Project Plan

Team Name: Memory Leak
Sparsh Agarwal | agarwal39@wisc.edu
Joanne Lee | joanne.lee@wisc.edu

Project Design

Design Review [27 Feb]: Schematic of an unpipelined WISC-SP13 implementation

1. Identify required modules [Joanne/19 Feb]
2. Identify interface and datapath between modules [Sparsh/19 Feb]
3. Rough draft of schematic [Sparsh & Joanne/21 Feb]
4. Final schematic [Sparsh & Joanne/21 Feb]

(Update the documentation)

Demo 1 [14 Mar]: Single-cycle, non-pipelined implementation

1. ISA Implementation [Joanne/5 March]
2. Implement first 5 stages of cycle(load, read, execute, write, branch)[Sparsh/ 5 March]
3. Run test using given data sets[Joanne & Sparsh/ 7 March]
4. Synthesize processor and fix synthesis errors[Joanne & Sparsh/ 12 March]

(Update Schema)

Demo 2 [11 Apr]: Pipelined design with Perfect Memory(**Timing and area report required**)

1. Implement cycle for at least two instructions[Sparsh/20 March]
    a. Instruction Fetch (IF)
    b. Instruction Decode/Register Fetch (ID)
    c. Execute/Address Calculation (EX)
    d. Memory Access (MEM)
    e. Write Back (WB)
2. Implement it for multiple cycles and check for timing lags and latency in each module[Joanne/30 March]

Optional:

1. *Stalling Memory[Joanne & Sparsh/5 April]
2. *Aligned Memory[Joanne & Sparsh/7 April]

(Update the documentation, Test using given data sets)

Cache FSM turn-in [13 Apr]

1. Cache control for directly mapped cache of data and instructions[Sparsh/11 April]
2. Create FSM for cache controller [Sparsh & Joanne/12 Apr]

Cache Demo [20 Apr]: Working two-way set-associative cache

1. Design direct mapped cache: Compare Read/Write [Joanne/14 Apr]
2. Design direct mapped cache: Access Read/Write [Sparsh/14 Apr]

3. Test direct mapped cache [Joanne & Sparsh/15 Apr]
4. Synthesis of direct mapped cache and fix synthesis errors [Joanne & Sparsh/15 Apr]
5. Two-way set-associative cache: Compare Read/Write [Joanne/16 Apr]
6. Two-way set-associative cache: Access Read/Write [Sparsh/16 Apr]
7. Replacement algorithm [Joanne/16 Apr]
8. Test two-way set-associative cache [Joanne & Sparsh/16 Apr]
9. Synthesize cache design and fix synthesis errors [Joanne & Sparsh/16 Apr]
10. Schematic for design in each cache_*/ directory [Sparsh/18 Apr]

Optimization

1. No more than one of the following in series during any stage: register file, memory or cache, 16-bit full adder, barrel shifter. [Joanne/22 Apr]
2. All branches should be predicted not-taken. Pipeline should continue to execute sequentially until the branch resolves, and then squash instructions after the branch if the branch was actually taken. [Joanne/22 Apr]
3. Two register forwarding paths in the WISC-SP13; one within the ID stage and between the beginning of MEM and the beginning of EX. [Sparsh/22 Apr]

(Update the documentation)

Final demo[7 May]: Pipelined Multi-cycle Memory with Optimizations

1. Test using given data sets [Joanne & Sparsh/30 Apr]

Programs required for submission: (They can all be created by script run-final-all.sh)

- perf.summary.log
- complex_demofinal.summary.log
- rand_final.summary.log
- rand_ldst.summary.log
- rand_idcache.summary.log
- rand_icache.summary.log
- rand_dcache.summary.log
- complex_demo1.summary.log
- complex_demo2.summary.log
- rand_complex.summary.log
- rand_ctrl.summary.log
- inst_tests.summary.log

Final report[10 May]

Wrap up everything [Joanne & Sparsh/30 Apr]