CMPE 446: 3rd Assignment

Ömer Faruk Erzurumluoğlu

February 2024

1 Introduction

In this assignment, we were asked to design an "hart" unit for a RISC-V processor which can do ALU, un/conditional branch, lui & auipc operations. The current state of the hart is only able to do ALU operations (add, xor, addi, etc.), branching (beq, bne, etc.), load-store operations (lb, sw, etc.), lui, auipc. (Currently the hart uses 3 cycles to process the instruction in a pipelined matter) It uses a 32-bit data cache to hold data for load-store instructions. The extra instructions (load-store) can be enabled/disabled by defining "EXTRA" in parameters.hpp file. Currently the synthesis & C-simulation works as intended but cosimulation goes into a loop and I have no idea how it happened or how to fix it. The EE does not even do branch prediction, it waits for the return value, I do not have a clue.

The current hart support RV32I instruction set but it still gives error for unimplemented instructions.

2 EE

Execution environment handles the errors & interactions with instruction cache and harts. For ease of use, the error handler can be changed by the user. Currently the error handler only handle illegal opcodes but I think in the next weeks it will be able to support most of the errors defined in the RV32I standard. All of the necessary calculations are made by the hart unit. The execution environment just fetches the instruction to run & feeds it into the hart module and the execution environment gets the next program counter value from the hart module. It currently does not predict branches.

3 HART

Hart module handles the given instruction & gives the next program counter as output. If an illegal instruction was given, it changes the last bit of the output program counter to 1 in order to inform the execution environment, since EE handles the errors. The current memory unit does not inform the hart or EE for all of the possible errors (ex./ cache miss). The hart is able to the arithmetic operations, un/conditional branches given by the RV32I standard. Th hart has the register file inside of it. Can not found a way to make sure that x0 is always 0 without using if-clauses while writing on the register file.

4 Sample Code

There are two test codes which one of them checks if all of the un/conditional branch operations & lui, auipc done write & the other one checks if the error handling works fine. In the code we use the assembled instructions which is hard to understand by humans. To make it more readable, I wrote their corresponding assembly as comment. They are in the test_hart.cpp file. There are two flags for running the samples: "LEGAL" & "DEBUG". "DEBUG" is for testing with more console outputs. They are in the parameters.hpp file. "LEGAL" is for running the sample code with only legal instructions. When undefined, it runs the sample code with the illegal opcode.

solution1 has "LEGAL" cosimulated & solution_illegal has illegal cosimulated, currently it does not works as intended. The pipeline pragma doe not change anaything & since current EE does not do predictions, the hart can not use its potential. Since illegal instruction gives an error, the we were not able to finish it. Their respective simulation files are in their respective folders.