Universidad
Carlos III de Madrid

Instrumentación Electrónica con Microprocesador II: Procesadores Avanzados

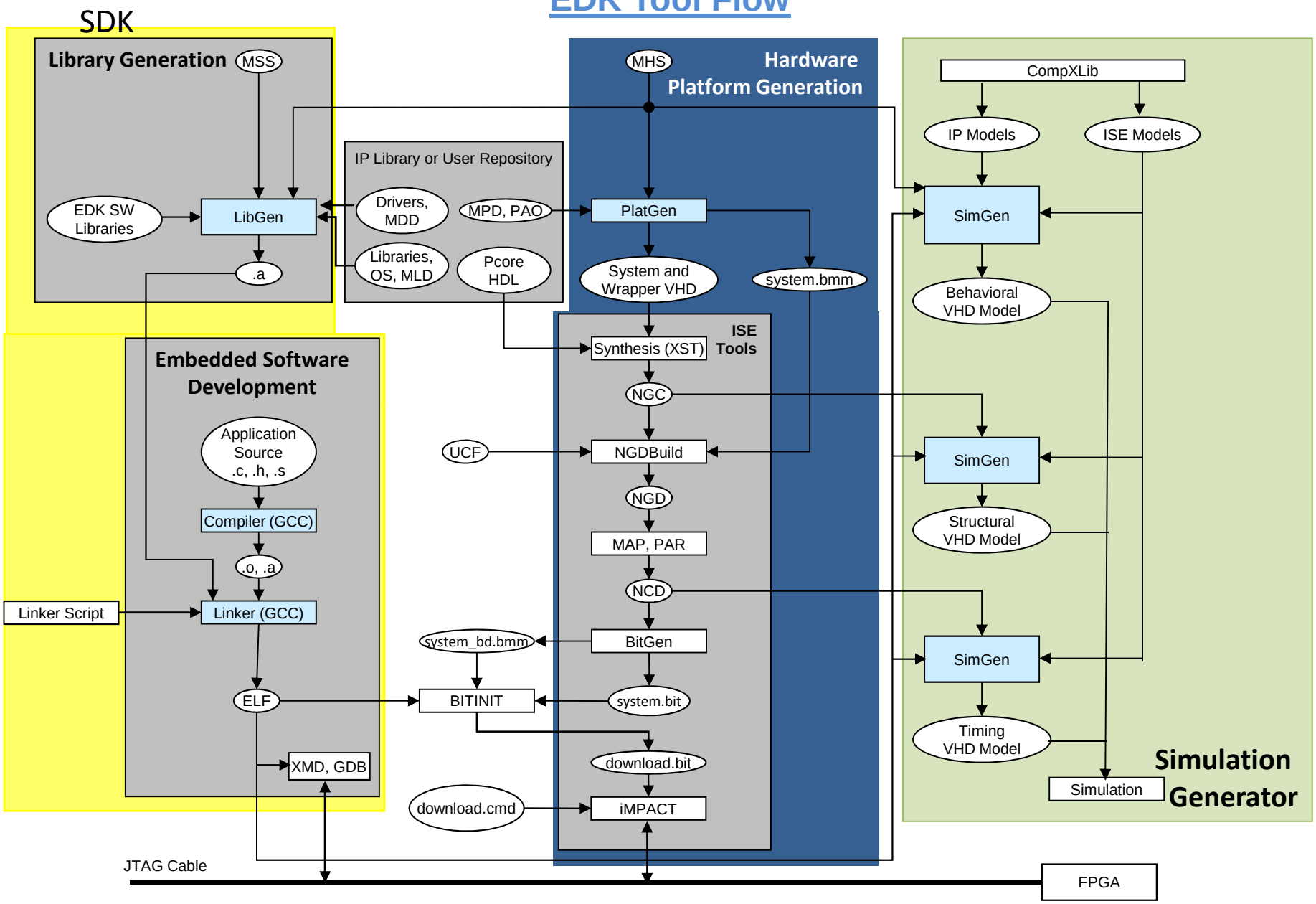
Microprocesadores empotrados en FPGAs
MicroBlaze™. Programación en C

Marta Portela García

SDK (SOFTWARE DEVELOPMENT ENVIRONMENT)

- SDK facilita el desarrollo de las aplicaciones software para los sistemas empotrados con FPGAs de Xilinx.
  - Para desarrollar el software necesario para programar el sistema hardware desarrollado con XPS (los periféricos y el procesador)
- Basado en Eclipse: entorno de desarrollo de código abierto.
- Incluye las siguientes características:
  - Editor de código enriquecido para C/C++ y un entorno para su compilación.
  - Gestión de proyectos.
  - Generación automática de ficheros *makefile* y configuración para la construcción de aplicaciones.
  - Entorno para la depuración y el análisis de código.
  - Control de versiones.

EDK Tool Flow

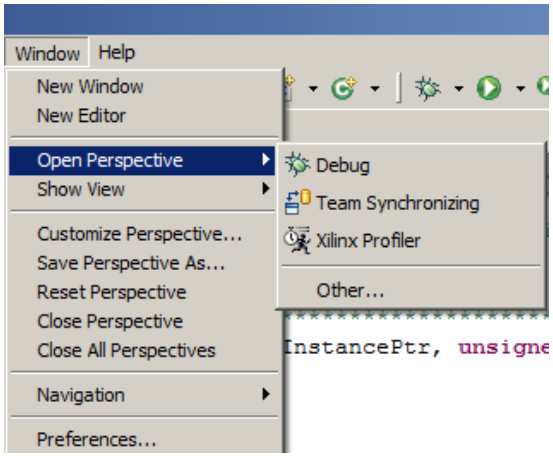


SDK: ELEMENTOS

- Plataforma Sofware
  - Colección de librerías y drivers que forman la capa más baja del software
  - 2 tipos:
    - Standalone
    - Xilkernel.- Nucleo ligero que proporciona servicios como gestión de hilos, planificación de tareas,…
  - Un mismo proyecto SDK puede contener varias plataformas SW
- Proyecto Software
  - Aplicación de usuario. Una plataforma SW puede contener varios proyectos SW. Contiene el código fuente (ficheros .c, .h)

SDK: ENTORNO

- Espacio de trabajo (workspace)
  - Directorio utilizado para guardar toda la información relacionada con un proyecto en SDK
  - Es necesario un workspace por cada diseño hardware
  - Para copiar o mover workspaces de un directorio a otro hay que usar las funciones de importar/esportar de SDK.
- Perspectivas
  - Conjunto de herramientas de desarrollo relacionada con una tarea concreta dentro del desarrollo del SW.
  - Cada perspectiva está formada por un conjunto distinto de ventanas. Cada ventana en una perspectiva se denomina vista.

