Experiment # 6
Introduction to FPGAs - Detour Signal Lab

1. Synopsis:

This lab introduces the use of Field Programmable Gate Arrays (or FPGAs, for short) for prototyping of digital circuits. Digilab experimental platform from Digilent Inc. will be used to demonstrate the design flow. The platform consists of a **`Xilinx Spartan-II FPGA`** along with various input/output devices like push buttons and LEDs.

2. Introduction:

An FPGA consists of an array of logic blocks connected via programmable interconnect. A typical FPGA contains anywhere from 64 to tens of thousands of logic blocks and an even greater number of flip-flops. Each configurable logic block (CLB) has one or more D-flip flops and some combinational logic such as muxes, etc. The CLB is capable of performing a reasonably complex function. Our objective in this lab is to understand the FPGA design flow. Detailed description of the architecture of an FPGA is given in Addendum #2. Abundant information on FPGAs can be found on the xilinx website: `www.xilinx.com` .

2.1 Xilinx Spartan-II XC2S30:

Xilinx™ Inc. is one of the major FPGA vendors in the market. We will be using the XC2S30 device from Xilinx's Spartan-II family of FPGAs. It has approximately 30,000 gates organized in configurable logic blocks (or CLBs). The XC2S30 device used in this lab comes in a TQ144 package. ("TQ" stands for "**`Thin Quad`**" while 144 represents the number of pins in the package)

2.2 Digilab D2XL Board:

Digilab D2XL™ board from Digilent Inc. (http://www.digilentinc.com/) is the mother board for the Digilent's FPGA testbed used in our lab. The board features Xilinx Spartan-II XC2S30 FPGA and two expansion slots labelled as E and F. These slots are used to interface the D2XL system board with various daughter boards like DIO1 that Digilent produces. The board also has a push button (referred to as BTN_MAIN) and a LED (LED_MAIN). We will generally use the button as system reset and the LED to observe a derivative of the system clock. Details about the board can be found in Addendum #1. Also, you may find Digilab 2XL reference manual on Digilent's website very useful. (http://www.digilentinc.com/)

2.3 Digilab DIO1 board:

Digilab DIO1 is one of the several daughter boards that can be connected to the D2XL motherboard. This board provides a wide range of I/O needed for our labs. It has four (or five on some newer boards) push buttons (labelled BTN1 through BTN4 (labelled BTN1 through BTN5 on some newer boards)), eight LEDs (LD1-LD8), eight switches (SW1-SW8) and four common anode seven-segment displays. Addendum #1 explains the functionality of this board in detail.

The DIO1 reference manual can be found at http://www.digilentinc.com/ . The 8 LEDs (LD1-LD8) on the DIO1 board are connected through an 8-bit latch. We need to enable this latch by connecting its gate (strobe) control (labelled as LDG) to VDD.

3. FPGA Design Flow:

In this section we will discuss the design flow for implementing an ePD-based design on to the FPGA on our Digilab board.

3.1 Design Entry:

As always, we will use ePD to specify the circuit schematics. Note that **now we must use the components from the Spartan/Spartan2 libraries only**. Apart from the usual components needed (like logic gates and flip flops etc.), Spartan2 library has the following special components which we will be needing:

