Kobold: Simplified Cache Coherence for Cache-Attached Accelerators

Jennifer Brana, Brian C. Schwedock, Yatin A. Manerkar, Nathan Beckmann

WDDSA 2022

Session 1: What are the potential accelerators?

University of Portland Shiley School of Engineering

College of Engineering Computer Science & Engineering University of Michigan

Electrical & Computer Engineering

Carnegie Mellon University Computer Science Department

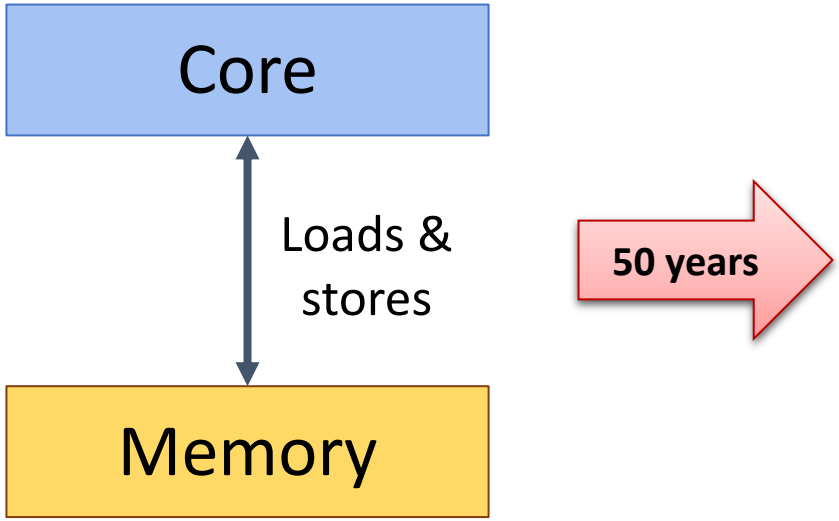
Executive Summary

- Discrete accelerators suffer from high communication costs with the cores
- Cache-attached accelerators perform ops where data exists and benefit from simple communication
- **Problem:** significant complexity required for coherent access to memory
- Kobold coherence protocol **simplifies system integration** for cache-attached accelerators **without degrading performance**
- Kobold adds an **area overhead of only 0.09%**

