Embedded System Design Lab 1

Stephen A. Edwards

Due August 2, 2005

Abstract

Write a C program that counts in decimal on the Digilent Spartan-3 starter board. Learn how to compile and run a program, program the FPGA, and use serial communication for debugging.

1 Introduction

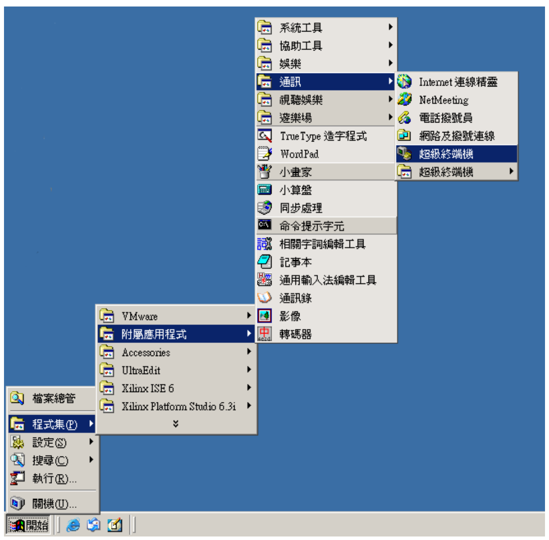
The Digilent Spartan-3 Starter Board consists of a Xilinx Spartan-3 XC3S400 FPGA, which has roughly 400 000 raw gates that can be user-programmed into any configuration. For this lab, you will use a project we have provided for you, consisting of the Microblaze 32-bit microprocessor, a UART, and a soft peripheral that controls the seven-segment LEDs on the board.

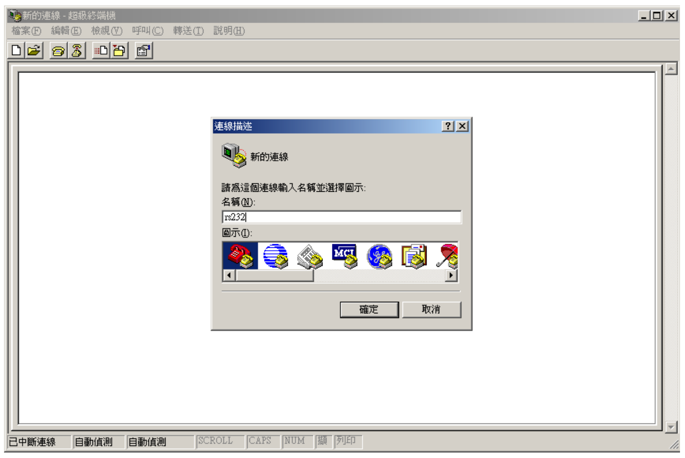
2 Hello World

First, get familar with the tools by compiling and running the sample project supplied by Xilinx.

1. The boards have a serial port on them and the sample project includes a UART (Universal Asynchronous Receiver Transmitter) that can be used to communicate through a modem cable to the workstation. We will use this for debugging since it provides a way to print things from the C program.

   Start Hyperterminal and set it to use COM1 at 9600 baud, no parity, one stop bit.

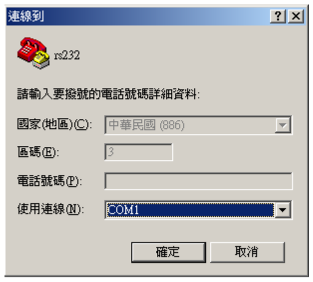




請為這個連線輸入名稱並選擇圖示:

名稱(N): rs232

圖示(I):



rs232

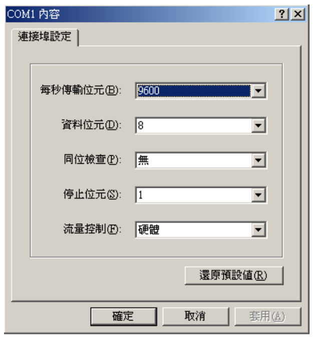
請輸入要撥號的電話號碼詳細資料:

國家(地區)(C): 中華民國 (886)

區碼(E): 3

電話號碼(P):

