ADS I Class Project **WS 2025/26**
Hardware Design with Chisel
Prof. Dr.-Ing. Wolfgang Kunz RPTU Kaiserslautern-Landau
M.Sc. Tobias Jauch Fachbereich Elektrotechnik und Informationstechnik
Andro Mazmishvili Entwurf informationstechnischer Systeme

Assignment 3 Pipelined RISC-V Core

The goal of this task is to implement a 5-stage pipeline that features a subset of RV32I (all R-type and I-type instructions). The pipeline features five stages that run in parallel, but is not (yet) equipped with a unit for hazard detection and resolution.

Structure of the Project

Before you start with the tasks, make yourself familiar with the directory for the third assignment (03_RV32I-pipeline) in your forked repository. The Chisel project already contains a skeleton of the core. PipelinedRISCV32I.scala serves as a wrapper to connect the core tile to the testbench, all implementations will be done in core.scala and the files for the individual modules.

The wrapper interacts with the testbench in PipelinedRISCV32I_tb.scala. The testbench runs a simplified software program by passing the BinaryFile to the core. The core reads the content of the binary file into its instruction memory and executes the instructions line by line. The test program in BinaryFile consists of 32-bit RISC-V assembly instructions encoded in HEX. BinaryFile_dump does not contribute to the testbench, but lists the assembly instructions in a human-readable way.
Remember, the testbench itself (PipelinedRISCV32I_tb.scala) only checks the expected results of the instructions and runs the clock. When adding new instructions to the test program, remember to also add a check for the corresponding result to the testbench.

Specification

The goal of this task is to implement a pipelined 5-stage 32-bit RISC-V processor supporting all R-type and I-type instructions of the RISC-V instruction set architecture (pp. 26 – 40). The RV32I core is kept very basic and does not include features like memory operations, exception handling, or branch instructions. It is designed for a simplified subset of the RISC-V ISA, mainly focusing on ALU operations and basic instruction execution.

Task 3.1 Preparation

Before you start with the implementation, please answer the following questions:

1. How many ports does the register file need in a pipelined processor?
2. What test cases (i.e., combination of instructions) will lead to a problem in this pipelined processor that would not be an issue in a multi-cycle processor?
3. What are typical corner cases to be tested in this pipelined processor?

Task 3.2 Implementation

Read the the RISC-V Instruction Set Manual, Volume I carefully and implement all R-type and I-type instructions of the 32I-Extension in a pipelined 5-stage processor core according to the specification provided in the files of the Chisel project.

Task 3.3 Test your Implementation

Check whether your design runs and produces correct results by running the provided Chisel test cases. Add more test cases to include more scenarios and corner cases. The test bench is located in PipelinedRISCV32I_tb.scala, test cases need to be added in HEX assembly to the “‘Binary File”’ in the “‘programs”’ folder.

Prof. Dr.-Ing. Wolfgang Kunz RPTU Kaiserslautern-Landau
M.Sc. Tobias Jauch Fachbereich Elektrotechnik und Informationstechnik
Andro Mazmishvili Entwurf informationstechnischer Systeme

Please note that based on our own experience, common generative AI tools (GitHub Co-Pilot, ChatGPT, etc.) are not a trustful source when creating RISC-V assembly code. A RISC-V encoder and a RISC-V interpreter might be way more helpful for you in this task.

Task 3.4 Hazard Resolution

Draw a high-level schematic of the RV32I core you implemented and add how you would implement hazard detection and resolution for this pipelined core.