Fundamentals of Reconfigurable Computing ECGR6090/8090

INTRODUCTION TO FPGAS

1 Objective

The goal of this lab is to introduce the process of creating bitstreams and applications for FPGAs used in this class. More details will be unveiled in future labs. At this time, the goal is to come away with:

- the understand of how to set up your environment to access the tools
- starting and using Xilinx Platform Studio (XPS)
- the basic steps to synthesize a hardware design
- the basic steps to add, compile, and run a C application on an FPGA

2 Lab Setup

This lab is divided into four sections; each of which has an accompanying tutorial. The tutorials can be found on the Class Webpage. When beginning a section, first read the appropriate tutorial to gain familiarity with the tool flow or concepts and then complete the setion. You will begin by building a base system using the Base System Builder. This will act as the foundation upon which the remainder of the lab will be built.

1. Getting Started

From any X-Windows machine on campus (i.e., the Mosaic Solaris labs), open a Terminal window and log in to homer. Read the first step in the **Base System Builder Tutorial** to login and add the appropriate environment variables.

- Create a directory build directory:

```
bash-2.05$ mkdir lab1
bash-2.05$ mkdir lab1/build
bash-2.05$ cd lab1/build
```

2. Building A Base System

Next, we are going to open Xilinx Platform Studio (XPS) and use the Base System Builder (BSB) wizard to create our Base System. BSB allows us to specify which IP cores we want to include in our system.

Finish reading the **Base System Builder Tutorial** which explains the how a base system is made. After reading the tutorial start XPS and create a new base system with the following components.

- Open Xilinx Platform Studio

  bash-2.05$ **xps**

- Create a new project using the **Base System Builder wizard**

  Click **Browse** to locate the lab1/build directory

  Name the file *lab01.xmp* and click **Save**

  Click OK start using the BSB wizard

Now we will setup the system by using the BSB:

- We will be using Xilinx's Virtex-II Pro ML310 Revision D
- Set the Reference Clock to 100 MHz, Processor's Clock Frequency to 300 MHz and the Bus Frequency to 100 MHz
- Turn off JTAG Debugging
- Include an RS232 Uart with a baudrate of 9600, 8 Data bits and no parity bits
- Add a PLB BRAM Interface Controller with a memory size of 64 KB
- Remove any other unnecessary cores not mentioned in this list
- Uncheck both the Memory and Peripheral selftests since we will be writing our own applications

At this point, the main XPS window should appear and look something like the window in Figure 1.

We will use this base system for the remainder of this lab. Take a few minutes to familiarize yourself with XPS. The component listing and their connections in the system view area (on the right side). On the bottom is a console window (which will log the output from various commands that XPS invokes). On the left side is the project information area. Of course, across the top are menus and a row of buttons that provide a short cut to specific menu items.

Also take some time to look at the various menu options. Pay close attention to the selections under "Hardware," "Software," and "Device Configuration." Explore the system information window (the tree of components is compressed by default, but you can expand some items).

3. Building an Application

First read the **Building an Application Tutorial**. We will start by creating a simple "Hello World" program. Create an application called "HelloWorld" following the directions in the tutorial. Compile the program to generate the executable.elf. The program should use xil_printf function to print out:

```
Hello World!
My Name is (your first and last name)
My ID is (your student ID number)
```

