Lab 2 Comparators

CSC 343 Fall 2017

September, 18 2017

Jeter Gutierrez

TABLE OF CONTENTS

1. Objective

In this lab we will be implementing a 1 bit, 2 bit and 8 bit comparator that will be capable of determining whether two 1 bit, 2 bit or 8 bit words are equal to each other. The purpose of implementing these circuits is to work better with memory storage and learn how memory works. With these components we would be able to test if our components for storing memory in future labs are storing the appropriate values. We will be designing these circuits in VHDL using

Quartus Prime Software. We will be testing the circuits using modelsim waveform simulation. Another method that will be used to test the circuits is testbench with predefined values for each of the n-bit words. After we have completed designing the circuits we will run the circuits in simulation to assert that they are working as expected.

The circuits we will be designing in this lab are:

1. 1-Bit Comparator.
2. 1-Bit Comparator Test.
3. 1-Bit Comparator With Not Equal Output.
4. 2-Bit Comparator.
5. 2-Bit Comparator Test.
6. 2-Bit Comparator Using 1-Bit Comparator.
7. 2-Bit Comparator Using 1-Bit Comparator Test.
8. 8-Bit Comparator Using 1-Bit Comparator.
9. 8-Bit Comparator Using 1-Bit Comparator Test.

2. 1-Bit Comparator.

2.1 Functionality and specifications for 1-Bit Comparator.

The purpose of the 1-Bit comparator is to compare whether two one bits of information are equal to each other. This will be used as a component in order to compare words larger than 1-Bit of information later in this lab. It will take as an input two one bits of information and compare them to each other, if they are the same a signal will be sent to eq which is the variable storing the value of equal between the two bits of information. The inputs will be stored in I0 and I1 which are short for input 0 and input 1. It is capable of comparing two 1 bit signals, if they are the same eq outputs 1, if they are different eq outputs 0.

```
Library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
entity GUTIERREZ_1BIT_COMPARATOR is--Jeter Gutierrez Due September, 18 2017
port ( I0, I1 : in std_logic;--Jeter Gutierrez Due September, 18 2017
Eq: out std_logic);--Jeter Gutierrez Due September, 18 2017
end GUTIERREZ_1BIT_COMPARATOR ;--Jeter Gutierrez Due September, 18 2017
architecture arch of GUTIERREZ_1BIT_COMPARATOR is--Jeter Gutierrez Due September, 18 2017
signal p0, p1 : std_logic;--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
EQ <= p0 or p1;--Jeter Gutierrez Due September, 18 2017
p0 <= (not I0) and (not I1);--Jeter Gutierrez Due September, 18 2017
p1 <= I0 and I1;--Jeter Gutierrez Due September, 18 2017
end arch;--Jeter Gutierrez Due September, 18 2017
```

Figure 1: VHDL Code for 1-Bit comparator.

2.2 Simulation for 1-Bit Comparator.

In this simulation I0 and I1 will be given different values, either 1 or 0, they will then be compared to each other in order to determine whether they are equal the result of that comparison will be sent to Eq.

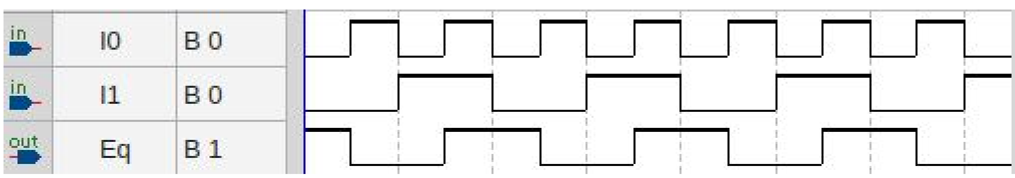

Figure 2: Vector Waveform Simulation for 1-Bit comparator. As we can see from the simulation, the values of I0 and I1 are being compared correctly, Eq always has the correct result based on each comparison every single time. This means that we were able to successfully design a 1-Bit comparator in VHDL.

3. 1-Bit Comparator Test.

3.1 Functionality and specifications for 1-Bit Comparator Test.

The purpose of this circuit is to create a test bench in order to simulate our 1-Bit comparator. We already simulated our 1-Bit comparator in Vector waveform but it is more professional and exact to use a test bench to test the correctness of our circuits because we have more control over the simulation this way. The test bench imitates a physical lab bench making our simulation cleaner and more organized.

```
library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
entity GUTIERREZ_TEST_1BIT_COMPARATOR is--Jeter Gutierrez Due September, 1
end GUTIERREZ_TEST_1BIT_COMPARATOR ;--Jeter Gutierrez Due September, 18 201
architecture arch_test of GUTIERREZ_TEST_1BIT_COMPARATOR is--Jeter Gutier
component GUTIERREZ_1BIT_COMPARATOR --Jeter Gutierrez Due September, 18 201
port ( I0, I1 : in std_logic;--Jeter Gutierrez Due September, 18 2017
Eq: out std_logic );--Jeter Gutierrez Due September, 18 2017
end component;--Jeter Gutierrez Due September, 18 2017
signal p1, p0, pout : std_logic;--Jeter Gutierrez Due September, 18 2017
signal error: std_logic := '0';--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
uut: GUTIERREZ_1BIT_COMPARATOR port map (I0 => p0, I1 => p1, Eq => pout);
process--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
p0 <= '1';--Jeter Gutierrez Due September, 18 2017
p1 <= '0';--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
 end if;--Jeter Gutierrez Due September, 18 2017
 wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
 p0 <= '1';--Jeter Gutierrez Due September, 18 2017
 p1 <= '1';--Jeter Gutierrez Due September, 18 2017
 wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
 if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
 end if;--Jeter Gutierrez Due September, 18 2017
 wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
 p0 <= '0';--Jeter Gutierrez Due September, 18 2017
 p1 <= '1';--Jeter Gutierrez Due September, 18 2017
 wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
 if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
 end if;--Jeter Gutierrez Due September, 18 2017
 wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
 p0 <= '0';--Jeter Gutierrez Due September, 18 2017
 p1 <= '0';--Jeter Gutierrez Due September, 18 2017
 wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
 if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
 end if;--Jeter Gutierrez Due September, 18 2017
 wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
 if (error = '0') then--Jeter Gutierrez Due September, 18 2017
```

```
report "No errors detected. Simulation successful"  severity-
 failure;--Jeter Gutierrez Due September, 18 2017
else--Jeter Gutierrez Due September, 18 2017
report "Error detected" severity failure;--Jeter Gutierrez D
 end if;--Jeter Gutierrez Due September, 18 2017
 end process;--Jeter Gutierrez Due September, 18 2017
 end arch_test;--Jeter Gutierrez Due September, 18 2017
```

Figure 3: VHDL code for the test bench for testing 1-Bit Comparator. In this design modelsim will be used in order to simulate what is written in the testing code. It will be used to test the values for I0 and I1 and determine whether for every possible case of comparing one bit of information after a certain period of time or after testing another state it will continue to be correct. The reason we do this is to have more control over our testing for correctness of our 1-Bit comparator circuit.

