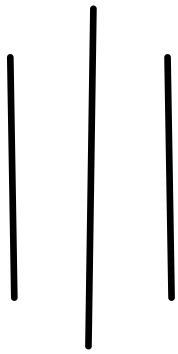
Tribhuvan University
Institute Of Science and Technology

Prithvi Narayan Campus

BSc.CSIT Program

PRACTICAL REPORT

(Computer Architecture)



Submitted To

Mrs. Amrita Thapa
Department of Computer Science & Information Technology
Prithvi Narayan Campus, Pokhara

Submitted By

Mahesh Kumar Udas
Roll No. 21
2080 Batch

2nd Year / 3rd Semester

INDEX

Name: Mahesh Kumar Udas | Roll No.: 21

Faculty: BSc.CSIT | Semester: Third

Subject: Computer Architecture | Year : Second

S.N.	Title	Date of Submission	Sign
1.	To design and simulate a 2-input AND gate using VHDL, and verify its functionality using a testbench.	**2081/12/**	
2.	To design and simulate a 2-input OR gate using VHDL, and verify its functionality using a testbench.	**2081/12/**	
3.	To design and simulate a NOT gate using VHDL, and verify its functionality using a testbench.	**2081/12/**	
4.	To design and simulate a NAND gate using VHDL, and verify its functionality using a testbench.	**2081/12/**	
5.	To design and simulate a **Half Adder Circuit** using VHDL, and verify its functionality using a testbench.	**2081/12/**	
6.	To design and simulate a **Full Adder Circuit** using VHDL, and verify its functionality using a testbench.	**2081/12/**	

LAB 1

OBJECTIVE: To design and simulate a 2-input AND gate using VHDL, and verify its functionality using a testbench.

TOOLS USED:

- VHDL (VHSIC Hardware Description Language)
- https://www.edaplayground.com/ (for simulation)

THEORY:

The aim of this lab is to create a simple 2-input AND gate in VHDL and simulate it using a testbench. The design code describes the logic of the AND gate, and the testbench applies all possible input combinations to validate the behavior of the gate.

VHDL CODE

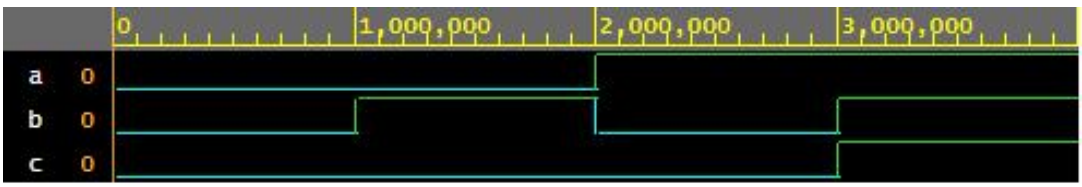
DESIGN:

```
library ieee;
use ieee.std_logic_1164.all;

entity andgate is
  port(  a,b:in std_logic;
         c: out std_logic);
end andgate;

architecture behavior of andgate is
         begin
         c <= a and b;
end behavior;
```

TESTBENCH

```
library ieee;
use ieee.std_logic_1164.all;

entity andgate_tb is
         ---> no content
end andgate_tb;

architecture test of andgate_tb is
         component andgate
         port(a,b: in std_logic;
                  c: out std_logic);
         end component;

         signal ak,bk,ck: std_logic;

         begin

         and_gate: andgate port map(a=>ak,
b=>bk, c=>ck);

         process begin

                  ak <= '0';bk <= '0';wait for 1 ns;

                  ak <= '0';bk <= '1';wait for 1 ns;

                  ak <= '1';bk <= '0';wait for 1 ns;

                  ak <= '1';bk <= '1';wait for 1 ns;

                  assert false report "Completed
successfully";
                  wait;
         end process;
end test;
```

Output:



CONCLUSION:

The 2-input AND gate was successfully simulated. The output matched the expected truth table for all input combinations. The use of a simple testbench allowed for effective verification of the circuit behavior in a VHDL simulation environment.

