

# Adición de un periférico a un SoC Vivado /Vitis

Hortensia Mecha  
López

+

•

o

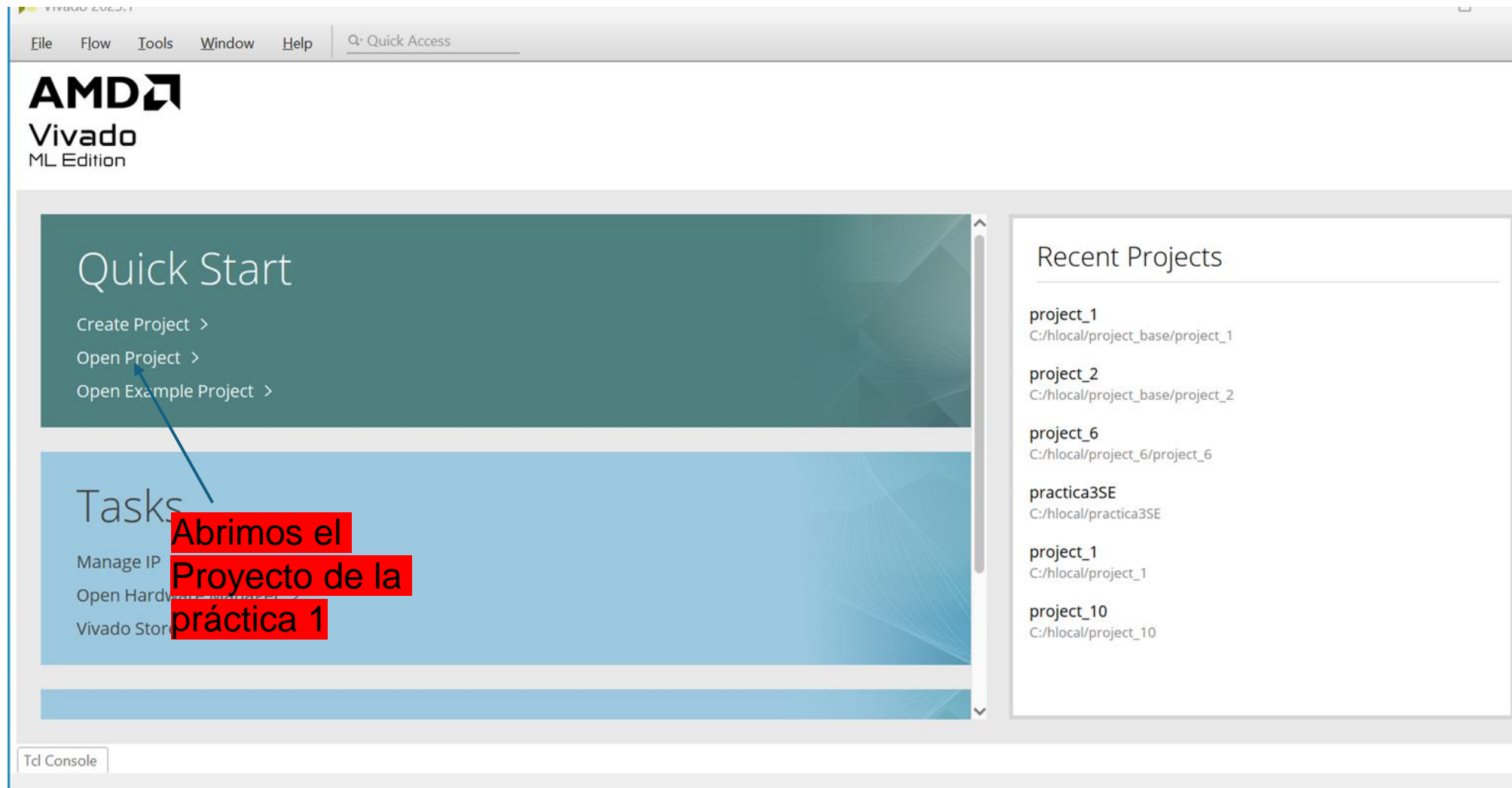
- En esta segunda práctica se trata de:

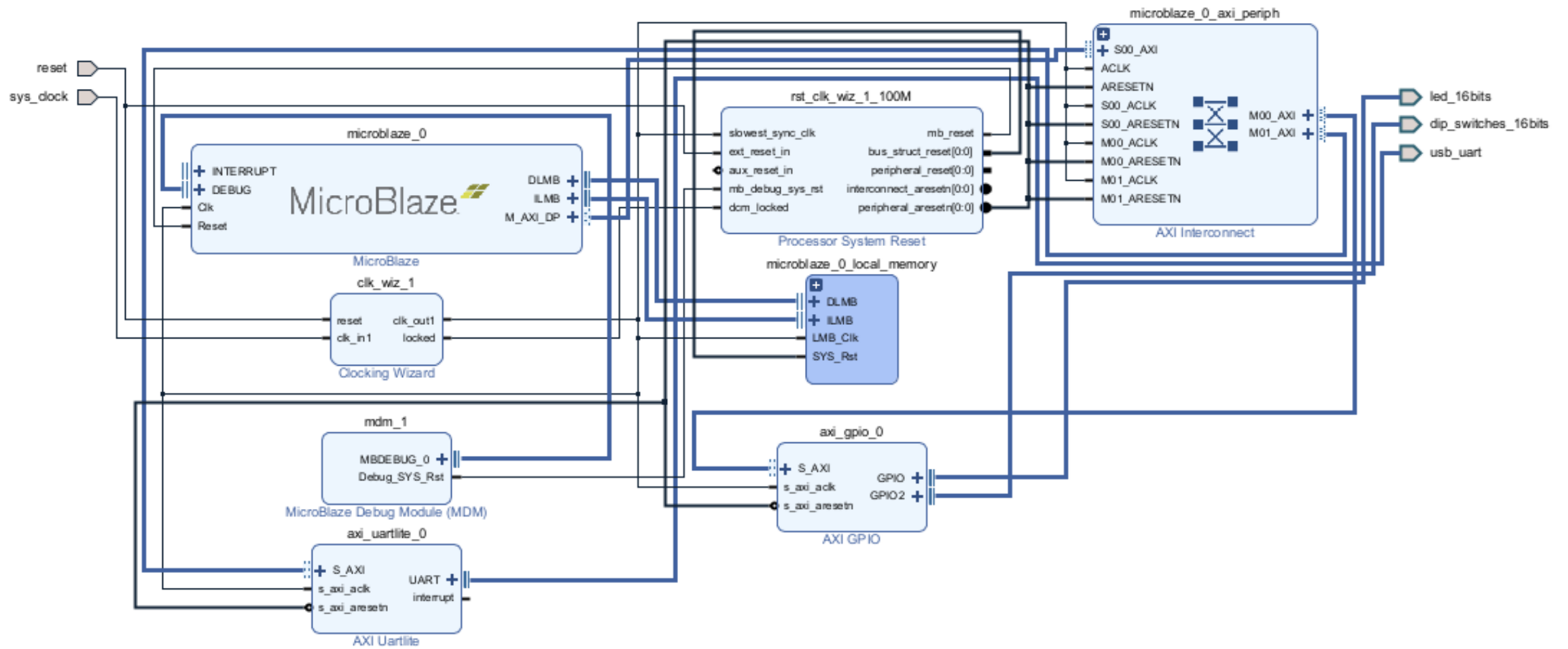
- Aprender a añadir un periférico utilizando Vivado
- Aprender a comunicarse con el periférico. Usaremos Vitis y c.

