LAB 1

(1)

Rachel Ruddy - 261162987
Natasha Lawford - 261178596
Rachel Bunsick - 261159677

(2) an executive summary (short description of what you have done in this VHDL assignment

In this assignment, we were given an OR gate along with a testbench to run and capture its waveform. This familiarized us with the Quartus and ModelSim software required to run VHDL. We then made our own implementation of a XNOR gate and testbench, ran it, and captured its waveform.

(3) answers to all questions (if applicable),

Below is our implementation of the XNOR gate and its testbench.

XNOR Gate:

```
-- Simple XNOR gate design
library IEEE;
use IEEE.std_logic_1164.all;

entity xnor_gate is
port(
  a: in std_logic;
  b: in std_logic;
  q: out std_logic);
end xnor_gate;

architecture rtl of xnor_gate is
begin
  process(a, b) is
  begin
    q <= a xnor b;
  end process;
end rtl;
```

XOR Gate Testbench:

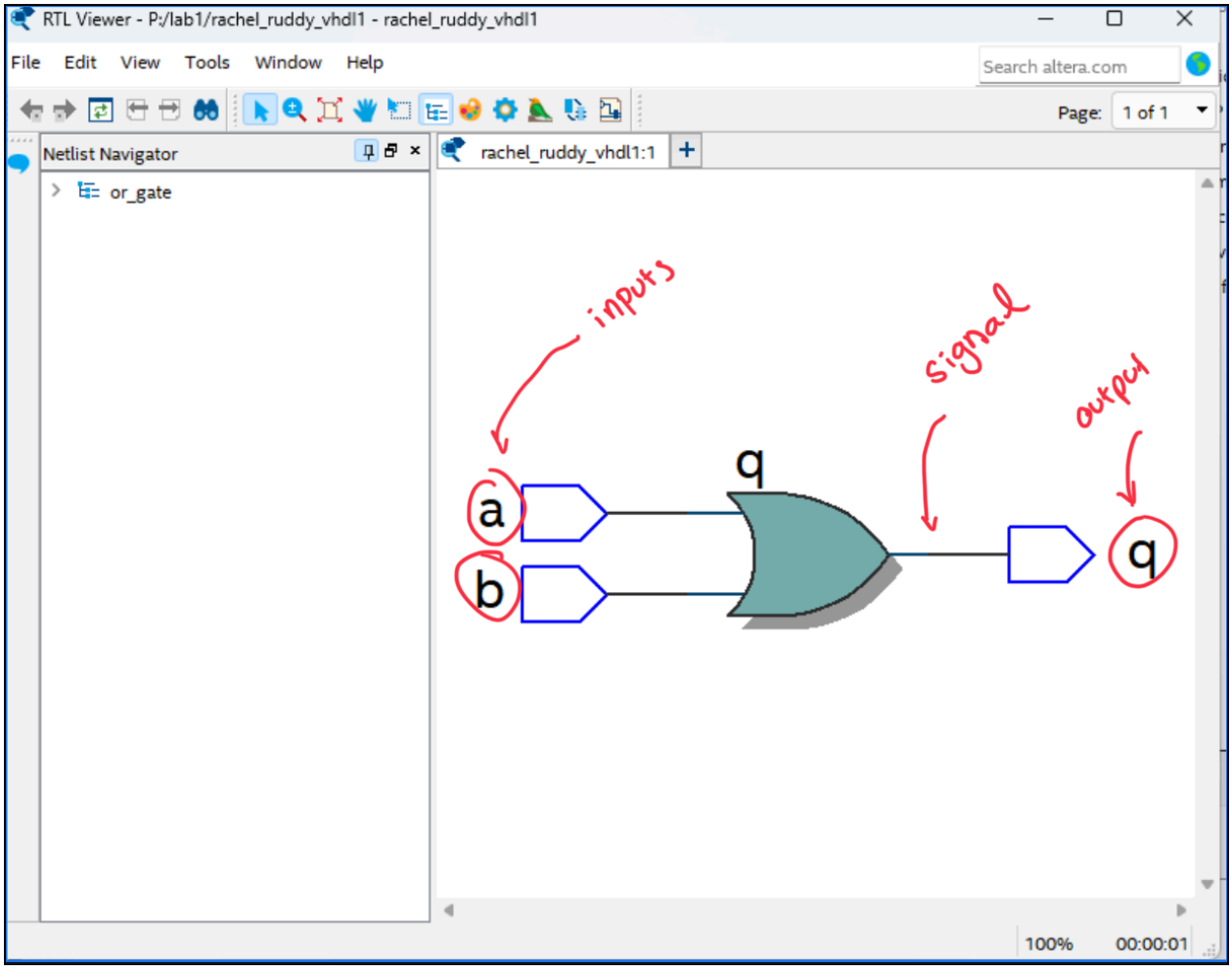
```
-- Testbench for XNOR gate
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
-- empty
end testbench;

architecture tb of testbench is

-- DUT component
component xnor_gate is
port(
  a: in std_logic;
```

```
  b: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, q_out: std_logic;

begin

  -- Connect DUT
  DUT: xnor_gate port map(a_in, b_in, q_out);

  process
  begin
    a_in <= '0';
    b_in <= '0';
    wait for 1 ns;
    assert(q_out='1') report "Fail 0/0" severity error;

    a_in <= '0';
    b_in <= '1';
    wait for 1 ns;
    assert(q_out='0') report "Fail 0/1" severity error;

    a_in <= '1';
    b_in <= '0';
    wait for 1 ns;
    assert(q_out='0') report "Fail 1/X" severity error;

    a_in <= '1';
    b_in <= '1';
    wait for 1 ns;
    assert(q_out='1') report "Fail 1/1" severity error;

    -- Clear inputs
    a_in <= '0';
    b_in <= '0';

    assert false report "Test done." severity note;
    wait;
```

```
  end process;
end tb;
```

