Hardware/Software Codesign for Wireless Systems (E168b)

Harris

Lab 0

An Introduction to the Xilinx Virtex II Pro Development Board

Introduction

As the first lab for E168B, this assignment will introduce you to the Xilinx Virtex II Pro Development Board and its capabilities. We will start with a self-test of the board to ensure its proper operation, followed by a basic hardware programming on the board using Verilog and the Xilinx Project Navigator. Then we will achieve the same results in C on the embedded PowerPC microprocessor using Xilinx's Embedded Development Kit (EDK) with Platform Studio and the Base System Builder. You will learn more about these as you go along. Important manuals and other documents are referenced at the end of this lab to help you through the understanding of this lab and other labs to follow. This handout will walk you step-by-step throughout this first lab, so that you can get acquainted with the functionality of the board and EDK.

0. **Board Documentation:** Skim through the XUPV2P user guide (http://www.xilinx.com/univ/XUPV2P/Documentation/XUPV2P_User_Guide.pdf). A hard copy of this guide, along with the FPGA and EDK manuals, is in a binder in the lab. Familiarize yourself with the capabilities of the board.

1. **Self-Test:** When the board is first turned on, a number of LEDs will light up, red, yellow and green in color. LEDs D17, D18 and D5 should be green in color and are indications for whether the 2.5V, 3.3V and 1.5V power supplies are working properly. The jumpers JP2, JP4 and JP6 can be used to turn of any of these power supplies. Doing so will cause the 'Reload PS Error' LED (LED D6 – next to the PS/2 port) to turn red. Remove the jumper(s), and press the reset/reload button SW1 for a few seconds, to restart the board. LED D12, next to the compact flash card slot, will flash red if no CF card is in the slot. If there is a CF card in the slot, a solid red light on LED D12 denotes an error. You can ignore this error for now. The four LEDs next to the PS/2 port indicate what program is running. A yellow light on LED D14 indicates that the 'Golden Configuration', i.e. the self-test is running. LED D19 turns green when the PROM configuration is loaded, and LED D20 turns green when the JTAG configuration is in use. These configurations are controlled via switches SW8 and SW9. When LED D4 turns red, it implies that the selected configuration has been successfully loaded. LEDs D7 through D10 are user LEDs and can be configured by the programmer. Similarly, switches SW7 are for user input.

   The Virtex II Pro Board has a 'Golden Configuration' saved on the Platform Flash present on the board. This is the self-test program for the board. To ensure that the board runs properly

and to get an idea of its capabilities, we will first run the self-test program. To do so, follow the steps below:

- Open a HyperTerminal Window – this can be done by going into the Start Menu -> Programs -> Accessories -> Communications -> HyperTerminal. Create a new connection, 'virtex2p', over the serial port, i.e. connect using 'COM1'. Set the Baud rate to be 9600, with 8 data bits. Set the parity as 'none' and stop bits = 1 with no flow control.
- On the board, make sure the 'Config Source' (SW9) is set to Prom configuration and the Golden setting. This means that the self-tests downloaded on the board are running. (Both switches on the Config Source, SW9, should be *closed* or *up*, LED D14 should be yellow, LED D19 should be green, and LED D4 should be red, once the board is switched on)
- Connect the Video port J13 to a VGA monitor, the serial port to a computer, the keyboard into one of the PS/2 ports, and the Ethernet interface to a LAN. Connect a mic into the 'line in' or 'mic in' and a pair of headphones/ speakers into the 'line out' or 'amp out', at ports J14 and J15.
- Open the hyper terminal connection window, 'virtex2p' you saved earlier and switch on the board (using SW11).
- Follow instructions on the screen to run the various self-tests to see the functionality of the board.
- You can ignore the first test, the SATA port test, because no hard disk is attached.
- During the Ethernet test, you will need an IP address. Remove the Ethernet cable from an adjacent computer in the lab and plug it into the Virtex board. When prompted for an IP address, enter the address of the computer from which the cable was taken (the address should be on a sticker on the computer). Note that this test works with Internet Explorer but not Firefox.
- The DDR SDRAM test will return 'failed' in certain cases; ignore that.
- Refer to board documentation (pages 97-114) to see exactly what your output should be.

