MEASURING AND MODELING ON-CHIP INTERCONNECT POWER ON REAL HARDWARE

VIGNESH ADHINARAYANAN, INDRANI PAUL, JOSEPH L. GREATHOUSE, WEI HUANG, ASHUTOSH PATTNAIK, WU-CHUN FENG

Power and Energy are First-Class Design Constraints

- Power and energy have become a first-class design constraint in all areas of computing
  - Phones and laptops must optimize for energy as they are battery operated
  - Desktops require loud cooling solutions
  - Supercomputers and other large data centers are constrained by power

Power and Energy are First-Class Design Constraints

AMD

- Power and energy have become a first-class design constraint in all areas of computing
  - Phones and laptops must optimize for energy as they are battery operated
  - Desktops require loud cooling solutions
  - Supercomputers and other large data centers are constrained by power

Modified from Image by Vernon Chan CC BY 2.0

VirginiaTech Invent the Future PennState

Power and Energy are First-Class Design Constraints

- Power and energy have become a first-class design constraint in all areas of computing
  - Phones and laptops must optimize for energy as they are battery operated
  - Desktops require loud cooling solutions
  - Supercomputers and other large data centers are constrained by power

Modified from Image by Vernon Chan CC BY 2.0

Image by Glenn Batuyong CC BY 2.0

Power and Energy are First-Class Design Constraints

AMD

- Power and energy have become a first-class design constraint in all areas of computing
  - Phones and laptops must optimize for energy as they are battery operated
  - Desktops require loud cooling solutions
  - Supercomputers and other large data centers are constrained by power

Modified from Image by Vernon Chan CC BY 2.0

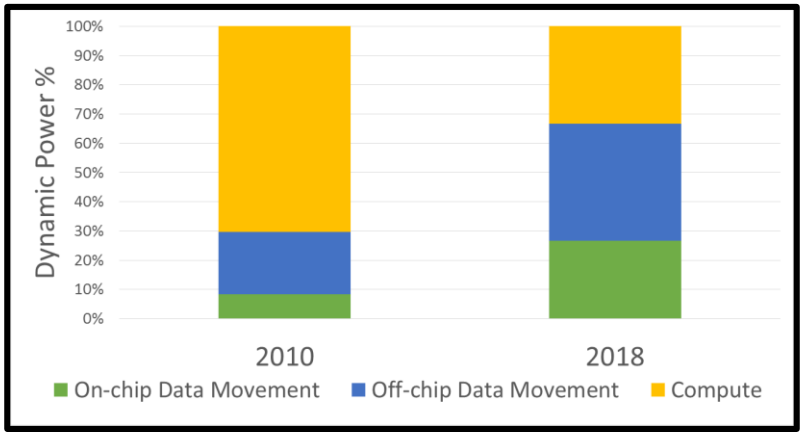
Image by Glenn Batuyong CC BY 2.0

Image by Patrick Strandberg CC BY-SA 2.0

VirginiaTech Invent the Future PennState

Data Movement: A Major Source of Power Consumption

- Data movement through the memory hierarchy thought to be a major source of power consumption



Based on: Shalf et al., "Exascale Computing Technology Challenges," VECPAR 2010

- Measurements on real system lacking
  - Current estimates based on simulation studies
  - Real-world measurements are coarse-grained and do not give break down for data movement

This Talk in One Sentence

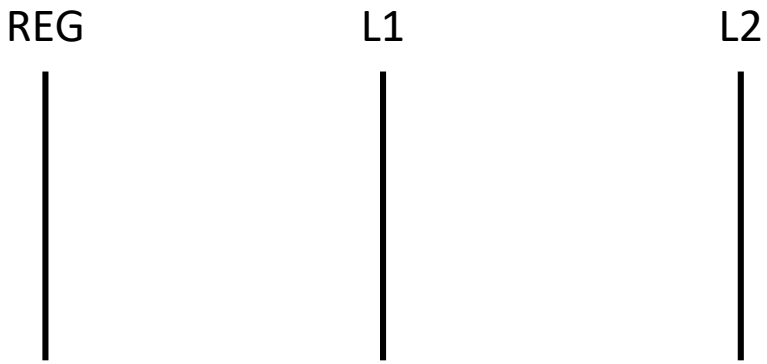
A new methodology to measure
data-movement power on real hardware

State-of-the-art Measurement Approach

State-of-the-art Measurement Approach

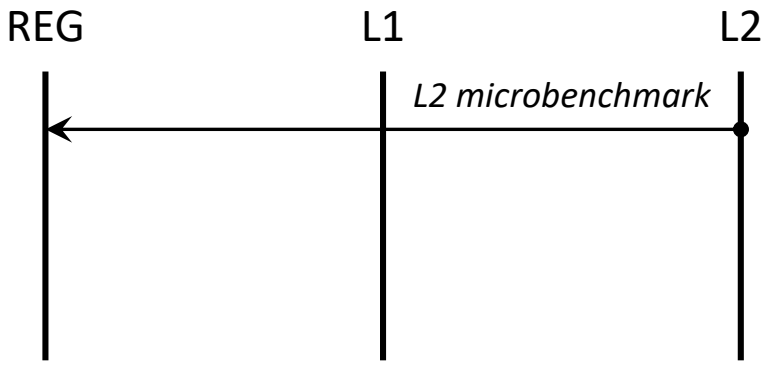
G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

State-of-the-art Measurement Approach



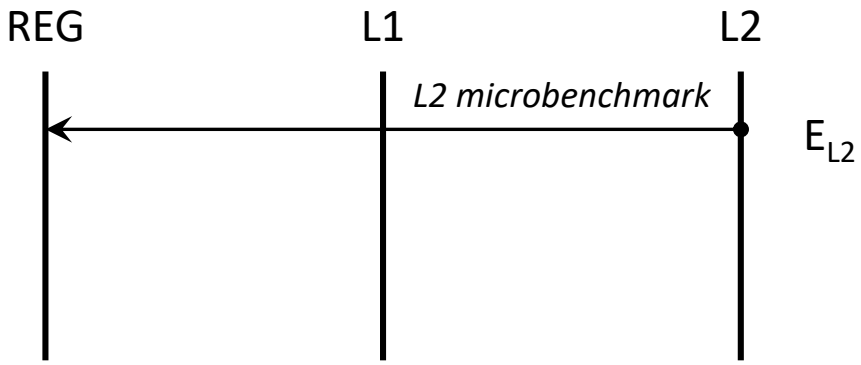
G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

State-of-the-art Measurement Approach



G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

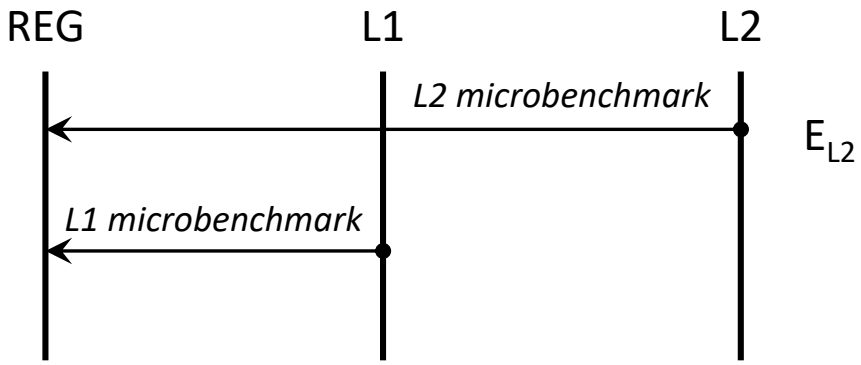
State-of-the-art Measurement Approach



G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

- E_{L2} = Energy consumed by L2 microbenchmark

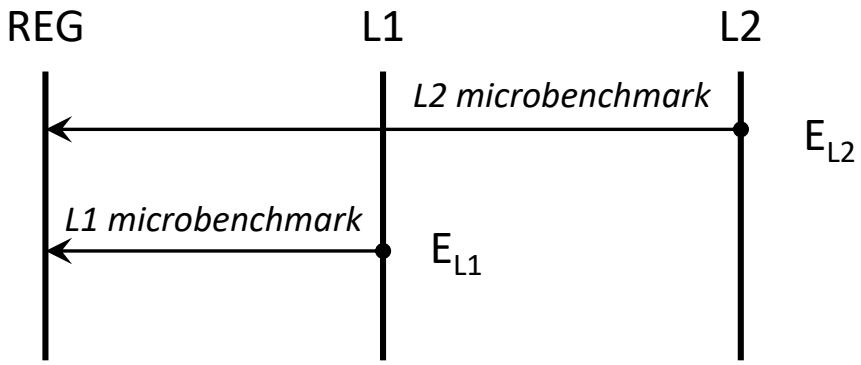
State-of-the-art Measurement Approach



G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

- E_{L2} = Energy consumed by L2 microbenchmark

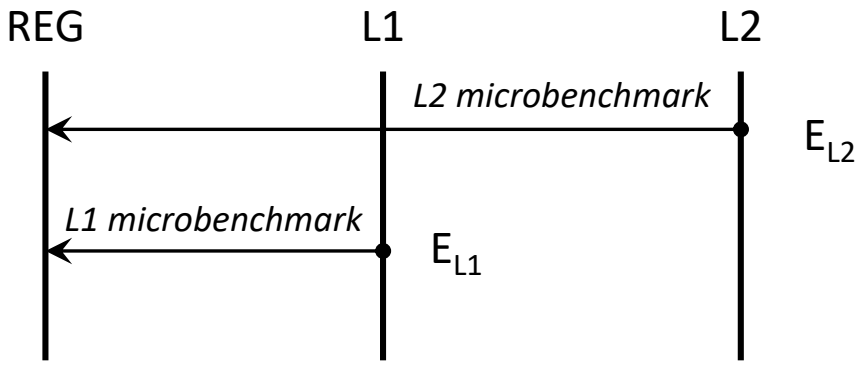
State-of-the-art Measurement Approach



G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

- E_{L2} = Energy consumed by L2 microbenchmark
- E_{L1} = Energy consumed by L1 microbenchmark

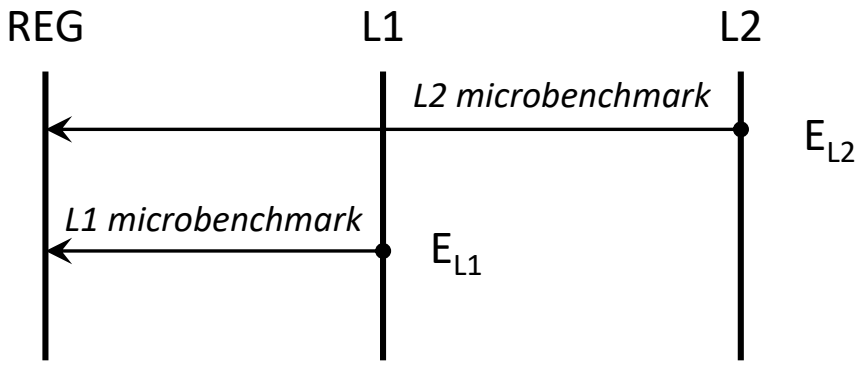
State-of-the-art Measurement Approach



G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

- E_{L2} = Energy consumed by L2 microbenchmark
- E_{L1} = Energy consumed by L1 microbenchmark
- Energy cost of moving data from L2 to L1 = $\mathbf{E_{L2} - E_{L1}}$

