February 19th 2025

Alp Efe Kılıçarslan, 22402390

EEE 102-2

LAB-02: Introduction to VHDL

Purpose:

The goal of this assignment was to debug and test a buggy project file for a BASYS3 FGPA using VHDL and Vivado. A combinational logic circuit, which is a time-independent circuit who is memoryless and whose output is determined only by it's inputs, was given as the project. With these parameters, the experimenters could get a basic introduction to VHDL and FGPA's, allowing them to understand design and implementation for future projects.

Design Specifications:

The given files included one top module, two submodules and one constraint file. The constraint files assigned each switch to a specific LED on the BASYS3 in accordance to the FGPA's manual. The top module (top_module) was the main project while that controlled the entire system. Using the constraint file and the top module, the code was made functionable. The first submodule (sub_module1) handled what gate handled what input. The gates in this file were changed using the instructions given in the lab assignment. The second submodule (sub_module2_the_beast) handled port and logic assignment for each switch and LED.

Certain variables in the code determined what each module worked with. In the first submodule, eight values were assigned to i_input_byte and o_output_byte each and were defined as logic vectors so that the logic gates were able to do the required function. The specified architecture used three 2-variable logic gates for the operation of the module.

The second module declared eight logic values for i_switch_inputs and o_output_vector in its port. These were used later on to get the input data from the BASYS3 and output the desired output LED on the same board. The declared variables were processed and later on used in the top module for the final output.

The top module included eight logic vectors for i_SW and o_LED in its port. i_SW was used as the input switch parameter and o_LED was used as the output LED's state determiner. Using the submodules' included data, these parameters were turned into usable signals and finalized the program.

Methodology:

Four questions that were asked in the laboratory assignment were researched before any interaction with the included files. These questions and their informed answers are as follows:

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

- The inputs and outputs of a module are specified within the entity declaration using the port section. The port section lists all the ports along with their names, directions (in, out, or inout), and data types such as std_logic for the given task. For example, in an entity declaration, inputs can be defined like i_SW: in std_logic; and o_LED: out std_logic;. This defines the interface for the module, and the actual functionality of the module is described in the architecture section, where the interaction between the inputs and outputs is implemented.

• How does one use a module inside another code/module? What does PORT MAP do?
- To use one module inside another, it must be instantiated within the architecture of the parent module. The PORT MAP statement is used to connect the ports of the instantiated module to the corresponding signals or variables in the parent module. This ensures that the internal signals of the instantiated module are properly linked to the external signals of the parent module, facilitating the flow of data between them. The PORT MAP statement is important for defining how inputs and outputs are routed and ensures correct interaction between the modules.

• What is a constraint file? How does it relate your code to the pins on your FPGA?
- A constraint file is used to map the logical signals in a design to specific physical pins on an FPGA. It provides the necessary information for the synthesis and implementation tools to connect the design's ports (such as inputs, outputs, and clock signals) to the corresponding FPGA pins. This file typically includes pin assignments, I/O standards, timing constraints, and other configuration details that specify how the FPGA should interact with external hardware. By using a constraint file, the logical design described in the code is tied to the physical FPGA device, ensuring proper communication with external components.

• What is the purpose of writing a testbench?

- The purpose of writing a testbench is to simulate and verify the functionality of a design before implementing it on hardware. A testbench provides a controlled environment where input signals can be applied to the design, and its outputs can be monitored and compared to expected results. Writing a testbench helps in debugging, validating the logic, and improving the overall reliability of the design.

After these questions were answered, the given tasks were done as follows:

Task-1) The first submodule file was edited according to the experimenters last ID digit. The changes made to the submodule was saved.

Task-2) The synthesis of the program was run to get the errors. After this, every file was analyzed and debugged according to the related errors and warnings. When the bugs were gotten rid of, the synthesis was run again to check if the program worked without any errors.

Task-3) The BASYS3 FGPA was connected to the computer running the code with a USB cable. A bitstream was generated for the FGPA to run on. The computer connected to the FGPA through Vivado's auto connect option. After the connection was established, the "Program Device" function was ran. After the board fully realized the written program, different states were created by using the switches on the FGPA and photographed.

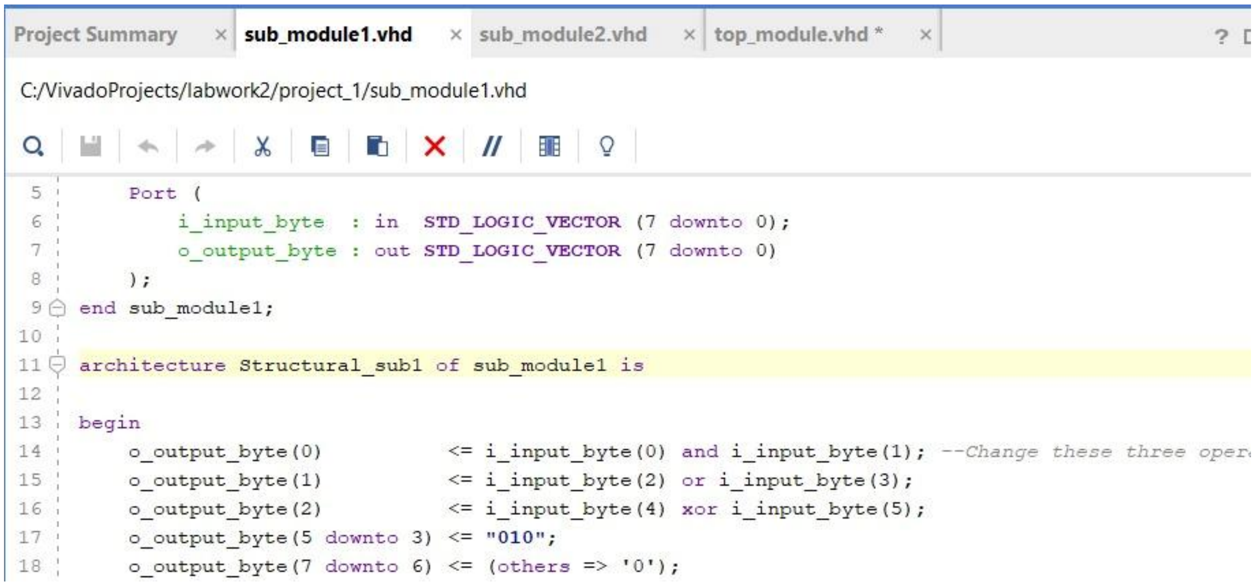
Task-4) A testbench which cycled different signals sequentially was coded with the help of the laboratory's teaching assistants. The signal that was generated by the testbench was run through the top module in a simulation whose runtime was edited to accommodate the full generated signal. The testbench utilized a loop, which helped the formatting of the simulation.

Task-5) Using Vivado's built in function, the RTL schematic was formed by clicking the schematic button under RTL Analysis. Likewise, the same process was repeated under Synthesis and Implementation in order to get the synthesized and implemented design respectively. The relation between all three schematics were noted under Results on the report.

Task-6) All pieces of edited code were noted and written on the Results section of the report.

Results:

Task-1) The gates that were required to be changed were determined by the experimenter's ID's last digit. Since the digit in question was zero, an AND gate, an OR gate and an XOR gate was added to the submodule file respectively (Figure 1.1) and the changes were saved.



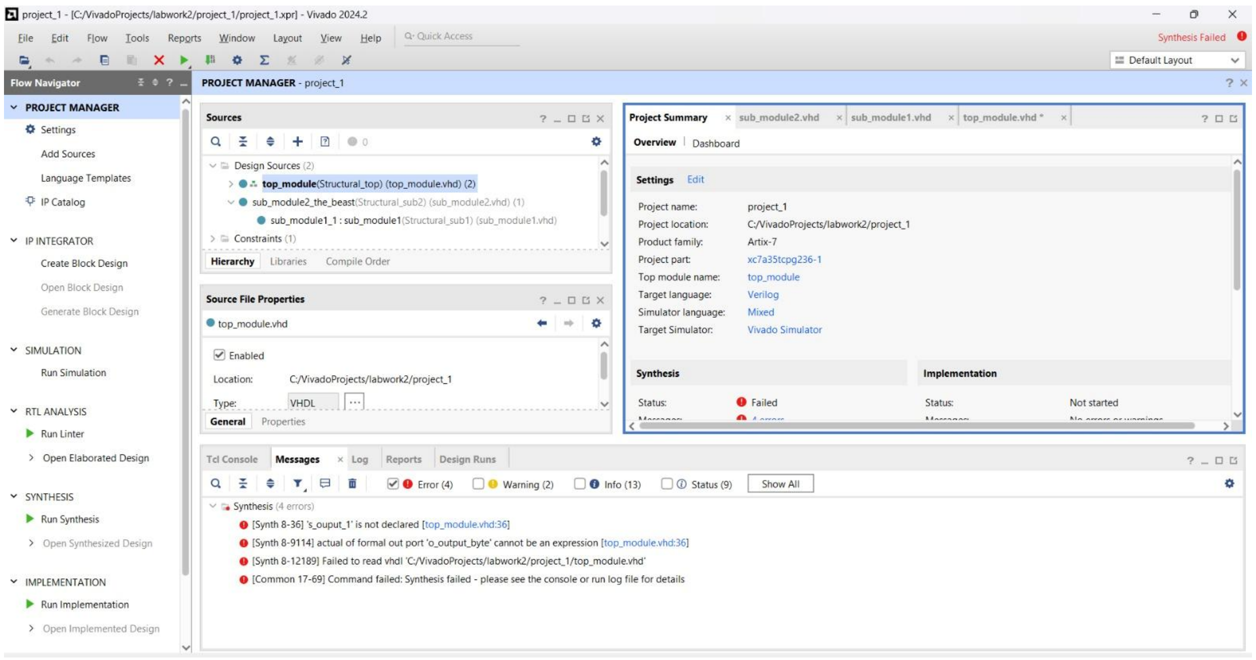
```
    Port (
        i_input_byte  : in  STD_LOGIC_VECTOR (7 downto 0);
        o_output_byte : out STD_LOGIC_VECTOR (7 downto 0)
    );
end sub_module1;

architecture Structural_sub1 of sub_module1 is

begin
    o_output_byte(0)          <= i_input_byte(0) and i_input_byte(1); --Change these three oper
    o_output_byte(1)          <= i_input_byte(2) or i_input_byte(3);
    o_output_byte(2)          <= i_input_byte(4) xor i_input_byte(5);
    o_output_byte(5 downto 3) <= "010";
    o_output_byte(7 downto 6) <= (others => '0');
```

