CSc 21100 (Spring 2021)
Project 03 (20 points)

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 -> Assignments -> 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) 14.7 to complete the following tasks.

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

Task 1: 2-to-4 Decoder (5 points)

A 2-to-4 decoder operates according to the following function table.

Inputs			Outputs			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Table 6-4

Truth table for a 2-to-4 binary decoder.

Implement a 2-to-4 decoder in VHDL using *structural design*: Please adopt the following as the entity declaration.

```
entity v2to4dec is
   port ( EN : in    std_logic;
          I0 : in    std_logic;
          I1 : in    std_logic;
          Y0 : out   std_logic;
          Y1 : out   std_logic;
          Y2 : out   std_logic;
          Y3 : out   std_logic);
end v2to4dec;
```

Note that, many logic gates are defined in the *UNISIM* library. In order to use these gates, you need to add the following lines in your VHDL program to include the *UNISIM* library.

```
library UNISIM;
use UNISIM.VComponents.all;
```

Write a test-bench program and run simulations to validate your design: Use the given test cases below in your test-bench program. Pay attention to the signal names, signal values, and the time.

```
-- insert stimulus here
EN <= '0';
I1 <= '0'; I0 <= '0'; wait for 10 ns;
I1 <= '0'; I0 <= '1'; wait for 10 ns;
I1 <= '1'; I0 <= '0'; wait for 10 ns;
I1 <= '1'; I0 <= '1'; wait for 10 ns;
EN <= '1';
I1 <= '0'; I0 <= '0'; wait for 10 ns;
I1 <= '0'; I0 <= '1'; wait for 10 ns;
I1 <= '1'; I0 <= '0'; wait for 10 ns;
I1 <= '1'; I0 <= '1'; wait for 10 ns;
```

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)

1.1. Use your own language to describe the function of the module to be implemented in VHDL. (1 point)
1.2. Include your VHDL entity declaration(s), architecture definition(s) and the testbench program. (1 point)
1.3. Show simulation results (e.g. the waveforms). Explain the outcome of each test case with screenshots. Show why the simulation result is correct. (1 point)

Rubric (Source Code)

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

Task 2: 3-to-8 Decoder (7 points)

Implement a 3-to-8 decoder using the 2-to-4 decoder you have implemented in Task 1. The input to the 3-to-8 decoder should be labeled as ***A***. It needs to be a 3-bit bus. The output should be labeled as ***O***, which is an 8-bit bus. Please use the following entity declaration for the 3-to-8 decoder.

```
entity v3to8dec is
   port ( A  : in    std_logic_vector (2 downto 0);
          EN : in    std_logic;
          O  : out   std_logic_vector (7 downto 0));
end v3to8dec;
```

Note that: (1) please adopt the *structural design approach*; (2) in addition to the 2-to-4 decoders, you also need to include some logic gates to make it work.

In order to use the 2-to-4 decoder from Task 1, you will need to declare the 2-to-4 decoder as a component in the VDHL architecture definition of the 3-to-8 decoder:

```
component v2to4dec
   port ( I0 : in    std_logic;
          I1 : in    std_logic;
          EN : in    std_logic;
          Y3 : out   std_logic;
          Y2 : out   std_logic;
          Y1 : out   std_logic;
          Y0 : out   std_logic);
end component;
```

Then you can use the decoder like this:

```
DEC1 : v2to4dec
   port map (EN=>YOUR_SIGNAL_1,
             I0=>YOUR_SIGNAL_2,
             I1=>YOUR_SIGNAL_3,
             Y0=>YOUR_SIGNAL_4,
             Y1=>YOUR_SIGNAL_5,
             Y2=>YOUR_SIGNAL_6,
             Y3=>YOUR_SIGNAL_7);
```

Once you successfully completed the implementation of the VHDL module, write a test-bench program and run simulations to validate your design. Use the given test cases below in your test-bench program. Pay attention to the signal names, signal values, and the time.

```
EN<='0'; A<="000"; wait for 10 ns;
EN<='0'; A<="001"; wait for 10 ns;
EN<='0'; A<="010"; wait for 10 ns;
EN<='0'; A<="011"; wait for 10 ns;
EN<='0'; A<="100"; wait for 10 ns;
EN<='0'; A<="101"; wait for 10 ns;
EN<='0'; A<="110"; wait for 10 ns;
EN<='0'; A<="111"; wait for 10 ns;

EN<='1'; A<="000"; wait for 10 ns;
EN<='1'; A<="001"; wait for 10 ns;
EN<='1'; A<="010"; wait for 10 ns;
EN<='1'; A<="011"; wait for 10 ns;
EN<='1'; A<="100"; wait for 10 ns;
EN<='1'; A<="101"; wait for 10 ns;
EN<='1'; A<="110"; wait for 10 ns;
EN<='1'; A<="111"; wait for 10 ns;
```

Deliverable(s):

Requirement(s):

(1) You must follow the structural design method.
(2) You must use the module(s) implemented before.
(3) You must follow the submission guidelines.

