LOST IN ABSTRACTION: PITFALLS OF ANALYZING GPUS AT THE INTERMEDIATE LANGUAGE LEVEL

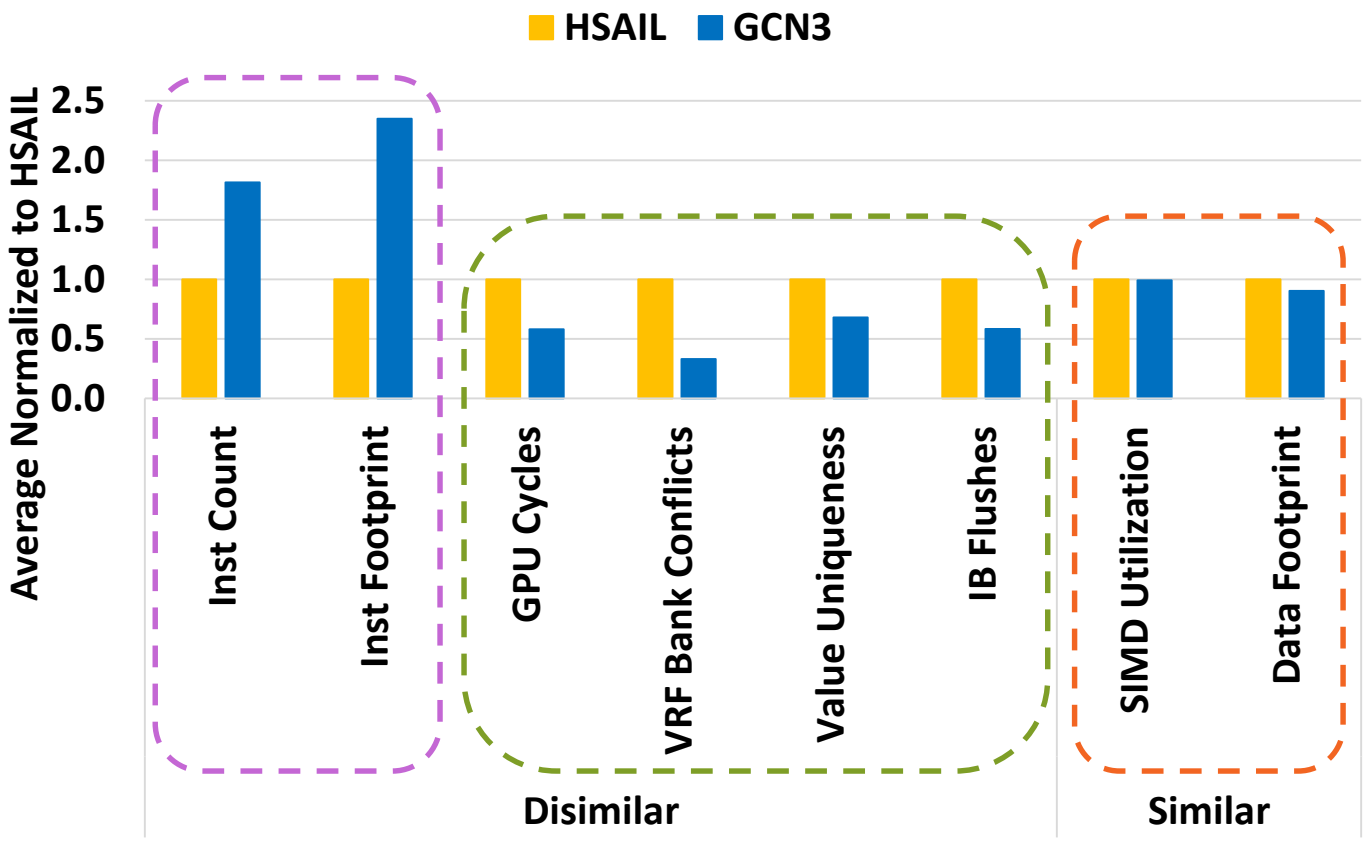
TONY GUTIERREZ, BRADFORD M. BECKMANN, ALEXANDRU DUTU, JOSEPH GROSS, JOHN KALAMATIANOS, ONUR KAYIRAN, MICHAEL LEBEANE, MATTHEW POREMBA, BRANDON POTTER, SOORAJ PUTHOOR, MATTHEW D. SINCLAIR, MARK WYSE, JIEMING YIN, XIANWEI ZHANG, AKSHAY JAIN[†], TIMOTHY G. ROGERS[†]

AMD RESEARCH, [†]PURDUE UNIVERSITY

EXECUTIVE SUMMARY

HIGH-LEVEL DIFFERENCES: IL VS. MACHINE ISA

- Intermediate Language (IL)
  - ISA for virtual machine
  - Represents data parallel execution well
  - Primarily designed for compiler optimizers
- Lots of details abstracted
  - GPU pipeline is SW managed
  - Machine ISA manages lots state for various HW/SW interfaces



