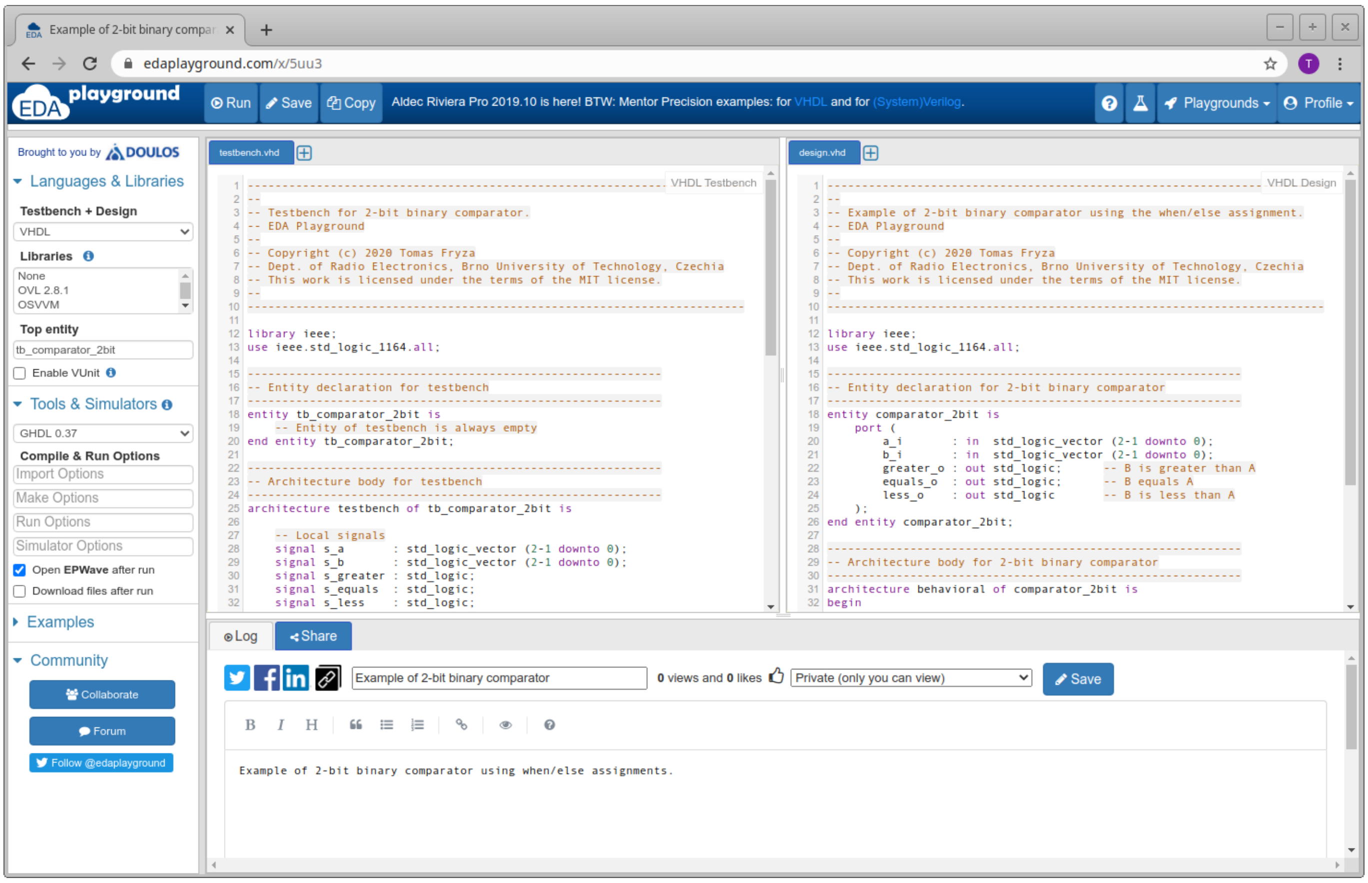
Lab 2: Combinational logic

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

Learning objectives

The purpose of this laboratory exercise is to learn to use different ways of writing combination functions (truth table, K-map, SoP/PoS forms), their minimization, the use of signal assignments in VHDL, and assertion statements in VHDL testbench.



