U.S. ARMY RDECOM
ARL RESEARCH LABORATORY
Brown Deer Technology

U.S. ARMY RESEARCH, DEVELOPMENT AND ENGINEERING COMMAND

Scaling OpenSHMEM for Massively Parallel Processor Arrays

James A. Ross (Army Research Lab) and David A. Richie (Brown Deer Technology)

James A. Ross

Computational Scientist

Army Research Laboratory

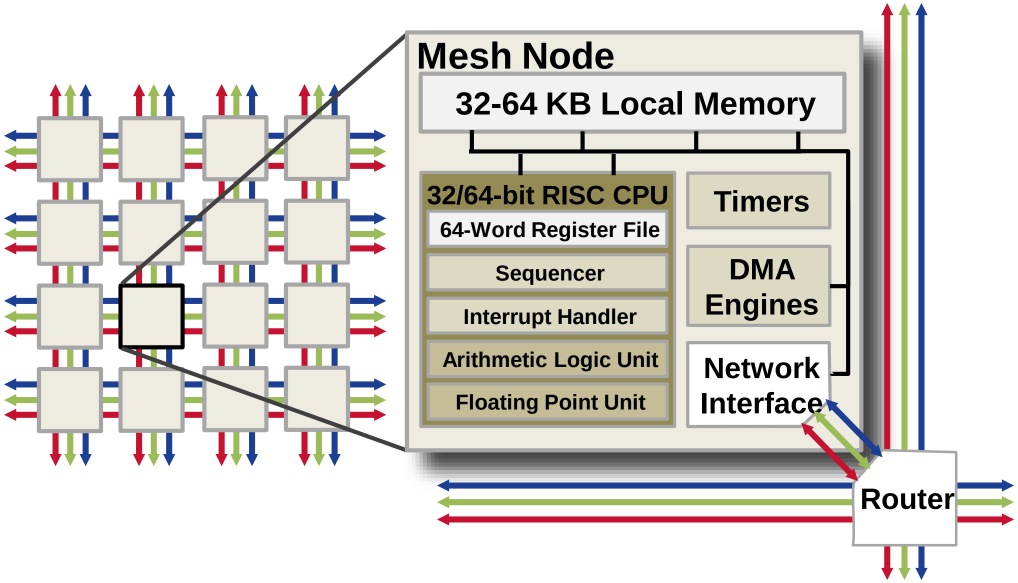
21 AUG 2018

OUTLINE

- Epiphany architecture
- Programming models
- Motivation for emulation and simulation
- Epiphany-based SoC emulation
- Simulation of multiple devices
- Simulated OpenSHMEM results
- Conclusions and future work

EPIPHANY RISC ARCHITECTURE

- Design emphasizes simplicity, scalability, power-efficiency
- 2D array of RISC cores, 2D Network on Chip (NoC)
- 32-64KB shared global scratch memory per mesh node
- Fully divergent cores
- Minimal un-core functionality, e.g., no data or instruction cache
- Design scales to thousands of cores
- High performance/power efficiency

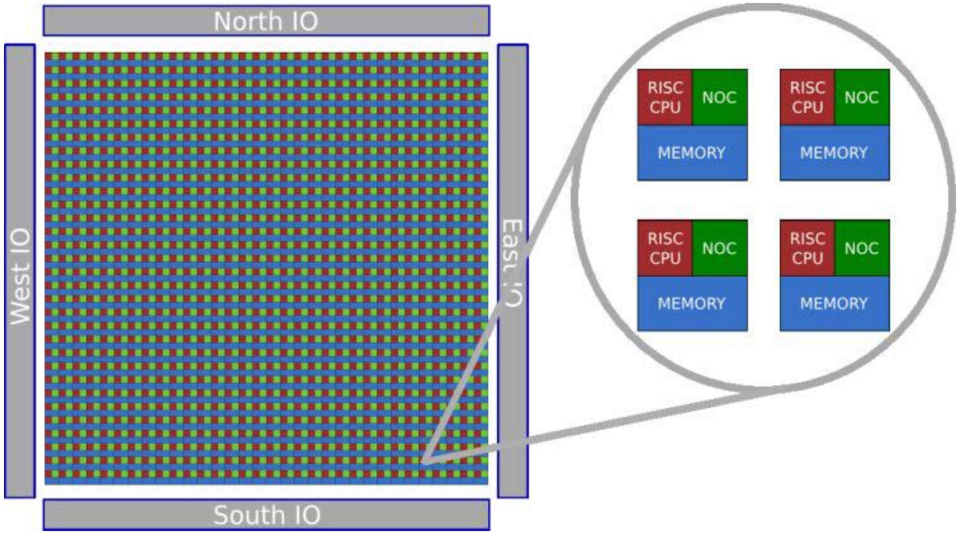


EPIPHANY RISC ARCHITECTURE

- Scalability of architecture has been demonstrated in silicon:

Device	Cores	Node	Address & FPU	Power Efficiency
Epiphany-III	16	65nm	32-bit	50 GFLOPS/W
Epiphany-IV	64	28nm	32-bit	70 GFLOPS/W
Epiphany-V	1024	16nm	64-bit	TBD

- Epiphany-V fabricated by TSMC
- Numerous firsts demonstrated ...
- Largest number of general-purpose processor cores per chip
- Highest density HPC chip
- Most efficient chip design team
- Increases motivation to advance the programming model for this architecture