Figure 1.1: Edited gates in sub_module1.vhd

Task-2) Four errors were gotten after the synthesis was run and observed under the Messages tab in Vivado (Figure 2.1). Using the information from this data, the following changes were made:



project_1 - [C:/VivadoProjects/labwork2/project_1/project_1.xpr] - Vivado 2024.2

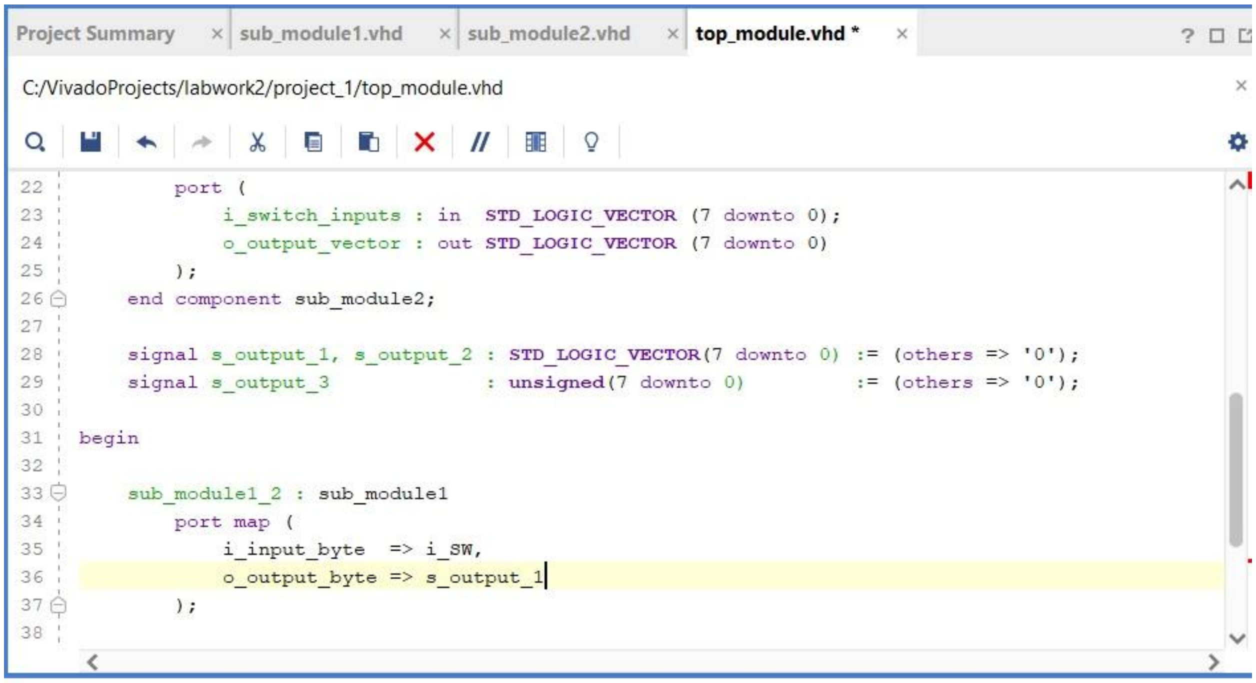
Project name: project_1
Project location: C:/VivadoProjects/labwork2/project_1
Product family: Artix-7
Project part: xc7a35tcpg236-1
Top module name: top_module
Target language: Verilog
Simulator language: Mixed
Target Simulator: Vivado Simulator

Synthesis Status: Failed
Implementation Status: Not started

```
Synthesis (4 errors)
[Synth 8-36] 's_ouput_1' is not declared [top_module.vhd:36]
[Synth 8-9114] actual of formal out port 'o_output_byte' cannot be an expression [top_module.vhd:36]
[Synth 8-12189] Failed to read vhdl 'C:/VivadoProjects/labwork2/project_1/top_module.vhd'
[Common 17-69] Command failed: Synthesis failed - please see the console or run log file for details
```

Figure 2.1: Received errors after synthesis run

1) A syntax error was located on top_module was located and fixed by changing “s_ouput_1” to “s_output_1” (Figure 2.2),

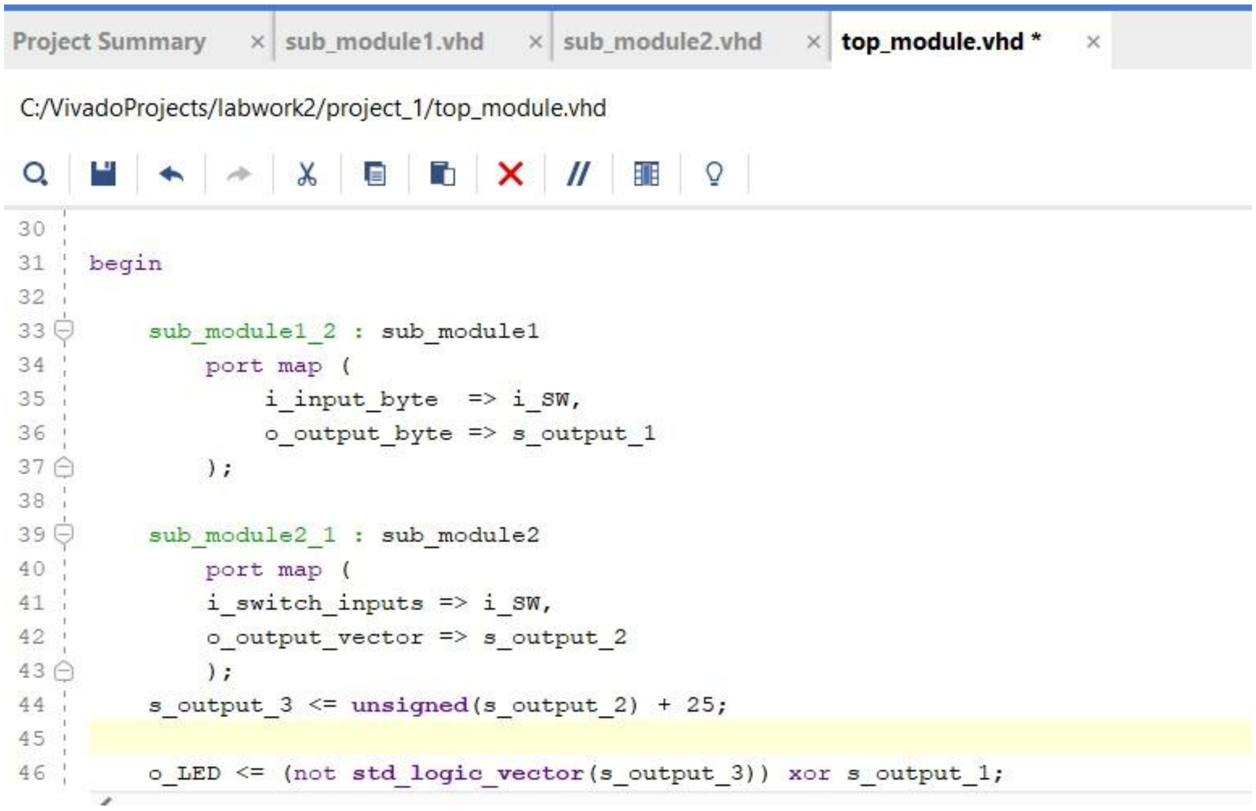


C:/VivadoProjects/labwork2/project_1/top_module.vhd

```
22          port (
23              i_switch_inputs : in  STD_LOGIC_VECTOR (7 downto 0);
24              o_output_vector : out STD_LOGIC_VECTOR (7 downto 0)
25          );
26      end component sub_module2;
27
28      signal s_output_1, s_output_2 : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
29      signal s_output_3             : unsigned(7 downto 0)         := (others => '0');
30
31  begin
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
```

Figure 2.2: Line 36 of top_module.vhd after the syntax error was removed

2) The wrong order for the port map was corrected by editing the 40th line in the top module file (Figure 2.3),

