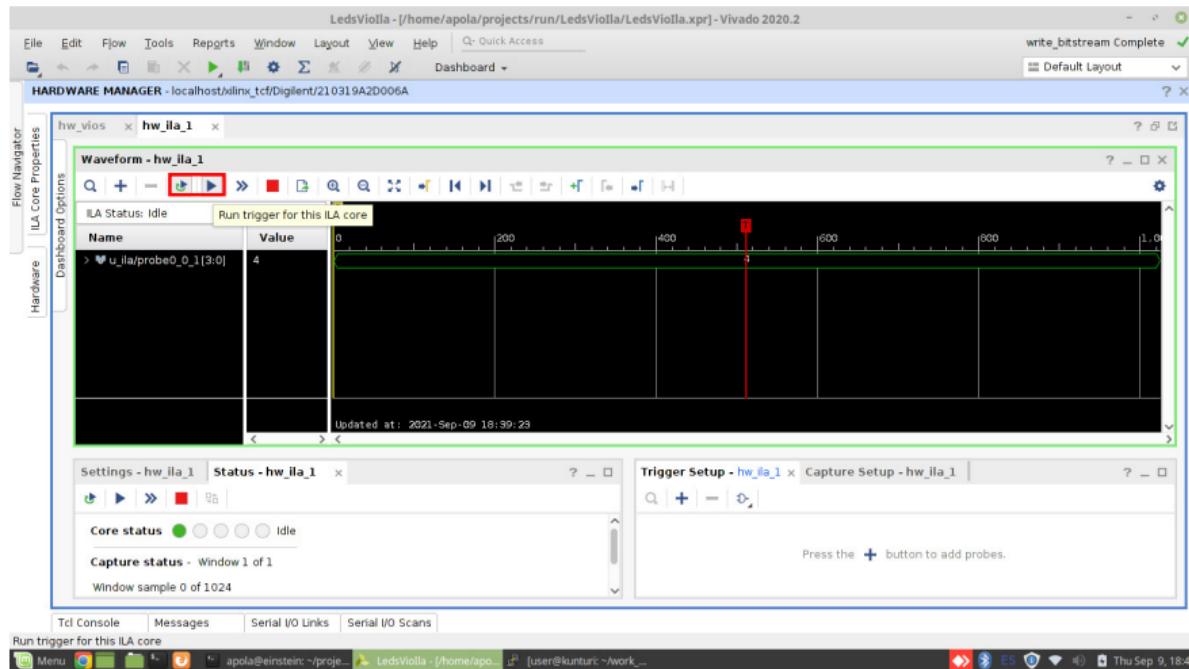
Puertos de control.

Programación y Ejecución



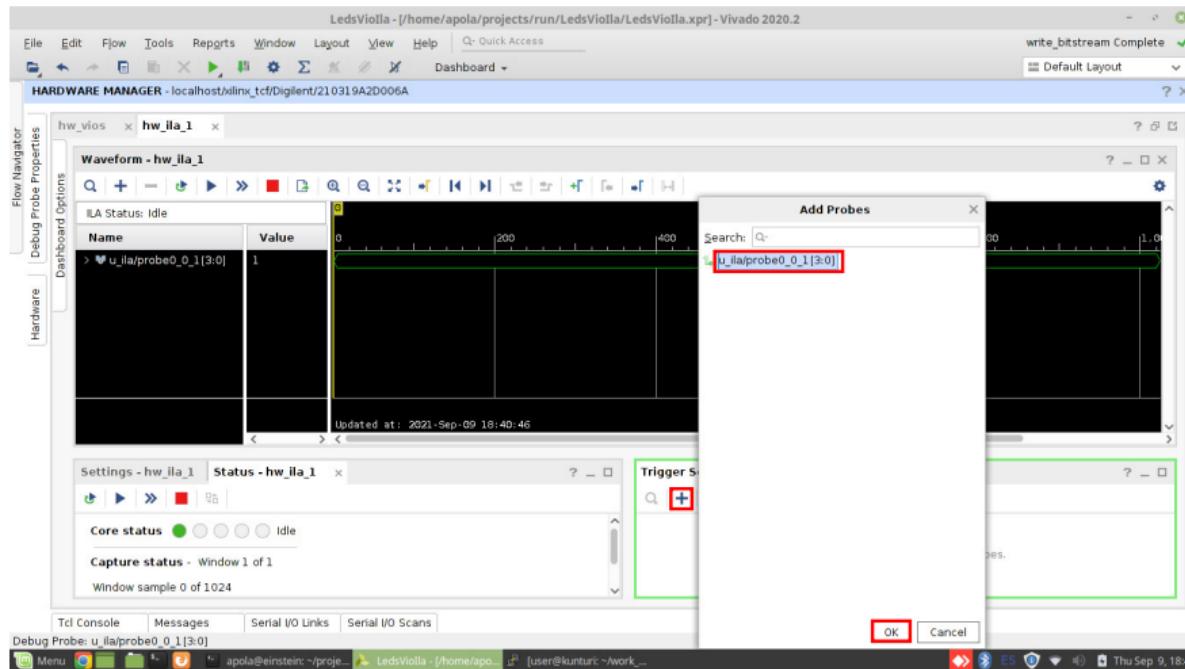
Ejecutar una vez el trigger.

