

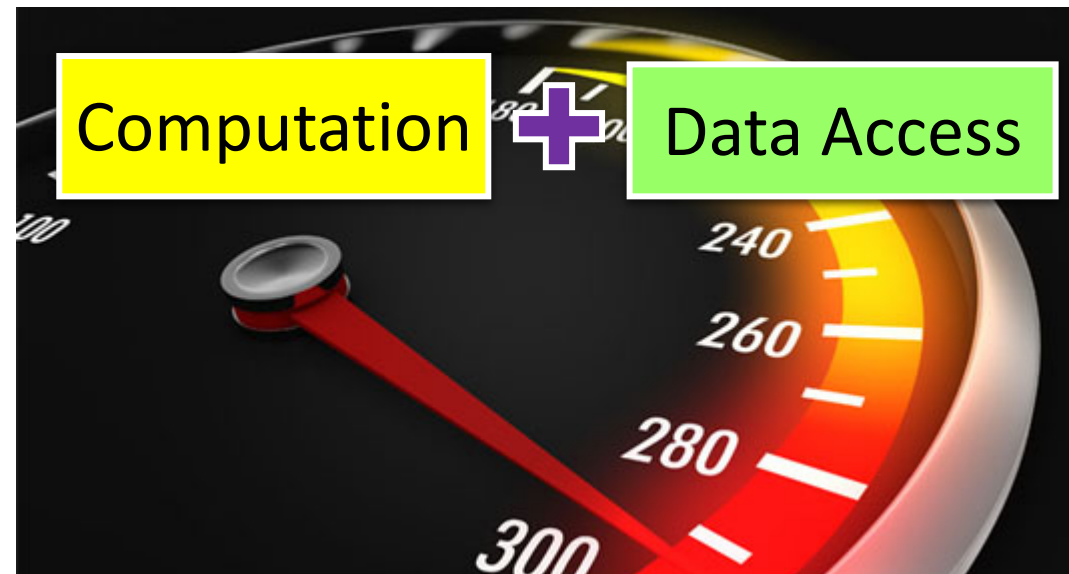
Industry Collaboration and Innovation



A data-centric approach to server design



- Key changes occurring in our industry
 - Historical microprocessor technology continues to deliver far less than the historical rate of cost/performance improvement per generation – Running out of steam
 - New advanced memory technologies changing the economics of computing
- Companies realizing need for **accelerated computing** with a coherent **high performance bus** to meet today's computational demand

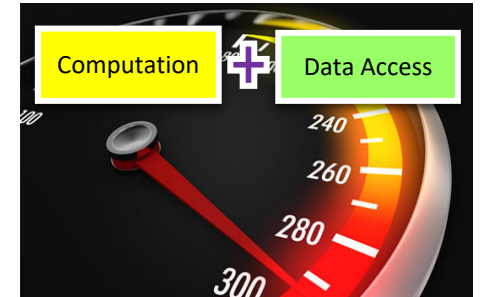


Accelerated Computing and High Performance Bus



■ Attributes driving Accelerators

- Emergence of **complex storage and memory solutions**
- Introduction of device **coherency requirements** (IBM's introduction in 2013)
- Growing demand for **network performance and network offload**
- Various form factors (e.g., GPUs, FPGAs, ASICs, etc.)
 - All are relevant in the modern data center
 - No single FF can address everything



■ Driving factors for a high performance bus - Consider the environment

- Increased industry dependence on hardware acceleration for performance
- **Hyperscale datacenters and HPC** are driving need for much higher network bandwidth
 - *100 Gb/s -> 200 Gb/s -> 400Gb/s are emerging*
- **Deep learning** and HPC require more bandwidth between accelerators and memory
- New **memory/storage technologies** are increasing the need for bandwidth with low latency

Two Bus Challenges



1. Coherent high performance bus needed
 - Hardware acceleration will become commonplace
 - But, if you are going to use Accelerators, you need to get data in/out very quickly
 - Today's system interfaces are insufficient to address this requirement
 - Traditional I/O architecture results in very high CPU overhead when applications communicate with I/O or Accelerator devices
 - Systems must be able to integrate multiple memory technologies with different access methods, coherency and performance attributes
2. These challenges must be addressed in an open architecture allowing full industry participation
 - Need to be architecture agnostic to enable the ecosystem growth and adaption
 - Establish sufficient volume base to drive cost down
 - Support broad ecosystem of software and attached devices

