Low-latency user-level communication for RISC-V clusters

Charisios Loukas[1]*, Pantelis Xirouchakis[1], Michalis Gianioudis[1], Aggelos Ioannou[1,2], Manolis Katevenis[1] and Nikos Chrysos[1]

[1]Computer Architecture and VLSI Systems laboratory, Foundation for Reasearch and Technology - Hellas
[2]Lawrence Berkeley National Laboratory, University of California

Abstract

Within the context of the RED-SEA project, we integrate novel low-cost interconnect technologies with open-source low-power, RISC-V processors. We present and measure a design that achieves sub-microsecond user-level latency on small packet generation and transmission between adjacent RISC-V cluster nodes.

Introduction

Low-latency inter-node communication is important in HPC clusters. Following the conclusion of the ExaNeSt[1] project, which studied the adoption of low-cost, power-efficient ARM processor clusters for Exascale-class systems, as part of the RED-SEA project, we swap the ARM processors for RISC-V. In this work[1], we tightly couple a lean, low-latency network interface with a modified Ariane RISC-V soft core.

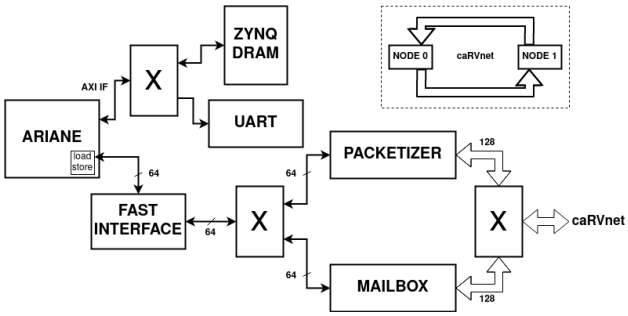

Figure 1: *Node architecture*

Methodology

Architecture The system used for the measurements consists of 2 TE0808 Trenz boards (with the option of adding more), each hosting a Xilinx XCZU9EG MPSoC and connected in a ring node topology, using 10Gbps transceivers. The design programmed on each node includes an Ariane RISC-V core, running Linux at 100 MHz. The core, as seen in Figure 1, is connected via AXI4 to a Packetizer and a Mailbox module[2], by means of a custom load/store interface that is tightly integrated into the core. This interface consumes specific ld/st operations at pipeline speed, and hands them to the AXI. It issues early acknowledgements, allowing back-to-back store instructions, so that transfer descriptors are efficiently produced.

Both the packetizer and the mailbox are peripherals of the custom HPC interconnect, named caRVnet. The caRVnet Packetizer is a virtualized peripheral that accepts descriptors for small transfers from the Ariane core and outputs small caRVnet packets, which may target local or remote mailboxes. The caRVnet Virtualized Mailbox implements multiple FIFOs, using FPGA BRAMs, in order to enqueue the incoming payload of packets. It receives packets of a predefined maximum size from the caRVnet packetizer, which can then be dequeued by the Ariane core using load commands. To allow use of the packetizer-mailbox hardware in user space, a user space library and a set of hardware drivers were implemented.

Measurement Software Latency between adjacent nodes is measured in one of the two nodes by a user space program which implements a ping-pong test. Each node sends a 32 byte message to a remote mailbox and waits for a response. This happens repeatedly, with time measured using the RISC-V timers, by means of the `rdcycle` assembly command, which only costs a minimal 2 Clock Cycles overhead per iteration. Instead of measuring the time of multiple iterations,

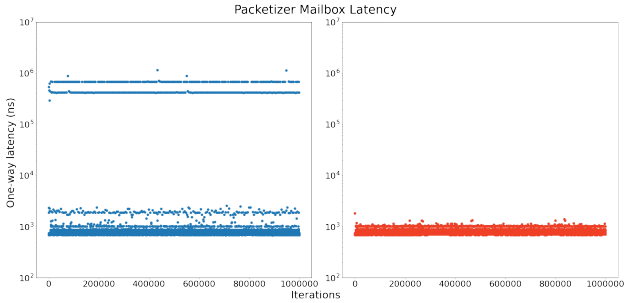

Figure 2: *Per iteration one-way latency measurements with and without interrupts*

*Corresponding author: cloukas@ics.forth.gr

[1] This work is part of RED-SEA project (grant No. 955776) part of the EU HPC JU, funded by the EU H2020 programme.

