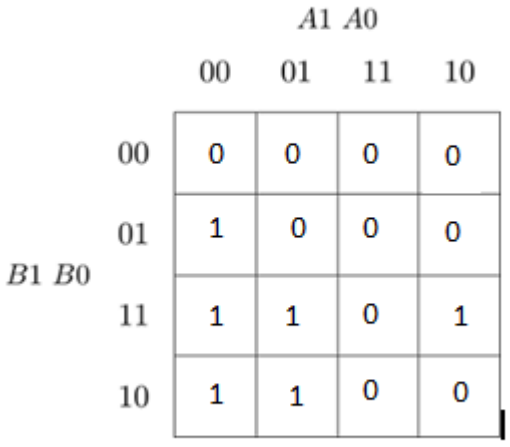
Lab 2: Combinational logic

1. 2-bit comparator truth table

Dec. equivalent	B[1:0]	A[1:0]	B is greater than A	B equals A	B is less than A
0	00	00	0	1	0
1	00	01	0	0	1
2	00	10	0	0	1
3	00	11	0	0	1
4	01	00	1	0	1
5	01	01	0	1	0
6	01	10	0	0	1
7	01	11	0	0	1
8	10	00	1	0	0
9	10	01	1	0	0
10	10	10	0	1	0
11	10	11	0	0	1
12	11	00	1	0	0
13	11	01	1	0	0
14	11	10	1	0	0
15	11	11	0	1	0

