Bridge of Life Education

SOC Design
Interconnect – IO Cache Access

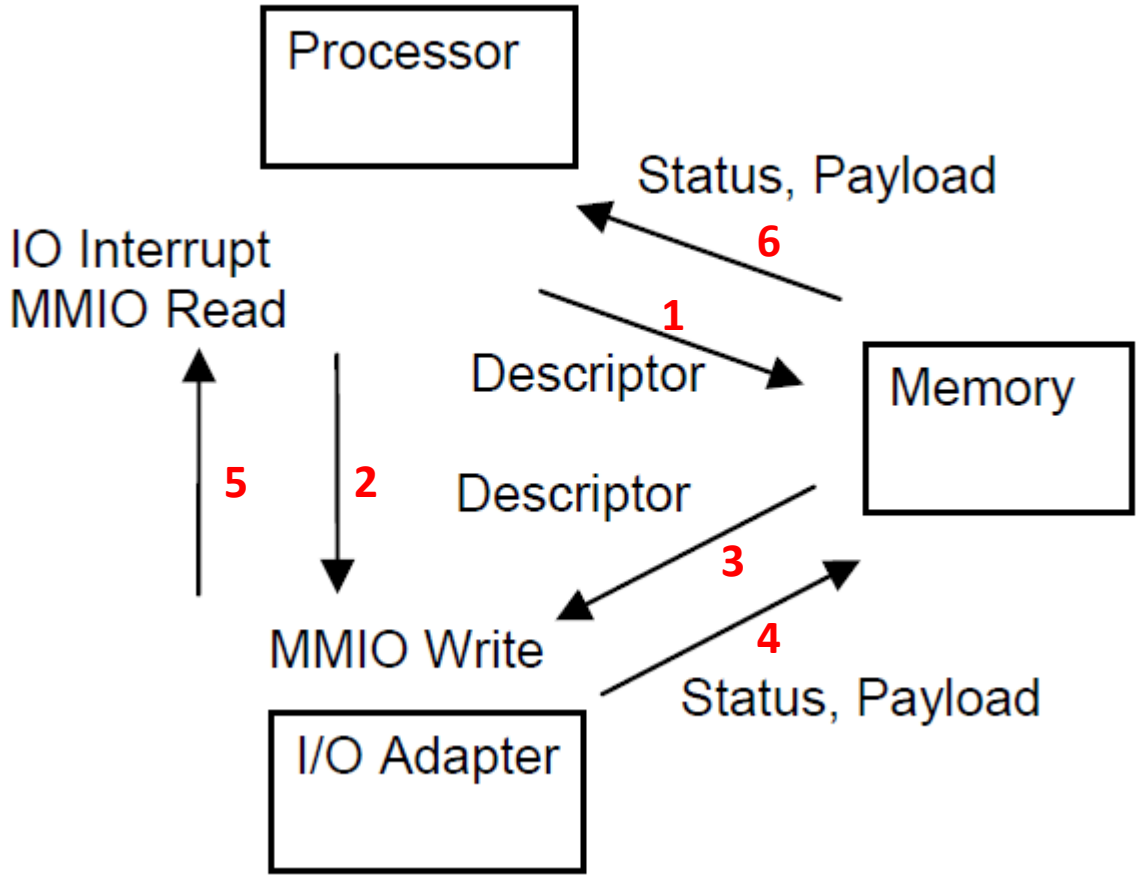
Jiin Lai

Topics

1. Processor/Memory/IO Interaction
2. Direct Cache Access
3. PCIe Transaction Processing Hit (TPH)
4. Xilinx Zynq IO Cache Coherent Access

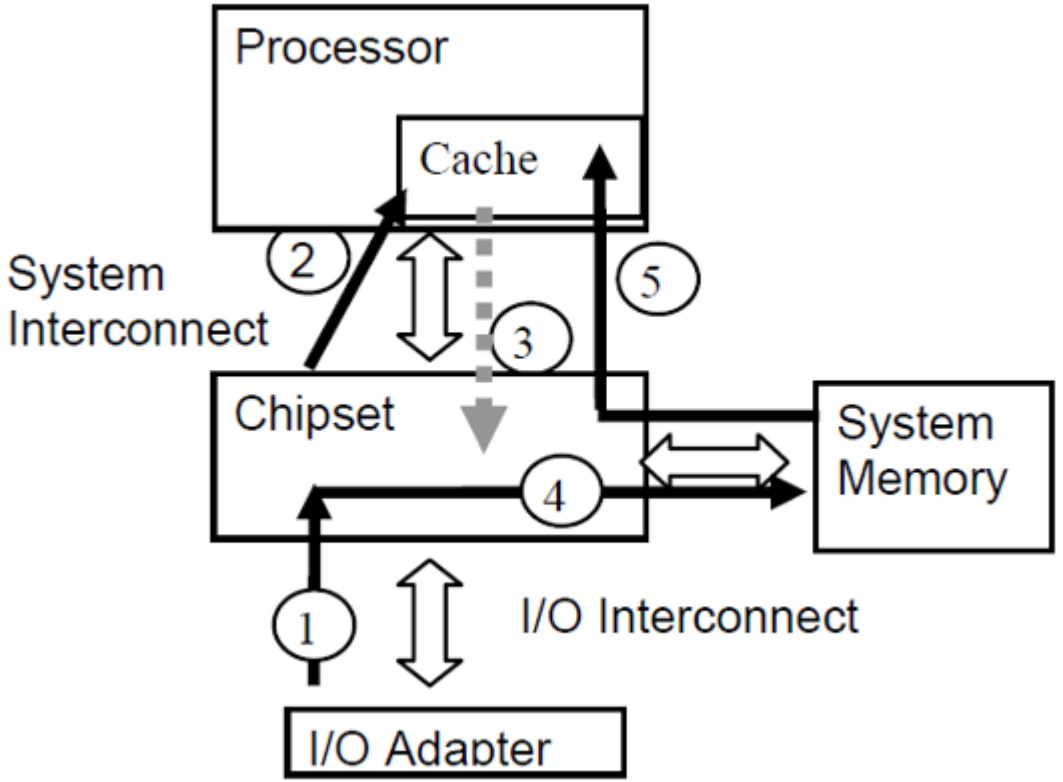
Interactions between Processor, Memory and I/O

1. Processor sets up buffer and descriptor (pointer to buffer)
2. Write to a MMIO register
3. I/O adaptor fetches descriptor
4. I/O adaptor completes a block transfer, and update status
5. I/O adapter sends an interrupt, Processor query the device using MMIO read
6. Processor reads stats and payload data



