VHDL Assignment #3: Design and Simulation of Digital Circuits

1 Instructions

- TA in charge: Muhammad Bilal Babar (muhammad.babar@mail.mcgill.ca) - please utilize discussion boards on myCourses for questions as much as possible.
- Due date: Sunday, October 9, 2022 by 11:59 pm EDT.
- 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 Learning Outcomes

After completing this assignment you should know how to:

- Use CAD tools and VHDL to implement logic functions, specifically, barrel shifters, and ripple-carry and BCD adders.
- Synthesize logic functions
- Perform functional simulation

3 Introduction

In this assignment, you will build upon previously implemented circuits to design a more complex circuit. You will learn how to design and simulate useful adder circuit blocks.

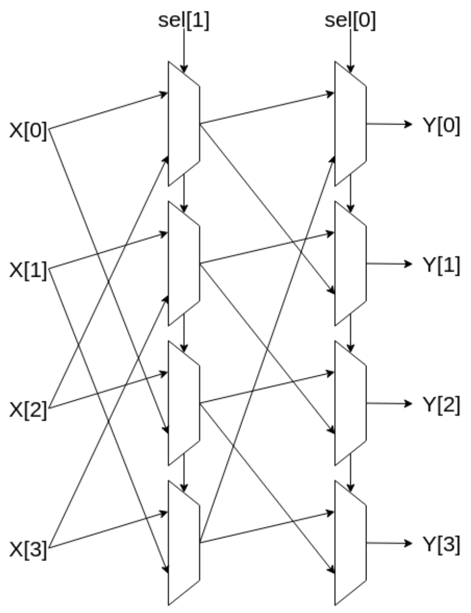
If you need any help regarding the lab materials, you can

- Ask the TA for help during lab sessions and office hours.
- Refer to the text book. In case you are not aware, Appendix A “VHDL Reference" provides detailed information on VHDL.
- You can also refer to the tutorial on Quartus and ModelSim provided by Intel (click here for Quartus and here for ModelSim).

It is highly recommended that you first try to resolve any issue by yourself (refer to the textbook and/or the multitude of VHDL resources on the Internet). Syntax errors, especially, can be quickly resolved by reading the error message to see exactly where the error occurred and checking the VHDL Reference or examples in the textbook for the correct syntax.

4 VHDL Implementation of a 4-bit circular barrel shifter

In this part of the lab, you will design a circuit that implements an 4-bit circular barrel shifter. The circular barrel shifter is a circuit that can shift its input by a given number of bits with combinational logic. The 4-bit circular barrel shifter is usually implemented by stacking two layers of 2×1 multiplexers as shown below. All 4 multiplexer in the leftmost layer use `sel(1)` as the selector signal. Similarly, all 4 multiplexers in the rightmost layer use `sel(0)` as the selector signal. Make sure that you familiarize yourself with how the barrel shifter works by calculating its output for several input combinations by hand.



You are required to implement the 4-bit barrel shifter in VHDL using both structural and behavioral styles. To obtain a structural description of the 4-bit barrel shifter, you are required to use the structural description of the 2-to-1 multiplexer. Write a structural VHDL description for the 4-bit circular barrel shifter by instantiating the structural description of the 2-to-1 multiplexer 8 times. Use the following entity declaration for your structural VHDL description of the 4-bit circular barrel shifter:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity firstname_lastname_barrel_shifter_structural is
    Port (X          : in  std_logic_vector (3 downto 0);
          sel        : in  std_logic_vector (1 downto 0);
          Y          : out std_logic_vector (3 downto 0));
end firstname_lastname_barrel_shifter_structural;
```

Once completed, you are required to implement the 4-bit circular barrel shifter using the behavioral style. One way to obtain a behavioral description of the 4-bit circular barrel shifter is the use of VHDL select statements. Write a behavioral VHDL code for the 4-bit circular barrel shifter using a single VHDL select statement only. Use the following entity declaration for your behavioral VHDL description of the 4-bit circular barrel shifter:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity firstname_lastname_barrel_shifter_behavioral is
    Port (X          : in  std_logic_vector (3 downto 0);
          sel        : in  std_logic_vector (1 downto 0);
          Y          : out std_logic_vector (3 downto 0));
end firstname_lastname_barrel_shifter_behavioral;
```

Hint: You may want to use the VHDL concatenate operator *ampersand* (&). It can be used to combine two or more signals together. Note that all input signals to the concatenation must be of the same type and the result of the concatenation needs to exactly fit the width of the concatenated input signals.

