Alp Tekin

22403597

EE102-02

8.10.2025

Bilkent University

Electrical and Electronics Engineering Department

Lab 2 Report: Introduction to VHDL

Purpose

The aim of this experiment is to learn how to design, debug and test a combinational logic circuit by using VHDL and Vivado Design Suite on Basys3 Board. Also, it includes learning how to implement and test the corrected VHDL design on Basys3 board. Additionally, we learned how to write a testbench code and how to simulate it in Vivado to verify that the design behaves properly.

Design Specifications

In this experiment, I used Vivado Standard Edition 2024.2 to debug a circuit designed in VHDL. The given combinational circuit logic design consists a top_module and two submodules named sub_module1 and sub_module2. The top_module receives an 8-bit input vector (i_SW) from switches on Basys3 Board and produces an 8-bit output vector (o_LED) which are displayed on LEDs. The sub_module1 process the input byte by applying logical operations to specific input bits. For my ID configuration, the first three output bits are obtained through OR, OR and XOR gates. The remaining output bits are assigned fixed binary values, for instance [5:3] is assigned to “010” and [7:6] the most significant two bits assigned to zero. In the top-level architecture, the output of sub_module2 is converted to unsigned type and added with a value of 25 by producing the signal s_output_3. Finally, the LEDs display the inverted s_output_3 XOR the output of sub_module1. I also created a testbench file to verify the functionality of my circuit through simulation. The testbench applies all possible 8-bit input combinations by using for loop and a wait statement. This allows me to observe output waveforms and check them in Vivado.

Methodology

In VHDL, the inputs and outputs of a module are defined in the entity part of the code. The entity specifies the ports of the module. It includes their names, directions (in, out, inout) and data types such as STD_LOGIC or STD_LOGIC_VECTOR. In our example, the top_module entity declares an 8-bit input vector (i_SW) and 8-bit output vector (o_LED). In VHDL, one module can use another by declaring it as a component inside its architecture.

Task 1: In the code section we are given, a design file contains the VHDL description of a combinational circuit. Using the last digit of my ID number**,** I changed the logic gates from submodule1.vhd. Lines 14-16 contained generic logical operations and I replaced them with the required logic gates. Since the last digit of my student ID is 7, I changed the logic operations with OR, OR and XOR.

Task 2: There are 6 errors that needed to be solved in this task.

1. First bug occurred because the file and entity names did not match the one which top_module was calling. The given file name was sub_module2_the_beast, while inside the top_module code it was defined as sub_module2. To fix this issue, I renamed the file as sub_module2 to ensure that the code works properly.
2. The second error was a minor typo mistake which occurred in top_module file in PORT MAP section. The PORT MAP statement is used to connect internal signals of the high-level module to the ports of the submodule. In this design, the top_module includes two submodules named sub_module1 and sub_module2 which are connected through PORT MAP. In my example the o_output_byte was assigned to s_output_1, where the first letter “t” in s_output_1 was missing. This caused an error but after correcting the typo, the issue was solved.
3. The third error was happened due to incorrect port connections in PORT MAP statement inside the top_module. While connecting the sub_module2 in the top module, the input and output signals were connected in the wrong order. The input port i_switch_inputs was mistakenly connected to the s_output_2 and the o_output_vector was connected to input signal i_SW. This mismatch caused an error in Vivado. After swapping the i_SW

and s_output_2 signals the connections was made properly and the code worked correctly.

4. In the fourth error was another minor typo, this time in constraint file. In this design, i_SW signals are mapped to the physical switch pins and the o_LED signals are mapped to the LED pins on Basys3 board. On the line 28 of the constraint file, a closing brace "}" was missing, which caused an error. After adding the missing character, the problem solved.
5. Fifth error was also found in constraint file. The pin assignments for LED0 and LED7 were swapped. To fix this problem, I checked the Basys3 board reference manual and compared it with the constraint file. After correcting the pin mapping, the error was fixed.
6. The last problem was caused by selecting the wrong FPGA device on Vivado project settings. At first different project device was selected, so the synthesis and implementation steps could not work properly. After selecting the correct device, the project runs correctly and all errors were solved.

Task 3: After fixing all the errors, I synthesized and implemented the corrected code on the Basys3 FPGA board using Vivado. The constraint file plays a key role because it defines how the signals in VHDL are connected to the physical pins of the FPGA. It ensures that each input and output correspond to the hardware component on the board. I used the Basys3 Reference Manual to check which physical pins corresponds to which switch/LED etc. In this design, input vector i_SW[7:0] is mapped to eight on board switches and the output vector o_LED[7:0] is mapped to the eight LEDs. After running the synthesis and implementation steps, I generated the bitstream file. It was successfully programmed into my Basys3 board and the LEDs responded correctly to the switch inputs.

Task 4: To verify that the circuit worked correctly before implementation, I wrote a testbench. The purpose of writing a testbench is to check whether the VHDL code works properly or not. Before implementing the VHDL design on FPGA, it allows us to apply different input combinations and observe the output waveforms through simulation. In a testbench file, the entity section is empty because the testbench does not have any physical input or output ports. In my experiment, I used testbench to simulate all 8-bit inputs and observe the output waveforms to ensure

whether LEDs are working correctly. The testbench that I wrote applies 256 possible input combinations and wait 10 ns after each input was implemented.

Task 5: In this task, I examined the generated RTL, Synthesized and Implemented schematics in Vivado. The RTL schematic shows the logical structure directly from the VHDL code. It clearly displays the hierarchy between top_module, sub_module1 and sub_module2 and the main signals such as i_SW, o_LED, i_switch_inputs[7:0]. The Synthesized schematic represents circuit after Vivado translated and optimized the original VHDL codes into logic elements. Vivado shows how each input and output signal connected through actual logic gates that will be used inside FPGA. The Implemented schematic represents the final design where the circuit is fully mapped onto physical sources of the FPGA. Unlike RTL or Synthesized schematic, which mainly show logical connections, the implemented schematic provides a hardware representation. In this part, I observed that schematics and the simulation results were consistent with the schematic views. I conclude that the combinational circuit works as expected.

