Quiz 3 Review

6.823 Spring 2019

Ryan Lee

Adapted from: Mark Jeffrey, 2017
Topics Snapshot

» Cache coherence
  - Directory, snoopy protocols
  - MSI, MESI, MOSI
  - Synchronization

» Memory consistency models
  - Sequential consistency
  - Fences

» On-chip interconnection networks
  - Metrics (diameter, bisection bandwidth, avg distance)
  - Routing, flow control deadlock freedom
  - Router microarchitecture
Cache Coherence

» Problem: If multiple processors cache the same block, how do they ensure “correct” view of the data?

» Concerned with accesses to a single address.

Two rules:

1. Write propagation: Writes eventually become visible to other processors
2. Write serialization: Writes to same location are serialized
Coherence Protocols

- Invalidation vs Update-based
- Snoopy vs Directory-based
Coherence Protocols

» MSI states:
  - Modified (M): Cache has exclusive access to line with read, write permissions.
  - Shared (S): Cache has shared, read-only copy of line.
  - Invalid (I): Cache does not have copy of data

» MSI optimizations:
  - Exclusive (E) state
  - Owned (O) state
Synchronization Primitives

» Why? Sequencing actions among multiple processes. Used to implement:
  - Mutex (locks)
  - Condition variables
  - Semaphores

» Some synchronization primitives:
  - Test-and-set, Test-and-test-and-set
  - Load-reserve, store-conditional
  - Swap
  - Compare-and-swap

Different primitives have different properties, and present different implementation tradeoffs.
Compare-and-swap

**CAS** old, new, Imm(base):  
  if (old == Mem[Imm+base])  
    Mem[Imm+base] \leftarrow new  
  else  
    old \leftarrow Mem[Imm+base]

» *Atomically* loads value at effective memory address, and compares value to value in register old.  
  - If both values are equal, update memory location with value in register new  
  - Else, update value in old with value in memory
Load-reserve, store-conditional

```ml
LR rs, Imm(rt):
    <flag, addr> ← <1, Imm + rt>
    rs ← Memory[addr]

SC rs, Imm(rt):
    if <flag, addr> == <1, Imm + rt>:
        Clear other procs flag
        Mem[addr] ← (rs)
        status ← 1
    else:
        status ← 0
```

See handout 13. It shows you how to implement a lock using CAS.

L14, 15 show you how to implement locking using swap.

Different primitives have different properties, and present different implementation tradeoffs.
Memory Consistency Model

A memory (consistency) model specifies the order in which memory accesses performed by one thread become visible to other threads in the program.

» Contract between the hardware and software

» Loosely speaking, it specifies:
  - Set of legal values a load can return
  - Set of legal final memory states for a program
Memory Consistency Model

» Sequential Consistency (SC)
  - Maintain program order
  - Loads, stores appear atomic
  - “Strongest”, most intuitive model

» Weak Memory Models
  - Total Store Order (TSO)
  - Partial Store Order (PSO)
  - Relaxed Memory Order (RMO)

Enable several optimizations on the processor, memory system, interconnection network.
Memory Model Litmus Tests

Proc. 1
ST X <- 1
ST Y <- 1

Proc Y
LD Ry <- Y
LD Rx <- X

» What final values of Rx and Ry are possible under SC? TSO? PSO? RMO?
- Reason with happens-before relationships
- A cycle indicates disallowed ordering!
Memory Fences

» Idea: Not all accesses need to be “strictly” ordered. Programmer identifies regions which need (not) be ordered.

» Primitives that prevent otherwise permitted reorderings of loads and stores

» Different flavors on different systems:
  - Sparc: MEMBAR
  - x86: LFENCE, SFENCE, MFENCE
Network-on-Chip

» Handles communication between various on-chip elements: caches, directory, memory-controller.

» Several characteristics:
   - Topology
   - Flow control
   - Routing
   - Router Microarchitecture
Topology

» Arrangement of channels and nodes i.e. how different nodes connect to each other.
  - Ring, mesh, torus, multi-dimensional structures etc.

» Properties
  - Number of links
  - Diameter
  - Average distance (hops or links)
  - Average latency (consider latency of links and routers)
  - Bisection bandwidth
Routing

» Path from source to destination

» Properties:
  - Deterministic/oblivious
  - Adaptive
  - Minimal
  - Balanced
  - Deadlock-free

Some of these are often at odds with each other.
Routing Deadlock
Turn Model

The eight possible turns and cycles in a 2D mesh

Only four turns are allowed in the XY routing algorithm
Restrict 1 turn

All possible turns

Restrict 1 turn

West-First

North-Last

Negative-First

AND
Channel Dependency Graph (CDG)

Cycles in the CDG point to potential deadlock.

» Vertices in the CDG represent network links. Edges represent if a particular turn is allowed.
Channel Dependency Graph (CDG)

Deadlock free? No
Channel Dependency Graph (CDG)

Minimal routing
Deadlock free? Yes
Flow Control

» How network resources are allocated to packets traversing the network

» Bufferless
  - Circuit switching, dropping, mis-routing

» Buffered
  - Store-and-forward, virtual cut-through, wormhole, virtual-channel
Head-of-line (HoL) Blocking
Head-of-line (HoL) Blocking

Solution: Virtual Channels
Router Microarchitecture
Router Pipeline

1. Buffer Write (BW)
2. Route Compute (RC)
3. Virtual Channel Allocation (VA)
4. Switch Allocation (SA)
5. Switch Traversal (ST)
6. Link Traversal (LT)
Good Luck!