2. **Xilinx Project Navigator: Control LEDs with DIP Switch:** Now that we are familiar with the board, let's write a piece of Verilog code to control the functionality of the board. The development board has a set of 4 DIP switches (SW 7) and 5 push buttons for user input. The outputs can be displayed using the 4 leds (LED0 through LED3). Create a new project called lab1_xx on your Charlie directory, where xx are your initials. Write a Verilog module that does the following –

   Led0 = SW0 & SW1
   Led1 = SW0 | SW1
   Led2 = SW2
   Led3 = ~SW3

   Remember that on the development board, for the user defined LEDs (D7-10), the pushbuttons (SW2-6) and switches (SW7) - 'When the FPGA drives a logic 0, the corresponding LED turns on. If the DIP switch is *up*, *closed*, or *on*, or the push button is pressed, a logic 0 is seen by the FPGA, otherwise a logic 1 is indicated.' (Pg 18, http://www.xilinx.com/univ/XUPV2P/Documentation/XUPV2P_User_Guide.pdf)

Refer to the XUPV2P manual to determine which package pins should be assigned for the inputs and outputs. Synthesize and implement your design. Generate the .bit file and then download your design on the Flash memory on the board using the procedure described below:

- After the '.bit' file has been generated, using 'Generate PROM, ACE or JTAG file', an iMPACT window will open. Since we are working with the on-board Programmable ROM, select 'Prepare a PROM file' on the window that shows up. Click 'Next'. Now select the options shown in Figure 1, since we are working with a 'Xilinx PROM', with the 'MCS' format. Enter a name for your prom file – lab1test_xx.

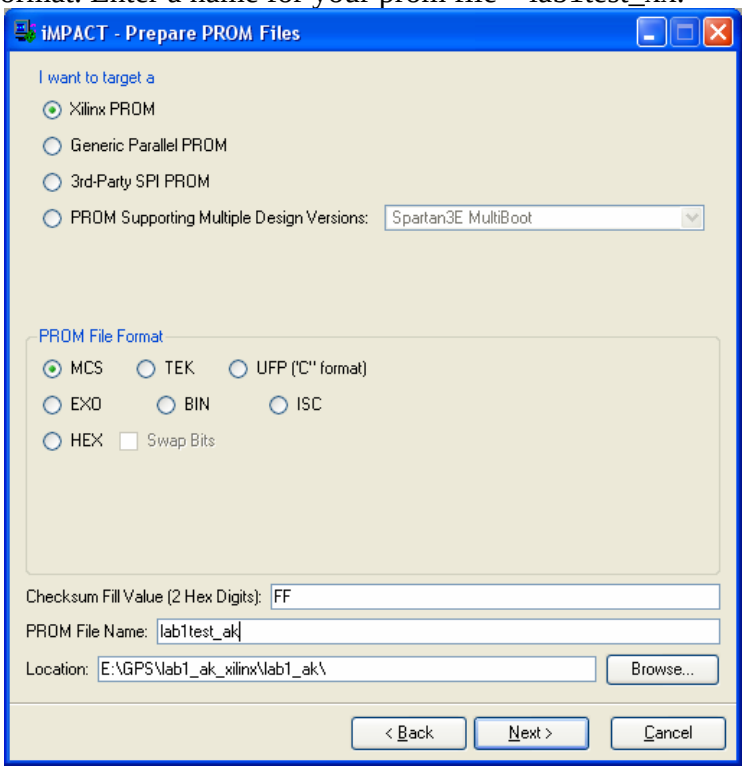

Figure 1: Prepare a PROM File

- On the next window – select Enable Revisioning and set the number of revisions to 2. This keeps the Golden Configuration in Revision 0 and lets you edit the Revision 1 to download you design, without erasing the Golden Configuration. The Prom we are using is the xcf32p. Select that and add it. DO NOT select 'Enable Compression'. This is shown in Figure 2. Click Finish.

