Research Statement

Jason Lowe-Power
University of Wisconsin-Madison

Computer systems are at a crossroads. Instead of counting on low-level device improvements from the semiconductor industry, processors are increasingly heterogeneous, using specialized accelerators to increase performance. Simultaneously, big-data is enabling new applications that consume terabytes of data in real-time, and the Internet of things is enabling developers to reach millions of devices. Computer architecture research is going to play a vital role in continuing the information revolution by bridging the gap between emerging applications and the underlying physical technology.

The increasing hardware heterogeneity driven by the slowdown of Moore's Law puts significant burdens on the application programmers. With current interfaces, developers must manually manage all aspects of accelerator computation. For instance, programmers must explicitly move data from CPU to GPU memory, even if the physical memory is shared because each device has a logically separate address space. My work ***increases the logical integration of physically integrated on-chip accelerators*** by providing conventional CPU properties to accelerators specifically on-die GPUs which are found on most products from AMD, Intel, and mobile systems. This logical integration simplifies programming heterogeneous devices. Through interdisciplinary collaborations, I have leveraged the performance improvements from this logical integration which was enabled by my previous research. I show that using ***integrated GPUs for analytic database*** workloads can increase performance and decrease energy consumption. I am excited to explore new research problems exposed by the evolving microprocessor industry from *novel processors* like always-on sensors to *new applications* that will consume processor cycles.

Current Research

Programmable accelerators

Until about 10 years ago, energy efficiency, and therefore performance, doubled every few years through a combination of Moore's Law and Dennard scaling. However, this trend is waning; for instance, Intel no longer reduces the transistor size every 2 years, shifting to an every 3–4 year cadence. In response, the microprocessor industry has increased the heterogeneity of systems. A single chip now has CPUs, GPUs, image processors, and video accelerators with many other accelerators proposed for the future. Unfortunately, although these heterogeneous components are *physically* on the same silicon die, most are not *logically* integrated. For instance, GPUs have been shown to provide significant performance improvement for some workloads. However, developers must use cumbersome programming models to interface with these accelerators which leads to suboptimal performance, wasting up to 95% of the execution time for some workloads. My work logically integrates the memory systems of CPUs and GPUs in two ways: coherent data access and a consistent address space. Additionally, I developed a mechanism to ensure safety in the presence of logically integrated third-party accelerators.

Coherent data access and a consistent address space simplify the programmer's interface to tightly-integrated accelerators. Parallel hardware has been ubiquitous for at least the last 15 years, and while programming parallel machines is still difficult, it is simplified by two important properties: *coherent data access* so when developers access a memory location it will hold the most up-to-date data and a *consistent address space* so programmers can use the same data structures on any processor in the system. The cache coherence hardware and the virtual memory management unit provide these properties on current CPU systems. Without coherent data access and a consistent address space, programmers manually transform and move data between the CPU and other accelerators. My work is the first to bring these semantics to on-die integrated GPUs, easing programmers' use of this pervasive hardware.

I found the main impediment to providing a shared virtual address space and cache coherence between the integrated GPU and the CPU is the immense bandwidth demand of the GPU. GPUs have more compute resources and can issue many more memory requests, producing an order of magnitude more bandwidth than typical CPU workloads. Thus, applying CPU techniques to heterogeneous systems results in poor performance for GPU applications.

For instance, when applying the traditional CPU coherence mechanisms to CPU-GPU coherence, I found the centralized shared hardware directory is the bottleneck. My insight in **Heterogeneous System Coherence (HSC)** [1] is to leverage the spatial locality inherent in accelerator programs and decouple coherence permission from memory access. HSC eliminates 94% of directory accesses, improving performance by 2× compared to traditional CPU techniques and paving the way for the GPU to be a first-class participant in the cache coherence protocol.

Similarly, conventional CPU address translation mechanisms applied directly to the GPU result in a 4× performance degradation due to queuing delays at the centralized address translation hardware. In **Supporting x86-64 Address Translation for 100s of GPU Lanes** [2] I show how to gain the performance back by parallelizing address translation and judiciously using CPU-centric memory management unit optimizations. These techniques logically unify the physically integrated GPU and pave the way for programmers to use the integrated GPU as just another CPU core.

