EMiT 2019, Direct Communication Between Distributed FPGA Resources

Joshua Lant, Javier Navaridas

Advanced Processor Technologies Group
School of Computer Science
The University of Manchester

EMiT Emerging Technology Conference 2019, Huddersfield

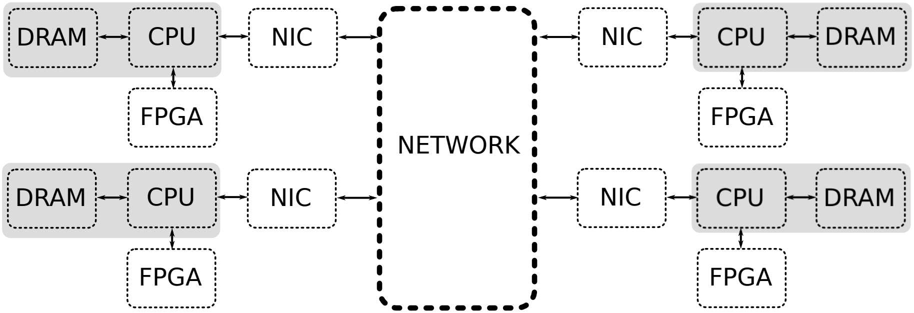
Overview

FPGAs for HPC, the need for direct communication

FPGAs for HPC

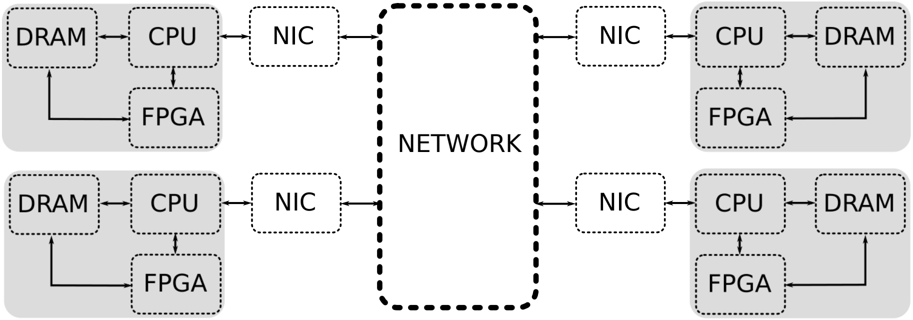
- Typically think of GPU as goto accelerator.
- Suitable for dataflow workloads or irregular parallelism.
- FPGAs can provide exceptional performance-per-watt.
- Reduced precision and custom data types.
- Improvements in memory bandwidth are good sign.

Bus Based Co-Processor



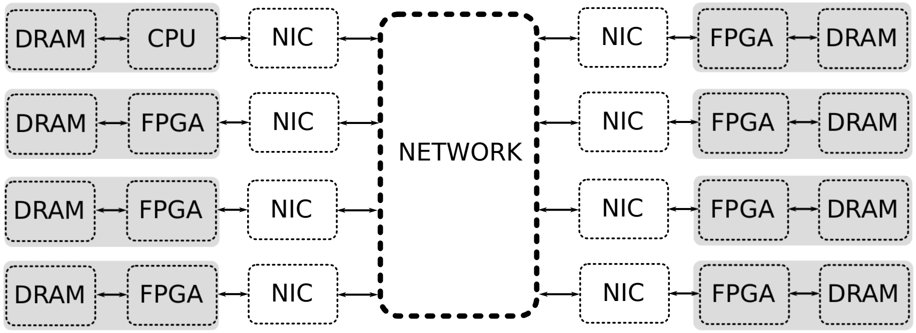
- e.g. PCIe bus.
- Network communication through the CPU, or separate FPGA network (point-to-point only).

System Bus, Shared-Memory



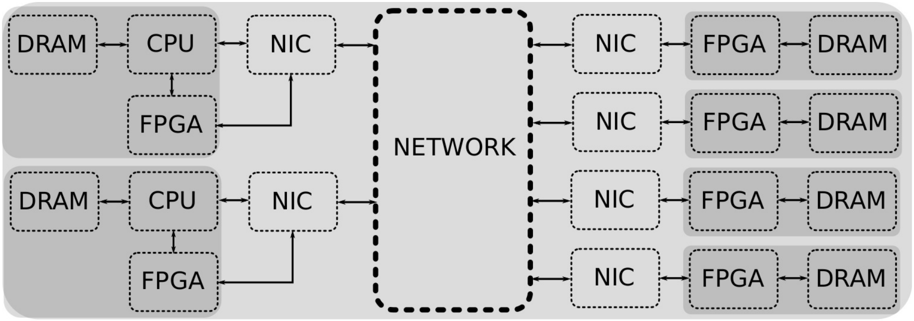
- New architectures, Xilinx Zynq Ultrascale+, Altera Stratix 10 (IOMMUs).
- Coherent access, tight coupling with CPU.
- Requires CPU for inter-FPGA transfer, reliable access to NIC.

Disaggregated Network Peer



- Allows communication between FPGAs without CPU involvement.
- TCP Offload Engines, non-scalable.
- Simpler solutions typically point-to-point.

Global Shared Memory Space



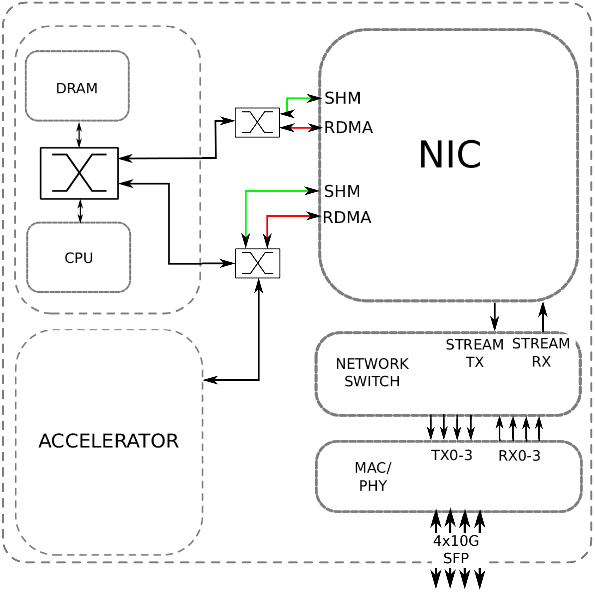
- All communication is to globally addressable location, direct to memory.
- Shared-memory access to remote nodes (NUMA).
- Traditional HPC communication (RDMA).
- Same communication method for CPU as FPGA.

Custom Network Interface

Our Solution

- Custom network interface and protocol
  - Addressed using geographic routing scheme.
  - Upper bits are node ID, lower is local memory address.
  - System bus protocol → network packet → system bus protocol.
- Novel transport layer
  - Completely hardware-offloaded.
    - Segregated transport mechanisms (RDMA or Shared Memory).
  - Connectionless (datagram) approach.
    - Keeps state information only about outstanding transactions, rather than persistent source-dest connections.
  - End-to-end reliability.

System Design



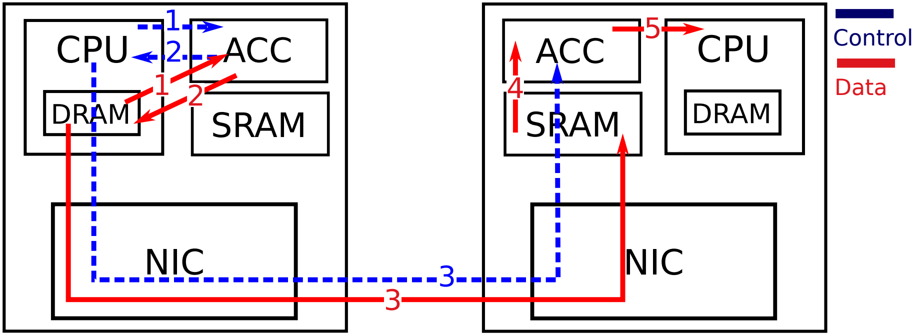
16B Latency Results

latency component	Shared memory (ACK'd)		RDMA (w/ notif.)	
	cycles	ns	cycles	ns
Total	**172**	**1101**	**232**	**1485**
Initial write- last flit at NIC output	24	154	69	442
Read from RAM	-	-	30	192
TX MAC in- RX MAC out	59	378	59	378
RX MAC out- Resp/Notif at TX MAC in	21	134	23	147
RX MAC out- Completion	9	58	22	371

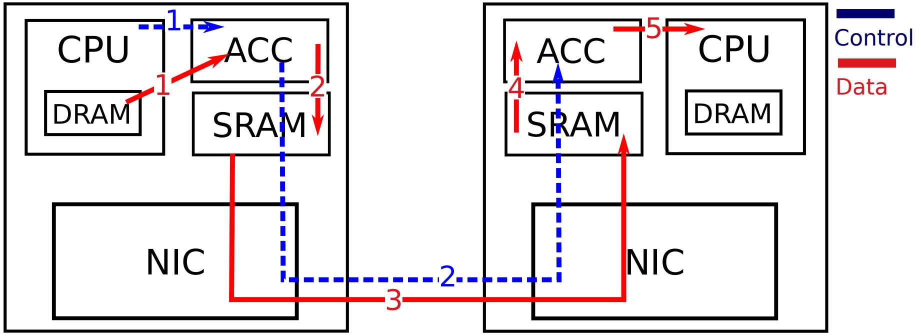
Using Distributed FPGA Resources

- SW based transport vs. HW offload
  - Software transport
    - Copy back to DRAM from Accelerator.
    - More complex control path.
    - CPU controls data movement.
  - Hardware offload
    - Low latency transfers.
    - Simple control path.
    - FPGA writes directly into remote memory.
- Block transfers to accelerator (512B-32KB).
- Implementation on ZCU102 development board.
- Transfers initiated from a user-space program.

SW Transport

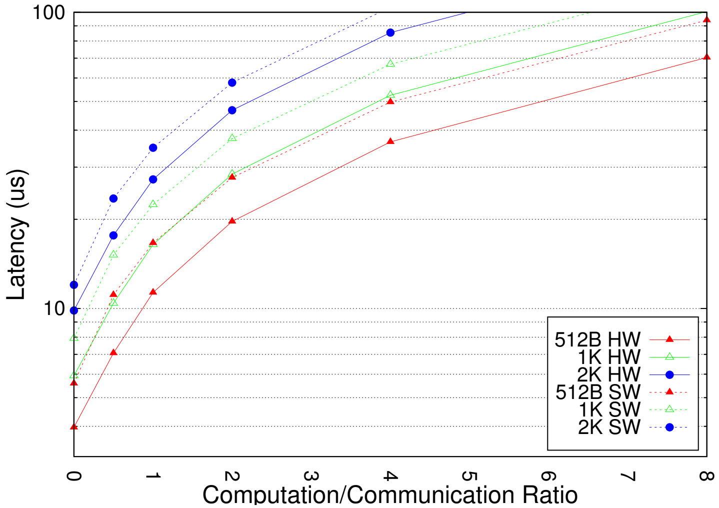


HW Offloaded Transport

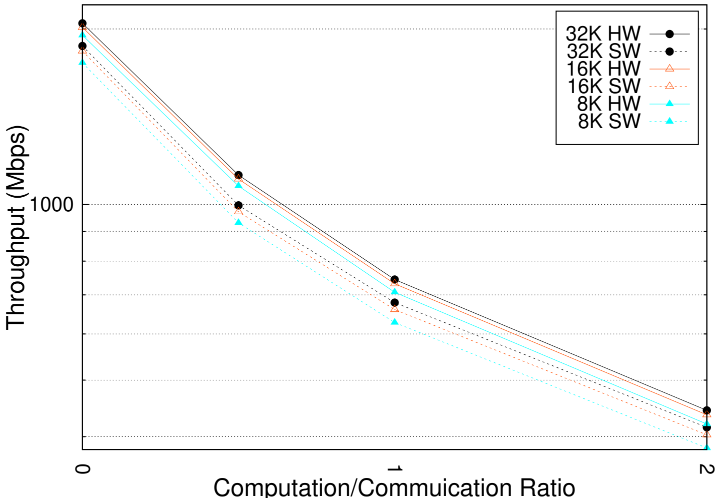


Results

Latency Results



Data Processing Throughput



Concluding Remarks

To Conclude...

- Hardware offloaded and connectionless transport is only solution to enable:
- Direct communications
  - Disaggregating FPGA from CPU resources.
- Tight memory coupling
  - Lower latency inter-FPGA communications
- Latency improvement of ≈29% for small block transfers.
- Throughput improvement of ≈9% for large block transfers.

EMiT 2019, Direct Communication Between Distributed FPGA Resources

Joshua Lant, Javier Navaridas

Advanced Processor Technologies Group
School of Computer Science
The University of Manchester

EMiT Emerging Technology Conference 2019, Huddersfield