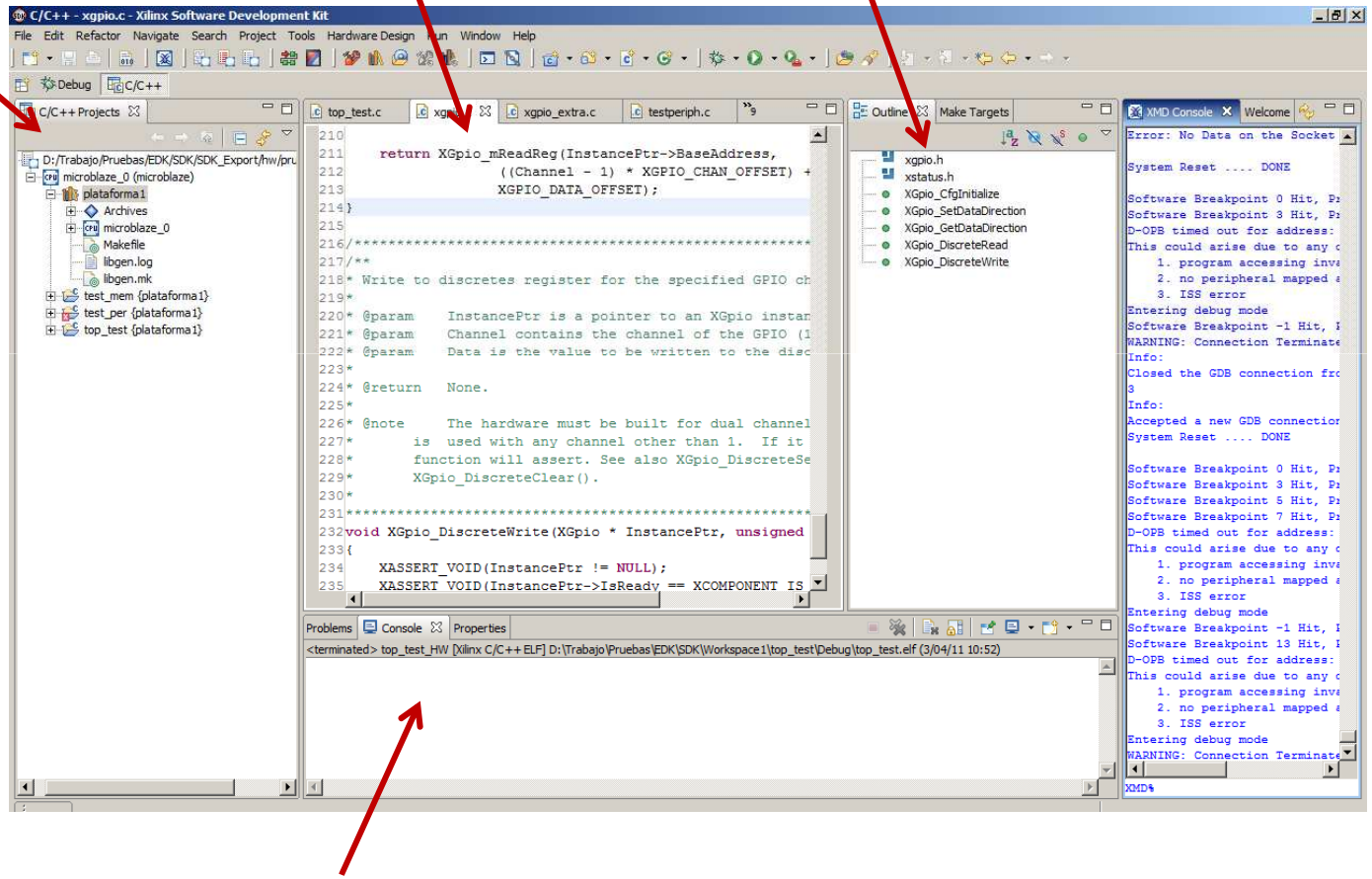


SDK: ENTORNO. PERSPECTIVA C/C++

Proyecto

Editor C/C++

Estructura



Salida de la ejecución, propiedades y problemas encontrados

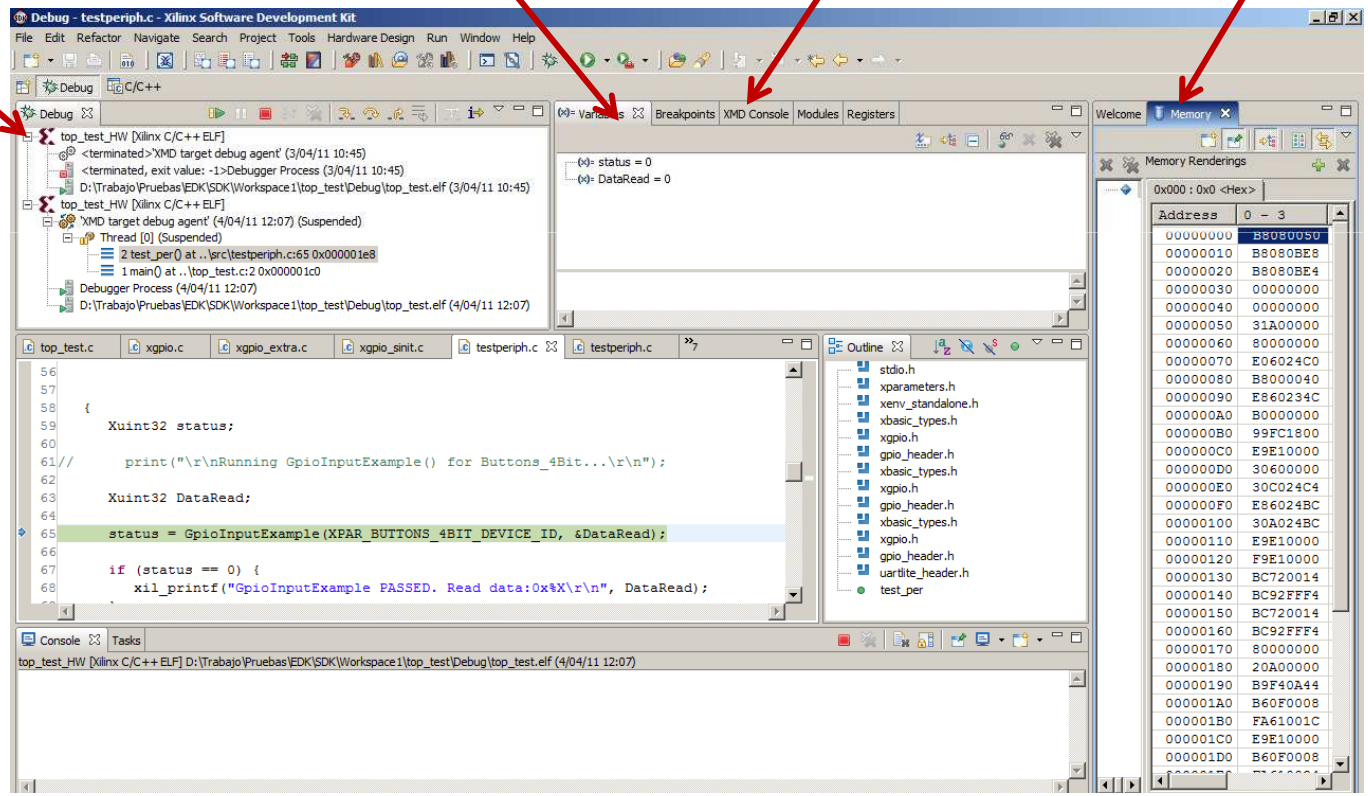
SDK: ENTORNO. PERSPECTIVA DE DEPURACIÓN

Módulos a depurar

Información del uP: variables, registros,…

Consola XMD

Contenido de la memoria

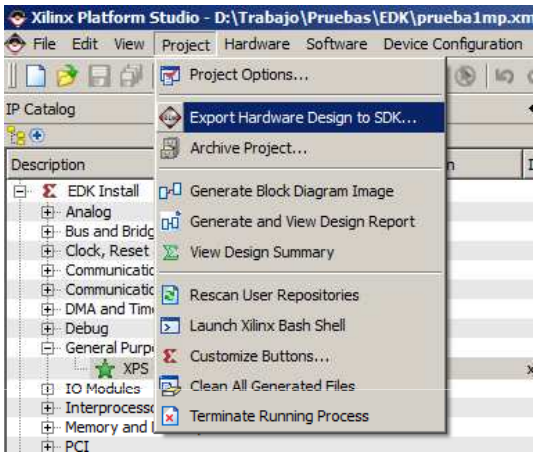


SDK: FLUJO DE DESARROLLO DE UNA APLICACIÓN SW

- Diseño del sistema HW con XPS.
- Exportar el diseño HW a SDK (formato XML)
- Abrir SDK y especificar el fichero con la descripción del sistema HW que se exportó desde XPS.
- Crear la plataforma SW que contenga una biblioteca de funciones a usar por la aplicación.
