CSc 21100 (Spring 2021)
Project 02 (20 points)

Note: Both project report and the source codes need to be submitted to Blackboard by the end of the due day.

IMPORTANT!

Please follow the **submission guidelines** below or your submission will be rejected.

1. You are expected to submit two files (a PDF lab report and a ZIP file containing the source files) to Blackboard. Note that both files must be upload in the same submission attempt.
2. Each task must have its own source file(s). For VHDL assignments, all source files of this assignment must be in a single Xilinx VHDL project.
3. For VHDL assignments, you must use the export function of the Xilinx ISE Design Suite to create the ZIP file properly. Please refer to Blackboard -> Content -> Lab Lecture Slides -> Exporting_VHDL_project_files
4. Naming convention:
   Report: "FirstName_LastName_Project_**XX**_CC**Y**.pdf"*
   Project: "FirstName_LastName_Project_**XX**_CC**Y**.zip"*
   *Replace "XX" and "Y" with the actual project number (two digits) and section number, respectively.
5. After the due day, all submissions are final. You cannot change it for any reasons. Double check before you make the submission.

In this project, students are expected to use the Xilinx ISE Design Suite (Webpack edition) to create a project and complete the following tasks.

Please read the instructions carefully. Failing to follow the instructions would lead to significant point deductions.

Task 1: Inhibit gate (5 points)

Implement an inhibit gate using the source code in Table 5-13.

Note: Please do not just copy & paste the codes into your VHDL source file. The purpose is to make sure you pay attention to all the details in the code. So please type in the codes and add your own comments.

```
entity Inhibit is      -- also known as 'BUT-NOT'
  port (X,Y: in BIT;    --  as in 'X but not Y'
        Z:   out BIT);  --  (see [Klir, 1972])
end Inhibit;

architecture Inhibit_arch of Inhibit is
begin
  Z <= '1' when X='1' and Y='0' else '0';
end Inhibit_arch;
```

Table 5-13
VHDL program for an "inhibit" gate.

Use the test-bench program below and run simulations to validate your design.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY inhibit_test IS
END inhibit_test;

ARCHITECTURE structural OF inhibit_test IS
    COMPONENT inhibit
    PORT(X : IN  BIT;
         Y : IN  BIT;
         Z : OUT BIT);
    END COMPONENT;
    signal X, Y, Z: BIT;
BEGIN
   uut: inhibit PORT MAP (X, Y, Z);
   stim_proc: process
   begin
      -- insert stimulus here
      X <= '0'; Y <= '0';
      wait for 20 ns;
      X <= '0'; Y <= '1';
      wait for 20 ns;
      X <= '1'; Y <= '0';
      wait for 20 ns;
      X <= '1'; Y <= '1';
      wait;
   end process;
END;
```

Deliverables

Requirement(s):

(1) You must follow the structural design method.
(2) You must follow the submission guidelines.
Note: no points will be given if any of the requirements are not satisfied.

Rubric (Report)

1.1 Use your own language to describe the function of the component to be implemented in VHDL. (1 points)
1.2 Include your VHDL entity declaration(s) and architecture definition(s). (1 points)
1.3 Show simulation results (e.g. the waveforms). Explain the outcome of each test case. (1 points)

Rubric (Source Code)

1.4 Can compile without any errors. (1 points)
1.5 Can run simulations without any errors. (1 points)

Task 2: Prime-number detector (7 points)

Implement a prime number detector according to the source codes in Table 5-30.

Note: Please do not just copy & paste the codes into your VHDL source file. The purpose is to make sure you pay attention to all the details in the code. So please type in the codes and add your own comments.

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

entity prime is
    port ( N: in STD_LOGIC_VECTOR (3 downto 0);
           F: out STD_LOGIC );
end prime;

architecture prime1_arch of prime is
signal N3_L, N2_L, N1_L: STD_LOGIC;
signal N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0: STD_LOGIC;
component INV port (I: in STD_LOGIC; O: out STD_LOGIC); end component;
component AND2 port (I0,I1: in STD_LOGIC; O: out STD_LOGIC); end component;
component AND3 port (I0,I1,I2: in STD_LOGIC; O: out STD_LOGIC); end component;
component OR4 port (I0,I1,I2,I3: in STD_LOGIC; O: out STD_LOGIC); end component;
begin
  U1: INV port map (N(3), N3_L);
  U2: INV port map (N(2), N2_L);
  U3: INV port map (N(1), N1_L);
  U4: AND2 port map (N3_L, N(0), N3L_N0);
  U5: AND3 port map (N3_L, N2_L, N(1), N3L_N2L_N1);
  U6: AND3 port map (N2_L, N(1), N(0), N2L_N1_N0);
  U7: AND3 port map (N(2), N1_L, N(0), N2_N1L_N0);
  U8: OR4 port map (N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0, F);
end prime1_arch;
```

Table 5-30

Structural VHDL program for a prime-number detector.

