Coherence Deep Dive for CXL

Rob Blankenship – Intel Corporation and CXL Protocol Working Group co-chair

August 2022

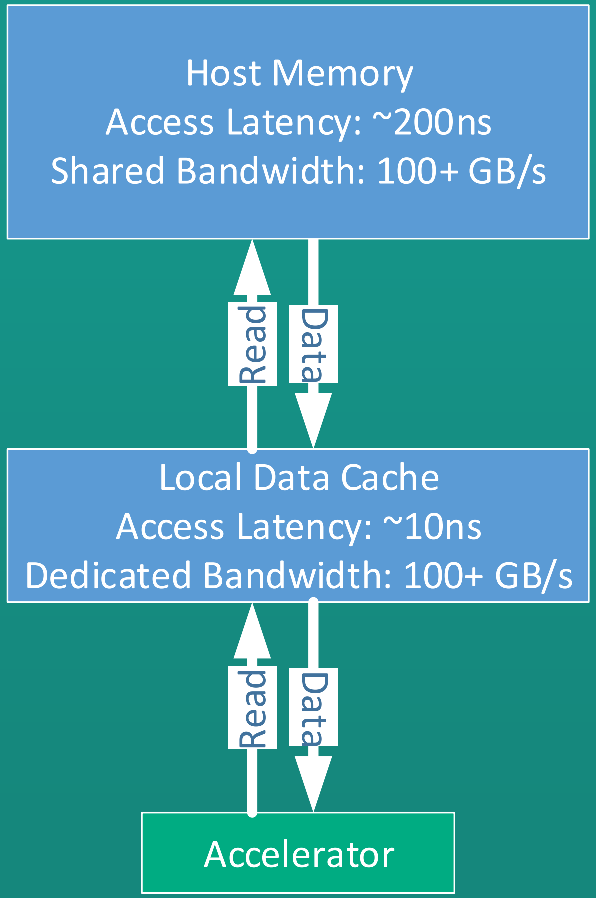
Agenda

- Coherence/Caching Primer
  - CXL Cache Hierarchy
- CXL.Cache Deep Dive
  - What is new in CXL3 (Device Scaling)
- CXL.Mem Deep Dive
  - What is new in CXL3
  - Direct P2P to HDM/Multi-Host Coherence

Caching Primer

Caching Overview

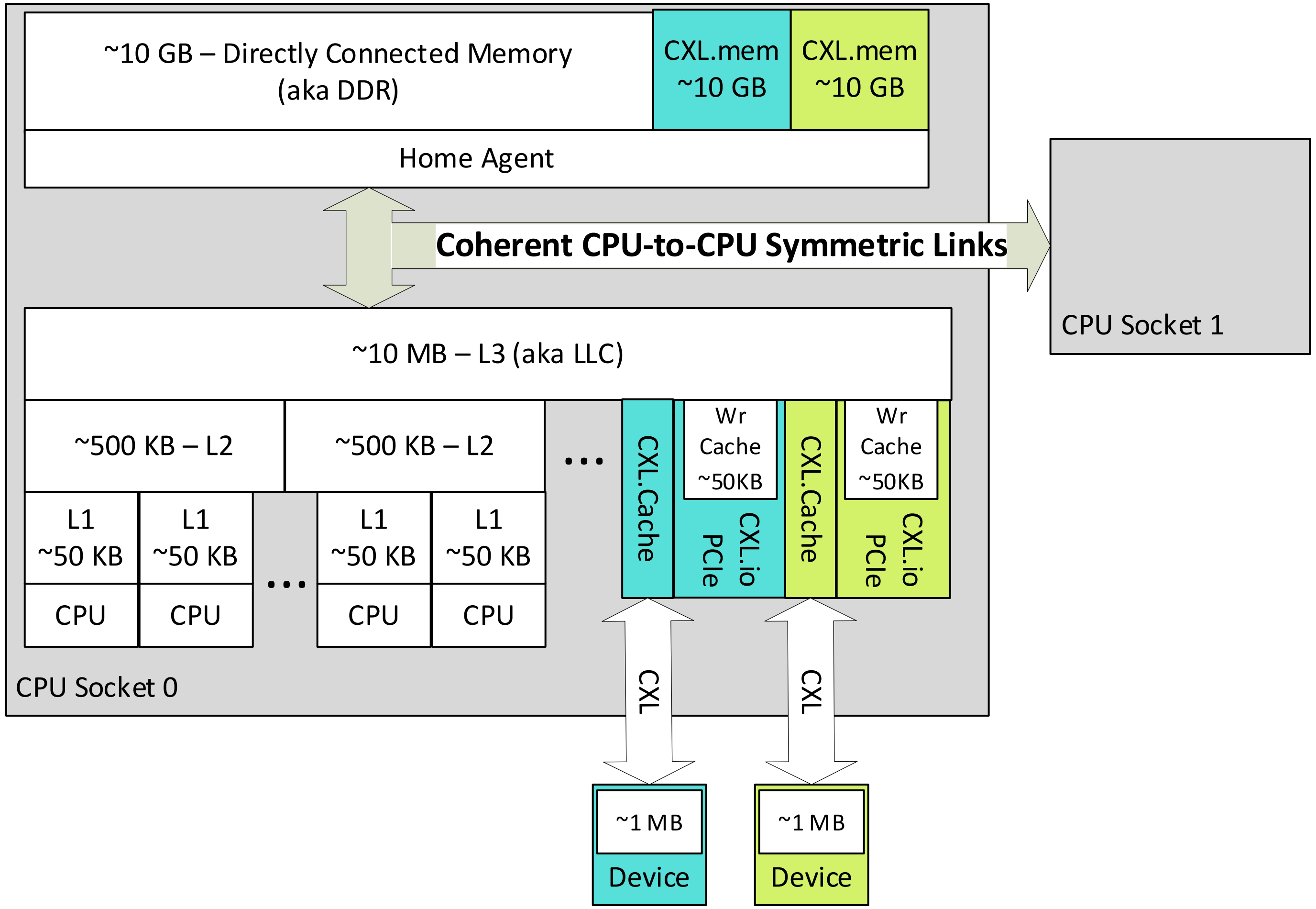
- Caching temporarily brings data closer to the consumer
- Improves latency and bandwidth using prefetching and/or locality
  - Prefetching: Loading Data into cache before it is required
  - Spatial Locality (locality is space): Access address X then X+n
  - Temporal Locality (locality in Time): Multiple access to the same Data


Host Memory
Access Latency: ~200ns
Shared Bandwidth: 100+ GB/s

Local Data Cache
Access Latency: ~10ns
Dedicated Bandwidth: 100+ GB/s

Accelerator

CPU Cache/Memory Hierarchy with CXL



- Modern CPUs have 2 or more levels of coherent cache
- Lower levels (L1), smaller in capacity with lowest latency and highest bandwidth per source.
- Higher levels (L3), less bandwidth per source but much higher capacity and support more sources
- Device caches are expected to be up to 1MB.

Note: Cache/Memory capacities are examples and not aligned to a specific product.

Cache Consistency

How do we make sure updates in cache are visible to other agents?

- Invalidate all peer caches prior to update
- Can managed with software or hardware → CXL uses hardware coherence

Define a point of "Global Observation" (aka GO) when new data is visible from writes

Tracking granularity is a "cacheline" of data → 64-bytes for CXL

All addresses are assumed to be Host Physical Address (HPA) in CXL cache and memory protocols → Translations done using Address Translation Services (ATS).

Cache Coherence Protocol

- Modern CPU caches and CXL are built on M,E,S,I protocol/states
  - `Modified` – Only in one cache, Can be read or written, Data **NOT** up-to-date in memory
  - `Exclusive` – Only in one cache, Can be read or written, Data **IS** up-to-date in memory
  - `Shared` – Can be in many caches, Can only be read, Data IS up-to-date in memory
  - `Invalid` – Not in cache

- M,E,S,I is tracked for each cacheline address in each cache
  - Cacheline address in CXL is Addr[51:6]

- Notes:
  - Each level of the CPU cache hierarchy follows MESI and layers above must be consistent
  - Other extended states and flows are possible but not covered in context of CXL

How are Peer Caches Managed?

- All peer caches managed by the “Home Agent” within the cache level.
- A “Snoop” is the term for the Home to check cache state and causing cache state changes.
- Example CXL Snoops:
  - Snoop Invalidate (SnpInv): Causes a cache to degrade to I-state, and must return any Modified data.
  - Snoop Current (SnpCurr): Does not change cache state, but does return indication of current state and any modified data.

CXL Cache Protocol

Cache Protocol Summary

Simple set of 15 reads and writes from the device to host memory

Keep the complexity of global coherence management in the host.

CXL3 enables up to 16 cache devices below each root port

- Prior generations limited to 1 per root port.

Cache Protocol Channels

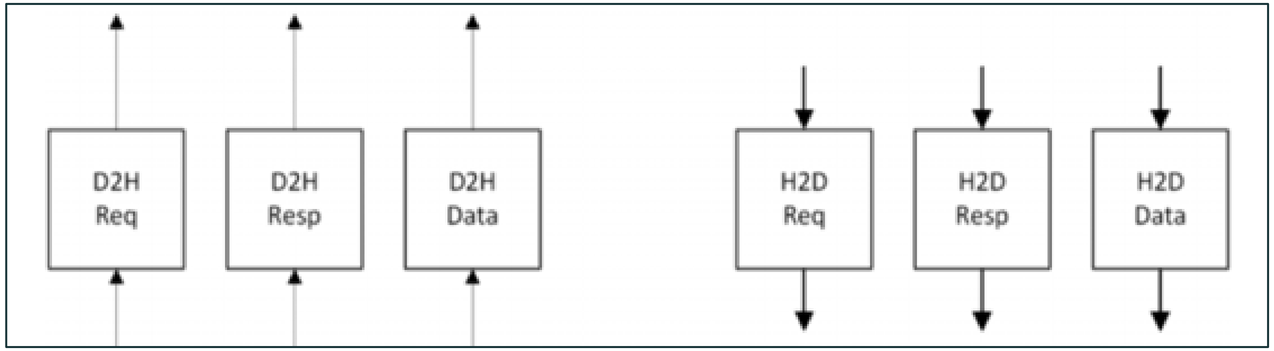
3 channels in each direction: D2H vs H2D