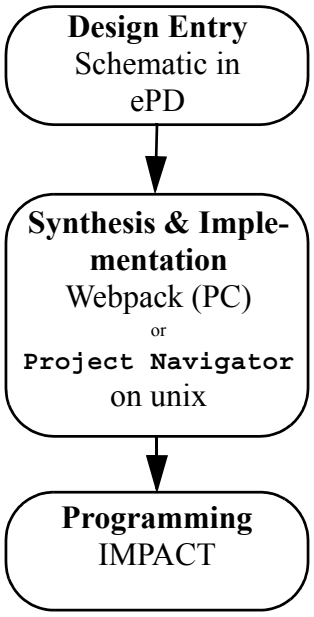


Fig1: Design flow for implementing a design on Xilinx FPGA

Input/Output Pads (IPAD, OPAD): Usually we need to connect the inputs and/or outputs of our circuits to I/O devices such as buttons, switches and LEDs. On the digilab lab board these devices are permanently wired to specific pins of the FPGA. **In order to connect signals (from our core design in FPGA) to the pins of the FPGA we use I/O pads**. The **location (LOC) attribute** of a pad specifies which pin the signal will be connected to. We use the input pads (symbol: IPAD) to connect to input devices such as buttons and switches, and output pads (symbol: OPAD) to connect outputs of our circuits to LEDs or seven-segment displays.

Input/Output Buffers (IBUF, OBUF): Xilinx FPGAs require that we connect buffers to the I/O pads to ensure that signals entering or leaving the FPGA are of appropriate strength. Spartan-II library provides special input buffers (symbol: IBUF) and output buffers (symbol: OBUF) for this purpose.

Global Buffers (BUFGP): Xilinx FPGAs have special buffers called “Global Primary Buffers” (symbol: bufgp) which must be used to buffer incoming signals such as clock, reset, etc. with high-fanout. For the clock pin, for example, after the input pad, you place a global buffer (bufgp) before taking the clock to circuit elements like flips-flops, counters, etc. You can not use general purpose buffers (symbol: buf) or input buffers (ibuf) for the clock signal.

EDIF and EDN: Recall that after having entered the design in ePD (in Viewdraw schematic), we normally used the `Create Digital Netlist` tool to generate a netlist for our design. This netlist file (with .vsm extension) is used by ViewSim to simulate the design using a command file. Similarly, to implement the design on a Xilinx FGPA, we will generate a netlist of our circuit in

the *Electronic Data Interchange Format* (EDIF) or *Electronic Data Netlist* (EDN) format. Unlike the VSM format which only ePD can understand, EDIF is a standardized format that the Xilinx software tools can also understand. Usually, EDIF or EDN files have the extension of "**`.edn`**".

3.2 Synthesis & Implementation:

Once the circuit has been specified (in a schematic file) and a standard netlist (.edn file) has been produced, Xilinx Synthesis Tool (XST) can use this netlist to decide how to use the CLBs and the programmable interconnect to implement our desired circuit. The process of interpreting the design, simplifying the circuit and then creating an implementation based on certain target FPGA, is called *synthesis and implementation*. We can use the Xilinx Webpack on Windows platforms or Xilinx Project Navigator on Unix platforms. Your scf accounts (on aludra/nunki) are setup to run Xilinx Project Navigator (command is "`xilinx &`"). The output of this synthesis tool is a configuration bit file (with an extension of ".bit"). Details of how to use these synthesis tools are given in the procedure section.

3.3 Programming:

We can program (or configure) the FPGA by downloading the configuration bit-file generated by the synthesizer onto the FPGA through the parallel port of a PC. We will use Xilinx's iMPACT configuration/programming software to download the bit file onto our FPGA board.

4. Description of the Circuit:

You all know the detour signal at road repair sites. It normally has four groups of lights (GL, G1, G2 and GR) controlled as shown to indicate "detour to the right" or "detour to the left". Assume that we have a L/$\overline{R}$ switch to choose left or right signal. The state machine looks at this switch only when it is in the Idle state and then performs the required (left or right) sequence and comes back to Idle state.