I/O Read/Write to Memory from A Cache Perspective -

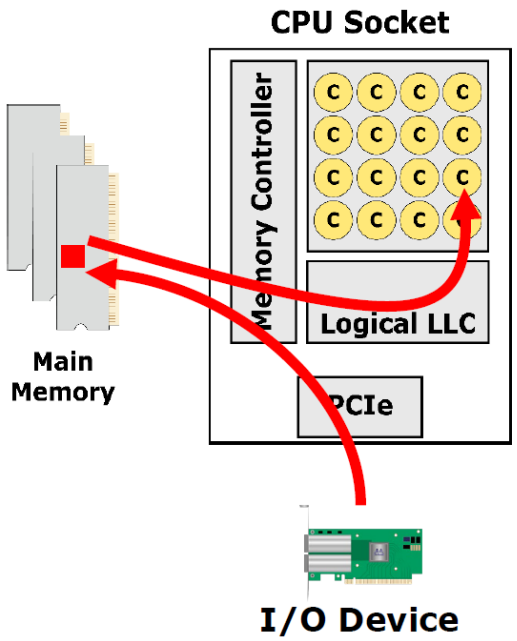
1. IO read/write to system memory
2. Snoop on system interconnect, if cache hit
    1. If write, invalidate from the process cache
    2. If read, marked as a shared cacheline
3. If cacheline is modified, write-back to system memory
4. Write back date merge with IO write data
5. Process subsequently read a cache miss



Direct Cache Access - DCA

- IO data directly into the processor’s cache
- CPU read will be cache hit
- Reduce memory bandwidth
- Write to cache, eliminate the need to write data to memory
- An industry example: PCIe Transaction Layer Packet Processing Hint

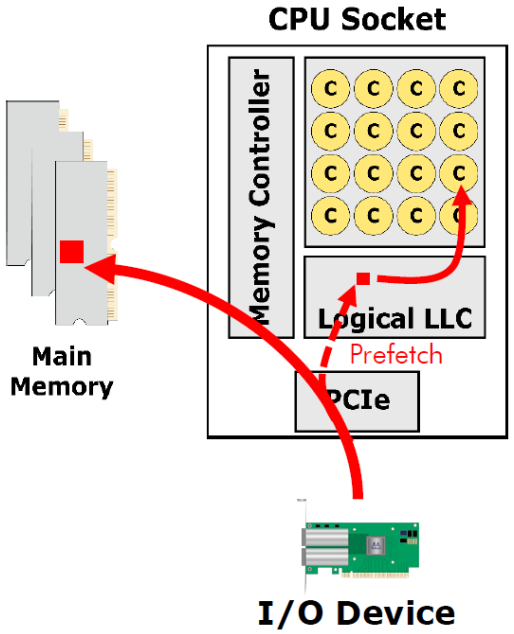
Convention I/O



1. I/O device DMAs* packets to main memory
2. 2. CPU later fetches them to cache

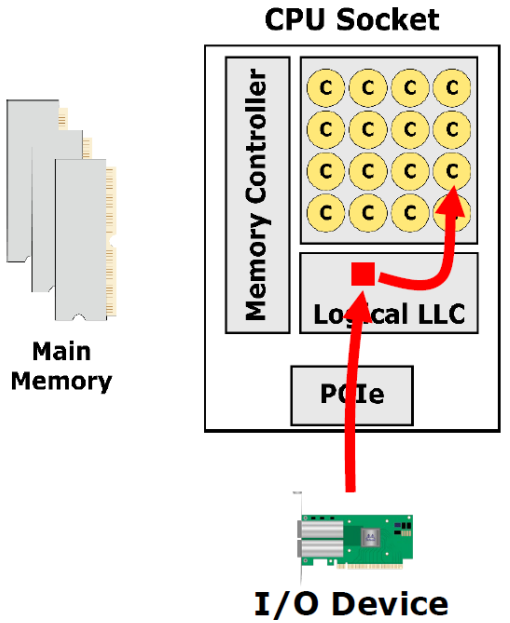
Inefficient:
- Large number of accesses to main memory
- High access latency (>60ns)
- Unnecessary memory bandwidth usage

Direct Cache Access (DCA)



1. I/O device DMAs packets to main memory
2.DCA exploits TPH* to prefetch a portion of packets into cache
3.CPU later fetches them from cache
Still inefficient
- in terms of memory bandwidth usage
- Requires OS intervention and support from processor

Intel DDIO



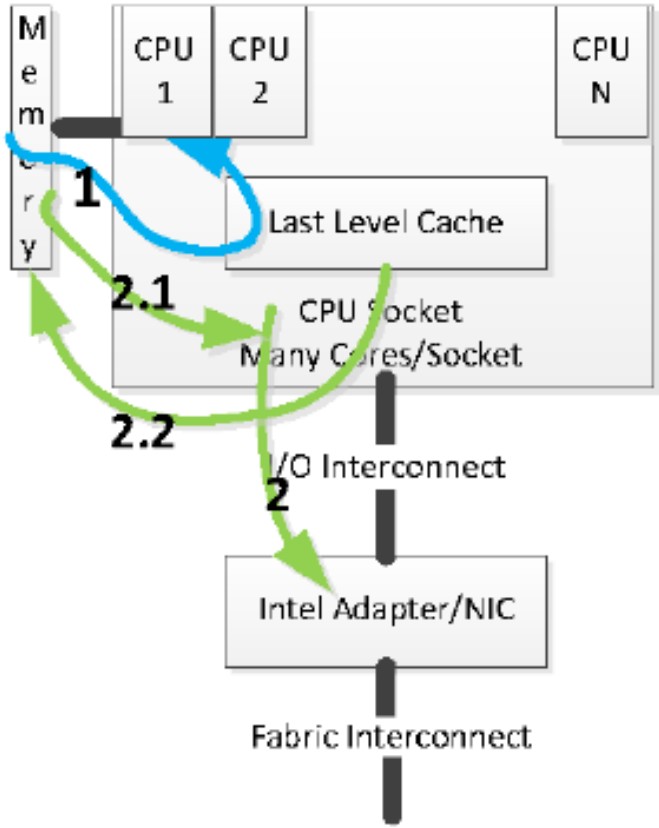
1. DDIO in Xeon processors since Xeon E5
2. DMA packets or descriptors directly to/from Last Level Cache (LLC)

IO Read System Memory

1. CPU allocates data buffer with data, and control structures. Then IO is notified to begin a transmit operation.

2. When the IO receives notification for starting a transmit operation, it first reads control structures and subsequently the packet data. The data was very likely in CPU cache.
   2.1 Each read operation triggered by the IO causes data to be write-back from the cache, if present and dirty;
   2.2 This read operation also causes a speculative read to be issued to memory in parallel, while the coherency protocol checks if the data happens to be in the CPU's caching hierarchy.

=> a single read operation caused two or three trips to memory, depending on the platform.