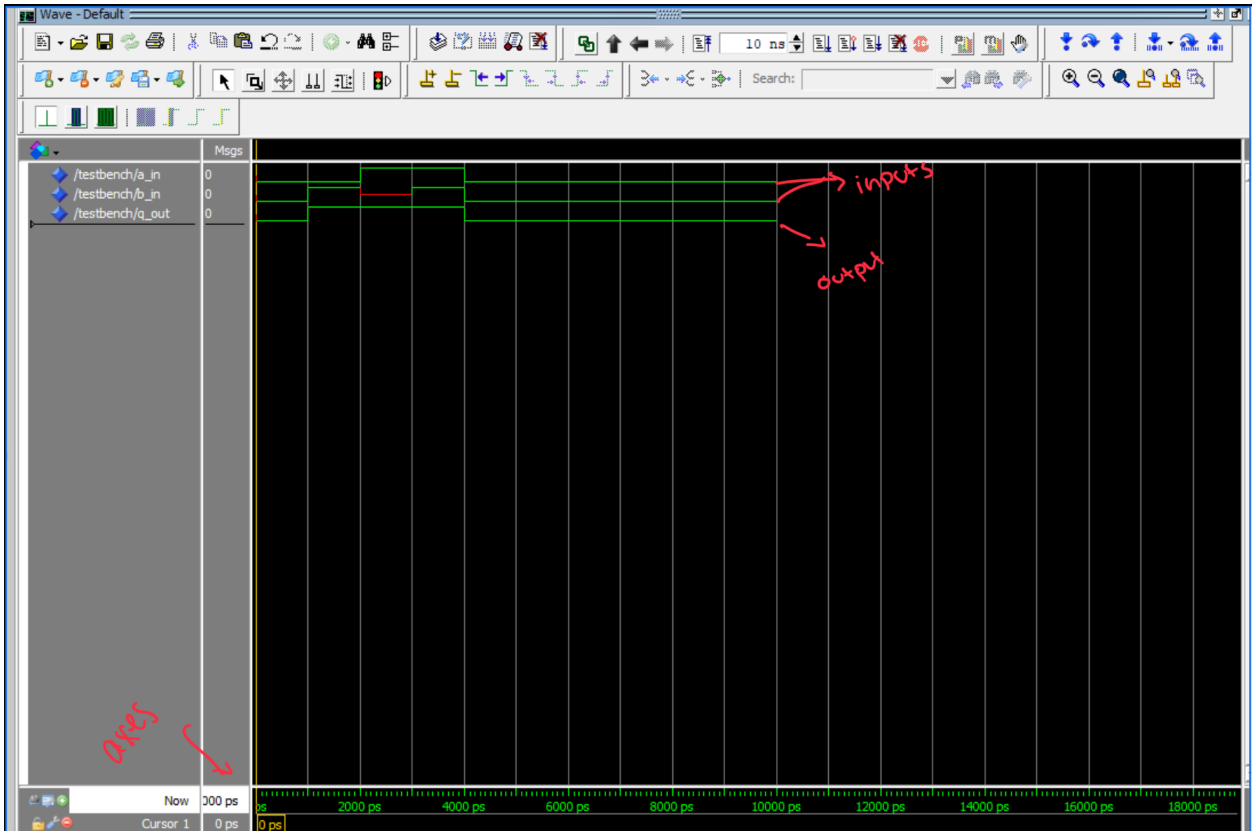
(4) legible figures (screenshots) of schematics and simulation results, where all inputs, outputs, signals, and axes are marked and visible (if applicable), and an explanation of the results obtained in the assignments (mark important points on the simulation plots)