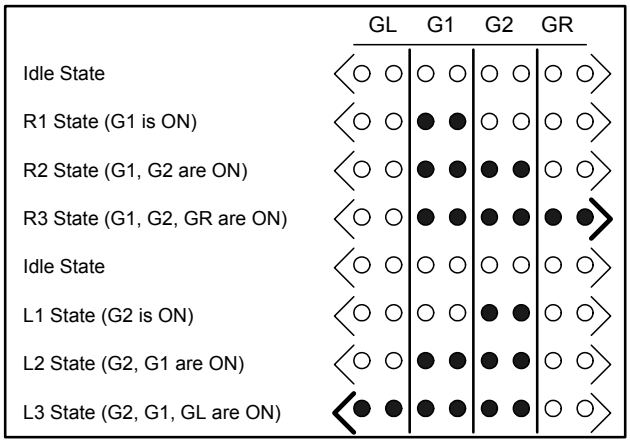


Fig 2: Four groups of LEDs in different

In this lab we will implement the 7-state controller using D-flip flops in One-hot method. The state machine given below is almost complete. Some obvious state transition conditions have been intentionally left out. Complete the state diagram before proceeding. (Recall that an unconditional state transition can be labelled with "1" as the condition)

For ease of understanding, we usually divide the state machine controller into three parts: State Memory (SM), Next State Logic (NSL), and Output Function Logic (OFL).

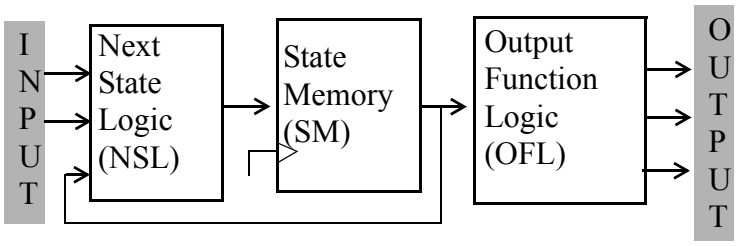


Fig 3: Block diagram showing the three parts of a state machine

Use switch 1 (SW1) on the Digilab board to exercise the left/right detour selection. Use the LEDs (LD1 through LD8, in groups of two each) to create the detour arrow growing towards left or growing towards right. Button on the main D2XL board (BTN1_MAIN) could be used as the System Reset.

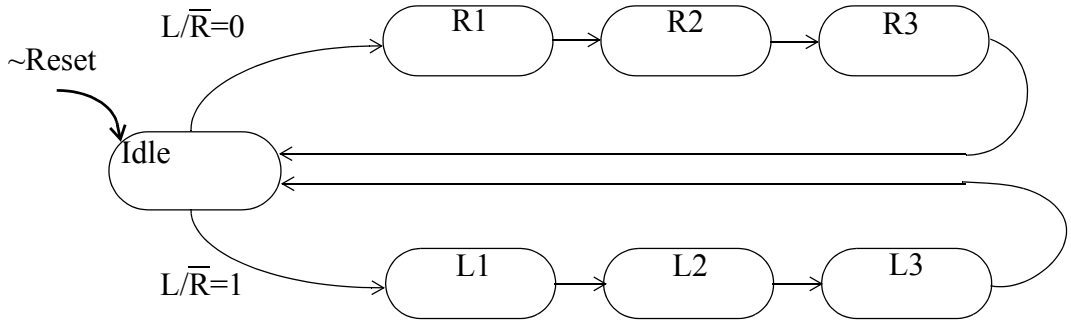
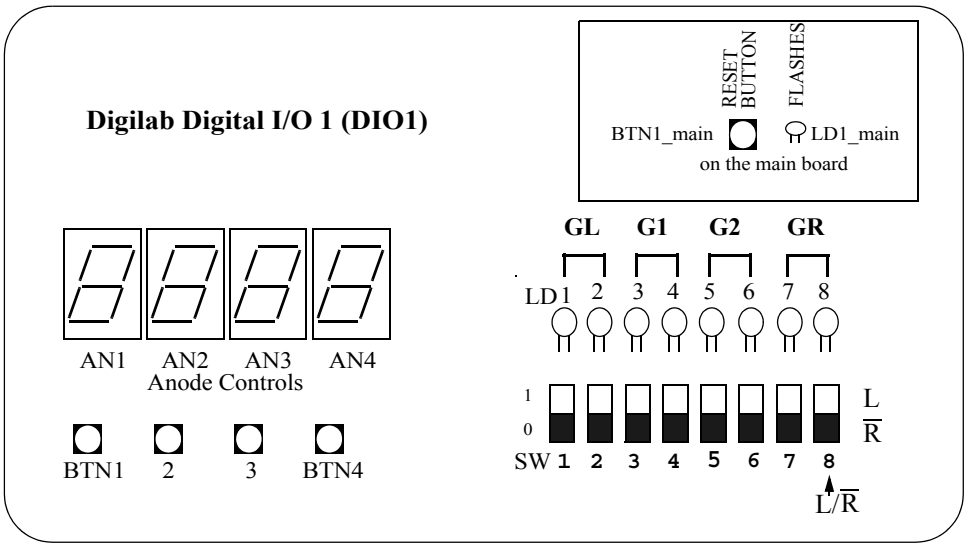


