
Ambiente de Desarrollo de Software

Zynq
Vivado 2018.1

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***

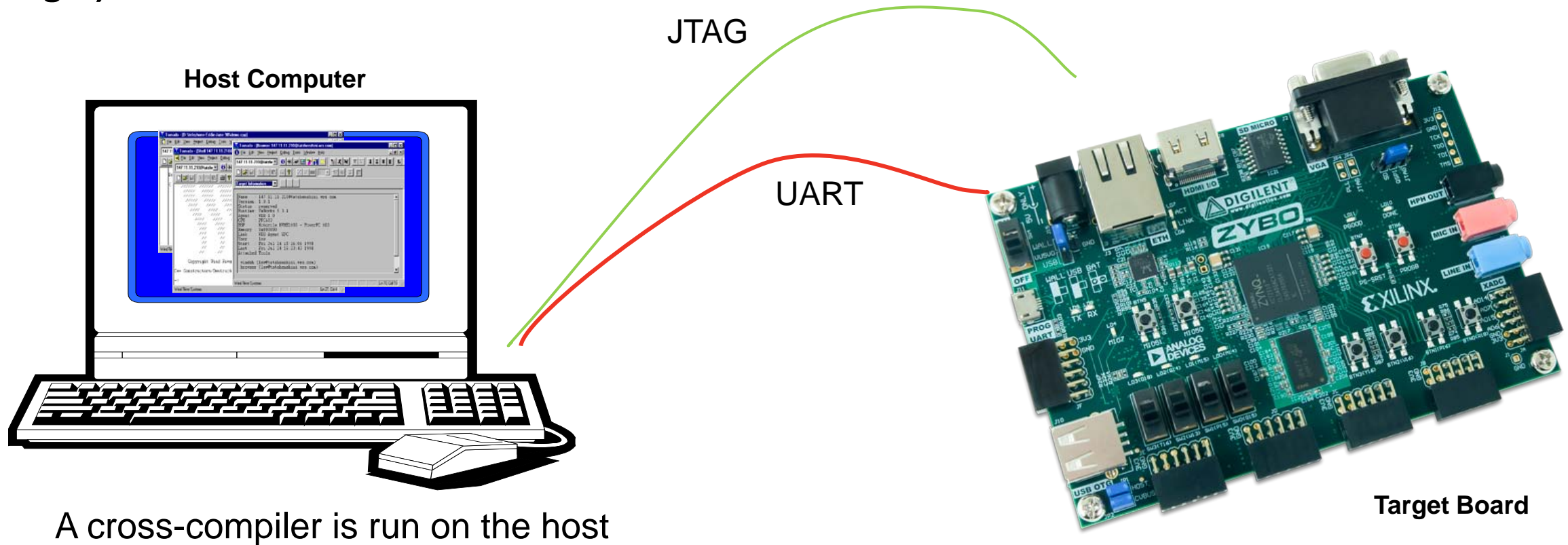
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto
- Linker script
-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

- El hardware es único para cada diseño
- Confiabilidad
- Requerimientos de respuesta en tiempo-real (algunas veces)
 - RTOS versus OS
- Código compacto
- Lenguajes de alto nivel y ensamblado

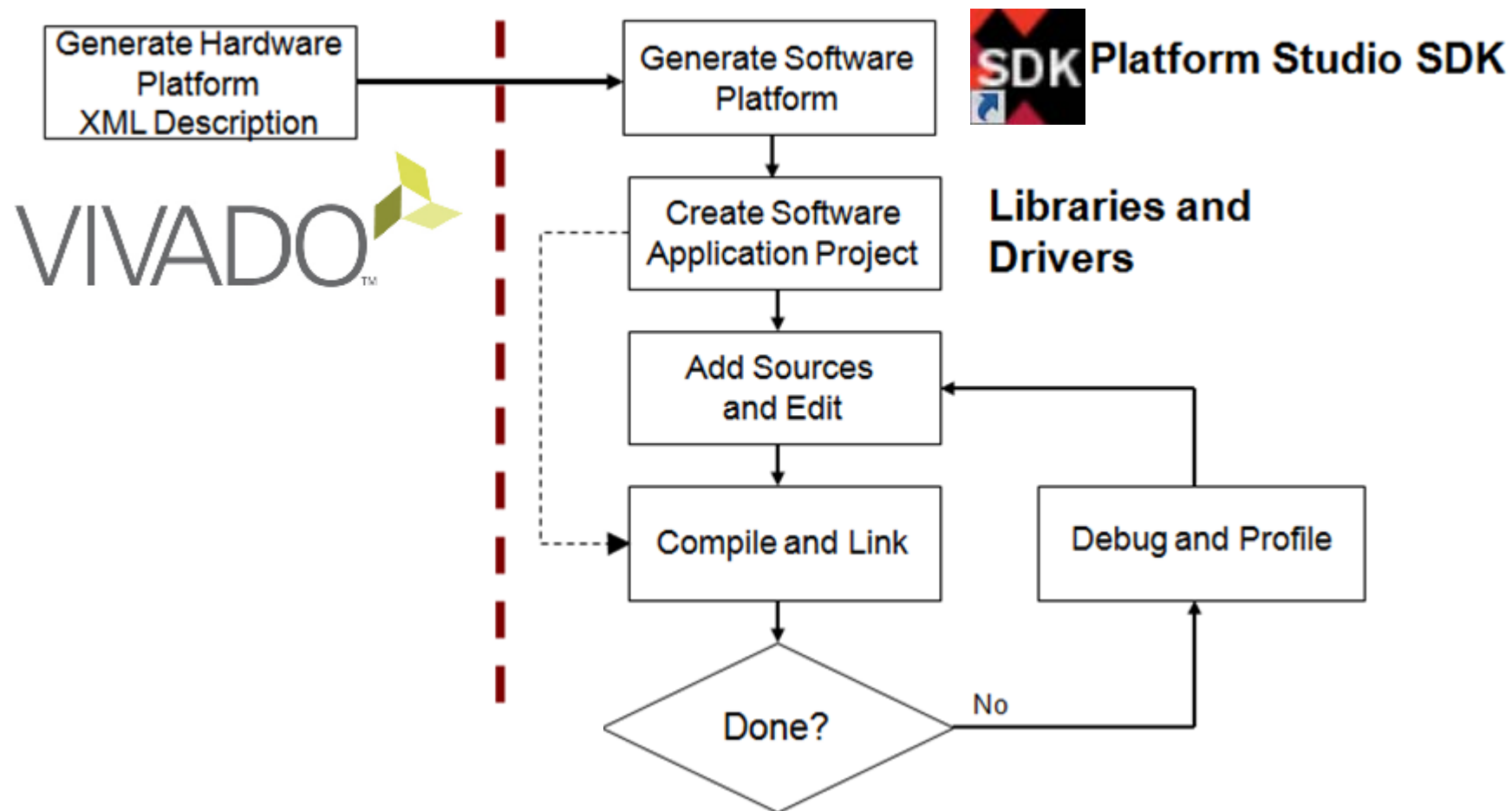
Herramientas de Desarrollo de Software

	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 del Desarrollo de una aplicación en SDK



Frameworks del SDK

➤ Builder framework

- Compila y Linkea archivos fuente
- Se crean opciones de compilación por defecto cuando la aplicación es creada: Elección del Debug, Release, configuraciones de Profile
- El usuario puede personalizar opciones de compilación más tarde cuando desarrolle la aplicación
- Tipos de Build: Standard Make, Managed Make

➤ Launch framework

- Especifica qué acciones son necesarias llevar a cabo: Run (+ Profile) application o Debug application

