Emre Akdemir

22302606

EE102-02

EE-102 LAB 2 REPORT: INTRODUCTION TO VHDL

19/02/2024

Purpose

The aim of this lab was to learn basics of debugging and testing process through bugged VHDL code in advance, getting used to syntax of the language, adjusting constraint file accordingly, and embedding it into BASYS3 FPGA board.

Research Questions

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

To define inputs and outputs, two constructs called “entity” and “architecture” are used. The “entity” specifies the name and ports of the module. Each port has a name which is a identifier for a signal, a mode which is the direction in, out, inout and buffer, and a type such as STD_LOGIC, STD_LOGIC_VECTOR etc. The “architecture” describes the behaviour of the module, in other words, relates the inputs with outputs.

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

A module can be used in another module by using component or direct instantiation. Component instantiation method declares the component inside the architecture using PORT MAP. The PORT MAP is used to connect signals between instantiated and other modules. In direct entity instantiation, on the other hand, instead of declaring a component, this time the module is directly instantiated by using the entity name. There are entities called “Child” and “Parent” entity. The PORT MAP connects the input and output signals of the child module to the signals in the parent module. Also, there are top module and sub-modules. Top module(parent module) controls and connects the sub-modules(child modules) and it is the highest-level entity in the hierarchy of modules.

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

A constraint file is used to specify physical and timing constraints for the design. It determines which pins on the FPGA board should related to the signals in VHDL code. It also determines the timing constraints to operate in a specified clock frequency.

What is the purpose of writing a testbench?

A testbench code is used to simulate the logic before programming the FPGA. Verification of the design carries importance due to concerns of functionality, timing and correctness. It also allows debugging without a FPGA board.

Design Specifications

The given combinational logic circuit includes some of the logic gates depending on student's ID number. In this lab, because that my ID's digit is 8, NAND NAND OR are the gates according to the given table that used to create the logic circuit.

Inputs and Outputs

top_module:

i_SW: Input for the switches on the board from (0 to 7. switch)

o_LED: Output for the leds on the board from (0 to 7. led)

sub_module_1:

i_input_byte: Input for the switches on the board from (0 to 7. switch)

o_output_byte: Output of the module in the form of logic vector

sub_module_2:

i_switch_inputs: Input for the switches on the board from (0 to 7. switch)

o_output_vector: Output of the module in the form of logic vector

There is two sub module and one top module was designed for this circuit. Sub module 1, takes logic vector and generates the first digit of the output by doing NAND operation on first two byte, then applies the NAND operation for the third and fourth byte of the input and generates as the second byte of the output. Finally, applies OR gate to fourth and fifth bytes and generatas as the third byte of the output. Fourth, fifth and sixth bytes are assigned as "010" and remaining bytes assigned as "0". Sub module 2, computes the compliment of the input and in top module, the output coming from sub module 2 turns into decimal numbers and 25 is added and assigned to s_output_3. As the final o_Led output, XOR gate is applied bitwise to the compliment of binary version of s_output_3 and the output of sub_module_2.

Methodology

1) First, I altered the code as expected, I changed the logic gates(figure) corresponding to the gates given according to the student ID numbers.
2) The given VHDL code had 6 bugs, when I have run the code, I encountered with some of the errors (Figure 11). For each bug, I fixed the code. The errors were as in figure. (Figure 10.1, 10.2, 10.3, 10.4, 10.5, 10.6).
3) After fixing the code, I run the synthesis, implementation respectively. After these, I generated the bitstream and programmed the device.
4) By using the Test Bench Tutorial.pdf, I used the given template and edited the template to suit for the code. I used “for loop” to test the code by creating 8 bit binary numbers starting from 0 to 255. (Figure 12). After running the simulation, the 256 combinations from 8 inputs in the waveform could be seen in (Figure 9).
5) The simulation has been compared to the implemented design on the FPGA board, 5 random input (Figure 1, Figure 1.1, ... Figure 5, Figure 5.1) and corresponding outputs from the board and simulation were the same.
6) I obtained the schematics (Figure 6, Figure 7, Figure 8) from “Synthesized Design”, “Implemented Design” and RTL schematics. RTL Schematic represents the high-level abstraction of the design. It shows the inputs, outputs, sub modules, logic operations, arithmetic units and so on. This schematic does not include FPGA-specific resources such as LUTs or flip-flops when compared to other design schematics, but it helps to verification of the functionality of the code. The Synthesis schematic transforms the design from gate representation to the mapping logic of FPGA resources like LUTs, Flip-Flops and so on. Without hardware implementation, the synthesis schematic consists of some of the basic logic gates, FPGA-specific elements such as LUTs, BRAMs, DSPs. On the other hand, implemented design schematic shows routing delays and critical path analysis. The implemented design schematic and synthesis schematic looks the same (Figure 7, Figure 8).
7) The correlation between those schematics and simulation results are about how design is changed through each level of RTL, Synthesis and Implementation and how simulations verify the functionality and timing at each stage. RTL simulation verifies the intended functionality without considering hardware implementation details. Synthesis simulation checks still functionality but do not check routing or timing delays. The implementation simulation considers real hardware constraints and gives the result of time delays. It can be summarized as the process of foreseen of problems occurring while moving from abstract design to the physical implementation.

