ASSIGNMENT 2

COMPUTER ORGANISATION AND ARCHITECTURE LAB (IT 2272)

Aditya Badayalya
IT 4th Semester
510819056 (Hy)

QUESTION 1:

DESIGN AND SIMULATE BEHAVIORAL MODEL OF A HALF ADDER:

VHDL Test Bench:

```
-- Testbench for HALF ADDER
library IEEE;
use IEEE.std_logic_1164.all;

entity HALFADDER is
-- empty
end HALFADDER;

architecture tb of HALFADDER is

-- DUT component
component half_adder is
port(
  a: in std_logic;
  b: in std_logic;
  carry: out std_logic;
  sum: out std_logic);
end component;

signal a_in, b_in, carry_out, sum_out: std_logic;

begin

  -- Connect DUT
  DUT: half_adder port map(a_in, b_in, carry_out, sum_out);

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

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

    -- Clear inputs
    a_in <= '0';
    b_in <= '0';
    assert false report "Test done." severity note;
    wait;
  end process;
end tb;
```

VHDL MODULE:

```
--HALF ADDER design
library IEEE;
use IEEE.std_logic_1164.all;

entity half_adder is
port(
  a: in std_logic;
  b: in std_logic;
  carry: out std_logic;
  sum: out std_logic);
end half_adder;

architecture behaviour of half_adder is
begin
  process(a, b) is
  begin
     sum <= a xor b;
    carry <= 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
carry_out
sum_out
a
b
carry
sum

QUESTION 2:

DESIGN AND SIMULATE BEHAVIORAL MODEL OF 1 BIT MAGNITUDE COMPARATOR:

VHDL Test Bench;

```
-- Testbench for 1 Bit Comparator
library IEEE;
use IEEE.std_logic_1164.all;

entity COMPARATOR is
-- empty
end COMPARATOR;

architecture tb of COMPARATOR is

-- DUT component
component single_comparator is
port(
  a: in std_logic;
  b: in std_logic;
  a_is_greater: out std_logic;
  Equal: out std_logic;
  b_is_greater: out std_logic);
end component;

signal a_in, b_in, G,E,L: std_logic;

begin

  -- Connect DUT
  DUT: single_comparator port map(a_in, b_in,G,E,L);

  process
  begin
    a_in <= '0';
    b_in <= '0';
    wait for 1 ns;
    assert(G='0' and E='1' and L='0') report "Fail 0/0"
severity error;
    a_in <= '0';
    b_in <= '1';
```

```
    wait for 1 ns;
    assert(G='0' and E='0' and L='1') report "Fail 0/0"
severity error;
    a_in <= '1';
    b_in <= '0';
    wait for 1 ns;
    assert(G='1' and E='0' and L='0') report "Fail 0/0"
severity error;
    a_in <= '1';
    b_in <= '1';
    wait for 1 ns;
    assert(G='0' and E='1' and L='0') report "Fail 0/0"
severity error;

    -- Clear inputs
    a_in <= '0';
    b_in <= '0';
    assert false report "Test done." severity note;
    wait;
  end process;
end tb;
```

VHDL MODULE:

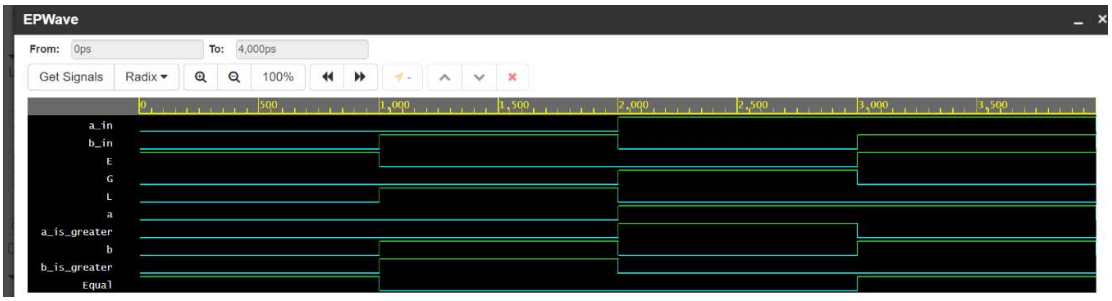
```
--1 Bit Comparator design
library IEEE;
use IEEE.std_logic_1164.all;

entity single_comparator is
port(
  a: in std_logic;
  b: in std_logic;
  a_is_greater: out std_logic;
  Equal: out std_logic;
  b_is_greater: out std_logic);
end single_comparator;

architecture behaviour of single_comparator is
begin
  process(a, b) is
  begin
     a_is_greater <= a and (not b);
    Equal <= not (a xor b);
```

```
    b_is_greater <= (not a)and b;
  end process;
end behaviour;
```

SIMULATION:



QUESTION 3:

DESIGN AND SIMULATE BEHAVIORAL MODEL OF A 4 BIT MAGNITUDE COMPARATOR:

VHDL Test Bench:

```
--VHDL testbench for 4-bit comparator
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity comparator4 is
end comparator4;

architecture behavior of comparator4 is
--signal declarations.
signal num1,num2 : std_logic_vector(3 downto 0) :=(others =>
'0');
signal less,equal,greater :  std_logic:='0';

begin
--entity instantiation
UUT : entity work.compare port
map(num1,num2,less,equal,greater);