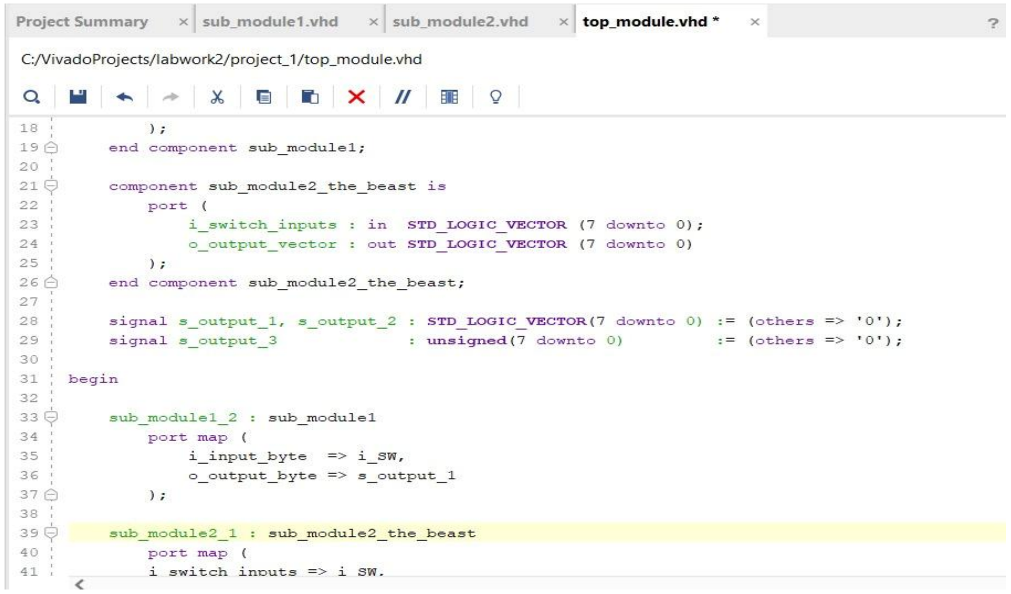


C:/VivadoProjects/labwork2/project_1/top_module.vhd

```
30
31  begin
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 (
41          i_switch_inputs => i_SW,
42          o_output_vector => s_output_2
43          );
44      s_output_3 <= unsigned(s_output_2) + 25;
45
46      o_LED <= (not std_logic_vector(s_output_3)) xor s_output_1;
```

Figure 2.3: Lines 39-43 after the ordering of the program was corrected

3) Every instance of “sub_module2” was changed to “sub_module2_the_beast” due to the fact that the Sources tab in Vivado referred to the file as such (Figure 2.4),

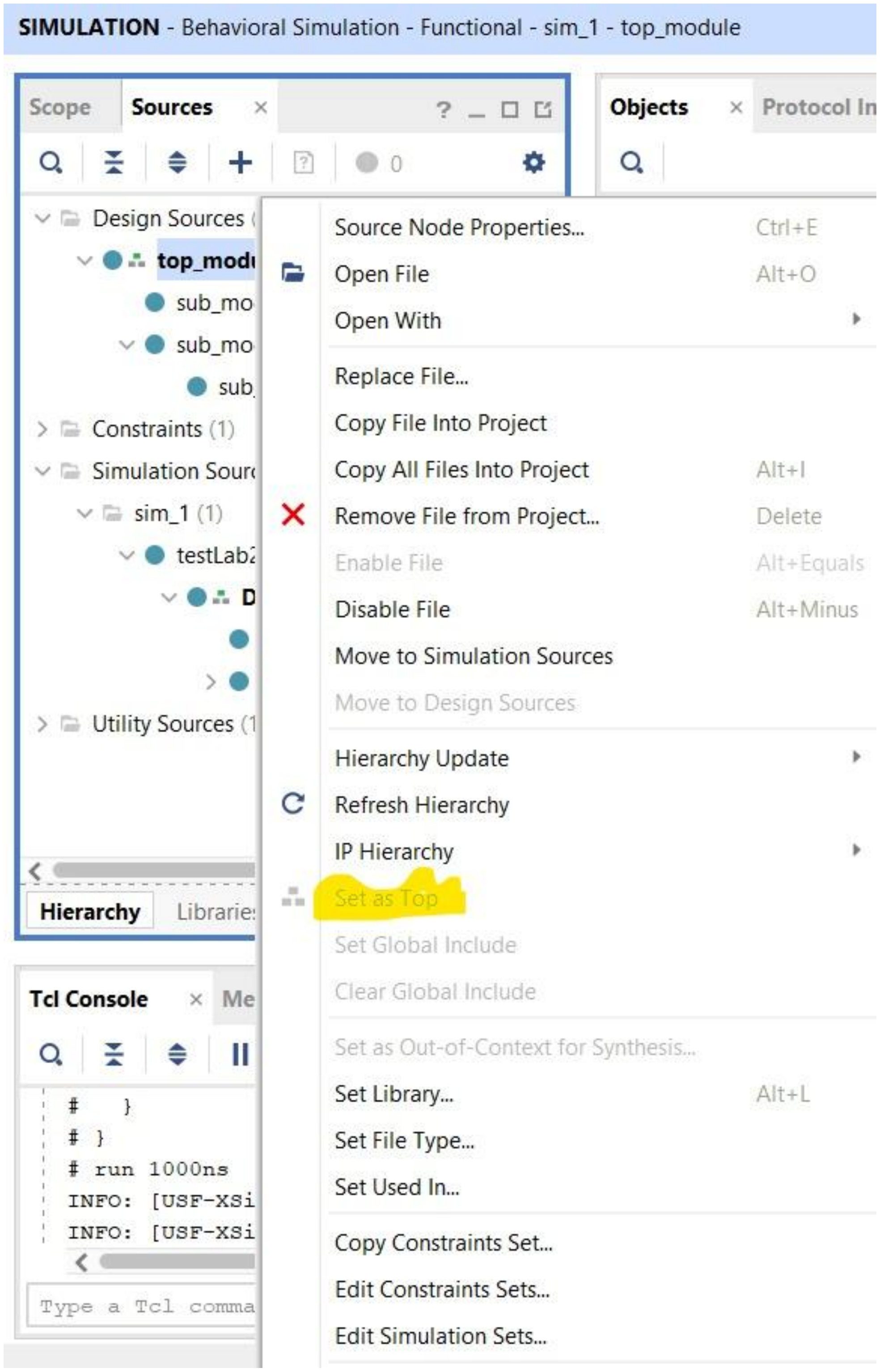


C:/VivadoProjects/labwork2/project_1/top_module.vhd

```
18          );
19      end component sub_module1;
20
21      component sub_module2_the_beast is
22          port (
23              i_switch_inputs : in  STD_LOGIC_VECTOR (7 downto 0);
24              o_output_vector : out STD_LOGIC_VECTOR (7 downto 0)
25          );
26      end component sub_module2_the_beast;
27
28      signal s_output_1, s_output_2 : STD_LOGIC_VECTOR(7 downto 0)  := (others => '0');
29      signal s_output_3             : unsigned(7 downto 0)          := (others => '0');
30
31  begin
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_the_beast
40          port map (
41          i_switch_inputs => i_SW,
```

Figure 2.4: top_module.vhd after editing the file to add “sub_module2_the_beast”

4) In the Sources tab in Vivado, the top_module was right clicked and set as top by clicking the related button in order to get the right simulation waveform for future steps (Figure 2.5).

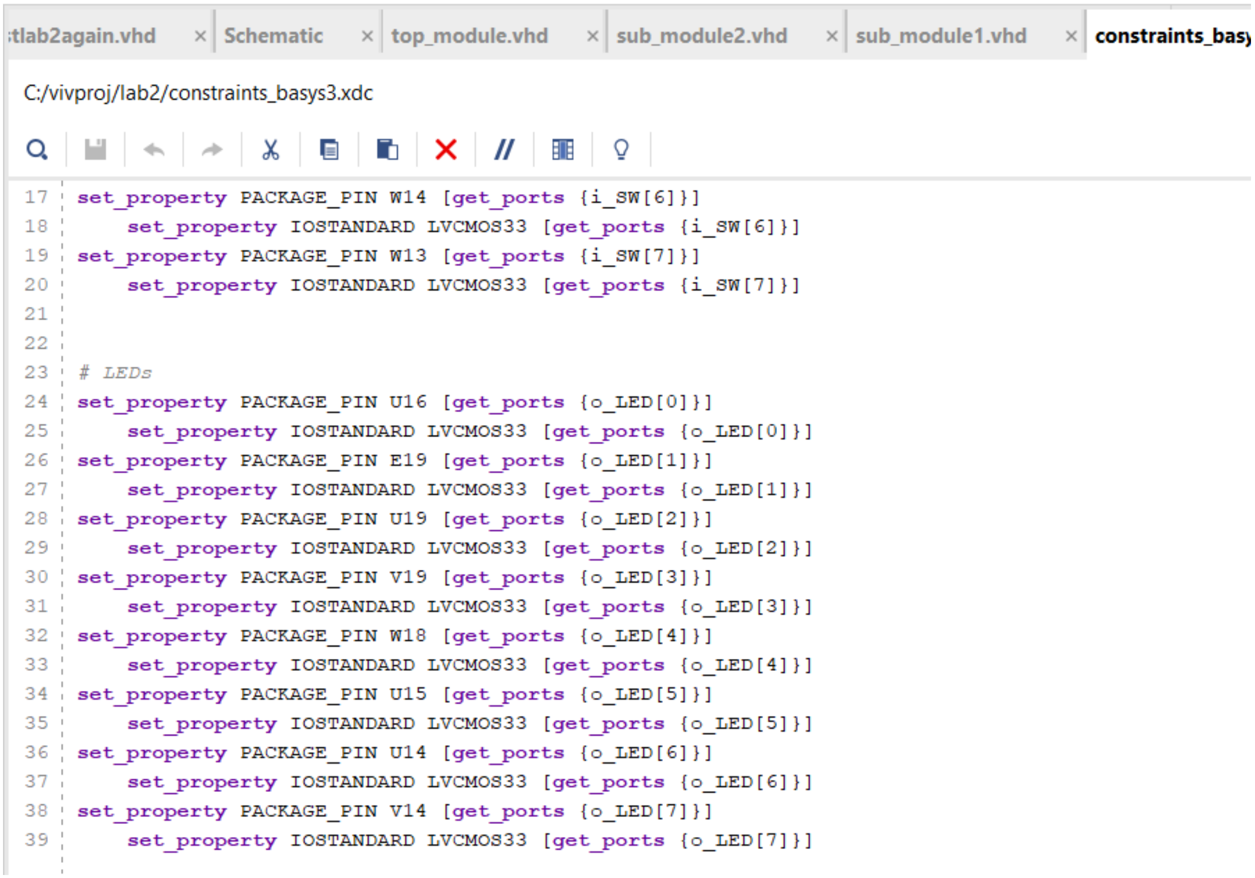


```
#   }
# }
# run 1000ns
INFO: [USF-XSi
INFO: [USF-XSi
Type a Tcl comma
```

Figure 2.5: Set as Top button for top_module.vhd highlighted after clicking

After these changes were made, no errors were observed in the Message tab of Vivado. Due to this reason, each file in the program was individually analyzed. After this analysis, two errors were observed in the constraints file. The LED layout had these errors in it. To fix these errors, the following steps were taken:

5) The package pin for line 24 and 38 (V14 and U16) were swapped to fit the BASYS3 manual (Figure 2.6),
6) A missing curly bracket in line 33 was filled in (Figure 2.6).