Results

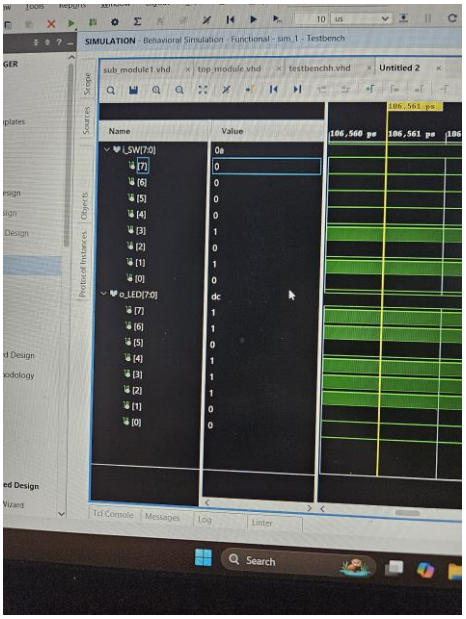

Figure 1.1 - Input - “010100001

Output - “00111011”

Figure 1.2 - On Basys3

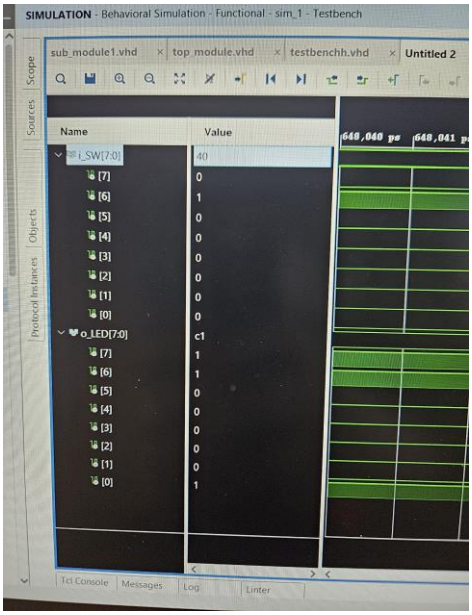

Figure 2.1 - Input - “00000010”

Output - “10000011”

Figure 2.2 - On Basys3

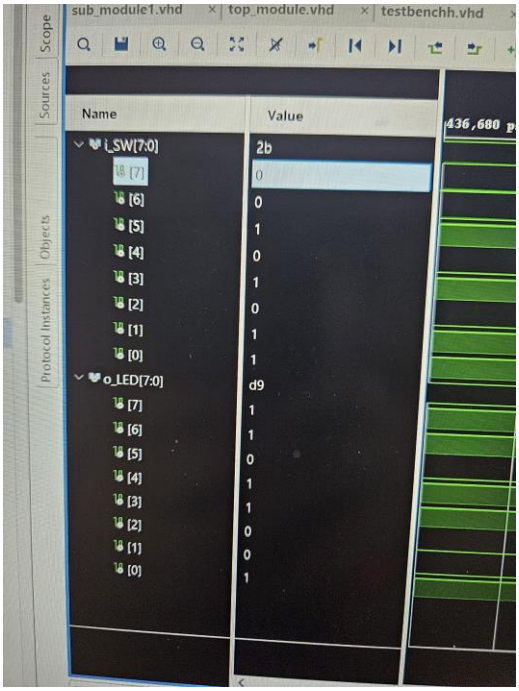

Figure 3.1 - Input - “11010100”

Output - “10011011”

