Weifan Lin
Csc343
02/27/2015

Quartus De 2 Tutorial with Comparator

Objectives:

The goal of this lab was to be able to build circuits, and run waveform simulation in Quartus and test circuits on DE2 board by implementing bits comparators. Also to understand some fundamentals of the software Quartus.

Specifications:

One-bit comparator: two input ports, two signals and one output port.

Two-bit comparator: two input ports, four signals and one output port.

Two-bit comparator build using one-bit comparator: two input ports, two signals and one output port.

Eight-bit comparator build using one-bit comparator: two input ports, eight signals and one output port.

Functionality:

One-bit comparator takes two inputs and compare them against each other, if they are equal then the output would be 1 otherwise 0.

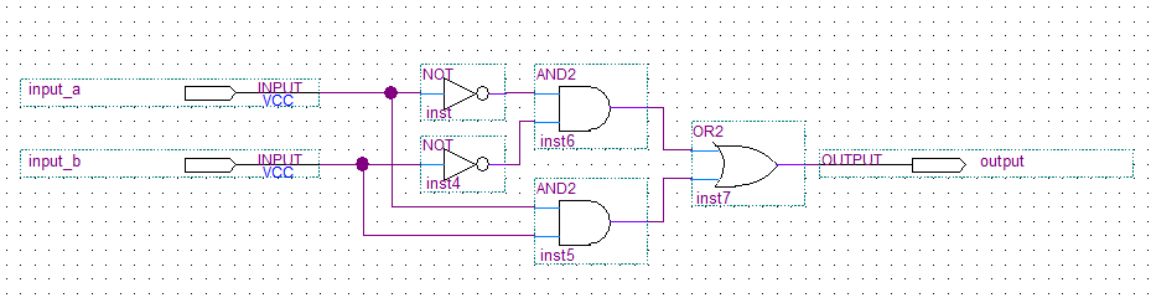
Two-bit comparator checks for equality in a similar way as one-bit comparator. However, two-bit comparator uses a logical statement given below:

aeqb = (a1'.b1').(a0'.b0') + (a1'.b1').(a0.b0) + (a1.b1).(a0'b0') + (a1.b1).(a0.b0)

Eight-bit comparator works using port mapping in the same way as the two-bit comparator, but eight-bit comparator uses eight one-bit comparator.

Design:

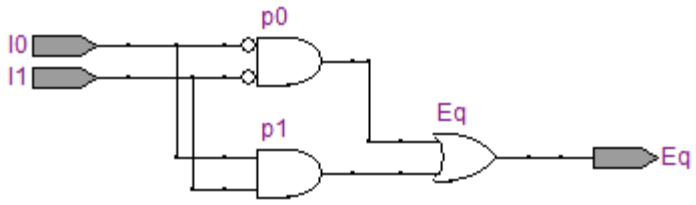
One-bit comparator block diagram:



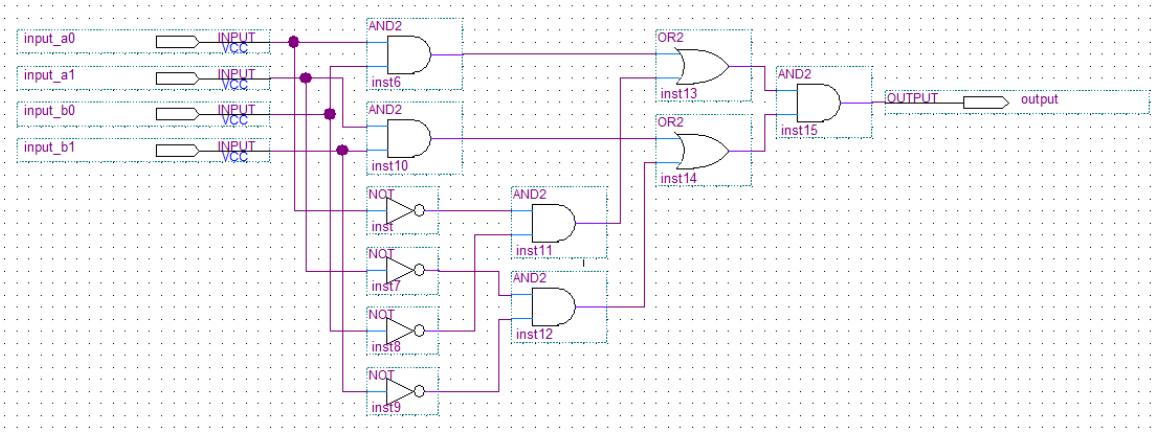
One-bit comparator VHDL code:

```
1   Library ieee;
2   Use ieee.std_logic_1164.all;
3
4   Entity equal is
5   Port (
6   I0, I1: in std_logic;
7   Eq    : out std_logic);
8   End equal;
9
10  Architecture arch of equal is
11  Signal p0, p1 : std_logic;
12  begin
13  Eq <= p0 or p1;
14  P0 <= (not I0) and (not I1);
15  P1<= i0 and i1;
16  End arch;
```

