EE20021 Coursework – Activity 1
Microprocessor Program Counter (PC)

Dr Uriel Martinez-Hernandez

Introduction

You will design and implement a Program Counter (PC) that will be part of a microprocessor. The name of your module will be ***ProgramCounter*** and should have the port names described below. To get started, download the zip file *'Files for CW1-ProgramCounter'* from Moodle, where you will find the following files:

- *ProgramCounterTestBench.sv* – A template testbench that you will need to complete to test the functionality of your module
- *PcFpgaTop.sv* – A top module to implement your PC in the DE1-SoC board

Design your PC with the following connections and capabilities:

- ***Clock*** – All the operations in your module are synchronised with the positive edge of this input signal.
- ***Reset*** – When this input is high, your design should synchronously set the *CounterValue* output signal to 0.
- ***LoadValue*** – This 16-bit signed input value is loaded to the *CounterValue* signal on every positive edge of the *Clock* at which the *LoadEnable* signal is high.
- ***LoadEnable*** – This is an active high input signal and indicates that the value in *LoadValue* should be stored into the *CounterValue*.
- ***Offset*** – This 9-bit signed value is added to (or subtracted from, depending on its sign) the *CounterValue* on every clock cycle when the *OffsetEnable* signal is active.
- ***OffsetEnable*** – This is an active high input signal and indicates that an offset specified by the *Offset* signal needs to be applied to the *CounterValue*.
- ***CounterValue*** – This 16-bit signed output signal is the value of the Program Counter.
- When *Reset*, *LoadEnable*, and *OffsetEnable* are all inactive the *CounterValue* should increment sequentially by 1 and synchronously with the clock signal.

Make sure that all port names of your ***ProgramCounter*** module match with the port names in the template testbench provided for this coursework activity.

Activity 1 – Design a Program Counter

Create a project named ***ProgramCounter*** in Quartus Primer and create a SystemVerilog file named ***ProgramCounter*** for the design your program counter module. This module must interface directly with the testbench provided, so create a module with all the connections listed above.

Activity 2 – Parallel load

Adapt your design to use the *LoadValue* and *LoadEnable* signals.

Activity 3 – Offset

Adapt your design further to use the *Offset* and *OffsetEnable* signals.

Activity 4 – Testbench

Complete the testbench provided ***(ProgramCounterTestBench.sv)*** to simulate and test the functionality of your PC with Questa. The testbench should test the full functionality of your PC (e.g., sequential increment, reset, parallel load, etc).

Activity 5 – Implementation

Use *PcFpgaTop.sv* to implement your design on the DE1-SoC board. Have a look at this file to see which physical buttons and switches are connected to the Program Counter in order to test your module in real-time.

IMPORTANT

You have the option to work in a group of ***2 students maximum***. Add your name (and the name of your teammate if you worked in a group) at the top of all the SystemVerilog files developed in this coursework activity.

Add clear comments and detailed description of all the components of your module, and do not forget to use meaningful names for the variables, ports, etcetera. Also remember that it is important to indent your code to make it easier to read by others. All these aspects will be considered for the mark of this module.