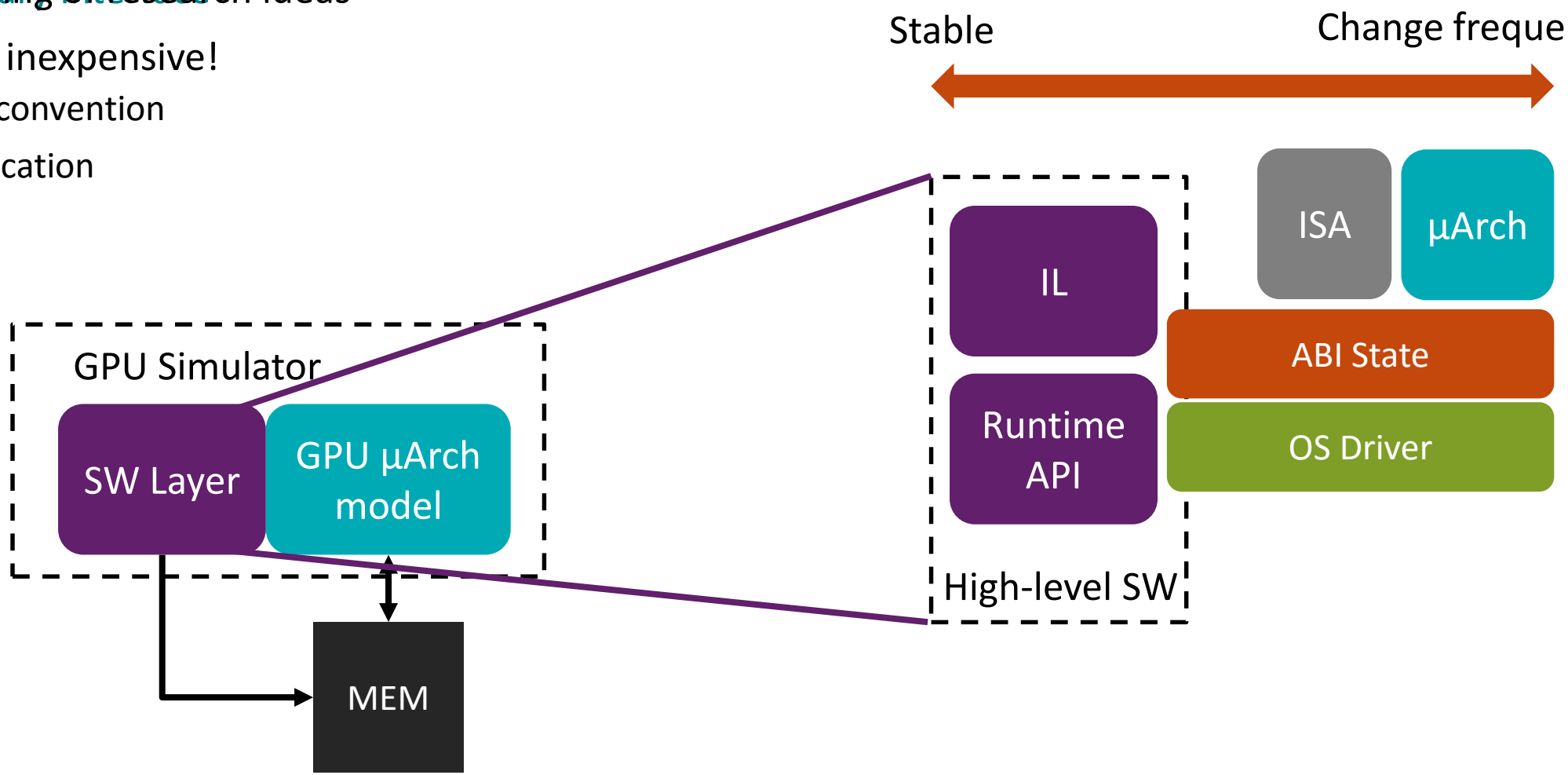
AGENDA

- Executive summary
- **Motivation and background**
- Pitfalls of analyzing GPUs using IL
- HW runtime correlation and error
- Conclusion

CYCLE-LEVEL SIMULATION IS IMPORTANT

SW ABSTRACTIONS

- Rapid prototyping of research ideas
- Open-source – inexpensive!
  - Functional call convention
  - Special value location
  - System calls
  - And more…

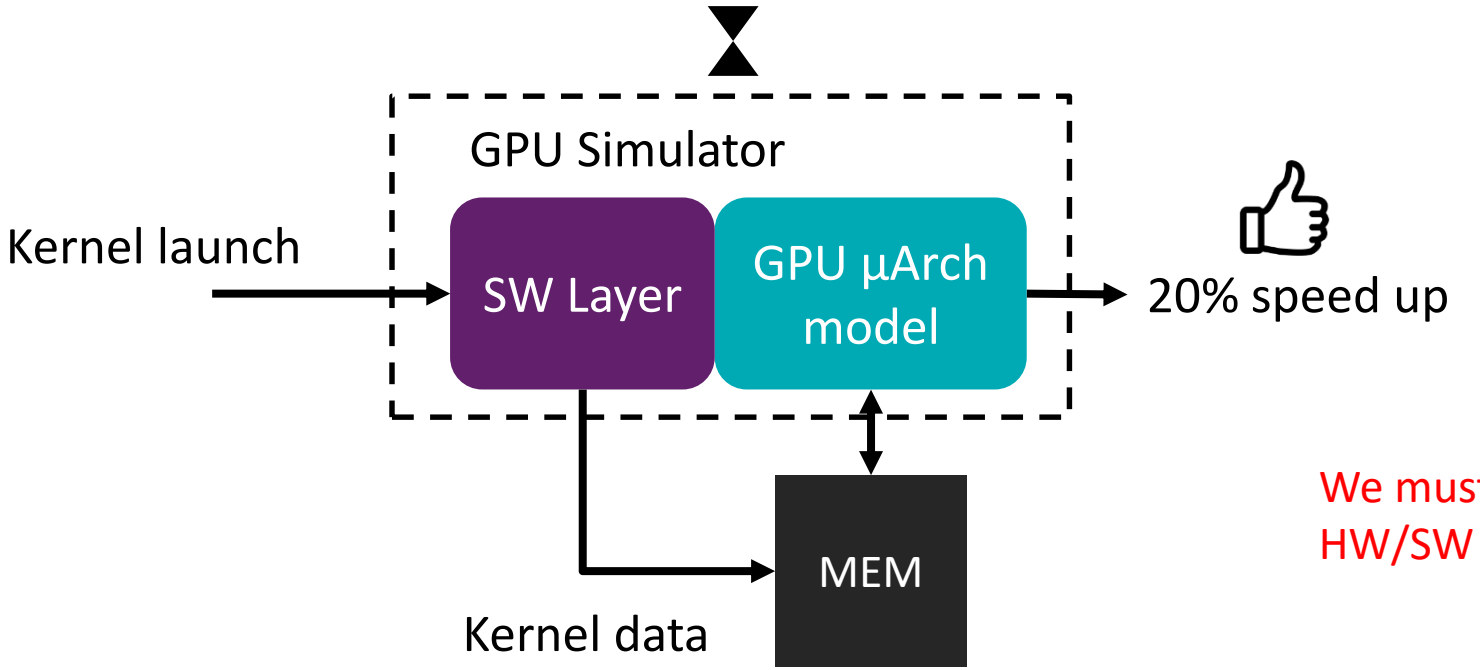


CYCLE-LEVEL SIMULATION IS IMPORTANT

GPUS: OLD VS. NEW VIEW

Old View: GPU is an accelerator for offloading data parallel functions from the CPU.

New View: GPU as primary HPC and datacenter compute device. CPU used for I/O, system services, etc.



We must understand how to properly model the HW/SW interfaces in light of this new view.

HIGH-LEVEL SOFTWARE INTERACTIONS

GPU IS A HW/SW CO-DESIGNED MACHINE

1) Two-phase compilation flow

2) Rich runtime layer

3) Co-designed HW tightly coupled to runtime

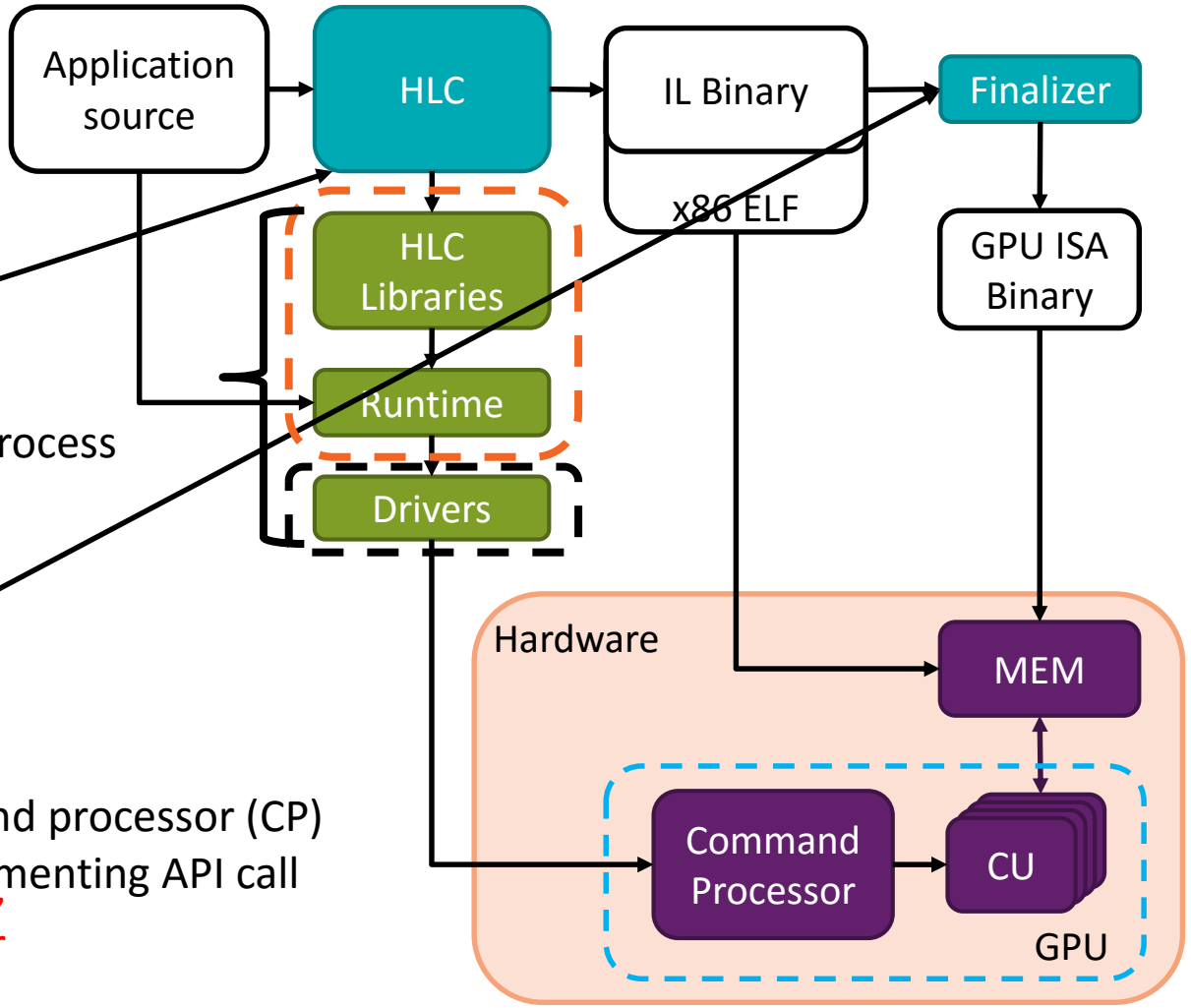
High level compiler (HLC) generates IL from source