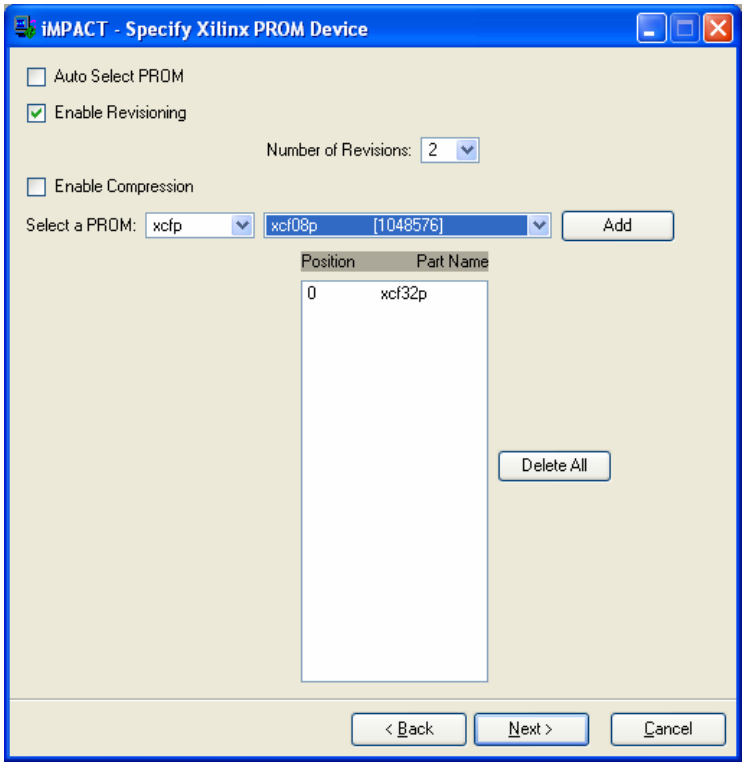

Figure 2: Xilinx PROM specifications

- After this, select the .bit file you just created and add the file to your 'revision0'. Select 'no' for adding another device to revision0.
- Add the same file to revision 1 when prompted to do so and select 'no' for adding another device to revision1.
- Generate files by selecting the option under 'Operations'. When prompted to compress the file, respond No, because the XUP Virtex-II Pro Development System hardware does not support this option.
- Now switch to 'Boundary Scan' mode by double-clicking on the side-tab
- Make sure the board is powered up and a USB cable connects the board to the Computer.
- In this mode, right click and select – 'Cable Setup'. In this case, we are downloading the design to the FPGA using a USB cable. Earlier, you have probably worked with a JTAG cable. It is necessary to detect and configure the baud rate of the usb port. Select the communication port as – 'Platform USB Cable' and for the speed, select 'Max Speed'. This aspect is more important when additional external boards are connected to the provided development board. It is to ensure that the JTAG clock frequency does not exceed the capability of additional devices.
- After it has detected the connection of the USB cable, right click on 'Initialize Chain'. This automatically detects the programmable devices present on the board - the Platform FLASH PROM (XCF32P), followed by the System ACE controller (XCCACE), and followed by the FPGA (XC2VP30).
- Now, right click on the first device – the XCF32P, and select 'Assign new configuration file'. Select this file to be the .mcs file you created earlier.
- Select 'BYPASS' for the other two devices.

- Right Click the XCF32P Prom and select ‘Program’. iMPACT brings up a window which allows you to decide which revisions are to be programmed. On the main page, in the ‘Program Properties’ Dialog, the ‘PROM Specific Properties’ are set to ‘Parallel Mode’. This is shown in Figure 3.

