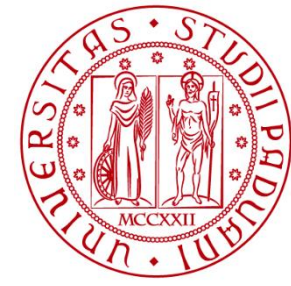




OF THE
DEPARTMENT OF
INFORMATION ENGINEERING



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

The screenshot shows the EDA Playground web interface. The browser address bar displays `edaplayground.com/home`. The top navigation bar includes a 'Run' button, a 'Save' button, and a notification about 'Aldec Riviera Pro 2020.04'. On the left sidebar, the 'Languages & Libraries' section is expanded, showing 'Testbench + Design' set to 'SystemVerilog/Verilog', 'UVM / OVM' set to 'None', and 'Other Libraries' including 'None', 'OVL 2.8.1', and 'SVUnit 2.11'. Below this, the 'Tools & Simulators' section is expanded, showing a 'Select...' dropdown and checkboxes for 'Open EPWave after run' and 'Download files after run'. The 'Examples' section lists 'VHDL', 'Verilog/SystemVerilog', 'UVM', 'EasierUVM', 'SVAUnit', and 'SVUnit'. The main workspace is divided into two panels: 'testbench.sv' and 'design.sv'. The 'testbench.sv' panel contains the text:

```
1 // Code your testbench here
2 // or browse Examples
3
```

 The 'design.sv' panel contains the text:

```
1 // Code your design here
2
```

 Two red arrows point to the 'Testbench + Design' dropdown and the 'Tools & Simulators' dropdown. A text box next to the first arrow says 'Testbench + Design: select "VHDL"'. A text box next to the second arrow says 'Tools & Simulators: select "Aldec Riviera Pro 2022.04"'. At the bottom of the workspace, there is a text area with the placeholder text: '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

The screenshot shows the EDA Playground web interface. The browser address bar displays `edaplayground.com/home`. The interface includes a top navigation bar with 'Run', 'Save*', and a notification about Aldec Riviera Pro 2020.04. The left sidebar contains sections for 'Languages & Libraries', 'Testbench + Design' (with a VHDL dropdown), 'Libraries' (showing None, OVL 2.8.1, OSVVM), 'Top entity' (with a text input 'for simulation ONLY' and an 'Enable VUnit' checkbox), 'Tools & Simulators' (with a dropdown for 'Aldec Riviera Pro 2020.04'), 'Compile Options' (with a text input '-2008 -o'), 'Run Options' (with a dropdown for 'Run Options'), and 'Run Time' (with checkboxes for 'Open EPW' and 'Download f'). The main area is split into two code editors: 'testbench.vhd' (labeled 'VHDL Testbench') and 'design.vhd' (labeled 'VHDL Design'). Both editors contain the same VHDL code:

```
1 -- Code your testbench here
2 library IEEE;
3 use IEEE.std_logic_1164.all;
4
```

 and

```
1 -- Code your design here
2 library IEEE;
3 use IEEE.std_logic_1164.all;
4
```

