Agenda

- Quiz 5
  - Stuff to Study
- Snoopy Protocol Handout
- Directory Protocol Handout
- PSET #5

Friday, April 22, 2011
• Apple iPad 2 (iPhone 5?)
• 9 metal layers

http://www.adafruit.com/blog/2011/03/15/apple-a5-floorplan/
http://www.electronista.com/articles/11/03/15/larger.die.faster.ram.built.by.samsung/

http://www.adafruit.com/blog/2011/03/15/apple-a5-floorplan/

Friday, April 22, 2011
A5 side-view package-on-package design

A5 side-view package-on-package electron microscope

http://www.electronista.com/articles/11/03/15/larger.die.faster.ram.built.by.samsung/
Quiz #5

• Final Quiz!
  – Wednesday (April 27th)
  – Last day of class
Quiz #5: What will it cover?

• Memory consistency
  – sequential consistency
    • is a given memory system SC?
    • how to write correct code using a given memory system
      – using given ISA memory atomics, memory fences, etc.

• Writing code
  – how to write correct, parallel code
  – how to write good performance code

• Cache Coherency
  – Snoopy protocol
  – Directory protocol
  – given a new memory system, does it break the protocol?
  – how to implement optimizations on the existing protocols discussed in the handouts?
Quiz #5: How to study?

- Past Quizzes (‘09, ‘10)
- Handout #7 (Snoopy Protocol)
- Handout #6 (Directory Protocol)
- PSET #5
Handout #7: Snoopy Protocol

- Invalidation Protocol
  - delete other caches’ copies when a write occurs
- write-back caches
  - update memory on cache eviction
Snoopy Protocol: MBus Transactions

• CR: Coherent reads
  – issued by cache on read miss

• CRI: Coherent Read & Invalidate
  – write-allocate after a write-miss
  – (I want to write but need to read value, also invalidate everyone else’s copy)

• CI: Coherent Invalidate
  – issued by cache on a write hit to a block in the shared state
  – (I want to write my copy, invalidate everyone else’s copy)

• WR: Block Write
  – issued by cache on write-back
  – (e.g., eviction)

• CWI: Coherent Write & Invalidate
  – issued by I/O processor (DMA) on block write

Friday, April 22, 2011
Snoopy Protocol: Cache to Cache

• CCI: Cache to Cache Intervention
  – get data from another cache (the “owner”)
  – much faster to get data from cache versus memory
Snoopy Protocol: Cache States

- Invalid (I)
  - block not present
- Clean exclusive (CE)
  - cached data consistent with memory
  - exclusively held by a single cache
  - cache has write permissions (but hasn’t yet written it)
- Owned exclusive (OE)
  - dirty copy (inconsistent with memory)
  - exclusively held by a single cache
  - cache has write permissions (and has written it)
- Clean shared (CS)
  - cache has read permission
  - can be inconsistent with memory... (but is clean, relative to the owner’s copy, who holds the data in the OS state)
- Owned shared (OS)
  - dirty copy (inconsistent with memory)
  - this cache is responsible for supplying future sharers the correct value
  - OS is entered only from OE
    - this core wrote the data (OE), but somebody else requests a copy, so it downgraded itself to OS
  - other caches who hold this block are in the CS state
Handout #6: Directory Protocols

• Assumptions:
  – Reliable network, FIFO message delivery between any given source-destination pair
Directory Protocol: Cache States

• C-Invalid (I)
  – block not present

• C-Shared (Sh)
  – read permissions
  – clean copy (consistent with memory)
  – multiple copies may exist

• C-modified (Ex)
  – cache has write permissions
  – dirty copy (inconsistent with memory)
  – exclusively held by a single cache

• C-transient (Pending)
  – waiting on messages, etc.
Directory Protocol: Home Directory States

- **R(dir)**
  - The memory block is shared by the sites specified in `dir` (`dir` is a set of sites). The data in memory is valid in this state. If `dir` is empty (i.e., `dir = \epsilon`), the memory block is not cached by any site.

- **W(id)**
  - The memory block is exclusively cached at site `id`, and has been modified at that site. Memory does not have the most up-to-date data.

- **TR(dir)**
  - The memory block is in a transient state waiting for the acknowledgements to the invalidation requests that the home site has issued.

- **TW(id)**
  - The memory block is in a transient state waiting for a block exclusively cached at site `id` (i.e., in C-modified state) to make the memory block at the home site up-to-date.
Directory Protocol: Messages

- **Cache to Memory Requests**
  - ShReq
    - read permission request
  - ExReq
    - write permission request

- **Memory to Cache Requests**
  - WbReq
    - write-back request (someone else wants to read your dirty copy)
  - InvReq
    - invalidate copy (someone else wants to write your copy)
  - FlushReq
    - flush copy back to memory (someone else wants to write your dirty copy)

- **Cache to Memory Responses**
  - WbReq
  - InvRep
  - FlushRep

- **Memory to Cache Responses**
  - ShRep
    - response with data/permission to read data
  - ExRep
    - response with data/permission to write data
PSET 5: Q3 Directory Protocols...

• P5.3.A
  – Why would we get a FlushReq while in the C-pending state?

  – we’re waiting on response
  – but we only get a FlushReq if the directory thinks we own the data (Ex)
  – therefore....
    • we must have voluntarily evicted the data, then decided to read or write it again, but the directory doesn’t know that yet

  – what does the receiving cache do then?
    • drops FlushReq on the floor
    • Why does this work?
      – directory will eventually receive the voluntary WBRep, and everything gets handled correctly!
• P5.3.B
  – What goes wrong if the network is non-FIFO?

  – example:
    • $A$ sends read request (ShReq)
    • $B$ sends write request (ExReq)

  – What if....
    • invalidating a cache ($A$) who hasn’t received his data...
    • $A$ (C-pending) will drop InvReq on the floor, doesn’t respond to directory
    • directory never receives Ack from $A$ (InvRep)
PSET 5: Q3 Directory Protocols...

• P5.3.C
  – What if the processor “silently drops” a cache line?
  – directory’s sharer list doesn’t get updated
  – a future invalidation will never Ack’ed
  – deadlock!
PSET 5: Q1

- **P5.1.A**

<table>
<thead>
<tr>
<th>Processor A</th>
<th>Processor B</th>
<th>Processor C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1: ST X, 1</td>
<td>B1: R := LD X</td>
<td>C1: ST X, 6</td>
</tr>
<tr>
<td></td>
<td>B5: R := ADD R, R</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B6: ST X, R</td>
<td></td>
</tr>
</tbody>
</table>

- Can X = 4?
PSET 5: Q1

- P5.1.D
  - For this program, can a processor that reorders instructions but follows local dependencies produce an answer that cannot be produced under the SC model?
  - No
  - all stores/loads are done in order because they go to the same address
Questions?