- Desarrollar la aplicación SW (incluyendo los drivers).
- Se puede modificar el mapa de memoria de la aplicación utilizando las herramientas de generación de scripts para el lincador (*linker script*).
- Desde SDK se puede descargar y ejecutar el software en la FPGA, así como depurar y analizar su rendimiento (profiling).

SDK: CREAR UN PROYECTO

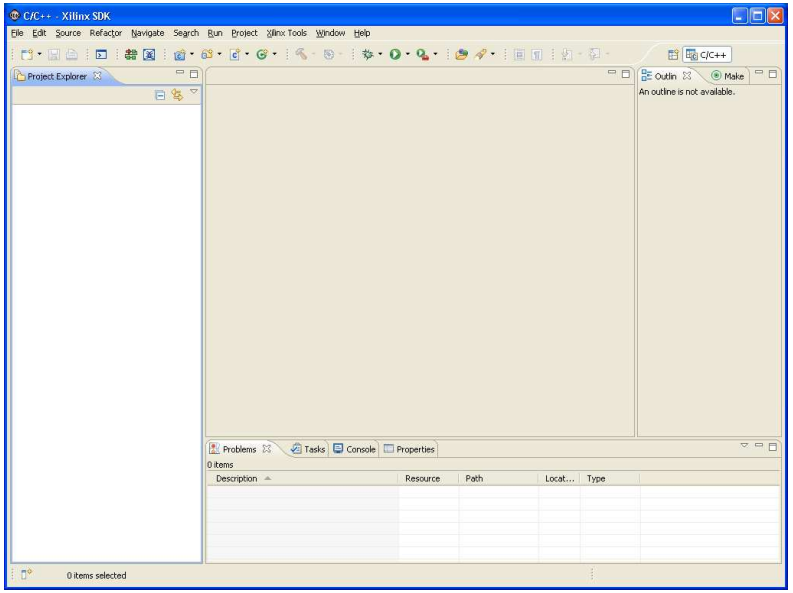
1 Exportar la plataforma HW desde XPS a SDK



2 Seleccionar la plataforma HW

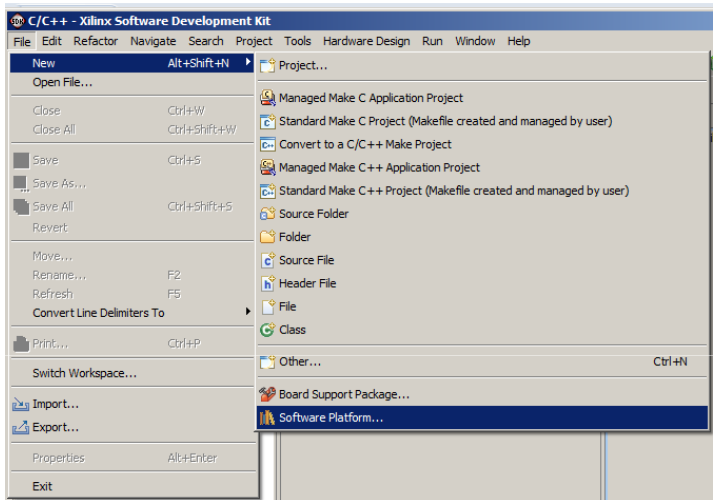
- Al abrir SDK lo primero que hay que hacer es indicar el Workspace
- A continuación es necesario seleccionar la plataforma HW

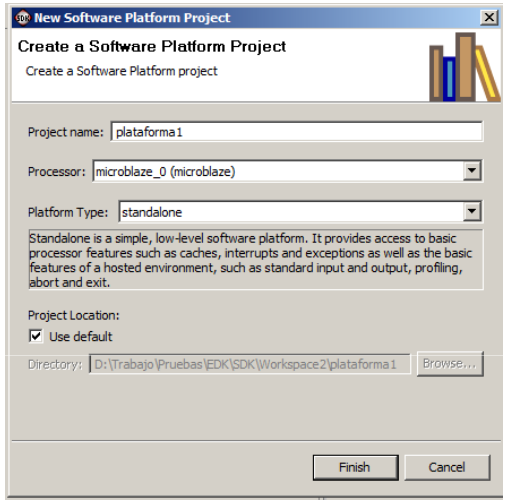
(../SDK_Export/HW/nameHW.xml)

