**VHDL CODE TEMPLATE FOR 7 LOGIC GATES**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity LogicGate is

Port (

A : in STD\_LOGIC;

B : in STD\_LOGIC; (\*remove this line only for INVERTER GATE)

Y : out STD\_LOGIC

);

end LogicGate;

architecture Behavioral of LogicGate is

begin (\*anyone from these green lines based on q)

Y <= A AND B; -- AND gate

Y <= NOT (A AND B); -- NAND gate

Y <= A OR B; -- OR gate

Y <= NOT (A OR B); -- NOR gate

Y <= A XOR B; -- XOR gate

Y <= NOT (A XOR B); -- XNOR gate

Y <= NOT A; -- INVERTER gate

end Behavioral;

.

**TESTBENCH CODE TEMPLATE FOR 7 LOGIC GATES**

library IEEE;

use IEEE.std\_logic\_1164.ALL;

entity gatebench is

end gatebench;

architecture tb of gatebench is

component LogicGate is

port (

a: in std\_logic;

b: in std\_logic; (\*remove this line only for single-input gates like NOT / INVERTER)

q: out std\_logic

);

end component;

signal a\_in, b\_in, q\_out : std\_logic;

begin

DUT: LogicGate port map(a\_in, b\_in, q\_out);

process

begin

-- Test case 1: a = 0, b = 0

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

-- Test case 2: a = 0, b = 1

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

-- Test case 3: a = 1, b = 0

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

-- Test case 4: a = 1, b = 1

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

wait;

end process;

end tb;