Lab 1A Handout

Starter Lab

Due: October 7, 2016

Lab Overview

Lab 1A is the starter lab, designed to introduce the development tools used for creating hardware/software interfaces. After completing Lab 1A, you will have created an RTL project using the **Vivado** design suite from Xilinx. Your basic RTL project will consist of the soft-core **Microblaze**[1] processor running on the **Nexys4 DDR development board**[2] purchased from Digilent. The Nexys4 DDR board utilizes a Xilinx FPGA chip, the Artix 7.

The Microblaze IP core is configurable, and permits creation of a system with different memory sizes, clock speed and # of pipeline stages. We use the Vivado design Suite to create hardware designs and to export the hardware to the Xilinx SDK, where we can then develop **C** programs to run on our custom designed processor. We set up small programs to gather timing information about the embedded system to see how many clock cycles different operations take. Finally, plots of gathered data are created for reporting on timing. Lab 1A explores tools used in an embedded system design flow.

1 Introduction

In the development of your embedded system it is important to monitor physical constraints, such as time, cost and power. The Vivado Design Suite provides tools to help a system designer understand the resources used in their design. The operation of an embedded system is time constrained. This means we need to understand two concepts: **Throughput** and **Latency**, and how they can influence the timing properties of an operation.

The latency of an operation is the amount of time it takes to perform that operation. The throughput of an operation is the amount of data that can be processed by the particular operation in a given time frame. It is important to understand that while it may seem so at first, a high latency does not necessarily guarantee a low throughput. For example, pipelining can be used to improve the throughput in a system with high latency. The high latency will still affect how long it takes to process the first operation, but each successive operation will take significantly less cumulative time to be processed. When this is done, the impact of the high latency is minimized. In this sort of implementation, performing a single operation with a burst of data will be much more time efficient than performing multiple smaller operations with smaller data chunks.

2 Equipment

Throughout the course we will be creating Vivado RT (Register Transfer) projects, which are based around the block level diagram. An RT project refers to the level of abstraction selected for describing the design.

Hardware Nexys4 DDR Digilent Development board with Artix-7 FPGA

Software Vivado 2016.2

Software Xilinx SDK (with Vivado Integration)

The **Lab Hardware Handout** provides information about setting up the Nexys4 DDR Development board with Vivado 2016.2 and the Xilinx SDK. It also contains general information which is useful for all of the Labs in ECE 153A. The **Interfacing with the DDR** Handout provides additional information on interfacing the DDR2 memory with Microblaze. For this lab, the DDR2 memory will need to be a part of your design. Read the **Lab Hardware Handout** and **Interfacing with DDR** Handout before starting the exercises.

3 Exercises

3.1 Microblaze

The configuration of the Microblaze processor provides various options, such as the size of the BRAM memory used by Microblaze, adding data and instruction caches and enabling debugging tools. Follow instructions in the **Interfacing with DDR** Handout to start Vivado and create a project containing a Microblaze processor along with DDR2 Memory. Explore the different configurations of the Microblaze processor by selecting a configuration for your processor from the list below.

1. Microblaze “Typical” Predefined Configuration, deselect “Use Instruction and Data Caches”
2. Microblaze “Max Performance” Predefined Configuration

Notice how the selection of different configurations changes the utilization report of the FPGA. (See *Lab Hardware Handout - Understanding the size of your design* to find the utilization report)

3.2 Adding Peripherals

The Microblaze processor provides a central processing unit for our embedded system to interact with the physical world. The next step in the design of our system is to begin adding I/O peripherals. We will also add the MIG (Memory Interface Generator) IP to interface Microblaze with the DDR2 memory. The Block RAM available on the FPGA is limited to 4680 Kbits and for applications that require a larger memory space, the DDR2 memory on the board should be used. Follow instructions in the **Lab Hardware Handout** to add the IP for 3 peripherals: (1) AXI Timer, (2) Seven Segment display and (3) LEDs to your design. Peripherals communicate with the processor using buses. In our design we are using the AXI bus. Buses are one of the design elements which face timing issues. As we complete the timing analysis in section 3.3, we begin to see when peripherals are competing for bus access.

