Report: Introduction to VHDL

Bilkent University Electrical and Electronics Department

EE102-01 Lab 2

Melih Ege Doğan – 22403329
February 17, 2025

Purpose:

Purpose of this lab was to study VHDL and learn how Vivado works with Basys3 FPGA board. This lab also helped us to understand that combinational logic circuits are memoryless and operate independently of time, with their outputs determined solely by the specific combination of inputs.

Methodology:

This lab word practiced on how to debug a pre-written VHDL program, importing it in our Basys3 FPGA board and check the results using a testbench simulation. First, we changed the gates according to our student ID. Then we debugged every bug we can found. After that we created a bitstream file and imported it in the Basys3 FPGA board. Finally, we created a testbench simulation for all possible inputs and checked our results using the testbench.

Questions:

How does one specify the inputs and outputs of a module in VHDL?

You specify inputs and outputs for entity in the port section. After you declared the entity, you can specify the inputs and outputs. In the port section you should write the information about the input or output such as name, mode (in or out) and, data type.

How does one use a module inside another code/module? What does PORT MAP do?

You can use a module inside another module by declaring it inside the top module with component function. After you declared you should insert a port section

to the declaration. Finally, you need to connect the signals of two modules. To do that you have to use port map. Port map connecs the inputs and outputs from a module to local signals ion the design where it's instantiated.

What is a constraint file? How does it relate your code to the pins on your FPGA?

Constraint file is a text file which has significant information about your design. Constraint file basically connect your VHDL code with the board. In the constraint file you must give where are inputs coming from on the board and where are outputs going on the board. Also, in constraint file you must define the electrical characteristics of the input and output pins.

What is the purpose of writing a testbench?

Testbench helps user to create a simulation made from waveforms and diagrams. By creating the waveform user can check if there is a input caused by input or output mistakes.

Changing Logic Gates:

This lab wants to modify the submodule1.vhdl file. We must change three gates in line 14,15,16 according to our student number. For me it was XNOR, XNOR, AND gates (Figure 1.1).

```
10
11  architecture Structural_sub1 of sub_module1 is
12
13  begin
14      o_output_byte(0)          <= i_input_byte(0) xnor i_input_byte(1); --
15      o_output_byte(1)          <= i_input_byte(2) xnor i_input_byte(3);
16      o_output_byte(2)          <= i_input_byte(4) and i_input_byte(5);
17      o_output_byte(5 downto 3) <= "010";
18      o_output_byte(7 downto 6) <= (others => '0');
19
20  end Structural_sub1;
21
```

Figure 1.1: Gates.

Debugging:

In the second step we found six bugs in the code using error messages.

1) There was typo error in the top_module.vhdl (Figure 2.1). Since there is nothing called s_ouput_1 this is a typo error and it should be s_output_1 (Figure 2.3).

```
31  begin
32
33      sub_module1_2 : sub_module1
34          port map (
35              i_input_byte  => i_SW,
36              o_output_byte => s_ouput_1
37          );
38
39      sub_mo
40          port map (s_output_2, i_SW);
41      s_output_3 <= unsigned(s_output_2) + 25;
42
43      o_LED <= (not std_logic_vector(s_output_3)) xor s_output_1;
44
```

Error: 's_ouput_1' is not declared
Error: actual of formal out port 'o_output_byte' cannot be an expression

Figure 2.1: s_ouput_1 error.

2) There was a syntax error in how port map was written in the top_module.vhdl (Figure 2.2). However variables are correct, they are inversed in the port map. To fix this error we simply changed the positions of the two variables (Figure 2.3).

