VHDL Assignment #2: Combinational Circuit Using VHDL Concurrent Statements

1 Instructions

- Due date: Friday, October 9, 2020 by 5pm.
- Submission is in teams using myCourses (only one team member submits). In the report, provide the names and McGill IDs of the team members.
- Late submissions will incur penalties as described in the course syllabus.

2 Introduction

In this assignment you will be required to write simple logic functions in VHDL employing the techniques you learned in class. This document contains material obtained directly from "all about circuits".

3 Learning Outcomes

After completing this lab you should know how to:

- Use VHDL concurrent statements to implement combinational logic functions
- Write efficient VHDL codes

You will also have a better understanding on the concept of synthesis of logic circuits.

4 Concurrent Statements in VHDL

Combinational circuits can be described in VHDL using "Concurrent statements". There are different types of concurrent statements in VHDL.

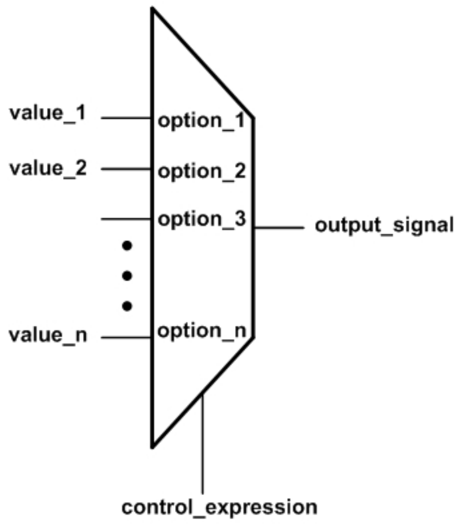
4.1 Direct Assignment

The simplest form of concurrent statements is the direct assignment. In VHDL assignment #1, you learned how to describe a 2-to-1 multiplexer using the direct assignment. Below is an example of direct assignment which performs XOR operation between two variables:

```vhdl
c <= a XOR b;
```

4.2 Selected Signal Assignment or the "With/Select" Statement

Consider an n-to-1 multiplexer shown below. This block selects one of its n inputs and transfers the value of this input to the output terminal, *i.e.*, *output_signal*.



The selected signal assignment allows to implement the functionality of a multiplexer:

```
with control_expression select
    output_signal <= value_1 when  option_1,
                     value_2 when  option_2,
                     ...
                     value_n when  option_n;
```

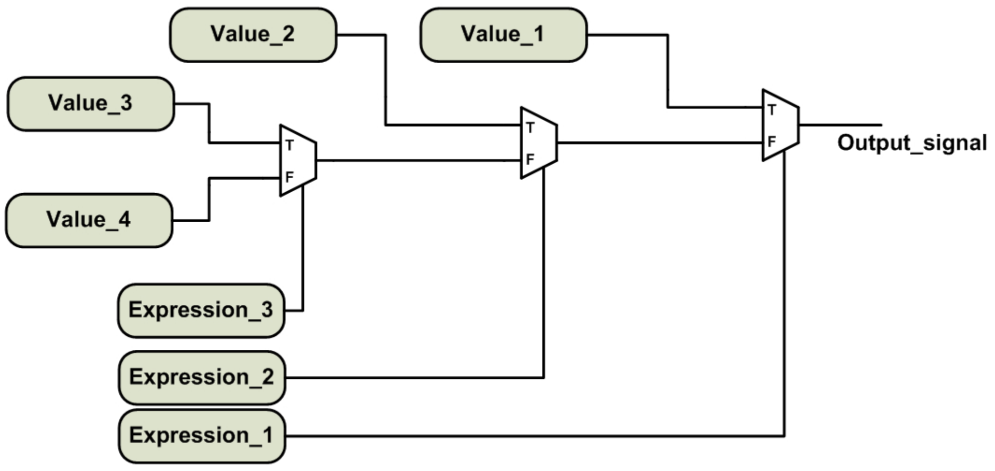
Here, the value of the *control_expression* will be compared with the *n* possible options, *i.e.*, *option_1*, *option_2*, ..., *option_n*. When a match is found, the value corresponding to that particular option will be assigned to the output signal, *i.e.*, *output_signal*. For example, if *control_expression* is the same as *option_2*, then *value_2* will be assigned to the *output_signal*. Note that the options of a "with/select" assignment must be mutually exclusive, *i.e.*, one option cannot be used more than once. Moreover, all possible values of the *control_expression* must be included in the set of options.