3.2 Simulation for 1-Bit Comparator Test.

In this simulation we will be using modelsim to run our test bench file for the 1-Bit comparator, if we get an output in the terminal of modelsim that says “Simulation successful” then we know we have designed a correct 1-Bit comparator. The difference in this state is that we already wrote what values we want to test for and modelsim will create those values for us instead of us having to use the cursor to select the values using waveform.

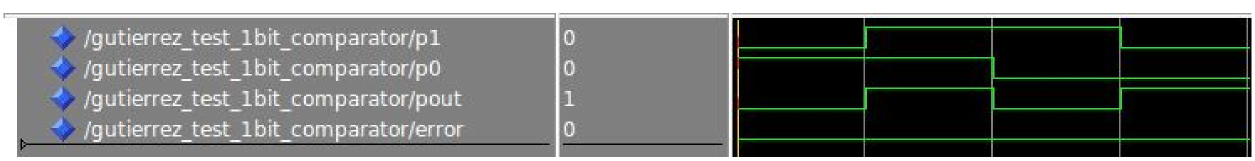

Figure 4: Vector waveform simulation for test bench of 1-Bit comparator. As we can see the error value is constantly returning 0 throughout the entire simulation, that means that our design was correct and that we did not make any mistakes, we were able to successfully design an 1-Bit

comparator. We also got a message in our terminal saying “Simulation successful” which means our design is correct.

4. 1-Bit Comparator With Not Equal Output.

4.1 Functionality and specifications for 1-Bit Comparator With Not Equal Output.

The purpose of the 1-Bit comparator with the Not Equal output is to compare whether two one bits of information are equal to each other and to show that we can also detect that values are different. If the 1-Bit signals are the same a signal will be sent to eq which is the variable storing the value of equal between the two bits of information. The inputs will be stored in I0 and I1 which are short for input 0 and input 1. It is capable of comparing two 1 bit signals, if they are the same eq outputs 1 and notEq outputs 0, if they are different eq outputs 0 and notEq outputs 1.

```
Library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September,
entity GUTIERREZ_1BIT_COMPARATOR_WITH_NOTEQ  is--Jeter Gutierr
port ( I0, I1 : in std_logic;--Jeter Gutierrez Due September,
Eq, notEq: out std_logic);--Jeter Gutierrez Due September, 18
end GUTIERREZ_1BIT_COMPARATOR_WITH_NOTEQ ;--Jeter Gutierrez Du
architecture arch of GUTIERREZ_1BIT_COMPARATOR_WITH_NOTEQ  is
signal p0, p1 : std_logic;--Jeter Gutierrez Due September, 18
begin--Jeter Gutierrez Due September, 18 2017
EQ <= p0 or p1;--Jeter Gutierrez Due September, 18 2017
p0 <= (not I0) and (not I1);--Jeter Gutierrez Due September,
p1 <= I0 and I1;--Jeter Gutierrez Due September, 18 2017
notEq <= (not(p0 or p1));--Jeter Gutierrez Due September, 18
end arch;--Jeter Gutierrez Due September, 18 2017
```

Figure 5: VHDL code for 1-Bit comparator with not equal output.

4.2 Simulation for 1-Bit Comparator With Not Equal Output.

In this simulation I0 and I1 will be given different values, either 1 or 0, they will then be compared to each other in order to determine whether they are equal the result of that comparison will be sent to Eq if they are equal Eq will have a value of 1, but if they are not equal then notEq is expected to have a value of 1.

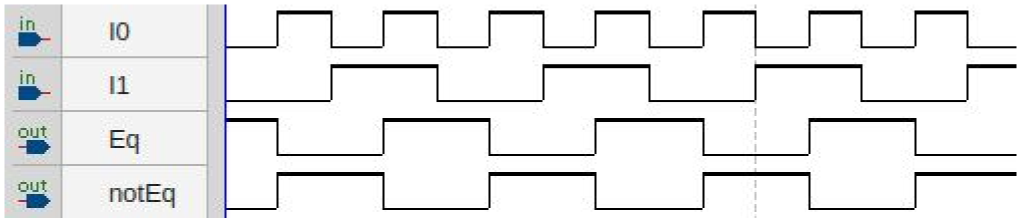

Figure 6: Vector Waveform Simulation for 1-Bit comparator with not equal output. As we can see from the simulation, the values of I0 and I1 are being compared correctly, Eq always has the correct result based on each comparison every single time. For example in the very first moment I0 is 0 and I1 is also 0, Eq has an expected result of 1 and notEq has an expected result of 0, and at the end we can see when I0 is 0 and I1 is 1 that Eq has the expected value of 0 while notEq has the expected value of 1. This means that we were able to successfully design a 1-Bit comparator with a not equal output in VHDL.

5. 2-Bit Comparator.

5.1 Functionality and specifications for 2-Bit Comparator.

The purpose of the 2-Bit comparator is to compare whether two 2-bit words are equal to each other. It will take as an input two 2-bit words and compare them to each other 1-Bit at a time, if they are the same a signal will be sent to aeqb which is the variable storing the value of equal between the two bits of information and represents “are equal bits”. The inputs will be stored in a and b which are both 2-bit vectors.

```
library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 20
entity GUTIERREZ_2BIT_COMPARATOR is--Jeter Gutierrez Due September
port ( a, b--Jeter Gutierrez Due September, 18 2017
: in std_logic_vector(1 downto 0);--Jeter Gutierrez Due September,
aeqb--Jeter Gutierrez Due September, 18 2017
: out std_logic);--Jeter Gutierrez Due September, 18 2017
end GUTIERREZ_2BIT_COMPARATOR ;--Jeter Gutierrez Due September, 18
architecture arch of GUTIERREZ_2BIT_COMPARATOR is--Jeter Gutierrez
signal p0, p1, p2, p3 : std_logic;--Jeter Gutierrez Due September,
begin--Jeter Gutierrez Due September, 18 2017
aeqb <= p0 or p1 or p2 or p3;--Jeter Gutierrez Due September, 18 2
p0 <=  (a(1) and b(1)) and (a(0)and b(0));--Jeter Gutierrez Due Se
p1 <=  (a(1)and b(1)) and ((not a(0)) and (not b(0)));--Jeter Gut
p2 <=  ((not a(1)) and (not b(1))) and (a(0) and b(0));--Jeter Gu
p3 <=  ((not a(1)) and (not b(1))) and ((not a(0))and(not b(0)));
end arch;--Jeter Gutierrez Due September, 18 2017
```

Figure 7: VHDL code for 2-Bit Comparator.