Fig 4: State diagram for the Detour Signal design

The layout of Digilab Digital I/O 1 (DIO1) (together with I/O resource assignments for this lab) is shown below:



5. Prelab:

Q 5. 1: What does “FPGA” stand for? (2pts)

Q 5. 2: What is the name of the FPGA family that we are using in this lab? (2pts)

Q 5. 3: Which FPGA of this family are we using? (2pts)

Q 5. 4: What does TQ144 denote? (2pts)

❑ Speed grade ❑ Package ❑ Serial number ❑Model number

Q 5. 5: The **state memory** is usually implemented using: (4pts)

- ❑ And-Or gates
- ❑ RAM
- ❑ Flip flops
- ❑ All of the above

Q 5. 6: Next State Logic (NSL) is a purely _____ (*combinational/sequential*) circuit. (4pts)

Q 5. 7: We will implement the detour signal controller using One-Hot method. How many D-flip flops are needed to implement the controller? (4pts)

Q 5. 8: “One-hot” refers to: (4pts)

- ❑ A. A state encoding scheme
- ❑ B. A method of designing signal light controllers
- ❑ C. A number system just like binary, decimal and hexadecimal systems.
- ❑ D. Both A and C above.

Q 5. 9: The three steps in FPGA design flow and their inputs/outputs are: (6pts)

Step	Name of the Step	Input	Output
1	Design Entry	Schematics entered by the designer	.edn file
2			.bit file
3			programmed FPGA

6. Procedure:

6.1 Import the necessary files into your account

```
cd ~/pv/ee201l
~eeview/ee201l_s2_detour.shar
```

This brings the following files.

```
sch/ee201l_s2_detour.1
sch/ee201l_s2_detour.2
cmd/ee201l_s2_detour.cmd
```

The first sheet of the schematic (`ee201l_s2_detour.1`) contains the incomplete state machine implementation while the second sheet (`ee201l_s2_detour.2`) contains the circuitry necessary to interface the design to the I/O devices on the Digilab board. **You are required to complete the schematic on both sheets**.

6.2 After completing the schematic, simulate the design using the command file (`cmd/ee201l_s2_detour.cmd`). Note that this step is not necessary, but we want to make sure that our design is working before we download it onto the FPGA. Study the waveform carefully to see if the design is working correctly.

After verifying the functionality of the design through simulations, we would want to implement it on the Digilent board. Next few steps detail the procedure to implement a design on to the Xilinx Spartan II FPGA on the Digilent board.

6.3 Create a netlist of the design in the "Electronic Data Interchange Format" (EDIF) using the **`Export EDIF (common)`** tool under ViewDraw's `Tools` pull down menu (see figure (a) below). This will open the `EDIF Interface` dialog box (figure (b) below).