C:/vivproj/lab2/constraints_basys3.xdc

```
17  set_property PACKAGE_PIN W14 [get_ports {i_SW[6]}]
18      set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[6]}]
19  set_property PACKAGE_PIN W13 [get_ports {i_SW[7]}]
20      set_property IOSTANDARD LVCMOS33 [get_ports {i_SW[7]}]
21
22
23  # LEDs
24  set_property PACKAGE_PIN U16 [get_ports {o_LED[0]}]
25      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[0]}]
26  set_property PACKAGE_PIN E19 [get_ports {o_LED[1]}]
27      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[1]}]
28  set_property PACKAGE_PIN U19 [get_ports {o_LED[2]}]
29      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[2]}]
30  set_property PACKAGE_PIN V19 [get_ports {o_LED[3]}]
31      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[3]}]
32  set_property PACKAGE_PIN W18 [get_ports {o_LED[4]}]
33      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[4]}]
34  set_property PACKAGE_PIN U15 [get_ports {o_LED[5]}]
35      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[5]}]
36  set_property PACKAGE_PIN U14 [get_ports {o_LED[6]}]
37      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[6]}]
38  set_property PACKAGE_PIN V14 [get_ports {o_LED[7]}]
39      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[7]}]
```

Figure 2.6: connstraints_basys3.xdc after the errors were debugged

In total, six errors were debugged and the design was in a completely working state.

Task-3) After the required hardware implementations were taken -Generate Bitstream, Connect Device, Open Target, Program Device-, certain LEDs lit up on the FPGA board (Figure 3.1). On top of the default state, five more different input and output states were implemented on the BASYS3 board and observed (Figure 3.2-3.6). The observed values matched the expected results from the implemented gates.