Programación y Ejecución



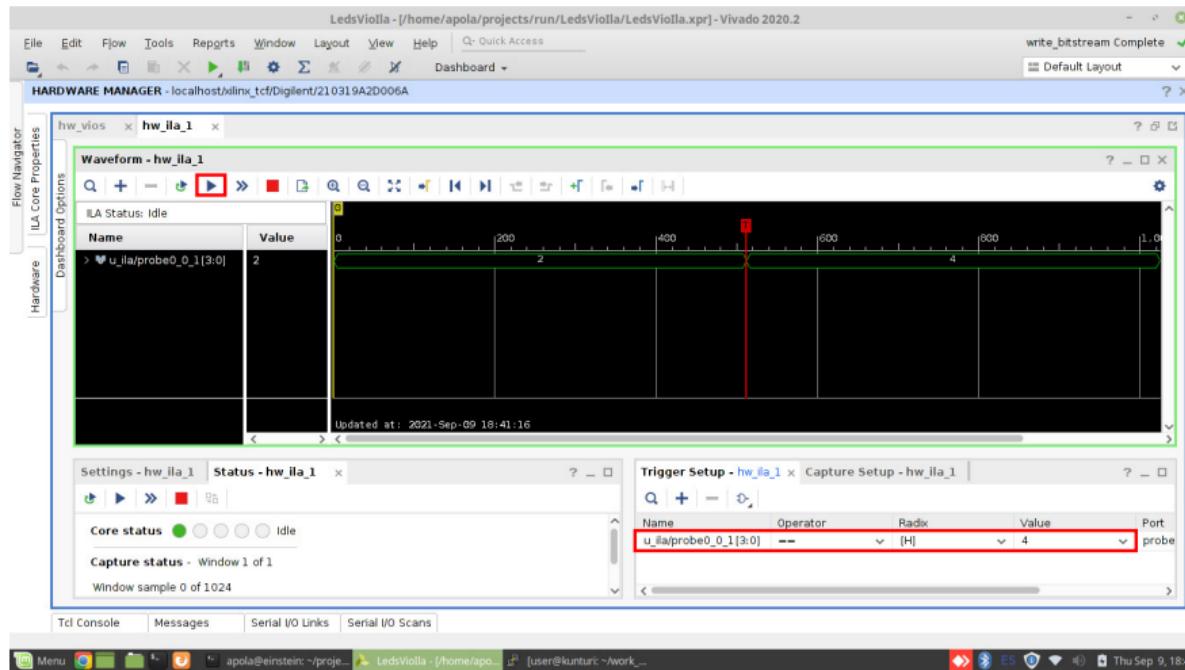
Seleccionar trigger continuo y luego ejecutar el trigger.

Programación y Ejecución



Agregar una condición de trigger y seleccionar el puerto.

Programación y Ejecución



Agregar una condición de comparación y ejecutar el trigger.