➤ Debug framework

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

➤ Search framework

- Ayuda al desarrollo de la aplicación

➤ Help System

- Sistema de ayuda en línea; sensible al contexto



Workspaces y Perspectives

➤ 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

➤ Perspectives

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

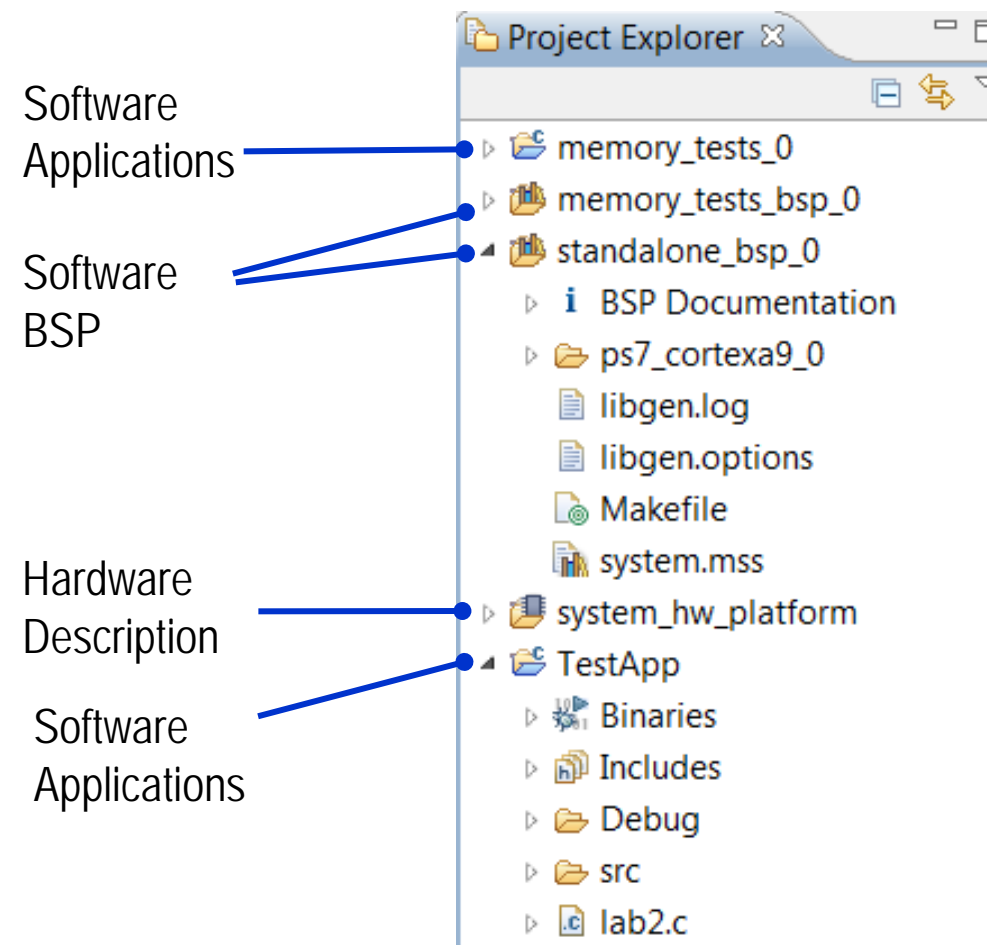
Vistas

- **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]

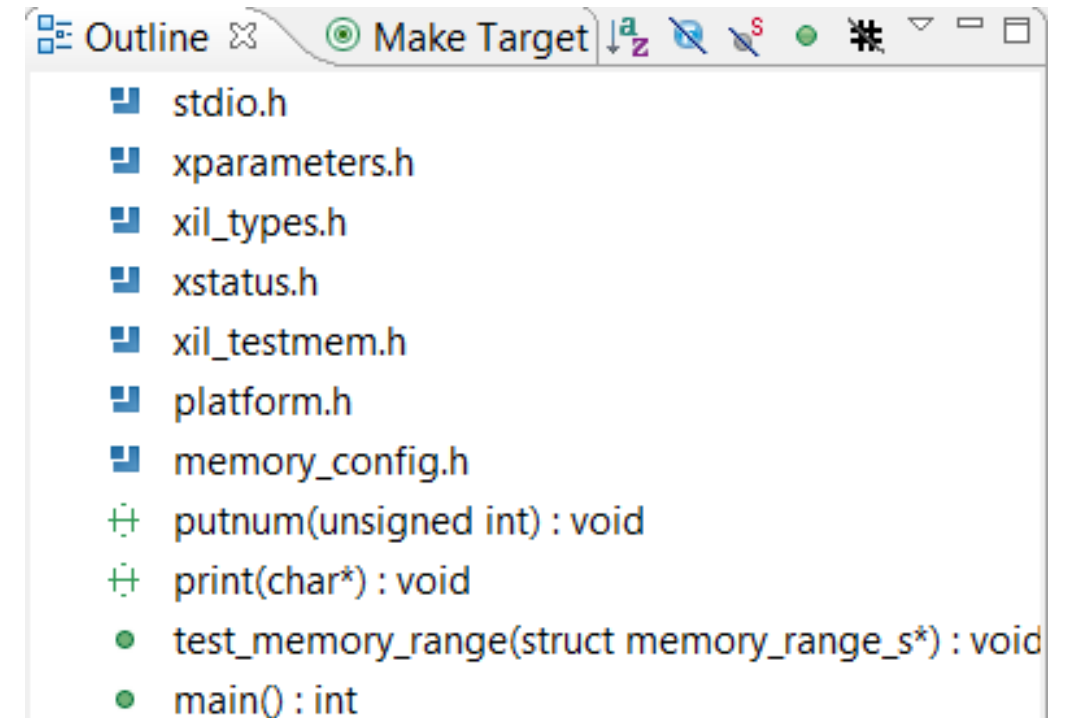
Vista de Proyecto C/C++

- Lista jerárquica de los proyectos
- Doble-click para abrir un archivo
- Hacer botón-derecho sobre el proyecto para acceder a sus propiedades



