Visualization of OpenCL Application Execution on CPU-GPU Systems

A. Ziabari*, R. Ubal*, D. Schaa**, D. Kaeli*

*NUCAR Group, Northeastern Universiy
**AMD

Northeastern University
Computer Architecture Research Group

Introduction and Motivation

- Simulators
  - Design evaluation (Pre- and Post- silicon)
  - Design validation
  - Education
  - … and much more
- Visualization
  - Complimentary to the simulator
  - Easy to interact
  - Thorough study of the simulated data
- Motivation
  - Teaching details of OpenCL application execution

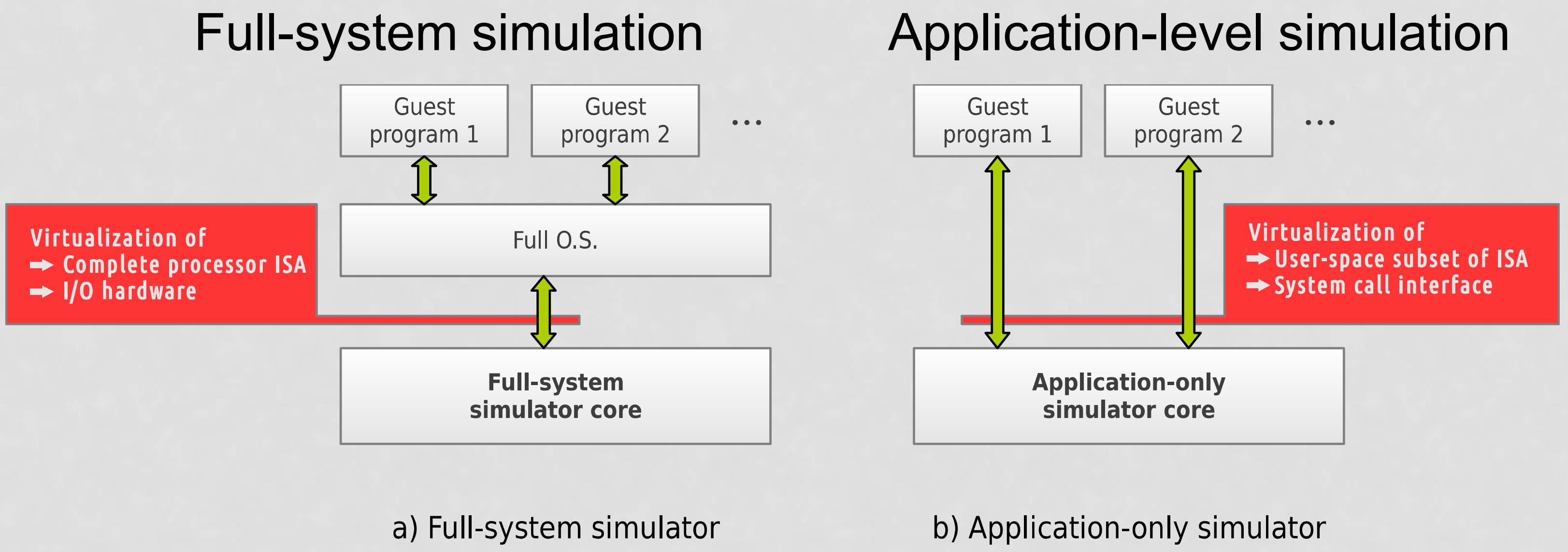
Outline

- Background and simulation methodology
- OpenCL application on the host
- OpenCL application on the GPU device
- Education through visualization
- Ongoing Work

Background and Simulation Methodology

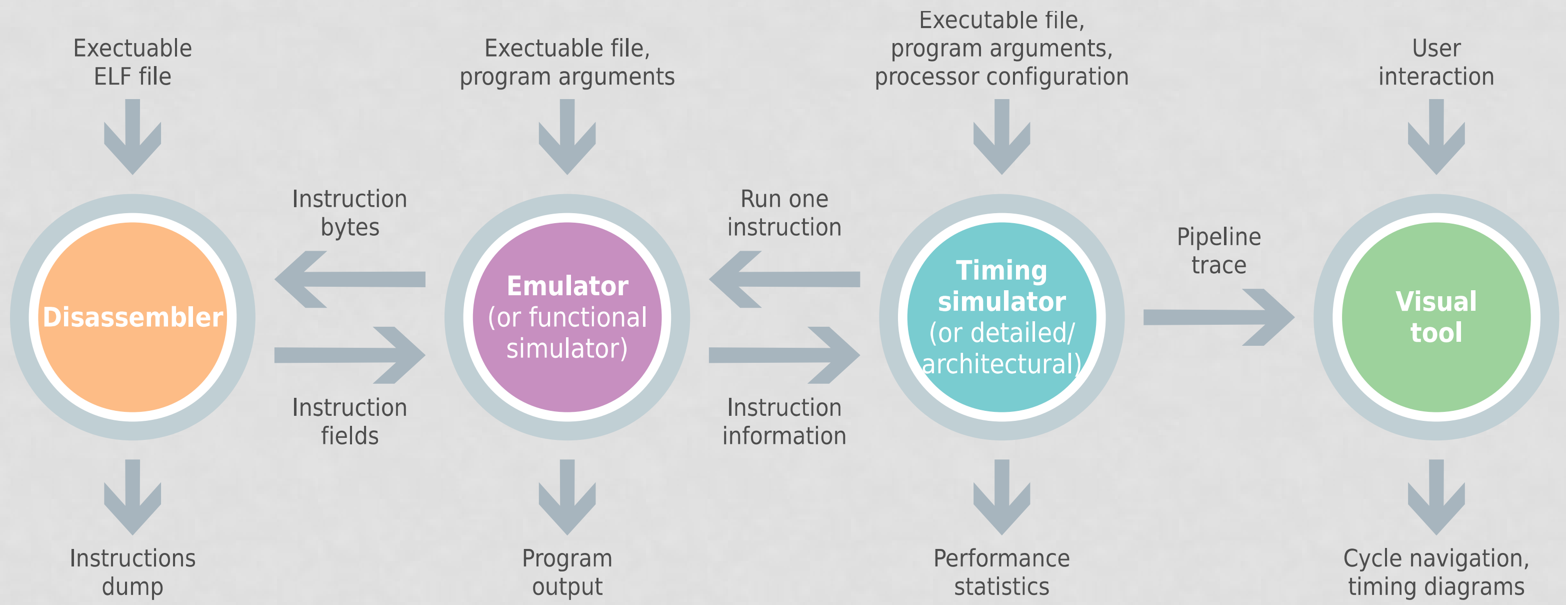
The Multi2Sim Simulation Framework

- Simulation framework for CPU, GPU, and heterogeneous systems
- Support for CPU architectures: x86, ARM, MIPS
- Support for GPU architectures: AMD Evergreen, AMD Southern Islands, NVIDIA Kepler, HSA intermediate language
- Application-level simulation



Background and Simulation Methodology

Four-Stage Simulation Process


Exectuable ELF file
Exectuable file, program arguments
Executable file, program arguments, processor configuration
User interaction
Disassembler
Instruction bytes
Instruction fields
Emulator (or functional simulator)
Run one instruction
Instruction information
Timing simulator (or detailed/ architectural)
Pipeline trace
Visual tool
Instructions dump
Program output
Performance statistics
Cycle navigation, timing diagrams

- Four isolated software modules for each architecture (x86, SI, ARM, ...)
- Each module has a command-line interface for stand-alone execution, or an API for interaction with other modules.

