OF THE
DEPARTMENT OF
INFORMATION ENGINEERING

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

VHDL: Exercises on Simple Logic Structures

Marta Bagatin, marta.bagatin@unipd.it

Degree Course in Information Engineering

Academic Year 2023-2024

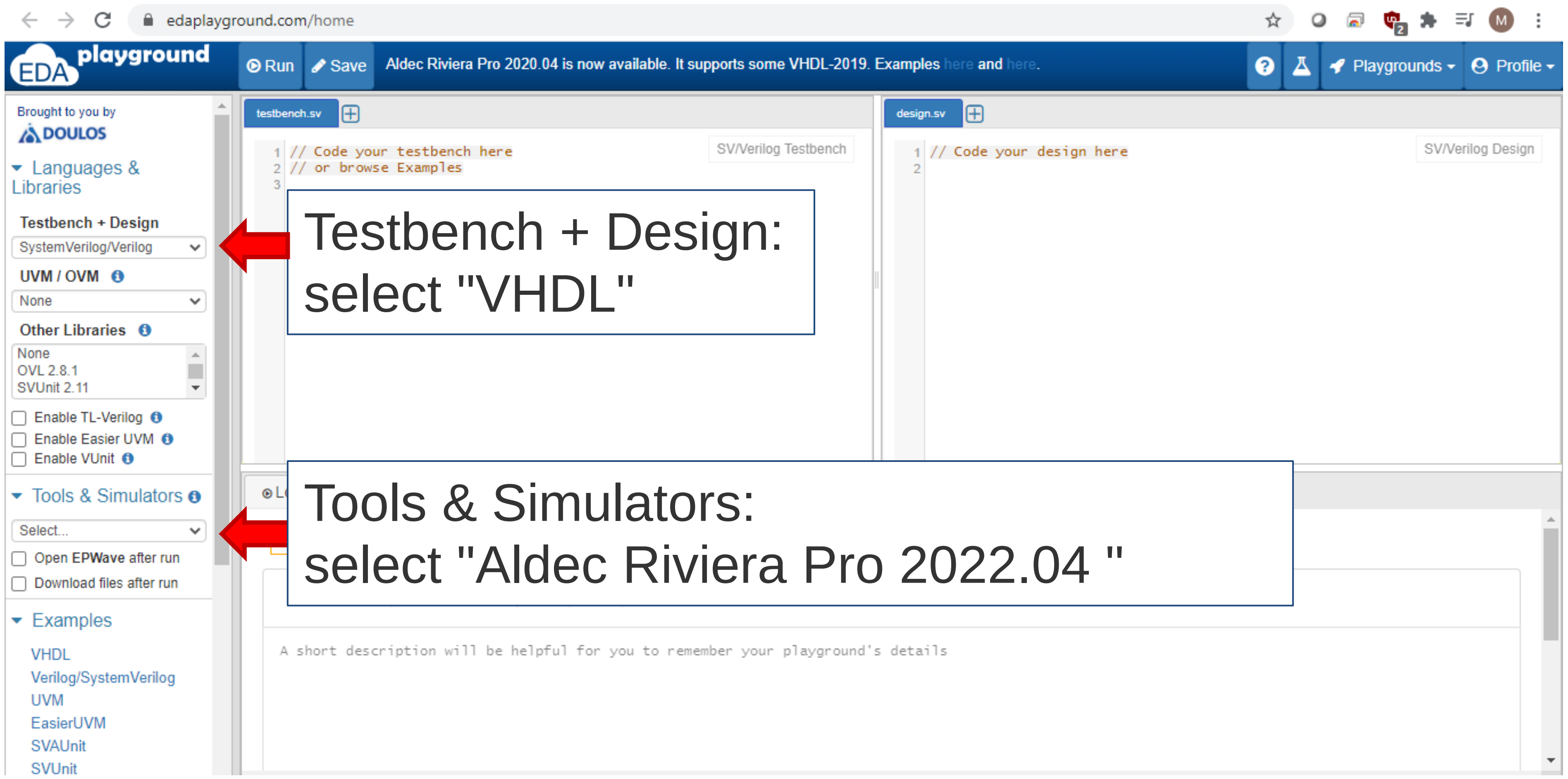
Purpose of the Lesson

- Describe simple logic circuits with **VHDL code**
- Introduce the **EDA Playground VHDL simulator** and simulate the operation of simple logic structures

EDA Playground

- Online simulator
  - https://www.edaplayground.com/
  - EDA = Electronic Design Automation
  - No software to be installed, you just need an account (e.g. gmail)
  - You can work from any PC once you have the account
  - You don't need a computer with great computing power

