Invited: Compute Express Link™ (CXL™): An Open Interconnect for Cloud Infrastructure

Debendra Das Sharma

Intel Senior Fellow, Data Center Group, Intel Corporation, Santa Clara, CA 95052, USA
debendra.das.sharma@intel.com

2023 60th ACM/IEEE Design Automation Conference (DAC) | 979-8-3503-2348-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/DAC56929.2023.10247726

***Abstract*— Compute Express Link is an open industry standard interconnect offering caching and memory semantics on top of PCI-Express (PCIe)®, with resource pooling and fabric capabilities. This paper delves into the role of CXL to solve some of the challenges in the cloud infrastructure.**

Keywords—CXL, accelerator, memory expansion, memory hierarchy, memory pooling and sharing, fabric, coherency

I. Introduction

Compute Express Link (CXL) is an open industry standard interconnect offering high-bandwidth, low-latency connectivity between host processor, CXL switch for fan-out, and devices such as accelerators, memory buffers, and smart I/O devices such as Network Interface Card (NIC). It is designed to address the growing high-performance computational workloads by supporting heterogeneous processing and hierarchical heterogeneous memory systems with applications in Artificial Intelligence, Machine Learning, analytics, cloud infrastructure, network and Edge, communication systems, and High Performance Computing by enabling coherency and memory semantics on top PCI-Express® (PCIe® - both PCIe 5.0 and PCIe 6.0). CXL is essential as processing data in these emerging applications requires a diverse mix of scalar, vector, matrix and spatial architectures deployed in CPU, GPU, FPGA, smart NICs, and other accelerators.

II. Challenges in Cloud Infrastructure and CXL

As described above, heterogeneous computing is essential to meet the increasing and diverse demands of applications supported by the cloud infrastructure. PCI-Express already offers a good infrastructure in terms of its wide adoption across platforms, power-efficient performance, and direct memory access semantics. However, memory accesses across PCIe are non-coherent and the entire data structure needs to be transferred back and forth between the device and system memory for processing. A lot of emerging heterogeneous processing entities require cache coherency. CXL addresses that by adding coherency and memory semantics (Figure 1).

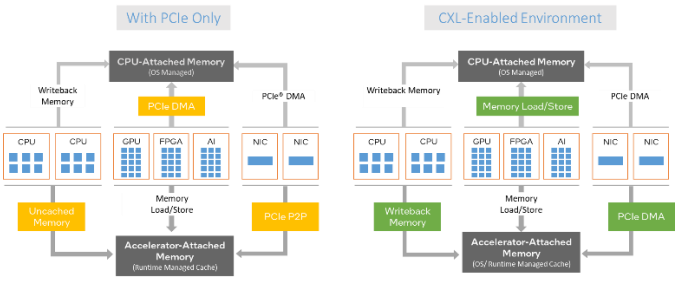

Figure 1: PCIe supports non-coherent access; CXL adds coherency and memory semantics on top of PCIe

A second aspect is the memory challenge. As shown in Figure 2, memory bandwidth per core is decreasing even as we increase the number of DDR channels per socket, the cost of memory per platform is increasing, as the computation demand is exploding using ML as an example. CXL helps ameliorate this challenge as it offers an order of magnitude more bandwidth per pin due to the pin-efficiency of PCIe based SERDES (e.g., a x16 PCIe Gen6 at 64.0 GT/s offers 256 GB/s of bidirectional raw bandwidth with 64 signal pins whereas a 300-pin DDR-5 6400 offers a bidirectional bandwidth of about 50 GB/s). CXL also helps with memory tiering enabling a mix of expensive and cheap memory due to its inherent support for coherent access to hierarchical and heterogeneous memory.

