Measuring and Modeling On-Chip Interconnect Power on Real Hardware

Vignesh Adhinarayanan*, Indrani Paul†, Joseph L. Greathouse†, Wei Huang†, Ashutosh Pattnaik‡, Wu-chun Feng*

*Dept. of Computer Science
Virginia Tech
{avignesh, wfeng}@vt.edu

†AMD Research
Advanced Micro Devices, Inc.
{Indrani.Paul, Joseph.Greathouse, WeiN.Huang}@amd.com

‡Dept. of Computer Science and Engineering
Penn State University
ashutosh@cse.psu.edu

***Abstract*—On-chip data movement is a major source of power consumption in modern processors, and future technology nodes will exacerbate this problem. Properly understanding the power that applications expend moving data is vital for inventing mitigation strategies. Previous studies combined data movement energy, which is required to move information across the chip, with data access energy, which is used to read or write on-chip memories. This combination can hide the severity of the problem, as memories and interconnects will scale differently to future technology nodes. Thus, increasing the fidelity of our energy measurements is of paramount concern.**

We propose to use physical data movement distance as a mechanism for separating movement energy from access energy. We then use this mechanism to design microbenchmarks to ascertain data movement energy on a *real* modern processor. Using these microbenchmarks, we study the following parameters that affect interconnect power: (i) distance, (ii) interconnect bandwidth, (iii) toggle rate, and (iv) voltage and frequency. We conduct our study on an AMD GPU built in 28 nm technology and validate our results against industrial estimates for energy/bit/millimeter. We then construct an empirical model based on our characterization and use it to evaluate the interconnect power of 22 real-world applications. We show that up to 14% of the dynamic power in some applications can be consumed by the interconnect and present a range of mitigation strategies.

I. Introduction

Power and energy usage are first-class design constraints in almost all areas of modern computing. Phones, tablets, and laptops must run on batteries, so inefficient designs inconvenience users by requiring more frequent charges. Desktop users must pay power bills and deal with loud cooling mechanisms. Data centers and their servers are estimated to account for up to 1.5% of global electricity usage [19]. Even supercomputers are power constrained: the U.S. Department of Energy (DoE) has a goal of limiting the power consumption of exascale supercomputers to at most 20 MW [31].

Two major challenges associated with new silicon technology nodes have exacerbated these issues:

1) Dennard scaling has faltered, meaning that transistor density continues to increase, but the power used by each transistor no longer decreases at the same rate.
2) The power density of wires is increasing even faster than that of transistors due to poor wire size scaling. The cost of communication is thus a large and growing concern.

While data movement power has been recognized as a problem that needs to be addressed, the extent of the problem is not yet clearly understood [8], [23]. No previous study has accurately measured the data movement power in real, modern processors. Some of the difficulties are highlighted in the work of Leng et al. [21], which states: "It is almost impossible to isolate L2 cache power from NOC power because each L2 cache access involves an NOC access."

An implication of the above statement is that it is difficult to separate data access and movement costs with conventional measurement approaches. This limitation is also observed in the work of Kestor et al. [18], who were among the first to attempt to measure the energy cost of data movement on *real* hardware. Thus, despite the perceived importance of data movement power, no previous study has accurately measured it separately from data access power.

In this paper, we devise a set of novel techniques that allow us to overcome these limitations and separate the power of data movement from that of data accesses. To do this, we design microbenchmarks that use distance-based metrics, instead of traditional data volume metrics, to study the on-chip interconnects. Our microbenchmarks each have the same data access rates and perform the same operations, but differ in the physical distance that the data must travel within the interconnect. This allows us to separate the interconnect's power from data access power.

Our microbenchmarks allow us to characterize the interconnect power used by an AMD GPU built in 28 nm technology. We observe that the interconnect's power increases linearly with the distance of data movement, the wire toggle rate, and the bandwidth of data movement. Nonetheless, applications with the same toggle rate can consume different power based on the values sent along the wires due to the effect of crosstalk. We then use this data to develop *architecture-specific* empirical models and to study the interconnect power of 22 real applications running on our GPU. We then use our model to analyze power-reduction techniques, including chip layouts optimized for lower interconnect energy and cache resizing.

In summary, this paper makes the following contributions:

- **We describe a novel methodology to measure the interconnect power in *real* processors**. We design a series of microbenchmarks that use the same operations to access on-chip memories in different locations at the same rate. We demonstrate this on a modern AMD GPU, though our methodology can be used on any architecture.

- **We characterize the interconnect power of 22 applications both in 28 nm technology and in a hypothetical 7 nm node**. We show that up to 14% of the dynamic power in these applications comes from the interconnect and that this may increase to 22% in the 7 nm node.
- **We demonstrate our model's utility by exploring two previously proposed data-movement, power-reduction techniques**. We study *layout-based optimization*, or the impact of the placement of L2 and memory controllers within the chip, and the effect of *varying L1 and L2 sizes*, which changes the interconnect bandwidth.

This paper is organized as follows. We discuss related work in Section II. Section III describes our test setup, and Section IV details our interconnect power measurement methodology. We present the results of our characterization studies in Section V and our models in Section VI. We use our models to study real applications in Section VII and evaluate interconnect power mitigation techniques in Section VIII. We discuss future work and conclude with Section IX.

II. Related Work

Analytical modeling and simulation: At the lowest level, it is possible to model data movement power with circuit simulators such as SPICE. These tools provide excellent low-level details but require a great deal of design information and are extremely slow. It is unlikely that hardware designers would release SPICE-level models of large microprocessors. Even if this data were available, however, SPICE models precludes analyzing real applications on full SoC designs.

To partially work around these limitations, higher-level tools such as Orion [14] provide reasonably detailed models for the various interconnect components. Orion relies on data released by the industry to validate and fine-tune its model. With the limited information that is available in the public domain, researchers were able to increase the accuracy of earlier versions of Orion [35], [13], but the model needs constant revision as various interconnect technology advancements are released [34]. The sparsity of publicly available data on power breakdown for modern processors (that is usually released by industry) makes this revision and validation difficult. Our methodology makes it possible to independently obtain this reference data. In addition, our methodology also makes it possible to run real applications on hardware and obtain the data movement power for an entire application run rather than rely on worst-case estimates from low-level tools.

Other analytical models for interconnect power have been proposed in DSENT [32], GPUWattch [21] and McPAT [22]. Our work enables rigorous validation of such models by making it possible to independently obtain real-world interconnect power measurements on much larger designs and applications.

Microbenchmarking approaches: Previous works have also attempted to analyze data movement power on real processors. Like our study, they have the benefit of working on full designs and on modern technology nodes. However, these previous studies conflate data access and data movement energy. Because these two factors will scale differently to future process technologies, we wish to analyze them separately.

