EE480 Assignment 1: AIK Gr8BOnd Assembler Specification

Implementor's Notes

DYLAN ETRIS, CHACE RITCHIE, and SETH SMITH

This project was to implement a simple pipelined design for the Gr8BOnd processor, which demonstrates our knowledge of Verilog and pipelining.

1 GENERAL APPROACH

Our intended design includes three modules: an ALU, a converter, and the processor itself. Within the processor, we plan to have 3 always statements that act as each stage of the pipelined process. The first stage, stage 0, updates the program counter and retrieves the instruction from instruction memory. The value for the PC register at this point is used at a later stage, stage 2, where the PC may be added to a register for offset. The second stage, stage 1, reads from the register file to select the source register, rs, and the destination register, rd. In the case that an instruction is an immediate instruction, that value is saved for stage 2. The third stage, stage 2, prepares values for the ALU or data memory. In the case of a ALU instruction, a set of MUX's decide whether to use the PC value, immediate value, or the source and destination register. In the case of a memory access or store, the values of rs and rd are used to access data memory. This stage also handles instructions like bz and bnz, so a zero signal is carried out, and used to decide the source for the PC register. The final stage, stage 3, simply writes back to the register file, or in the case of PC update, to the PC register. Value forwarding was required between stages to verify that there were no write/read conflicts in the register file. To combat this, we used if statements that references instructions in previous stages. Verilog Functions were used to simplify the conditional statements by checking if an instruction updated the program counter or the zero register. Each stage has registers that it depends on and registers that it owns. It reads from the registers it depends on and writes to the registers that it owns. Values like the instruction were passed throughout each stage, from ir0 to ir1. In stage 1, the values of rd and rs are stored in rd1, rs1, etc.

2 ISSUES

Our design does not currently work, as we have yet to successfully implement it. It compiles, but it does not correctly execute instructions. One challenge we encountered was the division of labor. We thought that it would be simple to divide the labor by stages, with one groupmate doing a stage, but it proved difficult due to the interlinking between stages. We encountered a problem with the register read stage, as the code was falling into an infinite loop within that stage.

Authors' address: Dylan Etris; Chace Ritchie; Seth Smith.