Testbench + Design: VHDL
Libraries: None
Top entity: tb_comparator_2bit
[ ] Enable VUnit
GHDL 0.37
Compile & Run Options
[✓] Open EPWave after run
[ ] Download files after run

testbench.vhd:

```vhdl
------------------------------------------------------------------------
--
-- Testbench for 2-bit binary comparator.
-- EDA Playground
--
-- Copyright (c) 2020 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_2bit is
    -- Entity of testbench is always empty
end entity tb_comparator_2bit;

------------------------------------------------------------
-- Architecture body for testbench
------------------------------------------------------------
architecture testbench of tb_comparator_2bit is

    -- Local signals
    signal s_a       : std_logic_vector (2-1 downto 0);
    signal s_b       : std_logic_vector (2-1 downto 0);
    signal s_greater : std_logic;
    signal s_equals  : std_logic;
    signal s_less    : std_logic;
```

design.vhd:

```vhdl
------------------------------------------------------------------------
--
-- Example of 2-bit binary comparator using the when/else assignment.
-- EDA Playground
--
-- Copyright (c) 2020 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_2bit is
    port (
        a_i       : in  std_logic_vector (2-1 downto 0);
        b_i       : in  std_logic_vector (2-1 downto 0);
        greater_o : out std_logic;      -- B is greater than A
        equals_o  : out std_logic;      -- B equals A
        less_o    : out std_logic       -- B is less than A
    );
end entity comparator_2bit;

------------------------------------------------------------
-- Architecture body for 2-bit binary comparator
------------------------------------------------------------
architecture behavioral of comparator_2bit is
begin
```

Example of 2-bit binary comparator
Private (only you can view)

```
Example of 2-bit binary comparator using when/else assignments.
```

Preparation tasks (done before the lab at home)

Digital or *Binary comparator* compares the digital signals A, B presented at input terminal and produce outputs depending upon the condition of those inputs. Complete the truth table for 2-bit *Identity comparator* (B equals A), and two *Magnitude comparators* (B is greater than A, B is less than A). Note that, such a digital device has four inputs and three outputs/functions.

Dec. equivalent	B[1:0]	A[1:0]	B is greater than A	B equals A	B is less than A

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

According to the truth table, write canonical SoP (Sum of Products) and PoS (Product of Sums) forms for "equals" and "less than" functions:

$$equals\_SoP =$$
$$less\_PoS =$$

Part 1: Synchronize Git and create a new folder

When you start working, always synchronize the contents of your working folder and local repository with remote version at GitHub. This way you are sure that you will not lose any of your changes.

Run Git Bash (Windows) of Terminal (Linux) and synchronize local and remote repositories.

```
## Windows Git Bash:
$ cd d:/Documents/
$ cd your-name/
$ ls
Digital-electronics-1/
$ cd Digital-electronics-1/
$ git pull

## Linux:
```

```
$ cd
$ cd Documents/
$ cd your-name/
$ ls
Digital-electronics-1/
$ cd Digital-electronics-1/
$ git pull
```

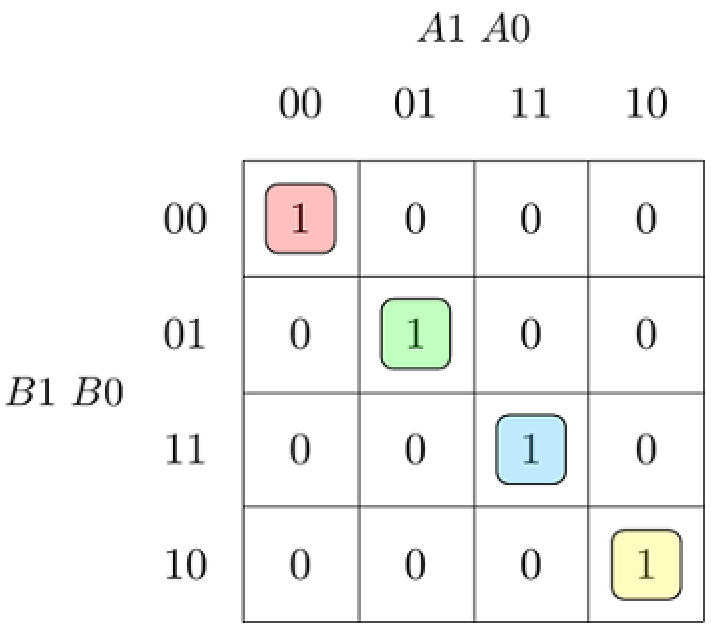
Create a new working folder `Labs/02-logic` for this exercise.

```
## Windows Git Bash or Linux:
$ cd Labs/
$ mkdir 02-logic
```

