**ELECTRICAL & COMPUTER ENGINEERING**

School of Engineering

**EGRE 365 – Digital Systems**

**Homework No. 2**

**Name: Luis Barquero**

**Major: Computer Engineering**

**Due Date: 10/02/17**

**Honor Pledge:** *I have neither given nor received any unauthorized help on this lab. Signed:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Luis Barquero\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Since both subprograms will use the same inputs to find both the maximum and minimum values, alongside their indices, both subprograms were combined into a single VHDL model for a better implementation.

1. **Write a VHDL subprogram that implements the function of finding the maximum of 4 inputs.**

The first step in implementing the subprogram was to first define the port, which includes 4 standard logic vectors size 8 (7 down to 0) for each one of the four inputs. From there, the initialization of the maximum and minimum index was created, both of standard logic type of size 2 (1 down to 0).

Next, a procedure labeled finding\_max was created that takes in inputs 0-3, finds the maximum value, and outputs the index of that maximum value.

The way the procedure works is that it checks to see if current input is greater than the rest of the inputs. For example, it first checks if input0 is greater than input1, input2, and input3. If so, the input0 is considered the max value, and it outputs the index of input0. The same method is used for the rest of the inputs.

At the same time, the procedure also outputs the value contained in the index, or in other words, it outputs the maximum number.

See Appendix A for the VHDL Model code.

1. **Write a VHDL subprogram that implements the function of finding the minimum of 4 inputs.**

Given that all inputs and outputs have already been initialized, then in order to find the minimum value, and a procedure labeled finding\_min is created that takes in inputs 0-3, finds the minimum value, and outputs the index of the minimum value.

This procedure works in a similar way as the finding\_max procedure, except instead of checking if the current input is greater than the rest of the inputs, it checks if the current input is less than the rest of the inputs. If the condition is met, then the procedure outputs the index of the minimum value, alongside the value contained in the index.

See Appendix A for the VHDL Mode code.

1. **Write a VHDL entity and architecture that implements a comparator that finds out the maximum number and minimum number in 4 8-bit inputs.**

For this part, there is a procedure call in the architecture for both the max and min finders that sends in the inputs, determines the max/min with their indices, and outputs the results.

1. **Write a test bench to test your code for a minimum of 10 test cases.**

In order to properly test out the model, the following 10 cases were constructed, as illustrated by table 1, where it shows all inputs alongside their max/min index and their max/min value.