Results

Task 1: I modified the given logic gates in sub_module1 according to last digit of my ID number. I used OR, OR and XOR logic operations and the output bits behaves as expected. In Figure 1.1, the part that I have changed can be seen.

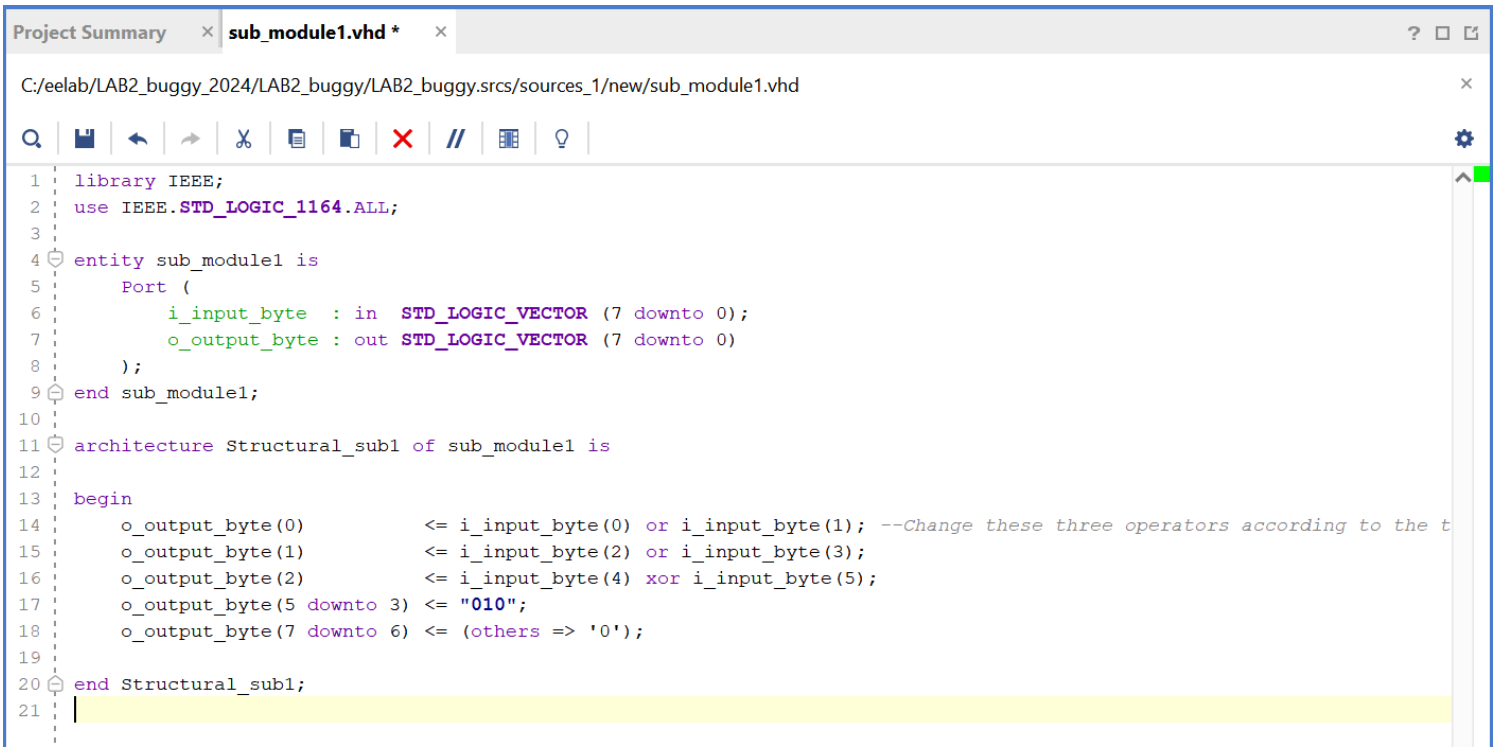

Figure 1.1 Logic Operations

Task 2: During the debugging process, I found 6 errors in VHDL design files.

1. In the first the file name was originally sub_module2_the_beast and caused an error in the design as can be seen in Figure 2.1.1. As shown in Figure 2.1.2, I renamed the file as sub_module2 in the code the problem was solved and Vivado recognized the module correctly.