Kestor et al. [18] present a methodology for measuring the energy cost of moving data across the memory hierarchy for scientific workloads. Pandiyan et al. [27] present a similar approach for mobile workloads. They develop microbenchmarks that move data from different levels of the memory hierarchy to the registers. By measuring the difference in energy consumption between these microbenchmarks, they estimate the energy spent towards data movement. Unfortunately, this technique does not separate the energy cost of data movement from data access. For example, subtracting the energy cost of their L1-$ workload from the L2-$ workload, the resultant energy is not just the cost to move data from L2 to L1, but also includes the energy expended within the L2 cache.

Manousakis et al. [24] also adopt a microbenchmark-based approach where they vary the operational intensity of the microbenchmarks and study power consumption. Their study is also measures data accesses rather than data *movement*.

Component-level modeling using performance counters: Regression-based power models constructed using performance counters have the potential to estimate the power consumption of several components within a processor. Several works [30], [4], [9], [17], [37] have provided a breakdown for many components within a processor. However, these power models were only validated for overall power consumption and cannot be relied upon for component-level estimation.

III. Experimental Methodology

This section details the hardware we use during our studies and describes some pertinent microarchitectural details.

A. AMD GCN Architecture

For our tests, we used an AMD FirePro™ W9100 GPU, a workstation-class discrete GPU that uses the Graphics Core Next (GCN) 1.1 ISA [2]. A simplified block diagram of this GPU, which nonetheless roughly represents the location of many important structures, is shown in Figure 1. This GPU consists of four shader engines (SEs), each containing a number of compute units (CUs) that are similar to 64-wide vector processors. The AMD FirePro W9100 has 11 CUs per SE, yielding a total of 44 CUs (only 8 are shown in the figure for brevity).

Each CU has its own dedicated L1 data cache that is connected to the CU by short wires (not shown in the figure). The L2 cache is divided into several partitions (16 on our GPU), but every CU can communicate with every L2 partition via a crossbar interconnect. Each L2 partition is directly connected to an on-chip DRAM controller. As we will discuss later, these controllers (and thus also the L2 cache) are address sliced, such that each controller accesses (and each L2 partition caches) a disjoint subset of the memory space.

These L2 cache partitions are located in different parts of the chip, meaning that the physical distance between any pair of CU/L1 and an L2 partition can vary measurably. Each quadrant of the L2 cache has an SE "local" to it (i.e., the physical

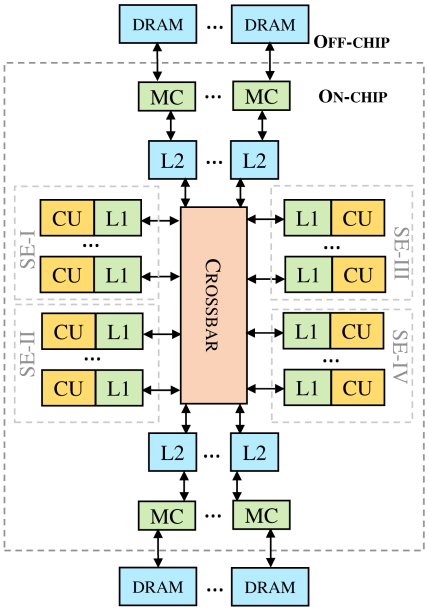


Fig. 1: Representative Block Diagram of the GPU, showing 8 out of 44 Compute Units (CUs)

Total Compute Units (CUs)	44
CUs per Shader Engine (SE)	11
Total SEs	4
Core Frequency	930 MHz
L1 Cache Size per CU	16 KB
Total L2 Cache Size	1024 KB
Number of L2 Partitions	16
Total DRAM Size	16 GB
Number of DRAM Channels	16
Memory Frequency	1250 MHz

TABLE I: Description of the AMD FirePro™ W9100 GPU

distance separating them is smaller compared to the distance between that SE and another L2 cache quadrant). For example, the CUs in SE-I in Figure 1 are closer to the L2 partitions at the top-left of the design than they are to L2 partitions at the bottom-right. We will exploit this observation to characterize the interconnect's power later in this paper.

The goal of this study is to estimate the power consumption of the on-chip interconnects and assess where data movement power is spent. As such, we focus on three major interconnects: (i) the wires between the CUs and L1 (ii) the crossbar connecting L1 and L2 and (iii) the wires between the L2 partitions and memory controllers. Characterizing the off-chip interconnects is beyond the scope of this study.

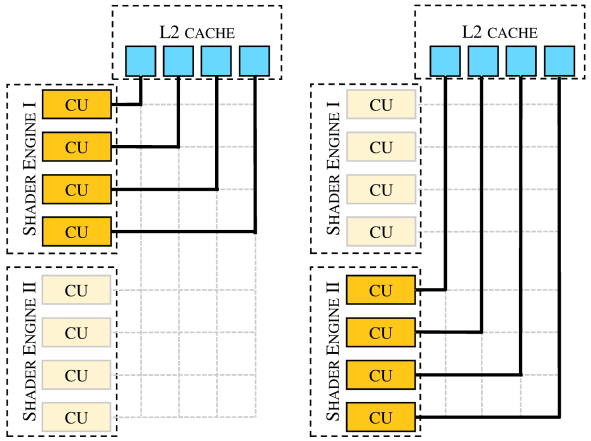
B. Experimental Setup

As previously mentioned, we performed our experiments on AMD FirePro™ W9100 discrete GPU. The key parameters of this GPU are listed in Table I.

Software Setup: We ran our experiments on a host with Ubuntu 14.04, v15.20.7 of the AMD FirePro drivers, and the AMD APP SDK v2.9.1. Our microbenchmarks use OpenCL™ 1.2.

Power Monitoring: To monitor the power consumption of our GPU, we use a high-precision power meter that measures current and voltage from the voltage regulators going into the chip. This instrument can provide power measurements at 1 kHz. The instrumentation setup is capable of measuring the power consumption of only the chip as a whole, and hence the study is limited to focusing just the on-chip data movement and not the off-chip movement (e.g., to main memory).

Performance Counters: To guide the design of the microbenchmarks and to validate them, we use AMD CodeXL v1.6. We later describe how AMD CodeXL performance counters can be used to estimate interconnect power in larger applications in Section VI and VII.