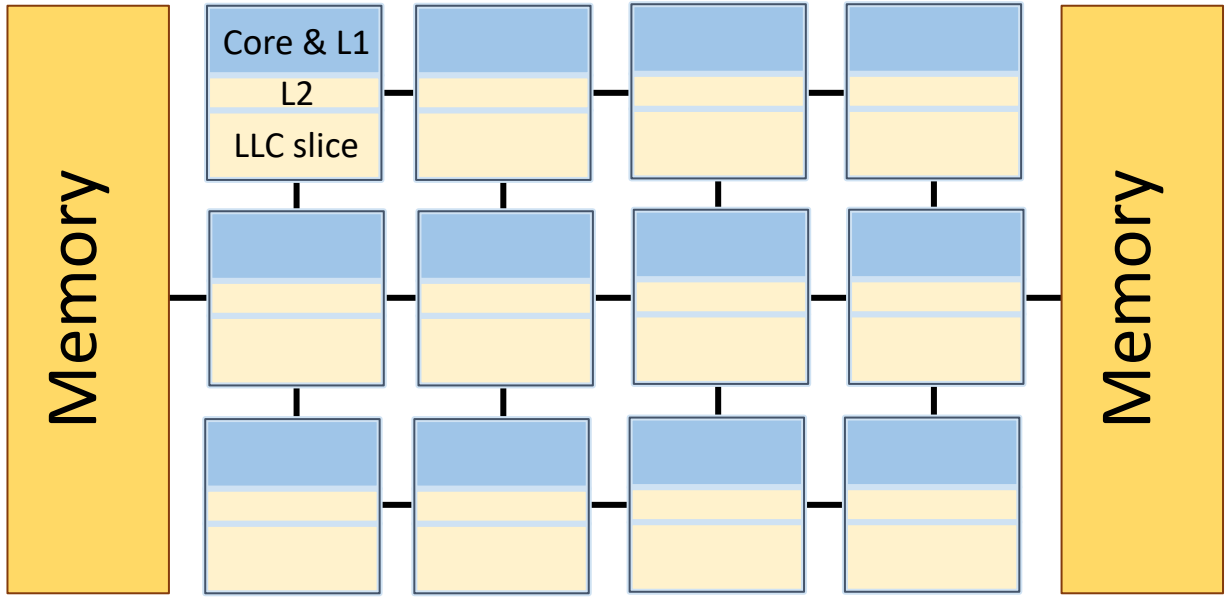
Outline

- Summary
- Motivation
- Design
- Performance Considerations
- Evaluation

Data movement costs keep getting worse



50 years



Near-data Computing to Reduce Data Movement

Prime *[P. Chi, ISCA 2016]*

Pinatubo *[S. Li, DAC 2016]*

Isaac *[A. Shafiee, ISCA 2016]*

Grapher *[L. Song, HPCA 2018]*

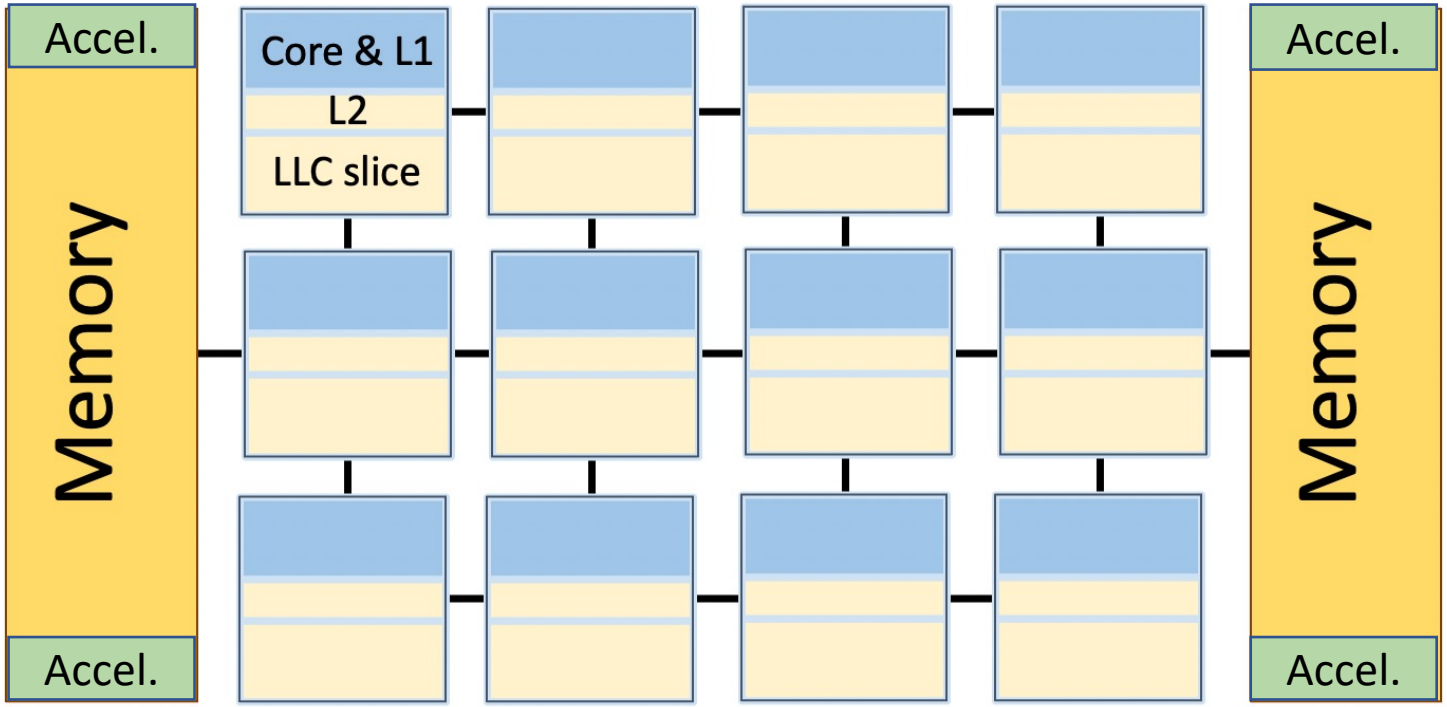
Terasys *[Gokhale, Computer 1995]*

FlexRAM *[Kang, ICCD 1999]*

EXECUBE *[Kogge, ICPP 1994]*

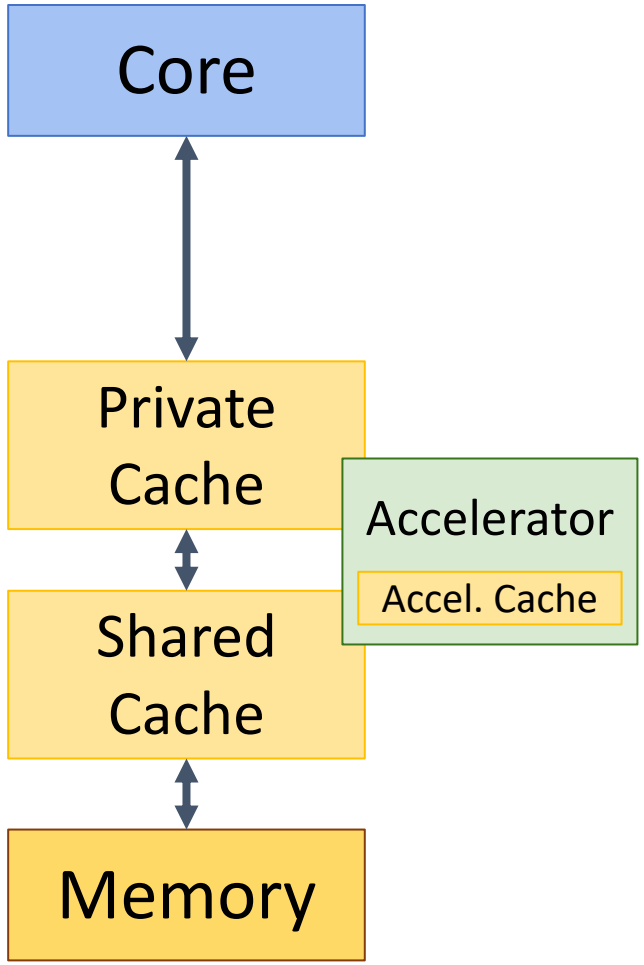
PIM Enabled Instructions *[Ahn, ISCA 2015]*

...



Cache-attached Accelerators

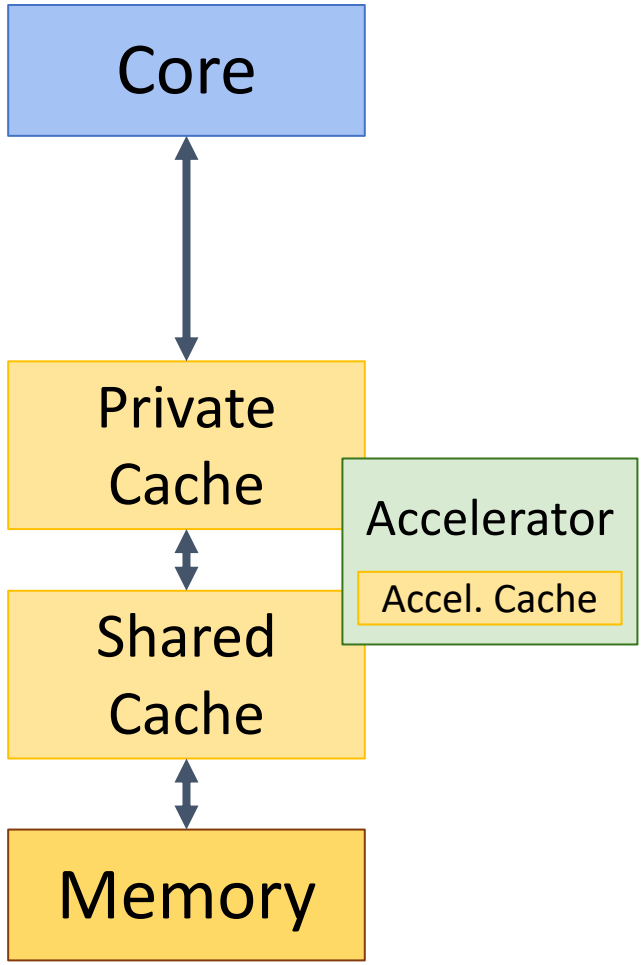
- täkō *[Schwedock, ISCA 2022]*
- Compute Caches *[S. Aga, HPCA 2017]*
- Livia *[Lockerman, ASPLOS 2020]*
- LLC-Compute *[Pattnaik, ISCA 2019]*
- Cache Automaton *[Subramaniyan, MICRO 2017]*
- Stream Floating *[Z. Wang, HPCA 2021]*



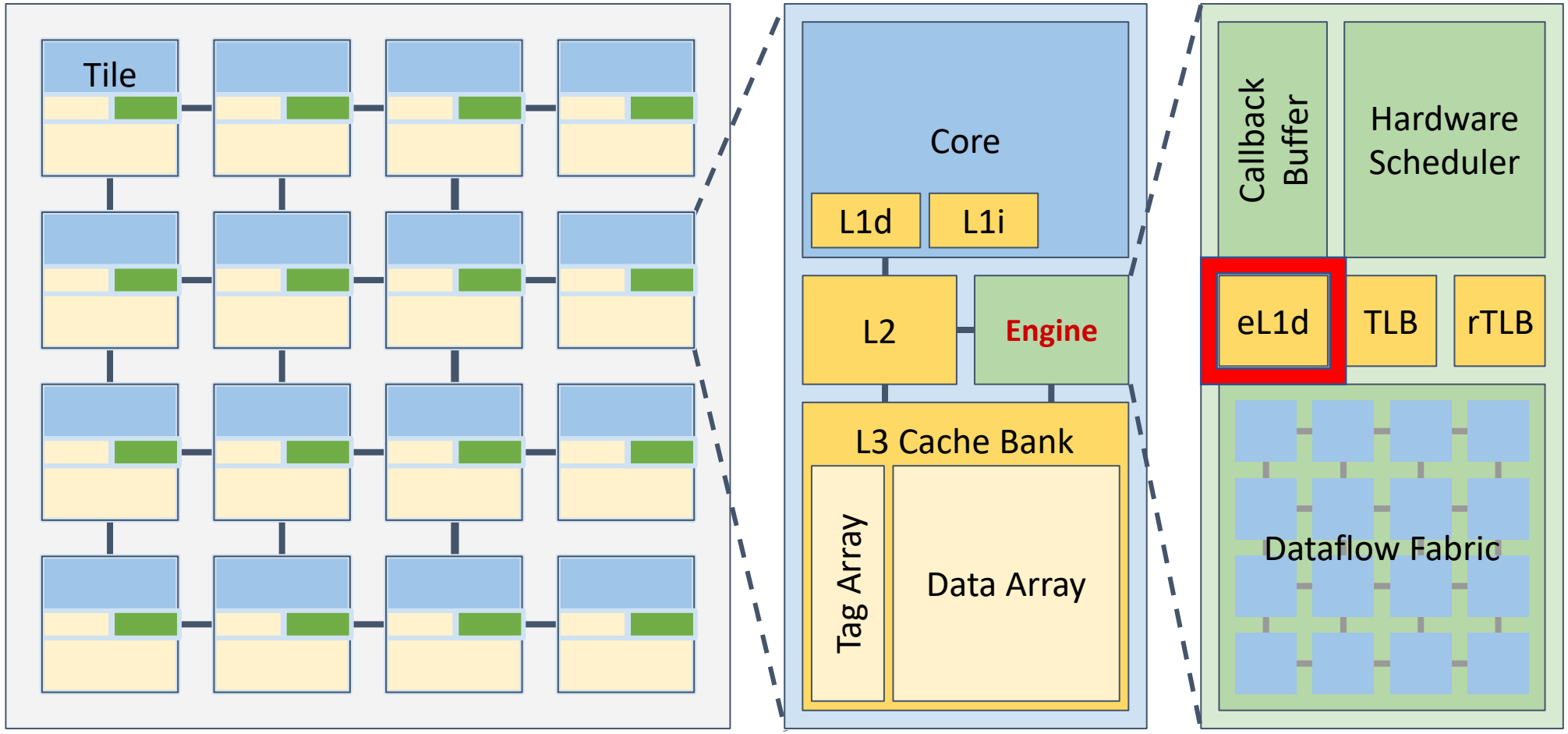
Cache-attached Accelerators

- Offload key computations.
- Simple fine-grained communication with processor.
- Eases the effort to use accelerators.
- … but to benefit, must maintain coherence with the rest of the system.