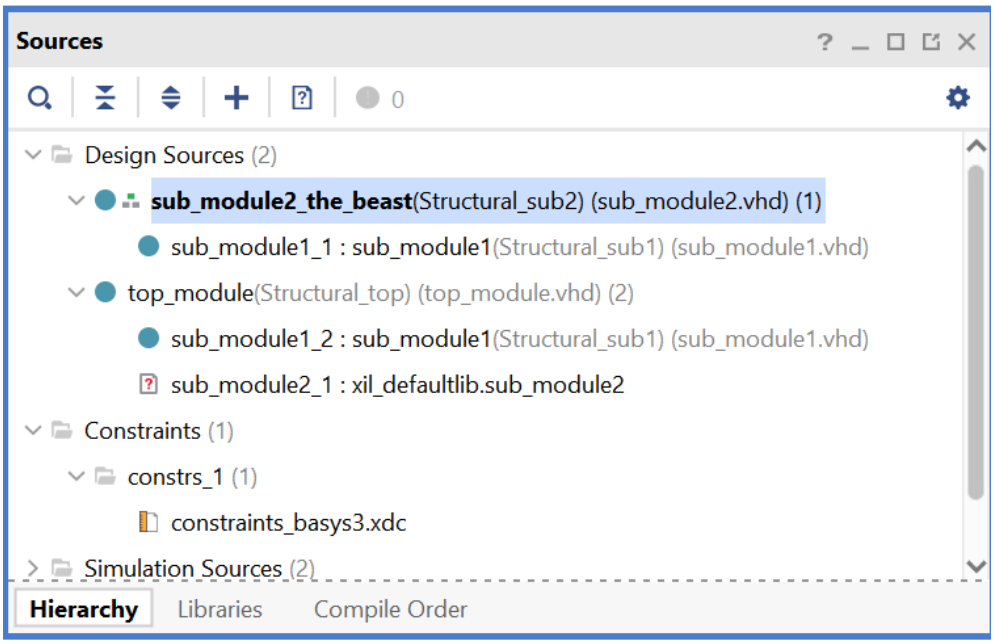

Figure 2.1.1 Wrong File Name

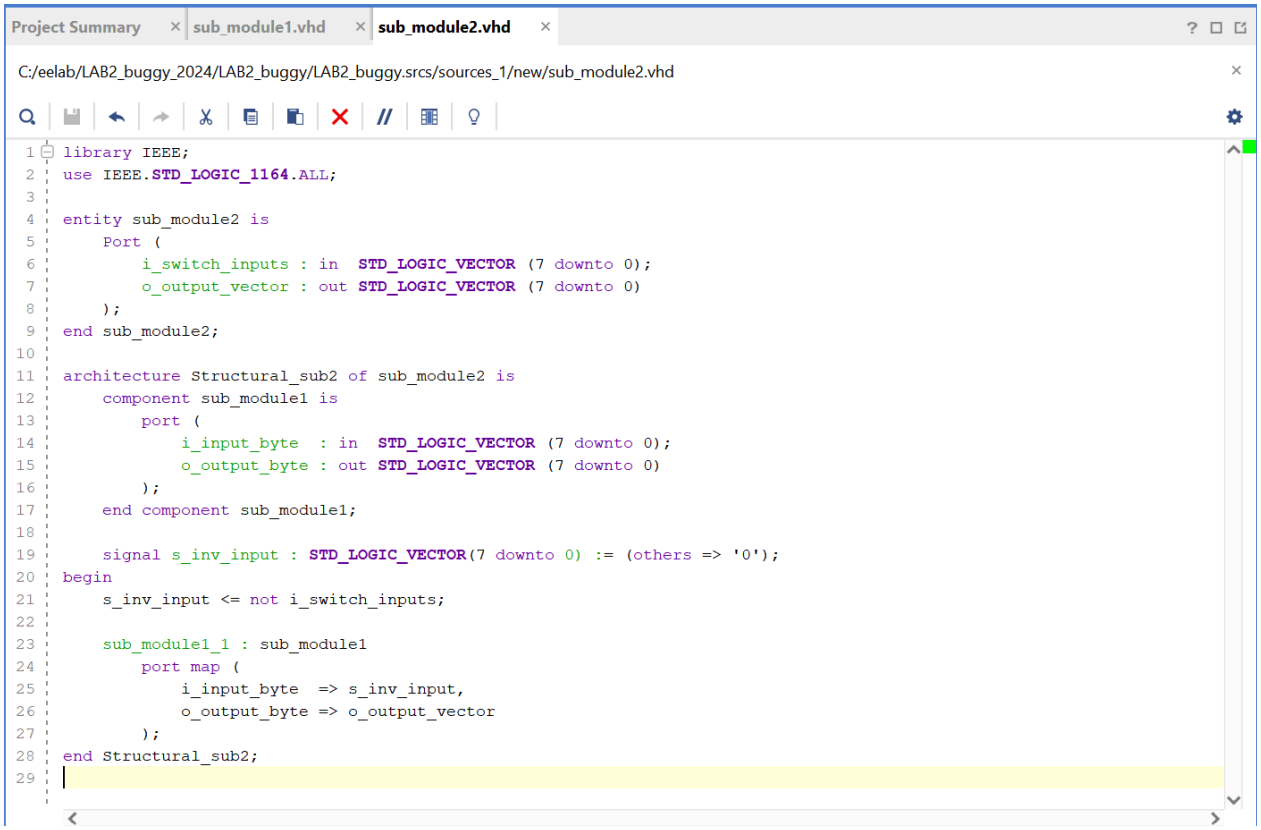

Figure 2.1.2 Corrected Code with sub_module2

2. I observed an error message as can be seen in Figure 2.2.1. I solved the typo by adding the letter “t” to s_output_1, as shown in Figure 2.2.2

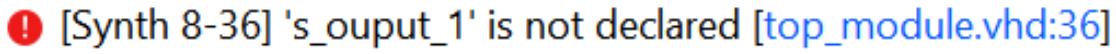

Figure 2.2.1 Error Message due to Typo

```
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 2.2.2 Corrected s_output_1 Signal Name

3. As shown in Figure 2.3.1, I found that the input and output pins were connected in the wrong order. I swapped the signals i_SW and s_output_2 (Figure 2.3.2). Vivado no longer displayed an error.

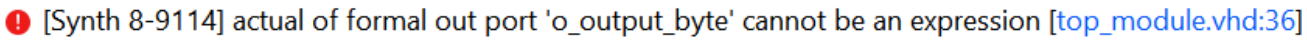

Figure 2.3.1 Wrong Port Mapping

```
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 2.3.2 Corrected Port Mapping

4. In Figure 2.4.1 an error message was occurred because of a syntax error in the constraint file. After adding missing "}" character in line 28, I solved the problem (Figure 2.4.2).

