Lab 1. Using the Integrated Software Environment (ISE)

Objective

The aim of the lab is to become familiar with ISE, by using it as means to step through the FPGA design flow. We will use ISE to light up the LEDs on the XUP board, depending on the status of the DIP switches on the board.

This task will be performed exclusively by the FPGA hardware, by programming it in Verilog.

Procedure

1. Launch the ISE Project Navigator and create a new design project.
   Select **Start → Programs → Xilinx ISE 8.1i → Project Navigator**
   Alternatively, double click on the ISE icon on the desktop.

   In the Project Navigator, select **File → New Project**
   The New Project Wizard opens:

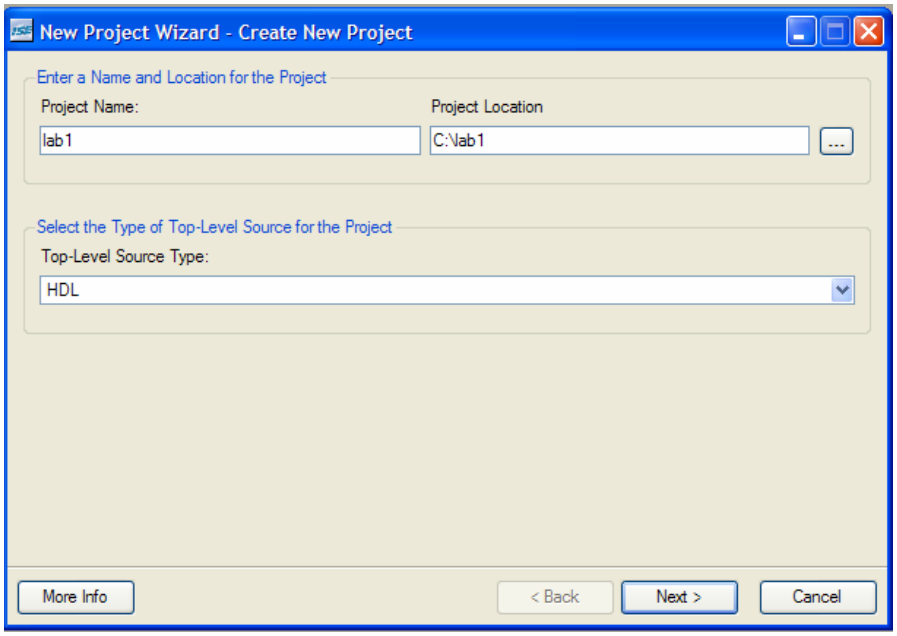

Figure 1. Create New Project

   Select a Project Location, say lab1 in home directory.
   Select a Project Name, say lab1.
   Chose HDL as the Top-Level Source Type and click Next >