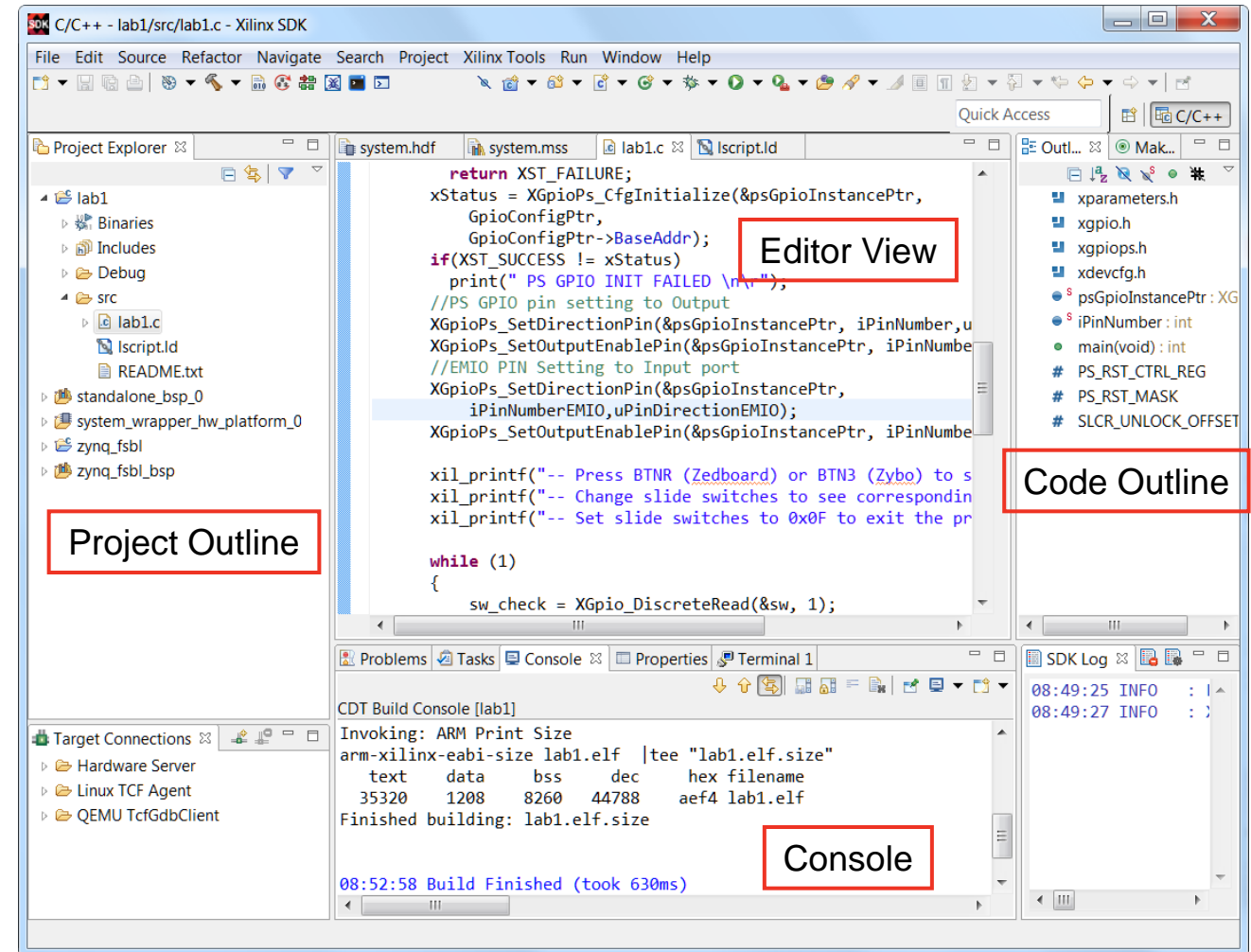
Vista Outline

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



Perspectiva C/C++

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



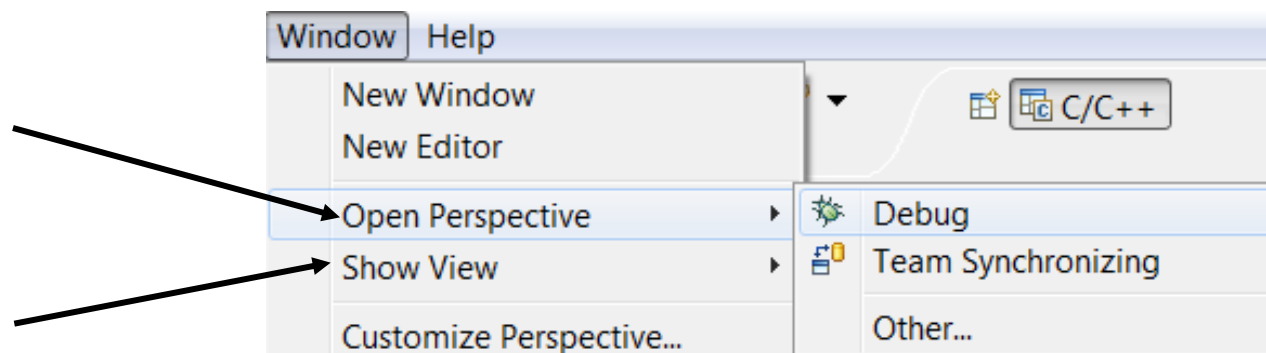
Abriendo Perspectivas y Vistas

➤ Para abrir una Perspective, usar

- Window → Open Perspective

➤ Para abrir una vista, usar

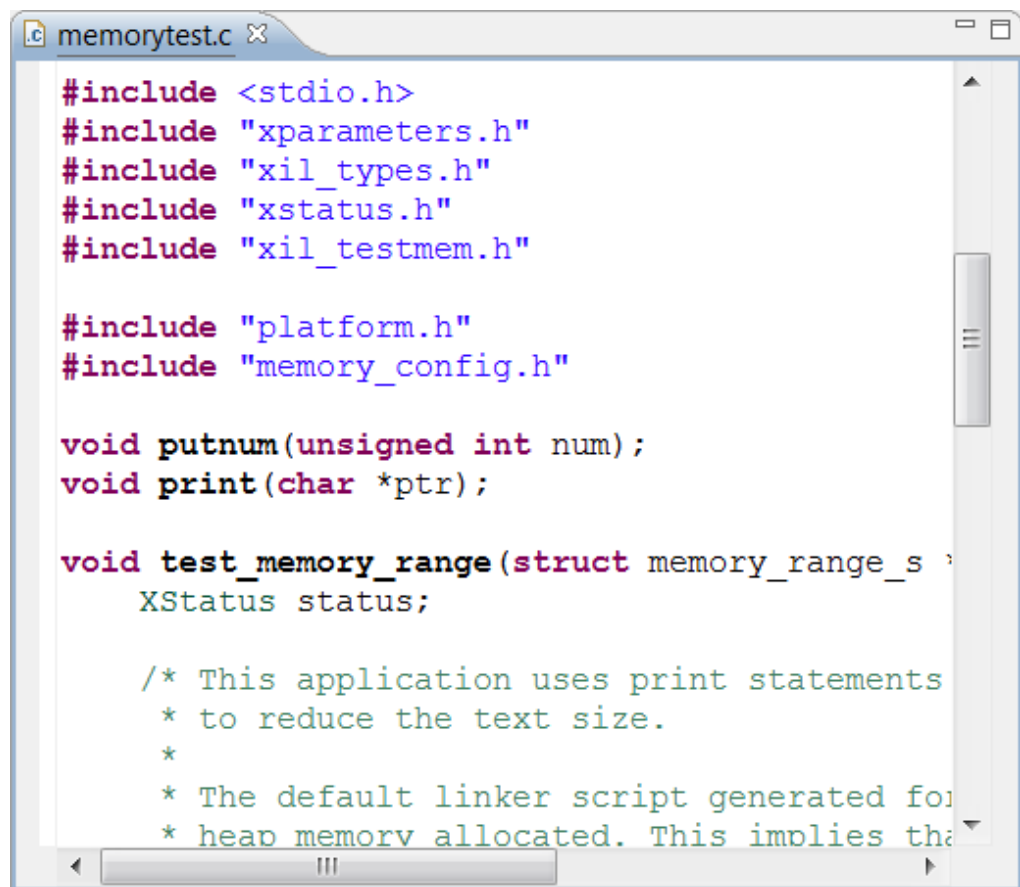
- Window → Show View
- Si la vista está ya presente en la perspectiva actual, es resaltada



