Memory Expansion and Storage Acceleration with CCIX Technology

Millind Mittal, Fellow, Xilinx
Jason Lawley, DC Platform Architect, Xilinx
Agenda

• Brief introduction to CCIX
• Memory Expansion through CCIX
• Persistent Memory support
• Storage with Compute offload
• Q&A
CCIX Context

- Slow down of performance scaling and efficient of general purpose processors
- Increasing “workload specific” computation requirements
  - Data analytics, 400G, ML, Security, compression, …
- Lower latency requirements
  - Cloud based services, IoT, 5G, …
- Need for open standard for advancing IO Interconnect to enable seamless expansion of compute and memory resources
  - Enable accelerator SoCs to be like a NUMA sockets from Data Sharing perspective
The CCIX Consortium

- 53 Members covering all aspects of ecosystems; Servers, CPU/SoC, Accelerators, OS, IP/NoC, Switch, Memory/SCM, Test & Measurement vendors.
- Specification Status
  - Rev 1.0 - 2018
  - Rev 1.1/Rev1.2 – 2019
  - SW Guide Rev 1.0- Sept, 19
- CCIX Hosts:
  - ARM 7nm test Processor SoC providing CCIX interface (N1SDP)
  - Huawei announced Kunpeng 920
  - A 3rd party ARM SoC, Sample 12/19
- CCIX Accelerator / EP
  - Xilinx VU3xP family
  - Alveo boards (U50 and U280) available
  - 7nm chip Versal with CCIX support announced
- SW Enablement
  - In progress ; Key enablement to be completed Sept, 19
Use of Caches for System Performance
Role of Slave Agent

• Slave Agent provides additional memory to a Home Agent
• Slave Agent is only protocol visible when residing on a different chip
CCIX and PCIe Transaction Layers
- Responsible for handling their respective packets
- PCIe & CCIX packets are split across virtual channels (VCs) sharing same link
- Optimized CCIX packets: Eliminates the PCIe overhead

PCle Data Link Layer
- Performs normal functions of the data link layer

CCIX Protocol Layer
- Responsible for the coherency including memory read and write flows
- CCIX Link Layer
- Responsible for formatting CCIX traffic for the target transport and non-blocking behavior between two CCIX devices
- Currently PCIe but could be mapped over a different transport layer in the future

CCIX/PCle Physical Layer
- Faster speed, known as ESM (Extended Speed Mode)
CCIX – Open Standard Memory Expansion and Fine-Grain Data Sharing Model with Accelerators

1. Coarse grain (producer consumer) data sharing model
2. PCIe style IOC based model but with high BW and lower latency
Enabling Seamless Expansion of Compute and Memory Resources – Accelerator SoCs are seen as NUMA Socket

KVS Database with Host-Only Processing (Default)
- All operations works out of host memory
- Adding persistence to updates requires additional IO to Persistent storage, e.g. NVMe

KVS Database with Memory Expansion using CCIX to connect to Persistent Memory (PMEM)
- Supports larger databases than DDR
- Eliminates filesystem processing for log/checkpoint work
- No risk of data loss even though there is no explicit back-up done
- Instantaneous restarts

Application works directly on PMEM

KVS Database with Processing Acceleration
- No disruption to Networking
- Control processing (Set, Delete, etc) remain on host processor
- Fast path operations (Get) move to accelerator
- CCIX enabled shared data structures, no copies
- Increase throughput multi-gets with almost no increase in CPU utilization

KVS Database with Memory Expansion Plus Acceleration

Combined Benefit of Memory Expansion with PMEM and Processing Offload
Direct attached, daisy chain, mesh and switched topologies
SW enablement in progress

- ACPI 6.3 and UEFI 2.8 enhancements for CCIX
  - Specific-purpose Memory
  - Generic Initiator Affinity Structure and associated _OSC bit
  - HMAT Table Enhancements
  - New CPER record for CCIX
- Ongoing Reference Code Implementation jointly done by Linaro, Arm and other members
  - Mail list ccix@linaro.org
  - JIRA Initiative [https://projects.linaro.org/browse/LDCG-713](https://projects.linaro.org/browse/LDCG-713)
  - Work presented at Linaro Connect BKK19 in April 2019
  - UEFI Firmware code is available as part of project
Memory Expansion Through CCIX
Demonstrated Extended memory through NUMA over CCIX at SC18