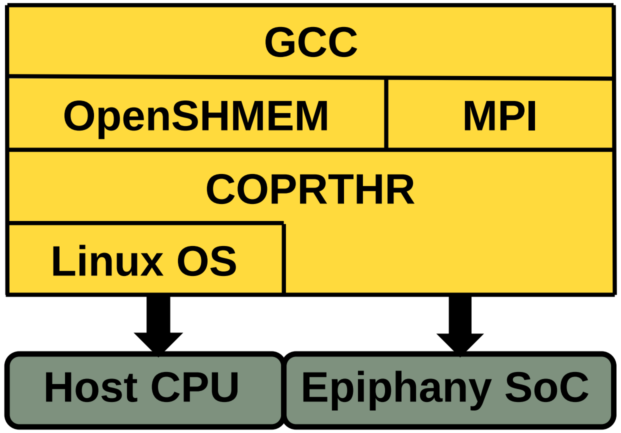


PROGRAMMING CHALLENGE

- Power-efficiency achieved by simplicity in architecture
- Software supports functionality typically done in hardware
- Distributed memory-mapped cores with limited memory per core
  - Epiphany-III has 32 KB local SRAM per core
  - Epiphany-V increased to 64 KB, same challenge
  - Local memory used for instructions and data
- Non-uniform memory access (NUMA) to mapped local memory
- No hardware data/instruction cache
- Best viewed as a “distributed cluster on chip”
- Device employed as co-processor – requires offload semantics
- Resource constraints prevent running full process image

EPIPHANY SOFTWARE SUPPORT

- Programming support for Epiphany has developed over time:
  - Low-level support: eSDK[1], COPRTHR[3], PAL[1]
  - Programming methods: GCC[2], OpenSHMEM[3], OpenCL[4], OpenMP[5], MPI[3,4], Erlang[6], BSP[7], Epython[8]
  - Programming software/models used in this work



- Code is compiled with a GCC front-end (coprcc)
- Epiphany parallel programming with OpenSHMEM or MPI
- Host CPU offload support with COPRTHR
- Linux runs on the HOST CPU
- Epiphany proto-OS support with COPRTHR
- Together provides a complete programming solution

[1]Adapteva, [2]Embecosm, [3]US Army Research Laboratory, [4]BDT, [5]University of Ioannina, [6]Uppsala University, [7]Coduin, [8]Nick Brown

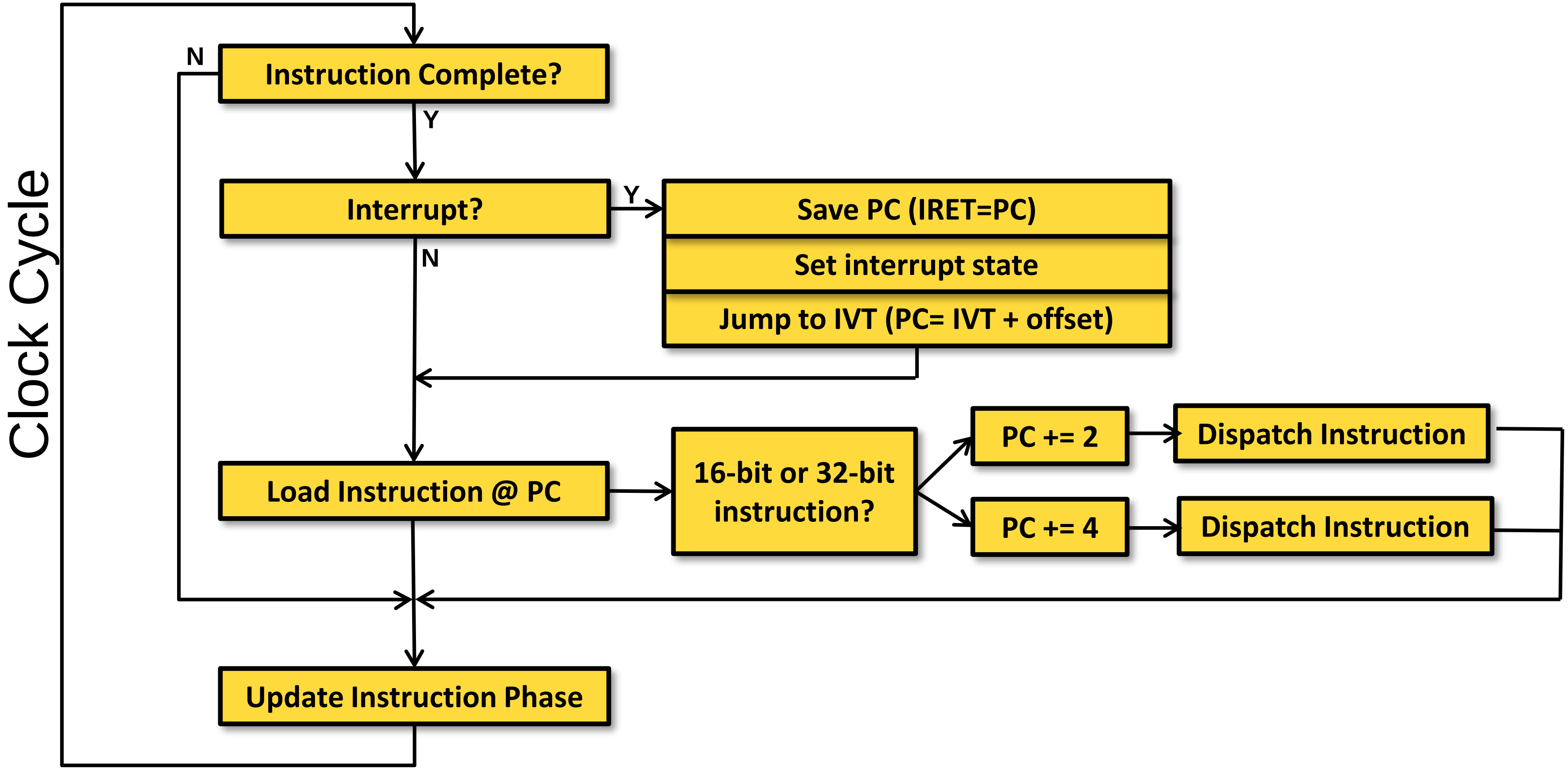
EMULATION AND SIMULATION

- Objective: develop an emulation and simulation capability for Epiphany-based architectures
- Integrate with real software development workflow
  - Seamless development and run-time integration
- Study architecture changes directly with real software
- Study multi-device integration/scaling/performance
- Enable software development before silicon is available
- Support hardware/software co-design for hybrid SoCs
- Desired accuracy: functional correctness, timing metrics representative of real hardware execution
  - Does not reproduce cycle-accurate state across large system
  - EDA tools can be used for studying design details, but do not scale