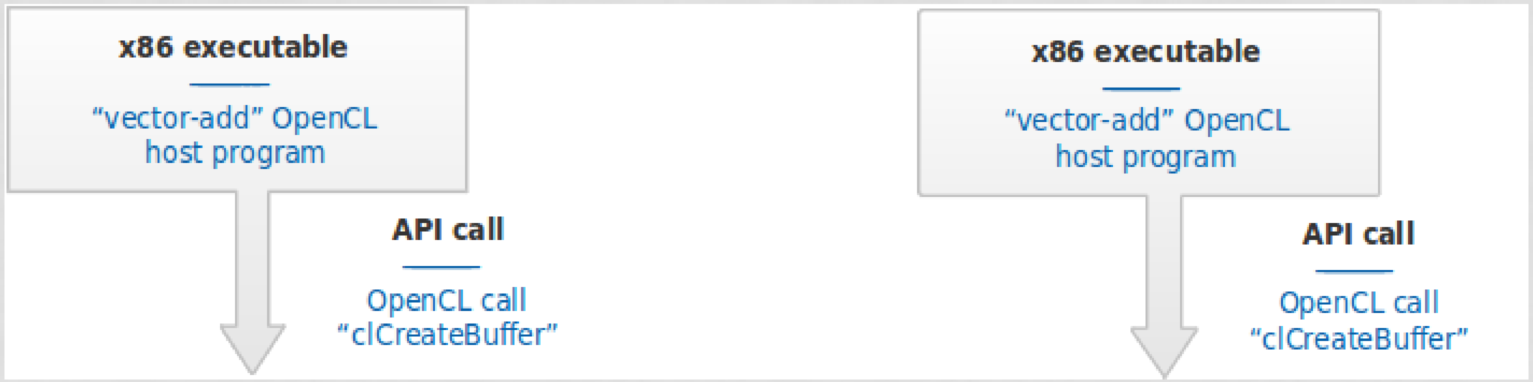
Outline

- Background and simulation methodology
- OpenCL application on the host
- OpenCL application on the GPU device
- Education through visualization
- Ongoing Work

OpenCL on the Host

The OpenCL CPU Host Program

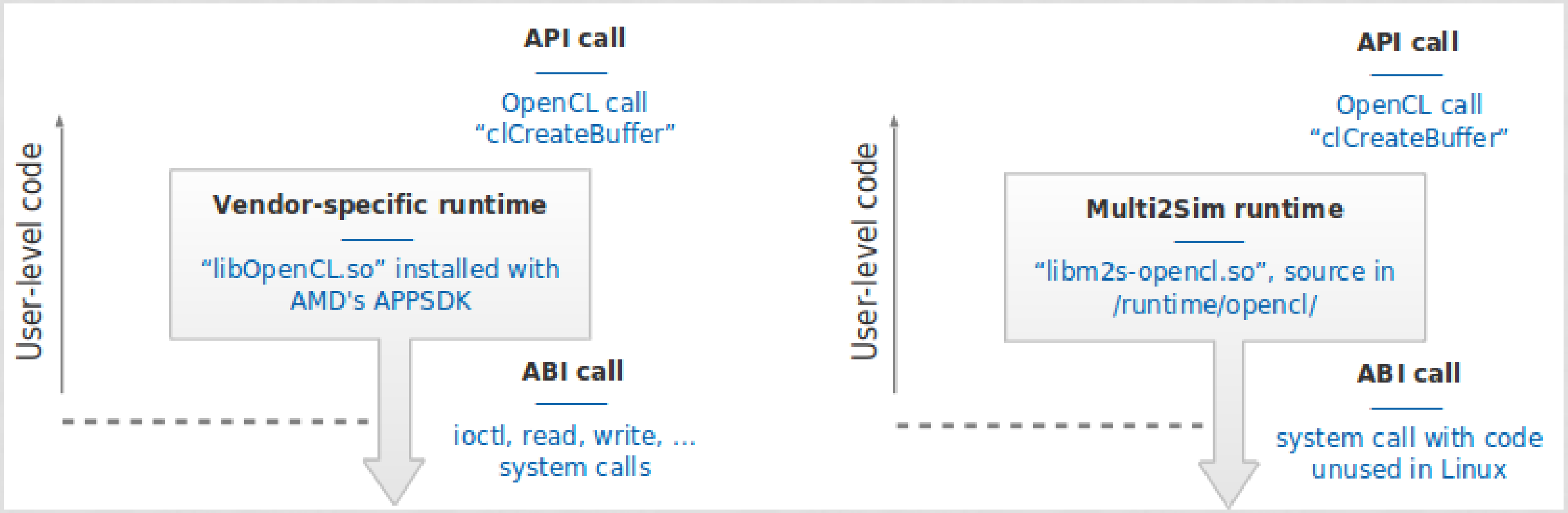
- Native
  - An x86 OpenCL host program performs an OpenCL API call.
- Multi2Sim
  - Same


x86 executable
"vector-add" OpenCL host program
API call
OpenCL call "clCreateBuffer"
x86 executable
"vector-add" OpenCL host program
API call
OpenCL call "clCreateBuffer"

OpenCL on the Host

The OpenCL Runtime Library

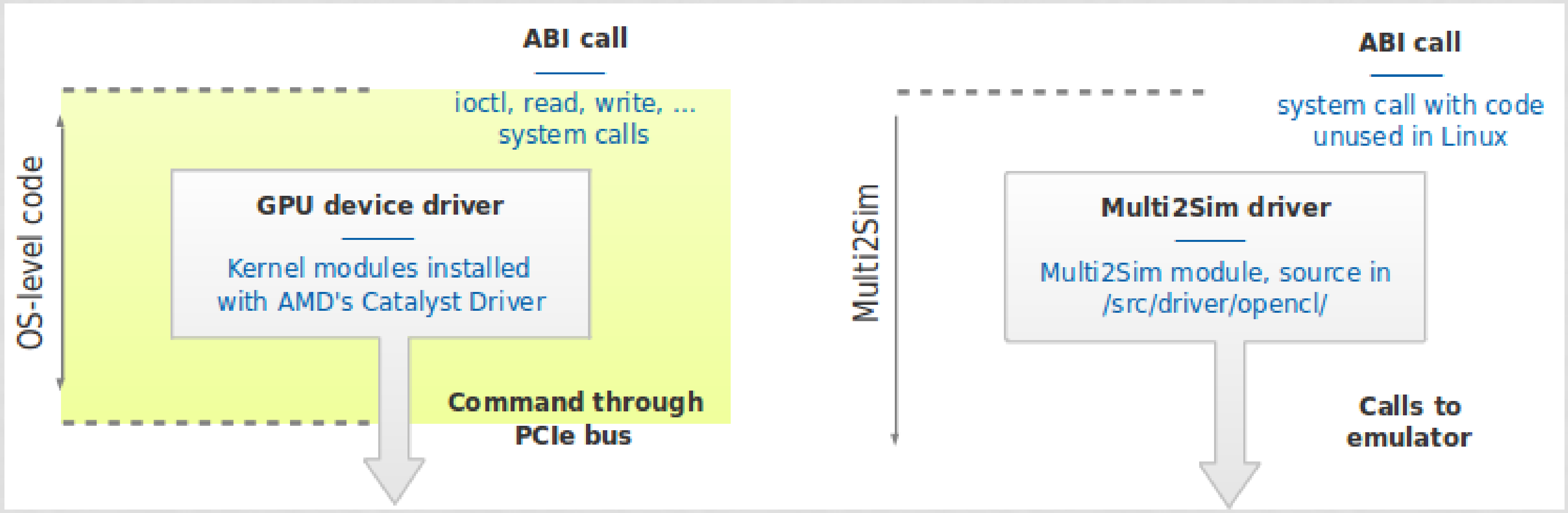
- Native
  - AMD's OpenCL runtime library handles the call, and communicates with the driver through system calls *ioctl*, *read*, *write*, etc. These are referred to as ABI calls.
- Multi2Sim
  - Multi2Sim's OpenCL runtime library, running with guest code, transparently intercepts the call. It communicates with the Multi2Sim driver using system calls with codes not reserved in Linux.


