

Cómo crear un Sistema Empotrado con Vivado /Vitis

Hortensia Mecha
López

Objetivos

- En esta primera práctica se trata de:
 - Aprender a crear un proyecto basado en microblaze y con algunos periféricos utilizando la herramienta Vivado
 - Escribir una aplicación sencilla en un lenguaje de programación de alto nivel, compilarla y ejecutarla en el microprocesador empotrado. Usaremos Vitis.

¿Cómo crear un sistema empotrado?

- Abrimos Vivado (versión 2023.1)
- Creamos el proyecto
 - RTL Project
 - Pulsamos Next hasta la ventana de selección de "board"
 - Seleccionamos la Basys 3
 - Next y Finish
- En el menú de la izquierda seleccionamos IP Integrator->Create Block Design
- En la Ventana Diagram empezamos a añadir componentes
 - MicroBlaze
 - Pulsamos Run Block Automation y seleccionamos 16KB de memoria Local
 - Añadimos Axi Interconnect
 - Añadimos Axi Uart Lite
 - Añadimos los switches y leds desde la pestaña board
 - Pulsamos Run Connection Automation y seleccionamos todo
- Ya tenemos nuestro sistema empotrado



Quick Start

- Create Project >
- Open Project >
- Open Example Project >

Tasks

- Manage IP >
- Open Hardware Manager >
- Vivado Store >

Recent Projects

- project_1
C:/hlocal/project_base/project_1
- project_2
C:/hlocal/project_base/project_2
- project_6
C:/hlocal/project_6/project_6
- practica3SE
C:/hlocal/practica3SE
- project_1
C:/hlocal/project_1
- project_10
C:/hlocal/project_10

Tcl Console

Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for the project. Then, you will specify your project sources and choose a default part.

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.




Project name:


Project location:

☒ Create project subdirectory

Project will be created at: C:/hlocal/project_base/project_3

 New Project ✕

Project Type
Specify the type of project to create.



☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

☐ Do not specify sources at this time

☐ Project is an extensible Vitis platform


☐ **Post-synthesis Project**
You will be able to add sources, view device resources, run design analysis, planning and implementation.

☐ Do not specify sources at this time

☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.



< Back

Next >

Finish

Cancel

New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

To fetch the latest available boards from git repository, click on 'Refresh' button. [Dismiss](#)

[Reset All Filters](#)

Vendor:

digilentinc.com

 Name:

All Remaining

 Board Rev:

Latest

Search:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board Rev	Available IOBs	LUT Elements	FlipFlops	Block RAMs	UI
Basy3		Installed	digilentinc.com	1.1	xc7a35tcpg236-1	236	C.0	106	20800	41600	50	0

Refresh

< Back

Next >

Finish

Cancel

project_3 - [C:/hlocal/project_base/project_3/project_3.xpr] - Vivado 2023.1

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

Default Layout

Flow Navigator

- PROJECT MANAGER
 - Settings
 - Add Sources
 - Language Templates
 - IP Catalog
- IP INTEGRATOR
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- SIMULATION
 - Run Simulation
- RTL ANALYSIS
 - Run Linter
 - Open Elaborated Design
- SYNTHESIS
 - Run Synthesis

PROJECT MANAGER - project_3

Sources

Design Sources

Constraints

Hierarchy Libraries Compile Order

Properties

Select an object to see properties

Project Summary

Overview Dashboard

Settings Edit

Project name: project_3

Project location: C:/hlocal/project_base/project_3

Product family: Virtex-7

Project part: xc7vx485tffg1157-1

Top module name: Not defined

Target language: Verilog

Tcl Console Messages Log Reports Design Runs

Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT
constrs_1	Not started												
constrs_1	Not started												

Creamos un
"block design"

Flow Navigator

PROJECT MANAGER - project_3

Sources

Design Sources

Hierarchy Libraries Compile Order

Properties

Create Block Design

Please specify name of block design.