使用連線(N): COM1



每秒傳輸位元(B): 9600

資料位元(D): 8

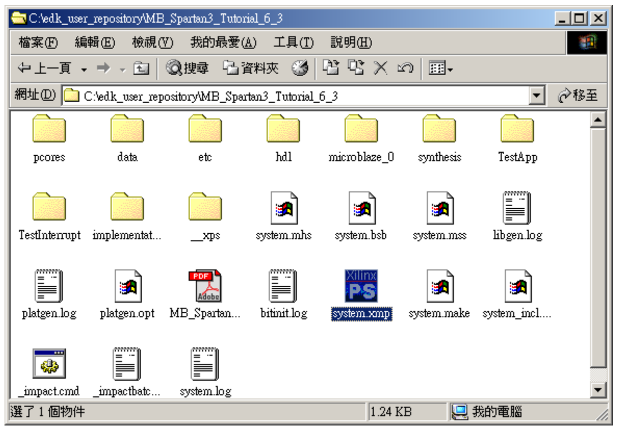
同位檢查(P): 無

停止位元(S): 1

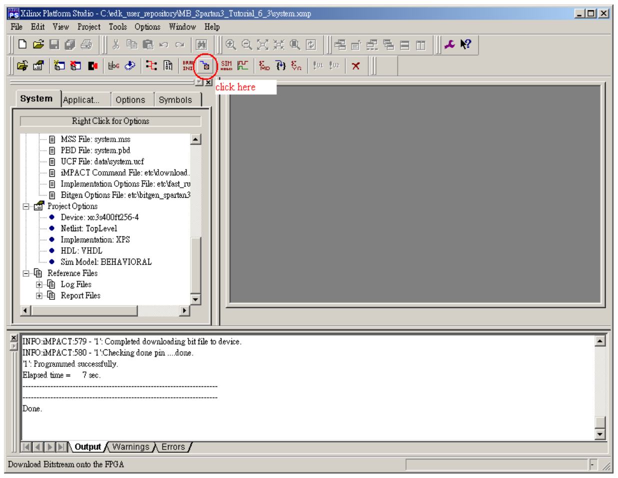
流量控制(F): 硬體

2. Unzip the lab1 source file (on the class home page, http://www1.cs.columbia.edu/~sedwards/classes/2005/emsys-summer/) in the directory `C:\edk_user_repository`. It should create a directory there named `lab1`.

3. Start Xilinx Platform Studio by clicking on the `system.xmp` file in the directory you just unzipped.



4. Start the compilation process and download the program to the board by clicking on the "download" button in the platform studio application. It's on the second row of icons, roughly in the middle. You can also use the Tools→Download menu item.


Xilinx Platform Studio - C:\edk_user_repository\MB_Spartan3_Tutorial_6_3\system.xmp
File Edit View Project Tools Options Window Help
click here
System Applicat... Options Symbols
Right Click for Options
Output Warnings Errors

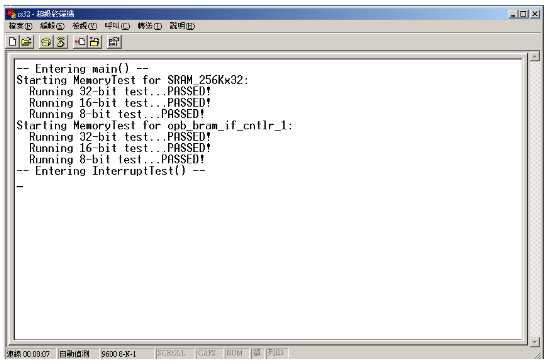
```
MSS File: system.mss
PBD File: system.pbd
UCF File: data\system.ucf
iMPACT Command File: etc\download.
Implementation Options File: etc\fast_ru
Bitgen Options File: etc\bitgen_spartan3
Project Options
    Device: xc3s400ft256-4
    Netlist: TopLevel
    Implementation: XPS
    HDL: VHDL
    Sim Model: BEHAVIORAL
Reference Files
    Log Files
    Report Files
```

```
INFO:iMPACT:579 - '1': Completed downloading bit file to device.
INFO:iMPACT:580 - '1':Checking done pin ....done.
'1': Programmed successfully.
Elapsed time =      7 sec.
----------------------------------------------------------------------
----------------------------------------------------------------------
Done.
```

Download Bitstream onto the FPGA

This takes a long time and generates lots of harmless messages and over a thousand temporary files, but will eventually compile the project into a .bit file suitable for the FPGA on the board and finally download it.