API call
OpenCL call "clCreateBuffer"
User-level code
Vendor-specific runtime
"libOpenCL.so" installed with AMD's APPSDK
ABI call
ioctl, read, write, ... system calls
API call
OpenCL call "clCreateBuffer"
User-level code
Multi2Sim runtime
"libm2s-opencl.so", source in /runtime/opencl/
ABI call
system call with code unused in Linux

OpenCL on the Host

The OpenCL Device Driver

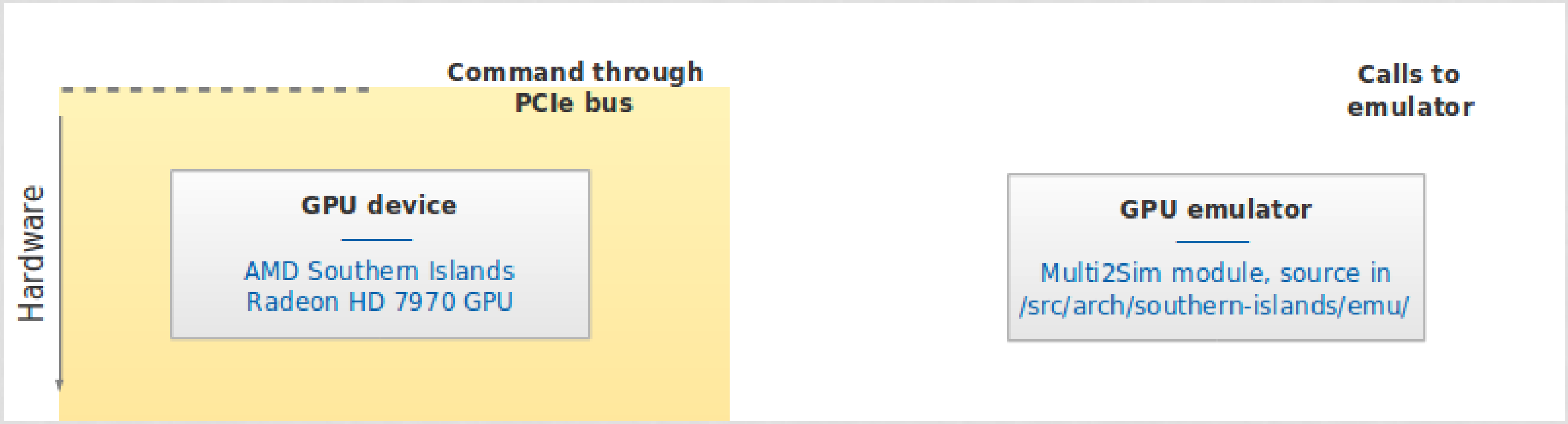
- Native
  - The AMD Catalyst driver (kernel module) handles the ABI call and communicates with the GPU through the PCIe bus
- Multi2Sim
  - An OpenCL driver module (Multi2Sim code) intercepts the ABI call and communicates with the GPU emulator


ABI call
ioctl, read, write, ... system calls
OS-level code
GPU device driver
Kernel modules installed with AMD's Catalyst Driver
Command through PCIe bus
ABI call
system call with code unused in Linux
Multi2Sim
Multi2Sim driver
Multi2Sim module, source in /src/driver/opencl/
Calls to emulator

OpenCL on the Host

The GPU Emulator

- Native
  - The command processor in the GPU handles the messages received from the driver
- Multi2Sim
  - The GPU emulator updates its internal state based on the message received from the driver


Command through PCIe bus
Hardware
GPU device
AMD Southern Islands Radeon HD 7970 GPU
Calls to emulator
GPU emulator
Multi2Sim module, source in /src/arch/southern-islands/emu/

OpenCL on the Host

Transferring Control

- The **host program** performs API call *clEnqueueNDRangeKernel*
- The **runtime** intercepts the call, and enqueues a new task in an OpenCL command queue object. A user-level thread associated with the command queue eventually processes the command, performing a *LaunchKernel* ABI call
- The **driver** intercepts the ABI call, reads ND-Range parameters, and launches the GPU emulator
- The **GPU emulator** enters a simulation loop until the ND-Range completes

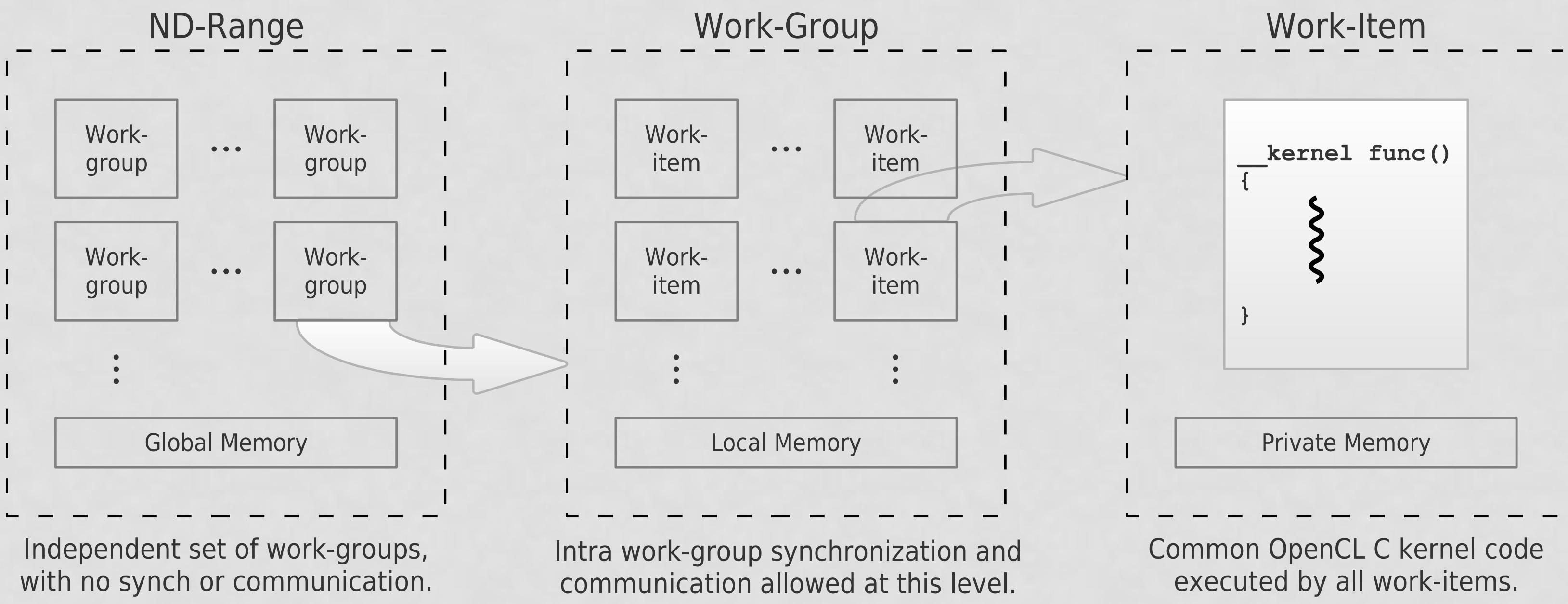
Outline

- Background and simulation methodology
- OpenCL application on the host
- OpenCL application on the GPU device
- Education through visualization
- Ongoing Work

OpenCL on the Device

Execution Model

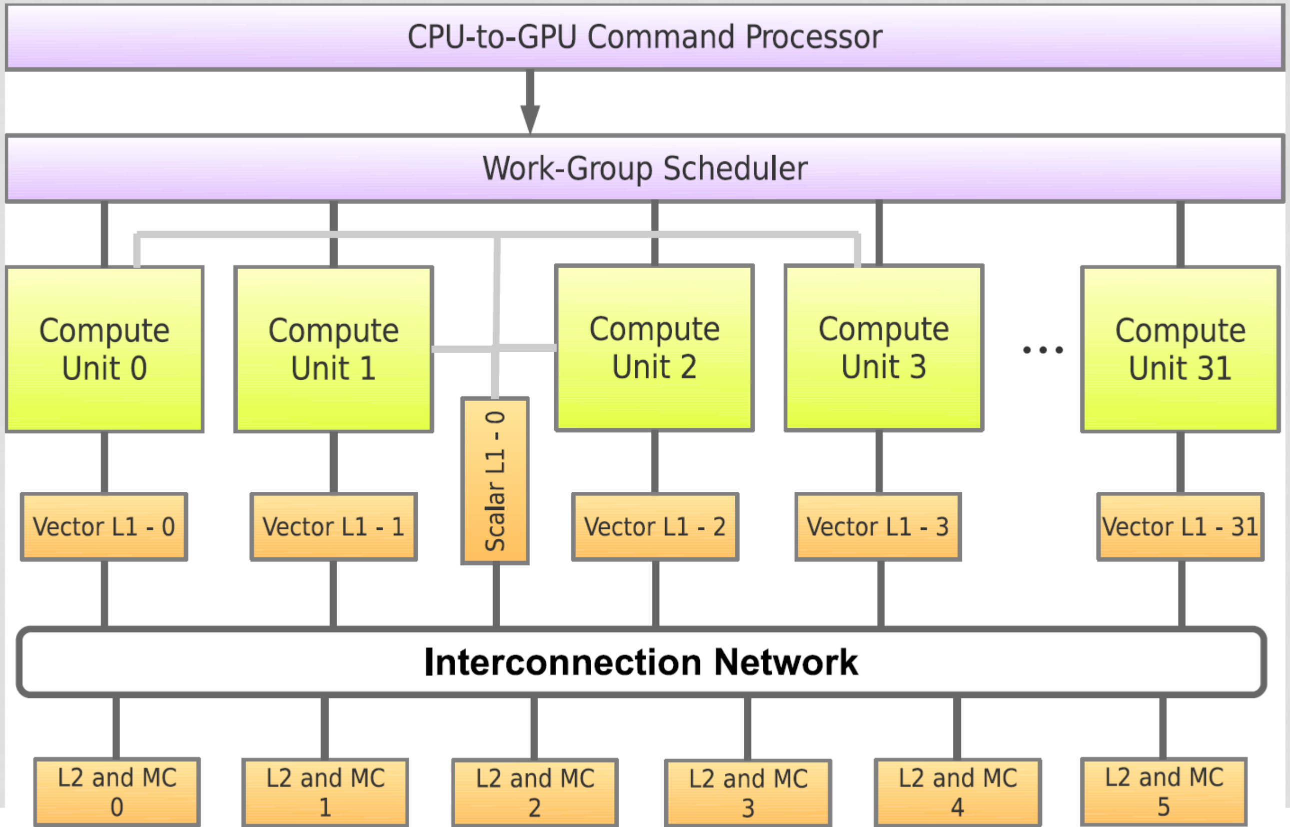
- Execution elements
  - **Work-items** execute multiple instances of the same kernel code
  - **Work-groups** are sets of work-items that can synchronize and communicate efficiently
  - The **ND-Range** is composed by all work-groups, not communicating with each other and executing in any order