Editor

➤ Resultado de Sintáxis

- Matcheo de paréntesis
- Coloreo de sintáxis
- Asistencia de contenido
- Atajos de teclado



```
memorytest.c
#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

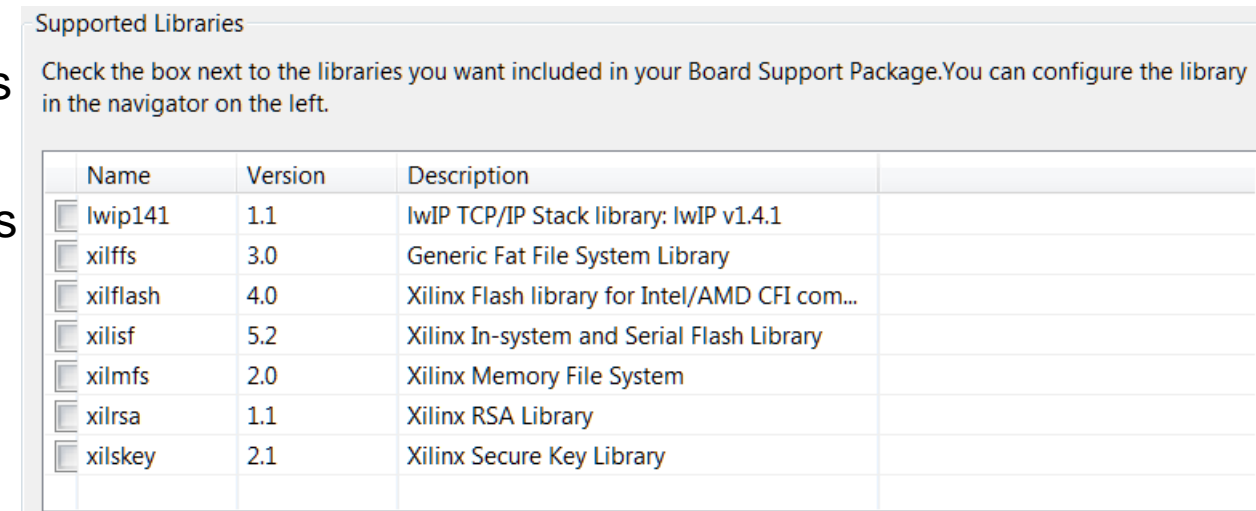
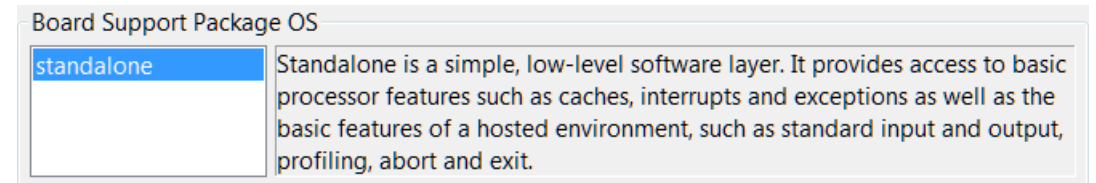
Lanzando el SDK

➤ Launch SDK

- Standalone
 - Choose workspace, choose Hardware Platform Specification
- En Vivado
 - **File> Export Hardware**
 - **File > Launch SDK**
- Exportando
 - Un archivo de Descripción de Hardware HDF es primeramente generado
 - Un proyecto de especificación de plataforma de hardware es entonces automáticamente creado
 - La aplicación de software (y el board support package) entonces puede ser creada y asociada con la plataforma de hardware

Creando un Board Support Package

- El Board Support Package provee servicios de software basados en procesador y periféricos que constituyen el sistema de procesamiento
- Puede ser creado automáticamente 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 apropiado para OS
 - Son soportados sistemas operativos de terceras partes con el apropiado BSP
 - Seleccionar soporte para las librerías requeridas



Creando un Proyecto de Aplicación de Software

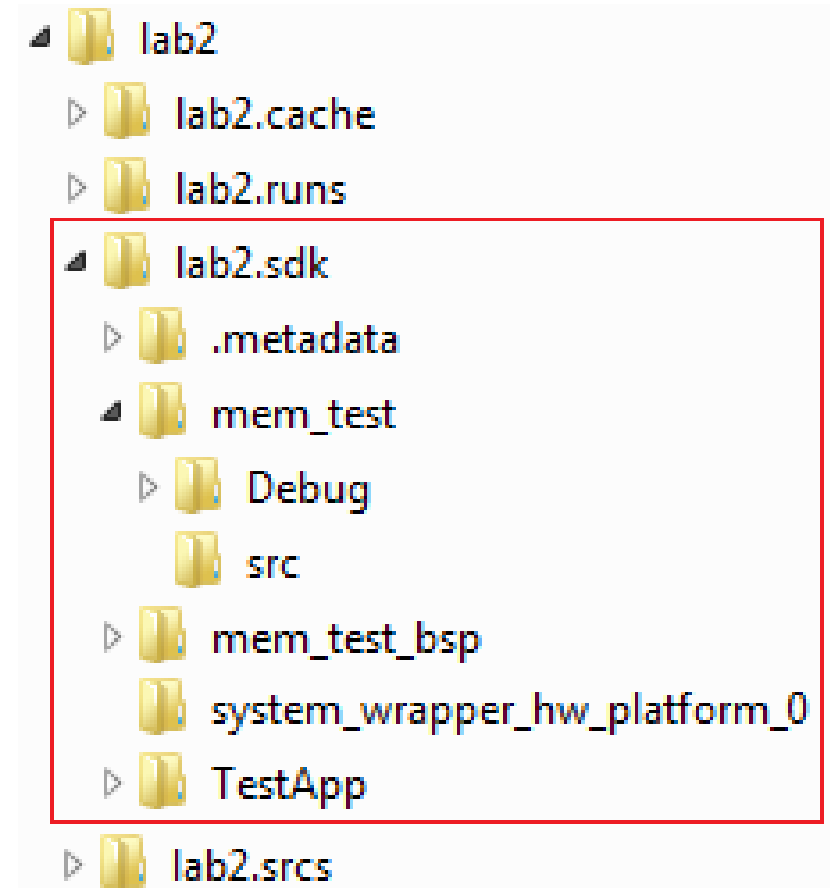
- **SDK soporta múltiples proyectos de aplicación de software**
- **Un proyecto de software es asociado a un proyecto BSP**
- **Se proveen aplicaciones de ejemplo**
 - Muy bueno para pruebas rápidas de hardware
 - Pruebas de periféricos
 - Punto de inicio en el que basar su propia aplicación
- **Típicamente es abierta una aplicación vacía para iniciar un proyecto no-standard**

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

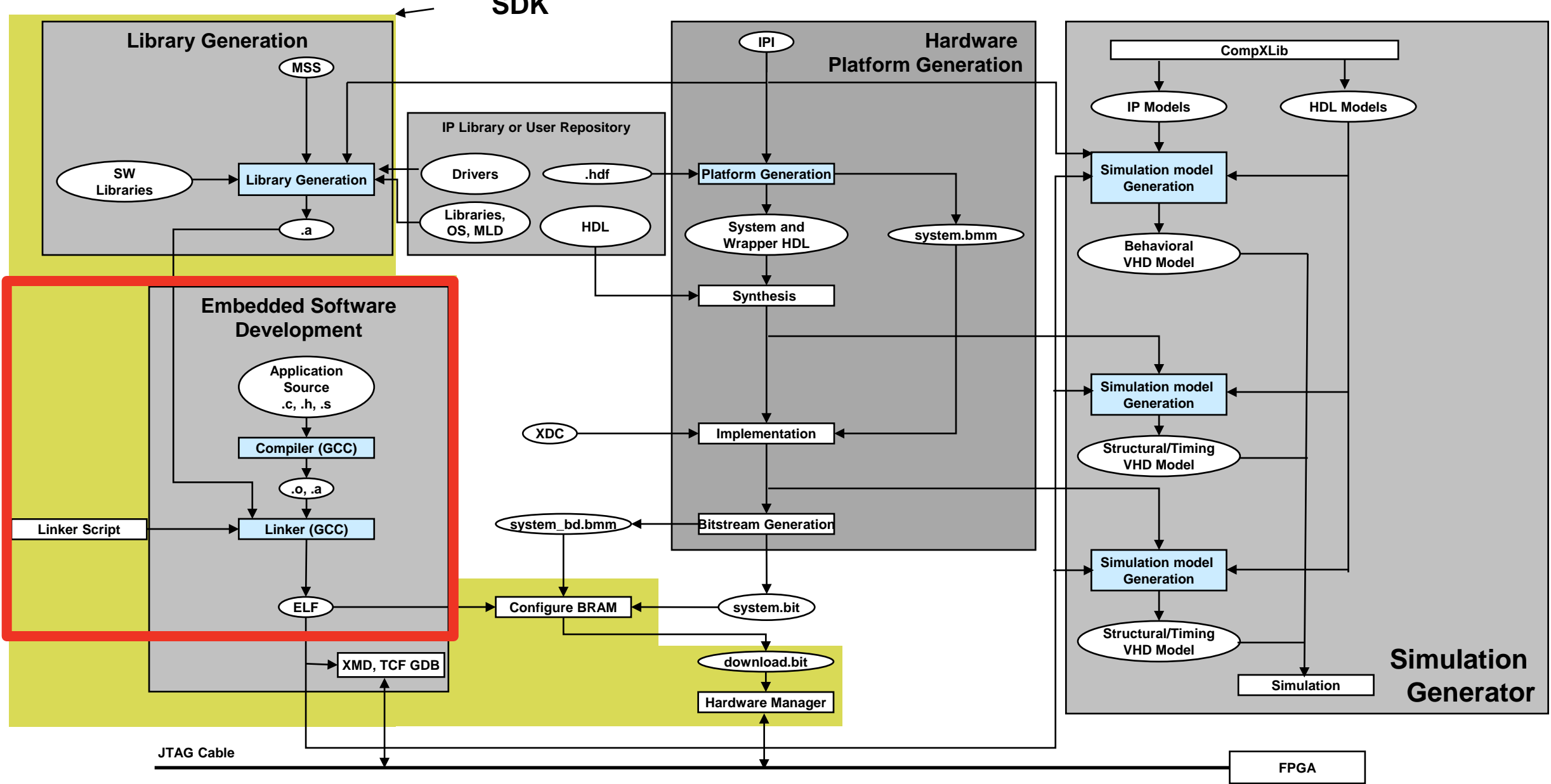
- Los proyectos SDK se sitúan en el directorio de aplicación que fue especificado cuando SDK fue lanzado
- Cada proyecto puede tener múltiples directorios para archivos de sistema y configuraciones
- A Debug configuration is created by default