# Objetivos

# Punto de partida: el Proyecto básico

- Abrimos Vivado (versión 2023.1)
- Abrimos el Proyecto de la práctica 1
- En el menú superior seleccionamos Tools->Create and Package New IP
- Pulsamos Next
- Seleccionamos Create a new AXI peripheral
- Next
- Le damos el nombre que queramos y dónde lo queremos guardar
- Next
- Seleccionamos el número de registros
- Pulsamos Next y Finish







File Edit Flow Tools Reports Window Layout View Help Q Quick Access

Project Navigator

PROJECT MANAGER

- Settings
- Add Sources
- Language Template
- IP Catalog

INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Run Linter
- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

Tools

- Validate Design F6
- Create and Package New IP...
- Create Interface Definition...
- Enable Dynamic Function eXchange...
- Run Tcl Script...
- Property Editor Ctrl+J
- Associate ELF Files...
- Generate Memory Configuration File...
- Compile Simulation Libraries...
- Vivado Store...
- Custom Commands
- Launch Vitis IDE
- Language Templates
- Settings...

Diagram x Address Editor x

reset sys\_clock

microblaze\_0

MicroBlaze

clk\_wiz\_1

reset clk\_in1 clk\_out1 locked

Clocking Wizard

mdm\_1

MBDEBUG\_0

Debug\_SYS\_Rst

MicroBlaze Debug Module (MDM)

axi\_uartlite\_0

S\_AXI s\_axi\_aclk s\_axi\_arstn

UART interrupt

AXI Uartlite

Properties

Select an object to see properties

Create and Package New IP

Create Peripheral, Package IP or Package a Block Design

Please select one of the following tasks.

Packaging Options

☐ Package your current project  
Use the project as the source for creating a new IP Definition.

☐ Package a block design from the current project  
Choose a block design as the source for creating a new IP Definition.  
Select a block design: design\_1

☐ Package a specified directory  
Choose a directory as the source for creating a new IP Definition.

Create AXI4 Peripheral

☒ Create a new AXI4 peripheral  
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

?

< Back

Next >


Finish

Cancel

Reports



```
lock -- xi
lock -- xi
lock -- xilinx.com:ip:axi_uartlite:2.0 - axi_uartlite_0
```

 Create and Package New IP

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

Name: copro


Version: 1.0

Display name: copro\_v1.0

Description: My new AXI IP

IP location: C:/hlocal/vivado/SE/base\_concopro/project\_1/..../ip\_repo

☐ Overwrite existing



< Back

Next >

Finish

Cancel



Create and Package New IP

Add Interfaces

Add AXI4 interfaces supported by your peripheral

Enable Interrupt Support

S00\_AXI

copro\_v1.0

+

-

Interfaces

S00\_AXI

Name

S00\_AXI

Interface Type

Lite

Interface Mode

Slave

Data Width (Bits)

32

Memory Size (Bytes)

64

Number of Registers

4

[4..512]

?

