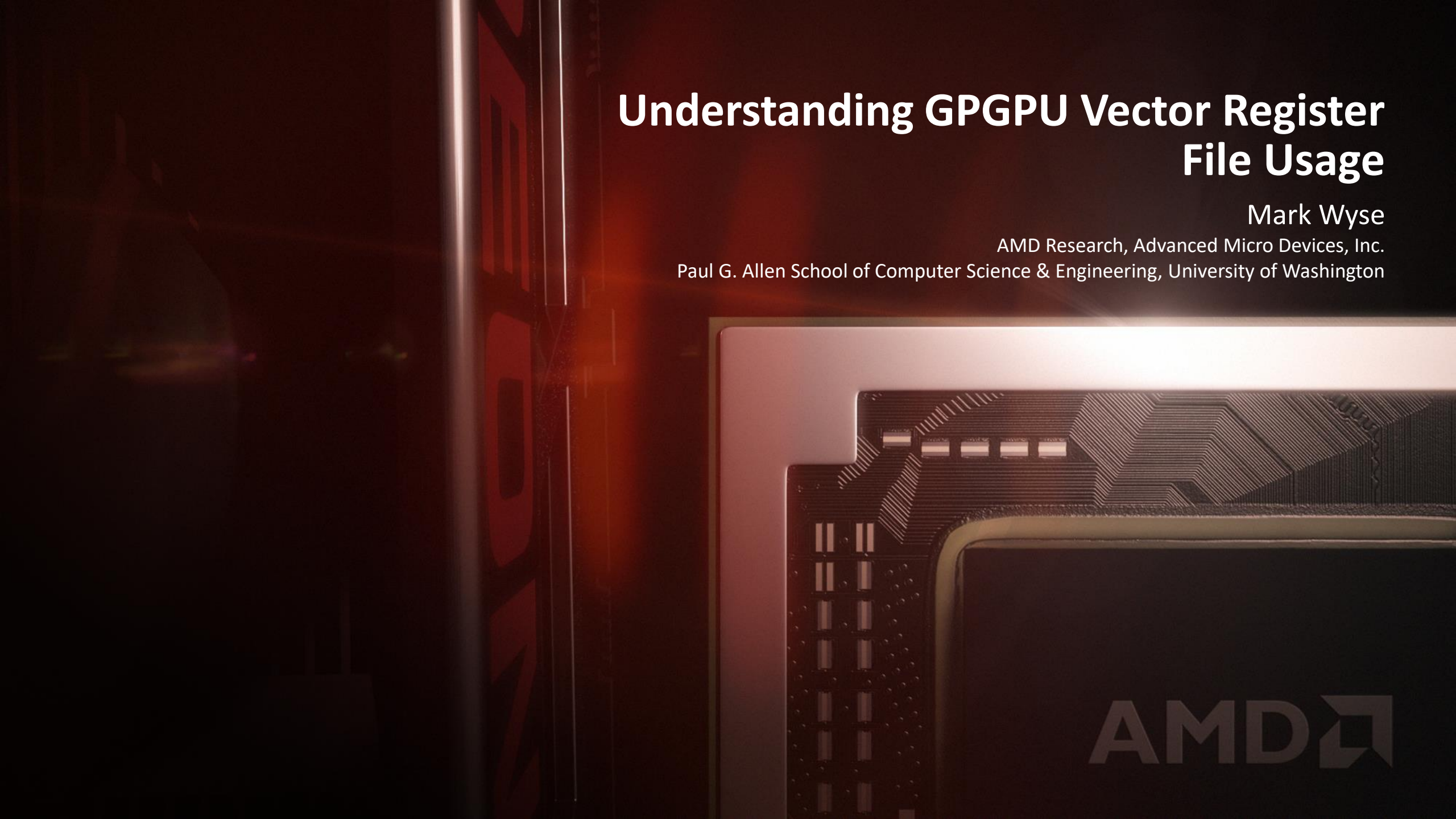
Understanding GPGPU Vector Register File Usage

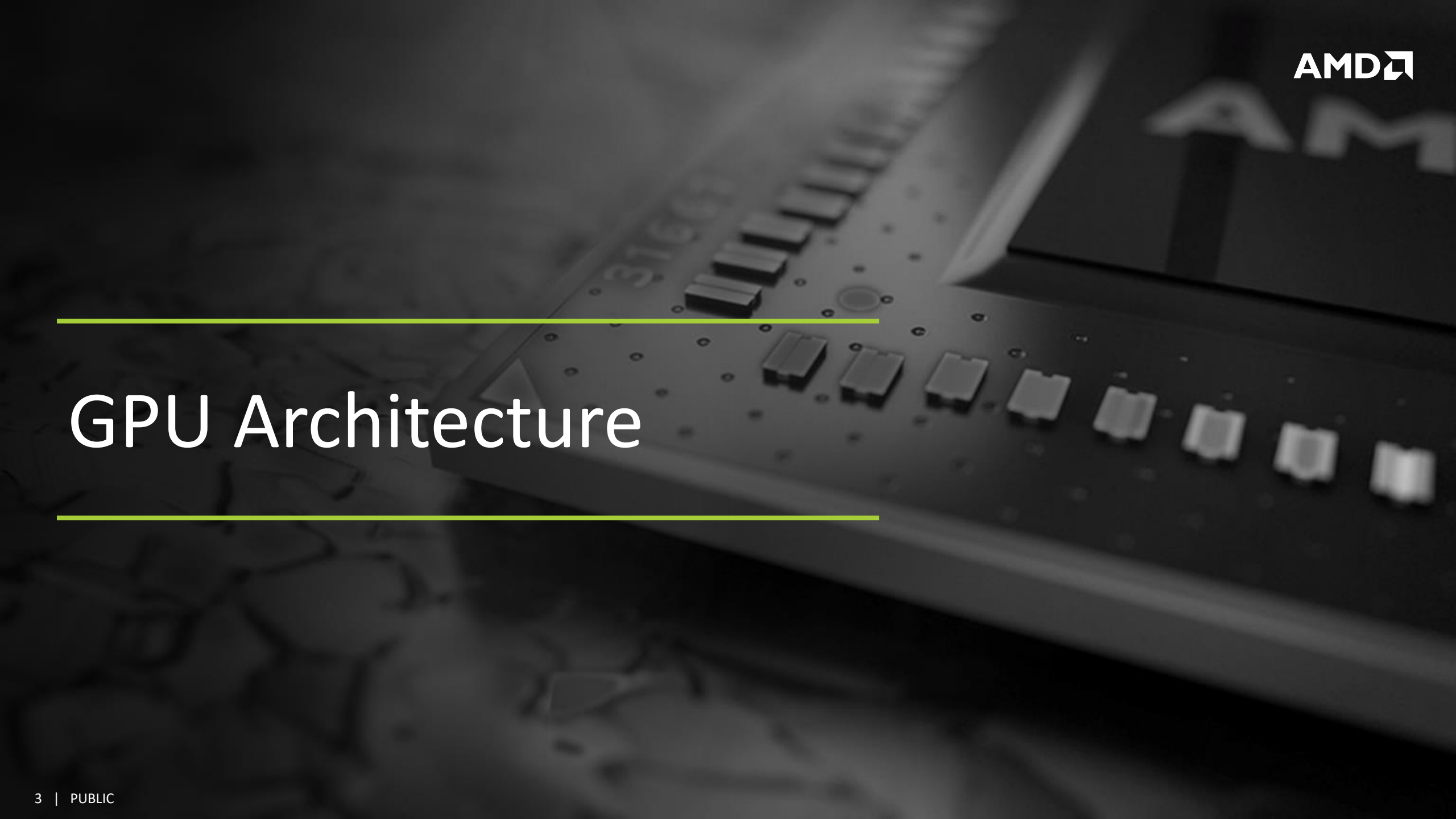
Mark Wyse
AMD Research, Advanced Micro Devices, Inc.
Paul G. Allen School of Computer Science & Engineering, University of Washington



