Mockups – coherence visualizations plan

CSC494 Summer 2025

Interactive Visualizations for Computer Architecture

Goal

- Goal: build an interactive state diagram that reflects cache coherence protocols with their state transitions under MSI (or later, MOSI, MESI, MOESI) coherence protocols. These protocols have well-defined finite state machines (FSMs), where events like processor reads (BusWrite) or bus reads (BusRd) trigger transitions between states like I, S, M.

User Outcomes

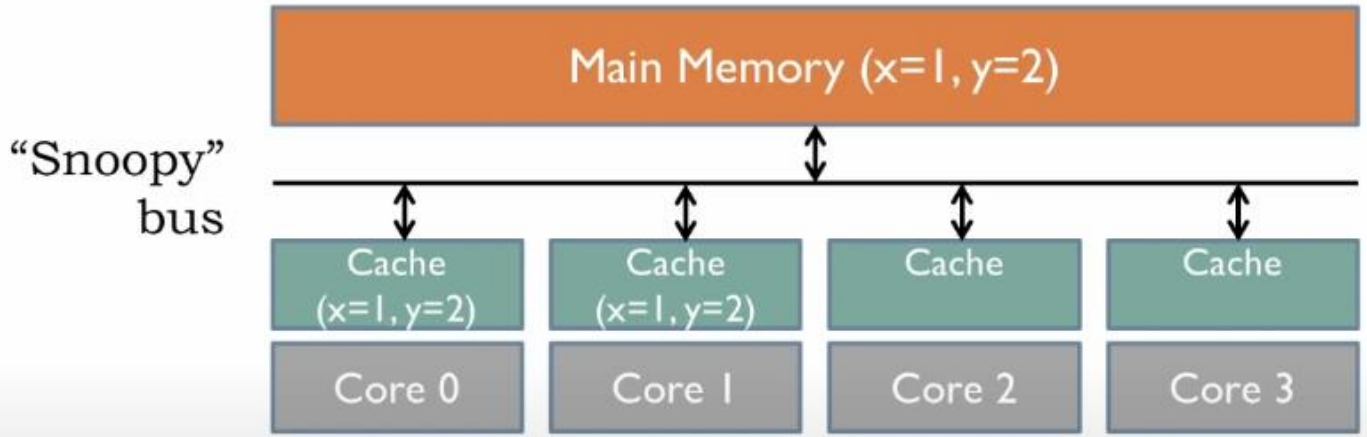
Users should gain:

- See cache contents per core
- Watch registers change due to reads/writes
- Observe data transfer via bus (e.g., from cache-to-cache or cache-to-memory)
- Understand miss types: true sharing, false sharing, etc.
- Compare behavior under MSI vs MOESI vs MOSI
- Hardware impact of different protocols

Two Ideas: idea 1

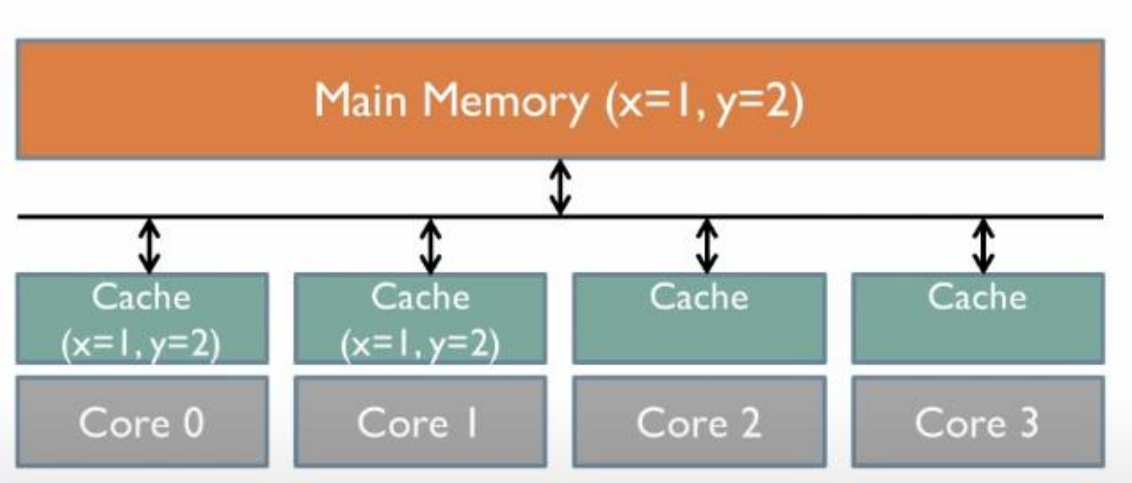
Fix: "Snoopy" Cache Coherence Protocol

Idea: Have caches communicate over shared bus, letting other caches know when a shared cached value changes