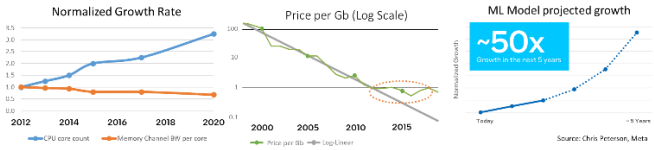

Figure 2: Memory challenge: Bandwidth and Cost challenges while demand explodes

A third challenge in the cloud infrastructure today is stranded or underutilized resources, resulting in higher cost, higher power, and lower performance. Resources such as memory, accelerators, or any I/O device connected to CPUs using tightly couple load-store interconnects such as PCIe, coherent links, DDR buses belong to that node or coherency domain. There was no mechanism using the tightly coupled load-store accesses to pool these resources to meet the elasticity in demand in the cloud infrastructure. Consequently, system designers have to over-provision these resources at system build time to meet the peak demand, resulting in stranded resources which costs money and power. For example, if a system has 2 TB of memory and an application needs more than 2 TB, it cannot go to another system in the same rack and borrow memory to execute the task. CXL solves this problem by enabling the load-store semantics to extend across multiple nodes while providing the quality of service and isolation guarantees in both the 2nd and 3rd generation of CXL specification (CXL 2.0 and CXL 3.0).

The first generation of CXL specification (CXL 1.0) was released in March 2019, followed by an enhanced version with compliance tests added (CXL 1.1) in September 2019. The first generation only supported direct connect between the CPU and a CXL device. The second-generation CXL specification (CXL 2.0) was released in November 2020 and the third generation specification (CXL 3.0) was released in August 2020. CXL has evolved in a fully backward compatible manner. Hence in any system one can have a mix of CXL 1.0/1.1, CXL 2.0, and CXL 3.0 components and they will interoperate seamlessly.

III. CXL 1.0/ 1.1 Specification: Direct Connect between Host Processor and CXL Device on a Single Node

CXL supports dynamic multiplexing of I/O (CXL.io, based on PCIe), caching (CXL.cache) and memory (CXL.mem) semantics on PCIe PHY at 32.0 GT/s data rate using 68-Byte units called Flits [1, 2, 8, 9]. CXL maintains a unified, coherent memory space between the CPU (host processor) and CXL device(s). The CPU is primarily responsible for coherency management. This helps reduce device cost and complexity, as well as the overhead traditionally associated with managing coherency. Unified coherent memory allows both the CPU and device to communicate with higher performance and reduced software stack complexity compared to the bulk transfer of PCIe.

CXL supports x16, x8, and x4 link widths natively and x2 and x1 widths in degraded mode. In addition to the target 32.0 GT/s data rate, CXL 1.0 also supports 16.0 GT/s and 8.0 GT/s data rates in degraded mode. CXL offers full interoperability with PCIe at the physical slot level on a platform since it uses the PCIe stack. A CXL device starts link training with PCIe Gen 1 Data Rate of 2.5 GT/s and negotiates CXL as the operating protocol using the alternate protocol negotiation mechanism defined in PCIe specification [7], if its link partner is capable of supporting CXL. Leveraging the PCIe infrastructure makes it easy for devices and platforms to adopt CXL without having to design and validate the PHY, channel, any channel extension devices such as Retimers, or the upper layers of PCIe, including the software stack.