2. 2-bit comparator



Greater

$B1\ B0$ \ $A1\ A0$	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

Less

$$A < B = \overline{a1}\,\overline{a0}b0 + \overline{a1}b1 + \overline{a0}b1b0$$

$$A > B = a1\overline{b1} + a0\overline{b1}\,\overline{b0} + a1a0\overline{b0}$$

Equations

Link to EDA Playground

3. 4-bit comparator

```
----------------------------------------------------------------------------------------------------------------
--
-- Example of 4-bit binary comparator using the when/else assignment.
-- EDA Playground
--
-- Copyright (c) 2020-2021 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.
--
----------------------------------------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

----------------------------------------------------------------------------------------------------------------
-- Entity declaration for 2-bit binary comparator
----------------------------------------------------------------------------------------------------------------
entity comparator_4bit is
```

```
    port(
        a_i           : in  std_logic_vector(4 - 1 downto 0);
        b_i           : in std_logic_vector(4 - 1 downto 0);
        B_greater_A_o : out std_logic;      -- B is greater than A
        B_equals_A_o  : out std_logic;      -- B equals A
        B_less_A_o    : out std_logic       -- B is less than A
    );
end entity comparator_4bit;

------------------------------------------------------------------------------------------------------------------------
-- Architecture body for 2-bit binary comparator
------------------------------------------------------------------------------------------------------------------------
architecture Behavioral of comparator_4bit is
begin
    B_less_A_o <= '1' when (b_i < a_i) else '0';
    B_greater_A_o <= '1' when (b_i > a_i) else '0';
    B_equals_A_o  <= '1' when (b_i = a_i) else '0';
    -- WRITE "GREATER" AND "EQUALS" ASSIGNMENTS HERE


end architecture Behavioral;




------------------------------------------------------------------------------------------------------------------------
--
-- Testbench for 4-bit binary comparator.
-- EDA Playground
--
-- Copyright (c) 2020-2021 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.
--
------------------------------------------------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

------------------------------------------------------------------------------------------------------------------------
-- Entity declaration for testbench
------------------------------------------------------------------------------------------------------------------------
entity tb_comparator_4bit is
    -- Entity of testbench is always empty
end entity tb_comparator_4bit;

------------------------------------------------------------------------------------------------------------------------
-- Architecture body for testbench
------------------------------------------------------------------------------------------------------------------------
architecture testbench of tb_comparator_4bit is

    -- Local signals
    signal s_a : std_logic_vector(4 - 1 downto 0);
    signal s_b : std_logic_vector(4 - 1 downto 0);
    signal s_B_greater_A : std_logic;
    signal s_B_equals_A : std_logic;
    signal s_B_less_A   : std_logic;

begin
    -- Connecting testbench signals with comparator_4bit entity (Unit Under Test)
    uut_comparator_4bit : entity work.comparator_4bit
        port map(
            a_i           => s_a,
            b_i           => s_b,
            B_greater_A_o => s_B_greater_A,
            B_equals_A_o  => s_B_equals_A,
```

```
        B_less_A_o    => s_B_less_A
    );

-----------------------------------------------------------------------------------------------------
-- Data generation process
-----------------------------------------------------------------------------------------------------
p_stimulus : process
begin
    -- Report a note at the beginning of stimulus process
    report "Stimulus process started" severity note;


    -- First test values
    s_b <= "0000";
    s_a <= "0000";
    wait for 100 ns;

    -- Expected output
    assert ((s_B_greater_A = '0') and (s_B_equals_A = '1') and           (s_B_less_A = '0'))
    -- If false, then report an error
    report "Test failed for input combination: 0000, 0000" severity error;

    -- WRITE OTHER TEST CASES HERE


    s_b <= "0001";
    s_a <= "0010";
    wait for 100 ns;
    -- Expected output
    assert ((s_B_greater_A = '0') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
    -- If false, then report an error
    report "Test failed for input combination: 0001, 0010" severity         error;

    s_b <= "0001";
    s_a <= "0011";
    wait for 100 ns;
    -- Expected output
    assert ((s_B_greater_A = '0') and (s_B_equals_A = '0') and          (s_B_less_A = '1'))
    -- If false, then report an error
    report "Test failed for input combination: 0001, 0011" severity         error;

    s_b <= "1001";
    s_a <= "0100";
    wait for 100 ns;
    -- Expected output
    assert ((s_B_greater_A = '1') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
    -- If false, then report an error
    report "Test failed for input combination: 0001, 0100" severity         error;



    s_b <= "0010";
    s_a <= "0010";
    wait for 100 ns;
    -- Expected output
    assert ((s_B_greater_A = '0') and (s_B_equals_A = '1') and          (s_B_less_A = '0'))
    -- If false, then report an error
    report "Test failed for input combination: 0010, 1010" severity         error;


    s_b <= "1111";
    s_a <= "0000";
    wait for 100 ns;
    -- Expected output
    assert ((s_B_greater_A = '1') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
    -- If false, then report an error
```

```
        report "Test failed for input combination: 00, 00" severity         error;

        s_b <= "0100";
        s_a <= "0110";
        wait for 100 ns;
        -- Expected output
        assert ((s_B_greater_A = '0') and (s_B_equals_A = '0') and          (s_B_less_A = '1'))
        -- If false, then report an error
        report "Test failed for input combination: 00, 00" severity         error;


        s_b <= "0110";
        s_a <= "0011";
        wait for 100 ns;
        -- Expected output
        assert ((s_B_greater_A = '1') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
        -- If false, then report an error
        report "Test failed for input combination: 00, 00" severity         error;


        s_b <= "1000";
        s_a <= "0100";
        wait for 100 ns;
        -- Expected output
        assert ((s_B_greater_A = '1') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
        -- If false, then report an error
        report "Test failed for input combination: 00, 00" severity         error;

        s_b <= "1000";
        s_a <= "0101";
        wait for 100 ns;
        -- Expected output
        assert ((s_B_greater_A = '1') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
        -- If false, then report an error
        report "Test failed for input combination: 00, 00" severity         error;


        s_b <= "1000";
        s_a <= "0110";
        wait for 100 ns;
        -- Expected output
        assert ((s_B_greater_A = '1') and (s_B_equals_A = '0') and          (s_B_less_A = '0'))
        -- If false, then report an error
        report "Test failed for input combination: 00, 00" severity         error;

        s_b <= "1111";
        s_a <= "1111";
        wait for 100 ns;
        -- Expected output
        assert ((s_B_greater_A = '0') and (s_B_equals_A = '1') and          (s_B_less_A = '0'))
        -- If false, then report an error
        report "Test failed for input combination: 00, 00" severity         error;

        -- Report a note at the end of stimulus process
        report "Stimulus process finished" severity note;
        wait;
    end process p_stimulus;

end architecture testbench;
```

[2021-02-23 15:52:57 EST] ghdl -i design.vhd testbench.vhd && ghdl -m tb_comparator_4bit && ghdl -r tb_comparator_4bit
analyze design.vhd analyze testbench.vhd elaborate tb_comparator_4bit testbench.vhd:51:9:@0ms:(report note): Stimulus process started
testbench.vhd:71:9:@200ns:(assertion error): Test failed for input combination: 0001, 0010 testbench.vhd:162:9:@1200ns:(report note): Stimulus process finished
Done

4-bits comparator