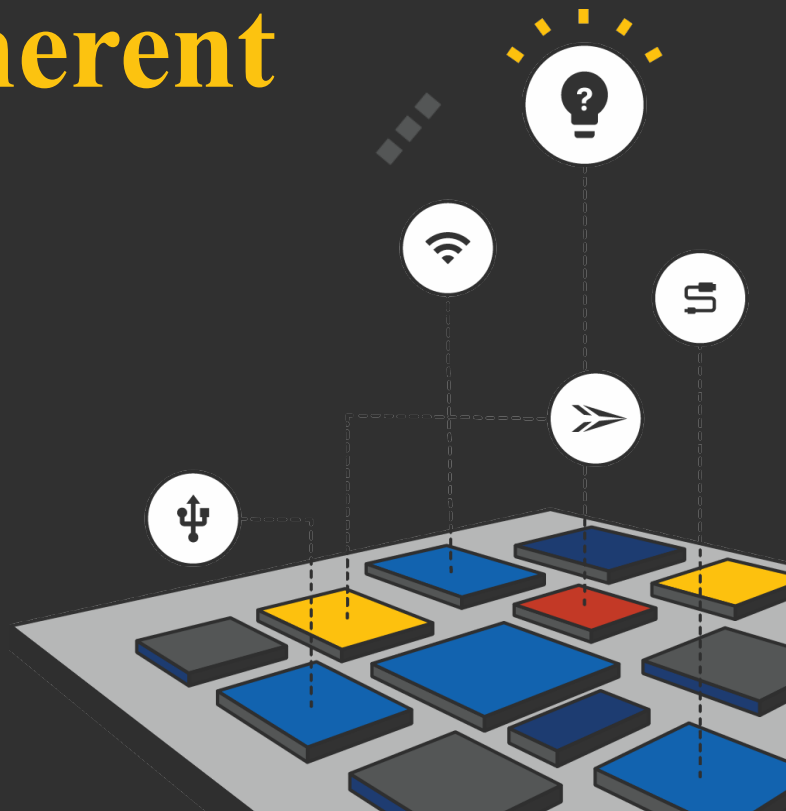
SiFive

TileLink: A free and open-source, high-performance scalable cache-coherent fabric designed for RISC-V

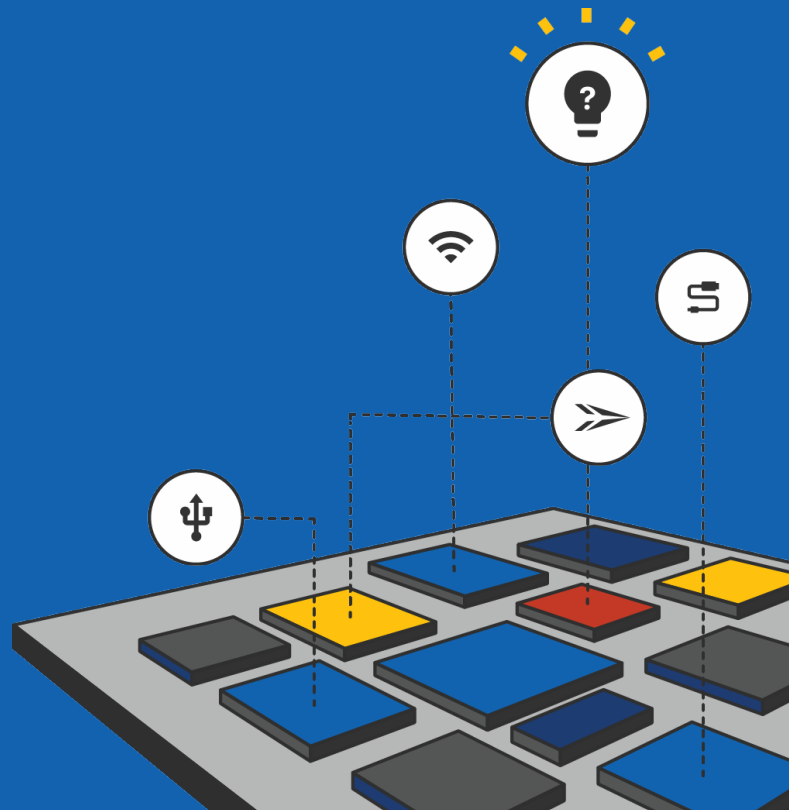
Wesley W. Terpstra

SiFive



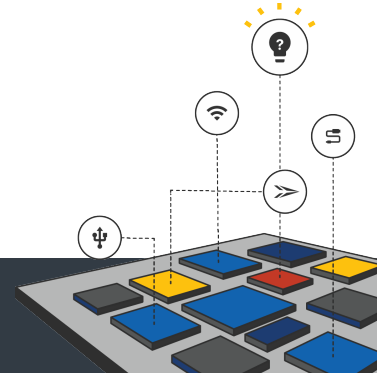
SiFive

TileLink Rationale



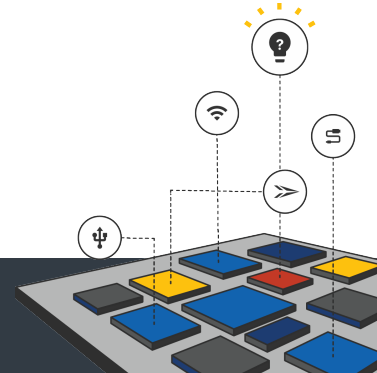
So you want to build a chip, ‘eh?

sounds like you need a coherent bus protocol!



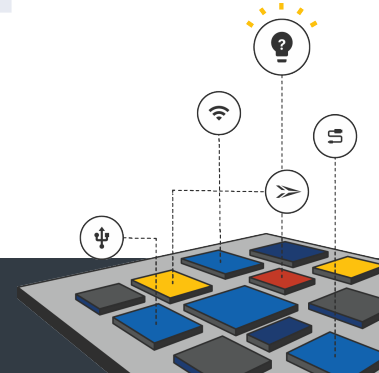
Requirements for a RISC-V bus

- Open standard
- Easy to implement
- Cache-coherent block motion
- Multiple cache layers
- Reusable on- and off-chip
- High performance



Requirements for a RISC-V bus

	AHB	Wishbone	AXI4	ACE	CHI	TileLink
Open standard	✔	✔	✔	✔	✗	✔
Easy to implement	✗	✔		✗	?	✔
Cache block motion	✗	✗	✗	✔	✔	✔
Multiple cache layers	-	-	-	✗	?	✔
Reusable off-chip	✗				?	✔
High performance	✗	✔	✔		?	✔

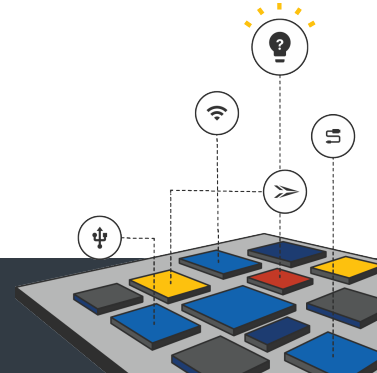


What about AMBA CHI/ACE?

- Open standard? CHI is not open!

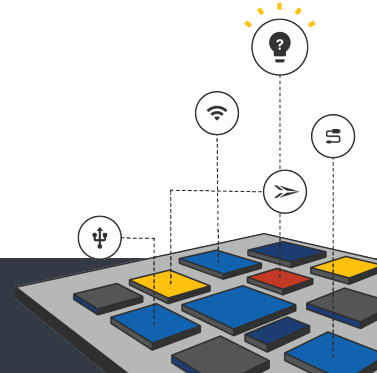
  "The AMBA 5 CHI specification is currently available to partners integrating SoCs or developing IP or tools that implement it. Please contact your Arm account manager for details on obtaining a copy."

- Easy to implement?
  - ACE: 10 probe message types, split control/data, narrow bursts, …
- Multiple cache layers?
  - ACE: No – designed for snoop-style coherency
- Depend on a standard controlled by a RISC-V competitor?



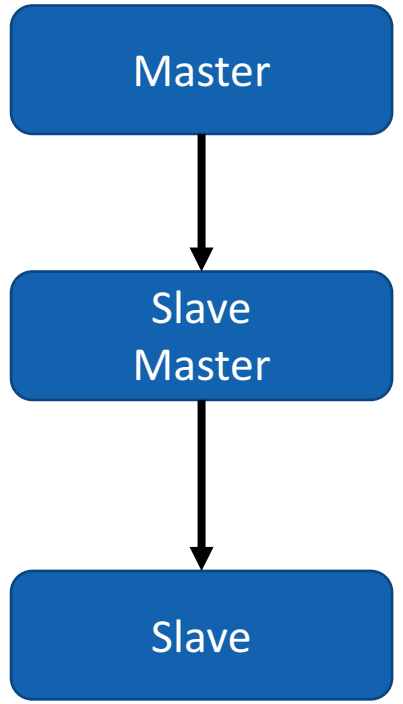
TileLink: Rolling our own has advantages

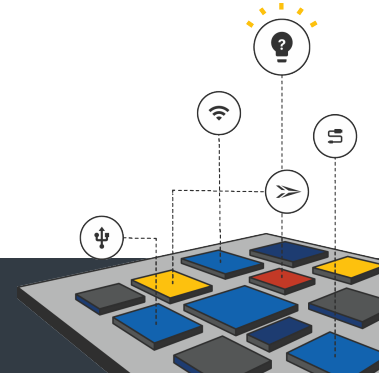
- Clean slate: avoid prior pitfalls
- Decouple message protocol from wire protocol
- Competitor cannot undermine RISC-V's future
- Designing for RISC-V requirements = simpler design
  - Reduced Message Protocol; RISC => RMP
  - Assume all connected hardware is trusted
  - Only power-of-2 block transfers



TileLink: Bird's-Eye View

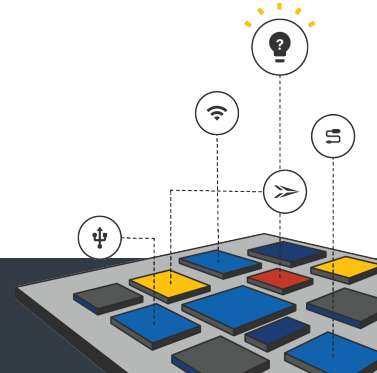
- Master-slave point-to-point protocol
- Message-based with 5 (ABCDE) priorities
- Out-of-order design with optional ordering
- Progressive conformance level complexity
- Cache-coherent memory block motion
- Designed for composability and deadlock freedom





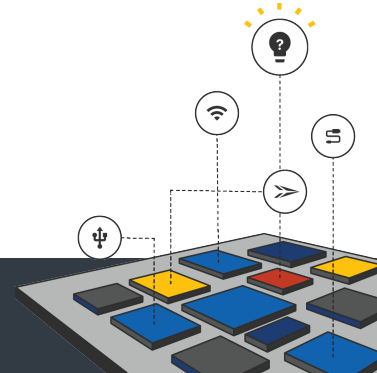
TileLink: Open Source and In Production

- Rocket-chip open source SoC is entirely built using TileLink
  - > 30 public modules; including cores, Xbars, and adapters
  - broadcast-hub coherency manager (ie: ACE-style snoop coherency)
  - bridges to AXI/AHB/APB, clock crossings, fuzzer, monitor, model checker
  - for more details, see our TileLink CARRV 2017 paper
- SiFive-blocks: open I2C, SPI, UART, GPIO, PWM TileLink slaves
- Foundation of publicly available SiFive chips (FE310 + FU500)
  - a banked directory-based wormhole MESI L2$
  - off-chip coherent TileLink interconnect to FPGA (ChipLink)

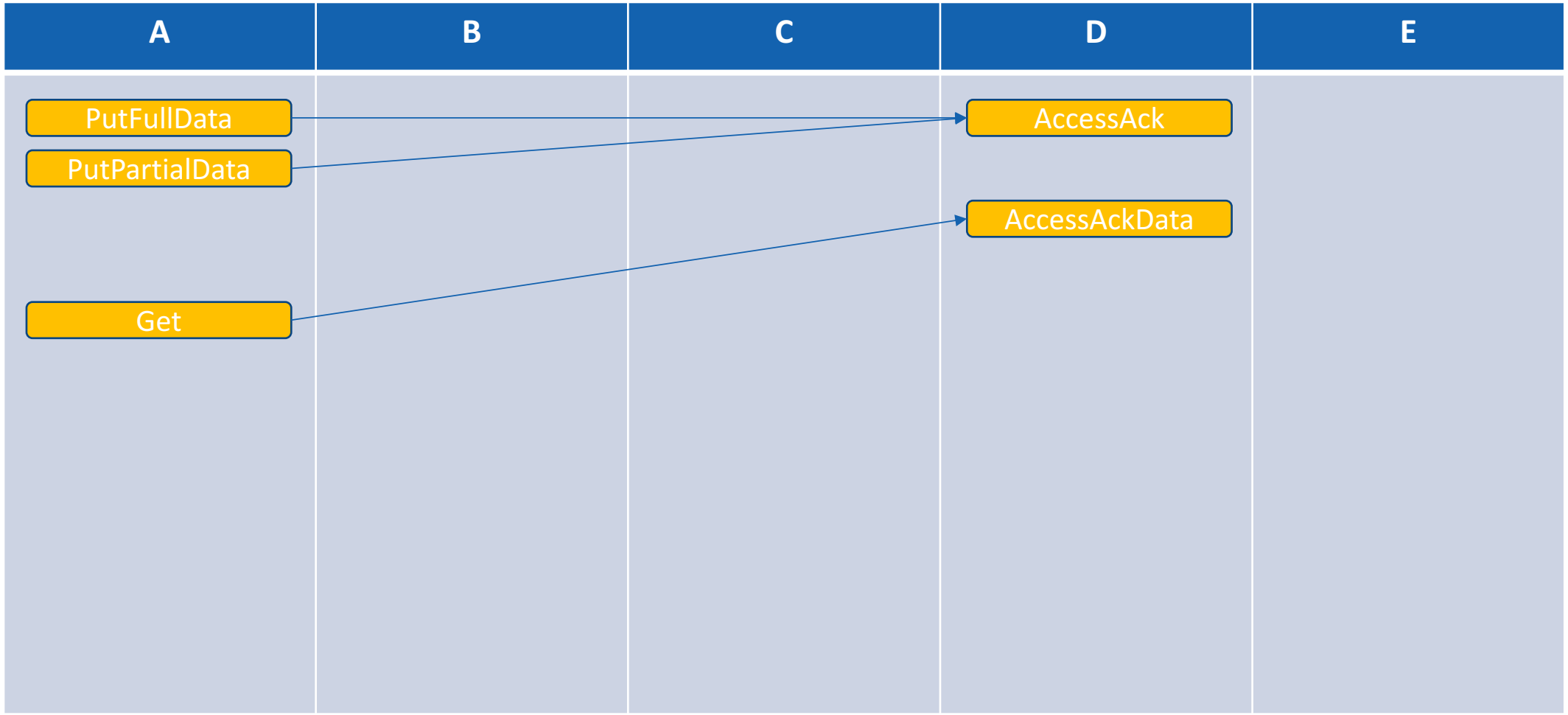


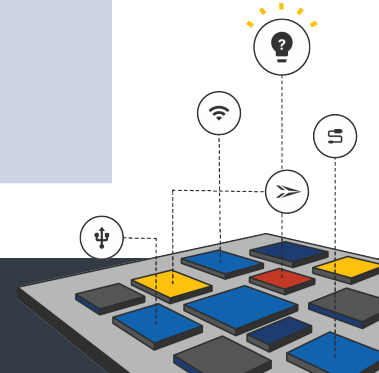
TileLink Protocol Conformance Levels

	TL-UL	TL-UH	TL-C
Read/Write Operations	✔	✔	✔
Multibeat Messages		✔	✔
Atomic Operations		✔	✔
Hint (Prefetch) Operations		✔	✔
Cache Block Transfers			✔
Priorities B+C+E			✔

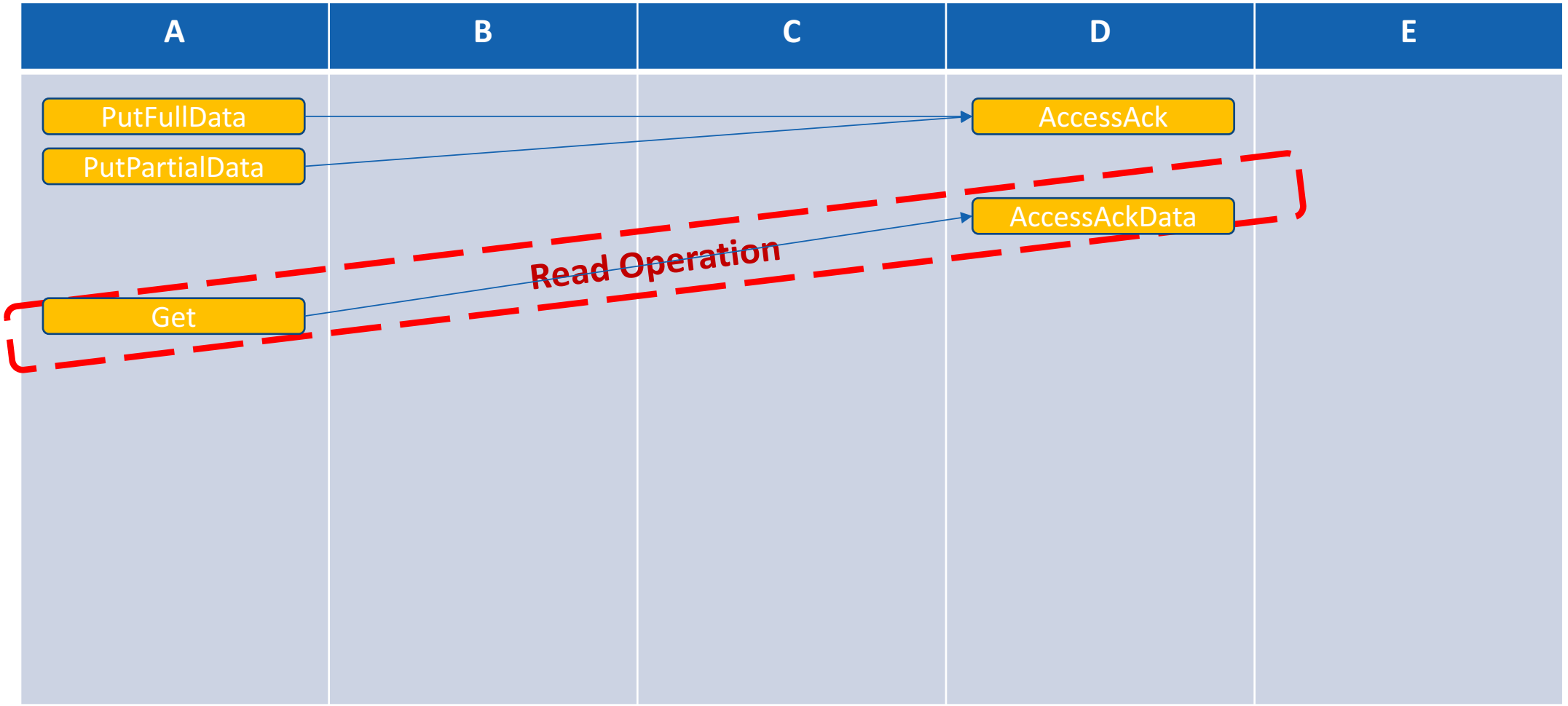


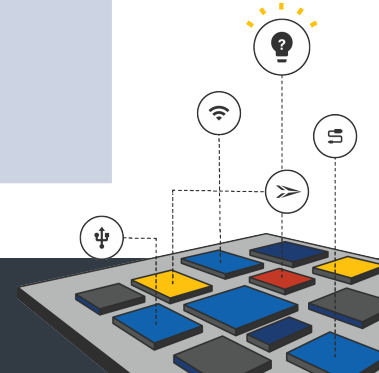
Operations are composed of Messages (TL-UL)



