RISCV32IMC Pipelined, Verilog Implementation

Computer Architecture Fall 2023: MS1

Ahmed Waseem Mohamed Adly Raslan Ahmed Elbarbary

October 5

Contents

1 Overview

RISC-V is an open and extensible instruction set architecture (ISA) that has gained popularity in recent years due to its simplicity and flexibility. The RV32IC and RV32M specifications provide the foundation for a basic 32-bit RISC-V processor with integer and multiply/divide instructions. We planned on implementing RV32IMC spec for MS2.

2 Methodology

As we stated in Milestone 1, We started A top-down approach for our design, we figured the lab code required a lot of refactoring, so we rewrote most modules with proper best practices. Specifically for Milestone 2 we reused our single cycle processor, added decompression logic, optimizations and pipeline functionality to it. This proved easier and more time-effiecient given the timeframe.

Most modules pass linting tests for Xilinix chips specifically virtex2 rule sets.

The Design is implemented as is in blockdiagram.pdf.

3 Implementation

Implementation was done in Verilog, and tested by either using Vivado or Gtkwave and Iverilog.

Appending the datapath for more instructions was a matter of adding multiplexer, adders or functionality to CU/ALU as outlined in the block diagram pdf.

4 Testing Methodology

Testing was done in increments, We improved upon our testing procedures of MS1, for time efficiency we implemented a python script that stages our desired test cases automatically, using Vivado's file interface (steps on usage in repo readme.md). we formally defined test cases that targeted certain grouping of instructions, functionality and data hazards. with all the outlined tests passing expected.

We were not able to test the processor on the FPGA due to time constraints, but we are confident that it will work as intended as it passed linting tests.

4.1 Future testing objectives

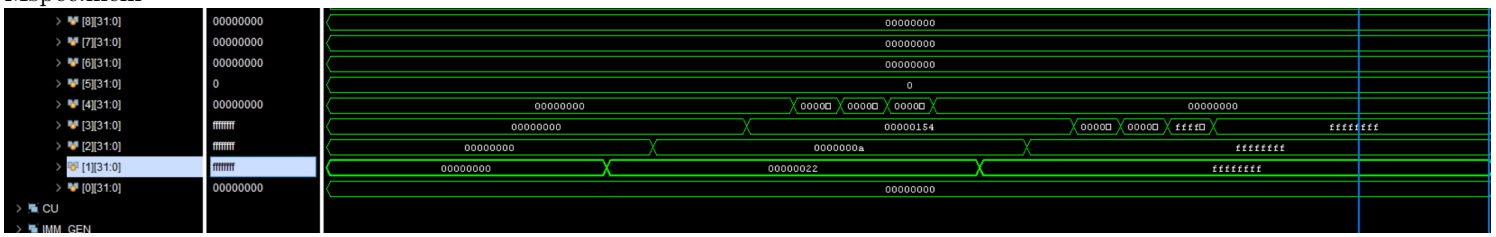
Using Verilog's file interface we think we could make a more robust testing methodology, either a golden randomized self checking test-bench (which was in plan but was left blank due to time constraints), or a implement riscv un-official testing suite programs.

