CXL™ 2.0: A High-Speed Interconnect for Persistent Memory Challenges

Presented by Andy Rudoff, PMem SW Architect, Intel
Persistent Memory Today

Connecting to the Memory Bus
Benefits of Persistent Memory

- **Moves Persistent Memory from Controller to CXL**
- **Enables Standardized Management of the Memory and Interface**
- **Supports a Wide Variety of Industry Form Factors**

<table>
<thead>
<tr>
<th>Memory</th>
<th>Latency (nanoseconds)</th>
<th>CXL 2.0</th>
<th>Persistent Memory</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>$10^0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAM CXL 1.1/1.0</td>
<td>$10^3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CXL 2.0</td>
<td></td>
<td>$10^2$–$10^3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance SSD</td>
<td>$10^4$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity SSD</td>
<td>$10^3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDD</td>
<td>$10^6$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CXL + PM Fills the Gap!
The ACPI “NVDIMM” Framework
The ACPI “NVDIMM” Framework

The SNIA NVM Programming Model

NVDIMM Firmware Interface Table (ACPI 6.0)
The ACPI “NVDIMM” Framework

The SNIA NVM Programming Model

Device Specific Methods
The ACPI “NVDIMM” Framework

The SNIA NVM Programming Model

- Management UI
- Application
- Application
- Application
- Management Library
- Standard Raw Device Access
- Standard File API
- Local/Store
- “DAX”
- File System
- pmem-Aware File System
- Generic NVDIMM Driver
- Persistent Memory

BIOS

NFIT

_DSM
Compute Express Link

A New Class of Interconnect
CXL Consortium

CXL Board of Directors

Open Industry Standard for Cache Coherent Interconnect

150+ Member Companies
Compute Express Link

- **CXL 1.1**
  - June 2019
- **CXL 2.0**
  - Nov 2020
  - 1.1 Compatible
  - Adds pmem

Our Focus

Introducing CXL

- **Open industry standard for high bandwidth, low-latency interconnect**
- **Connectivity between host processor and accelerators/ memory device/ smart NIC**
- **Addresses high-performance computational workloads across AI, ML, HPC, and Comms segments**
  - Heterogeneous processing: scalar, vector, matrix, spatial architectures spanning CPU, GPU, FPGA
  - Memory device connectivity
  - PCIe PHY completely leveraged with additional latency optimization
  - Dynamic multiplexing of 3 protocols
- **Based on PCIe® 5.0 PHY infrastructure**
  - Leverages channel, retimers, PHY, Logical, Protocols
  - CXL.io – I/O semantics, similar to PCIe - mandatory
  - CXL.cache – Caching Semantics – optional
  - CXL.memory – Memory semantics - optional
Representative CXL Usages

Caching Devices / Accelerators

Usages:
- PGAS NIC
- NIC atomics

Protocols:
- CXL.io
- CXL.cache

Accelerators with Memory

Usages:
- GPU
- FPGA
- Dense Computation

Protocols:
- CXL.io
- CXL.cache
- CXL.memory

Memory Buffers

Usages:
- Memory BW expansion
- Memory capacity expansion
- 2LM

Protocols:
- CXL.io
- CXL.mem

(Type 1 Device)

(type 2 Device)

(Type 3 Device)
Where PMem Fits In

Caching Devices / Accelerators

Usages:
- PGAS NIC
- NIC atomics

Protocols:
- CXL.io
- CXL.cache

Accelerators with Memory

Usages:
- GPU
- FPGA
- Dense Computation

Protocols:
- CXL.io
- CXL.cache
- CXL.memory

Memory Buffers

Usages:
- Memory BW expansion
- Memory capacity expansion
- 2LM

Protocols:
- CXL.io
- CXL.mem

(Type 1 Device)

(Type 2 Device)

