OAK RIDGE National Laboratory

Design and Analysis of CXL Performance Models for Tightly-Coupled Heterogeneous Computing

Anthony M Cabrera*, **Aaron R Young***, Jeffrey S Vetter

ExHET'22

ORNL
April 2, 2022

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

* Both authors contributed equally to this research.

OAK RIDGE National Laboratory

Introduction

Challenges with Heterogeneous Computing:

- Easy to use programming models
- Leverage different compute units
- Share data between accelerators
- Fine-grained, high-performance

Current heterogeneous system use host memory as an intermediary

How could CXL be used to addess these challenges?

Overview

CXL

- CXL = Compute Express Link
- Unveiled in March 2019
- Open industry standard processor interconnect
- Unified, coherent memory space between the CPU and any memory attached CXL device.
- High-bandwidth, low-latency connection between host and devices including accelerators, memory expansion, and smart I/O devices.
- Utilizes PCI Express 5.0 physical layer infrastructure and the PCIe alternate protocol.
- Designed to meet demanding needs of HPC work in AI, ML, communication systems through enablement of coherency and memory semantics across heterogeneous processing and memory systems.

CXL

- CXL is a non-symmetric protocol.
- The CXL transaction layer is comprised of three dynamically multiplexed sub-protocols on a single link:
  - CXL.io: Provides discovery, configuration, register access, interrupts, DMA, etc.
  - CXL.cache: Provides devices access to unified memory space.
  - CXL.memory: Allows devices to provide memory to the unified memory space.

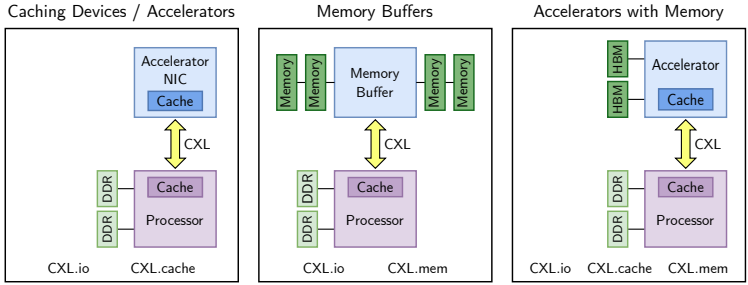


Figure 1: CXL device classes and sub-protocols [3].

CXL

- CXL is a non-symmetric proto
- The CXL transaction layer is c plexed sub-protocols on a single link:
  - CXL.io: Provides discovery , DMA, etc.
  - CXL.cache: Provides devic
  - CXL.memory: Allows devices to provide memory to the unified memory space.

For this work the focus is on accelerators with memory using host bias.

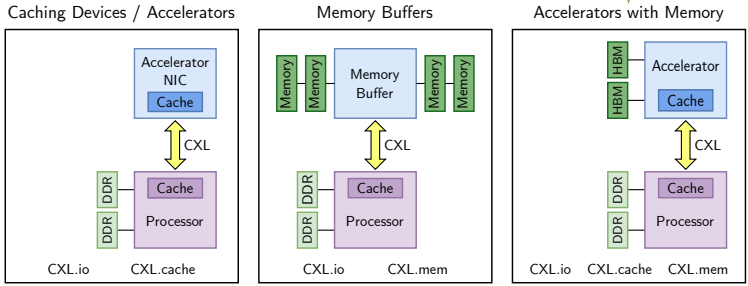


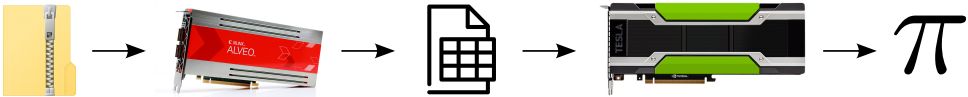
Figure 1: CXL device classes and sub-protocols [3].