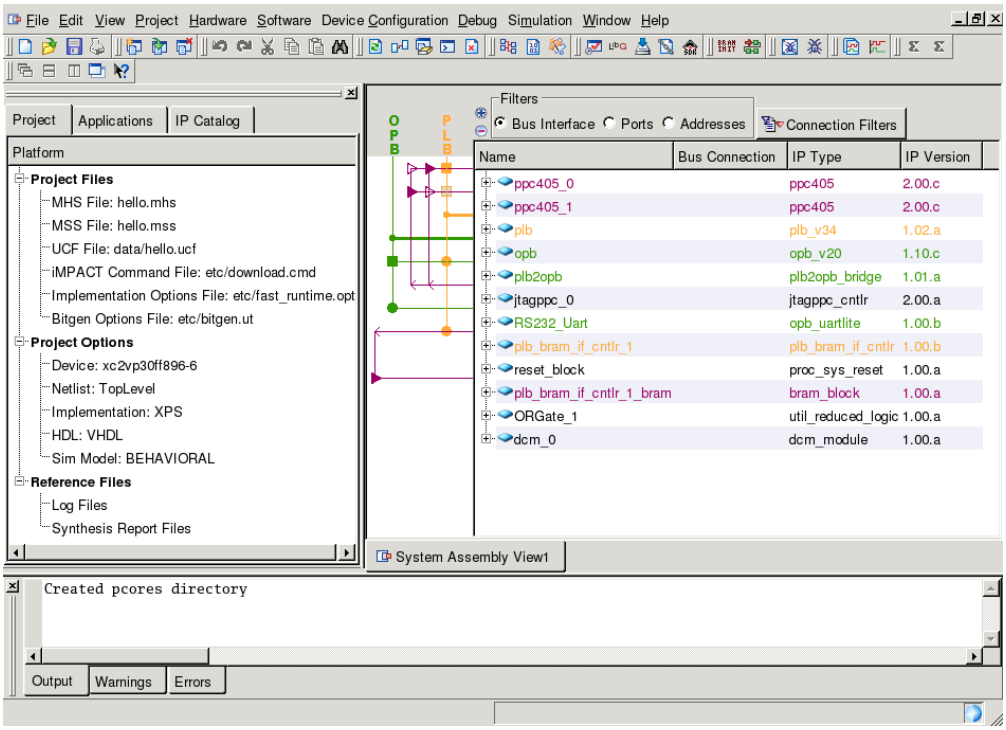

Figure 1: The Main XPS Window

4. Generating the Bitstream

Now we will use the tools to generate the hardware and software parts of our system. Starting with hardware generate the netlist followed by the bitstream. Using the software executable generated in the previous section initialize the BRAMs so the program will be loaded into memory when the bitstream is downloaded to the board. Follow the **Synthesizing Tutorial** which explains the steps in more detail.

5. Running the Application on the FPGA

Follow the **Synthesis Tutorial** to use XMD to create an ACE file and then the **FPGA-Session Tutorial** to use fpga-seseion to download the ACE file to an ML310 and run your program.

3 Assignment

Now that you are familiar with the tools to generate hardware and software we are going to create a new Application to read and write to memory. This requires some familiarity with pointers and understanding addresses. (You will still use the base system you have previously created; however, you will *add* an additional application and IP cores)

1. Add a second BRAM to your current design (requires 2 IP Cores from the **IP Catalog**):

   A PLB BRAM Controller (rename it "plb_bram_lab1_block_if_cntlr")

   A Block RAM (BRAM) Block (rename it "plb_bram_lab1_block")

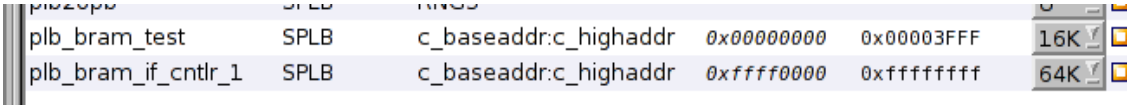

Figure 2: Correct BRAM memory addresses

2. Connect the SPLB port for the controller to the PLB (bus).
3. Connect Port A of the controller to Port A of the bram_block_test
4. Assign the new bram test block to be **16KB** in size. Use the automatic "Generate addresses" feature of XPS for the plb_bram_test controller
5. Build a bitstream with this design

Begin a new sofware application called "MyMemTest". In this application, you will write data to the newly added BRAM, then read the data back to verify its correctness. You can see from Figure 2 that the new block of memory resides from addresses 0x00000000 through 0x00003fff.

- Create a program with the following functionality (*Hint: Use pointers*)
    1. Write the value 0xCAFEBABE to the address 0x00003000
    2. Use the xil_printf function to print the value at address 0x00003000
    3. Use the **sleep(int seconds);** function to sleep for 2 seconds
       : sleep(2);
    4. Store your student ID number at the address 0x00001000 and then print the value at address 0x00001000
    5. Use the **sleep(int seconds);** function to sleep for 2 seconds
       : sleep(2);
    6. Declare a pointer of type Xuint32 (32 bit unsigned integers) to the base address of the test BRAM
    7. Treating this pointer as an array, initalize the BRAM with increasing integers, starting from 0, increasing by 3 for each value, untill the entire test BRAM is filled
    8. Print all of the values from the array using the xil_printf to verify it was correctly initalized
- Now finish building an ACE file and test it on the ML-310 board.

4 Grading

This lab assignment is due **Wednesday 9/26 by 5pm (EST)**. To receive credit, you must meet with either the T.A. or the instructor and demonstrate that you've completed the lab. By default, this can be done in Woodward Hall room 237 (the Unix lab). Alternative times are possible but need to be arranged in advance. Be prepared to answer in person any questions in the lab or recompile your program as part of demonstrating that you completed the lab. Do not wait until the last minute to begin the project; extensions will not be granted.