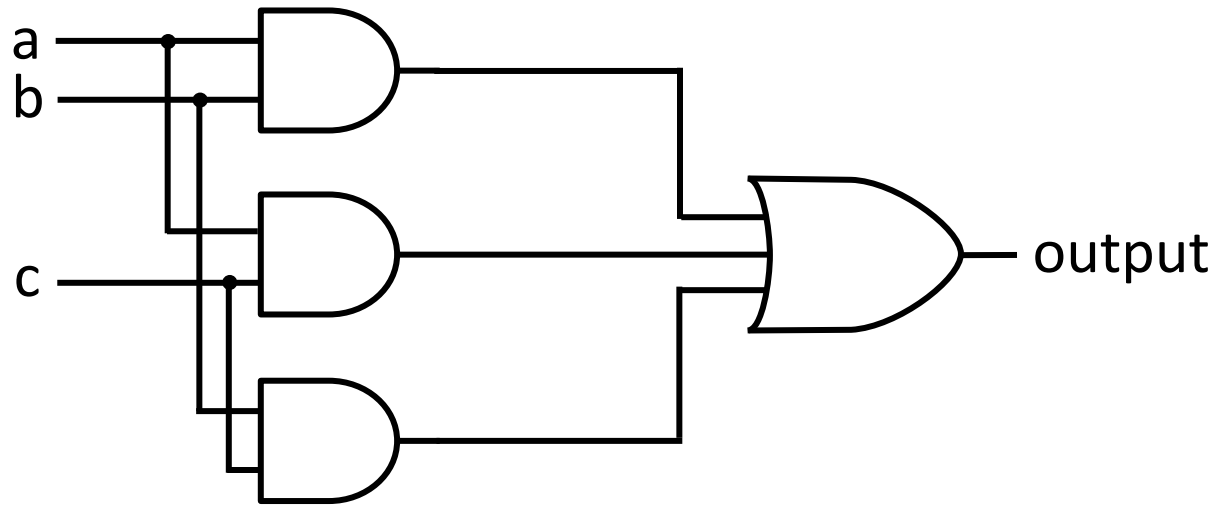
 respectively. Below the editors are 'Log' and 'Share' buttons. The 'Log' button is annotated with 'Log: simulation results'. The 'Share' button is annotated with 'Possibility to make the link available to others'. Below these buttons is a text input field 'Add a title to help you find your playground' (annotated with 'Name of the design (save in your online directory)') and a dropdown menu 'Public (anyone with the link can view)'. A 'Save*' button is also present. The bottom of the interface shows a toolbar with icons for text formatting and a preview window.

Annotations:

- Testbench**: Points to the `testbench.vhd` editor.
- Design**: Points to the `design.vhd` editor.
- Name of the top entity to simulate**: Points to the 'Top entity' input field.
- Possibility to make the link available to others**: Points to the 'Share' button.
- Log: simulation results**: Points to the 'Log' button.
- Name of the design (save in your online directory)**: Points to the title input field.

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;
```

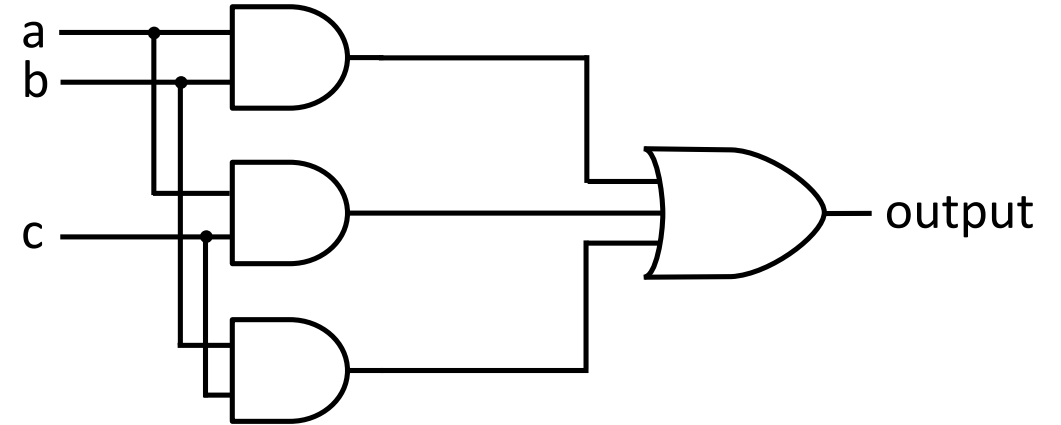
```
entity majority is  
    port ( a, b, c : in std_logic;  
           output : out std_logic );
```

```
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;

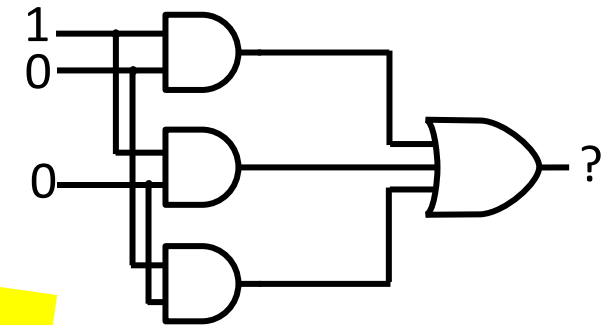
entity maj_test is
end maj_test;

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;
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

