Lab 1: Introduction to VHDL and Vivado

Bulat Khusainov and Eric Kerrigan

Learning outcomes

By the end of this lab, the student should be able to:

- Describe an inverter circuit with VHDL
- Use the Vivado synthesis tool to implement an inverter circuit on the Zedboard

Files provided

lab1.vhd	Circuit description
lab1_testbench.vhd	Testbench
lab1_constraints.xdc	Circuit constraints

Target platform

Familiarize yourself with the main interfaces of the Zedboard (Figure 1). In this lab we will use

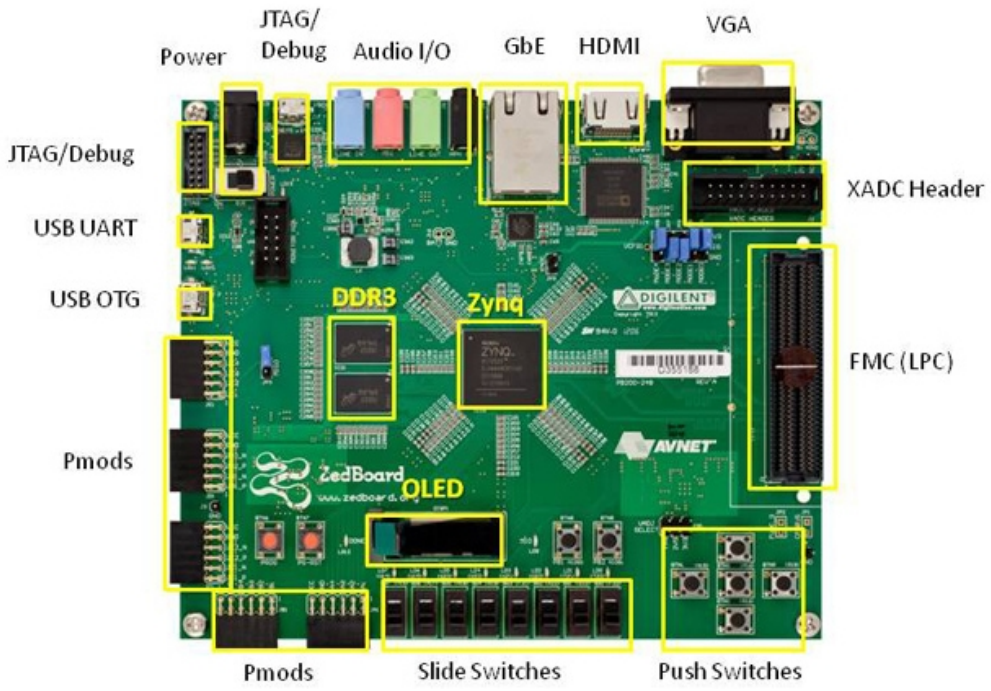

Figure 1: Zedboard

the following interfaces:

- JTAG/Debug for programming the platform.
- Slide switches to define circuit inputs.

- LEDs to observe circuit output signals. Note that LEDs are not highlighted on Figure 1 — the corresponding LED can be found next to each slide switch.

Circuit

The circuit of interest has two inputs and two outputs (Figure 2). Input 0 (Switch 0) is connected directly to the output, while the first input (Switch 1) is inverted. Switches and LEDs allow one to set inputs and observe outputs.

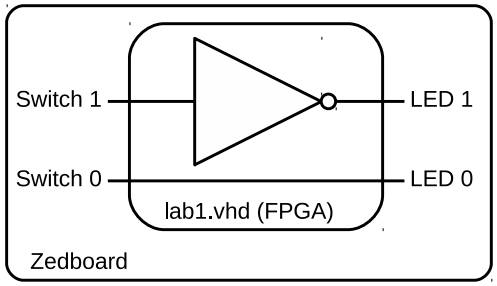

Figure 2: Circuit of interest

VHDL basics

Any piece of a regular VHDL code consists of the following subsections:

- Libraries and packages declarations
- ENTITY: circuit input/output ports
- ARCHITECTURE: defines the circuit behaviour

In this lab we will be using the *std_logic_1164* package from the *ieee* library. The package provides the STD_LOGIC data type that incorporates four values: '0', '1', '-' (don't care) and 'Z' (high impedance). Consider the VHDL code that implements the above circuit (this can also be found in the lab1.vhd file).

