Building a MicroBlaze System Using the EDK

Introduction

This lab will illustrate the Embedded Development Kit (EDK) flow. The lab design is partially completed. You will complete the MHS file, C application code, and linker script. You will also correct an application code error.

Objectives

After completing this lab, you will be able to:

- Design a hardware processor system including the MicroBlaze™ soft processor core and the associated IP
- Design a software application used to exercise the processor system
- Modify the software application and quickly update the hardware bit file using Data2BRAM
- Debug the software application using GNU Debugger (GDB) and Xilinx Microprocessor Debugger (XMD)

Design Description

When designing any embedded processor system you need the following items:

- Required hardware
- Memory map of the system
- And the software application

This EDK lab example consists of the following hardware:

- MicroBlaze soft processor core
- LMB Bus
  - LMB_LMB_BRAM_IF_CNTLR
  - BRAM_BLOCK
- OPB BUS
  - OPB_GPIO
  - OPB_BRAM_IF_CNTLR
  - OPB BRAM
  - OPB_UARTLITE

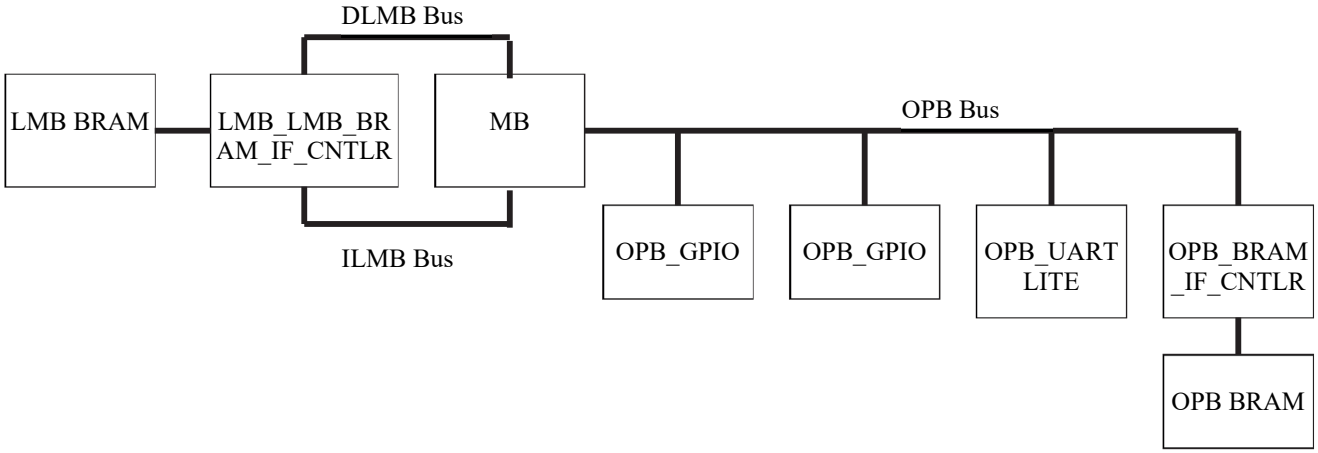

Figure 3-1. Design Layout.

Below is the memory map for this design

Device	Address		Size	Comment
	Min	Max		
LMB_BRAM	0x0000_0000	0x0000_1FFF	16kB	LMB Memory
OPB_GPIO	0xFFFF_4200	0xFFFF_42FF	256B	DIP Switch Input
OPB_GPIO	0xFFFF_4100	0xFFFF_41FF	256B	LED Output
OPB_UARTLITE	0xFFFF_4000	0xFFFF_40FF	256B	Serial Output
OPB_BRAM	0xFFFF_0000	0xFFFF_3FFF	16kB	OPB Memory

The provided lab files are incomplete. Through the course of the lab, you will complete the MHS file, the linker script, and the software application.

Procedure

This lab comprises four primary sections: you will create the MicroBlaze system hardware platform, build the application software for the MicroBlaze system, download the bitstream to the FPGA, and finally debug the application software. Below each general instruction for a given procedure, you will find accompanying step-by-step directions and illustrated figures providing more detail for performing the general instruction. If you feel confident about a specific instruction, feel free to skip the step-by-step directions and move on to the next general instruction in the procedure.

I. Creating the MicroBlaze System Hardware Platform

Create an XPS Project — Step 1

The Xilinx Platform Studio allows you to control the hardware and software development. It also provides an editor and a project management interface to create and edit source code. The XPS offers software tool flow configuration options. XPS also creates a Project Navigator project allowing you control of the hardware implementation flow in a familiar environment.

XPS supports the creation of the MSS file (Microprocessor Software Specification), the MVS file (Microprocessor Verification Specification), and software tool flows associated with this software specification. It supports customization of software libraries, drivers, interrupt handlers and the compilation of the user program using the EDK default Linker Script or by providing a Custom one.