2. The Device and Design Flow Dialog appears as below:
   Select the following options and click Next >

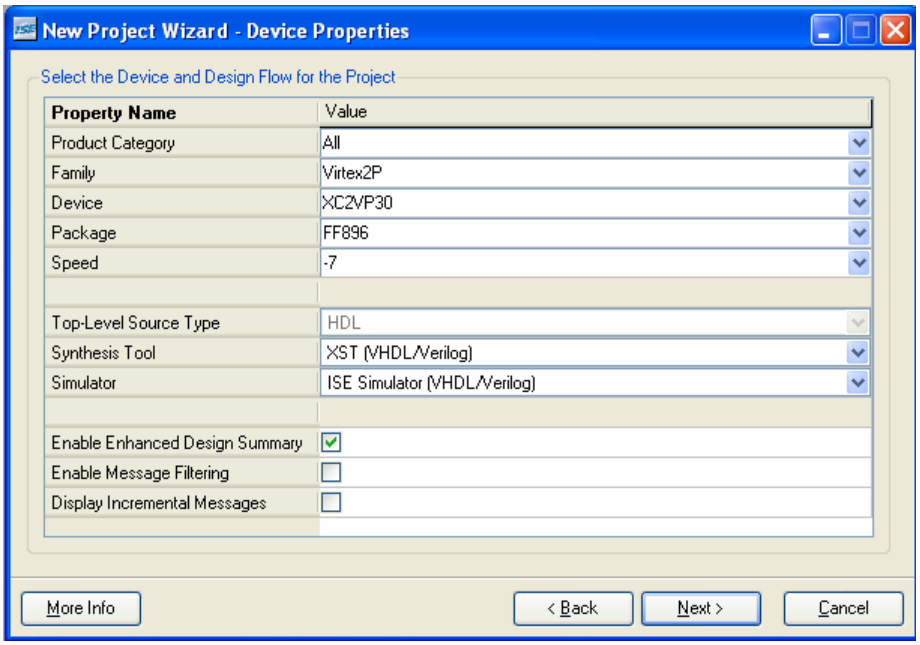

Figure 2. Device Properties

Device Family: **Virtex2P**
Device: **XC2VP30**
Package: **ff896**
Speed Grade: **–7**
Synthesis Tool: **XST (VHDL/Verilog)**
Simulator: **ISE Simulator**

3. The Create New Source Dialog appears as shown below:

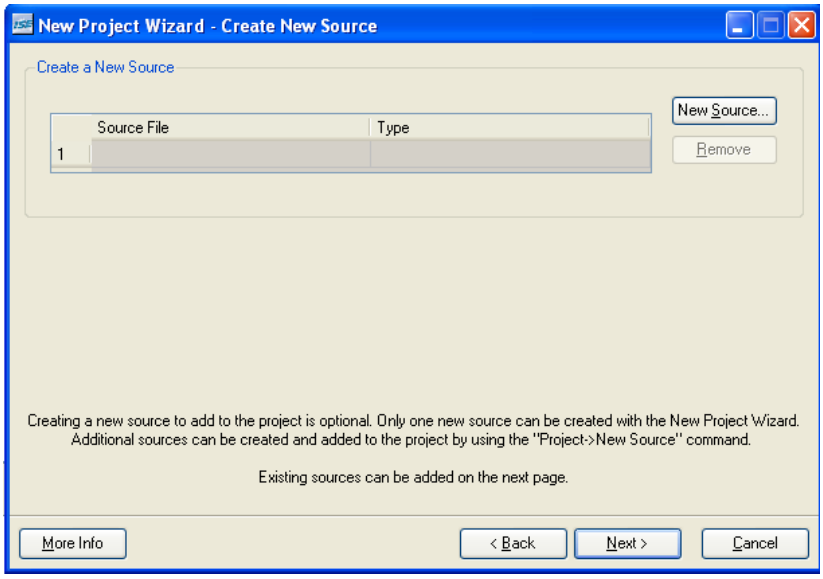

Figure 3. New Project Wizard

Click on New Source to add a new source file. This will open the ‘New Source Wizard – Select Source Type’ window. In Filename field, type ‘switch.v’. Select Verilog Module as Source type and click next.

Now, the ‘New Source Wizard – Define Module’ window will open, as shown in Figure 4. Add LEDS and SWITCHES as ports for the source file.