5.2 Simulation for 2-Bit Comparator.

In this simulation a and b are 2-Bit words that will each be given different values from 00 to 11 in different combinations in order to determine whether our design for the 2-Bit comparator that tells us if two 2-Bit words are identical is working correctly, the output of the two 2-Bit words being equal will be sent to aeqb.

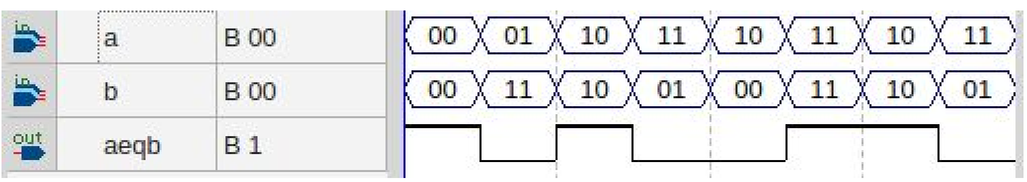

Figure 8: Vector waveform simulation for 2-Bit comparator, as we can see in this simulation when the 2-Bit words are equal we get an output of 1 to aeqb, when they are not equal aeqb is 0 this means that we were correct. Our expectations were correct this means we were able to successfully design a 2-Bit comparator.

6. 2-Bit Comparator Test.

6.1 Functionality and specifications for 2-Bit Comparator Test.

The purpose of this circuit is to create a test bench in order to simulate our 2-Bit comparator. We already simulated our 2-Bit comparator in Vector waveform but it is more professional and exact to use a test bench to test the correctness of our circuits because we have more control over the simulation this way. The test bench imitates a physical lab bench making our simulation cleaner and more organized.

```
library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
entity GUTIERREZ_TEST_2BIT_COMPARATOR  is--Jeter Gutierrez Due September, 1
end GUTIERREZ_TEST_2BIT_COMPARATOR ;--Jeter Gutierrez Due September, 18 201
architecture arch_test of GUTIERREZ_TEST_2BIT_COMPARATOR  is--Jeter Gutierr
component GUTIERREZ_2BIT_COMPARATOR --Jeter Gutierrez Due September, 18 201
port ( a, b: in std_logic_vector(1 downto 0);--Jeter Gutierrez Due Septemb
aeqb: out std_logic);--Jeter Gutierrez Due September, 18 2017
end component;--Jeter Gutierrez Due September, 18 2017
signal p1, p0: std_logic_vector(1 downto 0);--Jeter Gutierrez Due Septembe
signal pout :std_logic;--Jeter Gutierrez Due September, 18 2017
signal error :std_logic := '0';--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
uut: GUTIERREZ_2BIT_COMPARATOR  port map(a => p0, b=> p1, aeqb => pout);--
process--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
p0 <= "00";--Jeter Gutierrez Due September, 18 2017
p1 <= "00";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "01";--Jeter Gutierrez Due September, 18 2017
p1 <= "00";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "01";--Jeter Gutierrez Due September, 18 2017
p1 <= "11";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11";--Jeter Gutierrez Due September, 18 2017
p1 <= "00";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11";--Jeter Gutierrez Due September, 18 2017
```

```
p1 <= "11";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "10";--Jeter Gutierrez Due September, 18 2017
p1 <= "11";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "10";--Jeter Gutierrez Due September, 18 2017
p1 <= "10";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11";--Jeter Gutierrez Due September, 18 2017
p1 <= "01";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
if (error = '0') then--Jeter Gutierrez Due September, 18 201
report "No errors detected. Simulation successful"  severity
failure;--Jeter Gutierrez Due September, 18 2017
else--Jeter Gutierrez Due September, 18 2017
report "Error detected" severity failure;--Jeter Gutierrez D
end if;--Jeter Gutierrez Due September, 18 2017
end process;--Jeter Gutierrez Due September, 18 2017
end arch_test;--Jeter Gutierrez Due September, 18 2017
```

Figure 9: VHDL code for the test bench for testing 2-Bit Comparator. In this design modelsim will be used in order to simulate what is written in the testing code. It will be used to test the values for a and b and determine whether for every possible case of comparing 2-Bit words after a certain period of time or after testing another state it will continue to be correct. The reason we do this is to have more control over our testing for correctness of our 2-Bit comparator circuit.

6.2 Simulation for 2-Bit Comparator Test.

In this simulation we will be using modelsim to run our test bench file for the 2-Bit comparator, if we get an output in the terminal of modelsim that says “Simulation successful then we know we have designed a correct 2-Bit comparator. The difference in this state is that we already wrote what values we want to test for and modelsim will create those values for us instead of us having to use the cursor to select the values using waveform. Another difference is that in this case we have an additional output pout that will be used to test whether we have an error in our comparisons.

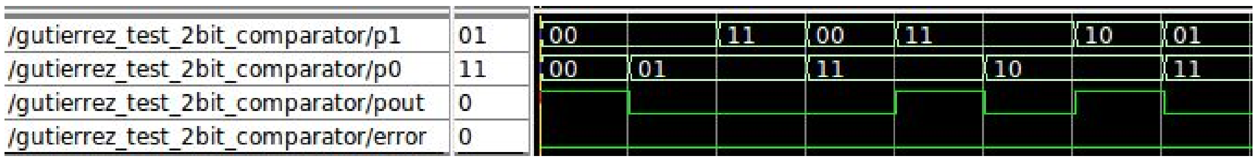

Figure 10: Vector waveform simulation for test bench of 2-Bit comparator. As we can see the error value is constantly returning 0 throughout the entire simulation, that means that our design was correct and that we did not make any mistakes, we were able to successfully design an 2-Bit comparator. The test bench here was used to test every possible combination of 2-Bit words. We also got a message in our terminal saying “Simulation successful” which means our design is correct.

7. 2-Bit Comparator Using 1-Bit Comparator.