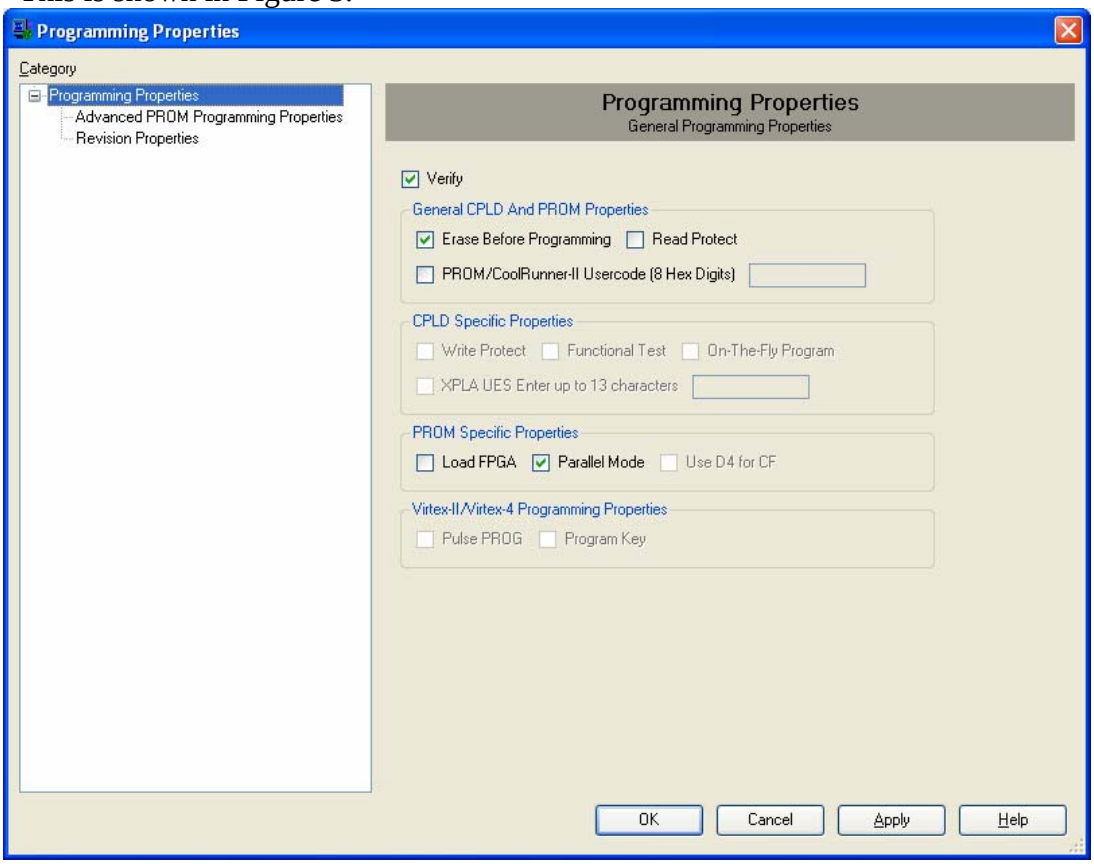

Figure 3: Programming Properties

- Now select ‘Advanced PROM Programming Properties’ from the left hand side tabs window. Ensure that the PROM is set to Slave Mode (clocked externally).
- Now click on ‘Revision Properties’ in the left tab. Here, uncheck Rev0 and select Rev1. In Rev1, check the Erase (ER) bit and make sure the write protect (WP) bit is not set. Then click ‘Ok’. This is shown in Figure 4 below.

