Datacenter CPUs and SoCs with OmniXtend fabric

Zvonimir Z. Bandic, Dejan Vucinic

Next Gen Platforms Technologies, Western Digital Corporation
Forward-Looking Statements

Safe Harbor | Disclaimers

This presentation contains certain forward-looking statements that involve risks and uncertainties, including, but not limited to, statements regarding: the RISC-V Foundation and its initiatives; our contributions to and investments in the RISC-V ecosystem; the transition of our devices, platforms and systems to RISC-V architectures; shipments of RISC-V processor cores; our business strategy, growth opportunities and technology development efforts; market trends and data growth and its drivers. Forward-looking statements should not be read as a guarantee of future performance or results, and will not necessarily be accurate indications of the times at, or by, which such performance or results will be achieved, if at all. Forward-looking statements are subject to risks and uncertainties that could cause actual performance or results to differ materially from those expressed in or suggested by the forward-looking statements.

Additional key risks and uncertainties include the impact of continued uncertainty and volatility in global economic conditions; actions by competitors; business conditions; growth in our markets; and pricing trends and fluctuations in average selling prices. More information about the other risks and uncertainties that could affect our business are listed in our filings with the Securities and Exchange Commission (the “SEC”) and available on the SEC’s website at www.sec.gov, including our most recently filed periodic report, to which your attention is directed. We do not undertake any obligation to publicly update or revise any forward-looking statement, whether as a result of new information, future developments or otherwise, except as otherwise required by law.
Agenda

• Datacenter CPU vision
• OmniXtend backplane reference design
• OmniXtend vs. other memory centric concepts
• OmniXtend system design:
  – Programmable Ethernet switches
  – Providing cache coherence
  – Dataplane implementation
• It works: two sockets and a switch...
• Performance measurements
• Conclusions and what is next?
Datacenter CPU vision I

• Multi-threaded, multi-core CPU:
  – 1) Medium performance, **OOO RISC-V Core** for general purpose OS and software applications
  – 2) Standardized and open JEDEC interface architecture (NVDIMM-P) for high density emerging non-volatile memories
  – 3) Support for standardized memory protocol fabric – e.g. **OmniXtend**
  – 4) Support for high bandwidth and low latency accelerator interfaces:
    • Supporting machine learning and inference engine accelerators
  – 5) Inference accelerator:
    • May contain large number of **small RISC-V Cores** with vector ALUs (similar to GP-GPU)
    • May contain set of hardware primitives allowing optimal acceleration
• Allows large number of RISC-V compute nodes to connect to universally shared memory (NUMA) – standardized

• Enables memory appliance, aggregation/disaggregation
OmniXtend backplane reference design
OmniXtend backplane reference design II

• Barefoot Networks Tofino programmable switch:
  – 64 4x25G ports
  – Supports P4 Programmable Pipeline

• On board Interfaces
  – 16x QSFP28 connectors for 100G (4x25G)
  – 4x QSFP28 connectors for 100G (4x25G) with retimers
  – 6x SFF-TA-1002 connectors w/ 8x25G
  – 1x SFF-TA-1002 connectors w/ 16x25G
  – 4x Firefly connectors for 4x25G Optical MTP/MPO
  – 12x 4x25G over TwinAx flyover cables for extension boards

• Extension boards
  – 2 Extension boards
  – 12x EDSFF SFF-TA-1002 connectors w/ 8x25G on each board
  – Alternate configurations possible
OmniXtend vs. other memory-centric concepts

• Memory fabric may mean different things to different people:
  – Page fault trap leading to RDMA request (incurs context switch and SW overhead)
  – Global address translation management in SW, leading to LD/ST across global memory fabric
  – Coherence protocol scaled out, global page management and no context switching

Context switch cost comparable to memory access latency

Require software/kernel support and/or rewriting of applications

This is OmniXtend
No rewriting of software, scalable like the algorithm
Computer and storage server architecture

Last 40 years

- Single node heavily dominated by Intel
- Top tier Google, Amazon, MSFT have access to some proprietary interfaces
- We could not get QPI/UPI license since first attempt in HGST (2013) (Zvonimir, Dejan)
- Architecture: CLOSED
- No guarantees – not even PCIe or SATA
- Amount of memory per node is limited
Data centers’ memory are struggling to keep up with data explosion

By 2020 IDC predicts the data volume will have reached 40,000 Exabytes

I/O struggling to keep up with CPU innovations

Big data and fast data ecosystem requires a standard scale-out memory-centric architecture

General purpose architectures no longer sufficient

*Big Data and Fast Data workloads exceed capability of uniform resource ratios*
OmniXtend system design

- OmniXtend is a fully open coherence protocol meant to restore unrestricted interoperability of heterogeneous compute engines with a wide variety of memory and storage technologies
- OmniXtend supports a four-hop MESI protocol and takes advantage of a new wave of Ethernet switches with stateful and programmable data planes to facilitate system scalability
The auspicious timing of programmable Ethernet switches made the choice of transport easy

• Current trends towards modern networking supports 6.4 Tb/s packet switching with application specific header parsing & customized match-action

• Programmable network devices provide:
  ➢ Programmable data plane and control plane
  ➢ High throughput (100Gbps) with low processing latency (<0.5μs)
  ➢ Data processing while being transmitted
  ➢ Protocol-independent

• Great Benefits for:
  ➢ Distributed Systems
  ➢ In-Network Computing
  ➢ Distributed Main Memory and Storage Systems
Protocol independent switch architecture enables coherence message routing over Ethernet

- OmniXtend leverages the programmability of smart switches for parsing, processing and routing coherence messages between computing nodes to suit cache coherence protocol's needs.
Providing cache coherence