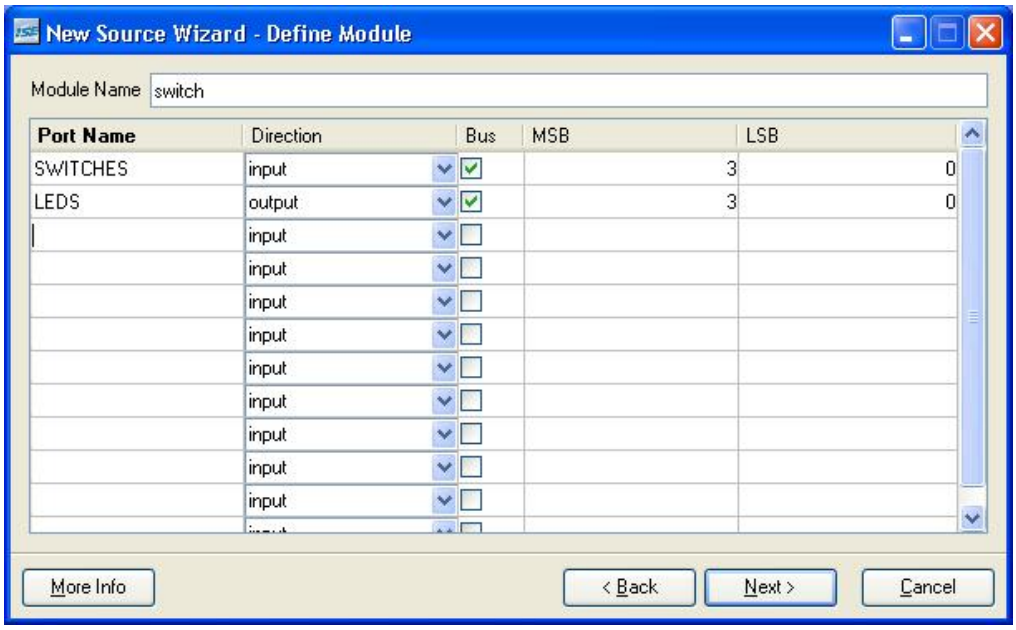

Figure 4. New Source Wizard

LEDS and SWITCHES are each 4bit ports. They will be used to interface to the 4 DIP switches and LEDs on the XUP board. Click Next. The next window will show the summary of design created, click Finish.

4. At this step, we have added a Verilog source file switch.v; which contains a module switch.v, with input and output port declarations. Click Next.

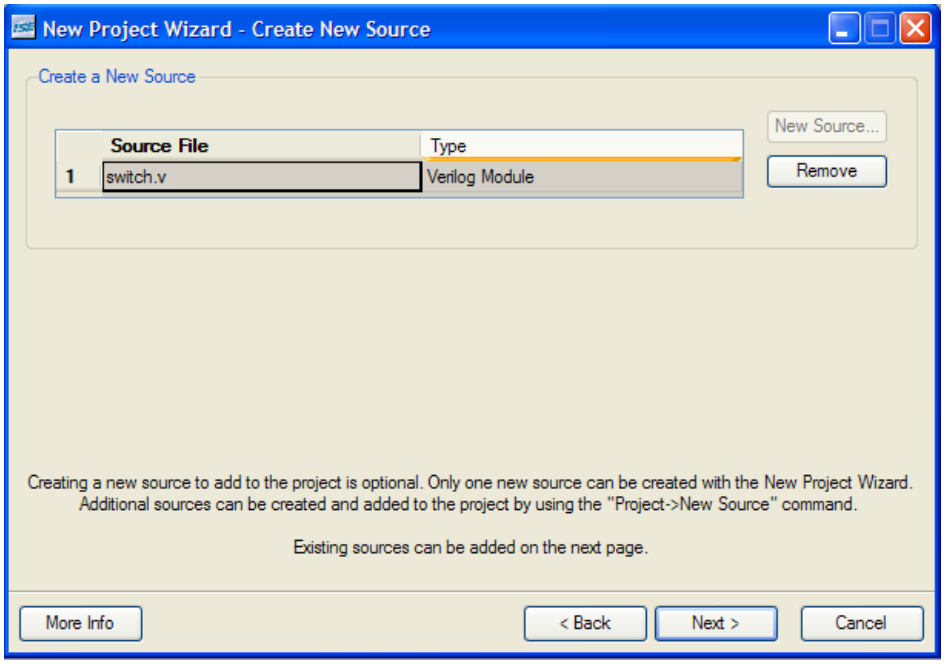

Figure 5. New Project Wizard – Create New Source

5. This will open 'New Project Wizard – add existing sources'. We do not have to add an existing source file. Click Next.
   The next window will show the summary of project that would be created. Click Finish.