Baseline Application: DecEval

- Decompress Evaluate (DecEval) is a generic dataflow problem.
- We created DecEval as an example collaborative accelerator application.
- Application flow:
  - Decompress double precision data that inflates to $1.6\,\mathrm{GB}$
  - Perform numerical integration to estimate π using the following formula:

$$\sum_{i=0}^{N} \frac{4.0}{1+x_i^2} \Delta x \approx \pi$$

- We map the decompression step to an FPGA, and the numerical integration on the GPU



Compressed Data | U250 | Uncompressed Values | Uncompressed Values | Pi

Why This Application?

- This flavor of application highlights the benefit of extremely heterogeneous systems
  - GPUs are adept at compute-intensive, floating-point computation
  - FPGAs can be leveraged to exploit sequential, pipeline-able applications
  - We want to leverage the strengths of different accelerators to accelerate applications with diverse workloads
  - We use this application as a case study to show how CXL can further enhance extremely heterogeneous systems by enabling tighter integration between compute and memory components

Baseline Implementation

- Explicit memory transfers between accelerators using the host memory.
- FPGA implemented using gzip from Xilinx Vitis Libraries (OpenCL 2.0 API).
- GPU implementation from Bristol HPC group [2] (OpenMP 4.0).

Test Platform

Component	Model	Additonal Information
CPU	Intel Xeon Gold 6130	VM configured with 24 vCPUs 92 GB RAM PCIe Passthrough
FPGA	Xilinx Alveo U250	Ultrascale+ μ-arch 64 GB off-chip DDR4 RAM PCIe Gen 3x16 Xilinx HLS Vitis Library + OpenCL Host
GPU	NVidia P100	Pascal μ-arch 16 GB off-chip HBM2 PCIe Gen 3x16 CUDA through OpenMP 4.0

Table 1: Experimental System Specification

CXL Performance Model

- Based on high-level CXL performance expectations.
- Integrates PCIe model from Neugebauer et al. [1]
- Uses PCIe 3.0 with x16 lanes.

PCIe Transfer Time

$$t_{pcie} = \frac{n}{b_n} + t_s$$

$t_s = 738.77$ ns (extracted from [1]).

CXL Transfer Time

$$t_{cxl} = \frac{n}{b_n \cdot cxl\_penalty} + t_s$$

$t_s = 40$ ns (CXL 1.1 Technical Training Videos [3]).
$cxl\_penalty = 60$–90% of PCIe

- b_n calculated from PCIe model [1].

CXL Performance Model

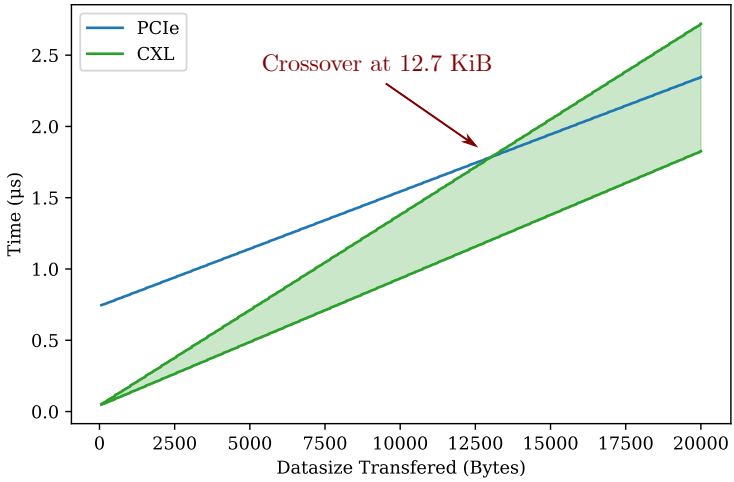


Figure 2: Single transfer time for various data sizes.

CXL Performance Model

- Converges to $cxl\_penalty$ value.
- At 1 cache line (64B) speed up of 14.0–$18.7\times$.