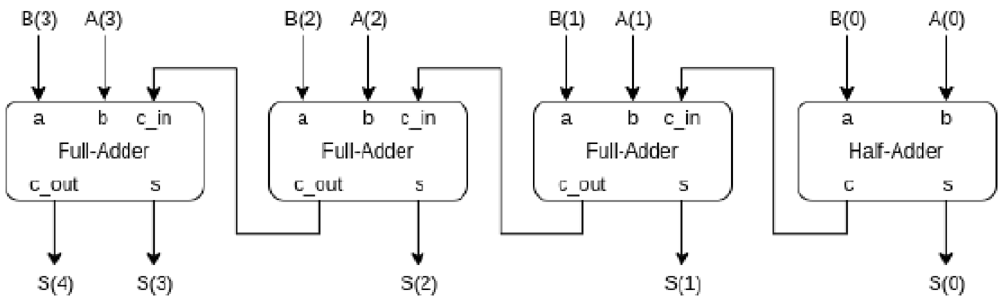
Once you have your circuit described in VHDL using both structural and behavioral styles, write a testbench code and perform an exhaustive test for your VHDL descriptions of the 4-bit circular barrel shifter.

5 VHDL Description of Adder Circuits

In this section, you will be asked to perform the design and simulation of the following two adder circuits: (a) a 4-bit ripple-carry adder; and (b) a one-digit binary coded decimal (BCD) adder. Details of the assignments are described below.

5.1 Ripple-Carry Adder (RCA)

In this section, you will implement a structural description of a 4-bit ripple-carry adder using basic addition components: half-adders and full-adders.



5.1.1 Structural Description of a Half-Adder in VHDL

A half-adder is a circuit that takes two binary digits as inputs, and produces the result of the addition of the two bits in the form of a sum and carry signals. The carry signal represents an overflow into the next digit of a multi-digit addition. Using the following entity definition for your VHDL code, implement a structural description of the half-adder.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity half_adder is
  port (a: in std_logic;
        b: in std_logic;
        s: out std_logic;
        c: out std_logic);
end half_adder;
```

After you have described your structural style of the half-adder in VHDL, you are required to test your circuit. Write a testbench code and perform an exhaustive test of your VHDL description of the half-adder.

5.1.2 Structural Description of a Full-Adder in VHDL

Unlike the half-adder, a full-adder adds binary digits while accounting for values carried in (from a previous stage addition). Write a structural VHDL description for the full-adder circuit using the half-adder circuit that you designed in the previous section. Use the following entity declaration for your structural VHDL description of the full-adder.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity full_adder is
  port (a: in std_logic;
        b: in std_logic;
        c_in: in std_logic;
        s: out std_logic;
        c_out: out std_logic);
end full_adder;
```

After you have described your circuit in VHDL, write a testbench code and perform an exhaustive test of your VHDL description of the full-adder.

5.1.3 Structural Description of a 4-bit Ripple-Carry Adder (RCA) in VHDL

Using the half-adder and full-adder circuits implemented in the two previous sections, implement a 4-bit carry-ripple adder. Write a structural VHDL code for the 4-bit RCA using the following entity declaration.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity rca_structural is
port ( A: in std_logic_vector (3 downto 0);
       B: in std_logic_vector (3 downto 0);
       S: out std_logic_vector (4 downto 0));
end rca_structural;
```

Note that $S(4)$ contains the carry-out of the 4-bit adder. After you have described your circuit in VHDL, write a testbench code and perform an exhaustive test of your VHDL structural description of the 4-bit RCA.

5.1.4 Behavioral Description of a 4-bit RCA in VHDL

In this part, you are required to implement the 4-bit RCA using behavioral description. One way to obtain a behavioral description is to use arithmetic operators in VHDL (*i.e.*, "+"). Write a behavioral VHDL code for the 4-bit RCA using the following entity declaration for your behavioral VHDL description.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity rca_behavioral is
port ( A: in std_logic_vector (3 downto 0);
       B: in std_logic_vector (3 downto 0);
       S: out std_logic_vector (4 downto 0));
end rca_behavioral;
```

After you have described your circuit in VHDL, write a testbench code and perform an exhaustive test of your VHDL behavioral description of the 4-bit RCA.