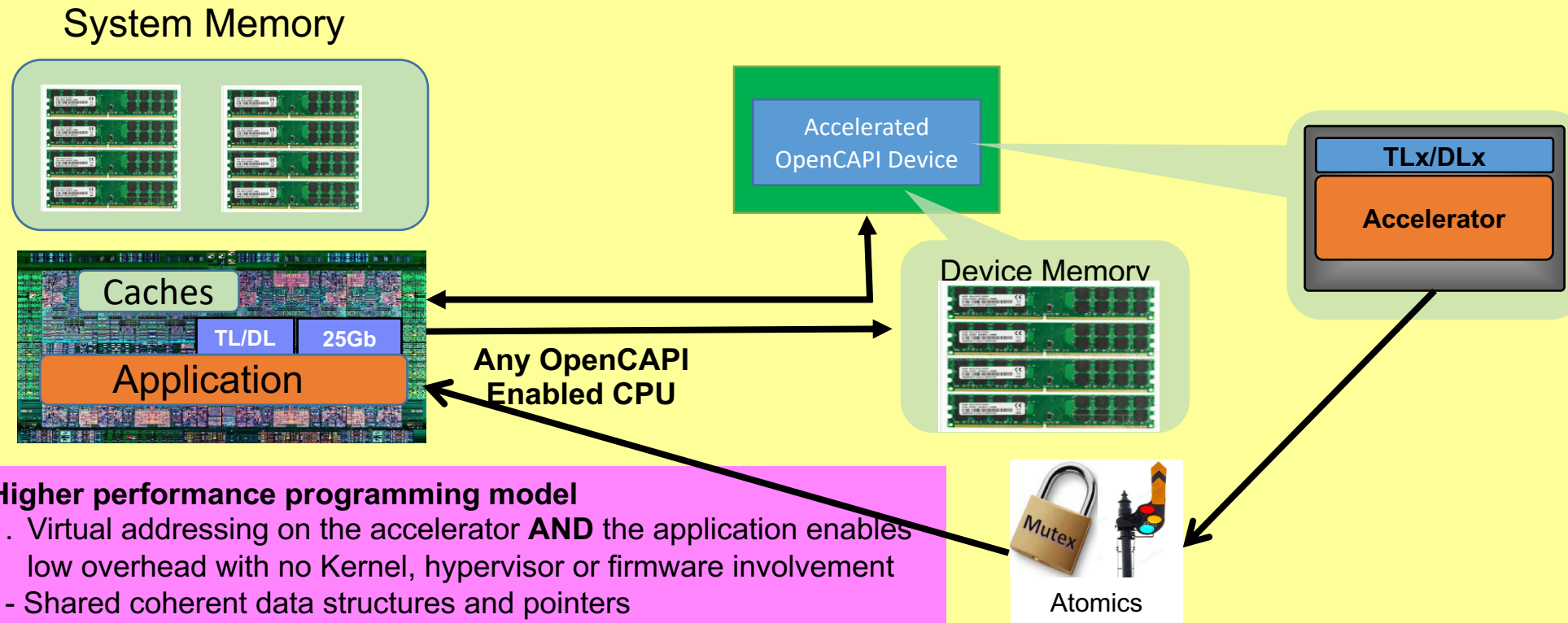
- Two bus challenges
 1. Coherent high performance bus
 - Needed to define a new technology
 2. A need to make this 'open'
 - Needed to establish open community
- Approach taken to define this new technology
 - Bottom's up design approach for advanced capabilities and performance
 - Clean sheet of paper with no incumbent overhead
 - Architecture and electrical view point
- **OpenCAPI** bus architecture was Born

Virtual Addressing and Benefits



- **An OpenCAPI device operates in the virtual address spaces of the applications that it supports**
 - Eliminates kernel and device driver software overhead
 - Allows device to operate on application memory without kernel-level data copies/pinned pages
 - Simplifies programming effort to integrate accelerators into applications
 - Improves accelerator performance
- **The Virtual-to-Physical Address Translation occurs in the host CPU**
 - Reduces design complexity of OpenCAPI-attached devices
 - Makes it easier to ensure interoperability between OpenCAPI devices and different CPU architectures
 - Security - Since the OpenCAPI device never has access to a physical address, this eliminates the possibility of a defective or malicious device accessing memory locations belonging to the kernel or other applications that it is not authorized to access