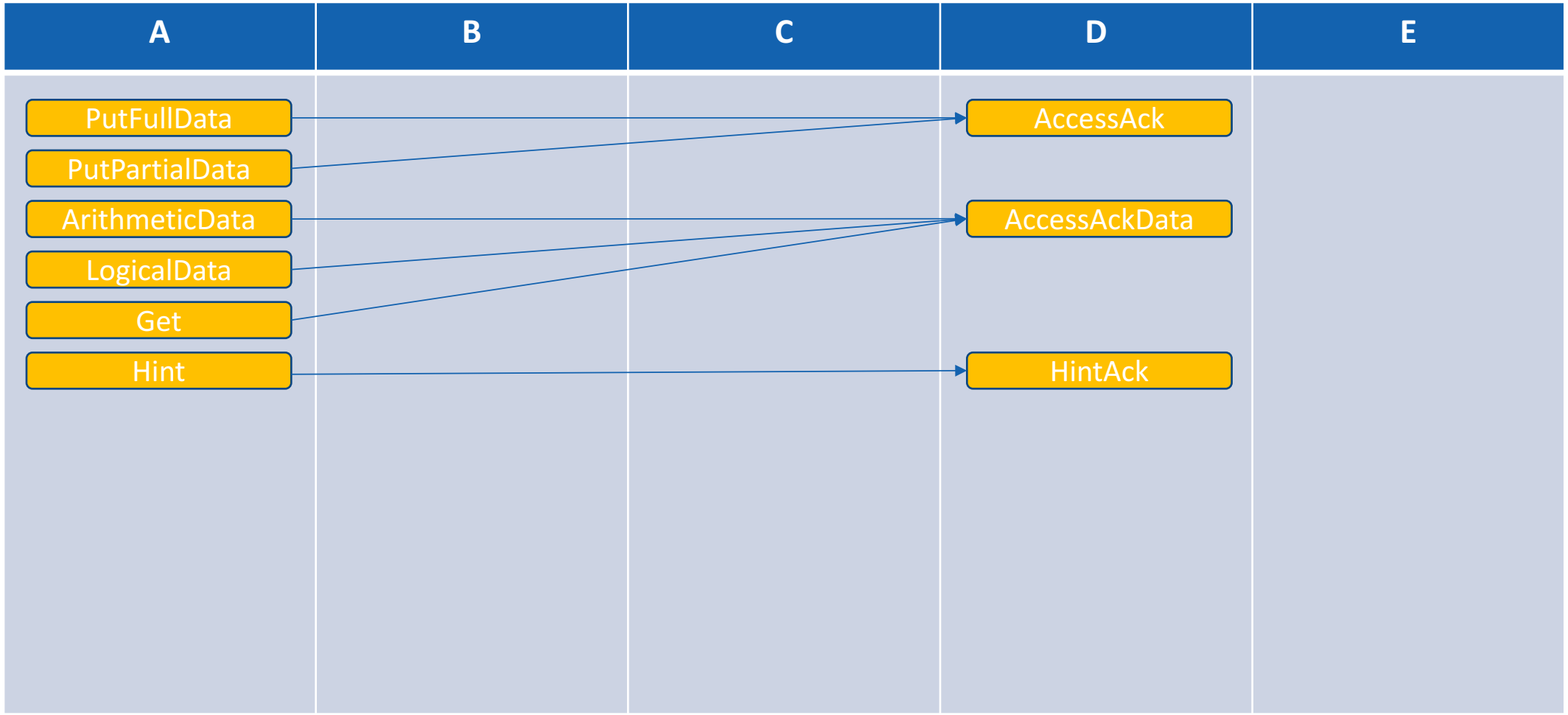


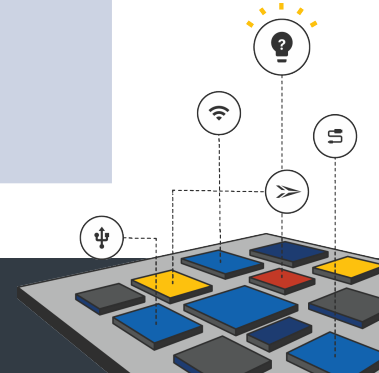
Operations are composed of Messages (TL-UL)



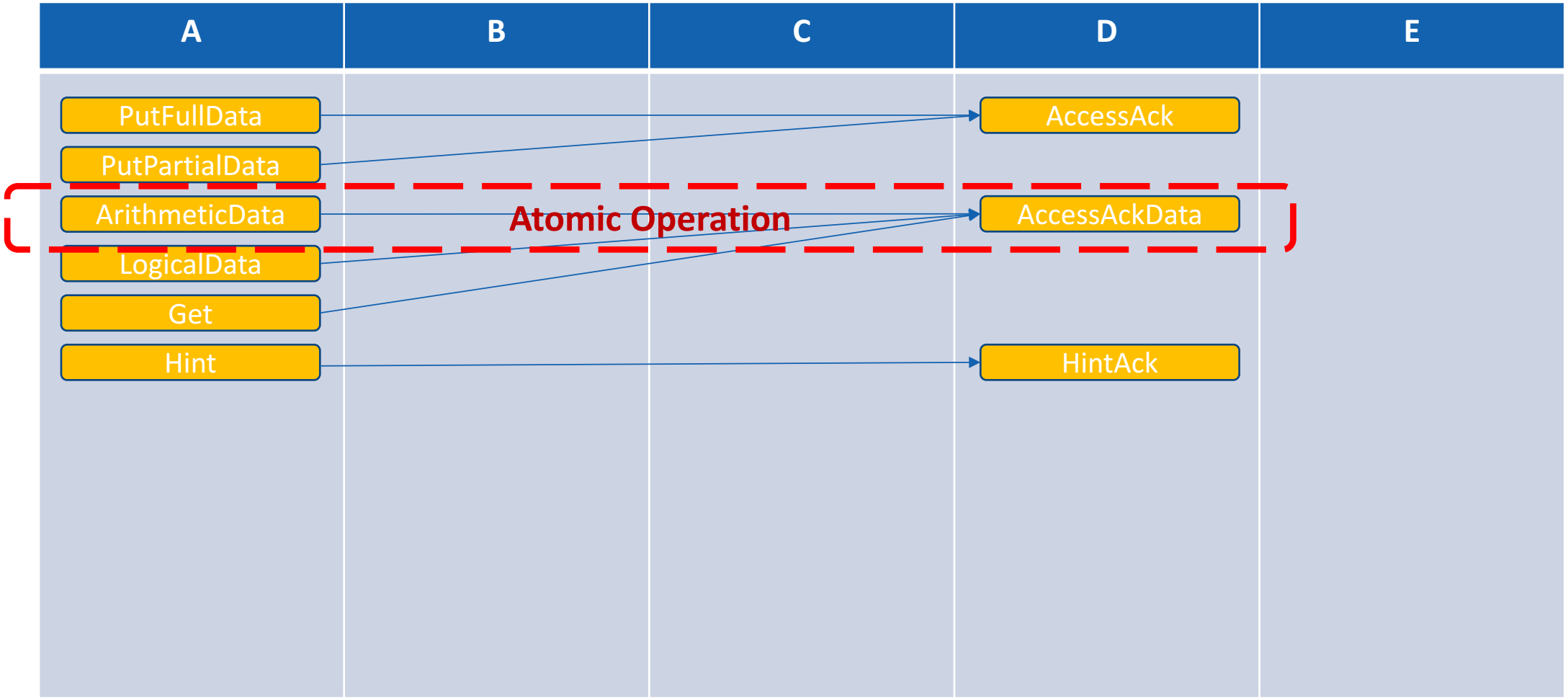


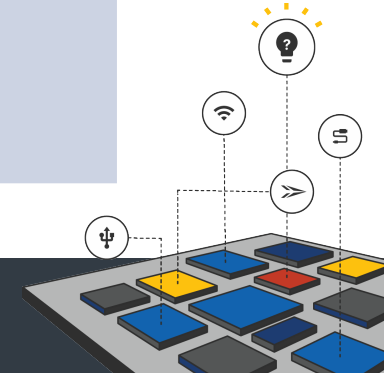
Operations are composed of Messages (TL-UH)



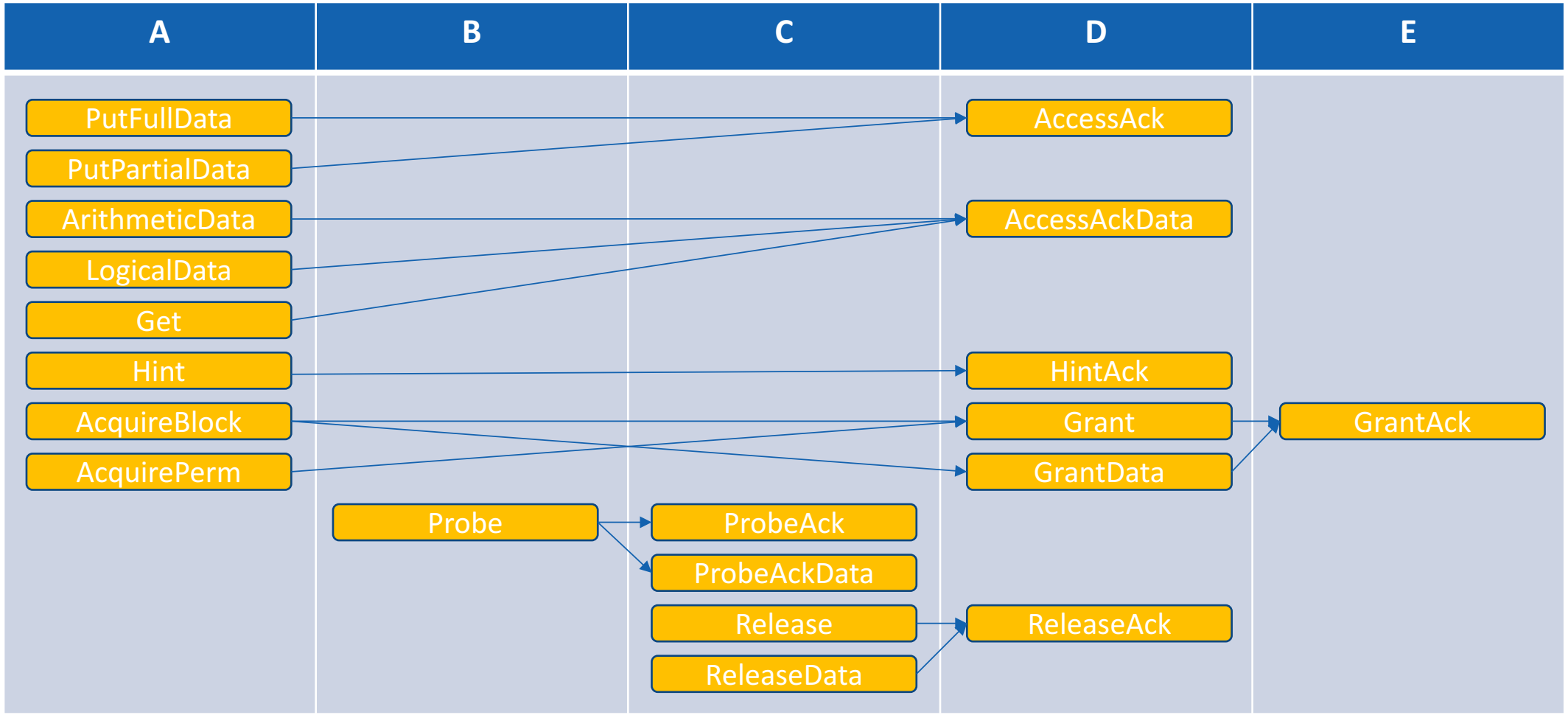


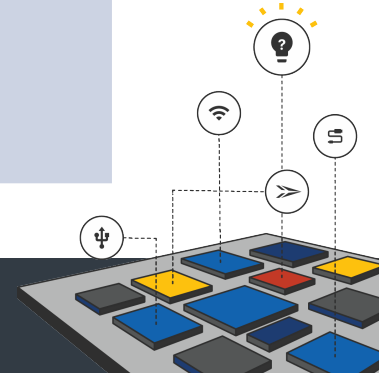
Operations are composed of Messages (TL-UH)



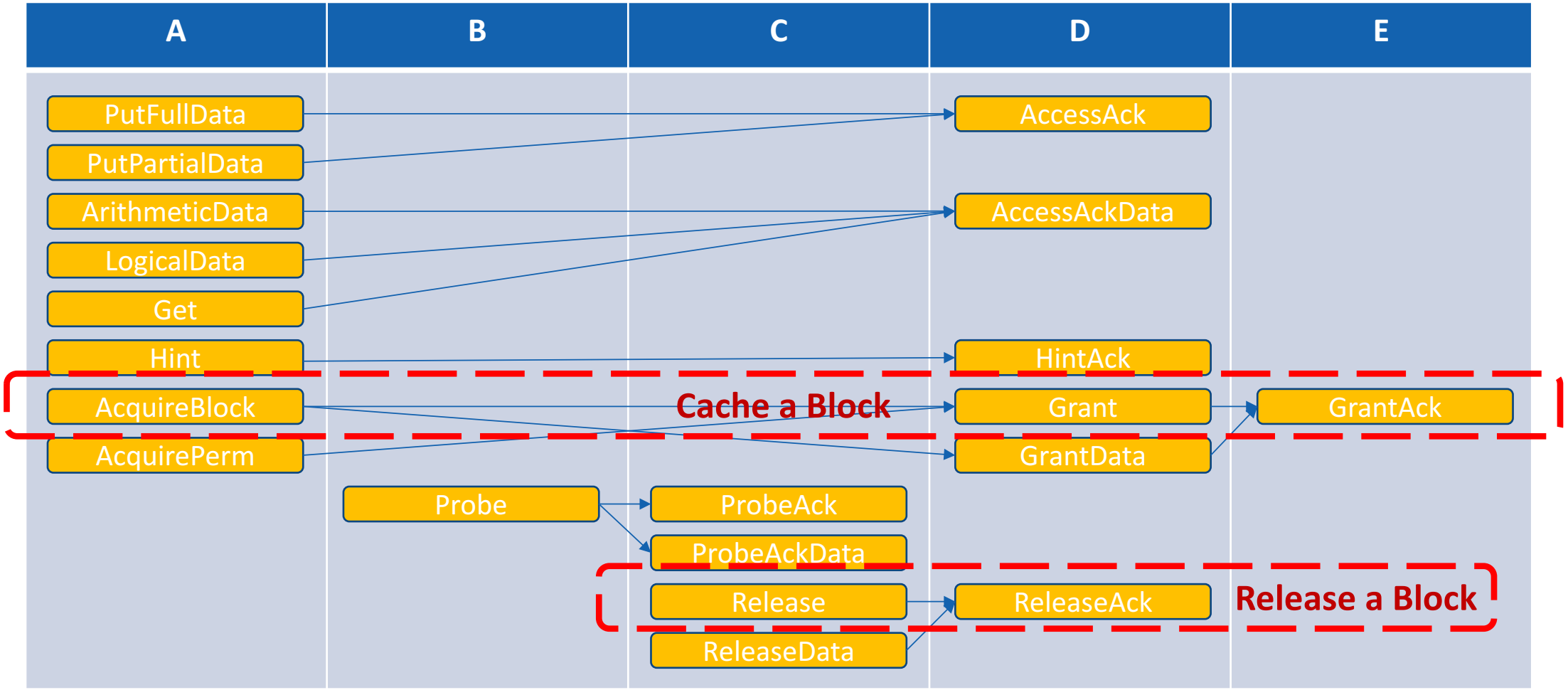


Operations are composed of Messages (TL-C)

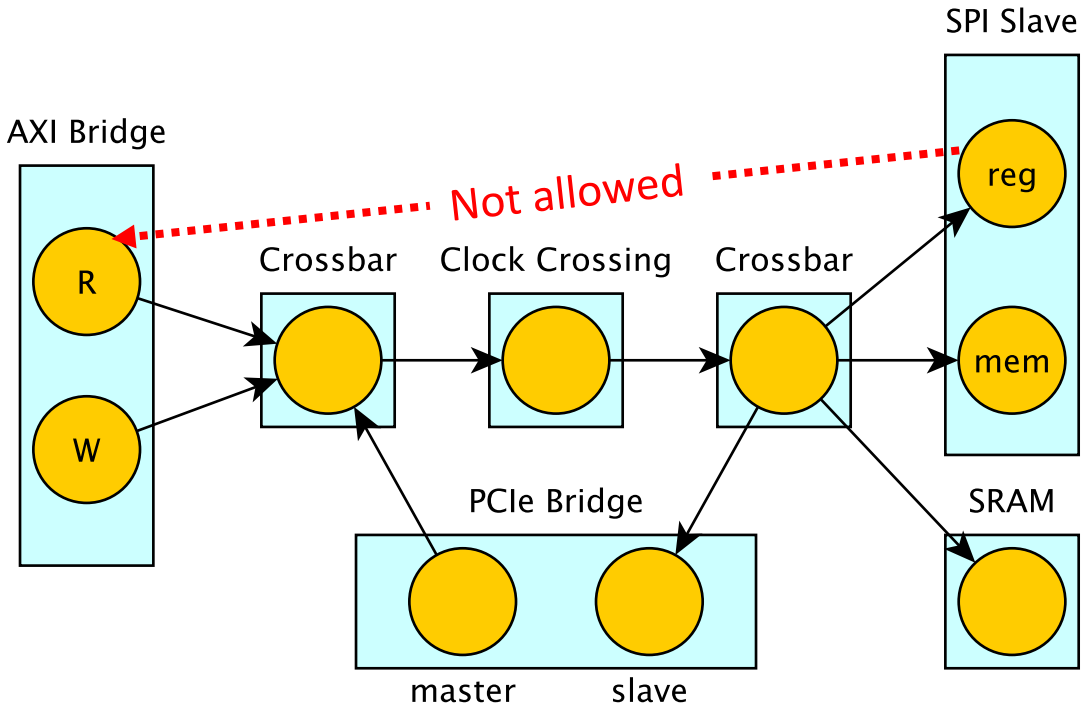




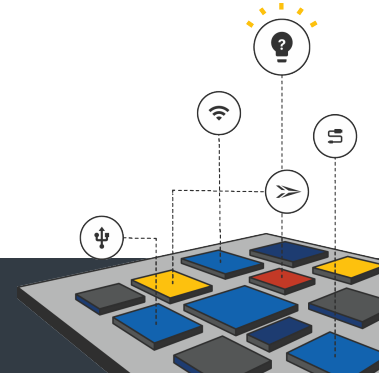
Operations are composed of Messages (TL-C)



Simplifying Requirement: No agent loops!