(a) Short Path (b) Long Path

Fig. 2: Design of our interconnect power microbenchmarks

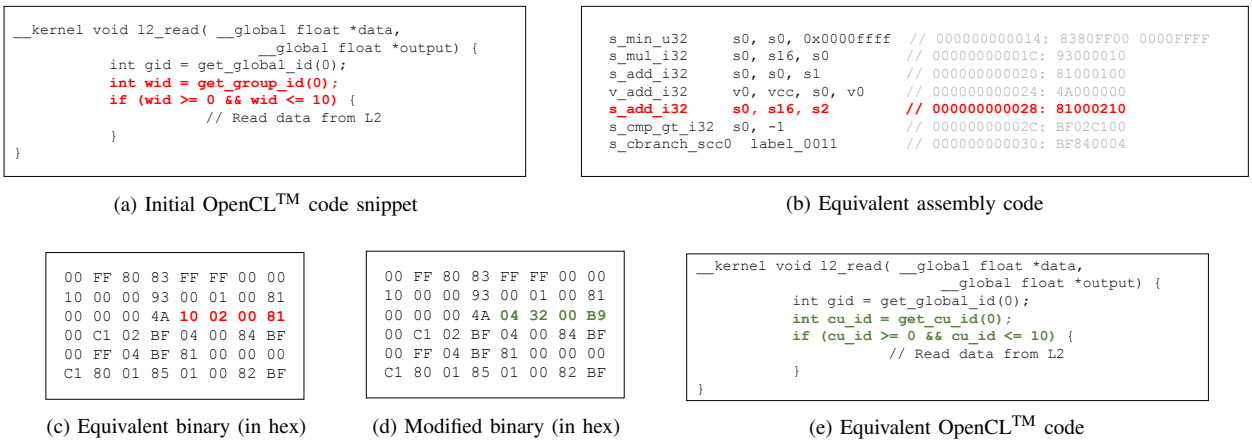
IV. Measuring Interconnect Power

This section describes our microbenchmarking strategy for measuring the interconnect power of the processor described in Section III. While the details are specific to our GPU, the methodology itself is generalizable.

A. A Power Measurement Technique

Our microbenchmarking methodology is based on the observation that longer wires consume more energy than shorter wires while carrying the same current. Therefore data that travels a longer physical distance within the chip consumes more energy than the same amount of data moving a shorter distance.

Our conjecture based on the above observation is that when we continuously move data from a partition of the L2 cache to the various L1 caches that are located in the different parts of the chip, we should observe a difference in power consumption. To test this conjecture, we design two microbenchmarks, illustrated in Figure 2. The first (referred to as **short-path**) continuously moves data between compute units (CUs) in shader engine I and the L2 quadrant closest to it. The second (referred to as **long-path**) moves the data between shader engine II and the same L2 quadrant, thereby moving the data through a longer physical distance.



(a) Initial OpenCL™ code snippet

(b) Equivalent assembly code

(c) Equivalent binary (in hex)

(d) Modified binary (in hex)

(e) Equivalent OpenCL™ code

Fig. 3: Steps to launch wavefronts on only one shader array

Realizing this design on real hardware and accurately measuring the power difference is a non-trivial task, which we will explain and solve in the following sections.

B. Details of Microbenchmark Implementations

Realizing the basic idea presented in Section IV-A on real hardware poses several challenges that must be mitigated:

1) We use OpenCL™ to implement our microbenchmarks, but it lacks native support to pin threads to programmer-specified locations on the chip.
2) Designing a microbenchmark where all of the data is fetched from one quarter of the L2 cache is challenging, since each L2 quadrant contains only 256 KB, whereas the total size of an SE's L1 caches is 176 KB.
3) The microbenchmarks must use as much bandwidth as possible to reliably observe and measure the chip-wide power difference between the two microbenchmarks.
4) Latency effects must be hidden from the long-path microbenchmark. Because the second shader engine is located on a physically different part of the chip from the first SE, there is an increase in latency when it accesses the top-left L2 quadrant. Sufficient L2 requests must be generated so that the long-path microbenchmark sees the same bandwidth as the short-path microbenchmark.
5) Temperature has a major impact on the power consumption of a processor. The effects of temperature on the two microbenchmarks should be properly isolated so that only the effect of data-movement distance is measured.

C. Locking OpenCL™ Kernel to Specific SEs

While OpenCL™ does not directly offer support for running threads on only one shader engine (SE), it is possible to achieve the effect by editing the binary which is generated by the OpenCL runtime. An example is shown in Figure 3, where work is performed only on CUs 0 through 10 (i.e., SE-I). In this approach, we write an initial OpenCL snippet, shown in Figure 3a, in which useful work is performed only if the wavefront ID is between 0 and 10. Wavefront ID is a placeholder that we will modify to hold the value of CU IDs.

A part of the equivalent GCN assembly code for this snippet is shown in Figure 3b. The instruction that writes the value of the wavefront ID to the variable used in the *if* conditional is boldfaced and highlighted in red. The scalar register corresponding to this variable is `s0` and the hex of the instruction that writes its value is `81000210`. Figure 3c shows the hex value of the instruction in little endian format in the binary file, which can be obtained using `clGetProgramInfo()`.

We then manually replaced this instruction with the `S_GETREG_B32` GCN instruction, which loads the CU and SE IDs out of the `HWID` register and puts them into `s0` (`B9003204`, as shown in Figure 3d). This derivation is based on the information provided in AMD ISA manuals [2], [1]. We further verified that this process resulted in the desired effect by analyzing the performance counters from AMD CodeXL.

After making these modifications, we can use `clCreateProgramWithBinary()` to load our custom binary into the application. This achieves the same effect as writing the hypothetical OpenCL code shown in Figure 3e.

D. Accessing Data Only from L2

Our microbenchmarks attempt to access data from one quadrant (4 out of the 16 partitions) of the L2 cache that is located closest to SE-I. In our target architecture, there is a one-to-one mapping between the memory channels and the L2 partitions. That is, the data that resides in one memory channel can be cached in only one L2 partition. The address interleaving for the memory channels is specified in AMD's ISA manuals [2].

Each channel holds contiguous 256 bytes of memory (equivalent of 64 floats) and, given an address, it is possible to identify the channel number from bits 8-11. Using the above information, it is possible to obtain the L2 cache partition given an array index for any data type and thus write a microbenchmark that only targets a particular partition.

E. Saturating the L1-L2 Interconnect Bandwidth

In order to obtain the best results from our microbenchmarks, we must use as much L1-L2 bandwidth as possible. To begin with, higher bandwidth means a greater difference in sum total of the data moved for each benchmark on a per-second basis, which should translate to a greater difference in power consumption between the two microbenchmarks. This difference will help minimize error from other uncontrollable sources, such as measurement noise. In addition, saturating the interconnect also helps in keeping the CU pipelines busy, helping to prevent long-path from stalling more often than short-path due to any difference in L2 access latency.

Unfortunately, launching a small number of wavefronts to a small number of CUs cannot saturate the L1-L2 interconnect if they only touch one cache line before stalling. We could design our microbenchmarks such that each thread accesses several cache lines. This would require extra address calculations and could potentially increase the global working set size, however, resulting in register pressure and unwanted main memory accesses. Alternately, we could increase the number of wavefronts kept in flight. This could inadvertently increase the L1 hit rate by scheduling threads in a way that keeps all of the data accessed by one wavefront in the L1 cache. Despite this, we chose the latter option.

To prevent an increase in the L1 cache hit rate, we modified the firmware of our GPU to artificially shrink the size of L1 cache to 4 KB per CU. This allowed us to increase the number of wavefronts in flight (thereby increasing the interconnect bandwidth and hiding the L2 access latency for long-path), avoid cache hits in L1, and keep accesses to the main memory to an absolute minimum and focus on compulsory misses only.