Design name: design_1

Directory: <Local to Project>

Specify source set: Design Sources

OK Cancel

Project Summary

Overview Dashboard

Project name: project_3

Project location: C:/hlocal/project_base/project_3

Product family: Virtex-7

Project part: xc7vx485tffg1157-1

Module name: Not defined

Target language: Verilog

Tcl Console Messages Log

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT
synth_1	constrs_1	Not started												
impl_1	constrs_1	Not started												

En setting seleccionamos VHDL

Settings

Add Sources

Language Templates

IP Catalog

IP INTEGRATOR

Create Block Design

Open Block Design

Generate Block Design

SIMULATION

Run Simulation

RTL ANALYSIS

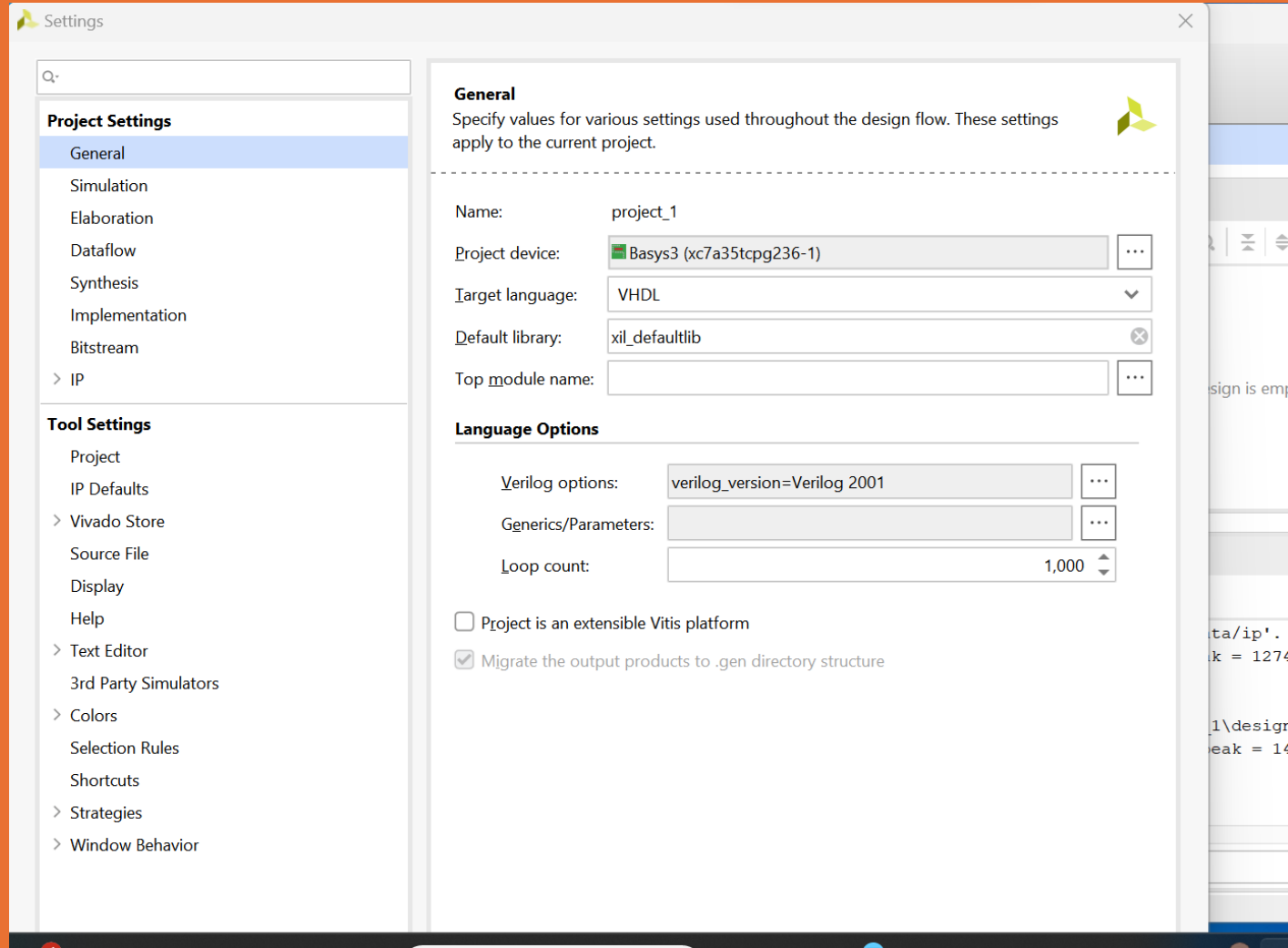
Run Linter

Open Elaborated Design

SYNTHESIS

Run Synthesis

Create and add an IP subsystem to the project



project_3 - [C:/hlocal/project_base/project_3/project_3.xpr] - Vivado 2023.1

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

Flow Navigator

- PROJECT MANAGER
 - Settings
 - Add Sources
 - Language Templates
 - IP Catalog
- IP INTEGRATOR**
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- SIMULATION
 - Run Simulation
- RTL ANALYSIS
 - Run Linter
 - Open Elaborated Design
- SYNTHESIS
 - Run Synthesis

