Programming the Adapteva Epiphany 64-core Network-on-chip Coprocessor

Anish Varghese, Robert Edwards, Gaurav Mitra and Alistair Rendell

Research School of Computer Science
The Australian National University

May 19,2014

Outline

Introduction

Adapteva Epiphany Coprocessor

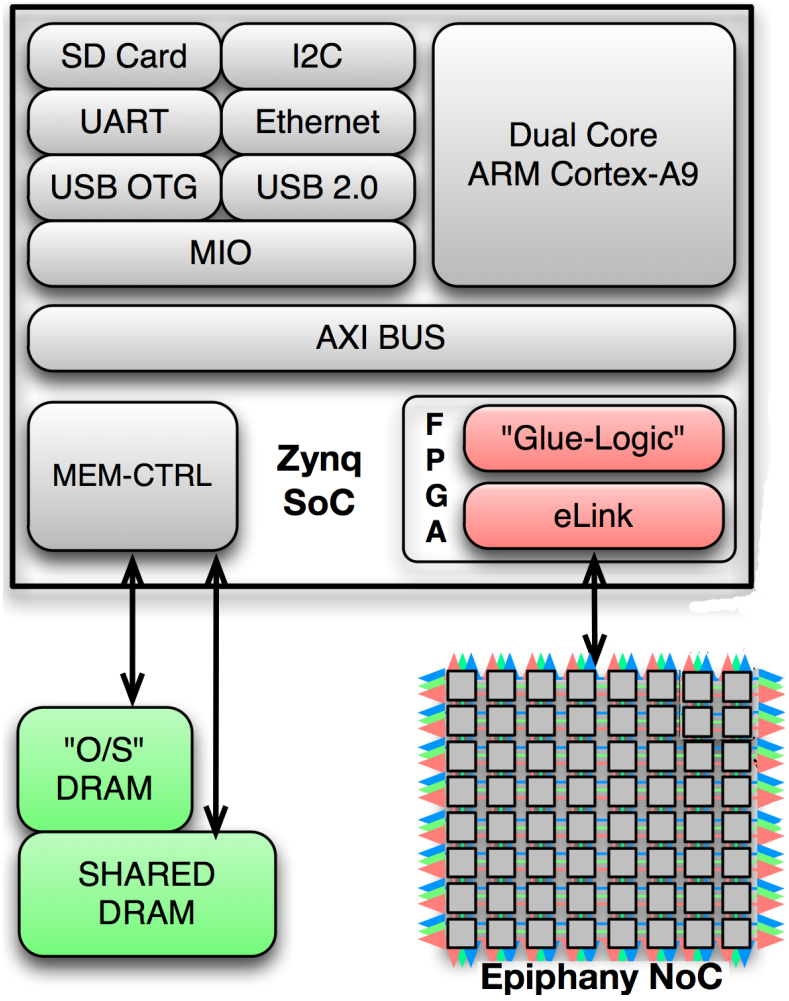
- New scalable many-core architecture
- Energy efficient platform (50 GFLOPS/Watt)
- $99 for a Parallella board