AGENDA

AMD

- GPU Architecture
- gem5 Model and Simulation
- Vector General Purpose Register Usage
- Operand Buffering and Register File Caching
- Dispatch Limits and Wave Level Parallelism
- Conclusion



GPU Architecture

GPU ARCHITECTURE

HIGH LEVEL OVERVIEW

- Massively multi-threaded compute engines
- Use parallelism to hide memory latency
- Programmed with Single Instruction, Multiple Thread (SIMT) model
  - Kernel – Threads – Workgroup – Wavefront
- Hardware executes instructions using Single Instruction, Multiple Data (SIMD) model
  - Lockstep execution of threads in a Wavefront
- Huge amount of on-chip context to enable single cycle thread context switching
  - Register files are larger than the data caches

EXAMPLE COMPUTE UNIT ARCHITECTURE

- Compute Unit (CU) responsible for executing workgroups
- Each CU contains:
  - Wavefront (WF) contexts
  - SIMD Vector ALUs
  - Scalar ALUs
  - Register Files
  - Memory Pipelines
  - L1 Vector Data Cache
  - Local Data Share (LDS)
- Each CU connects to:
  - Scalar Cache
  - Instruction Cache
  - L2 Data Cache / DRAM

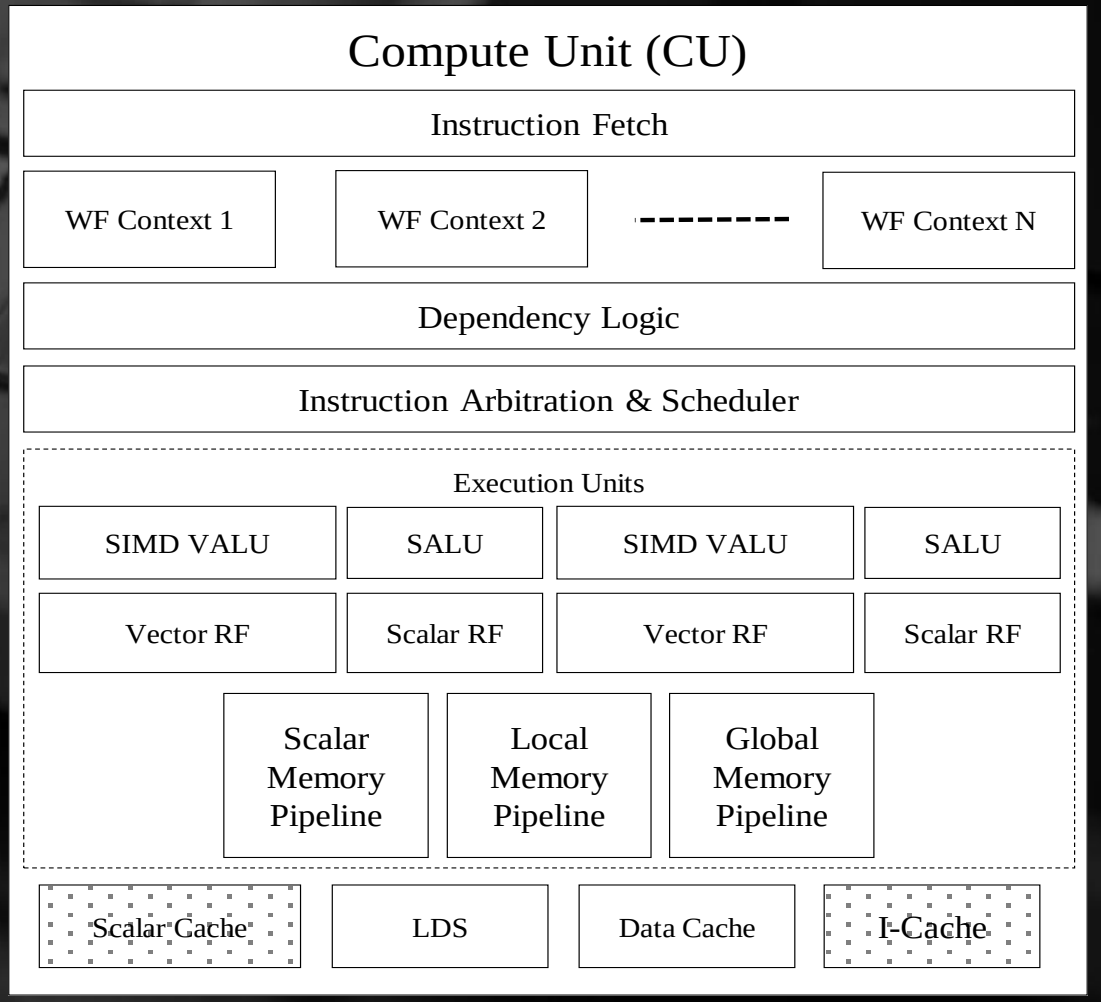


Figure 1. Sample CU Architecture

VECTOR REGISTER FILE SUBSYSTEM

- Each VALU has a private Vector Register File (VRF)
- 512 Vector General Purpose Registers (VGPRs) per VRF
  - 128 KB VGPR storage per VRF/SIMD VALU
  - VGPRs distributed across 4 SRAM banks
  - 1 read, 1 write per cycle per bank
- Operand Buffer (OB)
  - Instruction FIFO
  - Reads vector operands for VALU instructions
- Register File Cache (RFC)
  - VGPR cache, strict LRU replacement
  - Stores VALU results
  - Forwarding paths to OB, VALU, Vector Memory units