Temario

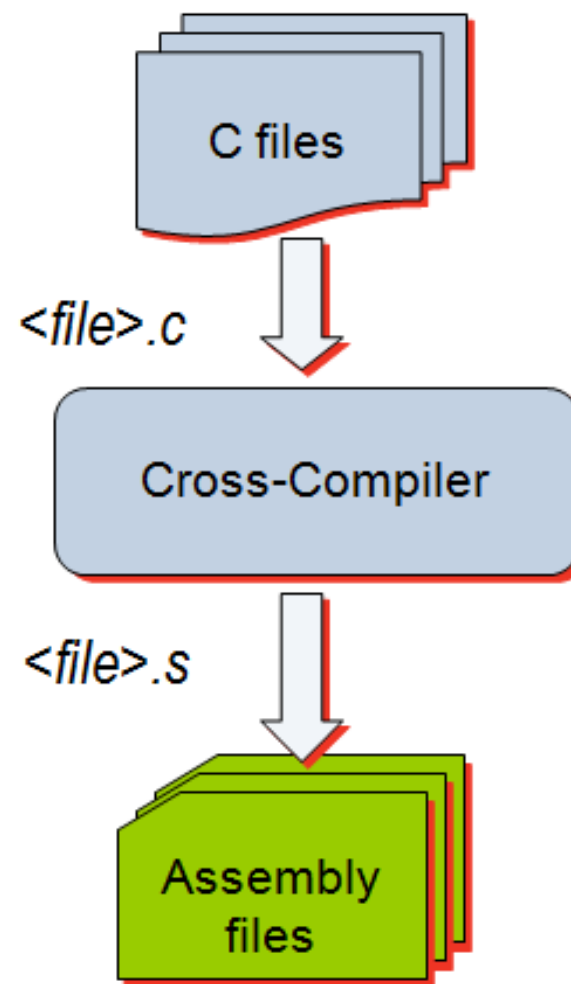
- Introducción
- Ambiente de Desarrollo SDK
- Creación de un proyecto en SDK
- ***Herramientas de Desarrollo GNU: GCC, AS, LD, Binutils***
- Configuración del Software
 - Configuración de la plataforma de software
 - Configuración de compilación
- Manejo de direcciones
- Secciones de los archivos objeto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

Embedded Tool Flow (SDK)



Herramientas GNU: GCC

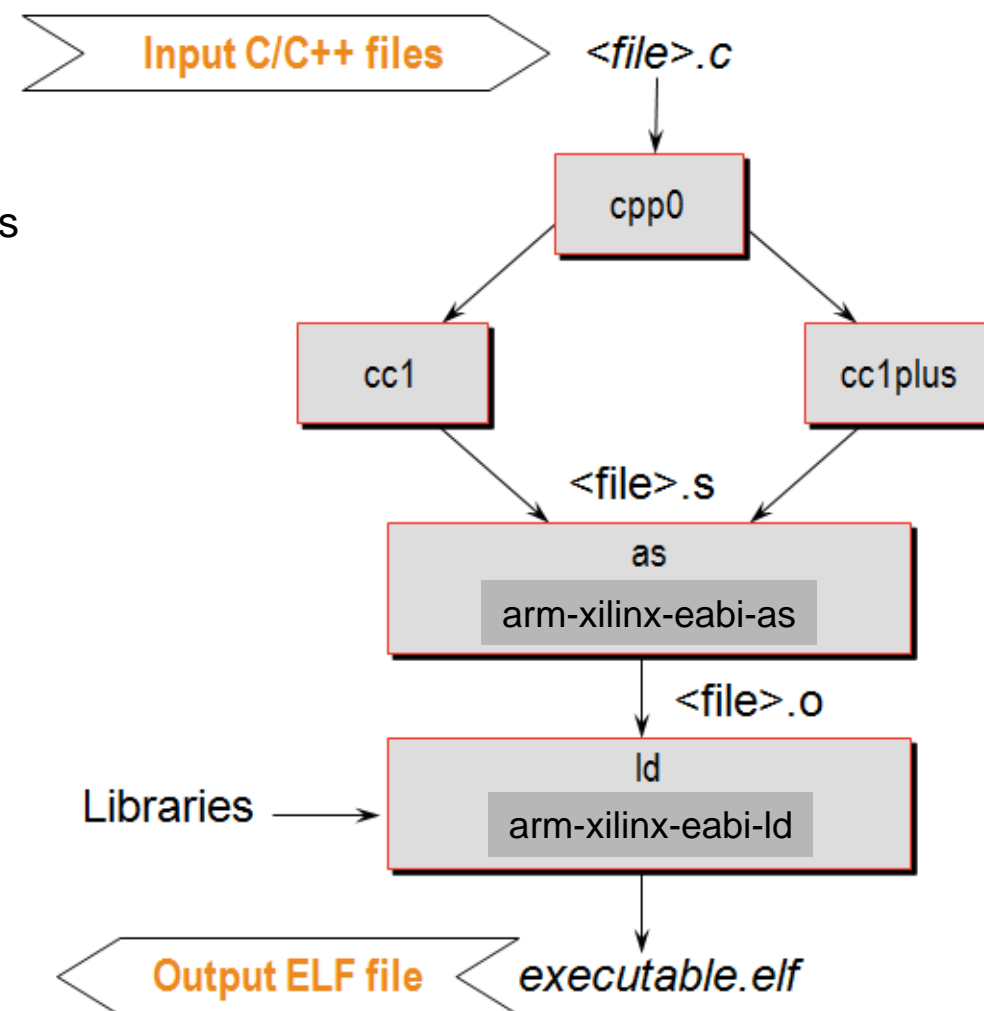
- GCC traduce código fuente C en lenguaje assembly
- GCC también funciona como la interfaz de usuario al assembler y al linker, llamando al ensamblador y al linker con los parámetros apropiados
- Cross-compiladores soportados:
 - GNU GCC (arm-xilinx-eabi-gcc)



Herramientas GNU: GCC

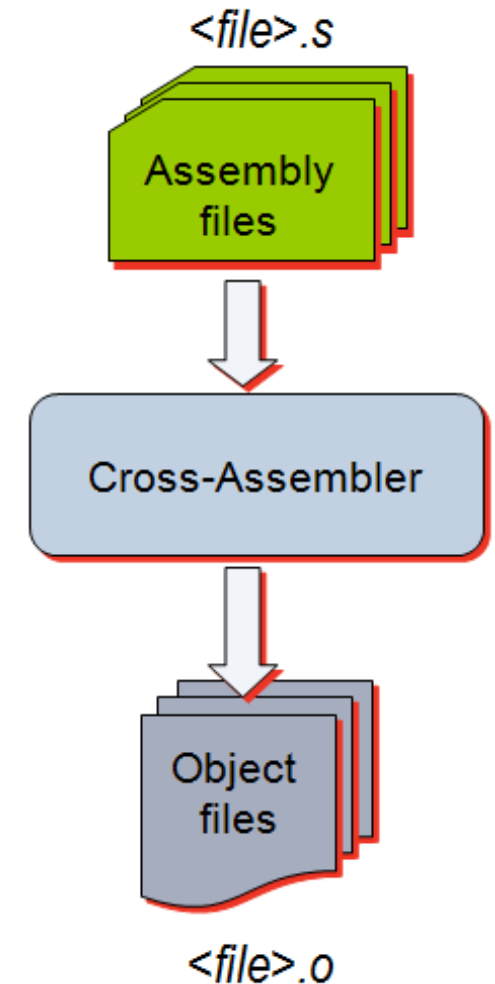
➤ Llamada a 4 ejecutables diferentes

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



Herramientas GNU: AS

- **Input: Archivos en lenguaje Assembly**
 - Extensión: `.s`
- **Salida: Código objeto**
 - Extensión: `.o`
 - Contiene
 - Pedazo de código ensamblado
 - Dato Constante
 - Referencias externas
 - Información de depuración
- **Usualmente el compilador llama automáticamente al ensamblador**
- **Usar el switch `-Wa` si los archivos fuente sólo son ensamblados y se quiere usar el gcc**



Herramientas GNU: LD

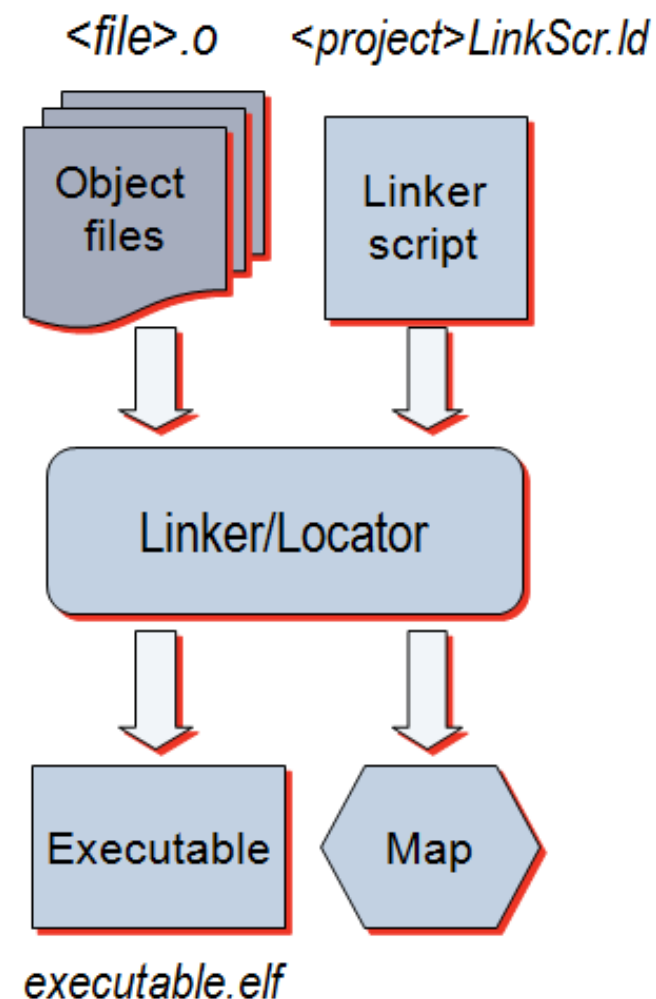
➤ Linker

➤ Entradas:

- Varios archivos objeto
- Archivos objeto archivados (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 de diferentes librerías

➤ Object Dump

- Muestra información de archivos objeto y ejecutables
 - Información de Header, mapa de memoria
 - Dato
 - 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      VMA       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  .data           00000000  001019af0  001019af0  000099af0  2**2
CONTENTS, ALLOC, LOAD, DATA
 5  .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
```