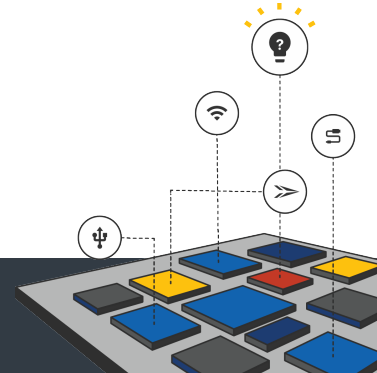
- Bus participants (agents) form a Directed Acyclic Graph (DAG)
  - Prevents message loops that can deadlock the system



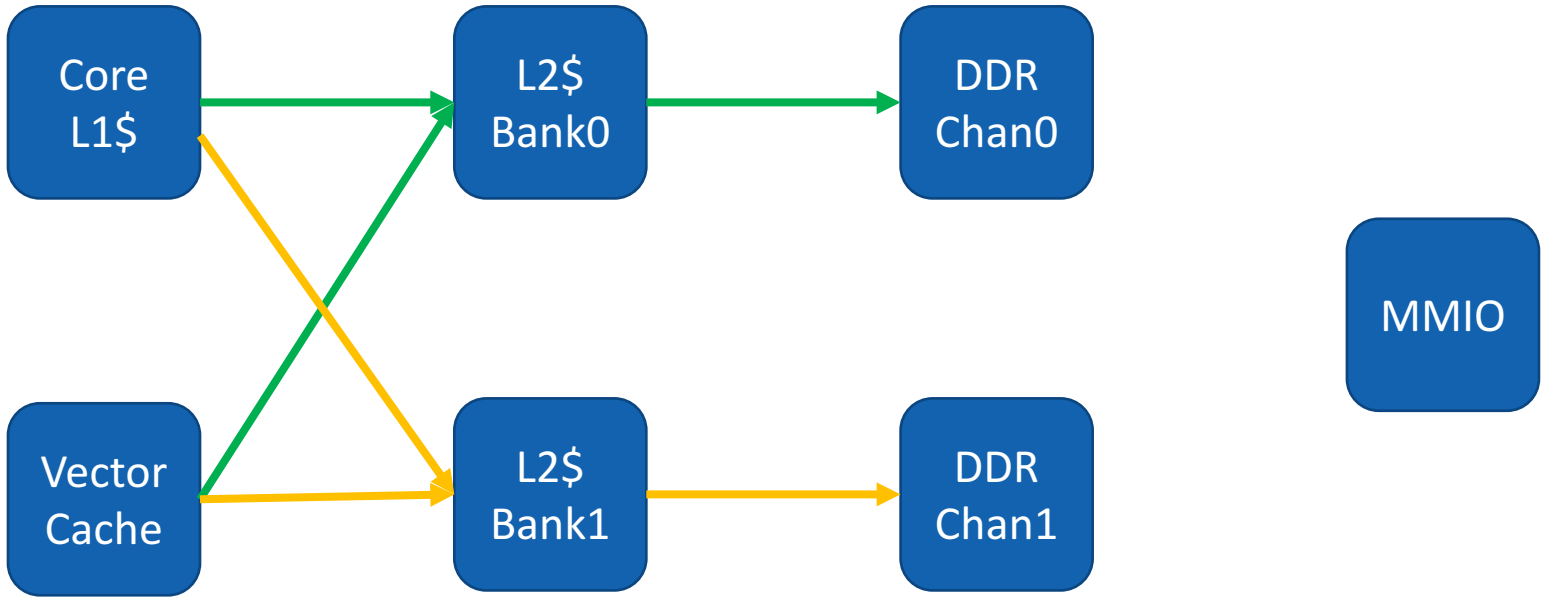
Simplifying Requirement: Strict Priorities!

- Messages have one of five priorities (A < B < C < D < E)
- Lower priority messages never block higher priority messages
- Responses have higher priority than requests

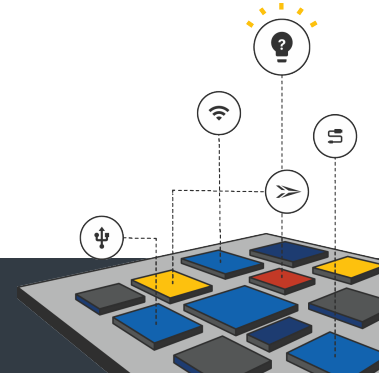
- Simplified protocol (TL-UH) only has two priorities (A < D)
- The BCE priorities prevent deadlock during cache block motion



Simplifying Requirement: Ownership tree!



- Agents which can cache a given cache block form an ownership tree
  - Multiple blocks means we have an ownership forest of logical trees
  - Rule applies to LOGICAL tree relationship; NOT physical links!
  - Uncacheable blocks = a tree of one node



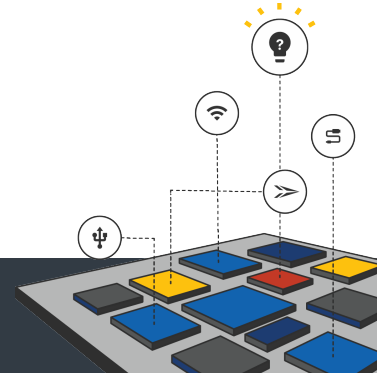
SiFive

On-chip TileLink Serialization

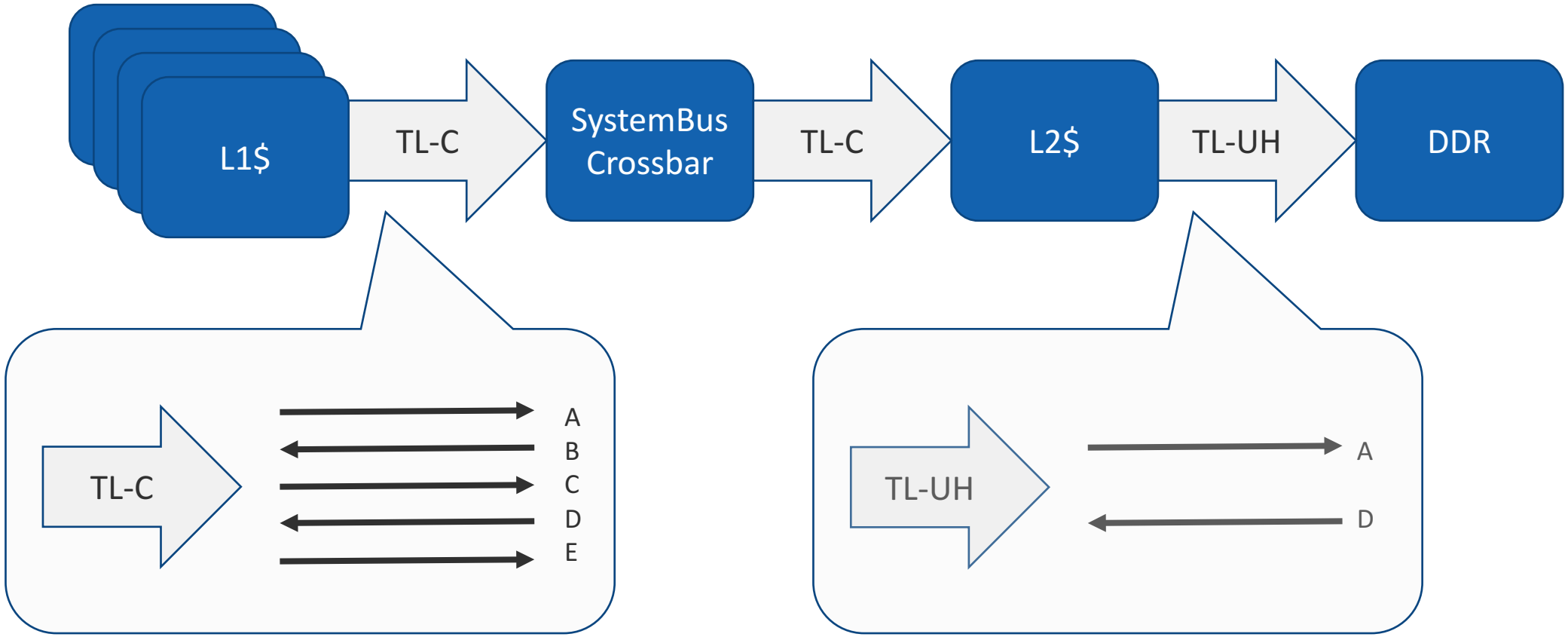
On-chip TileLink Wire Protocol

- Each message priority (ABCDE) has an independent channel
- Messages are transmitted using multibeat bursts
- Ready-valid used to regulate beats
- Variable bus width with 0+ cycle response

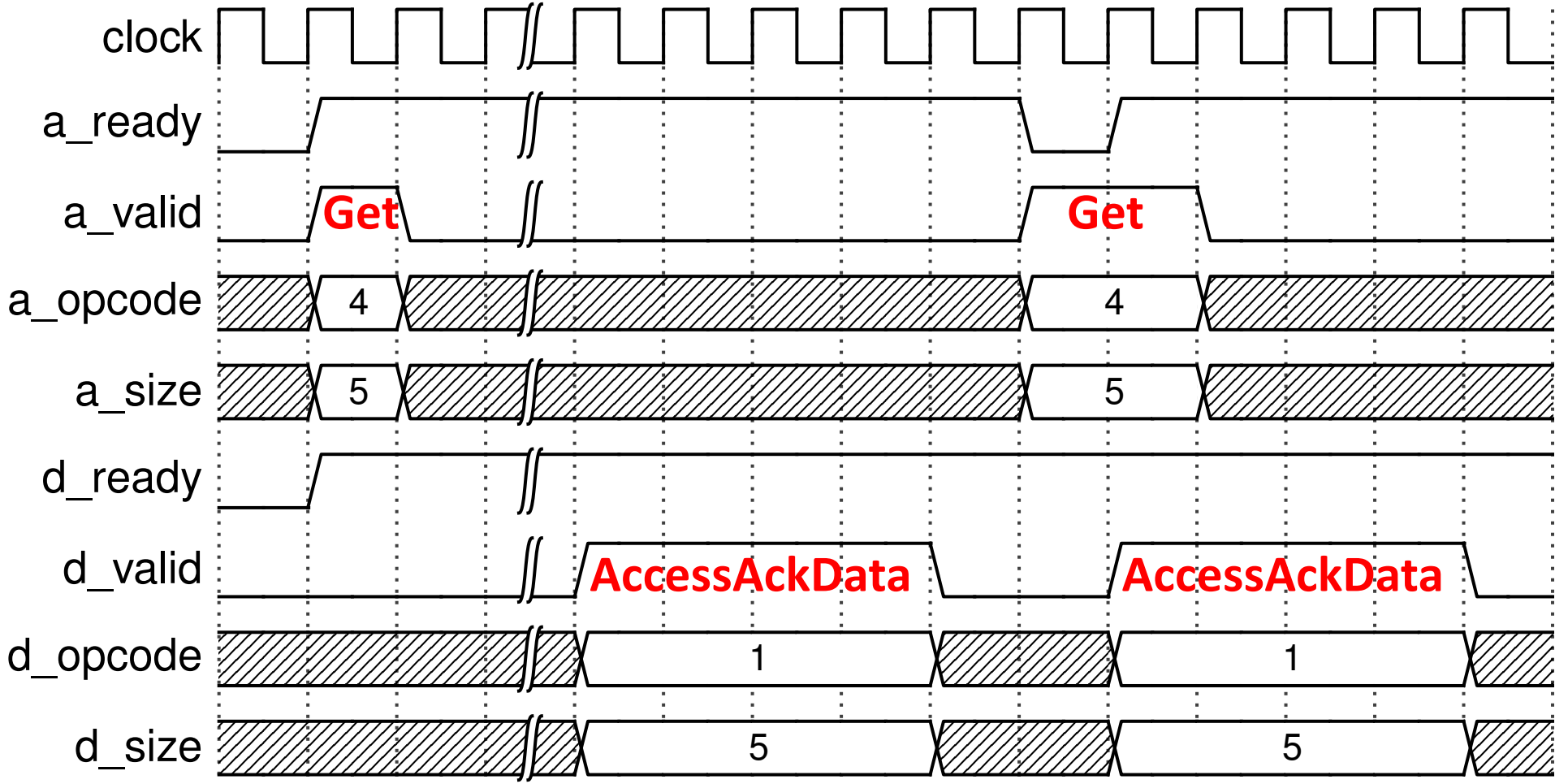
- Off-chip TileLink (ChipLink) makes different choices



