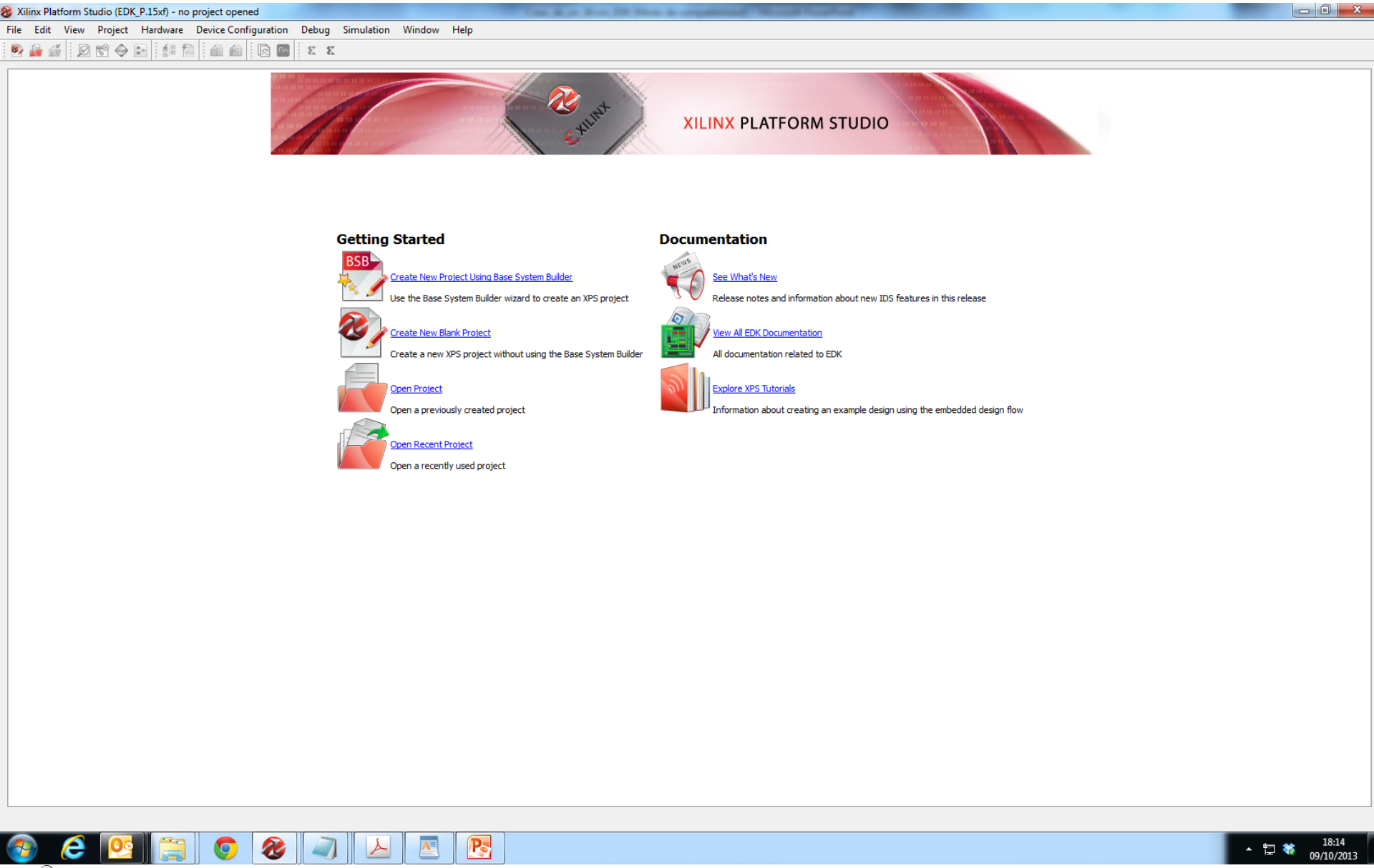
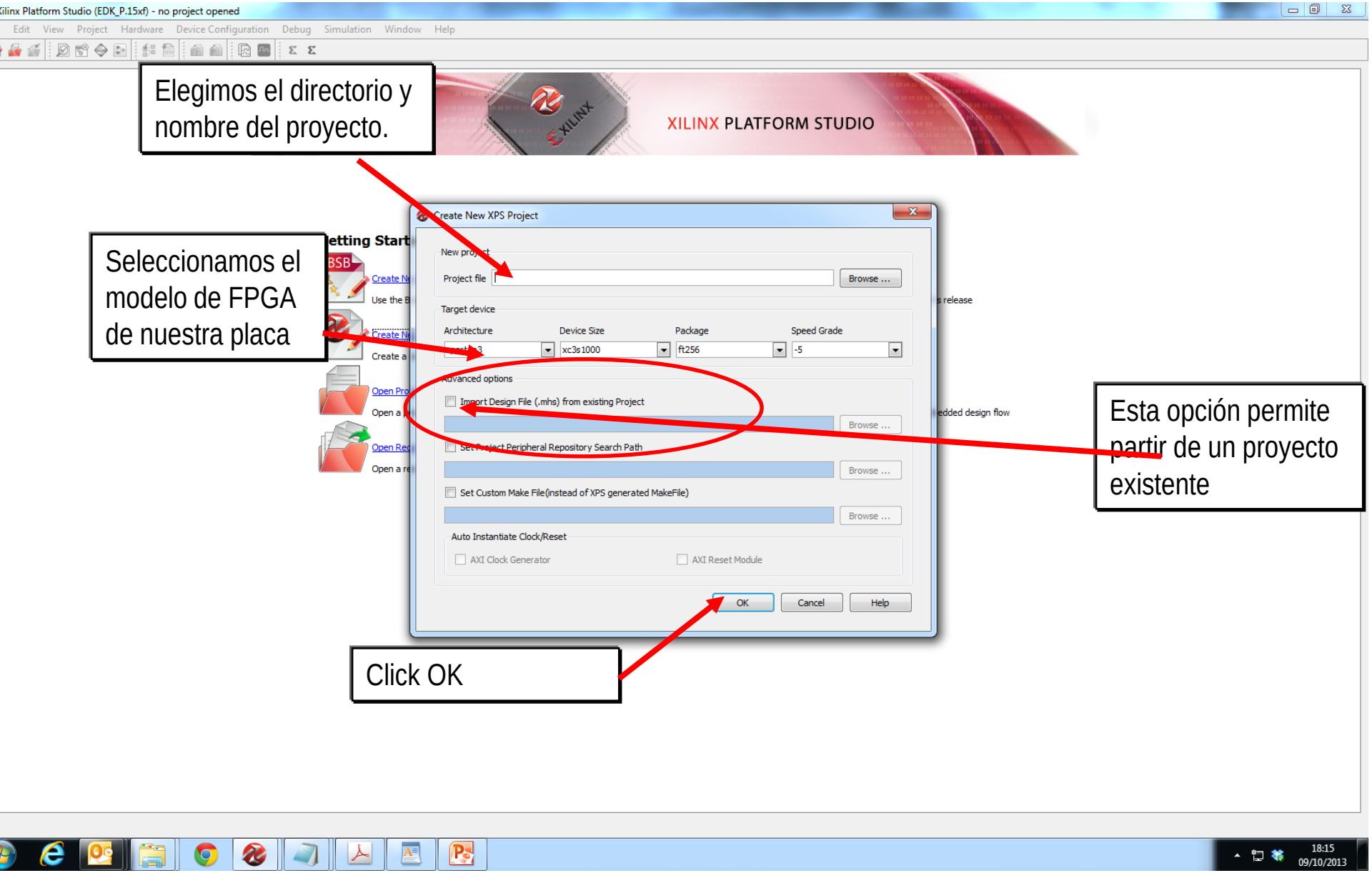
Creación de un Sistema empotrado con EDK

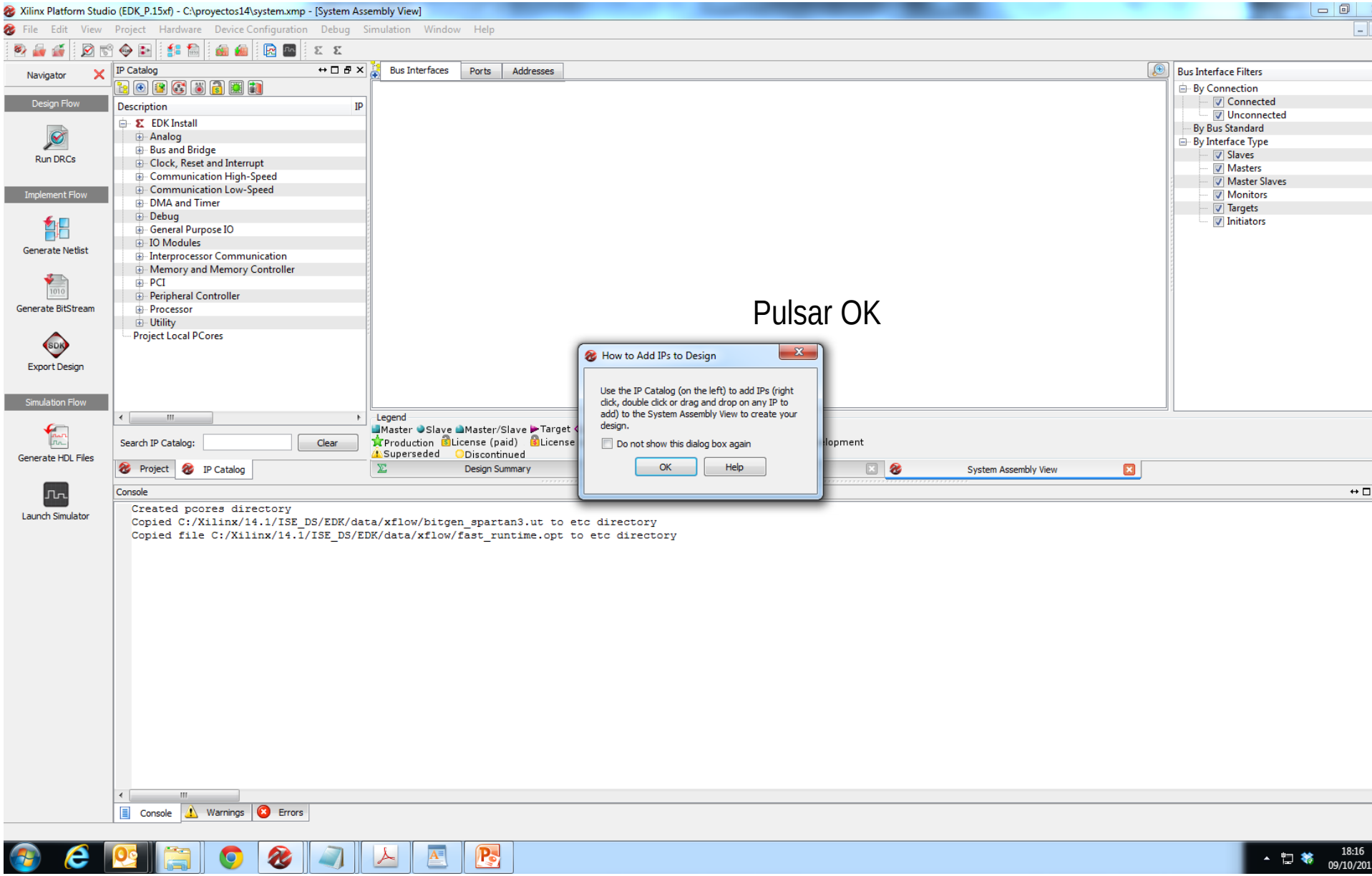
Hortensia Mecha López

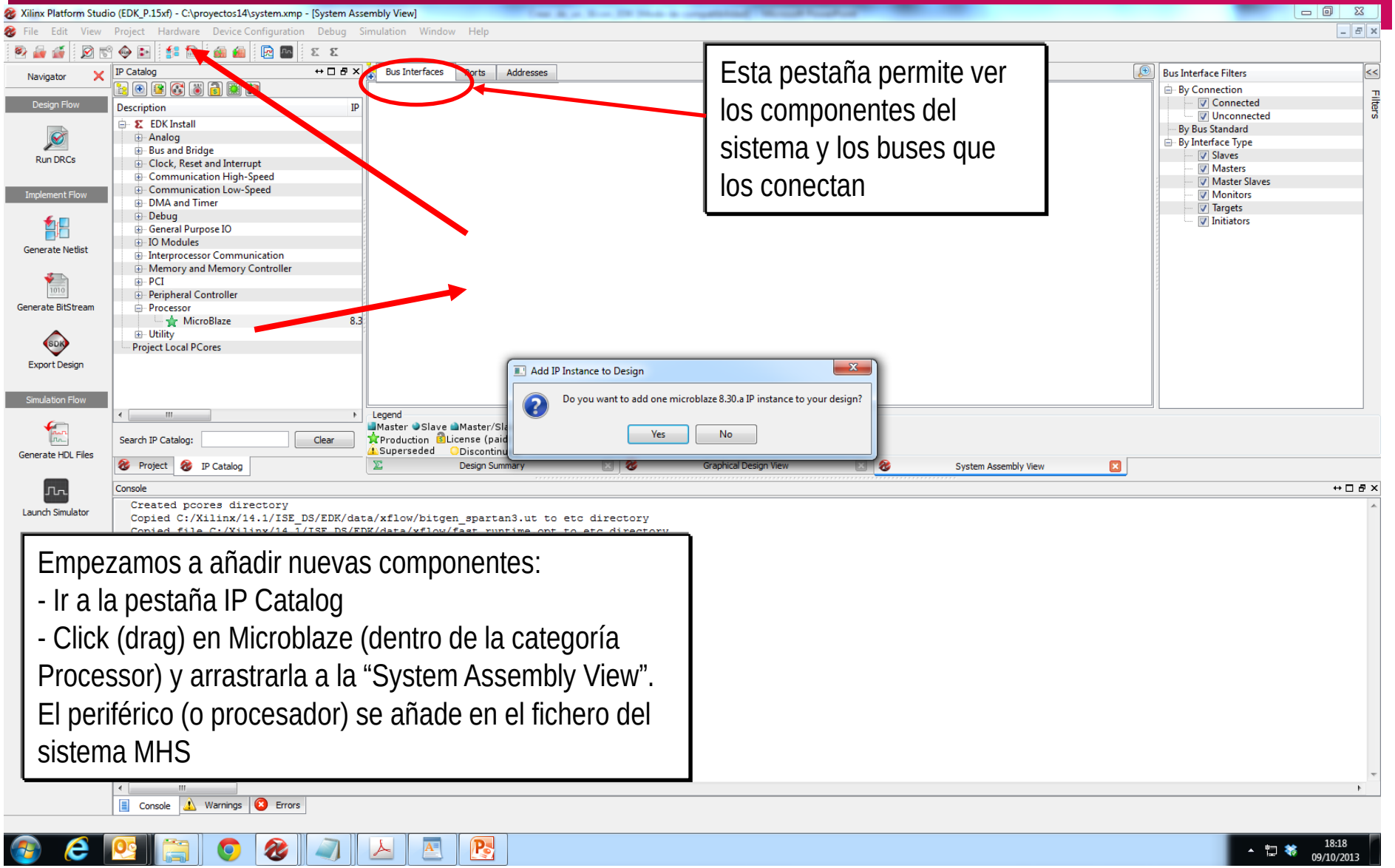
Objetivos:

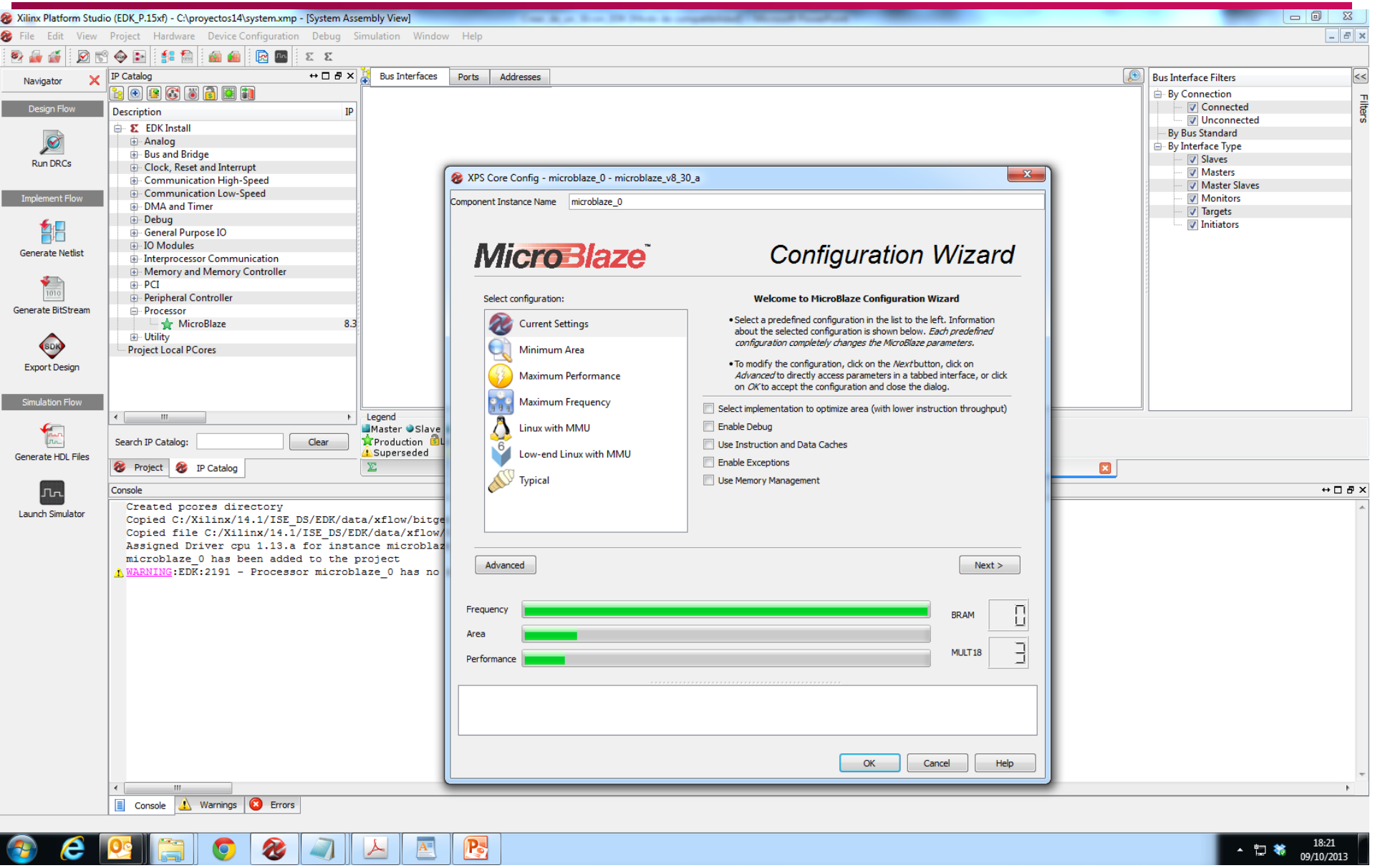
- En esta primera práctica se trata de:
  - Aprender a crear un proyecto basado en microblaze y con algunos periféricos utilizando la herramienta EDK
  - Escribir una aplicación sencilla en un lenguaje de programación de alto nivel, compilarla y ejecutarla en el microprocesador empotrado.


Xilinx Platform Studio (EDK_P.15xf) - no project opened
File Edit View Project Hardware Device Configuration Debug Simulation Window Help
XILINX PLATFORM STUDIO
Getting Started
Create New Project Using Base System Builder
Use the Base System Builder wizard to create an XPS project
Create New Blank Project
Create a new XPS project without using the Base System Builder
Open Project
Open a previously created project
Open Recent Project
Open a recently used project
Documentation
See What's New
Release notes and information about new IDS features in this release
View All EDK Documentation
All documentation related to EDK
Explore XPS Tutorials
Information about creating an example design using the embedded design flow
18:14
09/10/2013


Xilinx Platform Studio (EDK_P.15xf) - no project opened
Edit View Project Hardware Device Configuration Debug Simulation Window Help
XILINX PLATFORM STUDIO
Elegimos el directorio y nombre del proyecto.
Seleccionamos el modelo de FPGA de nuestra placa
Create New XPS Project
New project
Project file
Browse ...
Target device
Architecture
Device Size
Package
Speed Grade
xc3s1000
ft256
-5
Advanced options
Import Design File (.mhs) from existing Project
Set Project Peripheral Repository Search Path
Set Custom Make File(instead of XPS generated MakeFile)
Auto Instantiate Clock/Reset
AXI Clock Generator
AXI Reset Module
OK
Cancel
Help
Esta opción permite partir de un proyecto existente
Click OK
18:15
09/10/2013

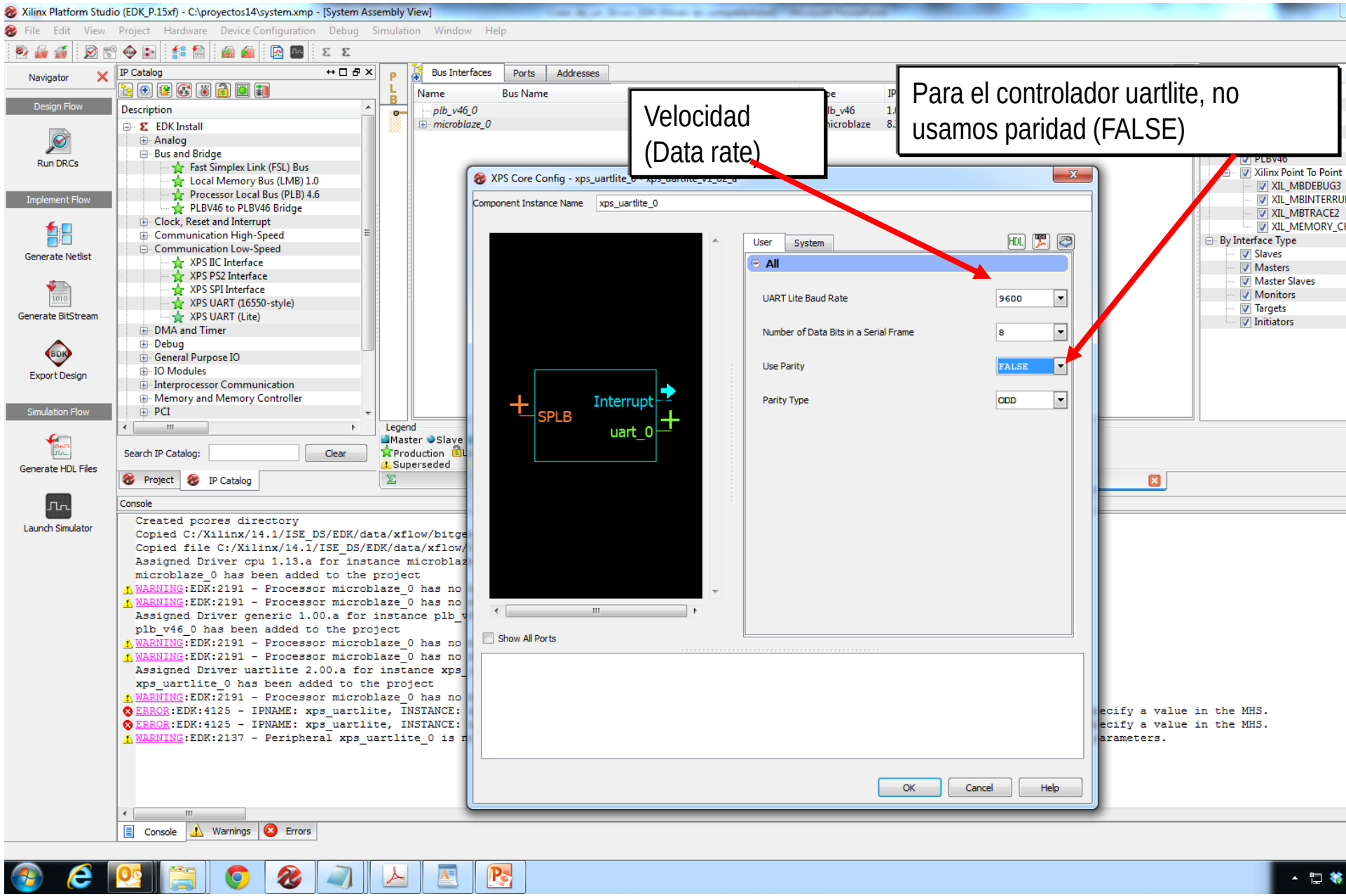

Xilinx Platform Studio (EDK_P.15xf) - C:\proyectos14\system.xmp - [System Assembly View]
File Edit View Project Hardware Device Configuration Debug Simulation Window Help
Navigator
Design Flow
Run DRCs
Implement Flow
Generate Netlist
Generate BitStream
Export Design
Simulation Flow
Generate HDL Files
Launch Simulator
IP Catalog
Description
IP
EDK Install
Analog
Bus and Bridge
Clock, Reset and Interrupt
Communication High-Speed
Communication Low-Speed
DMA and Timer
Debug
General Purpose IO
IO Modules
Interprocessor Communication
Memory and Memory Controller
PCI
Peripheral Controller
Processor
Utility
Project Local PCores
Search IP Catalog:
Clear
Project
IP Catalog
Bus Interfaces
Ports
Addresses
Bus Interface Filters
By Connection
Connected
Unconnected
By Bus Standard
By Interface Type
Slaves
Masters
Master Slaves
Monitors
Targets
Initiators
Pulsar OK
How to Add IPs to Design
Use the IP Catalog (on the left) to add IPs (right click, double click or drag and drop on any IP to add) to the System Assembly View to create your design.
Do not show this dialog box again
OK
Help
Legend
Master Slave Master/Slave Target
Production License (paid) License
Superseded Discontinued
Design Summary
System Assembly View
Console
Created pcores directory
Copied C:/Xilinx/14.1/ISE_DS/EDK/data/xflow/bitgen_spartan3.ut to etc directory
Copied file C:/Xilinx/14.1/ISE_DS/EDK/data/xflow/fast_runtime.opt to etc directory
Console
Warnings
Errors
18:16
09/10/201


Xilinx Platform Studio (EDK_P.15xf) - C:\proyectos14\system.xmp - [System Assembly View]
Esta pestaña permite ver los componentes del sistema y los buses que los conectan
Add IP Instance to Design
Do you want to add one microblaze 8.30.a IP instance to your design?
Yes
No
Empezamos a añadir nuevas componentes:
- Ir a la pestaña IP Catalog
- Click (drag) en Microblaze (dentro de la categoría Processor) y arrastrarla a la “System Assembly View”.
El periférico (o procesador) se añade en el fichero del sistema MHS