U.S. ARMY RDECOM
ARL RESEARCH LABORATORY
Brown Deer Technology

EPIPHANY EMULATOR

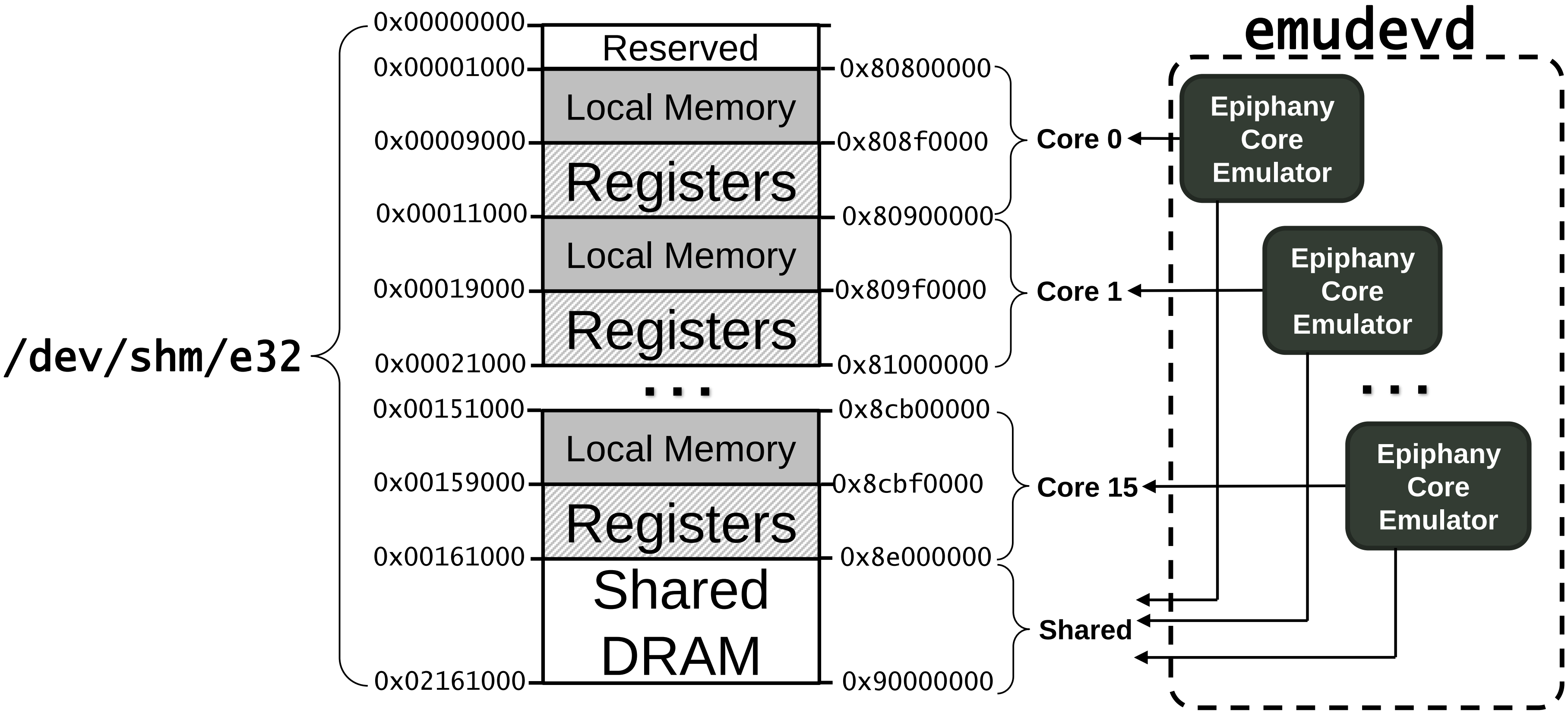
High-level design of Epiphany ISA emulator



VIRTUAL EMULATED DEVICE

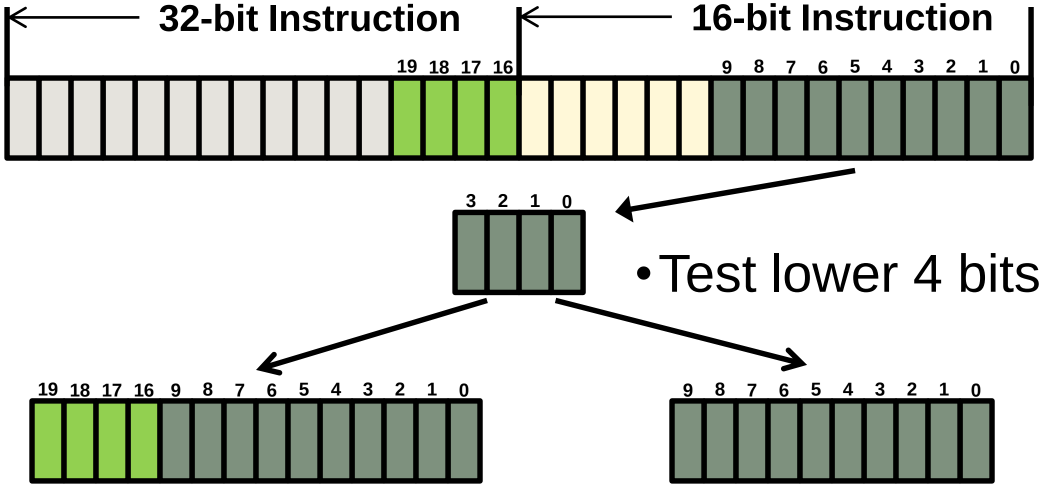
- Replicate and extend Parallella platform model
- Physical Epiphany SoC is interfaced using a driver and Linux device special file mounted at `/dev/epiphany/mesh0`
- COPRTHR API performs reads and writes to multiple memory mapped regions of this device special file
  - global memory, per-core local memory and register file
- We integrate (one or more) emulated Epiphany devices using POSIX shared memory regions under `/dev/shm`
- The emulator operates on the shared memory regions asynchronously with the host APi interactions via COPRTHR
- Integration with user software applications is seamless
- Design avoids stand-alone tool or framework
- Enables a compilation and execution environment identical to a platform with a physical Epiphany SoC