BLOCK DESIGN - design_1

Sources Design Signals ? _ □ □

design_1

Properties ? _ □ □ ×

Select an object to see properties

Diagram ? □ □ ×

This design is empty. Press the + button to add IP.

Tcl Console × Messages Log Reports Design Runs ? _ □ □

```
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx2/Vivado/2023.1/data/ip'.
create_project: Time (s): cpu = 00:00:08 ; elapsed = 00:00:08 ; gain = 55.934
create_bd_design "design_1"
Wrote : <C:/hlocal/project_base/project_3/project_3.srcs/sd_1/block_design_1.bd
create_bd_design: Time (s): cpu = 00:00:04 ; elapsed = 00:00:04 ; gain = 0.000
update_compile_order -fileset sources_1
```

Type a Tcl command here

Añadimos los IPs
El primero microblaze

project_3 - [C:/hlocal/project_base/project_3/project_3.xpr] - Vivado 2023.1

File Edit **Flow** Tools Reports Window Layout View Help

Flow Navigator

- PROJECT MANAGER
 - Settings
 - Add Sources
 - Language Templates
 - IP Catalog
- IP INTEGRATOR**
 - Create Block Design
 - Open Block Design
 - Generate Block Design
- SIMULATION
- RTL ANALYSIS

BLOCK DESIGN - design_1 *

Sources Design Signals

design_1

- microblaze_0 (MicroBlaze:11.0)

Properties

Select an object to see properties

Tcl Console

```
open_bd_design {C:/hlocal/project_base/project_3/project_3.xpr/sources/1700/design_1/design_1.tcl}
open_bd_design {C:/hlocal/project_base/project_3/project_3.xpr/sources/1700/design_1/design_1.tcl}
open_bd_design {C:/hlocal/project_base/project_3/project_3.xpr/sources/1700/design_1/design_1.tcl}
```

Diagram Address Editor

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

microblaze_0

MicroBlaze

INTERRUPT
DEBUG
Clk
Reset

DLMB
ILMB

Run Block Automation

Seleccionar 16kB de RAM

¿Qué hemos añadido?

- El clock wizard
- La memoria local
- El processor System Reset
- El módulo de debug

¿Qué nos falta?