Xilinx Platform Studio (EDK_P.15xf) - C:\proyectos14\system.xmp - [System Assembly View]
File Edit View Project Hardware Device Configuration Debug Simulation Window Help
Navigator
Design Flow
Run DRCs
Implement Flow
Generate Netlist
Generate BitStream
Export Design
Simulation Flow
Generate HDL Files
Launch Simulator
IP Catalog
Description IP
EDK Install
Analog
Bus and Bridge
Clock, Reset and Interrupt
Communication High-Speed
Communication Low-Speed
DMA and Timer
Debug
General Purpose IO
IO Modules
Interprocessor Communication
Memory and Memory Controller
PCI
Peripheral Controller
Processor
MicroBlaze 8.3
Utility
Project Local PCores
Search IP Catalog:
Clear
Project
IP Catalog
Bus Interfaces
Ports
Addresses
Legend
Master
Slave
Production
Superseded
Bus Interface Filters
By Connection
Connected
Unconnected
By Bus Standard
By Interface Type
Slaves
Masters
Master Slaves
Monitors
Targets
Initiators
Filters
Console
Created pcores directory
Copied C:/Xilinx/14.1/ISE_DS/EDK/data/xflow/bitge
Copied file C:/Xilinx/14.1/ISE_DS/EDK/data/xflow/
Assigned Driver cpu 1.13.a for instance microblaz
microblaze_0 has been added to the project
WARNING:EDK:2191 - Processor microblaze_0 has no
Console
Warnings
Errors
XPS Core Config - microblaze_0 - microblaze_v8_30_a
Component Instance Name
microblaze_0
MicroBlaze
Configuration Wizard
Select configuration:
Current Settings
Minimum Area
Maximum Performance
Maximum Frequency
Linux with MMU
Low-end Linux with MMU
Typical
Welcome to MicroBlaze Configuration Wizard
Select a predefined configuration in the list to the left. Information about the selected configuration is shown below. Each predefined configuration completely changes the MicroBlaze parameters.
To modify the configuration, click on the Next button, click on Advanced to directly access parameters in a tabbed interface, or click on OK to accept the configuration and close the dialog.
Select implementation to optimize area (with lower instruction throughput)
Enable Debug
Use Instruction and Data Caches
Enable Exceptions
Use Memory Management
Advanced
Next >
Frequency
Area
Performance
BRAM
MULT18
OK
Cancel
Help
18:21
09/10/2013

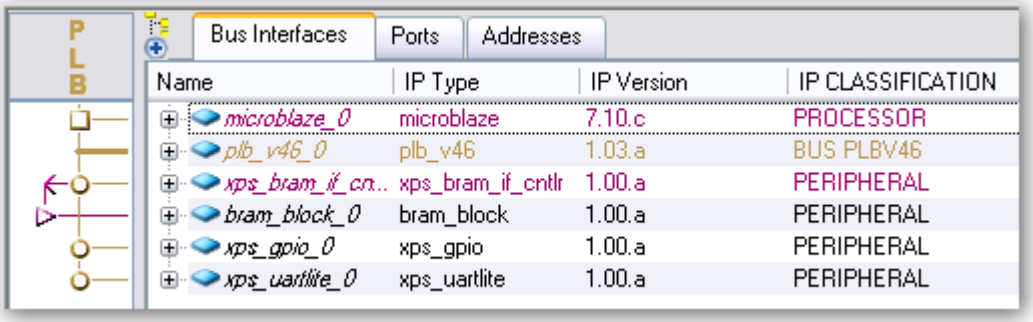
System assembly view: Añadir y conectar componentes

- Añadir los siguientes componentes al sistema:
  1. Procesador Microblaze
     - ✓ Processor -> Microblaze
  2. Bus local
     - ✓ Bus and Bridge -> Processor Local Bus (PLB)
  3. Controlador de memoria (Block Memory o BRAM)
     - ✓ Memory and Memory controller -> XPS BRAM controller
  4. Memoria (Block Memory o BRAM)
     - ✓ Memory and Memory controller -> Block RAM (BRAM)
  5. E/S de propósito general (Para la salida de los leds GPIO)
     - ✓ General Purpose IO -> XPS General Purpose IO
  6. Módulo comunicación serie UART
     - ✓ Communication Low-Speed -> XPS UART (lite)


Velocidad (Data rate)
Para el controlador uartlite, no usamos paridad (FALSE)
XPS Core Config - xps_uartlite_0
Component Instance Name
xps_uartlite_0
User
System
All
UART Lite Baud Rate
9600
Number of Data Bits in a Serial Frame
8
Use Parity
FALSE
Parity Type
ODD
SPLB
Interrupt
uart_0
Show All Ports
OK
Cancel
Help

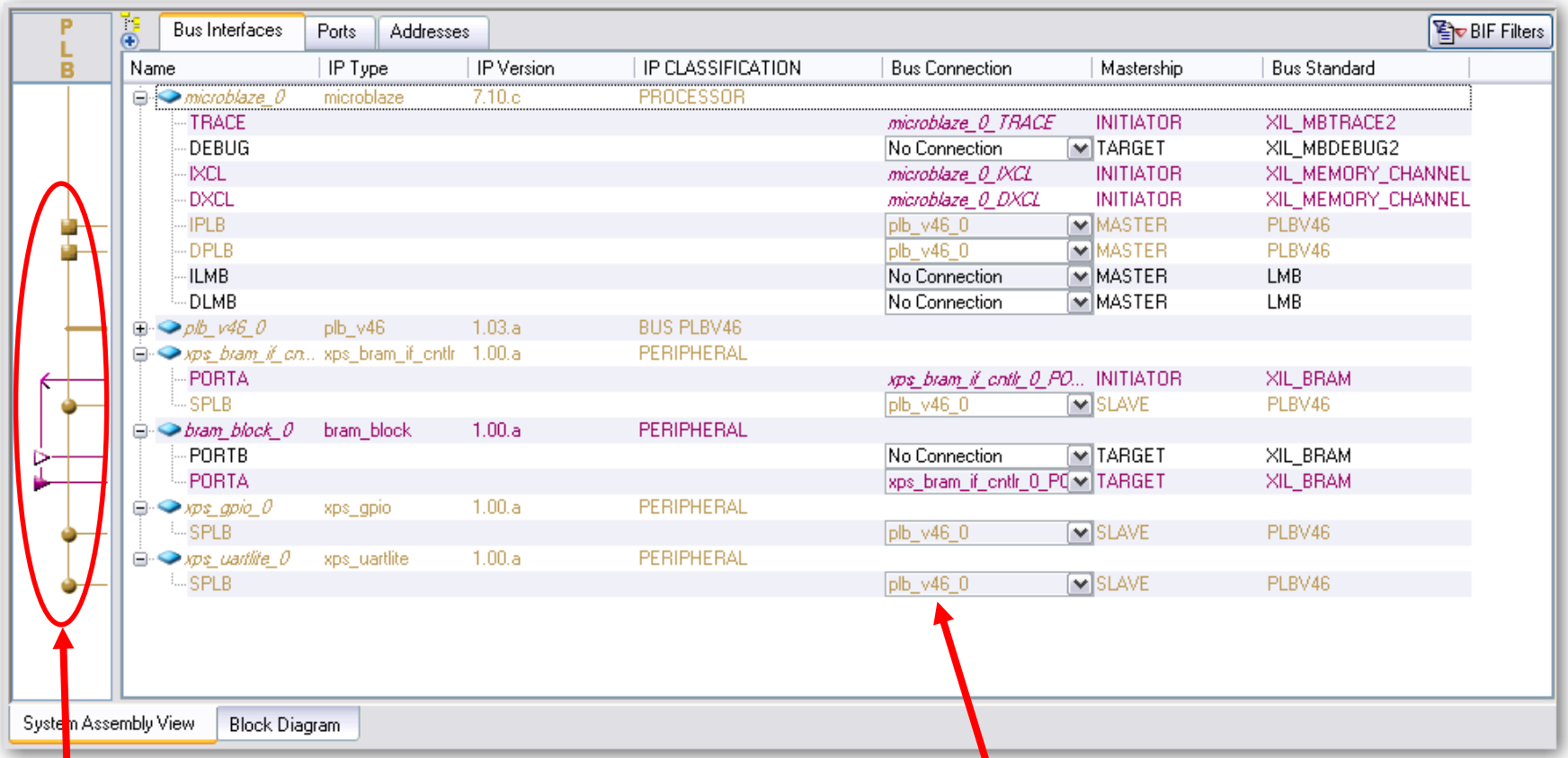
System assembly view: Añadir y conectar componentes

Vista del sistema después de añadir los componentes



A continuación hay que conectar y configurar las distintas componentes

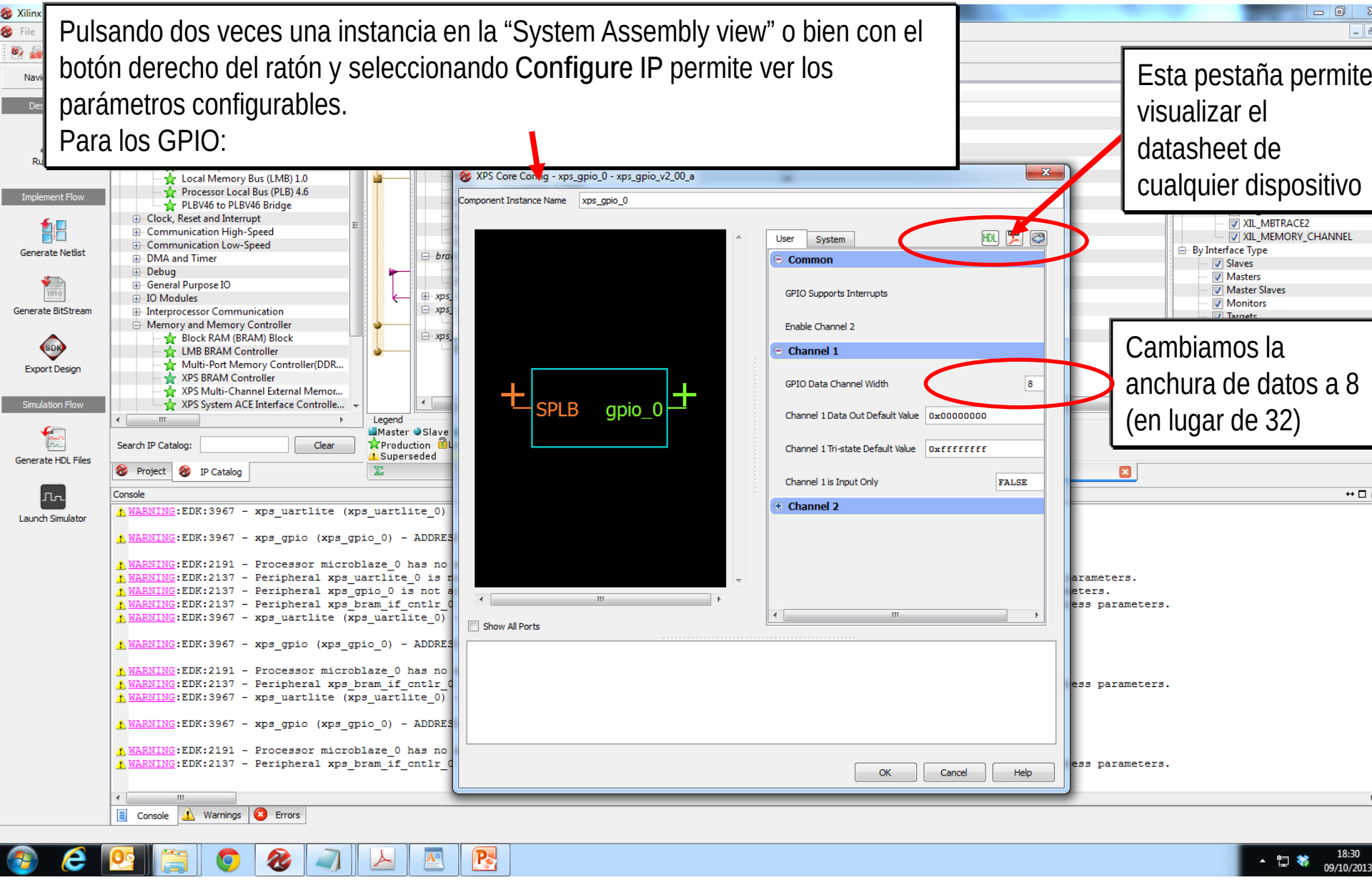
System assembly view: Añadir y conectar componentes