F. Isolating Temperature Effects

Modern silicon technology nodes consume a great deal of static power which is, in turn, affected by operating temperature. As explained in Section III-B, however, our power measurements come from the off-chip voltage regulators. These regulators must supply all power to the chip, both static and dynamic. This means that our measurements cannot directly differentiate between the two. To this end, we developed a small set of tests to help us isolate the dynamic power in the interconnect from static and other non-interconnect power.

We build a power model for idle power to capture the effect of temperature on power. We gathered the data required to build this model by fixing the frequency and voltage of the GPU and heating the chip with a computationally intensive application (e.g., the FurMark benchmark). After the GPU reaches our target temperature, we stop the benchmark and allow the chip to cool while still maintaining the frequency and voltages. As the chip cools, we continually measure the chip's temperature using the on-chip thermal sensors and the chip's power using our power monitor. Figure 4 shows the idle power of our target device across the range of temperatures that we studied. We can observe from this data that there is a non-linear relationship between idle power and temperature. This effect of temperature should thus be separated from our voltage regulator measurements to accurately measure power consumption of the interconnect.

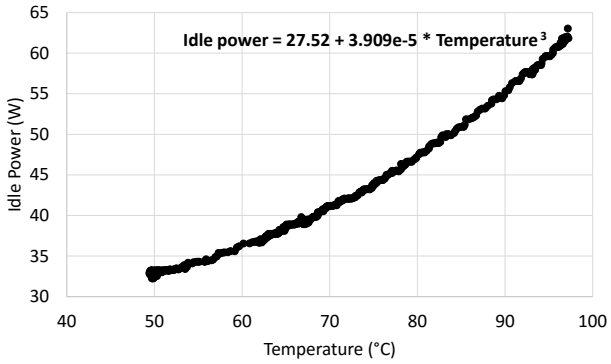


Fig. 4: Relationship between idle power and temperature

To achieve this separation during our microbenchmarks, we run the GPU's fan at high speed to constraint the device temperature. We then construct an idle power model for the device using a regression of the data we present in Figure 4, which models idle power as a cubic function of the device temperature. The model is optimized for the typical operating temperature range for our microbenchmarks in order to increase its accuracy. Using this model, we subtract out the idle power for the microbenchmark tests from our voltage regulator measurements. This allows us to separate out the effects of temperature from our tests and focus on interconnect power caused by communication.

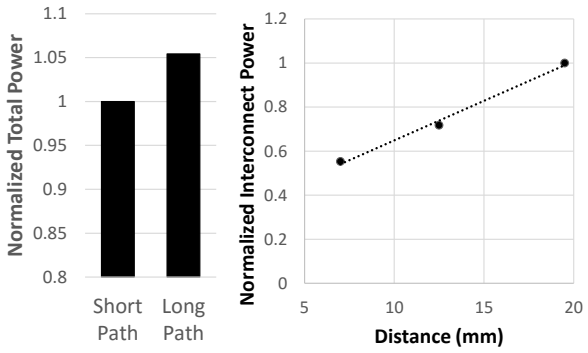
V. Characterization of Interconnect Power

In this section, we present the results of our microbenchmark studies that show the impact of the following parameters on interconnect power: (i) data-movement distance, (ii) toggle rate, (iii) voltage and frequency, and (iv) interconnect bandwidth.

A. Impact of Data Movement Distance

Figure 5a shows the average dynamic power consumption for the short-path and the long-path microbenchmarks. The values presented in the y-axis are normalized against the short-path microbenchmark. This figure shows that long-path consumes 5% more chip-wide dynamic power than short-path. These two microbenchmarks have identical computational and data access rates as verified from hardware performance counters. Therefore, the additional power can only be attributed to the higher data movement distance for the long-path microbenchmark. This additional distance is estimated to be 10.5 mm from an analysis of a die photo of the GPU [36].

Validation efforts. We converted the observed difference in power for a distance of 10.5 mm to a metric known as *energy/bit/mm*, which is the energy cost to move one bit of data through a physical distance of 1 mm. This value was compared against industrial estimates available for 40 nm [16] and 32 nm [5] technology nodes using appropriate scaling factors



(a) Benchmark power (b) Distance vs. interconnect power

Fig. 5: Impact of data movement distance on interconnect power.

from [5]. We found that our estimate for energy/bit/mm was within 10% and 15% of these two industrial estimates.

Next, we study the relationship between data movement distance and the interconnect power. For this study, we developed microbenchmarks that are variants of the short-path and the long-path microbenchmarks. The basic idea behind the microbenchmarks remain the same, but instead of running OpenCL™ threads on 11 CUs (i.e., an entire shader engine), we run them only on 4 CUs. This allows us to obtain the difference in power consumption for different distances. The values obtained for the interconnect power from *four* such microbenchmarks are presented in Figure 5b. In this figure, the x-axis represents data-movement distance and the y-axis represents interconnect power normalized against the highest value observed in this set of experiments. One of the four microbenchmarks is used to obtain reference power based on which the other three microbenchmarks are studied. Therefore, we have three data points in the graph. Our characterization result shows that the interconnect's power increases linearly with data-movement distance.

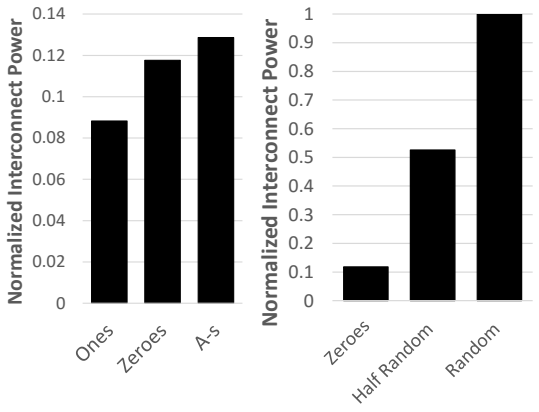
B. Impact of Toggle Rate

Next, we studied the impact of toggle rate on interconnect power. For this study, we moved different data patterns across the interconnect and observed the power difference for the short-path and the long-path. The patterns studied are shown in Figure 6. Of these, *zeros*, *ones*, and *A*s show no toggling. *zeros* and *ones* are self-explanatory; for *A*s, we send a pattern of alternate *1*s and *0*s, which when represented in hexadecimal is a string of *A*s. For the random data, each bit can take any value and the probability of bit toggling (i.e., a transition from *1 to 0* or *0 to 1*) is 0.5. For the half random dataset, a few bits of random data and a few bits of zeros alternate. The overall toggle rate for this dataset is 0.25.