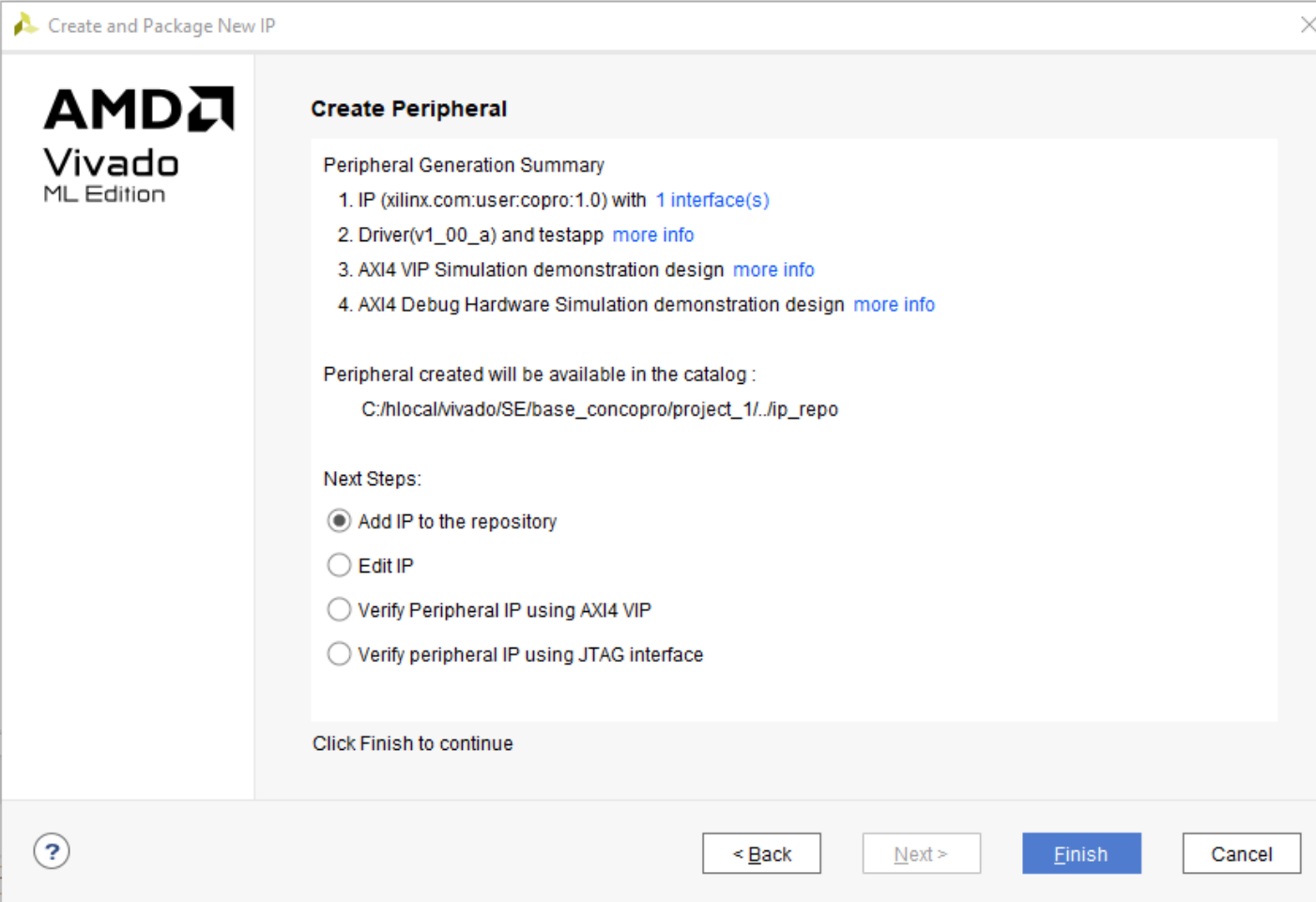
< Back

Next >

Finish

Cancel

```
xilinx.com:ip:axi_uartlite:2.0 - axi_uartlite_0
```

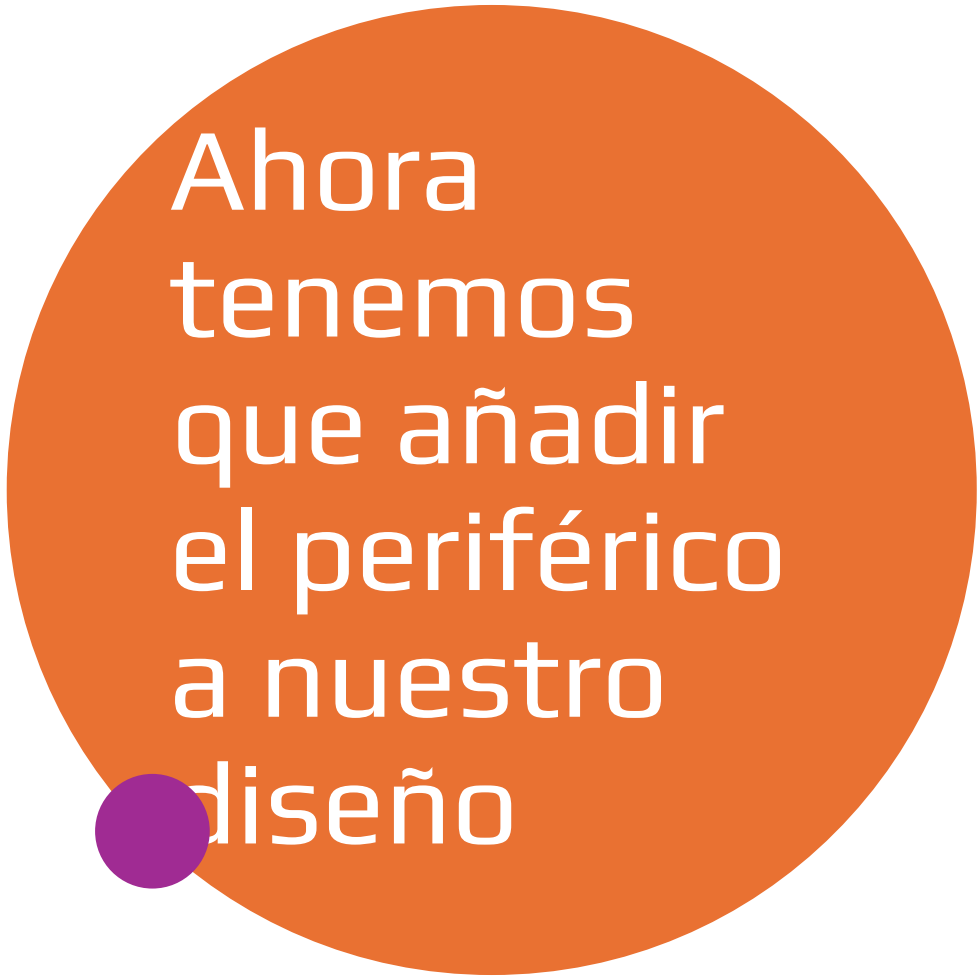


ports


xi

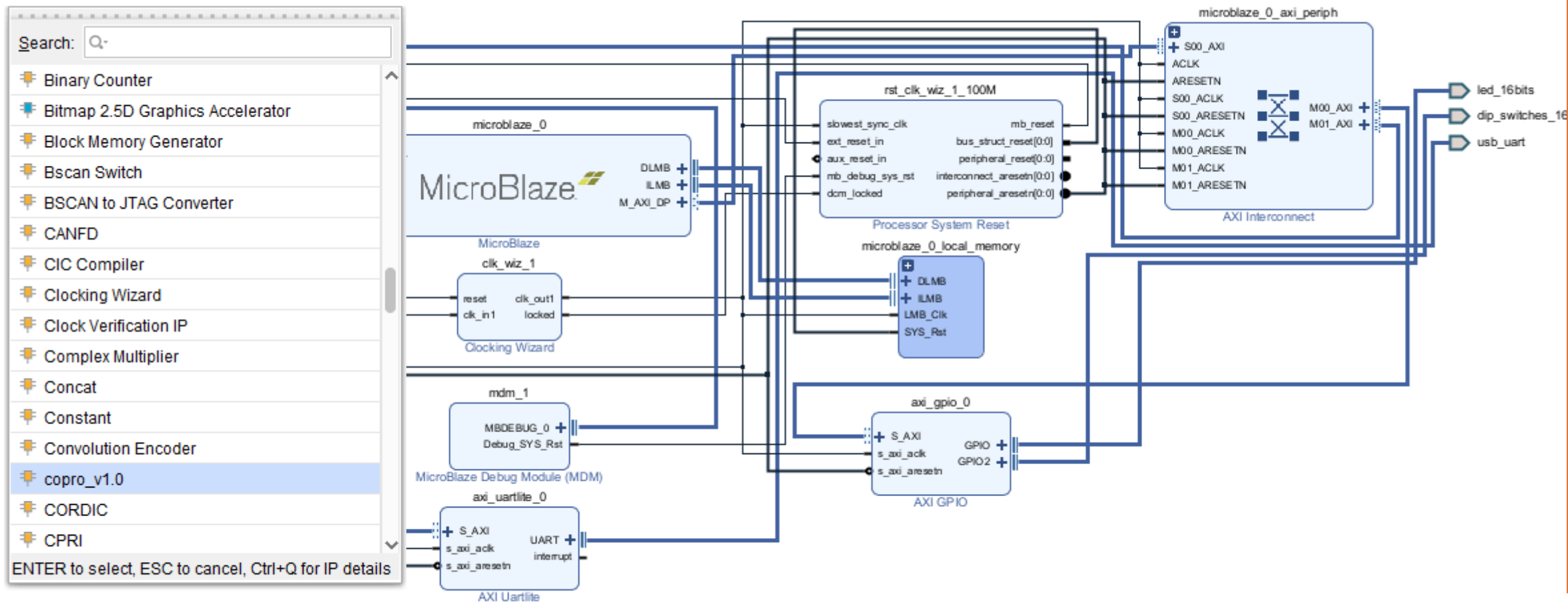
xi

xilinx.com:ip:axi\_uartlite:2.0 - axi\_uartlite\_0



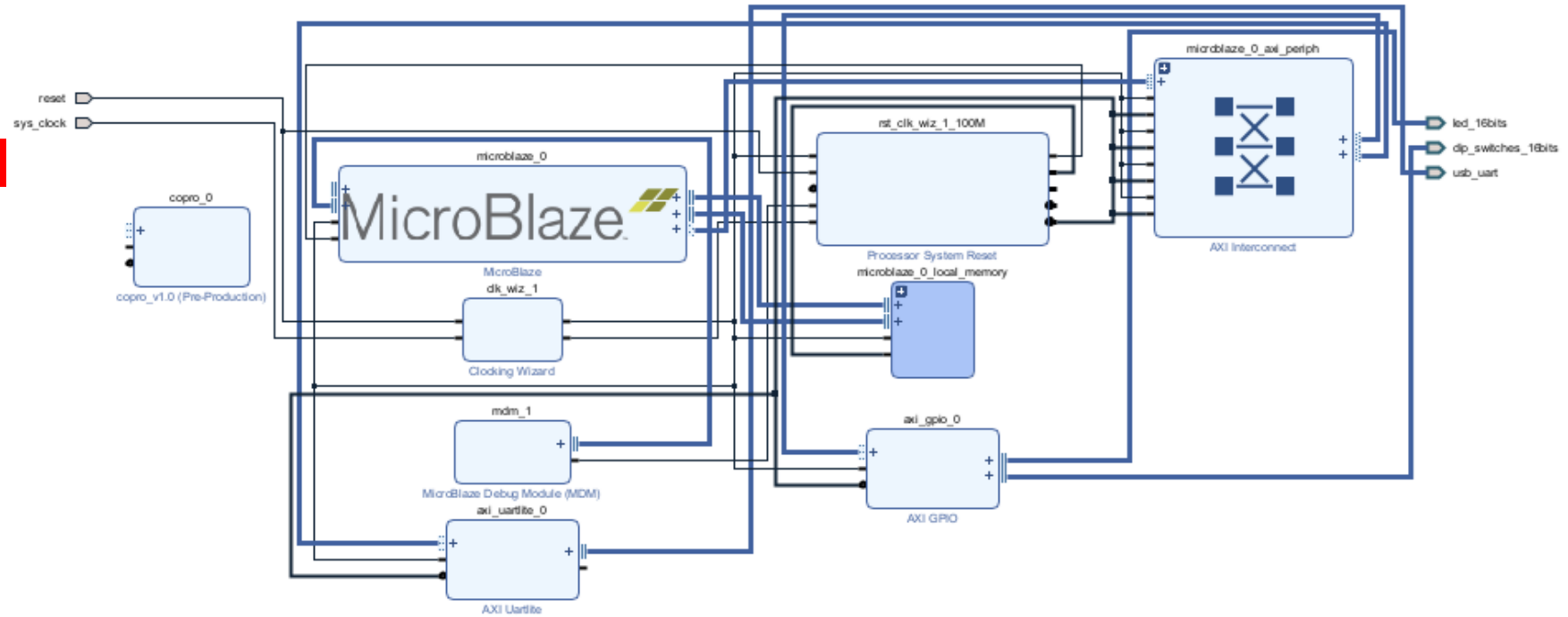