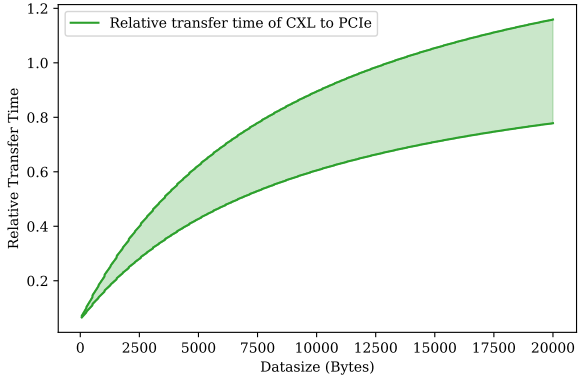


Figure 3: Relative transfer time of CXL and PCIe.

Baseline Compared to Model

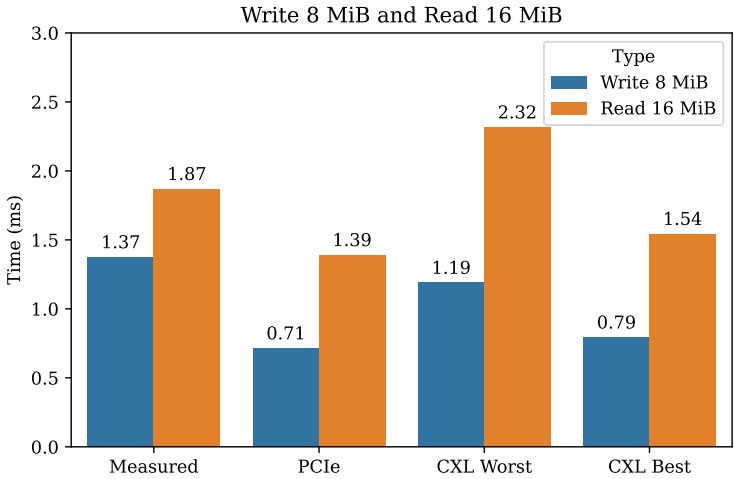


Figure 4: Comparison of actual data transfer time for DecEval and the times predicted by the model.

Effect of Varied Communication Block Sizes

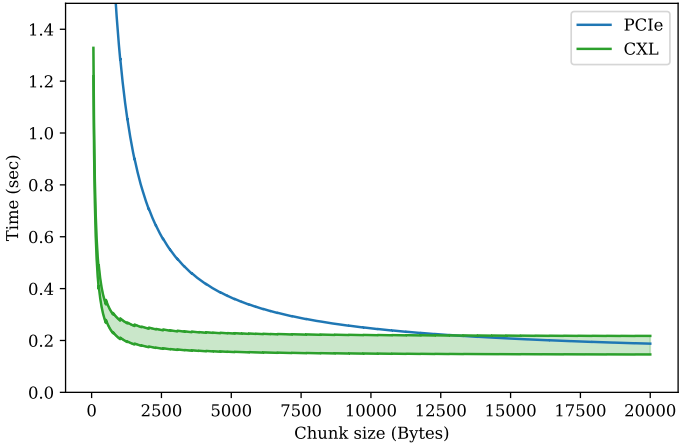
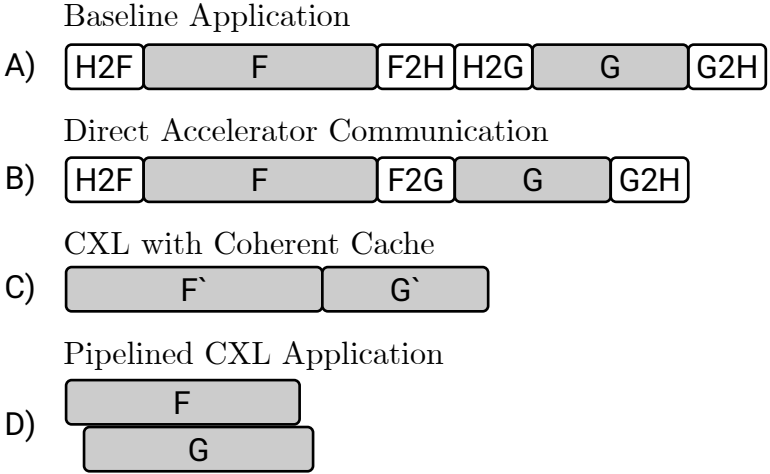


Figure 5: CXL Model: Total Transfer time for the entire application when transferring data in various chunk sizes.