5.2 VHDL Description of a One-Digit BCD Adder

In this section, you will implement a one-digit BCD adder in VHDL. A one-digit BCD adder adds two four-bit numbers represented in a BCD format. The result of the addition is a BCD-format 4-bit output, representing the decimal sum, and a carry that is generated if this sum exceeds a decimal value of 9 (see slides of Lecture #11).

5.2.1 Structural Description of a BCD Adder in VHDL

In this part, you are required to implement the BCD adder using structural description. You are allowed to use either behavioral or structural style of coding for your implementation. Using the following entity definition. Use the following entity declaration.

```
library IEEE;
use IEEE.STD_LOGIC_11164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity bcd_adder_structural is
port ( A: in std_logic_vector (3 downto 0);
       B: in std_logic_vector (3 downto 0);
       S: out std_logic_vector (3 downto 0);
       C: out std_logic);
end bcd_adder_structural;
```

After you have implemented the one-digit BCD adder in VHDL, you are required to test your circuit. Write a testbench code and perform an exhaustive test of your VHDL structural description of the one-digit BCD adder.

5.2.2 Behavioral Description of a BCD Adder in VHDL

In this part, you are required to implement the BCD adder using behavioral description. You are encouraged to base your code on the VHDL code in Section 5.7.3 of the textbook, so that you learn about conditional signal assignments (these are explained in detail in the same section as well as in the Appendix in Section A.7.4). Use the following entity declaration.

```
library IEEE;
use IEEE.STD_LOGIC_11164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity bcd_adder_behavioral is
port ( A: in std_logic_vector (3 downto 0);
B: in std_logic_vector (3 downto 0);
S: out std_logic_vector (3 downto 0);
C: out std_logic);
end bcd_adder_behavioral;
```

After you have implemented the one-digit BCD adder in VHDL, you are required to test your circuit. Write a testbench code and perform an exhaustive test for your VHDL behavioral description of the one-digit BCD adder.

6 Questions

1. Briefly explain your VHDL code implementation of all circuits.
2. Show representative simulation plots of the 4-bit circular shift register circuits for a given input sequence, *e.g.*, “1011” (X = “1011”), for all the possible shift amounts.
3. Show representative simulation plots of the half-adder circuit for all possible input values.
4. Show representative simulation plots of the full-adder circuit for all possible input values.
5. Show representative simulation plots of both behavioral and structural descriptions of the 4-bit RCA for all possible input values.
6. Show representative simulation plots of both behavioral and structural descriptions of the one-digit BCD adder circuit for all possible input values.
7. Report the number of pins and logic modules used to fit your designs on the FPGA board.

	4-bit circular barrel shifter		RCA		One-digit BCD adder	
	Structural	Behavioral	Structural	Behavioral	Structural	Behavioral
Logic Utilization (in LUTs)						
Total pins						

7 Deliverables

You are required to submit the following deliverables on MyCourses. Please note that a single submission is required per group (by one of the group members).

- Lab report. The report should include the following parts: (1) Names and McGill IDs of group members, (2) an executive summary (short description of what you have done in this VHDL assignment), (3) answers to all questions in previous section (if applicable), (4) legible figures (screenshots) of schematics and simulation results, where all inputs, outputs, signals, and axes are marked and visible, (5) an explanation of the results obtained in the assignments (mark important points on the simulation plots), and (6) conclusions. Note - students are encouraged to take the reports seriously, points will be deducted for sloppy submissions. Please also note that even if some of the waveforms may look the same, you still need to include them separately in the report.
- Project files. Create a single .zip file named `VHDL#_firstname_lastname` (replace `#` with the number of the current VHDL assignment and `firstname_lastname` with the name of the submitting group member). The .zip file should include the working directory of the project.