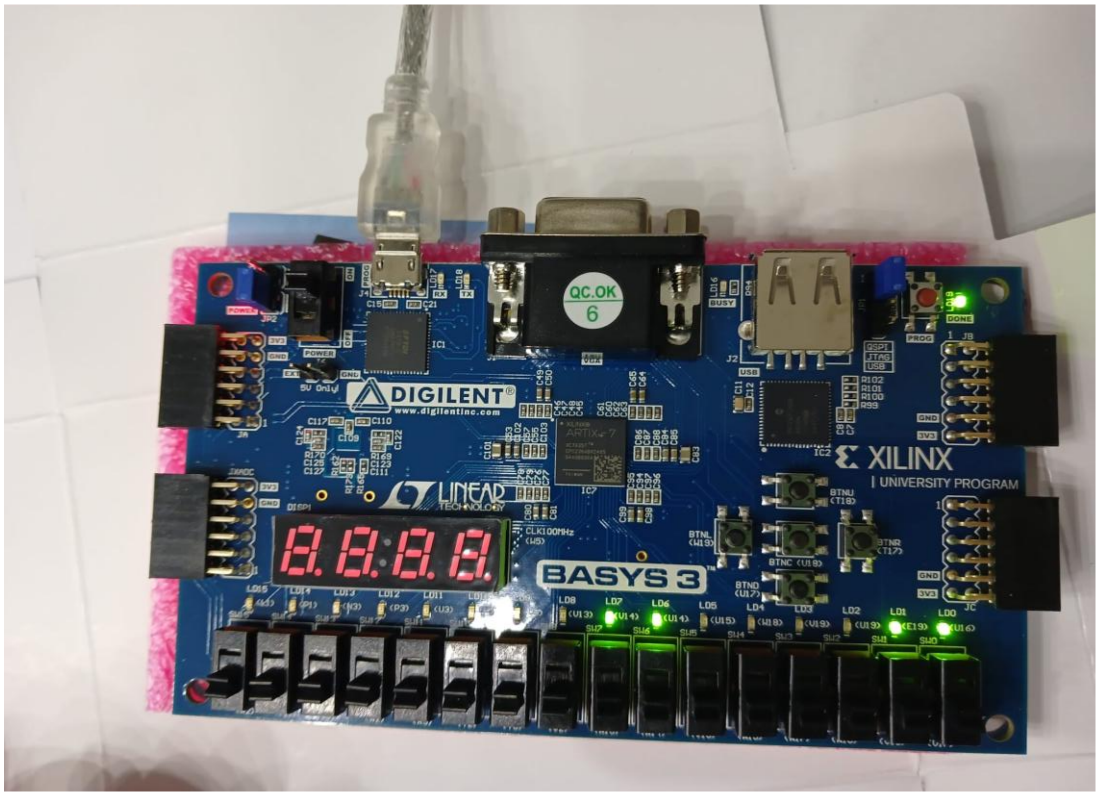


Figure 3.1: Default state of the BASYS3

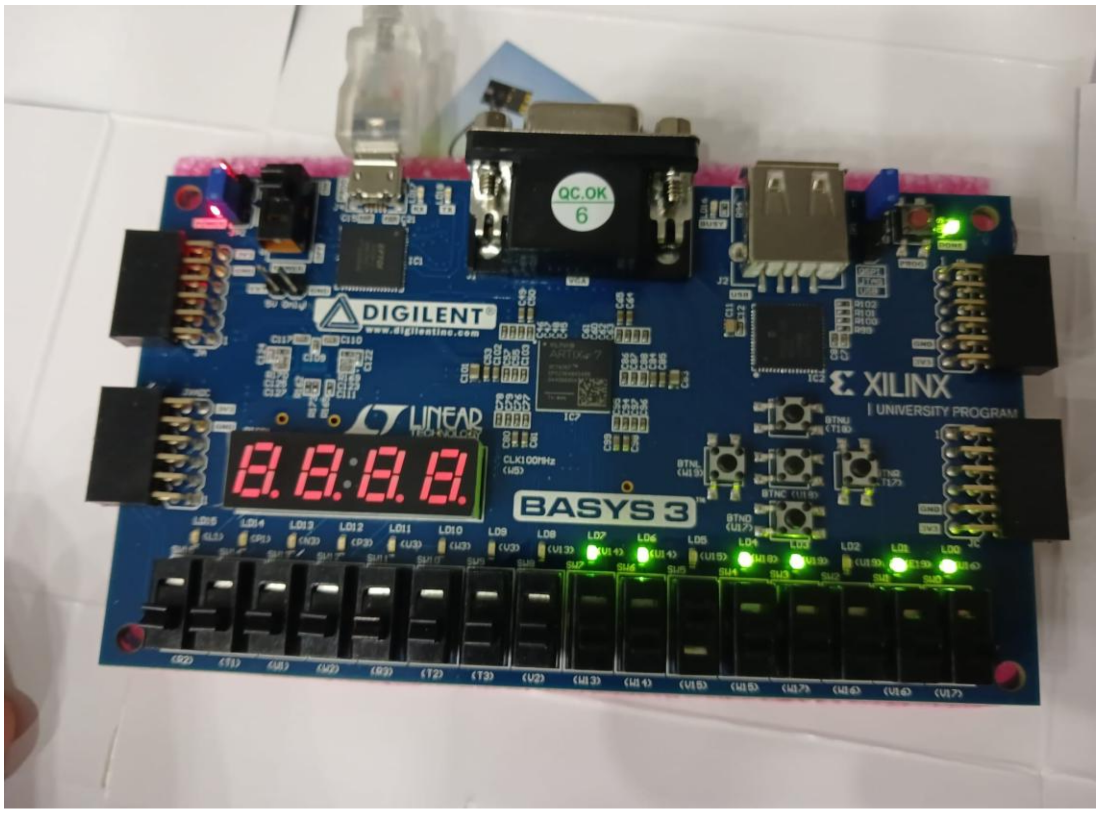


Figure 3.2: State of the BASYS3 when SW5 is on

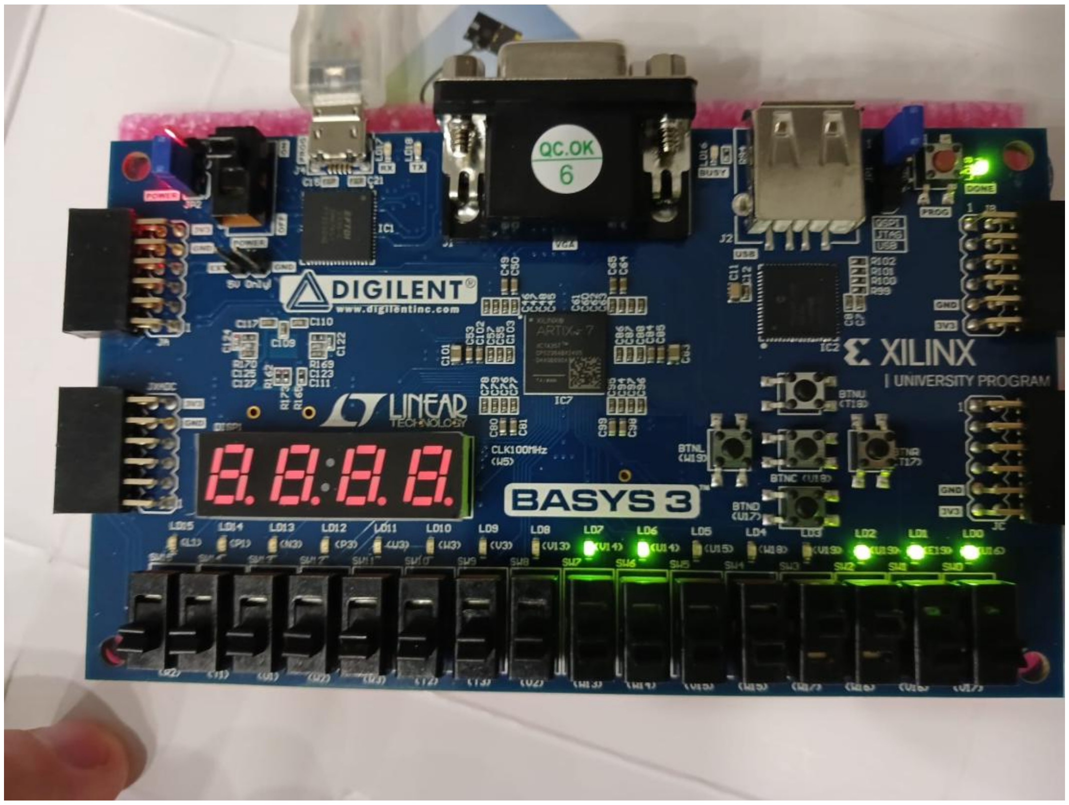


Figure 3.3: State of the BASYS3 when SW2 and SW3 are on

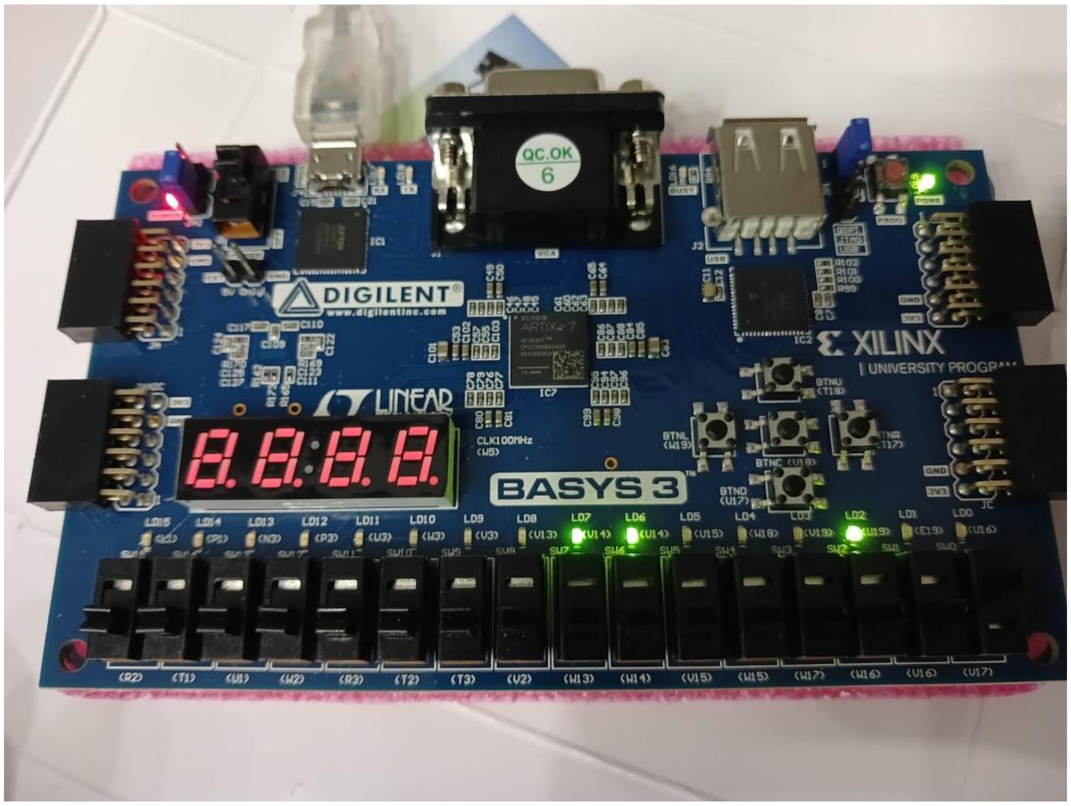


Figure 3.4: State of the BASYS3 when SW0 is on

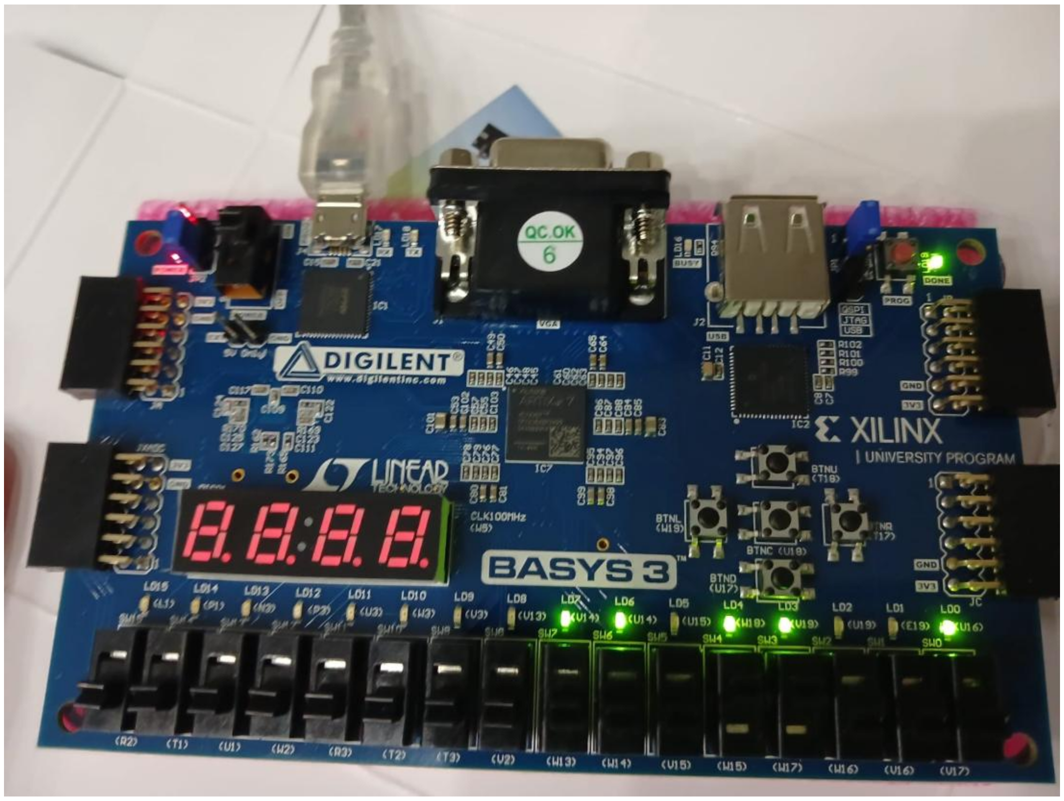


Figure 3.5: State of the BASYS3 when SW3 and SW4 are on

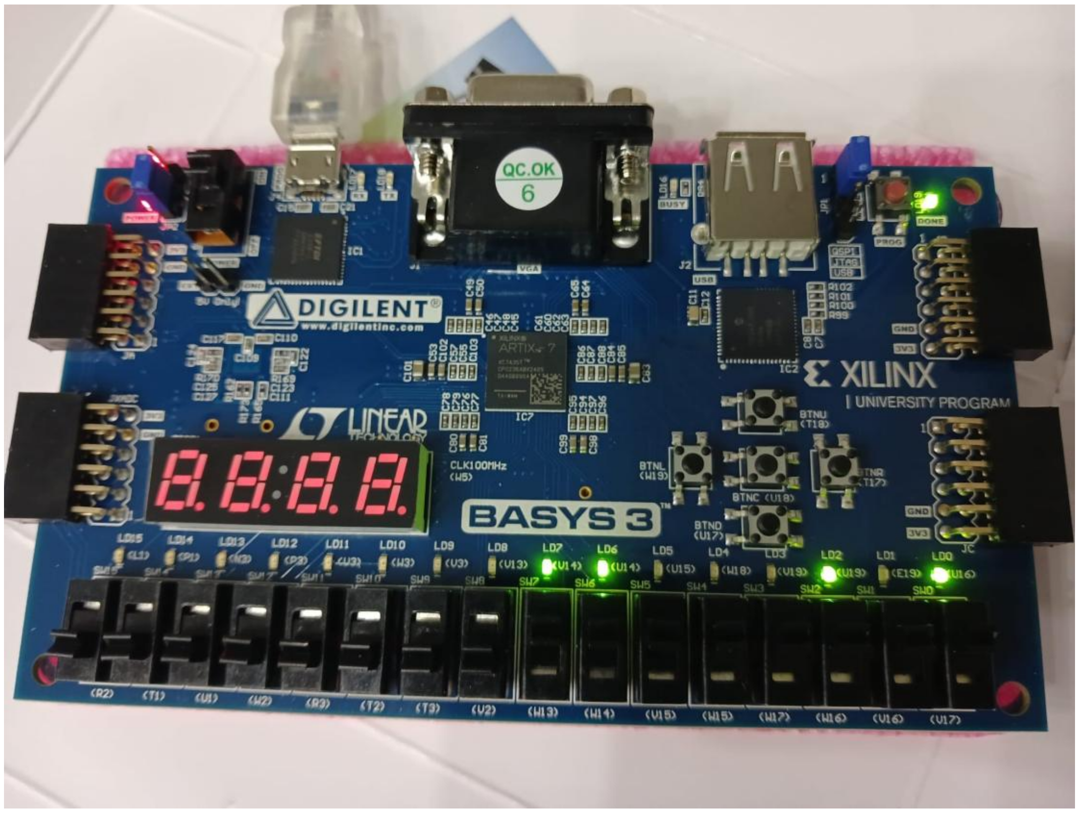
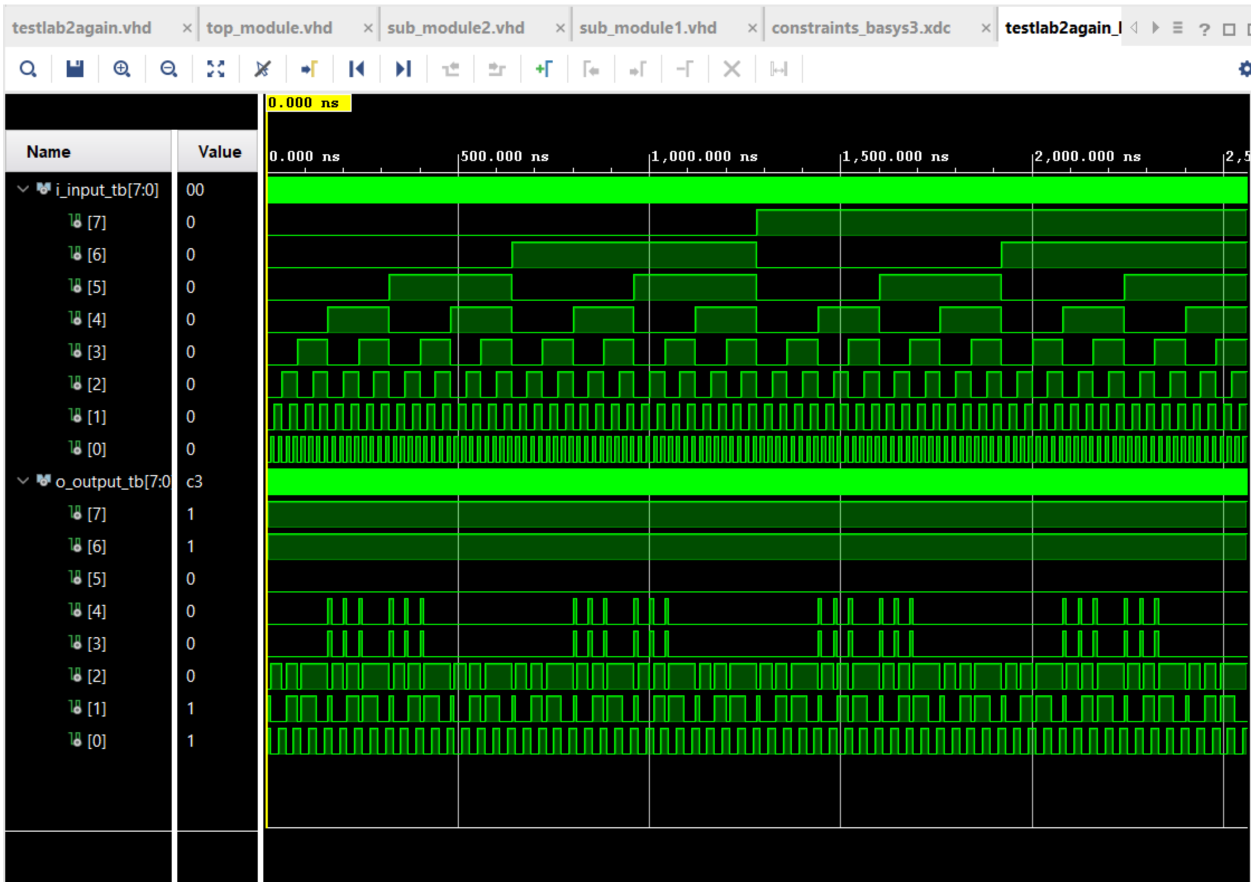


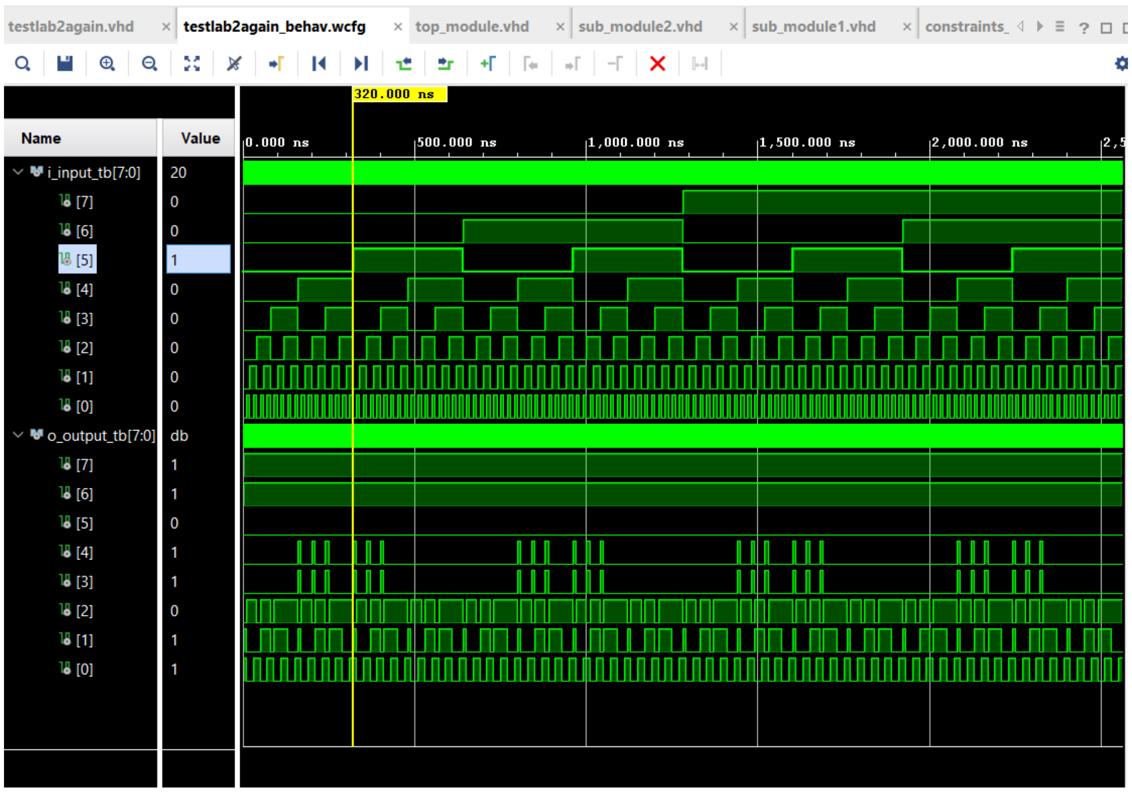
Figure 3.6: State of the BASYS3 when switches between SW0 and SW7 are on