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY lab1 IS
6      PORT (
7              input:    IN  STD_LOGIC_VECTOR (1 DOWNTO 0);
8                        output:   OUT STD_LOGIC_VECTOR (1 DOWNTO 0));
9  END lab1;
10 ----------------------------------------
11 ARCHITECTURE behavioral OF lab1 IS
```

```
12
13 BEGIN
14     output(0) <= input(0);
15     output(1) <= NOT input(1);
16 END behavioral;
17 ---------------------------------------
```

Although VHDL is not case sensitive, we will be using upper case for the reserved words and lower case for the rest of the code. The block (entity) of interest has two inputs (line 7) and two outputs (line 8) of the STD_LOGIC_VECTOR type. The internal architecture is defined in the subsequent subsection (lines 14–15). '<=' defines a physical connection, similar to connecting with an electrical wire. Hence, the order of lines 14–15 does not affect the resulting circuit. Note that the above code is *concurrent*, since the output is totally defined by the current input. *Sequential* circuits will be considered in the next lab.

Synthesising VHDL code with Vivado

We will synthesize the circuit using the Vivado software tool from Xilinx. The first step is creating a project:

- Open Vivado → *File* → *New Project* → *Next*.
- Enter the project name ('lab1') and choose the directory. *Next*.
- *RTL project* (selected), *Do not specify sources* (checked). *Next*.
- *Boards* → *Zedboard* → *Next* → *Finish*.

At this stage you should see a window similar to the one in Figure 3. Let us import the file 'lab1.vhd' to the project:

- *Sources* pane → Right mouse click on *Design Sources* → *Add sources* → *Add or create design sources* → *Next* → *Add files* → Select 'lab1.vhd' → 'Copy sources into project' checked → *Finish*.
