
Ambiente de Desarrollo de Software

(Adaptado del curso “Embedded System Design Flow” de Xilinx)

Objetivos

➤ Al completar este módulo el alumno será capaz de:

- Entender los conceptos básicos del IDE Eclipse en SDK
- Listar las características del SDK
- Identificar las funcionalidades de las herramientas GNU
- Listar los pasos en la creación de una aplicación de software
- Describir las secciones de los archivos objeto
- Describir lo que hace un linker script

Temario

➤ **Introducción**

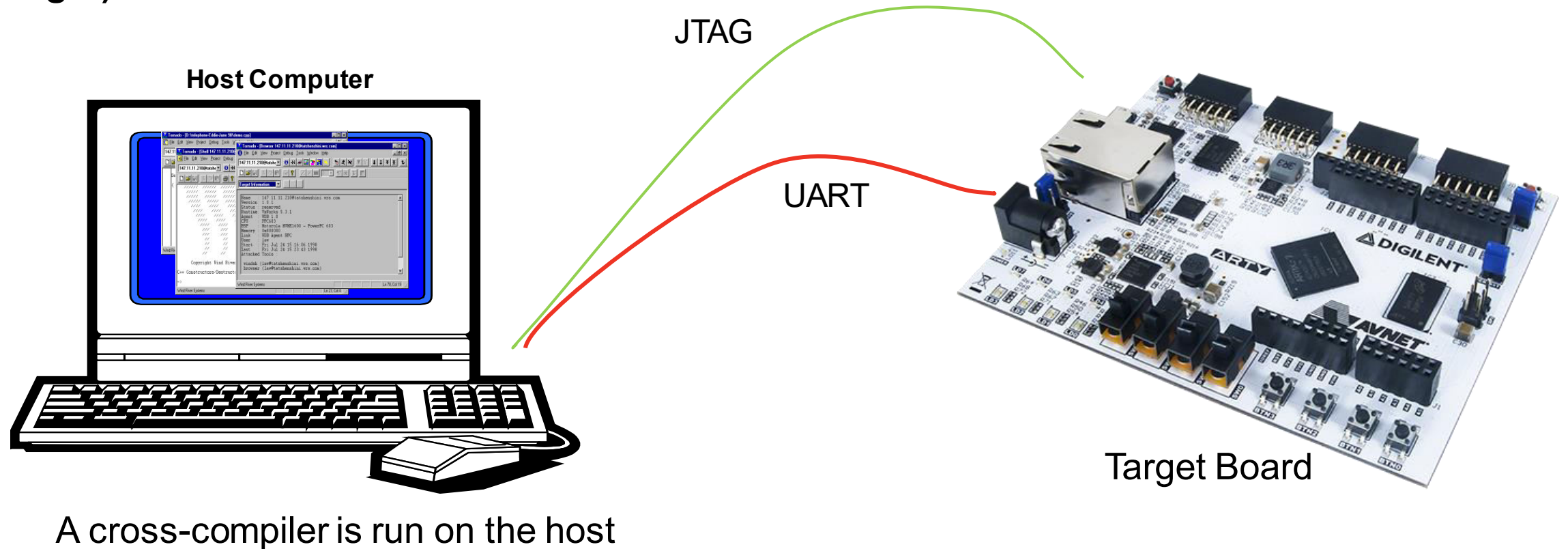
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- Resumen

Desktop versus Embebido

- **Desarrollo Desktop:** escrito, depurado, y corrido en la misma máquina
- **El SO carga el programa en la memoria cuando se requiere que el programa se ejecute**
- **La resolución de direcciones toma lugar al momento de la carga por un programa llamado loader**
 - El loader está incluido en el SO
- **El programador une todo en un archivo ejecutable llamado ELF**
 - Código de Booteo, código de aplicación, RTOS, e ISRs
 - La resolución de direcciones toma lugar durante la etapa de *gluing*
- **El archivo ejecutable es descargado en el sistema destino a través de diferentes métodos**
 - Programador Ethernet, serial, JTAG, BDM, ROM

Desktop versus Embebido

- **El desarrollo toma lugar en una máquina (host) y es descargado al sistema embebido (target)**



Desarrollo Embebido

➤ Diferentes problemas

- Hardware único para cada diseño
- Confiabilidad
- Requerimientos de respuesta en tiempo real (a veces)
 - RTOS vs OS
- Densidad de código
- Lenguajes de alto nivel y ensamblado

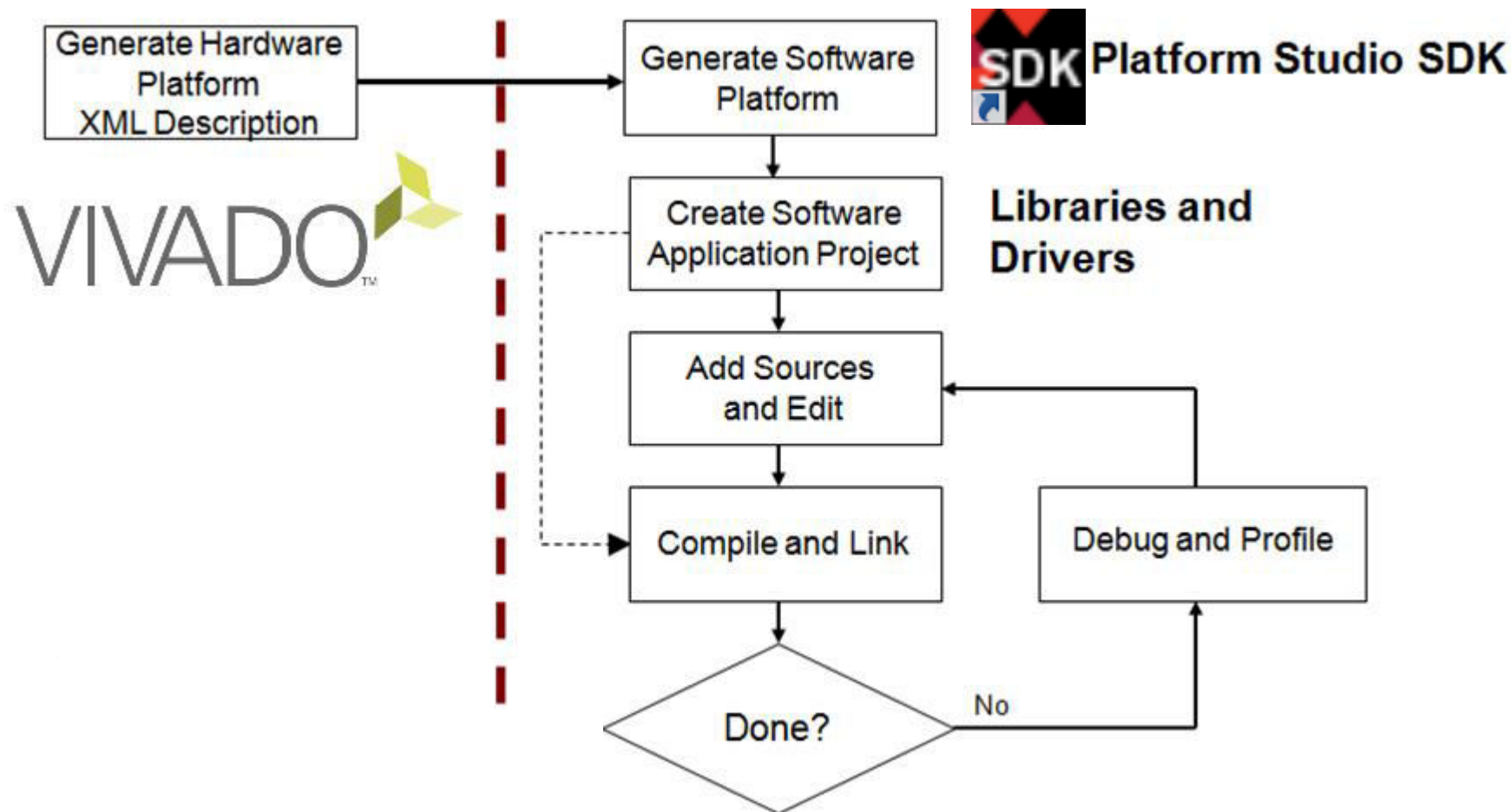
Herramientas de Desarrollo de Software

	Xilinx Supplied	ARM Supplied	3rd Party Supplied
	Xilinx Basic Features	ARM® Fully Featured	3rd Party Cortex™A9 Vendors
Software Platform	Standalone and Linux Drivers Example Boot Code	Drives Connected Community	Operating Systems Middleware Codecs, etc.
Software Development	Platform Studio SDK (Eclipse IDE) ARM GNU CC	ARM® Development Studio IDE (DS-5)	IDEs, OS-specific
Debug	Platform Studio SDK SDK Profiler	DS-5 Debugger / Profiler	Debuggers & Profilers
	USB Cable download, run-time control	RVI Debug DSTREAM Trace	ICE & Trace
JTAG / ARM CoreSight™ Infrastructure			

Temario

- Introducción
- ***Ambiente de Desarrollo SDK***
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- Resumen

Flujo de Desarrollo de una aplicación en SDK



Frameworks of SDK

➤ Builder framework

- Compila y linkea los archivos fuente.
- Se especifican opciones por defecto para el Build cuando la aplicación es creada: Elección de Debug, Release, Configuraciones
- El usuario puede personalizar dichas opciones más tarde al momento de desarrollar la aplicación.
- Tipos de Build: Standard Make, Managed Make

➤ Launch framework

- Especifica que acciones se necesita que se tomen: Ejecutar una aplicación o Depurarla

➤ Debug framework

- Lanza el depurador (gdb), carga la aplicación e inicia la sesión de depuración.
- Las vistas de depuración muestran información acerca del estado de la sesión de depuración

➤ Search framework

- Asiste en el desarrollo de la aplicación.

➤ Help System

- Sistema de ayuda en línea; sensible al contexto



Workspaces y Perspectivas

➤ **Workspace**

- Ubicación para almacenar preferencias e información interna acerca de los proyectos
- Transparente para los usuarios

➤ **Vistas, Editores**

- Elemento de la interfaz básica de usuario

➤ **Perspectivas**

- Colección de vistas de funcionalidad relacionadas
- El layout de vistas en una perspectiva puede ser personalizada de acuerdo a la preferencia del usuario.