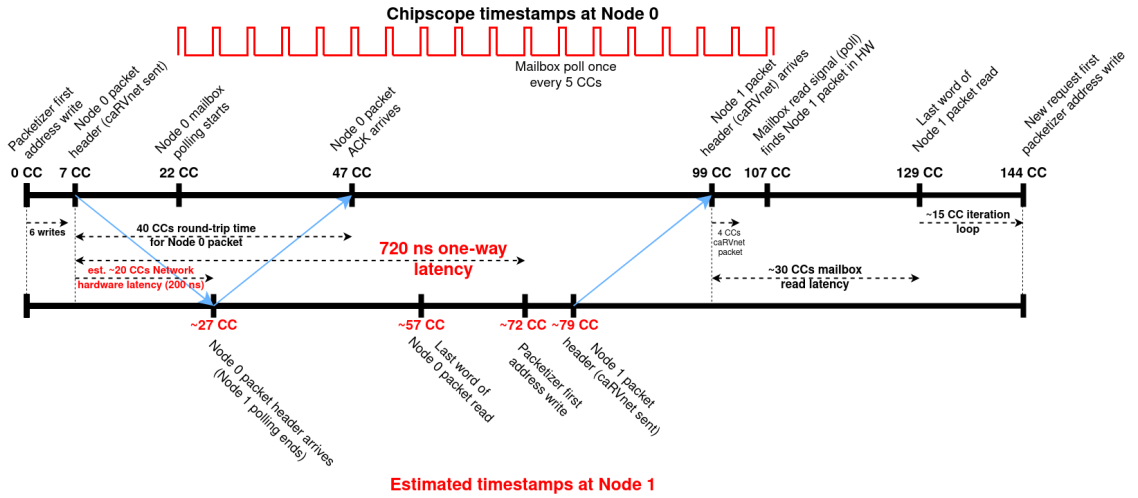

Figure 3: *Breakdown of two-way user level latency.*

we opted to measure and record the time of each iteration individually, in order to avoid loop overheads and study the delay variance across different iterations.

Results

Latency Measurements Running the measurement software initially returned an average round-trip latency of 1.86µs. This corresponds to an one-way latency of 0.93µs, which is already a satisfactory result, considering that the processor is running at 100 MHz.

On the left part of Figure 2, we depict the latency of individual iterations. As can be seen, the latency varies significantly, between 720ns and 500µs. We notice a concentration of values in 4 main plateaus. The overwhelming majority of the measurements is concentrated at 720ns. Occasionally the latency climbs up to 2000ns or even 500µs. We determined that this happens due to context switches, occurring in either of the two nodes. To cross-validate this result, we developed a hardware mechanism, which allows us to disable interrupts during measurements. The results can be seen in the right part of Figure 2, where latency does not exceed 1000ns. The **average one-way latency is 725ns**, with the majority of measurements at 720ns. A few reach up to 1000ns due to page walks, that occur while accessing the array which stores latency values.

Measurement breakdown Figure 3 breaks down the latency, based on hardware signals captured with Vivado Chipscope. The sequence of events is as follows:

1. **0 CCs:** Data starts being written in the packetizer memory on node 0.
2. **7 CCs:** Packet formation has been triggered and a header valid signal is active on the network.
3. **22 CCs:** The polling of the mailbox on node 0 has begun, with a polling period of 5 CCs.
4. **47 CCs:** Node 0 receives packet acknowledgement, 40 CCs after the packet was sent, indicating a 20 CC network traverse time.
5. **99 CCs:** The response packet header arrives in node 0, indicating the node 1 packetizer was triggered at 79 CCs (20 CCs network traverse).
6. **107 CCs:** The mailbox is polled in an active state 4 to 8 CCs after the header arrives, depending on time of arrival.
7. **129 CCs:** The last data of the response packet has been read.
8. **144 CCs:** The next packet begins being written on the Node 0 packetizer memory.

In total, we have 42 CCs software and 10 CCs of hardware overhead and a 200ns one-way network traverse, giving us the 720ns one-way latency.

Conclusion

In this paper we demonstrated the opportunities for low latency communication between RISC-V nodes, Our FPGA-based results suggest that this system, developed on ASIC with a 1 GHz clock can achieve one-way latency of 340ns.

References

[1] Manolis Katevenis et al. "Next Generation of Exascale-class Systems: ExaNeSt project and the status of its interconnect and storage development". In: *Microprocessors and Microsystems* 61 (May 2018). DOI: 10.1016/j.micpro.2018.05.009.

[2] Manolis Ploumidis et al. "Software and Hardware co-design for low-power HPC platforms". In: *High Performance Computing: ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers 34*. Springer. 2019, pp. 88–100.