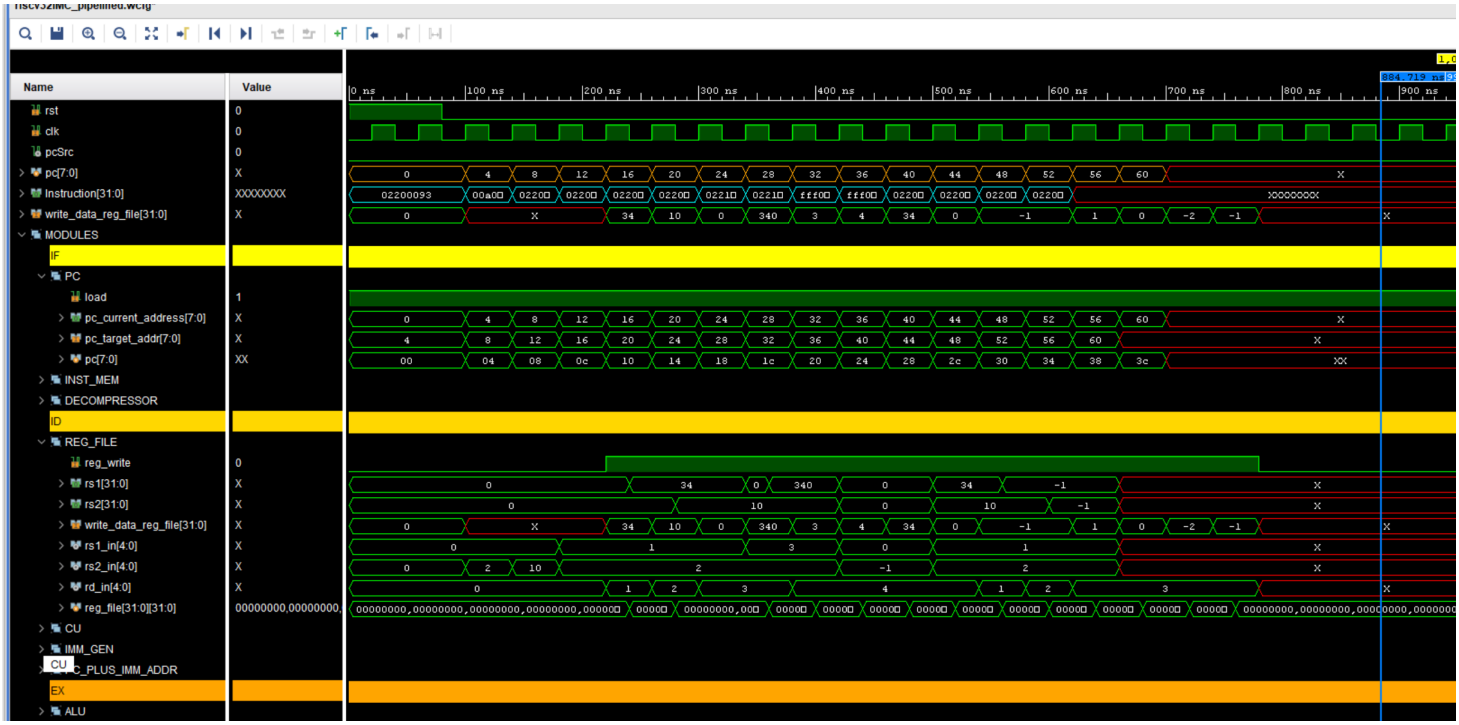
5 Results

The execution of all test files was successful, showcasing comprehensive support for RISC-V instructions and multiplication. It's worth mentioning that, given time constraints, support for compressed instructions was limited to one. Additionally, the flushing functionality performed effectively.

