CET4805 **Component and Subsystem Design II**

EXPERIMENT # 3: Implementing Logic Designs with VHDL

Name: _____Date: _____

Equipment/Parts Needed:

Quartus II^{R} Web Edition V9.1 SP2 software by Altera Corporation
USB drive to save your files

Objective:

- Use simple VHDL assignment statements to represent a function table
- Introduce the selected signal assignment WHEN-ELSE clause
- Compare bit, vector, and integer data types

Discussion:

Boolean equations are extracted from function tables using the Sum of Products (SOP) method. Typically, complex SOP expressions would be simplified either by using Boolean simplification techniques based on the laws, principles, and identities or using the Karnaugh mapping techniques.
Once the SOP equation to represent an output function has been extracted and simplified, the basic VHDL assignment statement can be written. Instead of creating the circuit using basic logic gates, one can write the VHDL code.

Statement: The output (X) of a circuit is a logic-HIGH only when input A is a logic-LOW and input B is a logic-HIGH or input A is a logic-HIGH and input B is a logic-LOW.

Input		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Table 3-1

The function table for the logic statement is shown in Table 1. The SOP output (Eq. 1) and assignment statement (Eq.2) are shown below.

$X = \overline{A}B + A\overline{B}$ Eq. 1)
$X <= A\ XOR\ B$ Eq. 2)

The assignment statement is then placed in the architecture body of the VHDL code as shown in Text Box 1.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY vhdl3_1a IS
      PORT(
                  A, B        :IN STD_LOGIC;
                  X           :OUT STD_LOGIC);
END vhdl3_1b;
ARCHITECTURE behavior OF vhdl3_1c IS
      BEGIN
                  X <= A XOR B;
END behavior;
```

Text Box 1

An alternative to using the assignment statement of Eq. 2) is to use the conditional signal assignment WHEN-ELSE clause. Examine the VHDL code shown in Text Box 2.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY vhdl3_1b IS
      PORT(
               A, B        :IN STD_LOGIC;
               X           :OUT STD_LOGIC)
END vhdl3_1b;
ARCHITECTURE behavior OF vhdl3_1b IS
      BEGIN
               X <= '1' WHEN A='1' OR B='1' ELSE '0';
END behavior;
```

Text Box 2

Notice that multiple WHEN-ELSE clauses appear in the assignment statement shown in Text Box 3.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY vhdl3_1c IS
      PORT(
                  A, B        :IN STD_LOGIC;
                  X           :OUT STD_LOGIC;
END vhdl3_1c;
ARCHITECTURE behavior OF vhdl3_1c IS
      BEGIN
                  X <=  '1' WHEN A='0' AND B='1' ELSE
                        '1' WHEN A='1' AND B='0' ELSE '0';
END behavior;
```

Text Box 3

Replacing input **A** and **B** bit declarations with a 2-bit bus, C[1..0], shown in Text Box 4, simplifies the VHDL code. Inputs and outputs that are defined as STD_LOGIC in the Entity declaration will be given a declared numeric value within single quotes. Inputs and outputs defined as STD_LOGIC_VECTOR(*high* DOWNTO *low*) are represented by the correct number of bits within double quotes.

As the size of the vectors increases to 4-, 8-, 16-, or other bit lengths, keeping track of the correct number of binary bits is tedious. Inputs and outputs may be defined as integers and, therefore, decimal numbers can be used without needing the single or double quotes. Compare Text Box 4 to Text Box 5 when defining **C** as an integer data type.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY vhdl3_1d IS
      PORT(
                  C           :IN STD_LOGIC_VECTOR(1 DOWNTO 0);
                  X           :OUT STD_LOGIC);
END vhdl3_1d;
ARCHITECTURE behavior OF vhdl3_1d IS
      BEGIN
                  X <=  '1' WHEN C="001" ELSE
                        '1' WHEN C="10" ELSE '0';
END behavior;
```