• All of the memory operations on a shared address space are completed through two-stage request and response transactions between active participants:
  – Master Agent: request for specific permission and perform memory operations
  – Slave Agent: manage access permissions, communicate with main memory, and respond to the original requester

• A Master Agent must first obtain necessary permissions on a specific block before performing memory operations:
  – Get: Read the data from a specified address
  – Put: Write the data at a specified address
  – Atomic: Atomically read a data value and write a new value that is the result of some arithmetic or logical operations
Data plane implementation

- Replaces Ethernet L2 with serialized TileLink messages
  - Keeps standard 802.3 L1 frame, interoperates with Barefoot Tofino and future Ethernet switches
  - Custom frames are parsed and processed in P4 language
  - Enables stateful message processing inside the switching fabric
  - FPGA or ASIC switch; not limited to 802.3
- Protocol translation and modification inside fabric:
  - Requires no new silicon
- 100 Gb/s is available today
  - Clear roadmap to 200 and 400 with 56Gb PAM4 and x8
Routing OmniXtend messages through a programmable switch

• Barefoot Tofino ASIC:
  - 64-port 100 GigE switch, 6.4 Tbit/s aggregate throughput, < 400 ns latency
  - Fully programmable parser allows describing TileLink message format in P4
  - Fully programmable match-action pipeline (a.k.a. “flow tables”) enables flexible coherence message processing to suit cache coherence protocol's needs at line-rate performance
  - Modifications to coherence domains, protocols require no new silicon
Performance measurements

System Setup

SiFive
RISC-V SoC with OmniXtend running in FPGA

BAREFOOT NETWORKS
Tofino Switch programmed with P4 code to support OmniXtend
Performance measurements

Memory access latency

Average cache line access latency (50 MHz uncore)
RISC-V system with 8 harts
Conclusions

• Realization of 7yr+ dream: open source cache coherence protocol and interconnect, and at a low cost

• True enablement of plug-n-play heterogenous compute:
  – Some more RISC-V ISA work and other ISA work may be needed!

• The backplane:
  – Allegro files are available to CHIPS Alliance members

• What’s next:
  – Compute and memory boards – CHIPS Alliance project 2020
Cache Coherence

Multi-Core CPU

PCle Root Complex

Core
Cache
Shared
Cache
Core
Cache
Core
Cache
Core
Cache
Shared
Cache
PCIe Root Complex
Home Agent and Memory Controller

System

PR = processor read
PW = processor write
BR = observed bus read
BW = observed bus write
S/~S = shared or not shared

Modified (M)
Exclusive (E)
Shared (S)
Invalid (I)

CPU Socket 1

CPU 1
Core 1
Control
ALU
L1 Cache

Core 2
Control
ALU
L1 Cache

L2 Cache

CPU Socket 2

CPU 2
Core 1
Control
ALU
L1 Cache

Core 2
Control
ALU
L1 Cache

L2 Cache

Coherency

DRAM

Network Interface

Storage Interface

SATA
SAS

HDD

NVMe

SSD

DDR

DRAM
OmniXtend™: A Cache Coherent Memory Fabric

Western Digital and SiFive collaborating on extension of TileLink cache coherency protocol over ethernet.

- Leverages ubiquitous Ethernet
- Routable and switchable fabric
- Scalable as Ethernet speeds increase

OmniXtend initial specification and FPGA bitstream are now live at https://github.com/westerndigital corporation/omnixtend
RDMA Networking latency

- As a direct memory transfer protocol RDMA is already suited to accessing remote byte-addressable persistent memory devices
- Typical latency for RDMA reads: 1.6-1.8us
  - For PCIe attached RNICs on host and target nodes
- Simultaneous PCIe tracing reveals as little as 600ns spent actually transferring data over the network
  - 70% of latency is spent in memory or PCIe-protocol overhead
- How much can be saved by attaching memory directly to network fabric (i.e., no PCIe overhead)
  - Maximal benefit (~1.0 us) would require integrated RNIC interfaces on both Initiator and Target
RDMA to persistent memory latency

• Raw RDMA access to remote PCM via PCIe peer2peer is 26% slower than to remote DRAM
Providing cache coherence

Memory block states

• **Modified**: The node has the only valid cached copy which contains dirty data with read and write permission

• **Exclusive**: The node has read and write permission on a cached copy which is valid and clean

• **Shared**: The node has read-only permission on a shared cached copy which is valid, clean

• **Invalid**: The node has no permission on that block
Providing cache coherence

Coherence messages

- A Master Agent must first obtain necessary permissions on a specific memory block through transfer operations to perform read and/or write operations on that specific memory block
  - None
  - Read
  - Read+write

- Memory operations are performed through exchanging coherence messages between master and slave agents:
  - **Acquire**: A request message from a Master Agent to obtain appropriate permissions on that address along with a copy of that data block
  - **Grant Data**: A response message from a Slave Agent to acknowledge the reception of an acquire and grant permission to the original requesting Master Agent along with a copy of the data block
  - **Release**: A request message from a Master Agent to downgrade its permissions on a cached copy
  - **Probe**: A request message from a Slave Agent to query, modify or revoke the permissions of a cached copy stored by a particular Master Agent
Providing cache coherence

State transitions

(a) Upgrading permissions and transmissions from I to S to perform read operation

(b) Upgrading permissions and transmissions from I/S to E to perform read and write operation

(c) Downgrading the permission on a black and state transmission from M/E to I to relinquish that copy back
OmniXtend memory centric system

RISC-V node

# cat /proc/cpuinfo
hart: 1
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 2
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 3
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 4
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 9
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 10
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 11
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo
hart: 12
isa: rv64imafdc
mmu: sv39
arch: sifive, rocket0

# cat /proc/cpuinfo | grep hart | wc -l