Providing safety for heterogeneous systems allows system designers the freedom to integrate third-party accelerators into their own products. Logically integrating accelerators provides enormous benefits; however, many emerging accelerators are not implemented in-house; for example, Apple used Imagination Technologies' GPUs in the iPhone6 system-on-a-chip. Logically integrated accelerators may be given unfettered access to the memory system to enable high performance. However, bugs or malicious third-party hardware can allow attackers to break the confidentiality and integrity of system memory even if the application on the accelerator has never accessed it. **Border Control** [3] sandboxes the processes running on accelerators. In my design, every memory access can be checked when crossing the border from the trusted to untrusted domain with low overhead by decoupling permission checking from address translation. Each access is checked against a small permission table, built on-the-fly from the operating system page table permissions.

Simulation infrastructure for heterogeneous systems is required to investigate these emerging technologies. I am one of the original creators and lead developers of infrastructure to simulate this emerging heterogeneous hardware, the **open source project gem5-gpu** [4]. gem5-gpu is built with the leading research simulators (gem5 and GPGPU-Sim), and it was quickly released to the public so others could use and improve it [5]. gem5-gpu is the main infrastructure in many published papers both within and external to the University of Wisconsin.

Emerging technology for big-data workloads

The logical integration of GPUs with the rest of the system opens the door to using the GPU for new applications. Previous work has shown GPUs can provide up to 100× performance improvement, but these results are limited to small working sets (e.g., less than 8 GB). My previous research enables the GPU to access the hundreds of gigabytes of system memory and significantly reduces the overheads of conventional GPU programming expanding the scope of applications that can leverage this accelerator. I have specifically applied these techniques to analytic databases.

Analytic queries are growing in complexity, and their response time requirements are shrinking. Service providers want to analyze terabytes of data in milliseconds to provide their users with pertinent information (e.g., serving ads, suggesting purchases, and search results). To provide low latency, much of the data is stored in memory, not on disk. Previous research had shown the significant potential of discrete GPUs; however, due to data movement costs, this potential had not been realized. I collaborated with database experts to develop new algorithms to take advantage of tight physically and logically integrated GPUs enabled by my previous research, and I showed how to elide data copy and transformation operations. My **new scan and aggregate algorithms** [6] are 3× faster than traditional GPU algorithms and use 30% less energy than a CPU-only system. I additionally showed integrating 3D die-stacked DRAM into analytic database appliances coupled with my new algorithms can improve performance by 15× [7]. Further, I conducted a limit study showing these 3D die-stacked systems provide the best power-performance tradeoff when the latency of big-data analytic workloads is paramount (2–5× less power for a 10 ms SLA) [8].

While integrating high-bandwidth 3D die-stacked memory (HBM) can provide significant performance improvements by mitigating the off-chip bandwidth bottleneck, its capacity is limited. Therefore, systems with 3D die-stacked memory will likely additionally have conventional off-chip RAM. In a heterogeneous memory system, using HBM as a DRAM cache of off-chip RAM provides transparency to programmers. However, I found that treating HBM as a cache wastes its bandwidth because of *access amplification*. Access amplification measures the non-demand accesses that check and update cache metadata. I found there are on average two accesses per DRAM cache request. My **adaptive victim DRAM cache** [9] design avoids access amplification by adaptively shifting between a write-through and a write-back caching policy by reading metadata only while reading data and writing metadata only while writing data. My design performs robustly better than a current DRAM cache design under a wide range of memory access patterns including high miss rates and high write traffic.

Future research directions

My work on integrated GPUs reveals the promise of architectural support for usability of analytic queries as discussed above, which is one example new technology. However, programmer support for many emerging technologies such as other on-die accelerators, heterogeneous memory technologies (e.g., non-volatile memory and high-

bandwidth memory), and other heterogenous components like sensors is in its infancy. My future research will expand on my previous work and focus on simplifying programmers' interaction with other new hardware by providing architectural support for conventional programming models like cache coherence and a consistent address space.

I plan to continue searching for emerging technology that can be adapted to increase the efficiency of diverse applications, and I intend to investigate architectural innovations to improve the efficiency of these applications. There are many workloads not covered by traditional architecture benchmarks. Collaborating across disciplines with workload experts and increasing the performance and energy efficiency of these emerging workloads is an important research direction to overcome the slowing of Moore's Law.

Extending programmability to more accelerators

My previous work focused on making integrated GPUs a first-class system component because they are ubiquitous and popular accelerators. However, there are many other kinds of processors on-chip each with unique memory access patterns. Two examples of accelerators that are becoming ubiquitous and popular are image processors and low-power sensors. These devices provide a stark contrast to the high-performance GPUs I have previously studied, and will require novel techniques to solve the problems of coherent and consistent data access.