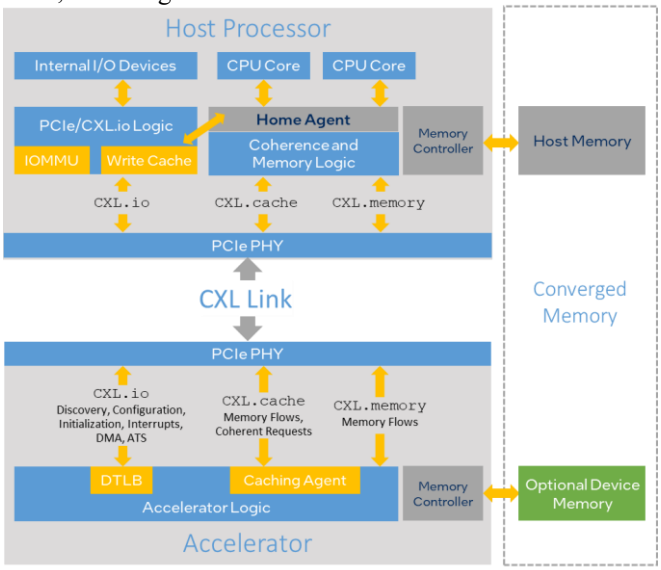

Figure 3: Coherency and Memory Semantics of CXL on top of PCIe

Figure 3 illustrates CXL's multi-protocol support. CXL.io support is mandatory for all usages, while the CXL.cache and CXL.mem are optional and usage specific. An accelerator (Type-1 device) without any attached memory would, for example, choose to implement only the CXL.io and CXL.cache protocols, while accelerators with attached memory (Type-2 device) would include support for all three. Similarly, a memory device (Type-3 device) would implement only the CXL.io and CXL.mem protocols.

The CXL.io protocol is based on PCIe and is used for various discovery and reporting functions as well as direct memory access (DMA) using non-coherent load-store semantics enforcing the producer-consumer ordering model [7, 8, 9]. The PCIe software infrastructure is expected to be reused with the device driver making the necessary enhancements to take advantage of the new capabilities such as CXL.cache and CXL.mem. CXL.cache enables a device to cache data from the host memory. CXL.cache employs a simple request and response protocol. CXL.mem allows a host processor to access memory attached to a CXL device. The CXL.mem transactions are simple memory load and store transactions that run downstream from the host processor which takes care of all coherency flows.

Low latency is a key attribute of CXL for CXL.cache and CXL.mem accesses. That is the reason for muxing these protocols at the PHY level (Figure 3). In addition, CXL Flits (68B) are optimized for the cache line transfer size of 64B with 2B for cyclic redundancy check (CRC) and 2B for the PHY layer framing.

CXL protocol is asymmetric to minimize the coherency overhead on devices. It also ensures that CXL evolves in a fully backward compatible manner. CXL adopts the MESI (Modified, Exclusive, Shared, Invalid) coherency protocol [2] with about a dozen commands of cache line requests including snoops and write backs and the associated responses, which are expected to be invariant through the future evolution of CXL. CXL does not burden device implementations with the complexity of a symmetric cache coherency protocol which is very dependent on the underlying micro-architecture. The complexity of implementing the home agent functionality belongs to the host processor which needs to implement that functionality anyway to orchestrate coherency between cores as well as different agents, including the PCIe root-ports. This enables PCIe devices to add CXL.cache and CXL.mem easily on their existing designs while protecting their investments with backwards compatibility of CXL going forward.

IV. CXL 2.0: Switching and Resource Pooling across Nodes

CXL 2.0 built on CXL 1.1 by introducing four major areas: pooling of devices across multiple domains (nodes), switching for fan-out, support for persistent memory, and security. One of the key contributions of CXL 2.0 is it is the first for a load-store interconnect to extend from the node level to the rack level.

CXL 2.0 supports pooling of multiple logical devices (MLD) as well as single logical device (SLD) with (or without) the help of a CXL switch connected to several Hosts (Root Ports). MLD is defined for Type-3 memory only. An MLD device can have multiple upstream ports to directly connect to hosts, without requiring a switch. Pooling of resources (both accelerators and memory) is critical to minimize stranded resources in a cloud infrastructure. Suppose a server temporarily needs an additional FPGA and a GP-GPU for a task it is executing. It can request for those resources from the resource manager in the rack and obtain those, if available, and later relinquish those resources when the task using them completes. Similarly, memory can be flexibly allocated and deallocated to different servers depending on their dynamic need. The addition or removal of these pooled resources follows a standard managed hot-plug flow defined in CXL 2.0

specification. In Figure 4, each color in the Host (H) represents a domain or node or server which defines a hierarchy. A CXL 2.0 switch can handle multiple domains (up to 16 of such hierarchies may reach any one MLD). CXL 2.0 defines a standardized fabric manager for the switch as well as MLD device to ensure that users have the same experience while pooling, independent of the type of device, host, switch or the usage models they have.

