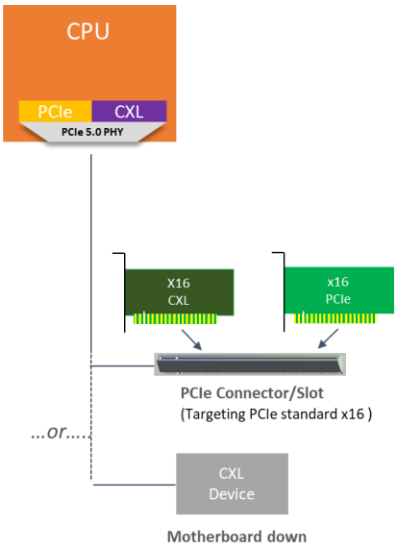
Compute Express Link

Dr. Debendra Das Sharma
Intel Fellow and Director, I/O Technology and Standards
Promoter Member, Compute Express Link

Compute Express Link (CXL) is an open industry standard interconnect offering high-bandwidth, low-latency connectivity between host processor and devices such as accelerators, memory buffers, and smart I/O devices. CXL is based on the PCI Express® (PCIe®) 5.0 physical layer infrastructure. It is designed to address the growing high-performance computational workloads by supporting heterogeneous processing and memory systems with applications in Artificial Intelligence, Machine Learning, communication systems, and High Performance Computing by enabling coherency and memory semantics. This is increasingly important 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.

CXL supports dynamic multiplexing between a rich set of protocols that includes I/O (CXL.io, based on PCIe®), caching (CXL.cache) and memory (CXL.memory) semantics. CXL maintains a unified, coherent memory space between the CPU (host processor) and any memory on the attached CXL device. This allows both the CPU and device to share resources for higher performance and reduced software stack complexity. Moreover, since the CPU is primarily responsible for coherency management, it can reduce device cost and complexity, as well as overhead traditionally associated with coherency across an I/O link.

CXL runs on PCIe® PHY and supports x16, x8, and x4 link widths natively and x2 and x1 widths in degraded mode. CXL 1.0 will debut at 32 GT/s, offering 64 GB/s bandwidth in each direction. CXL 1.0 also supports 16.0 GT/s and 8.0 GT/s data rates in degraded mode. The following diagram illustrates how CXL offers full interoperability with PCIe since it uses the PCIe stack. A CXL device starts link training in PCIe Gen 1 Data Rate and negotiates CXL as the operating protocol using the alternate protocol negotiation mechanism defined in the PCIe 5.0 specification, if its link partner is capable of supporting CXL. Leveraging the PCIe 5.0 infrastructure makes it really 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.

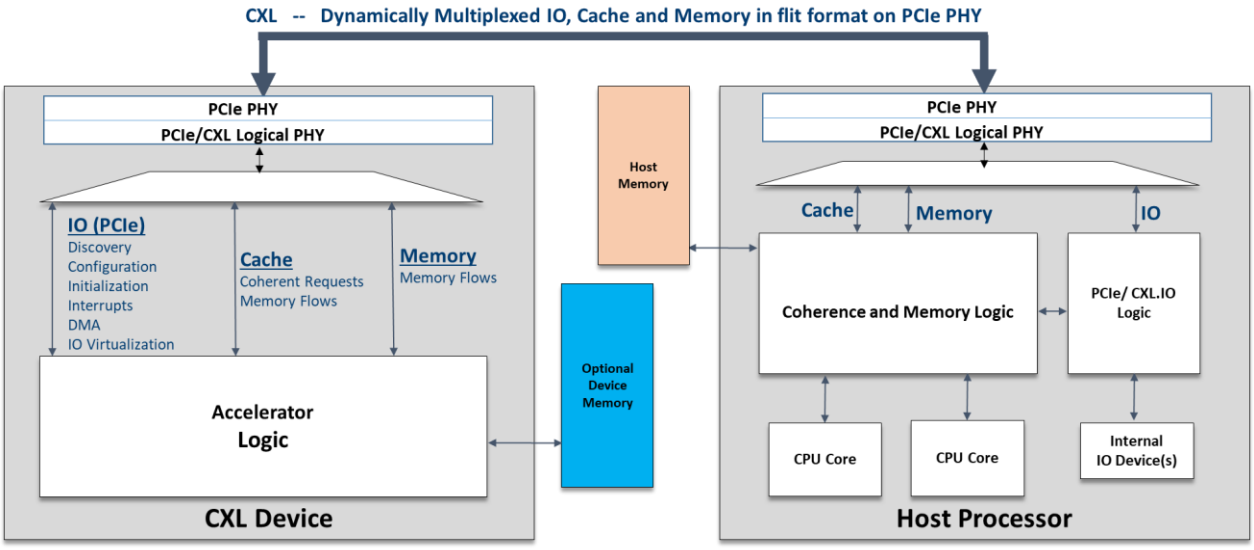


Compute Express Link has the benefit of supporting both standard PCIe devices as well as CXL devices – all on the same Link

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

- The CXL.io protocol is based on PCIe and is used for the functions such as device discovery, configuration, initialization, I/O virtualization, and direct memory access (DMA) using non-coherent load-store, producer-consumer semantics. While we expect the PCIe software infrastructure to be reused, device driver would make the necessary enhancements to take advantage of the new capabilities such as CXL.cache and CXL.memory and the system software to program the new set of registers associated with the new capabilities.
- CXL.cache enables a device to cache data from the host memory, employing a simple request and response protocol. The host processor manages coherency of data cached at the device by means of snoop messages.
- CXL.memory allows a host processor to access memory attached to a CXL device. CXL.memory transactions are simple memory load and store transactions that run downstream from the host processor which takes care of all the associated coherency flows.

Symmetric cache coherency protocols between host processors are complex. Different architectures take very different approaches to coherency, optimizing for the underlying micro-architecture, making it a challenge for the broader industry to adopt and enforcing backwards compatibility. CXL has addressed these challenges by moving the complexity of implementing the home agent functionality to the host processor. This enables PCIe devices to add CXL.cache and CXL.memory easily while protecting their investments with backwards compatibility of CXL going forward.



CXL multiplexes different protocols at the PCIe PHY layer. The unit of transfer for each protocol is flit-based. The CXL.cache and CXL.memory protocols are natively flit based, with CRC protecting each fixed-sized flit. CXL.io is packet-based, using the same Transaction Layer Packets (TLPs) and Data Link Layer Packets (DLLPs) as PCIe. The TLP/DLLPs are overlaid on the payload part of the CXL flit. CXL defines policies to deliver the desired quality of service (QoS) across different protocol stacks. The multiplexing of protocols at the PHY level ensures that latency-sensitive protocols such as CXL.cache and CXL.memory have identical low-latency as a native CPU to CPU symmetric coherency link. The CXL 1.0 specification defines an upper bound on the pin-to-pin response time for these latency-sensitive protocols to ensure that platform performance is not adversely impacted by a wide variation in latency between different devices implementing coherency and memory semantics.

In addition to the accelerator and memory expansion devices in a platform, CXL is expected to evolve to support new usages. These new usage models will be discussed and comprehended in the CXL 2.0 specification, which will start development in early Q2 2019. The CXL Consortium is committed to developing this open standard in a collaborative manner and welcomes active participation of the broader industry.

Acknowledgements: The author wants to thank Mike Ferron-Jones, Marcus Yam, Jim Pappas, Tom Stachura, and Steve Van Doren for their help in putting together this white paper.