Runtime/driver allocate and manage GPU HW per process

Runtime API used to trigger dispatch

Finalizer or JIT generates machine ISA from IL

GPU command processor (CP) aids in implementing API call



By coupling simulation infrastructure to OS layer only, we have the flexibility to easily support other programming models.

GPU SIMULATORS

OVERVIEW OF SOFTWARE ABSTRACTIONS IN STATE-OF-THE ART SIMULATORS

	Runtime Support	ISA	ABI Support
GPGPU-Sim	Emulated	IL	Simplified by simulator
Multi2Sim	Emulated	IL/Machine ISA	Simplified by simulator
gem5	Emulated	IL	Simplified by simulator

GPU SIMULATORS

OVERVIEW OF SOFTWARE ABSTRACTIONS IN STATE-OF-THE ART SIMULATORS

	Runtime Support	ISA	ABI Support
GPGPU-Sim	Emulated	IL	Simplified by simulator
Multi2Sim	Emulated	IL/Machine ISA	Simplified by simulator
gem5	Off-the-shelf	Machine ISA	Models real ABI

- This work adds
  - GCN3 support – AMD's GPU machine ISA
  - HSA ABI
  - Support for off-the-shelf ROCm stack (user space)
  - Emulated ROCk
- We evaluate the effects on simulation for both HSAIL and GCN3
- We demonstrate that HSAIL introduces significant additional error

Radeon Open Compute Platform (ROCm)

- The open-source implementation of HSA principles for AMD Devices
- ROCr – runtime
- ROCt – thunk (user driver)
- ROCk – kernel driver
- HCC – heterogenous compute compiler

AGENDA

AMD

- Executive summary
- Motivation and background
- **Pitfalls of analyzing GPUs using IL**
  - Methodology
  - Quantitative analysis
    - Instruction scheduling
    - Kernel argument access
    - Control flow
    - Instruction expansion
- HW runtime correlation and error
- Conclusion

METHODOLOGY

- gem5's GPU model
  - With GCN3 support added
  - Support for HSA standard and off-the-shelf ROCm
- ROCm version 1.1
  - HCC-hsail clang compiler version 3.5
    - Same binary used on hardware/gem5
    - For HSAIL extract the kernel code from binary before finalizing to GCN3
- HW runs on AMD Pro A12-8800B APU
  - Radeon open compute profiler (RCP) used to capture hardware data

Workload	Description
Array BW	Memory streaming
Bitonic Sort	Parallel merge sort
CoMD	DOE Molecular-dynamics algorithms
FFT	Digital signal processing
HPGMG	Ranks HPC systems
LULESH	Hydrodynamic simulation
MD	Generic molecular-dynamics algorithms
SNAP	Discrete ordinates neutral particle transport application
SpMV	Sparse matrix-vector multiplication
XSBench	Monte Carlo particle transport simulation

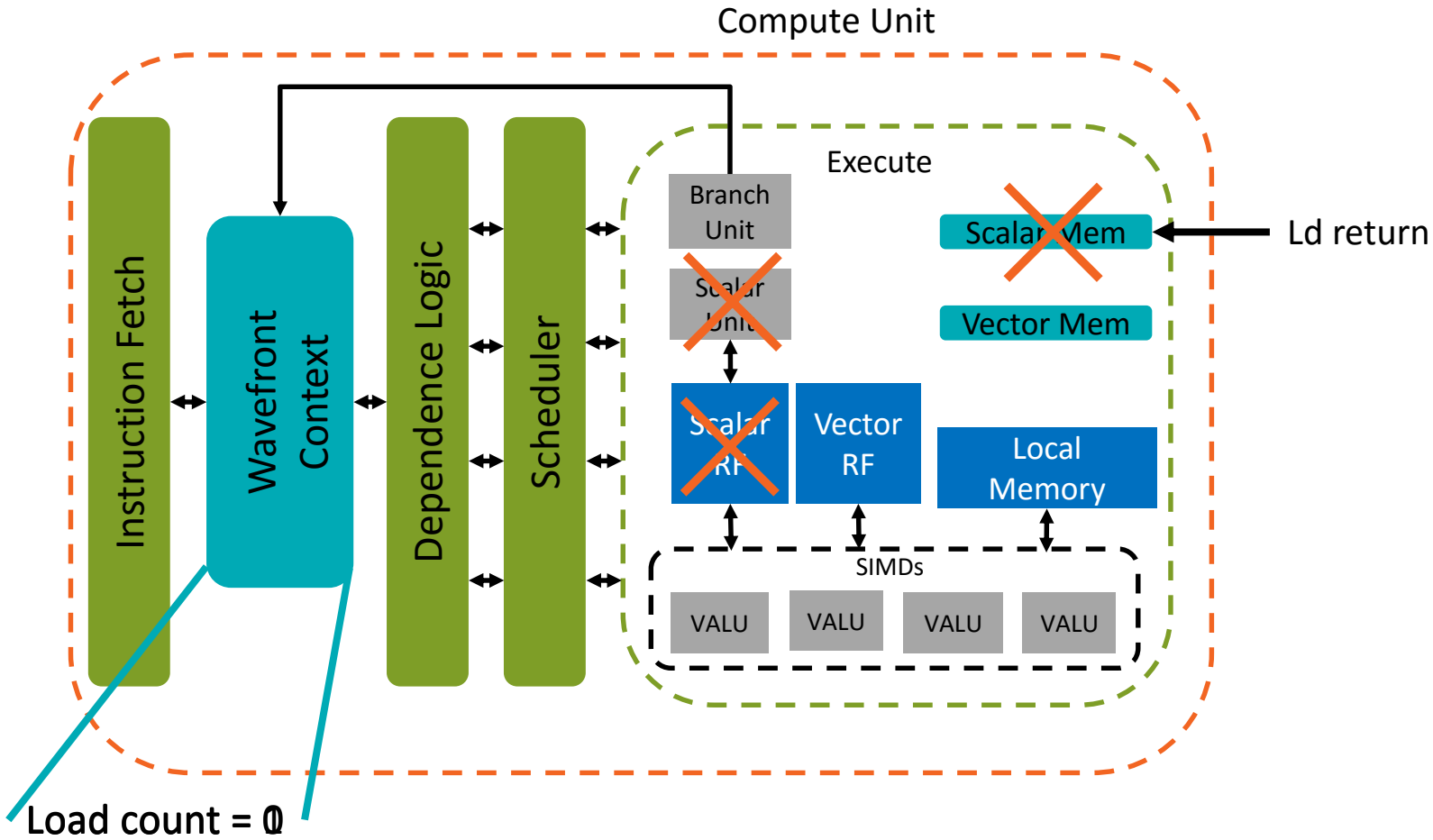
KNOWLEDGE OF UNDERLYING HW RESOURCES

GCN3 VIEW OF COMPUTE UNIT PIPELINE

```
s_load s[0]
   ●
   ●
s_waitcnt(0)
v_add v6, s0, v0
```

