HOCHSCHULE
RHEIN-WAAL
Rhine-Waal University
of Applied Sciences

LABORATORY 2

Model-based Hardware Design

Lab notes prepared by Mr. Thomas Grunenberg

Lab notes revised by: Prof. Andy Stamm

Contents

1. Safety instructions

ESD Warning

Caution Although this product has been designed to be as robust as possible, ESD (Electrostatic Discharge) can damage or upset this product. This product must be protected at all times from ESD. Static charges may easily produce potentials of several kilovolts on the human body or equipment, which can discharge without detection. Industry-standard ESD precautions must be employed at all times.

The NI Digital Electronics FPGA Board is designed and intended for use as a development platform for hardware or software in an educational/professional laboratory environment. To facilitate usage, the board is manufactured with its components and connecting traces openly exposed to the operator and the environment. As a result, ESD sensitive (ESDS) components on the board, such as the semiconductor integrated circuits, can be damaged when exposed to an ESD event. To indicate the ESD sensitivity of the NI Digital Electronics FPGA Board, it carries the symbol shown below.

Handling the NI Digital Electronics FPGA Board can damage the board components if ESD prevention measures are not applied. **Before handling or setup, equalize your potential with the board by touching one of the integrated ESD discharge pads.** During all handling and setup, ESD prevention measures must be applied. In addition, the NI Digital Electronics FPGA Board should be handled by the edges. Touching exposed circuits, components or connectors could result in an ESD event.

2. Introduction to NI Elvis

What is NI Elvis?

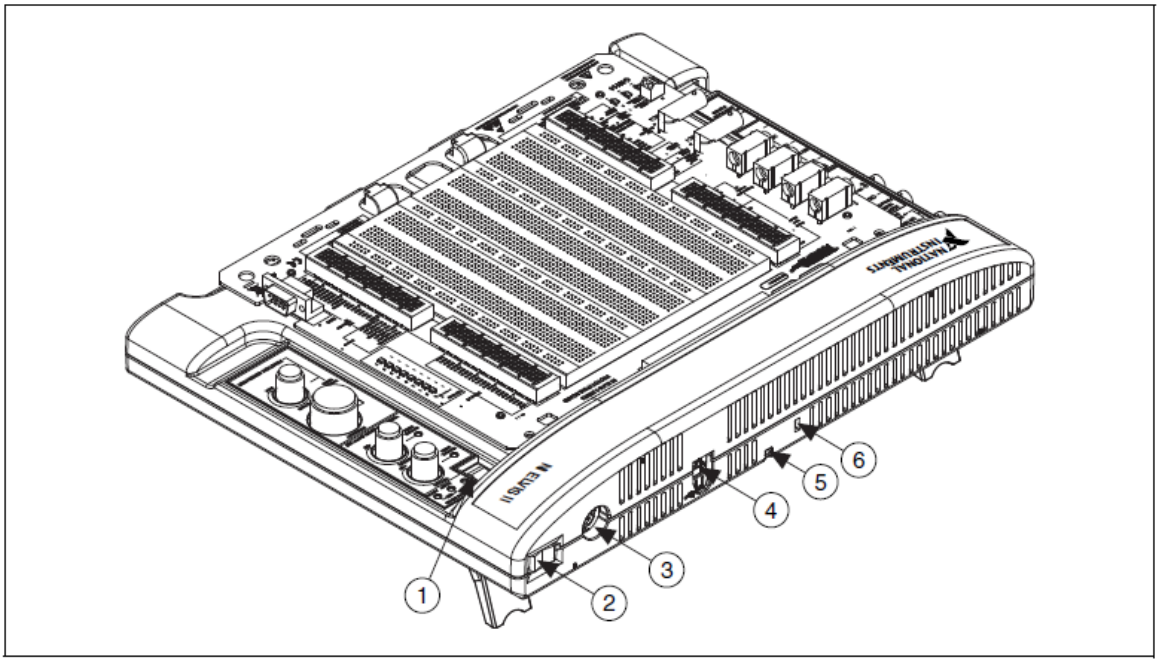
The National Instruments Educational Laboratory Virtual Instrumentation Suite II Series (NI ELVIS II Series) is a LabVIEW-based design and prototyping environment for university science and engineering laboratories. This document explains how to set up and configure NI ELVIS II Series.