VIRTUAL EMULATED DEVICE



EPIPHANY EMULATOR

- Indirect threaded dispatch of 16-bit and 32-bit instructions
- Instruction decoding uses efficiencies in the ISA design



- Test lower 4 bits

  - Form 10(14)-bit call vector offset for 16(32)-bit instruction
  - Dispatch instruction through pre-initialized call vector table

- Additional functionality of emulator:
  - Special registers control various functional behaviors
  - Dual DMA engines operate independently
  - Memory interface is abstracted for experimentation

VIRTUAL EMULATED DEVICE

- Compilation/execution identical for physical and emulated devices
- Output below shows the compilation and execution of an Epiphany Parallella benchmark on an ordinary x86 workstation
- The Cannon benchmark code was taken directly from a Parallella platform, compiled, and executed without modification

```
$ gcc -I. -I$COPRTHR_INC_PATH -c cannon_host.c
$ gcc -rdynamic -o cannon.x cannon_host \
       -L$COPRTHR_LIB_PATH -lcoprthr -lcoprthrcc -lm -ldl
$ coprcc -DCOPRTHR_MPI_COMPAT -o cannon_tfunc.e32 cannon_tfunc.c \
       -L$COPRTHR_LIB_PATH -lcoprthr_mpi
$ ./cannon.x -d 4 -n 32

COPRTHR-2 (Anthem) build 20180118.0014
main: Using -n=32, -s=1, -s2=1, -d=4
main: dd=0
main: 0x2248420 0x223f3f0
main: mpiexec time 0.117030 sec
main: # errors: 0
```

SIMULATING NEW ARCHITECTURES

- Emulator is able to configure Epiphany devices with no physical equivalent
  - Number of cores, size of local memory, etc.
  - New instructions can be added
  - New un-core functionality can be added
- Example configuration of a 256-core Epiphany-III device is shown below

```
### device-e32-256.conf
### Device specification conf file
### This is an emulated device based on Epiphany-III, 32-bit, 256 cores

devices = (
        {
                name="e32.0"; device_type="e32_emu";
                nrows=16; ncols=16; row_offset=32; col_offset=8;
                local_mem=0x8000; ext_mem=0x2000000;
        }
);
```

SIMULATING NETWORK DELAY

- Instruction dispatch design allows instructions to stall to support network latency
- Memory and network interfaces separate abstractions
- Network delay for transactions modeled as:

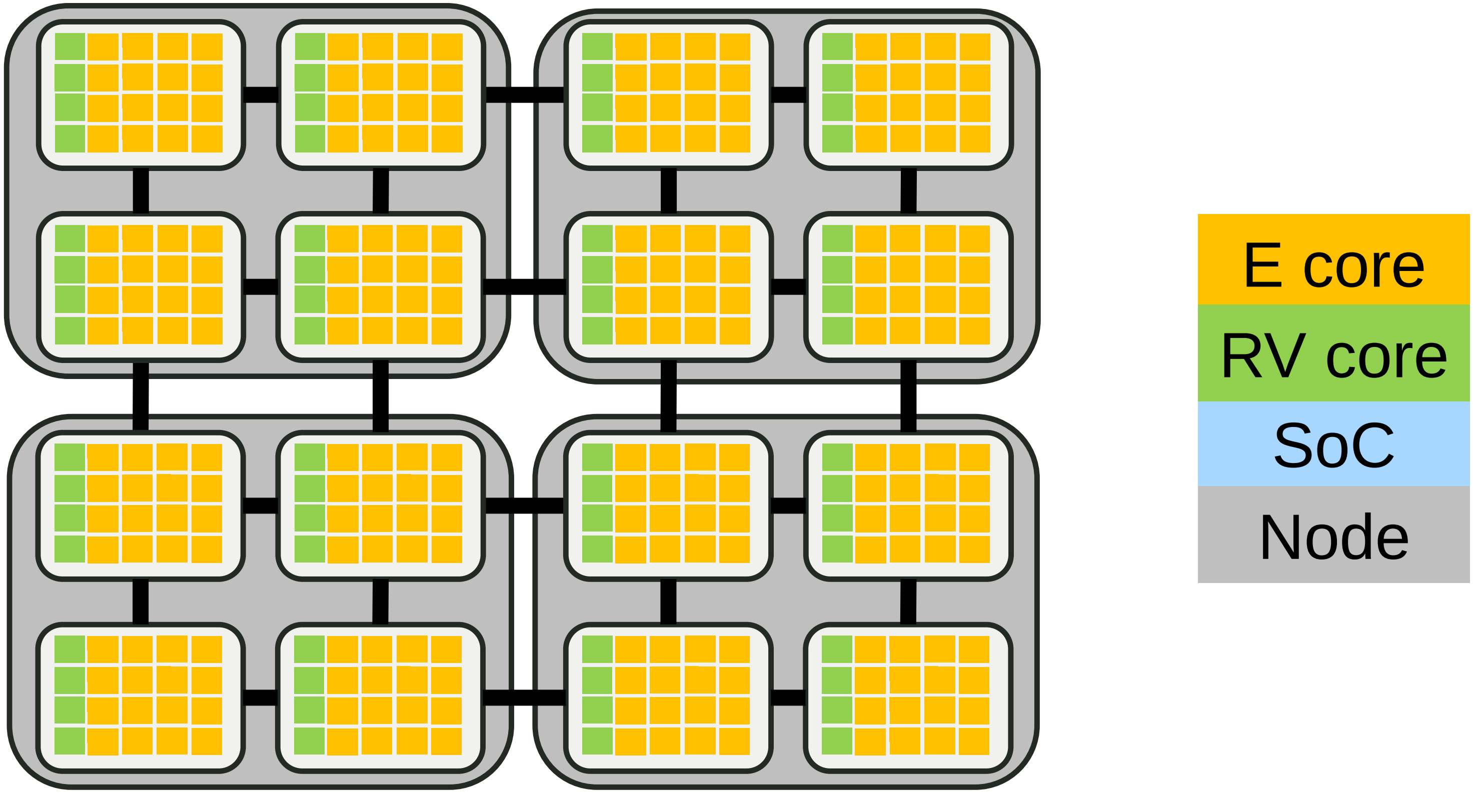
$$\tau = 1.5 \times (|r - r_0| + |c - c_0|)$$

