CXL 2.0 + Linux + QEMU = Yes
Agenda

- Introduction
- CXL 2.0 Background
  - Linux Driver
- QEMU
- Future
Introduction

Last slide first!!!
CXL 2.0 Background
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
An open industry-supported cache-coherent interconnect for processors, memory expansion and accelerators
Compute Express Link™ and CXL™ Consortium are trademarks of the Compute Express Link Consortium.
Usage Models

Caching Devices / Accelerators
- Processor
- CXL
- Accelerator
  - NIC
- Cache
- Protocol:
  - CXL_io
  - CXL_cache
- Usage:
  - I/O
  - Compute
  - Acceleration
  - High bandwidth

Accelerators with Memory
- Processor
- CXL
- Accelerator
  - HBM
- Cache
- Protocol:
  - CXL_io
  - CXL_cache
  - CXL_memory
- Usage:
  - CPU
  - GPU

Memory Buffers
- Processor
- CXL
- Memory Buffer
- Memory
- Protocol:
  - CXL_io
  - CXL_memory
- Usage:
  - Memory
  - Storage
  - Memory capacity expansion
  - Storage class memory
## Form 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
Software
Enumerable Entities
Interleave
Linux Driver
PCI and NVDIMM had a coherent byte addressable baby...

- 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
Software Responsibilities
Linux Drivers

Enumerable
- CXL capabilities
- CXL hierarchy

Enumerable
- PCI Class code
- ACPI discovery

Phase 1
- cxl_pci
- cxl_acpi

Phase 2
- LSA
- cxl_region

- cxl_port
- cxl_mem
- cxl_pmem
- cxl_core
- bus
- mbox
- regs
- memdev
cxl_core

- Maintains cxl_driver infra
- Interface with LIBNVDIMM
- Instantiates and manages device attrs (sysfs)
  - Services to add devices, ie. cxl_decoder_add()
- IOCTL
- Common functionality
  - Mailbox controls
  - Register mapping
cxl_pci

- Probed like a typical PCI device
  - `{ PCI_DEVICE_CLASS((PCI_CLASS_MEMORY_CXL << 8 | CXL_MEMORY_PROGIF), ~0)}`
- Implements mailbox transport (CXL) protocol
- Handles PCI specific functionality for sibling drivers
  - DVSEC
  - Register mappings from BAR
CEDT ACPI0017
ACPI0016
Root Port
USP (Switch)
DSP (Switch)
Endpoint
Endpoint

Host
ACPI0016
ACPI0016
ACPI0016

CXL_PCI
cxl_acpi

- Probed like a typical ACPI device
  - \{ "ACPI0017", (unsigned long) &native_acpi0017 \},
- ACPI0017 starts enumeration of CXL ports
  - CEDT
  - “Root level” ports (platform)
  - Hostbridges and root ports
- Ports are created for all components with an upstream port
  - Hostbridge
  - Switch
  - Endpoint
- Port driver enumerates and controls decoder resources
cxl_mem

- connects a device enumerated with cxl_pci to functionality provided by cxl_port.
- “exports” if device is CXL.mem routed and enabled
- Implements device functionality not handled by cxl_pci
Regions

- Region
  - Interleave set of devices
  - Parameters (IG, HPA, etc)

- Creation
  - Via sysfs ABI
  - Provisioned offline
    - Manufacturing 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.

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.ENI/W determines how many bits of the position are masked.

4) Verify each device on each Host Bridge has the same masked position value: Device[x].RegionLabel.Position >> (CFMWS.HBIG - Device[x].RegionLabel.IG) & ((2^CFMWS.ENI/W) - 1)

Valid XHB device configuration
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**
   - 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
CXL Arch Review

- **CXL 1.1 Device**:
  - CPU
  - CXL 1.1 DP
  - RCiEP

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

- **CXL 2.0 RP** appears as PCIe RP

- **CXL 2.0 RP** appears as PCIe RP

- **PCIe device**: Empty Slot, Hot add capable

- **CXL Host Bridge 1**: PCIe RP

- **CXL Host Bridge 2**: CXL 1.1 DP

- **CXL Arch Review**
PCIe 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)
Just like CXL
Options

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

- 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

```c
static const TypeInfo ct3d_info = {
    .name = TYPE_CXL_TYPE3_DEV,
    .parent = TYPE_PCI_DEVICE,
    .class_init = ct3_class_init,
    .instance_size = sizeof(CXLType3Dev),
    .instance_init = ct3_instance_init,
    .instance_finalize = ct3_finalize,
    .interfaces = (InterfaceInfo[])
```
- QEMU Emulation
  - Memory region
  - MMIO
  - Mailboxes
- Kernel driver
  - Documentation!
- Tools
- 2.0 spec is available
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
What to do

- Community didn’t adopt
  - Minimal feedback
  - Major reworks for interleave
  - cxl_test came along
- External contributions
  - DOE
  - CDAT
  - SPDM
- Commercial Adoption...
• 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
• qemu-system-x86_64
  • -machine q35,<cxl>
  • -object
    memory-backend-file,id=cxl-mem1,share,mem-path=cxl-type3,size=512M
  • -device
    pxb-cxl,id=cxl.0,bus=pcie.0,bus_nr=52,uid=0,len-window-base=1,window-base[0]=0x4c00000000,memdev[0]=cxl-mem1
  • -device cxl-rp,id=rp0,bus=cxl.0,addr=0.0,chassis=0,slot=0
  • -device cxl-type3,bus=rp0,memdev=cxl-mem1,id=cxl-pmem0,size=256M

• CXL utilities part of ndctl being developed
  • https://github.com/pmem/ndctl/tree/cxl-2.0v3