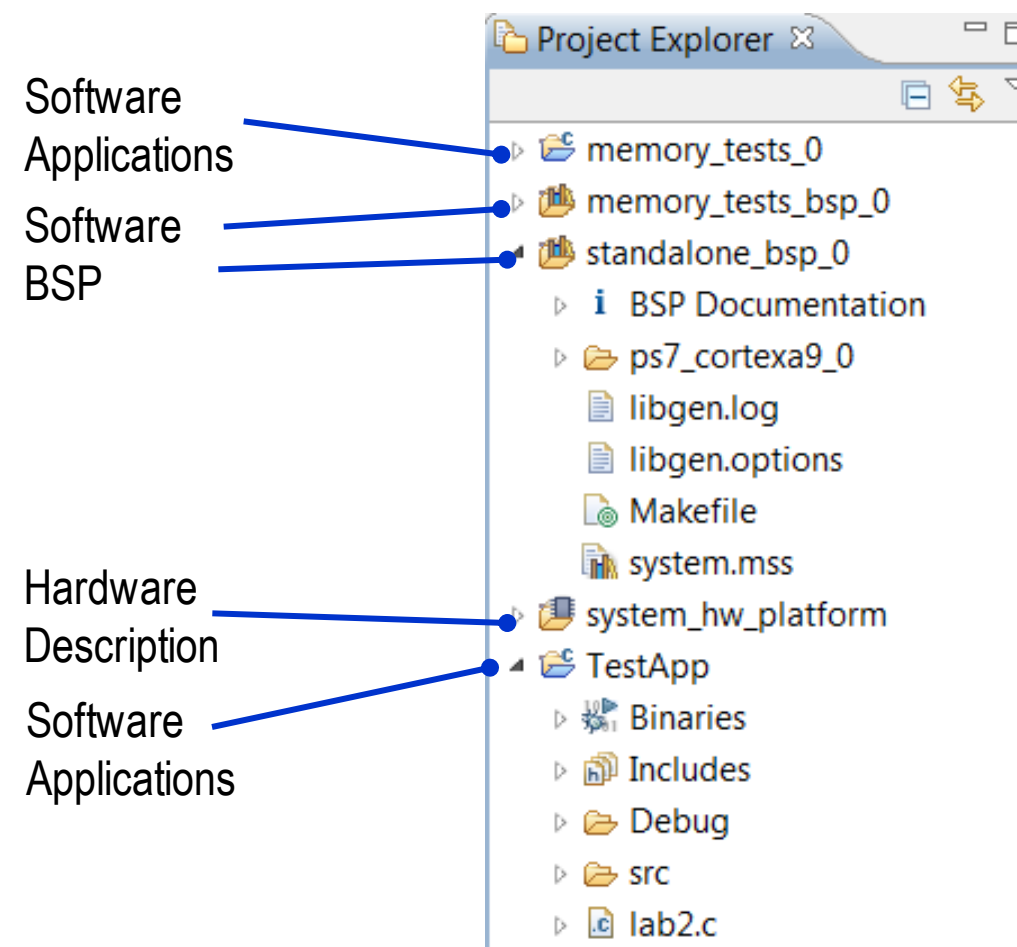
Vistas

- **Vistas de la Plataforma Eclipse: Navigator view, Tasks view, Problems view**
- **Debug views: Stack view, Variables view**
- **C/C++ views: Projects view, Outline view**

[illegible]

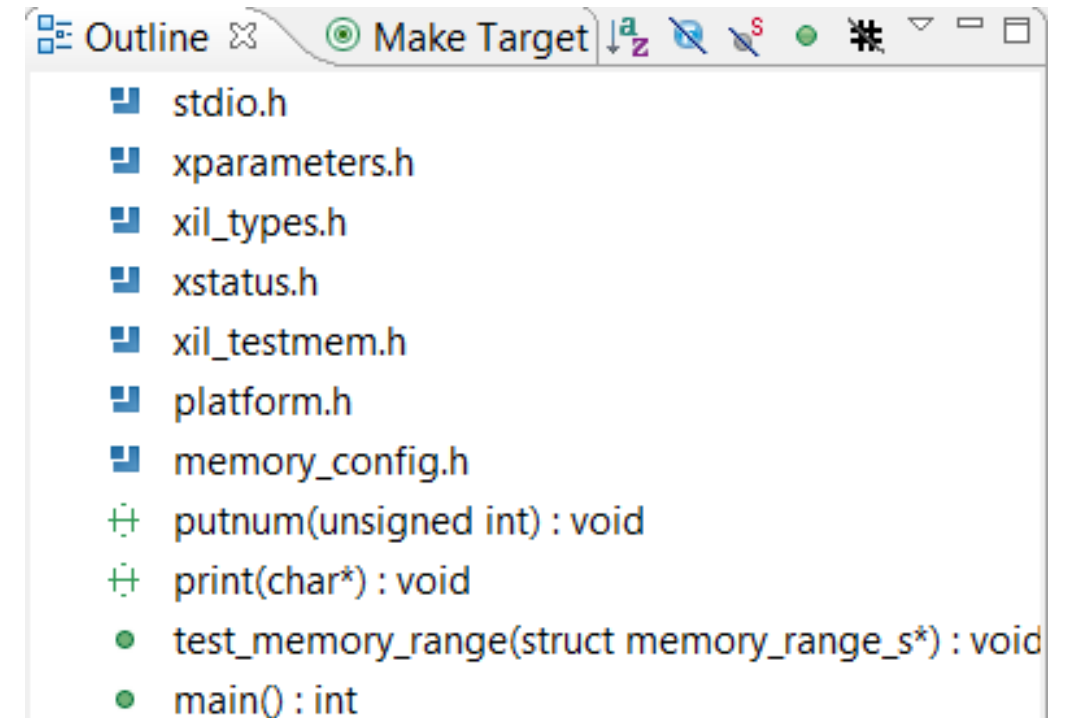
Vista de Proyecto C/C++

- Lista jerárquica de los workspace de los proyectos.
- Double-click para abrir un archivo.
- Right-click sobre el proyecto para acceder a sus propiedades.



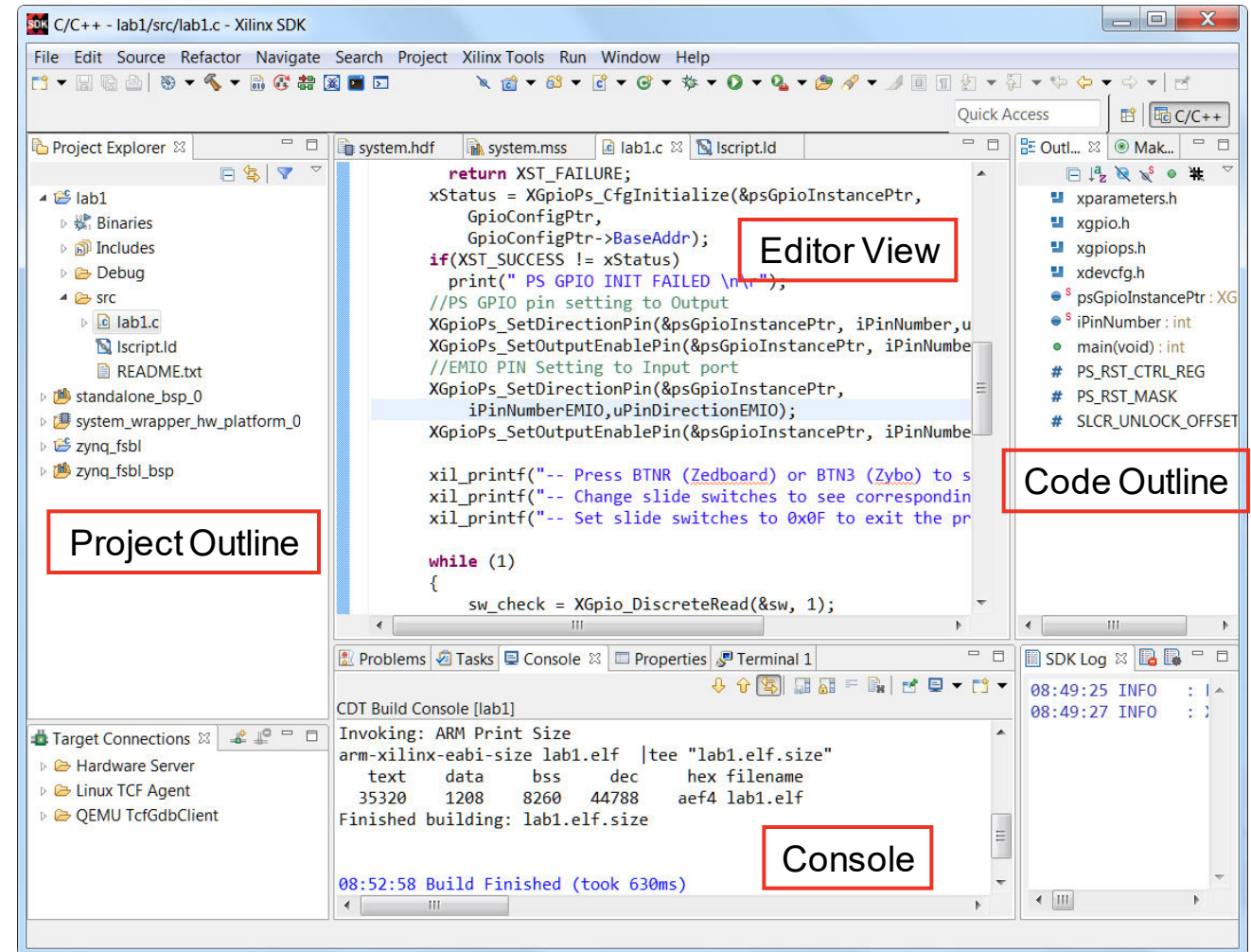
Vista Outline

- Muestra un esquema del archivo que está actualmente abierto en el editor
- El tipo de contenido está indicado por el icono
- Para un código en C, los iconos representan:
 - Sentencias *#define*
 - Archivos Include
 - Llamadas a Función
 - Declaraciones
- Seleccionando un símbolo se navegará al mismo en la ventana de edición.



Perspectiva C/C++

- C/C++ project outline muestra los elementos de un proyecto con iconos para una identificación sencilla.
- Editor C/C++ para la creación integrada de software.
- Code outline muestra los elementos de software bajo desarrollo con iconos para una sencilla identificación.
- Las vistas Problems, Console y Properties listan información de salida asociada con el flujo de desarrollo del software.