The image shows a screenshot of the EDA Playground interface. It features two side-by-side code editors and a bottom log window.

testbench.vhd (VHDL Testbench):

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity maj_test is
5 end maj_test;
6
7 architecture test of maj_test is
8 signal x, y, z, f: std_logic;
9 component majority is
10     port ( a, b, c : in std_logic;
11           output : out std_logic );
12 end component;
13 begin
14 DUT: majority port map (x, y ,z , f );
15 process begin
16     x <= '1';
17     y <= '0';
18     z <= '0';
19     wait for 10 ns;
20     report "output = " & to_string(f);
21     wait;
22 end process;
23 end test;
```

design.vhd (VHDL Design):

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity majority is
5     port ( a, b, c : in std_logic;
6           output : out std_logic );
7 end majority;
8
9 architecture maj_dataflow of majority is
10 begin
11     output <= (a and b) or (a and c) or (b and c);
12 end maj_dataflow;
```

Log Window:

Log Share

```
# 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;
```

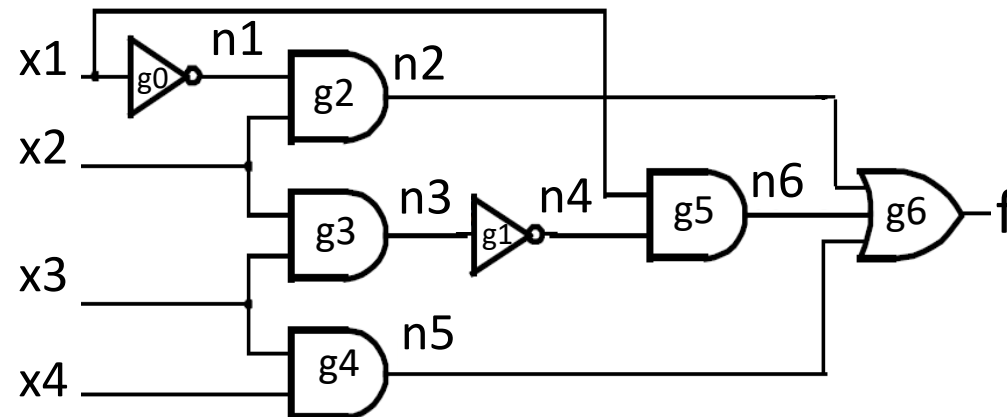
- 1) Draw the logic diagram of the circuit described in this VHDL code (the complemented inputs are not available)

