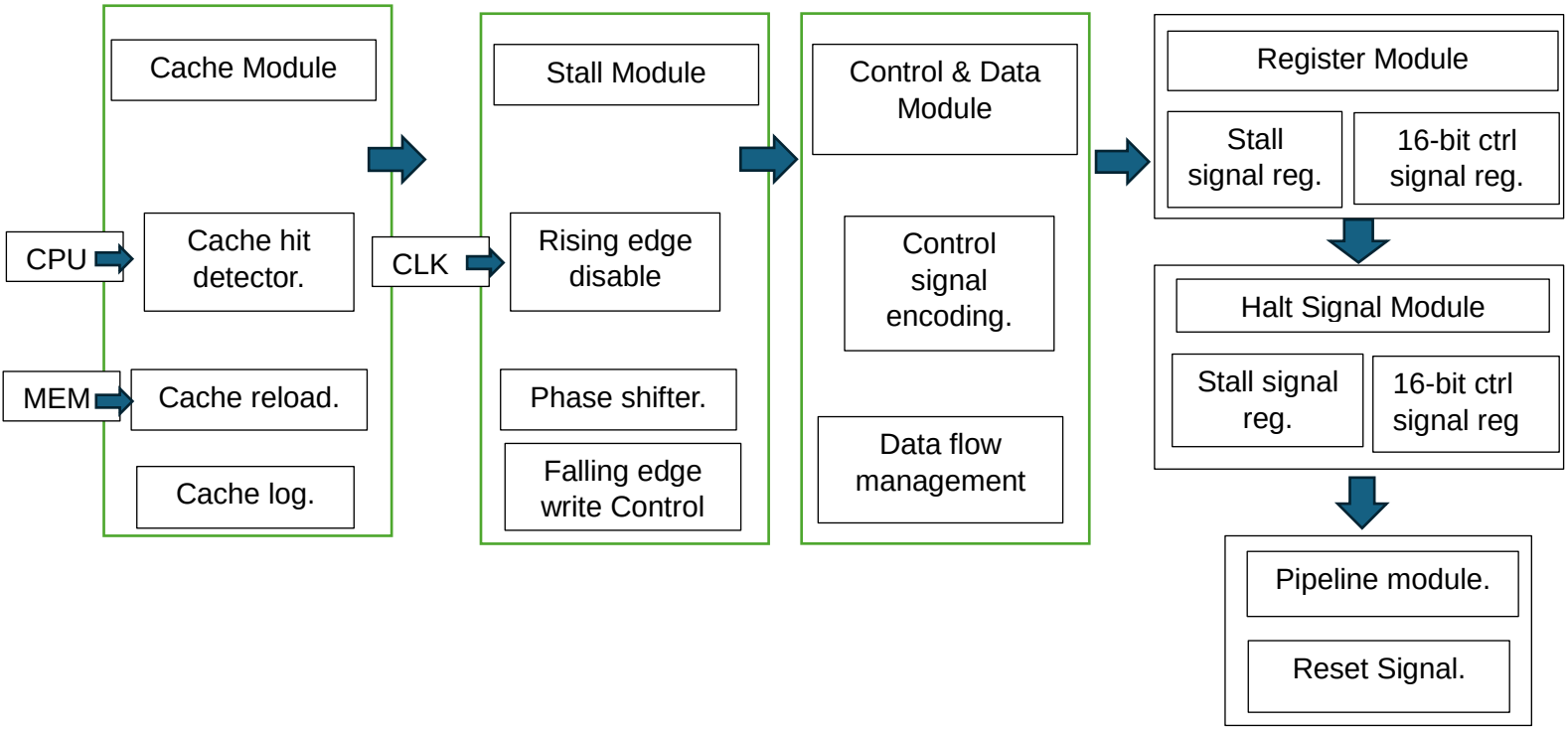
ECE 552 Project Report

By Station 02

Overview

This ASCII diagram shows the block diagram layout of the modules you described, with arrows indicating data flow and dependencies between them.



This project introduces a 5-stage pipelined processor for WISC-S24 ISA, incorporating specialized features such as Cache architecture, enhanced stall mechanism, separate control, and data flow management, and optimized control signal handling. The Cache system operates under cache hit conditions, with misses triggering reloads, impacting hit rates. The stall mechanism disables clock rising edges, improving pipeline control. Separate control/data flow enhances parallel processing efficiency. Control signals are consolidated into a 16-bit register for streamlined management. The halt signal ensures proper program termination across stages. Load-to-use stall hazards are addressed by resetting pipeline registers. Verilog is used, and simulation tools include Modelsim or Icarus. Deliverables include Verilog files, testbenches, support files, and performance logs.

Responsibility/Task Breakdown

Ziheng Ding	Muhammed Alimi
Data Path Implementation	Control signal Implementation
Testing (Debugging and test benches)	Testing (Including test benches)
Cache Implementation	Project Report
Testing and Validating all Test cases	Prepared all test cases files for testing
Design Architecture	Testing and Validating all Test cases

Special Features

- Our Cache architecture operates with memory managed under cache hit conditions. When a cache miss occurs, we load the cache and then repeat the memory operation. This reload of the cache affects the hit rate recorded in the log file. If an instruction initially causes a cache miss, once the memory is loaded, it will result in a cache hit.
- For our stall mechanism, we disable the rising edge of the clock.
- We managed control and data flow separately to allow parallel work for efficiency.
- We utilize a register to hold the stall signal, which is phase-shifted by half a cycle and written on a falling edge.
- All control signals are encoded into one 16-bit register for reusability, simplifying control signal management.
- The halt signal incorporates feedback to ensure it remains active across multiple stages.
- Regarding the load-to-use stall, we reset the EX to MEM pipeline register after this stall to address load-to-use hazards and maintain pipeline integrity.

Completeness

Our design meets all requirements for the three phases of the project. However, we faced challenges with the cache storage that was specified. We customized our cache storage to fit our architecture. Passed all benchmarks for all test cases except for test 4. Test 4 partially involves reading and writing to memory with some minor address errors. Specifically, the first two loops are intended to fill data from addresses 0x70 to 0x7F, and the subsequent loops from addresses 0x80 to 0x8F. However, due to address errors, the data at address 0x70 gets overwritten by data from 0x72, causing a shift in all subsequent data. A similar issue occurs with address 0x80. The root cause of this problem lies in data dependencies within the loop. The loop initially saves a word and then updates the address, leading to an incorrect address being forwarded to the next loop's save word operation. This issue prevented us from achieving the desired benchmark performance. Additionally, we encountered challenges with flag bit generation, where a flag bit generated by one instruction is immediately utilized by a subsequent branch instruction. To address this, we reversed the clock signal to ensure that the flag bit is written at the falling edge, mitigating the timing issue.

Testing Methodologies

The testing methodologies for the 5-stage pipelined processor project include unit testing for individual Verilog modules, integration testing for component interactions, functional testing for ISA compliance, performance testing for speed and efficiency, stress testing for robustness, simulation-based testing for detailed analysis, regression testing for change impact, and code coverage analysis for test completeness. These methods collectively ensure the processor's functionality, performance, reliability, and correctness.

Result

Test cases	Cycle count	Validity (*correct or incorrect*)
Test case 1	44	Correct
Test case 2	59	Correct
Test case 3	54	Correct
Test case 4	N/A	Incorrect