ND-Range
Work-group
Work-group
Work-group
Work-group
Global Memory
Independent set of work-groups, with no synch or communication.
Work-Group
Work-item
Work-item
Work-item
Work-item
Local Memory
Intra work-group synchronization and communication allowed at this level.
Work-Item
__kernel func()
{
}
Private Memory
Common OpenCL C kernel code executed by all work-items.

SI GPU Compute Pipelines

Compute Device

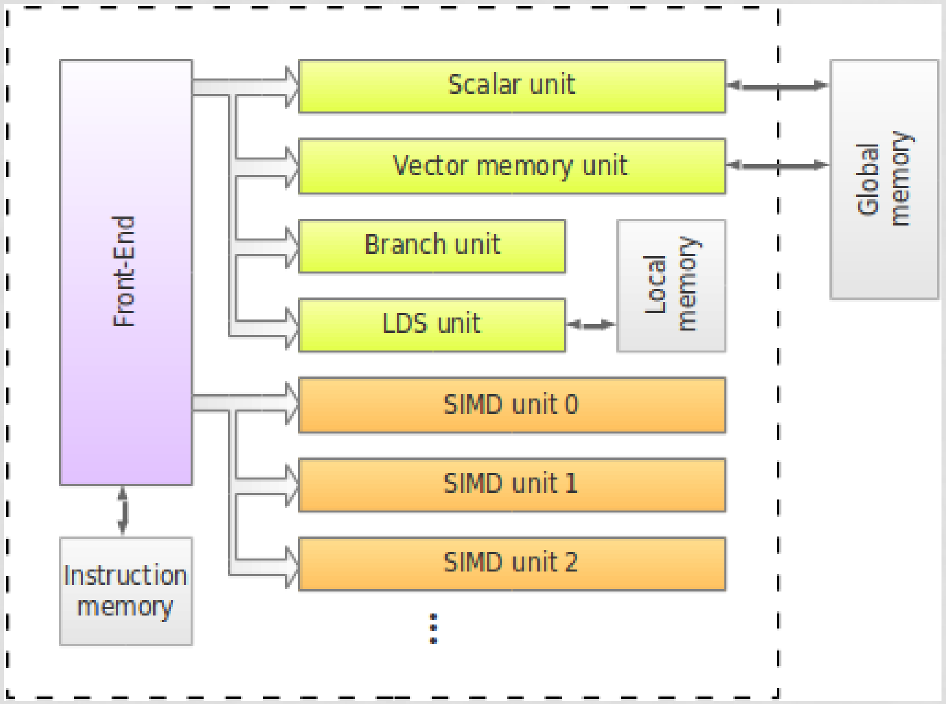
- A **command processor** receives commands and data from the CPU
- A **dispatcher** splits the ND-Range in work-groups and sends them into the compute units.
- A set of **compute units** runs work-groups.
- A **memory hierarchy** serves global memory accesses



SI GPU Compute Pipelines

Compute Unit

- The **instruction memory** of each compute unit contains a copy of the OpenCL kernel
- A **front-end** fetches instructions, partly decodes them, and sends them to the appropriate execution unit
- There is **one instance** of the following execution units: scalar unit, vector-memory unit, branch unit, LDS (local data store) unit
- There are **multiple instances** of SIMD units



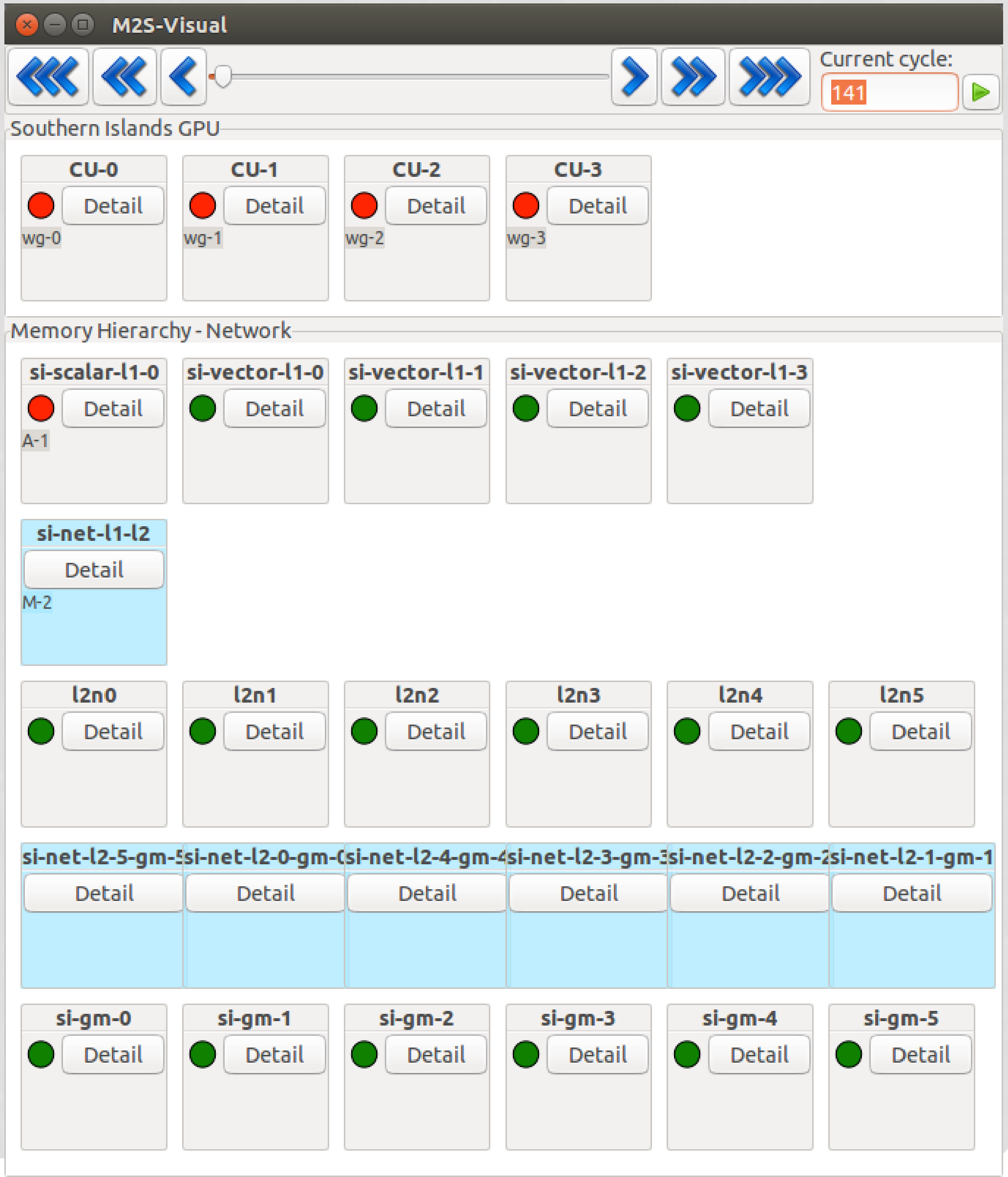
Outline

- Background and simulation methodology
- OpenCL application on the host
- OpenCL application on the GPU device
- Education through visualization
- Ongoing Work