Figure 3.2 - On Basys3

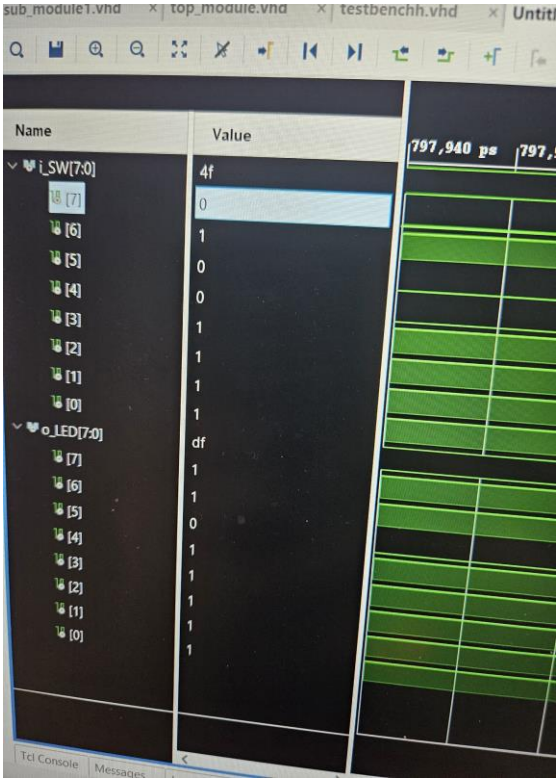

Figure 4.1 - Input - “11110010”

Output - “1111011”

Figure 4.2 - On Basys3

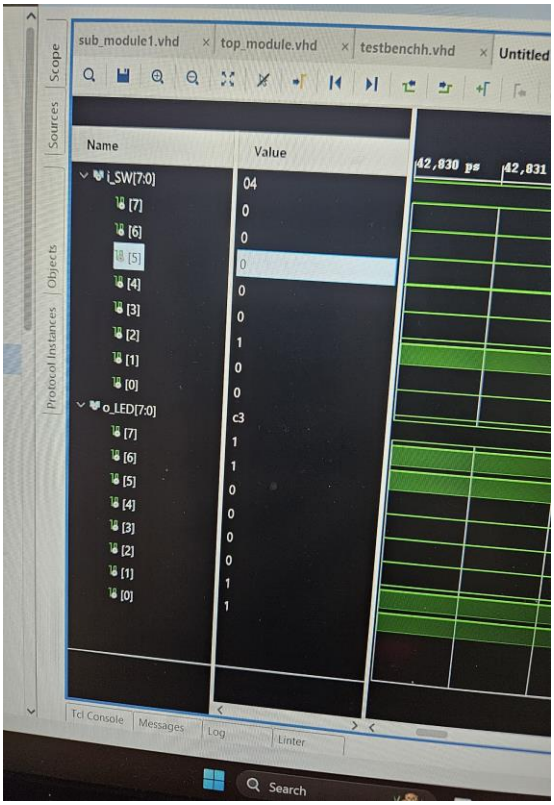

Figure 5.1 - Input - “00100000”

Output - “11000011”

