ASSIGNMENT 1

COMPUTER ORGANISATION AND ARCHITECTURE LAB (IT-2272)

Aditya Badayalya
IT 4th Semester
510819056(Hy)

QUESTION 1:

DESIGN AND SIMULATE THE BEHAVIOURAL MODELS OF THE FOLLOWING GATES:

1. AND GATE (2 INPUT):

VHDL Test Bench:

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

entity ANDGATE is
-- empty
end ANDGATE;

architecture tb of ANDGATE is

-- DUT component
component and_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: and_gate port map(a_in, b_in, q_out);

  process
  begin
    a_in <= '0';
    b_in <= '0';
    wait for 1 ns;
    assert(q_out='0') 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 1/0" 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;
```

VHDL MODULE:

```
--AND gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of and_gate is
begin
  process(a, b) is
  begin
    q <= a and b;
  end process;
end behaviour;
```

Simulation:


EPWave
From:
0ps
To:
4,000ps
Get Signals
Radix
100%
0
500
1,000
1,500
2,000
2,500
3,000
3,500
a_in
b_in
q_out
a
b
q
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

2. OR GATE (2 INPUT):

VHDL TEST BENCH:

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

entity ORGATE is
-- empty
end ORGATE;

architecture tb of ORGATE is

-- DUT component
component or_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: or_gate port map(a_in, b_in, q_out);

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

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

    a_in <= '1';
    b_in <= '0';
    wait for 1 ns;
```

```
    assert(q_out='1') report "Fail 1/0" 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;
```

VHDL MODULE:

```
-- OR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of or_gate is
begin
  process(a, b) is
  begin
    q <= a or b;
  end process;
end behaviour;
```

Simulation:


EPWave
From:
0ps
To:
4,000ps
Get Signals
Radix
100%
0
500
1,000
1,500
2,000
2,500
3,000
3,500
a_in
b_in
q_out
a
b
q
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

3. NOT GATE:

VHDL TEST BENCH:

```
-- Testbench for NOT gate
library IEEE;
use IEEE.std_logic_1164.all;

entity NOTGATE is
-- empty
end NOTGATE;

architecture tb of NOTGATE is

-- DUT component
component not_gate is
port(
  a: in std_logic;
  q: out std_logic);
end component;

signal a_in, q_out: std_logic;

begin

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

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

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

    -- Clear inputs
    a_in <= '0';

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

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

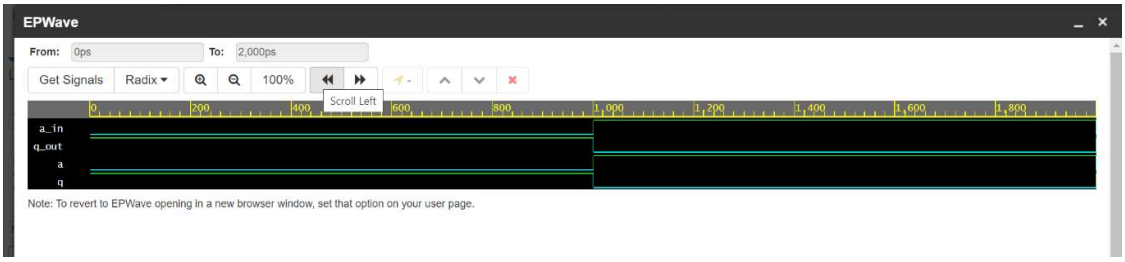
VDHL MODULE:

```
-- NOT gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of not_gate is
begin
 process(a) is
 begin
  q <= not a ;
 end process;
end behaviour;
```

Simulation:



QUESTION 2:

DESIGN AND SIMULATE THE BEHAVIOURAL MODELS OF UNIVERSAL GATES

(2 INPUT):

1. NOR GATE:

VHDL TEST BENCH:

```
-- Testbench for NOR gate
library IEEE;
use IEEE.std_logic_1164.all;

entity NORGATE is
--empty
end NORGATE;

architecture tb of NORGATE is

--DUT component

component nor_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: nor_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 1/0" severity error;