Figure 7a shows the normalized interconnect power for data patterns showing 0% toggle rate. Figure 7b shows the same for data patterns exhibiting toggle rates from 0% to 50%. The normalization is performed against the random dataset. Note

Zeroes	0	0	0	0	0	0	0	0
Ones	1	1	1	1	1	1	1	1
A-s	1	0	1	0	1	0	1	0
Half Random	0	0	x	x	0	0	x	x
Random	x	x	x	x	x	x	x	x

Fig. 6: Data patterns explored in this study



(a) 0% toggle rate (b) 0-50% toggle rate

Fig. 7: Toggle rate and data pattern impact on interconnect power

that the figures are drawn to different scale. We make the following observations from this study:

1) Sending only zeros or ones consumes a small amount of power in the interconnect (about 10% of the power seen for random data). This is primarily because the arbiters present within the interconnect consume a small amount of energy regardless of the data pattern.
2) Transmitting zeros consumes more power than ones.
3) Interference from neighboring bit lines has a small, but noticeable impact on the interconnect power. This can be seen from the fact that A-s consume more power than zeros despite showing 0% toggle and transmitting fewer power-hungry 0 bits.
4) Toggle rate has a significant impact on the interconnect power as seen from zeros (0% toggle), half-random (25%), and random data (50%). The relationship between toggle rate and interconnect power is linear.

C. Impact of Voltage and Frequency

We repeat our experiments while setting the GPU to different DVFS states (i.e., voltage and frequency combinations) in order to study the impact of voltage and frequency on the interconnect power. Figure 8 shows the normalized interconnect power for these DVFS states. In this figure, the interconnect power is plotted against V^2f which is the expected relationship between voltage, frequency and power. As expected, the relationship between them is linear. Note that

the interconnect bandwidth (or the amount of data) differs at the various points in the graph as the frequency changes.

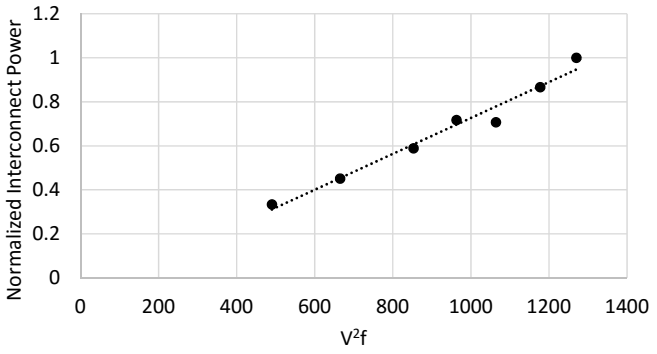


Fig. 8: Normalized interconnect power for moving data at different DVFS states

D. Impact of Interconnect Bandwidth

Next, we study changes in interconnect power when the amount of data that moves through it changes. To perform this experiment, we inserted NOPs in our code to reduce the frequency of data access from the L2 cache which also reduces the interconnect bandwidth. A lower bandwidth means fewer bit transitions per second and consequently lower power. Figure 9 shows this for two different bandwidths, where we observe that the interconnect power is roughly half when the interconnect bandwidth is reduced to half its original value.

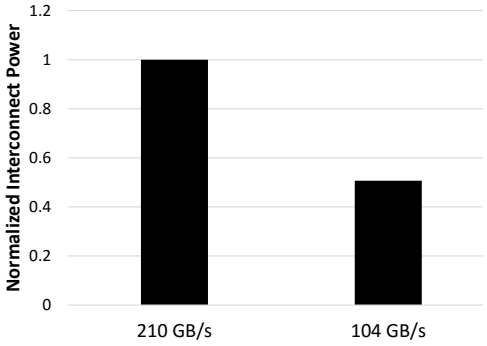


Fig. 9: Impact of interconnect bandwidth on interconnect power.

VI. Modeling Interconnect Power

The characterization results presented in Section V can be combined into a parameterized equation which naturally lends itself to model interconnect power of larger applications, different chips, and different technology nodes. The general form of the parameterized equation can be expressed as follows:

Interconnect Power = Constant $\times$ % Peak Bandwidth $\times$ Toggle Rate $\times$ Distance $\times$ Scaled Frequency $\times$ Scaled Voltage^2

Constant refers to the maximum power consumed by the interconnect for a given chip and a reference DVFS state. This value is calculated from the difference in power consumption between short-path and long-path (shown in Figure 5a), which is then scaled for peak bandwidth, 100% toggle rate, and unit wire distance. The *constant* value is architecture specific and can be derived for existing GPUs using the microbenchmarks described in Section III and extrapolated to future technology nodes using process scaling information [5].

Interconnect	Estimated distance
Register to L1	3.5 mm
L1 to L2	10.5 mm
L2 to memory controller	11.5 mm

TABLE II: Average distance estimates for the different parts of the interconnect

Next, we will describe how to estimate interconnect power for real applications at different interconnect segments of the memory hierarchy, as shown in Figure 1, using hardware performance counters (PCs). First, the obtained bandwidth (BW) is calculated for each interconnect segment from the PCs for *L1 accesses*, *L2 hits*, and *L2 misses*. This gives a measure of the actual data volume for an application at different segments.

$$Reg\ to\ L1\ BW = \frac{L1\ accesses}{Time} \times L1\ width$$

$$L1\ to\ L2\ BW = \frac{L2\ hits + L2\ misses}{Time} \times L2\ width$$

$$L2\ to\ MC\ BW = \frac{L2\ misses}{Time} \times MC\ width$$

The bus width of L1 cache and L2 cache is 64 bytes and the width of memory controller (MC) is 32 bytes for our target architecture. The obtained BW is then expressed as a percent of the peak interconnect BW. The calculation for the peak L1-L2 BW is shown as an example below:

$$Peak\ L1\ to\ L2\ BW = \#\ L2\ banks \times 64\ bytes\ per\ bank \times clock\ rate$$

Next, *toggle rate* is the probability of bit toggling for a given program. For completely random data, the expected probability of toggling is 0.5. The typical average toggle rate observed for the interconnects is 0.34 [3].

Distance is an estimate of the average distance the data has to move through the interconnect. For existing GPUs where application threads are not pinned to any particular CU and accesses are evenly distributed across all L2 slices, using average distance for calculations is a reasonable assumption. For the AMD FirePro™ W9100 GPU, we calculated the average distance for each part of the interconnect by using layout information from the design, though public die photos could also be used [36]. The distances we measured are presented in Table II.

The interconnect power for any voltage and frequency pair can be calculated by scaling these parameters with respect to the reference voltage and frequency pair. Alternatively, the constant factor may be recalculated from the microbenchmarks for the required voltage and frequency.

VII. Estimation of Interconnect Power for Real Applications

In this section, we estimate the interconnect power of 22 OpenCL™ applications obtained from various sources shown in Table III. These applications were chosen considering that the maximum frequency for our power meter is 1 kHz and the chosen applications all have OpenCL kernels that run long enough (over 2 ms) to get meaningful power measurements. Total GPU power for each application at run-time is measured using the power meters measuring voltage and current from the voltage regulators. We also measure the average temperature of the GPU chip across all its thermal sensors while running the applications. To extract dynamic power from these measurements, the idle power is subtracted using the temperature-idle power relationship described in Section IV-F.