Exercise 2.34 (Textbook Mano-Kime): Logic Diagram

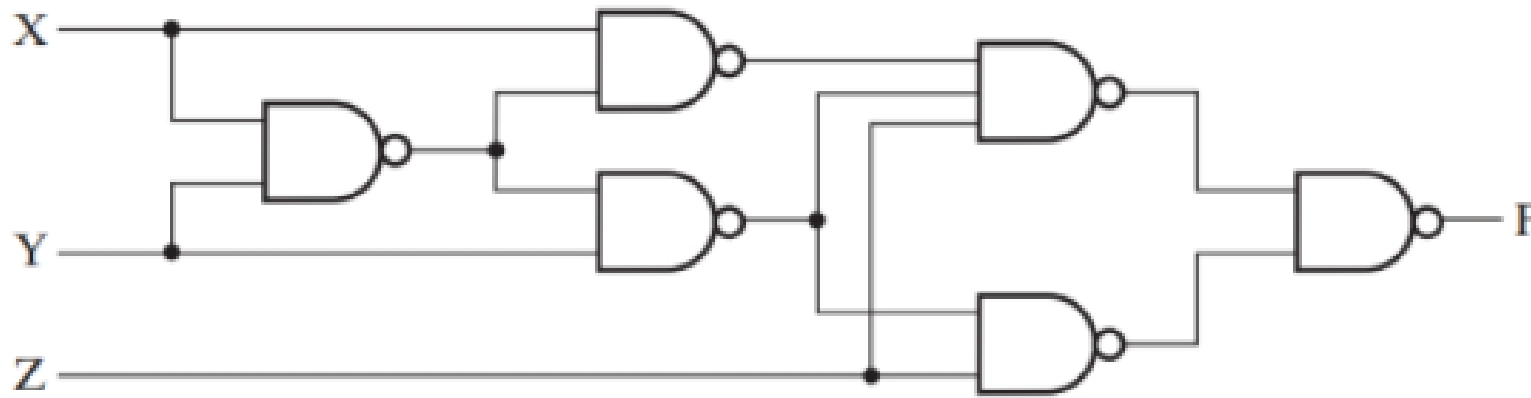
```
library ieee, lcdf_vhdl;
use ieee.std_logic_1164.all, lcdf_vhdl.func_prims.all;
entity comb_ckt_1 is
  port(x1, x2, x3, x4 : in std_logic;
        f : out std_logic);
end comb_ckt_1;

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

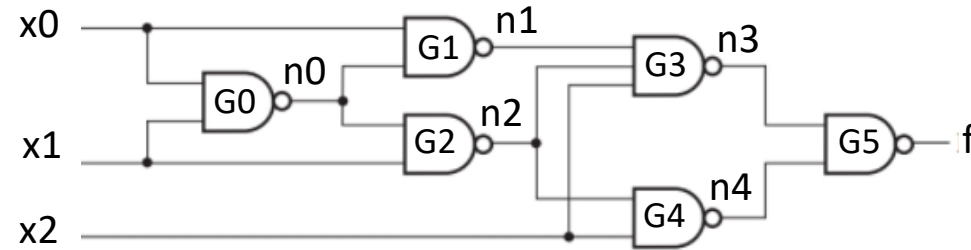
```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
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;
```

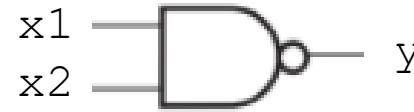
```
architecture NAND2_Impl of NAND2 is  
begin  
    y <= x1 nand x2;  
end NAND2_impl;
```

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

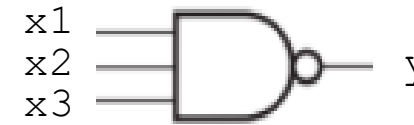
```
entity NAND3 is  
    port (x1, x2, x3: in std_logic;  
          y: out std_logic);  
end NAND3;
```

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

Entity and
architecture of
2-input NAND



Entity and
architecture of
3-input NAND



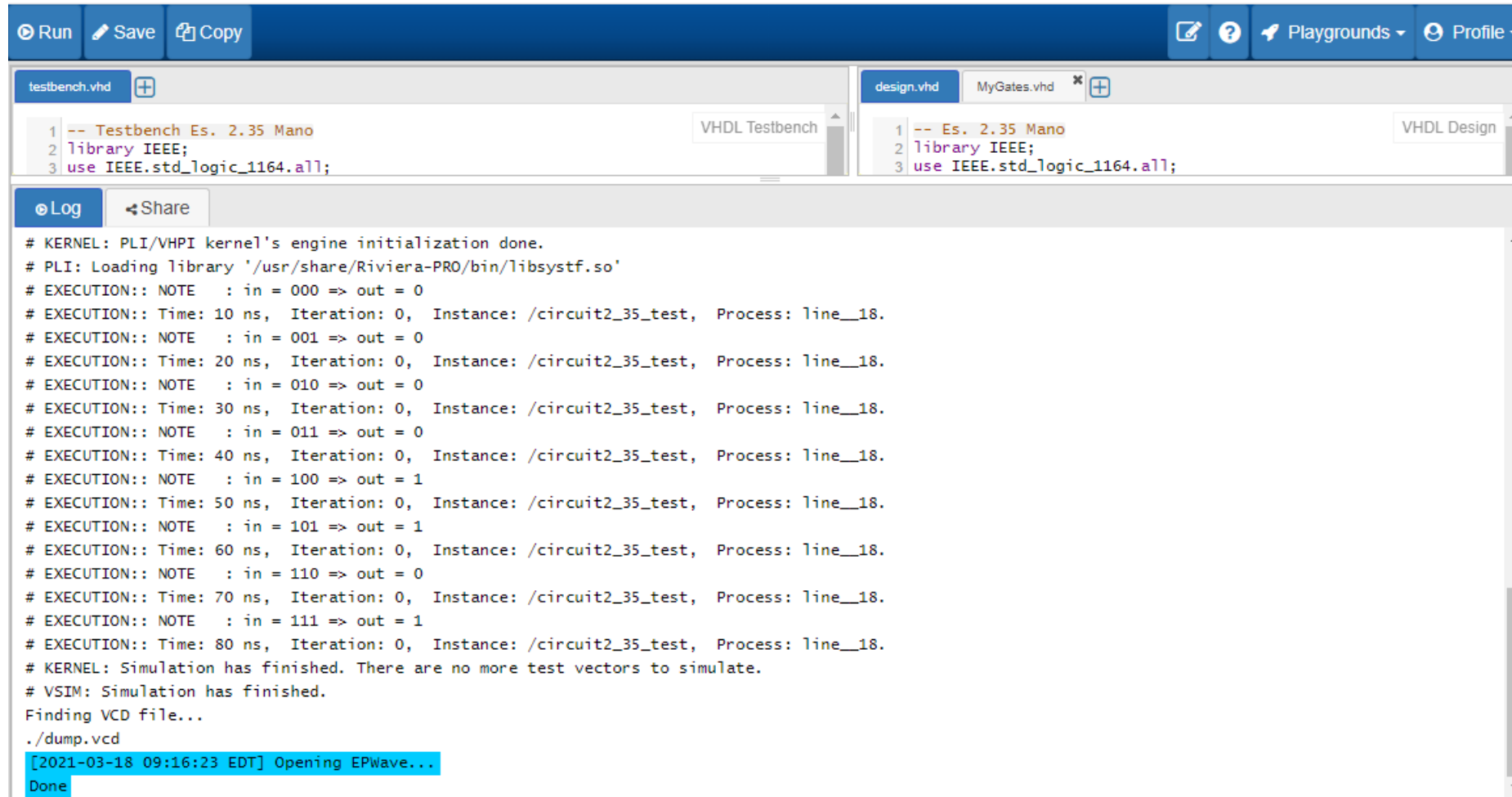
Exercise 2.35: VHDL Testbench