Task-4) The programmed testbench iterated through every integer between zero and two hundred and fifty-five. These products were turned into a waveform in binary in order to create a systematic waveform input that could be ran through the modules to get an environment that could be parsed freely to check the ideal outputs of the related inputs. A simulation was ran using the Run Simulation function of Vivado. Since the correct module was set to top to be simulated, no errors were ran into when the simulation was ran. A waveform configuration was observed and matched with the related input in the switch configurations that were gathered in the previous task (Figure 4.1-4.6).



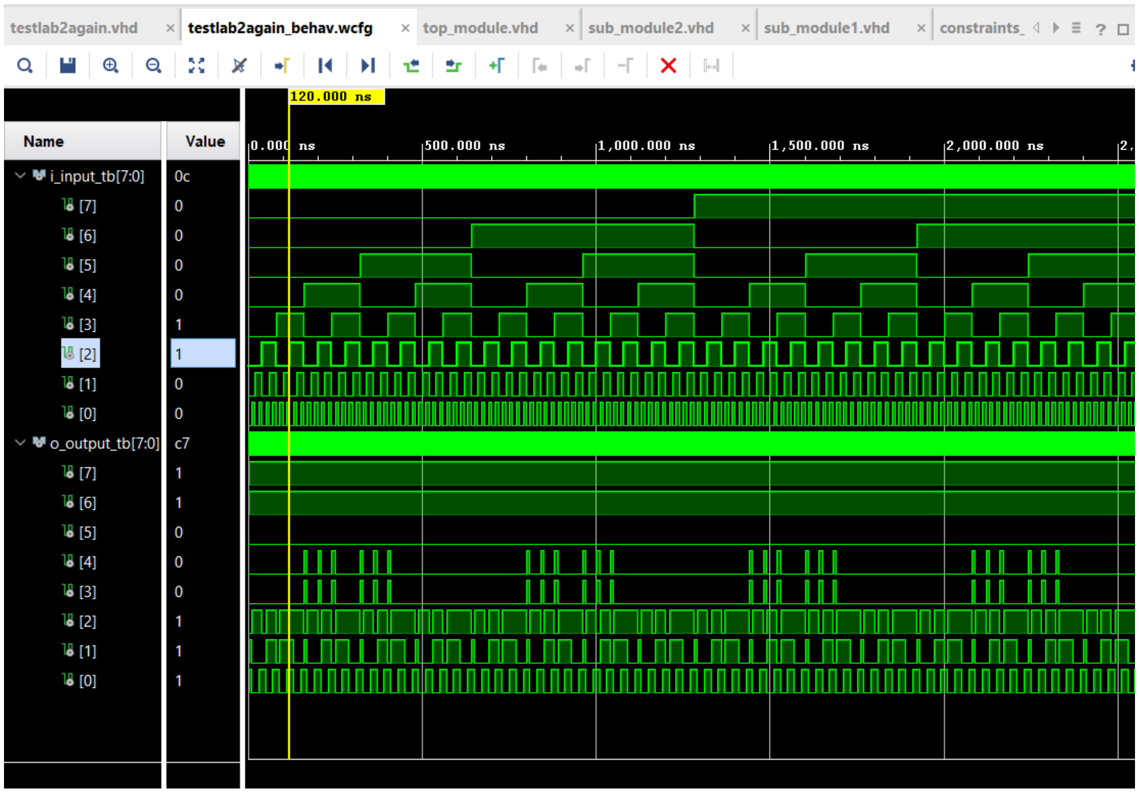
Name	Value
i_input_tb[7:0]	00
[7]	0
[6]	0
[5]	0
[4]	0
[3]	0
[2]	0
[1]	0
[0]	0
o_output_tb[7:0]	c3
[7]	1
[6]	1
[5]	0
[4]	0
[3]	0
[2]	0
[1]	1
[0]	1