- Open the file from the *Sources* pane to make sure it was imported correctly.

In addition to VHDL code, we import a constraints file, which describes the mapping between a circuit's ports and FPGA pins. In this lab we map input ports to Zedboard switch buttons and output ports to the LEDs.

- *Sources* pane → Right mouse click on *Constraints* → *Add sources* → *Add or create constraints* → *Next* → *Add files* → Select 'lab1_constraints.xdc' → 'Copy constraints files into project' checked → *Finish*.
- Open the file from the *Sources* pane to make sure it was imported correctly.

Now that we have source files ready we will go through the steps of the Flow Navigator pane:

- *RTL ANALYSIS* → *Open Elaborated Design*. At this stage Vivado will show a schematic representation of the circuit. This step should take around 1 min.

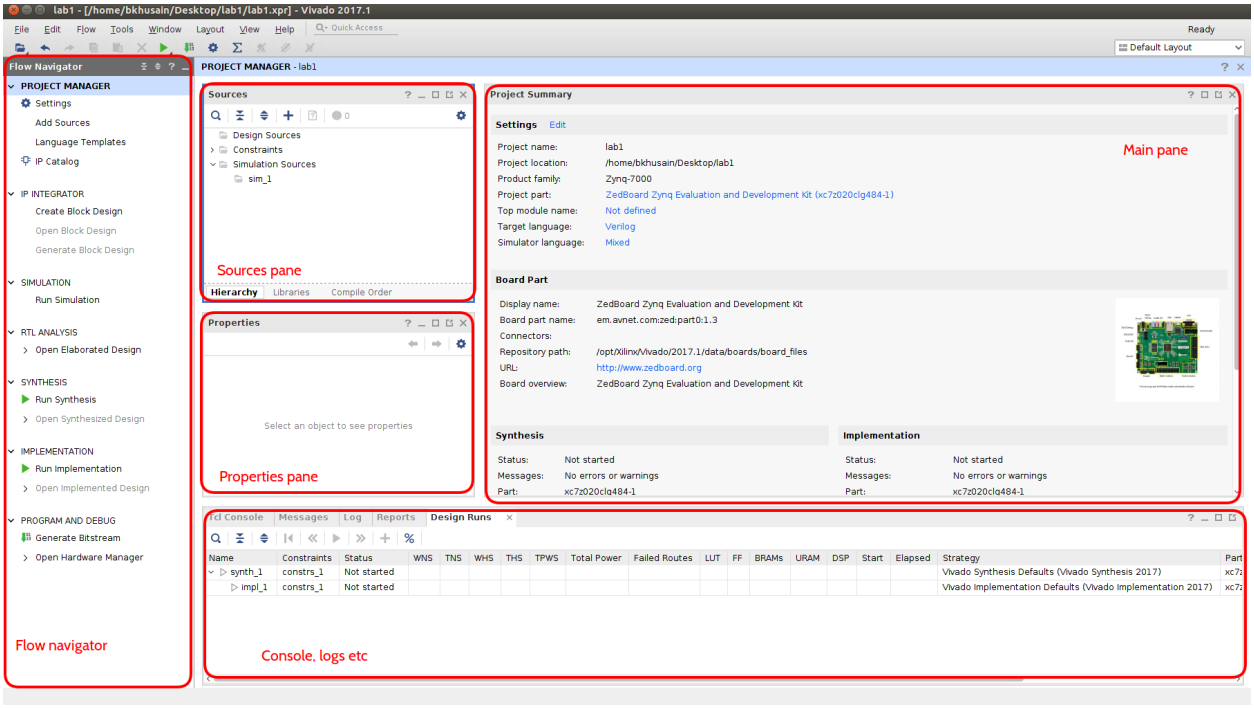

Figure 3: Vivado project window.

- *SYNTHESIS → Run synthesis.* Synthesis is converting VHDL code into a *netlist*. A netlist is a graph that shows connections between logic blocks. The step may take up to 2–3 minutes.
- *IMPLEMENTATION → Run Implementation.* In this context implementation means placing and routing, i.e. mapping the netlist onto a particular FPGA. 2-3 minutes.
- *PROGRAM AND DEBUG → Generate Bitstream.* Converts the final circuit into a sequence of bits to be uploaded to FPGA. 2-3 minutes.

Once the bitstream is ready it can be uploaded to the FPGA. Make sure that your Zedboard has a power supply and USB-JTAG (labelled with PROG) is connected to the computer. To upload the bitstream: *PROGRAM AND DEBUG → Open hardware manager → Open target → Auto connect.* Select *Program Device → Program.* Try changing the states of the slide switches on the Zedboard and observe the corresponding LEDs. The circuit should behave in accordance with Figure 2.