Use the test-bench program below and run simulations to validate your design.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY primel_tb IS
END primel_tb;

ARCHITECTURE behavior OF primel_tb IS
    COMPONENT primel
    PORT(
         N : IN  std_logic_vector(3 downto 0);
         F : OUT  std_logic
        );
    END COMPONENT;
   signal N : std_logic_vector(3 downto 0) := (others => '0');
   signal F : std_logic;
BEGIN
   uut: primel PORT MAP (
          N => N,
          F => F
        );
   stim_proc: process
   begin
      N <= "0000"; wait for 10 ns;
      N <= "0001"; wait for 10 ns;
      N <= "0010"; wait for 10 ns;
      N <= "0011"; wait for 10 ns;
      N <= "0100"; wait for 10 ns;
      N <= "0101"; wait for 10 ns;
      N <= "0110"; wait for 10 ns;
      N <= "0111"; wait for 10 ns;
      N <= "1000"; wait for 10 ns;
      N <= "1001"; wait for 10 ns;
      N <= "1010"; wait for 10 ns;
      N <= "1011"; wait for 10 ns;
      N <= "1100"; wait for 10 ns;
      N <= "1101"; wait for 10 ns;
      N <= "1110"; wait for 10 ns;
      N <= "1111"; wait for 10 ns;
      wait;
   end process;
END;
```

Deliverable(s):

Requirement(s)
(1) You must follow the structural design method.
(2) You must follow the submission guidelines.
Note: no points will be given if any of the requirements are not satisfied.

Rubric (Report)

2.1 Include your VHDL entity declaration(s) and architecture definition(s). (1 points)

2.2 Show simulation results (e.g. the waveforms). Explain the outcome of <u>each</u> test case. (4 points)

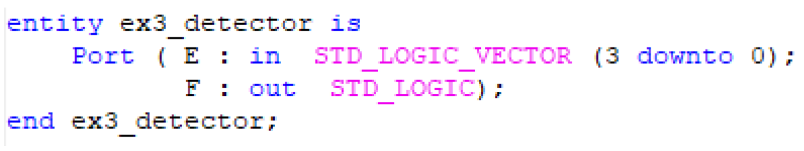
Rubric (Source Code)

2.3 Can compile without any errors. (1 points)

2.4 Can run simulations without any errors. (1 points)

Task 3: Excess-3 code detector (8 points)

In the homework assignment, you are asked to design an excess-3 code detector using a two-level NAND-NAND circuit (i.e. only NAND and NOT gates). Please write a VHDL program to implement the circuit using <u>*structural design*</u>. Make sure you use the entity declaration provided below. No points would be given if failed to follow it.



```
entity ex3_detector is
    Port ( E : in  STD_LOGIC_VECTOR (3 downto 0);
           F : out  STD_LOGIC);
end ex3_detector;
```

Use the test-bench program below and run simulations to validate your design.

```
-- Stimulus process
stim_proc: process
begin
   -- insert stimulus here
   E <= "0000"; wait for 10 ns;
   E <= "0001"; wait for 10 ns;
   E <= "0010"; wait for 10 ns;
   E <= "0011"; wait for 10 ns;
   E <= "0100"; wait for 10 ns;
   E <= "0101"; wait for 10 ns;
   E <= "0110"; wait for 10 ns;
   E <= "0111"; wait for 10 ns;
   E <= "1000"; wait for 10 ns;
   E <= "1001"; wait for 10 ns;
   E <= "1010"; wait for 10 ns;
   E <= "1011"; wait for 10 ns;
   E <= "1100"; wait for 10 ns;
   E <= "1101"; wait for 10 ns;
   E <= "1110"; wait for 10 ns;
   E <= "1111"; wait for 10 ns;
   wait;
end process;
```

Deliverables:

Requirement(s):
(1) You must follow the structural design method.
(2) You must follow the submission guidelines.
Note: no points will be given if any of the requirements are not satisfied.

Rubric (Report)

3.1 Draw a circuit diagram of the module to show the design. (1 point)
3.2 Include your VHDL entity declaration(s), architecture definition(s) and the testbench program. (1 point)
3.3 Show simulation results (e.g. the waveforms). Explain the outcome of each test case with screenshots. Show why the simulation result is correct. (4 points)

Rubric (Source Code)

3.4 Can compile without any errors. (1 point)
3.5 Can run simulations without any errors. (1 point)