Figure 4.1: Simulation of the board state in Figure 3.1



Name	Value
i_input_tb[7:0]	20
[7]	0
[6]	0
[5]	1
[4]	0
[3]	0
[2]	0
[1]	0
[0]	0
o_output_tb[7:0]	db
[7]	1
[6]	1
[5]	0
[4]	1
[3]	1
[2]	0
[1]	1
[0]	1

Figure 4.2: Simulation of the board state in Figure 3.2



Name	Value
i_input_tb[7:0]	0c
[7]	0
[6]	0
[5]	0
[4]	0
[3]	1
[2]	1
[1]	0
[0]	0
o_output_tb[7:0]	c7
[7]	1
[6]	1
[5]	0
[4]	0
[3]	0
[2]	1
[1]	1
[0]	1

Figure 4.3: Simulation of the board state in Figure 3.3

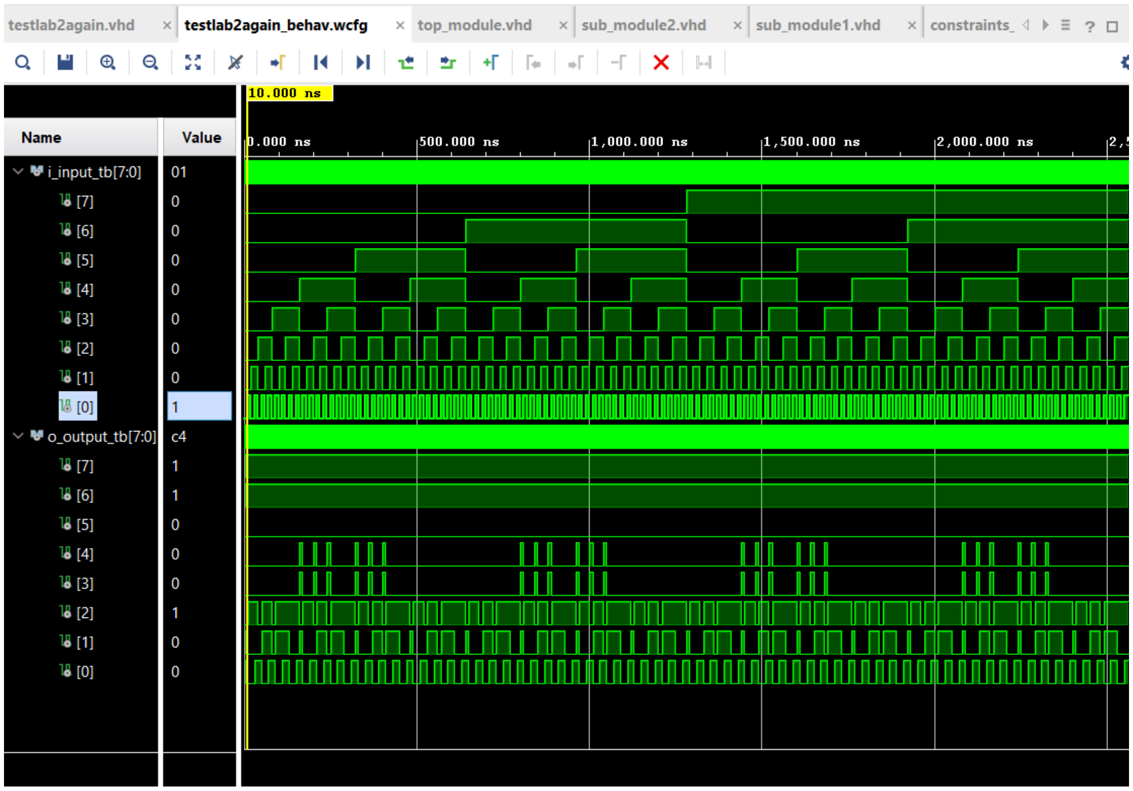


Figure 4.4: Simulation of the board state in Figure 3.4

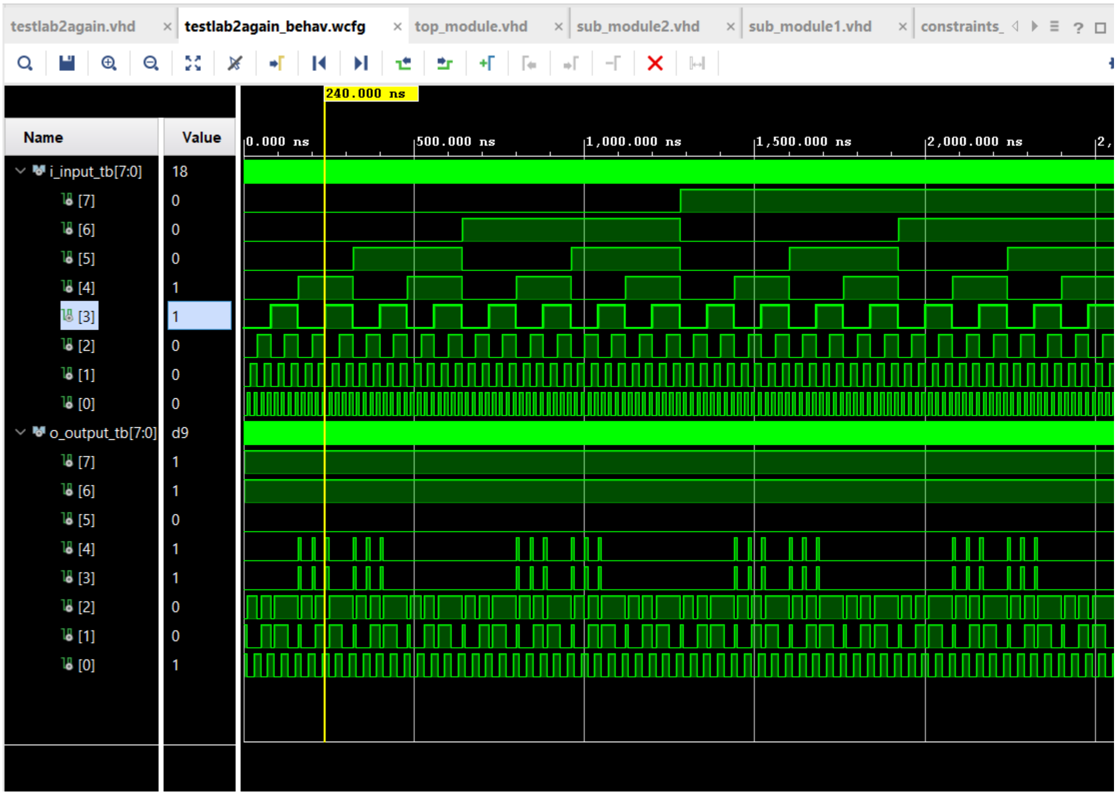


Figure 4.5: Simulation of the board state in Figure 3.5

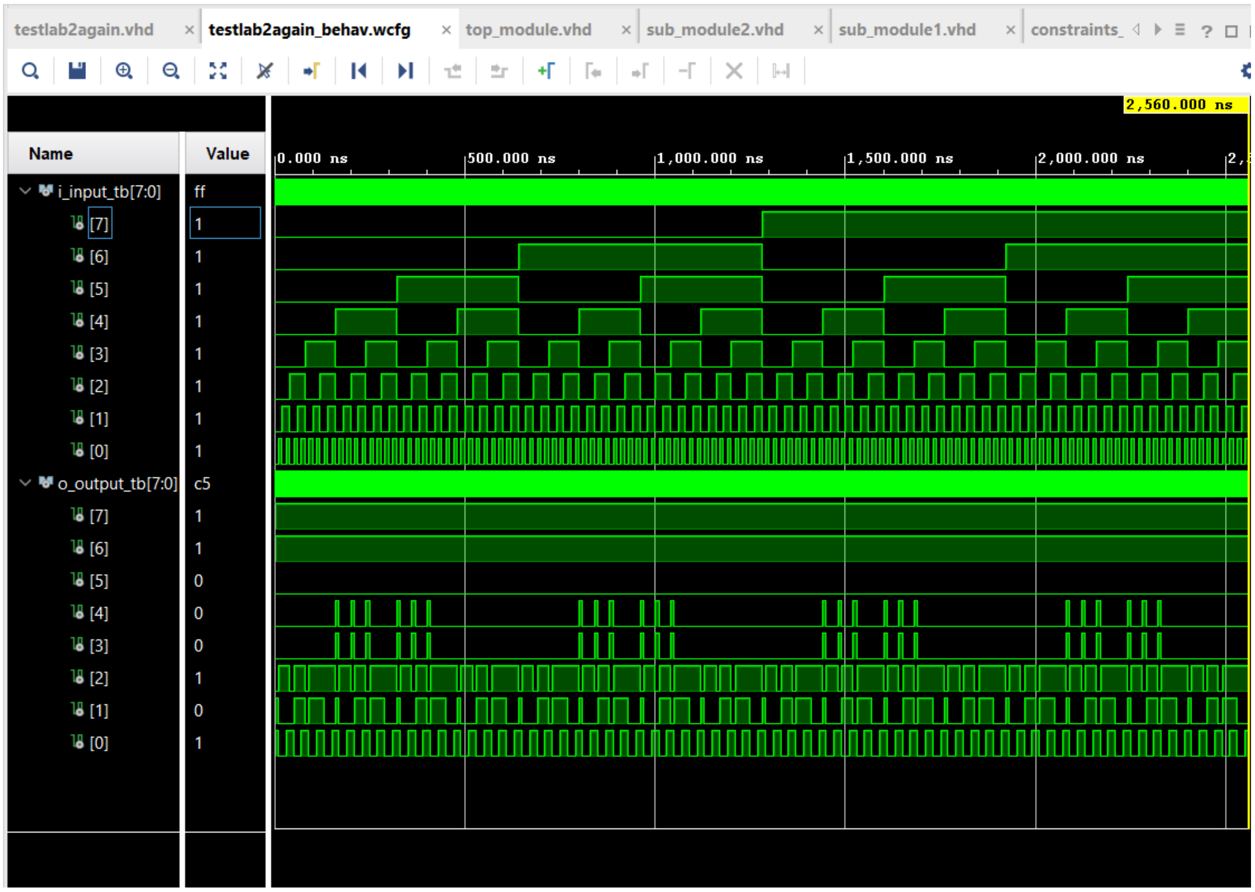


Figure 4.6: Simulation of the board state in Figure 3.6

Through analyzing each waveform, it was observed that for each input in the simulation matched the real-world LED output of the board. This meant that the simulation was successful and was working as expected.

Task-5) When each schematic was being generated, the program encountered zero errors. The RTL schematic (Figure 5.1) represents the three 2-input logic gates that were entered to the submodule inside the top module. This represents the main function of the program of getting inputs and processing them. The other schematics, the implemented and the synthesized schematic, represented the design tools structural layout as a logical function of eight binary input values (Figure 5.2). These schematics had the same representation since the project had no given time constraints and was made out of simple logic gates, making them identical in function.

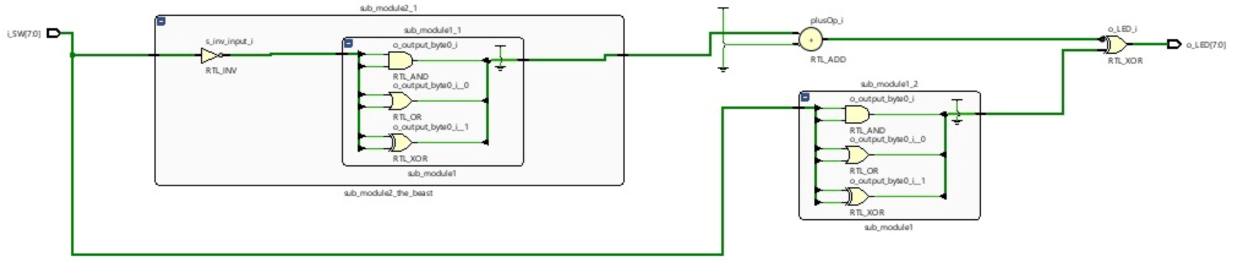


Figure 5.1: RTL Schematic

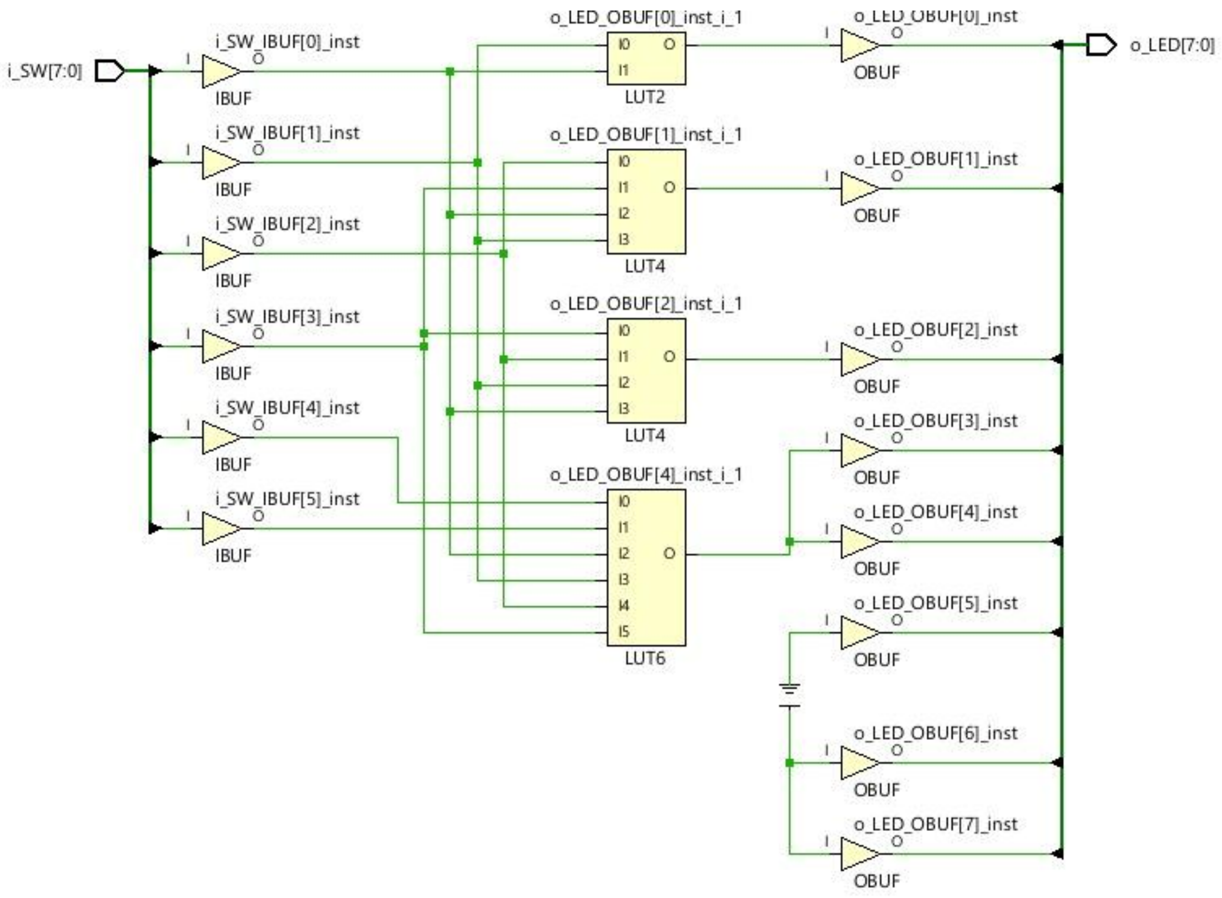


Figure 5.2: Implemented Schematic and Synthesized Schematic

Task-6) The following is every line of code that was used in the project:

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_the_beast 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_the_beast;

    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_the_beast
        port map (
        i_switch_inputs => i_SW,
        o_output_vector => 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;
```

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


end Structural_sub1;
```

sub_module2_the_beast.vhd

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


entity sub_module2_the_beast 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_the_beast;