Using testbenches*

In this work we have synthesized an extremely simple circuit, which can be implemented on FPGA directly. However, more complicated circuits are often tested with software simulations before time-consuming synthesis to detect bugs in the early stages of the design flow. Testing a circuit implies connecting (in simulation) its interface ports to another surrounding circuit that provides inputs and records outputs. The surrounding circuit is known as a *testbench* and can be implemented with VHDL. In this lab a testbench file is provided and can be added to the project by following these steps:

- *Sources* pane → Right mouse click on *Simulation Sources* → *Add sources* → *Add or create simulation sources* → *Next* → *Add files* → Select 'lab1_testbench.vhd' → 'Copy sources into project' checked → *Finish*.
- Open the file from the *Sources* pane to make sure it was imported correctly.

Testbench VHDL code has a similar structure to regular VHDL code, excluding the fact that interface ports are missing. In the ARCHITECTURE subsection (refer to **lab1_testbench.vhd**) the circuit under test is declared and port mappings are defined. Note that a 'WAIT FOR' expression is allowed for simulation, but not for synthesis, where all assignments happen in parallel.
To run a simulation: *SIMULATION* → *Run Simulation* → *Run Behavioral Simulation*. The results will appear in form of signals profiles (Figure 4). Check the circuit input and output signals. This might require zooming in/out.

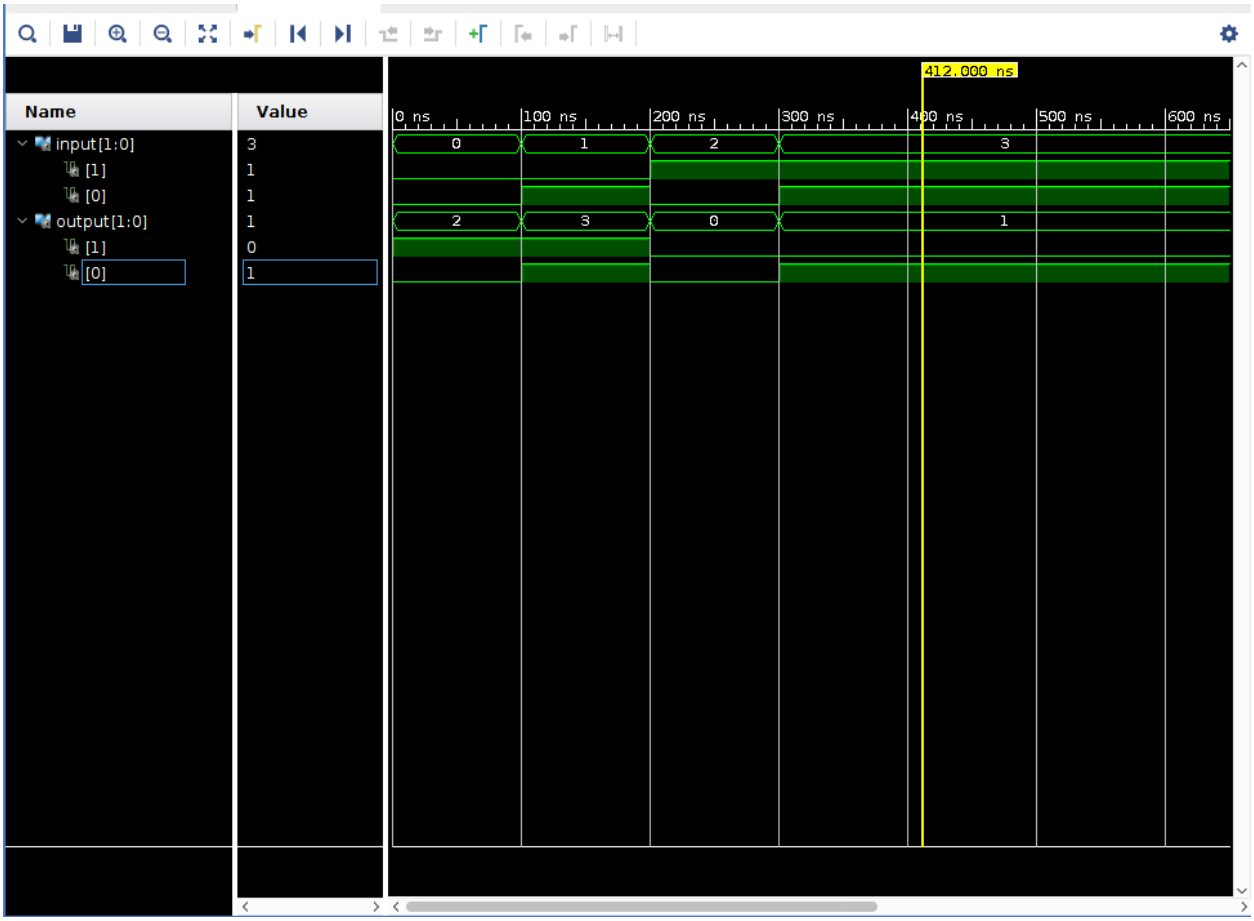

Figure 4: Simulation results.

Further reading and references

- V. Pedroni. Circuit Design with VHDL. MIT Press. 2010. Chapters 2-3.
- L. Crockett et.al. The Zynq Book. 2015. http://www.zynqbook.com