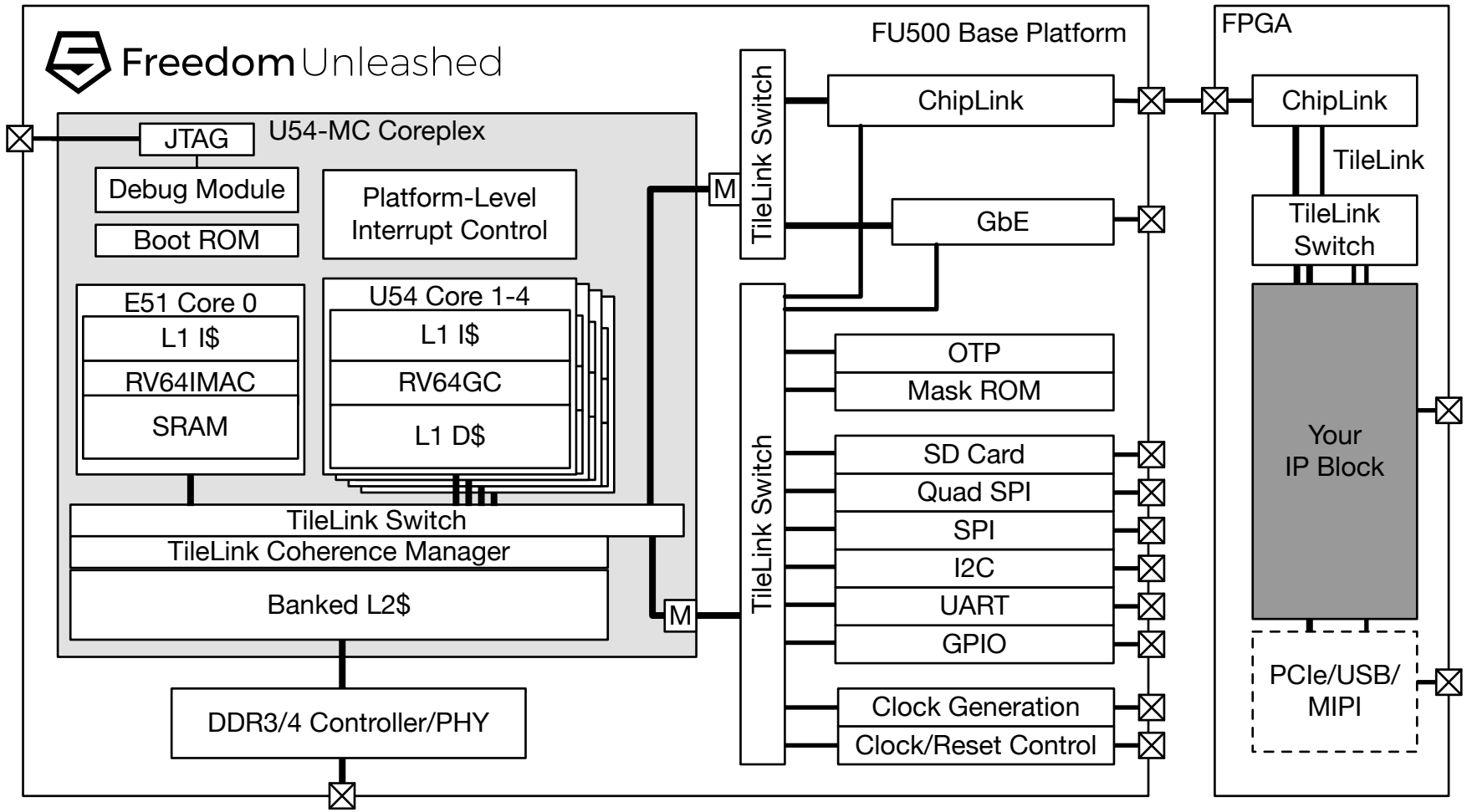
Physically Independent Channel Design



Read Operation

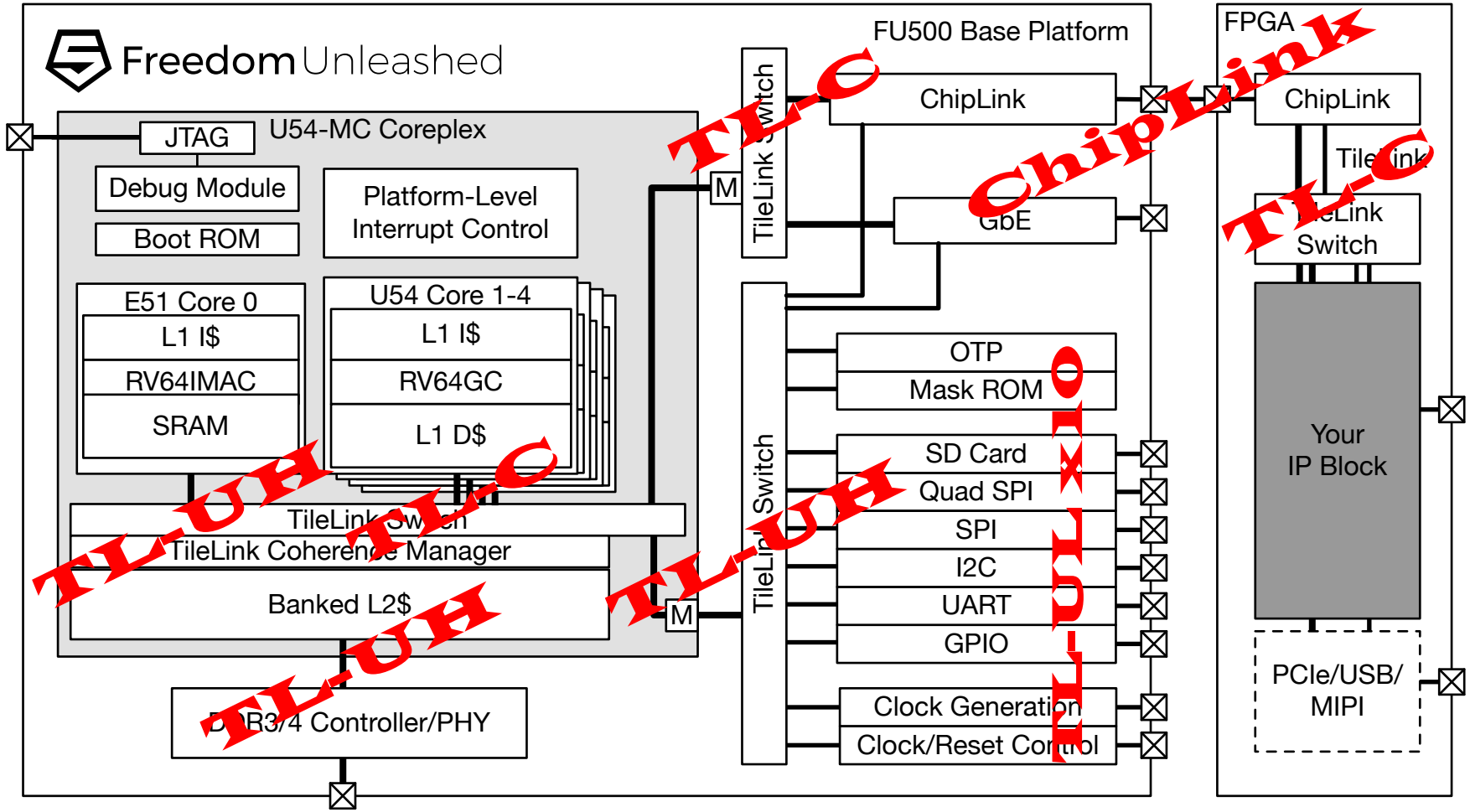


TileLink: The Foundation of SiFive's FU500

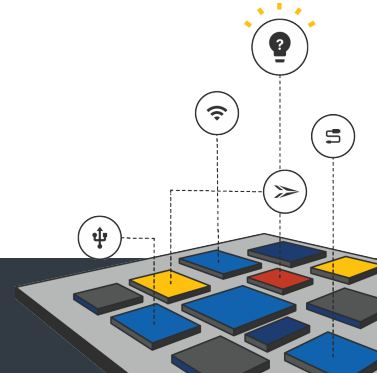


© 2017 SiFive. All Rights Reserved.

TileLink: The Foundation of SiFive's FU500

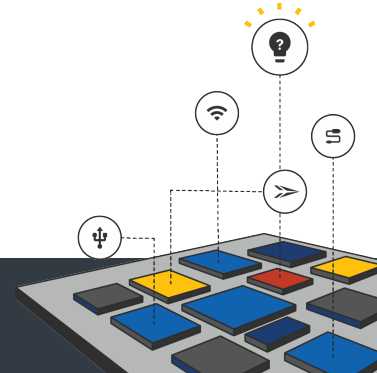


© 2017 SiFive. All Rights Reserved.



Conclusion

- Public spec, Open source, RMP, designed for RISC-V
- Can be bridged to your existing AMBA designs
- If you are building a RISC-V chip, why not use TileLink?
- If you want to learn more:
  - https://www.sifive.com/documentation/tilelink/tilelink-spec/
  - These slides – much more content than presented!

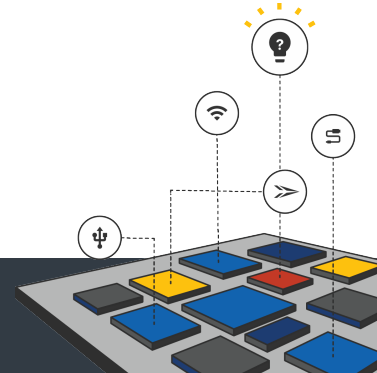



SiFive

Messages are composed of Beats, containing:

- The unchanging message header, including
  - opcode: the message type
  - size: the base-2 logarithm of the number of bytes in the data payload

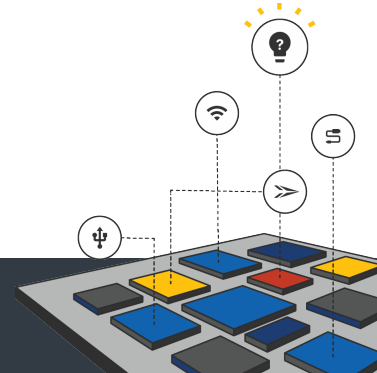
- An optional multibeat data payload
  - presence depends on the opcode (only *Data message type)
  - number of beats is calculated from the size



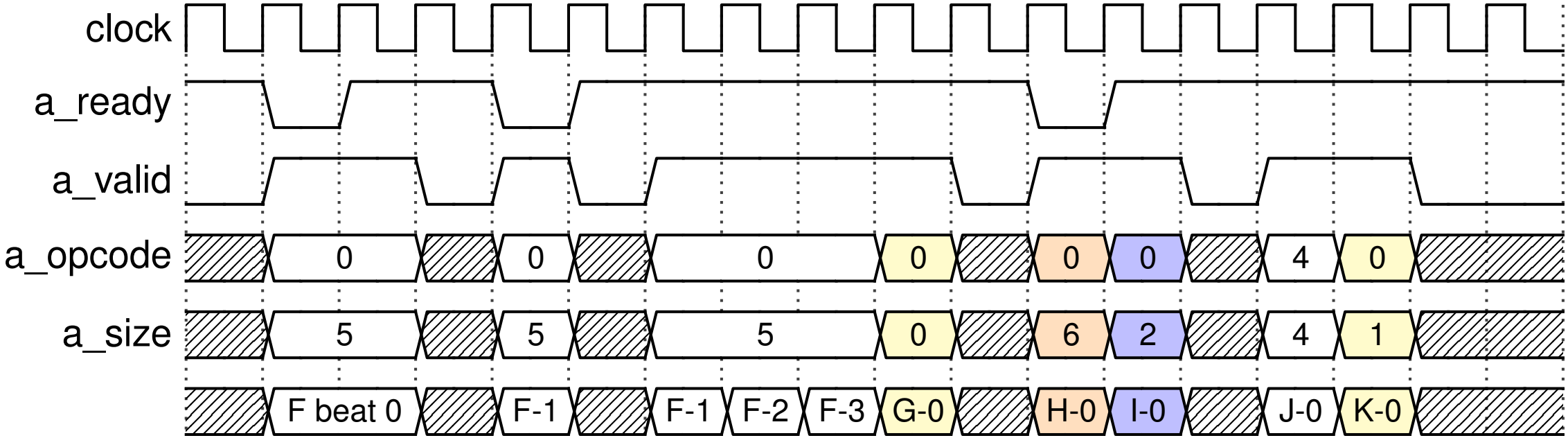
Beats are regulated by ready-valid

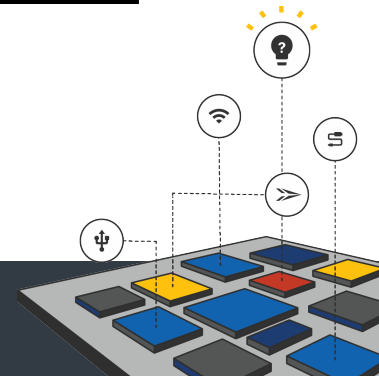
- The receiver provides ready
  - If ready is LOW, the receiver must not process the beat
  - If ready is LOW, the sender may elect to try a different message
  - Ready may combinationally depend on the beat payload

- The sender provides valid + the beat payload
  - If valid is LOW, the payload may be an illegal TileLink message
  - Valid + the payload may not combinationally depend on ready



Ready-Valid example



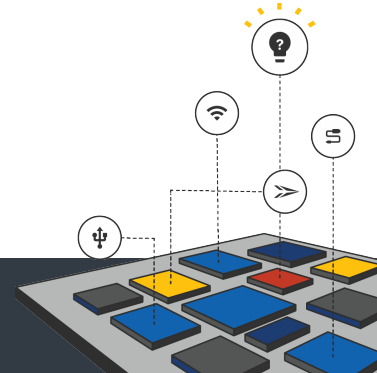


SiFive

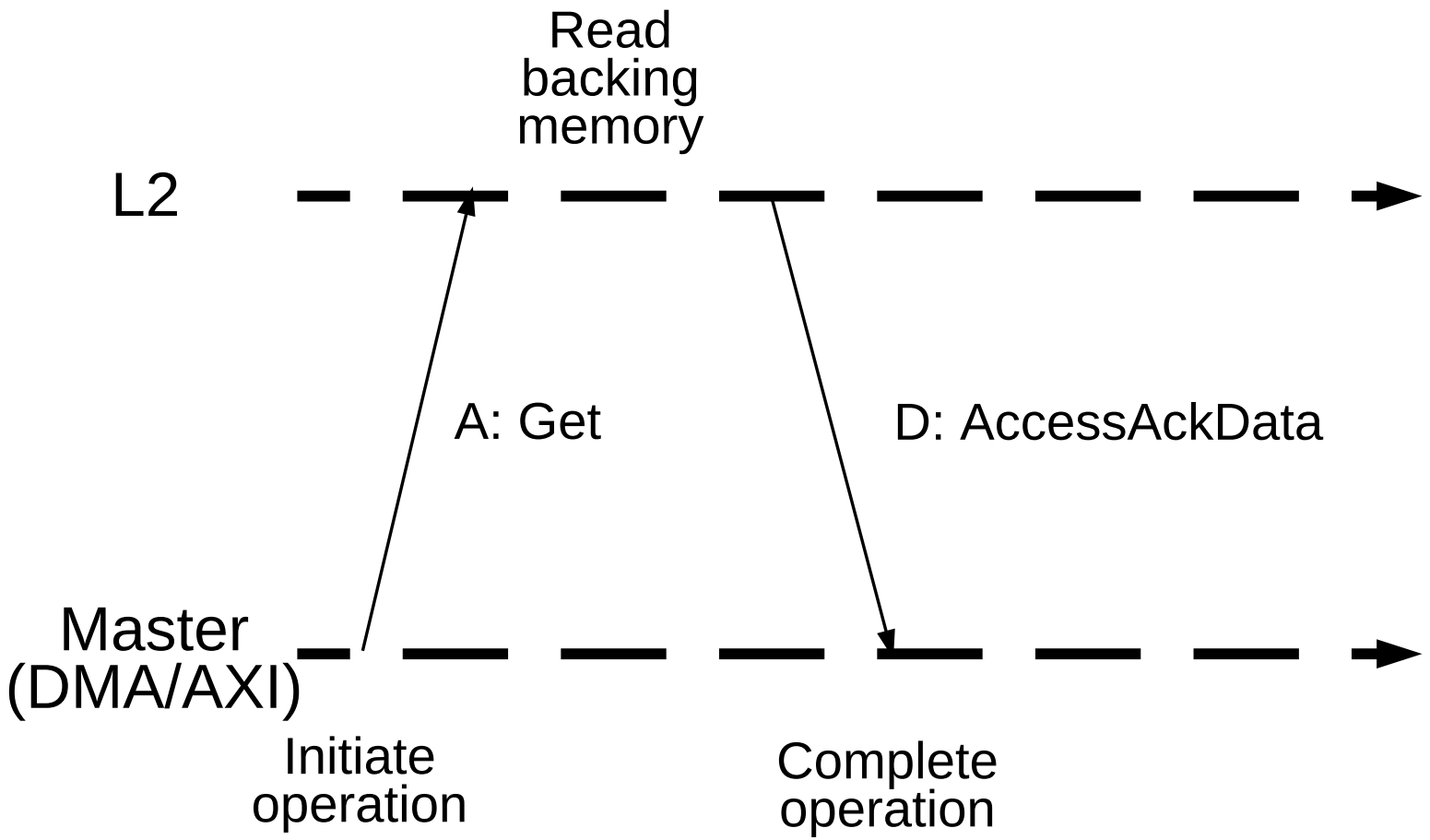
Request-Response timing

- The first beat of a response message may be presented
  - After an arbitrarily long delay
  - On the same cycle as the first beat of the request, but not before
  - Before all the beats of burst request have been accepted

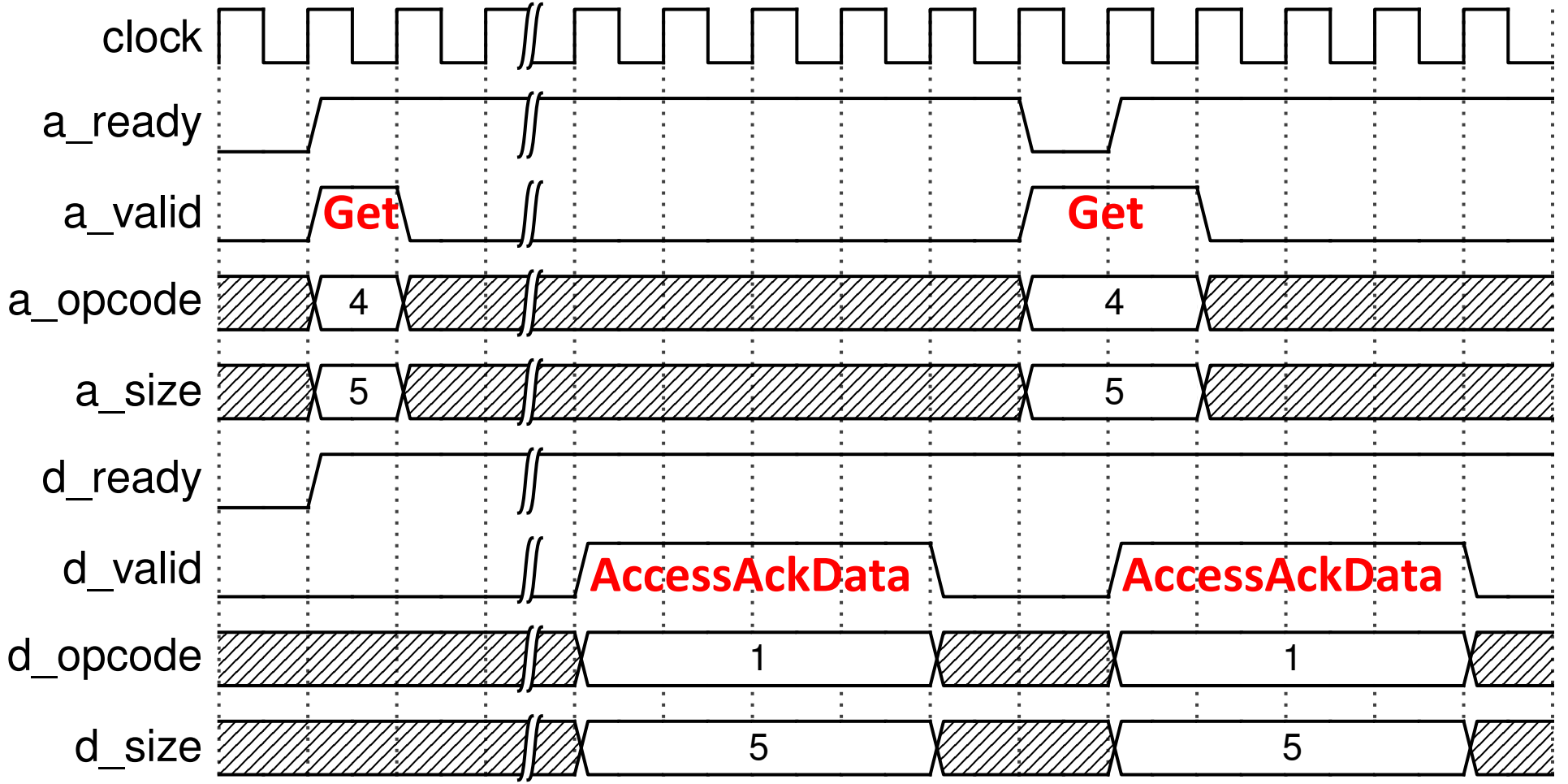
- Responses indicate receiver has successfully sequenced the request
  - The L2 may send an AccessAck before the last beat of a large Put has arrived
  - Simple slaves can send their response in the same cycle as their request