OpenCAPI's Programming Ease



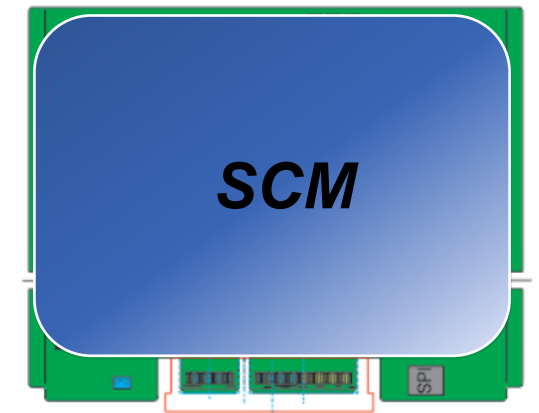
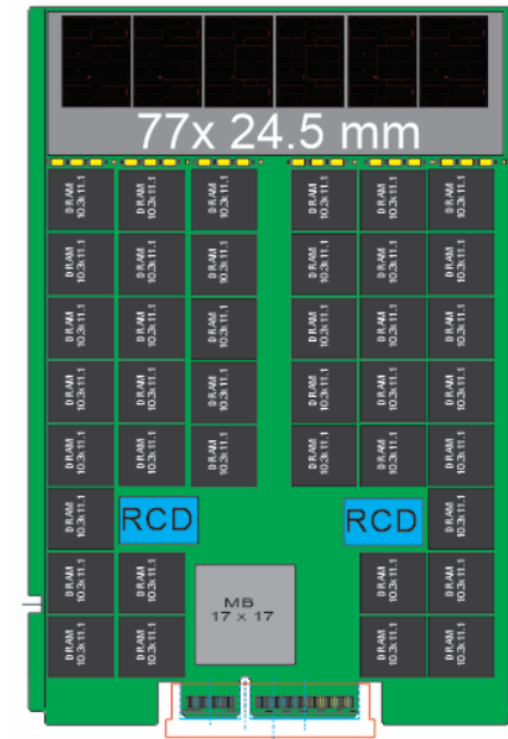
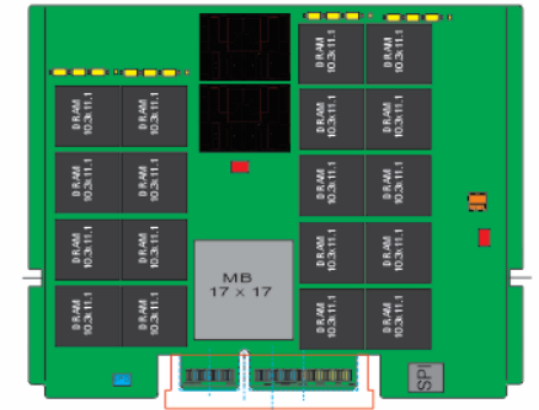
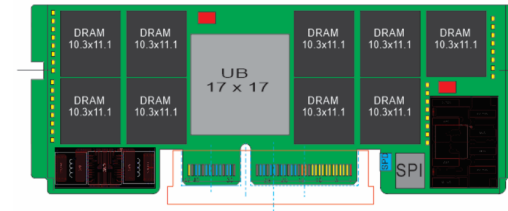
Higher performance programming model

1. Virtual addressing on the accelerator **AND** the application enables low overhead with no Kernel, hypervisor or firmware involvement
 - Shared coherent data structures and pointers
2. CPU <-> Accelerator
 - OpenCAPI removes PCIe layering (latency optimized)
 - Fast communication from CPU to Accelerator via Load/Store
3. CPU coherent device memory
 - Home Agent Memory using Load/Store
4. Traditional thread level programming
5. Atomic operations for barriers, locks, semaphores, etc.