```
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



The screenshot shows a web-based VHDL simulation interface. At the top, there is a toolbar with buttons for 'Run', 'Save', and 'Copy'. Below the toolbar, there are two tabs: 'testbench.vhd' and 'design.vhd'. The 'testbench.vhd' tab is active, showing the following code:

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

The 'design.vhd' tab is also visible, showing the following code:

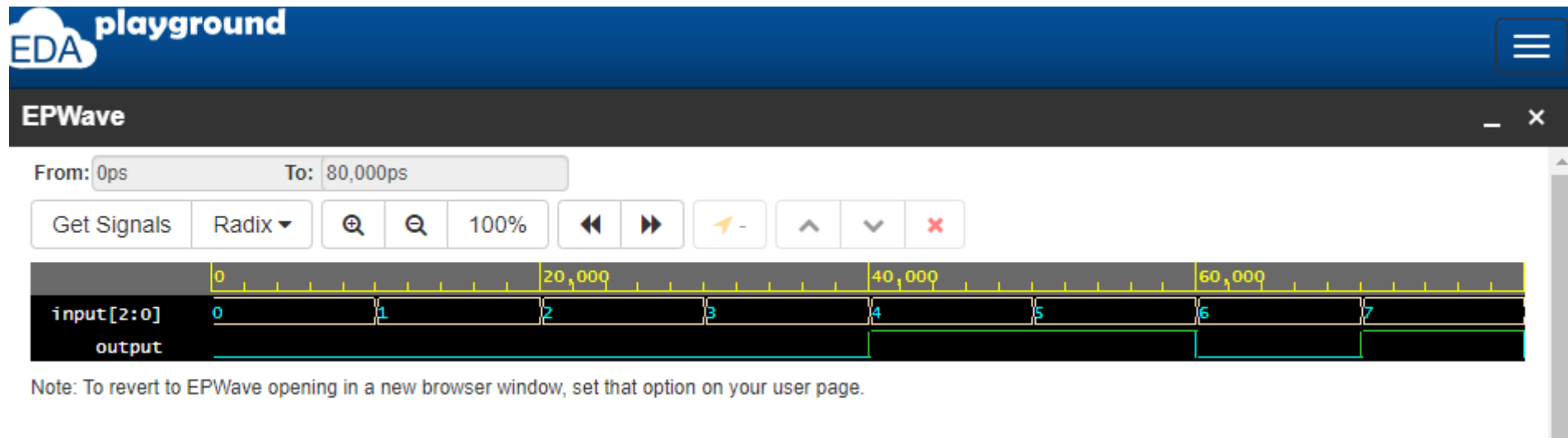
```
1 -- Es. 2.35 Mano
2 library IEEE;
3 use IEEE.std_logic_1164.all;
```