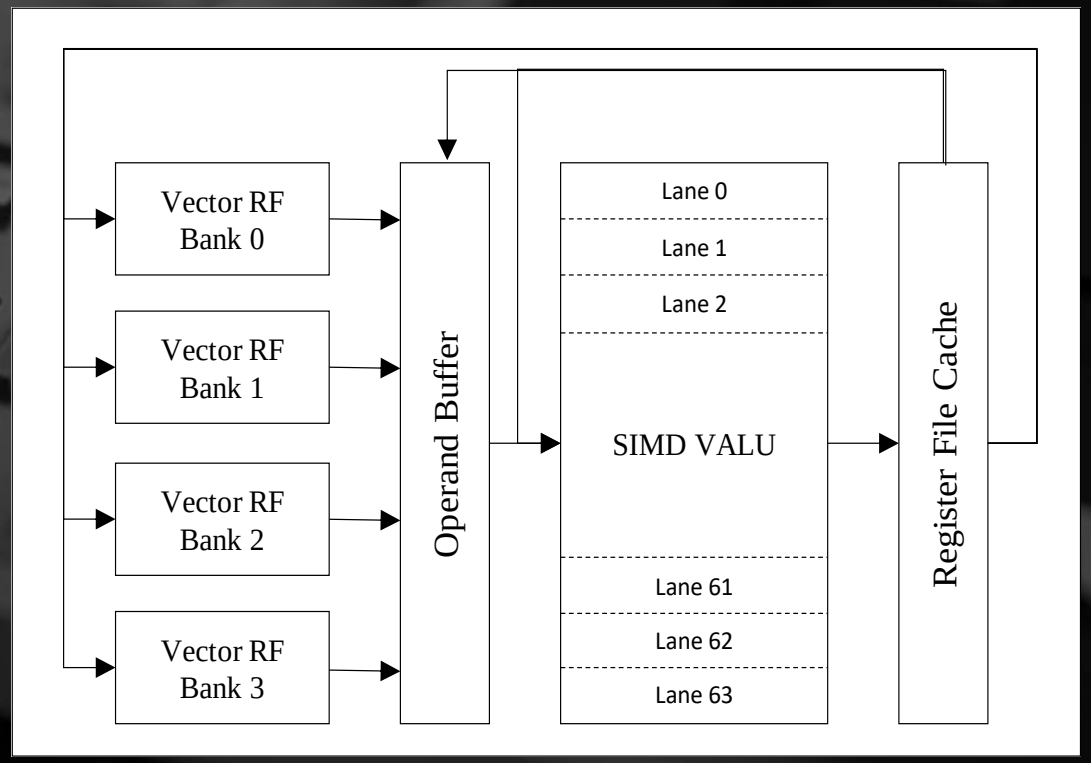


Figure 2. Vector Register File Subsystem Architecture

gem5 Model and Simulation

GEM5 MODEL AND SIMULATION

- Execution driven, cycle-level microarchitecture simulator
- System Call Emulation (SE) mode
- Faithfully implement CU as described in previous slides, capable of executing AMD GCN3 ISA
- Unmodified, publically available ROCm-1.1 software stack
- Emulated kernel driver functionality

- Wavefront and Instruction Readiness
  - Check every wavefront each cycle to determine if it has a ready instruction
- Instruction Dispatch and Execution
  - Select one wave per execution resource per cycle from list of ready waves
  - Initiate register reads
    - VALU operations sent to Operand Buffer
- Single CPU, single CU system