EDA Playground



```
// Code your testbench here
// or browse Examples
```

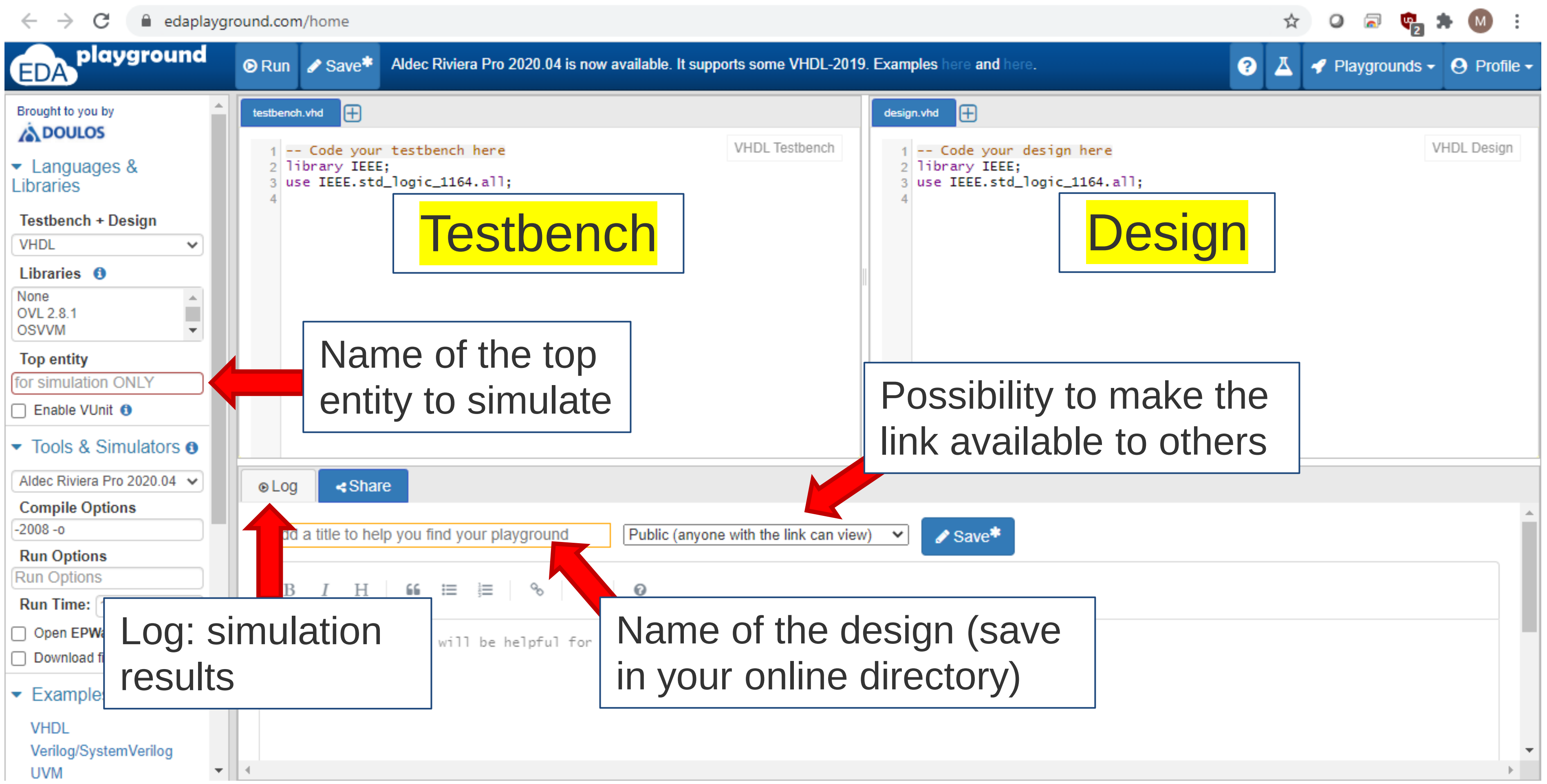
```
// Code your design here
```

A short description will be helpful for you to remember your playground's details

Testbench + Design:
select "VHDL"

Tools & Simulators:
select "Aldec Riviera Pro 2022.04 "

EDA Playground

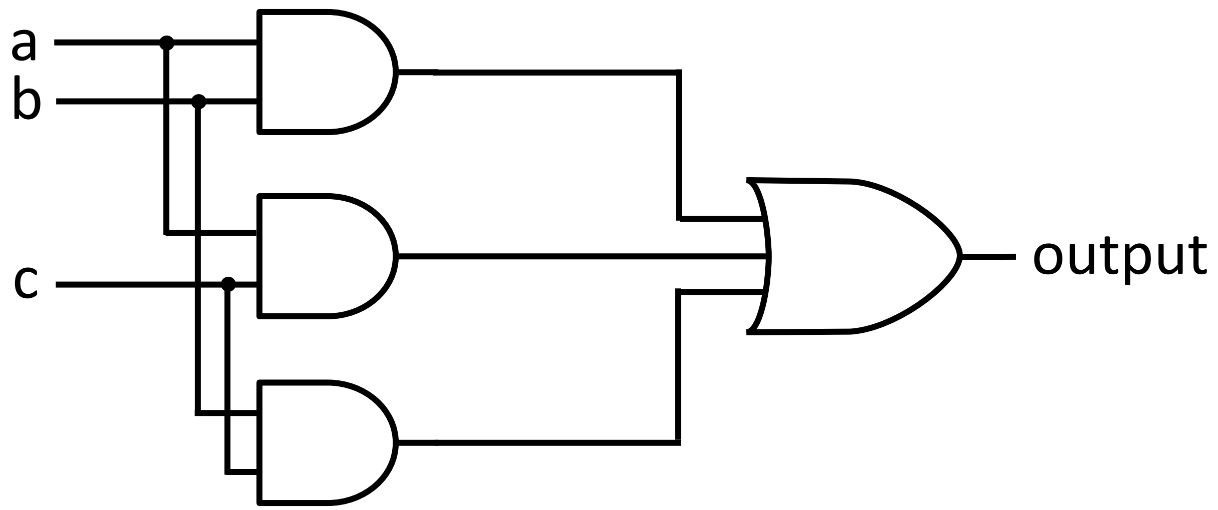


```
-- Code your testbench here
library IEEE;
use IEEE.std_logic_1164.all;
```

```
-- Code your design here
library IEEE;
use IEEE.std_logic_1164.all;
```

Ex: Majority Function

1) Describe in VHDL the circuit that describes the majority function
   - 3 inputs (1 bit) a, b, c
   - 1 output (1 bit): output = '1’ if at least two of the inputs are equal to '1'

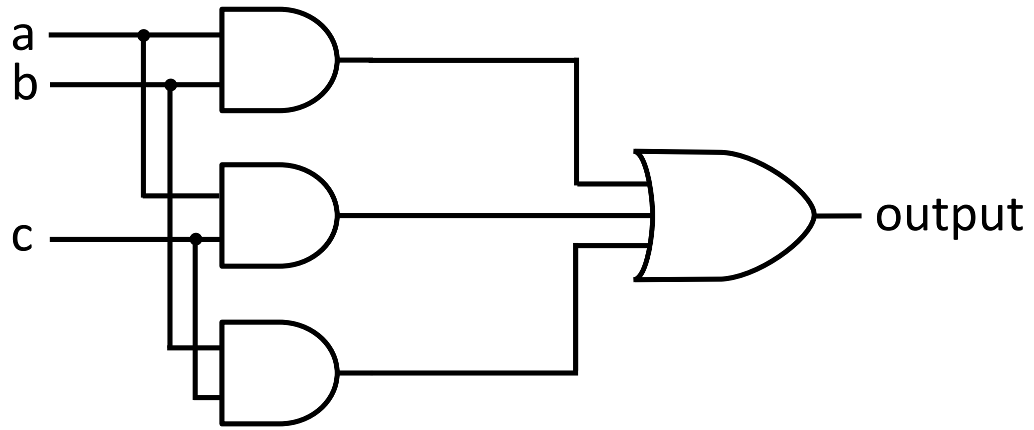


2) Simulate the operation of the circuit with EDA Playground with an arbitrary combination of the inputs