One-bit comparator VHDL RTL viewer:



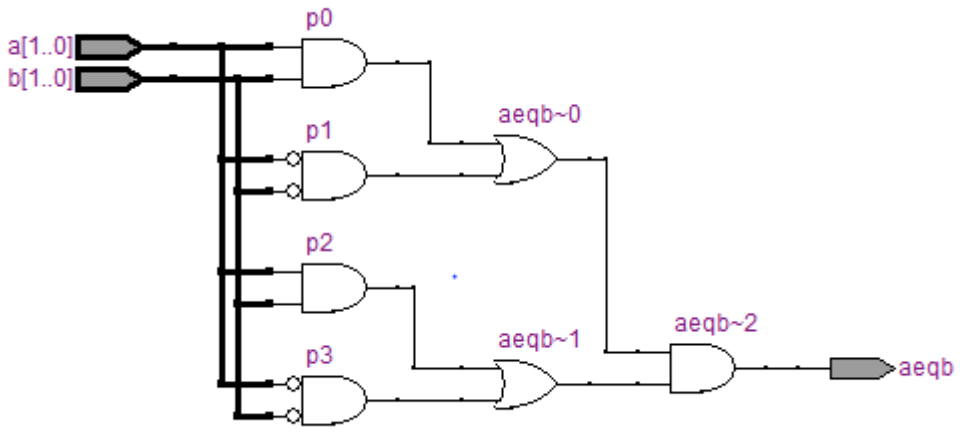
Two-bit comparator block diagram:



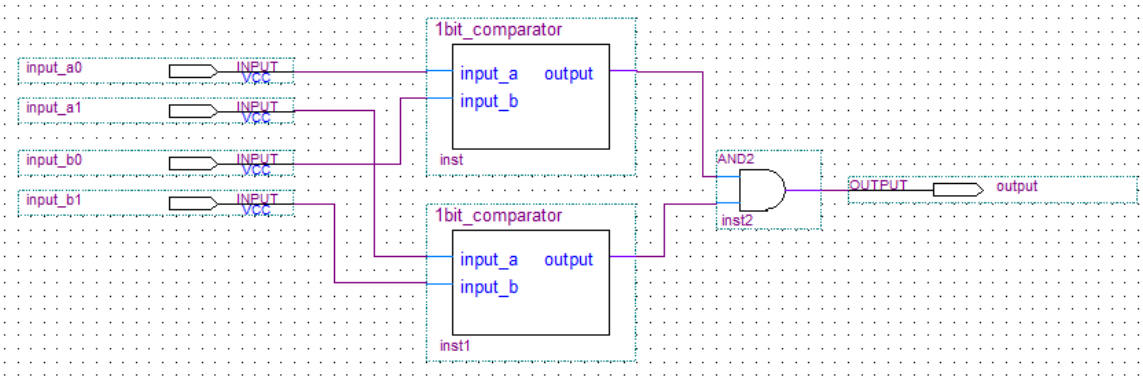
Two-bit comparator VHDL code:

```
1   Library ieee;
2       Use ieee.std_logic_1164.all;
3
4       Entity two_bit_equal is
5       Port (
6       a, b: in std_logic_vector(1 downto 0);
7       aeqb    : out std_logic);
8       End two_bit_equal;
9
10      Architecture arch of two_bit_equal is
11      Signal p0, p1,p2,p3 : std_logic;
12      begin
13      aeqb <=(p0 or p1) and (p2 or p3); --Enter you code here;
14      P0 <=a(0) and b(0); --Enter you code here;
15      P1<=(not a(0)) and (not b(0)); --Enter you code here.;
16      P2<=a(1) and b(1);--Enter you code here;
17      P3 <=(not a(1)) and (not b(1));--Enter you code here;
18      End arch;
```

Two-bit comparator VHDL RTL viewer:



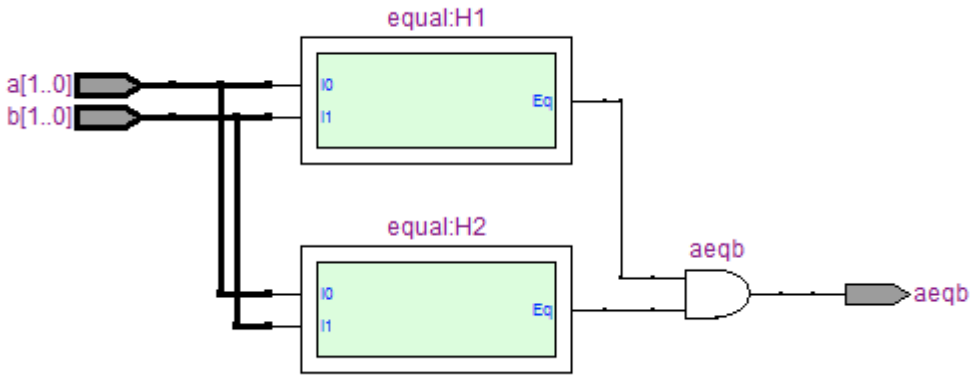
Two-bit comparator build using one-bit comparator block diagram:



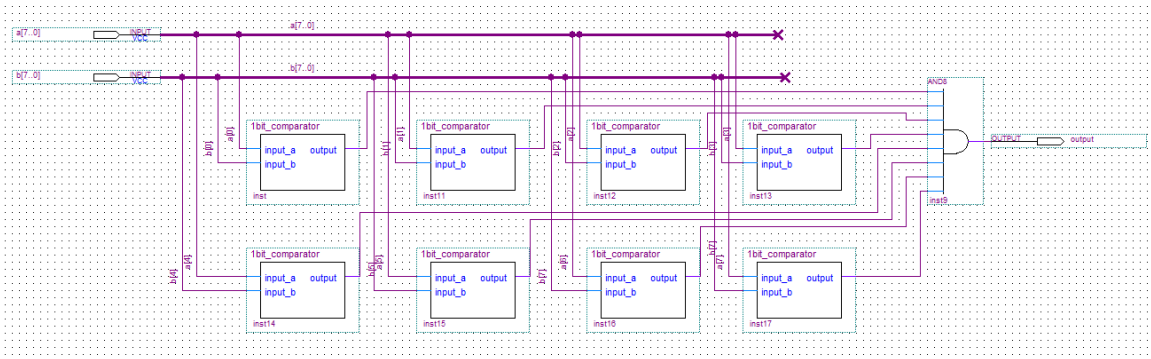
Two-bit comparator using port maps VHDL code:

```
1     Library ieee;
2     Use ieee.std_logic_1164.all;
3
4     Entity two_bit_equal_port is
5     Port (
6     a, b: in std_logic_vector(1 downto 0);
7     aeqb    : out std_logic);
8     End two_bit_equal_port;
9
10    Architecture arch of two_bit_equal_port is
11
12    component equal
13    Port (
14    I0, I1: in std_logic;
15    Eq    : out std_logic);
16    End component;
17
18    signal e0,e1: std_logic;
19
20    begin
21
22    H1: equal
23    port map(i0=>a(0), i1=>b(0), eq=>e0);
24    H2: equal
25    port map(i0=>a(1), i1=>b(1), eq=>e1);
26
27        aeqb <= e0 and e1;
28
29    end arch;
```

Two-bit comparator using port maps VHDL viewer:



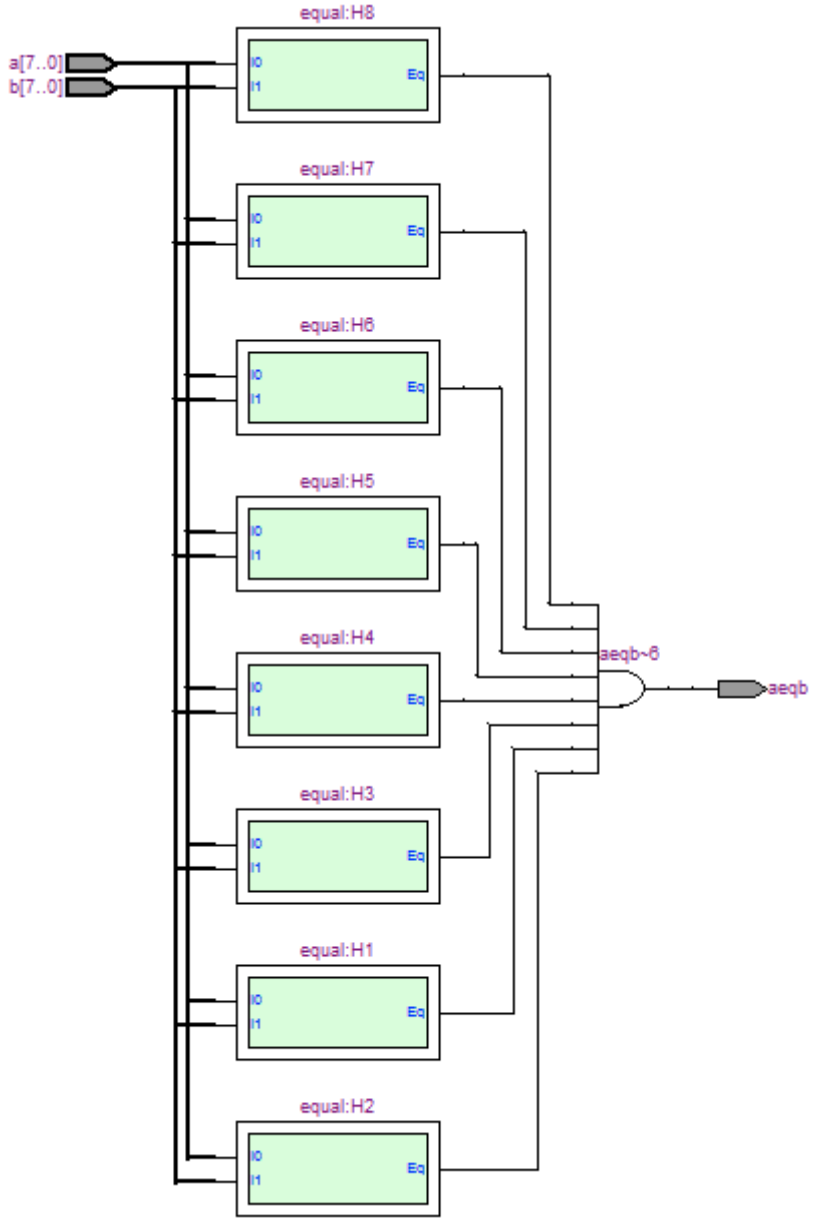
Eight-bit comparator build using one-bit comparator block diagram:



Eight-bit comparator VHDL code:

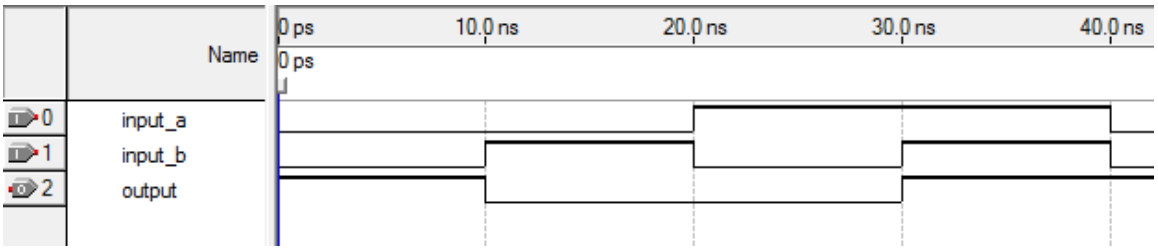
```
Library ieee;
    Use ieee.std_logic_1164.all;

    Entity eight_bit_equal_port is
    Port (
    a, b: in std_logic_vector(7 downto 0);
    aeqb    : out std_logic);
    End eight_bit_equal_port;

    Architecture arch of eight_bit_equal_port is

    component equal
    Port (
    I0, I1: in std_logic;
    Eq    : out std_logic);
    End component;

    signal e0,e1,e2,e3,e4,e5,e6,e7: std_logic;

    begin

    H1: equal
    port map(i0=>a(0), i1=>b(0), eq=>e0);
    H2: equal
    port map(i0=>a(1), i1=>b(1), eq=>e1);
    H3: equal
    port map(i0=>a(2), i1=>b(2), eq=>e2);
    H4: equal
    port map(i0=>a(3), i1=>b(3), eq=>e3);
    H5: equal
    port map(i0=>a(4), i1=>b(4), eq=>e4);
    H6: equal
    port map(i0=>a(5), i1=>b(5), eq=>e5);
    H7: equal
    port map(i0=>a(6), i1=>b(6), eq=>e6);
    H8: equal
    port map(i0=>a(7), i1=>b(7), eq=>e7);

    aeqb <= e0 and e1 and e2 and e3 and e4 and e5 and e6 and e7;

    end arch;
```