Kobold is a new coherence protocol to simplify the integration of cache-attached accelerators.



Example System: täkō

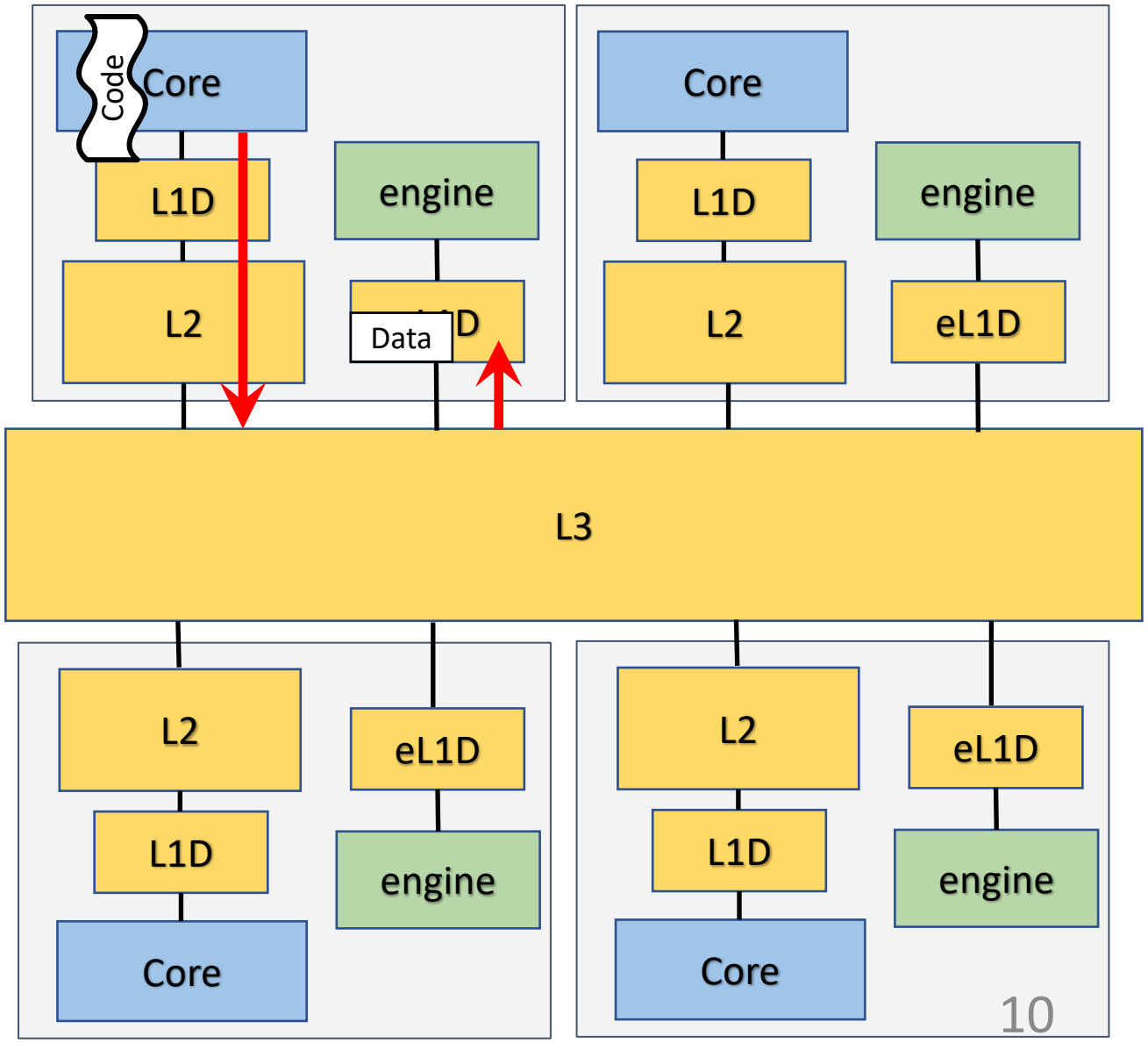


[Schwedock, ISCA'22]

Alternative Designs

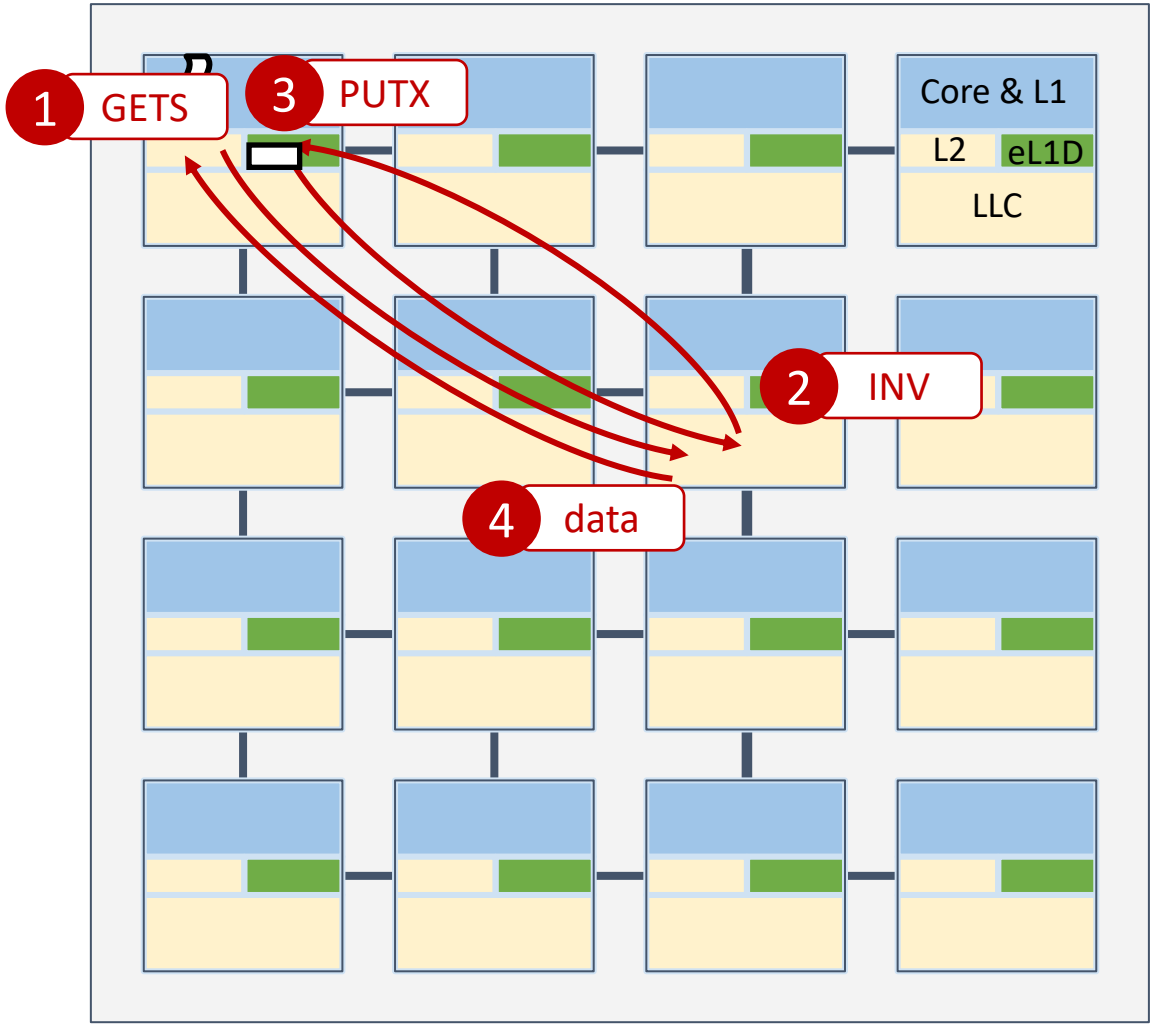
Naïve Coherence for Cache-attached Accelerators

- Naïve system treats eL1D as additional LLC sharer
- eL1D uses same protocol as L2
- *Results in excessive communication through LLC*