4.3 Conditional Signal Assignment or the "When/Else" Statement

The "when/else" statement is another way to describe concurrent signal assignments. In general, the syntax of the "when/else" statement is:

```
output_signal <=  value_1 when  expression_1 else
                  value_2 when  expression_2 else
                  ...
                  value_n;
```

In this case, the expressions after "when" are evaluated successively until a true expression is found. The assignment corresponding to this true expression will be performed. If none of these expressions are true, the last assignment will be executed. We should emphasize that the expressions after the "when" clauses are evaluated successively. As a result, expressions evaluated earlier have higher priority as compared to later expressions. Considering this, we can obtain the conceptual diagram of this assignment as shown below. This figure illustrates a conditional signal assignment with three "when" clauses.



4.4 Concurrent Statements at a Glance

Concurrent statements are executed at the same time and there is no significance to the order of these statements. This type of code is quite different from what we have learned in basic computer programming where the lines of code are executed one after the other.

The selected signal assignment or the "with/select" assignment allows us to implement the functionality of a multiplexer.

The options of a "with/select" assignment must be mutually exclusive, *i.e.*, one option cannot be used more than once. Moreover, all possible values of control_expression must be included in the set of options.

For the "when/else" statement, the expressions following the "when" clauses are evaluated successively. As a result, the expressions evaluated earlier have a higher priority as compared to the following expressions.

One important difference between the "with/select" and "when/else" assignment can be seen by comparing the conceptual implementation of these two statements. The "when/else" statement has a priority network; however, the "with/select" assignment avoids this chain structure and has a balanced structure.

5 Objective

Consider the truth table given below for functions f_1 and f_2.

x_1	x_2	x_3	x_4	f_1	f_2
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

You are required to implement the functions f_1 and f_2 using different VHDL concurrent statements. For all of your VHDL descriptions, use the following entity declaration:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity entity_name is
    Port (x1          : in   std_logic;
          x2          : in   std_logic;
          x3          : in   std_logic;
          x4          : in   std_logic;
          f1          : out   std_logic;
          f2          : out  std_logic);
end entity_name;
```

Perform the following tasks:

1. Derive the canonical SOP expressions for the functions f_1 and f_2 and calculate the cost of the canonical SOP expressions. Implement the given functions using **direct assignment** and the canonical SOP expressions. Use **can_SOP** as the *entity_name*. Perform an exhaustive test on your implementation.
2. Derive the canonical POS expressions for the functions f_1 and f_2 and calculate the cost of the canonical POS expressions. Implement the given functions using **direct assignment** and the canonical POS expressions. Use **can_POS** as the *entity_name*. Perform an exhaustive test on your implementation.
3. Derive the reduced (simplified/optimized) SOP expressions for the functions f_1 and f_2 and calculate the cost of the reduced SOP expressions. You can use the techniques you have learned so far in this course to derive the optimized expressions. Implement the obtained functions using **direct assignment** and the reduced SOP expressions. Use **sim_SOP** as the *entity_name*. Perform an exhaustive test on your implementation.
4. Derive the reduced (simplified/optimized) POS expressions for the functions f_1 and f_2 and calculate the cost of the reduced POS expressions. You can use the techniques you have learned so far in this course to derive the optimized expressions. Implement the obtained functions using **direct assignment** and the reduced POS expressions. Use **sim_POS** as the *entity_name*. Perform an exhaustive test on your implementation.
5. Implement the functions f_1 and f_2 using the **select signal assignment**. Perform an exhaustive test on your implementation. Use **select_assignment** as the *entity_name*. Perform an exhaustive test on your implementation.
6. Implement the functions f_1 and f_2 using the **conditional signal assignment**. Perform an exhaustive test on your implementation. Use **when_assignment** as the *entity_name*. Perform an exhaustive test on your implementation.
7. Find the most joint-optimized SOP expressions of the functions f_1 and f_2 and calculate the cost of your implementation. Implement the joint-optimized SOP expressions using **Direct** assignment. Use **joint_SOP** as the *entity_name*. Perform an exhaustive test on your implementation.

6 Deliverables

Once completed, you are required to submit a report explaining your work in detail (the techniques you used to optimize the functions), including answers to the questions related to this lab. You will find the questions in the next section. You must also submit all the design files (*i.e.*, your VHDL code) along with Log content of the synthesized circuits. You must also submit one of your testbench files. If you fail to submit any part of the required deliverables, you will incur grade penalties as described in the syllabus.

7 Questions

1. Explain your VHDL code.
2. Report the costs and the number of logic modules used to fit your designs on the FPGA board.

Task	1	2	3	4	5	6	7
Cost					N/A	N/A	
Logic Utilization (in LUTs)							

3. Show representative simulation plots of all tasks for all the possible input values (exhaustive test results). Note that you can simply include snapshots from the waveform that you obtained from EPWave. In order to fully capture all the signals from the waveform, you can adjust the display range using the magnifier icons. Make sure that all signal names and axes are properly visible.