Below the code editors, there is a 'Log' button and a 'Share' button. The 'Log' button is active, and the simulation log is displayed below it. The log contains the following text:

```
# KERNEL: PLI/VHPI kernel's engine initialization done.
# PLI: Loading library '/usr/share/Riviera-PRO/bin/libsysstf.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)



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

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

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

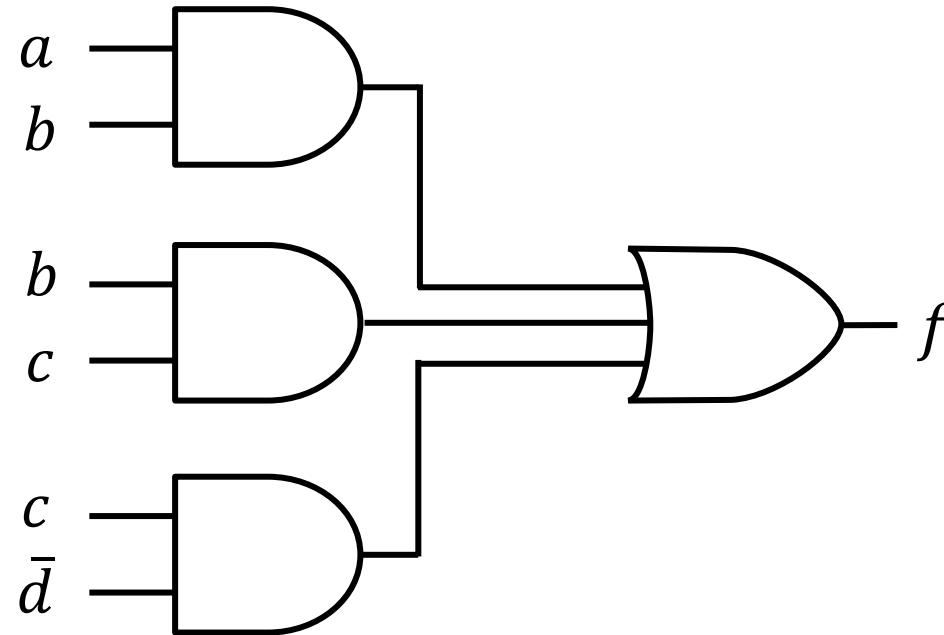
$$f = a \cdot b + b \cdot c + c \cdot \bar{d}$$

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

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

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

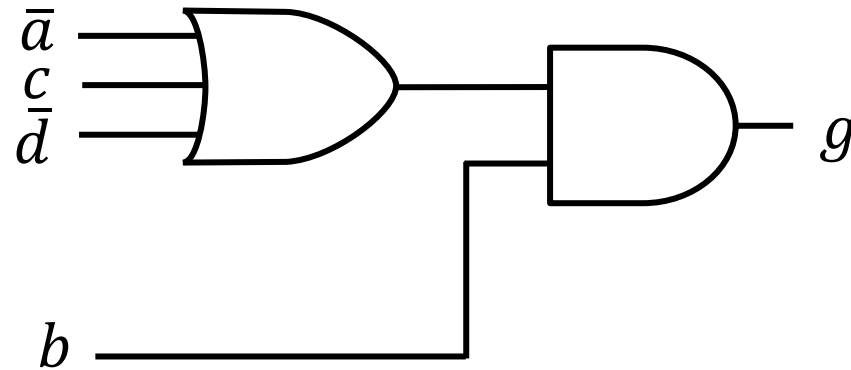
$$\begin{aligned}\bar{g} &= \bar{b} + a \cdot \bar{c} \cdot d \\ \Rightarrow g &= b \cdot (\bar{a} + c + \bar{d})\end{aligned}$$

$ab \backslash cd$					
		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

© 2016 Pearson Education, Ltd