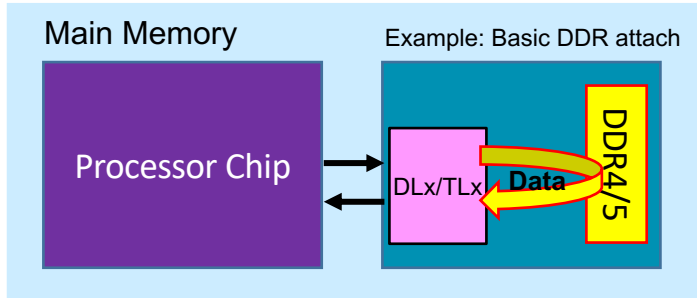
OpenCAPI Advantages for Memory



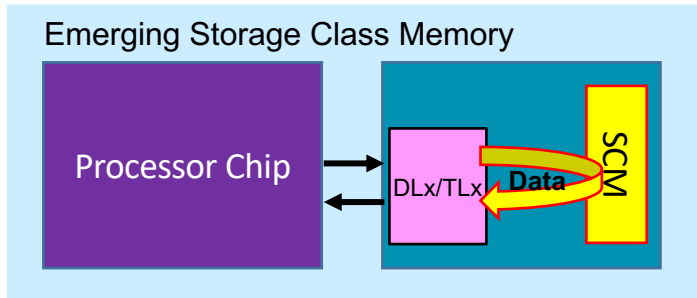
- Open standard interface enables to attach wide range of devices
- OpenCAPI protocol was architected to minimize latency
 - Especially advantageous for classic DRAM memory
- Extreme bandwidth beyond classical DDR memory interface
- Agnostic interface allows extension to evolving memory technologies in the future (e.g., compute-in-memory)
- Ability to handle a memory buffer to decouple raw memory and host interfaces to optimize power, cost and performance
- Common physical interface between non-memory and memory devices



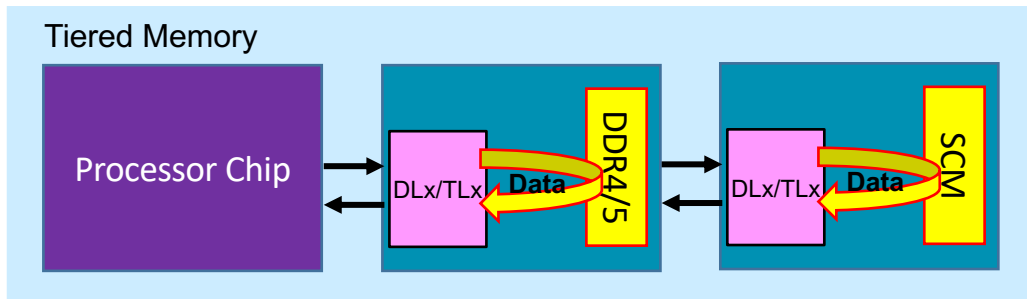
Comparison of Memory Paradigms



OpenCAPI 3.1 Architecture
Ultra Low Latency ASIC buffer chip adding +5ns
on top of native DDR direct connect!!