https://www.youtube.com/watch?v=ISaYWm8T8n4&t=570s

Two Ideas: idea 1



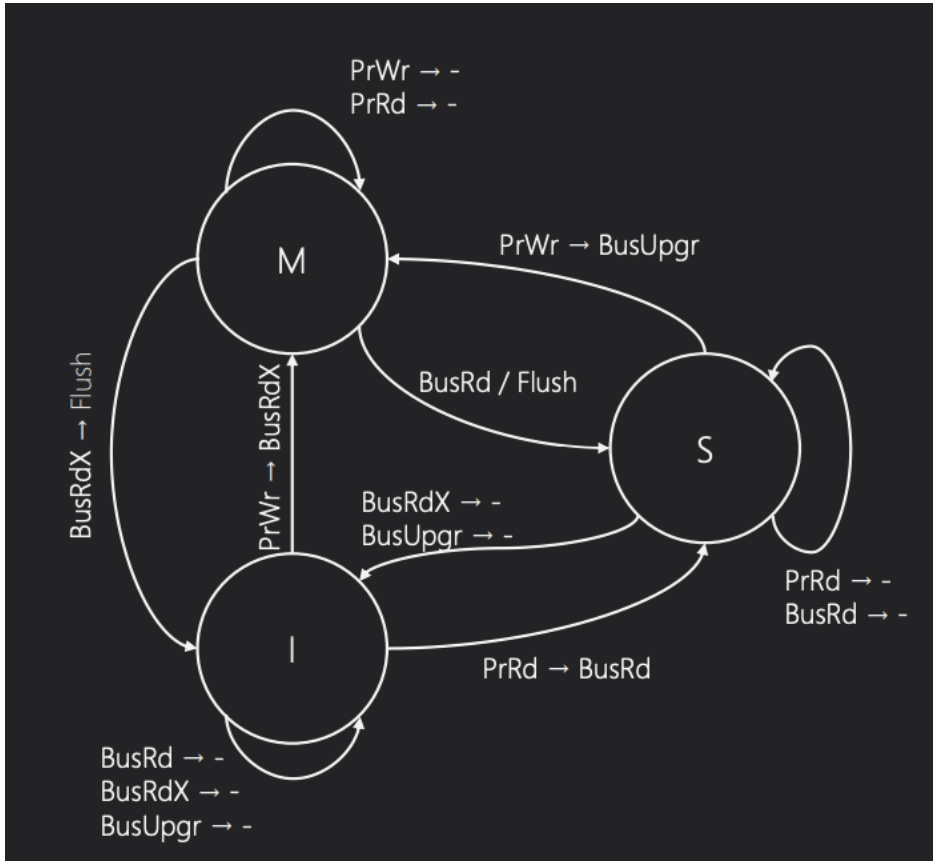
What happens when we run the above workload on the bus?

https://www.youtube.com/watch?v=ISaYWm8T8n4&t=570s

Reasoning for Idea 1

Here we can show what happens on the bus as we do various actions across different caches. We can easily see the changes reflected in terms of the length / number of transactions that occur on the bus.

