Lab 1A Handout

Starter Lab

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**[2] processor running on the **Artix A7-100T development board**[1] from Digilent. The Artix A7 board utilizes a Xilinx FPGA chip, the Artix 7. The Artix A7 board was previously named Nexys 4 DDR, and is identical to the Nexys 4 DDR in terms of functionality.
The Microblaze IP core is configurable, and permits creation of a system with different memory sizes, clock speed and # of pipeline stages. The goal of this lab is to use the Vivado design Suite to create a custom hardware design for a processor and its peripherals, export the hardware to the Xilinx SDK, and then develop **C** programs to run on the custom processor. The **C** programs will gather timing information about the embedded system in terms of the number of clock cycles to execute different operations. Finally, timing uncertainties in the gathered data will be statistically analyzed.

1 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 Nexys A7-100T Digilent Development board with Artix-7 FPGA

Software Vivado 2018.3

Software Xilinx SDK (with Vivado Integration)

The **Lab Hardware Handout** provides information about setting up the Artix A7 Development board with Vivado 2018.3 and the Xilinx SDK. It also contains information on interfacing the DDR2 memory with Microblaze, and general information which is useful for all of the labs. For this lab, the DDR2 memory will need to be a part of your design. Read the **Lab Hardware Handout** before starting the exercises.

2 Microblaze Configuration to be Explored

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. For this lab, build Micoblaze using the configuration instructions in the Lab hardware handout. Follow instructions in the **Lab Hardware Handout** to start Vivado and create a project containing a Microblaze processor along with DDR2 Memory.

3 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. Follow instructions in the **Lab Hardware Handout** to add IP for: (1) Two AXI Timers and (2) LEDs. You should add the other peripherals in the hardware handout, but they are not used in this lab. 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 of operations listed in Section 5, we begin to see when peripherals are competing for bus access.

4 Timing Analysis Guidelines

After building the hardware and generating bitstream in Vivado, export your hardware and launch SDK. Create a new "blank" Application Project in SDK. We have provided default source code files that you will need for this lab. Copy these files to the "src" directory of your Application Project. The main code for this project that will be edited by you 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.

Measure each operation repeatedly (1500 samples is the default) and note the value of 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.

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.

5 List of Operations to be Timed

1. Timing of writing to the USB Port:
   (a) Write a floating point number using ***printf()*** (Always use printf() with a "\n", eg.: printf("A\n") to avoid BRAM overflow)
   (b) Write a string of 10 characters using ***xil_printf()***
2. Timing of addition and multiplication using integers
3. Timing of addition and multiplication using floating point
4. Timing of turning on and turning off LEDs
5. Timing of reading a byte from the DDR2 memory
6. Timing of reading 8 bytes from the DDR2 memory

For tasks 5 and 6, generate a large buffer in the DDR2 memory, something like `char buffer[1000000];`, and use the rand() function and modulus to make random index, and read either 1 byte or 8 consecutive bytes from that address.

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 time for communicating with these peripherals.

6 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, including the following results in your report:

1. Plot histograms and CDFs of the measurement data for the operations listed in Section 5. Include expected values for your measurements.
2. What do you observe about the magnitude of measured values and the timing uncertainties associated with different operations?

References

[1] *Artix A7-100T FPGA Board.* URL: `https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/`.

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