Figure 1.0: OR gate schematic of the designed circuit



As seen above in figure 1.0, a logic gate diagram was produced from the VHDL code. As annotated above, a and b are the inputs to the OR gate, which then produces the output, q.

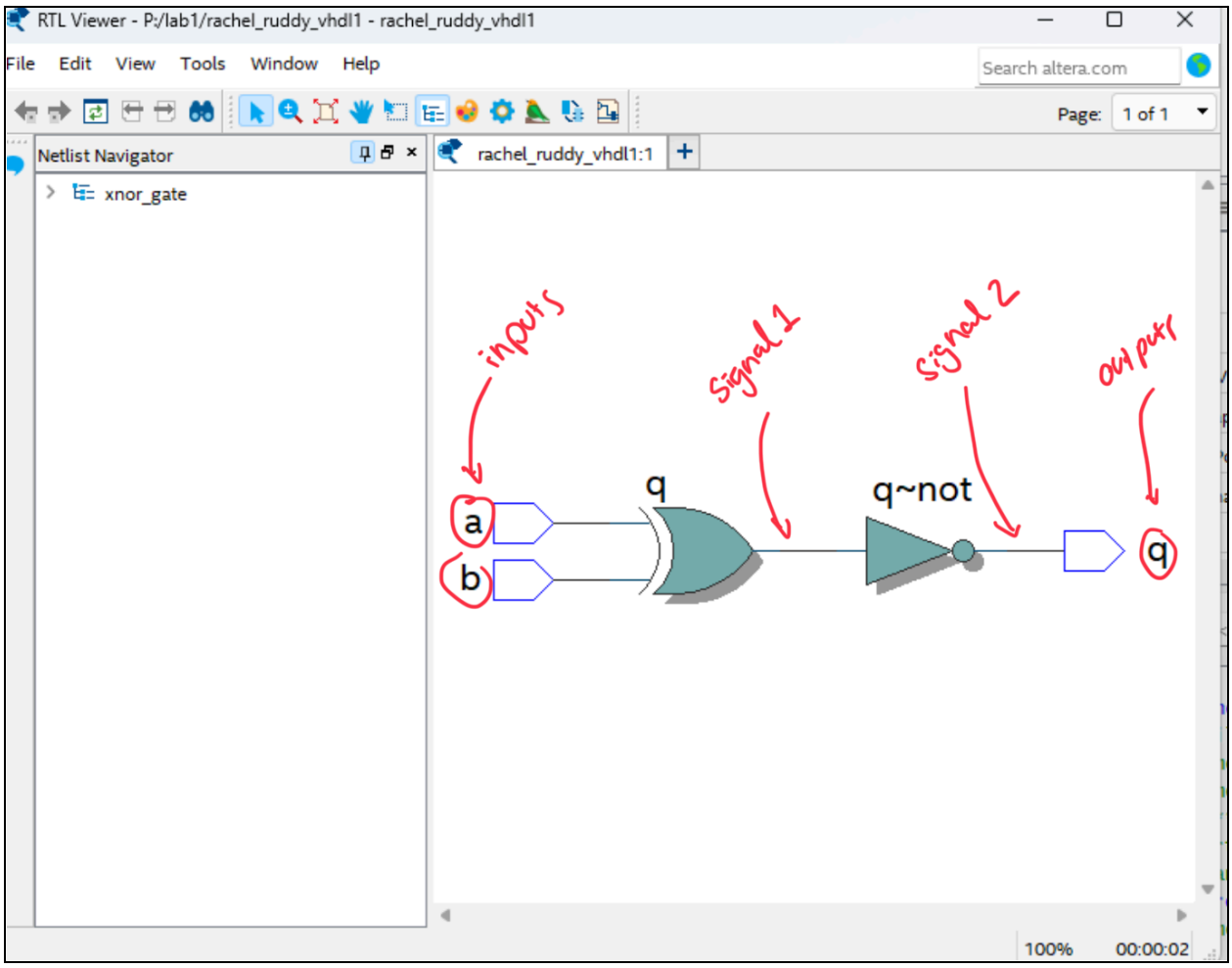
Figure 1.1: OR gate waveform



This figure represents a plot of the signals for the two inputs (a and b) as well as the output (q). Looking at the time interval [0 ps, 1000 ps], we can see that the value of the signals a, b, and q are all 0. However, on the time interval [1000 ps, 2000 ps] the value of b and q both switch to 1. Then on [2000 ps, 3000 ps], a switches to 1 and b switches back to 0, and the function remains at 1. Finally, on [3000 ps, 4000 ps] b switches back to 1 while a and f also remain constant at 1 before switching back to 0 indefinitely. This information can also be viewed in the following truth table, which matches the logic of an OR gate confirming that our code and testbench are correct.