Image processors which transform the digital information collected by the image sensor are ubiquitous in mobile processors. They are increasingly programmable, which gives camera designers the flexibility to quickly implement new algorithms to increase image quality or enable new applications. However, the interface to these processors is currently rigid, and only the camera's designers modify the software stack. Simplifying the interface to the image processor opens the path for any developer to innovate. For instance, analyzing the sensor data at different stages of transformation instead of only after all transformations have been applied could increase the efficiency for many computer vision applications such as autonomous driving, augmented reality, and image compression. These innovations can be hastened by logically integrating the image processor with the rest of the system.

However, unlike CPUs and GPUs, image processors do not process data from memory, but from the image sensor. The memory access pattern from a programmable image processor will be unique. For instance, image processors may provide accelerated edge or motion detection that an application on the CPU or another accelerator consumes. To ease programming and increase flexibility, it is important this data is in a coherent and consistent address space.

Always-on sensors are another emerging pervasive accelerator. Currently, these sensors and their processors are used for simple applications like step counting and activity detection. However, opening these systems to all developers could significantly increase their usefulness. A compelling question for low-power sensors is how to efficiently transfer the control from the "always-on" ultra-low power device to another more computationally powerful processor in the system. In addition to transferring control, programmers will also want to transparently transfer their data into and out of these processors. The key difficulty is enabling coherence and consistency without resorting to traditional high-power mechanisms.

In situ architectural exploration

Not only is hardware technology changing rapidly, but the workloads that execute on this hardware are also evolving. A few examples include machine learning, augmented reality, big-data analytics, and intelligent personal assistants. These applications are end-to-end solutions, consisting of many interacting kernels of computation, and they cannot easily or accurately be represented as a single microbenchmark. Optimizing these applications requires changes across the entire hardware-software stack from new accelerators and emerging programmable processors to system integration and new programming interfaces. However, current architecture evaluation infrastructure is not easily adapted to studying end-to-end applications. Instead, I propose *in situ* simulation to study applications in their native execution environment. It is currently possible to use *in situ* simulation to study CPUs by leveraging ubiquitous virtualization technology. This virtualization can be extended to other accelerators, programmable processors, and even to novel devices via fast emulation (e.g., with FPGAs).

With this infrastructure, everyone in the computer architecture community will be able to investigate previously unaddressable problems. I believe in reducing the impediments to research, and I believe in releasing more than just papers to the community. I have released simulation models [5], the data I have generated and collected [10], and all information on how I created the data [11]. By being open with our research, we can push the boundaries of knowledge more quickly.

[1] Jason Power, Arkaprava Basu, Junli Gu, Sooraj Puthoor, Bradford M. Beckmann, Mark D. Hill, Steven K. Reinhardt, David A. Wood. “Heterogeneous System Coherence for Integrated CPU-GPU Systems.” The 46th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 46. Dec 2013.

[2] Jason Power, Mark D. Hill, David A. Wood. “Supporting x86-64 Address Translation for 100s of GPU Lanes.” The 20th IEEE International Symposium On High Performance Computer Architecture, HPCA 20. Feb 2014.

[3] Lena E. Olson, Jason Power, Mark D. Hill, David A. Wood. “Border Control: Sandboxing Accelerators.” The 48th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 48. Dec 2015.

[4] Jason Power**,** Joel Hestness, Marc S. Orr, Mark D. Hill, David A. Wood. “gem5-gpu: A Heterogeneous CPU-GPU Simulator.” Computer Architecture Letters vol. 14, no. 1. Jan-Jun 2015.

[5] https://gem5-gpu.cs.wisc.edu/

[6] Jason Power, Yinan Li, Mark D. Hill, Jignesh M. Patel, David A. Wood.” Toward GPUs being mainstream in analytic processing: An initial argument using simple scan-aggregate queries.” Proceedings of the Eleventh International Workshop on Data Management on New Hardware, DaMoN '15. Jun 2015.

[7] Jason Power, Yinan Li, Mark D. Hill, Jignesh M. Patel, David A. Wood. “Implications of Emerging 3D GPU Architecture on the Scan Primitive.” SIGMOD Record. Volume 44, Issue 1. Mar 2015.

[8] Jason Lowe-Power, Mark D. Hill, David A. Wood. “When to use 3D Die-Stacked Memory for Bandwidth-Constrained Big-Data Workloads.” The Seventh Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware (BPOE 7) at ASPLOS. Apr 2016.

[9] Jason Lowe-Power, Mark D. Hill, David A. Wood. “Adaptive Victim Cache: Reducing Access Amplification in a DRAM Cache.” Under review.

[10] https://research.cs.wisc.edu/multifacet/gpummu-hpca14/

[11] https://research.cs.wisc.edu/multifacet/bpoe16_3d_bandwidth_model/