(Type 3 Device)
Adding PMem to CXL

The CXL 2.0 Specification
CXL 2.0 Changes for PMem

- Most changes should apply to all memory types
  - Minimize PMem-specific changes, rest apply to volatile memory too
- PCIe enumeration
  - NFIT isn’t used for CXL devices (they aren’t NVDIMMs!)
  - Leverage PCIe frameworks, including hot plug
- MMIO registers
  - Mailbox interface, etc.
- Command Interface
  - Was vendor-private for NVDIMMs
- Driver Writers Guide (delivered as separate doc)
- Minor changes to external specs like ACPI/UEFI
Mailbox Commands

- Was vendor private
- Standards are a double-edged sword
  - Generic Drivers
  - Committee visit for every change
- Learnings from NVDIMMs helped
  - Leverage what worked
  - Fix pain points
- Some commands apply to all CXL →

### CXL Device Command Opcodes

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Required *</th>
<th>Input Payload Size (B)</th>
<th>Output Payload Size (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01h Events</td>
<td>00h</td>
<td>Get Event Records</td>
<td>0100h M 1 20h+</td>
</tr>
<tr>
<td></td>
<td>01h</td>
<td>Clear Event Records</td>
<td>0101h M 8+ 0</td>
</tr>
<tr>
<td></td>
<td>02h</td>
<td>Get Event Interrupt</td>
<td>0102h M 0 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Policy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03h</td>
<td>Set Event Interrupt</td>
<td>0103h M 4 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Policy</td>
<td></td>
</tr>
<tr>
<td>02h Firmware Update</td>
<td>00h</td>
<td>Get FW Info</td>
<td>0200h O 0 50h</td>
</tr>
<tr>
<td></td>
<td>01h</td>
<td>Transfer FW</td>
<td>0201h O 80h+ 0</td>
</tr>
<tr>
<td></td>
<td>02h</td>
<td>Activate FW</td>
<td>0202h O 2 0</td>
</tr>
<tr>
<td>03h Timestamp</td>
<td>00h</td>
<td>Get Timestamp</td>
<td>0300h O 0 8</td>
</tr>
<tr>
<td></td>
<td>01h</td>
<td>Set Timestamp</td>
<td>0301h O 8 0</td>
</tr>
<tr>
<td>04h Logs</td>
<td>00h</td>
<td>Get Supported Logs</td>
<td>0400h M 0 8+</td>
</tr>
<tr>
<td></td>
<td>01h</td>
<td>Get Log</td>
<td>0401h M 18h 0</td>
</tr>
</tbody>
</table>
Memory Device Commands

- Most added commands →
  - DSMs are gone
    - OS uses mailbox directly
    - BIOS too
- Much complexity moved:
  - From BIOS to OS
- Allows generic:
  - BIOS
  - OS Drivers

### Memory Device Commands

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Required</th>
<th>Input Payload Size (B)</th>
<th>Output Payload Size (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify</td>
<td>M</td>
<td>0</td>
<td>42h</td>
</tr>
<tr>
<td>Get Partitions Info</td>
<td>O</td>
<td>0</td>
<td>20h</td>
</tr>
<tr>
<td>Get LSA</td>
<td>PM</td>
<td>B</td>
<td>0+</td>
</tr>
<tr>
<td>Get Health Info</td>
<td>N</td>
<td>0</td>
<td>12h</td>
</tr>
<tr>
<td>Get Alert Configuration</td>
<td>M</td>
<td>0</td>
<td>14h</td>
</tr>
<tr>
<td>Get Shutdown State</td>
<td>PM</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Get Scan Media Capabilities</td>
<td>PM</td>
<td>10h</td>
<td>4</td>
</tr>
<tr>
<td>Get Scan Media Results</td>
<td>PM</td>
<td>11h</td>
<td>0</td>
</tr>
<tr>
<td>Sanitize</td>
<td>O</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Secure Erase</td>
<td>O</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example: Identify Memory Device