WF is waiting for memory

Load count = 0

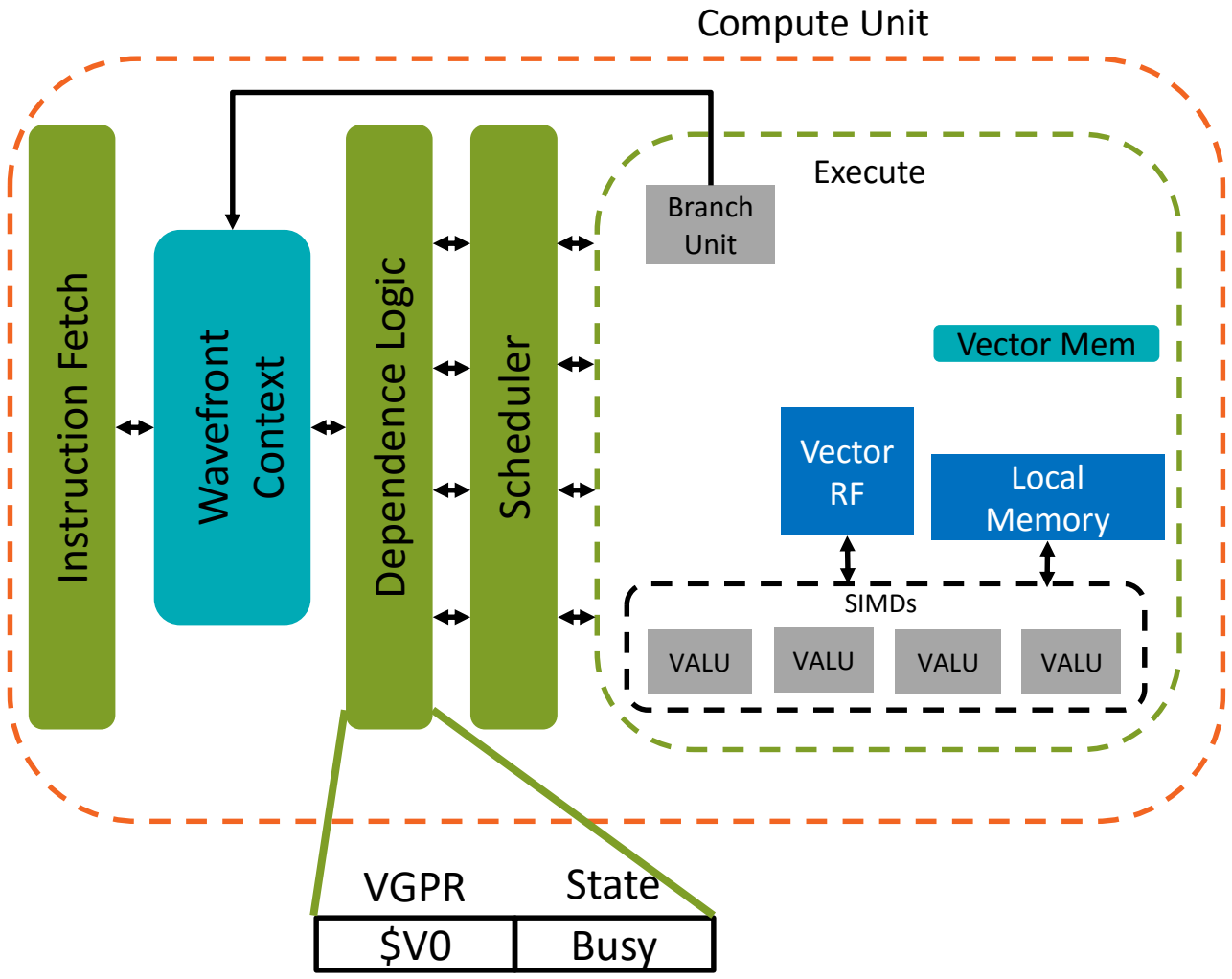


Not utilized by HSAIL instructions

KNOWLEDGE OF UNDERLYING HW RESOURCES

HSAIL'S VIEW OF COMPUTE UNIT PIPELINE

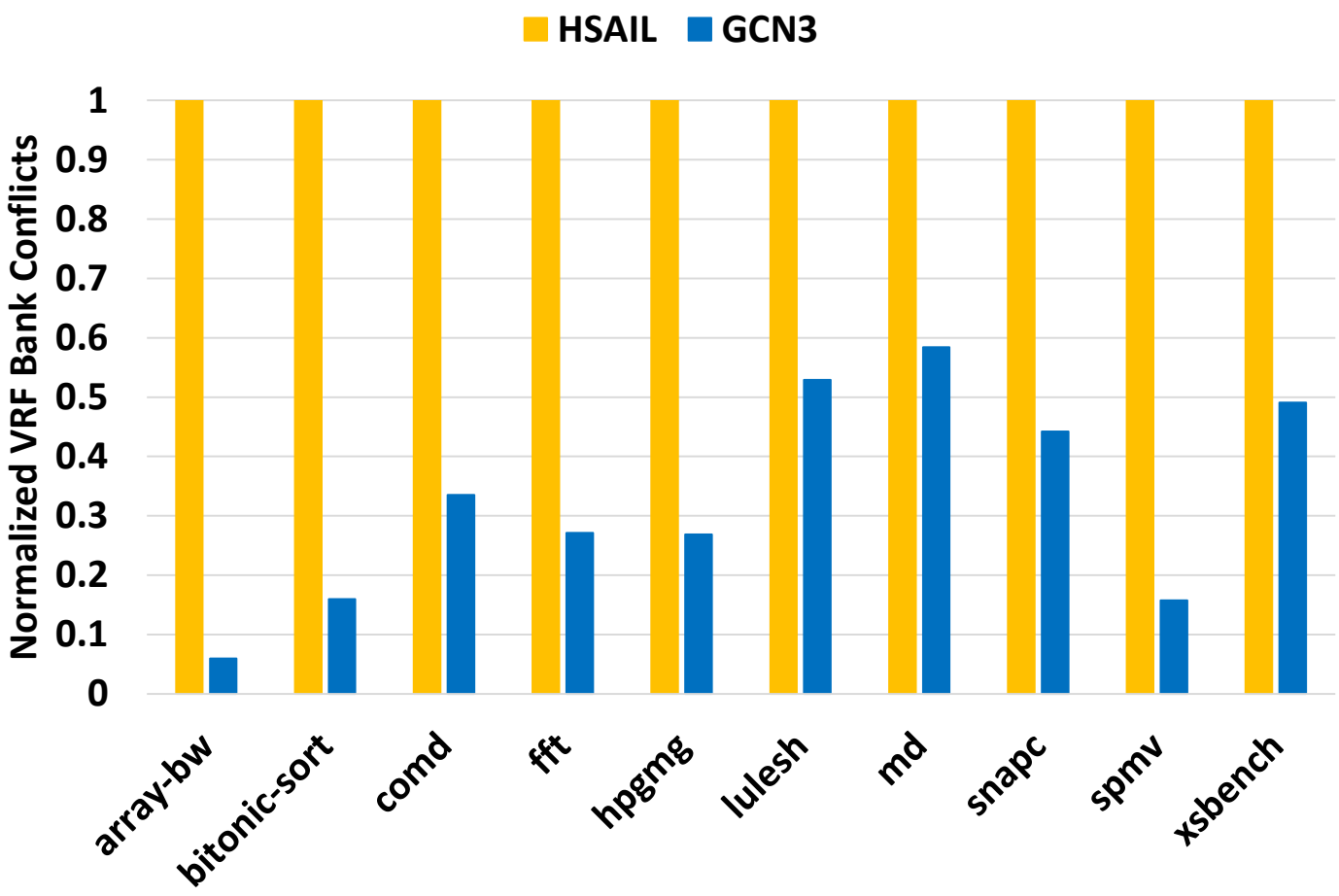
```
ld $v0, [%__arg_p1]
add $v2, $v0, $v1
```



