ECE 552 Extra Credits Reports

Author: Jarvis Jia, Josh Mei

Based on our 5-stage pipeline design with caches, we implemented some optimization based on given choices

Branch/Jump Decisions in Decode

In phase 3 design, we put the branch/jump decisions in the execution stage. After implementing the pipelined design, the branch/jump decisions will increase the performance and save area since we don't need to pipelined the branch/jump taken signal to the execution stage.We added another test by ourselves named branch_jump_test.asm to further test our code

By doing this, we found out that there are around 0.1-0.2 cycle reduction for our CPI.

Additional Forwarding Paths (MEM-ID, MEM-MEM)

In the forwarding unit, we add MEM to MEM forwarding when the previous instruction is stored and the next instruction is loaded, so it could reduce the cycle time between these two instructions.By doing this, there is no noticeable CPI reduction. Since we changed the branch and jump decisions from the execution stage into the decode stage, so we have to add MEM to ID forwarding to make that implementation possible. By doing this, there is no noticeable CPI reduction.

Exception Handling

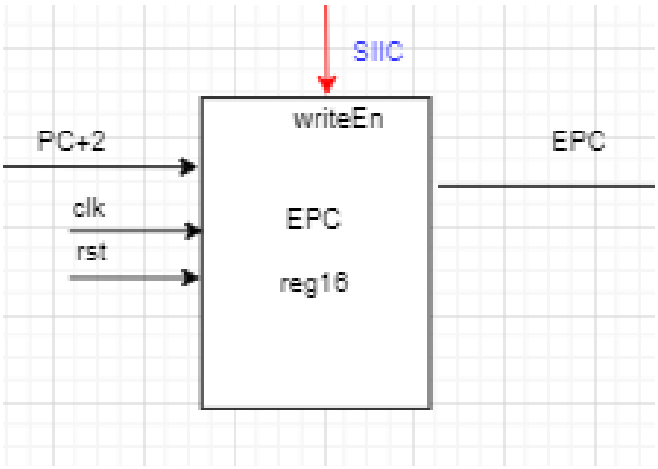

Figure 1. EPC Register

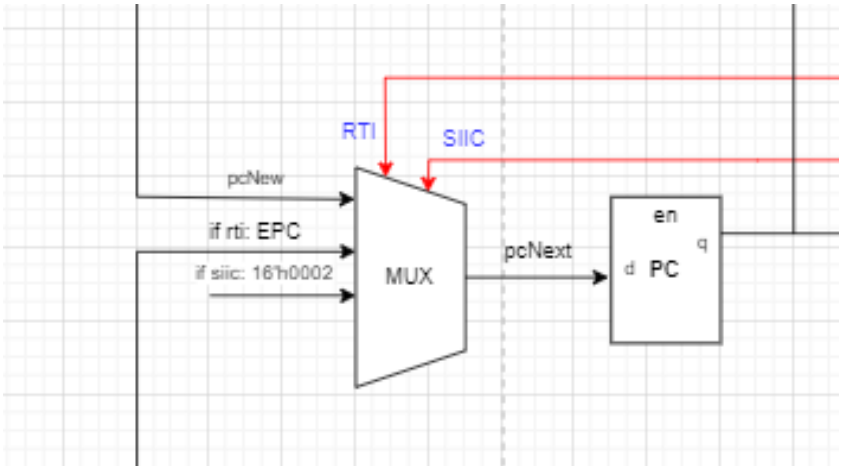

Figure 2. PC input MUX with EPC Handling

We have added an exception handler in our decode stage, so now we can pass all the extra credit tests provided by the professor including the SIIC_test, simple_exception_test and complex_exception_test. To further test if our exception is working correctly, we have written our own tests naming exception_test.asm

Design Synthesis

We have synthesized the design. After the first try, we have 20 violations in timing report, after analyze the critical path, we found that after pipelining all the signals, the combinational logic signals in ALU/forwarding unit/memory_system running to fast, which arrive earlier before setup time of DFFs, therefore, by adding the flops to pipeline the combinational signals, we cancel all of the violations, make it meet all the time requirements.

To optimize the area for our design,we removed the unused logic in “decode stage ALU” (which is for solving the branch/jump decisions), We also optimized the adder logic, we substitute the full adder with half adder, since half adder has no carry in/out logic, then CLA will solve the carry in/out logic.

Directory structure

Verilog/

Verification/

Synthesis/