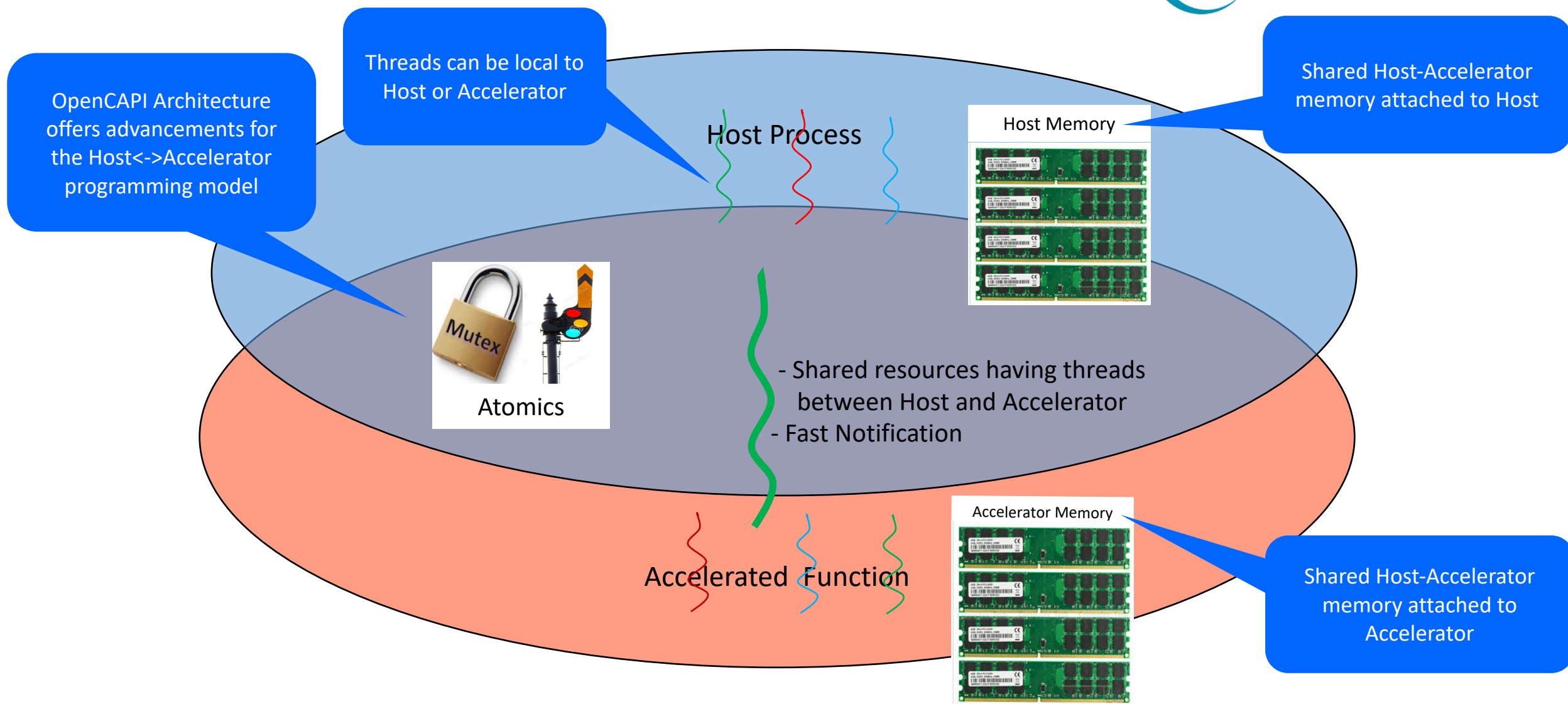


Storage Class Memories have the potential to be
the next disruptive technology.....
Examples include ReRAM, MRAM, Z-NAND.....
All are racing to become the defacto



Storage Class Memory tiered with traditional DDR
Memory all built upon OpenCAPI 3.1 & 3.0
architecture.
Still have the ability to use Load/Store Semantics

OpenCAPI Coherence Programming Model

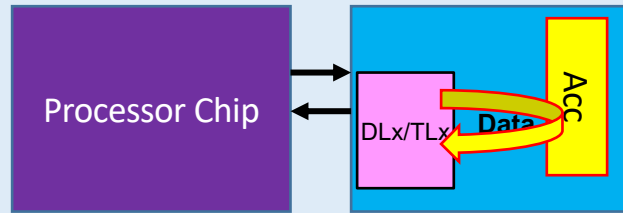


Acceleration Paradigms with Great Performance



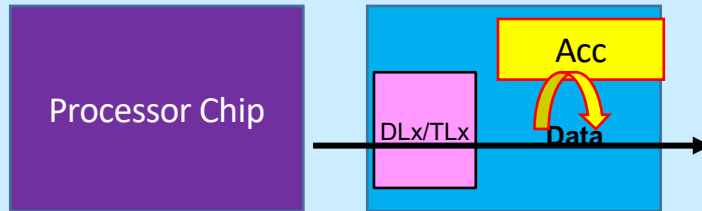
Memory Transform

Example: Basic work offload



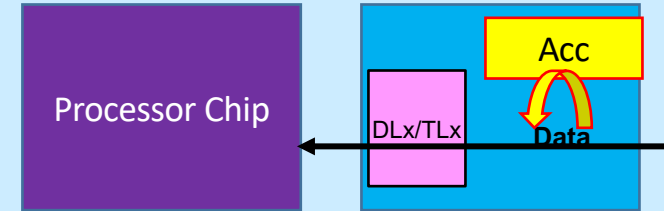
Examples: Machine or Deep Learning such as Natural Language processing, sentiment analysis or other Actionable Intelligence using OpenCAPI attached memory