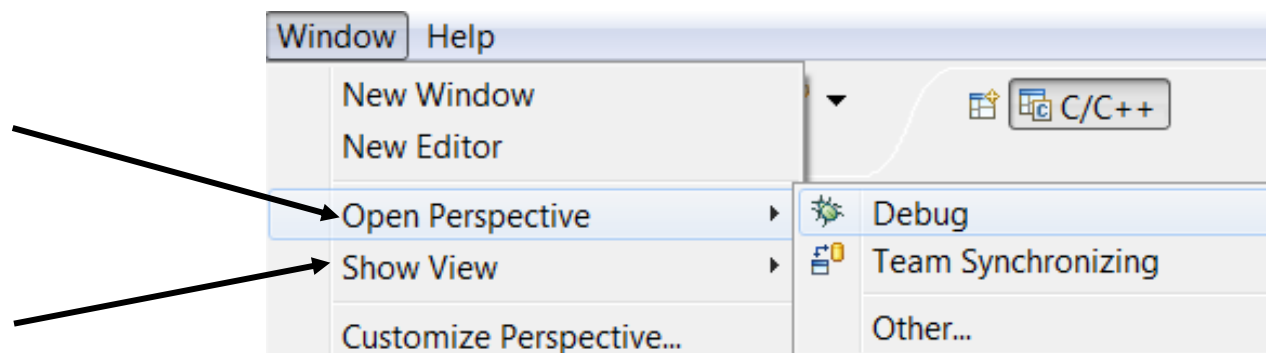
Abriendo Perspectivas y Vistas

➤ Para abrir una Perspectiva, usar

- **Window → Open Perspective**

➤ Para abrir una Vista, usar

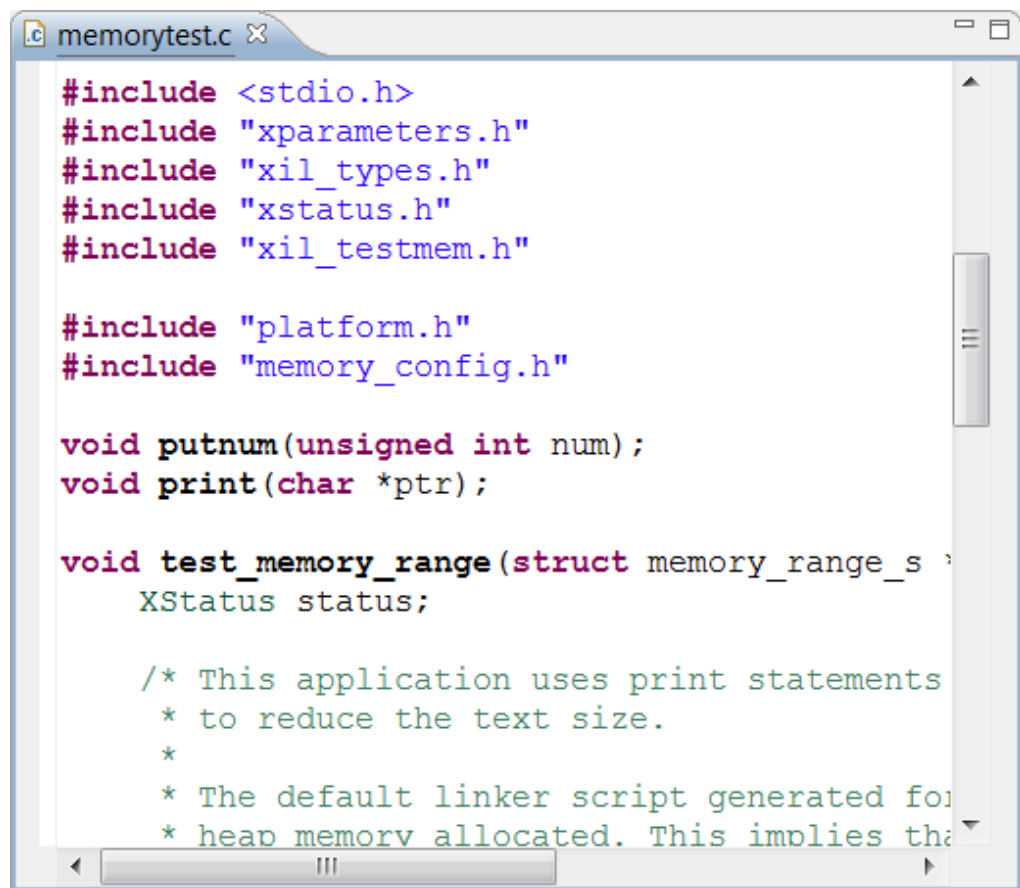
- **Window → Show View**
- Si la vista ya está presente en la actual perspectiva se la sobrealta



Editor

➤ Resultado de sintaxis

- Coincidencia de paréntesis
- Coloreo de sintaxis
- Asistencia de contenido
- Atajo de teclado



The screenshot shows a code editor window titled 'memorytest.c'. The code is written in C and features syntax highlighting: preprocessor directives are in purple, keywords like 'void', 'struct', and 'unsigned' are in red, and identifiers and literals are in black. Comments are in green. The code includes several headers, defines two functions, and contains a multi-line comment.

```
#include <stdio.h>
#include "xparameters.h"
#include "xil_types.h"
#include "xstatus.h"
#include "xil_testmem.h"

#include "platform.h"
#include "memory_config.h"

void putnum(unsigned int num);
void print(char *ptr);

void test_memory_range(struct memory_range_s ,
    XStatus status;

    /* This application uses print statements
     * to reduce the text size.
     *
     * The default linker script generated for
     * heap memory allocated. This implies the
```

Temario

- Introducción
- Ambiente de Desarrollo SDK
- ***Creación de un proyecto en SDK***
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- Resumen

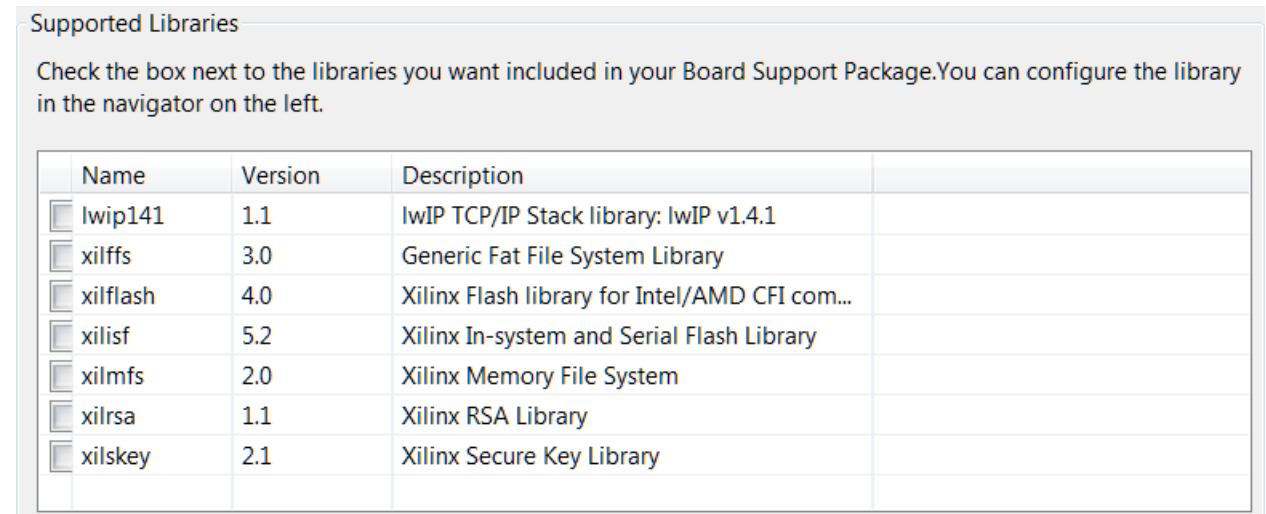
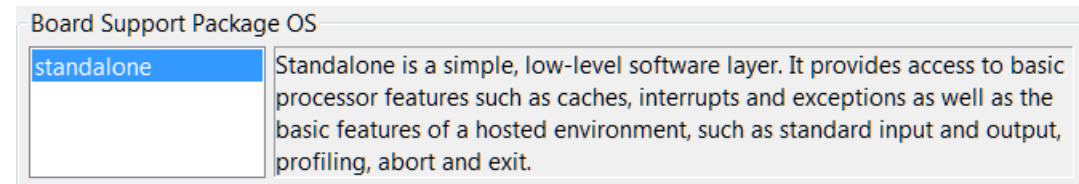
Lanzando el SDK

➤ Lanzamiento del SDK

- Standalone
 - Elegir el workspace, elegir Hardware Platform Specification
- En Vivado
 - **File> Export Hardware**
 - **File > Launch SDK**
- Exportando
 - Lo primero que se genera es un archivo de descripción de hardware (Hardware Description File - HDF).
 - Luego se crea automáticamente un proyecto de especificación de plataforma de hardware (hardware platform specification project).
 - Entonces la aplicación de software (y el board support package) puede ser creada y asociada con la plataforma de hardware.

Creando un Board Support Package

- El Board Support Package provee servicio de software basados en el procesador y los periféricos que conforman el sistema de procesamiento.
- Puede ser automáticamente creado cuando se crea el proyecto de aplicación.
- Puede ser creado de manera standalone.
- Debe ser asociado a una plataforma de Hardware
 - File > New > Board Support Package
 - Seleccionar soporte para el OS apropiado
 - Son soportados SO de terceras partes con la selección apropiada de BSP.
 - Seleccionar el soporte de librerías requeridas.



Creando un Proyecto de Aplicación de Software

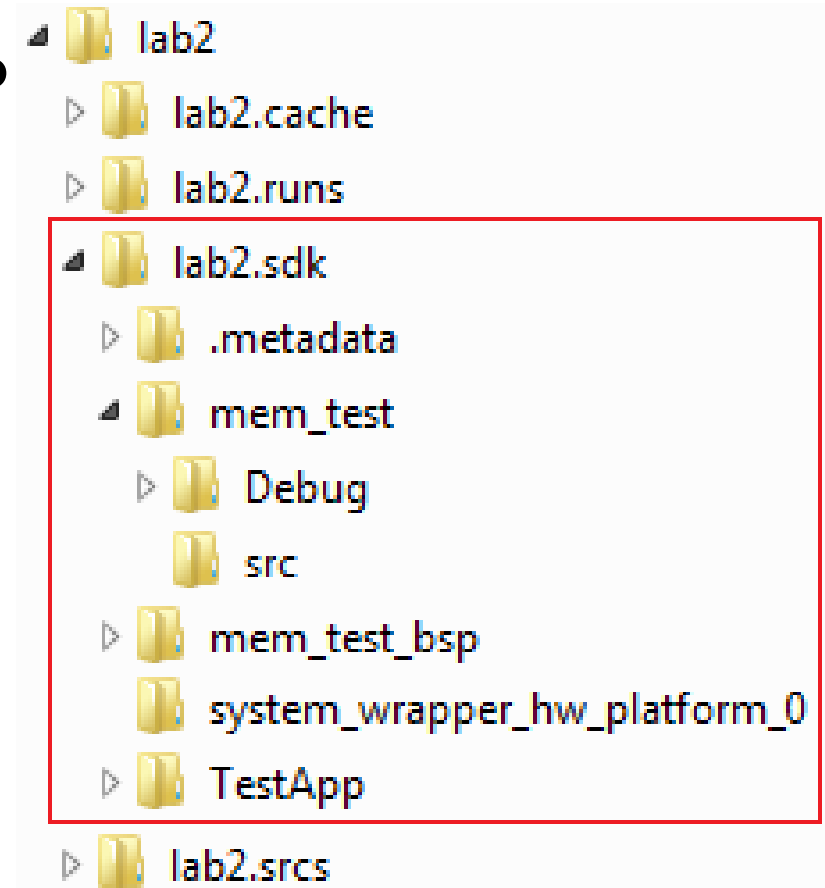
- **SDK soporta proyectos de aplicación de software múltiples.**
- **Un proyecto de software es asociado a un proyecto BSP.**
- **Se proveen aplicaciones de ejemplo.**
 - Muy bueno para pruebas rápidas de hardware.
 - Tests de periféricos.
 - Punto de entrada para usar de base para una aplicación propia.
- **Típicamente se abre una aplicación vacía para comenzar un proyecto no-estándar**