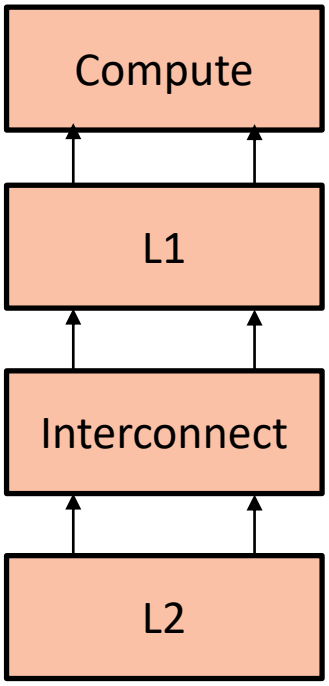
State-of-the-art Measurement Approach



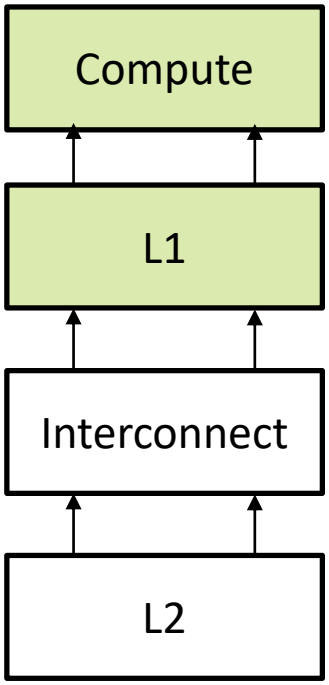
G. Kestor et al., "Quantifying the energy cost of data movement in scientific applications," IISWC 2013

- E_{L2} = Energy consumed by L2 microbenchmark
- E_{L1} = Energy consumed by L1 microbenchmark
- Energy cost of moving data from L2 to L1 = $\mathbf{E_{L2} - E_{L1}}$
- **Issue:** Over-estimation of data-movement energy

State-of-the-art Measurement Approach: Limitation



L2 microbenchmark

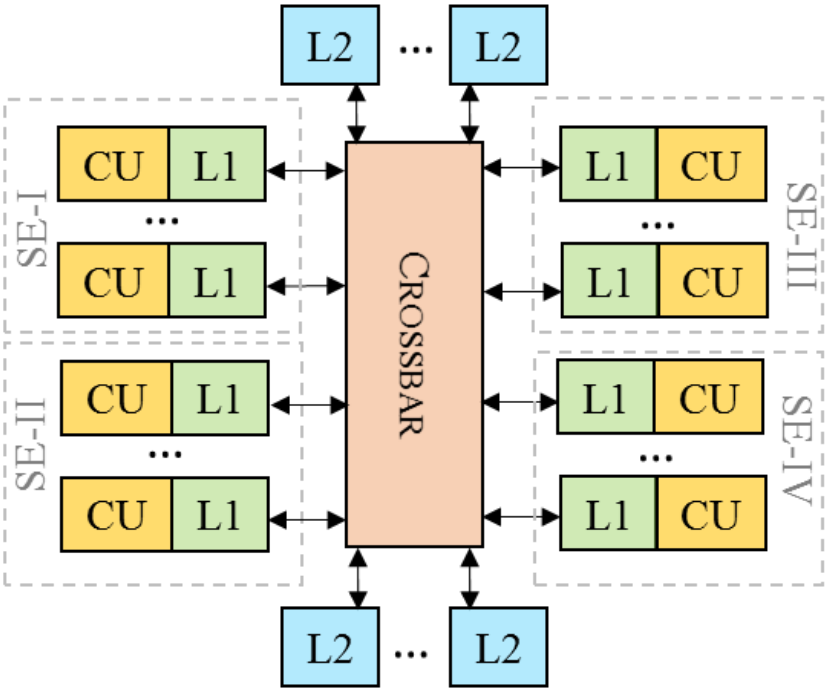


L1 microbenchmark

Issue: Data-movement power also includes L2-access power

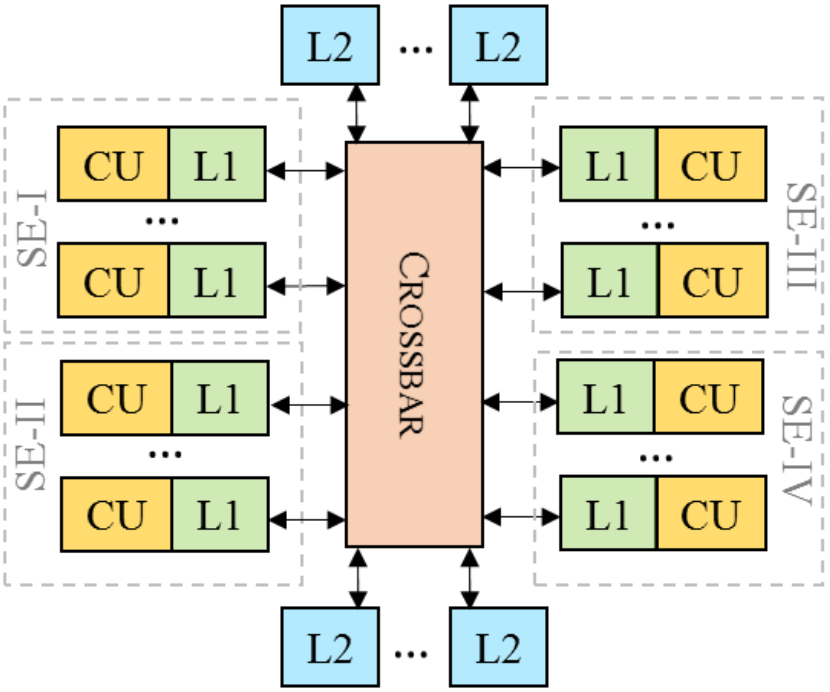
Our Proposed Approach

A Bit of Background ...



Representative block diagram of AMD FirePro™ W9100 GPU
(Previously Code-named "Hawaii")

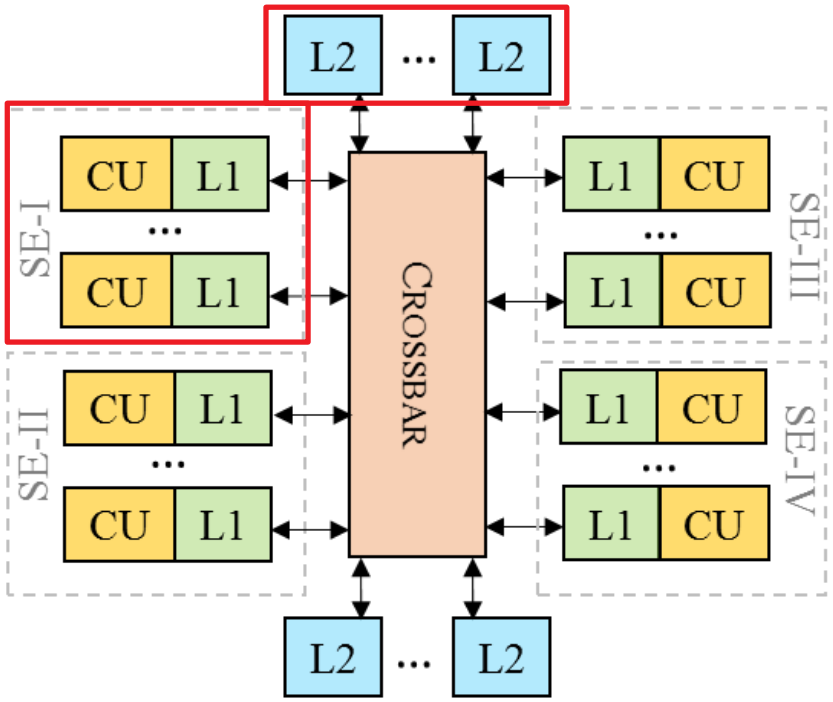
A Bit of Background ...



Representative block diagram of AMD FirePro™ W9100 GPU
(Previously Code-named "Hawaii")

Distance between shader engines and L2 banks differ

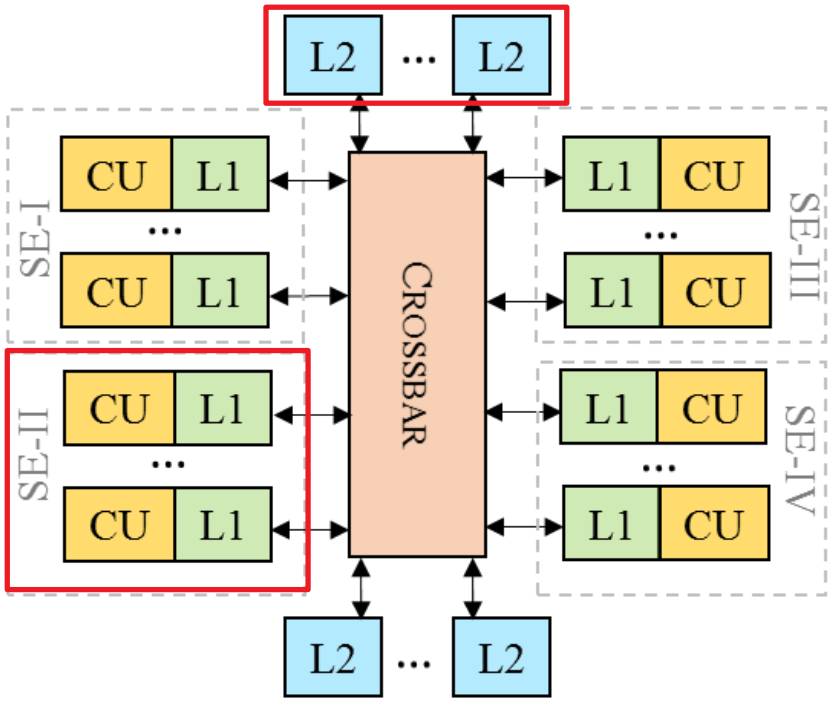
A Bit of Background ...



Representative block diagram of AMD FirePro™ W9100 GPU
(Previously Code-named "Hawaii")

Distance between shader engines and L2 banks differ

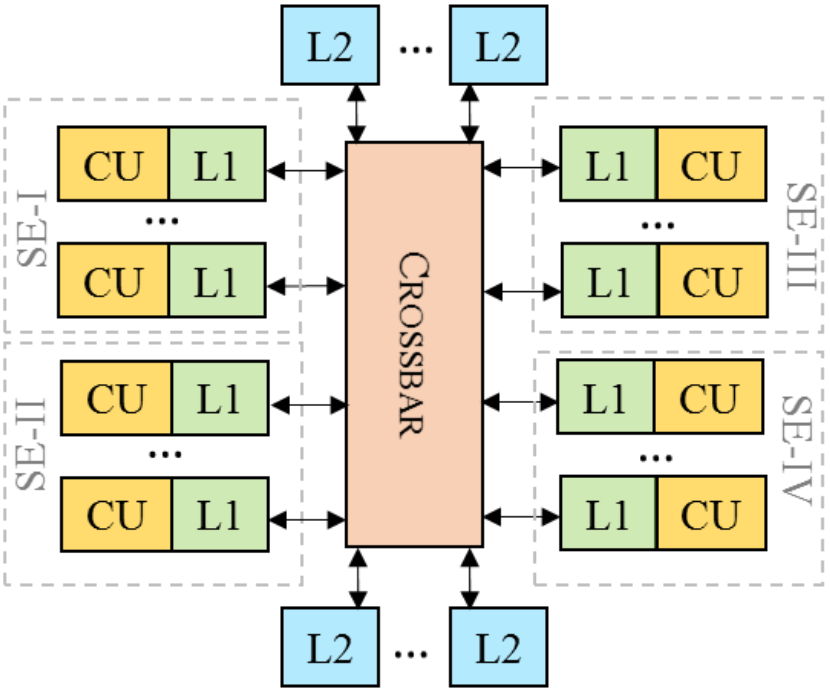
A Bit of Background ...