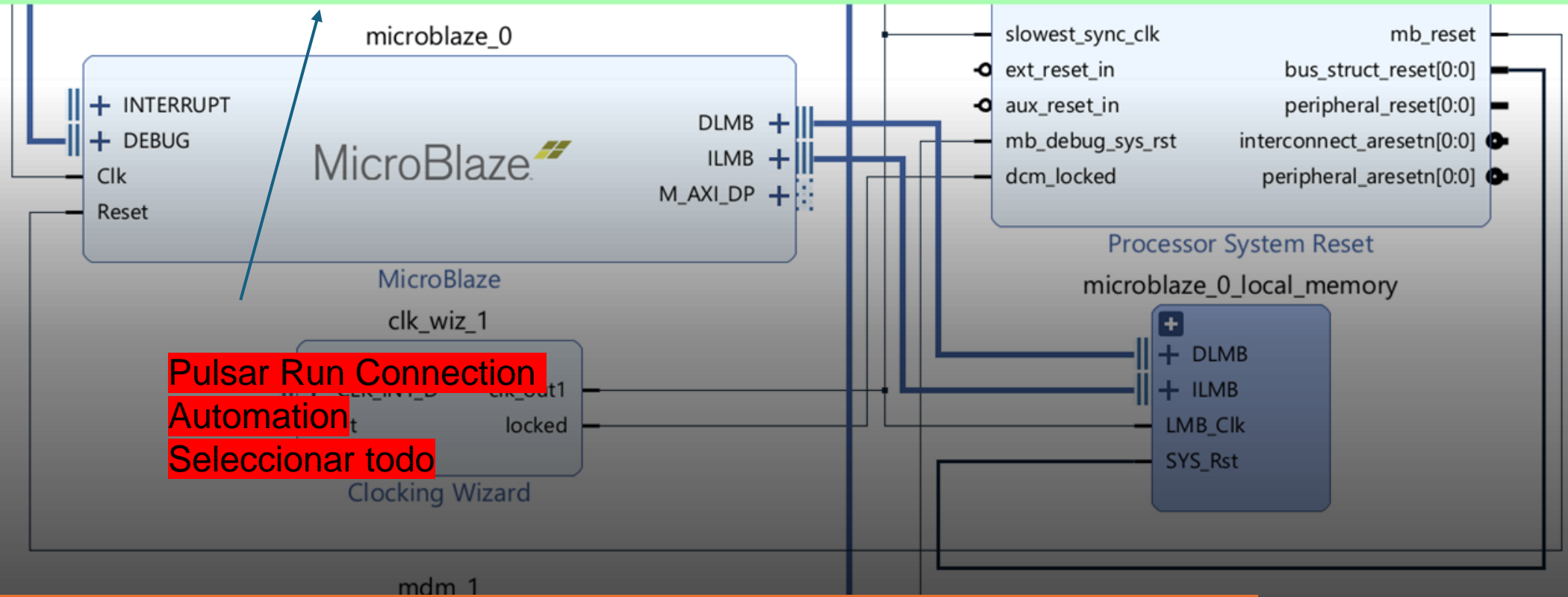
- Desde la pestaña board:
 - Añadimos system clock
 - Añadimos Axi Uart Lite
 - Añadimos los switches y leds desde la pestaña board
 - Se añade el controlador de axi automáticamente
- Pulsamos Run Connection Automation y seleccionamos todo
- Ya tenemos nuestro sistema empotrado