Data and RSP channels are pre-allocated

D2H Requests from the device

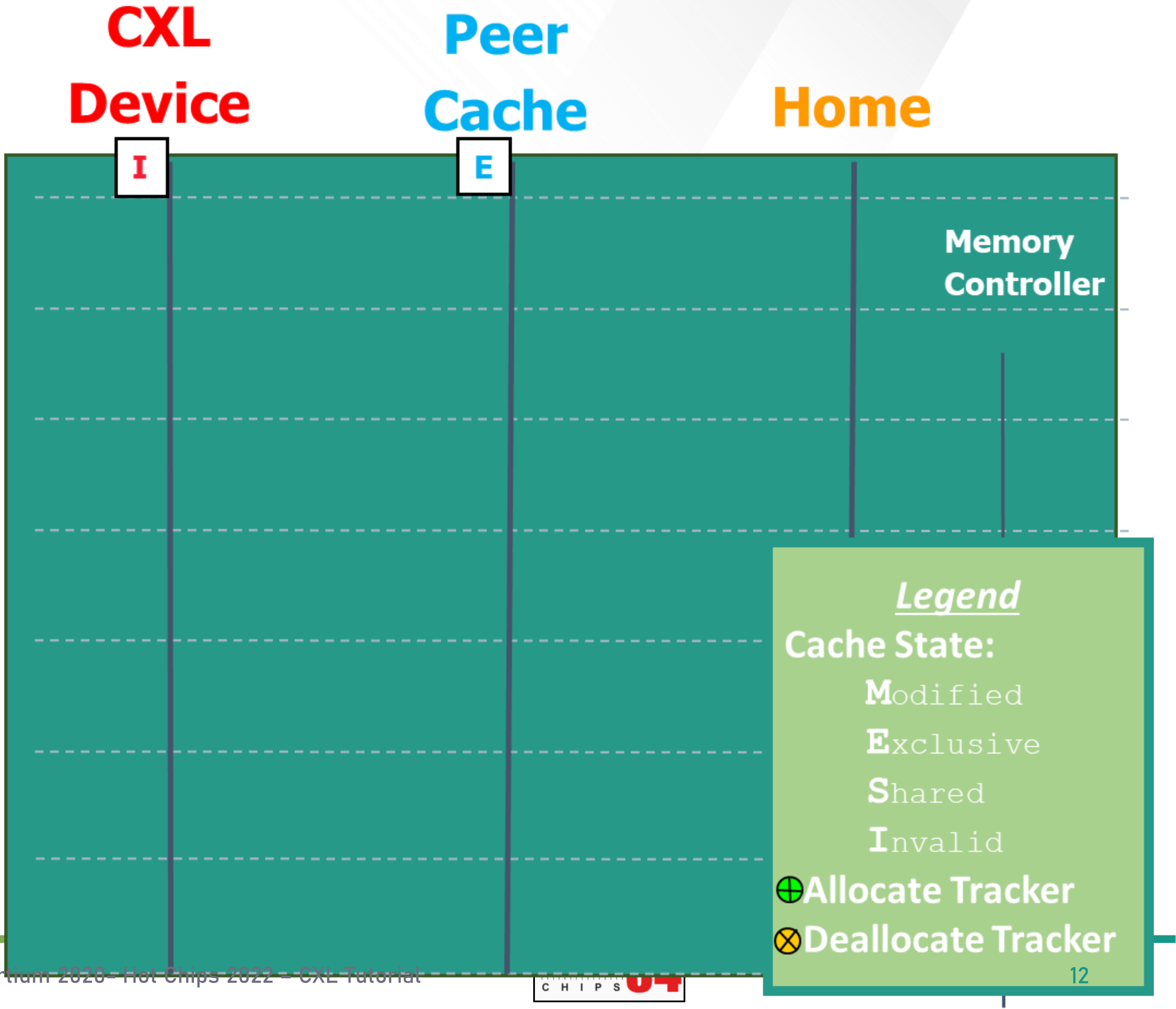
H2D Requests are snoops from the host

Ordering: H2D Req (Snoop) push H2D RSP



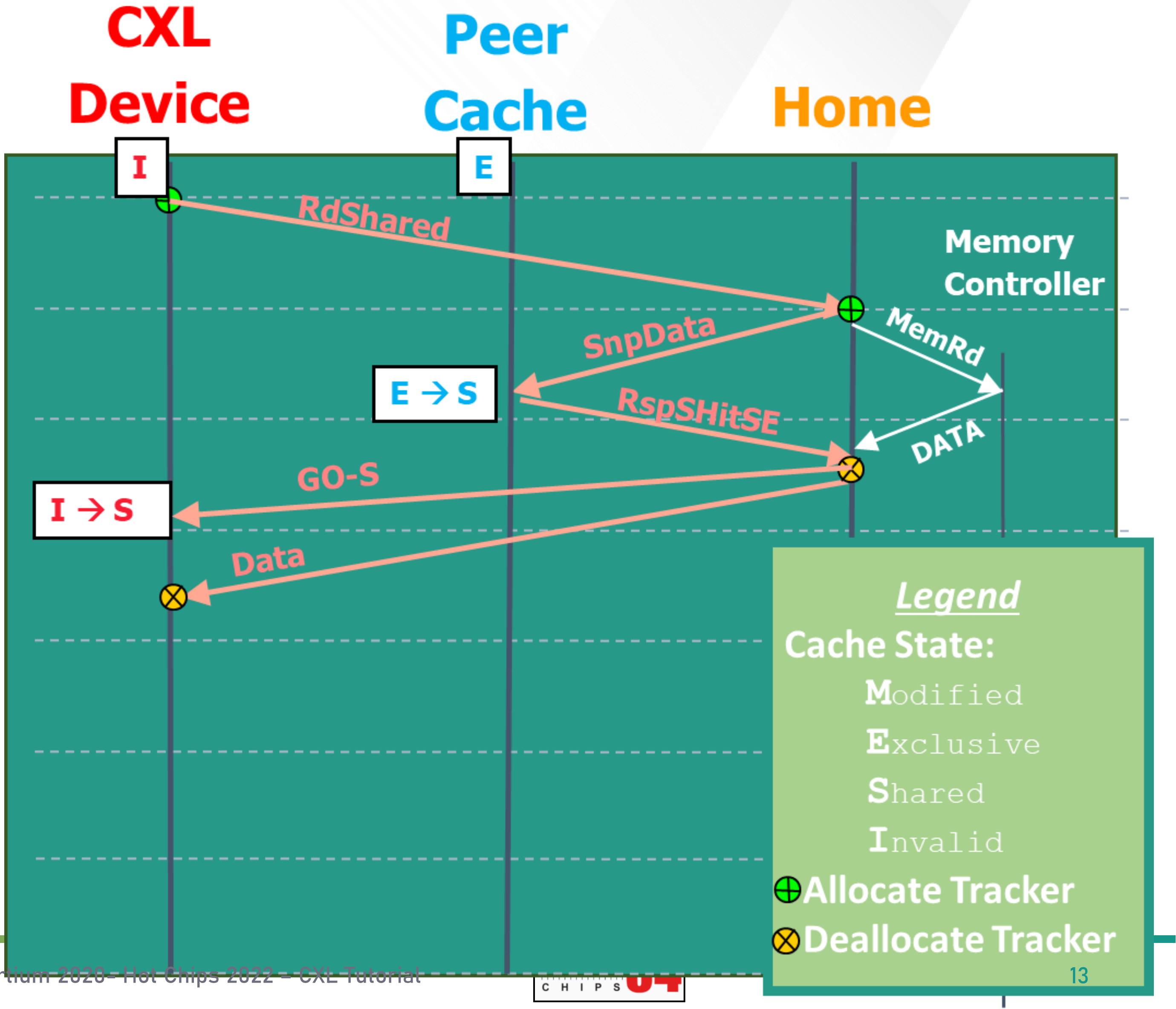
Read Flow

- Diagram to show message flows in time
  - X-axis: Agents
  - Y-axis: Time

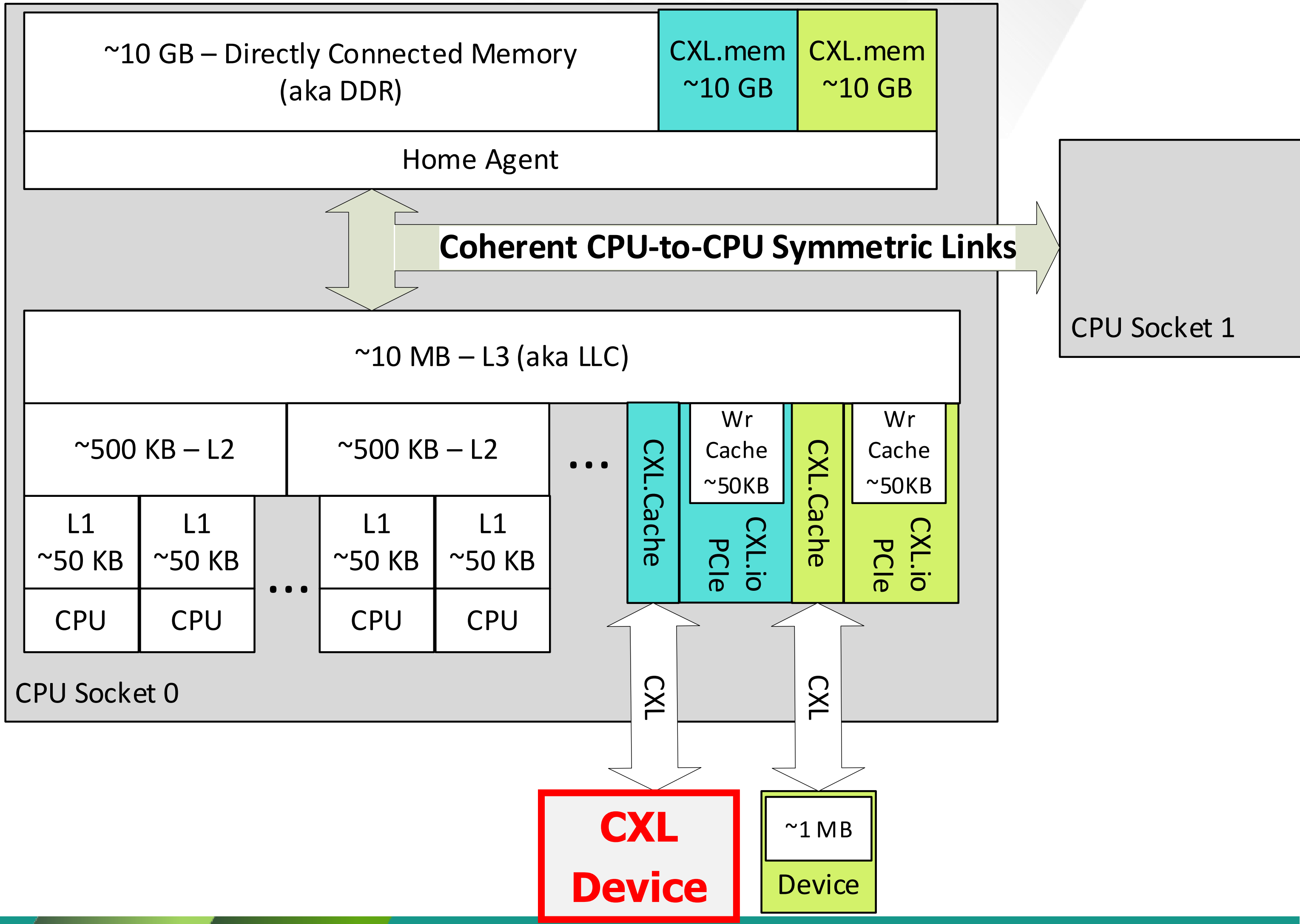


Read Flow

- Diagram to show message flows in time
  - X-axis: Agents
  - Y-axis: Time

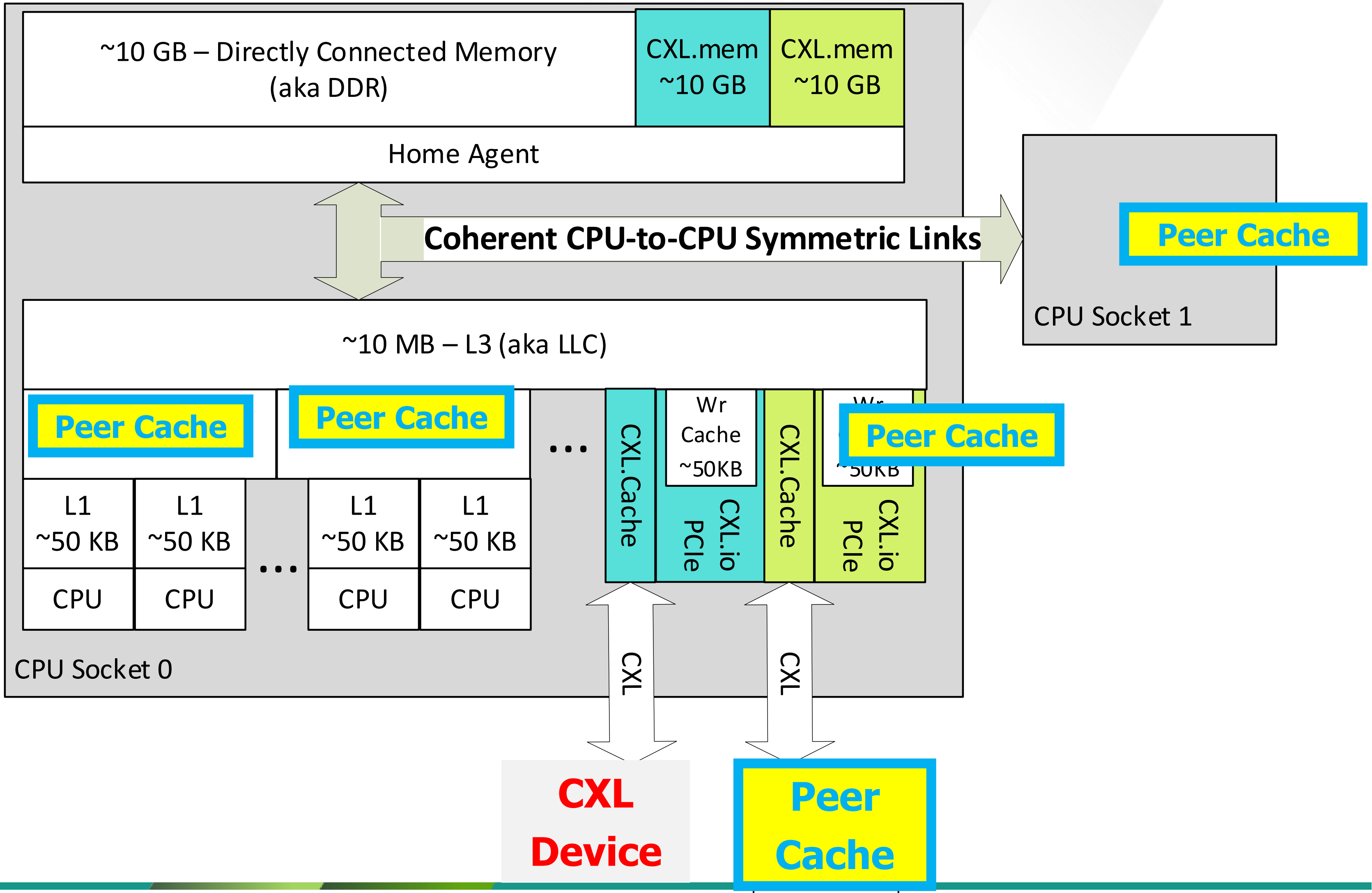


Mapping Flow Back to CPU Hierarchy



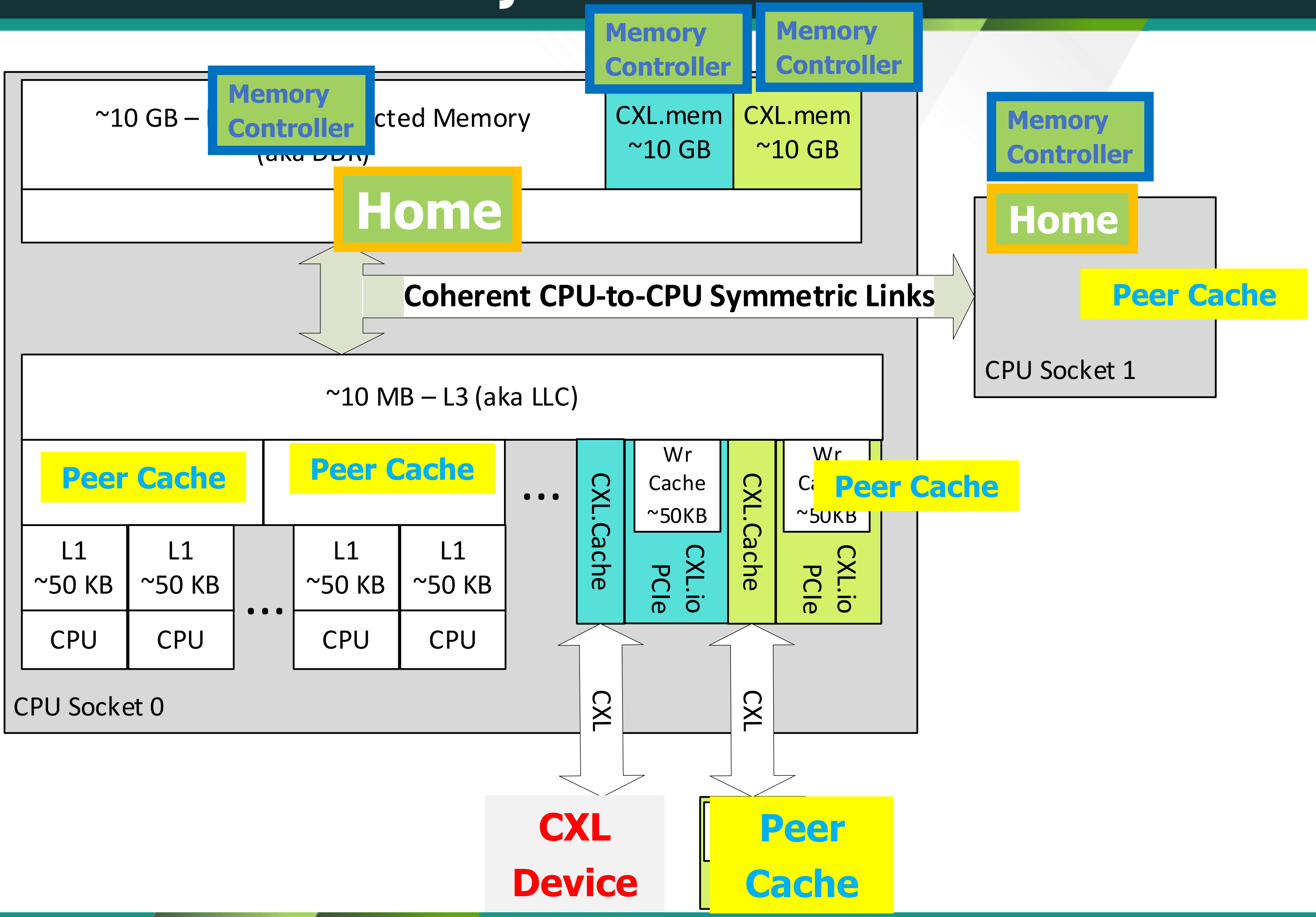
Mapping Flow Back to CPU Hierarchy

- Peer Cache can be:
  - Peer CXL Device with Cache
  - CPU Cache in Local Socket
  - CPU Cache in Remote Socket