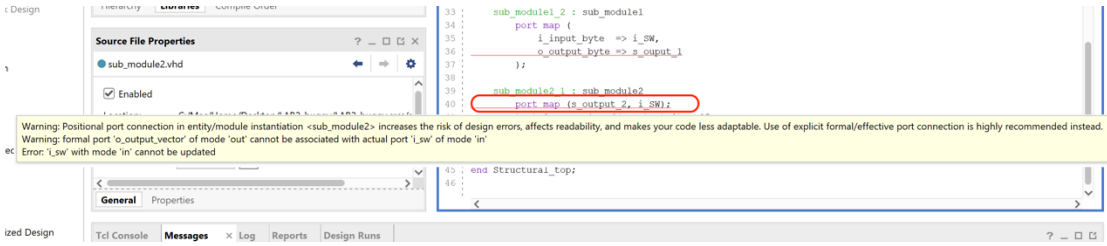

Figure 2.2: Port map error.

```
32
33      sub_module1_2 : sub_module1
34          port map (
35              i_input_byte  => i_SW,
36              o_output_byte => s_output_1
37          );
38
39      sub_module2_1 : sub_module2
40          port map (i_SW,s_output_2);
41      s_output_3 <= unsigned(s_output_2) + 25;
42
43      o_LED <= (not std_logic_vector(s_output_3)) xor s_output_1;
44
45  end Structural_top;
46
```

Figure 2.3: top_module.vhdl debugged.

3) We also found error in the sub_module2.vhdl. However, the name of the entity was sub_module2 in the code entity was declared as sub_module_2_the_beast (Figure 2.4). Solution of this error is very easy. We just get rid of the _the_beast part of the entities (Figure 2.5).

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity sub_module2_the_beast is
5       Port (
6           i_switch_inputs : in  STD_LOGIC_VECTOR (7 downto 0);
7           o_output_vector : out STD_LOGIC_VECTOR (7 downto 0)
8       );
9   end sub_module2_the_beast;
10
11  architecture Structural_sub2 of sub_module2_the_beast is
12      component sub_module1 is
13          port (
14              i_input_byte  : in  STD_LOGIC_VECTOR (7 downto 0);
15              o_output_byte : out STD_LOGIC_VECTOR (7 downto 0)
16          );
17      end component sub_module1;
```

Figure 2.4: sub_module_2_the_beast error.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity sub_module2 is
5       Port (
6           i_switch_inputs : in  STD_LOGIC_VECTOR (7 downto 0);
7           o_output_vector : out STD_LOGIC_VECTOR (7 downto 0)
8       );
9   end sub_module2;
10
11  architecture Structural_sub2 of sub_module2 is
12      component sub_module1 is
13          port (
14              i_input_byte  : in  STD_LOGIC_VECTOR (7 downto 0);
15              o_output_byte : out STD_LOGIC_VECTOR (7 downto 0)
16          );
17      end component sub_module1;
18
19      signal s_inv_input : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
20  begin
21      s_inv_input <= not i_switch_inputs;
22
23      sub_module1_1 : sub_module1
24          port map (
```

Figure 2.5: Fixed version of sub_module2.vhdl

4) There was 2 more error in the constraint_basys3.xdc file. After looking through every single line we found that there was a curly bracket missing in the line 28 (Figure 2.6). Solution for this is just adding the missing curly bracket (Figure 2.7).
5) Second error in the constraint_basys3.xdc was outputs led pins are in a wrong order. However in the basys3 board U16 is the first led pin and V14 is the eight led pin, in the file they were in the wrong place (Figure 2.6). Solution for this was the changing U16 and V14 in the code (Figure 2.7).

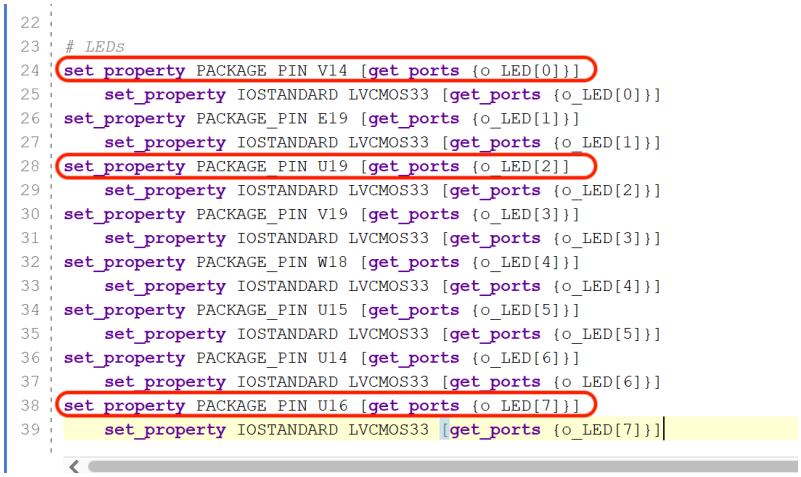

Figure 2.6: Constraint file with missing curly bracket and wrong led order.