Figure 5.2 - On Basys3

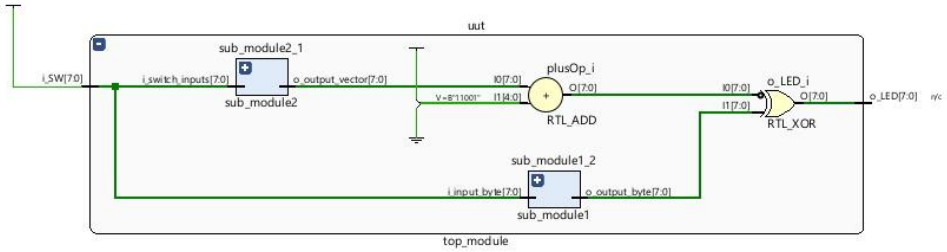

Figure 6 - RTL Schematics

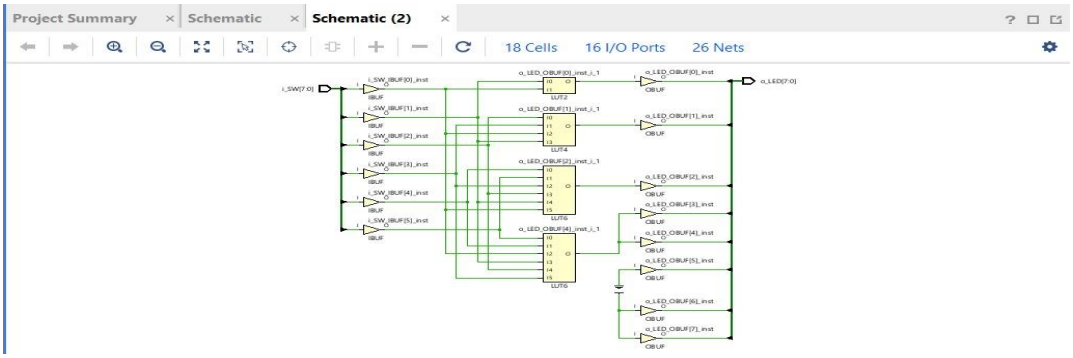

Figure 7 – Synthesis Schematics

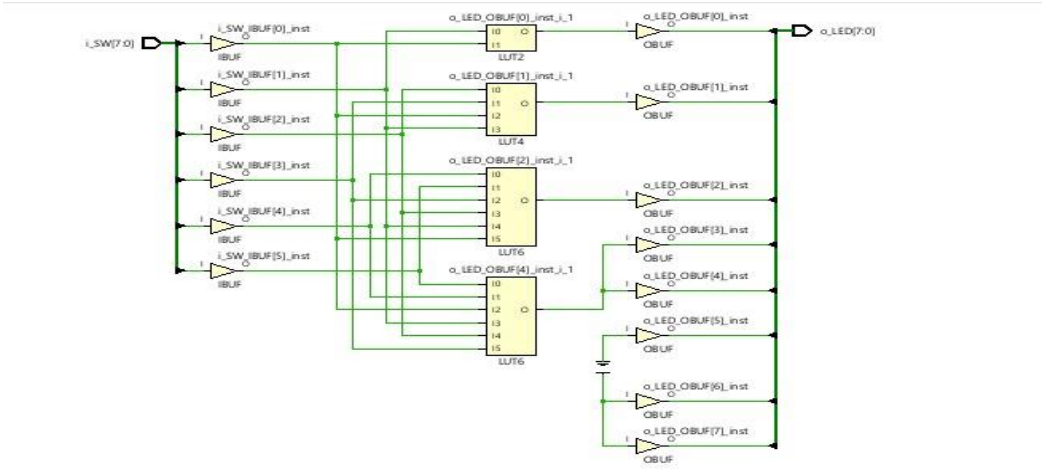

Figure 8 – Implemented Design Schematics

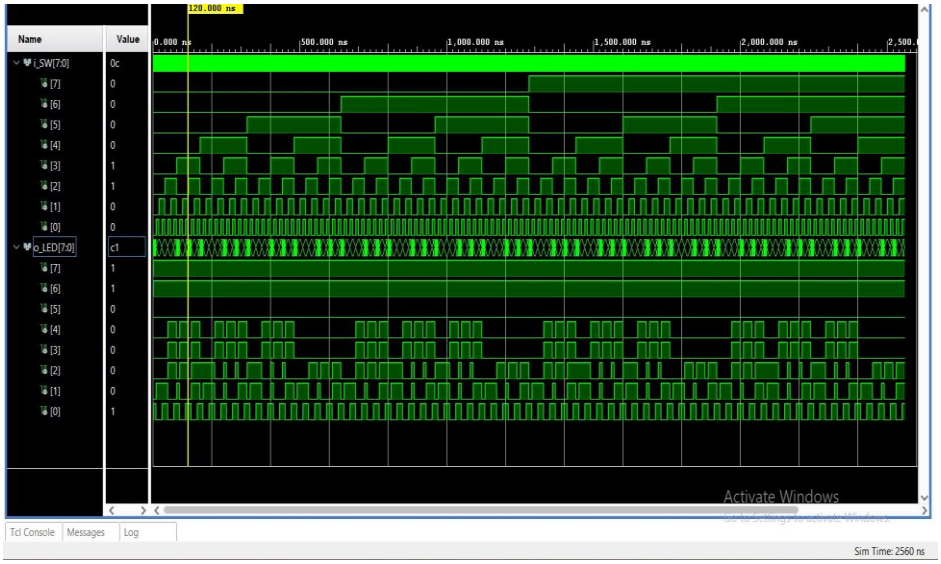

Figure 9 – Behavioral Simulation

Conclusion

The aim was to learn the basics of debugging process, handling errors in VHDL, getting used to the syntax, learning how to write testbenches, learning about the RTC Analysis, Synthesis and Implementation process, and analyzing the functionality of the code by comparing the result from behavioral simulation and FPGA board. This lab was crucial for learning about VHDL and its implementations. Although I encountered a lot of problems such as false project device model choice, random spawn errors, I asked help from TA's and fixed them by their instructions. As a result, the lab was successful.