Figure 2.4.1 Syntax Error

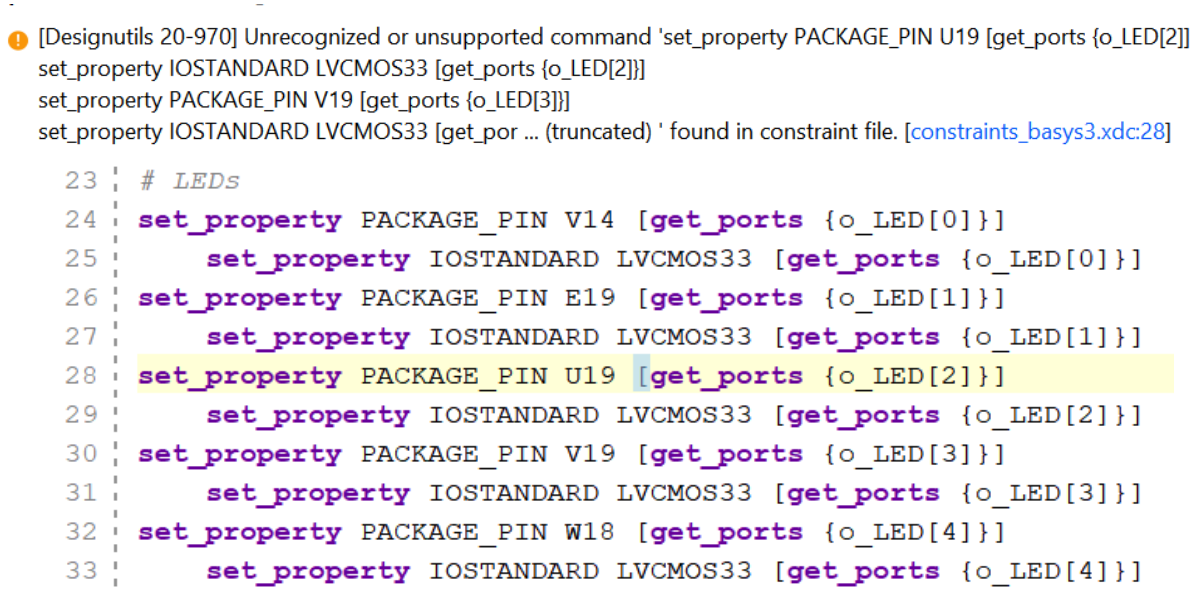

Figure 2.4.2 Corrected Constraint File

5. There is a wrong pin assignment in constraint file, as shown in Figure 2.5.1. The o_LED[0] was connected to pin V14 and the o_LED[7] was connected to U16, which caused the LEDs to display incorrect outputs on Basys3. After correcting pin mapping, the LEDs worked properly (Figure 2.5.2)

```
23  # LEDs
24  set_property PACKAGE_PIN V14 [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 U16 [get_ports {o_LED[7]}]
39      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[7]}]
```

Figure 2.5.1 Wrong Pin Assignment

```
23  # LEDs
24  set_property PACKAGE_PIN V14 [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 U16 [get_ports {o_LED[7]}]
39      set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[7]}]
```

Figure 2.5.2 Corrected Pin Assignment

6. I found that wrong device was selected for the project as shown in Figure 2.6.1. To fix this issue, I changed the project device to Basys3 board, which uses xc7a35tcpg236. After that, the problem was fixed

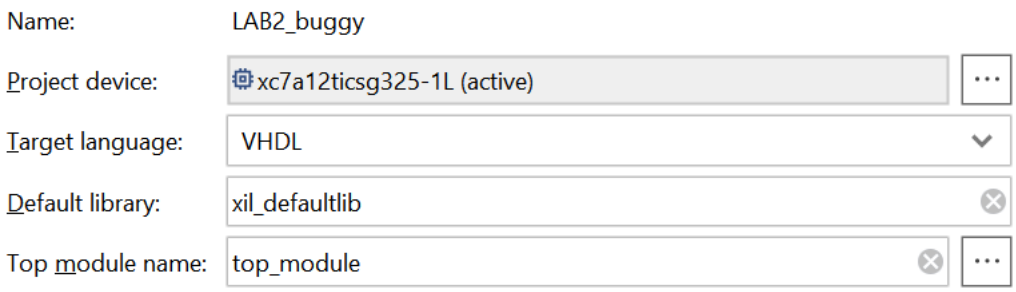

Figure 2.6.1 Wrong Device Selected

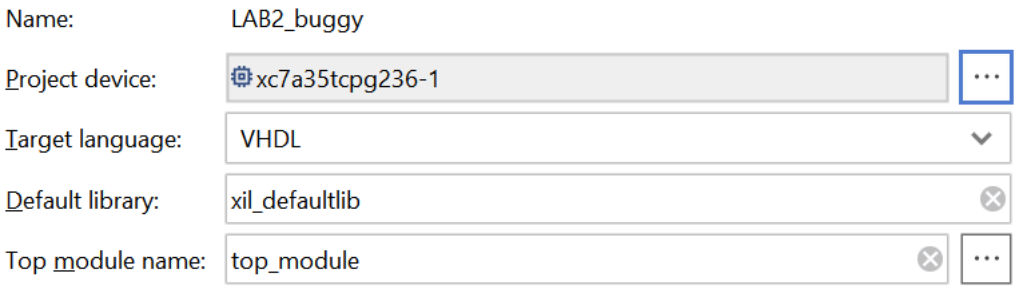

Figure 2.6.2 Basys3 Device Selected

Task 3: I implemented the design on Basys3 FPGA Board. I generated the bitstream file successfully and programmed onto device without any errors. The LEDs on the board responded correctly to the switch inputs. As shown in the five figures below with different inputs (Figure 3.1-3.5), each switch controlled the expected output LED according to the logic gates. I applied the constraint file and design files properly.