Mapping Flow Back to CPU Herarchy

- Peer Cache can be:
  - Peer CXL Device with Cache
  - CPU Cache in Local Socket
  - CPU Cache in Remote Socket
- Memory Controller can be:
  - Native DDR on Local Socket
  - Native DDR on Remote Socket
  - CXL.mem on peer Device



Example #2: Write

- For Cache Writes there are three phases:
  - Ownership
  - Silent Write
  - Cache Eviction

Legend

Cache State:

Modified

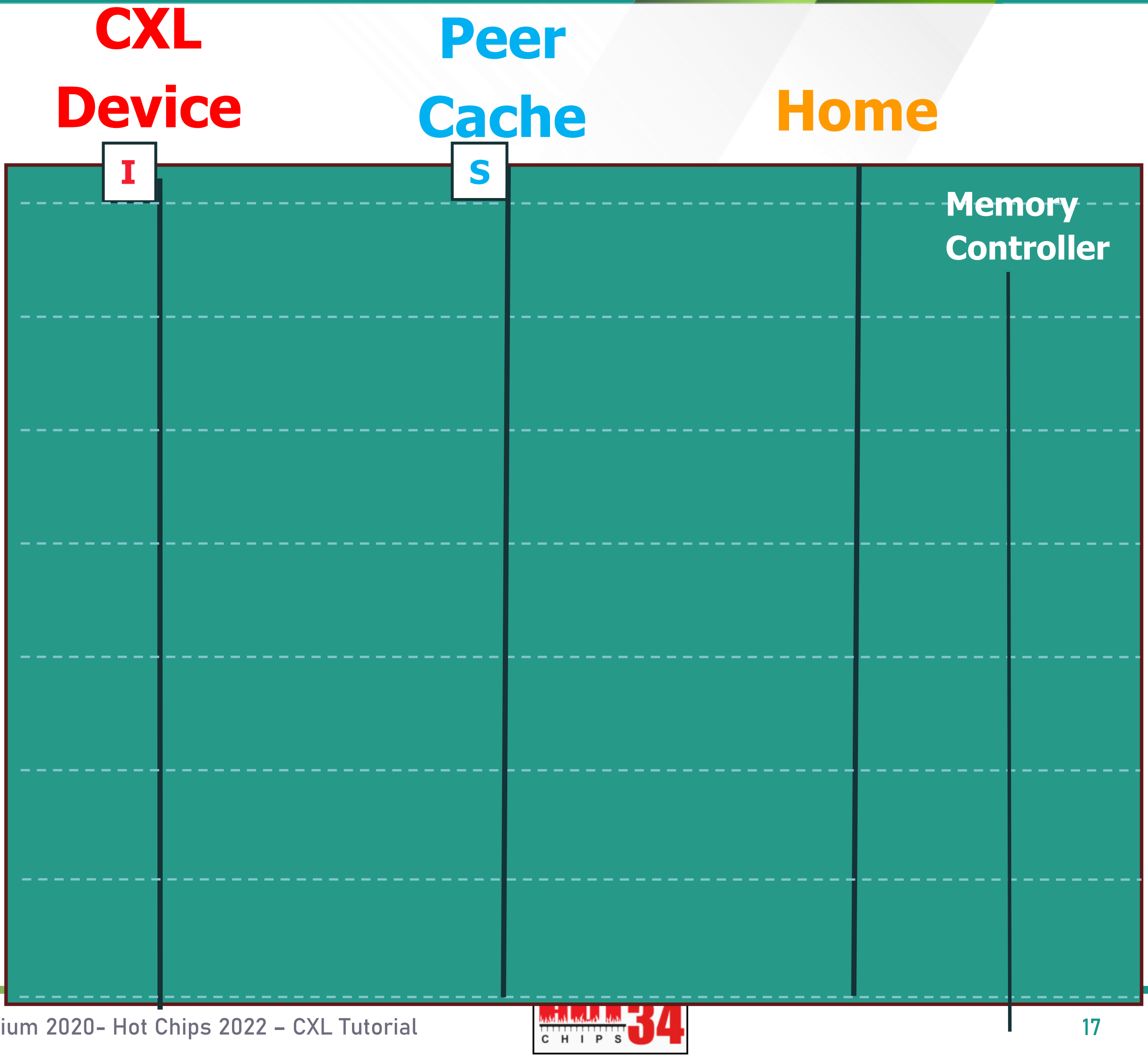
Exclusive

Shared

Invalid

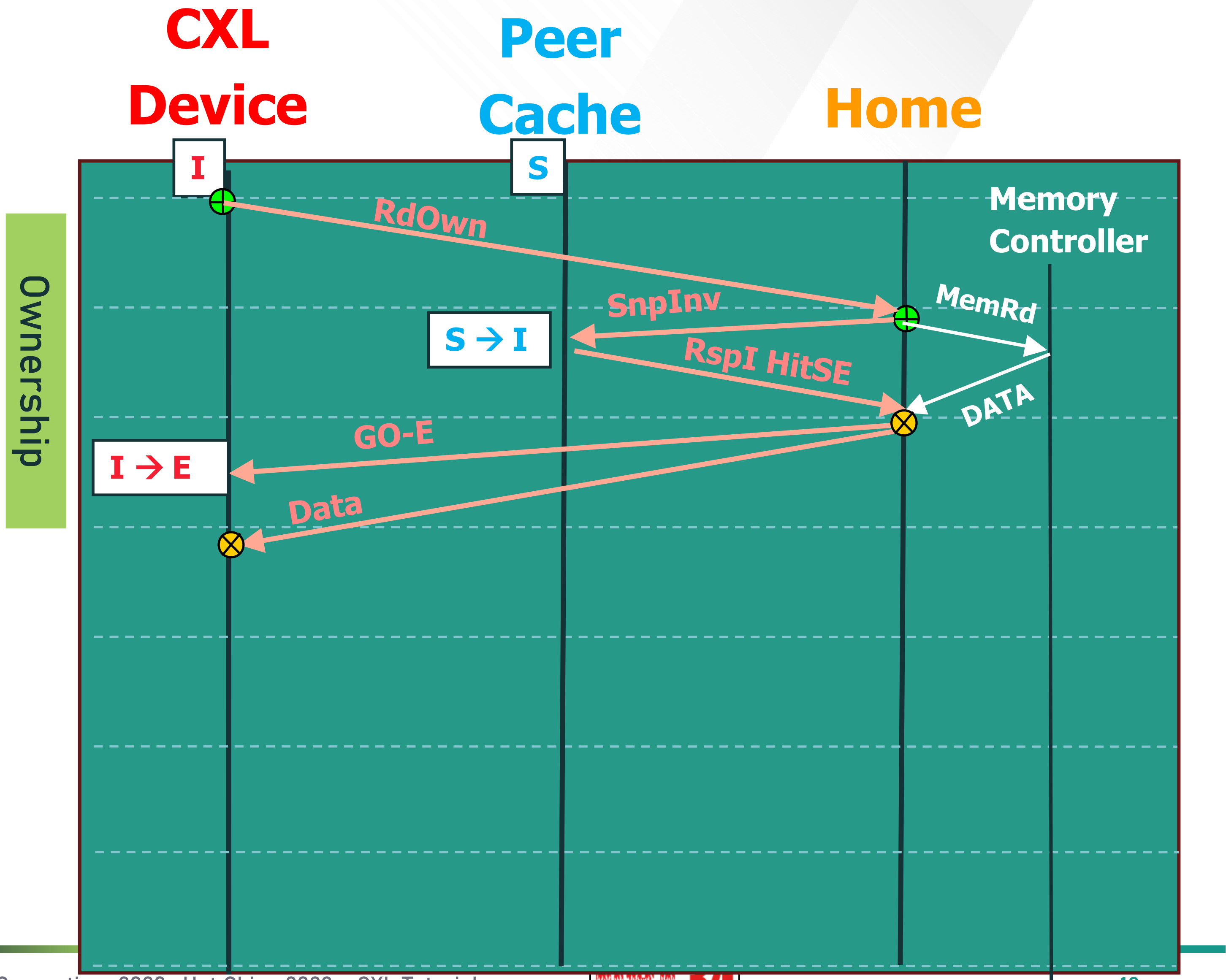
⊕ Allocate Tracker

⊗ Deallocate Tracker



Example #2: Write

- For Cache Writes there are three phases:
  - Ownership
  - Silent Write
  - Cache Eviction



Legend

Cache State:

Modified

Exclusive

Shared

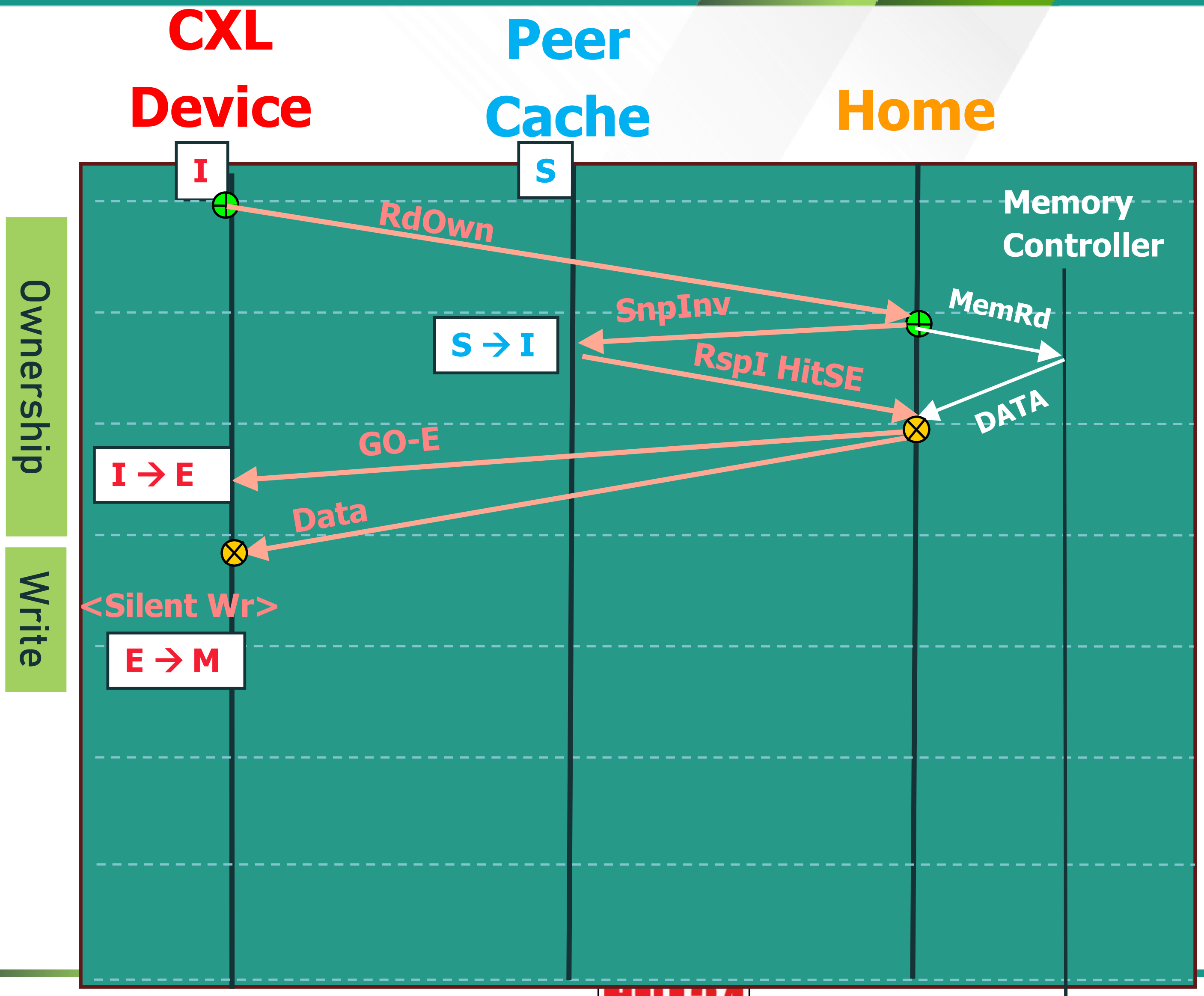
Invalid

⊕ Allocate Tracker

⊗ Deallocate Tracker

Example #2: Write

- For Cache Writes there are three phases:
  - Ownership
  - **Silent Write**
  - Cache Eviction



Legend

Cache State:

- **M**odified
- **E**xclusive
- **S**hared
- **I**nvalid

⊕ Allocate Tracker

⊗ Deallocate Tracker

Example #2: Write

- For Cache Writes there are three phases:
  - Ownership
  - Silent Write
  - Cache Eviction

Legend

Cache State:

Modified

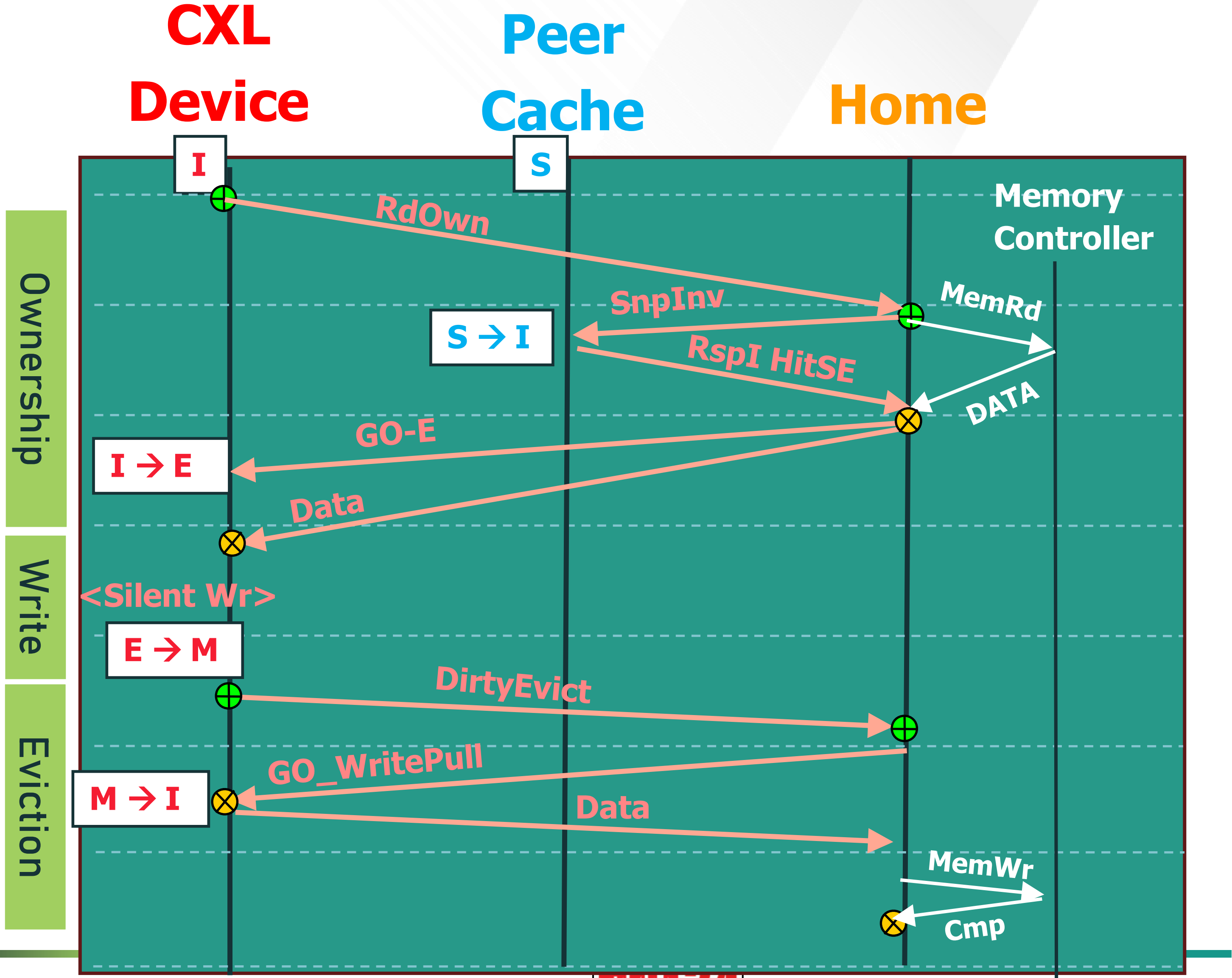
Exclusive

Shared

Invalid

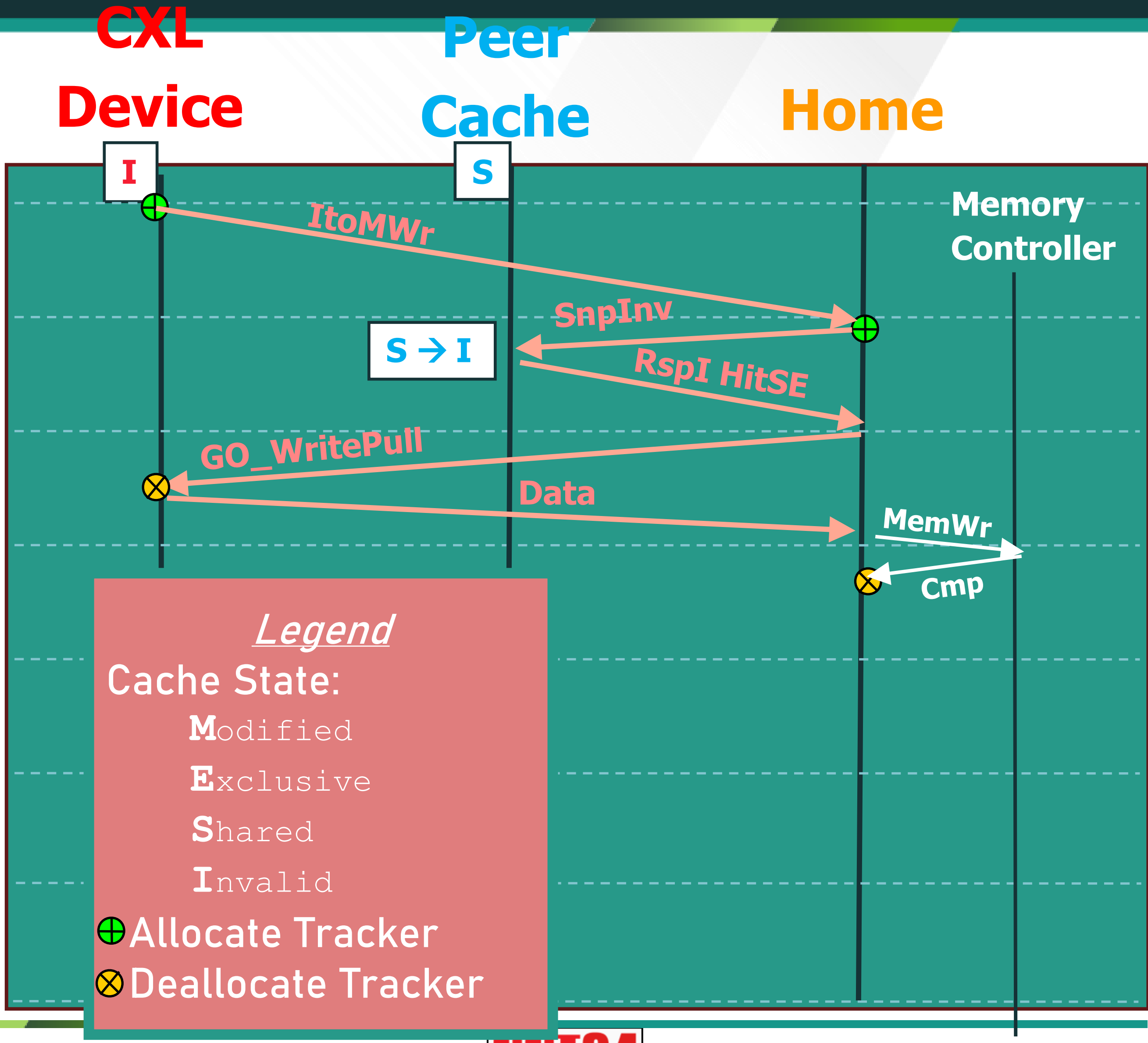
⊕ Allocate Tracker

⊗ Deallocate Tracker



Example #3: Steaming Write

- Direct Write to Host
  - Ownership + Write in a single flow.
- Rely on completion to indicate ordering
  - May see reduced bandwidth for ordered traffic
- Host may install data into LLC instead of writing to memory



15 Request in CXL

- Reads: RdShared, RdCurr, RdOwn, RdAny
- Read-0: RdownNoData, CLFlush, CacheFlushed
- Writes: DirtyEvict, CleanEvict, CleanEvictNoData
- Streaming Writes: ItoMWr, WrCur, WOWrInv, WrInv(F)

CXL Memory Protocol

Memory Protocol Summary

Simple reads and writes from host to memory

Memory Technology Independent

- HBM, DDR, PMem
- Architected hooks to manage persistence

Includes 2-bits of “meta-state” per cacheline

- Memory Only device: Up to host to define usage.
- For Accelerators: Host encodes required cache state.