where r (c) and r_0 (c_0) are local and remote row (column)

- Network congestion not presently modeled

SIMULATING NEW PLATFORMS

- Test topology with 4 nodes simulated on x86 platform:
  - 4 SoCs per node, 4 RISC-V + 16 Epiphany cores per SoC
- Network traffic modeled with NS-3
- Cores executed cross-compiled binaries



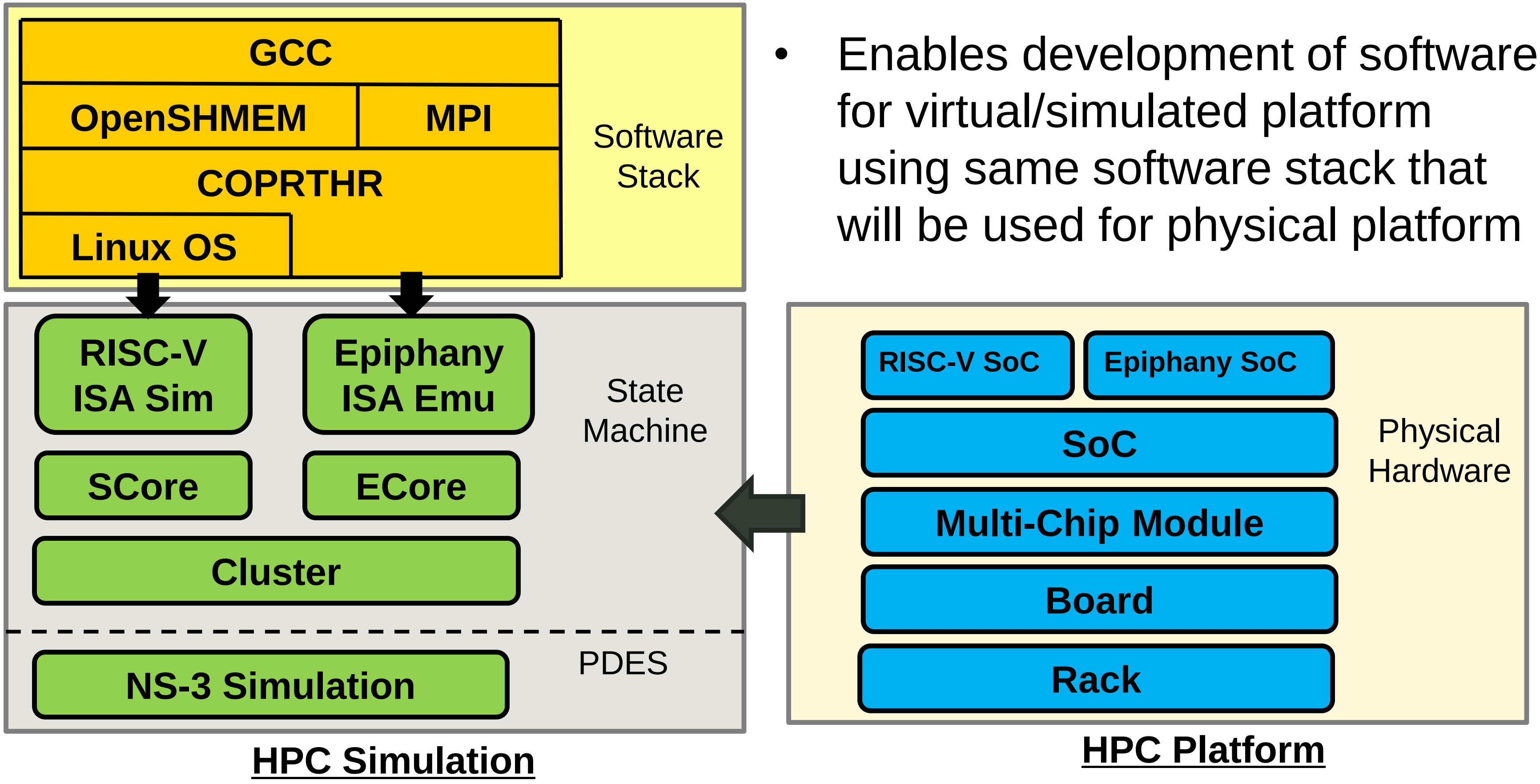
E core

RV core

SoC

Node

SIMULATION OF NEW PLATFORMS

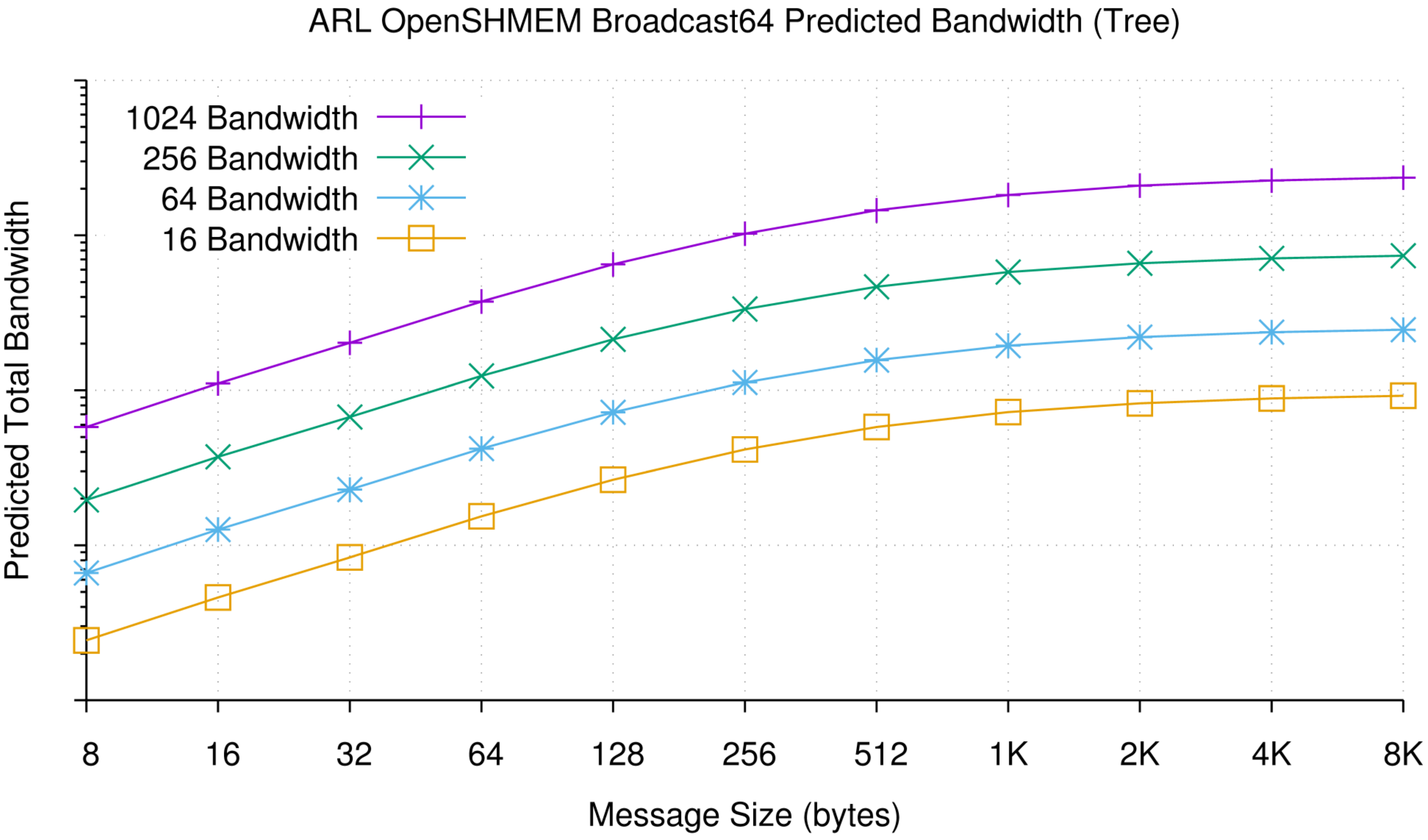


- Enables development of software for virtual/simulated platform using same software stack that will be used for physical platform