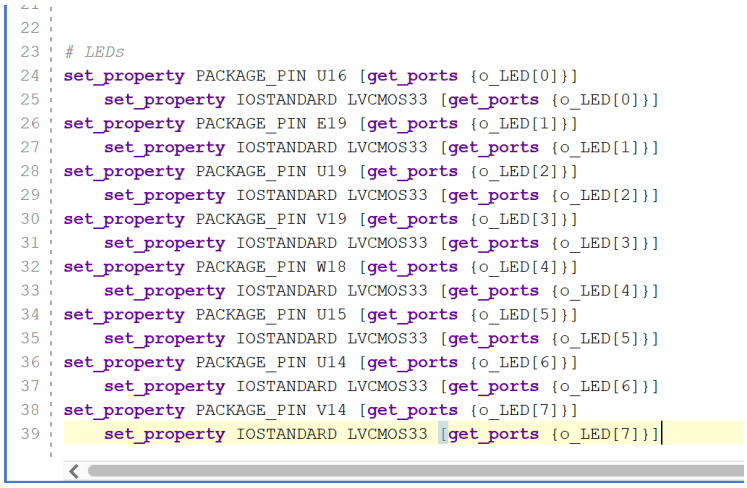

Figure 2.7: Debugged constraint file.

6) The last error was in the hierarchy section. top_module.vhdl file wasn't set as top in the hierarchy (Figure 2.8). This caused error because top_module was created with using the sub modules. To solve this error, just right click on the top_module.vhdl in the hierarchy and click set as top (Figure 2.9).

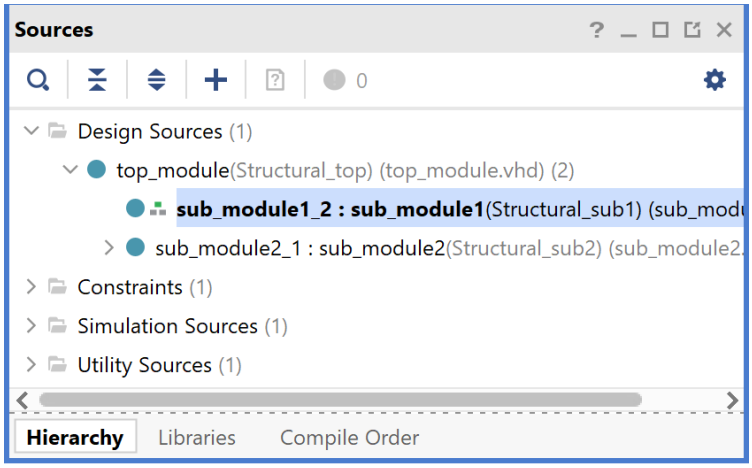

Figure 2.8: top_module not set as top.

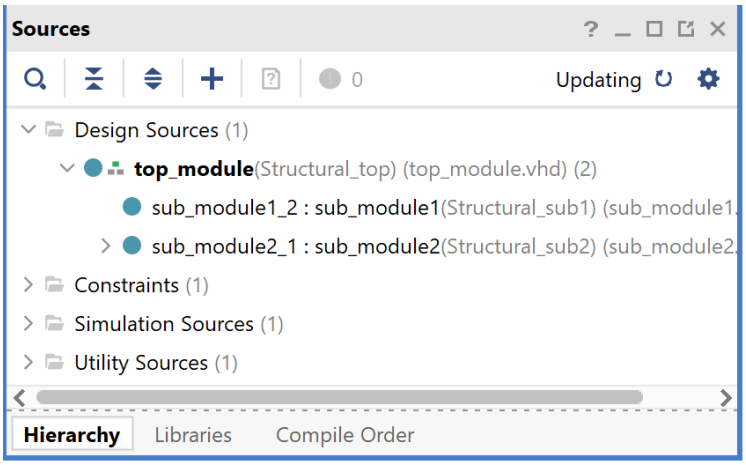

Figure 2.9: top_module set as top.

Programming Basys3:

After we debugged every error, now we can program our basys3 board. To program the card, the bitstream file should be created. After we created the bitstream file we changed device to our device in the settings (Figure 3.1). Finally, we connected our device to vivado and we programmed our basys3 by clicking program device.