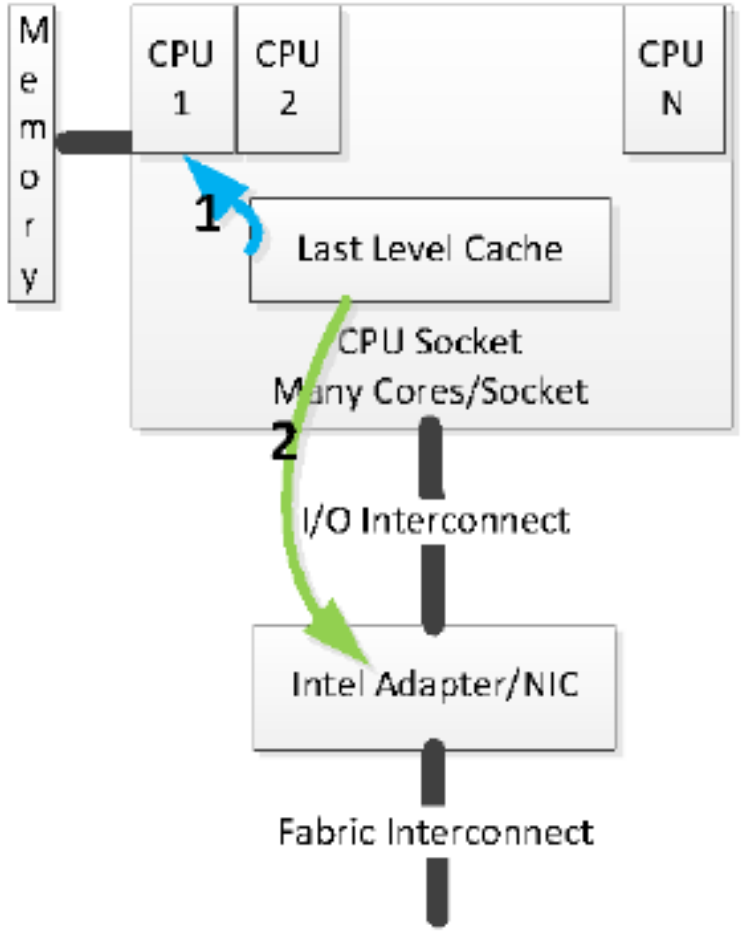
IO Read System Memory - DDIO

1. Data access operations associated with creating a packet are satisfied from within the cache. Thus, software running on the CPU does not encounter cache misses, and, therefore does not have to fetch data from memory.

2. The read operation initiated by the IO is satisfied by data from the cache, without causing evictions, that is, the data remains in the cache; since this data is re-used by software, it stays in the cache.

Benefit:

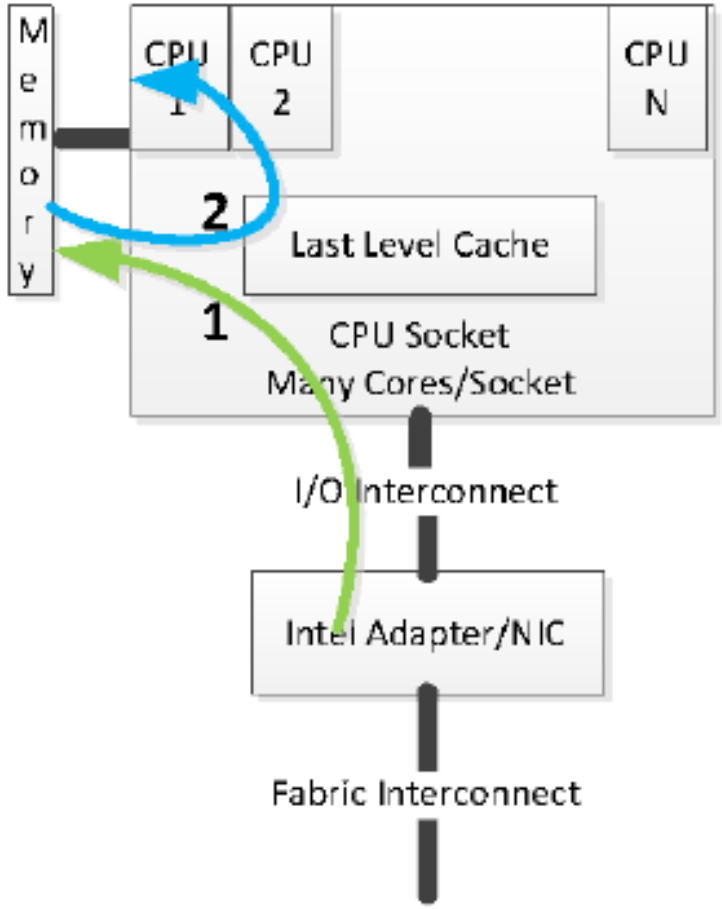
1. fewer trips to memory, and, in the ideal case, no trips to memory.
2. The data in the cache is not disturbed by an I/O data consumption operation



IO Write System Memory

1. IO transfers data (packets or control) to host memory. If the data being delivered happens to be in the CPU's caching hierarchy, it is invalidated.

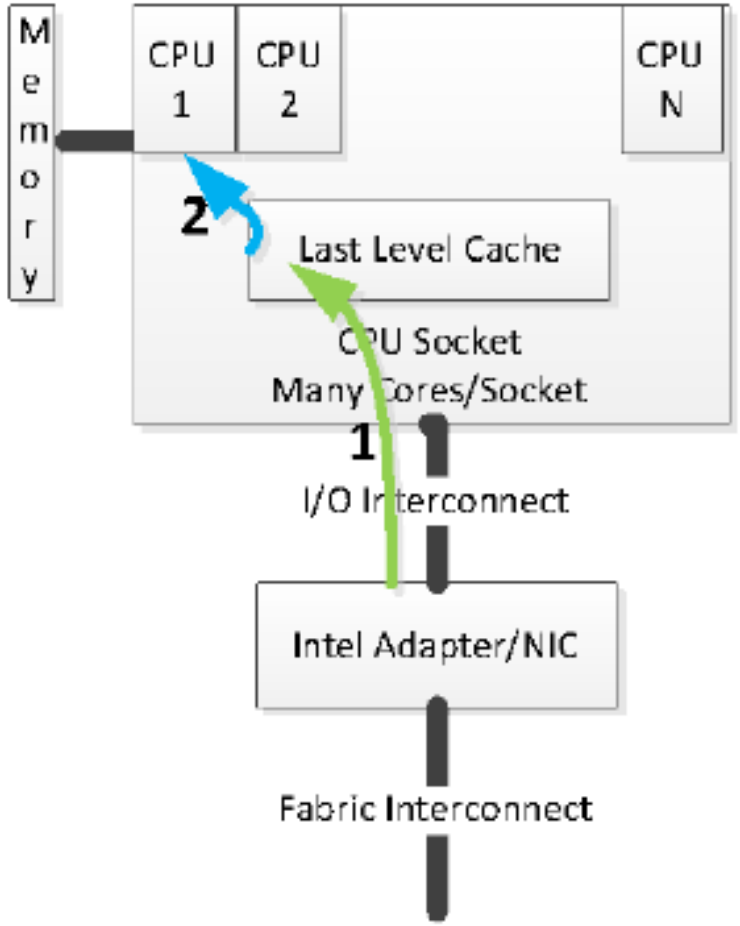
2. SW running on the CPU reads the data to perform processing tasks. These data access operations misses cache (See step 1 above for explanation) and causes data to be read in from memory, into the CPU's caching hierarchy.



