

## 6.175: Constructive Computer Architecture

### Tutorial 7

# Final Project Overview

Quan Nguyen

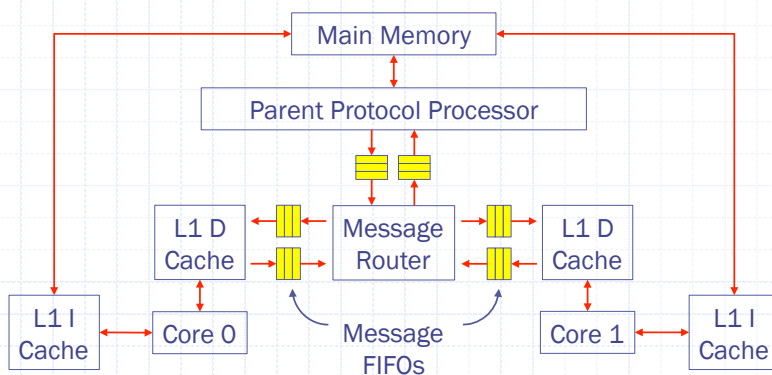
(Sometimes doesn't lose a game of 2048)

## Part 1: Store Queue

- Based on work of Lab 7
- Resolve rule conflicts in Lab 7
- Add store queue to blocking cache
- Allow load hit under store miss

## Part 2: Cache Coherence

- Final target system



Nov 18, 2016

<http://csg.csail.mit.edu/6.175>

T07-3

## Cache Coherence: Loads and Stores

- Implement each unit
  - Message FIFO
  - Message router
  - L1 data cache
  - Parent protocol processor (PPP)
- Test against simple unit testbench

Nov 18, 2016

<http://csg.csail.mit.edu/6.175>

T07-4

## Cache Coherence: Loads and Stores

- Tandem Verification
  - Reference model for cache-coherent memory hierarchy: monolithic memory
  - Debug interface passed to each core and D\$
    - ◆ issue: called when request is sent to D\$
    - ◆ commit: when request finishes processing, i.e. read/write D\$ data array; check correctness of response and cache line value

```
interface RefDMem;  
  method Action issue(MemReq req);  
  method Action commit(MemReq req, Maybe#(CacheLine) line,  
                       Maybe#(MemResp) resp);  
endinterface
```

## Cache Coherence: Loads and Stores

- Deficiency of tandem verification
  - Cannot check deadlock
- Testing whole memory hierarchy
  - Feed random data cache requests
  - Detect bugs with tandem verification
  - Add cycle counter to detect deadlock

## Cache Coherence: Loads and Stores

- Integrating to the processor
  - Three-cycle core: provided
  - Six-stage pipeline: from Lab 7
- Running multi-core programs
  - Similar to fork

```
int main() {  
    int coreid = getCoreId();  
  
    if (coreid == 0) {  
        return core0();  
    } else {  
        return core1();  
    }  
}
```

## Cache Coherence: Atomic Memory Instructions

- Add load-reserved (lr) and store-conditional (sc) to data cache
  - Only D\$ and processor pipeline need to be changed
- Run more programs

## Cache Coherence: Store Queue

- Add store queue to cache
  - Similar to part 1
  - Atomic instruction has special behavior
  - Programming model is not SC
  - Introduce fence to flush store queue
  - Allow load hit under store miss

## Schedule

- Part 1: tonight
- Part 2: after Thanksgiving
- Deadline December 14<sup>th</sup>
- Presentation in class (Dec 14<sup>th</sup>)
  - What problems/bugs you encounter
  - How you resolve them

# Project Part 1: Store Queue

## Problem with Lab 7

- Memory stage rule conflicts with D\$ rules
  - Memory stage is more urgent
  - Hurt performance: store miss processing is stalled

```
// processor memory stage
rule doMemory;
  dCache.req(...);
endrule
```

```
// data cache
rule startMiss(status == StartMiss); endrule
rule sendFillReq(status == SendFillReq); endrule
rule waitFillResp (status == WaitFillResp); endrule
method Action req(MemReq r) if (status == Ready); endmethod
```

## Resolving Conflicts

- Add 1-element bypass FIFO, reqQ
  - No performance loss (dead cycles)
- Request from processor placed in reqQ
- Separate rule processes reqQ

## Adding Store Queue

- Process new request: store queue or reqQ
  - reqQ.first == St → enq to store queue
    - ◆ Also process store from store queue
    - ◆ If not processing store queue, may deadlock
      - Store queue full
  - reqQ.first == Ld → process Ld
    - ◆ Store in store queue will not issue
      - Structural hazard
  - No need for the “lockL1” EHR in lecture

## Load Hit Under Store Miss

- Performance improvement of store queue is limited
- Make the cache somewhat non-blocking
  - Store miss: waiting for mem resp
    - ◆ Not accessing cache
    - ◆ Process a load at reqQ. first
    - ◆ If hit: resp to processor
    - ◆ If miss: keep it in reqQ
- Load hit is disallowed under load miss
  - Load resp cannot go out-of-order