Majority Function: Design

1) Describe in VHDL the circuit that describes the majority function

```
library ieee;
use ieee.std_logic_1164.all;

entity majority is
        port ( a, b, c : in std_logic;
                    output : out std_logic );
end majority;

architecture maj_dataflow of majority is
begin
        output <= (a and b) or (a and c) or (b and c);
end maj_dataflow;
```



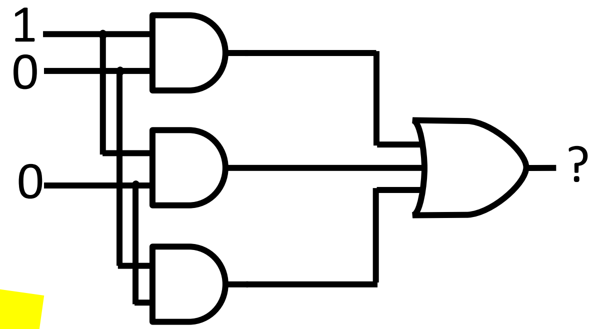
Logic operators AND and OR have the same precedence in VHDL (brackets are needed). If no brackets, the functions are evaluated from left to right

Majority Function: Testbench

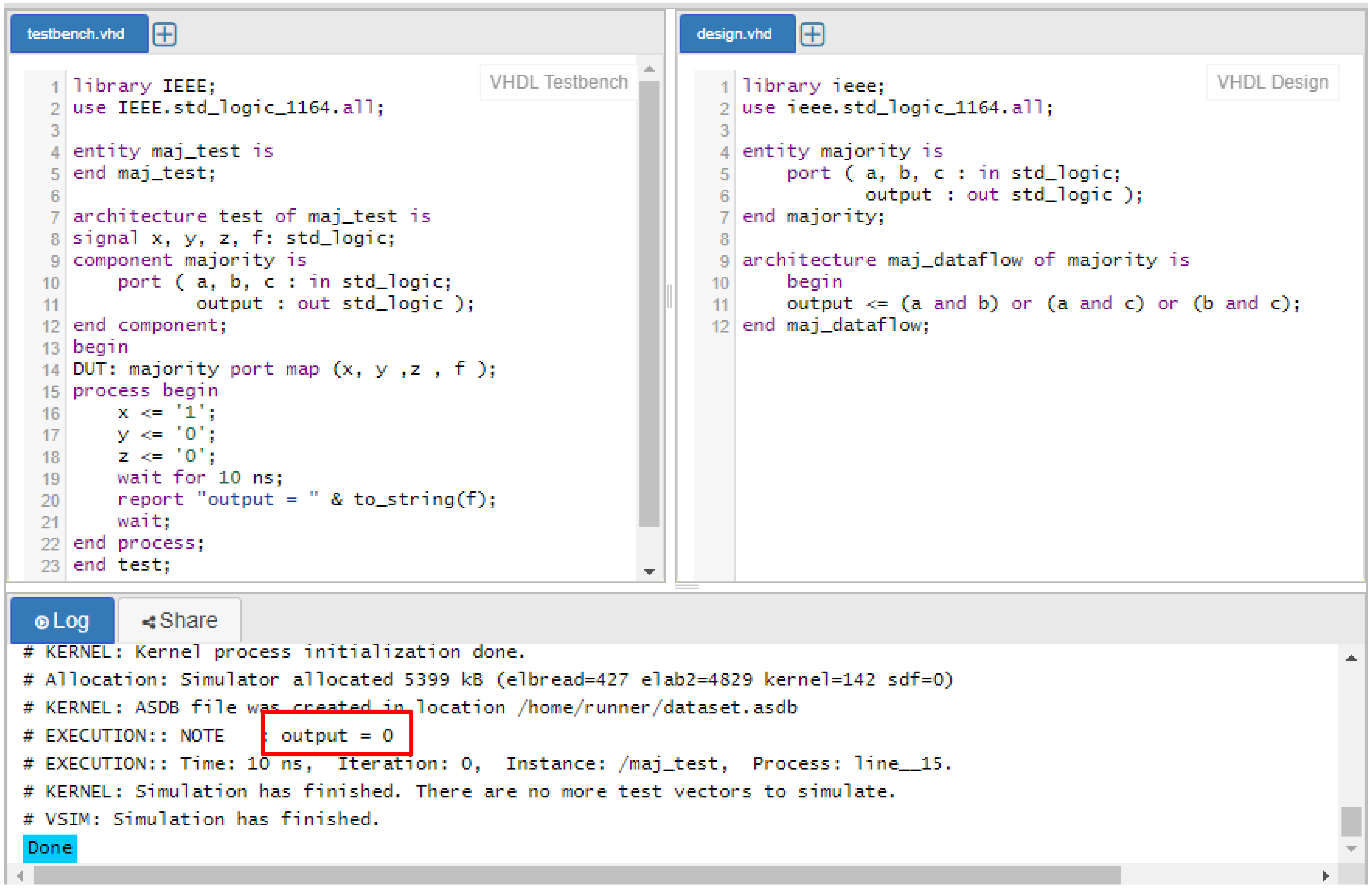
```
library IEEE;
use IEEE.std_logic_1164.all;

entity maj_test is
end maj_test;

architecture test of maj_test is
signal x, y, z, f: std_logic;

component majority is
    port ( a, b, c : in std_logic;
                output : out std_logic );
end component;
begin
DUT: majority port map (x, y, z, f );
process begin
    x <= '1'; -apply the stimuli to the DUT
    y <= '0';
    z <= '0';
    wait for 10 ns; -delay allows the output to be stable
    report "output =" & to_string(f);- displays the output value
    wait;                              - function to_string() converts vector to string
end process;
end test;
```

2) Simulate the operation of the circuit with EDA Playground with an arbitrary combination of the inputs

Instantiation of components inside architecture (declaration is needed)