BENCHMARKS

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

Table 1. Benchmarks

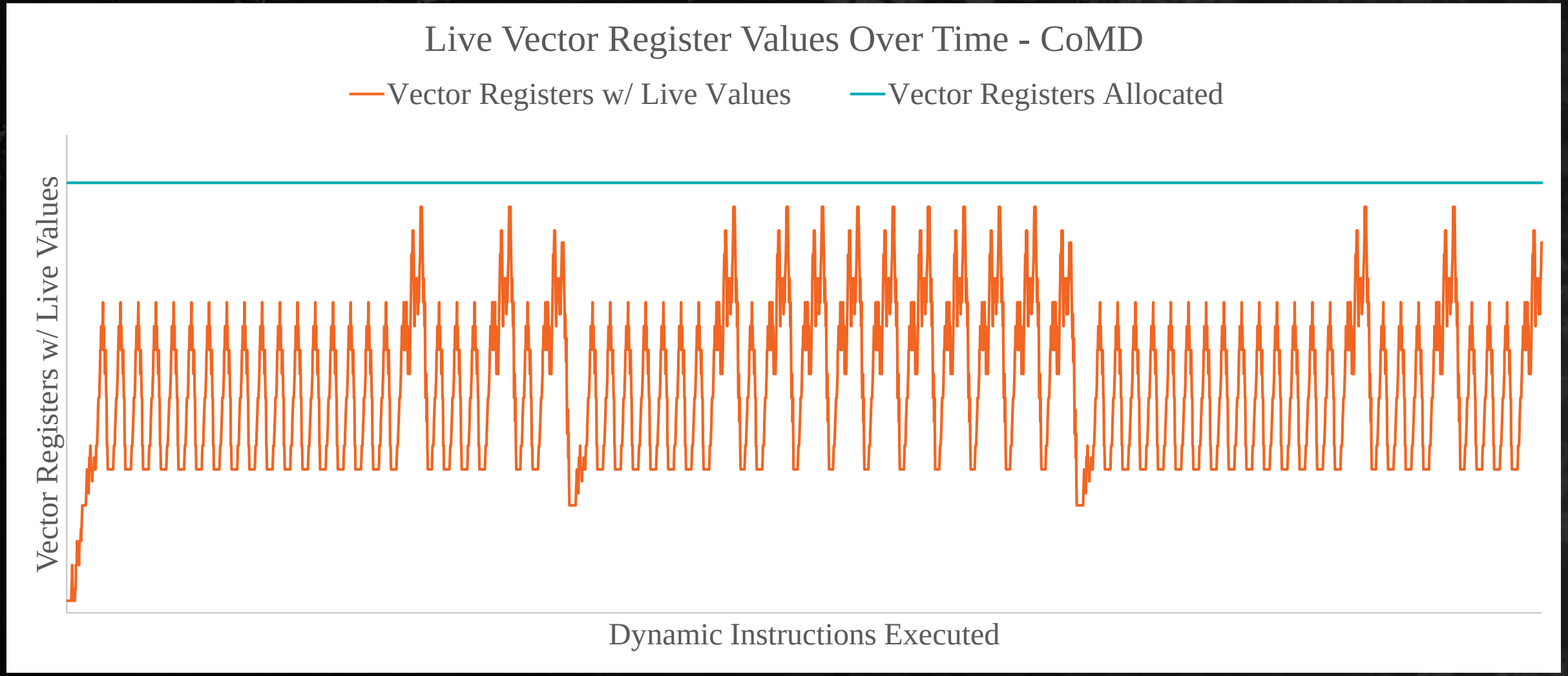
VGPR Usage

VGPR USAGE

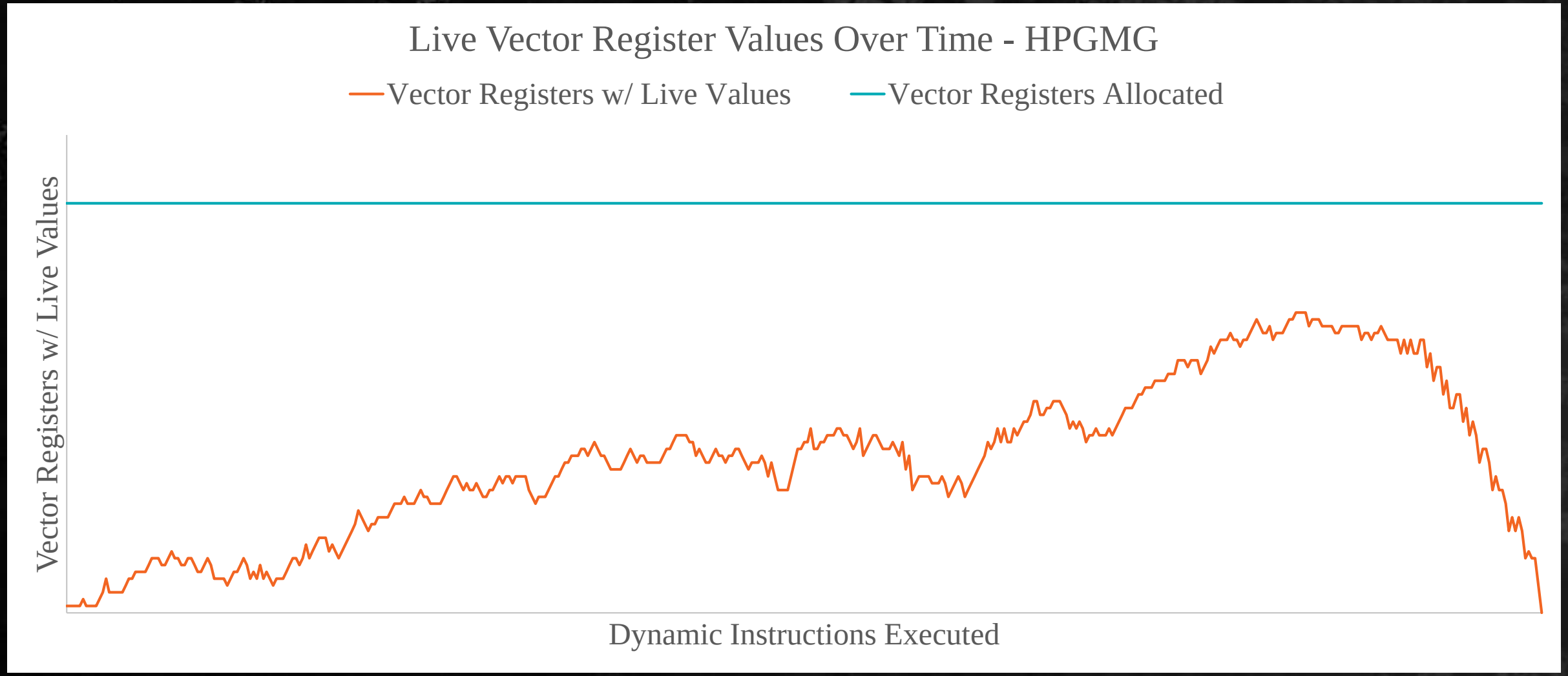
AMD

- Jeon et al. claim many registers contain no live value for significant portion of execution
- Gebhart et al. claim most register values produced are read only once, and the read occurs within a small distance of the value being produced
- **Question**: Do these observations hold for our simulated architecture and GCN3 code?
  - **Yes!**
- **Analysis**: Dynamic instrumentation of vector register usage in gem5 simulation to track read after write distance and number of reads per write per value produced

VGPR USAGE


Live Vector Register Values Over Time - CoMD
Vector Registers w/ Live Values
Vector Registers Allocated
Vector Registers w/ Live Values
Dynamic Instructions Executed

VGPR USAGE


Live Vector Register Values Over Time - HPGMG
Vector Registers w/ Live Values
Vector Registers Allocated
Vector Registers w/ Live Values
Dynamic Instructions Executed