Text Box 4

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY vhdl3_1e IS
      PORT(
                  C           :IN INTEGER RANGE 0 to 3;
                  X           :OUT STD_LOGIC);
END vhdl3_1e;
ARCHITECTURE behavior OF vhdl3_1e IS
      BEGIN
                  X <=  '1' WHEN C=1 ELSE
                        '1' WHEN C=5 ELSE '0';
END behavior;
```

Text Box 5

Part 1 Procedure

Creating a New Project

1. Open the Quartus II software. Select **File – New Project Wizard**. Enter the appropriate drive letter for the designated storage area on the computer you are using followed by the working directory C**:\altera\91sp2\quartus\kwon\Lab3**. You need to go through the step from 1 through 8 in the Part 1 of **Lab1** manual. Don't forget to create the folder **Lab3** under the subfolder of your last name. Assign the project name **Lab3_1**, assign ***Cyclone II*** for the device family, and select the ***EP2C35F672C6*** chip in the Family & device settings [page of 3 of 5].

Creating a VHDL File (bdf)

2. Open a new VHDL Device Design file (**File > New**) by highlighting VHDL File. Type the VHDL codes shown in Text Box 1.

3. Save the VHDL file as **vhdl3_1a.vhd** as part of our project under your subfolder. Place a check mark in the space labeled Add file to current project and press Save.

4. Select **File > Create/Update > Create Symbol Files for Current File** to create a symbol file for the VHDL code entered. A display window should soon appear stating that the **Create Symbol File was (or not) successful**. Click **OK** and close the Compilation Report window.

5. Open the **vhdl3_1a.vhd** text file and modify the VHDL file to match that of Text Box 2. And save the text file as **vhdl3_1b.vhd.** Create a symbol for the **vhdl3_1b.vhd** (**File > Create/Update > Create Symbol Files for Current File**).