LAB 2

OBJECTIVE: To design and simulate a 2-input OR gate using VHDL, and verify its functionality using a testbench.

TOOLS USED:

- VHDL (VHSIC Hardware Description Language)
- https://www.edaplayground.com/ (for simulation)

THEORY:

The aim of this lab is to create a simple 2-input OR gate in VHDL and simulate it using a testbench. The design code describes the logic of the OR gate, and the testbench applies all possible input combinations to validate the behavior of the gate.

VHDL CODE

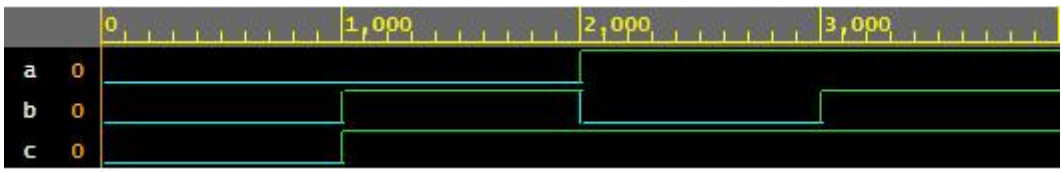
DESIGN:

```
library ieee;
use ieee.std_logic_1164.all;

entity orgate is
  port(  a,b:in std_logic;
         c: out std_logic);
end andgate;

architecture behavior of orgate is
         begin
         c <= a orb;
end behavior;
```

TESTBENCH

```
library ieee;
use ieee.std_logic_1164.all;

entity orgate_tb is
         ---> no content
end orgate_tb;

architecture test of orgate_tb is
         component orgate
         port(a,b: in std_logic;
                  c: out std_logic);
         end component;

         signal ak,bk,ck: std_logic;

         begin

         and_gate: orgate port map(a=>ak,
b=>bk, c=>ck);

         process begin

                  ak <= '0';bk <= '0';wait for 1 ns;

                  ak <= '0';bk <= '1';wait for 1 ns;

                  ak <= '1';bk <= '0';wait for 1 ns;

                  ak <= '1';bk <= '1';wait for 1 ns;

                  assert false report "Completed
successfully";
                  wait;
         end process;
end test;
```

Output:



CONCLUSION:

The 2-input OR gate was successfully simulated. The output matched the expected truth table for all input combinations. The use of a simple testbench allowed for effective verification of the circuit behavior in a VHDL simulation environment.

LAB 3

OBJECTIVE: To design and simulate a NOT gate using VHDL, and verify its functionality using a testbench.

TOOLS USED:

- VHDL (VHSIC Hardware Description Language)
- https://www.edaplayground.com/ (for simulation)

THEORY:

The aim of this lab is to create a NOT gate in VHDL and simulate it using a testbench. The design code describes the logic of the NOT gate, and the testbench applies both possible input combinations to validate the behavior of the gate.

VHDL CODE

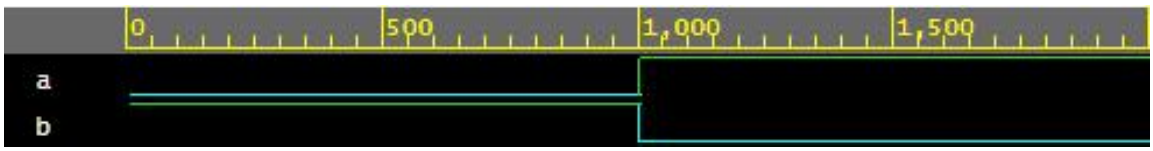
DESIGN:

```
library ieee;
use ieee.std_logic_1164.all;

entity notgate is
  port(  a: in std_logic;
         c: out std_logic);
end andgate;

architecture behavior of notgate is
         begin
         c <= not a;
end behavior;
```

TESTBENCH

```
library ieee;
use ieee.std_logic_1164.all;

entity notgate_tb is
         ---> no content
end notgate_tb;

architecture test of notgate_tb is
         component notgate
         port(a: in std_logic;
                  c: out std_logic);
         end component;

         signal ak,ck: std_logic;

         begin

    and_gate: notgate port map(a=>ak,c => ck );

         process begin

                  ak <= '0'; wait for 1 ns;
                  ak <= '1'; wait for 1 ns;

                  assert false report "Completed
successfully";
                  wait;
         end process;
end test;
```

Output:



CONCLUSION:

The NOT gate was successfully simulated. The output matched the expected behavior where ck is the inverse of ak for all input combinations. The use of a simple testbench allowed for effective verification of the circuit behavior in a VHDL simulation environment.