7.1 Functionality and specifications for 2-Bit Comparator Using 1-Bit Comparator.

The purpose of the 2-Bit comparator using 1-Bit comparators is to compare whether two 2-bit words are equal to each other. It will take as an input two 2-bit words and compare them to each other 1-Bit at a time, if they are the same a signal will be sent to aeqb which is the variable storing the value of equal between the two bits of information and represents “are equal bits”. The inputs will be stored in a and b which are both 2-bit vectors. The main difference between this design and the previous design of the 2-Bit comparator is that the previous one was very tedious and if our goal is to make an 8-Bit comparator it would take a lot of time to design it so tediously and would also leave more room for mistake, that is way instead of using conditions we are using 1-Bit comparators as components.

```
library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
entity GUTIERREZ_2BIT_COMPARATOR_PORT  is--Jeter Gutierrez Due Septemb
port (--Jeter Gutierrez Due September, 18 2017
a, b: in std_logic_vector(1 downto 0);--Jeter Gutierrez Due September
aeqb : out std_logic);--Jeter Gutierrez Due September, 18 2017
end GUTIERREZ_2BIT_COMPARATOR_PORT ;--Jeter Gutierrez Due September, 1
architecture arch of GUTIERREZ_2BIT_COMPARATOR_PORT  is--Jeter Gutier
component GUTIERREZ_1BIT_COMPARATOR --Jeter Gutierrez Due September, 1
port (--Jeter Gutierrez Due September, 18 2017
I0, I1: in std_logic;--Jeter Gutierrez Due September, 18 2017
Eq : out std_logic);--Jeter Gutierrez Due September, 18 2017
end component;--Jeter Gutierrez Due September, 18 2017
signal e0,e1: std_logic;--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
H1: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(0), i1=>b(0), eq=>e0);-
H2: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(1), i1=>b(1), eq=>e1);-
aeqb <= e0 and e1;--Jeter Gutierrez Due September, 18 2017
end arch;--Jeter Gutierrez Due September, 18 2017
```

Figure 11: VHDL code for 2-Bit comparator using 1-Bit comparators as components, as we can see the design of this circuit is different from the design of the 2-Bit comparator using logic gates, instead it compares each Bit of each word individually using the 1-Bit comparator and then compares their equalities.

7.2 Simulation for 2-Bit Comparator Using 1-Bit Comparator.

In this simulation a and b are 2-Bit words that will each be given different values from 00 to 11 in different combinations in order to determine whether our design for the 2-Bit comparator that tells us if two 2-Bit words are identical is working correctly, the output of the two 2-Bit words being equal will be sent to aeqb.

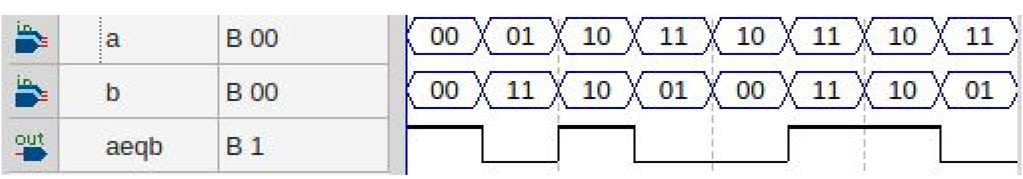

Figure 12: Vector waveform simulation for 2-Bit comparator, as we can see in this simulation when the 2-Bit words are equal we get an output of 1 to aeqb, when they are not equal aeqb is 0 this means that we were correct. Our expectations were correct this means we were able to successfully design a 2-Bit comparator.

8. 2-Bit Comparator Using 1-Bit Comparator Test.

8.1 Functionality and specifications for 2-Bit Comparator Using 1-Bit Comparator Test.

The purpose of this circuit is to create a test bench in order to simulate our 2-Bit comparator. We already simulated our 2-Bit comparator in Vector waveform but it is more professional and exact to use a test bench to test the correctness of our circuits because we have more control over the simulation this way. The test bench imitates a physical lab bench making our simulation cleaner and more organized. Additionally this design is using 1-Bit comparators as components instead of only using logic gates.

```
library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
entity GUTIERREZ_TEST_2BIT_COMPARATOR_PORT  is--Jeter Gutierrez Due September,
end GUTIERREZ_TEST_2BIT_COMPARATOR_PORT ;--Jeter Gutierrez Due September, 18 20
architecture arch_test of GUTIERREZ_TEST_2BIT_COMPARATOR_PORT  is--Jeter Gutie
component GUTIERREZ_2BIT_COMPARATOR_PORT --Jeter Gutierrez Due September, 18 20
port ( a, b: in std_logic_vector(1 downto 0);--Jeter Gutierrez Due September,
aeqb: out std_logic);--Jeter Gutierrez Due September, 18 2017
end component;--Jeter Gutierrez Due September, 18 2017
signal p1, p0: std_logic_vector(1 downto 0);--Jeter Gutierrez Due September, 1
signal pout :std_logic;--Jeter Gutierrez Due September, 18 2017
signal error :std_logic  := '0';--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
uut: GUTIERREZ_2BIT_COMPARATOR_PORT  port map(a => p0, b=> p1, aeqb => pout);-
process--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
p0 <= "00";--Jeter Gutierrez Due September, 18 2017
p1 <= "00";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "01";--Jeter Gutierrez Due September, 18 2017
p1 <= "00";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "01";--Jeter Gutierrez Due September, 18 2017
p1 <= "11";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11";--Jeter Gutierrez Due September, 18 2017
p1 <= "00";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11";--Jeter Gutierrez Due September, 18 2017
```

```
p1 <= "11";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "10";--Jeter Gutierrez Due September, 18 2017
p1 <= "11";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "10";--Jeter Gutierrez Due September, 18 2017
p1 <= "10";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11";--Jeter Gutierrez Due September, 18 2017
p1 <= "01";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
if (error = '0') then--Jeter Gutierrez Due September, 18 2017
report "No errors detected. Simulation successful"  severity-
failure;--Jeter Gutierrez Due September, 18 2017
else--Jeter Gutierrez Due September, 18 2017
report "Error detected" severity failure;--Jeter Gutierrez D
end if;--Jeter Gutierrez Due September, 18 2017
end process;--Jeter Gutierrez Due September, 18 2017
end arch_test;--Jeter Gutierrez Due September, 18 2017
```

Figure 13: VHDL code for the test bench for testing 2-Bit Comparator using 1-Bit comparators as components. In this design modelsim will be used in order to simulate what is written in the testing code. It will be used to test the values for a and b and determine whether for every possible case of comparing 2-Bit words after a certain period of time or after testing another state it will continue to be correct. The reason we do this is to have more control over our testing for correctness of our 2-Bit comparator circuit using 1-Bit comparators as components.

8.2 Simulation for 2-Bit Comparator Using 1-Bit Comparator Test.

In this simulation we will be using modelsim to run our test bench file for the 2-Bit comparator using 1-Bit comparators as components, if we get an output in the terminal of modelsim that says “Simulation successful then we know we have designed a correct 2-Bit comparator using 1-Bit comparators as components. The difference in this state is that we already wrote what values we want to test for and modelsim will create those values for us instead of us having to use the cursor to select the values using waveform. Another difference is that in this case we have an additional output pout that will be used to test whether we have an error in our comparisons.