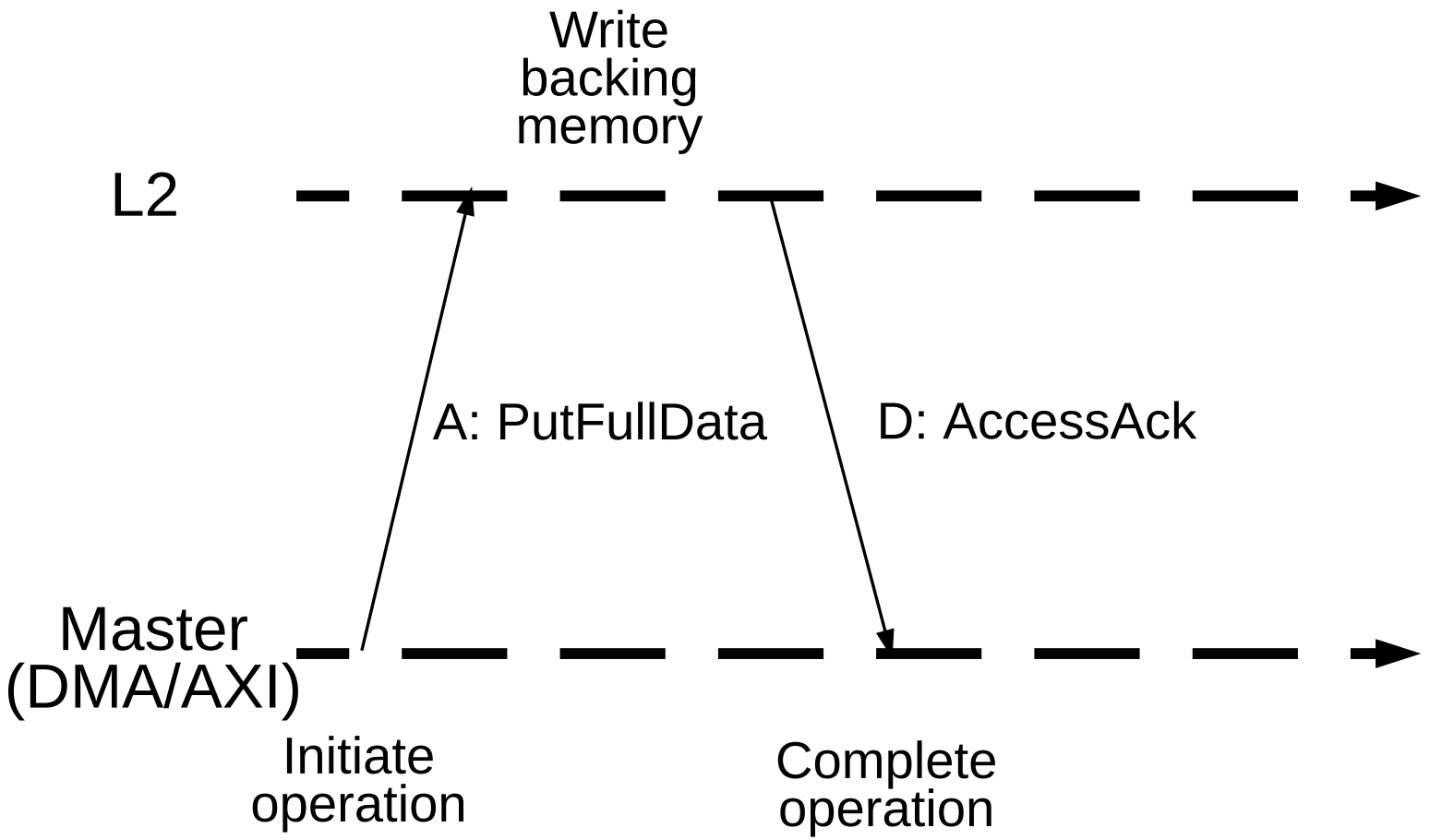
Read Operation



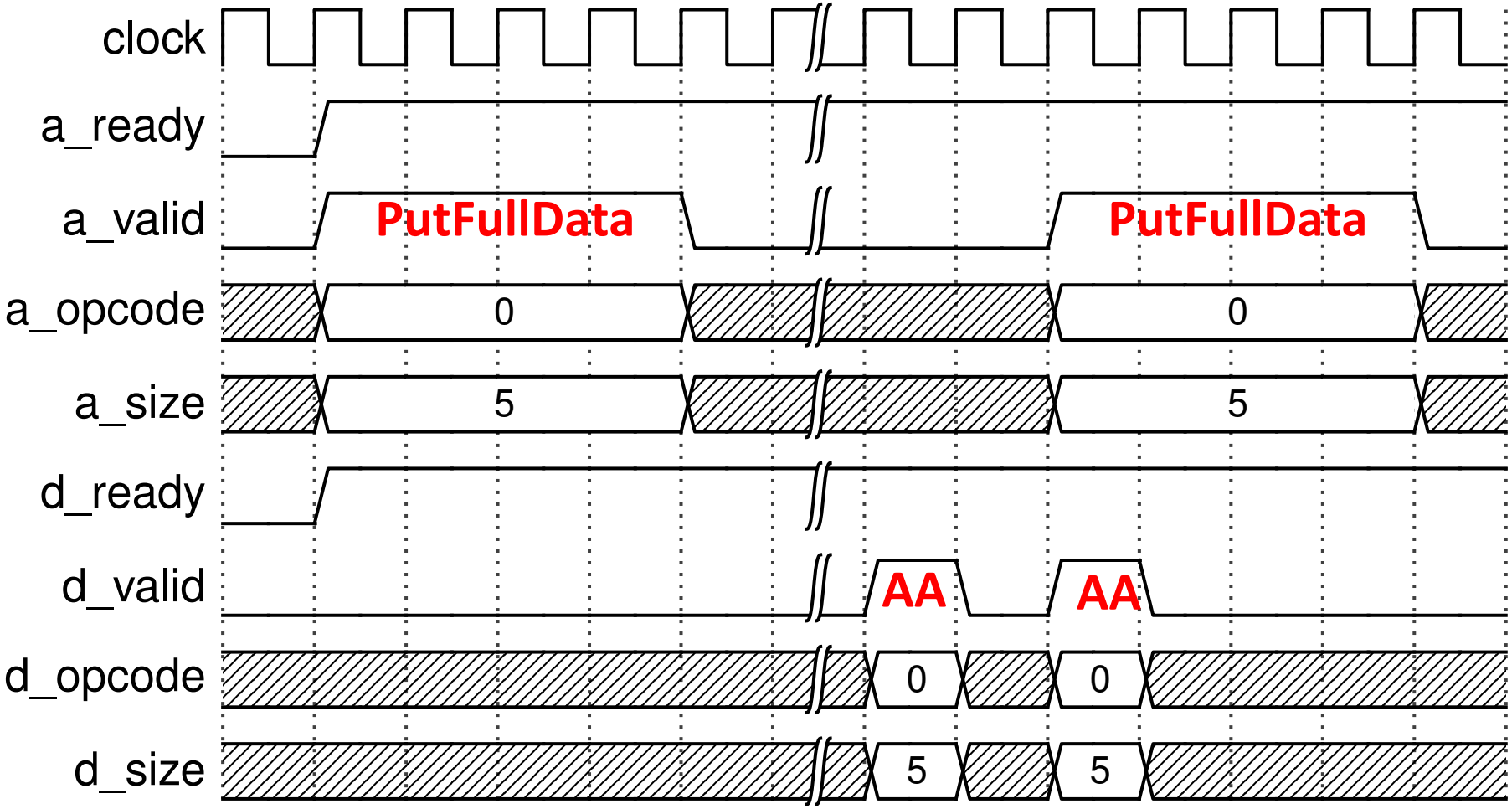
Read Operation



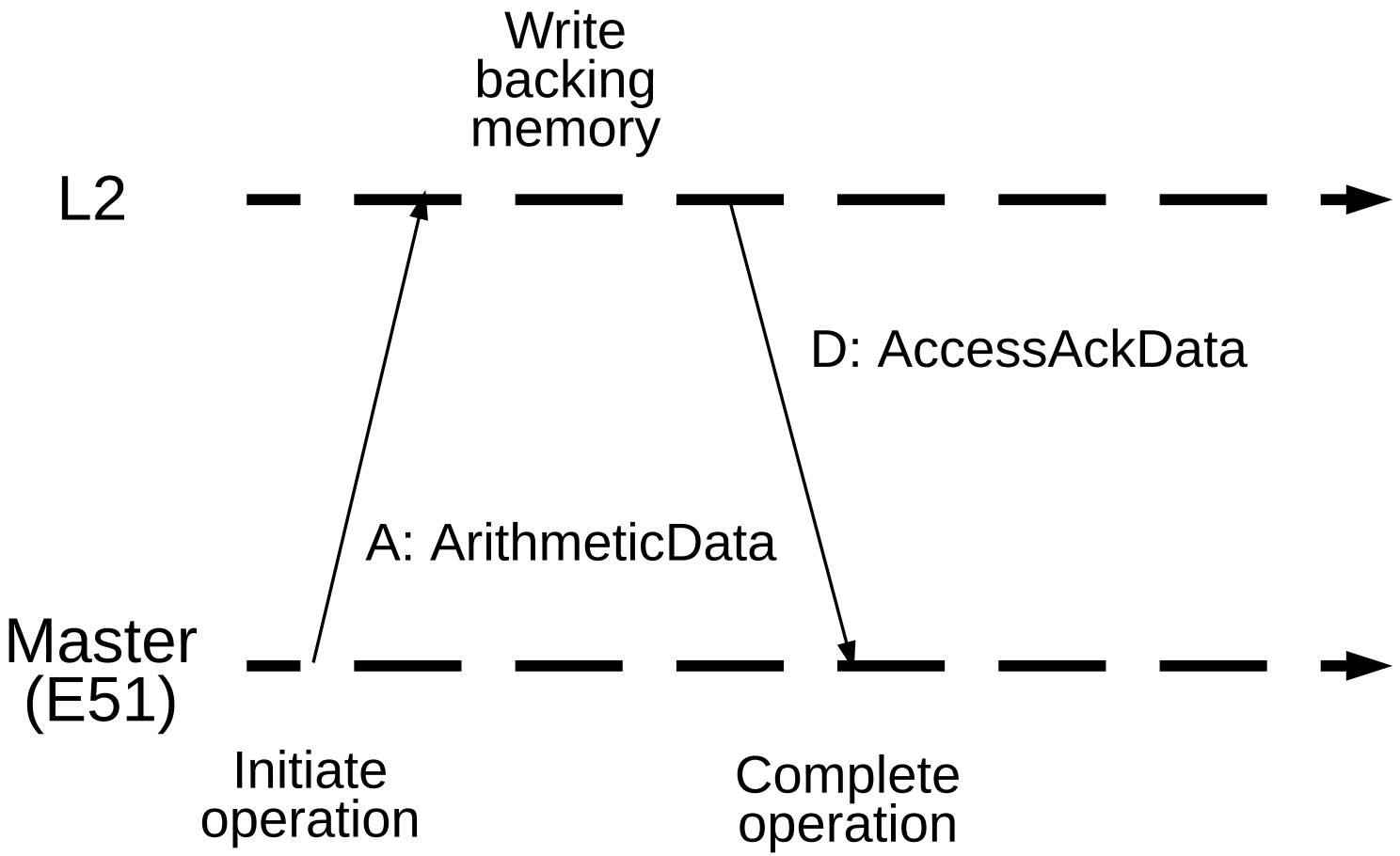
Write Operation



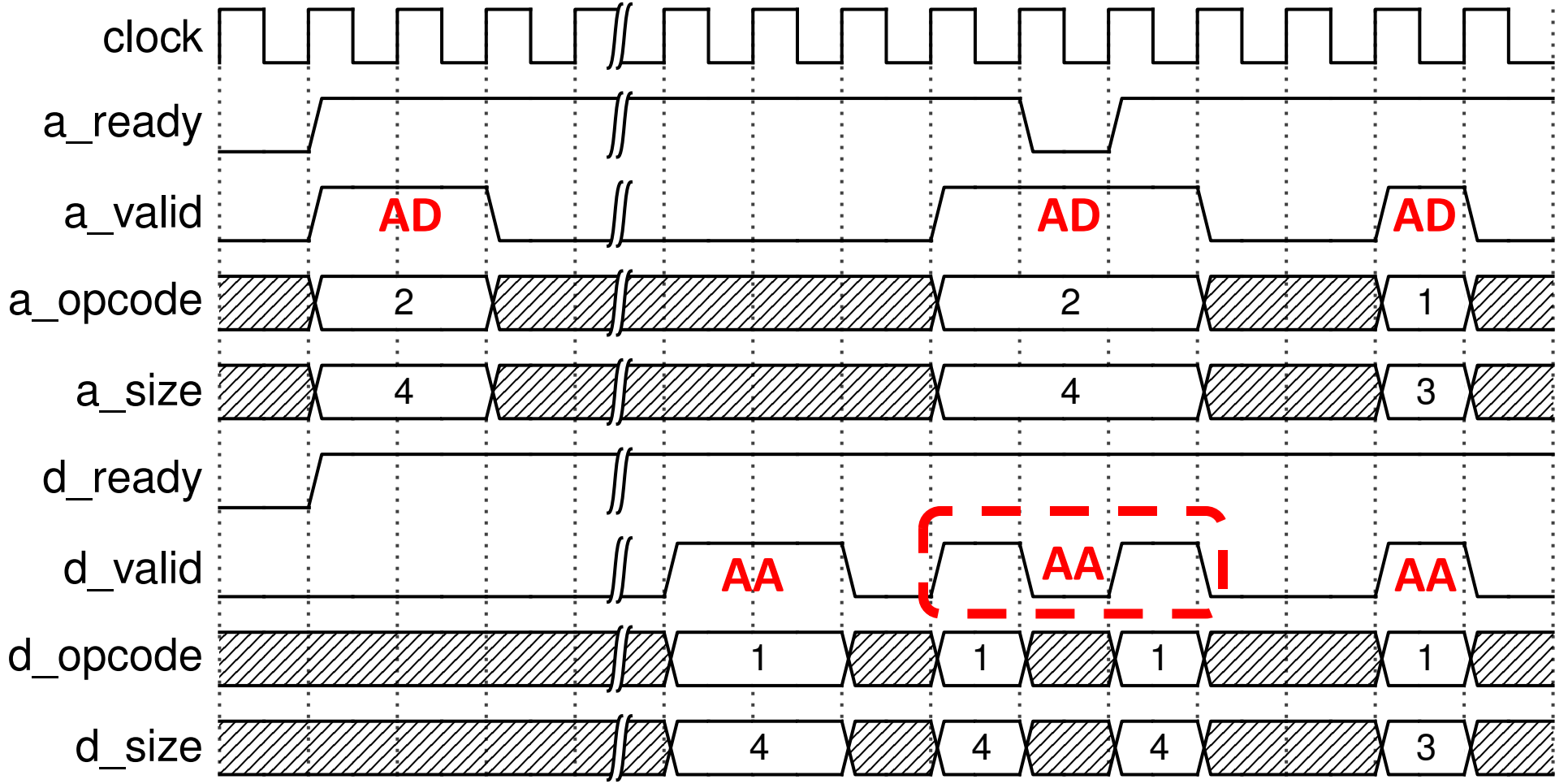
Write Operation



Atomic Operation (TL-UH)

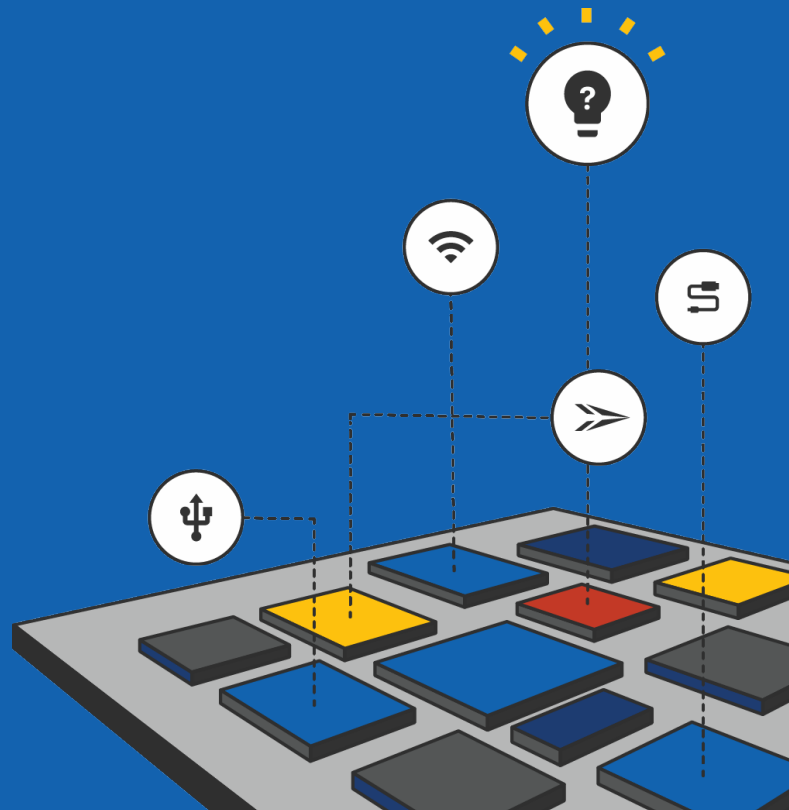


Atomic Operation (TL-UH)



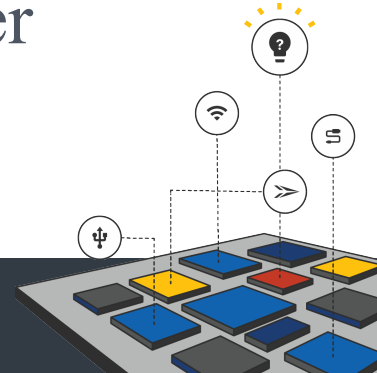
SiFive

TileLink Coherence



What does TL-C add?

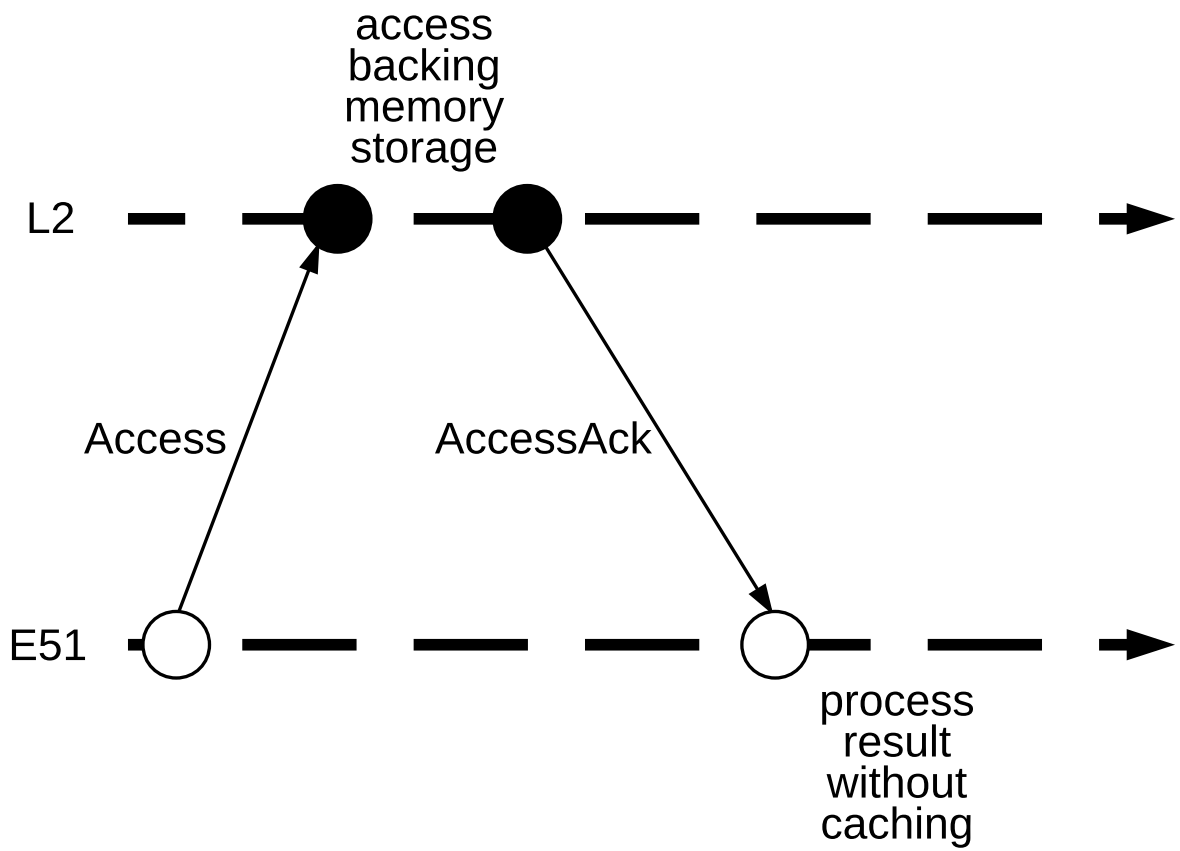
- Masters can create local replicas of cacheable blocks
  - This can improve both throughput and access latency
- Additional channels, B+C+E, are needed to maintain coherence
  - These are used to reclaim and sequence ownership of the block
- Three new operations (Acquire, Probe, Release) for block transfer



Who owns the block? Access Operations

Reads and Writes do not affect the ownership of the block.