***Table 1 – Table 1 shows all 10 cases with their corresponding max/min index/value.***

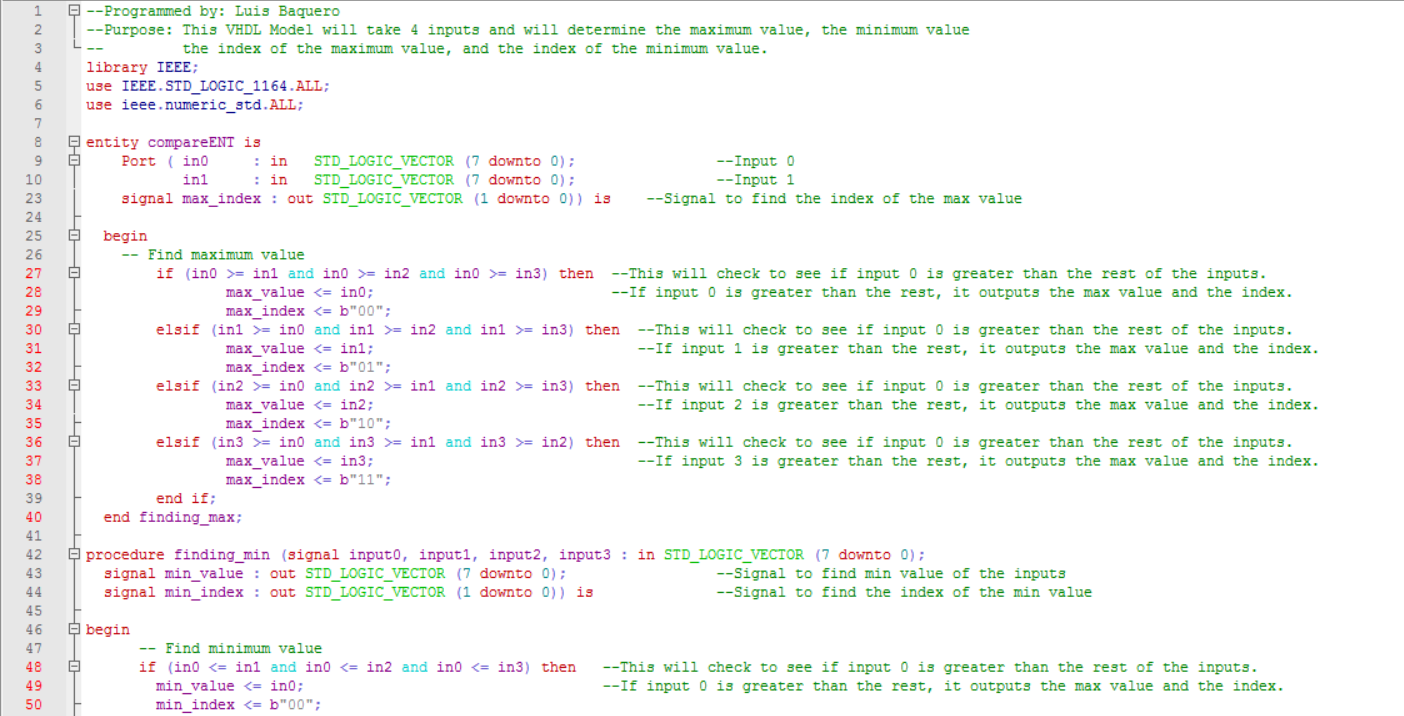
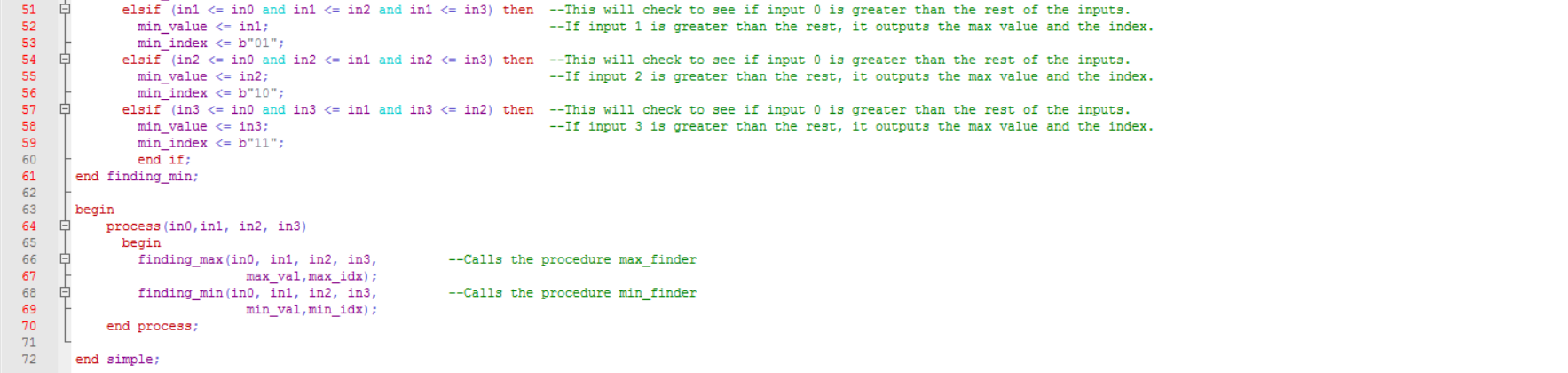
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Case** | **Input0** | **Input1** | **Input2** | **Input3** | **Max\_Index** | **Min\_Index** | **Max\_Value** | **Min\_Value** |
| **1** | 0010\_0010 (34) | 0011\_0011 (51) | 0000\_0000 (0) | 1111\_1111 (255) | 11 | 10 | 1111\_1111 (255) | 0000\_0000 (0) |
| **2** | 0000\_0000 (0) | 0000\_0000 (0) | 1111\_1111 (255) | 1111\_1111 (255) | 10 | 00 | 1111\_1111 (255) | 0000\_0000 (0) |
| **3** | 0000\_0001 (1) | 0001\_1001 (25) | 0100\_0000 (64) | 0100\_0000 (64) | 10 | 00 | 0100\_0000 (64) | 0000\_0001 (1) |
| **4** | 0010\_0100 (36) | 0010\_0100 (36) | 0100\_0000 (64) | 0110\_0101 (101) | 11 | 00 | 0110\_0101 (101) | 0010\_0100 (36) |
| **5** | 0000\_0011 (3) | 0000\_1100 (12) | 0000\_0110 (6) | 0000\_1001 (9) | 01 | 00 | 0000\_1100 (12) | 0000\_0011 (3) |
| **6** | 0111\_1010 (122) | 0111\_1000 (120) | 0111\_1011 (123) | 0111\_1001 (121) | 10 | 01 | 0111\_1011 (123) | 0111\_1000 (120) |
| **7** | 1001\_0010 (146) | 1001\_0010 (146) | 1001\_0010 (146) | 1001\_0010 (146) | 00 | 00 | 1001\_0010 (146) | 1001\_0010 (146) |
| **8** | 1000\_1000 (136) | 1000\_1000 (136) | 1000\_1000 (136) | 1000\_1000 (136) | 00 | 00 | 1000\_1000 (136) | 1000\_1000 (136) |
| **9** | 0010\_0100 (36) | 0010\_0100 (36) | 0100\_1000 (72) | 0100\_1000 (72) | 10 | 00 | 0100\_1000 (72) | 0010\_0100 (36) |
| **10** | 0000\_0010 (2) | 0000\_0101 (5) | 0000\_0001 (1) | 0000\_0111 (7) | 11 | 10 | 0000\_0111 (7) | 0000\_0001 (1) |

Appendix B contains the simulation results from the testbench.

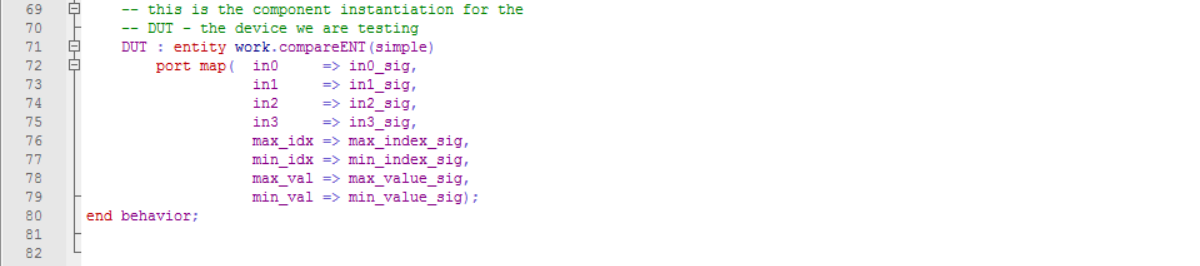
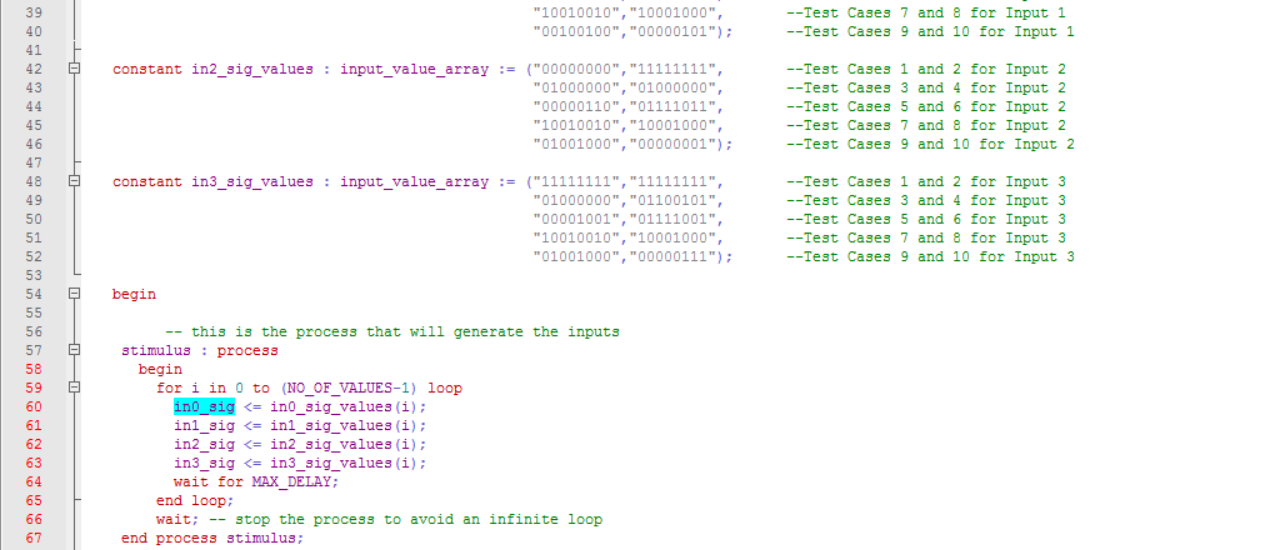
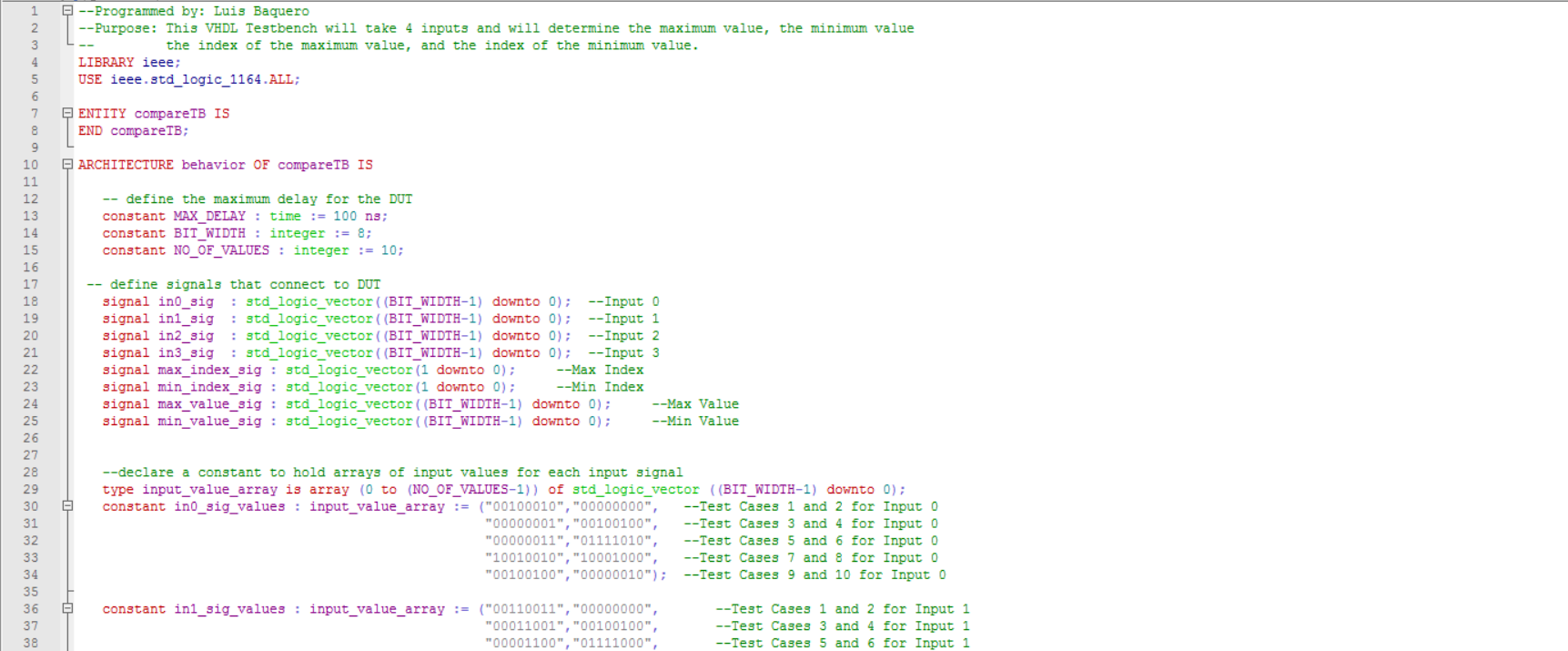
Appendix A

VHDL Code

***Figure 1 – Figure 1 shows the code used for the VHDL Model***



***Figure 2 – Figure 2 shows the code used for the VHDL Testbench***

******

Appendix B

VHDL Simulations