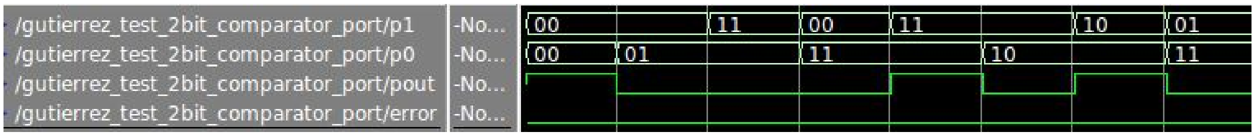

Figure 14: Vector waveform simulation for test bench of 2-Bit comparator using 1-Bit comparator as component. As we can see the error value is constantly returning 0 throughout the entire simulation, that means that our design was correct and that we did not make any mistakes, we were able to successfully design an 2-Bit comparator using 1-Bit comparator as component. The test bench here was used to test every possible combination of 2-Bit words. We also got a message in our terminal saying "Simulation successful" which means our design is correct.

9. 8-Bit Comparator Using 1-Bit Comparator.

9.1 Functionality and specifications for 8-Bit Comparator Using 1-Bit Comparator.

The purpose of the 8-Bit comparator using 1-Bit comparators is to compare whether two 8-bit words are equal to each other. It will take as an input two 8-bit words and compare them to each other 1-Bit at a time, if they are the same a signal will be sent to aeqb which is the variable storing the value of equal between the two 8-Bit words and represents "are equal bits". The inputs will be stored in a and b which are both 8-bit vectors. We will be using 1-Bit comparators as components in order to design this circuit in a more compact design.

```vhdl
1  library ieee;--Jeter Gutierrez Due September, 18 2017
2  use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
3  entity GUTIERREZ_8BIT_COMPARATOR_PORT  is--Jeter Gutierrez Due Septem
4  port (--Jeter Gutierrez Due September, 18 2017
5  a, b: in std_logic_vector(7 downto 0);--Jeter Gutierrez Due Septembe
6  aeqb : out std_logic);--Jeter Gutierrez Due September, 18 2017
7  end GUTIERREZ_8BIT_COMPARATOR_PORT ;--Jeter Gutierrez Due September,
8  architecture arch of GUTIERREZ_8BIT_COMPARATOR_PORT  is--Jeter Gutier
9  component GUTIERREZ_1BIT_COMPARATOR --Jeter Gutierrez Due September,
10 port (--Jeter Gutierrez Due September, 18 2017
11 I0, I1: in std_logic;--Jeter Gutierrez Due September, 18 2017
12 Eq : out std_logic);--Jeter Gutierrez Due September, 18 2017
13 end component;--Jeter Gutierrez Due September, 18 2017
14 signal e0,e1,e2,e3,e4,e5,e6,e7: std_logic;--Jeter Gutierrez Due Sept
15 begin--Jeter Gutierrez Due September, 18 2017
16 H1: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(0), i1=>b(0), eq=>e0);-
17 H2: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(1), i1=>b(1), eq=>e1);-
18 H3: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(2), i1=>b(2), eq=>e2);-
19 H4: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(3), i1=>b(3), eq=>e3);-
20 H5: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(4), i1=>b(4), eq=>e4);-
21 H6: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(5), i1=>b(5), eq=>e5);-
22 H7: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(6), i1=>b(6), eq=>e6);-
23 H8: GUTIERREZ_1BIT_COMPARATOR  port map(i0=>a(7), i1=>b(7), eq=>e7);-
24 aeqb <= e0 and e1 and e2 and e3 and e4 and e5 and e6 and e7;--Jeter
25 end arch;--Jeter Gutierrez Due September, 18 2017
```

Figure 15: VHDL code for 8-Bit Comparator using 1-Bit comparators as components, as we can see in lines 16-23 each bit of the 8-bit word in a is being compared to each bit in the 8-Bit word in B making sure to determine whether or not each of the values is equal or not.

9.2 Simulation for 8-Bit Comparator Using 1-Bit Comparator.

In this simulation a and b will be given alternating values within the range of 8-Bit words, we will be using higher values to show that it can compare large enough values to each other and determine if they are equal or not. The result of whether two 8-Bit words are equal or not will be sent to aeqb.

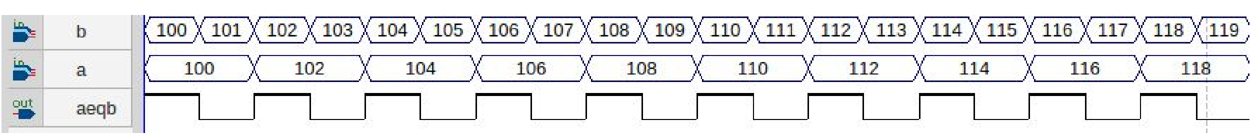

Figure 16: Vector waveform simulation for 8-Bit comparator. As we can see aeqb is outputting the correct result of 1 when the 8-Bit words a and b are equal to each other and 0 when they are not, this means we were able to successfully design a 8-Bit comparator in VHDL code correctly.

10. 8-Bit Comparator Using 1-Bit Comparator Test.

10.1 Functionality and specifications for 8-Bit Comparator Using 1-Bit Comparator Test.

The purpose of this circuit is to create a test bench in order to simulate our 8-Bit comparator. We already simulated our 8-Bit comparator in Vector waveform but it is more professional and exact to use a test bench to test the correctness of our circuits because we have more control over the simulation this way. The test bench imitates a physical lab bench making our simulation cleaner and more organized.

```
library ieee;--Jeter Gutierrez Due September, 18 2017
use ieee.std_logic_1164.all;--Jeter Gutierrez Due September, 18 2017
entity GUTIERREZ_TEST_8BIT_COMPARATOR_PORT  is--Jeter Gutierrez Due September,
end GUTIERREZ_TEST_8BIT_COMPARATOR_PORT ;--Jeter Gutierrez Due September, 18 20
architecture arch_test of GUTIERREZ_TEST_8BIT_COMPARATOR_PORT  is--Jeter Gutie
component GUTIERREZ_8BIT_COMPARATOR_PORT --Jeter Gutierrez Due September, 18 20
port ( a, b : in std_logic_vector(7 downto 0);--Jeter Gutierrez Due September,
aeqb : out std_logic);--Jeter Gutierrez Due September, 18 2017
end component;--Jeter Gutierrez Due September, 18 2017
signal p1, p0 : std_logic_vector(7 downto 0);--Jeter Gutierrez Due September,
signal pout: std_logic;--Jeter Gutierrez Due September, 18 2017
signal error : std_logic := '0';--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
uut: GUTIERREZ_8BIT_COMPARATOR_PORT  port map(a => p0, b => p1, aeqb => pout);
process--Jeter Gutierrez Due September, 18 2017
begin--Jeter Gutierrez Due September, 18 2017
p0 <= "00000000";--Jeter Gutierrez Due September, 18 2017
p1 <= "00000000";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "01010101";--Jeter Gutierrez Due September, 18 2017
p1 <= "00010101";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "01100101";--Jeter Gutierrez Due September, 18 2017
p1 <= "11111001";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11110011";--Jeter Gutierrez Due September, 18 2017
p1 <= "00010100";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11001100";--Jeter Gutierrez Due September, 18 2017
```

```vhdl
p1 <= "11001100";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "10010001";--Jeter Gutierrez Due September, 18 2017
p1 <= "11100111";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "10111001";--Jeter Gutierrez Due September, 18 2017
p1 <= "10111001";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '0') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
p0 <= "11010011";--Jeter Gutierrez Due September, 18 2017
p1 <= "01101001";--Jeter Gutierrez Due September, 18 2017
wait for 1 ns;--Jeter Gutierrez Due September, 18 2017
if (pout = '1') then--Jeter Gutierrez Due September, 18 2017
error <= '1';--Jeter Gutierrez Due September, 18 2017
end if;--Jeter Gutierrez Due September, 18 2017
wait for 200 ns;--Jeter Gutierrez Due September, 18 2017
if (error = '0') then--Jeter Gutierrez Due September, 18 201
report "No errors detected. Simulation successful"  severity
failure;--Jeter Gutierrez Due September, 18 2017
else--Jeter Gutierrez Due September, 18 2017
report "Error detected" severity failure;--Jeter Gutierrez D
end if;--Jeter Gutierrez Due September, 18 2017
end process;--Jeter Gutierrez Due September, 18 2017
end arch_test;--Jeter Gutierrez Due September, 18 2017
```