3.3 Timing Analysis Guidelines

The code for this project is enclosed in timing.c. This code provides you with the basic structure for exercising different peripherals and for measuring the number of clock cycles.

1. Measure each operation repeatedly (1500 samples is the default) and note the results of the timing value and the number of clock cycles it required. A histogram function that counts the number of samples in uniformly spaced, equally sized

bins is included in the code. You may choose an appropriate value for total number of bins and use this function for statistically analyzing your data. Remember the FPGA is using a 100MHz clock. For the longer string operations, you need not test to 1500 samples, but do enough testing to get reasonable statistics.

2. In order to properly characterize the timing of length-dependent operations, you will need to apply linear regression to the data collected. This will allow you to describe the timing properties of the peripheral as a linear function $t = \alpha + \beta x$ where α is a constant due to the latency of establishing communication with the peripheral, β is a scaling factor that corresponds to throughput, and x is the size of the input data. You should use a linear regression approach that minimizes the sum of least squares of error.

   Be careful about the number of variables you create in the data or stack segments – if the code suddenly stops running or halts, it is likely that you exceeded the available stack size, exceeding the data segment will cause loader or compiler errors.

4 List of Operations to be Timed

1. Timing of writing strings to the USB Port: (a) Write 2 strings of same length but different characters to check data dependence (b) Write 5 strings of different lengths to check length dependence. Do these measurements for the following functions:
   (a) print()
   (b) printf()
   (c) xil_printf()
2. Timing of writing **longer** strings to the USB Port using **print()**
3. Timing of 7 Segment Display to display all zeros
4. Timing of Addition and Multiplication using integers
5. Timing of Addition and Multiplication using floating point
6. Timing of LEDs
7. Timing of writing one byte to the DDR2 memory
8. Timing of reading one byte from the DDR2 memory

Software Timing: Modules take some number of processor cycles, often input data or size dependent.
*Typical model: T(total) = Throughput * Size + Setup*
Timing is subject to hardware overhead and conflicts.
While you are gathering data another method in main, extra_method() is also running. This extra_method() adds a source of resource contention, which will change the timing properties of communicating with these peripherals.

5 Reporting Results and Observations

It should be taken into consideration that embedded systems are targeted at tightly constrained environments, with multi-kHz sampling rates, multiple interrupt sources, and relatively small memories with realistic bus environments. Write a two-page report, covering critical aspects of embedded software design and answer the following questions:

1. Report on which Microblaze configuration was used for your timing measurements. Explain how selecting different Microblaze configurations affects the utilization of the FPGA resources.

2. For each type of print operation:

   (a) Make histograms of the measurement data. Include confidence figures - mean, standard deviation, and confidence estimate for the mean. Explain how the confidence figures were obtained.

   (b) Plot your data to illustrate the change in time with respect to the size of the input data. Perform a linear regression on this data in order to characterize the data dependency in the form of $t = \alpha + \beta x$. You will again need to include confidence figures for α and β to estimate the confidence in these linear models. What does this say about the operation's latency and throughput? Would it be more efficient to perform this operation once with one large burst of data or by performing this operation multiple times with smaller amounts of data? Why?

   (c) Was there any data-dependent variance in the timing? If yes, why do you think this particular operation showed time variance with different sets of data?

   (d) Was there any time variance not due to length dependence or data dependence? If yes, why do you think there was variance in the timing properties of this particular operation even though the same set of data was being used each time?

3. Make histograms for your timing measurements for operations 3-8 of Section 4. Include confidence figures for the measured data and explain why certain operations show lesser variation as compared to others.

References

[1] *ECE 153A - Course Microblaze Notes*. URL: `http://bears.ece.ucsb.edu/class/ece253/MicroBlaze_Overview.pdf`.

[2] *Nexys4 DDR Artix-7 FPGA Board*. URL: `http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,1338&Prod=NEXYS4DDR`.