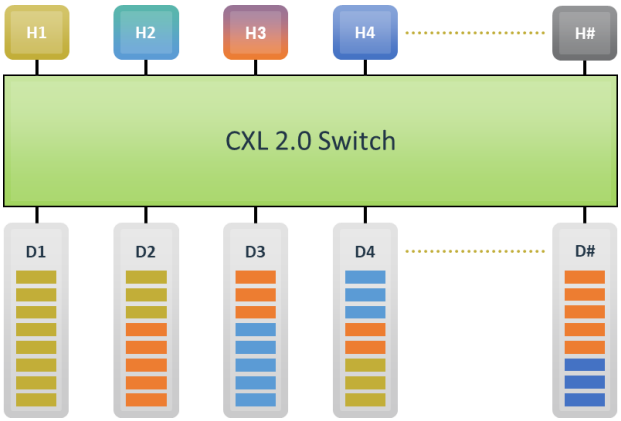

Figure 4: Pooling of resources across multiple hosts (servers) using CXL 2.0 [8, 9]- the colors indicate which host(s) the resource is currently assigned to

Recent innovations in non-volatile memory, have resulted in those approaching DRAM like latency and bandwidth characteristics, while possessing the advantages of low cost, high capacity, and persistence. CXL 2.0 defines architected flows to ensure persistence to memory [3, 4].

Security is essential for any technology to be successful, considering the ever-increasing sophistication and pervasiveness of vulnerability attacks. CXL, in collaboration with other industry-standard bodies such as PCI-SIG (Peripheral Component Interconnect, Special Interest Group) and DMTF (Distributed Management Task Force), is ensuring a seamless user experience while providing the best security mechanisms. CXL 2.0 enabled Link encryption that works seamlessly with existing security mechanisms such as device TLB.

V. CXL 3.0: Composable Systems with Fabric Topology Support

CXL 3.0 enables dynamically composable systems where one can dynamically assign resources to independent servers with load-store communication between them for collaborative computing at a much larger scale, extending to the pod level. It offers expanded capabilities such as fabric topology with fabric attached memory, memory sharing in addition to pooling, peer-to-peer memory access with the host involved to resolve coherency conflicts only if they arise, multi-level switching, higher bi-section bandwidth by removing the constraint of tree topology, and above all doubling of bandwidth by doubling the data rate while keeping latency flat.

CXL 3.0 is based on PCIe 6.0 technology and is fully backward compatible with prior generations of CXL. CXL 3.0 doubles the transfer rate to 64GT/s. This allows for aggregate raw bandwidth of up to 256GB/s for x16 width link. For low-latency transfers, CXL 3.0 leverages PCIe 6.0's combination of lightweight Forward Error Correction (FEC) and strong CRC for error free transmission with 256B flits on PAM-4 signaling to achieve 64GT/s [6, 11]. CXL 3.0, however, goes further in introducing a latency-optimized flit variant to further reduce 2-5ns of latency by breaking up the CRC in 128B sub-flit granular transfers to mitigate store-and-forward overheads in the physical layer [5, 12]. The large Flit Size combined with doubling of bandwidth enables several protocol enhancements which unlocks multiple use-cases, a few of which we'll briefly describe here.

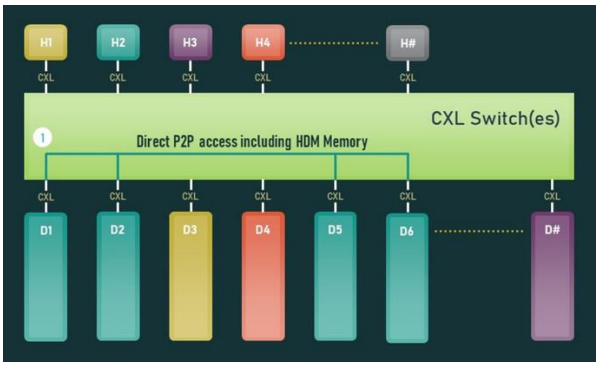

Figure 5: CXL3's protocol enhancements to enable direct P2P to memory hosted by devices: Unordered I/O and Back-Invalidate