Part 2: Logic function minimization

Karnaugh Maps (or K-maps) offer a graphical method of reducing a digital circuit to its minimum number of gates. The map is a simple table containing 1s and 0s that can express a truth table or complex Boolean expression describing the operation of a digital circuit.

The K-map for the "equals" function is as follows:



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

Create K-maps for other two functions.

Use K-maps to create a simplified SoP form of the "greater than" function and a PoS form of the "less than" function.

$$greater\_SoP\_min =$$
$$less\_PoS\_min =$$

Part 3: Binary comparator in VHDL language

Log in to your EDA Playground account and create a new project: you can copy your previous playground and save it under a different name.

In VHDL, define an entity for a 2-bit binary comparator (`comparator_2bit`).

Port name	Direction	Type	Description
`a_i`	input	`std_logic_vector(2 - 1 downto 0)`	Data A
`b_i`	input	`std_logic_vector(2 - 1 downto 0)`	Data B
`greater_o`	output	`std_logic`	B is greater than A
`equals_o`	output	`std_logic`	B equals A
`less_o`	output	`std_logic`	B is less than A

In VHDL, define an architecture for a 2-bit binary comparator. The combination logic can be written using low-level operators (`and`, `or`, etc.) as in the previous laboratory exercise. However, it is more efficient to use a higher notation with signal assignments. Use the assignment `when`, `else` to describe the three output functions, such as:

```
greater_o <= '1' when (b_i > a_i) else '0';
```

Part 4: Assertion statements in VHDL testbench

You can write any information to the console using the report statement. The basic syntax in VHDL is:

```
report <message_string> [severity <severity_level>];
```

where possible values for `severity_level` are: `note`, `warning`, `error`, `failure`. If the severity level is omitted, then the default value is `note`.

An assertion statement checks that a specified condition is true and reports an error if it is not. It is combined with a report statement as follows:

```
assert (<condition>)
report <message_string>
```

```
[severity <severity_level>];
```

The message is displayed to the console when the condition is NOT met, therefore the message should be an opposite to the condition.

```
    ------------------------------------------------------------------
    -- Data generation process
    ------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

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

        -- ADD OTHER TEST CASES

        report "Stimulus process finished" severity note;
        wait;
    end process p_stimulus;
```

In VHDL, write a testbench and verify the correct functionality of the comparator for all input combinations.

Update your local (not GitHub) `README.md` file with a screenshot of the simulation(s) and a link to your public EDA playground.

Synchronize git

Use git commands to add, commit, and push all local changes to your remote repository. Check the repository at GitHub web page for changes.

Experiments on your own

1. In EDA Playground, define entity and architecture for a 4-bit binary comparator (`comparator_4bit`).

Port name	Direction	Type	Description
`a_i`	input	`std_logic_vector(4 - 1 downto 0)`	Data A
`b_i`	input	`std_logic_vector(4 - 1 downto 0)`	Data B
`greater_o`	output	`std_logic`	B is greater than A
`equals_o`	output	`std_logic`	B equals A

Port name	Direction	Type	Description
`less_o`	output	`std_logic`	B is less than A

2. In VHDL, define a testbench for a 4-bit binary comparator. Verify at least ten random input combinations.

Lab assignment

1. Preparation tasks (done before the lab at home). Submit:

    - Binary comparator truth table.

2. A 4-bit binary comparator. Submit:

    - VHDL code (`design.vhd`),
    - VHDL testbench (`testbench.vhd`),
    - Screenshot with simulated time waveforms,
    - Link to your public EDA Playground example.

The deadline for submitting the task is the day before the next laboratory exercise. Use BUT e-learning web page and submit a single PDF file.