Section
Name

Section Size

Virtual
Memory
Address

Loadable
Memory
Address

Byte alignment

Offset from the beginning
of the section header table

Object Dump

Volcado del código fuente y ensamblado

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

Memory
location

Machine Language
Instruction

C code
instruction

Assembly
instruction

```
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);
```

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

Servicios Requeridos Mínimos

➤ Servicios estandar del lenguaje C

- *stdin* y *stdout*
- Librería Math
- *malloc*

➤ Soporte de Procesador requiere estos servicios

- Interrupción
- Cache
- Soporte de entorno de lenguaje

Sistemas Operativos

- **Operating systems are a collection of software routines that comprise a unified and standard set of system services**
- **The Standalone option is used when no operating system is desired**
 - Provides a minimal amount of processor and library services as previously illustrated
 - Can be considered a minimal, non-standard operating system
 - Installed as a software platform
- **Variety of third-party operating systems are available**
 - Linux – many flavors
 - RTOS – real-time operating system; also has many flavors; Free RTOS (an option for the Cortex™-A9 processor)
 - XilKernel – provided by Xilinx; small and simple; only for MicroBlaze
- **Operating systems are installed and become part of the Board Support Package (BSP)**

¿Qué provee un Sistema Operativo?

➤ Servicios de un Sistema Operativo

- GUI support
- TCP/IP services**
- Task management
- Resource management**
- Familiar programming services and tasks
- Easy connection to already written applications
- Ability to reload and change applications
- Full file system services**

** Also available as additions to the Standalone BSP

¿Necesito un Sistema Operativo?

- **The Standalone BSP includes the previously discussed items**
- **Design considerations for systems using the Standalone BSP**
 - All services needed are included in the BSP
 - The application is static—it never changes
 - The application fits in block RAM (MicroBlaze™ processor), OCM RAM (Zynq™ AP SoC), or DDR memory
 - The application is single-task based
 - Interrupts may or may not be used

Accediendo las Propiedades de la Plataforma de Software

- Select the created board support package in the Project Explorer view
- Xilinx Tools > Board Support Package Settings
- Sets all of the software BSP related options in the design
- Has multiple forms selection
 - Overview
 - Standalone
 - Drivers
 - CPU
- As individual Standalone services are selected a configurable menu selection item will appear

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

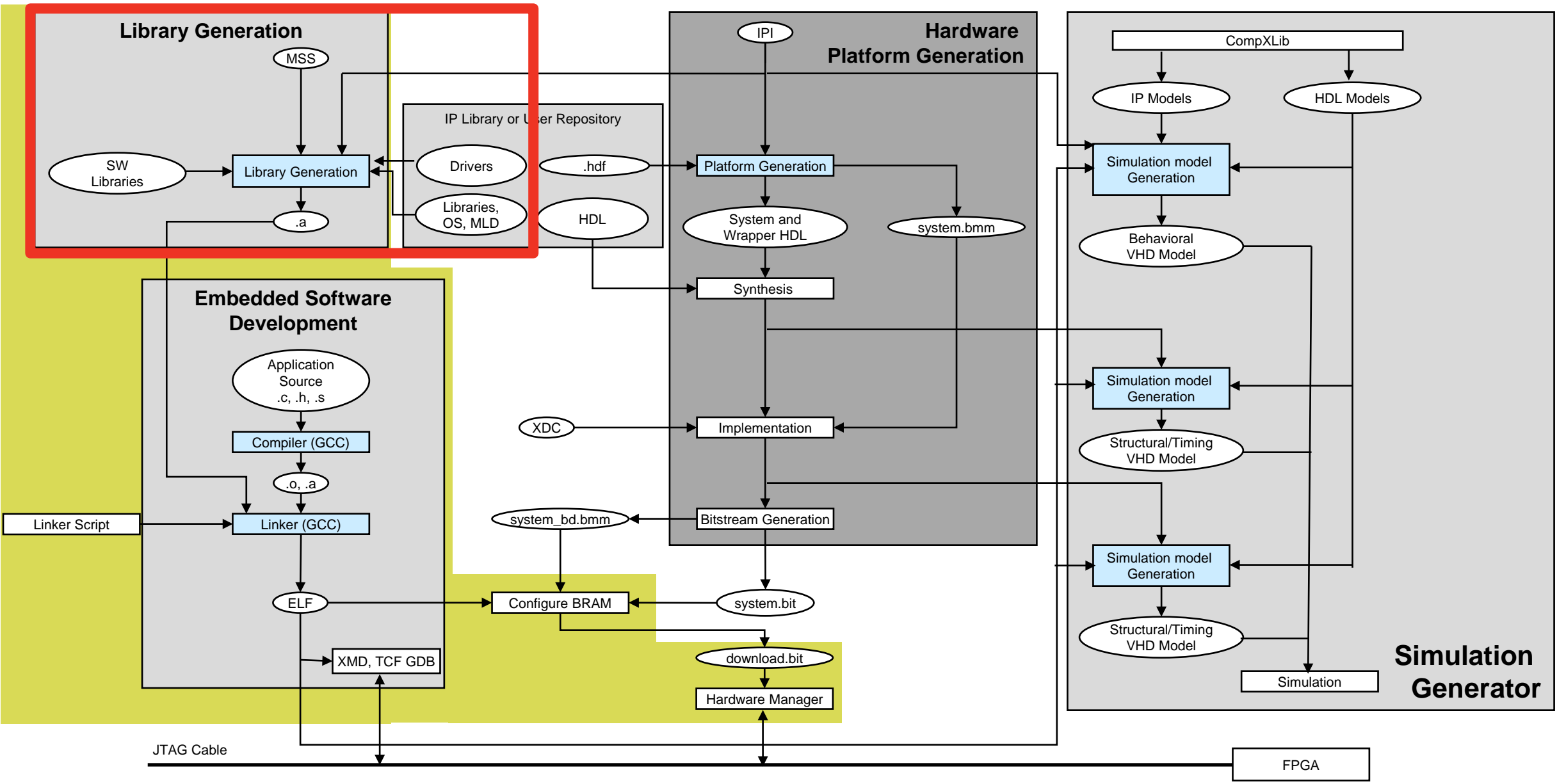
Processor: ps7_cortexa9_0

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"/> xilisf	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"/> xilskey	2.1	Xilinx Secure Key Library

Flujo de la Herramienta Embebida (SDK)



Flujo de la Generación de la Librería (en SDK)