NI ELVIS II Series has the following features:

- USB-based workstation with a removable prototyping board for use in circuit development and Experimentation

- Integrated instruments for computer-based measurement and control including the following:

– Multi-channel data acquisition capabilities including both analog and digital I/O
– Digital Multimeter (DMM)
– 2-channel oscilloscope with a sampling rate of 1.25 MS/s
– Function generator
– Fixed and variable power supplies
– 2- and 3-wire impedance analyzer



1	Prototyping Board Power Switch	4	USB Connector
2	Workstation Power Switch	5	Cable Tie Slot
3	AC/DC Power Supply Connector	6	Kensington Security Slot

Figure 1: Rear View of NI ELVIS II Workstation (NI ELVIS shown)

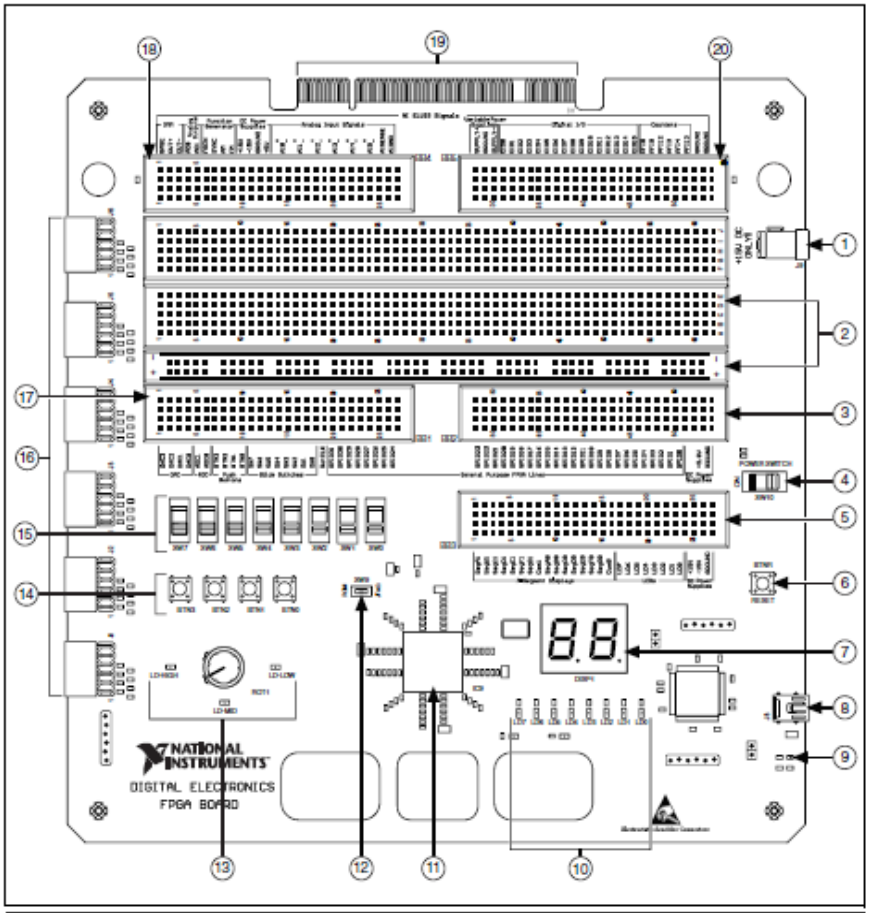
3. Introduction to NI FPGA

The NI FPGA board is an extension to the NI ELVIS which allows to program and test FPGA based prototyping.

3.1. The NI FPGA board features (shown in Figure 2):

- Eight slide switches, SW0 through SW7 (15)
- Four momentary-contact push-buttons, BTN0 through BTN3 (14)
- Eight individual surface-mount LEDs, LD0 through LD7. Each LED is connected on one side through a 390Ω current-limiting resistor to the power line (10)
- Two digit seven-segment display, DISP1, in a common cathode configuration. (7)
- Rotary push-button knob, ROT1, that is used to set the frequency range and value inside the range for an external clock generated by a microcontroller. (13)

-50 MHz onboard clock oscillator as the clock input. The 50 MHz clock output line, GCLK0.



1 Power Connector	7 Seven-Segment Displays	14 Push Buttons
2 General-Purpose Breadboard Banks	8 USB Connector	15 Slide Switches
	9 LD-G LED	16 Digilent Pmod Connectors
3 Signal Breadboard Bank BB2	10 LEDs	17 Signal Breadboard Bank BB1
4 Power Switch	11 FPGA	18 Signal Breadboard Bank BB4
5 Signal Breadboard Bank BB3	12 Switch SW9	19 NI ELVIS Connector
6 Reset Button	13 Rotary Push-Button Knob/LEDs	20 Signal Breadboard Bank BB5