6. From the Sources window, open the switch.v file. The file will contain a module with the input and output port declarations mentioned in Step 3.
   We will add Verilog code to provide the desired functionality (i.e. to turn on LED[i] when Switch[i] is high).

   To do this, add the following line:
   assign LEDS[3:0] = SWITCHES[3:0];

   The resulting verilog module should be as follows:

   ```
   module switch(SWITCHES,LEDS);
      input [3:0] SWITCHES;
      output [3:0] LEDS;

      assign LEDS[3:0] = SWITCHES[3:0];
   endmodule
   ```

   Click on File → Save to save your changes.

7. Click on the switch.v file in the Sources window, the Processes Window appears as shown in Figure 4. The Process Window lists a sequence of steps that we need

to perform in order that the code in the switch.v file can be run on the FPGA on the XUP board.

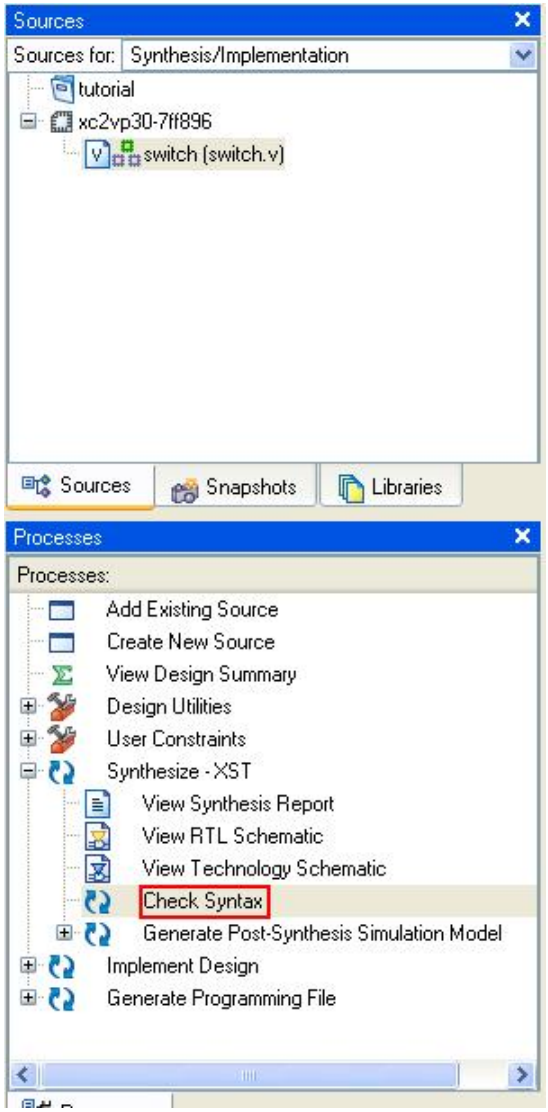

Figure 6. Sources and Processes windows

Double click on Check Syntax. A  should appear next to Check Syntax once compilation is complete. If you get any compilation errors, please correct them before continuing.

8. We now need to create the User Constraint File (.ucf) containing the location of the DIP switches and LEDs on the XUP Board. The ucf file will be used to connect signals in the Verilog file (LEDS[3:0] and SWITCHES[3:0] in our case) to pins of the FPGA chip, which are hardwired to the LEDS and DIP switches on the XUP board.

Expand the User Constraints tab in processes window and double click on Assign Package Pins. When prompted to add a UCF file to your project, click Yes. Xilinx PACE will open up.

Provide the location of LEDS and SWITCHES as shown in Figure 5.