In our evaluation, using our performance-counter driven model, we estimate the interconnect power spent by the application at the various parts of the on-chip interconnects: (i) Register to L1 (ii) L1 to L2 and (iii) L2 to memory controller (MC). The results are presented for the 28 nm AMD FirePro™ W9100 GPU architecture and a hypothetical 7 nm shrink of the same die. For the hypothetical chip, we use Borkar's scaling factors for wires and transistors [5] to scale the total dynamic power and interconnect power from 28nm to 7nm.

Figure 10 shows the power spent on the different parts of the interconnect, expressed as a percentage of overall dynamic power, for the various applications for the 28 nm and the hypothetical 7 nm GPUs. Due to the lack of toggle rate monitors in hardware, for these results, we assume an average toggle rate of 0.34 for all applications which is based on past studies [3]. Across applications, the on-chip interconnect consumes 5.6% of the total dynamic power on our GPU on an average. Within the interconnect, *register to L1* consumes the most power, using over 45% of the total interconnect power. The crossbar consumes 30% of the total interconnect power and the rest is consumed by *MC to L2*.

Among all applications, *color* shows the highest percentage of 14.3% for interconnect power. This is due to the fact that *color* is an irregular application with many branch and memory divergence, causing large amount of data accesses at different levels of the memory hierarchy. *Comd-lj*, *kmeans*, *lulesh*, and *scan* also consume significant amount of interconnect power, with over 10% of the overall dynamic power going towards the interconnect. Of these, *kmeans*, *lulesh*, and *scan* are either memory-bound or partially memory-bound, and understandably consume a greater amount of interconnect power as data has to be frequently fetched from the distant memory. *Comd-lj* is largely compute-bounded with most data accesses either going to register file or L1. Although the distance between the SIMD units and L1 is relatively small, it still has a significant amount of power spent in data movement because of the high data access counts to L1.

Source	Applications
AMD APP SDK	eigen, fwt, histo, montecarlo, nbody, scan
DOE proxy apps	CoMD and CoMD-LJ [25], XSBench [33], LULESH [15], and miniFE [12]
Graph500 [26]	graph500
OpenDwarfs [11]	crc, gemnoui, swat
Pannotia [6]	color
Phoronix [20]	mandelbulb, smallpt
Rodinia [7]	kmeans, streamcluster, srad
SHOC [10]	stencil, spmv

TABLE III: Applications used for evaluation

On the other extreme, applications such as *mandelbulb*, *montecarlo*, and *nbody* all consume nearly *zero* interconnect power. These are all compute-bounded, but unlike *comd*, the working set for these applications fits within the register files and therefore doesn't access L1 much. Therefore, they avoid short distance accesses as well and see a lower data movement power.

On the 7 nm architecture, the trends remain the same. But, the interconnect consumes 8.9% of the total dynamic power across applications. Individually, we see up to 21.9% for interconnect power as in the case of *color*. These values correspond to nearly 59% increase in the interconnect power for real applications. This highlights that data movement is going to be an even more significant problem in future GPUs.

VIII. Evaluating Optimization Techniques

The interconnect power model presented in this paper can be used to evaluate and guide several optimization techniques in a variety of scenarios ranging from design-time optimization to runtime management of interconnect power. This section presents some of these techniques as examples that show how our power model can be used in evaluations of such techniques and/or used as a part of these optimization techniques.

A. Layout-based Optimization

In this section, we present a use case for our model which is used to quickly evaluate different layouts in order to find the one that minimizes data movement power. Intuitively, by reducing the physical distance for the part of the interconnect that is being used the most, one can save data movement power. In this section, we quantify the savings possible using two sample layouts that optimizes different parts of the interconnect.

Figure 11 shows the two sample layouts. In the baseline case, the average Manhattan distance between L1 and L2 is 17.0 units and the average distance between L2 and MC is 7.6 units. The layout on the right tries to reduce the L1 to L2 distance at the cost of a significant increase in L2 to MC distance. The distances for this layout are 3.5 units for the L1-L2 interconnect and 12.0 units for the L2-MC interconnect.

We use our model to calculate the power consumed by these interconnects for the layouts presented in Figure 11. We assume that the conditions are similar to our experimental platform: (i) 28 nm technology node, (ii) 1.1687 V, (iii) 930 MHz, and (iv) the same constant factor in our equation, owing to equivalent wire capacitance. The normalized power for the interconnects between L1 and MC is presented in Figure 12 for our testing applications.

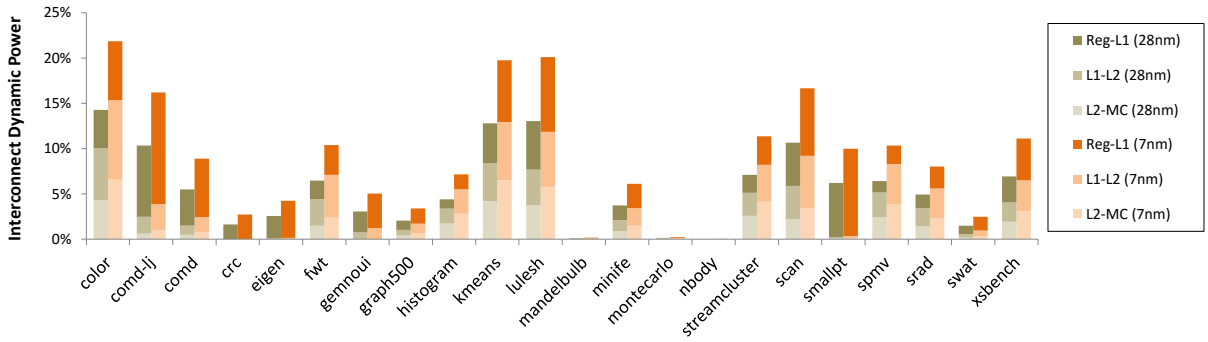


Fig. 10: Percentage of the total dynamic power spent by the interconnect on the 28 nm FirePro™ W9100 GPU and a hypothetical 7 nm die shrink. The assumed toggle rate is 0.34 for all the applications.

We observe that the L1-L2 optimized layout consistently consumed less power for all the applications. On an average, the L1-L2 optimized layout consumed 48% lower power for the interconnects between L1 and MC. A maximum of 79% reduction in power was observed for *eigen* as there are far fewer references to memory than to L2 for this application. Our results thus show the importance of prioritizing L1-L2 interconnects over L2-MC interconnects.