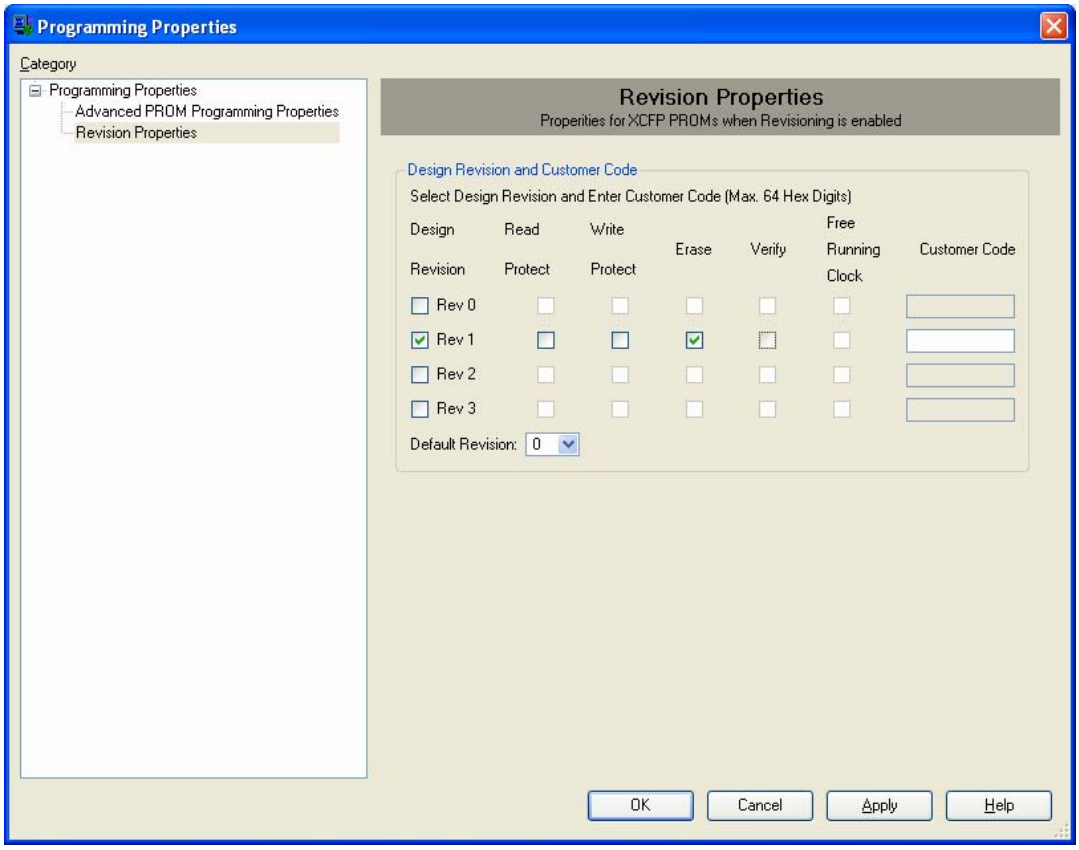

Figure 4: Revision Properties

- Your design is now being downloaded to the development board. This downloading takes a while, about 5-10 minutes. After the program has finished downloading, to load the new prom file, make sure the 'Config Source' is set to User PROM configuration mode (LEDs D19 and D4 should be on).
- Press the 'Reset/Reload' button to reload the FPGA with the current design.
- Verify your results by sending inputs via the Switches, SW7, and viewing the results on the LEDs D7-10.

This slow rate of downloading the design is because we are downloading to the Platform Flash PROM. If you don't need the design to remain on the board after reset, you can simply send it to the FPGA over the JTAG port. Follow the steps below to achieve a much faster configuration speed.

- Open iMPACT. Select 'Create a new project' and click 'ok'.
- Now select 'Configure devices using Boundary Scan (JTAG)'. This will detect the components on the system. These include the Platform Flash PROM, xcf32p, the System ACE device, xccace, and the Virtex II Pro FPGA, xc2vp30.
- Now assign new configuration files to each of these devices. For the PROM and SysAce devices, select 'Bypass', whereas for the FPGA, select the '.bit' file created by the Project Navigator.
- Click Ok
- If it gives a warning regarding the Startup Clock, ignore it and click Ok.

- Now right click the Virtex II Pro device and click on Program. Deselect ‘Verify’. And click ok.
- Again, check the implementation of your design to confirm results.

3. **Xilinx Platform Studio:** Xilinx’s Embedded Development Kit (EDK) and Platform Studio allow the user to create a hardware/software co-design project. In this lab, you will write a C program running on the embedded PowerPC microprocessor to read the LEDs and control the switches. In the next lab, you will create a project with a mixture of C software on the PowerPC and Verilog hardware on the FPGA. Follow the steps below to get started with EDK.

   - Open Xilinx Platform Studio. Create a new project using Base System Builder. In the ‘New Project’ window, for the Project File, browse to a new lab1_xx folder in your Charlie directory and save the file name as lab1_xx. (Refer to figure 5).
   - Select Use Repository paths – and browse to C:/xup_v2pro_dev_brd/lib and click ok.

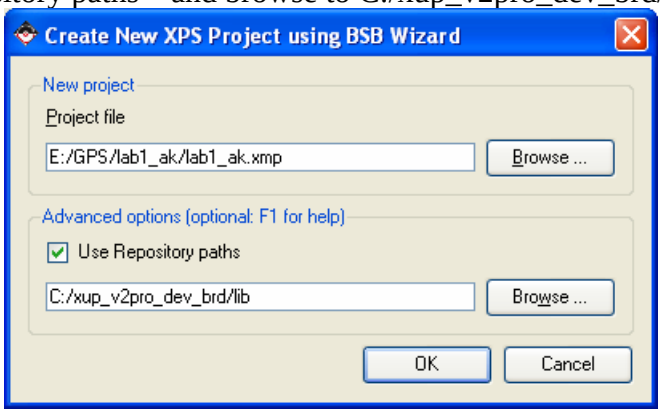

Figure 5: Create a New Project using Base System Builder

   - On the next window that shows up, select ‘I would like to create a new design’. Click Next.
   - Choose the Board Vendor as Xilinx, the Board name as ‘XUP Virtex-II Pro Development System’, and the Board Revision – C. If these options are unavailable, there is an error in your repository path and you will have to restart the process of creating the project.
   - On the next page, select Architecture as virtex2p, Device as xc2vp30, package as ff896, speed grade as -7, and the PowerPC as the processor in use. Click Next.
   - While configuring the PowerPC, change the Processor Clock Frequency to 300 MHz and On-chip Memory for Data and Instruction to 16 KB. Leave the other settings as default settings. (Figure 6). Click Next.