Unordered I/O (UIO) was introduced in CXL.io with CXL 3.0. UIO is always sent on a Virtual Channel that is non-0 (e.g., VC1) with no ordering in the interconnect within or across transactions across the 3 flow-control classes (FCs) in the VC; UIO Memory Writes are placed in "P" FC class, UIO Memory Reads in the "NP" FC class and the 3 flavors of UIO completions (UIO Write Completion, UIO Read completion with data, and UIO Read completion without data) are placed in the "C" FC class. The producer-consumer ordering model is enforced by the producer since it gets an explicit completion for each UIO Write. Due to no ordering, transactions need not follow a fixed path (and hence tree topology is not required) for any source-destination pair – we can support multiple paths to support the fabric topology.

The second flavor of new transactions is the Back Invalidate (BI - both request and completions) which are added as a new message classes in each direction (BI-Request BI-Response) to the CXL.Mem protocol. Host-managed Device Memory (HDM – that is device memory that is mapped to the cacheable/ write-back region) in Type-2/ Type-3 device can be directly accessed by other devices using peer-to-peer (p2p) UIO accesses without going through the host, as shown in Figure 5. These accesses are referred to as "I/O Coherent" since they don't enable a device to cache the HDM memory but enable the device to access the location coherently each time. Prior to serving the p2p access, the memory controller in the Type-2/Type-3 device checks if the directory (or snoop filter) state of the cache line enables for the transaction to complete (e.g., a UIO read request is to a cache line that is in I or S state or a UIO write request is to a cache line that is in I state). If the request can complete, the device completes the request and returns the UIO completion to the source. If there is a coherency conflict (e.g., a UIO Read to a cache line in E state), the memory controller sets aside the request and

issues a Back Invalidate (BI) request to the host using the BI-Req channel upstream. The host resolves the coherency conflict and provides a response (BI-Rsp in downstream) after which the memory controller in the Type-2/3 device completes the UIO request and issues the UIO completion to the source. Additionally, with the enhanced coherency of BI, a Type-2 device can implement a Snoop Filter for HDM address ranges which allows it to map and manage larger amounts of memory more efficiently than before. Any capacity miss in the snoop filter results in an eviction of a prior entry with the invocation of the BI flows to make room for the new request.

CXL 3.0 also supports multi-level switching. It provides major enhancements to memory pooling due to the large number of devices that can be pooled and the large number of hosts (up to 4095) a Type-3 device can be pooled.

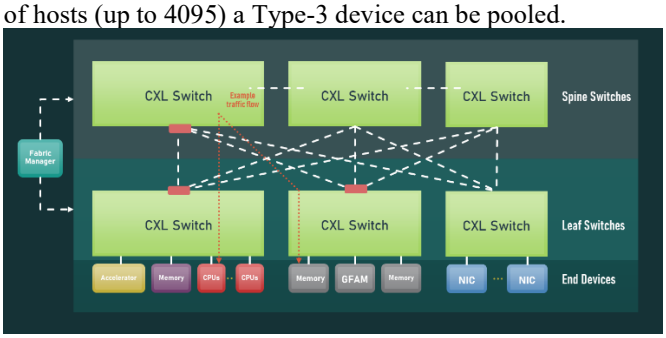

Figure 6: Fabric topology and multi-pathing supported by CXL 3.0

In addition to memory pooling, we introduce the concept of memory sharing coherently among nodes with CXL 3.0. Memory sharing is the ability of CXL-attached memory to be coherently shared across hosts using hardware coherency. Thus, unlike memory pooling, memory sharing allows the a given region of memory to be simultaneously accessible by more than one host and still guarantee that every host sees the most up to date data at that location, without the need for software-managed coordination. This allows system designers to build large clusters of machines to solve large problems through shared memory constructs. An example of a shared and pooled memory topology is shown in Figure 6. A Type-3 device supporting hardware-based shared coherent memory maintains a directory containing the identity of the hosts that have the cache line and uses BI flows to orchestrate cache coherency across hosts, when needed.