IO Write System Memory - DDIO

1. I/O data delivery uses Write Update or Write Allocate operations (depending on cache residency of the memory addresses to which data is being delivered), which causes data to be delivered to the cache, without going to memory.

2. The subsequent read operations initiated by SW are satisfied by data from the cache, without causing cache misses.



Not all I/O data updates on Processor's Cache

PCIe Transaction Processing Hit (TPH)

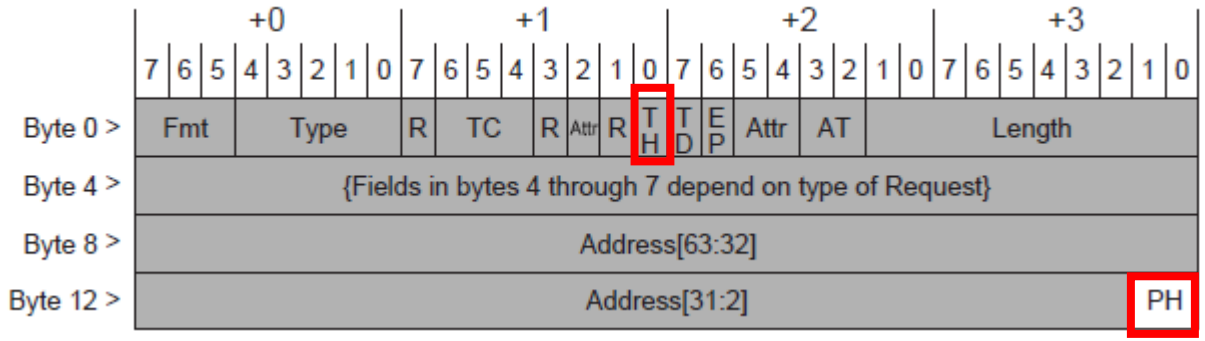
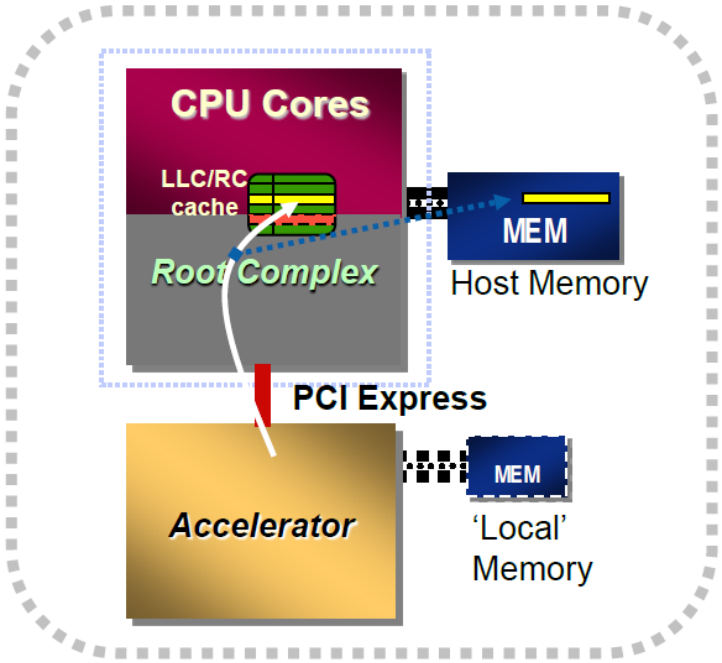


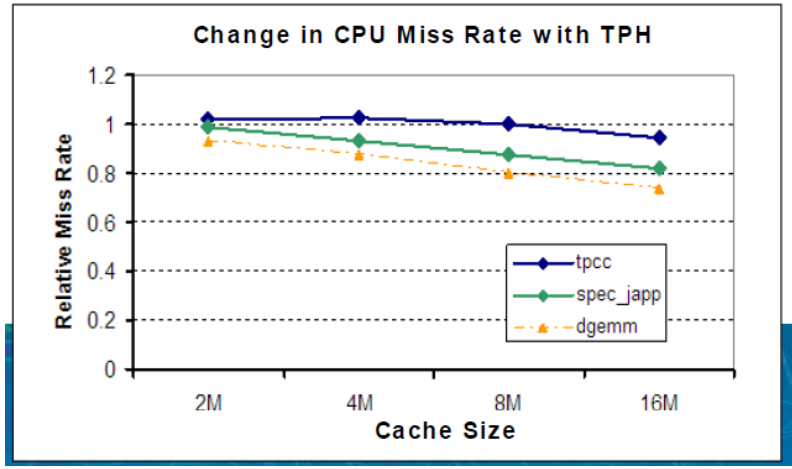
Table 2-16: Processing Hint Encoding

PH[1:0] (b)	Processing Hint	Description
00	Bi-directional data structure	Indicates frequent read and/or write access to data by Host and device
01	Requester	Indicates frequent read and/or write access to data by device
10	Target	Indicates frequent read and/or write access to data by Host
11	Target with Priority	Indicates frequent read and/or write access by Host and indicates high temporal locality for accessed data