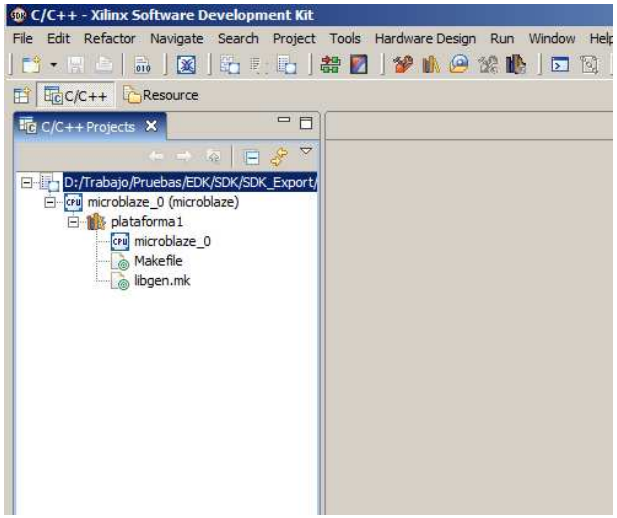


SDK: CREAR UN PROYECTO

3 Crear la plataforma SW

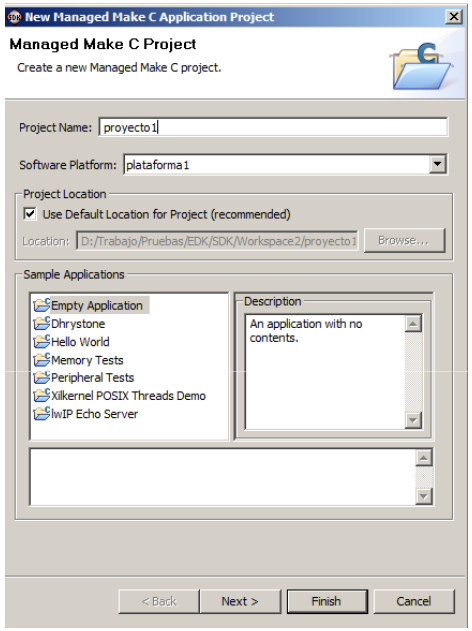






SDK: CREAR UN PROYECTO

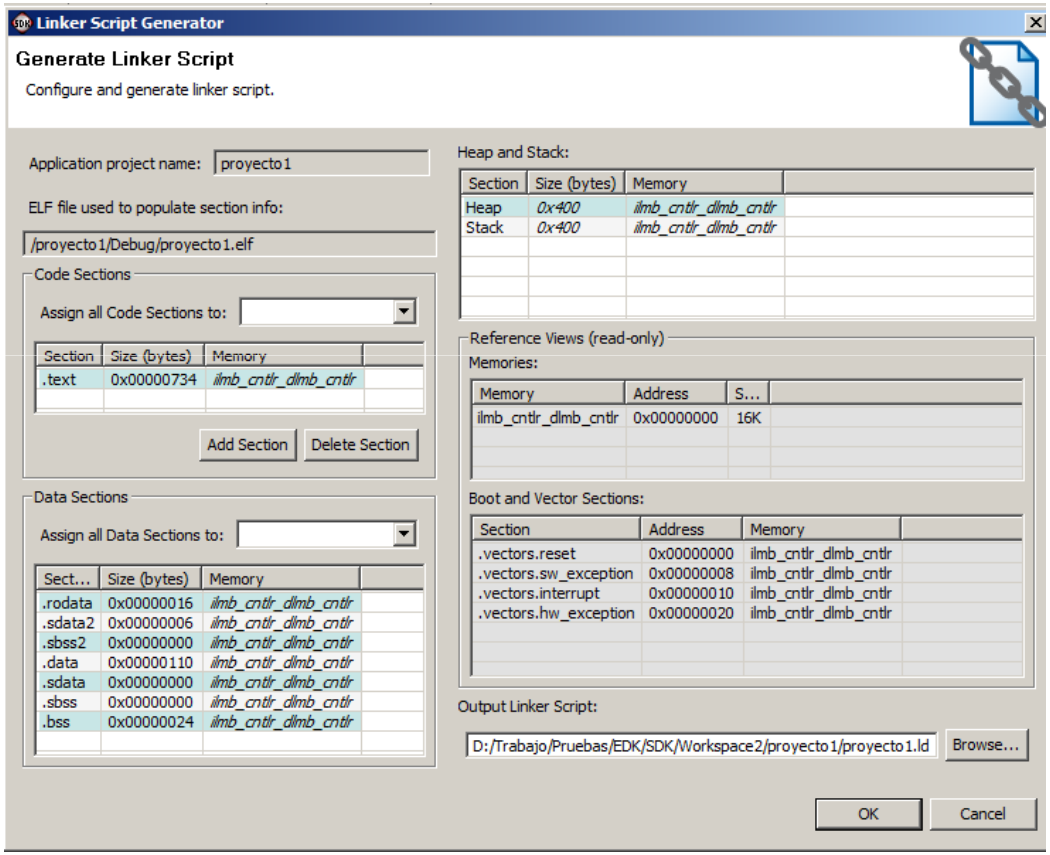
4 Crear el proyecto SW



- Importar un fichero: File → Import
  - Los ficheros importados se copian en el directorio de la aplicación
  - Se añaden automáticamente al fichero makefile
- Cada aplicación genera su propio fichero .elf

SDK: CREAR UN PROYECTO

5 Modificar o configurar el mapa de memoria: *Linker script*



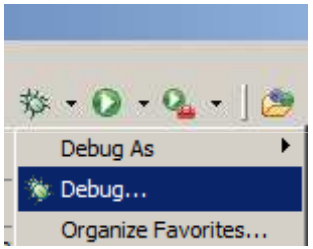
- Tools → Generate Linker script
- Fichero nombre_proyecto.ld
- Especifica dónde se almacenan código y datos dentro del sistema HW especificado.