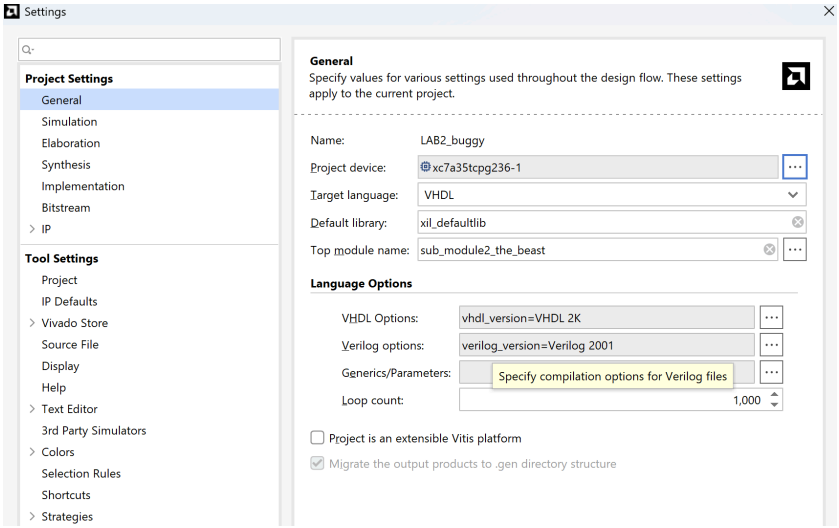

Figure 3.1: Selecting correct device.

When the device is programmed, we can observe that led are changing with the inputs we gave from switches. Here are some examples of inputs and outputs (Figures 3.2,3.3,3.4,3.5,3.6). We can observe that outputs from the leds are only changing from inputs. This shows that combinational logic circuits are memoryless and operate independently of time.

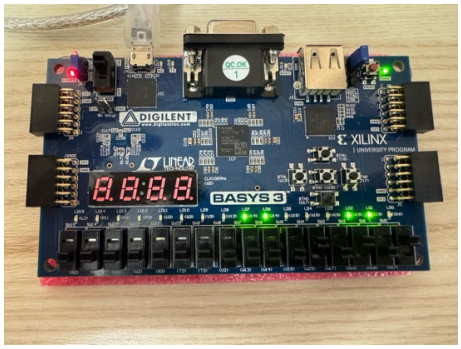

Figure 3.2: ‘10101000’.

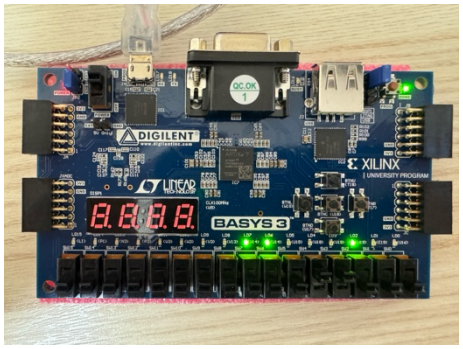

Figure 3.3: ‘00011000’.

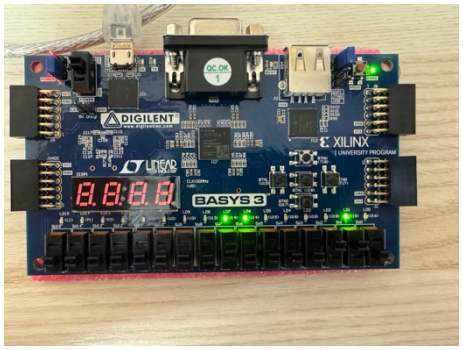

Figure 3.4: ‘01100001’.

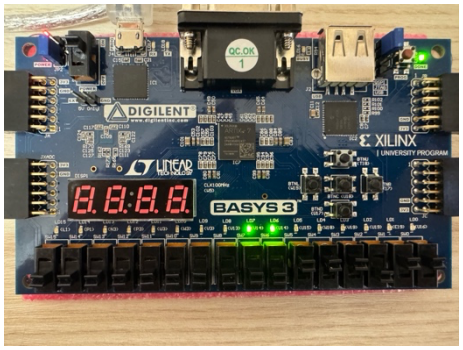

Figure 3.5: ‘11011100’.