Available Templates:

- Peripheral Tests
- Dhrystone
- Empty Application
- Hello World**
- IwIP Echo Server
- Memory Tests
- RSA Authentication App
- SREC Bootloader
- SREC SPI Bootloader
- Xilkernel POSIX Threads Demo
- Zynq DRAM tests
- Zynq FSBL

Estructura de Directorios

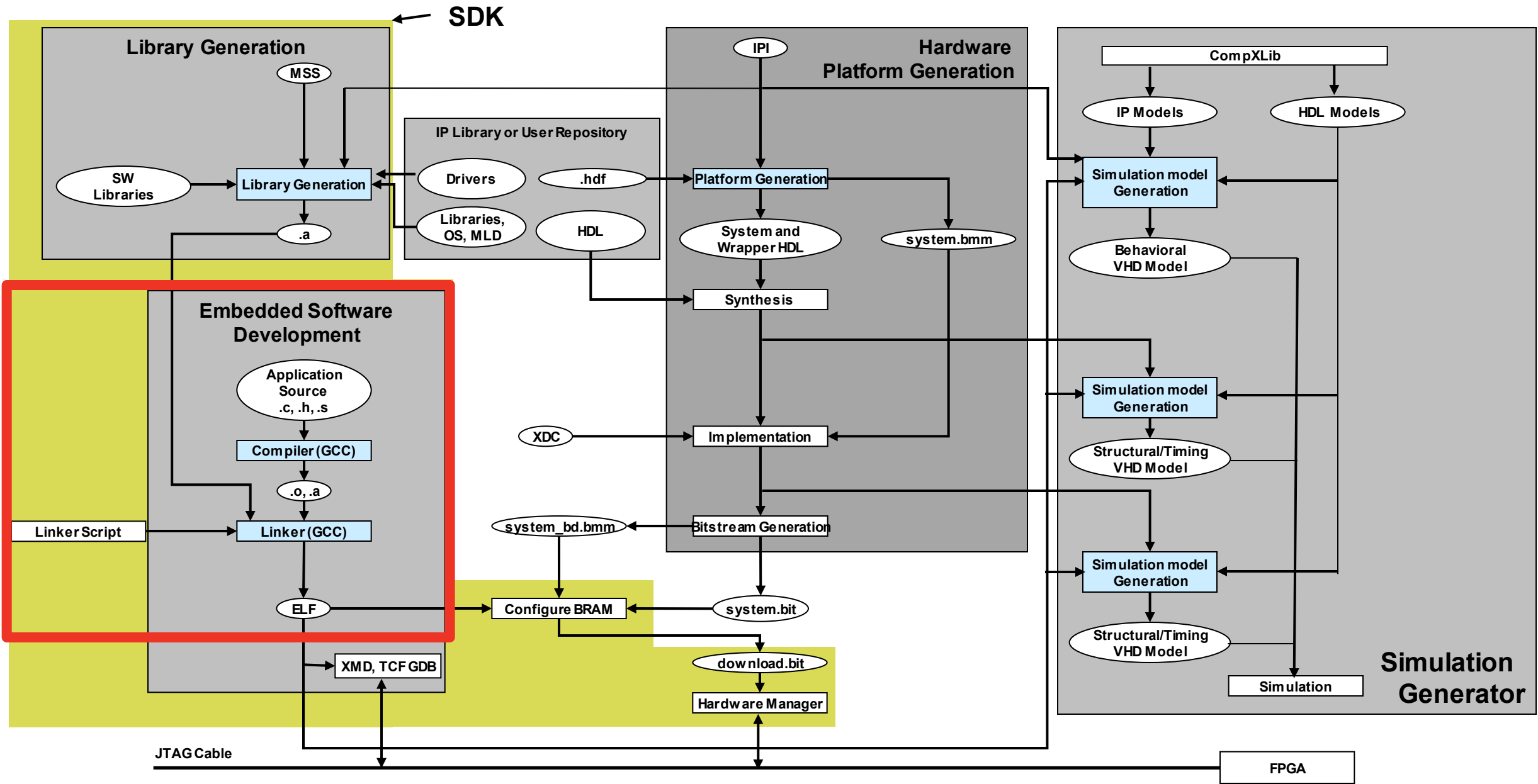
- Los proyectos SDK son ubicados en el directorio de la aplicación, que fue especificado cuando el SDK fue lanzado
- Cada proyecto puede tener múltiples directorios para los archivos de sistema y configuraciones
- Configuraciones
 - Release configuration
 - Debug configuration
 - Profile configuration
- A Debug configuration is created by default



Temario

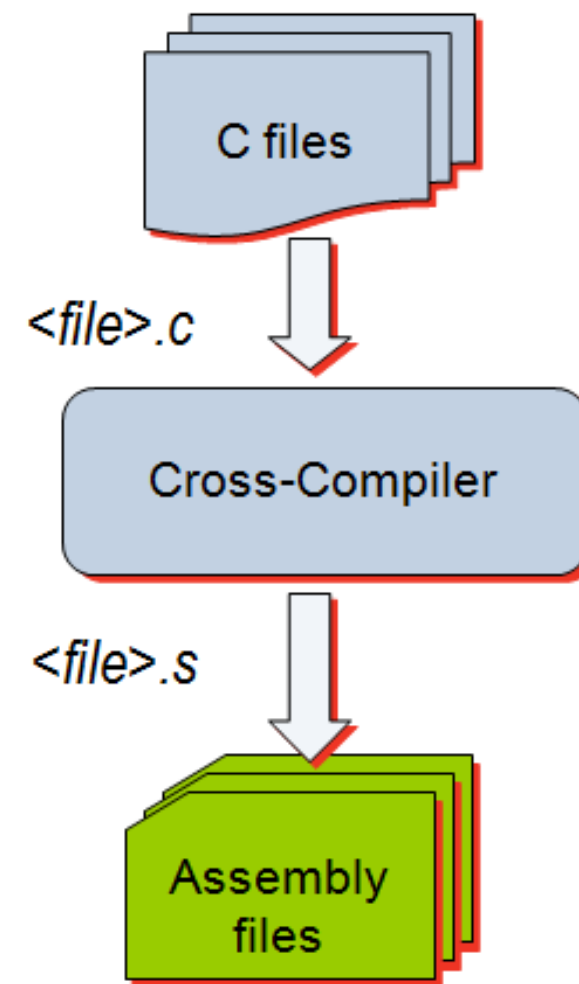
- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- ***Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils***
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- Resumen

Flujo de la herramienta (SDK)



Herramientas GNU: GCC

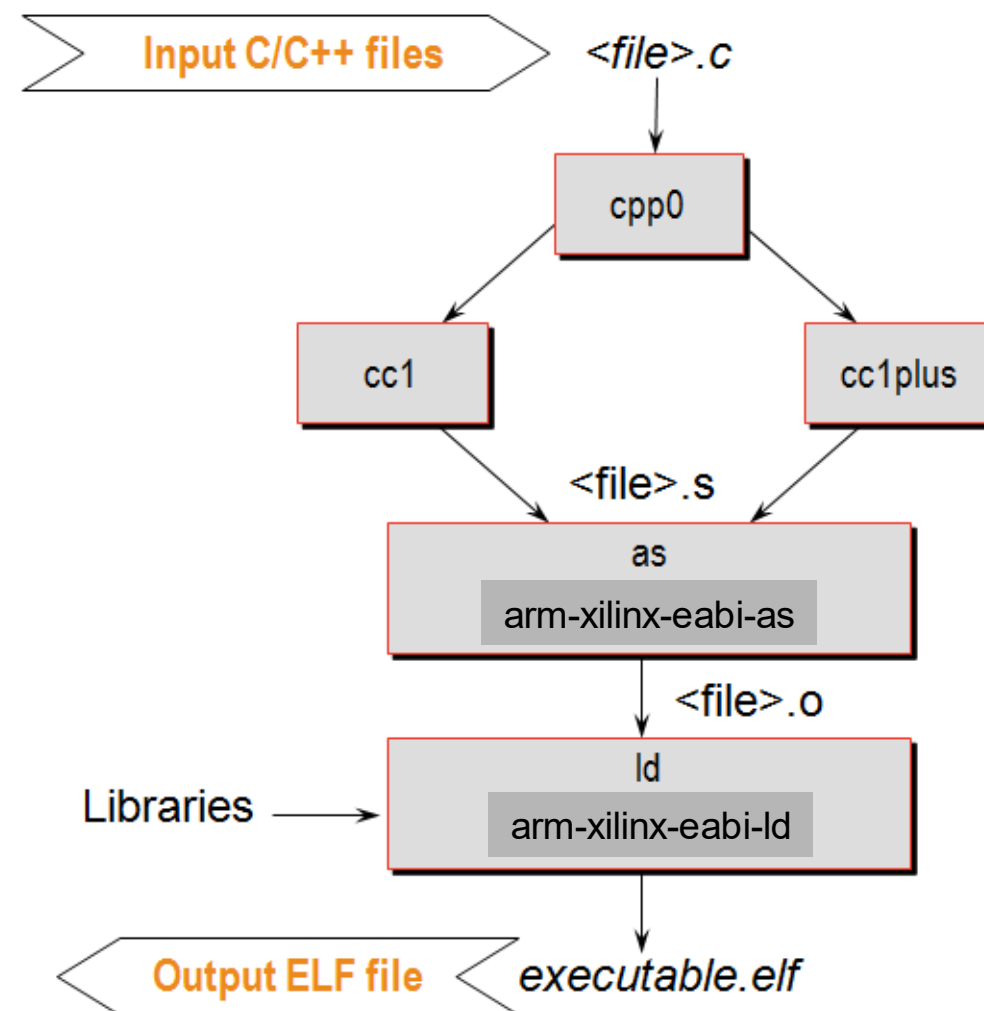
- El GCC traduce código C en lenguaje assembly
- También funciona como interfaz de usuario al ensamblador y al linker GNU, llamándolos con los parámetros apropiados
- Cross-compilers soportados:
 - GNU GCC (arm-xilinx-eabi-gcc)