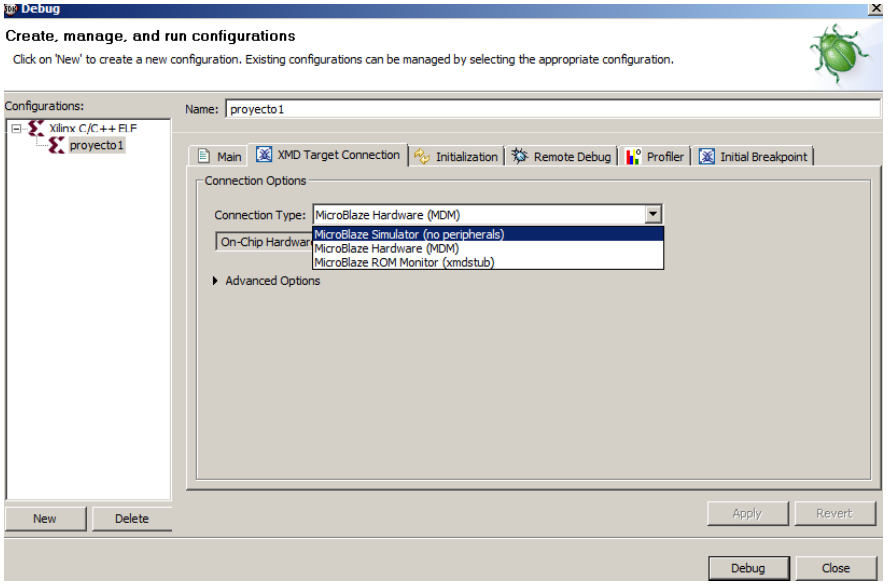
SDK: CREAR UN PROYECTO

6 Pruebas: Simulación y en HW

XMD

- Es necesario abrir una consola de la herramienta XMD
- Simulación



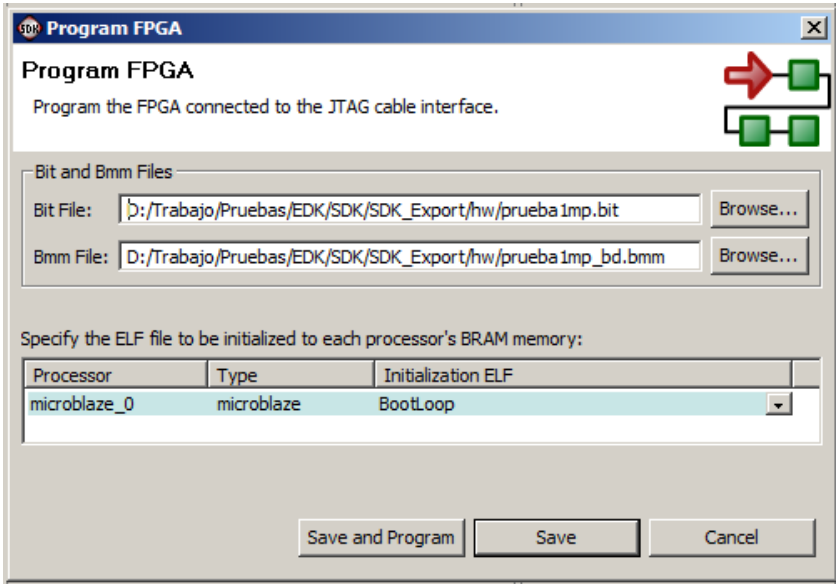


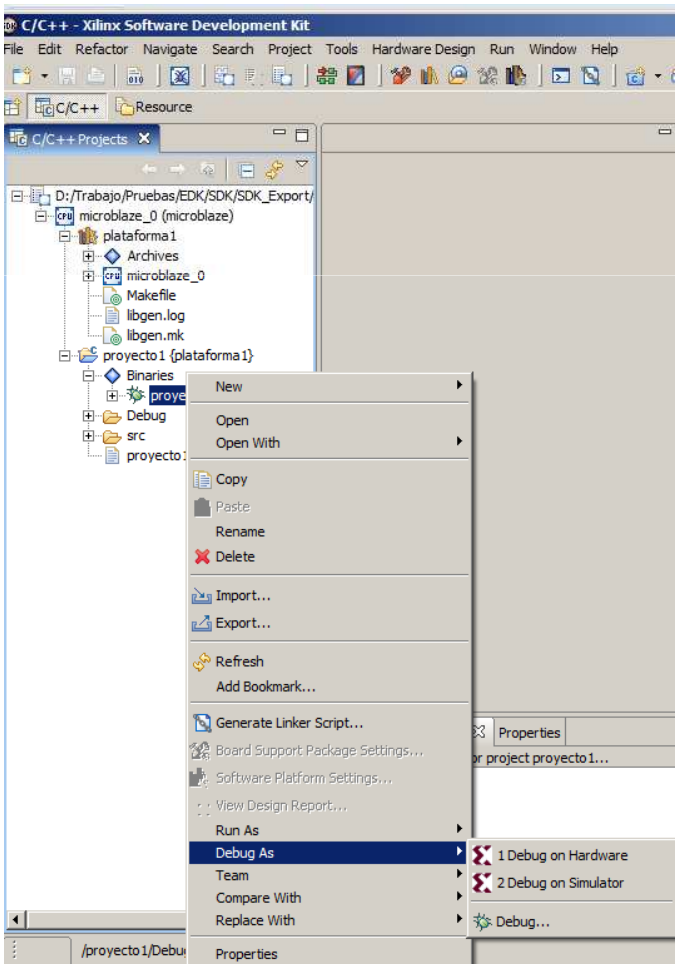
SDK: CREAR UN PROYECTO

6 Pruebas: Simulación y en HW

XMD

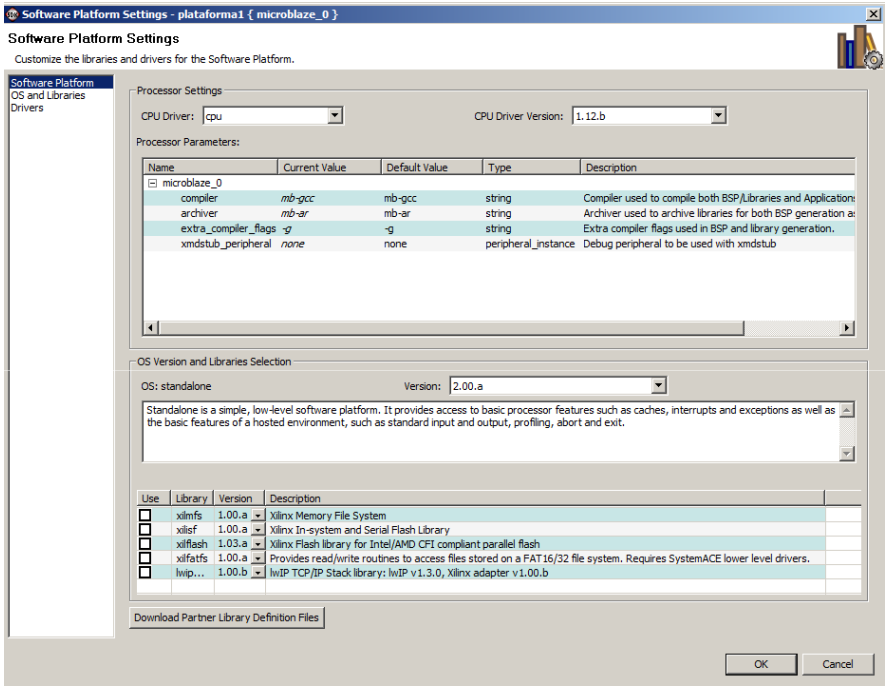
- Es necesario abrir una consola de la herramienta XMD
- Depuración Hardware
  - Tools → Program FPGA





SDK: PROPIEDADES DE UN PROYECTO

- Propiedades de la plataforma SW (Tools → SW Platform Settings)