VRF BANK CONFLICTS

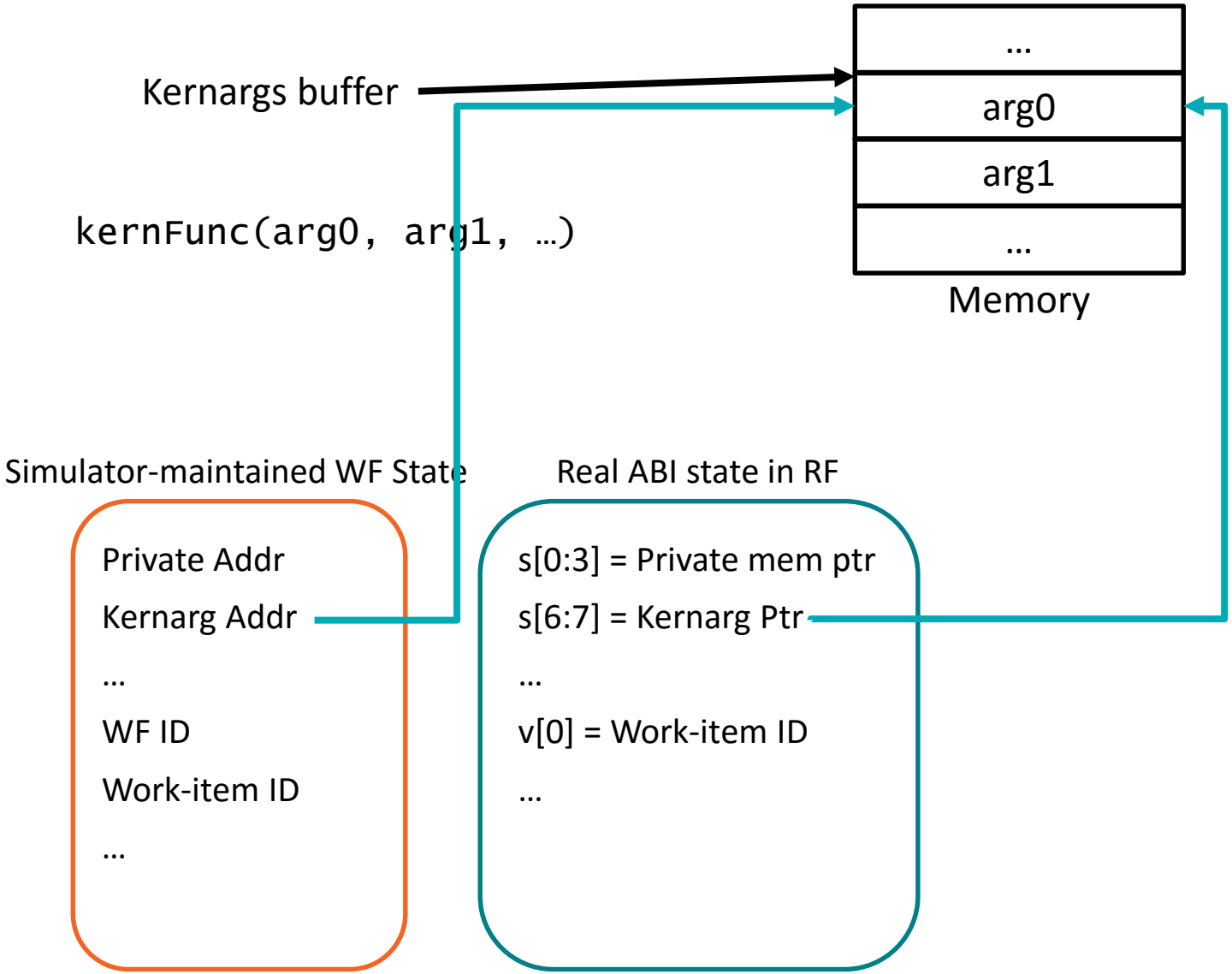
INSTRUCTION SCHEDULING EFFECTS

- Much better instruction scheduling from GCN3 compiler
- Higher reuse distance
  - Less probability of accesses same banks/registers
- Better register allocation



KERNEL ARGUMENT ACCESS

MEMORY SEGMENTS



HSAIL

```
# load kernarg 0 ptr
ld_kernarg $d0, [%__arg_p1]
# load kernarg 0
ld_global $d1, [$d0]
```

- HSAIL ld specifies **segment** + **arg num offset** only

GCN3

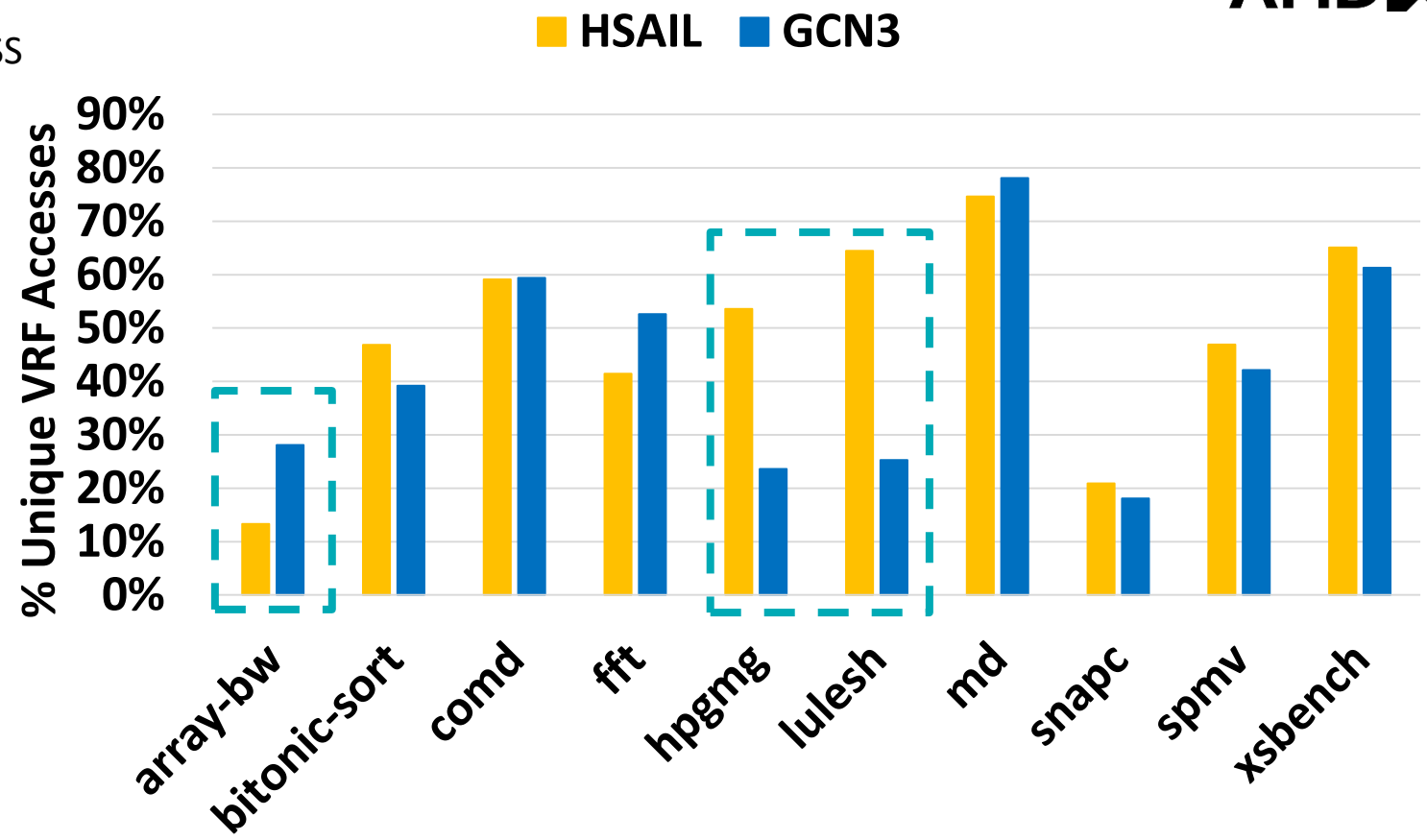
```
# s[6:7] = Kernarg ptr
s_load_dword s[0:1], s[6:7], 0x08
```