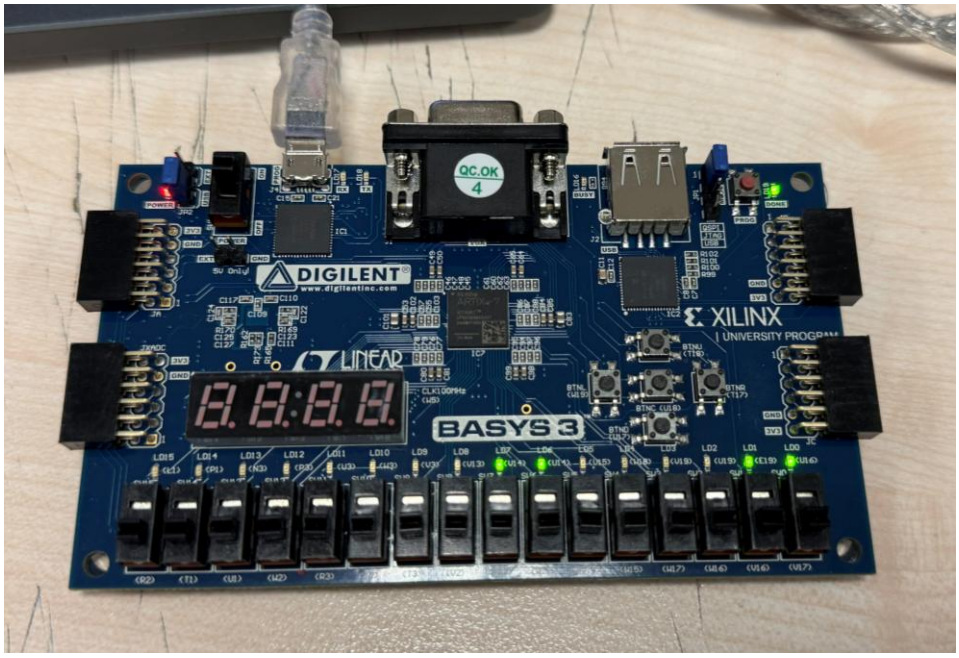

Figure 3.1 Input:00000000 Output:11000011

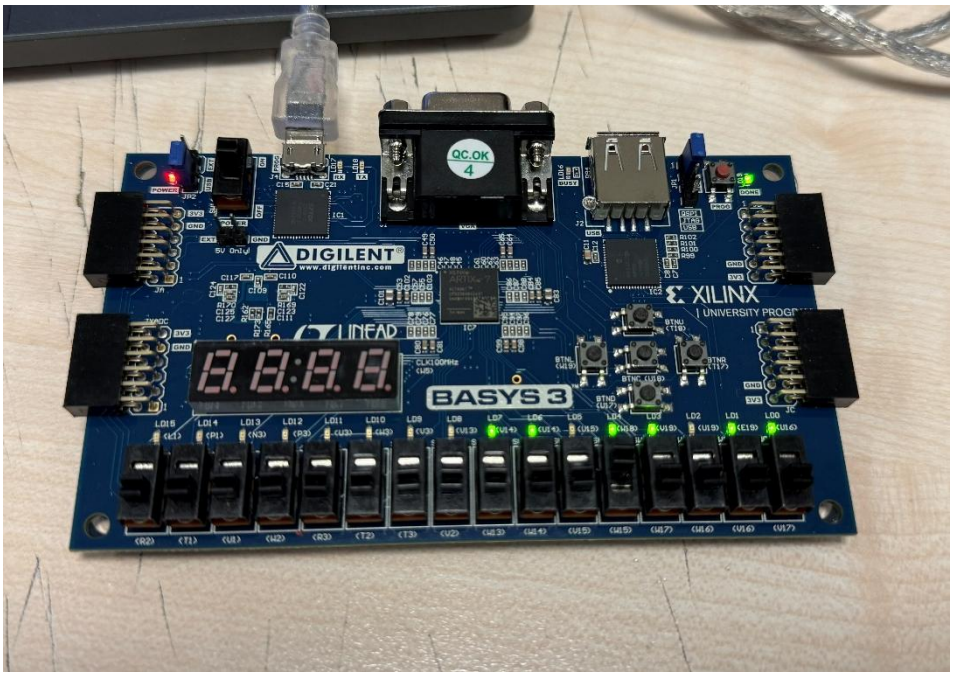

Figure 3.2 Input: 00010000 Output:11011011

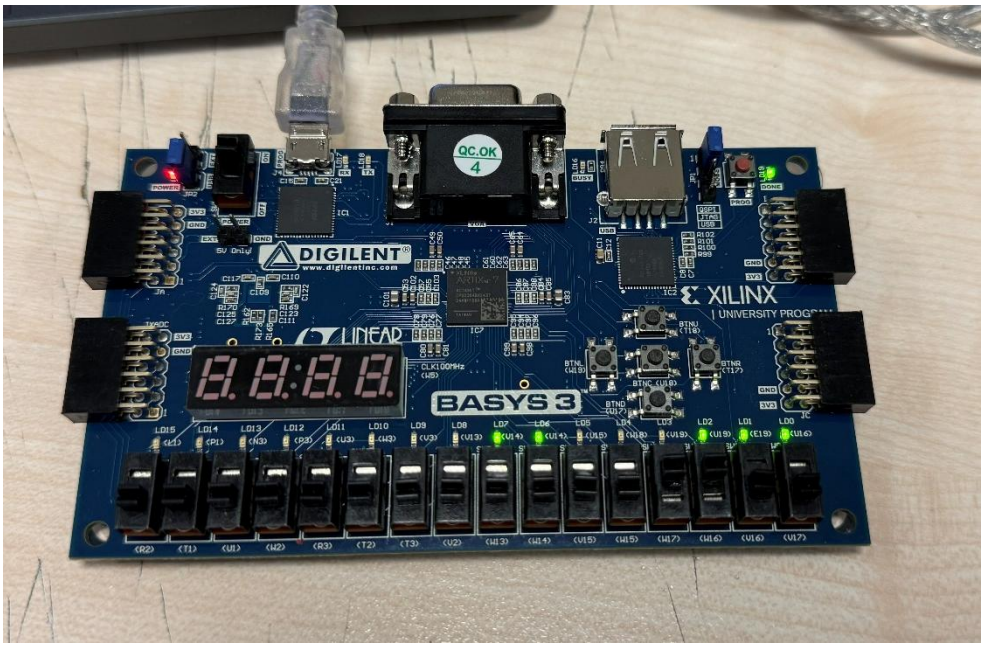

Figure 3.3 Input:00001100 Output:11000111

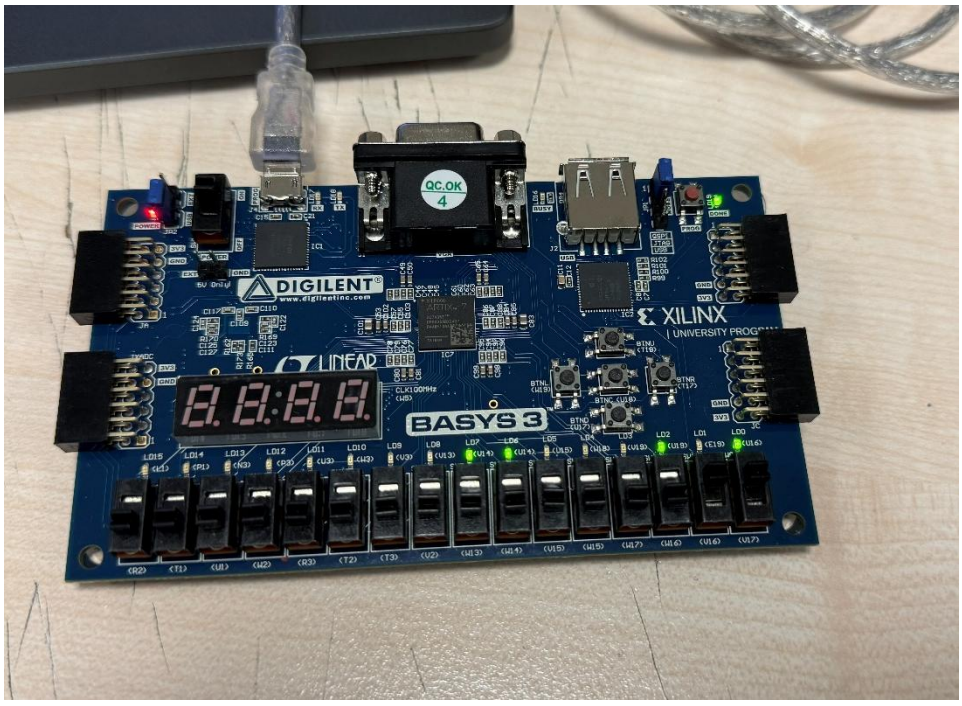

Figure 3.4 Input:00000011 Output:11000101

