The Enzian Coherent Interconnect (ECI): opening a coherence protocol to research and applications

Abishek Ramdas, David Cock, Timothy Roscoe, Gustavo Alonso
first_name.last_name@inf.ethz.ch
Dept. of Computer Science, ETH Zurich
Switzerland

1 INTRODUCTION

The rise of heterogeneous computing platforms composed of CPUs and hardware accelerators has thrown open the design space for hardware and software architectures. The field is moving fast, often driven by immediate product wins and early standards. This means that existing systems exercise complex and somewhat arbitrary trade-offs between programmability, performance, time-to-market, and energy efficiency based on how the accelerator is connected to the CPU and the memory model that is provided.

This is particularly true for FPGAs, whose execution model is often constrained by the platform and interconnect design. FPGA-based acceleration offers the chance to radically rethink the hardware/software interface, but today only two contrasting models predominate - the FPGA as PCIe-attached accelerator without coherence [2, 4, 8, 10], and closer, cache-coherent integration (CCIX [5], Gen-Z [9], Intel's CXL[6]) using newly-defined interoperabilty protocols.

An FPGA cache-coherent with a CPU has some immediate advantages: consistent access to shared data structures is easier, for example. However, all the interconnects above "black-box" the coherence protocol. Although FPGAs are very different from conventional cores, there is limited discussion to date on how coherency can be exploited by FPGAs, and whether it might be usefully customized to applications.

An FPGA has no cache. A flexible coherence implementation would serve a wide variety of purposes: simplifying development, enabling CPU monitoring from the FPGA and real time processing of the instrumentation data, using the FPGA as a sophisticated memory controller, bridging the cache coherency control across machines through the FPGA to, e.g., implement disaggregated memory, etc. Exploring these ideas is not possible in today's systems.

Here we report on opening up a cache coherence protocol for tailoring by applications, to enable a deeper exploration of the design space that commercial platforms allow. The context for our work is *Enzian* [1, 7], a research machine built in our group to explore heterogeneous computing options in a way less constrained by emerging standards. Enzian (Fig. 1) combines a server-class ARM CPU with a large FPGA (both with ample RAM and I/O) using the CPU's native coherence protocol.

LATTE '21, April 15, 2021, Virtual, Earth

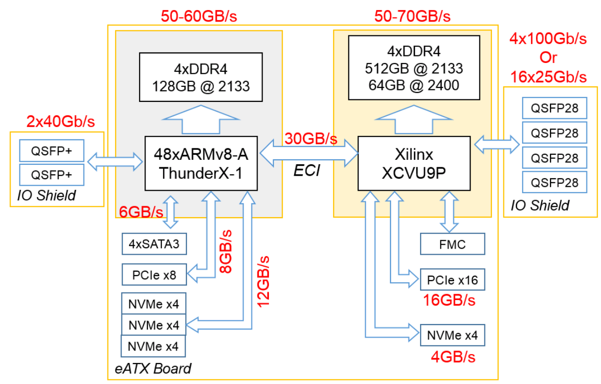

Figure 1: Enzian

Enzian is novel in breaking the traditional dichotomy between cache-coherent NUMA nodes and non-coherent (or DMA-coherent) I/O accelerators, through the *Enzian Coherent Interconnect* or ECI: our FPGA implementation of the CPU's native coherence protocol. ECI exposes the protocol to applications at the message level, opening up a range of new uses that go far beyond traditional cache coherence.

Through ECI, applications on the FPGA side can monitor and issue coherence messages directly to the CPU, thereby altering both how the CPU views FPGA memory and how it caches CPU-local RAM. While ECI includes a "vanilla" implementation of the protocol layer allowing plug-and-play coherence for any attached byte-addressable memory, its design adds a much richer set of operations to FPGA user logic: coherent caching/non-caching reads and writes, atomics, shootdowns, etc.

While we are not the first to consider new uses for a coherency protocol (see e.g. *PBerry*[3], where an FPGA supports remote memory by monitoring local coherence traffic), ECI enables direct interaction with the coherence protocol.

A simple but powerful example of an ECI usecase is maintaining coherent *logical views* over physical memory to software. The FPGA can transform and prefetch memory accesses to provide e.g. a row-store and column-store view on the same in-memory database relation, and ensure that writes to one cached view propagate to the other in the same CPU cache. However, in addition to non-traditional execution models for applications, developers can also use the control ECI provides to implement other functionality such as instrumentation, cache monitoring, and fault injection for software on the CPU.

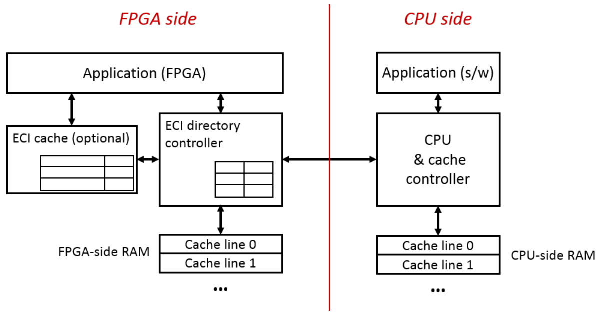

Figure 2: ECI flexible directory controller

2 IMPLEMENTING ECI IN ENZIAN

What is unusual about ECI is that our directory controller has an open, extensible structure allowing applications on the FPGA side to exploit the coherence protocol in new ways. The state space of the protocol layer can thus be modified to suit the needs of an application.