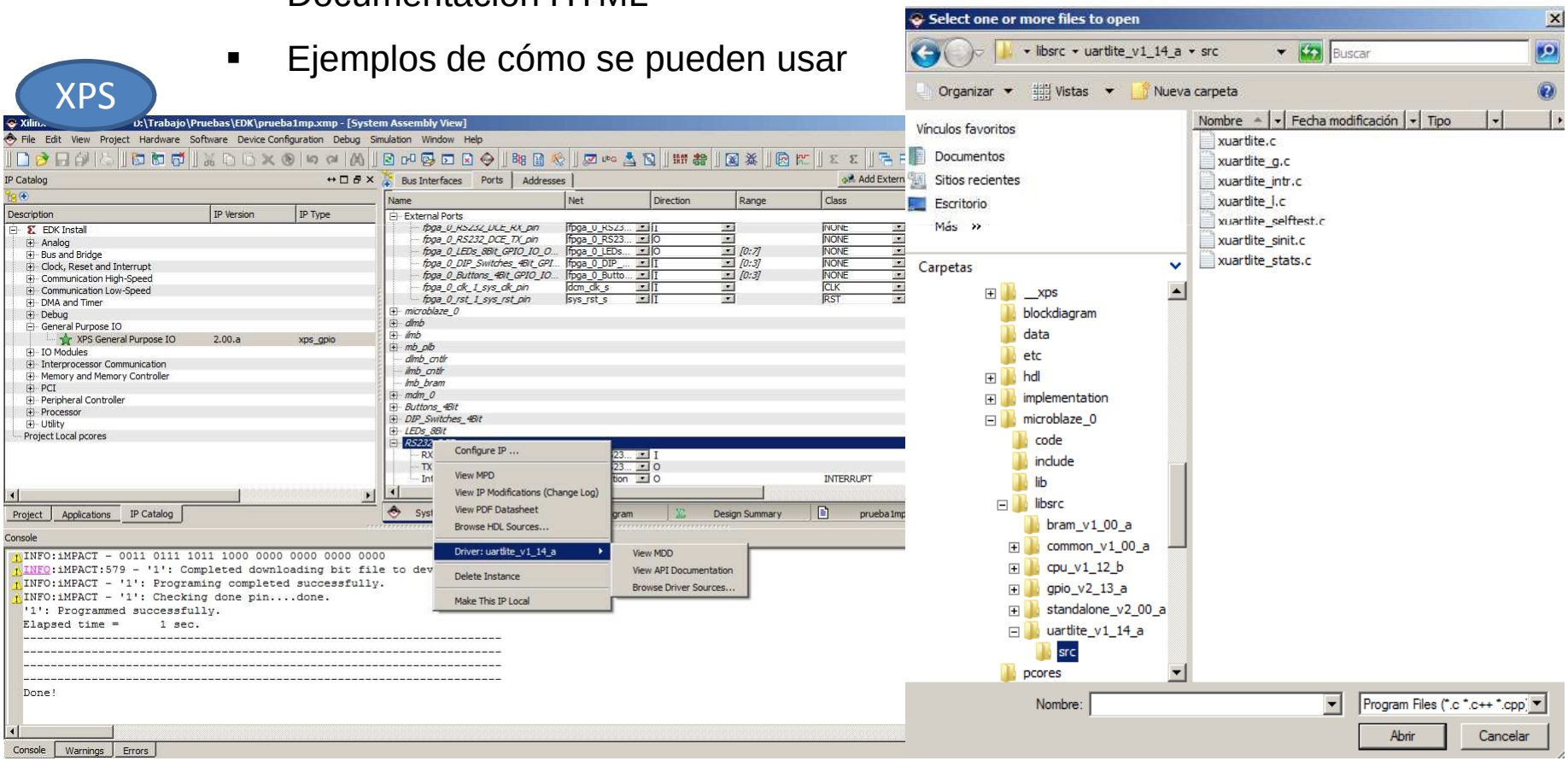
- Propiedades de cada proyecto SW
  - Compilación, depuración, linkado…

BIBLIOGRAFÍA:

• "EDK Concepts, Tools, and Techniques. *Guide to Effective Embedded System Design*" Xilinx → Manual

DRIVERS

- Xilinx proporciona drivers para los periféricos de la plataforma HW.
  - Código fuente
  - Documentación HTML
  - Ejemplos de cómo se pueden usar



LIBRERIAS

- Las librerías se configuran automaticamente con la herramienta Libgen al crear una plataforma HW.
- Xilinx proporciona tres librerias (.\microblaze_0\lib\)
  - **Libm**: para funciones matemáticas
  - **Libc**: para funciones estandard de C (stdio, stdlib, string) compiladas para MicroBlaze.

```
_ansi.h      fastmath.h   machine/     reent.h      stdlib.h     utime.h
_syslist.h   fcntl.h      malloc.h     regdef.h     string.h     utmp.h
ar.h         float.h      math.h       setjmp.h     sys/
assert.h     grp.h        paths.h      signal.h     termios.h
ctype.h      ieeefp.h     process.h    stdarg.h     time.h
dirent.h     limits.h     pthread.h    stddef.h     unctrl.h
errno.h      locale.h     pwd.h        stdio.h      unistd.h
```

  - **Libxil**: contiene drivers y librerías de Xilinx
    - LibXil Driver.- drivers de los dispositivos de Xilinx
    - LibXil MFS.- sistema de ficheros de memoria de Xilinx
    - LibXil flash/isf.- librería de programación de memoria flash paralela/serie

LIBRERIAS

- Además de las función estandard de C, Xilinx ofrece las siguientes funciones para entrada/salida:
  - void **print (char *)**
  - void **putnum (int)**
  - void **xil_printf (const *char ctrl1,...)**

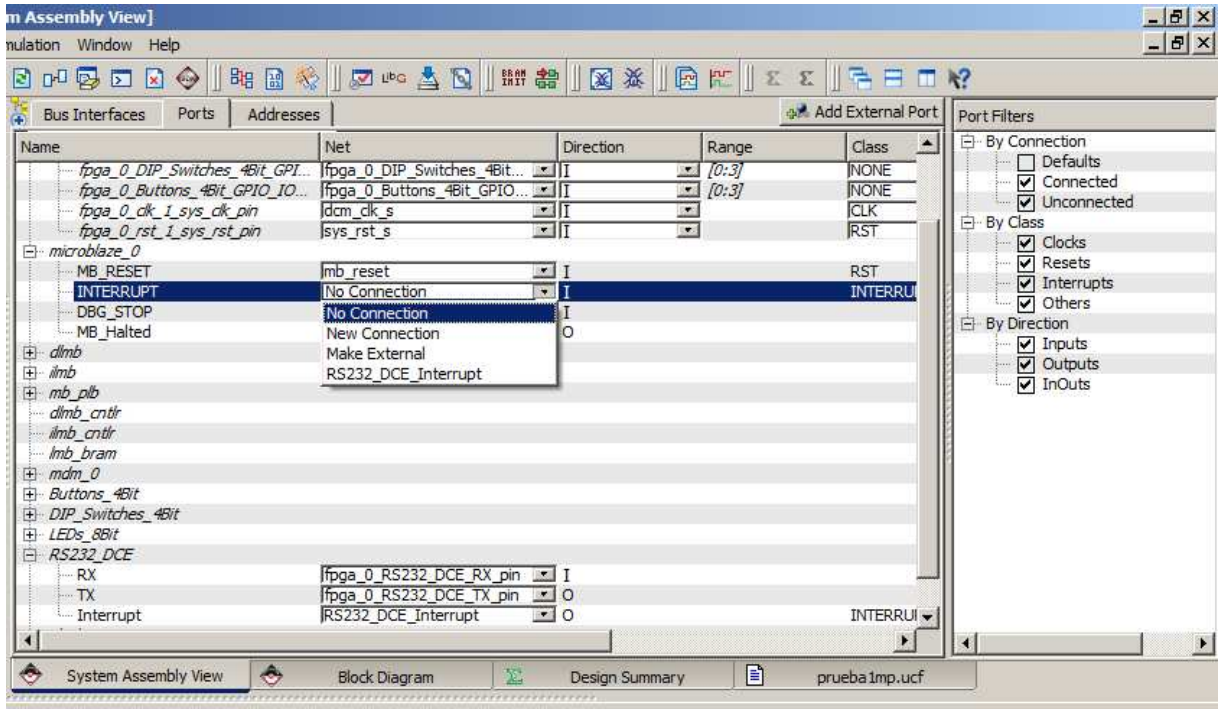
- Las cabeceras de las funciones de xilinx se guardan en .\microblaze_0\include\