- GCN3 s_load_dword uses **real address** + **byte offset**
- ABI specifies Kernarg pointer stored in s[6:7]

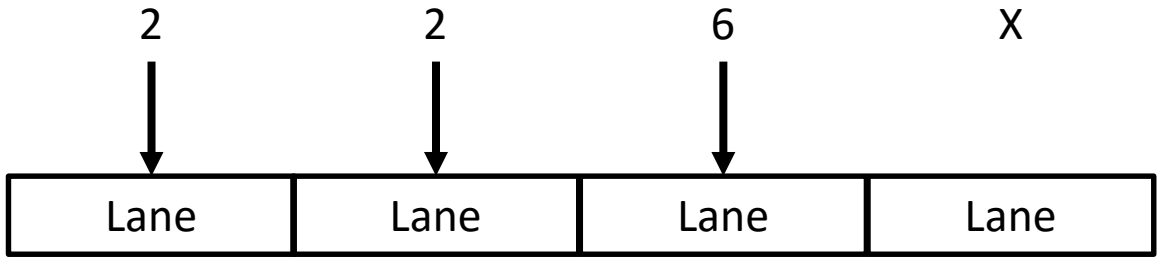
VRF VALUE UNIQUENESS

SCALAR UNIT DOES NOT IMPROVE VALUE UNIQUENESS

- Many VRF R/W are redundant
  - Typically GCN3 codes experience more value uniqueness
  - ABI abstraction in HSAIL hides some value redundancy
    - Base address storage



Uniqueness definition: ratio of unique lane values to active lanes.



EXEC = 1110 %Unique = 66%

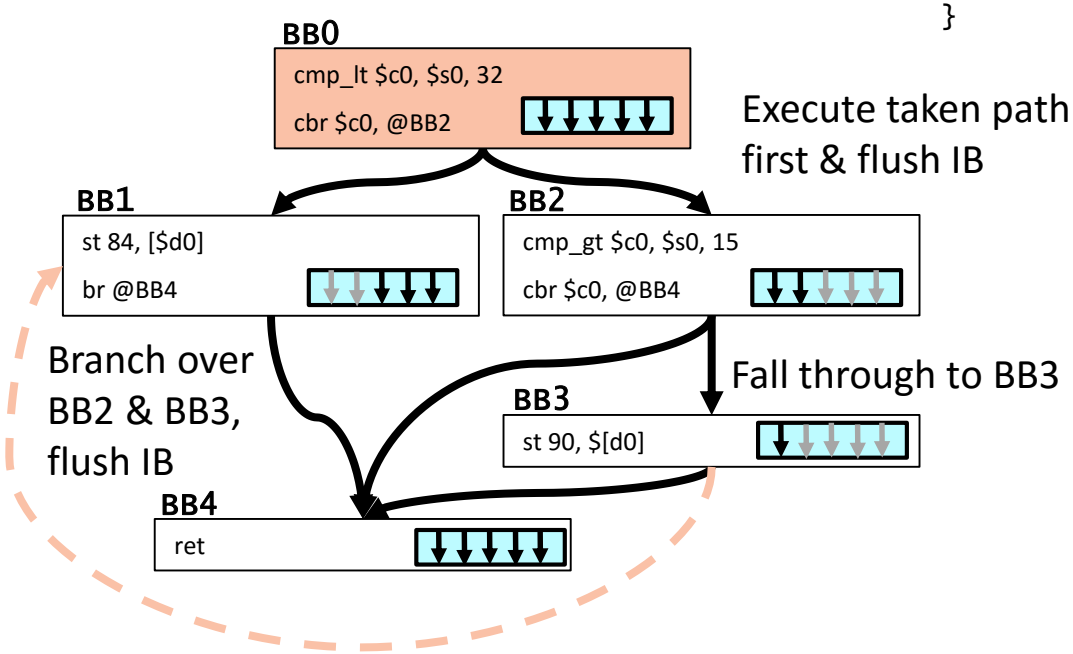
CONTROL FLOW DIVERGENCE

SIMT VS. VECTOR EXECUTION MODEL

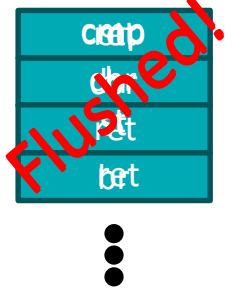
Source code:

```
if (i > 31) {
    *x = 84;
} else if (i < 16) {
    *x = 90;
}
```