❶ Open XPS: **Start→Programs→Xilinx Embedded Development Kit→Xilinx Platform Studio**

❷ In XPS, select **File→New Project**

Create New Project dialog box opens as shown in figure 3-2.

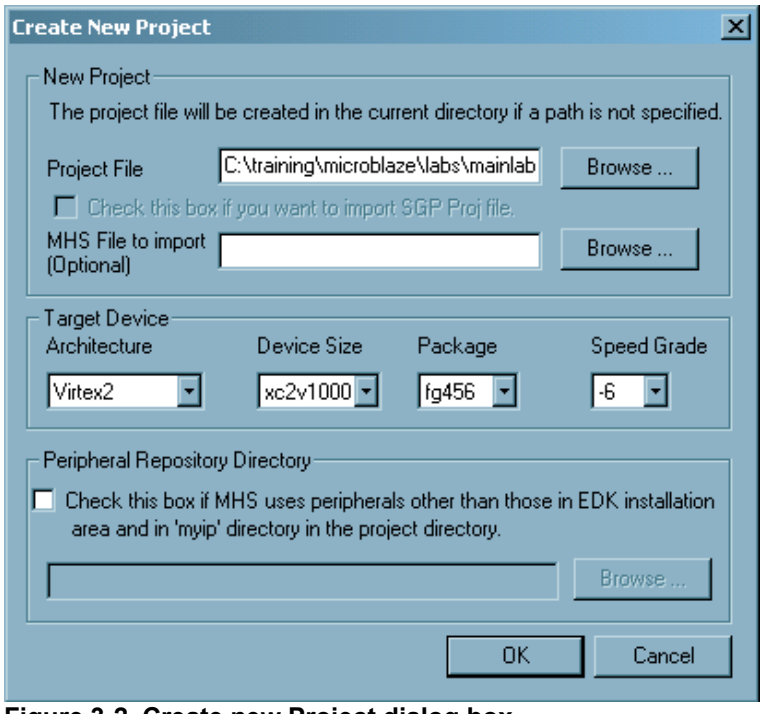

Figure 3-2. Create new Project dialog box.

❸ Use the Project File Browse button to browse to the C:\training\microblaze\labs\mainlab folder. Click **Open** to create the system.xmp file, as shown in figure 3-3

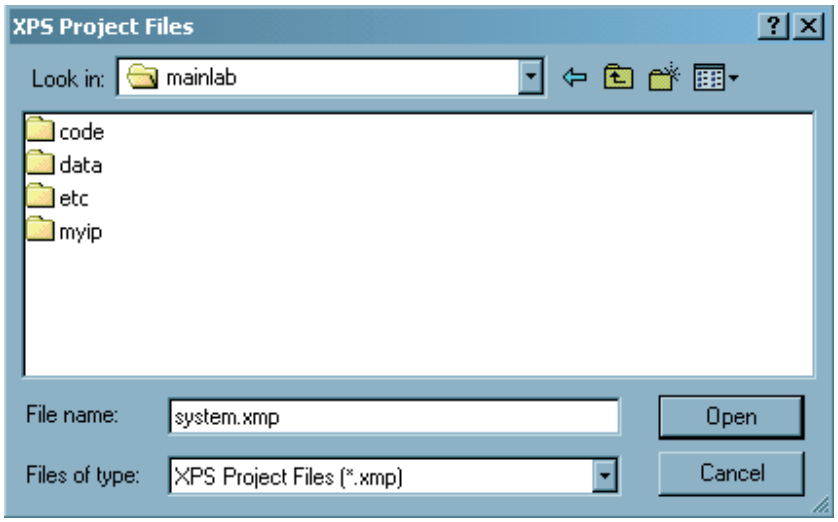

Figure 3-3. XPS Project Files Directory.