Eight-bit comparator VHDL RTL viewer:



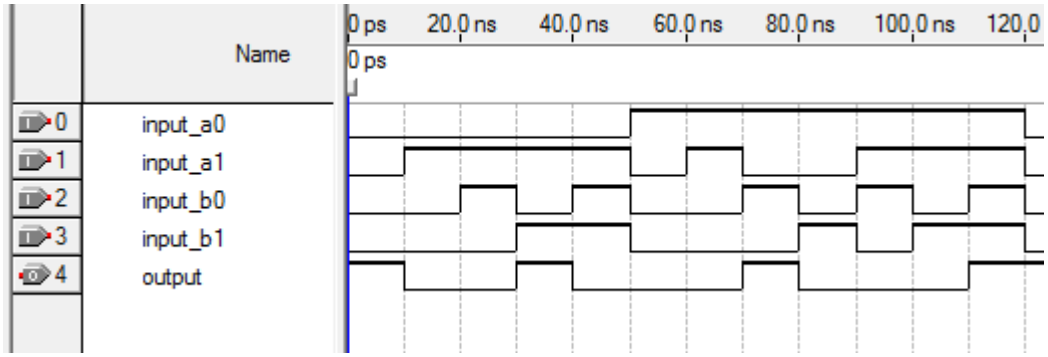
Simulation:

One-bit comparator waveform:



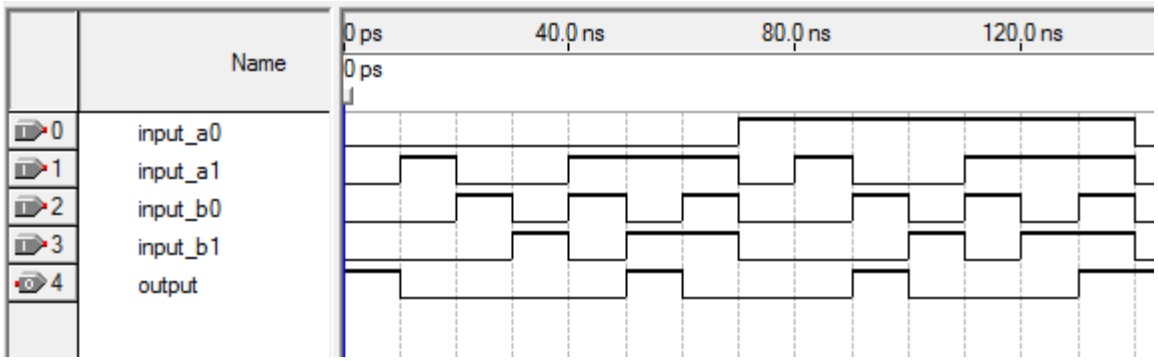
In one-bit comparator there are only two cases when the output is 1, these are when the 2 input are both 0 and when they are both 1. As we can see from the waveform, the output has correct results.

Two-bit comparator waveform:



Clearly from the waveform, we can see that in interval 0-10.0ns the inputs are 00 and 00 and the output is 1 because the inputs are equal. In interval 30-40ns the inputs are 01 and 01 and the output is 1. As well as in interval 70-80ns and 110-120ns, outputs display 1 when the inputs are the same. Also in all the other cases when the inputs are different, the output is 0.