5.1 simulation results

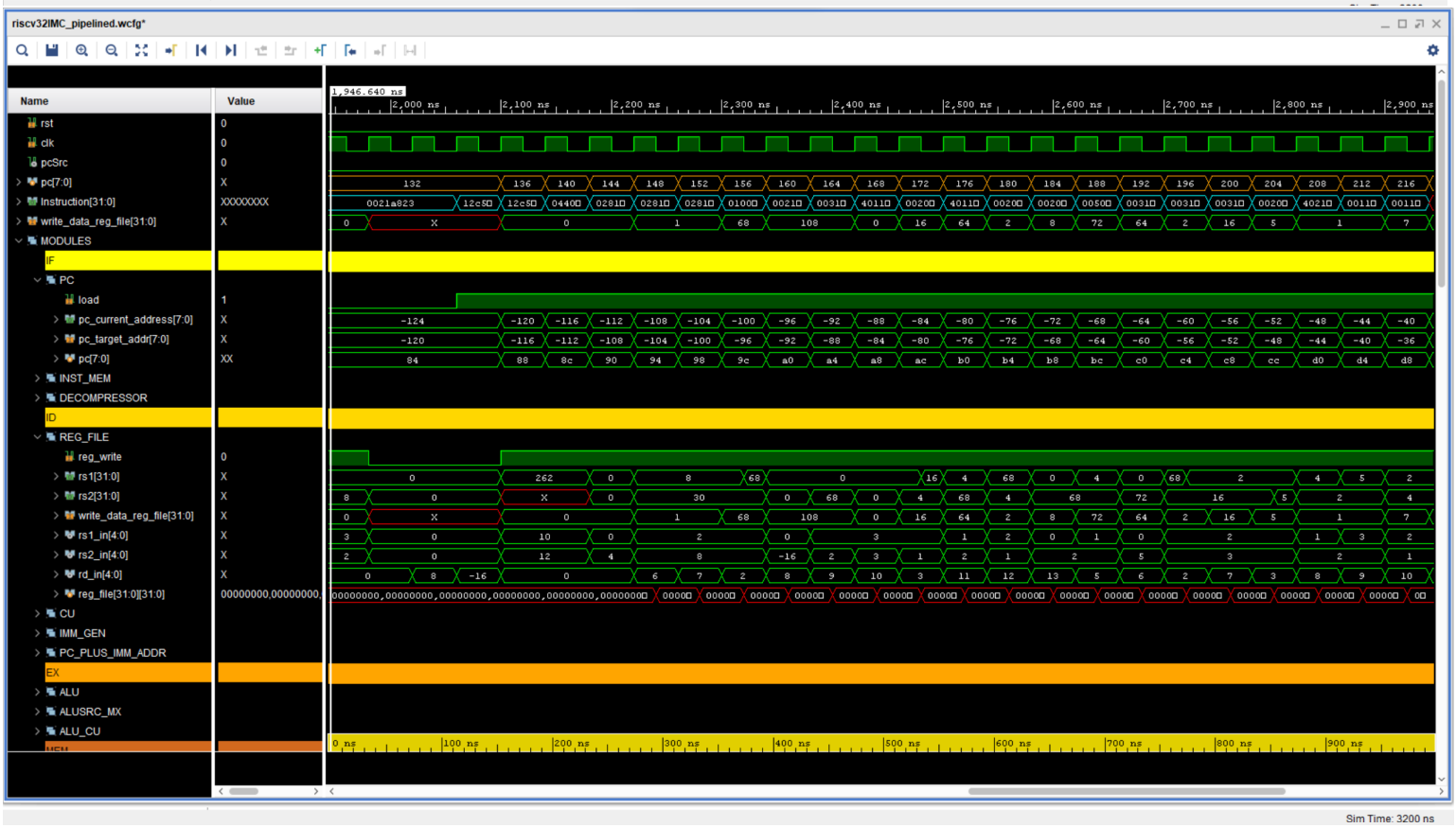
Mspec.mem



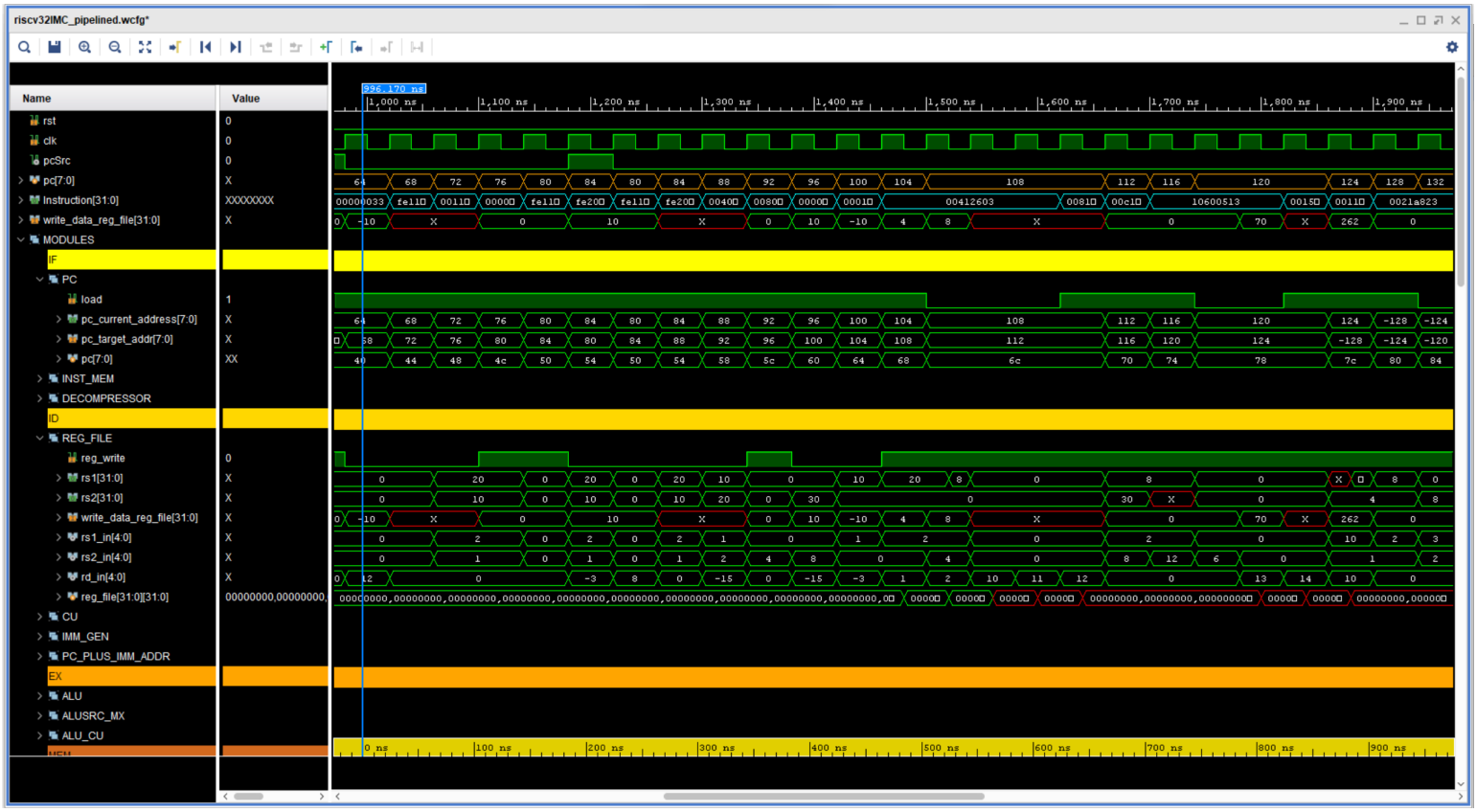

Name
Value
rst
clk
pcSrc
pc[7:0]
Instruction[31:0]
write_data_reg_file[31:0]
MODULES
IF
PC
load
pc_current_address[7:0]
pc_target_addr[7:0]
pc[7:0]
INST_MEM
DECOMPRESSOR
ID
REG_FILE
reg_write
rs1[31:0]
rs2[31:0]
write_data_reg_file[31:0]
rs1_in[4:0]
rs2_in[4:0]
rd_in[4:0]
reg_file[31:0][31:0]
CU
IMM_GEN
PC_PLUS_IMM_ADDR
EX
ALU

