Announcements

Midterm Exam  Wed 2/26

Project Milestone 1
- Report: due Sun 3/8
- Meeting: Mon 3/9

PA2
- Waypoint Report: due Mon 3/16
- Final Submission: due Mon 3/30
1. Cache Coherence

2. PA 2 Overview

3. Murphi (Tutorial)
Cache Coherence

• Why?
  • In the presence of caches, orchestrate access to shared memory in a multi-core system

• What?
  • A load returns the most recent value written
  • For a single memory location only

• How?
  • Well, many many flavors!
Cache Coherence – How?

- Interconnection network
  - Bus: Snoop-based protocols
  - Point-to-point: Directory-based protocols

- Stable states?
  - VI, MSI, MESI, MOSI, MOESI

- Optimizations employed – countless papers!!
  - 3-hop vs 4-hop
  - Self-downgrade (M->S)
  - Cruise missile invalidations, etc.
Basic Directory Operation: Read

Node #1  Directory  Node #2

Load A (miss)

Get-S A

Data A

A: Shared, #1

42
Basic Directory Operation: Write

Node #1

Load A (miss)

Load A (hit)

.oops

42

Node #2

A: Shared, #1

96

Directory

Get-S A

Data A

Invalidate A

Get-M A

Data A

Store A (hit)
Basic Directory Operation: Write

Load A (miss)

Node #1

Get-S A
Data A
Invalidate A
Inv-Ack A

Directory

A: Shared, #1

Node #2

Get-M A
A: Mod., #2
Data A

Store A (hit)

42

96
Deadlock!

- Protocol deadlock
  - Wait for a message that is never sent
  - **Solution:** Design your state machine correctly

- Network deadlock
  - Coherence messages hold resources in circular manner
  - **Solution:** Dedicated virtual networks for different messages
Virtual Networks

- Solve network-dependent deadlocks
  - Have separate VN for every message class
Assignment 2

• Design a CC protocol → state transition diagram
• Learn a formal verification language
• Specify your CC protocol formally and verify it

Requirements
• Verify with at least 3 processors, 1 memory location
• Connected via an arbitrary interconnect
  • Network can reorder messages
• Infinite buffers
• Multiple virtual channels as many as you need but h/w cost, so minimize
• Directory-based memory unit (directory co-located w/ memory)
Designing a CC Protocol

• MSI Base Protocol
• Figure out different message types needed
• Nack-free → more difficult
• Allow silent drop of clean data or maintain precise sharing? What are the implications?
• How many protocol lanes needed?
• Figure out all the transient states required for processors and directory

• At least one optimization over your base protocol
Murphi

"Protocol Verification as a Hardware Design Aid," David L. Dill, Andreas J. Drexler, Alan J. Hu and C. Han Yang, 1992

• Formal verification of finite state machines

  *State space exploration*
  • *explicit enumeration* - explores all reachable states
  • tracks queue of “to-be-explored” states
  • keeps giant table of all previously visited states
  • *canonical representations* & hashing for efficiency
  • exploits *symmetry* to canonicalize redundant states
State Space Exploration

- **States**
  - stable and transient

- **Actions**
  - Prerequisite for an action to happen?
  - What is the outcome?

- **Invariants**
  - To ensure correctness
  - Example?
3-Hop MSI Protocol

How you think it should look like
3-Hop MSI Protocol

How it really looks like

Transient states
Stable states
MESI w/ Self Downgrade on 4 Procs

What you end up implementing
Solutions

3-hop MSI (NACK-free), 3 procs
47744 states, 207008 rules fired in 4.42s.

+ Self-Downgrade + Cruise Missile Invalidation, 4 procs
4690993 states, 27254378 rules fired in 1594.70s.

Numbers will be different for your implementations
Murphi  Model Checker

( Tutorial )
Tips!

Sorin et al - A Primer on Memory Consistency and Cache Coherence, Ch. 8

*Start early*

• One change at a time
  • Start simple, add incrementally
  • Compile at each step
  • Use version control

• Murphi Options: `$./twostate -h`

• Memory
  • You will soon run out of default memory allocated
  • Use: `-m<n>`, n megabytes while running executable

• Debugging tips in Murphi Manual