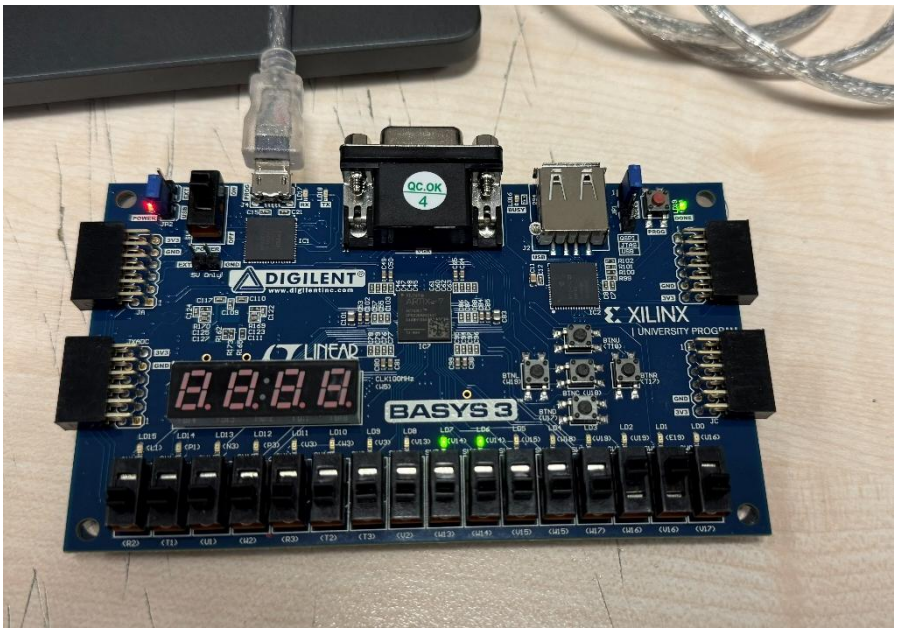

Figure 3.5 Input:00000110 Output:11000000

Task 4: For the testbench simulation, I wrote a simulation code and the waveform confirmed that all 256 possible input combinations for 8-bit inputs each with 10ns delay. Furthermore, I confirmed that my Basys3 Board behaved as expected by testing the results from above figures**.**