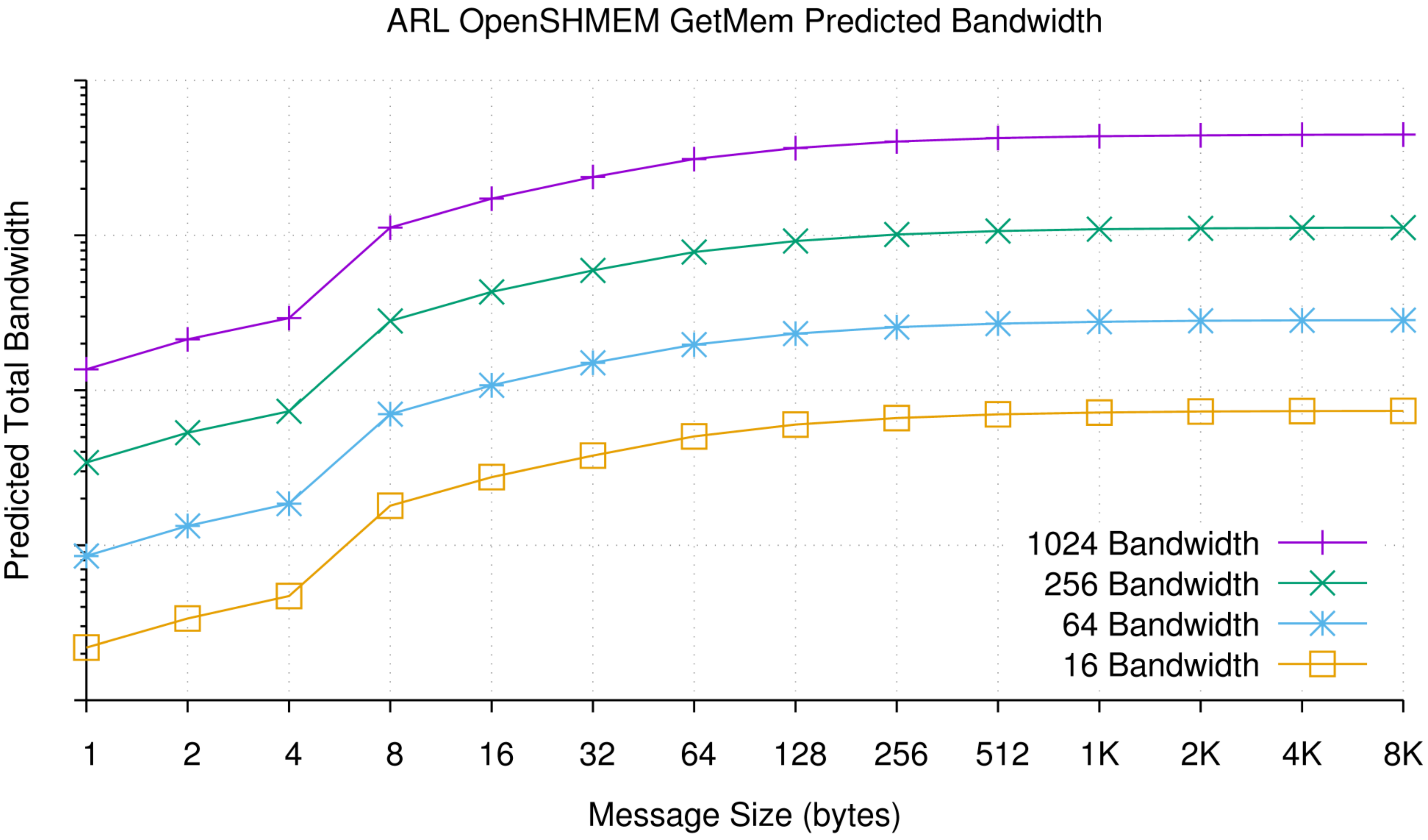
SIMULATING NEW PLATFORMS

- Target is a system comprised of hybrid RISC-V/Epiphany SoCs
  - RISC-V supervisor cores replace host CPUs, e.g., on Parallella
  - Large array of Epiphany cores perform computations
  - Architecture is scalable
- We want to use larger networked systems to develop software and study the behavior and performance of the overall system
- We will use the network simulator NS-3 for modeling network traffic between multiple nodes containing hybrid devices
- RISC-V simulator is available with the RISC-V toolchain
- Epiphany cores emulated with the newly developed emulator
- Represents ongoing work, but first demonstrations have been performed successfully