Note: no points will be given if any of the requirements are not satisfied.

Rubric (Report)

2.1 Use your own language to describe the function of the module to be implemented in VHDL. (1 point)
2.2 Draw a circuit diagram of the module to show the design. (1 point)
2.3 Include your VHDL entity declaration(s), architecture definition(s) and the testbench program. (1 point)
2.4 Show simulation results (e.g. the waveforms). Use the simulation results to explain why the implemented circuit is working correctly. (2 points)

Rubric (Source Code)

2.5 Can compile without any errors. (1 point)
2.6 Can run simulations without any errors. (1 point)

Task 3: A 4-bit Ripple Adder (8 points)

Implement a 4-bit ripple adder, similar to the one we discussed in class. You need to first implement a 1-bit full adder then cascade them to build the 4-bit ripple adder. Please use the following entity declarations for the 1-bit full adder and the 4-bit ripple adder.

```
entity adder1b is
    Port ( X : in  STD_LOGIC;
           Y : in  STD_LOGIC;
           CIN : in  STD_LOGIC;
           S : out  STD_LOGIC;
           COUT : out  STD_LOGIC);
end adder1b;
```

```
entity adder4b is
    Port ( X : in  STD_LOGIC_VECTOR (3 downto 0);
           Y : in  STD_LOGIC_VECTOR (3 downto 0);
           S : out  STD_LOGIC_VECTOR (3 downto 0);
           CIN : in  STD_LOGIC;
           COUT : out  STD_LOGIC);
end adder4b;
```

Note that: (1) please adopt the *structural design approach*; (2) please make sure you use the following libraries and packages:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;
```

In order to use the 1-bit full adder, you will need to declare it as a component in 4-bit ripple adder implementation:

```
component adder1b port (X, Y, CIN: in STD_LOGIC; COUT, S: out STD_LOGIC); end component;
```

Then you can use the 1-bit full adder like this:

```
LABEL1: adder1b
   port map (  X=>YOUR_SIGNAL_1,
               Y=>YOUR_SIGNAL_2,
               CIN=>YOUR_SIGNAL_3,
               COUT=>YOUR_SIGNAL_4,
               S=>YOUR_SIGNAL_5);
```

Once you successfully completed the implementation of the VHDL module, use the test-bench program below and run simulations to validate your design.

```
1
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4  USE ieee.numeric_std.ALL;
5  USE ieee.std_logic_unsigned.ALL;
6  USE ieee.std_logic_signed.ALL;
7  ENTITY adder4b_tb IS
8  END adder4b_tb;
9  ARCHITECTURE behavior OF adder4b_tb IS
10     COMPONENT adder4b PORT(
11          X : IN  std_logic_vector(3 downto 0);
12          Y : IN  std_logic_vector(3 downto 0);
13          S : OUT  std_logic_vector(3 downto 0);
14          CIN : IN  std_logic;
15          COUT : OUT  std_logic
16         );
17     END COMPONENT;
18    --Inputs
19    signal X : std_logic_vector(3 downto 0) := (others => '0');
20    signal Y : std_logic_vector(3 downto 0) := (others => '0');
21    signal CIN : std_logic := '0';
22    --Outputs
23    signal S : std_logic_vector(3 downto 0);
24    signal COUT : std_logic;
25 BEGIN
26    -- Instantiate the Unit Under Test (UUT)
27    uut: adder4b PORT MAP (X => X, Y => Y, S => S, CIN => CIN, COUT => COUT);
28    -- Stimulus process
29    stim_proc: process
30    begin
31       -- hold reset state for 10 ns.
32       wait for 10 ns;
33       -- insert stimulus here
34       CIN <= '0';
35       for i in 0 to 15 loop
36          X <= std_logic_vector(to_unsigned(i,4));
37          for j in 0 to 15 loop
38             Y <= std_logic_vector(to_unsigned(j,4));
39             wait for 2 ns;
40          end loop;
41       end loop;
42       CIN <= '1';
43       for i in 0 to 15 loop
44          X <= std_logic_vector(to_unsigned(i,4));
45          for j in 0 to 15 loop
46             Y <= std_logic_vector(to_unsigned(j,4));
47             wait for 2 ns;
48          end loop;
49       end loop;
50       wait;
51    end process;
52 END;
53
54
```

Deliverable(s):

Requirement(s):

(1) You must follow the structural design method.
(2) You must use the module(s) implemented before.
(3) You must follow the submission guidelines.

Note: no points will be given if any of the requirements are not satisfied.

Rubric (Report)

3.1 Use your own language to describe the function of the module to be implemented in VHDL. (1 point)
3.2 Draw a circuit diagram of the module to show the design. (1 point)
3.3 Include your VHDL entity declaration(s), architecture definition(s) and the testbench program. (1 point)
3.4 Show simulation results (e.g. the waveforms). Use the simulation results to explain why the implemented circuit is working correctly. (3 points)

Rubric (Source Code)

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