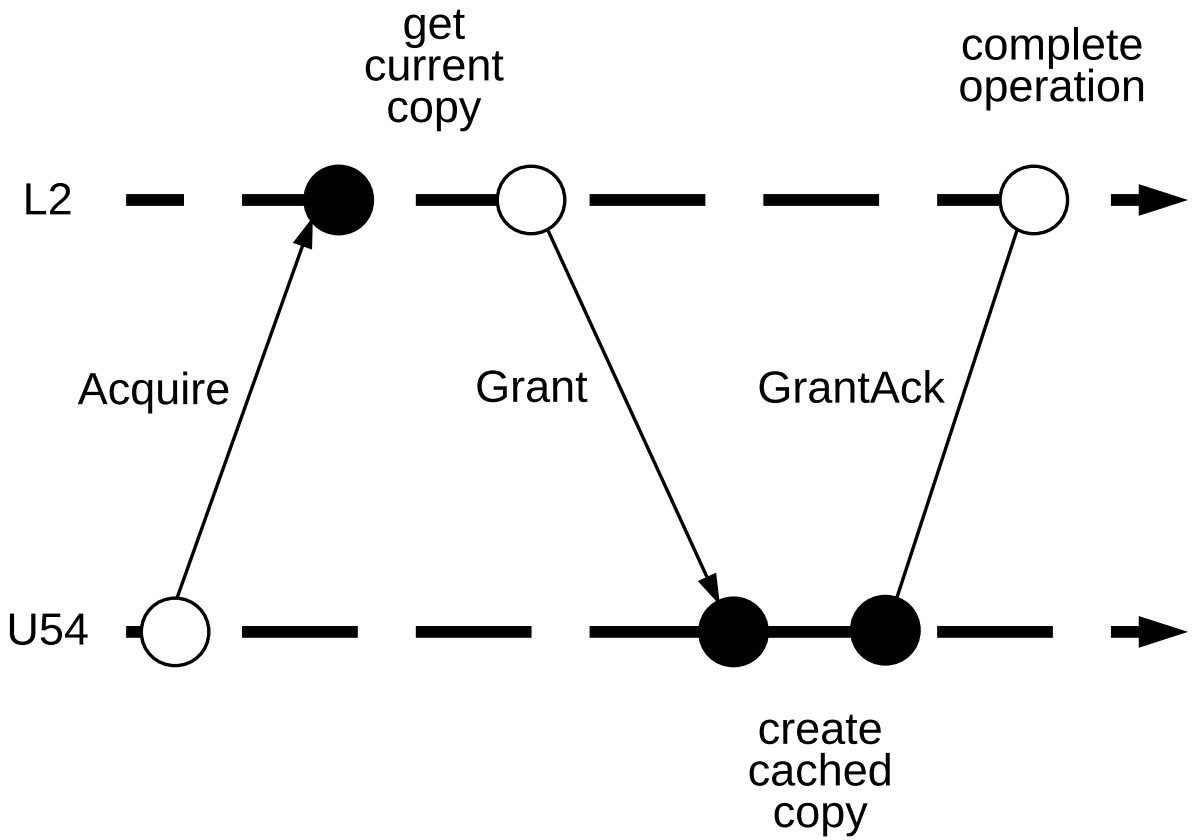
However, the access is coherent with the rest of the system.



Who owns the block? Acquire Transfer

A master can acquire a copy of the cache block by requesting it from a slave.

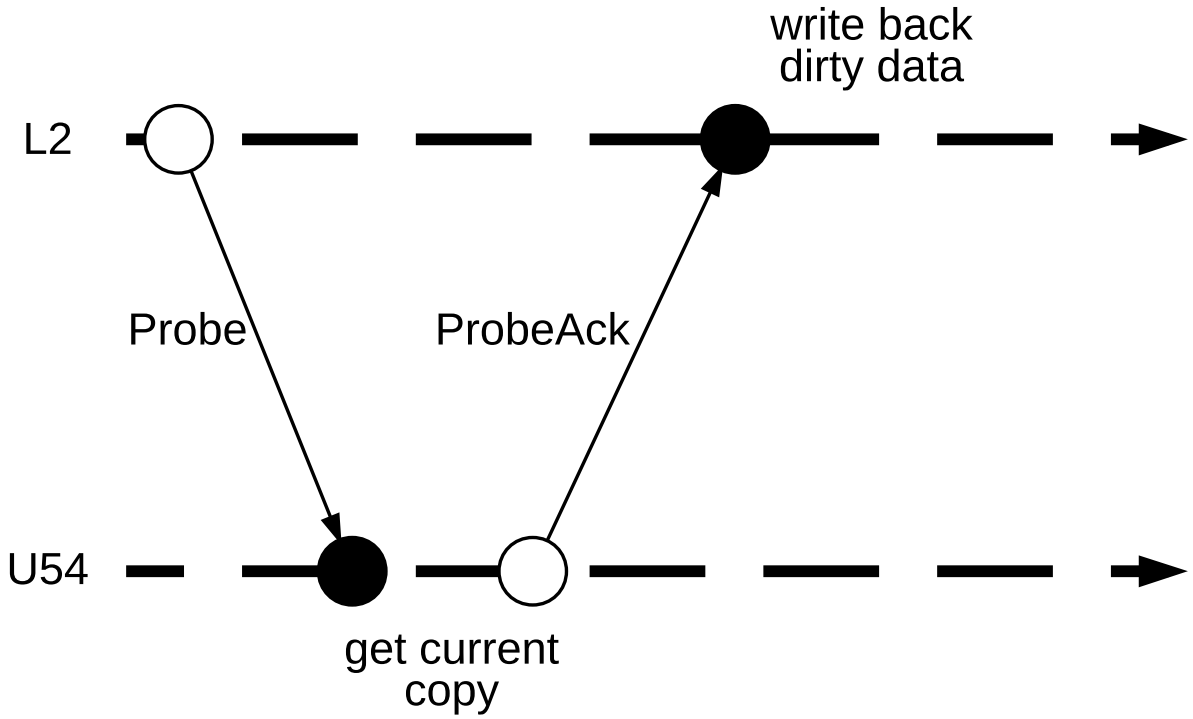
GrantAck is required to synchronize when the L2 can reacquire the block.

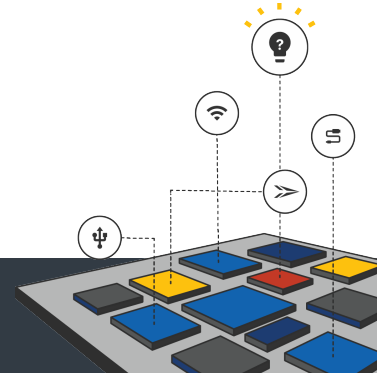


Who owns the block? Probe Transfer

The L2 can reacquire a copy of the block by probing it from the U54.

ProbeAck is required to synchronize when the L2 can allow a new Acquire.

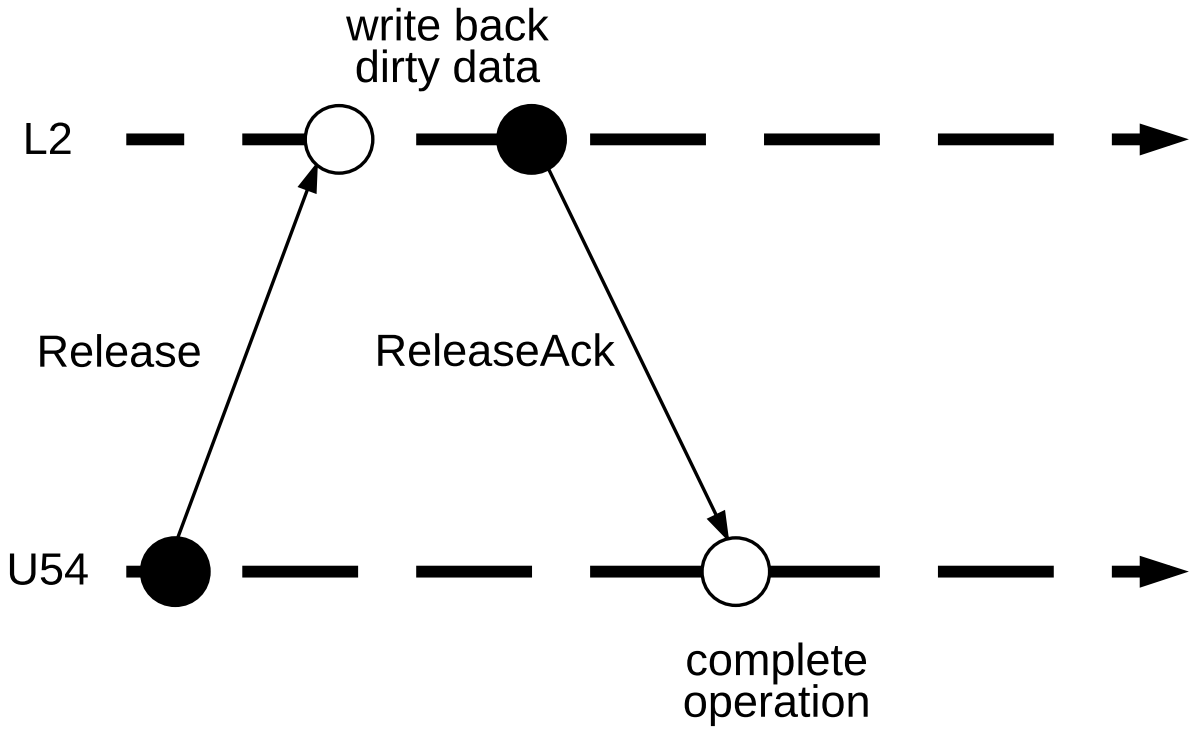




Who owns the block? Explicit Release Transfer

A master can release its copy of the cache block back to the L2.

ReleaseAck is required to synchronize when the U54 can reacquire the block.

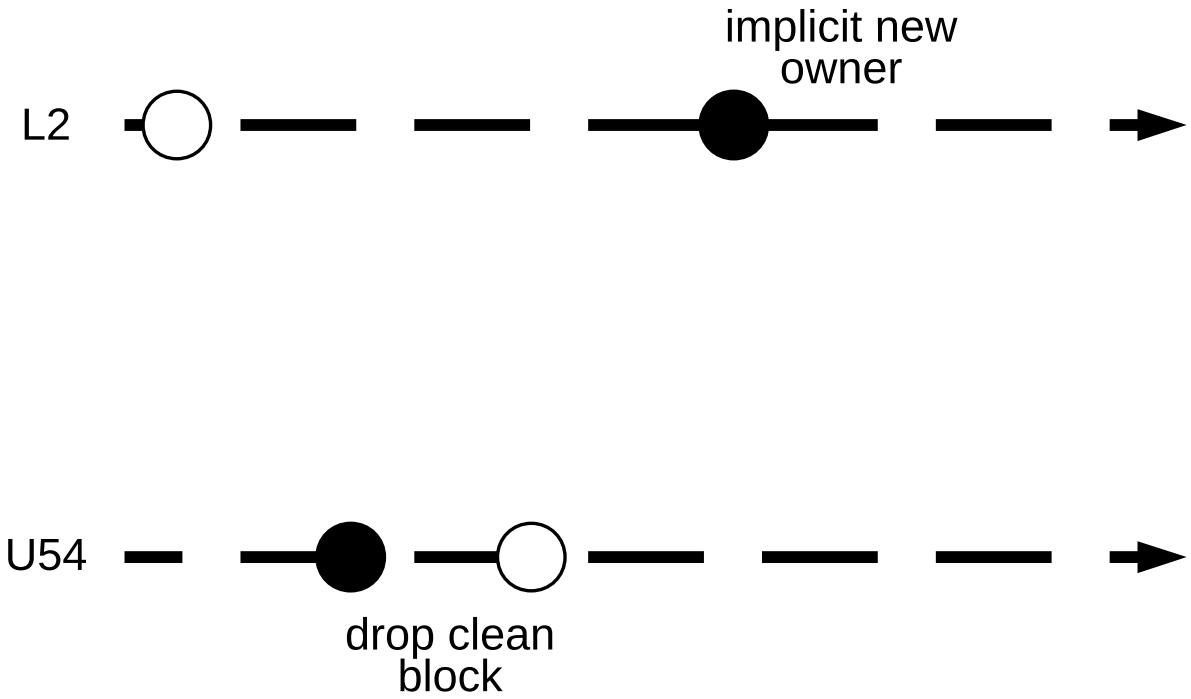


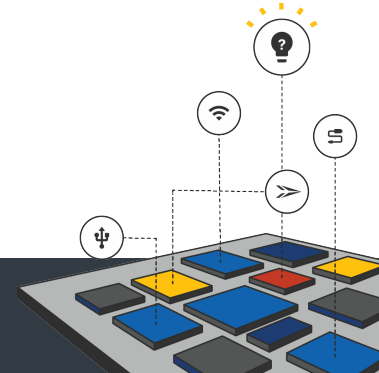
Who owns the block? Silent Release

A master can release its copy of the cache block without telling the L2.

When the L2 later needs the block, a Probe will indicate its absence in U54.

Only safe for clean blocks



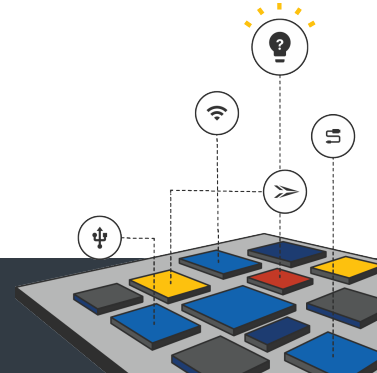


Composition of TileLink transfers

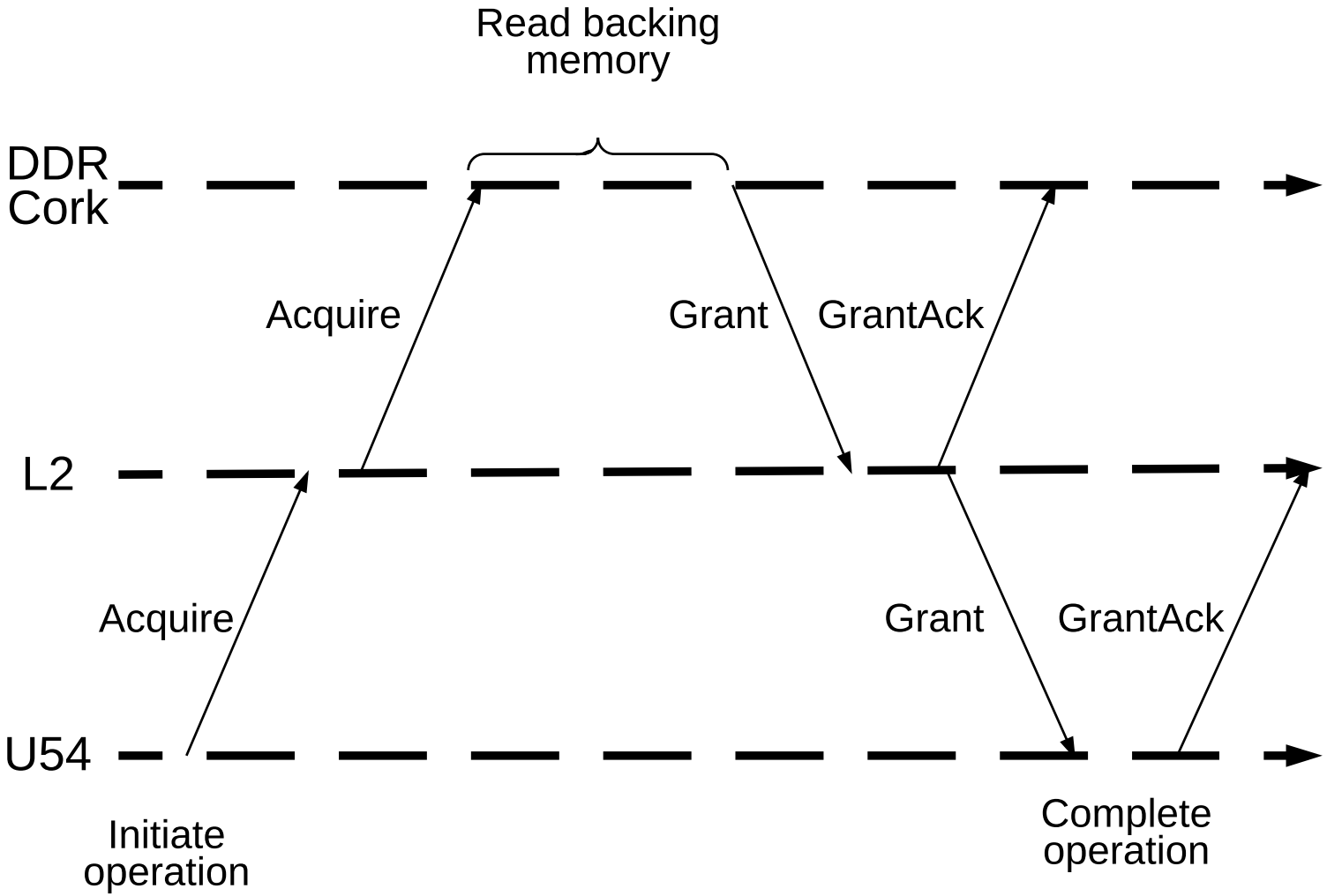
- The three transfer operations can occur concurrently
  - Move a block through more than one agent
  - Acquires often come with evictions (Releases)
  - Acquire race between cores
  - Racing Releases and Acquires