Ahora  
tenemos  
que añadir  
el periférico  
a nuestro  
diseño

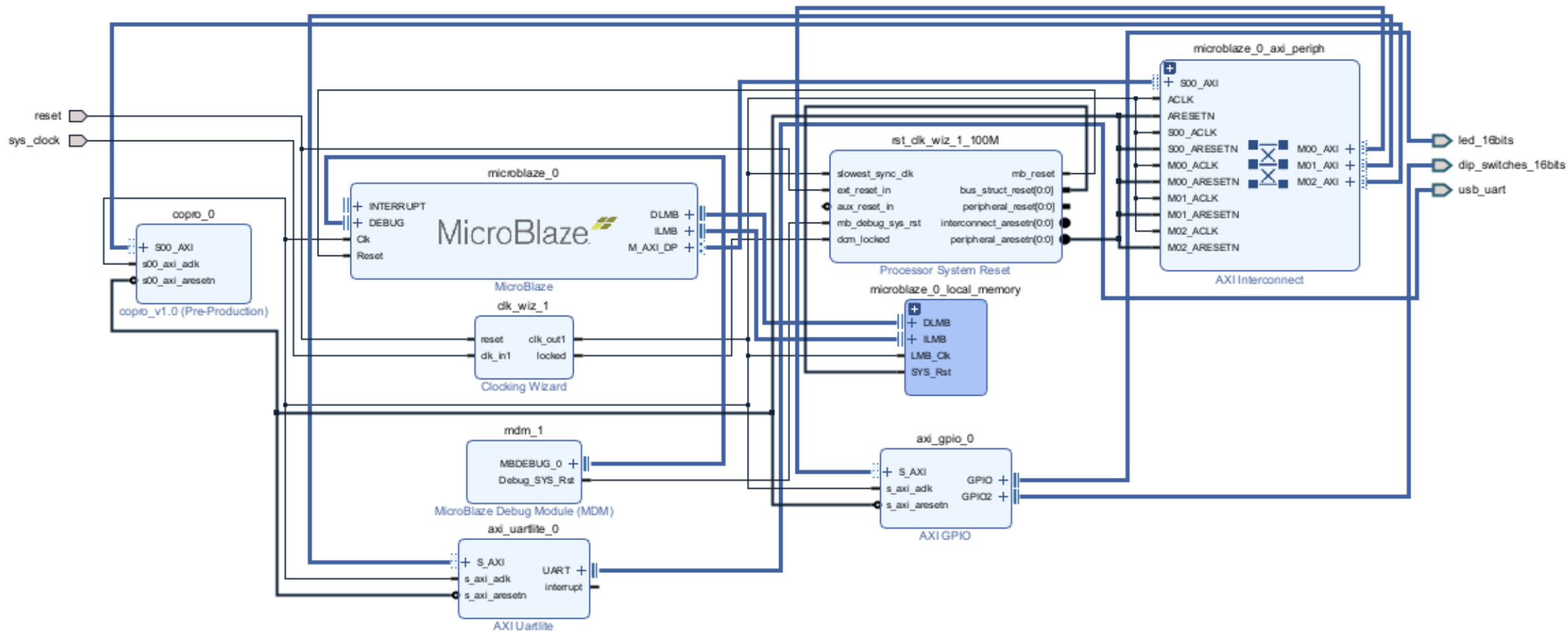
- 
- En la ventana del diagrama pulsamos +
  - En la lista de IPs que aparece buscamos el nuestro y lo seleccionamos y pinchamos 2 veces




✦ Designer Assistance available. [Run Connection Automation](#)