I/O Name	I/O Direction	Loc	Bank
LEDS<0>	Output	AC4	BANK
LEDS<1>	Output	AC3	BANK
LEDS<2>	Output	AA6	BANK
LEDS<3>	Output	AA5	BANK
SWITCHES<0>	Input	AC11	BANK
SWITCHES<1>	Input	AD11	BANK
SWITCHES<2>	Input	AF8	BANK
SWITCHES<3>	Input	AF9	BANK

Figure 7. PACE: User Constraint File

For example, LEDS<0> is an output, connected to the AC4 pin of the FPGA chip. This information is available on the XUP board documentation.
Save changes and close the PACE window.

9. In the Process window, click on Generate Programming File.
   This will run all the steps necessary to create the bitstream that can be downloaded on the board to program the FPGA. Running these processes may take several minutes; progress is indicated by the spinning icon and output to the console.
   When a process completes, a appears next to the Generate Programming File text. The steps that are run before the Programming File is created are: synthesis of the Verilog, mapping of the result to the FPGA hardware, placement of the mapped hardware, and routing of the placed hardware.

10. We now need to download the bitstream to the FPGA on the XUP board. Turn on the power to the Virtex-II Pro board.
    In Processes window, under the 'Generate Programming File' tab, double click on Configure Device (iMPACT). This will open iMPACT (the tool which performs the bitstream download, also referred to as device configuration).
    After scanning the board, iMPACT finds three devices. The first, "xcf32p", is the board's non-volatile PROM. The second, "xccace", is the System ACE controller (for compact flash). The third, "xc2vp30", is the actual FPGA. We only need to program the FPGA.

Figure 8. iMPACT: Program FPGA

You should now see the "Assign New Configuration File" dialog box. Do not assign configuration files to the first two devices; skip over them by clicking “Bypass”. The third device, the FPGA, is programmed with a .bit file (the bitstream to be downloaded). iMPACT should automatically show your project directory and a file, switch.bit (which was produce after ‘Generate Programming File’). Select this file and click Open, then click OK on the next dialog box to assign the configuration file to the FPGA. IMPACT:2257 warning will pop up, ignore this warning.

In the iMPACT main window, right-click on the FPGA (xc2vp30) and select Program. The Executing Command box should appear, followed by a blue "Program Succeeded" message. The red "Done" LED on the board (D4) should light up, indicating that the FPGA has been programmed.

11. Toggle the DIP switches, and the corresponding LEDs should turn on and off.

Deliverables:

1. Demonstrate your work to the TA after downloading the bitstream for the design created using the steps provided in the manual. [4 points, out of which 2 points are for the report]

2. Implement a 4-bit counter using the LEDs (You do not need the switches for this exercise). The count value should update approximately every 1 second.[6 points, out of which 3 points are for the report]

Hints:

- Add clock and reset as input pins to your verilog module.
- Using PACE, specify the location of clock and reset as *AJ15* and *AH5* respectively.
- After generating the ucf with PACE, open it using a text editor. It should be present in your lab1 directory. Add the following lines in the file to provide the time period of the clock (if your clock pin is called CLOCK):

```
NET "CLOCK" TNM_NET = "CLOCK";
TIMESPEC "TS_CLOCK" = PERIOD "CLOCK" 10 ns HIGH 50 %;
```

- The frequency of the clock on the XUP board (connected to the FPGA chip through pin *AJ15*) is 100MHz (which means its time period is 1/100M = 10ns). Thus, if we change the status of LEDs at every posedge (/negedge) of clock, the LEDs will appear to be blinking. A solution is to declare a n-bit wide signal which increases by 1 on every

posedge (/negedge) of the system clock. The LEDs can be assigned the 4 MSBs of this signal.

- Also, the reset (connected to pin *AH5*) is active low. Take this into account while writing your verilog code (When reset is low, assign your n-bit register to 0).

Lab Report Format:

The lab report should contain the following information:

- Aim of the experiment,
- Hardware and software required, and
- Commented code including the UCF files.

1 lab report per group is required for all the labs.