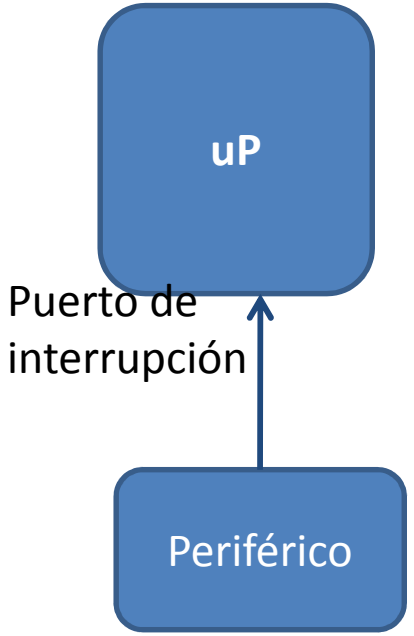
INTERRUPCIONES

- Sólo hay una señal de interrupción
- Se pueden gestionar distintas fuentes de interrupción:
  - Or de las distintas señales de interrupción
  - Utilizar un controlador de interrupciones (periférico)
- mb_interface.h
  - void microblaze_enable_interrupts(void)
  - void microblaze_disable_interrupts(void)
  - void microblaze_register_handler(XInterruptHandler *Handler,* void **DataPtr*)
- Para incluir interrupción en un sistema hay que seguir los siguientes pasos:
  - Realizar las conexiones HW necesarias
  - Desarrollar las funciones HW de atención a la interrupción

GESTIÓN DE INTERRUPCIONES: SIN CONTROLADOR

- El controlador de interrupciones no es necesario cuando sólo hay un dispositivo con interrupción
- La señal de interrupción del periférico debe conectarse a la entrada de interrupción del microprocesador.

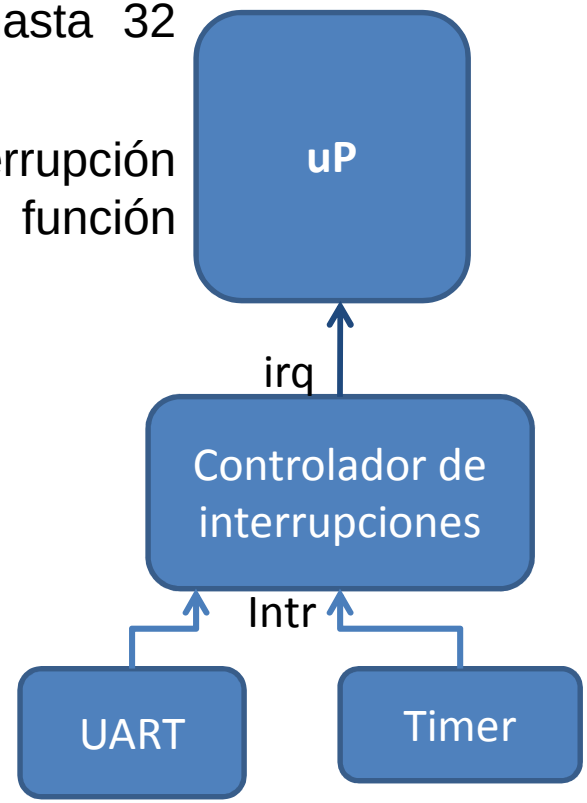




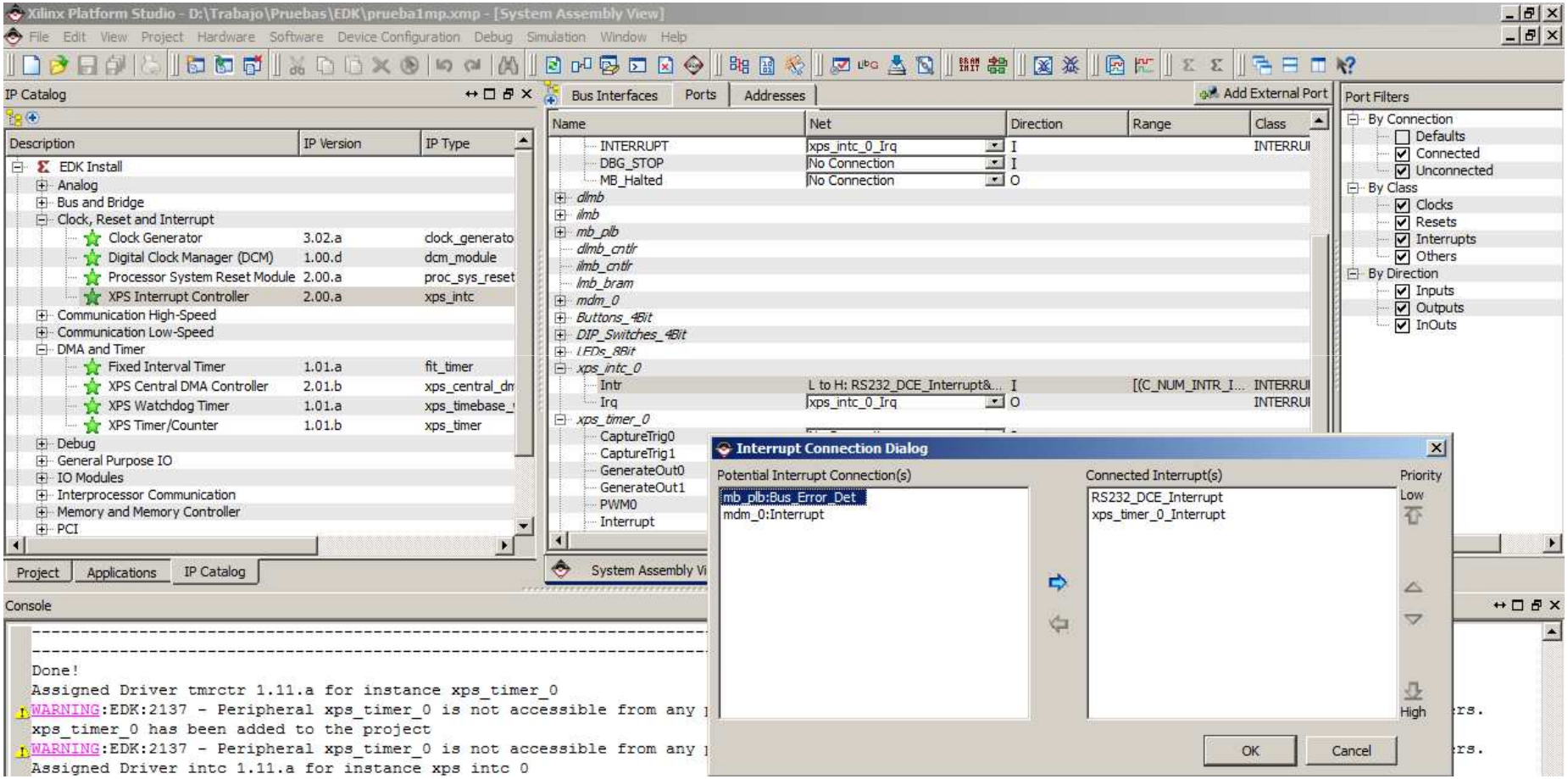
GESTIÓN DE INTERRUPCIONES: CON CONTROLADOR