Pulsamos Run  
Connectiona  
Automation




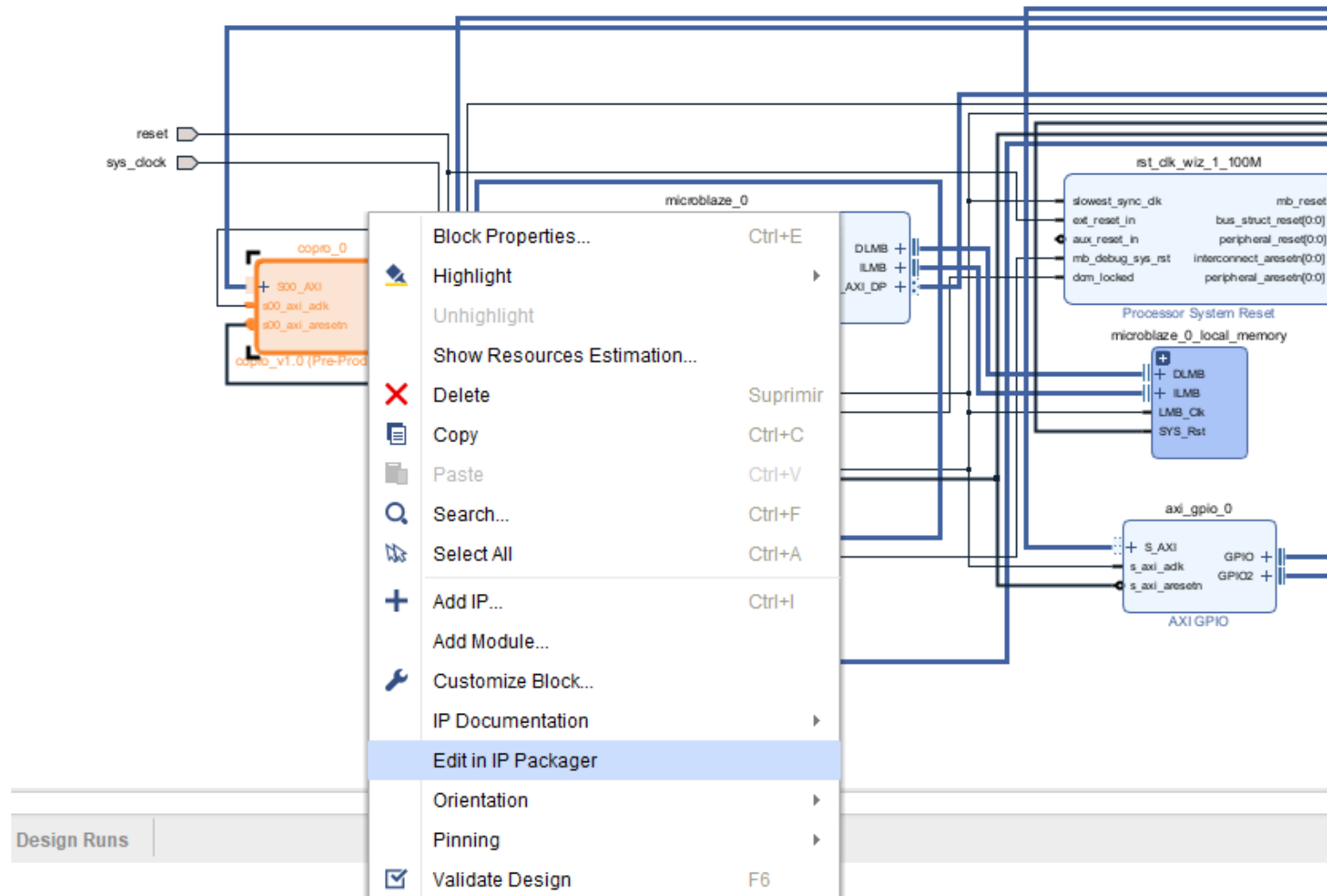






Ya tenemos  
el periférico  
añadido al  
proyecto

- 
- Ahora tenemos que editar el periférico y hacer las modificaciones que queramos
  - Pulsamos encima del periférico con el botón derecho y seleccionamos Edit in IP Packager y OK



## PROJECT MANAGER

## Settings

Add Sources

Language Templates

## IP Catalog

Edit Packaged IP

## IP INTEGRATOR

Create Block Design

Open Block Design

Generate Block Design

## SIMULATION

Run Simulation

## RTL ANALYSIS

Run Linter

Open Elaborated Design

## SYNTHESIS

Run Synthesis

Open Synthesized Design

## IMPLEMENTATION

Run Implementation

Open Implemented Design

## PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

## Sources

## Design Sources (2)

copro\_v1\_0(arch\_imp) (copro\_v1\_0.vhd) (1)

copro\_v1\_0\_S00\_AXI\_inst: copro\_v1\_0\_S00\_AXI(arch\_imp) (copro\_v1\_0\_S00\_AXI.vhd)

IP-XACT (1)

Constraints

Simulation Sources (1)

Utility Sources

Hierarchy Libraries Compile Order

## Source File Properties

copro\_v1\_0.vhd

Enabled

Location: c:/hlocal/vivado/se/base\_concoproip\_repo/copro\_v1\_0/hdl

Type: VHDL ...

Library: xil\_defaultlib ...

Size: 3.7 KB

Modified: Today at 12:29:25 PM

General Properties

Tcl Console

Messages


Log

Reports


Design Runs x

Q Z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	
▼ ▸ synth_1	constrs_1	Not started																				Vivado Synthesis Defaults (Vivado Synthesis 2023)
▸ impl_1	constrs_1	Not started																				Vivado Implementation Defaults (Vivado Implement



Se abre un  
nuevo  
Vivado con  
el proyecto  
del  
periférico

- 
- Ahora tenemos que editar el vhdl del periférico y hacer las modificaciones que queramos
  - Podemos añadir puertos de entrada salida y por ejemplo que el registro 3 sea la suma del uno y dos cuando reg0 valga 0 y la resta cuando valga 1

# En el top añadimos los botones

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity copro_v1_0 is
    generic (
        -- Users to add parameters here
        -- User parameters ends
        -- Do not modify the parameters beyond this line
        -- Parameters of Axi Slave Bus Interface S00_AXI
        C_S00_AXI_DATA_WIDTH          : integer    := 32;
        C_S00_AXI_ADDR_WIDTH          : integer    := 4);
    port (
        -- Users to add ports here
        buttons: in std_logic_vector(3 downto 0);
        -- User ports ends
        -- Do not modify the ports beyond this line
        -- Ports of Axi Slave Bus Interface S00_AXI
        s00_axi_aclk : in std_logic;
        s00_axi_aresetn      : in std_logic;
        -----
```

También  
actualizamos la  
component del  
copro

```
component copro_v1_0_S00_AXI is
    generic (
        C_S_AXI_DATA_WIDTH      : integer := 32;
        C_S_AXI_ADDR_WIDTH      : integer := 4
    );
    port (
        buttons: in std_logic_vector(3 downto 0);
        S_AXI_ACLK      : in std_logic;
        S_AXI_ARESETN   : in std_logic;
        .....
        S_AXI_RRESP      : out std_logic_vector(1 downto
        S_AXI_RVALID      : out std_logic;
        S_AXI_RREADY      : in std_logic
    );
end component copro_v1_0_S00_AXI;
```



# Y la instanciación del copro