all.mem

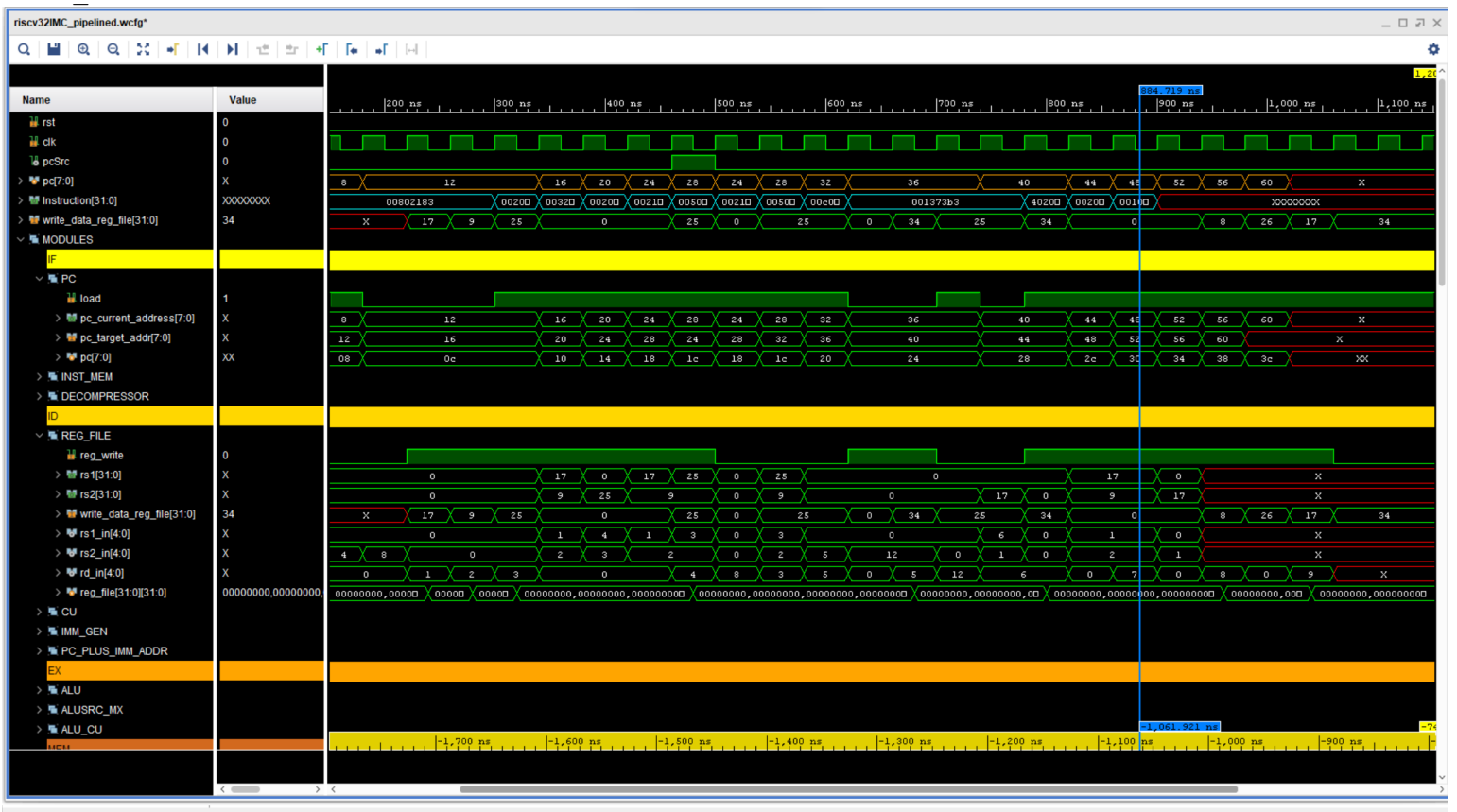

riscv32IMC_pipelined.wcfg*
Name
Value
rst
clk
pcSrc
pc[7:0]
Instruction[31:0]
write_data_reg_file[31:0]
MODULES
IF
PC
load
pc_current_address[7:0]
pc_target_addr[7:0]
pc[7:0]
INST_MEM
DECOMPRESSOR
ID
REG_FILE
reg_write
rs1[31:0]
rs2[31:0]
write_data_reg_file[31:0]
rs1_in[4:0]
rs2_in[4:0]
rd_in[4:0]
reg_file[31:0][31:0]
CU
IMM_GEN
PC_PLUS_IMM_ADDR
EX
ALU
ALUSRC_MX
ALU_CU



Sim Time: 3200 ns



sevens_instructions.mem



5.2 Current status

All instructions tested passed, The Compressed handling logic is implemented but the only supported instruction is *c.addi*.

5.3 Milestone Review

The delay in delivery is the elephant in the room. It was due to time constraints from other courses, both of us had a project submission the day of the submission and a midterm the day before. We are however very pleased with our implementation and our workflow and given better circumstances we could have really have improved on our design.