CXL 2.0 + Linux + QEMU = Yes
CXL 2.0 + Linux + QEMU = Yes
Introduction

Last slide first!!!
Agenda

- Introduction
- CXL 2.0 Background
  - Linux Driver Details
- QEMU
- Future
Intro/Links

● Communications
  ○ #cxl on OFTC
  ○ linux-cxl@vger.kernel.org

● Drivers
  ○ https://git.kernel.org/pub/scm/linux/kernel/git/cxl/cxl.git/
  ○ https://gitlab.com/bwidawsk/qemu
  ○ https://github.com/pmem/run_qemu

● Userspace
  ○ https://github.com/pmem/ndctl/tree/cxl-2.0v3
  ○ https://gitlab.com/bwidawsk-cxl/cxl_rs
CXL 2.0 Background
**Challenges**

- Industry trends driving demand for faster data processing and next-gen data center performance
- Increasing demand for heterogeneous computing and server disaggregation
- Need for increased memory capacity and bandwidth
- Lack of open industry standard to address next-gen interconnect challenges

**CXL**

An open industry-supported cache-coherent interconnect for processors, memory expansion and accelerators

**Coherent Interface**

Leverages PCIe® with 3 mix-and-match protocols

**Low Latency**

Cache and Memory targeted at near CPU cache coherent latency

**Asymmetric Complexity**

Eases burdens of cache coherent interface designs
Challenges

- Industry trends driving demand for faster data processing and next-gen data center performance
- Increasing demand for heterogeneous computing and server disaggregation
- Need for increased memory capacity and bandwidth
- Lack of open industry standard to address next-gen interconnect challenges

CXL Features

- **Coherent Interface**: Leverages PCIe® with 3 mix-and-match protocols
- **Low Latency**: Cache and Memory targeted at near CPU cache coherent latency
- **Asymmetric Complexity**: Eases burdens of cache coherent interface designs

CXL
An open industry-supported cache-coherent interconnect for processors, memory expansion and accelerators
Who Cares?

CXL Board of Directors

Compute Express Link™ and CXL™ Consortium are trademarks of the Compute Express Link Consortium.
Usage Models

Caching Devices / Accelerators

- Processor
- DDR
- CXL
- Accelerator
- NIC
- Cache

USAGES
- NUMA NIC
- NUMA atomsics

Accelerators with Memory

- Processor
- DDR
- CXL
- Accelerator
- NIC
- Cache

PROTOCOLS
- CXL
- CXL.io
- CXL.cache

USAGES
- CPU-DPU
- Dense computation

Memory Buffers

- Processor
- DDR
- CXL
- Memory Buffer
- Memory
- Memory

PROTOCOLS
- CXL
- CXL.io
- CXL.memory

USAGES
- Memory SW expansion
- Memory capacity expansion
- Storage class memory
Usage Models

Caching Devices / Accelerators
- Type 1
  - DRAM
  - Processor
  - CXL
    - CXL.io
    - CXL.cache
  - Accelerator
    - NIC
  - Cache

Accelerators with Memory
- Type 2
  - DRAM
  - Processor
  - CXL
    - CXL.io
    - CXL.cache
    - CXL.memory
  - Accelerator
  - Memory
  - HBM

Memory Buffers
- Type 3
  - DRAM
  - Processor
  - CXL
    - CXL.io
    - CXL.memory
  - Memory Buffer
  - Memory
  - HBM

USAGES
- Type 1
  - MIGAS NIC
  - ML architectures

- Type 2
  - GPU/GPU
  - Dense computation

- Type 3
  - Memory I/O expansion
  - Memory capacity expansion
  - Storage class memory
Form Factors

Ex #1: EDSFF E1.S
Ex #2: EDSFF E3.S / E3.L
Ex #3: Add-in Card (AIC)

Factors

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Area vs. DDRx Server DIMM</td>
<td>Smaller</td>
<td>Larger</td>
<td>Larger (larger than E3.S/L)</td>
</tr>
<tr>
<td>Expected Max. Power Range</td>
<td>12 ~ 25W</td>
<td>25W ~ 40W (1T), 40W ~ 70W (2T)</td>
<td>Similar range compared to E3.S/L</td>
</tr>
</tbody>
</table>

Reference: snia.org
CXL Topology

- CXL 2.0 hierarchy appears like PCIe hierarchy
  - Legacy PCI SW and CXL SW sees a RP or DSP with Endpoints below
  - CXL link/interface errors are signaled to RP, not RCEC
  - Port Control Override registers prevent legacy PCIe software from unintentionally resetting the device and the link
- Interleaving
  - Cross host bridge
  - Switch
  - Device
Interleave
CXL Persistent Memory Devices

PCI and NVDIMM had a coherent byte addressable baby... with atomics.