Figure 2: The NI Digital Electronics FPGA Board

3.2. UCF File Constraints

3.2.1. Slide Switches

The UCF file constraints for the eight slide switches, SW0 to SW7, are listed as follows. SWx refers to the slide switch line, LOC indicates the FPGA line location, and IOSTANDARD is the I/O standard used.

```
Net "SW0" LOC="J11" | IOSTANDARD = LVCMOS33;
Net "SW1" LOC="J12" | IOSTANDARD = LVCMOS33;
Net "SW2" LOC="H16" | IOSTANDARD = LVCMOS33;
Net "SW3" LOC="H13" | IOSTANDARD = LVCMOS33;
Net "SW4" LOC="G12" | IOSTANDARD = LVCMOS33;
Net "SW5" LOC="E14" | IOSTANDARD = LVCMOS33;
Net "SW6" LOC="D16" | IOSTANDARD = LVCMOS33;
```

3.2.2. Push Buttons

The UCF file constraints for the four push buttons, BTN0 to BTN3, are listed as follows. BTNx refers to the push button line, LOC indicates the FPGA line location, and IOSTANDARD is the I/O standard used.

```
Net "BTN0" LOC="C13" | IOSTANDARD = LVCMOS33;
Net "BTN1" LOC="D12" | IOSTANDARD = LVCMOS33;
Net "BTN2" LOC="C12" | IOSTANDARD = LVCMOS33;
Net "BTN3" LOC="C10" | IOSTANDARD = LVCMOS33;
```

3.2.3. LEDs

The UCF file constraints for the eight LEDs, LED0 to LED7, are listed as follows. LEDx refers to the LED line, LOC indicates the FPGA line location, IOSTANDARD is the I/O standard used, SLEW refers to the slew rate, the maximum rate of change of a signal, and DRIVE indicates the current drive strength on the FPGA in milliamps.

```
Net "LED0" LOC="C11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED1" LOC="D11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED2" LOC="B11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED3" LOC="A12" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED4" LOC="A13" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED5" LOC="B13" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED6" LOC="A14" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
Net "LED7" LOC="B14" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
```

4. Appending your Program of the first lab

In the first lab you wrote a program which was using one switch as an input and one LED as an output. You should now extend your program by three additional switches and LEDs.

- Uses four switches (SW0, SW1, SW2, SW3)
- Uses four LEDs (LED0, LED1, LED2, LED3)
- Swtich0 sets the output of LED0
- Swtich1 sets the output of LED1
- Swtich2 sets the output of LED2
- Swtich3 sets the output of LED3

5. Testing

You should upload your program to the FPGA and not the EEPROM (as done in lab one). For this and all following labs we will use the FPGA in JTAG mode.

5.1. Manual testing

Slide the switches manually and check that the LEDs behave correctly. There are 16 possible combinations which you need to check.

5.2. Automatic testing

Simple programs like this can be manually tested without big effort, but what about more complex programs? It would be very inefficient to test complex programs manually. Every amendment in a program would also require the engineer to change the testing scenario. How can we test input signals for which state changes within milliseconds are normal? This would be basically not feasible using manual testing with switches.

The solution for these problems will be the use of **Test Benches**, which are programs to simulate inputs and process the outputs of VHDL programs.

In the ISE Design Suite: switch from **Implementation** to **Simulation** and add a source file with the name “lab1test.vhd”. The result should look as presented in Figure 3. The created source file will be prefilled with some structures but not complete at this time. One very important point at this stage is the introduction of processes which allow the sequential execution of code.

Read and understand the generated file and complete the missing parts of it. Make sure that all inputs and outputs are set correctly.

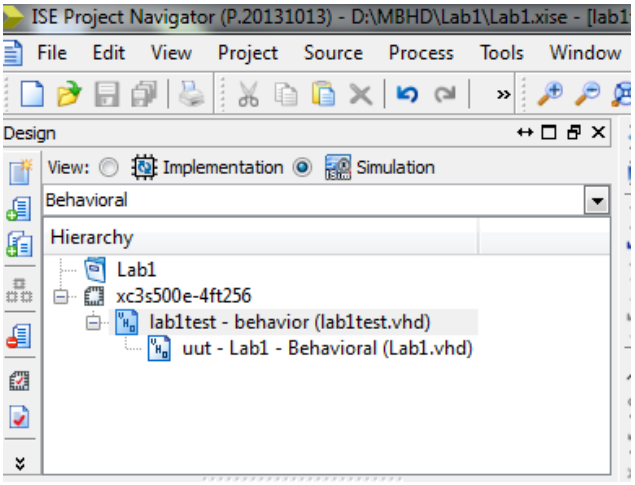


Figure 3: Simulation window

You need to consider the correct port mapping inside your test bench. Please check that your file looks similar to Figure 4.

```
uut: Lab1 PORT MAP (
      SW0 => SW0,
      LED0 => LED0
     );
```

Figure 4: Port mapping code fragment

This code fragment defines the variable names in the test bench file. Generally it would be a great idea to stick with the variable names used in the VHDL code to reduce confusions.