        a_in <= '1';
        b_in <='1';
        wait for 1 ns;
        assert(q_out='0') 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;
```

VHDL MODULE:

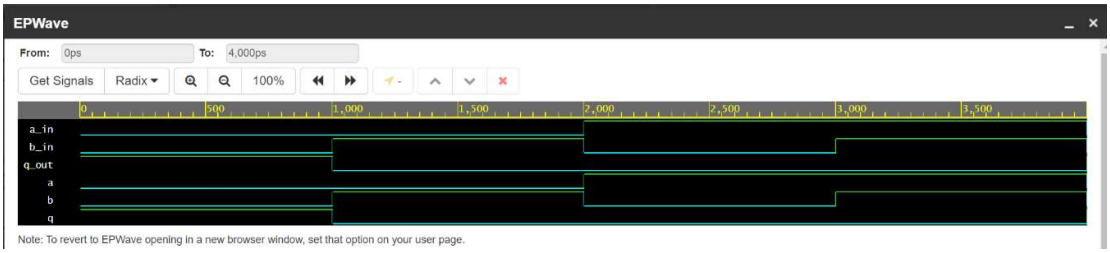
```
-- NOR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of nor_gate is
begin
    process(a,b) is
   begin
    q <= a nor b;
   end process;
```

```
end behaviour;
```

Simulation:



2. NAND GATE:

VHDL TEST BENCH:

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

entity NANDGATE is
--empty
end NANDGATE;

architecture tb of NANDGATE is

--DUT component

component nand_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: nand_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='1') report "Fail 0/1" severity error;
```

```
        a_in <= ‘1’;
        b_in <=’0’;
        wait for 1 ns;
        assert(q_out=’1’) report “Fail 1/0” severity error;

        a_in <= ‘1’;
        b_in <=’1’;
        wait for 1 ns;
        assert(q_out=’0’) 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;
```

VHDL MODULE:

```
-- NAND gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of nand_gate is
begin
        process(a,b) is
    begin
        q <= a nand b;
    end process;
end behaviour;
```

Simulation:


EPWave
From: 0ps
To: 4,000ps
Get Signals
Radix
100%
0ps
0
500
1,000
1,500
2,000
2,500
3,000
3,500
a_in 0
b_in 0
q_out 1
a 0
b 0
q 1
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

QUESTION 3:

DESIGN AND SIMULATE THE BEHAVIOURAL MODELS OF THE FOLLOWING GATES (2 INPUT):

1. XOR GATE:

VHDL TEST BENCH:

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

entity XORGATE is
--empty
end XORGATE;

architecture tb of XORGATE is

--DUT component

component xor_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: xor_gate port map(a_in, b_in, q_out);

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

        a_in <= '0';
```

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

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

        a_in <= '1';
        b_in <='1';
        wait for 1 ns;
        assert(q_out='0') 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;
```

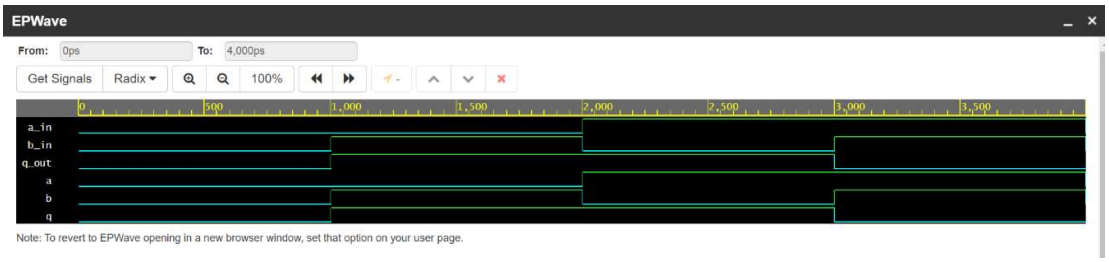
VHDL MODULE:

```
-- XOR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of xor_gate is
begin
        process(a,b) is
    begin
        q <= a xor b;
    end process;
end behaviour;
```

Simulation:


EPWave
From:
0ps
To:
4,000ps
Get Signals
Radix
100%
0
500
1,000
1,500
2,000
2,500
3,000
3,500
a_in
b_in
q_out
a
b
q
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

2. XNOR GATE:

VHDL TEST BENCH:

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

entity XNORGATE is
--empty
end XNORGATE;

architecture tb of XNORGATE 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 1/0" 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;
```

VHDL MODULE:

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

Simulation:


EPWave
From:
0ps
To:
4,000ps
Get Signals
Radix
100%
0
500
1,000
1,500
2,000
2,500
3,000
3,500
a_in
b_in
q_out
a
b
q
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

QUESTION 4:

DESIGN AND SIMULATE THE BEHAVIOURAL MODELS OF THE FOLLOWING 3 INPUT GATES:

1. AND GATE:

VHDL TEST BENCH:

```
-- Testbench for 3 input AND gate
library IEEE;
use IEEE.std_logic_1164.all;

entity ANDGATE is
-- empty
end ANDGATE;

architecture tb of ANDGATE is

-- DUT component
component and_gate is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, c_in, q_out: std_logic;

begin

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

  process
  begin
    a_in <= '0';
    b_in <= '0';
    c_in <= '0';
    wait for 1 ns;
```

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

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

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

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

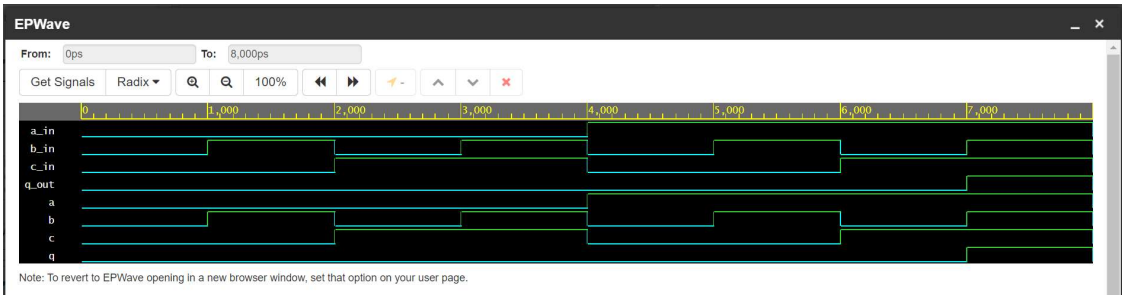
VHDL MODULE:

```
--3 input AND gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of and_gate is
begin
  process(a, b, c) is
  begin
    q <= a and b and c;
  end process;
end behaviour;
```

Simulation:



2. OR GATE:

VHDL TEST BENCH:

```
-- Testbench for 3 input OR gate
library IEEE;
use IEEE.std_logic_1164.all;

entity ORGATE is
-- empty
end ORGATE;

architecture tb of ORGATE is

-- DUT component
component or_gate is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, c_in, q_out: std_logic;

begin

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

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

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

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

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

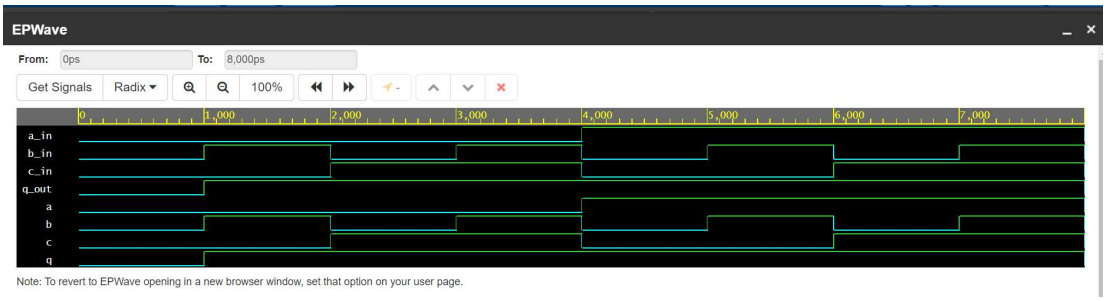
VHDL MODULE:

```
--3 input OR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of or_gate is
begin
  process(a, b, c) is
  begin
    q <= a or b or c;
  end process;
end behaviour;
```

Simulation:



3. NAND GATE:

VHDL TEST BENCH:

```
-- Testbench for 3 input NAND gate
library IEEE;
use IEEE.std_logic_1164.all;

entity NANDGATE is
-- empty
end NANDGATE;

architecture tb of NANDGATE is

-- DUT component
component nand_gate is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, c_in, q_out: std_logic;

begin

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

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

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

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

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

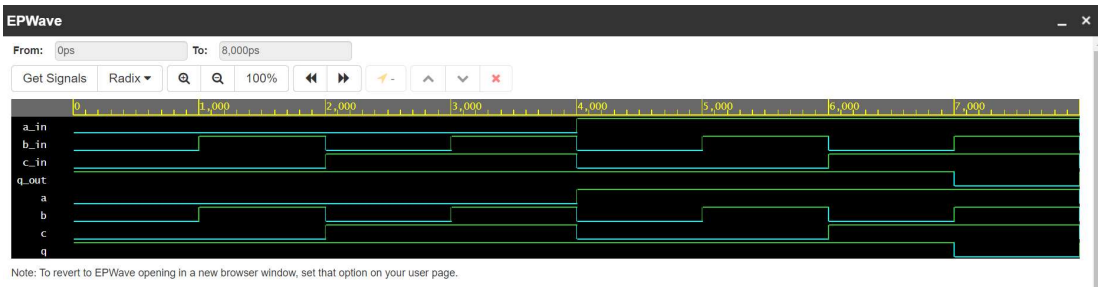
VHDL MODULE:

```
--3 input NAND gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of nand_gate is
begin
  process(a, b, c) is
  begin
    q <= not(a and b and c);
  end process;
end behaviour;
```

Simulation:



4. NOR GATE:

VHDL TEST BENCH:

```
-- Testbench for 3 input NOR gate
library IEEE;
use IEEE.std_logic_1164.all;

entity NORGATE is
-- empty
end NORGATE;

architecture tb of NORGATE is

-- DUT component
component nor_gate is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, c_in, q_out: std_logic;

begin

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

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

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

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

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

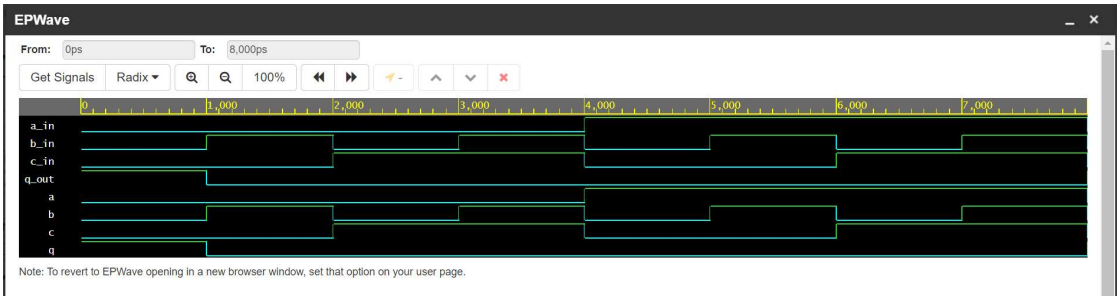
VHDL MODULE:

```
--3 input NOR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of nor_gate is
begin
  process(a, b, c) is
  begin
    q <= not(a or b or c);
  end process;
end behaviour;
```

Simulation:



5. XOR GATE:

VHDL TEST BENCH:

```
-- Testbench for 3 input XOR gate
library IEEE;
use IEEE.std_logic_1164.all;

entity XORGATE is
-- empty
end XORGATE;

architecture tb of XORGATE is

-- DUT component
component xor_gate is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, c_in, q_out: std_logic;

begin

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

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

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

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

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

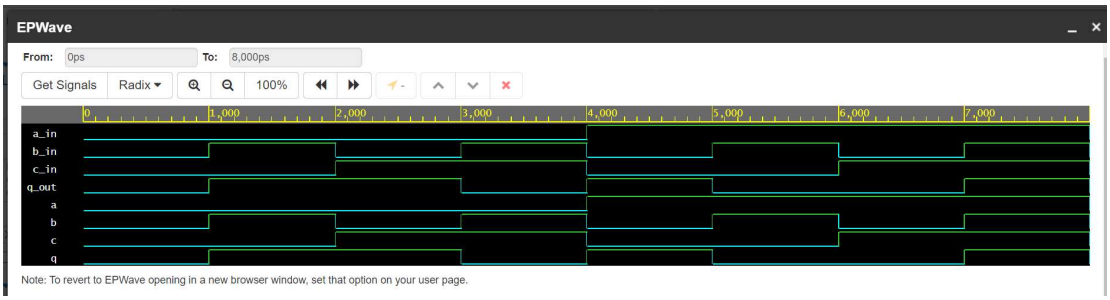
VHDL MODULE:

```
--3 input XOR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of xor_gate is
begin
  process(a, b, c) is
  begin
    q <= (a xor b xor c);
  end process;
end behaviour;
```

Simulation:



6. XNOR GATE:

VHDL TEST BENCH:

```
-- Testbench for 3 input XNOR gate
library IEEE;
use IEEE.std_logic_1164.all;

entity XNORGATE is
-- empty
end XNORGATE;

architecture tb of XNORGATE is

-- DUT component
component xnor_gate is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  q: out std_logic);
end component;

signal a_in, b_in, c_in, q_out: std_logic;

begin

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

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

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

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

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

VHDL MODULE:

```
--3 input XNOR gate design
library IEEE;
use IEEE.std_logic_1164.all;

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

architecture behaviour of xnor_gate is
begin
  process(a, b, c) is
  begin
    q <= not(a xor b xor c);
  end process;
end behaviour;
```

Simulation: