Offloading Collective Operations to Programmable Logic

Martin Swany

(With thanks to Omer Arap)

Center for Research in Extreme Scale Computing (CREST)

Department of Intelligent Systems Engineering

Indiana University, Bloomington, IN

FPGAs Should Be Used As MPI Accelerators

I know MPI accelerators. We have the best MPI acclerators. - DT

Martin Swany

Center for Research in Extreme Scale Computing (CREST)
Department of Intelligent Systems Engineering
Indiana University, Bloomington, IN

Overview

- We have developed an architecture and implementation for offloading collective operations to programmable logic in the communication substrate
  - Not the first to use FPGAs for collectives
- It is clear that there is a tradeoff between performance and generality
  - FPGAs are at one end of that spectrum
  - Some folks live at the other end
- Programmable logic is burgeoning
  - Xilinx Zynq
  - Intel Xeon+Altera (HARP)
  - Mellanox ConnectX-4 + FPGA
  - FPGAs on other NICs and in switches

Programming FPGAs

- The programmable logic provided by FPGAs is a powerful option for creating task-specific functionality for applications
- Programming with FPGAs is not easy
- Many agree: Accelerate common functionality
- Others want to make it easy to use general-purpose toolchains to deploy arbitrary kernels
  - Jeff and OpenACC
  - Franck's workshop
  - New NSF call for ways to make it easy
    - It may not ever be easy

Things that go bump in the wire

- We have been exploring a (relatively) generic approach for scenarios in which there is programmable logic in the communication pipeline
  - Addresses the problem of how to get data to and from the device
- A bump in the wire
  - Rich Graham clearly got this phrase from us
- In teaching how to use FPGAs, we have found this model to be quite useful
  - Students who haven't done Verilog can get to this in a semester
  - Important topic in Indiana U.'s new engineering program
- This also maps to a growing use case

Collective Offload

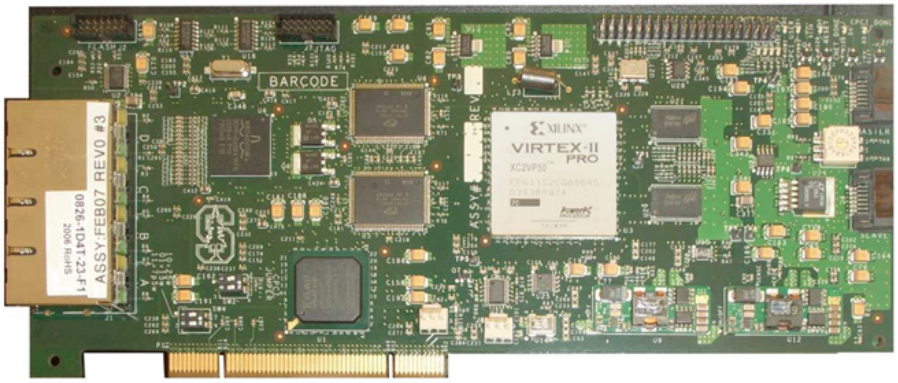
- Offloading collective logic is a common technique in various platforms to improve performance
- Mellanox's CORE-Direct is one such state of the art collective operation offload framework.
  - Defines primitive tasks: send, receive, wait, binary calculations.
  - Collective algorithm defines list of tasks and tasks are performed by the NIC without CPU involvement
- Collective offload systems have been done many times, but ours is easy to use and is a starting point

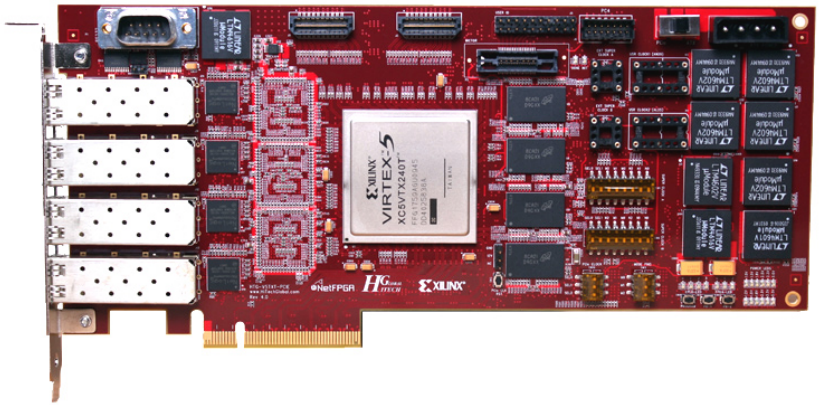
Goals of FPGA-based Offload

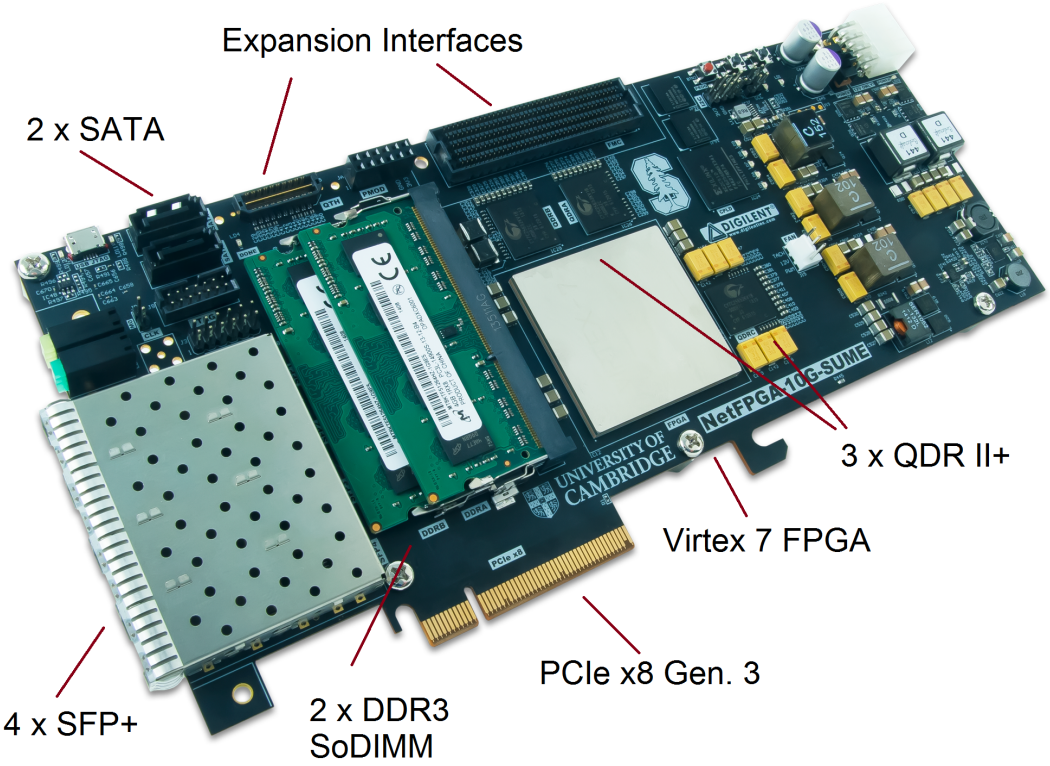
- Reduce collective operation latency and variance
- Provide selection of collective algorithms that are implemented in hardware
- Implement collective algorithms independent of end-to-end communication protocols
- Utilize hardware-level, protocol-independent multicasting to optimize, and when possible, redesign major collective operation algorithms
- Support collective operation offload for different classes of collective operations

NetFPGA

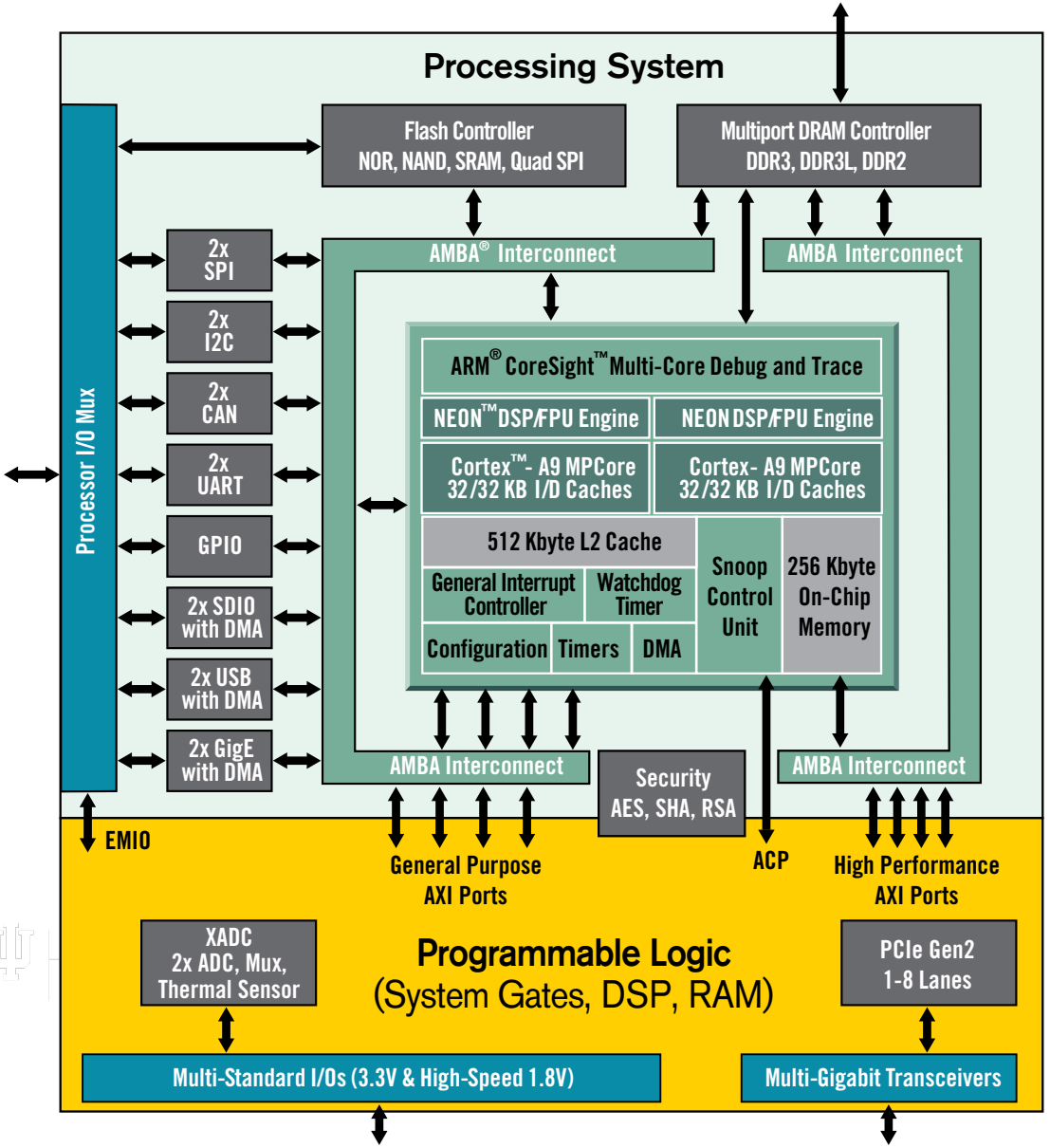
A line-rate, flexible, open networking platform for teaching and research



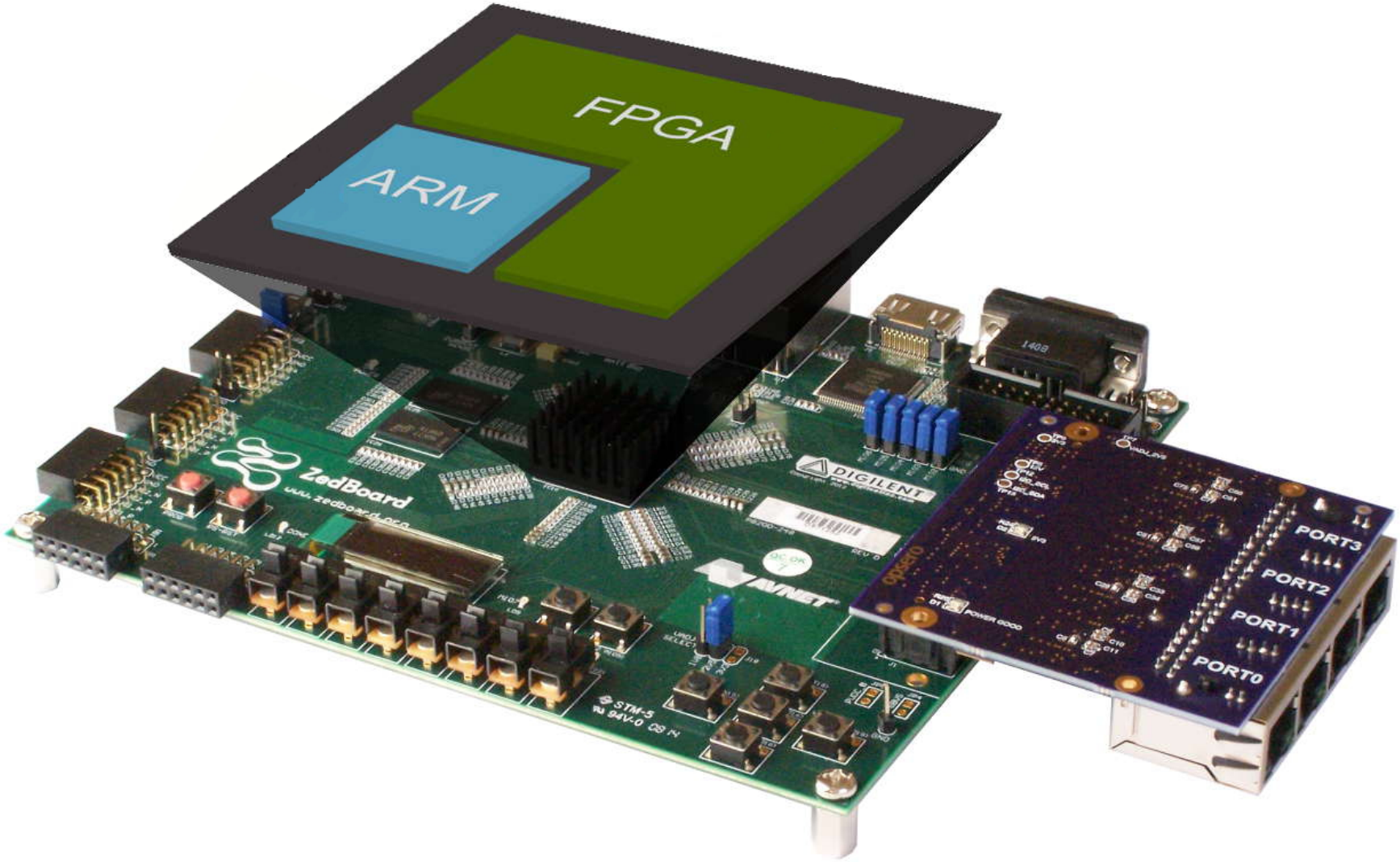




Xilinx Zynq


Processing System
Flash Controller
NOR, NAND, SRAM, Quad SPI
Multiport DRAM Controller
DDR3, DDR3L, DDR2
Processor I/O Mux
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
AMBA® Interconnect
AMBA Interconnect
ARM® CoreSight™ Multi-Core Debug and Trace
NEON™ DSP/FPU Engine
NEON DSP/FPU Engine
Cortex™- A9 MPCore 32/32 KB I/D Caches
Cortex- A9 MPCore 32/32 KB I/D Caches
512 Kbyte L2 Cache
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
Snoop Control Unit
256 Kbyte On-Chip Memory
AMBA Interconnect
AMBA Interconnect
Security AES, SHA, RSA
EMIO
General Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2x ADC, Mux, Thermal Sensor
Programmable Logic
(System Gates, DSP, RAM)
PCIe Gen2 1-8 Lanes
Multi-Standard I/Os (3.3V & High-Speed 1.8V)
Multi-Gigabit Transceivers

Zedboard & Ethernet FMC



Collective Offload Design

- Ease implementation with the “bump in the wire” model
- Host offloads collective to the Programmable Logic (PL) by sending a specially crafted UDP packet
  - Simplest solution
- Host blocks until the PL generates a release message, which is also a UDP packet

Collective Offload Packet

0..3	4..7	8..11	12..15	16..19	20..23	24..27	28..31	32..35	36..39	40..43	44..47	48..51	52..55	56..59	60..63
dst_MAC												src_MAC_1			
src_MAC_2								type				ver	IHL	Diff_Serv	
Total_Length				Identification				flags	fragment_offset			TTL		Protocol	
Hdr_Cksum				src_IP								dst_IP_1			
dst_IP_2				UDP_Source_Port				UDP_Dest_POrt				Length			
UDP_checksum				coll_id				comm_size				coll_type			
algo_type				node_type				msg_type				rank			
local_rank_count				operation				data_type				count			

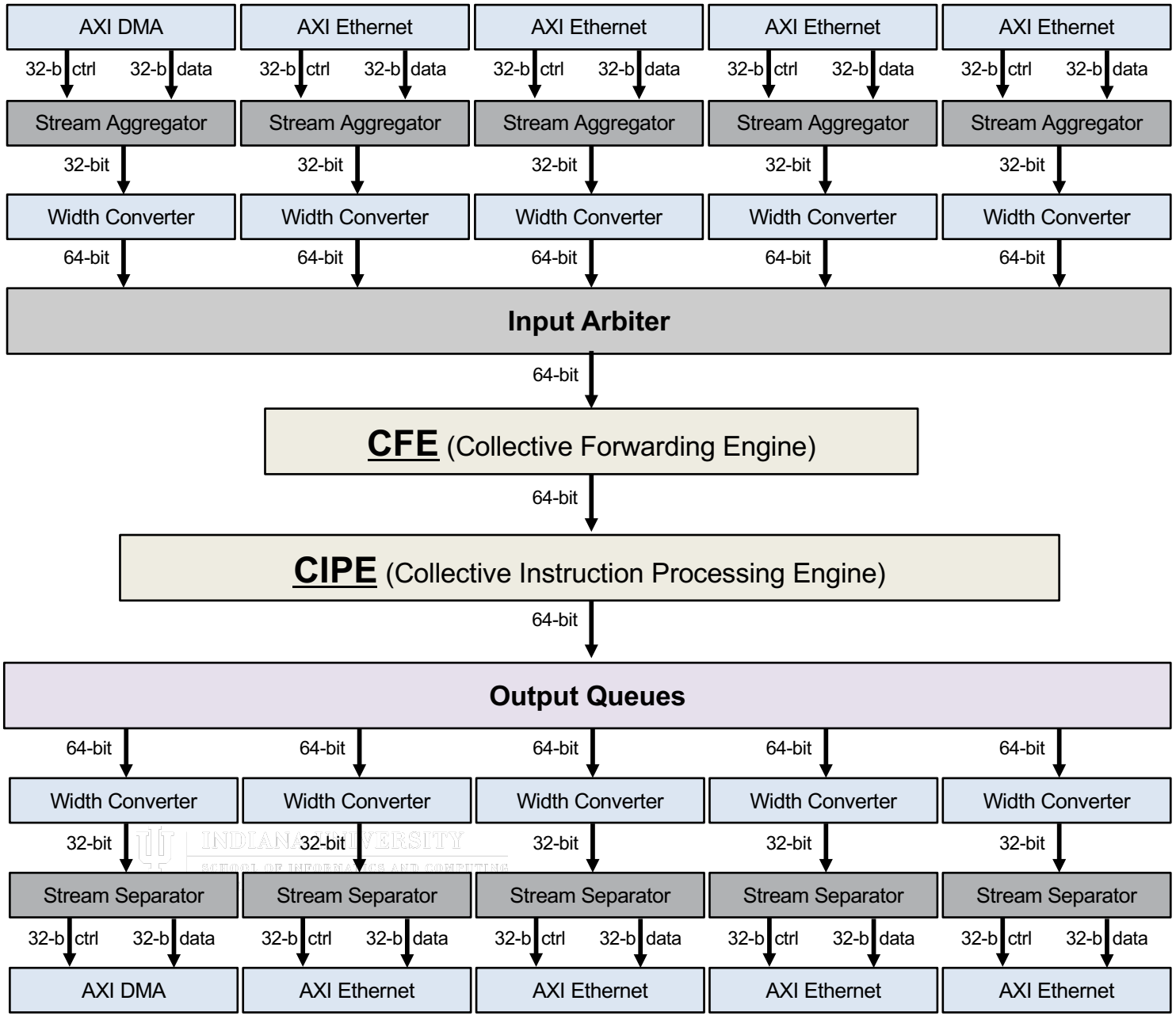
AXI-Based Platforms

- Advanced eXtensible Interface (AXI) standards define intermodule communication in programmable logic
- Widely utilized in recent Xilinx FPGA designs
  - Recent designs depend on AXI-based protocols
- AXIS (AXI-Stream) is for streaming data between modules.
  - We employ the AXIS protocol in our design
- Support for multiple ranks per host
  - Provides insight how our design scales

Implementation

- Modules inherited from the NetFPGA package
  - Input Arbiter, Output Queues
- Modules from Xilinx IP catalog
  - AXI Ethernet, AXI DMA, AXI Width Converter, FPU, AXIS Data FIFO
- New Modules
  - Stream Separator
  - Stream Aggregator
  - ***Collective Forwarding Engine (CFE)***
  - ***Collective Instruction Processing Engine (CIPE)***
- Implements MPI_Allreduce, MPI_Reduce, MPI_Barrier, MPI_Scan, MPI_Alltoall…

Data Path


AXI DMA
AXI Ethernet
AXI Ethernet
AXI Ethernet
AXI Ethernet
32-b ctrl
32-b data
Stream Aggregator
32-bit
Width Converter
64-bit
Input Arbiter
64-bit
CFE (Collective Forwarding Engine)
64-bit
CIPE (Collective Instruction Processing Engine)
64-bit
Output Queues
64-bit
Width Converter
32-bit
Stream Separator
32-b ctrl
32-b data
AXI DMA
AXI Ethernet
AXI Ethernet
AXI Ethernet
AXI Ethernet

Collective Forwarding Engine (CFE)

– Heart of the design

– Responsible for:

- Assessing the offload request header fields.
- Detecting collective operation and algorithm to run
- Perform state transitions and save the state for algorithms
- Based on the state, make forwarding decisions
- Determine the instructions which the CIPE will apply

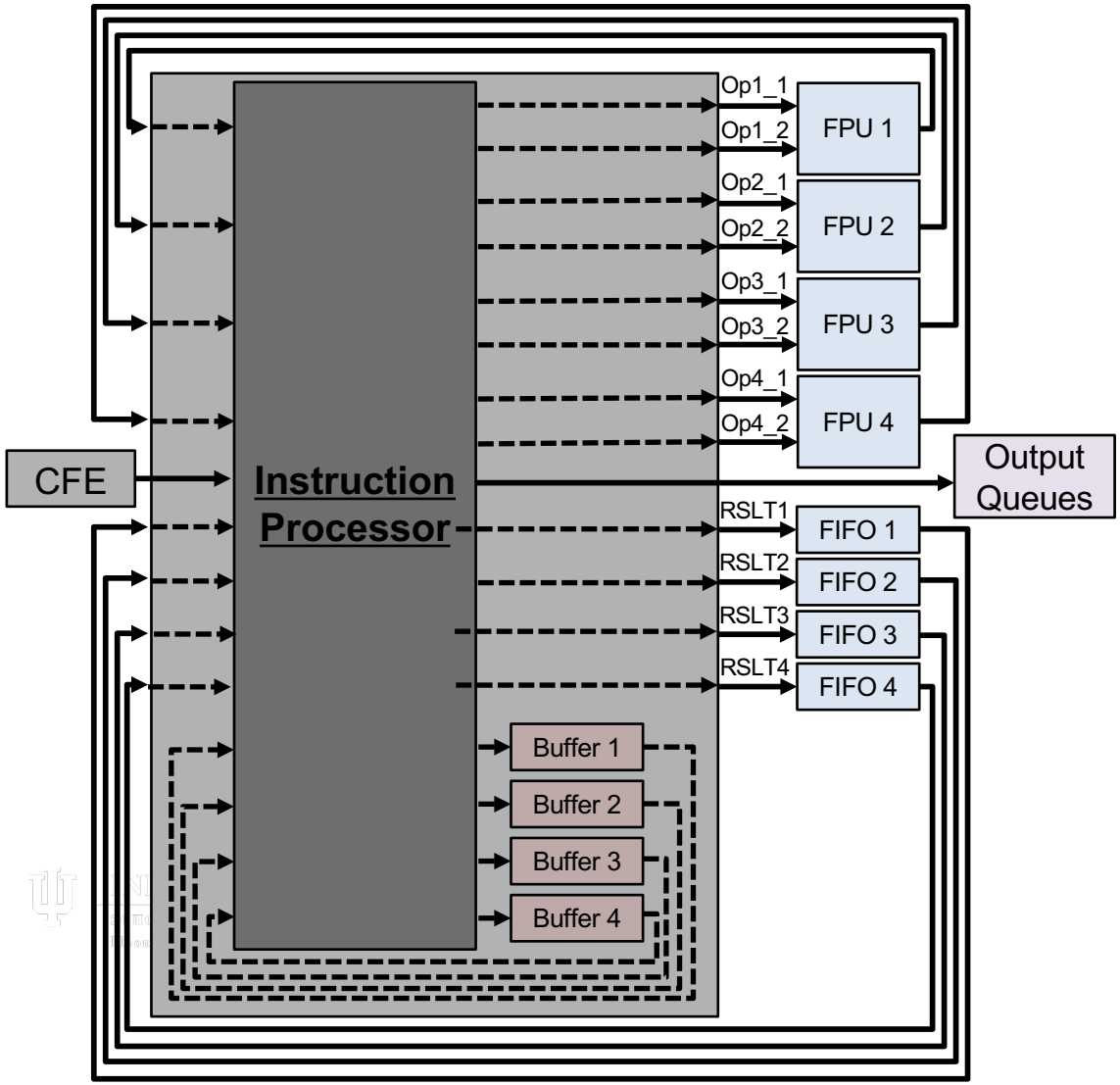
Collective Instruction Processing Engine (CIPE)

- CFE attaches 64-bit instruction header
- CIPE processes the instruction
- Example Instructions
  - Forward packet, do nothing
  - Buffer packet in specified buffer
  - Perform reduction operations on incoming packet and specified buffers
    - Buffer or not buffer the result
  - Perform reduction operations on specified buffers and ignore incoming packet
    - Buffer or not buffer the result
  - Ignore incoming packet and forward data on specified buffer

Collective Instruction Processing Engine (CIPE)

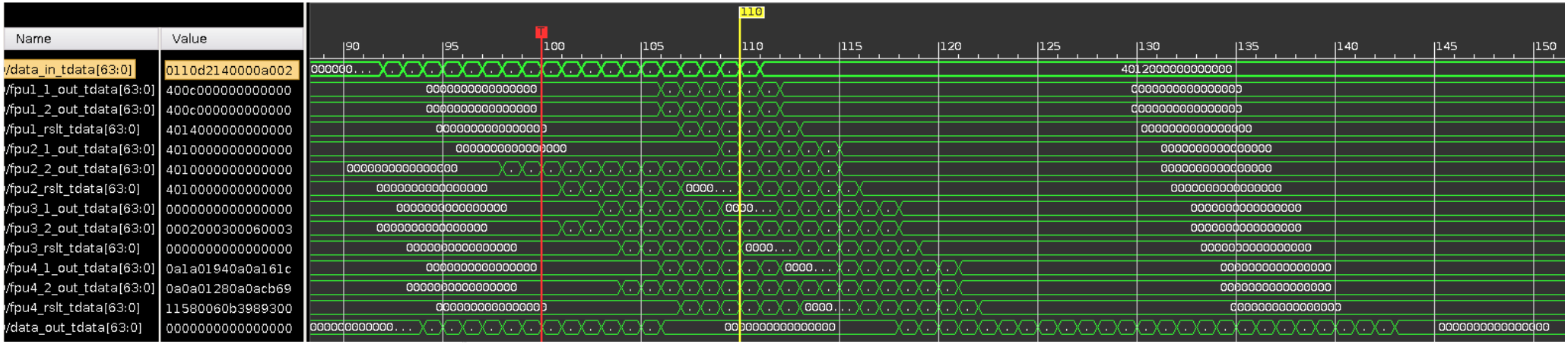
- Interfaces with FPU and FIFO cores
  - FPU cores for streaming and applying floating point arithmetic
  - FIFO cores for streaming intermediate results of single cycle arithmetic operations
- MIN, MAX, BOR, LOR, BAND, LAND, BXOR, LXOR, Integer SUM
- Internal BRAM buffers

Collective Instruction Processing Engine (CIPE)



Collective Instruction Processing Engine (CIPE)

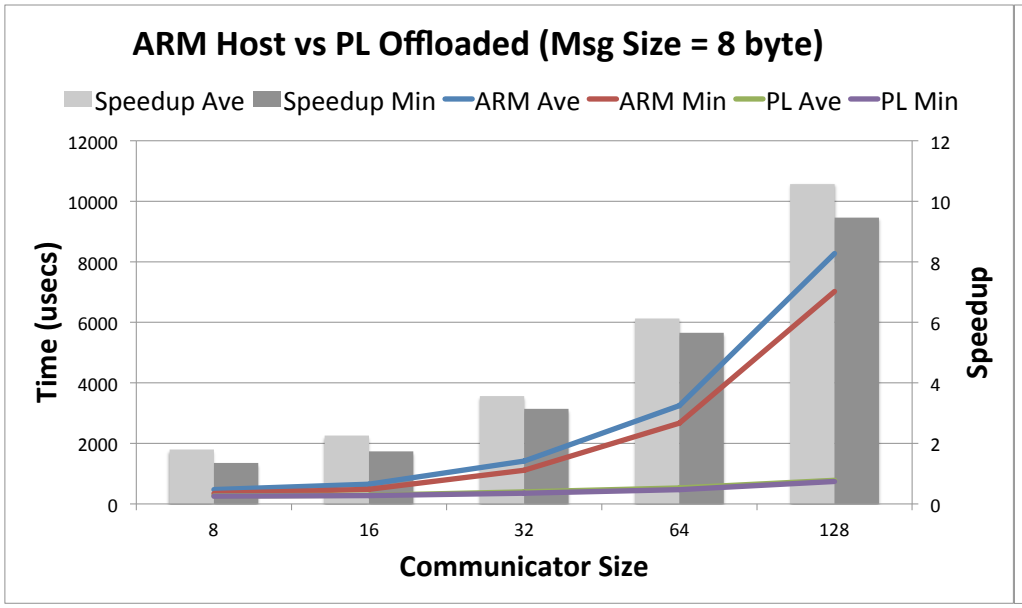
- Fully pipelined design
  - FPU cores can be configured to produce results in 1 cycle
    - Due to observed instabilities, we increased the first result cycle to 11 (the maximum)
  - Each extra FPU core introduces extra cycle
  - Each FIFO core employed introduces an extra cycle

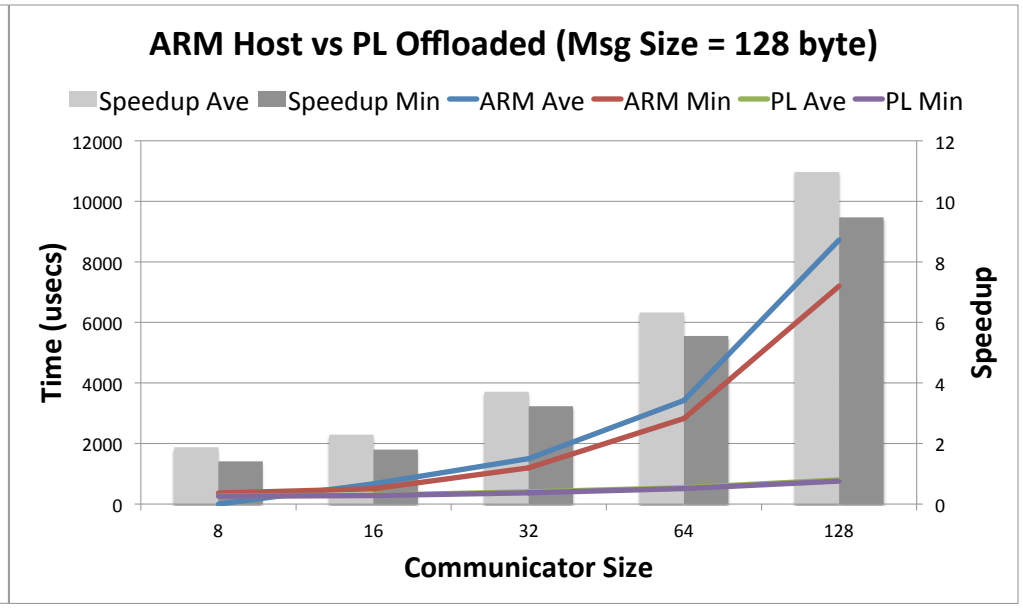


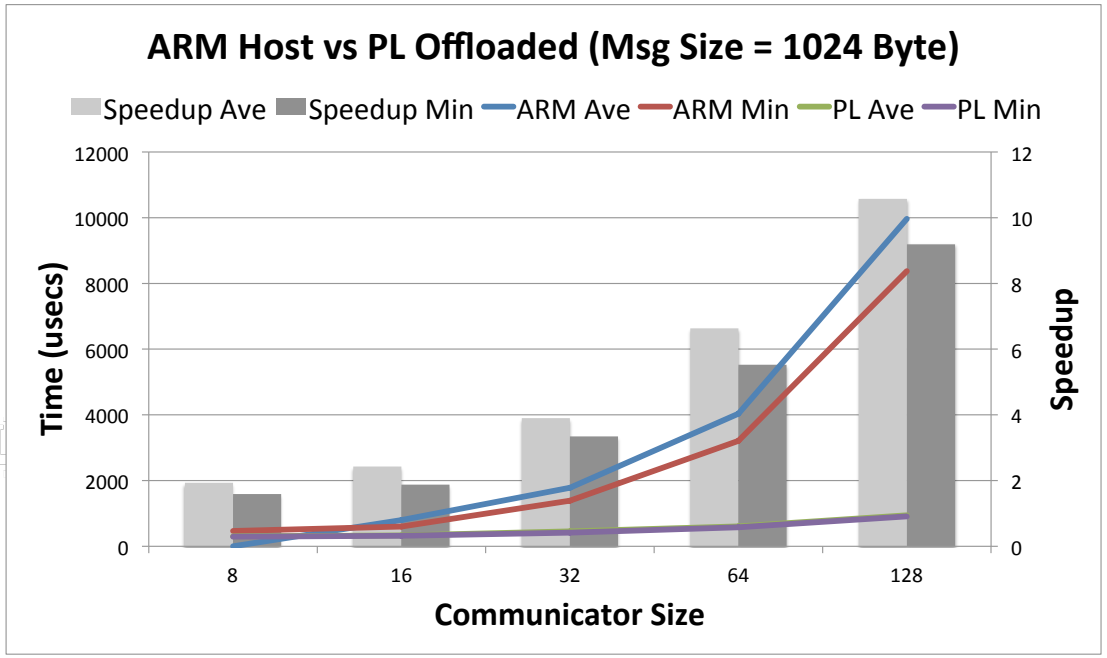
Evaluation

- Experimental Setup
  - 8 Zynq Zedboards with EthernetFMC adapters
    - Directly connected
- Come see it at SC

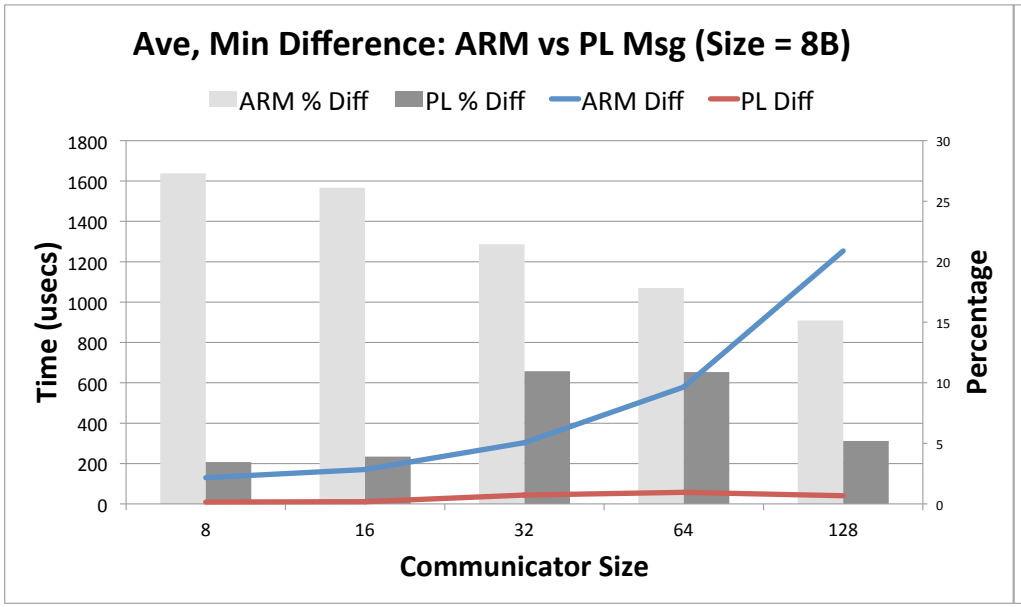
Zynq Cluster – MPI_Allreduce

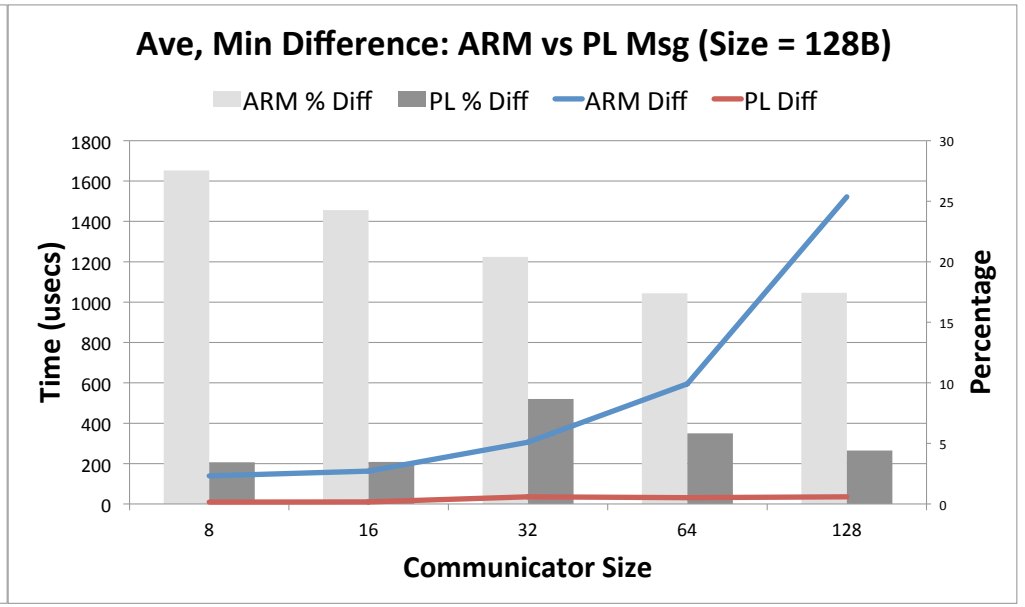


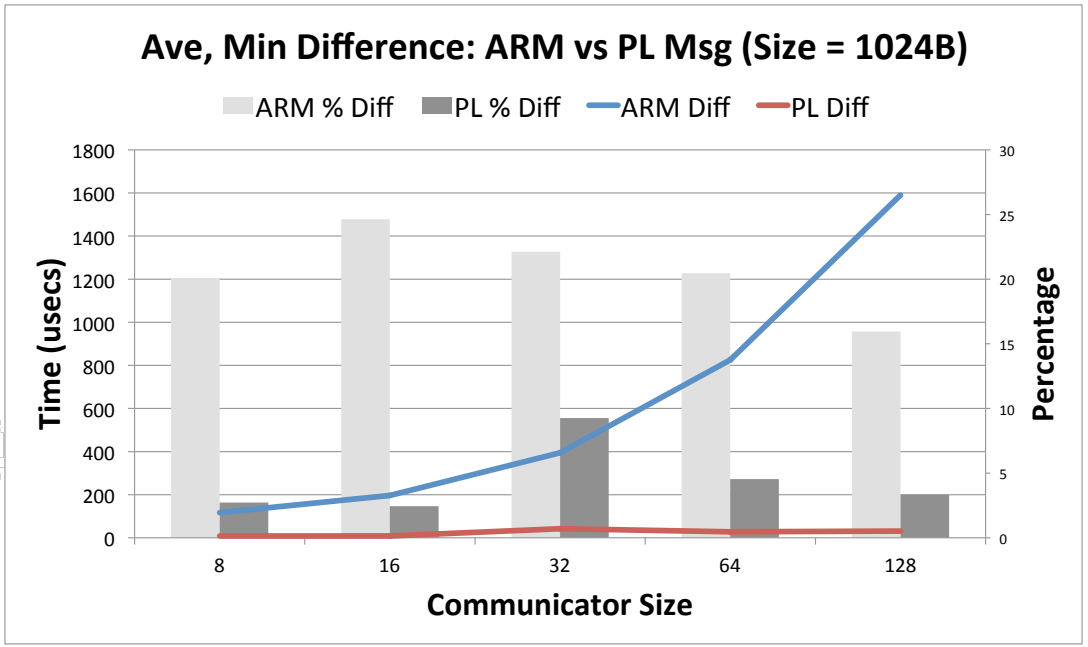




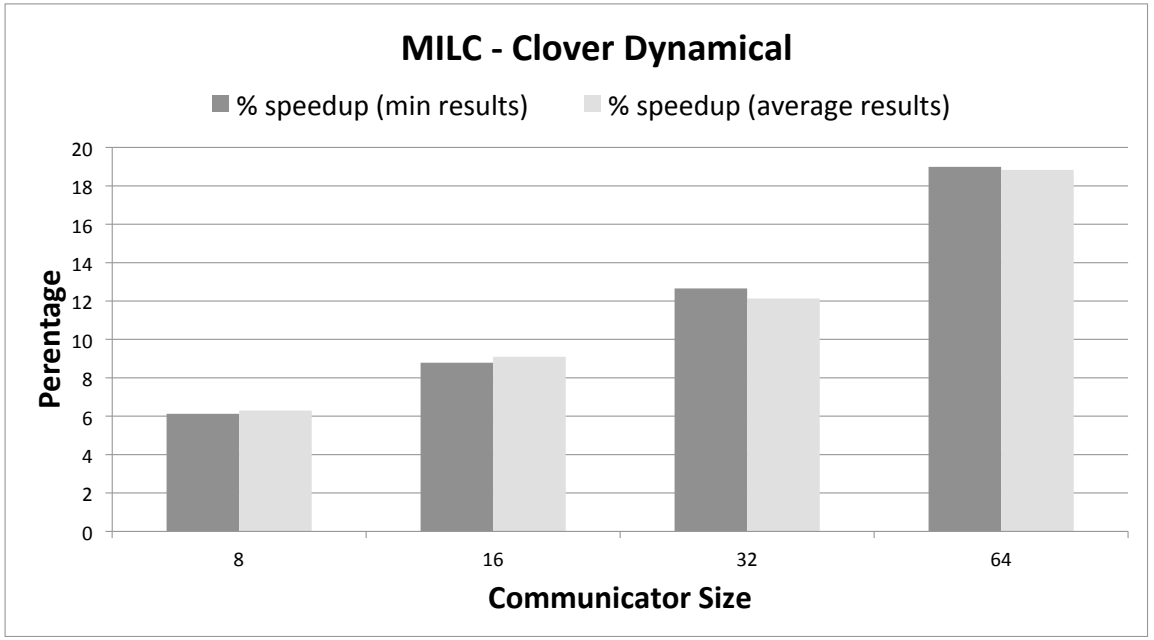
Zynq Cluster – MPI_Allreduce

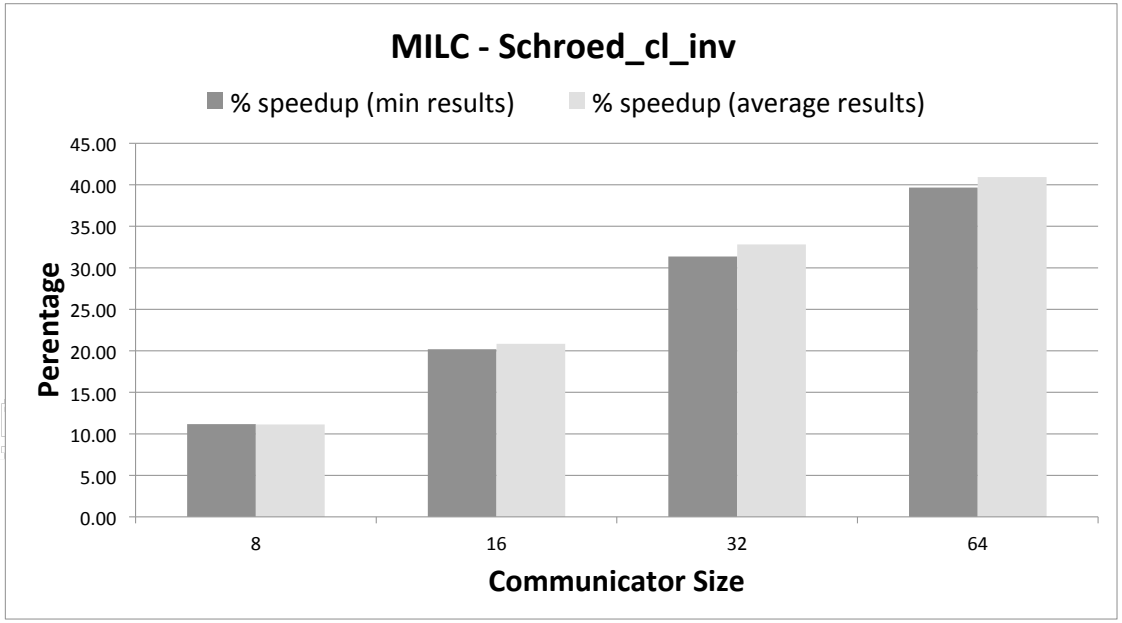






Zynq Cluster – MILC

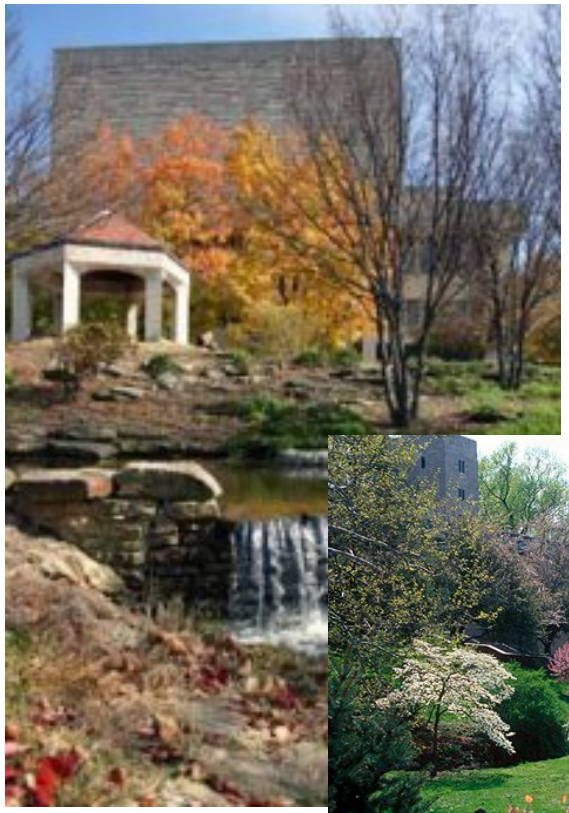


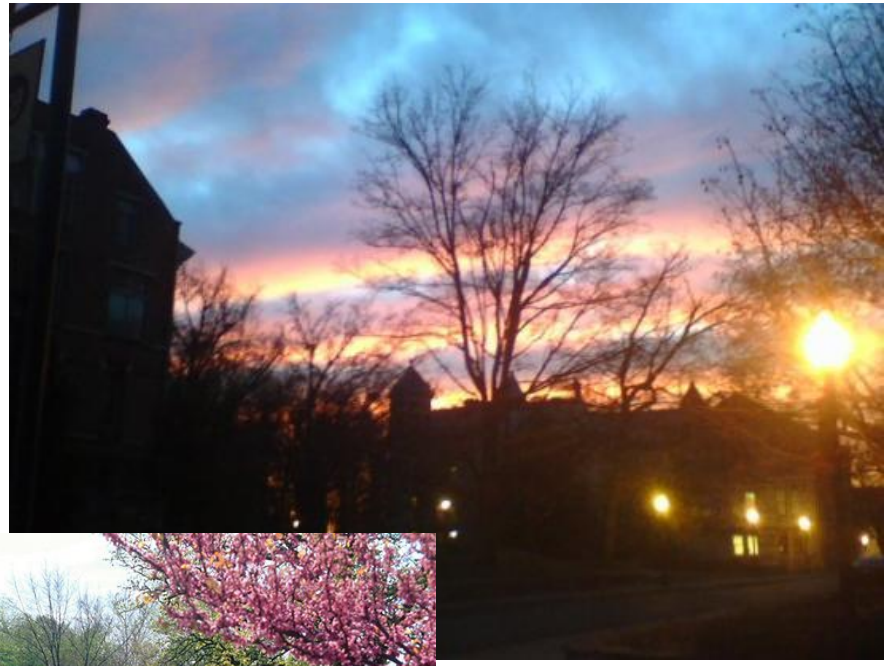


Conclusions and Future Work

- Empirical validation of the efficacy of collective offload
- Promising performance
- Increasing availability of programmable logic stands to increase impact
- The use of UDP allowed straightforward implementation, but is limiting
- Clearly, this the right way to use FPGAs


INDIANA UNIVERSITY






CardCow.com