Figure 17: VHDL code for the test bench for testing 8-Bit Comparator using 1-Bit comparators as components. In this design modelsim will be used in order to simulate what is written in the testing code. It will be used to test the values for a and b and determine whether for every possible case of comparing 8-Bit words after a certain period of time or after testing another state it will continue to be correct. The reason we do this is to have more control over our testing for correctness of our 8-Bit comparator circuit using 1-Bit comparators as components.

10.2 Simulation for 8-Bit Comparator Using 1-Bit Comparator Test.

In this simulation we will be using modelsim to run our test bench file for the 8-Bit comparator, if we get an output in the terminal of modelsim that says "Simulation successful then we know we have designed a correct 8-Bit comparator. The difference in this state is that we already wrote what values we want to test for and modelsim will create those values for us instead of us having to use the cursor to select the values using waveform. Another difference is that in this case we have an additional output pout that will be used to test whether we have an error in our comparison.

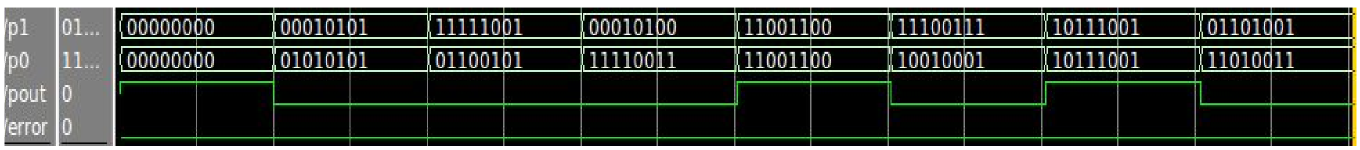

Figure 18: Vector waveform simulation for test bench of 8-Bit comparator. As we can see the error value is constantly returning 0 throughout the entire simulation, that means that our design was correct and that we did not make any mistakes, we were able to successfully design an 8-Bit. The test bench here was used to test many possible combination of 8-Bit words, but not all of them because there are 2^7*2^7 different ways to compare 8-Bit words. We also got a message in our terminal saying "Simulation successful" which means our design is correct.

11.1 Demonstration of 1 Bit Comparator on DE2-115 Board.

The inputs and outputs that are assigned to the DE2-115 Board are:

I0 is assigned to PIN_AB28

I1 is assigned to PIN_AC28

Eq is assigned to PIN_G19

notEq is assigned to PIN_F19

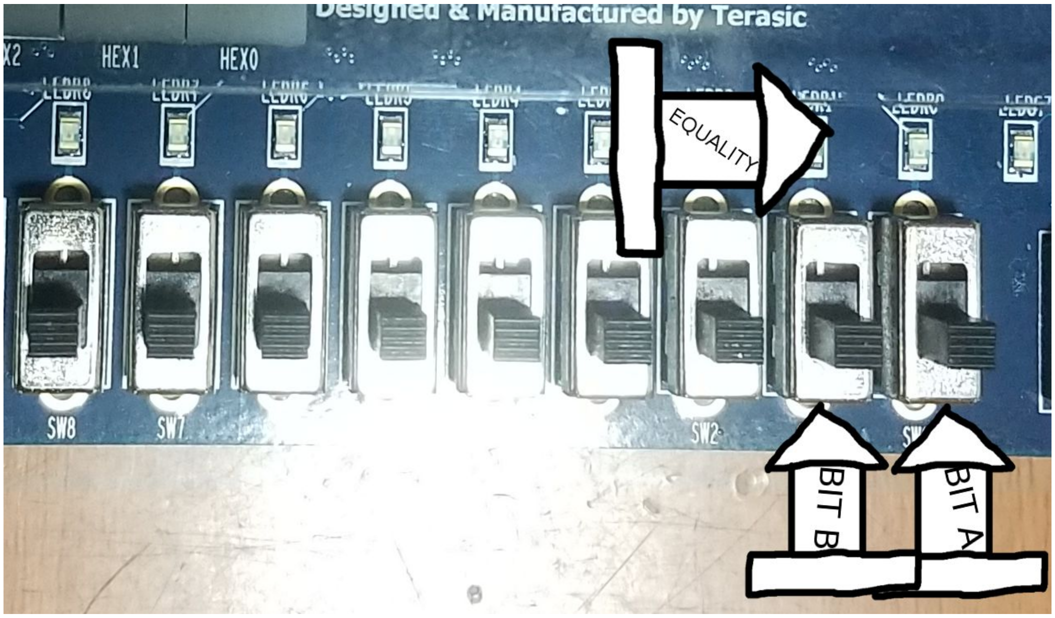

Figure 19: Digital circuit of One-Bit comparator on the DE2-115 Board.