Transaction Processing Hints



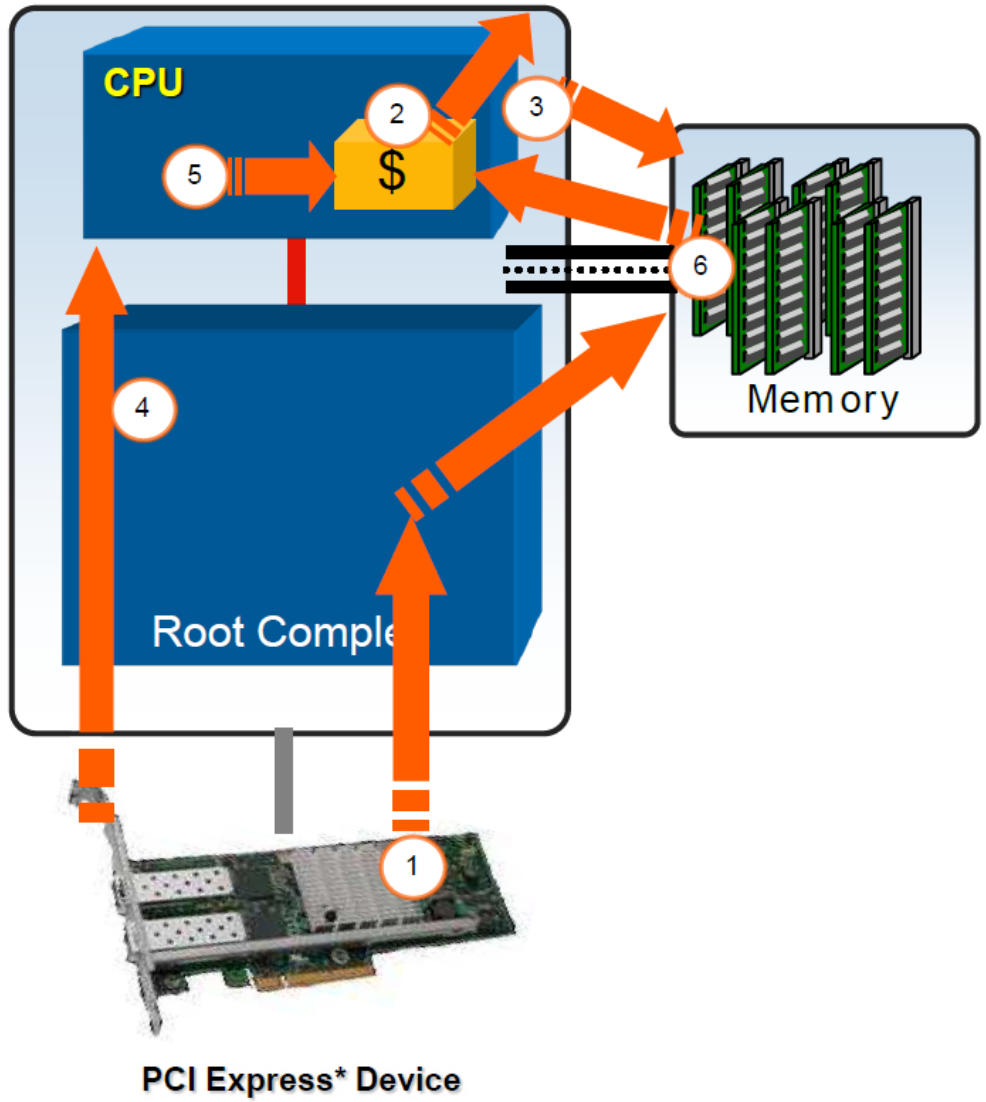
- Background:
  - **Small I O Caches implemented in server platforms**
    - Ineffective w/o info about intended use of IO data
- Feature:
  - **TPH= hints on a transaction basis**
    - Allocation & temporal reuse
  - **More direct CPU< ->I O collaboration**
    - Control structures (headers, descriptors) and data payloads
- Benefits:
  - **Reduced access latencies**
    - Improved data retention/allocation
  - **Reduced mem & QPI BW/ power**
    - Avoiding data copies
  - **New applications**
    - Comm adapters for HPC and DB clusters, Computational Accelerators,…



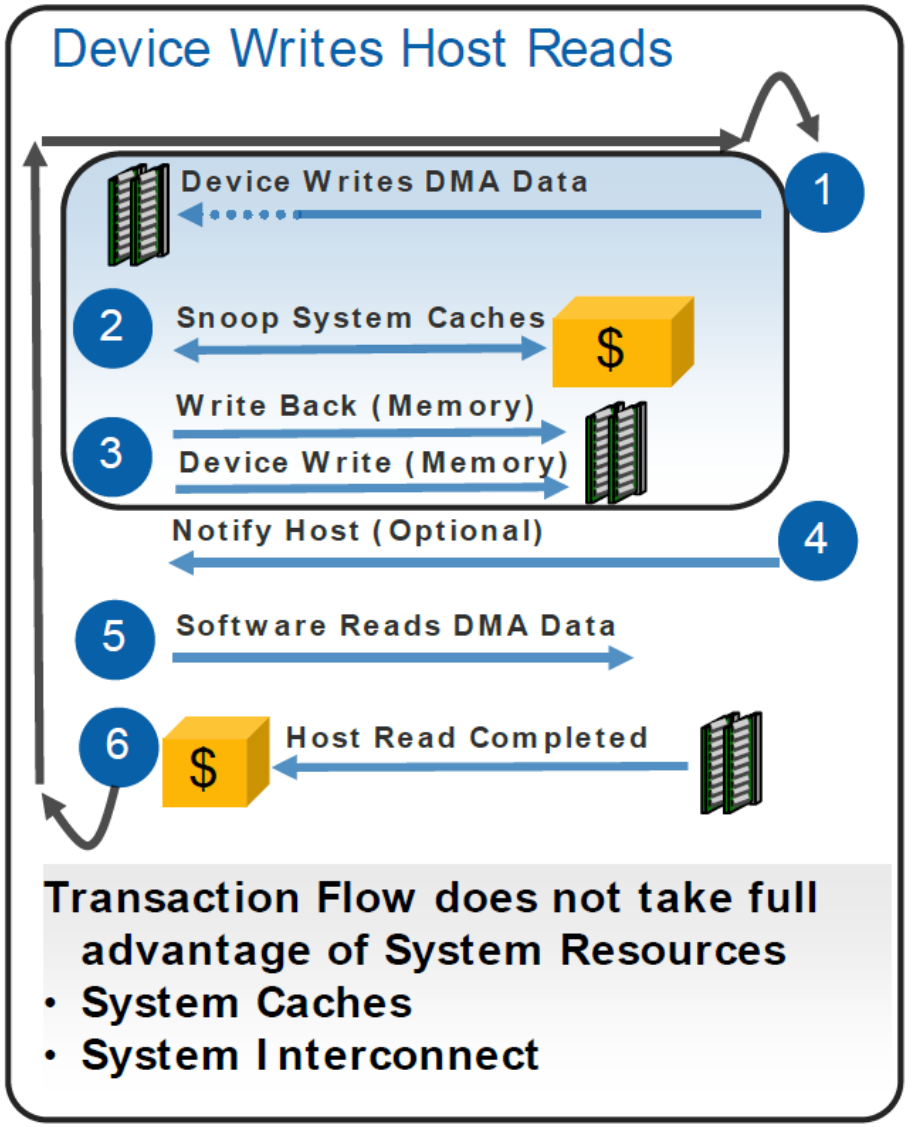
Provides stronger coupling between Host Cache/Memory hierarchy and IO

https://old.hotchips.org/wp-content/uploads/hc_archives/hc21/1_sun/HC21.23.1.SystemInterconnectTutorial-Epub/HC21.23.131.Ajanovic-Intel-PCIeGen3.pdf