Host-managed Device Memory (HDM) comes in 3 types:

- Host Managed Coherence (HDM-H)
- Device Managed Coherence (HDM-D)
- Device Managed Coherence with Back-Invalidation (HDM-DB) → new in CXL3

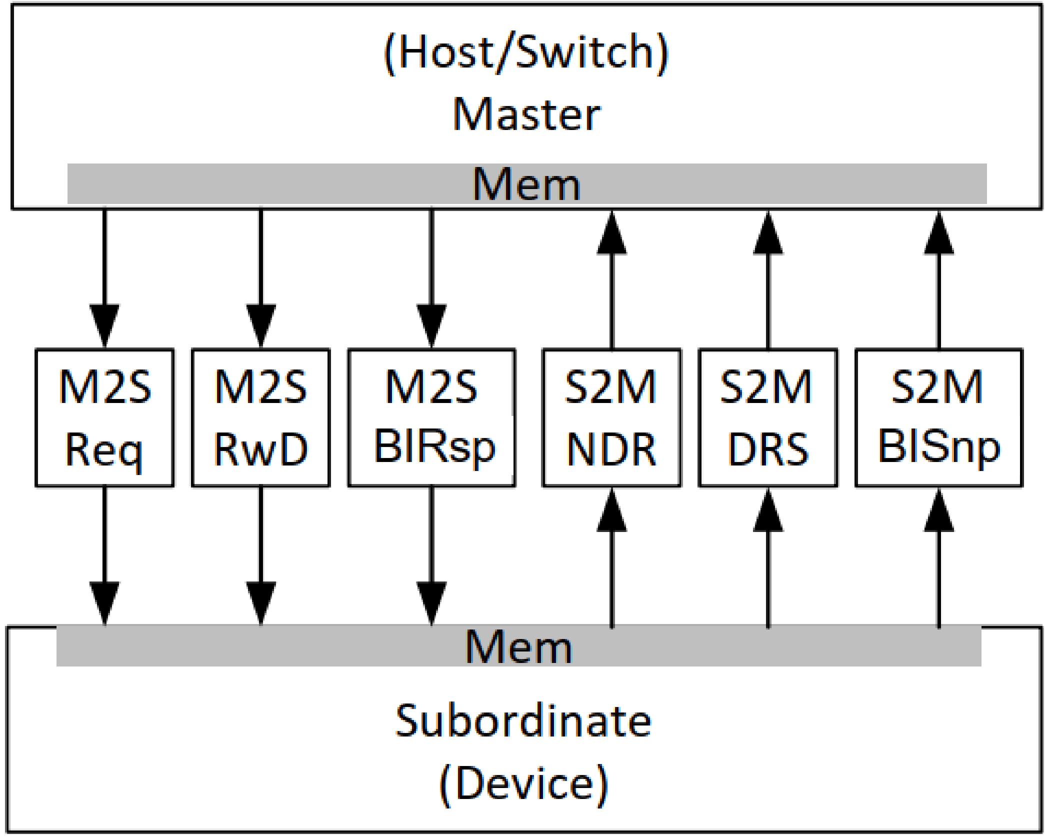
Memory Protocol Channels

3 channels in each direction

- M2S Request (Req), Request w/ Data (RwD)
- S2M Non-Data Response (NDR), Data Response (DRS) which are pre-allocated.
- M2S BIRsp, S2M BISnp used for HDM-DB to manage coherence → New in CXL3.

Limited Ordering

- Req channel for HDM-D memory (CXL2 Accelerators)
- NDR Channel for conflict flows with HDM-DB

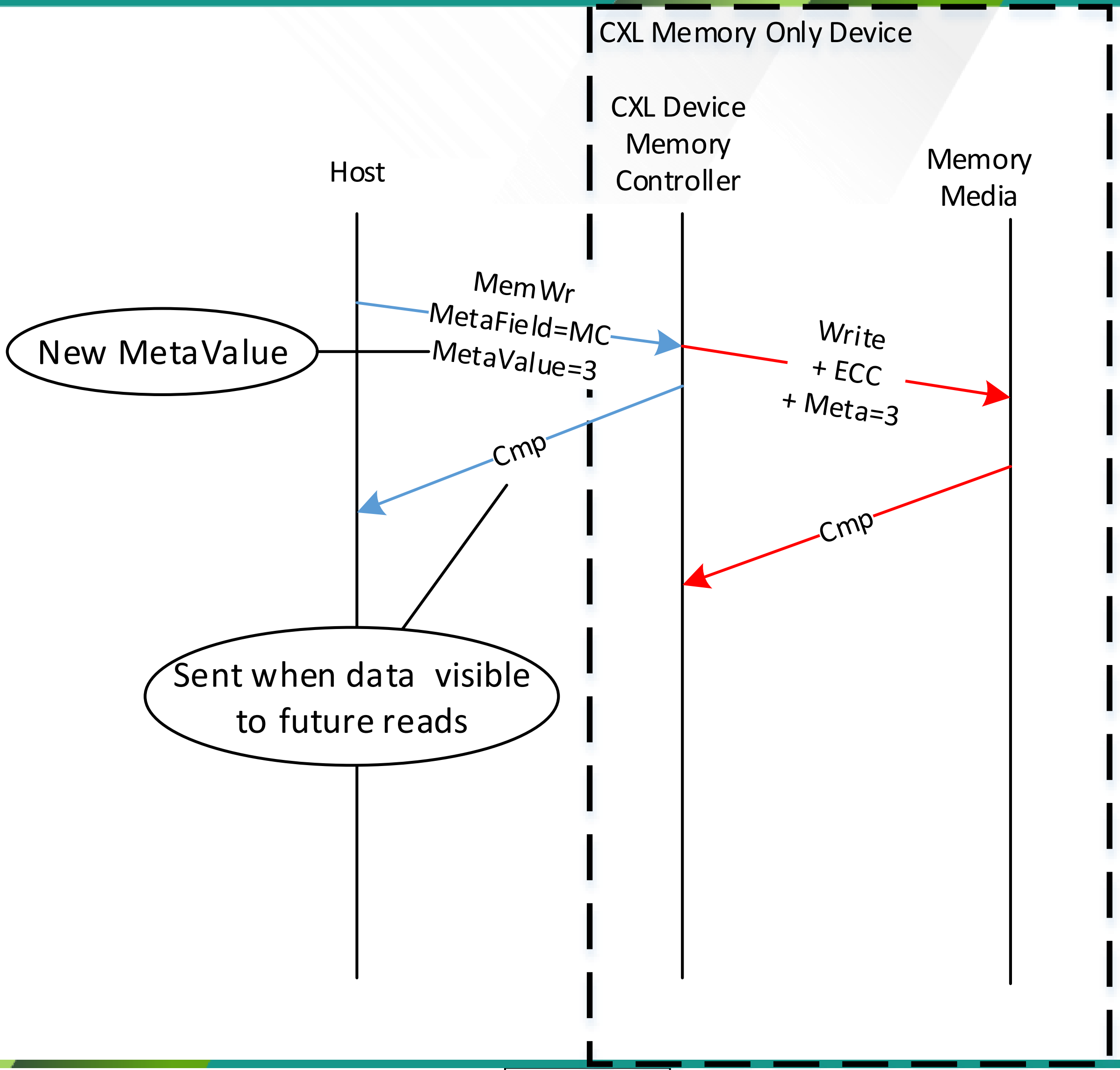


Example #1: Write

Media ECC handled by device

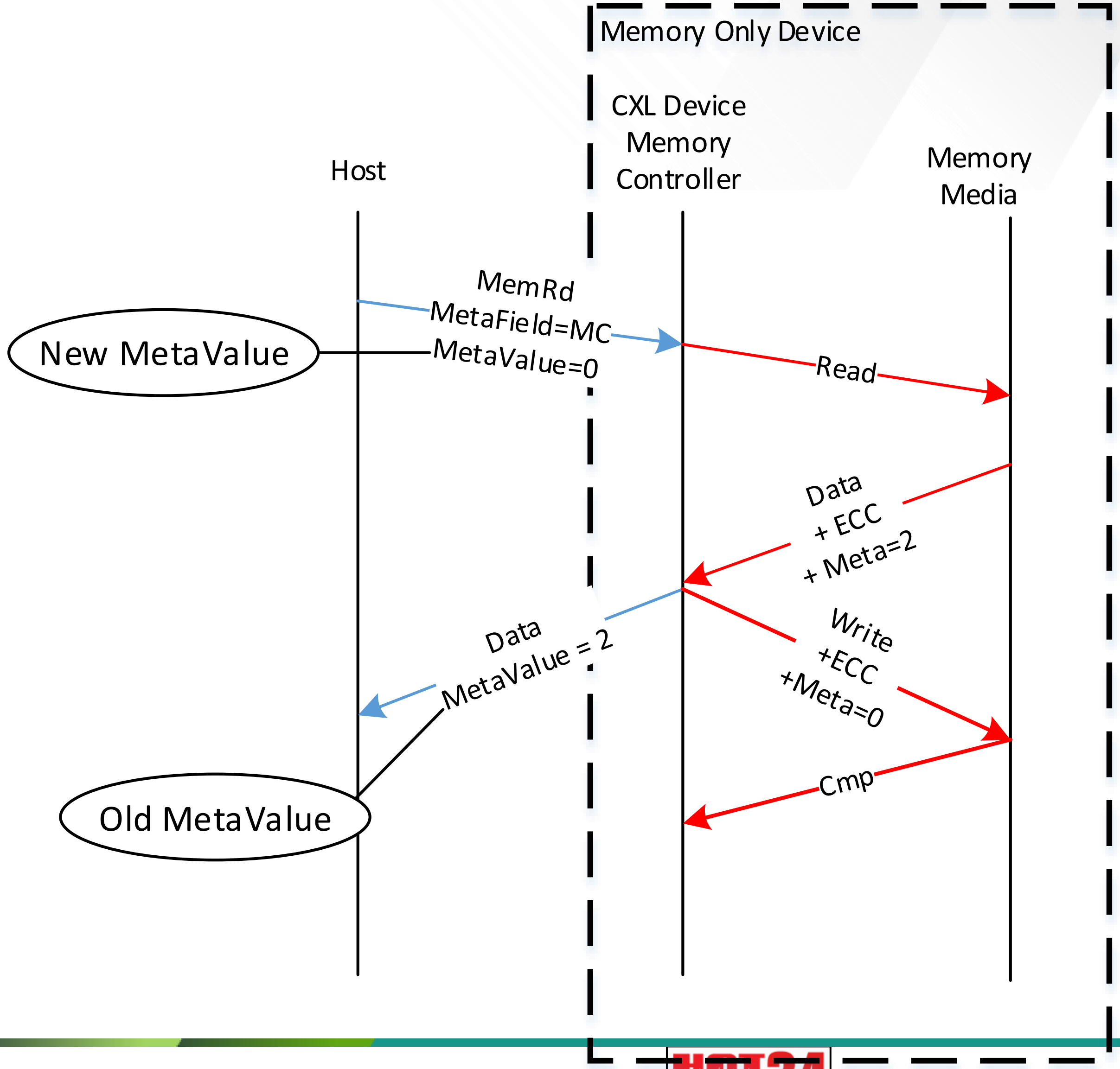
HDM-H provide 2-bits of host defined Meta Value which device optionally supports

Note: only host caching of HDM-H (Host only coherent)


CXL Memory Only Device
Host
CXL Device Memory Controller
Memory Media
MemWr MetaField=MC MetaValue=3
New MetaValue
Write + ECC + Meta=3
Cmp
Cmp
Sent when data visible to future reads

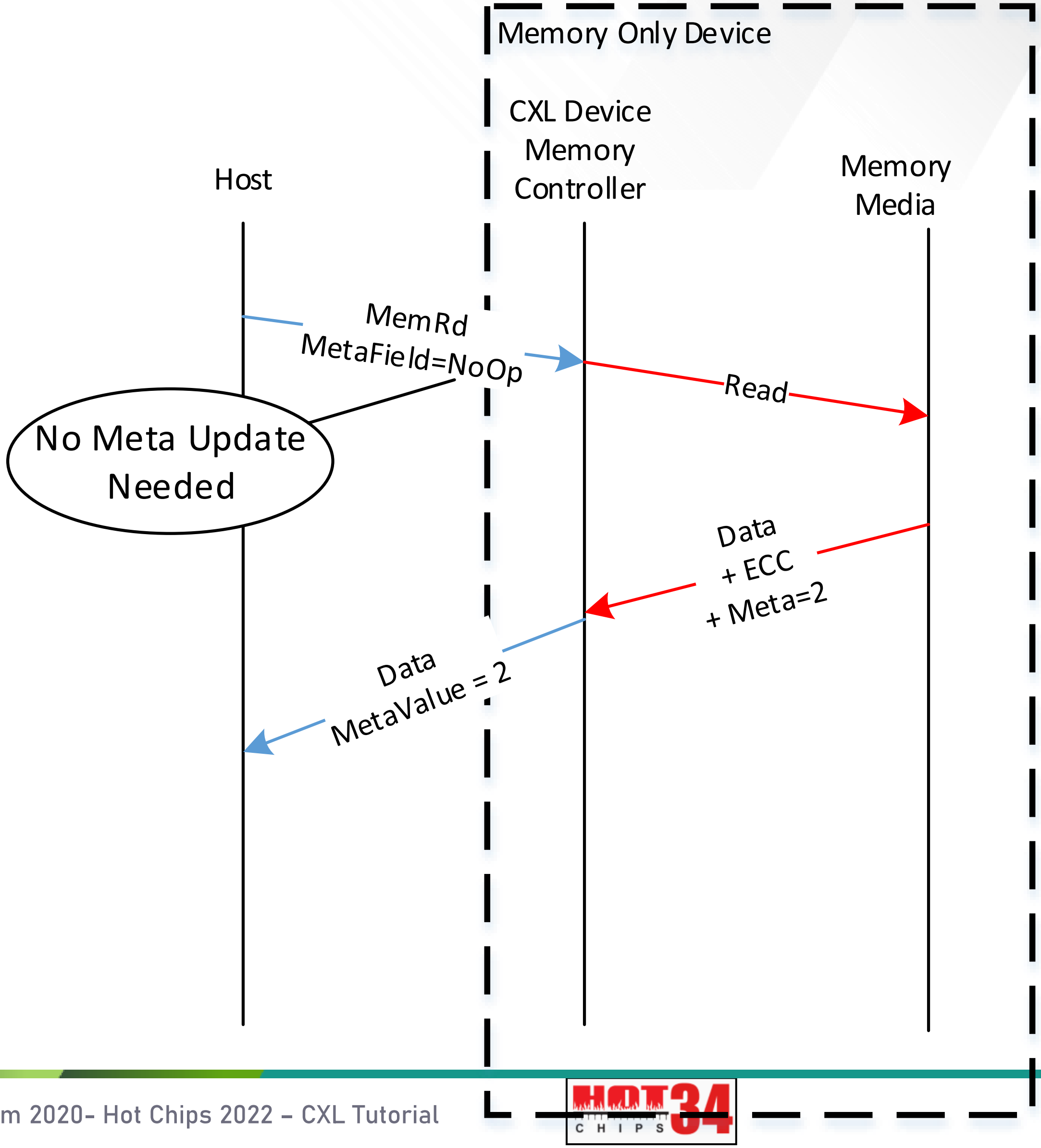
Example #2: Read

Meta Value Change requires device to write.