Representative block diagram of AMD FirePro™ W9100 GPU
(Previously Code-named "Hawaii")

Distance between shader engines and L2 banks differ

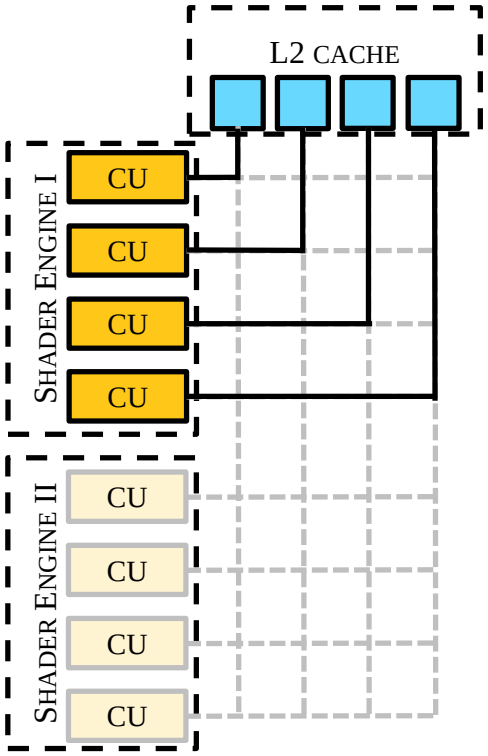
A Bit of Background ...



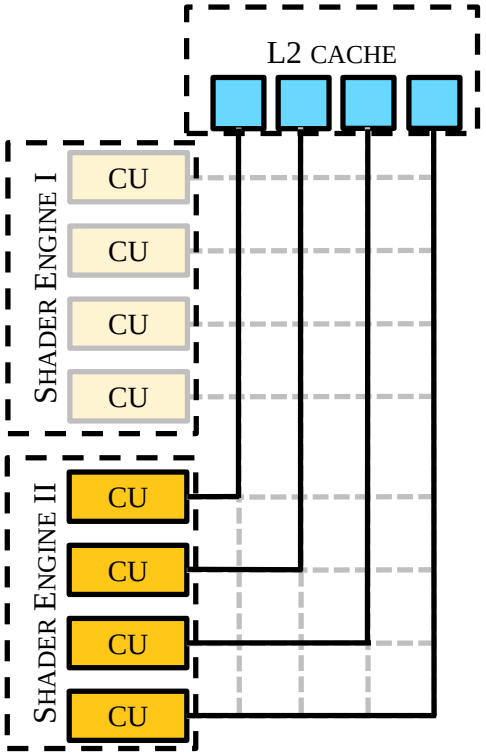
Representative block diagram of AMD FirePro™ W9100 GPU
(Previously Code-named "Hawaii")

Physical distance traversed by data thought to affect data movement power

Our Proposed Approach



Short-path microbenchmark



Long-path microbenchmark

Design microbenchmarks based on data-movement distance to properly isolate the interconnect

CHALLENGES

CHALLENGES

CHALLENGES

AMD

OpenCL™ lacks native support to pin threads to programmer-specified compute units

VirginiaTech Invent the Future PennState

CHALLENGES

OpenCL™ lacks native support to pin threads to programmer-specified compute units

Temperature of device during tests will affect the power consumption

CHALLENGES

AMD

OpenCL™ lacks native support to pin threads to programmer-specified compute units

Temperature of device during tests will affect the power consumption

Power difference between the two sets of microbenchmarks can be hard to observe

CHALLENGES

CHALLENGES

AMD

Only a small difference between used-L2 and L1 cache size which can make it challenging to write L2-only microbenchmarks

VirginiaTech Invent the Future PennState

CHALLENGES

AMD

Only a small difference between used-L2 and L1 cache size which can make it challenging to write L2-only microbenchmarks

Ensuring that the same amount of work is done by the two microbenchmarks can be challenging due to NUCA effects

ADDRESSING THE CHALLENGES

AMD

OpenCL™ lacks native support to pin threads to programmer-specified compute units

Temperature of device during tests will affect the power consumption

Power difference between the two sets of microbenchmarks can be hard to observe

VirginiaTech Invent the Future PennState

PINNING THREADS TO CORES WITH OPENCL™

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int wid = get_group_id(0);
          if (wid >= 0 && wid <= 10) {
                    // Read data from L2
          }
}
```

(a) Initial OpenCL code snippet

```
s_min_u32      s0, s0, 0x0000ffff  // 000000000014: 8380FF00 0000FFFF
s_mul_i32      s0, s16, s0         // 00000000001C: 93000010
s_add_i32      s0, s0, s1          // 000000000020: 81000100
v_add_i32      v0, vcc, s0, v0     // 000000000024: 4A000000
s_add_i32      s0, s16, s2         // 000000000028: 81000210
s_cmp_gt_i32   s0, -1              // 00000000002C: BF02C100
s_cbranch_scc0 label_0011          // 000000000030: BF840004
```

(b) Equivalent assembly code

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 10 02 00 81
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(c) Equivalent binary (in hex)

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 04 32 00 B9
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(d) Modified binary (in hex)

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int cu_id = get_cu_id(0);
          if (cu_id >= 0 && cu_id <= 10) {
                    // Read data from L2
          }
}
```

(e) Equivalent OpenCL code

Can be accomplished with some binary hacking

PINNING THREADS TO CORES WITH OPENCL™

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
        int gid = get_global_id(0);
        int wid = get_group_id(0);
        if (wid >= 0 && wid <= 10) {
                // Read data from L2
        }
}
```

(a) Initial OpenCL code snippet

```
s_min_u32      s0, s0, 0x0000ffff  // 000000000014: 8380FF00 0000FFFF
s_mul_i32      s0, s16, s0         // 00000000001C: 93000010
s_add_i32      s0, s0, s1          // 000000000020: 81000100
v_add_i32      v0, vcc, s0, v0     // 000000000024: 4A000000
s_add_i32      s0, s16, s2         // 000000000028: 81000210
s_cmp_gt_i32   s0, -1              // 00000000002C: BF02C100
s_cbranch_scc0 label_0011          // 000000000030: BF840004
```

(b) Equivalent assembly code

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 10 02 00 81
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(c) Equivalent binary (in hex)

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 04 32 00 B9
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(d) Modified binary (in hex)

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
        int gid = get_global_id(0);
        int cu_id = get_cu_id(0);
        if (cu_id >= 0 && cu_id <= 10) {
                // Read data from L2
        }
}
```

(e) Equivalent OpenCL code

1. Use workgroup ID as a placeholder for CU ID

PINNING THREADS TO CORES WITH OPENCL™

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int wid = get_group_id(0);
          if (wid >= 0 && wid <= 10) {
                    // Read data from L2
          }
}
```

(a) Initial OpenCL code snippet

```
s_min_u32      s0, s0, 0x0000ffff  // 000000000014: 8380FF00 0000FFFF
s_mul_i32      s0, s16, s0         // 00000000001C: 93000010
s_add_i32      s0, s0, s1          // 000000000020: 81000100
v_add_i32      v0, vcc, s0, v0     // 000000000024: 4A000000
s_add_i32      s0, s16, s2         // 000000000028: 81000210
s_cmp_gt_i32   s0, -1              // 00000000002C: BF02C100
s_cbranch_scc0 label_0011          // 000000000030: BF840004
```

(b) Equivalent assembly code

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 10 02 00 81
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(c) Equivalent binary (in hex)

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 04 32 00 B9
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(d) Modified binary (in hex)

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int cu_id = get_cu_id(0);
          if (cu_id >= 0 && cu_id <= 10) {
                    // Read data from L2
          }
}
```

(e) Equivalent OpenCL code

2. Identify instruction that writes workgroup ID to the register that gets checked in the conditional

PINNING THREADS TO CORES WITH OPENCL™

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int wid = get_group_id(0);
          if (wid >= 0 && wid <= 10) {
                    // Read data from L2
          }
}
```

(a) Initial OpenCL code snippet

```
s_min_u32      s0, s0, 0x0000ffff  // 000000000014: 8380FF00 0000FFFF
s_mul_i32      s0, s16, s0         // 00000000001C: 93000010
s_add_i32      s0, s0, s1          // 000000000020: 81000100
v_add_i32      v0, vcc, s0, v0     // 000000000024: 4A000000
s_add_i32      s0, s16, s2         // 000000000028: 81000210
s_cmp_gt_i32   s0, -1              // 00000000002C: BF02C100
s_cbranch_scc0 label_0011          // 000000000030: BF840004
```

(b) Equivalent assembly code

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 10 02 00 81
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(c) Equivalent binary (in hex)

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 04 32 00 B9
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(d) Modified binary (in hex)

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int cu_id = get_cu_id(0);
          if (cu_id >= 0 && cu_id <= 10) {
                    // Read data from L2
          }
}
```

(e) Equivalent OpenCL code

3. Identify instruction that writes workgroup ID to the register that gets checked in the conditional in binary

PINNING THREADS TO CORES WITH OPENCL™

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int wid = get_group_id(0);
          if (wid >= 0 && wid <= 10) {
                    // Read data from L2
          }
}
```

(a) Initial OpenCL code snippet

```
s_min_u32      s0, s0, 0x0000ffff  // 000000000014: 8380FF00 0000FFFF
s_mul_i32      s0, s16, s0         // 00000000001C: 93000010
s_add_i32      s0, s0, s1          // 000000000020: 81000100
v_add_i32      v0, vcc, s0, v0     // 000000000024: 4A000000
s_add_i32      s0, s16, s2         // 000000000028: 81000210
s_cmp_gt_i32   s0, -1              // 00000000002C: BF02C100
s_cbranch_scc0 label_0011          // 000000000030: BF840004
```

(b) Equivalent assembly code

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 10 02 00 81
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(c) Equivalent binary (in hex)

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 04 32 00 B9
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(d) Modified binary (in hex)

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int cu_id = get_cu_id(0);
          if (cu_id >= 0 && cu_id <= 10) {
                    // Read data from L2
          }
}
```

(e) Equivalent OpenCL code

4. Replace workgroup ID with CU ID (read from read-only HW register) in the correct register in the binary

PINNING THREADS TO CORES WITH OPENCL™

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int wid = get_group_id(0);
          if (wid >= 0 && wid <= 10) {
                    // Read data from L2
          }
}
```

(a) Initial OpenCL code snippet

```
s_min_u32      s0, s0, 0x0000ffff  // 000000000014: 8380FF00 0000FFFF
s_mul_i32      s0, s16, s0         // 00000000001C: 93000010
s_add_i32      s0, s0, s1          // 000000000020: 81000100
v_add_i32      v0, vcc, s0, v0     // 000000000024: 4A000000
s_add_i32      s0, s16, s2         // 000000000028: 81000210
s_cmp_gt_i32   s0, -1              // 00000000002C: BF02C100
s_cbranch_scc0  label_0011         // 000000000030: BF840004
```

(b) Equivalent assembly code

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 10 02 00 81
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(c) Equivalent binary (in hex)

```
00 FF 80 83 FF FF 00 00
10 00 00 93 00 01 00 81
00 00 00 4A 04 32 00 B9
00 C1 02 BF 04 00 84 BF
00 FF 04 BF 81 00 00 00
C1 80 01 85 01 00 82 BF
```

(d) Modified binary (in hex)