Contributions

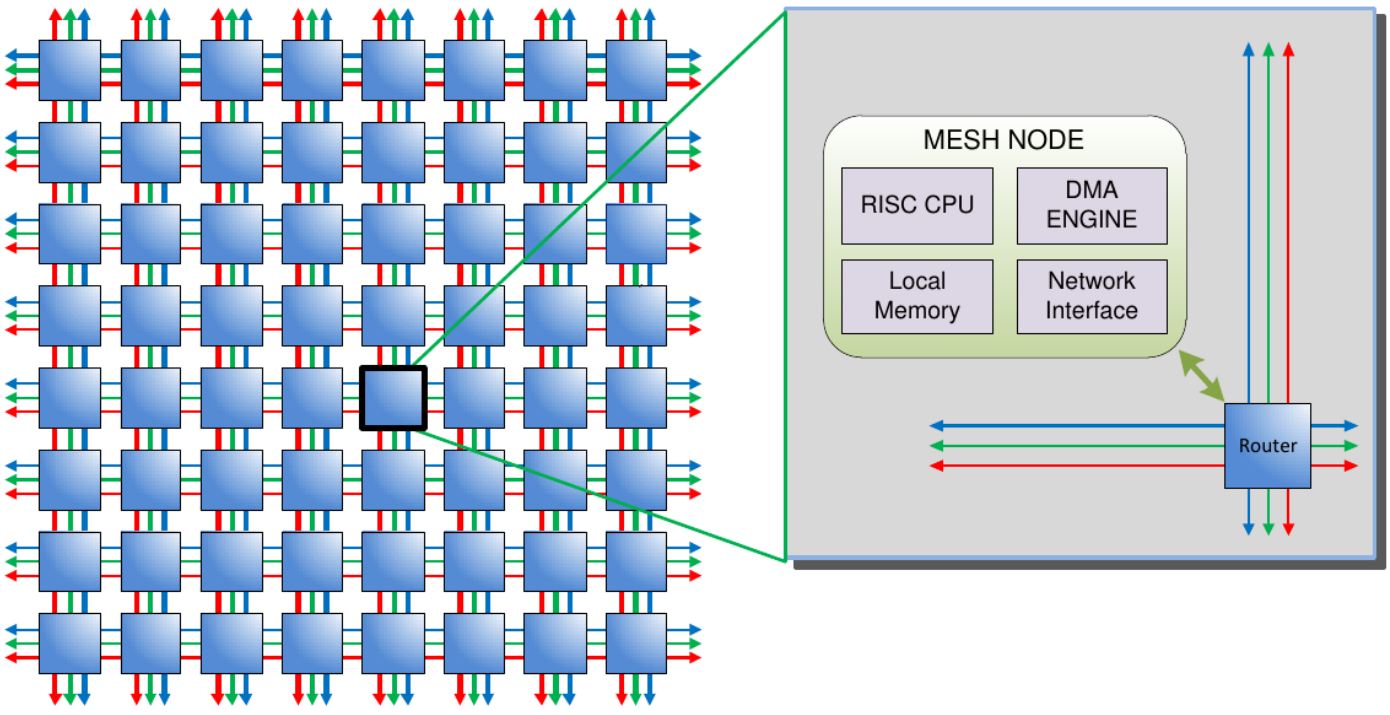
- Explored features of the Epiphany
- Evaluated the performance
- Demonstrated how to write high performance applications on this platform

Outline

Parallella Board



Epiphany Coprocessor



Features

- Multi-core MIMD architecture
- No cache
- 32 KB of local SRAM in four banks of 8 KB
- Shared address space
- 64 General purpose registers
- Epiphany Instruction set
- Superscalar CPU - two floating point operations (Fused Multiply-Add) and one 64-bit memory load/store operation

Outline

Software Environment

Programming Environment

- C/C++
- Epiphany SDK

Programming Considerations

- Memory Size
  - Relatively small 32 KB of local RAM per eCore (for storing both code and data)
  - Store code and data in different local memory banks
  - Distribute code between multiple cores
- Processor Capability
  - Currently no hardware support for integer multiply, floating point divide or double-precision floating point operations
  - Branching costs 3 cycles. Unroll inner loops to increase performance

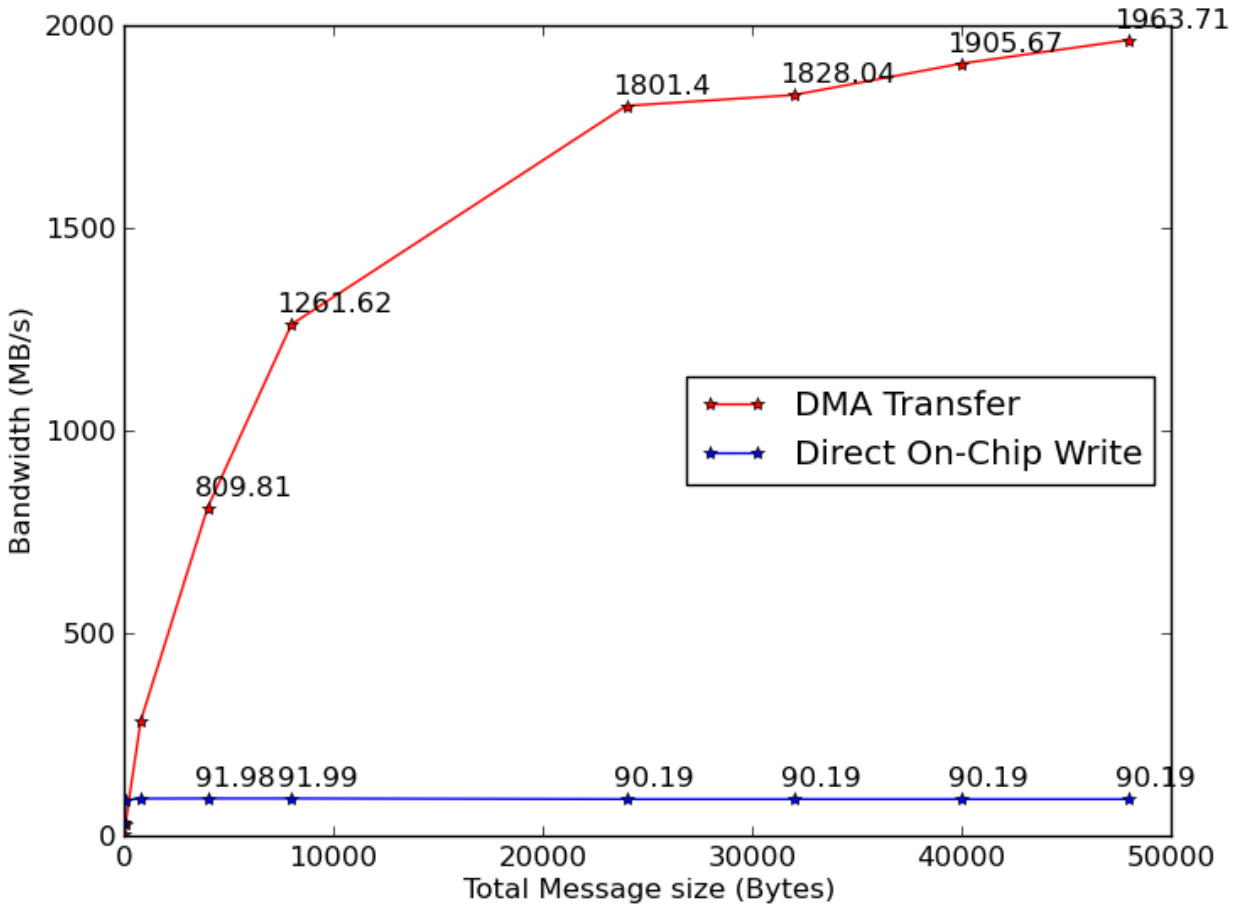
Outline

Experiment Platform

- ZedBoard evaluation module with Zynq SoC
- Daughter card with Epiphany-IV 64-core (E64G401)
- Dual core ARM Cortex-A9 host at 667 MHz
- Epiphany eCores at 600 MHz
- 512 MB of DDR3 RAM on the host
- 32 MB shared with eCores

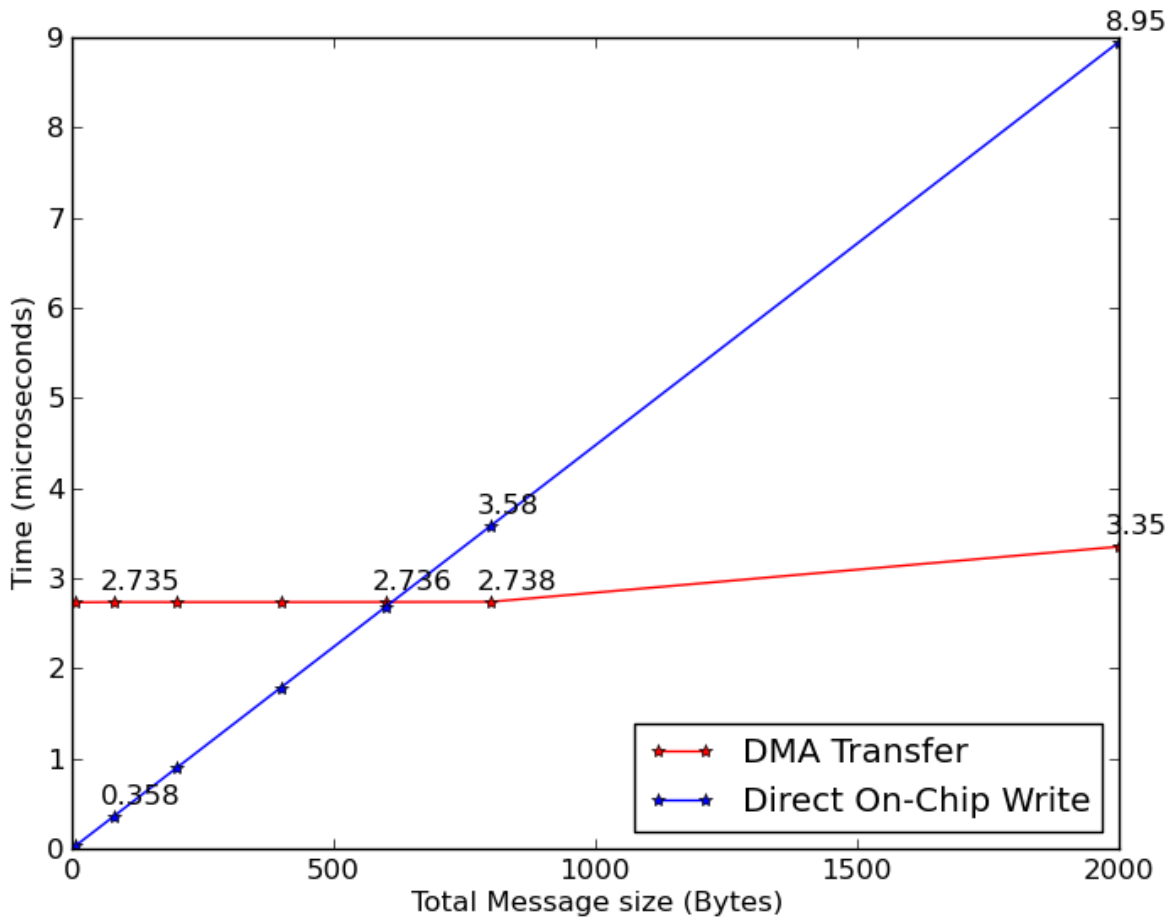
Bandwidth

Experiment: To evaluate cost of sending messages from one eCore to another



Latency

Latency for small message transfers



Latency

Experiment: To evaluate the effect of Node distance on Transfer Latency

Node 1	Node 2	Distance	Time per transfer (nsec)
0,0	0,1	1	11.12
0,0	0,2	2	11.14
0,0	1,2	3	11.19
0,0	0,4	4	11.38
0,0	3,3	5	11.62
0,0	4,4	6	11.86
0,0	7,7	14	12.57

- 80 bytes are transferred from one eCore to another
- $\approx$ 7 cycles per transfer

Outline

Shared Memory Access

Experiment: To evaluate the performance of the external shared memory, multiple nodes write to the shared memory simultaneously. Each eCore continuously writes blocks of 2 KBytes over 2 seconds and the utilization is measured

	Node (Total No)	Iterations	Utilization
2 * 2 nodes	0,0 0,1 1,0 1,1	61037 48829 24414 12207	0.41 0.33 0.17 0.08
8 * 8 nodes	0,7 1,7 2,7 3,7 (8) (4) (8) (9) (7) (24)	27460+ 3050+ 2040+ 100 - 1000 10 - 100 1 - 10 0	0.187 each 0.021 each 0.014 each

- Nodes closer to column 7 and row 0 get the best write access
- Write throughput of 150 MB/sec

Outline

Heat Stencil Equation

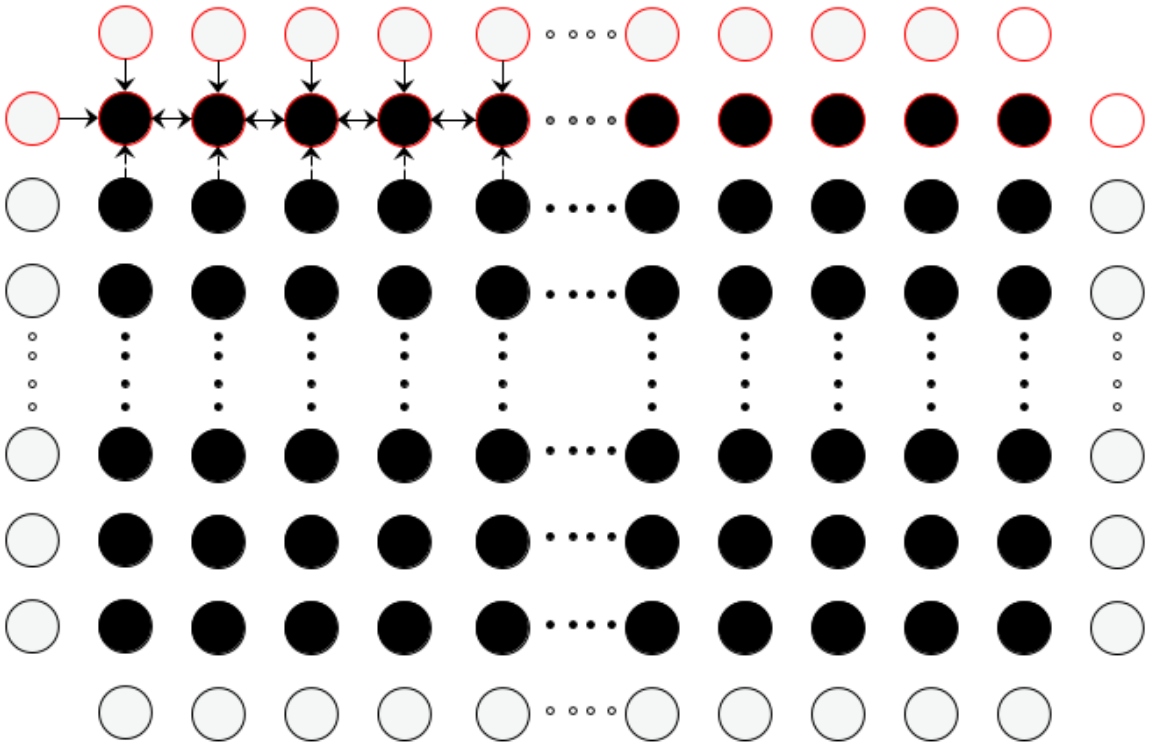
Five-point star-shaped stencil

$$\begin{aligned} T_{new_{i,j}} = & w_1 * T_{prev_{i,j+1}} + w_2 * T_{prev_{i,j}} \\ & + w_3 * T_{prev_{i,j-1}} + w_4 * T_{prev_{i+1,j}} \\ & + w_5 * T_{prev_{i-1,j}} \end{aligned}$$

Implementation

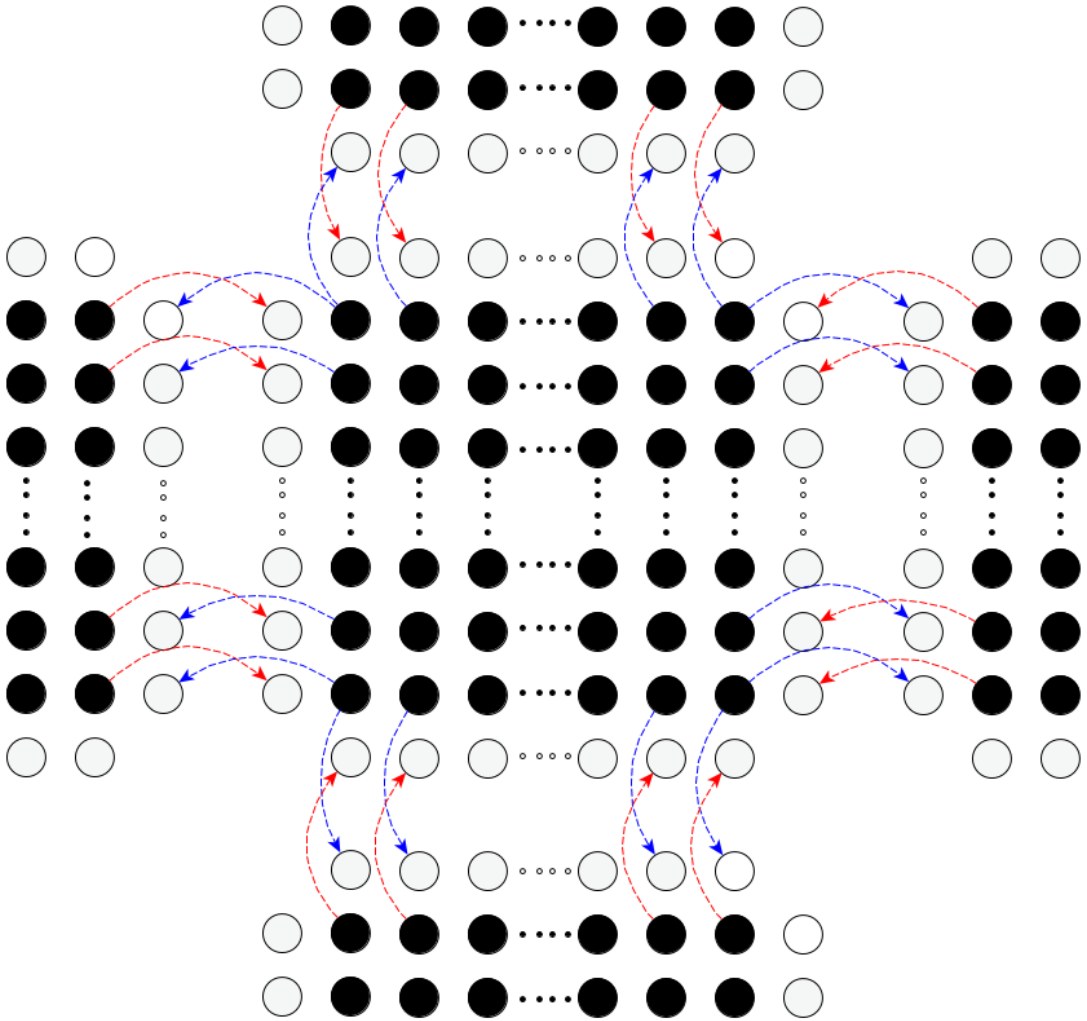
- Hand-tuned unrolled assembly code
- “In-place” implementation
- Size of grid limited by local memory (and size of assembly code)
- All 64 registers used and managed carefully
- Grid initialized in the host and transferred to each eCore
- Computation followed by communication phase in each iteration

Computation Phase



- Grid sizes of 20 $\times$ X. Width of 20 decided based on register availability
- Buffer 2 rows of grid points into registers and perform FMADDs
- Continuous runs of Fused Multiply-Add (FMADD) interleaved with 64-bit load/store
- 5 grid points accumulated at a time
- Each grid point loaded into register only once

Communication Phase

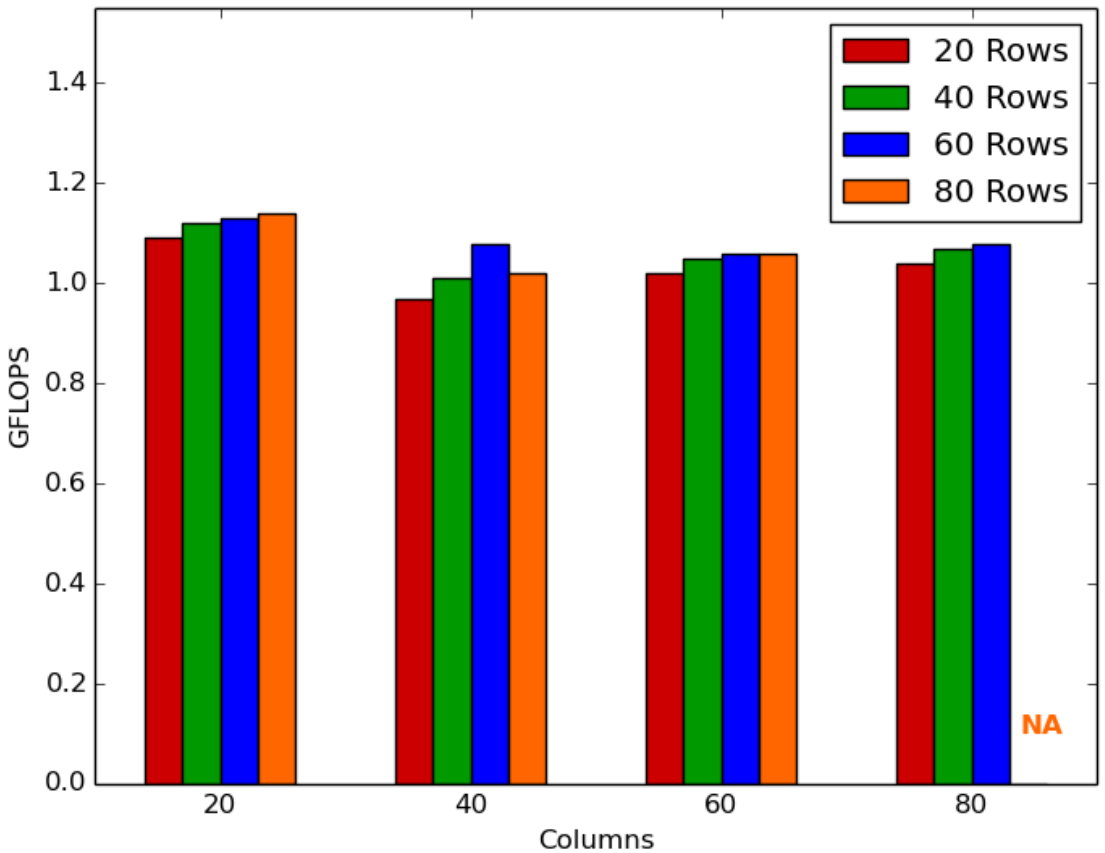


- Synchronization between neighbouring eCores
- Transfers started after neighbour's computation phase
- DMA for boundary transfers

Outline

Floating point performance

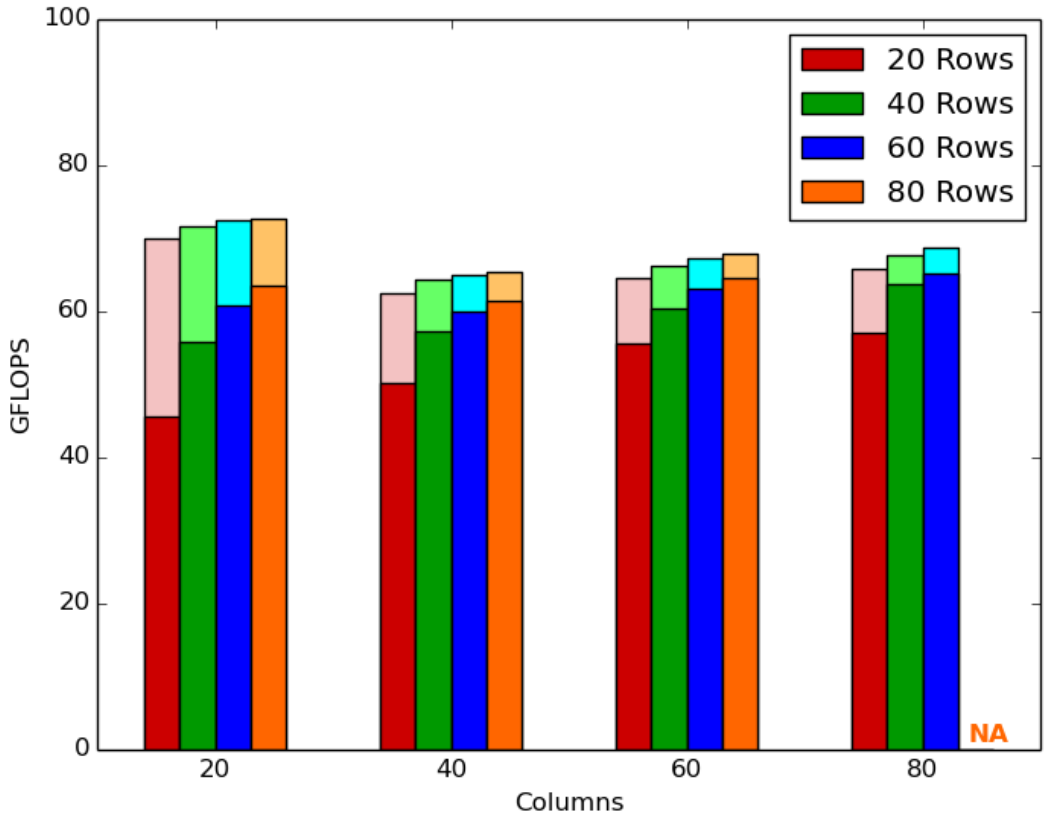
Single-core Floating point performance in GFLOPS



