MicroBlaze “Hello World!” and the Embedded Development Kit (EDK)

Learning Objectives

- To gain further experience with the Nexys development boards
- To become familiar with the Xilinx Platform Studio development environment
- To configure a functioning MicroBlaze soft-core processor with associated peripherals and synthesize a hardware implementation of the system
- To write firmware in a high-level language (“C”) to execute on the MicroBlaze
- To compile, link, and load the executable program and associated system onto an FPGA
- To learn typical start-up requirements and other characteristics for LCD displays
- To use General Purpose I/O to send ASCII characters to an LCD display

Introduction and Overview

Welcome to the wild world of Xilinx’s Embedded Development Kit (EDK). The world of EDK represents a higher-level of design abstraction than what you’ve previously experienced. Where as the requirements and specification steps are essentially the same as previous designs, the architecture, component, and integration steps are different. The blocks used in the architecture section of the design methodology are now constrained to being cores available in the EDK environment. In standard microprocessor-based digital design, you were forced to search for a microcontroller that contained the peripherals that you required. In the EDK design environment, you are now able to choose the exact peripherals you require for your system. These components are provided as intellectual property cores or “IP cores” and the Embedded Development Kit allows the designer to configure the system and to generate the appropriate hardware to be downloaded to an FPGA. In a similar manner, most of the system integration details, such as compiling linking and loading the firmware, are now implemented by the design tools.

Getting Started

With the completion of Lab 1, you should be comfortable with using the Xilinx ISE 9.1i and the Digilent Nexys board. Now you will use the Xilinx EDK 9.1i and its XPS GUI to implement embedded systems. The course website contains the “Embedded Development Environment (EDK) 9.1i – Nexys Board Tutorial”, and that tutorial will take you through the steps required to develop a working system using the EDK. The first step in starting this experiment is to run through and understand that tutorial.
“Hello World!” System Requirements

The system should print the message "Hello World!" on a LCD screen and use the minimum code space (number of words of BRAM) and the minimum number of FPGA programmable logic resources (slices).

“Hello World!” System Specification

Implement an embedded computer system on the Nexys board that can print the message “Hello World!” on the board’s LCD screen. The computer system should be centered about a MicroBlaze soft-core processor. Configure the MicroBlaze processor, write the firmware, and download the hardware and firmware into the Spartan-3 FPGA on the Nexys development board. The computer system will send the “Hello World!” message to the LCD on the board. You will need to review the DIO5 Reference Manual section on the LCD to understand the interface requirements for the LCD display (see the website material).

“Hello World” System Architecture

The minimum hardware configuration for the “Hello World!” project is comprised of a MicroBlaze soft-core processor, program memory, data memory, general purpose I/O peripherals, and the XNLCD module for the Nexys board. This hardware architecture is shown in Figure 1. The MicroBlaze Instruction Local Memory Bus (ILMB) and Data Local Memory Bus (DLMB) should be connected to dual port Block RAM memory (BRAM) as shown in Figure 1 using interface memory controller IP cores.

NOTE: The MicroBlaze processor has a Harvard architecture and therefore uses a different address space and memory bus for instruction and data memory. The GNU compiler used in the EDK produces code for Von Neumann architecture machines and therefore uses one address space for both the instruction and data memory. Since the BRAM is dual ported you can connect both the ILMB interface memory controller and DLMB interface memory controller busses directly to the dual ported BRAM. If the base address for the instruction and data memory is set to the same value then both address spaces will be mapped to the same physical memory in the BRAM and the machine code will execute properly.

Separate General Purpose I/O (GPIO) peripheral devices are used to connect MicroBlaze to the LCD data and control busses. These GPIO modules are used as parallel ports that can be configured as either input or output. The GPIO modules are all connected to MicroBlaze via the On-chip Peripheral Bus (OPB). These GPIO modules are memory mapped so I/O operations are performed by reading or writing to the appropriate memory locations. One GPIO module should be used for each of the data and control busses as shown in Figure 1.

The main firmware task of your system is to communicate with the LCD on the Nexys board. The two aspects of this communication are listed below. The specifics regarding both of these aspects are found in the “DIO5 Reference Manual” located on the CPE 329 website.

1. **Understanding LCD interface:** The bus timing diagrams for both the read and write bus cycles are detailed in the DIO5 reference manual.

2. **Understanding the operation of the LCD:** This aspect can be further subdivided into the following two areas:

   i. **Initialization of the LCD:** Devices such as LCDs generally require a sequence of initialization commands to be sent to their internal registers before the devices can be used. The DIO5 Reference Manual describes the command sequence required to initialize the LCD display.
ii. Writing the “Hello World!” message to the LCD: Once the LCD is properly configured, it will display ASCII characters written to the LCD data register. Other LCD commands may be used to clear and position the cursor at a specific location on the LCD display. Timing information and the LCD command formats can be found in the DIO5 Reference Manual.