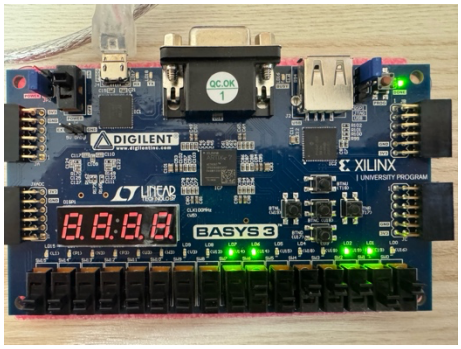

Figure 3.6: ‘10111010’.

Creating the testbench:

What test bench do basically is it tries every possible input and creates outputs for them. To do this sequentially we first turn off the first input for 10ns and turn on for 10ns, then we turn of the second input for 20ns and turn on for 20ns, then we do the same thing to the third one with 40ns and this goes on for every input. By this method we can create the outputs and sequential waveforms (Figure 4.1).

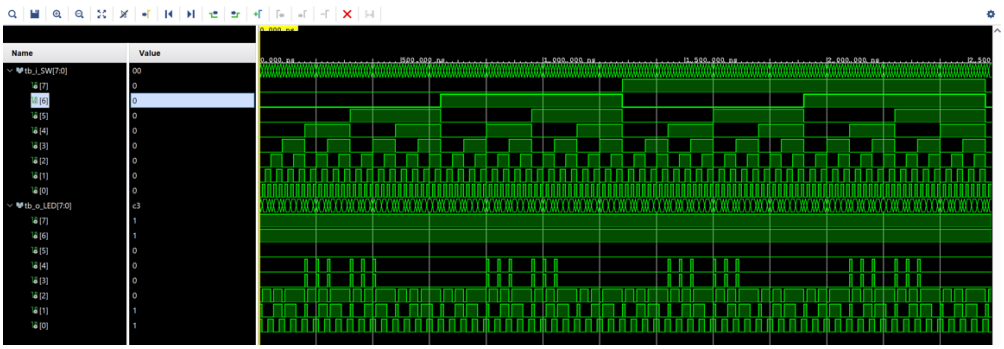

Figure 4.1: Sequential Testbench Waveforms.

Since there are 8 input pins and $2^8 = 256$ possible inputs, if we try to this by one by one it will take a year. To solve this issue, we created a for loop to do all of this (Figure 4.2).

```
37      begin
38          for i in 0 to 511 loop
39              tb_i_SW <= std_logic_vector(to_unsigned(i, 8));
40              wait for 10 ns;
41
42          end loop;
43
44          wait;
45      end process;
```

Figure 4.2: For Loop for Testbench.

After we created the testbench we compared our results from test bench with the board and checked if there was any error. We found out there is no error in the code. Here are some examples for checking (Figures 4.3,4.4,4.5,4.6).

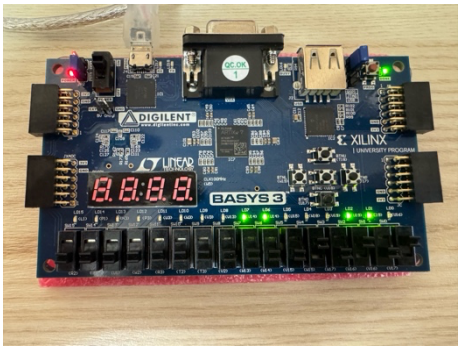

Figure 4.3: Input ‘10101000’.

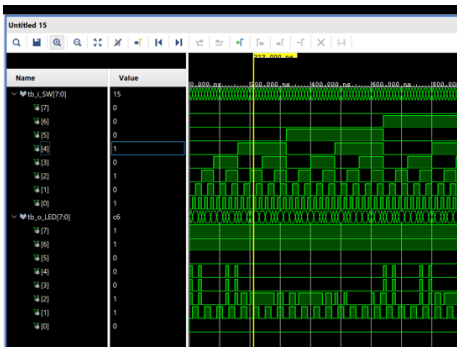

Figure 4.4: Testbench Input ‘10101000’.