```
copro_v1_0_S00_AXI_inst : copro_v1_0_S00_AXI
    generic map (
        C_S_AXI_DATA_WIDTH      =>
C_S00_AXI_DATA_WIDTH,
        C_S_AXI_ADDR_WIDTH      =>
C_S00_AXI_ADDR_WIDTH
    )
    port map (
        buttons => buttons,
        S_AXI_ACLK      => s00_axi_aclk,
        S_AXI_ARESETN   => s00_axi_aresetn,
        S_AXI_AWADDR     => s00_axi_awaddr,
        S_AXI_AWPROT     => s00_axi_awprot,
        S_AXI_AWVALID    => s00_axi_awvalid,
        S_AXI_AWREADY    => s00_axi_awready,
        .....
    );
```

# Modificamos el fichero copro

entity copro\_v1\_0\_S00\_AXI is

generic (

-- Users to add parameters here

-- User parameters ends

-- Do not modify the parameters beyond this line

-- Width of S\_AXI data bus

C\_S\_AXI\_DATA\_WIDTH : integer := 32;

-- Width of S\_AXI address bus

C\_S\_AXI\_ADDR\_WIDTH : integer := 4);

port (

-- Users to add ports here

buttons: in std\_logic\_vector(3 downto 0);

-- User ports ends

-- Do not modify the ports beyond this line

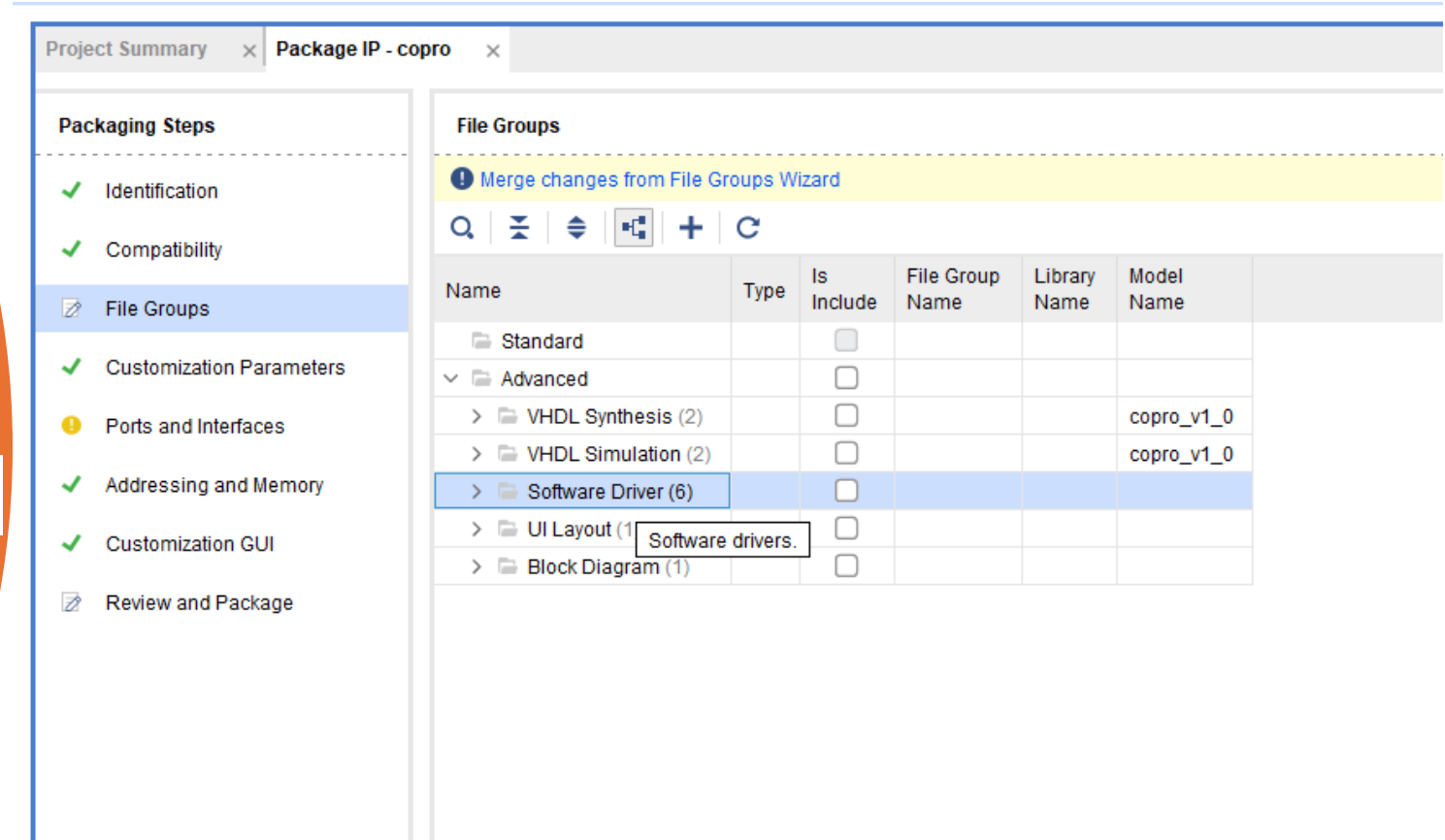
--

Modificamos el  
proceso de  
escritura de reg3  
Lo comentamos  
y añadimos lo  
que aparece en  
la viñeta de la  
derecha

```
when b"11" =>
-- for byte_index in 0 to (C_S_AXI_DATA_WIDTH/8-1) loop
--   if ( S_AXI_WSTRB(byte_index) = '1' ) then
--     slv_reg3(byte_index*8+7 downto byte_index*8) <= S_AXI_WDATA(byte_index*8+7 downto byte_index*8)
--   end if;
End loop;
When others => ...
End case;
End if;

if (buttons(0)) = '1' then
if (slv_reg0=x"00000000" ) then
    slv_reg3<= std_logic_vector(unsigned (slv_reg1) + unsigned (slv_reg2));
else
    slv_reg3<= std_logic_vector(unsigned (slv_reg1) - unsigned (slv_reg2));
end if;
else
slv_reg3 <= (others=> '1');
end if;
```

Antes de  
compilar  
modificamos el  
makefile



## File Groups

! Merge changes from File Groups Wizard



Name	Type	Is Include	File Group Name	Library Name	Model Name
Standard		<input type="checkbox"/>			
Advanced		<input type="checkbox"/>			
> VHDL Synthesis (2)		<input type="checkbox"/>			copro_v1_0
> VHDL Simulation (2)		<input type="checkbox"/>			copro_v1_0
Software Driver (6)		<input type="checkbox"/>			
drivers/copro_v1_0/data/copro.mdd	mdd d	<input type="checkbox"/>	xilinx_softwa		
drivers/copro_v1_0/data/copro.tcl	tclSou	<input type="checkbox"/>	xilinx_softwa		
drivers/copro_v1_0/src/Makefile	driver_	<input type="checkbox"/>	xilinx_softwa		
drivers/copro_v1_0/src/copro.h	cSour	<input type="checkbox"/>	xilinx_softwa		
drivers/copro_v1_0/src/copro.c	cSour	<input type="checkbox"/>	xilinx_softwa		
drivers/copro_v1_0/src/copro_selftest.c	cSour	<input type="checkbox"/>	xilinx_softwa		
> UI Layout (1)		<input type="checkbox"/>			
> Block Diagram (1)		<input type="checkbox"/>			

# Modificamos el makefile

Donde pone INCLUDEFILES=\*.h

Ponemos INCLUDEFILES=\$(wildcard \*.h)

Donde pone LIBSOURCES=\*.c

Ponemos LIBSOURCES=\$(wildcard \*.c)

Donde Pone OUTS = \*.o

Ponemos OUTS=\$(wildcard \*.o)

Y salvamos





A large orange circle on the left side of the slide, partially cut off by the edge.

En la pestaña  
File Groups  
pulamos  
Merge  
changes....

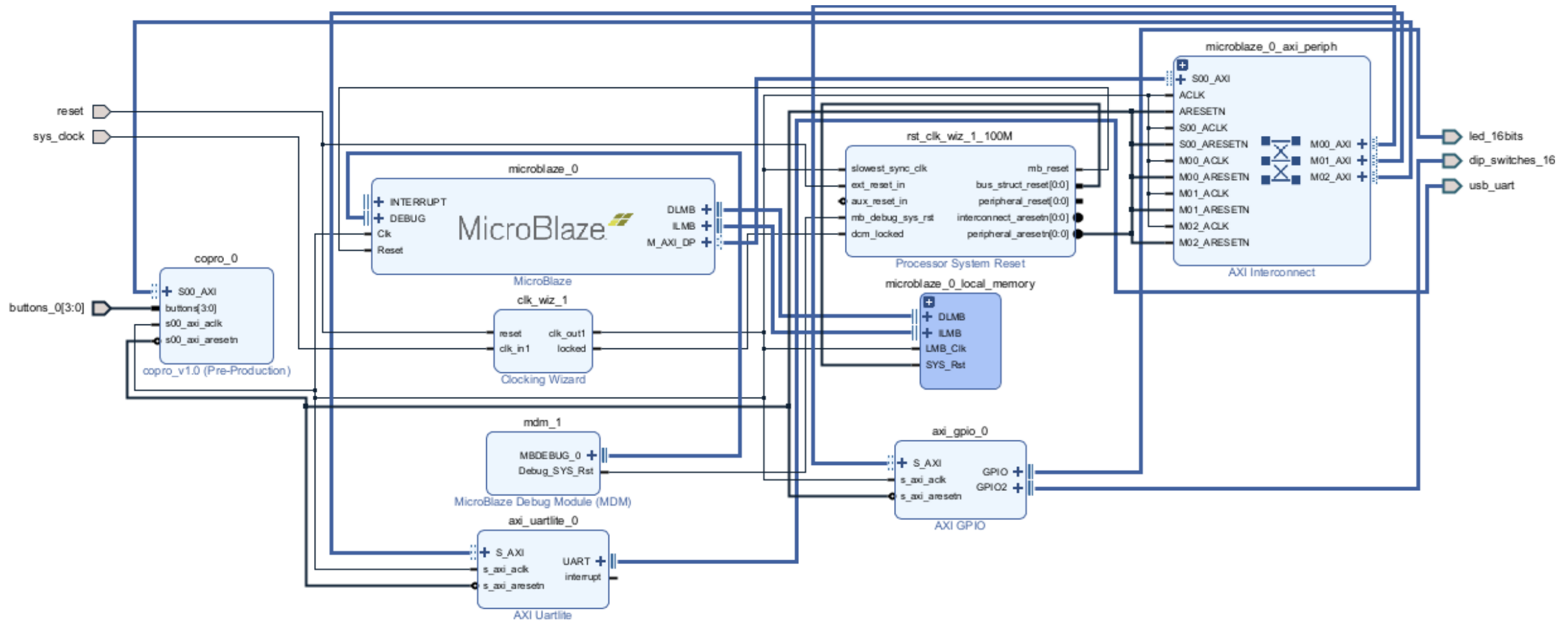
Actualizamos port and interfaces pulsando refresh para que aparezcan los nuevos ports y comprobamos que están correctos

Pulsamos Review and Package y Repage IP. Cerramos el Vivado del coprocesador

En el top le damos a Report IP Status y en la parte de Abajo Up-grade Selected  
Si no nos sale el nuevo puerto borrar la instancia y volverla a añadir.

Hacer los nuevos puertos añadidos externos, pulsando sobre ellos con el ratón, y con el botón derecho seleccionar make external.





# ¿Qué hemos añadido?

- Un periférico con 4 registros mapeados en memoria.
  - El registro 0 es de control
  - Los registros 1 y 2 son de entrada
  - El registro 3 es de salida

Falta añadir  
el fichero de  
restricciones  
solo para los  
puertos  
añadidos en  
el periférico