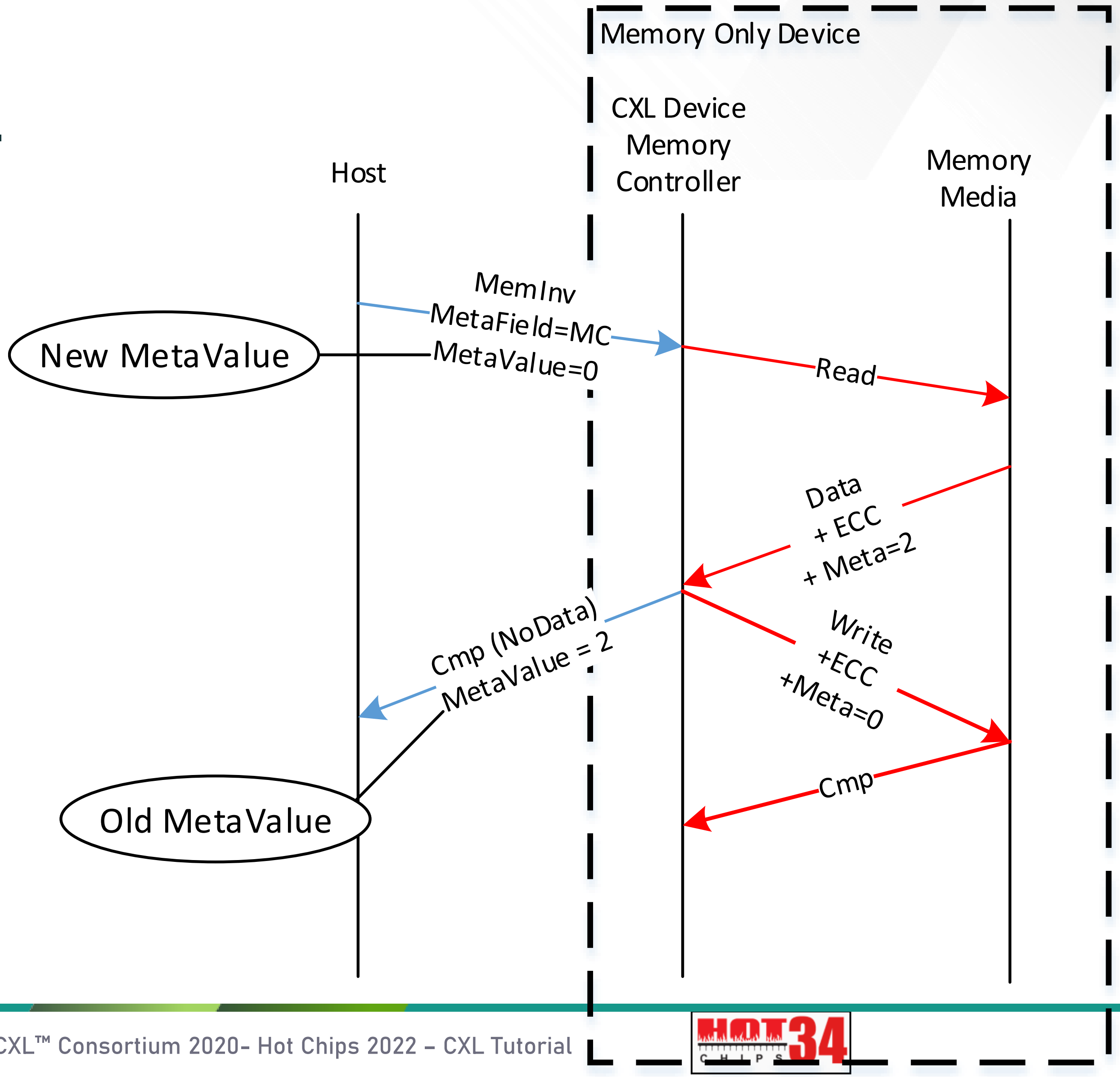
Example #3: Read no Meta

Host may indicate no Meta-state update required on reads



Example #4: MemInv

Used to read/update Meta-state without reading the data itself.



HDM-D/HDM-DB Common Attributes

“Device Coherent” → Provide ability for host and device to cache

Request MetaValue field indicates host cache state.

- Any – Host can be in M,E,S,I states
- Shared – Host can be in S or I states and indicating the host requesting S-state.
- Invalid – Host is in I-state and is not requesting cache state.

Request SnpType indicates Device Cache state change

- SnpInv – Invalidate Device Cache
- SnpData – Device Cache in I or S state.

Device Coherence Engine (Dcoh) is the final conflict resolution arbiter between host and device accesses for HDM-D* memory.

Device Coherent (HDM-D) Specifics

- CXL.mem requests indicate coherence required from the host.

- CXL.Cache used for device to change host cache state
  - Host must detect device accessing its own memory and trigger special flows which return a “Forward” message.
  - Can be blocked behind access to host memory.
  - Requires device to implement full directory tracking (aka Bias Table)

Full Directory → Bias Table

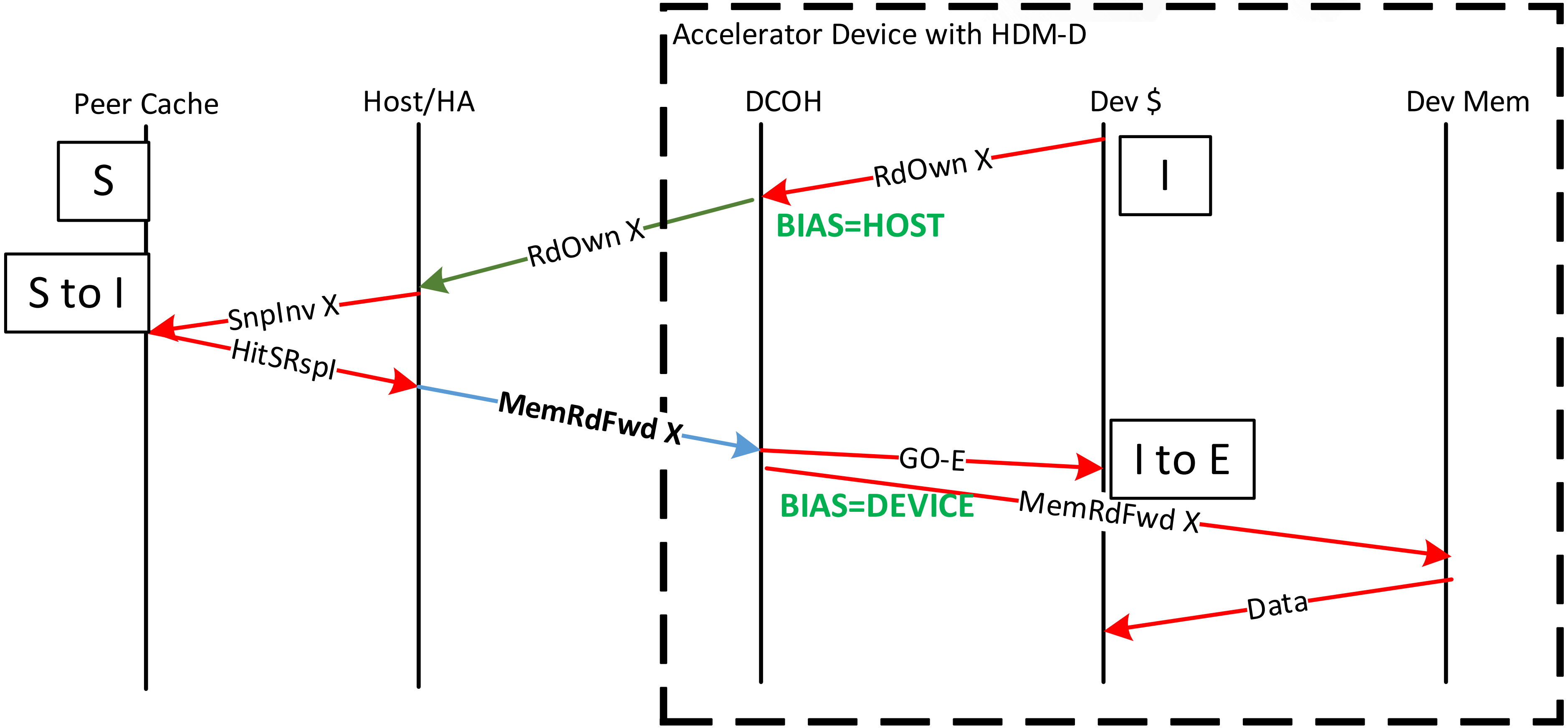
Device state of 1 or 2 bits per Cacheline indicating if host has a cached copy

- Device Bias: No host caching, allowing direct reads
- Host Bias: Host may have a cached copy, so read goes through the host
  - Optionally tracking Shared vs Any state in host.
  - With Shared State, the device may directly read data, but must not modify.