### Identify Memory Device Output Payload

<table>
<thead>
<tr>
<th>Byte Offset</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10h</td>
<td><strong>FW Revision</strong>: Contains the revision of the active FW formatted as an ASCII string. This is the same information that may be retrieved with the Get FW Info command.</td>
</tr>
<tr>
<td>10h</td>
<td>8</td>
<td><strong>Total Capacity</strong>: This field indicates the total usable capacity of the device. Expressed in multiples of 256 MB. Total device usable capacity is divided between volatile only capacity, persistent only capacity, and capacity that can be either volatile or persistent. Total Capacity shall be greater than or equal to the sum of Volatile Only Capacity and Persistent Only Capacity.</td>
</tr>
<tr>
<td>18h</td>
<td>8</td>
<td><strong>Volatile Only Capacity</strong>: This field indicates the total usable capacity of the device that may only be used as volatile memory. Expressed in multiples of 256 MB.</td>
</tr>
<tr>
<td>20h</td>
<td>8</td>
<td><strong>Persistent Only Capacity</strong>: This field indicates the total usable capacity of the device that may only be used as persistent memory. Expressed in multiples of 256 MB.</td>
</tr>
<tr>
<td>28h</td>
<td>8</td>
<td><strong>Partition Alignment</strong>: If the device has capacity that may be used either as volatile memory or persistent memory, this field indicates the partition alignment size. Expressed in multiples of 256 MB. Partitionable capacity is equal to Total Capacity - Volatile Only Capacity - Persistent Only Capacity. If 0, the device doesn’t support partitioning the capacity into both volatile and persistent capacity.</td>
</tr>
</tbody>
</table>

- **Informational Event Log Size**: The number of events the device can
Driver Writers Guide

- CXL 2.0 Spec defines commands
  - Can be terse, spec language
- Platforms decide some of the details
  - Example: BIOS on Intel platforms
- Document flows, algorithms
- Published on intel.com
The Hard Stuff

Where we spent much of our time
Interleaving

- HDM Decoders
  - Allow interleaving across devices
  - New to PCIe: interleave sets

- Important concept for PMem
  - For volatile memory, changing the interleave may impact performance
  - For PMem, changing the interleave loses your data

- Label Storage Area
  - Defined in CXL 2.0 spec
  - Provides region (interleave set) and namespace configuration
Label Example

Host Bridge

2 way@4k

Switch
4 way@1k

Device 0
Device 1
Device 2
Device 3

Device 4
Device 5
Device 6
Device 7

<table>
<thead>
<tr>
<th>Label</th>
<th>Label</th>
<th>Label</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>N 8</td>
<td>N 8</td>
<td>N 8</td>
<td>N 8</td>
</tr>
<tr>
<td>Pos 0</td>
<td>Pos 1</td>
<td>Pos 2</td>
<td>Pos 3</td>
</tr>
<tr>
<td>IG 1k</td>
<td>IG 1k</td>
<td>IG 1k</td>
<td>IG 1k</td>
</tr>
</tbody>
</table>

Label Example
Hot Plug

- Handled by OS
  - No BIOS in flow
- Uses “windows”
  - BIOS provided
- Flow used for PMem
  - Except boot dev
Software Enabling
Current State and Future Work
The Good News

- Momentum around CXL is Huge
  - NVDIMM enabling was a few of us in a room, updating ACPI
  - CXL is dozens of highly-engaged companies, including all major OSVs
- Preliminary generic Type 3 CXL Driver already upstream in Linux
  - Orchestrated by NVDIMM framework maintainer Dan Williams
- QEMU patches emulating CXL Type 3 devices posted
  - Written by Ben Widawsky, provided a development platform for the driver
- Cross-company, cross committee collaboration on specifications
  - Prevents messy collisions from different implementation decisions
The Challenges

- Memory bus, ACPI, NFIT was essentially done
  - OS versions that support NVDIMMs, continue to work across generations

- CXL requires a new set of drivers
  - Drivers do new things, like group into interleave sets
  - Not clear if drivers can be added to older kernels

- NVDIMM tooling → CXL tooling
  - Example: ndctl command on Linux begats cxl command
Summary

- The programming model remains the same
  - Applications written to the SNIA programming model continue to work
- CXL offers:
  - Moving PMem off the memory bus
  - Scalability (all types of memory)
  - Flexibility
- PMem on CXL specified as of CXL 2.0, published last November
  - OS enabling is emerging
Please take a moment to rate this session.

Your feedback is important to us.