Majority Function: EDA Playground

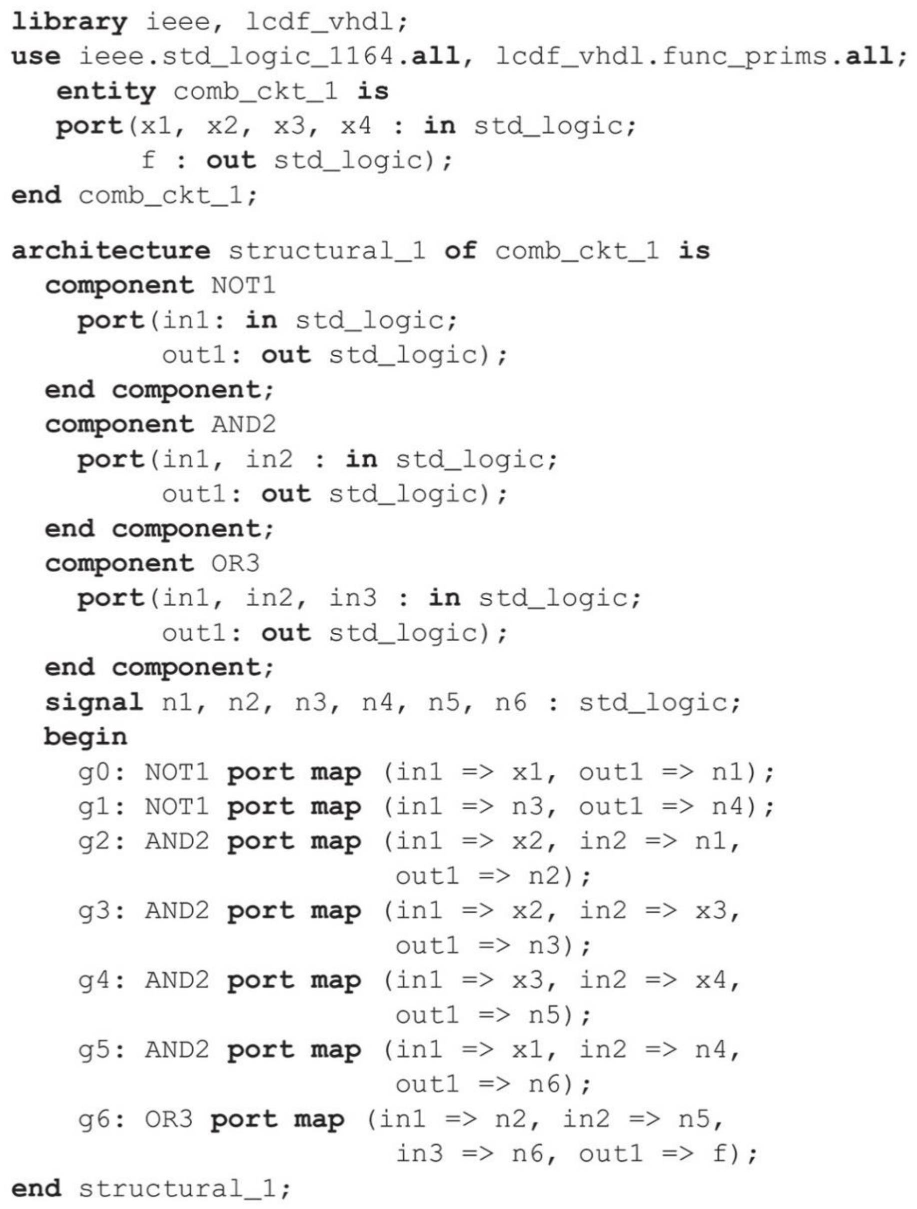


```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity maj_test is
end maj_test;

architecture test of maj_test is
signal x, y, z, f: std_logic;
component majority is
    port ( a, b, c : in std_logic;
           output : out std_logic );
end component;
begin
DUT: majority port map (x, y ,z , f );
process begin
    x <= '1';
    y <= '0';
    z <= '0';
    wait for 10 ns;
    report "output = " & to_string(f);
    wait;
end process;
end test;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity majority is
    port ( a, b, c : in std_logic;
           output : out std_logic );
end majority;

architecture maj_dataflow of majority is
    begin
    output <= (a and b) or (a and c) or (b and c);
end maj_dataflow;
```

```
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 5399 kB (elbread=427 elab2=4829 kernel=142 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# EXECUTION:: NOTE    : output = 0
# EXECUTION:: Time: 10 ns,  Iteration: 0,  Instance: /maj_test,  Process: line__15.
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
Done
```

Exercise 2.34 (Textbook Mano-Kime)



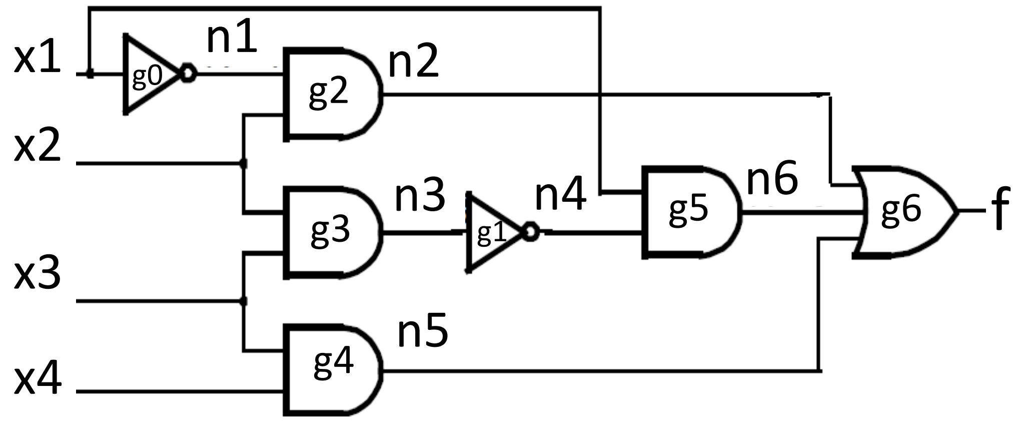
```
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
   entity comb_ckt_1 is
   port(x1, x2, x3, x4 : in std_logic;
         f : out std_logic);
end comb_ckt_1;

architecture structural_1 of comb_ckt_1 is
  component NOT1
    port(in1: in std_logic;
         out1: out std_logic);
  end component;
  component AND2
    port(in1, in2 : in std_logic;
         out1: out std_logic);
  end component;
  component OR3
    port(in1, in2, in3 : in std_logic;
         out1: out std_logic);
  end component;
  signal n1, n2, n3, n4, n5, n6 : std_logic;
  begin
    g0: NOT1 port map (in1 => x1, out1 => n1);
    g1: NOT1 port map (in1 => n3, out1 => n4);
    g2: AND2 port map (in1 => x2, in2 => n1,
                       out1 => n2);
    g3: AND2 port map (in1 => x2, in2 => x3,
                       out1 => n3);
    g4: AND2 port map (in1 => x3, in2 => x4,
                       out1 => n5);
    g5: AND2 port map (in1 => x1, in2 => n4,
                       out1 => n6);
    g6: OR3 port map (in1 => n2, in2 => n5,
                       in3 => n6, out1 => f);
end structural_1;
```