Egress Transform



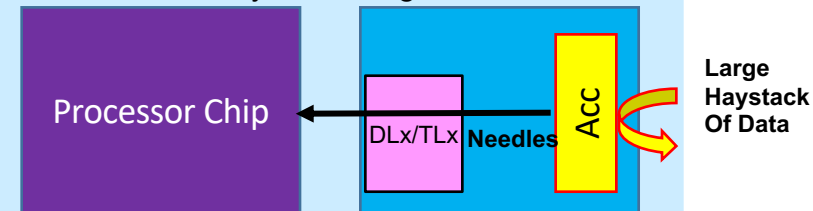
Examples: Encryption, Compression, Erasure prior to delivering data to the network or storage

Ingress Transform



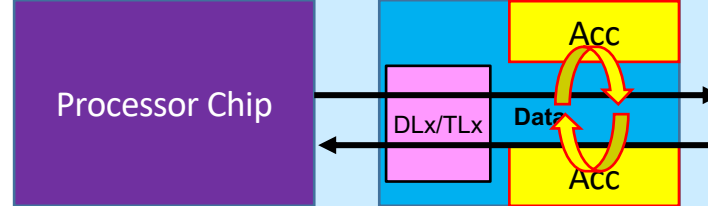
Examples: Video Analytics, Network Security, Deep Packet Inspection, Data Plane Accelerator, Video Encoding (H.265), High Frequency Trading etc

Needle-In-A-Haystack Engine



Examples: Database searches, joins, intersections, merges
Only the Needles are sent to the processor

Bi-Directional Transform



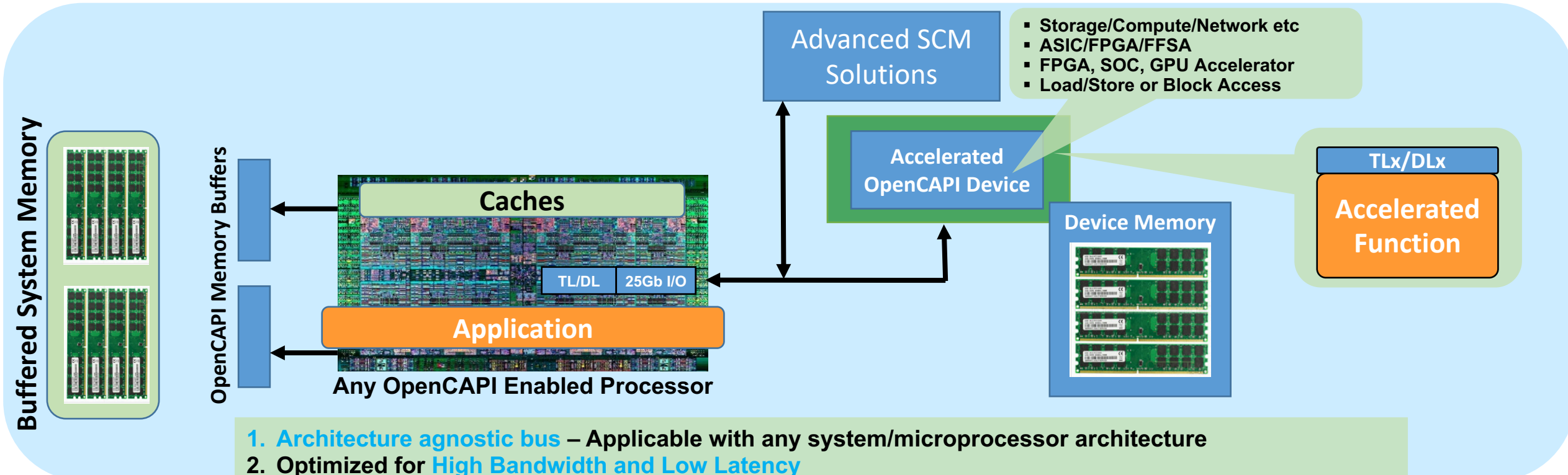
Examples: NoSQL such as Neo4J with Graph Node Traversals, etc