ECI thus provides not a single protocol, but a *family* of protocols interoperable with the CPU's existing proprietary protocol: To the CPU, an ECI node appears to be a full MOESI-based directory controller and cache (even if a particular instatiation implements only a subset of this, or manipulates the protocol in a nonstandard manner). The underlying transport protocol was robustly designed, and we benefited from close consultation with the original designers.

An ECI controller is generated from a machine readable specification, and verified against a comprehensive model of the CPU protocol validated through extensive trace-based testing. ECI is an extensible architecture, comprising tracing, parsing and monitoring tools alongside the validated specication and a reference implementation for a fully-coherent FPGA memory space with no local cache. Tools are included for the runtime verification of ECI protocol layers against formal temporal logic specifications.

The reference implementation includes a directory to track (40 bit) FPGA-owned addresses cached on the CPU, together with a pair of cache-line-sized AXI4 interfaces: One for coherent memory access from FPGA applications, and the other to permit coherent (CPU or FPGA) accesses to be routed as the user desires e.g. to DRAM or to any user logic presenting a read/write interface at a granularity of 128 bytes. The reference controller supports independent concurrent tracking of separate cache lines, with full support for out of order issue and completion of both coherence and DRAM transactions, including bursts. FPGA applications can directly initiate cache maintainance operations, such as evictions and writebacks.

The reference implementation is easily integrated via stanard interfaces, and customisable, permitting the easy exploration of application-oriented coherence policies.

3 STATUS AND PLANS

The ECI architecture and reference implementation open up a number of exciting research directions, some immediate and some longer-term. The reference implementation permits the immediate prototyping of ideas such as FPGA-side data transformations (e.g. row-column views), or the benefits of fine-grained coherent access to CPU-cached data from the FPGA.

Longer-term possibilities are extensive: FPGA-side monitoring of CPU traffic, exploration of the optimal coherence domain (or domains), forwarding packet data from Enzian's 100GbE interfaces without touching DRAM, the implementation of capability-based firewalls for a cluster-scale coherent system, any many more that both our group and others intend to explore.

As of writing, a number of complete Enzian systems are undergoing final acceptance testing, and the reference directory controller has been implemented and is itself now undergoing validation. We are in the process of performing detailed benchmarks to establish the performance characteristics of the system. We also intend to formally verify our reference implementation. Both Enzian itself and ECI will be released as open source, along with all supporting documentation. The hardware itself will be available either for remote access or purchase by interested groups.

ACKNOWLEDGMENTS

We would like to thank Xilinx, Cavium/Marvell, and VMware for their generous help with this project, and the entire Enzian team for input and feedback on this work. In particular, Mohsen Owaida did much initial implementation of ECI.

REFERENCES

[1] Gustavo Alonso, Timothy Roscoe, David Cock, Mohsen Ewaida, Kaan Kara, Dario Korolija, David Sidler, and Zeke Wang. 2020. Tackling Hardware/Software co-design from a database perspective. In *CIDR 2020, 10th Conference on Innovative Data Systems Research, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2020/papers/p30-alonso-cidr20.pdf

[2] Amazon Web Services [n.d.]. *AQUA (Advanced Query Accelerator) for Amazon Redshift*. Amazon Web Services. https://pages.awscloud.com/AQUA_Preview.html

[3] Irina Calciu, Ivan Puddu, Aasheesh Kolli, Andreas Nowatzyk, Jayneel Gandhi, Onur Mutlu, and Pratap Subrahmanyam. 2019. Project PBerry: FPGA Acceleration for Remote Memory. In *Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS 2019, Bertinoro, Italy, May 13-15, 2019*. ACM, 127–135. https://doi.org/10.1145/3317550.3321424

[4] Adrian M. Caulfield, Eric S. Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, Daniel Lo, Todd Massengill, Kalin Ovtcharov, Michael Papamichael, Lisa Woods, Sitaram Lanka, Derek Chiou, and Doug Burger. 2016. A cloud-scale acceleration architecture. In *49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2016, Taipei, Taiwan, October 15-19, 2016*. IEEE Computer Society, 7:1–7:13. https://doi.org/10.1109/MICRO.2016.7783710

[5] CCIX Consortium [n.d.]. *CCIX - Cache Coherent Interconnect for Accelerators*. CCIX Consortium. http://www.ccixconsortium.com

[6] Stephen Van Doren. 2019. Compute Express Link. In *2019 IEEE Symposium on High-Performance Interconnects, HOTI 2019, Santa Clara, CA, USA, August 14-16, 2019*. IEEE, 18. https://doi.org/10.1109/HOTI.2019.00017

[7] ETH Zurich [n.d.]. *Enzian, a research computer*. ETH Zurich. https://www.enzian.systems/

[8] Daniel Firestone, Andrew Putnam, Sambrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian M. Caulfield, Eric S. Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert G. Greenberg. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, Sujata Banerjee and Srinivasan Seshan (Eds.). USENIX Association, 51–66. https://www.usenix.org/conference/nsdi18/presentation/firestone

[9] Gen-Z Consortium [n.d.]. *Gen-Z*. Gen-Z Consortium. http://genzconsortium.org/

[10] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati,

Joo-Young Kim, Sitaram Lanka, James R. Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2014. A reconfigurable fabric for accelerating large-scale datacenter services. In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*. IEEE Computer Society, 13–24. https://doi.org/10.1109/ISCA.2014.6853195