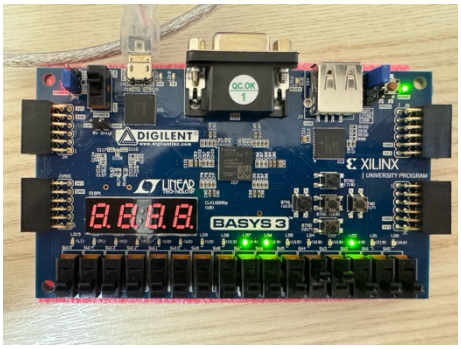

Figure 4.3: Input ‘00011000’.

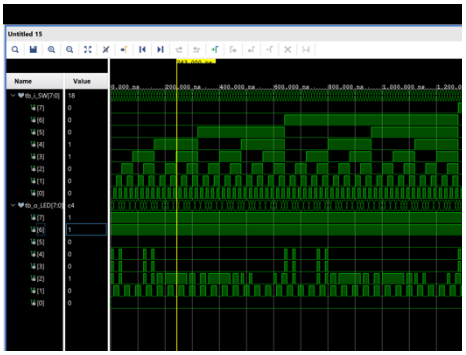

Figure 4.4: Testbench Input ‘00011000’.

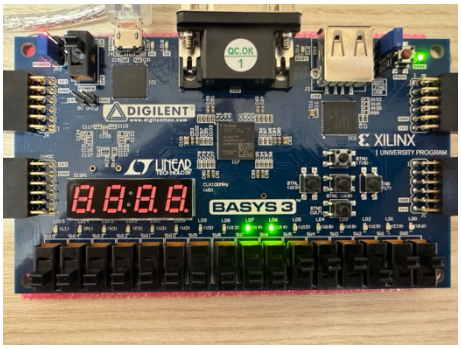

Figure 4.3: Input ‘00011000’.

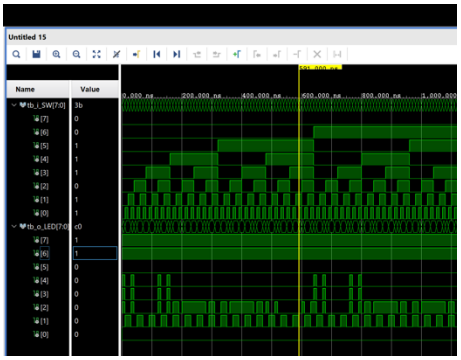

Figure 4.4: Testbench Input ‘00011000’.

Schematics:

Finally, we created the schematics for our circuit. First, we created the RTL schematic (Figure 5.1). As seen in the RTL schematic, top module uses sub module1 and sub module2. Also, we saw that top module has 8 inputs and 8 outputs as it should be. After we checked the schematic by trying random inputs, we were sure that circuit schematic is correct.

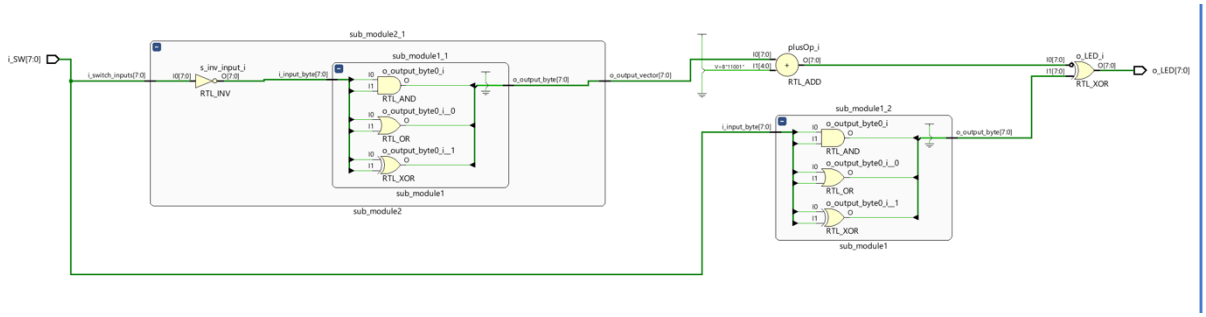

Figure 5.1: RTL schematic.

After generating the RTL schematic (Figure 5.1), we proceeded through synthesis and implementation. The resulting synthesized schematic (Figure 5.2) and implemented schematic (Figure 5.3) were visually very similar, or even identical, to each other. However, both the synthesized and implemented schematics were significantly different from the RTL schematic.