VGPR USAGE

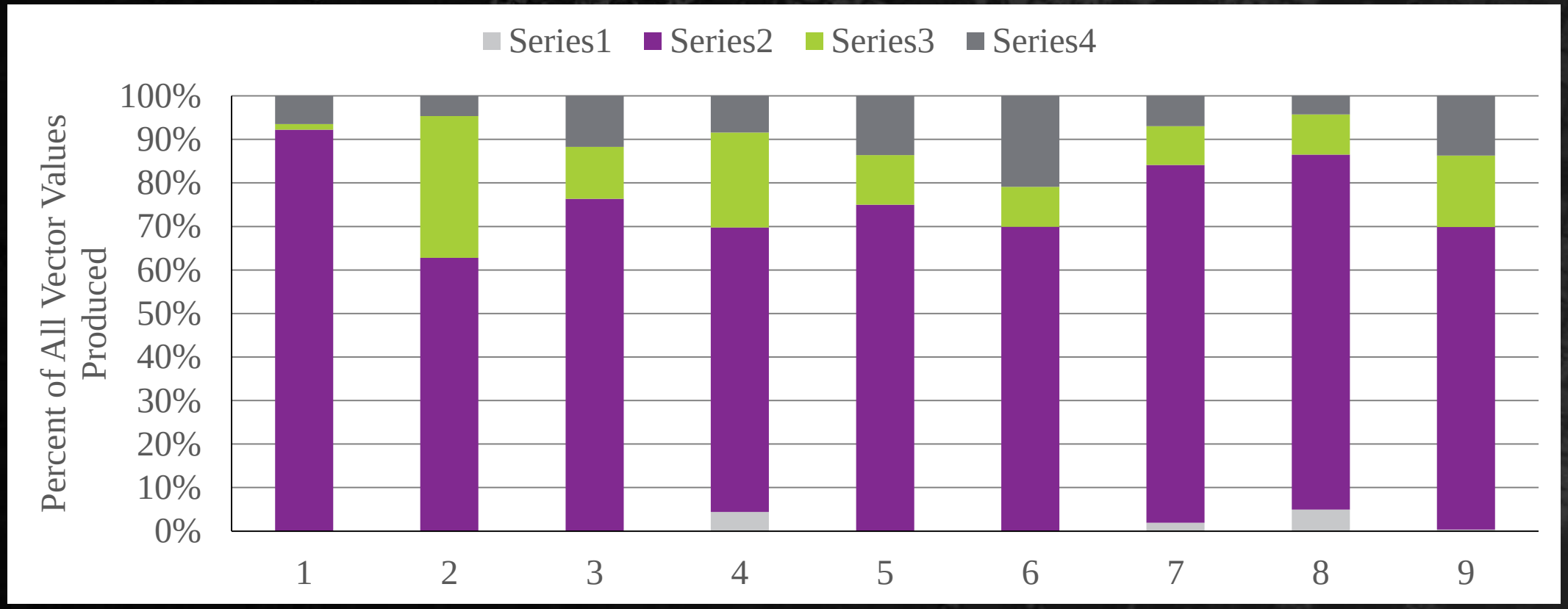


Figure 3. Number of reads per vector register value

VGPR USAGE

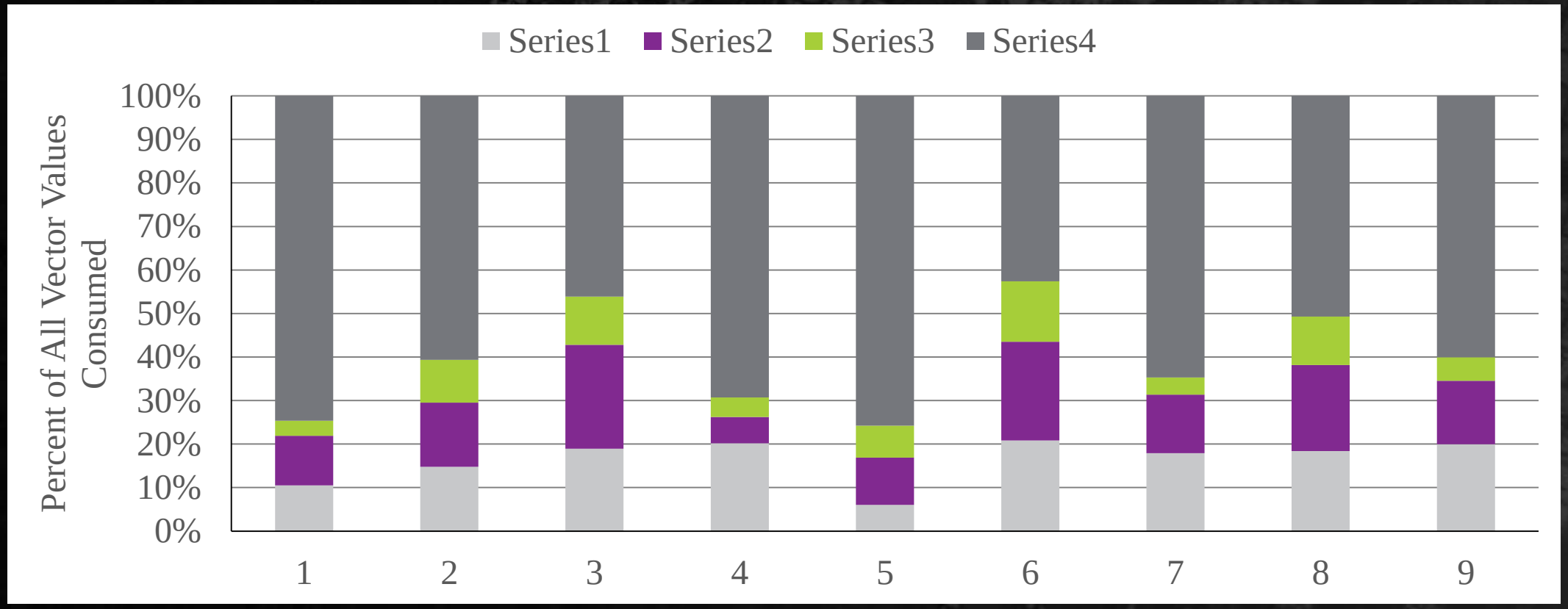


Figure 4. Producer to consumer distance


AMD
Operand Buffering and
Register File Caching

OB AND RFC EFFECTIVENESS – VALU VGPR READS SAVED

OVERVIEW

- **Goal**: evaluate effectiveness of the Register File Cache at reducing the number of register file reads performed by the Operand Buffer
- **Experiment**: sweep RFC size from 2 to 512 entries per SIMD/VRF and examine number of VALU VGPR reads saved and performance impact
- **Conclusion**: Register File Cache can reduce between 19% and 41% of VALU VGPR reads at size 8 with stable performance, and greater fraction of reads at larger sizes

OB AND RFC EFFECTIVENESS – VALU VGPR READS SAVED

NUMBER OF VALU VGPR READS SAVED BY RFC FORWARDING

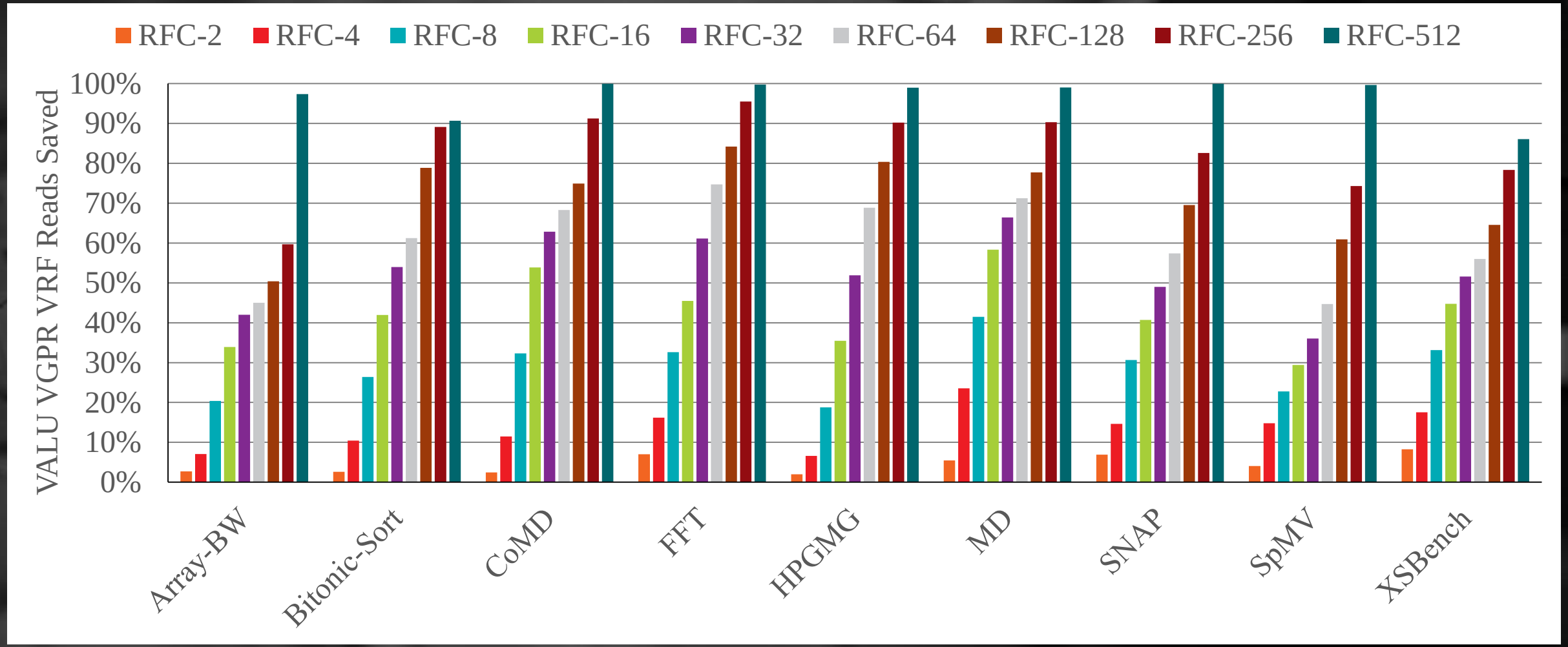


Figure 5. Number of VALU VGPR reads saved by RFC forwarding paths

OB AND RFC EFFECTIVENESS

OVERVIEW

- **Goal**: evaluate effectiveness of the Operand Buffer and Register File Cache structures in hiding register file bank conflicts

- **Conclusions**: Operand Buffer and Register File Cache are highly effective at hiding bank conflict stalls and latency penalties

OB AND RFC EFFECTIVENESS

EXPERIMENTAL SETUP

- **Experiment**: compare performance of baseline architecture to one that has no register file bank access conflicts
- Upper Bound Study
- Conflict Free configuration places each VGPR into its own VRF bank

OB AND RFC EFFECTIVENESS

PERFORMANCE ESTIMATION

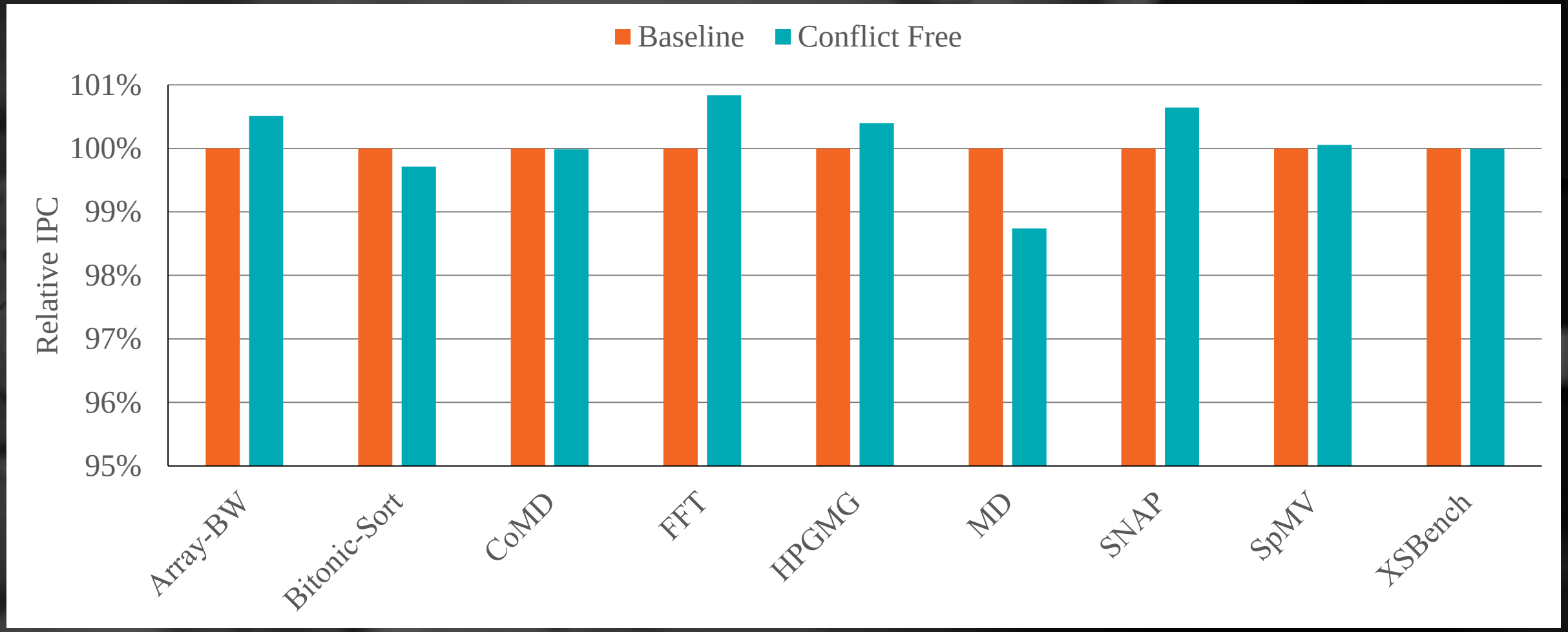


Figure 6. Relative performance of Conflict Free configuration compared to baseline (Note y-axis begins at 95%)

Dispatch Limits and Wave Level Parallelism

DISPATCH LIMITS AND WLP

OVERVIEW

- **Goal**: evaluate impact on WLP and performance of providing twice as many VGPRs per SIMD

- **Conclusions**: Providing additional VGPRs allows more workgroups/wavefronts to be dispatched per CU, but may not always provide a performance benefit

DISPATCH LIMITS AND WLP

RESULTS

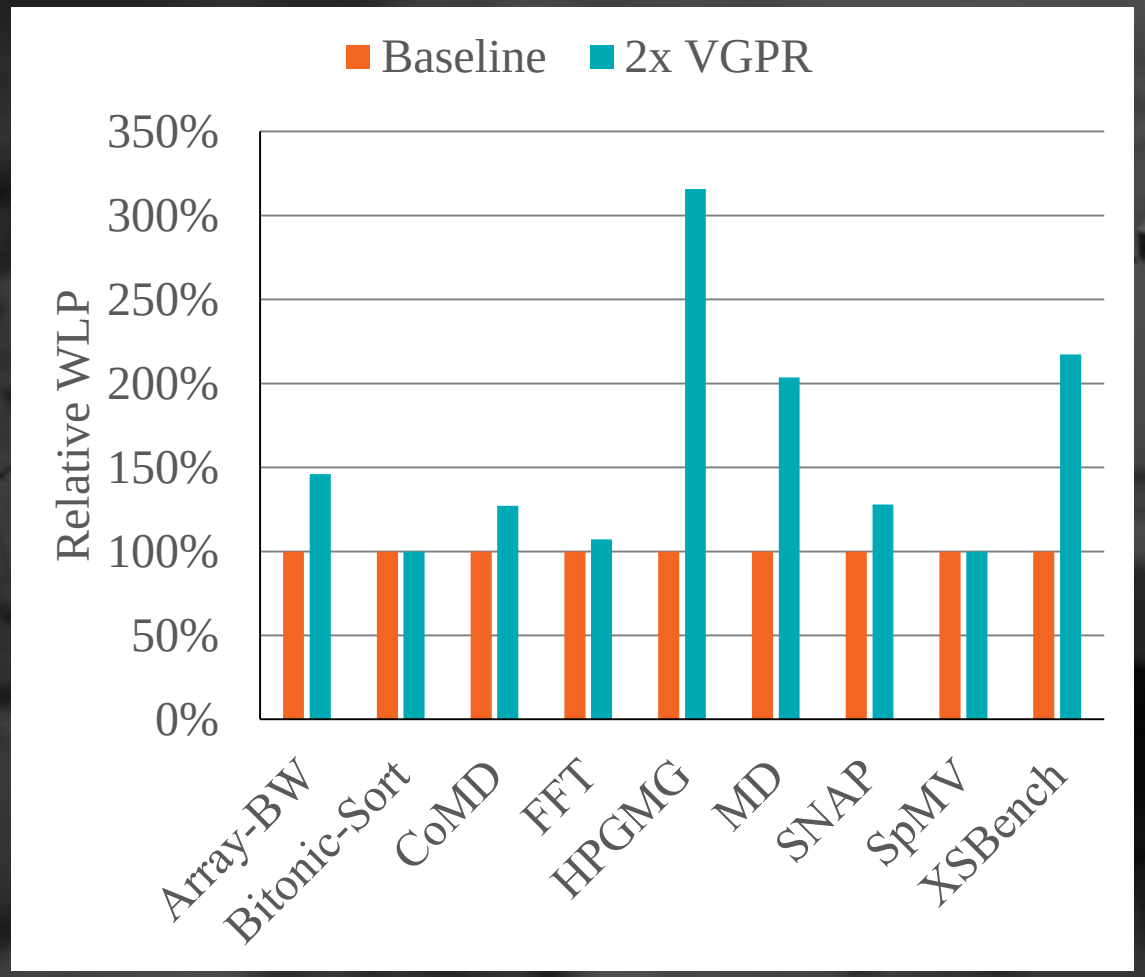


Figure 9. Relative WLP with 2x VGPR per SIMD

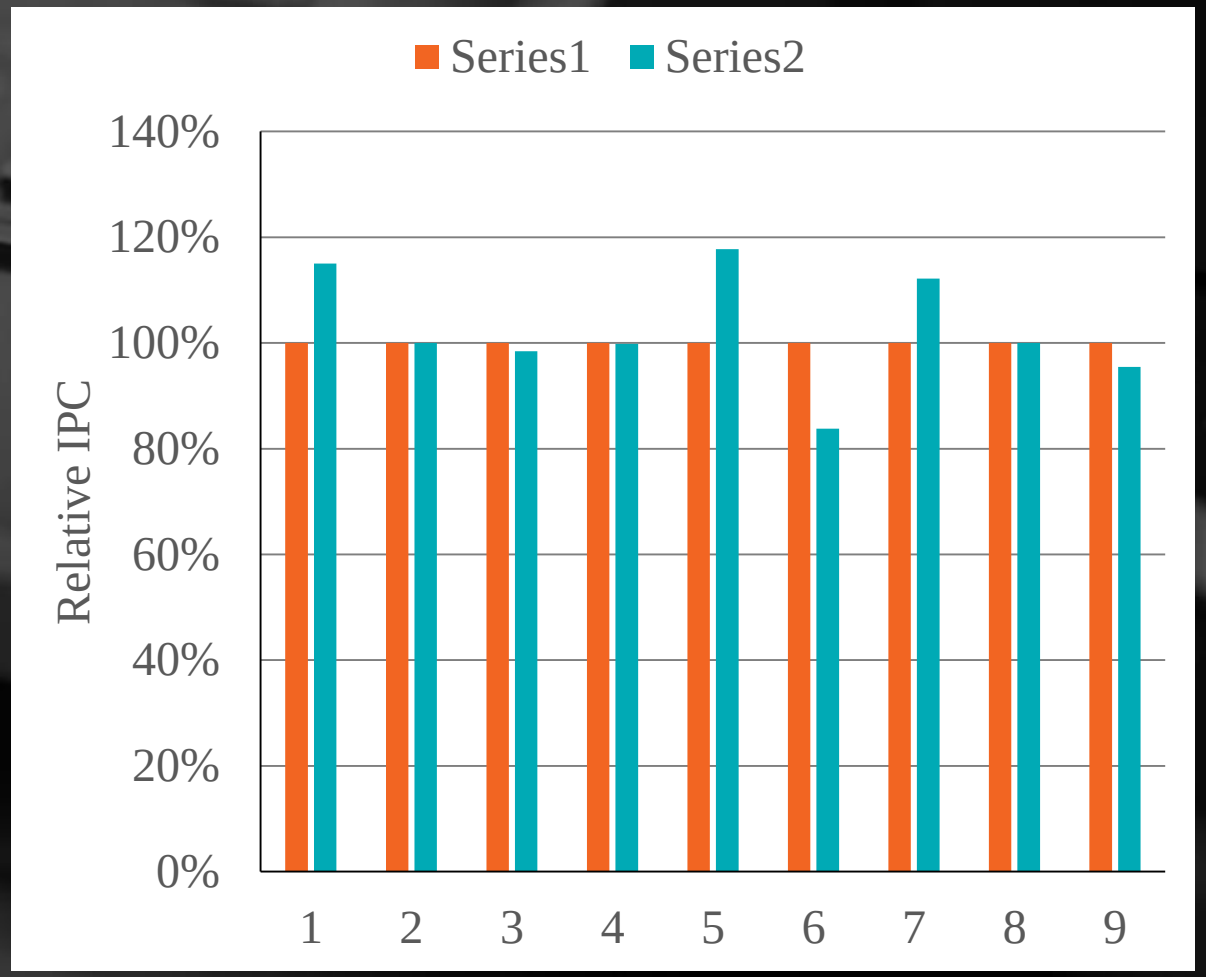


Figure 8. Relative performance with 2x VGPR per SIMD

DISPATCH LIMITS AND WLP

HEAD TAIL LATENCY

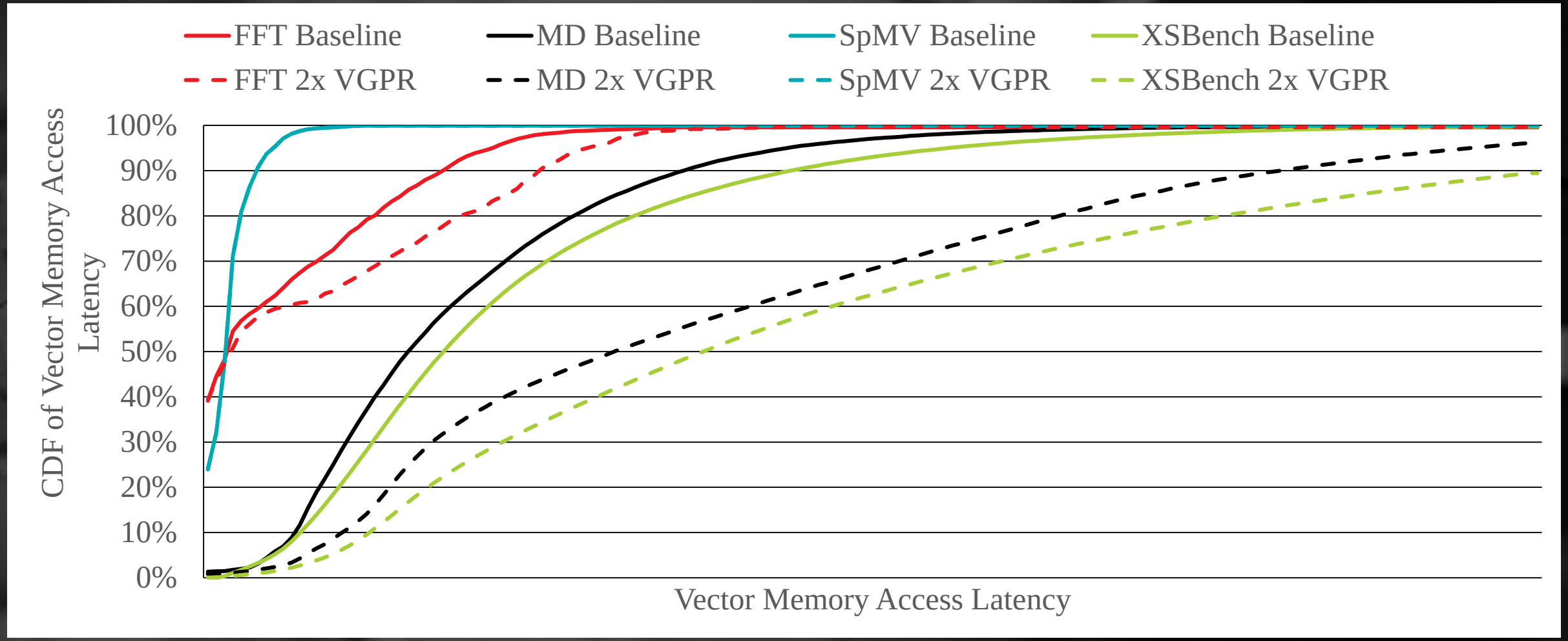


Figure 10. CDF of Vector Memory Access Latency – Selected Benchmarks

Conclusion

CONCLUSION

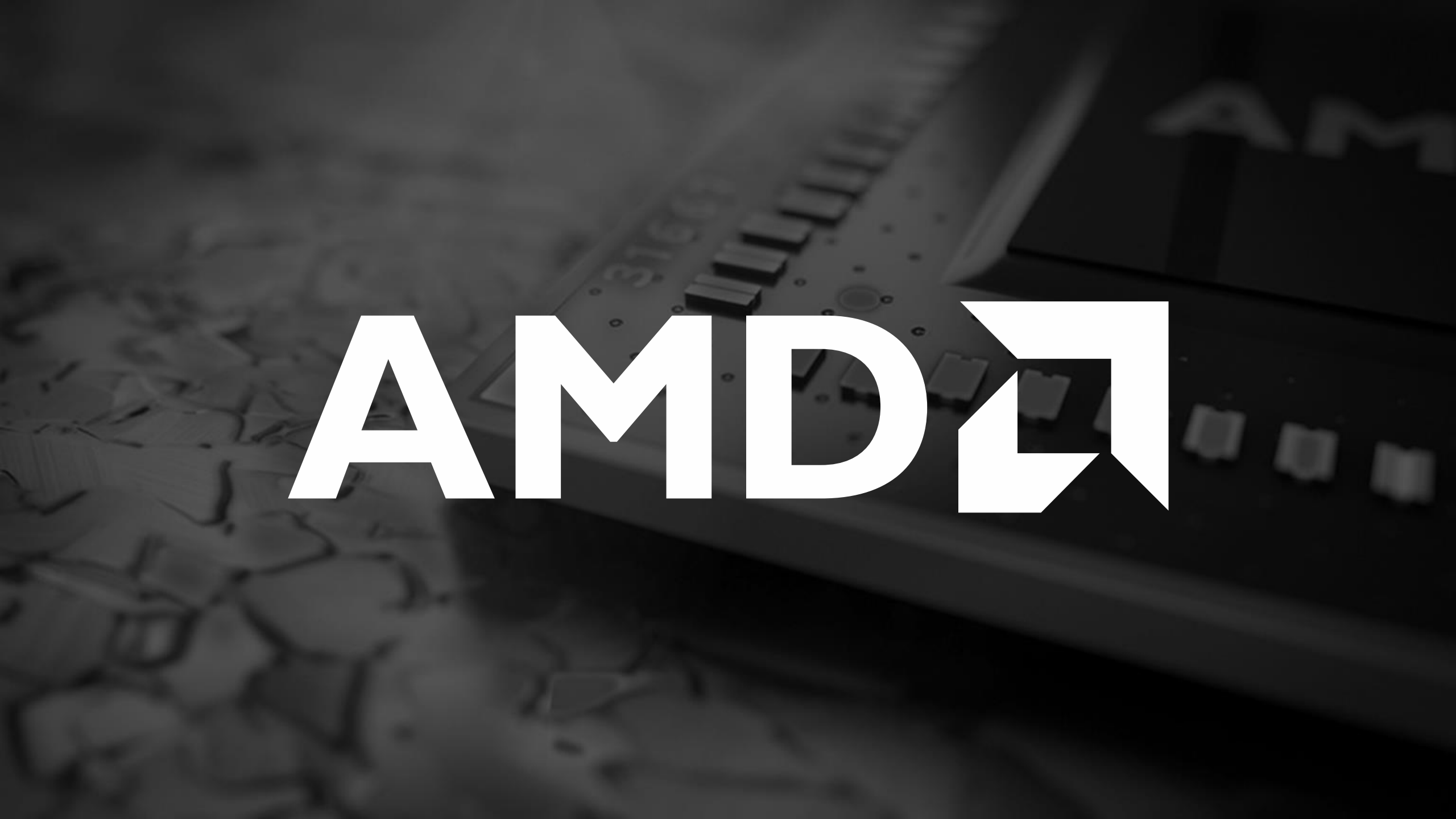
- Replicate studies in prior works on vector register usage. Values have short RAW distance and are read only a handful of times
- Operand Buffers and Register File Caches can hide penalties from register file bank conflicts
- Increased wave level parallelism (WLP) does not guarantee better performance

FUTURE RESEARCH DIRECTIONS

AMD

- Non-graphics influenced general purpose compute accelerator architectures
  - Architecture
  - Programming model

- Smarter strategies for handling memory divergence
  - Memory bandwidth provided is adequate for compute bound applications
  - But, it limits memory divergent and irregular applications


AMD

DISCLAIMER & ATTRIBUTION

AMD

Disclaimer

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, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.