1) Draw the logic diagram of the circuit described in this VHDL code (the complemented inputs are not available)

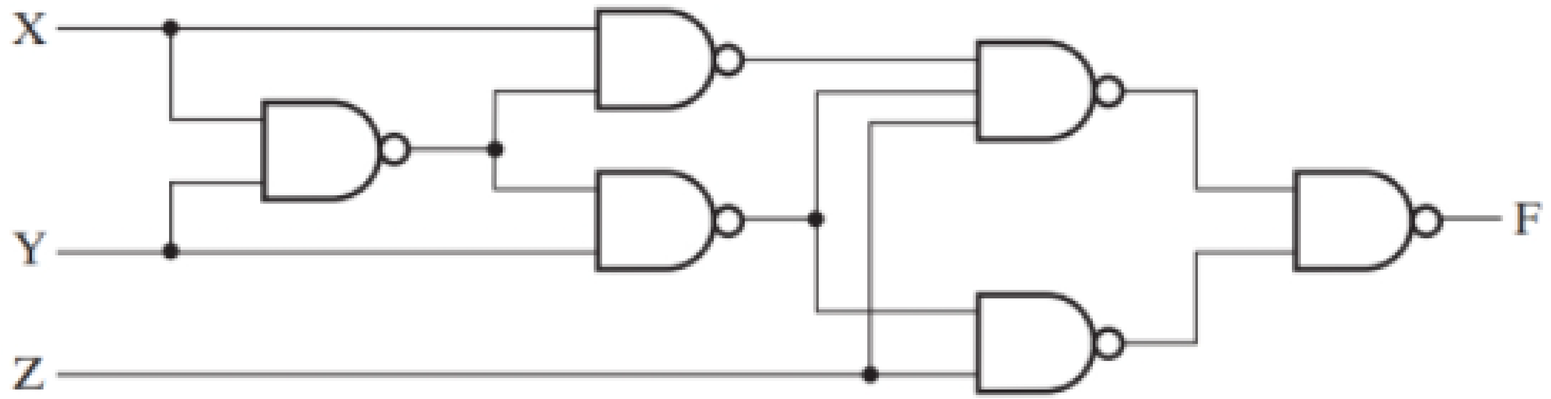
Exercise 2.34 (Textbook Mano-Kime): Logic Diagram

```
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
   entity comb_ckt_1 is
   port(x1, x2, x3, x4 : in std_logic;
        f : out std_logic);
end comb_ckt_1;

architecture structural_1 of comb_ckt_1 is
  component NOT1
    port(in1: in std_logic;
         out1: out std_logic);
  end component;
  component AND2
    port(in1, in2 : in std_logic;
         out1: out std_logic);
  end component;
  component OR3
    port(in1, in2, in3 : in std_logic;
         out1: out std_logic);
  end component;
  signal n1, n2, n3, n4, n5, n6 : std_logic;
  begin
    g0: NOT1 port map (in1 => x1, out1 => n1);
    g1: NOT1 port map (in1 => n3, out1 => n4);
    g2: AND2 port map (in1 => x2, in2 => n1,
                       out1 => n2);
    g3: AND2 port map (in1 => x2, in2 => x3,
                       out1 => n3);
    g4: AND2 port map (in1 => x3, in2 => x4,
                       out1 => n5);
    g5: AND2 port map (in1 => x1, in2 => n4,
                       out1 => n6);
    g6: OR3 port map (in1 => n2, in2 => n5,
                      in3 => n6, out1 => f);
end structural_1;
```

1) Draw the logic diagram of the circuit described in this VHDL code (the complemented inputs are not available)

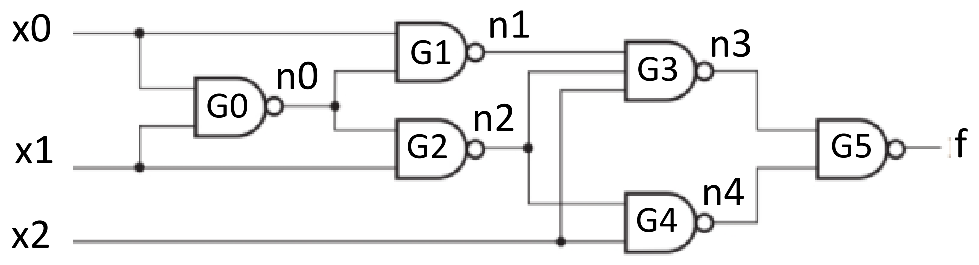


Exercise 2.35 (Textbook Mano-Kime)



1) Write a structural VHDL description of the circuit and replace X, Y, Z with X (2: 0)
2) Simulate and verify the correct operation of the code with EDA Playground, using a testbench and stimulating the circuit with all the possible input combinations

Exercise 2.35: VHDL Design



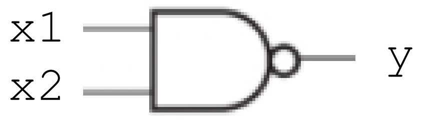
```
library IEEE;
use IEEE.std_logic_1164.all;

entity circuit2_35 is
        port ( x: in std_logic_vector (2 downto 0);
               f: out std_logic);
end circuit2_35;

architecture circuit2_35_struct of circuit2_35 is
signal n: std_logic_vector (4 downto 0);
begin
        G0: entity work.NAND2(NAND2_impl) port map(x(0), x(1), n(0));
        G1: entity work.NAND2(NAND2_impl) port map(x(0), n(0), n(1));
        G2: entity work.NAND2(NAND2_impl) port map(n(0), x(1), n(2));
        G3: entity work.NAND3(NAND3_impl) port map(n(1), n(2), x(2), n(3));
        G4: entity work.NAND2(NAND2_impl) port map(n(2), x(2), n(4));
        G5: entity work.NAND2(NAND2_impl) port map(n(3), n(4), f);

end circuit2_35_struct;
```

Direct instantiation (no declaration is needed)