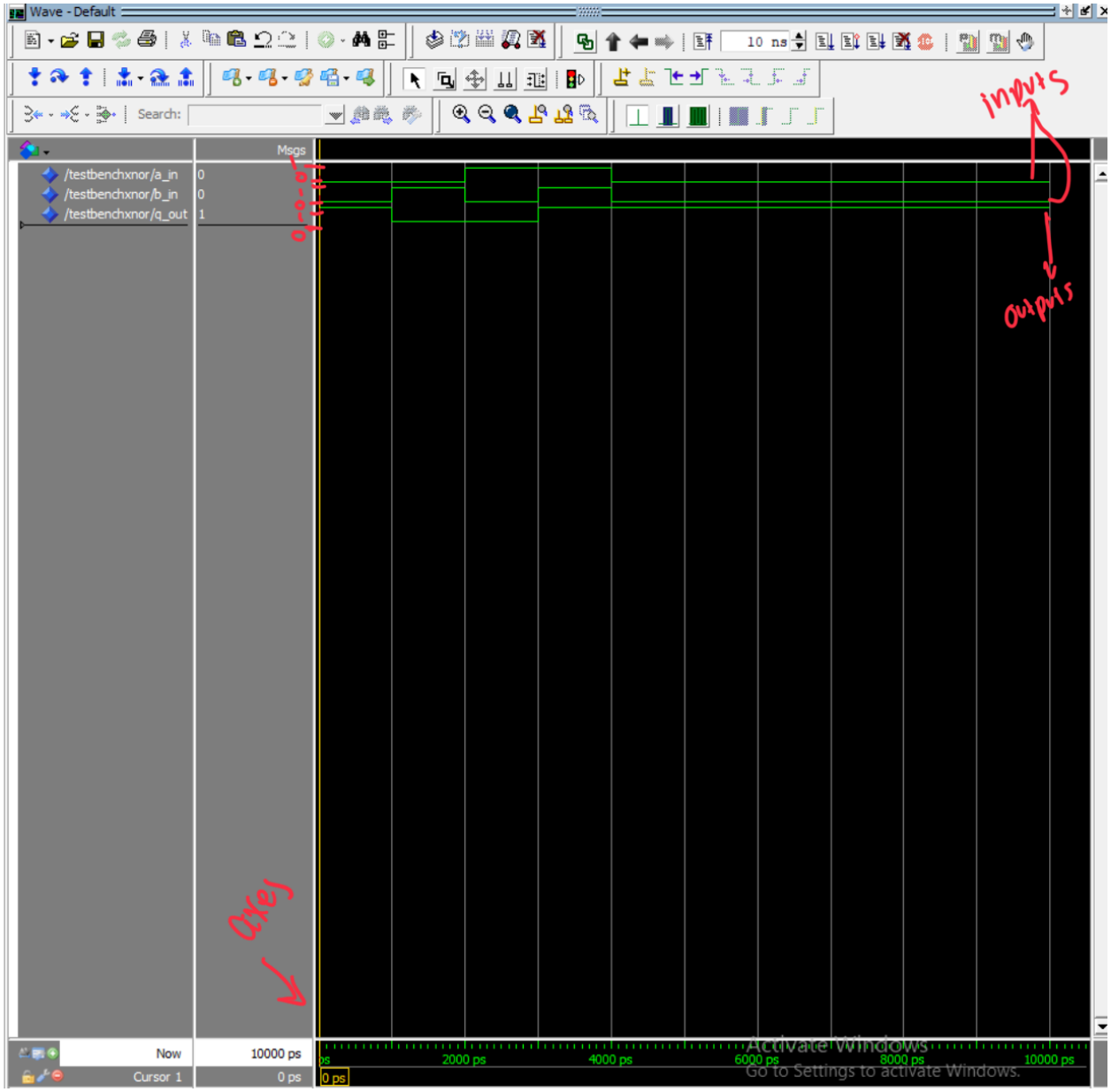
a	b	q
0	0	0
0	1	1
1	1	1
1	1	1

Figure 1.3: XNOR gate schematic of the designed circuit



As seen above in figure 1.3, a logic gate diagram was produced from the VHDL code. As annotated above, a and b are the inputs of the XOR gate, which then produces a signal that is the input of the NOT gate. Ultimately, the final output, q, is the output of this NOT gate.

Figure 1.4: XNOR waveform



This figure represents a plot of the signals for the two inputs (a and b) as well as the output (q). Looking at the time interval [0 ps, 1000 ps], we can see that the value of the signals a and b are both 0 while the function f is 1. On the time interval [1000 ps, 2000 ps] the value of b switches to 1 while f switches to 0. Then on [2000 ps, 3000 ps], a switches to 1 and b switches back to 0, and the function remains at 0. Finally, on [3000 ps, 4000 ps] b and q both switch to 1 while a remains constant at 1 before switching back to 0 indefinitely. This information can also be viewed in the following truth table, which matches the logic of an XNOR gate confirming that our code and testbench are correct.

a	b	q
0	0	1
0	1	0
1	1	0
1	1	1

(5) Conclusions:

Overall, this lab introduced us to programming in VHDL by using Quartus and Modelsim Altera. We got to understand the actual structure of VHDL files better by seeing example code and implementing our own XNOR gate code and testbench file. It was an educational and fun way to see how information is processed at the gate level, and it was extremely informative to see the waveforms and gate schematics side by side.