CXL 3.0 supports fabric capabilities, which removes the limitation of traditional tree topology that PCIe and previous CXL generations could only support. An example non-tree topology of a CXL fabric supporting multiple paths is shown in Figure 6. The CXL fabric can support up to 4095 nodes that can communicate with each other using a new scalable addressing mechanism called Port Based Routing (PBR) [5, 6]. Here, a node can be a CPU Host, a CXL accelerator with or without memory, a PCIe device or a Global Fabric Attached Memory (GFAM) device. A GFAM device is like a traditional CXL Type-3 device, except it can be accessed by multiple nodes (up to 4095) in flexible ways using port-based routing. Thus, CXL 3.0 enables constructing powerful composable systems comprising of compute and memory elements arranged to suffice the needs of particular workloads with coherent shared memory based message-passing between them.

VI. Conclusions

CXL technology has come a long way with 3 generations of evolution in the short 4 years of its existence. It has already been deployed in volume platforms [8, 9]. A lot of innovative usages with pooling and management of resources are being explored [10, 13, 14]. All these will lead to the imminent wide deployment in cloud infrastructure on a small set of nodes. Deployment to larger number of nodes will follow once the learnings from the initial volume deployment is comprehended. From a technology point of view, we expect the next innovations to be in finer-grain pooling of accelerators as well as smart I/O devices for cost and power benefits. We also expect to see a lot of innovations in fine-grained dynamic quality or service management by hardware. Keeping the system running without poisoning the data to work around dynamic link failures would be another challenge the industry needs to overcome if we use CXL as a large scale-out fabric.

We expect CXL technology evolution to continue for multiple decades, incorporating the learnings from real products as time goes on and accounting for new and emerging usages, like PCIe which is still going strong on its evolution after twenty years.

References

[1] D. Das Sharma, "Compute Express Link", white paper, Compute Express Link Consortium, March 2019

[2] CXL Consortium, "Compute Express Link 1.1 Specification", July 2, 2019

[3] CXL Consortium, "Compute Express Link 2.0 Specification", Sept 9, 2020

[4] D. Das Sharma and S. Tavallaei, "Compute Express Link™ 2.0", white paper, CXL Consortium, Nov 2020

[5] CXL Consortium, "Compute Express Link 3.0 Specification", www.computeexpresslink.org, Aug 2022

[6] D. Das Sharma and I. Agarwal, "Compute Express Link 3.0", white paper, CXL Consortium, Aug 2022

[7] PCI-SIG, "PCI Express® Base Specification Revision 6.0, Version 1.0", Jan, 2022

[8] D. Das Sharma, "Compute Express Link®: Enabling Heterogeneous Data-Centric Computing With Heterogeneous Memory Hierarchy", IEEE Micro, Mar-Apr 2023

[9] D. Das Sharma, "Novel Composable and Scale-out Architectures using Compute Express Link™", IEEE Micro, Special Issue on Interconnects, Special Issue on Hot Interconnects 29, Mar-Apr 2023

[10] D. Das Sharma, "The Compute Express Link (CXL)* open standard is changing the game for cloud computing", Opening Keynote in IEEE Hot Interconnects, 2021

[11] D. Das Sharma, ""A Low-Latency and Low-Power Approach for Coherency and Memory Protocols on PCI Express 6.0 PHY at 64.0 GT/s with PAM-4 Signaling"", IEEE Micro, Mar/ Apr 2022

[12] D. Das Sharma, ""PCI Express 6.0 Specification: A Low-Latency, High-Bandwidth, High-Reliability, and Cost-Effective Interconnect With 64.0 GT/s PAM-4 Signaling"", IEEE Micro, Jan/ Feb 2021

[13] H. AL Maruf et al, "TPP: Transparent Page Placement for CXL-Enabled Tiered Memory", June 2022

[14] H. Li et al, "Pond: CXL-Based Memory Pooling Systems for Cloud Platforms", ASPLOS '23, 2023