Exercise 2.35: VHDL Design

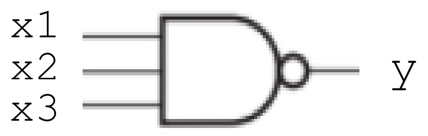
```
library IEEE;
use IEEE.std_logic_1164.all;

entity NAND2 is
        port (x1, x2: in std_logic;
                   y: out std_logic);
end NAND2;

architecture NAND2_Impl of NAND2 is
begin
        y <= x1 nand x2;
end NAND2_impl;
```

Entity and architecture of 2-input NAND



```
library IEEE;
use IEEE.std_logic_1164.all;

entity NAND3 is
        port (x1, x2, x3: in std_logic;
                       y: out std_logic);
end NAND3;

architecture NAND3_Impl of NAND3 is
begin
        y <= not (x1 and x2 and x3);
end NAND3_impl;
```

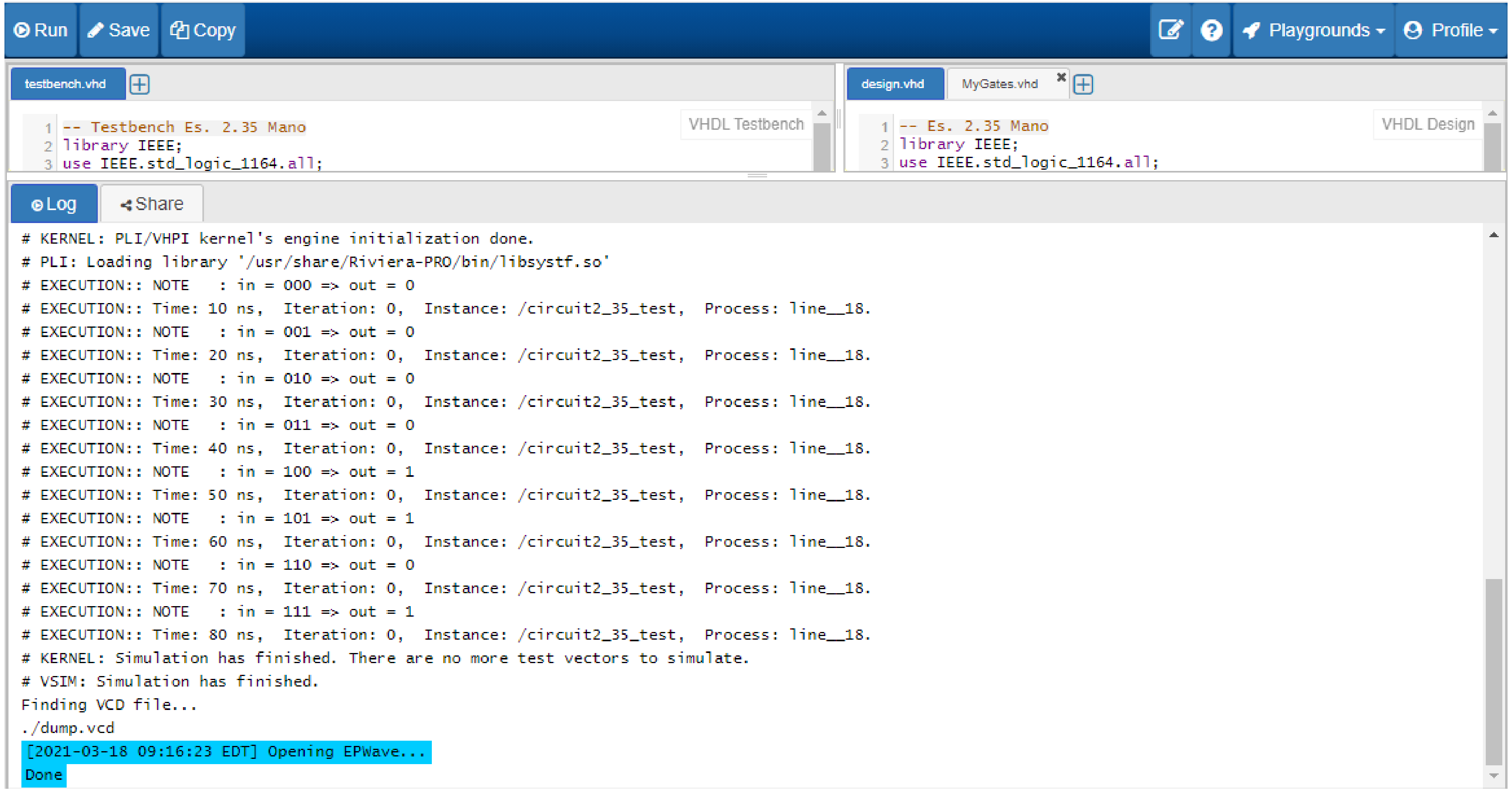
Entity and architecture of 3-input NAND



Exercise 2.35: VHDL Testbench

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.std_logic_unsigned.all; -- unsigned numbers package (sum)

entity circuit2_35_test is
end circuit2_35_test;