PHY: 25 Gbit/sec and Beyond



- OpenCAPI is agnostic to processor architecture and as such the electrical interface can be defined by any processor team
- However if a partner wishes to connect with IBM's POWER9 microprocessor the electrical interface is defined as follows
 - Definition is being driven by the 25G Work Group in the OpenPOWER Consortium
 - Based on the OIF CEI 28G SR specification
 - POWER9's 25Gbit/sec signaling and protocol built to enable very low latency interface on CPU and attached devices
- Looking forward, IBM's POWER microprocessor PHYs will be defined in the OpenCAPI Consortium within the PHY Signaling and PHY Mechanical Workgroups which are up and running (e.g., 32 Gb/s, 56 Gb/s signaling, etc.)

OpenCAPI Key Attributes



1. **Architecture agnostic bus** – Applicable with any system/microprocessor architecture
2. Optimized for **High Bandwidth and Low Latency**
3. High performance **industry standard** interface design with zero 'overhead'
4. **Coherency** - Attached devices operate natively within application's user space and coherently with host microprocessor
5. **Virtual addressing** enables low overhead with no Kernel, hypervisor or firmware involvement
6. Supports a **wide range of use cases and access semantics**
7. **CPU coherent device memory** (Home Agent Memory)
8. Architected for both Classic Memory and emerging Storage Class Memory
9. **Minimal OpenCAPI design overhead**

Comparison of IBM CAPI Implementations



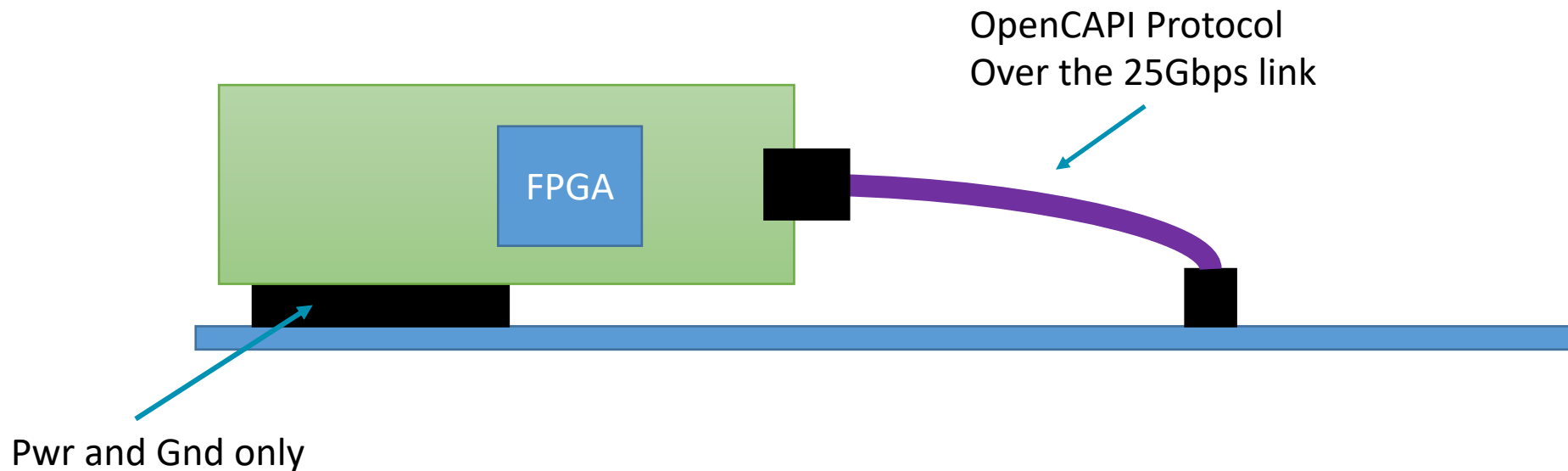
Feature	CAPI 1.0	CAPI 2.0	OpenCAPI 3.0	OpenCAPI 3.1	OpenCAPI 4.0
Processor Generation	POWER8	POWER9	POWER9	Power9 Follow-On	Power9 Follow-On
CAPI Logic Placement	FPGA/ASIC	FPGA/ASIC	NA DL/TL on Host DLx/TLx on endpoint FPGA/ASIC	NA DL/TL on Host DLx/TLx on Memory Buffer	NA DL/TL on Host DLx/TLx on endpoint FPGA/ASIC
Interface Lanes per Instance Lane bit rate	PCIe Gen3 x8/x16 8 Gb/s	PCIe Gen4 2 x (Dual x8) 16 Gb/s	Direct 25G x8 25 Gb/s	Direct 25G x8 25 Gb/s	Direct 25G x8 25 Gb/s
Address Translation on CPU	No	Yes	Yes	Yes	Yes
Native DMA from Endpoint Accelerator	No	Yes	Yes	NA	Yes
Home Agent Memory on OpenCAPI Endpoint with Load/Store Access	No	No	Yes	NA	Yes
Native Atomic Ops to Host Processor Memory from Accelerator	No	Yes	Yes	NA	Yes
Host Memory Caching Function on Accelerator	Real Address Cache in PSL	Real Address Cache in PSL	No	NA	Effective Address Cache in Accelerator