```
##buttons_0
```

```
# UP
```

```
    set_property PACKAGE_PIN T18 [get_ports buttons_0[1]]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports buttons_0[1]]
```

```
## left
```

```
    set_property PACKAGE_PIN W19 [get_ports buttons_0[2]]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports buttons_0[2]]
```

```
## right
```

```
    set_property PACKAGE_PIN T17 [get_ports buttons_0[3]]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports buttons_0[3]]
```

```
## down
```

```
    set_property PACKAGE_PIN U17 [get_ports buttons_0[0]]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports buttons_0[0]]
```

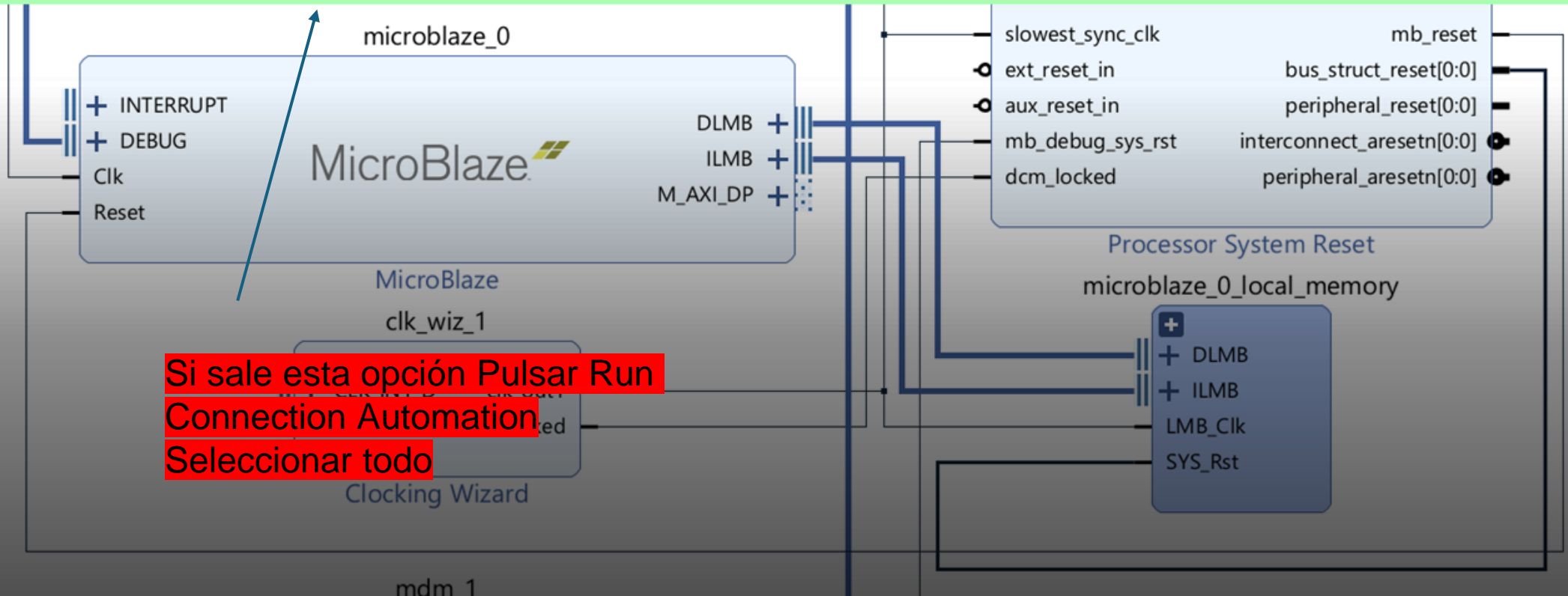
## BLOCK DESIGN - design\_1 \*

Diagram

Address Editor



Default View

Designer Assistance available. [Run Connection Automation](#)

Si sale esta opción Pulsar Run  
Connection Automation  
Seleccionar todo





- Pulsamos generate block
- Generamos el bitstream
- File ->Export ->Export hardware
- Next
- Include bitstream. Next
- Seleccionar un directorio para exportar el \*.xsa. Es importante quedarse con el nombre del directorio/fichero. Esa es nuestra plataforma hardware
- Podemos comprobar las restricciones en File->Export->Export Constraints (puede que haya que abrir antes el diseño implementado)
- Next y finish
- Cerramos Vivado



# Creamos el software

- Abrimos Vitis 2003.1
- Seleccionamos un directorio de workspace
- Creamos una Application Project
- Next
- Crear una nueva plataforma del hardware (XSA)
- Browse-> Seleccionar el fichero \*.xsa
- Next
- Dar nombre a la aplicación
- Next
- Next
- Seleccionar aplicación (Peripheral test o Hello world )
- Finish

Se ha creado un directorio workspace/design\_wrapper/hw con 3 ficheros

\*.bit

\*.mmi



\*.xsa (una copia del anterior)

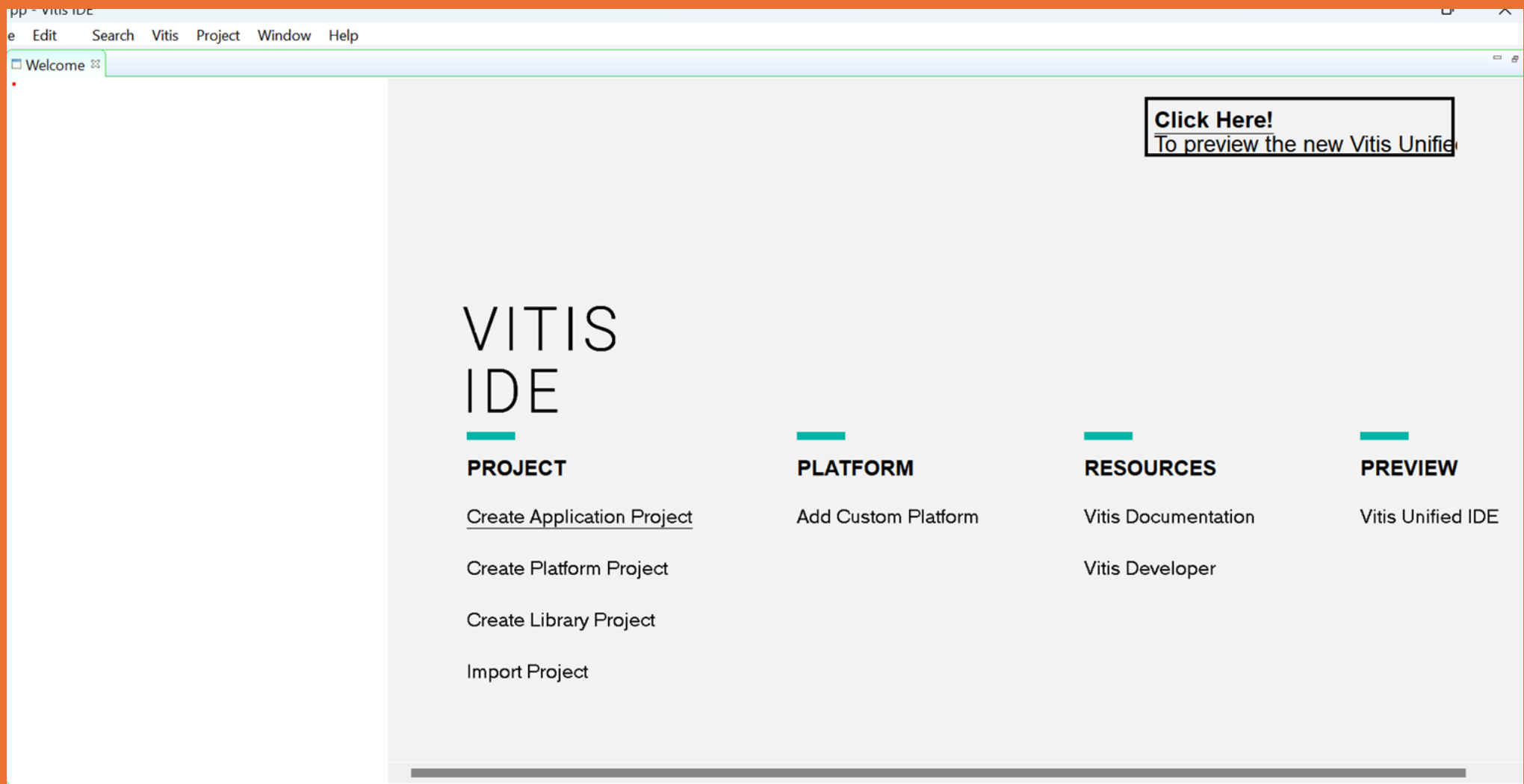



# Creamos el software

- Abrir el fichero \*.C y modificar lo que se desee
- Project build Project
- Vitis program device
- Seleccionar los ficheros \*.bit y \*.mmi creados para esta Plataforma (**muy importante por defecto puede tener seleccionados los de la práctica anterior**)
- Seleccionar fichero \*.elf (donde aparece bootloop)
- Program
- El software está cargado en la FPGA



- 
- Las siguiente diapositivas son para modificar el software.
  - Tenemos que poder leer y escribir en los registros de nuestro periférico
- 




- 
- Seleccionamos Create Application Project
  - Next
  - Seleccionamos Create a new platform from hardware (XSA)
  - Seleccionamos la plataforma que habíamos exportado
  - Y como template seleccionamos Peripheral test o Hello world

# Modificamos el software

- Escribimos en el registro de control (reg\_0) un 0 ó un 1 según queremos que sume o reste.
- Escribimos en los registros 1 y 2 los datos de entrada
- Mantenemos pulsado el botón inferior si queremos que el registro 3 se actualice
- Leemos el registro 3

- Al inicio ponemos los include