Application Flows



Pipelined Application Model

CXL Transfer Time

$$t_{app} = d \cdot t_{stage} + (n-1) \cdot t_{stage} + s$$

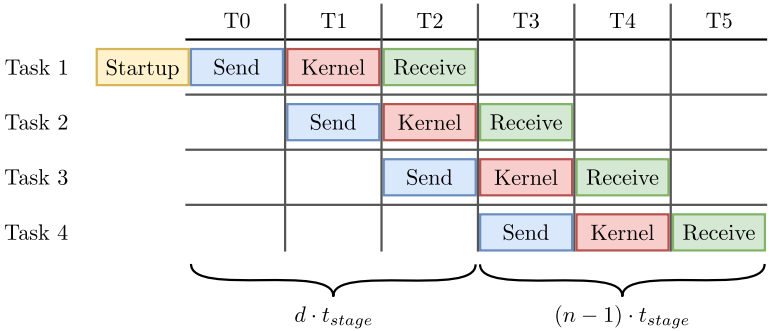
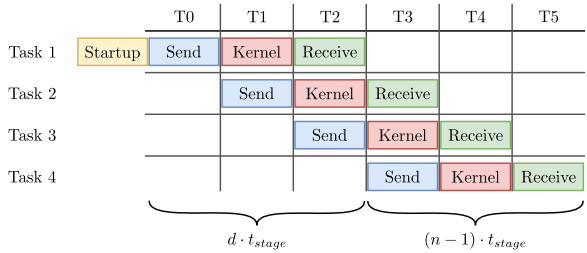


Figure 6: Diagram of the balanced pipeline application execution modeled by the simple application model. In this diagram $d = 3$ and $n = 4$.

Application Flow Results

Pipeline limited by FPGA Kernel



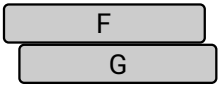
FPGA decompression pipeline:

FPGA Kernel = 70.9 ms
Reads = 1.87 ms
Writes = 1.37 ms

FPGA vs GPU Kernel execution:

FPGA Kernel = 7.70 sec (21× slower than GPU Kernel)
GPU Kernel = 0.36 sec
Total = 10.58 sec

D)

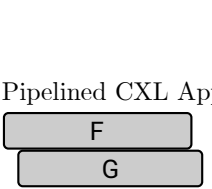


Application Flow Results

C)

$$\text{CXL Application Speedup} = \frac{T_{total}}{T_{GPU} + T_{FPGA}} = \mathbf{1.31}\times$$

D)



$$t_{app} = d \cdot t_{stage} + (n-1) \cdot t_{stage} + s = \mathbf{1.45}\times$$

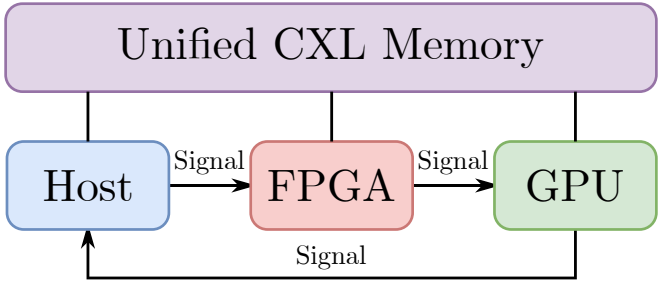
CXL-Enabled Programming Model

CXL.io device registers

- Setup accelerators.
- Control execution of kernel.
- Specify data locations.

CXL.cache

- Access to unified memory



Conclusion

- 14.0–18.7× speed-up for small data transfers.
- CXL outperforms PCIe for transfers less than 12.7–76.5 KiB
- CXL to avoid data transfers through host memory
  - 1.31× speedup with CXL Cache transfers.
  - 1.45× speedup with CXL Pipeline.
- Heterogeneous programming can be easier with a unified cache-coherent memory model.