Herramientas GNU: GCC

➤ Llama 4 ejecutables diferentes

- Preprocesador (cpp0)
 - Reemplaza todas las macros con las definiciones existentes en el código fuente y en los archivos de encabezado.
- Compilador C de lenguaje específico
 - cc1 lenguaje de programación C
 - cc1plus lenguaje C++
- Assembler
 - arm-xilinx-eabi-as
- Linker
 - arm-xilinx-eabi-ld



Herramientas GNU: AS

Ensamblador

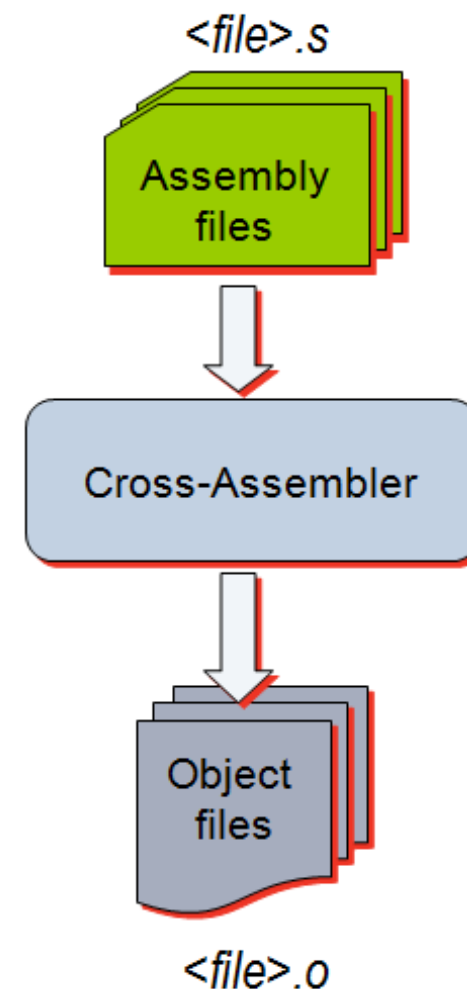
➤ Entrada: Archivos en lenguaje Assembly

- Extensión de archivo: .s

➤ Output: Código objeto

- Extensión de archivo: .o
- Contiene
 - Pedazo de código ensamblado
 - Datos constantes
 - Referencias externas
 - Información de depuración

➤ Típicamente, el compilador llama automáticamente al ensamblador



Herramientas GNU: LD

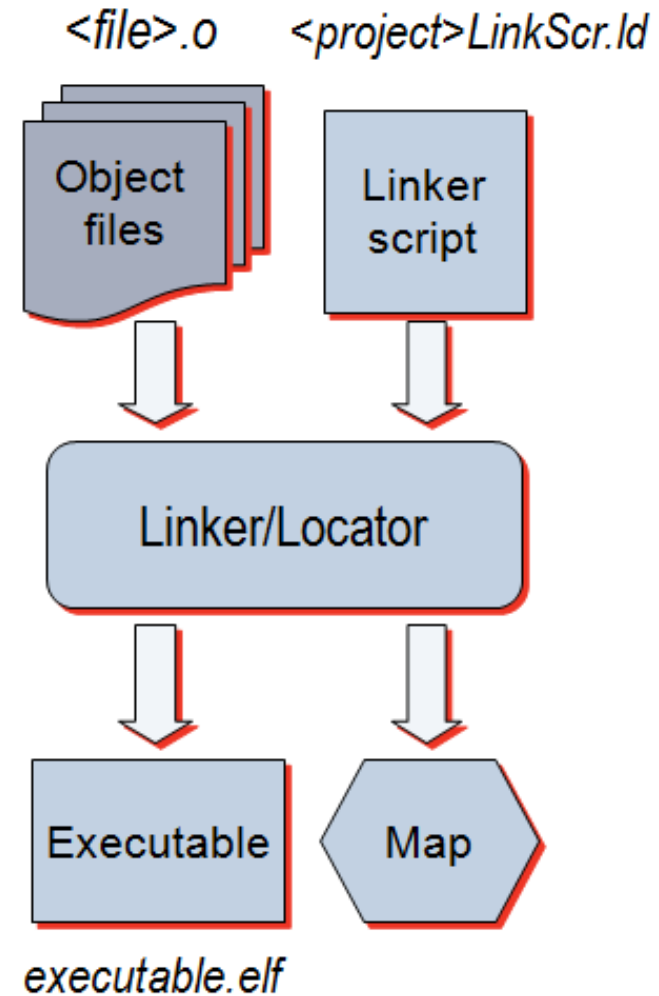
Linker

➤ Entradas:

- Several object files
- Archivos objeto empaquetados (library)
- Linker script (mapfile)

➤ Salidas:

- Imagen ejecutable (.ELF)
- Mapa de archivo



Utilidades GNU

➤ AR Archiver

- Crea, modifica, y extrae desde librerías
- Usado en SDK para combinar los archivos objeto del Board Support Package (BSP) en una librería
- Usado en SDK para extraer archivos objeto desde diferentes librerías

➤ Object Dump

- Muestra información de archivos objeto y ejecutables
 - Información de encabezado, mapa de memoria
 - Datos
 - Código desensamblado

Object Dump

Muestra un resumen de información de las secciones

arm-xilinx-eabi-objdump -h executable.elf

TestApp.elf: file format elf32-littlearm

Sections:

Idx	Name	Size	UMA	LMA	File off	Algn
0	.text	00001950	00100000	00100000	00008000	2**2
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
1	.init	00000018	00101950	00101950	00009950	2**2
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
2	.fini	00000018	00101968	00101968	00009968	2**2
	CONTENTS, ALLOC, LOAD, READONLY, CODE					
3	.rodata	00000000	00101980	00101980	00009980	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
4	.eh_frame	00000004	00101f54	00101f54	00009f54	2**2
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
6	.bss	0000005c	00101f58	00101f58	00009f58	2**2
	ALLOC					
7	.mmu_tbl	0000a04c	00101fb4	00101fb4	00009fb4	2**0
	CONTENTS, ALLOC, LOAD, READONLY, DATA					
8	.init_array	00000008	0010c000	0010c000	00014000	2**2
	CONTENTS, ALLOC, LOAD, DATA					

Nombre de Sección

Tamaño de Sección

Dirección de Memoria Virtual

Loadable Memory Address

Offset

Alineación por Byte

Object Dump

Volcado del código fuente y ensamblado

arm-xilinx-eabi-objdump -S executable.elf

```
int main (void)
{
    1003bc:    e92d4800    push    {fp, lr}
    1003c0:    e28db004    add     fp, sp, #4
    1003c4:    e24dd030    sub     sp, sp, #48    ; 0x30
    XGpio dip, push;
        int i, psb_check, dip_check;

    //xil_printf("-- Start of the Program --\r\n");

    XGpio_Initialize(&dip, XPAR_DIP_DEVICE_ID);
    1003c8:    e24b3020    sub     r3, fp, #32
    1003cc:    e1a00003    mov     r0, r3
    1003d0:    e3a01000    mov     r1, #0
    1003d4:    eb000326    bl      101074 <XGpio_Initialize>
    XGpio_SetDataDirection(&dip, 1, 0xffffffff);
    1003d8:    e24b3020    sub     r3, fp, #32
    1003dc:    e1a00003    mov     r0, r3
    :      e3a01001    mov     r1, #1
    :      e3e02000    mun     r2, #0
    :      eb00024e    bl      100d28 <XGpio_SetDataDirection>

    XGpio_Initialize(&push, XPAR_PUSH_DEVICE_ID);
```

Ubicación en
la memoria

Instrucción en
Código C

Instrucción en
Lenguaje de máquina

Instrucción en
Assembly

Temario

- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- ***Configuración del Software***
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- Resumen

Sistemas Operativos

- **Los SO son una colección de rutinas de software que componen un conjunto unificado y estándar de servicios de sistema.**
- **La opción Standalone es usada cuando no se desea usar un SO**
 - Provee una cantidad mínima de servicios de procesador y librerías.
 - Puede considerarse como un SO mínimo no estándar.
 - Instalado como una plataforma de software.
- **Existen una cantidad variada de SO de terceras partes:**
 - Linux (diferentes variantes)
 - RTOS – real-time operating system (diferentes variantes); Free RTOS (una opción para el procesador Cortex™-A9)
 - XilKernel – provisto por Xilinx; pequeño y simple; sólo para MicroBlaze
- **Los SO son instalados y se convierten en parte del Board Support Package (BSP)**

¿Qué provee un Sistema Operativo?

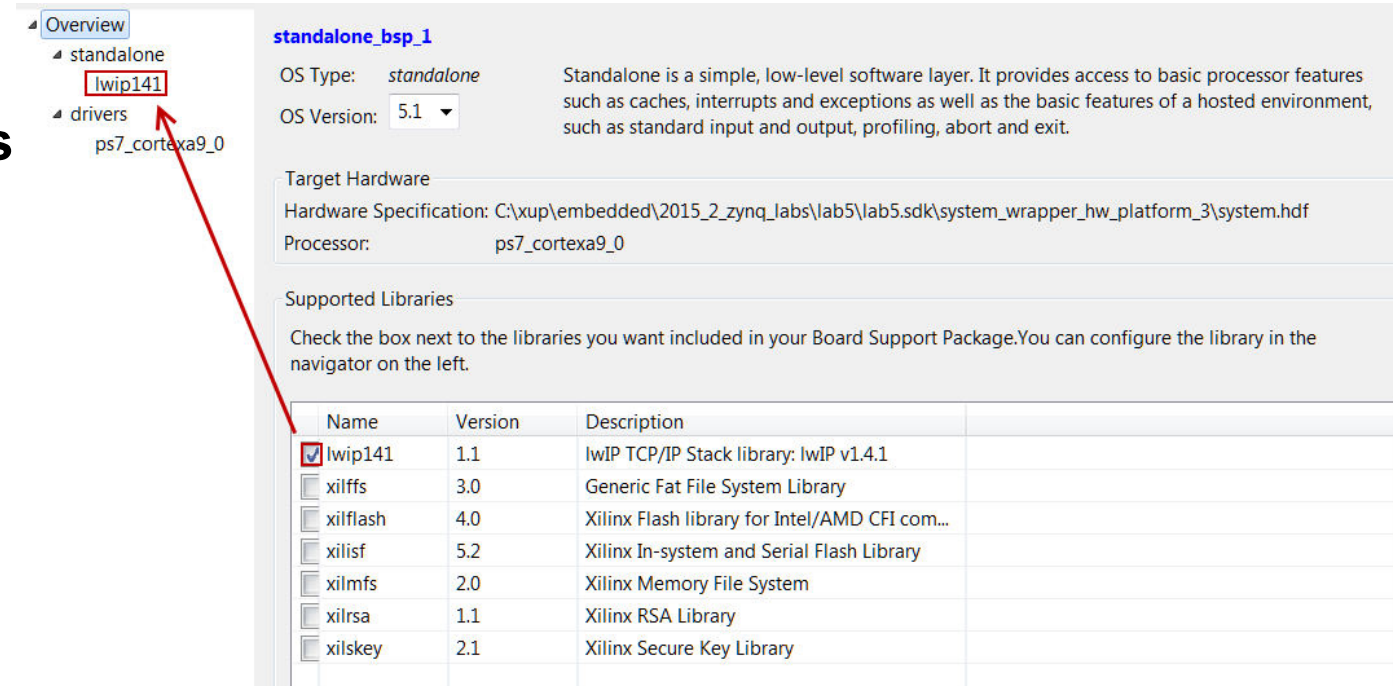
➤ Servicios de un Sistema Operativo

- Soporte para GUI
- Servicios TCP/IP**
- Manejo de Tareas
- Manejo de recursos**
- Conexión sencilla a aplicaciones ya escritas
- Habilidad para recargar y cambiar aplicaciones.
- Servicios completos para sistemas de archivo**

