VHDL Assignment #4: Design and Implementation of a 4-Bit Comparator

In this VHDL assignment, you will describe a 4-bit comparator circuit in VHDL and simulate it using the EDAPlayground platform.

1 Instructions

- Due date: Friday, November 6, 2020 by 5:00pm.
- 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 lab you should know how to

- Describe a circuit using sequential assignment statements
- Simulate digital circuits using the EDAPlaygrounf platform

3 Sequential Assignment Statements

In the previous VHDL assignments, you have learned several types of assignment statements such as simple assignment statements, selected assignment statements, and conditional assignment statements. All of these statements have the property that the order in which they appear in the VHDL code does not affect the meaning of the code, that is, it does not affect the synthesized circuit. For this reason, these statements are usually referred to as concurrent assignment statements.

VHDL also has a second category of statements, called sequential assignment statements, for which the order of the statements may affect the meaning of the code. In the following, we briefly discuss this new kind of statements. *They are described in more detail in Sections 6.6.6 and 6.6.7 of the textbook. Please read these sections before you attempt this assignment.* Additional details can be found in Appendix A.9 of the textbook.

The two main types of sequential assignment statements are: if-then-else statements and case statements. These sequential assignment statements must be placed inside a block, called a *process* block. The PROCESS block starts with the keyword "PROCESS". It has an optional label and a sensitivity list. Following the PROCESS keyword is the statement "BEGIN". Any statement between "BEGIN" and the "END PROCESS label;" are sequential statements.

```
label: PROCESS (sensitivity list)
BEGIN
--sequential statements
END PROCESS label;
```

The sensitivity list contains signals. Unlike concurrent statements which are executed all the time, the process block is activated only when one of the signals in its sensitivity list changes its value. Once activated, the statements inside the process block are executed *sequentially* in the order they appear in the block. There are two things to note: *(i)* Any assignments made to signals inside the process are not visible outside the process until all of the statements in the process have been evaluated; *(ii)* in case of multiple assignments to the same signal, only the last

one determines the final value of the signal. Also note that VHDL allows multiple processes to be described within the same architecture.

3.1 IF-THEN-ELSE Statements

IF-THEN-ELSE statements are used to modify the behavior of your function depending on whether one or more conditions hold. The syntax of IF-THEN-ELSE statements is shown below. Note that the "END IF" must be separated by a space.

```
IF condition THEN
-- sequential statements
ELSE
-- sequential statements
END IF;
```

You may have nested IF and ELSE as follows:

```
IF condition THEN
-- sequential statements
ELSIF condition THEN
-- sequential statements
ELSIF condition THEN
-- sequential statements
ELSE
-- sequential statements
END IF;
```

In this structure only one of the branches is executed depending on the condition. Even if there are several conditions which are true in this structure only the first TRUE condition will be followed. After the execution of the sequential statements within the first true condition the statements after the END IF will be executed next. Therefore it is very important to write the order of IF blocks and ELSIF blocks according to your desired behavior.

3.2 CASE Statements

CASE statements consider all of the possible values that an object can take and execute a different branch depending on the current value of the object as shown next.

```
CASE object IS
WHEN value1 =>
-- statements
WHEN value2 =>
-- statements
WHEN value3 =>
-- statements
-- etc.
WHEN OTHERS =>
-- statements
END CASE;
```

Note that the CASE statement must include a WHEN clause for each of the possible values of object. This necessitates a WHEN OTHERS clause if some of possible values of object are not covered by WHEN clauses.

4 4-Bit Comparator

Comparators are a useful type of arithmetic circuits that compare the relative sizes of two binary numbers. In this assignment you will implement a 4-bit comparator circuit that takes two 4-bit unsigned inputs A and B, and determines which one of the cases $A = (B+1), A < (B+1), A \leq (B+1), A > (B+1), A \geq (B+1)$ holds. It should detect the occurrence of overflow when performing $B+1$, *i.e.*, detect if $B+1$ requires more than 4 bits. Use the following entity declaration for your implementation of the comparator circuit.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity four_bit_comparator is
Port (A, B        : in  std_logic_vector(3 downto 0);
AgtBplusOne       : out std_logic;
AgteBplusOne      : out std_logic;
AltBplusOne       : out std_logic;
AlteBplusOne      : out std_logic;
AeqBplusOne       : out std_logic;
overflow          : out std_logic);
end four_bit_comparator;
```

Note that in case of overflow when performing $B+1$, the comparator circuit outputs 1 for the `overflow` signal while the remaining signals (*i.e.*, `AgtBplusOne`, `AgteBplusOne`, `AltBplusOne`, `AlteBplusOne` and `AlteBplusOne`) are set to 0. Otherwise, the circuits outputs proper values for `AgtBplusOne`, `AgteBplusOne`, `AltBplusOne`, `AlteBplusOne` and `AeqBplusOne` signals according to $A > (B+1), A \geq (B+1), A < (B+1), A \leq (B+1), A = (B+1)$, respectively, while the `overflow` signal is set to 0. For example, if $A = 9_{10}$ and $B = 5_{10}$ then both `AgtBplusOne`, `AgteBplusOne` should be 1, while the rest (including `overflow`) should be 0.

To describe your 4-bit comparator in VHDL, use sequential statements in a single process block. Once you have described your circuit in VHDL, simulate and verify the circuit.

5 Deliverables

Once completed, you are required to submit a report explaining your work in detail, including answers to the questions related to this lab. You will find the questions in the next section. You must also submit all of your VHDL, run.do, and .sdc files along with Log content of the synthesized circuits and timing analysis. You must also submit all 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.

6 Questions

Please note that even if some of the waveforms may look the same, you still need to include them separately in the report.

1. Briefly explain your VHDL code implementation of all circuits.
2. Given that $A = 5_{10}$, provide a **separate** simulation plot that demonstrates all possible cases for the 4-bit comparator, including a separate plot for the case where overflow occurs. A total of four plots should be included. Explain each plot and mark all inputs and outputs clearly.
3. Perform timing analysis of the 4-bit comparator and find the critical path(s) of the circuit. What is the delay of the critical path(s)?
4. Report the number of pins and logic modules used to fit your 4-bit comparator design on the FPGA board.