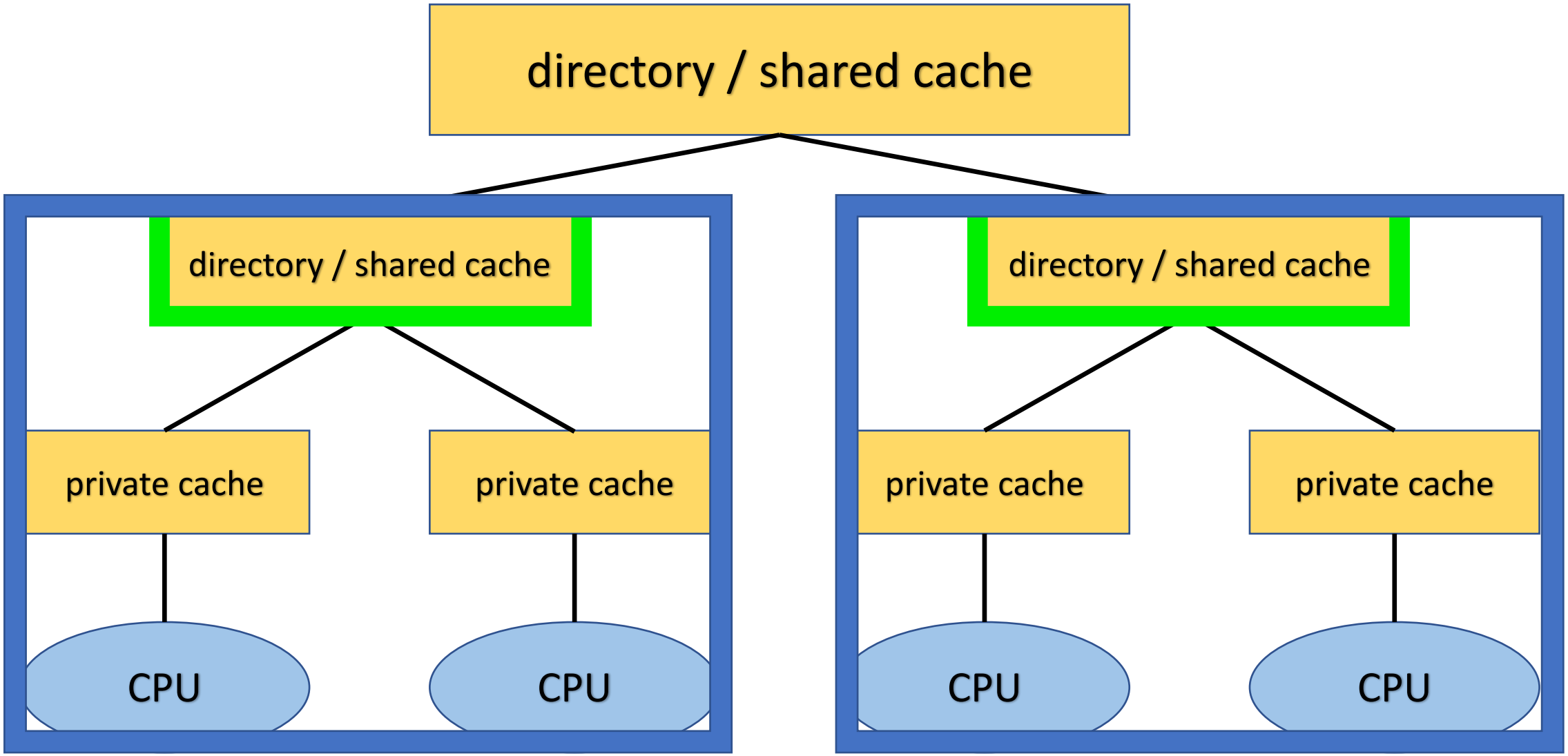


Naïve Coherence for Cache-attached Accelerators

- Naïve system treats eL1D as additional LLC sharer
- eL1D uses same protocol as L2
- *Results in excessive communication through LLC*

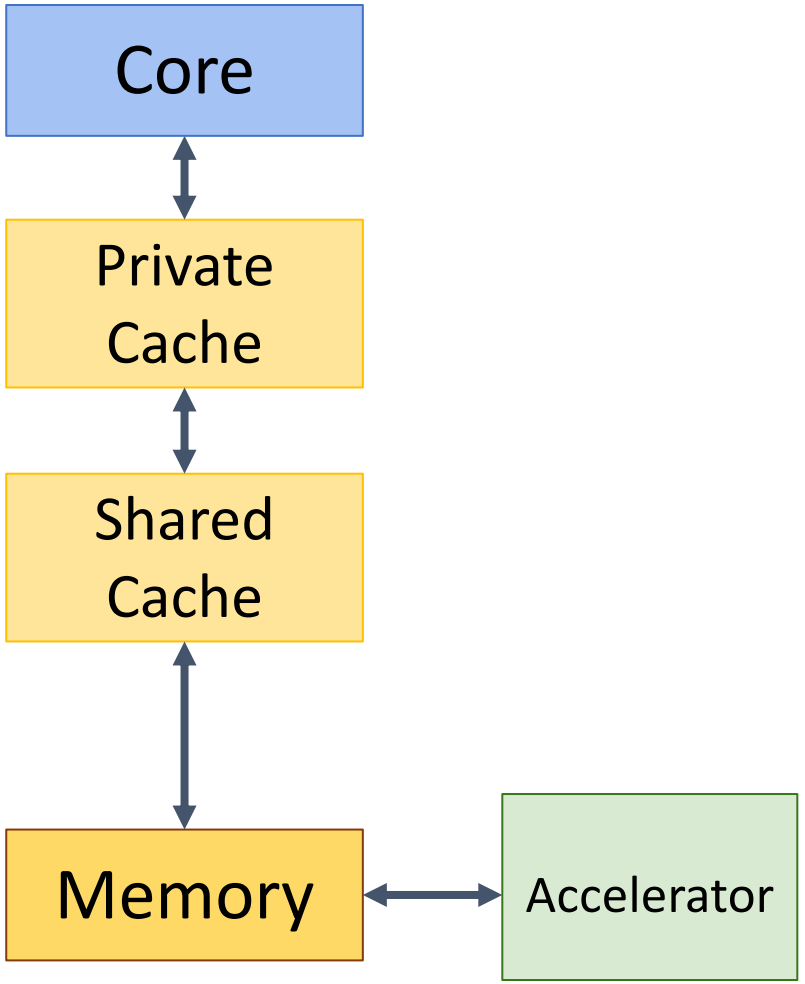


Hierarchical Cache Coherence



Prior Heterogenous Protocols

- Spandex *[J. Alsop, ISCA 2018]*
- CoNDA *[A. Boroumand, ISCA 2019]*
- Direct Memory Access (DMA) *[S. Ma, ISCAS 2019]*
- Mixed-proxy extensions *[Lustig, ISCA 2022]*



