CSE125, Spring'20, Lab 4: Pipelined CPU design

Due Date: 05/12/20

1 Introduction to DINO CPU

DINO CPU is a full RISC-V CPU implementation in Chisel, a derivative of Scala designed specifically for as a hardware description language (HDL). In this lab, you will be provided a complete single cycle RISC-V processor design and are tasked to implement the classic RISC-V five stage pipeline in Chisel. If you do not have a complete understanding of the five stage pipeline control and data flow please review it before continuing.

2 Installing Chisel and Supporting Tools

DINO CPU provides a Singularity (comparable to Docker or Vagrant) image which contains all tools needed to run DINO CPU is. It is recommended to not use this as it appears to be outdated. Instead please compile the software from source. If you are on Windows, please use either a virtual machine or the Windows Subsystem for Linux (WSL).
Start by installing the Scala Build Tool (sbt). This is likely available from your package manager's repositories, but the binaries can be found here: https://www.scala-sbt.org/download.html, and the source can be found here https://github.com/sbt/sbt. Confirm sbt is installed correctly by running the command:

```
sbt
```

You will also need Java Runtime Environment 8 (JRE8). As DINO CPU uses some legacy packages it is important that you use specifically version 8 if you are having issues with your setup. You will need to install the five Chisel packages with sbt:

- Chisel 3: https://github.com/freechipsproject/chisel3
- FIRRTL: https://github.com/freechipsproject/firrtl
- interpreter: https://github.com/freechipsproject/firrtl-interpreter
- Chisel-tester: https://github.com/freechipsproject/chisel-testers
- Treadle: https://github.com/freechipsproject/treadle

To install each package, clone the appropriate repository, cd into the cloned directory, run

```
sbt compile && sbt publishLocal
```

By default, this will install the package into your users ~/.ivy2 directory. After this, clone the dinocpu directory repo:
https://git.ucsc.edu/cse125/spring20/lab4

3 Usage

In order to start DINO CPU, navigate to your lab 4 (cloned from git.ucsc.edu) root directory and run sbt. This will drop you into an interactive session which you can run test cases in. By running the command:

```
test
```

You will be able to run all test cases for the CPU. This can be time consuming and instead you can run tests specific to the instruction you are implementing for example:

```
Lab3 / testOnly dinocpu.JumpTesterLab3
```

will test only the jump instruction.

4 Deliverables (20 Points)

For this lab, you will be required to implement your CPU as specified in the third DINOCPU lab which you find here:
https://github.com/jlpteaching/dinocpu/blob/master/assignments/assignment-3.md.

The linked DINO CPU assignment document will walk you through the implementation of required feature. You are required to complete this assignment. In order to get full points, you will need to ensure your design passes each of the provided test cases. The only files you are required to modify are:

- /src/main/scala/pipelined/cpu.scala
- /src/main/scala/components/hazard.scala
- /src/main/scala/components/forwarding.scala

cpu.scala instantiates components in the /src/main/scala/components directory including hazard.scala and forwarding.scala. As expected hazard.scala and forwarding.scala are control units which configure various muxes throughout the design to enable data forwarding and hazard flow control. Details for how to implement each of these files are found in the DINO CPU assignment page. Take a look at

https://github.com/jlpteaching/dinocpu/blob/master/assignments/assignment-2.md which walks through the implementation for the single cycle version of the CPU. You can use the single cycle version as a starting point, as much of the logic is the same. The completed single cycle CPU is provided in /src/main/scala/single-cycle/cpu.scala for reference.
Please disregard any comments about grading, point value, or submissions on the DINO CPU assignment page.

Push your design to git.ucsc.edu and submit the git hash via Canvas.

5 Tips and Tricks

This guide contains everything you need to know about Chisel to complete this assignment https://github.com/freechipsproject/chisel3/wiki/Cookbook. Although Chisel has a focus on higher level abstraction and object orientation, you will find it is not difficult to convert most verilog code to Chisel code in a one-to-one manner. It is recommended to avoid this and to make use of the appropriate abstractions, or create your own. The code you write in this lab will be used in the Lab 5, so do your best to make your code as flexible and robust as possible.
Although it is not required, the Chisel Diagrammer https://github.com/freechipsproject/diagrammer is a good tool to visualize your design much like an elaborated design view in Vivado. This is a good way to quickly perform sanity checks to ensure behavior such as wire connections are what you expect post compilation. Lastly, the 5-stage RISC-V pipeline is covered in detail in: Computer Organization and Design RISC-V Edition: The Hardware Software Interface, by Hennesy & Patterson. (You can find a copy online).