HSAIL



Instruction buffer



Reconvergence point reached, HW initiated jump to divergent path

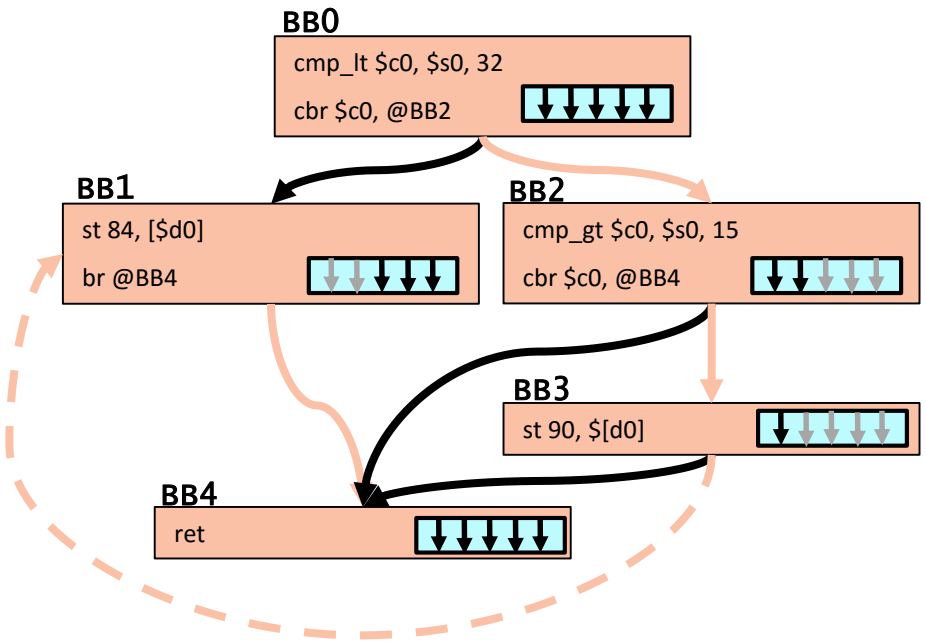
CONTROL FLOW DIVERGENCE

SIMT VS. VECTOR EXECUTION MODEL

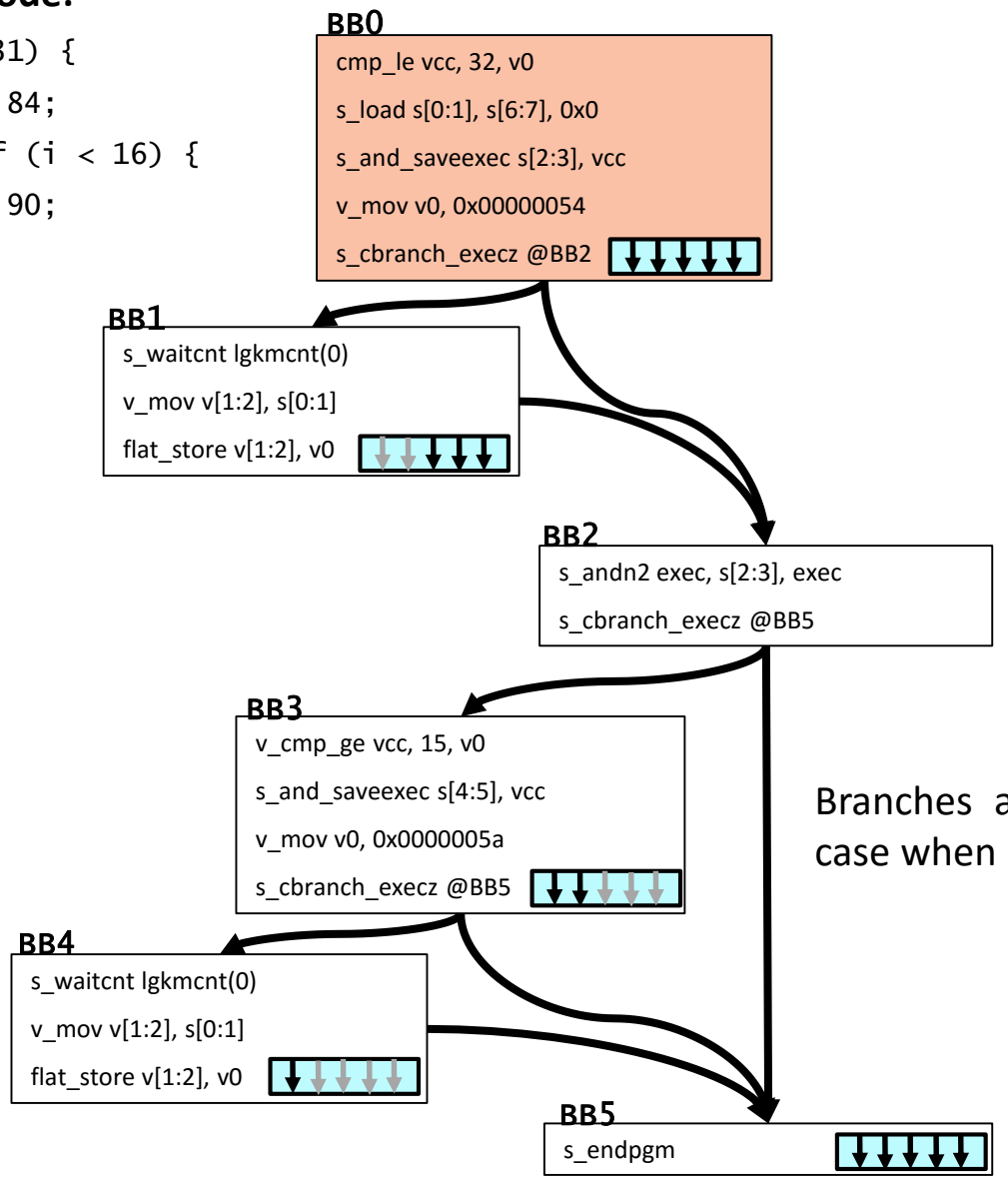
Source code:

```
if (i > 31) {
    *x = 84;
} else if (i < 16) {
    *x = 90;
}
```

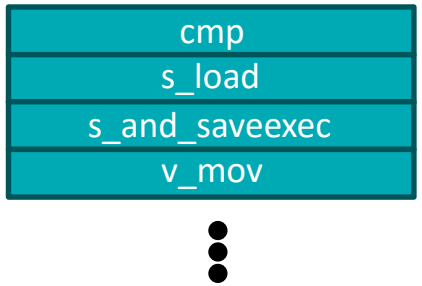
HSAIL



GCN3



Instruction buffer

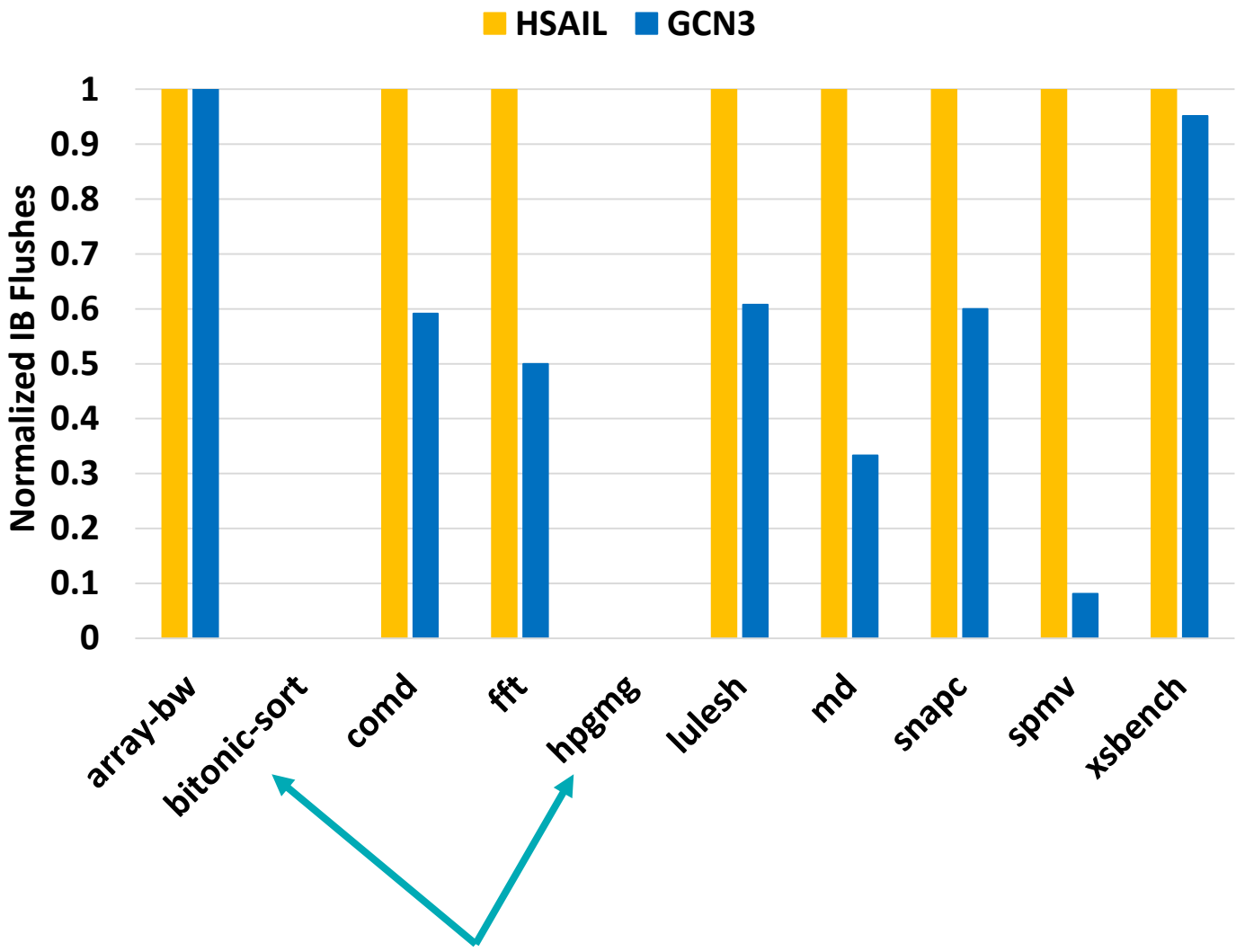


Branches are optimizations for case when EXEC = 0 for a BB

INSTRUCTION BUFFER FLUSHES

SIMT VS. VECTOR EXECUTION MODEL

- GCN3 relies on predication more frequently
  - Requires fewer hardware “jumps”
  - Requires fewer IB flushes