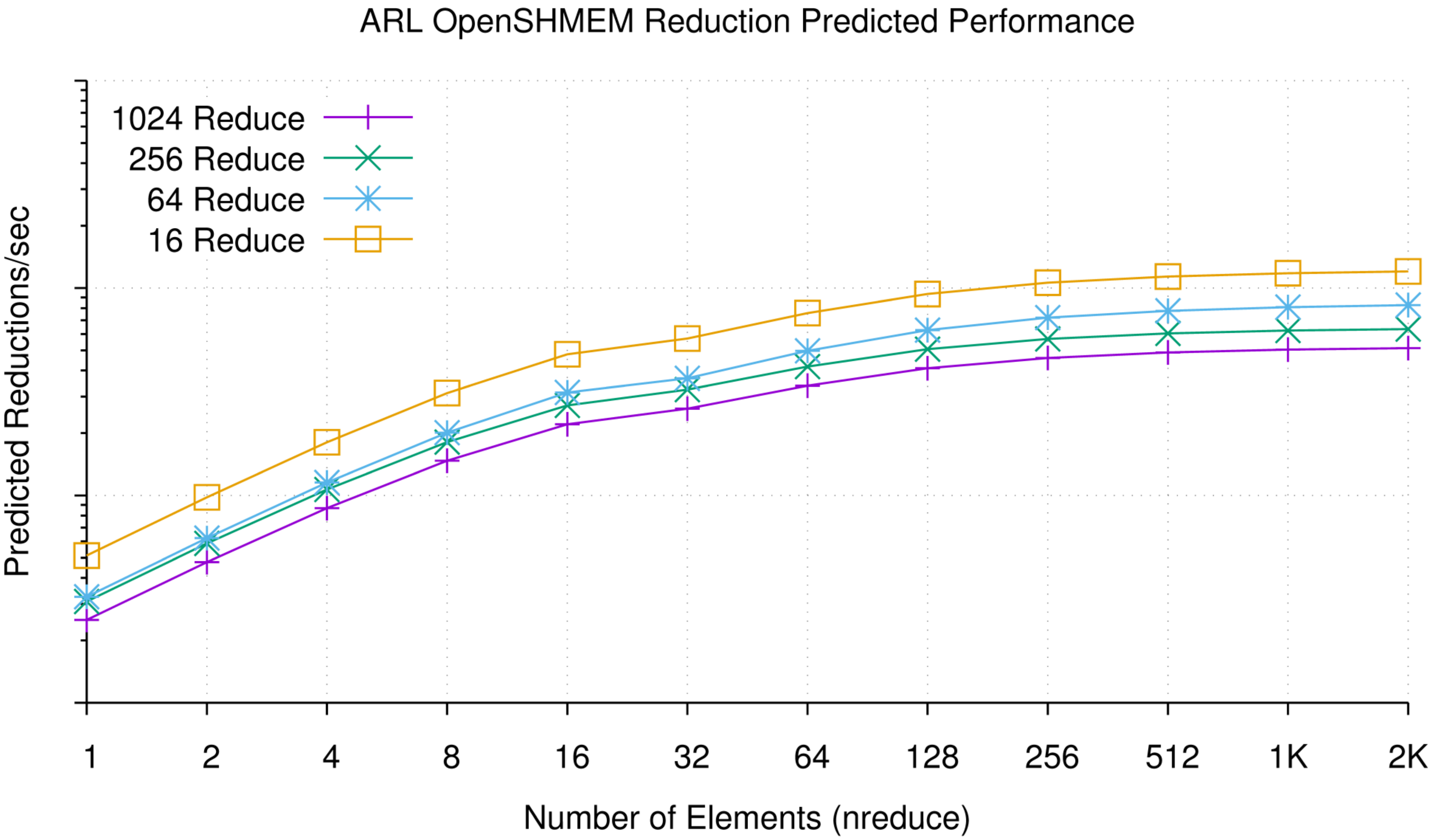
SIMULATED OPENSHMEM RESULTS



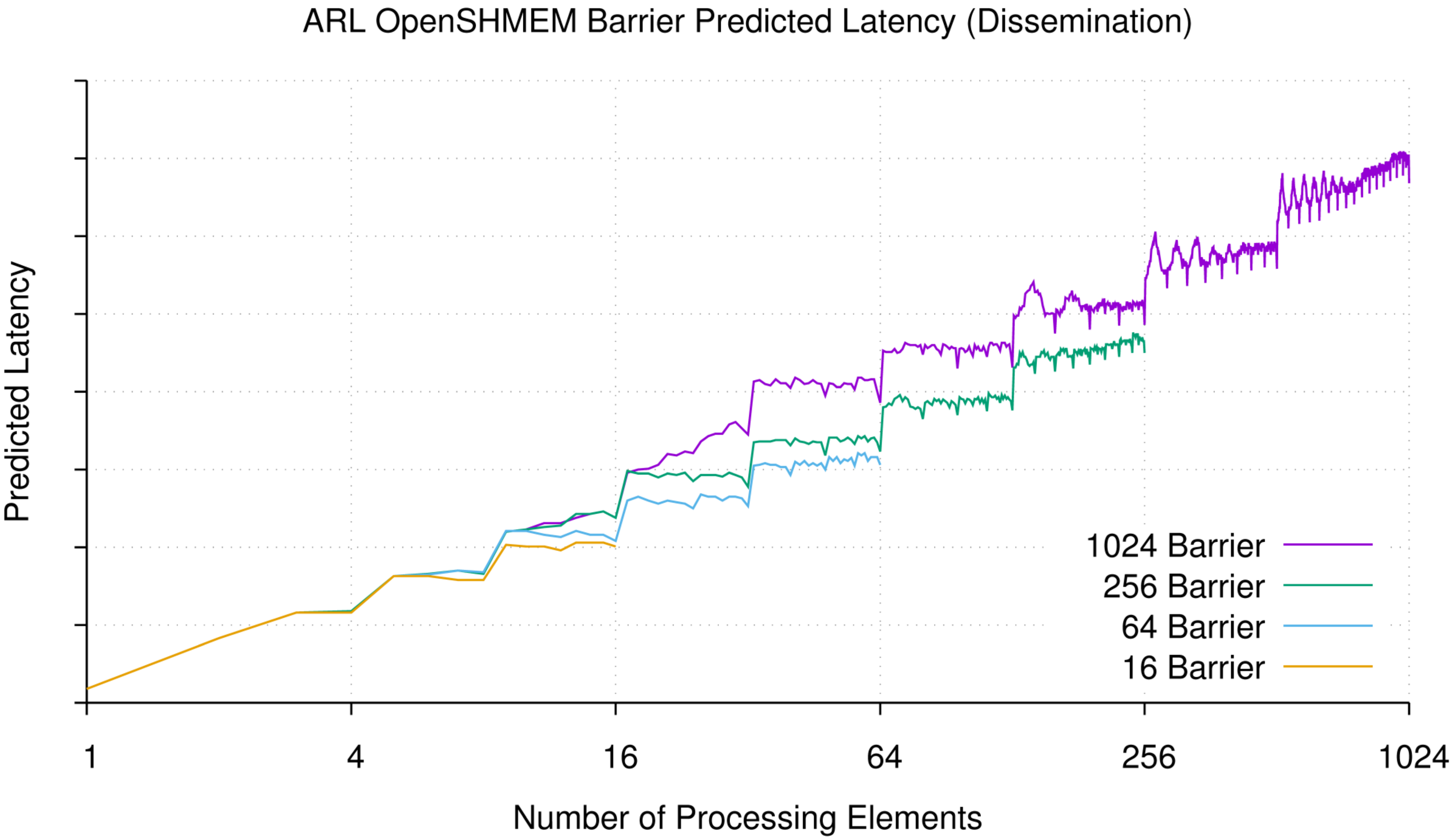
SIMULATED OPENSHMEM RESULTS



SIMULATED OPENSHMEM RESULTS



SIMULATED OPENSHMEM RESULTS



CONCLUSIONS AND FUTURE WORK

- We have implemented an Epiphany ISA emulator
  - 32-bit ISA support (for now)
  - Configurable as virtual many-core device for testing and software development on an ordinary x86 platform
- Design enables seamless interface, software development and execution is identical to that of a platform with physical Epiphany device
- Emulation and simulation will allow the study of future architectures based on the Epiphany architecture
- Ongoing work includes the development of a simulation framework for large multi-device platforms
  - NS-3 is used for network simulating traffic
  - RISC-V+Epiphany simulator/emulator for modeling hybrid SoC
- Enables the development and testing of software for future architectures being investigated