LAB 4

OBJECTIVE: To design and simulate a NAND gate using VHDL, and verify its functionality using a testbench.

TOOLS USED:

- VHDL (VHSIC Hardware Description Language)
- https://www.edaplayground.com/ (for simulation)

THEORY:

The aim of this lab is to create a NAND gate in VHDL and simulate it using a testbench. The design code describes the logic of the NAND gate, and the testbench applies both possible input combinations to validate the behavior of the gate.

VHDL CODE

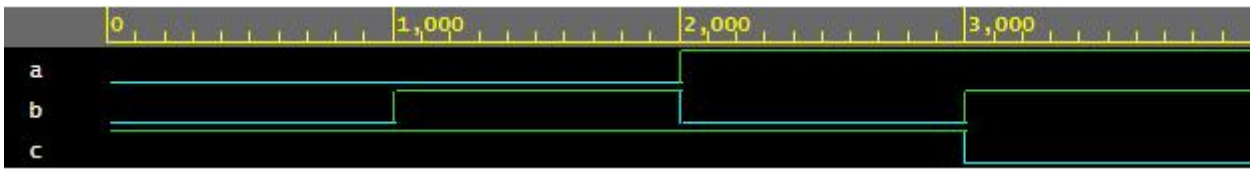
DESIGN

```
library ieee;
use ieee.std_logic_1164.all;

entity nandgate is
	port(
		a,b:in std_logic;
		c: out std_logic
	);
end nandgate;

architecture behavior of nandgate is
	begin
	c <= not (a and b);
end behavior;
```

TESTBENCH

```
library ieee;
use ieee.std_logic_1164.all;

entity nandgate_testbench is
end nandgate_testbench;

architecture test of nandgate_testbench is
	component nandgate
		port(
			a,b: in std_logic;
			c: out std_logic
		);
	end component;

	signal ak,bk,ck: std_logic;

	begin

	nand_gate: nandgate port map(a=>ak,
b=>bk, c=>ck);

	process begin

		ak <= '0';bk <= '0';wait for 1 ns;
		ak <= '0';bk <= '1';wait for 1 ns;
		ak <= '1';bk <= '0';wait for 1 ns;
		ak <= '1';bk <= '1';wait for 1 ns;

		assert false report "Completed
successfully";
		wait;
	end process;
end test;
```

Output:



CONCLUSION:

The NOT gate was successfully simulated. The output matched the expected behavior where ck is the inverse of ak for all input combinations. The use of a simple testbench allowed for effective verification of the circuit behavior in a VHDL simulation environment.

LAB 5

OBJECTIVE: To design and simulate a **Half Adder Circuit** using VHDL, and verify its functionality using a testbench.

TOOLS USED:

- VHDL (VHSIC Hardware Description Language)
- https://www.edaplayground.com/ (for simulation)

THEORY:

A Half Adder is a basic combinational circuit that adds two single-bit binary numbers. It produces two outputs:

- Sum (S): XOR of inputs A and B
- Carry (C): AND of inputs A and B

VHDL CODE

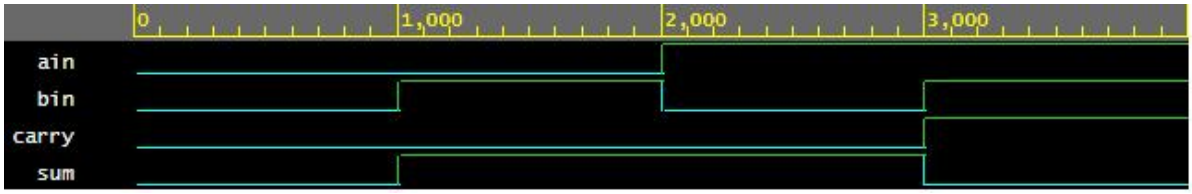
DESIGN

```
library ieee;
use ieee.std_logic_1164.all;

entity ha is
        port(   a: in std_ulogic;
                b: in std_ulogic;
                c: out std_ulogic;
                s: out std_ulogic);
end ha;

architecture behave of ha is
begin
        s <= a xor b;
        c <= a and b;
end behave;
```