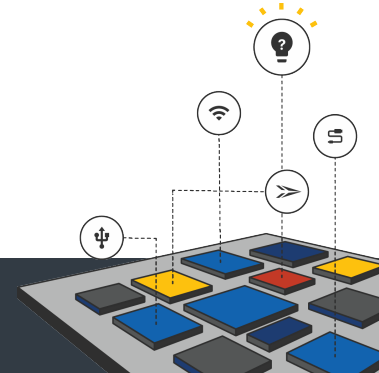
  … more exotic combinations possible

- Correct due to careful operation sequencing rules
  - C:Release > B:Probe > A:Acquire

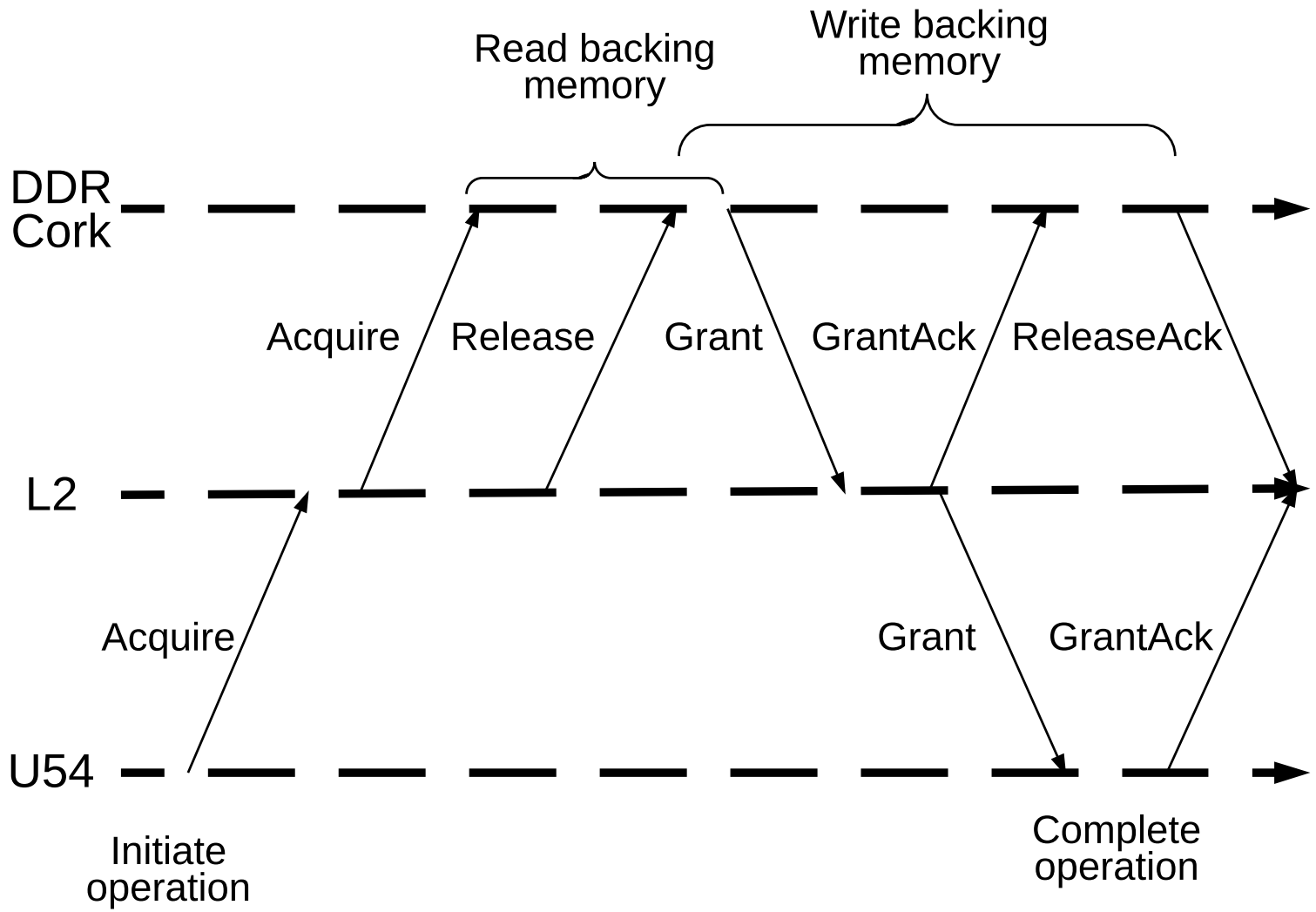


Cache miss in the L2, Recursive Acquire

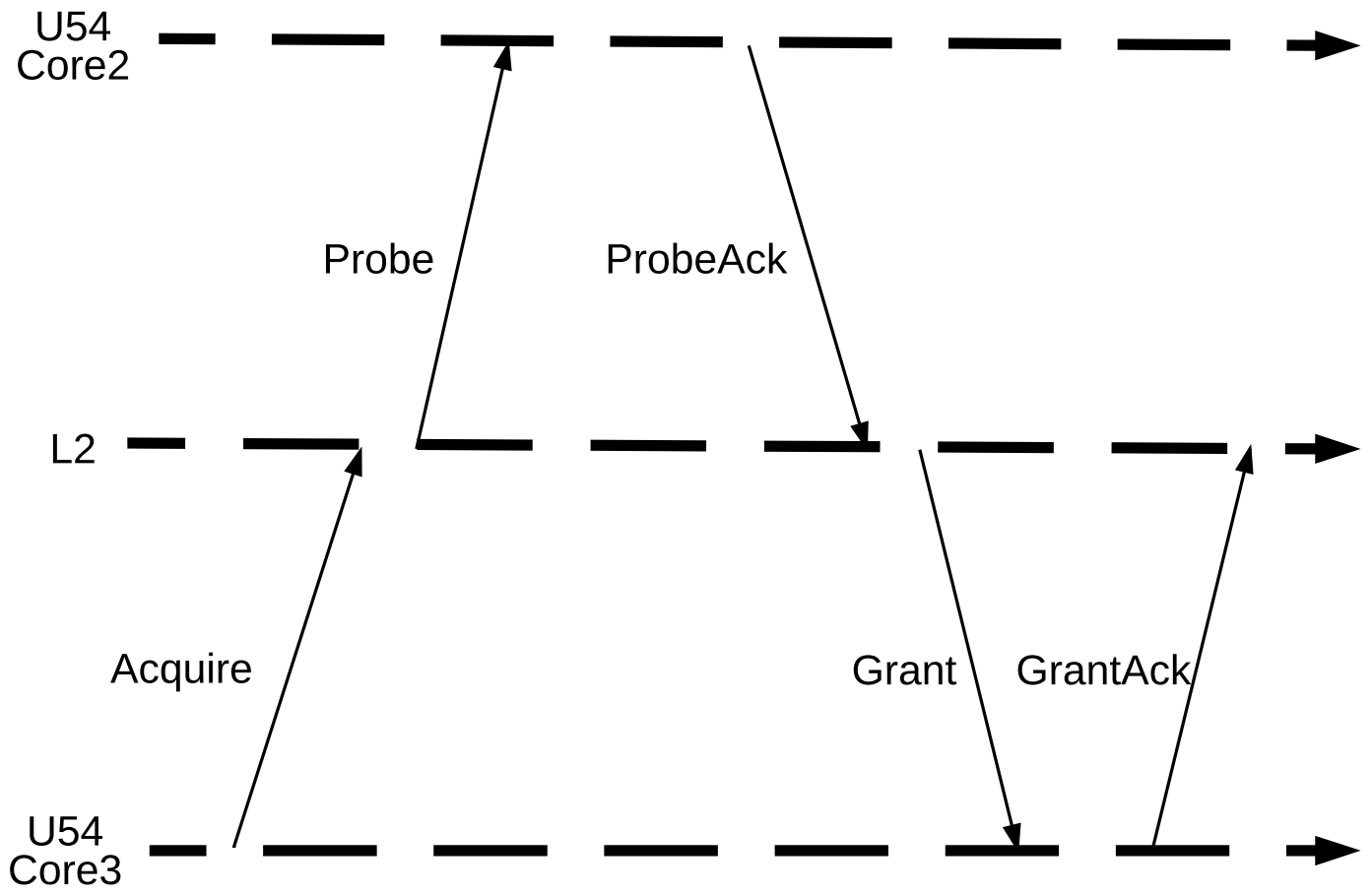




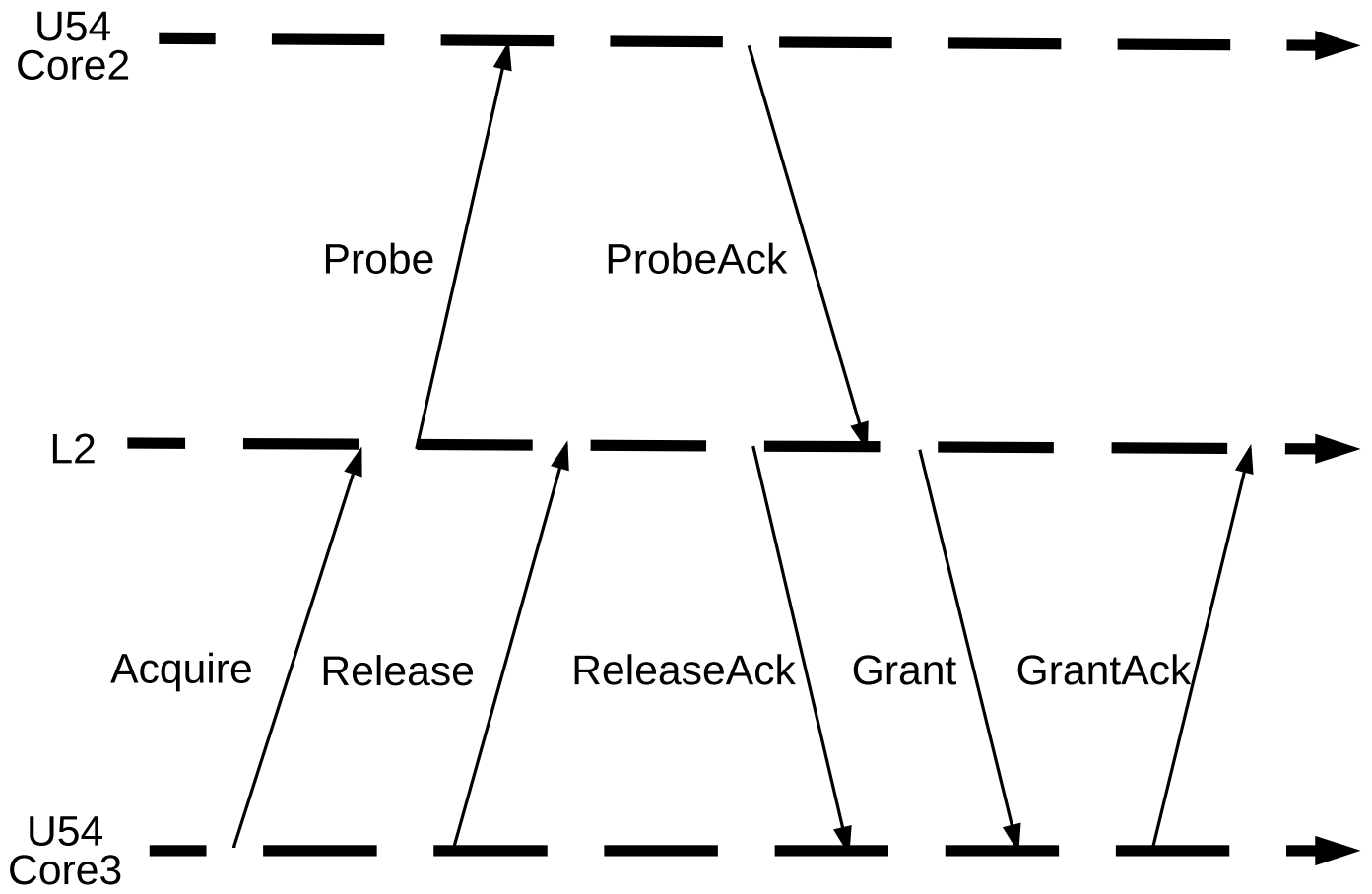
Cache miss in the L2, Recursive Acquire + Release