6. Repeat this step 5 to create symbols for Text Box 3, 4, and 5. Correct all errors if the **Create Symbol File was not successful**.

Question: You should get some errors when creating each symbol per Table Box. You need to correct the VHDL code to create symbols successfully. Explain why you had errors each Table Box.

7. Open a new Schematic file (**File > New**) by highlighting **Block Diagram/Schematic File**. And click **OK**. And construct the circuit shown in Figure 3-1 using the symbols you just created. Each symbol should be available in the Project Library in the Symbol diagonal box.

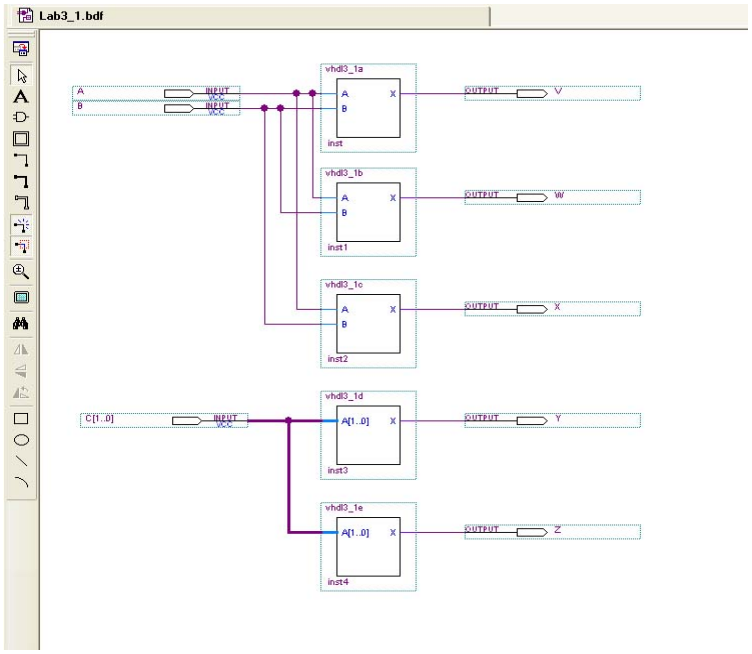


Figure 3-1

8. Before compiling this *bdf* file, we need to name this *bdf* file and save it as part of our project under your subfolder. Choose **File > Save As** and enter **File name** as *lab3_1*. Place a *check mark* in the space labeled **Add file to current project** and press **Save**.

9. Compile the project by selecting **Processing** > **Start Compilation**, or press Ctrl-L, or use the Compilation button in the toolbar. The compilation takes several seconds. When it is complete it should give a message that indicates, “Full compilation was sucessful”. Press **OK**. If unsuccessful, correct all errors and try to re-compile.

Simulating a Vector Waveform File (vwf)

10. As you have done step 23 through 28 in the **Part 1** of Lab1, you need to create a **Vector Waveform File (*vwf*)** to simulate a design(*bdf*) file. Add all inputs and output, specify an end time of 1 µs and a grid size of 100ns for our waveform display, and then save it as lab3_1.vwf.