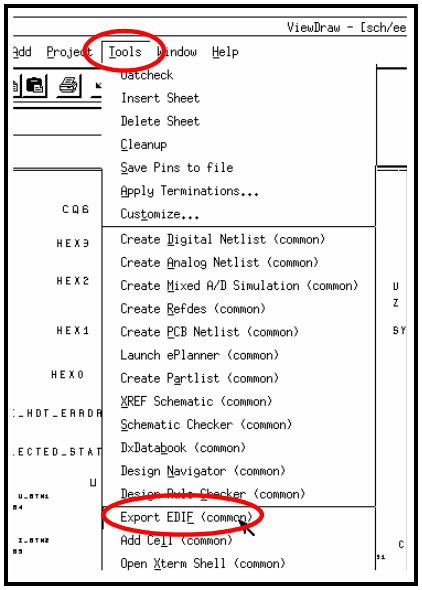


Figure (a): Export EDIF tool

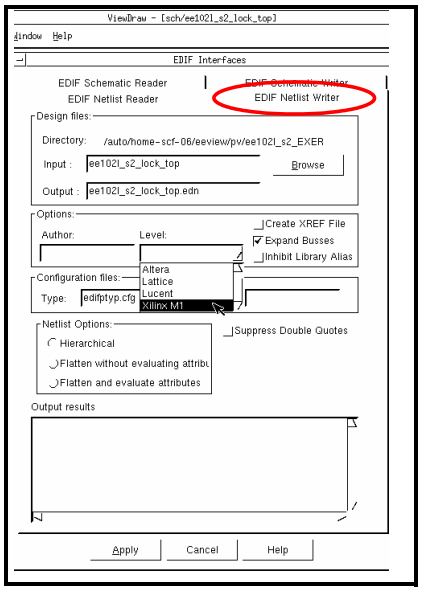


Figure (b): EDIF Netlist Writer

In this dialog box, go to the **EDIF Netlist Writer** tab and select the input design (ee201l_s2_detour in this case). Note that the Output field will automatically be filled with the same filename as the input file but with extension as .edn. Select the Level as Xilinx M1 and click Apply (all other options should be left as to default). Make sure no errors are reported in the Output results window. A file ee201l_s2_detour.edn would be created and saved in your project directory (pv/ee201l).

6.4 The next step is the generation of the configuration file for the FPGA (or, the "bit" file). As mentioned in Section 3, in order to synthesize your design you have the choice of using Xilinx Webpack software on the PCs[1] or Xilinx Project Navigator on Unix. Here we present the procedure for Webpack only. Details about using the Project Navigator software on Unix are given in Appendix I.

6.5 Create the directory C:\WebPACK_projects if it was not created previously. Invoke Xilinx webpack. Start a new project. Let us name this project ee201l_s2_detour . Create this project under C:\WebPACK_projects. The input design format is EDIF and the FPGA is XC2S30 TQ144 5C. (XC2S30-5TQ144). A directory called C:\WebPACK_projects\ee201l_s2_detour is automatically created. (Use windows explorer to confirm that the directory is created) Move the file **ee201l_s2_detour.edn** from your pv/ee201l directory on aludra/nunki to this directory (using Ws_Ftp available from USC-ISD or any other ftp software you like).

6.6 Use Project => Add Source to add the file **ee201l_s2_detour.edn** as the source file for the project. You will see **ee201l_s2_detour** in the Sources in Project window pane. Keep this selected to see the right (proper) associated processes in the Processes for Current Source window pane.

6.7 In the Processes for Current Source window pane, right click on Generate Programming file and select Properties... Click the Startup options tab. Specify JTAG clock for Start-Up Clock. Click OK and close the dialog box.

6.8 Make sure that Digilab boards (D2XL connected to DIO1) are connected to the PC via parallel cable. Also, the boards should be powered up using a 6V DC power supply. The SW1 switch (PORT - JTAG switch) on the main D2XL board should be set to JTAG side.

6.9 In the Webpack Project Navigator window expand the "Generate Programming File" tab under the Processes for Current Source pane. Double click on the "Configure Device (iMPACT)" tool. This will start the synthesis process (the process may take a few minutes).