architecture test of circuit2_35_test is
signal input: std_logic_vector (2 downto 0); -- to apply stimuli to the DUT
signal output: std_logic;                    -- signal connected at the output of the DUT
component circuit2_35 is
        port ( x: in std_logic_vector (2 downto 0);
               f: out std_logic);
end component;
begin
DUT: circuit2_35 port map(input, output);
process begin                              -apply inputs and display the corresponding output
        input <= "000";
        for i in 0 to 7 loop
            wait for 10 ns;
            report "in =" & to_string(input) & "=> out =" & to_string(output);
            input <= input + 1;
        end loop;
        wait;
end process;
end test;
```

Exercise 2.35: Simulation Log



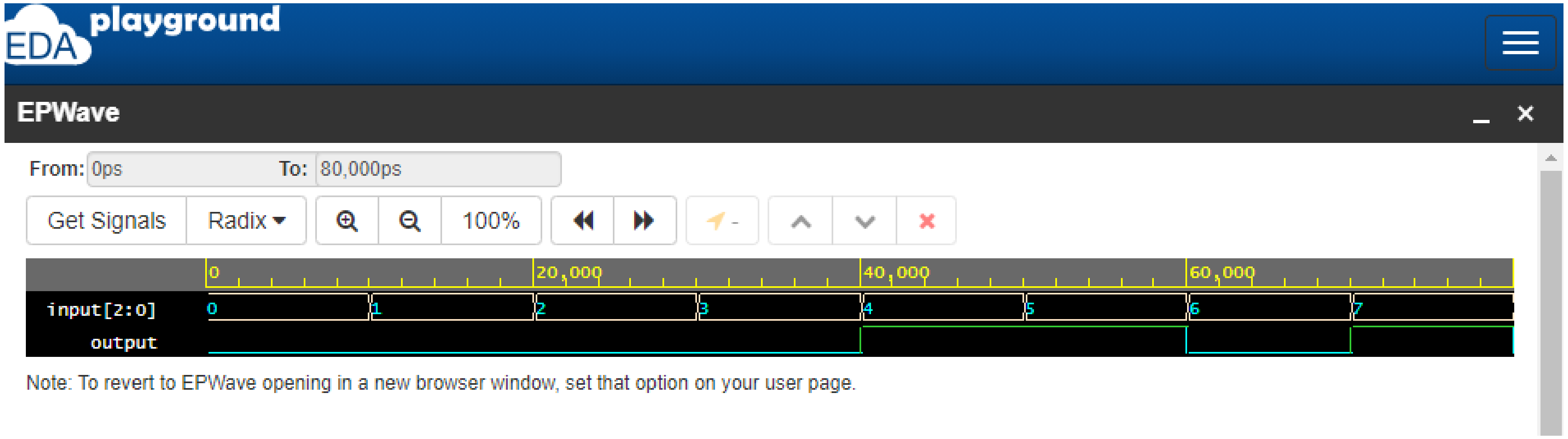
```
-- Testbench Es. 2.35 Mano
library IEEE;
use IEEE.std_logic_1164.all;
```

```
-- Es. 2.35 Mano
library IEEE;
use IEEE.std_logic_1164.all;
```

```
# KERNEL: PLI/VHPI kernel's engine initialization done.
# PLI: Loading library '/usr/share/Riviera-PRO/bin/libsystf.so'
# EXECUTION:: NOTE   : in = 000 => out = 0
# EXECUTION:: Time: 10 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 001 => out = 0
# EXECUTION:: Time: 20 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 010 => out = 0
# EXECUTION:: Time: 30 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 011 => out = 0
# EXECUTION:: Time: 40 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 100 => out = 1
# EXECUTION:: Time: 50 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 101 => out = 1
# EXECUTION:: Time: 60 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 110 => out = 0
# EXECUTION:: Time: 70 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# EXECUTION:: NOTE   : in = 111 => out = 1
# EXECUTION:: Time: 80 ns,  Iteration: 0,  Instance: /circuit2_35_test,  Process: line__18.
# KERNEL: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
Finding VCD file...
./dump.vcd
[2021-03-18 09:16:23 EDT] Opening EPWave...
Done
```

Exercise 2.35: Waveforms

- Selecting "Open **EPWave** after run" the **waveforms of the simulated signals** can be viewed (time diagram with time on x axis)



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Plot of the output corresponding to the 8 possible combinations of the 3 inputs

Exercise 2.37 (Textbook Mano-Kime)

- Find the logic diagram representing the minimized 2-level logic circuit needed to implement the following dataflow VHDL description. The inputs are available in both direct and complemented form

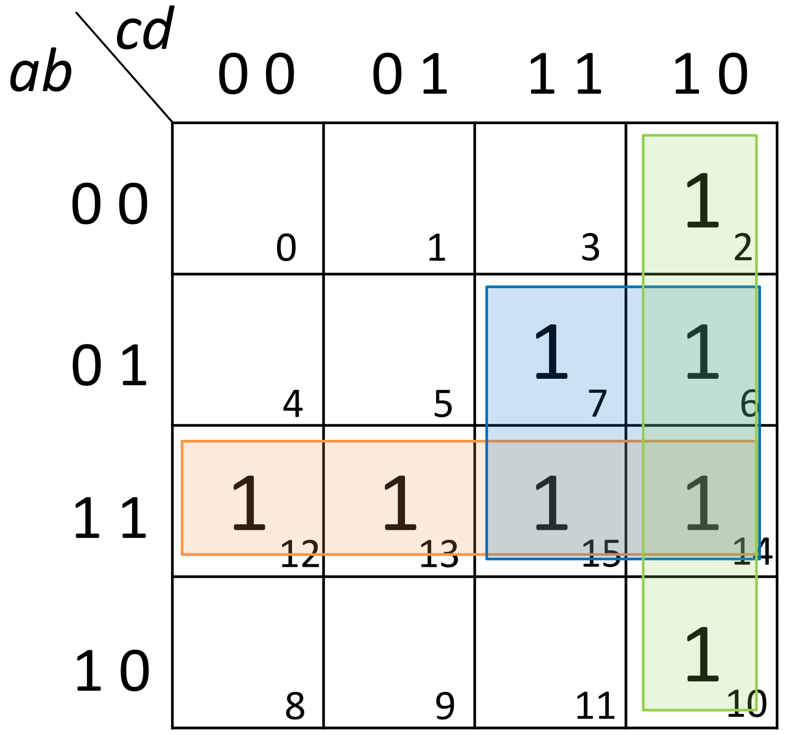
```
-- Combinational Circuit 2: Dataflow VHDL Description
library ieee;
use ieee.std_logic_1164.all;
entity comb_ckt_2 is
   port(a, b, c, d, a_n, b_n, c_n, d_n: in std_logic;
         f, g : out std_logic);
-- a_n, b_n, . . . are complements of a, b, . . . , respectively.