Conclusion

- 14.0–18.7× speed-up for small data transfers.
- CXL outperforms PCIe for transfers less than 12.7–76.5 KiB
- CXL to avoid data transfers through host memory
  - 1.31× speedup with CXL Cache transfers.
  - 1.45× speedup with CXL Pipeline.
- Heterogeneous programming can be easier with a unified cache-coherent memory model.

Questions?

OAK RIDGE National Laboratory

Design and Analysis of CXL Performance Models for Tightly-Coupled Heterogeneous Computing

Anthony M Cabrera*, **Aaron R Young***, Jeffrey S Vetter

ExHET'22

ORNL
April 2, 2022

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

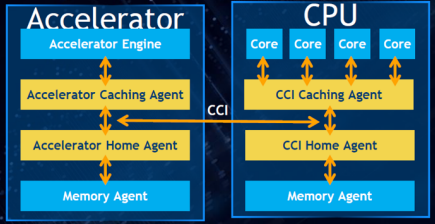
U.S. DEPARTMENT OF ENERGY

* Both authors contributed equally to this research.

OAK RIDGE National Laboratory

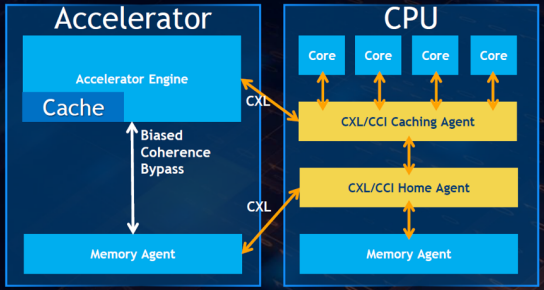
CXL's Protocol Asymmetry

CCI Model - Symmetric CCI Protocol



CCI - Cache Coherent Interconnect

CXL Model - Asymmetric Protocol

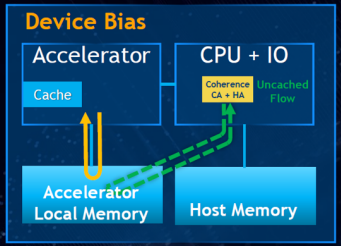


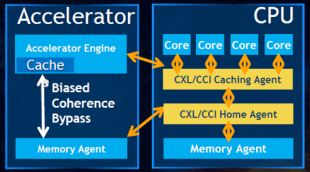
- ***CXL key advantages***:
  - \+ Avoid protocol interoperability hurdles/roadblocks
  - \+ Enable devices across multiple segments (e.g. client / server)
  - \+ Enable Memory buffer with no coherency burden
  - \+ Simpler, processor independent device development

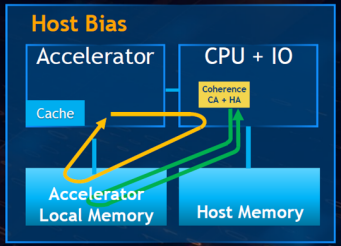
CXL Compute Express Link

From: Compute Express Link™ (CXL™): A Coherent Interface for Ultra-High-Speed Transfers [August 2019 - Presentation]

CXL's Coherence Bias







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

Two driver managed modes or "Biases"

HOST BIAS: pages being used by the host or shared between host and device

DEVICE BIAS: pages being used exclusively by the device

Both biases guaranteed correct/coherent

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

Coherency Guaranteed

From: Compute Express Link™ (CXL™): A Coherent Interface for Ultra-High-Speed Transfers [August 2019 - Presentation]

Minimum application speedup for acceleration

$$t - t' > 2d$$

where t is the time the task takes to run on the current device, t' is the time the task takes to run on a different device, and d is the time to move the data required to perform the task.

Bibliography I

[1] Neugebauer et al. “Understanding PCIe performance for end host networking”. In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 327–341. DOI: https://doi.org/10.1145/3230543.3230560.

[2] University of Bristol HPC Group. *Programming Your GPU with OpenMP*. https://github.com/UoB-HPC/openmp-tutorial. 2020.

[3] Compute Express Link Consortium. *CXL 1.1 Technical Training Videos*. Aug. 16, 2021. URL: https://www.computeexpresslink.org/cxl-regulated-videos.