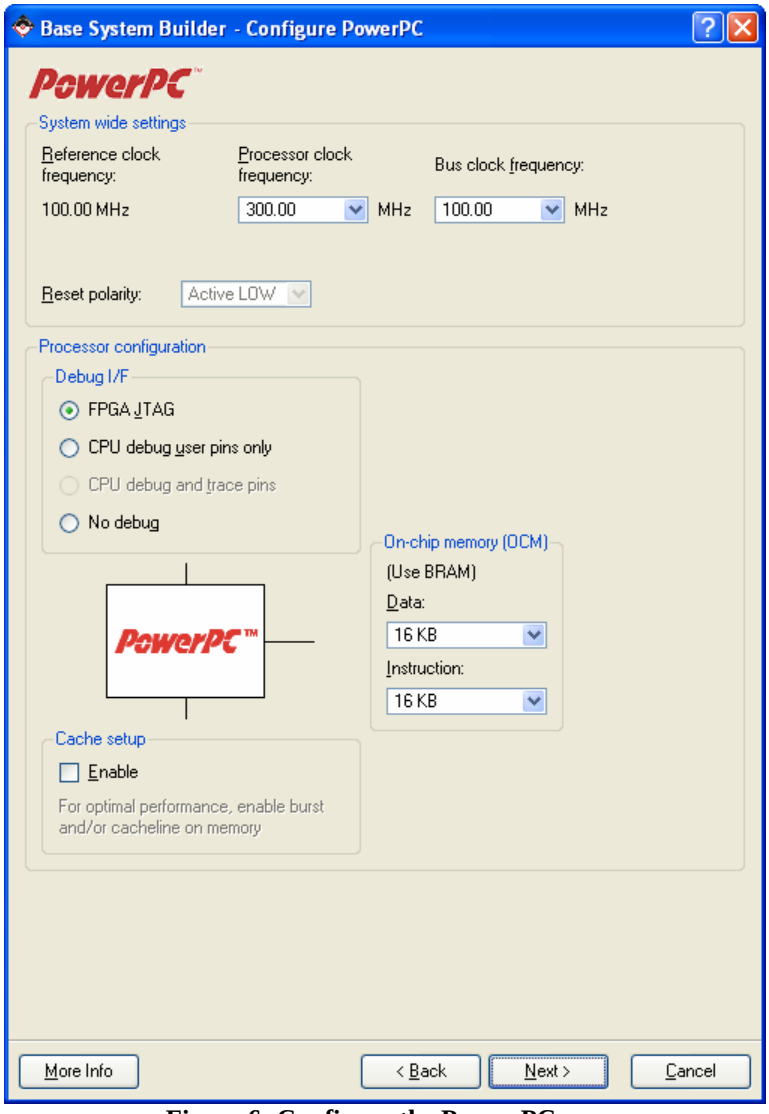

Figure 6: Configure the Power PC

- Next, BSB gives you the option of configuring IO interfaces. On the first page, select RS232. Set the baud rate to be '115200' (Figure 7). On the next page select the 4 bit LEDs, and 4 bit DIP switches. Unselect all other IO devices.