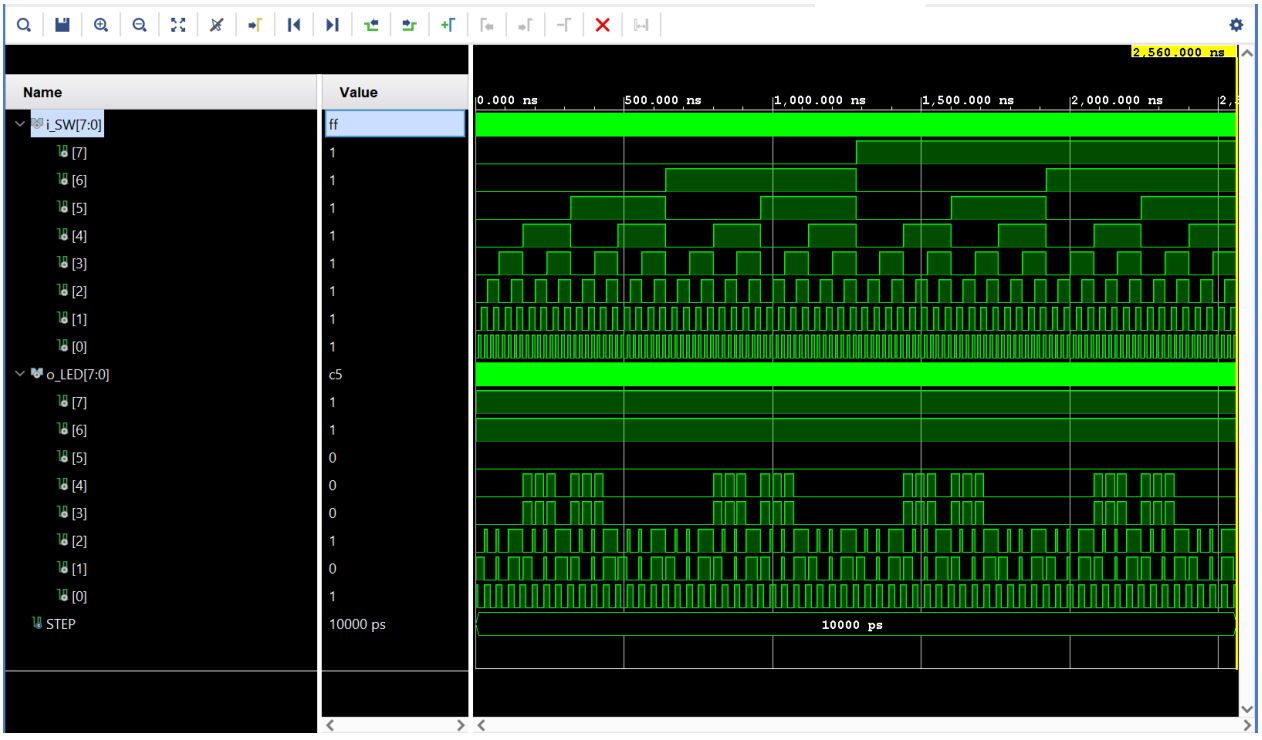

Figure 4 Simulation Result for all 256 combinations

Task 5: In this task, I analyzed the RTL, Synthesized and Implemented schematics in Vivado. All three confirmed the logical structure and connections were made correctly and implemented on the Basys3 FPGA. The first Figure 5.1 shows the RTL schematic while the Figure 5.2 shows Synthesized one. Also, in Figure 5.3 the Implemented schematic was shown and I observe that it is identical with the Figure 5.2 since this design is purely combinational circuit.