5. If all goes well, the project will be compiled, downloaded, and run on the board. The LEDs should glow for a while and finally a series of things should appear in Hyperterminal and the LEDs should start walking.


m32 - 超級終端機
檔案(F) 編輯(E) 檢視(V) 呼叫(C) 轉送(T) 說明(H)
連線 00:08:07 自動偵測 9600 8-N-1 SCROLL CAPS NUM 擷 列印

```
-- Entering main() --
Starting MemoryTest for SRAM_256Kx32:
  Running 32-bit test...PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
Starting MemoryTest for opb_bram_if_cntlr_1:
  Running 32-bit test...PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
-- Entering InterruptTest() --
-
```

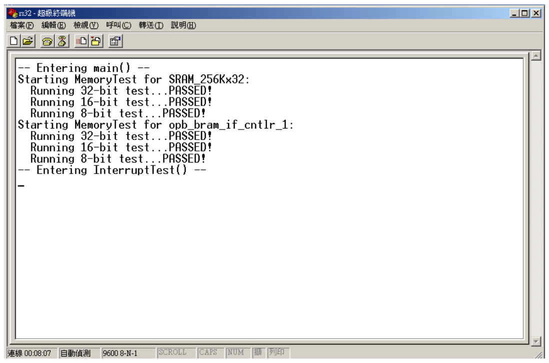
If something goes wrong, make sure your board is connected properly (it should have a power connection from the external power supply, a connection through a parallel cable, and a connection through a serial cable) and powered on. An LED near the power connector lights whenever power is applied to the board.

If you can't get things to work, pester someone for help.

3 What is going on?

A lot. The Xilinx Platform Studio tool is assembling and compiling a small computer system that is then downloaded to the FPGA. Platform Studio also compiles a small C program and downloads it into memory on the FPGA. Finally, it runs the program, which prints things to the serial port and blinks the LEDs.

There are two interesting tabs in Platform Studio: "System," which describes the hardware and peripherals, and "Applications," which describes the software that runs on this hardware.


m32 - 超級終端機
檔案(F) 編輯(E) 檢視(V) 呼叫(C) 轉送(T) 說明(H)
連線 00:08:07 自動偵測 9600 8-N-1 SCROLL CAPS NUM 擷 列印

```
-- Entering main() --
Starting MemoryTest for SRAM_256Kx32:
  Running 32-bit test...PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
Starting MemoryTest for opb_bram_if_cntlr_1:
  Running 32-bit test...PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
-- Entering InterruptTest() --
-
```

The demo project is centered around a single CPU, "microblaze_0." The Microblaze is Xilinx's own soft processor core specifically designed to be placed on an FPGA. It is a 32-bit RISC-style processor. Fortunately, the C compiler for it is good enough so that you do not have to know Microblaze assembly language to build an interesting project.

Under the CPU are three busses: mb_opb, which is the peripheral bus, ilmb, which is the instruction bus, and dlmb, which is the data bus (the Microblaze uses a Harvard architecture). Connected to these busses are various bus controllers (dlmb_cntlr, ilmb_cntlr), an on-chip shared instruction and data memory block (lmb_bram), a serial controller (RS232), a driver for the row of LEDs (LEDs_8Bit), a controller for the off-chip SRAM (SRAM_256Kx32_util_bus_split_0), as well as a timer and interrupt controller.

The `system.mhs` file lists how the hardware should be assembled to create the project. For example, it lists the LED and serial connections, the Microblaze processor, the UART, and the peripheral that controls the LEDs.

The `system.mss` file describes how the software should be assembled for the system, and is less interesting than `.mhs` file. It says to include the UART driver and to connect stdin and stdout of the program to the driver.

The `testApp.c` file (under Applications/Project: TestApp/Sources) contains the `main()` function that is run when the system is downloaded. You will modify this for this lab.

The `data\system.ucf` lists to what pins on the FPGA internal signals should be connected. These pin numbers came from the Digilent board documentation, which lists the role(s) of each pin.

Finally, `etc\bitgen_spartan3.ut` sets some options for generating the final bitstream.

4 The Assignment

Modify the testApp.c file to count in decimal from 0 to 99 on the LEDs. Make sure the numbers from 0–9 don't display a leading 0.

Show your working counter to a TA, have him sign a printout of your solution (i.e., `testApp.c`), and hand that in.

Shorter, elegant, and readable solutions will be scored higher than ones that merely work.