Basic Device Writes



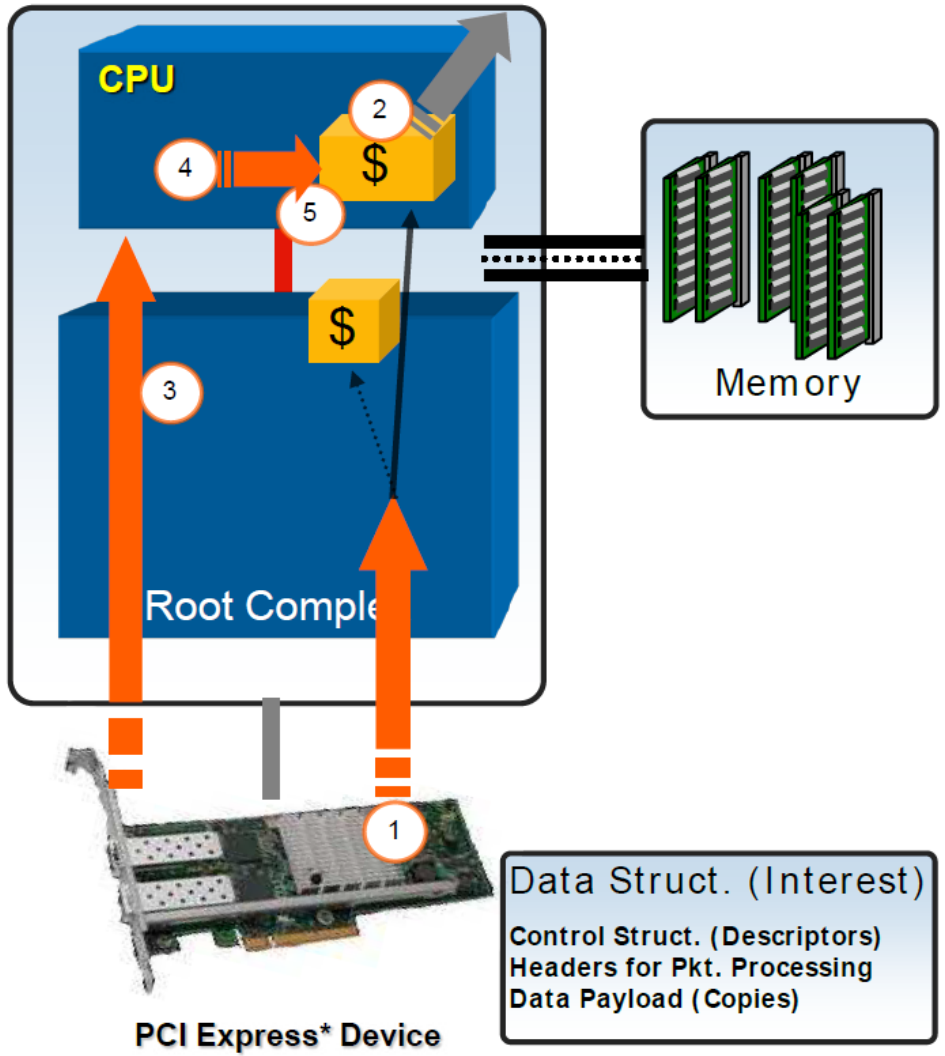
PCI Express* Device

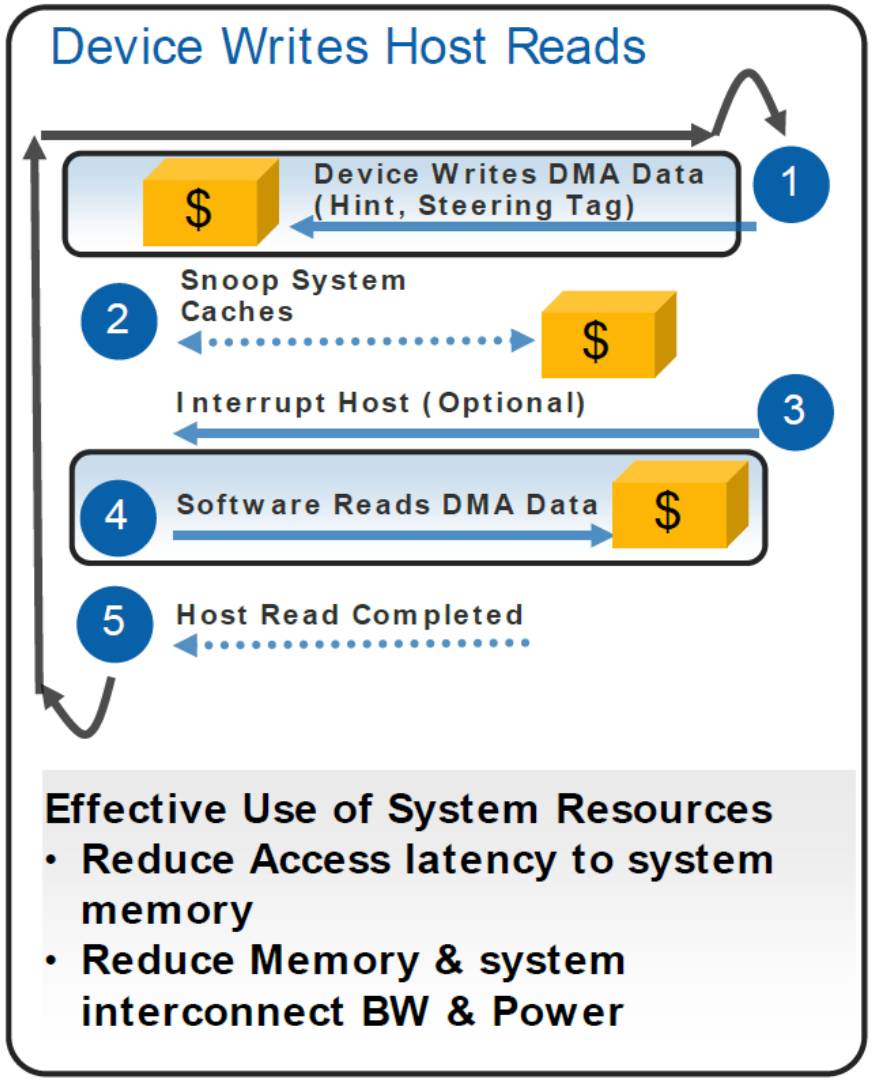


Transaction Flow does not take full advantage of System Resources

- **System Caches**
- **System Interconnect**

Device Writes with TPH

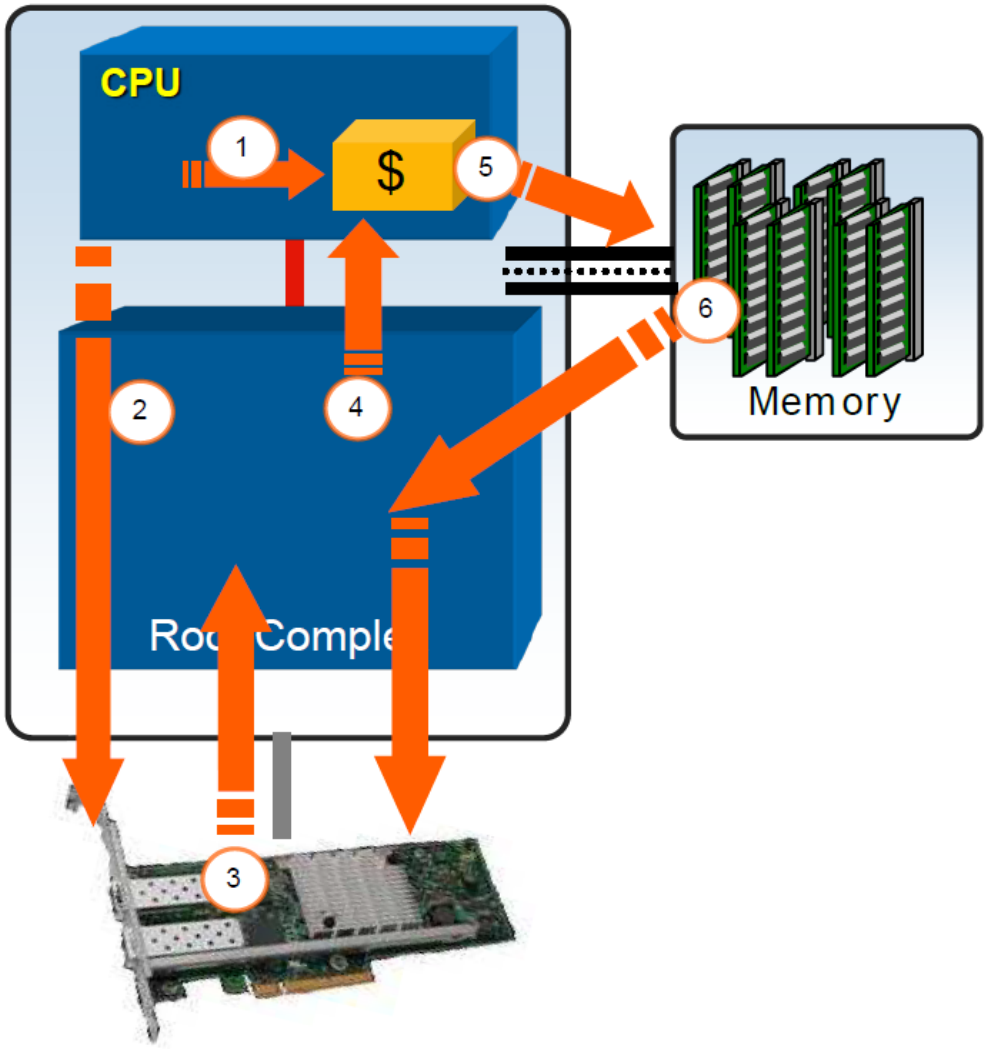




Effective Use of System Resources