architecture Structural_sub2 of sub_module2_the_beast 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;
```

<u>constraints_basys3.xdc</u>

```
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]}]
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]}]
```

testlab2again.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
-- default libraries are sufficient for this specific test bench

entity testlab2again is
end testlab2again;


Architecture Behavioral of testlab2again is
```

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

end Component;



Signal i_input_tb : std_logic_vector (7 downto 0) := (others => '0');
-- initial inputs are taken as zero
Signal o_output_tb : std_logic_vector (7 downto 0):= (others => '0');
-- same as input testbench values since the same type and amount of data is expected

begin
UUT: top_module
--defining the module for the output
Port Map(
 i_SW => i_input_tb,
 o_LED => o_output_tb
 --mapping each input and output to our design
 );

stim_proc: process
--actual process code for the testbench

begin
for s in 0 to 255 loop
--our range is defined for 8 binary vallues
  i_input_tb <= std_logic_vector(to_unsigned(s,8));
  --testbench input is assigned
  wait for 10 ns;
  --duration of each impusle is defined
end loop;
wait;
end process;
--program is finished

end Behavioral;
```

Conclusion:

The purpose of this laboratory assignment was to introduce basic concepts of FGPAs, VHDL and the BASYS3 board through debugging the main code and forming a testbench for simulations. After the main program was debugged out of six errors and simulated, an output determining real-world states of the logic board was formed. Additionally, through pre-experimentation research, certain programmable logic concepts were learnt such as port maps, testbenches and modules. The experiment also pushed experimenters to write individual testbench programs for general applications and signal generation applications.

References:

Brown, S., & Vranesic, Z. (2023). *Fundamentals of digital logic with VHDL design* (4th ed.). McGraw- Hill.

Digilent, Inc. (n.d.). *Basys 3 reference manual.* Digilent. Retrieved February 19, 2025, from https://digilent.com/reference/programmable-logic/basys-3/reference-manual?srsltid=AfmBOorb8gUx_cAXfwmU4GnXTQ33cTBA0kqw2g9RZtoN0aPWdStd3ih6

Xilinx, Inc. (2022). Vivado synthesis: User guide (UG901). Xilinx. Retrieved February 19, 2025, from https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug901-vivado-synthesis.pdf