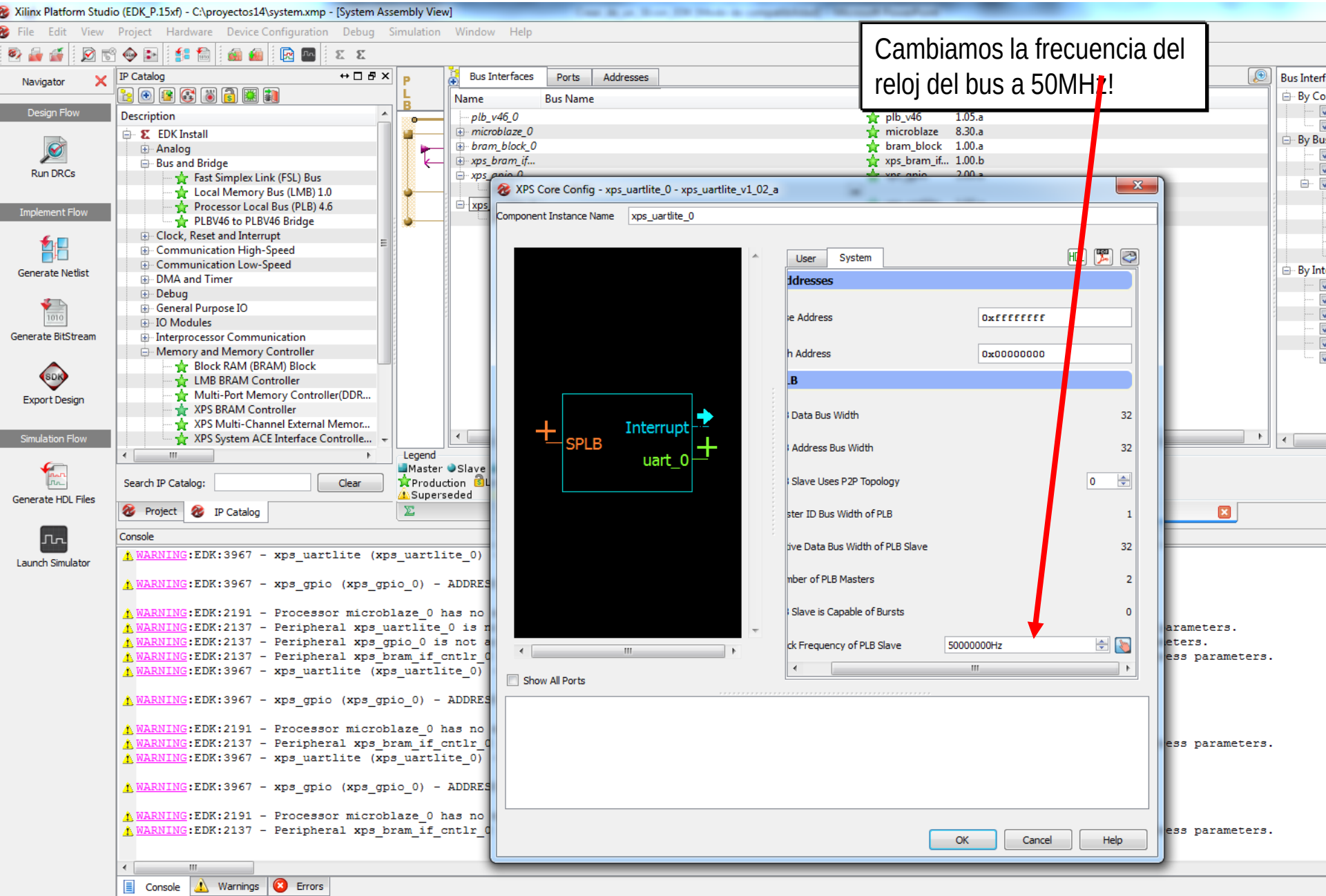
Clicking en los símbolos activa las conexiones

Para los bloques de BRAM pueden utilizarse tanto el puerto A como el B

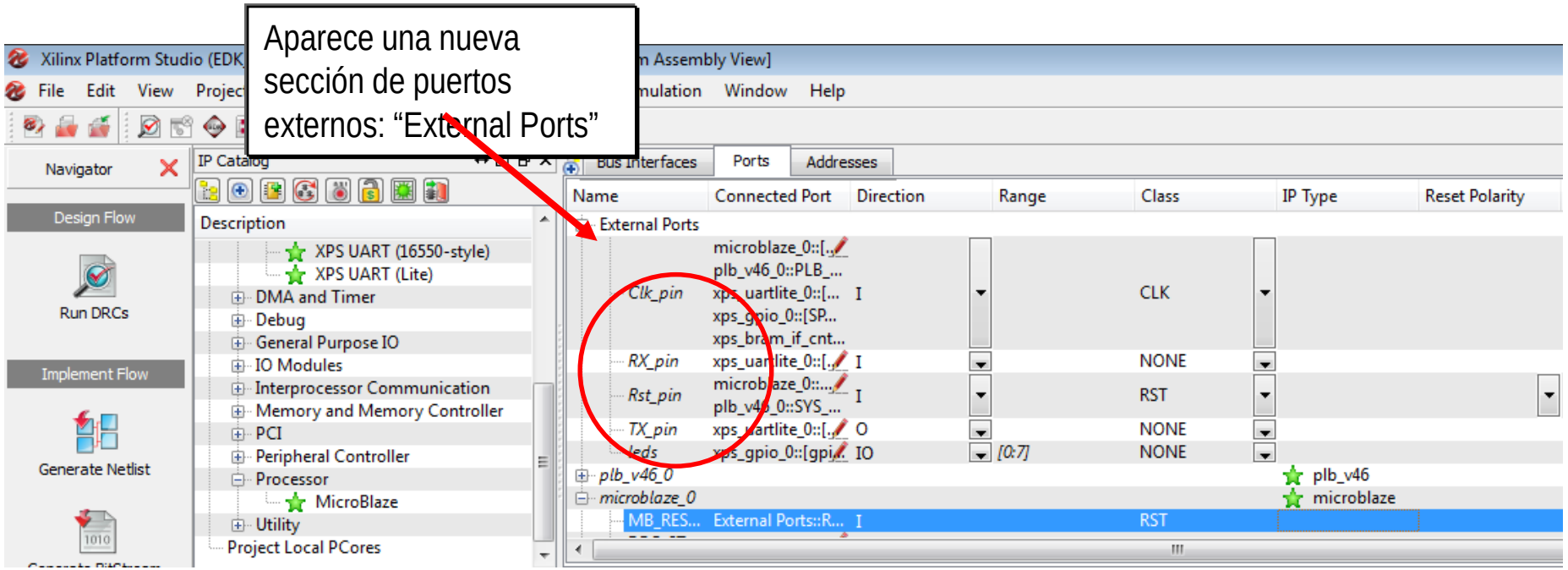
Pulsando dos veces una instancia en la “System Assembly view” o bien con el botón derecho del ratón y seleccionando **Configure IP** permite ver los parámetros configurables.
Para los GPIO:



Configuración de dispositivos: UART (II)

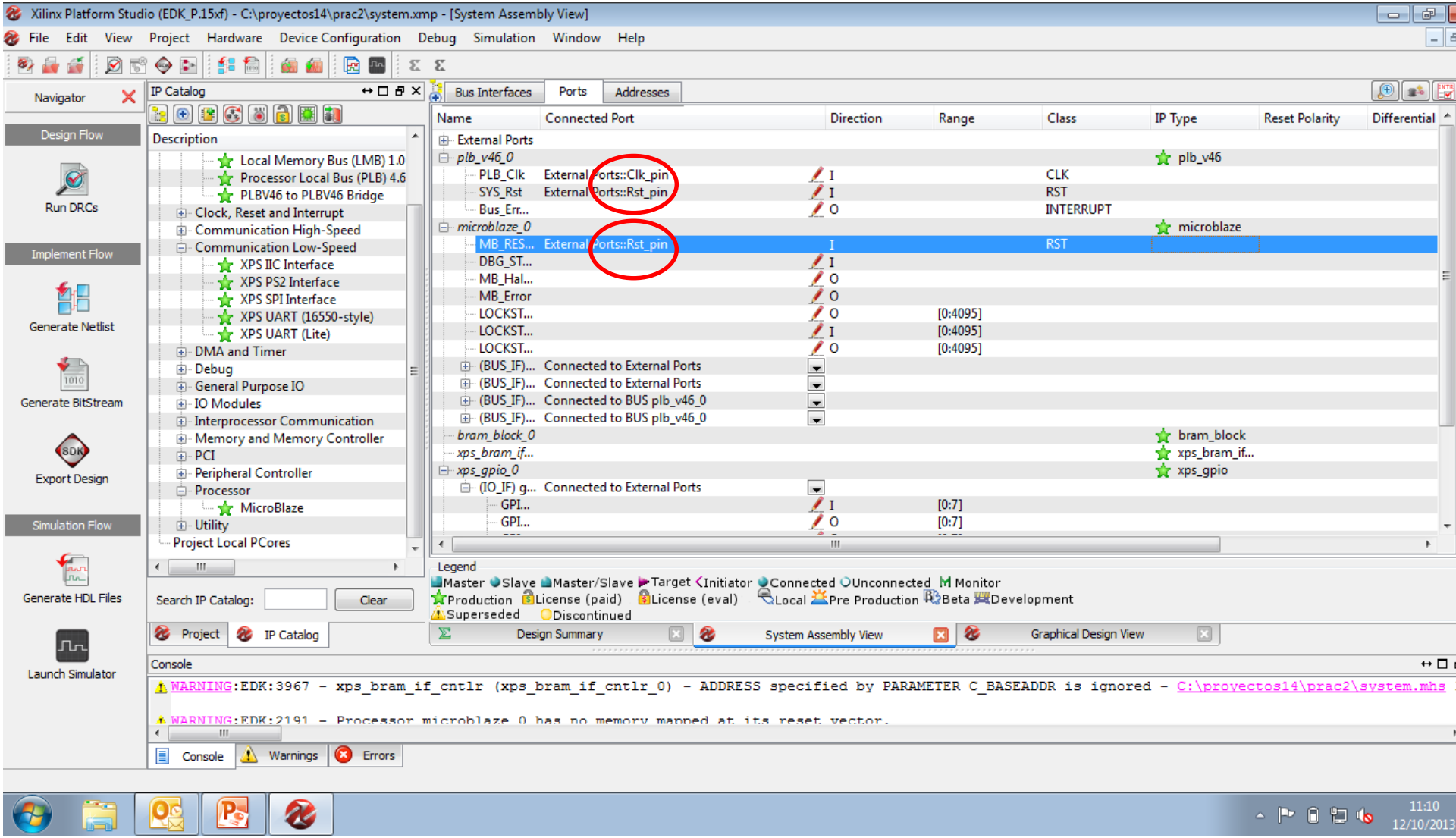


Configuración de los puertos internos y externos



En la pestaña “Ports” de la System Assembly View renombramos todas las señales. Para añadir un puerto externo, pulsamos en la “net combo-box” y seleccionamos “Make External”. Una nueva señal se añade al sistema como un puerto de E/S.

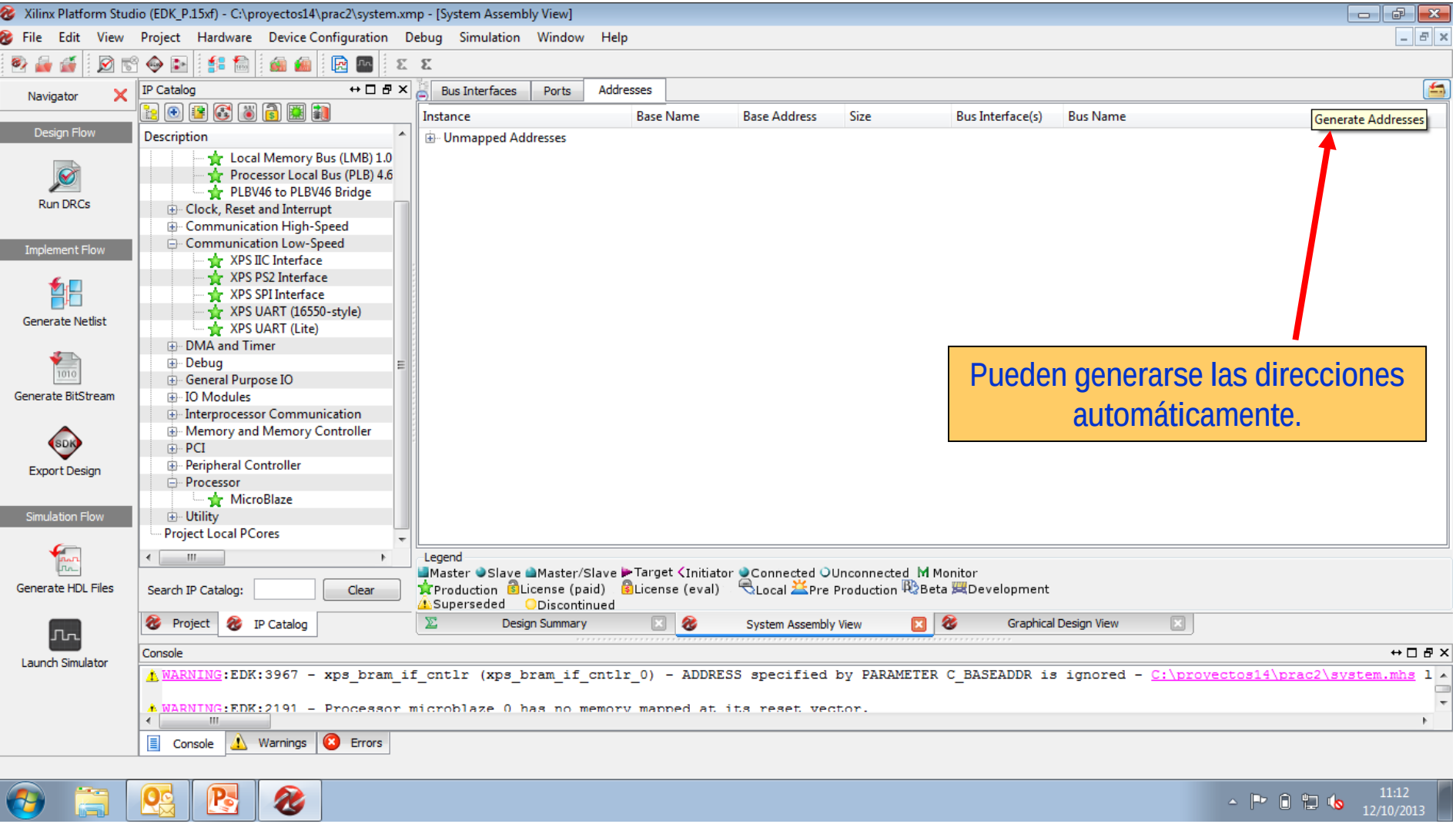
Configuración de los puertos internos y externos