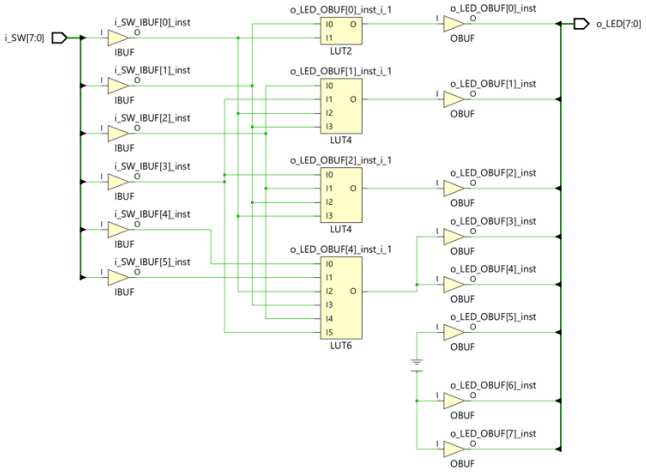

Figure 5.2: Synthesized schematic.

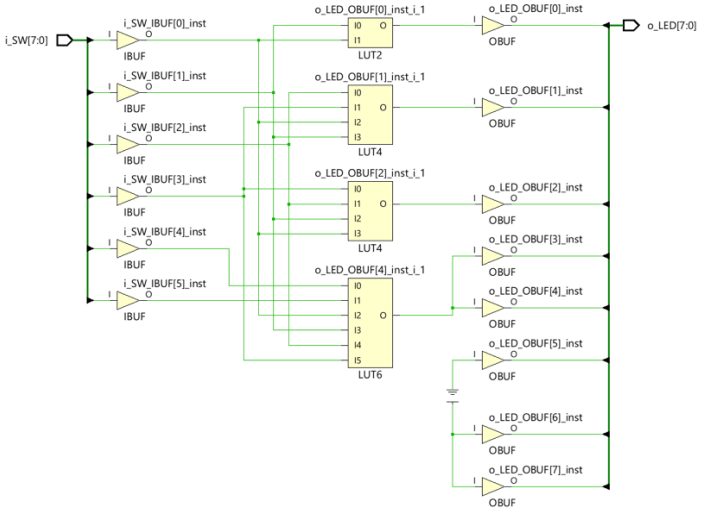

Figure 5.3: Implemented schematic.

Conclusion:

This lab provided an experience with VHDL, Vivado and the Basys3 board. We successfully debugged a VHDL design, correcting errors in code, constraints, and project settings while understanding error messages. We observed memoryless structure of the basys3 board. Creating a testbench helped us to understand the circuit with simulation. We observed different schematic types and looked the difference with each other.

Appendices:

Code 1: (sim1.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity sim1 is
end sim1;

architecture Behavioral of sim1 is

    COMPONENT top_module
        Port (
            o_LED : out STD_LOGIC_VECTOR (7 downto 0);
            i_SW  : in  STD_LOGIC_VECTOR (7 downto 0)

        );
    END COMPONENT;

    signal tb_o_LED : STD_LOGIC_VECTOR (7 downto 0);
    signal tb_i_SW  : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');




begin


    UUT: top_module
        port map (
            i_SW  => tb_i_SW,
            o_LED => tb_o_LED
        );


    stim_proc : process
    begin
        for u in 0 to 511 loop
            tb_i_SW <= std_logic_vector(to_unsigned(u, 8));
            wait for 10 ns;

        end loop;

        wait;
    end process;