Appendices

```
set_property PACKAGE_PIN V14 [get_ports {o_LED[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[0]}]
set_property PACKAGE_PIN U16 [get_ports {o_LED[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {o_LED[7]}]
```

Figure 10.1 - First Bug (Pin modes should be switched between led 0 and led 7.)

```
set_property PACKAGE_PIN U19 [get_ports {o_LED[2]]
```

Figure 10.2 - Second Bug("}" is missing before "]" .)

```
begin

    sub_module1_2 : sub_module1
        port map (
            i_input_byte  => i_SW,
            o_output_byte => s_ouput_1
        );
```

Figure 10.3 - Third Bug ("t" is missing for s_output_1.)

```
sub_module2_1 : sub_module2
    port map (s_output_2, i_SW);
```

Figure 10.4 - Fourth Bug (Wrong declarition for the sub_module2.)

```
sub_module2_1 : sub_module2
    port map (i_switch_inputs => i_SW,
              o_output_vector => s_output_2
    );
```

Figure 10.5 Correction of the Fourth Bug

```
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;
```

Figure 10.6 - Fifth Bug (Entity name should be sub_module2 not “sub_module2_the_beast”.)

Figure 10.7 - Sixth Bug (Project device should adjusted to the above device.)

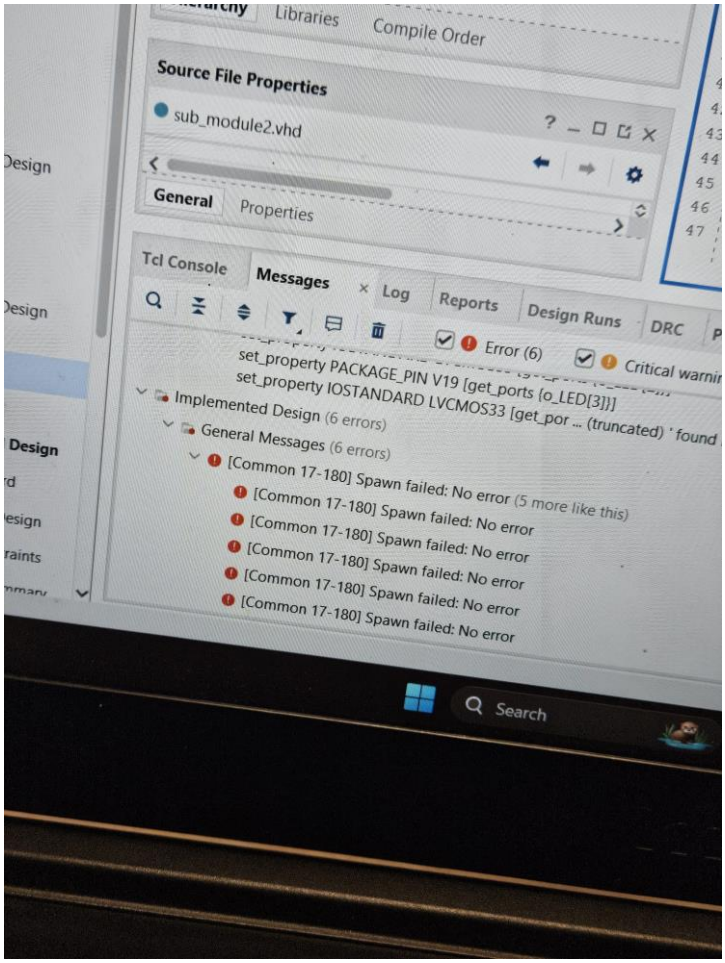

Figure 11 – Some of the errors occured while running the program.

```
20      UUT : top_module port map(i_SW ,o_LED);
21
22      STIM_PROCESS : process begin
23      for k in 0 to 255 loop
24       i_SW <= std_logic_vector(to_unsigned(k, 8));
25       wait for 10 ns;
26      end loop;
27      assert false report "FINISHED" severity failure;
28      end process;
```

Figure 12 – for loop written for the testbench.

VHDL Code

top_module:

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_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_module_1:

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

end Structural_sub1;

sub_module_2:

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;
```

Testbench:

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

entity Testbenchh is end Testbenchh;

architecture Behavioral of testBenchh 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) := "00000000"; signal o_LED : std_logic_vector(7 downto 0); begin

```
UUT : top_module port map(i_SW ,o_LED);

STIM_PROCESS : process begin
for k in 0 to 255 loop
 i_SW <= std_logic_vector(to_unsigned(k, 8));
 wait for 10 ns;
end loop;
assert false report "FINISHED" severity failure;
end process;


end Behavioral;
```