--definition of simulation process
tb : process
begin
num1<="0010"; --num1 =2
num2<="1001"; --num2 =9
wait for 1 ns;

num1<="1001";  --num1 = 9
num2<="0010";   --num2 =2
wait for 1 ns;

num1<="0100";  --num1 =10
num2<="0100";  --num2 =10
--more input combinations can be given here.
wait for 1 ns;
wait;
end process tb;
```

```
end;
```

VHDL MODULE:

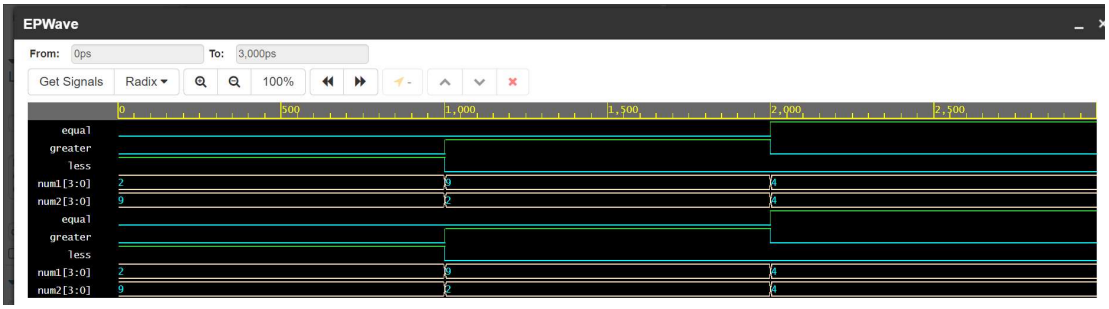
```
--VHDL module for 4-bit comparator
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity compare is port(
num1 : in std_logic_vector(3 downto 0); --first number
num2 : in std_logic_vector(3 downto 0); --second number
less : out std_logic; --first small
equal : out std_logic; --equal
greater : out std_logic --second small
);
end compare;

--architecture of entity

architecture Behavioral of compare is
begin
process(num1,num2)
begin
if(num1 > num2) then
less <= '0';
equal <= '0';
greater <= '1';
elsif (num1 < num2) then
less <= '1';
equal <= '0';
greater <= '0';
else
less <= '0';
equal <='1';
greater <='0';
end if;
end process;
end Behavioral;
```

SIMULATION:


EPWave
From:
0ps
To:
3,000ps
Get Signals
Radix
100%
0
500
1,000
1,500
2,000
2,500
equal
greater
less
num1[3:0]
num2[3:0]
equal
greater
less
num1[3:0]
num2[3:0]

QUESTION 4:

DESIGN AND SIMULATE BEHAVIORAL MODEL OF A 1-BIT FULL ADDER:

VHDL Test Bench:

```
-- Testbench for FULL ADDER
library IEEE;
use IEEE.std_logic_1164.all;

entity FULLADDER is
-- empty
end FULLADDER;

architecture tb of FULLADDER is

-- DUT component
component full_adder is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  carry: out std_logic;
  sum: out std_logic);
end component;

signal a_in, b_in, c_in, carry_out, sum_out: std_logic;

begin

  -- Connect DUT
  DUT: full_adder port map(a_in, b_in, c_in, carry_out,
sum_out);

  process
  begin
    a_in <= '0';
    b_in <= '0';
```

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

```
    a_in <= '1';
    b_in <= '1';
    c_in <= '1';
    wait for 1 ns;
    assert(carry_out='1' and sum_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;
```

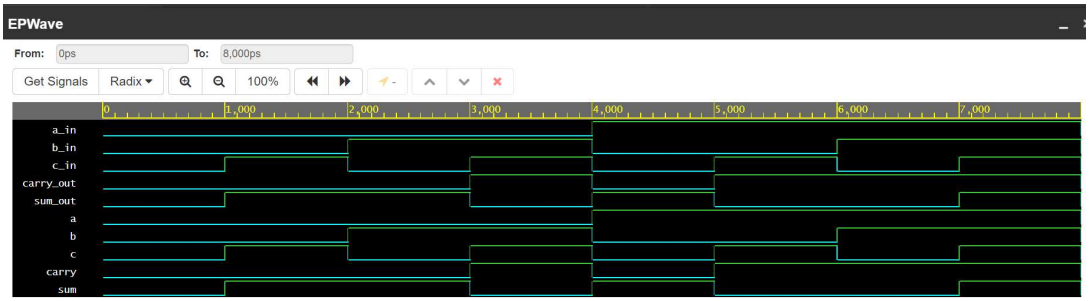
VHDL MODULE:

```
--FULL ADDER design
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
port(
  a: in std_logic;
  b: in std_logic;
  c: in std_logic;
  carry: out std_logic;
  sum: out std_logic);
end full_adder;

architecture behaviour of full_adder is
begin
  process(a, b, c) is
  begin
     sum <= c xor (a xor b);
    carry <= (a and b) or (a and c) or (c and b);
  end process;
end behaviour;
```

SIMULATION:


EPWave
From:
0ps
To:
8,000ps
Get Signals
Radix
100%
0
1,000
2,000
3,000
4,000
5,000
6,000
7,000
a_in
b_in
c_in
carry_out
sum_out
a
b
c
carry
sum