- Stencil evaluated for 50 iterations.
- 81-95% of peak performance

Floating point performance

64-core Floating point performance in GFLOPS

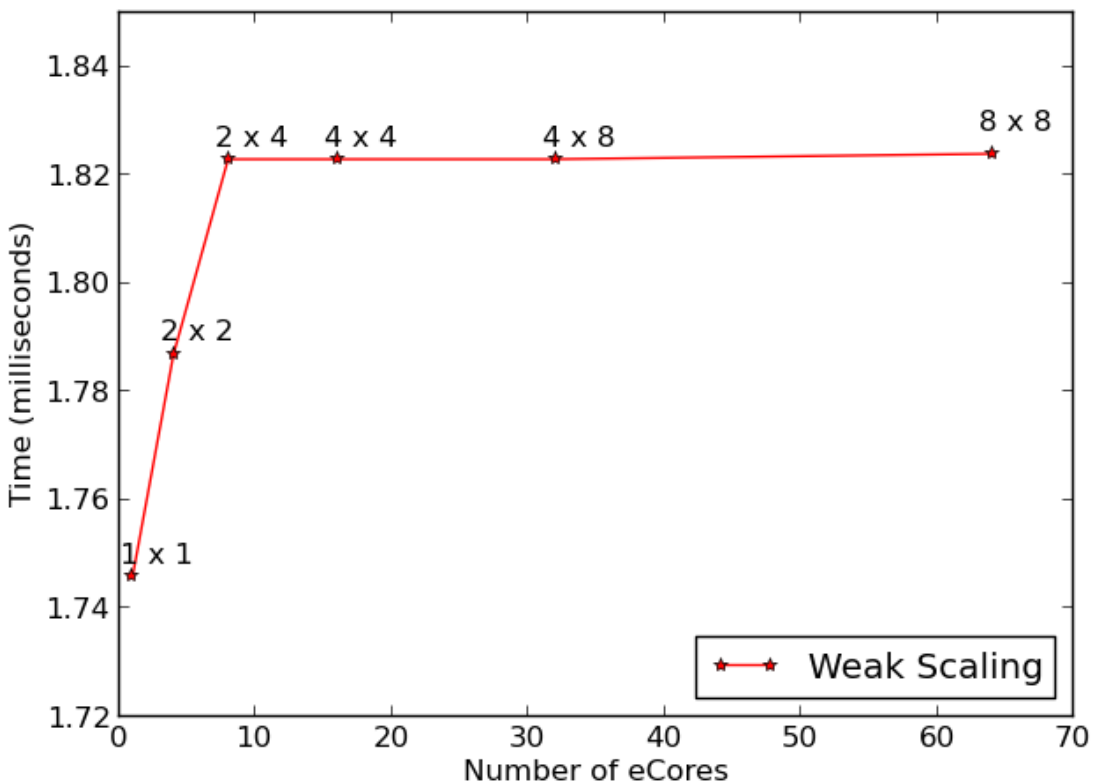


- 83% of peak performance with communication

*Lighter colors show performance without communication

Weak Scaling

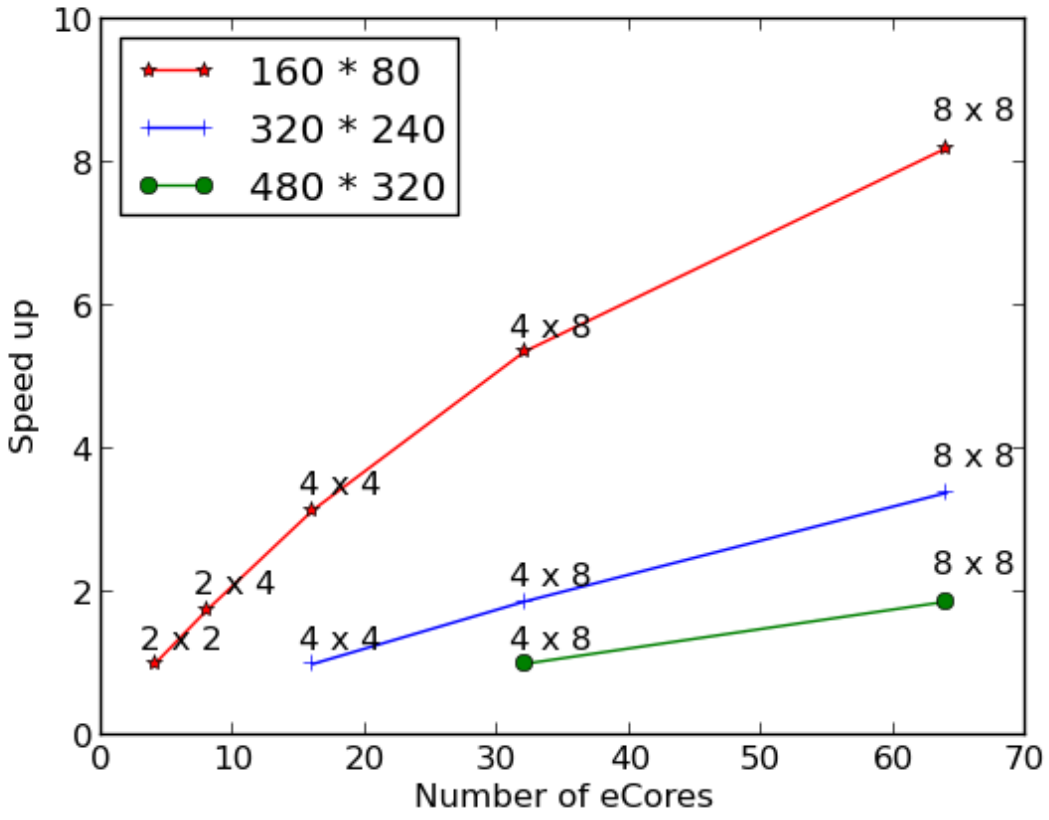
Weak Scaling - Number of eCores vs Time



- Number of eCores from 1 to 64
- Vary problem size from 60 $\times$ 60 to 480 $\times$ 480

Strong Scaling

Strong Scaling - Number of eCores vs Speedup



- Number of eCores from 1 to 64
- Problem size fixed

Conclusions and Future Work

- Heat Stencil running at 65 GFLOPS (83%)
  - $\approx$ 32 GFLOPS/Watt assuming 2W power consumption
  - Double-buffering of boundary regions to overlap computation and communication
- Epiphany platform holds high potential for HPC
  - Considerable effort to extract high performance
- Memory constraint important factor while designing algorithms
  - Streaming algorithm to process higher grid sizes
  - Future version of Epiphany to have 4096 cores (70 GFLOPS/Watt)

Questions?