➤ Input files → MSS

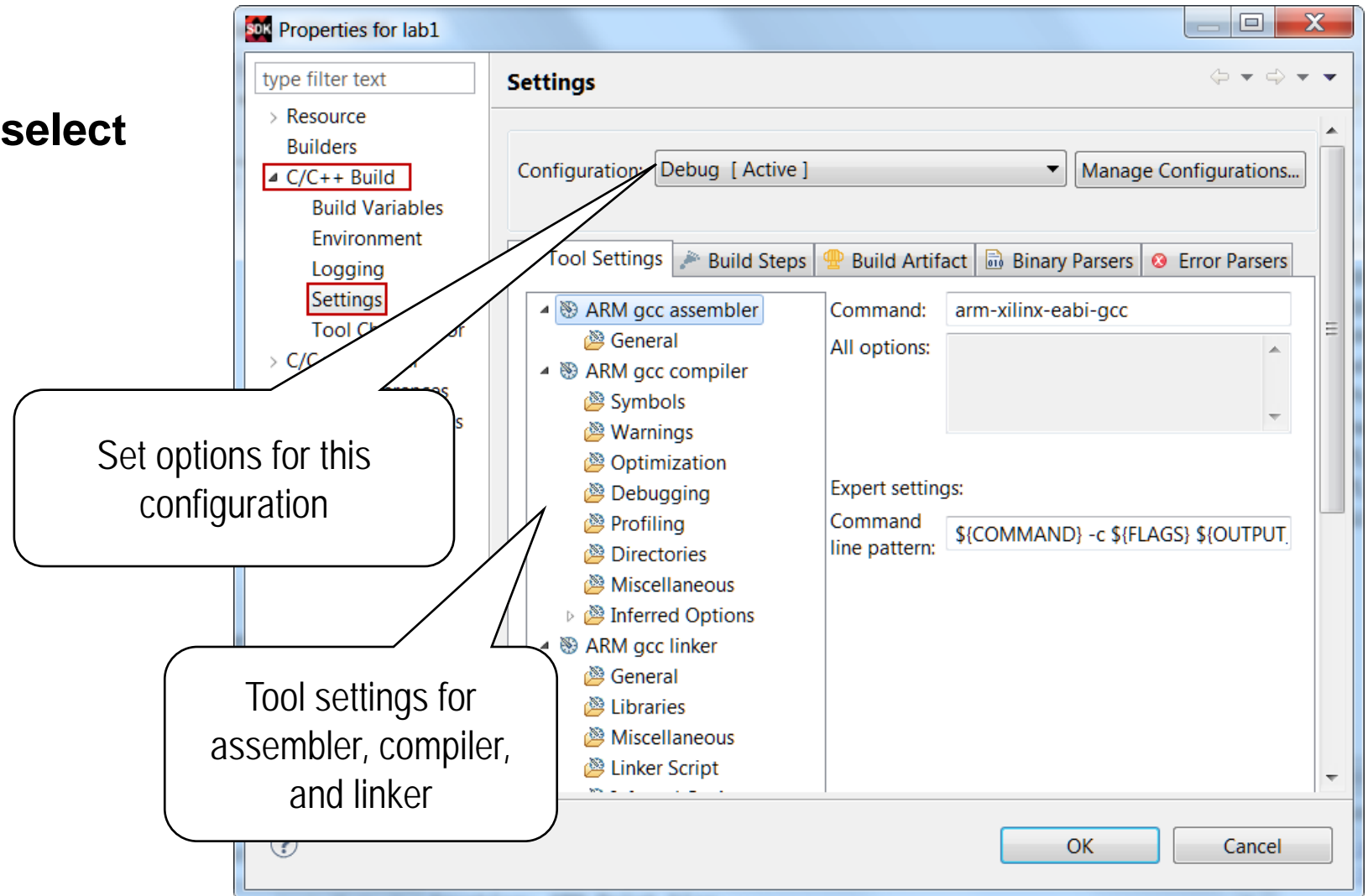
- Output files → libc.a, libXil.a, libm.a
- Library generator is generally the first tool run to configure libraries and device drivers
 - The MSS file defines the drivers associated with peripherals, standard input/output devices, and other related software features
- Library generator configures libraries and drivers with this information and produces an archive of object files:
 - libc.a - Standard C library
 - libXil.a - Xilinx library
 - libm.a - Math functions library

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

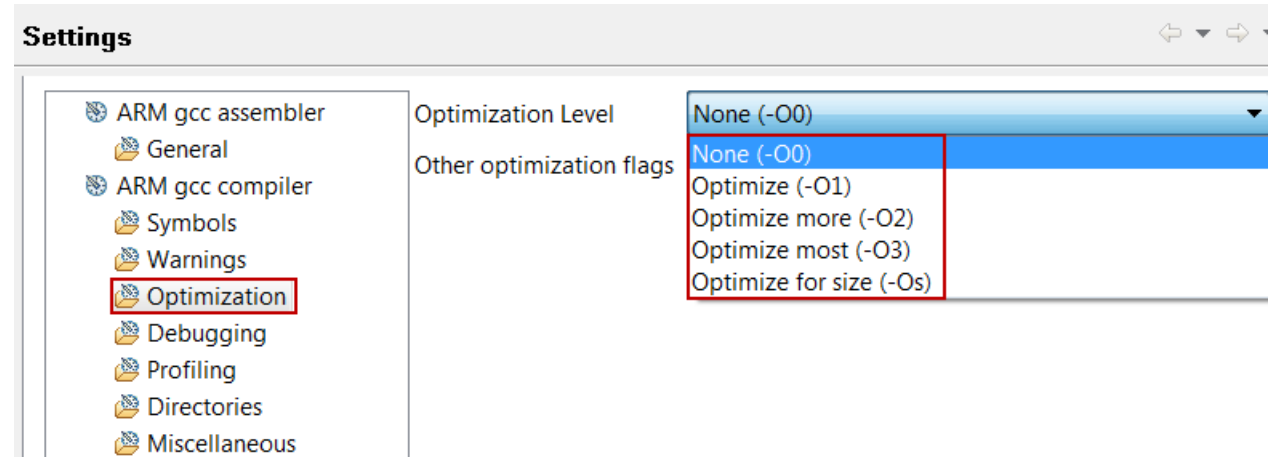
- Right-click the top level of an application project and select **C/C++ Build Settings**
- Most-accessed properties are in the **C/C++ Build** panel **Settings** tab
- Each configuration has its own properties



Propiedades para Depuración/Optimización

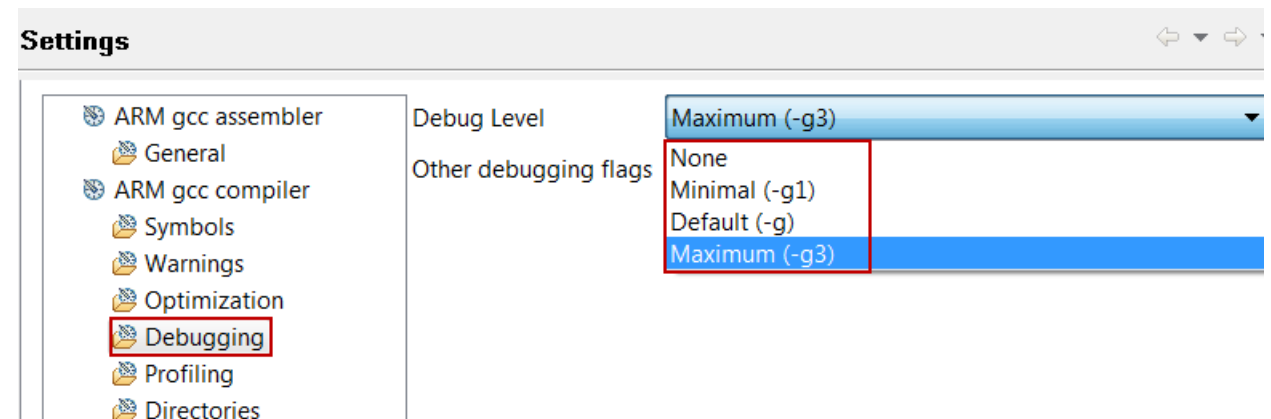
➤ Nivel de optimización del Compilador

- None
- Low
- Medium
- High
- Size Optimized



➤ Habilita símbolos de depuración en el ejecutable

- Necesario para depuración
- Establecer el nivel de optimización a none si es posible