Education through Visualization

Visualization tool - Main Panel

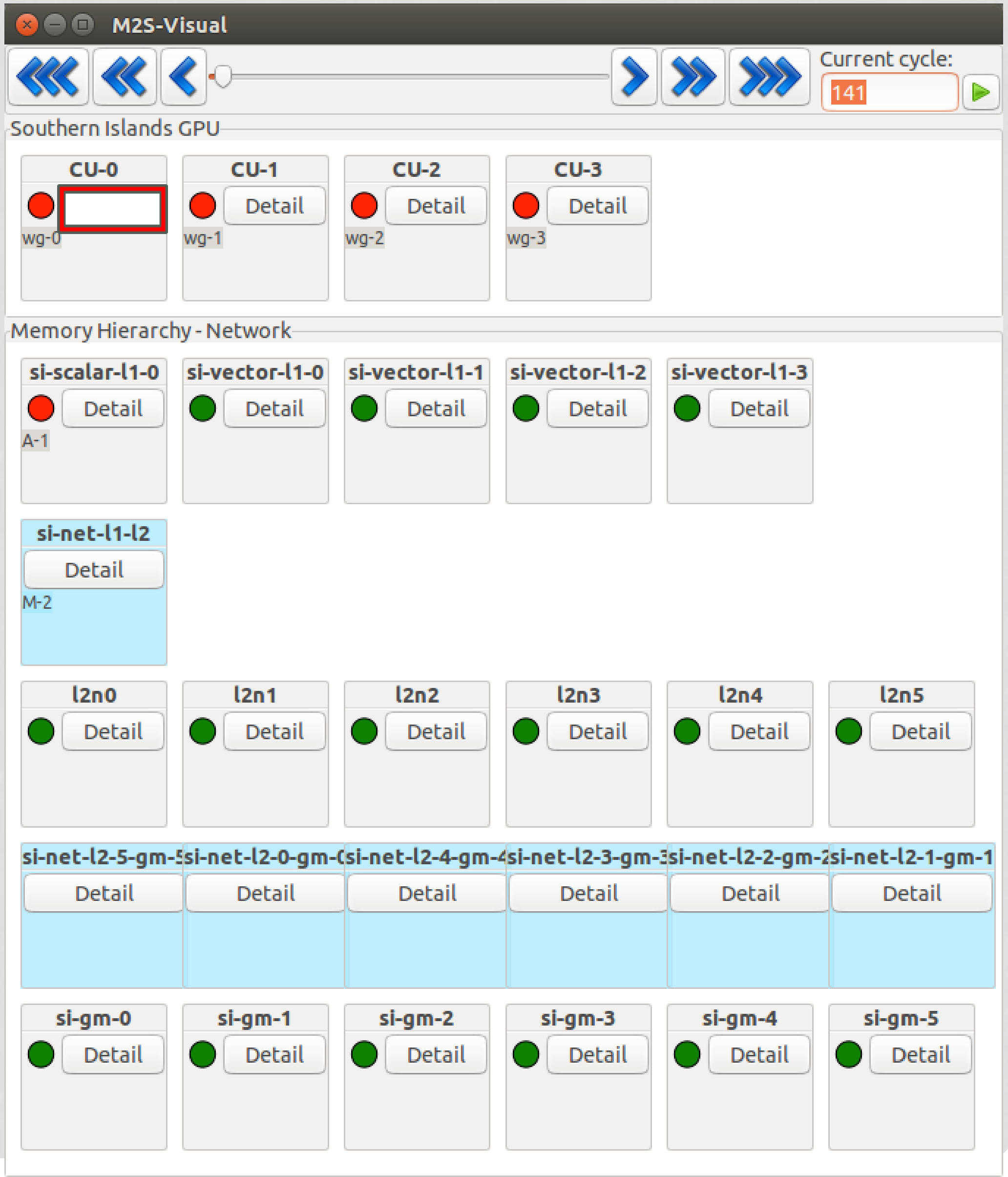
- Cycle bar on main window for navigation
- Panel on main window shows workgroups mapped to compute units



Education through Visualization

Visualization tool - Main Panel

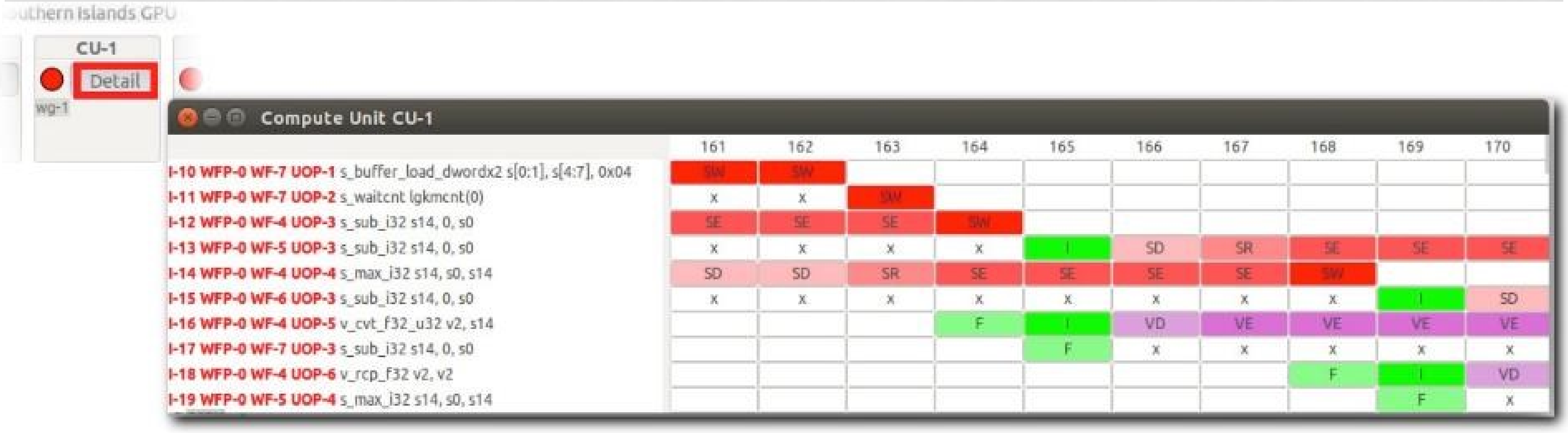
- Cycle bar on main window for navigation
- Panel on main window shows workgroups mapped to compute units
- Clicking on the **Detail** button opens a secondary window with a pipeline diagram



Education through Visualization

Visualization tool – GPU pipeline

- Front-end fetch and issue stages happens for every instruction
- After issue, the instruction continues in one of five different pipelines
    - Example: Vector unit has five pipeline stages; decode, read operand, <u>memory</u>, write to register and complete
- Each pipeline is color-coded to provide ease of differentiation

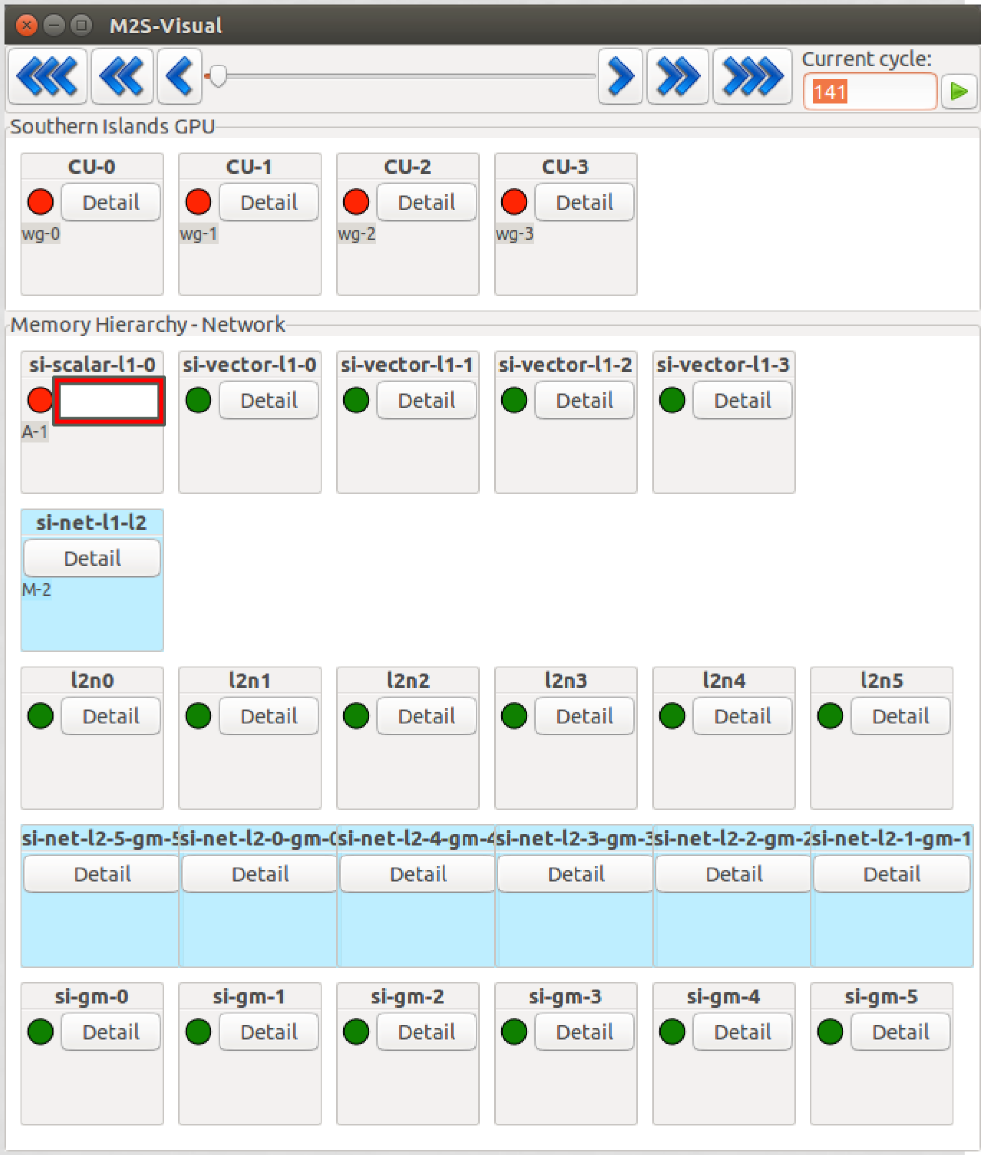


	161	162	163	164	165	166	167	168	169	170
I-10 WFP-0 WF-7 UOP-1 s_buffer_load_dwordx2 s[0:1], s[4:7], 0x04	SW	SW								
I-11 WFP-0 WF-7 UOP-2 s_waitcnt lgkmcnt(0)	x	x	SW							
I-12 WFP-0 WF-4 UOP-3 s_sub_i32 s14, 0, s0	SE	SE	SE	SW						
I-13 WFP-0 WF-5 UOP-3 s_sub_i32 s14, 0, s0	x	x	x	x	I	SD	SR	SE	SE	SE
I-14 WFP-0 WF-4 UOP-4 s_max_i32 s14, s0, s14	SD	SD	SR	SE	SE	SE	SE	SW		
I-15 WFP-0 WF-6 UOP-3 s_sub_i32 s14, 0, s0	x	x	x	x	x	x	x	x	I	SD
I-16 WFP-0 WF-4 UOP-5 v_cvt_f32_u32 v2, s14				F	I	VD	VE	VE	VE	VE
I-17 WFP-0 WF-7 UOP-3 s_sub_i32 s14, 0, s0					F	x	x	x	x	x
I-18 WFP-0 WF-4 UOP-6 v_rcp_f32 v2, v2								F	I	VD
I-19 WFP-0 WF-5 UOP-4 s_max_i32 s14, s0, s14									F	x

Education through Visualization

The Memory Hierarchy

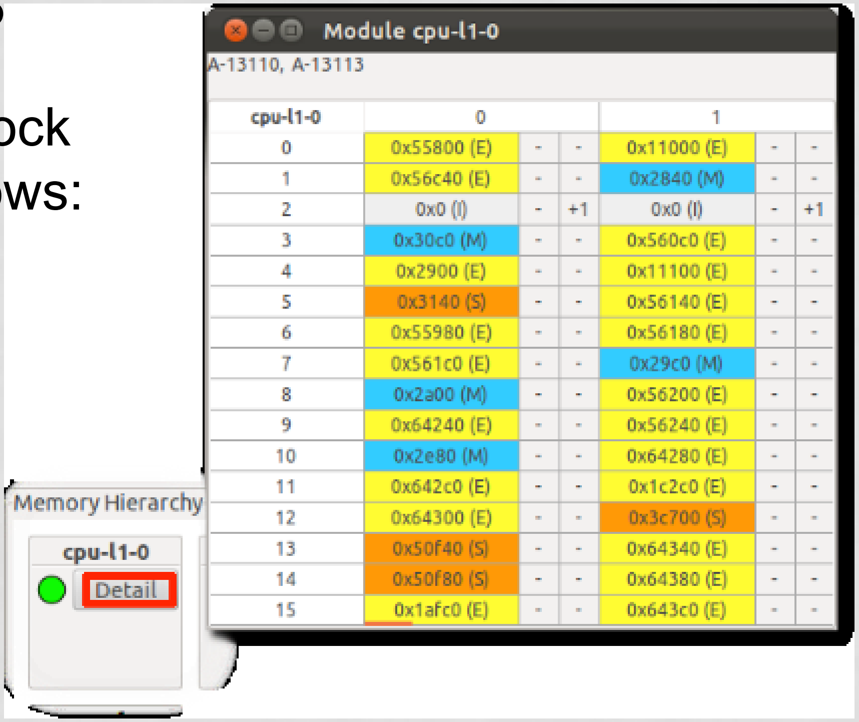
- Flexible hierarchies
  - **Any number of caches** organized in any number of levels
  - Cache levels connected through default switch cross-bar interconnects, or complex **custom interconnect** configurations
  - Clicking on the detail button opens a new window for the memory module



Education through Visualization

The Memory Hierarchy

- In this example:
    - 2-way set associative cache with 16 sets
- Each entry in the table is a cache block
- For each block visualization tool shows:
    - The state
    - The tag
    - Number of sharers
    - Number of in-flight accesses