11.2 Demonstration of 2 Bit Comparator on DE2-115 Board.

The inputs and outputs that are assigned to the DE2-115 Board are:

a[0] is assigned to PIN_AB28

a[1] is assigned to PIN_AC28

b[0] is assigned to PIN_AC27

b[1] is assigned to PIN_AD27

aeqb is assigned to PIN_G19

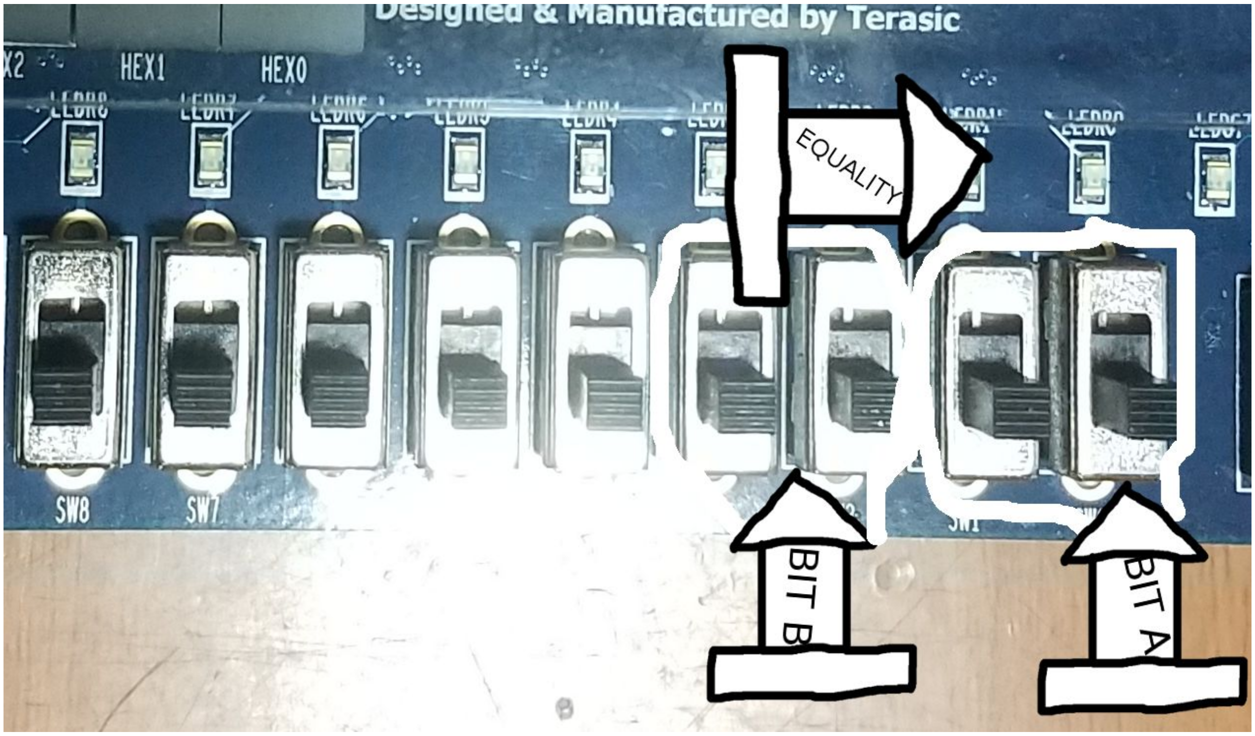

Figure 20: Digital Circuit of 2 bit comparator on DE2-115 board.

11.3 Demonstration of 8 Bit Comparator on DE2-115 Board.

The inputs and outputs that are assigned to the DE2-115 Board are:

a[0] is assigned to PIN_AB28

a[1] is assigned to PIN_AC28

a[2] is assigned to PIN_AC27

a[3] is assigned to PIN_AD27

a[4] is assigned to PIN_AB27

a[5] is assigned to PIN_AC26

a[6] is assigned to PIN_AD26

a[7] is assigned to PIN_AB26

b[0] is assigned to PIN_AC25

b[1] is assigned to PIN_AB25

b[2] is assigned to PIN_AC24

b[3] is assigned to PIN_AB24

b[4] is assigned to PIN_AB23

b[5] is assigned to PIN_AA24

b[6] is assigned to PIN_AA23

b[7] is assigned to PIN_AA22

aeqb is assigned to PIN_G19

Figure 21: Digital Circuit of 8 bit comparator on DE2-115 board.

12. Conclusion

As it turns out design n-Bit comparators is very useful, it is useful because it helps us make it easier to compare total n-Bit words in memory that is going to be more useful down the line in the future labs that we will design. I learned today that using components is essential to doing vhdl. It is essential because it enforces my understanding of the logic gates that are being used in the circuits designed and also helps me make less mistakes and understand how vectors work better. I learned how to be more efficient by making a correct design of a minor circuit and then use it as a component for a higher level design of a circuit like using a 1-Bit comparator as a component for an 8-Bit comparator. I also learned how to used test bench to test the correctness of my circuit designs and to be more exact and professional with my testing of circuits through simulations in test bench code using modelsim.

13. Appendix.

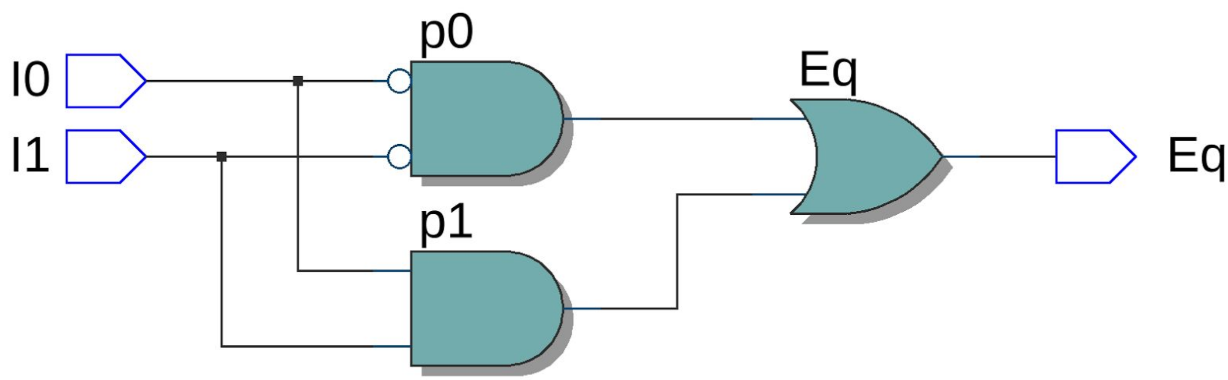

Figure 22: Block diagram for 1-Bit comparator using logic gates.