end Behavioral;
```

Code 2: (top_module.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity top_module is
    Port (
        i_SW  : in  STD_LOGIC_VECTOR (7 downto 0);
        o_LED : out STD_LOGIC_VECTOR (7 downto 0)
    );
end top_module;

architecture Structural_top of top_module is

    component sub_module1 is
        port (
            i_input_byte  : in  STD_LOGIC_VECTOR (7 downto 0);
            o_output_byte : out STD_LOGIC_VECTOR (7 downto 0)
        );
    end component sub_module1;

    component sub_module2 is
        port (
            i_switch_inputs : in  STD_LOGIC_VECTOR (7 downto 0);
            o_output_vector : out STD_LOGIC_VECTOR (7 downto 0)
        );
    end component sub_module2;

    signal s_output_1, s_output_2 : STD_LOGIC_VECTOR(7 downto 0) := (others =>
'0');
    signal s_output_3             : unsigned(7 downto 0)         := (others =>
'0');

begin

    sub_module1_2 : sub_module1
        port map (
            i_input_byte  => i_SW,
            o_output_byte => s_output_1
        );

    sub_module2_1 : sub_module2
        port map (i_SW,s_output_2);
    s_output_3 <= unsigned(s_output_2) + 25;

    o_LED <= (not std_logic_vector(s_output_3)) xor s_output_1;

end Structural_top;
```

Code 3: (constraints_basys3.xdc)

```
# Constraint file sets which input/output in the VHDL code connects to which pin in
FPGA.
# Three characters after PACKAGE_PIN gives the FPGA pin which is also written next
to all LEDs/Switches on the board.
# The second line (IOSTANDARD LVCMOS33) sets the voltage standard for the pins
which is constant and always the same for BASYS3.

set_property PACKAGE_PIN V17 [get_ports {i_SW[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[0]}]
set_property PACKAGE_PIN V16 [get_ports {i_SW[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[1]}]
set_property PACKAGE_PIN W16 [get_ports {i_SW[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[2]}]
set_property PACKAGE_PIN W17 [get_ports {i_SW[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[3]}]
set_property PACKAGE_PIN W15 [get_ports {i_SW[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[4]}]
set_property PACKAGE_PIN V15 [get_ports {i_SW[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[5]}]
set_property PACKAGE_PIN W14 [get_ports {i_SW[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[6]}]
set_property PACKAGE_PIN W13 [get_ports {i_SW[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[7]}]


# LEDs
set_property PACKAGE_PIN U16 [get_ports {o_LED[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[0]}]
set_property PACKAGE_PIN E19 [get_ports {o_LED[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[1]}]
set_property PACKAGE_PIN U19 [get_ports {o_LED[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[2]}]
set_property PACKAGE_PIN V19 [get_ports {o_LED[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[3]}]
set_property PACKAGE_PIN W18 [get_ports {o_LED[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[4]}]
set_property PACKAGE_PIN U15 [get_ports {o_LED[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[5]}]
set_property PACKAGE_PIN U14 [get_ports {o_LED[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[6]}]
set_property PACKAGE_PIN V14 [get_ports {o_LED[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[7]}]
```

Code 4: (sub_module1.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sub_module1 is
    Port (
        i_input_byte  : in  STD_LOGIC_VECTOR (7 downto 0);
        o_output_byte : out STD_LOGIC_VECTOR (7 downto 0)
    );
end sub_module1;

architecture Structural_sub1 of sub_module1 is

begin
    o_output_byte(0)          <= i_input_byte(0) xnor i_input_byte(1); --Change
these three operators according to the table in the lab document
    o_output_byte(1)          <= i_input_byte(2) xnor i_input_byte(3);
    o_output_byte(2)          <= i_input_byte(4) and i_input_byte(5);
    o_output_byte(5 downto 3) <= "010";
    o_output_byte(7 downto 6) <= (others => '0');

end Structural_sub1;
```

Code 5: (sub_module2.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sub_module2 is
    Port (
        i_switch_inputs : in  STD_LOGIC_VECTOR (7 downto 0);
        o_output_vector : out STD_LOGIC_VECTOR (7 downto 0)
    );
end sub_module2;

architecture Structural_sub2 of sub_module2 is
    component sub_module1 is
        port (
            i_input_byte  : in  STD_LOGIC_VECTOR (7 downto 0);
            o_output_byte : out STD_LOGIC_VECTOR (7 downto 0)
        );
    end component sub_module1;

    signal s_inv_input : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
begin
    s_inv_input <= not i_switch_inputs;

    sub_module1_1 : sub_module1
        port map (
```

```
            i_input_byte  => s_inv_input,
            o_output_byte => o_output_vector
        );
end Structural_sub2;
```

References:

- https://www.youtube.com/watch?v=Ts1be5T5u4&ab_channel=VHDLwhiz.com
- https://github.com/SemihAkkoc/EEE102