```
__kernel void l2_read( __global float *data,
                       __global float *output) {
          int gid = get_global_id(0);
          int cu_id = get_cu_id(0);
          if (cu_id >= 0 && cu_id <= 10) {
                    // Read data from L2
          }
}
```

(e) Equivalent OpenCL code

5. Get functionally equivalent OpenCL snippet

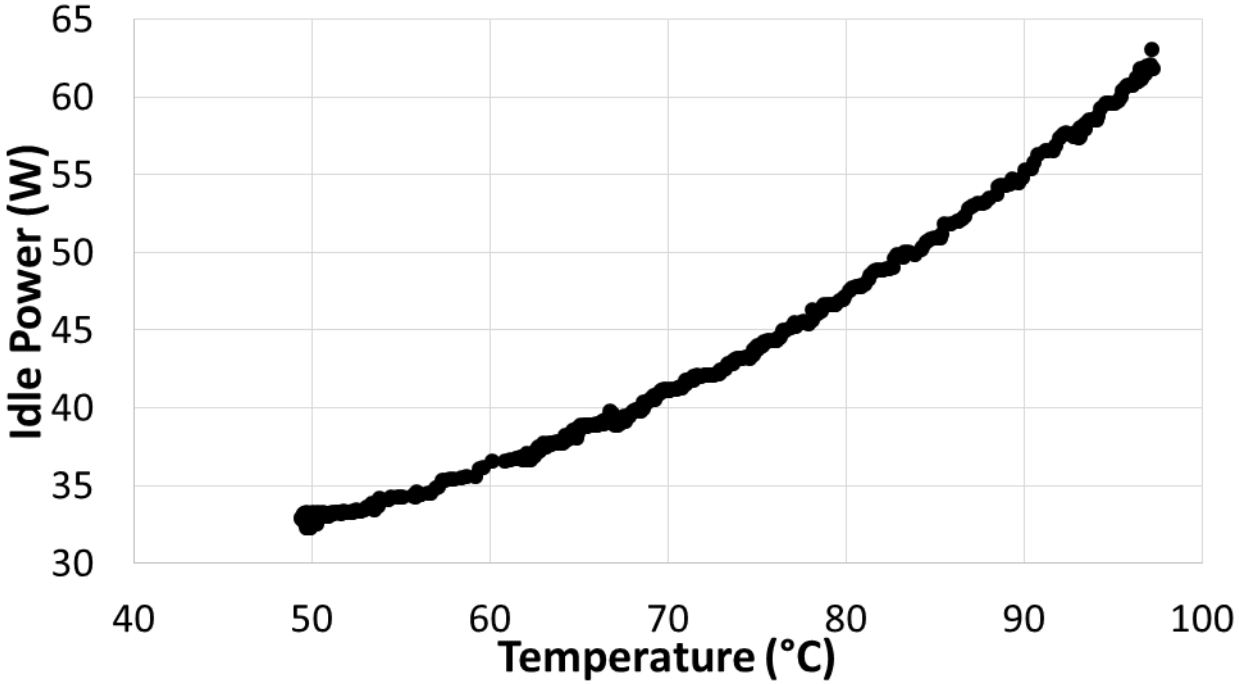
ADDRESSING THE CHALLENGES

OpenCL™ lacks native support to pin threads to programmer-specified compute units

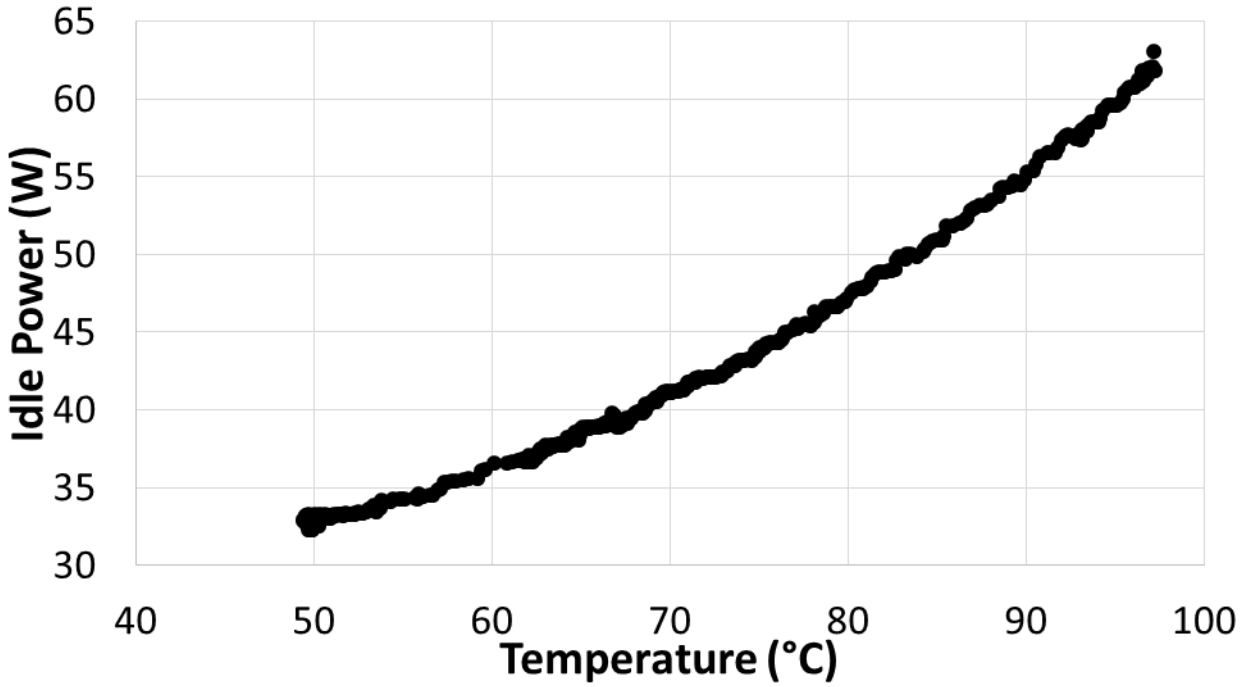
Temperature of device during tests will affect the power consumption

Power difference between the two sets of microbenchmarks can be hard to observe

ELIMINATING TEMPERATURE EFFECTS

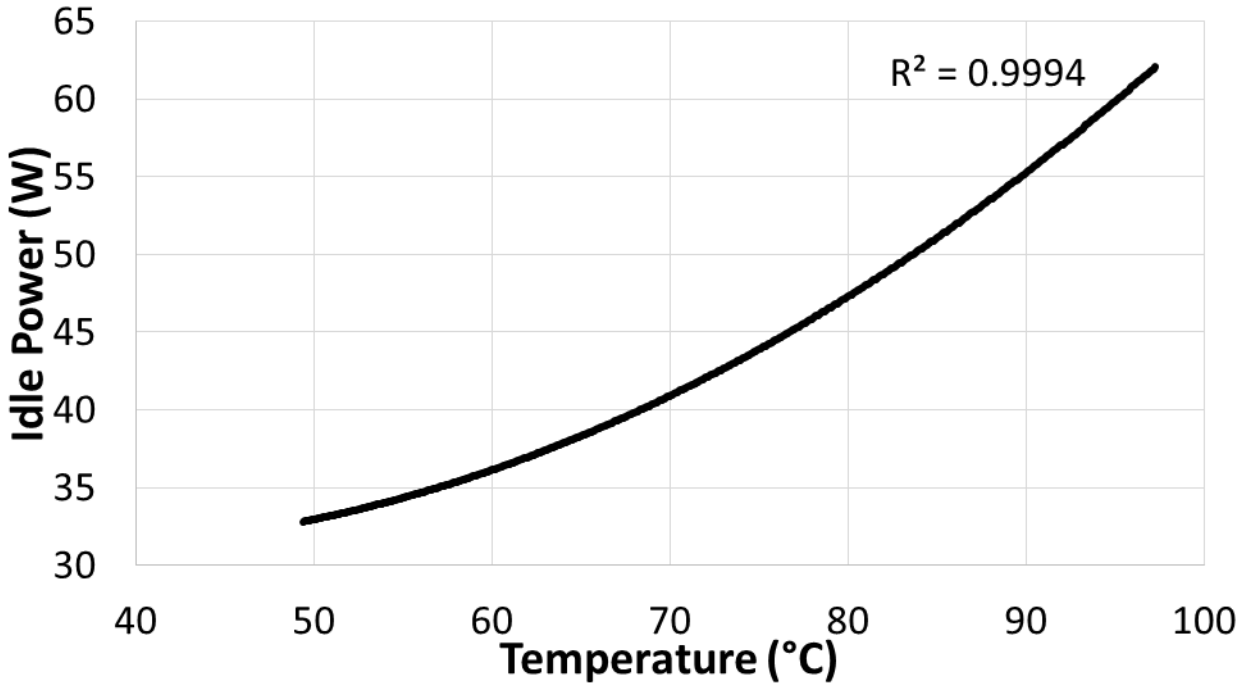


ELIMINATING TEMPERATURE EFFECTS



Solution 1: Run GPU fans at very high speed to limit temperature difference between runs

ELIMINATING TEMPERATURE EFFECTS



Solution 2: Model idle power separately and subtract from measured power

ADDRESSING THE CHALLENGES

AMD

OpenCL™ lacks native support to pin threads to programmer-specified compute units

Temperature of device during tests will affect the power consumption

Power difference between the two sets of microbenchmarks can be hard to observe

VirginiaTech Invent the Future PennState

SATURATING INTERCONNECT BANDWIDTH

- Power difference between microbenchmarks can be too low to be reliably observed
  - Unless the amount of data going through the interconnect increases

SATURATING INTERCONNECT BANDWIDTH

- Power difference between microbenchmarks can be too low to be reliably observed
  - Unless the amount of data going through the interconnect increases

- Difficult to saturate interconnect bandwidth without increasing the number of wavefronts
  - Fewer wavefronts can result in stalling exposing latency difference issues

SATURATING INTERCONNECT BANDWIDTH

- Power difference between microbenchmarks can be too low to be reliably observed
  - Unless the amount of data going through the interconnect increases
- Difficult to saturate interconnect bandwidth without increasing the number of wavefronts
  - Fewer wavefronts can result in stalling exposing latency difference issues
- More wavefronts can lead to one of the following issues
  - More L1 hits when accesses per thread is low
  - Register pressure and memory spills if access per thread is high

SATURATING INTERCONNECT BANDWIDTH

- Power difference between microbenchmarks can be too low to be reliably observed
  - Unless the amount of data going through the interconnect increases
- Difficult to saturate interconnect bandwidth without increasing the number of wavefronts
  - Fewer wavefronts can result in stalling exposing latency difference issues
- More wavefronts can lead to one of the following issues
  - More L1 hits when accesses per thread is low
  - Register pressure and memory spills if access per thread is high
- **Solution:** Modify firmware to artificially shrink L1 cache size and then increase number of wavefronts

CHALLENGES

AMD