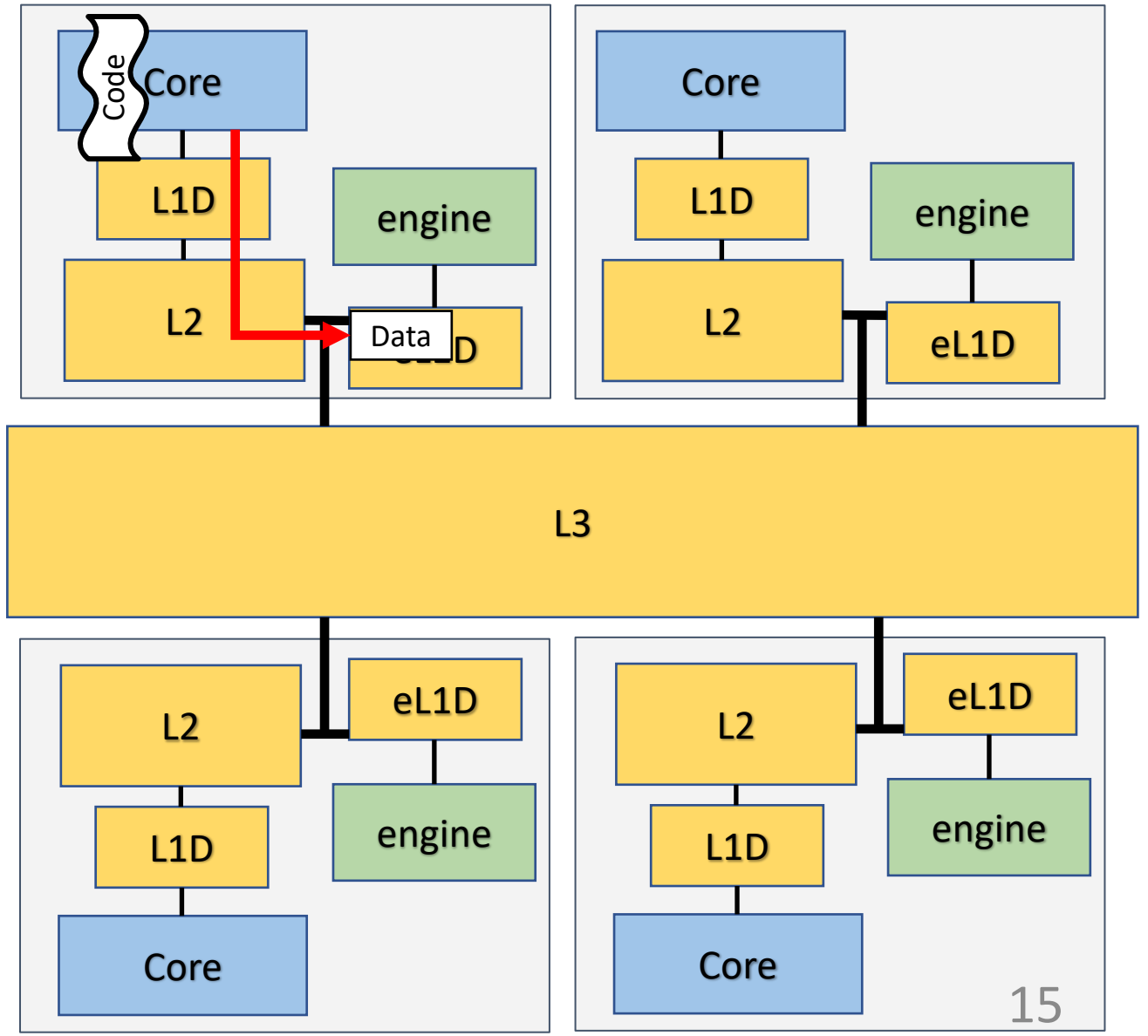
Designed for:

- ~~coarse-grain sharing~~
- ~~systems integrating very different protocols~~
- ~~caches with very different cache semantics~~

Kobold Design

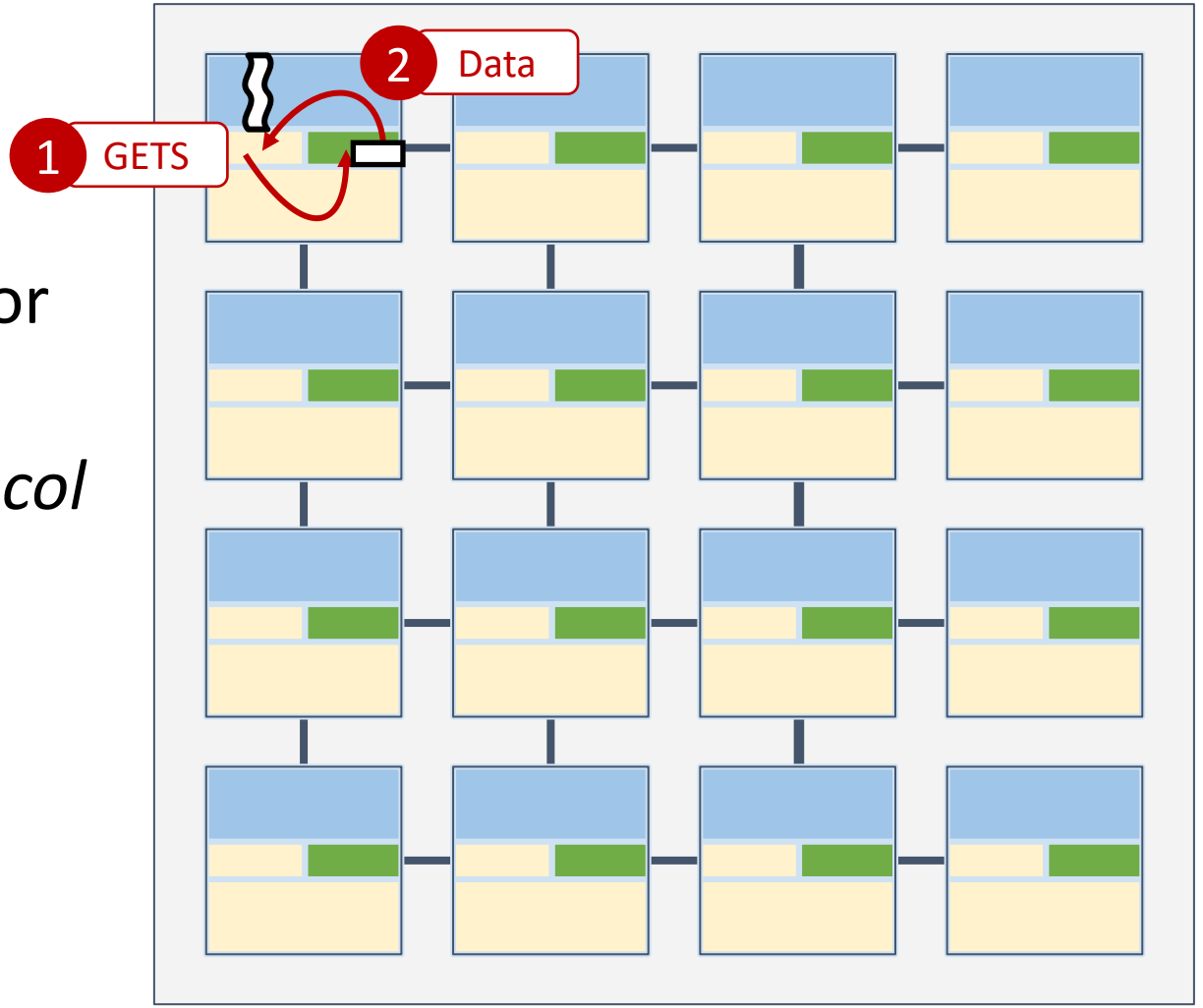
Cache Hierarchy Organization

- Goal: reduce unnecessary communication with the LLC
- Kobold attaches the accelerator to the L2 cache
- *Requires new coherence protocol*



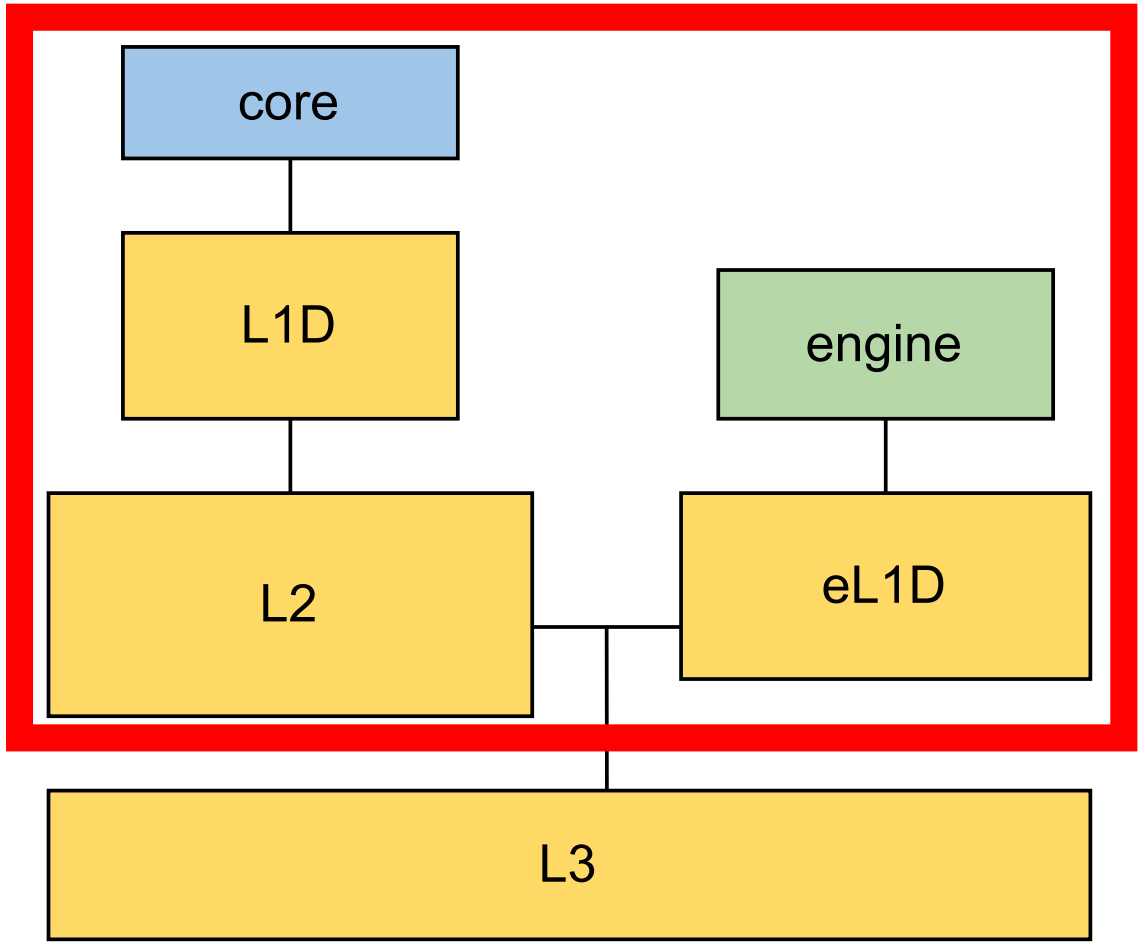
Cache Hierarchy Organization

- Goal: reduce unnecessary communication with the LLC
- Kobold attaches the accelerator to the L2 cache
- *Requires new coherence protocol*