Host tracks which peer caches have copies

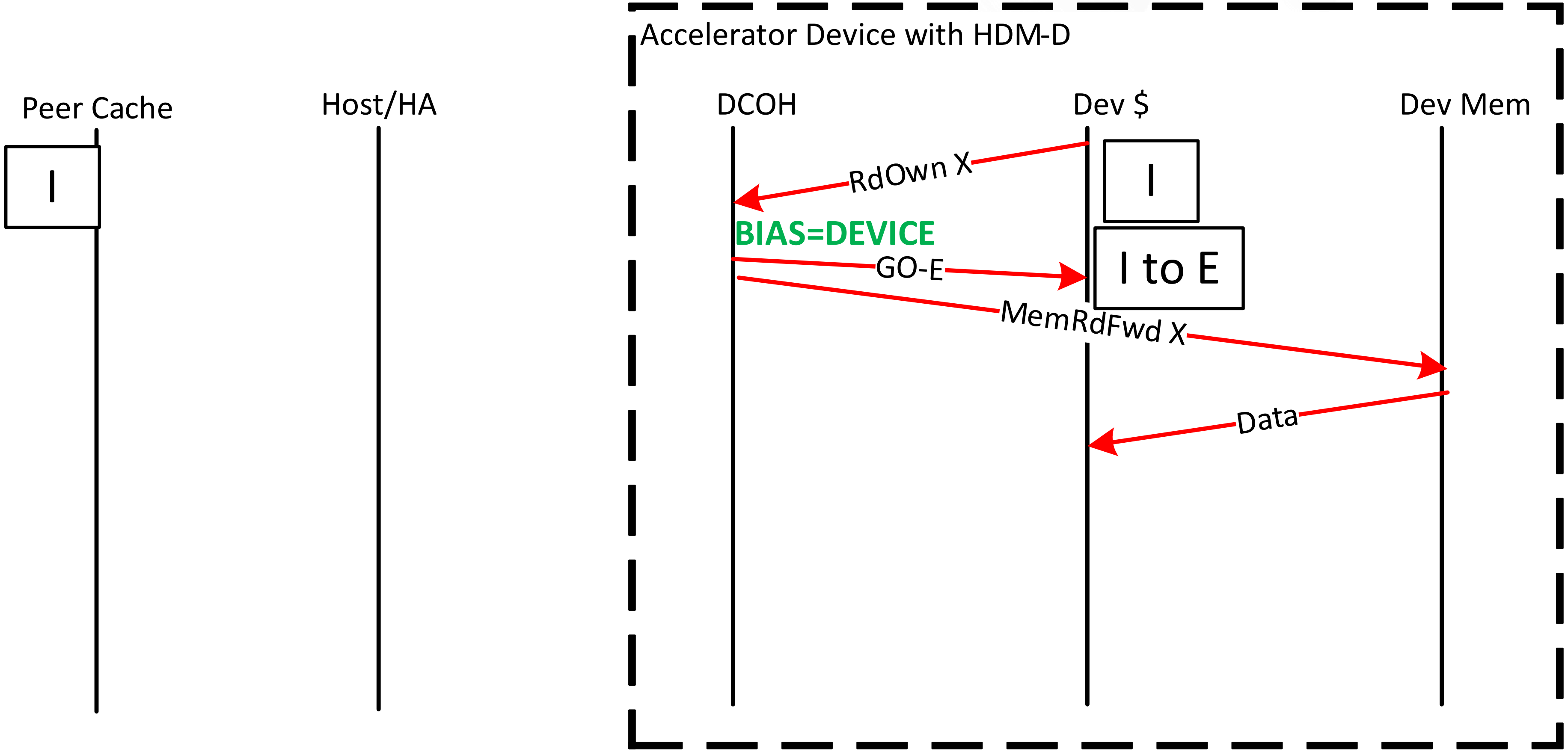
Host Bias Read

MemRdFwd message sent after coherence resolved on M2S Request Channel



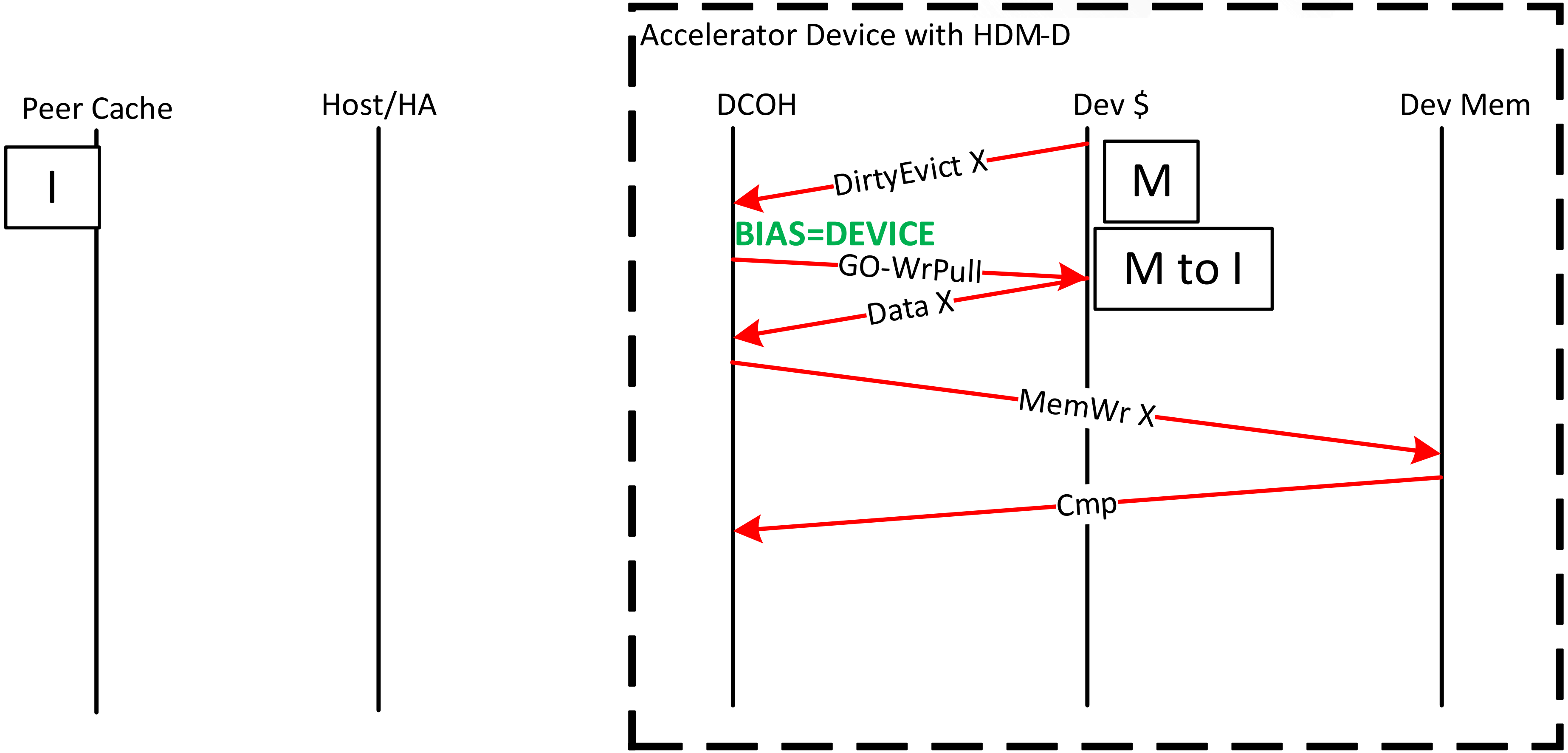
Device Bias Read

No messages on CXL interface



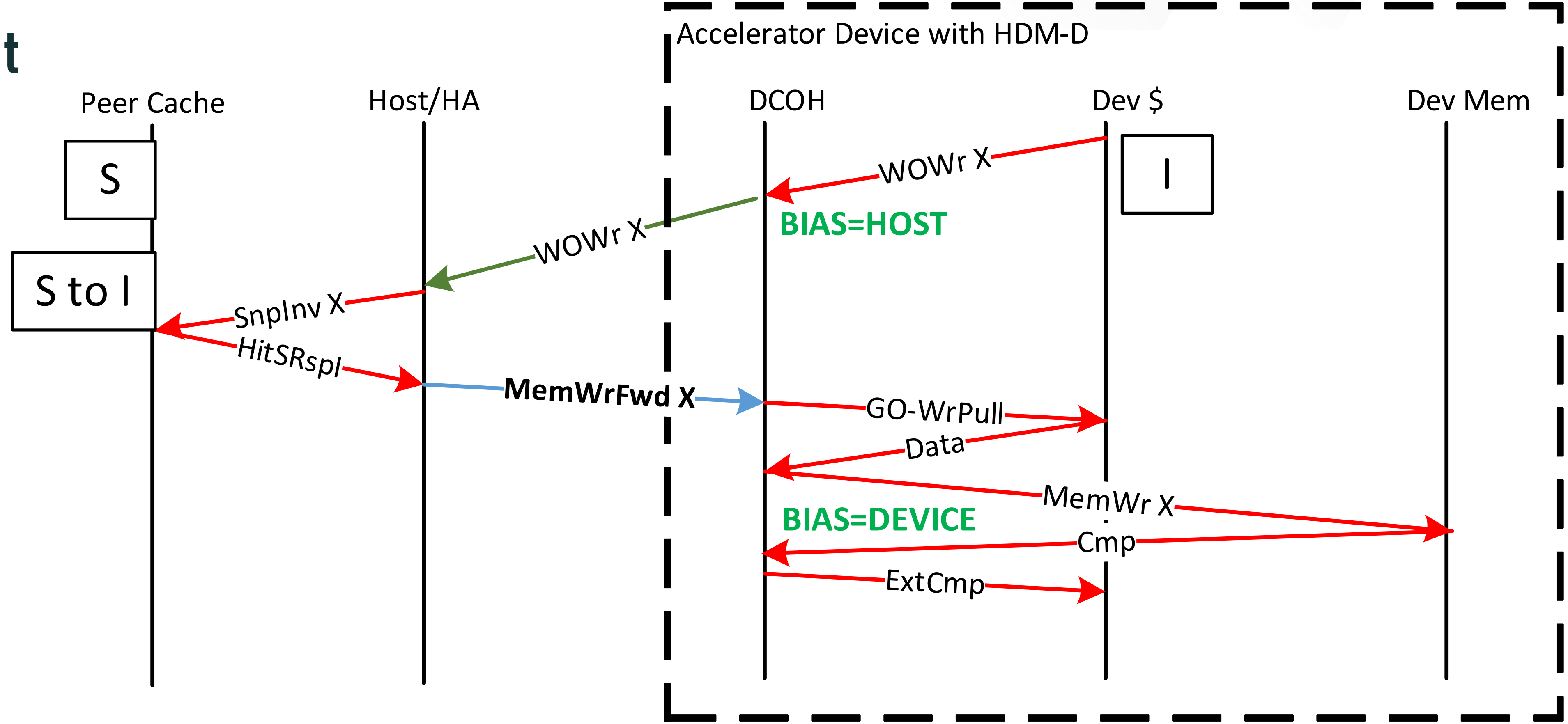
Device Cache Evictions

E/M in cache imply Bias=Device so no indication to host



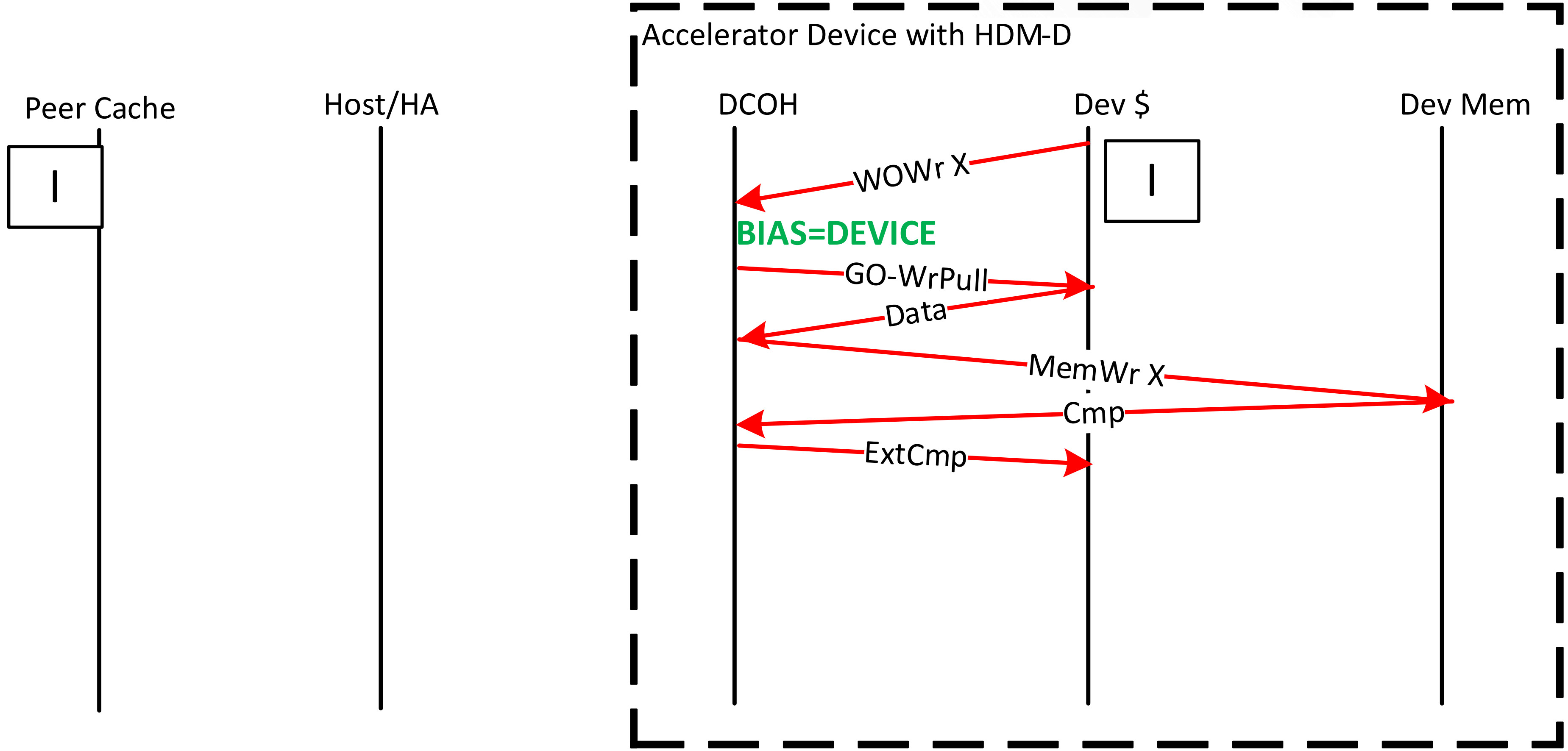
Host Bias Streaming Write

MemRdFwd message sent after coherence resolved



Device Bias Streaming Write

No message to host



HDM-DB → New in CXL3

HDM-DB

“Device Coherent with Back-Invaldation” (HDM-DB) adds BISnp and BIRsp channel for optimize coherence management enabling inclusive Snoop Filter (SF) architectures.