end comb_ckt_2;
architecture dataflow_1 of comb_ckt_2 is
begin
     f <= b and (a or (a_n and c)) or (b_n and c and d_n);
     g <= b and (c or (a_n and c_n) or (c_n and d_n));
end dataflow_1;
```

Exercise 2.37

$$f = b \cdot [a + (\bar{a} \cdot c)] + \left(\bar{b} \cdot c \cdot \bar{d}\right) = a \cdot b + \bar{a} \cdot b \cdot c + \bar{b} \cdot c \cdot \bar{d}$$

- Fill the K-map
- Find PIs and EPIs
- The optimized 2-level expression is

$$f = a \cdot b + b \cdot c + c \cdot \bar{d}$$

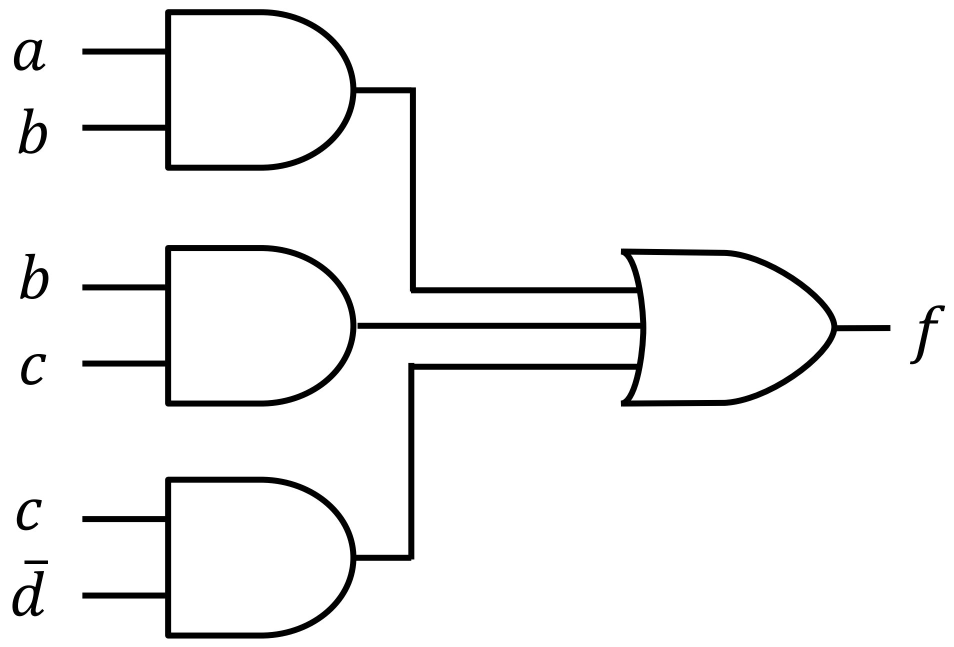


$ab \backslash cd$	0 0	0 1	1 1	1 0
0 0	0	1	3	1 2
0 1	4	5	1 7	1 6
1 1	1 12	1 13	1 15	1 14
1 0	8	9	11	1 10

Exercise 2.37

$$f = a \cdot b + b \cdot c + c \cdot \bar{d}$$

The logic diagram representing the function f is:



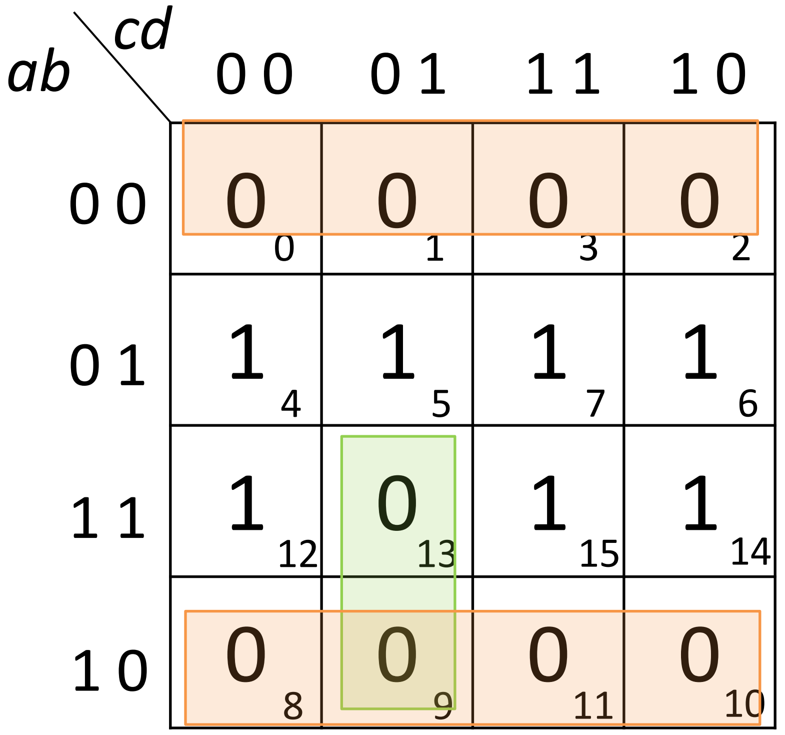
Exercise 2.37

$$g = b \cdot [\mathrm{c} + (\bar{a} \cdot \bar{c}) + \bar{c} \cdot \bar{d}] = b \cdot c + \bar{a} \cdot b \cdot \bar{c} + b \cdot \bar{c} \cdot \bar{d}$$

- Fill the K-map
- Find PIs and EPIs: working on $\bar{g}$ is more convenient
- The optimized 2-level logic expression is

$$\bar{g} = \bar{b} + a \cdot \bar{c} \cdot d$$

$$\Rightarrow g = b \cdot (\bar{a} + c + \bar{d})$$

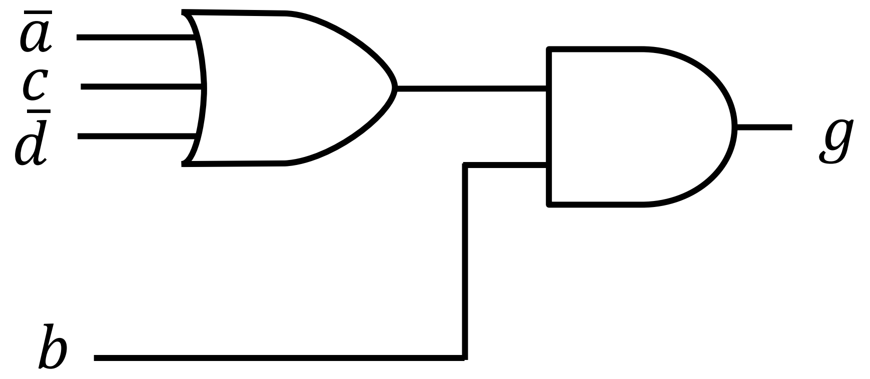


ab \ cd	0 0	0 1	1 1	1 0
0 0	0_{0}	0_{1}	0_{3}	0_{2}
0 1	1_{4}	1_{5}	1_{7}	1_{6}
1 1	1_{12}	0_{13}	1_{15}	1_{14}
1 0	0_{8}	0_{9}	0_{11}	0_{10}

Exercise 2.37

$$g = b \cdot (\bar{a} + c + \bar{d})$$

The logic diagram representing the function g is:



Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime| Martin