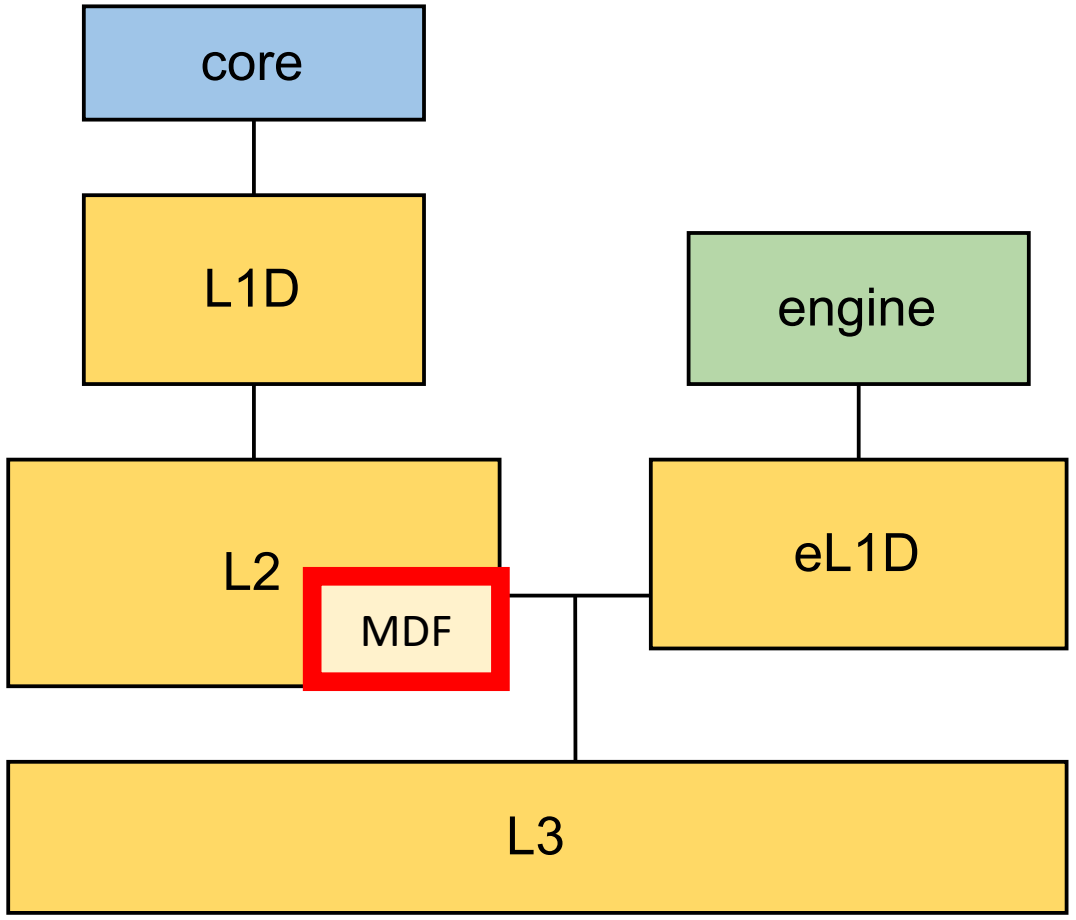
Verification Complexity

- Goal: restrict complexity of accelerator to **within a tile**
- LLC protocol remains unchanged

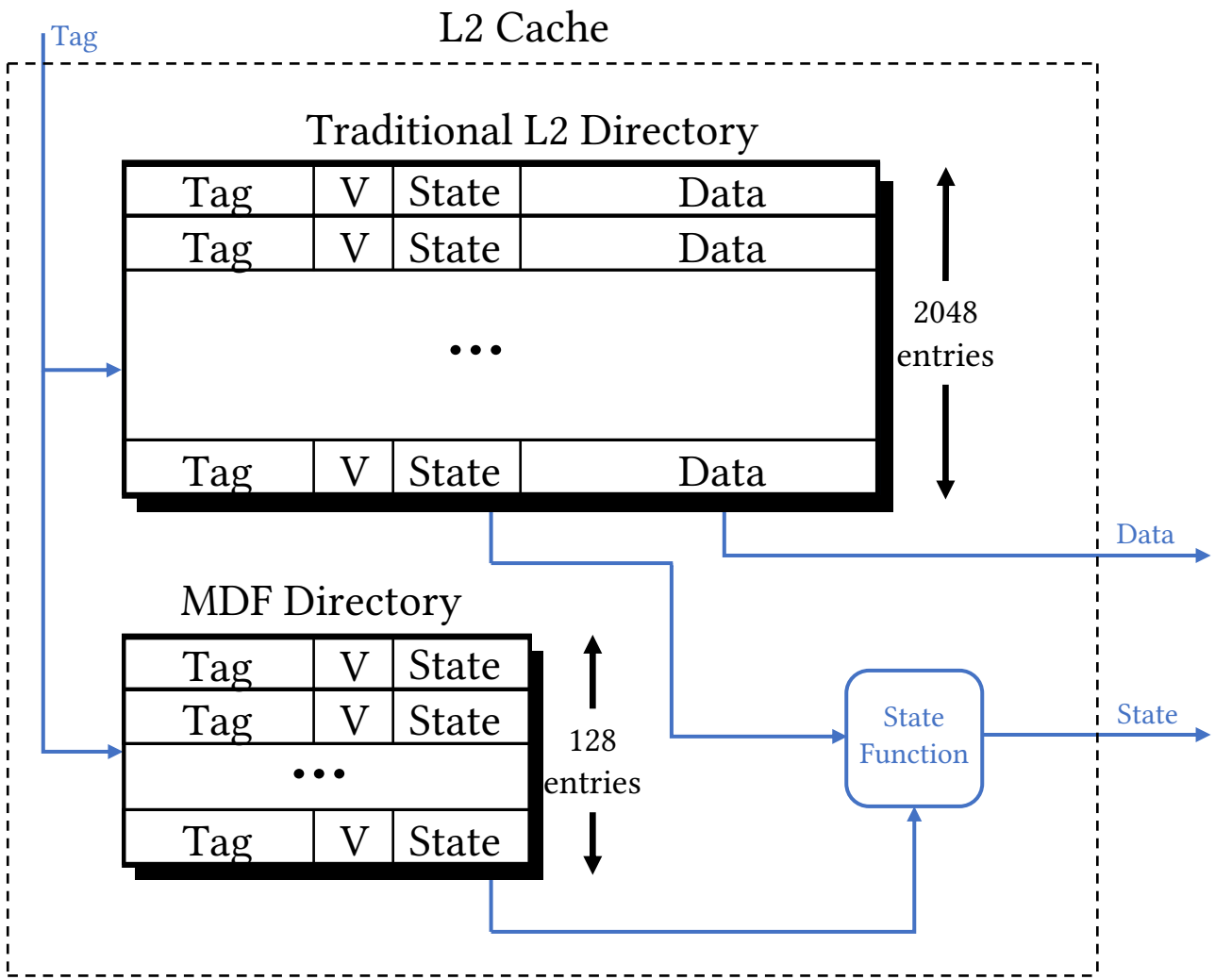


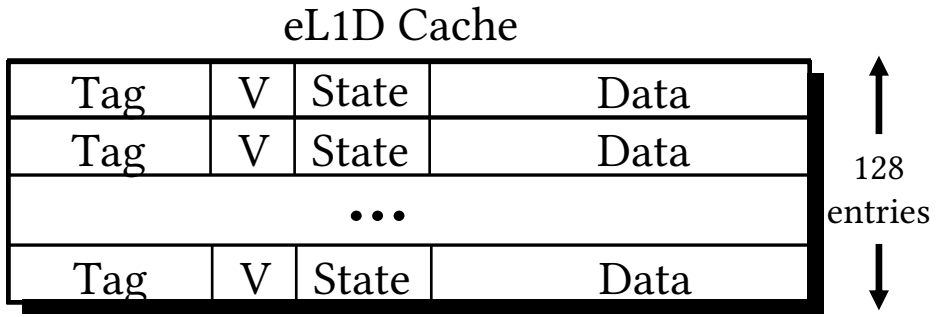
Kobold Cache Hierarchy

- Kobold is a type of hierarchical coherence.
- L2 is non-inclusive of the eL1D
- Logically eL1D & L2 form a hierarchy but operate as peers via snooping
- Mis-direction Filter (MDF) tracks the contents of the eL1D cache