- NVDIMM
  - Byte addressable
  - Direct mappable

- NVME
  - PCIe configurable
  - Hot swappable
• Persistent memory devices rely on System Software for provisioning and management
• CXL 2.0 introduces a standard register interface
• A generic memory device driver simplifies software enabling
• Architecture Elements
  • Defined as number of discoverable Capabilities
  • Capabilities includes Device Status and standard mailboxes, accessed via MMIO registers
  • Standardized mailbox commands that cover errors/health, alerts, partitioning, passphrases etc.
  • Allow Vendor specific extensions
Linux Driver Details
Software Responsibilities
SW Enumerable Components

- CEDT ACPI0017
- ACPI0016
- Root Port
- USP (Switch)
- DSP (Switch)
- Endpoint
- Endpoint
- Host
- ACPI0016
- ACPI0016
- ACPI0016
Linux Drivers

Enumerable
- CXL capabilities
- CXL hierarchy

cxl_port

cxl_mem

cxl_pmem

cxl_core
- bus
- mbox
- mem
- memdev
- regs

cxl_region

Enumerable
- PCI Class code
- ACPI discovery

Phase 1
- cxl_pci
- cxl_acpi

Phase 2
- LSA
cxl_core

- Maintains cxl_driver infra
- Interfaces with LIBNVDIMM
- Manages device (sysfs)
  - Services to add devices, i.e. cxl_decoder_add()
- IOCTL interface
- Common functionality
  - Mailbox controls (session layer)
  - Register mapping
cxl_pci

- Probed like a typical PCI device
  - \{ PCI_DEVICE_CLASS((PCI_CLASS_MEMORY_CXL << 8 | CXL_MEMORY_PROGIF), -0)\}
- CXL device manageability
  - Implements mailbox transport (CXL) protocol
- Enumerates CXL device for subsequent driver
  - `cxl_mem` can’t run until `cxl_pci` is done
- Attestation/Security/Whatever
CEDT ACPI0017

Host

ACPI0016

ACPI0016

ACPI0016

Root Port

USP (Switch)

DSP (Switch)

Endpoint

Endpoint

\text{cxl\_pci}
cxl_acpi

- Probed like a typical ACPI device
  - \{ "ACPI0017", (unsigned long) &native_acpi0017 },

- Platform specific CXL enumeration
  - Mostly specified in UEFI, and CXL

- ACPI0017 starts enumeration of CXL ports
  - CEDT
  - “Root level” ports (platform)
  - Hostbridges and root ports
cxl_port

- Ports are created for all components with an “upstream port”
  - Hostbridge
  - Switch
  - Endpoint
- Enumeration and control and control of decoder resources
  - Provides as a service for other drivers
cxl_mem

- connects a device enumerated with cxl_pci to CXL.mem.
  - Implements device functionality not handled by cxl_pci
- “exports” if device is CXL.mem routed and enabled
cxl_mem
For each endpoint...

- **cxl_acpi**
- **cxl_pci**
- **cxl_mem**
- **cxl_port**

Runs asynchronously

- Rescan root -> endpoint
- **cxl_core**
- **cxl_port**

Walks up to find root port. Won’t be there if cxl_acpi isn’t done.

Finds root port - **done**
Finds switch - **rescan**
Neither - **fail bind**
## Volatile vs. Persistent

<table>
<thead>
<tr>
<th></th>
<th>Persistent</th>
<th>Volatile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td>Configured by BIOS</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>BIOS configures all volatile capacities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BIOS may check PMEM devices, but will not configure regions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BIOS may configure bootable PMEM</td>
<td></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td>OS managed</td>
<td>✅</td>
</tr>
<tr>
<td></td>
<td>OS initializes persistent regions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS may create new persistent regions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manages hotplug, error, and reset sequences for both</td>
<td></td>
</tr>
<tr>
<td><strong>3</strong></td>
<td>Requires CXL.mem capability</td>
<td>✅</td>
</tr>
<tr>
<td></td>
<td>Requires CXL.mem capability</td>
<td></td>
</tr>
</tbody>
</table>
Regions

- **Region**
  - Interleave set of devices
  - Parameters (IG, HPA, etc)
  - Stored in the Label Storage Area

- **Creation**
  - Via sysfs ABI
  - Provisioned offline
    - Manufacture time

- **Responsibilities**
  - Validating region configuration
  - Programming HDM decoders
Region Validation

1) The ordering of the XHB CFMWS.InterleaveTargetList[] is fixed by System Firmware, so devices must be connected to the correct host bridge.

Verify each device has the same masked position value:

\[
\text{Device}[x].\text{RegionLabel}\cdot\text{Position} \gg (\text{CFMWS} \cdot \text{HBIG} - \text{Device}[x].\text{RegionLabel}\cdot\text{IG}) \& ((2^{\text{CFMWS} \cdot \text{ENIW}} - 1)
\]

2) The ratio of the CFMWS.HBIG to the RegionLabel.IG determines how many bits to shift the position mask to the left.

3) The CFMWS.ENIW determines how many bits of the position are masked.