BLOCK DESIGN - design_1 *

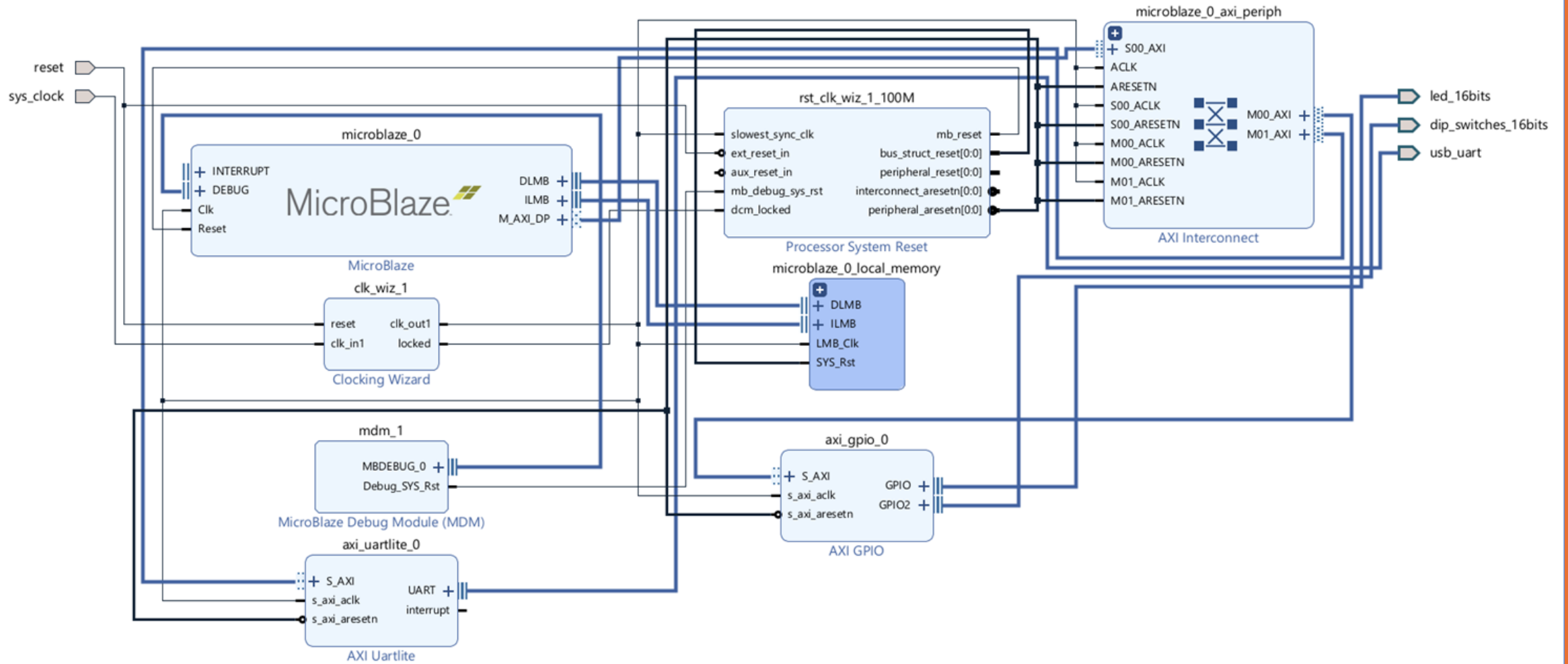
Diagram x Address Editor x

Diagram toolbar with icons for zoom, pan, and other editing tools. A dropdown menu shows "Default View".

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



Pulsar Run Connection
Automation
Seleccionar todo





- Pulsamos generate block
- Creamos el wrapper (desde la pestaña sources)
- Desde el módulo de nuestro diseño, botón derecho Create HDL Wrapper
- 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

¿Qué hacemos si da error en el módulo de reloj?

- Entramos en su configuración (doble click)
- Seleccionamos un reloj como sys_clock y el otro como custom
- Volvemos a pulsar Run Connection Automation y generar bitstream
- File 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
 - 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)
- Finish

Se ha creado un directorio workspace/design_wrapper/hw con 3 ficheros

*.bit



*.mmi

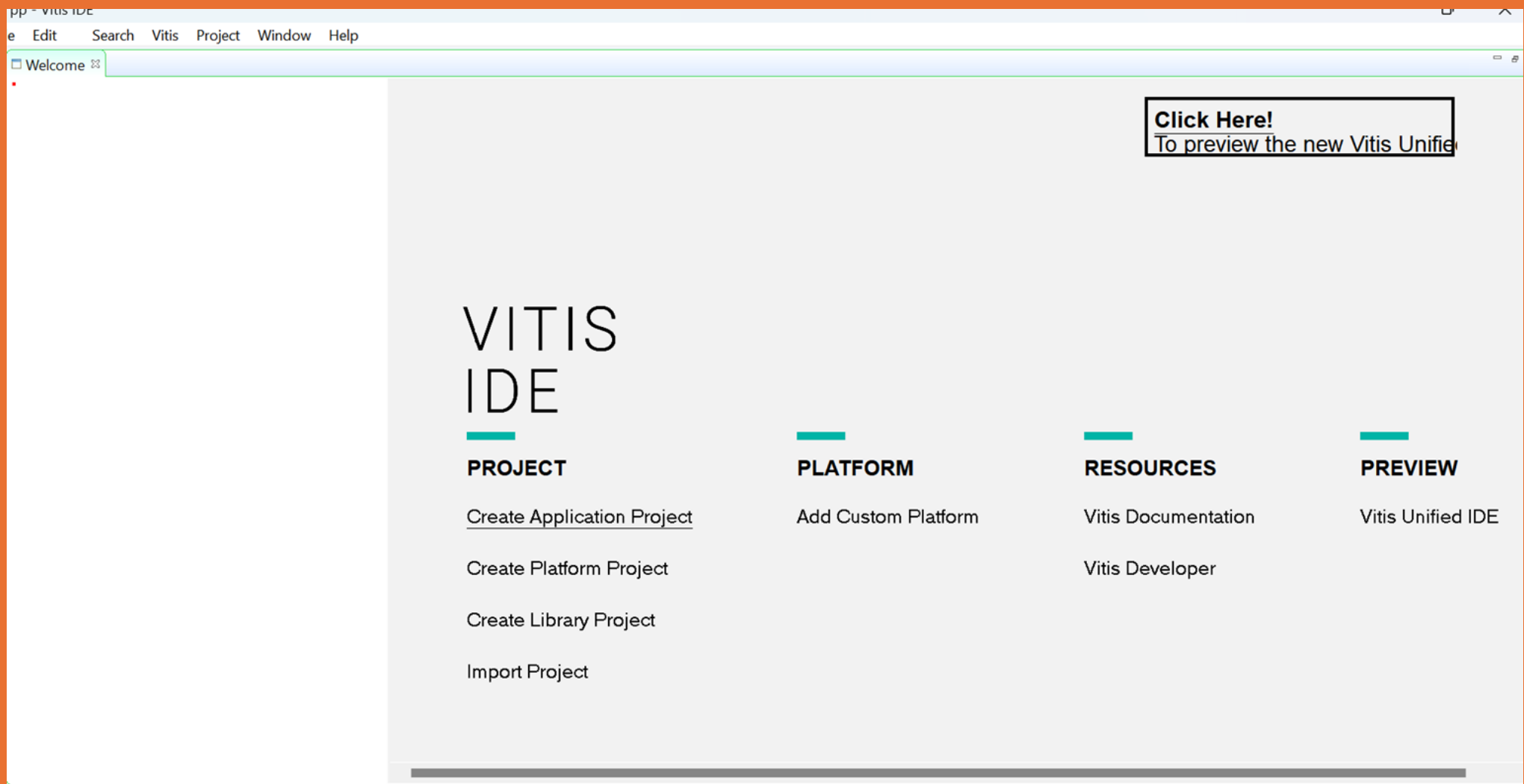
*.xsa (una copia del anterior)