CODE EXPANSION

- HSAIL instructions are semantically powerful
  - Single HSAIL inst => several GCN3 insts

HSAIL

```
# Perform divide
div $d17, $d11, $d1
```

Declarative: what operation to perform

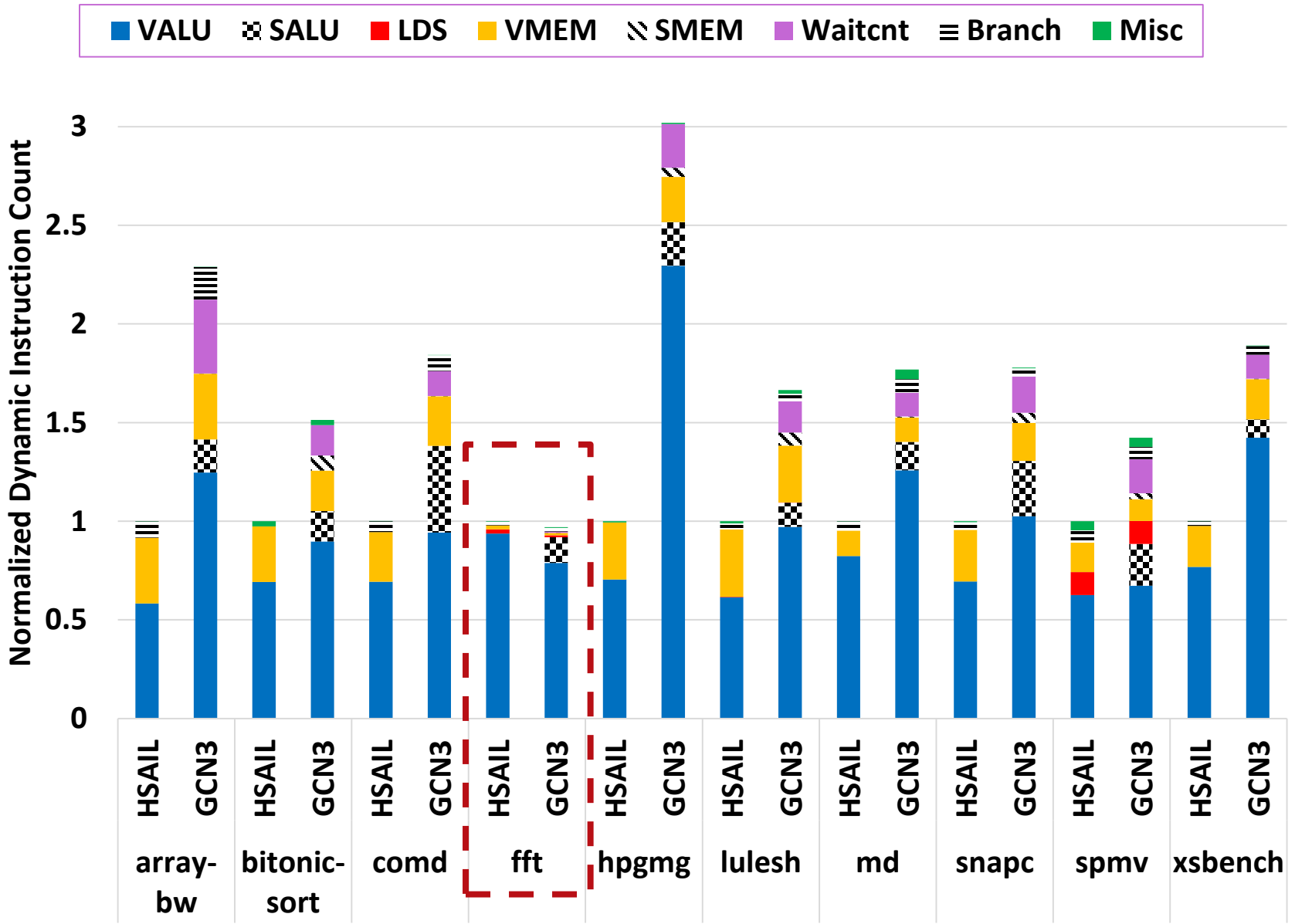
Imperative: how to perform operation (Newton-Raphson Method)

GCN3

```
# Scale D
v_div_scale v[3:4], vcc, v[1:2], v[1:2], s[4:5]
v_mov v[5:6], s[4:5]

# Scale N
v_div_scale v[5:6], vcc, v[5:6], v[1:2], v[5:6]

# 1/D
v_rcp v[7:8], v[3:4]

# Calculate Q and E
v_fma v[9:10], -v[3:4], v[7:8], 1.0
v_fma v[7:8], v[7:8], v[9:10], v[7:8]
v_fma v[9:10], -v[3:4], v[7:8], 1.0
v_fma v[7:8], v[7:8], v[9:10], v[7:8]
v_mul v[9:10], v[5:6], v[7:8]
v_fma v[3:4], -v[3:4], v[9:10], v[5:6]

# Calculate final Q
v_div_fmas v[3:4], v[3:4], v[7:8], v[9:10]

# Fixup Q
v_div_fixup v[1:2], v[3:4], v[1:2], s[4:5]
```

INSTRUCTION MIX

- GCN3 executes far more dynamic instructions
  - Code expansion
  - Extra instructions due to ABI
  - Dependency handling instructions
  - Intermixed scalar instructions
  - Varies across applications



AGENDA

AMD

- Executive summary
- Motivation and background
- Pitfalls of analyzing GPUs using IL
- **HW runtime correlation and error**
- Conclusion

HW CORRELATION

PERFORMANCE ERROR

Correlation		Mean Abs. Error	
HSAIL	GCN3	HSAIL	GCN3
0.972	0.973	75%	42%

- HSAIL adds significant, and unpredictable error
  - Inherent to using HSAIL and emulated runtime
  - With only publicly available information, GCN3 still improves error by > 30%
- Results correlate well
  - May indicate preservation of performance trends
  - Microarchitectural events, and absolute performance still left with significant error

CONCLUSION

AMD

- GPU Compute workloads are becoming more complex
  - Utilize many components of the system simultaneously
  - Lots of complex HW/SW interactions
- Modeling the full stack correctly is important
  - Challenging, as HW changes frequently
  - Abstracting at OS only provides nice balance
- Machine ISA instructions accurately capture application behavior
  - Microarchitecture characteristics skewed by IL
  - Machine ISA captures real HW events/state
- GPU simulators must capture full-system behavior and machine ISA/microarchitecture interaction

INTERESTED IN LEARNING MORE?

MODEL ENHANCEMENTS AND PUBLIC RELEASE

- Public release of GCN3 ISA and ROCm support coming soon
- ISCA 2018 tutorial
  - Will cover:
    - Model updates
    - ROCm simulation in detail
    - Toolchain and benchmarks
      - *HSAIL has been deprecated
      - Toolchain uses LLVM IL and compilers directly produces ISA binary

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION


AMD