Only a small difference between used-L2 and L1 cache size which can make it challenging to write L2-only microbenchmarks

Ensuring that the same amount of work is done by the two microbenchmarks can be challenging due to NUCA effects

Additional details in the paper

VirginiaTech Invent the Future PennState

AMD

Characterization Studies

Parameters Affecting Interconnect Power

Parameters Affecting Interconnect Power

AMD

Average interconnect distance

Parameters Affecting Interconnect Power

AMD

Average interconnect distance

Data toggle rate

Parameters Affecting Interconnect Power

AMD

- Average interconnect distance
- Data toggle rate
- Bandwidth observed on the interconnect

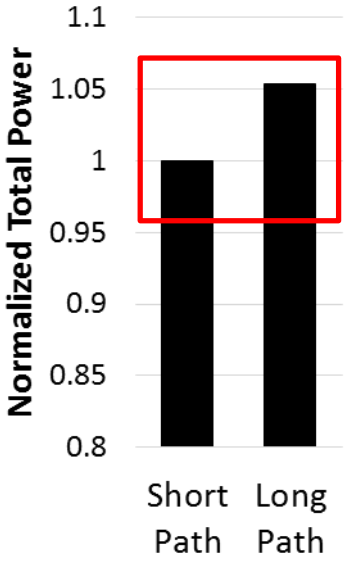
VirginiaTech Invent the Future PennState

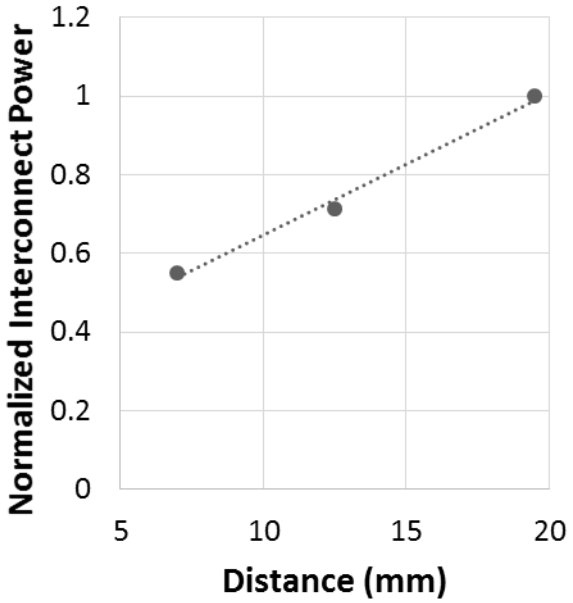
Parameters Affecting Interconnect Power

AMD

- Average interconnect distance
- Data toggle rate
- Bandwidth observed on the interconnect
- Interconnect's voltage and frequency

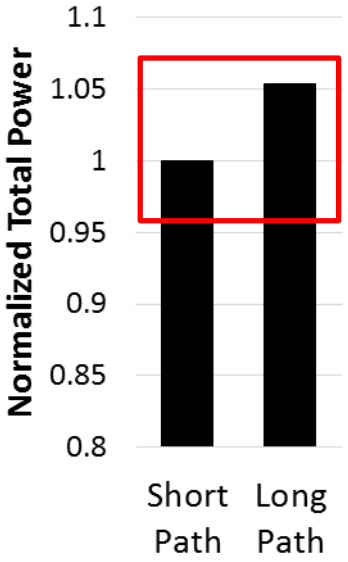
Interconnect Power vs Distance

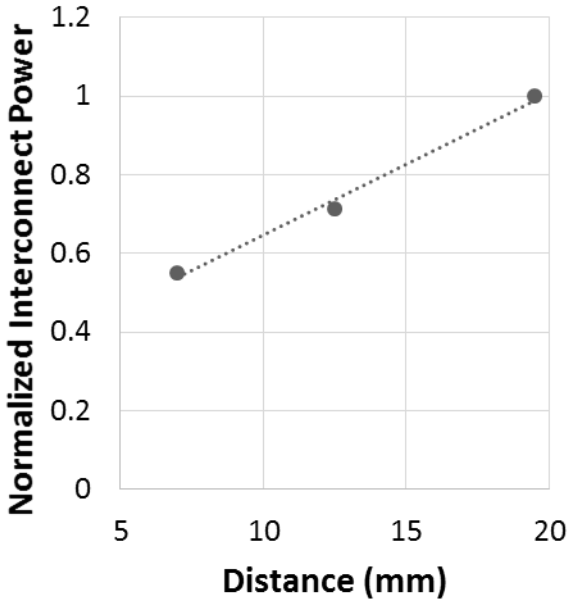




Our experiments confirm that the distance traversed by data affects power consumption

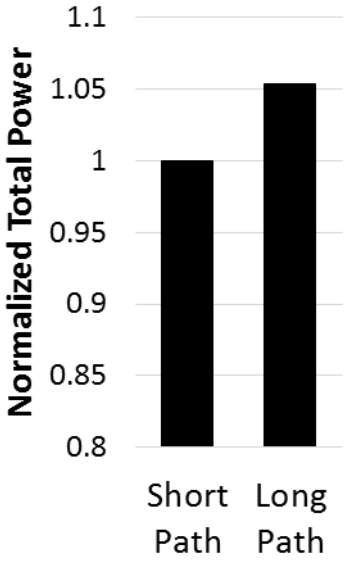
Interconnect Power vs Distance

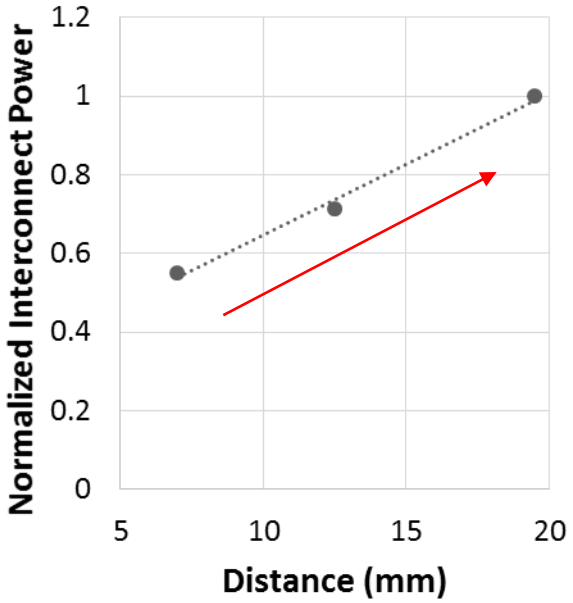




Within 15% of industrial estimates for energy/bit/mm

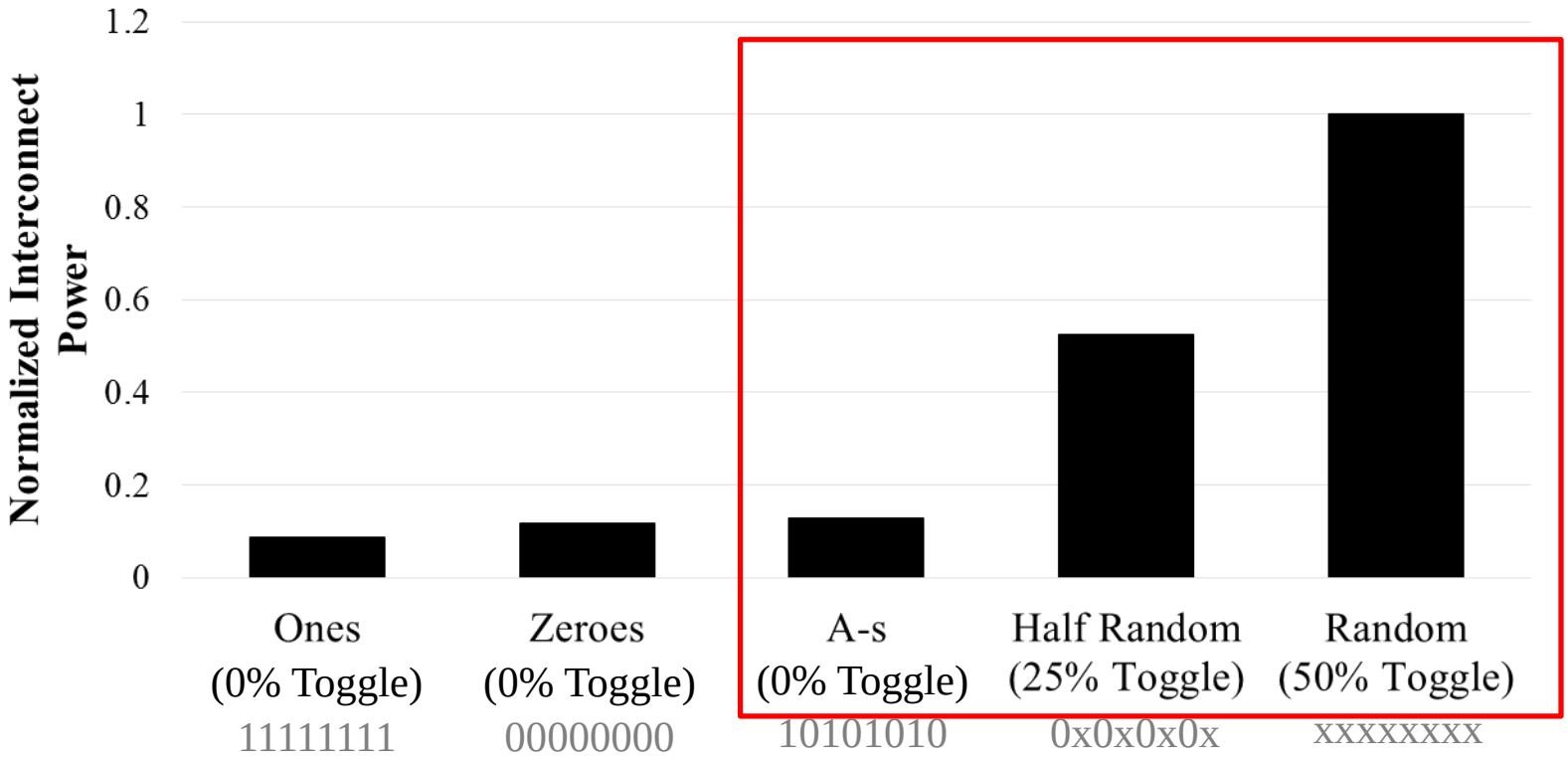
INTERCONNECT POWER VS DISTANCE





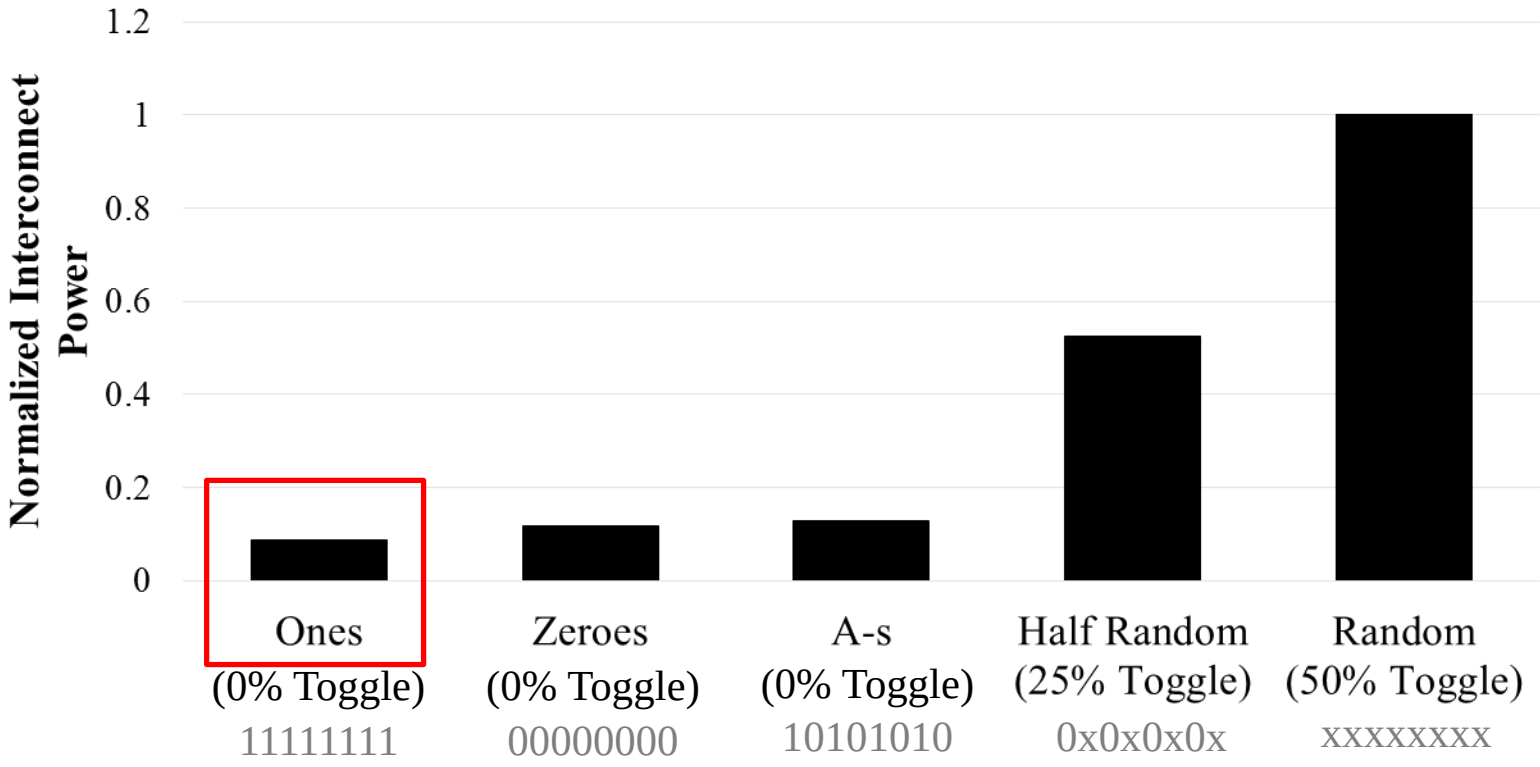
Linear relationship observed between data-movement distance and interconnect power