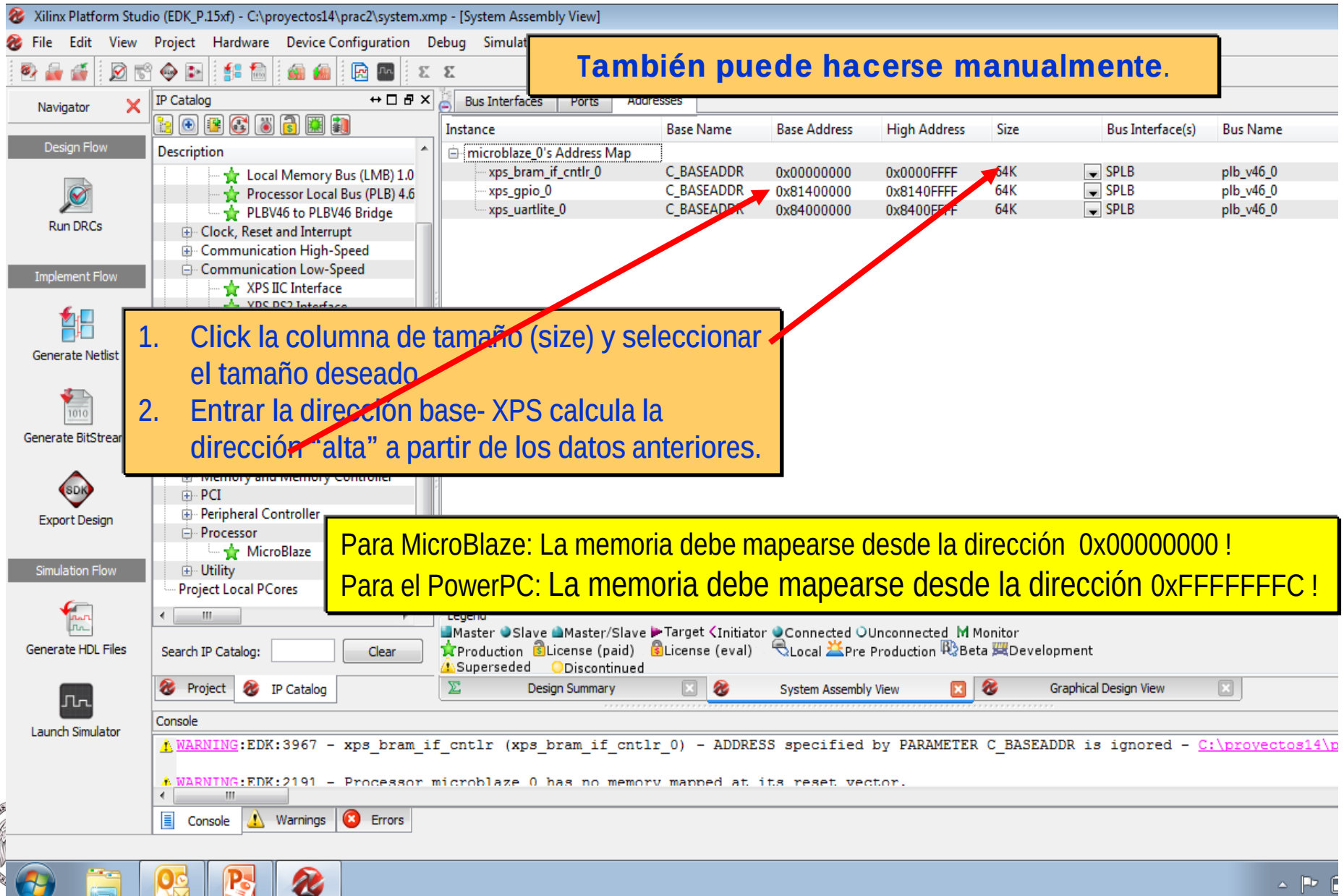
Configuración de los puertos internos y externos

- Señales que hay que convertir en puertos externos:
  - Uartlite:
    - ✓ Tx
    - ✓ Rx
  - GPIO:
    - ✓ GPIO_IO → leds
  - PLB:
    - ✓ Sys_Rst → reset
    - ✓ PLB_Clk → clk

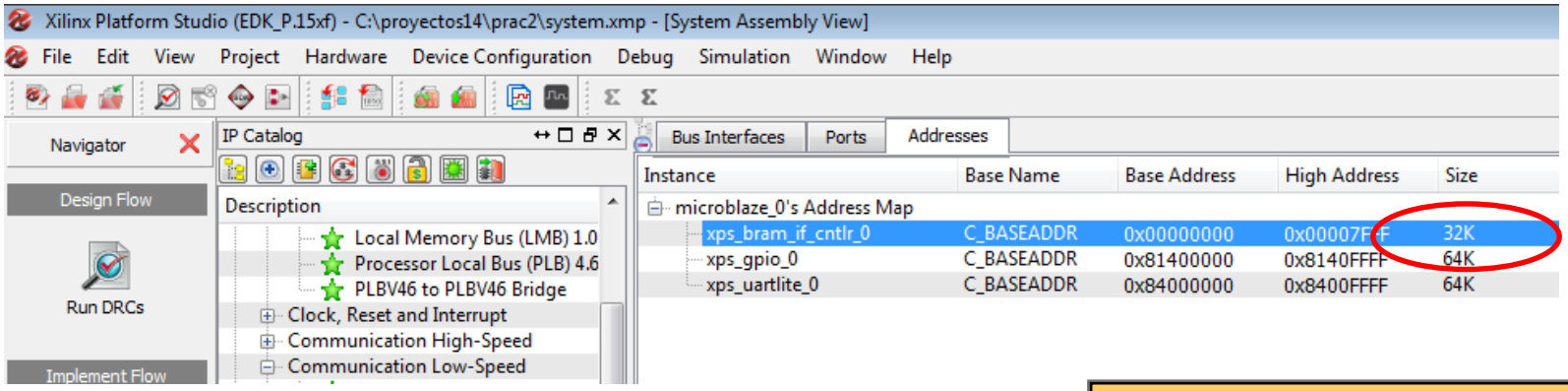
Mapeo de direcciones del sistema de memoria



Mapeo de direcciones del sistema de memoria

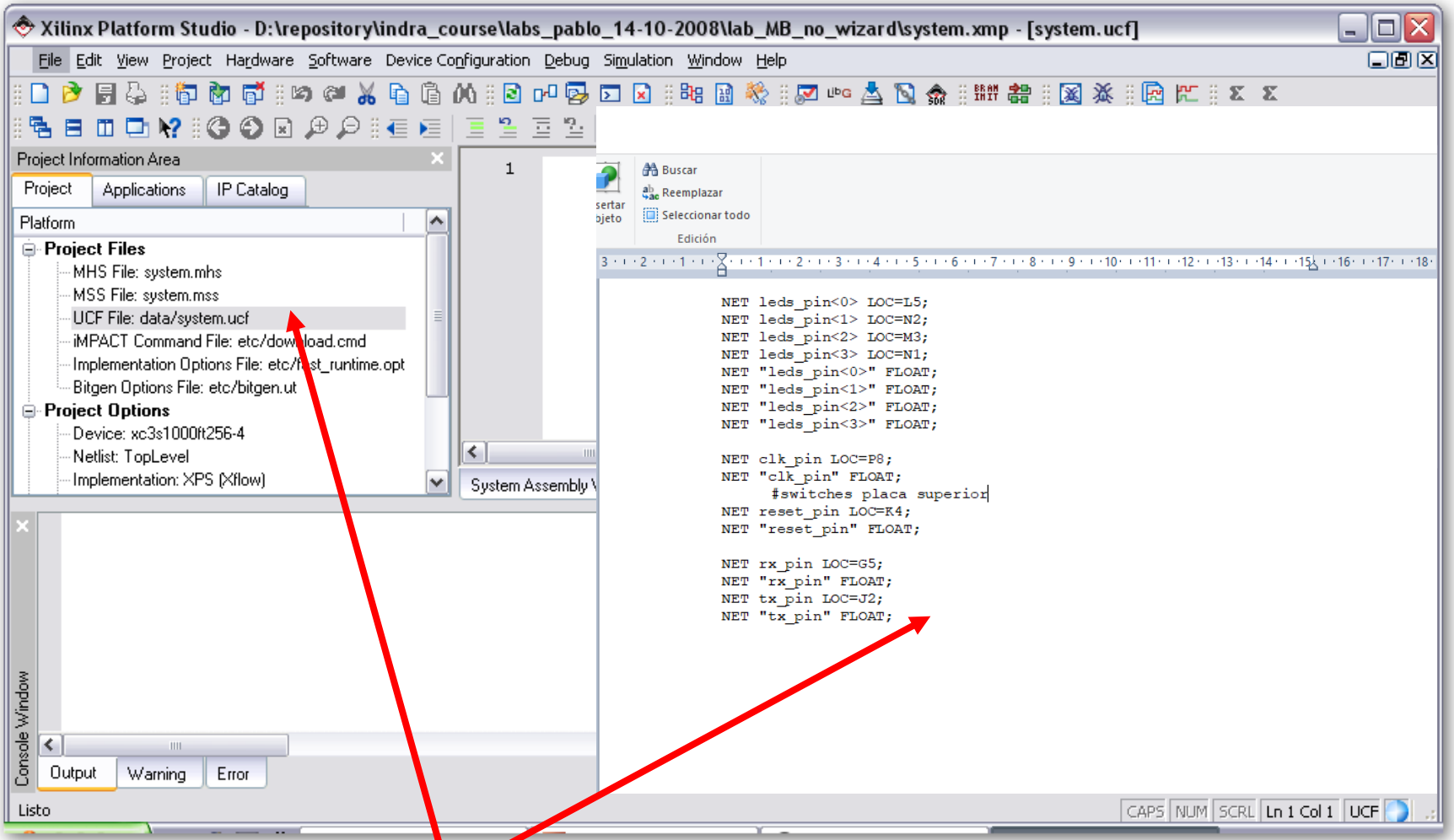


Mapeo de direcciones del sistema de memoria



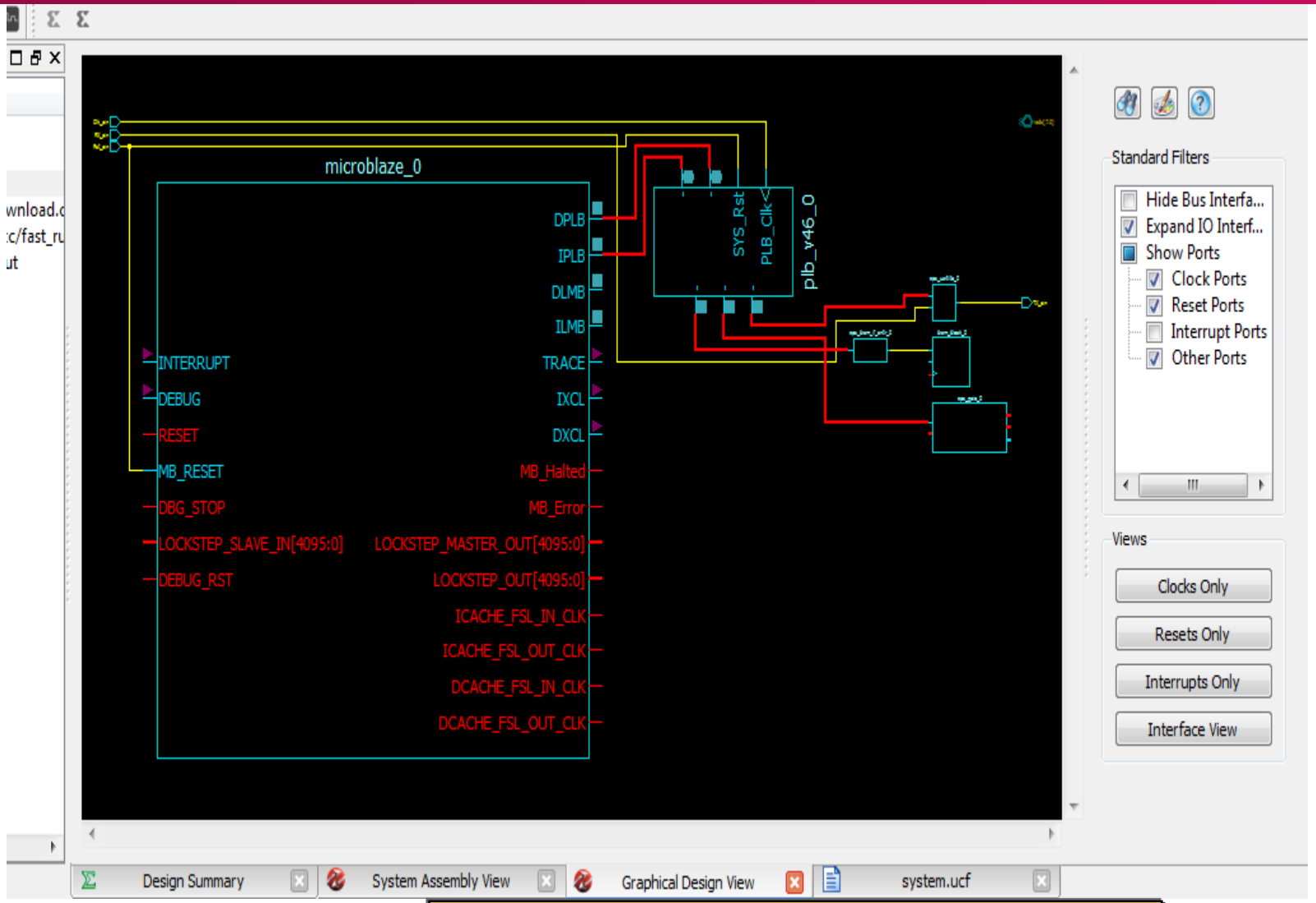
Cambiar el tamaño de la BRAM a 32KB

Añadir el fichero de restricciones de usuario (UCF)