** También disponible como adicionales al Standalone BSP

Accediendo las Propiedades de la Plataforma de Software

- Seleccionar el board support package creado en la vista Project Explorer
- Xilinx Tools > Board Support Package Settings
- Establecer todas las opciones del software BSP
- Se tiene selección de múltiples forms
 - Overview
 - Standalone
 - Drivers
 - CPU



standalone_bsp_1

OS Type: standalone OS Version: 5.1

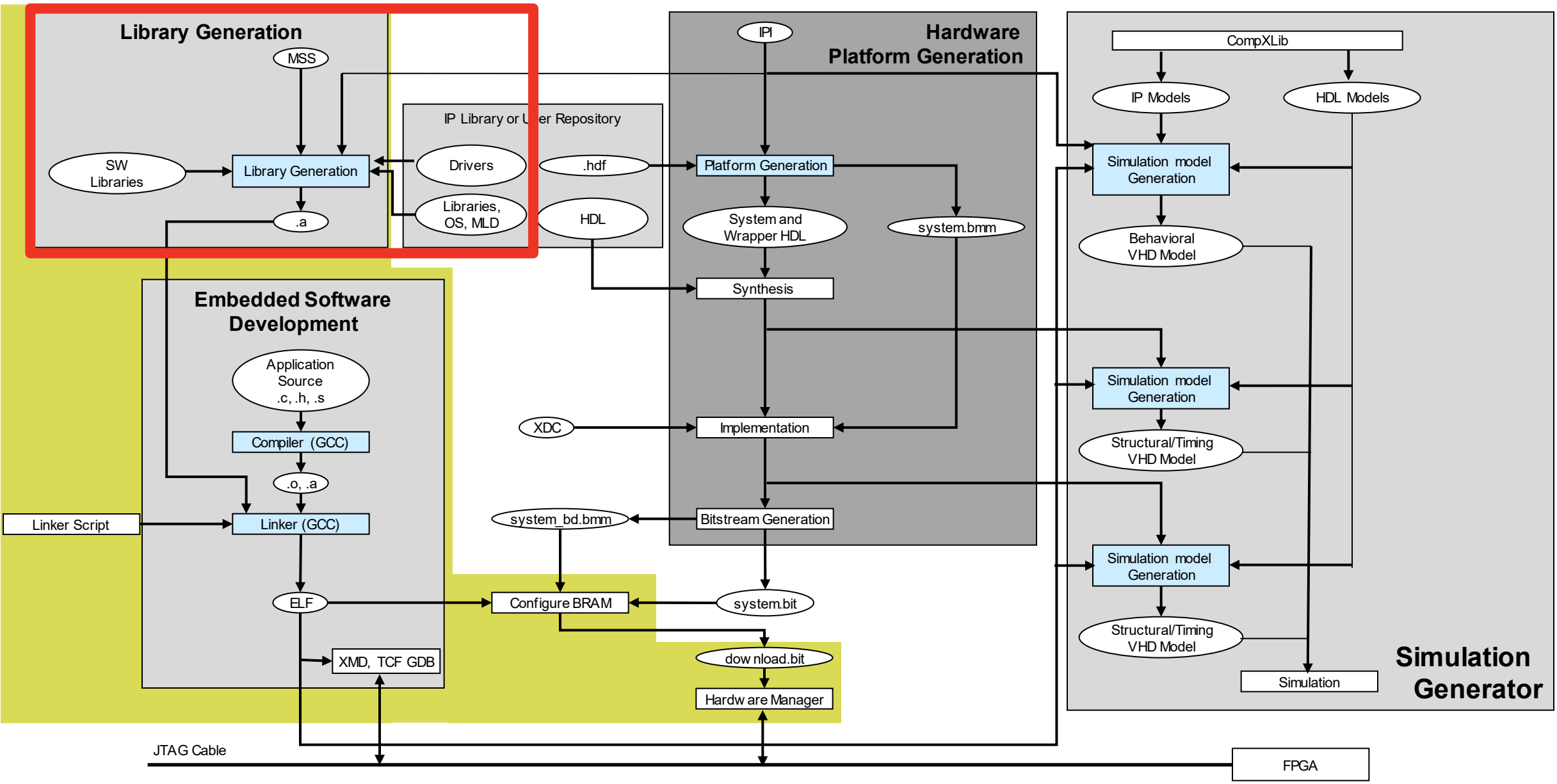
Target Hardware
Hardware Specification: C:\xup\embedded\2015_2_zynq_labs\lab5\lab5.sdk\system_wrapper_hw_platform_3\system.hdf
Processor: ps7_cortexa9_0

Supported Libraries

Check the box next to the libraries you want included in your Board Support Package. You can configure the library in the navigator on the left.

Name	Version	Description
<input checked="" type="checkbox"/> lwip141	1.1	lwIP TCP/IP Stack library: lwIP v1.4.1
<input type="checkbox"/> xilffs	3.0	Generic Fat File System Library
<input type="checkbox"/> xilflash	4.0	Xilinx Flash library for Intel/AMD CFI com...
<input type="checkbox"/> xilif	5.2	Xilinx In-system and Serial Flash Library
<input type="checkbox"/> xilmfs	2.0	Xilinx Memory File System
<input type="checkbox"/> xilrsa	1.1	Xilinx RSA Library
<input type="checkbox"/> xilkey	2.1	Xilinx Secure Key Library

Flujo de la Herramienta Embebida (SDK)



Flujo de la Generación de una Librería (en SDK)

➤ Archivos de entrada → MSS

- Archivos de salida → libc.a, libXil.a, libm.a
- El generador de librería es generalmente la primera herramienta que corre para configurar librerías y drivers de dispositivos
 - El archivo MSS define los drivers asociados a los periféricos, dispositivos de E/S estándar, y otras características de software relacionadas.
- El generador de librería configura librerías y drivers con esta información y produce un archivo de archivos objeto:
 - libc.a – Librería C estándar
 - libXil.a – Librería de Xilinx
 - libm.a – Librería de funciones matemáticas

Temario

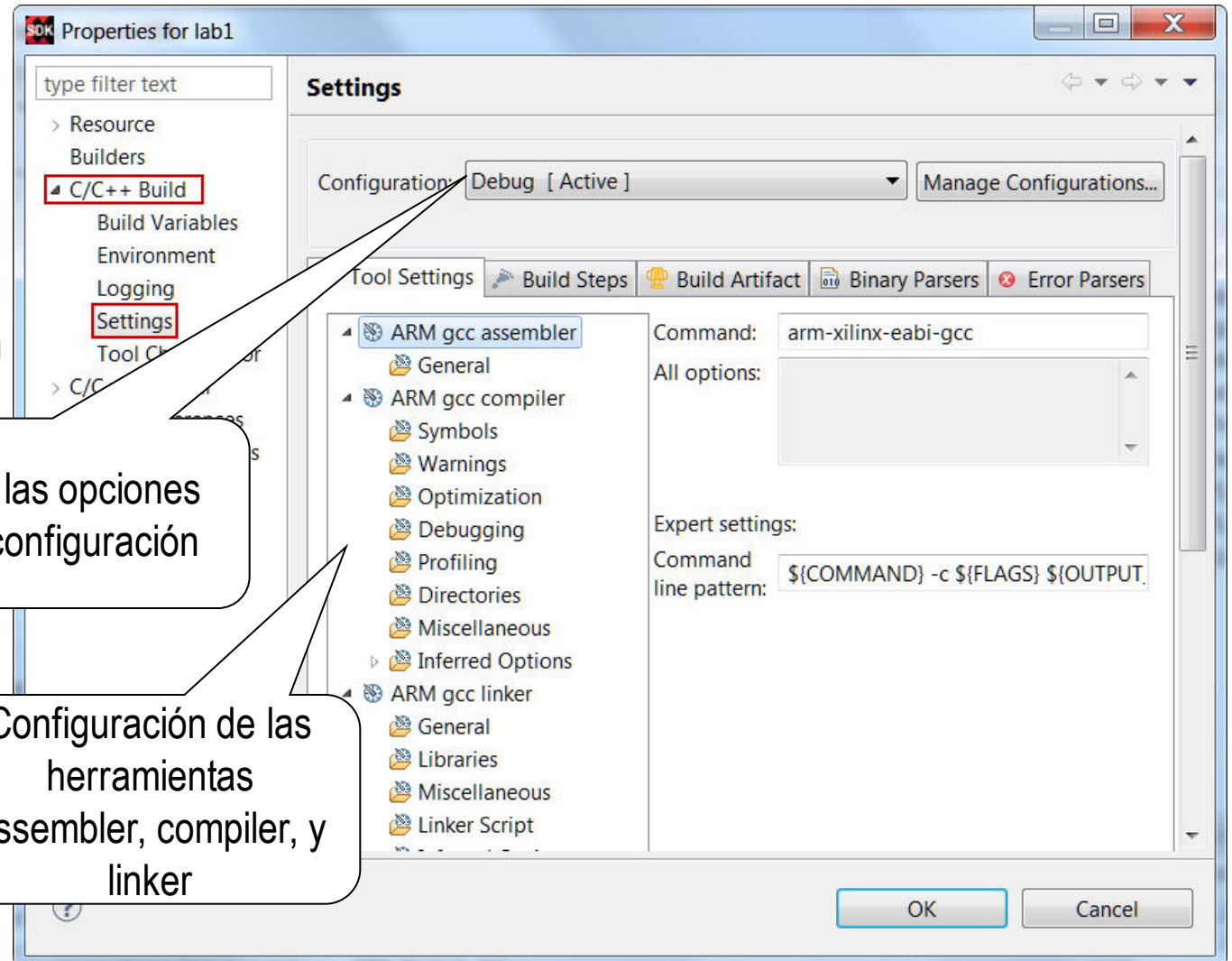
- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- ***Configuración del Software***
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- Resumen

Configuración de C/C++ Build

- Right-click sobre el top level de un proyecto de aplicación y seleccionar **C/C++ Build Settings**
- Las propiedades más accedidas están en la pestaña de configuración del panel **C/C++ Build**
- Cada configuración tiene sus propias propiedades.