INTERCONNECT POWER VS TOGGLE RATE



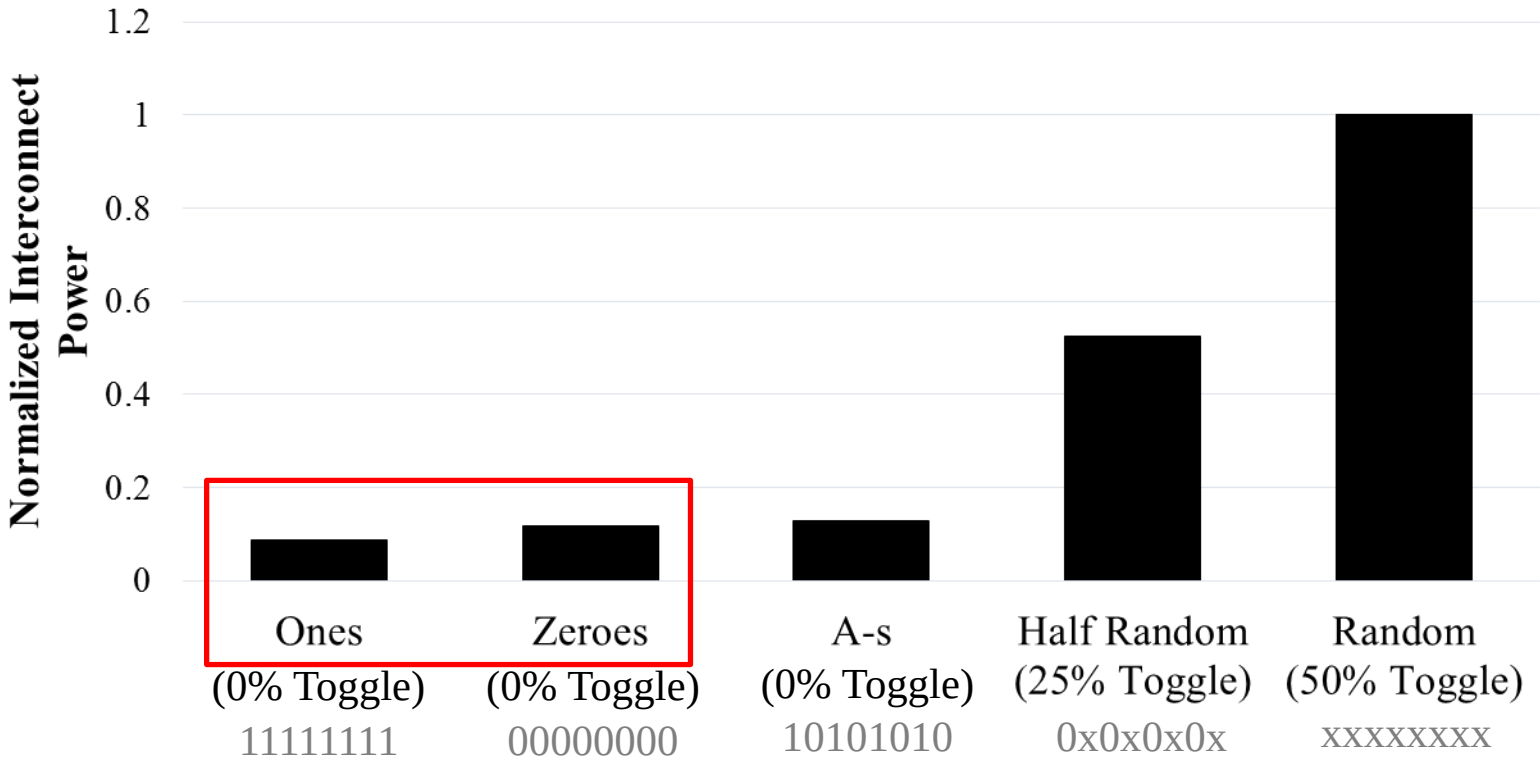
Linear relationship observed between toggle rate and interconnect power

Interconnect Power vs Toggle Rate



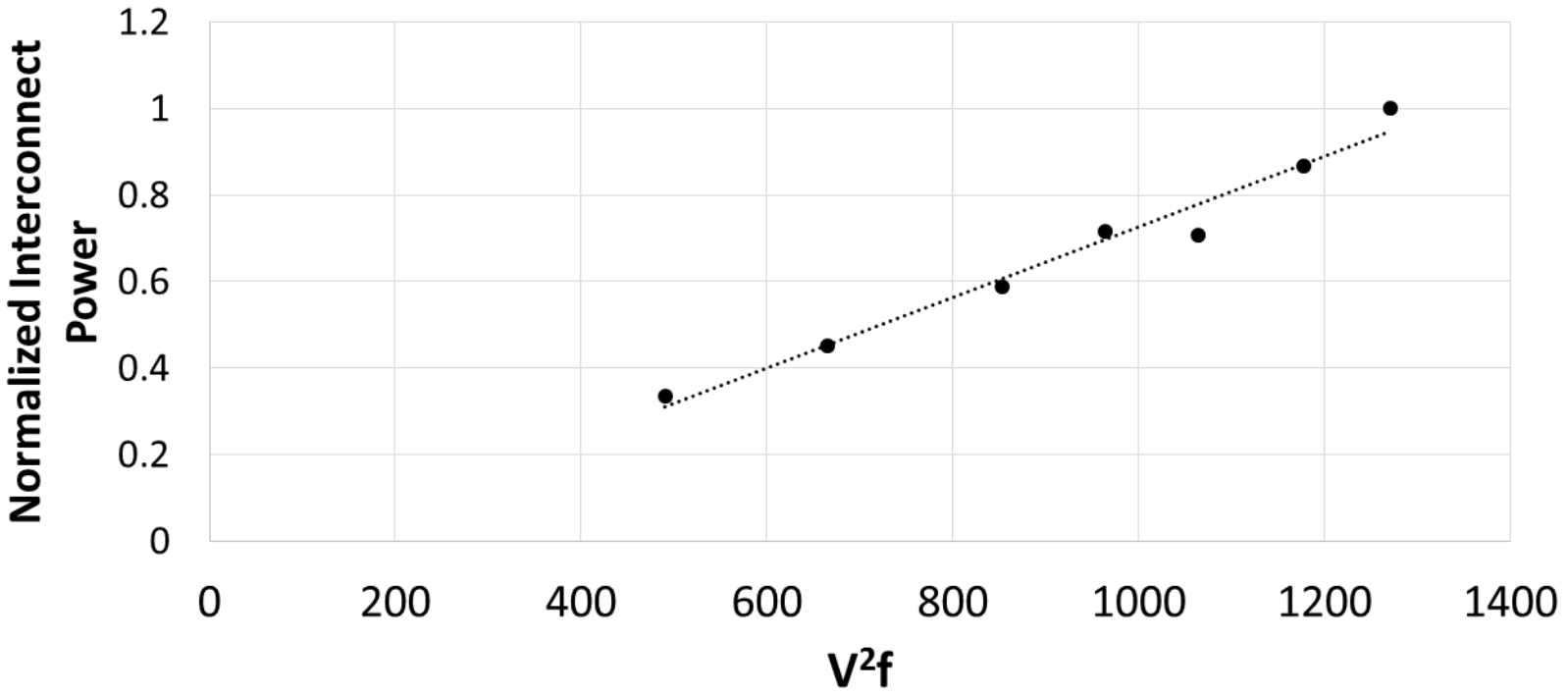
Approximately 10% of interconnect power goes towards arbitration, etc.

Interconnect Power vs Toggle Rate



Transmitting 0s slightly more expensive than 1s

Interconnect Power vs Voltage and Frequency



Impact of voltage and frequency as expected

PUTTING IT ALL TOGETHER

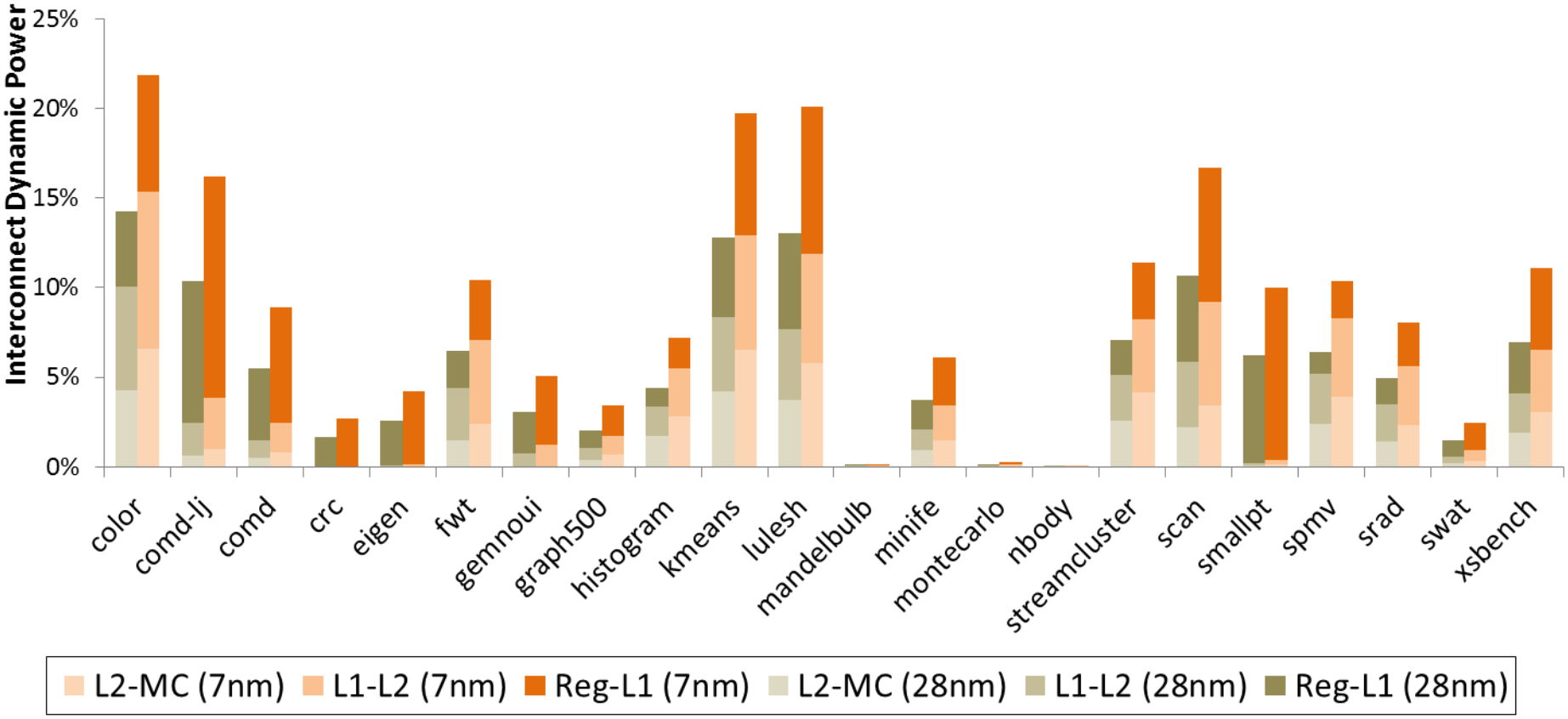
AMD

- Interconnect Power = Energy/bit/mm * avg. distance * avg. bits/sec * scaled voltage^2 x scaled frequency * avg. toggle rate

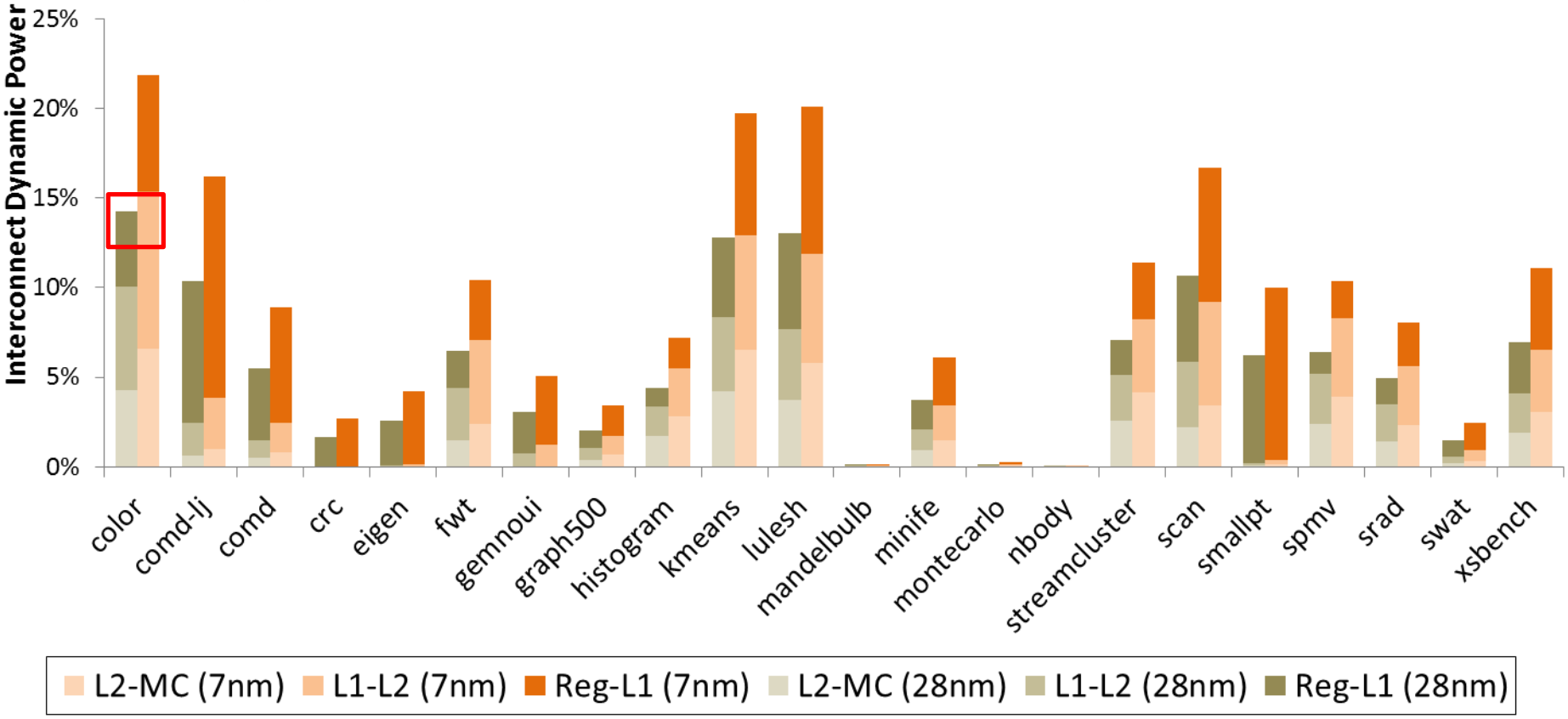
- For real applications,
  - Avg. bits per second is calculated from performance counters (e.g. L1 and L2 accesses)
  - The other parameters are pre-computed or pre-measured (e.g. Distance is measured from layout; energy/bit/mm is computed to be 110 fJ/bit/mm)

Evaluation of Applications

Energy Cost of Data Movement on Real Applications

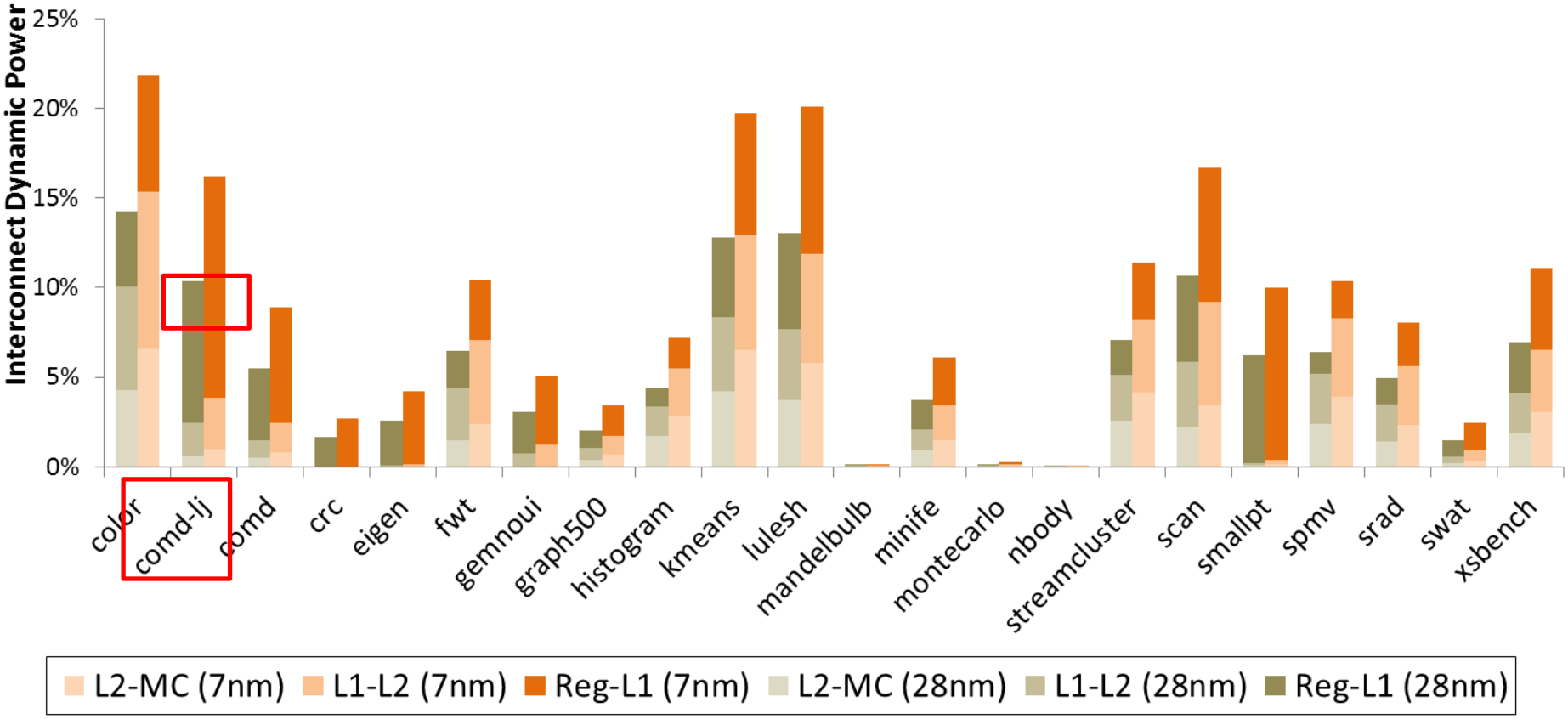


Energy Cost of Data Movement on Real Applications



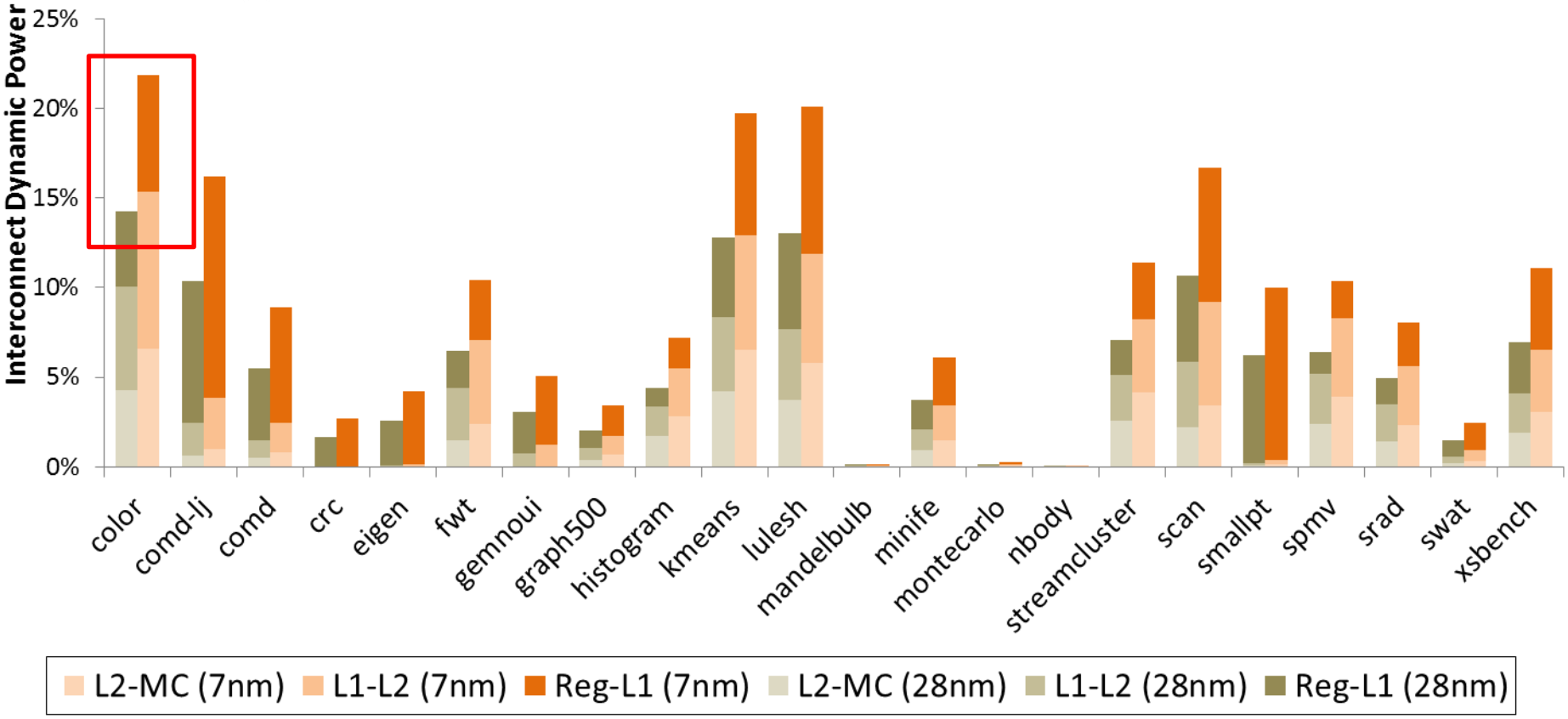
Up to 14% dynamic power spent on interconnects in today's chips

Energy Cost of Data Movement on Real Applications



Even non-memory bound applications can show high interconnect power (but at different hierarchy)

Energy Cost of Data Movement on Real Applications



Data-movement power problem exacerbated in future technology nodes