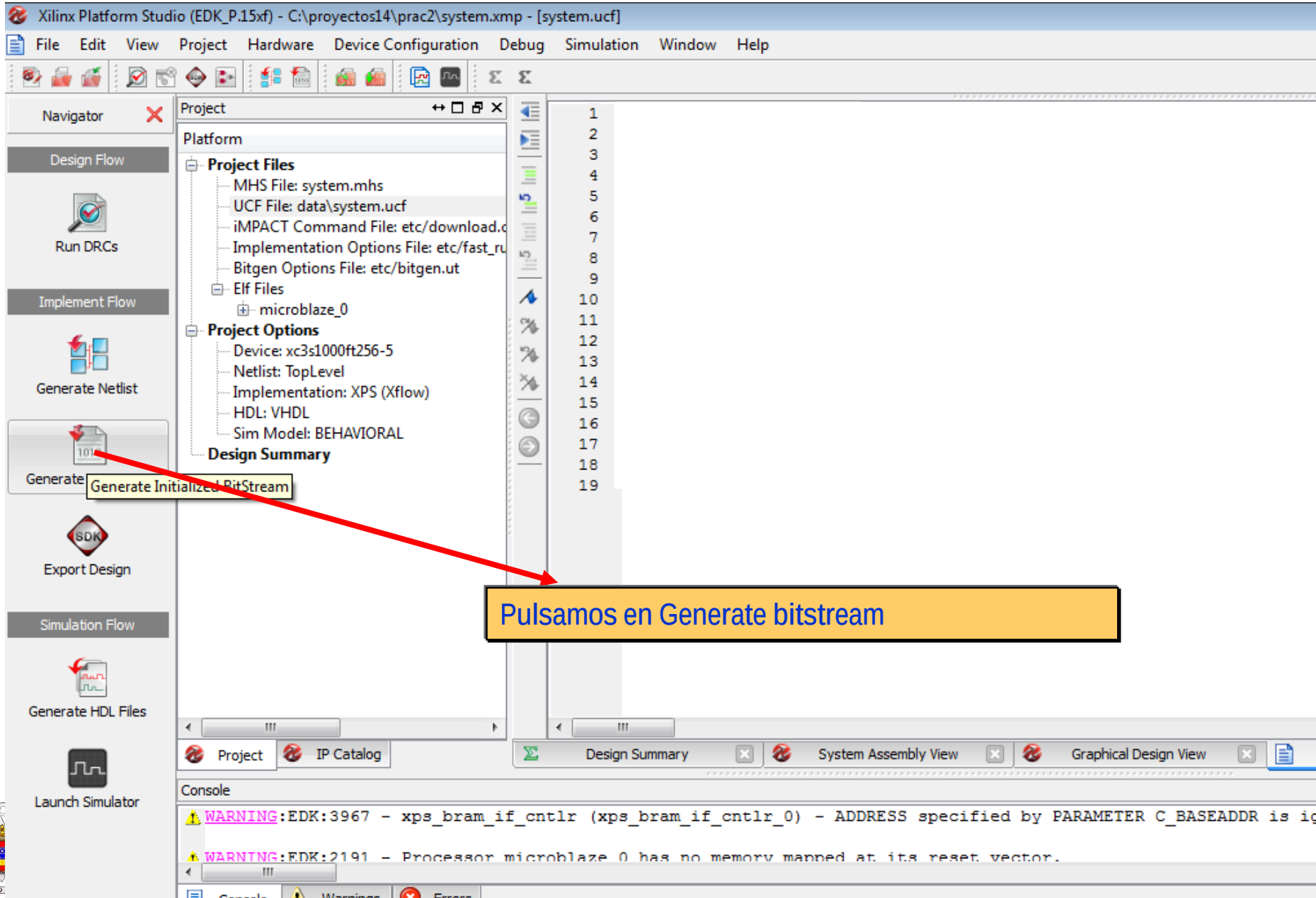
En la pestaña “Project”, clickear el fichero system.ucf para editarlo

Diagrama de bloques de la plataforma hardware

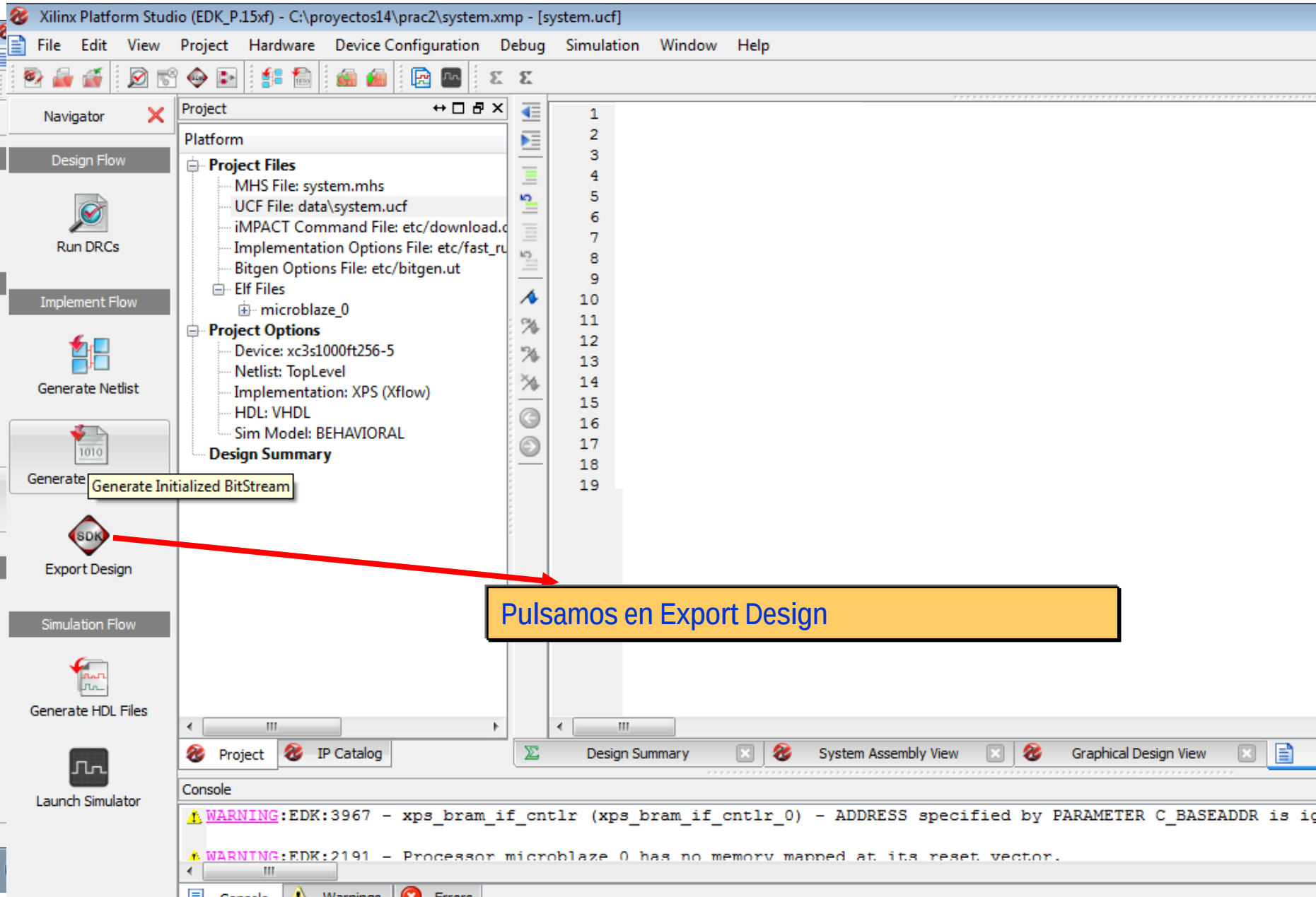


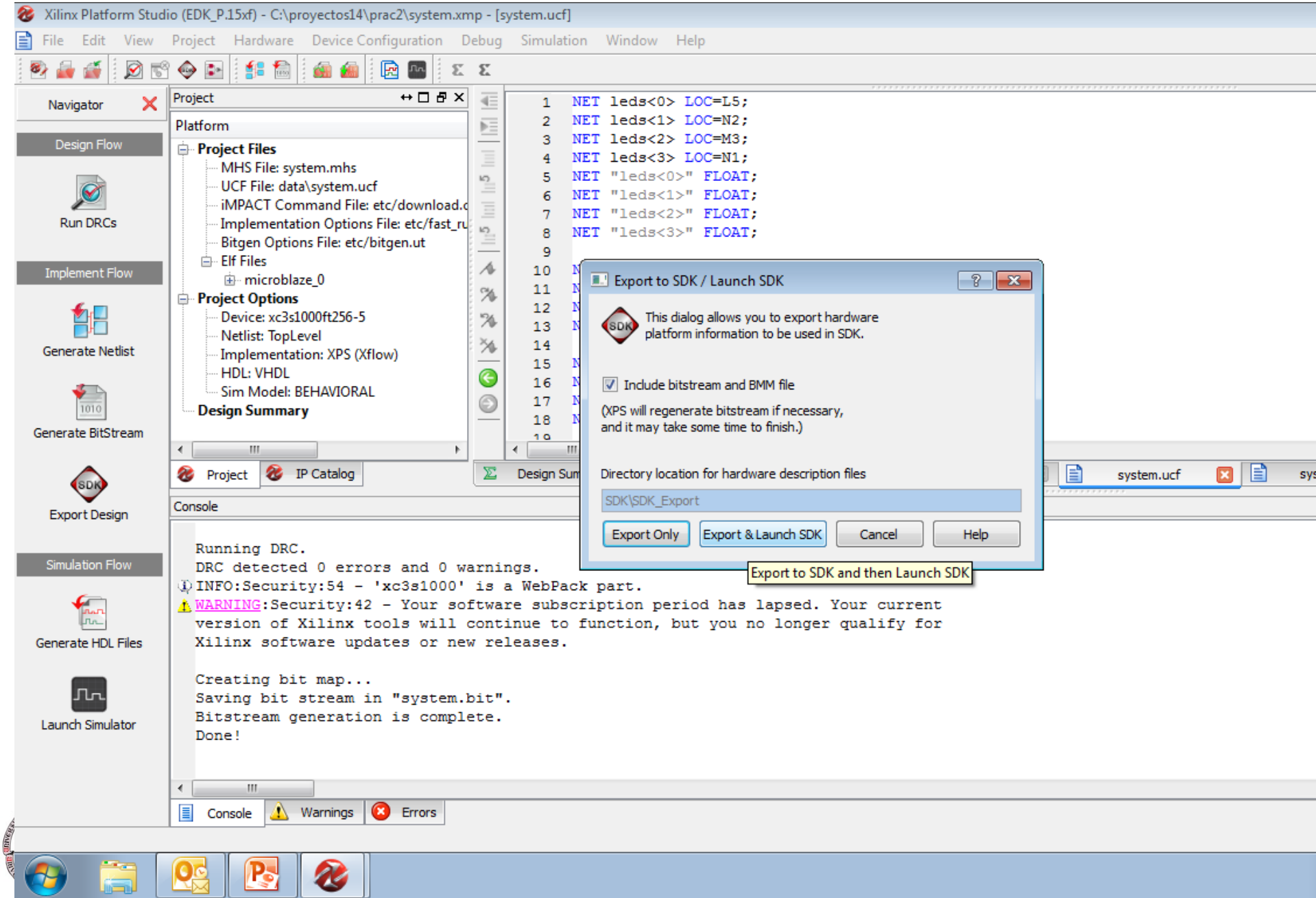
Seleccionar la pestaña“Graphical Design View”

Generamos el bitstream



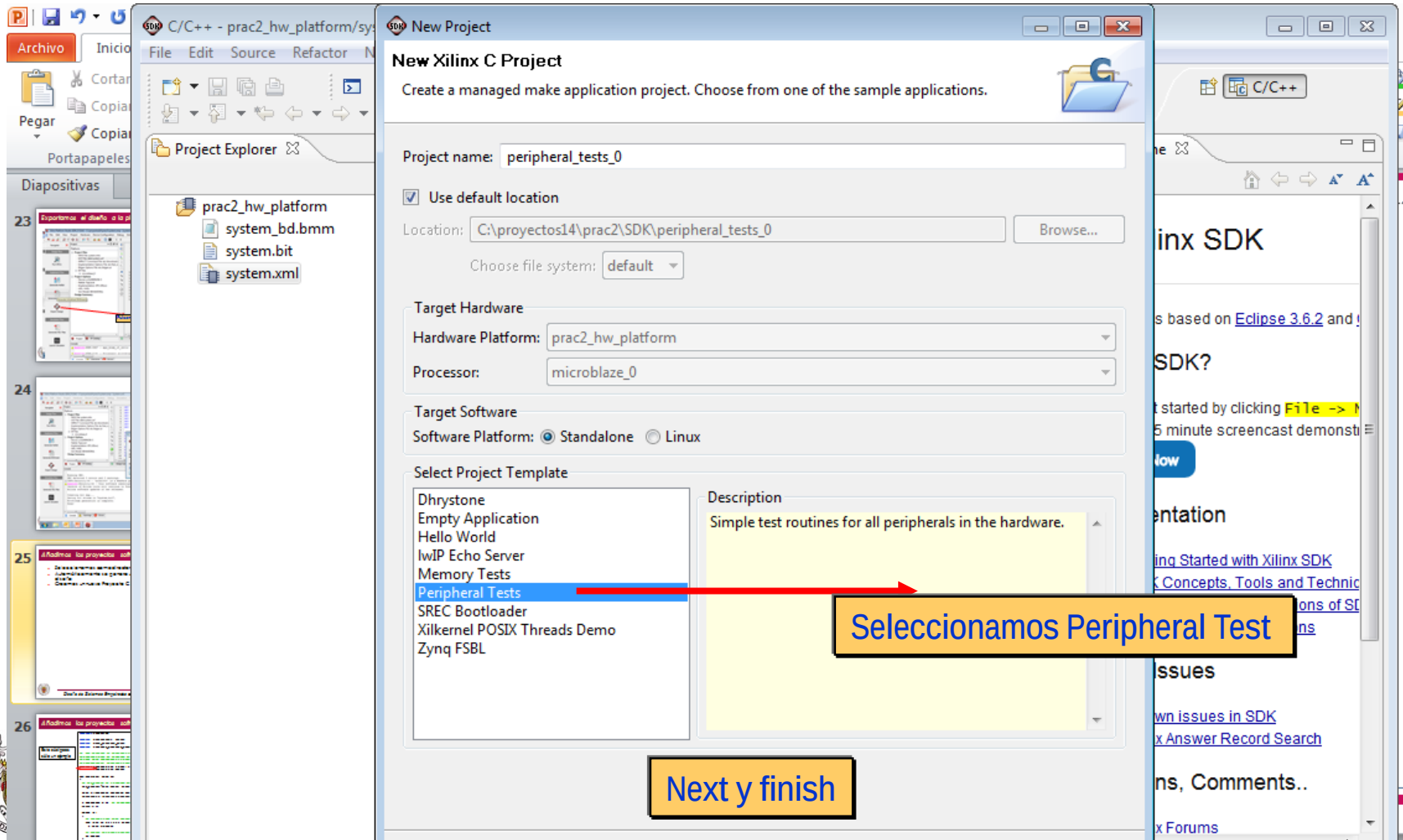
Exportamos el diseño a la plataforma software (SDK)