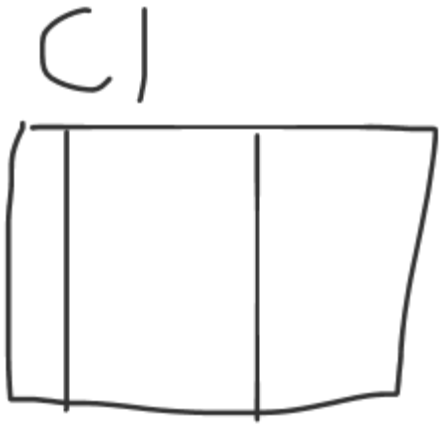
Two Ideas: idea 2

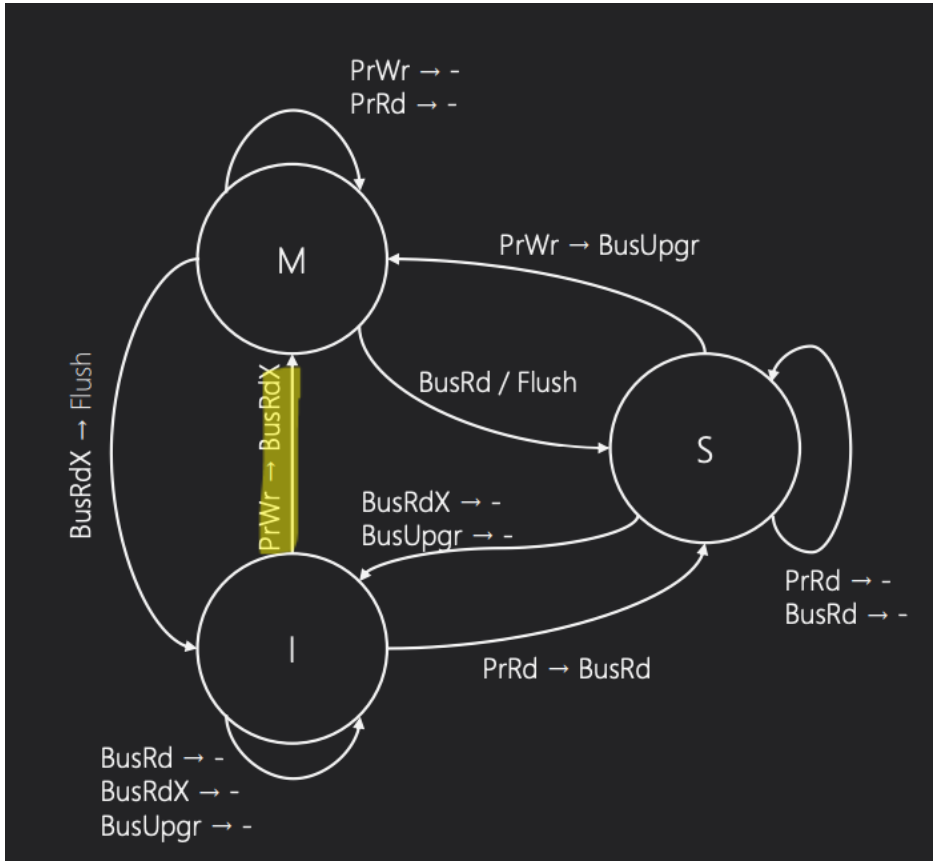


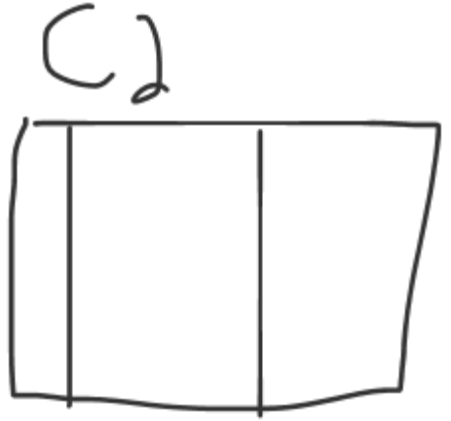
DDL to change between MSI, MESI, MISO, MOESI

https://q.utoronto.ca/courses/374379/pages/attend-lecture-8?module_item_id=6072263

Two Ideas: idea 2







What happens in the cache line when we have a _____ bus transaction?

What happens on the bus when we have a _____ bus transaction?

Reasoning for Idea 2

View transient states better.

Understand how the bus works and the amount of hardware that has to go into different systems to make them work

Enable users to switch between different configurations (MSI, MOSI, MESI, MOESI) and see the effects on the overall

The main goal is to show that there are differences in performance in terms of the number of reads/writes required between the different systems.

Show that for the user, coherence doesn’t matter. Sequential consistency ensure that program order is preserved.

Plan

- Implement both ideas separately and see if some ideas can be combined
- Visualize the Hardware differences

References

https://www.youtube.com/watch?v=ISaYWm8T8n4&t=570s

This video for MIT OpenCourseWare is good as it is a good example of where/why we should implement the visualizations. There's a section where we are trying to explain coherence and which it would be great if we had included an interactive viz there.

Consider https://github.com/pmndrs/zustand for FSM management. I will look into this compared to Redux.