- **Reduce Access latency to system memory**
- **Reduce Memory & system interconnect BW & Power**

Basic Device Reads



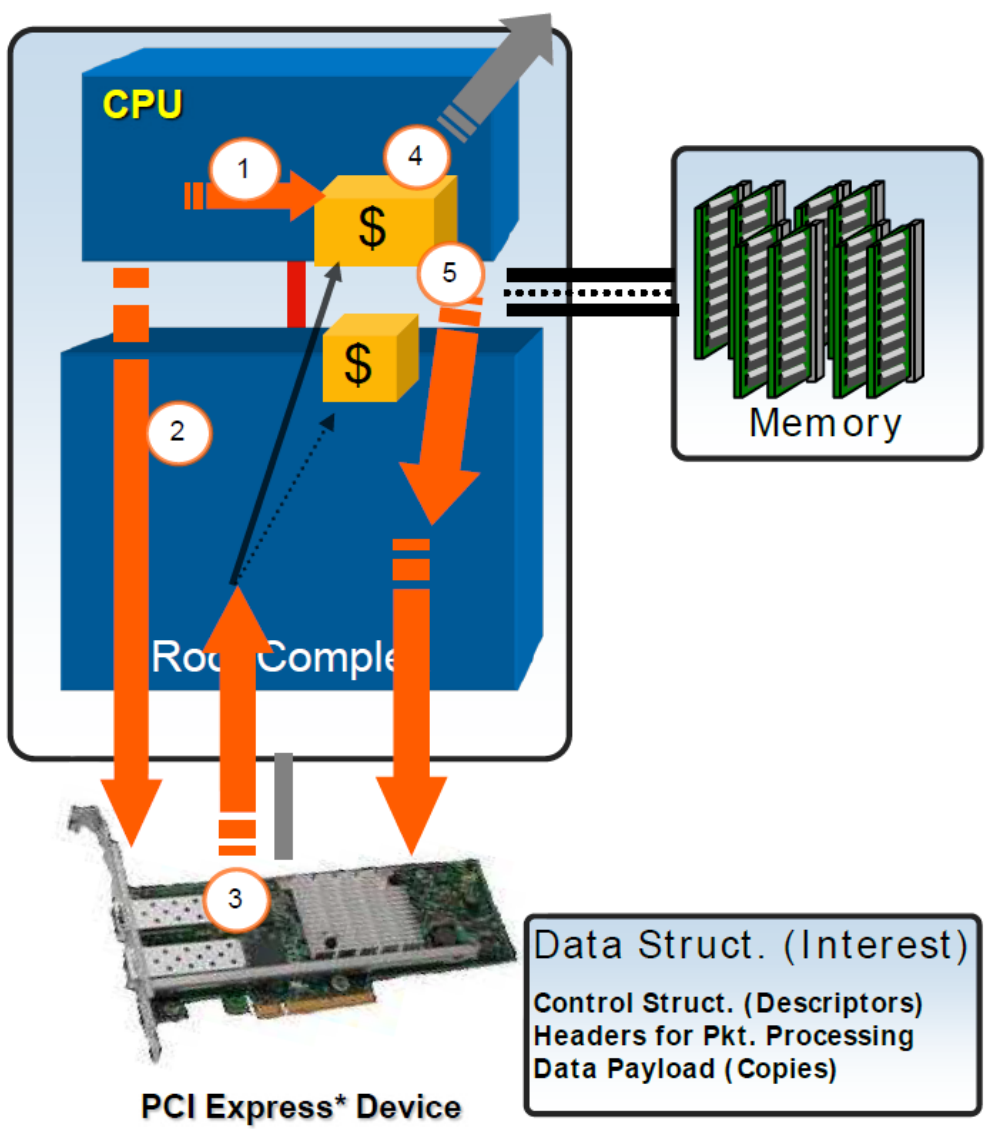
Host Writes Device Reads

1. Software Writes DMA Data → $
2. Command Write to Device (Optional)
3. Device Performs Read
4. Snoop System Caches → $
5. Write Back to Memory
6. Device Read Completed

