Experiment - 9.

Aim:- Write test bench for full adder circuit with VHDL code.

Software:- Model Sim.

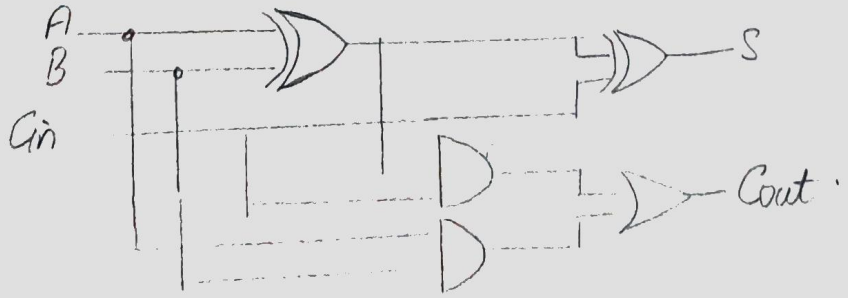
Theory:- A test bench is a model that is used to excersice and verify the correctness of a hardware model. A test bench is needed for three main purposes.

a) To generate stimulus for simulation (waveforms)
b) To apply this stimulus to the entity under test and to monitor the output responses.
c) To compose output responses with expected known values.

Steps to write a test bench:

1) The entity port list is always empty.
2) Under the architecture, all the components are declared for the under unit test (uut)
3) Inputs are declared & intialized to zero alongs with o/ps.
4) For sequential circuits, clock period is defined too.
5) UUT is initialized and process for stimulus is written for sequential designs, process for clock is coullen

Circuit Diagram:



Full adder.

BOOLEAN EXPRESSIONS:

C_{IN}	B	A	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = A \oplus B \oplus C$

$Cout = AB + BCin + ACin$

Result:- Test bench for full adder circuit was written and implemented in VHDL.

Codes

Full Adder vhdl code

```
library ieee;
use ieee.std_logic_1164.all;
entity full_adder is
    port (
        a  : in  std_logic;
        b  : in  std_logic;
        ci : in  std_logic;
        s  : out std_logic;
        co : out std_logic);
end;
architecture ADDER of full_adder is
begin
    s <= ((not a) and (not b) and ci) or (a and (not b) and (not ci)) or ((not
     a) and b and (not ci)) or (a and b and ci);
    co <= (a and b) or (b and ci) or (a and ci) ;
end;
```

Full Adder VHDL testbench

```
library ieee;
use ieee.std_logic_1164.all;
entity adder_tb is
end adder_tb;

architecture ADDER of adder_tb is
    component full_adder is
        port (
            a  : in  std_logic;
            b  : in  std_logic;
            ci : in  std_logic;
            s  : out std_logic;
            co : out std_logic);
    end component;
    signal input  : std_logic_vector(2 downto 0);
    signal output : std_logic_vector(1 downto 0);
begin
    uut: full_adder port map (
        a => input(2),
        b => input(1),
        ci => input(0),
        s => output(0),
        co => output(1)
    );

    stim_proc: process
    begin
        input <= "000"; wait for 10 ns; assert output = "00" report "0+0+0 failed";
        input <= "001"; wait for 10 ns; assert output = "01" report "0+0+1 failed";
        input <= "010"; wait for 10 ns; assert output = "01" report "0+1+0 failed";
        input <= "011"; wait for 10 ns; assert output = "10" report "0+1+1 failed";
        input <= "100"; wait for 10 ns; assert output = "01" report "1+0+0 failed";
        input <= "101"; wait for 10 ns; assert output = "10" report "1+0+1 failed";
        input <= "110"; wait for 10 ns; assert output = "10" report "1+1+0 failed";
        input <= "111"; wait for 10 ns; assert output = "11" report "1+1+1 failed";
        report "Full adder testbench finished";
        wait;
    end process;
end;
```

Output

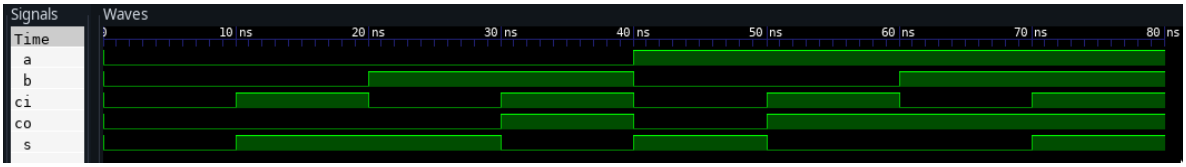


Figure 1: BASIC GATES

Result

Testbench for full adder circuit was written and implemented in VHDL