Assignment 3: A Faster IDIOT Implementor's Notes

Casey O'Kane
casey.okane@uky.edu
Joe Weisbrod
jtwe226@uky.edu
Tristan Cheam
tlcheam01@uky.edu

***Abstract*—The goal of this assignment involved implementing the pipelined version of the IDIOT instruction set using the AIK assembler, the Verilog Hardware Design Language and detailed test plan to exhaustively test the different components and logic of the design.**

I. General Approach

Following the approach of previous assignments, the implementation of this assignment closely followed a heavily modulated design. Both the data and instruction memories where treated as modules along with the ALU incrementor, ALU module (for ALU instructions), the dependency detection module, the register file and the processor itself.

Similar to the previous assignment, global constants were declared both to represent common data structures (like 'WORD) and the different Opcode signals that could be received by the processor (like 'OpAdd or any of the instructions specified by the IDIOT ISA). Also, the AIK specification IDIOT_SPEC provided by Dr. Dietz following the submission of Assignment 2 was used for the sake of consistency.

The main point of difference for this assignment is the addition of the three buffer modules that are used to share information between modules of the different cycles (Instruction Fetch, Register Read, ALU/Memory, Register Write). These buffers are used to implement the pipelined design and their implementations are described in the next section.

While there are still some incomplete aspects of this assignment (these issues are discussed in the **Issues** section of these notes), the most recent diagram is shown in **Appendix A** at the end of these notes.

II. Implementation

This section describes how each module was implemented.

A. Pipeline Buffers

1) Instruction Buffer:

2) Register Buffer:

3) ALU/Write Buffer:

B. Proccesor

C. Memory

1) Instruction Memory:

2) Data Memory:

III. Testing

A. Verilog Modules

B. Testing Results

IV. Issues

A. Features Not implemented

1) Jump Based Instructions: Currently, there is no implementation concerning Jump based instructions JZ/SZ/SYS. Hopefully this is something that will change prior to submission.

2) Memory based instruction: There is no implementation of Load or store

3) Load Immediate Instruction: As the group was unable to account for the jump or memory based instructions until far later than anticipated, the group was unable to complete li instruction handling for the pipelined processor.

B. Known Errors

As it currently stands there are no known issues