B. Cache Resizing Optimization

Cache resizing techniques have been explored in the past to optimize energy-delay of caches [38]. In these techniques, parts of the cache is turned off to reduce their static power as long as any additional delay encountered is offset by the reduction in the static power. Disabling caches may happen statically, before an application's execution, or dynamically, during an application's execution. In this paper, we point out the presence of another important parameter in this trade-off. Reducing the cache size not only increases delay, but also increases the amount of data moving in the longer wires thereby consuming more dynamic power. In this section, we quantify the change in data movement power as we increase or decrease cache sizes.

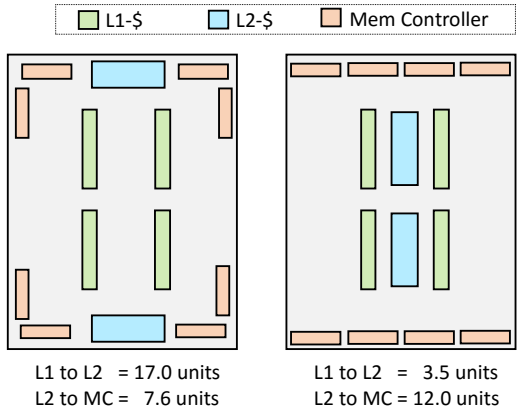


Fig. 11: Two sample layouts that are designed to reduce the distance between L2 cache and memory controller (left), and the distance between L1 cache and L2 cache (right).

Figure 13 shows the decrease in data movement power in the L1-L2 interconnect as we increase the L1 cache size from 4 KB per CU to 16 KB per CU. On an average, we observe a 9% reduction in the interconnect power by increasing the L1 cache. A maximum reduction of 37% is observed for *swat*. This decrease occurs because increasing the L1 cache size increases the hit rate and therefore, can reduce the average distance that the data has to move on an average. A runtime system may use such data along with the estimated increase in static power to make decisions on whether to increase or decrease cache size. The trade-off analysis and the implementation of such a system is beyond the scope of this paper. Here, we show only the savings in data movement power that is possible from cache resizing.

Similar to the above experiment, we also increased the L2 cache size from 256 KB to 1024 KB, a factor of 4. The interconnect power for this cache size, normalized against a baseline of 256 KB, is presented in Figure 14. By increasing the L2 cache size, we could save 9% of the L2-MC interconnect power on an average. Savings up to 25% is observed for L2-MC interconnect as in the case of *comd*. The decision to increase L2 cache size, however, would depend up on other additional parameters such as static energy and average access latency. However, the design of a decision algorithm is beyond the scope of this paper.

IX. Conclusion and Future Work

In this paper, we devised a novel methodology to measure interconnect power using carefully developed distance-based microbenchmarks. We then developed an empirical model using hardware performance counters to obtain the interconnect power for any large application. We evaluated 22 applications and showed that up to 22% of the dynamic power of a GPU can be consumed by the interconnect in the 7 nm node. Finally, we explored two solutions to reduce interconnect power and showed that optimizing the chip layout to reduce data movement and developing cache-resizing policies goes a long way in combating data-movement power issues.

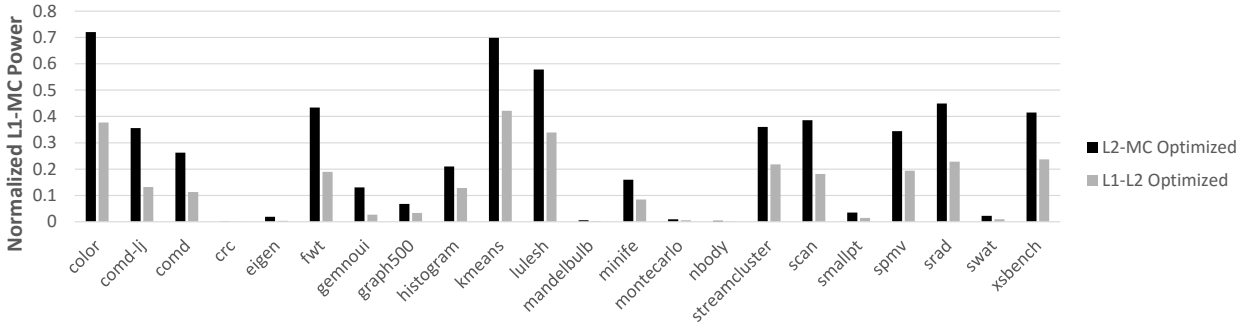


Fig. 12: Normalized interconnect power for L2-MC optimized layout and L1-L2 optimized layout

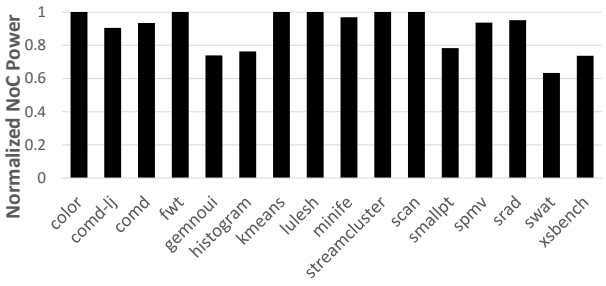


Fig. 13: Impact of changing the L1 cache size on L1-L2 interconnect

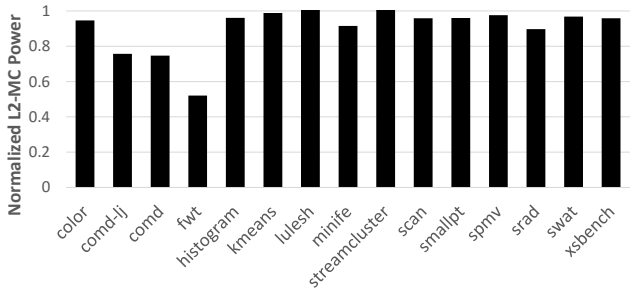


Fig. 14: Impact of changing the L2-cache size on L2-MC interconnect

Future Work. Previous works have shown methods for dynamically sharing power between components to optimize for energy usage or to increase performance under power caps [28]. Because interconnect power will become such a large power user in future technology nodes, it may be interesting to design of a runtime system that dynamically moves power away from the interconnect for bandwidth-tolerant workloads in order to provide additional power to other parts of the chip.

Our data implies that toggle- and crosstalk-aware compression schemes for reducing interconnect power may be an interesting future research direction. Such schemes would need to decide when to compress data depending upon the potential increase in latency compared to the reduction in toggle rate and crosstalk. This is similar to the work by Pekhimenko et al. [29], but taking into account the effects of crosstalk and using a model derived from realistic data.

Finally, the design of a runtime cache-resizing scheme that takes the cost of data movement into account in addition to the cache energy and delay considered earlier could potentially improve global energy efficiency [38].

ACKNOWLEDGEMENT

We thank peers at Advanced Micro Devices, Inc. for their assistance with this work. Thanks also to the anonymous reviewers for their feedback.

Vignesh Adhinarayanan and Ashutosh Pattnaik were interns at AMD Research when this research was performed.