```
#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xgpio.h"
#include "gpio_header.h"
#include "copro.h" // o como se llame el
periférico
```



Modificamos  
el software  
Escribimos 0  
ó 1 en el  
registro 0

Los drivers de L/E en los registros se encuentran en el registro \*.h que se ha creado en  
workspace/design\_1\_wrapper\_1/hw/drivers/copro\_v1.0/src/  
Ese directorio debería haberse añadido automáticamente al diseño si el makefile está correcto. Hay que comprobar que están los ficheros de nuestro periférico

Para escribir en un registro del periférico que se llama COPRO se usa la función  
COPRO\_mWriteReg.

Por ejemplo para escribir un 0 en el registro reg0 usaríamos:

```
COPRO_mWriteReg(XPAR_COPRO_0_S00_AXI_BASEADDR,  
COPRO_S00_AXI_SLV_REG0_OFFSET, 0x00000000);
```

Modificamos  
el software  
Escribimos  
los datos que  
queramos en  
los registros  
1 y 2

```
/* Modificamos los registros 1 y 2 */
COPRO_mWriteReg(XPAR_COPRO_0_S00_AXI_BASEADDR,
COPRO_S00_AXI_SLV_REG1_OFFSET, 0x0005);
    COPRO_mWriteReg(XPAR_COPRO_0_S00_AXI_BASEADDR,
COPRO_S00_AXI_SLV_REG2_OFFSET, 0x0002);

/* Leemos el contenido de los 3 registros */
data_read= COPRO_mReadReg(XPAR_COPRO_0_S00_AXI_BASEADDR,
COPRO_S00_AXI_SLV_REG0_OFFSET);

    xil_printf("valor del registro 0 = %d\n\r", data_read);

    data_read= COPRO_mReadReg(XPAR_COPRO_0_S00_AXI_BASEADDR,
COPRO_S00_AXI_SLV_REG1_OFFSET);
    xil_printf("valor del registro 1 = %d\n\r", data_read);

data_read= COPRO_mReadReg(XPAR_COPRO_0_S00_AXI_BASEADDR,
COPRO_S00_AXI_SLV_REG2_OFFSET);
    xil_printf("valor del registro 2 = %d\n\r", data_read);
```



## Leemos el registro de salida

Leemos el registro 3. Debería ser la suma (o la resta, en función del registro 0) del registro 1 y 2 siempre que pulsemos el botón 0, que es el de abajo.

```
data_read=
COPRO_mReadReg(XPAR_COPRO_0_S00_AXI_BASEADDR,
COPRO_S00_AXI_SLV_REG3_OFFSET);
```

```
xil_printf("valor del registro 3 = %d\n\r",
data_read);
```

Comprobad el resultado y verificar distintas opciones dando valores diferentes al registro 0 y pulsando o no el botón inferior.

Es mejor que hagáis un bucle indefinido para que os de tiempo a pulsar el botón. Solo entonces leéis el registro 3.