Every test bench uses a clock signal, which the engineer needs to create inside the test bench. This generated clock is independent from the hardware clock. You need to create an output signal for the clock in order to use the clock in your simulation.

signal Clock : std_logic;

For longer simulations is would be nice to have a simple way to check if all tests have been done successfully. One way of realizing this is to create an ok flag and use that within the simulation.

signal ok : BOOLEAN;

This flag can be set to true directly at the beginning followed by testing each possible input/output combination. Example: all switches are off, all LEDs must be off too (see Figure 5)

```
SW0 <= '0';
SW1 <= '0';
SW2 <= '0';
SW3 <= '0';
wait for 10 ns;
if LED0 /= '0' OR LED1 /= '0' OR LED2 /= '0' OR LED3 /= '0' then
        ok <= FALSE;
end if;
```

Figure 5: Example of ok signal

In the presented example (see Figure 5) is a short delay implemented after toggling the switches which is used by the simulation tool to simulate internal hardware delays.

You need to simulate the test bench by highlighting the test bench file and click on "Simulate Behavioral Model". You should see a window which shows an overview of the simulation as presented in Figure 6.

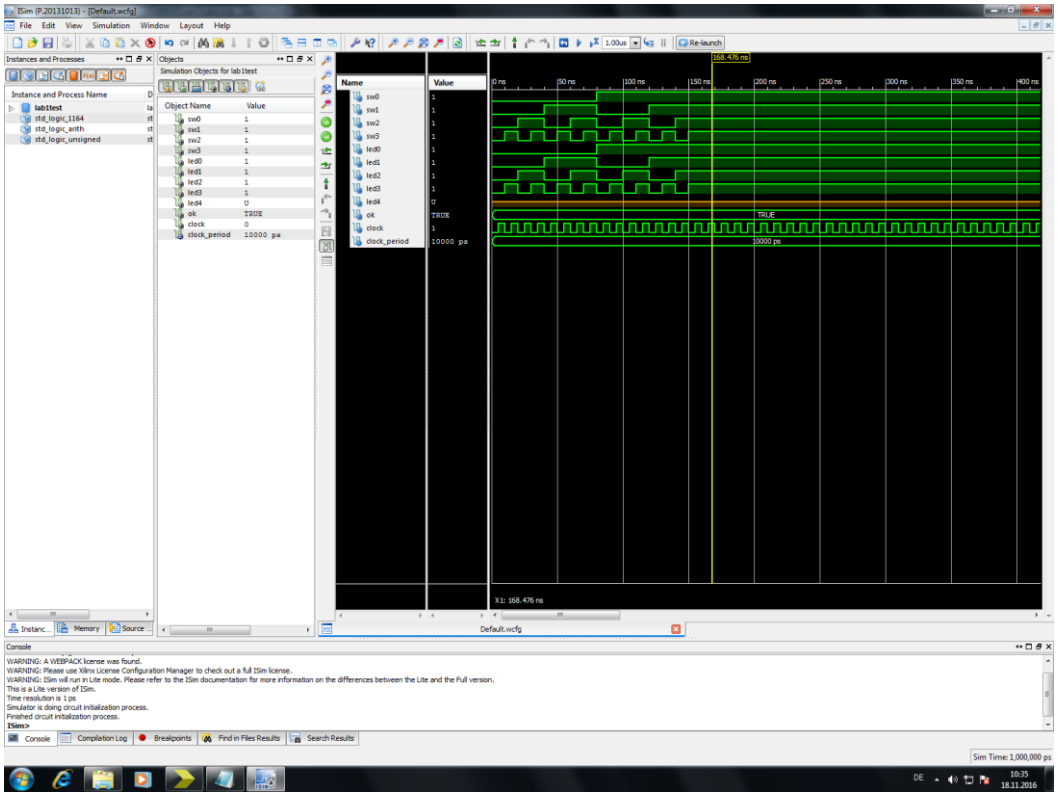


Figure 6: Simulation window

You can zoom in and out holding the control key (STRG) and use the mouse scroll wheel.

Edit your VHDL program by changing SW2 and SW3 as shown below.

```
LED0 <= SW0;
LED1 <= SW1;
LED2 <= SW3;
LED3 <= SW2;
```

Run the test bench again and compare the result to the previous result. What has changed and why?