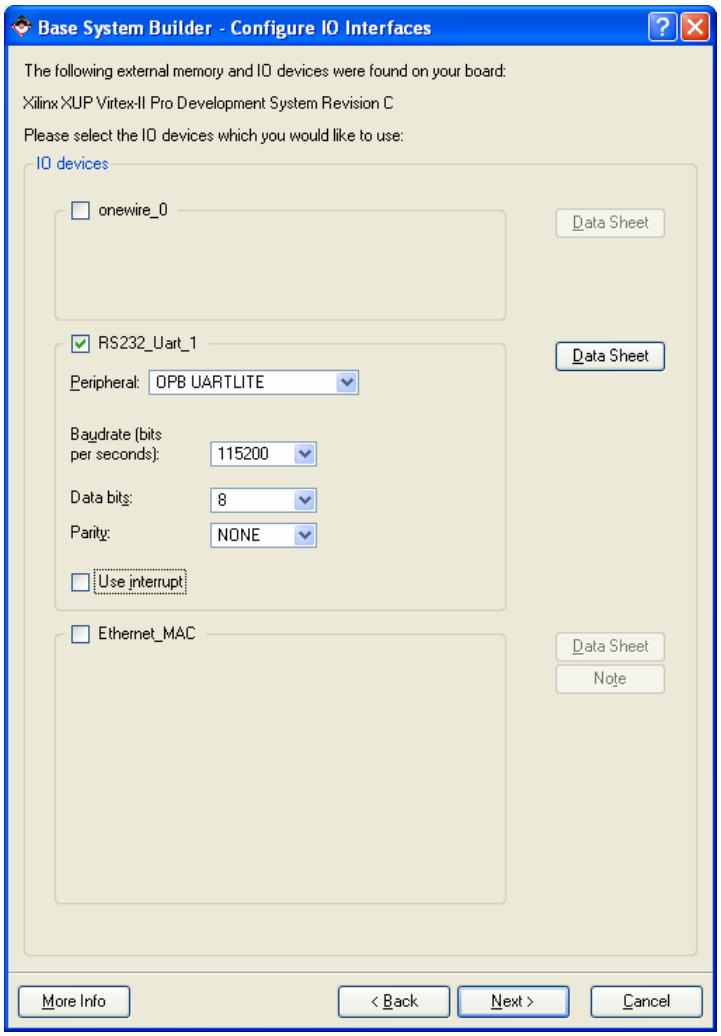

Figure 7: Configure IO Interfaces: RS232

- Leave the next screen, 'Add Internal Peripherals', at the default settings. Click next.
- On the next window, 'Software Setup', uncheck 'Memory test' and Click Next. This generates a sample peripheral test.
- Leave the next screen at the default settings and on the last page, hit 'Generate', and then 'Finish'.
- On the next screen, select download design. Make sure you have the previously saved HyperTerminal connection open and the board is powered up. In the HyperTerminal window, change the Baud rate to 115200 instead of the previously saved 9600. To do this, go to Call -> Disconnect. Then in File -> Properties, connect using COM1, click on Configure. Change the Bits per second to 115200. This is because we adjusted the baud rate in the step above.
- Platform studio takes a while to implement and download the design. When you click 'Download Design', it does a number of things, each of which can be performed from the XPS main window. These include – synthesis, implementation (with Place and Route), generating the .bit file, and calling iMPACT to download the design.

- Once it has been downloaded, look at the display on the hyper terminal window. The results should look like Figure 8 below. The read data will depend on the position of the DIP switches.

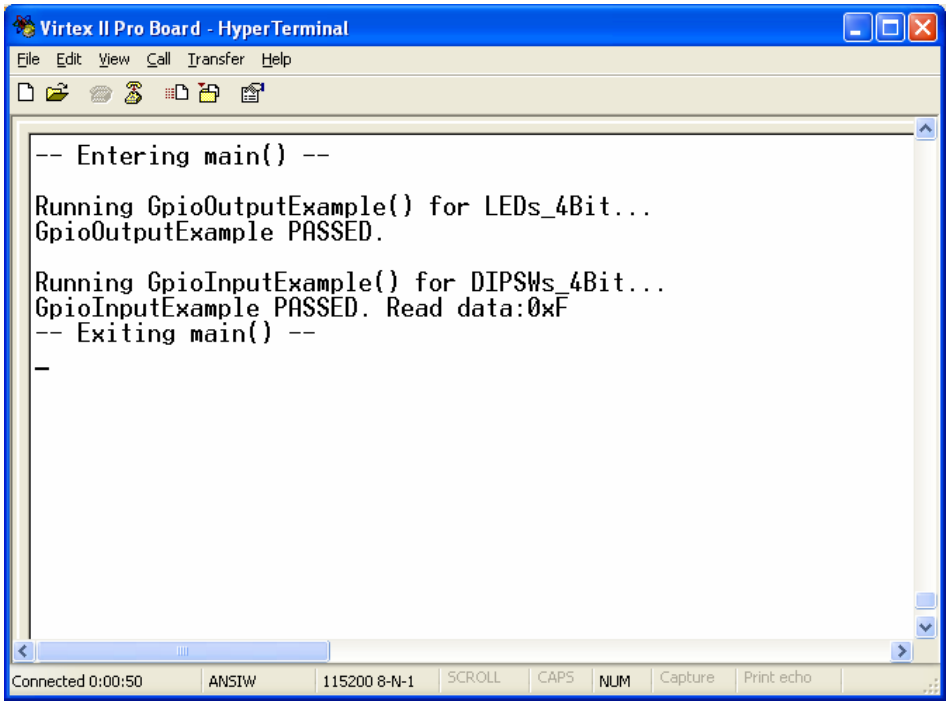

Figure 8: Results from Self-Tests

- Now go through the self-test files generated, 'TestApp_Peripheral.c' and 'xgpio_tapp_example.c', to get an understanding of what it is doing and the pre-generated code it uses to generate the results or perform read/ write operations

Once you are familiar with the generic syntax to read and write from the LEDs and switches, you are ready to write you own code.