11. When creating the C[1..0] bus, enter C for the bus name, select Hexadecimal for the Radix, and enter 2 for the Bus Width in the Node Properties window. When created, the C waveform will appear with a plus sign implying that it can be ungrouped to show the individual bits, C[1] and C[0].

12. Select **Processing – Start Simulation**, or press Ctrl-I, or use the Simulation button in the toolbar. After a few moments a message stating "Simulation was successful" should appear. Click **OK**.

13. The Simulation Waveforms appear in the Simulation Report shown in the Figure 3-2. You may have to expand the size of the Simulation Waveforms to suit your need and choose **View > Fit in Window** to see the entire 1µs waveform.

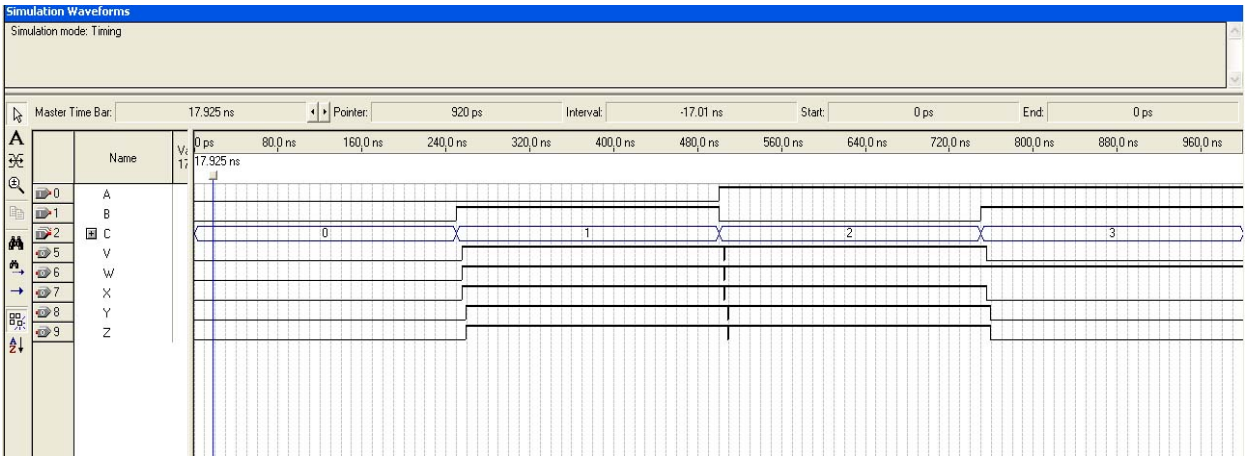


Figure 3-2

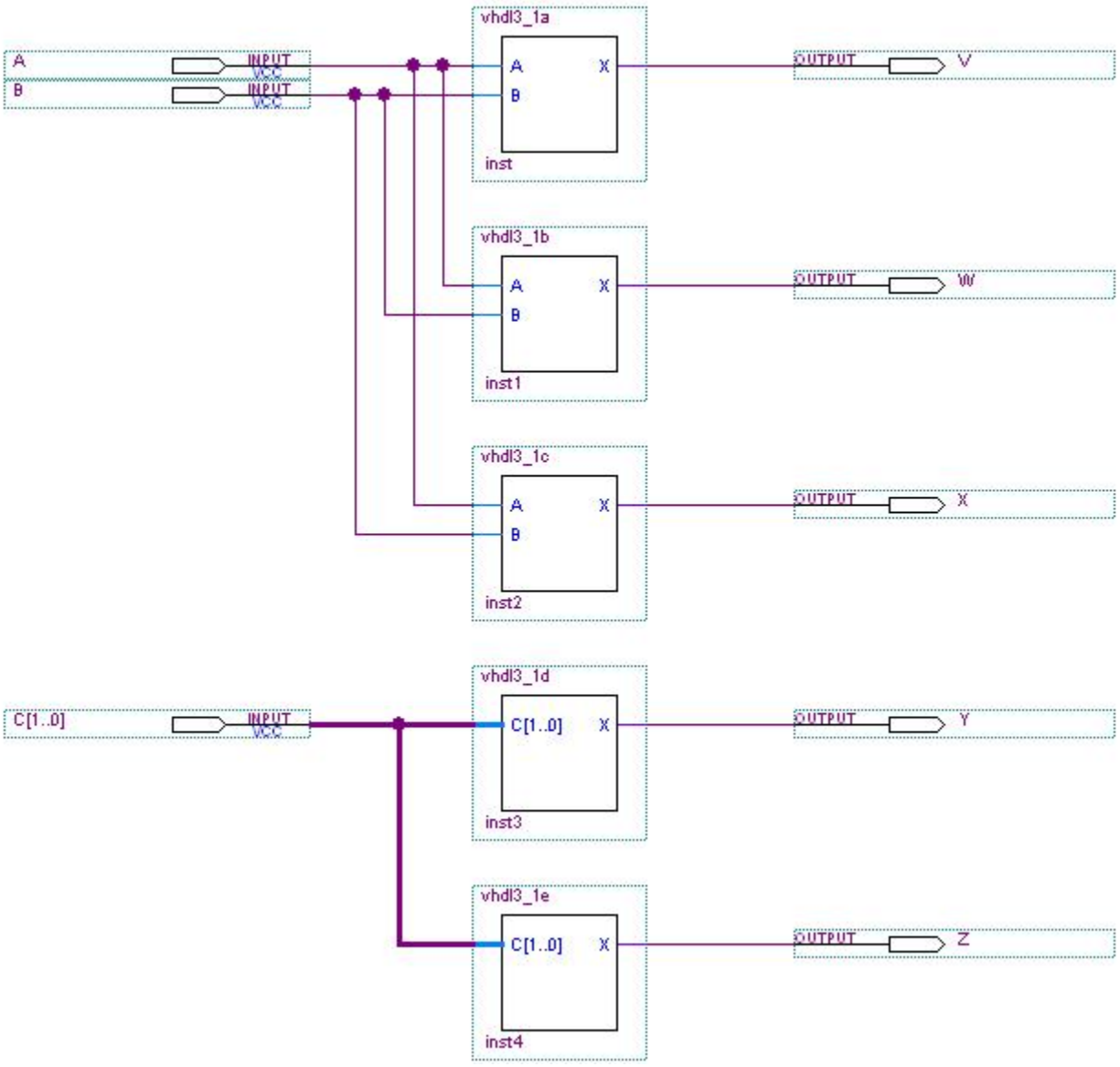
Question: You should get wrong waveform result for the output W in the simulation. You need to modify the VHDL code for Table box 2. Explain why you had wrong output result and how to modify to get the correct waveform result.

Part 2 Practice

1. Do it again to implement logic design for the Table 3-2.
   1) Create the SOP equation for X.
   2) Create the VHDL codes using the same format of Table Boxes in the Part 1. (5 Table Boxes)
   3) Create a Block Design File (*bdf* file) for X using the symbol created from the *vhdl* file.
   4) Create a Vector Waveform File (*vwf*) for X. The simulation should show all possible combination of inputs.
   5) Include the copies of *vhdl* codes and *bdf* file and *vwf* file in the lab report.

A	B	C	X
0	**0**	**0**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**
1	**1**	**1**	**0**

Table 3-2


A
B
INPUT
VCC
vhdl3_1a
A
B
X
inst
OUTPUT
V
vhdl3_1b
inst1
W
vhdl3_1c
inst2
X
C[1..0]
vhdl3_1d
inst3
Y
vhdl3_1e
inst4
Z