Establecer las opciones para esta configuración

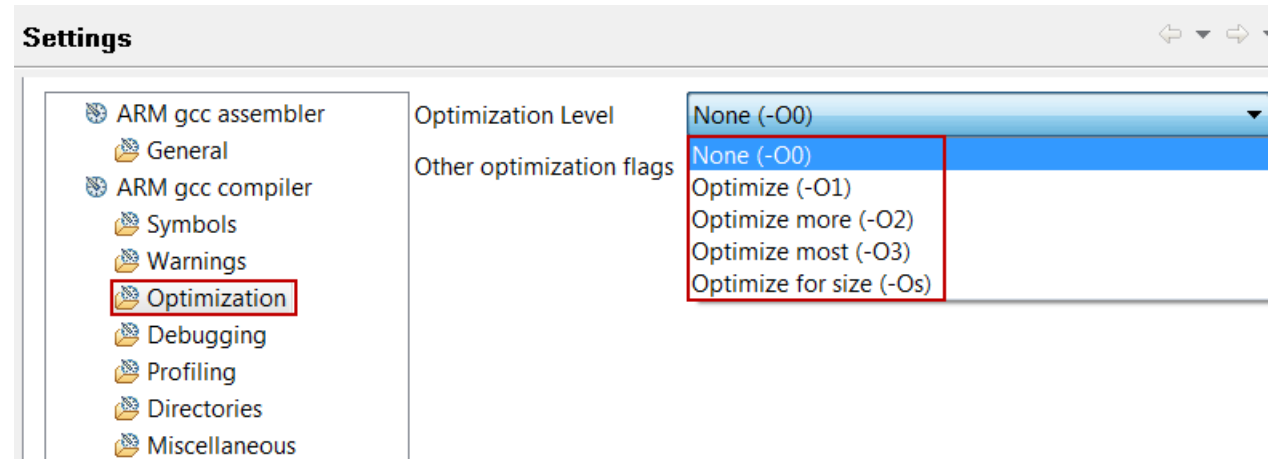
Configuración de las herramientas assembler, compiler, y linker



Propiedades para Depuración/Optimización

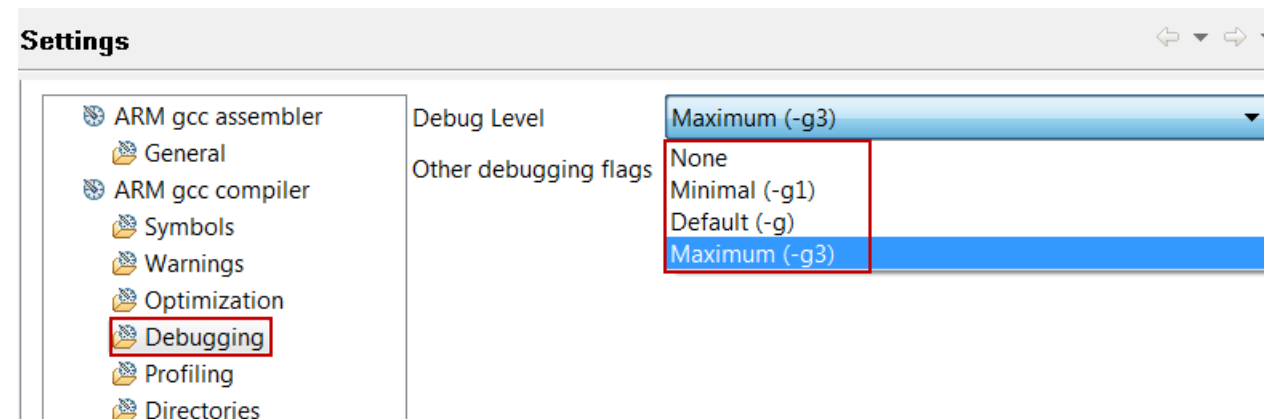
➤ Nivel de optimización del Compilador

- None
- Low
- Medium
- High
- Size Optimized



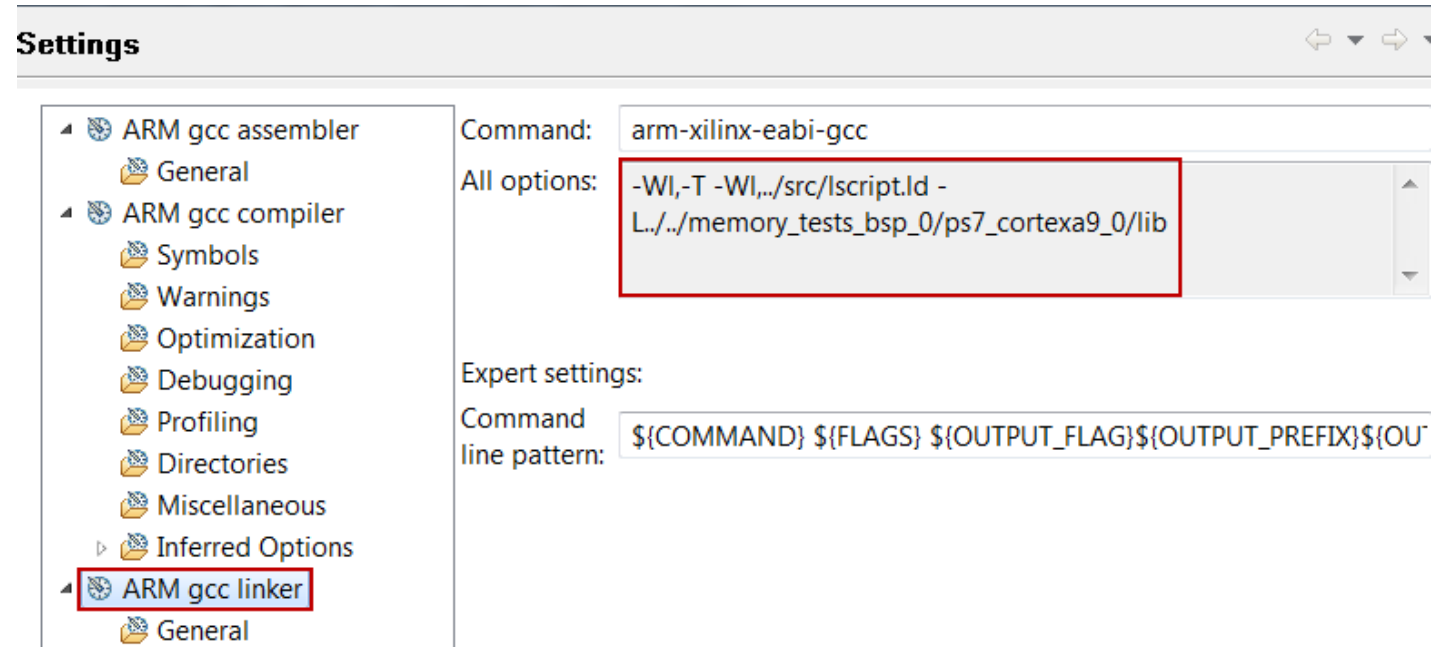
➤ Habilita símbolos de depuración en el ejecutable

- Necesario para depuración



Propiedades del Linker

- En la imagen se pueden ver las opciones del linker para la configuración de Debug
- La configuración por defecto está bien para aplicaciones simples



Temario

- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- ***Manejo de direcciones***
- Secciones de los archivos objeto
- Linker script
- Resumen

Manejo de Direcciones

➤ El diseño de un procesador embebido requiere que uno maneje lo siguiente:

- Mapa de direcciones para los periféricos
- Ubicación del código de la aplicación en el espacio de memoria
 - Block RAM
 - Memoria externa (Flash, DDR3, SRAM)

➤ Los requerimientos de memoria para sus programas están basados en lo siguiente:

- La cantidad de memoria requerida para almacenamiento de instrucciones
- La cantidad de memoria requerida para almacenamiento de datos asociados con el programa

Modelo de Programación Standard ARM

➤ **El sistema de procesamiento y la lógica programable se ven igual**

- Interfaces AMBA® y AXI
- E/S mapeadas a memoria
- Acceso a Registros

➤ **Consistencia para PS y PL = fácil de usar**

➤ **Mapa de memoria usado: total de 4 GB**

- 1 GB: DDR RAM
- 2 GB: dedicado a los periféricos de la PL
- 1 GB: periféricos del PS, OCM, flash externa

Start Address	Size	Description
0x0000_0000	1GB	External DDR RAM
0x4000_0000	2GB	Custom Peripherals (Programmable Logic including PCIe)
0xE000_0000	256MB	PS I/O Peripherals
0xF800_0000	32MB	Fixed Internal Peripherals (Timers, Watchdog, DMA, Interconnect)
0xFC00_0000	64MB	Flash Memory
0xFFFC_0000	256KB	On-Chip Memory

Vista del Programador de la Lógica Programable

➤ Mapa de memoria de la Lógica Programable (PL)

- 2 GB de espacio total
 - 1 GB para cada AXI master: GP0, and GP1
- Accesible desde cualquier sistema de procesamiento (PS) maestro
 - Cualquier Cortex-A9 CPU
 - Motor PS DMA
 - Motor PS peripheral DMA
 - Ethernet
 - USB
 - SD/SDIO

Custom Peripheral

Start Address	Description
0x4000_0000	Accelerator #1 (Video Scaler)
0x6000_0000	Accelerator #2 (Video Object Identification)
0x8000_0000	Peripheral #1 (Display Controller)

Code Snippet

```
int main() {

    int *data = 0x1000_0000;
    int *accel1 = 0x4000_0000;

    // Pure SW processing
    Process_data_sw(data);

    // HW Accelerator-based processing
    Send_data_to_accel(data, accel1);
    process_data_hw(accel1);
    Recv_data_from_accel(data, accel1);
}
```