TESTBENCH

```
library ieee;
use ieee.std_logic_1164.all;

entity ha_testbench is
end ha_testbench;

architecture test of ha_testbench is
        component ha
                port(a: in std_ulogic;
                        b: in std_ulogic;
                        c: out std_ulogic;
                        s: out std_ulogic);
        end component;

        signal ain, bin, carry, sum : std_logic;

begin
        half_adder: ha port map (a => ain, b =>
bin, c => carry, s => sum);

        process begin
        ain <= '0'; bin <= '0'; wait for 1 ns;
        ain <= '0'; bin <= '1'; wait for 1 ns;
        ain <= '1'; bin <= '0'; wait for 1 ns;
        ain <= '1'; bin <= '1'; wait for 1 ns;

assert false report "Reached end of test";wait;
        end process;
end test;
```

Output:



CONCLUSION:

The Half Adder was successfully implemented and simulated in VHDL. The outputs of the simulation matched the expected values based on the truth table of a half adder. The testbench applied all input combinations and validated the behavior effectively.

LAB 6

OBJECTIVE: To design and simulate a **Full Adder Circuit** using VHDL, and verify its functionality using a testbench.

TOOLS USED:

- VHDL (VHSIC Hardware Description Language)
- https://www.edaplayground.com/ (for simulation)

THEORY:

A Full Adder adds three one-bit binary numbers (A, B, and Carry-in) and outputs a Sum and a Carry-out. It performs the function:

- Sum (S) = A XOR B XOR Cin
- Carry-out (Cout) = (A XOR B) AND Cin OR (A AND B)

VHDL CODE

DESIGN

```
library ieee;
use ieee.std_logic_1164.all;

entity fa is
	port(
		a: in std_ulogic;
		b: in std_ulogic;
		cin: in std_ulogic;
		cout: out std_ulogic;
		s: out std_ulogic
	      );
end fa;

architecture	behave of fa is
begin
	s <= (a xor b) xor cin;
	cout <= ((a xor b) and cin) or (a and b);
end behave;
```

TESTBENCH

```
library ieee;
use ieee.std_logic_1164.all;

entity fa_testbench is
end fa_testbench;

architecture test of fa_testbench is
	component fa
		port(
			a: in std_ulogic;
			b: in std_ulogic;
			cin: in std_ulogic;
			cout: out std_ulogic;
			s: out std_ulogic
		);
	end component;

	signal ain, bin, cin, cout, sum : std_logic;

	begin
		full_adder: fa port map (a => ain, b => bin, cin => cin, cout => cout, s => sum);

		process begin
			ain <= '0';bin <= '0';cin <= '0';wait for 1 ns;
			ain <= '0';bin <= '0';cin <= '1';wait for 1 ns;
			ain <= '0';bin <= '1';cin <= '0';wait for 1 ns;
			ain <= '0';bin <= '1';cin <= '1';wait for 1 ns;
			ain <= '1';bin <= '0';cin <= '0';wait for 1 ns;
			ain <= '1';bin <= '0';cin <= '1';wait for 1 ns;
			ain <= '1';bin <= '1';cin <= '0';wait for 1 ns;
		ain <= '1';bin <= '1';cin <= '1';wait for 1 ns;
			assert false report "Reached end of test";
			wait;
		end process;
end test;
```

Output:

CONCLUSION:

The Full Adder was successfully implemented and simulated in VHDL. All possible input combinations were tested, and the output matched the expected values as per the truth table. This confirms the correct functionality of the Full Adder logic.