Xilinx Platform Studio (EDK_P.15xf) - C:\proyectos14\prac2\system.xmp - [system.ucf]
File Edit View Project Hardware Device Configuration Debug Simulation Window Help
Navigator
Design Flow
Run DRCs
Implement Flow
Generate Netlist
Generate BitStream
Export Design
Simulation Flow
Generate HDL Files
Launch Simulator
Project
Platform
Project Files
MHS File: system.mhs
UCF File: data\system.ucf
iMPACT Command File: etc/download.c
Implementation Options File: etc/fast_ru
Bitgen Options File: etc/bitgen.ut
Elf Files
microblaze_0
Project Options
Device: xc3s1000ft256-5
Netlist: TopLevel
Implementation: XPS (Xflow)
HDL: VHDL
Sim Model: BEHAVIORAL
Design Summary
Project
IP Catalog
1 NET leds<0> LOC=L5;
2 NET leds<1> LOC=N2;
3 NET leds<2> LOC=M3;
4 NET leds<3> LOC=N1;
5 NET "leds<0>" FLOAT;
6 NET "leds<1>" FLOAT;
7 NET "leds<2>" FLOAT;
8 NET "leds<3>" FLOAT;
Export to SDK / Launch SDK
This dialog allows you to export hardware platform information to be used in SDK.
Include bitstream and BMM file
(XPS will regenerate bitstream if necessary, and it may take some time to finish.)
Directory location for hardware description files
SDK\SDK_Export
Export Only
Export & Launch SDK
Cancel
Help
Export to SDK and then Launch SDK
system.ucf
Console
Running DRC.
DRC detected 0 errors and 0 warnings.
INFO:Security:54 - 'xc3s1000' is a WebPack part.
WARNING:Security:42 - Your software subscription period has lapsed. Your current version of Xilinx tools will continue to function, but you no longer qualify for Xilinx software updates or new releases.
Creating bit map...
Saving bit stream in "system.bit".
Bitstream generation is complete.
Done!
Console
Warnings
Errors

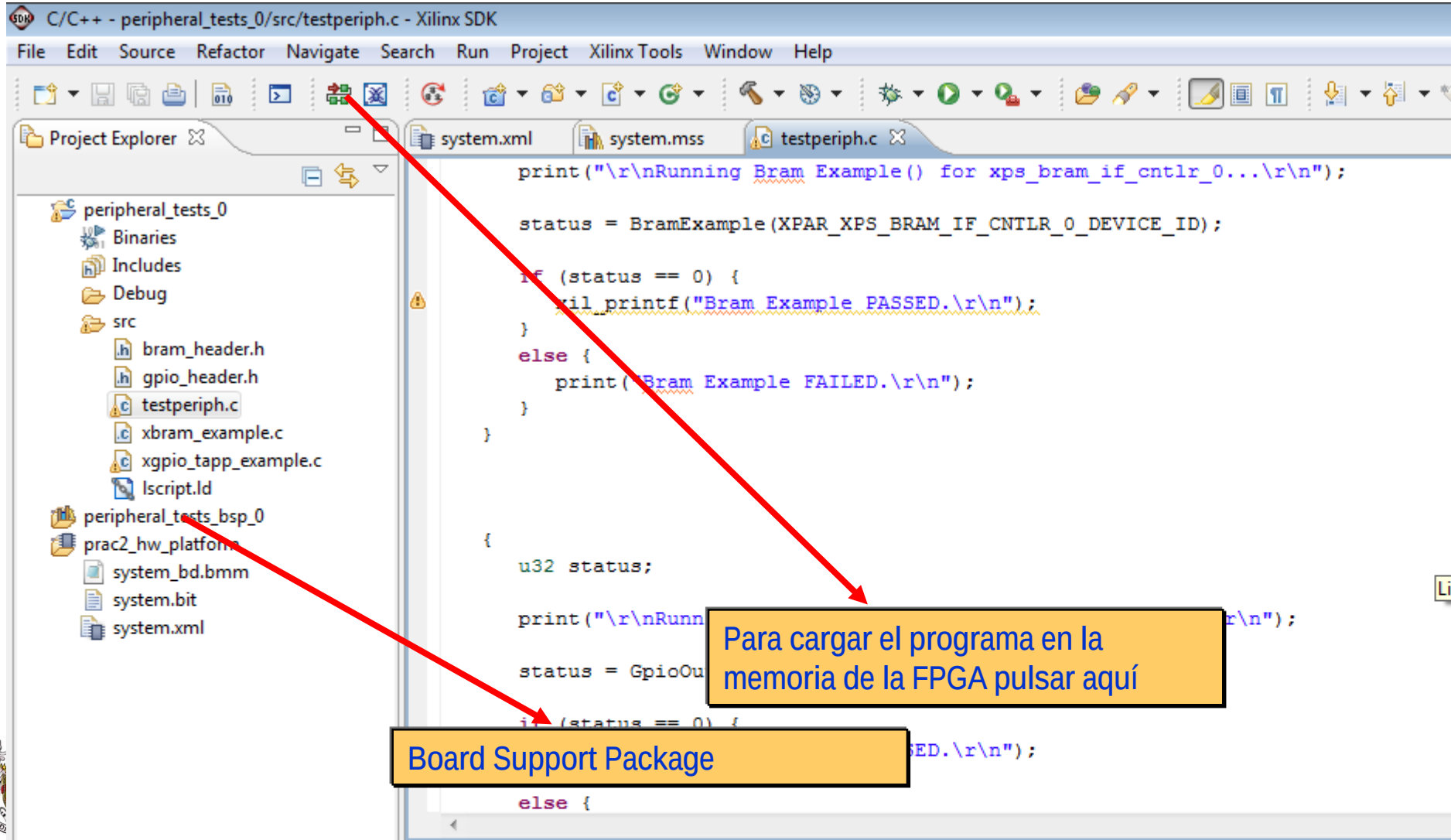
Añadimos los proyectos software

- Seleccionamos como directorio el de trabajo\SDK
- Automáticamente se genera una plataforma hw basada en nuestro diseño
- Creamos un nuevo Proyecto C (File->New->C xilinx Porject)



Proyecto Software

- Dentro de SRC en el fichero testperiph.c tenemos un programa que nos permite probar la uart, la memoria y los gpios



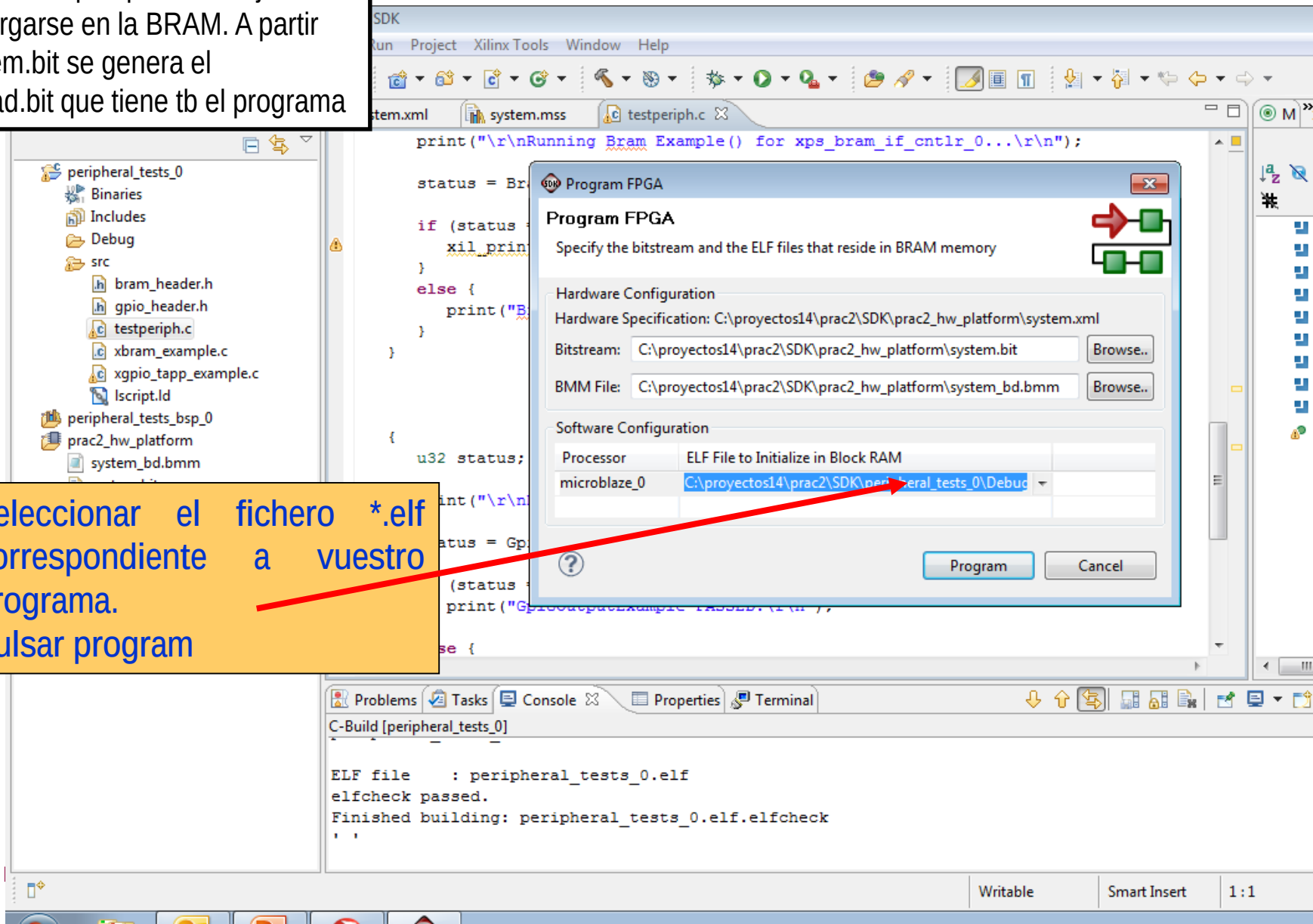
Podemos generar nuestro propio proyectos software

Este código es sólo un ejemplo

```
#include <stdio.h>

#define  REPETITIONS  100000
#define  DATA_REGISTER_GPIO0      0x81400000
#define  TRISTATE_REGISTER_GPIO0  0x81400004

//  All pointers to addresses that correspond to a peripheral instead of a memory
// position must be declared as "volatile". Otherwise, the compiler could assume
// that they correspond to memory positions and store them in registers during
// optimization, therefore precluding actual access to the peripherals.
volatile unsigned int * dato, * triestado;
volatile unsigned int valor;    // This one could be not volatile.

int main(int, char **)
{
  // xil_printf is a reduced version of printf that produces smaller binaries
  xil_printf("Hola mundo, desde el MicroBlaze en la Spartan-3!!!\n\r");

  dato = (volatile unsigned int *)DATA_REGISTER_GPIO0;
  triestado = (volatile unsigned int *)TRISTATE_REGISTER_GPIO0;

  * triestado = 0;  // Configures the port as output (input would have 1s on each bit)
  valor = 0;

  while (1)
  {
    // Consume some time accessing the register
    for (int ii = 0; ii < REPETITIONS; ++ ii)
      * dato = valor;

    // Update the value written into the register
    ++ valor;
  }

  return 0;  // Never reached!
}
```

Configuración de la aplicación para que resida en BRAM

La aplicación que queremos ejecutar debe cargarse en la BRAM. A partir del sistem.bit se genera el download.bit que tiene tb el programa