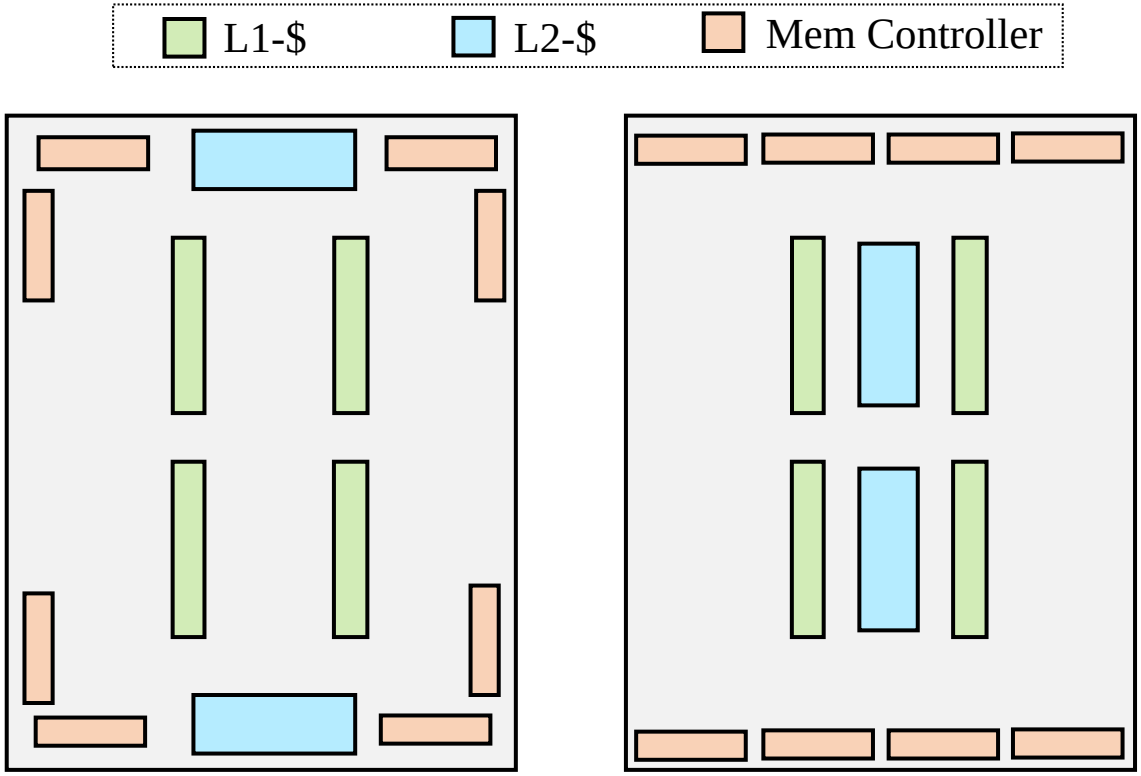
OPTIMIZATIONS

INTERCONNECT POWER OPTIMIZATIONS

AMD

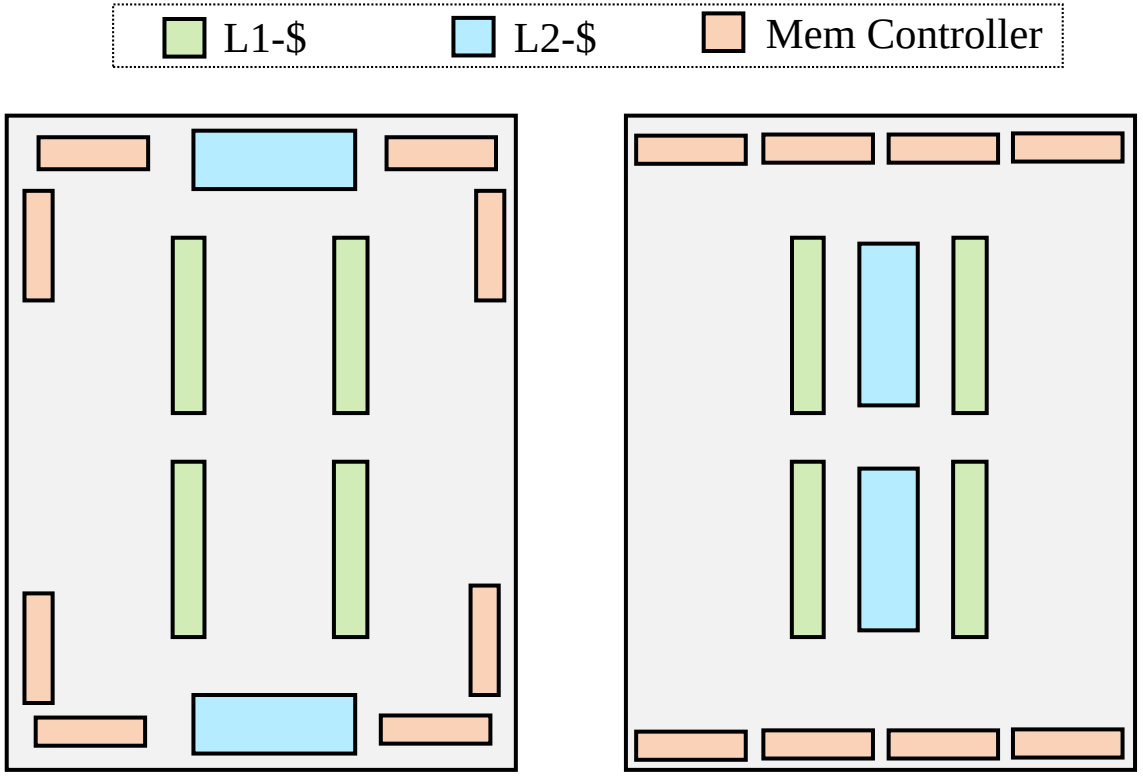
- **Layout-Based Optimizations**
  - We explore how much impact layout-based optimization can have on real world applications
  - We examine two layouts – one reduces distance between L1 and L2 and the other reduces distance between L2 and memory controller
- **Cache Resizing (details in the paper)**
  - Increasing cache size decreases average data movement distance (most data is fetched from nearer memories)
  - We quantify the magnitude of difference when we increase the cache size 4 times

Interconnect Power Optimizations – Layout Optimization

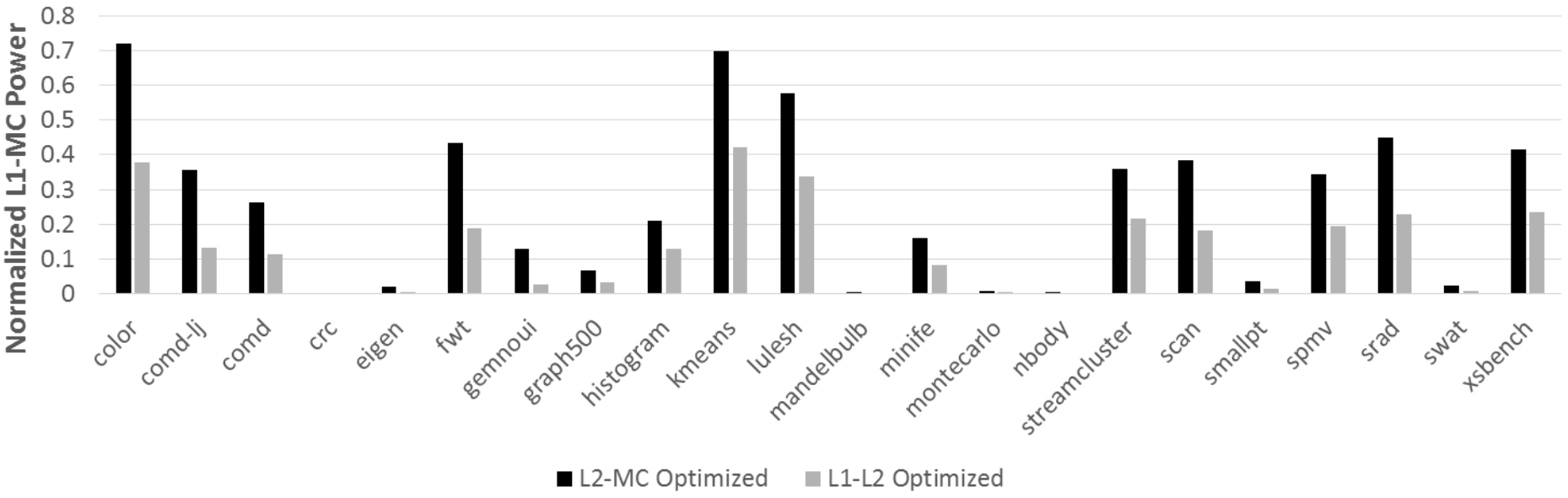


Optimized to reduce L2-MC distance.
Not too different from current GPUs.

INTERCONNECT POWER OPTIMIZATIONS – LAYOUT OPTIMIZATION



INTERCONNECT POWER OPTIMIZATIONS – LAYOUT OPTIMIZATION



Interconnect power reduces by 48% on average when we use an L1-L2 distance optimized layout

CONCLUSION

- Distance-based microbenchmarking is a promising approach to measure data movement power
- Over 14% of dynamic power can go towards on-chip data movement in today's chips
  - Lesser than past estimates as we separated out data access power from data movement power
  - Can increase to 22% by 7nm technology
- Optimizing on-chip interconnects to reduce the distance of frequently accessed portions can reduce on-chip data movement power by 48%

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