Transaction flow does not take full advantage of System Resources

- System Caches
- System Interconnect

Device Reads with TPH



Data Struct. (Interest)

Control Struct. (Descriptors)
Headers for Pkt. Processing
Data Payload (Copies)

Host Writes Device Reads

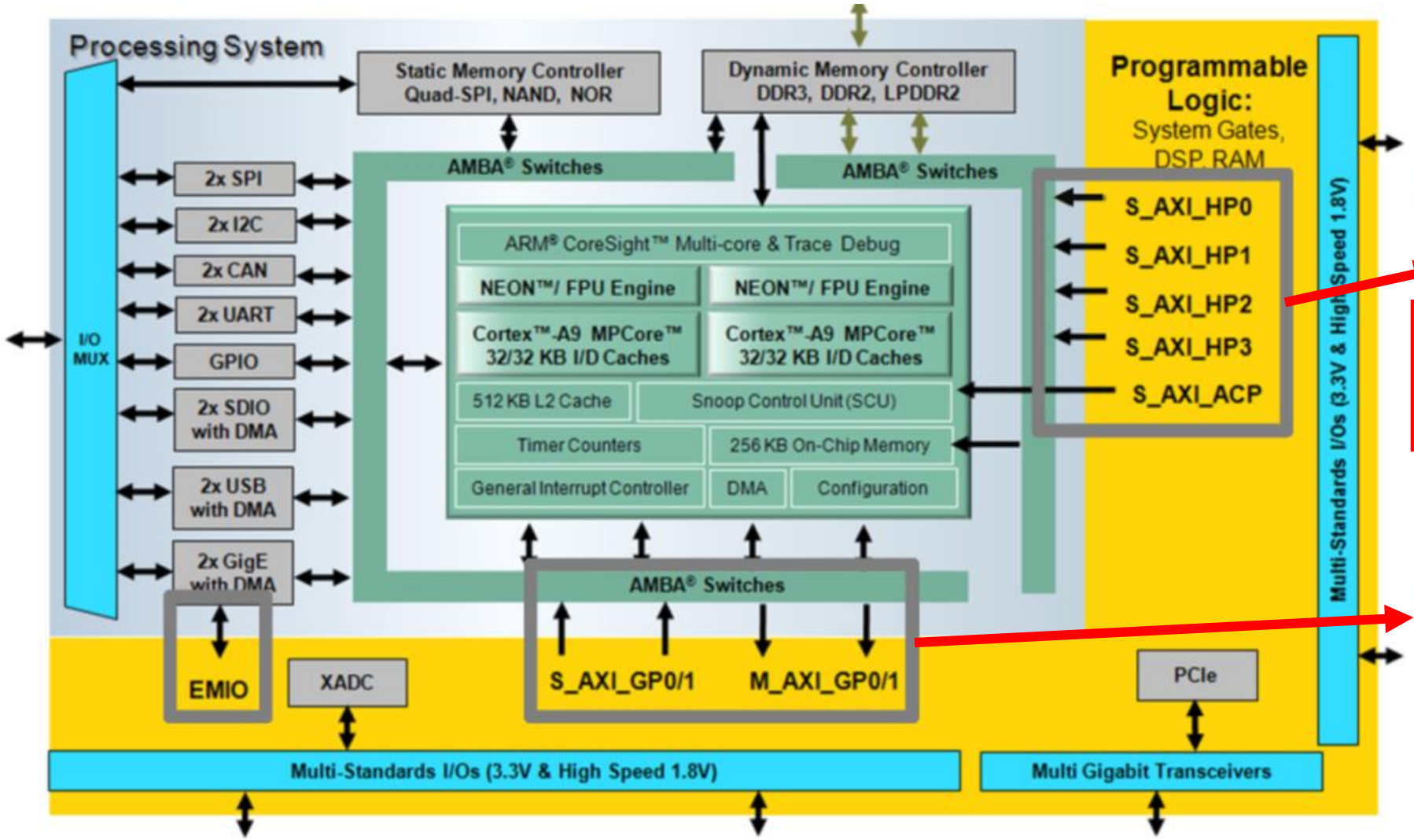
1. Software Writes DMA Data → $
2. Command Write to Device (Optional)
3. Device Performs Read (Hint, Steering Tag) ← $
4. Snoop System Caches ⇢ $
5. $ Device Read Completed

Effective Use of System Resources

- **Reduce Access latency to system memory**
- **Reduce Memory & system interconnect BW & Power**

Xilinx Zynq FPGA for IO Cache Access

Zynq Block Diagram



> **AXI High Performance**
>> 4x Slave (64-bit)
- 1K FIFOs

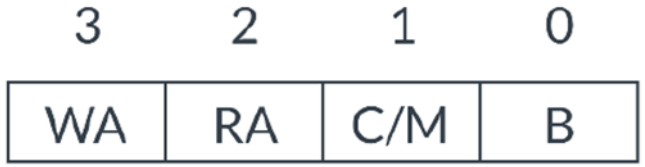
> **ACP**
>> 1x Slave (64-bit)
>> Cache access

> **AXI General Purpose ports**
>> 2x Master (32-bit)
>> 2x Slave (32-bit)

https://docs.xilinx.com/r/en-US/ug1085-zynq-ultrascale-trm/ACP-Interface?tocId=nBXmrCWPBNy5eVgxEHWuzQ

- ACP – Accelerator Coherency Port

- ACP Transaction is coherent to the APU L1/L2 Cache
- Read Transaction
  - Hit L1 or L2 Cache – data is returned from L1/L2 Cache
  - If miss, request forward to DDR memory
- Write Transaction
  - Optionally allocate into the L2 cache



- ARCACHE and AWCACHE limited ‘b0111, ‘b1011, ‘b1111 (Cachable & Bufferable)
- Up to 4 outstanding transactions (using different AXI ID)

Accelerator Coherence Port (ACP)

- ACP allows limited support for Hardware Coherency
  - Allows a PL accelerator to access cache of the Cortex-A9 processors
  - PL has access through the same path as CPUs including caches, OCM, DDR, and peripherals
  - Access is low latency (assuming data is in processor cache) no switches in path
- ACP does not allow full coherency
  - PL is not notified of changes in processor caches
  - Use write to PL register for synchronization
- ACP is compromise between bandwidth and latency
  - Optimized for cache line length transfers
  - Low latency for L1/L2 hits
  - Minimal buffering to hide external memory latency
  - Treating all ACP transaction coherent. May cause undesirable cache thrashing.