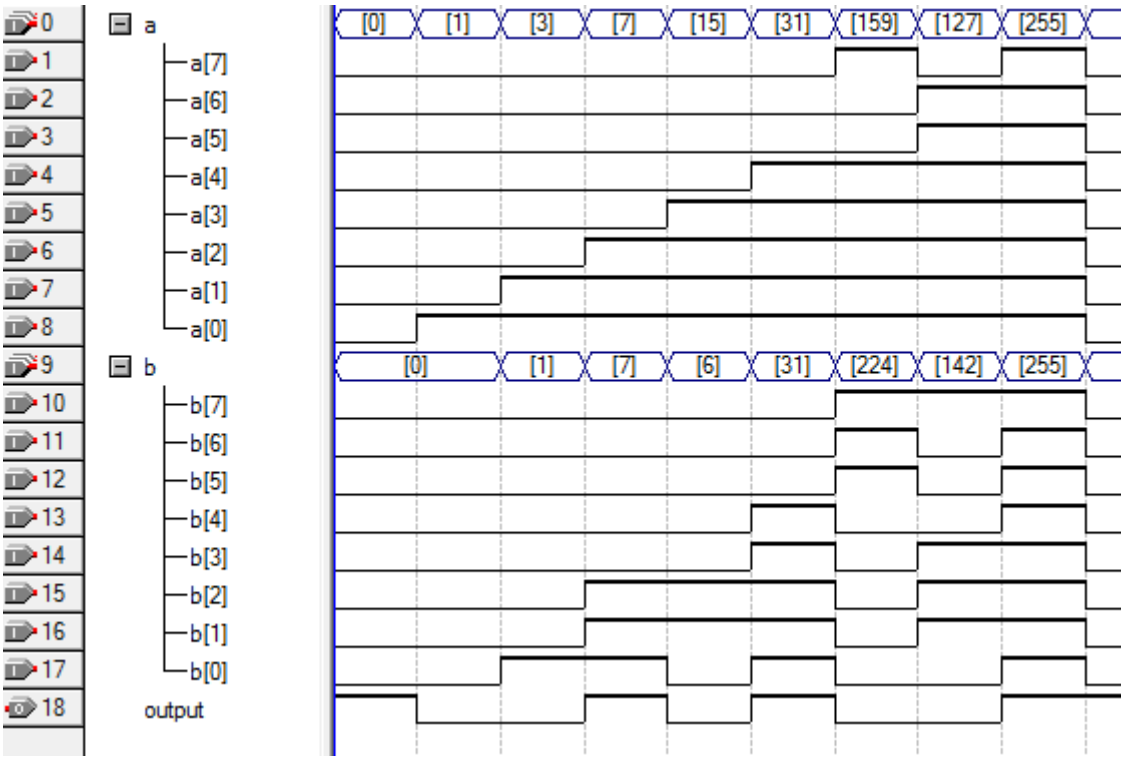
Two-bit comparator build using one-bit comparator:



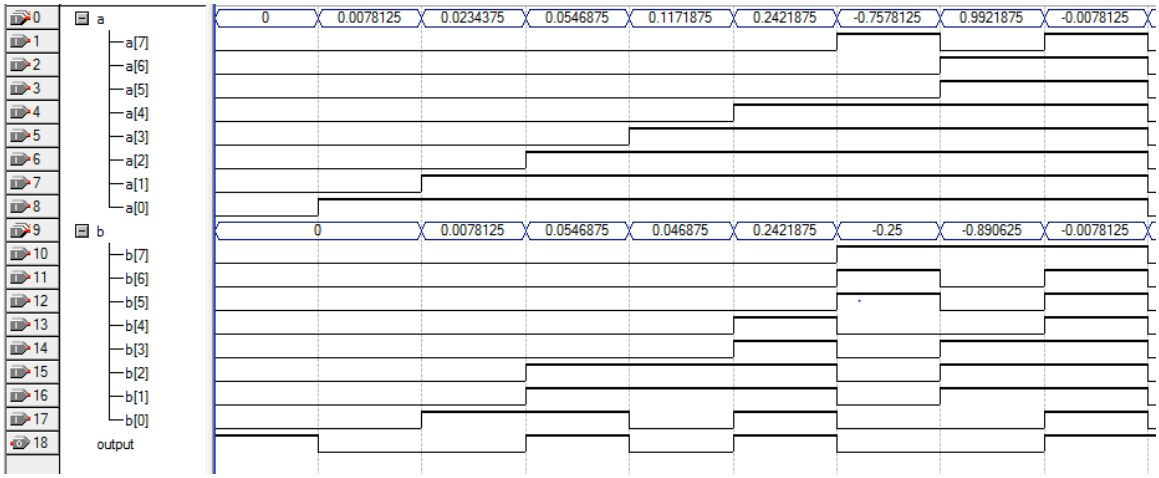
In this waveform, the results are same as the results from two-bit comparator waveform.

Output returns 1 when the inputs have the same two-bit value and 0 otherwise.