Cache Microarchitecture

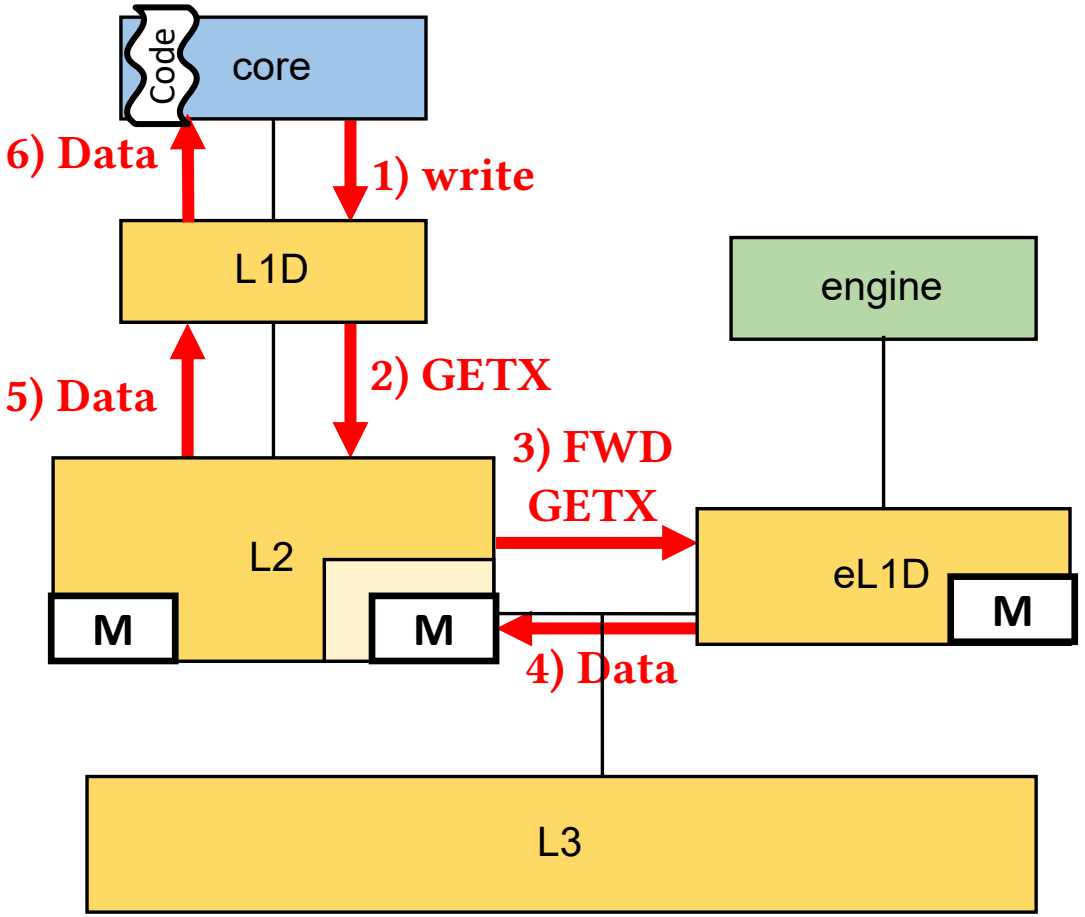




Kobold's intra-tile locality enables:

1. **Tile caches to transfer ownership w/o sending requests to the LLC.**

Example – core write

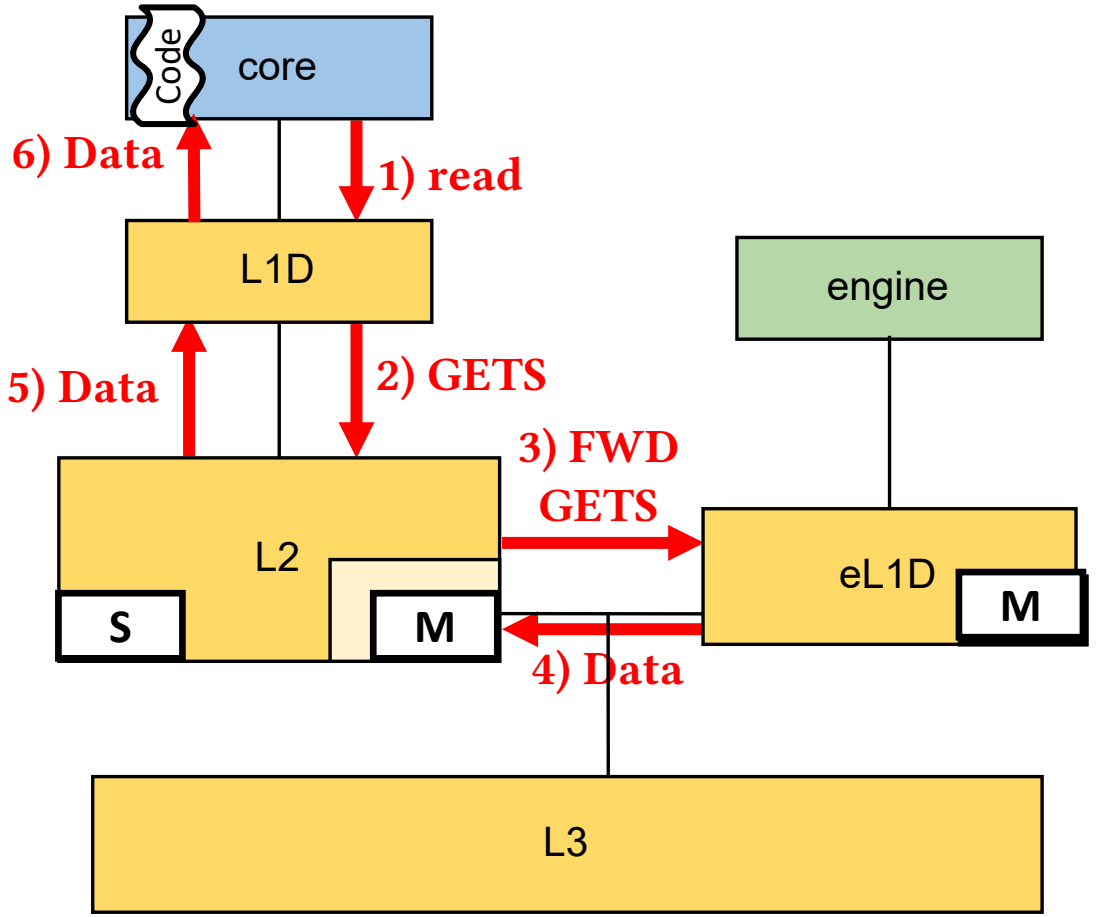


Step	L1D	eL1d	MDF	L2	L3
Init)	I	M	M	I	M
1)	I	M	M	I	M
2)	I	M	M	I	M
3)	I	I	M	I	M
4)	I	I	I	M	M
5)	M	I	I	M	M
6)	M	I	I	M	M

Kobold's Intra-tile locality enables:

1. Tile caches to transfer ownership w/o sending requests to the LLC.
2. **All tile caches to share data that is tracked as exclusive in the LLC directory.**