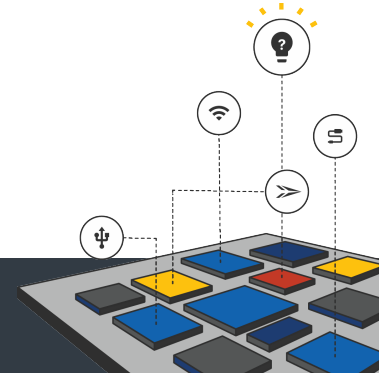


Bounce from L1 to L1, Acquire + Probe

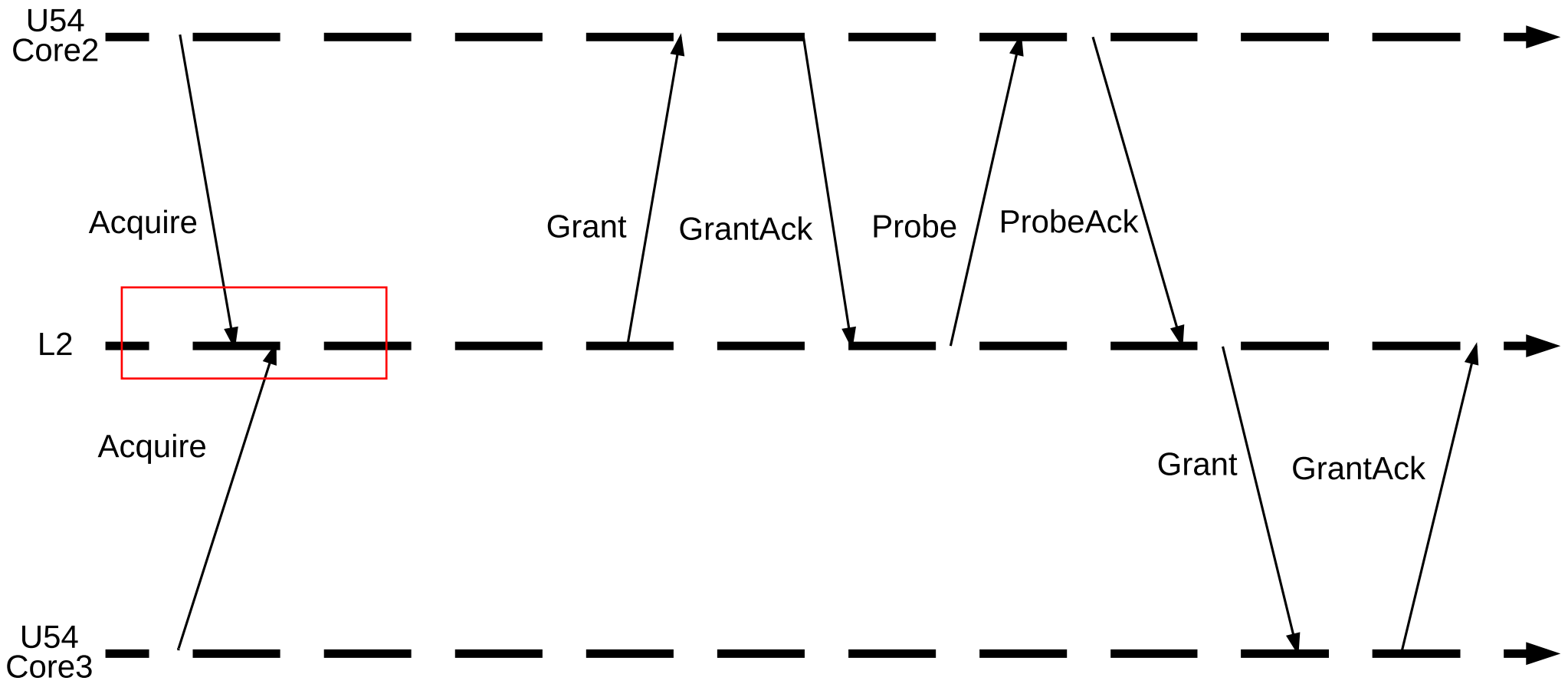


Bounce from L1 to L1, Acquire + Probe + Release

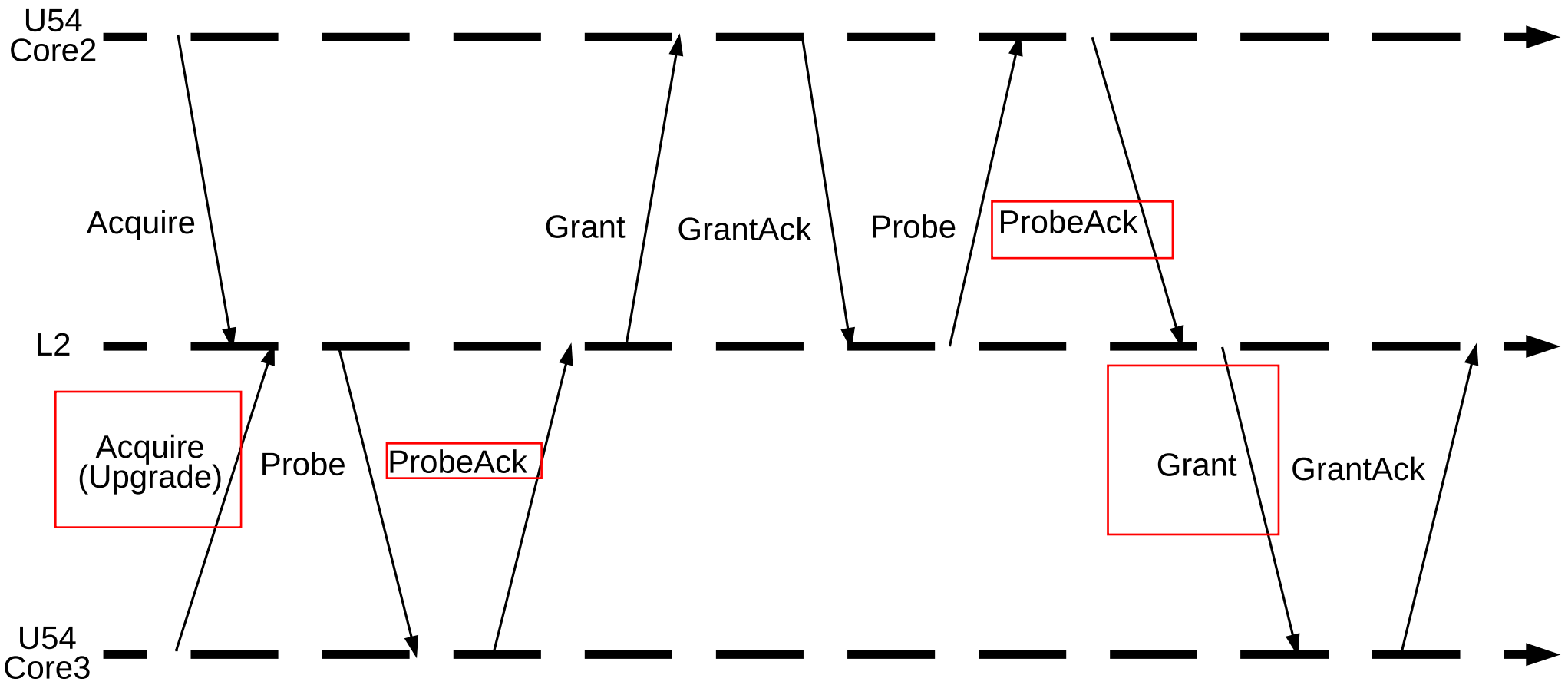




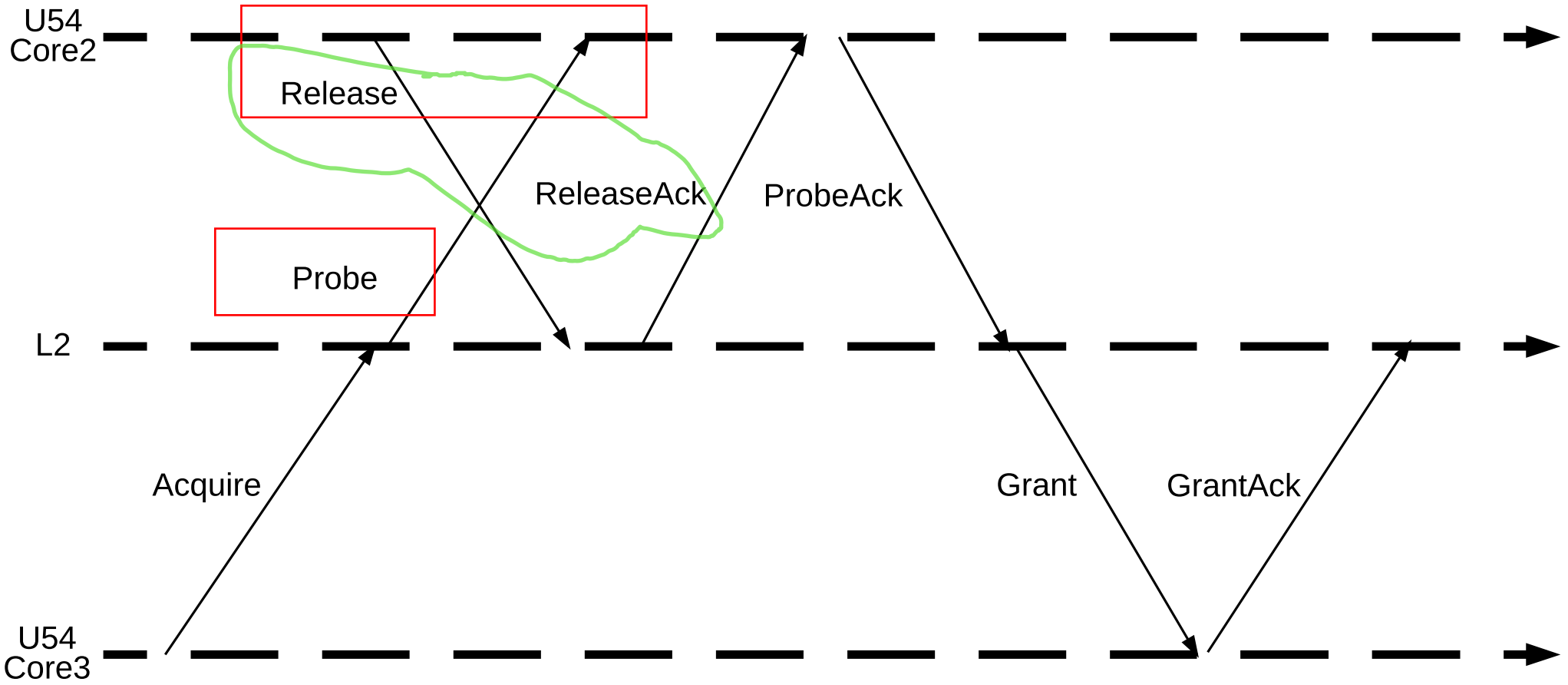
Acquire Race



Acquire Race, Double Probe

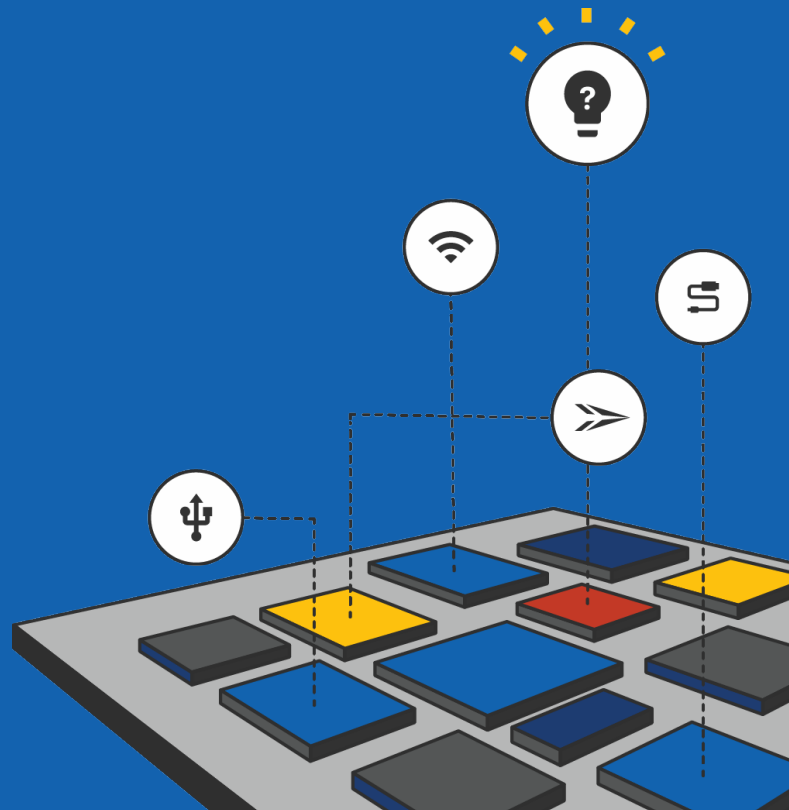


Acquire + Release Race



SiFive

Ordering Rules



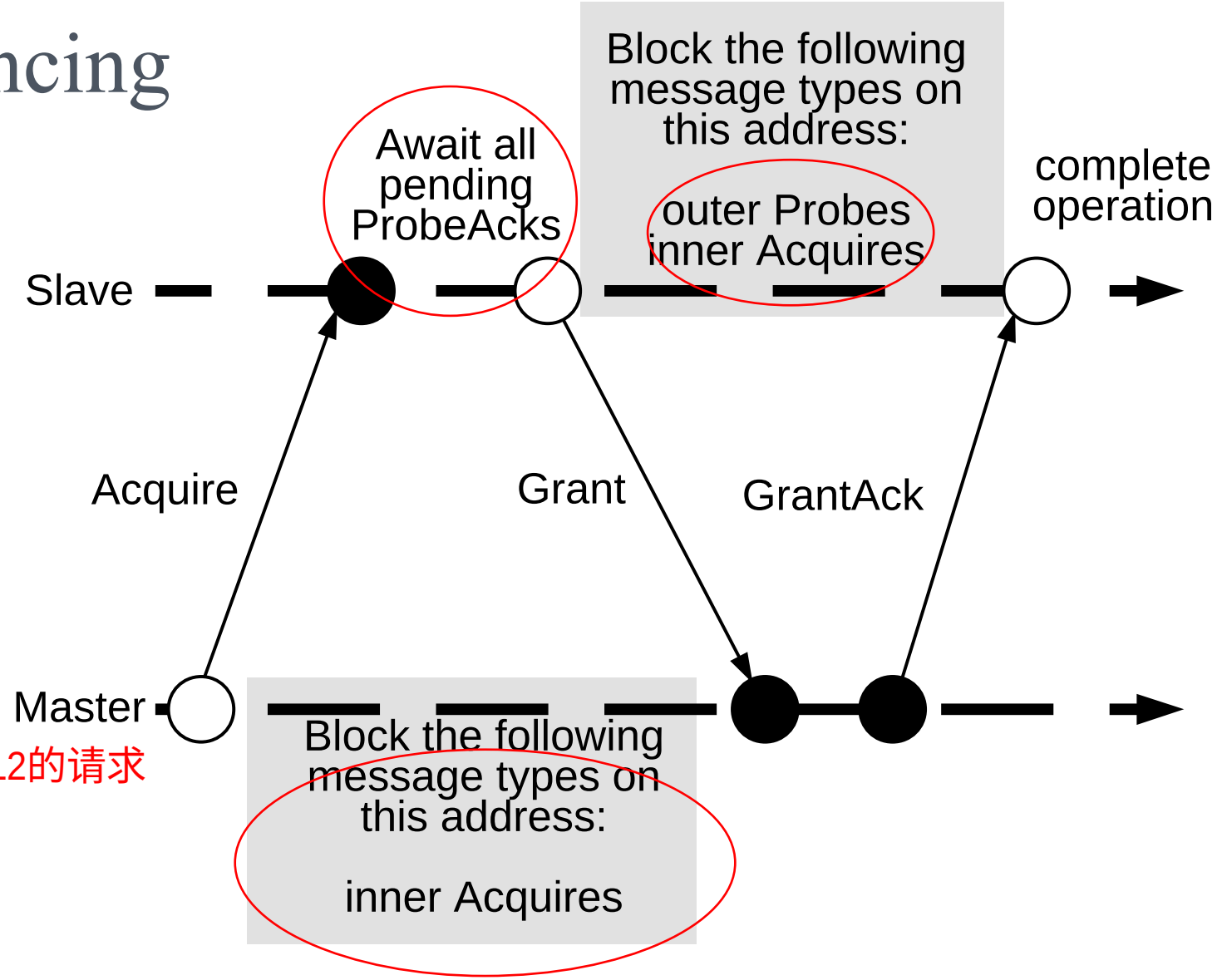
Acquire Sequencing

Acquire is the lowest priority transfer.

Outer probes and inner releases must be nested.

假设有一个缓存系统：
L1、L2、L3

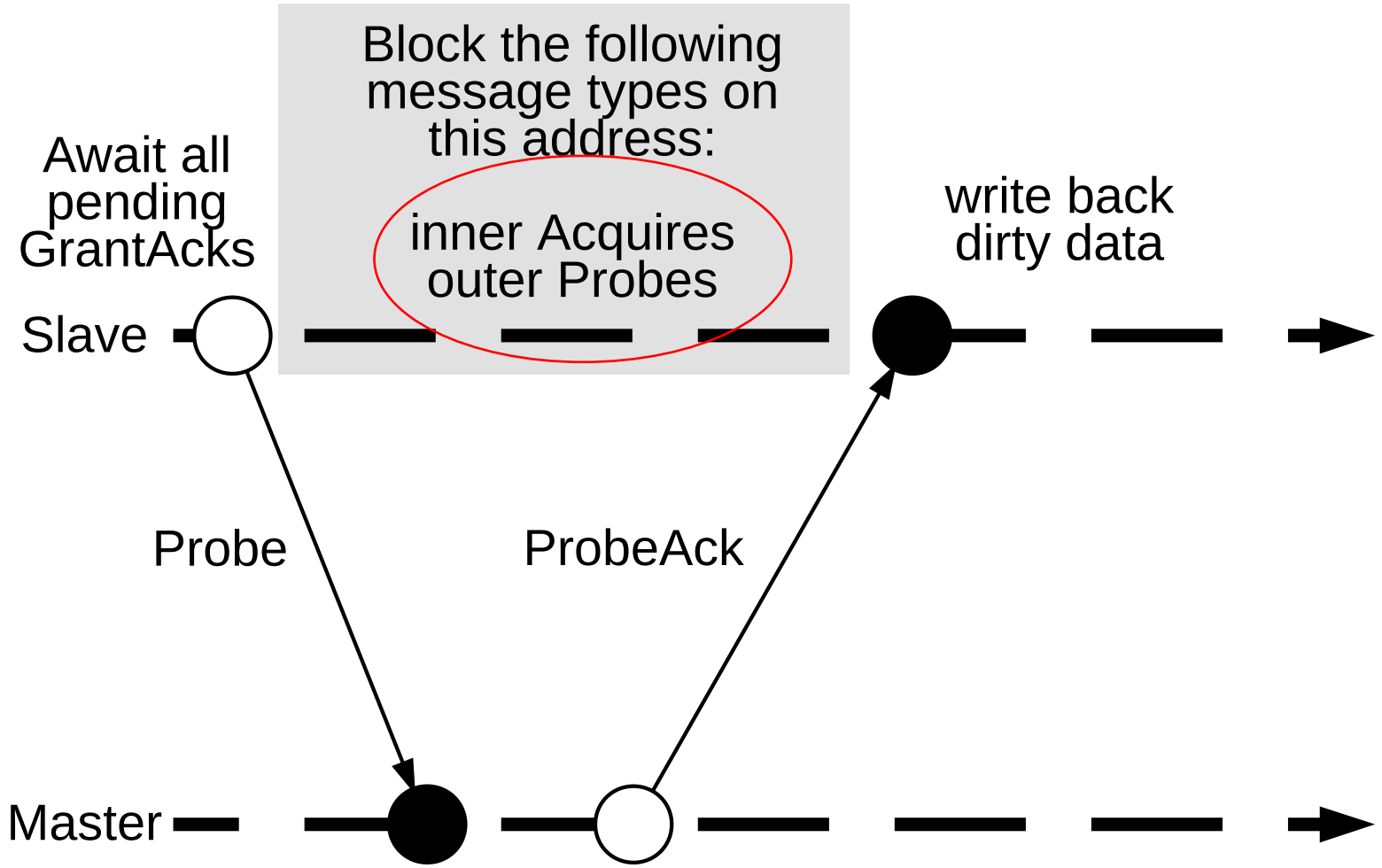
对于L2来说，Outer表示从L3发送到L2的请求
Inner表示从L1发送到L2的请求



Probe Sequencing

Probes are the middle priority transfer.

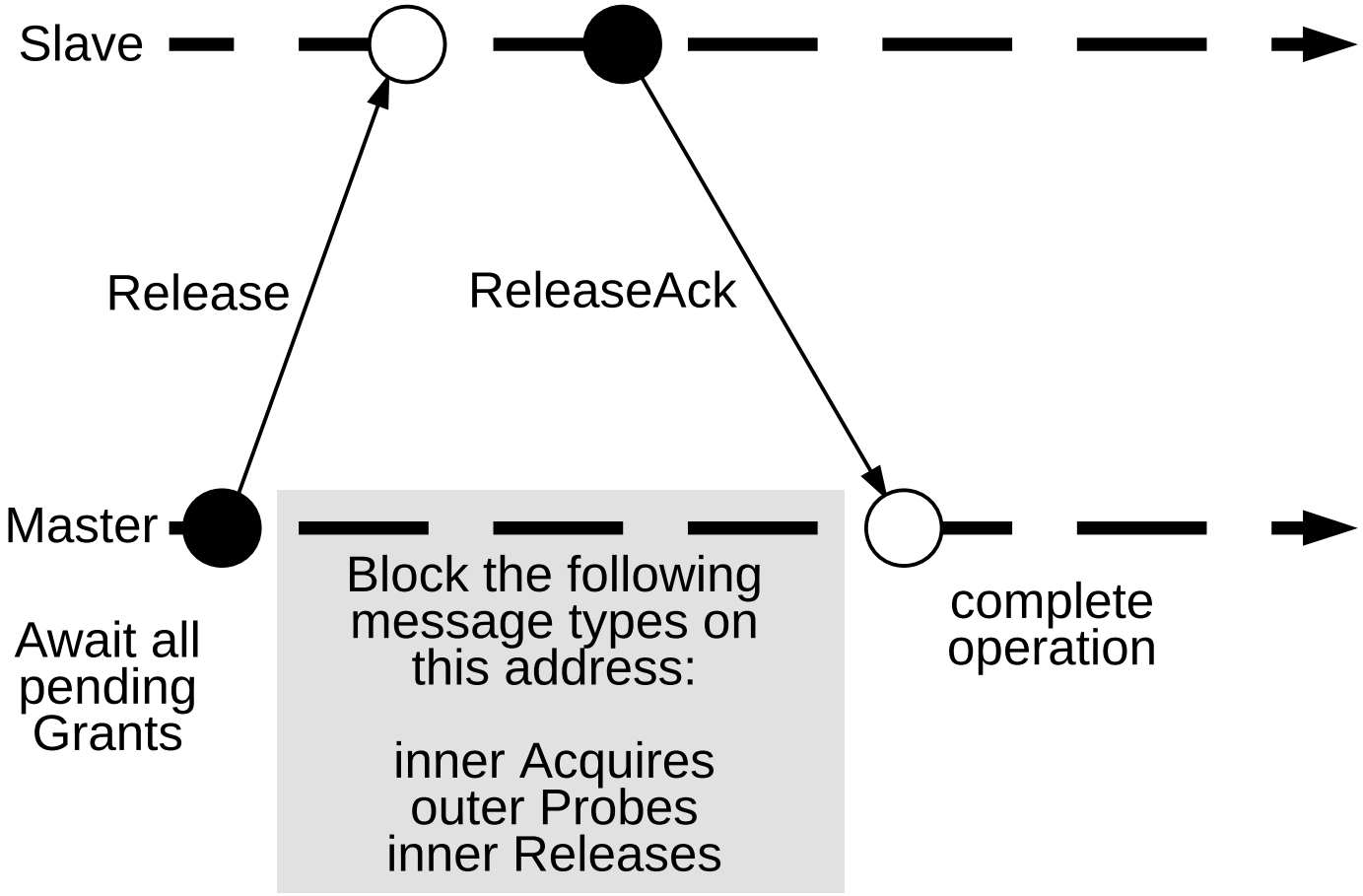
Inner acquires are blocked, while inner releases must be nested.



Release Sequencing

Release is the highest priority transfer.

Outer probes and inner acquires must be blocked.



Rules for Forward Progress

1. The TileLink network is a directed acyclic graph
2. Bounded delays are ok (DDR refresh/etc)
3. Messages are atomic
   - Deliver all beats of a message before starting another on the same channel
   - Sending one beat means you must be able to eventually send all
4. Channels have strictly increasing message priority
   - Sending (or sent) unanswered priority C request? lowering A or B ready is ok
     - ➢ combinationally forwarding A valid to B-E valid is ok, but B-E to A is not
5. You must eventually send a response to every request