Mapa de Direcciones: Periféricos E/S (Zynq AP SoC)

Register Base Address	Description
E000_0000, E000_1000	UART Controllers 0, 1
E000_2000, E000_3000	USB Controllers 0, 1
E000_4000, E000_5000	I2C Controllers 0, 1
E000_6000, E000_7000	SPI Controllers 0, 1
E000_8000, E000_9000	CAN Controllers 0, 1
E000_A000	GPIO Controller
E000_B000, E000_C000	Ethernet Controllers 0, 1
E000_D000	Quad-SPI Controller
E000_E000	Static Memory Controller (SMC)
E010_0000, E010_1000	SDIO Controllers 0, 1
E020_0000	IOP Bus Configuration

Mapa de Direcciones: Registros SLCR (Zynq AP SoC)

Register Base Address	Description
F800_0000	SLCR write protection lock and security
F800_0100	Clock control and status
F800_0200	Reset control and status
F800_0300	APU control
F800_0400	TrustZone control
F800_0500	CoreSight SoC debug control
F800_0600	DDR DRAM controller
F800_0700	MIO pin configuration
F800_0800	MIO parallel access
F800_0900	Miscellaneous control
F800_0A00	On-chip memory (OCM) control
F800_0B00	I/O buffers for MIO pins (GPIOB) and DDR pins (DDRIOB)

Mapa de Direcciones : Registros PS (Zynq AP SoC)

Register Base Address	Description
F800_1000, F800_2000	Triple timer counter 0, 1
F800_3000	DMAC when secure
F800_4000	DMAC when non-secure
F800_5000	System watchdog timer (SWDT)
F800_6000	DDR DRAM controller
F800_7000	Device configuration interface (DevC)
F800_8000	AXI_HP 0 high performance AXI interface w/ FIFO
F800_9000	AXI_HP 1 high performance AXI interface w/ FIFO
F800_A000	AXI_HP 2 high performance AXI interface w/ FIFO
F800_B000	AXI_HP 3 high performance AXI interface w/ FIFO
F800_C000	On-chip memory (OCM)
F800_D000	Reserved
F880_0000	CoreSight debug control

Temario

- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- ***Secciones de los archivos objeto***
- Linker script
- Resumen

Secciones de un Archivo Objeto

➤ Qué es un archivo objeto?

- Un archivo objeto es un pedazo de código ensamblado
 - Lenguaje de máquina:
li r31,0 = 0x3BE0 0000
- Datos constantes
- Puede haber referencias a objetos externos que deben ser definidas en otro lugar
- Este archivo puede contener información de depuración

Secciones del Archivo Objeto

Layout de las secciones de un archivo objeto o ejecutable

.text

Sección de Texto

.rodata

Sección de dato de Sólo-Lectura

.sdata2

Sección pequeña de dato de Sólo-Lectura (menos de 8 bytes)

.sbss2

Sección pequeña de dato Sólo-Lectura no inicializado

.data

Sección de dato de Lectura-Escritura

.sdata

Sección pequeña de dato de Lectura-Escritura

.sbss

Sección pequeña de dato no inicializado

.bss

Sección de dato no inicializado

Ejemplo de secciones

```
int ram_data[10] = {0,1,2,3,4,5,6,7,8,9};      /* DATA */

const int rom_data[10] = {9,8,7,6,5,4,3,2,1};   /* RODATA */

int I;     /* BSS */

main() {

    ...
    I = I + 10;  /* TEXT */
    ...

}
```

Temario

- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- ***Linker script***
- Resumen

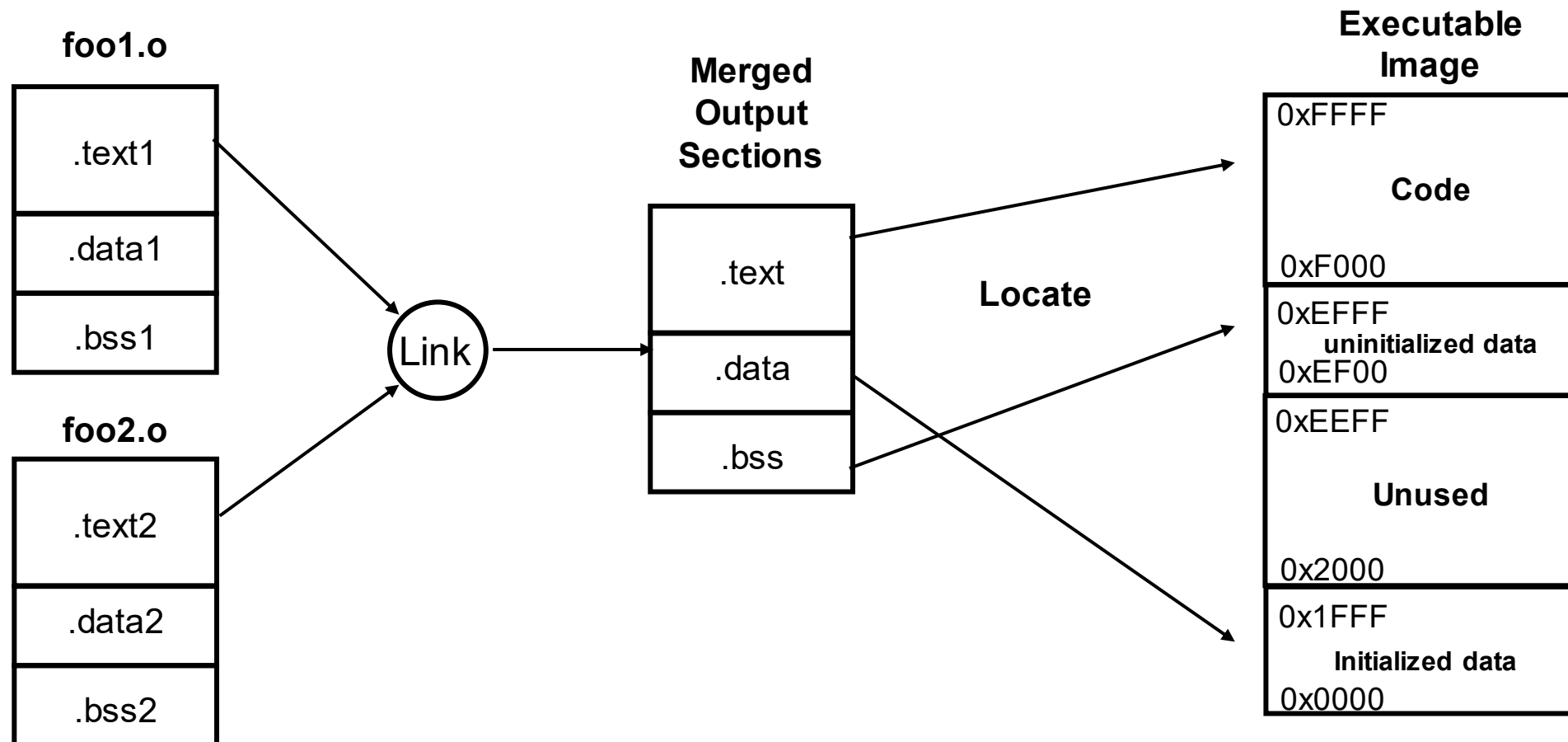
Linker Script

➤ El linker script controla el proceso de linking

- Mapea el código y los datos a un espacio de memoria específico
- Establece el punto de entrada al ejecutable
- Reserva espacio para la pila (stack)

➤ Requerido si el diseño contiene espacio de memoria discontinuo

Flujo del Linker



GUI del Generador del Linker Script

- Una GUI basada en tablas permite definir el espacio de memoria para las secciones de código y dato
- Se lanza desde Xilinx > Generate Linker Script, o desde la perspectiva C/C++, botón-derecho sobre <project> > Generate Linker Script
- La herramienta creará un nuevo linker script (el script viejo es guardado)

The screenshot displays the Xilinx Linker Script Generator GUI. The 'Basic' tab is active, showing options for placing code, data, heap, and stack sections. The 'Advanced' tab is also visible, showing detailed section assignments.

Basic Tab:

- Place Code Sections in: ps7_dds_0_S_AXI_BASEADDR
- Place Data Sections in: ps7_dds_0_S_AXI_BASEADDR
- Place Heap and Stack in: axi_bram_ctrl_0_S_AXI_BASEADDR
- Heap Size: 1 KB
- Stack Size: 1 KB

Advanced Tab:

Code Section Assignments

Section	Assigned Memory
.text	ps7_dds_0_S_AXI_BASEADDR...

Data Section Assignments

Section	Assigned Memory
.rodata	ps7_dds_0_S_AXI_BASEADDR...
.rodata1	ps7_dds_0_S_AXI_BASEADDR...
.sdata2	ps7_dds_0_S_AXI_BASEADDR...
.sbss2	ps7_dds_0_S_AXI_BASEADDR...
.data	ps7_dds_0_S_AXI_BASEADDR...
.data1	ps7_dds_0_S_AXI_BASEADDR...
.fixup	ps7_dds_0_S_AXI_BASEADDR...
.sdata	ps7_dds_0_S_AXI_BASEADDR...
.sbss	ps7_dds_0_S_AXI_BASEADDR...
.bss	ps7_dds_0_S_AXI_BASEADDR...

Heap and Stack Section Assignments

Section	Assigned Memory	Assigned Si...
Heap	axi_bram_ctrl_0_S_AXI_BAS...	1 KB
Stack	axi_bram_ctrl_0_S_AXI_BAS...	1 KB

Hardware Memory Map

Memory	Base Address	Size
ps7_dds_0_S_AXI_BASEADDR...	0x00100000	511...
ps7_ram_0_S_AXI_BASEADDR...	0x00000000	192...
ps7_ram_1_S_AXI_BASEADDR...	0xFFFF0000	~63...
axi_bram_ctrl_0_S_AXI_BAS...	0xBE810000	8 KB

Temario

- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
- ***Resumen***

Resumen

- El desarrollo de software para un sistema embebido en FPGA impone desafíos específicos debido a la plataforma de hardware única.
- SDK provee muchas perspectivas que posibilitan acceso fácil a la información a través de vistas relacionadas.
- Son usadas herramientas GNU para compilar archivos fuente C/C++, para linkear, creando salidas ejecutables, y para depuración.
- La configuración de la plataforma de software permite la inclusión de soporte de librerías de software.
- La configuración del compilador provee switches incluyendo compilación, linking, depuración.

Resumen

- **El diseño de un procesador embebido requiere que uno maneje**
 - El espacio de direcciones de los periféricos
 - El espacio de direcciones de memoria para almacenar datos e instrucciones
 - Bloque de memoria interna
 - Memoria externa
- **Un Linker script se requiere cuando los segmentos de software no residen en un espacio de memoria contiguo**