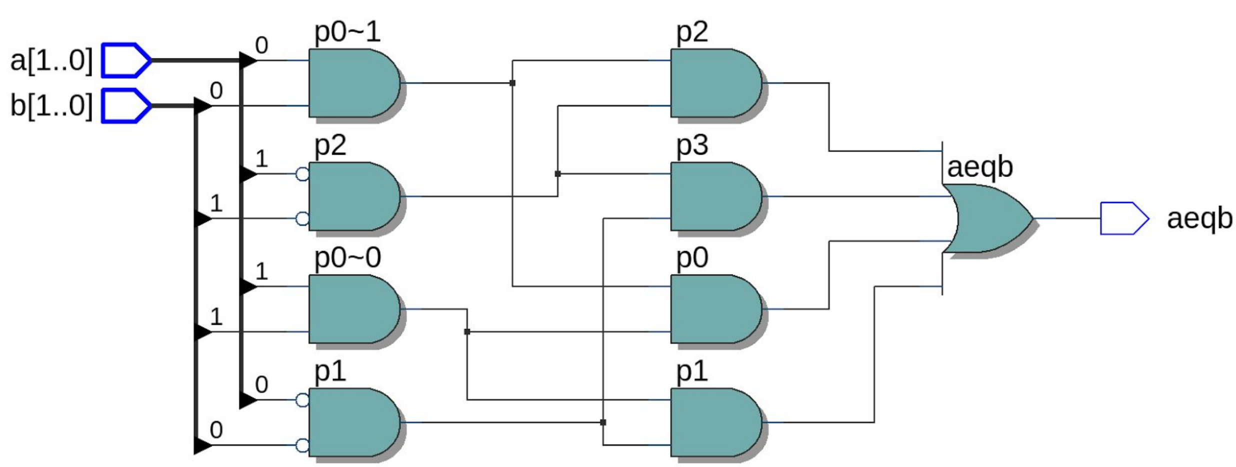

Figure 23: Block diagram for 2-Bit comparator using logic gates.

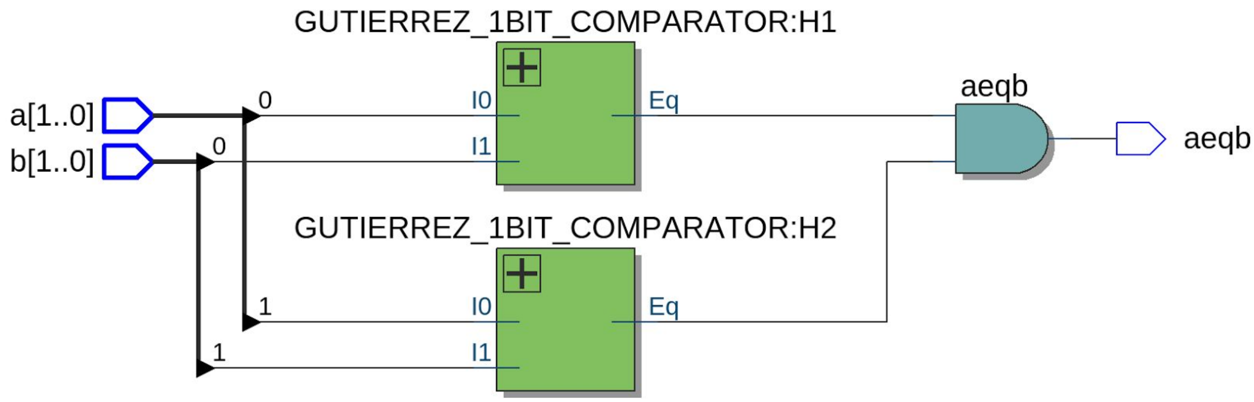

Figure 24: Block diagram for 2-bit comparator using 1-Bit comparators as components.

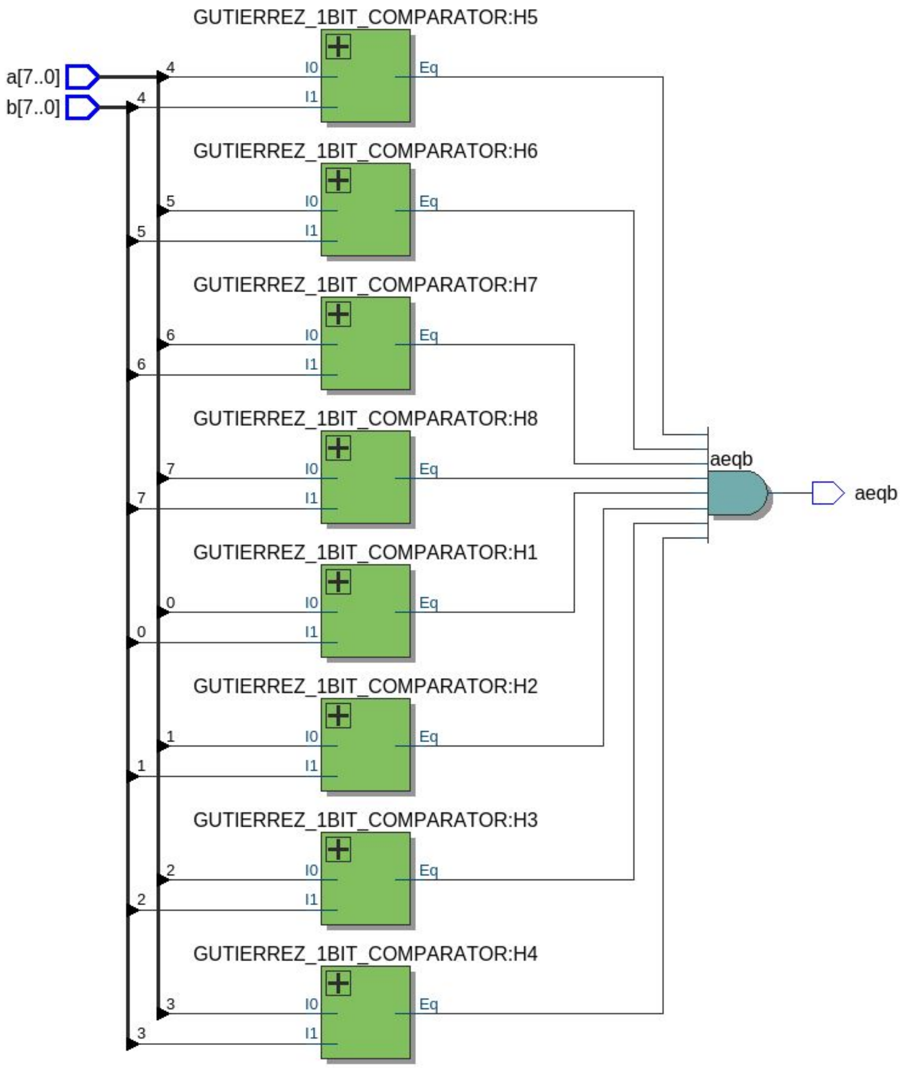

Figure 25: Block Diagram for 8-Bit comparator using 1-Bit comparators as components. In this extra image of the design for the 8-Bit comparator we can see that it is using 8 components of 1-Bit comparators making the design sleek and simplified, easier to understand.