❹ Use the MHS File to import **Browse** button to select your system.mhs file

❺ Select the system.mhs file and click **Open** as shown in figure 3-4

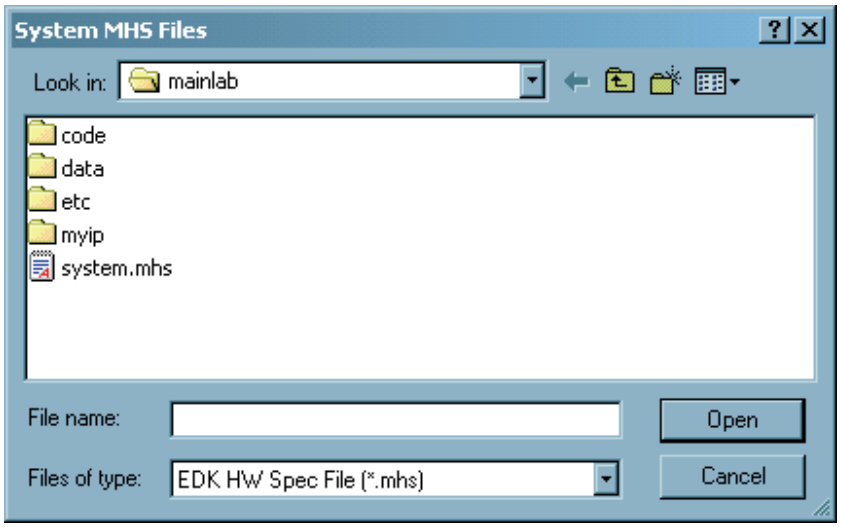

Figure 3-4. Adding the MHS file to the XPS Project.

❻ Set the Target Device to the following:

- Architecture: Virtex2
- Device Size: xc2v1000
- Package: fg456
- Speed Grade: -4

❼ Click **OK** to create the project

Hardware Entry: ADD IP Step 2

The Microprocessor Hardware Specification (MHS) file defines the hardware component of the design. An MHS file defines the configuration of the embedded processor system, and includes the following:

- Bus architecture
- Peripherals
- Processor
- Connectivity of the system
- Interrupt request priorities
- Address space

In this design, the lmb_lmb_bram_if_cntlr with the associated BRAM and the OPB UART have not been included in the MHS file. You will need to add this IP to finish the hardware design. XPS provides a utility to easily add the peripheral definition to a new or existing MHS file. The user is then required to specify the correct parameters and port connections.

❶ Double click on system.mhs in XPS to open it

❷ Examine the mhs file. Notice the section headers for the missing IP

❸ Close the system.mhs file

❹ Select **Project→Add Cores** to open the Cores List dialog. (see figure 3-5)

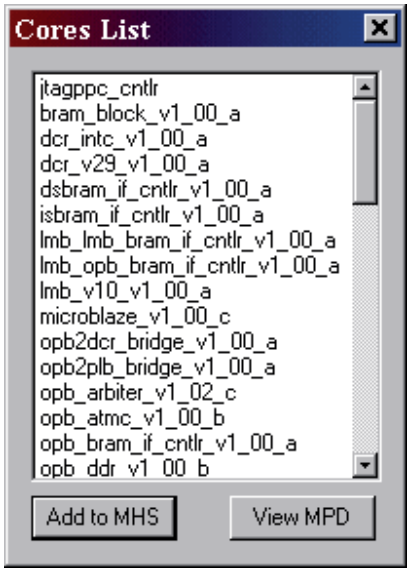

Figure 3-5.

❺ Select the opb_uartlite_v1_00_b and click **Add to MHS**

❻ Select the lmb_lmb_bram_if_cntlr_v1_00_a and click **Add to MHS**

❼ Select the bram_block_v1_00_a and click **Add to MHS**

❽ Click the "x" to close the **Cores List** dialog

Connect and configure IP.

❶ Double click on system.mhs in XPS to open it

❷ Move each of the peripherals to the appropriate place

❸ Properly configure the C_BASEADDR and C_HIGHADDR for the lmb_lmb_bram_if_cntlr and the opb_uartlite using the MEMORY MAP table

❹ Complete the lmb_lmb_bram_if_cntrl to match the following:

```
BEGIN lmb_lmb_bram_if_cntlr

 # Generics for vhdl or parameters for verilog
PARAMETER INSTANCE = inst_lmb_lmb_bram_if_cntlr
PARAMETER HW_VER = 1.00.a
PARAMETER C_MASK = 0x00800000
PARAMETER C_BASEADDR = 0x00000000
PARAMETER C_HIGHADDR = 0x00001fff

# Global ports
PORT LMB_Clk = sys_clk

# Bus Interfaces
BUS_INTERFACE ILMB = i_lmb
BUS_INTERFACE DLMB = d_lmb
BUS_INTERFACE PORTA = lmb_porta
BUS_INTERFACE PORTB = lmb_portb
END
```

The BUS_INTERFACE ILMB = i_lmb connects the slave i_lmb bus to the MB peripheral. The BUS_INTERFACE PORTA = lmb_porta connects the BRAM to this controller.

❺ Complete the bram_clock for use by the lmb_lmb_bram_if_cntlr to match the following:

```
BEGIN bram_block
PARAMETER INSTANCE = bram1
PARAMETER HW_VER = 1.00.a
BUS_INTERFACE PORTA = lmb_porta
BUS_INTERFACE PORTB = lmb_portb
END
```

You can see that the lmb_porta is used as the bus interface between the bram_block and the lmb_lmb_bram_if_cntlr block.