CPE480 Assignment 3: Pipelined Tangled

Implementor's Notes

Gerard (Jed) Mijares
Tanner Palin
Sam
Department of Electrical and Computer Engineering
University of Kentucky, Lexington, KY USA

ABSTRACT

This is a pipelined implementation of a processor for the Tangled instruction set.

1. RUNNING PROJECT

We decided to run our Verilog locally, so a few commands differ from Dr. Dietz's web interface. A Makefile is included to facilitate running the project. To run different assembly programs, such as the files demonstrating the halting from Qat instructions, the `readmemh` instructions must be changed to the appropriate filenames.

2. PIPELINED APPROACH

Our processor implements 3 testing stages: instruction fetch, register read, and ALU/memory access/register store.

Stage 0, instruction fetch, first checks if stage 1 is prompting it to wait. If so, it sets the current instruction to a NOP. If stage 1 is not blocking stage 0, it fetches the instruction for the current program counter value. If either stage 0 or stage 1's instruction can modify the PC, then stage 0 stalls. Otherwise, the program continues with the fetched instruction and increments the PC.

Stage 1, register read, first checks if the instruction in its instruction register can set a new value into a register. If so, it prompts stage 0 to wait, as mentioned above.

Stage 2 implements the bulk of the processor. Most of the Verilog to operate each instruction was lifted from Jed's team's Multi-Cycled Design, as his had been tested.

3. FLOAT NAN SUPPORT

For many of the float instructions, NaN support was implemented by simply adding the ternary `?:` operator to check if an input was NaN and set the result to NaN if so. In some cases, the result of the original library assignments is stored on a `tempR` wire, which is then possibly selected depending on the value of an input. For example, `frecip` outputs NaN only if the input `a` is 0.

4. TESTING

Much of the test code used was written for the last project, but some new instructions were added as well, such as instructions to test the NaN operations.

5. ISSUES

We had issues completing some aspects of this project. First, the processor does not block only in the occurrence that there is a data dependence issue - it blocks every time that a destination register is written to, which could cause a data dependence issue. Second, in stage 1, while we do read the value of the source and destination registers, for certain instructions using those values causes errors. For these instructions we instead read directly from the register file in stage 2. Third, some of the float instructions fail to operate unless they have been padded with other instruction to stall before or afterwards. In the test file extra `lex` instructions have been added to allow the test cases to pass. Still, one of the float instructions that worked on the multi-cycled design refused to work on the pipelined design, the float operation. Fourth, halting does not seem to happen immediately, instructions may still be processed after a halt.

ACM ISBN N.A..
DOI: N.A.