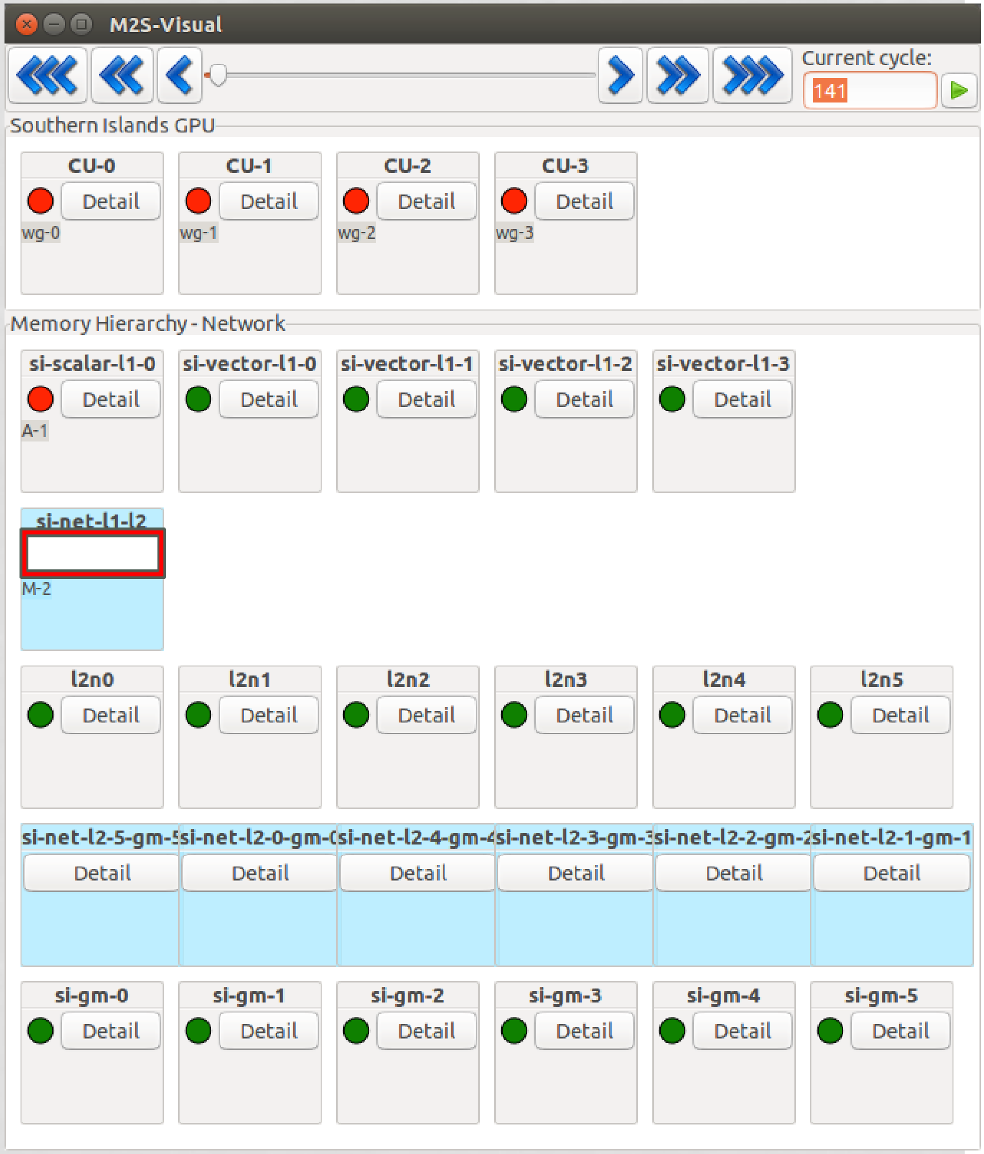
A-13110, A-13113

cpu-l1-0	0			1		
0	0x55800 (E)	-	-	0x11000 (E)	-	-
1	0x56c40 (E)	-	-	0x2840 (M)	-	-
2	0x0 (I)	-	+1	0x0 (I)	-	+1
3	0x30c0 (M)	-	-	0x560c0 (E)	-	-
4	0x2900 (E)	-	-	0x11100 (E)	-	-
5	0x3140 (S)	-	-	0x56140 (E)	-	-
6	0x55980 (E)	-	-	0x56180 (E)	-	-
7	0x561c0 (E)	-	-	0x29c0 (M)	-	-
8	0x2a00 (M)	-	-	0x56200 (E)	-	-
9	0x64240 (E)	-	-	0x56240 (E)	-	-
10	0x2e80 (M)	-	-	0x64280 (E)	-	-
11	0x642c0 (E)	-	-	0x1c2c0 (E)	-	-
12	0x64300 (E)	-	-	0x3c700 (S)	-	-
13	0x50f40 (S)	-	-	0x64340 (E)	-	-
14	0x50f80 (S)	-	-	0x64380 (E)	-	-
15	0x1afc0 (E)	-	-	0x643c0 (E)	-	-

Education through Visualization

The Interconnection Network

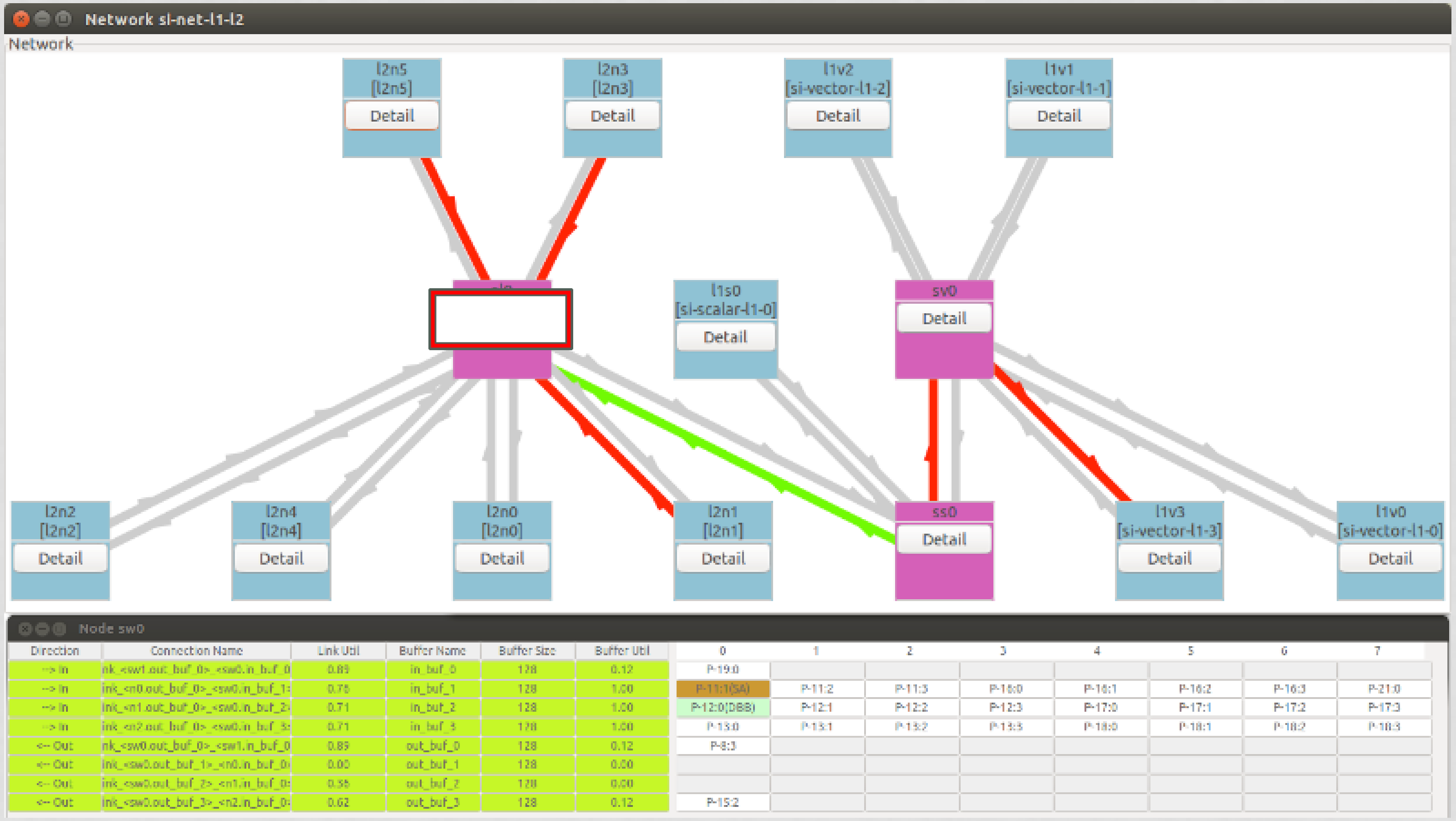
- Each message in the network is associated to an access from memory hierarchy
- Detail of the message lifetime can be followed by clicking the *detail* button on the main panel
- Detail button opens a window containing the network graph
- It shows:
  - The state of the links at each cycle
  - Congestions in the network due to the nature of OpenCL application



Education through Visualization

The Interconnection Network

- Information about individual nodes in the network graph can be obtained by clicking *detail* button on the node panel
  - State of the packets in the buffers
  - Occupancy of the buffers and links