Propiedades Varias del Compilador

➤ Define symbols para compilación condicional

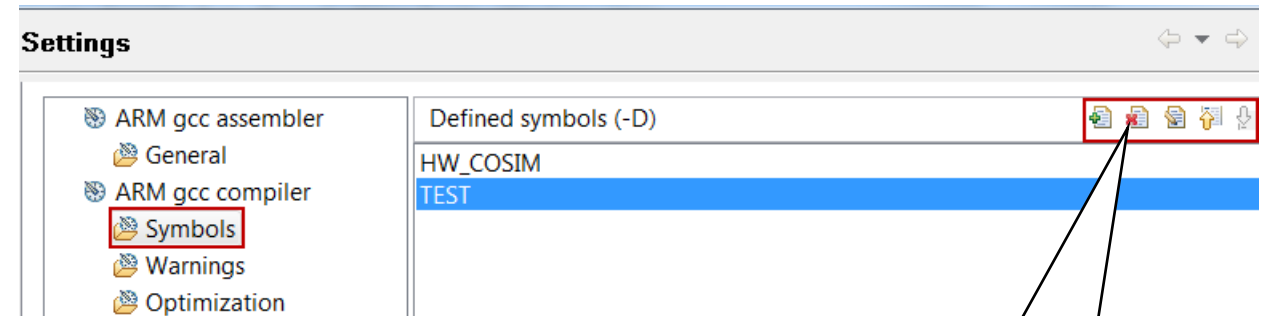
- Add
- Delete
- Edit

➤ References C source

```
#ifdef symbol  
conditional statements  
#endif
```

➤ Pasado al compilador como –*D* opcion

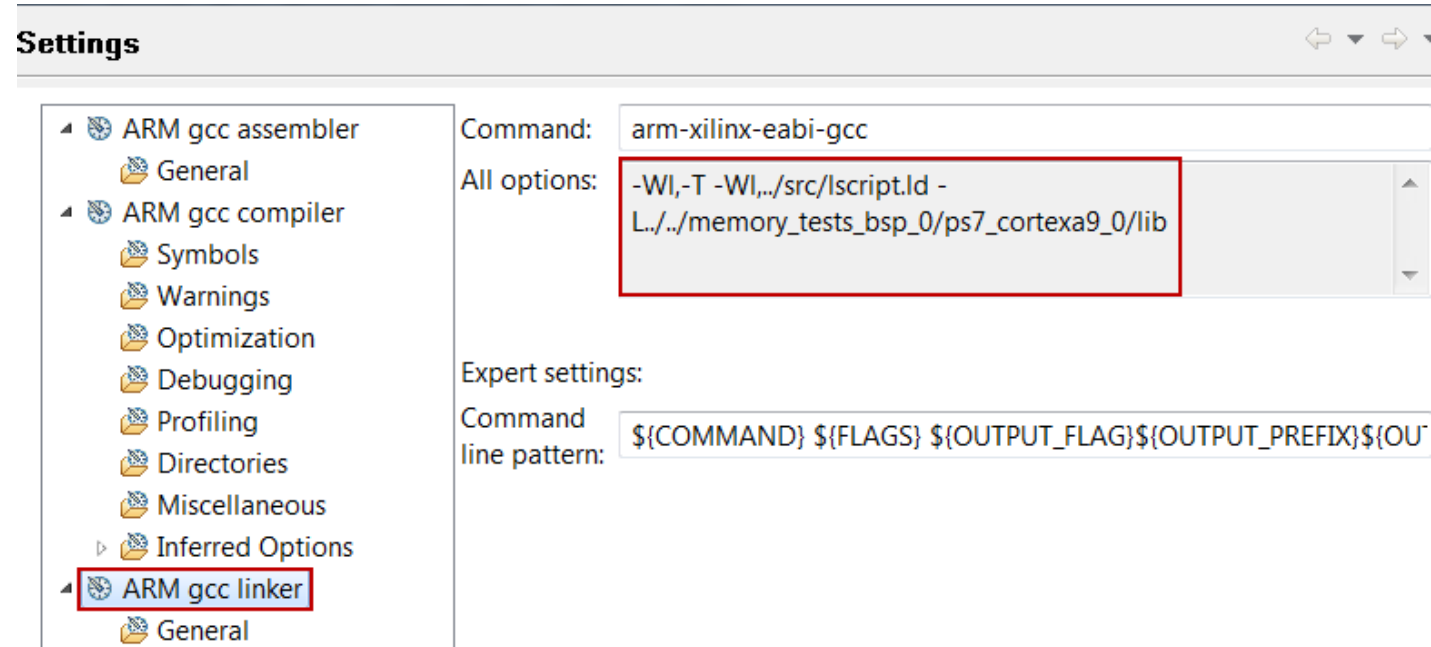
➤ Están disponibles otras opciones de compilación



Add, Delete, Edit
Icons

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 data asociada con el programa

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 delas 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 */
...

}
```

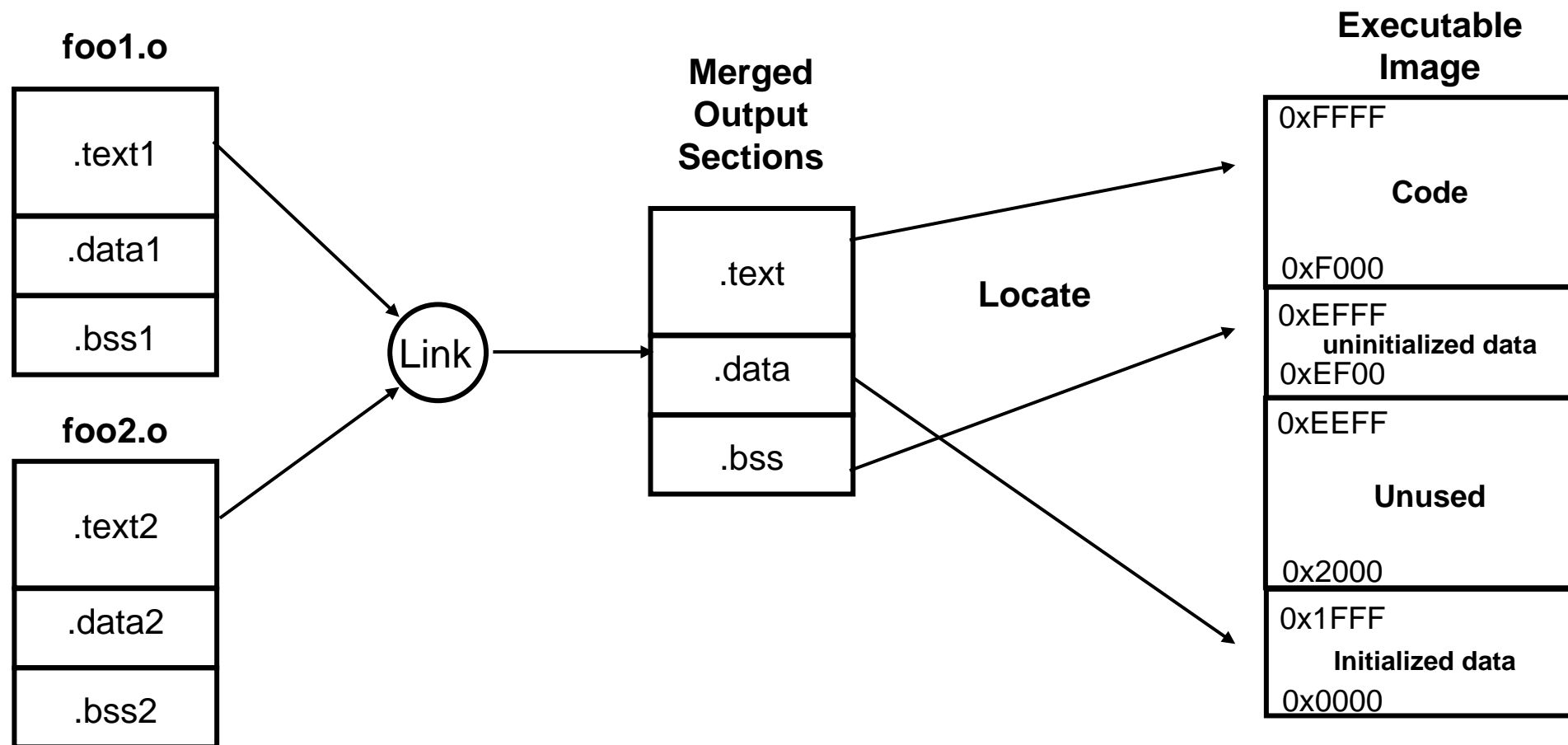
Temario

- 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 le 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)

Basic Advanced

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

Output Settings

Project: TestApp

Output Script: bs_1\lab4\lab1.sdk\SDK\SDK_Export\TestApp\src\lscript.ld Browse

Modify project build settings as follows:

Set generated script on all project build configurations

Hardware Memory Map

Memory	Base Address	Size
ps7_dds_0_S_AXI_BASEAD...	0x00100000	511...
ps7_ram_0_S_AXI_BASEAD...	0x00000000	192...
ps7_ram_1_S_AXI_BASEAD...	0xFFFF0000	~63...
axi_bram_ctrl_0_S_AXI_BAS...	0x8E810000	8 KB

Fixed Section Assignments

Basic Advanced

Code Section Assignments

Section	Assigned Memory
.text	ps7_dds_0_S_AXI_BASEAD...

Add Section Remove Section

Data Section Assignments

Section	Assigned Memory
.rodata	ps7_dds_0_S_AXI_BASEAD...
.rodata1	ps7_dds_0_S_AXI_BASEAD...
.sdata2	ps7_dds_0_S_AXI_BASEAD...
.sbss2	ps7_dds_0_S_AXI_BASEAD...
.data	ps7_dds_0_S_AXI_BASEAD...
.data1	ps7_dds_0_S_AXI_BASEAD...
.fixup	ps7_dds_0_S_AXI_BASEAD...
.sdata	ps7_dds_0_S_AXI_BASEAD...
.sbss	ps7_dds_0_S_AXI_BASEAD...
.bss	ps7_dds_0_S_AXI_BASEAD...

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

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 únicos debido a la plataforma de hardware única
- SDK provee muchas perspectivas ricas 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
 - 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**