REFERENCES

[1] AMD, "Reference Guide: Southern Islands Series Instruction Set Architecture," http://developer.amd.com/wordpress/media/2012/12/AMD_Southern_Islands_Instruction_Set_Architecture.pdf, 2012.

[2] AMD, "Reference Guide: Sea Islands Series Instruction Set Architecture," http://developer.amd.com/wordpress/media/2013/07/AMD_Sea_Islands_Instruction_Set_Architecture.pdf, 2013.

[3] J. H. Anderson and F. N. Najm, "Switching Activity Analysis and Prelayout Activity Prediction for FPGAs," in *Proc. of the Int'l Workshop on System-level Interconnect Prediction (SLIP)*, 2003.

[4] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and Responsive Power Models for Multicore Processors using Performance Counters," in *Proc. of the Int'l Conf. on Supercomputing (ICS)*, 2010.

[5] S. Borkar, "Exascale Computing - A Fact or a Fiction?" in *Int'l Symp. on Parallel Distributed Processing (IPDPS)*, 2013, pp. 3–3.

[6] S. Che, B. M. Beckmann, S. K. Reinhardt, and K. Skadron, "Pannotia: Understanding Irregular GPGPU Graph Applications," in *Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC)*, 2013.

[7] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC)*, 2009.

[8] J. Chen, A. Choudhary, S. Feldman, B. Hendrickson, C. Johnson, R. Mount, V. Sarkar, V. White, and D. Williams, "Synergistic Challenges in Data-Intensive Science and Exascale Computing," *DOE ASCAC Data Subcommittee Report, Department of Energy Office of Science*, pp. 1–70, 2013.

[9] G. Contreras and M. Martonosi, "Power Prediction for Intel XScale® Processors using Performance Monitoring Unit Events," in *Proc. of the Int'l Symp. on Low Power Electronics and Design (ISLPED)*, 2005.

[10] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The Scalable Heterogeneous Computing (SHOC) Benchmark Suite," in *Proc. of the Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU)*, 2010.

[11] W. Feng, H. Lin, T. Scogland, and J. Zhang, "OpenCL and the 13 dwarfs: A Work in Progress," in *Proc. of the Int'l Conf. on Performance Engineering (ICPE)*, 2012.

[12] M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich, "Improving Performance via Mini-Applications," *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, vol. 3, 2009.

[13] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *Proc. of the Conf. on Design, Automation and Test in Europe (DATE)*, 2009.

[14] A. B. Kahng, B. Lin, and S. Nath, "ORION3.0: A Comprehensive NoC Router Estimation Tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, 2015.

[15] I. Karlin, J. Keasler, and R. Neely, "LULESH 2.0 Updates and Changes," Tech. Rep. LLNL-TR-641973, August 2013.

[16] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the Future of Parallel Computing," *IEEE Micro*, no. 5, pp. 7–17, 2011.

[17] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Enabling Accurate Power Profiling of HPC Applications on Exascale Systems," in *Proc. of the Int'l Workshop on Runtime and Operating Systems for Supercomputers (ROSS@ICS)*, 2013.

[18] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the Energy Cost of Data Movement in Scientific Applications," in *Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC)*, 2013.

[19] J. G. Koomey, "Growth in Data Center Electricity Use 2005 to 2010," Analytics Press, Tech. Rep., August 2011. Available: http://www.analyticspress.com/datacenters.html

[20] M. Larabel and M. Tippett, "Phoronix Test Suite," http://www.phoronix-test-suite.com/, 2011.

[21] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: Enabling Energy Optimizations in GPGPUs," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 2013.

[22] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2009.

[23] R. Lucas, Ed., "Top Ten Exascale Research Challenges," *DOE ASCAC Subcommittee Report*, pp. 1–86, 2014.

[24] I. Manousakis and D. S. Nikolopoulos, "BTL: A Framework for Measuring and Modeling Energy in Memory Hierarchies," in *Proc. of the Int'l Symp. on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2012.

[25] J. Mohd-Yusof, "Codesign of Molecular Dynamics (CoMD) Proxy App," LA-UR-12-21782, Los-Alamos National Lab, Tech. Rep., 2012.

[26] R. C. Murphy, K. B. Wheeler, B. W. Barrett, and J. A. Ang, "Introducing the Graph 500," *Cray Users Group (CUG)*, 2010.

[27] D. Pandiyan and C.-J. Wu, "Quantifying the Energy Cost of Data Movement for Emerging Smart Phone Workloads on Mobile Platforms," in *Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC)*. IEEE, 2014, pp. 171–180.

[28] I. Paul, W. Huang, M. Arora, and S. Yalamanchili, "Harmonia: Balancing Compute and Memory Power in High-Performance GPUs," in *Proc. of the Int'l Symp. on Computer Architecture (ISCA)*, 2015.

[29] G. Pekhimenko, E. Bolotin, M. O'Connor, O. Mutlu, T. C. Mowry, and S. W. Keckler, "Toggle-Aware Compression for GPUs," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 164–168, July 2015.

[30] M. D. Powell, A. Biswas, J. S. Emer, S. S. Mukherjee, B. R. Sheikh, and S. Yardi, "CAMP: A Technique to Estimate Per-Structure Power at Run-time using a Few Simple Parameters," in *Proc. of the Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2009.

[31] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale Computing Technology Challenges," in *Proc. of the Int'l Conf. on High Performance Computing for Computational Science (VECPAR)*, 2011.

[32] C. Sun, C. H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. S. Peh, and V. Stojanovic, "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *Proc. of the Int'l Symp. on Networks on Chip (NoCS)*, 2012.

[33] J. R. Tramm, A. R. Siegel, T. Islam, and M. Schulz, "XSBench-The Development and Verification of a Performance Abstraction for Monte Carlo Reactor Analysis," *The Role of Reactor Physics toward a Sustainable Future (PHYSOR)*, 2014.

[34] A. N. Udipi, N. Muralimanohar, and R. Balasubramonian, "Non-Uniform Power Access in Large Caches with Low-Swing Wires," in *Proc. of the Int'l Conf. on High Performance Computing (HiPC)*, 2009.

[35] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," in *Proc. of the Int'l Symp. on Microarchitecture (MICRO)*, 2002.

[36] WCCFTech, "Dieshots of Pitcairn, Tahiti, and Hawaii GPUs," http://cdn.wccftech.com/wp-content/uploads/2013/12/AMD-Hawaii-GPU.jpg, accessed: 2016-03-18.

[37] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan, "A Systematic Method for Functional Unit Power Estimation in Microprocessors," in *Proc. of the Design Automation Conf. (DAC)*, 2006.

[38] S.-H. Yang, M. D. Powell, B. Falsafi, and T. N. Vijaykumar, "Exploiting Choice in Resizable Cache Design to Optimize Deep-submicron Processor Energy-Delay," in *Proc. of the Int'l Symp. on High-Performance Computer Architecture (HPCA)*, 2002.