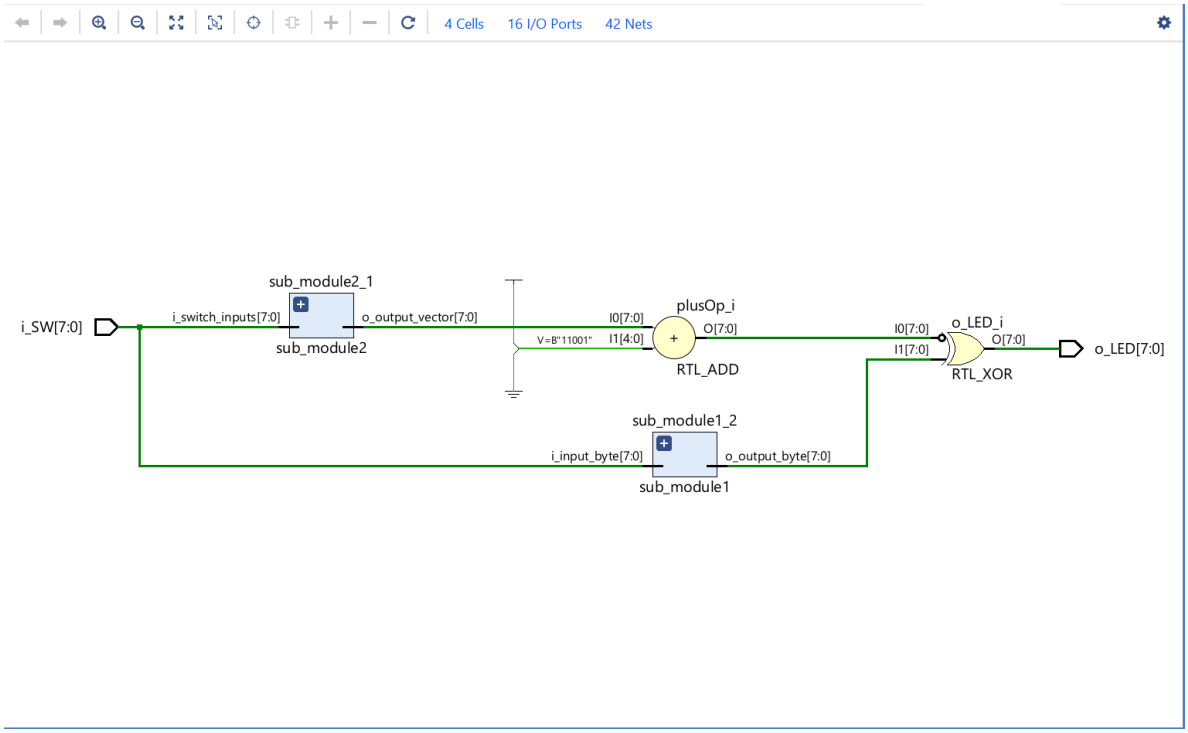

Figure 5.1 RTL Schematic

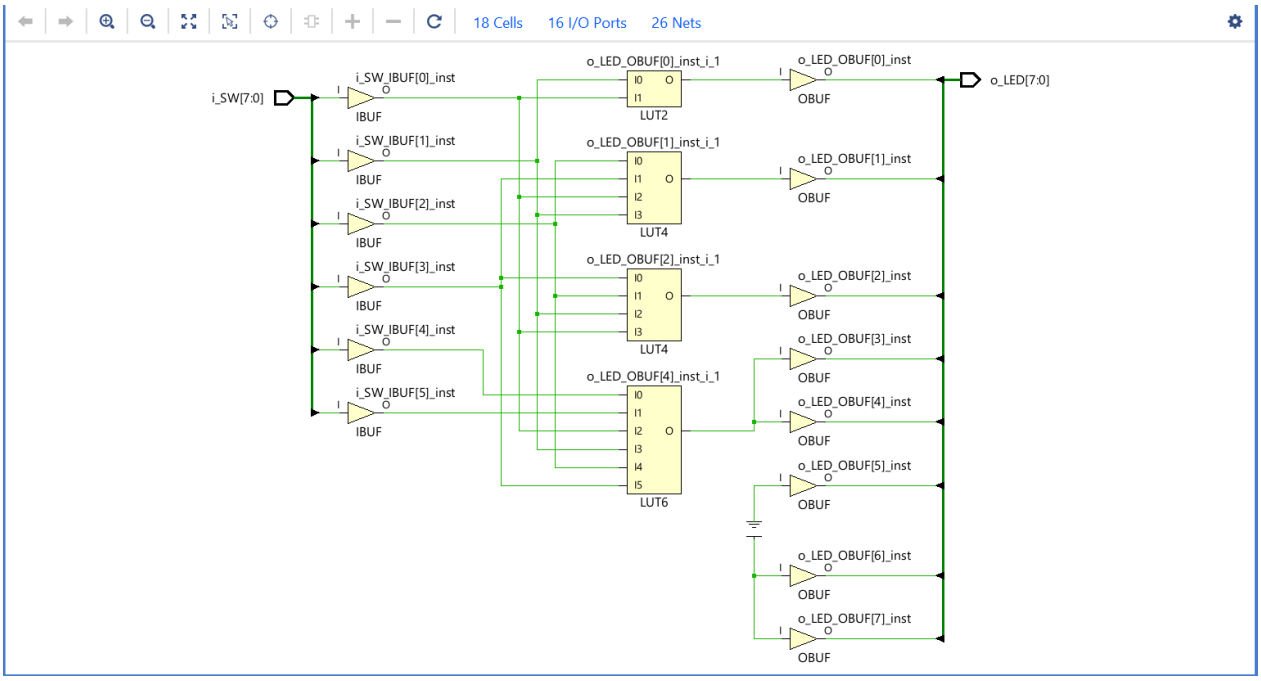

Figure 5.2 Synthesized Schematic

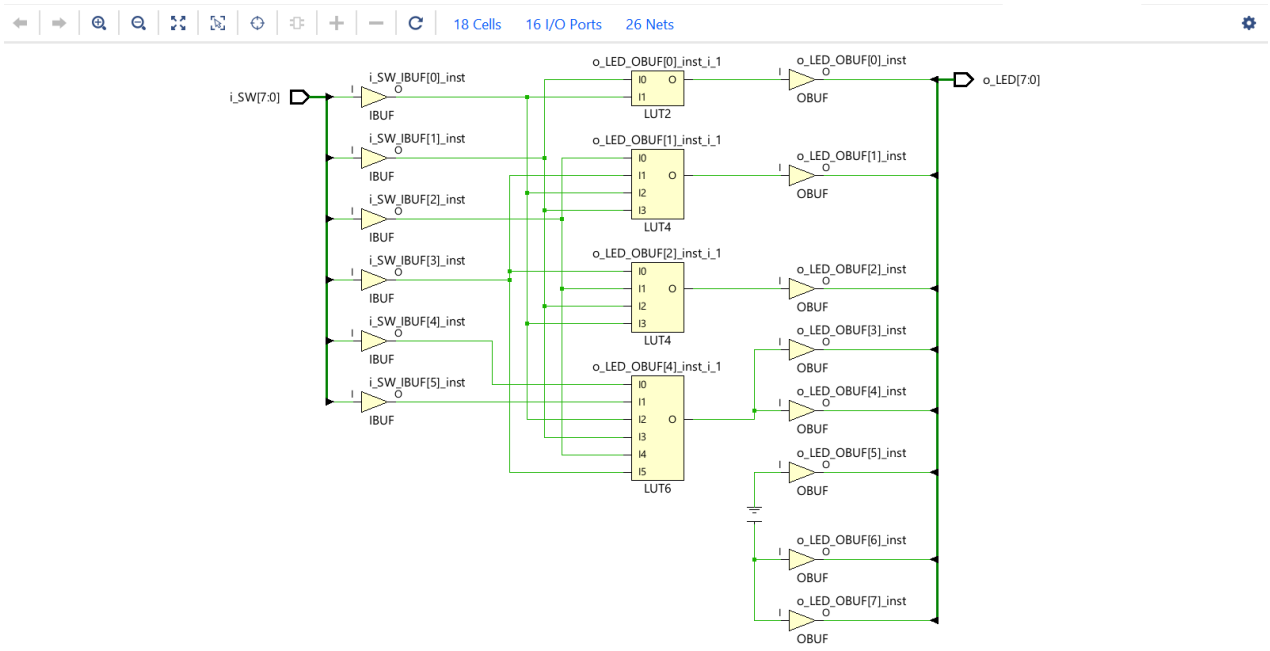

Figure 5.3 Implemented Schematic

Conclusion

In this experiment, I learned how to design, debug and implement a combinational logic circuit using VHDL and Vivado Design Suite. I fixed several typo, syntax and connection errors and wrote a testbench code to verify my design functionality. I implemented this design to a Basys3 FPGA board. The LEDs are responded correctly to the inputs that I adjusted by switches. Through this lab, I gained a practical experience with how to design a circuit and implement it to hardware. I also learned the importance of a constraint file and how to interpret RTL, Synthesized and Implemented schematics to visualize the design structure. I also understood how to create a testbench to simulate and verify the circuit behavior before implementation. Overall, this lab helped me to gain a deep understanding of FPGA design from coding to hardware implementation.

Appendices

Test Bench Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity tb_top_module is
end tb_top_module;
architecture sim of tb_top_module is
  component top_module is
    Port (
      i_SW  : in  STD_LOGIC_VECTOR (7 downto 0);
      o_LED : out STD_LOGIC_VECTOR (7 downto 0)
    );
  end component;
  signal i_SW   : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
  signal o_LED  : STD_LOGIC_VECTOR (7 downto 0);
begin
  UUT: top_module
    port map (
```

```
    i_SW  => i_SW,
    o_LED => o_LED
   );
  stim_proc : process
  begin
   for k in 0 to 255 loop
    i_SW <= std_logic_vector(to_unsigned(k, 8));
    wait for 10 ns;
   end loop;
   wait;
  end process;
end sim;
```

References

BASYS 3 – Vivado Tutorial.pdf

Basys3xdc.pdf

Test_Bench_Tutorial.pdf

https://nandland.com/introduction-to-vhdl-for-beginners-with-code-examples/

https://vhdlwhiz.com/port-map/