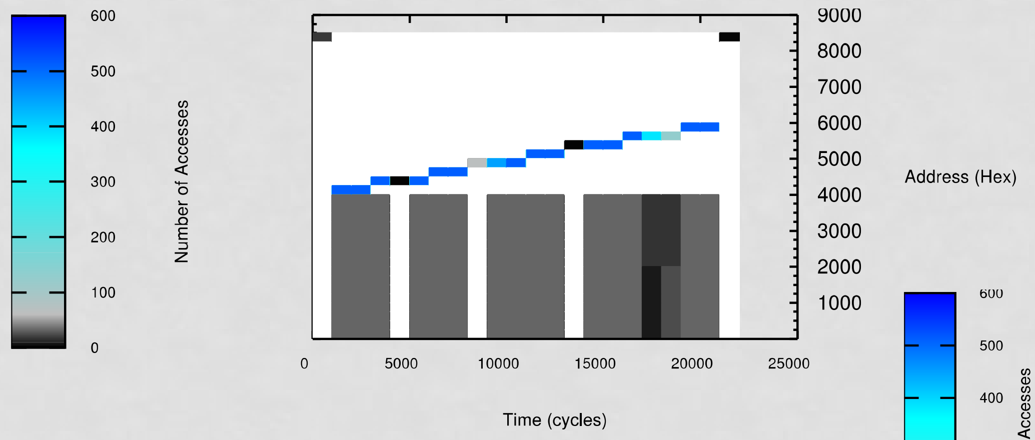


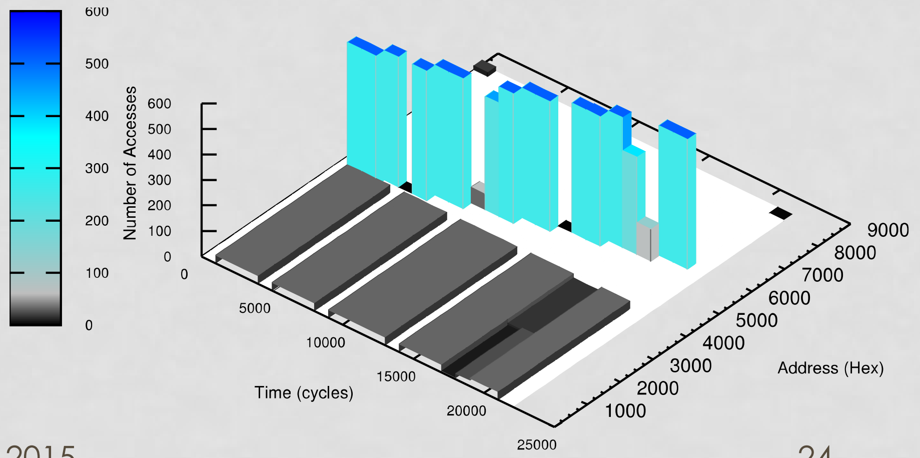
Direction	Connection Name	Link Util	Buffer Name	Buffer Size	Buffer Util	0	1	2	3	4	5	6	7
--> In	nk_<sw1.out_buf_0>_<sw0.in_buf_0	0.89	in_buf_0	128	0.12	P-19:0							
--> In	ink_<n0.out_buf_0>_<sw0.in_buf_1>	0.78	in_buf_1	128	1.00	P-11:1(SA)	P-11:2	P-11:3	P-16:0	P-16:1	P-16:2	P-16:3	P-21:0
--> In	ink_<n1.out_buf_0>_<sw0.in_buf_2>	0.71	in_buf_2	128	1.00	P-12:0(DBB)	P-12:1	P-12:2	P-12:3	P-17:0	P-17:1	P-17:2	P-17:3
--> In	ink_<n2.out_buf_0>_<sw0.in_buf_3>	0.71	in_buf_3	128	1.00	P-13:0	P-13:1	P-13:2	P-13:3	P-18:0	P-18:1	P-18:2	P-18:3
<-- Out	nk_<sw0.out_buf_0>_<sw1.in_buf_0	0.89	out_buf_0	128	0.12	P-8:3							
<-- Out	ink_<sw0.out_buf_1>_<n0.in_buf_0>	0.00	out_buf_1	128	0.00								
<-- Out	ink_<sw0.out_buf_2>_<n1.in_buf_0>	0.55	out_buf_2	128	0.00								
<-- Out	ink_<sw0.out_buf_3>_<n2.in_buf_0>	0.62	out_buf_3	128	0.12	P-15:2							

Education through Visualization

The Memory Snapshot – Identifying application patterns

- Visualizing the memory access pattern of the OpenCL workload
  - Identifying temporal and spatial locality
  - Identifying scattered or non-recurring accesses
  - Identifying patterns in loads and stores

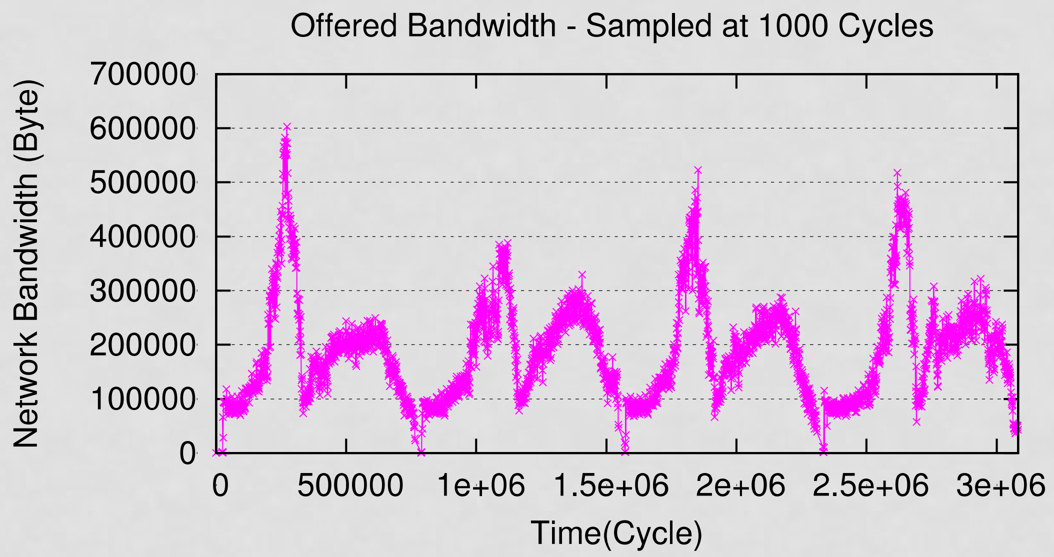




Education through Visualization

The Network Snapshot - Identifying application patterns

- Sampling network traffic
  - **Identifying network bottlenecks** in the OpenCL application execution
  - **Finding traffic patterns** in the execution of the application



Outline

Simulation Support

Supported Architectures

	Disasm.	Emulation	Timing Simulation	Graphic Pipelines
ARM	x	In progress	–	–
MIPS	x	In progress	–	–
x86	x	x	x	x
AMD Evergreen	x	x	x	x
AMD Southern Islands	x	x	x	x
NVIDIA Fermi	x	x	–	–
NVIDIA Kepler	x	x	–	–
HSA Intermediate Language	x	x	–	–

Simulation Support

Supported Benchmarks

- CPU benchmarks
  - SPEC 2000 and 2006
  - Mediabench
  - SPLASH2
  - PARSEC 2.1

- GPU benchmarks
  - AMD SDK 2.5 Evergreen
  - AMD SDK 2.5 Southern Islands
  - AMD SDK 2.5 x86 kernels
  - Rodinia
  - Parboil

Visualization Support

- HSA
  - Debugger
  - Profiler
- SimPoint
  - Fast-forwarding
  - Program phase analysis
- Accurate DRAM model
- Fault injection data
- Local memory and register file

The Multi2Sim Community

Collaboration Opportunities

- Current collaborators
  - Univ. of Mississippi, Univ. of Toronto, Univ. of Texas, Univ. Politecnica de Valencia (Spain), Boston University, AMD, NVIDIA

- Top of Trees (www.TopOfTrees.com)
  - Online framework for collaborative software development
  - Code peer reviews
  - Forum
  - Bug tracker

- Multi2Sim Project
  - 622 users registered (5/11/2015)

The Multi2Sim Community

Sponsors

AMD

ANALOG DEVICES

analogic

bk medical

LOOKING INTO HIDDEN WORLDS GORDON CenSSIS

NSF

NVIDIA CUDA™ RESEARCH CENTER

QUALCOMM®

SAMSUNG

Thank you!
Questions?