6.10 If the design has been synthesized successfully, you will see green colored check marks (or exclamatory marks) next to each process tab. A programming bitstream (ee201l_s2_detour.bit) is generated in the directory C:\WebPACK_projects\ee201l_s2_detour if the synthesis and implementation steps succeed.

1. To install Xilinx Webpack on your own PC, please follow the instructions contained in the "Xilinx Webpack - Downloading and testing at USC" document posted on the blackboard.

6.11 When the synthesis has completed successfully, Xilinx Programming tool iMPACT will open and a wizard will guide you through the set up of your board. All default selections (Parallel port LPT1, etc.) are valid for our setup. Actually the wizard comes up only if you invoke iMPACT tool separately. Now the tool knows what bit file, etc. When the wizards finishes, you will see an icon representing the FPGA chip in the main iMPACT window. Download the programming bit stream (`.bit` file) by selecting the FPGA icon and right click and select program (or by choosing `Operation -> Program`). You should see a progress indicator as the bitstream is downloaded on the FPGA.

6.12 If programming is successful, the LD1 on the main D2XL board starts flashing. You can check the operation of your design by operating SW8 and seeing the right sequence of LEDs glowing to indicate right detour or the left detour. Now it is time to celebrate!!

7. Lab Report:

Name:_____	Date: _____
Lab Session: _____	TA's Signature: _____

For TAs: Prelab (out of 30): ____ Implementation (out of 40): _____ Report (out of 30): ___
Comments:

Q 7. 1: What are the two different D-flip flops that we are using in this lab? What is the difference between them? (2pts)

Q 7. 2: Name 2 more D-flip flops that are available in the Spartan library and, in one line, explain how are they different from the ones that we are using in this lab.(8pts)

Hint: Software Manuals available at http://support.xilinx.com explain the various library components available in the Spartan library. (`Software Manuals->Library Guide-> Design Elements`). Note that according to Xilinx's naming convention, D-flip flops are named as FDxx.]

Q 7. 3: Which pin of the FPGA is connected to the clock generator on the D2XL board?. (4pts)

Q 7. 4: What are the two components that we MUST attach if we want to use one of the output devices (let's say an LED) on the board? Draw the schematics and label the components. (4pts)

Q 7. 5: Is the pin number associated with an `ipad` (input pad) or `ibuf` (input buffer)? (2pts)

Q 7. 6: Assume that the frequency of the clock entering the FPGA is 25MHz. Notice that we are dividing the input clock by using counters. Calculate the frequency of the divided clock that triggers the detour signal state machine. (5pts)

Q 7. 7: Why are we dividing the clock? (5pts)

Appendix I: Synthesis using Project Navigator from Xilinx

1. Make sure that your account is setup properly for Xilinx tools on the unix system. It should have been setup when you setup your account for ePD (when you did `~eeview/ee201l_ePD.setup`). Do the following on your unix account to verify that your account's setting. Note: "xilinx" is the command to invoke the xilinx tool on UNIX.

```
unix> which xilinx
```

You should receive the response similar to

```
unix>/usr/usc/xilinx/6.2i/bin/sol/xilinx
```

If this does not happen, approach your TA or Prof. Gandhi Puvvada (gandhi@usc.edu).

2. Create the design directory **`ee201l_s2_detour`** under the **`xilinx`** directory in your account and move the .edn file to this directory by using the following unix commands:

```
unix> cd ~/xilinx
unix> mkdir ee201l_s2_detour
unix> cd ee201l_s2_detour
unix> mv ~/pv/ee201l/ee201l_s2_detour.edn .
```

3. Start vncserver on unix if you are on a PC. Open vncviewer on your PC.

4. Invoke the Xilinx Project Navigator by typing the command **`xilinx &`** at the unix prompt from the design directory you just created **`~/xilinx/ee201l_s2_detour`**. The Xilinx Project Navigator window opens up. This look exactly like the Project Navigator you invoke on a PC using the Xilinx WebPACK.