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
- Seleccionar fichero *.elf (donde aparece bootloop)
- Program
- El software está cargado en la FPGA



- 
- Las siguientes diapositivas son para modificar el software.
 - Tenemos que controlar:
 - La uart (como salida con `xil_printf` y como entrada con la función que se proporciona `get_number`)
 - Los leds
 - Los switches
- 




- 
- 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

Modificam
os el
software
Inicializamos
los GPIOs
Canal 1 como
salida (leds)

- `print("\r\nRunning GpioOutputExample() for axi_gpio_0...\r\n");`
- `volatile int Delay;`
- `u32 LedBit;`
- `u32 LedLoop;`
- `int Status;`
- `XGpio Gpio;`
- `u32 DataRead;`
- `/* Initialize the GPIO driver so that it's ready to use,`
- `specify the device ID that is generated in xparameters.h */`
- `Status = XGpio_Initialize(&Gpio, XPAR_AXI_GPIO_0_DEVICE_ID);`
- `if (Status != XST_SUCCESS) {`
- `return XST_FAILURE; }`
- `/* Set the direction for all signals to be outputs in channel 1*/`
- `XGpio_SetDataDirection(&Gpio, 1, 0x0);`
- `/* Set the GPIO outputs to low */`
- `XGpio_DiscreteWrite(&Gpio, 1, 0x0);`


modificamos el software
Se encienden consecutivamente los leds



```
•for (LedBit = 0x0; LedBit < 16; LedBit++) {  
  
    for (LedLoop = 0; LedLoop < 1000000; LedLoop++) {  
  
        /* Set the GPIO Output to High */  
        XGpio_DiscreteWrite(&Gpio, 1,  
        1 << LedBit);}}
```

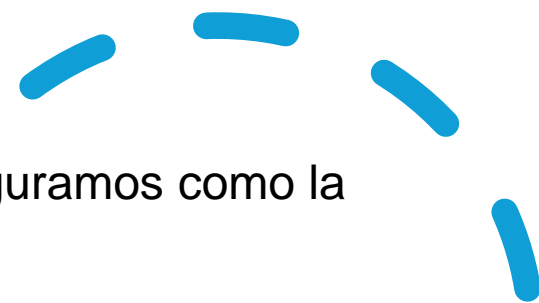



Configuramos el canal 2 como entrada (switches)

- 
- /* Leemos switches y escribimos en leds */
 - /* Set the direction for all signals to be inputs in channel 2 */
 - XGpio_SetDataDirection(&Gpio, 2, 0xFFFFFFFF);
 - /* Read the state of the data so that it can be verified */
 - DataRead = XGpio_DiscreteRead(&Gpio, 2);
 - XGpio_DiscreteWrite(&Gpio, 1, DataRead);
 - xil_printf("el dato leído es %d", DataRead);



Interfaz serie

- 
- Abrimos un interfaz serie y lo configuramos como la uart (normalmente a 9600 baudios)
 - Window-> show views->Terminal
 - Configurar el terminal serie
 - Seleccionar el port y demás parametros
 - Cargar el *.bit