- El controlador de interrupciones sí es necesario cuando hay más de un dispositivo capaz de producir una interrupción
- La señal de interrupción del periférico debe conectarse a la entrada del controlador de interrupciones

- El MicroBlaze puede gestionar hasta 32 fuentes de interrupción
- Las funciones de atención a la interrupción se escriben en C igual que otra función cualquiera:
  - void nombre(void *)



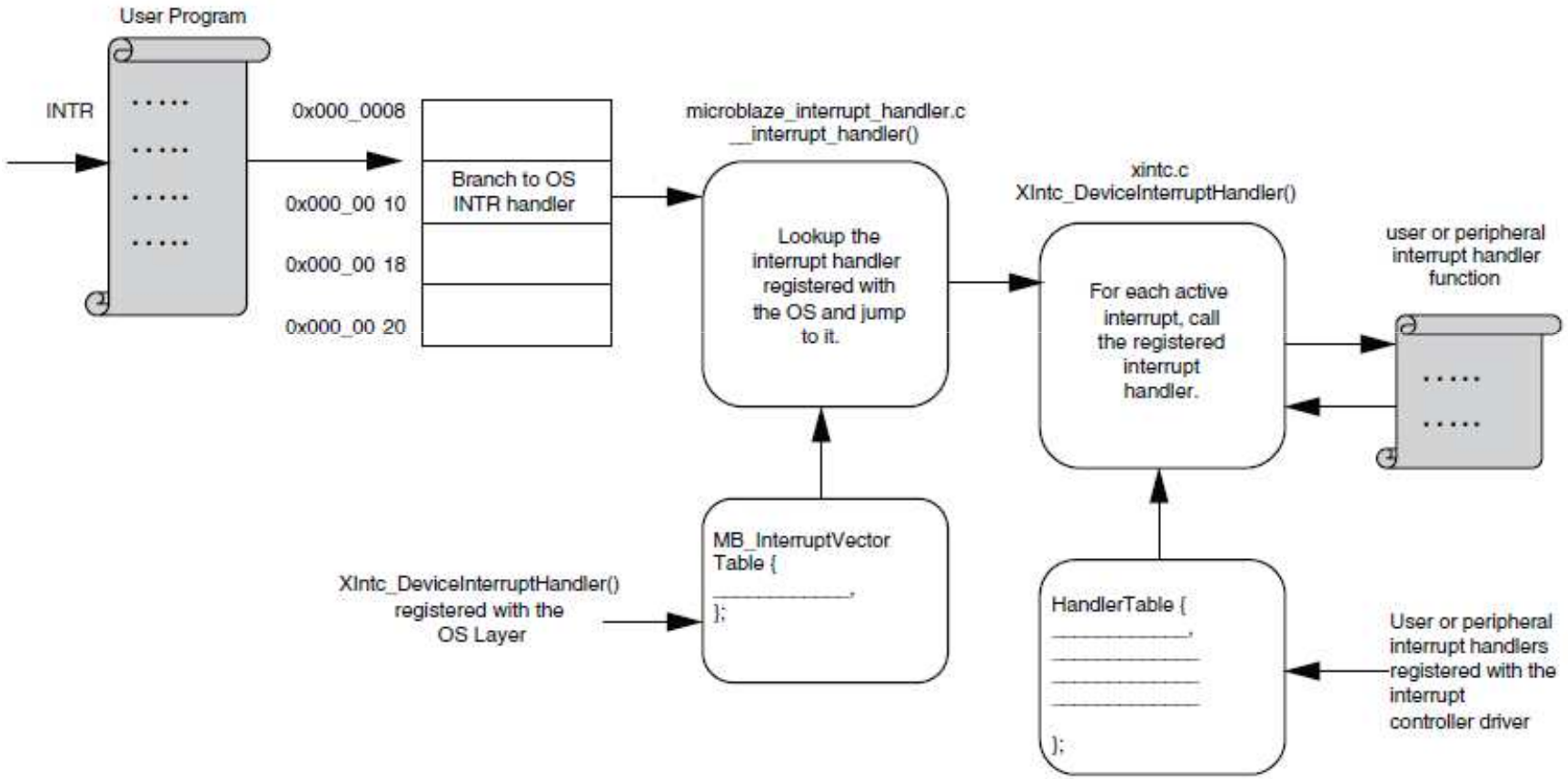
GESTIÓN DE INTERRUPCIONES: CON CONTROLADOR



GESTIÓN DE INTERRUPCIONES: CON CONTROLADOR

- Int Xintc_Initialize (Xintc **InstancePtr,* u16 *DeviceId*)
- Int Xintc_Connect(Xintc **InstancePtr*, u8 *Id*, XInterruptHandler *Handler*, void **CallBackRef*)
- Int Xintc_Disconnect(Xintc **InstancePtr*, u8 *Id*)
- Void Xintc_Enable (Xintc **InstancePtr*, u8 *Id*)
- Void Xintc_Disable (Xintc **InstancePtr*, u8 *Id*)
- Int Xintc_Start (Xintc **InstancePtr*, u8 *Mode*)
- Int Xintc_Stop(Xintc **InstancePtr*)

GESTIÓN DE INTERRUPCIONES : FLUJO DE ATENCIÓN A LA INTERRUPCIÓN (CON CONTROLADOR)



"Embedded System Tools Reference Guide", Appendix B ***Interrupt Management.*** *Xilinx*

GESTIÓN DE INTERRUPCIONES : CÓDIGO

```
71 int main() {
72
73   int count_mod_3;
74
75    // Enable MicroBlaze Interrupts
76   microblaze_enable_interrupts();
77
78   /* Register the Timer interrupt handler in the vector table */
79   XIntc_RegisterHandler(XPAR_XPS_INTC_0_BASEADDR,
80                         XPAR_XPS_INTC_0_DELAY_INTERRUPT_INTR,
81                         (XInterruptHandler) timer_int_handler,
82                         (void *)XPAR_DELAY_BASEADDR);
83
84   /* Initialize and set the direction of the GPIO connected to LEDs */
85   XGpio_Initialize(&gpio, XPAR_LEDS_8BIT_DEVICE_ID);
86   XGpio_SetDataDirection(&gpio,LEDChan, 0);
87
```

"Embedded System Tools Reference Guide", Appendix B ***Interrupt Management.*** *Xilinx*

BIBLIOGRAFÍA:

- "OS and Libraries Document Collection", Xilinx → Librerías
- "Embedded System Tools Reference Guide", Appendix B ***Interrupt Management***