KVS Database (Memcached) was enhanced to make use of NUMA expansion model over CCIX

Key allocations are done in Host DDR, while corresponding values were allocated on remote FPGA memory

Expansion memory can also be a persistent memory connected over CCIX link

https://www.youtube.com/watch?v=drJu4vluE&list=PLRr5m7hDN9TL13vuwlOqLbF7YeGt3UO9c&index=9
Redis with Persistent Memory support

Without Persistent Memory

With Persistent Memory

Flash (backing store)

RDB (checkpoint) file

AOF on-disk (log) file
Storage with Compute Offload
Analysis and Inference

- WiredTiger is a performance, scalable, production quality, NoSQL, Open Source extensible platform for data management.

- Run two performance benchmark tests & collected call stacks:
  - https://github.com/johnlpage/POCDriver
  - https://github.com/mdcallag/iibench-mongodb

- Major hot spots were identified as:
  - WiredTiger IO operations (IO intense)
  - Compression (CPU intense)

WiredTiger Storage Engine (http://source.wiredtiger.com/)
Accelerated Design Over CCIX

- IOPs are limited due to OS context switch and other SW overheads
- Enable user space calls to FS directly
- Offload performance critical operations (writes/reads) fully to FPGA with interface to storage
  - File system Meta data structures are maintained in shared FPGA memory
  - Actual file data is stored over FPGA connected storage class memory which is faster than SSDs
- Inline efficient Compression
- Seamless acceleration architecture through shared meta-data enabled by CCIX
Instead of full file system offload we propose a split file system with Metadata share over CCIX interface.

CPU Handled operations:
- `fs_open` – Creates new file or reopens the existing file
- `fs_exist` – Checks whether the file exists
- `fs_rename` – Renames existing file
- `fs_terminate` – Closes the file system
- `fs_create` – Creates the file system
- `file_size` – Returns the file size
- `file_close` – Closes the file
- `file_truncate` – Truncates the file to the specified size
- `fs_read` – Reads a data block from file

All these operations need not be sent to FPGA as these can read/edit the shared structures.

Only handle `fs_write` in FPGA with the focus to achieve accelerated performance for Writes.
- Be able to ingest the data into NoSQL DBs like MongoDB.
SC19 processing flow
Without data compression

In-memory document

Wired Tiger Storage Layer

Application Buffer

File_write

File_read

FS_read thread

Write-Engine

Write IO Engine

FS meta-data; Permissions, size, inode, ...

Indexed by FileID.offset

Buffer cache (DRAM or PMEM)

Block Storage

Optional
SC19 processing flow
With data compression

In-memory document

Wired Tiger Storage Layer

Application Buffer

File_read_uncompress

File_write_compress

Update “size” in WT

FS_read_thread

Write-Engine

Write IO Engine

FS meta-data; Permissions, size, inode, ....

Indexed by FileID.offset

Buffer cache (DRAM or PMEM)

Block Storage

Host Memory

FPGA Memory

HA

RA Cache

Host

FPGA

User

Kernel

FPGA with RA

Block Storage
Split File System Operation Distribution Between Host & FPGA

HOST

User space

App1
App2
App3

FSlib

Kernel

FPGA

FPGA File System HW Engine for 
FS_Write
FS_Write-with-compression

Meta Data

FS_Read and Control/Management operations

Disks

Meta-data sharing enabled by CCIX
Meta-data in the FPGA Attached Memory

HOST

User space

App1
App2
App3

FSlib

Kernel

FPGA

FPGA File System HW Engine for **FS_Write**
**FS_Write-with-compression**

Meta Data

FS_Read and Control/Management operations

Disks

Meta-data sharing enabled by CCIX
Current PoCs underway

• Storage layer acceleration
  • PMDK framework enablement for ARM processors for SCM
  • Write IO-Ops acceleration for MongoDB ← Show case at SC19
• Memory expansion on Xilinx Versal device ← XDF 19
CCIX enables new platform level capability to enable accelerated solutions for storage and other verticals.

CCIX technology is ready to develop PoCs and products.

Contact below to learn more:

https://www.ccixconsortium.com/ or

You can contact me at millind@Xilinx.com