Follow the steps below to achieve similar results as before, using Platform Studio this time. In this part, you will program the board to do the following:
Led0 = SW0 & SW1
Led1 = SW0 | SW1
Led2 = SW2
Led3 = flashes when SW3 is off

- In the already open XPS window, on the left hand side, there are 3 tabs – Project, Applications and IP Catalog. Click on Applications, and double click on 'Add Software Application Project'.
- Enter the Project Name as 'lab1_xx'. Set the Processor to be 'ppc405_0'. Click Ok. This creates a new project, where you will write the lab1_xx.c files to achieve the above tasks.
- In **Project: Lab1_xx,** right click on Sources and select 'Add New File…'. Browse to the folder where you have saved your lab and save the filename as lab1_xx.c.

- Open the above file by expanding sources. As is good practice, start the file by writing your name and the date (in a block comment).
- Then proceed by typing in the following code to get you started (or download it from the class web page and paste it into a file).

```
/* lab1_xx.c <your email address and data here>*/

/* Read switches and control LEDs */

/* #include files */

/* define I/O devices */
#include "xparameters.h"

/* define general purpose I/O functions for accessing peripherals */
#include "xgpio.h"

/* define printf */
#define printf xil_printf       /* A smaller footprint printf */

/* Main */

int main(void)
{
        int switches;
        XGpio gpio_leds;
        XGpio gpio_switches;

        printf("\r\nPreparing to read switches\r\n");

        XGpio_Initialize(&gpio_switches, XPAR_DIPSWS_4BIT_DEVICE_ID);
        XGpio_SetDataDirection(&gpio_switches, 1, 0xFFFFFFFF); // input
        XGpio_Initialize(&gpio_leds, XPAR_LEDS_4BIT_DEVICE_ID);
        XGpio_SetDataDirection(&gpio_leds, 1, 0x00000000); // output

        switches = XGpio_DiscreteRead(&gpio_switches, 1);
        printf ("Switches = %d\r\n", switches);
        XGpio_DiscreteWrite(&gpio_leds, 1, ~switches); // leds turn on when low
}
```

- Since two software projects are existent in the same Project, in order to download this new code, you must unselect the earlier self-generated project and select this one. Right-click on **Project: TestApp_peripheral** and uncheck ‘Mark to Initialize BRAMs’. This unselects this project. To initialize the lab1_xx project, right-click on **Project: Lab1_xx** and select ‘Mark to Initialize BRAMs’.

- To increase download speed slightly by preventing iMPACT from searching through all the possible cables that might be connected, go to the folder where the lab is saved. Open the 'etc' folder. Open the 'download.cmd' file using wordpad or EditPlus. In this file, line 2 reads – 'setCable -p auto'. Change this to – 'setCable -p usb21'. Save the file and close it.
- To download, go to **Device Configuration -> Download Bitstream.**
- Downloading the above code should read the input at the DIP switches and output the same onto the LEDs. Remember again that the LEDs turn on when a 0 is output and turn off when the output is 1. Also, for the Base System Builder design, the format for the DIP switches and the LEDs is the 'little endian' format. This means that the LSB is at the lowest address and the MSB is at the highest address.
- If the program does not seem to run, check the linker script. This should be under 'Compiler Options' for the project under consideration. If there is no linker script listed, go to **Software -> Generate Linker Script -> lab1_xx.** Click Ok, and then click generate on the next window. This needs to be done when you add an additional project to the default one already created by the Base System Builder.
- Try downloading the design again. Again, make sure the HyperTerminal window is open to display the results.
- Now, edit this file to include your code to perform the above mentioned tasks for the 4 LEDs. Download your design and test it.

May the Force be with you!

This lab was developed by Anu Kohli.

What To Turn In

1. Did the board complete the self-test? Explain any anomalies.
2. Turn in your Verilog code that controls the LEDs based on the switches. Also turn in the UCF file showing which pins are assigned to which signals. Describe your testing results.
3. Turn in your C code that controls the LEDs based on the switches. Describe your testing results.
4. How long did you spend working on this lab. The time will not affect your grade but will help adjust future labs to a reasonable workload.