Cache Coherency and Memory Consistency

Why On-Chip Cache Coherence is here to stay
- Motivation:

- There is skepticism about the scalability of cache coherence: Some argue:
  - Availability of other paradigms such as message passing and incoherent scratchpad memories
  - Some programs do not scale with coherency.

Contribution

- Addresses various concerns with in-depth analysis of each.
- Provides substantial reasons to support the continued use of coherency models.
- “... we find no compelling reason to abandon coherence”
  - “performance generally superior to what is achievable with software-implemented coherence”
  - backward compatible
-

Contribution

- Addresses various concerns with in-depth analysis of each.
- Provides substantial reasons to support the continued use of coherency models.
- “... we find no compelling reason to abandon coherence”
  - “performance generally superior to what is achievable with software-implemented coherence”
  - backward compatible
- Excellent arguments in favor of coherency - consistently refuting possible reasons why on-chip coherency cannot scale
  - traffic
  - storage cost
  - maintaining inclusion
  - latency
  - energy

Merits

- Uses practical examples to support these arguments
- If multiple scenarios exist, the paper accounts for them.
- Convincing and thorough on the cases covered

Failings

- Lacks hardware implementations to support arguments
- Does not account for scalability of supporting hardware, though the argument is that scalability concerns will come into place from other issues first
- Does not account for multi-chip coherence
- Could have spent more time discussing the alternatives to “on-chip” coherence.

Questions

- Does the paper hold true today? 8 years later, do you still agree with the authors?
- Is there anything the authors have done in order to eliminate few of the failings?

Token Coherence: Decoupling Performance and Correctness - Motivation:

- Snooping requires total ordering and is not scalable due to bus bandwidth limitations.
- Directory based coherence adds indirection, increases latency due to added communication.
- Coherence is not scalable

Contribution

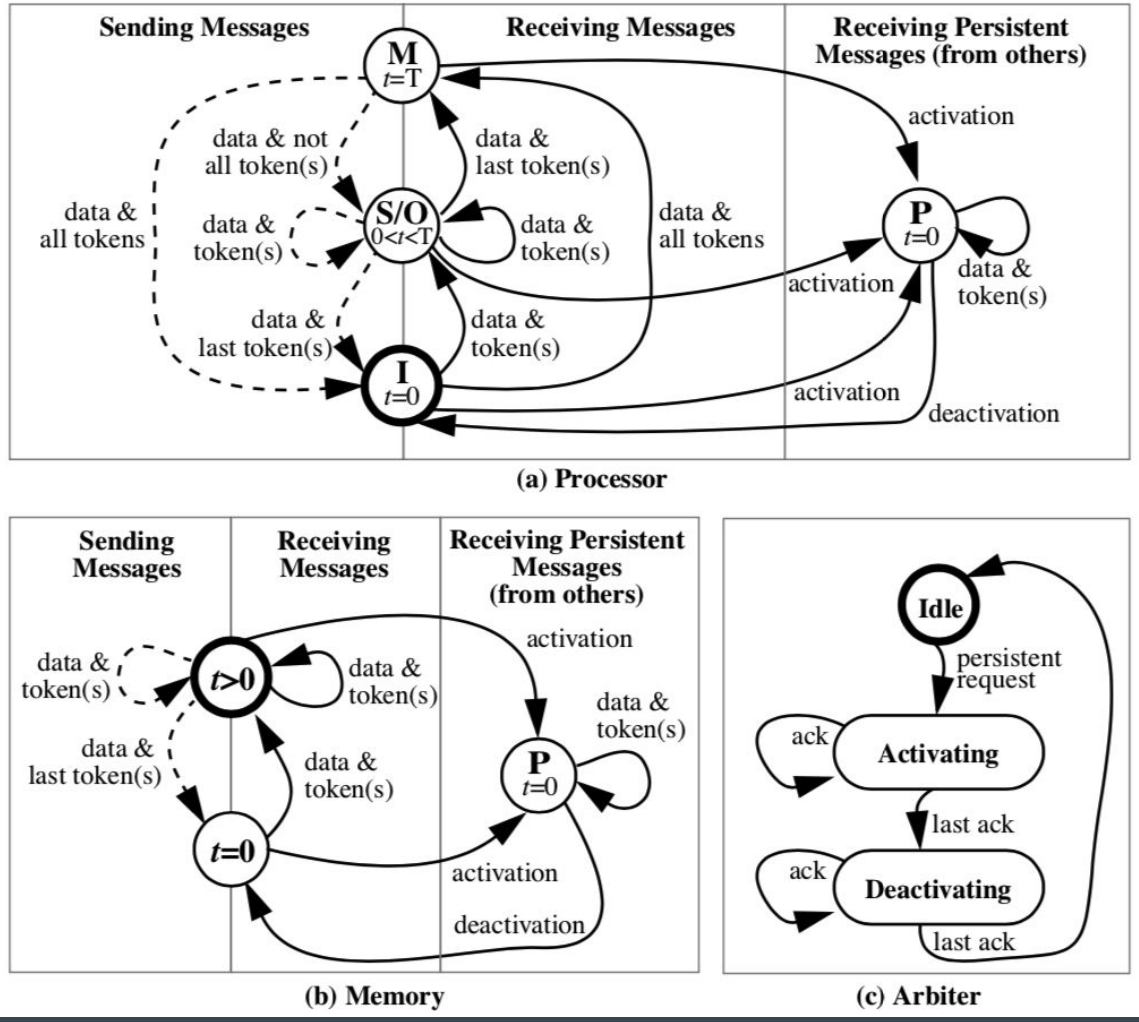
- TokenB - a new token coherence protocol
- Idea of separating protocol into two, one designed for performance and one designed to ensure correctness
  - performance for the common case
  - guaranteed correctness for the worst case

Merits

- Describes novel, correct, and performant principles for improving cache coherence protocols
- Allows for use of an unordered interconnect to serve cache-to-cache misses

Failings

- “correctness substrate” has not been implemented in hardware
- Efficiency arguments not fully convincing
- Broadcast required for implementation
- Cost of torus interconnect not justified


Sending Messages
Receiving Messages
Receiving Persistent Messages (from others)
M
t=T
S/O
0<t<T
I
t=0
P
t=0
data & not all token(s)
data & last token(s)
data & all tokens
data & token(s)
data & token(s)
data & all tokens
data & last token(s)
data & token(s)
activation
activation
data & token(s)
activation
deactivation
(a) Processor
Sending Messages
Receiving Messages
Receiving Persistent Messages (from others)
t>0
t=0
P
t=0
data & token(s)
data & token(s)
data & last token(s)
data & token(s)
activation
data & token(s)
activation
deactivation
(b) Memory
Idle
persistent request
ack
Activating
last ack
ack
Deactivating
last ack
(c) Arbiter

Questions

- Are the additional hardware costs worth the benefits? If so, why isn't this protocol widely implemented?
- Does the use of a modified broadcast network imply that this new protocol is about as unscalable as the ones that it was trying to replace?