Remove PCIe layers to reduce latency significantly

Reference Card Design Number 1



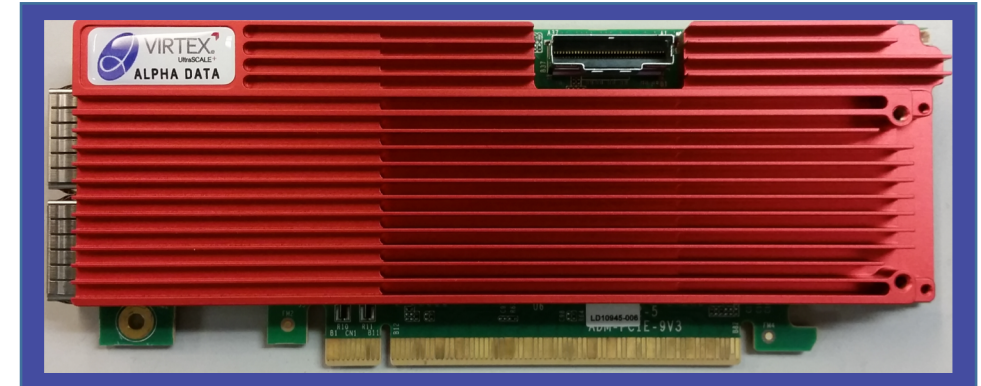
- Definition of the FPGA reference card(s) is driven as part of the Enablement Work Group
- Definition of the cable(s) is driven as part of the PHY Mechanical Work Group
- Representative Diagram is articulated below



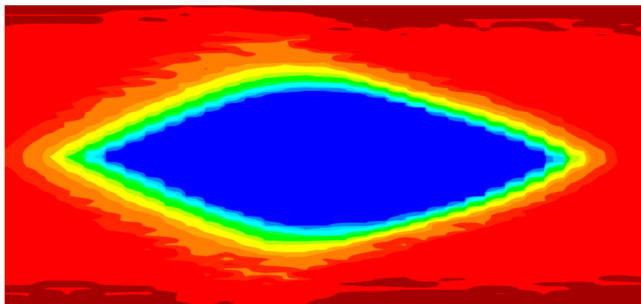
OpenCAPI Enabled FPGA Cards



Mellanox Innova2 Accelerator Card



Alpha Data 9v3 Accelerator Card



Typical eye diagram at 25Gb/s using these cards

Incorporated September 13, 2016

Announced October 14, 2016

Mission

Create an open coherent high performance bus interface based on a new bus standard called *Open Coherent Accelerator Processor Interface (OpenCAPI)* and grow the ecosystem that utilizes this interface.

- Provide a forum to give the industry ability to innovate the next generation bus protocol
- Drive hardware/software innovation to enable choice and efficiency in data center architectures
- Build an ecosystem with the flexibility to build servers and data centers best suited for their computational demands

OpenCAPI Consortium Next Steps



JOIN TODAY!

www.opencapi.org