![Lab 2 hardware architecture diagram]

**Figure 1: Lab 2 hardware architecture.**

"Hello World" System Components

The basic component of the “Hello World!” system is the FPGA design implemented on the Nexys board. The subcomponents of the FPGA design include the embedded computer system on the Nexys board, and its associated firmware. The hardware components are the Nexys board and the LCD peripheral device.

"Hello World" System Integration

You must make the proper pin assignments for the GPIO modules. The bit ordering in the EDK version 9.1i is different than that expected by the LCD interface. The least significant bit position for each GPIO port is swapped with the most significant bit position of the LCD signal lines. You should therefore be aware that the constraint file is such that the GPIO_data(0) connects to the LCD_data(7), the GPIO_data(1) connects to the LCD_data(6), … and the GPIO_data(7) connects to the LCD_data(0). The same reverse ordering is needed for the control busses. The system.ucf file is presented in Table 1.

Also note that the delays for the LCD interface protocol can be implemented as presented in the lecture material with a spin instruction: for ( i = 0; i < count_value; i++ ). The count_value was initially determined experimentally Carter Deleo, CPE 329 Lab Assistant in Spring 2007, and has been reexamined with the EDK 9.1i to by given be the following formula:

\[
\text{count value} = \left( \frac{\text{time}_{delay}}{f_{clk}} \right) / (0.037). 
\]

For example, if the \( f_{clk} = 50\text{MHz} \) and the delay required is \( \text{time}_{delay} = 20 \text{ ms} \), then the count value = 27M. Note that the compiler is assumed to use the default settings for this calculation, i.e., just follow the same procedure used in the tutorial to compile the C software. There are issues with ISE and EDK for version 9.1i that "optimize" the code, and therefore the timing equations presented in the lecture notes will not apply.

#system.ucf file for cpe 329 lab 2 (note that # identifies a comment)

#lcd control signal I/O pin assignments
#pin<2> = RS signal, pin<1> = R/W signal, and pin<0> = E (or R) signal
NET "<signal name>_GPIO_IO_pin<2>" LOC = "P15" ;
NET "<signal name>_GPIO_IO_pin<1>" LOC = "T7" ;
NET "<signal name>_GPIO_IO_pin<0>" LOC = "R5" ;

#lcd data signal I/O pin assignments
NET "<signal name>_GPIO_IO_pin<7>" LOC = "N15" ;
NET "<signal name>_GPIO_IO_pin<6>" LOC = "J16" ;
CPE 329 Lab 2

#system clock source from nexys board
NET "sys_clk_pin" LOC = "A8" ;

#reset signal from btn0 on nexys board
NET "sys_rst_pin" LOC = "J13" ;

Table 1: system.ucf constraints file.

Project Demonstration and Lab Report Submission

When your project is working properly and your lab report is ready to submit, zip up the project into a .zip file, and with your lab report (a word .doc file), submit it electronically to the Blackboard Assignment folder as you did in Lab 1. Your lab report should also include the following design sections that describe the stages of the standard design methodology as they relate to this experiment:

1. System Requirements
2. Specification
3. Architecture
4. Components
5. System Integration

Your report should be clear and concise in its description of the “Hello World!” design. Remember, diagrams are a viable method to conveying important design information; consider using them in conjunction with written descriptions of your design.

Your lab report should also include a Conclusions section. This section should not be a summary of the procedures or a description of what was done in the experiment. This section should include an analysis of the final system and may include topics such as: were the system requirements met? Is there any error in the final system and if so estimate it? Could you have met the system requirements with a better design or an alternate approach? If so how? What tradeoffs were made in the areas of cost, performance, accuracy, power consumption or reliability (if any of these are appropriate)?

Grading

Successful completion of the system requirements and a well written lab report will earn a maximum grade of 9 out of 10 points. In order to achieve a grade above 9 on this lab, you need to demonstrate a complete understanding of the lab by adding additional features/functionality to it. Some suggested lab enhancement ideas are listed below. Higher complexity will earn a higher grade. Adding other features/functionality is permitted as long as they are approved by an instructor and documented in an “Added Features” section following the conclusion of your report. Any changes made to the original specification due to added features must be listed along with an explanation of why the enhancement was made.

Suggested added features:
- Write text on the second line of the LCD
- Scroll text on the first and/or second lines of the LCD using LCD control functions
- Scroll text on the first and/or second lines of the LCD using a software algorithm
- Continuously change a character without clearing the screen