5. Create a new project `ee201l_s2_detour` (`File->New Project`). In the `New Project` dialog box, in the "Project Location:" make sure that your **`~/xilinx/ee201l_s2_detour`** directory shows up. If not, click on the on **`Browse`** button (appears as **`...`** on unix) next to it and select **`ee201l_s2_detour`** and click **`OK`** (it could take several minutes to browse through the UNIX directory structure starting from /auto). On the left side box titled "Project Name:" the **`ee201l_s2_detour`** should appear. In the "Top-Level Module Type:" box, choose **`EDIF`**. Click **`NEXT`**.

In the `next` dialog box, for the "Input Design:" browse and select **`ee201l_s2_detour.edn`** . Leave the "Constraint File:" box empty. Click **`NEXT`**.

A warning window pops up saying "`Unable to find device string in the specified input file. A default device type will be set.`" Click **`OK`** .

In the `next` dialog box, specify the **`Part`** by selecting:

```
Device Family: SPARTAN2
Device: XC2S30
Package: tq144
Speed Grade: -5
```

Click `Next` and then in the next dialog box, click `Finish` . This will return you to the Xilinx Project Navigator window.
Ignore the error message at the bottom, reading "Error occurred during the initialization of VM, Could not reserve enough space for object heap".

6. Now, specify the start-up clock as JTAG Clock (instead of Cclk which is default) by doing the following :
In the "Process for Source:" window pane in the middle, right click on "Generate Programming File" and select "Properties" at the bottom.
In the Process Properties dialog box, select the **Startup Options** tab .
Set the **FPGA Start-up Clock** to **JTAG Clock**. The default is **CCLK** and you need to use the pull-down button on the side to change **CCLK** to **JTAG Clock**. Click OK to return to the main window.

7. In the "Process for Source:" window pane, right click **Implement Design** and select **Run**. The bottom window displays messages as the tool runs and (hopefully) it returns at the end successfully.

8. Expand (if it is not already expanded) the "Generate Programming File" in the "Process for Source:" window pane. Right click **Programming File Generation Report** and select **Run**. The bottom window displays messages as the tool runs and (hopefully) it returns at the end successfully and displays the report in the top right window.

9. We are not so much interested in the report itself but we are interested in the **.bit** file generated in that step. Go to your unix window (to the directory ~/xilinx/ee201l_s2_detour) and confirm that a **.bit** file (ee201l_s2_detour.bit) was produced and also that it was produced '*recently*' by checking the data stamp of the file.

```
unix> ls -lt *.bit
```

10. FTP the file **ee201l_s2_detour.bit** to a PC to which your FPGA board is connected. Perhaps, you would like to create a directory called Bit_Files under C:\WebPACK_projects and ftp this bit file into that directory.

11. Power the FPGA board. To download the **.bit** file on to the board using the **iMPACT** utility from the Webpack suit, do the following:

11.1 Start iMPACT on your PC.

```
Start => Programs => Xilinx ISE => Accessories => iMPACT
```

11.2 In the iMPACT Project dialog box, select create a new project (.ipf). Browse and name the project **ee201l_s2_detour.ipf** under the directory C:\WebPACK_projects\Bit_Files. Click OK.
11.3 In the Operation Mode Selection dialog box, keep the Configure Devices selected. Click Next.
11.4 In the Configure Devices dialog box, keep the Boundary-Scan Mode selected. Click Next.
11.5 In the Boundary-Scan Mode Selection dialog box, keep the Automatically connect to cable and identify Boundary-Scan chain selected. Click Finish.
11.6 The tool should find and display **xc2s30** FPGA. Click OK in the information box. Assign the configuration file **ee201l_s2_detour.bit** to the FPGA. Right click on the FPGA, select **Program** and under Program Options, click OK. After downloading is complete, it displays "Programming Succeeded". When you exit iMPACT, it asks you to save the project as **ee201l_s2_detour.ipf** . Say Yes. It will be easy to download the same file next time.