Valid XHB device configuration.
Decoder Programming

512MB interleave across 2 devices @0x4c00000000

512MB linear across 2 devices @0x4c00000000
Linux Interfaces

- **Sysfs**
  - `/sys/bus/cxl/

- **IOCTL**
  - QUERY
  - SEND
  - Managed command set
  - RAW escape command

- **Future**
  - Region Creation ABI
Review Goals

1) Upstream Linux Driver
   a) 0 days of spec release (v5.12)
      i) Ease customer adoption
   b) Backportable
   c) Platform aiding hw implementation and validation
      i) Validate the spec for driver usage

2) Reusable past driver bringup
   a) infra for regression testing
   b) Virtualization

3) Scalable
   a) Community Contributions & Fixes
Pre-silicon State of the Art

- Hardware
  - No 2.0 FPGAs available
  - 1.1 is limited use and not readily available.
- Internal Simulation
  - Delays/
  - Can’t work with community
- Prior art
  - nfit_test
  - QEMU CCIX patches
Pre-silicon State of the Art

- **Hardware**
  - No 2.0 FPGAs available
  - 1.1 is limited use and not readily available.

- **Internal Simulation**
  - Delays/
  - Can’t work with community

- **Prior art**
  - nfit_test
  - QEMU CCIX patches
CXL Arch Review

CXL Host Bridge 1
- PCIe RP
- CXL 2.0 RP
- CXL 2.0 RP appears as PCIe RP
- Empty Slot Hot add capable

CXL Host Bridge 2
- CXL 1.1 DP
- CXL 1.1 hierarchy

CXL Upstream Switch Port, Appears as PCIe USP
- CXL 2.0 Switch
  - CXL DSP Appears as PCIe DSP
  - PCIe DSP

CXL 2.0 Device
- EP D0 F0
- CXL DVSEC

CXL 1.1 Device
- RCIEP D0 F0
- RCIEP Or RP
PCle in QEMU

What we all know and love

- Single root complex
  - Endpoints
  - Root ports
  - Switches
- All traffic is funneled to the single host bridge
  - QPI/UPI (not modeled)
PCle ~2014

Host bridge (0:0.0)

Host bridge (n:0.0)

Host bridge (z:0.0)

Root Port

RCIEP

RCIEP
Options

- Hacks to make Q35 - CXL 2.0
  - Limited potential for interleave scenarios
  - Touching Q35 is risky.
  - Mistakes make everything work incorrectly.

- Replace Q35 with something newer
  - Still Risky.
  - A lot of work for not much gain
  - What good does modeling UPI do for QEMU?
  - Doubtful community wants it (support burden).
PCI eXpander Bridge (PXB)
- CXL Type 3 device
  - hw/mem/cxl_type3.c
- CXL Root Port
  - hw/pci-bridge/cxl_root_port.c
- CXL PXB
  - hw/pci-bridge/pci_expander_bridge.c
NVDIMM & PCI had a baby...
Inherits from both interfaces
Mailbox handling
libcxl
• IOCTL

ndctl/cxl

• mailbox

Linux

• mailbox

QEMU

• mailbox

archlinux - # cxl list

```
[
  {
    "memdev":"mem0",
    "pmem_size":268435456,
    "ram_size":0,
    "fw_revision":"BWFW VERSION 00",
    "partition_align":0,
    "lsa_size":0
  }
]
```
November 10th 2020

Patchbomb all the things
● QEMU v3 patches sent
  ○ v4 is ready for submission
  ○ Community contributions for DOE, CDAT, and SPDM
  ○ High bar for adding more
    ■ Nothing exists quite like a CXL memory device
      ● Volatile + Persistent capacities
      ● Interleaving at multiple levels
● Linux phase 1 driver merged in 5.12
  ○ Phase 2 actively being developed.
  ○ Community contributions for DOE and CDAT
● Spec issues found and the fixes prototyped in QEMU
Future
CXL RUST FTW!!!!!!!!!!!
What to do

● Community didn’t adopt
  ○ Minimal feedback
  ○ Major reworks for interleave
  ○ cxl_test came along

● External contributions
  ■ DOE
  ■ CDAT
  ■ SPDM

● Commercial Adoption...
We’re Hiring...

https://jobs.intel.com/ShowJob/Id/3089754/Linux-Kernel-Development-Engineer
• Linux
  • DPA mapping (WIP)
    ▪ Libnvdimm integration
  • Interleave
    ▪ Provisioning
    ▪ Recognition
  • Hotplug
    ▪ Hot add
    ▪ Managed remove
  • Asynchronous mailbox

• Userspace
  • Testing

• QEMU
  • Better tests
  • Upstream/Downstream Ports
  • Interleave Support
    ▪ Host bridge
    ▪ switch
  • More firmware commands
  • Hotplug support
  • Error testing
  • Interrupt support
  • Memory class device overhaul
  • -----------------------
  • Make Q35 CXL capable
  • CXL type 1 and 2 devices
  • CXL 1.1