Eight-bit comparator build using one-bit comparator waveform:



The inputs are showing as decimal (however, we can change the radix when we right-click on the input and select properties. As we can see clearly from the waveform, the output is 1 when the inputs have the same value and is 0 when they are different. In additional, when we change inputs' radix the output will have the same results, we can see the figure below as we changed the radix to fractional.



Conclusion:

From this lab, we learned how to build circuits, and run waveform simulation in Quartus and test circuits on DE2 board. Implementing bits comparators and simulating waveforms in Quartus were an easy task since I've already taken CSc211 class. Overall, both Quartus and Model-Sim are good tool to be used to practice with implementing digital circuits and test them.

Appendix:

Pin Assignment:

to, location

a[0], PIN_N25

a[1], PIN_N26

a[2], PIN_P25

a[3], PIN_AE14

a[4], PIN_AF14

a[5], PIN_AD13

a[6], PIN_AC13

a[7], PIN_C13

b[0], PIN_B13

b[1], PIN_A13

b[2], PIN_N1

b[3], PIN_P1

b[4], PIN_P2

b[5], PIN_T7

b[6], PIN_U3

b[7], PIN_U4

output, PIN_AE23

1- bit comparator vhdl code:

```
Library ieee;
```

```
Use ieee.std_logic_1164.all;

Entity equal is
Port (
I0, I1: in std_logic;
Eq    : out std_logic);
End equal;

Architecture arch of equal is
Signal p0, p1 : std_logic;
begin
Eq <= p0 or p1;
P0 <= (not I0) and (not I1);
P1<= i0 and i1;
End arch;
```

2- 2-bit comparator vhdl code:

```
Library ieee;
    Use ieee.std_logic_1164.all;

    Entity two_bit_equal is
    Port (
```

```
a, b: in std_logic_vector(1 downto 0);
aeqb    : out std_logic);
End two_bit_equal;

Architecture arch of two_bit_equal is
Signal p0, p1,p2,p3 : std_logic;
begin
aeqb <=(p0 or p1) and (p2 or p3); --Enter you code here;
P0 <=a(0) and b(0); --Enter you code here;
P1<=(not a(0)) and (not b(0)); --Enter you code here.;
P2<=a(1) and b(1);--Enter you code here;
P3 <=(not a(1)) and (not b(1));--Enter you code here;
End arch;
```

3- 2-bit portmap code

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity two_bit_equal_port is
Port (
a, b: in std_logic_vector(1 downto 0);
aeqb    : out std_logic);
```

End two_bit_equal_port;

Architecture arch of two_bit_equal_port is

--component declaration...we are telling the compiler which components we want to use from the library.

component equal

Port (

I0, I1: in std_logic;

Eq : out std_logic);

End component;

signal e0,e1: std_logic;

begin

--instantiates two one-bit comparators

H1: equal

port map(i0=>a(0), i1=>b(0), eq=>e0);

H2: equal

port map(i0=>a(1), i1=>b(1), eq=>e1);

```
aeqb <= e0 and e1;

end arch;
```

4- 8-bit portmap vhdl code

```
Library ieee;
    Use ieee.std_logic_1164.all;

    Entity eight_bit_equal_port is
    Port (
    a, b: in std_logic_vector(7 downto 0);
    aeqb    : out std_logic);
    End eight_bit_equal_port;

    Architecture arch of eight_bit_equal_port is

    --component declaration...we are telling the compiler which components we want to use from the library.

    component equal
    Port (
```

```
I0, I1: in std_logic;
Eq    : out std_logic);
End component;

signal e0,e1,e2,e3,e4,e5,e6,e7: std_logic;

begin

--instantiates two one-bit comparators

H1: equal
port map(i0=>a(0), i1=>b(0), eq=>e0);
H2: equal
port map(i0=>a(1), i1=>b(1), eq=>e1);
H3: equal
port map(i0=>a(2), i1=>b(2), eq=>e2);
H4: equal
port map(i0=>a(3), i1=>b(3), eq=>e3);
H5: equal
port map(i0=>a(4), i1=>b(4), eq=>e4);
H6: equal
port map(i0=>a(5), i1=>b(5), eq=>e5);
H7: equal
```

```
        port map(i0=>a(6), i1=>b(6), eq=>e6);
        H8: equal
        port map(i0=>a(7), i1=>b(7), eq=>e7);

        --a and b are equal if individual bits are equal.

        aeqb <= e0 and e1 and e2 and e3 and e4 and e5 and e6 and e7;

end arch;
```