Compute Express Link® (CXL): A Coherent Interface for Ultra-High-Speed Transfers

Kurt Lender
CXL® Marketing Work Group
Ecosystem Enabling Manager
Intel Corporation
Agenda

• Industry Need for CXL
• Overview of CXL
• CXL Features / Benefits
• CXL Use Cases
• Bringing CXL to the Industry / How to Contribute
• Summary
INDUSTRY MEGA TRENDS

PROLIFERATION OF CLOUD COMPUTING

GROWTH OF AI & ANALYTICS

CLOUDIFICATION OF THE NETWORK & EDGE
Why a New Class of Interconnect?

Extend PCIe for heterogeneous computing and server disaggregation usages

- Create shared memory pools with efficient access mechanisms
- Enhance movement of operands and results between accelerators and target devices
- Enable efficient resource sharing
- Significant latency reduction to enable disaggregated memory

Accelerator-Attached Memory (Runtime managed cache)

CPU-attached Memory (OS Managed)

PCIe DMA

UnCached Memory

Writeback Memory

Memory Load/Store

NIC

PCIe P2P
Overview of CXL
Introducing CXL

**Processor Interconnect**

- Open industry standard
- High-bandwidth, low-latency
- Coherent interface
- Leverages PCI Express®
- Targets → High-performance computational workloads - AI, ML, HPC, & Comms

CXL: a new class of interconnect for device connectivity
What is CXL?

- CXL is an alternate protocol that runs across the standard PCIe physical layer
- CXL uses a flexible processor port that can auto-negotiate to either the standard PCIe transaction protocol or the alternate CXL transaction protocols
- First generation CXL aligns to 32 Gbps PCIe 5.0
- CXL usages expected to be key driver for an aggressive timeline to PCIe 6.0
The CXL transaction layer is comprised of 3 dynamically multiplexed sub-protocols on a single link:

- **CXL.io** - Discovery, configuration, register access, interrupts, etc.
- **CXL.cache** - Device access to processor memory
- **CXL.memory** - Processor access to device attached memory
CXL Features / Benefits
All 3 representative usages have latency critical elements:
- CXL.cache
- CXL.memory
- CXL.io

CXL cache and memory stack is optimized for latency:
- Separate transaction and link layer from IO
- Fixed message framing

CXL io flows pass through a stack that is largely identical a standard PCIe stack:
- Dynamic framing
- Transaction Layer Packet (TLP)/Data Link Layer Packet (DLLP) encapsulated in CXL flits
All 3 representative usages have latency critical elements:
- CXL.cache
- CXL.memory
- CXL.io

CXL cache and memory stack is optimized for latency:
- Separate transaction and link layer from IO
- Fixed message framing

CXL io flows pass through a stack that is largely identical a standard PCIe stack:
- Dynamic framing
- Transaction Layer Packet (TLP)/Data Link Layer Packet (DLLP) encapsulated in CXL flits
CXL’s Protocol Asymmetry

**CXL Model - Asymmetric Protocol**

- **Accelerator**
  - Accelerator Engine
  - Accelerator Caching Agent
  - Accelerator Home Agent
  - Memory Agent

- **CPU**
  - Core
  - CCI Caching Agent
  - CCI Home Agent
  - Memory Agent

- **Cache Core**
  - Biased Coherence Bypass

- **CXL/CCI Caching Agent**
  - CXL/CCI Home Agent
  - Memory Agent

**CCI Model - Symmetric CCI Protocol**

- **Accelerator**
  - Accelerator Engine
  - Accelerator Caching Agent
  - Accelerator Home Agent
  - Memory Agent

- **CPU**
  - Core
  - CCI Caching Agent
  - CCI Home Agent
  - Memory Agent

- **Cache**

**CXL key advantages:**
- Avoid protocol interoperability hurdles/roadblocks
- Enable devices across multiple segments (e.g. client / server)
- Enable Memory buffer with no coherency burden
- Simpler, processor independent device development

CCI - Cache Coherent Interconnect
CXL’s Coherence Bias

Device Bias

Accelerator
Cache

CPU + IO
Coherence
CA + HA
Uncached Flow

Critical access class for accelerators is “device engine to device memory”

“Coherence Bias” allows a device engine to access its memory coherently without visiting the processor

Coherency Guaranteed

Host Bias

Accelerator
Cache

CPU + IO
Coherence
CA + HA

Two driver managed modes or “Biases”

Both biases guaranteed correct/coherent

Guarantee applies even when software bugs or speculative accesses unexpectedly access device memory in the “Device Bias” state.

HOST BIAS: pages being used by the host or shared between host and device
DEVICE BIAS: pages being used exclusively by the device
Representative CXL Usages

**Caching Devices / Accelerators**
- Usages:
  - PGAS NIC
  - NIC atomics
- Protocols:
  - CXL.io
  - CXL.cache

**Accelerators with Memory**
- Usages:
  - GPU
  - Dense Computation
- Protocols:
  - CXL.io
  - CXL.cache
  - CXL.memory

**Memory Buffers**
- Usages:
  - Memory BW expansion
  - Memory capacity expansion
  - Storage Class Memory
- Protocols:
  - CXL.io
  - CXL.mem

Diagram:
- Accelerator
- NIC
- Cache
- Processor
- DDR
- HBM
- Memory Buffer
CXL enables a more fluid and flexible memory model
Single, common, memory address space across processors and devices

- More efficient population and update of operands
- More efficient extraction of results
- Memory resource “borrowing”
- User/Kernel level data access and data movement
- Low latency to memory, host to device and device to host

CPU-attached Memory (OS Managed)

Accelerator-Attached Memory (Runtime managed cache)
Bringing CXL to the Industry
Broad Industry support for CXL

Founding Promoters

CXL consortium - Currently 75 companies and growing

www.ComputeExpressLink.org
Call to Action: CXL Consortium – Get Involved

Contributors

Participate in Work Groups
- Protocol
- PHY
- System
- Software
- Compliance
- Marketing

Adopters

Join

Download Specification

Call to Action: Join and Contribute Now!
www.ComputeExpressLink.org

All information Confidential (CXL Consortium)
CXL Summary

• Coherent Interface
  → Leverages PCIe with 3 mix-and-match protocols

• Low Latency
  → Cache and Mem targeted at near CPU cache coherent latency

• Asymmetric Architecture
  → Eases burdens of cache coherent interface designs

• Open Industry Standard
  → With growing broad industry support

CXL has the right features and architecture to enable a broad, open eco-system for heterogeneous computing and server disaggregation
Thank You!