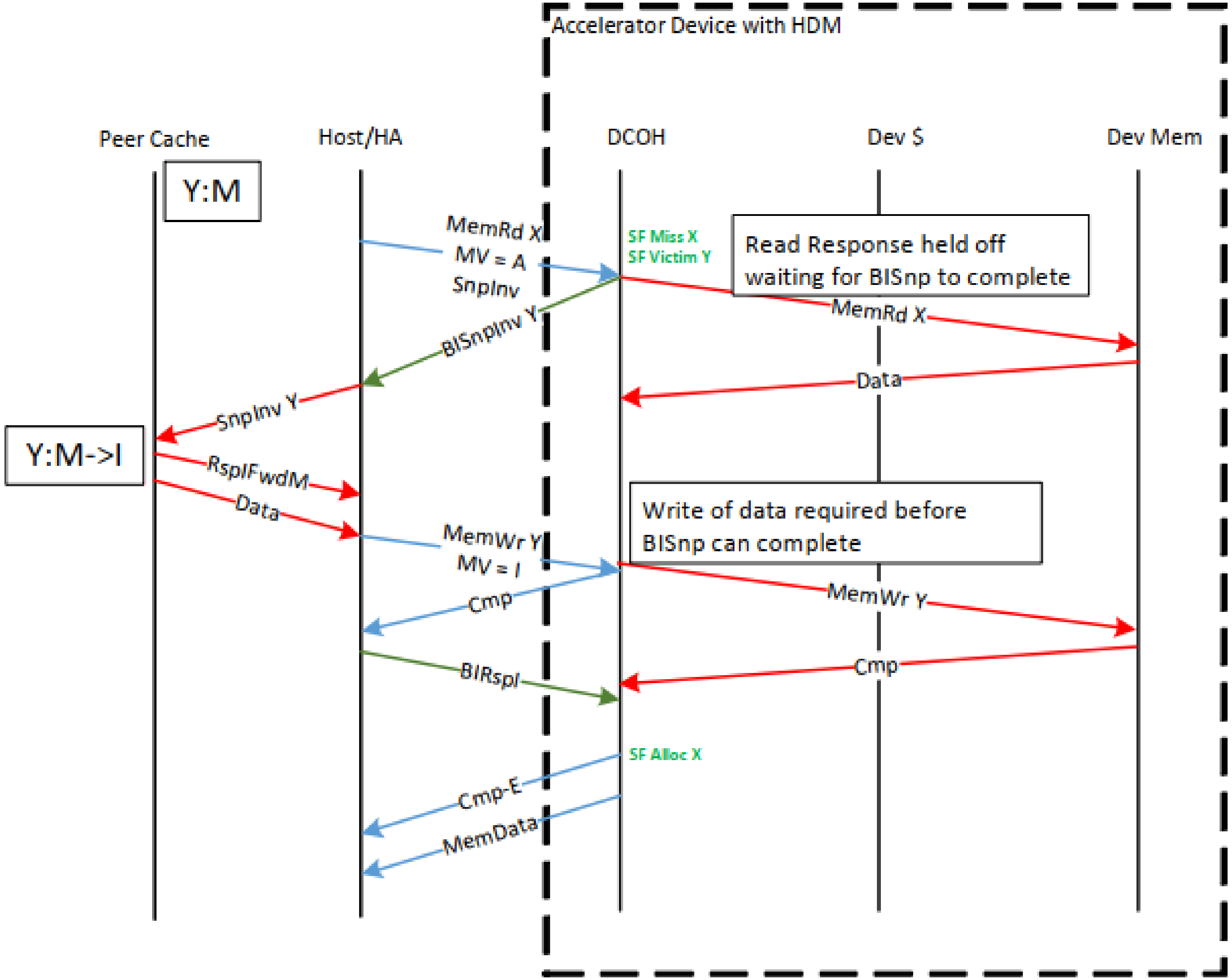
Same “BIAS Table” states tracking host coherence: I, S, A

Inclusive SF architecture may block M2S Request waiting for Back-Invalidation Snoop (BISnp) to complete which enables sizing to match host caching expect instead of memory capacity.

Back-Invalidation Snooping (HDM-DB)

Enables Inclusive Snoop Filter (SF) to track host caching

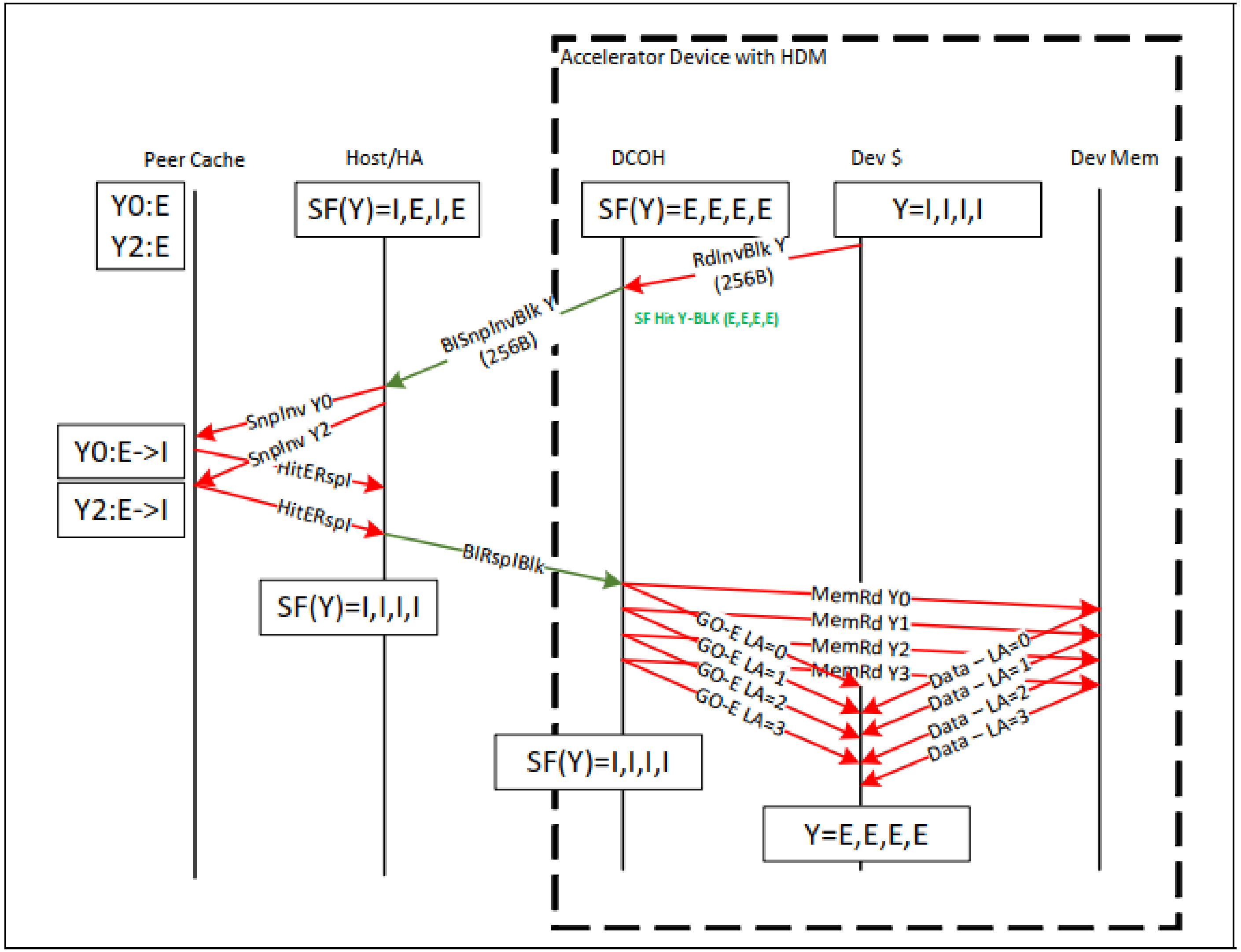
Device can block new requests waiting for SF Victim



Block Access with BISnp

- To improve efficiency there is BISnp messages that cover more than one cacheline (aka “Block”).

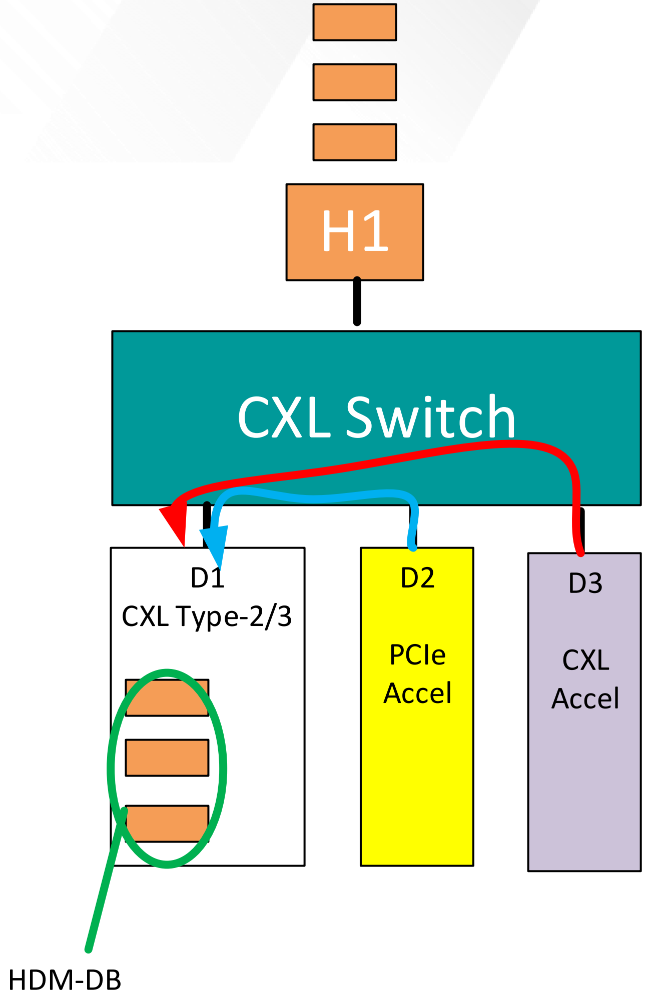
- Either 2 (128B) or 4 (256B) cachelines are supported.



New Use Models with HDM-DB

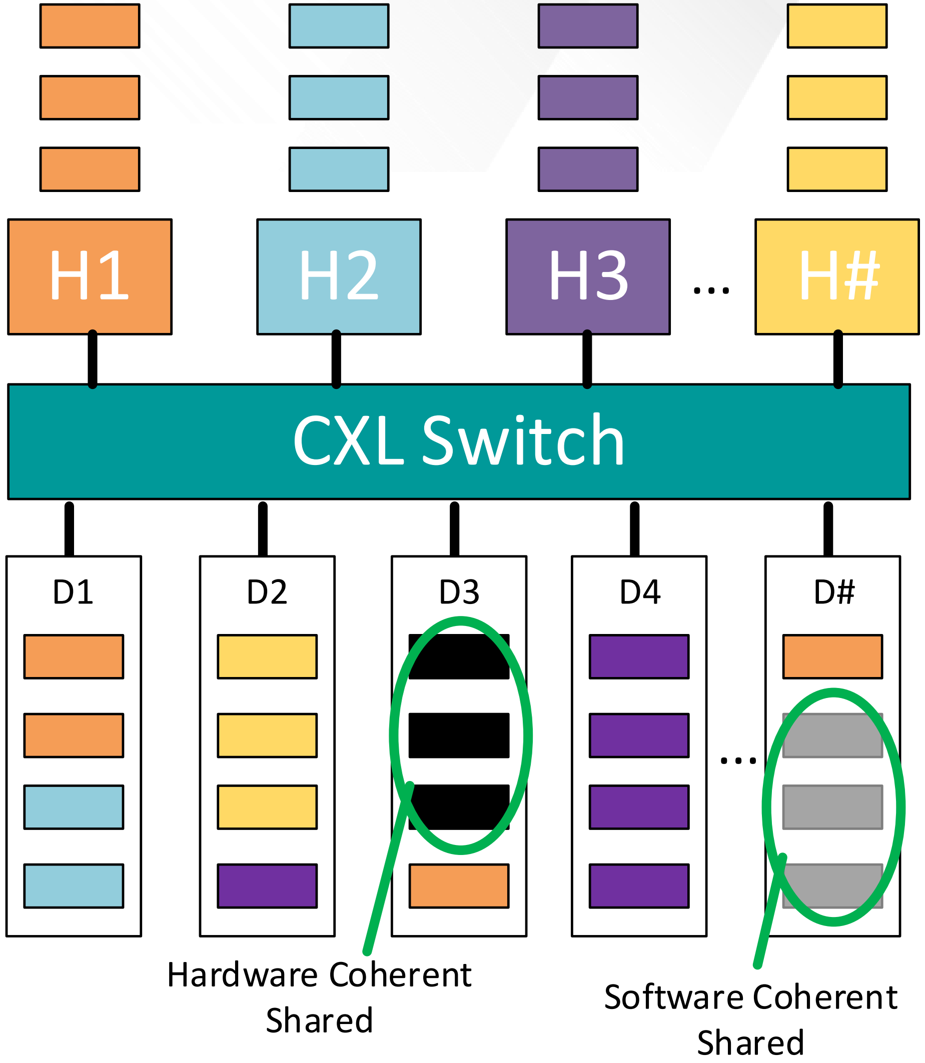
Direct Peer-to-Peer (P2P) to HDM

- HDM-DB enables direct P2P from CXL or PCIe sources in CXL3
- In prior generation all HDM access must go through the host CPU to resolve coherence.
- HDM-DB will directly resolve coherence with the host before committing the P2P.



Pooled and Shared Memory

- Pooled Memory and CXL Switching added in CXL2 allow for dedicated assignment of memory resources from to a host.
- Shared Memory assigned to multiple hosts enabled in CXL3
- Multi-Host Hardware Coherent Shared Memory possible with HDM-DB
- MORE on these uses in Fabric Tutorial



Summary

- CXL protocols are evolving
- CXL2 added switching and pooled memory capabilities.
- CXL3 enabling new capabilities:
  - CXL.Cache Scaling
  - CXL.Mem Back-Invalidation Channel for SF, Direct P2P, Multi-Host Coherence
  - Port Based Routing (covered in Fabric Tutorial))

Compute Express Link ™

Thank You

www.computeexpresslink.org/join

@ComputeExLink

www.linkedin.com/company/cxl-consortium/

CXL Consortium Channel

Audience Q&A