Example – core read



Step	L1D	eL1d	MDF	L2	L3
Init)	I	M	M	I	M
1)	I	M	M	I	M
2)	I	M	M	I	M
3)	I	S	M	I	M
4)	I	S	M	S	M
5)	S	S	M	S	M
6)	S	S	M	S	M

Kobold's Intra-tile locality enables:

1. Tile caches to transfer ownership w/o sending requests to the LLC.
2. All tile caches to share data that is tracked as exclusive in the LLC directory.
3. **Tile caches to coordinate responses to LLC requests.**

*see paper for more details

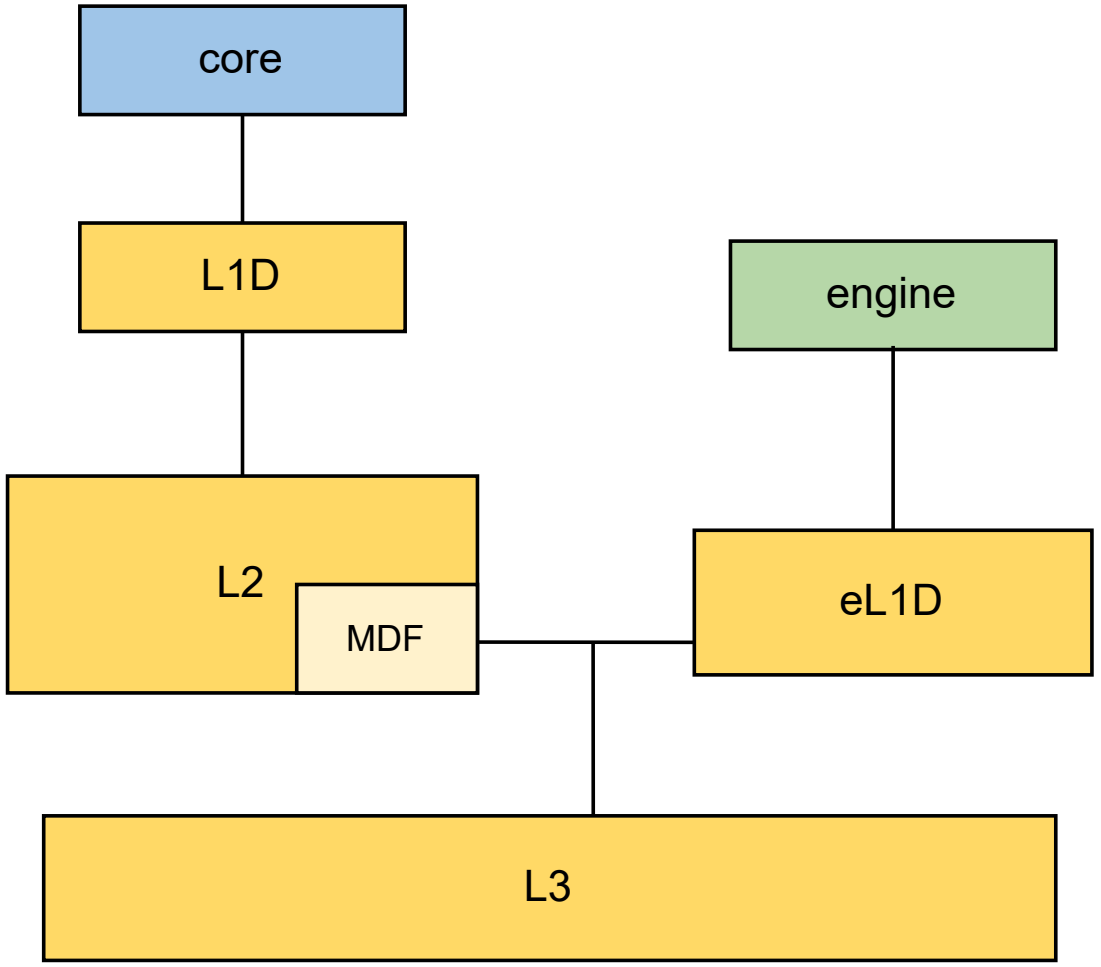
Kobold's Intra-tile locality enables:

1. Tile caches to transfer ownership w/o sending requests to the LLC → **Delay writebacks to the LLC**
2. All tile caches to share data that is tracked as exclusive in the LLC directory → **Reduce coherence requests required**
3. Tile caches to coordinate responses to LLC requests → **Ensures a single responder to requests**

Performance Considerations

- **Problem**: Cache-attached accelerators can pollute the L2 cache.
  - **Solution**: L2 cache is non-inclusive of eL1D
- **Problem**: Kobold adds L2 latency to eL1D misses.
  - **Solution**: w/ minor modification to LLC protocol, optionally allow eL1D speculative loads

*see paper for more details



Preliminary Results + Verification

We evaluate a system with a 128KB L2, 8KB eL1D, and 512KB LLC per tile.

Component	Area (mm^2)
L2 cache	0.2706 mm^2
LLC bank	0.5963 mm^2
MDF	0.00076 mm^2

Cacti results

```
State Space Explored:

        12534 states, 18738 rules fired in 0.28s.
```

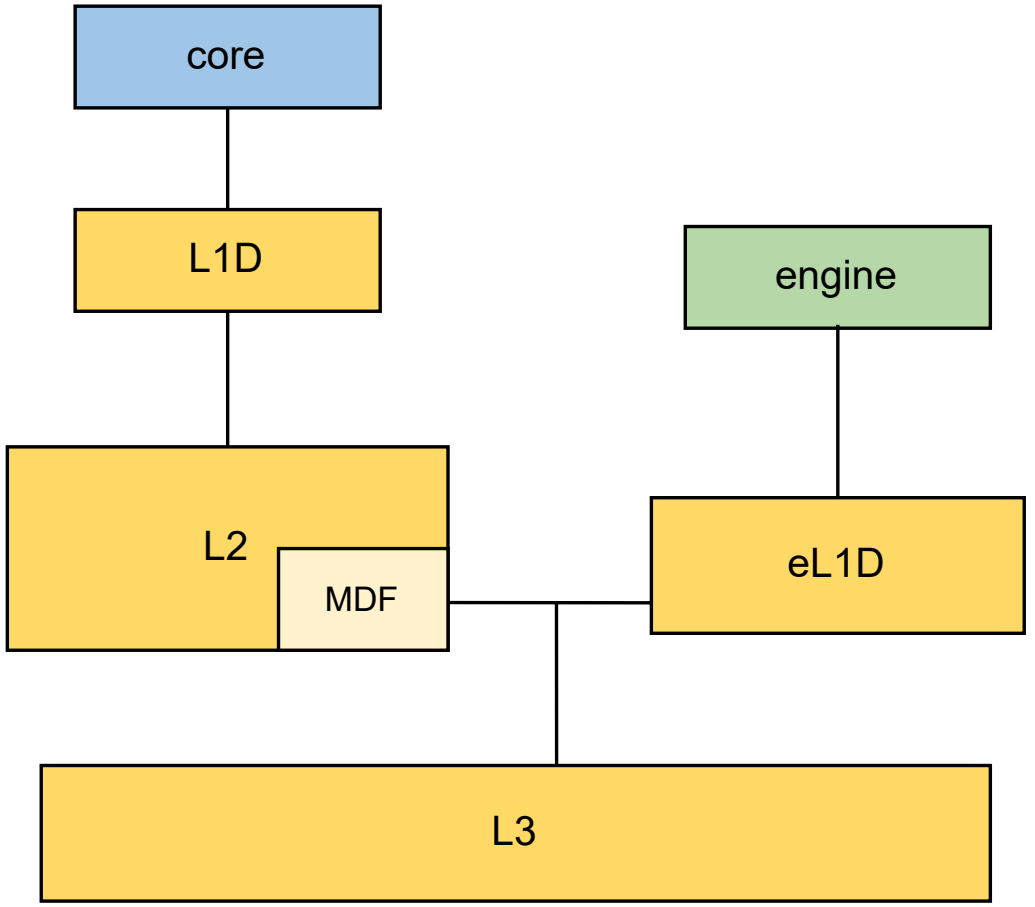
Murphi output

Estimated area overhead to be 0.09% of baseline area

Verified stable state protocols using Murphi model checker

Future Work

- Generate concurrent protocols using HieraGen
- Implement and test in the täkō system



Kobold: coherence for cache-attached accelerators

- Ease of integration is vital for cache-attached accelerators
- Kobold restricts the complexity of accelerator integration to within a tile
- Kobold minimizes on-chip network traffic and preserves the baseline processor performance