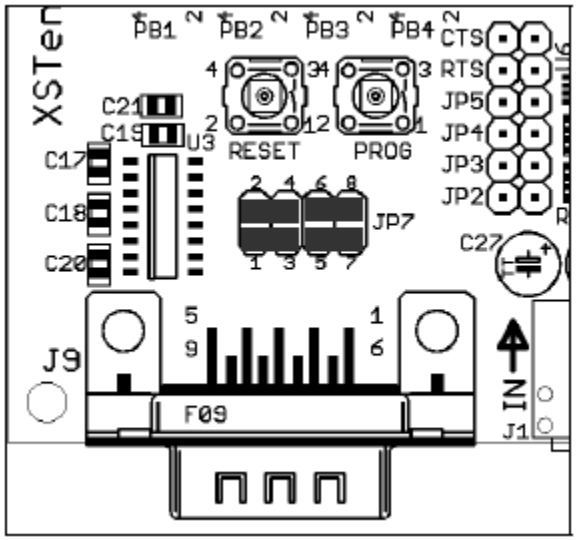
Seleccionar el fichero *.elf correspondiente a vuestro programa.
Pulsar program

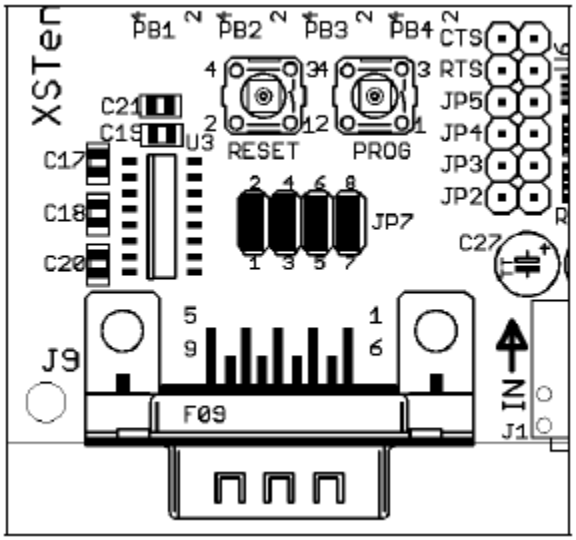
Conexión del sitema al PC y alimentación

- Conectar el cable serie a la FPGA y al PC
- Abrir una instancia de la aplicación Hyperterminal y configurarla con los mismos valores que la FPGA
  - ✓ Baudrate: 9600
  - ✓ Data bits: 8
  - ✓ Parity: NO
  - ✓ Stop bits: 1
  - ✓ Flow control: NONE
- Cargar el bitstream

Configuración de la placa para utilizar el puerto serie

- Dependiendo del tipo de cable serie utilizado, los jumpers de configuración de la UART (JP7) deben colocarse de la siguiente forma



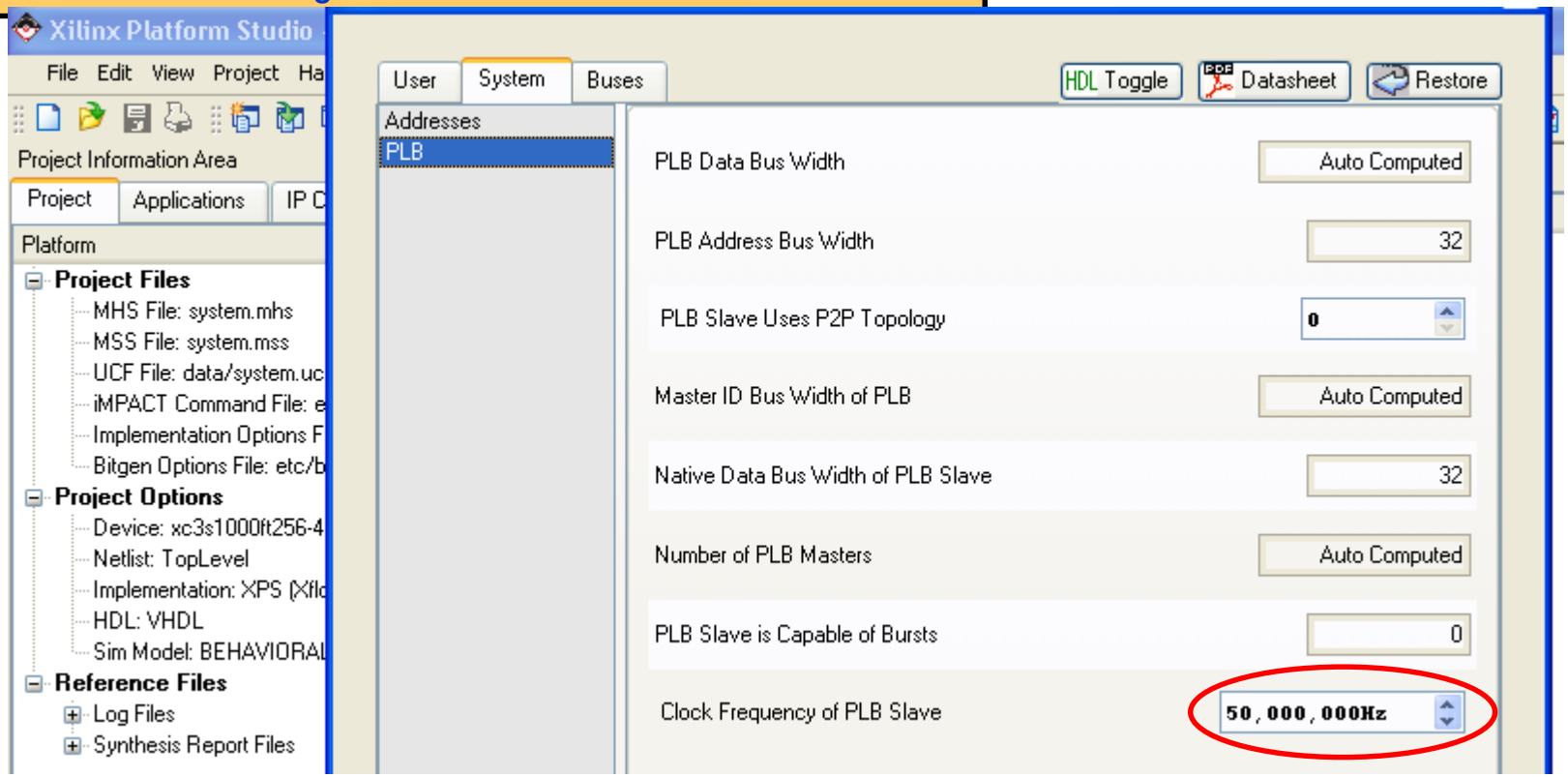


El juper JP7 debe colocarse como el lado izdo de la figura si se usa un cable serie directo.
Se utilizará la configuración de la derecha si se usa un cable null-modem

Ojo

- No olvidar actualizar la frecuencia del bus para la UART.
- Recordar que los botones sin pulsar generan un "1"

Seleccionar el módulo UART y con el botón derecho seleccionar → Configure IP



Xparameters.h

(peripheral-test-bsp-> microblaze->include)

```
#define STDIN_BASEADDRESS 0x84000000
#define STDOUT_BASEADDRESS 0x84000000
/******************************************************************/
/* Definitions for driver UARTLITE */
#define XPAR_XUARTLITE_NUM_INSTANCES 1
/* Definitions for peripheral XPS_UARTLITE_0 */
#define XPAR_XPS_UARTLITE_0_BASEADDR 0x84000000
#define XPAR_XPS_UARTLITE_0_HIGHADDR 0x8400FFFF
#define XPAR_XPS_UARTLITE_0_DEVICE_ID 0
#define XPAR_XPS_UARTLITE_0_BAUDRATE 9600
#define XPAR_XPS_UARTLITE_0_USE_PARITY 0
#define XPAR_XPS_UARTLITE_0_ODD_PARITY 1
#define XPAR_XPS_UARTLITE_0_DATA_BITS 8
```

Board Support Package: Drivers de dispositivo (GPIO)

Xparameters.h

```
/* Definitions for driver GPIO */
#define XPAR_XGPIO_NUM_INSTANCES 2

/* Definitions for peripheral XPS_GPIO_0 */
#define XPAR_XPS_GPIO_0_BASEADDR 0x81420000
#define XPAR_XPS_GPIO_0_HIGHADDR 0x8142FFFF
#define XPAR_XPS_GPIO_0_DEVICE_ID 0
#define XPAR_XPS_GPIO_0_INTERRUPT_PRESENT 0
#define XPAR_XPS_GPIO_0_IS_DUAL 0

/* Definitions for peripheral XPS_GPIO_1 */
#define XPAR_XPS_GPIO_1_BASEADDR 0x81400000
#define XPAR_XPS_GPIO_1_HIGHADDR 0x8140FFFF
#define XPAR_XPS_GPIO_1_DEVICE_ID 1
#define XPAR_XPS_GPIO_1_INTERRUPT_PRESENT 0
#define XPAR_XPS_GPIO_1_IS_DUAL 0
```

xgpio.h

peripheral-test-bsp-> microblaze->include

/*
* This file contains the software API definition of the Xilinx General Purpose
- I/O (XGpio) device driver.*/

```
typedef struct {
u32 BaseAddress;/* Device base address */
u32 IsReady;/* Device is initialized and ready */
int InterruptPresent;/* Are interrupts supported in h/w */
int IsDual;/* Are 2 channels supported in h/w */
} XGpio;
```

xgpio.c

peripheral-test-bsp-> microblaze-> libsource-> gpioxxx->source

```
/* The implementation of the XGpio driver's basic functionality. See xgpio.h
for more information about the driver.*/


/*
 * API Basic functions implemented in xgpio.c
*/
void XGpio_SetDataDirection(XGpio *InstancePtr, unsigned Channel,
  u32 DirectionMask);
u32 XGpio_DiscreteRead(XGpio *InstancePtr, unsigned Channel);
void XGpio_DiscreteWrite(XGpio *InstancePtr, unsigned Channel, u32 Mask);
```

xgpio.c

```
void XGpio_SetDataDirection(XGpio * InstancePtr, unsigned Channel,
    u32 DirectionMask)
{
Xil_AssertVoid(InstancePtr != NULL);
Xil_AssertVoid(InstancePtr->IsReady == XIL_COMPONENT_IS_READY);
Xil_AssertVoid((Channel == 1) ||
    ((Channel == 2) && (InstancePtr->IsDual == TRUE)));

XGpio_WriteReg(InstancePtr->BaseAddress,
((Channel - 1) * XGPIO_CHAN_OFFSET) + XGPIO_TRI_OFFSET,
DirectionMask);
}
```

xgpio.c

```
u32 XGpio_DiscreteRead(XGpio * InstancePtr, unsigned Channel)
{
    Xil_AssertNonvoid(InstancePtr != NULL);
    Xil_AssertNonvoid(InstancePtr->IsReady == XIL_COMPONENT_IS_READY);
    Xil_AssertNonvoid((Channel == 1) ||
    ((Channel == 2) && (InstancePtr->IsDual == TRUE)));

    return XGpio_ReadReg(InstancePtr->BaseAddress,
        ((Channel - 1) * XGPIO_CHAN_OFFSET) +
        XGPIO_DATA_OFFSET);
}
```

Board Support Package: Drivers de dispositivo (GPIO)

xgpio.c

```
void XGpio_DiscreteWrite(XGpio * InstancePtr, unsigned Channel, u32 Data)
{
Xil_AssertVoid(InstancePtr != NULL);
Xil_AssertVoid(InstancePtr->IsReady == XIL_COMPONENT_IS_READY);
Xil_AssertVoid((Channel == 1) ||
    ((Channel == 2) && (InstancePtr->IsDual == TRUE)));

XGpio_WriteReg(InstancePtr->BaseAddress,
((Channel - 1) * XGPIO_CHAN_OFFSET) + XGPIO_DATA_OFFSET, Data);
}
```

xgpio_tapp_example.c

```
/* This file contains a example for using GPIO hardware and driver.
* This example assumes that there is a UART Device or STDIO Device in the
* hardware system.
*/
XGpio GpioOutput; /* The driver instance for GPIO Device configured as O/P */
XGpio GpioInput;  /* The driver instance for GPIO Device configured as I/P */
```

Board Support Package: Drivers de dispositivo (GPIO)

xgpio_tapp_example.c

```
int GpioOutputExample(u16 DeviceId, u32 GpioWidth)
{
/***** Definiciones******/

/*
 * Initialize the GPIO driver so that it's ready to use,
 * specify the device ID that is generated in xparameters.h
 */
XGpio GpioOutput;                          XPAR_XPS_GPIO_0_DEVICE_ID

 Status = XGpio_Initialize(&GpioOutput, DeviceId);
 if (Status != XST_SUCCESS)  {
  return XST_FAILURE;
 }
```

xgpio_tapp_example.c

```
/*
 * Set the direction for all signals to be outputs
 */
XGpio_SetDataDirection(&GpioOutput, LED_CHANNEL, 0x0);

/*
 * Set the GPIO outputs to low
 */
XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL, 0x0);

* Set the GPIO Output to Valor
 */
XGpio_DiscreteWrite(&GpioOutput, LED_CHANNEL, Valor);
```

Board Support Package: Drivers de dispositivo (GPIO)

xgpio_tapp_example.c

```
int GpioInputExample(u16 DeviceId, u32 *DataRead)
{
 int Status;
 /*
  * Initialize the GPIO driver so that it's ready to use,
  * specify the device ID that is generated in xparameters.h
  */
 Status = XGpio_Initialize(&GpioInput, DeviceId);
 if (Status != XST_SUCCESS) {
  return XST_FAILURE;
 }
 /*
  * Set the direction for all signals to be inputs
  */
 XGpio_SetDataDirection(&GpioInput, CHANNEL, 0xFFFFFFFF);
 /*
  * Read the state of the data so that it can be  verified
  */
 *DataRead = XGpio_DiscreteRead(&GpioInput, CHANNEL);
 return XST_SUCCESS;

}
```